# Machine Learning
### moodle https://dl1.cuni.cz/course/view.php?id=5765

Marta Vomlelová

marta@ktiml.mff.cuni.cz
http://ktiml.mff.cuni.cz/~marta
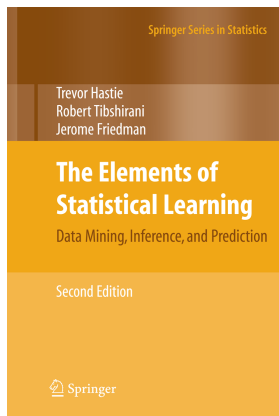
May 24, 2024

# Literature

- T. Hastie, R. Tishirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference and Prediction.* Springer Series in Statistics. Corrected 12th printing 2017 https://web.stanford.edu/ hastie/ElemStatLearn/printings/ESLII_print12_toc.pdf

- C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006

- Peter I. Frazier: A Tutorial on Bayesian Optimization, 2018

- Højsgaard, Søren, Edwards, David, Lauritzen, Steffen: Graphical Models with R, Springer 2012

- Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani: *An Introduction to Statistical Learning with Applications in R* (2013)

- S. Russel and P. Norwig. Artificial Intelligence: A Modern Approach. Prentice Hall, 2003.

# Exam

- Oral exam on topics covered by lectures.
- Most of it is covered by T. Hastie, R. Tishirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference and Prediction.* Springer Series in Statistics. Corrected 12th printing 2017 https://web.stanford.edu/ hastie/ElemStatLearn/printings/ESLII_print12_toc.pdf

# Table of Contens

# Statistical Decision Theory (for Regression)

- Let $X \in \mathbb{R}^p$ denote a real valued random input vector, and $Y \in \mathbb{R}$ a real valued random output variable, with joint distribution $P(X, Y)$.
- We seek a function $f(X)$ for prediction $Y$ given values of the input $X$.
- The theory requires a **loss function (chybovou funkci)** $L(Y, f(X))$ for penalizing errors in predictions.
- The far most common and convenient is **squared error loss (kvadratická chybová funkce)** $L(Y, f(X)) = (Y - f(X))^2$
- this leads us to a criterion for choosing $f$, the **expected (squared) prediction error (očekávanou chybu)** (EPE),

$$
\begin{aligned}
EPE(f) &= \mathbb{E}(Y - f(X))^2 \\
&= \int (y - f(x))^2 P(dx, dy)
\end{aligned}
$$

- by conditioning on $X$ we get

$$
EPE(f) = \mathbb{E}_X \mathbb{E}_{Y|X}([Y - f(X)]^2 | X)
$$

- and we see that it suffices to minimize EPE poinwise:

$$
f(x) = \operatorname{argmin}_c \mathbb{E}_{Y|X}([Y - c]^2 | X = x).
$$

- the solution is the conditional expectation also known as the **regression**

# $k$-NN and Conditional Expectation

- We seek the conditional expectation:

$$f(x) \;\; = \;\; \mathbb{E}(Y|X = x).$$

- Thus the best prediction of $Y$ at any point $X = x$ is the conditional mean, when the best is measured by the average squared error.
- Assume we have **a training set of data** $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$.
- The nearest neighbor methods attempt to directly implement this.
  - Since there are typically at most one observation at any point $x$, we settle for

$$\hat{f}(x) \;\; = \;\; mean(y_i | x \in N_k(x)),$$

  - where $mean$ denotes average, and $N_k(x)$ is the neighborhood containing $k$ points in $\mathcal{T}$ closest to $x$.
- Under mild regularity conditions on $P(X, Y)$ one can show as $k, N \to \infty$, such that $\frac{k}{N} \to 0$ then $\hat{f}(x) \to \mathbb{E}(Y|X = x)$.
- **The rate of convergence decreases as the dimension increases**. The problem is the speed of the convergence.

# Nearest-Neighbor Methods

- The **nearest-neighbor methods** use those observations in the training set $\mathcal{T}$ closest in the input space to $x$ to form $\hat{f}$.

$$\hat{f}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

- In classification, majority vote is used.

- Figures correspond to 15 nearest neighbor and 1 nearest neighbor respectively.

- Training error (usually) increases with increasing $k$.

### Effective number of parameters

The effective number of parameters of $k$ nearest neighbors is $N/k$ and is generally bigger than $p$ of the linear regression.



15-Nearest Neighbor Classifier
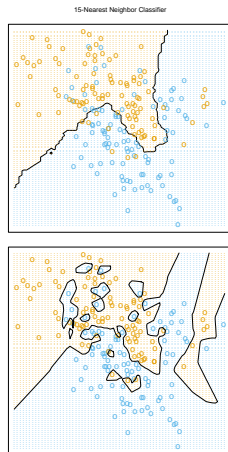


### Prediction complexity

'Naive' $O(Np)$.

# Overfitting

- Our goal is the minimal expected prediction error usually estimated by the error on the **test data** (orange).

- Usually, **overfitting** appears for complex models - an increase of the test error despite the decrease of the training error.

- This is the reason for other models then nearest neighbor model.

- Possible improvements:
  - Kernel methods
  - different weights for dimensions
  - local regression fits
  - linear models fit to a basis expansion
  - sums of non-linearly transformed linear models.
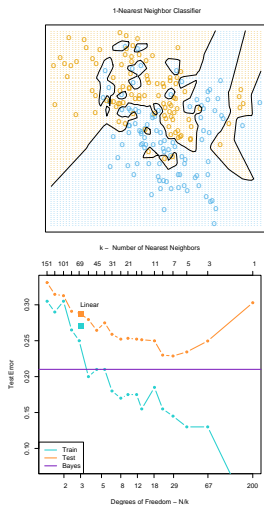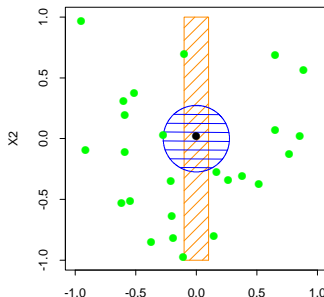
# Curse of Dimensionality demonstration

Prediction

- Assume $x_i$ uniformly generated form the interval $\langle -1, 1 \rangle^p$
- We have $Y = f(X) = e^{-8\|X\|^2}$, without any noise, for $x_i$ we know exactly $f(x_i)$.
- We use 1–NN to estimate $f(0)$ based on 1000 data sample.
- Predicted value for $x = \langle 0, \ldots, 0 \rangle$ is lower that 1 and in high dimensions $p$ it goes to 0.
- Increasing $k$ in $k$–NN does not help here.
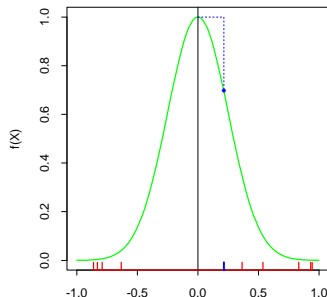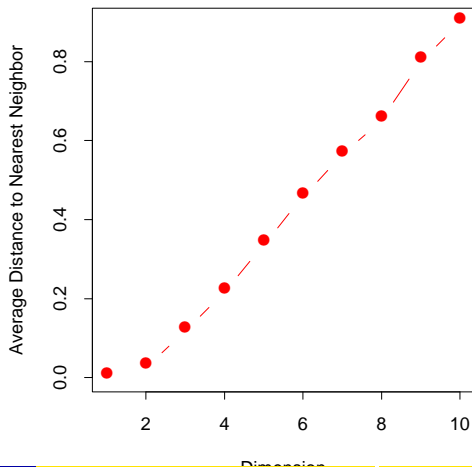
# Empirical Nearest Neighbor Distance

- Assume $x_i$ uniformly generated form the interval $\langle -1, 1 \rangle^p$
- We use 1–NN to estimate $f(0)$ based on 1000 data sample.

Distance to 1-NN vs. Dimension

# Curse of dimensionality

Most points are close to the border

- Consider $N$ instances uniformly distributed in a $p$–dimensional unit ball.
- Median distance of the nearest neighbor from the center is:

$$d(p, N) = \left(1 - \frac{1}{2}^{\frac{1}{N}}\right)^{\frac{1}{p}}$$

- The formula: 1 point inside: $\frac{d^p}{1^p}$, outside: $(1 - d^p)$, $N$ outside $(1 - d^p)^N = \frac{1}{2}$.
- For $N = 500$, $p = 10$, we get $d(p, N) \approx 0.52$, that is more than a half way to the border.
- For $N = 10^6$, $p = 200$, we get $d(p, N) \approx 0.93$.
- Close to the border, we must **extrapolate**, what is more difficult than interpolation.

# Training data (and their notation)

We have

- a set of random variables (features) $X_1, \ldots, X_p$
- numerical goal variable $Y$ (for regression)
- training data $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

|  |  |  |  | Goal attribute |
|---|---|---|---|---|
| $X^T$ = vector | $\langle X_1$ | $X_j$ | $X_p \rangle$ | Y or G |
| $\mathbf{x}_1^T$ |  |  |  |  |
|  |  |  |  |  |
| $\mathbf{x}_i^T$ = vector | $\langle x_1$ | $x_j$ | $x_p \rangle$ | y or g |
|  |  |  |  |  |
| $\mathbf{x}_N^T$ |  |  |  |  |

- $x$ and $\mathbf{x}_i$ are $p$-dimensional column vectors
- $\mathbf{X}$ is the $N \times p$ matrix
- $\mathbf{x}_j$ is the $N$ vector consisting of all observations on variable $X_j$.
- $\mathbf{y} = (y_1, \ldots, y_N)^T$ denotes the vector of training goal data.

# Linear regression

- Given a vector of inputs $X^T = (X_1, \ldots, X_p)$ we predict the output $Y$ via the model $f_\beta$, $\beta \in \mathbb{R}^{p+1}$

$$\hat{Y} = \hat{f}_\beta(X) = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

- $\hat{\beta}_0$ is the **intercept, bias**, **(průsečík)**.
- We include the constant variable 1 to $X$, include $\hat{\beta}_0$ in $\hat{\beta}$ to get the model in vector form as an inner product

$$\hat{Y} = \sum_{j=0}^{p} X_j \hat{\beta}_j = \mathbf{X^T} \widehat{\beta}$$

- The sum $\sum_{j=0}^{p} X_j \hat{\beta}_j$ can be written as $\mathbf{X^T} \widehat{\beta}$.

# Linear regression from the data

- Let $i$ range over the data samples, $\mathbf{X}$ be an $N \times p$ data matrix, $\mathbf{y}$ is a column vector of the goal variable. We can write:

$$\hat{\mathbf{y}} = \mathbf{X}\beta$$

- We search optimal $\hat{\beta}$ to minimize the residual sum squares RSS:

$$RSS(\beta) = \sum_{i=1}^{N}(y_i - \mathbf{x}_i^T \beta)^2 = (\mathbf{y} - \mathbf{X}\beta)^{\mathsf{T}}(\mathbf{y} - \mathbf{X}\beta) \tag{1}$$

- Differentiating w.r.t. $\beta$ we get *normal equations*

$$\mathbf{X}^{\mathsf{T}}(\mathbf{y} - \mathbf{X}\beta) = \mathbf{0}$$

- If $\mathbf{X}^{\mathsf{T}}\mathbf{X}$ is not singular, then the unique solution is given by

$$\hat{\beta} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y} \tag{2}$$

- For a given $x_i$ the estimate $\hat{y}_i$ is $\hat{y}_i = \hat{y}(x_i) = x_i^T \hat{\beta}$.
- From a singular $\mathbf{X}^{\mathsf{T}}\mathbf{X}$ we should remove dependent features or filter the data to make it invertible.

# Linear Regression

- Let us have a data $N = 6, p = 2$ (Fatt11, Meat11), 1 column is for $\beta_0$, does not count:

$$\mathbf{X} = \begin{bmatrix} 1 & Fat11 & Meat11 \\ 1 & 17 & 51 \\ 1 & 17 & 49 \\ 1 & 14 & 38 \\ 1 & 17 & 58 \\ 1 & 14 & 51 \\ 1 & 20 & 40 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} LeanMeat \\ 56.5 \\ 57.6 \\ 55.9 \\ 61.8 \\ 63.0 \\ 54.6 \end{bmatrix}$$

- We are searching parameters $\beta = (\beta_0, \beta_1, \beta_2)^T$ to minimize:

$$
\begin{aligned}
RSS(\beta, \mathbf{X}, \mathbf{y}) &= \sum_{i=1}^{N}(y_i - \mathbf{x}_i^T \beta)^2 = (\mathbf{y} - \mathbf{X}\beta)^\mathbf{T}(\mathbf{y} - \mathbf{X}\beta) \\
&= (56.5 - (1 * \beta_0 + 17 * \beta_1 + 51 * \beta_2))^2 + \ldots \\
&\quad \ldots + (54.6 - (1 * \beta_0 + 20 * \beta_1 + 40 * \beta_2))^2
\end{aligned}
$$

# Linear regression from the data

- If $\mathbf{X^T X}$ is not singular, then the unique solution is given by

$$\hat{\beta} = (\mathbf{X^T X})^{-1} \mathbf{X^T y}$$

$$\mathbf{X^T} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 17 & 17 & 14 & 17 & 14 & 20 \\ 51 & 49 & 38 & 58 & 51 & 40 \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} 1 & 17 & 51 \\ 1 & 17 & 49 \\ 1 & 14 & 38 \\ 1 & 17 & 58 \\ 1 & 14 & 51 \\ 1 & 20 & 40 \end{bmatrix}$$

$$\mathbf{X^T X} = \begin{bmatrix} 6 & 99 & 287 \\ 99 & 1659 & 4732 \\ 287 & 4732 & 14011 \end{bmatrix}$$

$$(\mathbf{X^T X})(\mathbf{X^T X})^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$(\mathbf{X^T X})^{-1} = \begin{bmatrix} 19.7320 & -0.6714120 & -0.1774305 \\ -0.6714 & 0.0392824 & 0.0004861 \\ -0.1774 & 0.0004861 & 0.0035416 \end{bmatrix}$$

# Linear regression from the data

$$\hat{\beta} = (\mathbf{X^T X})^{-1} \mathbf{X^T y}$$

$$\mathbf{X^T} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 17 & 17 & 14 & 17 & 14 & 20 \\ 51 & 49 & 38 & 58 & 51 & 40 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 56.5 \\ 57.6 \\ 55.9 \\ 61.8 \\ 63.0 \\ 54.6 \end{bmatrix}$$

$$\mathbf{X^T y} = \begin{bmatrix} 349.3498 \\ 5746.1340 \\ 16807.4663 \end{bmatrix}$$

$$\hat{\beta} = \begin{bmatrix} 19.7320 & -0.6714120 & -0.1774305 \\ -0.6714 & 0.0392824 & 0.0004861 \\ -0.1774 & 0.0004861 & 0.0035416 \end{bmatrix} \begin{bmatrix} 349.3498 \\ 5746.1340 \\ 16807.4663 \end{bmatrix} = \begin{bmatrix} 53.2097294 \\ -0.6653895 \\ 0.3343728 \end{bmatrix} \begin{matrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{matrix}$$

# Prediction

- Linear regression predicts:

$$\hat{y} = \hat{f}(\mathbf{x}) = \mathbf{x}^\mathsf{T}\widehat{\beta}$$

- Prediction for training data:

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{X}) = \mathbf{X}\widehat{\beta} = \mathbf{X}(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y} = \begin{bmatrix} 58.95112 \\ 58.28237 \\ 56.60044 \\ 61.29173 \\ 60.94729 \\ 53.27685 \end{bmatrix}$$

- **The hat matrix** $H = \mathbf{X}(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}$ transforms $\mathbf{y}$ to $\hat{\mathbf{y}}$.

$$H = \begin{bmatrix} 0.21 & 0.19 & 0.00 & 0.29 & 0.15 & 0.15 \\ 0.19 & 0.18 & 0.07 & 0.22 & 0.13 & 0.20 \\ 0.00 & 0.07 & 0.78 & -0.25 & 0.31 & 0.09 \\ 0.29 & 0.22 & -0.25 & 0.55 & 0.22 & -0.03 \\ 0.15 & 0.13 & 0.31 & 0.22 & 0.44 & -0.25 \\ 0.15 & 0.20 & 0.09 & -0.03 & -0.25 & 0.84 \end{bmatrix}$$

- You may notice that $trace(H) = sum(diag(H)) = 3$.

# Residual Sum of Squares

- Residual Sum of Squares is

$$
\begin{aligned}
RSS(\beta, \mathbf{X}, \mathbf{y}) &= \sum_{i=1}^{N}(y_i - \mathbf{x}_i^T \beta)^2 = (\mathbf{y} - \hat{\mathbf{y}})^T(\mathbf{y} - \hat{\mathbf{y}}) \\
&= \left( \begin{bmatrix} 56.5 \\ 57.6 \\ 55.9 \\ 61.8 \\ 63.0 \\ 54.6 \end{bmatrix} - \begin{bmatrix} 58.95112 \\ 58.28237 \\ 56.60044 \\ 61.29173 \\ 60.94729 \\ 53.27685 \end{bmatrix} \right)^T \left( \begin{bmatrix} 56.5 \\ 57.6 \\ 55.9 \\ 61.8 \\ 63.0 \\ 54.6 \end{bmatrix} - \begin{bmatrix} 58.95112 \\ 58.28237 \\ 56.60044 \\ 61.29173 \\ 60.94729 \\ 53.27685 \end{bmatrix} \right) \\
&= \left( \begin{bmatrix} -2.4263727 \\ -0.7027914 \\ -0.7105023 \\ 0.5254632 \\ 2.0123528 \\ 1.3018505 \end{bmatrix} \right)^T \left( \begin{bmatrix} -2.4263727 \\ -0.7027914 \\ -0.7105023 \\ 0.5254632 \\ 2.0123528 \\ 1.3018505 \end{bmatrix} \right) \\
&= 12.9065
\end{aligned}
$$

# Linear regression complexity

$$\hat{\beta} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$$

### Linear Regression Complexity

Training complexity

- The complexity of the direct approach to linear regression is $O(p^2 N + p^3)$.
- $\mathbf{X}^\mathsf{T}\mathbf{X}$ is $O(p^2 N)$
- the result is $p \times p$ matrix,
- its inversion takes $O(p^3)$.

Cholevsky decomposition

- $O(p^3 + \frac{p^2}{2} N)$

QR decomposition

- $O(p^2 N)$

Prediction complexity

- To calculate $\beta^T x$ takes $O(p)$.

# Improving Least Square Estimate

- Reasons
  - improve prediction accuracy (decrease variance)
  - improve interpret ability
- methods
  - Best Subset selection
  - **Forward**- and Backward-**Stepwise Selection**
  - Forward-Stagewise Regression
    - as Forward-Stepwise
    - do **not** change previous coefficients
    - slow convergence
    - may be useful in high dimension $p$!
  - Penalized methods.
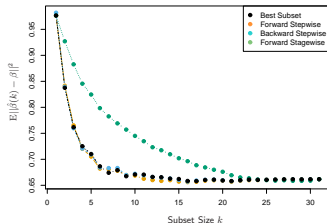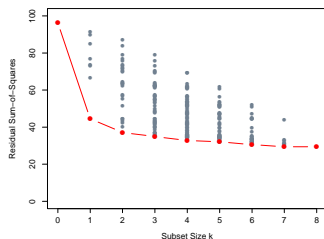
# Centering, Standardization

**Definition (Centering, Standardization)**

- To **center** the variables replace each feature to have zero mean,

$$\mathbf{x}_j \quad \leftarrow \quad \mathbf{x}_j - \overline{x_j}$$

- The sample **variance** of a variable $\mathbf{x}_j$ is defined,

$$s_j^2 \quad \leftarrow \quad \frac{1}{N} \sum_{i=1}^{N} (x_{ij} - \overline{x}_j)^2$$

Both my sources use $N$. I know about $N-1$ used in statistics.

- **Standardization** performs the centering and divides features by their standard deviation,

$$\mathbf{x_j} \quad \leftarrow \quad \frac{\mathbf{x}_j - \overline{x_j}}{s_j}.$$

# Sample Covariance, Correlation

## Definition (Sample Covariance, Correlation)

- The **sample covariance** is a $p \times p$ symmetric matrix

$$S = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{x})(\mathbf{x}_i - \bar{x})^T$$

- with elements

$$s_{j,k} = \frac{1}{N} \sum_{i=1}^{N} (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)$$

- The **sample correlation** of the columns $\mathbf{x}_j, \mathbf{x}_k$ is

$$\rho_{j,k} \leftarrow corr(\mathbf{x}_j, \mathbf{x}_k) \leftarrow \frac{s_{j,k}}{s_j s_k} = \frac{\frac{1}{N} \sum_{i=1}^{N} (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)}{\sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_{ij} - \bar{x}_j)^2} \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_{ik} - \bar{x}_k)^2}}$$

- For standardized features, the correlation is just $\frac{\mathbf{x}_j^T \mathbf{x}_k}{N}$.

# Penalized Methods

$$\hat{\beta} = argmin_{\beta}\left(\sum_{i=1}^{N}(y_i - \beta_0 - \mathbf{x}_i^T\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j|^q\right)$$

- We add the complexity penalty $\lambda\sum_{j=1}^{p}|\beta_j|^q$ to the RSS.
- **Ridge regression** $q = 2$
- **Lasso regression** $q = 1$
- **Elastic net** penalty $\lambda\sum_{j=1}^{p}(\alpha|\beta_j|^2 + (1-\alpha)|\beta_j|)$
  - a compromise between ridge and lasso
  - selects variable like the lasso, and shirks together the coefficients of correlated predictors like ridge.
  - It also has considerable computational advantage over the $L_q$ penalties.





```
from sklearn import linear_model
linear_model.BayesianRidge()
```
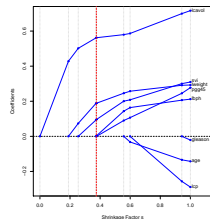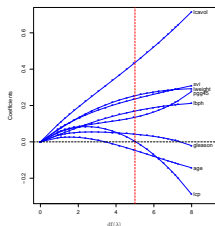
# Rigde Regression

$$\hat{\beta}^{ridge} \;=\; argmin_\beta \left( \sum_{i=1}^{N}(y_i - \beta_0 - \mathbf{x}_i^T \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j|^2 \right)$$

- The solution is

$$\mathbf{X} \;\leftarrow\; \text{centered } (N \times p) \text{ input matrix}$$

$$\hat{\beta}_0 \;=\; \frac{1}{N} \sum_{i=1}^{N} y_i$$

$$\hat{\beta} \;=\; (\mathbf{X}^T\mathbf{X} - \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}.$$
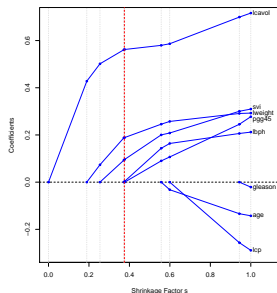
sklearn.linear_model.Ridge
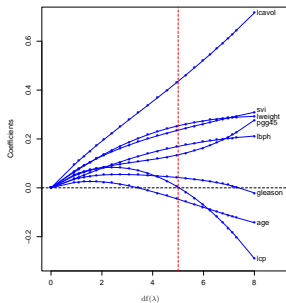
# Lasso Regression

$$\hat{\beta}^{lasso} \;\; = \;\; argmin_\beta \left( \sum_{i=1}^{N} (y_i - \beta_0 - \mathbf{x}_i^T \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right)$$

- Solved by a quadratic programming algorithm

  sklearn.linear_model.Lasso

- or LARS modification, that calculates full Lasso path in in $O(p^2 N + p^3)$.

- we use LARS on standardized data.

  sklearn.linear_model.Lars

# LARS Idea

- For active set of parameters $\mathcal{A}_k$ the parameters $\beta_{\mathcal{A}_k}$
- consider current **residuals** $\mathbf{r}_k = \mathbf{y} - \mathbf{X}_{\mathcal{A}_k}\beta_{\mathcal{A}_k}$
- and the correlation of each predictor with the residuals $\langle \mathbf{x}_j, \mathbf{r}_k \rangle$
- the correlations are equal for the predictors in the active set $\mathcal{A}_k$
- we change $\beta_{\mathcal{A}_k} \leftarrow \beta_{\mathcal{A}_k} + \alpha \delta_k$ in the direction

$$\delta_k = (\mathbf{X}_{\mathcal{A}_k}^{\mathsf{T}}\mathbf{X}_{\mathcal{A}_k})^{-1}\mathbf{X}_{\mathcal{A}_k}^{\mathsf{T}}\mathbf{r}$$

- and the correlation $X_{\mathcal{A}_k}$ with residuals decreases.
- Correlations of other features change linearly and we can calculate next intersection point.
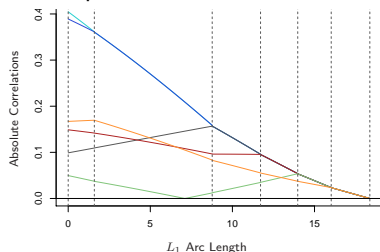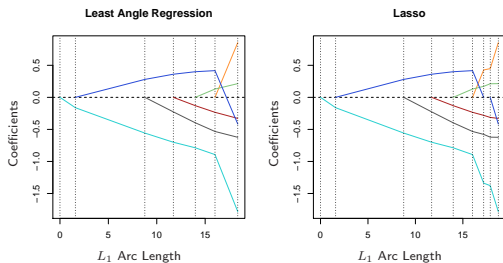
# LARS Least Angle Regression

- democratic version of forward stepwise regression
- provides an extremely efficient algorithm for computing the entire lasso path.

---

**Least Angle Regression**

1: **procedure** Least Angle Regression:($\mathbf{X}$, $\mathbf{y}$)
2:     $\beta_1, \ldots, \beta_p \leftarrow 0$ initialize
3:     $r \leftarrow y - \bar{y}$ residuals
4:     find the predictor $x_j$ most correlated with $r$
5:     Move $\beta_j$ from 0 towards its least-squares coefficient $\langle x_j, r \rangle$ until some other competitor $x_k$ has as much correlation with the current residual as does $x_j$.
6:     $\mathcal{A}_1 \leftarrow \{x_j\}$ active coefficients
7:     **for** $k = 2, \ldots, min(N-1, p)$ **do**
8:         move current set of $\beta_{\mathcal{A}_k}$ by their joint least squares coefficient of the current residual until some other competitor $x_\ell$ catches up.
9:         $\mathcal{A}_k \leftarrow \mathcal{A}_{k-1} \cup \{x_\ell\}$ active coefficients
10:     **end for**
11: **end procedure**

Least Angle Regression

Lasso

## Lasso Modification of the Least Angle Regression

8a: If a non-zero coefficient hits zero, drop its variable from the active set of variables and recompute the current joint least squares direction.

## Complexity of LARS

- LARS requires the same order of computation as that of a single least squares fit using the $p$ predictors.
- hidden in the $p$'s in $O(p^2 N + p^3)$ or Cholevsky decomposition.

# Model Complexity

## Effective Degrees of Freedom

- Linear regression $p$
- Ridge regression $df(\hat{y}) = tr(\mathbf{X}(\mathbf{X^T X} - \lambda \mathbf{I})^{-1}\mathbf{X^T})$.
- LARS: after $k$ steps, $df(\hat{y}) = k$.
- LASSO: roughly the number of predictors in the model (may take more, some predictors drop out.

# Summary

- Introduction
    - classification and regression,
    - training data,
    - RSS,
    - expected prediction error,
    - overfitting,
    - effective number of parameters,
    - curse of dimensionality,
- $k$ Nearest neighbor model,
- Linear regression and its modifications
    - Best subset
    - Ridge, BayesianRidge
    - Lasso
    - LARS.

# Pathwise Coordinate Optimization

- LARS modification, iteratively by the coordinates
- fix the penalty parameter $\lambda$
- optimize successively over each parameter, holding the other parameters fixed at their current values.
- Assume the predictors are all standardized to have mean zero and unit norm,
- $\tilde{\beta}_k(\lambda)$ the current estimate for $\beta_k$ at penalty parameter $\lambda$

$$R(\tilde{\beta}(\lambda), \beta_j) = \frac{1}{2} \sum_{i=1}^{N} \left( y_i - \sum_{k \neq j} x_{ik} \tilde{\beta}_k(\lambda) - x_{ij} \beta_j \right)^2 + \lambda \sum_{k \neq j} |\tilde{\beta}_k(\lambda)| + \lambda |\beta_j|$$

- this can be viewed as a univariate lasso problem with response variable the partial residual

$$y_i - \tilde{y}_i^{(j)} = y_i - \sum_{k \neq j} x_{ik} \tilde{\beta}_k(\lambda).$$
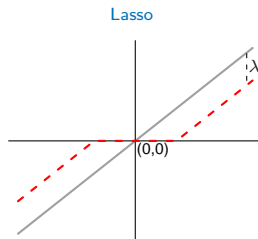
# Pathwise Coordinate Optimization

- this has an explicit solution, resulting in the update

$$\widetilde{\beta}_j(\lambda) \leftarrow S\left(\sum_{i=i}^{N} x_{ij}(y_i - \tilde{y}_i^{(j)}), \lambda\right)$$

- where $S$ is the soft-thresholding operator

$$S(t, \lambda) = sign(t)(|t| - \lambda)_+ \qquad (3)$$

- Estimators of $\beta_j$ in case of orthonormal columns of $X$.



Lasso

# Grouped Lasso - not presented this year

- Dummy variables for representing the levels of a categorical predictor.
- Genes that belong to the same biological pathway.
- Suppose that the $p$ predictors are divided into $L$ groups
  - with $p_\ell$ the number in group $\ell$.
  - a matrix $X_\ell$ represents the predictors corresponding to the $\ell$th group
  - with corresponding coefficient vector $\beta_\ell$.
- the grouped-lasso minimizes the convex criterion

$$min_{\beta \in \mathbb{R}^p} \left( ||y - \beta_0 1 - \sum_{\ell=1}^{L} X_\ell \beta_\ell||_2^2 + \lambda \sum_{\ell=1}^{L} \sqrt{p_\ell} ||\beta_\ell||_2 \right) \qquad (4)$$

- $|| \cdot ||_2$ is the Euclidean norm (not squared)
- $\sqrt{p_\ell}$ accounts for the varying group sizes.

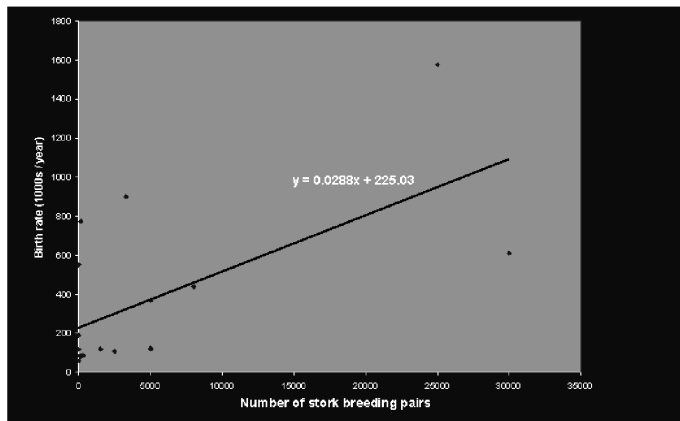# Example – Storch brings babies in Europa



**Fig 1.** How the number of human births varies with stork populations in 17 European countries.

# Linear methods for classification

- We are given two features $X_1, X_2$ and the goal _BLUE_ or _ORANGE_.

- Later, we will see better ways. For now, we encode _BLUE_ = 0 a _ORANGE_ = 1, and find a linear regression model.

- The fitted values $\hat{Y}$ are converted to a fitted class variable $\hat{G}$ as follows:
$$\hat{G} = \left\{ \begin{array}{l} \underline{BLUE} \text{ for } Y \leq 0.5 \\ \underline{ORANGE} \text{ for } Y > 0.5 \end{array} \right\}$$

- The hyperplane $\{x : x^T \beta = 0.5\}$ is called the **decision boundary (rozhodovací hranice)**.

- Better to use **logistic regression**, that gives also a linear decision boundary.
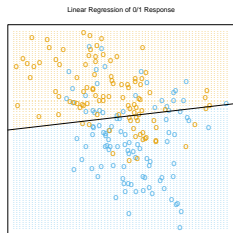


Linear Regression of 0/1 Response

**FIGURE 2.1.** *A classification example in two dimensions. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then fit by linear regression. The line is the decision boundary defined by $x^T \hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.*

# Two Scenarios

- The training data in each class were generated from bivariate Gaussian distribution with uncorrelated components and different means.
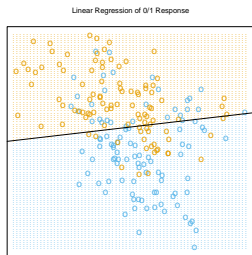
- The linear model is (almost) optimal.



**FIGURE 2.1.** *A classification example in two dimensions. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then fit by linear regression. The line is the decision boundary defined by $x^T \hat\beta = 0.5$. The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.*

- The training data in each class came from a mixture of 10 low-variance Gaussian distributions, with individual means themselves distributed as Gaussians.
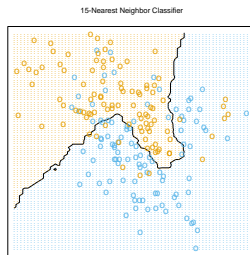
- The linear model is **not** optimal.



**FIGURE 2.2.** *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1) and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

# Basis expansion and regularization, Splines

Linear and logistic regression assume linear function of $X$.

- Regression: We estimate $f(X) = \mathbb{E}(Y|X)$
- Classification: We estimate $f(X) = log\frac{P(Y=1|X)}{P(Y=0|X)}$.

**Linear basis expansion** in $X$

- we replace the vector of inputs $X$ with additional variables $h_m$,
- $h_m(X) : \mathbb{R}^p \to \mathbb{R}$, $m = 1, \ldots, M$.

$$f(x) = \sum_{m=1}^{M} \beta_m h_m(X).$$

- 'the only change' is a different matrix of the features $X$, further fit is the same.
- Usually, we search $f_j(X_j)$ for each dimension by a backfitting algorithm in a **generalized additive model (GAM)**

$$\mathbb{E}(Y|X_1, \ldots, X_p) = \alpha + f_1(X_1) + \ldots + f_p(X_p)$$

- where $f_j$'s are unspecified smooth functions
- $X_j$ predictors, $Y$ the outcome.

- For now, we consider one-dimensional feature $X$.

# Simple derived features

- We fit the model:

$$f(x) = \sum_{m=1}^{M} \beta_m h_m(X).$$

- $h_m(X) = X_m$, $m = 1, \ldots, M$ recovers the original linear model.
- $h_m(X) = X_j^2$ or $h_m(X) = X_j X_k$ polynomial terms to achieve higher-order Taylor expansions.
  - ! The number of variables grows exponentially in the degreee of the polynomial.
- $h_m(X) = log(X_j)$, $\sqrt{X_j}$, $||X||, \ldots$, other nonlinear transformations.
- $h_m(X) = I(L_m \leq X_k < U_m)$, an indicator for a region of $X_k$.
  - piecewise constant contribution for $X_K$.
  - With non-overlapping regions used in regression trees.
- $h_m(X) = max((X_j - \xi_k)^3, 0)$ piecewise-polynomial spline basis
- wavelet bases.

# Piecewise Polynomials and Splines

- A **piecewise polynomial** function $f(X)$ is obtained by
  - division the domain of $X$ into continuous intervals by the knots $\xi_1, \ldots, \xi_{M-1}$
  - and representing $f$ by a separate polynomial in each interval.
  - Examples:
    - Three basis functions:
      $h_1(X) = I(X < \xi_1)$, $h_2(X) = I(\xi_1 \leq X < \xi_2)$, $h_3(X) = I(\xi_2 \leq X)$.
    - Additional linear functions:
      $h_{m+3} = h_m(X) \cdot X$, $m = 1, \ldots, 3$.
    - Additional cubic functions:
      $h_{m+6} = h_m(X) \cdot X^2$, $h_{m+9} = h_m(X) \cdot X^3$, $m = 1, \ldots, 3$.
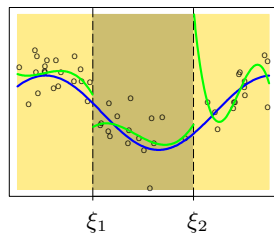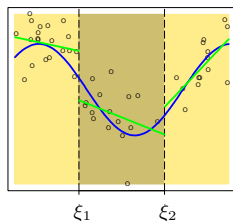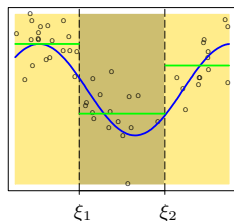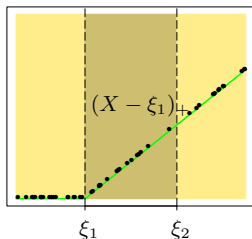
# Continuous functions

- We add the continuity restriction: the value in $\xi_j$ is the unique.
- Continuous piecewise linear basis:
  $h_1(X) = 1$, $h_2(X) = X$, $h_3(X) = (X - \xi_1)_+$, $h_4(X) = (X - \xi_2)_+$.
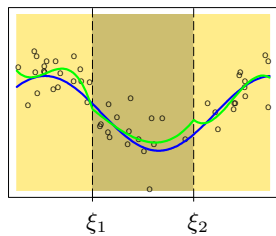- We have spared two parameters for two continuity conditions.
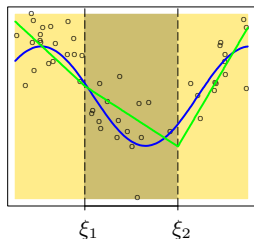


Continuous Piecewise Linear     Piecewise-linear Basis Function     Continuous

$(X - \xi_1)_+$

- For the cubic fit, the figure looks ugly, we need continous first and second derivative.

# Cubic spline

- **Cubic spline** is a piecewise cubic fit with continuous first and second derivatives at the knots $\xi_i$.

- The basis functions with knots $\xi_1, \xi_2$ are:
  $h_1(X) = 1$,
  $h_2(X) = X$,
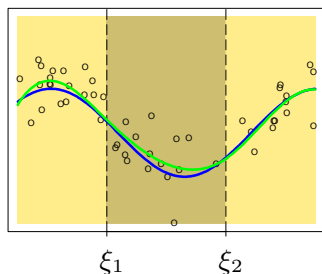  $h_3(X) = X^2$,
  $h_4(X) = X^3$,
  $h_5(X) = (X - \xi_1)_+^3$,
  $h_6(X) = (X - \xi_2)_+^3$.



Continuous Second Derivative

$\xi_1$      $\xi_2$

- Parameter count:
  (3 regions)x(4 pars per region)-(2 knots)x(3 constraints per knot)=6.

# Order-M splines

- Cubic spline is an order-4 spline.
- Generally, order-M spline with knots $\xi_j$, $j = 1, \ldots, K$ is a piecewise-polynomial of order $(M-1)$ and has continuous derivatives to order $(M-2)$.
- General truncated basis functions are:
  - $h_j(X) = X^{j-1}$, $j = 1, \ldots, M$,
  - $h_{M+\ell} = (X - \xi_\ell)_+^{M-1}$, $\ell = 1, \ldots, K$.
- **Regression splines**
  - splines with fixed knots
  - usually at percentiles of the data $X$.
  - the number of knots is specified by the degree an the degrees of freedom $(df - M)$. $h_0$ does not count.
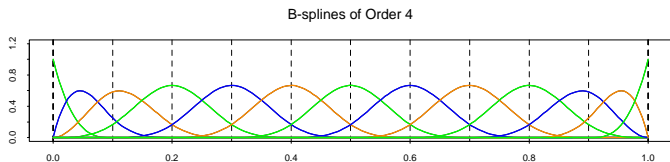
# B-splines

**B-splines** use other basis describing the same linear feature space.

- $\{h_i\}$ is a basis of a linear space of functions
- we may choose a different base to cover the same space of functions.
- B-splines are more stable numerically, useful for large number of knots $K$.
- B-splines have quite difficult recursive formula (not needed for the exam).

$$
\begin{aligned}
B_{i,1}(x) &= \begin{cases} 1 & \text{if } \xi_i \leq x \leq \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \\
B_{i,k+1}(x) &= \omega_{i,k}(x)B_{i,k}(x) + [1 - \omega_{i+1,k}(x)]B_{i+1,k}(x) \\
\omega_{i,k}(x) &= \begin{cases} \frac{x - \xi_i}{\xi_{i+k} - \xi_i} & \text{if } \xi_{i+k} \neq \xi_i \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$



B-splines of Order 4

# Computational complexity

## Spline fit time complexity

- (Standard) regression splines
    - $N$ observations, $K + M$ variables (basis functions) take $O(N(K + M)^2 + (K + M)^3)$.
- B-splines
    - sort values of $X$
    - Cubic B splines have local support, B is lower 4–banded.
    - order $(M + 1)$ B splines have local support, B is lower $(M + 1)$-banded.
    - Cholesky decomposition $B = LL^T$ can be computed easily.
    - Solution of $\hat{f}$ is in $O(N(M + 1))$ operations.

B-splines implemented in scipy and statmodels.

# Natural Cubic Spline

- Polynomial fit tends to be erratic near the boundaries.
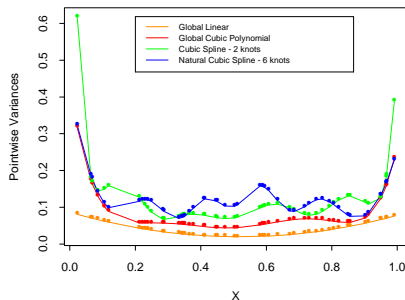


- **Natural cubic spline** is a spline that the function is linear beyond the boundary knots.
- Basis functions $N_i$, $i = 1, \ldots, K$:
  $N_1(X) = 1, N_2(X) = X, N_{k+2}(X) = d_k(X) - d_{K-1}(X)$ for

$$d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}.$$

# Smoothing Splines

- Maximal number of knots: $N$, the number of examples.
- But, we need a penalty for model complexity.

$$RSS(f, \lambda) = \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int (f''(t))^2 dt$$

- $\lambda$ is **smoothing parameter**
  - $\lambda = 0$: can be any function that interpolates the data.
  - $\lambda = \infty$: the simple least squares line fit, no nonzero second derivative is tolerated.
- Has a unique finite-dimensional minimizer, a natural cubic spline with knots at the unique values of the $x_i$, $i = 1, \ldots, N$.
- The solution is in the form: $f(x) = \sum_{j=1}^{N} N_j(x)\theta_j$.
- The criterion reduces for:

$$RSS(\theta, \lambda) = (\mathbf{y} - \mathbf{N}\theta)^T (\mathbf{y} - \mathbf{N}\theta) + \lambda \theta^T \Omega_N \theta$$

- where $\{\mathbf{N}\}_{ij} = N_j(x_i)$ and $\{\Omega\}_{jk} = \int N_j''(t)N_k''(t)dt$.

# $\Omega$

let $a = x_1 = 0$, $b = x_{101} = 1$, and knots $\xi_l = x_{l+1}$ for $l = 1, \ldots, K$ and $K = 99$. Also, the basis functions for a cubic spline $M = 4$ are

$$
\begin{aligned}
h_j(x) &= x^{j-1} & j = 1, \ldots, M, \\
h_{M+l}(x) &= (x - \xi_l)_+^{M-1} & l = 1, \ldots, K.
\end{aligned}
$$

Then, $\boldsymbol{H} = (h_j(x_i))_{N, M+K}$ where $h_j(x_i)$ is for the i-th row and the j-th column. Let $\boldsymbol{\Omega} = (\omega_{i,j})_{M+K, M+K}$ be a symmetric matrix and the upper triangular $\omega_{i,j} = \int_a^b h_i''(t) h_j''(t) dt$ is

$$
\begin{aligned}
\omega_{i,j} &= 0 & \text{for } i < M, \\
\omega_{M,j} &= \tfrac{1}{3} b^3 - \tfrac{1}{2} b^2 \xi_j + \tfrac{1}{6} \xi_j^3 & \text{for } j > M, and \\
\omega_{i,j} &= \tfrac{1}{3}(b^3 - \xi_0^3) - \tfrac{1}{2}(b^2 - \xi_0^2)(\xi_{i-M} + \xi_{j-M}) + (b - \xi_0)\xi_{i-M}\xi_{j-M} & \text{for } j \geq i > M,
\end{aligned}
$$

where $\xi_0 = \max\{\xi_{i-M}, \xi_{j-M}\}$.

https://vardeman.public.iastate.edu/stat602/602x_hw4_sol.pdf

# Smoothing Splines solution

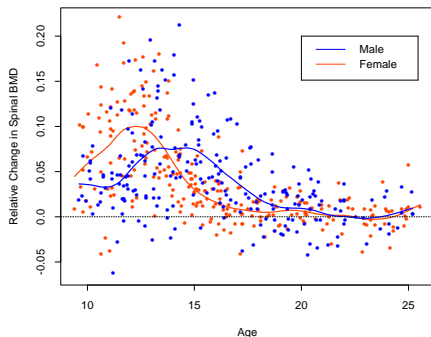- Smoothing spline solution is a generalized ridge regression

$$\hat{\theta} = (\mathbf{N}^T \mathbf{N} + \lambda \Omega_N)^{-1} \mathbf{N}^T \mathbf{y}$$

- The fitted smoothing spline is given by:

$$\hat{f}(x) = \sum_{j=1}^{N} N_j(x) \hat{\theta}_j$$

### Example

- Bone mineral density (BMD) in adolescents.

- Response: the change in BMD over two consecutive visits, typically about one year apart.

- coded by gender, females precedes growth spurt about two years.

- $\lambda \approx 0.00022$, $df_\lambda = 12$.

# Degrees of Freedom and Smoother Matrices

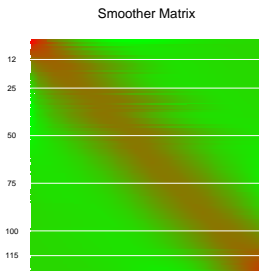- Smoothing spline is a linear smoother:

$$
\begin{aligned}
\hat{f} &= \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\Omega_N)^{-1}\mathbf{N}^T\mathbf{y} \\
&= \mathbf{S}_\lambda\mathbf{y}
\end{aligned}
$$

- $\mathbf{S}_\lambda$ is known as smoother matrix.
- $df_\lambda = trace(\mathbf{S}_\lambda)$
  - the sum of the diagonal elements
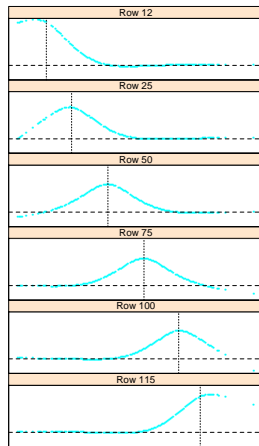  - $\lambda \approx 0.00022$ derived numerically by solving $trace(\mathbf{S}_\lambda) = 12$.

smoothing splines only in R

# Smoother Matrix

- rows $\mathbf{S}_\lambda$ ordered with $x$
- right: selected rows
- $\lambda \to 0$ means $df_\lambda \to N$ and $\mathbf{S}_\lambda \to \mathbf{I}$
- $\lambda \to \infty$ means $df_\lambda \to 2$ and $\mathbf{S}_\lambda \to \mathbf{H}$, the hat matrix for linear regression on $\mathbf{x}$.
- $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ since $(\hat{y} = Hy)$



Smoother Matrix

Equivalent Kernels

Row 12

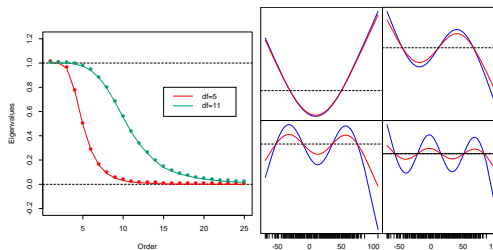Row 25

Row 50

Row 75

Row 100

Row 115

# Pollution data example

## Example

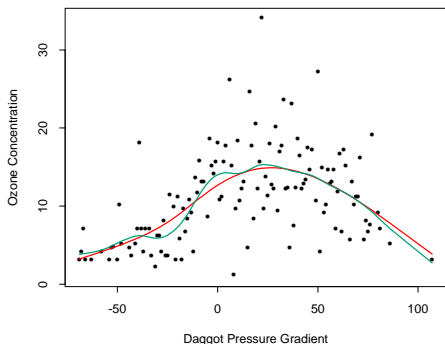- 128 observations of pressure and ozone.
- Two fitted smoothing splines.
- third to sixth eigenvectors of the spline smoother matrices $u_k$ against $x$.
- eigendecomposition of $S$: eigenvaules $d_k$ (right)

$$\mathbf{S}_\lambda = \sum_{k=1}^{N} \rho_k(\lambda) u_k u_k^T$$

- $\rho_k(\lambda) = \frac{1}{1+\lambda d_k}$.
- Right: 3rd-6th eigenvectors as a function of $x$ and smoothed $df = 5$ in red

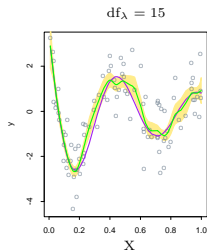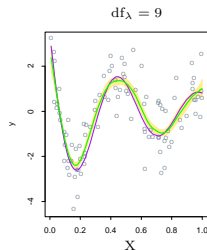# Selection degrees of freedom

- The degrees of freedom *df* (or the complexity penalty $\lambda$) are usually selected to minimize the expected prediction error.

- More specifically, the crossvalidation estimate of the error.

## Example

- $f(X) = \frac{\sin(12(X+0.2))}{X+0.2}$
- $Y = f(X) + \epsilon$
- $X \sim U[0,1]$, $\epsilon \sim N(0,1)$, $N = 100$.
- *df* selected by crossvalidation is 9.
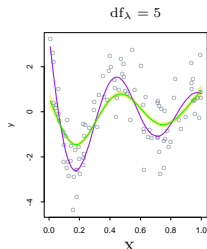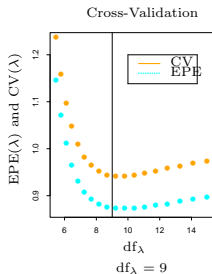
# Multidimensional Splines

- $X \in \mathbb{R}^2$
- $h_{1k}(X_1)$, $k = 1, \ldots, M_1$ in the first coordinate
- $h_{2k}(X_2)$, $k = 1, \ldots, M_2$ in the second coordinate.
- $M_1 \times M_2$ dimensional tensor product basis is defined by

$$g_{jk}(X) = h_{1j}(X_1)h_{2k}(X_2)$$

- can be used for representing a two-dimensional function:

$$g(X) = \sum_{j=1}^{M_1} \sum_{k=1}^{M_2} \theta_{jk} g_{jk}(X)$$

- coefficients can be fitted by least squares.
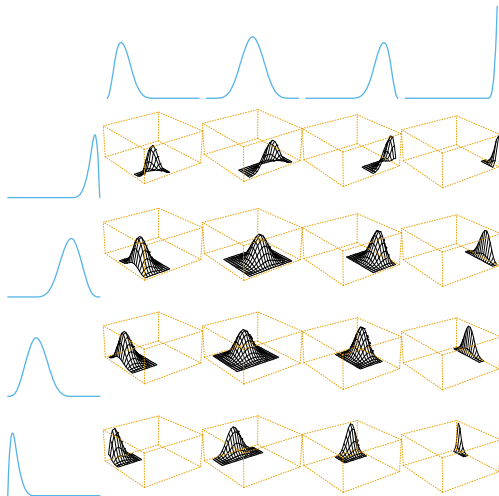
# Additive logistic regression vs. tensor product

- In higher dimensions, the number of basic functions and parameters grows rapidly.
- Consider to add the basic elements iteratively, as the additive MARS method introduced later.
- left: df=7, right df=16



Additive Natural Cubic Splines - 4 df each
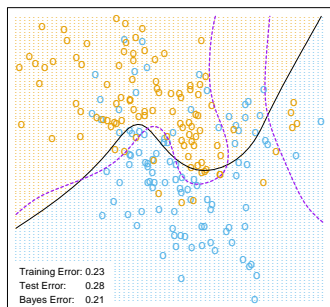
Natural Cubic Splines - Tensor Product - 4 df each
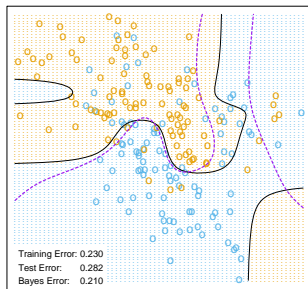
# Multidimensional smoothing splines

- Let us place the knot into each example
- and add a complexity penalty $J$ (below).
- It can be generalized for an arbitrary dimension.
- The solution has the form:
  - $f(x) = \beta_0 + \beta^T x + \sum_{j=1}^{N} \alpha_j h_j(x)$
  - where $h_j(x) = \eta(||x - x_j||)$ and $\eta(z) = z^2 2 \log z^2$.

- complexity $O(N^3)$
- or $O(NK^2 + K^3)$ with $K$ knots.



Systolic Blood Pressure
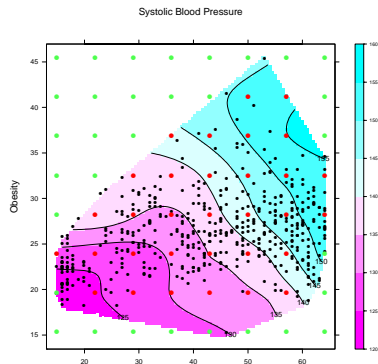
$$J[f] = \int \int_{\mathbb{R}^2} \left[ \left( \frac{\partial^2 f(x)}{\partial x_1^2} \right) + \left( \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} \right) + \left( \frac{\partial^2 f(x)}{\partial x_2^2} \right) \right] dx_1 \, dx_2$$

implemented in R and OpenCV

# Summary

We learned about

- regression **splines** (one dimensional $X$) - these formulas you should know
- **B-splines** - for faster fit - no formulas necessary
- **natural splines** - linear on the borders
- **smoothing splines** - complexity penalty for the second derivative
  - the solution is a natural spline.
- Generalizations to more dimensions
  - thin plate splines
  - multidimensional smoothing splines.

# Kernel Methods

- estimate regression function $f(x) \in \mathbb{R}$
- a different but simple model separately at each query point $x_0$.
- The resulting $\hat{f}(X)$ is smooth in $\mathbb{R}^p$.
- Localization is achieved via a weighting function er **kernel** $k_\lambda(x_0, x_i)$
  - assigns a weight to $x_i$ based on its distance form $x_0$.
- $\lambda$ is a parameter that dictates the width of the neighbourhood.
- **memory based methods**
  - little or no training
  - the model is the entire training data set.



Nearest-Neighbor Kernel



Epanechnikov Kernel
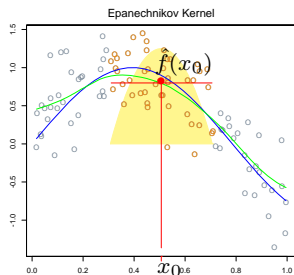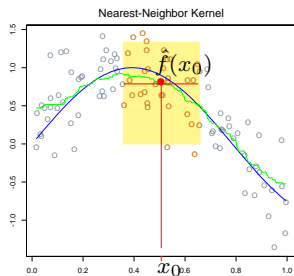
# k-NN, Epanechnikov Kernel

- **k-Nearest Heighbour** kernel
  - $N_k(x)$ is the set of $k$ points nearest to $x$ in squared distance
  - all have equal weight
  - $\hat{f}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$.
  - $\hat{f}(x)$ is bumpy, discontinuous.

- **Nadaraya-Watson** kernel-weighted average

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} k_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} k_\lambda(x_0, x_i)}$$

- with the **Epanechnikov** quadratic kernel
  $k_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right)$

- with
  $D(t) = \begin{cases} \frac{3}{4}(1 - t^2) & \text{if } |t| \leq 1 \\ 0 & \text{otherwise.} \end{cases}$



Nearest-Neighbor Kernel



Epanechnikov Kernel
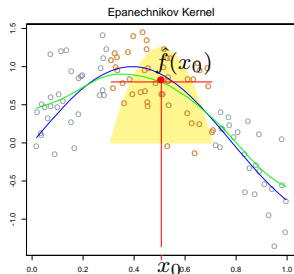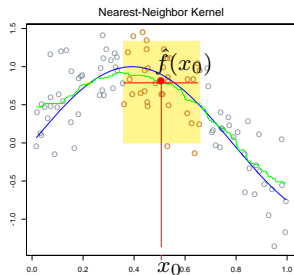
# Example

- Red circles: data
- Blue: epanechnikov kernel for $(-1.0)$
- Predicted values: green dashed line
- predicted value $\hat{f}(-1.0) = 1.55$.



Weighted estimate at $x_0 = -1$
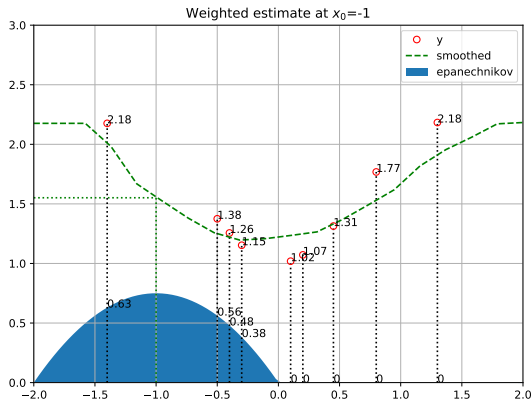
$$\frac{(0.63 * 2.18 + 0.56 * 1.38 + 0.48 * 1.26 + 0.38 * 1.15 + 0 * others)}{(0.63 + 0.56 + 0.48 + 0.38)} = 1.55$$

# Kernels - variable width, shapes

- The width $\lambda$ may vary $h_\lambda(x_0)$ with $x_0$
- mo general formula for he kernel
  $$k_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{h_\lambda(x_0)}\right)$$
- for k-NN, $h_k(x_0) = |x_0 - x_{|k|}|$
- where $x_{|k|}$ is the $k$th closest $x_i$ to $x_0$.

- **Tri-cube kernel**
  $$D(t) = \begin{cases} (1 - |t|^3)^3 & \text{if } |t| \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$
- **Gaussian kernel**
  $$D(t) = \frac{1}{\lambda} e^{-\frac{\|x - x_0\|^2}{2\lambda}}$$
- **Epanechnikov**
  $$D(t) = \begin{cases} \frac{3}{4}(1 - t^2) & \text{if } |t| \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$
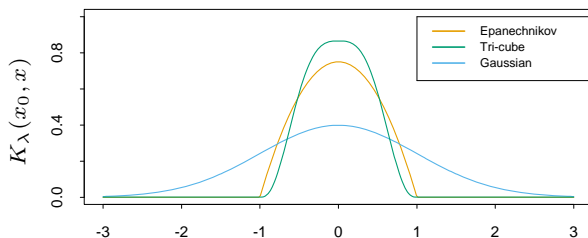
# Local Linear Regression

- Locally-weighted averages can be badly biased on the boundaries of the domain
- or whenever $X$ are not equally spaced.
- Fitting straight lines may help (a bit).
- **Locally weighted regression**

$$min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} k_\lambda(x_0, x_i)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2$$



N-W Kernel at Boundary

- The estimate is: $\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$.
  - For $x^T \to (1, x)$, **X** is $N \times (p + 1)$ matrix, **W** $N \times N$ diagonal matrix $k_\lambda(x_0, x_i)$. Then

$$\hat{f}(x_0) = x_0^T (X^T W(x_0)X)^{-1}X^T)W(x_0)y$$

- what is linear function of $y$.



Local Linear Regression at Boundary
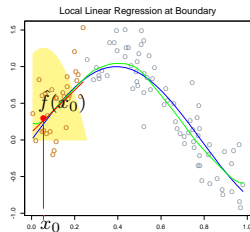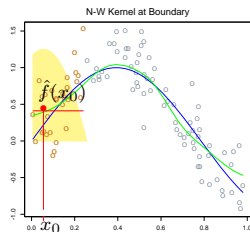
# Local Polynomial Regression

- Local linear fits can help bias dramatically at the boundaries.
- local quadratic fits tend to be most helpful in reducing bias due to curvature in the interior of the domain.
- Recommended to select the degree by the application, not to combine linear boundaries and quadratic interior.



Local Linear in Interior

$\hat{f}(x_0)$



Local Quadratic in Interior

$\hat{f}(x_0)$
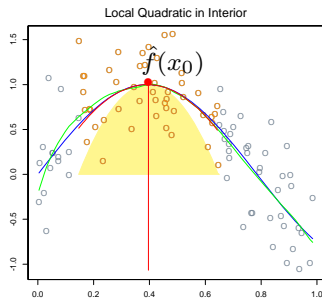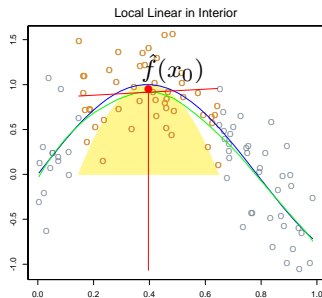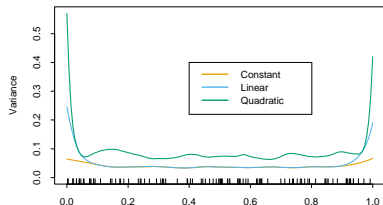
# Selecting the Width of the Kernel

- crossvalidation
- $\hat{f} = S_\lambda y$
  - $df = trace(S_\lambda)$
- Right: comparison of the tri-cube
  local linear regression kernels
  (orange) and smoothing splines
  (blue) with matching degrees of
  freedom 5.86.
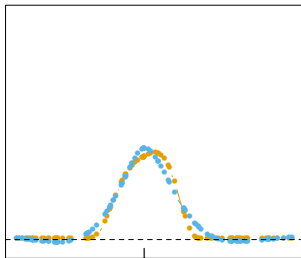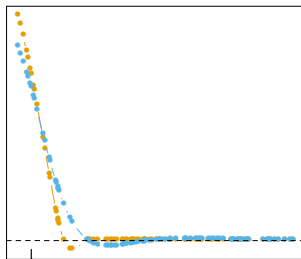
$$k_\lambda(x_0, x) = D\left(\frac{\|x - x_0\|}{h_\lambda(x_0)}\right)$$

- Structured local regression: a positive semidefinite matrix $A$ to weigh the different coordinates:

$$k_\lambda(x_0, x) = D\left(\frac{(x - x_0)^T A (x - x_0)}{h_\lambda(x_0)}\right)$$

# Computational Consideration
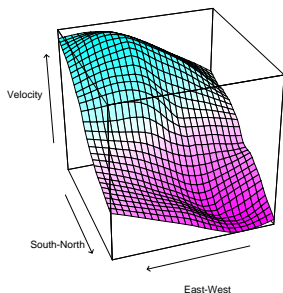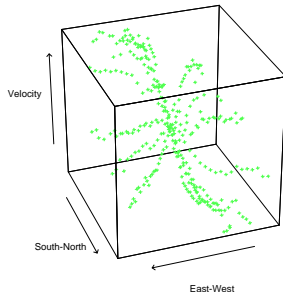
## Kernel smoothing complexity

- Model is the entire training data set.
- The fitting is done at evaluation or prediction.
- Single observation $x_0$ fit is $O(N)$,
- expansion in $M$ basis functions $O(M)$ for one evaluation, typically $M \sim O(logN)$.
- Basis function method have an initial cost at least $O(NM^2 + M^3)$.
- Smoothing parameter $\lambda$ usually determined off-line by cross-validation, at cost of $O(N^2)$.
- Popular implementations of local regression *loess* is S-PLUS compute the fit exactly at $M$ locations $O(NM)$ and interpolate to fit elsewhere ($O(M)$ per evaluation).

# Linear and k-NN Methods for Classification and Their Extensions

- likelihood example
- logistic regression
  - ext. logistic regression with $L_1$ penalty, elastic net penalty
- linear and quadratic discriminant analysis
  - ext. regularized discriminant analysis
  - ext. reduced rank discriminant analysis
  - ext. diagonal discriminant analysis
- Nearest-neighbor methods
  - k-NN
  - Local likelihood (local logistic regression)
  - ext. Discriminating Adaptive NN methods (DANN)
- ? Support Vector Machines

# Probability of the data given the model

- Assume we have 15 red balls and 5 blue balls in a bag.
- Repeat 5x:
  - select a ball
  - put it back.
- The probability of the sequence red, blue, blue,red, red is $\frac{3}{4} \cdot \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{3}{4} \cdot \frac{3}{4}$.
- The logarithm $log_2$ of the probability is $\approx -0.4 - 2 - 2 - 0.4 - 0.4 = -5.2$

# Likelihood of the model given the data

- Assume we do not know the probabilities, let $\theta$ be the probability of *red*. We have following probabilities of data for different $\theta$.

| $\theta$ | red | blue | blue | red | red | |
|---|---|---|---|---|---|---|
| $\frac{3}{4}$ | $\frac{3}{4}$ | $\frac{1}{4}$ | $\frac{1}{4}$ | $\frac{3}{4}$ | $\frac{3}{4}$ | $\frac{3^3}{4^5}$ |

- Take the $log_2$ of the probabilities:

| $\theta$ | red | blue | blue | red | red | |
|---|---|---|---|---|---|---|
| $\frac{1}{2}$ | $-1$ | $-1$ | $-1$ | $-1$ | $-1$ | $-5$ |
| $\frac{3}{5}$ | $-0.74$ | $-1.32$ | $-1.32$ | $-0.74$ | $-0.74$ | $-4.86$ |
| $\frac{3}{4}$ | $-0.4$ | $-2$ | $-2$ | $-0.4$ | $-0.4$ | $-5.2$ |

- Probability of the data given model is called **likelihood** of the model $\theta$ given the data.
- Maximum likelihood $\theta$ estimate is in our case $\frac{3}{5}$.
- Predicting probabilities, maximum likelihood estimate is the same as maximum log-likelihood estimate.

# (Log)likelihood

| train data | | prediction | | | likelihood | loglik |
|---|---|---|---|---|---|---|
| $x_i$ | $g_i$ | $P(green|x_i)$ | $P(blue|x_i)$ | $P(yellow|x_i)$ | | |
| 1 | green | $\frac{1}{2}$ | 0 | $\frac{1}{2}$ | $\frac{1}{2}$ | $-1$ |
| 1 | yellow | $\frac{1}{2}$ | 0 | $\frac{1}{2}$ | $\frac{1}{2}$ | $-1$ |
| 2 | green | $\frac{2}{3}$ | $\frac{1}{3}$ | 0 | $\frac{2}{3}$ | $log_2 \frac{2}{3}$ |
| 2 | green | $\frac{2}{3}$ | $\frac{1}{3}$ | 0 | $\frac{2}{3}$ | $log_2 \frac{2}{3}$ |
| 2 | blue | $\frac{2}{3}$ | $\frac{1}{3}$ | 0 | $\frac{1}{3}$ | $-log_2 3$ |
| 3 | blue | 0 | 1 | 0 | 1 | 0 |
| | | | | | | $-2 - log_2 3$ |
| | | | | | | $+2 log_2 \frac{2}{3}$ |

- **loglik** logarithm with base $e$ of likelihood function is defined as:

$$\ell(\theta) = \sum_{i=1}^{N} log_e(P(G = g_i|x_i, \theta))$$

- Logistic regression uses:

$$P(G = g_k|X = x) = \frac{e^{\beta_{k0} + \beta_k^T x}}{1 + \sum_{l=1,\dots,K-1} e^{\beta_{l0} + \beta_l^T x}}.$$

# Logistic Function

- With more than two classes, the linear regression to the class identifier may suffer the **masking problem**.
    - Some class is masked due to the linearity constraint (the blue class in the figure right).
    - Quadratic fit to the identifier function solves this case.
    - Logistic regression, LDA, SVM, k-NN solve this case naturally.
- Probability should be from the interval $\langle 0, 1 \rangle$.
- Linear prediction is transformed by logistic function (sigmoid) with the maximum $L$.
- **logistic** $\frac{L}{1+e^{-k(x-x_0)}}$.
- Inverse function is called logit.
- **logit** $\log \frac{p}{1-p}$,
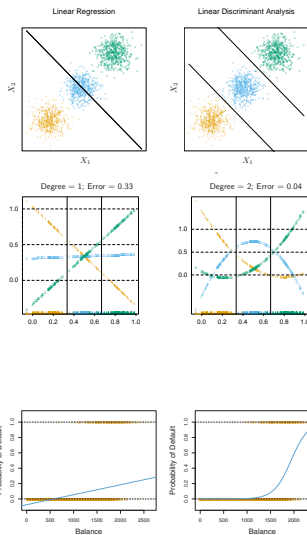
# Logistic Regression

- For $K-$ class classification we estimate $(p+1) \times (K-1)$ parameters $\theta = \{\beta_{10}, \beta_1^T, \ldots, \beta_{(K-1)0}, \beta_{K-1}^T\}$.

$$\log \frac{P(G = g_1 | X = x)}{P(G = g_K | X = x)} = \beta_{10} + \beta_1^T x$$

$$\log \frac{P(G = g_2 | X = x)}{P(G = g_K | X = x)} = \beta_{20} + \beta_2^T x$$

$$\vdots$$

$$\log \frac{P(G = g_{K-1} | X = x)}{P(G = g_K | X = x)} = \beta_{(K-1)0} + \beta_{K-1}^T x$$

that is

$$p_k(x; \theta) \leftarrow P(G = g_k | X = x) = \frac{e^{\beta_{k0} + \beta_k^T x}}{1 + \sum_{l=1,\ldots,K-1} e^{\beta_{l0} + \beta_l^T x}}$$

$$p_K(x; \theta) \leftarrow P(G = g_K | X = x) = \frac{1}{1 + \sum_{l=1,\ldots,K-1} e^{\beta_{l0} + \beta_l^T x}}.$$

# Fitting Logistic Regression Two class

- This model is estimated iteratively maximizing conditional likelihood of G given $X$.

$$\ell(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta)$$

- Two class model: $g_i$ encoded via a 0/1 response $y_i$; $y_i = 1$ iff $g_k = g_1$. Let $p(x; \theta) = p_1(x; \theta)$, $p_2(x; \theta) = 1 - p(x; \theta)$. Then:

$$\begin{aligned}
\ell(\theta) &= \sum_{i=1}^{N} (y_i \log p(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta))) \\
&= \sum_{i=1}^{N} (y_i \beta^T x_i - \log(1 + e^{\beta^T x_i}))
\end{aligned}$$

- Set derivatives to zero:

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i (y_i - p(x_i; \beta)) = 0,$$

- which is $p + 1$ nonlinear equations in $\beta$.
- First component: $x_i \equiv 1$ specifies $\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p(x_i; \beta)$ the expected

# Newton–Raphson Algorithm

- We use Newton–Raphson Algorithm to solve the system of equations

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i(y_i - p(x_i; \beta)) = 0,$$

- we need the second–derivative or Hessian matrix

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -\sum_{i=1}^{N} x_i x_i^T p(x_i; \beta)(1 - p(x_i; \beta)).$$

- Starting with $\beta^{old}$ a single Newton–Raphson update is

$$\beta^{new} = \beta^{old} - \left(\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T}\right)^{-1} \frac{\partial \ell(\beta)}{\partial \beta},$$

- where the derivatives are evaluated at $\beta^{old}$.

# Newton–Raphson Algorithm in Matrix Notation

Let us denote:

$\quad$ $\mathbf{y}$ $\quad$ the vector of $y_i$

$\quad$ $\mathbf{X}$ $\quad$ $N \times (p+1)$ data matrix $x_i$

$\quad$ $\mathbf{p}$ $\quad$ the vector of fitted probabilities with $i$th element $p(x_i; \beta^{old})$

$\quad$ $\mathbf{W}$ $\quad$ diagonal matrix with weights $p(x_i; \beta^{old})(1 - p(x_i; \beta^{old}))$

$$\frac{\partial \ell(\beta)}{\partial \beta} = \mathbf{X}^T(\mathbf{y} - \mathbf{p})$$

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -\mathbf{X}^T\mathbf{W}\mathbf{X}$$

The Newton–Raphson step is ($\beta^0 \leftarrow 0$)

$$
\begin{aligned}
\beta^{new} &= \beta^{old} + (\mathbf{X^T W X})^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{p}) \\
&= (\mathbf{X^T W X})^{-1}\mathbf{X}^T\mathbf{W}(\mathbf{X}\beta^{old} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})) \\
&= (\mathbf{X^T W X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{z} \\
\mathbf{z} &= \mathbf{X}\beta^{old} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p}) \quad \textbf{adjusted response}
\end{aligned}
$$

- $p, W, z$ change each step

This algorithm is reffered to as **iteratively reweighted least squares IRLS**

$$\beta^{new} \leftarrow \arg\min_{\beta}(\mathbf{z} - \mathbf{X}\beta)^T\mathbf{W}(\mathbf{z} - \mathbf{X}\beta)$$

# South African Heart Disease

- Analyzing the risk factors of myocardial infarction MI
- (Note: prevalence 5.1%, in the data 160 positive 302 controls, the controls are underrepresented, consider weighting the data.)

**TABLE 4.2.** *Results from a logistic regression fit to the South African heart disease data.*

|  | Coefficient | Std. Error | $Z$ Score |
|---|---|---|---|
| (Intercept) | $-4.130$ | 0.964 | $-4.285$ |
| sbp | 0.006 | 0.006 | 1.023 |
| tobacco | 0.080 | 0.026 | 3.034 |
| ldl | 0.185 | 0.057 | 3.219 |
| famhist | 0.939 | 0.225 | 4.178 |
| obesity | -0.035 | 0.029 | $-1.187$ |
| alcohol | 0.001 | 0.004 | 0.136 |
| age | 0.043 | 0.010 | 4.184 |

- Wald test: Z score $|Z| > 2$ is significant at at the 5% level.

**TABLE 4.3.** *Results from stepwise logistic regression fit to South African heart disease data.*

|  | Coefficient | Std. Error | $Z$ score |
|---|---|---|---|
| (Intercept) | $-4.204$ | 0.498 | $-8.45$ |
| tobacco | 0.081 | 0.026 | 3.16 |
| ldl | 0.168 | 0.054 | 3.09 |
| famhist | 0.924 | 0.223 | 4.14 |
| age | 0.044 | 0.010 | 4.52 |

# South African Heart Disease

- Wald test: Z score $|Z| > 2$ is significant at at the 5% level.

TABLE 4.3. *Results from stepwise logistic regression fit to South African heart disease data.*

|  | Coefficient | Std. Error | $Z$ score |
|---|---|---|---|
| (Intercept) | $-4.204$ | 0.498 | $-8.45$ |
| tobacco | 0.081 | 0.026 | 3.16 |
| ldl | 0.168 | 0.054 | 3.09 |
| famhist | 0.924 | 0.223 | 4.14 |
| age | 0.044 | 0.010 | 4.52 |

- $P(MI|x_i, \theta) = \frac{e^{-4.204 + 0.081 x_{tobacco} + 0.168 x_{ldl} + 0.924 x_{famhist} + 0.044 x_{age}}}{1 + (e^{-4.204 + 0.081 x_{tobacco} + 0.168 x_{ldl} + 0.924 x_{famhist} + 0.044 x_{age}})}$
- Interval estimate $odds_{tobacco} = e^{0.081 \pm 2 \times 0.026} = (1.03, 1.14)$ increase of odds of $MI$ based of the increase of $x_{tobacco}$.

# $L_1$ regularization 'Lasso'-like

$$argmax_{\beta_0, \beta} \left( \sum_{i=1}^{N} (y_i(\beta_0 + \beta^T x_i) - \log(1 + e^{(\beta_0 + \beta^T x_i)})) - \lambda \sum_{j=1}^{p} |\beta_j| \right)$$

- Newton–Raphson Algorithm or nonlinear programming.
- $\lambda = 0$ standard logistic regression.
- $\lambda \to \infty$ moves coefficients towards 0.
- $\beta_0$ is not included into the penalty.
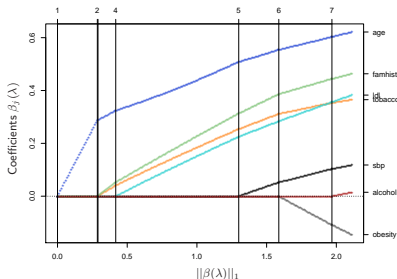
# Linear Discriminant Analysis

- LDA assumes multivariate gaussian distribution of each class with a common covariance matrix.

$$\phi(k) = \frac{1}{\sqrt{|2\pi\Sigma|}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)}$$

  - Under this assumptions it provides bayes optimal estimate.
- Different covariance matrix for each class leads to Quadratic Discriminant Analysis.
- Let us denote $N_k$ number of training data in the class $G_k$.
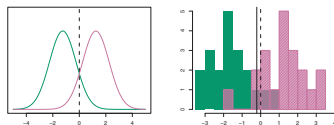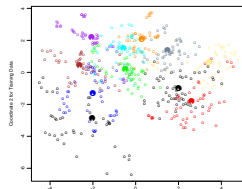


**FIGURE 4.4.** Left: *Two one-dimensional normal density functions are shown.*

# Linear Discriminant Analysis

The LDA model parameters: the mean and probability of each class $\{\mu_i, \pi_i\}_{i=1}^{K}$ and the common covariance matrix $\Sigma$ can be evaluated directly.

$$\hat{\pi}_k = \frac{N_k}{N}$$

$$\hat{\mu}_k = \frac{\sum_{\{x_i : G(x_i) = g_k\}} x_i}{N_k}$$

$$\hat{\Sigma} = \sum_{k=1}^{K} \sum_{\{x_i : G(x_i) = g_k\}} \frac{(x_i - \mu_k)(x_i - \mu_k)^T}{(N - K)}$$

$$\phi_k(x) = N(\mu_k, \Sigma)$$

$$P(G = g_k | X = x) = \frac{\phi_k(x)\pi_k}{\sum_{\ell=1}^{K} \phi_\ell(x)\pi_\ell}$$

To classify new instance $x$ we predict the $G_k$ with maximal $\delta_k$:



$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + log.$$
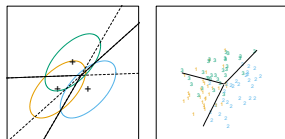
# Quadratic Discriminant Analysis

Quadratic discriminant analysis estimates the covariance matrix for each class independently. The rest is the same as for the LDA.

$$\hat{\pi}_k = \frac{N_k}{N}$$

$$\hat{\mu}_k = \frac{\sum_{\{x_i : G(x_i) = g_k\}} x_i}{N_k}$$

$$\hat{\Sigma}_k = \sum_{\{x_i : G(x_i) = g_k\}} \frac{(x_i - \mu_k)(x_i - \mu_k)^T}{(|G_k| - 1)}$$

$$f_k(x) = N(\mu_k, \Sigma_k)$$

$$P(G = g_k | X = x) = \frac{f_k(x)\pi_k}{\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell}$$

To classify new instance $x$ we predict the $G_k$ with maximal $\delta_k$:

$$\delta_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) - \frac{1}{2}\log |\Sigma_k| + \log \pi_k.$$

# Quadratic and Regularized Discriminant Analysis

- QDA has substantially more parameters. It is questionable whether it is worth to increase the model complexity.
  - LDA parameters: $(K-1) \times (p+1)$
  - QDA parameters: $(K-1) \times (\frac{p(p+3)}{2} + 1)$.
- **Regularized discriminant analysis** takes a weighted average of LDA and QDA to tune the model complexity.

$$\hat{\Sigma}_k(\alpha) = \alpha \hat{\Sigma}_k + (1 - \alpha) \hat{\Sigma}$$

Regularized Discriminant Analysis on the Vowel Data
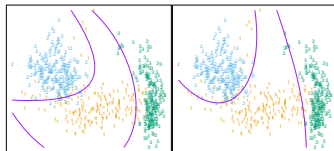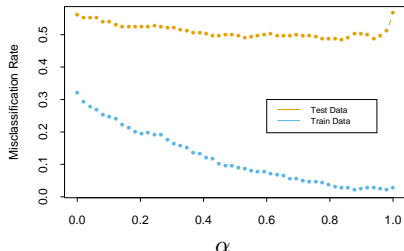




**FIGURE 4.6.** *Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $X_1, X_2, X_1 X_2, X_1^2, X_2^2$). The right plot shows the*

# Computations for LDA

## Linear and Quadratic Discriminant Analysis

- $O(N^3)$, often $O(N^{2.376})$
- QDA and LDA may be computed using matrix decomposition:
  - Compute the eigendecomposition for each
    $(x - \hat{\mu}_k)^T \hat{\Sigma}_k^{-1} (x - \hat{\mu}_k) = [\mathbf{U}_k^T (x - \hat{\mu}_k)]^T \mathbf{D}_k^{-1} [\mathbf{U}_k^T (x - \hat{\mu}_k)]$
  - $\log |\hat{\Sigma}_k| = \sum_\ell \log d_{k\ell}$.
- Using this decomposition, LDA classifier can be implemented by the following pair of steps:
  - *Sphere* the data with respect to the common covariance estimate $\hat{\Sigma}$:
    $X^* \leftarrow \mathbf{D}^{-\frac{1}{2}} \mathbf{U}^T X$, where $\hat{\Sigma} = \mathbf{U} \mathbf{D} \mathbf{U}^T$.
    The common covariance estimate of $X^*$ will now be the identity.
  - Classify to the closest class centroid in the transformed space, modulo the effect of the class prior probabilities $\pi_k$.

# Reduced–Rank Linear Discriminant Analysis
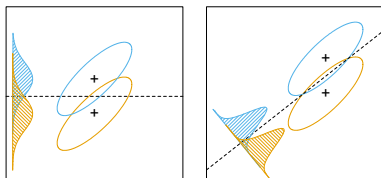


$$max_a \frac{a^T B a}{a^T W a}$$

Finding the sequence of optimal subspaces for LDA:

- compute the $K \times p$ matrix of class centroids $M$ and the common covariance matrix $W$ (**within–class** covariance);

- compute $M^* = MW^{-\frac{1}{2}}$ using the eigen–decomposition of $W$;

- compute $B^*$ **between–class** covariance, the covariance matrix of $M^*$ and its eigen–decomposition $B^* = V^* D_B V^{*T}$.

  - order $D_B$ in the decreasing order
  - $v_\ell^*$ of $V^*$ in sequence define the coordinates of the optimal subspaces
  - $Z_\ell = v_\ell^T X$ with $v_\ell = W^{-\frac{1}{2}} v_\ell^*$.

# Vovel Example

Example Vovel data ESL:

- $X \in \mathbb{R}^{10}$:
- $k = 11$ classes.

|  | train | test |
|---|---|---|
| Linear regression | 0.48 | 0.67 |
| Linear discriminant analysis | 0.32 | 0.56 |
| Quadratic discriminant analysis | 0.01 | 0.53 |
| Logistic regression | 0.22 | 0.51 |



LDA and Dimension Reduction on the Vowel Data



Classification in Reduced Subspace
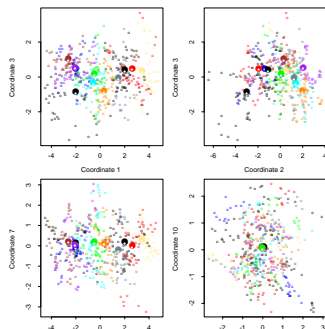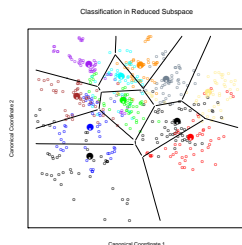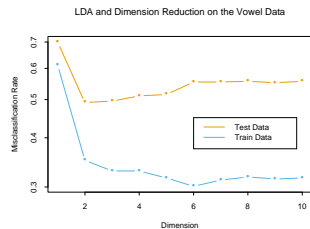
# Diagonal Linear Discriminant Analysis

- **With really many dimensions** the even reduced rank does not work.
- Gene expression experiment
  - 2308 genes (columns)
  - 63 samples (rows), from a set of microarray experiments.
  - The samples arose from small, round blue-cell tumors (SRBCT) found in children, and are classified into four major types:
    - BL (Burkitt lymphoma),
    - EWS (Ewing's sarcoma),
    - NB (neuroblastoma),
    - and RMS (rhabdomyosarcoma).
  - There is an additional test data set of 20 observations.
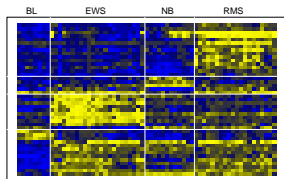


- diagonal-covariance LDA
$$\delta_k(x^*) = -\sum_{j=1}^{p} \frac{(x_j^* - \overline{x}_{jk})^2}{s_j^2} + 2\log(\pi_k)$$

- $s_j$ is the pooled within-class standard deviation of the jth gene
- $\overline{x}_{jk} = \sum_{i \in C_k} \frac{x_{ij}}{N_k}$
- $\tilde{x}_k = (\overline{x}_{1k}, \ldots, \overline{x}_{jk}, \ldots, \overline{x}_{pk})^T$ is the $k$ class centroid.

# Linear Regression with Elastic Net Penalty

- Elastic net penalty

$$max_{\{\beta_{0k}, \beta_k \in \mathbb{R}^p\}_1^K} \left[ \sum_{i=1}^{N} \log P(g_i|x_i) - \lambda \left( \sum_{k=1}^{K} \sum_{j=1}^{p} (\alpha|\beta_{kj}| + (1-\alpha)\beta_{kj}^2) \right) \right]$$
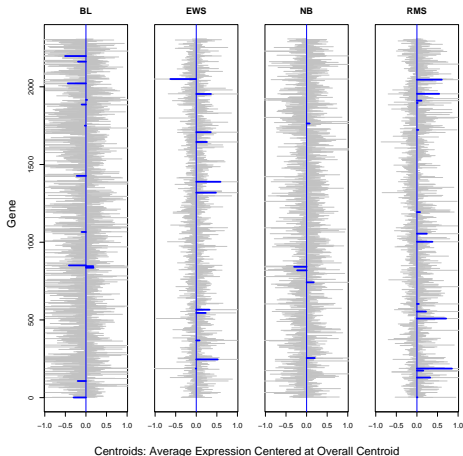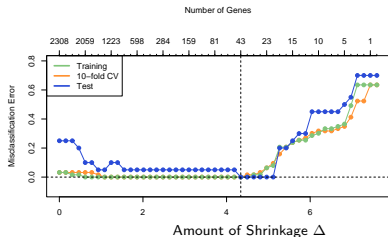


Centroids: Average Expression Centered at Overall Centroid

# Nearest-Neighbor Methods

- The **nearest-neighbor methods** use those observations in the training set $\mathcal{T}$ closest in the input space to $x$ to form $\hat{f}$.

$$\hat{g}(x) = majority_{x_i \in N_k(x)} g(x_i)$$

- nice, but suffers the curse of dimensionality.



15-Nearest Neighbor Classifier
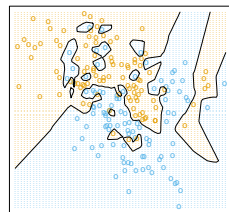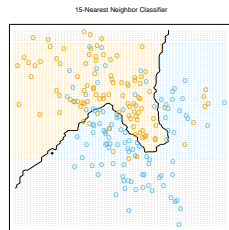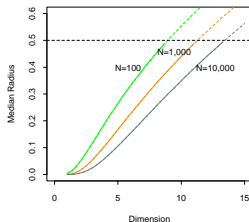
# Disciminang Adaptive Nearest-Neighbor Methods (DANN)

- The metric at a query point is defined by

$$D(x, x_0) = (x - x_0)^T \Sigma (x - x_0)$$

- where

$$\Sigma = W^{-\frac{1}{2}} [W^{-\frac{1}{2}} B W^{-\frac{1}{2}} + \epsilon I] W^{-\frac{1}{2}}$$

- where $\epsilon = 1$ adjusts the neighbourhood and
- $W, B$ are within and between class covariance fitted at the neighbourhood.



- Close to the class boundary, the $\Sigma$ shrinks out of the boundary
- in the interior it remains circular.
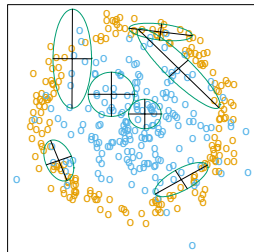
# Local Likelihood and other methods

- Logistic and log-linear models involve the covariates in a linear fashion.
- We fit the model locally at $x_0$ and weight the loglik by the kernel $k_\lambda$
- and center the estimate at $x_0$.

$$
\begin{aligned}
\ell(\beta_{x_0}) &= \sum_{i=1}^{N} k_\lambda(x_0, x_i) \ell(y_i, (x - x_0)^T \beta_{x_0}) \\
&= \sum_{i=1}^{N} k_\lambda(x_0, x_i) \left\{ y_i \beta_{x_0}^T (x_i - x_0) - \log(1 + e^{\beta_{x_0}^T (x_i - x_0)}) \right\}
\end{aligned}
$$
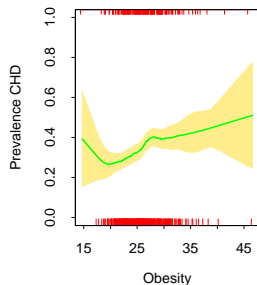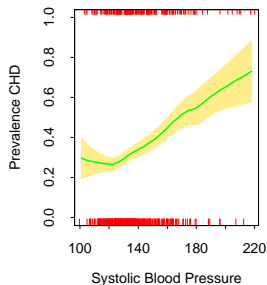


Note: Increased prevalence for small values due to retrospective data: some people with diagnosed CHD started more healthy life.

# Generalized Additive Model (gam)

- **Each feature $X_j$ is approximated by a natural spline.**
- The overall model is:

$$logit[P(CHD|X)] = \theta_0 + h_1(X_1)^T \theta_1 + h_2(X_2)^T \theta_2 + \ldots + h_p(X_p)^T \theta_p$$

- $\theta_j$ are vectors of coefficients multiplying their associated vector of natural spline basis functions $h_j$
- four basis functions (three inner knots) per spline in this example.
- binary *familyhist* with a single coefficient.
- Combine all $p$ vectors of basic functions into one big vector $h(X)$, $df = 1 + \sum_{j=1}^{p} df_j$
- each basis function is evaluated at each of the $N$ samples
- resulting in a $N \times df$ basis matrix **H**.
- and use 'standard' logistic regression.

# South African Heart Disease continued

- Alcohol not significant by AIC test
- covariance $Cov(\hat{\theta})$ is estimated by
  $\hat{\Sigma} = (\mathbf{H^T W H})^{-1}$
  - $W$ the diagonal weight matrix
- variance of a single variable $j$ is:
  - $v_j(X_j) = Var[f_j(X_j)] = h_j(X_j)^T \hat{\Sigma}_{jj} h_j(X_j)$
- error bounds $\hat{f}_j(X_j) \pm 2\sqrt{v_j(X_j)}$.
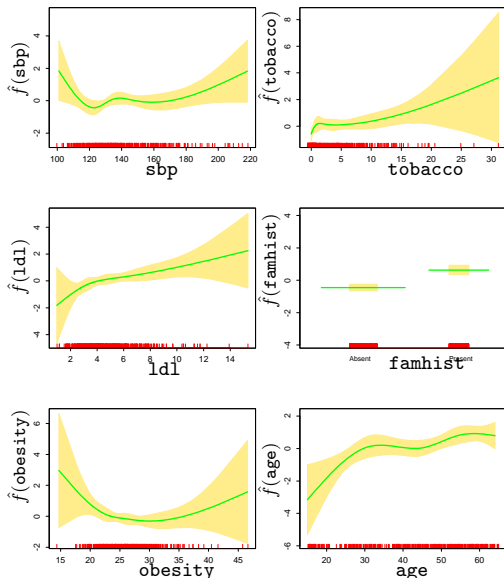
# Summary

- likelihood example
- logistic regression
  - ext. logistic regression with $L_1$ penalty, elastic net penalty
- linear and quadratic discriminant analysis
  - ext. regularized discriminant analysis
  - ext. reduced rank discriminant analysis
  - ext. diagonal discriminant analysis
- Nearest-neighbor methods
  - k-NN
  - Local likelihood (local logistic regression)
  - ext. Discriminating Adaptive NN methods (DANN)
- ? Support Vector Machines

# Model Assessment and Selection

- We assume i.i.d. data
  - independently (independent samples)
  - identically (the same distribution)
  - distributed
- Assume many iid datasets
  - 1 line= train/test curve for 1 set of samples (data)
  - data used to fit the model **Training error** is
  $\overline{err} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$,
  - data not used to fit the model **Test error** is
  $err = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$,
  - Test error is a point estimate of the generalization error.
  - Dark red line is an average of test errors, a more robust estimate.
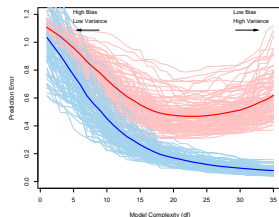


## Definition (Generalization error)

**Generalization error** is the expected prediction error over an independent test sample

$$Err = \mathbb{E}[L(Y, \hat{f}(X))]$$

where both $X$ and $Y$ are drawn randomly from their joint distribution (population).

# Loss functions for regression

## Definition

- **Square error loss** $L(y, \hat{y}) = (y - \hat{y})^2$
- **Absolute error loss** $L(y, \hat{y}) = |y - \hat{y}|$
- **Huber error loss**

$$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| \leq \delta, \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2, & \text{otherwise.} \end{cases}$$
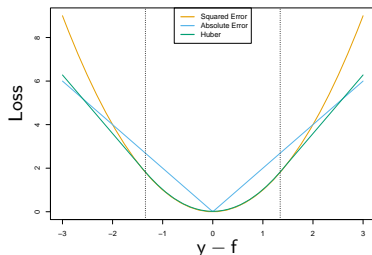
# Classifier Assessment and Selection

- Qualitative response $G$ taking one of $K$ values labeled as $1, \ldots, K$.
- Typically $\hat{G}(X) = \arg\max_k \hat{p}_k(X)$.
- Figure for binary response encoded $\{-1, +1\}$
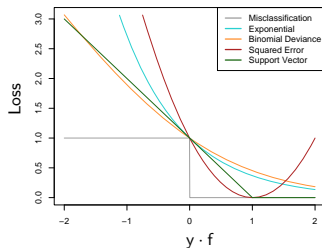


**Definition (Loss functions for classification)**

- **0-1 loss, misclassification**

$$L(G, \hat{G}(X)) = I(G \neq \hat{G}(X))$$

- **log-likelihood, cross-entropy, deviance**

$$L(G, \hat{p}(X)) = -2I(G = k) \log \hat{p}_k(X)$$

$$L_{\{0,1\}}(g_1, \hat{p}_1) = -2[p_1 \log \hat{p}_1 + (1-p_1) \log(1-p_1)]$$

- two classes encoded $\{-1, +1\}$
  - **exponential**
    $$e^{-yf}$$
  - **support vector**
    $$max(0, 1 - yf(x))$$

- Non-negative loss matrix $L \in \mathbb{R}^{K \times K}$ with 0 on the diagonal.

# Data Rich Situation

| Train | Validation | Test |
|:---:|:---:|:---:|

If we have enough data, we split the dataset

- **Train** train the model
- **Validate** select appropriate model parameter $\alpha$, $\lambda$, usually the model complexity, cost penalty
- **Test** estimate the test error on an independent sample.

Recommended ratios:

- $\frac{1}{2} : \frac{1}{4} : \frac{1}{4}$ with the validation set
- $\frac{2}{3} : \frac{1}{3}$ without the validation need.

# Stratified Selection

- Assume small disease prevalence.
  - We split the data healthy/sick and split each group separately.
- Consider a model over different branches of your company:
  - to estimate a new branch, all data from some branches should be selected as test ones
  - not a random sample from each branch.

Other methods only since we almost never have enough of the data.

# Error Estimation with a few data

Amount of data needed depends on:

- true function complexity
- the noise ratio.

The estimation method depends on the purpourse:

**Model selection**

- absolute value is not necessary, the difference is crucial
- any method can be used (AIC, BIC, cross-validation,...)
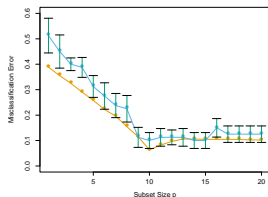
**Test error estimation**

- absolute value is necessary
- direct estimation (cross-validation, one-leave-out) preferred.

# Bias Variance Decomposition

- Assume $Y = f(X) + \epsilon$ where $\mathbb{E}(\epsilon) = 0$ and $Var(\epsilon) = \sigma_\epsilon^2$.
- we derive the expected prediction error of the regression fit $\hat{f}(X)$ at an input point $X = x_0$ using squared-error loss:

$$
\begin{aligned}
Err(x_0) &= \mathbb{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [\mathbb{E}\hat{f}(x_0) - f(x_0)]^2 + \mathbb{E}[\hat{f}(x_0) - \mathbb{E}\hat{f}(x_0)]^2 \\
&= \sigma_\epsilon^2 + Bias^2(\hat{f}(x_0)) + Var(\hat{f}(x_0)) \\
&= \text{Irreducible Error } + \text{ Bias}^2 + \text{Variance}.
\end{aligned}
$$

# Bias Variance Decomposition

K-Nearest neighbour

$$
\begin{aligned}
Err(x_0) &= \mathbb{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [\mathbb{E}\hat{f}(x_0) - f(x_0)]^2 + \mathbb{E}[\hat{f}(x_0) - \mathbb{E}\hat{f}(x_0)]^2 \\
&= \sigma_\epsilon^2 + [f(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})]^2 + \frac{\sigma_\epsilon^2}{k}.
\end{aligned}
$$

Linear fit

$$
\begin{aligned}
Err(x_0) &= \mathbb{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [\mathbb{E}\hat{f}_p(x_0) - f(x_0)]^2 + \|\mathbf{h}(x_0)\| \sigma_\epsilon^2 \\
\mathbf{h}(x_0)\mathbf{y} &= (x_0^T (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T})\mathbf{y} = \hat{f}_p(x_0).
\end{aligned}
$$

- hence $Var[\hat{f}_p(x_0)] = \|\mathbf{h}(x_0)\|\sigma_\epsilon^2$ and its average is $\frac{p}{N}\sigma_\epsilon^2$, hence

$$
\frac{1}{N}\sum_{i=1}^{N} Err(x_i) = \sigma_\epsilon^2 + \frac{1}{N}\sum_{i=1}^{N}[f(x_i) - \mathbb{E}\hat{f}(x_i)]^2 + \frac{p}{N}\sigma_\epsilon^2,
$$

# Penalized model complexity

- Consider ridge regression fit $\hat{f}_\lambda(x_0)$ with the fit

$$\mathbf{h}(x_0)\mathbf{y} = x_0^T (\mathbf{X^T X} + \lambda I)^{-1} \mathbf{X^T y}$$

- we break the bias more finely. Let $\beta_*$ be the fit with $\lambda = 0$,
  $\beta_* = \arg\min_\beta \mathbb{E}(f(X) - \beta^T X)^2$.

$$\begin{aligned}
\mathbb{E}_{x_0}[f(x_0) - \mathbb{E}\hat{f}_\lambda(x_0)]^2 &= \mathbb{E}_{x_0}[f(x_0) - \beta_*^T x_0]^2 + \mathbb{E}_{x_0}[\beta_*^T x_0 - \mathbb{E}\hat{\beta}_\lambda^T x_0]^2 \\
&= Ave[\text{Model Bias}]^2 + Ave[\text{Estimation Bias}]^2.
\end{aligned}$$

# Example: Bias-Variance Tradeoff

- 50 observations, 20 predictors, uniformly distributed in the hypercube $[0,1]^{20}$.

Left $Y$ is 0 if $X_1 \leq \frac{1}{2}$ and 1 if $X_1 > \frac{1}{2}$ and we apply $k$-NN

Right $Y$ is 1 if $\sum_{j=1}^{10} X_j > 5$ and 0 otherwise, and we use best subset linear regression of size $p$.

orange error (mean square resp. 0-1)

green squared bias

blue estimation variance



- For classification, only the error differ compared to regression.

# Optimism of the Training Error Rate

- **training error** $\overline{err} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$
- usually less than the true error $Err = \mathbb{E}[L(Y, \hat{f}(X))]$.
- we keep $x_i$ points fixed, we take new sample of **y** at these points.
- **in-sample error** $Err_{in} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathbf{y}} \mathbb{E}_{Y^{new}} L(Y_i^{new}, \hat{f}(x_i))$.
- **optimism** $op \leftarrow Err_{in} - \mathbb{E}_{\mathbf{y}}(\overline{err})$.
- for squared error, 0-1, and other loss functions: $op = \frac{2}{N} \sum_{i=1}^{N} cov(\hat{y}_i, y_i)$
- For a linear fit with $d = p$ inputs of basis functions and additive error model $Y = f(X) + \epsilon$ it simplifies

$$\sum_{i=1}^{N} cov(\hat{y}_i, y_i) = d\sigma_\epsilon^2$$

$$Err_{in} = \mathbb{E}_{\mathbf{y}} \overline{err} + 2\frac{d}{N}\sigma_\epsilon^2 = C_p = AIC_{gauss.}.$$

# The Optimism (Just for fun)

$$y_i - \hat{y}_i = (y_i - f(x_i)) + (f(x_i) - \mathbb{E}\hat{f}(x_i)) + (\mathbb{E}\hat{f}(x_i) - \hat{y}_i).$$

- We get six terms of the sum of squares:

$A_1 = \sum_i (y_i - f(x_i))^2 \qquad D_1 = 2\sum_i (y_i - f(x_i))(f(x_i) - \mathbb{E}\hat{f}(x_i))$

$B = \sum_i (f(x_i) - \mathbb{E}\hat{f}(x_i))^2 \qquad E = 2\sum_i (f(x_i) - \mathbb{E}\hat{f}(x_i))(\mathbb{E}\hat{f}(x_i) - \hat{y}_i)$

$C = \sum_i (\mathbb{E}\hat{f}(x_i) - \hat{y}_i)^2 \qquad F_1 = 2\sum_i (y_i - f(x_i))(\mathbb{E}\hat{f}(x_i) - \hat{y}_i)$

- For the new sample $Y$, taking the expectation.
- $A_2 = \sum_i \mathbb{E}_{Y^0}(Y_i^0 - f(x_i))^2$
- and similarly $D_2, F_2$.

$$
\begin{aligned}
N(Err_{in} - \overline{err}) &= (A_2 + B + C + D_2 + E + F_2) - (A_1 + B + C + D_1 + E + F_1) \\
&= (A_2 - A_1) + (D_2 - D_1) + (F_2 - F_1)
\end{aligned}
$$

- $\mathbb{E}(A_1) = \mathbb{E}(A_2) = N\sigma_\epsilon^2$,
- $\mathbb{E}(D_1) = 2\sum_i (\mathbb{E}(y_i) - f(x_i))(f(x_i) - \mathbb{E}\hat{f}(x_i)) = 0$ since $\mathbb{E}(y_i) = f(x_i)$
- $\mathbb{E}(D_2) = 0$ as well.
- $F_2 = 2\sum_i \mathbb{E}_{Y^0}\left[(Y_i^0 - f(x_i))(\mathbb{E}\hat{f}(x_i) - \hat{y}_i)\right] = 0$
  - as $\mathbb{E}(Y_i^0) = f(x_i)$ and $Y_i^0$ and $\hat{y}_i$ are independent.
- $\mathbb{E}(F_1) = -2\sum_i^N cov(y_i, \hat{y}_i)$.

$$cov(\hat{\mathbf{y}}, \mathbf{y}) = cov((\mathbf{X}(\mathbf{X^TX})^{-1}\mathbf{X^T})\mathbf{y}, \mathbf{y}) = (\mathbf{X}(\mathbf{X^TX})^{-1}\mathbf{X^T})cov(\mathbf{y}, \mathbf{y})$$
$$cov(y, y) = \sigma_\epsilon^2.$$

- The values $cov(\hat{y}_i, y_i)$ are the diagonal values of the above matrix $cov(\hat{\mathbf{y}}, \mathbf{y})$. Thus

$$\sum_{i=1}^{N} cov(\hat{y}_i, y_i) = trace(\mathbf{X}(\mathbf{X^TX})^{-1}(\mathbf{X}^T))\sigma_\epsilon^2$$
$$= trace((\mathbf{X^TX})^{-1}(\mathbf{X}^T\mathbf{X}))\sigma_\epsilon^2$$
$$= trace(I_d)\sigma_\epsilon^2 = d\sigma_\epsilon^2.$$

- Similarly,

$$\sum_{i=1}^{N} cov(\mathbf{S}_\lambda y_i, y_i) = trace(\mathbf{S}_\lambda).$$

https://www.waxworksmath.com/Authors/G_M/Hastie/WriteUp/Weatherwax_Epstein_Hastie_Solution_Manual.pdf

# AIC Akaike Information Criterion

## Definition (AIC)

The **AIC Akaike Information Criterion** is defined

- Logistic regression, binomial log likelihood

$$AIC = -\frac{2}{N}loglik + 2\frac{d}{N}.$$

- Gaussian model with variance $\sigma_\epsilon^2$

$$AIC(\lambda) = \overline{err(\lambda)} + 2\frac{d(\lambda)}{N}\sigma_\epsilon^2.$$

the sum of the training error $\overline{err(\lambda)}$ and the complexity penalty for $d$, $d(\lambda)$ model parameters, $N$ samples, $loglik$ the logarithm of likelihood.

- The effective number of parameters
  - for a linear fit $\hat{\mathbf{y}} = \mathbf{Sy}$ is $d(\mathbf{S}) = trace(\mathbf{S})$, the sum of the diagonal elements.
- For 0-1 loss does not hold in general, only as approximation.

# BIC Bayesian Information Criterion

$$
\begin{aligned}
P(\mathcal{M}_m | \mathbf{Z}) &= \frac{P(\mathbf{Z}|\mathcal{M}_m) \cdot P(\mathcal{M}_m)}{P(\mathbf{Z})} \\
&\propto P(\mathbf{Z}|\mathcal{M}_m) \cdot P(\mathcal{M}_m) \\
&\propto P(\mathcal{M}_m) \cdot \int P(\mathbf{Z}|\theta_m, \mathcal{M}_m) P(\theta_m|\mathcal{M}_m) d\theta_m
\end{aligned}
$$

Laplace approximation to the integral gives with $\hat{\theta}_m$ the ML estimate of $\theta$:

$$
\begin{aligned}
\log P(\mathbf{Z}|\mathcal{M}_m) &= \log P(\mathbf{Z}|\hat{\theta}_m, \mathcal{M}_m) - \frac{d_m}{2} \cdot \log N + O(1) \\
&= loglik - \frac{d_m}{2} \cdot \log N + O(1)
\end{aligned}
$$

---

**Definition (Bayesian Information Criterion (BIC))**

$$
BIC_m = -2 loglik_m + (\log N) \cdot d_m
$$

BIC may be used to compare the model posterior probabilities $\frac{e^{\frac{1}{2} \cdot BIC_m}}{\sum_{\ell=1}^{M} e^{\frac{1}{2} \cdot BIC_\ell}}$.

# Crossvalidation

## Crossvalidation

- Split the data into $K$ roughly equal-sized parts. Usually, $K = 5$ or $10$ or $K = N$
    - For $K = 10$ it is called **tenfold** crossvalidation.
    - For $K = N$ it is called **one-leave-out** crossvalidation.
    - $\kappa : \{1, \ldots, N\} \to \{1, \ldots, K\}$ is the partition function
    - $\hat{f}^{-k}$ is the fitted function with $k$th part removed.
- For $k = 1, \ldots, K$
    - For the $k$th part we fit the model to the other $K - 1$ parts of the data, and calculate the prediction error of the fitted model when predicting the $k$th part of the data.
- Average the error estimates.

$$CV = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i)).$$

10 times 10 Crossvalidation

# Parameter Tuning by Crossvalidation

## Parameter Tuning by Crossvalidation

- Given a set of models $f(x, \alpha)$ indexed by a tuning parameter $\alpha$, we define

$$CV(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i, \alpha)).$$

- The $CV(\alpha)$ provides an estimate of the test error curve and we find the tuning parameter $\hat{\alpha}$ that minimizes it.
- Our final model is $f(x, \hat{\alpha})$.
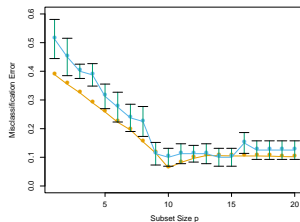


**FIGURE 7.9.** *Prediction error (orange) and tenfold cross-validation curve (blue) estimated from a single training set, from the scenario in the bottom right panel of Figure 7.3.*

## Definition (One standard error rule)

We choose the most parsimonious model whose error is no more than one standard error above the error of the best model.

Here, $p = 9$.

# Generalized Crossvalidation

- For a linear fitting method we can write

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}.$$

- For many linear fitting methods,

$$\frac{1}{N}\sum_{i=1}^{N}[y_i - \hat{f}^{-i}(x_i)]^2 = \frac{1}{N}\sum_{i=1}^{N}[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}]^2,$$

- The **generalized crossvalidation** *GCV* approximation is

$$GCV = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - trace(\mathbf{S})/N}\right]^2.$$

# Bad and good Crossvalidation

Our data

- $p = 500$ dimensions
- $N = 20$
- random data, $y$ independent of $x$.

Wrong way of the crossvalidation

1. choose 100 good predictors
2. Using just this subset of predictors, build a multivariate classifier, using all of the samples except those in fold $k$.
3. Use the classifier to predict the class labels for the samples in fold $k$.

- This approach achieves the crossvalidation error 3%.
- What is the true error?
- What is the correct crossvalidation approach?



**Wrong way**

Correlations of Selected Predictors with Outcome



**Right way**

Correlations of Selected Predictors with Outcome
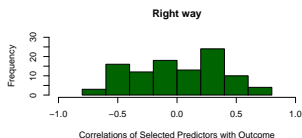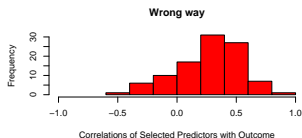
# Best Decision Stump Example

- Left top: Best decision stumps on full data

- 1/5 and 4/5 split

- bottom left: The stump fitted on 4/5 produces two missclassified validation data (blue)

- The crossvalidation error in 50 experiments is around 0.5, as it should be.
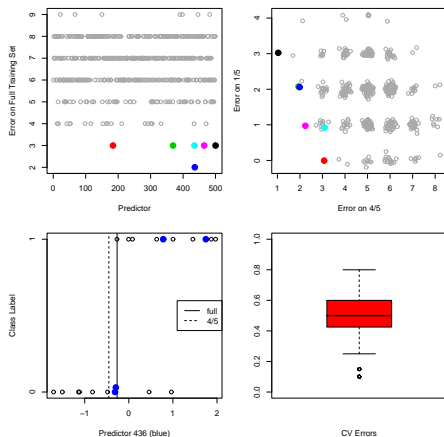
# One–leave–out

- Use all but one data samples for learning.
- Evaluate the error or the hidden sample.
- Repeat for each sample and calculate the average.

Advantage:

- The largest possible training set.
- Deterministic evaluation (no sense to repeat it).

Disadvantage:

- Time consuming.
- May be misleading – take randomly $50 : 50$ generated goal $G$, the one–leave–out error is 100%.

# Learning curve

- The learning curve suggests whether additional data would improve our model:
- curve flat close to maximal data size: small expected improvement.
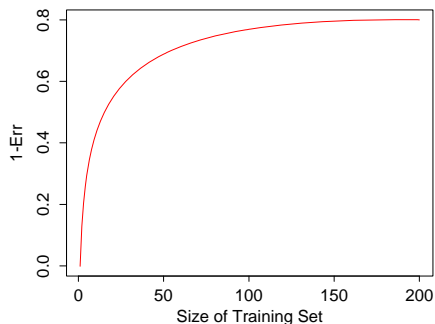
# Bootstrap

- Select elements with replacement.
- We have $N$ data samples, we select with replacement $N$ samples – some are selected more than one, some are not selected at all. *The not selected are used for testing.*
- The probability of not-selecting a sample is $\left(1 - \frac{1}{N}\right)^N \approx e^{-1} = 0.368$.
- The error estimate is pessimistic since we learn a model on $N$ samples that come from only 0.632 samples.
  The usual error estimate is:

$$err = 0.632 \cdot e_{\text{test}} + 0.368 \cdot e_{\text{train}}$$

- Again, may be misled by similar data as one–leave–out.
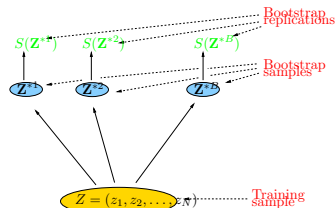


**FIGURE 7.12.** *Schematic of the bootstrap process. We wish to assess the statistical accuracy of a quantity $S(\mathbf{Z})$ computed from our dataset. $B$ training sets $\mathbf{Z}^{*b}$, $b = 1, \dots, B$ each of size $N$ are drawn with replacement from the original dataset. The quantity of interest $S(\mathbf{Z})$ is computed from each bootstrap training set, and the values $S(\mathbf{Z}^{*1}), \dots, S(\mathbf{Z}^{*B})$ are used to assess the statistical accuracy of $S(\mathbf{Z})$.*

# Confusion Matrix

| true class \ prediction | + | - |
|---|---|---|
| + | TP – true positive | FN – false negative |
| - | FP – false positive | TN true negative |

Česky se říká správně/falešně positivní/negativní.

Basic measures:

| celková správnost | accurancy | $Acc = \frac{TP+TN}{TP+TN+FP+FN}$ |
|---|---|---|
| chyba | error | $Err = \frac{FP+FN}{TP+TN+FP+FN}$ |
| přesnost | precision | $Prec = \frac{TP}{TP+FP}$ |
| úplnost, sensitivita | recall, sensitivity | $Rec = \frac{TP}{TP+FN}$ |
| specificita | specificity | $Specificity = \frac{TN}{TN+FP}$ |
| F míra | F measure | $F = \frac{2 \cdot Prec \cdot Rec}{Prec+Rec} = \frac{2 \cdot TP}{2 \cdot TP+FP+FN}$ |
| | TP rate | $\frac{TP}{TP+FN}$ |
| | FP rate (=1–Specificity) | $\frac{FP}{FP+TN}$ |

# Additional Remarks

- ROC - curve
  - Precision recall curve
- Interval estimates
- t-test
  - paired t-test
- ANOVA
- Q-Q plot
- $R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}$, 1 or 0 instead of nan and infinity.
  - explained variance - identical for zero mean residuals.
- Mc Nemar's test (two classifiers, confusion matrix with values $<5$; i.e. not $\chi^2$ test).
  - statsmodels.stats.contingency_tables.mcnemar()

The cost of missclassification may be different for each class. The general loss specification is a **loss matrix** $L_{kk'}$, an element represent the cost of classifying $k$ as $k'$. Must be zero at the diagonal, non-negative everywhere.

- we can modify
  $Gini(m) = \sum_{k \neq k'} L_{kk'} \hat{p}_{mk} \hat{p}_{mk'}$
- or weight the data samples $k$ $L_{kk'}$ times (only in binary classification)
- we classify according to
  $k(m) = argmin_k \sum_l L_{lk} \hat{p}_{ml}$ in the leaves.
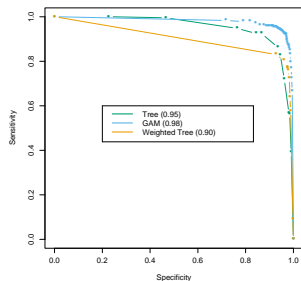


**FIGURE 9.6.** *ROC curves for the classification rules fit to the* spam *data. Curves that are closer to the northeast corner represent better classifiers. In this case the GAM classifier dominates the trees. The weighted tree achieves better sensitivity for higher specificity than the unweighted tree. The numbers in the legend represent the area under the curve.*

# Bias Variance Decomposition

- Assume $Y = f(X) + \epsilon$ where $\mathbb{E}(\epsilon) = 0$ and $Var(\epsilon) = \sigma_\epsilon^2$.
- we derive the expected prediction error of the regression fit $\hat{f}(X)$ at an input point $X = x_0$ using squared-error loss:

$$
\begin{aligned}
Err(x_0) &= \mathbb{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [\mathbb{E}\hat{f}(x_0) - f(x_0)]^2 + \mathbb{E}[\hat{f}(x_0) - \mathbb{E}\hat{f}(x_0)]^2 \\
&= \sigma_\epsilon^2 + Bias^2(\hat{f}(x_0)) + Var(\hat{f}(x_0)) \\
&= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}.
\end{aligned}
$$

# Bias Variance Decomposition

K-Nearest neighbour

$$
\begin{aligned}
Err(x_0) &= \mathbb{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [\mathbb{E}\hat{f}(x_0) - f(x_0)]^2 + \mathbb{E}[\hat{f}(x_0) - \mathbb{E}\hat{f}(x_0)]^2 \\
&= \sigma_\epsilon^2 + [f(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})]^2 + \frac{\sigma_\epsilon^2}{k}.
\end{aligned}
$$

Linear fit

$$
\begin{aligned}
Err(x_0) &= \mathbb{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [\mathbb{E}\hat{f}_p(x_0) - f(x_0)]^2 + \|\mathbf{h}(x_0)\|\sigma_\epsilon^2 \\
\mathbf{h}(x_0)\mathbf{y} &= (x_0^T(\mathbf{X^T X})^{-1}\mathbf{X^T})\mathbf{y} = \hat{f}_p(x_0).
\end{aligned}
$$

- hence $Var[\hat{f}_p(x_0)] = \|\mathbf{h}(x_0)\|\sigma_\epsilon^2$ and its average is $\frac{p}{N}\sigma_\epsilon^2$, hence

$$
\frac{1}{N}\sum_{i=1}^{N} Err(x_i) = \sigma_\epsilon^2 + \frac{1}{N}\sum_{i=1}^{N}[f(x_i) - \mathbb{E}\hat{f}(x_i)]^2 + \frac{p}{N}\sigma_\epsilon^2,
$$

# Penalized model complexity

- Consider ridge regression fit $\hat{f}_\alpha(x_0)$ with the fit

$$\mathbf{h}(x_0)\mathbf{y} = x_0^T(\mathbf{X^T X} + \alpha I)^{-1}\mathbf{X^T y}$$

- we break the bias more finely. Let $\beta_*$ be the fit with $\alpha = 0$,
$\beta_* = \arg\min_\beta \mathbb{E}(f(X) - \beta^T X)^2$.

$$
\begin{aligned}
\mathbb{E}_{x_0}[f(x_0) - \mathbb{E}\hat{f}_\alpha(x_0)]^2 &= \mathbb{E}_{x_0}[f(x_0) - \beta_*^T x_0]^2 + \mathbb{E}_{x_0}[\beta_*^T x_0 - \mathbb{E}\hat{\beta}_\alpha^T x_0]^2 \\
&= Ave[\text{Model Bias}]^2 + Ave[\text{Estimation Bias}]^2.
\end{aligned}
$$

# Example: Bias-Variance Tradeoff

- 50 observations, 20 predictors, uniformly distributed in the hypercube $[0, 1]^{20}$.

Left $Y$ is 0 if $X_1 \leq \frac{1}{2}$ and 1 if $X_1 > \frac{1}{2}$ and we apply $k$-NN

Right $Y$ is 1 if $\sum_{j=1}^{10} X_j > 5$ and 0 otherwise, and we use best subset linear regression of size $p$.



Fig. 7.3.

# Optimism of the Training Error Rate

- **training error** $\overline{err} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$
- usually less than the true error $Err = \mathbb{E}[L(Y, \hat{f}(X))]$.
- we keep $x_i$ points fixed, we take new sample of **y** at these points.
- **in-sample error** $Err_{in} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathbf{y}} \mathbb{E}_{Y^{new}} L(Y_i^{new}, \hat{f}(x_i))$.
- **optimism** $op \leftarrow Err_{in} - \mathbb{E}_{\mathbf{y}}(\overline{err})$.
- for squared error, 0-1, and other loss functions: $op = \frac{2}{N} \sum_{i=1}^{N} cov(\hat{y}_i, y_i)$
- For a linear fit with $d = p$ inputs of basis functions and additive error model $Y = f(X) + \epsilon$ it simplifies

$$\sum_{i=1}^{N} cov(\hat{y}_i, y_i) = d\sigma_\epsilon^2$$

$$Err_{in} = \mathbb{E}_{\mathbf{y}} \overline{err} + 2\frac{d}{N}\sigma_\epsilon^2 = C_p = AIC_{gauss.}.$$

# The Optimism (Just for fun)

$$y_i - \hat{y}_i = (y_i - f(x_i)) + (f(x_i) - \mathbb{E}\hat{f}(x_i)) + (\mathbb{E}\hat{f}(x_i) - \hat{y}_i).$$

- We get six terms of the sum of squares:

$A_1 = \sum_i (y_i - f(x_i))^2 \qquad D_1 = 2\sum_i (y_i - f(x_i))(f(x_i) - \mathbb{E}\hat{f}(x_i))$

$B = \sum_i (f(x_i) - \mathbb{E}\hat{f}(x_i))^2 \qquad E = 2\sum_i (f(x_i) - \mathbb{E}\hat{f}(x_i))(\mathbb{E}\hat{f}(x_i) - \hat{y}_i)$

$C = \sum_i (\mathbb{E}\hat{f}(x_i) - \hat{y}_i)^2 \qquad F_1 = 2\sum_i (y_i - f(x_i))(\mathbb{E}\hat{f}(x_i) - \hat{y}_i)$

- For the new sample $Y$, taking the expectation.
- $A_2 = \sum_i \mathbb{E}_{Y^0}(Y_i^0 - f(x_i))^2$
- and similarly $D_2, F_2$.

$$N(Err_{in} - \overline{err}) = (A_2 + B + C + D_2 + E + F_2) - (A_1 + B + C + D_1 + E + F_1)$$
$$= (A_2 - A_1) + (D_2 - D_1) + (F_2 - F_1)$$

- $\mathbb{E}(A_1) = \mathbb{E}(A_2) = N\sigma_\epsilon^2$,
- $\mathbb{E}(D_1) = 2\sum_i (\mathbb{E}(y_i) - f(x_i))(f(x_i) - \mathbb{E}\hat{f}(x_i)) = 0$ since $\mathbb{E}(y_i) = f(x_i)$
- $\mathbb{E}(D_2) = 0$ as well.
- $F_2 = 2\sum_i (\mathbb{E}_{Y^0}\left[(Y_i^0 - f(x_i))(\mathbb{E}\hat{f}(x_i) - \hat{y}_i)\right] = 0$
  - as $\mathbb{E}(Y^0) = f(x_i)$ and $Y_i^0$ and $\hat{y}_i$ are independent.
- $\mathbb{E}(F_1) = -2\sum_i^N cov(y_i, \hat{y}_i)$.

# The Covariance (Just for fun)

$$
\begin{aligned}
cov(\hat{\mathbf{y}}, \mathbf{y}) &= cov((\mathbf{X}(\mathbf{X^T X})^{-1}\mathbf{X^T})\mathbf{y}, \mathbf{y}) = (\mathbf{X}(\mathbf{X^T X})^{-1}\mathbf{X^T})cov(\mathbf{y}, \mathbf{y}) \\
cov(y, y) &= \sigma_\epsilon^2.
\end{aligned}
$$

- The values $cov(\hat{y}_i, y_i)$ are the diagonal values of the above matrix $cov(\hat{\mathbf{y}}, \mathbf{y})$. Thus

$$
\begin{aligned}
\sum_{i=1}^{N} cov(\hat{y}_i, y_i) &= trace(\mathbf{X}(\mathbf{X^T X})^{-1}(\mathbf{X}^T))\sigma_\epsilon^2 \\
&= trace((\mathbf{X^T X})^{-1}(\mathbf{X}^T \mathbf{X}))\sigma_\epsilon^2 \\
&= trace(I_d)\sigma_\epsilon^2 = d\sigma_\epsilon^2.
\end{aligned}
$$

- Similarly,

$$
\sum_{i=1}^{N} cov(\mathbf{S}_\lambda y_i, y_i) = trace(\mathbf{S}_\lambda).
$$

# Additive Models, Trees, and Related Methods

- **Generalized additive models (GAMs)** are automatic flexible statistical methods that may be used to identify and characterize nonlinear regression effects.

- GAM has the form

$$\mathbb{E}(Y|X_1, \ldots, X_p) = \alpha + f_1(X_1) + \ldots + f_p(X_p)$$

  - where $f_j$'s are unspecified smooth functions
  - $X_j$ predictors, $Y$ the outcome.

- We use a cubic smoothing spline, local polynomial regression or a kernel smoother

- **we simultaneously estimate all $p$ functions**.

# GAM for non Gaussian distributions

- Denote $\mu(X) = P(Y = 1|X)$ in a two class classification with 0-1 encoding and recall the logistic regression

$$\log\left(\frac{\mu(X)}{1 - \mu(X)}\right) = \alpha + \beta_1 X_1 + \ldots + \beta_p X_p,$$

- **Additive logistic regression** model replaces the linear terms by the smoothers

$$\log\left(\frac{\mu(X)}{1 - \mu(X)}\right) = \alpha + f_1(X_1) + \ldots + f_p(X_p),$$

- The conditional mean $\mu(X)$ of a response $Y$ is related to an additive function of the predictors via a **link function** $g$

$$g[\mu(X)] = \alpha + f_1(X_1) + \ldots + f_p(X_p).$$

- Examples of classical link functions are the following
  - $g(\mu) = \mu$ the identity link, used for linear and additive models for Gaussian response data.
  - $g(\mu) = logit(\mu)$ as above
  - $g(\mu) = probit(\mu)$ **probit** link function for modeling binomial probabilities is the inverse of Gaussian cumulative distribution function $probit(\mu) = \Phi^{-1}(\mu)$
  - $g(\mu) = \log(\mu)$ for log-linear or **log-additive models** for Poisson count data.

# Models with Feature Interactions

- The categorical variables are usually treated like identifiers (0-1 or -1,1)
- in the logistic regression, it leads to a 'constant' $\beta_j$ fitted for the variable
- the slope $\beta_{-j}$ of others does not depend on the identifier
- We can extend to a **semiparametric model**, that keeps the effect of the $k$th predictor and the effect of the predictor $Z$ additive
$$g(\mu) = X^T \beta + \alpha_k + f(Z).$$
- To allow different slopes/shapes of $Z$ based on qualitative variable $V$ we need an interaction term of two features
$$g(\mu) = f(X) + g_k(Z)$$
- Generally, we may add a function $g_{ZW}(Z, W)$ of two or more features
$$g(\mu) = f(X) + g_{ZW}(Z, W).$$
- Note: logit, probit, log, gamma and negative-binomial distributions belong the an exponential family, therefore have some nice properties (fit together).

# Fitting Additive Model

1: **procedure** GENERALIZED ADDITIVE MODEL FITTING:($\mathbf{X}$, $\mathbf{y}$)

2: $\quad \hat{\alpha} = \frac{1}{N} \sum_1^N y_i$, $\hat{f}_j \equiv 0$ initialize $\forall i, j$.

3: $\quad$ **repeat** for $j = 1, 2, \ldots, p, \ldots, 1, 2, \ldots$

4: $\quad\quad \hat{f}_j \leftarrow \mathcal{S}_j \left[ \{ y_i - \hat{\alpha} - \sum_{j \neq k} \hat{f}_k(x_{ij}) \}_1^N \right]$,

5: $\quad\quad \hat{f}_j \leftarrow \hat{f}_j - \sum_{i=1}^N \hat{f}_j(x_{ij})$.

6: $\quad$ **until** the functions $\hat{f}_j$ change less than a prespecified threshold.

7: **end procedure**

- $\mathcal{S}_j$ denotes the smoother, for example the smoothing spline with predefined degrees of freedom.
- All $\hat{f}_j$ should have zero mean, the constant is fitted by $\alpha$.
- Re-normalization is recommended because of rounding errors.

# Generalized Additive Logistic Regression

1: **procedure** ADDITIVE LOGISTIC REGRESSION:($\mathbf{X}$, $\mathbf{y}$ in 0-1 encoding)

2:     $\hat{y} = \frac{1}{N} \sum_1^N y_i, \hat{\alpha} = \log(\frac{\hat{y}}{1-\hat{y}}), \hat{f}_j \equiv 0$ initialize $\forall j$.

3:     $\hat{\eta}_i = \hat{\alpha} + \sum_j \hat{f}_j(x_{ij})$ and $\hat{p}_i = \frac{1}{1+\exp(-\hat{\eta}_i)}$

4:     **repeat**

5:         Construct the working target variable

$$z_i = \hat{\eta}_i + \frac{y_i - \hat{p}_i}{\hat{p}_i(1 - \hat{p}_i)}.$$

6:         Construct the weight $w_i = \hat{p}_i(1 - \hat{p}_i)$.

7:         Fit an additive model to the targets $z_i$ with weights $w_i$, using a weighted backfitting algorithm. This gives new estimates $\hat{\eta}_j, \hat{f}_j \; \forall j$

8:     **until** the functions change less than a prespecified threshold.

9: **end procedure**

# Spam Example

- Email classification as email/spam.
- word frequency as $X$ features.
- Important features:



**TABLE 9.2.** *Significant predictors from the additive model fit to the spam training data. The coefficients represent the linear part of $\hat{f}_j$, along with their standard errors and Z-score. The nonlinear P-value is for a test of nonlinearity of $\hat{f}_j$.*

| Name | Num. | df | Coefficient | Std. Error | Z Score | Nonlinear P-value |
|------|------|-----|-------------|------------|---------|-------------------|
| *Positive effects* | | | | | | |
| our | 5 | 3.9 | 0.566 | 0.114 | 4.970 | 0.052 |
| over | 6 | 3.9 | 0.244 | 0.195 | 1.249 | 0.004 |
| remove | 7 | 4.0 | 0.949 | 0.183 | 5.201 | 0.093 |
| internet | 8 | 4.0 | 0.524 | 0.176 | 2.974 | 0.028 |
| free | 16 | 3.9 | 0.507 | 0.127 | 4.010 | 0.065 |
| business | 17 | 3.8 | 0.779 | 0.186 | 4.179 | 0.194 |
| hp1 | 26 | 3.8 | 0.045 | 0.250 | 0.181 | 0.002 |
| ch! | 52 | 4.0 | 0.674 | 0.128 | 5.283 | 0.164 |
| ch$ | 53 | 3.9 | 1.419 | 0.280 | 5.062 | 0.354 |
| CAPMAX | 56 | 3.8 | 0.247 | 0.228 | 1.080 | 0.000 |
| CAPTOT | 57 | 4.0 | 0.755 | 0.165 | 4.566 | 0.063 |
| *Negative effects* | | | | | | |
| hp | 25 | 3.9 | −1.404 | 0.224 | −6.262 | 0.140 |
| george | 27 | 3.7 | −5.003 | 0.744 | −6.722 | 0.045 |
| 1999 | 37 | 3.8 | −0.672 | 0.191 | −3.512 | 0.011 |
| re | 45 | 3.9 | −0.620 | 0.133 | −4.649 | 0.597 |
| edu | 46 | 4.0 | −1.183 | 0.209 | −5.647 | 0.000 |
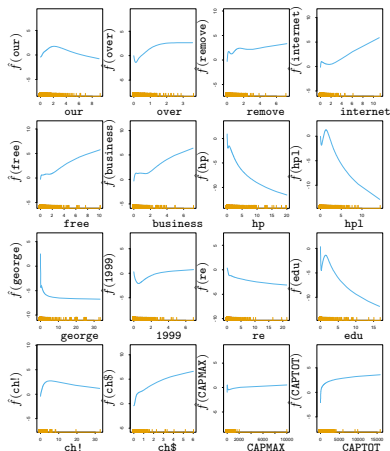
# Decision Trees

**Decision tree** for a given goal attribute $G$ is a rooted tree with

- a root and inner nodes labeled by attributes; for each possible value of the attribute there is an outgoing edge from the node;

- leaves are labeled with predicted goal class $g \in G$ assuming other attributes have values as labeled on the path from the root.

Attributes not present on the path from the root to the leaf are irrelevant.
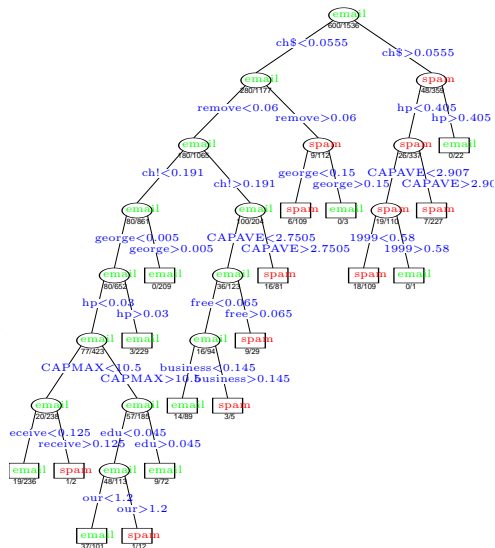
**Tree construction** idea:

- select an attribute; create a node and split the data according the value of the attribute
- for each attribute value construct a subtree based on the appropriate part of the data
- stop if there is a unique value of the goal $G$ in the data or no attributes to split, create a leaf labeled by the most common class $g \in G$.

The criterion to select an attribute follows.

# Entropy

The entropy of an attribute $A$ ('uncertainty', negative information) we would like:

- to be zero for the pure data (only one value of the attribute $A$)
- the highest entropy for uniform distribution on values of $A$ (no information at all)
- two step split leads to the same result as split at once:

$$H([2, 3, 4]) = H([2, 7]) + \frac{7}{9} \cdot H([3, 4])$$

---

### Definition

Entropy These properties has the **entropy** $H([p_1, \ldots, p_k]) = -\sum_{i=1}^{k} p_i \log p_i$, the base of the logarithm usually $e$, sometimes 2.

If we do not normalize we get the entropy multiplied by $\sum_{i=1}^{k} p_i$.

---

The lower index $A$ denotes the attribute to calculate the entropy $H_A$, for the goal attribute $H_G$.

# The Entropy for a binary attribute

x axis: $p_i$, y axis: entropy.

$Gini = 1 - \sum_i (p_i)^2$
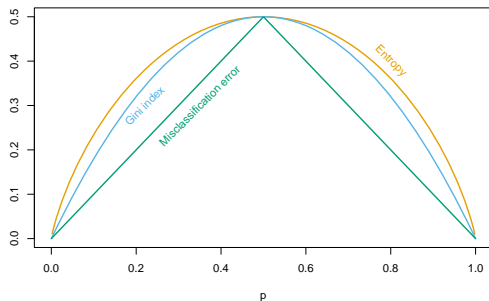


**FIGURE 9.3.** *Node impurity measures for two-class classification, as a function of the proportion $p$ in class 2. Cross-entropy has been scaled to pass through $(0.5, 0.5)$.*

# ID3 algorithm

We select an attribute with the maximal **information gain**, defined for the *data* and an attribute $X_j$:

$$Gain(data, X_j) = H_G(data) - \sum_{x_j \in X_j} \frac{|data_{X_j = x_j}|}{|data|} H_G(data_{X_j = x_j})$$

where $data_{X_j = x_j}$ is a subset of *data* where $X_j = x_j$, the entropy is defined

$$H_G(data) = \sum_{g \in G} -\frac{|data_{G=g}|}{|data|} \cdot log_2 \frac{|data_{G=g}|}{|data|} = \sum_{i=1}^{|G|} -p_i \cdot log_2 p_i$$

where $p_i$ denotes the ratio of $G = g_i$ in the *data*.

It is equivalent to minimize the weighted entropy after the split, that is

$$\arg \min_{X_i} \sum_{x_j \in X_j} \frac{|data_{X_j = x_j}|}{|data|} \sum_{g \in G} -\frac{|data_{G=g \& X_j = x_j}|}{|data_{X_j = x_j}|} \cdot log_2 \frac{|data_{G=g \& X_j = x_j}|}{|data_{X_j = x_j}|}$$

# ID3 algorithm($data$, $G$ goal, $Attributes$ attributes)

## ID3

Create the root $root$

If all data have the same $g$, label the root $g$ and return,

If no attributes $Attributes$, label the $root$

             by the most frequent $g$ in the $data$ and return

otherwise

        $X_j \leftarrow$ the attribute from $Attributes$ with the maximal $Gain(data, X_j)$

        label $root$ as $X_j$

        for each possible value $x_j$ of $X_j$,

              add an edge from $root$ labeled $X_j = x_j$

              $data_{X_j = x_j} \leftarrow$ the subset of the $data$ with $X_j = x_j$

              If $data_{X_j = x_j}$ is empty, add a leaf labeled by

                    the most common class $g$ in $data$ and return

              add a subtree $ID3(data_{X_j = x_j}, G, Attributes \setminus \{X_j\})$

return $root$

# Categorical Attribute Notes

- CART in the sklearn DecisionTreeClassifier does not support categorical attributes
  - uses just binary splits.
- It is requires exponential complexity with respect to the number of categories to find optimal binary split.
  - The recommended heuristic is to sort categories according to the goal class probabilities and search the split in a linear time.
- We should avoid the split into too many branches.
  - ID3 used penalization $Gain^*(X_i, data) = \frac{Gain(X_i, data)}{H(X_i)}$
  - so for the identifier with unique values $Gain^*(X_i, data) = \frac{Gain(X_i, data)}{\log N}$.
  - min_samples_split, min_samples_leaf, min_weight_fraction_leaf can do it too.

# Prunning Introduction

To avoid overfitting we try to remove unnecessary nodes

- **postprunning** – build a tree, prune afterwards;
  - usuall way
- **preprunning** – prune during the construction
  - This seems nice but we could prune two attributes combined by XOR since both has information gain (close to) zero.

Postprunning

- **subtree replacement** – select a tree and replace it by a leaf;
  - it increases the training error
  - it may decrease the error on validation data
  - step by step, we try to prune each subtree: we prune if we do not increase validation error.
- **subtree raising** – remove an inner node. Used in C4.5. The data samples must be re-send to the remaining branch, it is time consuming.
  - Usually checked only for the most frequent branch in the tree.

# Reduced Error Pruning

## Reduced Error Pruning

- **reduced error pruning** – we keep part of the data for validation (pruning).
- for each inner node compare

    - validation error with this node as a leaf
    - validation error with the (pruned) subtree of this node

- select whatever gives the lower error.

- there exist also a criterion based on the training data
- **Reduced Cost Pruning** CART - few slides later.
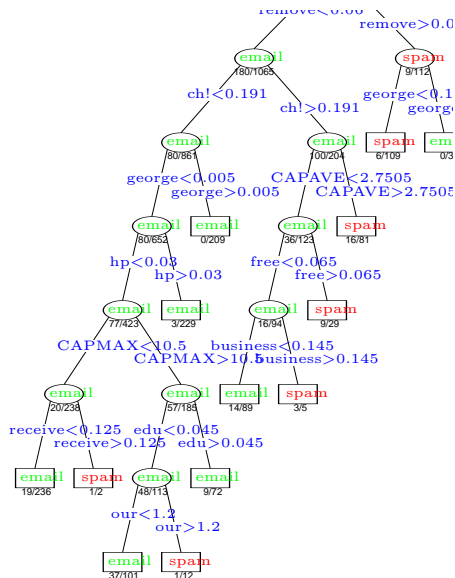
| 64 | 65 | 68 | 69 | 70 | 71 | 72 | 75 | 80 | 81 | 83 | 85 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| yes | no | yes | yes | yes | no | no,yes | yes,yes | no | yes | yes | no |

- we require a binary split
- 11 split points
- for each split we calculate the information gain
  -

$$H([9,5]) - H([4,2],[5,3]) = H([9,5]) - (\frac{6}{14} \cdot H([4,2]) + \frac{8}{14} \cdot H([5,3]))$$

$$= 0.940 - 0.939 \text{ bits.}$$

- We allow multiple splits based on this attribute.

# Regression trees - numerical prediction

- **Model tree** has linear fit in the leaves
  - not so popular as regression trees; increases complexity and discontunuity
- **CART**
  - use the decrease of the square error loss to select an attribute
  - binary splits
  - predict the average value in the leaves.
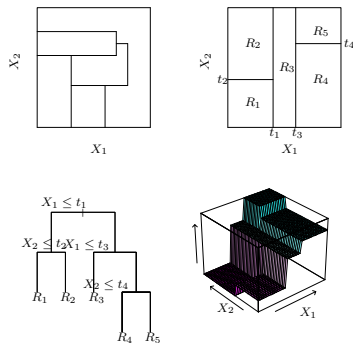


**FIGURE 9.2.** *Partitions and CART. Top right panel shows a partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the tree corresponding to the partition in the top right panel, and a perspective plot of the prediction surface appears in the bottom right*

# CART (Classification and) Regression Trees

- Regions $R_m$, we predict a constant $c_m$ inside any region.

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i.$$

## Single Regression Tree for CART

- Start with all data in one region $R_0$
- Select the best attribute $j$ and its value $s$ for the split:

$$\min_{j,s}[min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$$

   - Inner minimum is the average $\hat{c}_1 = ave(y_i | x_i \in R_1)$.
- iterate until stop (number of samples in the leaf $\leq n_0$).

## Reduced Cost Pruning

- Split the data into $K$ folds
- For each fold $k$:
  - use all except fold $k$ to train the tree $T$
  - Build a sequence of subtrees $T^k \supset T_1^k \supset T_2^k \ldots \supset T_{|T|}^k$
    - always join two leaves with the minimal increase in the training error
  - use fold $k$ do calculate the crossvalidation error of each tree
  - consider the error function $C_\alpha(T^k)$ as a function of $\alpha$
- Select $\alpha \leftarrow argmin_\alpha \sum_k C_\alpha(T^k)$
- Build a tree on the full training data
- Return the subree corresponding to the optimal $\alpha$.

- Average error on a leaf $m$

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \frac{1}{N_m} \sum_{x_i \in R_m} y_i)^2,$$

- Cost of the tree with $\alpha$ penalty for the number of leaves

$$C_\alpha(T) \quad = \quad \sum^{|T|} N_m Q_m(T) + \alpha |T|.$$

# Missing values, Class and Samples Weights

- Trees can handle missing data well.
- Often we cannot **omit** missing data since many samples have missing values.
  - Furthermore, missing values in unused attributes are irrelevant.
- If the value is **not missing at random** then take the missingness as another value of the attribute
  - example: very small and very high wages are more ofter missing
- If the data are **missing at random**
  - **split the instance**
  - according the data ratio following each branch
  - weight and average the predictions on leaves.
- Similarly, we use weighted information gain to select the attribute.
  - by setting setting class_weight
  - fit(X, y, sample_weight=None).

# Complexity considerations

## CART

- Let us have $N$ instances with $p$ attributes.
- Assume a reasonably balanced tree with the tree depth $O(logN)$.
- To build the tree we need $O(p \cdot N^2 \cdot logN)$ time.
  - At each depth, each instance occurs exactly ones, $logN$ depth levels, $p$ attributes on each level, the time $O(p \cdot N^2 \cdot \log N)$.
- Subtree replacement $O(N)$, Subtree raising $O(N(logN)^2)$.
- Naive tree construction comlpexity is $O(p \cdot N^2 \cdot logN) + O(N(logN)^2)$.
- With sorted features and clever indexing
  - Overall tree construction comlpexity is $O(p \cdot N \cdot logN) + O(N(logN)^2)$.

# MARS Multivariate Adaptive Regression Splines

- generalization of linear regression and decision trees CART
- for each feature and each data point we create a **reflected pair** of basis functions
- $(x - t)_+$ and $(t - x)_+$ where $+$ denotes non–negative part, minimum is zero.
- we have the set of functions

$$\mathcal{C} = \{(X_j - t)_+, (t - X_j)_+\}_{t \in \{x_{1,j}, x_{2,j}, \ldots, x_{N,j}\}, j=1,2,\ldots,p}$$

- that is $2Np$ functions for non–duplicated data points.
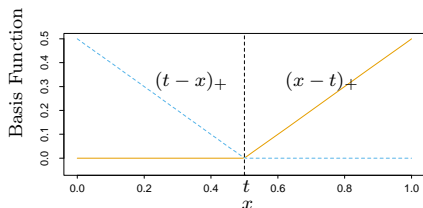
# MARS – continuation

- our model is in the form

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X)$$

where $h_m(X)$ is a function from $\mathcal{C}$ or a product of any amount of functions from $\mathcal{C}$

- for a fixed set of $h_m$'s we calculate coefficients $\beta_m$ by usual linear regression (minimizing RSS)

- the set of functions $h_m$ is selected iteratively.

# MARS – basis selections

- We start with $h_0 = 1$, we put this function into the model $\mathcal{M} = \{h_0\}$.

- We consider the product of any member $h_\ell \in \mathcal{M}$ with any pair from $\mathcal{C}$,

$$\hat{\beta}_{M+1} h_\ell(X) \cdot (X_j - t)_+ + \hat{\beta}_{M+2} h_\ell(X) \cdot (t - X_j)_+$$

  we select the one minimizing training error RSS (for any product candidate, we estimate $\hat{\beta}$).

- Repeat until predefined number of functions in $\mathcal{M}$

# MARS – model pruning

- The model is usually overfitted. We select (remove) iteratively the one minimizing the increase of training RSS. We have a sequence of models $\hat{f}_\lambda$ for different numbers of parameters $\lambda$.

- (we want to speed–up crossv-alidation for computational reasons)

- we select $\lambda$ (and the model) minimizing **generalized cross-validation**

$$GCV(\lambda) = \frac{\sum_{i=1}^{N}(y_i - \hat{f}_\lambda(x_i))^2}{(1 - M(\lambda)/N)^2}.$$

- where $M(\lambda)$ is the number of effective parameters, the number of function $h_m$ (denoted $r$) plus the number of knots $K$, the authors suggest to multiply $K$ by 3: $M(\lambda) = r + 3K$.

# Ensemble Methods

- Random forest ($+$ Bagging)
- Boosting
    - Adaboost - classification
    - Gradient boosting - regression and classification
- Stacking
- MARS ($=$earth).
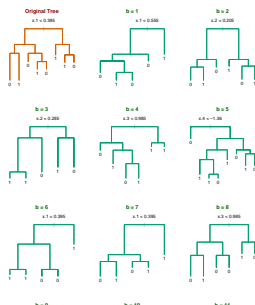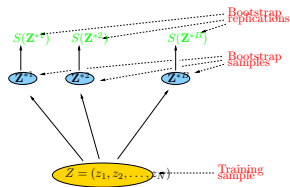
# Bootstrap

- Select elements with replacement.
- We have $N$ data samples, we select with replacement $N$ samples – some are selected more than one, some are not selected at all. *The not selected are used for testing.*
- The probability of not-selecting a sample is $\left(1 - \frac{1}{N}\right)^N \approx e^{-1} = 0.368$.
- Selected samples used to learn a model (usually a tree).
- These are used for the **OutOfBag** error computation.
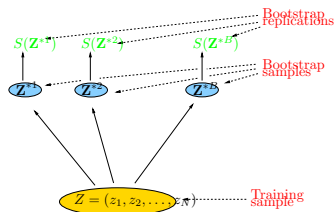


**FIGURE 7.12.** *Schematic of the bootstrap process. We wish to assess the statistical accuracy of a quantity $S(\mathbf{Z})$ computed from our dataset. B training sets $\mathbf{Z}^{*b}$, $b = 1, \ldots, B$ each of size N are drawn with replacement from the original dataset. The quantity of interest $S(\mathbf{Z})$ is computed from each bootstrap training set, and the values $S(\mathbf{Z}^{*1}), \ldots, S(\mathbf{Z}^{*B})$ are used to assess the statistical accuracy of $S(\mathbf{Z})$.*

## Random Forest for Regression or Classification

1: **procedure** RANDOM FOREST:( $X, y$ training data )
2:     **for**  $b = 1, 2, \ldots, B$ **do**
3:         Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$
4:         Grow a random forest tree $T_b$
5:         **repeat**
6:             Select $m$ variables at random from $p$ variables.
7:             Pick the best variable/split–point among the $m$
8:             Split the node into two daughter nodes.
9:         **until** the minimum node size $n_{min}$ is reached.
10:     **end for**
11:     Output the ensamble of trees $\{T_b\}_1^B$.
12: **end procedure**

To make a prediction at a new point $x$:

- Regression: $\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$.
- Classification: Let $\widehat{C}_b(x)$ be the class prediction of the $b$th random–forest tree.
    - Predict $\widehat{C}_{rf}^B(x) = $ *majority vote* $\{\widehat{C}_b(x)\}_1^B$.

# Bagging (Bootstrap aggregating)

- It is a Random Forest, where we use all predictors, that is $m = p$.
- both regression and classification.
- Training data $\mathbf{Z} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x).$$
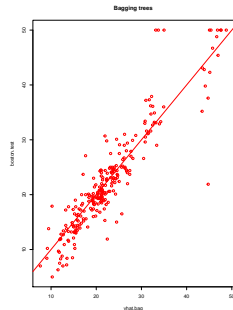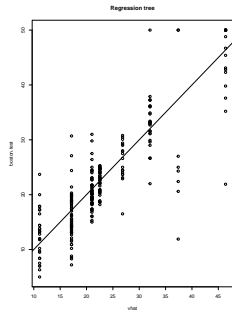
# Bagging for Classification

- Training data $\mathbf{Z} = \{(x_1, g_1), (x_2, g_2), \ldots, (x_N, g_N)\}$

- for each bootstrap sample, $b = 1, 2, \ldots, B$, we fit our model, giving prediction $\hat{f}^{*b}(x)$.

- Take either
  - predict probabilities of classes and find the class with the highest predicted probability over the bootstrap samples

  $$\hat{G}(x) = \text{argmax}_k \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

  - predict class and

  $$\hat{G}_{bag}(x) = \text{majority vote } \{\hat{G}^{*b}(x)\}_{b=1}^{B}.$$
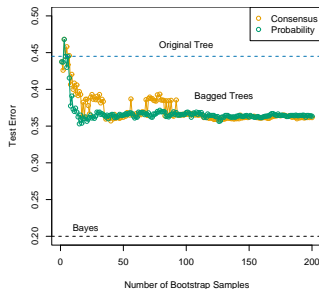
# Behind Random Forest

The variance of the random forest estimate $Var(\hat{f}_{rf}^{B}(x)) = \mathbb{E}(\hat{f}(x) - \mathbb{E}\hat{f}(x))^2$ is

- iid data variables, independent features, each with variance $\sigma^2$:
  - $\frac{1}{B}\sigma^2$
- id identically distributed data, each with variance $\sigma^2$ with positive pairwise correlation $\rho$:
  - $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$.
- The second part is addressed by bagging.
- The idea behind random random forest is to address the first part of the formula.
  - Before each split, select $m \leq p$ variables as candidates for splitting.
  - $m \leftarrow \sqrt{p}$ for regression, even as low as 1. $\frac{p}{3}$ for classification.
- Bagging does not change linear estimates, such as the sample mean
  - The pairwise correlation between bootstrapped means is about 50%.

# OOB Error

For each observation $z_i = (x_i, y_i)$, construct is random forest predictor by averaging only those trees corresponding to bootstrap samples in which $z_i$ did not appear.

- An OOB error estimate is almost identical to that obtained by $N$-fold crossvalidation.
- Unlike many other nonlinear estimators, random forests can be fit in one sequence.
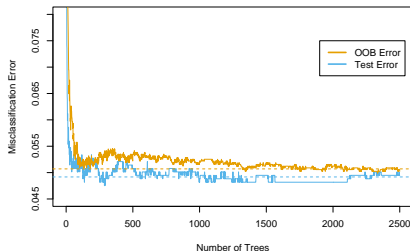
# Variable Importance (Gini, RSS)

- **Variable Importance** of a predictor $X_\ell$ in a single tree $T$ is

  $I_\ell^2(T) = \sum_{t=1}^{J} \hat{\imath}_t^2 \cdot I(v(t) = \ell)$

  - For each internal node $t$ of the tree, we calculate the *Gini* or *RSS* gain
  - where $\hat{\imath}_t^2$ is the Gini/RSS improvement of the predictor in the inner node $t$.
    - Gini $\hat{p}_k(t)(1 - \hat{p}_k(t))$ before and after the split
    - for $K$ goal classes, a separate tree for each class against others
    - weighted by the probability of reaching the node $t$.

- For a set of trees, we average over $M$ all trees

  $I_\ell^2 = \frac{1}{M} \sum_{i=1}^{M} I_\ell^2(T_m)$.

- Usually scaled to the interval $(0, 100)$.
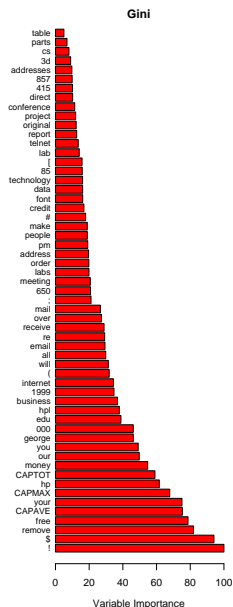
# OOB Variable Importance

## OOB Variable Importance

1: **procedure** OOBN VARIMPORTANCE:(data)
2:     **for** $b = 1, 2, \ldots, B$ **do**
3:         Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$
4:         Grow a random forest tree $T_b$
5:         Calculate accuracy on OOB samples
6:         **for** $j = 1, 2, \ldots, p$ **do**
7:             permute the values for the $j$th variable randomly in the OOB samples
8:             Calculate the decrease in the accuracy
9:         **end for**
10:     **end for**
11:     Output average accuracy gain for each $j = 1, 2, \ldots, p$.
12: **end procedure**

Alternative Variable Importance

with quite different results
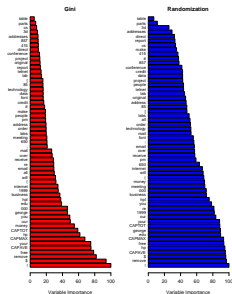


- The randomization voids the effect of a variable.

# Proximity plot

## Proximity plot

1: **procedure** PROXIMITY PLOT( $X, y$ training data )
2:     **for** $b = 1, 2, \ldots, B$ **do**
3:         Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$
4:         Grow a random forest tree $T_b$
5:         **Calculate prediction accuracy on OOB samples**
6:         **for** any pair of OOB samples sharing the same leaf **do**
7:             **increase the proximity by one.**
8:         **end for**
9:     **end for**
10: **end procedure**

- Distinct samples usually come from the pure regions
- Samples in the 'star center' are close to the decision boundary.
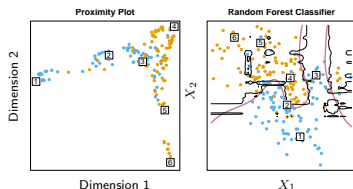
# Overfitting

- Though the random forest cannot overfit the limit distribution

$$\hat{f}_{rf}(x) = \mathbb{E}_\Theta \, T(x; \Theta) = \lim_{B \to \infty} \hat{f}_{rf}^B(x)$$

- the limit distribution (the average of fully grown trees) may overfit the data.
- Small number of relevant variables with many irrelevant hurts the random forest approach.
- With higher number of relevant variables RF is quite robust.

- 6 relevant and 100 noisy variables, $m = \sqrt{6 + 100} \sim 10$

- probability of a relevant variable being selected at any split is 0.46.
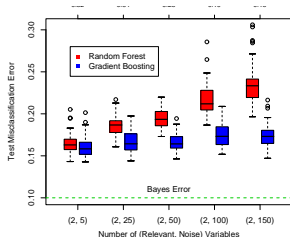
- Seldom the pruning improves the random forest result
- usually, fully grown trees are used.
  - Two additive vars, 10 noisy,
  - plus additive Gaussian noise.
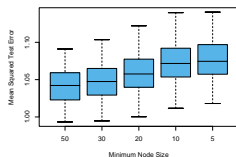




FIGURE 15.8. *The effect of tree size on the error.*

# Random Forest Experiments

Spam example misclassification error

- bagging 5.4%
- random forest 4.88%
- gradient boosting 4.5%.

Nested spheres in $\mathbb{R}^{10}$, 2500 trees, the number selected by 10–fold crossvalidation
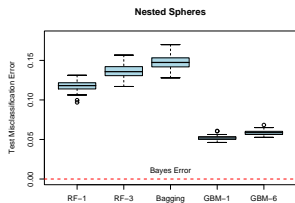


California housing data

- Random forests stabilize at about 200 trees, while at 1000 trees boosting continues to improve.
    - Boosting is slowed down by the shrinkage
    - the trees are much smaller (decision stumps, interaction depth=1 or 2).
- Boosting outperforms random forests here.
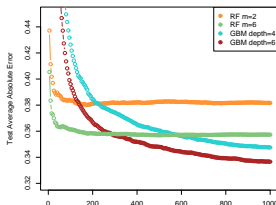
# Boosting

! Use a week classifier as a decision stump (a decision tree with the depth= 1).

## AdaBoost.M1

1: **procedure** ADABOOST CLASSIFIER( $X, G$ )
2:      Initialize the observation weights $w_i \leftarrow \frac{1}{N}$.
3:      **for** $m = 1, 2, \ldots, M$ **do**
4:          Fit a classifier $G_m(x)$ to the training data using weights $w_i$
5:          compute $err_m \leftarrow \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$
6:          compute $\alpha_m \leftarrow \log \frac{(1-err_m)}{err_m}$
7:          Set $w_i \leftarrow w_i \cdot e^{I(y_i \neq G_m(x_i)) \cdot \alpha_m}$
8:          (normalize weights)
9:      **end for**
10:      Output $G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$.
11: **end procedure**

- Two class problem with encoding $Y \in \{-1, 1\}$

- $\overline{err} = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq G(x_i))$.

FINAL CLASSIFIER

$G(x) = \text{sign} \left[ \sum_{m=1}^{M} \alpha_m G_n \right]$

Weighted Sample ···· $G_M(x)$

⋮

Weighted Sample ···· $G_3(x)$

Weighted Sample ···· $G_2(x)$

Training Sample ···· $G_1(x)$
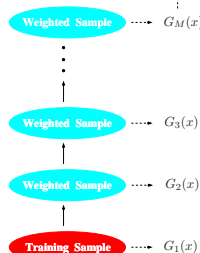
# Nested Spheres Example

- The features $X_1, \ldots, X_{10}$ are standard independent Gaussian
- deterministic target
  - $Y = 1$ iff $\sum_{j=1}^{10} X_j^2 > \chi_{10}^2(0.5) = 9.34$,
  - $Y = -1$ otherwise.
- 2000 training cases
- 10000 test observations.
- Decision stumps.
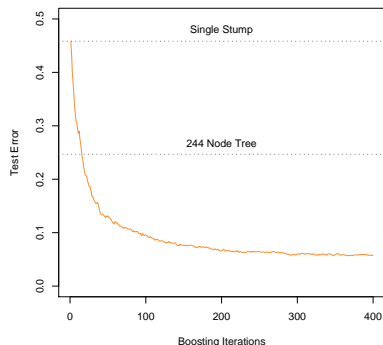
# Additive Model

- We encode the binary goal by $Y \in \{-1, +1\}$.
- Boosting fits an additive model:

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$$

- where $\beta_m$ for $m = 1, \dots, M$ are the expansion coefficients
- $b(x; \gamma) \in \mathbb{R}$ are usually simple functions of the multivariate argument $x$
  - characterized by a set of parameters $\gamma$.
- For trees, $\gamma$ parametrizes the split variables and split points at the internal nodes, and the predictions at the terminal nodes.
- Forward stagewise Additive Modeling sequentially adds one new basis function without adjusting the parameters and coefficients of the previously fitted.
- For squared–error loss

$$L(y, f(x)) = (y - f(x))^2,$$

we have

$$
\begin{aligned}
L(y_i, f_{m-1}(x) + \beta_m b(x_i; \gamma_m)) &= (y_i - f_{m-1}(x) - \beta_m b(x_i; \gamma_m))^2 \\
&= (r_{im} - \beta_m b(x_i; \gamma_m))^2
\end{aligned}
$$

# Exponential Loss and AdaBoost

- Let us use the $Y \in \{-1, 1\}$ encoding and the **exponential loss**

$$L(y, f(x)) = e^{-yf(x)}.$$

- We have to solve

$$
\begin{aligned}
(\beta_m, G_m) &= \arg\min_{\beta, G} \sum_{i=1}^{N} e^{[-y_i(f_{m-1}(x_i) + \beta G(x_i))]} \\
&= \arg\min_{\beta, G} \sum_{i=1}^{N} e^{[-y_i(f_{m-1}(x_i))]} e^{[-y_i \beta G(x_i)]} \\
&= \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} e^{[-y_i \beta G(x_i)]}
\end{aligned}
$$

- where $w_i^{(m)} = e^{[-y_i f_{m-1}(x_i)]}$ does not depend on $\beta$ nor $G(x)$.
- this weight depends on $f_{m-1}(x_i)$ and change with each iteration $m$.

# Exponential Loss and AdaBoost

- From

$$
\begin{aligned}
(\beta_m, G_m) &= \underset{\beta, G}{\arg\min} \sum_{i=1}^{N} w_i^{(m)} e^{[-y_i \beta G(x_i)]} \\
&= \underset{\beta, G}{\arg\min} \left[ e^{\beta} \cdot \sum_{y_i \neq G(x_i)} w_i^{(m)} + e^{-\beta} \cdot \sum_{y_i = G(x_i)} w_i^{(m)} \right] \\
&= \underset{\beta, G}{\arg\min} \left[ (e^{\beta} - e^{-\beta}) \cdot \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i)) + e^{-\beta} \cdot \sum_{i=1}^{N} w_i^{(m)} \right]
\end{aligned}
$$

- For any $\beta > 0$ the solution for $G_m(x; \gamma)$ is

$$
G_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i; \gamma)),
$$

- Recall the error definition:

$$
err_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i^{(m)}}
$$

# Adaboost Update

$$\arg\min_{\beta,G}\left[(e^{\beta}-e^{-\beta})\cdot\sum_{i=1}^{N}w_i^{(m)}I(y_i\neq G(x_i))+e^{-\beta}\cdot\sum_{i=1}^{N}w_i^{(m)}\right]$$

- The minimum w.r.t. $\beta_m$ is:

$$\beta_m=\frac{1}{2}log\frac{1-err_m}{err_m}$$

- The approximation is updated

$$f_m(x)=f_{m-1}(x)+\beta_m G_m(x)$$

- which causes the weights for the next iteration to be:

$$w_i^{m+1}=w_i^m\cdot e^{-\beta_m y_i G_m(x_i)}.$$

- using the fact $-y_i G_m(x_i)=2\cdot I(y_i\neq G_m(x_i))-1$ we get

$$w_i^{m+1}=w_i^m\cdot e^{\alpha I(y_i\neq G_m(x_i))}\cdot e^{-\beta_m}.$$

# Why exponential loss?

- The population minimizer is

$$f^*(x) = \arg\min_{f(x)} \mathbb{E}_{Y|x}(e^{-Yf(x)}) = \frac{1}{2} log \frac{P(Y=1|x)}{P(Y=-1|x)}.$$

- therefore

$$P(Y=1|x) = \frac{1}{1 + e^{-2f^*(x)}}.$$

- The same function $f^*(x)$ minimizes also deviance (cross–entropy, binomial negative log–likelihood)
  - interpreting $f^*$ as the logit transform. Let:

$$p(x) = P(Y=1|x) = \frac{e^{f^*(x)}}{e^{-f^*(x)} + e^{f^*(x)}} = \frac{1}{1 + e^{-2f^*(x)}}.$$

  - and define $Y^| = (Y+1)/2 \in \{0,1\}$. Log–likelihood is

$$\ell(Y, p(x)) = Y^| \log p(x) + (1 - Y^|) \log(1 - p(x))$$

  - or equivalently the deviance:

$$-\ell(Y, f(x)) = log\left(1 + e^{-2Yf(x)}\right).$$

- Exponential loss decreases long after misclassification loss is stable at zero.
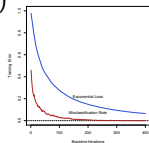
# Forward Stagewise Additive Modeling

- A general iterative fitting approach.
- In each step, we select the best function from the dictionary $b(x_i; \gamma)$, fit its parameters $\gamma$ and the weight of this basis function $\beta_m$.
- Stagewise approximation is often faster then iterative fitting of the full model.

### Forward Stagewise Additive Modeling

1: **procedure** FORWARD STAGEWISE ADDITIVE MODELING( $L, X, Y, b$)
2:     Initialize $f_0 \leftarrow 0$.
3:     **for** $m = 1, 2, \ldots, M$ **do**
4:         Compute $(\beta_m, \gamma_m) \leftarrow \arg\min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$.
5:         Set $f_m(x) \leftarrow f_{m-1}(x) + \beta_m b(x_i; \gamma_m)$
6:     **end for**
7: **end procedure**

- For example, our basis functions are decision trees, $\gamma$ represents the splits and fitted values $T(*; \gamma)$).
- For square error loss, any new tree $T(*; \gamma)$ is the best tree fitting residuals $r_i = y_i - f_{m-1}(x_i)$.

# Gradient Tree Boosting Algorithm

## Gradient Tree Boosting Algorithm

1: **procedure** GRADIENT TREE BOOSTING ALGORITHM( $X, Y, L$ )

2:      Initialize $f_0(x) \leftarrow \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.

3:      **for** $m = 1, 2, \ldots, M$ **do**

4:           **for** $i = 1, 2, \ldots, N$ **do**

5:                compute $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)} =^{[*]} y_i - f_{m-1}(x_i)$

6:           **end for**

7:           Fit reg. tree to the target $r_{im}$ giving regions $\{R_{jm}\}_{j=1,\ldots,J_m}$.

8:           **for** $j = 1, 2, \ldots, J_m$ **do**

9:                Compute $\gamma_{jm} \leftarrow \arg\min_\gamma \sum_{i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$.

10:          **end for**

11:          Set $f_m(x) \leftarrow f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.

12:     **end for**

13:     Output $\hat{f}(x) = f_M(x)$.

14: **end procedure**

[*] for square error loss.

# Stacking

- Over a set of models (possibly different types) learn a simple model (like a linear regression)
- Assume predictions $\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_M(x)$ under square error loss
- Predictors trained without $i$th example are denoted
  - $\hat{f}_1^{-i}(x), \hat{f}_2^{-i}(x), \ldots, \hat{f}_M^{-i}(x)$
- we can seek weights $w = (w_1, \ldots, w_m)$ such that

$$\hat{w}^{st} = \arg\min_w \sum_{i=1}^{N} \left[ y_i - \sum_{m=1}^{M} w_m \hat{f}_m^{-i}(x) \right]^2.$$

- The final prediction is

$$\hat{f}^{st}(x) = \sum_{m=1}^{M} w_m^{st} \hat{f}_m(x).$$

  - Using cross–validated predictions $\hat{f}_m^{-i}(x)$ stacking avoids giving unfairly high weight to models with higher complexity
  - Better results can be obtained by restricting the weights to be nonnegative and to sum to 1.

# Decision Rules from Decision Trees

- We can represent a tree as a set of rules
  - one rule for each leaf.
- These rules may be improved by testing each attribute in each rule
  - Has the rule without this test a better precision than with the test?
  - Use validation data
  - May be time consuming.
- These rules are sorted by (decreasing) precision.

# Patient Rule Induction Method PRIM = Bump Hunting

- Rule induction method
- We iteratively search regions with the high $Y$ values
  - for each region, a rule is created.
- CART runs of data after approximately $\log_2(N) - 1$ cuts.
- PRIM can affort $-\frac{\log(N)}{\log(1-\alpha)}$. For $N = 128$ data samples and $\alpha = 0.1$ it is 6 and 46 respectively 29, since the number of observations must be a whole number.
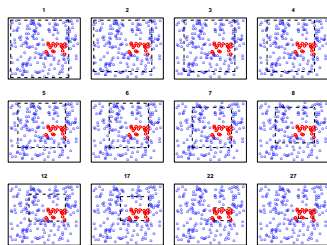


**FIGURE 9.7.** *Illustration of PRIM algorithm. There are two classes, indicated by the blue (class 0) and red (class 1) points. The procedure starts with a rectangle (broken black lines) surrounding all of the data, and then peels away points along one edge by a prespecified amount in order to maximize the mean of the points remaining in the box. Starting at the top left panel, the sequence of peelings is shown, until a pure red region is isolated in the bottom right panel. The iteration number is indicated at the top of each panel.*

# PRIM Patient Rule induction Algorithm

## PRIM

- Consider the whole space and all data. Set $\alpha = 0.05$ or $0.10$.
- Find $X_j$ and its upper or lower boundary such that the cut of $\alpha \cdot 100\%$ observations leads to the maximal mean of the remaining data.
- Repeat until less then 10 observations left.
- Enlarge the region in any direction that increases the mean value.
- Select the number of regions by the crossvalidation. All regions generated 1-4 are considered.
- Denote the best region $B_1$.
- Create a rule that describes $B_1$.
- Remove all data in $B_1$ from the dataset.
- Repeat 2-5, create $B_2$ continue until final condition met.
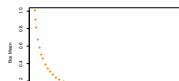
# CART Weaknesses

- the high variance
  - the tree may be very different for very similar datasets
  - ensemble learning addresses this issue
- the cuts are perpendicular to the axis
- the result is not smooth but stepwise.
  - MARS (Multivariate Adaptive Regression Splines) addresses this issue.
- it is difficult to capture an additive structure

$$Y = c_1 I(X_1 < t_1) + c_2 I(X_2 < t_2) + \ldots + c_k I(X_k < t_k) + \epsilon$$

- MARS (Multivariate Adaptive Regression Splines) addresses this issue.
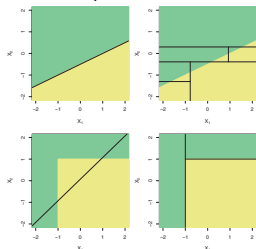


FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a de-*

# MARS Multivariate Adaptive Regression Splines

- generalization of linear regression and decision trees CART
- for each feature and each data point we create a **reflected pair** of basis functions
- $(x - t)_+$ and $(t - x)_+$ where $+$ denotes non–negative part, minimum is zero.
- we have the set of functions

$$\mathcal{C} = \{(X_j - t)_+, (t - X_j)_+\}_{t \in \{x_{1,j}, x_{2,j}, \ldots, x_{N,j}\}, j=1,2,\ldots,p}$$

- that is $2Np$ functions for non–duplicated data points.

# MARS – continuation

- our model is in the form

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X)$$

where $h_m(X)$ is a function from $\mathcal{C}$ or a product of any amount of functions from $\mathcal{C}$

- for a fixed set of $h_m$'s we calculate coefficients $\beta_m$ by usual linear regression (minimizing RSS)

- the set of functions $h_m$ is selected iteratively.

- We start with $h_0 = 1$, we put this function into the model $\mathcal{M} = \{h_0\}$.

- We consider the product of any member $h_\ell \in \mathcal{M}$ with any pair from $\mathcal{C}$,

$$\hat{\beta}_{M+1} h_\ell(X) \cdot (X_j - t)_+ + \hat{\beta}_{M+2} h_\ell(X) \cdot (t - X_j)_+$$

we select the one minimizing training error RSS (for any product candidate, we estimate $\hat{\beta}$).

- Repeat until predefined number of functions in $\mathcal{M}$

# MARS – model pruning

- The model is usually overfitted. We select (remove) iteratively the one minimizing the increase of training RSS. We have a sequence of models $\hat{f}_\lambda$ for different numbers of parameters $\lambda$.
- (we want to speed–up cross-validation for computational reasons)
- we select $\lambda$ (and the model) minimizing **generalized cross-validation**

$$GCV(\lambda) = \frac{\sum_{i=1}^{N}(y_i - \hat{f}_\lambda(x_i))^2}{(1 - M(\lambda)/N)^2}.$$

- where $M(\lambda)$ is the number of effective parameters, the number of function $h_m$ (denoted $r$) plus the number of knots $K$, the authors suggest to multiply $K$ by 3: $M(\lambda) = r + 3K$.

# MARS is a generalization of CART

- We select piecewise constant functions $I(x - t > 0)$ and $I(x - t \leq 0)$
- If $h_m$ uses multiplication we remove this function from the candidate list. It cannot be used any more.
  - This guarantees binary split.
- Its CART.
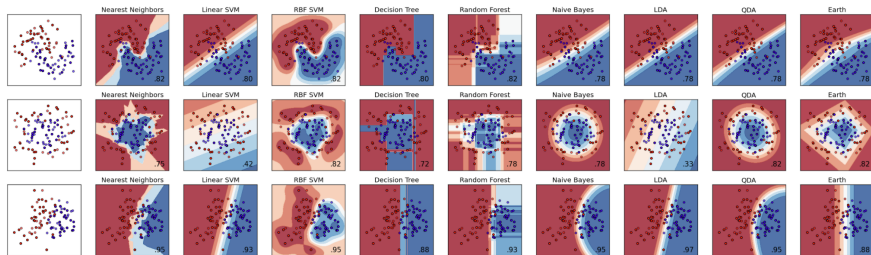


`https://contrib.scikit-learn.org/py-earth/auto_examples/plot_classifier_comp.html`
`https://contrib.scikit-learn.org/py-earth/auto_examples/index.html`

# Bayesian Learning

Complicated derivation of known things.

- **Maximum a posteriori probability hypothesis** (MAP)
  (nejpravděpodobnější hypotéza)
- **Maximum likelihood hypothesis** (ML) (maximálně věrohodná hypotéza)
- **Bayesian optimal prediction** (Bayes Rate)
- Bayesian methods, bayesian smoothing
- **EM algorithm**
- **Naive Bayes model (classifier)**.

- Our favorite candy comes in two flavors: cherry and lime, both in the same wrapper.
- They are in a bag in one of following rations of cherry candies and prior probability of bags:

| hypothesis (bag type) | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
|---|---|---|---|---|---|
| cherry | 100% | 75% | 50% | 25% | 0% |
| prior probability $h_i$ | 10% | 20% | 40% | 20% | 10% |

- The first candy is cherry.

  MAP Which of $h_i$ is the most probable given first candy is cherry?

  Bayes estimate What is the probability next candy from the same bag is cherry?

# Maximum Aposteriory Probability Hypothesis (MAP)

- We assume large bags of candies, the result of one missing candy in the bag is negligable.
- Recall Bayes formula:

$$P(h_i|B = c) = \frac{P(B = c|h_i) \cdot P(h_i)}{\sum_{j=1,\ldots,5} P(B = c|h_j) \cdot P(h_j)} = \frac{P(B = c|h_i) \cdot P(h_i)}{P(B = c)}$$

- We look for the MAP hypothesis **maximálně aposteriorně pravděpodobná**

$$argmax_i P(h_i|B = c) = argmax_i P(B = c|h_i) \cdot P(h_i).$$

- Aposteriory probabilities of hypotheses are in the following table.

| index | prior | cherry ratio | cherry AND $h_i$ | aposteriory prob. $h_i$ |
|:-----:|:-----:|:------------:|:----------------:|:-----------------------:|
| $i$ | $P(h_i)$ | $P(B = c\|h_i)$ | $P(B = c\|h_i) \cdot P(h_i)$ | $P(h_i\|B = c)$ |
| 1 | 0.1 | 1 | 0.1 | 0.2 |
| 2 | 0.2 | 0.75 | 0.15 | 0.3 |
| 3 | 0.4 | 0.5 | 0.2 | **0.4** |
| 4 | 0.2 | 0.25 | 0.05 | 0.1 |
| 5 | 0.1 | 0 | 0 | 0 |

- Which hypothesis is most probable?

$$h_{MAP} = argmax_i P(data|h_i) \cdot P(h_i)$$

- What is the prediction of a new candy according the most probable hypothesis $h_{MAP}$?

# Bayesian Learning, Bayesian Optimal Prediction

- **Bayesian optimal prediction** is weigthed average of predictions of all hypotheses:

$$P(N = c|data) = \sum_{j=1,\ldots,5} P(N = c|h_j, data) \cdot P(h_j|data)$$

$$= \sum_{j=1,\ldots,5} P(N = c|h_j) \cdot P(h_j|data)$$

- If our model is correct, no prediction has smaller expected error then Bayesian optimal prediction.
- We always assume i.i.d. data, independently identically distributed.
- We assume the hypothesis fully describes the data behavior. Observations are mutually conditionally independent given the hypothesis. This allows the last equation above.

# Candy Example: Bayesian Optimal Prediction

| $i$ | $P(h_i|B=c)$ | $P(N=c|h_i)$ | $P(N=c|h_i) \cdot P(h_i|B=c)$ |
|-----|--------------|--------------|-------------------------------|
| 1 | 0.2 | 1 | 0.2 |
| 2 | 0.3 | 0.75 | 0.225 |
| 3 | 0.4 | 0.5 | 0.2 |
| 4 | 0.1 | 0.25 | 0.02 |
| 5 | 0 | 0 | 0 |
| $\sum$ | 1 | | 0.645 |

# Maximum Likelihood Estimate (ML)

- Usually, we do not know prior probabilities of hypotheses.
- Setting all prior probabilities equal leads to **Maximum Likelihood Estimate, maximálně věrohodný odhad**

$$h_{ML} = argmax_i P(data|h_i)$$

- Probability of data given hypothesis = likelihood of hypothesis given data.
- Find the ML estimate:

| index | prior | cherry ratio | cherry AND $h_i$ | Aposteriory prob. $h_i$ |
|-------|-------|--------------|------------------|--------------------------|
| $i$ | $P(h_i)$ | $P(B = c|h_i)$ | $P(B = c|h_i) \cdot P(h_i)$ | $P(h_i|B = c)$ |
| 1 | 0.1 | 1 | 0.1 | 0.2 |
| 2 | 0.2 | 0.75 | 0.15 | 0.3 |
| 3 | 0.4 | 0.5 | 0.2 | 0.4 |
| 4 | 0.2 | 0.25 | 0.05 | 0.1 |
| 5 | 0.1 | 0 | 0 | 0 |

- In this example, do you prefer ML estimate or MAP estimate?
- (Only few data, over-fitting, penalization is useful. AIC, BIC)

# Maximum Likelihood: Continuous Parameter $\theta$

- New producer on the market. We do not know the ratios of candies, any $h_\theta$, kde $\theta \in \langle 0; 1 \rangle$ is possible, any prior probabilities $h_\theta$ are possible.
- We look for maximum likelihood estimate.
- For a given hypothesis $h_\theta$, the probability of a cherry candy is $\theta$, of a lime candy $1 - \theta$.
- Probability of a sequence of $c$ cherry and $l$ lime candies is:

$$P(data|h_\theta) = \theta^c \cdot (1 - \theta)^l.$$

# ML Estimate of Parameter $\theta$

- Probability of a sequence of $c$ cherry and $l$ lime candies is:

$$P(data|h_\theta) = \theta^c \cdot (1 - \theta)^l$$

- Usual trick is to take logarithm:

$$\ell(h_\theta; data) = c \cdot \log_2 \theta + l \cdot \log_2(1 - \theta)$$

- To find the maximum of $\ell$ (log likelihood of the hypothesis) with respect to $\theta$ we set the derivative equal to 0:

$$\frac{\partial \ell(h_\theta; data)}{\partial \theta} = \frac{c}{\theta} - \frac{l}{1 - \theta}$$
$$\frac{c}{\theta} = \frac{l}{1 - \theta}$$
$$\theta = \frac{c}{c + l}.$$

# ML Estimate of Multiple Parameters

- Producer introduced two colors of wrappers - red $r$ and green $g$.
- Both flavors are wrapped in both wrappers, but with different probability of the red/green wrapper.
- We need three parameters to model this situation:

| $P(B = c)$ | $P(W = r \mid B = c)$ | $P(W = r \mid B = l)$ |
|:---:|:---:|:---:|
| $\theta_0$ | $\theta_1$ | $\theta_2$ |

- Following table denotes observed frequences:

| wrapper\ flavor | cherry | lime |
|:---:|:---:|:---:|
| red | $r_c$ | $r_l$ |
| green | $g_c$ | $g_l$ |

# ML Estimate of Multiple Parameters

Parameters are:

| $P(B = c)$ | $P(W = r \mid B = c)$ | $P(W = r \mid B = l)$ |
|:---:|:---:|:---:|
| $\theta_0$ | $\theta_1$ | $\theta_2$ |

Probability of data given the hypothesis $h_{\theta_0, \theta_1, \theta_2}$ is:

$$
\begin{aligned}
P(data \mid h_{\theta_0, \theta_1, \theta_2}) &= \theta_1^{r_c} \cdot (1 - \theta_1)^{g_c} \cdot \theta_0^{r_c + g_c} \cdot \theta_2^{r_l} \cdot (1 - \theta_2)^{g_l} \cdot (1 - \theta_0)^{r_l + g_l} \\
\ell(h_{\theta_0, \theta_1, \theta_2}; data) &= r_c \log_2 \theta_1 + g_c \log_2 (1 - \theta_1) + (r_c + g_c) \log_2 \theta_0 \\
&\quad + r_l \log_2 \theta_2 + g_l \log_2 (1 - \theta_2) + (r_l + g_l) \log_2 (1 - \theta_0)
\end{aligned}
$$

We look for maximum:

$$
\begin{aligned}
\frac{\partial \ell(h_{\theta_0, \theta_1, \theta_2}; data)}{\partial \theta_0} &= \frac{r_c + g_c}{\theta_0} - \frac{r_l + g_l}{1 - \theta_0} \\
\theta_0 &= \frac{(r_c + g_c)}{r_c + g_c + r_l + g_l} \\
\frac{\partial \ell(h_{\theta_0, \theta_1, \theta_2}; data)}{\partial \theta_2} &= \frac{r_l}{\theta_2} - \frac{g_l}{1 - \theta_2} \\
\theta_2 &= \frac{r_l}{r_l + g_l}.
\end{aligned}
$$

- Maximum Likelihood estimate is the ratio of frequences.

# ML Estimate of Gaussian Distribution Parameters

- Assume $x$ to have Gaussian distribution with unknown parameters $\mu$ a $\sigma$.
- Our hypotheses are $h_{\mu,\sigma} = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$.
- We have observed $x_1, \ldots, x_n$.
- Log likelihood is:

$$
\begin{aligned}
LL &= \sum_{j=1}^{N} log \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \\
&= N \cdot \left( log \frac{1}{\sqrt{2\pi}\sigma} \right) - \sum_{j=1}^{N} \frac{(x_j - \mu)^2}{2\sigma^2}
\end{aligned}
$$

- Find the maximum.

# Linear Gaussian Distribution

- Assume random variable (feature) $X$.
- Assume goal variable $Y$ with linear gaussian distribution where $\mu = b \cdot x + b_0$ and fixed variance $\sigma^2$ $p(Y|X=x) = N(b \cdot x + b_0; \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y-((b \cdot x + b_0))^2}{2\sigma^2}}$.
- Find maximum likelihood estimate of $b, b_0$ given a set of observations $data = \{\langle x_1, y_1 \rangle, \ldots, \langle x_N, y_N \rangle\}$.
- (Look for maximum of the logarithm of it; change the max to min with the opostite sign. Do you know this formula?)

$$argmax_{b,b_0}(log_e(\Pi_{i=1}^{N}(e^{-(y_i-(b \cdot x_i + b_0))^2}))) = argmin_{b,b_0}(?)$$

# Bayesian Methods

- We specify a sampling model $P(\mathbf{Z}|\theta)$
- and a prior distribution for parameters $P(\theta)$
- then we compute

$$P(\theta|\mathbf{Z}) = \frac{P(\mathbf{Z}|\theta) \cdot P(\theta)}{\int P(\mathbf{Z}|\theta) \cdot P(\theta)d\theta},$$

  - we may draw samples
  - or summarize by the mean or mode.
  - it provides the **Bayesian optimal predictive distribution**:

$$P(z^{new}|\mathbf{Z}) = \int P(z^{new}|\theta) \cdot P(\theta|\mathbf{Z})d\theta.$$

## Example

Tossing a biased coin

- $P(Z = head|\theta) = \theta$
- $p(\theta) =$ uniform
- $P(\theta|\mathbf{Z})$ follows the Beta distribution.

# Discrete Model Parameter Learning

- For binary features, Beta function is used, $(a - 1)$ is the number of positive examples, $(b - 1)$ the number of negative examples.

$$beta[a, b](\theta) = \alpha\theta^{a-1}(1 - \theta)^{b-1}$$

- Beta Function:
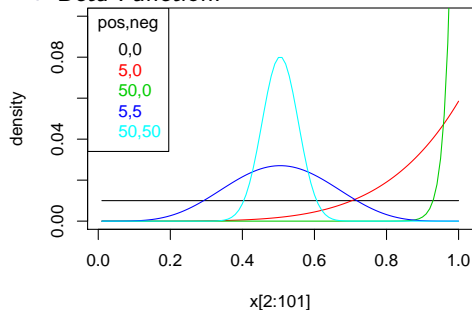


- For categorical features, Dirichlet priors and multinomial distribution is used. (Dirichlet-multinomial distribution).
- For Gaussian, $\mu$ has Gaussian prior, $\frac{1}{\sigma}$ has gamma prior (to stay in exponential family).

# MAP and Penalized Methods

- MAP hypothesis maximizes:

$$h_{MAP} = argmax_i P(data|h_i) \cdot P(h_i)$$

- therefore minimizes:

$$
\begin{aligned}
h_{MAP} &= argmax_h P(data|h)P(h) \\
&= argmin_h[-log_2 P(data|h) - log_2 P(h)] \\
&= argmin_h[-loglik + \text{complexity penalty}] \\
&= argmin_h[RSS + \text{complexity penalty}] \text{ Gaussian models} \\
&= argmax_h[loglik - \text{complexity penalty}] \text{ Categorical models}
\end{aligned}
$$

# Bayesian smoothing example

- Training data $\mathbf{Z} = \{z_i, \ldots, z_N\}$, $z_i = (x_i, y_i)$, $i = 1, \ldots, N$.

- We look for a cubic spline with three knots in quartiles of the $X$ values. It corresponds to B-spline basis $h_j(x)$, $j = 1, \ldots, 7$.



- We estimate the conditional mean $\mathbb{E}(Y|X = x)$: $\mu(x) = \sum_{j=1}^{7} \beta_j h_j(x)$

- Let $\mathbf{H}$ be the $N \times 7$ matrix $h_j(x_i)$.

- RSS $\beta$ estimate is $\hat{\beta} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$.
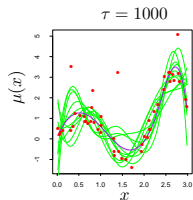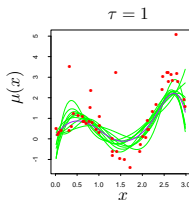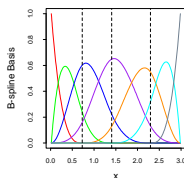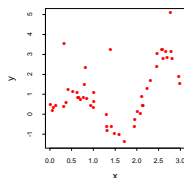


We assume to know $\sigma^2$, fixed $x_i$, we specifying prior on $\beta \sim N(0, \tau\Sigma)$.

$$\mathbb{E}(\beta|\mathbf{Z}) = (\mathbf{H}^T\mathbf{H} + \frac{\sigma^2}{\tau}\Sigma^{-1})^{-1}\mathbf{H}^T\mathbf{y}$$

$$\mathbb{E}(\mu(x)|\mathbf{Z}) = h(x)^T(\mathbf{H}^T\mathbf{H} + \frac{\sigma^2}{\tau}\Sigma^{-1})^{-1}\mathbf{H}^T\mathbf{y}.$$

# Naive Bayes Model, Bayes Classifier

- Maximum Likelihood estimate is the ratio of frequences.
  - We may use smoothed estimate adding $\alpha$ samples to each possibility to avoid zero probabilities.
- ML estimite of a gaussian distribution parameters are the mean and the variance (or covariance matrix for multivariate distribution).
- **Naive Bayes Model, Bayes Classifier** assumes independent features given the class variable.
  - Calculate prior probability of classes $P(c_i)$
  - For each feature $f$, calculate for each class the probability of this feature $P(f|c_i)$
  - For a new observation of features $f$ predict the most probable class $argmax_{c_i} P(f|c_i) \cdot P(c_i)$.
- **Bayes factor**
  - We can start with a comparison ratio of two classes $\frac{P(c_i)}{P(c_j)}$
  - after each observation $x_p$ multiply it by the **bayes factor** $\frac{P(x_p|c_i)}{P(x_p|c_j)}$
  - that is:
  $$\frac{P(c_i|x_1, \ldots, x_p)}{P(c_j|x_1, \ldots, x_p)} = \frac{P(c_i)}{P(c_j)} \cdot \frac{P(x_1|c_i)}{P(x_1|c_j)} \cdot \ldots \cdot \frac{P(x_p|c_i)}{P(x_p|c_j)}.$$
- Bayesian Networks learn more complex (in)dependencies between features.

# Expectation Maximization Algorithm (EM Algorithm)

- EM algorithm estimates the maximum likelihood model based on the data with missing values.
- used in HMM
- used in clustering (Gaussian mixture model estimation)
- but not restricted to this applications
- It is a general approach to fill missing values based on the maximum likely model.

## Example (EM Algorithm for Missing Data)

- Two bags of bonbons mixed together. Each bonbon has a *Wrapper* and flavor *Flavor* and may have *Holes*. Each bag had another ratio of *Wrapper* color and *Flavor*.
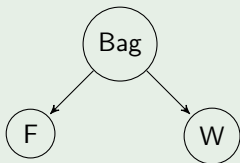


| Bag | F | W |
|-----|---|---|
| ?   | c | r |
| 1   | l | r |
| 1   | c | ? |
| 1   | c | g |
| ?   | l | ? |

- Initialize all parameters randomly close to uniform distribution, $\theta_* \approx 0.5$.

$\mathbb{E}$ step

| $w = \hat{P}(\mathbf{Z}^m \mid \theta, \mathbf{Z})$ | Bag | F | W |
|------|-----|---|---|
| $P_\theta(Bag = 1 \mid F = c, W = r)$ | 1 | c | r |
| $P_\theta(Bag = 2 \mid F = c, W = r)$ | 2 | c | r |
| 1 | 1 | l | r |
| $P_\theta(W = r \mid Bag = 1, F = c)$ | 1 | c | r |
| $P_\theta(W = g \mid Bag = 1, F = c)$ | 1 | c | g |
| 1 | 1 | c | g |
| $P_\theta(Bag = 1, W = r \mid F = l)$ | 1 | l | r |
| $P_\theta(Bag = 1, W = g \mid F = l)$ | 1 | l | g |
| $P_\theta(Bag = 0, W = r \mid F = l)$ | 2 | l | r |
| $P_\theta(Bag = 0, W = g \mid F = l)$ | 2 | l | g |

M step – update $\theta$s

$$\theta_{Bag=1} \leftarrow \frac{\sum_{Bag=1} w}{\sum w}$$

$$\theta_{F=c \mid Bag=1} \leftarrow \frac{\sum_{Bag=1, F=c} w}{\sum_{Bag=1} w}$$

$$\theta_{F=c \mid Bag=2} \leftarrow \frac{\sum_{Bag=2, F=c} w}{\sum_{Bag=2} w}$$

$$\theta_{W=r \mid Bag=1} \leftarrow \frac{\sum_{Bag=1, W=r} w}{\sum_{Bag=1} w}$$

$$\theta_{W=r \mid Bag=2} \leftarrow \frac{\sum_{Bag=2, W=r} w}{\sum_{Bag=2} w}$$

# EM as a Maximization-Maximization Procedure

- **Z** the observed data (the usual $X$ with missing values)
- $\ell(\theta; \mathbf{Z})$ the log–likelihood of the model $\theta$
- $\mathbf{Z}^m$ the latent or missing data
- $T = (\mathbf{Z}, \mathbf{Z}^m)$ the complete data with the log–likelihood $\ell_0(\theta; \mathbf{T})$.
- $\hat{P}(\mathbf{Z}^m), \hat{P}(\mathbf{Z}^m | \theta, \mathbf{Z})$ any distribution over the latent data $\mathbf{Z}^m$.



- Consider the function $F$

$$F(\theta', \hat{P}) = \mathbb{E}_{\hat{P}}[\ell_0(\theta'; \mathbf{T})] - \mathbb{E}_{\hat{P}}[log \hat{P}(\mathbf{Z}^m)]$$

- for $\hat{P} = \hat{P}(\mathbf{Z}^m | \theta', \mathbf{Z})$ is $F$ the log–likelihood of the observed data
  - $F(\theta', \hat{P}(\mathbf{Z}^m | \theta', \mathbf{Z})) = \mathbb{E}[\ell_0(\theta'; \mathbf{T}) | \theta', \mathbf{Z}] - \mathbb{E}[\ell_1(\theta'; \mathbf{Z}^m | \mathbf{Z}) | \theta', \mathbf{Z}]$
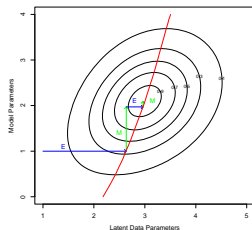
# The EM Algorithm in General

$$
\begin{aligned}
P(\mathbf{Z}^m|\mathbf{Z}, \theta') &= \frac{P(\mathbf{Z}^m, \mathbf{Z}|\theta')}{P(\mathbf{Z}|\theta')}, \\
P(\mathbf{Z}|\theta') &= \frac{P(\mathbf{Z}^m, \mathbf{Z}|\theta')}{P(\mathbf{Z}^m|\mathbf{Z}, \theta')},
\end{aligned}
$$

- In the log–likelihoods

$$
\ell(\theta'; \mathbf{Z}) = \ell_0(\theta'; \mathbf{T}) - \ell_1(\theta'; \mathbf{Z}^m|\mathbf{Z})
$$

- where $\ell_1$ is based on the conditional density $P(\mathbf{Z}^m|\mathbf{Z})$.
- Taking the expectation w.r.t. $\mathbf{T}|\mathbf{Z}$ governed by parameter $\theta$ gives

$$
\begin{aligned}
\ell(\theta'; \mathbf{Z}) &= \mathbb{E}[\ell_0(\theta'; \mathbf{T})|\theta, \mathbf{Z}] - \mathbb{E}[\ell_1(\theta'; \mathbf{Z}^m|\mathbf{Z})|\theta, \mathbf{Z}] \\
&\equiv Q(\theta', \theta) - R(\theta', \theta)
\end{aligned}
$$

- $R()$ is the expectation of a density with respect the same density
  - it is maximized when $\theta' = \theta$.
- Therefore:

$$
\begin{aligned}
\ell(\theta'; \mathbf{Z}) - \ell(\theta; \mathbf{Z}) &= [Q(\theta', \theta) - Q(\theta, \theta)] - [R(\theta', \theta) - R(\theta, \theta)] \\
&\geq 0.
\end{aligned}
$$

## The EM Algorithm

1: **procedure** THE EM ALGORITHM:( **Z** observed data, the model($\theta$) )
2:     $\hat{\theta}^{(0)} \leftarrow$ an initial guess (usually close to the uniform distribution)
3:     **repeat**
4:         *Expectation step:* at the $j$th step, compute

$$Q(\theta', \hat{\theta}^{(j)}) = \mathbb{E}(\ell_0(\theta'; \mathbf{T})|Z, \hat{\theta}^{(j)})$$

5:                         as a function of the dummy argument $\theta'$.
6:         *Maximization step:* determine the new estimate $\hat{\theta}^{(j+1)}$
7:                         as the maximizer of $Q(\theta', \hat{\theta}^{(j)})$ over $\theta'$.
8:     **until** convergence
9:     return $\hat{\theta}$
10: **end procedure**

- Full maximization is not necessary.
- We need to find a value $\hat{\theta}^{(j+1)}$ so that $Q(\hat{\theta}^{(j+1)}, \hat{\theta}^{(j)}) > Q(\hat{\theta}^{(j)}, \hat{\theta}^{(j)})$.
- Such prodecures are called **generalized EM algorithms (GEM)**.

# BN example of EM algorithm (Russel, Norvig) - can be omitted

- Two bags of bonbons mixed together. Each bonbon has a *Wrapper* and flavor *Flavor* and may have *Holes*. Each bag had another ratio of *Wrapper* color, *Flavor* and *Holes*.

We can model the situation by a naive bayes model, *Bag* as the class variable.

Example We have tested 1000 bonbones and observed:

| | W=red | | W=green | |
|---|---|---|---|---|
| | H=1 | H=0 | H=1 | H=0 |
| F=cherry | 273 | 93 | 104 | 90 |
| F=lime | 79 | 100 | 94 | 167 |
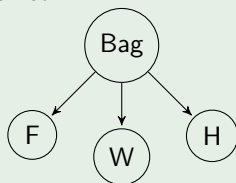


We choose the initial parameters

$$\theta^{(0)} = 0.6, \ \theta_{F1}^{(0)} = \theta_{W1}^{(0)} = \theta_{H1}^{(0)} = 0.6, \ \theta_{F2}^{(0)} = \theta_{W2}^{(0)} = \theta_{H2}^{(0)} = 0.4$$

# EM example - can be omitted

- Expectation of $\theta$ is the ratio of the expected counts

$$\theta^{(1)} = \frac{1}{N} \sum_{j=1}^{N} \frac{P(flavor_j|Bag=1)P(wrapper_j|Bag=1)P(holes_j|Bag=1)P(Bag=1)}{\sum_{i=1}^{2} P(flavor_j|Bag=i)P(wrapper_j|Bag=i)P(holes_j|Bag=i)P(Bag=i)}$$

(normalization constant **depends** on parameter values).

For the type *red*, *cherry*, *holes* we get:

$$\frac{\theta_{F1}^{(0)}\theta_{W1}^{(0)}\theta_{H1}^{(0)}\theta^{(0)}}{\theta_{F1}^{(0)}\theta_{W1}^{(0)}\theta_{H1}^{(0)}\theta^{(0)} + \theta_{F2}^{(0)}\theta_{W2}^{(0)}\theta_{H2}^{(0)}\theta^{(0)}} \approx 0.835055$$

we have 273 bonbons of this type, therefore we add $\frac{273}{N} \cdot 0.835055$.
Similarly for all seven other types and we get

$$\theta^{(1)} = 0.6124$$

# EM example continued - can be omitted

- The estimate of $\theta_{F1}$ for fully observed data is $\frac{\#(Bag=1, Flavor=cherry)}{\#(Flavor=cherry)}$
- We have to use expected counts $Bag = 1 \& F = cherry$ and $Bag = 1$,

$$\theta_{F1}^{(1)} = \frac{\sum_{j;Flavor_j=cherry} P(Bag = 1|Flavor_j = cherry, wrapper_j, holes_j)}{\sum_j P(Bag = 1|cherry_j, wrapper_j, holes_j)}$$

- Similarly we get:

$$\theta^{(1)} = 0.6124, \ \theta_{F1}^{(1)} = 0.6684, \theta_{W1}^{(1)} = 0.6483, \theta_{H1}^{(1)} = 0.6558,$$

$$\theta_{F2}^{(1)} = 0.3887, \theta_{W2}^{(1)} = 0.3817, \theta_{H2}^{(1)} = 0.3827.$$

# Hierarchical Mixture of Experts

- a hierarchical extension of naive Bayes (latent class model)

- a decision tree with 'soft splits'

- splits are probabilistic functions of a linear combination of inputs (not a single input as in CART)

- terminal nodes called 'experts'

- non–terminal nodes are called gating network

- may be extended to multilevel.
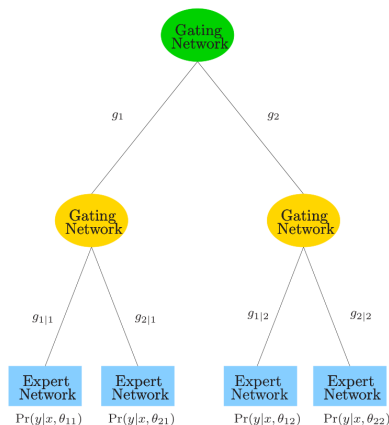
# Hierarchical Mixture of Experts



- data $(x_i, y_i)$, $i = 1, \ldots, N$, $y_i$ continuous or categorical, first $x_i \equiv 1$ for intercepts.

- $g_i(x, \gamma_j) = \dfrac{e^{\gamma_j^T x}}{\sum_{k=1}^{K} e^{\gamma_k^T x}}$, $j = 1, \ldots, K$ children of the root,

- $g_{\ell|j}(x, \gamma_{j\ell}) = \dfrac{e^{\gamma_{j\ell}^T x}}{\sum_{k=1}^{K} e^{\gamma_{jk}^T x}}$, $\ell = 1, \ldots, K$ children of the root,

- Terminals (Experts)

Regression Gaussian linear reg. model,
$$\theta_{j\ell} = (\beta_{j\ell}, \sigma_{j\ell}^2), \quad Y = \beta_{j\ell}^T + \epsilon$$

Classification The linear logistic reg. model:
$$Pr(Y = 1|x, \theta_{j\ell}) = \frac{1}{1 + e^{-\theta_{j\ell}^T x}}$$
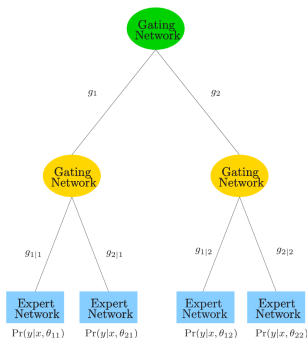
- EM algorithm

- $\Delta_i$, $\Delta_{\ell|j}$ 0–1 latent variables – branching

E step expectations for $\Delta$'s

M step estimate parameters HME by a version of multiple logistic.

# Missing data (T.D. Nielsen)

Die tossed $N$ times. Result reported via noisy telephone line. When transmission not clearly audible, record missing value:

$$4, 2, ?, 6, 5, 4, ?, 3, 4, 1, \ldots$$

"2" and "3" sound similar, therefore:

$$P(Y_i = ? | X_i = k) = P(M_i = 1 | X_i = k) = \left\{ \begin{array}{ll} 1/4 & k = 2{,}3 \\ 1/8 & k = 1{,}4{,}5{,}6 \end{array} \right.$$

Distribution of the Y is (for fair die):

| | |
|---|---|
| ? | $\frac{1}{3}\frac{1}{4} + \frac{2}{3}\frac{1}{8} = \frac{1}{6}$ |
| 2,3 | $\frac{1}{6}\frac{3}{4} = \frac{1}{8}$ |
| 1,4,5,6 | $\frac{1}{6}\frac{7}{8} = \frac{7}{48}$ |

If we simply ignore the missing data items, we obtain as the maximum likelihood estimate for the parameters of the die:

$$\theta^* = (\frac{7}{48}, \frac{1}{8}, \frac{1}{8}, \frac{7}{48}, \frac{7}{48}, \frac{7}{48}) * \frac{6}{5} = (0.175, 0.15, 0.15, 0.175, 0.175, 0.175)$$

# Incomplete data

How do we handle cases with missing values:

- Faulty sensor readings.
- Values have been intentionally removed.
- Some variables may be unobservable.

How is the data missing?
We need to take into account how the data is missing:

- **Missing completely at random** The probability that a value is missing is independent of both the observed and unobserved values (a monitoring system that is not completely stable and where some sensor values are not stored properly).

- **Missing at random** The probability that a value is missing depends only on the observed values (a database containing the results of two tests, where the second test has only performed (as a "backup test") when the result of the first test was negative).

- **Non-ignorable** Neither MAR nor MCAR (an exit poll, where an extreme right-wing party is running for parlament).

# Unsupervised Learning

- No goal class (either $Y$ nor $G$).
- We are interested in relations in the data:

Clustering  Are the data organized in natural clusters? (Clustering, Segmentation)
EM algorithm for clustering
(Dirichlet Process Mixture Models)
(Spectral Clustering)

Association Rules  Are there some frequent combinations, implication relations? (Market Basket Analysis) *later*

Other  The Elements of Statistical Learning Chapter 14

SOM  Self Organizing Maps
PCA  Principal Component Analysis Linear Algebra; $k$ linear combinations of features minimizing reconstruction error ($=$ first $k$ principal components).
- Principal Curves and Surfaces, Kernel and Spare Principal Components
ICA  Independent Component Analysis.

# Clustering Example



**Scaled Data**

**2 Clusters**

**Pitch, Yawn, Roll Clustering**
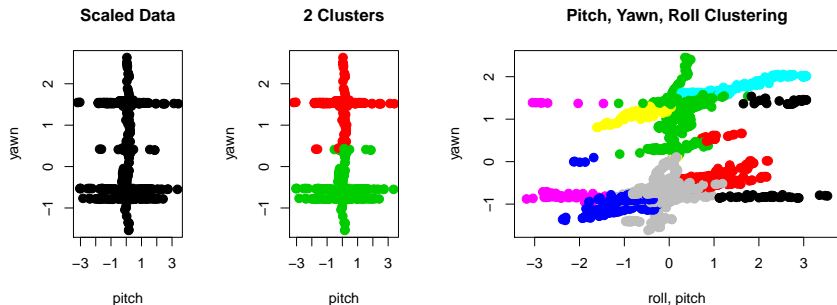
- We set the color of items, no colour in train data.
- We want to assign same color to nearby points.

# K – means !

## K–means

1: **procedure** K–SMEANS:(X data, K the number of clusters )
2:     select randomly K centers of clusters $\mu_k$
3:     # either random data points or random points in the feature space
4:     **repeat**
5:         **for** each data record **do**
6:             $C(x_i) \leftarrow argmin_{k \in \{1,...,K\}} d(x_i, \mu_k)$
7:         **end for**
8:         **for** each cluster k   **do** # find new centers $\mu_k$
9:             $\mu_k = \sum_{x_i : C(x_i) = k} \frac{x_i}{|C(k)|}$.
10:         **end for**
11:     **until** no chance in assignment
12: **end procedure**

# K – means

> **K–means**
>
> The $t$ iterations of K–means algorithm take $O(tkpN)$ time.

- To find global optimum is NP-hard.
- The result depends on initial values.
- May get stuck in local minimum.
- May not be robust to data sampling.
  - We may generate datasets by bootstrap method.
  - The cluster centers found in different dataset may be quite different.

  (for example, different bootstrap samples may give very different clustering results).
- Each record must belong to some cluster. Sensitive to outliers.

# Distance measures

the most common distance measures:

| Euclidian | $d(x_i, x_j) = \sqrt{\sum_{r=1}^{p}(x_{ir} - x_{jr})^2}$ |
|---|---|
| Hamming (Manhattan) | $d(x_i, x_j) = \sum_{r=1}^{p}|x_{ir} - x_{jr}|$ |
| overlap (překrytí) categorical variables | $d(x_i, x_j) = \sum_{r=1}^{p} I(x_{ir} \neq x_{jr})$ |
| cosine similarity | $s(x_i, x_j) = \dfrac{\sum_{r=1}^{p}(x_{ir} \cdot x_{jr})}{\sqrt{\sum_{r=1}^{p}(x_{jr} \cdot x_{jr}) \cdot \sum_{r=1}^{p}(x_{ir} \cdot x_{ir})}}$ |
| cosine distance | $d(x_i, x_j) = 1 - \dfrac{\sum_{r=1}^{p}(x_{ir} \cdot x_{jr})}{\sqrt{\sum_{r=1}^{p}(x_{jr} \cdot x_{jr}) \cdot \sum_{r=1}^{p}(x_{ir} \cdot x_{ir})}}$ |

# Other Distance Measures
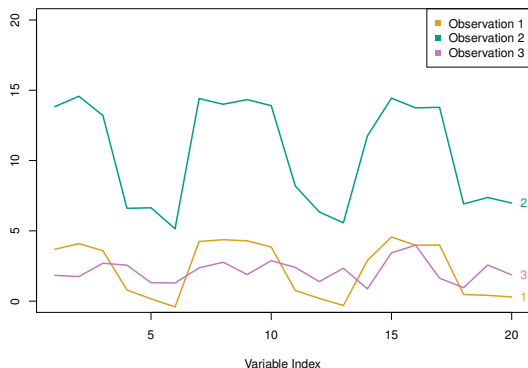


Correlation Proximity

- Euclidian distance: Observations 1 and 3 are close.
- Correlation distance: 1 and 2 look very similar.

$$\rho_{X,Y} = corr(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

# Distance – key issue, application dependent

- The result depends on the choice of distance measure $d(x_i, \mu_k)$.
- The choice is application dependent.
- Scaling of the data is recommended.
- Weights for equally important attributes $j$ are: $w_j = \frac{1}{\hat{d}_j}$ where

$$\hat{d}_j = \frac{1}{N^2} \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} d_j(x_{i_1}, x_{i_2}) = \frac{1}{N^2} \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} (x_{i_1}[j] - x_{i_2}[j])^2$$

- Total distance as a weighted sum of attribute distances.
- Distance may be specified directly by a symmetric matrix, 0 at the diagonal, should fulfill triangle inequality

$$d(x_i, x_\ell) \leq d(x_i, x_r) + d(x_r, x_\ell).$$

# Alternative Ideas



- Scaling may remove natural clusters
- Weighting Attributes
  - Consider internet shop offering socks and computers.
  - Compare: number of sales, standardized data, $
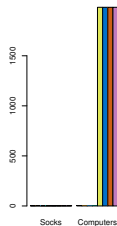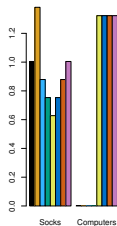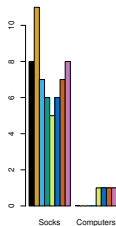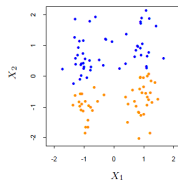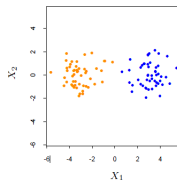
# Number of Clusters

- We may focus on the **Within cluster variation** measure:

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i^|)=k} d(x_i, x_{i^|})$$

- Notice that $W(C)$ is decreasing also for uniformly distributed data.
- We look for small drop of $W(C)$ as a function of $K$ or maximal difference between $W(C)$ on our data and on the uniform data.
- **Total** cluster variation is the sum of **between** cluster variation and **within** cluster variation

$$
\begin{aligned}
T(C) &= \frac{1}{2} \sum_{i,i^|=1}^{N} d(x_i, x_{i^|}) = W(C) + B(C) \\
&= \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \left( \sum_{C(i^|)=k} d(x_i, x_{i^|}) \right) + \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \left( \sum_{C(i^|)\neq k} d(x_i, x_{i^|}) \right)
\end{aligned}
$$

# GAP function for Number of Clusters

- denote $W_k$ the expected $W$ for uniformly distributed data and $k$ clusters, the average over 20 runs
- GAP is expected $log(W_k)$ minus observed $log(W(k))$

$$K^* = argmin\{k | G(k) \geq G(k+1) - s_{k+1}^{|}\}$$

$$s_k^{|} = s_k \sqrt{1 + \frac{1}{20}} \text{ where } s_k \text{ is the standard deviation of } log(W_k)$$
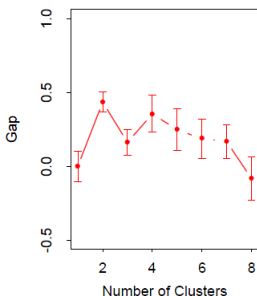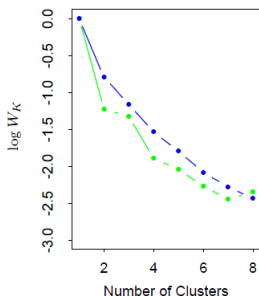
# Silhouette

For each data sample $x_i$ we define

- $a(i) = \frac{1}{|C_i|-1} \sum_{j \in C_i, i \neq j} d(i,j)$ if $|C_i| > 1$
- $b(i) = min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$

## Definition (Silhouette)

Silhouette $s$ is defined

- $s(i) = \frac{b(i)-a(i)}{max\{a(i),b(i)\}}$ if $|C_i| > 1$
- $s(i) = 0$ for $|C_i| = 1$.

Optimal number of clusters $k$ may be selected by the SC.

## Definition (Silhouette Score)

The Silhouette score is $\frac{1}{N} \sum_i^N s(i)$.

Silhouette is always between

- $-1 \leq s(i) \leq 1$.



**Silhouette analysis for KMeans clustering on sample data with n_clusters = 3**
The silhouette plot for the various clusters.     The visualization of the clustered data.
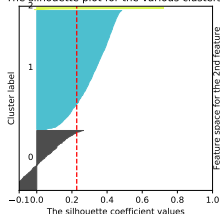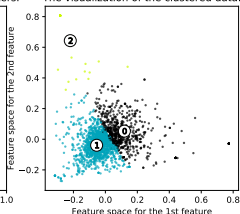
Note: One cluster $(-1,1),(1,1)$, other cluster $(0,-1.2),(0,-1.1)$, the point $(0,0)$ is assigned to the first cluster but has a negative silhouette. https://stackoverflow.com/a/66751204

# Country Similarity Example

- Data from a political science survey: values are average pairwise dissimilarities of countries from a questionnaire given to political science students.

|     | BEL  | BRA  | CHI  | CUB  | EGY  | FRA  | IND  | ISR  | USA  | USS  | YUG  |
|-----|------|------|------|------|------|------|------|------|------|------|------|
| BRA | 5.58 |      |      |      |      |      |      |      |      |      |      |
| CHI | 7.00 | 6.50 |      |      |      |      |      |      |      |      |      |
| CUB | 7.08 | 7.00 | 3.83 |      |      |      |      |      |      |      |      |
| EGY | 4.83 | 5.08 | 8.17 | 5.83 |      |      |      |      |      |      |      |
| FRA | 2.17 | 5.75 | 6.67 | 6.92 | 4.92 |      |      |      |      |      |      |
| IND | 6.42 | 5.00 | 5.58 | 6.00 | 4.67 | 6.42 |      |      |      |      |      |
| ISR | 3.42 | 5.50 | 6.42 | 6.42 | 5.00 | 3.92 | 6.17 |      |      |      |      |
| USA | 2.50 | 4.92 | 6.25 | 7.33 | 4.50 | 2.25 | 6.33 | 2.75 |      |      |      |
| USS | 6.08 | 6.67 | 4.25 | 2.67 | 6.00 | 6.17 | 6.17 | 6.92 | 6.17 |      |      |
| YUG | 5.25 | 6.83 | 4.50 | 3.75 | 5.75 | 5.42 | 6.08 | 5.83 | 6.67 | 3.67 |      |
| ZAI | 4.75 | 3.00 | 6.08 | 6.67 | 5.00 | 5.58 | 4.83 | 6.17 | 5.67 | 6.50 | 6.92 |

## K−medoids

```
 1: procedure K−MEDOIDS:( X data, K the number of clusters )
 2:     select randomly K data samples to be centroids of clusters
 3:     repeat
 4:         for each data record do
 5:             assign to the closest cluster
 6:         end for
 7:         for each cluster k  do # find new centroids i_k^* ∈ C_k
 8:             i_k^* ← argmin_{i:C(i)=k} ∑_{C(i')=k} d(x_i, x_{i'})
 9:         end for
10:     until no chance in assignment
11: end procedure
```

- To find a centroid requires quadratic time compared to linear $k$−means.
- We may use any distance, for example number of differences in binary attributes.

## Complexity

The $t$ iterations of $K$−medoids take $O(tkpN^2)$.

# Clusters of Countries

- Survey of country dissimilarities.
- Left: dissimilarities
  - Reordered and blocked according to 3-medoid clustering.
  - Heat map is coded from most similar (dark red) to least similar (bright red).
- Right: Two-dimensional multidimensional scaling plot
  - with 3-medoid clusters indicated by different colors.



Reordered Dissimilarity Matrix

First MDS Coordinate
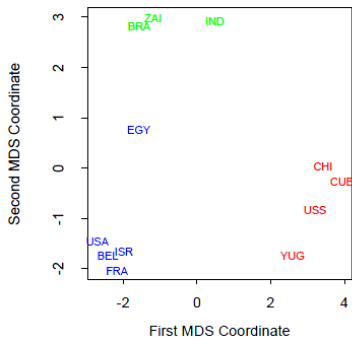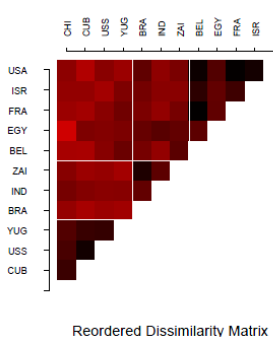
# Multidimensional Scaling

- The right figure on previous slide was done by Multidimesional scaling.
- We know only distances of countries, not a metric space.
- We try to keep proximity of countries (*least squares scaling*).
- We choose the number of dimensions $p$.

---

**Definition (Multidimensional Scaling)**

For a given data $x_1, \ldots, x_N$ with their distance matrix $d$, we search $(z_1, \ldots, z_N) \in \mathbb{R}^p$ projections of data minimizing stress function

$$S_D(z_1, \ldots, z_N) = \left[ \sum_{i \neq \ell} (d[x_i, x_\ell] - \|z_i - z_\ell\|)^2 \right]^{\frac{1}{2}}.$$

---

- It is evaluated gradiently.
- Note: Spectral clustering.

# Hierarchical clustering – Bottom Up

Start with each data sample in its own cluster. Iteratively join two nearest clusters. Measures for join

- closest points (**single linkage**)
- maximally distant points (**complete linkage**)
- **average linkage**, $d_{GA}(C_A, C_B) = \frac{1}{|C_A| \cdot |C_B|} \sum_{x_i \in C_A, x_j \in C_B} d(x_i, x_j)$
- **Ward distance** minimizes the sum of squared differences within all clusters.

$$
\begin{aligned}
Ward(C_A, C_B) &= \sum_{i \in C_A \cup C_B} d(x_i, \mu_{A \cup B})^2 - \sum_{i \in C_A} d(x_i, \mu_A)^2 - \sum_{i \in C_B} d(x_i, \mu_B)^2 \\
&= \frac{|C_A| \cdot |C_B|}{|C_A| + |C_B|} \cdot d(\mu_A, \mu_B)^2
\end{aligned}
$$

- where $\mu$ are the centers of clusters ($A$, $B$ and joined cluster).
- It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.

# Dendrograms

- Dendrogram is the result plot of a hierarchical clustering.
- Cutting the tree of a fixed high splits samples at leaves into clusters.
  - The length of the two legs of the U-link represents the distance between the child clusters.
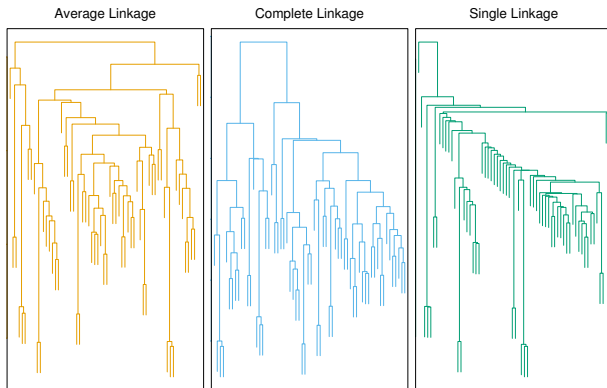
Samples fused at very bottom are close each other.
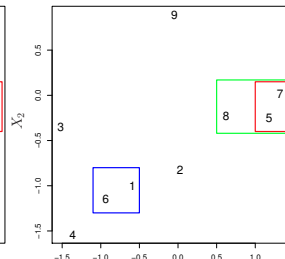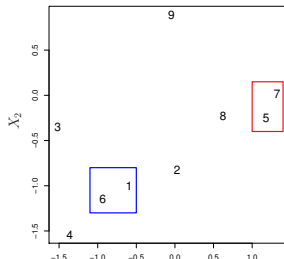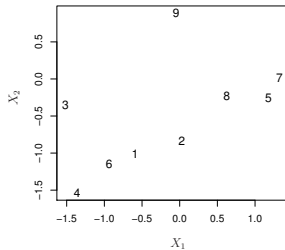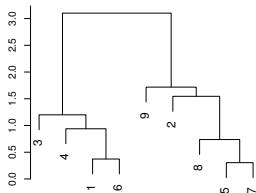
# Mean Shift Clustering

## Mean Shift Clustering

1: **procedure** MEAN SHIFT CLUSTERING:($X$ data, $K(\cdot)$ the kernel, $\lambda$ the bandwidth )
2:    $\mathcal{C} \leftarrow \emptyset$
3:    **for** each data record **do**
4:       **repeat** # shift each mean $x$ to the weighted average
5:          $m(x) \leftarrow \dfrac{\sum_{i=1}^{N} K(x_i - x) x_i}{\sum_{i=1}^{N} K(x_i - x)}$
6:       **until** no chance in assignment
7:       add the new $m(x)$ to $\mathcal{C}$
8:    **end for**
9:    return pruned $\mathcal{C}$
10: **end procedure**

Kernels:

- flat kernel $\lambda$ ball
- Gaussian kernel $K(x_i - x) = e^{\frac{\|x_i - x\|^2}{\lambda^2}}$

# Gaussian Mixture Model

- Assume the data come from a set of $k$ gaussian distributions
- each with
    - prior probability $\pi_k$
    - mean $\mu_k$
    - covariance matrix $\Sigma_k$
    - $\phi_{\mu_k, \Sigma_k}(x) = \frac{1}{\sqrt{(2\pi)^p |\Sigma_k|}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}$.
- We want to find the maximum likelihood estimate of the model parameters.
- We use (more general) EM algorithm.
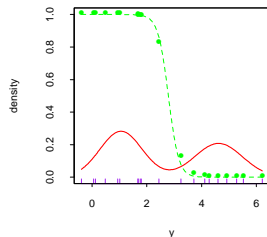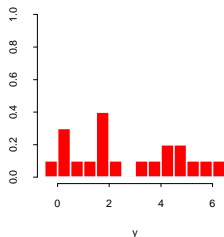
# EM learning of Mixture of $K$ Gaussians !

- Model parameters $\pi_1, \ldots, \pi_k, \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k$ such that $\sum_{k=1}^{K} \pi_k = 1$.
- **E**xpectation: weights of unobserved 'fill–ins' $k$ of variable $C$:

$$
\begin{aligned}
p_{ik} &= P(C = k | x_i) = \alpha \cdot P(x_i | C_i = k) \cdot P(C_i = k) \\
&= \frac{\pi_k \phi_{\theta_k}(x_i)}{\sum_{l=1}^{K} \pi_l \phi_{\theta_l}(x_i)} \\
p_k &= \sum_{i=1}^{N} p_{ik}
\end{aligned}
$$

- **M**aximize: mean, variance and cluster 'prior' for each cluster $k$:

$$
\begin{aligned}
\mu_k &\leftarrow \sum_i \frac{p_{ik}}{p_k} x_i \\
\Sigma_k &\leftarrow \sum_i \frac{p_{ik}}{p_k} (x_i - \mu_k)(x_i - \mu_k)^T \\
\pi_k &\leftarrow \frac{p_k}{\sum_{l=1}^{K} p_l}.
\end{aligned}
$$
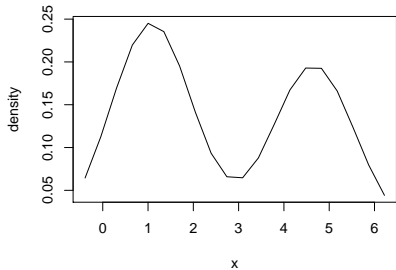
# Kernel Density Estimation

- Kernel Density Estimation is an unsupervised procedure
- We smooth the density estimate in the neighbourhood $\mathcal{N}(x_0)$ with lenghtscale $\lambda$

$$\hat{f}_X(x_0) = \frac{\#x_i \in \mathcal{N}(x_0)}{N\lambda}$$

- by the Parzen kernel estimate

$$\hat{f}_X(x_0) = \frac{1}{N\lambda} \sum_{i=1}^{N} K_\lambda(x_0, x_i),$$

- Popular choice for $K_\lambda$ is the Gaussian kernel density $\phi_\lambda$.

# Kernel Density Classification

- We may estimate Kernel Density for each target class $k = 1, \ldots, K$, estimate class priors $\pi_k$ and use Bayes' theorem:

$$\hat{P}r(G = k | X = x_0) = \frac{\pi_k \hat{f}_k(x_0)}{\sum_{j=1}^{K} \pi_j \hat{f}_k(x_0)}.$$

- by the Parzen kernel estimate

$$\hat{f}_X(x_0) = \frac{1}{N\lambda} \sum_{i=1}^{N} K_\lambda(x_0, x_i),$$

- Popular choice for $K_\lambda$ is the Gaussian kernel density $\phi_\lambda$.
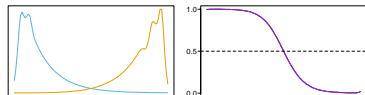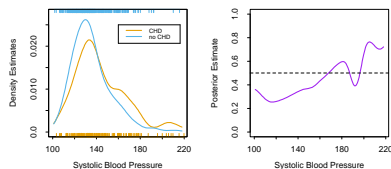


**FIGURE 6.15.** *The population class densities may have interesting structure (left) that disappears when the posterior probabilities are formed (right).*

# Radial Basis Functions and Kernels for Regression

- The kernels do not have to be placed at all observation points.
- We may select (fit) prototype parameters $\xi_j$ and scale patameters $\lambda_j$ to place pre-defined number of kernels $K_{\lambda_j}(\xi_j, x)$, $j \in 1, \ldots, M$, $\lambda_j \in \mathbb{R}$, $\xi_j \in X$.
- and then fit the density as a linear function of kernels as basis

$$f(x) = \sum_{j=1}^{M} K_{\lambda_j}(\xi_j, x)\beta_j,$$

- We should either fit the lengthscale parameters $\lambda_j$ or re-normalize the radial basis functions. Otherwise, the RBF can leave holes (upper figure, re-normalized down).
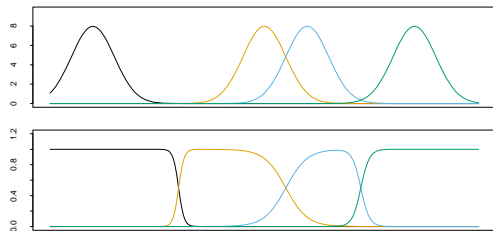
# Mixuture Models for Density Estimation and Classification

- One RBF kernel was fitted for each class
- The data sample is classified according the more probable label (let kernels vote).
- If the covariance matrices are constrained to be scalar $\Sigma_m = \sigma_m I$, we actually fit the naive Bayes model.
- In this case, this method was as good as logistic regression.
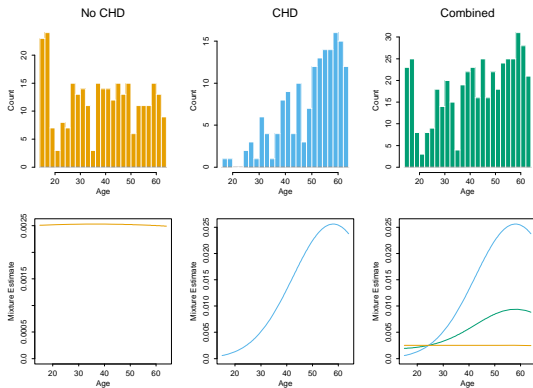
# Summary

- K-means clustering - the basic one
  - the number of clusters:
  - GAP
  - Silhouette
- The distance is crucial.
  - Consider standardization or weighting the features.
- K-medoids - does need metric, just a distance
- hierarchical clustering
  - different distance measures
  - dendrogram
- other approaches (mean shift clustering, Self Organizing Maps, Spectral Clustering).

# Frequent itemsets, Association Rules

Unsupervised learning

- No goal class (either $Y$ nor $G$).
- Usually binary data $X_{ij} \in \{0,1\}^{N \times p}$
- Value $= 1$ is our interest; for example purchase.
- $p$ may be very large; for example the size of the range of goods in an market.
- Popular application: Market basket analysis.
- Generally: We look for $L$ prototypes $v_1, \ldots, v_L \in X^p$ such that $P(v_\ell)$ is relatively large.
- With large $p$, we do not have enough data to estimate $P(v_\ell)$ since number of observations with $P(X = v_\ell)$ is too small.
- We seek for regions where $P(x)$ is large, that can be written as conjunctive rule on dimension conditions $\bigcap_{j=1}^{p}(X_j \in s_j)$ where $s_j$ are selected values of the feature $X_j$.

# Hypothesis space for Apriori

ESL book Figure:
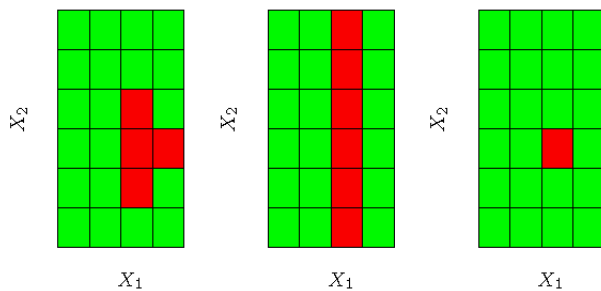


FIGURE 14.1. *Simplifications for association rules. Here there are two inputs $X_1$ and $X_2$, taking four and six distinct values, respectively. The red squares indicate areas of high density. To simplify the computations, we assume that the derived subset corresponds to either a single value of an input or all values. With this assumption we could find either the middle or right pattern, but not the left one.*

# Market Basket Analysis

- For very large datasets, $p \approx 10^4$, $N \approx 10^8$; in unit ball is the distance to the nearest neighbour $\approx 0.9981$.
- Simplifications: Test on feature $X_j$ either equal to a specific value or no restriction at all,
- I select combinations of items with higher number of occurences (**support**) than <u>predefined threshold</u> $t$.
- I select <u>all</u> combinations fulfilling conditions above.
- Categorical variables may be codded by dummy variables in advance (if not too many).
    - **OneHotEncoder** for each class $g$, a new variable $X_g = [X == g]$.

## Apriori Algorithm!

```
 1: procedure APRIORI:(X dataset, t threshold for support )
 2:     i ← 1
 3:     Generate list of candidates of the length i
 4:     while Candidate set not empty do
 5:         for each data sample do
 6:             for each candidate do
 7:                 if all items of candidate appear in the data sample then
 8:                     increase the candidate counter by 1
 9:                 end if
10:             end for
11:         end for
12:         i ← i + 1
13:         Discard candidates with support less than t.
14:         Generate list of candidates of the length i
15:             Join any two candidates from previous step having i − 2
    elements common. (More pruning possible.)
16:     end while
17: end procedure
```

# Example: Apriori Algorithm

- $t = 0.2$
- $t * N = 2 = 0.20 * 10$
- Data

| a b c e f o |
| a c g |
| e i |
| a c d e g |
| a c e g l |
| e j |
| a b c e f p |
| a c d |
| a c e g m |
| a c e g n |

**i=1**

a=8
b=2
c=8
d=2
e=8
f=2
g=5
i=j=l=o=1
p=m=n=1

**i=2**

ab=2
ac=8
ad=2
ae=6
af=2
ag=5
bc=2
bd=0
be=2
bf=2
bg=0
cd=2
ce=6
cf=2
cg=5
de=1
df=0
dg=1
ef=2

**i=3**

abc=2
abd=0
abe=2
abf=2
abg=0
acd=2
ace=6
acf=2
acg=5
ade=1
adf=0
adg=1
aeg=4
...

**i=4 ...**

abce=2
abcf=2
abef=2
abeg=0
acef=2
aceg=4
adeg=1
aefg=0
...

# Properties of the Apriori Algorithm

- Applicable for very large data (with high threshold $t$).
- The key idea:
  - Only few of $2^K$ combinations have high support $> t$,
  - **subset of high–support combination has also high support**.
- The number of passes through the data is equal to the size of the longest supported combination. The data does not to be in memory simultaneously.
- *FPgrowth* algorithm needs only two passes through the data.

- From each supported itemset $\mathcal{K}$ found by Apriori algorithm we create a list of **association rules**, implications of the form $A \Rightarrow B$ where:
    - $A, B$ are disjoint and $A \cup B = \mathcal{K}$
    - $A$ is called **antecedent**
    - $B$ is called **consequent**.
- Support of the rule $T(A \Rightarrow B)$ is defined as normalized support of the itemset $\mathcal{K}$, that is normalized support of the conjunction $A\&B$.

$$
\begin{aligned}
T(\mathcal{K}) &= \frac{|data_{\mathcal{K}}|}{|data|} \\
T(A \Rightarrow B) &= \frac{|data_{A\&B}|}{|data|}
\end{aligned}
$$

# Rule Confidence and Lift

There are two important measures for a rule $A \Rightarrow B$:

- **Confidence** (predictability, přesnost)

$$C(A \Rightarrow B) = \frac{T(A \Rightarrow B)}{T(A)}$$

  that is an estimate of $P(B|A)$,
- Support $T(B)$ is an estimate of $P(B)$,
- **Lift** is the ration of confidence and expected precision:

$$L(A \Rightarrow B) = \frac{C(A \Rightarrow B)}{T(B)}$$

  that is an estimate of $\frac{P(A\&B)}{P(A) \cdot P(B)}$.
- **Leverage** is the difference of supports:

$$leverage(A \Rightarrow B) = T(A \Rightarrow B) - T(A) \cdot T(B)$$

- **Conviction** is the ratio:

$$conviction(A \Rightarrow B) = \frac{1 - T(B)}{1 - C(A \Rightarrow B)}.$$

# Association Rule Example

ESL book example:

**Association rule 2:** Support 13.4%, confidence 80.8%, and lift 2.13.

$$
\begin{bmatrix}
\text{language in home} & = & \textit{English} \\
\text{householder status} & = & \textit{own} \\
\text{occupation} & = & \textit{\{professional/managerial\}}
\end{bmatrix}
$$
$$
\Downarrow
$$
$$
\text{income} \geq \textit{\$40,000}
$$

- $\mathcal{K} = \{\text{English, own, prof/man, income>\$40000}\}$,
- 13.4% people has all four properties,
- 80.8% of people with $\{\text{English, own, prof/man}\}$ have income$> \$40000$,
- $T(\text{income} > \$40000) = 37.94\%$, therefore *Lift* $= 2.13$.

# The Goal of Apriori Algorithm !

- Apriori finds <u>all</u> rules with <u>high support</u>.
- Frequently, it finds many of rules.
- We usually select lower threshold $c$ on confidence, that is we select rules with $T(A \Rightarrow B) > t$ and $C(A \Rightarrow B) > c$.
- Conversion of itemsets to rules is usually relatively fast compared to search of itemsets.
- See <u>lispMiner</u> for user interface and a lot of more.
- Python Apriori library:

```
from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.frequent_patterns import fpgrowth,fpmax

from mlxtend.frequent_patterns import hmine
```

# Demographical Data ESL Example

| Feature | Demographic | # Values | Type |
|---------|-------------|----------|------|
| 1 | Sex | 2 | Categorical |
| 2 | Marital status | 5 | Categorical |
| 3 | Age | 7 | Ordinal |
| 4 | Education | 6 | Ordinal |
| 5 | Occupation | 9 | Categorical |
| 6 | Income | 9 | Ordinal |
| 7 | Years in Bay Area | 5 | Ordinal |
| 8 | Dual incomes | 3 | Categorical |
| 9 | Number in household | 9 | Ordinal |
| 10 | Number of children | 9 | Ordinal |
| 11 | Householder status | 3 | Categorical |
| 12 | Type of home | 5 | Categorical |
| 13 | Ethnic classification | 8 | Categorical |
| 14 | Language in home | 3 | Categorical |

# Demographical Example – Continuing

- $N = 9409$ questionnaires, the ESL authors selected 14 questions.
- Preprocessing:
  - `na.omit()` remove records with missing values,
  - ordinal features cut by median to binary,
  - for categorical create dummy variable for each category.
- Apriori input was matrix $6876 \times 50$.
- Output: 6288 association rules
  - with max. 5 elements
  - with support at least 10%.

# Negated Literals – Useful, Problematic

**Association rule 3:** Support 26.5%, confidence 82.8% and lift 2.15.

$$
\begin{bmatrix}
\text{language in home} & = & \textit{English} \\
\text{income} & < & \textit{\$40,000} \\
\text{marital status} & = & \textit{not married} \\
\text{number of children} & = & \textit{0}
\end{bmatrix}
$$

$$\Downarrow$$

$$\text{education} \notin \{\textit{college graduate, graduate study}\}$$

# Non–frequent Values Dissapear



Relative Frequency in Association Rules
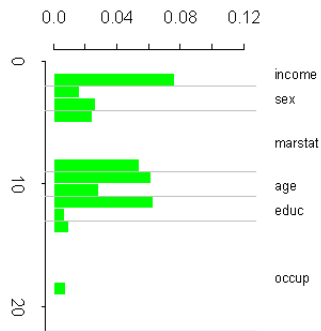
Relative Frequency in Data
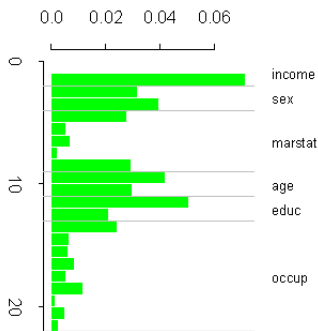
# Unsupervised Learning as Supervised Learning
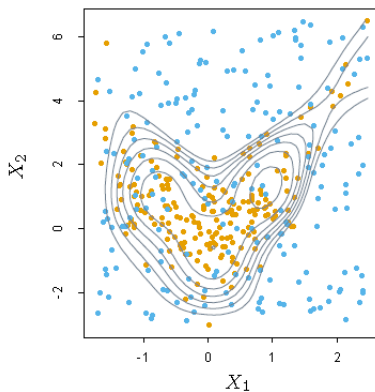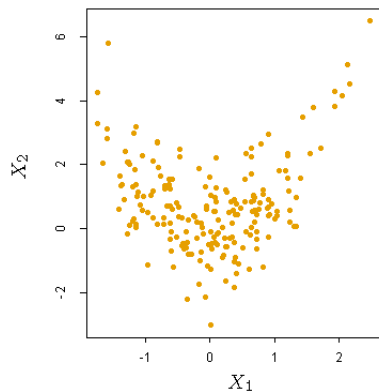


- We add additional attribute $Y_G$.
- $Y_G = 1$ for all our data.
- We generate randomly a dataset of similar size with uniform distribution, set $Y_G = 0$ for this artificial data.
- The task is to separate $Y_G = 1$ and $Y_G = 0$.

# Generalize Association Rules

- We search for high lift, where probability of conjunction is greater than expected.
- Hypothesis is specified by column indexes $j$ and subsets of values $s_j$ corresponding features $X_j$. We aim:

$$\hat{P}\left(\bigcap_{j \in \mathcal{J}}(X_j \in s_j)\right) = \frac{1}{N}\sum_1^N I\left(\bigcap_{j \in \mathcal{J}}(x_{ij} \in s_j)\right) >> \Pi_{j \in \mathcal{J}}\hat{P}(X_j \in s_j)$$

- On the data from previous slide, CART (decision tree alg.) or PRIM ('bump hunting') may be used.
- Figure on previous slide: Logistic regression on tensor product of natural splines.
- Other methods may be used. All are heuristics compared to full evaluation by Apriori.

# Frequent Pattern–tree data structure

- The number of passes through the data of Apriory is equal to the length of the longest frequent itemset.
- With an internal data structure, we are able to reduce it to 2 passes.

1: **procedure** FP–TREE:(*Data* )
2:    Calculate counts of items (singletons)
3:    Create table header ordered by decreasing item count
4:    **for** each data sample **do**
5:        order items according to header
6:        insert branch into the tree
7:        increase all counters on the inserted branch
8:    **end for**
9:    return the tree
10: **end procedure**

| Data | ordered |
|------|---------|
| a b c e f o | e c a b f |
| a c g | c a g |
| e i | e |
| a c d e g | e c a g d |
| a c e g l | e c a g |
| e j | |
| a b c e f p | |
| a c d | |
| a c e g m \\ a c e g n | |

Header table

| item | Head of node–links |
|------|---------|
| e:8 | |
| c:8 | |
| a:8 | |
| g:5 | |
| b:2 | |
| f:2 | |
| d:2 | |
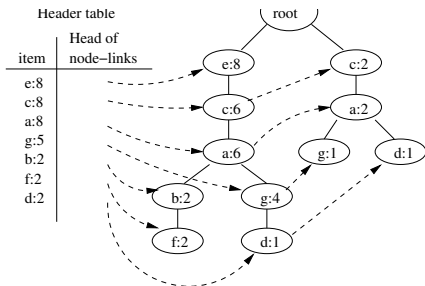
# Frequent Itemsets with only 2 pass through data

- Build an internal structure called FP-tree
- Call FP-growth to generate frequent itemsets
  - Each construction of a conditional tree needs 2 pass through the parent tree
  - an optimized version with only 1 pass is presented. (It needs an additional data structure array.)
- FP-max to find maximal itemsets
  - non of immediate supersets is frequent
- FP-close to find close itemsets
  - non of immediate supersets has the same support.

# FP-tree

- **FP-tree** contains all frequency information of the database.
- Principle: If $X$ and $Y$ are two itemsets, the count of itemsets $X \cup Y$ in the database is exactly that of $Y$ in the restriction of the database to those transactions containing $X$.



i=3
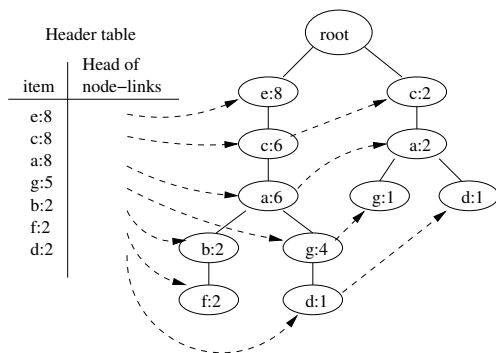
abc=2
abd=0
abe=2
abf=2
abg=0
acd=2
ace=6
acf=2
acg=5
ade=1
adf=0
adg=1
aeg=4
. . .

## FPgrowth*

```
 1: procedure FPGROWTH*:(T a conditional FP-tree )
 2:     if T only contains a single path P then
 3:         for each subpath Y of P do
 4:             output pattern Y ∪ T.base with
 5:                 count = smallest count of nodes in Y
 6:         end for
 7:     else
 8:         for each i in T.header do
 9:             Y ← T.base ∪ {i} with i.count
10:             if T.array is not NULL then
11:                 construct a new header table for Y's FP-tree from T.array
12:             else
13:                 construct a new header table for Y's from T
14:             end if
15:             construct Y's conditional FP-tree T_Y and its array A_Y;
16:             if T_Y ≠ ∅ then
17:                 call FPgrowth*(T_Y)
18:             end if
19:         end for
20:     end if
```

$$t = 2$$
$$T_{\{g\}} = [a:5, c:5, e:4]$$



(a) $A_\emptyset$
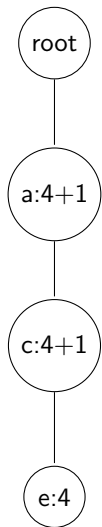
(b) $A_{\{g\}}$
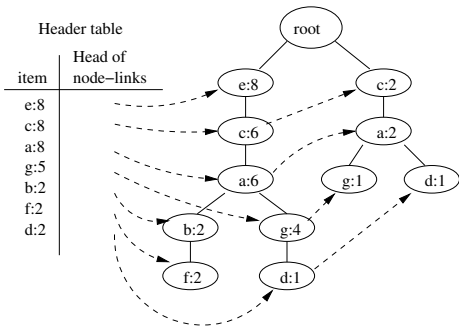
# Version Space Search

- Version space search is one of the first Machine Learning algorithms.
- For us, introduction to Inductive Logic Programming.
- Our (Tom Mitchell's) toy data:

### Example (Tennis Dataset)

| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|-----|---------|-------------|----------|--------|------------|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Overcast | Mild | High | Weak | Yes |
| D5 | Overcast | Mild | High | Strong | Yes |
| D6 | Overcast | Hot | Normal | Weak | Yes |
| D7 | Rain | Mild | High | Strong | No |

# Version Space Search

- Our **hypothesis** is a conjunction of attribute tests that imply $PlayTennis = yes$.
  - $h = \langle ?, Cold, High, ?, ?, ? \rangle$ represents the hypothesis $Temperature = cold$ & $Humidity = high \Rightarrow PlayTennis = yes$.
    - ? is satisfied by any value
    - $\emptyset$ cannot be satisfied
  - For binary attributes, we have $3^{|\#attributes|} + 1$ hypotheses
    - hypotheses with $\emptyset$ are not satisfiable, therefore they are equivalent.
    - We perform a systematic search.
    - The hypothesis space is partially ordered by the subsumption.

### Definition (More general, more specific)

- The hypothesis $h_g$ is **more general** than the hypothesis $h_g \succeq h_s$ iff any sample that satisfies $h_s$ satisfies also $h_g$.
- In the above case, the hypothesis $h_s, h_g \succeq h_s$ is called **more specific that** $h_g$.

  - $\langle ?, ?, ?, ? \rangle$ is more general than $\langle Sunny, \ldots, Same \rangle$.
  - The most general hypothesis $\langle ?, ?, ?, ? \rangle$ is satisfied by all data.
  - The most specific hypothesis $\langle \emptyset, \ldots \rangle$ is not satisfied by any data.
  - The hypothesis space for a **lattice** partially ordered by the 'more general' relation.

# Find–S

- We search for a hypothesis satisfied by all positive examples and no negative example.

## Find–S (to be improved)

```
 1: procedure FIND-S:(X dataset with the goal attritute yes/no )
 2:     h ← ⟨∅, ∅, ∅, ∅⟩ # the most specific hypothesis
 3:     for each positive data sample xᵢ do
 4:         for each attribute condition Xⱼ = xᵢ,ⱼ in h do
 5:             if xᵢ does not satisfy Xⱼ = xᵢ,ⱼ then
 6:                 replace the condition by
 7:                         a closest more general condition satisfied by xᵢ
 8:             end if
 9:         end for
10:     end for
11:     return h
12: end procedure
```

# Version Space Search

## Example (Tennis Dataset)

| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|-----|---------|-------------|----------|------|------------|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Overcast | Mild | High | Weak | Yes |
| D5 | Overcast | Mild | High | Strong | Yes |
| D6 | Overcast | Hot | Normal | Weak | Yes |
| D7 | Rain | Mild | High | Strong | No |

# Version Space

- Now we look for all hypotheses consistent with the data.

## Definition (Version Space)

- **The version space** for the hypothesis space $H$ and the data $X$ is a subset of $H$ that is consistent with $X$

$$VS(H, X) = \{h \in H | Consistent(h, X)\}.$$

- The version space is characterized by the most general and the most specific boundary.
- Any hypothesis between these boundaries is consistent with the data.

## Definition (General Boundary)

- **The general boundary** for the hypothesis space $H$ and the data $X$ is a set of most general hypothesis from $H$ that are consistent with $X$

$$G(H, X) = \{g \in H | Consistent(g, X) \& (\nexists g_1 \in H)[g_1 \succ g \& Consistent(g_1, X)]\}.$$
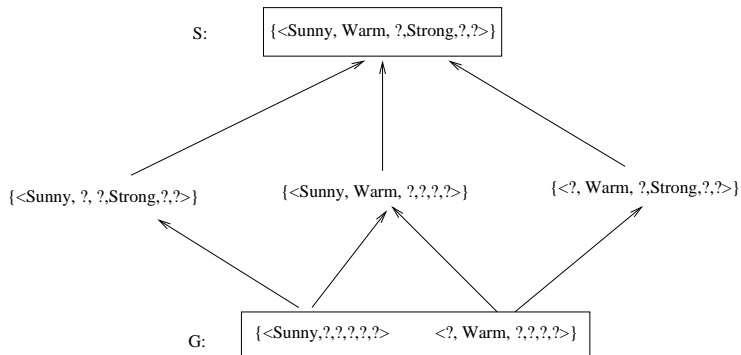
- **The specific boundary** for the hypothesis space $H$ and the data $X$ is a set of most specific hypothesis from $H$ that are consistent with $X$

$$S(H, X) = \{s \in H | Consistent(s, X) \& (\nexists s_1 \in H)[s \succ s_1 \& Consistent(s_1, X)]\}.$$



- We search for a hypothesis satisfied by all positive examples and no negative example.

1: **procedure** CANDIDATE–ELIMINATION:($X$ data,the goal att. *yes/no*)
2:     $G \leftarrow \{\langle ?, ?, ?, ? \rangle\}, S \leftarrow \{\langle \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$ # general,specific
3:     **for** each data sample $x_i$ **do**
4:         **if** $x_i$ is positive **then**
5:             remove from $G$ all $h$ inconsistent with $x_i$
6:             **for** each $s \in S$ inconsistent with $x_i$ **do**
7:                 add to $S$ all minimal generalizations $h$
8:                 $Consistent(h, x_i) \& (\exists g \in G)(g \succeq h)$
9:                 remove from $S$ $\{s | (\exists s_1 \in S)(s \succ s_1)\}$ # not most specific
10:            **end for**
11:        **else** $x_i$ is negative example
12:            remove from $S$ all $h$ inconsistent with $x_i$
13:            **for** each $g \in G$ inconsistent with $x_i$ **do**
14:                add to $G$ all minimal specifications $h$
15:                $Consistent(h, X) \& (\exists s \in S)(h \succeq s)$
16:                remove from $G$ $\{g | (\exists g_1 \in G)(g_1 \succ g)\}$ # not most gen.
17:            **end for**
18:        **end if**
19:    **end for**
20:    return $G, S$
21: **end procedure**

# Literature

- A. Cropper and S. Dumancic. Inductive logic programming at 30: a new introduction. CoRR, abs/2008.07912, 2020.
- S. Muggleton & all.: Meta-interpretive learning: application to grammatical inference, http://www.doc.ic.ac.uk/~shm/FLOC_ILP/Paper03.pdf
- Patsantzis, S., Muggleton, S.H. Top program construction and reduction for polynomial time Meta-Interpretive learning. Mach Learn 110, 755–778 (2021).
- https://www.cs.ox.ac.uk/activities/programinduction/Aleph/aleph.html
- https://github.com/stassa/louise
- https://github.com/logic-and-learning-lab/Popper
- `http://www.doc.ic.ac.uk/~shm/FLOC_ILP/Lecture1.1.pdf`

# Predicate Logic

- Recall predicate logic.
- CNF, DNF the conjunctive and disjunctive normal form
- **clause**: a disjunction of literals $father(X, Y) \vee \neg parent(X, Y) \vee \neg male(X)$
- **Horn clauses** with at most one positive literal, written as a rule
  - **definite clause** $father(X, Y) : -male(X), parent(X, Y)$.
  - **fact** - no negative literal $male(adam)$.
  - **goal clause** - no positive literal $false : -father(X, bob)$.
- **Ground term, clause** - a term, a clause without variables.
- We have our data in the form of a set of clauses $B$, $E^+$, $E^-$,
  - the background knowledge $B$ is a set of (Horn) clauses,
  - the positive and examples $E^+$, $E^-$ are sets of ground literals (facts).

## Example

$$B = \left\{ \begin{array}{c} lego\_builder(alice). \\ enjoys\_lego(A) : -lego\_builder(A). \\ estate\_agent(dave). \\ enjoys\_lego(alice). \\ enjoys\_lego(claire). \end{array} \right\}$$

$E^+ = \{ happy(alice). \}$

$$E^- = \left\{ \begin{array}{c} happy(bob). \\ happy(claire). \\ happy(dave). \end{array} \right\}$$

# Substitution, Subsumption

- Clauses ale implicitly generally quantified.
- They should not have a variable with the same name.

## Definition (Substitution, Subsumption)

- Given a **substitution** $\theta = \{v_i/t_i\}$ and formula $F$. $F\theta$ is formed by replacing every variable $v_i$ in $F$ by $t_i$.
- Substitution $\theta$ **unifies** atom $A$ and $B$ in the case $A\theta = B\theta$.
- **Atom $A$ subsumes atom $B$**, $A \succeq B$, iff there exists a substitution $\theta$ such that $A\theta = B$.
- **Clause $C$ subsumes clause $D$**, $C \succeq D$, iff there exists a substitution $\theta$ such that $C\theta \subseteq D$.

## Example

- $C_1 = f(A, B) : -head(A, B)$.
- $C_2 = f(X, Y) : -head(X, Y), empty(Y)$.
- $C_1$ subsumes $C_2$ since $C_1\theta \subseteq C_2$ with $\theta = \{A/X, B/Y\}$.

## Definition (Generalisation)

- Clause $C$ **is more general than** clause $D$, iff $C \models D$.
- Clause $C$ is more general than clause $D$ with respect to $B$, iff $B, C \models D$.
  - $B$ is the **background knowledge**.

## Example

- Statement A: Daffy Duck can fly. $can\_fly(daffy)$
- Statement B: All ducks can fly. $can\_fly(X) \succeq can\_fly(daffy)$.

## Example

- Statement C: Marek lives in London.
- Statement D: Marek lives in England.

- $lives(marek, london)$
- $lives(marek, england)$
- Background knowledge $lives(x, england) : -lives(x, london)$.
- $B, C \models D$, '$C$ is more general than $D$ with respect to $B$'.
- $C \succeq D$ with respect to $B$.
- http://www.doc.ic.ac.uk/~shm/FLOC_ILP/Lecture1.1.pdf

# ILP general logical setting

## Definition (Hypothesis Properies)

The background knowledge $B$ and the hypothesis $H$ should entail $E$, that is:

| | | | | |
|---|---|---|---|---|
| **Necessity** | $B$ | $\not\models$ | $E^+$ | we need $H$ |
| **Sufficiency, Completeness** | $B \,\&\, H$ | $\models$ | $E^+$ | $H$ explains positive examples |
| **Weak consistency** | $B \,\&\, H$ | $\not\models$ | $\perp$ | $H$ does not contradict $B$ |
| (Strong) **consistency** | $B \,\&\, H \,\&\, E^-$ | $\not\models$ | $\perp$ | ... neither negative examples |

## Definition (ILP task)

ILP task is

- Given
    - $B$ background knowledge (logic program)
    - $E^+, E^-$ examples – sets of ground unit clauses
- Given $B, E$ find a logic program $H$ such that is necessary, sufficient and consistent.
- Often, we assume noisy data and accept some errors, but we try to minimize them.

## Example

$$B = \left\{ \begin{array}{l} lego\_builder(alice). \\ lego\_builder(bob). \\ estate\_agent(claire). \\ estate\_agent(dave). \\ enjoys\_lego(alice). \\ enjoys\_lego(claire). \end{array} \right\}$$

$$E^+ = \{ happy(alice). \}$$

$$E^- = \left\{ \begin{array}{l} happy(bob). \\ happy(claire). \\ happy(dave). \end{array} \right\}$$

Our hypothesis space:

$$\mathcal{H} = \left\{ \begin{array}{l} h_1 : \ happy(A) : -lego\_builder(A). \\ h_2 : \ happy(A) : -estate\_agent(A). \\ h_3 : \ happy(A) : -enjoys\_lego(A). \\ h_4 : \ happy(A) : -lego\_builder(A), estate\_agent(A). \\ h_5 : \ happy(A) : -lego\_builder(A), enjoys\_lego(A). \\ h_6 : \ happy(A) : -estate\_agent(A), enjoys\_lego(A). \end{array} \right\}$$

- $B \cup h_1 \vDash happy(bob)$ therefore $h_1$ is inconsistent.
- $B \cup h_2 \nvDash happy(alice)$ therefore $h_2$ is incomplete.
- $B \cup h_3 \vDash happy(claire)$ therefore $h_3$ is inconsistent.
- $B \cup h_4 \nvDash happy(alice)$ therefore $h_4$ is incomplete.
- $h_5$ is both complete and consistent.
- $B \cup h_6 \nvDash happy(alice)$ therefore $h_1$ in incomplete.

# Hypothesis Space

- To specify (restrict) the hypothesis space usually mode declarations are used.

---

**Definition (Mode declarations)**

Mode declarations denote which literals may appear in the head/body of a rule. A mode declaration is of the form:

$$mode(recall, pred(m_1, m_2, \ldots, m_a))$$

where *recall* is the maximum number of occurrences of the predicate
$m_i$ are the argument types and they may be assigned as input $+$, output $-$, constant $\#$.

---

**Example**

```
modeb(2,parent(+person,-person)).
modeh(1,happy(+person)).
modeb(*,member(+list,-element)).
modeb(1,head(+list,-element)).
```

*A. Cropper and S. Dumancic. Inductive logic programming at 30: a new introduction.*

# Non-monotonic reasoning

- In Prolog, there is negation as a failure.

### Example

$$Program = \left\{ \begin{array}{c} sunny. \\ happy: -sunny, not\ weekday. \end{array} \right\}$$

- Prolog tries to prove *weekday*.
- It does not prove it, therefore it concludes *happy*.
- With additional knowledge *weekday* some of entailments are not true any more.

### Definition (Normal logic program)

Normal logic programs may include negated literals in the body of a clause, e.g.

$$h: -b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m.$$

# Aleph ILP system (based on Progol)

- Given
  - A set of mode declaration $M$
  - Background knowledge $B$ in the form of a normal program allows negation, with the semantics negation as a failure
  - Positive $E^+$ and negative $E^-$ examples as a set of ground facts
- Return: A normal program hypothesis $H$ that:
  - $H$ is consistent with $M$
  - $\forall e \in E^+$, $H \cup B \vDash e$ ($H$ is complete)
  - $\forall e \in E^-$, $H \cup B \nvDash e$ ($H$ is consistent).

## Aleph

1. Select a positive example to generalize.
2. Construct the most specific clause consistent with $M$ that entails the example (the bottom clause).
3. Search for the 'best' clause more general than the bottom clause.
4. Add the clause to the hypothesis and remove all examples covered.
5. If a positive example left, return to step 1.

# Bottom Clause Construction

## Definition (Bottom clause)

Let $H$ be a clausal hypothesis and $C$ be a clause. The bottom clause $\perp(C)$ is the most specific clause such that:

$$H \cup \perp(C) \models C.$$

- The purpose is to bound the search in the step in 3.
- Without mode declarations, the bottom clause may have infinite cardinality.

## Example (Bottom clause)

$$M = \left\{ \begin{array}{l} : -modeh(*, pos(+shape)). \\ : -modeb(*, red(+shape)). \\ : -modeb(*, square(+shape)). \\ : -modeb(*, triangle(+shape)). \\ : -modeb(*, polygon(+shape)). \end{array} \right\} \quad B = \left\{ \begin{array}{c} red(s1). \\ blue(s2). \\ square(s1). \\ triange(s2). \\ polygon(A) : -rectangle(A). \\ rectangle(A) : -square(A). \end{array} \right\}$$
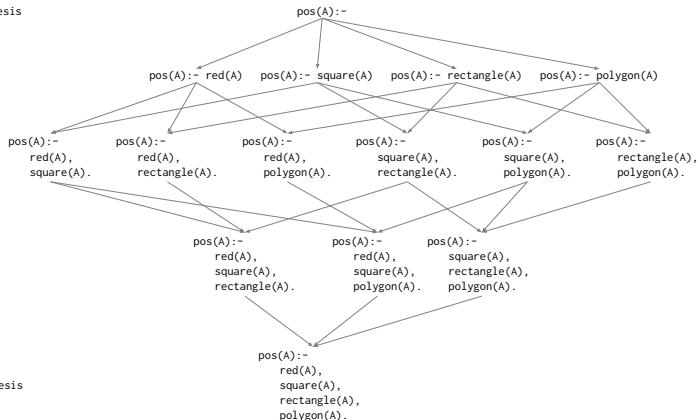
Let $e$ be the positive example $pos(s1)$. Then:

$$\perp(e) = pos(A) : -red(A), square(A), rectangle(A), polygon(A).$$

# Clause Search

- Aleph performs a bounded breadth-first search to enumerate the shorter clauses before longer ones.
- The search is bounded by several parameters (max. clause size, max. proof depth).

# Aleph 2, Popper, FlexFringe

- Aleph default evaluation function is **coverage** defined as $P - N$,
  - $P$ is the number of positive examples covered by the clause
  - $N$ is the number of negative examples covered by the clause
  - that means it accepts some noise.
- It starts from the most general one $pos(A) : -.$
- It tries to specialize the clause
  - by adding literals to the body of it, which it selects from the bottom clause
- or by instantiating variables.
- Each specialization is called **refinement**.
- Aleph Advantages
  - one Prolog file, easy to download and use.
    - https://www.cs.ox.ac.uk/activities/programinduction/Aleph/aleph.html
  - It has good empirical performance.
  - Allows numerical reasoning, user defined cost functions, handles noisy data.
- Aleph Disadvantages
  - It has many parameters to tune.
  - It struggles to learn recursive programs and optimal programs
    - since it learns only a single clause a time.

# Metagol

- Given
    - A set of metarules $M$
    - Background knowledge $B$ in the form of a normal program
    - Positive $E^+$ and negative $E^-$ examples as a set of facts (atoms).
- Return: A definite program hypothesis $H$ that:
    - $H$ is consistent with $M$
    - $\forall e \in E^+$, $H \cup B \models e$ ($H$ is complete)
    - $\forall e \in E^-$, $H \cup B \not\models e$ ($H$ is consistent)
    - $\forall h \in H$, $\exists m \in M$ such that $h = m\theta$
        - where $\theta$ is a substitution that grounds all the existentially quantified variables in $m$.

## Example (Metarule)

- An example is the **chain** metarule $P(A, B) \leftarrow Q(A, C), R(C, B)$

- that allows Metagol to induce programs such as

$$f(A, B) \quad :- \quad tail(A, C), tail(C, B).$$
$$grandparent(A, B) \quad :- \quad parent(A, C), parent(C, B).$$

# Metagol

- Metagol is a form of ILP besed on a Prolog meta-interpreter.

## Metagol

1. Select a positive example to generalize.
   - If none exists, test the hypothesis on the negative examples.
     - If the hypothesis does not entail any negative example
       stop and return the hypothesis.
     - otherwise backtrack to a choice point at step 2 and continue.
2. Try to prove the atom by:
   - using given BK or an already induced clauses
   - unifying the atom with the head of a metarule
   - binding the variables in the metarule to symbols in the predicate and constant signatures
   - save the substitution
   - try to prove the body of the metarule
     by treating the body atoms as examples and applying step 2 to them.

# Recursion

- Metagol can learn recursive programs.

## Example (Reachability)

Consider learning the concept of *reachability* in a graph. Without recursion, with the maximal depth 4 we could learn:

$$reachable(A, B) \quad :- \quad edge(A, B).$$
$$reachable(A, B) \quad :- \quad edge(A, C), edge(C, B).$$
$$reachable(A, B) \quad :- \quad edge(A, C), edge(C, D), edge(D, B).$$
$$reachable(A, B) \quad :- \quad edge(A, C), edge(C, D), edge(D, E), edge(E, B).$$

With recursion, we can learn:

$$reachable(A, B) \quad :- \quad edge(A, B).$$
$$reachable(A, B) \quad :- \quad edge(A, C), reachable(C, B).$$

# Iterative deepening

## iterative deepening

- Metagol uses iterative deepening to search for hypotheses.
  - at depth $d = 1$, at most one metasub.
  - at iteration $d$, it introduces $d - 1$ new predicate symbols and is allowed to use $d$ clauses.

# Metagol Example

**Example (Kinship example)**

$$B = \left\{ \begin{array}{c} mother(ann, amy).mother(ann, andy). \\ mother(amy, amelia), mother(amy, bob). \\ mother(linda, gavin). \\ father(steve, amy).father(steve, andy). \\ father(andy, spongebob).father(gavin, amelia). \end{array} \right\}$$

$$metarule(ident, [P, Q], [P, A, B], [[Q, A, B]]).$$
$$metarule(chain, [P, Q, R], [P, A, B], [[Q, A, C], [R, C, B]]).$$

$$E^+ = \left\{ \begin{array}{c} grandparent(ann, amelia). \\ grandparent(steve, amelia). \\ grandparent(ann, spongebob). \\ grandparent(linda, amelia). \end{array} \right\}$$
$$E^- = \{ grandparent(amy, amelia). \}$$

# Tracing Metagol

- It select the first example to generalize $grandparent(ann, amelia)$.
- It tries to prove it from BK and induced clauses. It fails.
- Metagol tries to use the first metarule:

$$grandparent(ann, amelia) : -Q(ann, amelia).$$

  stores $sub(ident, [grandparent, Q])$
- and tries to unify $Q$, but fails.
- Metagol tries to use the second metarule:

$$grandparent(ann, amelia) : -Q(ann, C), R(C, amelia).$$

  stores $sub(chain, [grandparent, Q, R])$
- and recursively tries to prove $Q(ann, C)$ and $R(C, amelia)$.
- It succeedes with the metasum $sub(chain, [grandparent, mother, mother])$
- and induces the first clause;

$$grandparent(A, B) : -mother(A, C), mother(C, B).$$

# Metagol Trace 2

- Then, it select the second example to generalize *grandparent*(*steve*, *amelia*).
- It tries to prove it from BK and induced clauses. It fails.
- Metagol can again use the second metarule with another substitution: stores *sub*(*chain*, [*grandparent*, *father*, *mother*])
- and induces the second clause;

$$grandparent(A, B) :- father(A, C), mother(C, B).$$

- Given no bound on the program size, the Metagol would prove the other two examples the same way and form the program:

$$
\begin{array}{ll}
grandparent(A, B) & :- \quad mother(A, C), mother(C, B). \\
grandparent(A, B) & :- \quad father(A, C), mother(C, B). \\
grandparent(A, B) & :- \quad father(A, C), father(C, B). \\
grandparent(A, B) & :- \quad mother(A, C), father(C, B).
\end{array}
$$

With predicate invention, it learns:

$$
\begin{array}{ll}
grandparent(A, B) & :- \quad grandparent\_1(A, C), grandparent\_1(C, B). \\
grandparent\_1(A, B) & :- \quad father(A, B). \\
grandparent\_1(A, B) & :- \quad mother(A, B).
\end{array}
$$

# Tail Recursive Metarule

> **Example (Tail Recursive Metarule)**
>
> - An example is the **tail recursive** metarule $P(A, B) \leftarrow Q(A, C), P(C, B)$
> - Metagol can also learn mutually recursive programs, such:
>
> $$
> \begin{aligned}
> even(0). \\
> even(A) \quad &:- \quad successor(A, B), even\_1(B). \\
> even\_1(A) \quad &:- \quad successor(A, B), even(B).
> \end{aligned}
> $$

We even do not have to provide the concept of an odd number. We can let the Metagol to invent such predicate ($even\_1$).

# Automata Example

(a)

| Finite acceptor | | Production rules | | | Definite Clause Grammar (DCG) | |
|---|---|---|---|---|---|---|
| | | $q_0$ | $\rightarrow$ | | $q_0([],[])$ | $\leftarrow$ |
| | | $q_0$ | $\rightarrow$ | $0\ q_0$ | $q_0([0|A],B)$ | $\leftarrow q_0(A,B)$ |
| | | $q_0$ | $\rightarrow$ | $1\ q_1$ | $q_0([1|A],B)$ | $\leftarrow q_1(A,B)$ |
| | | $q_1$ | $\rightarrow$ | $0\ q_1$ | $q_1([0|A],B)$ | $\leftarrow q_1(A,B)$ |
| | | $q_1$ | $\rightarrow$ | $1\ q_0$ | $q_1([1|A],B)$ | $\leftarrow q_0(A,B)$ |

(b)

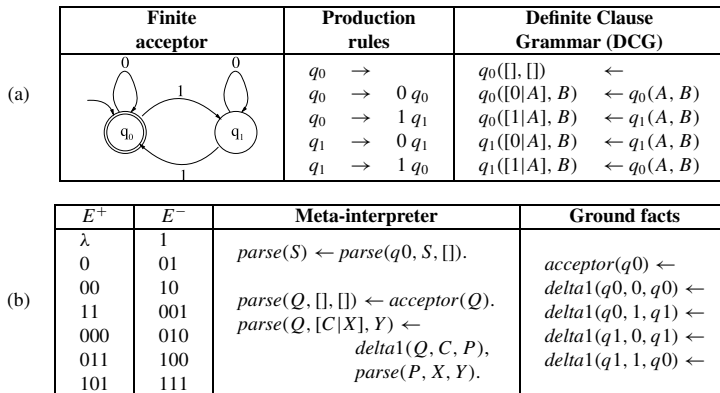| $E^+$ | $E^-$ | Meta-interpreter | Ground facts |
|---|---|---|---|
| $\lambda$ | 1 | $parse(S) \leftarrow parse(q0,S,[]).$ | |
| 0 | 01 | | $acceptor(q0) \leftarrow$ |
| 00 | 10 | $parse(Q,[],[]) \leftarrow acceptor(Q).$ | $delta1(q0,0,q0) \leftarrow$ |
| 11 | 001 | $parse(Q,[C|X],Y) \leftarrow$ | $delta1(q0,1,q1) \leftarrow$ |
| 000 | 010 | $\qquad delta1(Q,C,P),$ | $delta1(q1,0,q1) \leftarrow$ |
| 011 | 100 | $\qquad parse(P,X,Y).$ | $delta1(q1,1,q0) \leftarrow$ |
| 101 | 111 | | |

**Fig. 1** (**a**) Parity acceptor with associated production rules, DCG; (**b**) positive examples ($E^+$) and negative examples ($E^-$), Meta-interpreter and ground facts representing the Parity grammar

http://www.doc.ic.ac.uk/~shm/FLOC_ILP/Paper03.pdf

# Louise Example Grammar Learning

## Example (Module)

```
:-module(anbn, [background_knowledge/2
    ,metarules/2
    ,positive_example/2
    ,negative_example/2
    ,a/2
    ,b/2
    ]).
```

## Example (Background knowledge)

```
background_knowledge(s/2,[a/2,b/2]).
a([a|T],T).
b([b|T],T).
```

## Example (Metarules)

metarules(s/2,[chain]).
% (Chain) $\exists.P,Q,R \; \forall.x,y,z$: P(x,y)← Q(x,z),R(z,y)

# Louise Example Continued

## Example (Positive Examples)

```
positive_example(s/2,E):-
  member(E, [%s([a,b],[])
   s([a,a,b,b],[])
   ]).
```

## Example (Negative Examples)

```
negative_example(s/2,E):-
  member(E,[s([a,a],[])
   ,s([b,b],[])
   ,s([a,a,b],[])
   ,s([a,b,b],[])
   ]).
```

## Example (Parameter Tuning)

```
:- auxiliaries:set_configuration_option(clause_limit, [3]).
:- auxiliaries:set_configuration_option(max_invented, [1]).
:- auxiliaries:set_configuration_option(reduction, [none]).
```

# Louise Example Continued 2

## Example (Learned Program)

```
'$1'(A,B):-a(A,C),a(C,B).
'$1'(A,B):-a(A,C),s(C,B).
'$1'(A,B):-b(A,C),b(C,B).
'$1'(A,B):-s(A,C),b(C,B).
s(A,B):-'$1'(A,C),'$1'(C,B).
s(A,B):-'$1'(A,C),b(C,B).
s(A,B):-a(A,C),'$1'(C,B).
s(A,B):-a(A,C),b(C,B).
true.
```

## Example (Learned Program)

```
?- auxiliaries:set_configuration_option(unfold_invented, [true]).
?-make.
?- learn(s/2).
s(A,B):-a(A,C),b(C,B).
s(A,B):-a(A,C),s(C,D),b(D,B).
true.
```

# ASPAL algorithm

- ASPAL uses Answer Set Programming.
- ASP program can have one, many, or none models (answer sets).
- Computation in ASP is the process of finding models.
- We may specify the range of the number of clauses from a set beeing true.
  $0\{sunny., weekday., happy(A): -lego\_builder(A)\}3$
- We may specify an evaluation function to optimize (like to minimize the number of 'true' clauses, e.g. the size of the hypothesis.

## ASPAL

- Generate all possible rules consistent with the given mode declarations.
  Assign each rule a unique identifier and add an guessable atom in each rule.
- Use an ASP solver to find a minimal subset of the rules
  by formulating the problem as an ASP optimization problem.

## Example (ASPAL)

$$B = \left\{ \begin{array}{c} bird(alice). \\ bird(betty). \\ can(alice, fly). \\ can(betty, swim). \\ ability(fly). \\ ability(swim). \end{array} \right\} \qquad M = \left\{ \begin{array}{c} modeh(1, penguin(+bird)). \\ modeb(1, bird(+bird)). \\ modeb(*, not\ can(+bird, \#ability)). \end{array} \right\}$$

$$E^+ = \{penguin(betty).\}$$
$$E^- = \{penguin(alice).\}$$

Given the modes, the possible rules are:

$$
\begin{aligned}
penguin(X) &:- bird(X). \\
penguin(X) &:- bird(X), not\ can(X, swim). \\
penguin(X) &:- bird(X), not\ can(X, fly). \\
penguin(X) &:- bird(X), not\ can(X, swim), not\ can(X, fly).
\end{aligned}
$$

ASPAL replaces constants and adds extra literal:

$$
\begin{aligned}
penguin(X) &:- bird(X), rule(r1). \\
penguin(X) &:- bird(X), not\ can(X, C1), rule(r2, C1). \\
penguin(X) &:- bird(X), not\ can(X, C1), not\ can(X, C2), rule(r3, C1, C2).
\end{aligned}
$$

ASPAL passes to an ASP solver:

> $bird(alice)$.
>
> $bird(betty)$.
>
> $can(alice, fly)$.
>
> $can(betty, swim)$.
>
> $ability(fly)$.
>
> $ability(swim)$.
>
> $penguin(X) : -bird(X), rule(r1)$.
>
> $penguin(X) : -bird(X), not\ can(X, C1), rule(r2, C1)$.
>
> $penguin(X) : -bird(X), not\ can(X, C1), not\ can(X, C2), rule(r3, C1, C2)$.
>
> $0\{rule(r1), rule(r2, fly), rule(r2, swim), rule(r3, fly, swim)\}4$
>
> $goal : -penguin(betty), not\ penguin(alice)$.
>
> $: -not\ goal$.

- The answer is: $rule(r2, c(fly))$
- Which is translated to a program:

$$penguin(A) : -bird(A), not\ can(A, fly).$$

# ILP aplications

- Bioinformatics
  - ILP can make predictions based on the (sub)structured biological data.
  - Predict mutagenic activity of molecules and alert the causes of chemical cancers
  - learning protein folding signatures.
- Robot scientist.
  - BK knowledge represents the relationship between protein-coding sequences, enzymes, and metbolites in pathway.
  - Automatically generates hypotheses, run experiments, iterprets results.
- Games
  - Sokoban
  - Bridge
  - Checkers.

# Undirected (Pairwise, Continuous) Graphical Models

- The **generative model** represents the full probability distribution $P(X)$.
- Missing edges represent conditional independence of the variables.
- Chapter 17 Elements of Statistical Learning
  - Gaussian graphical models
  - Markov random fields
  - Ising model, (restricted) Boltzmann machine
    other not mentioned
  - Bayesian networks, Mixed interaction models, . . .

- Cytometry dataset (ESLII)
- $N = 7466$ cells
- $p = 11$ proteins
- We ame to model protein co-occurence probability.
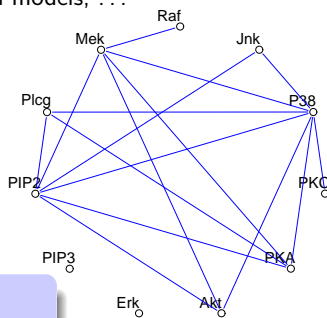


```
sklearn.covariance.GraphicalLasso # basics

gRbase # the recommended R package
```

# Sparse Conditional Gaussian Graphical Model Application

Yin, Jianxin & Li, Hongzhe. (2011). *A sparse conditional Gaussian graphical model for analysis of genetical genomics data.* The annals of applied statistics. 5. 2630-2650.

- Cytometry dataset (ESLII)
- $p_Y = 54$ gene level expressions
- $p_X = 188$ markers (discrete)
- $Y^{p_Y}|X^{p_X} \sim$
  $\mathcal{N}(M^{p_Y \times p_X} X^{p_X}, \Sigma^{p_Y \times p_Y})$
  conditional Gaussian distribution
- Top: Black color indicates significant association
  $p - value < 0.01$ in the linear regression.
- Bottom: The undirected graph of 43 genes constructed on the cGGM.



(a)
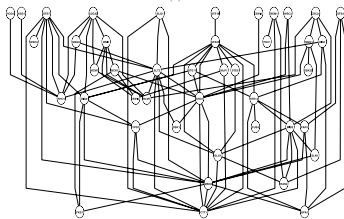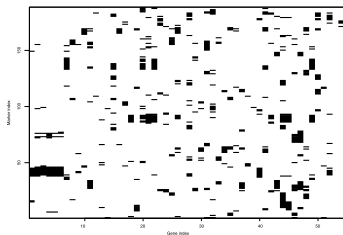
# Data: carcass

Data: carcass #Source: Soren Hojsgaard, David Edwards, Steffen Lauritzen: *Graphical Models with R*, Springer.

|  | mean. |
| --- | --- |
| Fat11 | 16.00 |
| Meat11 | 52.00 |
| Fat12 | 14.00 |
| Meat12 | 52.00 |
| Fat13 | 13.00 |
| Meat13 | 56.00 |
| LeanMeat | 59.00 |

| Σ | Fat11 | Meat11 | Fat12 | Meat12 | Fat13 | Meat13 | LeanMeat |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Fat11 | 11.34 | 0.74 | 8.42 | 2.06 | 7.66 | -0.76 | -9.08 |
| Meat11 | 0.74 | 32.97 | 0.67 | 35.94 | 2.01 | 31.97 | 5.33 |
| Fat12 | 8.42 | 0.67 | 8.91 | 0.31 | 6.84 | -0.60 | -7.95 |
| Meat12 | 2.06 | 35.94 | 0.31 | 51.79 | 2.18 | 41.47 | 6.03 |
| Fat13 | 7.66 | 2.01 | 6.84 | 2.18 | 7.62 | 0.38 | -6.93 |
| Meat13 | -0.76 | 31.97 | -0.60 | 41.47 | 0.38 | 41.44 | 7.23 |
| LeanMeat | -9.08 | 5.33 | -7.95 | 6.03 | -6.93 | 7.23 | 12.90 |

# Gaussian Graphical Models (Undirected Graphs)

- **Multivariate Gaussian Distribution** on variables $X = (X_1, \ldots, X_p)$
- $\phi(\mathbf{x}) = \frac{1}{\sqrt{|2\pi\Sigma|}} e^{-\frac{1}{2}(\mathbf{x}-\mu)\Sigma^{-1}(\mathbf{x}-\mu)}$
  - $|.|$ is the determinant. we denote $p$ the number of components in $\mathbf{x}$. Then $|2\pi\Sigma| = (2\pi)^p|\Sigma|$.
- If $\Sigma$ is not invertible it has dependent columns. It means that the variables $\mathbf{x}_j$ are lineary dependent.
  - If the **rank** of $\Sigma$ is $\ell$ then there exists a matrix $A$ and a vector $\nu$ so:
  - $x = Az + \nu$ for new coordinates $z$ with $\ell$ dimensions
  - We just consider the new coordinates and assume $\Sigma$ has a full rank.
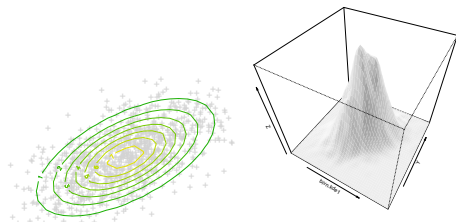
# Concentration matrix

- **Concentration (Precision, koncentrační) matrix**

$$K = \Sigma^{-1}$$

> ### Lemma
> For $u \neq v$, $k_{uv} = 0$ if and only if $y_u$ and $y_v$ are conditionally independent given all other variables.

| k*100 | Fat11 | Meat11 | Fat12 | Meat12 | Fat13 | Meat13 | LeanMeat |
|---|---|---|---|---|---|---|---|
| Fat11 | 44 | 3 | -20 | -7 | -16 | 4 | 10 |
| Meat11 | 3 | 16 | -3 | -6 | -6 | -6 | -3 |
| Fat12 | -20 | -3 | 54 | 6 | -21 | -5 | 9 |
| Meat12 | -7 | -6 | 6 | 14 | -1 | -9 | -0 |
| Fat13 | -16 | -6 | -21 | -1 | 56 | 3 | 7 |
| Meat13 | 4 | -6 | -5 | -9 | 3 | 16 | -1 |
| LeanMeat | 10 | -3 | 9 | -0 | 7 | -1 | 26 |

- If looking for small values better to 'scale' the entries into Partial Correlation matrix.

# Partial correlation matrix

**Definition (Partial correlation matrix)**

Partial correlation matrix is defined from $K$ by

$$\rho_{uv|V\setminus\{uv\}} = \frac{-k_{uv}}{\sqrt{k_{uu}k_{vv}}}.$$

**Lemma**

*In contrast to concentrations, the partial correlations are invariant under a change of scale and origin in the sense that if $X_j^* = a_j X_j + b_j$, $j = 1, \ldots, p$ then $a_v a_u k_{uv}^* = k_{uv}$ and $\rho_{uv|V\setminus\{uv\}}^* = \rho_{uv|V\setminus\{uv\}}$.*

| $\rho * 100$ | Fat11 | Meat11 | Fat12 | Meat12 | Fat13 | Meat13 | LeanMeat |
|---|---|---|---|---|---|---|---|
| Fat11 | - | -11 | 41 | 30 | 32 | -16 | -29 |
| Meat11 | -11 | - | 9 | 41 | 19 | 35 | 16 |
| Fat12 | 41 | 9 | - | -24 | 38 | 18 | -24 |
| Meat12 | 30 | 41 | -24 | - | 2 | 61 | 2 |
| Fat13 | 32 | 19 | 38 | 2 | - | -9 | -18 |
| Meat13 | -16 | 35 | 18 | 61 | -9 | - | 7 |
| LeanMeat | -29 | 16 | -24 | 2 | -18 | 7 | - |

# Models

- The simplest model just removes edges with small $|\rho_{uv|V\setminus\{uv\}}|$. Penalized criteria will be introduced later.
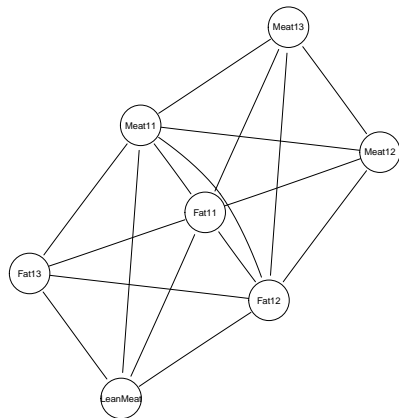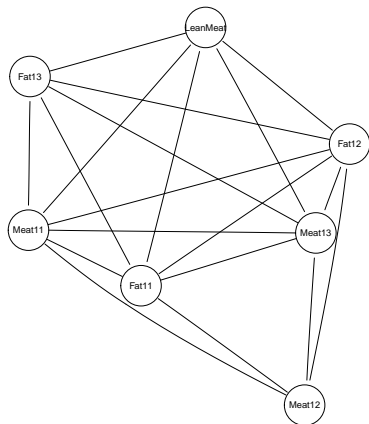
# Undirected Gaussian graphical model

## Definition (Undirected Gaussian graphical model)

An **undirected Gaussian graphical model** is represented by an undirected graph $\mathcal{G} = (X, E)$, $X = \{X_1, \ldots, X_p\}$ represent the set of variables and $E$ is a set of undirected edges.

When a random vector $\mathbf{x}$ follows a Gaussian distribution $N_p(\mu, \Sigma)$, the graph $G$ represents the model where $K = \Sigma^{-1}$ is a positive definite matrix with $k_{u,v} = 0$ whenever there is no edge between vertices $u, v$ in $G$.

This graph is called the **dependence graph** of the model.

## Lemma

*For any non adjacent vertices $u, v \in \mathcal{G}$ it holds: $u \perp\!\!\!\perp v | \mathbf{X} \setminus \{u, v\}$.*

## Definition (Generating class)

Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be the set of cliques of the dependence graph $\mathcal{G}$. A set of functions $g_1(), g_2(), \ldots, g_k()$ defined on $g_i(\mathbf{x}_{C_i})$ is called a **generating class** for the distribution

$$f(\mathbf{x}) = \prod_{i=1,\ldots,k} g_i(\mathbf{x}_{C_i}).$$

# Marginalization

- We have $\frac{1}{\sqrt{|2\pi\Sigma|}} e^{-\frac{1}{2}(\mathbf{x}-\mu)\Sigma^{-1}(\mathbf{x}-\mu)}$
- We want the distribution over variables $\{x_3, x_5, x_7\} \subset \{x_1, \ldots, x_p\}$

### Marginal of a Gaussian Distribution

The marginal of a Gaussian distribution is calculated by removing appropriate dimensions from the mean and covariance matrix.



Histogram of s1[, 2]

- $\mu_{3,5,7} = (\mu_3, \mu_5, \mu_7)$ and
$$\Sigma_{3,5,7} = \begin{bmatrix} \Sigma_{33} & \Sigma_{35} & \Sigma_{37} \\ \Sigma_{53} & \Sigma_{55} & \Sigma_{57} \\ \Sigma_{73} & \Sigma_{75} & \Sigma_{77} \end{bmatrix}$$

- $\phi_{x_3,x_5,x_7} =$
$\frac{1}{\sqrt{|2\pi\Sigma_{3,5,7}|}} e^{-\frac{1}{2}(x_{3,5,7}-\mu_{3,5,7})\Sigma_{3,5,7}^{-1}(x_{3,5,7}-\mu_{3,5,7})}$
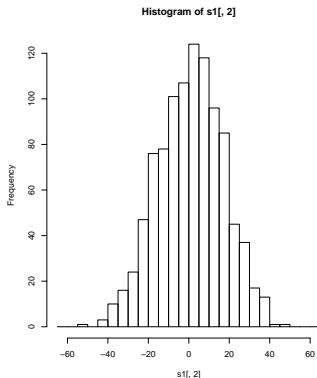
# Conditioning

- We ame for $\phi(A|B)$ where
  - $A \subset \{x_1, \ldots, x_p\}$ having $q$ elements,
  - the rest $B = \{x_1, \ldots, x_p\} \setminus A$ has $(p - q)$ elements.
- We rearrange the rows and columns to have $A$ together. Then we get

$$x = \begin{bmatrix} x_A \\ x_B \end{bmatrix} \text{ (one column), } \mu = \begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix} \text{ (one column),}$$

$$\Sigma = \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix} \text{ with dimensions } \begin{bmatrix} q \times q & q \times (p - q) \\ (p - q) \times q & (p - q) \times (p - q) \end{bmatrix}.$$

### Conditional Gaussian

The parameters of the conditional Gaussian distribution $\phi(A|B = b) = N(\mu_{A|B=b}, \Sigma_{A|B=b})$ are:

$$\begin{aligned} \mu_{A|B=b} &= \mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(b - \mu_B) \\ \Sigma_{A|B=b} &= \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}. \end{aligned}$$

Covariance matrix differs but does not depend on the observation $b$. It depends on the fact $B$ was observed.

# Conditional Gaussian Example

- $\mu^T = (1, 2, 3, 4)$

- $\Sigma = \begin{bmatrix} 10 & 1 & 5 & 4 \\ 1 & 10 & 2 & 6 \\ 5 & 2 & 10 & 3 \\ 4 & 6 & 3 & 10 \end{bmatrix}$

- We observed $(X_3, X_4)$ to be $(2.8, 4.1)$

- We ask for $\phi(A|B) = \phi(\{X_1, X_2\}|\{X_3, X_4\})$

- $\Sigma_{AB} = \begin{bmatrix} 5 & 4 \\ 2 & 6 \end{bmatrix}$

- $\Sigma_{BB} = \begin{bmatrix} 10 & 3 \\ 3 & 10 \end{bmatrix}$

- $\Sigma_{BB}^{-1} \doteq \begin{bmatrix} 0.11 & -0.033 \\ -0.033 & 0.11 \end{bmatrix}$

- $\Sigma_{AB}\Sigma_{BB}^{-1} \doteq \begin{bmatrix} 0.418 & 0.275 \\ 0.0220 & 0.593 \end{bmatrix}$

- $\mu_{A|B=b} = \mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(b - \mu_B)$

- $\mu_{A|B} \doteq \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.418 & 0.275 \\ 0.0220 & 0.593 \end{bmatrix} \begin{bmatrix} (2.8 - 3) \\ (4.1 - 4) \end{bmatrix}$

- $\mu_{A|B} \doteq \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} -0.056 \\ 0.055 \end{bmatrix} = \begin{bmatrix} 0.944 \\ 2.055 \end{bmatrix}$

- $\Sigma_{A|B=b} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$

- $\Sigma_{A|B=b} \doteq \begin{bmatrix} 10 & 1 \\ 1 & 10 \end{bmatrix} - \begin{bmatrix} 2.53 & 2.26 \\ 2.26 & 4.13 \end{bmatrix}$

- $\Sigma_{A|B=b} \doteq \begin{bmatrix} 7.47 & -1.26 \\ -1.26 & 3.65 \end{bmatrix}$

# Partition Matrix Inverse Properties

- The concentration matrix $K = \Sigma^{-1}$ is the inverse of the correlation matrix, therefore:

$$\begin{pmatrix} K_{AA} & K_{AB} \\ K_{BA} & K_{BB} \end{pmatrix} \begin{pmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{pmatrix} = \begin{pmatrix} I_{AA} & \mathbf{0} \\ \mathbf{0} & I_{BB} \end{pmatrix}$$

- From the top right part we get:

$$\begin{aligned} K_{AA}\Sigma_{AB} \quad + \quad K_{AB}\Sigma_{BB} &= \mathbf{0} \\ -K_{AA}\Sigma_{AB}\Sigma_{BB}^{-1} &= K_{AB}(1) \\ \Sigma_{AB}\Sigma_{BB}^{-1} &= -K_{AA}^{-1}K_{AB}(2). \end{aligned}$$

$$(5)$$
$$(6)$$

- Take the top left part and substitute (1):

$$\begin{aligned} K_{AA}\Sigma_{AA} \quad + \quad K_{AB}\Sigma_{BA} &= I_{AA} \\ K_{AA}\Sigma_{AA} \quad + \quad (-K_{AA}\Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}) &= I_{AA} \\ K_{AA}^{-1} &= \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}. \end{aligned}$$

# Regression Coefficients

$$\begin{aligned}
\mu_{A|B=b} &= \mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(b - \mu_B) \\
\Sigma_{A|B=b} &= \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}
\end{aligned}$$

- Consider $x_1$ to be a linear function of others with the noise $\epsilon_1 \sim N(0, \sigma_1^2)$:

$$x_{1|2\ldots p} = \beta_1 + \beta_{12}x_2 + \beta_{13}x_3 + \ldots + \beta_{1p}x_p + \epsilon_1$$

- Set $A$ the first dimension, $B$ the remaining $(p-1) \times (p-1)$ matrix:

$$x_{1|B=(x_2,\ldots,x_p)^T} = \mu_{A|B} + \Sigma_{AB}\Sigma_{BB}^{-1}\left(\begin{bmatrix} x_2 \\ \ldots \\ x_p \end{bmatrix} - \mu_B\right) + \epsilon$$

- Recall (2): $\qquad\qquad \Sigma_{AB}\Sigma_{BB}^{-1} = -K_{AA}^{-1}K_{AB}$
- then $\sigma_1^2 = \frac{1}{k_{11}}$ with coefficients $\beta$

$$(\beta_{12}, \ldots, \beta_{1p}) = -\frac{(k_{12}, \ldots, k_{1p})}{k_{11}}.$$

# Fit Linear Gaussian CPD

- To fit ML model of a linear gaussian CPD,
  - you fit the linear regression.

$$
\begin{aligned}
y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \epsilon_1 \\
\hat{\beta} &= (\mathbf{X^T X})^{-1} \mathbf{X^T y} \\
\hat{\sigma}_Y &= Cov(Y, Y) - \sum_i \sum_j \beta_i \beta_j Cov[X_i; X_j] \\
Cov(X_i; X_j) &= \mathbb{E}\left[(X_i - \mathbb{E}[X_i])(X_j - \mathbb{E}[X_j])\right] \\
\mathbb{E}[X_j] &= \frac{1}{N_{rows}} \sum_{i \in rows} x_{ij}
\end{aligned}
$$

```
from pgmpy.factors.continuous import LinearGaussianCPD
ml=maximum_likelihood_estimator(data, states)
cpdY.fit(data, states, estimator=ml, complete_samples_only=True)
```

https://cedar.buffalo.edu/~srihari/CSE674/Chap7/7.2-GaussBNs.pdf

# Parameter Learning for a Gaussian Graphical Model

- Let us have the data $\mathbf{x}_1^T, \ldots, \mathbf{x}_N^T$ over variables $\mathbf{x} \sim N_p(\mu, \Sigma)$.
- $S = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{x})(\mathbf{x}_i - \bar{x})^T$ is the empirical covariance matrix.
- Our model is represented by the concentration matrix $\Theta = \Sigma^{-1}$ and mean $\mu$.
- Log-likelihood of the data is

$$loglik(\Theta, \mu) = \frac{N}{2} \log |\Theta| - \frac{N}{2} tr(\Theta S) - \frac{N}{2} (\bar{x} - \mu)^T \Theta (\bar{x} - \mu).$$

- for a fixed $\Theta$ is the maximum for $\mu$: $\mu = \bar{x}$ and the last term is 0. We get
- $loglik(\Theta, \mu) \propto log|\Theta| - tr(\Theta S)$
- where $tr(\Theta S) = \sum_u \sum_v \theta_{uv} s_{uv}$, therefore only $s_{uv}$ corresponding to non-zero $\theta_{uv}$ are considered by the sum.
- We replace the equality conditions by Lagrange multiplyers:
  $\ell_C(\Theta) = log|\Theta| - tr(\Theta S) - \sum_{(j,k) \notin E} \gamma_{jk} \theta_{jk}$
- We maximize. The derivative $\Theta$ should be zero ($\Gamma$ is a matrix with non-zero for missing edges):

$$\Theta^{-1} - S - \Gamma = 0$$

# Towards the Algorithm

- We iterate one row/column after another.
- We start with the sample covariance matrix

$$W_0 \leftarrow S$$

- We derive the formula for the last row/column: the derivative

$$\begin{pmatrix} W_{11} & w_{12} \\ w_{12}^T & w_{22} \end{pmatrix} - \begin{pmatrix} S_{11} & s_{12} \\ s_{12}^T & s_{22} \end{pmatrix} - \begin{pmatrix} \Gamma_{11} & \gamma_{12} \\ \gamma_{12}^T & \gamma_{22} \end{pmatrix} = 0$$

- The upper right block can be written as $w_{12} - s_{12} - \gamma_{12} = 0$.
- $W$ is inverse of $\Theta$

$$\begin{pmatrix} W_{11} & w_{12} \\ w_{12}^T & w_{22} \end{pmatrix} \begin{pmatrix} \Theta_{11} & \theta_{12} \\ \theta_{12}^T & \theta_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0^T & 1 \end{pmatrix}$$

- therefore the last column without last row is:

$$w_{12} = -W_{11}\theta_{12}/\theta_{22} = W_{11}\beta$$

- Substitute into the derivative $W_{11}\beta - s_{12} - \gamma_{12} = 0$
- we solve for the rows with zero $\gamma$: $\hat{\beta}^* = (W_{11}^*)^{-1}s_{12}^*$.
- The diagonal $\theta_{22}$ is (1 bottom right): $\frac{1}{\theta_{22}} = w_{22} - w_{12}^T\beta$.

1: **procedure** GRAPHICAL REGRESSION:( $S$ sample covariance )
2:     $W \leftarrow S$ initialize
3:     **repeat**
4:         **for** $j = 1, 2, \ldots, p$ **do**
5:             Partition $W$; $j$th row and column, $W_{11}$ the rest
6:             solve $W_{11}^* \beta^* - s_{12}^* = 0$ for reduced system
7:             $\hat{\beta} \leftarrow \hat{\beta}^*$ by padding with zeros
8:             update $w_{12} \leftarrow W_{11}\hat{\beta}$
9:         **end for**
10:     **until** convergence
11:     **for** $j = 1, 2, \ldots, p$ **do**
12:         lines 5:-8: above and set
13:                 $\hat{\theta}_{22} \leftarrow \frac{1}{w_{22} - w_{12}^T \hat{\beta}}$
14:                 $\hat{\theta}_{12} \leftarrow -\hat{\beta} \cdot \hat{\theta}_{22}$
15:     **end for**
16: **end procedure**

# Example (ESLII)
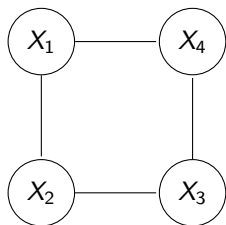


$$W_0 = S = \begin{bmatrix} 10.00 & 1.00 & 5.00 & 4.00 \\ 1.00 & 10.00 & 2.00 & 6.00 \\ 5.00 & 2.00 & 10.00 & 3.00 \\ 4.00 & 6.00 & 3.00 & 10.00 \end{bmatrix}$$

$$W_{11} = \begin{bmatrix} 10.00 & 2.00 & 6.00 \\ 2.00 & 10.00 & 3.00 \\ 6.00 & 3.00 & 10.00 \end{bmatrix} \qquad W_{22} = \begin{bmatrix} 10.00 & 1.16 & 4.00 \\ 1.16 & 10.00 & 3.00 \\ 4.00 & 3.00 & 10.00 \end{bmatrix}$$

$$W_{11}^* = \begin{bmatrix} 10.00 & 6.00 \\ 6.00 & 10.00 \end{bmatrix} \qquad W_{22}^* = \begin{bmatrix} 10.00 & 1.16 \\ 1.16 & 10.00 \end{bmatrix}$$

$$W_{11}^{*,-1} = \begin{bmatrix} 0.156 & -0.094 \\ -0.094 & 0.156 \end{bmatrix} \qquad W_{22}^{*,-1} = \begin{bmatrix} 0.101 & -0.012 \\ -0.012 & 0.101 \end{bmatrix}$$

$$\beta^* = [-0.22, 0.53]^T \qquad \beta 2^* = [0.08, 0.19]^T$$

$$\beta = [-0.22, 0, 0.53]^T \qquad \beta 2 = [0.08, 0.19, 0]^T$$

$$w_{12} \leftarrow [1.00, 1.16, 4.00]^T \qquad w_{2r} \leftarrow [1.00, 2, 0.88]^T$$

# Structure Learning

- We add a lasso penalty $||\Theta||_1$ which denotes the $L_1$ norm
  - the sum of the absolute values of the elements of $\Theta$ and we ignore the diagonal.
  - The negative penalized log-likelihood is a convex function of $\Theta$.
- we maximize penalized log-likelihood

$$log|\Theta| - tr(\Theta S) - \lambda||\Theta||_1 \qquad (7)$$

- the gradient equation is now

$$\Theta^{-1} - S - \lambda Sign(\Theta) = 0 \qquad (8)$$

  - sub-gradient notation
  - $Sign(\theta_{jk}) = sign(\theta_{jk})$ for $\theta_{jk} \neq 0$
  - $Sign(\theta_{jk}) \in [-1, 1]$ for $\theta_{jk} = 0$
- the update for the first row and column will be

$$W_{11}\beta - s_{12} + \lambda Sign(\beta) = 0 \qquad (9)$$

  - since $\beta$ and $\theta_{12}$ have opposite signs.

1: **procedure** GRAPHICAL LASSO:( $S$ sample covariance,$\lambda$ penalty )
2: $\quad W \leftarrow S + \lambda I$ initialize
3: $\quad$ **repeat**
4: $\quad\quad$ **for** $j = 1, 2, \ldots, p$ **do**
5: $\quad\quad\quad$ Partition $W$; $j$th row and column, $W_{11}$ the rest
6: $\quad\quad\quad$ solve $W_{11}\beta - s_{12} + \lambda Sign(\beta) = 0$ using the cyclical
7: $\quad\quad\quad$ ... coordinate-descent algorithm for the modified lasso
8: $\quad\quad\quad$ update $w_{12}$ by $W_{11}\hat{\beta}$
9: $\quad\quad$ **end for**
10: $\quad$ **until** convergence
11: $\quad$ **for** $j = 1, 2, \ldots, p$ **do**
12: $\quad\quad$ solve $\hat{\theta}_{22} \leftarrow \frac{1}{s_{22} - w_{12}^T \hat{\beta}}$
13: $\quad\quad$ solve $\hat{\theta}_{12} \leftarrow -\hat{\beta} \cdot \hat{\theta}_{22}$
14: $\quad$ **end for**
15: **end procedure**
16: **procedure** COORDINATEDESCENT:( $V \leftarrow W_{11}$ )
17: $\quad$ **repeat** $j = 1, 2, \ldots, p - 1$
18: $\quad\quad$ $\hat{\beta}_j \leftarrow S(s_{12j} - \sum_{k \neq j} V_{kj}\hat{\beta}_k, \lambda)/V_{jj}$
19: $\quad$ **until** convergence
20: **end procedure** $\quad\quad\quad\quad\quad\quad\quad$ #$S(x, t) = sign(x)(|x| - t)_+$

# Example (glasso)

- $\lambda \leftarrow 1$

$$W_0 = S + \lambda I = \begin{bmatrix} 11.00 & 1.00 & 5.00 & 4.00 \\ 1.00 & 11.00 & 2.00 & 6.00 \\ 5.00 & 2.00 & 11.00 & 3.00 \\ 4.00 & 6.00 & 3.00 & 11.00 \end{bmatrix}$$

$$W_{11} = \begin{bmatrix} 11.00 & 2.00 & 6.00 \\ 2.00 & 11.00 & 3.00 \\ 6.00 & 3.00 & 11.00 \end{bmatrix}$$

$$s_{12}^T = \begin{bmatrix} 1.00 & 5.00 & 4.00 \end{bmatrix}$$

$$\beta^{T,(0)} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

$$V \leftarrow W_{11}$$

$$\beta_2^{(1)} = S(1 - 0, 1)/11 = 0$$

$$\beta_3^{(1)} = S(5 - 0, 1)/11 = \frac{4}{11}$$

$$\beta_4^{(1)} = S(4 - \frac{3 \cdot 4}{11}, 1)/11 = \frac{21}{121}$$

$$\beta_2^{(2)} = S(1 - \frac{2 \cdot 4}{11} - \frac{6 \cdot 21}{121}, 1)/11 \approx -0.16$$

$$\beta_3^{(2)} = S(5 + 0.32 - \frac{3 \cdot 21}{121}, 1)/11 \approx 0.35$$

$$\beta_4^{(2)} = \ldots$$

$$\ldots$$

$$\hat{\beta}_1 \approx [-0.22; 0.32; 0.30]$$

$$W_1 \approx \begin{bmatrix} 11.00 & 0.05 & 4.03 & 3.01 \\ 0.05 & 11.00 & 2.00 & 6.00 \\ 4.03 & 2.00 & 11.00 & 3.00 \\ 3.01 & 6.00 & 3.00 & 11.00 \end{bmatrix}$$

# Graphical Lasso Properties

- Computational speed
  - The graphical lasso algorithm is extremely fast
  - can solve a moderately sparse problem with 1000 nodes in less than a minute.
  - It can be modified to have edge–specific penalty parameters $\lambda_{jk}$
  - setting $\lambda_{jk} = \infty$ will force $\hat{\theta}_{jk}$ to be zero
  - graphical lasso subsumes the parameter learning algorithm.
- Missing data
  - some missing observations may be imputed by EM algorithm from the model
  - latent – fully unobserved variables – do not bring more power in Gaussian graphical model
  - latent variables are very important in discrete distributions.

sklearn.covariance.graphical_lasso

# Model Quality (Model Selection)

### Definition (Saturated model, GGM Deviance, iDeviance, Likelihood Ratio Test)

- **saturated model** - full model with all edges, it has maximal loglikelihood
- **Deviance**

$$D = dev = 2 \cdot (\hat{\ell}_{sat} - \hat{\ell}) = N \log \frac{|S^{-1}|}{|\hat{K}|} = -N \log |S\hat{K}|$$

- **independent model** - no edges, it has minimal likelihood
- **iDeviance**

$$iD = idev = 2 \cdot (\hat{\ell} - \hat{\ell}_{ind}) = N \left( \log |\hat{K}| + \sum_{i=1}^{p} \log s_{ii} \right)$$

- **lrt likelihood ratio test** for models $\mathcal{M}_1 \subseteq \mathcal{M}_0$

$$lrt = 2 \cdot (\hat{\ell}_0 - \hat{\ell}_1) = N \log \frac{|\hat{K}_0|}{|\hat{K}_1|}.$$

# Undirected Graphical Models and Their Properties

---

**Definition (Undirected Graphical Model, Markov Graph)**

An **Undirected Graphical Model** (Markov graph, Markov network) is a graph $\mathcal{G} = (V, E)$, where nodes $V$ represent random variables and the absence of an edge $(A, B)$ denoted $A \perp\!\!\!\perp_{\mathcal{G}} B$ implies that the corresponding random variables are conditionally independent given the rest in the probability distribution $P(V)$.

$$A \perp\!\!\!\perp_{\mathcal{G}} B \implies A \perp\!\!\!\perp_P B | V \setminus \{A, B\}. \tag{10}$$

is known as the **pairwise Markov independencies** of $\mathcal{G}$.

---

**Definition (Separators)**

- If $A$, $B$ and $C$ are subgraphs, then $C$ **is said to separate $A$ and $B$** if every path between $A$ and $B$ intersects a node in $C$.

- $C$ is called a **separator**.

- Separators break the graph into conditionally independent pieces.

# Markov Properties

## Definition (Global Markov Property)

A probability measure $P$ over $V$ is **(globally) Markov** with respect to an undirected graph $\mathcal{G}$ iff for any subgraphs $A$, $B$ and $C$ holds:

- if $C$ separates $A$ and $B$ then the conditional independence $A \perp\!\!\!\perp_P B|C$ holds, that is

$$A \perp\!\!\!\perp_{\mathcal{G}} B|C \implies P(A|C) \cdot P(B|C) = P(A,B|C). \tag{11}$$

## Theorem

*The pairwise and global Markov properties of a graph are equivalent* <span style="color:red">*for graphs with strictly positive distributions*</span>.

- Gaussian distribution is always positive.
- We may infer global independence relations from simple pairwise properties.
- The global Markov property allows us to decompose graphs into smaller more manageable pieces.

# Markov Random Fields (Markovská náhodná pole)

- A probability density function $f$ over a Markov graph $\mathcal{G}$ with the set of maximal cliques $\{C_1, \ldots, C_k\}$ can be represented as

$$f(x) = \prod_{i=1,\ldots,k} \psi_i(x_{C_i}) = \psi_1(x_{C_1}) \cdot \ldots \cdot \psi_k(x_{C_k}) \tag{12}$$

- where $\psi_i$ are positive functions called **clique potentials**.
- they capture the dependence in $X_{C_i}$ by scoring certain instances $x_{C_i}$ higher than others.
- with the **normalizing constant** (partition function) $Z$

$$Z = \int_X exp\left(\sum_{i=1,\ldots,k} \log g_i(x_{C_i})\right).$$

- Such set of random variables is called **Markov Random Field** or **Markov graph**. If the potentials represent conditional probabilities with respect to some observation, it is called a **Conditional Markov Random Field**.
- For Markov networks with positive distributions the probability density function (12) implies a graph with independence properties defined by the cliques in the product.

# Pairwise Markov Graphs

- A graphical model does not always uniquely specify the higher–order dependence structure of ta joint probability distribution.

$$
\begin{aligned}
f^{(2)}(x,y,z) &= \frac{1}{Z}\psi_1(x,y)\psi_2(x,z)\psi_3(y,z) \\
f^{(3)}(x,y,z) &= \frac{1}{Z}\psi(x,y,z)
\end{aligned}
$$
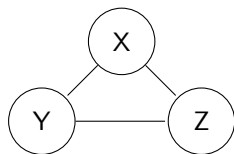


- For Gaussian distribution, pairwise interactions fully specify the model.
- We focus on **pairwise Markov Graphs**
  - where at most second order interactions are represented (like $f^{(2)}$).

# MRF for Image Denoising

- Given a noisy image $v$, perhaps with missing pixels, recover an image $u$, $u_{n,m} \in \mathbb{R}$ that is both smooth and close to $v$.

- Let each pixel be a node in a graph $\mathcal{G} = (V, E)$, with 4-connected neighborhood. Only pairwise interactions are present.

- We minimize the **energy function** (add missing margin on your own)

$$E(u) = \sum_{(m,n) \in P} (u_{m,n} - v_{m,n})^2 + \lambda \sum_{(m,n) \in P} \left[ (u_{n+1,m} - u_{n,m})^2 + (u_{n,m+1} - u_{n,m})^2 \right]$$

  We can solve $u$ iteratively

- $s_{m,n} = u_{n-1,m} + u_{n+1,m} + u_{n,m-1} + u_{n,m+1}$,

  - $u_{n,m}^{(t+1)} = \begin{cases} \frac{1}{1+4\lambda}(v_{n,m} + \lambda s_{n,m}^{(t)}) & \text{for } (n,m) \in v \\ \frac{1}{4} s_{n,m}^{(t)} & \text{for missing } v \end{cases}$



- The goal is to find the signal $u$ that minimizes the energy $E(u)$.

- Estimation by Max-flow/Min-Cut in a specific graph or Gibbs sampling.

- https://www.cs.toronto.edu/~fleet/courses/2503/fall11/Handouts/mrf.pdf
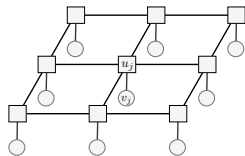
- The **Ising model** is defined by a graph $\mathcal{G} = (\mathcal{X}, E)$ of binary variables $X_i \in \mathcal{X}$ and a set of parameters $\Theta$. The joint probabilities are given by:

$$
\begin{aligned}
p(X, \Theta) &= e^{\sum_{(j,k) \in E} \theta_{jk} X_j X_k - \Phi(\Theta)} \quad \text{for } X \in \mathcal{X} \\
\Phi(\Theta) &= \log \sum_{x \in \mathcal{X}} \left[ e^{\sum_{(j,k) \in E} \theta_{jk} x_j x_k} \right].
\end{aligned}
$$

- it models only binary interactions (and unary)
- for technical reasons requires *constant* node $X_0 \equiv 1$ to be included.
- Originally from statistical mechanics.
- This model is equivalent to a first-order-interaction Poisson log-linear model for multiway tables of counts (Bishop et al., 1975).
- it implies a logistic form for each node conditional on the others

$$
P(X_j = 1 | X_{-j} = x_{-j}) = \frac{1}{1 - \exp(-\theta_{j0} - \sum_{(j,k) \in E} \theta_{jk} x_k)}.
$$

- $\Theta$ is fitted iteratively (Iterative proportional fitting, gradient descend, Poisson log-linear modeling, Mean field approximation, Gibbs sampling).

# Restricted Boltzmann Machines (RBM)

- We have visible $\mathcal{V}$ and hidden $\mathcal{H}$ variables.

- 'restricted' means the variables are organized in two layers:
  - the hidden layer
  - the visible layer that is split to input $\mathcal{V}_1$ and output variables $\mathcal{V}_2$. (pixels of image/digit label).
  - no edges inside any layer
  - an edge between each hidden and visible variable.

- The structure enables faster parameter estimation.
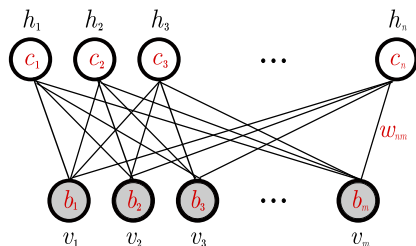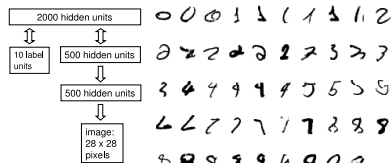


$$E(v, h) = -\sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$

## Contrastive divergence ✕

```
 1: procedure CONTRASTIVE DIVERGENCE:( S batch for update)
 2:     Δw_ij, Δb_j, Δc_i ← 0
 3:     repeat
 4:         for each training sample in a batch S do
 5:             Sample H given V_1, V_2
 6:             Sample V_1^(last), V_2^(last) given H
 7:             Sample H given V_1^(last), V_2^(last)
 8:             Δw_ij ← Δw_ij + p(H_i = 1|v^(0)) · v_j^(0) − p(H_i = 1|v^(last)) · v_j^(last)
 9:             Δb_j ← Δb_j + v_j^(0) − v_j^(last)
10:             Δc_i ← Δc_i + p(H_i = 1|v^(0)) − p(H_i = 1|v^(last))
11:         end for
12:     until convergence
13:     return Δw_ij, Δb_j, Δc_i to adjust the parameters.
14: end procedure
```

*Fischer, A., Igel, C. (2012). An Introduction to Restricted Boltzmann Machines. In: Alvarez, L., Mejail, M., Gomez, L., Jacobo, J. (eds) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. CIARP 2012.*
*Lecture Notes in Computer Science, vol 7441. Springer, Berlin, Heidelberg.*

# Example: Hidden CRF for Gesture Recognition

- Sy Bor Wang, A. Quattoni, L. . -P. Morency, D. Demirdjian and T. Darrell, "Hidden Conditional Random Fields for Gesture Recognition," 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006, pp. 1521-1527, doi: 10.1109/CVPR.2006.132.



FB          SV          EV          DB          PB          EH
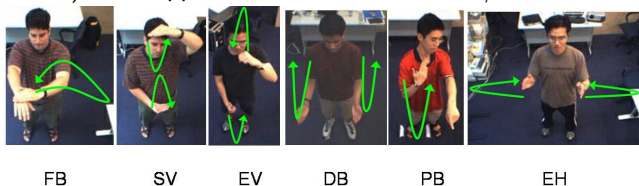
Figure 1. Illustrations of the six gesture classes for the experiments. Below each image is the abbreviation for the gesture class. These gesture classes are: FB - Flip Back, SV - Shrink Vertically, EV - Expand Vertically, DB - Double Back, PB - Point and Back, EH - Expand Horizontally. The green arrows are the motion trajectory of the fingertip and the numbers next to the arrows symbolize the order of these arrows.
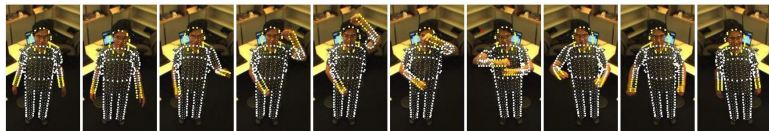


Figure 5. Sample image sequence with the estimated body pose superimposed on the user in each frame.

# HCRF Example
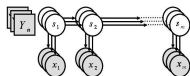


Figure 2. HMM model
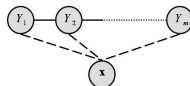


Figure 3. CRF Model
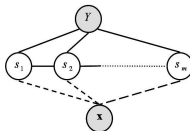


Figure 4. HCRF Model

top Hidden Markov Model for each gesture.

middle Conditional Random Field: For each frame, the gesture type $Y$ is estimated, the most frequent one is used for classification.

- Different window size $\omega$ is used.

bottom 12 hidden states for each frame are learned; the overall classification is base on these estimated states.

## Potentials

$$
\begin{aligned}
\Psi(y, \mathbf{s}, \mathbf{x}; \theta, \omega) = & \sum_{j=1} \varphi(\mathbf{x}, j, \omega) \cdot \theta_s[s_j] + \sum_{j=1} \theta_y[y, s_j] \\
& + \sum_{(j,k) \in E} \theta_e[y, s_j, s_k]
\end{aligned} \tag{3}
$$

| Models | Accuracy (%) |
|---|---|
| HCRF $\omega = 0$ | 86.44 |
| HCRF $\omega = 1$ | 96.81 |
| HCRF $\omega = 2$ | 97.75 |

| Models | Accuracy (%) |
|---|---|
| HMM $\omega = 0$ | 84.22 |
| CRF $\omega = 0$ | 86.03 |
| CRF $\omega = 1$ | 81.75 |
| HCRF (one-vs-all) $\omega = 0$ | 87.49 |
| HCRF (multi-class) $\omega = 0$ | 91.64 |
| HCRF (multi-class) $\omega = 1$ | 93.81 |

# Mixed interaction models

- Discrete and Gaussian variables together.
- *Conditional Gaussian density* for $x = (i, y)$, $i$ is the list of discrete variables, $y$ is the list of continuous variables.
- **Directed graphs**: a discrete child of a continuous parent is not allowed.
- **Undirected graphs**: If there is a path between two discrete variables $A, B$, then they are connected by a path without any continuous variable.
- $f(i, y) = p(i)(2\pi)^{-\frac{q}{2}} |\Sigma|^{-\frac{1}{2}} \exp(-\frac{1}{2}(y - \mu(i))\Sigma^{-1}(y - \mu(i)))$
- The parameters $p(i), \mu(i), i \in \mathcal{I}, \Sigma$ are called **moment parameters**
- in the exponential form we get

$$
\begin{aligned}
f(i, y) &= \exp\left\{ g(i) + h(i)^T y - \frac{1}{2} y^T K y \right\} \\
&= \exp\left\{ g(i) + \Sigma_u h_u(i) y_u - \frac{1}{2} \Sigma_{u,v} K_{u,v} y_u y_v \right\}
\end{aligned}
$$

- parameters $g(i), h(i), i \in \mathcal{I}, K$ are called **canonical parameters**.

# Mixed interaction models

- A marginal distribution is not necessarily a conditional Gaussian distribution
- it is a mixture of conditional Gaussians
- it is still tractable for evaluation
- and learning.

- Is this possible for other kind of distributions?

# Markov Properties (Zeros are dangerous)

## Definition (Markov properties: Global, Local, Pairwise)

Let $G$ be an undirected graph over $V$, let $P$ be a probability measure $P$ over $V$.

(GM) $P$ is **(globally) Markov** with respect to $\mathcal{G}$ iff

$$\forall (\mathcal{A}, \mathcal{B} \in V, \mathcal{C} \subseteq V) \; \mathcal{A} \perp\!\!\!\perp_{\mathcal{G}} \mathcal{B} | \mathcal{C} \Rightarrow \mathcal{A} \perp\!\!\!\perp_P \mathcal{B} | \mathcal{C} \text{ in P.}$$

(LM) A probability measure has the **local Markov property** iff
$(\forall A \in V) : \; A \perp\!\!\!\perp_P V \setminus Fa_A | N_A$

(PM) $P$ has the **pairwise Markov property** iff $\forall A, B \in V, A \neq B$ not connected by an edge holds $A \perp\!\!\!\perp_P B | V \setminus \{A, B\}$.

## Theorem

*These properties are equivalent for strictly positive measures.*

Counterexamples for measures with zero probability everywhere except $(0, 0, 0)$ and $(1, 1, 1)$.

See [Milan Studený:*Struktury podmíněné nezávislosti, Matfyzpress 2014*].

# Examples

> **Example** ($P$ has the pairvise but not the local property)
>
> $V = \{A, B, C\}, E = \{(b, c)\}$. Let us have a binary probability measure $V$ nonzero at points $(0, 0, 0)$ and $(1, 1, 1)$ [Studený p.101].
> $A \perp\!\!\!\perp B|\{C\}$
> $A \perp\!\!\!\perp C|\{B\}$ & does not imply $A \perp\!\!\!\perp BC|\{\}$.



> **Example** ($P$ has the local but not the global property)
>
> $V = \{A, B, C, D\}, E = \{(a, b), (c, d)\}$. Let $P(V)$ be nonzero only at points $(0, 0, 0, 0)$ and $(1, 1, 1, 1)$ [Studený p.101].
> $A \perp\!\!\!\perp CD|\{B\}$
> $B \perp\!\!\!\perp CD|\{A\}$
> $C \perp\!\!\!\perp AB|\{D\}$ & does not imply $A \perp\!\!\!\perp C|\{\}$.
> $D \perp\!\!\!\perp AB|\{C\}$

# Nonparanormal Graphical Models

- A continuous pairwise interaction model.
- We model marginal distributions,
- and the most important relations by gaussian copula.
- https://www.stat.cmu.edu/~larry/=sml/GraphicalModels.pdf

# Active Learning
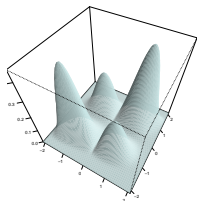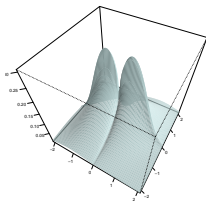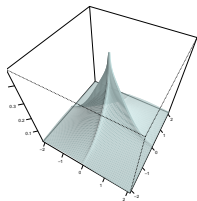
- You havee 'easy to get' unlabeled data.
- The evaluation of data is expensive.
- The task is to select the next data sample to evaluate.

## scikit-activeml

- Gaussian Processes coupled with Bayesian optimization may be viewed as a special case for active learning.

# Gaussian Processes

- An infinite (continuous) number of Gaussian variables
- to any value $x$ a new variable $N(\mu = f(x), \Sigma_{x|rest})$
- we have only a finite number of observations which means a finite number of variables
  - we can marginalize unobserved variables out (the integral is 1, we multiply by 1, we just remove),
- we can predict at any $x$, continuously.
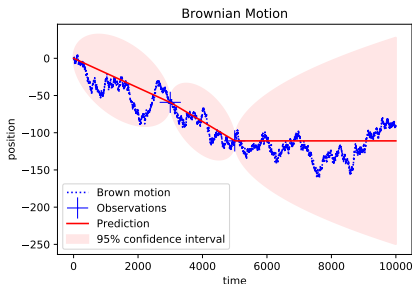
# Gaussian Processes

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006

> ### Definition (Gaussian Process)
>
> A Gaussian process is a set of random variables where any finite subset follows multivariate Gaussian distribution.

We define the mean $m(x)$ and the symmetric positive semidefinite covariance function $k(x, x')$:

$$
\begin{aligned}
m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]
\end{aligned}
$$

a Gaussian process is

$$
f(\mathbf{x}) = \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).
$$

We assume $m(\mathbf{x}) = \mathbf{0}$ it simplifies the formulas.

# Brownian Motion (Wiener Process)

## Definition (Brownian motion 1)

- $B_0 = 0$ for sure
- stationary and independent increments
- $B_s - B_t \sim N(0, s - t)$



Brownian Motion



Brownian Motion: Histogram over $10^4$ trajectories



Brown motion: $\sigma_t^2 = t$
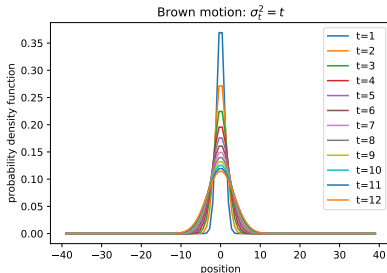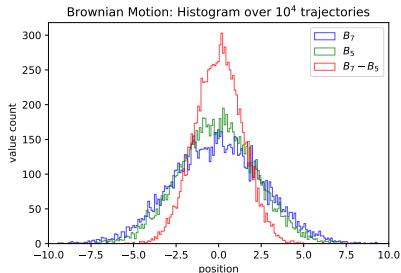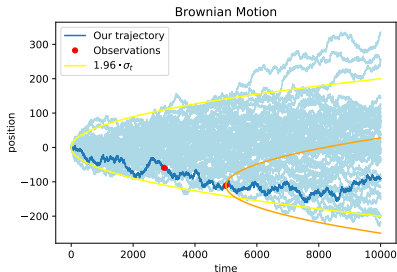
## Definition (Brownian motion 1)

- $B_0 = 0$ almost surely
- $B_t$ stationary and independent increments
- $B_s - B_t \sim N(0, s - t)$

## Definition (Brownian motion 2)

Gaussian process with

- $m = 0$ and
- $k(x, x') = min(x, x')$.

Positive semidefinite:

- $min(t, s) = \int_0^\infty f_t(x) f_s(x) dx$
- $f_t(x) f_s(x) = 1$ iff $x \in [0, t]$ & $x \in [0, s]$

## Lemma (2⇒1)

- $K(0, 0) = min(0, 0) = 0$
- The process has variance 0 at $t = 0$ and $m(0) = 0$.
- covariance is linear in both arguments, $s \geq t$

$$cov(B_s - B_t, B_s - B_t) = cov(B_s, B_s) - cov(B_t, B_s) - cov(B_s, B_t) + cov(B_t,$$
$$= s - 2t + t = s - t$$

- increments, $s \geq t \geq b \geq a$ # independence skipped, from Gaussian vectors

$$cov(B_b - B_a, B_s - B_t) = cov(B_b, B_s) - cov(B_a, B_s) - cov(B_b, B_t) + cov(B_a, B_t)$$
$$= b - a - b + a = 0.$$

# Normal Distribution

## Definition (Brownian motion 2)

Gaussian process with

- $m = 0$ and
- $k(x, x') = min(x, x')$.

- The covariance on $\mathbf{y}$ is defined by the covariance on the inputs $\mathbf{x}$.
- the covariance defines also the distribution on functions $f$:

$$\mathbf{f}_* \sim N(\mathbf{0}, K(X_*, X_*)).$$

- Without noise, we observe $\mathbf{y}$ and we want to predict $\mathbf{f}_*$:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim N \left( \mathbf{0}, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right)$$
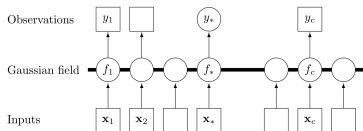
X=[3,7]
$y^T$=[ 0.5, 1.11]
K(xs,X)=[min(xs,a) for a in X]

$$K(X, X) = \begin{bmatrix} min(3, 3) & min(3, 7) \\ min(3, 7) & min(7, 7) \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 3 & 7 \end{bmatrix}$$

# Prediction

- noisy-free observations $y = f(\mathbf{x})$

$$cov(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q)$$

- noisy observations $y = f(\mathbf{x}) + \epsilon$, $\epsilon \sim N(0, \sigma_n^2)$

$$
\begin{aligned}
cov(y_p, y_q) &= k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \\
cov(\mathbf{y}) &= K(X, X) + \sigma_n^2 I
\end{aligned}
$$

- We observe $\mathbf{y}$ and we want to predict $\mathbf{f}_*$:

$$
\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim N \left( \mathbf{0}, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right)
$$

- Predictive distribution

$$
\begin{aligned}
\mathbf{f}_* | X, \mathbf{y}, X_* &\sim N(\overline{\mathbf{f}}_*, cov(\overline{\mathbf{f}}_*)) \\
\overline{\mathbf{f}}_* &\triangleq \mathbb{E}[\mathbf{f}_* | X, \mathbf{y}, X_*] = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} \mathbf{y} \\
cov(\mathbf{f}_*) &= K(X_*, X_*) - K(X_*, X)[K(X_*, X) + \sigma_n^2 I]^{-1} K(X, X_*)
\end{aligned}
$$

$$\mathbf{f}_* | X, \mathbf{y}, X_* \sim N(\bar{\mathbf{f}}_*, cov(\bar{\mathbf{f}}_*))$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_* | X, \mathbf{y}, X_*] = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} \mathbf{y}$$

$$cov(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X_*, X) + \sigma_n^2 I]^{-1} K(X, X_*)$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}\left[\mathbf{f}_* \middle| \begin{bmatrix} 3 \\ 7 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 11 \end{bmatrix}, [4]\right] = [3, 4] \begin{bmatrix} 3 + \sigma_n^2 & 3 \\ 3 & 7 + \sigma_n^2 \end{bmatrix}^{-1} \begin{bmatrix} 0.5 \\ 11 \end{bmatrix}$$

$$cov(\mathbf{f}_*) = min(4, 4) - [3, 4] \begin{bmatrix} 3 + \sigma_n^2 & 3 \\ 3 & 7 + \sigma_n^2 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$



Brownian Motion
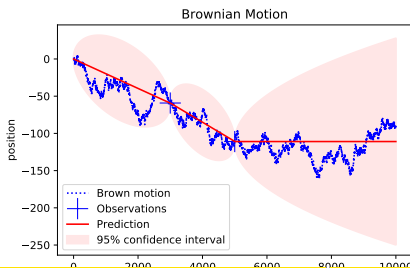
# Predictive distribution

## Prediction

- is a linear function of observations $\mathbf{y}$
  - for $\alpha \Leftarrow (K + \sigma_n^2 I)^{-1} \mathbf{y}$
  - we predict
    $\overline{f}(\mathbf{x}_*) \Leftarrow \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*)$

- The red vertical bars show the variance due to the observation noise.



Brownian Motion, $\sigma_{noise} = 25$
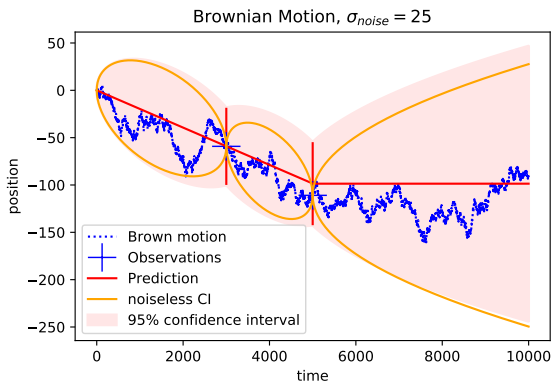
## Definition (First Set of Kernel Functions)

- **Radial Basis Function (RBF)** covariance function with the **length scale** parameter $\ell$ is defined

$$cov(f(\mathbf{x}_p), f(\mathbf{x}_q)) = RBF(\mathbf{x}_p, \mathbf{x}_q) = \exp\left(-\frac{1}{2}\frac{|\mathbf{x}_p - \mathbf{x}_q|^2}{\ell^2}\right).$$

- **Constant** covariance function with the **constant** parameter is defined

$$cov(f(\mathbf{x}_p), f(\mathbf{x}_q)) = Constant(\mathbf{x}_p, \mathbf{x}_q) = constant.$$

- **Squared exponential (SE)** covariance function with hyperparameters $\ell^2$ lenghtscale and $\sigma_f^2$ signal variance

$$\begin{aligned} k(\mathbf{x}_p, \mathbf{x}_q) &= \sigma_f^2 \exp\left(-\frac{1}{2}\frac{|\mathbf{x}_p - \mathbf{x}_q|^2}{\ell^2}\right) \\ &= Constant(\mathbf{x}_p, \mathbf{x}_q) * RBF(\mathbf{x}_p, \mathbf{x}_q) \end{aligned}$$

- can be defined as a **product kernel** of the Constant and RBF kernels.
- There is also a **sum kernel** kernel function $+$.

# Scikitlearn Examples

- Noiseless observations.

- The red vertical bars show the variance due to the observation noise.
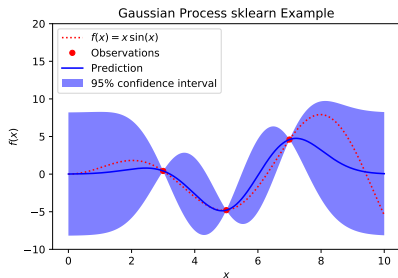


- The parameters may be fitted by the gradient update.
- The observation noise *alpha* may be specific for each observation (right), identically 0 (left) or constant.
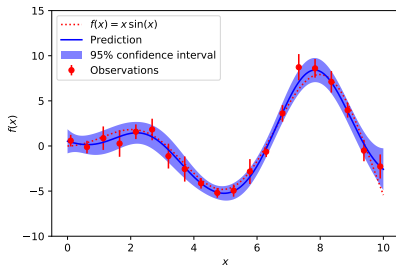
```
kernel = C(1.0, (1e-3, 1e3)) * RBF(10, (100e-2, 100e2))
gp = GaussianProcessRegressor(kernel=kernel, alpha=dy ** 2)
gp.fit(X, y)
```

# Marginal likelihood

- The parameters may be automatically tuned by gradiently maximize the marginal likelihood.
- 'In sample' prediction $\mathbf{f}$ follows: $\mathbf{f} \sim N(\mathbf{0}, K(X, X))$.

---

**Lemma**

*The marginal log likelihood is*

- *for noisy-free observations $\mathbf{y} = \mathbf{f}$:*

$$\log p(\mathbf{y}|X) = \log p(\mathbf{f}|X) = -\frac{1}{2}\mathbf{f}^T K^{-1}\mathbf{f} - \frac{1}{2}\log|K| - \frac{N}{2}\log 2\pi$$

- *For noisy observations $\mathbf{y}|\mathbf{f} \sim N(\mathbf{f}, \sigma_n^2 I)$, $\mathbf{y} \sim N(0, K + \sigma_n^2 I)$*

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T (K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{N}{2}\log 2\pi.$$

---

The noise level may be tuned as well by (sum)adding the **WhiteKernel**.
**WhiteKernel**= *noise_level* iff we address the same variable $(\mathbf{x}_p, \mathbf{x}_p)$, otherwise *WhiteKernel* = 0.

# Hyperparameter Fit

Scikitlearn example:

- The log-marginal function has two local maxima.
- The log-marginal maxima corresponds to the two models.



Log-marginal-likelihood
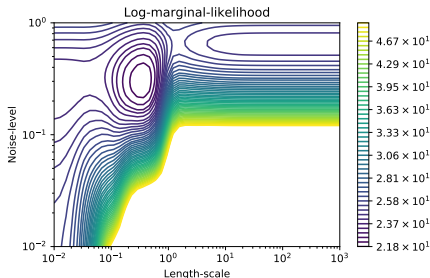
Initial: 1**2 * RBF(length_scale=100) + WhiteKernel(noise_level=1)
num: 0.00316**2 * RBF(length_scale=109) + WhiteKernel(noise_level=0.
Log-Marginal-Likelihood: -23.87233736198489
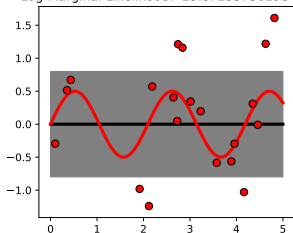
Initial: 1**2 * RBF(length_scale=1) + WhiteKernel(noise_level=1e-05)
num: 0.64**2 * RBF(length_scale=0.365) + WhiteKernel(noise_level=0.
Log-Marginal-Likelihood: -21.80509089016203
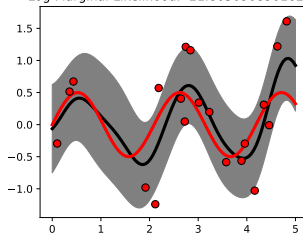
## Definition (Further Kernel Functions)

- **ExpSineSquared** kernel function with the parameters **length scale** $\ell$ and the **periodicity** $p > 0$ ($d$ is the distance) is defined

$$cov(f(\mathbf{x}_q), f(\mathbf{x}_r)) = \exp\left(-\frac{2\sin^2(\pi d(\mathbf{x}_q, \mathbf{x}_r)/p)}{\ell^2}\right).$$

  Usefull for periodic functions.

- **Dot product** kernel function with the **inhomogenicity** parameter $\sigma_0$ is defined

$$cov(f(\mathbf{x}_p), f(\mathbf{x}_q)) = \sigma_0 + \mathbf{x}_p \cdot \mathbf{x}_q.$$

  Useful to capture the trend, often combined with exponential kernel.

- **Rational Quadratic** kernel function with hyperparameters $\ell^2$ lenghtscale and mixture $\alpha$

$$k(\mathbf{x}_p, \mathbf{x}_q) \quad = \quad \left(1 + \frac{d(\mathbf{x}_p, \mathbf{x}_q)^2}{2\alpha\ell^2}\right)^{-\alpha}$$

  The mixure of many RBF kernel lengthscales.

# Conditional Covariance

- Consider the conditional covariance, the relation of two unobserved points $x$ and $x_0$.

**left** Noiseless Brown motion example. The covariance is zero outside the $x_0$ closest observations interval.

**right** Brown motion with a high noise level.



Brownian Motion, $\sigma_{noise} = 25$



Conditional covariance $cov(x, x_0|X)$



Conditional covariance $cov(x, x_0|X)$, $\sigma_{noise} = 25$
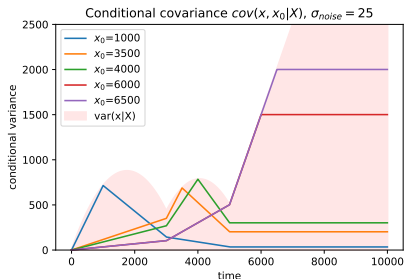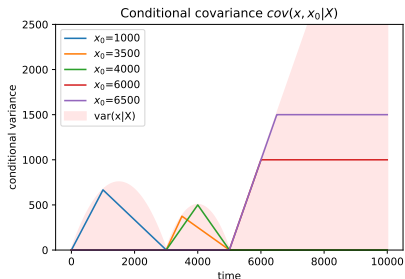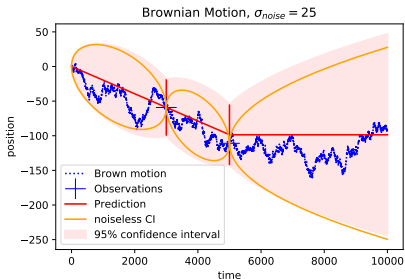
# Conditional Covariance Rassmussen Example

- The conditional covariance may be also negative.
- Most kernels have a continuous first derivative. This makes the conditional covariance negative with points on the other side of the closest observation.
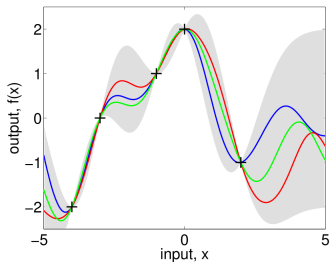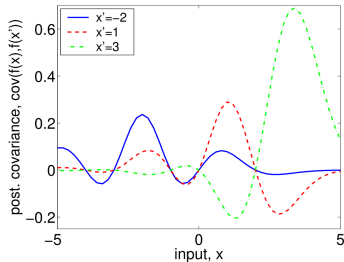


(a), posterior        (b), posterior covariance

# Matérn

- Most kernel function have many derivatives.
- The Matérn kernel $\nu = 1.5$ ('nu') has only the first derivative. It is able to model less smooth functions.
- As $\nu \to \infty$, it becomes a RBF kernel.

---

### Definition (Matérn kernel)

The **Matérn** kernel with parameters $\nu = k + \frac{1}{2}$ and $\ell$ is defined

$$k(\mathbf{x}_p, \mathbf{x}_q) = \frac{1}{2^{\nu-1}\Gamma(\nu)} \left( \frac{\sqrt{2\nu}}{\ell} d(\mathbf{x}_p, \mathbf{x}_q) \right)^\nu K_\nu \left( \frac{\sqrt{2\nu}}{\ell} d(\mathbf{x}_p, \mathbf{x}_q) \right)$$

The **Modified Bessel functions** (for $\alpha$ not integer, the limit otherwise) are defined

- $I_\alpha(x) = \sum_{m=0}^\infty \frac{1}{m!\Gamma(m+\alpha+1)} \left( \frac{x}{2} \right)^{2m+\alpha}$
- $K_\alpha(x) = \frac{\pi}{2} \frac{I_{-\alpha}(x) - I_\alpha(x)}{\sin\alpha\pi}$.

# Bayesian Optimization

- Bayesian Optimization is used when
  - We are solving: $x^* = \arg\min_x f(x)$
  - $f(x)$ is a black box function
  - $f$ is expensive to evaluate
  - the evaluations may be noisy.
- If any condition is not true, a better algorithm exists.
- We search the point $\mathbf{x}$ to observe.
- scikit-optimize = skopt Python package
- we minimize $\mathbf{y}$ and search the maximal probability of improvement
- 'the chance to improve' is expressed by the **Expected improvement** ($EI$)

---

### Bayesian Optimization Algorithm

- Evaluate $\mathbf{y}$ on $X$, let $\mathbf{y} = y(X)$ and calculate conditional means and covariances
- repeat forever
  - $x^{new} = argmax_x EI(x)$ add $x$ into $X$
  - Evaluate $\mathbf{y} = y(x)$ and add $y$ to $\mathbf{y}$.
  - re–estimate the Gaussian process (the parameters of the covariance).

# Bayesian Optimization Example [Skopt]
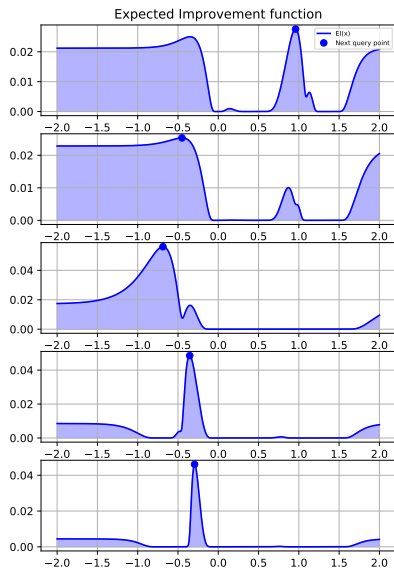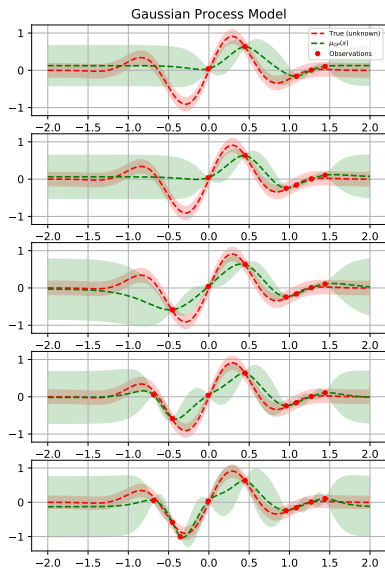
# Expected Improvement Aquisition Function

- we search the point $\mathbf{x}$ to observe
- we minimize $y$, we already have the training data $X$, $\mathbf{y}$
- the search the maximal probability of improvement is expressed by the **Expected improvement** ($EI$)

$$
\begin{aligned}
EI(x) &= \mathbb{E}[(min(Y(X)) - Y(x))^+ | Y(X) = \mathbf{y}] \\
&= \mathbb{E}[(min(\mathbf{y}) - Y(x))^+ | Y(X) = \mathbf{y}]
\end{aligned}
$$

this can be solved analytically ($\Phi$ cummulative df, $\phi$ pdf Gaussian distribution):

$$
EI(x) = (min(\mathbf{y}) - \mu(x))\Phi\left(\frac{min(\mathbf{y}) - \mu(x)}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{min(\mathbf{y}) - \mu(x)}{\sigma(x)}\right)
$$

to maximize $\mathbf{y}$:

$$
EI(x) = (\mu(x) - max(\mathbf{y}))\Phi\left(\frac{\mu(x) - max(\mathbf{y}) - \xi}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{\mu(x) - max(\mathbf{y}) - \xi}{\sigma(x)}\right).
$$

- if 'xi' $\xi > 0$ we ignore small improvements.

# Paralelization: The Constant Liar

$$
\begin{aligned}
EI(x) &= \mathbb{E}[(min(Y(X)) - min(Y(x^{(n+1)}), Y(x^{(n+2)}), \ldots, Y(x^{(n+k)})))^+ | Y(X) = \mathbf{y}] \\
&= \mathbb{E}[(min(\mathbf{y}) - Y(x))^+ | Y(X) = \mathbf{y}]
\end{aligned}
$$

it does not have direct formula. It is solved by Markov Chain simulation.

- We estimate the observations $y$ by an estimate (min, max, mean)
- and run the evaluation in parallel.

That means the covariance is correctly estimated, the mean must be corrected later.

ParBayesianOptimization R package

---

**Definition (Other Aquisition Functions)**

- Probability of Improvement: $PI(f(x_*) < min(\mathbf{y})) = \Phi\left(\frac{min(\mathbf{y}) - \mu(x_*)}{\sigma(x_*)}\right)$
- Lower Confidence Bound: $LCB(x) = \mu(x) - \kappa \cdot \sigma(x)$.

# GP for Classification

- GP for classification are more complex and 'only an approximation'
- still, it is worth to try the
  `sklearn.gaussian_process.GaussianProcessClassifier` .
- We estimate a latent function $f$ as before
- we link it to $\langle 0, 1 \rangle$ interval by the sigmoid function (or $\Phi$).
- The log-marginal-likelihood does not have a closed analytical form anymore.
- can be approximated by Hessian matrix, the algorithm works in $O(N^3)$, not too bad.

# POMDP Applications
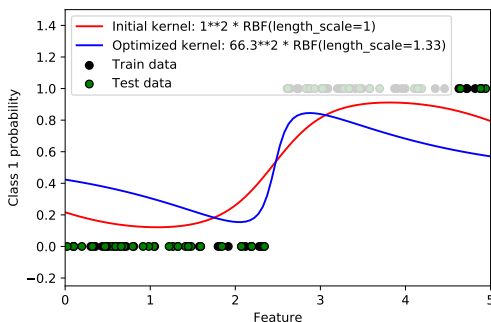
- Karkus, Hsu, Lee: **QMDP-Net: Deep Learning for Planning under Partial Observability**
  https://proceedings.neurips.cc/paper/2017/file/e9412ee564384b987d086df32d4
  Paper.pdf

- Eric Mueller and Mykel J. Kochenderfer :**Multi-Rotor Aircraft Collision Avoidance using Partially Observable Markov Decision Processes**,
  American Institute of Aeronautics and Astronautics
  https://aviationsystemsdivision.arc.nasa.gov/publications/2016/AIAA-2016-3673.pdf

# POMDP Aircraft Collision Avoidance

- the algorithms designed for fixed-wing aircraft analyze
  - turns
  - vertical meneuvers
- multirotor aircraft (drones) and helicopters can also
  - horizontal plane accelerations
- state 2D, (3D)
  - relative range states $r_x$, $r_y$, ($r_z$)
  - velocities for the ownship $v_{ox}$, $v_{oy}$, ($v_{oz}$)
  - velocities for the intruder $v_{ix}$, $v_{iy}$, ($v_{iz}$)
  - absolute displacement from the desired trajectory $d_x$, $d_y$, ($d_z$)
  - the desired trajectory is normalized to unit velocity in the $x$ axis and zero velocity in the $y$ axis.
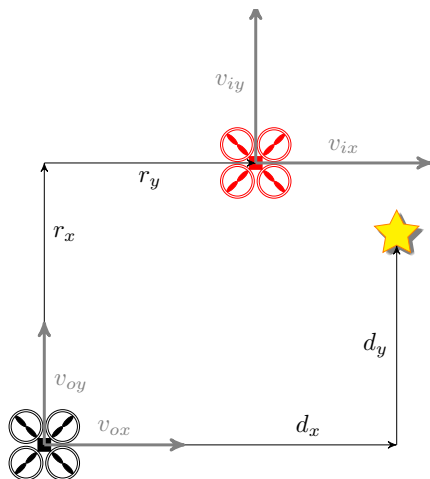
# MDP Transitions

- the prediction horizon is very short
- updates done every 0.1 to 1 seconds
- simple update equation are sufficient
- not a benefit to using more complex dynamic equations.
- $a_x$, $a_y$ acceleration by the ownship
- $N_*$ noise to the ownship, intruder, $x$ and $y$ axis
  - $N_o(\mu = 0, 0.30s^{-2})$, $N_i(\mu = 0, 0.45s^{-2})$,
- Bellman update

  transition from $s$ with acceleration $a$ to $s^|$

  $$Q[s, a] \leftarrow R(s, a) + \gamma \sum_{s^|} T(s^| | s, a) max_{a^|} Q[s^|, a^|].$$

$$
\begin{aligned}
\dot{r}_x &= v_{ix} - v_{ox} \\
\dot{r}_y &= v_{iy} - v_{oy} \\
\dot{v}_{ox} &= a_x + N_{ox} \\
\dot{v}_{oy} &= a_y + N_{oy} \\
\dot{v}_{ix} &= N_{ix} \\
\dot{v}_{iy} &= N_{iy} \\
\dot{d}_x &= v_{tx} - v_{ox} \\
\dot{d}_y &= v_{ty} - v_{oy}
\end{aligned}
$$

# Reward

- Minimum reward $R_{min}$
  - collision
  - physically impossible states
  - keeps the sum finite
- we prefer no acceleration
- we prefer long distance to the intruder
- we prefer short distance to the desired trajectory
- $K_s, K_T, R_{min}$ weights was learned, $k$ weights was $= 1$.

$$R(s, a) = max \left[ R_{min}, -(k_{ax}|a_x| + k_{ay}|a_y|) - K_s \frac{1}{k_{rx} r_x^2 + k_{ry} r_y^2} - K_T(k_{dx} d_x^2 + k_{dy} d_y^2) \right]$$

# QMDP Approximation

- offline optimization
  - a few hours for coarse discretization, 1 PC
  - initially stationary intruders
  - intruders moving at uniform velocity with a variety of relative headings angles
  - intruders state and dynamic uncertainty were added to the encounters.
- all values normalized
  - the coarse set contained a total of 765,625 discrete states
  - the finely discretized version contained 9,529,569 states.

| State variable | State Description | Discretization |
|---|---|---|
| $r_x$, $r_y$ | Intruder range components | $-15, [-7, -3], -1, 0, 1, [3, 7], 15$ |
| $v_{ox}$, $v_{oy}$ | ownship velocity components | $-5, -3, -1, 0, 1, 3, 5 \ s^{-1}$ |
| $v_{ix}$, $v_{iy}$ | intruder velocity components | $-5, [-3], -1, 0, 1, [3], 5 \ s^{-1}$ |
| $d_x$, $d_y$ | desired trajectory distance | $-10, [-3], -1, 0, 1, [3], 10$ |

# Evaluation Function

- The primary goal is to remain safely separated from the intruder aircraft.

  - $r_{5\%CPA}$ 'the closest point of approach', we allow 5% trajectories a little bit closer.

- Figure: required 1.5 units, never closer than 1.1 units.

- Mean deviation distance from the desired trajectory $\mu_{dev}$.
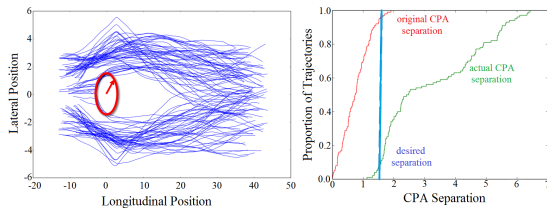


Figure 3: Separation metric used to evaluate the collision avoidance algorithm
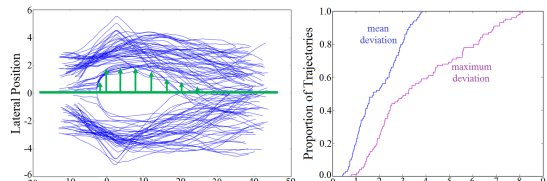
# Reward Tuning – Bayesian Optimization

- We tune $R_P = (K_T, K_s, R_{min})$
- $\beta$ weights the two objective functions

$$F(R_P) = (\beta \times (r_{5\%CPA})^{-1} + (1-\beta) \times \mu_{dev}).$$

- Gaussian process models $F(R_P)$.
- We determine the point at which the objective function is expected to have the largest improvement, $E[I(F(R_P))]$ over that of the current minimum.
- This set of $R_P$ is passed to QMDP to evaluate.
- until convergence.
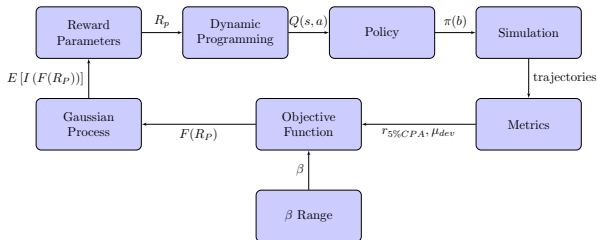


Figure 5: Process for tuning POMDP reward parameters

# Bayesian Optimization

- we know QMDP and F values for one or more $\mathbf{x} = R_P$ points
- we search the point $x = R_P^*$ to observe
- we minimize $y = F(R_P^*)$ and search the maximal probability of improvement
- 'the chance to improve' is expressed by the **Expected improvement** ($EI$)
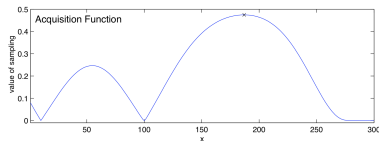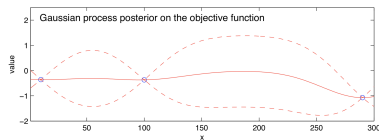


Peter I. Frazier: A Tutorial on Bayesian Optimization, rXiv:1807.02811v1 [stat.ML] 8 Jul 2018

# Value Iteration, QMDP Policies

- considerable improvement in convergence speed by initializing by the value of previously evaluated policy
- the value of maximum negative reward influenced the convergence speed
- $\gamma \leftarrow 0.99$ taking hundred iterations to converge.
- smaller $\gamma$ did not ensure the return to the desired path.

- Figures: Owhship at the origin
- different intruder positions
- policies indicated by color: black=up, red=right
- left: both own and intruder velocities are zero, $d = 0$
- right: owhship is moving in the positive $y$–axis direction at $1\ s^{-1}$ with zero trajectory error and nominal trajectory matches the velocity.
- The intruder is stationary.
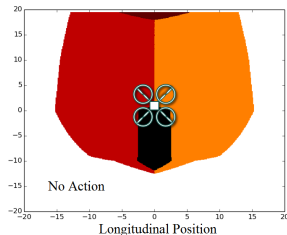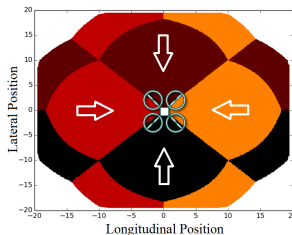
# Beliefs

- Uncertainty does not increase with time
- State uncertainty is incorporated only when actions are selected
- a set of potential states is calculated from the observations received at each step.
- the potential states become the beliefs used to select an action.

$$\pi(b) = max_a \left[ \sum_k Q(s^{(k)}, a) b^{(k)} \right]$$

- The value $Q(s^{(k)}, a) b^{(k)}$ approximated from QMDP solutions
  - rectangular interpolation between $2^n$ nearest neighbor
  - simplex interpolation between $n + 1$ nearest neighbor
  - prior work has found little benefit to using more sophisticated approaches.

# Pareto Optimal frontier

- 194 parameter sets evaluated
- $\beta$ between 0.01 and 0.99 .
- resulting in nine non-dominated, Pareto–optimal designs.
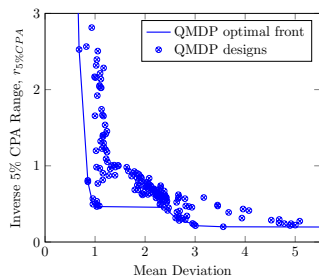
# Human Expert Check

- Left: intruder starts at $(0, 0)$,
- random heading, fixed velocity of the intruder
- the ownship starts at the blue cross

- Right: The goal is hovering
- the intruder comes from the right with the unknown behaviour.
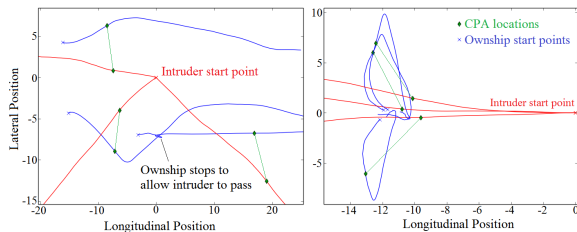
# State Discretization

The fine discretization improves the results.



(a) CPA separations

(b) Maximum trajectory deviations
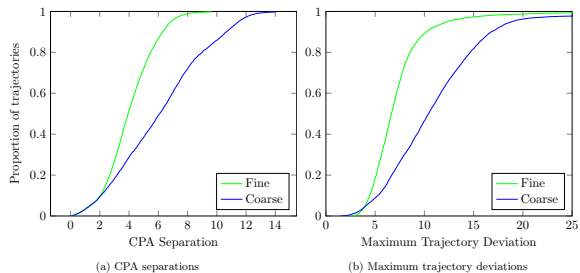
Figure 10: Cumulative distributions of encounter model metrics as a function of state discretization

# Further ML Methods

- Linear Projections
    - **Principal Component Analysis (PCA)**: 'the most spread variance' directions
        - **Sparse PCA**. (sklearn)
    - **Partial Least Squares**: not mentioned here. (sklearn)
    - **Archetypal analysis**: extremes, instead of 'centers' from clustering; data=lin. comb. of archetypes (archetypes)
    - **NMF Nonnegative Matrix Factorization**: 'linear $r$-dimensional autoencoder' (sklearn)
    - **Factor analysis**: A view on 'independent factors' observed via a linear combination mixture with a gaussian noise (sklearn)
    - **Independent Component Analysis**: splits the signal according to non-gaussian features (max. divergence from gaussian) (sklearn)
    - **Procrustes transformation** - curve fitting.
- **Principal curves and surfaces** (predefined $f_j(\lambda)$, curve paramater $\lambda$) (prinPy),
- **Kernel PCA**. (sklearn)
- **Spectral Clustering**. (sklearn)

# PCA Principal Components, Curves and Surfaces

- The **principal components** of a set of data in $\mathbb{R}^p$ provide a sequence of best linear approximations to that data, of all ranks $q \leq p$.

- let $\mu$ be a location vector in $\mathbb{R}^p$, $V_q$ is a $p \times q$ matrix with $q$ orthogonal unit vectors as columns, $\lambda$ is a $q$ vector of parameters.

- $f(\lambda) = \mu + V_q \lambda$ represents an affine hyperplane of rank $q$.

- We minimize the **reconstruction error** (by least squares)
  - $min_{\mu, \{\lambda_i\}} \sum_{i=1}^{N} \|x_i - \mu - V_q \lambda_i\|^2$.

- We can partially optimize
  - $\hat{\mu} = \overline{x}$
  - $\hat{\lambda}_i = V_q^T(x_i - \overline{x})$.

- This leaves us to find the **orthogonal matrix** $V_q$
  $min_{V_q} \sum_{i=1}^{N} \left\| (x_i - \overline{x}) - V_q V_q^T (x_i - \overline{x}) \right\|^2$.

- We center the data $\overline{x} = 0$ to simplify the formulas.
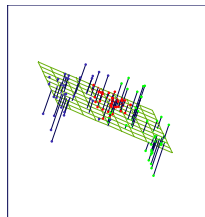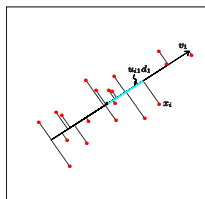
# Handwritten 3 Example

- 130 handwritten digits 3, each $16 \times 16$ grayscale image.
  - $x \in \mathbb{R}^{256}$
- First two principal component plot
  - For the first two principal components quantiles 5,25,50,75,95 percent.
  - First component - $x$ axis: mainly the length of 3
  - Second component - the thickness.
- The projection on the first two components is:

$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

$$= \boxed{3} + \lambda_1 \cdot \boxed{3} + \lambda_2 \cdot \boxed{3}.$$

- First 12 components account for 63% data variations.
- Explained variance by PCA (blue) and randomized directions (orange).
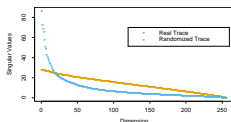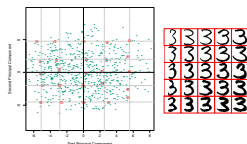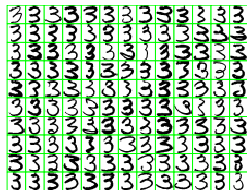
# Sparse Principal Components

- We often interpret PCA by examining **loadings**: direction vectors $v_j$.
- This interpretation is easier if the loadings are sparse.

> **Definition (Sparse PCA)**
>
> **Sparse principal component** technique solves for a single component:
>
> $$min_{\theta,v} \sum_{i=1}^{N} \|x_i - \theta v^T x_i\|_2^2 + \lambda \|v\|_2^2 + \lambda_1 \|v\|_1$$
>
> $$\text{subject to } \|\theta\|_2 = 1.$$

- If both $\lambda = \lambda_1 = 0$ and $N > p$, than $v = \theta$ is the largest principal component direction.

- When $p \gg N$ the solution may not be unique unless $\lambda > 0$. For $\lambda > 0$ and $\lambda_1 = 0$ is the solution proportional to the largest principal component direction.

**Sparse principal component** for multiple components minimizes $\Theta$ and $V$ $p \times K$ matrices

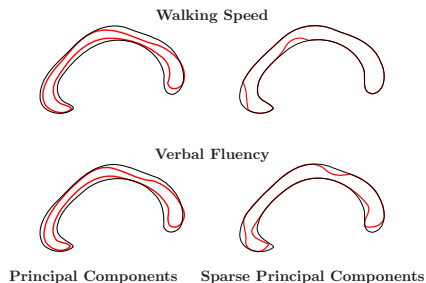$$min_{\Theta,V} \sum_{i=1}^{N} \|x_i - \Theta V^T x_i\|_2^2 + \lambda \sum_{k=1}^{K} \|v_k\|_2^2 + \sum_{k=1}^{K} \lambda_{1k} \|v_k\|_1$$

$$\text{subject to } \Theta^T \Theta = I_K.$$

# Corpus Callosum (CC) Sparse PCA Example
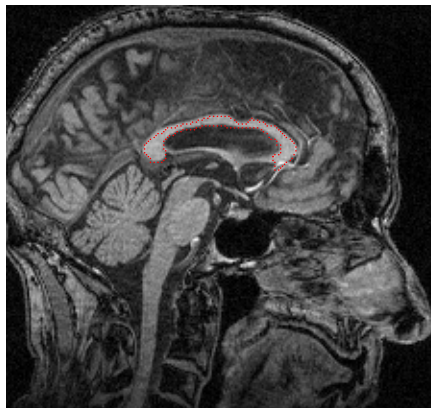


- The Corpus Callosum scan.
- The area represented by a number of points aligned by Procrustes analysis,
- a set of 2d points for now.



**Walking Speed**

**Verbal Fluency**

**Principal Components**     **Sparse Principal Components**

# Archetypal Analysis

- **Archetypal Analysis** approximates data points by a linear combination of prototypes
  - that are themselves linear combinations of data points.
  - Each data point is approximated by a convex combination of prototypes.
  - This forces the prototypes to lie on the convex hull of the data cloud.
  - In this sense, they are 'archetypal'.

- K-means clustering
  - approximates any point by one prototype
  - each prototype is a linear combination of samples (the mean of a cluster).
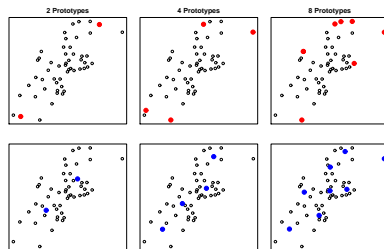
# Archetypal Analysis

- "linear autoencoder" of the dimension $r$

> **Definition (Archetypal Analysis)**
>
> - A non-negative $N \times p$ data matrix $X$ is modeled $X \sim WH$,
>   - $H = BX$ is $r \times p$ matrix of $r$ **archetypes** (rows of $H$),
>   - $B$ is $r \times N$ matrix where $b_{ki} \geq 0$ and $(\forall k)$ $(\sum_{i=1}^{N} b_{ki} = 1)$.
>   - $W$ is $N \times r$ matrix where $w_{ik} \geq 0$ and $(\forall i)$ $(\sum_{k=1}^{r} w_{ik} = 1)$.
>   - We minimize over $W$ and $B$: $J(W, B) = \|X - WH\|^2 = \|X - WBX\|^2$.

- Its minimized in an alternating fashion, with each separate minimization involving a convex optimization.

- Converges to a local minimum.

- Figure: 2,3, and 4 prototypes for the Handwritten 3 example.

- Extreme 3's both in size and shape.
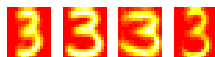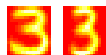
# Non-negative Matrix Factorization



Definition (Non-negative Matrix Factorization)

- A centered $N \times p$ data matrix $X$ is modeled $X \sim WH$,
- $W$ is $N \times r$ matrix, $H$ is $r \times p$, $r \leq \max(N, p)$.
- We assume $x_{ij}, w_{ik}, h_{kj} \geq 0$.
- We maximize over $W$ and $H$: $L(W, H) = \sum_{i=1}^{N} \sum_{j=1}^{p} [x_{ij} \log(WH)_{ij} - (WH)_{ij}]$.

- NMF assumes $x_{ij}$ has a Poisson distribution with mean $(WH)_{ij}$
- we maximize the loglikelihood.

# NMF
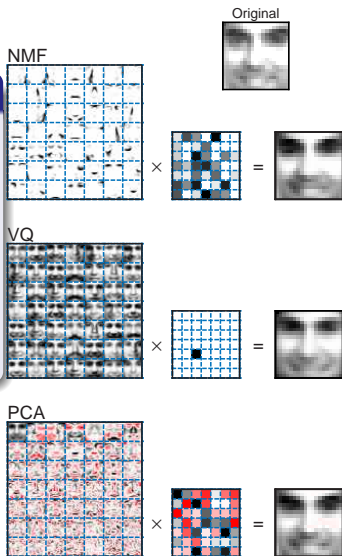
## NMF

1: **procedure** $\mathrm{NMF}$:($X$ centered data)
2:     **repeat**
3:         $w_{ik} \leftarrow w_{ik} \dfrac{\sum_{j=1}^{p} h_{kj} x_{ij}/(WH)_{ij}}{\sum_{j=1}^{p} h_{kj}}$,
4:         $h_{kj} \leftarrow h_{kj} \dfrac{\sum_{i=1}^{N} w_{ik} x_{ij}/(WH)_{ij}}{\sum_{i=1}^{N} b_{ik}}$
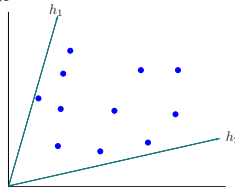5:     **until** convergence
6:     return $W$, $H$
7: **end procedure**

- The NMF solution is not unique.

Any $h_1$, $h_2$ basis vectors in the open space between the coordinate axes and data work (given an exact reconstruction of the data).



sklearn.decomposition.NMF has the objective function:

$$0.5\|X - WH\|_{loss}^2 \quad + \quad \alpha_W \cdot l1_{ratio} p\|vec(W)\|_1 + 0.5\alpha_W \cdot (1 - l1_{ratio})p\|W\|_{Fro}^2$$
$$+ \quad \alpha_H \cdot l1_{ratio} N\|vec(H)\|_1 + 0.5\alpha_H \cdot (1 - l1_{ratio})N\|H\|_{Fro}^2$$

- $\|vec(W)\|_1 = \sum_{i,j} abs(W_{i,j})$ elementwise L1 norm
- $\|W\|_{Fro}^2 = \sum_{i,j} W_{i,j}^2$ Frobenius norm
- $loss$ is Frobenius norm or another beta-divergence loss, $l1_{ratio} = 0$.

# Independent Component Analysis

- Multivariate data as multiple indirect measurements from an underlying source.
- Examples: EEG brain scans, 'body fat', trading prices.
- **Factor analysis**
  - typically wed to Gaussian distributions
  - which has hindered their usefulness
  - and has no unique solution
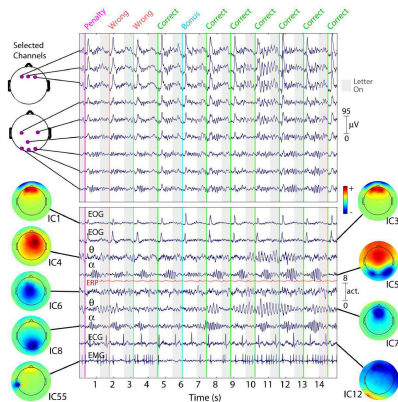    - any linear transformation is a solution.



**FIGURE 14.41.** *Fifteen seconds of EEG data (of 1917 seconds) at nine (of 100) scalp channels (top panel), as well as nine ICA components (lower panel). While nearby electrodes record nearly identical mixtures of brain and non-brain activity, ICA components are temporally distinct. The colored scalps represent the ICA unmixing coefficients $\hat{a}_j$ as a heatmap, showing brain or scalp location of the source.*

# Latent Variables and Factor Analysis

- Take the singular value decomposition $X = UDV^T$
  - we assume that the columns of $X$ have zero mean
  - where $D$ is a diagonal matrix
  - $U$ is orthogonal.
- $X$ has a **latent variable decomposition** $X = SA^T$
  - where $S = \sqrt{N}U$, $A^T = \frac{1}{\sqrt{N}}DV^T$
  - each of the columns of $X$ is a linear combination of the columns of $S$
  - columns of $S$ have zero mean, are uncorrelated and have unit variance, $Cov(S) = I$.
  - we can interpret the SVD, or the corresponding PCA as an estimate of a latent variable model $X = AS$

$$
\begin{aligned}
X_1 &= a_{11}S_1 + a_{12}S_2 + \ldots + a_{1p}S_p \\
X_2 &= a_{21}S_1 + a_{22}S_2 + \ldots + a_{2p}S_p \\
&\ldots \\
X_p &= a_{p1}S_1 + a_{p2}S_2 + \ldots + a_{pp}S_p
\end{aligned}
$$

- Notice that for any orthogonal $p \times p$ matrix $R$ is $X = AS = AR^TRS = A^*S^*$.

# Factor Analysis

- In the SVD decomposition any rank $q < p$ truncated decomposition approximates $X$ in an optimal way.
- **Factor analysis model** (popular in psychometrics)
  - with $q < p$, a factor analysis model has the form $X = AS + \epsilon$

$$
\begin{aligned}
X_1 &= a_{11}S_1 + a_{12}S_2 + \ldots + a_{1q}S_q + \epsilon_1 \\
X_2 &= a_{21}S_1 + a_{22}S_2 + \ldots + a_{2q}S_q + \epsilon_2 \\
&\ldots \\
X_p &= a_{p1}S_1 + a_{p2}S_2 + \ldots + a_{pq}S_q + \epsilon_p
\end{aligned}
$$

  - $S$ is a vector of $q < p$ underlying latent variables or factors
  - $A$ is a $p \times q$ matrix of factor **loadings**
    - used to name and interpret the factors
  - $\epsilon_j$ are uncorrelated zero–mean disturbances.
  - Typically, $S_\ell$ and $\epsilon_j$ are modeled as Gaussian random variables, and the model is fit by maximum likelihood.
  - The parameters all reside in the covariance matrix

$$
\Sigma = AA^T + D_\epsilon
$$

  - where $D_\epsilon = diag[Var(\epsilon_1), Var(\epsilon_2), \ldots, Var(\epsilon_p)]$
  - $S$ independent factors like intelligence, drive in a battery of educational tests.

# Independent Component Analysis

$$
\begin{aligned}
X_1 &= a_{11}S_1 + a_{12}S_2 + \ldots + a_{1p}S_p \\
X_2 &= a_{21}S_1 + a_{22}S_2 + \ldots + a_{2p}S_p \\
&\ldots \\
X_p &= a_{p1}S_1 + a_{p2}S_2 + \ldots + a_{pp}S_p
\end{aligned}
$$

- $S$ are assumed **statistically independent** rather than uncorrelated
  - correlation: second order interaction
  - independence: all orders of interactions.
  - Multivariate Gaussian is determined by its second moments alone (up to rotation).
  - Otherwise, the extra moments allow to identify the elements of $A$ uniquely.
- We assume $X$ has been **whitened** to have $Cov(X) = I$; Simplest: multiply by $W = \Sigma^{-\frac{1}{2}}$, typically achieved via the SVD to $D^{-\frac{1}{2}}V^T$.
  - $Var(S) = I$, therefore is $A$ orthogonal.
  - ICA searches an orthogonal $S$ such that $S = A^T X$ are independent (not-Gaussian) components.

# Example

- **Cocktail party problem** Different microphones $X_j$ pick up mixtures of different independent sources $S_\ell$ (music, speech from different speakers).
- ICA is able to perform blind source separation
  - by exploiting the independence and non–Gaussianity of the original sources.
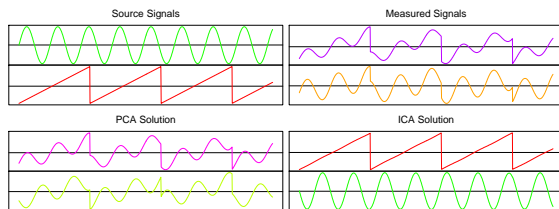
# Entropy

$$\text{entropy } H(Y) \;=\; -\int g(y) \log g(y) dy$$

$$\text{mutual information } I(Y) \;=\; \sum_{j=1}^{p} H(Y_j) - H(Y)$$

$$I(Y) \;=\; \sum_{j=1}^{p} H(Y_j) - H(X) - \log|\det(A)|$$

$$=\; \sum_{j=1}^{p} H(Y_j) - H(X)$$

- since $Cov(X) = I$, $Y = A^T X$ and $A$ is orthogonal.
- We search $A$ to minimize $I(Y) = I(A^T X)$
  - looks for the orthogonal transformation that leads to the most independence between its components
  - minimizes the sum of the entropies of the separate components of $Y$
  - this amounts to maximizing their departures from Gaussianity.

# Negentropy, FastICA

- For each $Y_j$, let $Z_j$ be a Gaussian random variable with the same variance as $Y_j$.

- The **negentropy** $J(Y_j)$ is defined

$$J(Y_j) = H(Z_j) - H(Y_j)$$

- It is non–negative, and measures the departure of $Y_j$ from Gaussianity.

- Can be approximated by

$$J(Y_j) \sim [\mathbb{E}G(Y_j) - \mathbb{E}G(Z_j)]^2$$

- $G(u) = \frac{1}{a}\log\cosh(au)$ for $1 \leq a \leq 2$.



## FastICA

- ICA starts from essentially a factor analysis solution

- and looks for rotations that lead to independent components.
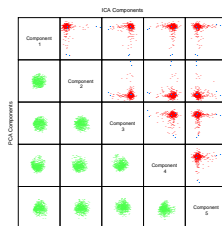
- Above diagonal: first five ICA components

- Below diagonal: first five PCA components

- all standardized to unit variance.

# FastICA

- $X$ are centered and whitened data with $Cov(X) = I$
  - typically achieved via the SVD, $X \leftarrow \sqrt{D}U$ from $X = UDV^T$.
- $q$ the number of components
  - Only one is allowed to follow Gaussian distribution.
- $a$ a parameter.

---

### FastICA

1: **procedure** $\textsc{FastICA}$:($\mathbf{X}$, $a \in \langle 1, 2 \rangle$, $q \leq p$ )
2:      $w_1, \ldots, w_q \leftarrow$ randomly initialize $N$-dimensional weight vectors
3:      **for** $\ell = 1, \ldots, q$ **do**
4:          **repeat**
5:             $w_\ell^+ \leftarrow \sum_{i=1}^{N} x \tanh(a w_\ell^T x) - \left( \sum_{i=1}^{N} \frac{1}{cosh^2(w_\ell^T x)} \right) w$
6:             $w_\ell \leftarrow w_\ell^+ - \sum_{j=1}^{\ell-1} w_\ell^T w_j w_j$ # orthogonal to previous
7:             $w_\ell \leftarrow \frac{w_\ell}{\sqrt{w_\ell^T w_\ell}}$ # normalize
8:          **until** convergence
9:      **end for**
10: **end procedure**

# Procrustes Transformations

- Assume each handwritten S is represented as $N = 96$ points.

- Both $X_1$ and $X_2$ are $N \times 2$ matrices (green, orange curve).

  - with column means $\overline{x}_1, \overline{x}_2$, centered to $\tilde{X}_1, \tilde{X}_2$.

  - To find landmarks (points) are difficult and subject specific.

  - In this example, dynamic time wrapping of the speed signal along each signature was used.
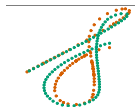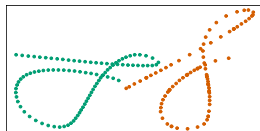




- **Procrustes** transformation

  - $R$ is an orthonormal $p \times p$ matrix,

    - $\hat{R} \leftarrow UV^T$ from $\tilde{X}_1^T \tilde{X}_2 = UDV^T$.

  - $\mu$ a $p$ vector of location coordinates

    - $\mu \leftarrow \overline{x}_2 - \hat{R}\overline{x}_1$.

  - $\|X\|_F^2 = trace(X^T X) = \sum_{i=1}^{N} \sum_{j=1}^{p} |x_{ij}|^2$ is the squared **Frobenius** matrix norm

  - We minimize the **Procrustes distance**

$$min_{\mu, R} \left\| X_2 - (X_1 R + 1\mu^T) \right\|_F^2.$$

# Shape Averaging, Procrustes Average

! From now on, we assume the data are centered.

**Definition (Procrustes average)**

The **Procrustes average** of a collection of $L$ shapes is $M$ that minimizes

$$min_{\{R_\ell\}_1^L, M} \sum_{\ell=1}^{L} \|X_\ell R_\ell - M\|_F^2 .$$
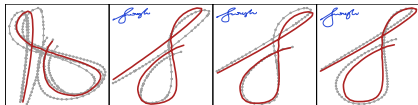


**Procrustes Average**

```
1: procedure PROCRUSTES AVERAGE:(N × p shapes {X_ℓ}_{ℓ=1}^L )
2:     M ← X_1 # init the average
3:     repeat
4:         X'_ℓ ← X_ℓ R̂_ℓ # M fixed, solve L Procruster rotations R̂_ℓ
5:         M ← (1/L) Σ_{ℓ=1}^L X'_ℓ # average
6:     until convergence
7: end procedure
```

# Principal Curves and Surfaces

- To find a principal curve $f(\lambda)$ of a distribution, we consider

- $f(\lambda) = [f_1(\lambda), f_2(\lambda), \ldots, f_p(\lambda)]$ its coordinate functions and let

- $X^T = (X_1, \ldots, X_p)$



$f(\lambda) = [f_1(\lambda), \ f_2(\lambda)]$

---

### Principal Curves and Surfaces

1: **procedure** PRINCIPAL CURVE:$(f(\lambda), X\ )$
2:    **repeat**
3:        $\hat{f}_j(\lambda) \leftarrow \mathbb{E}[X_j | \lambda(X) = \lambda], j = 1, \ldots, p,$
4:        $\hat{\lambda}_f(x) \leftarrow \arg\min_{\lambda'} \left\| x - \hat{f}(\lambda') \right\|^2.$
5:    **until** convergence
6: **end procedure**

---



- A scatterplot smoother is used to estimate the conditional expectations in step 3: by smoothing each $X_j$ as a function of the arc-length $\lambda(\hat{X})$.
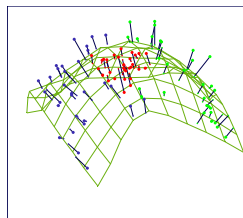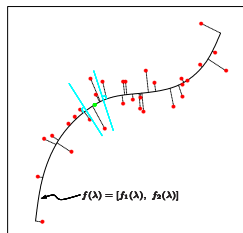
# Kernel Principal Components

- We can select any kernel function, like radial $K(x_i, x_{i'}) = e^{-\frac{\|x - x'\|^2}{c}}$.

- We set $M = 11^T/N$ and calculate double-centered version of $K$

$$\tilde{K} = (I - M)K(I - M) = UD^2U^T$$

- then principal components variables are $Z = UD$.

  - The elements of the $m$th component $z_m$ ($m$th column of Z) can be written (up to centering)

  - $z_{im} = \sum_{j=1}^{N} \alpha_{jm}K(x_i, x_j)$, where $\alpha_{jm} = \frac{u_{jm}}{d_m}$.

- Figure: Radial kernel (top) and spectral clustering without NN (bottom right) on the previous 3-'circles' example.
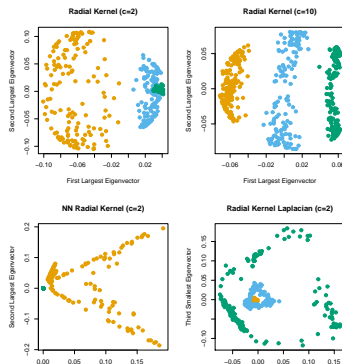
# Spectral Clustering

- The idea is to put close points into the same cluster.
- We form a weighted adjacency graph for data samples.



## Spectral Clustering

1: **procedure** $\mathrm{SC}{:}(X$ as $N$ points in $\mathbb{R}^p$, $c > 0$ scale, $k > 0$ )
2: $\quad s_{ii'} \leftarrow exp(-d(i,i')^2/c)$ # calculate the similarity matrix
3: $\quad W, G \leftarrow$ zero matrix $N \times N$
4: $\quad$ **for** $i, i'$ symmetric nearest neighbors **do**
5: $\quad\quad w_{ii'} \leftarrow s_{ii'}$ # connect them
6: $\quad$ **end for**
7: $\quad$ **for** $i \in X$ **do**
8: $\quad\quad g_{ii} \leftarrow \sum_{i'} w_{ii'}$ # the degree of vertex $i$
9: $\quad$ **end for**
10: $\quad L \leftarrow G - W$ # the graph Laplacian (unnormalized)
11: $\quad$ (or $\tilde{L} \leftarrow I - G^{-1}W$ # (normalized))
12: $\quad$ find $m$ eigenvectors $Z_{N \times m}$ with **smallest** eigenvalues of $L$
13: $\quad$ **return** $Z_{N \times m}$ rows clustered by standard $k - means$
14: **end procedure**

# Spectral Clustering

- For any vector $f$

$$f^T L f = \sum_{i=1}^{N} g_{ii} f_i^2 - \sum_{i=1}^{N} \sum_{i'=1}^{N} f_i f_{i'} w_{ii'}$$

$$= \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} w_{ii'} (f_i - f_{i'})^2.$$



- $1^T L 1 = 0$ for any graph.
- For a graph with $m$ connected components,
  - reordered so that $L$ is a block diagonal with a block for each component
- then $L$ has $m$ eigenvectors of eigenvalue zero.
- In practice zero eigenvalues are approximated by small eigenvalues.
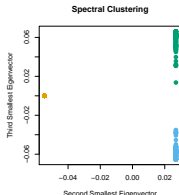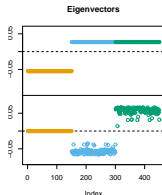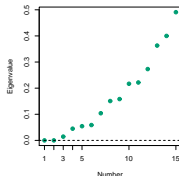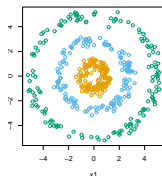
# List of topics

1. Linear, ridge, lasso regression, k-neares neighbours,(formulas) overfitting, curse of dimensionality, (LARS)
2. Splines - the base, natural splines, smoothing splines; kernel smoothing: kernel average, Epanechnikov kernel.
3. Logistic regression, Linear discriminant analysis, generalized additive models
4. Train/test error and data split, square error, 0-1, crossentropy, AIC, BIC,(formulas) crossvalidation, one-leave-out CV, wrong estimate example
5. decision trees, information gain/entropy/gini, CART prunning,(formulas)
6. random forest (+bagging), OOB error, Variable importance, boosting (Adaboost(formulas) and gradient boosting), stacking, MARS ,
7. Bayesian learning: MAP, ML hypothesis (formulas), Bayesian optimal prediction, EM algorithm
8. Clustering: k-means, Silhouette, k-medoids, hierarchical
9. Apriori algorithm, Association rules, support, confidence, lift
10. Inductive logic programming basic: hypothesis space search, background knowledge, necessity, sufficiency and consistency of a hypothesis, Aleph
11. Undirected graphical models, Graphical Lasso procedure, deviance , MRF
12. Gaussian processes: estimation of the function and its variance (figures, ideas).

# Table of Contens