Machine Learning

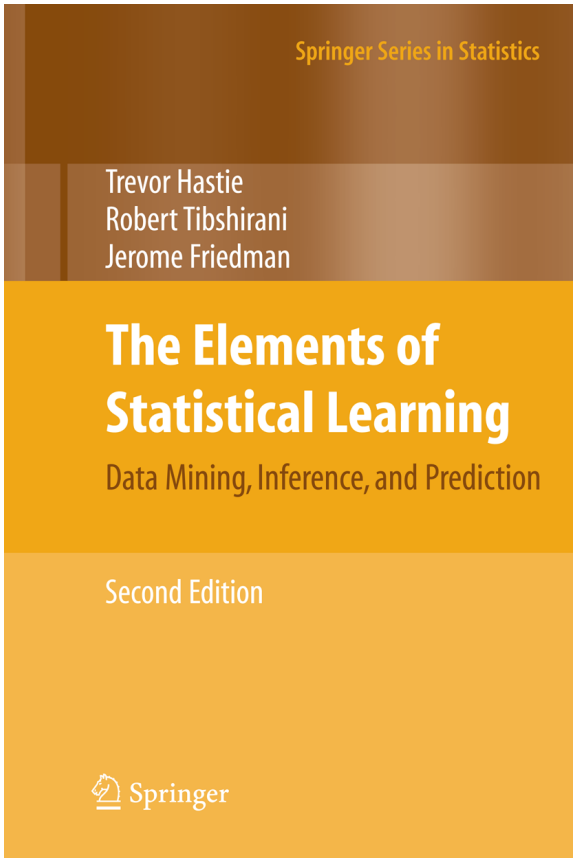
moodle https://dl1.cuni.cz/course/view.php?id=5765

Marta Vomlelová

marta@ktiml.mff.cuni.cz
http://ktiml.mff.cuni.cz/~marta

February 23, 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

1. Overview of Supervised Learning
2. Kernel Methods, Basis Expansion and regularization
3. Linear methods for classification
4. Model Assessment and Selection
5. Additive Models, Trees, and Related Methods
6. Ensamble Methods
7. Clustering
8. Bayesian learning, EM algorithm
9. Association Rules, Apriori
10. Inductive Logic Programming
11. Undirected (Pairwise Continuous) Graphical Models
12. Gaussian Processes
13. PCA Extensions, Independent CA
14. Support Vector Machines

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:

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

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

k-NN and Conditional Expectation

- We seek the conditional expectation:

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

- 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

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

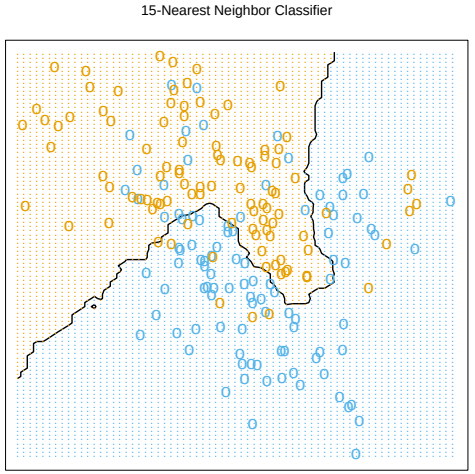
  - where *mean* denotes average, and $N_k(x)$ is the neighborhood containing k points in $\mathcal{T}$ closest to x.
- Under mild regularity conditions on $P(X, Y)$ one can show as $k, N \to \infty$, such that $\frac{k}{N} \to 0$ then $\hat{f}(x) \to \mathbb{E}(Y|X = x)$.
- **The rate of convergence decreases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

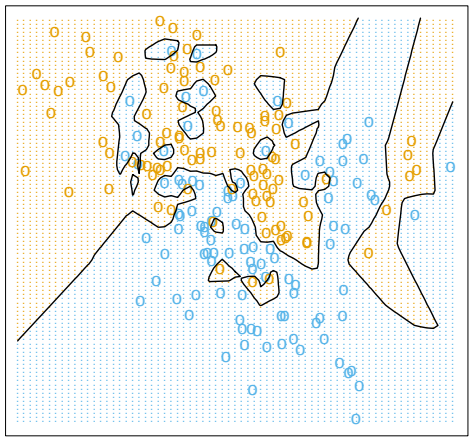
Nearest-Neighbor Methods

- 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

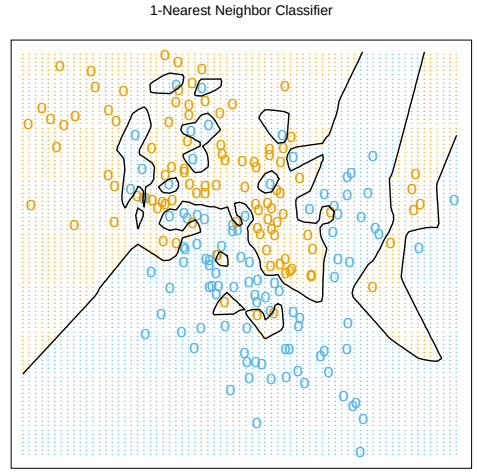
The effective number of parameters of k nearest neighbors is N/k and is generally bigger than p of the linear regression.

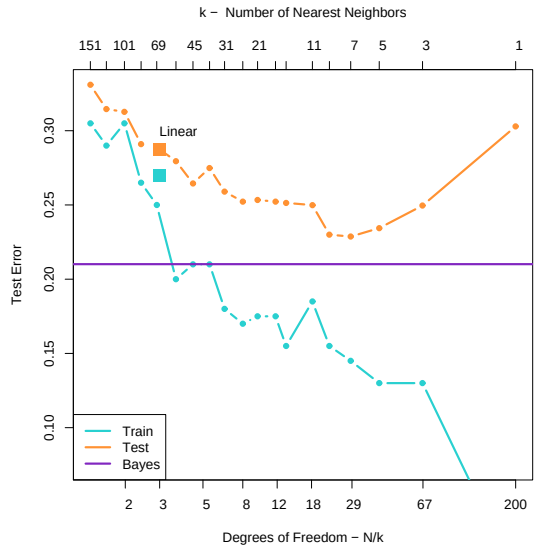
Prediction complexity

'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.

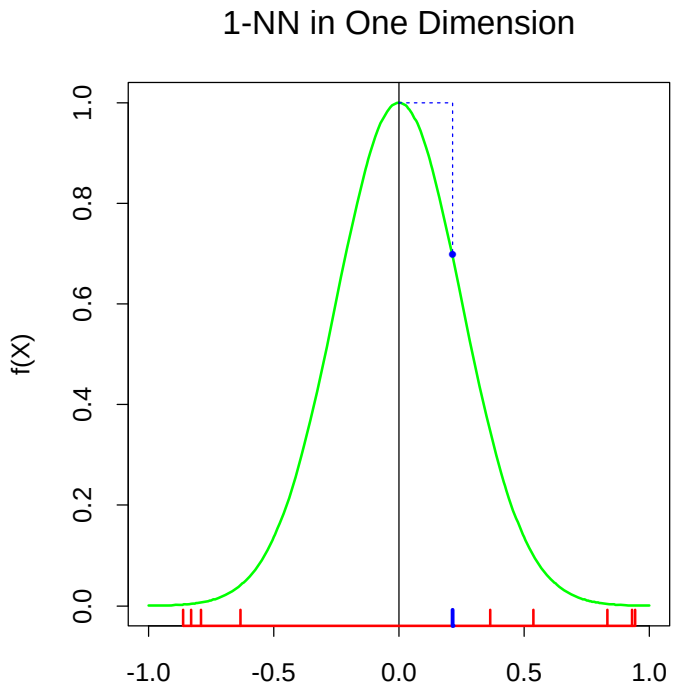


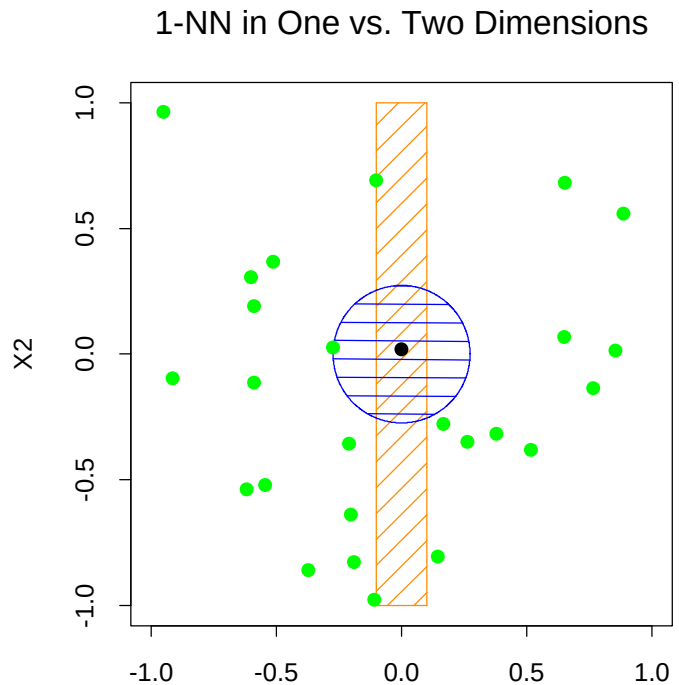


Curse of Dimensionality demonstration

Prediction

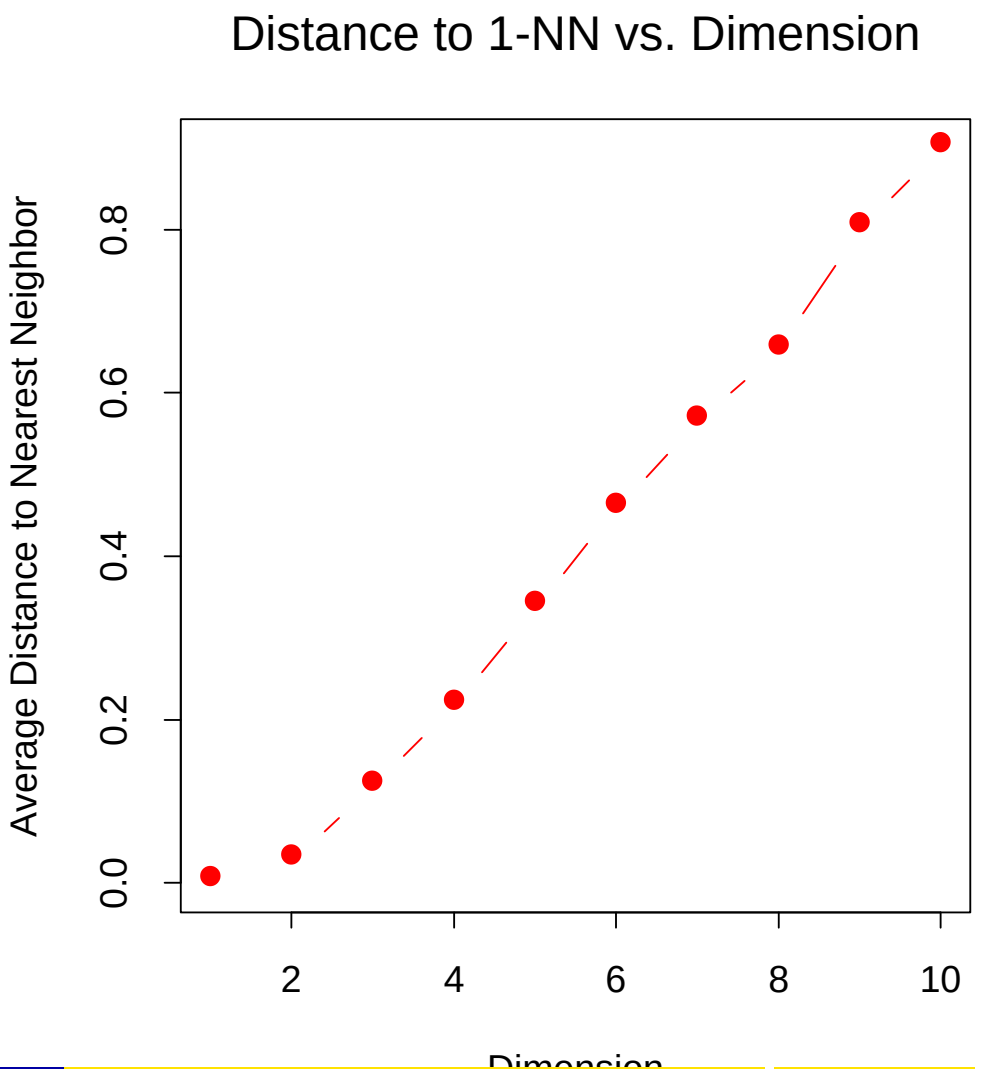
- Assume x_i uniformly generated form the interval $\langle -1, 1 \rangle^p$
- We have $Y = f(X) = e^{-8\|X\|^2}$, without any noise, for x_i we know exactly $f(x_i)$.
- We use 1–NN to estimate $f(0)$ based on 1000 data sample.
- Predicted value for $x = \langle 0, \ldots, 0 \rangle$ is lower that 1 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.



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	$\langle X_1 \quad X_j \quad X_p \rangle$	Y or G
$\mathbf{x}_1^T$		
$\mathbf{x}_i^T$ = vector	$\langle x_1 \quad x_j \quad x_p \rangle$	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_β, $\beta \in \mathbb{R}^{p+1}$

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

- $\hat{\beta}_0$ 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:

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

- Differentiating w.r.t. β we get *normal equations*

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

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

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

- 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 $\mathbf{X}^{\mathbf{T}}\mathbf{X}$ 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 β_0, does not count:

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

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

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

Linear regression from the data

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

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

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

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

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

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

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

Linear regression from the data

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

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

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

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

Prediction

- Linear regression predicts:

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

- Prediction for training data:

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

- **The hat matrix** $H = \mathbf{X}(\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}$ 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

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

Linear Regression Complexity

Training complexity

- The complexity of the direct approach to linear regression is $O(p^2N + p^3)$.
- $\mathbf{X}^\mathbf{T}\mathbf{X}$ is $O(p^2N)$
- 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

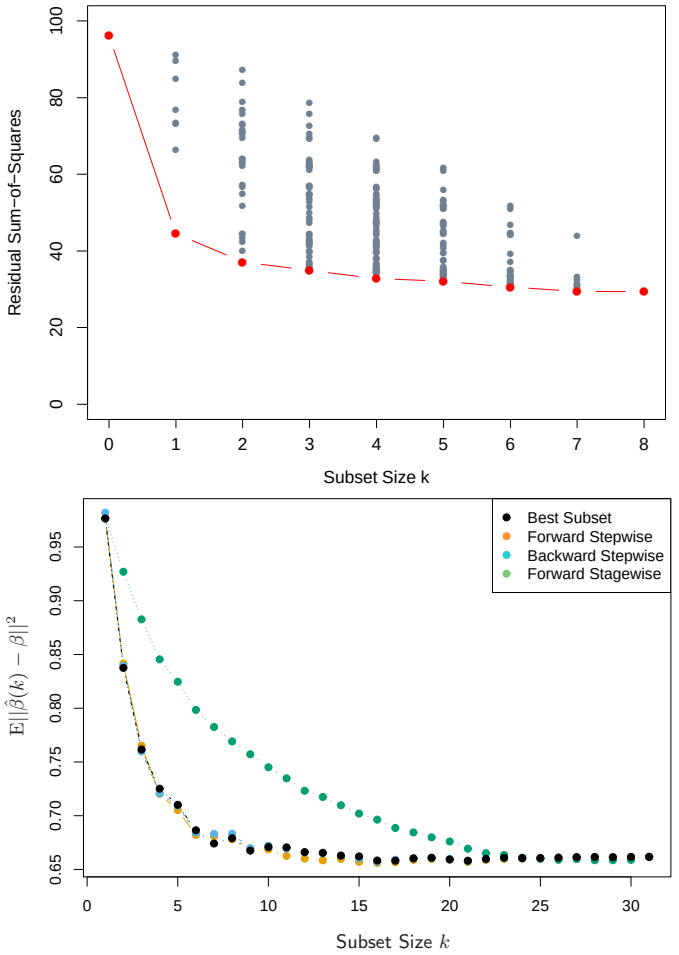
- $O(p^2N)$

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,

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

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

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

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

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

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

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

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

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

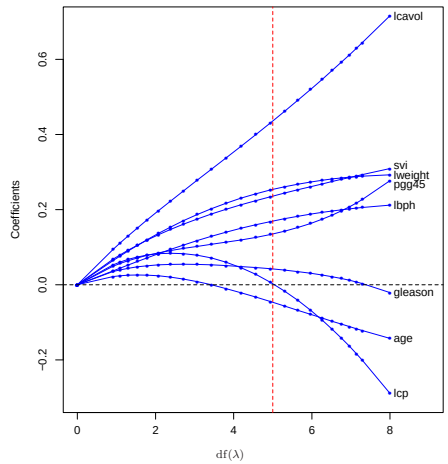
$$\rho_{j,k} \leftarrow corr(\mathbf{x}_j, \mathbf{x}_k) \leftarrow \frac{s_{j,k}}{s_j s_k} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \overline{x}_j)(x_{ik} - \overline{x}_k)}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \overline{x}_j)^2}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{ik} - \overline{x}_k)^2}}$$

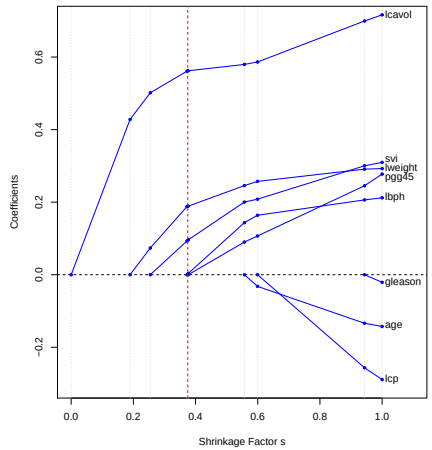
- For standardized features, the correlation is just $\frac{\mathbf{x}_j^T \mathbf{x}_k}{N}$.

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()
```

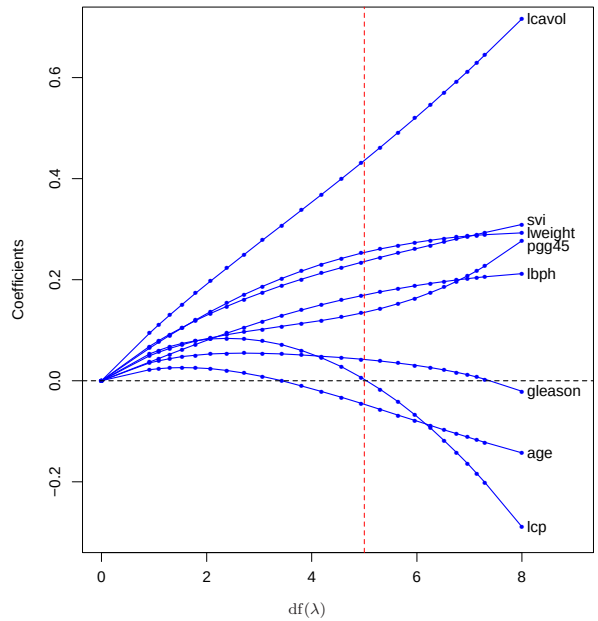
Rigde Regression

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

- The solution is

$$\begin{aligned} \mathbf{X} &\leftarrow \text{centered } (N \times p) \text{ input matrix} \\ \hat{\beta}_0 &= \frac{1}{N} \sum_{i=1}^{N} y_i \\ \hat{\beta} &= (\mathbf{X^T X} - \lambda \mathbf{I})^{-1} \mathbf{X^T y}. \end{aligned}$$

sklearn.linear_model.Ridge



Lasso Regression

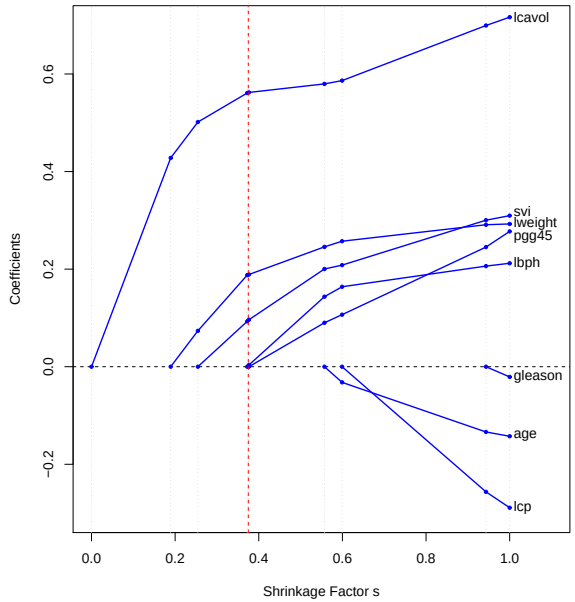
$$\hat{\beta}^{lasso} = argmin_{\beta} \left(\sum_{i=1}^{N} (y_i - \beta_0 - \mathbf{x}_i^T \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right)$$

- Solved by a quadratic programming algorithm

  sklearn.linear_model.Lasso

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

  sklearn.linear_model.Lars

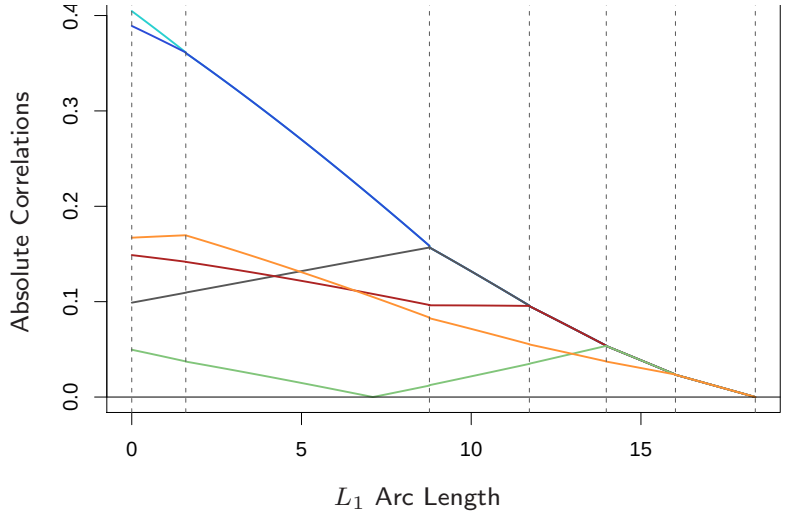


LARS Idea

- 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.

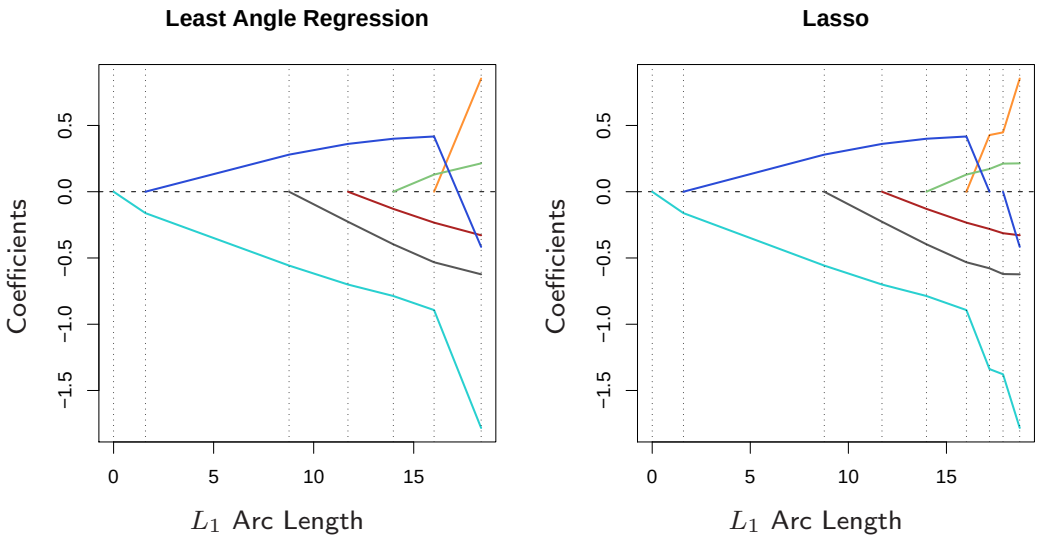


LARS Least Angle Regression

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

Least Angle Regression

procedure LEAST ANGLE REGRESSION:(**X**, **y**)
    $\beta_1, \ldots, \beta_p \leftarrow 0$ initialize
    $r \leftarrow y - \bar{y}$ residuals
    find the predictor x_j most correlated with r
    Move β_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.
    $\mathcal{A}_1 \leftarrow \{x_j\}$ active coefficients
    for $k = 2, \ldots, min(N-1, p)$ **do**
        move current set of $\beta_{\mathcal{A}_k}$ by their joint least squares coefficient of the current residual until some other competitor x_ℓ catches up.
        $\mathcal{A}_k \leftarrow \mathcal{A}_{k-1} \cup \{x_\ell\}$ active coefficients
    end for
end procedure



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 $O(p^2N + p^3)$ or Cholevsky decomposition.

Model Complexity

Effective Degrees of Freedom

- Linear regression p
- Ridge regression $df(\hat{y}) = tr(\mathbf{X}(\mathbf{X}^\mathbf{T}\mathbf{X} - \lambda\mathbf{I})^{-1}\mathbf{X}^\mathbf{T})$.
- 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 λ
- 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 β_k at penalty parameter λ

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

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

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

Pathwise Coordinate Optimization

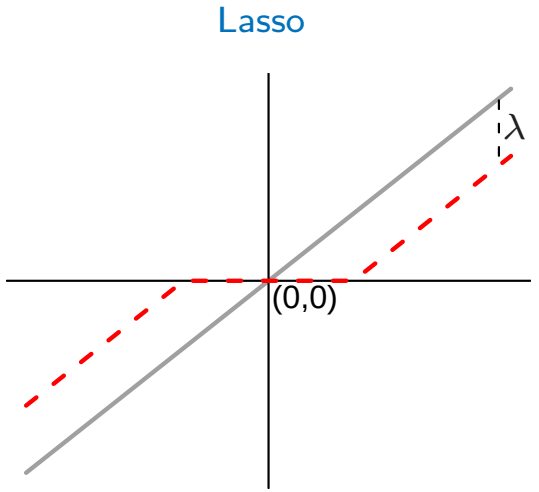
- this has an explicit solution, resulting in the update

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

- where S is the soft-thresholding operator

$$S(t, \lambda) = sign(t)(|t| - \lambda)_+ \tag{3}$$

- Estimators of β_j in case of orthonormal columns of X.



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_ℓ the number in group ℓ.
  - a matrix X_ℓ represents the predictors corresponding to the ℓth group
  - with corresponding coefficient vector β_ℓ.
- the grouped-lasso minimizes the convex criterion

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

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

Example – Storch brings babies in Europa

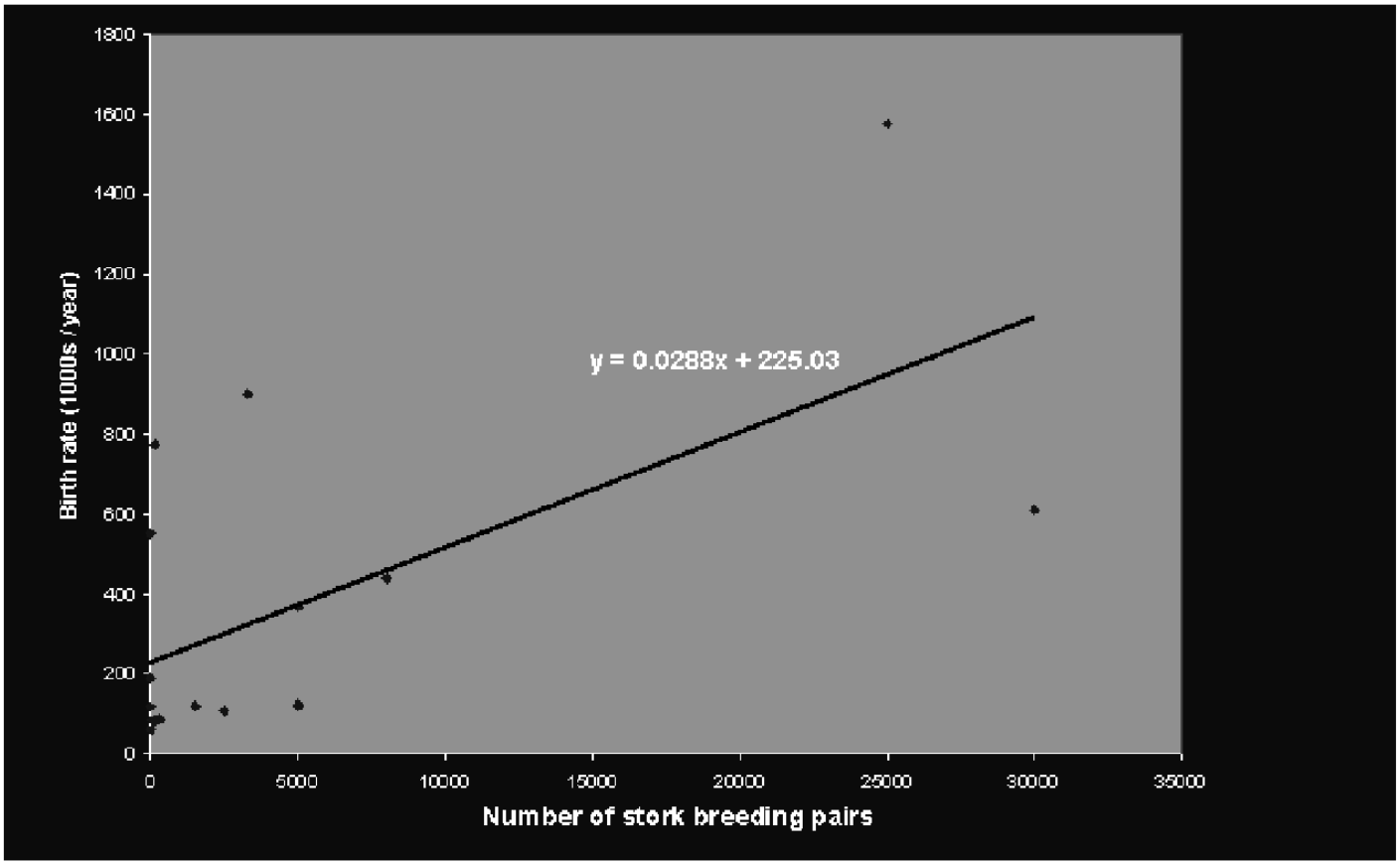


Fig 1. How the number of human births varies with stork populations in 17 European countries.

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:
$$\hat{G} = \left\{ \begin{array}{l} \textit{BLUE} \text{ for } Y \leq 0.5 \\ \textit{ORANGE} \text{ for } Y > 0.5 \end{array} \right\}$$
- 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.

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.

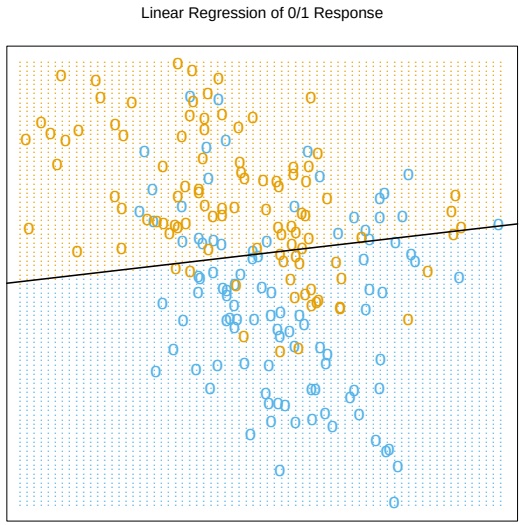


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$*. The orange shaded region denotes that part of input space classified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

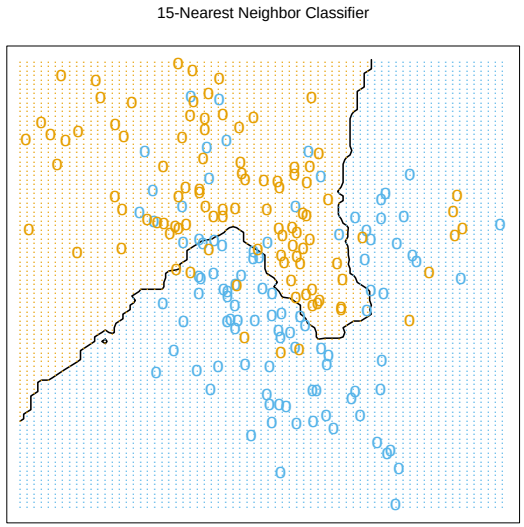


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*) and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

Table of Contens

1. Overview of Supervised Learning
2. Kernel Methods, Basis Expansion and regularization
3. Linear methods for classification
4. Model Assessment and Selection
5. Additive Models, Trees, and Related Methods
6. Ensamble Methods
7. Clustering
8. Bayesian learning, EM algorithm
9. Association Rules, Apriori
10. Inductive Logic Programming
11. Undirected (Pairwise Continuous) Graphical Models
12. Gaussian Processes
13. PCA Extensions, Independent CA
14. Support Vector Machines