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. diagonal discriminant analysis
- Nearest-neighbor methods
  - **k-NN**
  - **curse of dimensionality**
  - ext. Local likelihood (local logistic regression)
  - ext. Discriminating Adaptive NN methods (DANN)
  - ext. **GAM generalized additive models**
- ? 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 θ be the probability of *red*. We have following probabilities of data for different θ.

θ	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:

θ	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 θ given the data.
- Maximum likelihood θ 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:

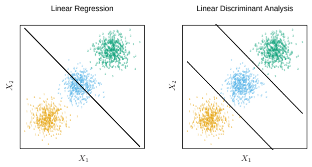
$$\ell(\theta) = \sum_{i=1}^{N} log_e(P(G = g_i|x_i, \theta))$$

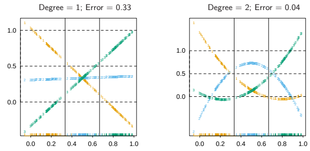
- Logistic regression uses:

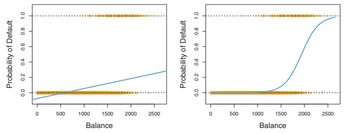
$$P(G = g_k|X = x) = \frac{e^{\beta_{k0} + \beta_k^T x}}{1 + \sum_{l=1,\dots,K-1} e^{\beta_{l0} + \beta_l^T x}}.$$

Logistic Function

- We could encode the class numeraically and fit a linear regression.
- 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 solves it naturally.
- Probability should be from $\langle 0, 1 \rangle$.
- 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 $\theta = \{\beta_{10}, \beta_1^T, \ldots, \beta_{(K-1)0}, \beta_{K-1}^T\}$.

$$\begin{aligned}
\log \frac{P(G=g_1|X=x)}{P(G=g_K|X=x)} &= \beta_{10} + \beta_1^T x \\
\log \frac{P(G=g_2|X=x)}{P(G=g_K|X=x)} &= \beta_{20} + \beta_2^T x \\
&\vdots \\
\log \frac{P(G=g_{K-1}|X=x)}{P(G=g_K|X=x)} &= \beta_{(K-1)0} + \beta_{K-1}^T x
\end{aligned}$$

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 β.
- First component: $x_i \equiv 1$ specifies $\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p(x_i;\beta) = \mathbb{E}|G = g_1|$.

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 β^{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 β^{old}.

Newton–Raphson Algorithm in Matrix Notation

Let us denote:

$\mathbf{y}$	the vector of y_i
$\mathbf{X}$	$N \times (p+1)$ data matrix x_i
$\mathbf{p}$	the vector of fitted probabilities with ith element $p(x_i; \beta^{old})$
$\mathbf{W}$	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 ($\beta^0 \leftarrow 0$)

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

- 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 myocardian 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.

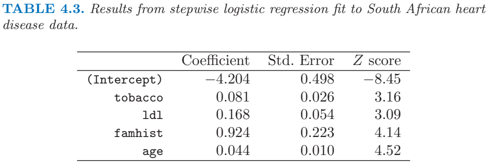


TABLE 4.3. *Results from stepwise logistic regression fit to South African heart disease data.*

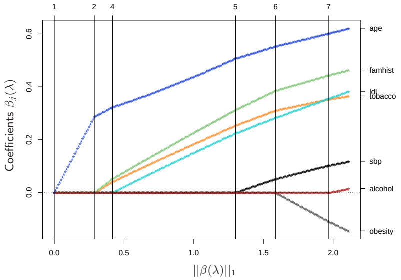
	Coefficient	Std. Error	Z score
`(Intercept)`	−4.204	0.498	−8.45
`tobacco`	0.081	0.026	3.16
`ldl`	0.168	0.054	3.09
`famhist`	0.924	0.223	4.14
`age`	0.044	0.010	4.52

- $P(MI|x_i,\theta) = \frac{e^{-4.204+0.081x_{tobacco}+0.168x_{ldl}+0.924x_{famhist}+0.044x_{age}}}{1+(e^{-4.204+0.081x_{tobacco}+0.168x_{ldl}+0.924x_{famhist}+0.044x_{age}})}$
- Interval estimate $odds_{tobacco} = e^{0.081\pm2\times0.026} = (1.03, 1.14)$ 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.
- β_0 is not included into the penalty.

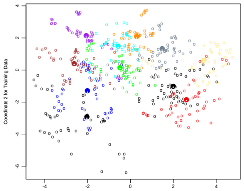


Linear Discriminant Analysis

- 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.



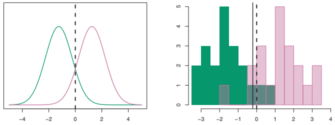


FIGURE 4.4. Left: *Two one-dimensional normal density functions are shown.*

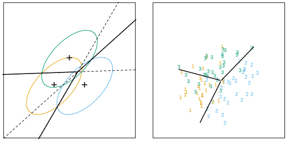
Linear Discriminant Analysis

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

$$\begin{aligned}
\hat{\pi}_k &= \frac{N_k}{N} \\
\hat{\mu}_k &= \frac{\sum_{\{x_i : G(x_i) = g_k\}} x_i}{N_k} \\
\hat{\Sigma} &= \sum_{k=1}^{K} \sum_{\{x_i : G(x_i) = g_k\}} \frac{(x_i - \mu_k)(x_i - \mu_k)^T}{(N - K)} \\
\phi_k(x) &= N(\mu_k, \Sigma) \\
P(G = g_k | X = x) &= \frac{\phi_k(x)\pi_k}{\sum_{\ell=1}^{K} \phi_\ell(x)\pi_\ell}
\end{aligned}$$

To classify new instance x we predict the G_k with maximal δ_k:

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



Quadratic Discriminant Analysis

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

$$
\begin{aligned}
\hat{\pi}_k &= \frac{N_k}{N} \\
\hat{\mu}_k &= \frac{\sum_{\{x_i : G(x_i)=g_k\}} x_i}{N_k} \\
\hat{\Sigma}_k &= \sum_{\{x_i : G(x_i)=g_k\}} \frac{(x_i - \mu_k)(x_i - \mu_k)^T}{(|G_k| - 1)} \\
f_k(x) &= N(\mu_k, \Sigma_k) \\
P(G = g_k | X = x) &= \frac{f_k(x)\pi_k}{\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell}
\end{aligned}
$$

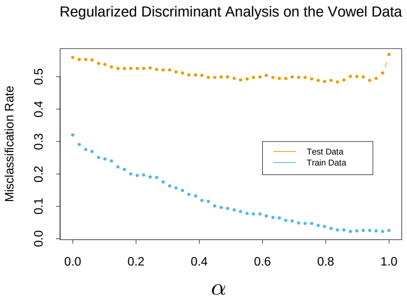
To classify new instance x we predict the G_k with maximal δ_k:

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

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.

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



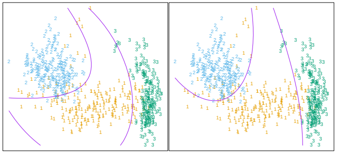
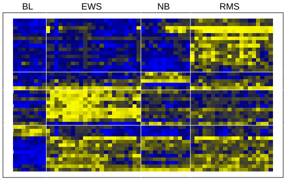


FIGURE 4.6. *Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $X_1, X_2, X_1X_2, X_1^2, X_2^2$). The right plot shows the*

Diagonal Linear Discriminant Analysis

- **With really many dimensions** we restrict Σ to a diagonal matrix.
- 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**

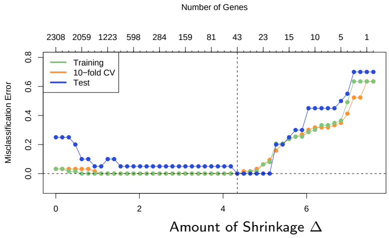
$$\delta_k(x^*) = -\sum_{j=1}^{p} \frac{(x_j^* - \overline{x}_{jk})^2}{s_j^2} + 2\log(\pi_k)$$

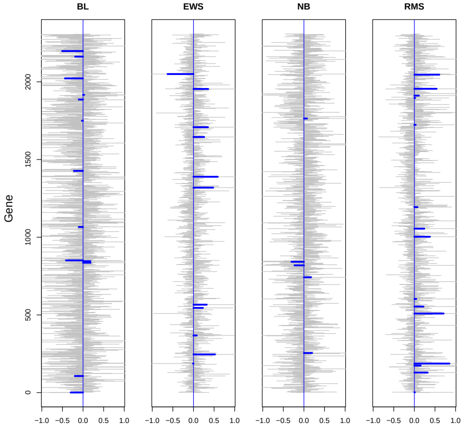
- s_j is the pooled within-class standard deviation of the jth gene
- $\overline{x}_{jk} = \sum_{i \in C_k} \frac{x_{ij}}{N_k}$
- $\tilde{x}_k = (\overline{x}_{1k}, \ldots, \overline{x}_{jk}, \ldots, \overline{x}_{pk})^T$ is the k class centroid.

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

Computations for LDA

Linear and Quadratic Discriminant Analysis

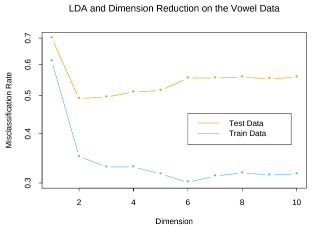
- $O(N^3)$, often $O(N^{2.376})$
- QDA and LDA may be computed using matrix decomposition:
  - Compute the eigendecomposition for each $(x-\hat{\mu}_k)^T\hat{\Sigma}_k^{-1}(x-\hat{\mu}_k) = [\mathbf{U}_k^T(x-\hat{\mu}_k)]^T\mathbf{D}_k^{-1}[\mathbf{U}_k^T(x-\hat{\mu}_k)]$
  - $\log|\hat{\Sigma}_k| = \sum_\ell \log d_{k\ell}$.
- 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 $\hat{\Sigma} = \mathbf{UDU}^T$.
    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 π_k.

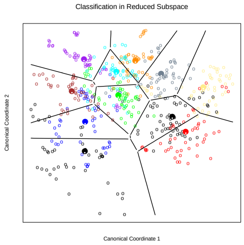
Vovel Example

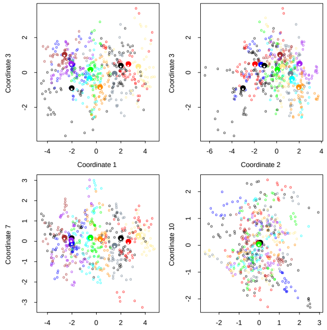
Example Vovel data ESL:

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





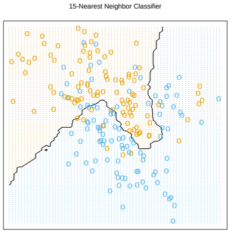


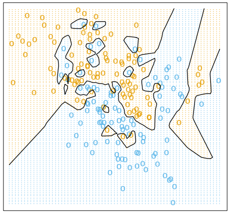
Nearest-Neighbor Methods

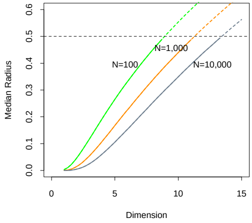
- The **nearest-neighbor methods** use those observations in the training set $\mathcal{T}$ closest in the input space to x to form $\hat{f}$.

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

- nice, but suffers the curse of dimensionality.



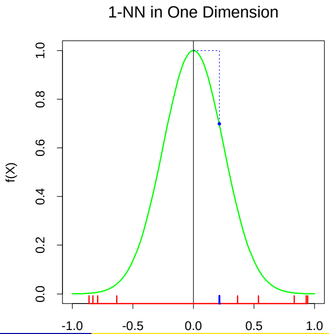


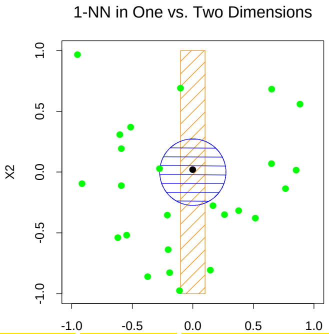


Curse of Dimensionality demonstration

Prediction

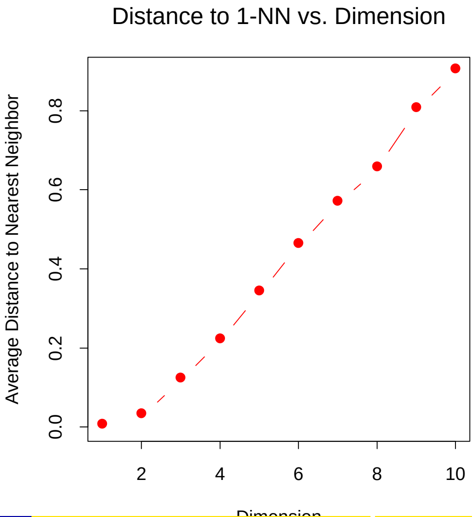
- Assume x_i uniformly generated form the interval $\langle -1, 1\rangle^p$
- We have $Y = f(X) = e^{-8\|X\|^2}$, without any noise, for x_i we know exactly $f(x_i)$.
- We use 1–NN to estimate $f(0)$ based on 1000 data sample.
- Predicted value for $x = \langle 0, \ldots, 0\rangle$ is lower that 1 and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

Disciminang Adaptive Nearest-Neighbor Methods (DANN)

- The metric at a query point is defined by

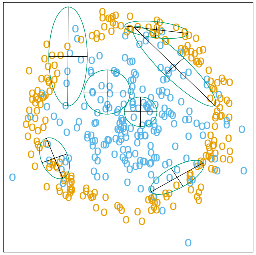
$$D(x, x_0) = (x - x_0)^T \Sigma (x - x_0)$$

- 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.

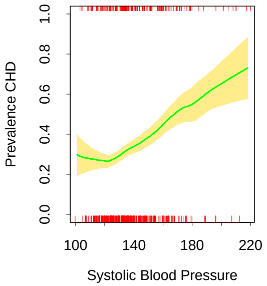
- Close to the class boundary, the Σ shrinks out of the boundary
- in the interior it remains circular.

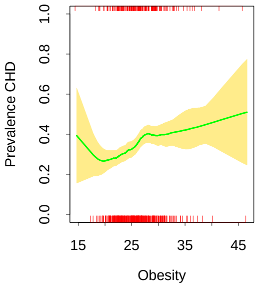


Local Likelihood and other methods

- 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_λ
- 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)

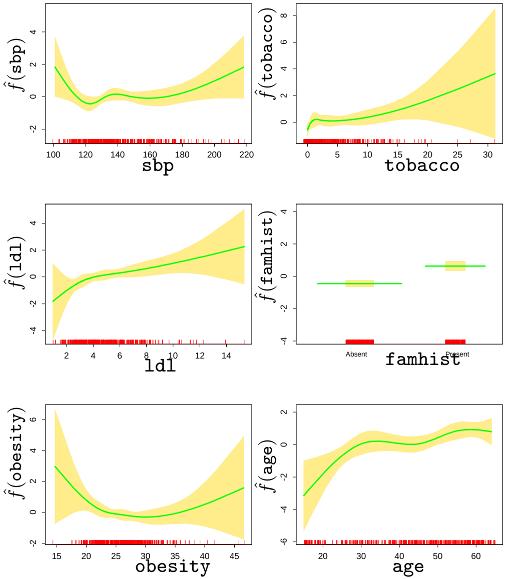
- **Each feature X_j is approximated by a natural spline.**
- The overall model is:

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

- θ_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 $\mathbf{H}$.
- 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

Separating hyperplane, Optimal separating hyperplane

- Classification, we encode the goal class by -1 and 1, respectively.
- separate the space X by a hyperplane
- **Linear Discriminant Analysis** LDA is not necessary optimal.
- **Logistic regression** finds one if it exists.
- **Perceptron** (a neural network with one neuron) finds separating hyperplane if it exists.
  - The exact position depends on initial parameters.

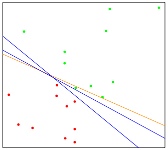


FIGURE 4.14. *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the* perceptron learning algorithm *with different random starts.*

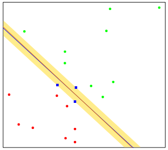


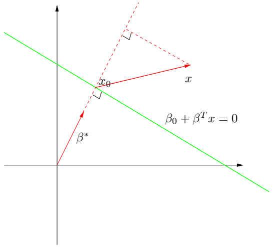
FIGURE 4.16. *The same data as in Figure 4.14. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the boundary found using logistic regression (red line), which is very close to the optimal separating hyperplane (see Section 12.3.3).*

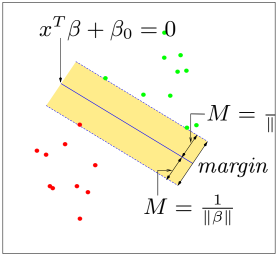
Optimal Separating Hyperplane (separble case)

We define **Optimal Separating Hyperplane** as a separating hyperplane with maximal free space M without any data point around the hyperplane.
Formally:

$$\max_{\beta,\beta_0,\|\beta\|=1} M$$

subject to $y_i(x_i^T\beta + \beta_0) \geq M$ for all $i = 1, \dots, N$.





Formally:

$$\max_{\beta,\beta_0,\|\beta\|=1} M$$

subject to $y_i(x_i^T\beta+\beta_0)\geq M$ for all $i=1,\ldots,N$.
We re-define: $\|\beta\|=1$ can be moved to the condition (and redefine β_0):

$$\frac{1}{\|\beta\|}y_i(x_i^T\beta+\beta_0)\geq M$$

Since for any β and β_0 satisfying these inequalities, any positively scaled multiple satisfies them too, we can set $\|\beta\|=\frac{1}{M}$ and we get:

$$\min_{\beta,\beta_0}\frac{1}{2}\|\beta\|^2$$

subject to $y_i(x_i^T\beta+\beta_0)\geq 1$ pro $i=1,\ldots,N$.
This is a convex optimization problem. The Lagrange function, we look for the saddle point w.r.t. β and β_0:

$$L_P=\frac{1}{2}\|\beta\|^2-\sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta+\beta_0)-1].$$

$$L_P = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{N} \alpha_i [y_i ({x_i}^T \beta + \beta_0) - 1].$$

Setting the derivatives to zero, we obtain:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i$$

Substituing these in L_P we obtain the so–called Wolfe dual:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k {x_i}^T x_k$$

subject to $\alpha_i \geq 0$
The solution is obtained by maximizing L_D in the positive orthant, for which standard software can be used.

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k$$

subject to $\alpha_i \geq 0$.
In addition the solution must satisfy the Karush–Kuhn–Tucker conditions:

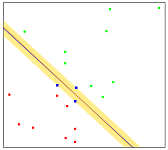
$$\alpha_i [y_i (x_i^T \beta + \beta_0) - 1] = 0$$

for any i, therefore for any $\alpha_i > 0$ must $[y_i (x_i^T \beta + \beta_0) - 1] = 0$, that means x_i is on the boundary and for all x_i outside the boundary is $\alpha_i = 0$.
The boundary is defined by x_i with $\alpha_i > 0$ – so called **support vectors**.

We classify new observations

$$\hat{G}(x) = sign(x^T \beta + \beta_0)$$

- where $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$,
- $\beta_0 = y_s - x_s^T \beta$ for any support vector $\alpha_s > 0$.



Optimal Separating Hyperplane (nonseparble case)

- We have to accept incorrectly classified instances in a non–separable case.
- We limit the number of incorrectly classified examples.

We define **slack** ξ for each data point $(\xi_1, \ldots, \xi_N) = \xi$ as follows:

- ξ_i is the distance of x_i from the boundary for x_i at the wrong side of the margin
- and $\xi_i = 0$, for x_i at the correct side.

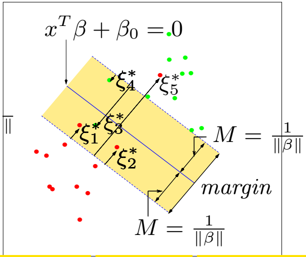
We require $\sum_{i=1}^{N} \xi_i \leq K$.
We solve the optimization problem

$$\max_{\beta, \beta_0, \|\beta\|=1} M$$

subject to:

$$y_i(x_i^T \beta + \beta_0) \geq M(1 - \xi_i)$$

where $\forall i$ is $\xi_i \geq 0$ a $\sum_{i=1}^{N} \xi_i \leq K$.



Optimal Separating Hyperplane (nonseparble case)

Again, we omit replace the condition $\|\beta\|$ by defining $M = \frac{1}{\|\beta\|}$ and optimize

$$\min \|\beta\| \text{ subject to } \begin{cases} y_i(x^T\beta + \beta_0) \geq (1 - \xi_i) \forall i \\ \xi_i \geq 0, \sum \xi_i \leq constant \end{cases}$$

We replace the constant by a multiplicative parameter γ and solve

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{N} \xi_i$$

subject to $\xi_i \geq 0$ and $y_i(x^T\beta + \beta_0) \geq (1 - \xi_i)$.

- We can set $\gamma = \infty$ for the separable case.
- Large γ: a complex boundary, fewer support vectors.
- Small γ: a smooth boundary, a robust model, many support vectors.
- γ usually set by crossvalidation.

We solve

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{N} \xi_i$$

subject to $\xi_i \geq 0$ and $y_i(x_i^T \beta + \beta_0) \geq (1 - \xi_i)$.
Lagrange multipliers again for α_i, μ_i:

$$L_P = \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^{N} \mu_i \xi_i$$

Setting the derivative $= 0$ we get:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i$$

$$\alpha_i = \gamma - \mu_i.$$

Substitute to get Wolfe dual:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k {x_i}^T x_k$$

and maximize L_D subject to $0 \leq \alpha_i \leq \gamma$ a $\sum_{i=1}^{N} \alpha_i y_i = 0$.
Solution satisfies:

$$\begin{aligned} \alpha_i [y_i({x_i}^T \beta + \beta_0) - (1 - \xi_i)] &= 0 \\ \mu_i \xi_i &= 0 \\ \left[y_i({x_i}^T \beta + \beta_0) - (1 - \xi_i)\right] &\geq 0 \end{aligned}$$

- The solution is $\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$.
- **support points** with nonzero coefficients $\widehat{\alpha_i}$ are
  - points at the boundary
    - $\widehat{\xi_i} = 0$ (therefore $0 < \widehat{\alpha_i} < \gamma$),
  - and points on the wrong side of the margin
    - $\widehat{\xi_i} > 0$ (and $\widehat{\alpha_i} = \gamma$).
- Any point with $\widehat{\xi_i} = 0$ can be used to calculate $\widehat{\beta_0}$, typically an average.
  - $\widehat{\beta_0}$ for a boundary point $\alpha_i > 0$, $\xi_i = 0$:

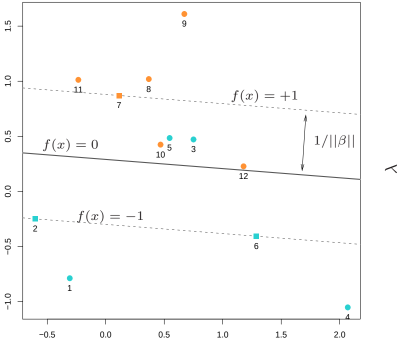
$$\alpha_i \left[y_i(x^T \widehat{\beta} + \widehat{\beta_0}) - (1 - 0) \right] = 0$$

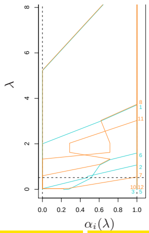
- Parameter γ settled by tuning (crossvalidation)

SVM Solution

- The solution is $\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$.
- **support points** with nonzero coefficients $\widehat{\alpha_i}$ are
  - points at the boundary
    - $\widehat{\xi_i} = 0$ (therefore $0 < \widehat{\alpha_i} < \gamma$),
  - and points on the wrong side of the margin
    - $\widehat{\xi_i} > 0$ (and $\widehat{\alpha_i} = \gamma$).
- Any point with $\widehat{\xi_i} = 0$ can be used to calculate $\widehat{\beta_0}$, typically an average.
  - $\widehat{\beta_0}$ for a boundary point $\xi_i = 0$:

$$\alpha_i \left[y_i (x^T \widehat{\beta} + \widehat{\beta_0}) - (1 - 0) \right] = 0$$





- $\alpha = \xi = 0$ for points 1,4,8,9,11
- $\alpha > 0, \xi = 0$ for points 2,6,8
- missclassified points 3,5.

Support Vector Machines

Let us have the training data $(x_i, y_i)_{i=1}^N$, $x_i \in \mathbb{R}^p$, y_i in $\{-1, 1\}$. We define a hyperplane

$$\{x : f(x) = x^T\beta + \beta_0 = 0\} \tag{1}$$

where $\|\beta\| = 1$.
We classify according to

$$G(x) = sign\left[x^T\beta + \beta_0\right]$$

where $f(x)$ is a signed distance of x from the hyperplane.
Support vector machines replace the scalar product $\langle x_i, x\rangle$ by a kernel function.

$$\begin{aligned}
\hat{f}(x) &= \beta x + \hat{\beta}_0 \\
\hat{f}(x) &= \sum_{k=1}^{N} \hat{\alpha}_i y_i x_i^T x + \hat{\beta}_0 \\
\hat{f}(x) &= \sum_{k=1}^{N} \hat{\alpha}_i y_i \langle x_i, x\rangle + \hat{\beta}_0 \\
\hat{f}(x) &= \sum^{N} \hat{\alpha}_i y_i K(x_i, x) + \hat{\beta}_0
\end{aligned}$$

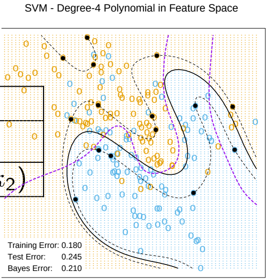
SVM Example

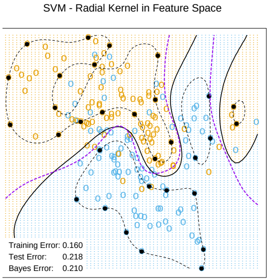
- **kernel functions** are function to replace scalar product with a scalar product in a transformed space.

dth Degree polynomial:	$K(x,x^{\|}) = (1 + \langle x, x^{\|}\rangle)^d$
Radial basis	$K(x,x^{\|}) = exp(\frac{-\|x-x^{\|}\|^2}{\ell})$
Neural network	$K(x,x^{\|}) = tanh(\kappa_1\langle x, x^{\|}\rangle + \kappa_2)$

- For example a degree 2 with two dimensional input: $K(x,x') = (1+\langle x,x'\rangle)^2 =$ $(1 + 2x_1x_1' + 2x_2x_2' + (x_1x_1')^2 + (x_2x_2')^2 + 2x_1x_1'x_2x_2')$
- that is $M = 6$, $h_1(x) = 1$, $h_2(x) = \sqrt{2}x_1$, $h_3(x) = \sqrt{2}x_2$, $h_4(x) = x_1^2$, $h_5(x) = x_2^2$, $h_6(x) = \sqrt{2}x_1x_2$.

The classification function $\hat{f}(x) = h(x)^T\beta + \beta_0 = \sum_{i=1}^N \alpha_i y_i \langle h(x), h(x_i)\rangle + \beta_0$ does not need evaluation of $h(i)$, only the scalar product $\langle h(x), h(x_i)\rangle$.





String Kernels and Protein Classification

IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDY**LQE**FLGVMNTEWI

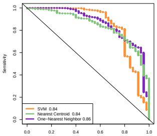
PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAER**LQE**NLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK

- Consider all possible sequences of length m.
- We define a feature map

$$\Phi_m(x) = \{\phi_a(x)\}_{a \in \mathcal{A}_m}$$

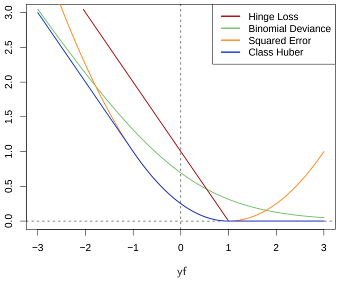
- The kernel function is the inner product:

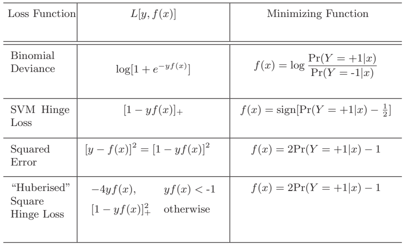
$$K_m(x_1, x_2) = \langle \Phi_m(x_1), \Phi_m(x_2) \rangle.$$



SVM as a Penalization Method

- We fit a linear function wrt. basis $\{h_i(x)\}$: $f(x) = h^T\beta + \beta_0$.
- Consider the loss function $L(y, f) = [1 - yf]_+$
  - The optimization problem $min_{\beta_0,\beta} \sum_{i=1}^N [1 - yf]_+ + \lambda\|\beta\|^2$
- is equivalent to SVM
  - $\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^N \xi_i$
  - subject to $\xi_i \geq 0$ and $y_i(x^T\beta + \beta_0) \geq (1 - \xi_i)$.
- is similar to smoothing splines penalty:
  - $\min_{\alpha,\alpha_0} \sum_{i=1}^N [1 - yf]_+ + \lambda\alpha^T\mathbf{K}\alpha$
  - where $\alpha^T\mathbf{K}\alpha = J(f)$ is the smoothing penalty.





Loss Function	$L[y, f(x)]$	Minimizing Function
Binomial Deviance	$\log[1 + e^{-yf(x)}]$	$f(x) = \log \frac{\Pr(Y=+1\|x)}{\Pr(Y=-1\|x)}$
SVM Hinge Loss	$[1 - yf(x)]_+$	$f(x) = \text{sign}[\Pr(Y=+1\|x) - \frac{1}{2}]$
Squared Error	$[y - f(x)]^2 = [1 - yf(x)]^2$	$f(x) = 2\Pr(Y=+1\|x) - 1$
"Huberised" Square Hinge Loss	$-4yf(x)$, $yf(x) <$ -1 $[1 - yf(x)]_+^2$ otherwise	$f(x) = 2\Pr(Y=+1\|x) - 1$

SVM and Kernel Dimsension

- The first Simulated example
  - 100 observations of each class
  - First class: four standard normal independent features X_1, X_2, X_3, X_4.
  - Second class conditioned on $9 \leq \sum X_j^2 \leq 16$.
- Second example
  - The first one augmented with an additional six standard Gaussian noise features.
- BRUTTO: Additive spline model.
- BRUTTO and MARS has the ability to ignore noisy features.
- We can see the overfitting of SVM. The degree 2 polynomial kernel is the best since the decision boundary is quadratic.

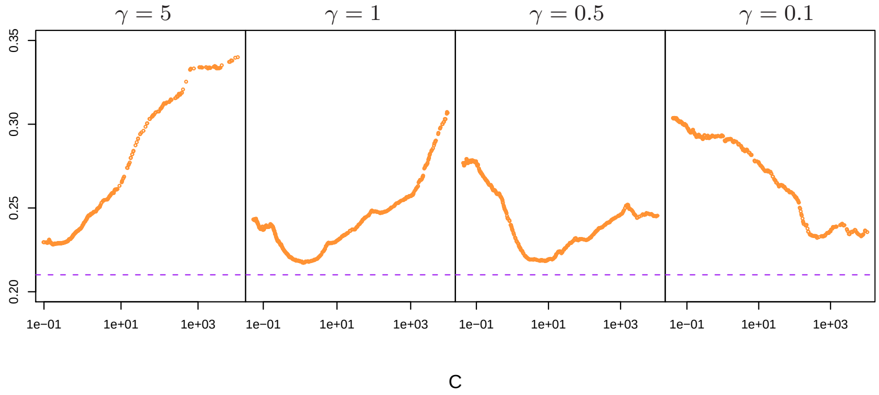
	Test Error (SE)	
Method	No Noise Features	Six Noise Features
SV Classifier	0.450 (0.003)	0.472 (0.003)
SVM/poly 2	0.078 (0.003)	0.152 (0.004)
SVM/poly 5	0.180 (0.004)	0.370 (0.004)
SVM/poly 10	0.230 (0.003)	0.434 (0.002)
BRUTO	0.084 (0.003)	0.090 (0.003)
MARS	0.156 (0.004)	0.173 (0.005)
Bayes	0.029	0.029

SVM

SVM Complexity

The SVM complexity is $m^3 + mN + mpN$, where m is the number of support vectors.

Parameter tuning for different radial basis lengthscale γ.



SVM for Regression

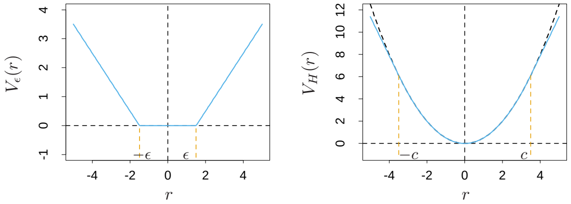
- In regression, we fit a function: $f(x) = x^T\beta + \beta_0$
- We consider error function V_ϵ (left figure)

$$V_\epsilon(r) = \begin{cases} 0 & \text{if } |r| < \epsilon, \\ |r| - \epsilon, & \text{otherwise.} \end{cases}$$

- and minimize:

$$H(\beta, \beta_0) = \sum_{i=1}^{N} V_\epsilon(y_i - f(x_i)) + \frac{\lambda}{2}\|\beta\|^2$$

-



SVM for Regression 2

- The solution has the form: $\hat{\alpha}_i, \hat{\alpha}_i^* \geq 0$

$$\begin{aligned} \hat{\beta} &= \sum_{i=1}^{N} (\hat{\alpha}_i^* - \hat{\alpha}_i) x_i, \\ \hat{f}(x) &= \sum_{i=1}^{N} (\hat{\alpha}_i^* - \hat{\alpha}_i) \langle x, x_i \rangle + \beta_0, \end{aligned}$$

- and solve the quadratic programming problem

$$\min_{\alpha_i, \alpha_i^*} \epsilon \sum_{i=1}^{N} (\alpha_i^* + \alpha_i) - \sum_{i=1}^{N} y_i (\alpha_i^* - \alpha_i) + \frac{1}{2} \sum_{i,i'=1}^{N} (\alpha_i^* - \alpha_i)(\alpha_{i'}^* - \alpha_{i'}) \langle x_i, x_{i'} \rangle$$

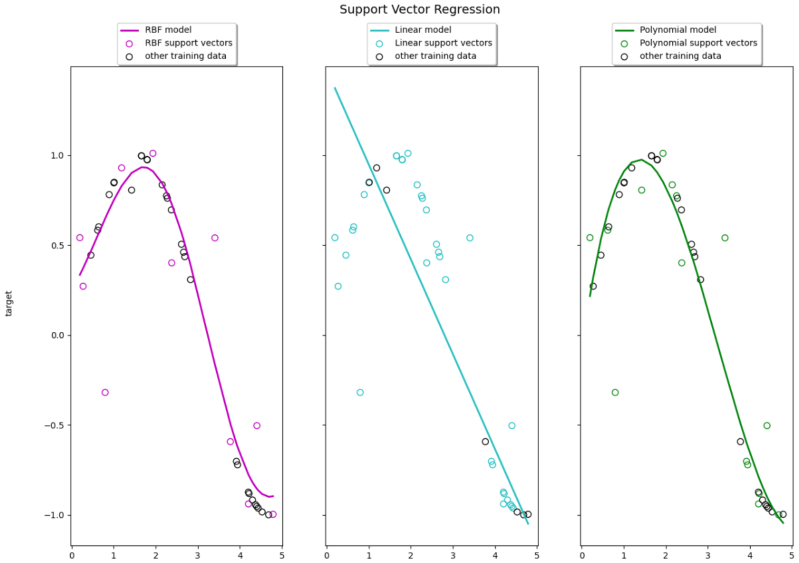
- subject to the constraints

$$\begin{gathered} 0 \leq \alpha_i,\ \alpha_i^* \leq 1/\lambda, \\ \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) = 0, \\ \alpha_i \alpha_i^* = 0. \end{gathered}$$

- Support vectors are those with nonzero $(\hat{\alpha}_i^* - \hat{\alpha}_i)$.
- With scaled response y, you may use the default ϵ.
- λ is tuned by cross-validation.

SVR

sklearn.svm.SVR



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).

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

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. Bayesian learning, EM algorithm
8. Clustering
9. Association Rules, Apriori
10. Inductive Logic Programming
11. Undirected Graphical Models
12. Gaussian Processes
13. Support Vector Machines
14. (PCA Extensions, Independent CA)