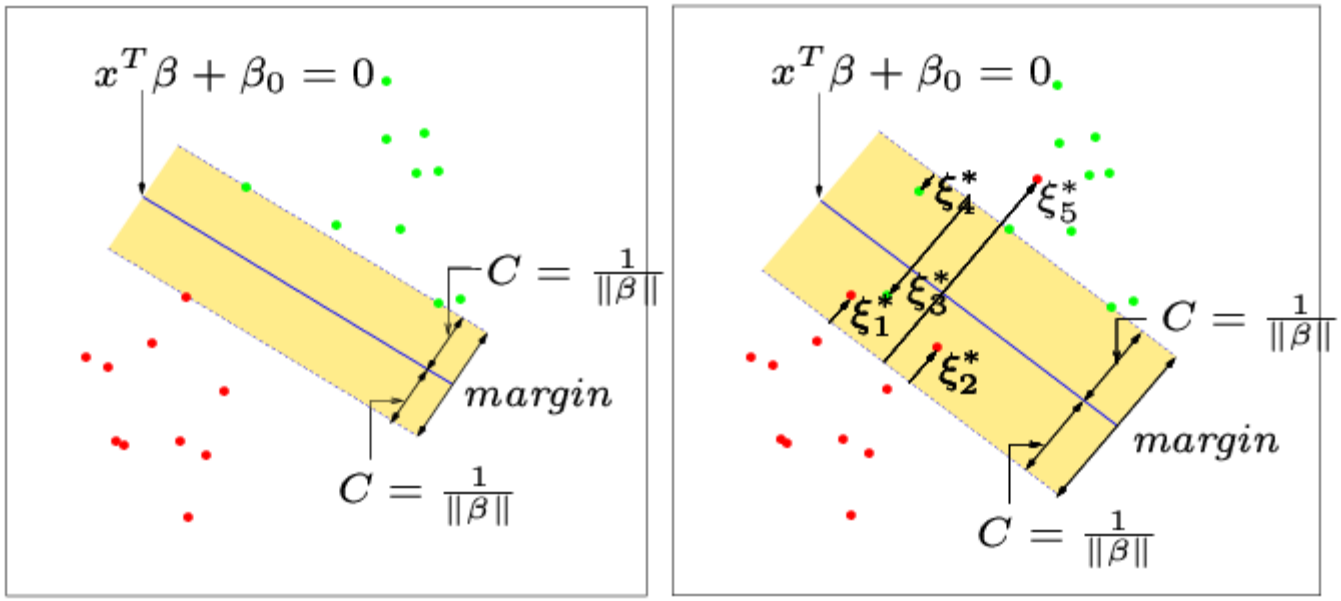
Support Vector Machines et al.



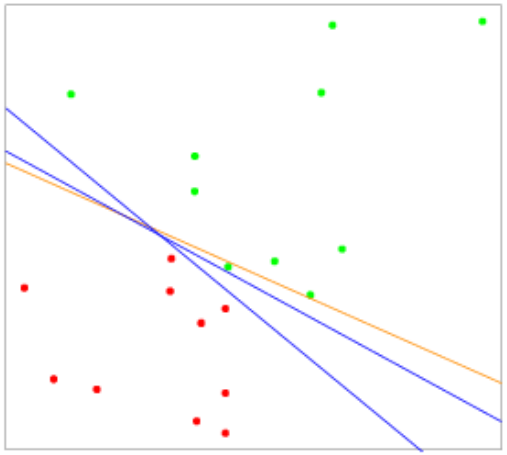
Univ.-Lektor Dr.techn. Alexander K. Seewald
Österreichisches Forschungsinstitut
für Artificial Intelligence

alex@seewald.at / alex.seewald.at

Linear Models Revisited

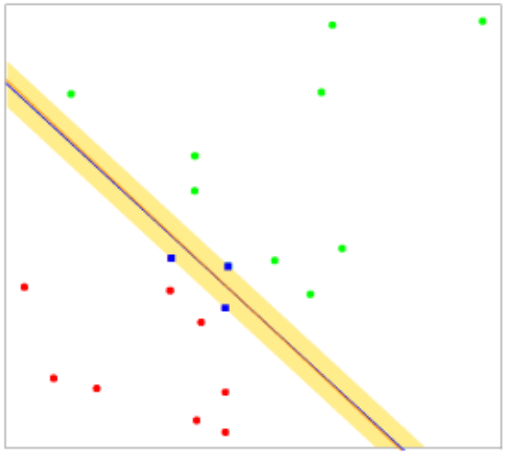
Previously explained

- Linear Regression: very efficient, but yields suboptimal classification performance
- Perceptron: good classification performance (converges to an arbitrary separating hyperplane) However, no convergence guaranteed if data are not linearly separable! Stochastic algorithm: Learns different hyperplane depending on (random) starting point.



Today's Lecture Plan

- Support Vector Machine: maximum margin hyperplane explicitly maximizes classification performance; guaranteed convergence (convex decision problem); deals gracefully with not linearly separable data.
- Logistic Regression: Simpler related model, also works quite well in practice.



alex@seewald.at / alex.seewald.at

Logistic Regression

Logistic Regression arises from the desire to model posterior probabilities via linear functions in **x**, while ensuring they sum to one and remain within [0,1] (one *regularization* approach among many). LR is used in data analysis and inference, where the goal is to understand the role of the input variables in *explaining* the outcome. The model has the form:

$$\left.\begin{aligned}
\log\frac{P(Cl_1\mid\mathbf{x})}{P(Cl_K\mid\mathbf{x})} &= \beta_{10}+\beta_1{}^T\mathbf{x}\\
\log\frac{P(Cl_2\mid\mathbf{x})}{P(Cl_K\mid\mathbf{x})} &= \beta_{20}+\beta_2{}^T\mathbf{x}\\
&\vdots\\
\log\frac{P(Cl_{K-1}\mid\mathbf{x})}{P(Cl_K\mid\mathbf{x})} &= \beta_{(K-1)0}+\beta_{K-1}{}^T\mathbf{x}
\end{aligned}\right\}
\begin{aligned}
P(Cl_k\mid\mathbf{x}) &= \frac{\exp(\beta_{k0}+\beta_k{}^T\mathbf{x})}{1+\sum_{j=1}^{K-1}\exp(\beta_{j0}+\beta_j{}^T\mathbf{x})},\ k=1,2,\ldots K-1\\
P(Cl_K\mid\mathbf{x}) &= \frac{1}{1+\sum_{j=1}^{K-1}\exp(\beta_{j0}+\beta_j{}^T\mathbf{x})}\\
\text{Parameter set } \theta &= \{\beta_{10},\beta_1,\beta_{20},\beta_2,\ldots,\beta_{(K-1)0},\beta_{(K-1)}\}
\end{aligned}$$

Very simple for two classes (K=2), since then there is only a single linear model. Widely used in biostatistical applications where binary responses (two classes) occur quite frequently. For example, patients survive or die, have heart disease or not, or a condition is present or absent. θ is usually chosen by maximum likelihood, thus ignoring *P(TD)* and assuming uniform *P(f)*.

alex@seewald.at / alex.seewald.at

Logistic Regression (2)

For simplification, let K=2, and let the two classes be *yes* (=1) and *no* (=0).

Let $p_1(\mathbf{x_i};\theta) = P(y_i=yes \mid \mathbf{x_i} \in TD; \theta)$ and $p_2(\mathbf{x_i};\theta) = P(y_i=no \mid \mathbf{x_i} \in TD; \theta) = 1-p_1(\mathbf{x_i};\theta)$.

Assume all training examples $\mathbf{x_i}$ contain the constant term $x_{i0}=1$ to accomodate the intercept (constant term) β_{10}. $\beta=\{\beta_{10}, \beta_1\}= \theta$. Then the log-likelihood P(f|TD) can be written as:

$$P(f \mid TD) = P(\beta \mid TD) = \ell(\beta) = \sum_{i=1}^{|TD|} y_i \log p_1(\mathbf{x_i};\beta) + (1-y_i)\log p_2(\mathbf{x_i};\beta)) =$$

$$= \sum_{i=1}^{|TD|} y_i \beta^T \mathbf{x_i} - \log(1+\exp(\beta^T \mathbf{x_i}))$$

To maximize log-likelihood $\ell(\beta)$, set its derivative to zero:

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{|TD|} \mathbf{x_i}(y_i - p_1(\mathbf{x_i};\beta)) = 0$$

...which are $p+1$ equations nonlinear in β.

Direct solution is not possible, need iterative algorithm!

alex@seewald.at / alex.seewald.at

Logistic Regression (3)

Solve nonlinear equation of log-likelihood via Newton-Raphson algorithm

Assume N=|TD|, p=num. of attributes in TD and...

- $\mathbf{X}$ is an N×(p+1) matrix with each row an input vector $\mathbf{x_i}$ (extended with $x_{i0}=1$)
- $\mathbf{y}$ is an N-vector of the outputs from TD.
- $\mathbf{W}$ is an N×N diagonal matrix with $\mathbf{W_{ii}} = p_1(\mathbf{x_i};\beta^{old})\,(1- p_1(\mathbf{x_i};\beta^{old}))$; $\mathbf{W_{ij}}=0$ f. $i\neq j$
- $\mathbf{p}$ is the vector of fitted probabilities $p_1(\mathbf{x_i};\beta^{old})$
- $\mathbf{z}=\mathbf{X}\beta^{old}+\mathbf{W}^{-1}(\mathbf{y}-\mathbf{p})$ (*adjusted response*)

$\Rightarrow$ $\beta^{new} = (\mathbf{X^T W X})^{-1}\mathbf{X^T W z}$ computes the new β from the old one.

Start with e.g. $\beta = \{0\}^{p+1}$, repeat until convergence.

Also called *iteratively reweighted least squares*, because it solves a weighted linear regression problem in each iteration step (compare with (unweighted) linear regression: $\mathbf{X^T}(\mathbf{y}-\mathbf{X}\mathbf{w})=0 \Leftrightarrow \mathbf{w} = (\mathbf{X^T X})^{-1}\mathbf{X^T y}$)

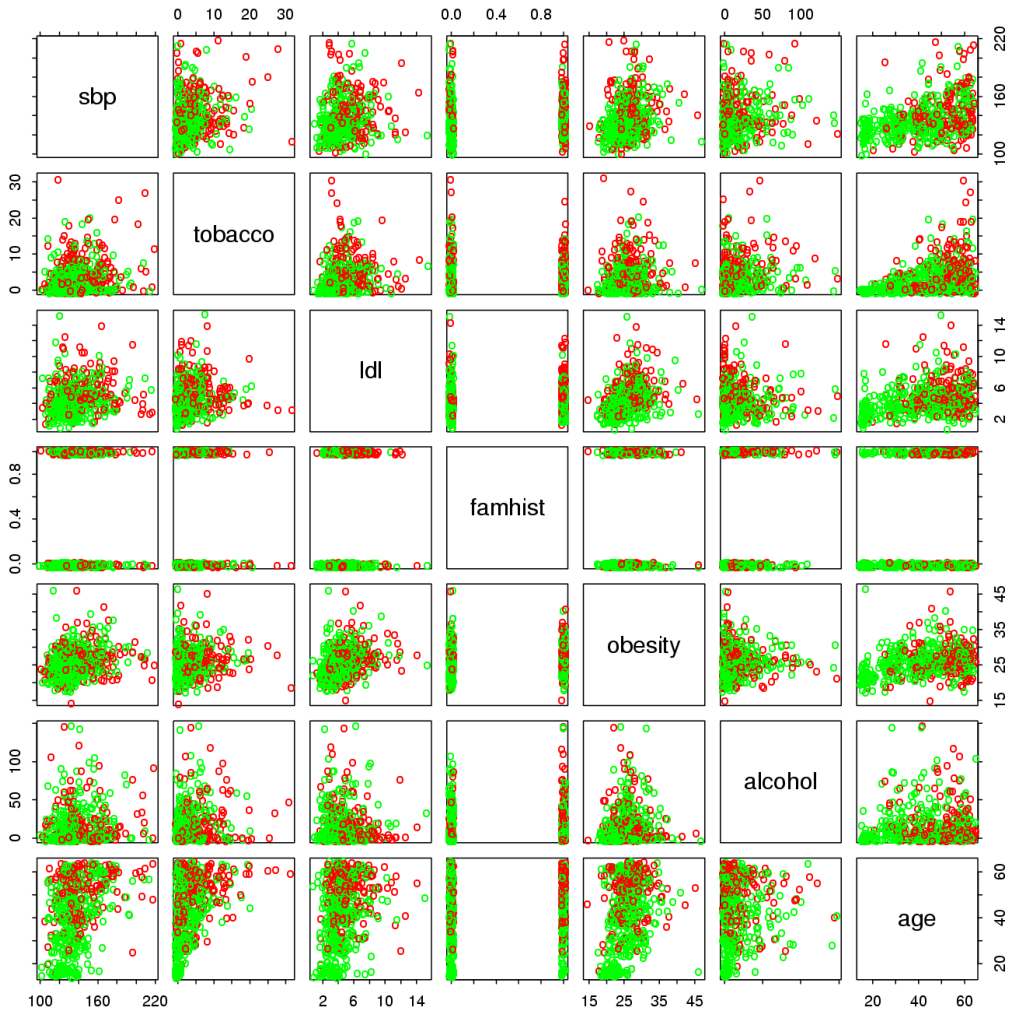
In the two-class case the obtained hyperplane is very similar to SVMs, provided the training data is linearly separable. Similar to SVMs, the training examples further away from the hyperplane have less influence on the final model.

alex@seewald.at / alex.seewald.at

Example: African Heart Disease

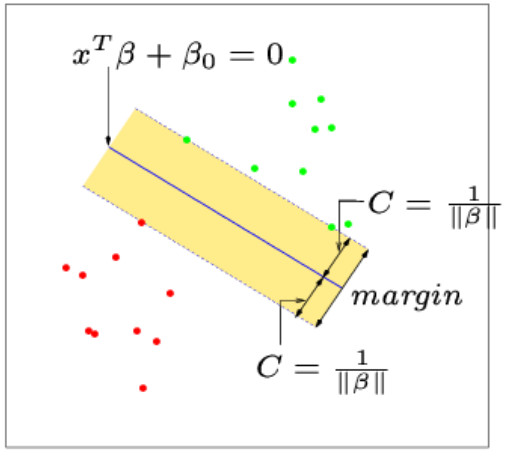
Attribute	Coeff.	ZScore
sbp	0.006	1.023
tobacco	0.080	**3.034**
ldl	0.185	**3.219**
famhist	0.939	**4.178**
obesity	-0.035	-1.187
alcohol	0.001	0.136
age	0.043	**4.184**
(Constant)	-4.130	**-4.285**

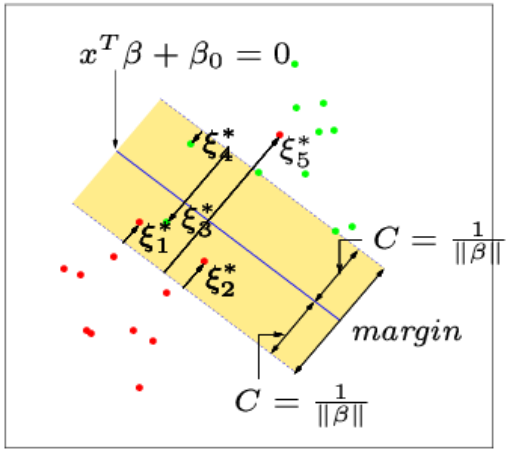
Logistic Regression yields the model shown above. Z-Score computes the Wald test with null hypothesis = given coefficient is zero while all others are not zero. Approximately significant if |Z-Score|>2 (shown in **bold**)



alex@seewald.at / alex.seewald.at

Support Vector Machines





Initially, a linear model in **x**. We don't minimize residual squared error (RSS) or log-likelihood, but maximize the margin $\|\beta\|$ resp. minimize $C=1/\|\beta\|$. $f(\mathbf{x}) = \mathbf{x}^T.\beta+\beta_0$ is our function/model, where $y_i \in \{+1,-1\}$ (i.e. the sign of $f(\mathbf{x})$ determines the class)**,** but the weights β and constant term β_0 are determined differently. This new regularization again guarantees an unique solution.

- If the data is linearly separable, we minimize $\|\beta\|$ subject to the constraints $y_i(\mathbf{x_i}^T.\beta+\beta_0)\geq 1$ for $\forall$ i=1,2,...|TD|. See top left figure.
- If the data is not linearly separable, we introduce slack variables ξ_i to let some examples be on the wrong side of the margin. See top right figure.

$$\min\|\beta\| \text{ subject to } \begin{cases} y_i(\mathbf{x_i}^T\beta+\beta_0) \geq 1-\xi_i \\ \xi_i \geq 0, \sum_{i=1}^{|TD|}\xi_i \leq \text{constant} \approx \frac{1}{\lambda} \end{cases}$$

alex@seewald.at / alex.seewald.at

Support Vector Machines (2)

This optimization problem is quadratic with linear inequality constraints and is thus convex. A quadratic programming solution using Lagrange multipliers is therefore feasible. An equivalent form of the nonseparable case is:

$$\min \frac{1}{2}\|\beta\|^2 + \lambda \sum_{i=1}^{|TD|} \xi_i \text{ subject to } \xi_i \geq 0,\ y_i(\mathbf{x_i}^{\mathbf{T}}\beta + \beta_0) \geq 1 - \xi_i \ \forall i$$

Parameter λ determines the weight given to optimizing the slack variables ξ_i versus optimizing the margin. The separable case corresponds to $\lambda=\infty$.

The Lagrange primal function combines minimalization and constraints into a single formula. The constraints are weighted by Lagrange multipliers α_i and μ_i.

$$L_p = \frac{1}{2}\|\beta\|^2 + \lambda \sum_{i=1}^{|TD|} \xi_i - \sum_{i=1}^{|TD|} \alpha_i [y_i(\mathbf{x_i}^{\mathbf{T}}\beta + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^{|TD|} \mu_i \xi_i$$

which will be maximized w.r.t. β, β_0 and ξ_i. Setting the derivates to zero yields:

$$\beta = \sum_{i=1}^{|TD|} \alpha_i y_i \mathbf{x_i}$$

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

$$\mu_i = \lambda - \alpha_i$$

$$\alpha_i, \mu_i, \xi_i \geq 0$$

Substituting into L_P yields the Lagrange (Wolfe) dual objective function

$$L_D = \sum_{i=1}^{|TD|} \alpha_i - \frac{1}{2} \sum_{i=1}^{|TD|} \sum_{j=1}^{|TD|} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^{\mathbf{T}} \mathbf{x_j}$$

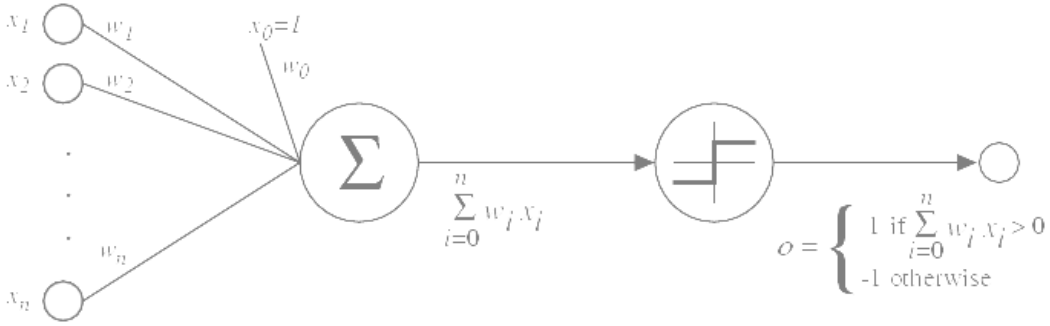
and the Karush - Kuhn - Tucker conditions (indicates a solution)

$$(\lambda - \alpha_i)\xi_i = 0;\ \alpha_i[y_i(\mathbf{x_i}^{\mathbf{T}}\beta + \beta_0) - (1 - \xi_i)] = 0$$

$$y_i(\mathbf{x_i}^{\mathbf{T}}\beta + \beta_0) - (1 - \xi_i) \geq 0 \text{ (resp. for } \forall i)$$

alex@seewald.at / alex.seewald.at

A high-bias learner: Perceptron



If the weight vector **w** is initially set to all zeros, the final **w** after convergence will be a linear combination of the training examples, similar to a SVM.

Perceptron (linear binary threshold unit, linear model)

- Computes a linear function of **x** (assume adding an $x_0=1$ to **x**, so that constant term w_0 can be handled). $f(\mathbf{x}) = \text{sign}(\mathbf{x}^T.\mathbf{w})$, **w** is initialized randomly.
- *Perceptron training rule*: $\mathbf{w} \leftarrow \mathbf{w}+\eta(y-f(\mathbf{x})).\mathbf{x}^T$, where y is the true output value from training data (±1), and η is the learning rate.
- Intuitively, concept boundary is a hyperplane which separates classes +1 & -1.

  Update rule is applied to each training example in turn, repeating until all training examples are classified correctly. Provided η is small enough, and the training set is linearly separable, this algorithm converges in a finite number of steps. If data is not linearly separable, convergence is not assured.

alex@seewald.at / alex.seewald.at

Support Vector Machines (3)

Features of the optimization problem for SVMs

- No local minima, only one global minimum – the maximal margin hyperplane.
- To solve the dual problem we need only the dot product $\mathbf{x_i}^T\mathbf{x_j}$ for each combination of training instances. The function computing the dot product, *K(u,v)* is called *kernel*. This **kernel trick** enables us to expand the original feature space via $\phi(x)$, thus learning a maximum margin hyperplane in higher-dimensional feature space which gives a nonlinear decision boundary in the original, lower-dimensional feature space. Usually only a few α_i are nonzero - the associated examples $\mathbf{x_i}$ are called *support vectors*.

$$L_D = \sum_{i=1}^{|TD|}\alpha_i - \frac{1}{2}\sum_{i=1}^{|TD|}\sum_{j=1}^{|TD|}\alpha_i\alpha_j y_i y_j K(\mathbf{x_i},\mathbf{x_j}) \text{ where } K(u,v) = \langle\phi(u)\cdot\phi(v)\rangle$$

$$\phi : \Re^p \mapsto \Re^m \ (m >> p) \text{ and } \langle\phi(u)\cdot\phi(v)\rangle = \sum_{i=1}^{m}\phi_i(u)\phi_i(v) \ = \text{the dot product.}$$

- The weight vector β is a linear combinations of training examples $\mathbf{x_i}$. We can use this relation to compute model f(**x**) via the kernel function. This allows us even an infinite-dimensional $\phi(x)$, i.e. $m=\infty$, without explicit computation of β.

$$f(\mathbf{x}) = \sum_{i=1}^{|TD|}\alpha_i y_i K(\mathbf{x},\mathbf{x_i}) + \beta_0$$

alex@seewald.at / alex.seewald.at

Example: A Polynomial Kernel of Degree 2

$$K(u,v) = \langle u \cdot v \rangle^2 = \left(\sum_{i=1}^{p} u_i v_i \right)^2 = \left(\sum_{i=1}^{p} u_i v_i \right) \left(\sum_{j=1}^{p} u_j v_j \right) \sum_{i=1}^{p} \sum_{j=1}^{p} u_i u_j v_i v_j = \sum_{(i,j)=(1,1)}^{(p,p)} (u_i u_j)(v_i v_j)$$

which is equivalent to a dot product using $\phi(u) = (u_i u_j)_{(i,j)=(1,1)}^{(p,p)}$

Usually only the kernel function *K(u,v)* is defined explicitly, and the feature mapping ϕ is defined implicitly. Not all functions can be written as dot product

⇒ Necessary and sufficient conditions for a kernel function in the finite case

- **Symmetry**: *K(u,v)=K(v,u)*
- **Cauchy-Schwarz Inequality**: $K(u,v)^2 \leq K(u,u)K(v,v)$
- Kernel Matrix **K** is **positive semi-definite** ($\mathbf{x}^T\mathbf{K}\mathbf{x} \geq 0$ for all $\mathbf{x} \neq \{0\}^p$)

$$\mathbf{K} = \left(K(\mathbf{x_i}, \mathbf{x_j}) \right)_{(i,j)=(1,1)}^{(n,n)}$$

In the infinite-dimensional case: *Mercer's Theorem*

Common kernel functions

polynomial kernel of degree *d*: $K(u,v)=(\langle u.v \rangle + c)^d$ (*linear kernel* if $d=1$)

Radial basis function (RBF): $K(u,v)=\exp(-\|u-v\|^2/c)$

Kernels may be used as background domain knowledge, but are quite opaque.

alex@seewald.at / alex.seewald.at

Sequential Minimal Optimization (SMO)

Many ways exist to solve the dual optimization problem iteratively. We will focus on one simple algorithm, Sequential Minimal Optimization (SMO).

- Start with $\alpha_i=0$ for all *i*. This ensures that $\Sigma\alpha_i y_i=0$ initially.
- Choose α_i , α_j arbitrarily (usually by heuristic to speed up convergence)
- The partial solution for α_i and α_j can be computed analytically:

$$\alpha_j^{new,unc} = \alpha_j^{old} + \frac{y_j\left((f(\mathbf{x_i})-y_i)-(f(\mathbf{x_j})-y_j)\right)}{K(\mathbf{x_i},\mathbf{x_i})+K(\mathbf{x_j},\mathbf{x_j})-2K(\mathbf{x_i},\mathbf{x_j})}$$

$$\alpha_j^{new} = \begin{cases} V & \text{if } \alpha_j^{new,unc} > V \\ \alpha_j^{new,unc} & \text{if } U \le \alpha_j^{new,unc} \le V \\ U & \text{if } \alpha_j^{new,unc} < U \end{cases}$$

$$\alpha_i^{new} = \alpha_i^{old} + y_i y_j (\alpha_j^{old} - \alpha_j^{new})$$

$$U = \begin{cases} \max(0, \alpha_j^{old} - \alpha_i^{old}) & \text{if } y_i \neq y_j \\ \max(0, \alpha_j^{old} + \alpha_i^{old} - \lambda) & \text{if } y_i = y_j \end{cases}$$

$$V = \begin{cases} \min(\lambda, \alpha_j^{old} - \alpha_i^{old} + \lambda) & \text{if } y_i \neq y_j \\ \min(\lambda, \alpha_j^{old} + \alpha_i^{old}) & \text{if } y_i = y_j \end{cases}$$

This brings us one step nearer to the solution by increasing L_D while maintaining the simpler constraints for the dual problem, i.e. $\Sigma\alpha_i y_i=0$

Repeat until convergence

Determine $\beta_0=y_i-f(\mathbf{x_i})$ (computing $f(\mathbf{x})$ with $\beta_0=0$) by averaging over all support vectors $0<\alpha_i<\lambda$ (implies $\xi_i=0$) for numerical stability.

alex@seewald.at / alex.seewald.at

Example: One Step of SMO

(from 2nd Lecture, Slide 11: coded nominal attributes as 1-of-n; normalized temp & hum. to mean=0 and StD=1 by (t-19.5)/8.27 rand (h-73.8)/16.52)

TD=	overcast	rainy	sunny	Temp.	Hum.	W.=t	W.=f	y
$\mathbf{x}_1$	1	0	0	1.391	0.981	0	1	y_1=+1
$\mathbf{x}_2$	0	1	0	-0.786	-0.533	1	0	y_2=-1
$\mathbf{x}_3$	0	1	0	0.0605	0.678	1	0	y_3=-1
$\mathbf{x}_4$	0	0	1	-0.665	-1.138	0	1	y_4=+1

SMO w/ linear kernel: d=1, c=0, complexity parameter λ=1. Assume α=(0.2224,0.1813,0.3275,0.2864) and chose α_2 and α_4 for one step of SMO.

$$f(\mathbf{x}) = 0.2224K(\mathbf{x},\mathbf{x}_1) - 0.1813K(\mathbf{x},\mathbf{x}_2) - 0.3275K(\mathbf{x},\mathbf{x}_3) + 0.2864K(\mathbf{x},\mathbf{x}_4) + \beta_0$$

$$\alpha_4^{new,unc} = 0.2864 + \frac{(+1)\big((-1.0832+\beta_0-(-1))-(0.900+\beta_0-(+1))\big)}{2.902+3.737-2*(1.129)} = 0.2864 + \frac{0.0168}{4.381} = 0.2902$$

$$y_2 \neq y_4 \Rightarrow U = \max(0, 0.2864-0.1813) = 0.1051,\ V = \min(1, 0.2864-0.1813+1) = 1$$

$$U \leq \alpha_j^{new,unc} \leq V \Rightarrow a_4^{new} = a_4^{new,unc}$$

$$\alpha_2^{new} = 0.1813 + (-1)(+1)(0.2864-0.2902) = 0.1851 \Rightarrow \text{new } \alpha = (0.2224, 0.1851, 0.3275, 0.2902)$$

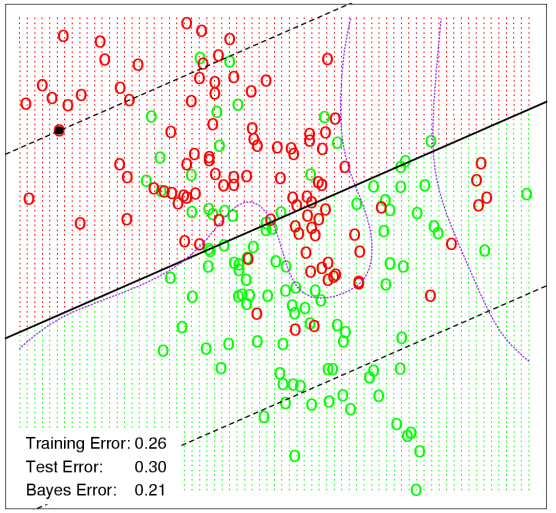
alex@seewald.at / alex.seewald.at

Complexity parameter λ

Influence of λ on SVM performance

Linear kernel ($\phi(x)=x$, $d=1$), $\lambda=0.01$

- Focusses more on data which is further away from the maximum margin hyperplane. Bigger margin reflects this behaviour. Error = 30.0%

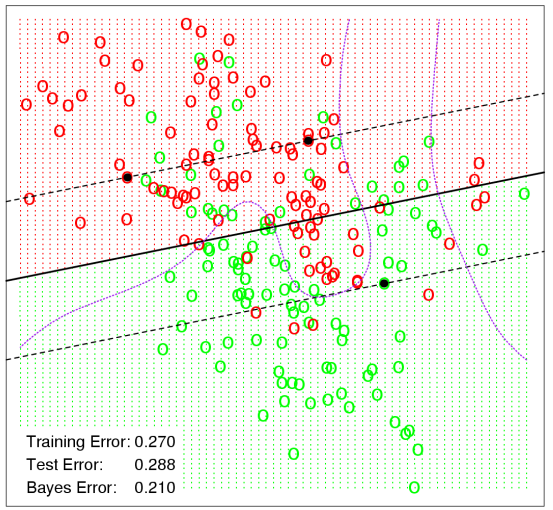


Linear kernel ($\phi(x)=x$, $d=1$), $\lambda=10000$

- Focusses more on data which is nearer to the maximum margin hyperplane. Smaller margin reflects this behaviour. Error = 28.8%

In both cases, all examples which are on the wrong side of the margin are given weight depending on their distance from the margin.

As we can see, the example is not well separable with just a linear kernel. Can we do better?



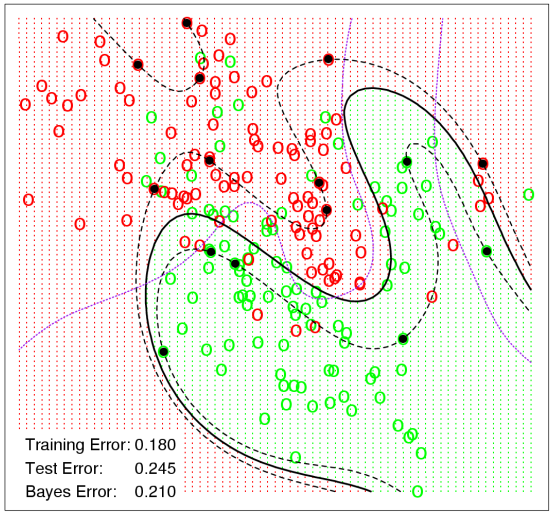
alex@seewald.at / alex.seewald.at

Non-linear Kernel Functions

Influence of Kernel on SVM performance

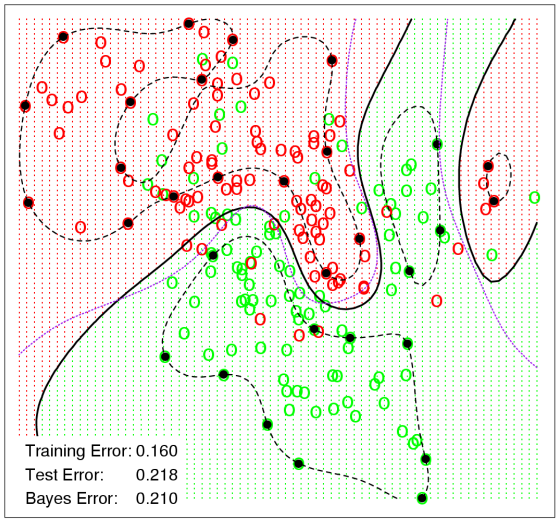
polynomial kernel of degree d=4, λ=1

- Non-linear decision boundary, small margin; slightly better generalization performance but tends to overshoot at the boundaries (a common problem of polynomials): Error = 24.5%



Gaussian (RBF) *kernel*, c=0.01, λ=1

- Non-linear decision boundary, larger margin; but almost optimal generalization performance. Error = 21.8% vs. optimal Bayes Error = 21.0%. This is probably due to the synthetic dataset which was generated by a mixture of Gaussian distributions.



alex@seewald.at / alex.seewald.at

Linear Methods' Loss functions: Overview

Logistic regression, linear regression and SVMs all basically use the same model $f(\mathbf{x}) = \text{sign}(\mathbf{x}^T.\beta+\beta_0)$. However, for each of them the weights (β,β_0) are estimated by minimizing different loss functions.

Loss function	L(y,f(**x**))	Minimizing function
(-)Binomial Log-Likelihood [*Logistic Regression*]	log(1+exp(-y*f(**x**))	$f(\mathbf{x}) = \log \frac{P(y=+1 \mid \mathbf{x})}{P(y=-1 \mid \mathbf{x})}$
(Residual) Squared Error [*Linear Regression*]	$(y-f(\mathbf{x}))^2$	$f(\mathbf{x}) = P(y=+1 \mid \mathbf{x}) - P(y=-1 \mid \mathbf{x})$
SVM Loss [*Support Vector Machine*]	max(0,1-y*f(**x**))	$f(\mathbf{x}) = \begin{cases} +1, & \text{if } P(y=+1 \mid \mathbf{x}) > \frac{1}{2} \\ -1, & \text{otherwise} \end{cases}$

More complex linear models include cubic splines, wavelets and nonparametric logistic regression (where log(probability) may be an arbitrary function), and additive mixtures of multiple linear models with far more complex loss functions.

alex@seewald.at / alex.seewald.at