Lecture 12

Support Vector Machines

ISLR 9

Jilles Vreeken
Aleksandar Bojchevski

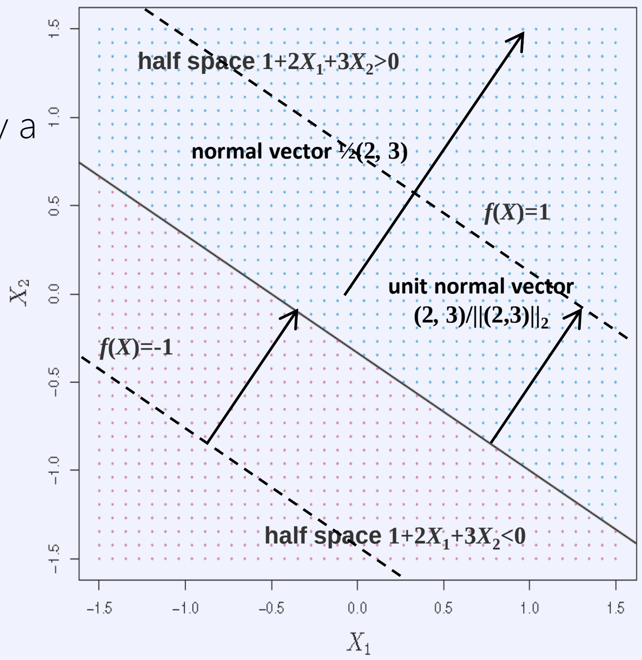
UNIVERSITÄT DES SAARLANDES

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Hyperplanes

- in p-dimensional vector space, a linear hyperplane is a $(p-1)$-dimensional subspace
- equivalently, a linear hyperplane is the set of points that satisfy a linear equation of the form $\beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$
- an affine hyperplane is the set of points that fulfills $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$ for some $\beta_0 \neq 0$
- a hyperplane divides the vector space into two half spaces
- the vector $(\beta_1, \dots, \beta_p)$ is the **normal** vector of the hyperplane

the hyperplane $f(X) = 1 + 2X_1 + 3X_2 = 0$



Classification Using Separating Hyperplanes

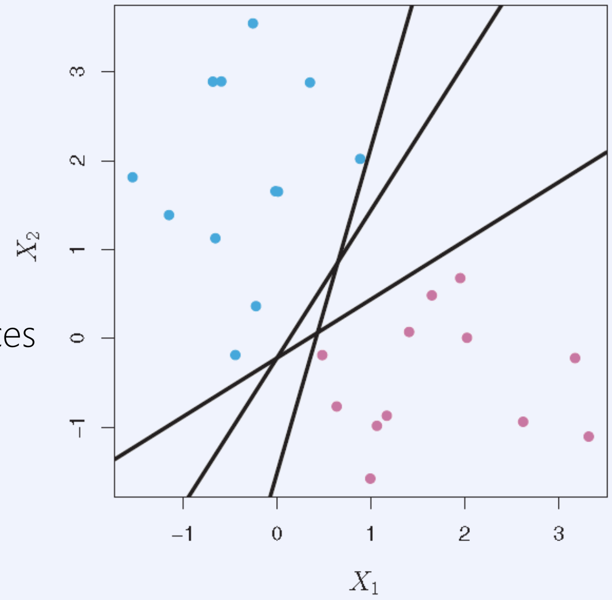
- assume a data matrix $x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$

  for a binary classification problem with classes $\{1, -1\}$
- assume further a test vector $x^* = \left(x_1^*, \ldots, x_p^*\right)^T$

We define a classifier based on a **separating hyperplane**

- the data points of the two classes locate in separate half spaces

different separating hyperplanes

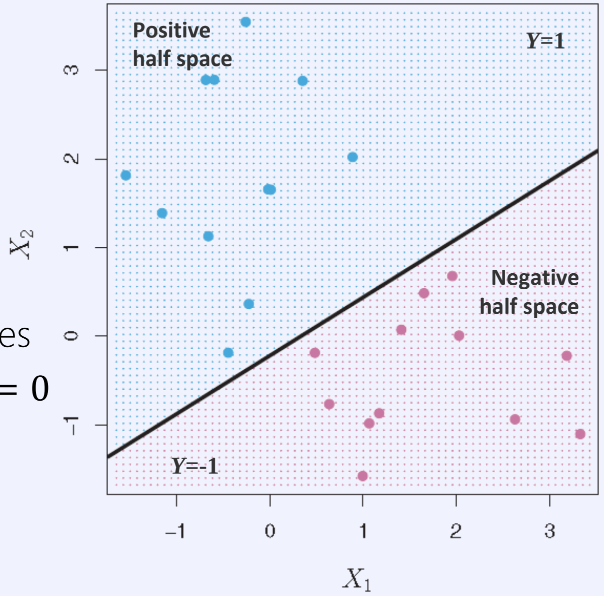


Classification Using Separating Hyperplanes

- assume a data matrix $x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$

  for a binary classification problem with classes $\{1, -1\}$
- assume further a test vector $x^* = (x_1^*, \dots, x_p^*)^T$

separating hyperplane and resulting classifier



We define a classifier based on a **separating hyperplane**

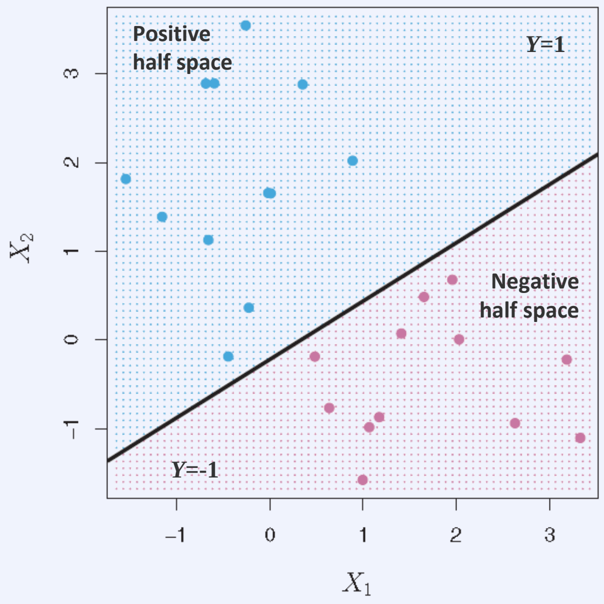
- the data points of the two classes locate in separate half spaces
- the hyperplane is defined by $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$
- the classification is $\mathbf{sign}(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p)$

- the distance of a point from the hyperplane is informative about the confidence in the classification

The Maximal Margin Classifier

- a hyperplane that maximizes the **distance of the closest point** in the training set to it can be considered optimal

separating hyperplane and resulting classifier



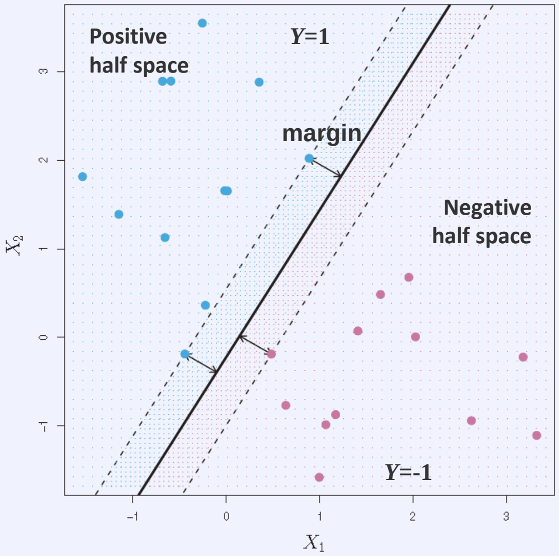
The Maximal Margin Classifier

- a hyperplane that maximizes the **distance of the closest point** in the training set to it can be considered optimal
- this distance is called the **margin**

The closest data points are called the **support vectors**

- only they determine the hyperplane
- can be a small subset of all points

separating hyperplane and resulting classifier



(ISLR 9.1.3)

Constructing the Maximal Margin Classifier

The optimization problem is

$$\max_{\beta_0,\beta_1,\ldots\beta_p,M} M$$

normal vector is unit vector

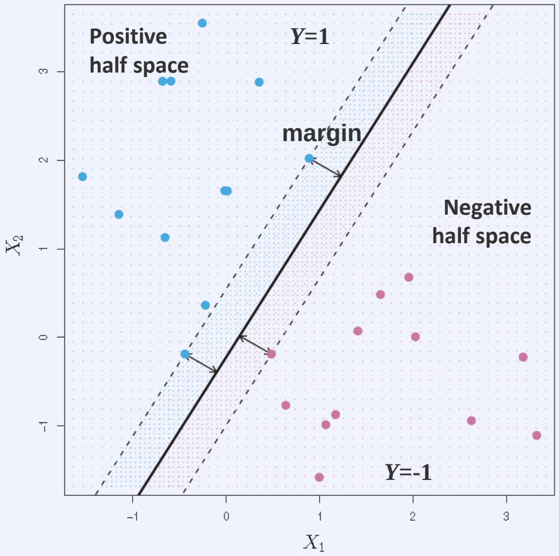
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

correct classification, if $M>0$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M, \;\; i = 1, \ldots, n$$

- since the normal is a unit vector, the distance of point i from the hyperplane is given by $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip})$
- solve the optimization problem with convex optimiz. techniques
- often, there is no separating hyperplane

separating hyperplane and resulting classifier



Constructing the Maximal Margin Classifier

The optimization problem is

$$\max_{\beta_0,\beta_1,\ldots\beta_p,M} M$$

normal vector is unit vector

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

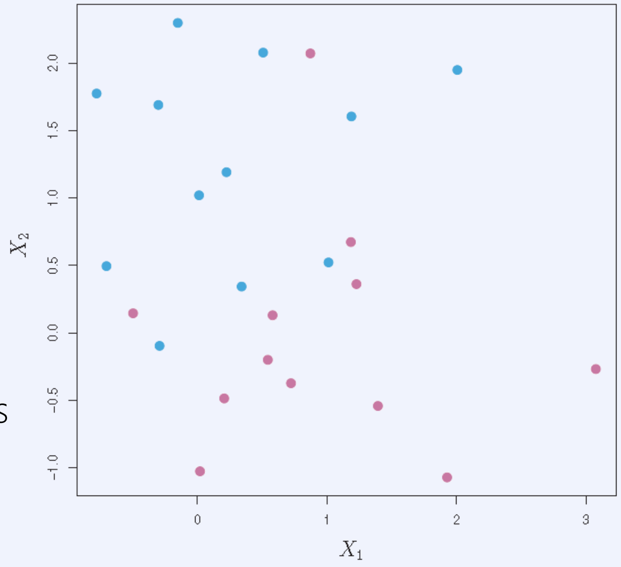
correct classification, if $M>0$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M,\ \ i = 1, \ldots, n$$

- since the normal is a unit vector, the distance of point $\boldsymbol{i}$ from the hyperplane is given by $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip})$
- solve the optimization problem with convex optimiz. techniques

- often, there is no separating hyperplane
- then we have to generalize to allow for misclassifications

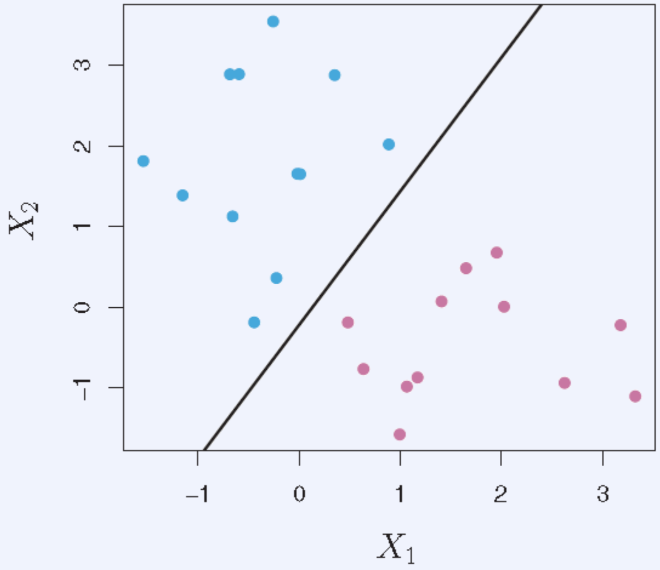
a non-separable dataset



The Support Vector Classifier

Even if the dataset is separable, a separating hyperplane may not be desirable

a nicely separable dataset



X

The Support Vector Classifier

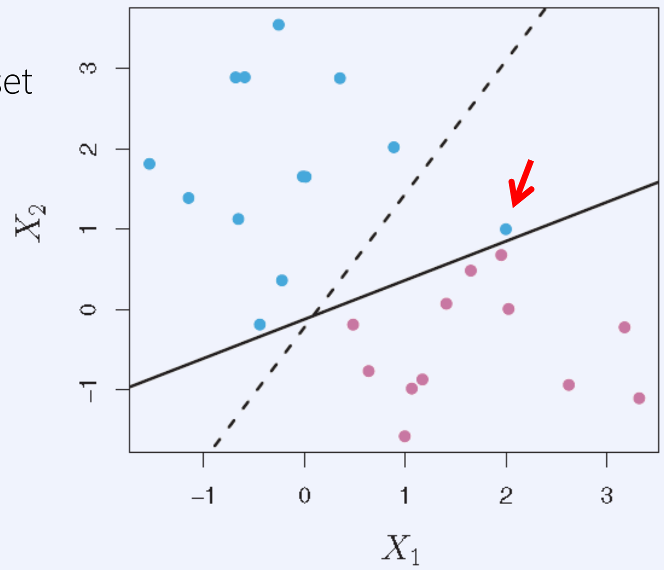
Even if the dataset is separable, a separating hyperplane may not be desirable

- adding a single data point leads to a hard-to-separate dataset
- the classifier is extremely sensitive to changes in the data

Sometimes it may be preferable to have a classifier that misplaces a few points in the training set but has a large margin to the other data points

- greater robustness w.r.t to small changes in the data
- better classification of most of the training points
- the soft-margin classifier does exactly this

a nicely separable dataset with an outlier



The Support Vector Classifier

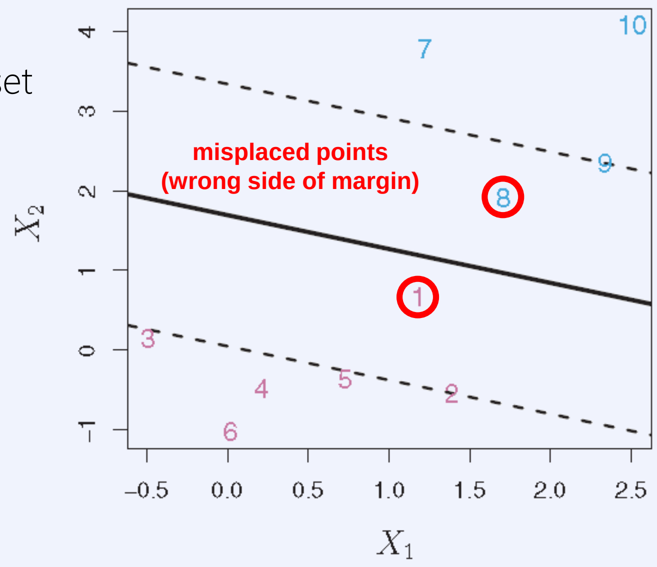
Even if the dataset is separable, a separating hyperplane may not be desirable

- adding a single data point leads to a hard-to-separate dataset
- the classifier is extremely sensitive to changes in the data

Sometimes it may be preferable to have a classifier that misplaces a few points in the training set but has a large margin to the other data points

- greater robustness w.r.t to small changes in the data
- better classification of most of the training points
- the soft-margin classifier does exactly this

soft-margin classifier



The Support Vector Classifier

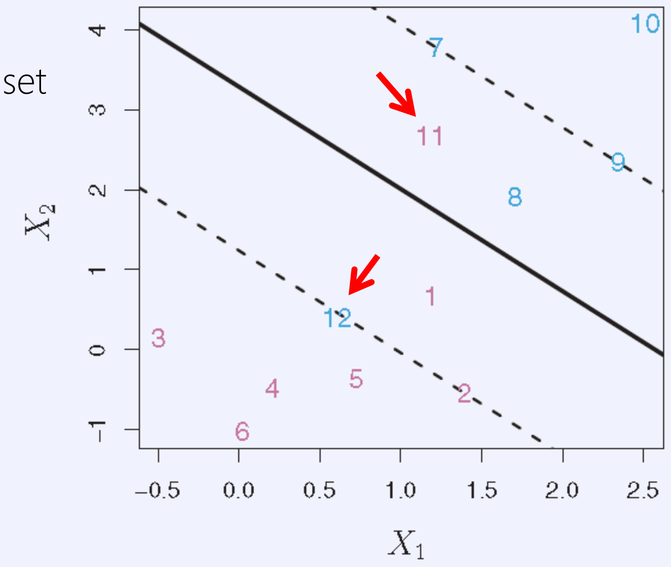
Even if the dataset is separable, a separating hyperplane may not be desirable

- adding a single data point leads to a hard-to-separate dataset
- the classifier is extremely sensitive to changes in the data

Sometimes it may be preferable to have a classifier that misplaces a few points in the training set but has a large margin to the other data points

- greater robustness w.r.t to small changes in the data
- better classification of most of the training points
- the soft-margin classifier does exactly this

soft-margin classifier



The Support Vector Classifier

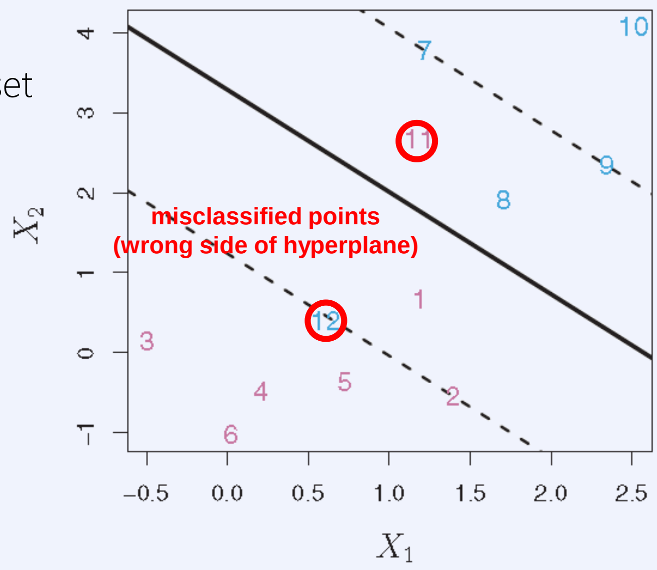
Even if the dataset is separable, a separating hyperplane may not be desirable

- adding a single data point leads to a hard-to-separate dataset
- the classifier is extremely sensitive to changes in the data

Sometimes it may be preferable to have a classifier that misplaces a few points in the training set but has a large margin to the other data points

- greater robustness w.r.t to small changes in the data
- better classification of most of the training points
- the soft-margin classifier does exactly this

soft-margin classifier



- points can be on the wrong side of the margin (misplaced but correct) or the hyperplane (misclassified)

Details of Soft-Margin Support Vector Classifier

The optimization problem is now

$$\max_{\beta_0,\beta_1,\ldots\beta_p,M} M$$

subject to $\sum_{j=1}^{p} \beta_j^2 = 1$

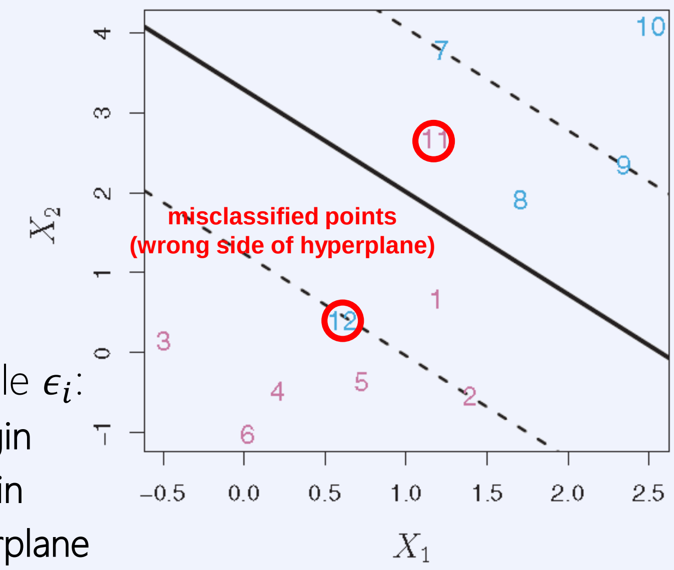
$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$

$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$

Slack variables allow for a fractional violation of the hard margin constraint

Budget for total admissible misclassification

soft-margin classifier



The following holds if we also choose the smallest possible ϵ_i:

- $\epsilon_i = 0 \Rightarrow$ the observation is on the **correct** side of the **margin**
- $\epsilon_i > 0 \Rightarrow$ the observation is on the **wrong** side of the **margin**
- $\epsilon_i > 1 \Rightarrow$ the observation is on the **wrong** side of the **hyperplane**
- Furthermore: no more than C observations can be on the wrong side of the hyperplane

On the Effect of the Budget $\boldsymbol{C}$

The optimization problem is now

$$\max_{\beta_0,\beta_1,\ldots\beta_p,M} M$$

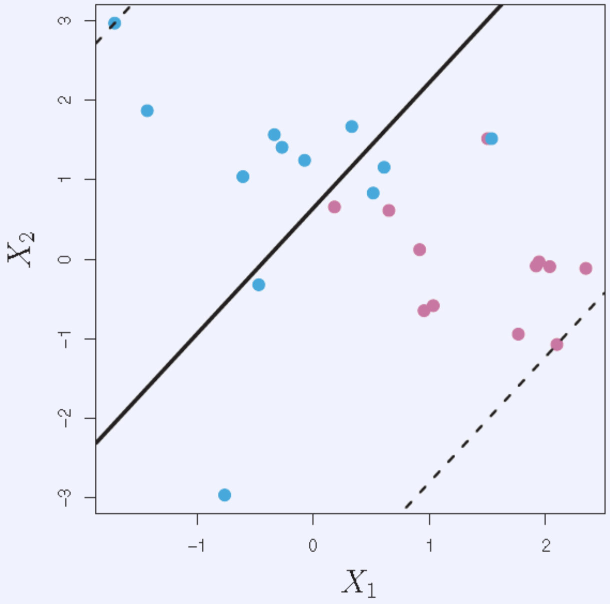
subject to $\sum_{j=1}^{p} \beta_j^2 = 1$

$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$

$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$

As $\boldsymbol{C}$ increases, we become more tolerant to violations

large C



X

On the Effect of the Budget $\boldsymbol{C}$

The optimization problem is now

$$\max_{\beta_0,\beta_1,\ldots\beta_p,M} M$$

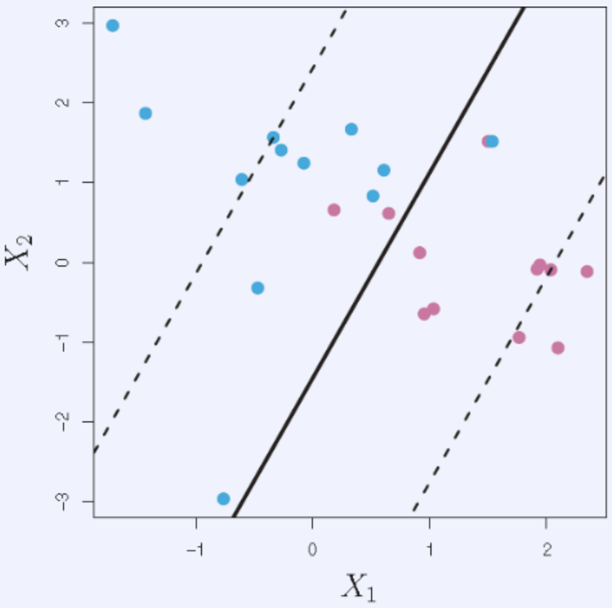
subject to $\sum_{j=1}^{p} \beta_j^2 = 1$

$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$

$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$

As $\boldsymbol{C}$ increases, we become more tolerant to violations

smaller C



(ISLR 9.2.2)

On the Effect of the Budget C

The optimization problem is now

$$\max_{\beta_0,\beta_1,\dots\beta_p,M} M$$

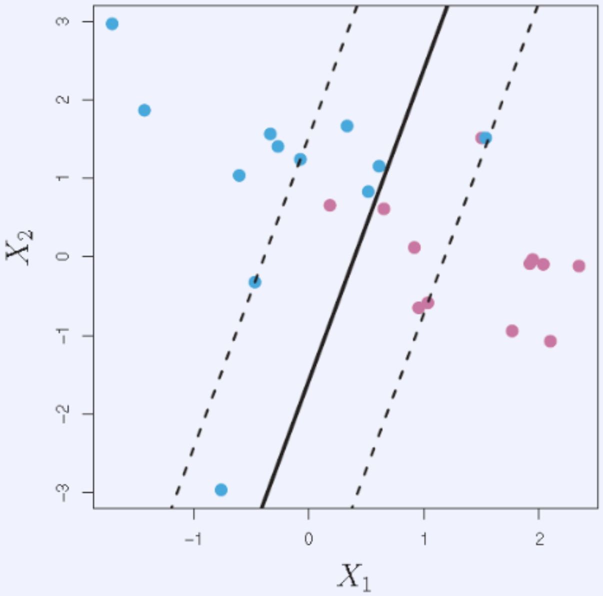
subject to $\sum_{j=1}^{p} \beta_j^2 = 1$

$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M(1 - \epsilon_i)$

$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$

As C increases, we become more tolerant to violations

even smaller C



On the Effect of the Budget C

The optimization problem is now

$$\max_{\beta_0,\beta_1,\ldots\beta_p,M} M$$

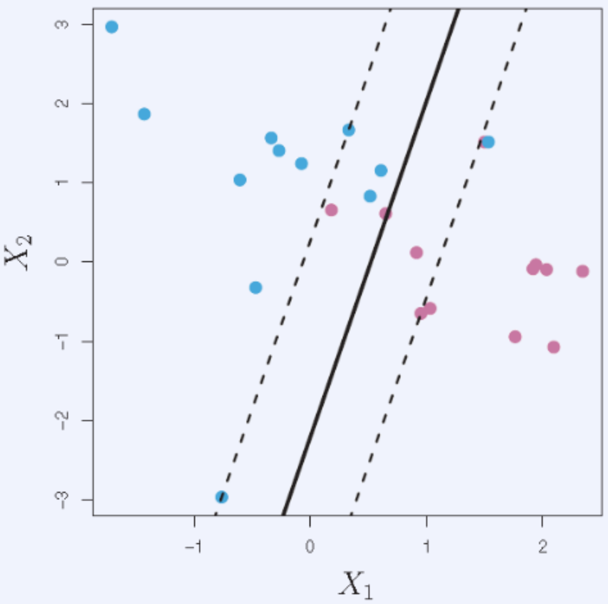
subject to $\sum_{j=1}^{p} \beta_j^2 = 1$

$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$

$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$

As C increases, we become more tolerant to violations

very small C



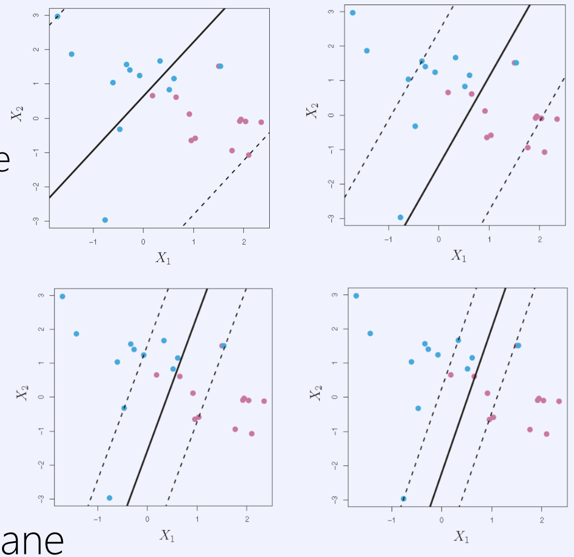
The Margin and the Support Vectors

We choose C via cross-validation

For the soft-margin classifier support vectors all either lie exactly on the margin or on the wrong side of the margin

- intuition: since only changing those points would affect the hyperplane
- C controls the bias-variance tradeoff
- with large C the margin is wide and there are many support vectors
  - low variance and potentially high bias
- with small C the margin is thin and there are a few support vectors
  - high variance and small bias

as C decreases we become less tolerant to violations



The fact that correctly classified points far away from the hyperplane do not affect the classifier is a property of the support-vector classifier

Nonlinear Decision Boundaries

Sometimes, data is inherently nonlinear

- then there is no soft margin that will do the trick
- we need a nonlinear version of support vector machines
- we could add nonlinear features to the feature space, e.g. $X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2$ instead of $X_1, X_2, \dots, X_p$
- the resulting optimization program would become

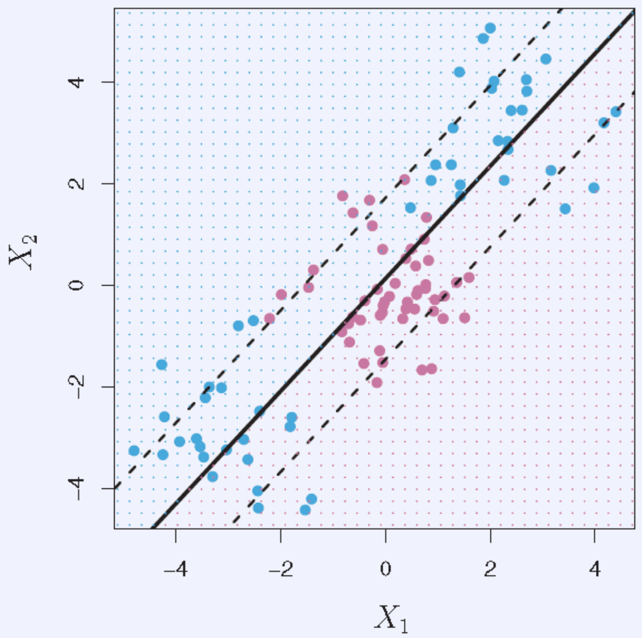
$$\max_{\beta_0,\beta_{11},\beta_{12},\dots\beta_{p1},\beta_{p2},\epsilon_1,\dots,\epsilon_n,M} M$$

$$\text{subject to } \epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C, \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1$$

$$y_i \left(\beta_0 + \sum_j^p \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i), \; i = 1, \dots n$$

- we could add higher-order, interaction terms, or use functions other than polynomials

the true boundary is non-linear



The Kernel Trick

With **support vectors machines (SVMs)** there is a different very elegant trick – the **kernel trick**

- builds on the optimization procedure for SVMs, which we will not detail
- it suffices to say that the linear support vector classifier can be rewritten as $f(x^*) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x^*, x_i \rangle$
- $\langle x^*, x_i \rangle = \sum_{j=1}^{p} x_j^* x_{ij}$ is the inner product,
- and the α_i are parameters that result from the training

set of support vectors

Important: Only the α_i for the support vectors are nonzero $f(x^*) = \beta_0 + \sum_{i \in S} \alpha_i \langle x^*, x_i \rangle$

The Kernel Trick

Only the α_i for the support vectors are nonzero $f(x^*) = \beta_0 + \sum_{i \in S} \alpha_i \langle x^*, x_i \rangle$

- to calculate α_i and β_0 we only need $\frac{n(n-1)}{2}$ inner products $\langle x_i, x_{i'} \rangle$ between all pairs of training points
- the actual coordinates of the training observations or the test point are never needed!

We can generalize inner products to (nonlinear) kernels $K(x_i, x_{i'})$

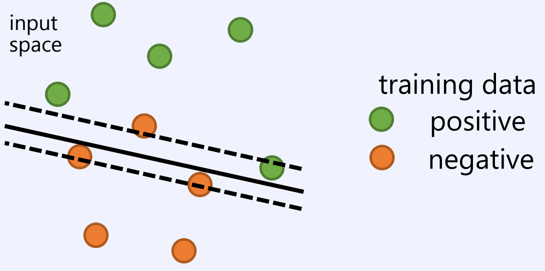
- a kernels quantifies the similarity between two data points
- a simple linear kernel is $K(x_i, x_{i'}) = \langle x_i, x_{i'} \rangle$
- it quantifies similarity in terms of the standard (Pearson) correlation

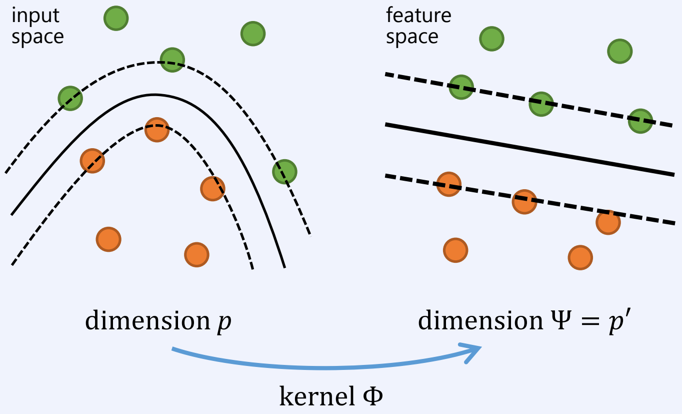
Nonlinear Kernels

Two popular choices:

- The polynomial kernel with degree d
  $K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} \langle x_{ij}, x_{i'j} \rangle\right)^d$
- The radial-basis kernel
  $K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2\right)$
- in general, a kernel is any symmetric and positive definite function[1] of its two arguments

A VERY important theorem says that for any kernel K there is a function $\Phi\colon \mathbb{R}^p \to \Psi$ such that $K\left(x_i, x_j\right) = \left\langle \Phi(x_i), \Phi(x_j) \right\rangle$



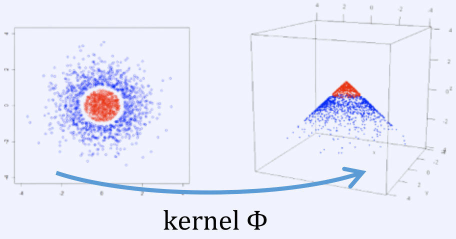


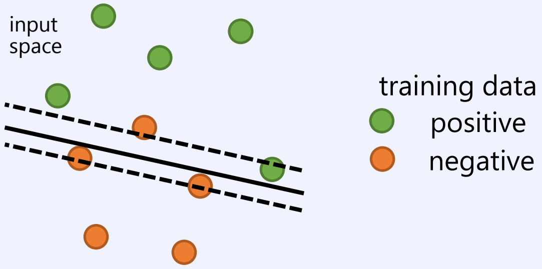
1) i.e. for any non-zero real vector $(c, \ldots, c_N)$ we have $\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j K(X_i, X_j) > 0$ (ISLR 9.3.2)

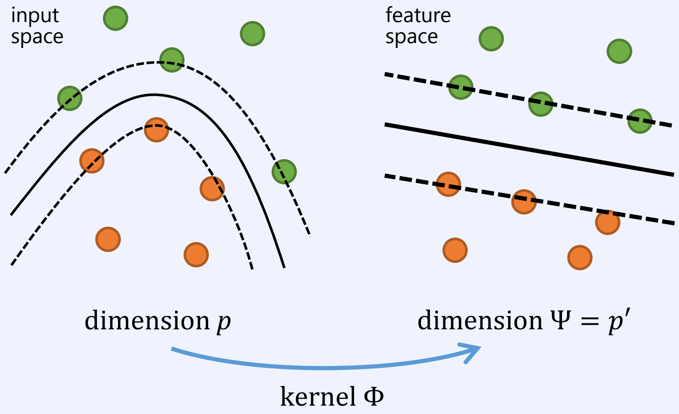
Reproducing Kernel Hilbert Space (RKHS)

Applying the kernel actually means performing an inner product in some space Ψ, the so-called [RKHS]

- neither Φ nor Ψ generally can (or need) be constructed in a computationally usable form
- however in some cases, they can, e.g., for $p = 2$ and the polynomial kernel with $d = 2$, we have $\dim \Psi = p' = 6$ and
  $\Phi_1(X) = 1, \quad \Phi_2(X) = \sqrt{2}X_1, \quad \Phi_3(X) = \sqrt{2}X_2$
  $\Phi_4(X) = X_1^2, \quad \Phi_5(X) = X_2^2, \quad \Phi_6(X) = \sqrt{2}X_1X_2$
- for the radial basis kernel, p' is infinite[1]







1) Steinwart et al (2006)

(ISLR 9.3.2)

The Radial Basis Kernel

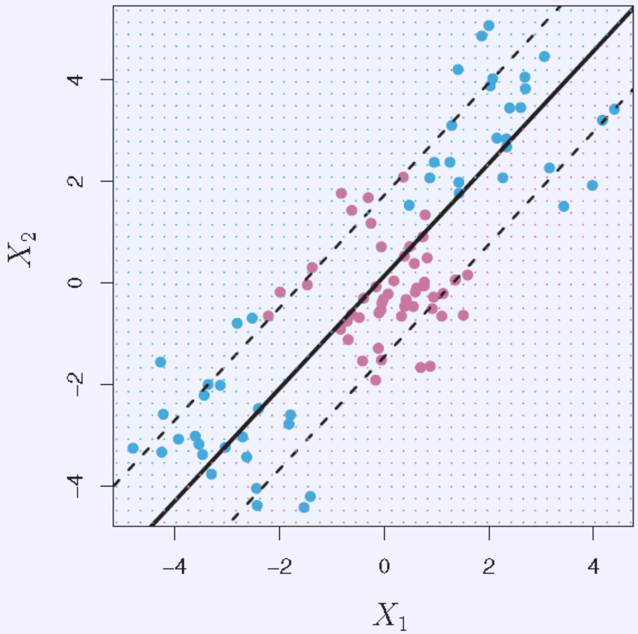
If our test point x^* is far from the training point x_i then $\sum_{j=1}^{p}\left(x_j^* - x_{ij}\right)^2$ will be large, so the kernel value $\exp\left(-\gamma \sum_{j=1}^{p}\left(x_j^* - x_{ij}\right)^2\right)$ will be tiny

- thus x_i will not influence the value of $f(x^*)$ by much

Since the class label is based on the sign of $f(x^*)$ the radial basis kernel thus has very local behavior

- γ controls the locality
- decreasing γ increases locality

linear kernel



The Radial Basis Kernel

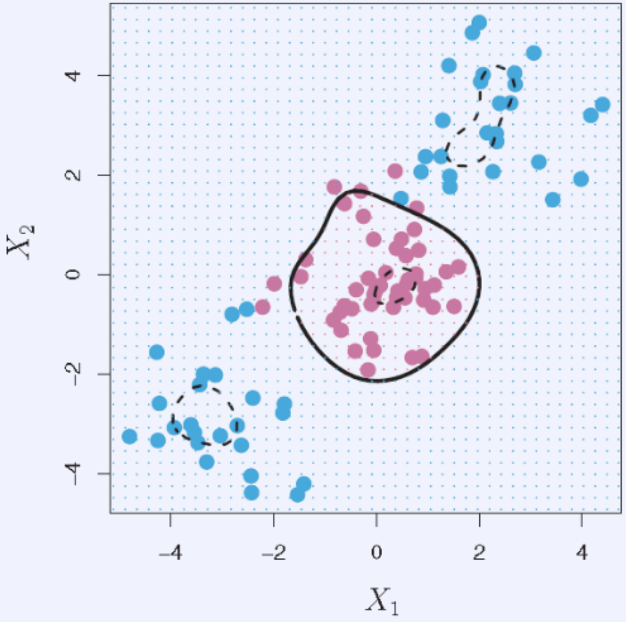
If our test point x^* is far from the training point x_i then $\sum_{j=1}^{p}\left(x_j^* - x_{ij}\right)^2$ will be large, so the kernel value $\exp\left(-\gamma \sum_{j=1}^{p}\left(x_j^* - x_{ij}\right)^2\right)$ will be tiny

- thus x_i will not influence the value of $f(x^*)$ by much

Since the class label is based on the sign of $f(x^*)$ the radial basis kernel thus has very local behavior

- γ controls the locality
- decreasing γ increases locality

radial basis kernel



Advantages of Kernels

To calculate the SVM you only need the kernel matrix for the pairs of training points

- in contrast, enlarging the feature space is computationally expensive

Can be applied to arbitrary observations that are not vectors: graphs, strings, molecules, etc.

The kernel trick can also be used with other statistical learning methods such as LDA or PCA

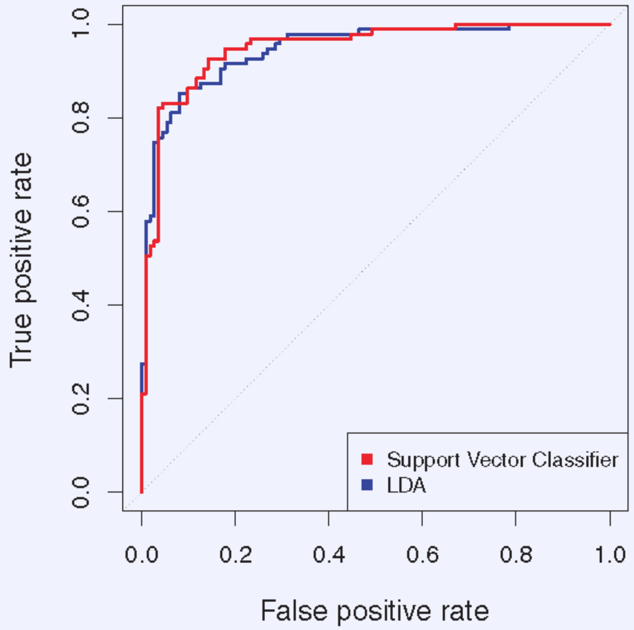
Application to the Heart Disease Data

- 13 predictors are used for classification
- binary target: whether an individual has heart disease
- 207 training, 90 test observations

Comparison of LDA and linear SVM

- use a threshold on $f(x)$ to parameterize SVM
- use a threshold on the linear discriminant to parameterize LDA
- similar performance on the training data

ROC curve for classification performance on the Heart dataset – training data



X

Application to the Heart Disease Data

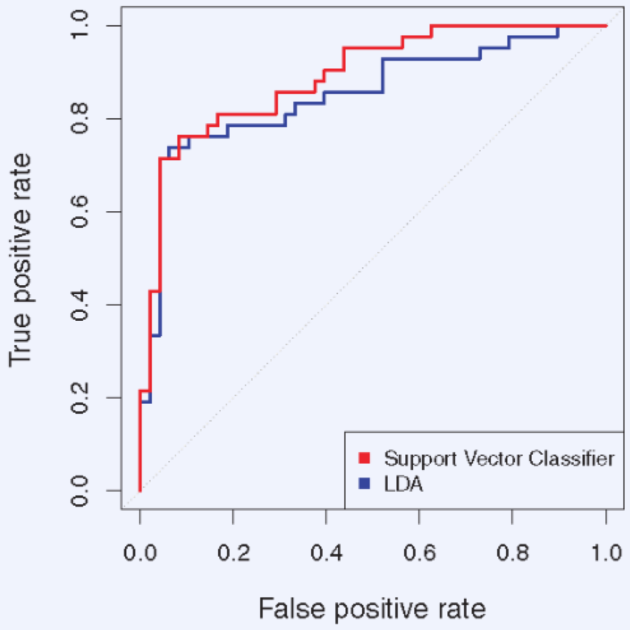
- 13 predictors are used for classification
- binary target: whether an individual has heart disease
- 207 training, 90 test observations

Comparison of LDA and linear SVM

- use a threshold on $f(x)$ to parameterize SVM
- use a threshold on the linear discriminant to parameterize LDA
- similar performance on the training data

- SVM outperforms LDA on the test set – generalizes better

ROC curve for classification performance on the Heart dataset – test data



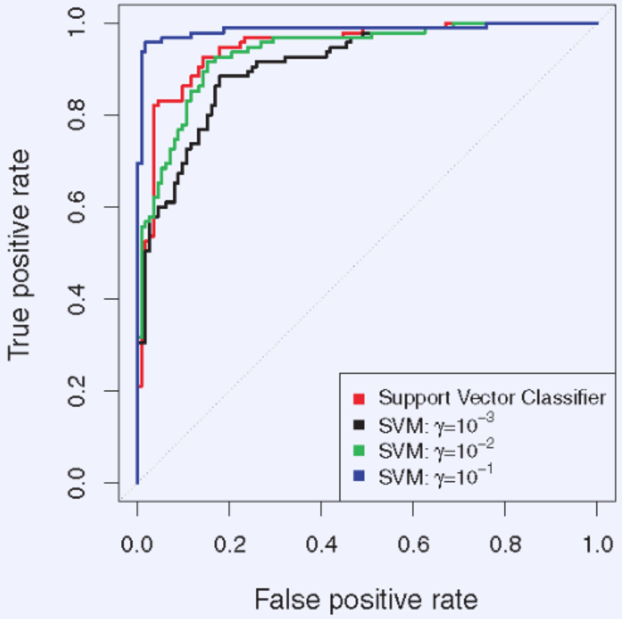
Application to the Heart Disease Data

- 13 predictors are used for classification
- binary target: whether an individual has heart disease
- 207 training, 90 test observations

Comparison of linear and nonlinear (radial basis kernel) support vector classifiers

- $\gamma = 10^{-1}$ is best on the training set

ROC curve for classification performance on the Heart dataset – training data



X

Application to the Heart Disease Data

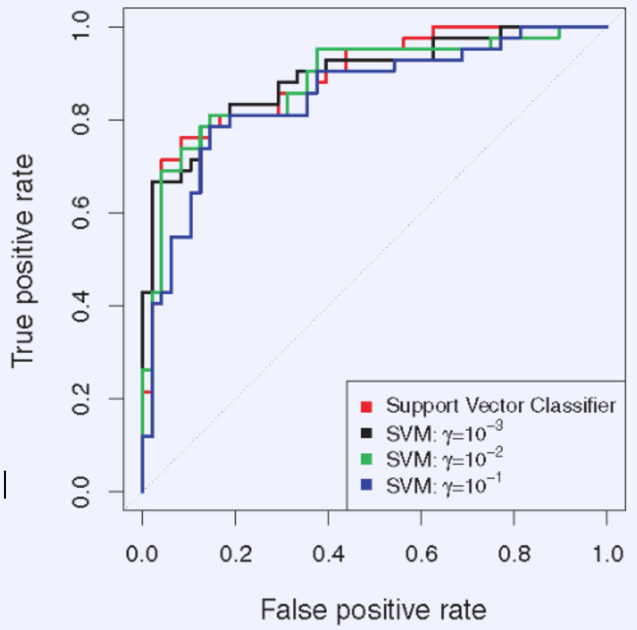
- 13 predictors are used for classification
- binary target: whether an individual has heart disease
- 207 training, 90 test observations

Comparison of linear and nonlinear (radial basis kernel) support vector classifiers

- $\gamma = 10^{-1}$ is best on the training set

- $\gamma = 10^{-1}$ is worst on the training set
- this amounts to a very local kernel which incurs high variance
- other nonlinear kernels perform comparably with the linear kernel

ROC curve for classification performance on the Heart dataset – test data

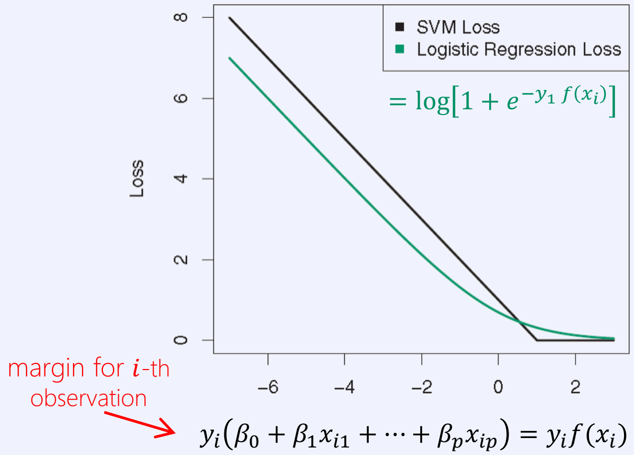


Relationship to Logistic Regression

The SVM optimization problem can be rewritten as

$$\min_{\beta_0,\beta_1,\dots,\beta_p} \left\{\sum_{i=1}^{n} \max[0, 1 - y_i f(x_i)]_+ + \lambda \sum_{j=1}^{p} \beta_j^2\right\}$$

- his has a general form of a regularized regression $\min_{\beta_0,\beta_1,\dots,\beta_p} \{L(\mathbf{X}, \mathbf{y}, \beta) + \lambda P(\beta)\}$ with loss L and penalty P



SVM uses the same penalty as in ridge regression, but a different loss function, called **hinge** loss

- similar to that used in logistic regression, thus both classifiers often give similar results
- with better separation, SVM is better, with more overlap logistic regression tends to be better

The budget C for margin violations is inversely proportional to the penalty parameter λ

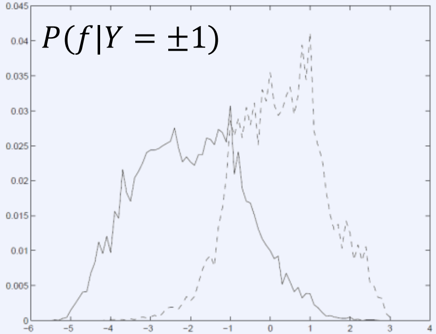
Posterior Probabilities from SVMs

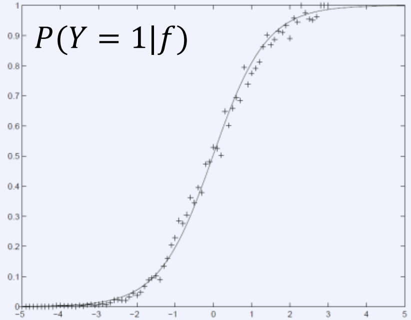
Turning SVM output into ROC curves

- compute posterior probability of the input belonging to class 0 and 1 respectively using the formula $P(y=1|x) = \frac{1}{1+\exp(Af(x)+B)}$ where $f(x)$ is the SVM output
- A and B are parameters that are trained discriminatively

The original distributions are not Gaussian and ragged

- but, logistic fit works well





(Platt, J. C. (1999). In Advances in Large Margin Classifiers. A. J. Smola et al., eds., MIT Press: 61-74.)

Multiclass classification

Standard SVM cannot handle multiple classes. We show strategies to address the issue.

- they can be generally applied anytime a binary classifier is the only option

One-vs-rest: Train K SVM models for K classes, where each SVM is being trained for classification of one class against all the remaining ones.

- winner is then the class, where the distance from the hyperplane is maximal

One-vs-one: train $\binom{K}{2}$ classifiers (all possible pairings) and evaluate all

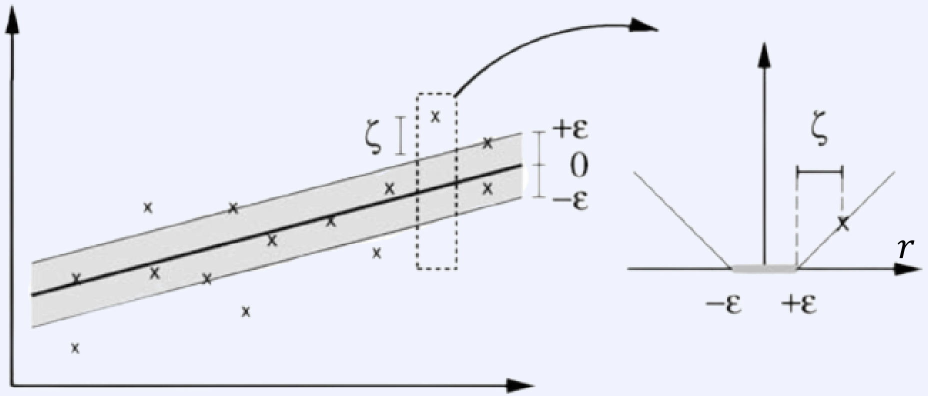
- winner is the class with the majority vote
- votes can be weighted according to the distance from the margin

One-class SVM: an unsupervised algorithm to learn a decision function for novelty detection

Support Vector Machines for Regression

Want to fit a linear model $f(x) = x^T\beta + \beta_0$ such that all data points lie inside a margin of width ϵ of the regression hyperplane

- impose a square penalty on model complexity



The loss function is the ϵ-insensitive $V_\epsilon(r)$

- only data points **on or outside** the tube change the model (this is different from classification)
- these are the support vectors
- kernels distort the tube

(Smola and Schölkopf. A Tutorial on Support Vector Regression. 1998)

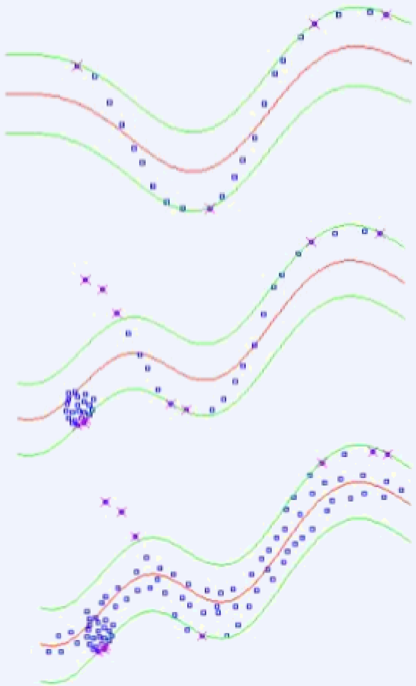
Example SVR with a Radial Basis Kernel

SVR with RBF kernel on synthetic data

- Green lines show the ϵ-boundaries
- Blue points represent data instance
- Marked blue points are the support vectors

The fitted model adapts well to the structure of the data

Introducing new datapoints change the model only if they are on the ϵ-boundary or outside of it



(Smola and Schölkopf. A Tutorial on Support Vector Regression. 1998)

Summary

The main ideas behind SVMs is to find the max-margin hyperplane that separate the data

Hard SVM requires that all training data is correctly separated by can overfit

Soft SVM allows violations of the margin up to a budget C to get a better hyperplane overall

We can rewrite the SVM classifier only in terms of inner products – replacing those with a kernel is the kernel trick which allow us to efficiently introduce non-linearity

- the kernel trick is an important general idea that also applies to LDA, PCA and other models

Linear SVM is similar to logistic ridge regression but uses a hinge loss instead