Support Vector Machines

Diana Mateus

with contributions from Sebastian Poelster and Olivier Pauly

MLMI ss2016

Technische Universität München

Johns Hopkins Whiting School of Engineering

Outline

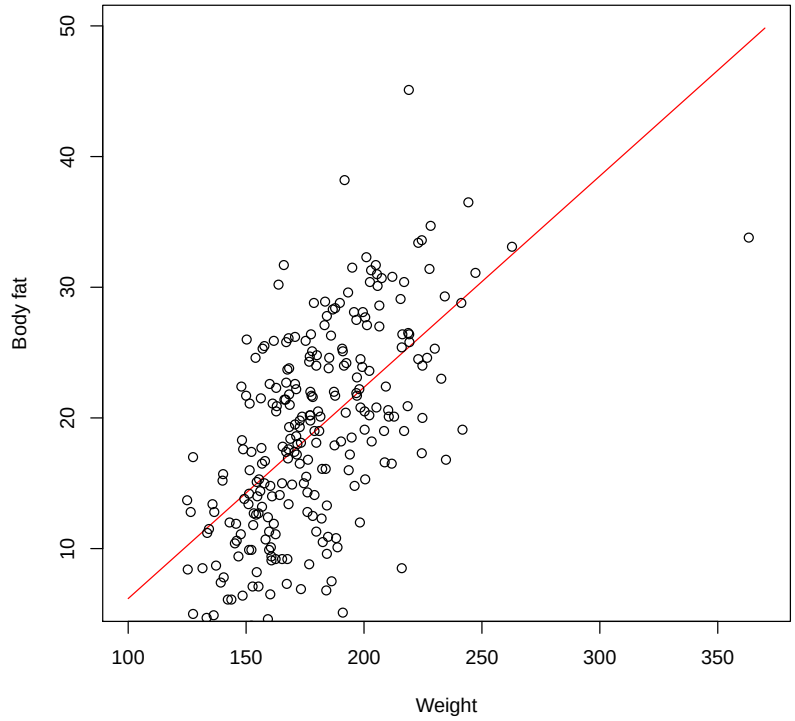
How to fit a line?

Data:

- $\mathbf{x}_i$ is a data point
- each $\mathbf{x}_i$ has m features.

$$\mathbf{x}_i = (x_{i1}, \ldots, x_{im})^\top$$

- there are n such data points
- Data matrix $\mathbf{X}$ (size $n \times m$)



How to fit a line?

Line equation:

- 1D

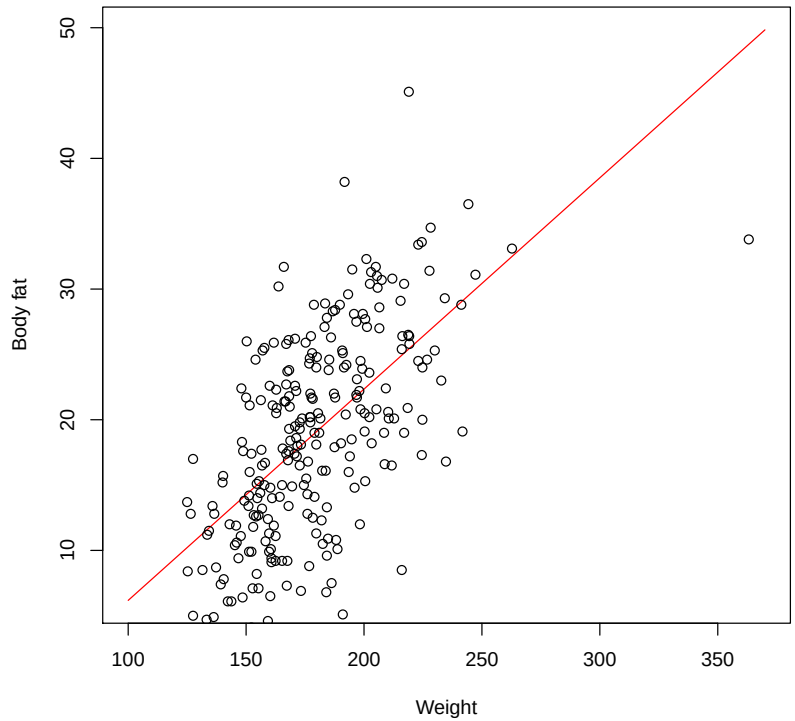
$$y_i = w_0 + w_1 x_{i1}$$

- 2D

$$y_i = w_0 + w_1 x_{i1} + w_2 x_{i1}$$

- 3D

$$y_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3}$$



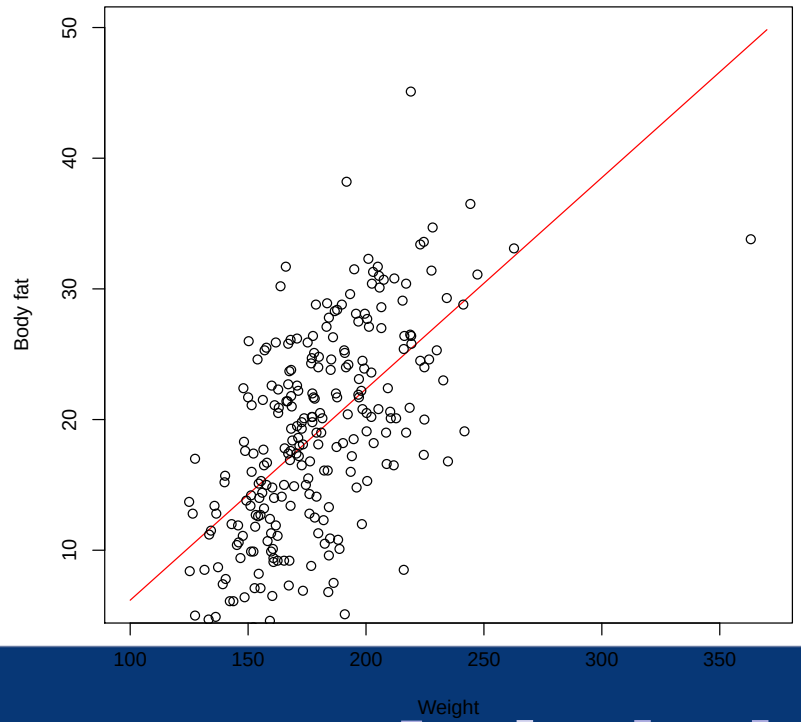
How to fit a line?

Definition (Linear Model)

$$y_i = w_0 + w_1 x_{i1} + \ldots + w_m x_{im}$$

$$y_i = w_0 + \mathbf{x}_i^\top \mathbf{w}$$

- w_i are **coefficients** or weights of each features.
- w_0 denotes the **intercept**.



How to fit a line?

Assumptions

- We have n data pairs input-output $\{\mathbf{x}_i, y_i\}$
- There is a function f that relates features to outputs:

$$\begin{aligned} y_1 &= f(\mathbf{x}_1) \\ y_2 &= f(\mathbf{x}_2) \\ &\vdots \\ y_i &= f(\mathbf{x}_i) \end{aligned}$$

and the function is linear:

$$f(\mathbf{x}_i) = w_0 + \mathbf{x}_i^\top \mathbf{w}$$
$$i \in \{1, \ldots, n\}$$

How to fit a line?

Goal: approximate f

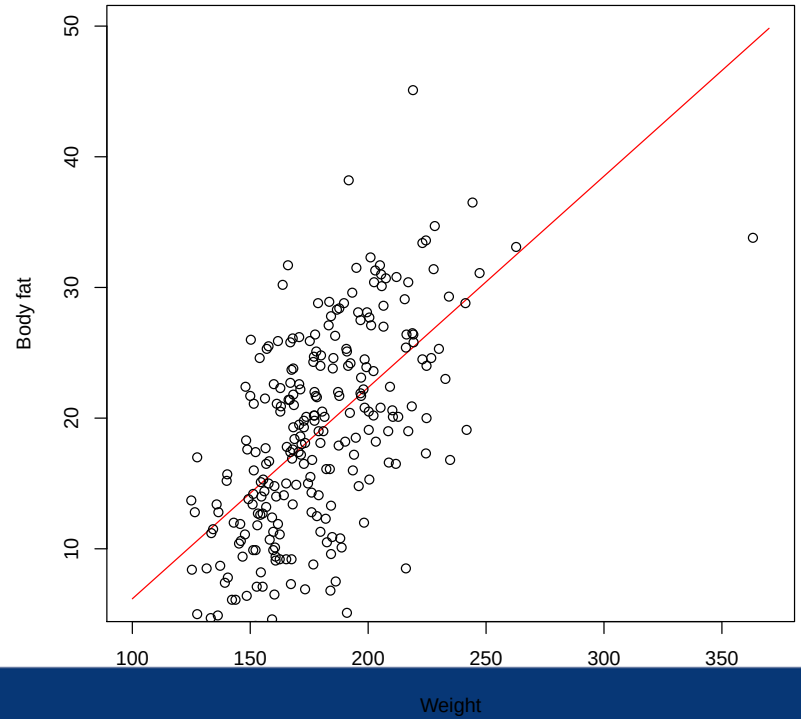
Estimate the model coefficients w from training data

$$\hat{f}(x_1, \ldots, x_m) = \hat{w}_0 + \hat{w}_1 x_1 + \ldots + \hat{w}_m x_m$$

- Estimates are denoted by a **hat**: $\hat{w}_i$
- In the example

  $w_0 = -9.995$ (y-intercept)

  $w_1 = 0.1617$ (slope; coefficient of *weight* feature).

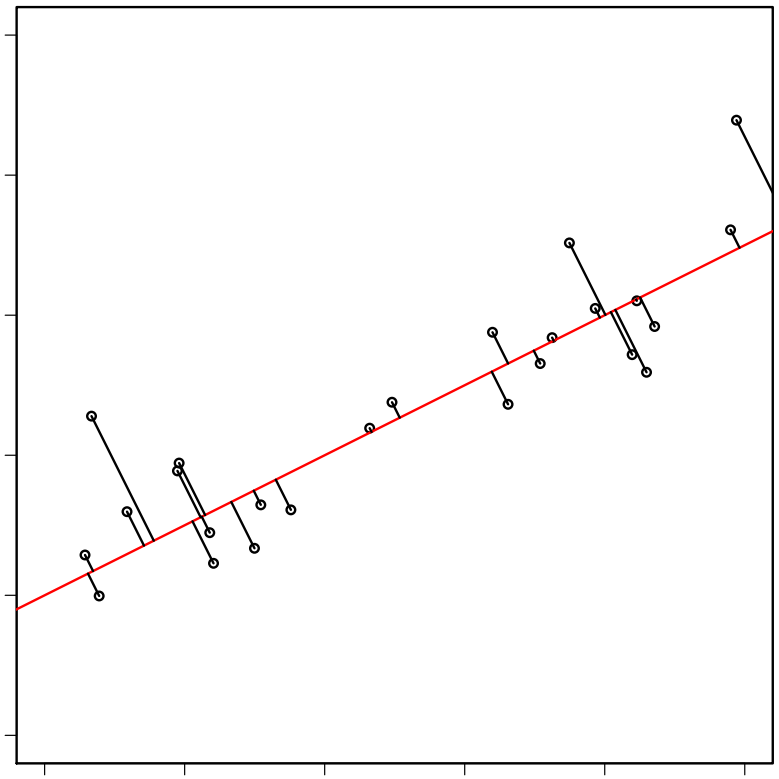


Linear Models – Loss Function

How good is our estimation of the model?

- Compare
  - predictions $\hat{y}_i = \hat{f}(\mathbf{x}_i)$
  - training data y_i
- Use a **loss function**

$$\text{Loss} = L\left(y_i\,,\,\hat{f}(\mathbf{x}_i)\right)$$



Linear Models – Ordinary Least Squares Estimation

Definition (Residual Sum of Squares; RSS)

$$\mathrm{RSS}(w_0, \ldots, w_m) = \sum_{i=1}^{n} (y_i - \hat{f}(\mathbf{x}_i))^2$$

- RSS gives the total loss over the whole training set
- Choose the coefficients $w_0, \ldots, w_m$ such that the total loss according to RSS is **minimized**.
- **How can this be achieved?**

Linear Models – Ordinary Least Squares Estimation

- Set the partial derivative of RSS to zero
- In matrix notation:

$$\mathrm{RSS}(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) \tag{1}$$

$$\frac{\partial \mathrm{RSS}(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\mathbf{w}). \tag{2}$$

- **Note**:here $\mathbf{w} = (w_0, \ldots, w_m)^\top$ and the first column of $\mathbf{X}$ contains only 1 to accommodate the intercept w_0, i.e. $\mathbf{X}$ is a $n \times m + 1$ matrix.

Linear Models – Ordinary Least Squares Estimation

Definition (Ordinary Least Squares Estimate)

$$\hat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

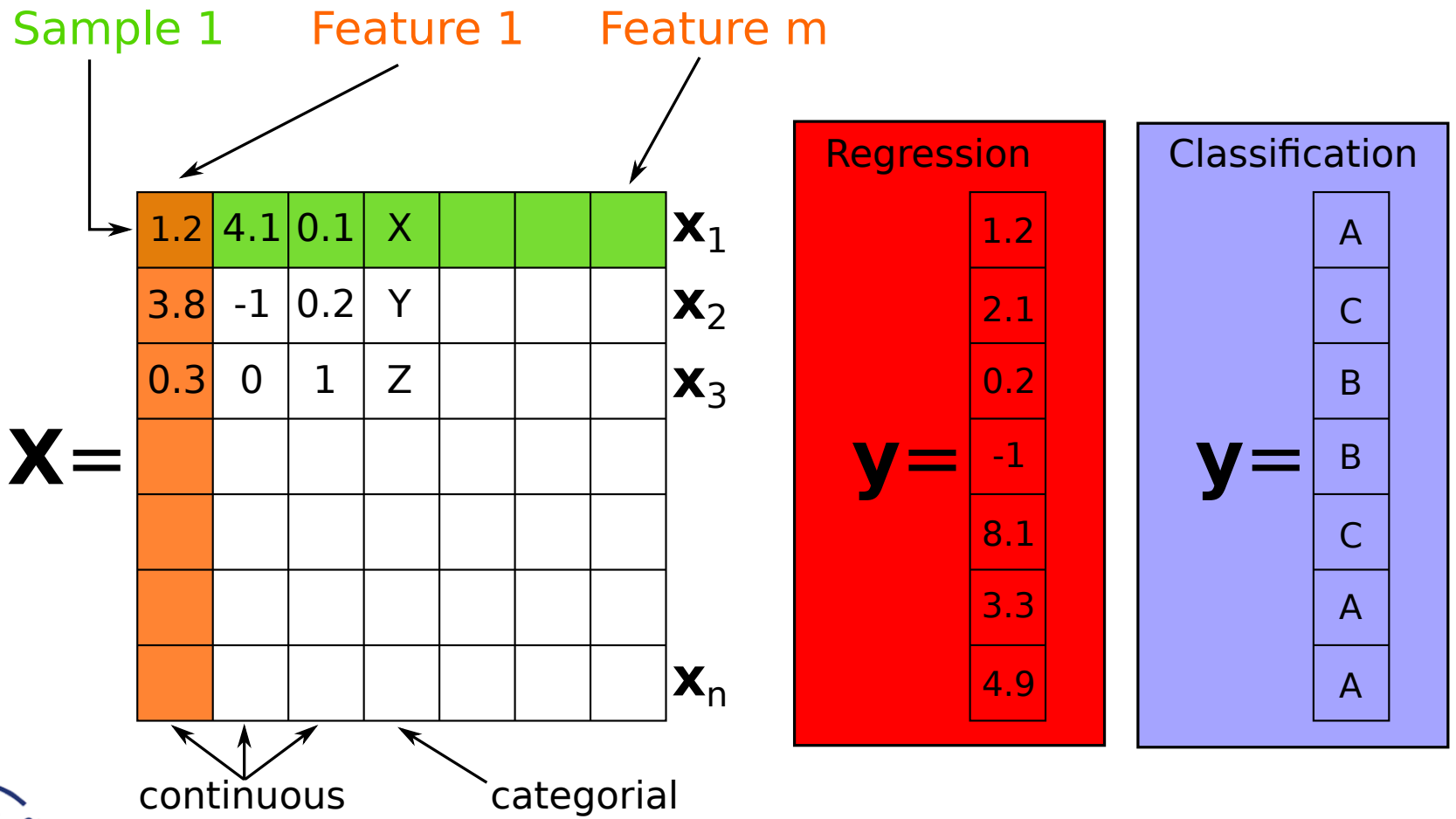
- The minimum of the loss function is **unique**.
- Estimates of the coefficients can be obtained in **closed form** solution and therefore no optimization is required.
- $\mathbf{X}$ must have full column rank $\Rightarrow \mathbf{X}^\top \mathbf{X}$ is positive definite.
- Prediction (regression) is performed by

$$\hat{f}(x_1, \ldots, x_m) = \hat{w}_0 + \hat{w}_1 x_1 + \ldots + \hat{w}_m x_m$$

Classification – Definitions

- **Training sample** $\mathbf{x}_i$ consists of m **features** $(x_{i1}, \ldots, x_{im})^\top$
- To each training sample $\mathbf{x}_i$ is associated a training **output** y_i.
- **Training set** $\mathcal{T} = \{\mathbf{x}_i, y_i\}_{i=1}^n$.
- Data matrix $\mathbf{X}$ with the i-th sample in the i-th row
- $\mathbf{y} = (y_1, \ldots, y_n)^\top$ the vector of all outputs.

Classification – Definitions



Classification – Definitions

- Each **feature** can be:
  - **continuous** (a number)
  - **discrete** (from a predefined set of values).

- The **output** can be:
  - continuous, we perform regression.
  - discrete, we perform classification.

Linear classification – Definitions

Classification Goal

Find a **Decision function** f so that: $f(\mathbf{x}_i) = y_i$

- Assume a **linear model** for f.
- We solved linear **regression** with least squares ...
- Problem: in **classification** the outcome y_i is **categorical**.

Binary classification problem

Input: Objects from 2 classes:  and 

Goal: Find a **decision function** f so that :

$$f(\mathbf{x}_i) = y_i,$$
$$\text{for all } i \in \{1, \cdots, n\}$$

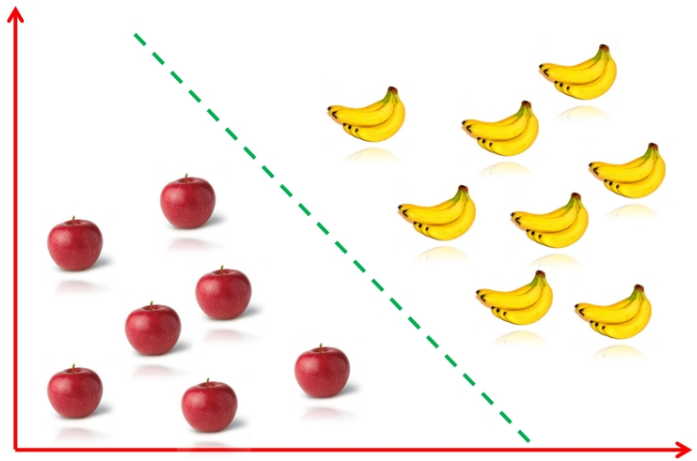
ex:

$f($  $) = \texttt{'apple'}$

Generalized Linear Classifier

- Given training data $\{\mathbf{x}_i,\ y_i\}_{i\in\{1,\cdots,n\}}$
- Find a **hyperplane** that separates the data points in 2 classes:

$$h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}_i + w_0$$



- The line of interest is created when the hyperplane cuts the feature space $(h = 0)$

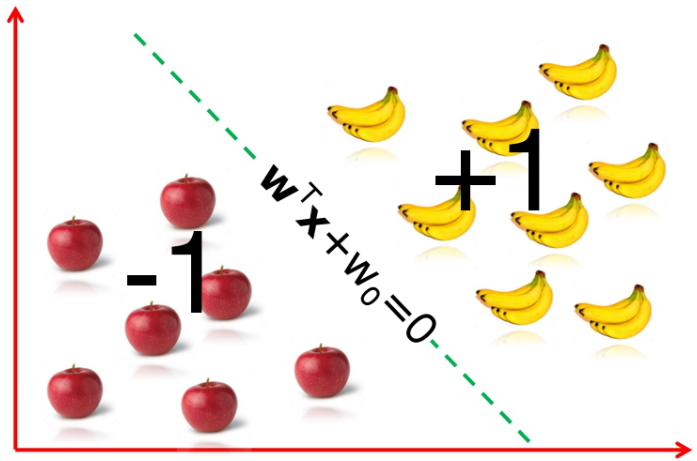
Generalized Linear Classifier

- Given training data $\{\mathbf{x}_i,\ y_i\}_{i\in\{1,\cdots,n\}}$
- Find hyperplane

$$h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

Subject to constraints for all i:

$$\mathbf{w}^\top \mathbf{x}_i + w_0 > 0 \text{ if } y_i = +1$$
$$\mathbf{w}^\top \mathbf{x}_i + w_0 < 0 \text{ if } y_i = -1$$



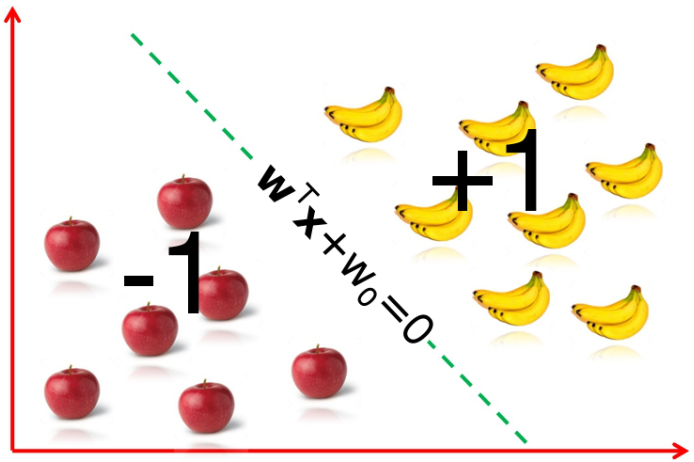
Generalized Linear Classifier

- Given training data $\{\mathbf{x}_i,\ y_i\}_{i\in\{1,\cdots,n\}}$
- Find hyperplane

$$h = \mathbf{w}^\top \mathbf{x} + w_0 = 0$$

Subject to constraints for all i:

$$\mathbf{w}^\top \mathbf{x}_i + w_0 > 0 \text{ if } y_i = +1$$
$$\mathbf{w}^\top \mathbf{x}_i + w_0 < 0 \text{ if } y_i = -1$$



Decision function $f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + w_0)$

Generalized Linear Classifier – Loss

GOAL: Find the optimal parameters $\hat{\mathbf{w}}$ and $\hat{w}_0$ which minimize the classification errors by using a loss function.

Empirical Risk Minimization

$$[\hat{\mathbf{w}}, \hat{w}_0] = \text{argmin}_{\mathbf{w},w_0} \left(\frac{1}{n} \sum_{i=1}^{n} \left(y_i \neq \text{sign}(\mathbf{w}^T \mathbf{x}_i + w_0) \right) \right)$$

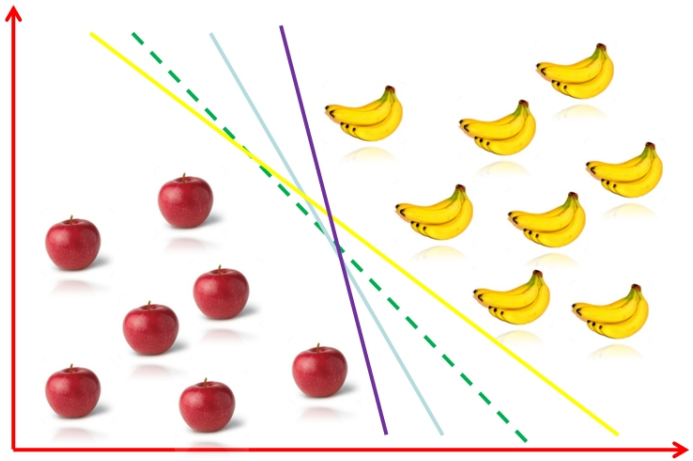
Can be solved if data are linearly separable, if not NP-hard!

Prediction $\hat{f}(\mathbf{x}) = \text{sign}(\hat{\mathbf{w}}^\top \mathbf{x} + \hat{w}_0)$

A lot of solutions!

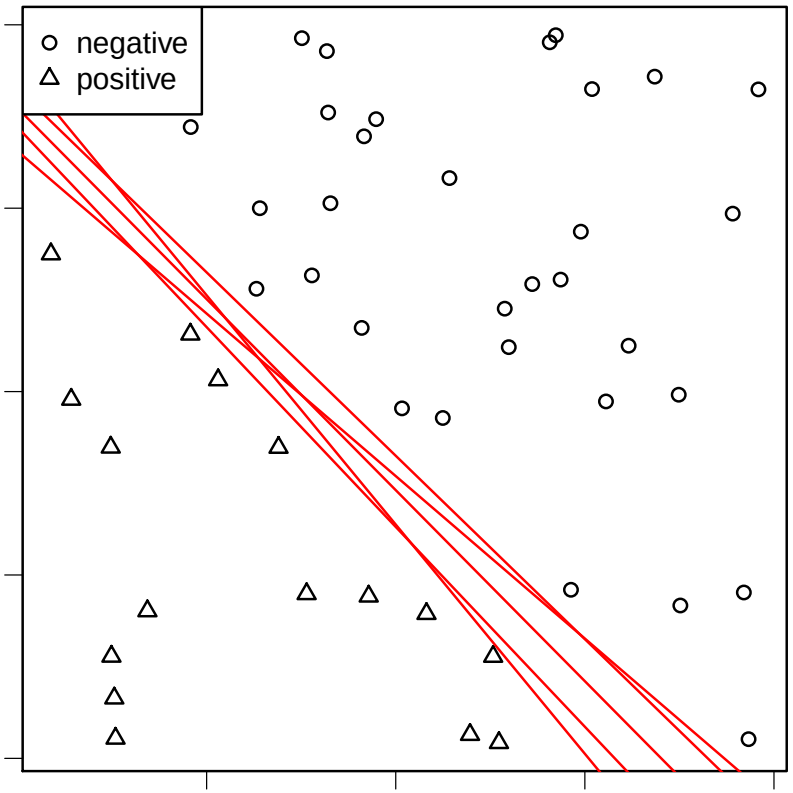
Empirical Risk Minimization

$$[\hat{\mathbf{w}}, \hat{w}_0] = \text{argmin}_{\mathbf{w},w_0} \left(\frac{1}{n} \sum_{i=1}^{n} \left(y_i \neq \text{sign}(\mathbf{w}^T \mathbf{x}_i + w_0) \right) \right)$$



Optimal Separating Hyperplanes

- Consider a binary classification problem where two classes are optimally separable.
- A lot of hyperplanes solve this problem but which one is the best?
- **Intuition**: the **margin** separating both classes has to be **maximized**.

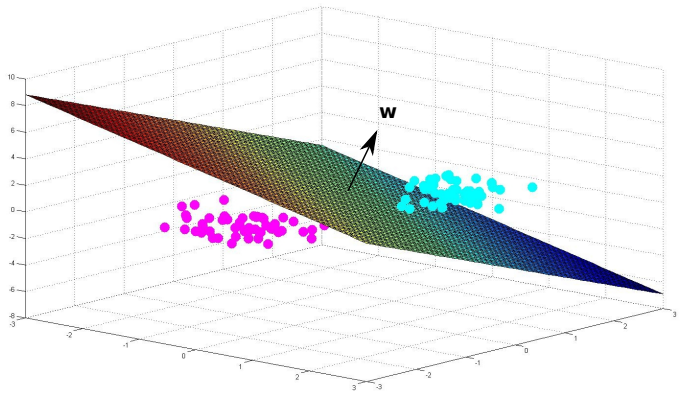


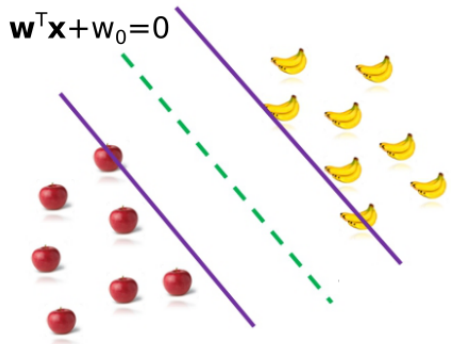
The geometric margin

- Hyperplane equation

  $h(\mathbf{x}) = w_0 + \mathbf{x}^\top \mathbf{w}$
- To find the decision boundary

  $h(\mathbf{x}) = 0$
- $\mathbf{w}$ is orthogonal to the hyperplane.

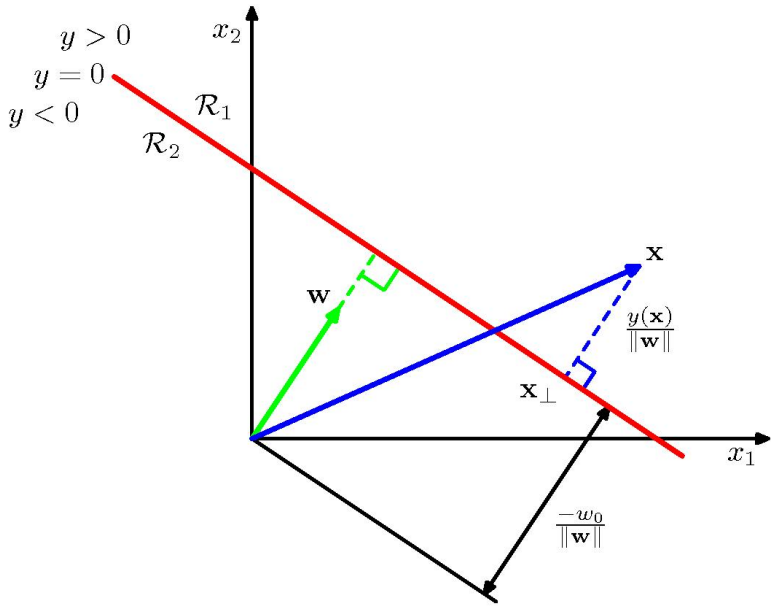




The geometric margin

- $\mathbf{x}_{i\perp}$ is the projection of $\mathbf{x}_i$ onto the hyperplane
- $\tilde{\mathbf{x}}_i = (\mathbf{x}_i - \mathbf{x}_{i\perp})$ a vector from the hyperplane to point $\mathbf{x}_i$
- Project $\tilde{\mathbf{x}}_i$ onto $\mathbf{w}$

$$\text{proj}_{\mathbf{w}}\tilde{\mathbf{x}}_i = \frac{\tilde{\mathbf{x}}_i \cdot \mathbf{w}}{\|\mathbf{w}\|} = \frac{(\mathbf{x}_i - \mathbf{x}_{i\perp})^\top \mathbf{w}}{\|\mathbf{w}\|}$$



Bishop, 2006

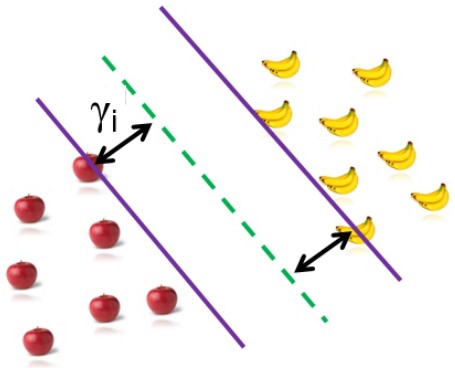
Since $\mathbf{x}_{i\perp}$ on the plane $w_0 + \mathbf{x}_{i\perp}^\top \mathbf{w} = 0$

$$\text{proj}_{\mathbf{w}}\tilde{\mathbf{x}}_i = \frac{\mathbf{x}_i^\top \mathbf{w} - \cancel{\mathbf{x}_{i\perp}^\top \mathbf{w}}}{\|\mathbf{w}\|} = \frac{\mathbf{x}_i^\top \mathbf{w} + w_0}{\|\mathbf{w}\|}$$

The geometric margin

Margin: Signed distance of a point to the hyperplane

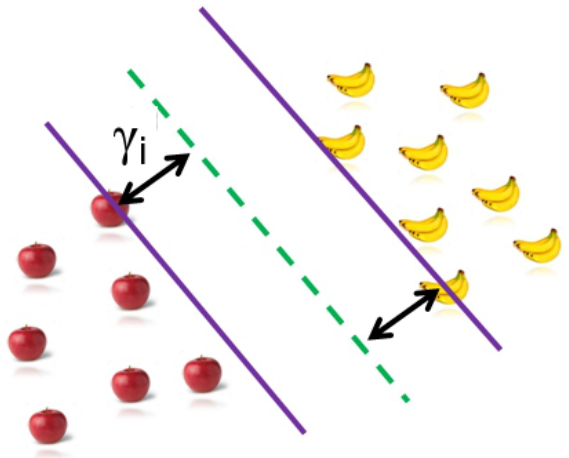
$$\gamma_i = \frac{y_i \left(\mathbf{w}^\top \mathbf{x}_i + w_0 \right)}{\|\mathbf{w}\|}$$



The geometric margin

Margin: smallest distance over all points

$$\gamma = \min_i (\gamma_i) = \min_i \left(\frac{y_i \left(\mathbf{w}^\top \mathbf{x}_i + w_0 \right)}{\|\mathbf{w}\|} \right)$$



Optimal Separating Hyperplanes

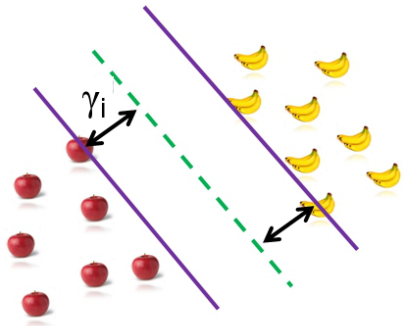
Goal

$$\textbf{Maximize}_{w_0,\mathbf{w}}\ \gamma$$

$$\textbf{subject to}\ \frac{1}{\|\mathbf{w}\|} y_i(\mathbf{w}^\top \mathbf{x}_i^\top + w_0) \geq \gamma, \quad i = 1, \ldots, n$$

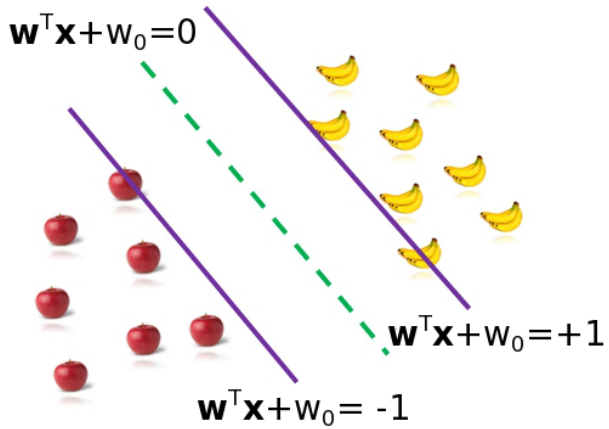
Find a hyperplane that

- separates the two classes
- maximizes the distance to the closest point from either class



Optimal Separating Hyperplanes

$$\gamma_i = \frac{y_i \left(\mathbf{w}^\top \mathbf{x}_i + w_0 \right)}{\|\mathbf{w}\|}$$



- Convert this into an easier equivalent problem.
- Since any scaling of $\mathbf{w}$ and w_0 does not change the margin, we can set: $y_i \left(\mathbf{w}^\top \mathbf{x}_i + w_0 \right) \geq 1$

Margin to closest point $\gamma = \frac{1}{\|\mathbf{w}\|}$

Optimal Separating Hyperplanes

$\max \frac{1}{\|\mathbf{w}\|}$ is equivalent to $\min \|\mathbf{w}\|^2$ so:

Maximum margin problem

- **Minimize** $\frac{1}{2}\|\mathbf{w}\|^2$
- **subject to**: $y_i\left(\mathbf{w}^\top \mathbf{x}_i + w_0\right) \geq 1$ for all $i = 1, \ldots, n$

Optimal Separating Hyperplanes – Optimization

Maximum margin problem

- **Minimize** $\frac{1}{2}\|\mathbf{w}\|^2$
- **subject to**: $y_i\left(\mathbf{w}^\top \mathbf{x}_i + w_0\right) \geq 1$ for all $i = 1, \ldots, n$

The **functional to miminize** $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ is **convex**,

The **constraints** are **linear**: $1 - y_i\left(\mathbf{w}^\top \mathbf{x}_i + w_0\right) \leq 0$

Can be solved with convex optimization, e.g. using **quadratic programming**! (`quadprog` in matlab)

Optimal Separating Hyperplanes - Optimization

Maximum margin problem

- **Minimize** $\frac{1}{2}\|\mathbf{w}\|^2$
- **subject to**: $y_i\left(\mathbf{w}^\top \mathbf{x}_i + w_0\right) \geq 1$ for all $i = 1, \ldots, n$

Construct the **Lagrangian**:

Lagrangian

$$\mathcal{L}(\mathbf{w}, w_0, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \alpha_i \left(1 - y_i(\mathbf{w}^\top \mathbf{x}_i + w_0)\right)$$

Use duality to solve for α_i

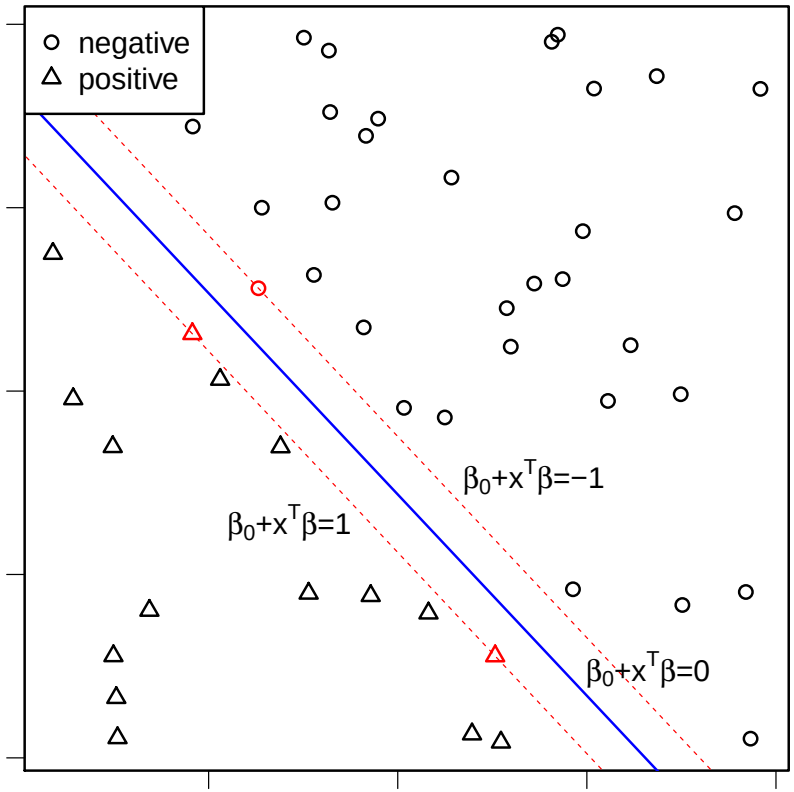
Optimal Separating Hyperplanes

After optimization the estimated weights are found to be:

$$\hat{\mathbf{w}} = \sum_{i}^{n} \alpha_i y_i \mathbf{x}_i$$

which are just to be replaced inside the hyperplane equation:

$$\hat{h}(\mathbf{x}) = \hat{\mathbf{w}}^{\top}\mathbf{x} + \hat{w}_0$$



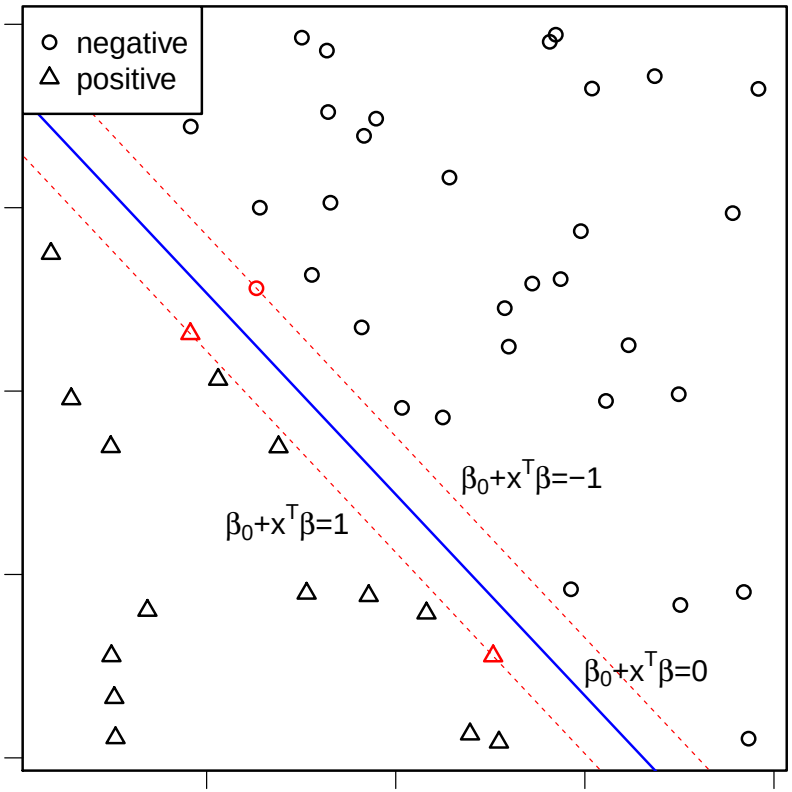
Optimal Separating Hyperplanes

After optimization the estimated hyperplane weights are found to be:

$$\hat{\mathbf{w}} = \sum_{i}^{n} \alpha_i y_i \mathbf{x}_i$$

Weights are now a simple **linear combination** of the data points

$$\hat{h}(\mathbf{x}) = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i^{\top} \mathbf{x} + \hat{w}_0$$



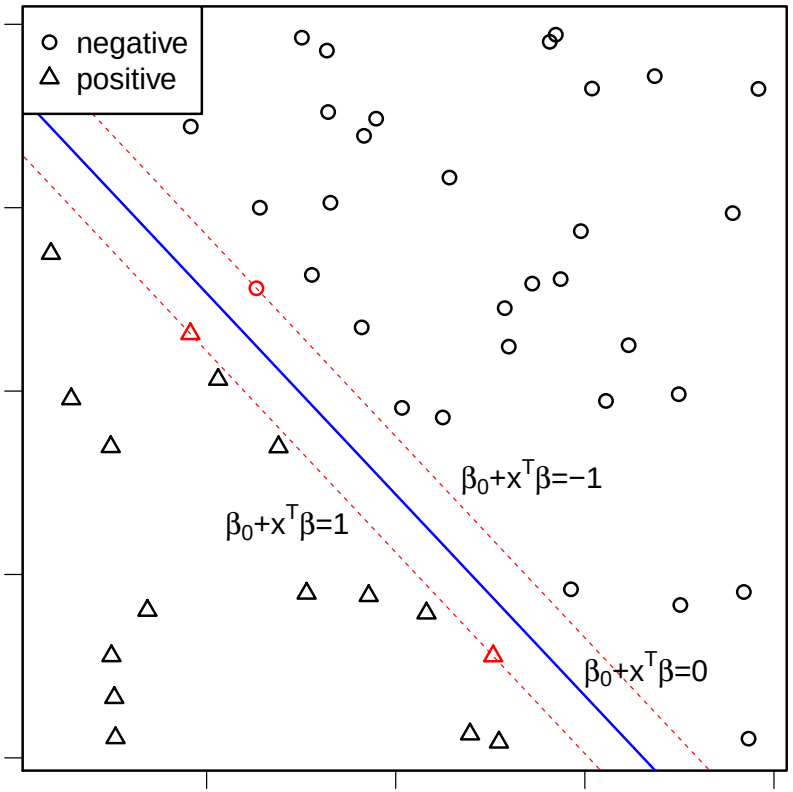
Optimal Separating Hyperplanes – Support Points

$$\hat{h}(\mathbf{x}) = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i^\top \mathbf{x} + \hat{w}_0$$

$\hat{\alpha}_i \geq 0$ are **weights** such that

- $\hat{\alpha}_i \neq 0$ if $\mathbf{x}_i$ **support vector**
- $\hat{\alpha}_i = 0$ otherwise

The solution only depends on the support points not on the whole data set!!!



Optimal Separating Hyperplanes – Prediction

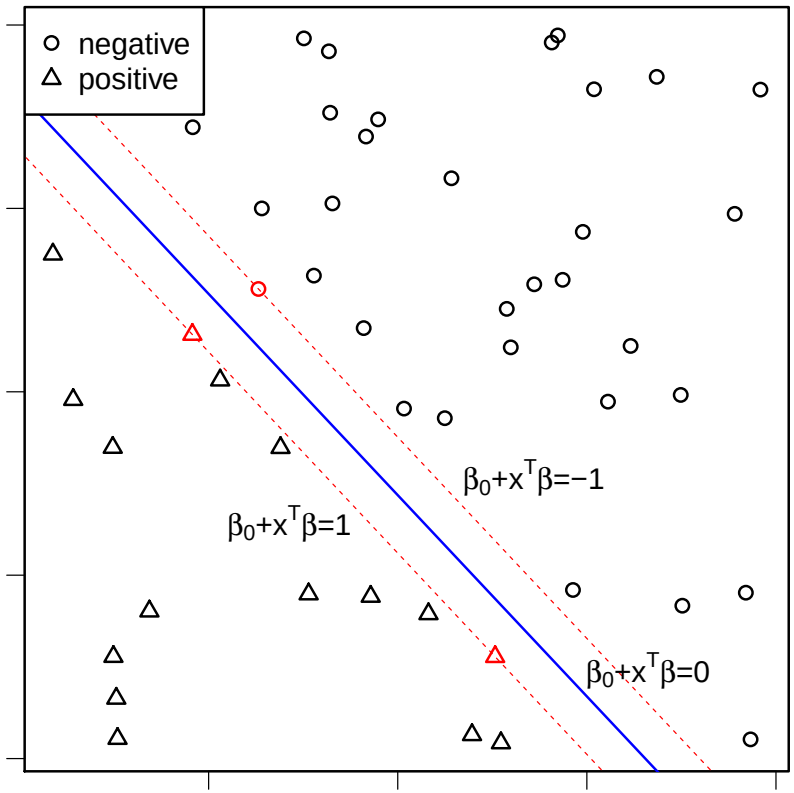
At optimum $\hat{\mathbf{w}}$ is

$$\hat{\mathbf{w}} = \sum_{i}^{n} \alpha_i y_i \mathbf{x}_i$$

A new sample is classified by

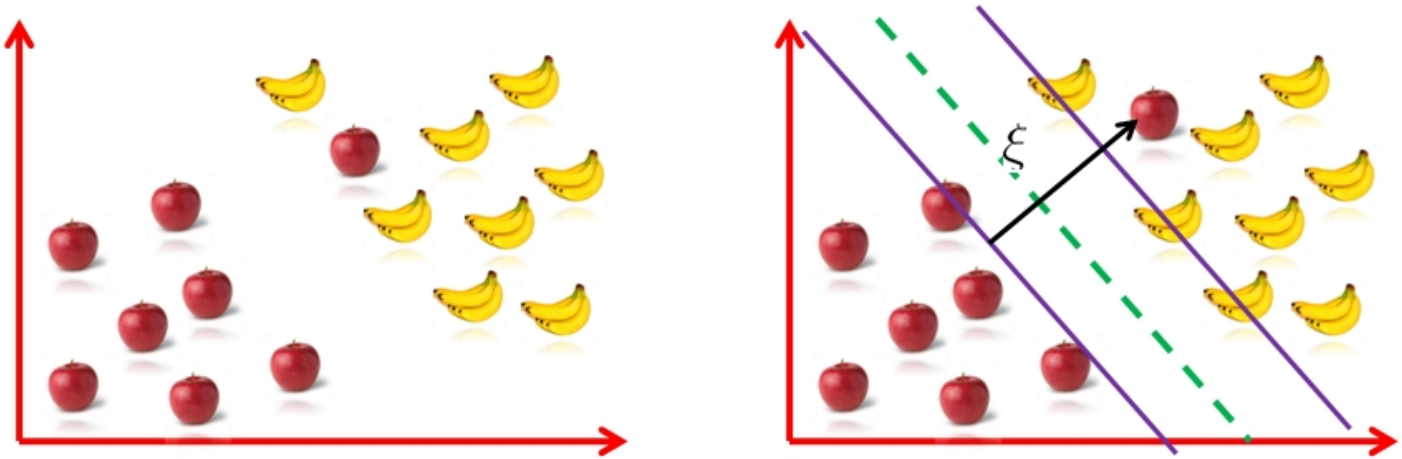
$$f(\mathbf{x}_j) = \text{sign}(\hat{h}(\mathbf{x}_j))$$

$$f(\mathbf{x}_j) = \text{sign}(\hat{w}_0 + \mathbf{x}_j^\top \hat{\mathbf{w}})$$



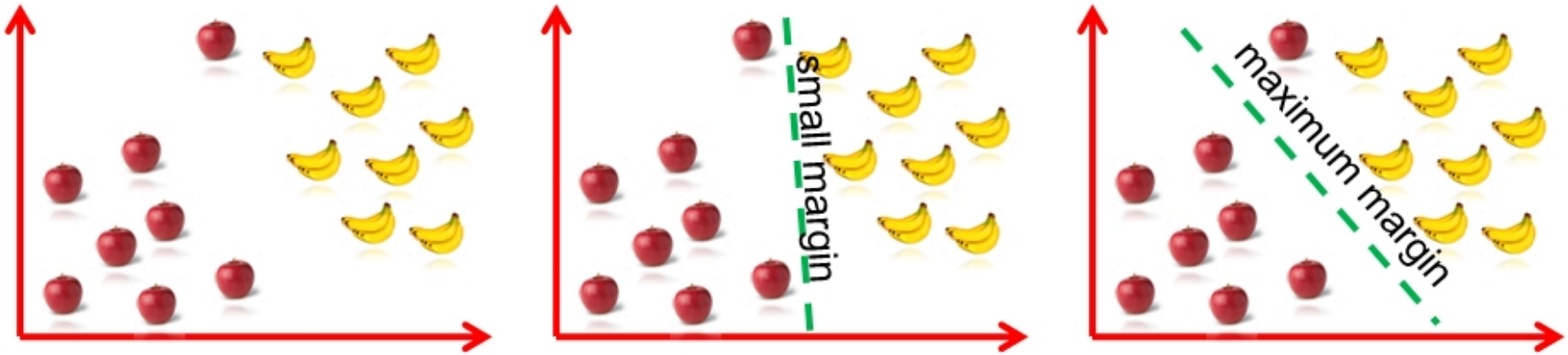
Outline

Linear separability



- In real applications classes are rarely **linearly separable**
- In addition: Noise in the features? Mislabelled data? Outliers?
- **Soft margin**: allow **margin violations**, i.e. misclassification errors

Soft margin – generalization

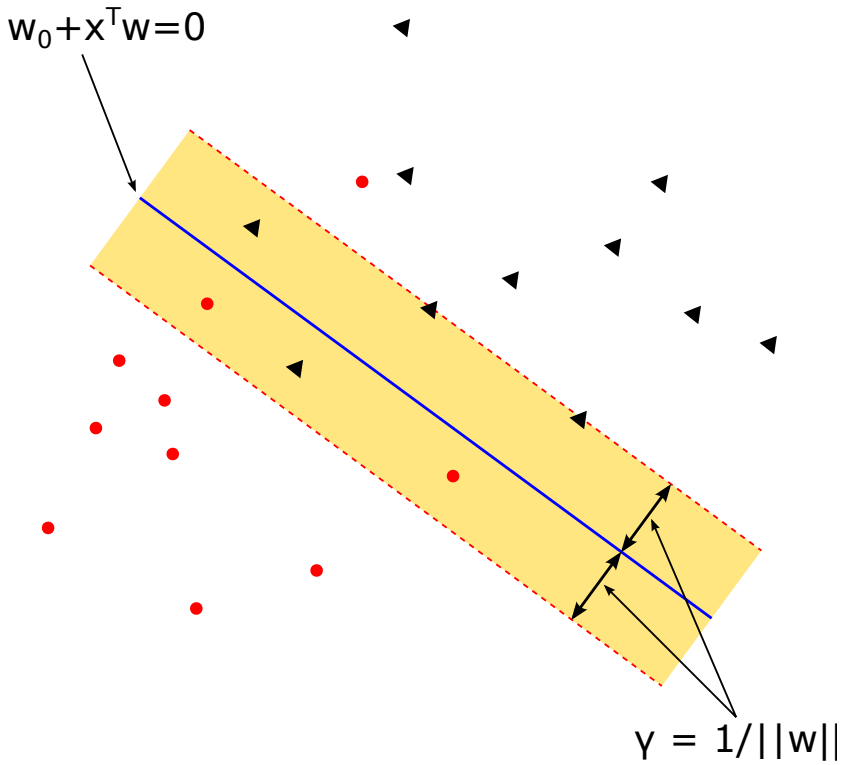


Soft margin

- deals with noisy data and outliers
- provides a better **generalization** ability
- prevents from **overfitting** the data, i.e. prevents the model of describing random errors or noise.

Support Vector Machines–Slack variables

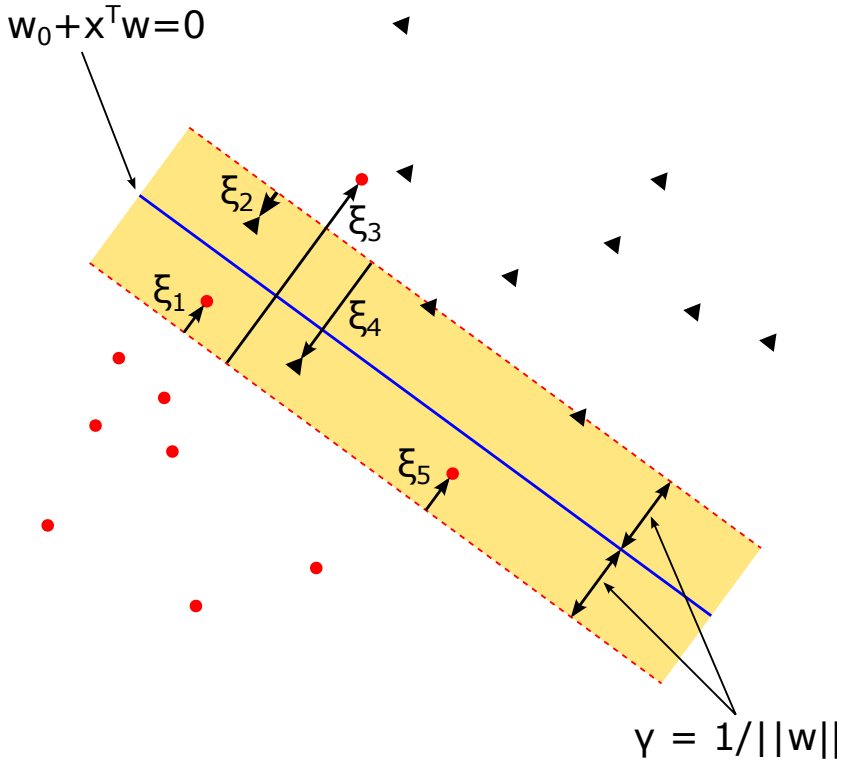
- Still maximise the margin but allow for some points to reside on the wrong side of the margin (**soft margin**).



Support Vector Machines

Introduce for each sample a **slack variable** $\xi_i \geq 0$

- $\xi_i = 0$ **correct** classification
- if $\xi_i > 1$: **missclassification**
- If $0 < \xi_i \leq 1$: point lies between the margin and the correct side of the margin.



ξ_i gives the relative amount, with respect to the margin, by which the prediction falls on the wrong side of the hyperplane.

Support Vector Machines

Definition (SVM Optimization)

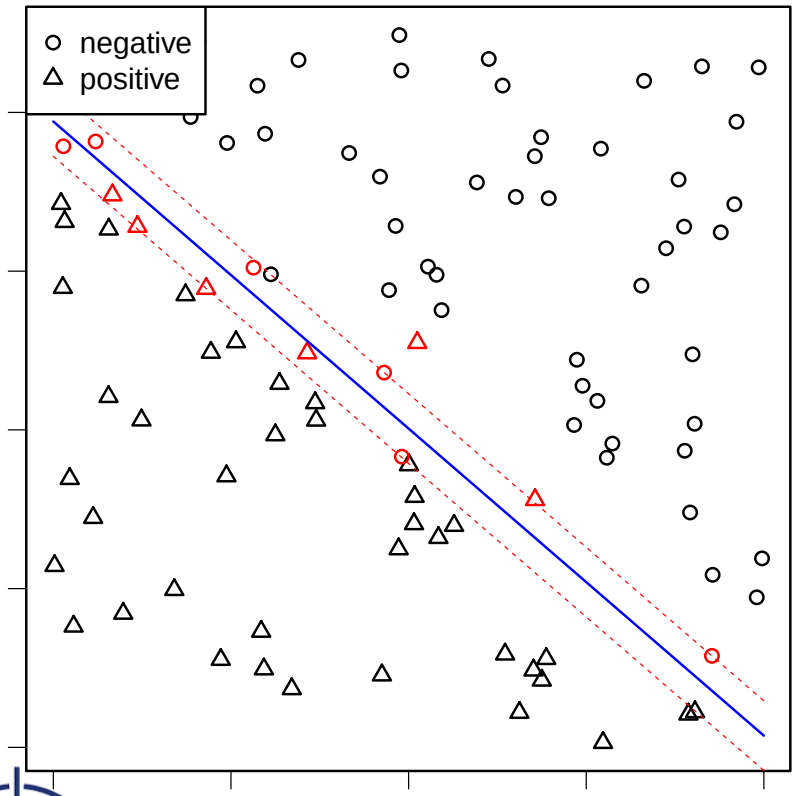
$$\min_{w_0, \mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{subject to } \xi_i \geq 0 \text{ and } y_i(w_0 + \mathbf{x}_i^\top \mathbf{w}) \geq 1 - \xi_i$$

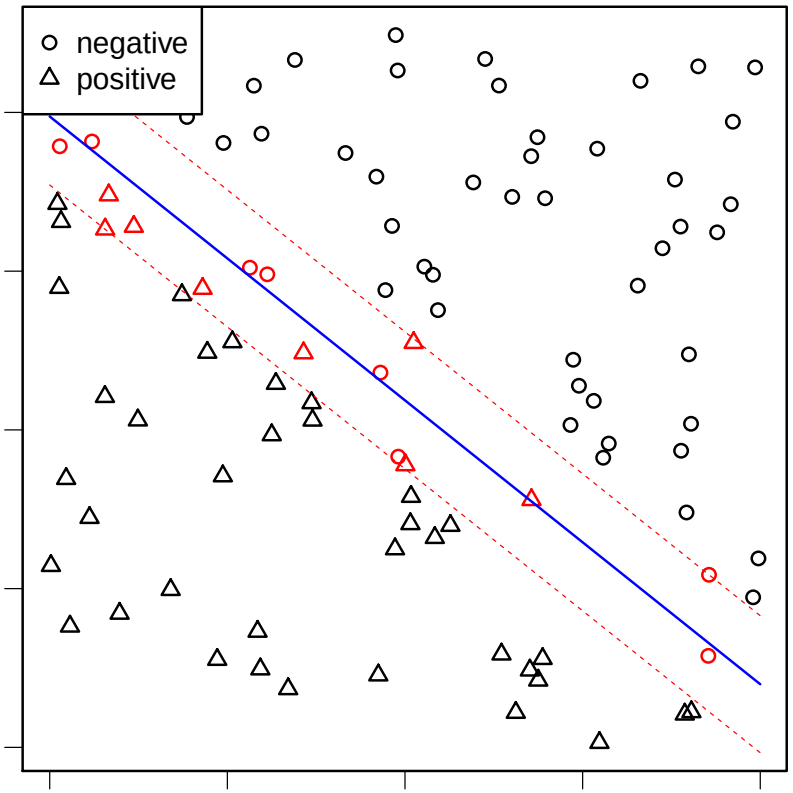
- The parameter $C > 0$ controls the trade-off between the slack variable penalty and the margin.
- If $C = \infty$, result equal to *optimal separating hyperplanes*.
- $\sum \xi_i$ is an upper bound on the number of misclassified points.

Support Vector Machines – Examples

$C = 10000$



$C = 1$



SVM Optimization

Regularized maximum margin

- **Minimize** $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$
- **subject to**: $y_i\left(\mathbf{w}^\top\mathbf{x}_i + w_0\right) \geq 1 - \xi_i$ for all i
  $\xi_i \geq 0$ for all i

How to solve this Convex Optimization Problem

- Use **Lagrange Multipliers** to convert a constrained optimization problem into an unconstrained one
- Introduce new variables $\alpha_i \geq 0$ $r_i \geq 0$ to weight the constraints in a unique cost function

SVM optimization

Lagrangian

$$\mathcal{L}(\mathbf{w}, w_0, \xi, \alpha, \mathbf{r}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i\left(1 - \xi_i - y_i(\mathbf{w}^\top \mathbf{x}_i + w_0)\right) - \sum_{i=1}^{N} r_i\xi_i$$

Use **Lagrange duality** ([Bishop 2006]) and an efficient method **Sequential Minimal Optimization** (SMO)

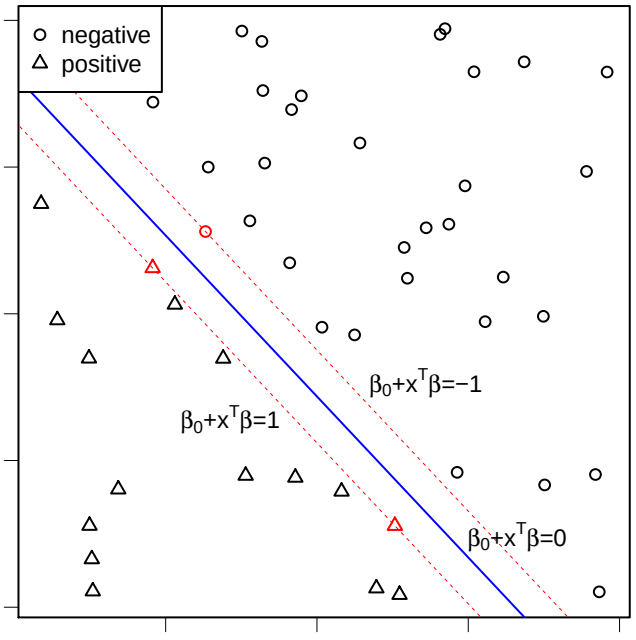
SVM – Prediction

At optimum $\hat{\mathbf{w}}$ can be estimated as

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

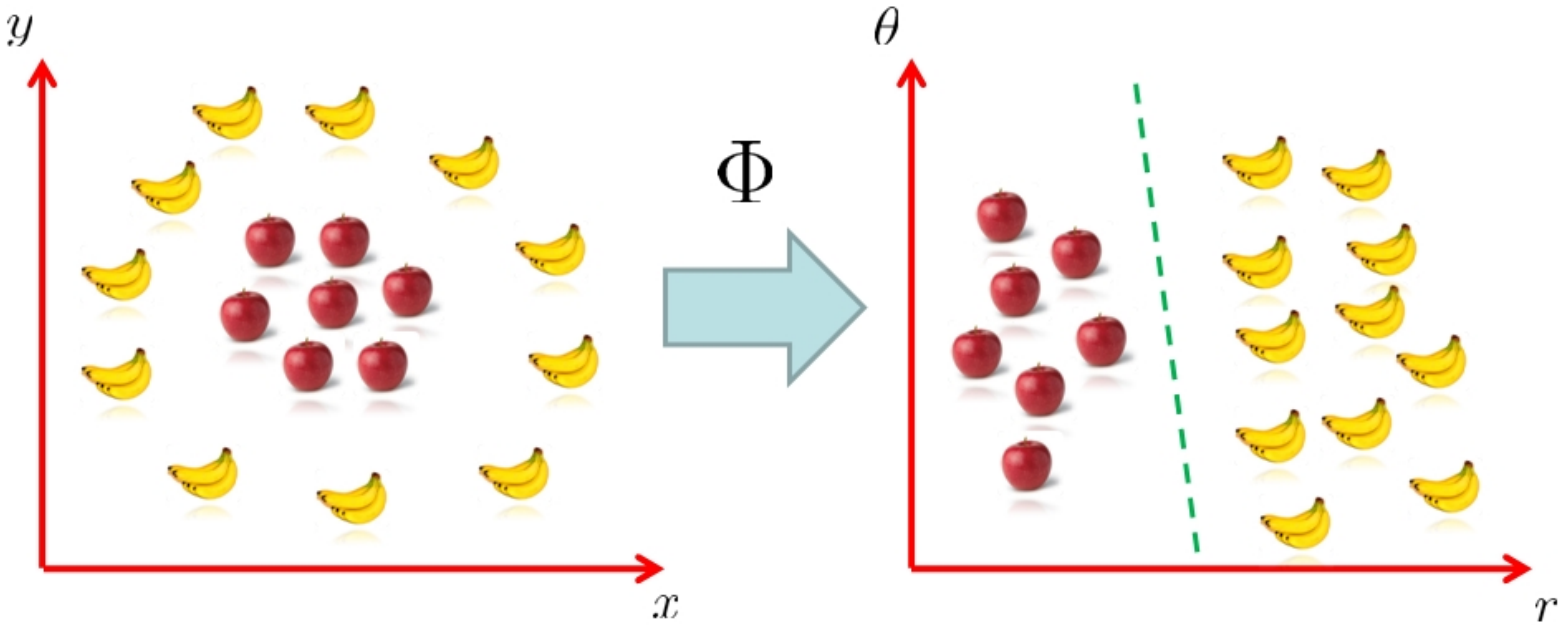
Prediction

$$\hat{f}(\mathbf{x}_i) = \text{sign}(\hat{w}_0 + \mathbf{x}_i^\top \hat{\mathbf{w}})$$
$$\hat{f}(\mathbf{x}_i) = \text{sign}(\hat{w}_0 + \sum_{i=1}^{n} \hat{\alpha}_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle)$$



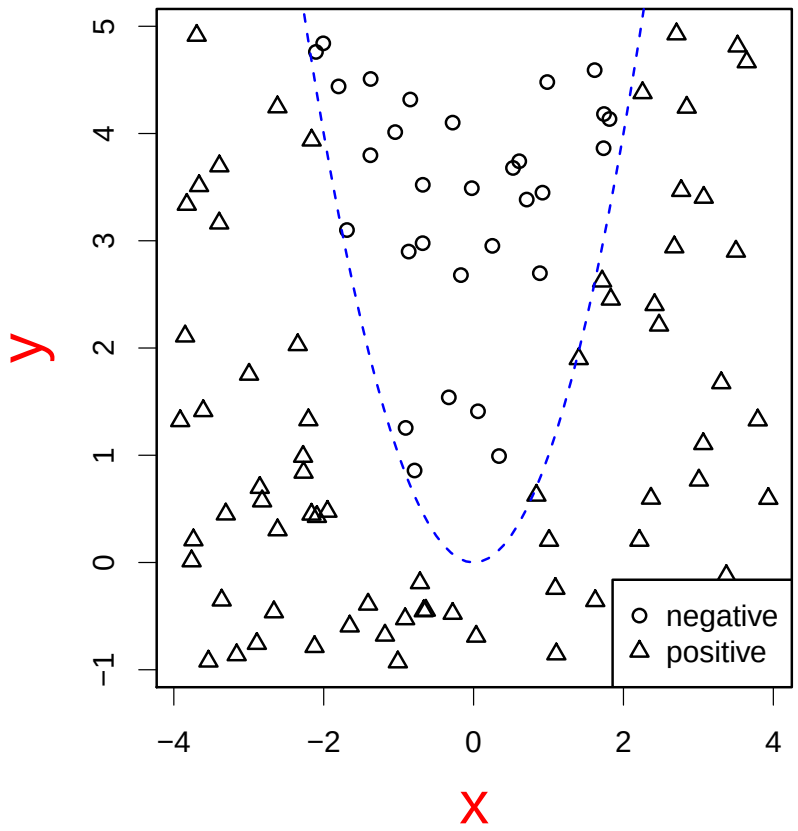
Outline

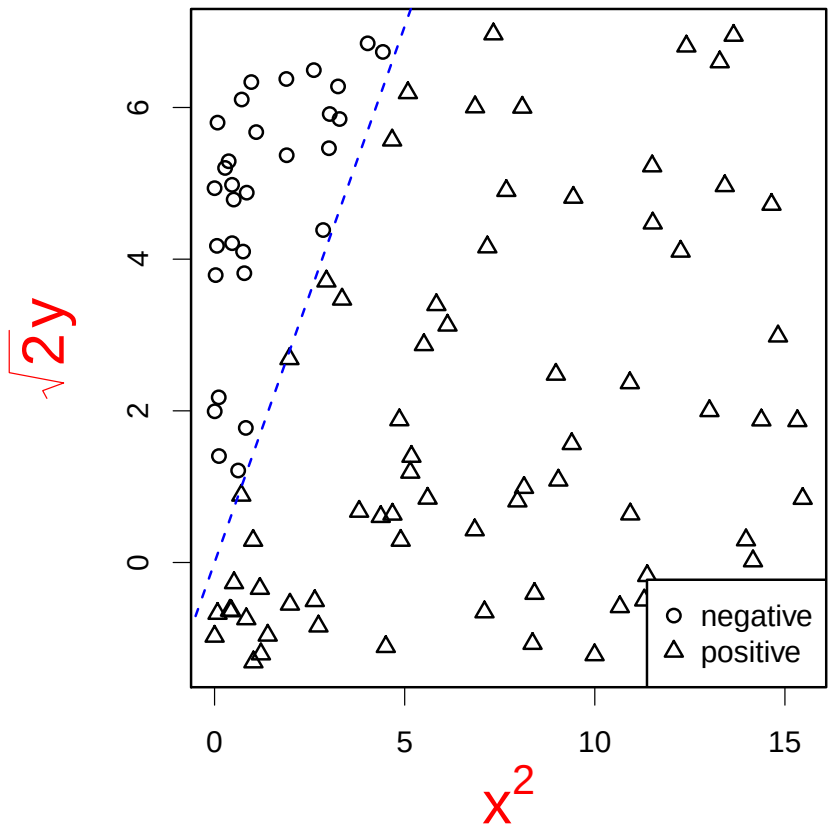
Non-linear separability



- Data are not always **linearly separable**, however...they may be separable in another (higher dimensional) space!
- **Idea**: find a **non-linear mapping** from the input space into a (higher dimensional) feature space in which data are separable.

Non-linear SVMs – Transformation





Example: Transform point (x, y) to $(x^2, \sqrt{2}y)$ where the data can be separated linearly.

Non-linear SVMs – Transformation

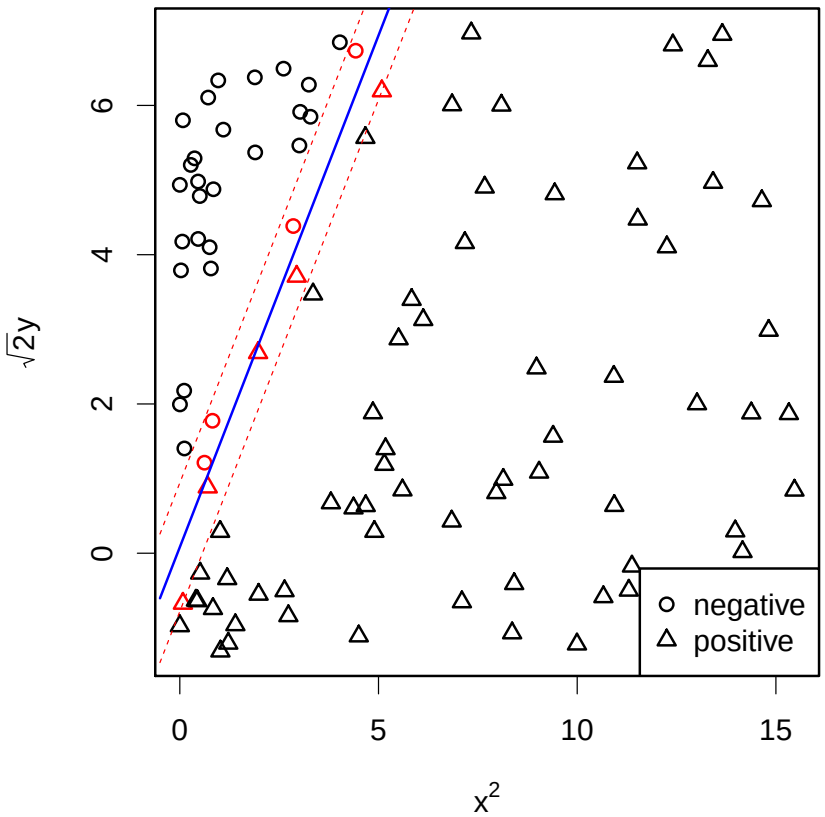
Map data

- from the input space $\mathcal{X} \subseteq \mathbb{R}^d$
- to feature space $\mathcal{F} \subseteq \mathbb{R}^D$
- using a **non-linear function**

$$\phi : \mathcal{X} \rightarrow \mathcal{F}$$

The hyperplane function becomes

$$h(\mathbf{x}_i) = w_0 + \phi(\mathbf{x}_i)^\top \mathbf{w}$$



Non-linear SVMs – Optimization

Given any (non-linear) mapping $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^D$, with $d \leq D$:

Non-linear regularized maximum margin

- **Minimize** $\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i$
- **subject to**: $y_i \left(\mathbf{w}^\top \phi(\mathbf{x}_i) + w_0 \right) \geq 1 - \xi_i$ for all i
  $\xi_i \geq 0$ for all i

Non-linear SVMs – Optimization

- In the linear SVMs

$$h(\mathbf{x}) = w_0 + \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x}$$

- Applying the hyperplane after the transformation function ϕ :

$$h(\mathbf{x}) = w_0 + \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i)^\top \phi(\mathbf{x})$$

The solution $\hat{\mathbf{w}}$ is a linear combination of the training data!

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i)$$

Non-linear SVMs – Kernel

Definition (Kernel Function)

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$$

Definition (Kernel SVM)

$$h(\mathbf{x}_0) = w_0 + \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}_0)$$

Kernel Trick

- If the Kernel function can be computed efficiently, we can avoid to explicitly transform the data into the feature space.
- No explicit representation of ϕ is required.

Kernel Functions

- Linear:

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$$

- d-th degree Polynomial:

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + c)^d$$

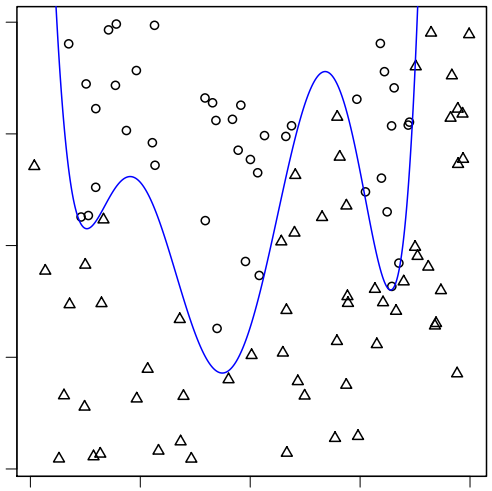
- Radial Basis Function (RBF):

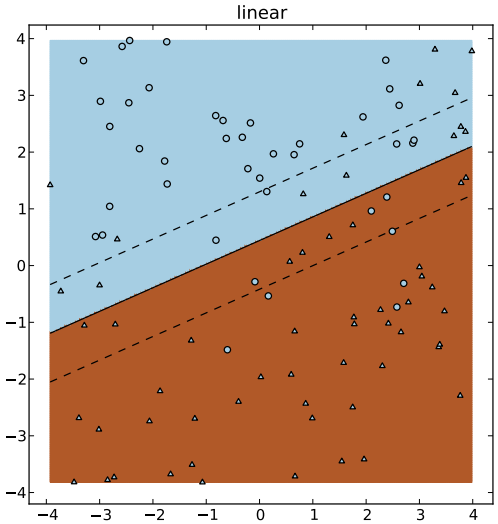
$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right)$$

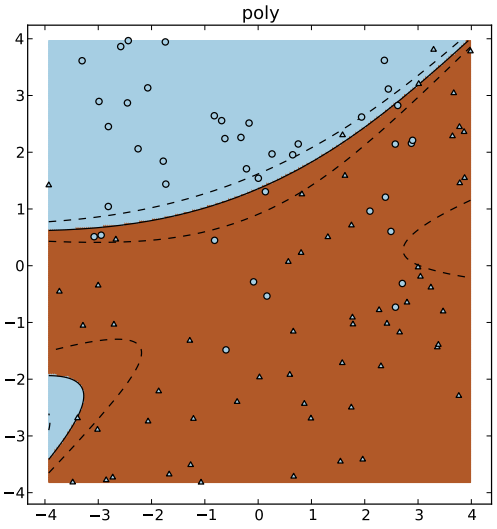
- Sigmoid:

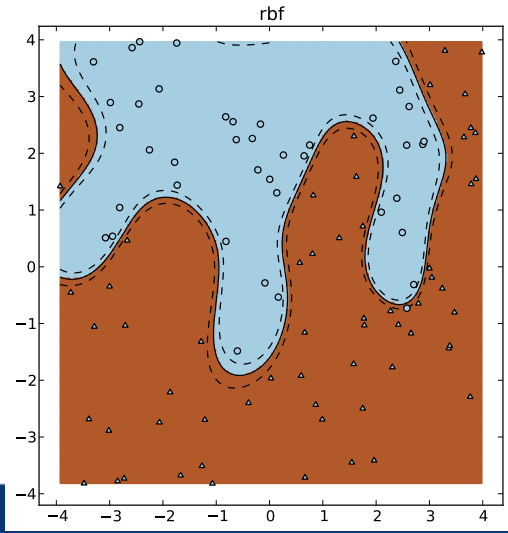
$$K(\mathbf{x}, \mathbf{x}') = \tanh(\gamma \cdot \mathbf{x}^\top \mathbf{x}' + c)$$

Kernel Functions – Examples









Conclusion

Support Vector Machines

- SVMs find the **optimal** separating hyperplane which **maximizes** the margin between the classes
- In case of non separable data, using a **soft margin** allows misclassification errors. This provides also better generalization ability in the cases of noisy data or outliers.
- Non-linear decision function can be modeled by using **Kernels**. They can project data in higher dimensional features space in which they become linearly separable.

However: no formulation for multi-class classification...

Multi-class SVMs

- SVMs as previously discussed are only applicable to binary classification problems.
- **Idea**: Construct multiple binary SVMs to distinguish $k > 2$ classes from each other.
- **One vs. all**: Train k classifiers where the i-th classifier is given the labels of the i-th class as positives and everything else as negative.
- **One vs. One**: Train $\sum_{i=1}^{k-1} i$ classifiers where each classifier is trained on samples from the i-th and j-th class, respectively.

Summary

- **Optimal separating hyperplanes** can be applied rarely.
- **Support vector machines** can be used both for classification and regression and thanks to the **Kernel trick** in a wide range of applications. The best choice of Kernel and its parameters is not obvious and requires lots of testing.

True Positives and False Negatives

	$\hat{y} = 1$	$\hat{y} = 0$
$y = 1$	TP	FN
$y = 0$	FP	TN

True Positive Rate/Recall/Sensitivity $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$

False Positive Rate/(1-Specificity) $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP+TN}}$

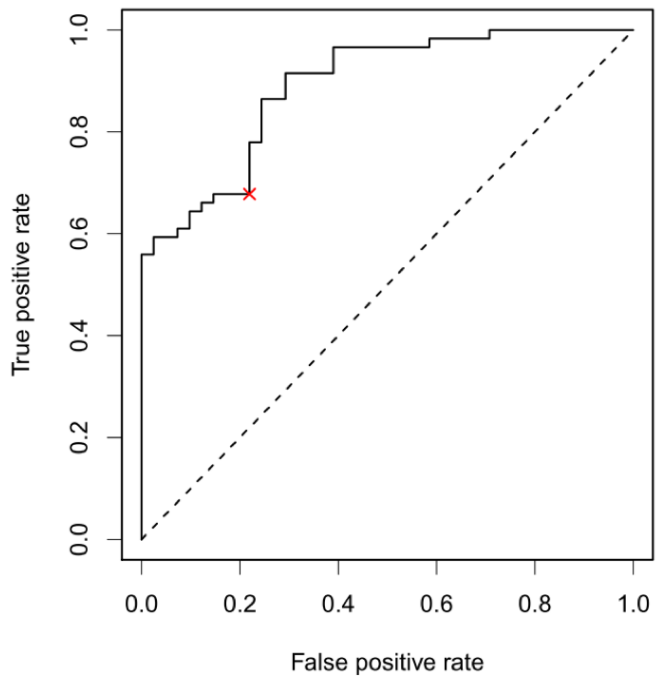
Precision $P = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$

Receiver operating characteristic (ROC) curve

True Positive Rate/Recall/Sensitivity $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$

False Positive Rate/(1-Specificity) $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP+TN}}$

- Compares sensitivity (y-axis) with false positive rate (x-axis)
- Requires probability or score output of binary classifier
- One point = one threshold of the classifier's score
- Tradeoff between the benefits (sensitivity) and costs (FPR)

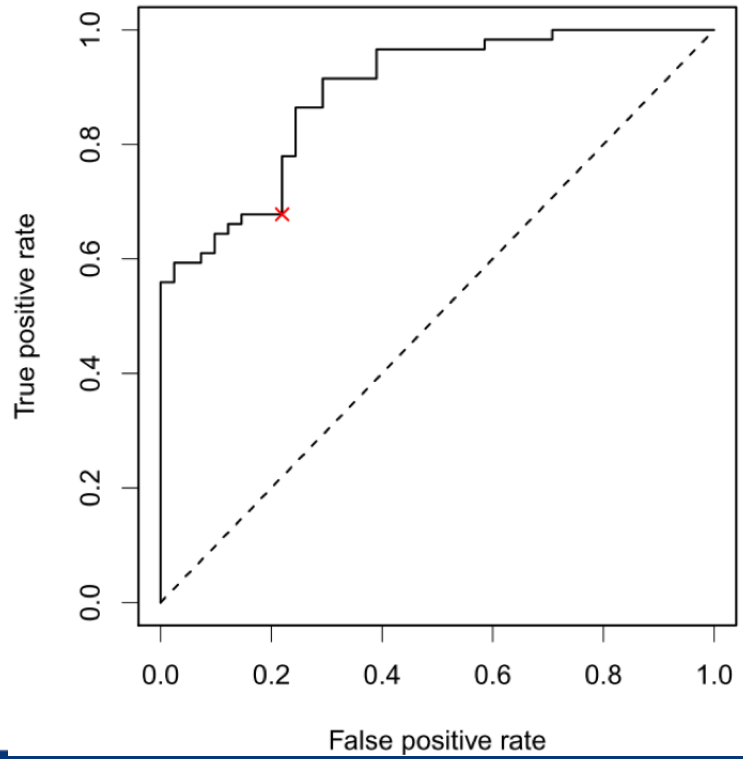


ROC curve

True Positive Rate/Recall/Sensitivity $\mathrm{TPR} = \frac{|\hat{y}_i{=}1 \wedge y_i{=}1|}{|y_i{=}1|}$

False Positive Rate/(1-Specificity) $\mathrm{FPR} = \frac{|\hat{y}_i{=}1 \wedge y_i{=}0|}{|y_i{=}0|}$

- **Random** classifier: Line from low left to upper right corner
- **Perfect** classifier: goes through the upper left corner at (0,1)
- A single confusion matrix corresponds to one point



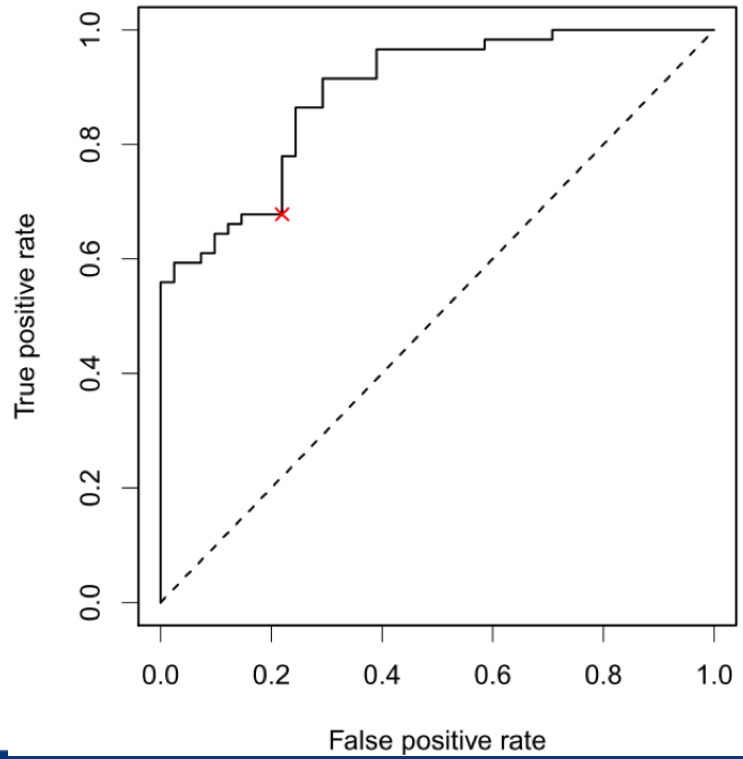
ROC - AUC

True Positive Rate/Recall/Sensitivity $\mathrm{TPR} = \frac{|\hat{y}_i{=}1 \wedge y_i{=}1|}{|y_i{=}1|}$

False Positive Rate/(1-Specificity) $\mathrm{FPR} = \frac{|\hat{y}_i{=}1 \wedge y_i{=}0|}{|y_i{=}0|}$

The area under curve (AUC) is:

- the probability that the classifier will rank a randomly chosen positive instance higher than a negative instance
- a good measure of the classifier performance
- related to the Gini coefficient: $\mathrm{Gini\,coeff} = 1 - 2 * \mathrm{AUC}$



Disadvantages of ROC

- ROC curves can present an overly optimistic view of an algorithm's performance if there is a large skew in the class distribution, i.e. the data set contains much more samples of one class.
- A large change in the number of false positives can lead to a small change in the false positive rate (FPR) if negative samples dominates.

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

- Comparing false positives to true positives (precision) rather than true negatives (FPR), captures the effect of the large number of negative examples.

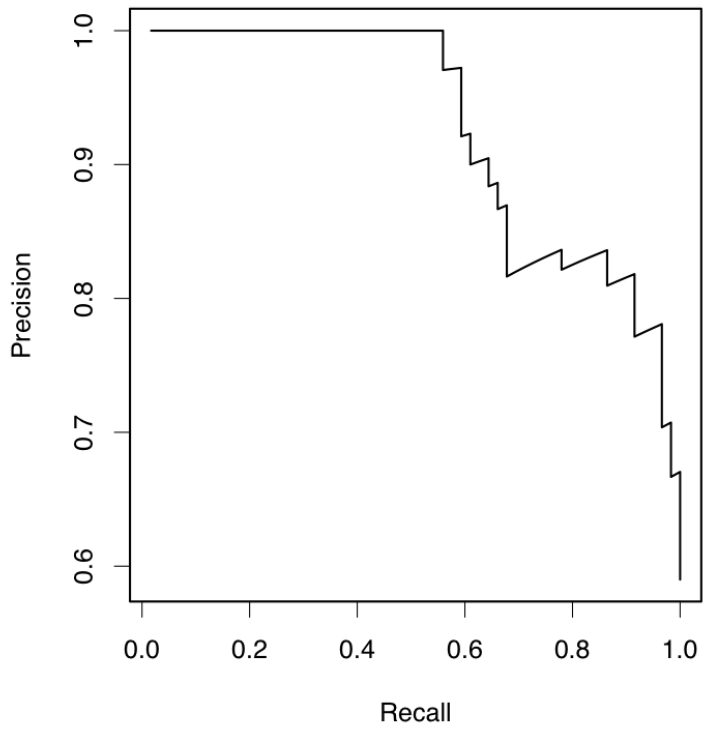
$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Precision Recall Curve

True Positive Rate/Recall/Sensitivity $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} = \frac{|\hat{y}_i{=}1 \wedge y_i{=}1|}{|y_i{=}1|}$

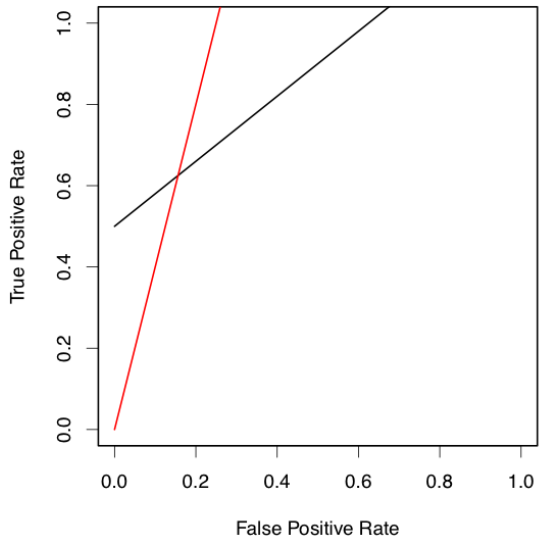
Precision $P = \frac{\mathrm{TP}}{\mathrm{TP+FP}} = \frac{|\hat{y}_i{=}1 \wedge y_i{=}1|}{|\hat{y}_i{=}1|}$

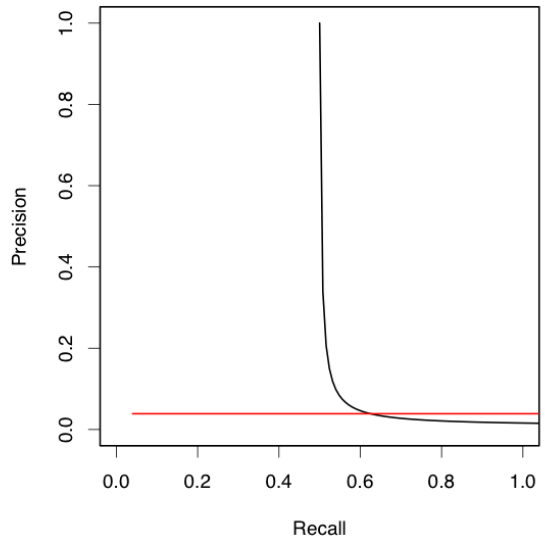
- Compares precision to recall.
- One point is equivalent to
  - one threshold value
  - to a single confusion matrix
- PR curve of optimal classifier is in the upper-right corner
- Average precision is the area under the PR curve



ROC vs Precision Recall Curves

Algorithms that optimize the area under the ROC curve are not guaranteed to optimize the area under the PR curve





Dummy example: Dataset for retrieval with 20 positive and 2000 negative documents
(black) 10 of the 20 hits are in the top ten ranks and the remaining 10 hits evenly spread out over the first 1500 ranks.
(red) the 20 hits are evenly spread over the first 500 ranks.