Introduction to Machine Learning
NPFL 054

http://ufal.mff.cuni.cz/course/npfl054

Barbora Hladká
hladka@ufal.mff.cuni.cz

Martin Holub
holub@ufal.mff.cuni.cz

Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Outline

- Support Vector Machines
- Evaluation of binary classifiers (cntnd): ROC curve

Support Vector Machines
Key idea for binary classification

We find a hyperplane that separates the two classes in the feature space.

If it is not possible

- allow some training errors, or
- enrich the feature space so that finding a separating hyperplane is possible

Support Vector Machines
Key points & concepts

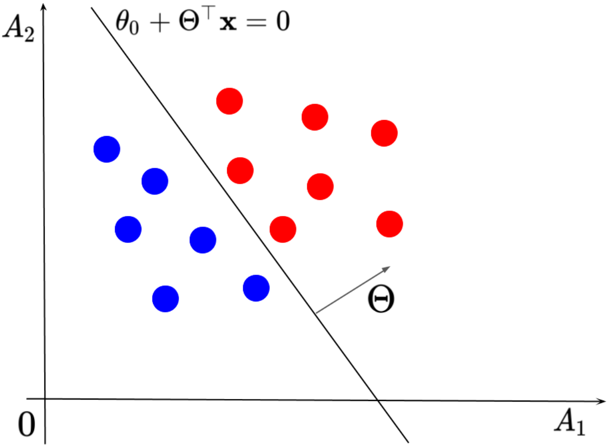
Key points

1. Maximizing the margin
2. Duality optimization task
3. Kernels

Key conceppts

1. Hyperplane
2. Dot product
3. Quadratic programming

SVM
Hyperplane



Support Vector Machines
Classification using hyperplane

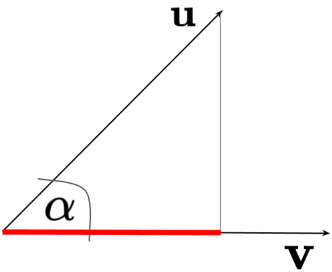
Recall the classification rule using $\Theta^\top \mathbf{x} = 0$

$$f(\mathbf{x}) = \begin{cases} 1 \text{ if } \theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m \geq 0 \\ 0 \text{ if } \theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m < 0 \end{cases}$$

We can use the classification rule for different values of threshold.
Here threshold = 0.

Support Vector Machines
Dot product

- $\mathbf{u} = \langle u_1, \ldots, u_m \rangle$, $\mathbf{v} = \langle v_1, \ldots, v_m \rangle$
- algebraic definition $\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + \cdots + u_m v_m$
- geometric definition $\mathbf{u} \cdot \mathbf{v} = ||\mathbf{u}|| \cdot ||\mathbf{v}|| \cdot \cos\alpha$



$$||\mathbf{v}|| = 1 \rightarrow \mathbf{u} \cdot \mathbf{v} = ||\mathbf{u}|| \cdot \cos\alpha$$

Support Vector Machines
Quadratic programming

Quadratic programming optimizes a quadratic objective function subject to constraints.

I.e. minimize/maximize

$$f(\mathbf{x}), \mathbf{x} \in \mathcal{D}$$

subject to

$$g_i(\mathbf{x}) \leq 0, i = 1, \ldots, G$$

$$h_j(\mathbf{x}) = 0, j = 1, \ldots, H$$

Support Vector Machines
Margins

Training examples $D = \{\langle \mathbf{x}_i, y_i \rangle, \mathbf{x}_i \in X, y_i \in \{-1, +1\}\}$

Hyperplane g: $\theta_0 + \Theta^\top \mathbf{x} = 0$

- **Margin of x** w.r.t. g is distance of $\mathbf{x}$ to g

$$\rho_g(\mathbf{x}) = \frac{|\theta_0 + \Theta^\top \mathbf{x}|}{||\Theta||}$$

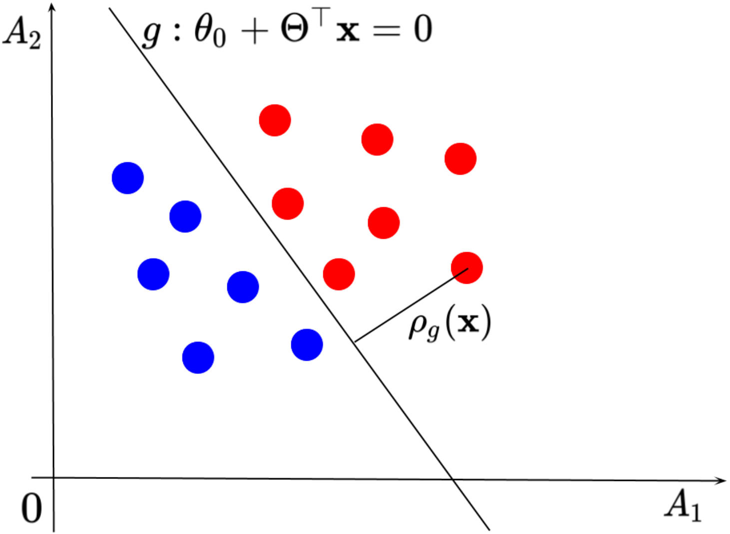
- **Functional margin of** $\langle \mathbf{x}, y \rangle$ w.r.t. g is

$$\overline{\rho}_g(\mathbf{x}, y) = y(\Theta_0 + \Theta^\top \mathbf{x})$$

- **Geometric margin of** $\langle \mathbf{x}, y \rangle$ w.r.t. g is functional margin scaled by $||\Theta||$

$$\rho_g(\mathbf{x}, y) = \overline{\rho}_g(\mathbf{x}, y)/||\Theta||$$

Support Vector Machines
Margins



Support Vector Machines
Margins

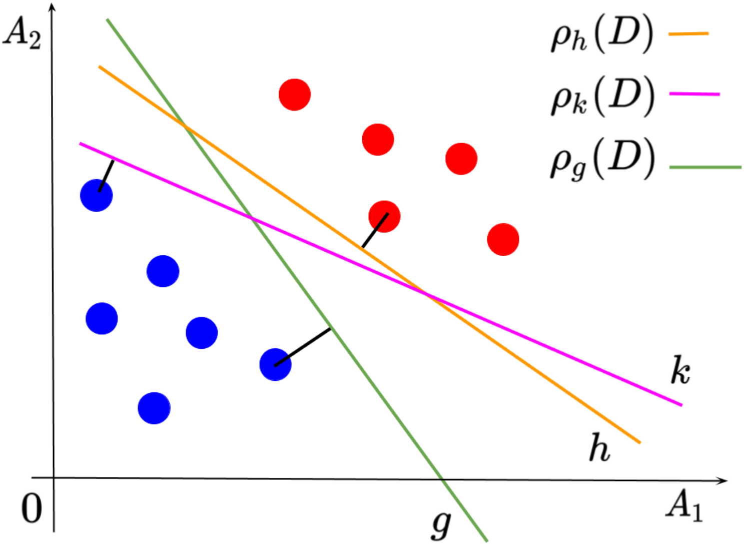
- **Functional margin of** D w.r.t. g

$$\overline{\rho}_g(D) = \min_{\langle \mathbf{x}, y\rangle \in D} \overline{\rho}_g(\mathbf{x}, y)$$

- **Geometric margin of** D w.r.t. g

$$\rho_g(D) = \min_{\langle \mathbf{x}, y\rangle \in D} \rho_g(\mathbf{x}, y)$$

Support Vector Machines
Margins



Support Vector Machines
Margins

Data set $D = \{\langle \mathbf{x}_i, y_i \rangle, \mathbf{x}_i \in X, y_i \in \{-1, +1\}\}$ is **linearly separable** if there exists a hyperplane $g : \theta_0 + \Theta^\top \mathbf{x} = 0$ that separates the two classes completly, i.e.

$$\forall \langle \mathbf{x}, y \rangle \in D : \quad \overline{\rho_g}(\mathbf{x}, y) > 0$$

Support Vector Machines
Binary classification task $Y = \{+1, -1\}$

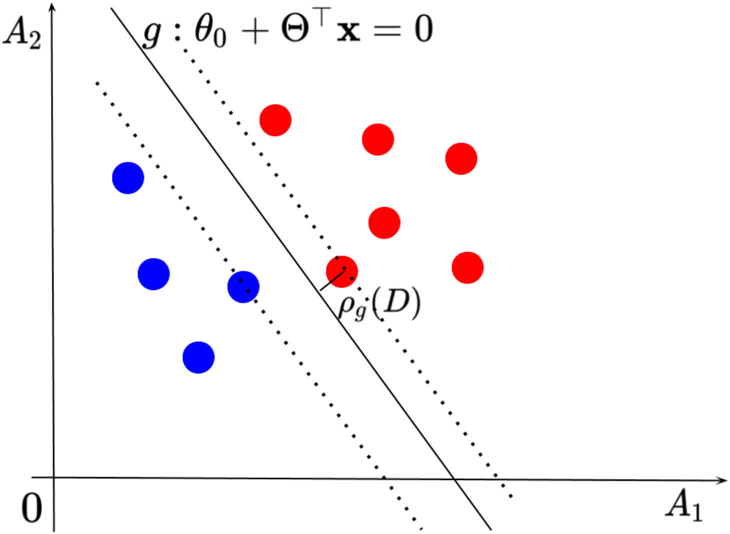
1. Large margin classifier (linear separability)
2. Soft margin classifier (not linear separability)
3. Kernels (non-linear class boundaries)

Large Margin Classifier
Training data is linearly separable

Optimization task

$$g^{\star} = \mathrm{argmax}_g \rho_g(D)$$

Large Margin Classifier
Training data is linearly separable



Large Margin Classifier
Training data is linearly separable

$\Theta_0 + \Theta^\top \mathbf{x}$ and $k\Theta_0 + (k\Theta)^\top \mathbf{x}$ define the same hyperplane.

$$\frac{y_i(\Theta_0 + \Theta^\top \mathbf{x}_i)}{||\Theta||} = \frac{y_i(k\Theta_0 + (k\Theta)^\top \mathbf{x}_i)}{||k\Theta||}$$

Therefore we can scale Θ so that $\overline{\rho}_g(D) = 1$.

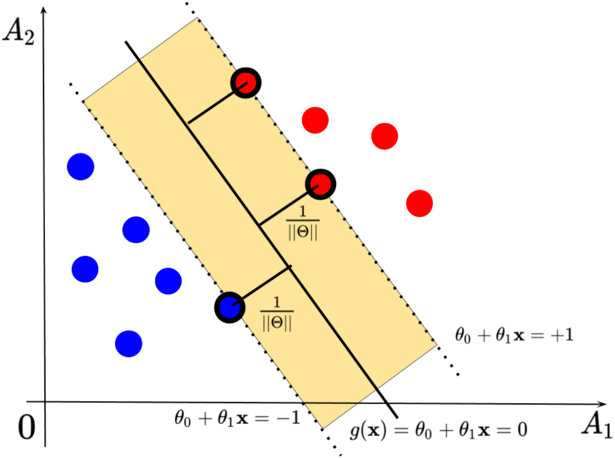
Then

$$g^\star = \mathrm{argmax}_g \rho_g(D) = \max_\Theta \frac{2}{||\Theta||}$$

Large Margin Classifier
Training data is linearly separable

Supporting hyperplanes $\theta_0 + \Theta^\top \mathbf{x} = -1$ and $\theta_0 + \Theta^\top \mathbf{x} = +1$.

Support vectors are the instances touching the supporting hyperplanes.



Large Margin Classifier
Training data is linearly separable

Primal problem

Minimize

$$\frac{1}{2}||\Theta||^2$$

subject to

$$y_i(\theta_0 + \Theta^\top \mathbf{x}_i) \geq 1, i = 1, \ldots, n$$

quadratic function and linear constraints $\rightarrow$ quadratic programming

Large Margin Classifier
Quadratic programming

Lagrangian function $\mathcal{L}_P(\Theta, \Theta_0, \boldsymbol{\alpha})$

$$\mathcal{L}_P(\Theta, \theta_0, \boldsymbol{\alpha}) = \frac{1}{2}||\Theta||^2 + \sum_{i=1}^{n} \alpha_i (1 - y_i(\theta_0 + \Theta^\top \mathbf{x}_i)) \quad (1)$$

Large Margin Classifier
Quadratic programming

$$\min_{\Theta,\theta_0} \max_{\alpha} \mathcal{L}_P(\Theta, \theta_0, \boldsymbol{\alpha}) \tag{2}$$

$$\max_{\alpha} \min_{\Theta,\theta_0} \mathcal{L}_P(\Theta, \theta_0, \boldsymbol{\alpha}) \tag{3}$$

subject to

$$\alpha_i \geq 0, i = 1, \ldots n$$

1. Minimize $\mathcal{L}_P$ w.r.t. Θ
   Therefore differentiate $\mathcal{L}_P$ w.r.t. Θ and $\frac{\partial \mathcal{L}}{\partial \Theta} = 0$. It gives

$$\Theta = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{4}$$

2. Minimize $\mathcal{L}_P$ w.r.t. θ_0
   Therefore differentiate $\mathcal{L}_P$ w.r.t. θ_0 and $\frac{\partial L}{\partial \theta_0} = 0$. It gives

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{5}$$

Large Margin Classifier
Quadratic programming

3. Substitute (4) into the primal form (1) and solve **Wolfe dual optimization problem**

$$\max_{\alpha} \mathcal{L}_D(\boldsymbol{\alpha}) = \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

subject to

$$\alpha_i \geq 0, i = 1 \ldots n$$

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

4. Get $\boldsymbol{\alpha}^\star$
5. Get $\Theta^\star = \sum_{i=1}^{n} \alpha_i^\star y_i \mathbf{x}_i$. Assume that $\mathbf{x}_i$ is a support vector. Then $1 = y_i(\theta_0^\star + {\Theta^\star}^\top \mathbf{x}_i) \rightarrow \theta_0^\star = y_i - {\Theta^\star}^\top \mathbf{x}_i$

Large Margin Classifier
Quadratic programming

- $\boldsymbol{\alpha}^\star$ is the solution to the dual problem
- $\Theta^\star$ is the solution to the primal problem
- the solutions $\boldsymbol{\alpha}^\star$ and $\Theta^\star$ must satisfy the Karush-Kuhn-Tucker conditions where one of them is *KKT dual complementarity*:

$$\alpha_i^\star \cdot (1 - y_i(\theta_0^\star + {\Theta^\star}^\top \mathbf{x}_i)) = 0$$

  - $y_i(\theta_0^\star + {\Theta^\star}^\top \mathbf{x}_i) \neq 1$, i.e., $\mathbf{x}_i$ is not support vector $\Rightarrow \alpha_i^\star = 0$
  - $\alpha_i^\star \neq 0 \Rightarrow y_i(\theta_0^\star + {\Theta^\star}^\top \mathbf{x}_i) = 1$, i.e., $\mathbf{x}_i$ is support vector

I.e., finding $\Theta^\star$ is equivalent to finding support vectors and their weights

Large Margin Classifier
Prediction

Prediction for a new instance $\mathbf{x}$

$$f(\mathbf{x}) = \mathrm{sign}(\sum_{i=1}^{n} \alpha_i^\star y_i \mathbf{x}_i \cdot \mathbf{x} + \theta_0^\star)$$

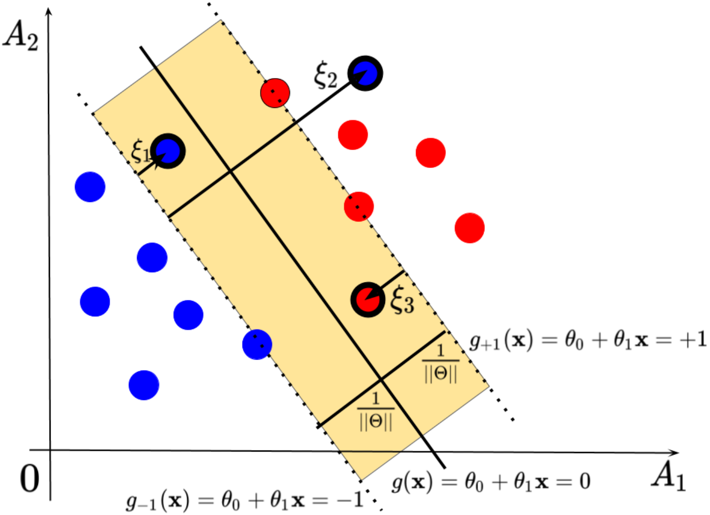
- similarity between $\mathbf{x}$ and support vector $\mathbf{x}_i$: a support vector that is more similar contributes more to the classification
- support vector that is more important, i.e. has larger α_i, contributes more to the classification
- if y_i is positive, than the contribution is positive, otherwise negative

Soft Margin Classifier
Training data is not linearly separable

In a real problem it is unlikely that a hyperplane will exactly separate the data – even if a curved decision boundary is possible. So exactly separating the data is probably not desirable – if the data has noise and outliers, a smooth decision boundary that ignores a few data points is better than one that loops around the outliers.
Therefore

minimize $||\mathbf{\Theta}||^2$ **AND** the number of training mistakes

Soft Margin Classifier
Training data is not linearly separable



Soft Margin Classifier
Slack variables

$\xi_i \geq 0$

- $\xi_i = 0$ if $\mathbf{x}_i$ is correctly classified
- ξ_i is distance to y_i's supporting hyperplane" otherwise
  - margin violation – $0 < \xi_i \leq 1/||\Theta||$, see ξ_1, ξ_3 above
  - misclassification – $\xi_i > 1/||\Theta||$, see ξ_2 above

Soft Margin Classifier
Optimization problem

Minimize

$$\frac{1}{2}||\Theta||^2 + C\sum_{i=1}^{n}\xi_i$$

subject to

$$\xi_i \geq 0, y_i(\theta_0 + \Theta^\top \mathbf{x}_i) \geq 1 - \xi_i, i = 1, \ldots n$$

$C \geq 0$ trade-off parameter

- small $C \Rightarrow$ large margin
  relaxed model; misclassifications are not penalized
- large $C \Rightarrow$ narrow margin
  misclassifications are penalized strongly
  the model will not generalize much

Soft Margin Classifier
Quadratic programming

For each constraint $y_i(\theta_0 + \Theta^\top \mathbf{x}_i) \geq 1 - \xi_i$
introduce Lagrange multiplier $\alpha_i \geq 0$.

Let $\boldsymbol{\alpha} = \langle \alpha_1, ..., \alpha_n \rangle$.

Primal Lagrangian $\mathcal{L}_P(\Theta, \theta_0, \xi, \boldsymbol{\alpha})$ is given by

$$\mathcal{L}_P(\Theta, \theta_0, \xi, \boldsymbol{\alpha}) = \frac{1}{2}||\Theta||^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i (y_i(\theta_0 + \Theta^\top \mathbf{x}_i) - 1 + \xi_i) \quad (6)$$

Soft Margin Classifier
Quadratic programming

Primal Lagrangian problem

$$\min_{\Theta,\theta_0,\xi}\max_{\alpha}\mathcal{L}_P(\Theta,\theta_0,\xi,\boldsymbol{\alpha}) \tag{7}$$

$$\max_{\alpha}\min_{\Theta,\theta_0,\xi}\mathcal{L}_P(\Theta,\theta_0,\xi,\boldsymbol{\alpha}) \tag{8}$$

subject to

$$\alpha_i \geq 0, \xi_i \geq 0, i = 1, \quad \dots \quad , n$$

1. Minimize $\mathcal{L}_P$ w.r.t. Θ. Therefore $\frac{\partial \mathcal{L}}{\partial \Theta} = 0$. It gives

$$\Theta = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{9}$$

2. Minimize $\mathcal{L}_P$ w.r.t. θ_0. Therefore $\frac{\partial L}{\partial \theta_0} = 0$. It gives

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{10}$$

Soft Margin Classifier
Quadratic programming

3. Minimize $\mathcal{L}_P$ w.r.t. ξ. Therefore $\frac{\partial L}{\partial \xi} = 0$. It gives

$$C\xi - \alpha = 0 \tag{11}$$

4. Substitute (9) into the primal form (6) and solve **Wolfe dual opt. problem**

$$\max_\alpha \mathcal{L}_D(\boldsymbol{\alpha}) = \max_\alpha \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to

$$\alpha_i \geq 0, \quad i = 1, \ldots, n$$

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

5. Get $\boldsymbol{\alpha}^\star$
6. Get $\Theta^\star$

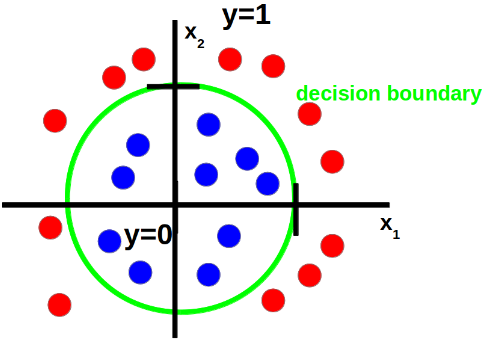
Soft Margin Classifier
Prediction

Prediction for a new instance **x**

$$f(\mathbf{x}) = \mathrm{sign}(\sum_{i=1}^{n} \alpha_i^\star y_i \mathbf{x}_i \cdot \mathbf{x} + \theta_0^\star)$$

Non-linear boundary

If the examples are separated by a nonlinear region



Non-linear boundary

Recall polynomial regression

Polynomial regression is an extension of linear regression where the relationship between features and target value is modelled as a d-th order polynomial.

Simple regression

$y = \Theta_0 + \Theta_1 x_1$

Polynomial regression

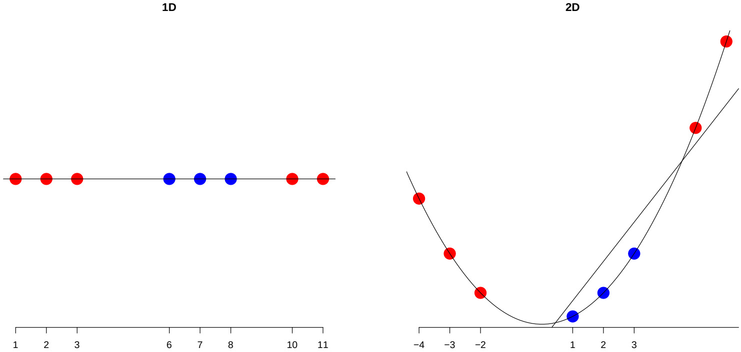
$y = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_1^2 + \dots \Theta_d x_1^d$

It is still a linear model with features $A_1, A_1^2, \dots, A_1^d$.

This defines a feature mapping $\phi(x_1) = [x_1, x_1^2, \dots, x_1^d]$

Non-linear boundary

- for example $\phi(x_1) = [x_1 - 5, (x_1 - 5)^2]$

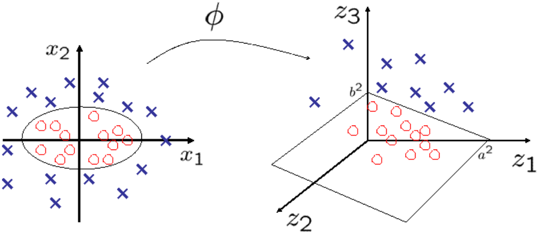


Kernels

Idea

- Apply Large/Soft margin classifier not to the orginal features but to the features obtained by the feature mapping $\phi(\mathbf{x}) : \mathcal{R}^m \rightarrow \mathcal{F}$
- Large/Soft margin classifier uses dot product $\mathbf{x}_i \cdot \mathbf{x}_j$.
  Replace it with $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$.

Kernels



$$\phi : (x_1, x_2) \longrightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \longrightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

Source: `http://omega.albany.edu:8008/machine-learning-dir/notes-dir/ker1/ker1-l.html`

Kernels

However, finding ϕ could be expensive.

Kernel trick

- No need to know what ϕ is and what the feature space is, i.e. no need to explicitly map the data to the feature space
- Define a kernel function $K : \mathcal{R}^m \times \mathcal{R}^m \rightarrow \mathcal{R}$
- Replace the dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ with a Kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$:

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

Kernels
Prediction

Prediction for a new instance **x**

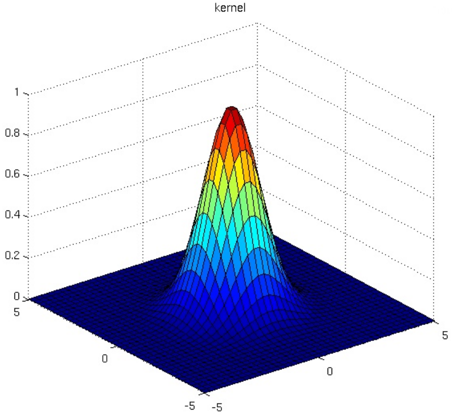
$$f(\mathbf{x}) = \mathrm{sign}(\sum_{i=1}^{n} \alpha_i^\star y_i K(\mathbf{x}_i, \mathbf{x}) + \theta_0^\star)$$

Common Kernel functions

- **Linear**
  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- **Polynomial**
  $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i \cdot \mathbf{x}_j + c)^d$
  – smaller degree can generalize better
  – higher degree can fit (only) training data better
- **Radial basis function**
  $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma(||\mathbf{x}_i - \mathbf{x}_j||^2))$
  – very robust
  – use it when polynomial kernel is weak to fit data
- **Sigmoid**
  $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i \cdot \mathbf{x}_j + c)$, where $\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$

Radial Basis Function Kernel

- $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2}$



Source: `http://www.cs.toronto.edu/~duvenaud/cookbook/index.html`

Evaluation of binary classifiers
Sensitivity vs. specificity

Confusion matrix

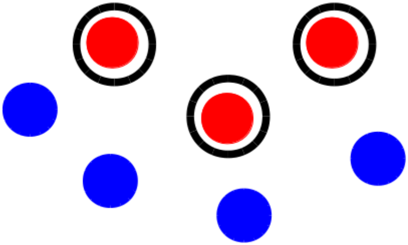
		Predicted class	
		Positive	**Negative**
True class	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Measure	**Formula**
Precision	TP/(TP+FP)
Recall/Sensitivity	TP/(TP+FN)
Specificity	TN/(TN+FP)
1-Specificity	FP/(TN+FP)
Accuracy	(TP+TN)/(TP+FP+TN+FN)

Evaluation of binary classifiers
Sensitivity vs. specificity

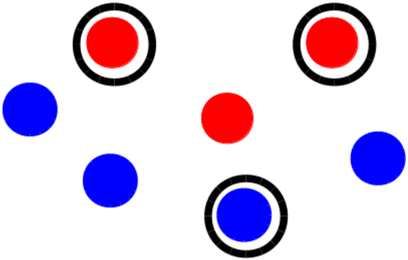
Seven training examples

Perfect classifier – no error



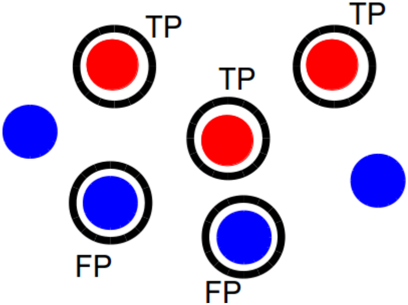
Evaluation of binary classifiers
Sensitivity vs. specificity

Reality – e.g. 1 error



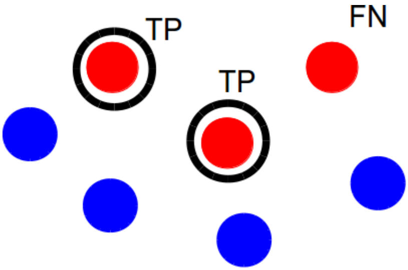
Evaluation of binary classifiers
Sensitivity vs. specificity

100% sensitive classifier



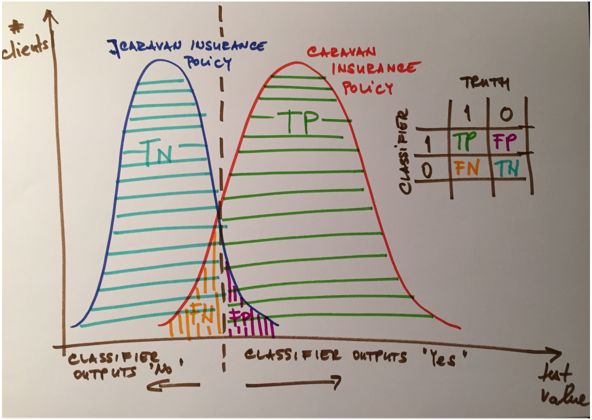
Evaluation of binary classifiers
Sensitivity vs. specificity

100% specific classifier



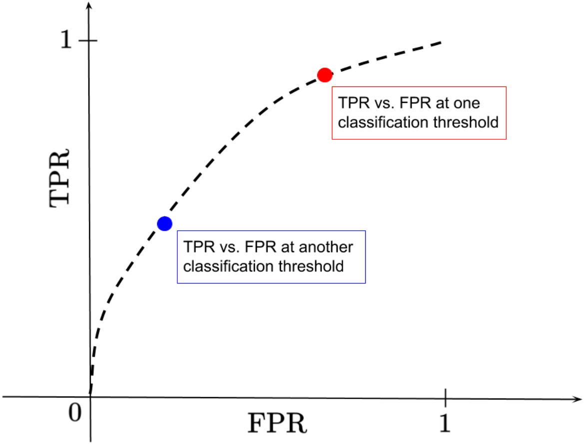
Evaluation of binary classifiers
Sensitivity vs. specificity

Sensitivity vs. specificity



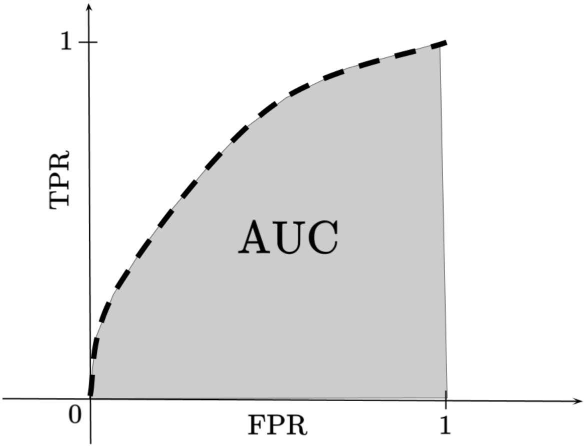
Evaluation of binary classifiers
ROC curve

An **ROC curve** plots True Positive Rate vs. False Positive Rate at different classification thresholds (see p. 6).



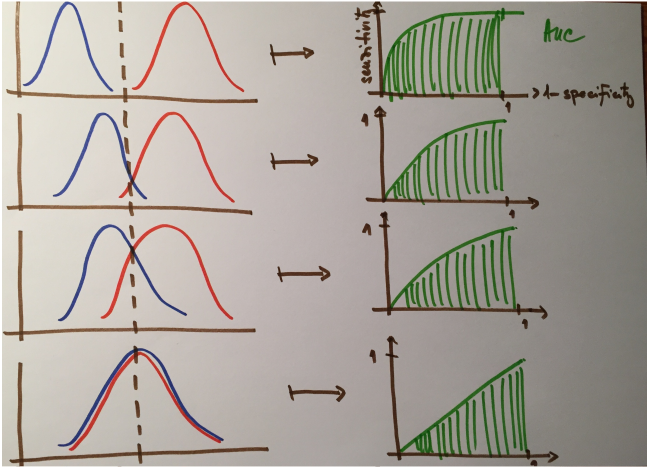
Evaluation of binary classifiers
AUC measure

Area Under ROC (= AUC)
is a measure of how good is a distinguishing property of classifier



Evaluation of binary classifiers
ROC & AUC

Curves closer to the top-left corner indicate a better performance.



Summary of Examination Requirements

- Hyperplane, margin, functional margin, geometric margin of example and data set
- Large margin classifier
  linearly separable data, supporting hyperplanes, support vectors, optimization task, prediction function
- Soft margin classifier
  not linearly separable data, supporting hyperplanes, support vectors, slack variables, optimization task, hyperparameter C, prediction function
- Kernel trick
  feature mapping, Kernel functions, prediction function
- Binary classifier evaluation
  sensitivity, specificity, ROC curve, AUC