COMPUTATIONAL INTELLIGENCE

(INTRODUCTION TO MACHINE LEARNING) SS16

Lecture 6:

- Support Vector Machine (SVM)
- Kernel methods
- Multiclass classification

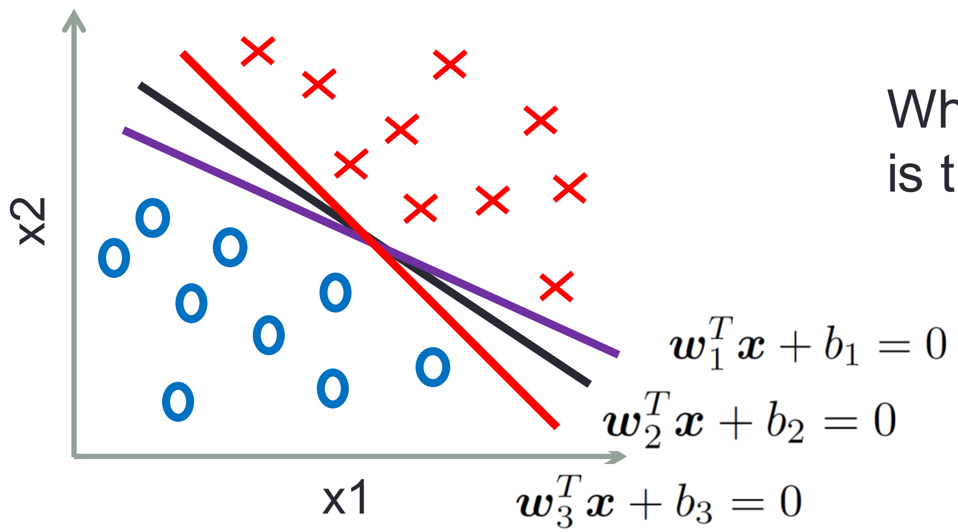
SUPPORT VECTOR MACHINE (SVM)

Separation of linearly separable classes

- Consider a training set consisting of *m* samples:

$$\langle \boldsymbol{x}^{(1)}, y^{(1)} \rangle \ldots \langle \boldsymbol{x}^{(m)}, y^{(m)} \rangle \quad \text{where} \quad y^{(i)} \in \{-1, 1\}$$

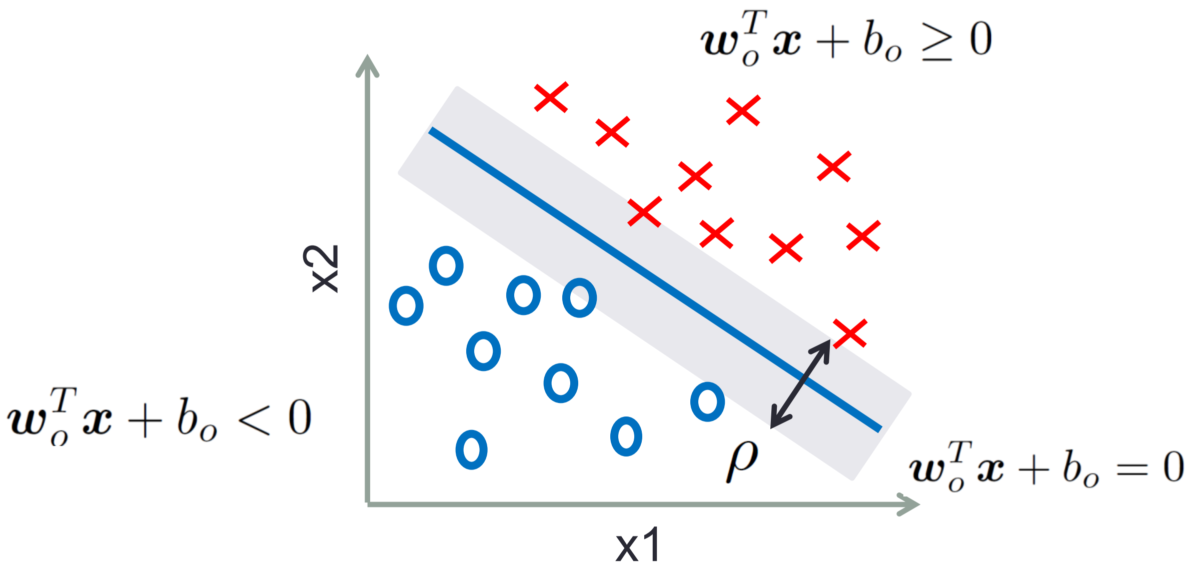
- If samples are linearly separable, there are multiple possible decision boundaries or separation hyperplanes



Which one is the best?

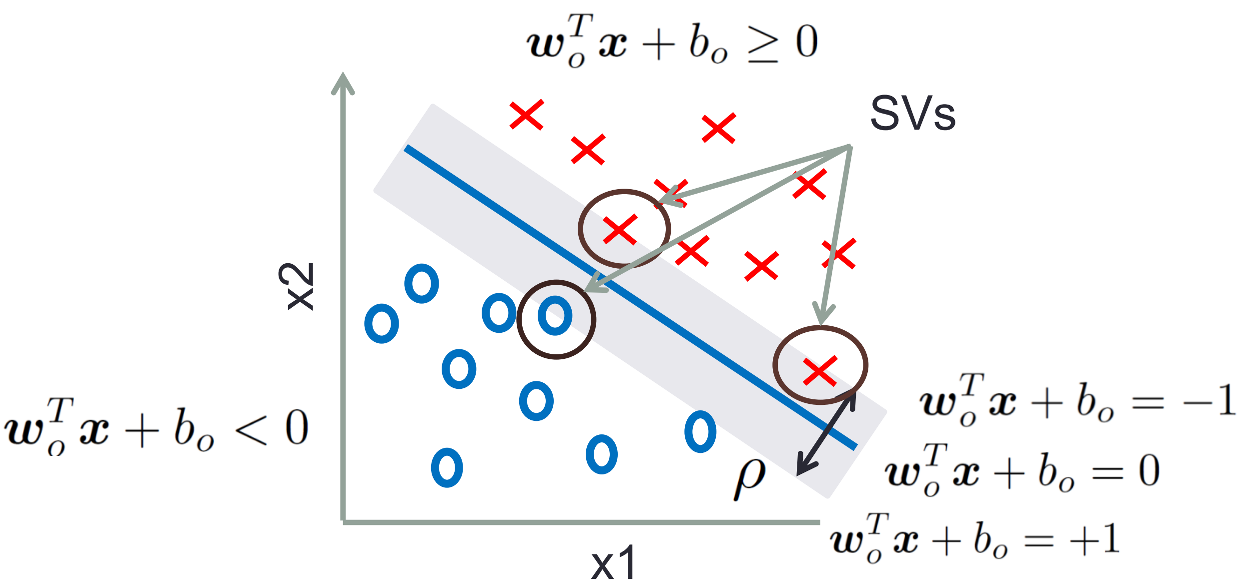
Margin of separation

- SVM tries to find an optimal decision boundary (hyperplane) determined with $\boldsymbol{w}_o$ and b_o, for separation of two classes which maximizes the separation margin or the separation between classes – the region between classes without samples



Support vectors

- Support vectors(SVs) are:
  - the closest points(samples) to the separation hyperplane
  - used for definition of the optimal separation hyperplane



- SVs are samples for which holds:

$$\boldsymbol{w}_o^T\boldsymbol{x}^{(i)} + b_o = \pm 1$$

Separation hyperplane

- The separation hyperplane is given by

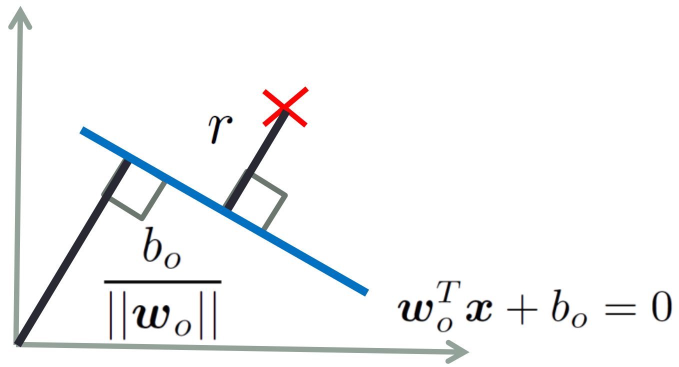
$$\boldsymbol{w}_o^T \boldsymbol{x} + b_o = 0$$

- Discrimination function:

$$h(\boldsymbol{x}) = \boldsymbol{w}_o^T \boldsymbol{x} + b_o$$

The class of a new sample is determined based on the sign of $h(\boldsymbol{x})$

- Distance of the sample from the separation hyperplane:



The distance is: $r = \frac{h(\boldsymbol{x})}{||\boldsymbol{w}_o||}$

$||\boldsymbol{w}_o||$ is Euclidean norm

$$||\boldsymbol{w}|| = \sqrt{\sum_i w_i^2}$$

Choice of support vectors

- Scaling of $||\boldsymbol{w}_o||$ and b_o does not change the separation hyperplane
- Therefore, SVs are chosen such that:

$$h(\boldsymbol{x}^{(s)}) = \boldsymbol{w}_o^T \boldsymbol{x}^{(s)} + b_o = \pm 1 \qquad \text{for} \qquad y^{(s)} = \pm 1$$

- The distance of SV from the hyperplane is:

$$r = \frac{h(\boldsymbol{x}^{(s)})}{||\boldsymbol{w}_o||} = \frac{\pm 1}{||\boldsymbol{w}_o||}$$

- The width of the resulting margin is then:

$$\rho = 2|r| = \frac{2}{||\boldsymbol{w}_o||}$$

Maximizing the margin of separation

- Maximizing the margin is equivalent to minimizing the $||\boldsymbol{w}||$

- The $||\boldsymbol{w}||$ norm involves the square root, so minimization of $||\boldsymbol{w}||$ is replaced by minimization of $\frac{1}{2}||\boldsymbol{w}||^2$, which does not change the solution

- SVM finds the *maximum margin hyperplane*, the hyperplane that maximizes the distance from the hyperplane to the closest training point

Optimization

- Optimization problem can be written as:

$$arg\min_{\boldsymbol{w}} \frac{1}{2}||\boldsymbol{w}||^2$$

under condition for all samples (that all of them are correctly classified):

$$y^{(i)}(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b) \geq 1 \qquad \text{for} \quad i = 1 \ldots m$$

- This problem can be solved by using Lagrange multipliers:

$$J(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \underbrace{\frac{1}{2}||\boldsymbol{w}||^2}_{\textit{min}} - \underbrace{\sum_{i=1}^{m} \alpha_i[y^{(i)}(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b) - 1]}_{\textit{max}}$$

where Lagrange multipliers $\alpha_i \geq 0$

- Solution is in the saddle $\min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha}} J(\boldsymbol{w}, b, \boldsymbol{\alpha})$

Solution

- To find Lagrange multipliers α_i the dual form is used, which is solved through quadratic optimization

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} \boldsymbol{x}^{T(i)} \boldsymbol{x}^{(j)}$$

under conditions: $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$ and $\alpha_i \geq 0$

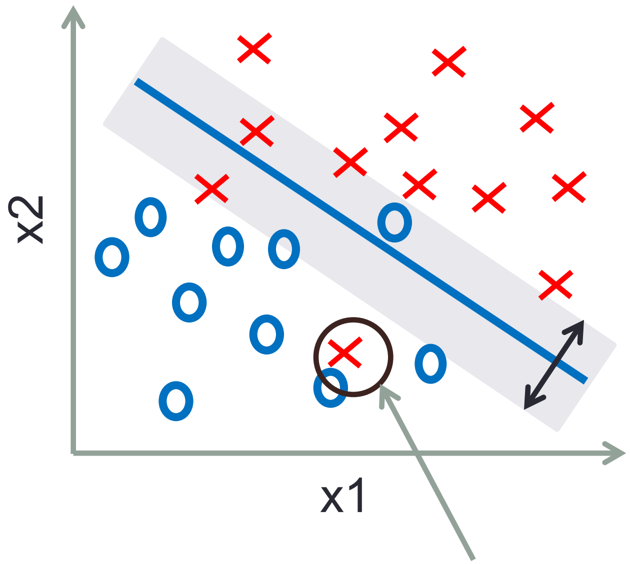
- The solution can be expressed as a linear combination of training vectors

$$\boldsymbol{w}_o = \sum_{i=1}^{m} \alpha_i y^{(i)} \boldsymbol{x}^{(i)} \qquad b_o = y^{(i)} - \boldsymbol{w}_o^T \boldsymbol{x}^{(i)}$$ (for some SV)

- Note that only few of α_i will be greater then 0, and for those the corresponding samples will be support vectors!

Separation of linearly non-separable classes (Soft margin method)

- Main idea: use a soft margin which allows for mislabeled samples
- Introduce slack variables ξ_i and solve slightly different optimization problem



Regularization term

$$arg\min_{\boldsymbol{w}} \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{m}\xi_i$$

$$y^{(i)}(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

- The free parameter C controls the relative importance of minimizing the norm $||\boldsymbol{w}||$ and satisfying the margin constraint for each sample
- It allows to control the sensitivity of SVM to outliers

SVM: pros and cons

- SVM is not necessarily better than other machine learning methods (except perhaps in situations with little training data), but it performs at the state-of-the-art level and has a nice theoretical background

- **Pros**:
  - Finding a minimum of optimization problem is guaranteed
  - Usage of kernel methods (solve nonlinear problems)
  - By choosing a specific hyperplane among many SVM avoids overfitting (this depends on the choice of parameter C)

- **Cons**:
  - Speed of execution – no direct control of number of SVs
  - Solution parameters are hard to interpret
  - Hard to add a priori knowledge:
    - Solutions: add “artificial” samples or add additional optimization conditions

SVM extensions

- Multiclass SVM (multiple classes)
- Transductive SVM (partially labeled data, transduction)
- Structured SVM (structured output labels)
    - E.g. Input is natural language sentence, output is annotated parse tree
- Regression (Support Vector Regression)

SVM applications

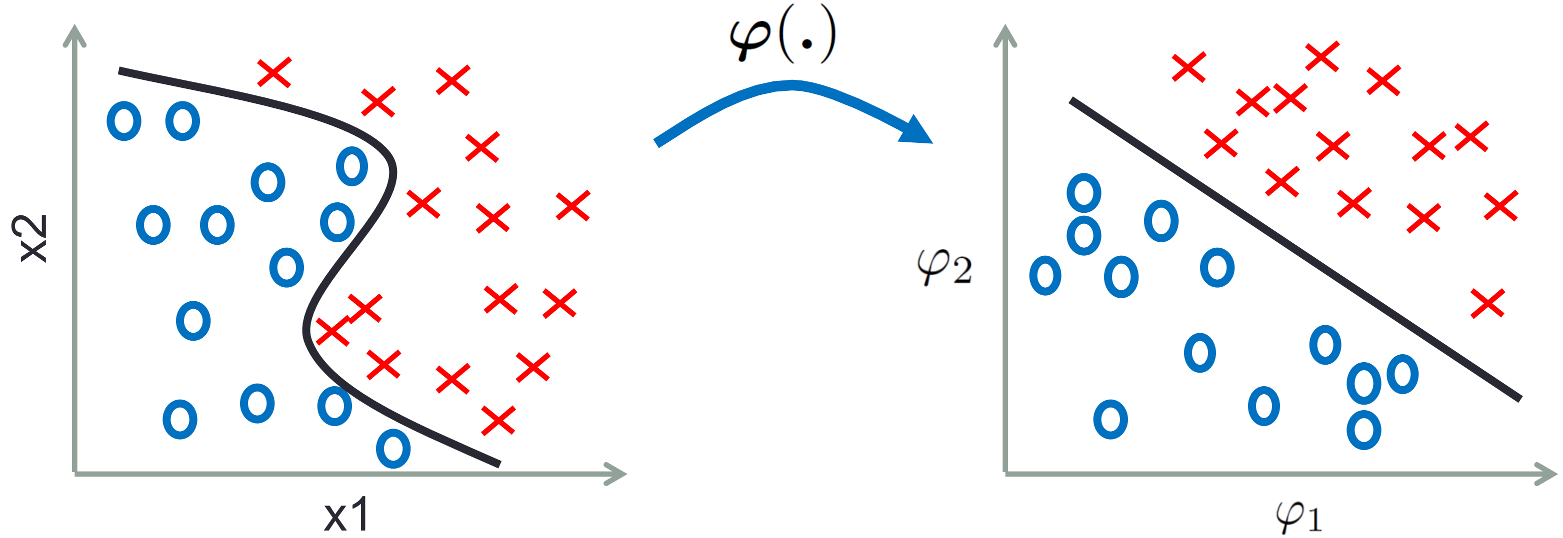
- SVMs are used to solve many real world problems:
  - Text classification
  - Image classification
  - Hand-written recognition
  - Protein classification
  - ...

KERNEL METHODS

Motivation

- If data samples are not linearly separable, it would be nice if we could make them linearly separable and apply well studied linear classifiers (e.g. SVM) to separate them

- **Cover's theorem:** given a set of training data that is not linearly separable, one can with high probability transform it into a training set that is linearly separable by projecting it into a higher dimensional space via some non-linear transformation



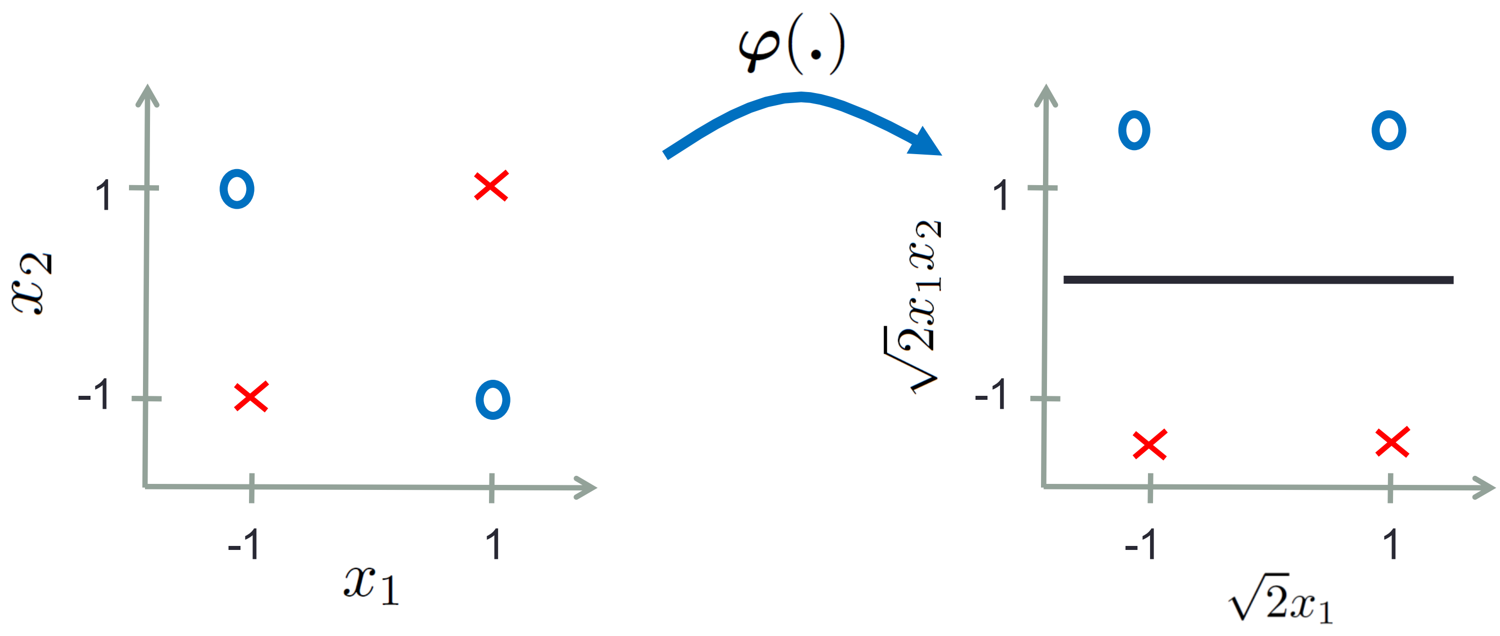
Projection example

Original space:

$$\boldsymbol{x} = (x_1, x_2)$$

High-dimensional feature space:

$$\boldsymbol{\varphi}(\boldsymbol{x}) = (1, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2)$$



Kernels

- Kernels are functions that return the inner products between the images of data points in some space (they are often interpreted as a similarity measures)

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\varphi}(\boldsymbol{x}_1)^T \boldsymbol{\varphi}(\boldsymbol{x}_2)$$

- They allow for operating in a high-dimensional *implicit* feature space without ever computing the coordinates of the data in that space - no need for explicit mapping!

- This operation is often computationally cheaper than the explicit computation and is called the **kernel trick**

Kernel example

- Consider 2-dimensional vectors: $\boldsymbol{u} = (u_1 \quad u_2)^T$ and $\boldsymbol{v} = (v_1 \quad v_2)^T$

  and a quadratic kernel in two dimensions:

$$K(\boldsymbol{u}, \boldsymbol{v}) = (1 + \boldsymbol{u}^T \boldsymbol{v})^2$$

If $\boldsymbol{\varphi}(\boldsymbol{x}) = (1 \quad x_1^2 \quad \sqrt{2}x_1 x_2 \quad x_2^2 \quad \sqrt{2}x_1 \quad \sqrt{2}x_2)^T$ then:

$$\begin{aligned} K(\boldsymbol{u}, \boldsymbol{v}) &= (1 + \boldsymbol{u}^T \boldsymbol{v})^2 \\ &= 1 + u_1^2 v_1^2 + 2u_1 v_1 u_2 v_2 + u_2^2 v_2^2 + 2u_1 v_1 + 2u_2 v_2 \\ &= (1 \quad u_1^2 \quad \sqrt{2}u_1 u_2 \quad u_2^2 \quad \sqrt{2}u_1 \quad \sqrt{2}u_2)^T (1 \quad v_1^2 \quad \sqrt{2}v_1 v_2 \quad v_2^2 \quad \sqrt{2}v_1 \quad \sqrt{2}v_2) \\ &= \boldsymbol{\varphi}(\boldsymbol{u})^T \boldsymbol{\varphi}(\boldsymbol{v}) \end{aligned}$$

Kernels usage

- *Where*:
  - Within learning algorithms that only require dot products between the vectors in the original space (choose the mapping such that the high-dimensional dot products can be computed within the original space, by means of a *kernel function)*
- *How*:
  - Calculate *the kernel matrix* – the inner product between all pairs of data samples
- *Under what condition*:
  - Given by ***Mercer's theorem***: the kernel matrix must be symmetric positive definite

Standard kernels

- Polynomial kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T \boldsymbol{y} + c)^d$$

- RBF kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = exp(-\frac{||\boldsymbol{x} - \boldsymbol{y}||^2}{2\sigma^2})$$

- Sigmoid kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = tanh(a\boldsymbol{x}^T \boldsymbol{y} + b)$$

- String kernels
- Graph kernels
- …

Kernels with various methods

- Any linear model can be turned into a non-linear model by applying the "kernel trick" to the model: replacing its features by a kernel function

- Methods capable of operating with kernels:
  - SVM
  - Perceptron (Kernel perceptron)
  - Principal component analysis
  - Cluster analysis
  - Gaussian process
  - Fisher discriminant
  - ...

Nonlinear (kernel) SVM

- The most famous application of kernels is with SVM
- Kernels allow non-linear classification with SVMs

- The separation hyperplane can be rewritten in the feature space as:

$$\boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x}) = 0 \qquad \text{where we used} \qquad \varphi_0(\boldsymbol{x}) = 1 \quad \text{and} \quad w_0 = b_o$$

- The solution to the optimization problem of the maximal margin can be written as:

$$\boldsymbol{w}_o = \sum_{i=1}^{m} \alpha_i y^{(i)} \boldsymbol{\varphi}(\boldsymbol{x}^{(i)})$$

- Combining these gives:

$$\sum_{i=1}^{m} \alpha_i y^{(i)} \boldsymbol{\varphi}(\boldsymbol{x}^{(i)})^T \boldsymbol{\varphi}(\boldsymbol{x}) = 0 \quad \rightarrow \quad \sum_{i=1}^{m} \alpha_i y^{(i)} K(\boldsymbol{x}^{(i)}, \boldsymbol{x}) = 0$$

Applications

- Kernels can be applied on general types of data (besides vector data also on sequences, trees, graphs, etc.)

- Application areas:
  - Information extraction
  - Bioinformatics
  - Handwriting recognition
  - Text classification (string kernels)
  - 3D reconstruction
  - Image recognition
  - ...

MULTICLASS CLASSIFICATION

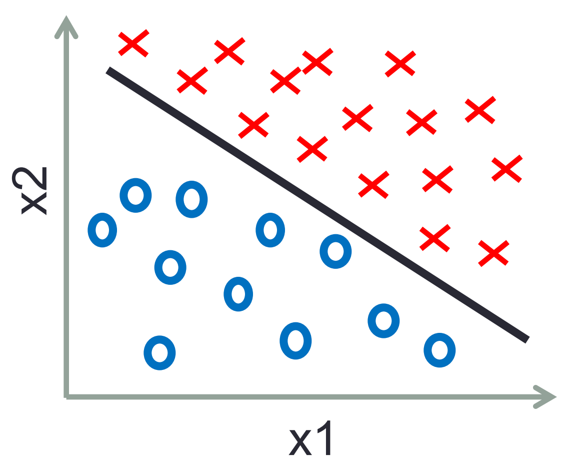
Classification problems

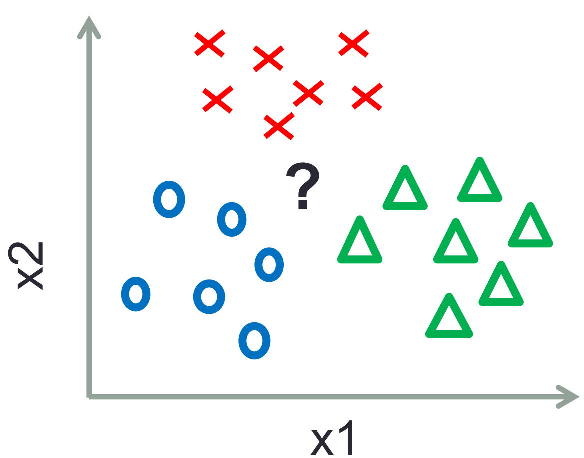
Some are naturally **binary:**

- Spam vs not spam
- Medical tests
- Quality control
- ...

But many are **mutli-class**:

- Text classification
- POS tagging
- Object recognition
- Biological sequences
- ...





Multiclass classification

- Consider a training set consisting of *m* samples:

$$\langle \boldsymbol{x}^{(1)}, y^{(1)} \rangle \ldots \langle \boldsymbol{x}^{(m)}, y^{(m)} \rangle$$

- Each training sample belongs to **only one** of the N classes

$$y^{(i)} \in [1, \ldots, N]$$

- The goal is to find a function which correctly predicts the class to which a new sample belongs

- It is different from a multilabel classification, where the goal is to assign to each sample a set of target labels (multiple classes)!

Classifiers

Some are directly multiclass:

- Decision trees
- Naive Bayes
- MaxEnt
- Multiclass SVM
- AdaBoost.MH

They directly output more than two class labels

Many are binary:

- Logistic regression
- Perceptron
- Neural Network
- SVM

They output only 2 class labels (e.g. 0 and 1). Can we use them for multiclass problems and how?

Binary classifiers for multiclass problems

- Idea:
    - Decompose multiclass problem into a set of binary problems
    - Create binary classifiers for binary problems
    - Combine the output of binary classifiers as a multiclass classifier

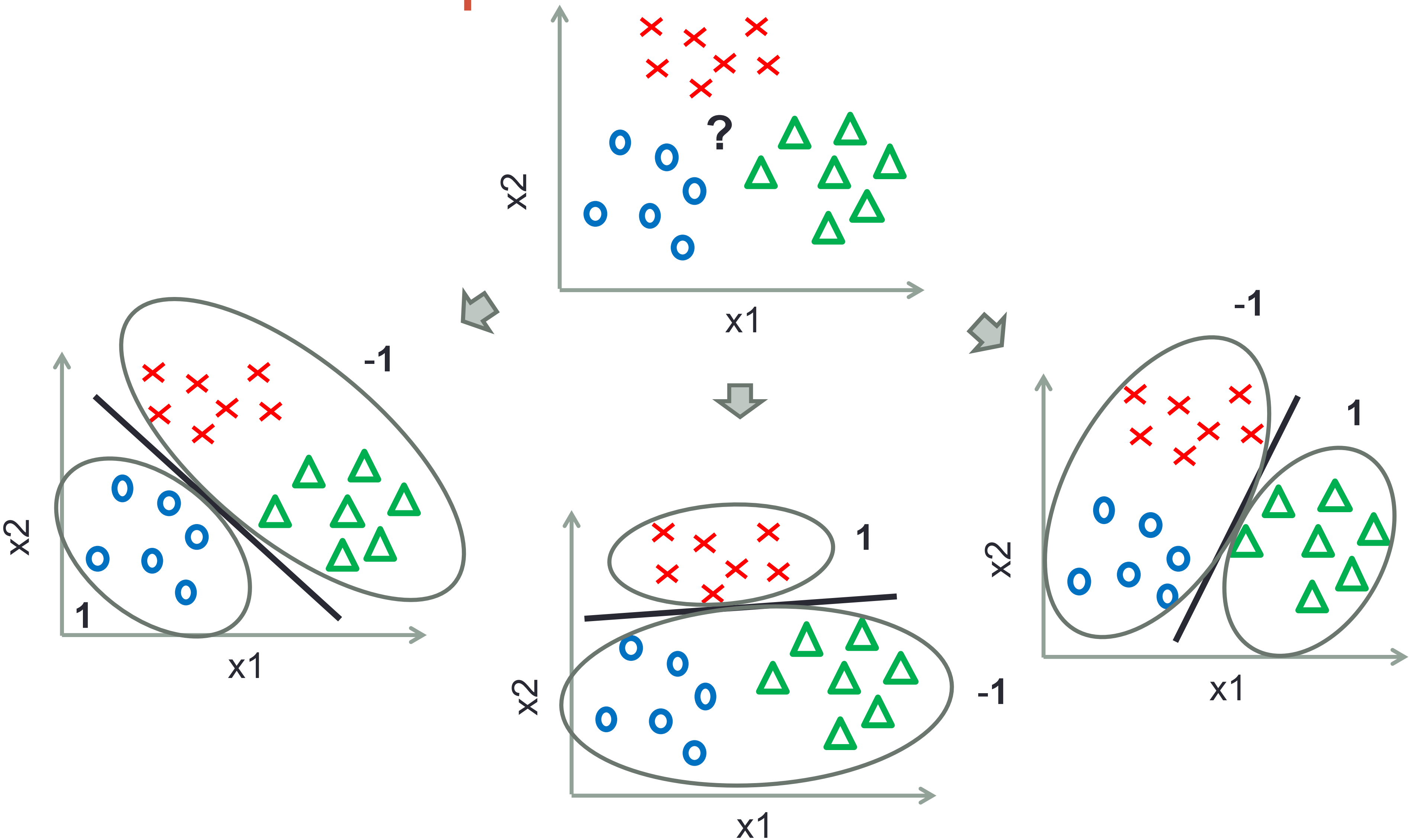
- Methods:
    - One-vs-all (OVA)
    - One-vs-one (OVO)
    - Error Correcting Output Codes (ECOC)

One-vs-all (OVA)

- Create classifiers that distinguish each class from all other classes
- There is 1 classifier per class: N classes $\rightarrow$ N classifiers
- Mapping from a multiclass training set to binary training sets:
  - For each class C :
    - For each sample $\langle \boldsymbol{x}^{(i)}, y^{(i)} \rangle$:
      - If $y^{(i)} = C$ create sample $\langle \boldsymbol{x}^{(i)}, 1 \rangle$
      - Otherwise, create sample $\langle \boldsymbol{x}^{(i)}, -1 \rangle$
- Train N classifiers $h_k(\boldsymbol{x})$ where $k \in [1, \ldots, N]$
- Testing:
  - Classify new sample using all classifiers
  - Select the prediction (class) with the highest confidence score

$$\text{Class} = arg \max_k h_k(\boldsymbol{x}^{(i)})$$

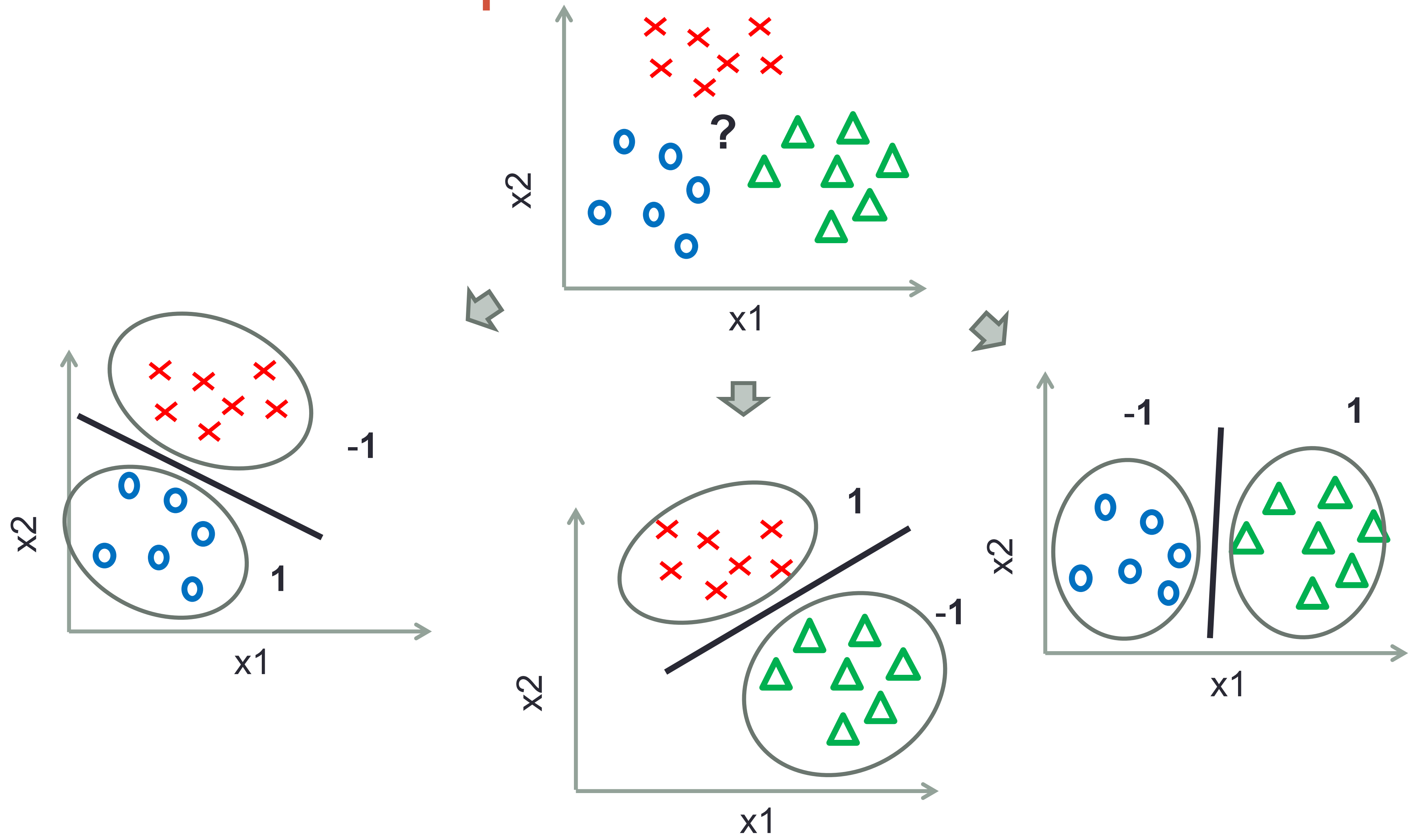
OVA: example



One-vs-one (OVO)

- Create classifiers that distinguish between each pair of classes
- There are for N classes $\rightarrow N(N-1)/2$ classifiers
- Mapping from a multiclass training set to binary training sets:
  - For each class C_i:
    - For each class C_j :
      - For each sample $\langle \boldsymbol{x}^{(l)}, y^{(l)} \rangle$:
        - If $y^{(l)} = C_i$ create sample $\langle \boldsymbol{x}^{(l)}, 1 \rangle$
        - If $y^{(l)} = C_j$ create sample $\langle \boldsymbol{x}^{(l)}, -1 \rangle$
        - Otherwise, ignore sample
- Train all the classifiers
- Testing:
  - Classify new sample using all classifiers
  - Select the class with most votes

OVO: example



Error Correcting Output Codes (ECOC)

- Each class is represented by a binary code of length *n* (e.g. NN)
- Each bit position corresponds to the output of a classifier (feature)

- Training: 1 classifier per bit position
- Testing: get the output from classifiers and find the closest binary code (distance: Euclidean, cosine, Manhattan, etc.) to decide the class

- ECOC can recover from some bit errors (caused by limited data, bad features etc.), but this can also be limited due to the correlated mistakes

Comparison

- The most used (and the simplest) method is OVA
- Complexity (the number of classifiers):
  - OVA: N
  - OVO: $N(N-1)/2$
  - ECOC: *n* (code length)
- OVO can be faster then OVA (due to the smaller datasets), but can have problem with overfitting (too few samples per dataset)

Confusion matrix

- Is an important tool for visualizing and analyzing the performance of a classifier for multiple classes

- It shows for each pair of classes how many samples were incorrectly assigned. From this it is easy to see if the classifier is confusing two classes

- It can help pinpoint opportunities for improving the accuracy of the system
  - e.g. one can easily identify the place(classifier) of largest error and try to introduce additional features to improve classification and reduce the error

		Predicted class		
		cow	motorbike	car
Actual class	cow	6	0	3
	motorbike	1	7	0
	car	4	1	7

Classification metrics

- Classification accuracy
  - Proportion of samples that were classified correctly
  - (No. of samples that were classified correctly) / N

- Classification error
  - Proportion of samples that were classified **in**correctly
  - (No. of samples that were classified **in**correctly) / N

SUMMARY (QUESTIONS)

Some questions…

- What is the margin of separation?
- What are support vectors?
- What is SVM?
- What is the separation hyperplane and the discrimination function?
- What is a distance of a sample from the hyperplane?
- How is the margin of separation maximized?
- Why do we use soft margin?

- What is kernel?
- State Cover's theorem
- What is the kernel trick?
- Condition for kernel matrix?
- Name few standard kernels

Some questions…

- Multiclass vs multilabel classification
- Methods for multiclass problems
- What is OVA?
- OVA vs OVO
- What is ECOC?
- What is confusion matrix and why do we use it?