IN 5520 – Support Vector Machine Classifiers (SVM)

Anne Solberg (anne@ifi.uio.no)

21.10.20

- Linear classifiers with maximum margin for two-class problems
- The kernel trick – from linear to a high-dimensional generalization
- Generation from 2 to M classes
- Practical issues/Parameter estimation

Mandatory 2 out now

- Deadline November 9
- Any questions??
- Remark: png-file converions will be updated

Curriculum

- Lecture foils are most important!
- The lectures are based on selected sections from Pattern Recognition, Third Edition, by Theodoridis and Koutroumbas:
  - 3.1-3.2, 3.7 (but 3.7.3 is a SVM-variant that we will skip)
  - 4.17
  - These sections use optimization theory described in Appendic C. We only include enough mathematics to state the optimization problem, and you are not required to understand how this optimization is solved.
  - Another useful note: Andrew Ng’s note http://cs229.stanford.edu/notes/cs229-notes3.pdf

From last week: Case 1: $\Sigma_j=\sigma^2 I$

- An equivalent formulation of the discriminant functions:

$$g_i(\mathbf{x}) = \mathbf{w}_i^t\mathbf{x} + w_{i0}$$

$$\text{where } \mathbf{w}_i = \frac{1}{\sigma^2}\boldsymbol{\mu}_i \text{ and } w_{i0} = -\frac{1}{2\sigma^2}\boldsymbol{\mu}_i^t\boldsymbol{\mu}_i + \ln P(\omega_i)$$

- An equation for the decision boundary $g_i(\mathbf{x})=g_j(\mathbf{x})$ can be written as

$$\mathbf{w}^t(\mathbf{x}-\mathbf{x}_0)=0$$

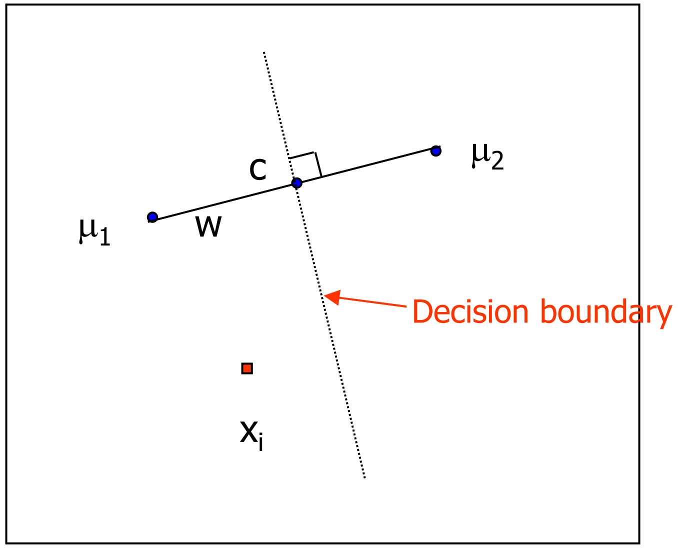
$$\text{where } \mathbf{w} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j$$

$$\text{and } x_0 = \frac{1}{2}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) - \frac{\sigma^2}{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2}\ln\frac{P(\omega_i)}{P(\omega_j)}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$$

- **w**$=\mu_i-\mu_j$ is the vector between the mean values.
- This equation defines a hyperplane through the point x_0, and orthogonal to **w**.

Gaussian model with $\Sigma=\sigma I$ Linear decision boundary

- We found that the discriminant function (when $\Sigma_j=\sigma^2 I$) that defines the border between class 1 and 2 in the feature space is a straight line.
- The discriminant function intersects the line connecting the two class means at the point $c=(\mu_1- \mu_2)/2$ (if we do not consider prior probabilities).
- The discriminant function will also be normal to the line connecting the means.



Reminder: Linear algebra basics: Inner product between two vectors.

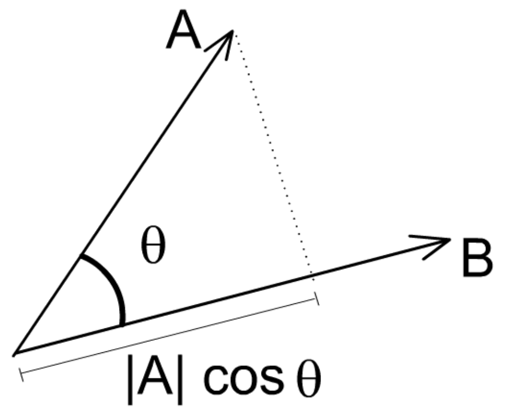
- The inner product (or dot product) between two vectors (of length N) a and b or is given by

$$\langle a,b\rangle = \sum_{i=1}^{N} a_i b_i = a^T b$$

- The angle between two vectors A and B is defined as:

$$\cos\theta = \frac{\langle A,B\rangle}{\|A\|\|B\|}$$

- If the inner product of two vectors is zero, they are normal to each other.
- If two vectors are similar (small angle), cos θ will be close to 1 : the inner product measures similarity



Introduction to Support Vector Machine classifiers

- To understand Support Vector Machine (SVM) classifiers, we need to study the linear classification problem in detail.
- We will need to see how classification using this (the linear case on the previous slide) can be computed using inner products.
- We start with two linearly separable classes.
- Extension to two non-separable classes.
- Extension to M classes.
- The last step will be to use kernels to separate classes that cannot be separated in the input space.

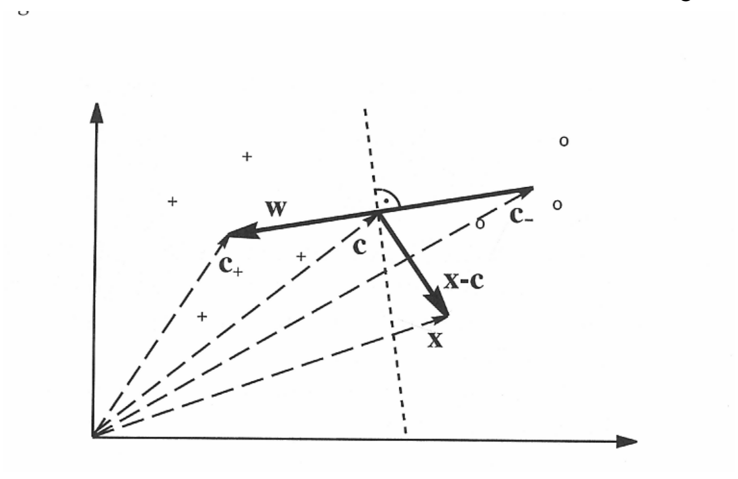
A new view at linear Gaussian classification using inner products

- We have two classes (+ and -) represented by the class means c_+ and c_-.
- Let the feature vector be x_i, and let y_i be the class of feature vector i.
- If we have m_+ samples from class + and m_- samples from class -, the class means are given by

$$c_+ = \frac{1}{m_+} \sum_{\{i|y_i=+)} x_i$$

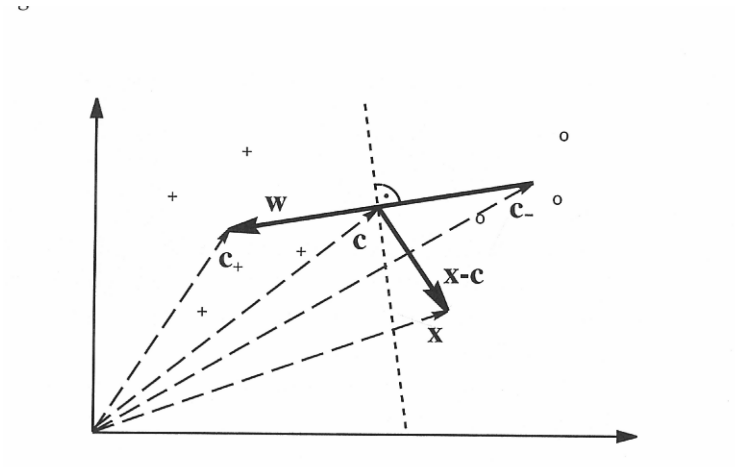
$$c_- = \frac{1}{m_-} \sum_{\{i|y_i=-)} x_i$$

- A new pattern x should be classified to the class with the closest mean.



- Half way between the means lies the point $c=(c_+ + c_-)/2$.

  If a point x is on the decision boundary hyperplane, then $w^T x=0$.



We can compute the class of a new sample x by checking whether the vector x-c (connecting x to c) encloses an angle smaller than $\pi/2$ (in terms of absolute value) with the vector $w=c_+ - c_-$.

This angle is given by the inner product between w and x-c: $w^T(x-c)$

- The angle between two vectors is computed by the inner product, which changes sign as the angle passes through $\pi/2$.

Support Vector Machines
Two linear separable classes

- Let x_i, i=1,...N be all the l-dimensional feature vectors in a training set with N samples.
- These belong to one of two classes, ω_1 and ω_2.
- We assume that the classes are **linearly separable**.
- This means that a hyperplane

  $g(x)=w^Tx+w_0=0$

  correctly classifies all these training samples.
- $w=[w_1,...w_l]$ is called a weight vector, and w_0 is the threshold.

Introduction to linear SVM

- Discriminant function:
  $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$

  Weights/orientation

  Threshold/bias

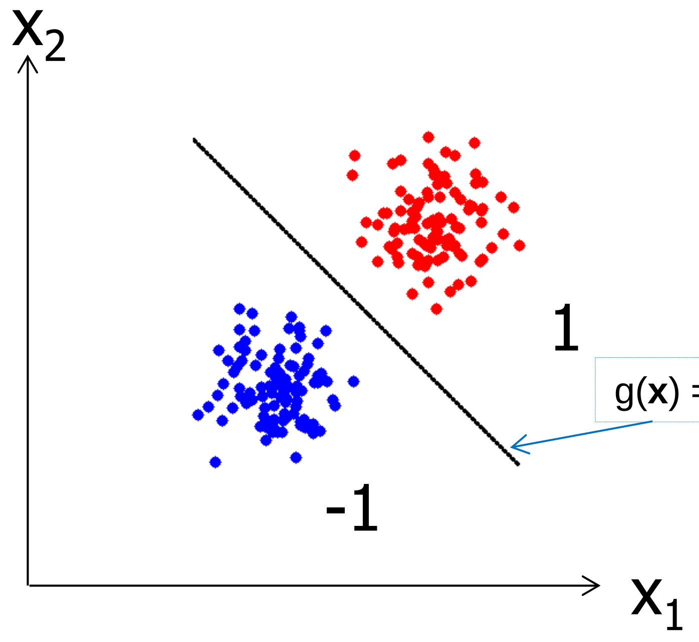
- Two-class problem,
  $y_i \in \{-1,1\}$

  Class indicator for pattern *i*

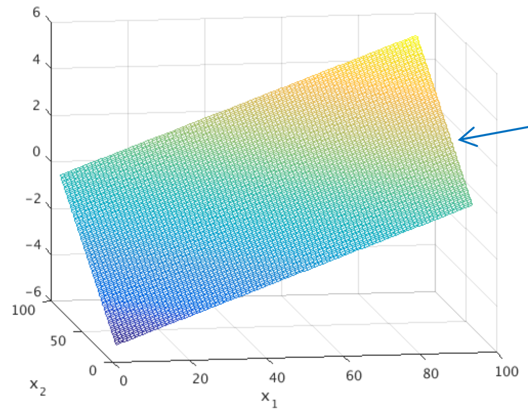
- $y_i = \begin{cases} -1, & g(\mathbf{x}_i) < 0 \\ 1, & g(\mathbf{x}_i) > 0 \end{cases}$

  Class prediction

  Input pattern



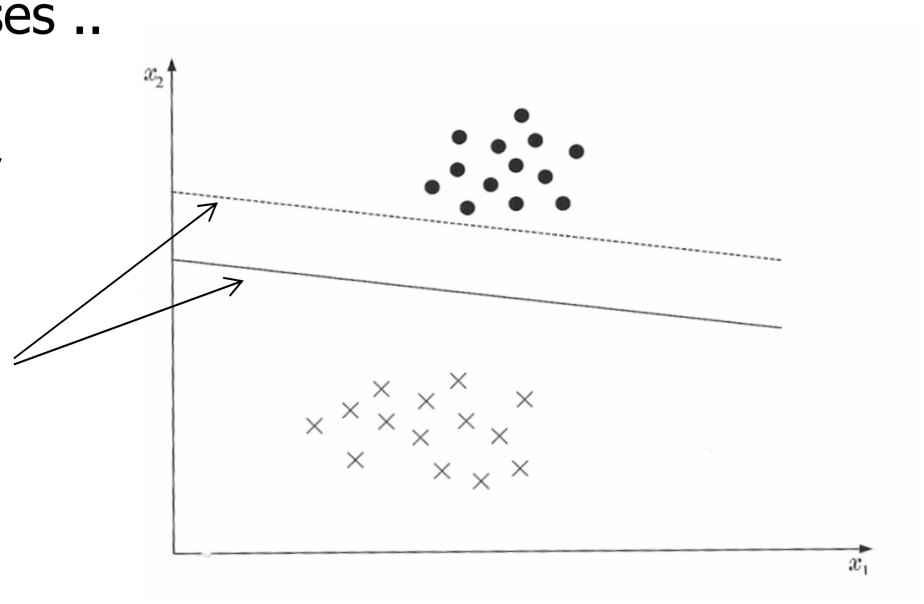
$g(\mathbf{x}) = 0$



$g(\mathbf{x})$

Separable case: Many candidates

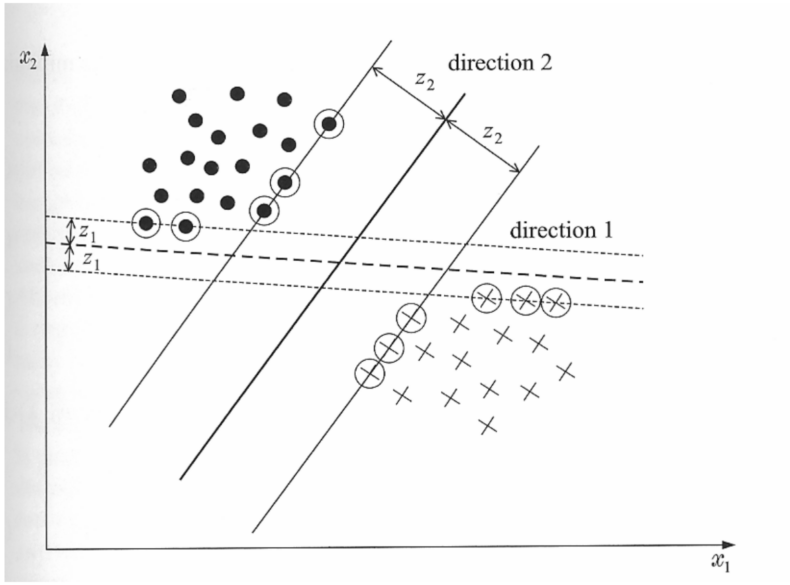
- Obviously we want the decision boundary to separate the classes ..
- .. however, there can be many such hyperplanes.
- Which of these two candidates would you prefer? Why?



Hyperplanes and margins

- If both classes are equally probable, the **distance from the hyperplane to the closest points** in both classes should be equal. This is called the margin.

- SVM finds the decision boundary that maximize the distance from the closest points in both classes to the boundary.



Goal: Find **w** and w_0 maximizing the margin!

How would you write a program finding this? Not easy unless we state the objective function cleverly!

Distance to the decision boundary,z

- Since w/||w|| is a unit vector in the direction w,

  B=x-z*w/||w||

- Because B lies on the decision boundary,

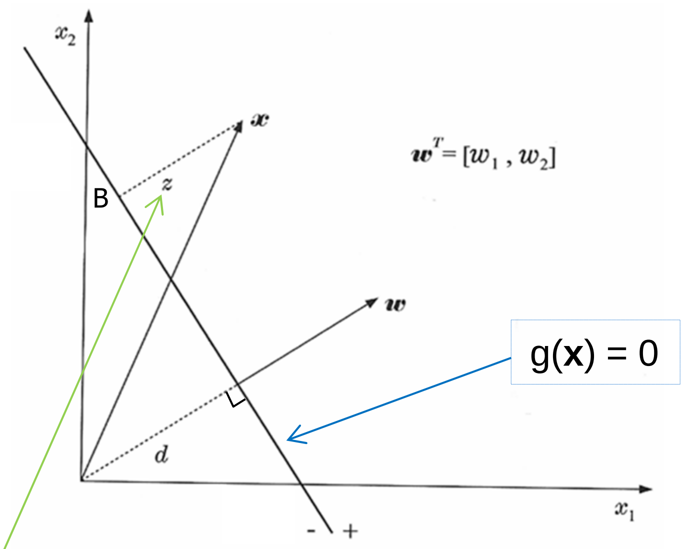
  $w^T B + w_0 = 0$

$$w^T\left(x - z\frac{w}{\|w\|}\right) + w_0 = 0$$

Solve this for z :

$$\mathrm{z} = \frac{\mathrm{w}^{\mathrm{T}} x + w_0}{\|w\|} = \left(\frac{w}{\|w\|}\right)^T x + \frac{w_0}{\|w\|}$$

z is called the **margin** of the classifier



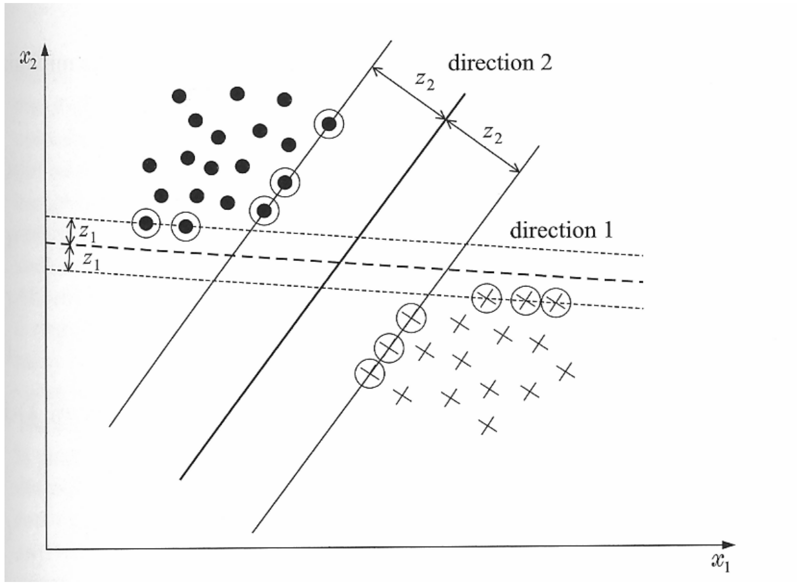
Distance from **x** to the decision boundary

Hyperplanes and margins

- The margin for «direction 1» is $2z_1$, and for «direction 2» it is $2z_2$.
- $g(x) = \mathbf{w}^T\mathbf{x} + w_0$

- The distance from a point to the separating hyperplane is

$$z = \frac{w^T x + w_0}{\|w\|} = \frac{g(w)}{\|w\|}$$



Goal: Find **w** and w_0 maximizing the margin!

How would you write a program finding this? Not easy unless we state the objective function cleverly!

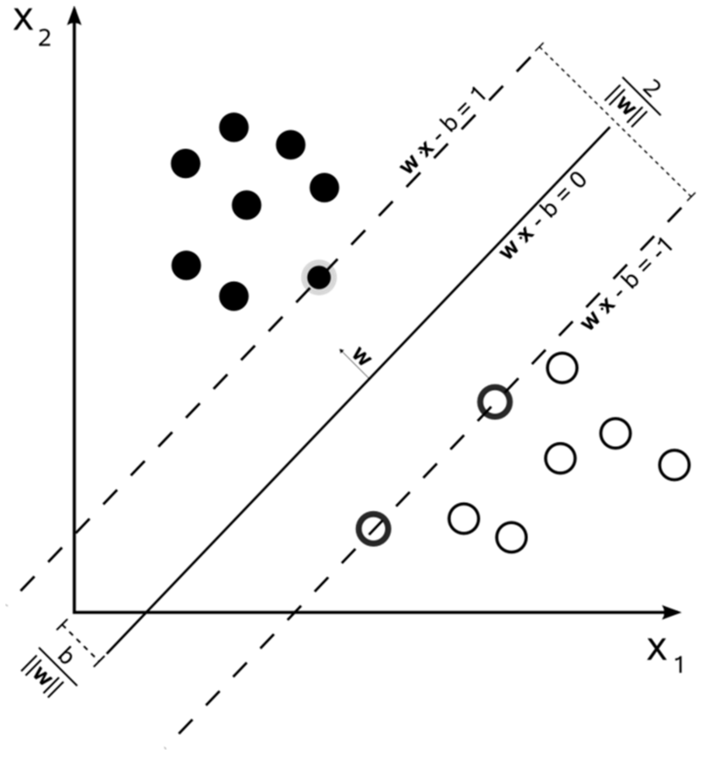
Rescaling g(x)

Does not change the margin

- We can scale g(**x**) such that g(**x**) will be equal to 1 or -1 at the closest points in the two classes. This is equivalent to:

1. Have a margin of $\frac{1}{\|w\|} + \frac{1}{\|w\|} = \frac{2}{\|w\|}$
2. Require that

$$g(x) = w^T x + w_0 \geq 1, \quad \forall x \in \omega_1$$
$$g(x) = w^T x + w_0 \leq -1, \quad \forall x \in \omega_2$$



Remember our goal: Find **w** and w_0 yielding the maximum margin
NOTE: in this figure, b is used instead of w0

Maximum-margin objective function

- The hyperplane with maximum margin can be found by solving the optimization problem (w.r.t. **w** and w_0):

The ½ factor is for later convenience

$$\text{minimize} \quad J(w) = \frac{1}{2}\|w\|^2$$

$$\text{subject to} \quad y_i(w^T x_i + w_0) \geq 1, \quad i = 1,2,\ldots N$$

Note: We assume here fully class-**separable** data!

- Checkpoint: Do you understand the formulation?
- How is this criterion related to maximizing the margin?

Note! We are somewhat done -- Matlab (or similar software) can solve this now.

But we seek more insight!

QUESTIONS
ANSWERS

The optimization problem with margins

$$\text{minimize} \quad J(w) = \frac{1}{2}\|w\|^2$$

$$\text{subject to} \quad y_i(w^T x_i + w_0) \geq 1, \quad i = 1,2,\ldots N$$

- This is a **quadratic optimization** task with a set of **linear inequality contraints**.
- It can be shown that the solution has the form:

$$\mathrm{w} = \sum_{i=1}^{N} \lambda_i y_i x_i \quad \text{where} \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

- The λ_i's are called Lagrange multipliers.
- The λ_i's can be either 0 or positive.
- We see that the solution w is a <u>linear combination of $N_S \leq N$ feature vectors associated with a $\lambda_i > 0$.</u>

Solving the optimization problem

- The optimization problem

$$\text{minimize} \quad J(w) = \frac{1}{2}\|w\|^2$$

$$\text{subject to} \quad y_i(w^T x_i + w_0) \geq 1, \quad i = 1,2,\ldots N$$

has a dual representation with equality constraints:

$$\max_{\lambda} \boxed{\left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i^T x_j \right)} \longleftarrow L$$

$$\text{subject to} \quad \sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_{\mathrm{i}}\, y_i = 0 \quad \text{and} \quad \lambda_i \geq 0 \forall i$$

- This is easier to solve and can be reformulated as:

$$\text{maximize} \quad L(w, w_0, \lambda)$$

$$\text{subject to} \quad \mathrm{w} = \sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_{\mathrm{i}} y_i x_i$$

$$\sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_{\mathrm{i}} y_i = 0 \quad \text{and} \quad \lambda_i \geq 0 \forall i$$

- L is called the Lagrange multipliers
- They can be either 0 or positive.
- The solution w can be expressed as a linear compination of Ns<N feature vectors with $\lambda_i > 0$
- These are known as the support vectors of the problem

$$\text{maximize} \quad L(w, w_0, \lambda)$$

$$\text{subject to} \quad w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0 \quad \text{and} \quad \lambda_i \geq 0 \forall i$$

Solving the optimization problem

This is easier to solve and can be reformulated as:

$$\max_{\lambda}\left(\sum_{i=1}^{N}\lambda_i-\frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j \boxed{x_i^T x_j}\right)$$

$$\text{subject to}\quad \sum_{i=1}^{N}\lambda_i y_i = 0 \quad\text{and}\quad \lambda_i \geq 0\,\forall i$$

- Note that the training samples x_i and x_j occurr as inner products of pairs of feature vectors. **The solution does not depend on the dimensionality of the feature vector, only on the scalar inner product.**
- The computational complexity can be expected to depend on the number of pixels in the training data set, N.
- In this setting we just accept that the solution can be found in optimization theory.
- Matlab and python libraries are used.

Support vectors

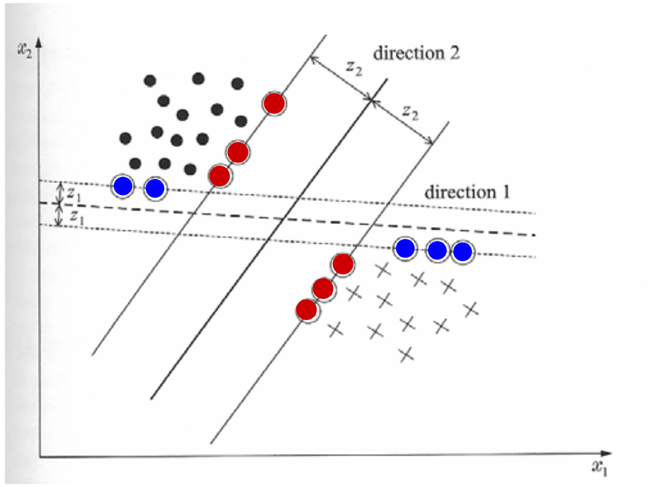
- The feature vectors $\mathbf{x}_i$ with a corresponding $\lambda_i>0$ are called the support vectors for the problem.

$$w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

- The classifier defined by this hyperplane is called a Support Vector Machine.
- Depending on y_i (+1 or -1), the support vectors will thus lie on either of the two hyperplanes

  $\mathbf{w}^T\mathbf{x}+w_0=\pm 1$

- The support vectors are the points in the training set that are closest to the decision hyperplane.
- The optimization has a unique solution, only one hyperplane satisfies the conditions.



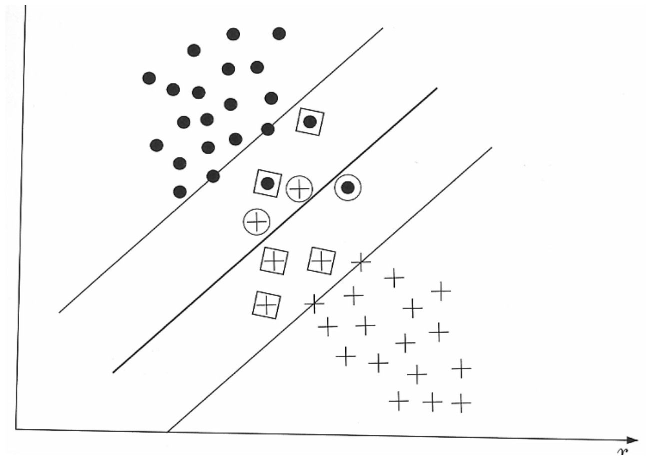
The support vectors for hyperplane 1 are the blue circles.
The support vectors for hyperplane 2 are the red circles.

QUESTIONS
ANSWERS

The nonseparable case

- If the two classes are nonseparable, a hyperplane satisfying the conditions $w^Tx - w_0 = \pm 1$ cannot be found.
- The feature vectors in the training set are now either:

1. Vectors that fall outside the band and are correctly classified.
2. Vectors that are inside the band and are correctly classified. They satisfy $0 \le y_i(w^Tx + w_0) < 1$ □
3. Vectors that are misclassified – expressed as $y_i(w^Tx + w_0) < 0$ ○



□ Correctly classified

○ Erroneously classified

- The three cases can be treated under a single type of contraints if we introduce slack variables ξ_i:

$$y_i[w^T x + w_0] \geq 1 - \xi_i$$

  - The first category (outside, correct classified) have $\xi_i=0$
  - The second category (inside, correct classified) have $0\leq \xi_i \leq 1$
  - The third category (inside, misclassified) have $\xi_i > 1$

- The optimization goal is now *to keep the margin as large as possible and the number of points with $\xi_i > 0$ as small as possible.*

Cost function – nonseparable case

- The cost function to minimize is now

$$J(w, w_0, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} I(\xi_i)$$

$$\text{where} \qquad \mathrm{I}(\xi_i) = \begin{cases} 1 & \xi_\mathrm{i} > 0 \\ 0 & \xi_\mathrm{i} = 0 \end{cases}$$

and ξ is the vector of parameters ξ_i.

- C is a parameter that controls how much misclassified training samples is weighted.
- We skip the mathematics and present the alternative dual formulation:

$$\max_{\lambda}\left(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j x_i^T x_j\right)$$

$$\text{subject to} \quad \sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_\mathrm{i} y_i = 0 \quad \text{and} \quad 0 \le \lambda_i \le C \quad \forall i$$

- All points between the two hyperplanes (ξ_i>0) can be shown to have λ_i=C.

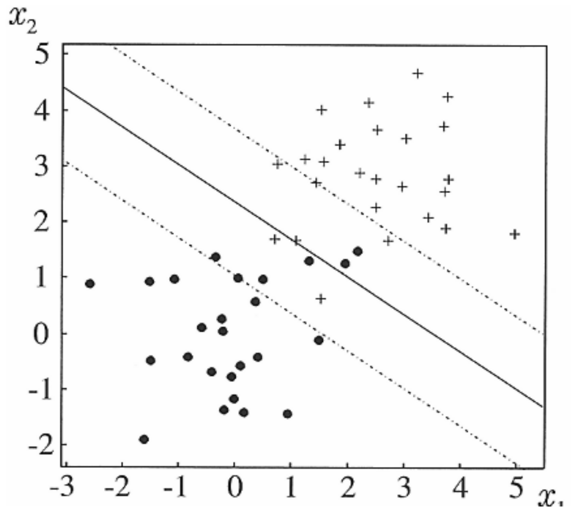
Nonseparable vs. separable case

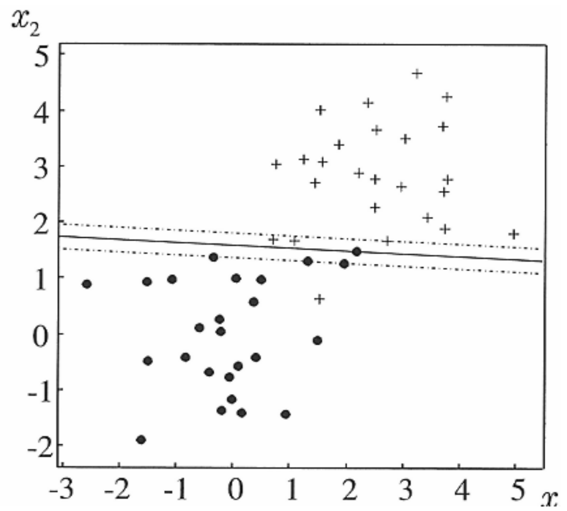
- Note that the slack variables ξ_i does not enter the dual problem explicitly.
- The only difference between the linear separable and the non-separable case is that the Lagrange-multipliers are bounded by C.
- Training a SVM classifier consists of solving the optimization problem.
  - The problem is quite complex since it grows with the number of training pixels.
  - It is computationally heavy.

An example – the effect of C

- C is the misclassification cost.

C=0.2





C=100

- Selecting too high C will give a classifier that fits the training data perfect, but fails on different data set.

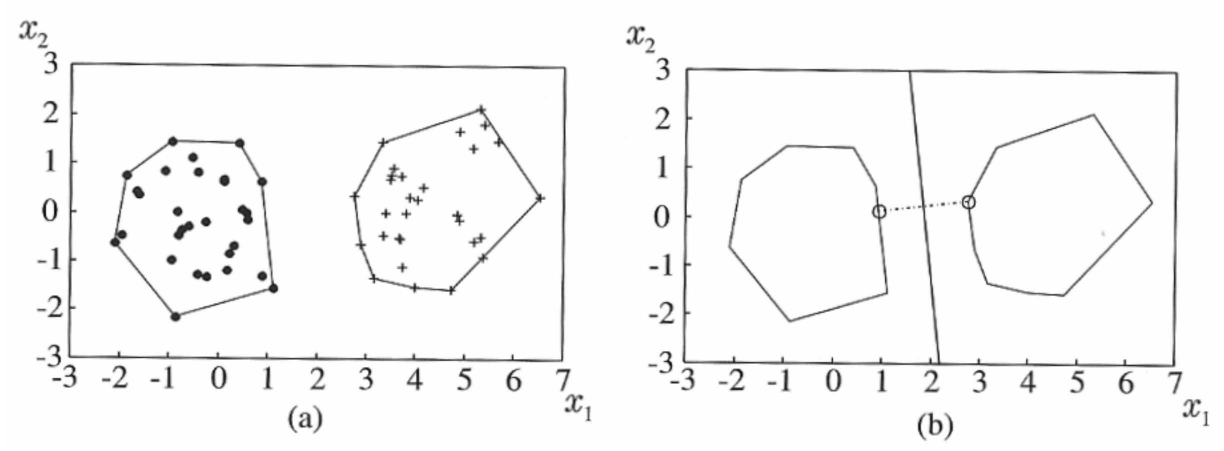
- The value of C should be selected on the validation set.

QUESTIONS
ANSWERS

SVM: A geometric view

- SVMs can be related to the convex hull of the different classes. Consider a class that contains training samples X={x_1,...x_N}.
- The convex hull of the set of points in X is given by all convex combinations of the N elements in X.

  - A region *R* is convex if and only if for any two points x_1,x_2 in *R*, the whole line segment between x_1 and x_2 is inside the *R*.
  - The convex hull of a region is the smalles convex region H which satisfies the conditions $R \subseteq H$.



- The convex hull for a class is the smallest convex set that contains all the points in the class (X).
- Searching for the hyperplane with the highest margin is equivalent to searching for the two nearest points in the two convex sets.
  - This can be proved, but we just take the result as an aid to get a better visual interpretation of the SVM hyperplane.

Reduced convex hull

- To get a useable interpretation for nonseparable classes, we need the reduced convex hull.
- The convex set can be expressed as:

$$conv\{X\} = \left\{ y : \; y = \sum_{i=1}^{N} \lambda_i x_i : x_i \in X, \right.$$
$$\left. \sum_{i=1}^{N} \lambda_i = 1, \;\; 0 \le \lambda_i \le 1 \; \right\}$$

- The **reduced convex hull** is :

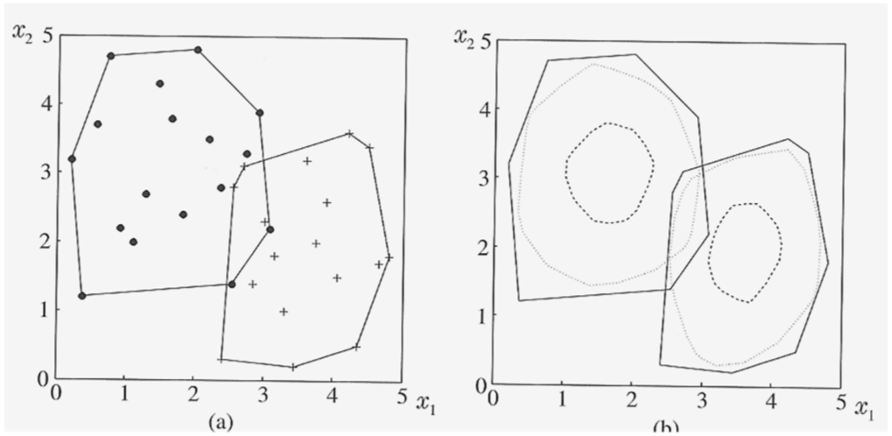
$$R\{X, \mu\} = \left\{ y : \; y = \sum_{i=1}^{N} \lambda_i x_i : x_i \in X, \right.$$
$$\left. \sum_{i=1}^{N} \lambda_i = 1, \;\; 0 \le \lambda_i \le \mu \; \right\}$$

Here we add a restriction that λ_i must also be smaller than μ

- μ is a scalar between 0 and 1. $\mu = 1$ gives the regular convex hull.

Reduced convex hull - example

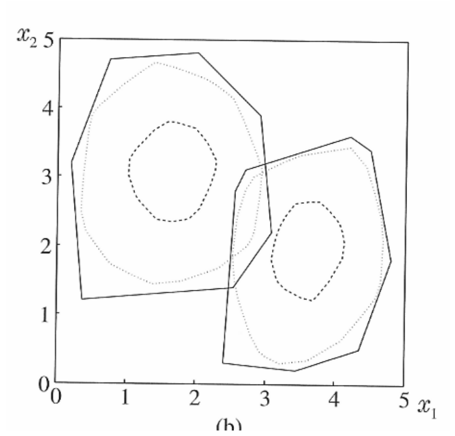
μ=1
Regular
convex
hulls

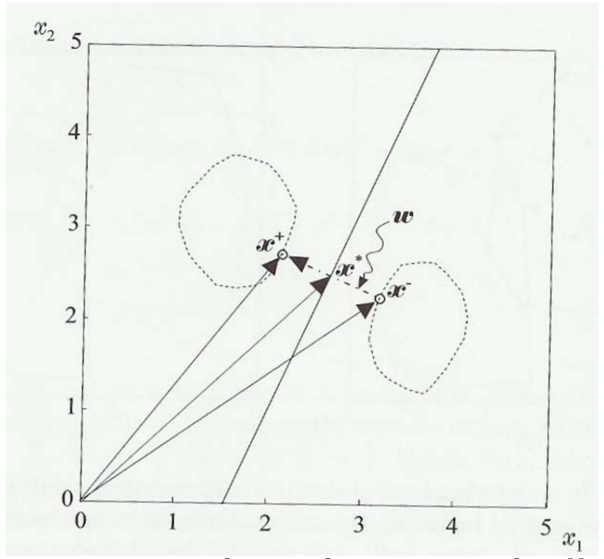


.....: μ=0.4
----: μ=0.1
Reduced
convex
hulls

- Data set with overlapping classes.
- For small enough values of μ, we can make the two reduced convex hulls non-overlapping.
- A very rough explanation of the non-separable SVM problem is that a value of μ that gives non-intersecting reduced convex hulls must be found.
- Given a value of μ that gives non-intersecting reduced convex hulls, the best hyperplane will bisect the line between the closest points in these two reduced convex hulls.

Relating μ and C





- Given a value of μ that gives non-intersecting reduced convex hulls, find the hyperplane by finding the closest two points in the two sets.
- Several values of μ can give nonintersecting reduced hulls.
- μ is related to C, the cost of misclassifying training regions (see page 101).
- A high C will give regions that just barely give nonintersecting regions.
- The most robust considering a validation data set is probably a smaller value of C (and μ).

Checkpoint

- What does this criterion mean:

$$J(w, w_0, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} I(\xi_i)$$

where $\quad \mathrm{I}(\xi_i) = \begin{cases} 1 & \xi_\mathrm{i} > 0 \\ 0 & \xi_\mathrm{i} = 0 \end{cases}$

and ξ is the vector of parameters ξ_i.

- Which points are the support vectors in the linear case?

QUESTIONS
ANSWERS

SVMs: The nonlinear case intro.

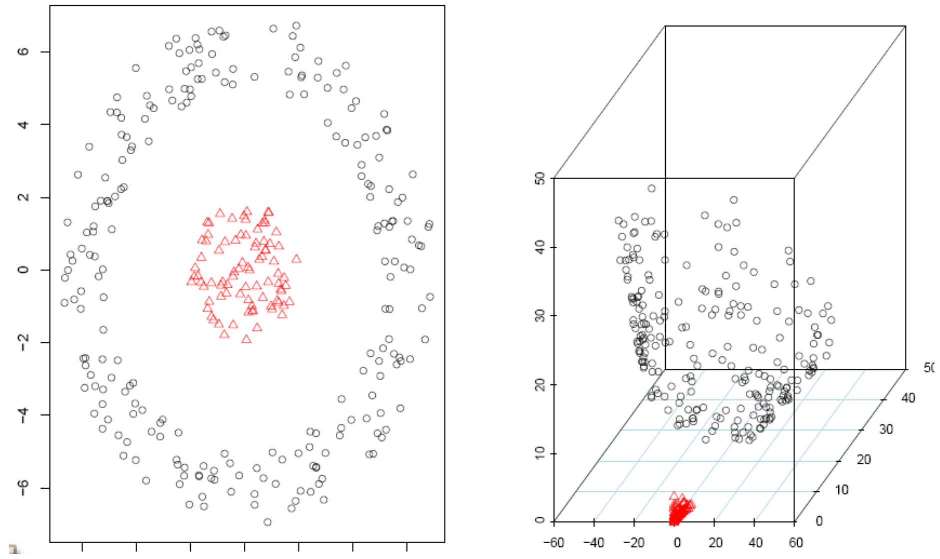
- The training samples are l-dimensional vectors; we have until now tried to find a linear separation in this l-dimensional feature space
- This seems quite limiting
- What if we increase the dimensionality (map our samples to a higher dimensional space) before applying our SVM?
- Perhaps we can find a better linear decision boundary in that space? Even if the feature vectors are not linearly separable in the input space, they might be (close to) separable in a higher dimensional space

An examle: from 2D to 3D

- Let **x** be a 2D vector **x**=[x_1,x_2].
- In the toy example on the right, the two classes can <u>not</u> be linearly separated in the original 2D space.
- Consider now the transformation

$$y = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

- Now, the transformed points in this 3D space <u>can</u> be separated by a plane.
- The separating plane in 3D maps out an ellipse in the original 2D space



Cf. next slide, note that $y_i^T y_j = (x_i^T x_j)^2$.

Introducing the kernel trick

- A kernel can take the feature vectors into a higher dimensional space where they are linearly separable.
- Consider a simple example with two points:
  - **x** = $(x_1,x_2,x_2)^T$ and **y**=$(y_1,y_2,y_3)^T$
- If we want to add interactions between the points, we could transform the data into 9-dimensional space:
  - $\phi(\mathbf{x}) = (x_1^2,x_1x_2,x_1x_3,x_2x_1,x_2^2,x_2x_3,x_3x_1,x_3x_2,x_3^2)^T$
  - $\phi(\mathbf{y}) = (y_1^2,y_1y_2,y_1y_3,y_2y_1,y_2^2,y_2y_3,y_3y_1,y_3y_2,y_3^2)^T$
- The inner product in 9-dimensional space then gives us the interactions:
- $\varphi(x)^T\varphi(y) = \sum_{i,j=1}^{3} x_i x_j y_i y_j$
- If we use a kernel function, we can calculate this i 3-dimensional space using kernels
- $k(x,y) = (x^T y)^2 = \sum_{i,j=1}^{3} x_i x_j y_i y_j$

SVMs and kernels

- Note that in both the optimization problem and the evaluation function, g(**x**), the samples come into play as inner products only

$$\max_{\lambda}\left(\sum_{i=1}^{N}\lambda_i-\frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j \boxed{x_i^T x_j}\right)$$

$$\text{s.t.}\quad 0\le\lambda_i\le C\quad\forall i$$

$$\sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}y_i=0$$

$$g(x)=w^T x+w_0$$

$$=\sum_{i=1}^{N_s}\lambda_i y_i \boxed{x^T x_i}+w_0$$

Called «kernel»

- If we have a function evaluating inner products, K(**x**$_i$,**x**$_j$), we can ignore the samples themselves
- Let's say we have K(**x**$_i$,**x**$_j$) evaluating inner products in a higher dimensional space:

  -> no need to do the mapping of our samples explicitly!

Useful kernels for classification

- Polynomial kernels

$$K(x,z) = \left(x^T z - 1\right)^q, \quad q > 0$$

- Inner product measures similarity between x and z.
- If x has dimension d, the kernel takes the data into dxd-dimensions

Useful kernels for classification

- Radial basis function kernels (very commonly used!)

The inner product is related to the similarity of the two samples.

$$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{\sigma^2}\right)$$

Note the we need to set the σ parameter

The «support» of each point is controlled by σ.

- This measures the similarity between x and z, scaled between 0 and 1
- σ determines the size of the similarity regions
- If we rewrite the exponential term, we see that this expresses the inner product in an infinite number of dimensions

$$\exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2\right) = \sum_{j=0}^{\infty} \frac{(\mathbf{x}^\top \mathbf{x}')^j}{j!} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right)$$

$$= \sum_{j=0}^{\infty} \sum_{\sum n_i = j} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \frac{x_1^{n_1} \cdots x_k^{n_k}}{\sqrt{n_1! \cdots n_k!}} \exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right) \frac{x'^{n_1}_1 \cdots x'^{n_k}_k}{\sqrt{n_1! \cdots n_k!}}$$

Useful kernels for classification

- Hyperbolic tangent kernels (often with β=2 and γ=1)

$$K(x,z) = \tanh\left(\beta x^T\ z + \gamma\right)$$

The kernel formulation of the objective function

- Given the appropriate kernel (e.g. «radial» with width σ) and the cost of misclassification C, the optimization task is:

$$\max_{\lambda}\left(\sum_i \lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j K(x_i, x_j)\right)$$

$$\text{subject to} \quad 0 \le \lambda_i \le C, \quad i = 1,....N$$

$$\sum_i \lambda_i y_i = 0$$

- The resulting classifier is:

$$\text{assign x to class } \omega_1 \text{ if } g(x) = \sum_{i=1}^{N} \lambda_i y_i K(x_i, x) + w_0 > 0 \text{ and to class } \omega_2 \text{ otherwise}$$

Example of nonlinear decision boundary

- This illustrates how the nonlinear SVM might look in the original feature space
- RBF kernel used

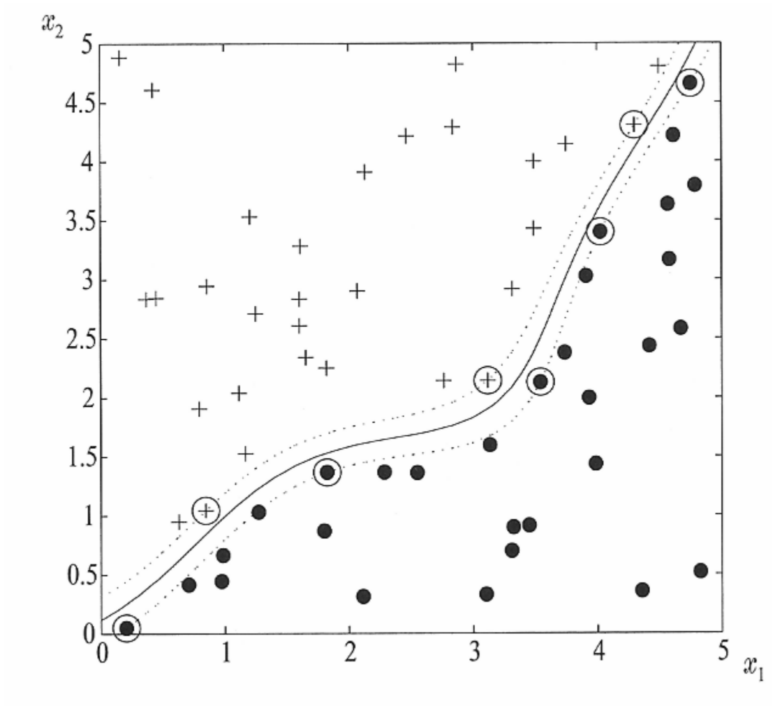


Figure 4.23 in
PR by Teodoridis et.al.

From 2 to M classes

- All we have discussed up until now involves only separating 2 classes. How do we extend the methods to M classes?
- Two common approaches:
  - One-against-all
    - For each class m, find the hyperplane that best disciminates this class from all other classes. Then classify a sample to the class having the highest output. (To use this, we need the VALUE of the inner product and not just the sign.)
  - **Compare all sets of pairwise classifiers**
    - **Find a hyperplane for each pair of classes. This gives M(M-1)/2 pairwise classifiers. For a given sample, use a voting scheme for selecting the most-winning class.**

How to use a SVM classifier

- Find a library with all the necessary SVM-functions ☺
  - Matlab: for example fitcsvm
  - Python: scikit-learn
- Read the introductory guides.
- Often a radial basis function kernel is a good starting point.
- Scale the data to the range [-1,1] (features with large values will not dominate).
- **Find the optimal values of C and σ by performing a grid search on selected values and using a validation data set.**
- Retrain the classifier using the best value from the grid search.
- Test using a separate test set.

How to do a grid search

- Grid search: try pairs of (C,σ). Compute the accuracy on the validation data set.
- Select the pair that gets the best classification performance on the validation data set.
- Use the following values of C and σ:
    - $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$
    - $\sigma = 2^{-15}, 2^{-13}, \ldots., 2^{3}$

Summary / Learning goals

- Understand enough of SVM classifiers to be able to use it for a classification application.
  - Understand the basic linear separable problem and what the meaning of the solution with the largest margin is.
    - Geometrically and by using formulas
  - Understand how SVMs work in the non-separable case using a cost for misclassification.
  - Accept the kernel trick: that the original feature vectors can be transformed into a higher dimensional space, and that linear SVM is applied in this space without explicitly doing the feature transform
  - Know briefly how to extend from 2 to M classes.
  - Know which parameters (C, etc.) the user must specify and how to perform a grid search for these.
  - Be able to find a SVM library and use it correctly

QUESTIONS
ANSWERS