INF 5300 – Support Vector Machine Classifiers (SVM)

Anne Solberg (anne@ifi.uio.no)

- Introduction:
- Linear classifiers for two-class problems
- The kernel trick – from linear to a high-dimensional generalization
- Generation from 2 to M classes
- Practical issues

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.

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 means.
  - 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.
  - Know briefly how to extend from 2 to M classes.
  - Know which parameters (C,) the user must specify and how to perform a grid search for these.
  - Be able to find a SVM library and use it correctly ☺

Linear classification
What we learned in INF 4300

- The classes were described by multivariate Gaussian distributions given the mean vector μ_s and covariance matrix Σ_s :

$$p(x \mid \omega_s) = \frac{1}{(2\pi)^{P/2}\left|\Sigma_s\right|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_s)^t \Sigma_s^{-1}(x-\mu_s)\right]$$

- Finding the best class for a new pattern was done by computing the probability for each class.
- This could be reformulated as computing discriminant functions:

$$g_j(\mathbf{x}) = -\frac{1}{2}\underbrace{(\mathbf{x}-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)}_{\text{Scaled distance from } \boldsymbol{x} \text{ to } \boldsymbol{\mu}_j} - \frac{P}{2}\ln 2\pi - \frac{1}{2}\ln\left|\boldsymbol{\Sigma}_i\right| + \ln P(\omega_j)$$

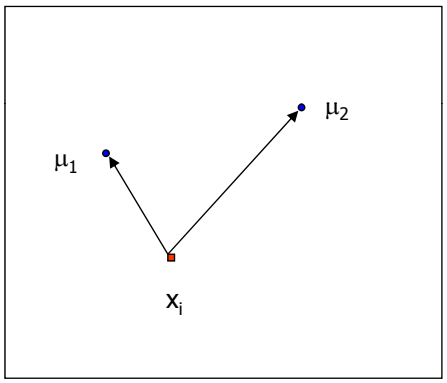
Linear classification
What we learned in INF 4300

- We look at some special cases of this (uncorrelated features, common covariance matrix, and full covariance matrix).
- If the features were indepenent ($\Sigma_j=\sigma^2 I$) the discriminant function was simplified to:

$$g_j(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu}_j)^T(\mathbf{x}-\boldsymbol{\mu}_j) + \ln P(\omega_j)$$

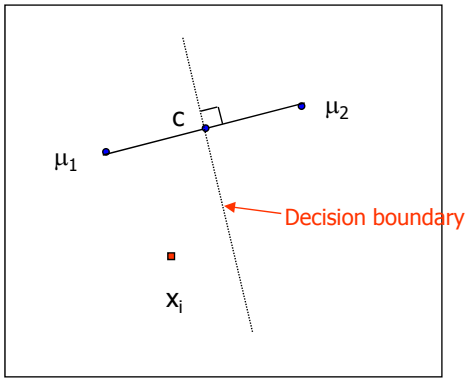
$$= -\frac{1}{2\sigma^2}\|\mathbf{x}-\boldsymbol{\mu}_j\|^2 + \ln P(\omega_j)$$

- This resulted in linear decision boundaries.
- Computing this discriminant function to classify pattern x_i involves computing the distance from the point to the mean values μ_s for each class.



Linear classification
What we learned in INF 4300

- We also found that the discriminant functions (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 connect 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.



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 computing using inner products.
- We start with two linearly separable classes.
- Extension to two non-separable classes.
- Extenison to M classes.
- Using kernels to separate classes that are cannot be separated in the input space.

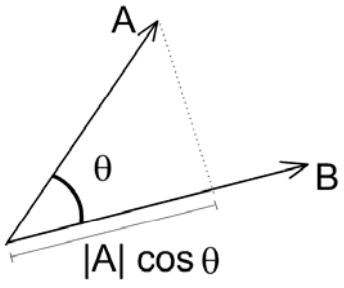
Linear algebra basics: Inner product between two vectors.

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

$$\langle x, y \rangle = \sum_{i=1}^{N} x_i y_i = x^T y$$

- The angle between two vectors a and b is defined as:

$$\cos\ \theta = \frac{\langle a, b \rangle}{\|a\| \|b\|}$$



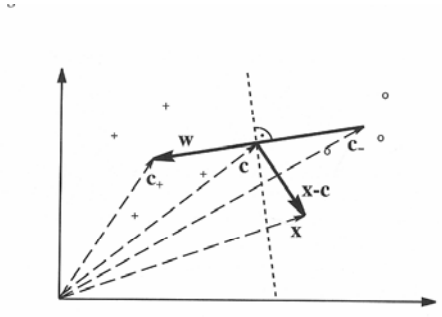
A new view at linear classification using inner products

- 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 $c=(c_+ + c_-)/2$.

  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 $w=c_+ - c_-$.
- 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.

What is w related to the hyperplane?

- If x_1 and x_2 are two points on the decision boundary (on the hyperplane) the following must be true:

$$w^T x_1 + w_0 = w^T x_2 + w_0 \Rightarrow$$
$$w^T (x_1 - x_2) = 0$$

- Thus w must be normal to the line connecting x_1 and x_2.

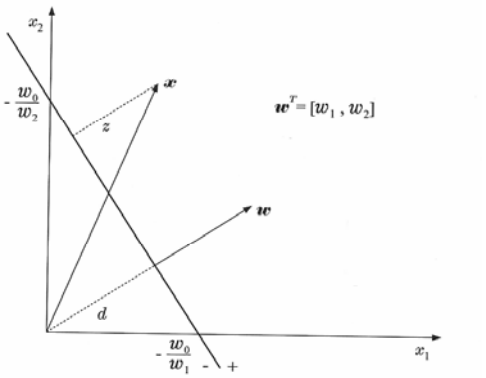


FIGURE 3.1: Geometry for the decision line. On one side of the line $g(x) > 0(+)$ and on the other $g(x) < 0\ (-)$.

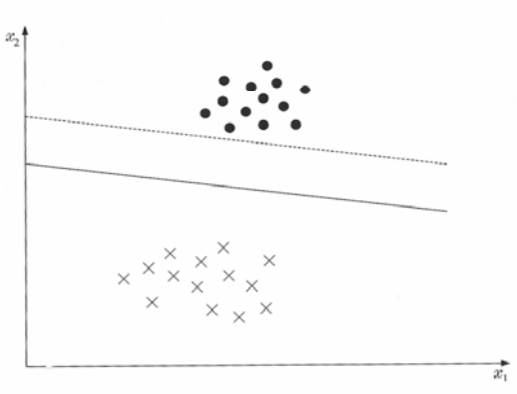
Remark: in this figure we consider w_0=0.

- If the classes are linearly separable, there exist a hyperplane $w^{*T}x=0$ such that:

$$w^{*T}\, x > 0 \quad \forall x \in \omega_1$$
$$w^{*T}\, x < 0 \quad \forall x \in \omega_2$$

- The above also covers the situation where the hyperplane does not cross the origin, $w^{*T}x+w_0=0$, since this can reformulated as $x'=[x,1]^T$, $w'=[w^{*T},w_0]^T$. Then $w^{*T}x+w_0=w'^Tx'$.
- Remember from 4300 that the decision boundary was defined as the surface where the discriminant function $g_1(x)-g_2(x)=0$ ($g_1(x)=g_2(x)$).

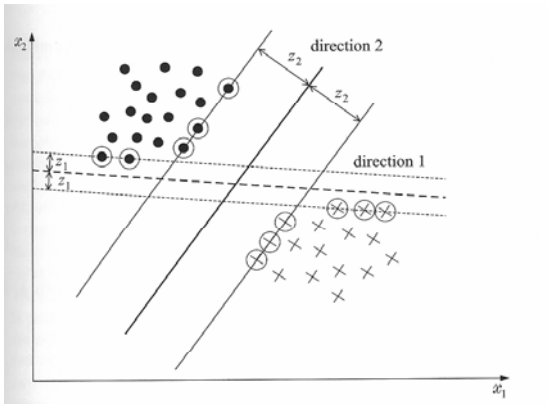
- There can be many such hyperplanes.
- Which of these two is best, and why?



Hyperplanes and margins

- A hyperplane is defined by its direction (w) and exact position (w_0).
- 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.
- The margin for direction 1 is $2z_1$, and for direction 2 it is $2z_2$.
- The distance from a point to a hyperplane is

$$z = \frac{|g(x)|}{\|w\|}$$

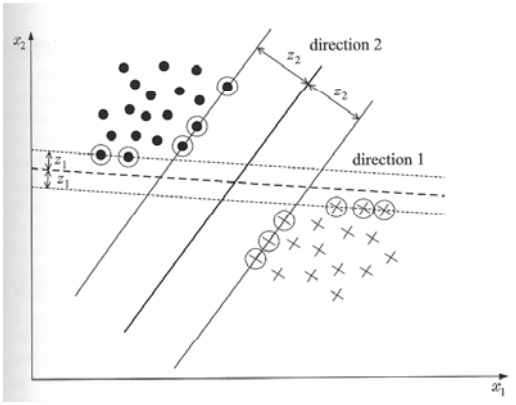


Hyperplanes and margins

- We can scale w and w_0 such that g(x) will be equal to 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

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



The optimization problem with margins

- The class indicator for pattern i, y_i, is defined as 1 if y_i belongs to class ω_1 and -1 if it belongs to ω_2.
- The best hyperplane with margin can be found by solving the optimization problem with respect to w:

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

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

The optimization problem with margins

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

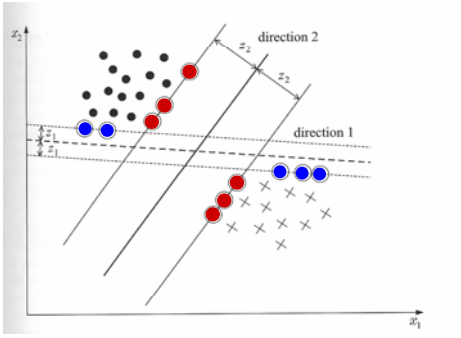
- This is a quadratic optimization task with a set of linear inequality contraints.
- It can be shown that the solution has the form:

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

- 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 linear combination of $N_s \leq N$ feature vectors associated with a $\lambda_i > 0$.

- The feature vectors x_i with a corresponding $\lambda_i>0$ are called the support vectors for the problem.
- 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

  $w^T x+w_0=\pm 1$
- 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.

Solving the optimization problem

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

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

  has a dual representation with equality constraints:

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

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

  $\sum_{i=1}^{N}\lambda_i y_i=0 \quad \text{and} \quad \lambda_i\geq 0\forall i$
- This is easier to solve and can be reformulated as:

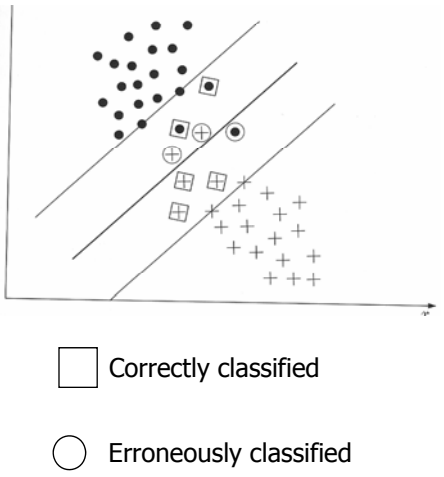
  $$\max_{\lambda}\left(\sum_{i=1}^{N}\lambda_i-\frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j x_i^T x_j\right)$$

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

The nonseparable case

- If the two classes are nonseparable, a hyperplane satisfying the conditions $w^T x - w_0 = \pm 1$ 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 $0 \leq y_i(w^T x + w_0) < 1$
3. Vectors that are misclassified – expressed as $y_i(w^T x + w_0) < 0$



- 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

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

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

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 an alternative formulation:

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

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

- All points between the two hyperplanes ($\xi_i>0$) can be shown to have $\lambda_i=C$.

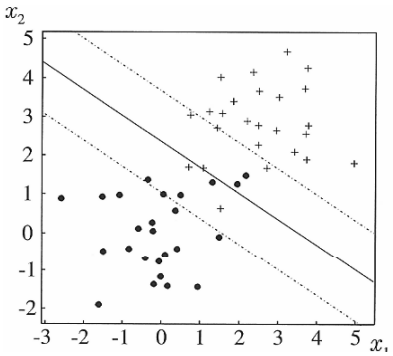
Nonseparable vs. separable case

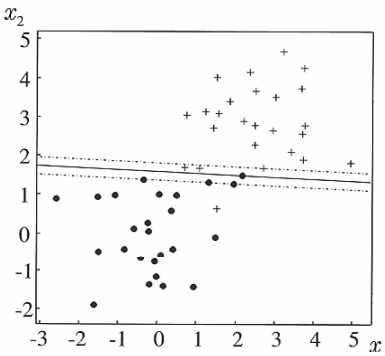
- Note that the slack variables ξ_i does not enter the problem explicitly.
- The only difference between the linear separable and the non-separable case is that the Lagrange-multipliers are bounded by C.
- Training a SVM classifier consists of solving the optimization problem.
  - The problem is quite complex since it grows with the number of training pixels.
  - It is computationally heavy.
  - We will get back with hints of software libraries to use at the end of the lecture....

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 using a separate validation set. Separate the training data into a part used for training, train with different values of C and select the value that gives best results on the validation data set. Then apply this to new data or the test data set. (explained later)

How to go from 2 to M classes

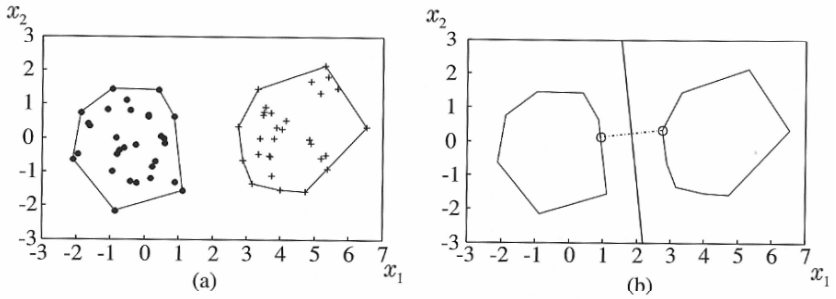
- All we have discussed up to now involves only separating 2 classes. How do we extend the methods to M classes?
- Two common approaches:
  - One-against-all
    - For each class m, find the hyperplane that best disciminates this class from all other classes. Then perform a majority vote across classifiers or use some other measure to select the best of these
  - Compare all sets of pairwise classifiers
    - Find a hyperplane for each pair of classes. This gives M(M-1)/2 pairwise classifiers. Again, use of voting scheme for selecting the best of these.

ν-SVM

- ν-SVM is a variant of SVM.
- Ordinary SVM is most commonly used, but ν-SVM can be more effective if the two classes are imbalances (more data from one of the classes).
- We will not go through ν-SVM (section 3.7.3).....

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.
- From INF 4300:
  - 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:

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

- The **reduced convex hull** is :

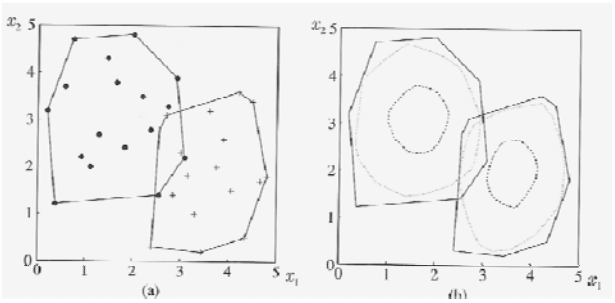
$$R\{X, \mu\} = \{y : \; y = \sum_{i=1}^{N} \lambda_i x_i : x_i \in X,$$
$$\sum_{i=1}^{N} \lambda_i = 1, \;\; 0 \le \lambda_i \le \mu \;\}$$

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

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

Reduced convex hull - example

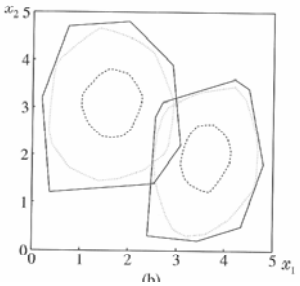
μ=1
Regular
convex
hulls

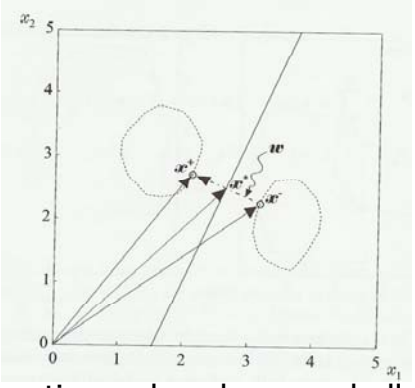


.....: μ=0.4
----: μ=0.1
Reduced
convex
hulls

- 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 $\quad I(\xi_i) = \begin{cases} 1 & \xi_i > 0 \\ 0 & \xi_i = 0 \end{cases}$

and ξ is the vector of parameters ξ_i.

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

SVMs: The nonlinear case

- We have now found a classifier that is not defined in terms of the class centres or the distributions, **but in terms of patterns close to the borders between classes, the support vectors.**
- It gives us a solution in terms of a hyperplane. This hyperplane can be expressed as a inner product between the training samples:

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

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

- The training samples are l-dimensional vectors.
- What if the classes overlap in l-dimensional space:
  - Can we find a mapping to a higher dimensional space, and use the SVM framework in this higher dimensional space?

- Assume that there exist a mapping from l-dimensional feature space to a k-dimensional space **(k>l)** :

$$x \in R^l \rightarrow y \in R^k$$

- Even if the feature vectors are not linearly separable in the input space, they might be separable in a higher dimensional space.
- Classification of a new pattern x is to be computed by computing the sign of

$$g(x) = w^T x + w_0$$
$$= \sum_{i=1}^{N_s} \lambda_i y_i x_i^T x_i + w_0$$

- In k-dimensional space, this involves the inner product between two k-dimensional vectors.
- Can it really help to go to a higher dimensional space?

An examle: from 2D to 3D

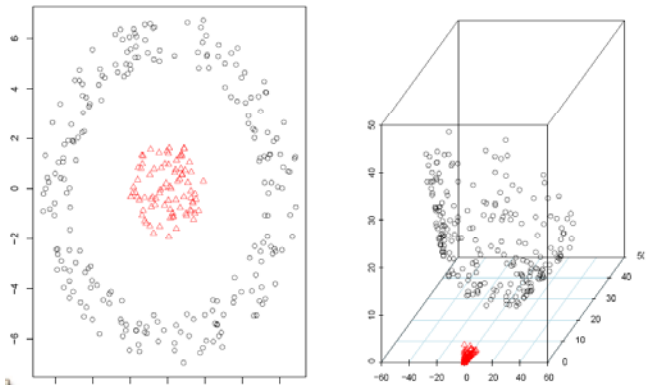
- Let x be a 1x2 vector $x=[x_1,x_2]$.
- Consider the transformation

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

- On the toy example, the two classes can not be linearly separated in the original 2D space.
- It can be shown that

$$y_i^T y_j = \left(x_i^T x_j\right)^2$$

- Given the transformation above, these points in a 3D space CAN actually be separated by a hyperplane.
- In 2D, we would need an ellipse to separate the classes.
- In 3D, this ellipse can be expressed as a linear function of y.



Remark: we don't know yet how to construct this mapping or other useful mappings.

A useful trick: Mercer's theorem – finding a mapping to the high-dimensional space using a kernel

What we need from all this math is just that the inner product can be computed using the kernel K(x,z). Someone has also identified some useful kernels.

Assume that ϕ is a mapping:

$$x \to \phi(x) \in H$$

where H is an Euclidean space.

The inner product has an equivalent representation

$$\sum_r \phi_r(x)\phi_r(z) = K(x,z)$$

where $\phi_r(x)$ is the r-component of the mapping $\phi(x)$ of x, and K(x,z) is a symmetric function satisfying

$$\int K(x,z)g(x)g(z)dxdz \geq 0$$

for any g(x), $x \in R^l$ such that

$$\int g(x)^2 dx < +\infty$$

K(x,z) defines a inner product. K(x,z) is called a kernel.
Once a kernel has been defined, a mapping to the higher dimensional space is defined.

Useful kernels for classification

- Polynomial kernels

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

- Radial basis function kernels (most commonly used)

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

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

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

- The most common type of kernel is the radial basis function. It has an extra parameter σ that must be tuned.

The kernel formulation of the cost function

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

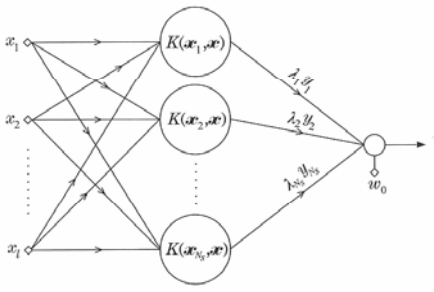
$$\max_{\lambda}\left(\sum_i \lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j K(x_i, x_j)\right)$$
$$\text{subject to} \quad 0 \le \lambda_i \le C, \quad i = 1,....N$$
$$\sum_i \lambda_i y_i = 0$$

- The resulting classifier is:

$$\text{assign x to class } \omega_1 \text{ if } g(x) = \sum_{i=1}^{N} \lambda_i y_i K(x_i, x) + w0 > 0 \text{ and to class } \omega_2 \text{ otherwise}$$

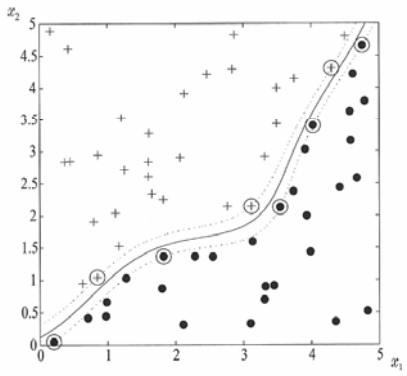
SVM architecture

- Notice how the kernels are used to compute the inner product between all pairs of samples x_i in the training data set.



Example of the decision boundary

- This illustrates how the nonlinear SVM might look in the original feature space.



How to use a SVM classifier

- Find a library with all the necessary SVM-functions ☺
  - For example libSVM http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Read the introductory guide http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
- Use a radial basis function kernel.
- Scale the data to the range [-1,1] (will not be dominated with features with large values).
- Find the optimal values of C and σ by performing a grid search on selected values and using a validation data set.
- Train the classifier using the best value from the grid search.
- Test using a separate test set.

How to do a grid search

- Use n-fold cross valiation (e.g. 10-fold cross-validation).
  - 10-fold: divide the training data into 10 subsets of equal size. Train on 9 subsets and test on the last subset. Repeat this procedure 10 times.
- Grid search: try pairs of (C,σ). Select the pair that gets the best classification performance on average over all the n validation test subsets.
- Use the following values of C and σ:
  - C = 2^{-5}, 2^{-3}, ..., 2^{15}
  - σ = 2^{-15}, 2^{-13}, ..., 2^{3}

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 means.
  - 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.
  - Know briefly how to extend from 2 to M classes.
  - Know which parameters (C,) the user must specify and how to perform a grid search for these.
  - Be able to find a SVM library and use it correctly ☺