INF 5300 – Basics of Support Vector Machine Classifiers (SVM)

- Two-class linear classifiers and the concept of margins
- From two to M classes
- The kernel trick – from linear to a high-dimensional generalization
- Practical issues

Sections 3.1-3.2, 3.7 (3.7.3 is a SVM-variant that we will skip), 4.17 in "Pattern Recognition" by S. Theodoridis and K. Koutroumbas.

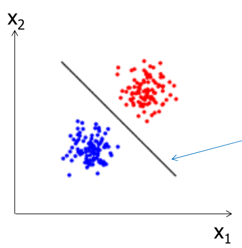
Low-level, practical details on how to actually solve the stated optimization problems are not required.

(see links on the course's web-page for pdfs)

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 implicitly transformed into a higher dimensional space in which the SVM is applied.
  - Know briefly how to extend from 2 to M classes.
  - Know which parameters the user must specify and how to perform a grid search for these.
  - Be able to find a SVM library and use it correctly ☺

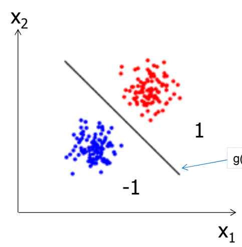
Linear classifiers I/II



Linear decision boundary

Linear classifiers II/II

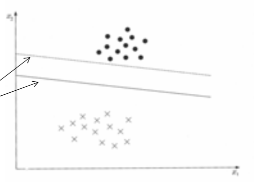
- Discriminant function:
  $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$
  Weights/orientation Threshold/bias
- Two-class problem,
  $y_i \in \{-1,1\}$
  Class indicator for pattern *i*
- $y_i = \begin{cases} -1, & g(\mathbf{x}_i) < 0 \\ 1, & g(\mathbf{x}_i) > 0 \end{cases}$
  Input pattern



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

Multiple 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?

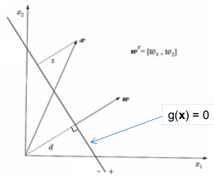


Distance to the decision line

Project **x** onto **w**

- $\mathbf{w}^T\mathbf{x}/|\mathbf{w}| = d+z$
- $g(d\mathbf{w}/|\mathbf{w}|)=0 \Rightarrow$
  $d = -w_0/|\mathbf{w}|$
- $z = \mathbf{w}^T\mathbf{x}/|\mathbf{w}| - d$
  $= \mathbf{w}^T\mathbf{x}/|\mathbf{w}| + w_0/|\mathbf{w}|$
  $\boxed{= g(\mathbf{x})/|\mathbf{w}|}$

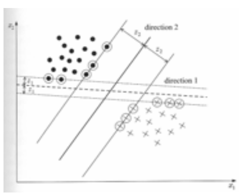
Distance from **x** to the decision boundary



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.
- The margin for «direction 1» is $2z_1$, and for «direction 2» it is $2z_2$.
- From previous slide; the distance from a point to the separating hyperplane is

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



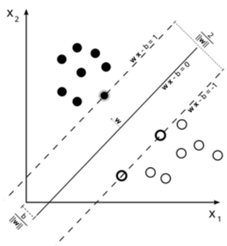
Hyperplanes and margins

Does not change the margin

- We can scale g [**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
   $w^T x + w_0 \geq 1, \quad \forall x \in \omega_1$
   $w^T x + w_0 \leq -1, \quad \forall x \in \omega_2$

- **Goal: find w and w_0 yielding the maximum margin**



Maximum-margin problem-formulation

- 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

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

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

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

More on the optimization problem

We recommend, again, to read the note on Lagrangian multipliers (see web).

Generalized Lagrangian function:

$$\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} \lambda_i [y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1]$$

$$\left.\begin{aligned} \frac{\partial}{\partial \boldsymbol{w}}\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) &= \boldsymbol{0} \\ \frac{\partial}{\partial w_0}\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) &= 0 \end{aligned}\right\} \Rightarrow \quad \boxed{\boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x}_i} \qquad \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$

$$\lambda_i [y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1] = 0, \quad i = 1, 2, \ldots, N$$

Either λ_i= 0 or g(x)=1

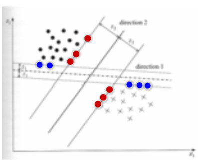
Karush-Kuhn-Tucker (KKT) conditions

Support vectors

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

$$\boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x}_i$$

- 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.

Dual representation

- Plugging $\boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x}_i$, $\sum_{i=1}^{N} \lambda_i y_i = 0$ back into L(**w**,w_0,λ) gives us

$$\max_{\lambda}\left(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j x_i^T x_j\right)$$
$$\text{s.t.} \quad \sum_{i=1}^{N}\lambda_i y_i = 0$$
$$\lambda_i \geq 0 \forall i$$

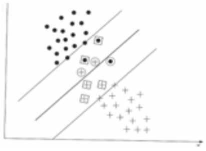
Important (for later): The samples come into play as inner products only!

3

The nonseparable case

- If the two classes are nonseparable, a hyperplane satisfying the conditions $|\mathbf{w}^T\mathbf{x}-w_0|\geq 1$ 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 but are correctly classified. They satisfy $0\leq y_i(\mathbf{w}^T\mathbf{x}+w_0)<1$ □
3. Vectors that are misclassified; expressed as $y_i(\mathbf{w}^T\mathbf{x}+w_0)<0$ ○



□ Correctly classified

○ Erroneously classified

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

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

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

- A simple change in cost function to reflect this:

$$\arg\min_{\mathbf{w},\xi,b}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i\right\}$$

$$y_i(\mathbf{w}\cdot\mathbf{x_i}-b)\geq 1-\xi_i, \quad \xi_i\geq 0$$

- **C is a parameter that controls how much misclassified, or margin-crossing, training samples are weighted.**

- Following the Lagrange path we end up with the following dual formulation:

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

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

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

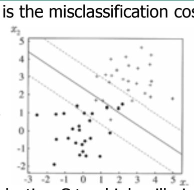
The non-zero λ_i (the support vectors) are now those on the margin, those within the margin, and those misclassified

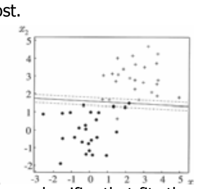
Only difference between this and the solution on slide 12.

An example – the effect of C

- C is the misclassification cost.

C=0.2





C=100

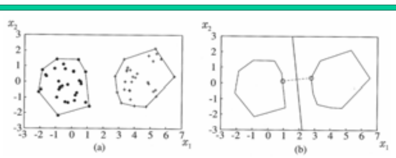
- Selecting C too high will give a classifier that fits the training data better, but likely fails on new data.
- The value of C should be selected using a separate validation set. Separate the training data into a part used for training, train with different values of C and select the value that gives best results on the validation data set. Then apply this to new data or the test data set.

How to go 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 discriminates 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.

SVM – a different geometric view

- SVMs can be related to the convex hull of the different classes. Consider a class that contains training samples $X=\{x_1,...x_N\}$.
- 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 proven, but here we just use the result as an aid to get a better geometric interpretation of the SVM hyperplane.

Reduced convex hull

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

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

- The **reduced convex hull** is :

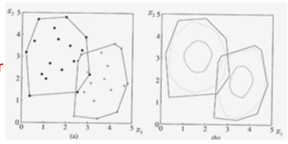
$$R\{X,\mu\} = \{y : y = \sum_{i=1}^{N} \lambda_i x_i : x_i \in X, \quad \sum_{i=1}^{N} \lambda_i = 1, \; 0 \le \lambda_i \le \mu \}$$

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

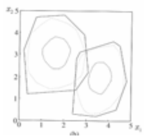
μ=1
Regular
convex
hulls

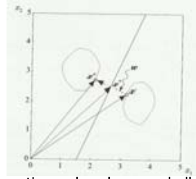


.....: μ=0.4
----: μ=0.1
Reduced
convex
hulls

- Data set with overlapping classes.
- For small enough values of μ, we can make the two reduced convex hulls non-overlapping.
- A 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

$$\arg\min_{\mathbf{w},\xi,b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \right\}$$

$$y_i(\mathbf{w}\cdot\mathbf{x}_i - b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

- Do you understand the different terms and criteria in the above minimization problem?
- Which points/samples turn out to be the support vectors?

SVMs: The nonlinear case intro.

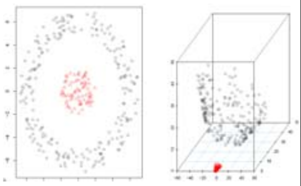
- The training samples are l-dimensional vectors; we have until now tried to find a linear separation in this l-dimensional feature space
- This seems quite limiting
- What if we increase the dimensionality (map our samples to a higher dimensional space) before applying our SVM?
- Perhaps we can find 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 not 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 can be separated by a (hyper)plane.
- The separating plane in 3D maps out an ellipse in the original 2D space



«Nonlinear»!

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

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

$$\max_{\lambda}\left(\sum_{i=1}^{N}\lambda_i-\frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j \underline{x_i^T x_j}\right)$$
$$\text{s.t.}\quad 0\le\lambda_i\le C\quad\forall i$$
$$\sum_{i=1}^{N}\lambda_i y_i=0$$

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

Called «kernel»

- If we have a function evaluating inner products, K($\mathbf{x}_i$,$\mathbf{x}_j$), we can ignore the samples themselves when solving the optimization
- Let's say we have K($\mathbf{x}_i$,$\mathbf{x}_j$) evaluating inner products in a higher dimensional space:

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

Useful kernels for classification

- Polynomial kernels

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

- Radial basis function kernels (very commonly used!)

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

Note the we need to set the σ parameter

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

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

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

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

The kernels give inner-product evaluations in the, possibly infinite-dimensional, transformed space.

The kernel formulation of the optimization function

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

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

- 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

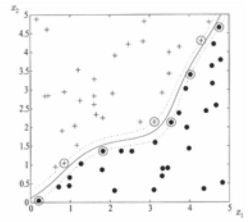


Figure 4.23 in
PR by Teodoridis et.al.

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/
  - Or use the PRTools toolbox http://www.37steps.com/prtools/
- Read the introductory guides.
- Often a radial basis function kernel is a good starting point.
- 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:
  - $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$
  - $\sigma = 2^{-15}, 2^{-13}, \ldots., 2^{3}$

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.
  - 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