

Machine Learning

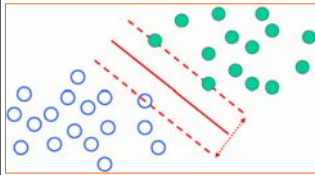
10-701/15-781, Fall 2006

Support Vector Machines

Eric Xing

Lecture 8, October 5, 2006

Reading: Chap. 6&7, C.B book



Outline

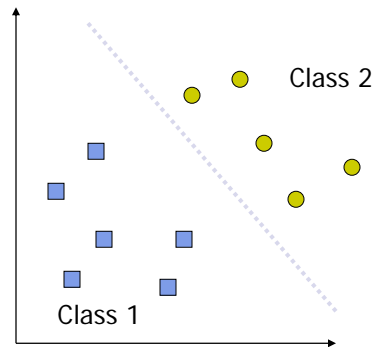
- Maximum margin classification
- Constrained optimization
- Lagrangian duality
- Kernel trick
- Non-separable cases



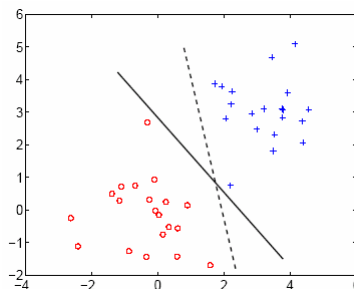
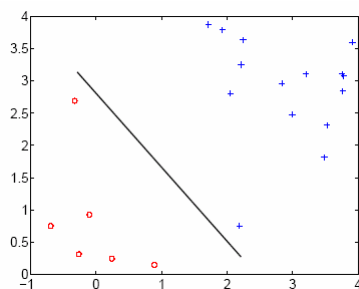
What is a good Decision Boundary?



- Consider a binary classification task with $y = \pm 1$ labels (not 0/1 as before).
- When the training examples are linearly separable, we can set the parameters of a linear classifier so that all the training examples are classified correctly
- Many decision boundaries!
 - Generative classifiers
 - Logistic regressions ...
- Are all decision boundaries equally good?



Examples of Bad Decision Boundaries



- Why we may have such boundaries?
 - Irregular distribution
 - Imbalanced training sizes
 - outliers

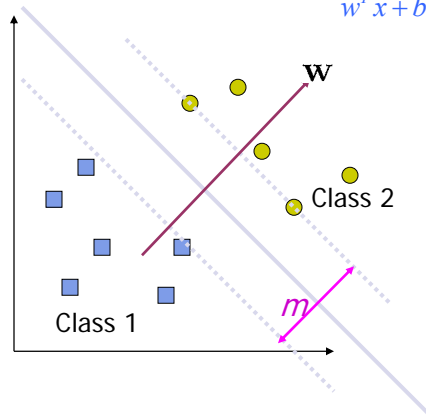
Classification and Margin



- Parameterizing decision boundary

- Let w denote a vector orthogonal to the decision boundary, and b denote a scalar "offset" term, then we can write the decision boundary as:

$$w^T x + b = 0$$



- Margin

$$\begin{aligned} w^T x + b &> 0 && \text{for all } x \text{ in class 2} \\ w^T x + b &< 0 && \text{for all } x \text{ in class 1} \end{aligned}$$

Or more compactly:

$$(w^T x_i + b) y_i > 0$$

The margin between two points

$$m = (w^T x_i + b) - (w^T x_j + b) = w^T (x_i - x_j)$$

Maximum Margin Classification



- The margin is:

$$m = w^T (x_{i^*} - x_{j^*})$$

- It make sense to set constrains on W :
- Here is our Maximum Margin Classification problem:

$$\begin{aligned} \max_{w,b} \quad & m \\ \text{s.t} \quad & y_i (w^T x_i + b) \geq m, \quad \forall i \\ & \|w\| = 1 \end{aligned}$$

- Equivalently, we can instead work on this:

$$\begin{aligned} \max_{w,b} \quad & \frac{m}{\|w\|} \\ \text{s.t} \quad & y_i (w^T x_i + b) \geq m, \quad \forall i \end{aligned}$$

Maximum Margin Classification, con'd.



- The optimization problem:

$$\begin{aligned} \max_{w,b} \quad & m \\ \text{s.t} \quad & y_i(w^T x_i + b) \geq m, \quad \forall i \end{aligned}$$

- But note that the magnitude of m merely scales w and b , and does not change the classification boundary at all!
- So we instead work on this cleaner problem:

$$\begin{aligned} \max_{w,b} \quad & 1 \\ \text{s.t} \quad & y_i(w^T x_i + b) \geq 1, \quad \forall i \end{aligned}$$

- The solution to this leads to the famous **Support Vector Machines** - -- believed by many to be the best "off-the-shelf" supervised learning algorithm

Support vector machine



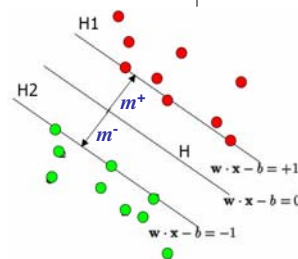
- A convex quadratic programming problem with linear constraints:

$$\begin{aligned} \max_{w,b} \quad & 1 \\ \text{s.t} \quad & y_i(w^T x_i + b) \geq 1, \quad \forall i \end{aligned}$$

- The attained margin is now given by $\frac{1}{\|w\|}$
- Only a few of the classification constraints are relevant → **support vectors**

- Constrained optimization

- We can directly solve this using commercial quadratic programming (QP) code
- But we want to take a more careful investigation of Lagrange duality, and the solution of the above is its dual form.
- deeper insight: support vectors, kernels ...
- more efficient algorithm



Lagrangian Duality



- The Primal Problem

$$\begin{array}{ll}\text{Primal:} & \min_w f(w) \\ & \text{s.t.} \quad g_i(w) \leq 0, \quad i = 1, \dots, k \\ & \quad \quad h_i(w) = 0, \quad i = 1, \dots, l\end{array}$$

The generalized Lagrangian:

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w)$$

the α 's ($\alpha_i \geq 0$) and β 's are called the Lagrangian multipliers

Lemma:

$$\max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{o/w} \end{cases}$$

A re-written Primal:

$$\min_w \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

Lagrangian Duality, cont.



- Recall the Primal Problem:

$$\min_w \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

- The Dual Problem:

$$\max_{\alpha, \beta, \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta)$$

- Theorem (weak duality):

$$d^* = \max_{\alpha, \beta, \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta) \leq \min_w \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = p^*$$

- Theorem (strong duality):

Iff there exist a saddle point of $\mathcal{L}(w, \alpha, \beta)$, we have

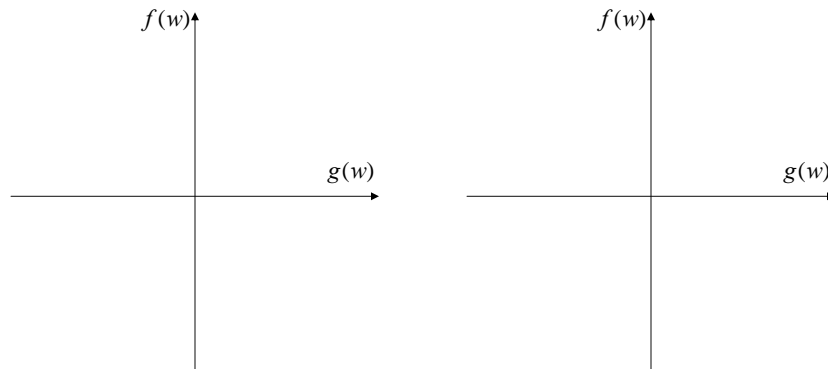
$$d^* = p^*$$

A sketch of strong and weak duality



- Now, ignoring $h(x)$ for simplicity, let's look at what's happening graphically in the duality theorems.

$$d^* = \max_{\alpha_i \geq 0} \min_w f(w) + \alpha^T g(w) \leq \min_w \max_{\alpha_i \geq 0} f(w) + \alpha^T g(w) = p^*$$



The KKT conditions



- If there exists some saddle point of \mathcal{L} , then the saddle point satisfies the following "Karush-Kuhn-Tucker" (KKT) conditions:

$$\frac{\partial}{\partial w_i} \mathcal{L}(w, \alpha, \beta) = 0, \quad i = 1, \dots, n$$

$$\frac{\partial}{\partial \beta_i} \mathcal{L}(w, \alpha, \beta) = 0, \quad i = 1, \dots, l$$

$$a_i g_i(w) = 0, \quad i = 1, \dots, k$$

$$g_i(w) \leq 0, \quad i = 1, \dots, k$$

$$\alpha_i \geq 0, \quad i = 1, \dots, k$$

- Theorem:** If w^* , α^* and β^* satisfy the KKT condition, then it is also a solution to the primal and the dual problems.

Solving optimal margin classifier



- Recall our opt problem:

$$\begin{aligned} \max_{w,b} \quad & \frac{1}{\|w\|} \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1, \quad \forall i \end{aligned}$$

- This is equivalent to

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} w^T w \\ \text{s.t.} \quad & 1 - y_i(w^T x_i + b) \leq 0, \quad \forall i \end{aligned} \quad (*)$$

- Write the Lagrangian:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^m \alpha_i [y_i(w^T x_i + b) - 1]$$

- Recall that (*) can be reformulated as $\min_{w,b} \max_{\alpha_i \geq 0} \mathcal{L}(w, b, \alpha)$
Now we solve its **dual problem**: $\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w, b, \alpha)$

The Dual Problem



$$\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w, b, \alpha)$$

- We minimize \mathcal{L} with respect to w and b first:

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^m \alpha_i y_i x_i = 0, \quad (*)$$

$$\nabla_b \mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i y_i = 0, \quad (**)$$

Note that (*) implies: $w = \sum_{i=1}^m \alpha_i y_i x_i \quad (***)$

- Plus (***) back to \mathcal{L} , and using (**), we have:

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

The Dual problem, cont.

- Now we have the following dual opt problem:

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

$$\text{s.t. } \alpha_i \geq 0, \quad i=1, \dots, k$$

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

- This is, (again,) a **quadratic programming** problem.

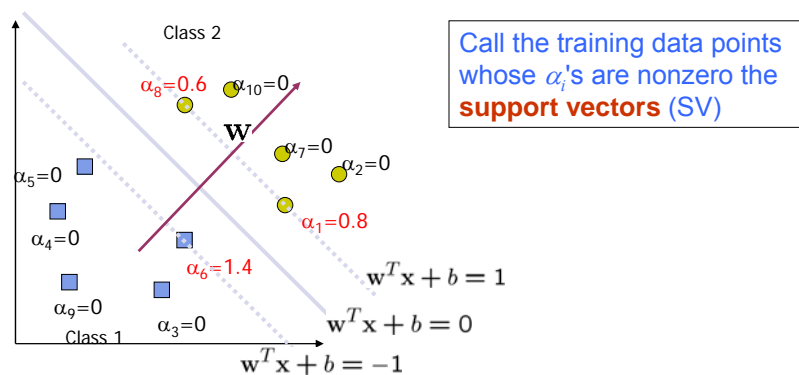
- A global maximum of α_i can always be found.
- But what's the big deal??
- Note two things:

- w** can be recovered by $\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$ See next ...
- The "kernel" $\mathbf{x}_i^T \mathbf{x}_j$ More later ...

Support vectors

- Note the KKT condition --- only a few α_i 's can be nonzero!!

$$\alpha_i g_i(\mathbf{w}) = 0, \quad i=1, \dots, k$$



Support vector machines



- Once we have the Lagrange multipliers $\{\alpha_i\}$, we can reconstruct the parameter vector w as a weighted combination of the training examples:

$$w = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

- For testing with a new data z

- Compute

$$w^T z + b = \sum_{i \in SV} \alpha_i y_i (\mathbf{x}_i^T z) + b$$

and classify z as class 1 if the sum is positive, and class 2 otherwise

- Note: w need not be formed explicitly

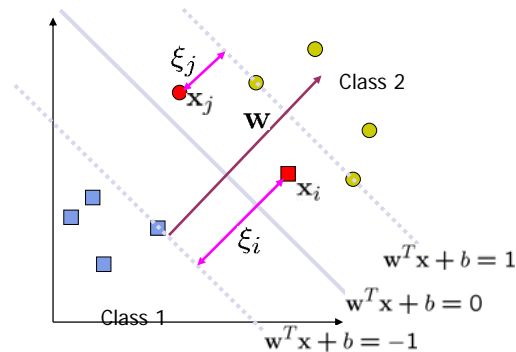
Interpretation of support vector machines



- The optimal w is a linear combination of a small number of data points. This “sparse” representation can be viewed as data compression as in the construction of kNN classifier
- To compute the weights $\{\alpha_i\}$, and to use support vector machines we need to specify only the inner products (or kernel) between the examples $\mathbf{x}_i^T \mathbf{x}_j$
- We make decisions by comparing each new example z with only the support vectors:

$$y^* = \text{sign} \left(\sum_{i \in SV} \alpha_i y_i (\mathbf{x}_i^T z) + b \right)$$

Non-linearly Separable Problems



- We allow “error” ξ_i in classification; it is based on the output of the discriminant function $w^T x + b$
- ξ_i approximates the number of misclassified samples

Soft Margin Hyperplane

- Now we have a slightly different opt problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned}$$

- ξ_i are “slack variables” in optimization
- Note that $\xi_i = 0$ if there is no error for x_i
- ξ_i is an upper bound of the number of errors
- C : tradeoff parameter between error and margin

The Optimization Problem



- The dual of this new constrained optimization problem is

$$\begin{aligned} \max_{\alpha} \quad & \mathcal{J}(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, k \\ & \sum_{i=1}^m \alpha_i y_i = 0. \end{aligned}$$

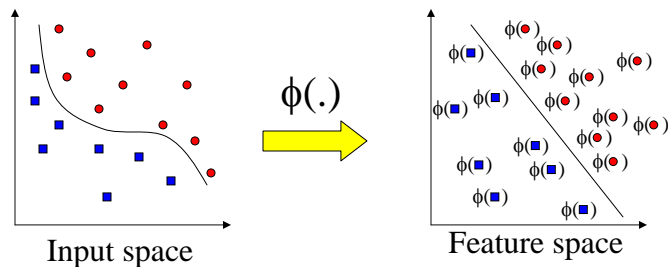
- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound C on α_i now
- Once again, a QP solver can be used to find α_i

Extension to Non-linear Decision Boundary



- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform \mathbf{x}_i to a higher dimensional space to “make life easier”
 - Input space: the space the point \mathbf{x}_i are located
 - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
 - Linear operation in the feature space is equivalent to non-linear operation in input space
 - Classification can become easier with a proper transformation. In the XOR problem, for example, adding a new feature of $x_1 x_2$ make the problem linearly separable (homework)

Transforming the Data



Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
 - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

The Kernel Trick

- Recall the SVM optimization problem

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

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \dots, k$$

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

- The data points only appear as **inner product**
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function K by $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

An Example for feature mapping and kernels



- Consider an input $\mathbf{x}=[x_1, x_2]$
- Suppose $\phi(\cdot)$ is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2$$

- An inner product in the feature space is

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right) \right\rangle =$$

- So, if we define the **kernel function** as follows, there is no need to carry out $\phi(\cdot)$ explicitly

$$K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^2$$

More examples of kernel functions



- Linear kernel (we've seen it)

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- Polynomial kernel (we just saw an example)

$$K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^p$$

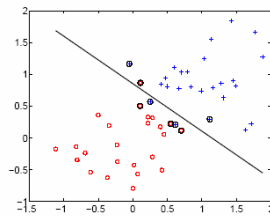
where $p = 2, 3, \dots$ To get the feature vectors we concatenate all p th order polynomial terms of the components of \mathbf{x} (weighted appropriately)

- Radial basis kernel

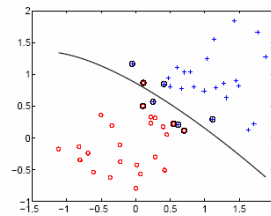
$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$$

In this case the feature space consists of functions and results in a non-parametric classifier.

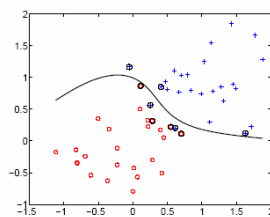
SVM examples



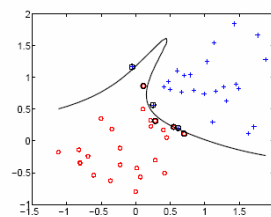
linear



2nd order polynomial

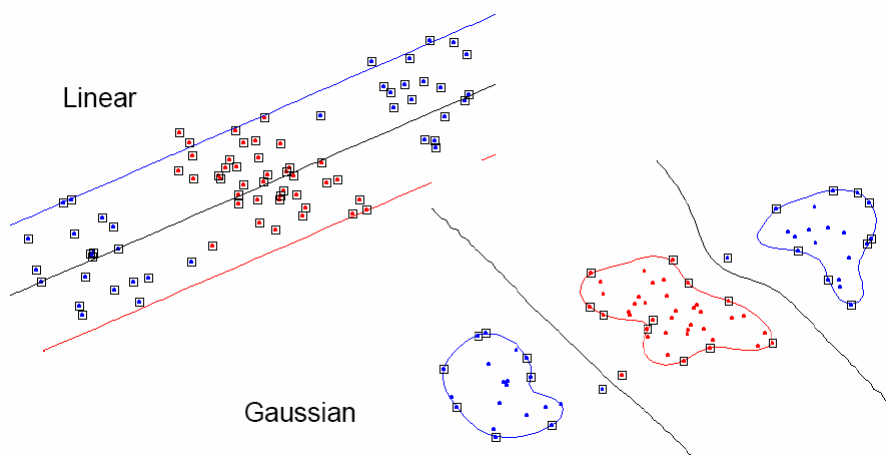


4th order polynomial



8th order polynomial

Examples for Non Linear SVMs – Gaussian Kernel



Cross-validation error



- The leave-one-out cross-validation error does not depend on the dimensionality of the feature space but only on the # of support vectors!

$$\text{Leave-one-out CV error} = \frac{\# \text{ support vectors}}{\# \text{ of training examples}}$$

