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
  - Outliners
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
  - $w^T x + b > 0$ for all $x$ in class 2
  - $w^T x + b < 0$ for all $x$ in class 1
  - Or more compactly:
    $$\left( w^T x + b \right)_i > 0$$

- 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:

$$\max_{w, b} \quad m$$

s.t

$$y_i (w^T x_i + b) \geq m, \quad \forall i$$
$$\|w\| = 1$$

- Equivalently, we can instead work on this:

$$\max_{w, b} \quad \frac{m}{\|w\|}$$

s.t

$$y_i (w^T x_i + b) \geq m, \quad \forall i$$
Maximum Margin Classification, con'd.

- The optimization problem:
  \[
  \max_{w,b} \quad \frac{m}{\|w\|} \\
  \text{s.t} \quad y_i(w^T x_i + b) \geq m, \quad \forall i
  \]
  - 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:
  \[
  \max_{w,b} \quad \frac{1}{\|w\|} \\
  \text{s.t} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i
  \]
  - 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 constrains:
  \[
  \max_{w,b} \quad \frac{1}{\|w\|} \\
  \text{s.t} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i
  \]
  - The attained margin is now given by \(\frac{1}{\|w\|}\)
  - Only a few of the classification constraints are relevant \(\Rightarrow\) **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.
  \(\Rightarrow\) deeper insight: support vectors, kernels …
  \(\Rightarrow\) more efficient algorithm
Lagrangian Duality

- The Primal Problem
  \[
  \min_w \ f(w) \\
  \text{s.t.} \ g_i(w) \leq 0, \ i = 1, \ldots, k \quad \text{and} \quad h_i(w) = 0, \ i = 1, \ldots, l
  \]

  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 \( \alpha \)'s (\( \alpha \geq 0 \)) and \( \beta \)'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):
  If 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.

\[
d^* = \max_{w \geq 0} \min_w f(w) + \alpha^T g(w) \leq \min_{w} \max_{\alpha \geq 0} f(w) + \alpha^T g(w) = \rho^*
\]

The KKT conditions

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

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

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

\[
\alpha_i g_i(w) = 0, \quad i = 1, \ldots, k
\]

\[
g_i(w) \leq 0, \quad i = 1, \ldots, k
\]

\[
\alpha_i \geq 0, \quad i = 1, \ldots, k
\]

- **Theorem**: If \( w^*, \alpha^* \) and \( \beta^* \) 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:
  \[
  \begin{align*}
  \max_{w,b} & \quad \frac{1}{2} \|w\|^2 \\
  \text{s.t.} & \quad y_i (w^T x_i + b) \geq 1, \quad \forall i
  \end{align*}
  \]

- This is equivalent to
  \[
  \begin{align*}
  \min_{w,b} & \quad \frac{1}{2} w^T w \\
  \text{s.t.} & \quad 1 - y_i (w^T x_i + b) \leq 0, \quad \forall i
  \end{align*} \tag{\*}
  \]

- 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 \( \min_{w,b} \max_{\alpha \geq 0} \mathcal{L}(w, b, \alpha) \)
  - Now we solve its dual problem: \( \max_{\alpha \geq 0} \min_{w,b} \mathcal{L}(w, b, \alpha) \)

The Dual Problem

\[
\max_{\alpha \geq 0} \min_{w,b} \mathcal{L}(w, b, \alpha)
\]

- We minimize \( \mathcal{L} \) with respect to \( w \) and \( b \) first:
  \[
  \begin{align*}
  \nabla_w \mathcal{L}(w, b, \alpha) &= w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0, \tag{\*} \\
  \nabla_b \mathcal{L}(w, b, \alpha) &= \sum_{i=1}^{m} \alpha_i y_i = 0, \tag{\**}
  \end{align*}
  \]

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

- 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:
  \[
  \max_{\alpha} J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j)
  \]
  s.t. \( \alpha_i \geq 0, \quad i = 1, \ldots, k \)
  \[
  \sum_{i=1}^{m} \alpha_i y_i = 0.
  \]

- This is, (again,) a quadratic programming problem.
  - A global maximum of \( \alpha_i \) can always be found.
  - But what’s the big deal??
  - Note two things:
    1. \( w \) can be recovered by \( w = \sum_{i=1}^{m} \alpha_i y_i x_i \)
    2. The "kernel" \( x_i^T x_j \)

Support vectors

- Note the KKT condition --- only a few \( \alpha_i \)’s can be nonzero!!
  \[
  \alpha_i g_i(w) = 0, \quad i = 1, \ldots, k
  \]

Call the training data points whose \( \alpha_i \)'s are nonzero the support vectors (SV)
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:

\[
w = \sum_{i \in SV} \alpha_i y_i x_i
\]

- For testing with a new data \( z \)
  - Compute
    \[
    w^T z + b = \sum_{i \in SV} \alpha_i y_i (x_i^T z) + b
    \]
    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 \( x_i^T x_j \)

- We make decisions by comparing each new example \( z \) with only the support vectors:

\[
y^* = \text{sign} \left( \sum_{i \in SV} \alpha_i y_i (x_i^T z) + b \right)
\]
Non-linearly Separable Problems

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

Soft Margin Hyperplane

- Now we have a slightly different optimization problem:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i$$

s.t. $y_i(w^T x_i + b) \geq 1 - \xi_i$, $\forall i$

- $\xi_i \geq 0$, $\forall i$

- $\xi_i$ are “slack variables” in optimization
- Note that $\xi_i=0$ if there is no error for $x_i$
- $\xi_i$ 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

\[
\max_{\alpha} \quad J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j) \\
\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, k \\
\sum_{i=1}^{m} \alpha_i y_i = 0.
\]

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

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 \( x_i \) to a higher dimensional space to “make life easier”
  - Input space: the space the point \( x_i \) are located
  - Feature space: the space of \( \phi(x_i) \) 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 \( x_1x_2 \) make the problem linearly separable (homework)
Transforming the Data

- 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
  \[
  \max_{\alpha} \quad J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j)
  \]
  s.t. \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, k
  \quad \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 \( K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \)
An Example for feature mapping and kernels

- Consider an input $x = [x_1, x_2]$
- Suppose $\phi(.)$ 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
  $$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right)\right\rangle = \left(1 + x_1^T x_1'ight)^2$$
- So, if we define the kernel function as follows, there is no need to carry out $\phi(.)$ explicitly
  $$K(x, x') = \left(1 + x^T x'\right)^2$$

More examples of kernel functions

- Linear kernel (we’ve seen it)
  $$K(x, x') = x^T x'$$
- Polynomial kernel (we just saw an example)
  $$K(x, x') = \left(1 + x^T x'\right)^p$$
  where $p = 2, 3, \ldots$ To get the feature vectors we concatenate all $p$th order polynomial terms of the components of $x$ (weighted appropriately)
- Radial basis kernel
  $$K(x, x') = \exp\left(-\frac{1}{2} \|x - x'\|^2\right)$$
  In this case the feature space consists of functions and results in a non-parametric classifier.
Kernelized SVM

• Training:

\[
\max_{\alpha} \quad J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j)
\]

s.t. \(0 \leq \alpha_i \leq C, \quad i = 1, \ldots, k\)

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

• Using:

\[
y^* = \text{sign} \left( \sum_{i \in \mathcal{S}_F} \alpha_i y_i K(x_i, z) + b \right)
\]
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 number of support vectors!

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