Lecture Outline

Reading: Chapter 7 Sparse Kernel Machines

This lecture covers

- Overlapping class distributions
- Solving the quadratic programming problem
- Multiclass SVMs
- SVMs for Regression

7.1.1 Overlapping Class Distributions

So far, we have assumed that the training data points are linearly separable in the feature space $\boldsymbol{\phi}(\mathbf{x})$. In practice linearly separable data is unlikely. We therefore need a way to modify the SVM so as to allow some of the training points to be misclassified but to penalize (in the cost function) for being on the 'wrong side' of the margin boundary. The penalty should increase with the distance from the boundary and it is convenient to make this penalty a linear function of this distance.

We introduce *slack variables*, $\xi_n = |t_n - y(\mathbf{x}_n)| \geq 0$ where $n = 0, \ldots, N$ with one slack variable for each training data point. Numerical values for the slack variables indicate classification and position with respect to the margin boundary:

- $\xi_n = 0$ indicates training data points are correctly classified and are either on the margin or inside the correct *margin* boundary
- $0 < \xi_n \leq 1$ indicates training data points are correctly classified and lie inside the margin but on the correct side of the *decision* boundary
- $\xi_n > 1$ indicates training data points are misclassified and lie on the wrong side of the *decision* boundary

This is illustrated in Figure 1. This is sometimes described as relaxing the hard margin constraint to give a *soft margin* and allows some of the training set data points to be misclassified.

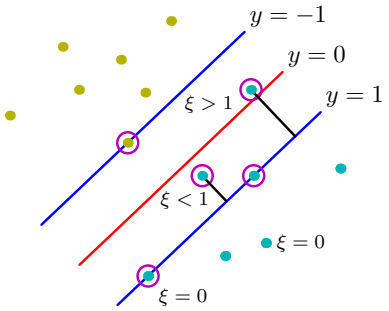

Figure 1:

The exact constraint

$$t_n y(\mathbf{x}_n) \geq 1, \quad n = 1, \ldots, N \tag{1}$$

is now replaced with

$$t_n y(\mathbf{x}_n) \quad \geq \quad 1 - \xi_n, \quad n = 1, \ldots, N. \tag{2}$$

In effect, we are relaxing the hard margin constraint to give a *soft margin* and allow some of the training data points to be misclassified.

The corresponding Lagrangian is given by

$$L(\mathbf{w}, b, \mathbf{a}) \quad = \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n \left\{t_n(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + b) - 1 + \xi_n\right\} - \sum_{n=1}^{N}\mu_n\xi_n \tag{3}$$

where a_n and μ_n are Lagrange multipliers, $C > 0$ is a parameter that controls the trade-off between the slack variable penalty and the margin, and the constraints are given in the text.

The *dual formulation* is given by (see text for derivation)

$$\tilde{L}(\mathbf{a}) \quad = \quad \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \tag{4}$$

where we have eliminated $\mathbf{w}$, b, and μ_n and is identical to before except the the constraints are now

$$0 \leq a_n \leq C \tag{5}$$

$$\sum_{n=1}^{N} a_n t_n = 0. \tag{6}$$

Methods for actually solving the quadratic programming problem are discussed in the next section.

Solving the Quadratic Programming Problem

Although predictions for new inputs are made using only the support vectors, the training phase (computation of the SVM weights, $\mathbf{a}$ and bias b) makes use of the whole data set. It is important, therefore, the have efficient algorithms for solving the quadratic programming problem.

The technique of *chunking* exploits the fact that the value of the Lagrangian is unchanged if we remove the rows and columns of the kernel matrix corresponding to Lagrange multipliers that have value zero. This allows the full quadratic programming problem to be broken down into a series of smaller ones, whose goal is eventually to identify all of the nonzero Lagrange multipliers and discard the others.

The concept of chunking can be taken to the extreme limit by considering just two Lagrange multipliers at a time. This popular approach to training SVMs is known as *sequential minimal optimization* (SMO). With two Lagrange multipliers, the subproblem can be solved analytically, thereby avoiding numerical quadratic programming altogether. Heuristics are given for choosing the pairs of Lagrange multipliers to be considered at each step. In practice, SMO is found to have a scaling with the number of training points that is somewhere between linear and quadratic depending on the particular application.

The MATLAB Statistics Toolbox has SVM functions. The Statistical Pattern Recognition Toolbox (STPR)

http://cmp.felk.cvut.cz/cmp/software/stprtool/

has an implementation of smo and a MATLAB interface to SVM^{light}.

http://svmlight.joachims.org

Finally, libsvm is a popular SVM library

http://www.csie.ntu.edu.tw/~cjlin/libsvm/

7.1.3 Multiclass SVMs

The SVM is fundamentally a two-class (binary) classifier. In practice, however, we often have to tackle problems involving $K > 2$ classes. Various methods have therefore been proposed for combining multiple two-class SVMs in order to build a multiclass classifier.

One-Versus-the-Rest

One common approach is to construct K separate SVMs, in which the kth model $y_k(\mathbf{x})$ is trained using data from class $\mathcal{C}_k$ as the positive ($t_n = +1$) examples and data from the remaining $K - 1$ classes as the negative ($t_n = -1$) examples. This known as the *one-versus-the-rest* or *one-versus-all* approach. However, in Figure 4.2 we saw that using the decisions of the individual classifiers can lead to inconsistent results in which an input is assigned to multiple classes simultaneously.

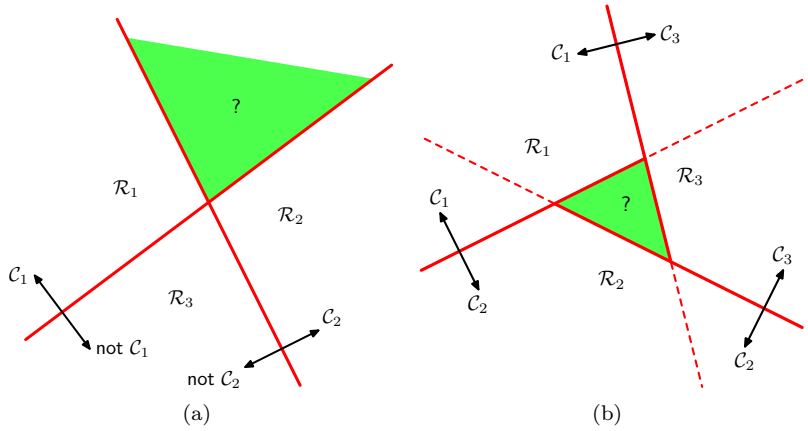

Figure 2:

In practice, this problem is usually addressed by making predictions for new inputs $\mathbf{x}$ using

$$y(\mathbf{x}) \quad = \quad \max_k y_k(\mathbf{x}). \tag{7}$$

Unfortunately, this heuristic approach suffers from the problem that the different classifiers were trained on different tasks, and there is no guarantee that the real-valued quantities $y_k(\mathbf{x})$ for different classifiers will have the appropriate scales.

Another problem with the one-versus-the-rest approach is that the training sets are imbalanced. For instance, if we have ten classes each with equal numbers of training points, then the individual classifiers are trained on data sets comprising 90% negative examples and only 10% positive examples, and the symmetry of the original problem is lost. A simple variant is to modify the target values so that the positive class has target $+1$ and the negative class has target $-1/(K - 1)$.

One-Versus-One

Another approach is to train $K(K - 1)/2$ different two-class SVMs on all posssible pairs of classes and then classify test points according to which class has the highest number of 'votes'. This approach that is sometimes called *one-versus-one*. As we saw in Figure 4.2, this can also lead to ambiguities in the resulting classification not to mention significantly increased training and test computation.

7.1.4 SVMs for Regression

We now extend SVMs to regression problems while at the same time preserving the property of sparseness. In simple linear regression, we minimize a regularized quadratic (LS) error function given by

$$\frac{1}{2}\sum_{n=1}^{N}\{y(\mathbf{x}_n)-t_n\}^2+\frac{\lambda}{2}\|\mathbf{w}\|^2. \tag{8}$$

To obtain sparse solutions, the quadratic error function is replaced by an ϵ-*insensitive error function* which gives zero error if the prediction is within $\pm\epsilon$ of the target, i.e. $y(\mathbf{x})-\epsilon\leq t\leq y(\mathbf{x})+\epsilon$:

$$E_\epsilon(y(\mathbf{x})-t) \quad = \quad \begin{cases} 0, & \text{if } |y(\mathbf{x})-t|<\epsilon \\ |y(\mathbf{x})-t|-\epsilon, & \text{otherwise.} \end{cases} \tag{9}$$

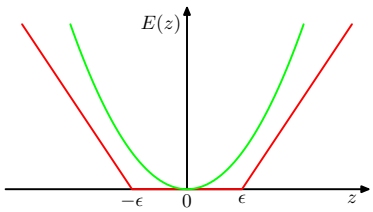

Figure 6:

In the SVM for regression we therefore minimize a regularized error function given by

$$C\sum_{n=1}^{N}E_\epsilon(y(\mathbf{x}_n)-t_n)+\frac{1}{2}\|\mathbf{w}\|^2 \tag{10}$$

where

$$y(\mathbf{x}) \quad = \quad \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})+b. \tag{11}$$

This error function is optimized for finding a model so all predictions with training data are within $\pm\epsilon$ of the targets.

Since not all training data points fall into the ϵ neighborhood, we penalize points that fall outside through the use of slack variables. As before, we re-express the optimization using two sets of slack variables so that $\xi_n>0$ corresponds to a point for which $t_n>y(\mathbf{x}_n)+\epsilon$ and $\hat{\xi}_n>0$ corresponds to a point for which $t_n<y(\mathbf{x}_n)-\epsilon$ as illustrated in Figure 7.

The condition for a point to lie inside an ϵ-tube is

$$y(\mathbf{x}_n)-\epsilon\leq t_n\leq y(\mathbf{x}_n)+\epsilon. \tag{12}$$

Introducing slack variables allows points outside the tube provided the slack variables are non-zero, i.e. for $\xi>0$, $t_n>y(\mathbf{x}_n)+\epsilon$. The corresponding conditions are

$$t_n\leq y(\mathbf{x}_n)+\epsilon+\xi_n \tag{13}$$

$$t_n\geq y(\mathbf{x}_n)-\epsilon-\hat{\xi}_n. \tag{14}$$

The revised error function to be minimized is

$$C\sum_{n=1}^{N}(\xi_n+\hat{\xi}_n)+\frac{1}{2}\|\mathbf{w}\|^2 \tag{15}$$

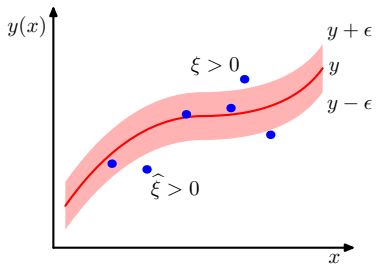

Figure 7:

subject to (13), (14), $\xi_n \geq 0$ and $\hat{\xi}_n \geq 0$.

The optimization problem is solved through the Lagrangian

$$\begin{aligned} L &= C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N}(\mu_n\xi_n + \hat{\mu}_n\hat{\xi}_n) - \\ &\quad \sum_{n=1}^{N} a_n(\epsilon + \xi_n + y_n - t_n) - \sum_{n=1}^{N}\hat{a}_n(\epsilon + \hat{\xi}_n + y_n - t_n) \end{aligned} \tag{16}$$

where $a_{n\geq 0}$, $\hat{a}_n \geq 0$, $\mu_n \geq 0$, $\hat{\mu}_n \geq 0$ are the Lagrange multipliers.

The dual formulation is expressed in terms of the kernel function

$$\tilde{L}(\mathbf{a}, \tilde{\mathbf{a}}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a}_m)k(\mathbf{x}_n, \mathbf{x}_m) - \epsilon\sum_{n=1}^{N}(a_n + \hat{a}_n) + \sum_{n=1}^{N}(a_n + \hat{a}_n)t_n. \tag{17}$$

Once the dual formulation has been solved and a_n and $\hat{a}_n$ are determined, predictions for new inputs $\mathbf{x}'$ can be made using (see text for how to solve for b)

$$y(\mathbf{x}') = \sum_{n=1}^{N}(a_n - \hat{a}_n)k(\mathbf{x}', \mathbf{x}_n) + b \tag{18}$$

As it turns out for every training data point $\mathbf{x}_n$, either a_n or $\hat{a}_n$ (or both) are zero. Those training points for which either $a_n \neq 0$ or $\hat{a}_n \neq 0$ lie on the boundary of the ϵ-tube or outside the tube. Those training points for which $a_n = \hat{a}_n = 0$ lie within the tube. We again have a sparse solution, and the only terms that have to be evaluated in (18) are those that involve the support vectors.

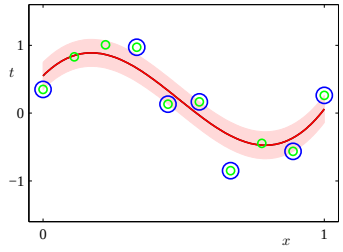

Figure 8: