67577 – Intro. to Machine Learning **Fall semester, 2003/4**

Lecture 5: Support Vector Machines and Kernel Functions II

Lecturer: Amnon Shashua *Scribe: Amnon Shashua* [1]

5.1 Support Vector Machine

We return now to the primal problem (eqn. 4.3) representing the maximal margin separating hyperplane with margin errors:

$$\begin{aligned} \min_{\mathbf{w},b,\epsilon_i} \quad & \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + \nu\sum_{i=1}^{l}\epsilon_i \\ & \textit{subject to} \\ & y_i(\mathbf{w}\cdot\mathbf{x}_i - b) \geq 1-\epsilon_i \qquad i=1,...,l \\ & \epsilon_i \geq 0 \end{aligned}$$

We will now derive the Lagrangian Dual of this problem. By doing so a new key property will emerge facilitated by the fact that the criteria function $\theta(\mu)$ (note there are no equality constraints thus there is no need for λ) involves only inner-products of the training instance vectors $\mathbf{x}_i$. This property will form the key of mapping the original input space of dimension n to a higher dimensional space thereby allowing for non-linear decision surfaces for separating the training data.

Note that with this particular problem the strong duality conditions are satisfied because the criteria function and the inequality constraints form a convex set. The Lagrangian takes the following form:

$$L(\mathbf{w},b,\epsilon_i,\mu) = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + \nu\sum_{i=1}^{l}\epsilon_i - \sum_{i=1}^{l}\mu_i\left[y_i(\mathbf{w}\cdot\mathbf{x}_i - b) - 1 + \epsilon_i\right] - \sum_{i=1}^{l}\mu_{l+i}\epsilon_i$$

Recall that

$$\theta(\mu) = \min_{\mathbf{w},b,\epsilon_i} L(\mathbf{w},b,\epsilon_i,\mu).$$

Since the minimum is obtained at the vanishing partial derivatives of the Lagrangian with respect to $\mathbf{w}, b$, the next step would be to evaluate those constraints and substitute them back into $L()$ to obtain $\theta(\mu)$:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \mu_i y_i \mathbf{x}_i = 0 \tag{5.1}$$

$$\frac{\partial L}{\partial b} = \sum_i \mu_i y_i = 0 \tag{5.2}$$

$$\frac{\partial L}{\partial \epsilon_i} = \nu - \mu_i - \mu_{l+i} = 0 \tag{5.3}$$

[1]

From the first constraint (5.1) we obtain $\mathbf{w} = \sum_i \mu_i y_i \mathbf{x}_i$, that is, $\mathbf{w}$ is described by a linear combination of a *subset* of the training instances. The reason that not all instances participate in the linear superposition is due to the KKT conditions: $\mu_i = 0$ when $y_i(\mathbf{w} \cdot \mathbf{x}_i - b) > 1$, i.e., the instance $\mathbf{x}_i$ is classified correctly and is not a boundary point, and conversely, $\mu_i > 0$ when $y_i(\mathbf{w} \cdot \mathbf{x}_i - b) = 1 - \epsilon_i$, i.e., when $\mathbf{x}_i$ is a boundary point or when $\mathbf{x}_i$ is a margin error ($\epsilon_i > 0$) — note that for a margin error instance the value of ϵ_i would be the smallest possible required to reach an equality in the constraint because the criteria function penalizes large values of ϵ_i. The boundary points (and the margin errors) are called *support vectors* thus $\mathbf{w}$ is defined by the support vectors *only*. The third constraint (5.3) is equivalent to the constraint:

$$0 \leq \mu_i \leq \nu \qquad i = 1, ..., l,$$

since $\mu_{l+i} \geq 0$. Substituting these results/constraints back into the Lagrangian $L()$ we obtain the *dual problem*:

$$\begin{aligned} \max_{\mu_1,...,\mu_l} \quad & \theta(\mu) = \sum_{i=1}^{l} \mu_i - \frac{1}{2} \sum_{i,j} \mu_i \mu_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ & \textit{subject to} \\ & 0 \leq \mu_i \leq \nu \qquad i = 1, ..., l \\ & \sum_{i=1}^{l} y_i \mu_i = 0 \end{aligned} \tag{5.4}$$

The criterion function $\theta(\mu)$ can be written in a more compact manner as follows: Let M be a $l \times l$ matrix whose entries are $M_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$ then $\theta(\mu) = \mu \cdot \mathbf{1} - \frac{1}{2}\mu^\top M \mu$ where $\mathbf{1}$ is the vector of $(1, ..., 1)$ and μ is the vector $(\mu_1, ..., \mu_l)$ and $\mu^\top$ is the transpose (row vector). Note that M is *positive definite*, i.e., $\mathbf{x}^\top M \mathbf{x} > 0$ for all vectors $\mathbf{x} \neq 0$ — a property which will be important later.

The key feature of the dual problem is not so much that it is simpler than the primal (in fact it isn't since the primal has no equality constraints) or that it has a more "elegant" feel, the key feature is that the problem is completely described by the inner products of the training instances $\mathbf{x}_i$, $i = 1, ..., l$. This fact will be shown to be a crucial ingredient in the so called "kernel trick" for the computation of inner-products in high dimensional spaces using simple functions defined on pairs of training instances.

5.2 The Kernel Trick

We ended with the dual formulation of the SVM problem and noticed that the input data vectors $\mathbf{x}_i$ are represented by the Gram matrix M. In other words, only inner-products of the input vectors play a role in the dual formulation — there is no explicit use of $\mathbf{x}_i$ or any other function of $\mathbf{x}_i$ besides inner-products. This observation suggests the use of what is known as the "kernel trick" to replace the inner-products by non-linear functions.

The common principle of kernel methods is to construct nonlinear variants of linear algorithms by substituting inner-products by nonlinear kernel functions. Under certain conditions this process can be interpreted as mapping of the original measurement vectors (so called "input space") onto some higher dimensional space (possibly infinitely high) commonly referred to as the "feature space". Mathematically, the kernel approach is defined as follows: let $\mathbf{x}_1, ..., \mathbf{x}_l$ be vectors in the input space, say R^n, and consider a mapping $\phi(\mathbf{x}) : R^n \to \mathcal{F}$ where $\mathcal{F}$ is an inner-product space.

The kernel-trick is to calculate the inner-product in $\mathcal{F}$ using a kernel function $k : R^n \times R^n \rightarrow R$, $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$, while avoiding explicit mappings (evaluation of) $\phi()$.

Common choices of kernel selection include the d'th order polynomial kernels $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + \theta)^d$ and the Gaussian RBF kernels $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2)$. If an algorithm can be restated such that the input vectors appear in terms of inner-products only, one can substitute the inner-products by such a kernel function. The resulting kernel algorithm can be interpreted as running the original algorithm on the space $\mathcal{F}$ of mapped objects $\phi(\mathbf{x})$.

We know that M of the dual form is semi-positive definite because M can be written is $M = Q^\top Q$ where $Q = [y_1\mathbf{x}_1, ..., y_l\mathbf{x}_l]$. Therefore $\mathbf{x}^\top M\mathbf{x} = \|Q\mathbf{x}\|^2 \geq 0$ for all choices of $\mathbf{x}$. If the entries of M are to be replaced with $y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$ then the condition we must enforce on the function $k()$ is that it is a *positive definite kernel* function. A positive definite function is defined such that for any set of vectors $\mathbf{x}_1, ..., \mathbf{x}_q$ and for any values of q the matrix K whose entries are $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ is semi positive definite.

Consider for example the polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^d$ and consider the case where $n = d = 2$:

$$k(\mathbf{x}, \mathbf{x}') = (1 + x_1x_1' + x_2x_2')^2 = \phi(\mathbf{x})^\top \phi(\mathbf{x}'),$$

where

$$\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)^\top$$

is mapping from R^2 to R^6 holding all the monomials up to and including the d'th order ones — i.e., all terms of the form $x_1^{p_1} \cdots x_n^{p_n}$ where $p_i \geq 0$ and $\sum p_i \leq d$. Hyperplanes $\phi(\mathbf{w})^\top \phi(\mathbf{x}) - b = 0$ in R^6 correspond to *conics*

$$(w_1^2)x_1^2 + (w_2^2)x_2^2 + (2w_1w_2)x_1x_2 + (2w_1)x_1 + (2w_2)x_2 + (1 - b) = 0$$

in R^2. Assume we would like to find a separating *conic* function rather than a line in R^2. The discussion so far suggests we construct the Gram matrix M in the dual form with the $d = 2$ polynomial kernel. The extra effort we will need to invest is negligible — simply replace every occurrence $\mathbf{x}_i^\top \mathbf{x}_j$ with $(\mathbf{x}_i^\top \mathbf{x}_j + 1)^2$.

We then proceed to solve for $\phi(\mathbf{w})$ and b using some QLP solver. The QLP solution of the dual form will yield the solution for the Lagrange multipliers $\mu_1, ..., \mu_l$. We saw from eqn. (5.1) that we can express $\phi(\mathbf{w})$ as a function of the (mapped) examples:

$$\phi(\mathbf{w}) = \sum_i \mu_i y_i \phi(\mathbf{x}_i).$$

Rather than explicitly representing $\phi(\mathbf{w})$ — a task which may be prohibitly expensive since in general the dimension of the feature space of a polynomial mapping is $\binom{n+d}{d}$ — we store all the support vectors (those input vectors with corresponding $\mu_i > 0$) and use them for the evaluation of test examples:

$$f(\mathbf{x}) = sign(\phi(\mathbf{w})^\top \phi(\mathbf{x}) - b) = sign(\sum_i \mu_i y_i \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}) - b) \quad (5.5)$$

$$= sign(\sum_i \mu_i y_i k(\mathbf{x}_i, \mathbf{x}) - b). \quad (5.6)$$

We see that the kernel trick enabled us to look for a non-linear separating surface by making an implicit mapping of the input space onto a higher dimensional feature space using the same dual form of the SVM formulation — the only change required was in the way the Gram matrix was

constructed. The price paid for this convenience is to carry *all* the support vectors at the time of classification $f(\mathbf{x})$.

A couple of notes may be worthwhile at this point. The constant b can be recovered from any of the support vectors. Say, $\mathbf{x}^+$ is a positive support vector (but not a margin errors, i.e., $\epsilon = 0$). Then $\phi(\mathbf{w})^\top \phi(\mathbf{x}^+) - b = 1$ from which b can be recovered. The second note is that the number of support vectors is typically around 10% of the number of training examples (empirically). Thus the computational load during evaluation of $f(\mathbf{x})$ may be relatively high. Approximations have been proposed in the literature by looking for a reduced number of support vectors (not necessarily aligned with the training set) — but this is beyond the scope of this course.

The kernel trick gained its popularity with the introduction of the SVM but since then has taken a life of its own and has been applied to principal component analysis (PCA), ridge regression, canonical correlation analysis (CCA), QR factorization and the list goes on. We will meet again with the kernel trick in class 7.