**01: Pattern Recognition and Machine Learning** **May 2016**

# Lecture 1: Sparse Kernel Machines

*Author: Christopher M. Bishop* *Scribes: scribe-name1,2,3*

**Note**: *Reading notes for pattern recognition and machine learning.*



In this chapter we shall look at kernel-based algorithms that have sparse solutions, so that predictions for new inputs depend only on the kernel function evaluated at a **subset** of the training data point.

First the detail of SVM, in which the determination of the model parameters corresponds to a convex optimization problem - so that any local solution is also a global optimum.

SVM is a decision machine and so does not provide posterior probabilities. RVM proposed based on a Bayesian formulation and provides posterior probabilistic outputs - as well as having typical much sparser solutions than the SVM.

## 1.1 Maximum Margin Classifiers

Two-class classification problem:

$$y(x) = w^T\phi(x) + b \tag{1.1}$$

where $\phi$ denotes a fixed feature-space transformation, $b$ explicit.

Training set comprises N input vectors $x_1, \cdots, x_N$ with target $t_1, /cdots, t_N$ where $t_N \in \{-1, 1\}$ and new data points are classified according to the sign of $y(x)$.

Training data - linearly separable in feature space, so that there exits one choice of $w$ and $b$ such that by 1.1 satisfies $y(x_n) > 0$ for points having $t_n = +1$ and $y(x_n) < 0$ for $t_n = -1$.

In section 4, the perceptron algorithm is guaranteed to find a solution for the linearly separable problem. The solution deeps on the initial value and the order of the data points are presented - if there are multiple solutions we should try to find one with the smallest generalizaiton error. SVM approaches this problem through the concept of the **margin** - the smallest distance between the decision boundary and any of the samples.

Perpendicular distance of a point $x$ from a hyperplane: $|y(x)|/\|w\|$. We are only interested in solutions for which all data points are correctly classified ($t_n y(x_n)) > 0$ for all n). So we have the distance of a point $x_n$ to the decision surface is given by:

$$\frac{t_n y(x_n)}{\|w\|} = \frac{t_n(w^T\phi(x_n) + b)}{\|w\|} \tag{1.2}$$

The margin is given by the perpendicular distance to the closest point $x_n$ from the data set, $w$ and $b$ need optimized in order to maximize the distance.

Then the maximum margin solution is found by

$$\arg\max_{w,b}\{\frac{1}{\|w\|}\min_n[t_n(w^T\phi(x_n) + b)]\} \tag{1.3}$$

in which $\|w\|$ does not depend on $n$. The direct solution of this optimization problem would be complex $\Rightarrow$ an equivalent problem easier to solve:

$$w \to \kappa w \tag{1.4}$$

$$b \to \kappa b \tag{1.5}$$

the rescaling would not change the distance. We then use this freedom to set

$$t_n(w^T\phi(x_n) + b) = 1 \tag{1.6}$$

for the point that is closest to the surface.

The canonical representation of the decision hyperplane:

$$t_n(w^T\phi(x_n) + b) \geq 1, n = 1, \cdots, N \tag{1.7}$$

When the equality holds, the constraints are said to be active.

There would always be at least one active constraint, because there will always be a closest point, and once the margin has been maximized there will be at leat two active constraints. The optimization problem minimizing $\|w\|^2$ is equivalent to maximize $\|w\|^{-1}$, so we need to solve the problem:

$$\arg\min_{w,b} \frac{1}{2}\|w\|^2 \tag{1.8}$$

subject to the constraints given by 1.7.

We introduce the Lagrange multipliers $a_n \geq 0$ with one multiplier $a_n$ for each of the constraints in1.7 giving the Lagrangian function:

$$L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} a_n\{t_n(w^T\phi(x_n) + b) - 1\} \tag{1.9}$$

where $a = (a_1, \cdots, a_N)^T$.

Setting the derivatives of $L(w, b, a)$ with respect to $w$ abd $b$ equal to zero, we obtain the following conditions:

$$w = \sum_{n=1}^{N} a_n t_n \phi(x_n) \tag{1.10}$$

$$0 = \sum_{n=1}^{N} a_n t_n \tag{1.11}$$

Eliminating $w$ and $b$ from $L(w, b, a)$ using these conditions then gives the dual representation of the **maximum margin problem** in which we maximize

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m) \tag{1.12}$$

with respect to a subject to the constraints

$$a_n \geq 0, n = 1, \cdots, N \tag{1.13}$$

$$\sum_{n=1}^{N} a_n t_n = 0 \tag{1.14}$$

Here the kernel function is defined by $k(x, x') = \phi(x)^T \phi(x')$. This takes the form of a **quadratic programming problem**.

Computational complexity - $O(M^3)$.

With dual formulation, the original optimization problem involved minimizing over $M$ variables into the dual problem with $N$ variables.

For fixed set of basis functions $M$ is smaller than the data points number $N$, dual representation seems useless. However, it allows the model to be reformulated using kernels, and the maximum margin classifier can be applied efficiently to feature spaces whose dimensionality $M$ exceeds the number of data points $N$, including infinite feature spaces.

The kernel formulation also makes clear the role of the constraint that the kernel function $k(x, x')$ be **positive definite** - this ensures the Lagrangian function $\tilde{J}(a)$ is bounded below, giving rise to a well-defined optimization problem.

So new data point can be evaluated by:

$$y(x) = w^T \phi(x) + b = \sum_{n=1}^{N} a_n t_n k(x, x_n) + b \tag{1.15}$$

For the Karush-Kuhn-Tucker KKT condtion, which in this case require that the following three properties hold:

$$a_n \geq 0 \tag{1.16}$$

$$t_n y(x_n) - 1 \geq 0 \tag{1.17}$$

$$a_n \{t_n y(x_n) - 1\} = 0 \tag{1.18}$$

---

check the Appendix E

---

For every data point, either $a_n = 0$ or $t_n y(x_n) = 1$, any data point for which $a_n = 0$ will not appear in the sum in1.15 thus play no role in making predictions for new data points. **The remaining data points are called support vectors** - and they satisfy $t_n y(x_n) = 1$, they lie on the maximum margin hyperplanes in feature space – once the model is trained only the support vectors retained.

Solving the quadratic programming problem for $a$, determine $b$ by

$$t_n \Big( \sum_{m \in \mathcal{S}} a_m t_m k(x_n, x_m) + b \Big) = 1 \tag{1.19}$$

where $\mathcal{S}$ denotes the set of indices of the support vectors.

A more stable solution for b is:

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{m \in \mathcal{S}} \Big( t_n - \sum_{m \in \mathcal{S}} a_m t_m k(x_n, x_m) \Big) \tag{1.20}$$

where $N_{\mathcal{S}}$ is the total number of support vectors.

### 1.1.1 Overlapping class distributions

### 1.1.2 Relation to logistic regression

### 1.1.3 Multiclass SVMs

### 1.1.4 SVMs for regression

### 1.1.5 Computational learning theory

## 1.2 Relevance Vector Machines

### 1.2.1 RVM for regression

### 1.2.2 Analysis of sparsity

### 1.2.3 RVM for classification

We now delve right into the proof.

**Lemma 1.1** *This is the first lemma of the lecture.*

**Proof:** The proof is by induction on .... For fun, we throw in a figure.



Figure 1.1: A Fun Figure

This is the end of the proof, which is marked with a little box. ■

### 1.2.4 A few items of note

Here is an itemized list:

- this is the first item;
- this is the second item.

Here is an enumerated list:

1. this is the first item;

2. this is the second item.

Here is an exercise:

**Exercise:** Show that P $\neq$ NP.

Here is how to define things in the proper mathematical style. Let $f_k$ be the $AND - OR$ function, defined by

$$f_k(x_1, x_2, \ldots, x_{2^k}) = \begin{cases} x_1 & \text{if } k = 0; \\ AND(f_{k-1}(x_1, \ldots, x_{2^{k-1}}), f_{k-1}(x_{2^{k-1}+1}, \ldots, x_{2^k})) & \text{if } k \text{ is even;} \\ OR(f_{k-1}(x_1, \ldots, x_{2^{k-1}}), f_{k-1}(x_{2^{k-1}+1}, \ldots, x_{2^k})) & \text{otherwise.} \end{cases}$$

**Theorem 1.2** *This is the first theorem.*

**Proof:** This is the proof of the first theorem. We show how to write pseudo-code now.

Consider a comparison between $x$ and $y$:

**if** $x$ or $y$ or both are in $S$ **then**
    answer accordingly
**else**
    Make the element with the larger score (say $x$) win the comparison
    **if** $F(x) + F(y) < \frac{n}{t-1}$ **then**
        $F(x) \leftarrow F(x) + F(y)$
        $F(y) \leftarrow 0$
    **else**
        $S \leftarrow S \cup \{x\}$
        $r \leftarrow r + 1$
    **endif**
**endif**

This concludes the proof. ■

## 1.3 Next topic

Here is a citation, just for fun [CW87].

## References

[CW87] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987, pp. 1–6.