Machine Learning homework 7 solution

Soft-margin SVM and Kernels

Wiktor Jurasz - M.Nr. 03709419

December 8, 2018

1 Problem 1

It is **not** guaranteed that all samples will have correct labels assigned. In Soft-margin SVM the objective function provides trade-off between margin size and classification accuracy. So it might happen that for linearly separable dataset with some parameter C misclassification will yield a better (smaller) value of objective function.

1.1 Example

Let's consider following one dimensional training dataset:

X	Y
-10	-1
0	-1
1	1

Objective function for soft-margin SVM:

$$f_0(w, b, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \tag{1}$$

To split correctly the hyperplane P should be equal to 0.5 and size of the margin $m = 1$. From equation $m = \frac{2}{||w||}$ we can calculate that $||w|| = 2 = w$. We also know that all ξ_i are equal to 0 since all points are correctly classified. Thus:

$$f_0(w, b, \xi) = \frac{1}{2} 2 * 2 = 2$$

Let's see now what happens when the decision boundary is moved to $P = -4.5$ taking -10 and 1 as support vectors and misclassifying $x = 0$
In this case $m = 11$ and $w = \frac{2}{11}$. Also ξ_1 and ξ_3 are equal to 0 (as those points are classified correctly). ξ_2 is misclassified and for optimal solution we have

$$\xi_2 = 1 - y_2(wx_2 + b) = 1 + b \tag{2}$$

We can also calculate b with $d = -\frac{b}{||w||}$ as we know distance from origin.

$$\begin{aligned} -4.5 &= -\frac{b}{\frac{2}{11}} \\ -b &= \frac{9}{2} * \frac{2}{11} \\ b &= \frac{9}{11} \end{aligned} \tag{3}$$

Plugging it all back to the objective function:

$$f_0(w, b, \xi) = \frac{1}{2} * (\frac{2}{11})^2 + C * \frac{20}{11} = \frac{2}{121} + C * \frac{20}{11} \tag{4}$$

Thus for all $C < \frac{12}{11}$ Soft-margin SVM will misclassify $x = 0$ regardless of the fact that the dataset is linearly separable.

2 Problem 2

For soft-margin SVM we have:

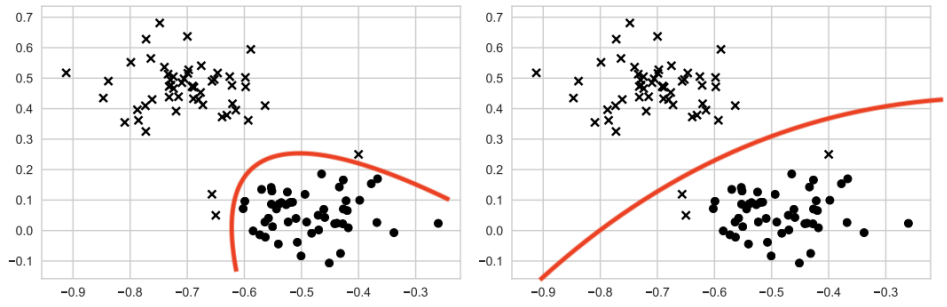
$$f_0(w, b, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \tag{5}$$

with condition

$$\xi_i \geq 0 \tag{6}$$

Thus for $C < 0$, $C \sum_{i=1}^{N} \xi_i < 0$. As this is minimization problem our solution will favour big values of ξ_i. The consequence of this is the bigger misclassification the "better" the result. Which contradicts the whole idea of using SVM as classifying method.

3 Problem 3



(a) $C = 10^{10}$ (b) $C = 10^{-10}$

For the graph (a) we have $C = 10^{10}$ which means the penalty for misclassification very high, thus SVM prefers smaller margin with more points correctly classified.

For the graph (b) $C = 10^{-10}$ which makes is more "profitable" to misclassify few points in order to gain greater margin.

4 Problem 4

To prove $k(x_1, x_2)$ is a valid kernel we can show that it can be derived from different valid kernel using kernel preserving operations.

Let's take $k_0(x_1, x_2) = x_1^T x_2 =< x_1, x_2 >$ this is obviously a valid kernel as it directly corresponds to the inner product in the original feature space.

- From multiplication operation we know that: $k'(x_1, x_2) = k_0(x_1, x_2)^i$ is a valid kernel $\forall i \in \mathbb{N}$
- From scala multiplication we know that $k''(x_1, x_2) = a_i k'(x_1, x_2)$ is a valid kernel

- From addition operation we know that $k'''(x_1, x_2) = \sum_i^N k''(x_1, x_2)$ is a valid kernel
- Finally we can also say that adding a positive constant to a valid kernel preserves validity. To see why Let's consider matrix K which is k''' kernel matrix. We know that it's PSD so the following is true: $\sum_{i,j} v_i v_j k_{ij} \geq 0$ now if we add positive constant a_0 we have $\sum_{i,j} v_i v_j (k_{ij} + a_0) = \sum_{i,j} v_i v_j k_{ij} + a_0 (\sum_i v_i)^2 \geq 0$.

Thus: $k'''(x_1, x_2) + a_0 = k(x_1, x_2)$ is a valid kernel

5 Problem 5

Since $x_1 x_2 < 0$ we can expand $k(x_1, x_2)$ into Taylor's series in the following way:

$$k(x_1, x_2) = \frac{1}{1 - x_1 x_2} = \sum_{i=0}^{\infty} (x_1 x_2)^i \tag{7}$$

Now to express this sum as inner product we have to use following mapping:

$$\phi : \mathbb{R} \to \mathbb{R}^{\infty} \tag{8}$$

$$\phi(x) = [x^0, x^1, x^2, ...] \tag{9}$$

Then:

$$k(x_1, x_2) = \phi(x)^T \phi(x) \tag{10}$$

6 Problem 6

6.1 Algorithm Description

Algorithm compares every possible pair of characters from strings x and y and counts how many of those pairs contain identical characters.

6.2 Kernel

Let's consider a function:

$$f(x, y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$$

Also lets transform the alphabet in a way that every character is assigned an unique natural number (i.e. first character is encoded as 1, second as 2 ... and last character is encoded by v). In this way we will work on vector of numbers rather then vector of characters.

Now we can write down Algorithm 1 in more compact form:

$$k(x, y) = \sum_{i=1}^{m} \sum_{j=1}^{n} f(x_i, y_j) \tag{11}$$

As sum of valid kernels is also a valid kernel in order to show validity of k it's enough to prove that f is valid.

To do this let's consider a mapping:

$$\phi : \mathbb{R} -> \mathbb{R}^v \tag{12}$$

$$\phi(x) = [max(0, (-|x - 1| + 1)), max(0, (-|x - 2| + 1)), ..., max(0, (-|x - v| + 1))] \tag{13}$$

So for $x \in \mathbb{N}$, $\phi(x)$ is a v dimensional vector with all but one elements equal 0. The only non zero element is equal to 1 and is on the $x - th$ place. For example with $v = 5$, $\phi(3) = [0, 0, 1, 0, 0]$.

Now we can see that $k'(x, y) =< \phi(x), \phi(y) >= 1 \Leftrightarrow x = y$.

As k' is an inner product and also behaves identically as f, thus we can write:

$$k(x, y) = \sum_{i=1}^{m} \sum_{j=1}^{n} k'(x, y) = \sum_{i=1}^{m} \sum_{j=1}^{n} \phi(x)^T \phi(x) \tag{14}$$

Which shows that k is a valid kernel.

7 Problem 7

Short answer: **yes**. If we look closely at Gaussian kernel, we will see that it depends on the distance between two data points. By σ we can influence how "close" two points are to each other. With small sigma even if $|x_1 - x_2|^2$ is very small, the overall value of transformation is large, thus we can very precisely classify all the points, as all of the seems to be very far from each other. Of course this may lead to heavy overfitting. The only case when two points cannot be separated using this Kernel is when they have the same coordinates (and yet belong to different classes).