Machine Learning Homework Sheet 07

Soft-margin SVM and Kernels

1 Soft-margin SVM

Problem 1: Assume that we have a linearly separable dataset $\mathcal{D}$, on which a soft-margin SVM is fitted. Is it guaranteed that all training samples in $\mathcal{D}$ will be assigned the correct label by the fitted model? Explain your answer.

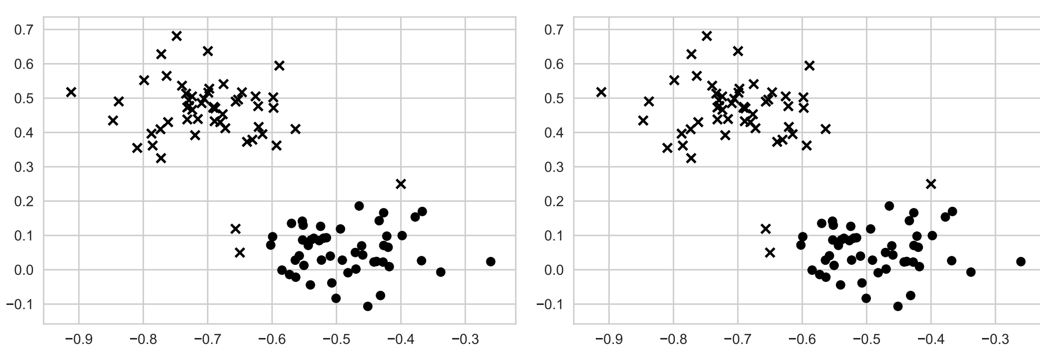
No, as it might happen that misclassifying a few points that are very close to the decision boundary would lead to a significantly increasing the margin. In general, larger values of C make this behavior less likely.

Problem 2: Why do we need to ensure that $C > 0$ in the slack variable formulation of soft-margin SVM? What would happen if this was not the case?

If $C = 0$, all constraints would be trivially fulfilled by setting $\xi_i = 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)$ and letting $||\boldsymbol{w}||$ be arbitrarily large.

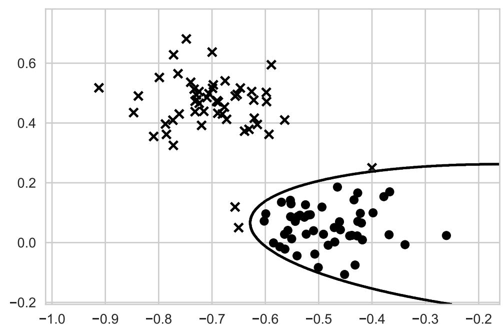
For $C < 0$ the resulting objective function would be not bounded below, so the optimal solution would be to let $\xi_i \to \infty$ for all i.

Problem 3: Sketch the decision boundary of an SVM with a quadratic kernel (polynomial with degree 2) for the data in the figure below, for two specified values of the penalty parameter C. (The two classes are denoted as •'s and ×'s.)

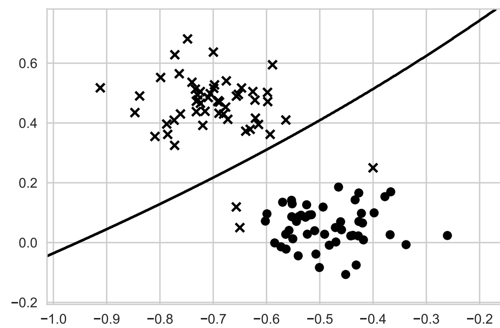


(a) $C = 10^{10}$ (b) $C = 10^{-10}$

Explain the reasoning behind your sketch of the decision boundary for both cases (one sentence for each plot).



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



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

a) With such a large penalty SVM will try to correctly classify **all** of the instances in the training set.
b) Given the small penalty, we can allow few misclassified instances, and obtain a larger margin between the two classes. The decision boundary looks linear.

2 Kernels

Problem 4: Show that for $N \in \mathbb{N}$ and $a_i \geq 0$, with $i \in [0, N]$ the function

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{i=1}^{N} a_i \left(\boldsymbol{x}_1^T \boldsymbol{x}_2\right)^i + a_0$$

is a valid kernel.

The term $\boldsymbol{x}_1^T \boldsymbol{x}_2$ is a kernel because it is the scalar product of the input vectors. The constant $a_0 \geq 0$ is a kernel because we can define the feature map $\phi(\boldsymbol{x}) = \sqrt{a_0}$ and obtain this kernel by calculating the scalar product in feature space $\phi(\boldsymbol{x}_1)^T \phi(\boldsymbol{x}_2) = \sqrt{a_0}^2 = a_0$.

k is a kernel since it is built up of sums and products of kernels and multiplication with non-negative scalars.

Problem 5: Find the feature transformation $\boldsymbol{\phi}(x)$ corresponding to the kernel

$$k(x_1, x_2) = \frac{1}{1 - x_1 x_2},$$

with $x_1, x_2 \in (0, 1)$.

Hint: Consider an infinite-dimensional feature space.

We use the geometric series to transform k:

$$k(x_1, x_2) = \frac{1}{1 - x_1 x_2} = \sum_{i=0}^{\infty} x_1^i x_2^i = \boldsymbol{\phi}(x_1)^T \boldsymbol{\phi}(x_2),$$

with the feature transformation

$$\boldsymbol{\phi}(x) = \left(1, x, x^2, x^3, x^4, \dots\right)^T$$

Problem 6: Consider the following algorithm.

Algorithm 1: Counting something

input : Character string x of length m (one based indexing)
input : Character string y of length n (one based indexing)
output: A number $s \in \mathbb{R}$

$s \leftarrow 0$;
for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow 1$ **to** n **do**
    if $x[i] == y[j]$ **then**
      $s \leftarrow s + 1$;

a) Explain, in no more than two sentences, what the above algorithm is doing.

The algorithm sums how many times each character c from string x appears in string y (or vice versa).

b) Let $\mathcal{S}$ denote the set of strings over a finite alphabet of size v. Define a function $k : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ as the output of running algorithm 1 on a pair of strings x, y. Show that $k(x, y)$ is a valid kernel.

Algorithm 1 can be rewritten as follows

$s \leftarrow 0$;
for $i \leftarrow 1$ **to** m **do**
  $s \leftarrow s +$ (number of occurrences of $x[i]$ in y);

and furthermore

$s \leftarrow 0$;
for $c \leftarrow 1$ **to** v **do**
  $s \leftarrow s +$ (number of occurrences of c in x) $\times$ (number of occurrences of c in y);

Thus, defining the feature map $\boldsymbol{\phi} : \mathcal{S} \to \mathbb{R}^v$, where $\boldsymbol{\phi}(x)_c$ is the number of occurrences of character c in string x, we see that the above algorithm computes $k(x, y) = \boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(y)$. Since we constructed an explicit feature map $\boldsymbol{\phi}$, k is a valid kernel.

3 Gaussian kernel

Problem 7: Can any *finite* set of points be linearly separated in the feature space of the Gaussian kernel

$$k_{\mathrm{G}}(x_1, x_2) = \exp\left(-\frac{|x_1 - x_2|^2}{2\sigma^2}\right),$$

if σ can be chosen freely?

Consider the limit $\sigma \to 0$. The kernel function becomes

$$k(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 = x_2 \\ 0 & \text{if } x_1 \neq x_2 \end{cases}.$$

Thus the kernel matrix for any finite set of points is the identity matrix, $k = I$.

All training samples are correctly classified if

$$y_i(\boldsymbol{w}^T \boldsymbol{\phi}_{\mathrm{G}}(x_i) + b) > 0 \quad \text{for all } i\,.$$

By substituting the dual representation $\boldsymbol{w} = \sum_j y_j \alpha_j \boldsymbol{\phi}_{\mathrm{G}}(x_j)$ into this expression and replacing the scalar product $\boldsymbol{\phi}_{\mathrm{G}}(x_i)^T \boldsymbol{\phi}_{\mathrm{G}}(x_j)$ by the kernel function $k(x_i, x_j)$ this condition translates to

$$y_i \left(\sum_j y_j \alpha_j k(x_i, x_j) + b \right) > 0\,.$$

With the above kernel function in particular the condition becomes

$${y_i}^2 \alpha_i + y_i b > 0\,.$$

By choosing $b = 0$ we see that the resulting condition is fulfilled for all training samples, since $\forall i \;\; y_i^2 = 1$, and we can simply set all $\alpha_i > 0$. (Note, that this means that every sample is a support vector in this case).

Hence all finite sets of points can be linearly separated using the Gaussian kernel if the variance σ is chosen small enough.