Machine Learning - Assignment 2

Brian Pulfer

October 2019

1 Kernels I

1.1

$K(x, y) = x^T y + (x^T y)^2$ is a valid kernel. Here's the proof:

1.1.1 First property

The first property that can be used to prove that the given kernel is a valid one is the following:

$$K(x, x') = q(K_1(x, x'))$$

This property states that K(x,x') is a valid kernel given that K_1 is a valid kernel and that q() is a polynomial with non-negative coefficients (like in the given case). We must now thus just prove that $x^T y$ is a valid kernel.

1.1.2 Second property

The **linear kernel** is defined as

$$K(x, x') = x^T x'$$

and since this kernel is a valid one, thus also $x^T y$ is a valid kernel (since it's the same kernel).

1.2

$K(x, y) = x^2 e^{-y}, d = 1$ is a valid kernel. Here's the proof:
The **kernel matrix** for d=1 of this kernel is the following

$$K = \left[x_1^2 e^{-y_1}\right]$$

which is, as a matter of fact, a positive definite matrix. In fact, for any value of **t**, the formula

$$t * x^2 e^{-y} * t = t^2 * x^2 e^{-y} \geq 0$$

is always true for t > 0 and any value of x and y.

1.3

$K(x, y) = ck_1(x, y) + k_2(x, y)$ where $k_1(x, y), k_2(x, y)$ are valid kernels in $\mathbb{R}^d$ is a valid kernel if $c > 0$. Here's the proof:

1.3.1 First property

$$K(x, x') = cK_1(x, x'), c > 0$$

This property states that K(x,x') is a valid kernel given that K_1 is a valid kernel and that the c constant value is bigger than zero.

1.3.2 Second property

$$K(x, x') = K_1(x, x') + K_2(x, x')$$

This property states that K(x,x') is a valid kernel given that both K_1 and K_2 are valid kernels.

Combining the two properties, we have that the given kernel

$$K(x, y) = ck_1(x, y) + k_2(x, y)$$

is a valid one for any $c > 0$.

2 Kernels II

2.1 Dataset a

The kernel trick can be applied in this dataset. The kernel that could be used is the **RBF kernel** ($K(x, x') = e^{-\gamma(x-x')^2}$). **RBF** kernels are able to apply infinite dimension mappings, and thus also to handle hard cases like this one (XOR-like).

2.2 Dataset b

The kernel trick can be applied in this dataset as well, and both polynomial and RBF kernels are actually suitable. The kernel I have chosen for this particular case is the **polynomial kernel** because it can efficiently separate linearly the data points and is not as computationally expensive as the RBF kernel.

2.3 Dataset c

The linear kernel can be applied on this dataset, since the data points are already linearly separable.

2.4 Dataset d

Applying the kernel trick does not seem to be a good approach, since no good enough kernel can be found for this dataset. Another transformation that we could do is **embedding** the data (a method used to represent discrete variable as continuous vectors).

For example, we can create a function of the 'radius' **f(r)** (r = $\sqrt{x_1^2 + x_2^2}$) and model it so that it acts as we want for the various values of r. Here's a table of the desired values of such function depending on the radius.

r	f(r)
0.25	0
0.5	0.5
0.75	1
1	0.5
1.25	0
1.5	0.5
1.75	1

Table 1: Values of the radius and relative desired f(r)

A polynomial approximating this function can be found applying interpolation, for example.

3 SVMs I

3.1

Yes. The decision boundary of an SVM with a linear kernel is always one dimension lower with respect to the data points.

3.2

No. Depending on the value of C, the optimization problem changes and thus also the optimal solution. A higher value of C would mean that less errors have to be made and that the margin width is not as important and vice-versa.

3.3

The dimension of the decision boundary is $\mathbb{R}^2$.

3.4

No. The computational effort does not increase because only the dot product of those mappings is computed (through simpler formulas) and the coordinates are not actually calculated.

3.5

The gaussian kernel reaches its maximum value when the term $\frac{(x-x')}{(2\sigma^2)}$ is close to zero. In that case, the gaussian kernel is

$$exp\{-||\frac{(x-x')}{(2\sigma^2)}||\} = exp\{0\} = 1$$

3.6

A feature expansion won't get rid of the noise, which will be propagated to further dimensions. The best would be to reduce the noise in the data by changing the way data is measured.

4 SVMs II

4.1

The class of point x_1 can be found calculating:

$$sign(w * x_1^T + b) = sign(\begin{bmatrix}3 & 2\end{bmatrix} * \begin{bmatrix}3\\-1\end{bmatrix} + 2) = sign(9) = 1$$

So point x_1 belongs to the positive class.

4.2

We might need to retrain the SVM if the new data point ends up between the positive and the negative hyperplanes. In other words, we need to check if -1 $< w * x_2^T + b < 1$.

$$w * x_2^T + b = \begin{bmatrix}3 & 2\end{bmatrix} * \begin{bmatrix}-2\\2\end{bmatrix} + 2 = 0$$

Since the new datapoint is exactly between the positive and negative hyperplanes, the SVM needs to be retrained.

4.3

No. The SVM is not retrained when the new datapoints are correctly classified and are not placed between the hyperplanes. Only if a new datapoint is miss-classified or lies within the classification boundary the solution to the optimization problem (minimizing margin width and reducing missclassifications) might change and thus also the SVM's boundary.