Università della Svizzera italiana | **Institute of Computational Science ICS**

Machine Learning **2019**

Student: Sanchit Chiplunkar

Solution for Assignment 2

Due date: 1 November 2019, 21:00

Answer 1

Answer 1.1 Valid Kernel
Reason As we know that K$(x_1, x_2) = k_1(x_1, x_2) + k_2(x_1, x_2)$ is a valid kernel if k$_1$ and k$_2$ are valid kernels. Therefore the first part of this equation is the linear kernel while the second part is just a form of polynomial kernel.Therefore K(x,y) is a valid kernel.

Answer 1.2 Not a Valid kernel.As the kernel isn't symmetric.
K(x,y) $= x^2 \times \exp(-y)$. As we know that a kernel should satisfy the property K$(x_1, x_2) = K(x_2, x_1)$ which this given kernel doesn't satisfy. As can be seen below.

K(y,x) $= y^2 \times \exp(-x)$. Hence K(x,y) $\neq$ K(y,x). We can also see from a simple example. If x=0 and y=1. K(0,1) = 0 while K(1,0) = 1. Thus this isn't a valid kernel

Answer 1.3 Can't be said as it depends on the value of c in this case. When c≥0 we will have a valid kernel as c*k1 will be a valid kernel and k2 is a valid kernel Therefore there summation would be a valid kernel.On the other hand when c<0 we might get an invalid kernel or there is a possibility that we may also get a valid kernel.

Answer 2

Answer 2.1 Clearly data isn't linearly separable in the input space. Even if we use a linear kernel with a soft margin we might get bad results or high misclassification error because of the way data is distributed. A polynomial kernel with a soft margin should give us a decent result in this case while a gaussian kernel as we know can fit most of the train data but will have problem during the test set.

Answer 2.2 Clearly we can see from visual inspection that data is not linearly separable. So in this case a polynomial kernel should do the trick as we would like 2 transform our data to a high dimensional space to make it linearly separable. Note that a gaussian kernel would work too but it is more prone to overfitting relative to polynomial kernel.
As told on the iCorsi discussion forum a simple kernel should be preferred to a higher(in this case infinite dimensional) Gaussian kernel.

Answer 2.3 In this problem a linear SVM with a Soft margin would be a suitable choice. We can even use a polynomial kernel or a gaussian but they might be more prone to overfitting and not generalise well in the test set. Also Note we can also use a polynomial kernel with a soft margin in this case but as discussed in class a lower order would be the preferred answer.

Answer 2.4 As we can see from the plot that the data isn't linearly separable in 2-Dimension. Using a Gaussian kernel in this case would work well as it is capable of fitting most of the train data set with very low accuracy. Although there is a chance that our model doesn't generalise well to test set

Answer 3

Answer 3.1 Yes the decision boundary of an SVM in 2 dimension is a straight line. As we know that (w) in the SVM is of the same the dimension as the input vector i.e (x) As x $\in R^2$ therefore w $\in R^2$ and so we will have a straight line as our decision boundary in this case

Answer 3.2 No
Short answer is no we will get different w as we change our C but it would also depends on the data we are training our SVM on.
C i.e the penalty gives us a trade-off between having a higher margin of our hyperplane versus our misclassification rate. Even if the data is linearly separable, the model for a very low value of C will try to maximise the margin which might increase our misclassification(The model doesn't know if the data is linearly separable). While when the value of C is high the model will be extremely strict(hard margin) and might compromise on margin width to keep the misclassification rate low(0 in case of linearly separable data). Thus different value of C will give us different value of parameters W.
Also there are definitely cases on certain datasets where changing C might have no effect on our final weight. For example let's say we have 2 data points(1 + and 1-) and both of them are also the support vectors which gives the maximum margin possible. Now changing C won't have much effect in this case and we will get the same parameters (support vectors are responsible for computation of w).

Answer 3.3 As we know from answer 3.1 dimension of w is similar to that of x. So w $\in R^3$ in this case and we get a hyperplane as a decision boundary.

Answer 3.4 When we are using feature expansion what we mainly want to do is go from a current dimension U to another higher dimension V where the data is linearly separable and then compute the dot product between different data points in this higher dimension V to get a scalar value. Using the kernel function saves us the trouble of computing the values of our vectors in the V space/dimension and then followed by doing dot product of our vectors in the V space.Using a kernel function we can directly compute the scalar value which we would have gotten by doing all these steps and so in short our overall computational effort doesn't increase. This is also the reason why using a gaussian kernel we can compute infinite dimensional space which is impossible to compute in reality using the traditional methods.
Also on a side note if we increase the dimension of our input vector x $\in R^d$ all we have to do is more multiplication and summation when we do dot product between two vectors which again isn't that expensive relative to the other computations we are doing.

Answer 3.5 The maximum value of a Gaussian kernel is 1.
The Gaussian kernel is as follows
$\exp(-\frac{(x_1-x_2)^2}{2\sigma^2})$ Let's say $\alpha = \frac{(x_1-x_2)^2}{2\sigma^2}$. Therefore α is always positive because both the numerator and denominator are squares.Thus $\exp(-\alpha)$ will always be in the range from $[0, 1]$. For high value of α, $\exp(-\alpha)$ will be approaching to 0 while when $x_1 = x_2$ alpha =0 thus $\exp(-\alpha) = 1$. Thus the maximum value of the gaussian kernel will be 1.

Answer 3.6 In the end it depends on our data. Feature expansion might definitely help us in giving lower training error but there is no guarantee that it will generalise better with the test set. It is similar to trade off between model complexity and training error. Higher model complexity for example Gaussian kernels fit the training data extremely well but are more prone to overfitting and might give higher test error. While a model with low model complexity for example Linear SVM with soft margin might have higher training error relative to Gaussian kernels but may perform well on the test set relative to more complex model so False.

Answer 4

Answer 4.1
w $\times x_k$ +b for data point $x_k = [3, -1]$ with w $= [3, 2]$ and b=2 gives us: 3 $\times$3 + 2$\times$ -1 + 2 = 9 which is positive therefore label of this data point is positive.

Answer 4.2
For the new sample x_2 = [-2,2]. Using Equation from Answer 4.1 we will get -2$\times$3 + 2$\times$2 +2 = 0. Thus our new sample lies inside our margin more specifically exactly on the hyperplane.Therefore we would need to retrain our SVM in this case to learn the new support vectors which maximise our margin.

Answer 4.3
It will depend on the kind of training data we get i.e if: 1.) If our train data lies inside the margin or on the wrong side of margin then we need to retrain our model this is true for Soft as well as Hard margin SVM(in hard SVM we don't allow our data to be in the margin or on the wrong side of it).In our dual problem all the data points which are support vectors have a positive Lagrange multipliers and so thus only these are responsible for computation of our hyperplane and margin as our overall dual formulation will change and so will our primal solution and thus w.
2.) If our data lies on the correct margin i.e it is a support vector or lies on the correct side of margin(i.e aren't support vectors) we don't need to retrain our model. Even if we retrain our model in this case our Lagrange multiplier α_i will change in our dual solution for support vectors but the overall value our primal solution will still be the same. Also as data which aren't support vectors have a α_i =0 so they aren't responsible for changing our primal solution or computation of w.