Machine Learning Homework #2

Joe Ellis - jge2105

October 14, 2013

1 Problem 1

In this problem we take a look at VC-dimensions, and how they operate and work.

First, we will look at this problem, where we must prove that $VC(H) \leq log_2|H|$, where H is a hypothesis space, and $|H|$ are the amount of possible hypothesis available in the space. First, we assume that $VC(H) = d$. Thus, whatever our shattering function is (lines, hyperbolas, circles, etc.) it can shatter d-points in the hypothesis space. This is by the defninition of VC-dimension. If we are seperating the points into two different classes, and we have d-points, then we know that there are atleast 2^d different hypotheses available to us. Thus, for our space H, we know that $|H| \geq 2^d$. However, since $|H| \geq 2^d$, then we also have that,

$$\begin{aligned} log_2|H| &\geq log_2(2^d) && (1)\\ &= d && (2)\\ &= VC(H). && (3) \end{aligned}$$

Thus, we have shown that $VC(H) \leq log_2|H|$, as desired.

In this problem we address a linear perceptron in $\mathbb{R}^d$, and find its VC-dimension. We want to find the number of points that a perceptron can shatter in this dimension, so let's analyze. We know that a linear perceptron in $\mathbb{R}^2$, can shatter 3 points. However, we know from slide #7 in Topic #5 slides that for any high-dimensional linear classifier in $\mathbb{R}^d$ that we can shat any set of linearly independent points. For a d-dimensional space, we can have d+1 linearly independent points. Therefore, for any linear classifier in high-dimensional space the VC-dimension is $d+1$, where d is the dimensionality of the

space. Since a linear perceptron is in fact a linear classifier, then this theorem applies. Therefore, for any linear perceptron, θ in $\mathbb{R}^d$ space, we have $VC(\theta) = d + 1$.

We want to find the VC-dimension of a class of circles on points in the space $\mathbb{R}^2$. Assume we have 2 points, to shatter the points then we simply place our circle on one of the points small enough so the other point does not lie within it, and they are shattered. Now, let's assume that we have 3 points in $\mathbb{R}^2$, where none of the points are equal to one another. Therefore, we will have two points of either positive or negative class, and one of the other classes. We can shatter these points by choosing a circle of radius d around the point with only one class called x_1, s.t. $d < \underset{x}{\text{argmin}}(dist(x_1, x_2), (dist(x_1, x_3))$. Thus, we have now shattered three points. Finally, let's attempt to shatter 4 points. We place 4 points in the space, and then the opponent labels the points, but he/she labels the points such that there are 2 positive, and 2 negative points. Let's call these points, $(x_{+1}, x_{+2}, x_{-1}, x_{-2})$. If we choose three points to be in a line equi-distant from each other as seen below with one equidistant but directly above the top point, we can shatter any combination of these points with a circle. This is shown in Figure 1. Therefore, we have shattered 4 points. However, 5 points in any formation are unshatterable by a circle in $\mathbb{R}^2$, therefore we have $VC(circles) = 4$. For the below example of 5 points, an unshatterable labeling would be to label all points in a line one label, and then the outer points would have the opposite label.

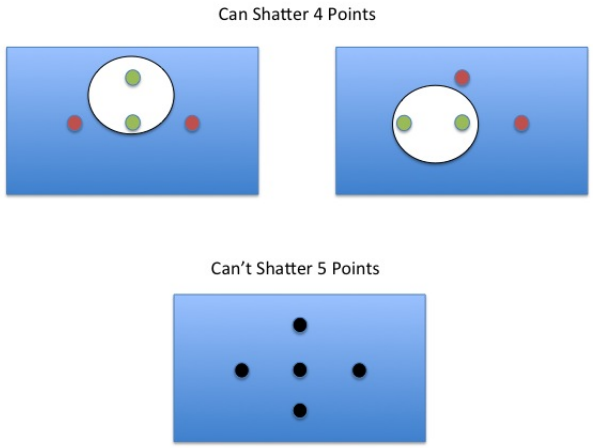

Figure 1: Shattering 4 points with a circle

2 Problem 2

A stationary kernel is some kernel such that the kernel result is translation invariant, we denote this as $K(x,z) = K_s(x-z)$. We have two examples of stationary kernels for this problem. The first example is the exponential kernel, $K_e(x,z) = e^{(-\frac{||x-z||}{\theta})}$, and the second example is the gaussian kernel, which is $K_g(x,z) = e^{(-\frac{||x-z||^2}{\theta})}$.

We will show that they are mercer kernels based on the definition given in the lecture slides, that a mercer kernel is a kernel which is kernel that takes two inputs and outputs a scalar via denoted as,

$$k(x_1,x_2) =< \phi(x_i), \phi(x_2) >= \begin{cases} \phi(x_i)^t\phi(x_2) & \text{for } finite\phi \\ \int_t \phi(x_i,t)\phi(x_2,t) & \text{otherwise} \end{cases} \tag{4}$$

Both of these are mercer kernels, because they both output scalar values, for both conditions. Let's assume that ϕ is finite, and then we know that on the top of the exponential function we have a magnitude which is scalar divided by a scalar parameter θ which yields some scalar output value. We also, know that the integral of the kernel provided is bounded due to the nature of the gaussian and exponential kernel, and therefore $\int_t \phi(x_i,t)\phi(x_2,t)$, is also bounded. It is shown in second section of this problem also that the gaussian kernel without sigma can be broken down into the inner product of two functions ϕ. The break down of both the gaussian and exponential kernels can be found in the same way, and therefore they satisfy the fact that they are kernels. Therefore, both are mercer kernels.

Now let's show that both of these are stationary kernels if they are summed or multiplied together. First, assume we sum the kernels together,

$$\begin{aligned} K_{sum}(x,z) &= K_e(x,z) + K_g(x,z) && (5)\\ &= e^{(-\frac{||x-z||}{\theta})} + e^{(-\frac{||x-z||^2}{\theta})}, && (6)\\ &= K_{se}(x-z) + K_{sg}(x-z). && (7)\\ & && (8) \end{aligned}$$

Since both values that we receive in the summation depend only on the lag vectors, then we find that they are stationary under summation.

Now let's show that these are also stationary kernels under multiplication. For multiplicaiton we have,

$$\begin{aligned} K_{mult}(x,z) &= K_e(x,z)K_g(x,z) && (9)\\ &= e^{(-\frac{||x-z||}{\theta})}e^{(-\frac{||x-z||^2}{\theta})}, && (10)\\ &= e^{(-\frac{||x-z||+||x-z||^2}{\theta})} && (11)\\ & && (12) \end{aligned}$$

Since the exponent of the above multiplication of the kernels also only depends on the value $x - z$, we have shown that these are also stationary under multiplication.

For the second part of this problem, we will explicitly find the ϕ function such that $K(x_1, x_2) = \phi(x_1)^t \phi(x_2) = e^{(-|||x_1 - x_2||^2)}$. We will use the taylor series expansion equality as follows in the proof below, $e^{2x_1x_2} = \sum_{k=0}^{\infty} \frac{2^k x_1^k x_2^k}{k!}$.

$$
\begin{aligned}
K(x_1, x_2) &= e^{-||x_1 - x_2||^2} && (13)\\
&= e^{-(x_1 - x_2)(x_1 - x_2)} && (14)\\
&= e^{-(x_1^2 + x_2^2 - 2x_1x_2)} && (15)\\
&= e^{-x_1^2} e^{-x_2^2} e^{2x_1x_2} && (16)\\
&= e^{-x_1^2} e^{-x_2^2} \sum_{k=0}^{\infty} \frac{2^k x_1^k x_2^k}{k!} && (17)\\
&= e^{-x_1^2} e^{-x_2^2} \sum_{k=0}^{\infty} \frac{2^k}{k!} x_1^k x_2^k && (18)\\
&= \phi(x_1)^t \phi(x_2) && (19)
\end{aligned}
$$

Thus, $\phi_k(x) = e^{-x^2} \frac{2^{k/2}}{\sqrt{k!}} x^k$, where $\phi(x) = (\phi_1(x), \phi_2(x), ..., \phi_\infty(x))$.

3 Problem 3

For this problem we calculated the gram matrix for a given kernel $f(x, \lambda, k)$. We define $f(x, \lambda, k)$ as 0 for $x < 0$, and $f(x, \lambda, k) = (\frac{k}{\lambda})(\frac{x}{\lambda})^{k-1}(e^{-(\frac{x}{\lambda})^k})$. We define our kernel function $K(x_i, x_j) = F(x_i^t x_j, \lambda = 0.75, k = 3)$, where $F(x, \lambda, k) = \int_0^x f(x, \lambda, k)\, dx$.

When we integrate $(f(x, \lambda, k)$, we get for the indefinite integral,

$$
\int f(x, \lambda, k)\, dx = F(x, \lambda, k) = -e^{(\frac{x}{\lambda})^k}. \tag{20}
$$

Therefore, using this knowledge and computing the definite integral between 0 and x, we created a Matlab script to calculate the desired gram matrix, which is submitted and called "ComputeProb3.m". The gram matrix can be seen here,

$$
\begin{pmatrix}
0.0698 & 0.0456 & 0.0104 & 0.0072 & 0.0635 \\
0.0456 & 0.0420 & 0.0043 & 0.0061 & 0.0191 \\
0.0104 & 0.0043 & 0.0053 & 0.0007 & 0.0080 \\
0.0072 & 0.0061 & 0.0007 & 0.0009 & 0.0039 \\
0.0635 & 0.0191 & 0.0080 & 0.0039 & 0.2478
\end{pmatrix}
$$

This is the gram matrix for the points given in problem 3.

4 Problem 4

Here we create an object recognition system using SVM for classification. Multiple SVM kernels were chosen and evaluated, and all of the kernels chosen were able to perform well with the proper parameters. We performed cross-validation across the data-set using a 50-50 train and test split for the sample data points. We performed cross-validation with 10 random splits for each parameter to show the performance of the SVM with each different kernel. Given the proper parameters we were able to get 100% accuracy using each of the described kernels, and this is because the data provided to us was seperable. Figure 5 shows the positive and negative samples from our dataset plotted with respect to their first two dimensions. We can see that they are seperable through a linear classifier.

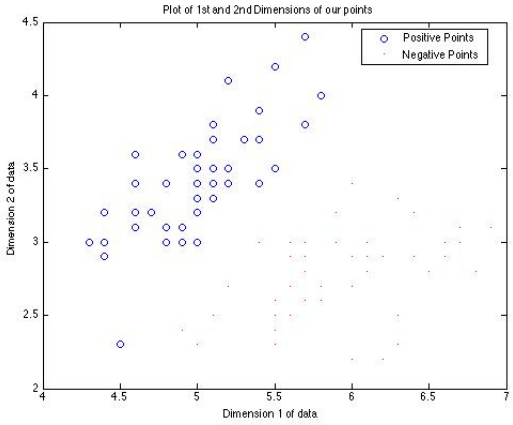

Figure 2: Plot of the 1st and 2nd dimensions of the dataset

For classification we used linear, RBF, and polynomial SVM with different values for the cost variable (C), and the number of degrees for the polynomial function and the size of sigma in the RBF. Each of these paramters effected the classification accruacy, but proper parameters for each kernel scheme could be found. The results of a different parameters can be seen in the upcoming figures, where accuracy is on the z-axis, and the x and y axis are the independent variables.

As you can see, we have found parameters for each type of kernel that achieves maximum accuracy. However, the complexity of each of the kernels was not all the same. To do a quick analysis we used some of the kernel adn parameters that had the best possible classification accuracy on our dataset, and then analyzed how many support vectors each learned classifier used. The results can be seen below in Table 1.

The kernel that in general used the least amount of support vectors, and was quickest in classification was the linear SVM. Therefore, for this particular classification task I would reccomend using the linear SVM with $C = 10$. However, if the dataset is more difficult and not linearly seperable we would instead need to focus on using a more flexible kernel function such as the RBF.

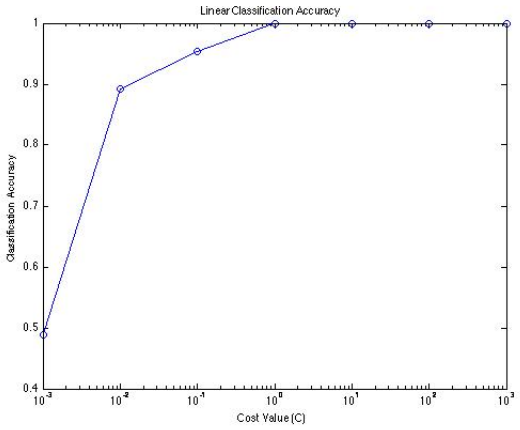

Figure 3: Linear SVM classification accuracy for different parameters

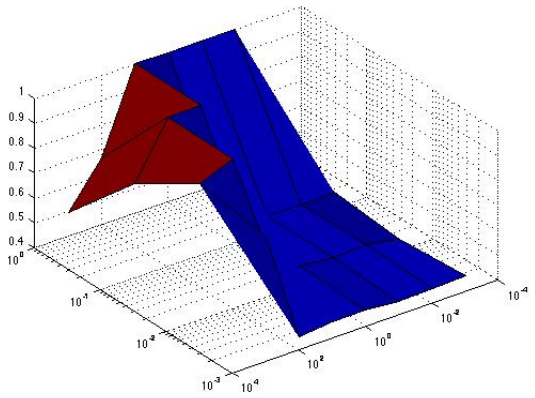

Figure 4: RBF SVM classification accuracy for different parameters

Table 1: Average number of support vectors for 10 cross-validation trials

Classifier Type	Cost-Variable	# of support vectors
linear	10	2.9
2-d polynomial	.001	20.5
2-d polynomial	10	6.7
RBF sigma=1	1	11.3
RBF sigma=1	.01	50

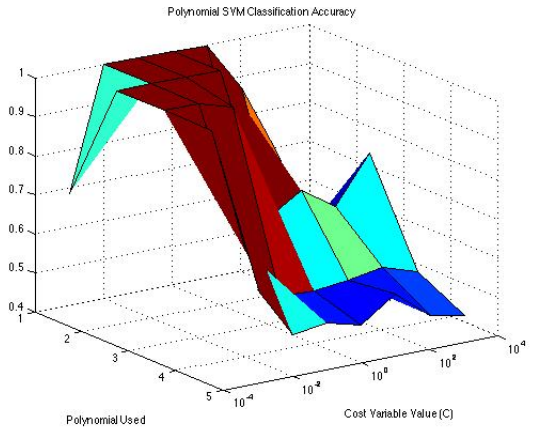

Figure 5: Polynomial SVM classification accuracy for different parameters