CSL 865: Assignment 2

Due Date: Tuesday October 16, 2012 (in class)

Note: Refer to the course website for assignment submission instructions.

1. **[16 points]Constructing kernels**

   In class, we saw that by choosing a kernel $K(x, z) = \phi(x)^T\phi(z)$, we can implicitly map data to a high dimensional space, and have the SVM algorithm work in that space. One way to generate kernels is to explicitly define the mapping ϕ to a higher dimensional space, and then work out the corresponding K.

   However in this question we are interested in direct construction of kernels. I.e., suppose we have a function $K(x, z)$ that we think gives an appropriate similarity measure for our learning problem, and we are considering plugging K into the SVM as the kernel function. However for $K(x, z)$ to be a valid kernel, it must correspond to an inner product in some higher dimensional space resulting from some feature mapping ϕ. Mercer's theorem tells us that $K(x, z)$ is a (Mercer) kernel if and only if for any finite set $\{x^{(1)}, \cdots, x^{(m)}\}$, the matrix K is symmetric and positive semidefinite, where the square matrix $K \in R^{m\times m}$ is given by $K_{ij} = K(x^{(i)}, x^{(j)})$.

   Now here comes the question: Let K_1 , K_2 be kernels over $R^n \times R^n$, let $a \in R^+$ be a positive real number, let $f : R^n \longmapsto R$ be a real-valued function, let $\phi : R^n \longmapsto R^d$ be a function mapping from R^n to R^d , let K_3 be a kernel over $R^d \times R^d$, and let $p(x)$ a polynomial over x with positive coefficients. For each of the functions K below, state whether it is necessarily a kernel. If you think it is, prove it; if you think it isn't, give a counter-example.

   (a) $K(x, z) = K_1(x, z) + K_2(x, z)$
   (b) $K(x, z) = K_1(x, z) - -K_2(x, z)$
   (c) $K(x, z) = aK_1(x, z)$
   (d) $K(x, z) = -aK_1(x, z)$
   (e) $K(x, z) = K_1(x, z)K_2(x, z)$
   (f) $K(x, z) = f(x)f(z)$
   (g) $K(x, z) = K_3(\phi(x), \phi(z))$
   (h) $K(x, z) = p(K_1(x, z))$

2. **[15 points]Kernelizing the Perceptron**

   Let there be a binary classification problem with $y \in \{0, 1\}$. The perceptron uses hypotheses of the form $h_\theta(x) = g(\theta^T x)$, where $g(z) = \mathbf{1}\{z \geq 0\}$. In this problem, we will consider a stochastic gradient descent-like implementation of the perceptron algorithm where each update to the parameters θ is made using only one training example. However, unlike stochastic gradient descent, the perceptron algorithm will only make one pass through the entire training set. The update rule for this version of the perceptron algorithm is given by

   $$\theta^{(i+1)} := \theta^{(i)} + \alpha[y^{(i+1)} - h_{\theta^{(i)}}(x^{(i+1)})]x^{(i+1)}$$

   where $\theta^{(i)}$ is the value of the parameters after the algorithm has seen the first i training examples. Prior to seeing any training examples, $\theta^{(0)}$ is initialized to $\vec{0}$.

   Let K be a Mercer kernel corresponding to some very high-dimensional feature mapping ϕ. Suppose ϕ is so high-dimensional (say, ∞-dimensional) that it's infeasible to ever represent $\phi(x)$ explicitly. Describe how you would apply the "kernel trick" to the perceptron to make it work in the high-dimensional feature space ϕ, but without ever explicitly computing $\phi(x)$. [Note: You don't have to worry about the intercept term. If you like, think of ϕ as having the property that $\phi_0(x) = 1$ so that this is taken care of.] Your description should specify

(a) (5 points) How you will (implicitly) represent the high-dimensional parameter vector $\theta^{(i)}$, including how the initial value $\theta^{(0)} = \vec{0}$ is represented (note that $\theta^{(i)}$ is now a vector whose dimension is the same as the feature vectors $\phi(x)$);

(b) (5 points) How you will efficiently make a prediction on a new input $x^{(i+1)}$. I.e., how you will compute $h_{\theta^{(i)}}(x^{(i+1)}) = g(\theta^{(i)^T}\phi(x^{(i+1)}))$, using your representation of $\theta^{(i)}$; and

(c) (5 points) How you will modify the update rule given above to perform an update to θ on a new training example $(x^{(i+1)}, y^{(i+1)})$; i.e., using the update rule corresponding to the feature mapping ϕ:

$$\theta^{(i+1)} := \theta^{(i)} + \alpha[y^{(i+1)} - h_{\theta(i)}(x^{(i+1)})]\phi(x^{(i+1)})$$

[Note: If you prefer, you are also welcome to do this problem using the convention of labels $y \in \{-1, 1\}$, and $g(z) = sign(z) = 1$ if $z \geq 0, -1$ otherwise.]

3. **[32 points]Spam classification** In this problem, we will use the naive Bayes and perceptron training algorithms to build a spam classifier. The dataset we will be using is a subset of 2005 TREC Public Spam Corpus. It contains a training set and a test set. Both files use the same format: each line represents the space-delimited properties of an email, with the first one being the email ID, the second one being whether it is a spam or ham (non-spam), and the rest are words and their occurrence numbers in this email. The dataset presented to you is processed version of the original dataset where non-word characters have been removed and some basic feature selection has been done. [1] You can download the dataset from here.

   (a) (16 points) Implement both Naive bayes and perceptron algorithms to classify spam. Use your algorithms to learn from the training set and report the accuracy on the test set. Make sure to use the Laplace smoothing for Naive bayes (as discussed in class) to avoid any zero probabilities.

   Note: You should implement the learning algorithms from the first principles and not use any existing Matlab modules.

   (b) (4 points) Vary the number of examples used in training from 1000 to 9000 at the interval of 1000. Plot on a graph the test set accuracies (y-axis) as you vary the number of training examples (x-axis) for the two algorithms. These are called learning curves. What do you observe? Explain.

   (c) (2 points) Intuitively, some tokens may be particularly indicative of an email being in a particular class. For Naive bayes, we can try to get an informal sense of how indicative token i is for the spam class by looking at $\frac{p(x_j=i|y=1)}{p(x_j=i|y=0)}$ ($y = 1$ denotes that an email is a spam). Similarly, for neural networks, we can find out the most indicative tokens by looking at the relative weights assigned to the corresponding features. Find out the 5 most indicative words for classifying spam for both the algorithms.

   (d) (7 points) Train an SVM on this dataset using the LibSVM library, available for download from LibSVM. Convert the data into the format required by LibSVM and train a linear SVM. You can use the default set of parameters that come with the package. Report the accuracies on the test set.

   Note: You should submit any code that you wrote as a wrapper to get LibSVM to run on this dataset.

   (e) (3 points) Which of the three algorithms gives best test set accuracies? Which runs faster? Comment.

4. **[14 points]Decision Trees**

   (a) (4 points) Give decision trees to represent the following boolean functions:

      i. $A \wedge \neg B$
      ii. $A \vee [B \wedge C]$
      iii. $A\ XOR\ B$
      iv. $[A \wedge B] \vee [C \wedge D]$

   (b) (5 points) Consider the set of examples given in Table 4b:

      i. (2 points) What is the entropy of this collection of training examples with respect to the target classification?
      ii. (3 points) What is the information gain of a_2 relative to these training examples?

   (c) (5 points) Given two random variables X and Y, the mutual information between X and Y is defined as $I(X, Y) = H(Y) - \sum_{x_k} H(Y|X = x_k) * P(X = x_k)$. Show that mutual information is symmetric i.e. $I(X, Y) = I(Y, X)$.

[1] Some of the pre-processing was done for the Machine Learning course taught by Pedro Domingos at the University of Washington.

Instance	Classification	a_1	a_2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T
6	-	F	T

Table 1: Examples

5. **(23 points) Neural Networks:**

   (a) (10 points) One of the ways to avoid overfitting during neural network learning is to add a penalty term to the cost function, which penalizes large weights. This has the effect of trading-off minimization of the squared error with keeping some function of the weights' magnitude low. Typically, 2-norm of the weight vector is used as the penalty term. Consider the alternate error function defined as:

$$J(\theta) = \frac{1}{2} \sum_{i \in \{1 \cdots m\}} \sum_{k \in outputs} (y_k^{(i)} - o_k^{(i)})^2 + \gamma \sum_{l,k} \theta_{kl}^2$$

   Derive the stochastic gradient descent update rule for this definition of J. Show that it can be implemented by multiplying each weight by some constant before performing the standard gradient descent update.

   (b) (13 points) Consider learning the target concepts corresponding to circles in the x, y plane. Describe each hypothesis h_{cr} by the co-ordinates of the center (c_x, c_y) and the radius r of the circle. An instance (x, y) is labeled positive by hypothesis h_{cr} if and only if the point (x, y) lies inside the corresponding circle.

      i. (3 points) Write the mathematical expression for the classification rule imposed by a hypothesis h_{cr} as defined above.

      ii. (2 points) Let the training error of a hypothesis be defined as in the case of perceptron i.e. number of examples classified incorrectly by the hypothesis. Can you devise a gradient descent algorithm to the learn the hypothesis minimizing this error function. Argue.

      iii. (3 points) Devise an alternate error function so that error is a continuous function of the inputs. You should come up with an error function which is reasonable i.e. makes intuitive sense.

      iv. (3 points) Derive the gradient descent rule for the error function defined in the part above.

      v. (2 points) Is your algorithm guaranteed to converge the global optimum? Why?

Reminder: Make sure to include a printout of your code and figures (for the programming questions) in your submission.