CS 536: Homework 2

Due: Feb 26, 6pm EST

Instructions:
Same as homework 1.

1 Decision Lists (10 pts)

Show that the number of decision lists over n Boolean variables is at most $n!8^n$. Let Y be the $0/1$ output of a decision list over n Boolean variables $X_1, X_2, \ldots X_n$. Show that Y can also be written as a linear classifier over $X_1, X_2, \ldots X_n$, i.e., Y can be written as $sgn(b + \sum_i w_i X_i)$.

k-decision lists:
The decision lists that we saw in the class test the value of one variable at each step. In a k-decision list is one can test the value of any function of up to k variables in each step. So, the decision lists from the lecture are 1-decision lists. Show that if $Y = f(X_1, X_2, \ldots X_n)$ where $f()$ is a k-decision list, then given training data, one can find a k-decision list of zero training error in time $O(n^k)$. How much training data would one need to guarantee the true error[1] of the list you found will likely be at most ϵ?

2 Perceptron and Margin (20 pts)

Design a training set consisting of $O(n)$ points in $\Re^n$ that is linearly separable but on which the perceptron algorithm will take $\exp(\Omega(n)))$ steps to converge. **Note:** the Euclidean length of every point in your construction should be at most polynomial in n.

[Hint: Consider using a subset of $\{0, 1\}^n$]

3 SVMs with no label (20 pts)

You are given a set of m points of unit length in $\Re^d$ but with no labels. However, you are given the promise that there exists a non-trivial[2] way to label them as $+/-$ such that they become separable with margin $\frac{1}{4}$. Give an efficient algorithm to find such a labeling. Your algorithm should run in time polynomial in m and d.

4 Kernels (10 pts)

Recall that $K : X \times X \mapsto \Re$ is a legal kernel if there exists an implicit function ϕ such that $K(x, y) = \phi(x) \cdot \phi(y)$. (Here, X is our space of examples.) Often the easiest way to prove that some function is a legal kernel is to build it out of other legal kernels. In particular, suppose $K(x, y) = \phi(x) \cdot \phi(y)$ and $K'(x, y) = \phi'(x) \cdot \phi'(y)$ where $: X \mapsto \Re^N$ and $: X \mapsto \Re^{N'}$ for some N, N' . (Let's not worry about infinite-dimensional implicit feature spaces.)

1. Show that for any constant $c \geq 0$, cK is a legal kernel.

[1] As in classification, the error here is 0/1 loss, i.e., the probability that a new example drawn from the same distribution as the training set will be labeled incorrectly.

[2] A trivial labeling is one that assigns all of them as + or -.

2. Show that the sum, $K + K'$, is a legal kernel.
3. Show that the product, KK', is a legal kernel.

5 Decision trees and Decision lists (10pts)

Decision tree rank: The rank of a decision tree is defined as follows. If the tree is a single leaf then the rank is 0. Otherwise, let r_L and r_R be the ranks of the left and right subtrees of the root, respectively. If $r_L = r_R$ then the rank of the tree is $r_L + 1$. Otherwise, the rank of the tree is the maximum of r_L and r_R.

1. Prove that a decision tree with ℓ leaves has rank at most $\log_2 \ell$.
2. Show that if one can fit a training data using a decision tree with ℓ leaves, then one can also fit the data using a k-decision list for $k = \log_2 \ell$. You can assume binary features.

6 More Kernels (10 pts)

1. Show that if $K : X \times X \mapsto \Re$ is a legal kernel then so is K^d for any $d \in \mathbb{N}$. Here, K^d refers to the Kernel that produces dot products as $K^d(x, x')$.
2. Let $\hat{X}$ denote the set of all finite subsets of X. Prove that if K is a valid kernel on $X \times X$ then
$$\hat{k}(A, B) = \sum_{x \in A, x' \in B} k(x, x')$$
is a valid kernel on $\hat{X} \times \hat{X}$.
3. Prove that if $\sigma : X \mapsto X$ is a function and $K(x, x')$ is a valid kernel, then so is $K(\sigma(x), \sigma(x'))$.
4. Given a kernel K with feature map Φ, construct a corresponding normalized kernel K' by normalizing the feature map Φ such that all points have unit length in the new space. Give an expression for computing dot products in the normalized feature space.
5. Given an example of a kernel with two valid feature maps Φ_1 and Φ_2, mapping into spaces of different dimensions.

7 Kernel Perceptron (20pts)

The basic Perceptron algorithm that we saw in the class will loop forever if the data is not linearly separable. One way to fix this is to stop the algorithm after T passes over the data for some fixed value T. Is this a good strategy? Justify your answer with the help of examples.

Download the web spam dataset from the course webpage. This is a dataset of web pages classified as spam or not spam.

In the downloaded archive you will find:
webspam_wc_normalized_unigram.svm: This file contains info on 350000 web pages. Each webpage is a feature of length 254. The file contains info for each webpage, one per line. Each line has the following format *label feature_index1:feature_val1 feature_index2:feature_val2 ..... .* Label is $+1/-1$. "feature_index:feature_val" format describes that the given feature index has the corresponding value. For instance if the line is *[+1, 10:0.001 112:1.3]*, then it means the document has label +1, feature number 10 has value 0.001, feature number 112 has value 1.3 and all other features are 0.

1. Randomly partition the dataset into 250000 examples for training and 100000 for testing.
2. Implement the Perceptron algorithm mentioned above. Train it with different values of T. Plot T vs accuracy on test data. Which value T^* gives the best tradeoff between training time and accuracy?

3. Fix the value of T^* found above. Again randomly partition the dataset into 250000 examples for training and 100000 for testing. Implement a kernelized version of your algorithm with $T = T^*$. Choose a Gaussian Kernel and experiment with different values of σ. Plot σ vs accuracy on test set. Which value of σ achieves the best accuracy on the test set?
4. Come up with your own modification of the Perceptron algorithm that can outperform your implementation both in terms of train time and test set accuracy. Describe your new algorithm and show results.