Machine Learning *CS 4364/6364, Fall 2016*

HW4

Instructions

Collaboration: You are allowed to discuss the problems with other students. However, you must write up your own solutions for the math questions, and implement your own solutions for the programming problems. Please list all collaborators and sources consulted at the top of your homework.

Submission: Please submit homework pdf's at this address: machine.learning.gwu 'at' gmail.com. Electronic submissions are preferred. Math can be formatted in Latex (some easy editors for Latex are Lyx, TexShop), Word, or other editors. Math solutions can be optionally submitted on paper at the Department of Computer Science in person (or in the dropbox to the right of the entrance to SEH 4000 if submitted past 5 pm), but all code should be submitted electronically. The assignment is due Friday, November 18, at 5 pm.

Questions

1. **VC dimension - 10 pts**

    1. Let the input space be the real line, and let H be the hypothesis class of *intervals*. That is, each hypothesis h is associated with a closed interval $[a, b]$, for some constants $a \leq b$, and $h(x)$ is $+1$ if and only if x lies within this interval. What is the VC dimension of H? Prove that your answer is correct.
    2. Let the input space be the real line, and let H be the hypothesis class of *unions of k intervals*. That is, each hypothesis h is associated with k closed intervals $[a_1, b_1], \ldots, [a_k, b_k]$ (for constants $a_1 \leq b_1 \leq \cdots \leq a_k \leq b_k$); and $h(x)$ is $+1$ if and only if x lies in the union of these intervals. What is the VC dimension of H? Justify your answer.
    3. Let the input space be $\mathbf{R}^2$, and let H consist of all *homogeneous* linear separators (i.e. linear separators which pass through the origin). Show that H has a VC dimension of 2.

2. **Online learning - MATLAB - 10 pts**
    NOTE: This problem is required for grad students. Undergrads can solve it for extra credit

    In this problem you will implement and plot a learning curve for a simple online learning algorithm that uses multiplicative updates. The algorithm is Static-Expert, which is also similar to the Weighted Majority Algorithm. The pseudocode is given below. In this regression framework, we will use $\texttt{loss}(\hat{y}, y) = (\hat{y} - y)^2$, which can be applied to any prediction $\hat{y}$. The prediction of the algorithm is given in the first line inside the first for-loop. As experts, we will simply use the components of x, so the loss of expert i will therefore be $\texttt{loss}(\hat{y}_i, y) = (x_i - y)^2$.

Use the Cloud data provided (already downloaded from UCI Machine Learning Repository and included in the HW). Filter it to use the 7th feature (the 7th column) as the label, and remove this column from the input features. For this online learning problem, do **not** permute the data set since the order matters. You will use this batch of data to simulate a stream by going through the whole data set once.

Algorithm 1 Static-Expert Algorithm [Herbster and Warmuth '98]

```
Input: $d$ the number of features (columns of the data), and learning rate $b$
Initialize: $t = 1$, $p_t(i) = 1/d$ for all $i \in \{1, \dots, d\}$
For each example $(x, y)$ in the stream
    Output prediction: $\hat{y} = \sum_{i=1}^{d} p_t(i) x_i$
    For $i \in \{1, \dots, d\}$
        loss $= (x_i - y)^2$
        $p_{t+1}(i) = p_t(i) \exp\{-b \cdot \texttt{loss}\}$
    Normalize the distribution $p_{t+1}$        // keep track of the sum inside the for-loop.
    $t = t + 1$
```

1. Let the learning rate $b = 1$. Make a plot of the learning curve. The x-axis is iteration t. The y axis is the current value of the average loss of the algorithm. The average loss at time t is given as,

$$L_t = \frac{1}{t} \sum_{i=1}^{t} Loss(i)$$

   where $Loss(i)$ is the loss at iteration i.

   Turn in your code, and this plot.

2. Experiment with 3 other values of b, and plot the results super-imposed on the plot above. How does the performance change with b?

3. **LASSO and Ridge regression - MATLAB - 15 pts**

We've discussed the difference between LASSO and Ridge regression due to their different regularization models (L1 norm v.s. L2 norm). That is, the regularization term in LASSO is $\lambda||\boldsymbol{w}||_1$, while the regularization term in ridge regression is $(\lambda/2)||\boldsymbol{w}||^2$ (where $\boldsymbol{w}$ denotes the set of parameters for the linear regression model). LASSO typically enforces more *sparsity* on the resulting $\boldsymbol{w}$. That is, the resulting classifier will have fewer non-zero coordinates. In this exercise, you'll explore differences in performance between LASSO and Ridge regression algorithms.

You will be using the following data sets:

A. A randomly simulated data set that you will create as follows. Create training and test data sets as follows:

$$\begin{aligned}
\texttt{trainX} &= \texttt{randn}(1000, 20)\\
\texttt{trainY} &= \texttt{randn}(1000, 1)\\
\texttt{testX} &= \texttt{randn}(500, 20)\\
\texttt{testY} &= \texttt{randn}(500, 1)
\end{aligned}$$

B. The Cloud data set used in Question 2 with the same processing (use the 7th column as a label, remove it from the input features, and do not permute the dataset). Use the first 800 points as a training set, and use the remaining points as the test set.

C. **[Required for grad students, and optional for undergrads]**: The Forest Fires data set from the UCI Machine Learning Repository; you can download it here: `http://archive.ics.uci.edu/ml/datasets/Forest+Fires`. First randomly permute the data. Use the **logarithm** of the 13th feature (13th column) as labels, and remove it from the input features. Use the first 400 points for training and the remaining for testing. **NOTE:** there are several columns in text (month, and day). You can delete them, but this might hurt performance. Alternatively, you can preprocess the data to convert these features into numeric categories.

For each of the data sets above, answer the following questions:

1. Fit a regressor to the data set using the Matlab built-in function `lasso`. For more details, type `help lasso` in Matlab.
   (a) Train a LASSO model using 5 different values for the regularization parameter λ. For data set A, use $\lambda = 0.01, 0.05, 0.1, 0.2, 0.3$. For the other data sets, choose 5 values. Report and plot the number of non-zero coefficients in the resulting $\boldsymbol{w}$, as you vary λ.
   (b) Report the value of λ that yields the minimum number of non-zero coefficients in the resulting $\boldsymbol{w}$, and report the number of non-zero coefficients in that case. Keep track of this $\boldsymbol{w}$, we will refer to it as $\boldsymbol{w}_b$. [This step is only needed if any $\boldsymbol{w}$ has coefficients that are 0.]
   (c) For each of the classifiers learned in part (a), compute their test error. Plot the test error as a function of λ.
   (d) Report the value of λ that yields the $\boldsymbol{w}$ with the minimum test error. Save this $\boldsymbol{w}$ as $\boldsymbol{w}_d$. How many non-zero coordinates does $\boldsymbol{w}_d$ have? Keep track of this, as well as the answers for $\boldsymbol{w}_b$ for the comparison to Ridge regression below. It might be helpful to do so in a table, i.e. the number of non-zero coordinates and the test error for $\boldsymbol{w}_b$ and $\boldsymbol{w}_d$ for each data set.
   (e) For data set A, by increasing the value of λ, can you get most of the entries to be 0? Can you get every entry to be 0?

2. Now, we turn to Ridge regression. The MATLAB function $B = \texttt{ridge}(y, X, k)$ outputs a matrix B using ridge regression, where each column of B corresponds to a set of parameters based on a single λ, where each λ is stored in the vector k.
   (a) Repeat each of the experiment above using Ridge regression. For data set A, use $\lambda = 1, 50, 100, 200, 1000$. The values for λ can be different from the values used in LASSO.
   (b) Compare the two algorithms on each data set as follows: compare the number of non-zero coordinates of the $\boldsymbol{w}_d$'s, and compare the test error rates of the $\boldsymbol{w}_b$'s. Report your results and comment on them.

4. **Parameter Tuning and Cross Validation - MATLAB Programming - 10 pts**

   We will use character recognition data again, this time from the US Post Office. The digit-recognition data from the USPS dataset has been turned into a binary classification problem: 0 vs. All. In this problem, points in the dataset denoting the digit '0' are labeled 1 and all other points are labeled -1. This contains 7291 data points, each with 256 features.

   (i) `Parameter Tuning`. The default parameter values might not be the best choice for the data set in question. The parameters need to be tuned on a **separate** data set, typically called validation (or hold-out) data. Extract the first $\frac{1}{6}$th from the training data and labels as the validation set. The remaining $\frac{5}{6}$th will be used in the next section. For simplicity, instead of tuning all the parameters, we will only tune the C parameter.

   Using the `svmtrain` function and the validation data set, train an SVM classifier on the first $\frac{2}{3}$rd of the validation data (and labels) using each of the kernel functions and the 5 values of C specified in the table. Report the error rates on the remaining $\frac{1}{3}$rd of the validation data using `svmclassify`.

   [*Note : If you are using the built-in Matlab functions for SVM:*
   While computing the validation error for the Kernel SVM for various values of c, you might need to set the `autoscale` parameter to `false`. By default, it is set to `true`, and the columns of the training data are shifted and scaled to have zero mean unit variance. If all the validation errors turn out to be the same for the specified values of c, it's recommended to set the `autoscale` parameter to `false` during tuning.]

   Table 1:

Kernel	C=0.01	C= 0.1	C = 1	C = 20	C = 50
Linear					
Polynomial					
Gaussian Radial Basis Function					

   For each kernel, state the C value that minimizes the test error on the validation set (and remember it for the next problem).

   (ii) **Comparing learning algorithms** For this problem, use the remaining $\frac{5}{6}$th of the data that was *not* used for validation. You now have access to the following 4 learning algorithms:

   (a) SVM with each of the 3 kernels stated above. (For each kernel, use the best C on the validation set, computed above).

   (iii) **n-fold Cross Validation**
   n-fold cross validation is used to determine a robust estimate of the error of a learning algorithm. The basic idea is to divide the given data into n disjoint sets $\{S_1..S_n\}$ of equal size. The process now proceeds in n steps. In the first step, we run the learning algorithm on training data set composed of $S_2 \cup S_3 \cup \cdots \cup S_n$. The set S_1 is used to compute an error rate for the learning algorithm. Call this e_1. The same procedure is repeated $n - 1$ times (each time, set S_i is used as test data and the remaining sets are used as training

data) to compute error rates $e_2 ... e_n$. The performance of the learning algorithm is now determined by computing the average of these error rates.

Divide the data into 4 equal sized sets, call them $\{S_1, S_2, S_3, S_4\}$. For each learning algorithm, perform 4-fold cross validation, and report the resulting average error for each of the methods listed in (ii).