GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 6254 Spring 2018
Problem Set #4

Assigned: 12 Mar 18
Due Date: 28 Mar 18

Suggested reading:

- *Elements of Statistical Learning* (by Hastie, Tibshirani, and Friedman): Section 4.5 (pages 129–135) discusses optimal separating hyperplanes; Sections 12.1–12.3 (pages 417–438) discuss support vector machines and kernels in more detail.
- *Learning from Data* (by Abu-Mostafa, Magdon-Ismail, Lin): Sections 2.1–2.2 (pages 39–62) contain a beautiful description of the VC generalization bound that closely mirrors what we did in class.
- "Introduction to Statistical Learning Thoery" by Bousquet, Boucheron, and Lugosi: If you read and liked the beginning of this paper, try to go back through this and re-read Sections 1-3 (first 13 pages) and then read Section 4, which provides another take on VC theory. You can download the paper on the course web page.

Homework is due in class at the beginning of the class period on the due date.

For **distance learning students**, the homework should be scanned and uploaded to the t-square assignments page **one week after the regular due date**.

PROBLEM 4.1:
Calculate the growth function $m_{\mathcal{H}}(n)$ for the following classes of classifiers:

1. The set of classifiers on $\mathbb{R}$ that can be written as either $h(x) = \text{sign}(x - a)$ for some $a \in \mathbb{R}$ or $h(x) = -\text{sign}(x - a)$ for some $a \in \mathbb{R}$, i.e., the set of both positive and negative rays.
2. The set of classifiers on $\mathbb{R}$ that can be written as either

$$h(x) = \begin{cases} +1 & \text{for } x \in [a, b] \\ -1 & \text{otherwise} \end{cases}$$

or

$$h(x) = \begin{cases} -1 & \text{for } x \in [a, b] \\ +1 & \text{otherwise} \end{cases}$$

for some $a, b \in \mathbb{R}$, i.e., the set of both positive and negative intervals.

Consider classifiers defined by positive circles in $\mathbb{R}^2$, i.e., h such that for any $\boldsymbol{x} \in \mathbb{R}^2$,

$$h(\boldsymbol{x}) = \begin{cases} +1 & \text{if } \|\boldsymbol{x} - \boldsymbol{c}\|_2 \leq r \\ -1 & \text{otherwise,} \end{cases}$$

for some $\boldsymbol{c} \in \mathbb{R}^2, r \in \mathbb{R}$.

1. Show that for this set of classifiers, $m_{\mathcal{H}}(3) = 8$.
2. Argue that $m_{\mathcal{H}}(4) < 16$.

PROBLEM 4.2:
Suppose that our input space $\mathcal{X} = \mathcal{Re}$ and consider the hypothesis set

$$\mathcal{H} = \left\{ h : h(x) = \text{sign}\left(\sum_{i=0}^{d} c_i x^i \right) \quad \text{for some } c_0, \dots c_d \in \mathcal{Re} \right\}$$

In words, $\mathcal{H}$ is the set of classifiers obtained by evaluating some polynomial of degree d and comparing the result to a threshold. Prove that the VC dimension of $\mathcal{H}$ is exactly $d+1$ by showing that

1. There are $d+1$ points which are shattered by $\mathcal{H}$. [Hint: Root placement.]
2. There are no $d+2$ points which are shattered by $\mathcal{H}$.

PROBLEM 4.3:
Suppose that the VC dimension of our hypothesis set $\mathcal{H}$ is $d_{\text{VC}} = 3$ (e.g., linear classifiers in $\mathcal{Re}^2$) and that we have an algorithm for selecting some $h^* \in \mathcal{H}$ based on a training sample of size n (i.e., we have n example input-output pairs to train on).

1. Using the generalization bound given in class and derived in the handout, give a precise upper bound on $R(h^*)$ that holds with probability at least 0.95 in the case where $n = 100$. Repeat for $n = 1\,000$ and $n = 10\,000$.
2. Again using the generalization bound given in class, how large does n need to be to obtain a generalization bound of the form

   $$R(h^*) \leq \widehat{R}_n(h^*) + 0.01$$

   that holds with probability at least 0.95? How does this compare to the "rule of thumb" given in class?

PROBLEM 4.4:
In this problem we will compare the performance of traditional least squares, ridge regression, and the LASSO on a real-world dataset. We will use the "Boston House Prices" dataset which contains the median sale price (as of some point in the 1970's, when the dataset was created) of owner occupied homes in about 500 different neighborhoods in the Boston area, along with 13 features for each home that might be relevant. These features include factors such as measures of the crime rate; measures of school quality; various measures of density; proximity to things like highways, major employment centers, the Charles River; pollution levels; etc.[1] To get this dataset in Python, simply type

```
from sklearn.datasets import load_boston
boston = load_boston()
X = boston.data
y = boston.target
```

In order to judge the quality of each approach, you should split the dataset into a training set and a testing set. The training set should consist of 400 observations, and you can use the remaining observations for testing.

Before training any of these algorithms, it is a good idea to "standardize" the data. By this, I mean that you should take each feature (i.e., each column of the matrix `X`) and subtract off its mean and divide by the standard deviation to make it zero mean and unit variance. Otherwise, the regularized methods will implicitly be placing bigger penalties on using features which just happen to be scaled to have small variance. You should determine how to "standardize" your training data by appropriately shifting/scaling each feature *using only the training data*, and then apply this transformation to both the training data and the testing data so that your learned function can readily be applied to the test set.

For all parts of the problem below, I would like you to submit your code.

1. First, I would like you to evaluate the performance of least squares. You should implement this yourself using the equation we derived in class. Report the performance of your algorithm in terms of mean-squared error on the test set, i.e.,

$$\frac{1}{n_{\text{test}}}\|\boldsymbol{y}_{\text{test}} - \boldsymbol{X}_{\text{test}}\widehat{\boldsymbol{\theta}}\|_2^2.$$

2. Next, using the formula derived in class, implement your own version of ridge regression. You will need to set the free parameter λ. You should do this using the training data in whatever manner you like (e.g., via a holdout set) – but you should *not* allow the testing dataset to influence your choice of λ. Report the value of λ selected and the performance of your algorithm in terms of mean-squared error on the test set.

3. Finally, I would like you to evaluate the performance of the LASSO. You do not need to implement this yourself. Instead, you can use scikit-learn's built in solver via

```
from sklearn import linear_model
reg = linear_model.Lasso(alpha = ???)
reg.fit(Xtrain,ytrain)
reg.predict(Xtest)
```

[1]See `http://bit.ly/2lTueYY` for more details about this dataset.

Above, `alpha` corresponds to the λ parameter from the lecture notes. As in part (b), you will need to do something principled to choose a good value for this parameter. Report the value of `alpha` used in your code, the performance of your algorithm in terms of mean-squared error, and the number of nonzeros in $\boldsymbol{\theta}$. (You can get $\boldsymbol{\theta}$ via `reg.coef_`.)