ECE 6254, Spring 2017

Homework # 3

Due Thursday February 16, in class
(For distance students: due Thursday February 23 by 3:05pm EST)

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.

Problems:

1. In the notes I provided on kernels, we proved that if the function $k(\boldsymbol{u}, \boldsymbol{v})$ is a symmetric and positive semi-definite kernel, then it is an "inner product kernel" (i.e., there is a Hilbert space $\mathcal{H}$ with inner product $\langle \cdot, \cdot \rangle$ and a map $\boldsymbol{\Phi} : \mathbb{R}^d \to \mathcal{H}$ such that $k(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{\Phi}(\boldsymbol{u}), \boldsymbol{\Phi}(\boldsymbol{v}) \rangle$). Show that the converse is also true: an inner product kernel must be positive semi-definite. (This direction is much easier.)

2. Let $L(x) = \max(0, c - x)$ for some $c > 0$.

   (a) Sketch $L(x)$; make sure that c is labeled clearly on your plot.

   (b) We have been finding the optimal soft margin classifier by solving a constrained quadratic program.[1] In this problem, I want you to argue that there is an equivalent unconstrained formulation of the form:

   $$\underset{\boldsymbol{w},b}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n} L(?).$$

   In other words, fill in the question mark and then argue that this unconstrained problem is equivalent to the original constrained quadratic program we used to describe the optimal soft margin classifier in Lecture 6.

[1] Both the original primal problem and the dual problem can be expressed as particular instances of the general class of constrained quadratic programs. Read the supplemental notes for Lecture 8 for more details.

3. In this problem you will use SVMs to build a simple text classification system. To begin, you need to get the MNIST dataset. You can do this in Python using the commands

```
from sklearn.datasets import fetch_mldata
mnist = fetch_mldata('MNIST original')
```

This downloads the dataset and stores it in a default location (`~/scikit_learn_data/`). You can also use the optional parameter `data_home=path` in `fetch_mldata` to direct the dataset to a custom path if you wish. The first time you run this command, the data will be downloaded from `mldata.org`, but once it has been downloaded this will simply load the dataset into the variable `mnist`.

This data set contains 70 000 handwritten digits, each of size 28×28 pixels.[2] You should begin by putting this data into a more convenient form by setting

```
X = mnist.data
y = mnist.target
```

Each row of `X` corresponds to one image. You can use the following to plot the $\texttt{j}^{\text{th}}$ image:

```
import matplotlib.pyplot as plt
plt.title('The jth image is a {label}'.format(label=int(y[j])))
plt.imshow(X[j].reshape((28,28)), cmap='gray')
plt.show()
```

In this problem you will build a classifier to classify between the (sometimes very similar) images of the digits "4" and "9". To get just this data, you can use

```
X4 = X[y==4,:]
X9 = X[y==9,:]
```

There are a little under 7 000 examples from each class. You should begin by using this data to form three distinct datasets: the first 4 000 points from each class will be used for designing the classifier (this is the *training set*), and the remainder will be used to evaluate the performance of our classifier (this is the *testing set*).

Since SVMs involve tuning a parameter C, you will also need to set this parameter. We will do this in a principled way using the so-called "holdout method". This means that you take the training set and divide it into two parts: you use the first to fit the classifier, and the second – which we call the *holdout set* – to gauge performance for a given value of C. You will want to decide on a finite set of "grid points" on which to test C (I would suggest a logarithmic grid of values to test both $C \ll 1$ and $C \gg 1$). For each value of C, you will train an SVM on the first part of the set, and then compute the error rate on the holdout set. In the end, you can then choose the value of C that results in a classifier that gives the smallest error on the holdout set

[2] You can learn more about this dataset at `http://yann.lecun.com/exdb/mnist/`.

Note: Once you have selected C, you may then retrain on all the entire training set (including the holdout set), but you should **never** use the testing set until every parameter in your algorithm is set. These are used exclusively for computing the final test error.

To train the SVM , you can use the built in solver from scikit-learn. As an example, to train a linear SVM on the full dataset with a value of $C = 1$, we would use

```
from sklearn import svm
clf= svm.SVC(C=1.0,kernel='linear')
clf.fit(X,y)
```

Once you have trained the classifier, you can calculate the probability of error for the resulting classifier on the full dataset via

```
Pe = 1 - clf.score(X,y)
```

(a) Train an SVM using the kernels $k(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^{\mathrm{T}}\boldsymbol{v} + 1)^p$, $p = 1, 2$, that is, the inhomogeneous linear and quadratic kernel. To do this, you will want to set the parameters `kernel='poly'` and `degree=1` or `degree=2`.

For each kernel, report the best value of C, the test error, and the number of data points that are support vectors (returned via `clf.support_vectors_`. Turn in your code.

(b) Repeat the above using the radial basis function kernel $k(\boldsymbol{u}, \boldsymbol{v}) = e^{-\gamma\|\boldsymbol{u}-\boldsymbol{v}\|^2}$ (by setting the parameters `kernel='rbf', gamma=gamma`. You will now need to determine the best value for both C and γ. Report the best value of C and γ, the test error, and the number of support vectors

(c) For each kernel, turn in a 4×4 subplot showing images of the 16 support vectors that violate the margin by the greatest amount (these are, in a sense, the "hardest" examples to classify), and explain how these are determined. Above each subplot, indicate the true label (4 or 9).

To help get started with this, you can use the following code:

```
f, axarr = plt.subplots(4, 4)
axarr[0, 0].imshow(X[j].reshape((28,28)), cmap='gray')
axarr[0, 0].set_title('{label}'.format(label=int(y[j])))
...
plt.show()
```

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

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

(b) 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.

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

   (a) Show that for this set of classifiers, $m_{\mathcal{H}}(3) = 8$.

   (b) Argue that $m_{\mathcal{H}}(4) < 16$.

6. Suppose that our input space $\mathcal{X} = \mathbb{R}$ and consider the hypothesis set

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

   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

   (a) There are $d + 1$ points which are shattered by $\mathcal{H}$. [Hint: Root placement.]

   (b) There are no $d + 2$ points which are shattered by $\mathcal{H}$.

7. Suppose that the VC dimension of our hypothesis set $\mathcal{H}$ is $d_{\text{VC}} = 3$ (e.g., linear classifiers in $\mathbb{R}^2$) 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).

   (a) 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 $n = 10,000$.

   (b) 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?