Problem Set 8: SVMs

CMSC 422, Fall 2017
Assigned 11/14, Due 11/29

Submission Instructions
You will submit a zip file named YourID_ps8.zip containing a directory named YourID. So if your login ID is steveholt, you will submit steveholt_ps8.zip containing the folder steveholt. Inside the folder should be your version of the file ps8.py. Separately (not in the zip), submit a PDF named YourID_ps8.pdf with the necessary answers to written questions. Comment your code, and paste it at the end of the PDF (make sure it is well-formatted enough to be readable in the PDF). Failure to follow these guidelines will result in points being taken off.

Grading
In addition to grading your PDF, we will test your code on withheld data to make sure it works. Do not modify the provided names and interfaces, or the autograder might fail.

There are four main problems in this problem set. If something is marked with **Code**, then code must be written for that part. If something is marked with **Report**, then it must be answered/addressed in your written report. We provide you with some code in a file called PS8.py. Add all additional code to this file. Do not rename the file. All your functions must have the definitions as detailed below.

Problem 1: Binary Classification Brute Force

You will implement a linear classifier with a decision boundary that maximizes the margin between positive and negative samples (SVM). The details of SVM can be quite complex, and so here you will implement a simpler version of it for 2D data.

We provide you with a function for generating data (features and labels):

```
data = generate_training_data_binary(num)
```

`num` indicates the training set. There are four different datasets (`num`=1, `num`=2, `num`=3, num=4) which you will need to train on and provide results for. All the datasets are linearly separable. `data` is a list containing 2D features and their corresponding label in each row. So `data[0]` will have the form [x1, x2, c] where [x1, x2] is the feature for the first training sample and it has the label c. c can be -1 or 1.

We have also provided you with a function to plot the training data so you can visualize it:

```
plot_training_data_binary(data)
```

Code: You must write the following function:

```
[w,b,S] = svm_train_brute(training_data)
```

This function will return the decision boundary for your classifier and a list of support vectors `S`. That is, it will return the line characterized by `w` and `b` that separates the training data with the largest margin between the positive and negative class.

Since the decision boundary is a line in the case of separable 2D data, you can find the best decision boundary (the one with the maximum margin between positive and negative samples) by looking at lines that separate the data and choosing the best one (the one with the maximum margin). The maximum margin separator depends only on the support vectors. So you can find the maximum margin separator via a brute force search over possible support vectors. In 2D there can only be either 2 or 3 support vectors.

Report: Why is it the case that there are 2 or 3 support vectors? (You may find it helpful to draw out some examples to see why this is true).

In order to implement your training function, you will need to write some helper functions:

Code: Write a function that compute the distance between a point and a hyperplane. In 2D, the hyperplane is a line, defined by w and b. You code must work for hyperplanes in 2D. Do not worry about higher dimensions.

```
dist = distance_point_to_hyperplane(pt, w, b)
```

Code: Write a function that takes a set of data, and a separator, and computes the margin.

```
margin = compute_margin(data, w, b)
```

Code: You will need to write the following function to test new data given a decision boundary. The following function will output the class (`c`) for a given test point (`x`).

```
c = svm_test_brute(w,b,x)
```

Use the above helper functions when writing your training function.

Report: Give the details of your method in your report. Plot the training data and your decision boundary for each of the three datasets provided by the `generate_training_data_binary` function. Also provide the margins for each of the separators in your report.

Problem 2: Multi-Class Classification Brute Force

Now, let's assume we're working with data that comes from more than two classes. You can still use an SVM to classify data into one of the C classes by using multiple SVMs in a one-vs-all fashion. Instead of thinking about it as class 1 versus class 2 versus class 3… you think of it as class 1 versus not class 1, and class2 versus not class 2. So you would have one binary classifier for each class to distinguish that class from the rest.

We will provide you with a function to generate training data.

```
[data, C] = generate_training_data_multi(num)
```

Similarly to the binary data generation function, `num` indicates the particular set of data, and there are 2. Each dataset has a different number of classes. The classes are identified from 1 to `C`, where `C` is the number of classes.

We have also provided you with a function to plot the training data so you can visualize it:

```
plot_training_data_multi(data)
```

Code: Implement the following function:

```
[W, B] = svm_train_multiclass(training_data)
```

This function should use your previously implemented `svm_train_brute` to train one binary classifier for each class. It will return the C decision boundaries, one for each class (one-vs-rest). `W` is a list of w_is and `B` is a list of b_is where $w_i x+b_i$ is the decision boundary for the i^{th} class.

Code: You will also implement a test function:

```
c = svm_test_multiclass(W,B,x)
```

This function will take the C decision boundaries as input and a test point `x`, and will return the predicted class, `c`. There are two special cases you will need to account for. You may find that a test point is in the "rest" for all of your one-vs-rest classifiers, and so it belongs to no class. You can just return -1 (null) when this happens. You may also find that a test point belongs to two classes. In this case you may choose the class for which the test point is the farthest from the decision boundary.

Report: Plot the training data, and the decision boundaries for each one-vs-rest problem on all multi-class datasets.

Problem 3: Kernel SVM

Suppose we have the following 1D training data.

Feature	Label
(-1,-1)	-1
(-1,1)	1
(1,1)	-1
(1,-1)	1

This data is not linearly separable. However, if you transform the data from $[x_1,x_2]$ to $[x_1,x_1x_2]$, it becomes linearly separable.

Code: Edit the `generate_training_data_binary` function to handle an additional case that generates the above training data (using `num`=5).

Code: Write a function that applies the transformation above and uses your `svm_train_brute` function to find the separating hyperplane.

```
[w,b] = kernel_svm_train(training_data)
```

Report: Plot the original data, along with the transformed data and the separator. Is this the lowest-dimensional transformation of the data that has a linear separator? If so, explain why. If not, give a lower-dimensional transformation.

Challenge Problem: Kernel Trick

Given the data from problem 3, use the kernel trick to find w and b. To do this, you will use the polynomial kernel $K(x_i,x_j) = (1+x_i \bullet x_j)^2$ and maximize:

$$Q(a) = \Sigma a_i - \tfrac{1}{2}\Sigma a_i a_j y_i y_j K(x_i,x_j)$$

Subject to: $\Sigma a_i y_i = 0$ and $a_i \geq 0$

By maximizing Q, you will find a_is which will be used to construct w and b. Since you want to maximize Q with respect to a, you will need to solve:

$$\nabla Q = 0$$

This will give you four equations in addition to the constraint equations above ("Subject to"). With this you can solve for the a_is. You will then use the a_is to find w and b.

Remember that $K(x_i,x_j) = \Phi(x_i) \bullet \Phi(x_j)$, so for a given $x=[x_1,x_2]$, we have

$$\Phi(x) = [1, \sqrt{2}\, x_1, \sqrt{2}\, x_2, \sqrt{2}\, x_1 x_2, x_1^2, x_2^2]$$

Report: Show that this is true.

Once you solve for the a_is you will need Φ to construct w:

$$w = \Sigma a_i y_i \Phi(x_i)$$

And you can solve for b using one of the support vectors (s).

$$w\Phi(s) + b = -1 \text{ or } w\Phi(s) + b = +1$$

Report: Work through this problem by hand (in Word, or LaTeX). Give your final w, and b. What are the support vectors? How do you know this? How does this compare to the kernel in problem 3? Plot the data and your separator in the original space. For some test $x=[x_1,x_2]$, what will the decision be? (Give a formula for the decision in terms of x_1 and x_2.)