Further Details for Homework 1, CS 273A

Kevin Bache

October 3, 2015

Problem 1a describes how to generate a single dataset. We're going to be generating a bunch of datasets in parts b and c. The generating process works like this:

For each dataset:

1. Draw μ_{-1} and μ_1 from a $Normal_9(\vec{0}, \mathbf{I}_9)$. These will be the centers of your clusters. Here $Normal_9(\vec{0}, \mathbf{I}_9)$ indicates that we're drawing form a 9-dimensional Gaussian distribution which has it's mean set to a 9-dimensional vector of all zeros and it's covariance matrix set to a 9-by-9 identity matrix.

2. Pick α in the range $(0, 1)$. That means you can't pick 0 and you can't pick 1, but you can pick any real number in between (in fact you'll have to use various values of α for different parts of the problem. More details below).

3. Pick N, the number of training datapoints for this dataset and N_{test}, the number of test datapoints for this dataset. (Sometimes you'll be experimenting with different values of N and sometimes with different values of N_{test}. See below for more information on when to set which, and when in doubt setting N_{test} to be the same as N is a reasonable default.)

4. For each cluster center i in $(-1, 1)$:

   (a) Draw N datapoints $Normal_9(0, \alpha * \mathbf{I}_9)$ and call the resulting N-by-9 matrix $X_{train,i}$. Save the labels for these N datapoints in a vector we'll call $y_{train,i}$ (which is set to $[i, i, ..., i]$).

   (b) Draw N_{test} datapoints $Normal_9(0, \alpha * \mathbf{I}_9)$ and call the resulting N_{test}-by-9 matrix $X_{test,i}$. Save the labels to these datapoints as well in a vector we'll call $y_{test,i}$ (which is also set to $[i, i, ..., i]$).

5. Now combine the rows from $X_{train,-1}$ and $X_{train,1}$ together to make X_{train}. Concatenate $y_{train,-1}$ and $y_{train,1}$ to make y_{train}. Do the same for the test data to make X_{test} and y_{test}.

In problem 1b you're going to come up with a closed-form solution to determine a decision boundary between the two clusters in a given dataset. Check the sections of the textbooks by Kevin Murphy and Chris Bishop on Fisher's Linear Discriminant Analysis for some helpful ideas on how to do this. We want to see how accurate this method is across a wide range of N and α values. You can show this data lots of ways, but one example might be to make a table where each row represents a single value of N and each column represents a single value of α, and each internal cell in the table contains accuracy information about datasets created with that α-and-N combination. That said, feel free to approach this any way you'd like as long as you somehow convey the same information.

Then for each α-and-N combination:

1. Generate a bunch of data sets (e.g., 100).

2. For each dataset:

   (a) Train your model by feeding in X_{train} and y_{train} to your decision boundary formula. It will output the weights for a linear decision boundary. Congratulations, you've just learned your first machine learning model!

(b) Test your model by feeding X_{test} (**without** y_{test}) to your trained model and using the model to generate $\hat{y}_{test}$. In machine learning, we usually put a hat above a variable name when we want to say that it represents a prediction for what we think that value should be. So $\hat{y}_{test}$ would represent our model's predictions for what it thinks the values of the vector y_{test} would be.

(c) Compare $\hat{y}_{test}$ to y_{test} to come up with an accuracy score for this model on this dataset. You could use a variety of different accuracy scores, but 0-1 loss (i.e.: percent correct) is a good default for our purposes.

(d) Record the accuracy score for this dataset on this set of hyperparameter values (α and N are your hyperparameter values).

3. In this cell, report the mean and standard deviation of the test accuracies across the bunch of datasets you generated for this cell. This tells us:

   (a) How accurate this model is with this set of hyperparameters.

   (b) How much variability there is in that accuracy across randomly generated datasets.

For 1c, you're going to do something similar to 1b, but now with an iterative algorithm, namely the perceptron learning algorithm, rather than the closed-form learning algorithm we used in part 1b. Furthermore, where in 1b we were interested in varying α and N, in 1c we're going to fix alpha and N (i.e.: choose your favorite α-and-N combination from part 1b and use those values for ALL of part 1c). Instead, in part 1c we're going to see how the perceptron learning algorithm performs across various values for N_{test}, the number of datapoints in your test set, and $N_{iterations}$, the maximum number of iterations that you allow your perceptron learning algorithm to perform. Again, you might make something like a table with each column representing a particular value of $N_{iterations}$ and each row representing a particular value of N_{test} and then:

For each N_{test}-and-$N_{iterations}$ combination:

1. Generate a bunch of data sets (e.g., 100)

2. For each dataset:

   (a) Train your perceptron model on the X_{train} and y_{train} for this dataset.

   (b) Test the trained perceptron model on X_{test}, outputting $\hat{y}_{test}$) as in 1b.

   (c) Calculate and store an accuracy score from the y_{test} and $\hat{y}_{test}$ vectors in 1b.

3. Report the mean and standard deviation of the test accuracies across the bunch of datasets you generated for this cell as in 1b.

Problem 2 is very similar to problem 1 but you just generate your cluster means for each dataset differently. Instead of generating your initial cluster means from a $Normal_9(\vec{0}, \mathbf{I}_9)$, you're now going to generate your cluster means for each dataset as follows:

1. Randomly select a number between 2 and 7. Call it k.

2. For each cluster center in $(-1, 1)$:

   (a) Set μ_i to be a vector of all 0s of length 9.

   (b) Randomly set k entries in μ_i to be 1.

Now as before, you wrap those cluster centers into Gaussians with small variance in order to generate your actual data from each cluster. In problem 1 the variance of your small Gaussians was α * $\mathbf{I}_9$ and it will be here again. However, in problem 1 you varied your α values in different cells of the output table, whereas here you'll just select one α value (a small one, e.g.: 0.3, 0.1, 0.01 or something like that) and use that for the entirety of problem 2.

As a final note, it's ok if some of your noisy "pixel" values go above 1 or below 0. That might not make much sense for actual pixel intensity values, but it's fine for what we're doing here, which is just generating data that looks a little different than the data from problem 1.

Good luck!