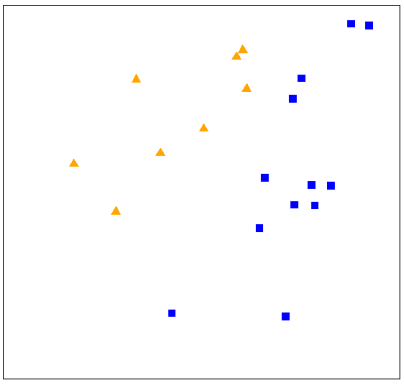
Yeshiva University

AIM 5007 Neural Networks and Deep Learning
Problem Set 1

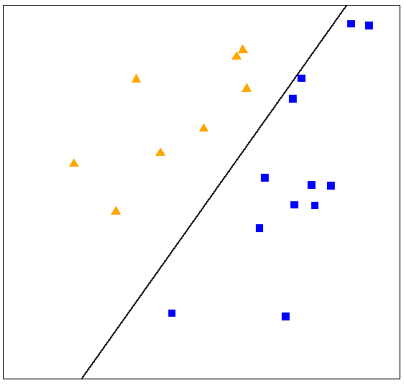
Introduction

In this problem set/mini-project, we will build a working perceptron to classify observations in a linearly separable data set. In machine learning terms, our goal is to take a training data set that has a specific (in this case linear) target function and learn that target function by using the training data to approximate it.

In order to be able to visualize our work, we will restrict the data to a two-dimensional space of continuous numerical inputs ($X = \mathbb{R}^d$) and one binary output ($\mathbb{Y} = \{-1, +1\}$). Consider the following plot in two dimensions, where each point is classified as either a "success" (the blue squares) or a "failure" (the orange triangles):



Can you visualize a line that would separate the orange triangles from the blue squares? I would imagine you can visualize several lines of slightly different slopes and intercepts! Here is one (this happens to be the one I used to generate the data):



This is where we make a somewhat loose distinction between classical statistical modeling and modern machine learning. In a classical statistical model, we would assume a structure (a linear separation) and then estimate the parameters for that structure. If you recall your linear regression coursework, you estimated the coefficients of a linear model.

In machine learning, we instead try to learn the structure by examining the data. Instead of generating the data from a model (statistics), we generate the model from data. In this problem set, we are assuming some structure because we are working with a perceptron model. Neural networks will give us much more flexibility to estimate nonlinear functions.

The primary goal for this problem set is to use a perceptron learning algorithm to determine, using the input data, an approximation for the true function that separates the data into successes and failures. (And please note that, like in the binomial distribution, successes can be either good or bad. The label is arbitrary.)

Lab Instructions

Using a programming environment of your choosing (Python notebooks are the preference but please reach out to me if you wish to use something else), complete each of the following steps. We will use these exercises to learn how a perceptron works. Later, we will create the final deliverable.

1. **Generate Some Data**. Simulate 100 data points in the space $\mathbb{R}^2$. You can generate two vectors of random numbers in an interval (for instance, let each of x_1, x_2 be in the interval $(-10,10)$ and use that as your initial data.

2. **Choose a Target Function**. Choose a line that you will use to separate the data into successes and failures. For instance, you might choose the line $1.5x_1 - x_2 - 1 = 0$. You are free to choose whatever line you like, but make sure some of your points will get classified as successes and some as failures. Your chosen line should split your data set into two parts.

   Using your chosen line, classify each point as a success or as a failure. You can do this by calculating, for instance, $1.5x_1 - x_2 - 1$ for each point and assigning $+1$ when this formula gives an answer greater than or equal to zero and -1 when the formula gives an answer less than zero.

3. **Plot Your Data**. Create a scatterplot like the first plot above that labels the points according to their classification. Can you visualize the line you chose? If not, go back and make sure you are assigning labels correctly. Create a second plot that also includes the target line. Does it correctly separate the data?

4. **Create the Training Data**. Package your data together into a data frame. The data frame should have three columns (one each for $x_1, x_2, and\ y$), where x_1 and x_2 are real numbers and y is either -1 or 1.

5. **Initialize the Perceptron**. We are going to start by proposing a simple line that could be the target function. (It almost certainly won't be correct!) Remember, we won't usually know the target function, but in this case you do. Our perceptron is going to take the following form:

$$f(x_1, x_2) = sign(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2)$$

Notice that this is a line in two dimensions. There are three coefficients, called weights, in our model. We put a weight on each of the two input variables (w_1 on x_1, w_2 on x_2) and we also create what we will later refer to as the bias (w_0). Our perceptron algorithm's job is to learn weights that fit the data well.

We will implement our sign function with a small modification. If $w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 = 0$, assign a positive sign. If you generated random continuous inputs, this is unlikely to happen, but let's make the choice now.

Create a vector of weights $w(0) = (w_0, w_1, w_2)$ and set all three weights to zero. We will talk more about how to initialize weights later in the course. For now, just start with all zeros. Note that we write $w(0)$ to indicate this is our first attempt at a vector of weights. We will then choose $w(1)$, $w(2)$, and so on until we find our solution.

6. **Classify the Data**. Using the function in part 5 above, classify each data point using these initial weights. That is, calculate $f(x_1, x_2)$ for each point and assign the correct output. These are our predicted classifications.

7. **Generate the Confusion Matrix.** Create a simple table of your predicted results versus actual results. It should look something like this:

	Actual $+1$	Actual -1
Predicted $+1$	# observations correctly predicted to be $+1$	# observations incorrectly predicted to be $+1$
Predicted -1	# observations incorrectly predicted to be -1	# observations correctly predicted to be -1

Those of you who have taken machine learning will recognize this as a confusion matrix. Calculate your accuracy as well. This is just the percentage of observations that you got right. If you correctly classified 83 out of 100, your accuracy is 83%.

8. **Evaluate the Result.** Are all the observations correctly classified? Probably not, and that's okay. If all the observations are classified correctly, we are done. We have learned a set of weights that fits the data well. We would output the solution and use it to classify future observations where the classification is not known.

If there are mistakes in the classifications, though, we need to modify our weights and try again. In that case, proceed to the next step.

9. **Update the Weights.** We will use a simple rule to update the weights. Choose one observation at random from the observations that were misclassified. This observation has individual input values of x_{1i} and x_{2i} and a correct output value of y_i. Note that y_i is the true classification, not our predicted classification.

   We update the new weights according to the following rules:

$$w_0(1) = w_0(0) + 1 \cdot y_i$$
$$w_1(1) = w_1(0) + x_1 \cdot y_i$$
$$w_2(1) = w_2(0) + x_2 \cdot y_i$$

   In matrix terms, we would write $w(1) = w(0) + y_i x_i$, or more generally we can write $w(t+1) = w(t) + y_i x_i$. Each time we update, we will choose a misclassified point at random, so the vector $x_i = (1, x_1, x_2)$ for that point. (Note that we have a 1, much like in linear regression, for the weight on the intercept term.)

10. **Classify the Data.** Use the new weights that have been generated to classify the data according to our perceptron function.

11. **Generate the Confusion Matrix.** As before, create your confusion matrix and calculate your accuracy. If you have achieved 100% accuracy, you are done. Otherwise, return to step 9 above and update your weights. Note that you will repeat steps 9-11 until you have either reached 100% accuracy or given up. You should be able to create a loop through steps 9-11 that stops when no observations are misclassified.

    (In order to avoid an infinite loop error, I recommend also putting a hard stop at the 10,000th update. If this happens to you, you have probably not generated your initial data correctly.)

12. **Deploy the Model.** Output your approximated target function. This is the function you have settled on to predict future classifications. As a test, generate a few (let's say 30) new observations and their true classifications using the line you chose in step 2. Apply your approximated target function and predict the classifications for these new observations. How do you do? (You may not get 100% accuracy, but you should do reasonably well.)

This, in a nutshell, is how to create a simple perceptron. You will turn in two files for this assignment. Your work in completing the above steps will be the first file. It need not be pretty, just turn in what you end up with.

The second file will be the deliverable below. Read on.

Deliverable

Your task is to write a program that takes an input data set, applies the perceptron, and outputs the final set of weights as well as a final confusion matrix. More specifically, your program should:

1. Accept a data frame of training data with three columns as above: two input columns of continuous numbers and a classification column of either $+1$ or -1 of true classifications.
2. Run our perceptron learning algorithm as above. Take the work you did above, and polish it into a clean, streamlined process to determine the final weights.
3. If your program can reach 100% accuracy on the training data in a reasonable time, it should output a message that includes the chosen weights. (Don't forget the bias term!)
4. If your program cannot reach 100% accuracy in a reasonable time (let's say 10,000 iterations), it should end gracefully and output an error message that says something like "We were unable to converge to a set of weights that correctly classifies the data. These data may not be linearly separable."

I will have a few test data sets ready to go, some that are linearly separable and some that are not. When you submit your program, I will run it against these test data sets to make sure each is handled correctly.

This deliverable program should be submitted as a separate program. Please give your two programs meaningful names so that I know which is your deliverable and which is your "lab" work above.

Your grade will be based primarily on your deliverable code, though I may use your lab work to gain insight into what you are doing (in other words to award partial credit!).