Homework 1: Perceptron

Due: 9/16/2021

1. *Exploratory data analysis.*

   (a) Pick a dataset you are interested in exploring.
   We have listed some datasets on the course project page that you might consider:

   `https://people.orie.cornell.edu/mru8/orie4741/projects.html`

   but you are free to find your own dataset in a field that interests you. We encourage you to pick a dataset you might want to use for your project.

   (b) Pose a question about the dataset.

   (c) Create four visualizations of the data that help answer your question. For each visualization, describe briefly what conclusions you drew from it. (Remember to label your axes!)

   For this problem, you may use any programming language. You do not need to submit your code; just submit your visualizations. If you use Python, you might follow the syntax in the eda demo

   `https://github.com/ORIE4741/demos/blob/master/eda.ipynb`

   or in the first recitation section to get started:

2. *Vectors, matrices, and inner products.*

   What is a vector? Is it just a matrix with one column, or is it something different? This problem explores this question using Python. You will find Python distinguishes between vectors (one-dimensional arrays) and matrices (two-dimensional arrays), even when the matrix has only one column. If you'd like, you may try out (translations of) these code examples in any other programming language for full credit. This code uses a few functions from the `numpy` package, which you can load with the command

   ```
   import numpy as np
   from numpy.random import rand
   ```

(a) The inner product, or dot product, of two vectors is the sum of the products of their elements. Here, let's use the dot function `dot` to compute inner products. Compare the following code

```
u = rand(3,1)
v = rand(3,1)
u.dot(v)
```

with

```
x = rand(3)
y = rand(3)
x.dot(y)
```

Do you get any errors? If they both work, are the shapes of the results the same or different? You can use the `shape` attribute of the array to check. What rule do you think numpy is using?

(b) There are many other ways to compute inner products. For example, try the following:

```
sum(u*v)
  u.T @ v
```

What results do these functions have when applied to u and v, and to x and y? If they both work, are the shapes of the results the same or different? What rule do you think numpy is using?

(c) Generate a random matrix (2d array) `A = rand(5,5)`. Consider the first column `A[:,0]`, and the first row `A[0,:]`. What shape do these have? Are they vectors or matrices?

(d) How would you take the inner product between the first column `A[:,0]` and the first row `A[0,:]`? State at least two methods. Your result should be a number, not an array.

(e) Let's check the rules for linear algebra with pandas dataframes. Load pandas and form an array:

```
import pandas as pd
X = pd.DataFrame(rand(5,3))
X.columns = ["feature 1", "feature 2", "feature 3"]
```

Consider the first column `X.iloc[:,0]`, and the first row `X.iloc[0,:]`. What shape do these have? Are they vectors or matrices? Do you notice any other attributes?

(f) Try making a vector three ways:

```
w_array = randn(3)
w_pd = pd.Series(w_array)
w_id = pd.Series(w_array, index=X.columns)
```

Which of these objects support the dot product `X.dot(w)`? If you see an error, what do you think went wrong?

3. *Perceptron.*

(a) Code the perceptron algorithm by filling in the appropriate portion of the notebook `perceptron.ipynb` found at

`https://github.com/ORIE4741/homework`

For the purposes of this problem, let's distinguish between iterations and checks: your code will perform many *checks* to see whether a datapoint (x, y) is misclassified; if it is, your code will update w, thereby completing an *iteration*. Your code will perform many checks in each iteration, particularly as the iteration nears convergence. Feel free to increase the maximum number of iterations (and checks) to ensure your code converges.

You are also welcome to use any other language (that your TAs can read) to solve the problem; you'll just need to rewrite the starter code from the notebook in your favorite language.

When most of the data points are misclassified, it's silly to check them *all* before making an update. Hence a good solution to this problem will not check whether *every* point is correctly classified at *every* iteration. If your code is written efficiently, you'll find it does much less than nd flops of work per iteration when many points are misclassified, but might need about nd flops of work per iteration when most points are correctly classified.

(b) How many iterations are required for the algorithm to converge? Plot the datapoints, the true vector `wtrue`, and the final hypothesis of the Perceptron algorithm. You can use the plotting function we've provided: `plot_perceptron(X,y,w)`.

(c) Repeat (b) with a randomly generated data set of size 20, 100, and 1000. Compare your results with (b).

(d) Randomly generate a linearly separable data set of size 1000 with $x_i \in \mathbb{R}^{10}$ instead of $\mathbb{R}^2$. How many iterations does the algorithm take to converge?

(e) Repeat the algorithm on the same data set as (d) for 100 experiments. In this part of the problem, pick your next point $x(t)$ to check randomly instead of deterministically. Plot a histogram for the number of iterations that the algorithm takes to converge.

(f) Summarize your conclusions with respect to accuracy and running time as a function of n (size of data set) and d (dimension).

(g) Go back to $n = 50$ and $d = 2$. Add an outlier to your dataset. What happens when we run the perceptron algorithm?

(h) How could we fix the perceptron algorithm? Try out at least one idea and report your results. *Note: this question has no right answer.*

You should submit a PDF of the notebook, which you can generate by printing the notebook to PDF from your browser. Make sure to clearly mark each problem section.

4. *Calibration.* How long did you spend on each problem in this homework assignment, and on the homework assignment, in total?