CS2810 Mathematics of Data Models	Gold
Spring 2019	January 16, 2019

Homework 02

Due: January 28, 9PM
Point total: 60

Instructions:

- Submit your PDF and .py file to Blackboard by the due date and time. Please do not zip your files together, as this interferes with Blackboard's preview functionality. Always show all your work, and for full credit, you must use the method that the problem instructs you to use (unless none is mentioned). Handwritten or typeset solutions are both acceptable, but unreadable submissions will be penalized. You may discuss problems with other students, but you may not write up solutions together, copy solutions from a common whiteboard, or otherwise share your written work or code. Do not use code or language that is copied from the Internet or other students; attribute the ideas *and* rephrase in your own words.

Problem 1 (12 points, 4 each)

In each case, find the matrix that performs the transformation described. (Note that you are looking for a *single* matrix, which you can find by multiplying out matrices that perform each individual operation. Be careful about which matrix goes first or last.) You can assume the direction of rotation is the one that is most convenient for our formulas.

-1 per incorrect matrix going into the multiplication, -1 per incorrect matrix multiplication besides ordering, -2 for an incorrect matrix ordering, to a minimum of 0 points per part.

i. In two dimensions: Swap the x and y coordinates, then rotate 30 degrees about the origin.

Solution: $\begin{bmatrix} \sqrt{3}/2 & -1/2 \\ 1/2 & \sqrt{3}/2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -1/2 & \sqrt{3}/2 \\ \sqrt{3}/2 & 1/2 \end{bmatrix}$

ii. In three dimensions: double the length, then rotate 45 degrees around the x axis, then 45 degrees around the z axis

Solution: $\begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 & 0 \\ \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sqrt{2}/2 & -\sqrt{2}/2 \\ 0 & \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} \sqrt{2}/2 & -1/2 & 1/2 \\ \sqrt{2}/2 & 1/2 & -1/2 \\ 0 & \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} \sqrt{2} & -1 & 1 \\ \sqrt{2} & 1 & -1 \\ 0 & \sqrt{2} & \sqrt{2} \end{bmatrix}$

iii. In four dimensions: Project down to three dimensions (xyz) by removing the w coordinate (and producing a three-dimensional vector), then rotate 60 degrees around the y-axis. (For the projection matrix, notice that the matrix needs to "shift" the elements to occupy new places in the smaller vector, since w came first but is now dropped.)

Solution: $\begin{bmatrix} 1/2 & 0 & \sqrt{3}/2 \\ 0 & 1 & 0 \\ -\sqrt{3}/2 & 0 & 1/2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1/2 & 0 & \sqrt{3}/2 \\ 0 & 0 & 1 & 0 \\ 0 & -\sqrt{3}/2 & 0 & 1/2 \end{bmatrix}$

Problem 2 (10 points)

In each subproblem, you are given three vectors, $\vec{q}, \vec{r}$, and $\vec{s}$. Determine whether the three vectors are linearly independent. If they are not, find the dimension of their span.

2 points for each correct conclusion (dependent or not), 2 points for correct span dimension when they aren't dependent.

i. $\vec{q} = \begin{bmatrix}1\\2\\3\end{bmatrix}, \vec{r} = \begin{bmatrix}3\\2\\1\end{bmatrix}, \vec{s} = \begin{bmatrix}5\\2\\-1\end{bmatrix}$

Solution: Not linearly independent; dimension of the span is 2 (verifiable with Gauss-Jordan).

ii. $\vec{q} = \begin{bmatrix}1\\2\\3\end{bmatrix}, \vec{r} = \begin{bmatrix}1\\2\\4\end{bmatrix}, \vec{s} = \begin{bmatrix}1\\2\\5\end{bmatrix}$

Solution: Not linearly independent; dimension of the span is 2 (again verifiable with Gauss-Jordan). While these might seem more redundant, as long as one vector isn't a multiple of another, there are still at least two dimensions in the span.

iii. $\vec{q} = \begin{bmatrix}\pi\\1\\1\end{bmatrix}, \vec{r} = \begin{bmatrix}-1\\\pi\\0\end{bmatrix}, \vec{s} = \begin{bmatrix}-1\\1\\0\end{bmatrix}$

Solution: Linearly independent (provable with Gauss-Jordan elimination).

Problem 3 (8 pts, 4 each)

i. Find an example of four 4-dimensional vectors such that any three of them are linearly independent, but all four together are not. (Hint: You can do this with vectors of just zeros and ones.) Explain how you know how all four are linearly dependent, and explain how you know that any three of them are linearly independent.

Solution: An example would be the vectors $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\\0\end{bmatrix}$. All four together are clearly dependent, because the fourth is the sum of the first three. The first three alone are independent because they're standard basis vectors (and clearly none is the sum of the others). Any set of three that includes the last vector must also be linearly independent, because the two standard basis vectors are definitely independent from each other, and the last vector always provides a component that neither of the other two can provide.

1 point for a correct set of vectors, 1 point for the argument that they're dependent, 1 point for an argument that at least one set is independent, 1 point for arguments supporting all of the other sets as independent.

ii. A student has proposed a strange variant of a perceptron in which the weights are always altered by $\alpha\vec{1}$ or $-\alpha\vec{1}$, where $\vec{1}$ is the all ones vector and α is a small constant. (This is instead

of using the input $\vec{x}$ to calculate the adjustment to the weights.) What is the dimension of the span of these vectors? How does the dimension of the span lead us to conclude that this perceptron can't possibly learn all possible decision boundaries for n-dimensional input?

Solution: The span has dimension 1, since one is just a constant multiplied by the other. Since the full space of possible perceptron boundaries has $n + 1$ dimensions (counting the bias), this can't possibly represent all possible decision boundaries.

2 points for the correct span, 2 points for an answer that relates the span to the set of all decision boundaries. -1 for an answer that seems to appreciate that not all hypotheses are possible, but doesn't relate this back to the span idea in any way.

Problem 4 (9 points, 3 each)

Try going through the perceptron tutorial (`perceptron_fish_tutorial.ipynb`) before answering these questions.

i. In the provided demo code (with 20 epochs), how many total times is a perceptron classification compared to a true classification?

Solution: Since there are 20 epochs, 100 fish per group, and 2 groups, that's 4000 classifications compared to true classifications.

-1 for neglecting the fact that there are 2 groups, getting an answer half as big. No credit for other incorrect answers.

ii. Why do we see no decision boundary after the first epoch (and before the second)? Justify your explanation with an equation for the decision boundary in $y = mx + b$ form (round to two digits after the decimal place).

Solution: We can't see the line because it's outside the viewbox. Specifically, the decision boundary is $1 + 7.17x + 2.53y = 0$, and rearranging those terms gives $y = -2.83x - 0.40$. That's a negative y-intercept and a negative slope.

1 point for the basic explanation, 1 point for more or less understanding what equation to manipulate to get the decision boundary, 1 point for a correct y = mx + b equation.

iii. If a perceptron has no bias term (weight for a constant input), is it possible that an incorrect classification of a point could result in no update to the weights? If so, describe what input(s) lead to that situation. If not, explain why not.

Solution: Yes, with no bias term, the zero vector will have this effect. Any other vector will have a nonzero dot product with itself, and cause an update.

1 point for knowing it's possible, 2 points for identifying the zero vector.

Problem 5 (21 points)

This problem is contained in `hw2_perceptron_transform.ipynb`. Please submit your work as a .py file along with a PDF for the other problems. (Don't submit a whole Python notebook, please.)

Unfortunately, we accidentally posted an old version of this programming assignment for a little while. The ALTERNATE instructions give how to grade this version when it's different.

Solution:

Part A: w stops changing because eventually, all points are classified correctly, so no point trigger a change. `x_easy` allows this to happen because the points are linearly separable; they can be separated by a linear surface.

The key idea here is that the points are linearly separable in this example; the first question is worded in such a way that this is arguably good enough for both parts. An answer that has some good insight but doesn't realize that this is why the learning stops is 2/4.

ALTERNATE assignment: In the .py file that was posted accidentally, this was replaced with a norm-based stopping criterion that the students had to program. This is worth 4 points, just like the first thought question. (And not 5, as this early draft of the assignment suggested.) Give 4 points for something that works perfectly, 3 for something that has a subtle bug, 2 points for a major bug, 1 point for effort.

Solution:

Part B:

```
def stop_angle(w, w_old):
    if (np.linalg.norm(w) == 0 or np.linalg.norm(w_old) == 0):
            return False
    cosTheta = np.dot(w, w_old)/(np.linalg.norm(w)*np.linalg.norm(w_old))
    if np.arccos(cosTheta) < 0.05:
        return True
    return False
```

4 points for working perfectly, 3 for subtle bug, 2 for incorrect formula but good attempt, 1 for effort. Note that a major thing that could be missing is a check for a zero vector. Deduct a point if the weights start at zero and they don't handle this case. But if they have a different way around it, like initializing the weights to be nonzero, just make a -0 note that they should handle a zero vector somehow.

Solution: Part C: A threshold that is too low (a small number of radians) will cause it to never stop the learning (unless something else stops it first, such as correctly classifying everything or reaching a maximum number of epochs). Choosing a threshold that is too high will cause the learning to stop too early, before the classifier works well.

2 points for each correctly identified effect, with -1 if it's phrased poorly or doesn't quite nail the idea. -1 total if they're not clear which problem they're talking about when they list the effects.

ALTERNATE: The alternate version of the assignment here has a question about which stopping criterion is better. Unfortunately, neither looks great in this version of the assignment; one runs forever, and the other stops really early. So give 4 points to any reasonable argument.

Solution: Part D:

```
A = np.array([[1, 0, 0], [0, 2, 0], [0, 0, 100]])
B = np.array([[np.sqrt(2)/2, -np.sqrt(2)/2],[np.sqrt(2)/2,np.sqrt(2)/2]])
```

2 points per matrix.

Solution: Part E: The perceptron focuses on only one feature at a time because the adjustments to the weights scale with the input size – so most of the change in the vector is being driven by the features that have a large magnitude, while the small features only change the boundary a little.

Full credit for correctly identifying the problem as stemming from how perceptron learning works, adjusting weights proportionally to the size of the feature. -2 if the student says "it's focusing on the big features" but doesn't explain why that would matter. Minimum 1 point for effort.

Solution: Part F: We could scale the data, dividing each feature by its maximum value, before handing it to the perceptron. This will ensure that changes will be proportional to how big a feature is relative to its usual scale, and not how big the feature is in an absolute sense.

Answers that are totally reasonable and would work well can get full extra credit even if they aren't this one. But if an idea has some merit but also fatal flaws – one student suggested to me just handing the perceptron the ratio of two features as a single feature, for example – then it's probably just 1 point, or 2 if the flaw is quite subtle.