Homework 3
KNN, Perceptron, Linear Regression[1]

CMU 10-601: Machine Learning (Fall 2018)

piazza.com/cmu/fall2018/10601bd
OUT: Thursday, Sep 20th, 2018

DUE: Friday, Sept 28th, 2018, 11:59pm
TAs: Brynn Edmunds, Zhuoran Liu, Emilio Arroyo-Fang, George Xu

START HERE: Instructions

Homework 3 covers topics on KNN, perceptron, and linear regression. The homework includes multiple choice, True/False, and short answer questions.

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section on the course site for more information: http://www.cs.cmu.edu/~mgormley/courses/10601bd-f18/about.html#7-academic-integrity-policies
- **Late Submission Policy:** See the late submission policy here: http://www.cs.cmu.edu/~mgormley/courses/10601bd-f18/about.html#6-general-policies
- **Submitting your work:**
  - **Gradescope:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using Gradescope (https://gradescope.com/). Please use the provided template. Submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Alternatively, submissions can be written in LaTeX. Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. Each derivation/proof should be completed on a separate page. For short answer questions, you **should not** include your work in your solution. If you include your work in your solutions, your assignment may not be graded

[1]Compiled on Thursday 20th September, 2018 at 09:27

correctly by our AI assisted grader. In addition, please tag the problems to the corresponding pages when submitting your work.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For LaTeXusers, use ■ and ●for shaded boxes and circles, and don't change anything else.

Instructions for Specific Problem Types

For "Select One" questions, please fill in the appropriate bubble completely:

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie

○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie

⊗ Noam Chomsky

For "Select all that apply" questions, please fill in all appropriate squares completely:

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

☐ I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

⊠ I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

Fill in the blank: What is the course number?

10-601

10-~~7~~601

1 K-Nearest Neighbors [30 pts]

1. **[3pt]** Consider the description of two objects below:

	Object A	**Object B**
Feature 1	3	9.1
Feature 2	2.1	0.7
Feature 3	4.8	2.2
Feature 4	5.1	5.1
Feature 5	6.2	1.8

We can reason about these objects as points in high dimensional space.

Consider the two different distance functions below. Under which scheme are they closer in 5-D space?

(a) Euclidean Distance: $d(x, y) = \sqrt{\sum_{i-1}^{n}(x_i - y_i)^2}$ = 8.08

(b) Manhattan Distance: $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$ 14.5

Select one:

● Euclidean Distance

○ Manhattan Distance

Training error here is the error you'll have when you input your training set to your KNN as test set.

.

When K = 1, you'll choose the closest training sample to your test sample. Since your test sample is in the training dataset, it'll choose itself as the closest and never make mistake.

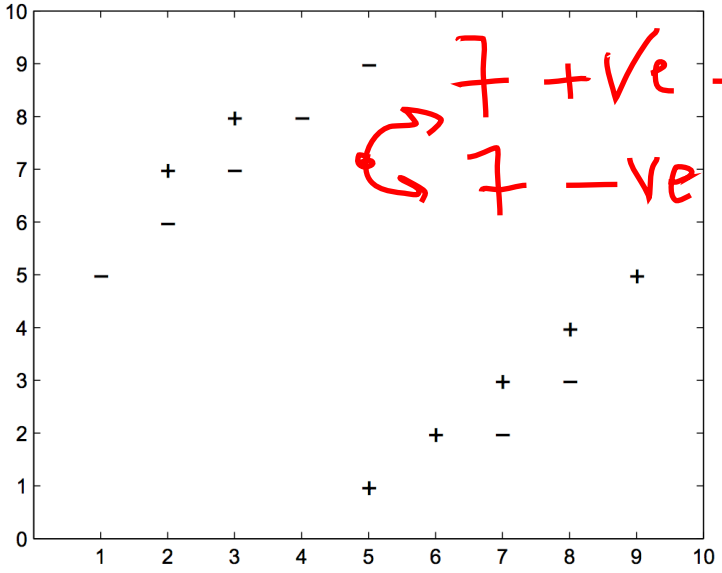
.

For this reason, the training error will be zero when K = 1, irrespective of the dataset.

.

There is one logical assumption here by the way, and that is your training set will not include same training samples belonging to different classes, i.e. conflicting information. Some real world datasets might have this property though.

https://stats.stackexchange.com/questions/367010/training-error-in-knn-classifier-when-k-1/367015



7 +ve → 2 misses

7 −ve → 2 misses

$\frac{2}{14} + \frac{2}{14} = \frac{2}{7}$

error on training

2. [**3pt**] Consider a k-nearest neighbors binary classifier which assigns the class of a test point to be the class of the majority of the k-nearest neighbors, according to a Euclidean distance metric. Using the data set shown above to train the classifier and choosing $k = 5$, which is the classification error on the training set? Assume that a point can be its own neighbor.

Answer as a decimal with precision 4, e.g. (6.051, 0.1230, 1.234e+7)

0.2857

3. [**3pt**] In the data set shown above, what is the value of k that minimizes the training error? Note that a point can be its own neighbor.

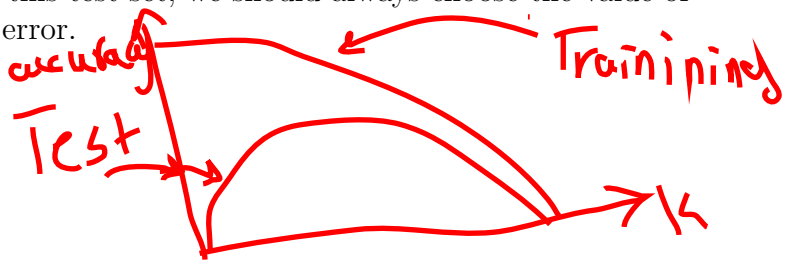
1

4. [**3pt**] Assume we have a training set and a test set drawn from the same distribution, and we would like to classify points in the test set using a k-NN classifier. In order to minimize the classification error on this test set, we should always choose the value of k which minimizes the training set error.

Select one:

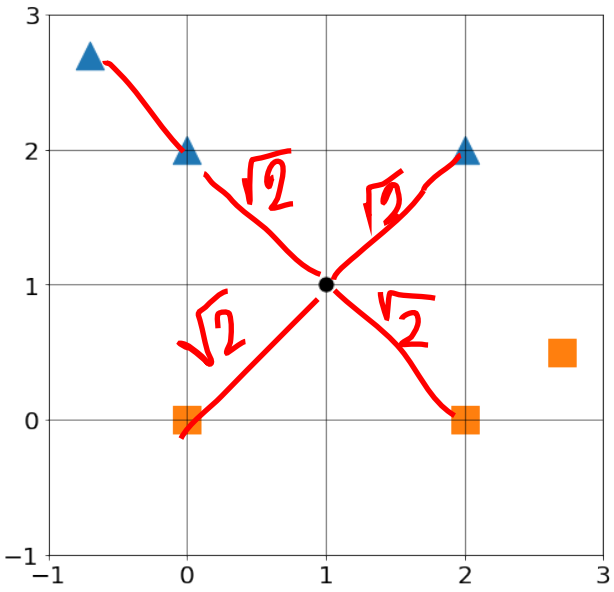
○ True

● False



5. **[3pt]** Consider a binary k-NN classifier where $k = 4$ and the two labels are "triangle" and "square".

   Consider classifying a new point $\mathbf{x} = (1, 1)$, where two of the $\mathbf{x}$'s nearest neighbors are labeled "triangle" and two are labeled "square" as shown below.



   Which of the following methods can be used to break ties or avoid ties on this dataset?

   (a) Assign x the label of its nearest neighbor

   (b) Flip a coin to randomly assign a label to $\mathbf{x}$ (from the labels of its 4 closest points)

   (c) Use $k = 3$ instead

   (d) Use $k = 5$ instead

   Select one:

   ○ a only

   ○ b only

   ○ b,c,d

   ● b,d

   ○ d only

   ○ a,b,c,d

   ○ None of the above

https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955?gi=54b88e70365a

6. [**3pt**] Consider the following data concerning the relationship between academic performance and salary after graduation. High school GPA and university GPA are two numerical variables (predictors) and salary is the numerical target. Note that salary is measured in thousands of dollars per year.

Student ID	High School GPA	University GPA	Salary
1	2.2	3.4	45
2	3.9	2.9	55
3	3.7	3.6	91
4	4.0	4.0	142
5	2.8	3.5	88
6	3.5	1.0	2600
7	3.8	4.0	163
8	3.1	2.5	67
9	3.5	3.6	unknown

Among Students 1 to 8, who is the nearest neighbor to Student 9, using Euclidean distance?

Answer the Student ID only.

3

7. [**3pt**] In the data set shown above, our task is to predict the salary Student 9 earns after graduation. We apply k-NN to this regression problem: the prediction for the numerical target (salary in this example) is equal to the average of salaries for the top k nearest neighbors.

If $k = 3$, what is our prediction for Student 9's salary?

Round your answer to the nearest integer. Be sure to use the same unit of measure (thousands of dollars per year) as the table above.

132

```python
l = [[2.2, 3.4],
     [3.9, 2.9],
     [3.7, 3.6],
     [4.0, 4.0],
     [2.8, 3.5],
     [3.5, 1.0],
     [3.8, 4.0],
     [3.1, 2.5],
     [3.5, 3.6]]

labels = [45, 55, 91, 142, 88, 2600, 163, 67]
i = 1;
L2_Norm = {}

# Calculate the distances
for l_li in l:
  print("Euclidean distance with student ID: ", i, " is: ",np.sqrt(np.sum((np.square(np.array(l[8])- np.array(l_li))))))
  L2_Norm[i] = np.sqrt(np.sum((np.square(np.array(l[8])- np.array(l_li)))))
  i += 1

ordered = {k: v for k, v in sorted(L2_Norm.items(), key=lambda item: item[1])}
index_distance = list(ordered.items())
print(index_distance)
print(index_distance[1], " Is the closest neighbor")

k = 3
expected_salary = 0
# Take average of K-NNs
for i in range(1, k+1):
  expected_salary += labels[index_distance[i][0]-1]
print("Expected salary = ", expected_salary /k)
```

```
Euclidean distance with student ID:  1  is:  1.3152946437965904
Euclidean distance with student ID:  2  is:  0.806225774829855
Euclidean distance with student ID:  3  is:  0.20000000000000018
Euclidean distance with student ID:  4  is:  0.6403124237432848
Euclidean distance with student ID:  5  is:  0.7071067811865478
Euclidean distance with student ID:  6  is:  2.6
Euclidean distance with student ID:  7  is:  0.49999999999999983
Euclidean distance with student ID:  8  is:  1.1704699910719625
Euclidean distance with student ID:  9  is:  0.0
[(9, 0.0), (3, 0.20000000000000018), (7, 0.49999999999999983), (4, 0.6403124237432848), (5, 0.7071067811865478), (2, 0.806225774829855),
(3, 0.20000000000000018)  Is the closest neighbor
Expected salary =  132.0
```

8. **[3pt]** Suppose that the first 8 students shown above are only a subset of your full training data set, which consists of 10,000 students. We apply KNN regression using Euclidean distance to this problem and we define training loss on this full data set to be the mean squared error (MSE) of salary.

   Now consider the possible consequences of modifying the data in various ways. Which of the following changes **could** have an effect on training loss on the full data set as measured by mean squared error (MSE) of salary? Select all that apply.

   Select all that apply:

   ■ Rescaling only "High School GPA" to be a percentage of 4.0

   ■ Rescaling only "University GPA" to be a percentage of 4.0

   ☑ Rescaling both "High School GPA" and "University GPA", so that each is a percentage of 4.0 (scale by the same percentage).

   ☐ None of the above.

9. **[3pt]** In this question, we would like to compare the differences among KNN, the perceptron algorithm, and linear regression. Please select all that apply in the following options.

   Select all that apply:

   ■ For classification tasks, both KNN and the perceptron algorithm can have linear decision boundaries.

   ☐ For classification tasks, both KNN and the perceptron algorithm always have linear decision boundaries.

   ☒ All three models can be susceptible to overfitting.

   ☒ In all three models, after the training is completed, we must store the training data to make predictions on the test data.

   ☒ None of the above.

A related issue with KNN is feature scale. Suppose that we are trying to classify whether some object is a ski or a snowboard (see Figure 3.5). We are given two features about this data: the width and height. As is standard in skiing, width is measured in millimeters and height is measured in centimeters. Since there are only two features, we can actually plot the entire training set; see Figure 3.6 where ski is the positive class. Based on this data, you might guess that a KNN classifier would do well.

Suppose, however, that our measurement of the width was computed in millimeters (instead of centimeters). This yields the data shown in Figure 3.7. Since the width values are now tiny, in comparison to the height values, a KNN classifier will effectively *ignore* the width values and classify almost purely based on height. The predicted class for the displayed test point had changed because of this feature scaling.

We will discuss feature scaling more in Chapter 5. For now, it is just important to keep in mind that KNN does not have the power to decide which features are important.

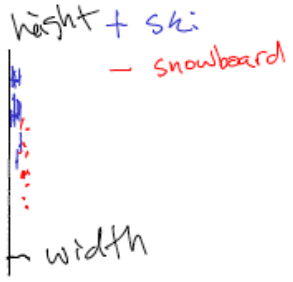


Figure 3.7: Classification data for ski vs snowboard in 2d, with width rescaled to mm.

3.3 Decision Boundaries

The standard way that we've been thinking about learning algorithms up to now is in the *query model*. Based on training data, you learn something. I then give you a query example and you have to guess it's label.

An alternative, less passive, way to think about a learned model is to ask: what sort of test examples will it classify as positive, and what sort will it classify as negative. In Figure 3.9, we have a set of training data. The background of the image is colored blue in regions that *would* be classified as positive (if a query were issued there) and colored red in regions that *would* be classified as negative. This coloring is based on a 1-nearest neighbor classifier.

In Figure 3.9, there is a solid line separating the positive regions from the negative regions. This line is called the **decision boundary** for this classifier. It is the line with positive land on one side and negative land on the other side.

Decision boundaries are useful ways to visualize the complexity of a learned model. Intuitively, a learned model with a decision boundary that is really jagged (like the coastline of Norway) is really complex and prone to overfitting. A learned model with a decision boundary that is really simple (like the bounary between Arizona and Utah) is potentially underfit.

Now that you know about decision boundaries, it is natural to ask: what do decision boundaries for decision trees look like? In order

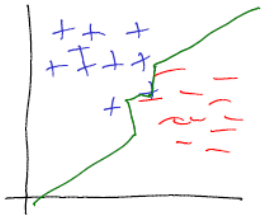


Figure 3.9: decision boundary for 1nn.

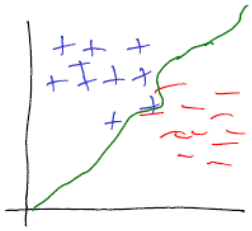


Figure 3.10: decision boundary for knn with k=3.

The first step in this algorithm is to compute distances from the test point to all training points (lines 2-4). The data points are then sorted according to distance. We then apply a clever trick of *summing* the class labels for each of the K nearest neighbors (lines 6-10) and using the SIGN of this sum as our prediction.

The big question, of course, is how to choose K. As we've seen, with $K = 1$, we run the risk of overfitting. On the other hand, if K is large (for instance, $K = N$), then KNN-PREDICT will always predict the majority class. Clearly that is underfitting. So, K is a hyperparameter of the KNN algorithm that allows us to trade-off between overfitting (small value of K) and underfitting (large value of K).

The only **hyperparameter** of the perceptron algorithm is *MaxIter*, the number of passes to make over the training data. If we make many many passes over the training data, then the algorithm is likely to overfit. (This would be like studying *too long* for an exam and just confusing yourself.) On the other hand, going over the data only one time might lead to underfitting. This is shown experimentally in Figure 4.3. The x-axis shows the number of passes over the data and the y-axis shows the training error and the test error. As you can see, there is a "sweet spot" at which test performance begins to degrade due to overfitting.

One aspect of the perceptron algorithm that is left underspecified is line 4, which says: loop over all the training examples. The natural implementation of this would be to loop over them in a constant order. The is actually a bad idea.

Consider what the perceptron algorithm would do on a data set that consisted of 500 positive examples followed by 500 negative examples. After seeing the first few positive examples (maybe five), it would likely decide that *every* example is positive, and would stop

? itive examples ($y = +1$). It should also hold for negative examples. Work it out.

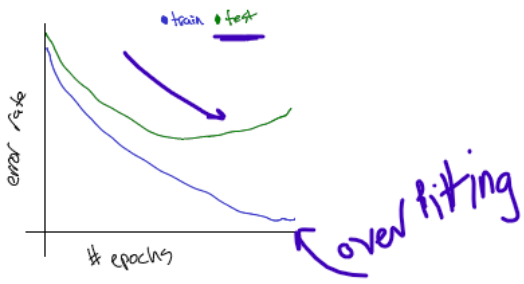


Figure 4.3: training and test error via early stopping

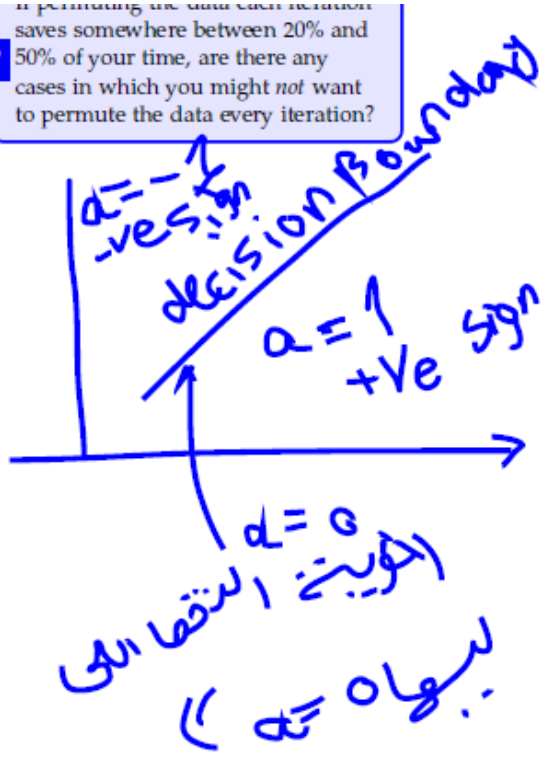
? saves somewhere between 20% and 50% of your time, are there any cases in which you might *not* want to permute the data every iteration?

4.3 Geometric Intrepretation

A question you should be asking yourself by now is: what does the decision boundary of a perceptron look like? You can actually answer that question mathematically. For a perceptron, the decision boundary is precisely where the sign of the activation, a, changes from -1 to $+1$. In other words, it is the set of points $\boldsymbol{x}$ that achieve *zero* activation. The points that are not clearly positive nor negative. For simplicity, we'll first consider the case where there is no "bias" term (or, equivalently, the bias is zero). Formally, the decision boundary $\mathcal{B}$ is:

$$\mathcal{B} = \left\{ \boldsymbol{x} : \sum_d w_d x_d = 0 \right\} \tag{4.7}$$

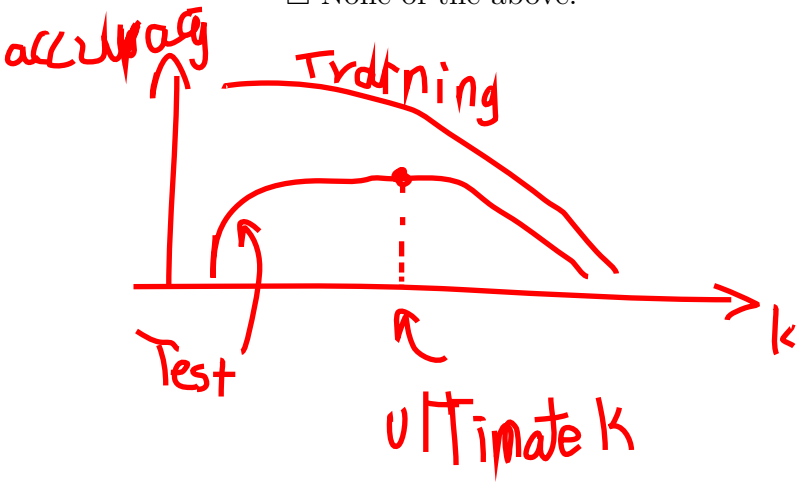
We can now apply some linear algebra. Recall that $\sum_d w_d x_d$ is just the dot product between the vector $\boldsymbol{w} = \langle w_1, w_2, \ldots, w_D \rangle$ and the vector $\boldsymbol{x}$. We will write this as $\boldsymbol{w} \cdot \boldsymbol{x}$. Two vectors have a zero dot product if and only if they are perpendicular. Thus, if we think of the weights as a vector $\boldsymbol{w}$, then the decision boundary is simply the plane perpendicular to $\boldsymbol{w}$.



10. [**3pt**] Please select all that apply about kNN in the following options.

Select all that apply:

- ☑ Large k gives a smoother decision boundary
- ☑ To reduce the impact of noise or outliers in our data, we should increase the value k.
- ☐ If we make k too large, we could end up overfitting the data.
- ☑ We can use cross-validation to help us select the value of k.
- ☐ We should never select the k that minimizes the error on the validation dataset.
- ☐ None of the above.



2 Perceptron [30 pts]

1. **[3pt]** Consider running the Perceptron algorithm on some sequence of examples S (an example is a data point and its label). Let S' be the same set of examples as S, but presented in a different order.

   True or False: the perceptron algorithm is guaranteed to make the same number of mistakes on S as it does on S'.

   Select one:

   ○ True

   ● False

2. **[3pt]** Suppose we have a perceptron whose inputs are 2-dimensional vectors and the vector component is either 0 or 1, i.e., $x_i \in \{0, 1\}$. The prediction function $y = \text{sign}(w_1 x_1 + w_2 x_2 + b)$, and

   $$\text{sign}(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$

   And Gate:
   00 0
   01 0
   10 0
   11 1
   Or Gate:
   00 0
   01 1
   10 1
   11 1
   Xor Gate:
   00 0
   01 1
   10 1
   11 0

   Which of the following functions can be implemented with the above perceptron? That is, for which of the following functions does there exist a set of parameters w, b that correctly define the function. Select all that apply.

   Select all that apply:

   ☑ AND function, i.e., the function that evaluates to 1 if and only if all inputs are 1, and 0 otherwise.

   ☑ OR function, i.e., the function that evaluates to 1 if and only if at least one of the inputs are 1, and 0 otherwise.

   ☒ XOR function, i.e., the function that evaluates to 1 if and only if the inputs are not all the same. For example

   $$\text{XOR}(1, 0) = 1, \text{ but } \text{XOR}(1, 1) = 0.$$

   ☐ None of the above.

The second question is: how long does it take to converge? By "how long," what we really mean is "how many updates?" As is the case for much learning theory, you will not be able to get an answer of the form "it will converge after 5293 updates." This is asking too much. The sort of answer we can hope to get is of the form "it will converge after *at most* 5293 updates."

المده اللي هيحصل فيها الكونفيرجنس اللي هو ان البارمترز تبطّل تأبديت .. ده سؤال مالوش اجابه صريحه ... انما الاجابه في الحاله ديه بنقا at mot .. يعني انا عندي ماكسيمم ليميت علي المده .. انما كرقم صريح فالاجابه مش موجوده

its decision boundaries can *only* be linear. The classic way of showing this limitation is through the XOR problem (XOR = exclusive or). The XOR problem is shown graphically in Figure 4.12. It consists of four data points, each at a corner of the unit square. The labels for these points are the same, along the diagonals. You can try, but you will not be able to find a linear decision boundary that perfectly separates these data points.

One question you might ask is: do XOR-like problems exist in the real world? Unfortunately for the perceptron, the answer is yes. Consider a sentiment classification problem that has three features that simply say whether a given word is contained in a review of a course. These features are: EXCELLENT, TERRIBLE and NOT. The EXCELLENT feature is indicative of positive reviews and the TERRIBLE feature is indicative of negative reviews. But in the presence of the NOT feature, this categorization flips.

One way to address this problem is by adding feature combinations. We could add two additional features: EXCELLENT-AND-NOT and TERRIBLE-AND-NOT that indicate a conjunction of these base features. By assigning weights as follows, you can achieve the desired effect:

$$w_{\text{EXECELLENT}} = +1 \qquad w_{\text{TERRIBLE}} = -1 \qquad w_{\text{NOT}} = 0$$
$$w_{\text{EXECLLENT-AND-NOT}} = -2 \qquad w_{\text{TERRIBLE-AND-NOT}} = +2$$

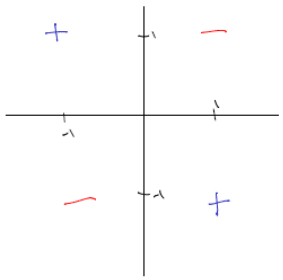


Figure 4.12: picture of xor problem

What you might expect to see is that the perceptron will converge more quickly for easy learning problems than for hard learning problems. This certainly fits intuition. The question is how to *define* "easy" and "hard" in a meaningful way. One way to make this definition is through the notion of margin. If I give you a data set and hyperplane that separates itthen the *margin* is the distance between the hyperplane and the nearest point. Intuitively, problems with large margins should be easy (there's lots of "wiggle room" to find a separating hyperplane); and problems with small margins should be hard (you really have to get a very specific well tuned weight vector).

البرسيبترون هيكونفيرج أسرع للمشاكل السهله أكيد .. بس ازاي هنعرف المشكله هل هي سهله ولا صعبه .. فعملو تعريف اسمو المارجن .. فبيقولك لو حد اداك الداتاسيت و الهايبر بلين اللي بيفصل ما بين الموجب و السالب ... في الحاله ديه المسافه ما بين الهايبر بلين اللي اداهولك و اقرب نقطه منو .. هي ديه المارجن .. كل ما كانت المارجن كبيره .. كل ما كان عندك مساحه أكتر و منطقه اكتر تقدر تلاقي فيها هايبر بلين بيفصل ما بين الموجب والسالب فبالتالي كل ما كانت المشكله سهله انك تحلها .. انما المارجن الضيقه اوي ديه . بتبقا صعبه و بتحتاج تعمل تيوننج للويت فيكتور

Formally, given a data set D, a weight vector $\boldsymbol{w}$ and bias b, the margin of $\boldsymbol{w}, b$ on D is defined as:

$$\text{margin}(\mathbf{D}, \boldsymbol{w}, b) = \begin{cases} \min_{(\boldsymbol{x},y)\in\mathbf{D}} y(\boldsymbol{w}\cdot\boldsymbol{x}+b) & \text{if } \boldsymbol{w} \text{ separates D} \\ -\infty & \text{otherwise} \end{cases} \tag{4.8}$$

In words, the margin is only defined if $\boldsymbol{w}, b$ actually separate the data (otherwise it is just $-\infty$). In the case that it separates the data, we find the point with the minimum activation, after the activation is multiplied by the label.

For some historical reason (that is unknown to the author), margins are always denoted by the Greek letter γ (gamma). One often

? So long as the margin is not $-\infty$, it is always positive. Geometrically this makes sense, but why does Eq (4.8) yield this?

Taking the left-most and right-most terms, we get that $\sqrt{k} \geq k\gamma$. Dividing both sides by k, we get $\frac{1}{\sqrt{k}} \geq \gamma$ and therefore $k \leq \frac{1}{\gamma^2}$. This means that once we've made $\frac{1}{\gamma^2}$ updates, we cannot make any more! □

3. **[3pt]** Suppose we have a dataset $\left\{\left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(N)}, y^{(N)}\right)\right\}$, where $\mathbf{x}^{(i)} \in \mathbb{R}^M$, $y^{(i)} \in \{+1, -1\}$. We would like to apply the perceptron algorithm on this dataset. Assume there is no bias term. How many parameter values is the perceptron algorithm learning?

Select one:

- ○ N
- ○ $N \times M$
- ● M

4. **[3pt]** Which of the following are true about the perceptron algorithm? Select all that apply.

Select all that apply:

- ☒ The number of mistakes the perceptron makes is proportional to the size of the dataset.
- ☒ The perceptorn algorithm converges on any dataset.
- ■ Perceptron algorithm can be used in the context of online learning.
- ☒ For linearly spearable data, the perceptron algorithm finds the separating hyperplane with the largest margin.
- ☐ None of the above.

5. **[3pt]** Suppose we have the following data:

$$\mathbf{x}^{(1)} = [1, 2] \quad \mathbf{x}^{(2)} = [-1, 2] \quad \mathbf{x}^{(3)} = [-2, 3] \quad \mathbf{x}^{(4)} = [1, -1]$$
$$y^{(1)} = 1 \quad y^{(2)} = -1 \quad y^{(3)} = -1 \quad y^{(4)} = 1$$

Starting from $\mathbf{w} = [0, 0]$, what is the vector $\mathbf{w}$ after running the perceptron algorithm with exactly one pass over the data? Assume we are running the perceptron algorithm without a bias term. If the value of the dot product of a data point and the weight vector is 0, the algorithm makes the prediction 1.

Select one:

- ● $[1, -2]$
- ○ $[2, 0]$
- ○ $[-1, 1]$
- ○ $[1, -3]$

6. **[3pt]** Please refer to previous question for the data. Assume we are running perceptron in the batch setting. How many passes will the perceptron algorithm make before converging to a perfect classifier, i.e., one that does not make false prediction on this dataset?

Select one:

- ● 2
- ○ 3
- ○ 5
- ○ Infinitely many (the algorithm does not converge)

7. **[3pt]** We can view the perceptron algorithm as trying to minimize which of the following objective functions with stochastic gradient descent? Assume that we apply the notation where $x_0 = 1$. θ_0 is the bias term, and N is the number of data points. Note the function

$$f(x) = (x)_+ = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Select one:

- ○ $J(\boldsymbol{\theta}) = \sum_{i=1}^{N} -y^{(i)} \left(\boldsymbol{\theta} \cdot \mathbf{x}^{(i)}\right)$
- ○ $J(\boldsymbol{\theta}) = \sum_{i=1}^{N} y^{(i)} \left(\boldsymbol{\theta} \cdot \mathbf{x}^{(i)}\right)$
- ● $J(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left(-y^{(i)} \left(\boldsymbol{\theta} \cdot \mathbf{x}^{(i)}\right)\right)_+$
- ○ $J(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left(y^{(i)} \left(\boldsymbol{\theta} \cdot \mathbf{x}^{(i)}\right)\right)_+$

8. **[3pt]** Continuing with the above question, what is the gradient of the correct loss function when the current data we are seeing is $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$?

Select one:

- ● $\begin{cases} -y^{(i)}\mathbf{x}^{(i)}, & \text{if } -y^{(i)} \left(\theta \cdot \mathbf{x}^{(i)}\right) \geq 0 \\ 0, & \text{otherwise.} \end{cases}$
- ○ $-y^{(i)}\mathbf{x}^{(i)}$
- ○ $y^{(i)}\mathbf{x}^{(i)}$
- ○ $\begin{cases} y^{(i)}\mathbf{x}^{(i)}, & \text{if } -y^{(i)} \left(\theta \cdot \mathbf{x}^{(i)}\right) \geq 0 \\ 0, & \text{otherwise.} \end{cases}$

$w \leftarrow w + y^{(i)} x^{(i)}$

$\theta \leftarrow \theta - \gamma g$

$\therefore g = -y^{(i)} x^{(i)}$

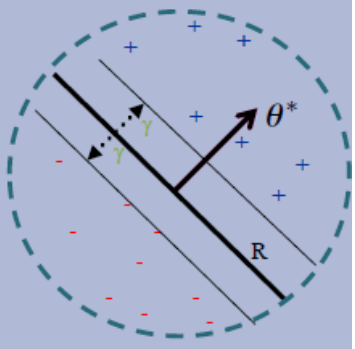
Algorithm 1 Perceptron Learning Algorithm (Online)

procedure PERCEPTRON($\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots\}$)
  $\theta \leftarrow 0, k = 1$ ▷ Initialize parameters
  for $i \in \{1, 2, \ldots\}$ **do** ▷ For each example
    if $y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)}) \leq 0$ **then** ▷ If mistake ←
      $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + y^{(i)}\mathbf{x}^{(i)}$ ▷ Update parameters
      $k \leftarrow k + 1$
  return θ

احنا بنحاول نقلل ال Mistakes

Theorem 0.1 (Block (1962), Novikoff (1962)).
Given dataset: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$.
Suppose:

1. *Finite size inputs:* $||x^{(i)}|| \leq R$
2. *Linearly separable data:* $\exists \boldsymbol{\theta}^*$ *s.t.* $||\boldsymbol{\theta}^*|| = 1$ *and* $y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)}) \geq \gamma, \forall i$



Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$

9. **[3pt]** Please select the correct statement(s) about the mistake bound of the perceptron algorithm. Select all that apply.

Select all that apply:

☐ If the minimum distance from any data point to the separating hyperplane of the data is increased, the mistake bound will also increase.

■ If the maximum distance from any data point to the origin is increased, then the mistake bound will also increase.

■ If the maximum distance from any data point to the mean all data points is increased, then the mistake bound will also increase.

☐ The mistake bound is linearly inverse-proportional to the minimum distance of any data point to the separating hyperplane of the data.

☐ None of the above.

10. **[3pt]** Suppose we have data whose elements are of the form $[x_1, x_2]$, where $x_1 + x_2 = 0$. We do not know the label for each element. Suppose the perceptron algorithm starts with $\boldsymbol{\theta} = [1, 2]$, which of the following values will $\boldsymbol{\theta}$ never take on in the process of running the perceptron algorithm on the data?

Select one:

○ $[3, 0]$

○ $[-2, 5]$

○ $[1, 2]$

● $[2, 3]$

3 Linear Regression [40 pts]

1. **[4pt]** Suppose you have data $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$ and the solution to linear regression on this data is $y = w_1 x + b_1$. Now suppose we have the dataset $(x^{(1)} + \alpha, y^{(1)} + \beta), \ldots, (x^{(n)} + \alpha, y^{(n)} + \beta)$ where $\alpha > 0, \beta > 0$ and $w_1 \alpha \neq \beta$. The solution to the linear regression on this dataset is $y = w_2 x + b_2$. Please select the correct statement about w_1, w_2, b_1, b_2 below. Note that the statement should hold no matter what values α, β take on within the specified constraints.

   Select one:

   - ○ $w_1 = w_2, b_1 = b_2$
   - ○ $w_1 \neq w_2, b_1 = b_2$
   - ● $w_1 = w_2, b_1 \neq b_2$
   - ○ $w_1 \neq w_2, b_1 \neq b_2$

2. **[4pt]** We would like to fit a linear regression estimate to the dataset

$$D = \left\{ \left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), \cdots, \left(\mathbf{x}^{(N)}, y^{(N)}\right) \right\}$$

   with $\mathbf{x}^{(i)} \in \mathbb{R}^M$ by minimizing the ordinary least square (OLS) objective function:

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - \sum_{j=1}^{M} w_j x_j^{(i)} \right)^2$$

   Specifically, we solve for each coefficient w_k $(1 \leq k \leq M)$ by deriving an expression of w_k from the critical point $\frac{\partial J(\mathbf{w})}{\partial w_k} = 0$. What is the expression for each w_k in terms of the dataset $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})$ and $w_1, \cdots, w_{k-1}, w_{k+1}, \cdots, w_M$?

   Select one:

   - ● $w_k = \frac{\sum_{i=1}^{N} x_k^{(i)} (y^{(i)} - \sum_{j=1, j \neq k}^{M} w_j x_j^{(i)})}{\sum_{i=1}^{N} (x_k^{(i)})^2}$
   - ○ $w_k = \frac{\sum_{i=1}^{N} x_k^{(i)} (y^{(i)} - \sum_{j=1, j \neq k}^{M} w_j x_j^{(i)})}{\sum_{i=1}^{N} (y^{(i)})^2}$
   - ○ $w_k = \sum_{i=1}^{N} x_k^{(i)} (y^{(i)} - \sum_{j=1}^{M} w_j x_j^{(i)})$
   - ○ $w_k = \frac{\sum_{i=1}^{N} x_k^{(i)} (y^{(i)} - \sum_{j=1, j \neq k}^{M} w_j x_j^{(i)})}{\sum_{i=1}^{N} (x_k^{(i)} y^{(i)})^2}$

3. **[3pt]** Continuing from the above question, how many coefficients do you need to estimate? When solving for these coefficients, how many equations do you have?

   Select one:

   - ○ N coefficients, M equations
   - ○ M coefficients, N equations
   - ● M coefficients, M equations
   - ○ N coefficients, N equations

4. **[4pt]** We are trying to derive the closed form solution for linear regression:

   In the following, each row in $\mathbf{X}$ denotes one data point and Y is a column vector.

   First we take the derivative of the objective function $L = \frac{1}{2}(\mathbf{X}\mathbf{w} - Y)^T(\mathbf{X}\mathbf{w} - Y)$ with respect to $\mathbf{w}$ and set it to zero, arriving at equation $(*)$.

   Then after some algebraic manipulation, we get the solution $\mathbf{w} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T Y$. What should equation $(*)$ be?

   Select one:

   - ● $(\mathbf{X}\mathbf{w} - Y)^T\mathbf{X} = 0$
   - ○ $(\mathbf{X}\mathbf{w} + Y)^T\mathbf{X} = 0$
   - ○ $\mathbf{X}^T\mathbf{X}\mathbf{w} + Y^T\mathbf{X} = 0$
   - ○ $Y\mathbf{X}^T\mathbf{X}\mathbf{w} + \mathbf{X} = 0$

5. **[3pt]** Suppose we are working with datasets where the number of features is 3. The optimal solution for linear regression is always unique regardless of the number of data points that are in this dataset.

   Select one:

   - ○ True
   - ● False

① Write $J(\theta)$

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left(y^{(i)} - \vec{\theta}^T \vec{x}^{(i)} \right)^2$$

← Ordinary Least Squares (OLS)

$$= \frac{1}{N} \cdot \frac{1}{2} \left(X\vec{\theta} - \vec{y} \right)^T \left(X\vec{\theta} - \vec{y} \right)$$

② "Normal Equations" ← gradient set to zero

$$\nabla J(\theta) = X^T X \vec{\theta} - X^T \vec{y} = 0$$

③ Solve for $\vec{\theta}$

$$\vec{\theta}^{MLE} = (X^T X)^{-1} (X^T \vec{y})$$

$$= \underset{\theta}{\operatorname{argmin}}\, J(\vec{\theta})$$

6. **[3pt]** Identifying whether a function is a convex function is useful because a convex function's local minimum has the nice property that it has to be the global minimum. Please select all functions below that are convex functions. Note $dom(f)$ denotes the domain of the function f.

Select all that apply:

- ■ $f(x) = x, dom(f) = \mathbb{R}$
- ☐ $f(x) = x^3 + 2x + 3, dom(f) = \mathbb{R}$
- ☐ $f(x) = \log x, dom(f) = \mathbb{R}_{++}$ (the set of positive real numbers)
- ■ $f(x) = |x|, dom(f) = \mathbb{R}$
- ■ $f(x) = ||\mathbf{x}||_2, \ dom(f) = \mathbb{R}^n$
- ☐ None of the above.

7. **[3pt]** Typically we can solve linear regression problems in two ways. One is through direct methods, e.g. solving the closed form solution, and the other is through iterative methods, e.g. using stochastic or batch gradient descent methods. Consider a linear regression on data $(\mathbf{X}, \mathbf{y})$. We assume each row in $\mathbf{X}$ denotes one input in the dataset. Please select all options that are correct about the two methods.

Select all that apply:

- ☐ If the matrix $\mathbf{X}^T\mathbf{X}$ is invertible, the exact solution is always preferred for solving the solution to linear regression as computing matrix inversions and multiplications are fast regardless of the size of the dataset.
- ■ Assume N is the number of examples and M is the number of features. The computational complexity of N iterations of batch gradient descent is $\mathcal{O}(MN)$.
- ■ When the dataset is large, stochastic gradient descent is often the preferred method because it gets us reasonably close to the solution faster than both the direct method and batch gradient descent.
- ☐ None of the above.

8. **[3pt]** A data scientist is working on a regression problem on a large data set. After trying stochastic gradient descent (gradient is evaluated on a portion of the data set in each step) and batch gradient descent (gradient is evaluated on the entire data set in each step), the scientist obtained the values of the loss function for the two methods with respect to training time. Note that the same learning rate is used in both cases.

Time in hours	Stochastic GD	Batch GD
1	102.34	120.12
2	80.45	92.37
3	65.23	73.64
4	56.77	58.23
5	52.33	49.21
6	50.74	45.98
7	49.88	43.64

Select all the choices consistent with this table.

Select all that apply:

☐ The table shows, in practice, that stochastic gradient descent can compute a more accurate gradient direction than batch gradient descent.

☐ Within the first 3 hours, the table suggests that batch gradient descent makes more progress in finding the optimum of the objective function than stochastic gradient descent.

■ In general, stochastic gradient descent does not necessarily take a descent step in each step. However, stochastic gradient descent takes much less time to evaluate per step. In this table, during the first hour, stochastic gradient descent makes more update steps to the weights while batch gradient descent makes less updates. Hence it is reasonable that using batch gradient descent likely results in a higher value for the loss function (worse performance) than that of stochastic gradient descent at the 1 hour time point.

☐ None of the above.

9. [**3pt**] Consider the following dataset:

x	1.0	2.0	3.0	4.0	5.0
y	3.0	8.0	9.0	12.0	15.0

If we initialize the weight as 2.0 and bias term as 0.0, what is the gradient of the loss function with respect to the weight w, calculated over all the data points, in the first step of the gradient descent update? Note that we do not introduce any regularization in this problem and and our objective function looks like $\frac{1}{N}\sum_{i=1}^{N}(wx_i + b - y_i)^2$, where N is the number of data points, w is the weight, and b is the bias term.

Fill in the blank with the gradient on the weight you computed, rounded to 2 decimal places after the decimal point.

-23.60

10. [**4pt**] Based on the data of the previous question, please compute the direct solution of the weight and the bias for the objective function defined in the previous question, rounded to 2 decimal places after the decimal point.

Weight: 2.80

Bias: 1.00

11. [**3pt**] Using the dataset and model given in question 9, perform two steps of batch gradient descent on the data. Fill in the blank with the value of the weight after two steps of batch gradient descent. Let the learning rate be 0.01. Round to 2 decimal places after the decimal point.

2.42

12. [**3pt**] Using the dataset and model given in question 9, which of the following learning rates leads to the most optimal weight and bias after performing two steps of batch gradient descent? (Hint: The most optimal learned parameters are the parameters that lead to the lowest value of the objective function.)

Select one:

- ○ 1
- ○ 0.1
- ● 0.01
- ○ 0.001

```
my_array = np.array([1, 1, 1, 1, 1])
my_array_2 = np.array([1, 2, 3, 4, 5])
y = np.array([3, 8, 9, 12, 15])
my_array = np.vstack([my_array, my_array_2])
x = my_array.T
A = np.linalg.inv((np.dot(x.T, x)))
B =  np.dot(x.T, y)
A@B
```

```
array([1. , 2.8])
```

Algorithm 21 GRADIENTDESCENT($\mathcal{F}$, K, $\eta_1, \ldots$)

$z^{(0)} \leftarrow \langle o, o, \ldots, o \rangle$ // initialize variable we are optimizing
for $k = 1 \ldots K$ **do**
  $g^{(k)} \leftarrow \nabla_z \mathcal{F}|_{z^{(k-1)}}$ // compute gradient at current location
  $z^{(k)} \leftarrow z^{(k-1)} - \eta^{(k)} g^{(k)}$ // take a step down the gradient
end for
return $z^{(K)}$

$$\nabla J(\theta) = \frac{2}{N} \sum_{1}^{N} ((Qx - y)x)$$

```
w = 2
learning_rate = 0.1
gradient = np.sum((w * my_array_2 - y) * my_array_2) * 2 / 5
w = w - gradient * learning_rate
gradient = np.sum((w * my_array_2 - y) * my_array_2) * 2 / 5
w = w - gradient * learning_rate
w
```

```
1.5280000000
```

```
w = 2
learning_rate = 0.01
gradient = np.sum((w * my_array_2 - y) * my_array_2) * 2 / 5
w = w - gradient * learning_rate
gradient = np.sum((w * my_array_2 - y) * my_array_2) * 2 / 5
w = w - gradient * learning_rate
w
```

```
2.4200800000
```

```
w = 2
learning_rate = 0.001
gradient = np.sum((w * my_array_2 - y) * my_array_2) * 2 / 5
w = w - gradient * learning_rate
gradient = np.sum((w * my_array_2 - y) * my_array_2) * 2 / 5
w = w - gradient * learning_rate
w
```

```
2.0466808000
```

```
w = 2
learning_rate = 1
gradient = np.sum((w * my_array_2 - y) * my_array_2) * 2 / 5
w = w - gradient * learning_rate
gradient = np.sum((w * my_array_2 - y) * my_array_2) * 2 / 5
w = w - gradient * learning_rate
w
```

```
-470.0000000000
```

Collaboration Questions Please answer the following:

After you have completed all other components of this assignment, report your answers to the collaboration policy questions detailed in the Academic Integrity Policies found here.

1. Did you receive any help whatsoever from anyone in solving this assignment? Is so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? Is so, include full details.
3. Did you find or come across code that implements any part of this assignment ? If so, include full details.

Solution

1. No
2. No
3. No