

UNIVERSITY OF  
**WATERLOO**



UNIVERSITY OF WATERLOO  
CHERITON SCHOOL OF COMPUTER SCIENCE

## CS 480 - Homework 1

**Prepared by:**

Jianxiang (Jack) Xu [20658861]

9 February 2021

## CS480/680: Introduction to Machine Learning

## Homework 1

Due: 11:59 pm, January 28, 2021, submit on Crowdmark (yet to be set up, stay tuned).

## Exercise 1: Perceptron Implementation (5 pts)

**Convention:** All algebraic operations, when applied to a vector or matrix, are understood to be element-wise (unless otherwise stated).

## Algorithm 1.1: The perceptron algorithm.

**Input:**  $X \in \mathbb{R}^{n \times d}$ ,  $\mathbf{y} \in \{-1, 1\}^n$ ,  $\mathbf{w} = \mathbf{0}_d$ ,  $b = 0$ ,  $\text{max\_pass} \in \mathbb{N}$

**Output:**  $\mathbf{w}, b, \text{mistake}$

```

1 for  $t = 1, 2, \dots, \text{max\_pass}$  do
2    $\text{mistake}(t) \leftarrow 0$ 
3   for  $i = 1, 2, \dots, n$  do
4     if  $y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \leq 0$  then
5        $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$  //  $\mathbf{x}_i$  is the  $i$ -th row of  $X$ 
6        $b \leftarrow b + y_i$ 
7      $\text{mistake}(t) \leftarrow \text{mistake}(t) + 1$ 

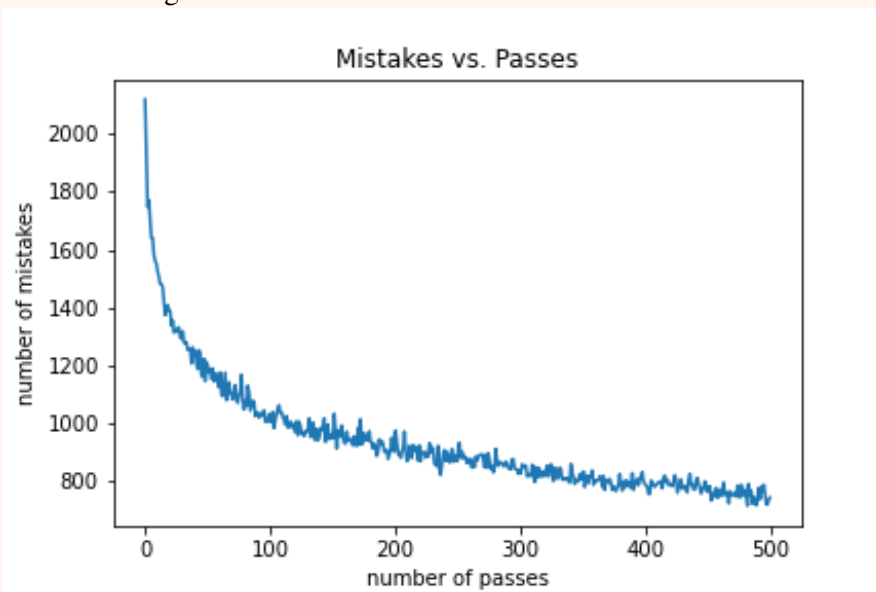
```

Implement the perceptron in Algorithm 1.1. Your implementation should take input as  $X = [\mathbf{x}_1^\top, \dots, \mathbf{x}_n^\top]^\top \in \mathbb{R}^{n \times d}$ ,  $\mathbf{y} \in \{-1, 1\}^n$ , an initialization of the hyperplane parameters  $\mathbf{w} \in \mathbb{R}^d$  and  $b \in \mathbb{R}$ , and the maximum number of passes of the training set [suggested  $\text{max\_pass} = 500$ ]. Run your perceptron algorithm on the [spambase](#) dataset (use the version on the [course website](#)), and plot the number of mistakes (y-axis) w.r.t. the number of passes (x-axis).

Ans: [Answer 1.1](#)

## Answer 1.1

Resultant image:



## Perceptron Implementation:

```
1  def perceptron(  
2      X: List[List[float]],  
3      y: List[int],  
4      max_pass=500  
5  )-> [List[float], float, List[int]]:  
6      """  
7      @param      X: \in R^{nxd}  
8      @param      y: \in {-1,1}^n  
9      @param      max_pass: \in N  
10     """  
11     X = np.array(X)  
12     y = np.array(y)  
13     [n, d] = np.shape(X)  
14     w = [0] * d # w = 0_d  
15     b = 0  
16     mistake = []  
17     for t in range(0, max_pass): # max passes / iterations  
18         mistake.append(0)  
19         for i in range(0, n):  
20             x_i = X[i, :]  
21             if (y[i] * (np.dot(x_i, w) + b)) <= 0:  
22                 w = w + y[i] * x_i  
23                 b = b + y[i]  
24                 mistake[t] += 1  
25  
26     return w, b, mistake  
27
```

**Exercise 2: Perceptron Questions (5 pts)**

1. (3 pts) The perceptron algorithm makes an update every time it witnesses a mistake. What if it makes an update on every point, regardless of whether or not that point is correctly classified? For simplicity, consider a setting where  $b$  is fixed to 0. Give an example of an infinite sequence of points  $(x_i, y_i)$  with the following properties:

- I The sequence is strictly linearly separable with  $b = 0$  (i.e., the margin is some constant  $\gamma > 0$ ),
- II The sequence has  $\max \|x_i\|_2 \leq 1$ ,
- III The modified perceptron algorithm makes **an infinite number of mistakes on this sequence**.

Prove that it has these properties. Note that the perceptron convergence theorem and the first two conditions imply that, at some point, the unmodified perceptron algorithm would only make a finite number of mistakes.

Ans: [Answer 2.1](#)

2. (1 pt) Give examples of where the perceptron algorithm converges to a 0 margin halfspace, and a **separate example where it converges to a maximum margin halfspace**. **As pointed out by some on Piazza, technically, if a halfspace has 0 margin, then it would misclassify anything that still lies on the hyperplane, and the perceptron algorithm would not yet have halted. If you came up with an example that ignores these cases and halts with a point on the hyperplane, that's fine. However, the intent was more like the following, so consider solving the problem where it converges to an arbitrarily small margin halfspace.** More precisely: for any  $0 < \epsilon < 1/2$ , give a dataset (with margin at least 1) and a order in which to process the points such that the perceptron algorithm halts providing a halfspace with margin  $\leq \epsilon$ . **This problem has to do with the original perceptron, not the modified perceptron from part 1.**

Ans: [Answer 2.2](#)

3. (1 pt) Suppose that in each iteration of the perceptron algorithm, a point is chosen uniformly at random from the dataset. Show how the perceptron algorithm can be viewed as an instantiation of stochastic gradient descent (SGD). In particular, you must provide a loss function and a learning rate such that SGD with these parameters and perceptron are identical.

Ans: [Answer 2.3](#)

**Answer 2.1: Infinite Mistakes**

From the Property III and Property I, we may conclude the following:

$$\therefore \text{mistake } \forall i, \text{ and set threshold } (\delta = 0) \quad (1)$$

$$\therefore y_i(\langle \mathbf{x}_i, \mathbf{w}_{i-1} \rangle + b) \leq 0, \forall i \in [1, \infty) \quad (2)$$

$$\Rightarrow \mathbf{w}_i = \mathbf{w}_{i-1} + y_i \mathbf{x}_i, \forall i \in [1, \infty) \quad (3)$$

$$\therefore \mathbf{w}_k = \mathbf{w}_0 + y_1 \mathbf{x}_1 + \dots + y_k \mathbf{x}_k, \forall k \in [1, \infty) \quad (4)$$

$$\mathbf{w}_k = \mathbf{w}_0 + \sum_{i=1}^k y_i \mathbf{x}_i, \forall k \in [1, \infty) \quad (5)$$

$$\therefore \text{For simplicity, } b = 0 \quad (6)$$

$$\therefore b_i = 0, \forall i \quad (7)$$

$$\therefore \text{By convention, } \mathbf{w}_0 = \mathbf{0} \text{ will make the first always as mistake} \quad (8)$$

$$\therefore \mathbf{w}_k = \sum_{i=1}^k y_i \mathbf{x}_i, \forall k \in [1, \infty) \quad (9)$$

$$y_{k+1} \langle \mathbf{x}_{k+1}, \mathbf{w}_k \rangle \leq 0 \forall k \in [1, \infty) \quad (10)$$

$$\therefore \langle y_{k+1} \mathbf{x}_{k+1}, \mathbf{w}_k \rangle \leq 0 \quad (11)$$

$$\langle y_{k+1} \mathbf{x}_{k+1}, \sum_{i=1}^k y_i \mathbf{x}_i \rangle \leq 0 \quad (12)$$

$$\therefore \text{Let } \mathbf{a}_i = y_i \mathbf{x}_i, \forall i \in [1, \infty) \quad (13)$$

$$\therefore \langle \mathbf{a}_{k+1}, \sum_{i=1}^k \mathbf{a}_i \rangle \leq 0 \Rightarrow \sum_{j=1}^d \left( a_{k+1,j} \sum_{i=1}^k a_{i,j} \right) \leq 0 \Rightarrow \sum_{j=1}^d \sum_{i=1}^k a_{k+1,j} a_{i,j} \leq 0 \quad (14)$$

Hence:

$$\therefore \text{ if } \mathcal{D} = \left\{ (\mathbf{x}_i, y_i) : \sum_{j=1}^d \sum_{i=1}^k a_{k+1,j} a_{i,j} \leq 0, \mathbf{a}_i = y_i \mathbf{x}_i, i \in [1, \infty) \right\} \quad (15)$$

$$\rightarrow \text{Make mistakes on infinity many points} \quad (16)$$

From the Property II, we may add an additional constraint on the example:

$$\sum_{j=1}^d \mathbf{x}_{ij}^2 \leq 1 \Rightarrow \sum_{j=1}^d a_{ij}^2 \leq 1 \quad (17)$$

Hence, in order to satisfy all three properties, the dataset would be:

$$\mathcal{D} = \left\{ (\mathbf{x}_i, y_i) : \sum_{j=1}^d \sum_{i=1}^k a_{k+1,j} a_{i,j} \leq 0, \sum_{j=1}^d a_{i,j}^2 \leq 1, \mathbf{a}_i = y_i \mathbf{x}_i, i \in [1, \infty), y_i \in \{1, -1\} \right\} \quad (18)$$

**Alert 2.1**

here are infinity many examples for this question. Here, we will mathematically derive one with assumptions.

For simplicity, let's consider a 2D case (Assumption II), which allows us to visualize geometrically, since  $w_i^T \mathbf{a}_i = |w_i| |\mathbf{a}_i| \cos(\theta_{w_i, \mathbf{a}_i})$ . The thought process here is that if we make all labels as +1 (Assumption III),

then  $a_i = x_i$  and  $w_{i+1} = w_i + x_i$ , which will simplify many things. In order to ensure the linearly separable property, we can assume all the points are on the right half of the plane and may only include one side of the vertical axis (Assumption IV).

Let's summarize up these assumptions:

I Property II  $\Rightarrow \|x_i\|_2 \leq 1$

II 2-D data:  $d = 2$

III  $y_i = 1 \forall i$

IV All the points are on the right hand plane:  $a_{i,1} \geq 0$

Hence, from Assumption II, we may simplify Equation (14) to:

$$\sum_{j=1}^{d=2} \sum_{i=1}^k a_{k+1,j} a_{i,j} \leq 0 \quad (19)$$

$$a_{k+1,1} \sum_{i=1}^k a_{i,1} + a_{k+1,2} \sum_{i=1}^k a_{i,2} \leq 0 \quad (20)$$

$$a_{k+1,1} \sum_{i=1}^k a_{i,1} \leq -a_{k+1,2} \sum_{i=1}^k a_{i,2} \quad (21)$$

Expanding per  $(k+1)$ :

$$k+1=1 \quad 0 \leq 0 \quad (22)$$

$$k+1=2 \quad a_{2,1}(a_{1,1}) \leq -a_{2,2}(a_{1,2}) \quad (23)$$

$$k+1=3 \quad a_{3,1}(a_{1,1} + a_{2,1}) \leq -a_{3,2}(a_{1,2} + a_{2,2}) \quad (24)$$

$$k+1=4 \quad a_{4,1}(a_{1,1} + a_{2,1} + a_{3,1}) \leq -a_{4,2}(a_{1,2} + a_{2,2} + a_{3,2}) \quad (25)$$

$$\vdots \quad (26)$$

From this inequality, we can see the solution would be alternating in quadrant 2 and 4.

In addition,

$$\therefore w_i^T \mathbf{a}_i = |w_i| |\mathbf{a}_i| \cos(\theta_{w_i, \mathbf{a}_i}) \quad w_i^T \mathbf{a}_i \leq 0 \quad (27)$$

$$\therefore \pi \geq \theta_{w_i, \mathbf{a}_i} \geq \frac{\pi}{2}, \forall \theta_{w_i, \mathbf{a}_i} \in [0, \pi] \quad (28)$$

This would ensure all points locate on one side of the plane, and makes mistakes.

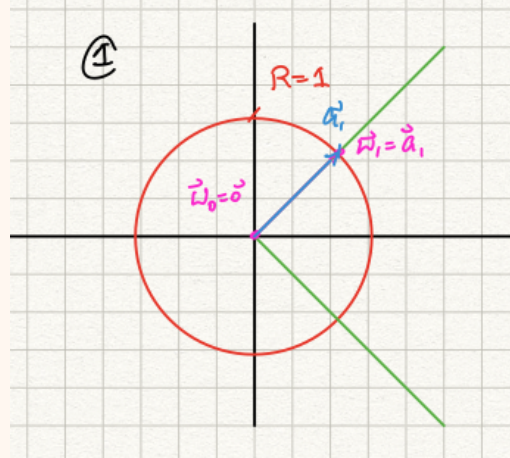
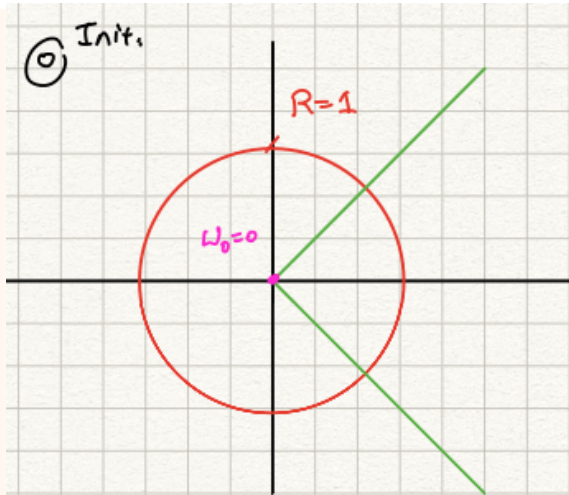
Let's draw out the concept in 2D:

Initially,  $w_0$  starts from the origin. Updating  $w_{i+1} = w_i + \mathbf{a}_i$  essentially means  $w_i$  vector keeps accumulates the  $\mathbf{a}_i$ . If the  $w_i$  vector forms an angle larger than or equal to 90 degree, it guarantee it makes a mistake. Property III indicates an infinity mistakes, but permits some mistakes in between. Hence, the goal is to make this aggressive updating method make mistakes and never converges.

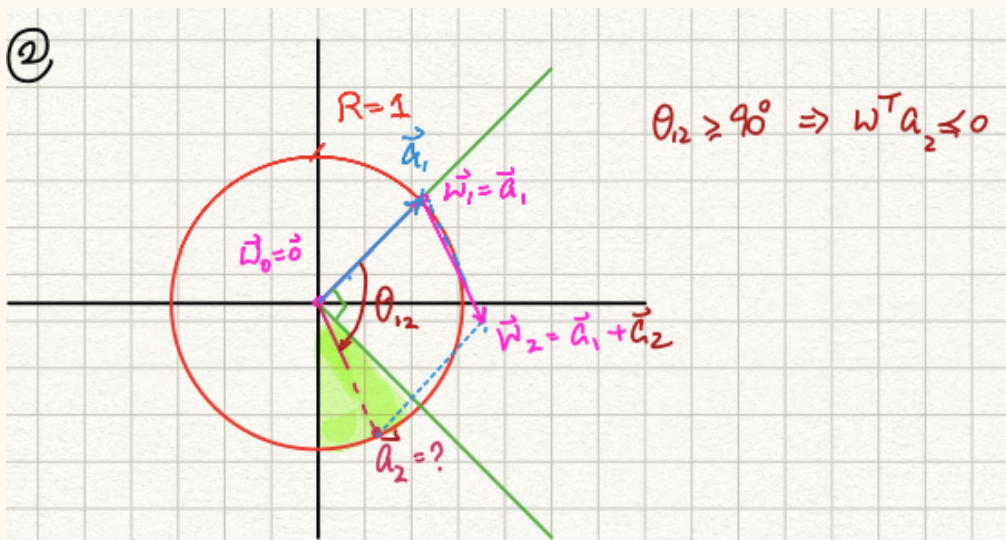
Say if we assume there exists repetitive series of points, that ensures the last point return back to the starting axis, but also guarantee makes an angle larger or equal than 0 all the time between the last point and the prior weight before updating.

As a result, we shall try to make a set of three vectors that can bring the  $w_{3n}, n \in \mathbb{Z}$  back to the x-axis ( $w_{3n,2} = 0$ ), and we want  $w_{3n-1,2}$  and  $\mathbf{a}_{3n}$  form an angle larger or equal than zero ( $\theta_{3n-1 \rightarrow 3n} = 0$ )

For simplicity, let's start with a 45 degree angle ray sourced from the origin and intersect with  $R$  (L2 norm boundary), where, we obtain first point  $\mathbf{a}_1 = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$ .

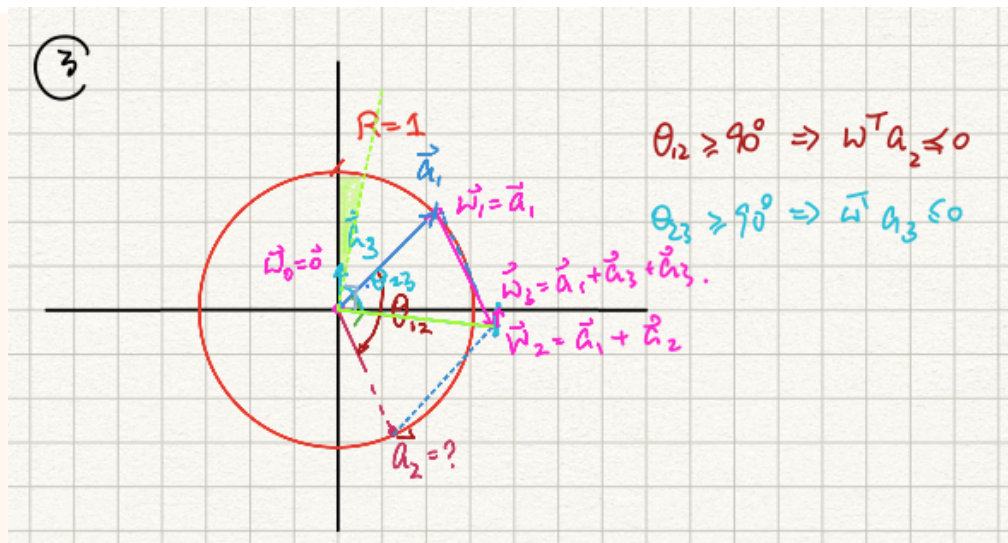


Hence, the second point has to be in the green region in quadrant 2, so that it forms an angle larger than 90 degree. In addition, we want the updated  $w_2$  point fall in quadrant 4, so that it can be pulled back to x-axis with the third point in quadrant 2 ( $a_{22} + a_{12} < 0$ ). For simplicity, we can assume a point that forms a 60 degree from the y axis with an l2 norm of 1. (You may pick any points from this region.) As a result, we compute  $\mathbf{a}_2 = (\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$ .



Now, we shall pull the weight vector back to x-axis. Hence,  $a_{12} + a_{22} + a_{32} = 0$ , resulting  $a_{32} = \frac{\sqrt{3}}{2} - \frac{1}{\sqrt{2}}$ . Most important, we need to make sure this vector  $\mathbf{a}_3$  would form an angle larger than 90 degree. Since the weight vector  $w_2$  will always fall below the x-axis, hence, we shall make every third point locates on positive y-axis ( $a_{31} = 0$ ), so that it guarantees a minimum of 90 degree as  $i \rightarrow \infty$ . Hence,  $\mathbf{a}_3 = (0, \frac{\sqrt{3}}{2} - \frac{1}{\sqrt{2}})$ .

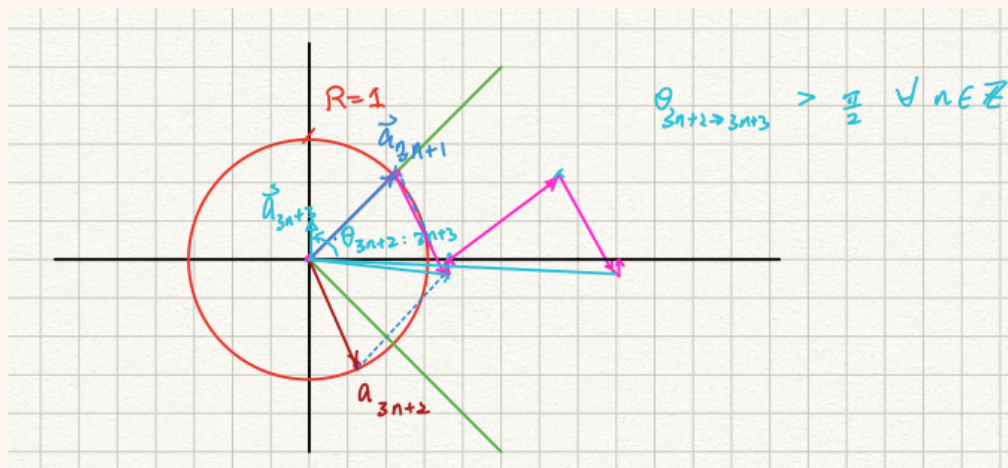




Recall  $\mathbf{x}_i = \mathbf{a}_i, y_i = 1$ , therefore, we have obtained the infinity sequence:

**Remark 2.1: Final Derived Three-Point Example of an infinite sequence of points**

$$(\mathbf{x}_i, y_i) = \begin{cases} ((\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}), 1) & i = 3n + 1 \\ ((\frac{1}{2}, -\frac{\sqrt{3}}{2}), 1) & i = 3n + 2, \quad n \in \mathbb{Z} \\ ((0, \frac{\sqrt{3}}{2} - \frac{1}{\sqrt{2}}), 1) & i = 3n + 3 \end{cases} \quad (29)$$



By observation, this repeated series will satisfies three properties from Property I, Property II, and Property III. Now, let's formerly prove it does indeed meet all properties:

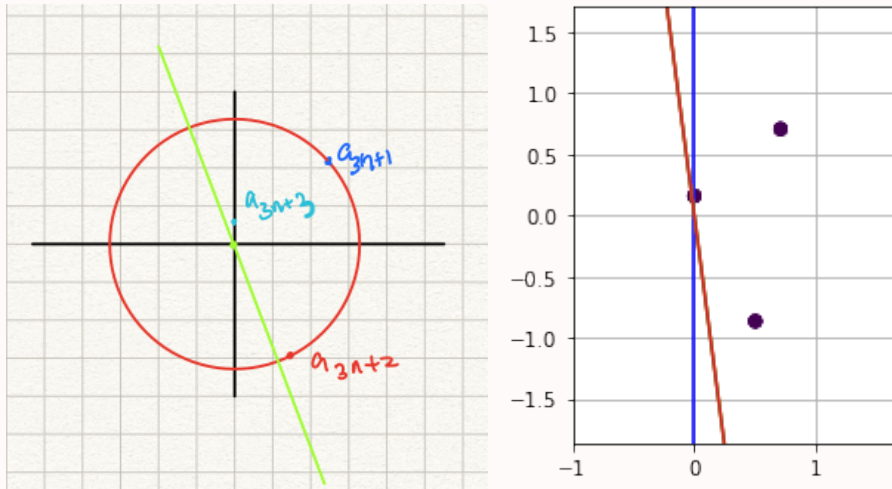


### Proof 2.1: Property I - Strictly Linearly Separable

Since all labels are +1, and all data points are asymmetric, hence the data  $\mathcal{D}$  is strictly linear separable with  $b = 0$ .

And the algorithm shall only make the first mistake in the ordinary perceptron with  $b = 0$ , simply updating  $w$  and  $b$  to  $w = (1/\sqrt{2}, 1/\sqrt{2})$ ,  $b = 0$ , and then halt update. Hence, there exists  $w = (1/\sqrt{2}, 1/\sqrt{2})$ ,  $b = 0$  such that  $\forall i, \mathbf{a}_i^T w \geq s \geq 0$ . As a result, we conclude the dataset is truly strictly linearly separable.

As shown in the graph below, there would exist some hyperplane, where all points on one side of the hyperplane through the origin with  $\gamma > 0, b = 0$ .



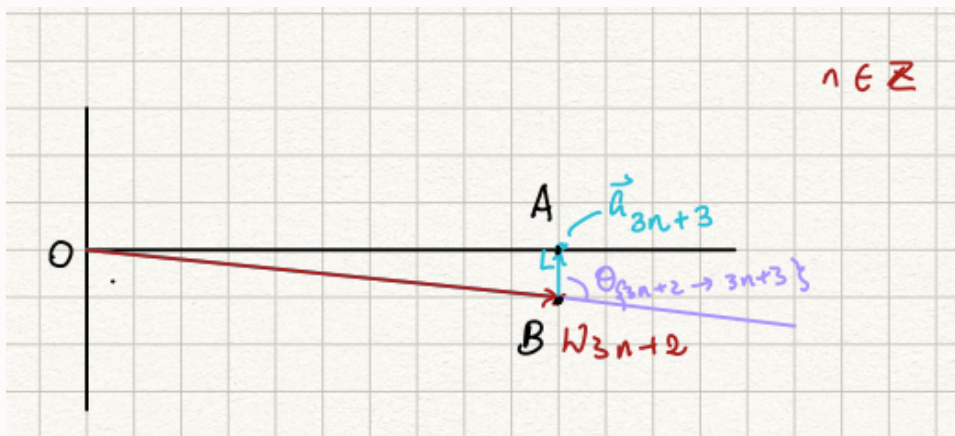
**Proof 2.2: Property II -  $\max ||x_i||_2 \leq 1$**

$$\therefore \quad \|\mathbf{a}_{3n+1}\|_2 = 1 \quad \|\mathbf{a}_{3n+2}\|_2 = 1 \quad \|\mathbf{a}_{3n+3}\|_2 = \frac{\sqrt{3}}{2} - \frac{1}{\sqrt{2}} \quad \forall n \in \mathbb{Z} \quad (30)$$

$$\therefore \|x_i\|_2 = \|a_i\|_2 \leq 1, \forall i \in \mathbb{Z} \quad (31)$$

### Proof 2.3: Property III: infinite number of mistakes

We can guarantee the angle formed between every third point and prior weight vector would be larger or equal than 90 degree. As the figure below shown, the angle  $\theta_{w_{3n+2}} \rightarrow \mathbf{a}_{3n+3} \geq \frac{\pi}{2}, \forall n \in \mathbb{Z}$ .



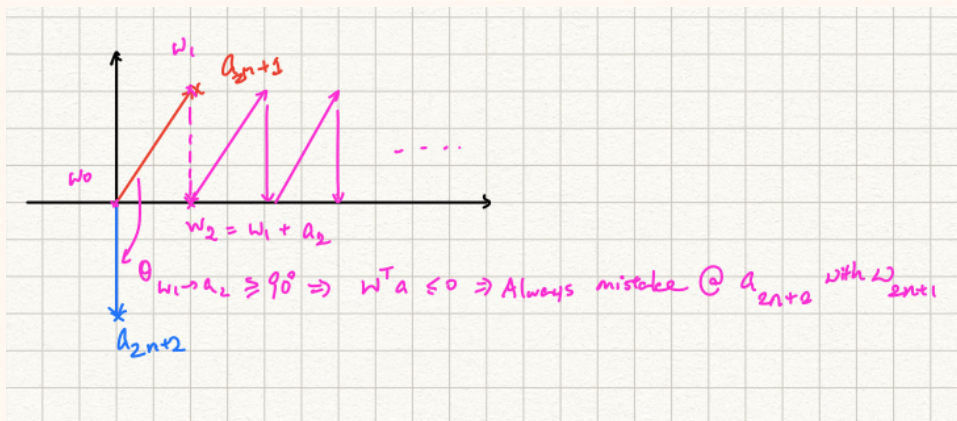
## Interesting Conclusion

Interestingly to note, for given any repeated set of points, as long as we ensure the last point  $\mathbf{a}_k$  will make  $w_k = w_{k-1} + \mathbf{a}_k$  back to x-axis and form an 90 degree or larger angle with with prior weight  $w_{k-1}$ .

For a 2 point series case in 2D, it is much simpler, simply any arbitrary first point in quadrant 1, and a vector on negative y-axis that will bring the first point back to x-axis:

### Remark 2.2: Generalized Two-point Solution of an infinite sequence of points

$$(\mathbf{x}_i, y_i) = \begin{cases} (\mathbf{x}_{2n+1}, y_{2n+1}) = ((x_{2n+1,1}, x_{2n+1,2}), 1) \\ (\mathbf{x}_{2n+2}, y_{2n+2}) = ((0, -x_{2n+1,2}), 1) \end{cases} \quad x_{2n+1,1} \in \mathbb{R}^+, x_{2n+1,2} \in \mathbb{R}, n \in \mathbb{Z} \quad (32)$$

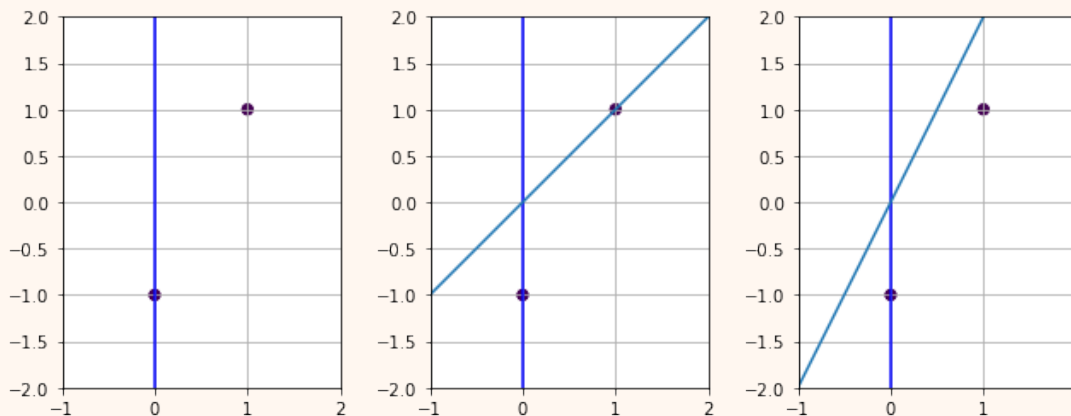


For example:

**Remark 2.3: Two-point Example of an infinite sequence of points**

$$(\mathbf{x}_i, y_i) = \begin{cases} ((1, 1), 1) & i = 2n + 1 \\ ((0, -1), 1) & i = 2n + 2 \end{cases}, \quad n \in \mathbb{Z} \quad (33)$$

It is strictly linearly separable with  $b = 0$ :



**Q.E.D.**

**Answer 2.2: Margin****a. Example for  $\varepsilon$  margin halfspace:**

It happens when an arbitrary working hyperplane is formed biased towards the last mistaken points.

Simplest Example:

$$(x_i, y_i) = \begin{cases} ((2, 2), 1) & i = 2n + 1 \\ ((-0.5, -0.5), -1) & i = 2n + 2 \end{cases}, \forall n \in \mathbb{Z}$$

Proof:

Iteration 1: Since  $y_1(\langle w_0, x_1 \rangle + b_0) = 0 \leq 0, \Rightarrow w_1 = w_0 + y_1 x_1 = (2, 2)$  and  $b_1 = b_0 + y_1 = 1$

Iteration 2: Since  $y_2(\langle w_1, x_2 \rangle + b_1) = -1 * (\langle (-0.5, -0.5), (2, 2) \rangle + 1) = 1 > 0, \Rightarrow$  Do not update

Iteration 3: Since  $y_3(\langle w_2, x_3 \rangle + b_2) = 1 * (\langle (2, 2), (2, 2) \rangle + 1) = 9 > 0, \Rightarrow$  Do not update

Iteration 2i: As stated in Iteration 2

Iteration 3i: As stated in Iteration 3

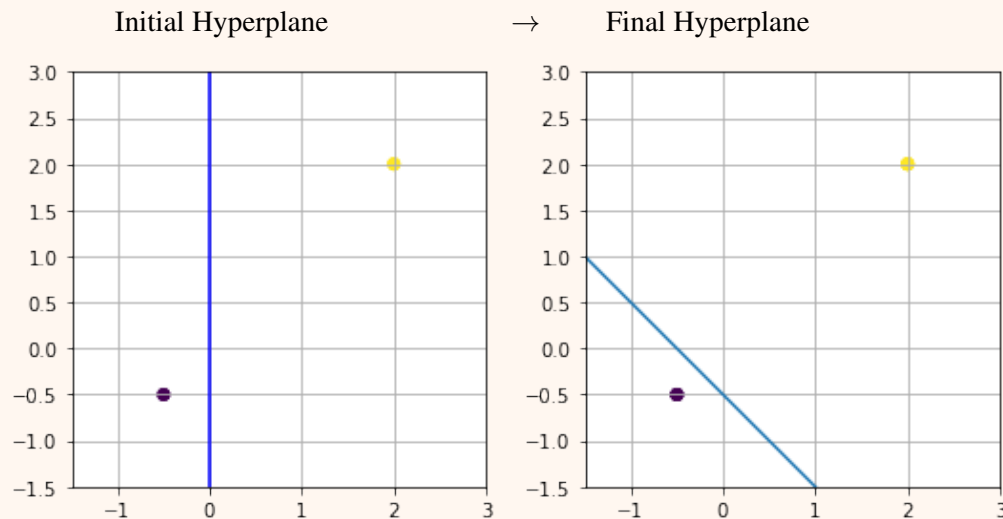
Hence, it will never update after the first update.

The converged hyperplane is defined by  $w = (2, 2), b = 1$

Hence, we may compute the margin:  $\gamma = 0.5 / \sqrt{2} = 0.3535 < 1/2$

And, the max geometric margin:  $\hat{\gamma} = \|(2, 2) - (-0.5, -0.5)\| = 3.5355 > 1$

Hence, in this case, the margin obtained is smaller than the max possible geometric margin.



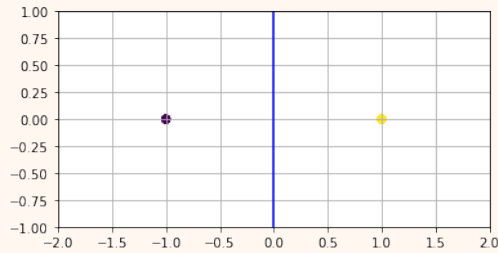
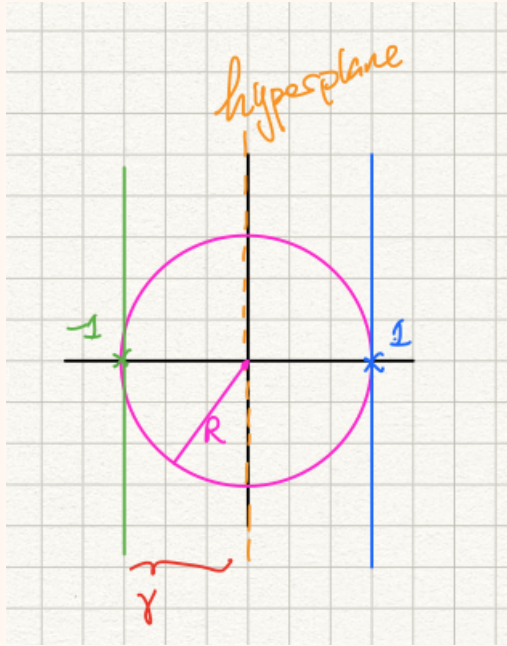
**b. Example for a maximum margin halfspace:**

It happens when  $\gamma = R = \max_i \|a_i\|_2$ ,

hence, all the data points are concentrated on two symmetrical points with opposing label.

It would also make one mistake  $\lim_{\gamma \rightarrow R} (\frac{R}{\gamma})^2 = 1$

Example:  $(x_i, y_i) = (((-1)^i, 0), (-1)^i)$ ,  $\gamma = R = \max_i \|a_i\|_2 = 1$



**Answer 2.3: SGD**

The goal for Stochastic Gradient Descent (SGD) is to learn a model to minimize the loss  $\ell(\mathbf{w}; \mathbf{x}_i, y_i)$  over a training set  $\mathcal{D} = (\mathbf{x}_i, y_i)$ . Similarly, we can consider the minimization is related to a linear mode with form of  $f_w(x) = \langle \mathbf{w}, x \rangle$ . Hence:

$$\min_w \frac{1}{n} \sum_{i=1}^n \ell(\mathbf{w}; \mathbf{x}_i, y_i) = \min_w \frac{1}{n} \sum_{i=1}^n \ell(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i) \quad (34)$$

Then SGD basically updates the weight vector  $\mathbf{w}_t$  by performing an approximate gradient descent:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \ell(\mathbf{w}; \mathbf{x}_i, y_i) \quad (35)$$

Recall the perceptron update:

$$\text{if } y_i(\langle \mathbf{w}_i, \mathbf{x}_i \rangle + b_i) \leq 0 \Rightarrow (y_i \langle \mathbf{w}_i, \mathbf{x}_i \rangle + y_i b_i) \leq 0 \quad (36)$$

$$\text{then } \mathbf{w}_{i+1} \leftarrow \mathbf{w}_i + y_i \mathbf{x}_i \quad (37)$$

$$b_{i+1} \leftarrow b_i + y_i \quad (38)$$

As Equation (36) stated, when  $y(\langle \mathbf{w}, \mathbf{x} \rangle + b)$  is larger than zero, the system does not update. It updates only when  $y(\langle \mathbf{w}, \mathbf{x} \rangle + b)$  is less than zero (when there is a mistake).

Hence, it is equivalent to update when the  $-y(\langle \mathbf{w}, \mathbf{x} \rangle + b)$  is larger than zero. This can be the loss function we try to minimize for when the term is larger than zero.

Hence, we may assume the loss function as:

$$\ell(\mathbf{w}; \mathbf{x}_i, y_i) = \max(0, -y_i \langle \mathbf{w}, \mathbf{x}_i \rangle - y_i b) \quad (39)$$

Conveniently, this result a gradient:

$$\nabla_{\mathbf{w}, b} \ell(\mathbf{w}; \mathbf{x}_i, b; y_i) = \frac{\partial \max(0, -y_i \langle \mathbf{w}, \mathbf{x}_i \rangle - y_i b)}{\partial \mathbf{w}} \quad (40)$$

We may also separate them into two distinct components for  $w$  and  $b$ :

$$\nabla_{\mathbf{w}} \ell(\mathbf{w}; \mathbf{x}_i, y_i) = \max(0, -y_i \mathbf{x}_i) \quad (41)$$

$$\nabla_b \ell(b; y_i) = \max(0, -y_i) \quad (42)$$

Since the data at each iteration 't' of SGD is randomly selected from the dataset  $\mathcal{D}$ , hence we may replace  $(y_i, \mathbf{x}_i)$  with sampled dataset  $(y_t, \mathbf{x}_t)$ , which is characterized in a Stochastic GD instead of the GD method.

As a result, the SGD update function (Equation (35)) becomes:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \ell(\mathbf{w}; \mathbf{x}_i, y_i) = \mathbf{w}_t - \eta_t \max(0, -y_i \mathbf{x}_i) = \max(\mathbf{w}_t, \mathbf{w}_t + \eta_t y_i \mathbf{x}_i) \quad (43)$$

$$b_{t+1} \leftarrow b_t - \eta_t \nabla_b \ell(b; y_i) = b_t - \eta_t \max(0, -y_i) = \max(b_t, b_t + \eta_t y_i) \quad (44)$$

As Equation (43) and (44) stated, the SGD formulation stated above returns exact result as the update function for the weight in the perceptual algorithm (Equation (37) and Equation (38) respectively), when the learning rate is the step size  $\eta_t = 1$ . It either maintain the current  $\mathbf{w}_t$  and  $b_t$ , or update towards a better location to minimize the loss function (in another word, when it makes a mistake).

In conclusion, we may conclude the perceptron algorithm can be viewed as an instantiation of stochastic gradient descent (SGD), with:

- a loss function:  $\ell(\mathbf{w}; \mathbf{x}_i, y_i) = \max(0, -y_i \langle \mathbf{w}, \mathbf{x}_i \rangle - y_i b)$
- a learning rate:  $\eta_t = 1$

**Exercise 3: Regression Implementation (8 pts)**

Recall that ridge regression refers to

$$\min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \underbrace{\frac{1}{2n} \|X\mathbf{w} + b\mathbf{1} - \mathbf{y}\|_2^2}_{\text{error}} + \underbrace{\lambda \|\mathbf{w}\|_2^2}_{\text{loss}}, \quad (45)$$

where  $X \in \mathbb{R}^{n \times d}$  and  $\mathbf{y} \in \mathbb{R}^n$  are the given dataset and  $\lambda > 0$  is the regularization hyperparameter.

1. (1 pt) Show that the derivatives are

$$\frac{\partial}{\partial \mathbf{w}} = \frac{1}{n} X^\top (X\mathbf{w} + b\mathbf{1} - \mathbf{y}) + 2\lambda \mathbf{w} \quad (46)$$

$$\frac{\partial}{\partial b} = \frac{1}{n} \mathbf{1}^\top (X\mathbf{w} + b\mathbf{1} - \mathbf{y}). \quad (47)$$

Ans: [Answer 3.1](#)

2. (2 pts) Implement the gradient descent algorithm for solving ridge regression. The following **incomplete** pseudo-code may of help.

Test your implementation on the Boston **housing** dataset (to predict the median house price, i.e.,  $y$ ). Use the train and test splits provided on **course website**. Try  $\lambda \in \{0, 10\}$  and report your training error, training loss and test error. [Your training loss should monotonically decrease during iteration; if not try to tune your step size  $\eta$ , e.g. make it smaller.]

**Algorithm 1:** Gradient descent for ridge regression.

**Input:**  $X \in \mathbb{R}^{n \times d}$ ,  $\mathbf{y} \in \mathbb{R}^n$ ,  $\mathbf{w}_0 = \mathbf{0}_d$ ,  $b_0 = 0$ ,  $\text{max\_pass} \in \mathbb{N}$ ,  $\eta > 0$ ,  $\text{tol} > 0$

**Output:**  $\mathbf{w}, b$

```

1 for  $t = 1, 2, \dots, \text{max\_pass}$  do
2    $\mathbf{w}_t \leftarrow$ 
3    $b_t \leftarrow$ 
4   if  $\|\mathbf{w}_t - \mathbf{w}_{t-1}\| \leq \text{tol}$  then // can use other stopping criteria
5     break
6  $\mathbf{w} \leftarrow \mathbf{w}_t$ ,  $b \leftarrow b_t$ 
```

Ans: [Answer 3.2](#)

For the next part, you may use the Python package scikit-learn.

3. (5 pts) Train (unregularized) linear regression, ridge regression, and lasso on the mystery datasets A, B, and C on the course website (using `X_train` and `Y_train` for each dataset). For ridge regression and lasso, use parameters 1 and 10 (note that you will have to divide these by  $n$  for lasso in scikit-learn -- why?). Report the average mean squared error on the test set for each method. Which approach performs best in each case? Plot the five parameter vectors obtained for each dataset on the same histogram, so they're all visible at once (change the opacity setting for the bars if necessary): specifically, for each parameter vector, plot a histogram of its value in each coordinate. Given which approach performs best, and how its parameter histogram looks, how do you suspect the true parameter vector and responses might have been generated?

Ans: [Answer 3.3](#)



**Answer 3.1: Derivatives Derivation**

Let's simplify the loss function (in Equation (45)) with some sub-group functions:

$$f_{error}(X, \mathbf{w}, \mathbf{y}, b) = X\mathbf{w} + b\mathbf{1} - \mathbf{y} \quad (48)$$

$$f_{reg}(\mathbf{w}) = \mathbf{w} \quad (49)$$

We will also apply partial derivative to these sub-group functions with respect to  $\mathbf{w}$  and  $b$ :

$$\frac{\partial f_{error}(X, \mathbf{w}, \mathbf{y}, b)}{\partial \mathbf{w}} = \frac{\partial (X\mathbf{w} + b\mathbf{1} - \mathbf{y})}{\partial \mathbf{w}} \quad (50)$$

$$= \frac{\partial (X\mathbf{w})}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial (\mathbf{x}_1^T \mathbf{w})}{\partial \mathbf{w}} \\ \frac{\partial (\mathbf{x}_2^T \mathbf{w})}{\partial \mathbf{w}} \\ \vdots \\ \frac{\partial (\mathbf{x}_n^T \mathbf{w})}{\partial \mathbf{w}} \end{bmatrix} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \quad \mathbf{x}_n] = X^T \quad (51)$$

$$\frac{\partial f_{error}(X, \mathbf{w}, \mathbf{y}, b)}{\partial b} = \frac{\partial (X\mathbf{w} + b\mathbf{1} - \mathbf{y})}{\partial b} \quad (52)$$

$$= \frac{\partial (b\mathbf{1})}{\partial b} = \mathbf{1}^T \quad (53)$$

$$\frac{\partial f_{reg}(\mathbf{w})}{\partial \mathbf{w}} = 1 \quad (54)$$

$$\frac{\partial f_{reg}(\mathbf{w})}{\partial b} = 0 \quad (55)$$

Hence, the loss function now becomes:

$$\ell(X, \mathbf{w}, \mathbf{y}, b) = \frac{1}{2n} \|f_{error}(X, \mathbf{w}, \mathbf{y}, b)\|_2^2 + \lambda \|f_{reg}(\mathbf{w})\|_2^2 \quad (56)$$

Now, we may apply 1st order derivatives with Chain Rule:

$$\frac{\partial \ell(X, \mathbf{w}, \mathbf{y}, b)}{\partial \mathbf{w}} = \frac{\partial \frac{1}{2n} \|f_{error}(X, \mathbf{w}, \mathbf{y}, b)\|_2^2 + \lambda \|f_{reg}(\mathbf{w})\|_2^2}{\partial \mathbf{w}} \quad (57)$$

$$= \frac{1}{2n} \cdot 2 \cdot \frac{\partial f_{error}}{\partial \mathbf{w}} \cdot f_{error} + \lambda \cdot 2 \cdot \frac{\partial f_{reg}}{\partial \mathbf{w}} \cdot f_{reg} \quad (58)$$

$$= \frac{1}{n} \cdot X^T \cdot (X\mathbf{w} + b\mathbf{1} - \mathbf{y}) + 2\lambda \cdot 1 \cdot \mathbf{w} \quad (59)$$

$$= \frac{1}{n} X^T (X\mathbf{w} + b\mathbf{1} - \mathbf{y}) + 2\lambda \mathbf{w} \quad (60)$$

$$\frac{\partial \ell(X, \mathbf{w}, \mathbf{y}, b)}{\partial b} = \frac{\partial \frac{1}{2n} \|f_{error}(X, \mathbf{w}, \mathbf{y}, b)\|_2^2 + \lambda \|f_{reg}(\mathbf{w})\|_2^2}{\partial b} \quad (61)$$

$$= \frac{1}{2n} \cdot 2 \cdot \frac{\partial f_{error}}{\partial b} \cdot f_{error} + 0 \quad (62)$$

$$= \frac{1}{n} \cdot \mathbf{1}^T \cdot (X\mathbf{w} + b\mathbf{1} - \mathbf{y}) \quad (63)$$

$$= \frac{1}{n} \mathbf{1}^T (X\mathbf{w} + b\mathbf{1} - \mathbf{y}) \quad (64)$$

**Q.E.D.**

**Answer 3.2**

For the test, we have setup tuning parameters as following:

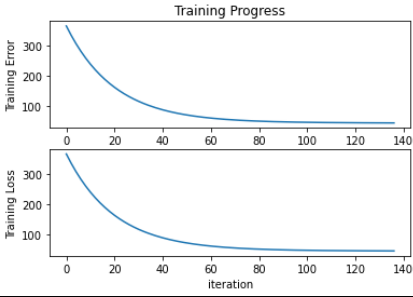
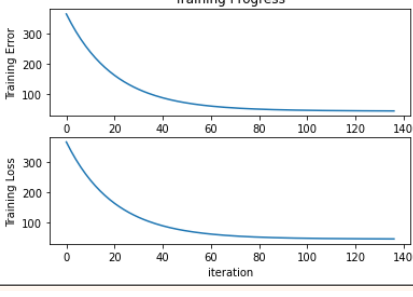
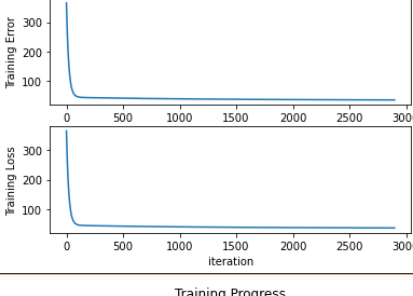
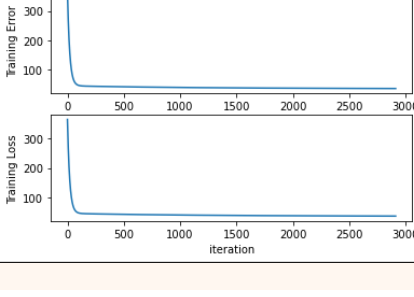
```

1 COMMON.MAX.PASS    = 10000
2 COMMON.ETA         = 1e-7 # >0
3 COMMON.TOLERANCE   = 5e-5 # >0
4 COMMON.ERR.TOL     = 35.8 # Disable: None

```

With the default termination criterion (which terminates when the slope of  $w$  below certain threshold), the algorithm seems to perform similarly as Trial 1 and 2 in the table below.

Hence, we modified the stopping criterion so that it terminates when training error reaches below the given tolerance. As Trial 3 and 4 in the table below shown, the trial with  $\lambda = 10$  takes longer passes with better performance on test dataset with lower test error. We do observe both trial 2 and 4 has better performance than without regularization term ( $\lambda = 0$ ).

Trial	$\lambda$	Passes	Training Progress	Training Error	Training Loss	Test Error
1	0	136		44.42208	44.42208	108.42167
2	10	136		44.42367	44.44802	108.39676
3	0	2899		35.79958	35.79958	49.89885
4	10	2911		35.79973	35.84981	49.88967

The complete algorithm is completed as shown below:

---

**Algorithm 2:** Completed Pseudocode Gradient descent for ridge regression.

---

**Input:**  $X \in \mathbb{R}^{n \times d}$ ,  $\mathbf{y} \in \mathbb{R}^n$ ,  $\mathbf{w}_0 = \mathbf{0}_d$ ,  $b_0 = 0$ ,  $\text{max\_pass} \in \mathbb{N}$ ,  $\eta > 0$ ,  $\text{tol} > 0$

**Output:**  $\mathbf{w}, b$

```

1 for  $t = 1, 2, \dots, \text{max\_pass}$  do
2    $\mathbf{w}_t \leftarrow \frac{1}{n} X^\top (X\mathbf{w} + b\mathbf{1} - \mathbf{y}) + 2\lambda \mathbf{w}$ 
3    $b_t \leftarrow \frac{1}{n} \mathbf{1}^\top (X\mathbf{w} + b\mathbf{1} - \mathbf{y})$ 
4   if  $\|\mathbf{w}_t - \mathbf{w}_{t-1}\| \leq \text{tol}$  then           // can use other stopping criteria
5     break
6  $\mathbf{w} \leftarrow \mathbf{w}_t$ ,  $b \leftarrow b_t$ 

```

---

The equivalent code implementation is as shown below:

```

1 def gradient_descent_ridge_regression_training(
2     X: List[List[float]],
3     y: List[float],
4     # Configuration with Default Settings
5     max_pass: int = 500,
6     eta: float = 0.3, # >0
7     tol: float = 0.1, # >0
8     lamb: float = 0, # Try \in {0,10}
9     error_tol: Optional[float] = None,
10 )-> [List[float], float, Dict]:
11     """
12     @param X: \in R^{nxd}
13     @param y: \in R^n
14     @param max_pass: \in N
15     @param eta: step size
16     @param tol: tolerance
17     @param lamb: regulation weight '\lambda'
18     """
19     XT = X
20     X = np.transpose(XT)
21     y = np.array(y)
22     [n, d] = np.shape(X)
23     w = np.zeros(d) # w = 0_d
24     b = 0
25     mistake = []
26     # optimization: pre-compile
27     div = 1/n
28     # logger
29     training_log = {
30         "t": [],
31         "w": [],
32         "b": [],
33         "training_error": [],
34         "training_loss": [],
35     }
36     # training
37     for t in range(0, max_pass): # max passes / iterations
38         pw = copy.deepcopy(w)
39         # update:
40         f_err = ( np.dot(X, w) + b - y ) # pred - y
41         dw = div * np.dot(XT, f_err) + 2 * lamb * w
42         db = div * np.sum(f_err) # 1^T f_err
43         w = w - eta * dw
44         b = b - eta * db
45         # compute loss and error:
46         error = div / 2 * (np.linalg.norm(f_err) ** 2)
47         loss = error + lamb * (np.linalg.norm(w) ** 2)
48         # log progress:
49         training_log["t"].append(t)
50         training_log["w"].append(w)
51         training_log["b"].append(b)

```

```
52     training_log["training_error"].append(error)
53     training_log["training_loss"].append(loss)
54     # stopping criteria:
55     if error_tol is None:
56         if np.linalg.norm(pw - w) <= tol: # can use other stopping criteria
57             break # STOPPING
58     else:
59         if error <= error_tol:
60             break # STOPPING
61
62     return w, b, training_log
63
```

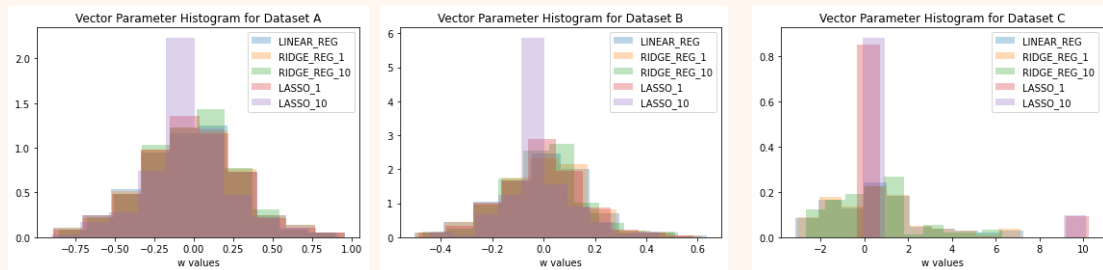
**Answer 3.3**

To thoroughly compare the lasso and ridge regression methods, we will have to divide these alpha parameters (1 and 10) by  $n$  for lasso in scikit-learn. This is mainly caused by the difference in objective function in scikit-learn. Specifically, the objective for the lasso incorporates the sample size  $n$  as the denominator of the mean squared error [1], whereas, the objective for the ridge regression does not take into account this term [2]. Hence, the regulation term  $\alpha$  shall also divide by the sample size  $n$  in the lasso to match up with the ridge regression, so that both algorithm has similar regulation effects.

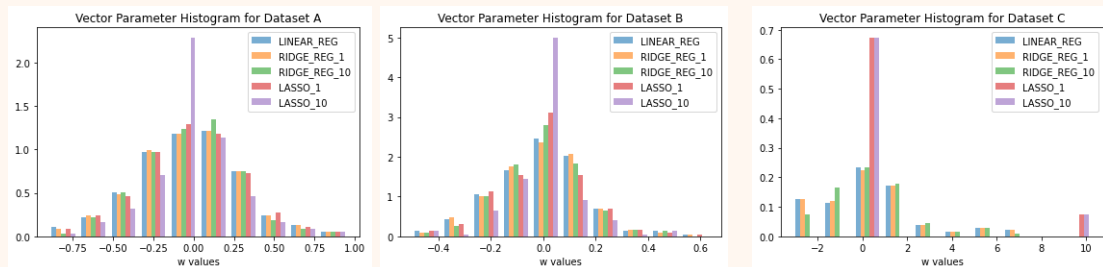
Test results are tabulated here:

Methods	Dataset A	Dataset B	Dataset C
Linear	MSE = 3.24740	MSE = 2.74268	MSE = 506.37271
Ridge 1	MSE = 3.13939	MSE = 2.61621	MSE = 505.27731
Ridge 10	MSE = 2.77803	MSE = 2.05971	MSE = 515.89174
Lasso 1	MSE = 3.02040	MSE = 2.26517	MSE = 1.87747
Lasso 10	MSE = 3.60263	MSE = 1.80967	MSE = 1.35256

(Hist.)



(Hist.)



It appears that "Ridge 10" performs the best in dataset A, and "Lasso 10" performed the best for both dataset B and C. The classic ridge regression uses L2 regularization, whereas the Lasso uses the L1 regularization. Hence, the L2 norm will place the outside penalty on large components of the weight vector, hence, the histogram of L2 (Ridge Regression) would be more evenly distributed across many features. L1 norm leads to a concentrate weights on a small set of features by clearing other weights to zero, which leads to a concentrated and spiky histogram with respect to ridge regression model. The L1 behaviour is also known as feature extraction.

As we can see from the datasets, as the alpha increases, the Lasso becomes more concentrated. In dataset C, the "Lasso 10" and "Lasso 1" has some large weights at 10, where classical regression and ridge regression failed to register. For this particular dataset C, Lasso performed the best, hence, we may conclude the dataset C is quite sparse and complex. In the case of sparse data, rigid and classical would be normalized. Hence, we may see the rigid regression with high L2 regularization make worser prediction on the dataset. Whereas, the L1 regularization in Lasso helps to compensate the sparsity and complexity. It is also possible that the ridge regression and linear modelling over-fit the training dataset, while the lasso will reduce this over-fit by feature extraction.

For dataset A, the Ridge with large coefficient performed the best, whereas the lasso with the large coefficient performed the worst. This indicates the dataset A has a significant multi-collinearity. The ridge is capable to shrink the model complexity and multi-collinearity. Whereas, the lasso would reduce the complexity by

feature extraction but suffers if the data has a significant multi-collinearity. The lasso would select one of the dominant feature to enhance and attenuates all other features. Hence, the lasso with high regulation factor in this case performed the worst, due to feature losses (or over-simplified).

For dataset B, it is observed that the ridge perform better as the coefficient increases, and the lasso with a large gain perform the best overall. This indicates the dataset has a small degree of complexity.

As per requested on Piazza to think about how we generate the response  $Y$  from the true weight distribution  $\hat{W}$ , and based on the above observation and analysis, we may suspect the response is generated from three different  $\hat{W}$  with different distribution model:

The dataset A might generate the weight based on a Gaussian distribution centered at zero mean, resulting a smooth curved histogram centered at  $w = 0$ . In addition, there are additive noises centered at zero mean with small variance introduced when computing the response. As a result, the ridge performs the best, since there exists multiple features that are correlated at the same time.

The dataset B might be based on a double exponential distribution centered at zero, since the tip of the predicted weight appeared to be quite spiky, and there exists a significant feature resulting a better performance in Lasso.

The dataset C might be based on a double exponential distribution centered at zero as well, but with an additive noise centered at 10, resulting a sparse response  $Y$ .

**Exercise 4: An Alternative to Least Squares? (3 pts)**

Suppose we are in a setting with a dataset  $X \in \mathbb{R}^{n \times d}$ , and labels are generated according to  $Y = XW + \varepsilon$ , where  $W \in \mathbb{R}^d$ ,  $Y \in \mathbb{R}^n$ , and  $\varepsilon \in \mathbb{R}^n$  is a random vector, where all entries are independent random variables with 0 mean and variance  $\sigma^2$  (you can imagine Gaussian, but it isn't necessary). As we saw in class, the least-squares solution  $\hat{W}$  can be written as  $\hat{W} = (X^T X)^{-1} X^T Y$  -- this is a linear transformation of the response vector  $Y$ . Consider some *different* linear transformation  $((X^T X)^{-1} X^T + N) Y$ , where  $N \in \mathbb{R}^{d \times n}$  is a non-zero matrix.

1. (1 pt) Show that the expected value of this linear transformation is  $(I_d + NX)W$ . Conclude that its expected value is  $W$  if and only if  $NX = 0$ . (1 pt)

Ans: [Answer 4.1](#)

2. (2 pts) Compute the covariance matrix of this linear transformation when  $NX = 0$ , and show that it is equal to  $\sigma^2(X^T X)^{-1} + \sigma^2 N N^T$ . Since the former term is the covariance of the least squares solution<sup>a</sup> and the latter matrix is positive semi-definite, this implies that this alternative estimator only increased the variance of our estimate.

Ans: [Answer 4.2](#)

<sup>a</sup>Verify this for yourself, but no need to submit it.

**Answer 4.1**

Let us name this linear transformation as  $\hat{W}$ .

$$E[\hat{W}] = E[(X^T X)^{-1} X^T + N] Y \quad (65)$$

$$= E[(X^T X)^{-1} X^T + N] (XW + \varepsilon) \quad (66)$$

$$= E[(X^T X)^{-1} X^T + N] XW + ((X^T X)^{-1} X^T + N) E[\varepsilon] \quad (67)$$

$$= E[(X^T X)^{-1} X^T + N] XW + E[(X^T X)^{-1} X^T + N] E[\varepsilon] \quad (68)$$

$$= E\left[\left(\cancel{(X^T X)^{-1} X^T} \xrightarrow{I_d} I_d + N\right) W\right] + E[(X^T X)^{-1} X^T + N] \cancel{E[\varepsilon]}^0 \quad (69)$$

$$= E[(I_d + NX)W] \quad (70)$$

$$= (I_d + NX)W \quad (71)$$

We may also prove the expected value is  $W$  if and only if  $NX = 0$ :

$$E[\hat{W}] = (I_d + NX)W \quad (72)$$

$$= W + NXW \quad (73)$$

$$E[\hat{W}] - W = NXW \quad (74)$$

$$\text{If } NXW \neq 0 \Rightarrow E[\hat{W}] - W \neq 0 \Rightarrow E[\hat{W}] \neq W \quad (75)$$

$$\text{If } NXW = 0 \Rightarrow E[\hat{W}] - W = 0 \Rightarrow E[\hat{W}] = W \quad (76)$$

$$\therefore E[\hat{W}] = W \text{ iff } NXW = 0 \quad (77)$$

**Q.E.D.**



**Answer 4.2**

Similar to Answer 4.1, we may derive  $\hat{W}$  and  $(\hat{W} - W)$ :

$$\therefore \hat{W} = ((X^T X)^{-1} X^T + N) XW + ((X^T X)^{-1} X^T + N) \varepsilon \quad (78)$$

$$= \cancel{(X^T X)^{-1} X^T} \overset{I_d}{XW} + \cancel{NXW} \overset{0}{+} (X^T X)^{-1} X^T \varepsilon + N\varepsilon \quad (79)$$

$$= W + (X^T X)^{-1} X^T \varepsilon + N\varepsilon \quad (80)$$

$$\therefore \hat{W} - W = (X^T X)^{-1} X^T \varepsilon + N\varepsilon \quad (81)$$

We may now compute the covariance matrix:

$$\Sigma = E[(\hat{W} - W)(\hat{W} - W)^T] \quad (82)$$

$$= E\left[\left((X^T X)^{-1} X^T \varepsilon + N\varepsilon\right)\left((X^T X)^{-1} X^T \varepsilon + N\varepsilon\right)^T\right] \quad (83)$$

$$= E\left[\left((X^T X)^{-1} X^T \varepsilon + N\varepsilon\right)\left(\varepsilon^T X (X^T X)^{-1} + \varepsilon^T N^T\right)\right] \quad (84)$$

$$= E\left[(X^T X)^{-1} X^T \varepsilon \varepsilon^T X (X^T X)^{-1} + N\varepsilon \varepsilon^T X (X^T X)^{-1} + \varepsilon^T N^T (X^T X)^{-1} X^T + N\varepsilon \varepsilon^T N^T\right] \quad (85)$$

$$= E\left[(X^T X)^{-1} (X^T X) \varepsilon \varepsilon^T (X^T X)^{-1} + NX \varepsilon \varepsilon^T (X^T X)^{-1} + N^T X^T \varepsilon^T (X^T X)^{-1} + N\varepsilon \varepsilon^T N^T\right] \quad (86)$$

$$= E\left[\cancel{(X^T X)^{-1} (X^T X)} \overset{I_d}{\varepsilon \varepsilon^T} (X^T X)^{-1} + \cancel{NX} \overset{0}{\varepsilon \varepsilon^T} (X^T X)^{-1} + \cancel{(N^T X^T)} \overset{0}{\varepsilon^T} (X^T X)^{-1} + N\varepsilon \varepsilon^T N^T\right] \quad (87)$$

$$= E\left[\varepsilon \varepsilon^T (X^T X)^{-1} + N\varepsilon \varepsilon^T N^T\right] \quad (88)$$

$$= E\left[\varepsilon \varepsilon^T (X^T X)^{-1}\right] + E\left[N\varepsilon \varepsilon^T N^T\right] \quad (89)$$

$$= E\left[\varepsilon \varepsilon^T\right] (X^T X)^{-1} + E\left[\varepsilon \varepsilon^T\right] NN^T \quad (90)$$

$$= \sigma^2 (X^T X)^{-1} + \sigma^2 NN^T \quad (91)$$

**Q.E.D.**

**Remark 4.4:  $\varepsilon$  Term**

$E[\varepsilon \varepsilon^T] = \sigma^2 I_d$ : The covariance of additive noise term is the variance, which is  $\sigma^2$  as provided

$E[\varepsilon] = 0$ : The expectation of the additive noise term is the mean, which is 0 as provided

**Exercise 5: Sample Statistics (2 pts)**

1. (1 pt) Suppose there is a dataset  $x_1, \dots, x_n$  sampled from a distribution with mean  $\mu$  and variance  $\sigma^2$ . Compute the expected value of the sample mean:  $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ . Describe any modifications that might be required to make the expected value  $\mu$  (recall that  $\mu$  and  $\sigma^2$  are unknown).

Ans: [Answer 5.1](#)

2. (1 pt) Suppose there is a dataset  $x_1, \dots, x_n$  sampled from a distribution with mean  $\mu$  and variance  $\sigma^2$ . Compute the expected value of the sample variance:  $\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$ , where  $\bar{x}$  is the sample mean from the previous part. Describe any modifications that might be required to make the expected value  $\sigma^2$  (recall that  $\mu$  and  $\sigma^2$  are unknown).

Ans: [Answer 5.2](#)

**Answer 5.1**

$$E[\bar{x}] = E \left[ \frac{1}{n} \sum_{i=1}^n x_i \right] \quad (92)$$

$$= \frac{1}{n} \sum_{i=1}^n E[x_i] \quad (93)$$

$$\because E[x_i] = \mu \forall i \quad (94)$$

$$= \frac{1}{n} \sum_{i=1}^n \mu \quad (95)$$

$$= \frac{1}{n} \cdot n\mu \quad (96)$$

$$= \mu \quad (97)$$

Hence:

$$\mu = E[\bar{x}] = E \left[ \frac{1}{n} \sum_{i=1}^n x_i \right] \quad (98)$$

Hence, the expected value of the sample mean is  $\mu$ . No additional modification is required.

**Q.E.D.**

**Answer 5.2**

$$E \left[ \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \right] = E \left[ \frac{1}{n} \sum_{i=1}^n (x_i^2 - 2x_i\bar{x} + \bar{x}^2) \right] \quad (99)$$

$$= E \left[ \frac{1}{n} \left( \sum_{i=1}^n x_i^2 - 2 \sum_{i=1}^n x_i\bar{x} + \sum_{i=1}^n \bar{x}^2 \right) \right] \quad (100)$$

$$\therefore \sum_{i=1}^n x_i = n\bar{x} \leftarrow \text{as stated in Answer 5.1} \quad (101)$$

$$\therefore = E \left[ \frac{1}{n} \left( \sum_{i=1}^n x_i^2 - 2n\bar{x}\bar{x} + n\bar{x}^2 \right) \right] \quad (102)$$

$$= E \left[ \frac{1}{n} \left( \sum_{i=1}^n x_i^2 - n\bar{x}^2 \right) \right] \quad (103)$$

$$= \frac{1}{n} \left( \sum_{i=1}^n E[x_i^2] - nE[\bar{x}^2] \right) \quad (104)$$

$$\therefore \text{Assume it is independent} \quad (105)$$

$$\therefore E[x_i^2] = \sigma^2 + \mu^2 \quad E[\bar{x}^2] = \frac{\sigma^2}{n} + \mu^2 \quad (106)$$

$$\therefore = \frac{1}{n} \left( \sum_{i=1}^n (\sigma^2 + \mu^2) - n \left( \frac{\sigma^2}{n} + \mu^2 \right) \right) \quad (107)$$

$$= \frac{1}{n} \left( n(\sigma^2 + \mu^2) - n \left( \frac{\sigma^2}{n} + \mu^2 \right) \right) \quad (108)$$

$$= \frac{1}{n} \left( n(\sigma^2 + \mu^2 - \frac{\sigma^2}{n} - \mu^2) \right) \quad (109)$$

$$= \frac{n-1}{n} \sigma^2 \quad (110)$$

$$(111)$$

Hence, we need modification:

$$\sigma^2 = \frac{n}{n-1} E \left[ \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \right] \quad (112)$$

The expected variance  $\sigma^2$  shall be multiplying  $\frac{n}{n-1}$  with the sample variance.

In the special case of  $n \rightarrow \infty$  (or simply large enough), we may assume  $\sigma^2 = \lim_{n \rightarrow \infty} \frac{n}{n-1} E \left[ \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \right] = E \left[ \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \right]$ .

**Q.E.D.**