6.036: Midterm, Spring 2019

Solutions

- This is a closed book exam. One page (8 1/2 in. by 11 in.) of notes, front and back, is permitted. Calculators are not permitted.
- The problems are not necessarily in any order of difficulty.
- Record all your answers in the places provided. If you run out of room for an answer, continue on a blank page and mark it clearly.
- If a question seems vague or under-specified to you, make an assumption, write it down, and solve the problem given your assumption.
- If you absolutely *have* to ask a question, come to the front.
- **Write your name on every page.**

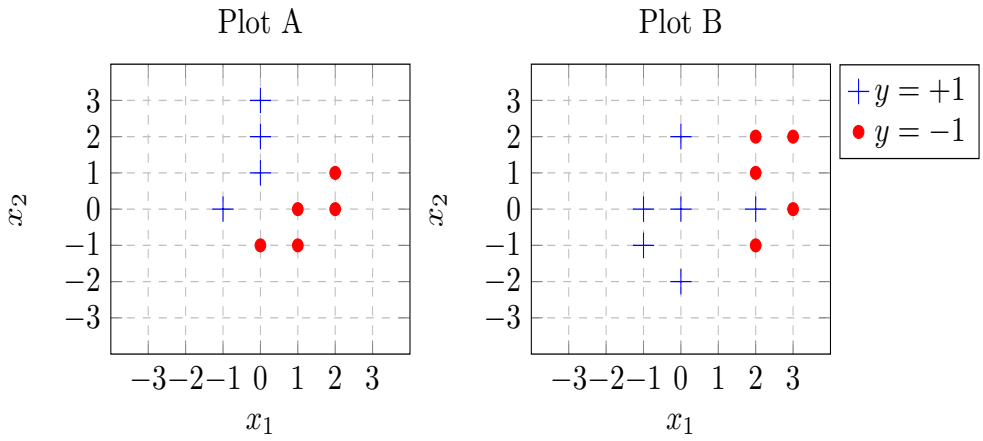
Name: _____ Athena ID: _____

Question	Points	Score
1	14	
2	17	
3	15	
4	10	
5	14	
6	15	
7	15	
Total:	100	

Name: _____

Linear Classifiers

1. (14 points) In the plots below, we give you 2D points with +1 and −1 labels.



Answer the following questions for both plot A and plot B:

(a) Using a linear separator $h(p; \theta, \theta_0) = sign(\theta^\top p + \theta_0)$, what is the minimum possible number of misclassified points?

Plot A:

Solution: 0. A separator which achieves this is given in the solution to part (b).

Plot B:

Solution: 1. A separator which achieves this is given in the solution to part (b).

(b) What are the values of $\theta \in \mathbb{R}^2$ and $\theta_0 \in \mathbb{R}$ that define your separator?

Plot A:

Solution: $\theta = [-1, 1]^T$, $\theta_0 = 0$

Plot B:

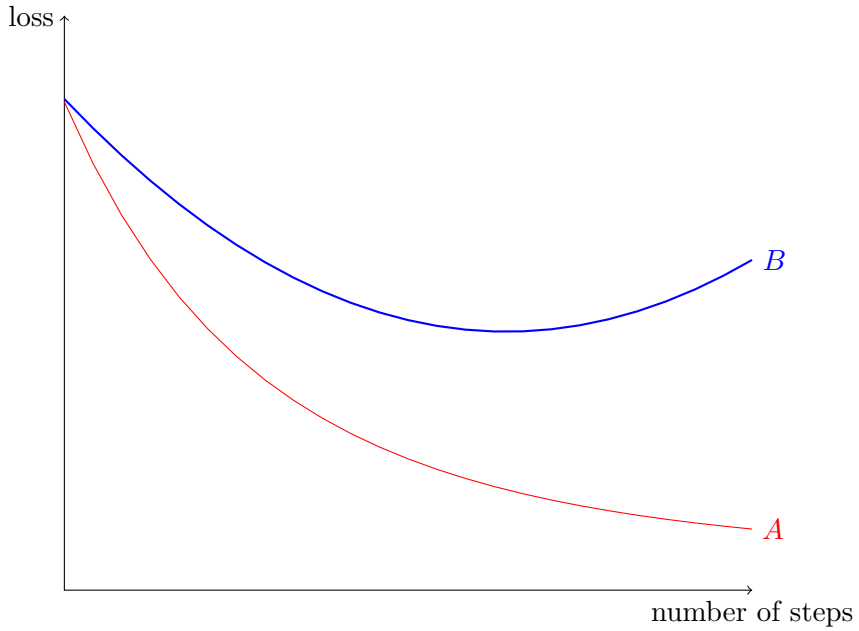
Solution: $\theta = [-1, 0]^T$, $\theta_0 = 1$

(c) For a given point p, what does $\frac{\theta^\top p + \theta_0}{\|\theta\|}$ intuitively represent?

Solution: Signed distance from the point p to the separating line.

Name: _____

Consider the following plot from the previous classification task. The two curves show the train and test error vs. the number of steps in the optimization algorithm.



(d) Assign the appropriate labels:
Test error (select one): ◯ A √ **B**
Train error (select one): √ **A** ◯ B

> **Solution:** The training error will always be decreasing (given that our step size is small enough) as our optimization algorithm is directly minimizing it. Our test error will start increasing after a certain point due to overfitting.

(e) Which of the following options can improve the final performance of the trained classifier on the test data set? Note: augmentation of a data set refers to taking the existing data set and adding many points which are slightly perturbed versions of the original points. Select all that apply.

√ **A. Augment the training data set and retrain the classifier.**

◯ B. Augment the test data set and retrain the classifier.

√ **C. Terminate the training process earlier.**

√ **D. Add a penalty on the magnitude of the parameter values and retrain the classifier.**

Solution: Augmenting the training data set can help improve generalization, as the requirement of correctly classifying the noisy, new points tends to keep the new classifier boundary away from the original points. As the margin tends to increase in size, generalization improves.

Augmenting the test set does not change the classifier that was trained– it only gives you a less accurate idea of what the generalization error will be.

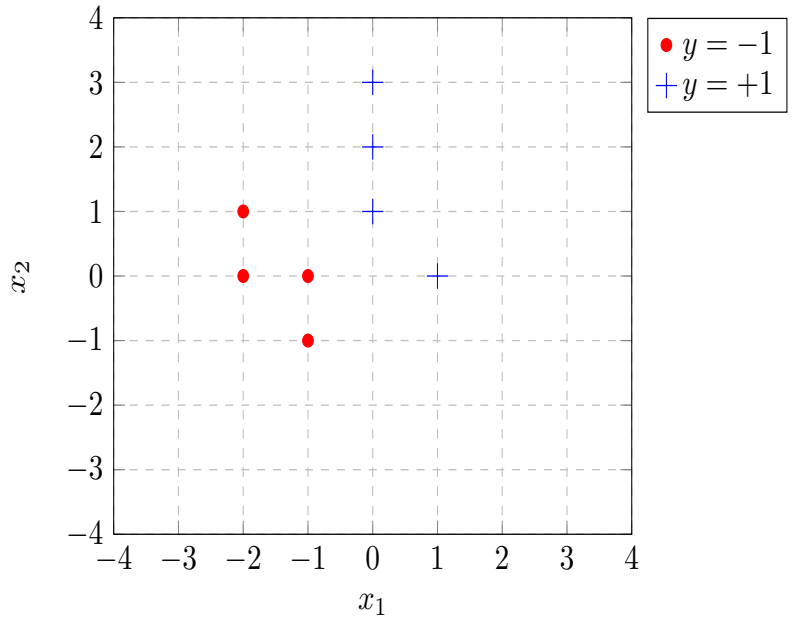
Terminating training early can prevent overfitting (e.g. stopping near the minimum of the curve B would have been ideal in order to minimize test error).

Regularizing our parameters tends to prevent overfitting, as well.

Name: _____

Perceptron Algorithm

2. (17 points) Alice plans to apply the Perceptron algorithm to solve a classification problem on the data set shown in the figure below. For this problem, we will only consider linear separators that pass through the origin.



(a) What is the theoretical upper bound on the number of steps for the Perceptron algorithm to find the linear separator on this data set?

Solution: 18. The Perceptron convergence theorem gives that an upper bound is $\left(\frac{R}{\gamma}\right)^2$, where R is the maximum distance from the origin to a point in the data set, and γ is the minimum distance from some valid separator to a point in the data set. Here, R is 3, from the positive example at $(0, 3)$, and the best γ is $\frac{\sqrt{2}}{2}$ from looking at the maximum margin separator, $x_1 + x_2 = 0$. The answer is then 18.

(b) Alice suggests a feature transformation of the form: $\phi((x_1, x_2)) = (\alpha x_1, x_2)$. Is there a value of α that would reduce the upper bound on the number of mistakes made by the Perceptron algorithm? If so, provide one such value. If not, explain why not.

Solution: Yes, for $|\alpha| \in (1, \infty)$. When the x_1 coordinate is scaled by a factor slightly greater than 1, the value of R in the above does not change (as the point $(0, 3)$ remains fixed upon scaling), but γ does increase slightly (intuitively, we can tilt the separator $x_1 + x_2 = 0$ slightly counterclockwise to increase the margin). When we use a larger scaling factor, while R does increase the margin increases enough that the upper bound on the number of mistakes still decreases. Note that students were only required to provided one value of α that works, not the entire valid range.

First note that $0 < \alpha < 1$ will not work. This is because for this range, R stays constant at 3 while all of the points get closer together, decreasing the margin, thus decreasing γ, and increasing the perceptron convergence upper bound.

For $\alpha > 1$, we need to consider 2 cases:

For $\alpha \in (1, \sqrt{2}]$, the point furthest from the origin is $(0, 3)$, so that $R = 3$. For values of α in this range, the max margin separator is defined by the line midway through the the line passing through the red points at $(-2\alpha, 1)$ and $(-\alpha, 0)$ and the line through the blue points at $(0, 1)$ and $(\alpha, 0)$ (as their slopes are equal). As these lines have respective equations $y = -\frac{x}{\alpha} - 1$ and $y = -\frac{x}{\alpha} + 1$, the maximum margin separator is their average: $y = -\frac{x}{\alpha}$, or $x + \alpha y = 0$. The distance from $(0, 1)$ (or any of these points) to this separator is $\frac{\alpha}{\sqrt{\alpha^2+1}}$. Thus, the mistake bound is $\left(\frac{R}{\gamma}\right)^2 = \left(\frac{3\sqrt{1+\alpha^2}}{\alpha}\right)^2 = 9\left(1 + \frac{1}{\alpha^2}\right)$ which is less than 18 for any α in this interval.

For $\alpha \in (\sqrt{2}, \infty)$, the point furthest from the origin is $(-2\alpha, 1)$, so that $R = \sqrt{1 + 4\alpha^2}$. We must consider two subcases, when the max-margin separator is the same diagonal line as in the previous case, and when the separator is the vertical line halfway between the red point at $(-\alpha, 0)$ and the blue point at $(0, 1)$. The first separator has margin $\frac{\alpha}{\sqrt{\alpha^2+1}}$, while the second has margin $\frac{\alpha}{2}$. Equating these, we see that the separator is the diagonal line for $\alpha < \sqrt{3}$, and is the vertical line for $\alpha > \sqrt{3}$.

Thus, for $\alpha \in (\sqrt{2}, \sqrt{3}]$, the mistake bound is $\left(\frac{R}{\gamma}\right)^2 = \left(\frac{\sqrt{1+4\alpha^2}}{\frac{\alpha}{\sqrt{\alpha^2+1}}}\right)^2 = \left(\frac{\sqrt{1+4\alpha^2} \cdot \sqrt{1+\alpha^2}}{\alpha}\right)^2$, which is less than 18 for values of α in this range.

For $\alpha \in (\sqrt{3}, \infty)$, the mistake bound is $\left(\frac{R}{\gamma}\right)^2 = \left(\frac{\sqrt{1+4\alpha^2}}{\frac{\alpha}{2}}\right)^2 = 16 + \frac{4}{\alpha^2}$, which is also less than 18 for values of α in this range.

So, any $\alpha > 1$ reduces our upper bound.

Finally, changing the sign of α does not affect the upper bound, such that $\alpha \in (-\infty, -1)$ is also valid.

(c) Will scaling both coordinates uniformly, i.e., $\phi((x_1, x_2)) = (\beta x_1, \beta x_2)$, decrease the bound on the number of mistakes? Explain.

Solution: No. Scaling both coordinates uniformly scales both R and γ by the same amount, which does not change the upper bound.

(d) When the point is classified incorrectly, the algorithm updates θ. Is the point guaranteed to be classified correctly after the update is made? Explain.

Solution: No. If θ has a very large magnitude, and the incorrectly classified point has small norm but is on the opposite side of the decision boundary, the update $\theta \leftarrow \theta + yx$ will barely change the orientation of θ, leaving the point still misclassified.

(e) A separator is trained using Perceptron with the points $i = 1, ..., N$ in the data set. Write the expression for the general final separator in terms of $\{x^{(i)}\}$, their labels $\{y^{(i)}\}$, and the number of mistakes $\{n_i\}$ that Perceptron made on each of the points.

Solution: $\theta = \sum_{i=1}^{N} n_i y^{(i)} x^{(i)}$
This follows from the form of the Perceptron update: we add $y^{(i)}x^{(i)}$ to our separator every time that we make a mistake on example i. Thus, the end result will be the sum of these $y^{(i)}x^{(i)}$, weighted by the number of times that we made a mistake on example i.

(f) Is this algorithm guaranteed to find the classifier with maximum margin?
◯ yes √ **no**

Solution: No. The Perceptron algorithm can only be guaranteed to find a perfect separator– the margin of the separator that it finds can be arbitrarily bad.

Name: _____

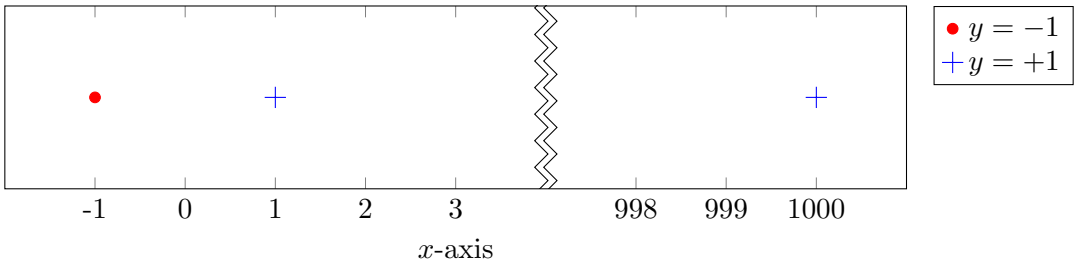
Margin Maximization

3. (15 points) In this problem, we will consider using linear regression for classification problems, and its relationship to margin maximization.

   Student Chris has a data set of samples $\{(x_i, y_i)\}$ with $x_i \in \mathbb{R}$ and classes $y_i \in \{-1, +1\}$. Chris mixed up classification and regression, and ended up computing a linear regression $\hat{y} = \theta^\top x + \theta_0$ instead of a classifier. Having made this mistake, Chris figures out that it is possible to convert the regression into a classifier by taking its sign, i.e. $\hat{y} = \text{sign}(\theta^\top x + \theta_0)$ where:

$$\text{sign}(z) = \begin{cases} 1 & \text{when} \quad z > 0 \\ -1 & \text{when} \quad z \leq 0 \end{cases}$$

   Consider the data set $x = [[-1], [1], [1000]]^\top$ with labels $y = [-1, 1, 1]^\top$.



   (a) What is its maximum margin separator: θ, θ_0 ?

   Solution: $\theta = 1$, $\theta_0 = 0$

   (b) Chris uses a loss function $L(g, a) = (g - a)^2$ that takes g (guess) and a (actual) as parameters. Specify Chris's objective function for the linear regression problem using θ, θ_0, x_i, y_i:

   Solution: Chris's objective is of the form $\sum_i (\theta x_i + \theta_0 - y_i)^2$.

With this data set, it turns out that the linear regression solution is approximately

$$\hat{y} = 0.001x - 0.000999$$

.

(c) What then is the decision boundary defined by Chris's classifier?

> **Solution:** Chris's classifier has the boundary defined by the regression line. This leads to a separation at $\hat{y} = 0 = 0.001x - 0.000999$, i.e. at $x = 0.999$.

(d) Does the classifier correctly classify all of the points in the training set?

√ **Yes.**

◯ No.

(e) Can you add another point to the data set so that the data set is still linearly separable, but so that using linear regression to train it would result in a classifier that mis-classifies one or more points? If yes, specify such a data point. If no, explain why not.

> **Solution:** Yes. For example, a point at $x = 0.9$ with label $y = -1$ will produce a regression line that misclassifies the previous point at $x = 1$, i.e., $\hat{y} = 0.00134x - 0.3342$. Indeed, almost any additional $y = -1$ point nearby and to the left of $x = 1$ will shift the regression line so that the point at $x = 1$ is misclassified.
>
> The full range is: any negative point at $a \in (-\infty, -1004] \cup [-996, 1)$ or any positive point at $a \in (-1, 912], [1092, \infty)$.
>
> There are 2 cases. One is when we add a negative datapoint at $a < 1$. Our regression problem can then be posed as $\min_x \|Ax - b\|^2$, where:
>
> $$A = \begin{bmatrix} -1 & 1 \\ 1 & 1 \\ 1000 & 1 \\ a & 1 \end{bmatrix}$$
>
> $$x = \begin{bmatrix} \theta \\ \theta_0 \end{bmatrix}$$
>
> $$b = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$
>
> The solution is then $x = (A^T A)^{-1} A^T b$. Multiplying out, we find:
>
> $$(A^T A) = \begin{bmatrix} 10^6 + 2 + a^2 & a + 1000 \\ a + 1000 & 4 \end{bmatrix}$$
>
> Inverting,
>
> $$(A^T A)^{-1} = \frac{1}{\det(A^T A)} \begin{bmatrix} 4 & -a - 1000 \\ -a - 1000 & 10^6 + 2 + a^2 \end{bmatrix}$$

In addition,

$$A^T b = \begin{bmatrix} 1002 - a \\ 0 \end{bmatrix}$$

This then gives:

$$\begin{bmatrix} \theta \\ \theta_0 \end{bmatrix} = (A^T A)^{-1} A^T b = \frac{1}{\det(A^T A)} \begin{bmatrix} 4(1002 - a) \\ (-a - 1000)(1002 - a) \end{bmatrix}$$

As we are using the output of the regression to perform classification, only the sign of a prediction matters. As $A^T A$ is positive definite, it has positive determinant, and as we are considering $a < 1$, $1002 - a > 0$. We may thus drop these overall factors when looking at the sign of the prediction. The sign is then:

$$\hat{y} = \text{sign}(4x - (a + 1000))$$

Note that the decision boundary is $x_b = \frac{a+1000}{4}$. This gives correct predictions on all of the points iff $-1, a < x_b < 1$. This is true iff $a \in [-1004, -996]$.

The other case is when we add a positive datapoint at $a > -1$. The setup is the same as before, except that we instead have:

$$b = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

In this case,

$$A^T b = \begin{bmatrix} 1002 + a \\ 2 \end{bmatrix}$$

And then:

$$\begin{bmatrix} \theta \\ \theta_0 \end{bmatrix} = (A^T A)^{-1} A^T b = \frac{1}{\det(A^T A)} \begin{bmatrix} 4(1002 + a) - 2(a + 1000) \\ (-a - 1000)(1002 + a) + 2(10^6 + 2 + a^2) \end{bmatrix}$$

Then, the sign of the predictor is:

$$\hat{y} = \text{sign}\left((2a + 2008)x + (-a - 1000)(1002 + a) + 2(10^6 + 2 + a^2)\right)$$

$$= \text{sign}\left(x + \frac{(-a - 1000)(1002 + a) + 2(10^6 + 2 + a^2)}{(2a + 2008)}\right)$$

The decision boundary is then:

$$x_b = \frac{(a + 1000)(1002 + a) - 2(10^6 + 2 + a^2)}{(2a + 2008)}$$

We have correct predictions at all data points iff $-1 < x_b < 1, a$. We have $x_b > -1$ only for $a \in [1002 - 2\sqrt{2002}, 1002 + 2\sqrt{2002}] \approx [912, 1092]$. The other two inequalities are always satisfied.

(f) Would you expect this classifier to generalize well? Explain.

Solution: This classifier is likely not to generalize well since its decision boundary is disproportionately close to the sample $(1, 1)$.

(g) Would you expect a maximum margin classifier to generalize better than Chris's? Explain.

Solution: Yes. It would not longer be biased towards the points with positive labels.

Name: _____

Model Evaluation

4. (10 points) Lisa trains models for classification problems. She is provided with different image data sets (e.g., trains, people, cars, cats, dogs) by Snapbook. Each data set has both positive and negative examples. In fact, Snapbook provides Lisa only a fraction of each data set, the remainder is left for internal Snapbook testing. Lisa trains a separate model on each data set. She measures model training accuracy, and she estimates test accuracy using cross-validation. For each model, Snapbook measures the accuracy of the model on the data that was held out (not provided to Lisa). These experiments yield the following results:

	training accuracy	cross-validation accuracy	held-out tests accuracy
data set 1	52%	54%	51%
data set 2	97%	71%	70%
data set 3	93%	92%	55%
data set 4	91%	91%	89%
data set 5	50%	53%	70%

For which data set(s):

(a) Lisa's model is overfitting (check all that apply):
◯ data set 1 √ **data set 2** ◯ data set 3 ◯ data set 4 ◯ data set 5

Solution: In data set 2, the training accuracy is extremely high, while the cross-validation and test accuracy are significantly lower, pointing to overfitting. None of the other data sets show this significant of a difference.

(b) It is likely that more training data drawn from the same distribution would improve the quality of the held-out accuracy (check all that apply):
◯ data set 1 √ **data set 2** ◯ data set 3 ◯ data set 4 ◯ data set 5

Solution: As above, Lisa's model for data set 2 is overfitting, and so having more training data can mitigate this issue.
Data set 3 also has a held-out accuracy that is much lower than the training accuracy, but the distinction is that here, the cross-validation accuracy is high. This implies that more training data would not help in generalization, and that instead there is a more fundamental problem with Lisa's training data not being drawn from the same distribution as the test data.

(c) Lisa's hypothesis class might not be expressive enough (check all that apply):
√ **data set 1** ◯ data set 2 ◯ data set 3 ◯ data set 4 √ **data set 5**

Solution: Low training and cross-validation accuracies can point to a hypothesis class not being expressive enough. This occurs for data sets 1 and 5.

(d) Held-out data set is not likely from the same distribution as Lisa's (check all that apply):
◯ data set 1 ◯ data set 2 √ **data set 3** ◯ data set 4 √ **data set 5**

Solution: This is usually the case when the cross-validation accuracy is very different from the held-out test accuracy. This occurs for data sets 3 and 5.

Name: _____

Learning as Optimization

5. (14 points) Ben develops a new hypothesis class: $h(x; w_1, w_2) = w_1 x_1 + w_1 x_1^2 + w_2 x_2 + w_2 x_2^2$, where $x = (x_1, x_2)$. He plans to use it for a regression problem on the data set $S_n = \{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$.

   (a) Ben will use batch gradient descent to compute model parameters w_1, w_2. His loss function is mean squared error (MSE). Derive an update rule for w_1 given the learning rate η.

   Solution:
   Assume, for simplicity, that the batch size is equal to n.

$$L(g, a) = \frac{1}{n} \sum_{i=1}^{n} (w_1 x_1^{(i)} + w_1 x_1^{(i)^2} + w_2 x_2^{(i)} + w_2 x_2^{(i)2} - y^{(i)})^2$$

$$\frac{\delta L}{\delta w_1} = \frac{2}{n} \sum_{i=1}^{n} (w_1 x_1^{(i)} + w_1 x_1^{(i)2} + w_2 x_2^{(i)} + w_2 x_2^{(i)2} - y^{(i)})(x_1^{(i)} + x_1^{(i)2})$$

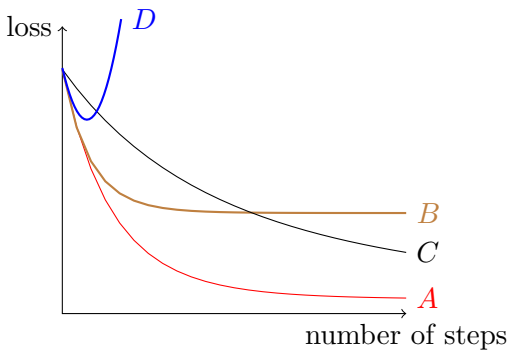
$$w_1 := w_1 - \frac{2\eta}{n} \sum_{i=1}^{n} (w_1 x_1^{(i)} + w_1 x_1^{(i)2} + w_2 x_2^{(i)} + w_2 x_2^{(i)2} - y^{(i)})(x_1^{(i)} + x_1^{(i)2})$$

   (b) Describe the shape of the MSE as a function of w_1 and w_2. How many minima will it have? Assume that the data set S_n is fixed.

   Solution: Paraboloid (all values positive). Single minimum.

Name: _____

(c) Ben tries different settings of the learning rate η. Depending on the setting he obtains different behavior of the gradient descent algorithm. Match each plot (A,B,C,D) to the best fitting description (assume MSE loss).



Learning rate too low (select one):
◯ A ◯ B √ **C** ◯ D

Learning rate about right (select one):
√ **A** ◯ B ◯ C ◯ D

Learning rate too high (select one):
◯ A √ **B** ◯ C ◯ D

Learning rate much too high (select one):
◯ A ◯ B ◯ C √ **D**

Solution: A low learning rate leads to slow decay of the loss– this occurs for C. A good learning rate leads to moderately quick decay to low loss, as in A. A high learning rate can prevent gradient descent from reaching the global minimum of the objective, and can cause it to oscillate between parameter values that give a higher loss value, as in B. A very high learning rate can cause gradient descent to diverge, as in D.

(d) Alyssa suggests using a mean absolute error, instead, defined by:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y^{(i)} - h(x^{(i)}, w_1, w_2)\right|$$

What could be an advantage of this approach?

Solution: Robustness to outlier (or high noise). The MSE loss over-penalizes for the samples that have large errors comparing to the smaller ones. The MAE loss equally penalizes all the samples, because the gradient slope is always 1 (except at 0), which favors more majority opinions than MSE.

Name: _____

Neural Networks

6. (15 points) Mira's father is an archaeologist who appraises Chinese antiques. Since his daughter recently took 6.036, he asked her a favor: to build a classifier to predict from which dynasty each antique artifact originates. Specifically, each antique artifact was built by one of the four dynasties: `Tang` (A.D. 618-907), `Song` (A.D. 960-1276), `Ming` (A.D. 1368–1644), `Qing` (A.D. 1636-1912). Mira decides to build a classifier using a neural network and train it using negative log likelihood (NLL) loss. Recall that the negative log likelihood loss for a single example is defined as:

$$L_{NLL}(\hat{y}, y) = -\sum_{i=1}^{n_y} y_i \log \hat{y}_i$$

where $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_{n_y})$ denotes the predicted probability distribution over the classes and $y = (y_1, \ldots, y_{n_y})$ is the ground truth, a one hot vector that has zero at each index except at the correct class: $y = (1, 0, 0, 0)$ for `Tang`, $y = (0, 1, 0, 0)$ for `Song`, $y = (0, 0, 1, 0)$ for `Ming`, $y = (0, 0, 0, 1)$ for `Qing`.

(a) Assume that Mira is given an antique that belongs to `Ming` dynasty ($y = (0, 0, 1, 0)$). Which of these predictions has the smallest NLL loss?

◯ A. $\hat{y} = (0.25, 0.20, 0.30, 0.25)$

√ **B.** $\hat{y} = (0.01, 0.01, 0.44, 0.54)$

◯ C. $\hat{y} = (0.25, 0.25, 0.25, 0.25)$

◯ D. $\hat{y} = (0.97, 0.01, 0.01, 0.01)$

Solution: The NLL loss is smallest when the predicted probability of the correct class (class 3, here) is highest. This occurs for option B.

(b) Apart from the NLL loss, Mira is also thinking about trying out other loss functions. In particular, she is thinking about using the accuracy:

$$L_{accuracy}(\hat{y}, y) = \begin{cases} -1 & \arg\max(y) = \arg\max(\hat{y}) \\ 0 & otherwise \end{cases}$$

or the squared loss function:

$$L_{squared}(\hat{y}, y) = (1 - \hat{y}y^\top)^2$$

as her new loss functions. Which of the these loss functions can be minimized by the stochastic gradient descent (SGD) algorithm (mark all that apply)?

√ **A. NLL-loss,** L_{NLL}

◯ B. Accuracy, $L_{accuracy}$

√ **C. Squared loss,** $L_{squared}$

Solution: NLL-loss and squared loss are both differentiable functions that can be minimized using SGD, as seen in previous labs in this class. $L_{accuracy}$ is discontinuous, and has a derivative of 0 everywhere that the derivative is defined, and so cannot be minimized using SGD.

(c) After trying out different model architectures, Mira finds that softmax classifier works well. When she uses softmax, the last layer of her network computes pre-activations $z = (z_1, ..., z_{n_y})$ which may be arbitrarily large or small (Note: a pre-activation is the linearly weighted sum that is an input to the activation function). Softmax function then computes $\hat{y} = (\hat{y}_1, ..., \hat{y}_{n_y})$ by normalizing z such that the sum of the $\hat{y}_i$ is 1:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{n_y} e^{z_j}}.$$

Mira finds that for some settings of the pre-activation values, the basic softmax function works poorly. She finds out that subtracting the maximum value $\max z_i$ from all pre-activations z_i produces more reliable results. Explain why.

Solution: First, note that this transformation does not change the ultimate estimate of $\hat{y}$. Let $z_m = \max_i z_i$.

$$\begin{aligned} \hat{y}_i &= \frac{e^{(z_i - z_m)}}{\sum_{j=1}^{n_y} e^{(z_j - z_m)}} \\ &= \frac{e^{z_i} e^{-z_m}}{\sum_{j=1}^{n_y} e^{z_j} e^{-z_m}} \\ &= \frac{e^{z_i}}{\sum_{j=1}^{n_y} e^{z_j}} \end{aligned}$$

Calculating softmax values can go wrong if any value is very large. The e^{z_i} of even a moderate-magnitude positive number z_i can be astronomically huge. This makes the scaling sum very large. Dividing by a very large number can cause arithmetic computation problems.

(d) Say that Mira wanted to solve a slightly different problem: given an artifact, Mira would like to figure out what the probability is that the artifact is "typical" of each of the four time periods. E.g. there could be an antique crafted in a style which was popular both during the `Tang` and `Ming` eras, but not at all in the other two eras, in which case ideally we would output $y = (1, 0, 1, 0)$. Choose a different structure (activation function and number of nodes) for the last layer. Specify a loss function that would work better for this multi-class labeling task.

Activation function and output nodes:

Solution: It can be formulated as n_y independent logistic regression tasks, each trying to predict whether the example belongs to the corresponding class or not. Specifically, each output in the last layer is fed to the sigmoid function, then the NLL loss is computed with respect to each class. The final loss can be the mean of the n_y NLL losses over all classes.

Loss function:

Solution: It can be formulated as n_y independent logistic regression tasks, each trying to predict whether the example belongs to the corresponding class or not. Specifically, each output in the last layer are fed to the sigmoid function, then the NLL loss is computed with respect to each class. The final loss can be the mean of the n_y NLL losses over all classes.

Name: _____

Initialization is Important

7. (15 points) In this problem we will try to understand why proper initialization of weights in a network is important.

   Kim constructs a fully connected deep neural network with L=4 layers using negative log-likelihood (NLL) loss and ReLU activation functions for all hidden layers, and a softmax for the output layer. The ReLU activation function is implemented as $\text{ReLU}(z) = max(0, z)$, with $\partial\text{ReLU}(z)/\partial z = 1$ if $z > 0$, and 0 otherwise. Kim uses random initialization for all of the layers except for layer 2, where he uses zero initialization (*i.e.,* the layer weights are $W^2 = 0$ and $W_0^2 = 0$).

   (a) Before training, he is curious about the output of his network as initialized. What will Kim observe on the output when he provides different input examples, $x^{(i)}$?

   Solution: All z_i^2 outputs from layer 2 will be 0, but since W^3 are random including the weight offset term, the z_j^3 pre-activation outputs will be random, and thus he will observe some random output for $\hat{y}_i$. However, this will be the *same* random output for all of the inputs that he tries.

   (b) The network will be trained with stochastic gradient descent (SGD). Specify an update rule for W^1 (layer 1 weights) in terms of $\frac{\partial L}{\partial W^1}$ and step size η. Similarly, specify an update rule for W^2 in terms of $\frac{\partial L}{\partial W^2}$ and step size η.

   Solution:

   $$W^1 := W^1 - \eta \frac{\partial L}{\partial W^1}$$

   and

   $$W^2 := W^2 - \eta \frac{\partial L}{\partial W^2}$$

   Kim (correctly) derives the gradient of the loss function with respect to weights W^1 in terms of the activation functions A^l, weights W^l, pre-activations Z^l, and partials $\frac{\partial L}{\partial A^4}$, $\frac{\partial A^l}{\partial Z^l}$, for l = 1,...,4:

   $$\frac{\partial L}{\partial W^1} = \frac{\partial Z^1}{\partial W^1} \cdot \frac{\partial L}{\partial Z^1}, \tag{1}$$

   where

   $$\frac{\partial L}{\partial Z^1} = \frac{\partial A^1}{\partial Z^1} \cdot W^2 \cdot \frac{\partial A^2}{\partial Z^2} \cdot W^3 \cdot \frac{\partial A^3}{\partial Z^3} \cdot W^4 \cdot \frac{\partial A^4}{\partial Z^4} \cdot \frac{\partial L}{\partial A^4}. \tag{2}$$

(c) After one iteration of gradient descent, will the new weights W^1 be different than the initial weights W^1? Explain why.

Solution: After one iteration, W^1 will have the same random values as initialized, i.e., W^1 is not updated because W^2 is zero for that update, so $\frac{\partial L}{\partial Z^1} = 0$ and $\frac{\partial L}{\partial W^1} = 0$.

(d) After that first iteration of gradient descent, will the new weights W^2 be different than the initial weights $W^2 = 0$? Explain why.

Solution: After one iteration, W^2 will still be zero. In this case, it is because $\frac{\partial A^2}{\partial Z^2} = 0$ since all of the Z^2 are zero for $W^2 = 0, W_0^2 = 0$, given that this ReLU implementation has zero slope at $z = 0$. That is to say, for this ReLU activation in a layer with all zero weights, the gradients of the activations with respect to the pre-activations are all zero, and the weights in that layer never update.

(e) Kim finds that his network with initialization of $W^2 = 0, W_0^2 = 0$ and his ReLU activation as defined above for $L = 2$ performs poorly. He switches to a sigmoid activation function for the hidden nodes in layer $L = 2$ but still uses zero initialization as previously for $L = 2$. Kim finds that the network trains and performs much better. Explain why.

Solution: Solution: In this case, on the first SGD iteration W^1 is still not updated because W^2 is zero. However, now W^2 *does* get updated on the first iteration because the gradient of the sigmoid $\frac{\partial A^2}{\partial Z^2} \neq 0$, even though $Z^2 = 0$. This enables W^2 to update in this case, unlike in the ReLU case. Thus on the second and later SGD iterations, the weights in both W^1 and W^2 will update, allowing the network to learn and perform adequately even with the disadvantage of zero W^2 initialization.