Solutions

10-315 Intro to ML
Spring 2023
Midterm Exam 1

Name: _____

Andrew ID: _____

Instructions:

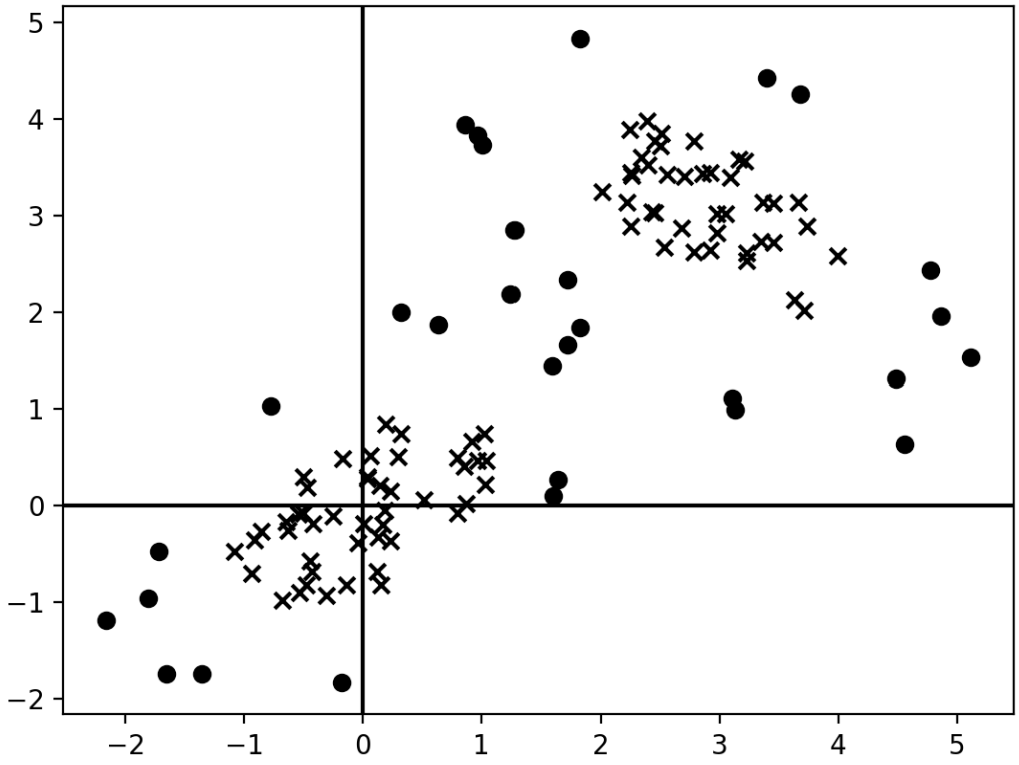
- Please fill in your name and Andrew ID above.
- Be sure to write neatly and dark enough or you may not receive credit for your exam.
- This exam contains 16 pages (including this cover page and a blank page at the end). The total number of points is 74.
- Repeated from Piazza post: You use a one-page, two-sided, hand-written sheet of paper containing your notes to use during the exam. Note: this may NOT be created digitally and printed out. No calculators will be necessary or allowed.
- Note: All vectors on this exam are column vectors unless we state otherwise.
- Note: All logs are natural logs unless we state otherwise.

Q1	Decision Boundaries	8 pts
Q2	Decision Trees	10 pts
Q3	Model Selection	6 pts
Q4	Parameters	8 pts
Q5	Multivariable Calculus	14 pts
Q6	Linear Regression	16 pts
Q7	Neural Networks	12 pts

Below is a space for you to write any assumptions you make as you go through the exam that you would like us to consider. If you have nothing to state, you can leave this box blank.

Note: If at all possible, you should raise your hand to ask a clarifying question during the exam rather than relying on any assumptions.

1 Decision Boundaries (8 points)



1. (5 points) Which of the following could possibly perfectly classify the above dataset? Select all that apply.

   ■ K-nearest neighbor with $K = 1$

   ☐ K-nearest neighbor with $K = N/2$, where N is the number of data points

   ☐ K-nearest neighbor with $K = N$, where N is the number of data points

   ☐ Decision stump (tree with depth limit 1)

   ■ Decision tree with depth limit 2

   ☐ Logistic regression

   ■ Logistic regression with polynomial features

   ■ Neural network with one hidden layer

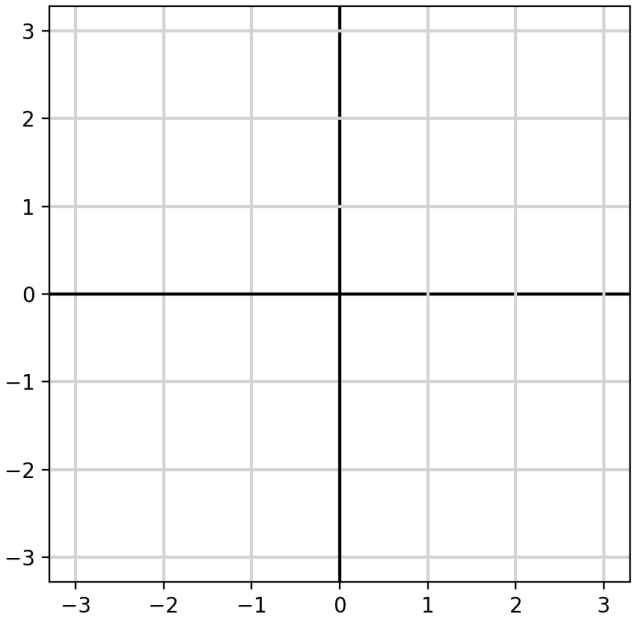
   ■ Neural network with two hidden layers

   ☐ None of the above

2. (3 points) Draw a dataset of a total of 4 unique points with input in $\mathbb{R}^2$, where 2 of the points have positive labels (drawn as $\times$) and 2 of the points have negative labels (drawn as $\bullet$) such that the 1-nearest neighbor decision boundary is the same as an optimal decision tree boundary but different than the optimal logistic regression boundary.

   Also, draw the resulting decision tree/1-NN boundary.

   To make it more clear when grading, only select points with integer input values.



   One dataset that would work is:

   0,0,+ 0,1,- 1,0,- 1,1,+

   With boundaries along vertical and horizontal lines at $x_1 = 0.5$ and $x_2 = 0.5$, respectively.

2 Decision Trees (10 points)

Consider the following training dataset with output binary variable Y and binary input features A, B, C:

A	B	C	Y
F	F	F	F
T	F	T	T
T	T	F	T
T	T	T	F

In this problem, we'll be using decision trees with classification error rate as the splitting criteria.

Note: Break any ties using alphabetical order

When drawing any decision tree below, make sure to:

- Label each node with the attribute name used to split (A, B, or C)
- Label each branch with the attribute value for that branch (T or F)
- Label each leaf with majority classification (T or F)

1. (2 points) Using a decision stump (depth limit one), what is the classification error rate on the training data?

   0.25

(Question continued on next page)

2. (4 points) Draw the decision tree that would be learned by the decision tree algorithm without any limit on the depth.

   Note: See drawing instructions on previous page

   Split on A
   –A=F leaf (predict F)
   –A=T: Split on either B (or C with incorrect tie breaking)
   —-B=F leaf (predict T)
   —-B=T: Split on C (or B with incorrect tie breaking)
   ———C=F leaf (predict T)
   ———C=T leaf (predict F)

3. (4 points) Is it possible to come up with a decision tree (without any additional feature engineering) that has a smaller depth than the tree above but with the same classification rate? If so, draw that tree, otherwise mark "Not possible".

   ◯ Not possible

   Correct depth two tree

   Split B
   –B=T Split C
   –B=F Split A or C

   or

   Split C
   –C=T Split B
   –C=F Split A or B

3 Model Selection (6 points)

1. (3 points) Consider a situation where we want to use K-fold cross-validation to find the best number of polynomial features to use for multiclass logistic regression (with softmax) on a particular dataset.

   Given the following constants, how many times will we train a model?

   - P: the number of polynomial degrees we are considering
   - C: the number of classes (note: using C here rather than the traditional K for the number of classes)
   - M: the dimensionality of the input features, $\mathbf{x}$ (note: this doesn't yet include an explicit 1 to account for the bias term
   - N: the number of data points in the dataset
   - K: the number of folds for K-fold cross-validation. You may assume that N is a multiple of K.
   - E: the number of epochs we will run with stochastic gradient descent

   Write an expression for the number of times a model is trained in terms of P, C, M, N, K, E

   PK

2. (3 points) Suppose we have a dataset with N data samples of rocks. We compute the K-fold cross-validation error for one specific hyperparameter setting of our classification model on this given dataset. Unfortunately, the first N/K samples in the dataset are collected from rocks on Mars, so they are significantly different from the other samples. As a result, we are getting a 1.0 error rate on the first fold of cross-validation and we are getting a 0.0 error rate on all other folds.

   What would be the overall cross-validation error? Write this as an expression in terms of N and K.

   $\frac{K-1}{K}$

4 Parameters (8 points)

Given input data $\mathbf{x} \in \mathbb{R}^3$, how many parameters do the following models have? Assume that linear components of all models have bias terms.

1. (2 points) Binary logistic regression

   4

2. (2 points) Multiclass logistic regression with two output values, i.e. two classes

   8

3. (2 points) Neural network with zero hidden layers, one output value from a logistic function, and a cross-entropy loss

   4

4. (2 points) Neural network with zero hidden layers, two output values from a softmax function, and a cross-entropy loss

   8

5 Multivariable Calculus (14 points)

1. Consider the following matrix and vector: $A \in {\mathbb{R}_{>0}}^{N \times M}$ and $\mathbf{v} \in {\mathbb{R}_{>0}}^{M}$. The entries for A and $\mathbf{v}$ are all strictly positive real numbers.

   Give the count of the total number of entries as well as the count of nonzero entries in the following partial derivatives (we are not concerned about the format/shape, just the count). Write your answers in terms of N, M, and K.

   (a) (2 points) $\frac{\partial}{\partial \mathbf{v}}\ 3\mathbf{v}$

   Num entries: MM Num nonzeros: M

   (b) (2 points) $\frac{\partial}{\partial \mathbf{v}}\ A\mathbf{v}$

   Num entries: NM Num nonzeros: NM

   (c) (2 points) $\frac{\partial}{\partial A}\ 3A$

   Num entries: $NMNM$ Num nonzeros: NM

2. (8 points) Prove that $\frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T A \mathbf{u} = \mathbf{u}^T A^T$ for $\mathbf{v}, \mathbf{u} \in \mathbb{R}^2$, $A \in \mathbb{R}^{2\times 2}$ when using numerator format.

   - Use **numerator** format.
   - Define

$$f(\mathbf{v}) = \mathbf{v}^T A \mathbf{u} \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

   - Start by expanding $A\mathbf{u}$ and then $\mathbf{v}^T A\mathbf{u}$, i.e. write $f(v_1, v_2)$ in terms of scalars $v_1, v_2, u_1, u_2, A_{1,1}, A_{1,2}, A_{2,1}, A_{2,2}$.
   - Write out scalar partial derivatives $\frac{\partial f}{\partial v_1}$ and $\frac{\partial f}{\partial v_2}$.

   You should be able to justify each step in your proof, however, you don't have to write that justification. In other words, don't take steps that are too big.

First note

$$\begin{aligned} \mathbf{v}^T A \mathbf{u} &= \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} u_1 A_{1,1} + u_2 A_{1,2} \\ u_1 A_{2,1} + u_2 A_{2,2} \end{bmatrix} \\ &= v_1 u_1 A_{1,1} + v_1 u_2 A_{1,2} + v_2 u_1 A_{2,1} + v_2 u_2 A_{2,2} \end{aligned}$$

Taking partial derivatives...

$$\begin{aligned} \frac{\partial}{\partial v_1} \mathbf{v}^T A \mathbf{u} &= u_1 A_{1,1} + u_2 A_{1,2} \\ \frac{\partial}{\partial v_2} \mathbf{v}^T A \mathbf{u} &= u_1 A_{2,1} + u_2 A_{2,2} \\ \frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T A \mathbf{u} &= \begin{bmatrix} \frac{\partial}{\partial v_1} & \frac{\partial}{\partial v_2} \end{bmatrix} \\ &= \begin{bmatrix} u_1 A_{1,1} + u_2 A_{1,2} & u_1 A_{2,1} + u_2 A_{2,2} \end{bmatrix} \\ &= \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} A_{1,1} & A_{2,1} \\ A_{1,2} & A_{2,2} \end{bmatrix} \\ &= \mathbf{u}^T A^T \end{aligned}$$

6 Linear Regression (16 points)

Suppose we are given a supervised training dataset containing exactly two points, $x^{(1)}, y^{(1)}$ and $x^{(2)}, y^{(2)}$, where both the input and output values are scalar real values, i.e., $x^{(1)}, x^{(2)} \in \mathbb{R}$ and $y^{(1)}, y^{(2)} \in \mathbb{R}$.

We are going to use a linear model with a scalar weight, w, and no bias term. We'll use squared error loss and linear regression to determine an optimal estimate for w.

Let $J(w)$ be the objective function representing the empirical risk for this problem.

1. (2 points) What shape is function $J(w)$? We are looking for a one-word answer for the shape. Be specific, e.g. don't say rectangle if it is in fact a square. (You don't need to consider any degenerate cases for this shape, e.g. if it is a rectangle, you don't have to worry about a rectangle of height zero.)

   parabola

2. (2 points) Which of the following are true about the objective function $J(w)$?

   Select all that apply.

   ☐ $J(w)$ is linear

   ☐ $J(w)$ is affine

   ■ $J(w)$ is convex

   ☐ $J(w)$ is concave

   ☐ None of the above

   Note: The two parts above will be graded independently from each other. For example, you can't (incorrectly) answer sinusoid for the first part and then get credit for (correctly) describing the properties of a sinusoid.

(Question continued on next page)

Note: Showing your work for both parts below is not required for full credit. However, we will consider any work you show when determining partial credit.

3. (8 points) Write the derivative of the objective function with respect to w. You must write dJ/dw in terms of just $w, x^{(1)}, x^{(2)}, y^{(1)}, y^{(2)}$. In other words, you may not use summation notation or linear algebra in your answer.

dJ/dw

$$
\begin{aligned}
J(w) &= \frac{1}{N}\|\mathbf{y} - w\mathbf{x}\|_2^2 \\
&= \frac{1}{N}\left[\mathbf{y}^\top\mathbf{y} - 2w\mathbf{y}^\top\mathbf{x} + w\mathbf{x}^\top\mathbf{x}w\right] \\
&= \frac{1}{N}\left[{y^{(1)}}^2 + {y^{(2)}}^2 - 2w\left(y^{(1)}x^{(1)} + y^{(2)}x^{(2)}\right) + w^2\left({x^{(1)}}^2 + {x^{(2)}}^2\right)\right] \\
\frac{dJ}{dw} &= -\left(y^{(1)}x^{(1)} + y^{(2)}x^{(2)}\right) + w\left({x^{(1)}}^2 + {x^{(2)}}^2\right)
\end{aligned}
$$

4. (4 points) Write the closed-form solution for the estimate of w. You must write your answer in terms of just $x^{(1)}, x^{(2)}, y^{(1)}, y^{(2)}$. Again, you may not use summation notation or linear algebra in your answer.

Estimate for w

$$
w = \frac{y^{(1)}x^{(1)} + y^{(2)}x^{(2)}}{{x^{(1)}}^2 + {x^{(2)}}^2}
$$

7 Neural Networks for Logistic Regression (12 points)

Now that you've done so much work to implement the forward pass and backpropagation for neural networks, you are going to write the `predict` and `train` functions for multiclass logistic regression (with softmax) by reusing your neural network functions.

Assume that the following functions are already implemented and working. Unless otherwise noted, you may also assume that all arguments and return values are NumPy arrays with the appropriate shape.

Assume that any bias term or feature engineering has already been accounted for in the x vectors in the data.

- `output = linear_forward(input, params)`
- `dJ_dparams, dJ_dinput = linear_backward(input, params, dJ_doutput)`
- `output = softmax_forward(input)`
- `loss = cross_entropy_forward(y, y_hat)`
- `dJ_dinput = cross_entropy_and_softmax_backward(y, y_hat)`

1. (4 points) Use the above functions as necessary to implement the Python function `predict` that takes the input for one data point, `x`, and the multiclass logistic regression parameter matrix, `params`, and returns the vector of predicted probabilities for each class.

Predict

```
def predict(x, params):









    # Don't forget to return the prediction vector
```

Functions repeated for convenience:

- `output = linear_forward(input, params)`
- `dJ_dparams, dJ_dinput = linear_backward(input, params, dJ_doutput)`
- `output = softmax_forward(input)`
- `loss = cross_entropy_forward(y, y_hat)`
- `dJ_dinput = cross_entropy_and_softmax_backward(y, y_hat)`

2. (8 points) Use the above functions as necessary to implement the Python function `train` that takes in a training dataset (input and output), initial multiclass logistic regression parameter matrix, the number of epochs (int), and the learning rate (float) and returns the trained multiclass logistic regression parameter matrix after running stochastic gradient descent. No need to shuffle data. We've started the SGD code for you.

Train

```
def train(train_data, initial_params, num_epochs, learning_rate):
    params = initial_params
    for e in range(num_epochs):
        for x, y in train_data:









    return params
```

```
def predict(x, params):
    z = linear_forward(x, params)
    return softmax_forward(z)

def train(train_data, initial_params, num_epochs, learning_rate):
    params = initial_params
    for e in range(num_epochs):
        for x, y in train_data:
            z = linear_forward(x, params)
            yhat = softmax_forward(z)

            dJ_dz = cross_entropy_and_softmax_backward(y, yhat)
            dJ_dparams, dJ_dx = linear_backward(x, params, dJ_dz)

            params = params - learning_rate*dJ_dparams

    return params
```

This page intentionally left blank.