6.390: Midterm Exam, Spring 2025

Solutions

- This exam is closed-book, and you may **not** use any electronic devices (including computers, calculators, phones, etc.). The total exam time is 2 hours.
- One reference sheet (8.5 in. by 11 in.) with notes on both sides is permitted. Blank scratch paper will also be provided if needed. You do **not** need to submit your reference sheet or the scratch paper.
- The problems are not necessarily presented in any order of difficulty.
- Please write all answers in the provided boxes. If you need more space, clearly indicate near the answer box where to find your work.
- Please write your Kerberos on every page of this exam.
- For all multiple choice questions, please **choose all that apply.**
- If you have a question, please **come to us directly**. You may also raise your hand, but if we do not see you, please approach us.
- You may **not** discuss the details of the exam with anyone other than the course staff until exam grades have been assigned and released.

Name: ______________________ Kerberos: ______________________

Question:	1	2	3	4	5	6	Total
Points:	15	22	9	18	16	20	100
Score:							

Professor Regu LaRisashun

1. (15 points) Professor Regu LaRisashun loves exploring different forms of regularization, and is currently testing a variety of choices for the regularization term $R(\theta)$.

   She is only interested in linear regression, where the linear hypothesis $h(x;\theta) = \theta^T x$ has **no** offset θ_0. Further, all of her work only considers objective functions with the form:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(\theta^T x^{(i)} - y^{(i)})^2 + \lambda R(\theta).$$

   You are pursuing a UROP (an undergraduate research position) in Prof. Regu's lab, but you have to pass an interview with Dr. Poe Stdoc first!

   (a) Dr. Poe wants to test your understanding of ridge regression, where the regularization term is given by $R(\theta) = \|\theta\|^2$. He provides various λ settings, and your task below is to match each one to the corresponding magnitude of the optimal parameters θ^*.

   i. λ goes to $+\infty$

   Solution:
   - [✓ $\|\theta^*\| = 0$]
   - ○ $\|\theta^*\|$ goes to $+\infty$
   - ○ $\|\theta^*\|$ depends on the data

   ii. $\lambda = 0$

   Solution:
   - ○ $\|\theta^*\| = 0$
   - ○ $\|\theta^*\|$ goes to $+\infty$
   - [✓ $\|\theta^*\|$ **depends on the data**]

   iii. λ goes to $-\infty$

   Solution:
   - ○ $\|\theta^*\| = 0$
   - [✓ $\|\theta^*\|$ **goes to** $+\infty$]
   - ○ $\|\theta^*\|$ depends on the data

   Congratulations, you passed! Prof. Regu and Dr. Poe agree that you should join the team and put you to work immediately.

   (b) First, Prof. Regu wants to construct a regularizer that forces all entries of θ to be close to her favorite number, 3.

How would you construct a regularization term $R(\theta)$ to achieve this?

Solution: $R(\theta) = \|\theta - \mathbf{3}\|^2$.

(c) Now, Professor Regu introduces a new "flexible" regularizer:

$$R(\theta) = \left\| \left(\frac{\alpha}{\gamma}\right)^{\beta} \theta \right\|^2,$$

where α, β, and γ are strictly positive scalars.

Dr. Poe argues that this regularizer is not truly "new"—the same regularization effect from any given λ, α, β, and γ can be achieved by an appropriate choice of hyperparameter λ_{ridge} in ridge regression.

Do you agree with Dr. Poe? If yes, describe how you would choose λ_{ridge} to replicate the regularization effect of the proposed regularizer. If no, explain why not.

Solution: $\lambda_{\text{ridge}} = \lambda \left(\frac{\alpha}{\gamma}\right)^{2\beta}$.

(d) Finally, you investigate how nonlinear feature transformations affect model generalization and performance by constructing a polynomial feature space of degree $k > 0$ and applying analytical ridge regression.

To determine the optimal k for a given dataset $\mathcal{D}$, you implement a 10-fold cross-validation approach.

Below is incomplete pseudo-code for "Cross-Validation for Selecting Polynomial Basis Order," which you will refine to complete your algorithm:

Algorithm 1 Cross-Validation for Selecting Polynomial Basis Order

```
Require: Data $\mathcal{D}$, ——(A)——
1: Divide $\mathcal{D}$ into 10 chunks $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_{10}$ (of roughly equal size)
2: ——(B)——
3: for ——(C)—— do
4:     ——(D)——
5:     for $i = 1$ to 10 do
6:         Train $h_i$ on $\mathcal{D}^k \setminus \mathcal{D}_i^k$
7:         Compute "test" error $\mathcal{E}_i(h_i)$ on withheld data $\mathcal{D}_i^k$
8:     end for
9:     ——(E)——
10: end for
11: return ——(F)——
```

Match each pseudo-code line below to its corresponding location in the algorithm. Each line is used exactly once, and each location corresponds to a single pseudo-code line.

Solution:

Algorithm 2 Filled-In Cross-Validation for Selecting Polynomial Basis Order

```
Require: Data $\mathcal{D}$, $\boxed{\texttt{List of polynomial degrees } \mathcal{C}}$
1: Divide $\mathcal{D}$ into 10 chunks $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_{10}$ (of roughly equal size)
2: $\boxed{\texttt{Error} \leftarrow \texttt{Empty List []}}$
3: for $\boxed{k \texttt{ in } \mathcal{C}}$ do
4:     $\boxed{\texttt{Transform Data Chunks ...}}$
5:     for $i = 1$ to 10 do
6:         Train $h_i$ on $\mathcal{D}^k \setminus \mathcal{D}_i^k$
7:         Compute "test" error $\mathcal{E}_i(h_i)$ on withheld data $\mathcal{D}_i^k$
8:     end for
9:     $\boxed{\texttt{AddItem(Errors, } \frac{1}{10}\sum_{i}^{10} \mathcal{E}_i(h_i))}$
10: end for
11: return $\boxed{\texttt{Polynomial degree in } \mathcal{C} \texttt{ that produced min(Errors)}}$
```

Divide and Conquer

2. (22 points) Amy and Brandon want to compare different algorithms for learning linear regression hypothesis $h(x;\theta) = \theta^T x$ with **no** offset θ_0.

   They first consider this data set $\mathcal{D}_{\text{train}}$ with $n = 4$ data points and $d = 3$ features below:

```python
import numpy as np
X_train = np.array([
    [-2,  4, -0.5],
    [-1,  2, -1 ],
    [ 1, -4,  1 ],
    [ 2, -2,  0.5]
])

Y_train = np.array([[-2, -1, 1, 2]]).T
```

   (a) Amy says that it's always the best to use the ordinary least squares closed-form formula:

$$\theta^* = (X^\top X)^{-1} X^\top Y$$

   where $X \in \mathbb{R}^{n \times d}, Y \in \mathbb{R}^n$. She used this implementation from her 6.390 homework:

```python
def lin_reg_analytic(X, Y):
    return np.linalg.inv(X.T@X)@X.T@Y
```

   Amy calls `lin_reg_analytic(X_train, Y_train)`, which returns $\theta^* =$ `[[1, 0, 0]].T`.

   i. Using squared-error loss $\mathcal{L}(g,a) = (g-a)^2$ and the learned θ^*, what is the value of the mean-squared-error (MSE) on $\mathcal{D}_{\text{train}}$?

   Solution: MSE on $\mathcal{D}_{\text{train}} = 0$

   ii. Consider a validation data set $\mathcal{D}_{\text{val}}$ with 6 data points:

```python
X_val = np.array([
    [-3,    -2.2,  0.6],
    [-2.5,  -4.4,  1.2],
    [-1.5,  -1.6,  0.46],
    [0,      1.1, -0.3],
    [1.75,   2.2, -0.6],
    [3,      6.6, -1.8],
    ])
Y_val = np.array([[1, 2, 0.75, -0.5, -1, -3]]).T
```

   Amy found the MSE on this $\mathcal{D}_{\text{val}}$ is 14.19. Compare it to the MSE on $\mathcal{D}_{\text{train}}$ you calculated above. How would you characterize the model's performance based on these errors?

   Solution: The two MSEs suggest the model is overfitting.

iii. Which of the following could help alleviate the issue observed above?
Briefly justify your answer — explain why each option *would* or *would not* be helpful.

Solution:

○ Learn θ^* to minimize the MSE on $\mathcal{D}_{\text{train}}$ by using gradient descent and terminate only when $|\nabla_\theta J(\theta)|$ is sufficiently small.

✓ **Add ridge regularization to the objective function, then calculate the optimal parameters of the ridge objective.**

○ Change how we measure the validation error. Instead of MSE on $\mathcal{D}_{\text{val}}$, we calculate the mean-absolute-value loss $\mathcal{L}(g^{(i)}, y^{(i)}; \theta^*) = |g^{(i)} - y^{(i)}|$ on $\mathcal{D}_{\text{val}}$.

(b) Unsatisfied with Amy's closed-form solution, Brandon is developing his own algorithm: BLA ("Brandon's Learning Algorithm"):

```
 1  # lin_reg_analytic, X_train and Y_train are the same as defined
        previously in part (a), copied below for convenience.
 2
 3  X_train = np.array([
 4          [-2,  4, -0.5],
 5          [-1,  2, -1 ],
 6          [ 1, -4,  1 ],
 7          [ 2, -2,  0.5]
 8      ])
 9
10  Y_train = np.array([[-2, -1, 1, 2]]).T
11
12  def lin_reg_analytic(X, Y):
13      return np.linalg.inv(X.T@X)@X.T@Y
14
15  # now, Brandon's own algorithm below
16  def BLA(X, Y):
17      # X is nxd, Y is nx1
18      n, d = X.shape
19      th_is = []
20      for i in range(d):
21          th_i = lin_reg_analytic(X[:, i:i+1], Y)
22          th_is.append(th_i)
23  # recall that np.vstack() stacks arrays vertically
24      return (1/d)*np.vstack(th_is)
```

Brandon runs `BLA(X_train, Y_train)`.

i. When `i = 1` in line 20, what is the value of `th_i` calculated in line 21?
(We expect a numerical answer; a fraction is fine.)

Solution:
$-\frac{9}{20}$ or -0.45.

ii. What is the output shape of the returned array (in line 24)?
(We expect just the shape, not the entry values.)

> **Solution:**
> `(3,1)` or 3×1

iii. Brandon gets θ^*_{BLA} from running `BLA(X_train, Y_train)`. Out of curiosity, he also tried to minimize the ridge regression objective with $\lambda = 1.9$, and he learned parameters θ^*_{ridge} via the ridge-regression closed-form formula:

$$\theta^*_{\text{ridge}} = (X^\top X + n\lambda I)^{-1} X^\top Y$$

Using these learned parameters, Brandon calculated the MSE on the training data set, and the validation data set. The results are shown in the table below:

Parameters	MSE on $\mathcal{D}_{\text{train}}$	MSE on $\mathcal{D}_{\text{val}}$
θ^*_{BLA}	0.26	1.64
θ^*_{ridge}	0.26	1.64

Does the table suggest that $\theta^*_{\text{BLA}} = \theta^*_{\text{ridge}}$?

> **Solution:**
>
> ◯ Yes, because only a unique θ can produce the matching MSE on the given $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$.
>
> ◯ Yes, because one can always rearrange the BLA solution formula to get the analytical ridge regression solutions for any choice of λ.
>
> ◯ Yes, because while the formulae for BLA and ridge regression are generally different, they are always equivalent when $d \leq 3$.
>
> [✓ **No, because there are multiple θ values that could produce the matching MSE on the given $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$, so there is not enough info to determine if $\theta^*_{\text{ridge}} = \theta^*_{\text{BLA}}$.**]

(c) Consider a new training data set:

```python
# New training data set
X2_train = np.array([
    [1, 2, 3],
    [2, 3, 5],
    [3, 4, 7],
    [4, 5, 9],
    ])
Y2_train = np.array([[2, 4, 6, 8]]).T

# Amy's and Brandon's learning algorithms; same as before, copied below
    for convenience.

# X is nxd, Y is nx1

def lin_reg_analytic(X, Y):
    return np.linalg.inv(X.T@X)@X.T@Y

def BLA(X, Y):
    n, d = X.shape
    th_is = []
    for i in range(d):
        th_i = lin_reg_analytic(X[:, i:i+1], Y)
        th_is.append(th_i)
    return 1/d*np.vstack(th_is)
```

Amy and Brandon use their learning algorithms on this new training data set. How will their methods perform on the new data?

i. Amy's approach of calling `lin_reg_analytic`(`X2_train`, `Y2_train`) will:

Solution:

○ Will return $\theta =$`[[7, 4.99, 3]].T`.

○ Will return $\theta =$`[[6.67,5.11,3.01]].T`.

○ Will return $\theta =$`[[0.67, 0.49, 0.28]].T`.

✓ **Will not return a numerical vector, that is, will have run-time error.**

ii. Brandon's approach of calling `BLA`(`X2_train`, `Y2_train`) will:

Solution:

○ Will return $\theta =$`[[7, 4.99, 3]].T`.

○ Will return $\theta =$`[[6.67,5.11,3.01]].T`.

✓ **Will return $\theta =$`[[0.67, 0.49, 0.28]].T`.**

○ Will not return a numerical vector, that is, will have run-time error.

Out and About in the Sun

3. (9 points) To help address climate change, there have been significant efforts to increase the proportion of energy that is produced from low-carbon resources such as solar. One challenge, however, is that the amount of solar power available at any given time varies based on the weather.

   To help manage this, power grid operators train machine learning models (often neural networks) to predict how much solar power will be available at particular times and locations.

   We will consider the output layer and loss function design choices for our neural networks.

   For each goal below, specify the number of neurons and the activation function in the *output layer*, and loss function that you would use.

   (a) Your goal is to predict the total amount of power (in MW) that will be produced.

   Solution:

   - Number of neurons: 1
   - Activation function: Linear (ReLU is accepted too)
   - Loss function: Squared-error

   (b) Your goal is to predict whether or not the total amount of power will exceed a threshold of 500 MW.

   Solution:

   - Number of neurons: 1
   - Activation function: Sigmoid
   - Loss function: NLL

   (c) New England is split into 6 states, and your goal is to predict whether or not the total amount of power in each state will exceed a particular (state-specific) threshold.

   Solution:

   - Number of neurons: 6
   - Activation function: 6 separate sigmoids
   - Loss function: sum of NLL losses; or average of NLL loss

Kerberos: ______________________

Chasing Minima

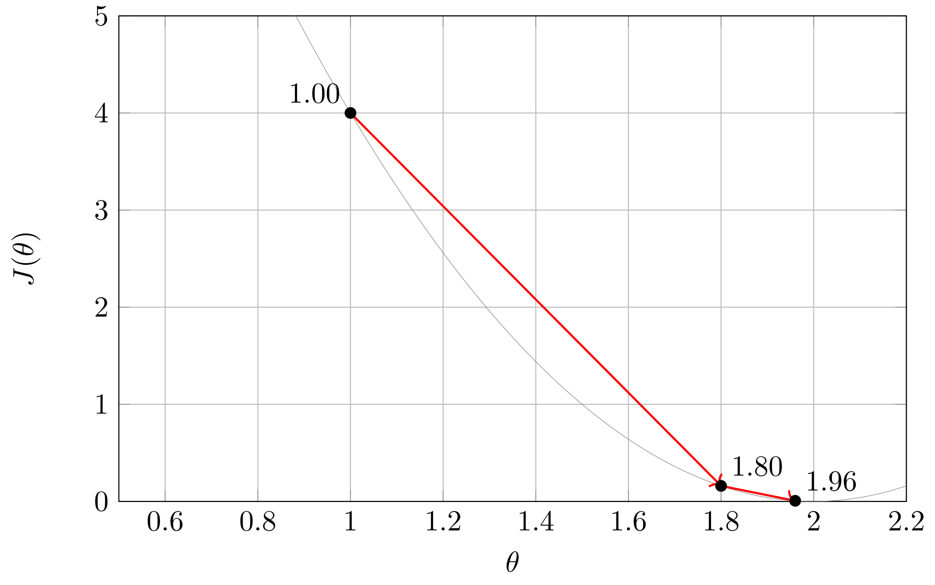
4. (18 points) In this problem, we consider two simple data sets, each consisting of one-dimensional features and labels. Our goal is to learn a linear regressor $h(x;\theta) = \theta x$ by minimizing the mean squared error (MSE).

   (a) First, consider a trivial data set made of a single data point with feature $x = 2$ and label $y = 4$. The objective is to minimize

   $$J(\theta) = (2\theta - 4)^2.$$

   The plots below show the behavior of running gradient descent (GD) in order to minimize this objective function. We initialize GD with $\theta_{\text{initial}} = 1$, and run the algorithm for two iterations. The first iteration uses learning rate η_1, and the second iteration η_2. The numbers labeled on the plot indicate the value of θ at each iteration.

   i. Which statement about the learning rates η_1 and η_2 is true?



Solution:

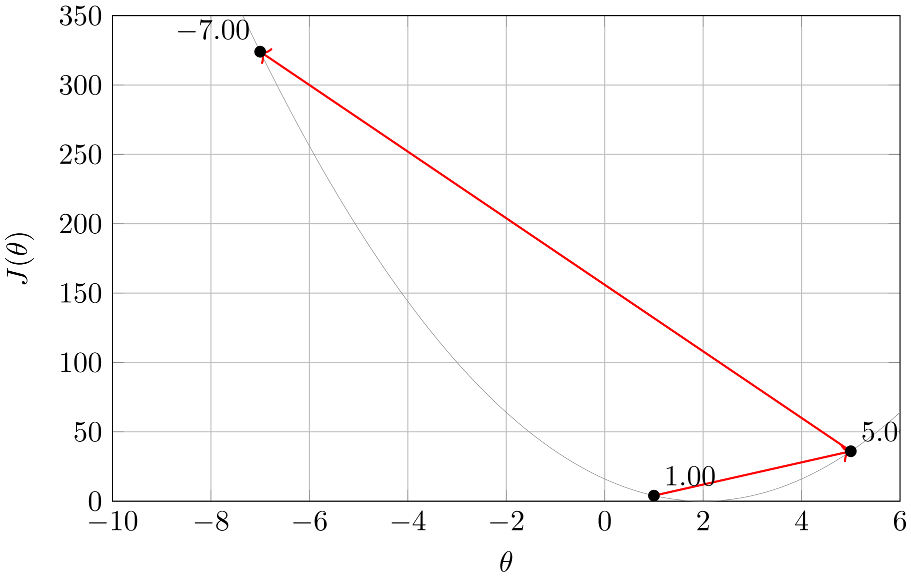
[✓ $\eta_1 = \eta_2$]

○ $\eta_1 > \eta_2$

○ $\eta_1 < \eta_2$

○ not enough information to tell

ii. Which statement about the learning rates η_1 and η_2 is true?



Solution:

[✓ $\eta_1 = \eta_2$]

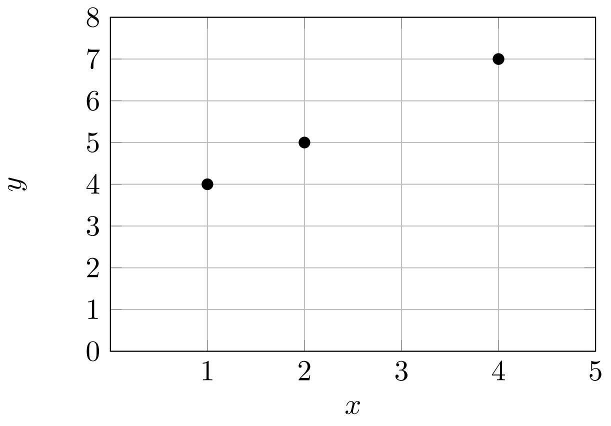
◯ $\eta_1 > \eta_2$

◯ $\eta_1 < \eta_2$

◯ not enough information to tell

(b) Now, consider the slightly more interesting training data set, consisting of 3 data points:

$$\mathcal{D}_{\text{train}} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{3} = \{(1,4),(2,5),(4,7)\}.$$



and the objective is to minimize, again, the MSE:

$$\begin{aligned} J(\theta) &= \frac{1}{3}\Big[(\theta-4)^2 + (2\theta-5)^2 + (4\theta-7)^2\Big] \\ &= \frac{1}{3}\left(21\theta^2 - 84\theta + 90\right) \\ &= 7\theta^2 - 28\theta + 30. \end{aligned}$$

i. Consider running gradient descent (GD) to learn θ. We initialize GD with $\theta_{\text{initial}} = 4$ and use a fixed learning rate of $\eta = 0.02$. After just one iteration of the GD update, which of the following is a *possible* value for the resulting updated parameter θ_{new}? Briefly justify your answer.

Solution:

○ $\theta_{\text{new}} = 4$

○ $\theta_{\text{new}} = 3.76$

[✓ $\boldsymbol{\theta_{\textbf{new}}} = 3.44$]

○ $\theta_{\text{new}} = 2.56$

○ $\theta_{\text{new}} = 0.88$

○ Not enough information to determine any *possible* value of θ_{new}.

ii. Consider running stochastic gradient descent (SGD) to learn θ. We initialize SGD with $\theta_{\text{initial}} = 4$ and use a fixed learning rate of $\eta = 0.02$. After just one iteration of the GD update, which of the following is a *possible* value for the resulting updated parameter θ_{new}? Briefly justify your answer.

Solution:

- √ $\theta_{\textbf{new}} = 4$
- √ $\theta_{\textbf{new}} = 3.76$
- ◯ $\theta_{\text{new}} = 3.44$
- √ $\theta_{\textbf{new}} = 2.56$
- ◯ $\theta_{\text{new}} = 0.88$
- ◯ Not enough information to determine any *possible* value of θ_{new}.

Go Separate Ways

5. (16 points) Consider a binary classification problem with training data $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, where each $x^{(i)} \in \mathbb{R}^d$ and $y^{(i)} \in \{0, +1\}$.

   (a) Assume $d = 3$. Consider linear binary classifiers that predict $+1$ when $\theta^T x + \theta_0 > 0$ and 0 otherwise.

   i. For a linear binary classifier given by $\theta = [1, -1, 2]^T, \theta_0 = 0$. Which of the following points x are classified as $+1$?

   Solution:

   ✓ $x^{(1)} = [1, -1, 2]^T$

   ✓ $x^{(2)} = [1, 2, 3]^T$

   ○ $x^{(3)} = [-1, -1, -1]^T$

   ✓ $x^{(4)} = [1, 1, 1]^T$

   ii. Recall that a separator is defined as $\{x : \theta^T x + \theta_0 = 0\}$. Suppose the parameters from the previous part define the separator s_1.
   Now consider a separator s_2 defined by $\theta_{\text{new}} = [-1, 1, -2]^T, \theta_{0_{\text{new}}} = 0$.
   Is the separator s_2 identical to s_1?

   Solution:

   ✓ **Yes**

   ○ No

   iii. Which of the following points are classified as $+1$ using the classifier given by $\theta_{\text{new}}, \theta_{0_{\text{new}}}$?

   Solution:

   ○ $x^{(1)} = [1, -1, 2]^T$

   ○ $x^{(2)} = [1, 2, 3]^T$

   ✓ $x^{(3)} = [-1, -1, -1]^T$

   ○ $x^{(4)} = [1, 1, 1]^T$

(b) Recall that a linear logistic classifier predicts +1 if $\sigma(\theta^T x + \theta_0) > 0.5$ and 0 otherwise, where

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

i. Which of the following points are classified as +1 using a linear logistic classifier given by $\theta = [1, -1, 2]^T, \theta_0 = 0$?

Solution:

✓ $x^{(1)} = [1, -1, 2]^T$

○ $x^{(2)} = [1, 2, 3]^T$ [✓]

○ $x^{(3)} = [-1, -1, -1]^T$

✓ $x^{(4)} = [1, 1, 1]^T$

(c) Again, assume $d = 3$. Consider a data point with $x = [10, -1, 2]^T$ and $y = +1$. A linear logistic classifier with $\theta = [1, -1, 2]^T$ and $\theta_0 = 0$ correctly predicts the label for this point. Recall that the negative log-likelihood (NLL) loss is:

$$\mathcal{L}(g, y) = -\left[y \log g + (1 - y) \log (1 - g)\right],$$

where $g = \sigma(\theta^T x + \theta_0)$.

Consider logistic classifiers defined by the following parameters.

i. Which classifier achieves the lowest NLL loss on this data point?

Solution:

○ $\theta = [1, -1, 2]^T, \theta_0 = 0$

○ $\theta = 10 * [1, -1, 2]^T, \theta_0 = 10$

✓ $\theta = 20 * [1, -1, 2]^T, \theta_0 = 20$

○ $\theta = -30 * [1, -1, 2]^T, \theta_0 = -30$

ii. Which classifier achieves the highest NLL loss on this data point?

Solution:

○ $\theta = [1, -1, 2]^T, \theta_0 = 0$

○ $\theta = 10 * [1, -1, 2]^T, \theta_0 = 10$

○ $\theta = 20 * [1, -1, 2]^T, \theta_0 = 20$

✓ $\theta = -30 * [1, -1, 2]^T, \theta_0 = -30$

(d) Consider a *linearly separable* dataset with features $x \in \mathbb{R}^d$. If we permute the order of these d features, does the dataset remain linearly separable?

Solution:

[✓ **Always yes**]

○ Always no

○ Depends on the permutation order.

(e) Consider a *not linearly separable* dataset with features $x \in \mathbb{R}^d$. Suppose we select two features, x_i (the i-th feature) and x_j (the j-th feature), and form a new feature $ax_i + bx_j$ using real scalars a, b.

We then construct an augmented dataset with features and labels:

$$x_{\text{new}} = [x,\, ax_i + bx_j]^T \in \mathbb{R}^{d+1}, \quad y_{\text{new}} = y.$$

Is the new dataset linearly separable?

Solution:

○ Always yes

[✓ **Always no**]

○ Depends on the choice of a, and b.

I LAF in the Face of Neural Networks

6. (20 points) Recall that in class, we have studied fully-connected feed-forward neural networks. In this setting, a neuron maps the m-dimensional input x to scalar output a as follows:

$$a = f\left(\left(\sum_{j=1}^{m} x_j\, w_j\right) + w_0\right) = f\left(w^T x + w_0\right) \tag{1}$$

where $f : \mathbb{R} \to \mathbb{R}$ is some *fixed* activation function, and $w \in \mathbb{R}^m, w_0 \in \mathbb{R}$ are the learnable parameters.

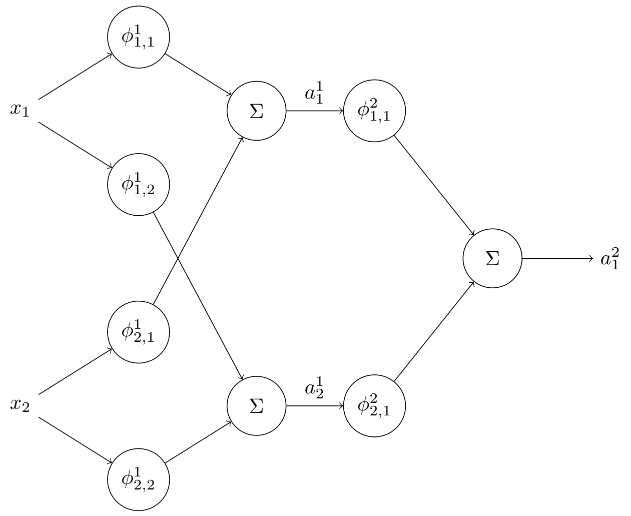
Let's now envision an alternative neural network structure, called a Learnable Activation Function (LAF) network, in which the *activation functions* are learnable. In this LAF setting, a neuron maps the m-dimensional input x to scalar output a as follows:

$$a = \sum_{j=1}^{m} \phi_j(x_j) = \sum_{j=1}^{m} \left(b_j g(x_j) + c_j h(x_j)\right).$$

where a separate activation function $\phi_j : \mathbb{R} \to \mathbb{R}$ is defined for each feature x_j, all $b_j, c_j \in \mathbb{R}$ are *learnable* parameters, and $g : \mathbb{R} \to \mathbb{R}, h : \mathbb{R} \to \mathbb{R}$ are some *fixed* functions.

(a) Consider a LAF depicted below. It has $L = 2$ layers, where the first layer has 2 neurons and the second layer, the output layer, has 1 neuron.

We use $\phi^{\gamma}_{\alpha,\beta}$ to represent the activation function of the α-th feature, in the β-th neuron, of the γ-th layer.



(This is loosely inspired by a recent paper proposing a new neural network structure called a Kolmogorov–Arnold Network.)

i. Write an expression for a^1_1 in terms of the network input x and learnable activation functions ($\phi^1_{1,1}$, $\phi^1_{1,2}$, $\phi^1_{2,1}$, $\phi^1_{2,2,}$, $\phi^2_{1,1}$, $\phi^2_{2,1}$)

Solution: $a^1_1 = \phi^1_{1,1}(x_1) + \phi^1_{2,1}(x_2)$.

ii. Write an expression for a_1^2 in terms of the network input x and learnable activation functions ($\phi_{1,1}^1$, $\phi_{1,2}^1$, $\phi_{2,1}^1$, $\phi_{2,2,}^1$, $\phi_{1,1}^2$, $\phi_{2,1}^2$)

> **Solution:** $a_1^2 = \phi_{1,1}^2(a_1^1) + \phi_{2,1}^2(a_2^1)$
> $= \phi_{1,1}^2\left(\phi_{1,1}^1(x_1) + \phi_{2,1}^1(x_2)\right) + \phi_{2,1}^2\left(\phi_{1,2}^1(x_1) + \phi_{2,2}^1(x_2)\right).$

(b) Consider a network with L layers: $L-1$ hidden layers and one output layer. Assume that the network input feature is $N-$dimensional (i.e. in equation (1), $m = N$). Assume also that every layer has N neurons. In other words, all layers have the same width N.

For reference, the LAF described in part (a) violates the equal-width assumption.

i. Given the network specification (i.e., L layers of equal width N), how many learnable parameters does our typical *feed-forward neural networks* have?
(We expect your answer to be a function of L and N.)

> **Solution:** $L(N^2 + N)$

ii. Given the network specification (i.e., L layers of equal width N), how many learnable parameters does a *LAF* have?
(We expect your answer to be a function of L and N.)

> **Solution:** $L \times 2N^2$.

(c) As part of backpropagation through the LAF, we will need to compute gradient updates with respect to all layer inputs and parameters. For convenience, we repeat the activation equation below:

$$a = \sum_{j=1}^{m} \phi_j(x_j) = \sum_{j=1}^{m} \left(b_j g(x_j) + c_j h(x_j)\right).$$

For all subparts below, you may express your answers as functions of g and h as needed.

i. What is $\partial a/\partial b_j$ for any given $j = 1, \ldots, m$?

> **Solution:** $\partial a/\partial b_j = g(x_j)$.

ii. What is $\partial a/\partial c_j$ for any given $j = 1, \ldots, m$?

> **Solution:** $\partial a/\partial c_j = h(x_j)$.

iii. What is $\partial a/\partial x_j$ for any given $j = 1, \ldots, m$?

> **Solution:** $\partial a/\partial x_j = b_j \frac{\partial g(x_j)}{\partial x_j} + c_j \frac{\partial h(x_j)}{\partial x_j}$.

This is the end of the exam.