Solutions

10-601 Machine Learning
Spring 2025
Practice Problems
Updated: March 15, 2025
Time Limit: N/A

Name:
Andrew ID:
Room:
Seat:
Exam Number:

Instructions:

- Verify your name and Andrew ID above.
- This exam contains 36 pages (including this cover page).
  The total number of points is 0.
- Clearly mark your answers in the allocated space. If you have made a mistake, cross out the invalid parts of your solution, and circle the ones which should be graded.
- Look over the exam first to make sure that none of the 36 pages are missing.
- No electronic devices may be used during the exam.
- Please write all answers in pen or *darkly* in pencil.
- You have N/A to complete the exam. Good luck!

Question	Points
1. Optimization	0
2. Logistic Regression and Regularization	0
3. Feature Engineering and Regularization	0
4. Neural Networks	0
5. Algorithmic Bias	0
6. Learning Theory	0
7. MLE/MAP	0
Total:	0

Instructions for Specific Problem Types

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

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie

○ Noam Chomsky

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

Select One: Who taught this course?

● Henry Chai

○ Marie Curie

⊗ Noam Chomsky

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

Select all that apply: Which are instructors for this course?

■ Matt Gormley

■ Henry Chai

□ I don't know

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

Select all that apply: Which are the instructors for this course?

■ Matt Gormley

■ Henry Chai

⊠ I don't know

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

Fill in the blank: What is the course number?

10-601 10-6301

1 Optimization (0 points)

1.1. (a) **Select all that apply:** Determine if the following 1-D functions are convex. Assume that the domain of each function is $\mathbb{R}$. The definition of a convex function is as follows:

$f(x)$ is convex $\iff$ $f(\alpha x+(1-\alpha)z) \leq \alpha f(x)+(1-\alpha)f(z), \forall \alpha \in [0,1]$ and $\forall x, z$.

- ☐ $f(x) = x + b$ for any $b \in \mathbb{R}$
- ☐ $f(x) = c^2 x$ for any $c \in \mathbb{R}$
- ☐ $f(x) = ax^2 + b$ for any $a \in \mathbb{R}$ and any $b \in \mathbb{R}$
- ☐ $f(x) = 0$
- ☐ None of the above

A, B, D

C is nonconvex for $a < 0$

(b) **Select all that apply:** Suppose we are trying to minimize the convex function $f(z) = z^2$ using gradient descent. Let α be the learning rate and assume that we use an inital value of $z^{(0)} = 1$.

For which values of α will graident descent converge to the optimal value, $x^* = 0$?

- ☐ $\alpha = 0$
- ☐ $\alpha = \frac{1}{2}$
- ☐ $\alpha = 1$
- ☐ $\alpha = 2$
- ☐ None of the above

$\alpha = \frac{1}{2}$

$\alpha = 1$ causes gradient descent to bounce between 1 and -1 while $\alpha = 2$ causes gradient descent to diverge.

1.2. **Fill in the Blanks:** Complete the following sentence by circling the best option in each square (options are separated by "/"s):

The mean squared error objective function for linear regression is [non-convex / convex / strictly convex] which means that gradient descent will converge to a [global minimum / local minimum which might not be a global minimum] if it converges; however, a global minimum [does not / does] always exist.

The mean squared error objective function for linear regression is convex which means that gradient descent will converge to a global minimum; however, a global minimum does always exist.

1.3. **Select all that apply:** Which of the following statements about gradient descent (GD) and stochastic gradient descent (SGD) are correct?

- ☐ Each update step in SGD pushes the parameter vector closer to the parameter vector that minimizes the objective function.
- ☐ The gradient computed in SGD is, in expectation, equal to the gradient computed in GD.
- ☐ The gradient computed in GD has a higher variance than that computed in SGD, which is why in practice SGD converges faster in time than GD.
- ☐ Both SGD and GD are guaranteed to converge if and only if the objective function is strictly convex.
- ☐ None of the above.

B.

A is incorrect, SGD updates are high in variance and may not go in the direction of the true gradient. C is incorrect, for the same reason. D is incorrect because GD can converge if the function is convex but not strictly convex.

1.4. **True or False:** For a given dataset and objective function, one *epoch* of stochastic gradient descent will always have the same big-O computational cost as one iteration of gradient descent.

- ◯ True
- ◯ False

True

2 Logistic Regression and Regularization (0 points)

2.1. **Math:** A generalization of logistic regression to a multiclass settings involves expressing the per-class probabilities $P(y = c \mid x)$ as the softmax function

$$\frac{\exp(w_c^T x)}{\sum_{d \in C} \exp(w_d^T x)},$$

where c is some class from the set of all classes C.

Consider a 2-class problem with labels 0 or 1. Rewrite the above expression for this situation to derive the expressions for $P(Y = 1|x)$ and $P(Y = 0|x)$ that we have already seen in class for binary logistic regression.

$P(y=1|x) = \frac{\exp(w_1^T x)}{\exp(w_0^T x)+\exp(w_1^T x)} = \frac{\exp((w_1-w_0)^T x)}{1+\exp((w_1-w_0)^T x)} = \frac{\exp(w^T x)}{1+\exp(w^T x)} = p$
Therefore, $1-p = \frac{1}{1+\exp(w^T x)}$

2.2. **Short answer:** Given 3 data points $(1,1),(1,0),(0,0)$ with labels $0,1,0$ respectively, consider 2 models that define $p(y = 1 \mid \mathbf{x})$:

$$\text{Model 1: } \sigma(w_1 x_1 + w_2 x_2)$$

$$\text{Model 2: } \sigma(w_0 + w_1 x_1 + w_2 x_2)$$

As usual, $\sigma(z)$ is the sigmoid function $\frac{1}{1+e^{-z}}$. Using the given dataset, suppose we learn parameters for both models, $\hat{\mathbf{w}}_1$ and $\hat{\mathbf{w}}_2$, by maximizing the conditional log-likelihood.

If we switched $(0,0)$ to label 1 instead of label 0, would the parameters we learn for Model 1 change? What about Model 2?

The parameters for Model 1 wouldn't change because $w_1x_1 + w_2x_2 = 0$ for $(0, 0)$. Hence $p = 0.5$ irrespective of the labels or the values of $\mathbf{w}$.

Model 2 has a bias term which remains non-zero for $(0, 0)$, and can thus change the model depending on the label assigned.

2.3. Given a training dataset $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^d$ is a feature vector and $y_i \in \{0, 1\}$ is a binary label, we want to find the parameters $\hat{\mathbf{w}}$ that maximize the likelihood of the training dataset, assuming a parametric model of the form

$$p(y = 1 \mid \mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}.$$

The conditional log likelihood of $\mathcal{D}$ is

$$\ell(\mathbf{w}) = \sum_{i=1}^{n} y_i \log p(y_i \mid \mathbf{x}_i; \mathbf{w}) + (1 - y_i) \log(1 - p(y_i \mid \mathbf{x}_i; \mathbf{w})),$$

and the gradient is

$$\nabla\ell(\mathbf{w}) = \sum_{i=1}^{N} (y_i - p(y_i \mid \mathbf{x}_i; \mathbf{w}))\mathbf{x}_i.$$

(a) **Short answer:** Is it possible to solve for the optimal parameters $\hat{\mathbf{w}}$ in closed form? If yes, explain how and if no, describe how you would compute $\hat{\mathbf{w}}$ in practice?

There is no closed form expression for maximizing the conditional log likelihood. One has to consider iterative optimization methods, such as gradient descent, to compute $\hat{w}$.

(b) **Math:** For a binary logistic regression model, we predict $y = 1$ when $p(y = 1 \mid \mathbf{x}) \geq 0.5$. Show that this is a linear classifier in $\mathbf{x}$.

Using the parametric form for $p(y = 1 \mid \mathbf{x})$:

$$\begin{aligned} p(y = 1 \mid \mathbf{x}) \geq \frac{1}{2} &\implies \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})} \geq \frac{1}{2} \\ &\implies 1 + \exp(-\mathbf{w}^T \mathbf{x}) \leq 2 \\ &\implies \exp(-\mathbf{w}^T \mathbf{x}) \leq 1 \\ &\implies -\mathbf{w}^T \mathbf{x} \leq 0 \\ &\implies \mathbf{w}^T \mathbf{x} \geq 0, \end{aligned}$$

so we predict $\hat{y} = 1$ if $\mathbf{w}^T \mathbf{x} \geq 0$, which precisely defines a linear function of $\mathbf{x}$.

(c) Consider the case where all features are binary, i.e, $\mathbf{x} \in \{0, 1\}^d$. Furthermore, suppose feature x_1 is rare and just happens to take on value 1 in the training set only when the label is also 1.

What is $\hat{w}_1$ i.e. the optimal weight on the first feature? Is the gradient ever zero for any finite value of w_1?

If a binary feature fired for only label 1 in the training set then, by maximizing the conditional log likelihood, we will make the weight associated to that feature be infinite; this implies that the gradient will never be zero: we can always improve the solution by increasing the weight. This is because, when this feature is observed in the training set, we will want to predict 1 irrespective of everything else. This is an undesired behavior from the point of view of generalization performance, as most likely we do not believe this rare feature to have that much information about class 1. Most likely, it is a spurious co-occurrence. Controlling the norm of the weight vector will prevent these pathological cases.

2.4. **Math:** Given the following dataset, $\mathcal{D}$, and a fixed parameter vector, $\mathbf{w}$, write an expression for the conditional likelihood of a binary logistic regression model on this dataset.

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)} = 0), (\mathbf{x}^{(2)}, y^{(2)} = 0), (\mathbf{x}^{(3)}, y^{(3)} = 1), (\mathbf{x}^{(4)}, y^{(4)} = 1)\}$$

- Write your answer in terms of $\mathbf{w}$, $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$, and $\mathbf{x}^{(4)}$.
- Do not include $y^{(1)}$, $y^{(2)}$, $y^{(3)}$, or $y^{(4)}$ in your answer.
- Do not try to simplify your expression.

$\left(1 - \frac{1}{1+e^{-\mathbf{w}^T x^1}}\right)\left(1 - \frac{1}{1+e^{-\mathbf{w}^T x^2}}\right)\frac{1}{1+e^{-\mathbf{w}^T x^3}}\frac{1}{1+e^{-\mathbf{w}^T x^4}}$

2.5. Suppose we apply feature engineering to a two-dimensional input, $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$, mapping it to a new input vector: $\mathbf{x} = \begin{bmatrix} 1 \\ {x_1}^2 \\ {x_2}^2 \end{bmatrix}$

(a) Write an expression for the decision boundary of binary logistic regression with this feature vector $\mathbf{x}$ and the corresponding parameter vector $\boldsymbol{\theta} = [b, w_1, w_2]^T$. Assume that the decision boundary occurs when $P(Y = 1 \mid x, \boldsymbol{\theta}) = P(Y = 0 \mid x, \boldsymbol{\theta})$. Write your answer in terms of x_1, x_2, b, w_1, and w_2.

Decision boundary expression:

$0 = b + w_1 x_1^2 + w_2 x_2^2$.

(b) Assume that $w_1 > 0$, $w_2 > 0$, and $b < 0$. What is the geometric shape defined by this equation?

An ellipse

(c) If we add an L2 regularization term when learning $[w_1, w_2]^T$, what happens to the **parameters** as we increase the λ that scales this regularization term?

The magnitude of the parameters will decrease.

(d) If we add an L2 regularization term when learning $[w_1, w_2]^T$, what happens to the **decision boundary shape** as we increase the λ that scales this regularization term?

The parameters shrink, so the ellipse will get bigger.

3 Feature Engineering and Regularization (0 points)

3.1. Suppose you have D-dimensional data points $\mathbf{x} = [x_1, x_2, \ldots, x_D]^T$ and you want to apply a 2-dimensional polynomial feature expansion to them, ϕ_2. This feature expansion contains all first-order terms as well as *all possible second-order terms*, including combinations of features i.e.

$$\{x_4, x_1^2, x_2 x_3\} \subset \phi_2(\mathbf{x})$$

(a) **Math:** What is the dimensionality of $\phi_2(\mathbf{x})$? Express you answer in terms of D, the dimensionality of $\mathbf{x}$.

D first order terms + D squared terms + $\binom{D}{2}$ combination terms = $\frac{D(D+3)}{2}$

(b) **Math:** Given a dataset consisting of N data points, suppose a unique closed-form solution exists for linear regression using the ϕ_2-transformation. What is the big-O computational cost of computing $(\Phi^T \Phi)^{-1}$, where Φ is the design matrix of the transformed data points i.e.,

$$\Phi = \begin{bmatrix} 1 & \phi_2(\mathbf{x}^{(1)}) \\ 1 & \phi_2(\mathbf{x}^{(2)}) \\ \vdots & \vdots \\ 1 & \phi_2(\mathbf{x}^{(N)}) \end{bmatrix}$$

Express you answer in terms N and D.

Hint: don't forget about the computational costs associated with the bias or intercept parameters.

$$O\left(N\left(\frac{D(D+3)}{2}+1\right)^2+\left(\frac{D(D+3)}{2}+1\right)^3\right)=O(ND^4+D^6)$$

3.2. **Model Complexity:** In this question we will consider the effect of increasing the model complexity, while keeping the size of the training dataset fixed. To be concrete, consider a classification task on the real line $\mathbb{R}$ with some unknown distribution over data points D and unknown target function $c^* : \mathbb{R} \to \{\pm 1\}$.

Suppose we have a randomly sampled dataset S of size N, drawn i.i.d. from D. For each degree d, let ϕ_d be the feature map given by $\phi_d(x) = (1, x, x^2, \ldots, x^d)$ that maps points on the real line to $(d+1)$-dimensional space.

Now consider the learning algorithm that first applies the feature map ϕ_d to all the training data points and then runs logistic regression. A new example is classified by first applying the feature map ϕ_d and then using the learned classifier.

(a) For a given dataset S, is it possible for the training error to increase when we increase the degree d of the feature map? Briefly justify your answer in 1 to 2 sentences.

No. Every linear separator using the feature map ϕ_d can also be expressed using the feature map ϕ_{d+1}, since we are only adding new features. It follows that the training error will not increase for any given sample S.

(b) Suppose we plot the true error of our algorithm as a function of d: we observe that it initially decreases and then increases as we increase the degree d. Briefly

explain this trend in 2 to 3 sentences.

When the dimension d is small, the true error is high because the target function is not well approximated by any linear separator in the ϕ_d feature space. As we increase d, our ability to approximate c^* improves, so the true error drops. But, as we continue to increase d, we begin to overfit the data and the true error increases again.

3.3. **Short Answer:** Your friend is training a logistic regression model with ridge regularization, where λ is the regularization constant. They run cross-validation for $\lambda = [0.01, 0.1, 1, 10]$ and compare train, validation and test errors. They choose $\lambda = 0.01$ because that had the lowest *test* error.

However, you observe that the test error linearly increases from $\lambda = 0.01$ to 10 and thus, there exists a value of $\lambda < 0.01$ that gives a lower test error. You tell your friend that they should run the cross-validation for $\lambda = [0.0001, 0.001, 0.01]$ to get the optimal model.

Do you think you did the right thing by giving your friend this suggestion? Briefly justify your answer in 1 to 2 concise sentences.

No. because we should not be using test error at all in making any model selection decisions.

3.4. **Select all that apply:** Suppose you fit a linear regression model with regularization to some high-dimensional dataset using gradient descent. You observe that it is overfitting to the training dataset (as measured by its test error rate on some held out test data). Which of the following actions would *probably* decrease the test error rate?

- ☐ Increase the amount of training data used to train your model

- ☐ Increase the regularization coefficient in the objective function
- ☐ Increase the number of iterations that you run gradient descent for
- ☐ Increase the dimensionality of your data by using a polynomial feature transformation
- ☐ None of the above

A, B

4 Neural Networks (0 points)

4.1. **Matching:** Match the corresponding neural network component to its role in the neural network.

(a) Cross-Entropy

(b) Identity

(c) Mean Absolute Error

(d) Mean Squared Error

(e) ReLU

(f) Sigmoid

(g) Softmax

(h) Stochastic Gradient Descent

(i) Tanh

Activation Function

Loss Function

Optimizer

Activation function: Identity, ReLU, Sigmoid, Tanh, Softmax; Loss function: Cross-entropy, Mean Absolute Error, Mean Squared Error; Optimizer: Stochastic Gradient Descent

4.2. Consider the neural network architecture shown above for a binary classification problem. The values for the weights are shown in the figure. We define:

$a_1 = w_{11}x_1 + b_{11}$

$a_2 = w_{12}x_1 + b_{12}$

$a_3 = w_{21}z_1 + w_{22}z_2 + b_{21}$

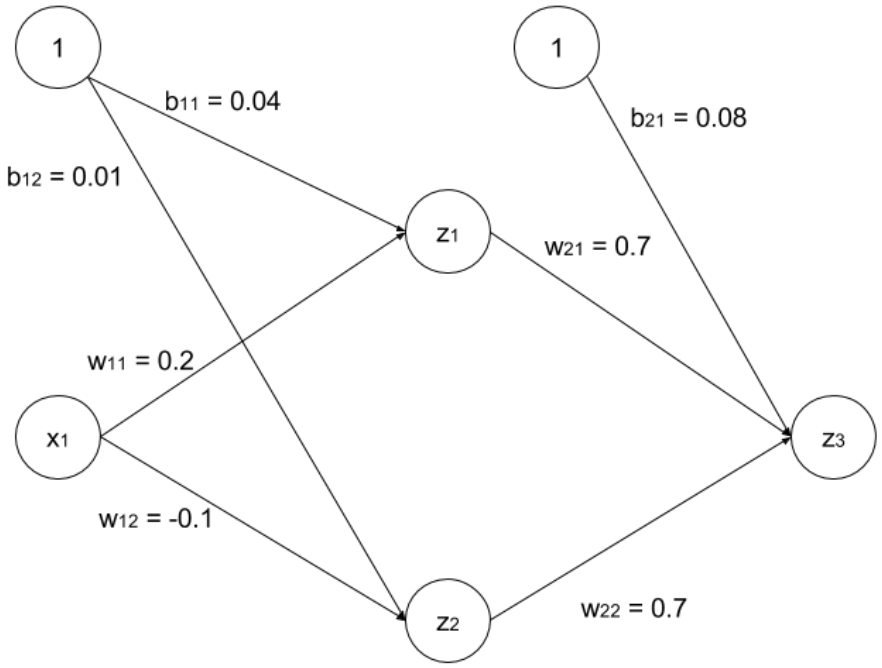


Figure 1: Neural Network

$z_1 = \text{ReLU}(a_1)$

$z_2 = \text{ReLU}(a_2)$

$z_3 = \sigma(a_3)$, $\sigma(x) = \frac{1}{1+e^{-x}}$

where $\text{ReLU}(x) = \max(0, x)$.

(a) **Math:** For $x_1 = 0.3$, compute z_3 in terms of e.

$z_3 = \frac{1}{1+e^{-0.15}}$

(b) **Select one:** Which class does the network predict for the data point $x_1 = 0.3$, assuming that $\hat{y} = 1$ if $z_3 > \frac{1}{2}$ and otherwise, $\hat{y} = 0$.

◯ 1

◯ 0

$\hat{y}(x_1 = 0.3) = 1$

(c) **Math:** Perform backpropagation on the bias term b_{21} by deriving the expression for the gradient of the loss function $L(y, z_3)$ with respect to the bias term

b_{21}, $\frac{\partial L}{\partial b_{21}}$.

Express your answer in terms of partial derivatives of the form $\frac{\partial \alpha}{\partial \beta}$, where α and β can be any of $L, z_i, a_i, b_{ij}, w_{ij}, x_1$ for all valid values of i, j. Your backpropagation algorithm should be as explicit as possible — that is, make sure each partial derivative $\frac{\partial \alpha}{\partial \beta}$ cannot be decomposed further into simpler partial derivatives. Do *not* evaluate the partial derivatives.

$$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial a_3}\frac{\partial a_3}{\partial b_{21}}$$

(d) **Math:** Perform backpropagation on the bias term b_{12} by deriving the expression for the gradient of the loss function $L(y, z_3)$ with respect to the bias term b_{12}, $\frac{\partial L}{\partial b_{12}}$.

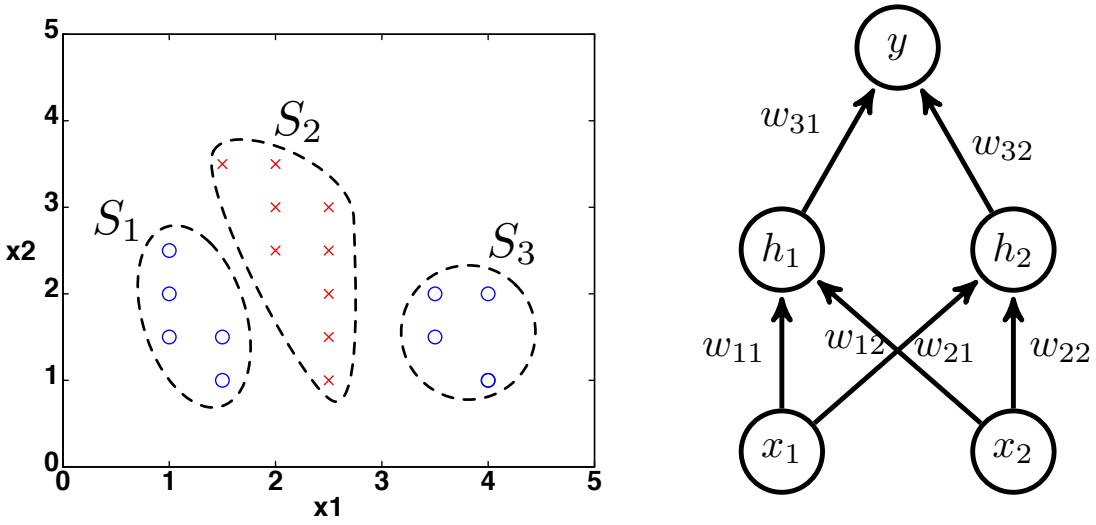
Express your answer in terms of partial derivatives of the form $\frac{\partial \alpha}{\partial \beta}$, where α and β can be any of $L, z_i, a_i, b_{ij}, w_{ij}, x_1$ for all valid values of i, j. Your backpropagation algorithm should be as explicit as possible — that is, make sure each partial derivative $\frac{\partial \alpha}{\partial \beta}$ cannot be decomposed further into simpler partial derivatives. Do *not* evaluate the partial derivatives.

$$\frac{\partial L}{\partial b_{12}} = \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial a_3}\frac{\partial a_3}{\partial z_2}\frac{\partial z_2}{\partial a_2}\frac{\partial a_2}{\partial b_{12}}$$

4.3. In this problem we will use a neural network to distinguish the crosses ($\times$) from the circles ($\circ$) in the simple data set shown in Figure 2a. Even though the crosses and circles are not linearly separable, we can break the examples into three groups, S_1, S_2, and S_3 (shown in Figure 2a) so that S_1 is linearly separable from S_2 and S_2 is linearly separable from S_3. We will exploit this fact to design weights for the neural network shown in Figure 2b in order to correctly classify this training dataset. For

all nodes, we will use the threshold activation function

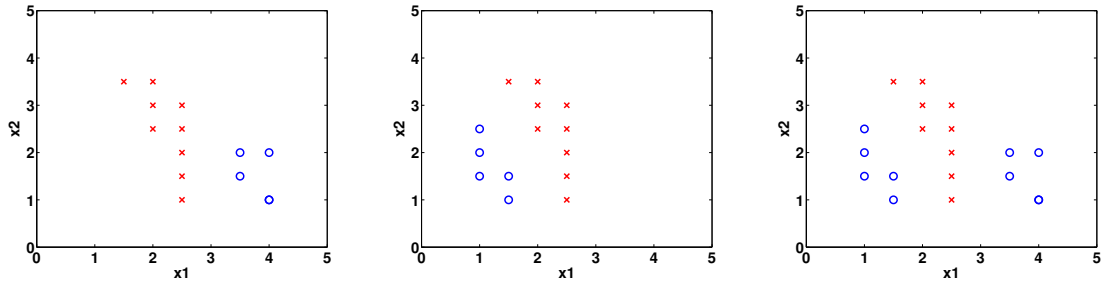
$$\phi(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0. \end{cases}$$



(a) The data set with groups S_1, S_2, and S_3.

(b) The neural network architecture

Figure 2



(a) Set S2 and S3

(b) Set S1 and S2

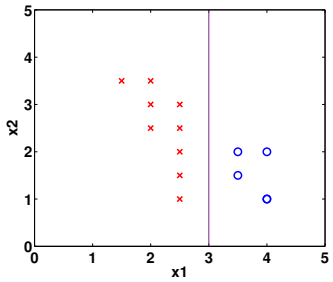
(c) Set S1, S2 and S3

Figure 3: NN classification.

(a) First we will set the parameters w_{11}, w_{12} and b_1 of the neuron labeled h_1 so that its output $h_1(x) = \phi(w_{11}x_1 + w_{12}x_2 + b_1)$ forms a linear separator between the sets S_2 and S_3.

i. On Fig 3a, draw a linear decision boundary that separates S_2 and S_3.

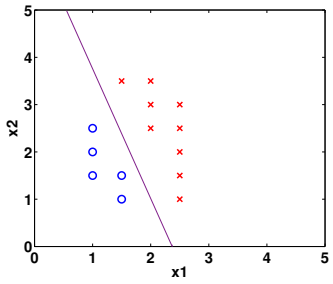
ii. Write down a possible setting of the weights w_{11}, w_{12}, and b_1 such that $h_1(x) = 0$ for all points in S_3 and $h_1(x) = 1$ for all points in S_2.



$w_{11} = -1, w_{12} = 0, b_1 = 3$

(b) Next we will set the parameters w_{21}, w_{22} and b_2 of the neuron labeled h_2 so that its output $h_2(x) = \phi(w_{21}x_1 + w_{22}x_2 + b_2)$ forms a linear separator between the sets S_1 and S_2.

i. On Fig 3b, draw a linear decision boundary that separates S_1 and S_2.



ii. Write down a possible setting of the weights w_{21}, w_{22}, and b_2 such that $h_2(x) = 0$ for all points in S_1 and $h_2(x) = 1$ for all points in S_2.

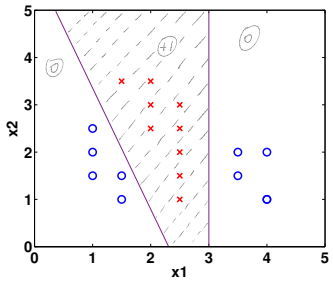
$w_{21} = 3, w_{22} = 1, b_2 = -7$

(c) Now we have two classifiers h_1 (to classify S_2 from S_3) and h_2 (to classify S_1 from S_2). We will set the weights of the final neuron of the neural network based on the results from h_1 and h_2 to classify the crosses from the circles. Let $h_3(x) = \phi\big(w_{31}h_1(x) + w_{32}h_2(x) + b_3\big)$.

i. Write down a possible setting of the weights w_{31}, w_{32}, b_3 such that $h_3(x)$ correctly classifies the entire data set.

$w_{31} = 1, w_{32} = 1, b_3 = -1.5$

ii. Draw your decision boundary in Fig 3c.



4.4. Consider the following neural network for a 2-D input, $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ where:

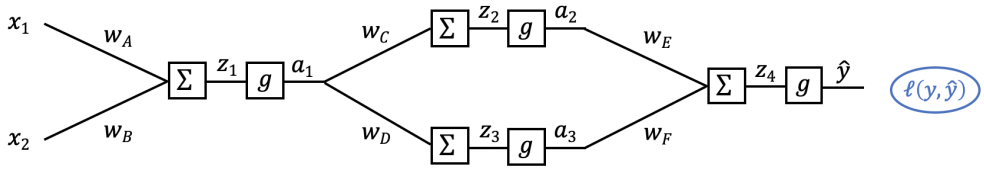


Figure 7: Neural Network

- All occurrences of the function g are the same arbitrary non-linear activation function with no parameters
- $\ell(y, \hat{y})$ is an arbitrary loss function with no parameters, and:

$$z_1 = w_A x_1 + w_B x_2,\ a_1 = g(z_1)$$

$$z_2 = w_C a_1,\ a_2 = g(z_2)$$

$$z_3 = w_D a_1,\ a_3 = g(z_3)$$

$$z_4 = w_E a_2 + w_F a_3,\ \hat{y} = g(z_4)$$

Note: There are no bias terms in this network.

(a) What is the chain of partial derivatives needed to calculate the derivative $\frac{\ell}{w_E}$?

Your answer should be in the form: $\frac{\ell}{w_E} = \frac{?}{?}\frac{?}{?}\ldots$ Make sure each partial derivative $\frac{?}{?}$ in your answer cannot be decomposed further into simpler partial derivatives. **Do not evaluate the derivatives.** Be sure to specify the correct subscripts in your answer.

$\frac{\ell}{w_E} =$

$\frac{\ell}{w_E} = \frac{\ell}{\hat{y}} \frac{\hat{y}}{z_4} \frac{z_4}{w_E}$

(b) The network diagram from above is repeated here for convenience: What is the

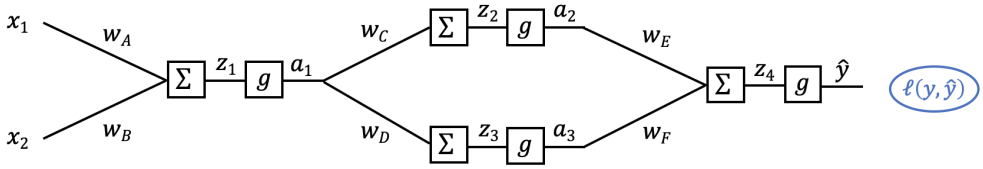


Figure 8: Neural Network

chain of partial deriviatives needed to calculate the derivative $\frac{\ell}{w_C}$? Your answer should be in the form:

$$\frac{\ell}{w_C} = \frac{?}{?}\frac{?}{?}\cdots$$

Make sure each partial derivative $\frac{?}{?}$ in your answer cannot be decomposed further into simpler partial derivatives. **Do not evaluate the derivatives**. Be sure to specify the correct superscripts in your answer.

$\frac{\ell}{w_C} =$

$\frac{\ell}{w_C} = \frac{\ell}{\hat{y}} \frac{\hat{y}}{z_4} \frac{z_4}{a_2} \frac{a_2}{z_2} \frac{z_2}{w_C}$

(c) We want to modify our neural network objective function to add an L2 regularization term on the weights. The new objective is:

$$\ell(y, \hat{y}) + \lambda \frac{1}{2} \|w\|_2^2$$

where λ (lambda) is the regularization hyperparamter and $\mathbf{w}$ is all of the weights in the neural network stacked into a single vector, $\mathbf{w} = [w_A, w_B, w_C, w_D, w_E, w_F]^T$.

Write the right-hand side of the new gradient descent update step for weight w_C given this new objective function. You may use $\frac{\ell}{w_C}$ in your answer.

Update: $w_C \leftarrow \ldots$

Update for w_C: $w_C \leftarrow w_C - \alpha\left(\frac{\ell}{w_C} + \lambda w_C\right)$

4.5. Backpropagation in neural networks can lead to slow or unstable learning because of the vanishing or exploding gradients problem. Understandably, Neural the Narwhal does not believe this. To convince Neural, Lamar Jackson uses the example of an N layer neural network that takes in a scalar input x, and where each layer consists of a single neuron. More formally, $x = o_0$, and for each layer $i \in \{1, 2, ..., N\}$, we have

$$s_i = w_i o_{i-1} + b_i$$
$$o_i = \sigma(s_i)$$

where σ is the sigmoid activation function. Note that w_i, b_i, o_i, s_i are all scalars.

(a) Give an expression for $\frac{\partial o_N}{\partial w_1}$. Your expression should be in terms of the s_i's, the w_i's, N, x_i, and $\sigma'(\cdot)$, the derivative of the sigmoid function.

$$\frac{\partial o_N}{\partial w_1} = \frac{\partial o_N}{\partial o_{N-1}}\frac{\partial o_{N-1}}{\partial o_{N-2}} \cdots \frac{\partial o_1}{\partial w_1}$$
$$= \frac{\partial o_1}{\partial w_1}\prod_{i=2}^{N}\frac{\partial o_i}{\partial o_{i-1}}$$
$$= \sigma'(s_1)x\prod_{i=2}^{N}\sigma'(s_i)w_i$$

(b) Knowing that $\sigma'(\cdot)$ is at most $\frac{1}{4}$ and supposing that all the weights are 1 (i.e. $w_i = 1$ for all i), give an upper bound for $\frac{\partial o_N}{\partial w_1}$. Your answer should be in terms of x and N.

$\frac{\partial o_N}{\partial w_1} \leq x \left(\frac{1}{4}\right)^N$

4.6. Define a function $\texttt{floor} : \mathbb{R}_n \to \mathbb{R}_n$ such that

$$\texttt{floor}(\mathbf{z}) = \left[\lfloor z_i \rfloor \text{ for } 0 \leq i \leq D\right]^T$$

or essentially, a function that produces an output vector by applying $\lfloor \cdot \rfloor$ element-wise to the input vector.

Neural wants to use this function as an activation function to train his neural network. Is this possible? Explain why or why not.

Yes, it is possible. Since the function is piecewise, we will not be able to use automatic differentiation to solve the gradients, but we can still use the finite difference method to approximate the gradient and train the model.

5 Algorithmic Bias (0 points)

5.1. **Numerical answer:** Suppose you have binary classification dataset where 20% of the data points have label $+1$ and the remaining 80% have label -1. What are the positive predictive value (a.k.a. precision) and true positive rate (a.k.a. recall) of a classifier that always predicts $+1$?

precision = 0.2, true positive rate = 1

5.2. **Select all that apply:** Suppose you have a classifier h that is 100% accurate at some binary classification task. Furthermore, suppose that there is some protected attribute, A, and the percentage of positive labels is *constant* across different values of A. In this setting, which of the following definitions of algorithmic fairness is satisfied by h?

- ☐ Independence
- ☐ Separation
- ☐ Sufficiency
- ☐ None of the above.

A, B and C: A is correct because the distribution of labels is the same across different values of A and h will predict exactly that distribution. Similarly, B is correct because conditioned on the label, the distribution of h's prediction is the same across different values of A. Finally, C is also true because h is always correct so the conditional distribution of the label is pure across all different values of A.

5.3. **Select all that apply:** Suppose you have a classifier h that simply returns a random label for some binary classification task. Furthermore, suppose that there is some protected attribute, A, and the percentage of positive labels varies across different values of A. In this setting, which of the following definitions of algorithmic fairness is satisfied by h?

- ☐ Independence
- ☐ Separation
- ☐ Sufficiency
- ☐ None of the above.

A and B: because the classifier's predictions do not depend on any feature or the label, it will always be independent of A. However, the true label and A may not

be independent (as the base rates vary across different values of A) so sufficiency may not be satisfied.

5.4. **True or False:** Given a binary classification task with a binary protected attribute, A, it is *never* possible for a classifier to satisfy both separation and sufficiency with respect to A. Briefly justify your answer in 1-2 concise sentences.

False: if the baseline rates of the label across both values of A are equal, then separation and sufficiency can be achieved simultaneously.

6 Learning Theory (0 points)

6.1. Consider a classification problem with an unknown distribution over data points D and an unknown target function $c^* : \mathcal{R}^d \mapsto \pm 1$. For any sample of points S drawn from D, answer whether the following statements are true or false, along with a brief explanation.

(a) **True or False:** For a given hypothesis space $\mathcal{H}$, it is always possible to define a sufficient number of examples in S such that the true error is within a margin of ϵ of the sample error for all hypotheses $h \in H$ with a given probability.

False. If $VC(\mathcal{H}) = \infty$, then there is no (finite) number of examples sufficient to satisfy the PAC bound.

(b) **True or False:** The true error of any hypothesis h is an *upper* bound on its training error on the sample S.

False. We said true error is close to training error, but it might be smaller than training error, so it is not an upper bound.

6.2. **Short answer:** Briefly describe the difference between the realizable case and agnostic case of PAC learning?

Realizable- the true classifier c^* is in $\mathcal{H}$.

Agnostic- we don't know whether c^* is in $\mathcal{H}$. It may or may not be.

6.3. **True or False:** Consider two finite hypothesis sets $\mathcal{H}_1$ and $\mathcal{H}_2$ such that $\mathcal{H}_1 \subset \mathcal{H}_2$. Let $h_1 = \arg\min_{h \in \mathcal{H}_1} err_S(h)$ and $h_2 = \arg\min_{h \in \mathcal{H}_2} err_S(h)$.

Because $|\mathcal{H}_2| \geq |\mathcal{H}_1|$, $err_D(h_2) \geq err_D(h_1)$. Briefly justify your answer

False. Since there are more hypotheses in $\mathcal{H}_2$ there might be one that better fits the data than those in $\mathcal{H}_1$.

6.4. **Fill in the Blanks:** Complete the following sentence by circling one option in each square (options are separated by "/"s):

In order to prove that the VC-dimension of a hypothesis set $\mathcal{H}$ is D, you must show that $\mathcal{H}$ [can / cannot] shatter [any set / some set / multiple sets] of D data points and [can / cannot] shatter [any set / some set / multiple sets] of $D+1$ data points.

In order to prove that the VC-dimension of a hypothesis set $\mathcal{H}$ is D, you must show that $\mathcal{H}$ can shatter some set of D data points and cannot shatter any set of $D+1$ data points.

6.5. Consider the hypothesis set $\mathcal{H}$ consisting of all positive intervals in $\mathbb{R}$, i.e. all hypotheses of the form $h(x; a, b) = \begin{cases} +1 & \text{if } x \in [a, b] \\ -1 & \text{if } x \notin [a, b] \end{cases}$

(a) **Short Answer:** In 1-2 sentences, briefly justify why the VC dimension of $\mathcal{H}$ is less than 3.

We only need to show any 3 points cannot be shattered. Consider the case where the two outer points have label +1 and the middle point has label -1.

(b) **Select one:** What is the VC dimension of $\mathcal{H}$?

◯ 0

◯ 1

◯ 2

C

(c) **Numerical Answer:** Now, consider hypothesis sets $\mathcal{H}_k$ indexed by k, such that $\mathcal{H}_k$ consists of all hypotheses formed by k **non-overlapping** positive intervals in $\mathbb{R}$. Give an expression for the VC dimension of $\mathcal{H}_k$ in terms of k.

Hint: Think about how to repeatedly apply the result you found in Part (b).

$2k$

6.6. Your friend, who is taking an introductory ML course, is preparing to train a model for binary classification. Having just learned about PAC Learning, she informs you that for her given model choice, $\mathcal{H}$, she is in the finite, agnostic case.

Now she wants to know how changing certain values will change the sample complexity i.e., the number of labeled training data points required to satisfy the PAC criterion:

$$P\left(|R(h) - \hat{R}(h)| \leq \epsilon\right) \geq 1 - \delta \; \forall \; h \in \mathcal{H}$$

where $R(h)$ and $\hat{R}(h)$ are the expected and empirical risks respectively.

For each of the following changes, determine whether the sample complexity will increase, decrease, or stay the same.

(a) **Select one:** Using a simpler model (decreasing $|\mathcal{H}|$)

◯ Sample complexity will increase

◯ Sample complexity will decrease

◯ Sample complexity will stay the same

B

(b) **Select one:** Choosing a new hypothesis set $\mathcal{H}^*$, such that $|\mathcal{H}^*| = |\mathcal{H}|$

◯ Sample complexity will increase

◯ Sample complexity will decrease

◯ Sample complexity will stay the same

C

(c) **Select one:** Decreasing δ

◯ Sample complexity will increase

◯ Sample complexity will decrease

◯ Sample complexity will stay the same

A

(d) **Select one:** Decreasing ϵ

◯ Sample complexity will increase

◯ Sample complexity will decrease

◯ Sample complexity will stay the same

A

7 MLE/MAP (0 points)

7.1. Magnetic Resonance Imaging (MRI) scans are commonly used to generate detailed images of patients' internal anatomy at hospitals. The scanner returns an image with N pixels. For each pixel, we extract the noise from that pixel to obtain a vector of noise terms $\mathbf{x} \in \mathbb{R}^N$ s.t. $\forall\ i \in \{1...N\}$, $x_i \geq 0$ and x_i is independent and identically distributed and follows a Rayleigh distribution. The probability density function of a Rayleigh distribution is given by:

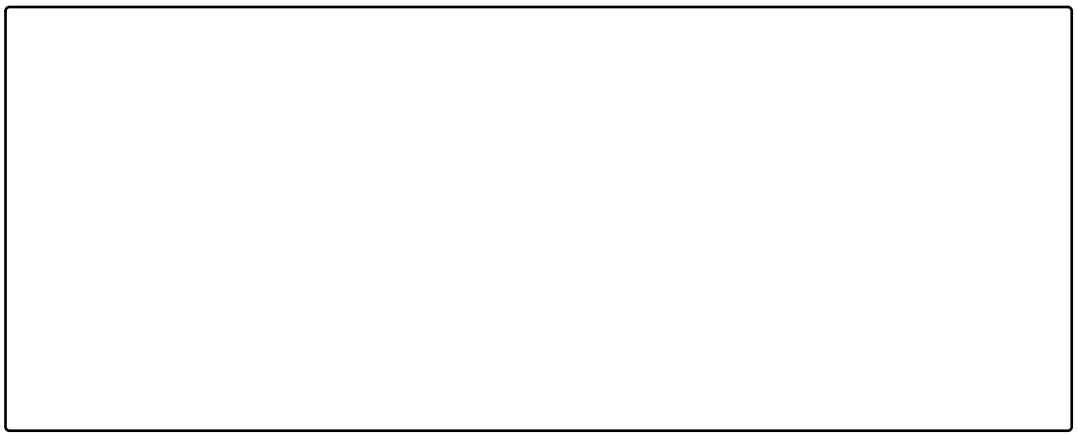
$$f(x \mid \sigma) = \frac{x}{\sigma^2} \exp\left(\frac{-x^2}{2\sigma^2}\right)$$

for scale parameter $\sigma \geq 0$ and $x \geq 0$.

(a) **Math:** Write the log-likelihood $\ell(\sigma)$ of a noise vector $\mathbf{x}$ obtained from one image. Report your answer in terms of the variables x_i, i, N, σ, the function $\exp(\cdot)$, and any constants you may need. For full credit you must push the log through to remove as many multiplications/divisions as possible.

$$\ell(\sigma) = \sum_{i=1}^{N} \left[\log x_i - 2\log\sigma - \frac{x_i^2}{2\sigma^2}\right]$$

(b) **Math:** Report the maximum likelihood estimator of the scale parameter, σ, for a single image's noise vector $\mathbf{x}$.



$$0 = \frac{\partial}{\partial \sigma} \sum_{i=1}^{N} \log p(x_i \mid \sigma) = \sum_{i=1}^{N} \frac{-2}{\sigma} + \frac{x_i^2}{\sigma^3}$$

$$\implies \hat{\sigma} = \left[\frac{1}{2N} \sum_{i=1}^{N} x_i^2 \right]^{\frac{1}{2}}$$

7.2. Suppose a random variable k follows a **Poisson distribution** with unknown rate parameter λ:

$$p(k \mid \lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \qquad k = 1, 2, \ldots$$

The Poisson distribution is used for modeling the number of times an event occurs within a fixed time interval given a mean occurrence rate assuming that the occurrences are independent.

A conjugate prior for the rate parameter in a Poisson likelihood is a **gamma distribution** with shape parameter α and rate parameter β:

$$f(\lambda \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \qquad \lambda > 0$$

where Γ is some normalizing constant that does not depend on λ.

(a) **Math:** Suppose we receive a set of N observations of k: $\mathcal{D} = \{k_1, k_2, \ldots, k_N\}$; assume the observations are independent and identically distributed. Using the Poisson distribution and gamma prior with parameters α and β, derive an expression for the unnormalized log posterior of λ i.e. the sum of the log prior and the log likelihood of $\mathcal{D}$.

Express your answer in terms of α, β, λ, $k_1, k_2, \ldots, k_N$, N and Γ; simplify your answer as much as possible.

$$\log p(\lambda \mid \alpha, \beta) + \log \prod_{i=1}^{N} P(k_i \mid \lambda) = \log \left(\frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \right) + \sum_{i=1}^{N} \log \left(\frac{\lambda^{k_i} e^{-\lambda}}{k_i!} \right)$$

$$= \alpha \log \beta + (\alpha - 1) \log \lambda - \beta\lambda - \log \Gamma(\alpha) - N\lambda + \sum_{i=1}^{N} k_i \log \lambda - \log(k_i!)$$

(b) **Math:** Take the partial derivative of the expression you derived in the previous part with respect to λ.

Express your answer in terms of α, β, λ, $k_1, k_2, \ldots, k_N$, and N; simplify your answer as much as possible.

$$\frac{\alpha - 1}{\lambda} - \beta - N + \sum_{i=1}^{N} \frac{k_i}{\lambda}$$

(c) **Math:** Finally, compute the MAP estimate of λ given $\mathcal{D}$ by setting the partial derivative you computed in the previous part equal to 0 and solving for λ.

Express your answer in terms of α, β, $k_1, k_2, \ldots, k_N$, and N; simplify your answer as much as possible.

$$\frac{\alpha-1}{\hat{\lambda}} - \beta - N + \sum_{i=1}^{N} \frac{k_i}{\hat{\lambda}} = 0$$

$$\rightarrow \alpha - 1 + \sum_{i=1}^{N} k_i = (\beta + N)\hat{\lambda}$$

$$\rightarrow \hat{\lambda} = \frac{\alpha - 1 + \sum_{i=1}^{N} k_i}{\beta + N}$$

Intuitively, the MLE estimate for λ is the empirical mean of $\mathcal{D}$. The β and α parameters of the gamma distribution can be interpreted as the number of "pseudo-samples" previously observed and the total number of "hits" observed in those pseudo-samples respectively.

7.3. **True or False:** A random variable x follows a probability distribution with a single, real-valued parameter, θ. In the limit of infinitely many samples of x, the MAP estimate of θ will *always* approach the MLE estimate, regardless of your choice of prior on θ. Briefly justify your answer in 1-2 sentences.

False: if the prior does not have support over the entire domain (e.g., a uniform prior over just the range [0,1]) and the MLE estimate falls outside the prior, then the MAP estimate will approach the boundary nearest to the MLE, not the MLE estimate itself.

Do not remove this page! Use this page for scratch work.

Do not remove this page! Use this page for scratch work.

Do not remove this page! Use this page for scratch work.

Do not remove this page! Use this page for scratch work.