Solutions

10-601 Machine Learning
Fall 2024
Exam 2 Practice Problems
Updated: October 27, 2024
Time Limit: N/A

Name:
AndrewID:

Instructions:

- Fill in your name and Andrew ID above. Be sure to write neatly, or you may not receive credit for your exam.
- Clearly mark your answers in the allocated space **on the front of each page.** If needed, use the back of a page for scratch space, but you will not get credit for anything written on the back of a page. If you have made a mistake, cross out the invalid parts of your solution, and circle the ones which should be graded.
- No electronic devices may be used during the exam.
- Please write all answers in pen.
- You have N/A to complete the exam. Good luck!

Instructions for Specific Problem Types

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

Select One: Who taught this course?

● Henry Chai

○ 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 scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

□ 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 scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

☒ 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-~~7~~601

1 Optimization

1. **Select all that apply:** Which of the following are correct regarding Gradient Descent (GD) and stochastic gradient descent (SGD)?

   □ 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.

   □ 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.

2. (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:

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

   □ $f(x) = x + b$ for any $b \in \mathbb{R}$

   □ $f(x) = c^2x$ 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

   $f(x) = x + b$ for any $b \in \mathbb{R}$, $f(x) = c^2x$ for any $c \in \mathbb{R}$, $f(x) = 0$.

   (b) **Select all that apply:** Consider the convex function $f(z) = z^2$. Let α be our learning rate in gradient descent.

   For which values of α will $\lim_{t\to\infty} f(z^{(t)}) = 0$, assuming the initial value of z is $z^{(0)} = 1$ and $z^{(t)}$ is the value of z after the t-th iteration of gradient descent?

   □ $\alpha = 0$

   □ $\alpha = \frac{1}{2}$

   □ $\alpha = 1$

   □ $\alpha = 2$

□ None of the above

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

(c) **Numerical answer:** Give the range of all values for $\alpha \geq 0$ such that $\lim_{t \to \infty} f(z^{(t)}) = 0$, assuming the initial value of z is $z^{(0)} = 1$.

$(0, 1)$.

2 Logistic Regression and Regularization

1. A generalization of logistic regression to a multiclass settings involves expressing the per-class probabilities $P(y = c|x)$ 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 (labels 0 or 1). Rewrite the above expression for this situation to end up with expressions for $P(Y = 1|x)$ and $P(Y = 0|x)$ that we have already come across 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. Given a training set $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}, \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$ 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 $\hat{w}$ that maximize the likelihood for the training set, assuming a parametric model of the form

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

   The conditional log likelihood of the training set is

   $$\ell(w) = \sum_{i=1}^{n} y_i \log p(y_i, |x_i; w) + (1 - y_i) \log(1 - p(y_i, |x_i; w)),$$

   and the gradient is

   $$\nabla \ell(w) = \sum_{i=1}^{n} (y_i - p(y_i|x_i; w))x_i.$$

   a) Is it possible to get a closed form for the parameters $\hat{w}$ that maximize the conditional log likelihood? How would you compute $\hat{w}$ 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) For a binary logistic regression model, we predict $y = 1$ when $p(y = 1|x) \geq 0.5$. Show that this is a linear classifier.

Using the parametric form for $p(y = 1|x)$:

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

so we predict $\hat{y} = 1$ if $w^T x \geq 0$.

c) Consider the case with binary features, i.e, $x \in \{0,1\}^d$, where feature x_1 is rare and happens to appear in the training set with only label 1. What is $\hat{w}_1$? Is the gradient ever zero for any finite w? Why is it important to include a regularization term to control the norm of $\hat{w}$?

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 is because, when this feature is observed in the training set, we will want to predict predict 1 irrespective of everything else. This is an undesired behaviour 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 spurious co-occurrence. Controlling the norm of the weight vector will prevent these pathological cases.

3. Given the following dataset, $\mathcal{D}$, and a fixed parameter vector, $\boldsymbol{\theta}$, write an expression for the binary logistic regression conditional likelihood.

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

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

   Conditional likelihood:

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

4. Write an expression for the decision boundary of binary logistic regression with a bias term for two-dimensional input features $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ and parameters b (the intercept parameter), w_1, and w_2. Assume that the decision boundary occurs when $P(Y = 1 \mid \mathbf{x}, b, w_1, w_2) = P(Y = 0 \mid \mathbf{x}, b, w_1, w_2)$.

(a) Write your answer in terms of x_1, x_2, b, w_1, and w_2.

Decision boundary equation:

$0 = b + w_1 x_1 + w_2 x_2$

(b) What is the geometric shape defined by this equation?

A line.

5. We have now feature engineered the 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.

6. **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 1-2 concise sentences.

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

3 Feature Engineering and Regularization

1. **Model Complexity:** In this question we will consider the effect of increasing the model complexity, while keeping the size of the training set fixed. To be concrete, consider a classification task on the real line $\mathbb{R}$ with distribution D and target function $c^* : \mathbb{R} \to \{\pm 1\}$, and suppose we have a random sample S of size n drawn iid 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 examples 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? **Please explain 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 cvv for any given sample S.

   b) Briefly **explain in 1 to 2 sentences** why the true error first drops and then increases as we increase the degree d. When the dimension d is small, the true error is high because it is not possible to 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.

4 Neural Networks

1. Match the corresponding neural network component to its role in the neural network.

   Cross-Entropy

   ELU

   Linear

   Mean Absolute Error

   Mean Squared Error

   ReLU

   Sigmoid

   Softmax

   Stochastic Gradient Descent

   Tanh

   Activation Function

   Loss Function

   Optimizer

   Layer

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

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

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

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

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

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

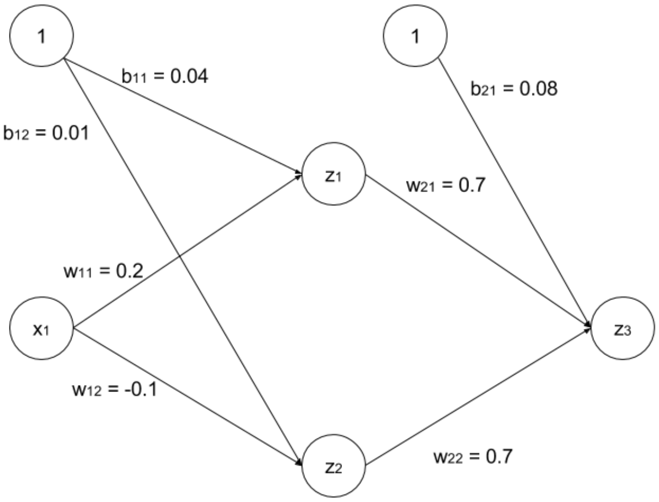


Figure 1: neural network

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

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

(i) For $x_1 = 0.3$, compute z_3 in terms of e.

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

(ii) Which class does the network predict for the data point ($x_1 = 0.3$)? Note that $\hat{y} = 1$ if $z_3 > \frac{1}{2}$, else $\hat{y} = 0$.

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

(iii) 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}, $\frac{\partial L}{\partial b_{21}}$, in terms of the partial derivatives $\frac{\partial \alpha}{\partial \beta}$, 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.

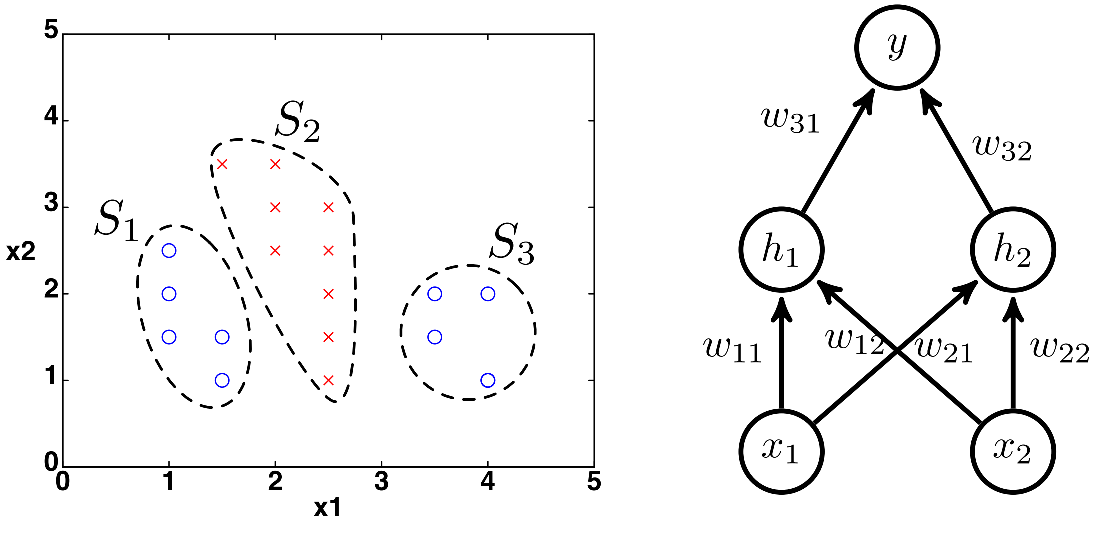
$$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial a_3}\frac{\partial a_3}{\partial b_{21}}$$

(iv) 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}, $\frac{\partial L}{\partial b_{12}}$, in terms of the partial derivatives $\frac{\partial \alpha}{\partial \beta}$, 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.

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

3. In this problem we will use a neural network to distinguish the crosses (×) from the circles (∘) 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 set. For all nodes, we will use the threshold activation function

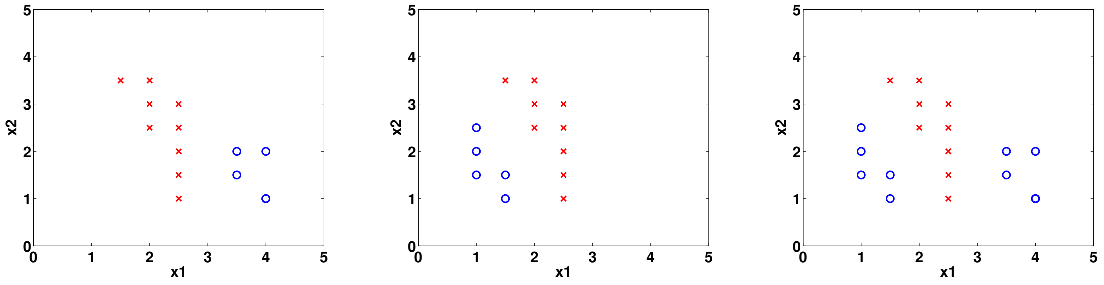
$$\phi(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0. \end{cases}$$



(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

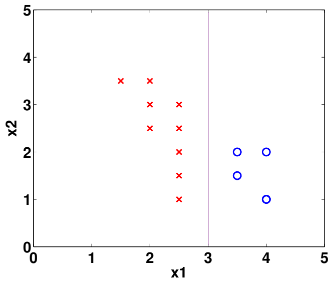
(b) Set S1 and S2

(c) Set S1, S2 and S3

Figure 3: NN classification.

(i) 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.

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

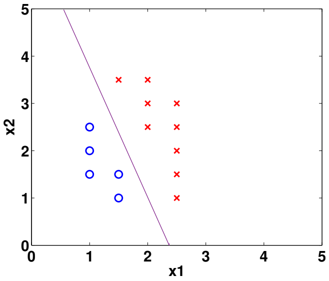


(b) Write down the corresponding weights w_{11}, w_{12}, and b_1 so that $h_1(x) = 0$ for all points in S_3 and $h_1(x) = 1$ for all points in S_2. One solution suffices and the same applies to (ii) and (iii).

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

(ii) Next we 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.

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



(b) Write down the corresponding weights w_{21}, w_{22}, and b_2 so that $h_2(x) = 0$ for all points in S_1 and $h_2(x) = 1$ for all points in S_2.

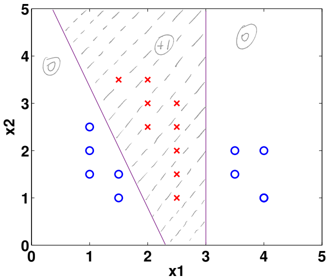
$w_{21} = 3, w_{22} = 1, b_2 = -7$

(iii) 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 $h_3(x) = \phi\big(w_{31}h_1(x) + w_{32}h_2(x) + b_3\big)$.

(a) Compute w_{31}, w_{32}, b_3 such that $h_3(x)$ correctly classifies the entire data set.

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

(b) Draw your decision boundary in Fig 3c.



4. One part of learning parameters in a neural network is getting the gradients of the parameters.

Suppose we have a dataset $\mathcal{D}$ with N data points x_i with label y_i, where $i \in [1, N]$. x_i is a $d \times 1$ vector and $y_i \in \{0, 1\}$. We use the data to train a neural network with one hidden layer:

$$\begin{aligned} h(x) =& \sigma(W_1 x + b_1) \\ p(x) =& \sigma(W_2 h(x) + b_2), \end{aligned}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function, W_1 is a n by d matrix, b_1 is a n by 1 vector, W_2 is a 1 by n matrix, and b_1 is a 1 by 1 vector.

We use cross entropy loss and minimize the negative log likelihood to train the neural network:

$$\ell_{\mathcal{D}}(W) = \frac{1}{N}\sum_{i=1}^{N} \ell_i(W) = \frac{1}{m}\sum_{i=1}^{N} -(y_i \log p_i + (1 - y_i)\log(1 - p_i)),$$

where $p_i = p(x_i), h_i = h(x_i)$.

(a) Describe how you would derive the gradients w.r.t the parameters W_1, W_2 and b_1, b_2. You do not need to write out the actual mathematical expression.

Use the chain rule.

(b) When N is large, we typically use a small subset of the dataset to estimate the gradient — stochastic gradient descent (SGD). Explain why we use SGD instead of gradient descent.

SGD converges faster than gradient descent.

(c) Derive expressions for the following gradients: $\frac{\partial l}{\partial p_i}, \frac{\partial l}{\partial W_2}, \frac{\partial l}{\partial b_2}, \frac{\partial l}{\partial h_i}, \frac{\partial l}{\partial W_1}, \frac{\partial l}{\partial b_1}$. When deriving the gradient w.r.t. the parameters in lower layers, you may assume the gradient in upper layers are available to you (i.e., you can use them in your equation). For example, when calculating $\frac{\partial l}{\partial W_1}$, you can assume $\frac{\partial l}{\partial p_i}, \frac{\partial l}{\partial W_2}, \frac{\partial l}{\partial b_2}, \frac{\partial l}{\partial h_i}$ are known.

$$
\begin{aligned}
\frac{\partial l}{\partial p_i} &= \frac{1}{m}\left(-\frac{y_i}{p_i} + \frac{1-y_i}{1-p_i}\right)\\
\frac{\partial l}{\partial W_2} &= \frac{1}{m}\sum_i \frac{\partial l_i}{\partial p_i}\frac{\partial p_i}{\partial W_2} = \frac{1}{m}\sum_i \frac{\partial l_i}{\partial p_i} p_i(1-p_i)h_i^T\\
\frac{\partial l}{\partial b_2} &= \frac{1}{m}\sum_i \frac{\partial l_i}{\partial p_i} p_i(1-p_i)\\
\frac{\partial l}{\partial h_i} &= \frac{\partial p_i}{\partial h_i}\frac{\partial l}{\partial p_i} = W_2^T p_i(1-p_i)\frac{\partial l}{\partial p_i}\\
\frac{\partial l}{\partial W_1} &= \frac{1}{m}\sum_i \frac{\partial l_i}{\partial h_i}\frac{\partial h_i}{\partial W_1} = \frac{1}{m}\sum_i \left[\frac{\partial l_i}{\partial h_i} \circ h_i \circ (1-h_i)\right] x_i^T\\
\frac{\partial l}{\partial b_1} &= \frac{1}{m}\sum_i \frac{\partial l_i}{\partial h_i}\frac{\partial h_i}{\partial b_1} = \frac{1}{m}\sum_i \frac{\partial l_i}{\partial h_i} \circ h_i \circ (1-h_i)
\end{aligned}
$$

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

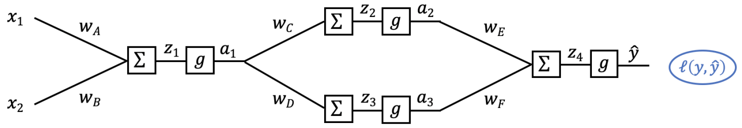


Figure 7: Neural Network

- All g functions 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 $\frac{\partial \ell}{\partial w_E}$?

Your answer should be in the form: $\frac{\partial \ell}{\partial w_E} = \frac{\partial ?}{\partial ?}\frac{\partial ?}{\partial ?}\ldots$ Make sure each partial derivative $\frac{\partial ?}{\partial ?}$ 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.

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

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

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

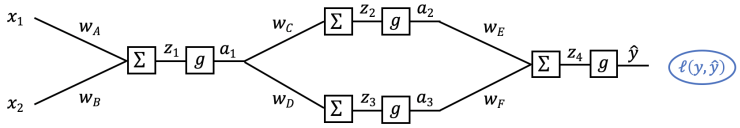


Figure 8: Neural Network

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

$$\frac{\partial \ell}{\partial w_C} = \frac{\partial ?}{\partial ?} \frac{\partial ?}{\partial ?} \ldots$$

Make sure each partial derivative $\frac{\partial ?}{\partial ?}$ 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.

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

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

(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 $\frac{\partial \ell}{\partial w_C}$ in your answer.

Update: $w_C \leftarrow \ldots$

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

6. 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.

i. (1 point) 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.

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

ii. (1 point) 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.

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

7. 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

1. 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, 1 - false positive rate) of a classifier that always predicts +1?

   precision = 0.2, true positive rate = 1

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.

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.

6 Learning Theory

1. **True and Sample Errors:** Consider a classification problem with distribution D and target function $c^* : \mathcal{R}^d \mapsto \pm 1$. For any sample 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.

2. Let X be the feature space and D be a distribution over X. We have a training data set

$$\mathcal{D} = \{(x_1, c^\star(x_1)), \cdots, ((x_N, c^\star(x_N)))\},$$

x_i i.i.d from D. We assume labels $c^\star(x_i) \in \{-1, 1\}$.

Let $\mathcal{H}$ be a hypothesis class and let $h \in \mathcal{H}$ be a hypothesis. In this question we restrict ourselves to $\mathcal{H}$. We use

$$err_S(h) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(h(x_i) \neq c^\star(x_i))$$

to denote the training error and

$$err_D(h) = P_{x\sim D}(h(x) \neq c^\star(x))$$

to denote the true error. Recall that if the concept class is finite, in the realizable case

$$m \geq \frac{1}{\epsilon}\left[\ln(|\mathcal{H}|) + \ln\left(\frac{1}{\delta}\right)\right]$$

labeled examples are sufficient so that with probability at least $1-\delta$, all $h \in \mathcal{H}$ with $err_D(h) \geq \epsilon$ have $err_S(h) > 0$; in the agnostic case,

$$m \geq \frac{1}{2\epsilon^2}\left[\ln(|\mathcal{H}|) + \ln\left(\frac{2}{\delta}\right)\right]$$

labeled examples are sufficient such that with probability at least $1-\delta$, all $h \in \mathcal{H}$ have $|err_D(h) - err_S(h)| < \epsilon$.

a) Briefly describe the difference between the realizable case and agnostic case.

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.

b) What is the full name of PAC learning? How do ϵ and δ tie into the name?

"Probably approximately correct." The hypotheses we find with m examples are *probably* (with probability $p \geq 1-\delta$) *approximately* correct, with $err_D(h) \leq \epsilon$

c) **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 $|\mathcal{H}_2| \geq |\mathcal{H}_1|$, $err_D(h_2) \geq err_D(h_1)$.

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

3. **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.

4. 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).

   □

   $2k$

5. **Select one:** 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 the model is in the finite, agnostic case.

   Now she wants to know how changing certain values will change the number of labelled training data points required to satisfy the PAC criterion. For each of the following changes, determine whether the sample complexity will increase, decrease, or stay the same.

   i. (1 point) Using a simpler model (decreasing $|\mathcal{H}|$)

   ○ Sample complexity will increase

○ Sample complexity will decrease

○ Sample complexity will stay the same

B

ii. (1 point) 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

iii. (1 point) Decreasing δ

○ Sample complexity will increase

○ Sample complexity will decrease

○ Sample complexity will stay the same

A

iv. (1 point) Decreasing ϵ

○ Sample complexity will increase

○ Sample complexity will decrease

○ Sample complexity will stay the same

A

7 MLE/MAP

1. **Math:** Let $X_1, X_2, ..., X_N$ be data drawn independently from a uniform distribution over a diamond-shaped area with edge length $\sqrt{2}\theta$ in $\mathbb{R}^2$, where $\theta \in \mathbb{R}^+$ (see Figure 9). Thus, $X_i \in \mathbb{R}^2$ and the distribution is

$$p(x|\theta) = \begin{cases} \frac{1}{2\theta^2} & \text{if } \|x\| \le \theta \\ 0 & \text{otherwise} \end{cases}$$

where $\|x\| = |x_1| + |x_2|$ is the L_1 norm. Find the maximum likelihood estimate of θ.

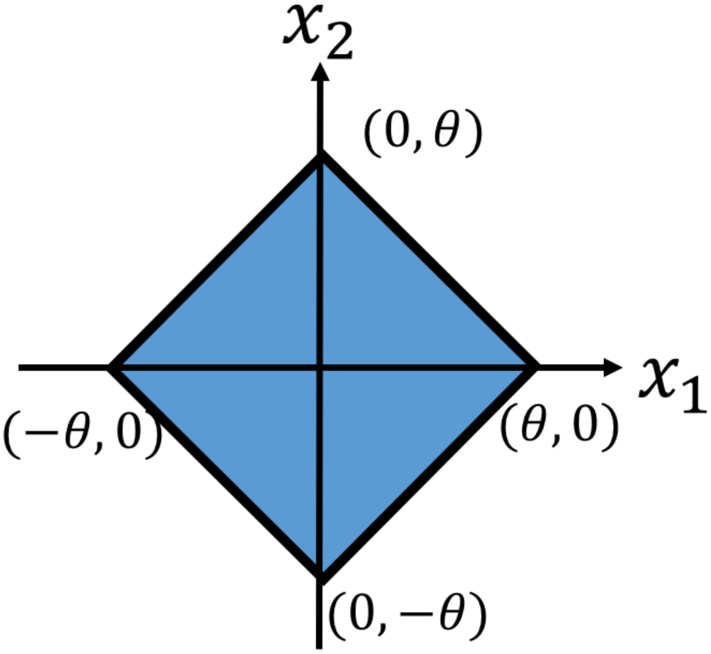


Figure 9: Area of $\|x\| \le \theta$

Analysis: As $X_1, X_2, ..., X_N$ are samples that have been drawn from the distribution, their probability of occurrence must be non-zero. Therefore, each of $X_1, X_2, ..., X_N$ must obey the condition $\|x\| \leq \theta$. Knowing this, we can write the likelihood function as

$$L(X_1, X_2, ..., X_N; \theta) = \frac{1}{(2\theta^2)^N} \mathbb{1}\left\{\underset{1 \leq i \leq N}{max} \|X_i\| \leq \theta\right\}$$

To maximize likelihood, we want θ to be as small as possible with the constraint of $\underset{1 \leq i \leq N}{max} \|X_i\| \leq \theta$, otherwise the likelihood drops to 0. So the MLE of θ is

$$\widehat{\theta} = \underset{1 \leq i \leq N}{max} \|X_i\|$$

2. **Math:** Suppose we want to model a 1-dimensional dataset of N real valued features $\left(x^{(i)}\right)$ and targets $\left(y^{(i)}\right)$ by:

$$y^{(i)} \sim \mathcal{N}\left(\exp(wx^{(i)}), 1\right),$$

where w is our unknown (scalar) parameter and $\mathcal{N}$ is the normal distribution with probability density function:

$$f(a)_{\mathcal{N}(\mu,\sigma^2)} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right)$$

Can the maximum conditional negative log likelihood estimator of w be solved analytically? If so, find the expression for w_{MLE}. If not, say so and write down the update rule for w using gradient descent.

Cannot be found analytically. The negative log likelihood can be defined as:

$$\text{NLL} = -\log\left(\prod_{i=1}^{n} \frac{\exp(-(y_i - \exp(wx_i))^2/2)}{\sqrt{2\pi}}\right) = -\log\left(\frac{\exp(-\sum_{i=1}^{n}(y_i - \exp(wx_i))^2/2)}{(\sqrt{2\pi})^n}\right)$$

This can be further simplified as

$$\text{NLL} = \sum_{i=1}^{n}(y_i - \exp(wx_i))^2/2 + \log((\sqrt{2\pi})^n)$$

Taking the derivative of the negative log likelihood with respect to w yields:

$$\frac{\partial \text{NLL}}{\partial w} = \sum_{i}^{N} -x^{(i)} y^{(i)} \exp(wx^{(i)}) + x^{(i)} \exp(2wx^{(i)})$$

Update rule is thus

$$w \leftarrow w - \eta \frac{\partial \text{NLL}}{\partial w}$$

3. Assume we have n random variables $x_i, i \in [1, n]$, each drawn independently from a Bernoulli distribution with mean θ. Recall that in a Bernoulli distribution $X \in \{0, 1\}$ and the pdf is:

$$p(X|\theta) = \theta^x (1 - \theta)^{1-x}$$

a) Derive the likelihood $L(\theta; X_1, \ldots, X_n)$.

$L(\theta; X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i; \theta)$
$L(\theta; X_1, \ldots, X_n) = \prod_{i=1}^{n} \theta^{x_i} (1 - \theta)^{1-x_i}$
$L(\theta; X_1, \ldots, X_n) = \theta^{\sum_{i=1}^{n} x_i} (1 - \theta)^{n - \sum_{i=1}^{n} x_i}$
Either of the final two steps are acceptable.

b) Show that the log-likelihood is:

$$l(\theta; X_1, \ldots, X_n) = \left(\sum_{i=1}^{n} X_i \right) \log(\theta) + \left(n - \sum_{i=1}^{n} X_i \right) \log(1 - \theta)$$

$l(\theta; X_1, \ldots, X_n) = \log L(\theta; X_1, \ldots, X_n)$
$l(\theta; X_1, \ldots, X_n) = \log \left[\theta^{\sum_{i=1}^n x_i}(1-\theta)^{n-\sum_{i=1}^n x_i}\right]$
$l(\theta; X_1, \ldots, X_n) = (\sum_{i=1}^n x_i) \log(\theta) + (n - \sum_{i=1}^n x_i) \log(1-\theta)$

c) Show that the MLE is $\hat{\theta} = \frac{1}{n}\sum\limits_{i=1}^{n} X_i$.

Take the derivative of the log likelihood and set it to zero
$\frac{dl}{d\theta} = \frac{d}{d\theta}\left[(\sum_{i=1}^n x_i) \log(\theta) + (n - \sum_{i=1}^n x_i) \log(1-\theta)\right] = 0$
$\frac{\sum_{i=1}^n x_i}{\theta} - \frac{n - \sum_{i=1}^n x_i}{1-\theta} = 0$
$\left(\sum_{i=1}^n x_i\right)(1-\theta) - \left(n - \sum_{i=1}^n x_i\right)\theta = 0$
$\sum_{i=1}^n x_i - n\theta = 0$
$\hat{\theta} = \frac{1}{n}(\sum_{i=1}^n X_i)$

4. 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 $x \geq 0$.

i. (2 points) 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.

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

ii. (2 points) Report the maximum likelihood estimator of the scale parameter, σ, for a single image's noise vector $\mathbf{x}$.

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

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