CS 475 Machine Learning: Homework 3 Analytical (35 points)

Assigned: Friday, September 24, 2021

Due: Friday, October 8, 2021, 11:59 pm US/Eastern

Partner 1: NAME (JHED), Partner 2: NAME (JHED)

Instructions

We have provided this LaTeX document for turning in this homework. We give you one or more boxes to answer each question. The question to answer for each box will be noted in the title of the box. You can change the size of the box if you need more space.

Other than your name, do not type anything outside the boxes. Leave the rest of the document unchanged.

Do not add text outside of the answer boxes. You are allowed to make boxes larger if needed.

We strongly recommend you review your answers in the generated PDF to ensure they appear correct. We will grade what appears in the answer boxes in the submitted PDF, NOT the original latex file.

1 Neural Networks

1. Consider a neural network model with n layers, where the vertex for each internal layer is determined by a ReLU activation function applied to a weighted combination of vertices of the previous layer, while the output layer is the given by a weighted linear function of the outputs of the previous layer (without ReLU).

   (a) Show that if we replace all ReLU units by the identity function $f(x) = x$, the resulting neural network yields a linear regression model of the inputs.

   (b) Is there a unique loss minimizer setting of weights for this model? Explain.

(a)
In a neural network, the j-th output at the i-th layer is given by $x_{ij} = g(\beta_0 + \beta_{\mathbf{i-1}}^{\mathbf{T}} \mathbf{x_{i-1}})$. In this case, the activation function is $g(x) = x$.
Thus, $x_{ij} = \beta_0 + \beta_{\mathbf{i-1}}^{\mathbf{T}} \mathbf{x_{i-1}}$. The output from the previous layer will be exactly the input for the next layer.
We want to show that if the input of a layer is a linear combination of model inputs, then the output of this layer is also a linear combination of model inputs.
Let $\mathbf{x_{i-1}} = \beta_{\mathbf{0,i-1}} + \mathbf{B_{i-1}} \mathbf{x_0}$, where $\mathbf{x_{i-1}}$ is the vector of output from layer i-1, $\beta_{\mathbf{0,i-1}}$ is the vector of biases, $\mathbf{B_{i-1}}$ is the matrix whose rows are the linear combination coefficients for each output of the layer, and $\mathbf{x_0}$ is the model inputs.
Then, for each output x_{ij} in layer i,

$$\begin{aligned} x_{ij} &= \beta_{0,ij} + \beta_{\mathbf{ij}}^{\mathbf{T}} \mathbf{x_{i-1}} \\ &= \beta_{0,ij} + \beta_{\mathbf{ij}}^{\mathbf{T}} (\beta_{\mathbf{0,i-1}} + \mathbf{B_{i-1}} \mathbf{x_0}) \\ &= (\beta_{0,ij} + \beta_{\mathbf{ij}}^{\mathbf{T}} \beta_{\mathbf{0,i-1}}) + (\beta_{\mathbf{ij}}^{\mathbf{T}} \mathbf{B_{i-1}}) \mathbf{x_0} \end{aligned}$$

Which is a linear combination of $\mathbf{x_0}$.
The whold neural model is but stacking of such layers. Since the each of model inputs is obviously a linear combination of itself, the model output is also a linear combination of the inputs.

(b)
Assume that we are talking about the model discussed in part (a).
For an arbitrary type of loss, assume that we now have a setting of weights that minimizes the loss. If we now multiply all the β_0 and $\beta_{\mathbf{i-1}}^{\mathbf{T}}$ in layer i-1 by λ, and divide all the $\beta_{\mathbf{i}}^{\mathbf{T}}$ in layer i by λ, the final output of the model will not change because all the layers are linear combinations of the previous layers. Thus, the modified weights also minimize the loss. Thus, loss minimizer setting of weights is not unique. We can prove this quickly:

If $\mathbf{x_{i-1}} = \lambda(\beta_{\mathbf{0,i-1}} + \mathbf{B_{i-1}} \mathbf{x_0})$, $x_{ij} = \beta_0 + \frac{1}{\lambda} \beta_{\mathbf{i-1}}^{\mathbf{T}} \mathbf{x_{i-1}}$

$$\begin{aligned} x_{ij} &= \beta_{0,ij} + \frac{1}{\lambda} \beta_{\mathbf{ij}}^{\mathbf{T}} \mathbf{x_{i-1}} \\ &= \beta_{0,ij} + \frac{1}{\lambda} \beta_{\mathbf{ij}}^{\mathbf{T}} \lambda (\beta_{\mathbf{0,i-1}} + \mathbf{B_{i-1}} \mathbf{x_0}) \\ &= (\beta_{0,ij} + \beta_{\mathbf{ij}}^{\mathbf{T}} \beta_{\mathbf{0,i-1}}) + (\beta_{\mathbf{ij}}^{\mathbf{T}} \mathbf{B_{i-1}}) \mathbf{x_0} \end{aligned}$$

Which does not change.

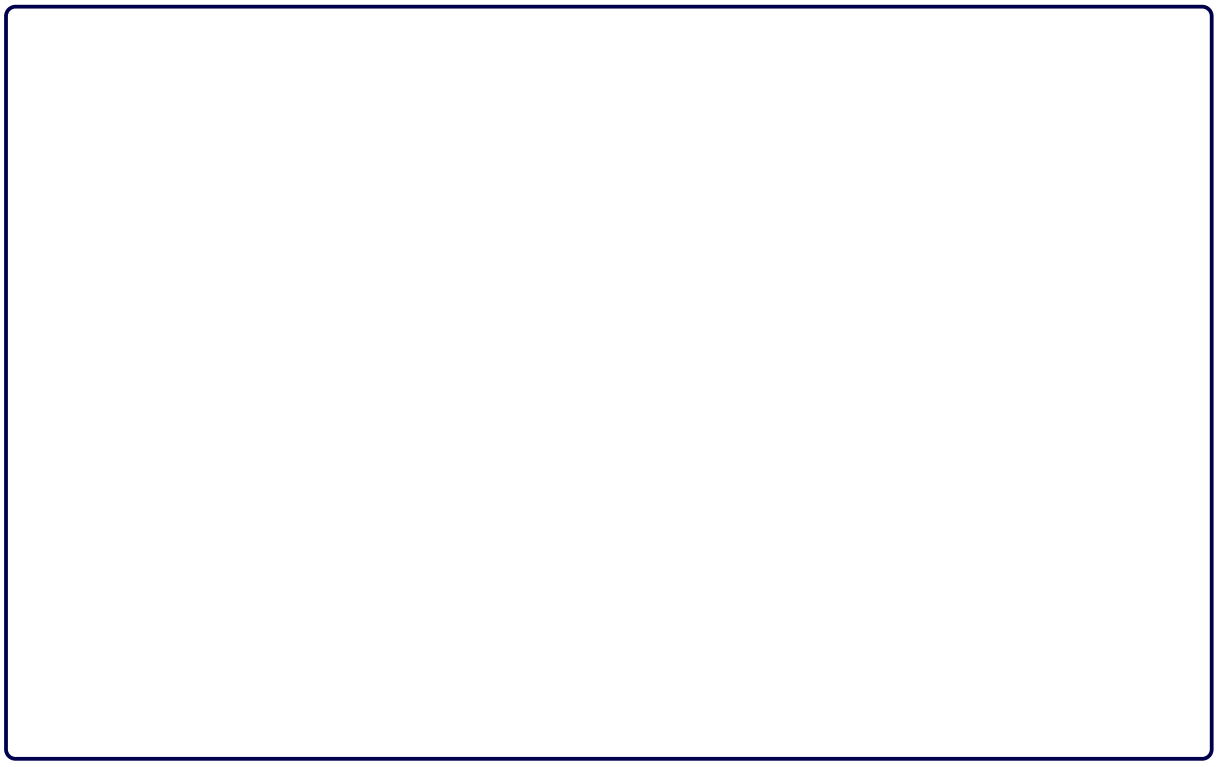
2. Consider a neural network with a single intermediate layer and a single output, where the the activation function for every vertex in the intermediate layer is the sigmoid function $\sigma(v) = \frac{1}{1+\exp\{-v\}}$ of a weighted linear combination of inputs, while the output is a weighted linear combination of the all the intermediate vertices, and all weights are non-negative. Show that this model has a likelihood convex with respect to all weight parameters.

   Hints:

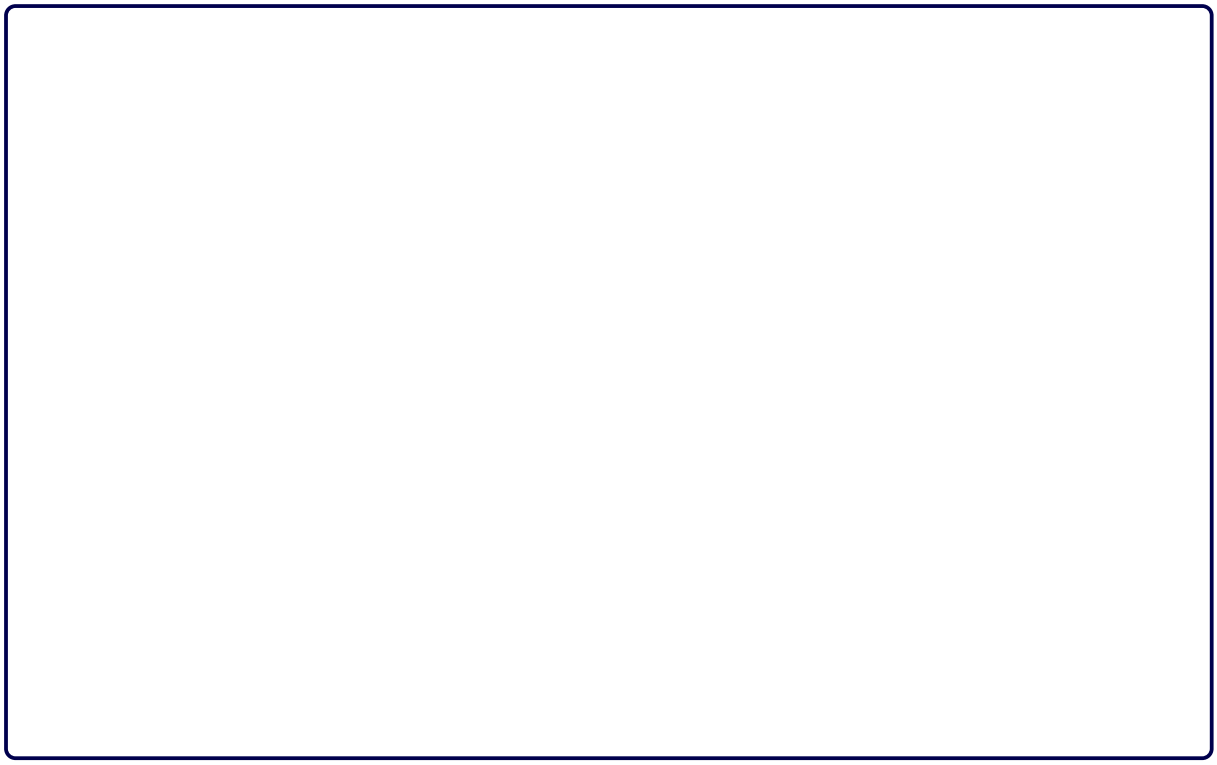
   - First show that the negative log likelihood for each intermediate layer vertex v (viewed as the outcome), with the weighted combination of inputs for that vertex (viewed as a single feature x) is a convex function. That is, show that:

     $$h(x) = -v \log \sigma(x) - (1 - v) \log(1 - \sigma(x))$$

     is convex in x. You can do this by showing that the second derivative of this function with respect to x is always non-negative.
   - Then, show that a composition of a linear function $g(w_1, \ldots, w_k)$ of several arguments and a convex function $h(x)$ that is convex in its single argument x is convex in each argument. That is, show that $h(g(w_1, \ldots, w_k))$ is convex in each $w_1, \ldots, w_k$.
   - Finally, show that a weighted combination of convex functions is also convex, if all weights are non-negative.

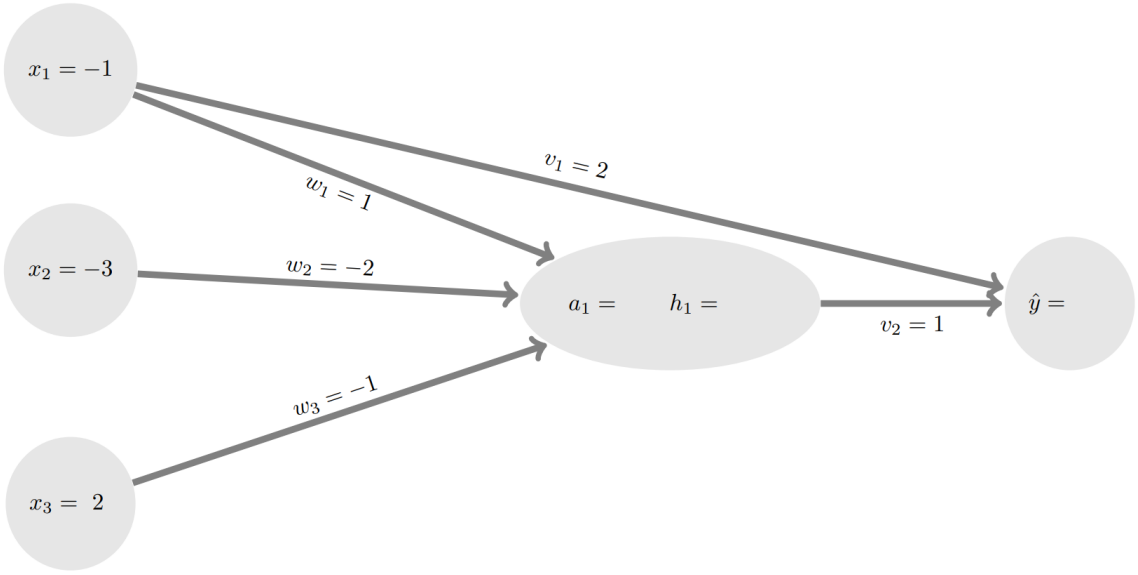


3. Assume there exist a setting of parameters for this model that minimizes the squared loss, such that at least two intermediate nodes have different weights feeding into their activation function. Show that there exists more than one setting of parameters that minimizes the squared loss. (This shows the parameters of this model are not identified).



4. This question contains multiple parts. Please answer all of these parts in large the answer box following the question. (Feel free to increase the box for additional space.)

   Consider the 2-layer neural network shown below. There are three input features x_1, x_2, and x_3 which get fed into one activation (a_1) from which a hidden value is computed (h_1). The non-linearities connecting activations to hidden values are **rectified linear units** $ReLU$: $h_1 = ReLU(a_1)$, with $ReLU(x) = 0$ if $x < 0$, and $ReLU(x) = x$ otherwise. The output unit takes 2 inputs: the hidden value h_1 and x_1.



   (i) Execute forward propagation on this network, writing the appropriate values for a_1, a_2, h_1, h_2 and $\hat{y}$ in the figure above.

   (ii) Give the expression of $\hat{y}$ as a function of x_1, x_2, x_3, w_1, w_2, w_3, v_1, v_2 and the $ReLU(\cdot)$ function.

   (iii) The correct class for example $x = [x_1, x_2, x_3] = [-1, -3, -2]$ is $y = -1$. Please run the backpropagation algorithm to minimize the squared error loss $l = \frac{1}{2}(y - \hat{y})^2$ on this single example. Derive the mathematical expression of the gradients of the loss l with respect to weights w_1, w_2, and w_3, and calculate its numerical value.

   (iv) Indicate how the value of each parameter below changes after the update: does it increase, decrease, or stay the same?

   (v) Derive the update rule for parameters v_1 and v_2 when running the backpropagation algorithm on the same example x, with the squared loss l and a step size $\eta = 12$. *Hint: you will need to (1) derive the appropriate gradient, (2) evaluate the gradient, (3) update your guess for the parameter β_{new} by subtracting the gradient times the step size from the old guess β_{old}.*

2 SVMs

1. Consider 2D classification problem with 3 classes with the following training sample:

   Class 1: (0,1), (0,2), (1,1)
   Class 2: (-2,0), (-1,0), (-1,-1)
   Class 3: (2,0), (3,0), (2,-1)

   (a) Explain how you would formulate an SVM model to solve this problem. (Hint: How many classifiers do you need?)

   (b) Find w, b, and corresponding support vectors for each classifier. Please draw the plot. You can take a screenshot or photo and use the `\includegraphics{}` directive.

   (c) Given a test sample, explain how to make a decision.