Discussion Session Problems 3

01/29/2026

1. The universal approximation theorem states that a feedforward neural network with a single hidden layer can approximate any function over a compact set, given enough neurons. If this is true, why is deep learning so successful in practice? Why do we still need depth instead of just using one very wide layer?

   Although a single hidden layer network can approximate any function, it may require an extremely large number of neurons and be very difficult to train. Deep neural networks learn hierarchical feature representations, where early layers detect simple patterns and later layers combine them into more complex features. This structured representation makes learning more efficient. A shallow network would have to learn all these complex features directly from the input, which is theoretically possible but much harder in practice.

2. In neural networks, what is the role of the activation function and why do we need it?

   The activation function introduces nonlinearity into the network. Without activation functions, every layer would be a linear transformation, and multiple layers would collapse into a single linear layer. This would prevent the network from learning complex nonlinear relationships in the data.

3. For a softmax applied to the values $(3, 4, 1, 7)$, which of the following could be a possible output?

   A) 0.0171, 0.0465, 0.0023, 0.9341
   B) 0.0011, 0.0085, 0.0003, 0.8362
   C) 0.0023, 0.0171, 0.0465, 0.9341
   D) 0.0211, 0.0785, 0.0103, 0.9362

   Answer: A)
   A softmax output must consist of positive values that sum to 1, and larger inputs should correspond to larger probabilities. Since 7 is the largest input, it should receive the highest probability. Option A satisfies both conditions.

4. Suppose you are solving three problems: linear regression, logistic regression, and a small neural network. Which one is most likely to benefit from a newly discovered extremely fast large-scale matrix multiplication algorithm? Why?

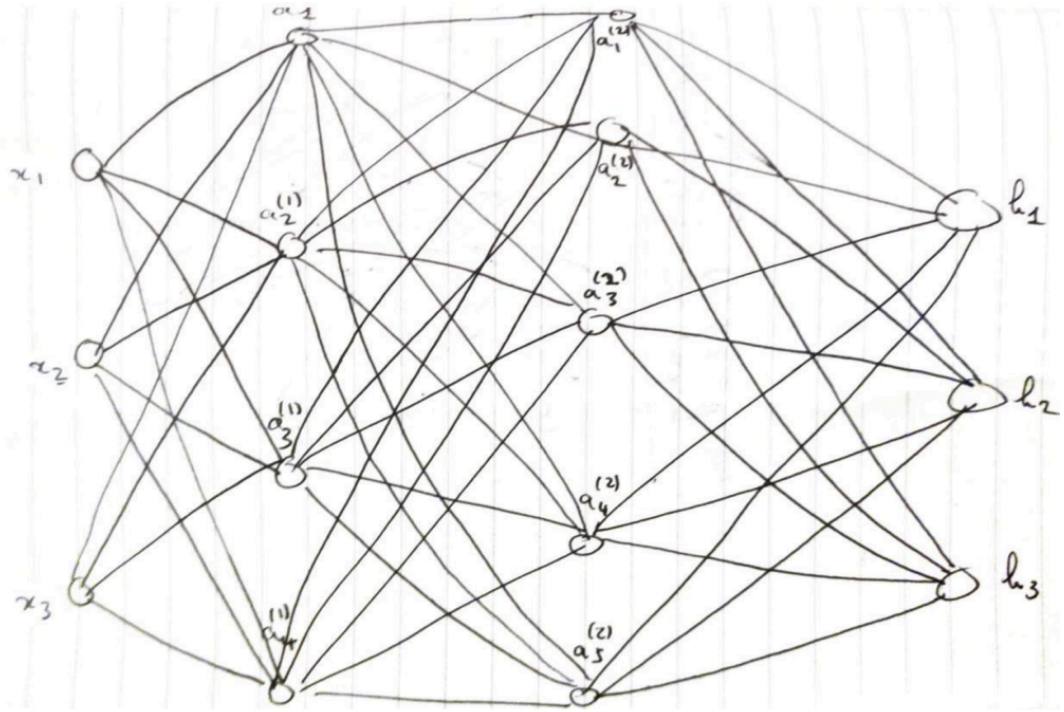
   Linear regression would benefit the most, especially when using the normal equation, which involves computing matrix products and a matrix inverse. Faster matrix multiplication would significantly speed up these operations.

5. Consider a dataset $D = \{x^{(1)}, \ldots, x^{(100)}\}$ where each $x^{(i)} \in \mathbb{R}^3$. This is a 3-class classification problem. We use a neural network with two hidden layers of sizes 4 and 5, using sigmoid activations in the hidden layers and a softmax output layer.

   a) How would you graphically represent this neural network?
   The network can be drawn as a layered graph:
   - Input layer: 3 nodes (one for each input feature)
   - Hidden layer 1: 4 nodes with sigmoid activation

- Hidden layer 2: 5 nodes with sigmoid activation
- Output layer: 3 nodes with softmax activation (one per class)

Edges connect every node in one layer to every node in the next layer.



b) What are the feedforward equations for a single training example $x \in \mathbb{R}^3$?

Let $g(\cdot)$ denote the sigmoid function.

$$z^{(1)} = W^{(1)}x + b^{(1)}, \quad W^{(1)} \in \mathbb{R}^{4\times 3},\ b^{(1)} \in \mathbb{R}^4$$
$$a^{(1)} = g\left(z^{(1)}\right)$$
$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)}, \quad W^{(2)} \in \mathbb{R}^{5\times 4},\ b^{(2)} \in \mathbb{R}^5$$
$$a^{(2)} = g\left(z^{(2)}\right)$$
$$z^{(3)} = W^{(3)}a^{(2)} + b^{(3)}, \quad W^{(3)} \in \mathbb{R}^{3\times 5},\ b^{(3)} \in \mathbb{R}^3$$
$$\hat{y} = \text{softmax}\left(z^{(3)}\right)$$

c) Describe the relationship between the graphical representation and the feedforward equations. What do the nodes and edges represent?

Each node represents one component of a vector, such as an input feature or a neuron activation. Each edge represents a weight that multiplies the value from one node and contributes to the next node. When multiple edges point into the same node, their weighted values are summed together and then passed through an activation function.

d) What is the total number of parameters in this neural network?

Layer 1: 4×3 weights + 4 biases = 16
Layer 2: 5×4 weights + 5 biases = 25
Output layer: 3×5 weights + 3 biases = 18

Total parameters = $16 + 25 + 18 = 59$

6. A wildlife monitoring system uses a Softmax regression model to classify images of animals into $K = 4$ classes:

$$\{\text{deer}, \text{fox}, \text{bear}, \text{rabbit}\}.$$

Each image is represented by a feature vector $x \in \mathbb{R}^n$. The model assigns a score

$$z_k = x^T W_k + b_k$$

to each class k.

(a) Explain why the scores $\{z_k\}_{k=1}^K$ cannot be directly interpreted as probabilities.

(b) Write down the Softmax probability assigned to class k.

(c) Suppose the true class label is ℓ. Using a one-hot encoding for the label, derive the Softmax cross-entropy cost for this single example and simplify it as much as possible.

(a) The raw scores z_k can be negative and do not necessarily sum to one, so they cannot be interpreted as probabilities.

(b) The Softmax probability for class k is

$$\hat{y}_k = P(Y = k \mid x, W, b) = \frac{e^{z_k}}{\sum_{i=1}^K e^{z_i}}.$$

(c) Let y be the one-hot encoded label vector with $y_\ell = 1$ and $y_k = 0$ for $k \neq \ell$. The Softmax cross-entropy cost for a single example is

$$J = -\sum_{k=1}^K y_k \log \hat{y}_k.$$

Since only the ℓ-th entry of y is nonzero (intuitively this means we only care about the predicted probability of the true class!), this simplifies to

$$J = -\log \hat{y}_\ell = -\log \left(\frac{e^{z_\ell}}{\sum_{j=1}^K e^{z_j}} \right).$$

Using log rules,

$$J = -z_\ell + \log \sum_{j=1}^K e^{z_j}.$$

7. Suppose we are training a Softmax regression classifier for handwritten digit recognition, where each image belongs to one of $K = 10$ digit classes. Consider a single training example (x, y), where y is one-hot encoded and the true class is ℓ.

(a) Starting from the simplified cost

$$J = -z_\ell + \log \sum_{j=1}^K e^{z_j},$$

compute the partial derivative $\frac{\partial J}{\partial z_k}$ for an arbitrary class k.

(b) Show that your result can be written in terms of the predicted probabilities $\hat{y}_k$ and the true labels y_k.

(c) Give an intuitive interpretation of the gradient when $k = \ell$ and when $k \neq \ell$.

(a) Taking the derivative with respect to z_k,

$$\frac{\partial J}{\partial z_k} = \frac{\partial}{\partial z_k}\left(-z_\ell + \log \sum_{j=1}^{K} e^{z_j}\right).$$

The derivative of $-z_\ell$ is -1 if $k = \ell$ and 0 otherwise, which we write as $-1_{\{k=\ell\}}$. The derivative of the second term is

$$\frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}.$$

Thus,

$$\frac{\partial J}{\partial z_k} = -1_{\{k=\ell\}} + \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}.$$

(b) Since $y_k = 1_{\{k=\ell\}}$ and $\hat{y}_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$, we obtain

$$\boxed{\frac{\partial J}{\partial z_k} = \hat{y}_k - y_k}.$$

(c) If $k = \ell$, the gradient is $\hat{y}_\ell - 1$, which is negative when the model underestimates the true class probability. If $k \neq \ell$, the gradient is $\hat{y}_k$, which encourages the model to decrease probability mass assigned to incorrect classes.

8. You are training a Softmax regression model on a dataset with m examples, n features, and K classes. Let:

- $X \in \mathbb{R}^{m \times n}$ be the data matrix,
- $W \in \mathbb{R}^{n \times K}$ be the weight matrix,
- $Y \in \mathbb{R}^{m \times K}$ be the matrix of one-hot labels,
- $\hat{Y} \in \mathbb{R}^{m \times K}$ be the matrix of predicted Softmax probabilities.

(a) Write the total Softmax cross-entropy cost over all m examples.

(b) Using your knowledge of the single-example gradient, derive the gradient of the cost with respect to $W_{k,j}$.

(c) Show how this expression can be written in fully vectorized form and verify that the dimensions are correct.

(a) The total cost is

$$J = -\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log \hat{y}_k^{(i)}.$$

(b) From the single-example result,

$$\frac{\partial J}{\partial W_{k,j}} = \sum_{i=1}^{m} (\hat{y}_k^{(i)} - y_k^{(i)}) x_j^{(i)}.$$

$W_{k,j}$ is the weight connecting feature j to class k. We need to find:

$$\frac{\partial J}{\partial W_{k,j}}$$

For example i, the score for class k is:

$$z_k^{(i)} = \sum_{j=1}^{n} W_{k,j} \cdot x_j^{(i)} + b_k = (W_k)^T x^{(i)} + b_k$$

where W_k is the k-th column of W^T (or equivalently, the k-th row of W).

$$\frac{\partial z_k^{(i)}}{\partial W_{k,j}} = x_j^{(i)}$$

Because $z_k^{(i)}$ is linear in $W_{k,j}$:

$$z_k^{(i)} = \cdots + W_{k,j} \cdot x_j^{(i)} + \cdots$$

Taking $\partial / \partial W_{k,j}$ gives us $x_j^{(i)}$.
From Question 7, we know:

$$\frac{\partial J^{(i)}}{\partial z_k^{(i)}} = \hat{y}_k^{(i)} - y_k^{(i)}$$

By the chain rule:

$$\frac{\partial J^{(i)}}{\partial W_{k,j}} = \frac{\partial J^{(i)}}{\partial z_k^{(i)}} \cdot \frac{\partial z_k^{(i)}}{\partial W_{k,j}} = (\hat{y}_k^{(i)} - y_k^{(i)}) \cdot x_j^{(i)}$$

The total cost is $J = \sum_{i=1}^{m} J^{(i)}$, so:

$$\frac{\partial J}{\partial W_{k,j}} = \sum_{i=1}^{m} (\hat{y}_k^{(i)} - y_k^{(i)}) \cdot x_j^{(i)}$$

(c) Writing this as a dot product,

$$\frac{\partial J}{\partial W_{k,j}} = X_j^T (\hat{Y}_k - Y_k).$$

Stacking these derivatives for all features j and classes k, we obtain the fully vectorized gradient

$$\boxed{\frac{\partial J}{\partial W} = X^T (\hat{Y} - Y)}.$$

Since X^T has shape (n, m) and $(\hat{Y} - Y)$ has shape (m, K), the result has shape (n, K), matching the shape of W.

9. Why is cross-entropy loss typically preferred over mean squared error when training a neural network for classification with a Softmax output layer?

Cross-entropy is better suited for classification because it directly measures the difference between probability distributions. When combined with Softmax, it leads to simple and stable gradients. Mean squared error can result in slower learning and poor gradients when probabilities saturate.

10. In deep neural networks, why can using the sigmoid activation function in many layers lead to slow or difficult training?

    Sigmoid activations can cause the vanishing gradient problem. When inputs are very positive or negative, the sigmoid output saturates and its gradient becomes very small. During backpropagation, gradients shrink as they pass through many layers, making it hard for earlier layers to learn.

11. What is the purpose of bias terms in a neural network layer? What would happen if we removed all bias terms?

    Bias terms allow each neuron to shift its activation function left or right, making the model more flexible. Without biases, every neuron's output would be forced to pass through the origin, which reduces the types of functions the network can represent.

12. Why is vectorization important when implementing neural networks and gradient computations?

    Vectorization allows us to compute operations for many examples and many neurons at once using efficient matrix operations. This greatly speeds up training, simplifies code, and takes advantage of optimized linear algebra libraries and hardware like GPUs.

13. A neural network with nonlinear activation functions can produce nonlinear decision boundaries. Explain why a network with only linear activations cannot do this, no matter how many layers it has.

    If all activations are linear, each layer performs a linear transformation. Composing multiple linear transformations results in another linear transformation. Therefore, the entire network is equivalent to a single linear model and can only produce linear decision boundaries.