24-788 Introduction to Deep Learning (Spring 2025)
Midterm Exam

Ryan Wu (Andrew ID: weihuanw)

Thursday February 27, 2025

1 Short Answer Questions

(a) Why do modern deep learning models no longer use the threshold function (like shown in Eq. 1) as an activation function?

$$f(x) = \begin{cases} 0, & \text{if } x < 0, \\ 1, & \text{if } x \geq 0. \end{cases} \tag{1}$$

Modern deep learning models no longer use the step function as the threshold functions because they are not differentiable. This lack of differentiability prevents gradients from being computed effectively during backpropagation.

(b) Name 2 reasons why ReLU (Rectified Linear Unit) is better than Sigmoid and Tanh (hyperbolic tangent) activation functions in deep neural networks.

1. ReLU addresses the vanishing gradient problem by providing a non-saturating gradient for positive inputs. When the input is positive, the derivative of ReLU is 1, which ensures that gradients remain large enough during backpropagation even in deep networks.

2. ReLU is more computationally efficient because it involves a simple thresholding operation defined as:

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

This operation only requires a basic comparison and does not involve computationally intensive exponential functions used in Sigmoid or Tanh (hyperbolic tangent) activation functions in deep neural networks.

(c) Assume you are training a fully connected neural network for classification and there is no bug in your code. After the loss function converges, although the prediction accuracy is high on the training set, the prediction accuracy on the validation set is low. List 2 things you could try to improve the performance, and explain why they should work.

1. We can apply regularization techniques such as dropout or weight decay. These methods help prevent the model from overfitting by reducing its reliance on specific neurons or weight values. By randomly deactivating neurons during training (dropout) or penalizing large weights (weight decay), the network is can learn more robust and generalized features.

2. We can apply data augmentation (e.g., rotations, flips, scaling, noise) on the training set to enhance the training data. It generates additional training examples, at the same time, increases the diversity of the training data, helping the model to generalize better to unseen examples and improve validation accuracy.

(d) Assume you are training a fully connected neural network for classification and there is no bug in your code. This time, the loss function converges after only a few epochs but the prediction accuracy on both the training set and validation set is low. List 2 things you could try to improve the performance, and explain why they should work.

1. We can increase the complexity of the network by adding more layers or neurons to the network. Increasing the model's capacity allows it to capture more complex patterns in the data, which can help improve both training and validation accuracy.

2. We can adjust the network's learning rate to improve both training and validation accuracy. Fine-tuning the learning rate enables the optimizer to explore the parameter space more efficiently, at the same time, prevents premature convergence to a suboptimal solutions.

(e) What are the two central features of the Adam optimizer that make it a successful optimizer for training neural networks?

1. An adaptive and dynamic learning rate for each parameter by leveraging the first moment (mean) and second moment (variance) of the gradients. It enables the optimizer to adjust the learning rate for each parameter individually, making it more efficient for training neural networks.

2. Bias correction for the moment estimates. It compensates for the initialization bias, leading to more accurate estimates and improving the stability and convergence of the optimizer for training neural networks.

(f) Name 2 advantages of CNNs (Convolutional Neural Networks) compared to fully connected neural networks.

1. CNNs are more parameter-efficient compared to fully connected neural networks. By employing convolutional layers with weight sharing, the same set of filters is applied across the entire image, reducing the number of parameters and mitigating overfitting.

2. CNNs exhibit translation invariance, which increases data stability. It allows CNNs to recognize features regardless of their spatial location in the image, leading to more robust and stable performance across various input transformations.

(g) Explain the difference between Max pooling and Average pooling. When would you prefer one over the other while working with an image-based dataset?

Max pooling operation selects the maximum pixel value from each pooling window, capturing the most prominent features in that region. On the other hand, average pooling operation computes the average pixel value from the pooling window, which results in a smoother and more generalized representation of the feature map.

When working with an image-based dataset, max pooling is preferred when object recognition or edge detection is required because it preserves the most prominent features in each pooling window. Average pooling is preferred when a smoother representation is desired or when capturing the overall texture of an image is important.

2 Calculation

(a) Your friend wants to build a fully-connected one-hidden-layer Neural Network. Her inputs have 99 features. She wants 7 hidden neurons in the first layer and 2 neurons in the output layer. The output is passed through a softmax function. Each layer is a linear layer which is fully-connected to the previous layer and includes bias terms. What is the number of parameters she will need to create this neural network? Each scalar counts as one parameter.

Given:
Build a fully-connected one-hidden-layer Neural Network with:

- Number of Input Neurons: 99
- Number of Hidden Neurons: 7
- Number of Output Neurons: 2

Formula for the Total Number of Parameters:

$$\begin{aligned}\text{Parameters} = &\underbrace{(\text{Number of Input Neurons}) \times (\text{Number of Hidden Neurons})}_{\text{Input-Hidden Weights}} + \underbrace{(\text{Number of Hidden Neurons})}_{\text{Hidden Biases}} \\ &+ \underbrace{(\text{Number of Hidden Neurons}) \times (\text{Number of Output Neurons})}_{\text{Hidden-Output Weights}} + \underbrace{(\text{Number of Output Neurons})}_{\text{Output Biases}}\end{aligned}$$

Solution:

$$\begin{aligned}\text{Parameters} &= 99 \times 7 + 7 + 7 \times 2 + 2 \\ &= 693 + 7 + 14 + 2 \\ &= 716.\end{aligned}$$

Thus, a total of **716 parameters** is required to build the network.

(b) A 7 × 7 feature map is input into a layer of a CNN model which has a 3 × 3 filter with padding size 2 and stride size 2. What is the size of the new feature map output from this convolutional layer?

If the same 7×7 feature map is passed into a CNN layer which has a 5 ×5 filter with stride 1, what padding size does the layer need to use to make the size of the outputted feature map the same size as the original input (7 × 7)?

Given:
(Part 1) A 7×7 feature map is input into a layer of a CNN model with:

- **Filter Size:** 3×3
- **Padding Size:** 2
- **Stride Size:** 2

(Part 2) A 7×7 feature map is input into a layer of a CNN model with:

- **Filter Size:** 5×5
- **Stride Size:** 1

Formulas for the Output Feature Map Size & Padding:

$$W_{\text{out}} = \frac{W - F + 2P}{S} + 1.$$

$$P = \frac{S(W_{\text{out}} - 1) - W + F}{2}.$$

- W: The width (or height) of the input feature map.
- F: The size (width or height) of the filter/kernel.
- P: The padding applied to the input feature map.
- S: The stride with which the filter moves across the input.
- W_{out}: The width (or height) of the output feature map.

Solution:

Part 1:

$$W_{\text{out}} = \frac{7 - 3 + 2 \times 2}{2} + 1 = 5.$$

Thus, the output feature map size is **5×5**.

Part 2:

$$P = \frac{1(7 - 1) - 7 + 5}{2} = 2.$$

Thus, the required padding size is **2**.

(c) Suppose in one layer of a CNN, you get the feature map given in Fig. 1. and want to conduct convolution using the filter given in Fig. 1., with stride 1 and padding 0. Calculate the new feature map after the convolution operation.

Given:
One layer of a CNN with:

$$I = \begin{bmatrix} 1 & 2 & -1 \\ 0 & 3 & 5 \\ 2 & 1 & 4 \\ 3 & 0 & -2 \end{bmatrix}$$

convolve it (stride $S = 1$, padding $P = 0$) with the following 2×2 filter:

$$K = \begin{bmatrix} -0.5 & 1 \\ 1.5 & 0 \end{bmatrix}.$$

Formulas for Convolution (Stride = 1, Padding = 0):

$$O(i,j) = \sum_{u=1}^{F} \sum_{v=1}^{F} I\big(i+u-1,\ j+v-1\big) \cdot K(u,v),$$

where $F = 2$ is the filter size. The output feature map O has dimensions:

$$\big(4-2+1\big) \times \big(3-2+1\big) \ = \ 3 \times 2.$$

Solution:
Compute each element $O(i,j)$ by overlaying the 2×2 filter on the corresponding 2×2 region of I:

Output Element	**Calculation**	**Result**
$O(1,1)$	$1 \cdot (-0.5) + 2 \cdot 1 + 0 \cdot 1.5 + 3 \cdot 0$	1.5
$O(1,2)$	$2 \cdot (-0.5) + (-1) \cdot 1 + 3 \cdot 1.5 + 5 \cdot 0$	2.5
$O(2,1)$	$0 \cdot (-0.5) + 3 \cdot 1 + 2 \cdot 1.5 + 1 \cdot 0$	6.0
$O(2,2)$	$3 \cdot (-0.5) + 5 \cdot 1 + 1 \cdot 1.5 + 4 \cdot 0$	5.0
$O(3,1)$	$2 \cdot (-0.5) + 1 \cdot 1 + 3 \cdot 1.5 + 0 \cdot 0$	4.5
$O(3,2)$	$1 \cdot (-0.5) + 4 \cdot 1 + 0 \cdot 1.5 + (-2) \cdot 0$	3.5

Thus, the output feature map O is a 3×2 matrix:

$$\boldsymbol{O} = \begin{bmatrix} 1.5 & 2.5 \\ 6.0 & 5.0 \\ 4.5 & 3.5 \end{bmatrix}.$$

(d) Fig. 2 shows the architecture of LeNet, where in each convolutional layer, C@W × W represents the size of the output feature maps (C for number of channels and W for height and width). Calculate the number of parameters in this CNN model. (hint: don't forget biases in both convolutional filters and fully connect layers, also you should only consider trainable parameters)

Given:

- input: $1 \times 32 \times 32$
- conv1: $f(6 \times 5 \times 5)$@stride $= 1 \ \rightarrow \ 6 \times 28 \times 28$
- s2: pooling$(2 \times 2) \ \rightarrow \ 6 \times 14 \times 14$
- conv3: $f(16 \times 5 \times 5)$@stride $= 1 \ \rightarrow \ 16 \times 10 \times 10$
- s4: pooling$(2 \times 2) \ \rightarrow \ 16 \times 5 \times 5$
- conv5: $f(120 \times 5 \times 5)$@stride $= 1 \ \rightarrow \ 120 \times 1 \times 1$
- fc6: $120 \ \rightarrow \ 84$
- fc7: $84 \ \rightarrow \ 10$

Formulas:

- **Convolutional layer with input depth** D_{in}, filter size $K \times K$, and F filters:

$$\text{Weights} = F \times (D_{\text{in}} \times K \times K), \quad \text{Biases} = F.$$

- **Fully connected layer from** N_{in} **to** N_{out}:

$$\text{Weights} = N_{\text{in}} \times N_{\text{out}}, \quad \text{Biases} = N_{\text{out}}.$$

- **Total Parameters:**

$$\text{Parameters} = \text{Weights} + \text{Biases}.$$

Solution:
Calculate the number of parameters (weights + biases) in each layer. Pooling layers have no trainable parameters, so only consider the convolutional and fully connected layers.

Layer	**Weights**	**Biases**	**Total**
conv1	$6 \times (1 \times 5 \times 5) = 150$	6	$150 + 6 = 156$
conv3	$16 \times (6 \times 5 \times 5) = 2400$	16	$2400 + 16 = 2416$
conv5	$120 \times (16 \times 5 \times 5) = 48\,000$	120	$48\,000 + 120 = 48\,120$
fc6	$120 \times 84 = 10\,080$	84	$10\,080 + 84 = 10\,164$
fc7	$84 \times 10 = 840$	10	$840 + 10 = 850$
Total			**61 706**

Thus, the total number of trainable parameters in this LeNet model is **61,706**.

(e) Let $f(\cdot)$ be the objective function you would like to minimize. At time step t, you get $x_t = 3$, $f(x_t) = 6$ and $f'(x_t) = 1$. After one iteration of gradient descent with learning rate $\eta = 0.1$, what is the value of x_{t+1}. Now assume you get $f'(x_{t+1}) = 0$, does it guarantee x_{t+1} is the global optimum? If yes, explain why. If not, give a solution that will help find the global optimum.

Given:
Let $f(\cdot)$ be the objective function you would like to minimize. At time step t:

- $x_t = 3$,
- $f(x_t) = 6$,
- $f'(x_t) = 1$,
- Learning rate $\eta = 0.1$.

Formula for x_{t+1}:

$$x_{t+1} = x_t - \eta\, f'(x_t).$$

Solution:

Part 1:

$$x_{t+1} = 3 - 0.1 \times 1 = 3 - 0.1 = 2.9.$$

Part 2:
$x_{t+1} = 2.9$, $f'(x_{t+1}) = 0$ indicates that 2.9 is a critical point of f. However, a zero derivative does not guarantee that this point is the global optimum:

- If f is convex, any stationary point is the global minimum, $x_{t+1} = 2.9$ would be the global optimum.
- If f is non-convex, $x_{t+1} = 2.9$ might be a local minimum, local maximum, or a saddle point.

For the non-convex case, to increase the likelihood of finding the global optimum:

- Run gradient descent from multiple different initial points and compare the outcomes.
- Use more advanced optimization algorithms (SDG with momentum, Adam, etc.) that can escape local minima.

3 Multiple Choice Questions

(a) In CNN training, when backpropagating through a max pooling layer, the gradient at the input locations of the non-maximum values is:

A) Identical to the derivative at the location of the maximum value

B) Identical to the input value at the location of the maximum value

C) Identical to the input value at the location of the non maximum value

D) 0

Answer: D

(b) In CNN training, when backpropagating through a mean pooling layer, the gradient at the output of a mean pool filter is:

A) equally distributed over the input pool

B) assigned to the input location of the maximum value

C) 0

D) distributed over the inputs in proportion to their values

Answer: A

(c) Consider applying a 5x5x5 convolution filter with a stride of 1 and padding of 2 on an RGB image input of dimension (3,H,W). Which of the following statements are true?

A) The output is of dimension (5,H,W)

B) The output is of dimension (3,H,W)

C) The output is of dimension (5,5,5)

D) The convolution is not valid

Answer: A

(d) When using gradient descent to find the minimum of a function, which of the following is correct about gradient descent:

A) Accurately solves for the location that has minimum gradient.

B) Uses the Hessian of the function to solve for the location of the minimum.

C) Starts at some initial location and iteratively moves in the opposite direction of the gradient, until it finds the minimum.

D) Starts at some initial location and iteratively moves in the direction of the gradient, until it finds the minimum.

Answer: C

(e) Select all the statements that are correct:

A) The output of a neuron has a probabilistic interpretation when using sigmoid activations.

B) A multi-layer fully-connected neural network with linear activation functions is equivalent to a single-layer fully-connected neural network with linear activation.

C) For a multilayer fully connected neural network with Tanh as an activation function at each layer, if you scale up all the weights (including biases) by 2, the output of the model remains exactly the same.

D) Threshold activations cannot be used for backpropagation.

Answer: A, B, D

(f) What are the benefits of using L2 regularization of weights in a network?

A) It ensures that neuron activations are sparse (i.e. only a few are non-zero), improving the interpretability of their output.

B) It prevents activation functions from becoming too steep, which helps keep the network output from changing too steeply with respect to changes in the input.

C) It prevents overfitting to training data.

D) It guarantees to restrict the weights to lie within a unit hypersphere, preventing floating point overflow.

Answer: C

(g) What are the improvements of the momentum learning rule compared to vanilla gradient descent?

A) The magnitude of the changes to the parameters can increase without bound if the gradients don't change very much across iterations.

B) Momentum learning allows us to forget about the learning rate since the learning rule will automatically adjust the step size.

C) Momentum learning guarantees the network will find global minima as opposed to local minima.

D) Momentum learning smooths noisy gradients, reduces oscillations, and encourages updates in directions with smooth convergence behaviors.

Answer: D

(h) Which of the following guidelines is applicable to the initialization of the weight vector in a fully connected neural network (assume ReLU activation)?

A) Should not set it to zero since it will cause overfitting.

B) Should not set it to zero since it will cause all neurons to have the same gradient.

C) Should set it to zero to avoid introducing an unlearned bias into the network.

D) Should set it to zero in order to preserve symmetry across all neurons.

Answer: B

(i) Which of the following statements are true regarding Multilayer perceptrons:

A) If there is an upper limit on the width of the layers but no limit on the depth of the network (number of layers), the MLP can model/approximate any function.

B) A one-hidden layer network with infinite neurons in the hidden layer can compose an exact model of any real-valued function.

C) A one-hidden layer network with infinite neurons in the hidden layer can compose an approximate model of any real-valued function.

D) For a deep neural network, if all the weight parameters are initialized with an appropriate initialization method, biases can be initialized with zeros or constants.

Answer: A, C, D

(j) Select all the true statements about a convolution layer:

A) The number of "channels" in any filter equals the number of input maps.

B) The number of "channels" in any filter equals the number of output maps. (Affine maps output by the layer)

C) The number of filters equals the number of input maps.

D) The number of filters equals the number of output maps.

Answer: A, D