CS230: Deep Learning

Fall Quarter 2021

Stanford University

Midterm Examination

Suggested duration: 180 minutes

	Problem	Full Points	Your Score
1	Multiple Choice	16	
2	Short Answers	16	
3	Weight Initialization	20	
4	Backpropagation	14	
5	CNNs	21	
6	Normalization Techniques	15	
Total		102	

The exam contains 14 pages including this cover page.

- This exam is open book, but collaboration with anyone else, either in person or online, is strictly forbidden pursuant to The Stanford Honor Code.
- In all cases, and especially if you're stuck or unsure of your answers, **explain your work, including showing your calculations and derivations!** We'll give partial credit for good explanations of what you were trying to do.

Name: _____

SUNETID: _____@stanford.edu

The Stanford University Honor Code:
I attest that I have not given or received aid in this examination, and that I have done my share and taken an active part in seeing to it that others as well as myself uphold the spirit and letter of the Honor Code.

Signature: _____

Question 1 (Multiple Choice Questions, 16 points)

For each of the following questions, circle the letter of your choice. Each question has AT LEAST one correct option unless explicitly mentioned. No explanation is required.

(a) **(2 points)** Given the two layer network $n(x) = W_2 f(W_1 x + b_1) + b_2$ with $W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$, which of the following statements about $f(x)$ are **false** (incorrect)?

(i) Choosing $f(x)$ to be an additional layer $f(x) = W_f x + b_f$, where $W_f \in \mathbb{R}^{H \times H}$ will likely decrease the training accuracy compared to $f(x) = x$

(ii) Using $f(x) = \text{ReLU}(-abs(x))$ to train and test our network will lead to a much lower accuracy than $f(x) = \text{ReLU}(x)$

(iii) Choosing $f(x)$ to be an additional layer $f(x) = W_f x + b_f$, where $W_f \in \mathbb{R}^{H \times H}$, will decrease the model's tendency to overfit on the training data compared to $f(x) = \text{ReLU}(x)$

(iv) None of the above

Correct answer: (i)

(b) **(2 points)** Which statements about regularization techniques are correct?

(i) Batch normalization can help with reducing overfitting

(ii) L_2 regularization encourages the weights to be smaller in magnitude compared to no regularization

(iii) Dropout with keep probability 0.5 usually causes the model to converge slower compared to not using dropout

(iv) None of the above

Correct answers: (i), (ii), and (iii)

(c) **(2 points)** Which of the following are activation functions you could use with SGD in a neural network (that is, which functions could be effective when training a neural net with SGD in practice)?

(i) $f(x) = \min(0, x)$

(ii) $f(x) = \max(-x, -2x)$

(iii) $f(x) = \text{ceiling}(x)$, where $\text{ceiling}(x)$ is a function that rounds x up to the nearest integer

(iv) None of the above

Correct answers: (i) and (ii)

(d) **(2 points)** Which of the following are hyperparameters?

(i) The bias b of the last hidden layer of a neural network

(ii) The weight W of the last hidden layer of a fully connected neural network

(iii) The regularization strength used in L_1 regularization

(iv) The regularization strength used in L_2 regularization

Correct answers: (iii) and (iv)

(e) **(2 points)** Given that we are using gradient descent, which of the following loss functions converge ($\mathcal{L} \to 0$) for the given initialization (x_0) and step-size ($step$)?

(i) $\mathcal{L} = |x|$ for $x_0 = 0.1$ and $step = 0.5$

(ii) $\mathcal{L} = x^2$ for $x_0 = 1$ and $step = 0.99$

(iii) $\mathcal{L} = |x|$ for $x_0 = 1$ and $step = 1.5$

(iv) $\mathcal{L} = x^2$ for $x_0 = 1$ and $step = 1$

Correct answers: (ii)

(f) **(2 points)** Which of the following prevent overfitting?

(i) L_2 regularization

(ii) Dropout

(iii) Data Augmentation

(iv) Batch normalization

Correct answers: (i), (ii), (iii) and (iv)

(g) **(2 points)** A single $(15 \times 15 \times 3)$ image is passed through a convolutional layer with 15 filters, each of size $(3 \times 3 \times 3)$. The padding size is 1 and the stride size is also 1. What is the size (that is, what are the dimensions) of the output image?

(i) $(15 \times 15 \times 3)$

(ii) $(12 \times 12 \times 6)$

(iii) $(15 \times 15 \times 15)$

(iv) None of the above

Correct answers: (iii)

(h) **(2 points)** Which of the following are true about Generative Adversarial Networks (GANs)?

(i) The Generator can learn faster by maximizing the probability that an image is real as classified by the Discriminator rather than minimizing the probability that an image is fake as classified by the Discriminator

(ii) The Discriminator aims to learn the distribution of input images but the Generator does not

(iii) After training the GAN, the Discriminator loss eventually reaches a constant value

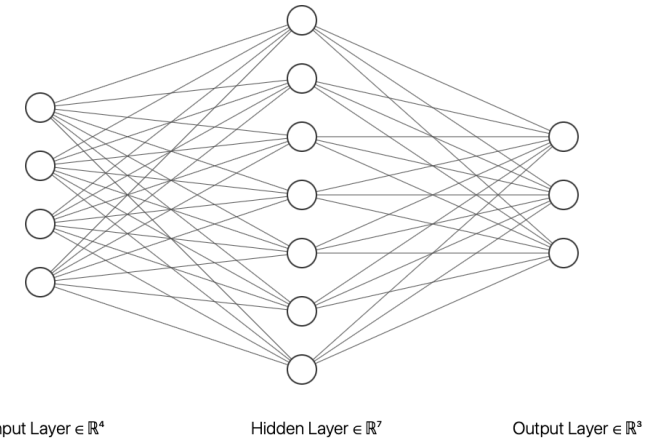
(iv) None of the above

Correct answers: (i) and (iii)

Question 2 (Short Answers, 16 points)

The questions in this section can be answered in 2-4 sentences. Please be concise in your responses.

(a) **(4 points)** A 2-layer neural network that has 7 neurons in the first layer and 3 neurons in the output layer takes in a 4-dimensional input vector. How many parameters (weights and biases) are in the network?



Solution: 59

(b) **(4 points)** Why do we need activation functions in neural networks (e.g. ReLU)?

Solution: A chief purpose of activation functions is introducing non-linearities. Without a non-linear activation function in the network, a neural network, no matter the amount of layers, would behave just like a single-layer perceptron.

(c) **(4 points)** Your friend is researching a new activation function: $h(x) = x\log(1 + \tanh(e^x))$. He'd like your help in calculating it's gradient. Please calculate $\frac{dh}{dx}$. Recall that the gradient of $\tanh(z)$ is $1 - \tanh^2(z)$. Hint: use the product rule and the chain rule.

Solution: Let $y = \log(1 + \tanh(e^x))$.

$$\frac{dy}{dx} = \frac{e^x(1 - \tanh^2(e^x))}{1 + \tanh(e^x)}$$

$$h(x) = xy$$

$$\frac{dh}{dx} = x\frac{dy}{dx} + y$$

$$\frac{dh}{dx} = \frac{xe^x(1 - \tanh^2(e^x))}{1 + \tanh(e^x)} + \log(1 + \tanh(e^x))$$

(d) **(4 points)** Consider a multi-class classification network (one-hot targets). Describe the activation function you want to use at the output layer and what it does.

Solution: Softmax. Softmax gives the probability of each class where the probability sum of all classes is equal to 1.

$$
\begin{aligned}
Var(a_i^{[\ell-1]}) &= Var(a_i^{[\ell]}) \\
&= Var(z_i^{[\ell]}) \qquad \longleftarrow \text{linearity of } \textbf{tanh} \text{ around zero } tanh(z) \approx z \\
&= Var(\sum_{j=1}^{n^{[\ell-1]}} w_{ij}^{[\ell]} a_j^{[\ell-1]}) \\
&= \sum_{j=1}^{n^{[\ell-1]}} Var(w_{ij}^{[\ell]} a_j^{[\ell-1]}) \qquad \longleftarrow \text{variance of independent sum } Var(X+Y) = Var(X) + Var(Y) \\
&= \sum_{j=1}^{n^{[\ell-1]}} E[w_{ij}^{[\ell]}]^2 Var(a_j^{[\ell-1]}) + \\
&\qquad E[a_j^{[\ell-1]}]^2 Var(w_{ij}^{[\ell]}) + \qquad \longleftarrow \text{variance of independent product } Var(XY) = E[X]^2 Var(Y) + E[Y]^2 Var(X) + Var(X)Var(Y) \\
&\qquad Var(w_{ij}^{[\ell]}) Var(a_j^{[\ell-1]}) \\
&= n^{[\ell-1]} Var(w_{ij}^{[\ell]}) Var(a_j^{[\ell-1]}) \implies Var(W) = \frac{1}{n^{[\ell-1]}}
\end{aligned}
$$

Figure 1: Derivation of Xavier Initialization Variance

Question 3 (Weight Initialization, 20 points)

To prevent gradients from exploding or disappearing during backpropagation, we can use initialization functions that set the variance of the outputs of all layers in our network to be equal to one another.

One such initialization method is **Xavier Initialization**. In Xavier Initialization, we initialize all weights randomly using a Gaussian (Normal) distribution with zero mean and $Var(W^{[l]}) = \frac{1}{n^{[l-1]}}$ for layer l. The proof of this can be found in Section 4, shown in Figure 1.

In this question, we will try to further understand the motivation for using an initialization function. For all parts (unless explicitly specified), assume that we are working with a network in which **all** layers use the same activation function.

Xavier initialization is often used in cases where the activation function is $\tanh(x)$ or $\sigma(x)$. We assume in the derivation for Xavier initialization that the activation is $\tanh(x)$. Consider the implications of using Xavier initialization for a network in which all layers use $\sigma(x)$ as the activation function.

(a) **(4 points)** Define $\sigma(x)$ as a function of $\tanh(x)$.

Solution:

$$\sigma(x) = \frac{1}{2}\tanh(\frac{x}{2}) + \frac{1}{2}$$

(b) **(4 points)** For values close to 0, what is the approximate value of $\sigma(x)$ as a function of x?

Solution:

$$\sigma(x) \approx \frac{1}{4}x + \frac{1}{2}$$

(c) **(4 points)** We assume $Var(\tanh(x)) = Var(x)$ in the derivation for Xavier initialization above. Find the equation for $Var(\sigma(x))$ in terms of $Var(x)$.

Solution:

$$Var(\sigma(x)) = \frac{1}{16}Var(x)$$

(d) **(4 points)** Using the equation for $Var(\sigma(x))$ from the previous part, what is the variance we should use to initialize weights? Write your answer as an equation for $Var(W^{[l]})$.

Solution:

$$Var(W^{[l]}) = \frac{16}{n^{[l-1]}}$$

(e) **(4 points)** Assume we have a network with all sigmoid activations. However, we still choose to use $Var(W^{[l]} = \frac{1}{n^{[l-1]}}$. Consider the implications for the variance of outputs as we get deeper and deeper into the network. Will we run into an exploding gradients problem or a vanishing gradients problem or neither? Why?

Solution: Neither. Having a lower variance means inputs to the activation function are closer to the mean (near 0) where the slope (0.25) is not too high or too low to cause exploding/vanishing gradients.

Partial credit given for "vanishing gradients" with the assumption that the slope (0.25) around the mean was small enough to cause vanishing gradients in very deep networks.

Question 4 (Backpropagation, 14 points)

Consider the following neural network for K-class classification using softmax activation and cross-entropy loss, as defined below:

$$\begin{aligned}
\mathbf{z}^{[1]} &= W^{[1]}\mathbf{x} + \mathbf{b}^{[1]} \\
\mathbf{a}^{[1]} &= \text{Softplus}(\mathbf{z}^{[1]}) \\
\mathbf{z}^{[2]} &= W^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]} \\
\hat{\mathbf{y}} &= \text{LogSoftmax}(\mathbf{z}^{[2]}) \\
L &= -\sum_{i=1}^{K} \mathbf{y}_i \hat{\mathbf{y}}_i
\end{aligned}$$

where the model is given input $\mathbf{x}$ of shape $D_x \times 1$, and one-hot encoded label $\mathbf{y} \in \{0,1\}^K$. Assume that the hidden layer has D_a nodes, i.e. $\mathbf{z}^{[1]}$ is a vector of size $D_a \times 1$. Recall the softmax function is computed as follows:

$$\hat{\mathbf{y}} = [\log(\frac{\exp(\mathbf{z}_1^{[2]})}{Z}), \ldots, \log(\frac{\exp(\mathbf{z}_K^{[2]})}{Z})]$$

where $Z = \sum_{j=1}^{K} \exp(\mathbf{z}_j^{[2]})$
SoftPlus is a smooth approximation to the ReLU function and can be used to constrain the output to always be positive. It is an element-wise function defined as follows:

$$\text{Softplus}(x) = \frac{1}{\beta}\log(1 + \exp(\beta x))$$

(i) **(1 points)** What are the shapes of $W^{[2]}, b^{[2]}$?

Solution: $W^{[2]} \in \mathcal{R}^{K \times D_a}, b^{[2]} \in \mathcal{R}^{K \times 1}$.

(ii) **(2 points)** What is $\partial \hat{\mathbf{y}}_k / \partial z_k^{[2]}$?

Solution:

$$\frac{\partial \hat{\mathbf{y}}_k}{\partial z_k^{[2]}} = \frac{Z}{\exp(z_k^{[2]})} \frac{\exp(z_k^{[2]})Z - \exp(z_k^{[2]})^2}{Z^2} = 1 - \frac{\exp(z_k^{[2]})}{Z}$$

(iii) **(2 points)** What is $\partial \hat{\mathbf{y}}_k / \partial z_i^{[2]}$, for $i \neq k$?

Solution:

$$\frac{\partial \hat{\mathbf{y}}_k}{\partial z_i^{[2]}} = \frac{Z}{\exp(z_k^{[2]})}(-\frac{\exp(z_k^{[2]})\exp(z_i^{[2]})}{Z^2}) = -\frac{\exp(z_i^{[2]})}{Z}$$

(iv) **(1 points)** What is $\partial z^{[2]}/\partial a^{[1]}$? Refer to this result as δ_1.

Solution: $\delta_1 = W^{[2]}$

(v) **(2 points)** What is $\partial a^{[1]}/\partial z^{[1]}$? Refer to this result as δ_2.

Solution:

$$\delta_2 = \frac{1}{\beta}\frac{1}{1+e^{\beta \mathbf{z}^{[1]}}}e^{\beta \mathbf{z}^{[1]}}\beta = \frac{e^{\beta \mathbf{z}^{[1]}}}{1+e^{\beta \mathbf{z}^{[1]}}}$$

(vi) **(6 points)** Denote $\partial L/\partial z^{[2]}$ with δ_0. What is $\partial L/\partial W^{[1]}$ and $\partial L/\partial b^{[1]}$? You can reuse notations from previous parts. Hint: Be careful with the shapes.

Solution: Using chain rule for partial derivatives, we get:
(Note the element-wise product)

$$\partial L/\partial W^{[1]} = \delta_1^T * \delta_0 \circ \delta_2 * \mathbf{x}^T$$

$$\partial L/\partial b^{[1]} = \delta_1^T * \delta_0 \circ \delta_2$$

Question 5 (CNNs, 21 points)

Consider the hypothetical convolution neural network defined by the layers in the left column below. Fill in the shape of the output volume and the number of parameters at each layer. You can write the activation shapes in the format (H, W, C), where H, W, C are the *height*, *width* and *channel* dimensions, respectively. Unless specified, assume *padding 1*, *stride 1* where appropriate.

Notation:

- CONVx-N denotes a convolution layer with N filters with height and width equal to x.
- POOL-n denotes a n×n max-pooling layer with stride of n and 0 padding.
- FLATTEN flattens its inputs, identical to torch.nn.flatten / tf.layers.flatten
- FC-N denotes a fully-connected layer with N neurons

(1 point each)

Layer	Activation Volume Dimensions	Number of Parameters
Input	128*128*6	0
CONV5-8	126*126*8	8*(5*5*6 + 1) = 1208
ReLU	126*126*8	0
CONV3-16	126*126*16	16*(3*3*8 + 1) = 1168
POOL-2	63*63*16	0
CONV2-64	64*64*64	64*(2*2*16 + 1) = 4160
CONV1-32 with padding 0	64*64*32	32*(1*1*64 + 1) = 2080
POOL-2	32*32*32	0
POOL-2	16*16*32	0
FLATTEN	8192	0
FC-27	27	(8192 + 1)*27 = 221211

Question 6 (Normalization Techniques, 15 points)

Batch normalization is a powerful technique for training deep neural networks. Recall its formula:

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$
$$y = \gamma \hat{x} + \beta$$

where $x \in \mathbb{R}^d$ is a single data point. The batch size is B.

i. **(4 points)** Why is batch normalization superior to technique that makes each neuron's input 0 mean and unit standard deviation? What about pre-processing all **samples** into 0 mean and unit standard deviation?

Solution: Batch normalization includes ways to 'undo' the effect of batch normalization, allowing the network to use non-normalized inputs / re-shifting and scaling the input distribution. Since the parameters are learned, the network can learn to have no batch norm.

Compared to data pre-processing, batch normalization continues throughout the learning process on all applied layers: introducing long-lasting stability.

ii. **(2 points)** What happens to batch normalization if batch size B is small?

Solution: Batch normalization estimates the true population mean and variance from batches. If B is small, the batch mean and variance will be inaccurate, leading to instability in training.

Layer normalization is another normalization technique that is designed to overcome the drawbacks of batch normalization. Here is the formula for layer normalization:

$$\hat{x} = \frac{x - \mu_L}{\sqrt{\sigma_L^2 + \epsilon}}$$
$$y = \gamma \hat{x}_i + \beta$$
$$\mu_L = \frac{1}{d} \sum_{j=1}^{d} x_j$$
$$\sigma_L = \sqrt{\frac{1}{d} \sum_{j=1}^{d} (x_j - \mu_L)^2}$$

iii. **(2 points)** Based on the definition of layer normalization, describe what is the biggest difference between batch normalization and layer normalization.

Solution: Layer normalization normalizes input across the feature axis instead of normalizing input features across the batch dimension in batch normalization.

iv. **(2 points)** What happened to layer normalization if batch size B is small?

Solution: Nothing will happen because layer normalization doesn't depend on batch size.

v. **(5 points)** In what situation is batch normalization preferred over layer normalization? How about the other way around? Why?

Solution: Batch normalization introduces dependencies between training samples in the same batch. On the other hand, layer normalization only depends on activation of a single sample, removing the dependency between samples.

END OF PAPER