

ECE C147/C247, Fall 2020

Department of Electrical and Computer Engineering
University of California, Los Angeles

Midterm

Prof. J.C. Kao
TAs: W. Feng, J. Lee,
K-F. Liang, M. Kleinman, C. Zheng

UCLA True Bruin academic integrity principles apply.

Open: Four cheat sheets allowed.

Closed: Book, computer, internet.

2:00pm-3:50am.

Wednesday, 19 Feb 2020.

State your assumptions and reasoning.

No credit without reasoning.

Show all work on these pages.

Name: _____

Signature: _____

ID#: _____

Problem 1 _____ / 30

Problem 2 _____ / 20

Problem 3 _____ / 20

Problem 4 _____ / 30

BONUS _____ / 6 bonus points

Total _____ / 100 points + 6 bonus points

1. **Short Answer on ML Basics** (30 points)

- (a) (5 points) Your colleague is tasked with designing the image recognition component for a self-driving vehicle. His goal is to classify images in real-time, identifying objects on the road (like a stop sign, pedestrian, traffic light, etc.) Each image is 1024×1024 pixels². He desires to use a k -nearest neighbors classifier, with distance measured by the Euclidean (L2) norm. What are two reasons he should avoid using the classifier? Justify your answer in no more than 4 sentences.
- (b) (5 points) Batch normalization aims to normalize each artificial neuron so that its mean is 0 and its variance is 1. Why, intuitively, should this help with training deep neural networks? Justify your answer in no more than 4 sentences.

(c) (10 Points) The loss function of a softmax classifier can be written as:

$$\mathcal{L} = \sum_{i=1}^m \left(\log \sum_{j=1}^c e^{a_j(x^{(i)})} - a_{y^{(i)}}(x^{(i)}) \right)$$

where $(\mathbf{x}^{(i)}, y^{(i)})$ are data pairs (of which there are m) and $a_j(x^{(i)}) = \mathbf{w}_j^T \mathbf{x}^{(i)} + b_j$ is the score of class j (there are c total classes). Calculate the gradient of \mathcal{L} with respect to all the weights and biases.

- (d) (10 Points) Assume that you have a handwritten digit image dataset for a classification task. All the images are of size 32×32 . This dataset is split into training, validation and testing. Your CNN takes inputs of size 28×28 . You resize the 32×32 images to 28×28 and then feed them into the network. During training, you observe that the training error is always decreasing while the validation error decreases first and then increases. After training completes, the testing error is much higher than the training error.
- i. (5 Points) What is the problem with your trained CNN? What are 2 potential reasons for this? Justify your answer in no more than 4 sentences.
 - ii. (5 Points) Your colleague suggests that you need a larger training set with more images. However, no additional images are available. What can you do to the existing training images to increase the number of training set examples? Recall that the images are of digits. Justify your answer in no more than 3 sentences.

2. **Backpropagation** (20 points). Suppose you aim to predict \mathbf{y} from \mathbf{x} using the following neural network:

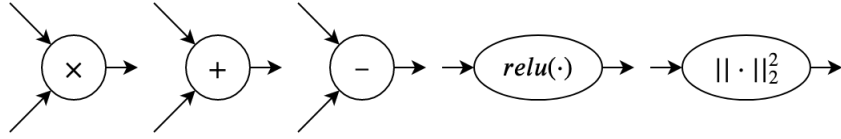
$$\mathbf{z} = \mathbf{W}_3 \text{relu}(\mathbf{W}_1 \mathbf{x}) + \mathbf{W}_2 \mathbf{x}.$$

Here, \mathbf{z} is an estimate of \mathbf{y} and the loss function is:

$$\mathcal{L} = \frac{1}{2} \|\mathbf{z} - \mathbf{y}\|_2^2 = \frac{1}{2} (\mathbf{z} - \mathbf{y})^T (\mathbf{z} - \mathbf{y}),$$

with $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y}, \mathbf{z} \in \mathbb{R}^m$, $\mathbf{W}_1 \in \mathbb{R}^{p \times n}$, $\mathbf{W}_2 \in \mathbb{R}^{m \times n}$, and $\mathbf{W}_3 \in \mathbb{R}^{m \times p}$,

- (a) (4 points) Draw a computational graph of this network to compute the loss \mathcal{L} . You may use the following elements:



- (b) (4 points) What is $\frac{\partial \mathcal{L}}{\partial \mathbf{z}}$ (i.e., $\nabla_{\mathbf{z}} \mathcal{L}$)? Your answer must be simplified to only contain the following allowed terms: $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{W}_1, \mathbf{W}_2$, and \mathbf{W}_3 ? (To be clear, you do not need to simplify \mathbf{z} , since that is an allowed term.)

(c) (12 Points) In terms of $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$, and $\frac{\partial \mathcal{L}}{\partial \mathbf{z}}$, find:

- i. (6 points) $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1}$
- ii. (6 points) $\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$

You should not need to derive any local gradients, as we have derived all of the local gradients for these operations in class. In particular, recall the following:

- For $\mathbf{p} = \mathbf{C}\mathbf{q}$,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{C}} &= \frac{\partial \mathcal{L}}{\partial \mathbf{p}} \mathbf{q}^T \\ \frac{\partial \mathbf{p}}{\partial \mathbf{q}} &= \mathbf{C}^T\end{aligned}$$

- For $\mathbf{v} = \text{relu}(\mathbf{u})$,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \mathbb{1}(\mathbf{u}) \odot \frac{\partial \mathcal{L}}{\partial \mathbf{v}}$$

where $\mathbb{1}(\cdot)$ is the indicator function defined as:

$$\mathbb{1}(\mathbf{u}) = \begin{cases} 1, & \mathbf{u}_i \geq 0 \\ 0, & \mathbf{u}_i < 0 \end{cases}$$

and \odot is the Hadamard (elementwise) product. Be sure also to define any intermediate variables used or label them on the computational graph.

(Additional space for Question 2c)

3. Regularization and Optimization (20 points)

For each statement below, determine whether it is true or false. You must justify your answer to receive full credit.

(a) (5 points) Regularization is helpful to achieve lower loss on the training set.

(b) (5 points) When performing (minibatch) stochastic gradient descent, you are calculating the gradient of the training set exactly.

(c) (5 points) An optimizer using Adagrad will always have an equal or smaller learning rate in successive iterations.

(d) (5 points) You are performing stochastic gradient descent with a learning rate $\epsilon = 5 \times 10^{-6}$. You believe your network is correctly implemented, but when training your network, the value of the loss remains approximately constant across many training iterations. You should try increasing ϵ .

4. **Fully Connected and Convolutional Neural Networks** (30 points)

- (a) **Residual Fully connected networks.** In this problem, we modify the two-layer FC network with residual connections, according to the equation below. Given the input vector $\mathbf{x} \in \mathbb{R}^n$, the forward propagation is,

$$\text{First layer: } \mathbf{h}_1 = f(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{W}_s\mathbf{x}$$

$$\text{Second (Output) layer: } \mathbf{z} = \mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2$$

where $\mathbf{b}_1 \in \mathbb{R}^m$ and $\mathbf{b}_2 \in \mathbb{R}^l$. \mathbf{W}_s is the linear mapping for the "shortcut" connection. f is the non-linear activation function like ReLU.

- i. (3 points) Write the dimension of the following variables: $\mathbf{W}_1, \mathbf{W}_s, \mathbf{W}_2$.

- ii. (3 points) How many trainable parameters does this residual FC network have? Your answer should be expressed in terms of l, m, n .

- iii. (9 points) To select a proper activation function f , we are considering whether it: (1) saturates (and if so, for what values does it saturate), (2) is differentiable everywhere, (3) approximates identity near the origin (activation functions where $f(0) = 0$ and $f'(0) = 1$ and f' is continuous at 0 have this property.). For the following candidate activation functions, explain from the three aspects and **Briefly** justify each solution you give.

A. Hyperbolic tangent: $\tanh(x)$.

B. Leaky ReLU: $\max(0.01x, x)$.

C. Gaussian Error Linear Unit (GELU): it is popular in recent transformer structures like BERT. This activation function can be approximated as $\text{GELU}(x) = 0.5x(1 + \tanh[\sqrt{2/\pi}(x + 0.044715x^3)])$

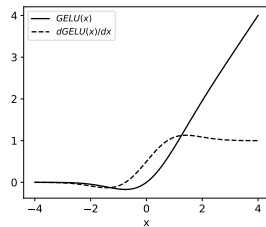


Figure 1: Gaussian Error Linear Unit (GELU)

- (b) (15 points) **Convolutional neural networks.** Consider a convolutional neural network (CNN) that is used to classify CIFAR-10 images that are $32 \times 32 \times 3$. The first layer of the CNN is a convolutional layer, whose output is $26 \times 26 \times 16$.
- i. (5 points) How many filters are in the first convolutional layer, and what are the dimensions of each filter? Assume there is no zero padding and the stride is equal to 1.

- ii. (5 points) There is a second convolutional layer immediately after the first convolutional layer. In the second convolutional layer, there are 32 filters that are 3×3 . What is the depth of each filter, and how many total trainable parameters are there in this layer? (Do not include biases in the number of parameters.)

- iii. (5 points) In general, why do convolutional layers have less parameters than fully connected layers? Justify your answer in no more than 3 sentences.

5. **(Bonus, 6 points).** Binary cross-entropy loss. In class, we have said that maximizing the likelihood of the observed data is equivalent to minimizing the “cross-entropy” loss in machine learning. Consider that we have data: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(N)}, y^{(N)})$. The data comes from one of two classes, class 0 or class 1, i.e., $y^{(j)} \in \{0, 1\}$. For each input sample, $\mathbf{x}^{(j)}$, we calculate $\hat{y}^{(j)}$, which is the probability that $y^{(j)} = 1$, i.e.,

$$\hat{y}^{(j)} = \Pr(y^{(j)} \text{ belongs to class 1} | \mathbf{x}^{(j)}, \theta)$$

Though not necessary for this question, this probability could be given by, for example, the sigmoid function:

$$\begin{aligned} \hat{y}^{(j)} &= \sigma(\theta, \mathbf{x}^{(j)}) \\ &= \frac{1}{1 + \exp(-\theta^T \mathbf{x}^{(j)})} \end{aligned}$$

The cross-entropy loss is:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{j=1}^N \left[y^{(j)} \log \hat{y}^{(j)} + (1 - y^{(j)}) \log(1 - \hat{y}^{(j)}) \right]$$

Show that minimizing the cross-entropy loss is equivalent to maximizing the likelihood of the data,

$$\arg \max_{\theta} p((\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(N)}, y^{(N)}) | \theta).$$

Assume that each data point $(\mathbf{x}^{(j)}, y^{(j)})$ is independent given the parameters θ (i.e., the same assumption we made in class to simplify the joint probability).

Hint: When $y^{(j)}$ is binary, show that $p(y^{(j)} | \mathbf{x}^{(j)}, \theta)$, which is the probability that $\mathbf{x}^{(j)}$ belongs to class $y^{(j)}$ under a model with parameters θ , can be written as

$$p(y^{(j)} | \mathbf{x}^{(j)}, \theta) = (\hat{y}^{(j)})^{y^{(j)}} \cdot (1 - \hat{y}^{(j)})^{1-y^{(j)}}$$

Additional space for bonus question.