CSCI567 Machine Learning (Fall 2025)
Instructor: Haipeng Luo

Exam One (Sample)

Duration: 120 minutes

NAME _____

Stu ID _____

INSTRUCTIONS:

1. Please fill in your name and ID in the space above.
2. When the exam ends, you have 20 minutes to upload your solutions to Gradescope using your phone similarly to how you upload your homework (**do not use your phone for any reason before that!**). Make sure to match your solutions to the corresponding questions. **Do not continue working on the exam during these last 20 minutes.**
3. After finishing uploading, please also turn in your exam before leaving the room.
4. If you encounter any difficulty in uploading (or you are taking the exam in a test center that prohibits cell phones), simply turn in your exam and we will upload it for you.
5. This exam has a total of 18 pages (including this cover page). Check and make sure that you are not missing any pages. Additionally, there are 5 pages provided for scratch work at the end.
6. Any kind of cheating will be reported and lead to a 0 score for the entire exam.

1 Multiple-choice Questions (30 points)

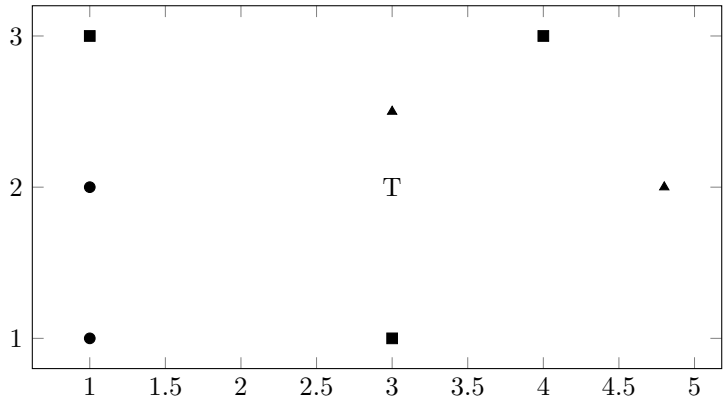
IMPORTANT: *Select ALL answers that you think are correct. You get 0.5 point for selecting each correct answer and similarly 0.5 point for not selecting each incorrect answer.*

(1) Which of the following on machine learning is correct?

(A) Cross-validation is often used to tune the hyper-parameters of a machine learning algorithm.
(B) Overfitting refers to the phenomenon when the training error is low but the test error is high.
(C) One should prevent overfitting by using the test set during training.
(D) Logistic regression is an algorithm for regression tasks.

(2) Consider the following two-dimensional dataset with $N = 7$ training points of three classes (triangle, square, and circle), and additionally one test point denoted by the letter T. Which of the following configuration of the K-nearest neighbor algorithm will predict square for the test point?

(A) $K = 1$, L2 distance
(B) $K = 3$, L1 distance
(C) $K = 3$, L2 distance.
(D) $K = 7$, any distance.



(3) Which of the following on linear regression is correct?

(A) The least square solution has a closed-form formula, even if L1 regularization is applied.
(B) The covariance matrix $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is not invertible if the number of data points N is smaller than the dimension D.
(C) When the covariance matrix $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is not invertible, the Residual Sum of Squares (RSS) objective has infinitely many minimizers.
(D) Linear regression is a parametric method.

(4) Perceptron algorithm is applying SGD with learning rate $\eta = 1$ to the Perceptron loss, and $\boldsymbol{w}^* = \mathbf{0}$ is clearly a minimizer of the Perceptron loss. Which of the following statements related to these two facts is correct?

(A) Perceptron always converges to $\boldsymbol{w}^* = \mathbf{0}$.
(B) Perceptron converges to $\boldsymbol{w}^* = \mathbf{0}$ when it is initialized at $\mathbf{0}$.
(C) Perceptron does not converge to $\boldsymbol{w}^* = \mathbf{0}$ because the learning rate does not decrease over time.
(D) Perceptron does not converge to $\boldsymbol{w}^* = \mathbf{0}$ because SGD uses a stochastic gradient instead of the

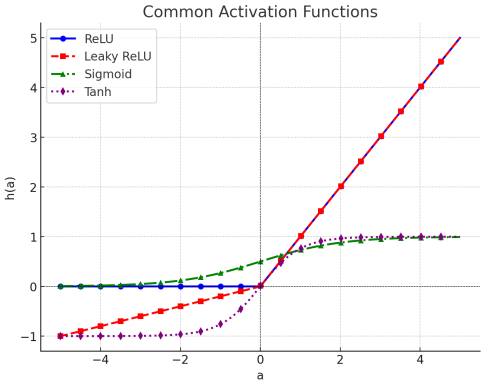
exact gradient.

(5) Which of the following statements is correct when running SGD to minimize a non-convex function?

(A) SGD with random initialization can still get stuck at saddle points.
(B) SGD with random initialization escapes all saddle points with high probability.
(C) As long as we run SGD long enough, it will find a local minimum.
(D) As long as we run SGD long enough, it will find a global minimum.

(6) Which of the following activation functions has a vanishing derivative when its input becomes too large (either too positive or too negative)? You might find the plot below useful.

(A) ReLU: $h(a) = \max\{a, 0\}$
(B) Leaky ReLU: $h(a) = \begin{cases} a & \text{if } a \geq 0 \\ 0.2a & \text{else} \end{cases}$
(C) Sigmoid: $h(a) = \frac{1}{1+e^{-a}}$
(D) TahH: $h(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$



(7) Which of the following about neural nets is correct?

(A) A fully-connected neural net with a million neurons can approximate any continuous function.
(B) Data augmentation is useful for preventing overfitting when training a nerual net.
(C) Momentum and adaptive learning rate are useful for speeding up the training of a neural net.
(D) One should keep running Backpropagation until the training error goes down to 0.

(8) Suppose that a convolution layer takes a 4×6 image with 3 channels as input and outputs a $3 \times 4 \times 10$ volume. Which of the following is a possible configuration of this layer?

(A) Ten 2×3 filters with depth 10, stride 1, and no zero-padding.
(B) Ten 2×2 filters with depth 3, stride 2, and 1 pixel of zero-padding.
(C) Ten 2×2 filters with depth 3, stride 2, and 2 pixels of zero-padding.
(D) One 2×3 filter with depth 10, stride 1, and no zero-padding.

(9) How many parameters do we need to learn for the following network structure? An $8 \times 8 \times 3$ image input, followed by a convolution layer with 8 filters of size 3×3 (stride 1 and 1 pixel of zero-padding), then another convolution layer with 4 filters of size 2×2 (stride 2 and no zero-padding), and finally an average pooling layer with a 2×2 filter (stride 2 and no zero-padding). (Note: the depth of all filters are not explicitly spelled out, and we assume no bias/intercept terms used.)

(A) 144 (B) 344 (C) 348 (D) 360

(10) What is the final output dimension of the last question?

(A) $4 \times 4 \times 1$ (B) $4 \times 4 \times 4$ (C) $2 \times 2 \times 1$ (D) $2 \times 2 \times 4$

(11) Suppose that k_1 and k_2 are two kernel functions with $\boldsymbol{\phi}_1 : \mathbb{R}^D \to \mathbb{R}^M$ and $\boldsymbol{\phi}_2 : \mathbb{R}^D \to \mathbb{R}^M$ being the corresponding feature maps. Which of the following is the corresponding feature map $\boldsymbol{\phi}$ for the product of k_1 and k_2 (which we know is also a kernel function)?

(A) $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}_1(\boldsymbol{x})\boldsymbol{\phi}_2(\boldsymbol{x})^\top \in \mathbb{R}^{M \times M}$
(B) $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}_1(\boldsymbol{x})^\top \boldsymbol{\phi}_2(\boldsymbol{x}) \in \mathbb{R}$
(C) $\boldsymbol{\phi}(\boldsymbol{x}) = (\boldsymbol{\phi}_1(\boldsymbol{x}), \boldsymbol{\phi}_2(\boldsymbol{x})) \in \mathbb{R}^{2M}$
(D) $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}_1(\boldsymbol{x}) \circ \boldsymbol{\phi}_2(\boldsymbol{x}) \in \mathbb{R}^M$ (element-wise product)

(12) Which of the following on SVM is correct? In case you need a reminder, the primal formulation of SVM is $\min_{\boldsymbol{w},b,\{\xi_n\}} C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2$ subject to $1 - y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) \le \xi_n$ and $\xi_n \ge 0$ for all n, and the dual formulation can be found in Problem 4.1.

(A) The larger the hyper-parameter C, the smaller the amount of L2 regularization.
(B) The larger the hyper-parameter C, the larger the amount of L2 regularization.
(C) To handle multiclass classification, one can use the one-versus-one reduction together with SVM.
(D) The α coefficients obtained by SVM satisfy $\sum_{n:y_n=+1} \alpha_n = \sum_{n:y_n=-1} \alpha_n$.

(13) Which of the following about decision trees is correct?

(A) Good interpretability is a key advantage of decision trees.
(B) Decision tree algorithms are usually implemented using recursion.
(C) Shannon entropy can be used to measure the uncertainty of a node when building a decision tree.
(D) Random forest is a random ensemble of decision trees.

(14) Consider a binary dataset with 50 positive examples and 50 negative examples. Decision stump $\mathcal{T}_1$ splits this dataset into two children where the left one has 20 positive examples and 40 negative examples, while another decision stump $\mathcal{T}_2$ results in a left child with 25 positive examples and 25 negative examples. Which of the following is correct? (Recall that entropy is defined as $H(P) = -\sum_{k=1}^{C} P(Y=k)\log P(Y=k)$.)

(A) The entropy of the left child of $\mathcal{T}_1$ is $\frac{1}{3}\log 3 + \frac{2}{3}\log\frac{3}{2}$.
(B) The entropy of the right child of $\mathcal{T}_1$ is $\frac{1}{4}\log 4 + \frac{3}{4}\log\frac{4}{3}$.
(C) The entropy of either child of $\mathcal{T}_2$ is $\frac{1}{2}\log 2 + \frac{1}{2}\log 2$.
(D) Based on conditional entropy, $\mathcal{T}_1$ is a better split than $\mathcal{T}_2$.

(15) Which of the following about boosting is correct?

(A) Boosting is guaranteed to achieve zero training error.
(B) AdaBoost is often resistant to overfitting.
(C) AdaBoost never overfits.
(D) The idea of boosting is to repeatedly reweight the examples so that "difficult" ones get more attention.

2 Linear Regression with Huber Loss (16 points)

In the lecture, we discussed the pros and cons of using the squared loss $\ell_{\text{sq}}(r) = r^2$ versus the absolute loss $\ell_{\text{abs}}(r) = |r|$ for a regression task, where the residual r is the different between a prediction and the true outcome. To combine the advantages of ℓ_{sq} and ℓ_{abs}, one could use the *Huber loss* instead, defined as

$$\ell_\delta(r) = \begin{cases} \frac{1}{2}r^2, & \text{if } |r| \leq \delta, \\ \delta(|r| - \frac{1}{2}\delta), & \text{else,} \end{cases}$$

for some parameter $\delta > 0$. The intuition is that for a small residual ($|r| \leq \delta$), the Huber loss behaves the same way as the squared loss, while for a large residual ($|r| \geq \delta$), the Huber loss behaves the same way as the absolute loss.

2.1 Write down the derivative $\ell'_\delta(r)$ of $\ell_\delta(r)$ with respect to r (okay to skip reasoning). (6 points)

2.2 To apply Huber loss to linear regression for a training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^D \times \mathbb{R}$, we consider the following objective function

$$F(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} \ell_\delta(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n), \tag{1}$$

and apply SGD to minimize it. Fill in the missing details in the following implementation of this idea. No reasoning is required, but the stochastic gradient should be written down explicitly (instead of using a generic form such as ℓ'_δ). (6 points)

Algorithm 1: SGD for minimizing Eq. (1)

Input: A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^D \times \mathbb{R}$, learning rate $\eta > 0$, and parameter $\delta > 0$.
Initialization: $\boldsymbol{w} = \boldsymbol{0} \in \mathbb{R}^D$.
Repeat:

2.3 Suppose that you believe the dataset contains outliers. Would you use a large or a small value of δ (explain briefly)? More generally, if you do not have any prior knowledge on the dataset, how would to decide the value of δ? (4 points)

3 Multiclass Logistic Regression and Kernel (16 points)

In the lecture, we discussed multiclass logistic regression, with the following (partial) implementation.

Algorithm 2: SGD for minimizing multiclass logistic loss

1 **Input:** A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^D \times [C]$, learning rate $\eta > 0$.
2 **Initialize** $\boldsymbol{W} \in \mathbb{R}^{C \times D}$ randomly.
3 **Repeat:**
4 Randomly pick an example $(\boldsymbol{x}_n, y_n)$.
5 Compute $\mathbb{P}(y = c \mid \boldsymbol{x}_n; \boldsymbol{W})$ for each $c \in [C]$ (using softmax):
6 Update the parameters: $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \begin{pmatrix} \mathbb{P}(y = 1 \mid \boldsymbol{x}_n; \boldsymbol{W}) \\ \vdots \\ \mathbb{P}(y = y_n \mid \boldsymbol{x}_n; \boldsymbol{W}) - 1 \\ \vdots \\ \mathbb{P}(y = C \mid \boldsymbol{x}_n; \boldsymbol{W}) \end{pmatrix} \boldsymbol{x}_n^{\mathrm{T}}$.

3.1 Fill in the missing detail in Line 5 of Algorithm 2 (no reasoning required). You can use $\boldsymbol{w}_1^\top, \ldots, \boldsymbol{w}_C^\top$ to denote the rows of $\boldsymbol{W}$. (4 points)

3.2 Just like multiclass Perceptron in HW2, we once again see that the parameters $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C$ computed by Algorithm 2 are always linear combinations of the training points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, that is, $\boldsymbol{w}_c = \sum_{n=1}^{N} \alpha_{c,n} \boldsymbol{x}_n$ for some coefficient $\alpha_{c,n}$. This means that one can kernelize the algorithm for any given kernel function $k(\cdot, \cdot)$ (with a corresponding feature mapping $\boldsymbol{\phi}$).

Specifically, fill in the missing details in the repeat-loop of the algorithm below which maintains and updates weights $\alpha_{c,n}$, $\forall c \in [C], n \in [N]$ such that $\boldsymbol{w}_c = \sum_{n=1}^{N} \alpha_{c,n} \boldsymbol{\phi}(\boldsymbol{x}_n)$ is always the same as what one would get by running Algorithm 2 with $\boldsymbol{x}_n$ replaced by $\boldsymbol{\phi}(\boldsymbol{x}_n)$ for all n. No reasoning is required. Keep in mind that in your solution $\boldsymbol{\phi}(\boldsymbol{x}_n)$ should never appear. (8 points)

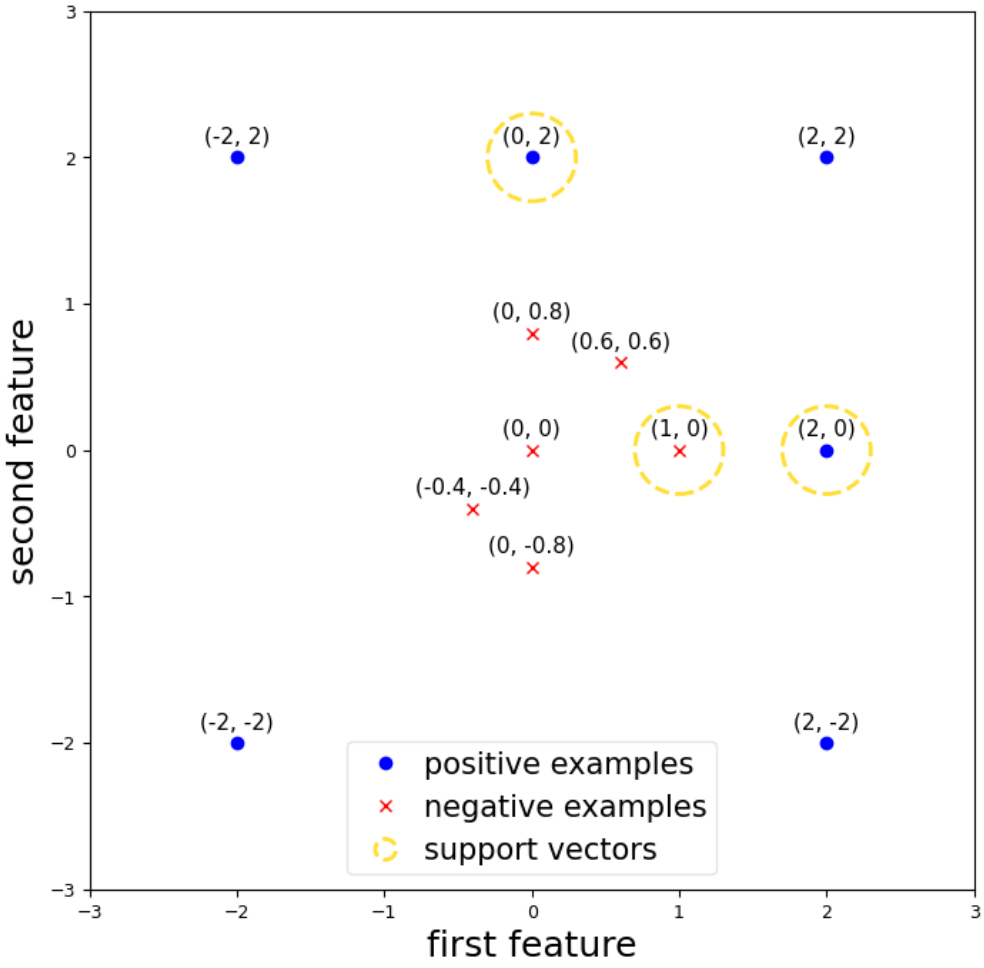
Algorithm 3: Kernelized Multiclass Logistic Regression

Input: A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^D \times [C]$, learning rate $\eta > 0$, a kernel function $k(\cdot, \cdot)$.
Initialize: $\alpha_{c,n} = 0$ for all $c \in [C]$ and $n \in [N]$.
Repeat:

3.3 After training with Algorithm 3, describe how you would make a prediction given a a new test point $\boldsymbol{x}$. (4 points)

4 SVM (12 points)

Consider the following 2-dimensional dataset of 12 examples for a binary classification task, where positive examples (i.e., with +1 label) are marked as blue dots and negative examples (i.e., with −1 label) are marked as red crosses. The two features of each example are also shown above its label.



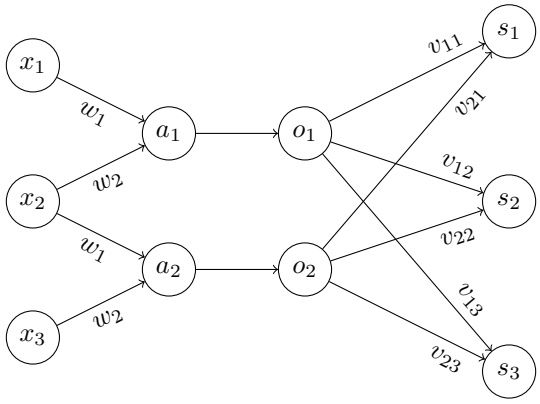
4.1 When running SVM on this dataset with a polynomial kernel $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^\top \boldsymbol{x}')^2$ and parameter $C = +\infty$, it turns out that there are only 3 support vectors, highlighted by dash circles in the figure above. Based on this information, simplify the following general SVM dual formulation by plugging in the actual values of this dataset. (Please write down your reasoning.) (6 points)

$$\begin{aligned} \max_{\alpha_1,\ldots,\alpha_N} \quad & \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} y_m y_n \alpha_m \alpha_n k(\boldsymbol{x}_m, \boldsymbol{x}_n) \\ \text{s.t.} \quad & \sum_{n=1}^{N} \alpha_n y_n = 0 \quad \text{and} \quad \alpha_n \geq 0, \quad \forall\ n \end{aligned}$$

4.2 Solve the dual formulation and obtain the α coefficients for the three support vectors. (Hint: the solutions are fractional numbers.) (6 points)

5 Backpropagation (26 points)

Consider the following mini convolutional neural net, where (x_1, x_2, x_3) is the 3-dimensional input, followed by a convolution layer with a filter (w_1, w_2), a ReLU activation layer, and finally a fully connected layer with 3 outputs that correspond to the scores of 3 classes.



More concretely, the computation is specified by

$$
\begin{aligned}
a_1 &= x_1 w_1 + x_2 w_2, \\
a_2 &= x_2 w_1 + x_3 w_2, \\
o_1 &= \max\{0, a_1\}, \\
o_2 &= \max\{0, a_2\}, \\
\begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} &= \boldsymbol{V} \begin{pmatrix} o_1 \\ o_2 \end{pmatrix}, \text{ where } \boldsymbol{V} = \begin{pmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \\ v_{13} & v_{23} \end{pmatrix}.
\end{aligned}
$$

For an example $(\boldsymbol{x}, y) \in \mathbb{R}^3 \times \{1, 2, 3\}$, the cross-entropy loss of the CNN is $\ell = \ln\left(1 + \sum_{c \neq y} e^{s_c - s_y}\right)$, which is a function of the parameters of the network: w_1, w_2, and $\boldsymbol{V}$. In the following questions, you are asked to calculate the derivative of ℓ with respect to some variable in terms of its derivative with respect to some other variables, and you can (and should in some cases) express your final solutions in terms of **other neurons/parameters of the network, but not other derivatives that are not mentioned**. You will need to use the following chain rule discussed in the lecture:

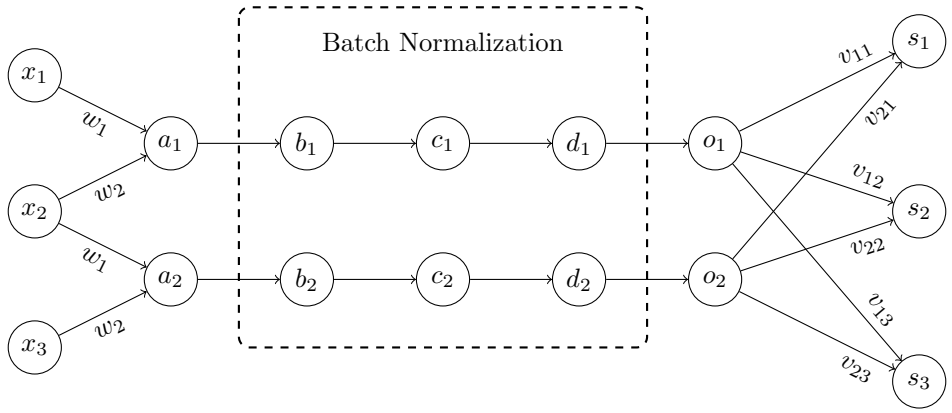
- for a composite function $f(g_1(z), \ldots, g_d(z))$, we have $\frac{\partial f}{\partial z} = \sum_{k=1}^{d} \frac{\partial f}{\partial g_k} \frac{\partial g_k}{\partial z}$.

In your solutions, **please show the intermediate steps that apply this chain rule.**

5.1 During training, the update of the last fully-connected layer (that is, parameter $\boldsymbol{V}$) is similar to the standard multiclass logistic regression, so we focus on the update of the first convolution layer (that is, parameters w_1 and w_2). To this end, first write down $\frac{\partial \ell}{\partial w_1}$ in terms of $\frac{\partial \ell}{\partial a_1}$ and $\frac{\partial \ell}{\partial a_2}$. (3 points)

5.2 Next, write down $\frac{\partial \ell}{\partial a_1}$ in terms of $\frac{\partial \ell}{\partial s_1}$, $\frac{\partial \ell}{\partial s_2}$, and $\frac{\partial \ell}{\partial s_3}$. (The derivative of the ReLU function is $H(a) = \mathbb{I}[a > 0]$, which you can use directly in your answer.) (4 points)

In the remaining questions, consider adding the Batch Normalization layer immediately after the convolution layer, as shown in the picture below.



More specifically, during one epoch of backpropagation, suppose that we sample a batch of M training points and denote their corresponding outputs after the convolution layer as $a_1^{(1)}, a_2^{(1)}, \ldots, a_1^{(M)}, a_2^{(M)}$. Then, what the Batch Normalization layer does for the m-th training point in the batch is as follows, where we ignore the subscript 1 or 2 since the operations are the same (and **you should do the same as well in your solution**):

$$b^{(m)} = a^{(m)} - \mu, \text{ where } \mu = \frac{1}{M}\sum_{k=1}^{M} a^{(k)}, \qquad \text{(shifted by the mean)}$$

$$c^{(m)} = \frac{b^{(m)}}{\sigma}, \text{ where } \sigma = \sqrt{\frac{1}{M}\sum_{h=1}^{M}(b^{(h)})^2}, \qquad \text{(divided by the standard deviation)}$$

$$d^{(m)} = \gamma c^{(m)} + \beta. \qquad \text{(another linear transformation)}$$

Here, the last linear transformation step parameterized by two numbers γ and β (that are shared by all training points in the batch, but different for $c_1^{(m)}, d_1^{(m)}$ and $c_2^{(m)}, d_2^{(m)}$) is to make sure that the network can still represent the original architecture that has no batch normalization (if that happens to be the right thing to do). To train this new architecture, we need to additionally figure out $\frac{\partial \ell}{\partial \gamma}$ and $\frac{\partial \ell}{\partial \beta}$ to update γ and β, and also $\frac{\partial \ell}{\partial a^{(m)}}$ in order to update w_1 and w_2. Please do so following the steps below (which eventually express all these derivatives in terms of $\frac{\partial \ell}{\partial d^{(m)}}$).

5.3 Write down $\frac{\partial \ell}{\partial \beta}$ in terms of $\frac{\partial \ell}{\partial d^{(m)}}$ for $m = 1, \ldots, M$. (3 points)

5.4 Write down $\frac{\partial \ell}{\partial \gamma}$ in terms of $\frac{\partial \ell}{\partial d^{(m)}}$ for $m = 1, \ldots, M$. (3 points)

5.5 For a fixed $m \in [M]$, write down $\frac{\partial \ell}{\partial c^{(m)}}$ in terms of $\frac{\partial \ell}{\partial d^{(m)}}$. (2 points)

5.6 For a fixed $m \in [M]$, write down $\frac{\partial \ell}{\partial a^{(m)}}$ in terms of $\frac{\partial \ell}{\partial b^{(h)}}$ for $h = 1, \ldots, M$. (4 points)

5.7 For a fixed $m \in [M]$, write down $\frac{\partial \ell}{\partial b^{(m)}}$ in terms of $\frac{\partial \ell}{\partial c^{(k)}}$ for $k = 1, \ldots, M$. (7 points)

You can use this page for scratch work

You can use this page for scratch work

You can use this page for scratch work

You can use this page for scratch work

You can use this page for scratch work