STANFORD UNIVERSITY
CS 229, Autumn 2017
Midterm Examination Solutions

Question	Points
1 Short answers	/25
2 Linear regression with noisy targets	/17
3 Generalized discriminant analysis	/12
4 Kernels on discrete sequences	/7
5 EM for the exponential family	/25
6 NNs with shortcut connections	/14
Total	/100

Name of Student: ______________________________

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.

Signed: ______________________________

1. **[25 points] Short answers**

The following questions require a reasonably short answer (usually at most 2-3 sentences or a figure for each question part, though some may require longer or shorter explanations).

No credit will be given on true/false and multiple choice questions for answers without a correct explanation.

(a) **Softmax regression** [4 points] In this question we'll consider softmax regression.

i. [2 points] Show that the negative log-likelihood for a k-class softmax regression classifier for a *single example* (x, y) where $y = k$ and

$$p(y = i|x;\theta) = \frac{\exp(\theta_i^\top x)}{\sum_{j=1}^k \exp(\theta_j^\top x)} \qquad (1)$$

can be written as an expression of the form $\log \sum_{j=1}^k e^{z_j}$.

Answer: First, dividing the ratio by the numerator, we have

$$p(y = i|x;\theta) = \frac{1}{\sum_{j=1}^k \exp(\theta_j^\top x - \theta_i^\top x)}$$

Then we have the negative log-likelihood

$$-\ell(\theta) = \log \sum_{j=1}^k \exp(\theta_j^\top x - \theta_i^\top x)$$

in the desired form where $z_j = \theta_j^\top x - \theta_i^\top x$.

Remark: The composition of a convex function with an affine function is convex. Hence if we have $\ell(\theta) = f(\theta^\top x)$, to show that $\ell(\theta)$ is convex we need only show that $f(z)$ is convex in z. Second, any function of the form $f(z) = \log \sum_{i=1}^n e^{z_i}$ (log-sum-exp), where $z \in \mathbb{R}^n$, is convex in z. Thus, if we can write the softmax in the form of a log-sum-exp function, we know that it is convex.

ii. [2 points] Suppose you implement your softmax classifier using formula (1) given in the first part of this question. When testing out your softmax regression classifier, you find you often have numerical underflow issues when computing the probability for each class (meaning that one of the numbers computed in the numerator or denominator is smaller than the representable range using the number of allocated bits, e.g. if it is smaller than 2^{-149} for 32-bit floats). However, you're confident that after dividing the numerator by the denominator, the final values for the softmax probabilities are in the representable range.

How would you try and remedy the underflow issue? Please give an update to the procedure for computing the softmax; your solution must not affect the computed softmax probabilities and must also give $p(y = i|x; \theta)$, not just $\log p(y = i|x; \theta)$.

Answer: To deal with this issue the standard approach is to subtract $\max z_j$ from all the z_j *prior* to exponentiating. For example, if we had the final expression

$$p(y|x; \theta) = \frac{e^{-150}}{e^{-150} + e^{-152}}$$

by doing this we would subtract -150 from all the exponents beforehand and thus instead compute

$$p(y|x; \theta) = \frac{e^{0}}{e^{0} + e^{-2}}.$$

(b) **Optimization** [4 points] Suppose you have two models you would like to train on a dataset $X \in \mathbb{R}^{100,000 \times 1024}$ $(100,000$ examples with a input feature dimension of 1024). The task is binary classification; hence Y is a 100,000 dimensional vector with binary entries. The first model is a logistic regression model with $\theta \in \mathbb{R}^{1024}$, and the second is a neural network with a single hidden layer containing 1024 hidden units that uses sigmoid nonlinearities for both the hidden and output layer. Given the number of examples, computing the gradient for every example in the dataset takes considerable time for both the logistic regression as well as the neural network model.

For each of the following optimization methods, state whether there may be significant computational drawbacks (time required to run the algorithm) to using the optimization method to train each of the two models. If there issues, explain **briefly** why these issues would occur. (We will not grade extremely verbose answers.) You may assume for each case you can select an appropriate (but fixed) learning rate α. Unless otherwise indicated, assume parameters are initialized randomly such that the initial parameter setting does not cause any issues with optimization.

i. [1 points] stochastic gradient descent

A. logistic regression

B. neural network

Answer: Stochastic gradient descent should work fine in each case as it's a first-order method which only requires computing the gradient.

ii. [1 points] batch gradient descent

A. logistic regression

B. neural network

Answer: As mentioned in the problem statement, computing the gradient for all examples in the dataset takes a while, so batch gradient descent would be considerably slower than SGD.

iii. [1 points] Newton's method

A. logistic regression

B. neural network

Answer: Newton's method should work for logistic regression. However, even a single hidden layer neural network in this case will have 1024×1024 parameters in the input-to-hidden weight matrix alone; hence inverting the Hessian will be infeasible.

iv. [1 + 2 extra credit points] Finally, suppose we use stochastic gradient descent with parameters θ initialized to 0. Will there be any issues with optimization for each of the two models? *Extra credit*: How would you expect each model to perform? For this part, you should carefully explain your reasoning.

A. logistic regression

B. neural network

Answer: There should not be any issues for logistic regression (refer to the gradient update derived in the homework to verify this); however, the neural network will have the same gradient update for all hidden units and hence won't be able to fit the data as well as it should.

[Extra credit] Note also that although there is the issue of symmetry breaking, the network can still learn updates. One way to see this is to simply treat the first hidden layer's units as the inputs in which case we have (non-zero, due to the sigmoid used to compute $a^{[1]}$) updates to $W^{[2]} \in \mathbb{R}^{1024}$ like the updates we would have for logistic regression. The updates will be the same for all entries of $W^{[2]}$, giving only 1 free parameter. The gradient will then propagate back from $a^{[1]}$ to $W^{[1]}$ where each row of $W^{[1]}$ receives the same update. However, $W^{[1]} \in \mathbb{R}^{1024 \times 1024}$ still has 1024 free parameters. So we end up with a model that should perform similarly to logistic regression (where $\theta \in \mathbb{R}^{1024}$).

(c) **Debugging machine learning algorithms** [7 points] For each of the following scenarios, consider whether the issue described is due to either: poor optimization (including poor parameter initialization), data mismatch, choice of model or features, or amount of regularization. Assume the issues are due these causes and no other possible causes. There may be more than one valid answer in some cases. Give a brief explanation for your answer.

i. [2 points] You tried implementing an SVM with a polynomial kernel to overfit on a tiny subset of your training data, but your training accuracy never exceeds 90%, even though an off-the-shelf linear SVM you trained was able to achieve 100% training accuracy on the subset. No regularization was applied in either case.

Answer: The SVM as described should be able to overfit. We are considering the training set so data mismatch is not a consideration, and regularization will only hurt training performance. Optimization must be the issue.

ii. [2 points] Your model attains good performance on the training set but poor performance on the train-dev set.

Answer: Since training performance is fine, optimization is likely not the issue. But it seems overfitting has occurred. The train-dev set is also split off from the original training set, so data mismatch should not be the issue. Poor choice of model or too little regularization are both possible causes.

iii. [1 points] Your model attains good performance on the training and train-dev sets but performs poorly on the dev set.

Answer: Since training performance and train-dev performance are fine, the issue is not likely to be optimization, model/features, or regularization. Instead, this is a case of *data mismatch* between the training and dev set.

iv. [2 points] Your model attains mediocre performance on both your train and train-dev set. You have not yet applied any regularization.

Answer: Regularization would only hurt training performance, and the train and train-dev set both come from the training distribution, so data mismatch is not the culprit. Either optimization or a poorly chosen model/poorly chosen features are possible issues.

(d) **Naive Bayes** [5 points] Recall that there two common forms of Naive Bayes when dealing with discrete data: multivariate and multinomial Naive Bayes. Let's briefly consider the trade-offs when using each. As described in the lecture notes, assume we are performing binary spam classification with a vocabulary size V, and let's assume that the average length of an email is L words.

i. [1 points] How many parameters are learned for multivariate Naive Bayes? How about multinomial Naive Bayes?
Answer: For both cases, we have one parameter for $p(y = 1)$. For multivariate NB we also have $2V$ additional parameters, and same for multinomial NB since we assume the $p(x_j|y)$ is the same for all positions j in an email. Hence in both cases we have $1 + 2V$ parameters.

ii. [1 points] When making a prediction of the most likely class, how many lookups of $p(x_j|c)$ does each method require, on average, to make a single prediction (spam or not spam) for an email? Express your answer in terms of V and L. **Answer:** Multivariate NB involves $2V$ lookups (V lookups for each class). Multinomial NB involves $2L$ lookups (L lookups for each class). Note that we ask for the number of operations required to compute the most likely class, not the probability of each class (which requires we compute the normalizing denominator).

iii. [2 points] In text classification, it's common to give special consideration to *stop words*, which are words such as "a" and "the" that commonly appear no matter the class y, and thus tend to be uninformative. We would hope that for any stop word x, $p(x|y = 0) \approx p(x|y = 1) \approx 0.5$, and hence the presence of x would not affect our predictions. However, suppose we somehow arrived at a bad estimate for the stop word x such that $p(x|y = 1) = 0.95$. Which method, multivariate or multinomial Naive Bayes, would you expect to be most adversely affected by this bad estimate? Why?

Answer: We would expect the probabilities for multinomial NB to be affected more, since the probability $p(x|y)$ is factored in every time x appears, whereas multivariate NB only considers whether x appears at all or not.

iv. [1 points] In both multivariate and multinomial Naive Bayes, we assume that every feature x_i is independent of all other features x_j, $i \neq j$. True or false? Explain briefly. **Answer:** False. In both cases, x_i and x_j are only *conditionally* independent given the class y.

(e) **Gradient descent and k-means** [5 points]

In this question, we will connect gradient descent with the k-means clustering algorithm. Recall that for a training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$, the distortion (or loss) function for k-means with k clusters is defined as:

$$J(c, \mu) = \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|_2^2,$$

Thus, $J(c, \mu)$ measures the sum of squared distances between each training example $x^{(i)}$ and the cluster centroid $\mu_{c^{(i)}}$ to which it has been assigned. Let's consider minimizing $J(c, \mu)$ using gradient descent while keeping c (the assignment of data points to centroids) fixed.

i. [3 points] Derive the update formula for μ_j, for any $j \in \{1, 2, \dots, k\}$, with batch gradient descent and using learning rate α. **Answer:** We have

$$\begin{aligned} J(c, \mu) &= \sum_{i=1}^{m} (x^{(i)} - \mu_{c^{(i)}})^\top (x^{(i)} - \mu_{c^{(i)}}) \\ \nabla_{\mu_j} J(c, \mu) &= \sum_{i:c^{(i)}=j} (2\mu_j - 2x^{(i)}) \end{aligned}$$

Thus we have the update:

$$\mu_j := \mu_j - 2\alpha \sum_{i:c^{(i)}=j} (\mu_j - x^{(i)})$$

ii. [2 points] In the update step of the standard k-means clustering algorithm, we update each centroid μ_j by setting each centroid to the mean of the $x^{(i)}$ assigned to that centroid. What value of the learning rate α_j results in both algorithms (batch gradient descent and k-means) performing the same update for each μ_j? (*Hint*: The learning rate may be different for each cluster j.) **Answer:** Inspecting the update rule above, we see that in order for μ_j to disappear in the updated value, for each cluster j, we must have

$$\alpha_j = \frac{1}{2|\{i : c^{(i)} = j\}|}$$

and this yields

$$\mu_j := \frac{\sum_{i:c^{(i)}=j} x^{(i)}}{|\{i : c^{(i)} = j\}|}$$

as desired.

2. **[17 points] Linear regression for noisy targets**

Often the targets $y^{(i)}$ we get when constructing our dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$ can be very noisy; in these cases the least squares objective can be overly sensitive to outliers. In this question we will consider approaches for dealing with noisy labels using weighted least squares.

Part 1 Recall the weighted least squares objective

$$J(\theta) = \sum_{i=1}^{m} w^{(i)}(\theta^\top x^{(i)} - y^{(i)})^2$$

For this question you may assume $w^{(i)} > 0$, $i = 1, \ldots, m$.

(a) [4 points] Show that $J(\theta)$ can be written in the form of $z^\top z$ where $z \in \mathbb{R}^m$ has entries $z_j \geq 0$. Give the expression for z. Intuitively, what does z represent?

Answer: Recall that we can write

$$J(\theta) = (X\theta - Y)^\top W(X\theta - Y)$$

where $W \in \mathbb{R}^{m\times m} = \mathrm{diag}([w^{(1)}, w^{(2)}, \ldots, w^{(m)}])$. Since $w^{(i)} > 0$, we can write W as $P^\top P$, where

$$P = \mathrm{diag}([\sqrt{w^{(1)}}, \sqrt{w^{(2)}}, \ldots, \sqrt{w^{(n)}}]).$$

Thus we have

$$z = P(X\theta - Y)$$

except to enforce the constraint that $z_j \geq 0$, we instead have

$$z = P|X\theta - Y|.$$

Intuitively, z represents how far we're off for each example, weighted by the $\sqrt{w^{(i)}}$'s.

(b) [4 points] Show that we can construct W such that we have

$$J(\theta) = (X\theta - Y)^\top W(X\theta - Y) = \sum_{i=1}^{m} |\theta^\top x^{(i)} - y^{(i)}|.$$

Make sure to clearly define W.

Answer: Again, for

$$W = \mathrm{diag}([w^{(1)}, w^{(2)}, \ldots, w^{(m)}]),$$

we have that

$$(X\theta - Y)^\top W(X\theta - Y) = \sum_{j=1}^{m} w^{(i)}(\theta^\top x^{(i)} - y^{(i)})^2.$$

Now, we want $w^{(i)}(\theta^\top x^{(i)} - y^{(i)})^2 = |\theta^\top x^{(i)} - y^{(i)}|$, which is achieved by simply setting

$$w^{(i)} = 1/|\theta^\top x^{(i)} - y^{(i)}|.$$

We accepted this answer without considering the case where the denominator is 0.

Remark: Using the sum of the absolute differences instead of the squared differences is one method for reducing the effect of outliers when estimating θ. Admittedly this is a roundabout way of doing this, but there are additional modifications we can make to W which we do not consider here.

Part 2 Thus far we have considered scalar-valued targets; however, it may sometimes be the case that we wish to regress vector-valued $y^{(i)} \in \mathbb{R}^k$ using $\theta \in \mathbb{R}^{n \times k}$ (hence $X \in \mathbb{R}^{m \times n}$, $Y \in \mathbb{R}^{m \times k}$). We'll use θ_j to denote the jth *column* of θ (this will simplify vectorization later). In this part of the question, we'll make some probabilistic assumptions in our model. As always, we'll assume our training examples $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$ are independently and identically distributed, and we'll give every example equal weight. We'll make another very strong assumption that the dimensions of the target are independent from one another and let

$$p(y|x;\theta) = \prod_{j=1}^{k} p(y_j|x;\theta_j)$$

$$\text{where } y_j|x \sim \mathcal{N}(\theta_j^\top x, \sigma_j^2)$$

for some σ_j, $j = 1, \ldots, k$ that are known and fixed beforehand.

(c) [4 points] Suppose instead of maximizing the likelihood, we instead minimize squared error, that is

$$J(\theta) = \sum_{i=1}^{m} \sum_{j=1}^{k} (\theta_j^\top x^{(i)} - y_j^{(i)})^2.$$

i. [1 points] By simply minimizing squared error, what assumption are we making which violates the assumptions given in our problem definition?

ii. [3 points] Despite this poor assumption, let's consider minimizing squared error regardless. Using the parameters learned by minimizing the squared error, what is the expected loss $\mathbb{E}[J_{\text{sq}}(\theta)]$ under the probabilistic assumptions given in the question? Do not assume the σ_j are the same for every j.

Answer: By simply minimizing squared error, we make the assumption that the error term $\epsilon_j^{(i)} = \theta_j^\top x^{(i)} - y_j^{(i)}$ for each dimension j has the same variance under the probabilistic assumptions we've defined, violating the assumption of different σ_j given in our problem definition. The expected loss is then

$$\begin{aligned}
\mathbb{E}[J(\theta)] &= \mathbb{E}\left[\sum_{i=1}^{m} \sum_{j=1}^{k} (\epsilon_j^{(i)})^2\right] \\
&= m \sum_{j=1}^{k} \mathbb{E}[(\epsilon_j^{(i)})^2] \\
&= m \sum_{j=1}^{k} \mathbb{E}[(\epsilon_j^{(i)})^2] - \mathbb{E}[\epsilon_j^{(i)}]^2 \\
&= m \sum_{j=1}^{k} \mathrm{Var}(\epsilon_j^{(i)}) \\
&= m \sum_{j=1}^{k} \sigma_j^2.
\end{aligned}$$

(d) [5 points] Now, let's consider maximizing the likelihood.

i. [3 points] Show that we can define some z such that we can write $J_{\text{ll}}(\theta) = z^\top z$, where the θ that minimizes $J_{\text{ll}}(\theta)$ also maximizes the likelihood. Your answer should explicitly give an expression for z in terms of the other problem variables. You may wish to refer to results proven in the first lecture notes for this part of the question.

Answer: Recall from the first lecture notes that in the case where $y^{(i)}$ is a scalar, maximizing $\ell(\theta)$ gives the same θ as minimizing the loss

$$J(\theta) = \frac{1}{\sigma^2}\sum_{i=1}^{m}(\theta^\top x^{(i)} - y^{(i)})^2$$

where $\epsilon^{(i)} = \theta^\top x^{(i)} - y^{(i)} \sim \mathcal{N}(0, \sigma^2)$. Since in this problem each dimension of y has a different variance σ_j^2, we instead have

$$\begin{aligned} J(\theta) &= \sum_{i=1}^{m}\sum_{j=1}^{k}\frac{1}{\sigma_j^2}(\theta_j^\top x^{(i)} - y_j^{(i)})^2 \\ &= \sum_{j=1}^{k}\frac{1}{\sigma_j^2}\sum_{i=1}^{m}(\theta_j^\top x^{(i)} - y_j^{(i)})^2. \end{aligned}$$

Also note that in this case, $(X\theta - Y)^\top(X\theta - Y) \in \mathbb{R}^{k\times k}$. This we must have $z = (X\theta - Y)L$, where $L \in \mathbb{R}^{k\times 1} = [1/\sigma_1, 1/\sigma_2, \ldots, 1/\sigma_k]^\top$. You should verify this vectorization results in the correct $J(\theta)$.

ii. [2 points] Using the parameters learned by maximizing the log-likelihood, what is the expected loss $\mathbb{E}[J_{\text{ll}}(\theta)]$ under the probabilistic assumptions given in the question?

Answer: The solution is similar to the previous part, except now we're scaling each $(\epsilon_j^{(i)})^2$ in the sum by $1/\sigma_j^2$ when computing $J(\theta)$. Thus,

$$\begin{aligned}\mathbb{E}[J(\theta) &= m\sum_{j=1}^{k}\frac{1}{\sigma_j^2}\mathbb{E}[(\epsilon_j^{(i)})^2] \\ &= m\sum_{j=1}^{k}\frac{1}{\sigma_j^2}(\sigma_j^2) \\ &= mk\end{aligned}$$

3. **[12 points] Generalized discriminant analysis**

In this question, we will show that whole families of generative models have posterior distributions that can be written in the form of the logistic function. Then, we'll have another look at Gaussian discriminant analysis.

Suppose we have a two class problem with $y \sim \text{Bernoulli}(\phi)$ and $p(y = 1; \phi) = \phi$, $p(y = 0; \phi) = 1-\phi$. Further, suppose the conditional probability distribution $p(x|y; \eta)$ falls in the exponential family, i.e.

$$p(x|y = k; \eta_k) = b(x) \exp(\eta_k^\top T(x) - a(\eta_k))$$

with $T(x) = x$ and $k \in \{0, 1\}$ denotes the class label.

(a) [5 points] Show that we can express the posterior distribution as

$$p(y|x; \eta_0, \eta_1, \phi) = \frac{1}{1 + \exp(\theta_0 + \theta_1^\top x)}$$

for some θ_0 and θ_1. Make sure to clearly define θ_0 and θ_1 in terms of the other problem variables.

Answer: Let's consider $p(y = 1|x; \phi, \eta_0, \eta_1)$. By Bayes' rule, we have

$$\begin{aligned} p(y = 1|x; \phi, \eta_0, \eta_1) &= \frac{\phi p(x|y = 1; \eta_0, \eta_1)}{(1 - \phi)p(x|y = 0; \eta_0) + \phi p(x|y = 1; \eta_1)} \\ &= \frac{\phi b(x) \exp(\eta_1^\top x - a(\eta_1))}{(1 - \phi)b(x) \exp(\eta_0^\top x - a(\eta_0)) + \phi b(x) \exp(\eta_1^\top x - a(\eta_1))} \\ &= \frac{1}{1 + \frac{(1-\phi)}{\phi} \exp([\eta_0 - \eta_1]^\top x - a(\eta_0) + a(\eta_1))} \\ &= \frac{1}{1 + \exp([\eta_0 - \eta_1]^\top x - a(\eta_0) + a(\eta_1) + \log(1 - \phi) - \log(\phi))} \end{aligned}$$

Considering the case with $y = 0$, we find that we can write both cases as

$$p(y|x; \phi, \eta_0, \eta_1) = \frac{1}{1 + \exp(c([\eta_0 - \eta_1]^\top x - a(\eta_0) + a(\eta_1) + \log(1 - \phi) - \log(\phi)))}$$

where $c = 2y - 1$. Which is in the expected form with

$$\begin{aligned} \theta_1 &= c(\eta_0 - \eta_1) \\ \theta_0 &= c(-a(\eta_0) + a(\eta_1) + \log(1 - \phi) - \log(\phi)) \end{aligned}$$

(b) [4 points] For the first part of this question, we assumed $T(x) = x$. For the Gaussian distribution, it turns out that:

$$\begin{aligned} \eta &= \left[\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}\right]^\top \\ T(x) &= [x, x^2]^\top \\ b(x) &= 1/\sqrt{2\pi} \end{aligned}$$

(We do not provide the expression for $a(\eta)$ to make the solution simpler.)

Consider the univariate Gaussian discriminant analysis problem where

$$p(x|y = k, \mu_k, \sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp(-(x - \mu_k)^2/(2\sigma_k^2))$$

for $k \in \{0, 1\}$. Suppose we have $\sigma_0 \neq \sigma_1$ and let $\phi = 1 - \phi = 1/2$. What is the form of the equation for the decision boundary? Justify your answer (by adapting your derivation from the first of this question) by finding an expression for the decision boundary in terms of x, μ_0, μ_1, σ_0, σ_1, $a(\eta_0)$, and $a(\eta_1)$.

Answer: To find the decision boundary, we simply set $p(y = 1|x; \eta_0, \eta_1) = 1/2$ and solve. From our previous derivation (where we replace x with $T(x)$), we must have $(\eta_0 - \eta_1)^\top T(x) - a(\eta_0) + a(\eta_1) = 0$. Note that the ϕ terms disappear since $\phi = 1/2$. Substituting in for η and $T(x)$, we have

$$\left(-\frac{1}{2\sigma_0^2} + \frac{1}{2\sigma_1^2}\right) x^2 + \left(\frac{\mu_0}{\sigma_0^2} - \frac{\mu_1}{\sigma_1^2}\right) x - a(\eta_0) + a(\eta_1) = 0$$

which is a quadratic.

(c) [3 points] Now, assume $\sigma_0 = \sigma_1 = \sigma$. With the univariate Gaussian discriminant analysis model, we can compute the joint probability $p(x, y; \phi, \mu_0, \mu_1, \sigma)$ as well the conditional probability $p(y|x; \phi, \mu_0, \mu_1)$ that follows the form of the logistic function.

Given a logistic regression model $p(y|x; \theta)$, however, can we compute a unique maximum likelihood ϕ, μ_0, μ_1, and σ from θ? If yes, give the expressions for ϕ, μ_0, μ_1, and σ in terms of θ. If not, provide a counter-example.

Answer: No; consider the two cases (1) $\mu_0 = -1$, $\mu_1 = 1$ and (2) $\mu_0 = -2$, $\mu_1 = 2$ which both have the same decision boundary.

4. **[7 points] Kernels on discrete sequences**

Thus far, we have dealt with kernel feature mappings that map real-valued inputs to some feature space. As we will see in this question, it is also possible to define feature mappings on discrete objects such as sequences of tokens. This is especially useful in applications such as bioinformatics and natural language processing.

Let S denote the set of strings (sequences) formed by concatenating elements of some alphabet V. For example, V might be the set $\{a, b, \ldots, z, \text{[space]}\}$ in the case of the English alphabet, in which case S contains all strings obtained by concatenating these characters (e.g. "the", "cat", "the cat", etc.). For any s_1, $s_2 \in S$, we define

$$\delta(s_1, s_2) = \begin{cases} 1, & \text{if } s_1 = s_2 \text{ (i.e. the two strings are the same)} \\ 0, & \text{otherwise.} \end{cases}$$

(a) [4 points] For any sequence $s \in S$, let $g_n(s)$ denote the set of all length n subsequences of s. For example,

$$g_2(abcd) = \{ab, bc, cd\}.$$

We define the kernel function:

$$K_g(s_1, s_2) = \sum_{x \in g_n(s_1)} \sum_{z \in g_n(s_2)} \delta(x, z).$$

Express the kernel function using an appropriate feature mapping $\phi(s)$. That is, define the feature mapping $\phi(s) \in \mathbb{R}^d$ such that $K_g(s_1, s_2) = \phi(s_1)^\top \phi(s_2)$. What is the dimension of the feature mapping space d?

Hint: The description of $\phi(s)$ should be brief.

Answer: Define $\phi(s)$ as follows:

$$\phi(s)_i = \begin{cases} 1 & \text{if } x_i \in g_n(s) \\ 0 & \text{otherwise} \end{cases}$$

where $\phi(s_1)^\top \phi(s_2)$ will be equal to the number of overlapping n-grams in s_1 and s_2. Let $X_n = \{x_1, x_2, \ldots, x_d\}$ denote the set of all length n strings in S. For finite n, X_n must also be finite and of size $|V|^n$, where $|V|$ denotes the size of the alphabet. The dimension of the feature mapping space d is $|X_n| = |V|^n$. Note that we can quickly compute overlapping n-grams in s_1 and s_2 assuming s_1 and s_2 are fairly short without ever computing $\phi(s_1)$ or $\phi(s_2)$.

(b) [3 points] Let $w(s)$ denote the set of words in s (substrings in s separated by the [space] character). As an example,

$$w(\text{winter is coming}) = \{\text{winter, is, coming}\}.$$

We define a new kernel function $K_w(s_1, s_2)$ as:

$$K_w(s_1, s_2) = \sum_{x \in w(s_1)} \sum_{z \in w(s_2)} \delta(x, z).$$

Express this function using an appropriate feature mapping ψ. That is, as in the previous part of this question, your task is to define the feature mapping $\psi(s)$ such that $K_w(s_1, s_2) = \psi(s_1)^\top \psi(s_2)$. What is the dimension of the feature mapping space in this case?

Answer: Now we define X to be the set of all possible words, and we can define $\phi(s)$ similarly as in the previous part. Note that $\phi(x)$ is now infinite-dimensional.

[This question is closely adapted from one of the assignments from CS4780 taught by Kilian Weinberger.]

5. **[25 points] Expectation maximization for the exponential family**

In class we studied the EM algorithm in the general setting, where the model $p(x, z; \theta)$ was any probability distribution. In this problem we study a more restricted setting, where the joint probability model (over observed and latent variables) belongs to the exponential family. This allows for simpler a E-step and M-step than the general setting, which will explore in this question.

Suppose we are given training data $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$ consisting of m independent examples, and each observed variable $x^{(i)} \in \mathbb{R}^k$ has a corresponding unobserved (hidden) latent variable $z^{(i)} \in \mathbb{R}^l$. Let us denote the complete data $y \in \mathbb{R}^{k+l}$ such that $y^{(i)} = (x^{(i)}, z^{(i)})$. The full model, including both observed data and unobserved latent variables, is in the exponential family, and has the following form:

$$p(y; \eta) = b(y) \exp\left(\eta^\top y - a(\eta)\right) \qquad (1)$$

where $y = (x, z)$, $\eta \in \mathbb{R}^{k+l}$ is the *natural parameter* of the exponential family, b is called the *base measure*, and a is called the *log-partition function*. For simplicity, we assume the sufficient statistic to be the identity function, i.e. $T(y) = y$. We denote $\eta = (\eta_x, \eta_z)$, where $\eta_x \in \mathbb{R}^k$ and $\eta_z \in \mathbb{R}^l$ such that

$$\eta^\top y = \eta_x^\top x + \eta_z^\top z.$$

For simplicity, we also assume z to be an element of a finite set Z, i.e $z \in Z$, $Z = \{Z_0, ..., Z_N\}$ where each $Z_j \in \mathbb{R}^l$.

Our goal is to estimate η, given only the observed data $\{x^{(1)}, ..., x^{(m)}\}$.

In this setting, the EM algorithm begins with a random initialization of $\eta = \eta^{(0)}$, and repeatedly iterates between the E-step and the M-step (where we denote the iteration number as t) until convergence:

- (E-step) For each i, set $z^{(i)}$ to be the expectation under the posterior $p(z|x; \eta^{(t)})$:

$$\begin{aligned} \hat{z}^{(i)} &:= \mathbb{E}\left[z^{(i)} \mid x^{(i)}; \eta^{(t)}\right] \\ \hat{y}^{(i)} &:= (x^{(i)}, \hat{z}^{(i)}) \end{aligned}$$

- (M-step) Update $\eta^{(t+1)}$ to be the maximum likelihood estimate of the "filled-in" data $\{\hat{y}^{(1)}, \hat{y}^{(2)}, ..., \hat{y}^{(m)}\}$ under the model $p(y; \eta)$.

$$\eta^{(t+1)} = \arg\max_\eta \sum_i \log p(\hat{y}^{(i)}; \eta) \qquad (2)$$

In the rest of this problem, we will step through the derivation of the closed-form update rules for both the E-step and M-step.

(a) [8 points] First, let's justify why the procedure described above is equivalent to the EM algorithm seen in class. Recall that in the general setting, EM algorithm estimates the next $\eta^{(t+1)}$ with the following update rule:

$$\eta^{(t+1)} = \arg\max_{\eta} \sum_i \sum_{z^{(i)}} Q(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \eta)}{Q(z^{(i)})} \tag{3}$$

where we chose $Q(z^{(i)})$ to be

$$Q(z^{(i)}) = p(z^{(i)} | x^{(i)}; \eta^{(t)})$$

Now, if $p(x, z; \eta)$ belongs to the exponential family and takes the form of Equation (1), show that Equations (2) and (3) are equivalent by proving:

$$\arg\max_{\eta} \sum_i \log p(x^{(i)}, \hat{z}^{(i)}; \eta) = \arg\max_{\eta} \sum_i \sum_{z^{(i)}} Q(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \eta)}{Q(z^{(i)})}$$

If you get stuck on this part of the question, you can solve the other parts without Part (a).

Answer: We'll start with the right-hand side and show that it's equivalent to the left-hand side.

$$\begin{aligned}
\text{RHS} &= \arg\max_{\eta} \sum_i \sum_{z^{(i)}} Q(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \eta)}{Q(z^{(i)})} \\
&= \arg\max_{\eta} \sum_i \mathbb{E}_{z^{(i)}} \left[\log \frac{p(x^{(i)}, z^{(i)}; \eta)}{Q(z^{(i)})} \mid x^{(i)}; \eta^{(t)} \right] \\
&= \arg\max_{\eta} \sum_i \left(\mathbb{E}_{z^{(i)}} \left[\log p(x^{(i)}, z^{(i)}; \eta) \mid x^{(i)}; \eta^{(t)} \right] - \mathbb{E}_{z^{(i)}} \left[\log Q(z^{(i)}) \mid x^{(i)}; \eta^{(t)} \right] \right) \\
&= \arg\max_{\eta} \sum_i \mathbb{E}_{z^{(i)}} \left[\log p(x^{(i)}, z^{(i)}; \eta) \mid x^{(i)}; \eta^{(t)} \right] \text{ [Second term was constant w.r.t. } \eta\text{].} \\
&= \arg\max_{\eta} \sum_i \mathbb{E}_{z^{(i)}} \left[\log b(x^{(i)}, z^{(i)}) + (\eta_x, \eta_z)^\top (x^{(i)}, z^{(i)}) - a(\eta) \mid x^{(i)}; \eta^{(t)} \right] \text{ [Using (1).]} \\
&= \arg\max_{\eta} \sum_i \mathbb{E}_{z^{(i)}} \left[\log b(x^{(i)}, z^{(i)}) \mid x^{(i)}; \eta^{(t)} \right] + \mathbb{E}_{z^{(i)}} \left[(\eta_x, \eta_z)^\top (x^{(i)}, z^{(i)}) \mid x^{(i)}; \eta^{(t)} \right] - a(\eta) \\
&= \arg\max_{\eta} \sum_i \log b(x^{(i)}, \hat{z}^{(i)}) + \mathbb{E}_{z^{(i)}} \left[(\eta_x, \eta_z)^\top (x^{(i)}, z^{(i)}) \mid x^{(i)}; \eta^{(t)} \right] - a(\eta)
\end{aligned}$$

[Since $\mathbb{E}_{z^{(i)}} \left[\log b(x^{(i)}, z^{(i)}) \mid x^{(i)}; \eta^{(t)} \right]$ and $\log b(x^{(i)}, \hat{z}^{(i)})$ are both constants w.r.t. η.]

$$\begin{aligned}
&= \arg\max_{\eta} \sum_i \log \left(b(x^{(i)}, \hat{z}^{(i)}) \exp((\eta_x, \eta_z)^\top (x^{(i)}, \hat{z}^{(i)}) - a(\eta)) \right) \\
&= \arg\max_{\eta} \sum_i \log p(x^{(i)}, \hat{z}^{(i)}; \eta) \\
&= \text{LHS}
\end{aligned}$$

Additional space for Part (a)

(b) [6 points] Now, show that the posterior distribution $p(z|x;\eta)$ also belongs to exponential family. Specifically, show that it can be represented in the form:

$$p(z|x;\eta) = b_x(z)\exp(\eta_z^\top z - a_x(\eta_z))$$

Clearly state what $b_x(z)$ and $a_x(\eta_z)$ are. Your final expressions for $b_x(z)$ and $a_x(\eta_z)$ should **not** contain integrals ($\int \cdot$), expectations ($\mathbb{E}[\cdot]$), or max/argmax terms. They can, however, include finite summation ($\sum \cdot$) terms.

Answer: Using the definition of conditional expectations, we have

$$\begin{aligned}
p(z|x;\eta) &= \frac{p(x,z;\eta)}{\sum_{z'\in Z} p(x,z';\eta)} \\
&= \frac{b(x,z)\exp((\eta_x,\eta_z)^\top (x,z) - a(\eta))}{\sum_{z'\in Z} b(x,z')\exp((\eta_x,\eta_z)^\top (x,z') - a(\eta))} \\
&= \frac{b(x,z)\exp((\eta_x,\eta_z)^\top (x,z))}{\sum_{z'\in Z} b(x,z')\exp((\eta_x,\eta_z)^\top (x,z'))} \\
&= \frac{b(x,z)\exp(\eta_x^\top x + \eta_z^\top z)}{\sum_{z'\in Z} b(x,z')\exp(\eta_x^\top x + \eta_z^\top z')} \\
&= \frac{b(x,z)\exp(\eta_z^\top z)}{\sum_{z'\in Z} b(x,z')\exp(\eta_z^\top z')} \\
&= b(x,z)\exp\left(\eta_z^\top z - \log\sum_{z'\in Z} b(x,z')\exp(\eta_z^\top z')\right) \\
&= b_x(z)\exp(\eta_z^\top z - a_x(\eta_z)) \\
\text{where } b_x(z) &= b(x,z) \\
\text{and } a_x(\eta_z) &= \log\sum_{z'\in Z} b(x,z')\exp(\eta_z^\top z').
\end{aligned}$$

(c) [3 points] Find a closed-form expression for the expectation of the hidden variable z under the posterior $p(z|x;\eta_z)$. That is, find:

$$\mathbb{E}_z[z|x;\eta] = \ldots$$

As before, the final form cannot have integrals ($\int \cdot$), expectations ($\mathbb{E}[\cdot]$), or max/argmax terms. It can have finite summation ($\sum \cdot$) terms. This completes the E-step.

Hint: You can use the fact that the expectation of a random variable in the exponential family is equal to the gradient of its log-partition function. This is shown in Problem Set 1 Question 2(d) (Poisson regression).

Answer: We calculated the log-partition of $p(z|x;\eta)$ in the previous section as

$$a_x(\eta_z) = \log \sum_{z' \in Z} b(x, z') \exp(\eta_z^\top z')$$

Using the hint, we have

$$\begin{aligned}
\mathbb{E}_z[z|x;\eta] &= \frac{\partial a}{\partial \eta_z}(\eta_z) \\
&= \frac{1}{\sum_{z' \in Z} b(x, z') \exp(\eta_z^\top z')} \sum_{z' \in Z} b(x, z') \exp(\eta_z^\top z') z'
\end{aligned}$$

(d) [2 points] For any probability distribution in the exponential family taking the form of Equation (1), find the relationship between the canonical link function and the log-partition function.

Answer: The gradient of the log-partition maps the natural parameter to the mean of the variable. (Hence the gradient of the log-partition is the same as the canonical response function, denoted g.)

The canonical link function maps the mean of the variable to the natural parameter. By convention the canonical link function is denoted by g^{-1}.

Thus, the canonical link function and the gradient of the log-partition function are the inverses of each other.

$$\begin{aligned} \frac{\partial a}{\partial \eta}(\eta) &= g(\eta) = \mathbb{E}[y] \\ g^{-1}(\mathbb{E}[y]) &= \eta \end{aligned}$$

(e) [6 points] Let us assume the canonical link function exists for $p(x, z; \eta)$, and denoted as g^{-1}. Find a closed-form expression for finding the maximum likelihood estimate of the natural parameter of a model in the exponential family in the form of Equation (1). Extend this to obtain a closed-form solution for solving the M-step given in Equation (2). This completes the M-step.

Hint: g^{-1} should appear in your closed-form solution for η.

Answer: The total log-likelihood of the complete data, $\ell(\eta; y)$ is

$$\begin{aligned}
\ell(\eta; y) &= \log \prod_{i=1}^{m} p(y^{(i)}; \eta) \\
&= \log \prod_{i=1}^{m} b(y^{(i)}) \exp(\eta^\top y^{(i)} - a(\eta)) \\
&= \sum_{i=1}^{m} \left(\log b(y^{(i)}) + \eta^\top y^{(i)} - a(\eta)\right) \\
&= \left(\sum_{i=1}^{m} \log b(y^{(i)})\right) + \eta^\top \left(\sum_{i=1}^{m} y^{(i)}\right) - ma(\eta)
\end{aligned}$$

For maximum-likelihood, the gradient of the log-likelihood will be 0.

$$\begin{aligned}
0 &= \frac{\partial \ell}{\partial \eta}(\eta) \\
&= \frac{\partial}{\partial \eta}\left[\left(\sum_{i=1}^{m} \log b(y^{(i)})\right) + \eta^\top \left(\sum_{i=1}^{m} y^{(i)}\right) - ma(\eta)\right] \\
&= \left(\sum_{i=1}^{m} y^{(i)}\right) - m\frac{\partial}{\partial \eta} a(\eta) \\
\Rightarrow \frac{\partial}{\partial \eta} a(\eta) &= \frac{1}{m}\sum_{i=1}^{m} y^{(i)} \\
\eta &= g^{-1}\left(\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\right)
\end{aligned}$$

This leads to the closed-form M-step:

$$\eta^{(t+1)} = g^{-1}\left(\frac{1}{m}\sum_{i=1}^{m} \hat{y}^{(i)}\right)$$

6. **[14 points] Neural networks with shortcut connections**

In this problem, we'll perform classification using a modified two-layer neural network. For any input vector $x \in \mathbb{R}^n$, our neural network outputs a probability distribution over K classes following the forward propagation rules:

$$\begin{aligned} z^{[1]} &= W^{[1]}x + b^{[1]} \\ a^{[1]} &= \sigma(z^{[1]}) \\ z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\ \hat{y} &= \text{softmax}(z^{[2]}) \end{aligned}$$

where $W^{[1]} \in \mathbb{R}^{p\times n}$, $b^{[1]} \in \mathbb{R}^p$, $W^{[2]} \in \mathbb{R}^{K\times p}$, $b^{[2]} \in \mathbb{R}^K$, and for any vector z, $\text{softmax}(z) = \exp(z)/\sum_k \exp(z_k)$ (where the division is performed elementwise).

We evaluate our model using the cross entropy loss (CE). For a single example (x, y), the cross entropy loss is:

$$\text{CE}(y, \hat{y}) = -\sum_{k=1}^{K} y_k \log \hat{y_k},$$

where $\hat{y} \in \mathbb{R}^K$ is the vector of softmax outputs for a single training example x, and $y \in \mathbb{R}^K$ is the ground-truth vector for training example x such that $y = [0, ..., 0, 1, 0, ..., 0]^\top$ contains a single 1 at the position of the correct class. For m training examples, we average the cross entropy loss over the m examples:

$$J(W^{[1]}, W^{[2]}, b^{[1]}, b^{[2]}) = \frac{1}{m}\sum_{i=1}^{m} \text{CE}(y, \hat{y}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k \log \hat{y_k}.$$

We modify the described network by adding a "shortcut" connection between the input x and the second layer. The forward propagation equations then become:

$$\begin{aligned} z^{[1]} &= W^{[1]}x + b^{[1]} \\ a^{[1]} &= \sigma(z^{[1]}) \\ z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} + Wx \\ \hat{y} &= \text{softmax}(z^{[2]}) \end{aligned}$$

where $W \in \mathbb{R}^{K\times n}$, and $J(W^{[1]}, W^{[2]}, b^{[1]}, b^{[2]}, W)$ is defined as before.

Figure 1 (on the next page) shows the two-layer neural network, before and after adding the shortcut connection. In practice, it is often observed that shortcut connections improve the learning of neural networks.

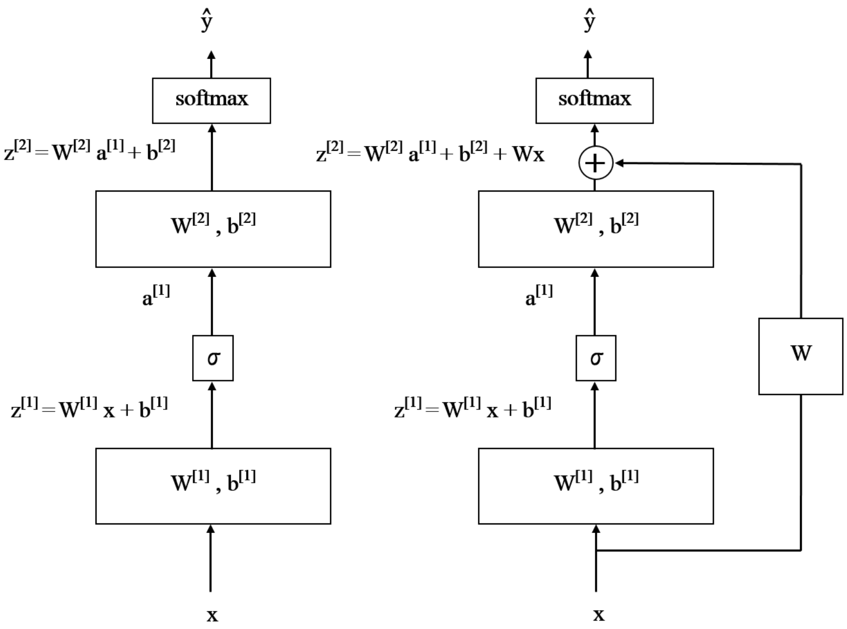

Figure 1: On the left, a two-layer neural network without shortcut connection. On the right, the same two-layer neural network with a shortcut connection.

(a) [2 points] How many parameters does the model including the shortcut connection have? Your answer should be expressed in terms of n, p, and K.

Answer: $W^{[1]}$: $p \times n$, $b^{[1]}$: p, $W^{[2]}$: $K \times p$, $b^{[2]}$: K, W: $K \times n$.
Total number of parameters: $p(n+1) + K(p+n+1)$.

(b) [4 points] In this part of the question, we'll consider a single input vector $x \in \mathbb{R}^n$ with true label vector y. Show that: $\nabla_{z^{[2]}}\mathrm{CE}(y, \hat{y}) = \nabla_{z^{[2]}}\mathrm{CE}(y, \mathrm{softmax}(z^{[2]})) = \hat{y} - y$. *Hint*: To simplify your answer, it might be convenient to denote the true label of x as $l \in \{1, \dots, K\}$. Hence l is the index such that that $y = [0, ..., 0, 1, 0, ..., 0]^\top$ contains a single 1 at the l-th position. You may also wish to compute $\partial\mathrm{CE}(y, \hat{y})/\partial z_j^{[2]}$ for $j \neq l$ and $j = l$ separately.

If you get stuck, the next part of this question can be done independently of this part.

Answer: First, let's expand the expression $\mathrm{CE}(y, \hat{y})$.

$$\begin{aligned}
\mathrm{CE}(y, \hat{y}) &= -\sum_{k=1}^{K} y_k \log \hat{y}_k = -y_l \log \hat{y}_l = -\log \hat{y}_l \\
&= -\log(\mathrm{softmax}(z^{[2]})_l) \\
&= -\log(\mathrm{softmax}(z_l^{[2]})) \\
&= -\log\left[\frac{\exp(z_l^{[2]})}{\sum_{k=1}^{K}\exp(z_k^{[2]})}\right] \\
&= -z_l^{[2]} + \log\sum_{k=1}^{K}\exp(z_k^{[2]})
\end{aligned}$$

To determine the gradient, first let's consider $\dfrac{\partial\mathrm{CE}(y, \hat{y})}{\partial z_l^{[2]}}$:

$$\begin{aligned}
\frac{\partial\mathrm{CE}(y, \hat{y})}{\partial z_l^{[2]}} &= \frac{\partial}{\partial z_l^{[2]}}(-z_l^{[2]} + \log(\sum_{k=1}^{K}\exp(z_k^{[2]}))) = -1 + \frac{\exp(z_l^{[2]})}{\sum_{k=1}^{K}\exp(z_k^{[2]})} \\
&= -1 + \mathrm{softmax}(z^{[2]})_l \\
&= -y_l + \hat{y}_l.
\end{aligned}$$

Now let's consider $\dfrac{\partial\mathrm{CE}(y, \hat{y})}{\partial z_j^{[2]}}$ for $j \neq l$:

$$\begin{aligned}
\frac{\partial\mathrm{CE}(y, \hat{y})}{\partial z_j^{[2]}} &= \frac{\partial}{\partial z_j^{[2]}}(-z_l^{[2]} + \log(\sum_{k=1}^{K}\exp(z_k^{[2]}))) = \frac{\exp(z_j^{[2]})}{\sum_{k=1}^{K}\exp(z_k^{[2]})} \\
&= \mathrm{softmax}(z^{[2]})_j \\
&= -y_j + \hat{y}_j
\end{aligned}$$

Combining the two cases, we have $\nabla_{z^{[2]}}\mathrm{CE}(y, \hat{y}) = \hat{y} - y$ as desired.

(c) [7 points] Find the expressions for $\nabla_{W^{[2]}}J$, $\nabla_{W^{[1]}}J$, $\nabla_{b^{[1]}}J$, and $\nabla_x J$ for a single training example (x, y). We've already provided $\nabla_{b^{[2]}}J$ for you. You may assume the result from Part (b) is true for this part of the question.

Using the result from Part (b), we have: $\delta_{z^{[2]}} = \nabla_{z^{[2]}}\text{CE}(y, \hat{y}) = \hat{y} - y$.

$$\begin{aligned}\nabla_{b^{[2]}}\text{CE}(y, \hat{y}) &= \delta_{z^{[2]}} \circ \nabla_{b^{[2]}} z^{[2]} \\ &= (\hat{y} - y) \circ \nabla_{b^{[2]}}(W^{[2]}a^{[1]} + b^{[2]} + Wx) \\ &= \hat{y} - y\end{aligned}$$

Answer:

$$\begin{aligned}\nabla_{W^{[2]}}\text{CE}(y, \hat{y}) &= \delta_{z^{[2]}} \circ \nabla_{W^{[2]}} z^{[2]} \\ &= (\hat{y} - y) \circ \nabla_{W^{[2]}}(W^{[2]}a^{[1]} + b^{[2]} + Wx) \\ &= (\hat{y} - y)a^{[1]\top}\end{aligned}$$

$$\begin{aligned}\nabla_{b^{[1]}}\text{CE}(y, \hat{y}) &= \delta_{z^{[2]}} \circ \nabla_{a^{[1]}} z^{[2]} \circ \nabla_{b^{[1]}} a^{[1]} \\ &= W^{[2]\top}(\hat{y} - y) \circ \nabla_{b^{[1]}} a^{[1]} \\ &= W^{[2]\top}(\hat{y} - y) \circ a^{[1]} \circ (1 - a^{[1]})\end{aligned}$$

$$\begin{aligned}\nabla_{W^{[1]}}\text{CE}(y, \hat{y}) &= \delta_{z^{[2]}} \circ \nabla_{a^{[1]}} z^{[2]} \circ \nabla_{W^{[1]}} a^{[1]} \\ &= W^{[2]\top}(\hat{y} - y) \circ \nabla_{W^{[1]}} a^{[1]} \\ &= \left[W^{[2]\top}(\hat{y} - y) \circ a^{[1]} \circ (1 - a^{[1]})\right] x\end{aligned}$$

$$\begin{aligned}\nabla_x\text{CE}(y, \hat{y}) &= \delta_{z^{[2]}} \circ \nabla_{a^{[1]}} z^{[2]} \circ \nabla_x a^{[1]} + \delta_{z^{[2]}} \circ \nabla_x (Wx) \\ &= W^{[2]\top}(\hat{y} - y) \circ \nabla_x a^{[1]} + W^\top \delta_{z^{[2]}} \\ &= W^{[1]\top}\left[W^{[2]\top}(\hat{y} - y) \circ a^{[1]} \circ (1 - a^{[1]})\right] + W^\top(\hat{y} - y)\end{aligned}$$

(d) [1 points] Give the (basic) gradient descent update rules for the model parameters (with step size α) for a single training example.

Answer: For every W ($W^{[1]}$, $W^{[2]}$, W) and b ($b^{[1]}$, $b^{[2]}$), we have the update

$$W := W - \alpha \nabla_W J$$
$$b := b - \alpha \nabla_b J$$

That's all! Congratulations on completing the midterm exam!