Answer the questions in the answer spaces provided on the question sheets. If you run out of room for an answer, note in the answer space that it is continued on another page, and continue on a blank sheet that you staple to the end of the exam.

No use of a computing device is allowed other than as a timer, clock, music player, or simple (non-programmable) scientific calculator (for example, https://www.calculator.net/scientific-calculator.html). One double-sided 8.5x11 page of handwritten notes is allowed.

If you think something about a question is open to interpretation, write any assumptions you’ve made as part of answering the question.

Be concise in your answers; you need not try to fill in all or even most of the lines provided for an answer.

You have two contiguous hours to complete this exam starting from when you unstaple the exam or look at any page other than the first.

Bring the completed exam to class on October 17.
1. You are trying to optimize a model of the form: \( f(x) = a_2 x^2 + a_1 x + a_0 \). Your loss function is: \( L(y, \hat{y}) = (y - \hat{y})^2 \). Your are using the standard optimization function modified to add weight decay (where the weight decays to 0.9 of its original value each timestep before adjusting by the gradient). The learning rate, \( \lambda \), is .01. The current values of the coefficients are \( a_2 = 1, a_1 = -3, a_0 = -3 \).

Given the training instance \((x, y) = (2, -7)\), what are the new values of the coefficients after one execution of the optimization function? Show your work.

**Solution:** First, let’s calculate \( \hat{y} \) and the loss:

\[
\hat{y} = a_2 x^2 + a_1 x + a_0 = 1 \times 4 + -3 \times 2 + -3 = -5
\]

\[
L(y, \hat{y}) = (y - \hat{y})^2 = (-7 - -5)^2 = 4
\]

Now, we need to calculate the gradient of the loss function with respect to each of the coefficients:

\[
\frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y) = 2(2) = 4
\]

\[
\frac{\partial L}{\partial a_0} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_0} = 4 \times 1 = 4
\]

\[
\frac{\partial L}{\partial a_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_1} = 4x = 8
\]

\[
\frac{\partial L}{\partial a_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_2} = 4 \times x^2 = 16
\]

Now, we need to update the coefficients using weight decay:

\[
a_i = .9a_i - \lambda \frac{\partial L}{\partial a_i}
\]

\[
a_0 = .9a_0 - \lambda \frac{\partial L}{\partial a_0} = .9 \times -3 - .01 \times 4 = -2.7 - .04 = -2.74
\]

\[
a_1 = .9a_1 - \lambda \frac{\partial L}{\partial a_1} = .9 \times -3 - .01 \times 8 = -2.7 - .08 = -2.78
\]

\[
a_2 = .9a_2 - \lambda \frac{\partial L}{\partial a_2} = .9 \times 1 - .01 \times 16 = .9 - .16 = .74
\]

As a double-check, we can calculate the new loss and see that it went down:

\[
\hat{y} = a_2 x^2 + a_1 x + a_0 = .74 \times 4 + -2.78 \times 2 + -2.74 = -5.34
\]

\[
L(y, \hat{y}) = (y - \hat{y})^2 = (-7 - -5.34)^2 = 2.76
\]
2. Assume the following neural network:

\[
\begin{array}{cccc}
\text{Input} & \text{Hidden} & \text{Hidden} & \text{Output} \\
\text{layer} & \text{layer 1} & \text{layer 2} & \text{layer} \\
\end{array}
\]

You are given that:

\[
\begin{align*}
x &= \begin{bmatrix} 3 & 5 \end{bmatrix} \\
y &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\
W^{[2]} &= \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \\
b^{[2]} &= \begin{bmatrix} 1 \\ -18 \end{bmatrix} \\
W^{[3]} &= \begin{bmatrix} 2 & 3 \\ -3 & 2 \end{bmatrix} \\
b^{[3]} &= \begin{bmatrix} 2 \\ 5 \end{bmatrix} \\
g^{[1]} &= g^{[2]} = \text{relu} \\
g^{[3]} &= \sigma \\
a^{[1]} &= \begin{bmatrix} 0.5 & 5 \end{bmatrix} \\
z^{[2]} &= \begin{bmatrix} 0.5 & -2.5 \end{bmatrix} \\
a^{[2]} &= \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \\
z^{[3]} &= \begin{bmatrix} 0 \\ -0.5 \end{bmatrix} \\
\frac{\partial L}{\partial z^{[3]}} &= \begin{bmatrix} -0.5 \\ 0.38 \end{bmatrix}
\end{align*}
\]

(a) 2 points
Circle the line in the figure above that has the bolded weight 2 from the \( W^{[2]} \) array.

**Solution:** It’s the line running from \( z_2^{[1]}, a_2^{[1]} \) to \( z_1^{[2]}, a_1^{[2]} \).

(b) 7 points

What is the value of \( \frac{\partial L}{\partial W^{[3]}} \)?

**Solution:**

\[
\begin{align*}
\frac{\partial L}{\partial W^{[3]}} &= a^{[2]T} \frac{\partial L}{\partial z^{[3]}} \\
&= \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}^T \begin{bmatrix} -0.5 & 0.38 \end{bmatrix} \\
&= \begin{bmatrix} -0.25 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} -0.25 & .19 \\ 0 & 0 \end{bmatrix}
\end{align*}
\]
(c) What is the value of $\frac{\partial L}{\partial z[2]}$?

**Solution:**

\[
\frac{\partial L}{\partial a[2]} = \frac{\partial L}{\partial z[2]} (W^{[3]})^T \\
= \begin{bmatrix} -0.5 & 0.38 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ -3 & 2 \end{bmatrix}^T \\
= \begin{bmatrix} -0.5 & 0.38 \end{bmatrix} \begin{bmatrix} 2 \\ -3 \\ 3 \\ 2 \end{bmatrix} \\
= \begin{bmatrix} .14 & 2.26 \end{bmatrix}
\]

\[
\frac{\partial L}{\partial z[2]} = \frac{\partial L}{\partial a[2]} \odot \text{relu}'(z[2]) \\
= \begin{bmatrix} .14 & 2.26 \end{bmatrix} \odot [1 \ 0] \\
= \begin{bmatrix} .14 & 0 \end{bmatrix}
\]

(d) How many total parameters need to be learned in this model (that is, what is $|\Theta|$)?

**Solution:** The parameters are the $W^{[i]}$ arrays and the $b^{[i]}$ arrays. Total number of parameters is $3(2 \times 2 + 2 \times 1) = 18$.

(e) What are the dimensions of $\hat{y}$? **1x2**
3. (a) Data augmentation and noise injection are both examples of: **regularization techniques**.

(b) Give two examples of data augmentation, each in a different domain:

i. Data augmentation example

**Solution:** For an image classification problem, taking \((x, y)\) input pairs and augmenting by producing \((x', y)\) pairs where \(x'\) is the \(x\) image flipped horizontally.

ii. Data augmentation example

**Solution:** When doing a speech recognition task (recognizing the phrase "Alexa") , taking \((x, y)\) input pairs and augmenting by producing \((x', y)\) pairs where \(x'\) is the \(x\) audio sample and adding in background noise (the sound of children playing, perhaps).

(c) Give two examples of noise injection:

i. Noise injection example

**Solution:** Adding a small amount of random noise to an input image when doing image recognition.

ii. Noise injection example

**Solution:** Adding small amount of random noise to weights when doing the forward pass (during training).

4. How are loss and accuracy measures used differently for training, validation, and test (holdback) datasets?

**Solution:** The training accuracy tells you whether the model is underfitting (if accuracy is low, the model is underfitting).

The validation loss vs. training loss tells whether the model is overfitting (if validation loss is much greater than training loss).

The test accuracy tells the expected accuracy when generalizing in the wild (the model may be overfit to the validation dataset since many hyperparameters have changed to ensure good results on the validation dataset).
5. Assume you are training a neural network on a training set $T$ and a validation set $V$ using the standard simple SGD optimization function:

$$\Theta = \Theta - \lambda \frac{\partial L}{\partial \Theta}$$

There are three different choices that can be used in the training loop on each epoch:

- **batch** gradient descent (optimizing on all of $T$),
- **mini-batch** SGD (optimizing using successive sub-batches of a random ordering of $T$), and
- **true** SGD (optimizing using single elements of $T$ in a random order)

For all of the following, assume the gradient after scaling by $\lambda$ is non-zero.

(a) For which, if any, of the following will the direction of the gradient always be in a direction that would decrease the *training* loss:

- [ ] batch SGD
- [ ] mini-batch SGD
- [ ] true SGD

**Solution:** For batch SGD, we’re computing the gradient across all the training set. That is, we’re determining in which direction to go to reduce the training loss across the entire dataset.

For the others, we are only using a subset of the training set, and thus, the gradient for that subset may not lead us in the right direction.

(b) For which, if any, of the following will the *training* loss always be improved after a single gradient descent/parameter-update:

- [ ] batch SGD
- [ ] mini-batch SGD
- [ ] true SGD

**Solution:** If the learning rate is too large, all losses will become worse. Even though the

(c) For which, if any, of the following will the direction of the gradient always be in a direction that would decrease the *validation* loss:

- [ ] batch SGD
- [ ] mini-batch SGD
- [ ] true SGD
Solution: If we’re overfitting, any or all of these choices can cause the gradient to be in a direction that would increase the training loss.

(d) For which, if any, of the following will the validation loss always be improved after a single gradient descent/parameter-update:
   - batch SGD
   - mini-batch SGD
   - true SGD

Solution: From part (c), our direction may be wrong, so our update may very well cause the loss to get worse.
If we’re overfitting, any or all of these choices can cause the validation loss to increase.
6. List 3 techniques for reducing overfitting in a Neural Network:

**Solution:**

- Regularization (many choices here)
- Reducing number of layers
- Reducing the size of each layer
- Increasing the training dataset size

7. I would like to hear your opinions of CS 134 so far. Please answer the following two questions (any answer except no answer at all will receive full credit).

(a) 2 points  What are the three best parts of CS 134?

**Solution:**

(b) 2 points  What are the three worst parts of CS 134?

**Solution:**
8. Bobby trains a model on a 90K training dataset and does validation using a 10K validation dataset. Once Bobby trains the model on a final set of hyper-parameters, the accuracy on the training and validation datasets shows similar loss and accuracy metrics. Call this model $M_{90}$.

Sandy suggests that rather than deploying $M_{90}$, that the model be retrained with all 100K samples (90K training dataset plus 10K validation dataset) used as a new training dataset to create $M_{100}$. This new model would be trained with exactly the same hyper-parameters as $M_{90}$, including the same number of epochs. Sandy claims that $M_{100}$ should be deployed rather than $M_{90}$.

(a) Which model ($M_{90}$ or $M_{100}$) would have better training set accuracy and why?

**Solution:** It’s always easier for a model to fit fewer samples rather than more, so $M_{90}$ should have no worse training set accuracy, although it may not be better.

(b) Which model would have better generalizability (have similar accuracy in the real world as compared to on the training dataset)?

**Solution:** Seems that $M_{100}$ should be the same or better since more data reduces underfitting.

(c) In general, are there any hyper-parameters that might need to change when training with more samples?

**Solution:** Might want to increase the number of epochs. Since under-fitting is reduced, we should be able to train for more epochs before beginning to overfit. However, it’s unclear to know when we’ll start overfitting (since we have no validation loss to examine). We could perhaps continue training until the training loss on $M_{100}$ matches the final training loss of $M_{90}$. 

Page 9