UNIVERSITY OF PENNSYLVANIA

ESE 546: PRINCIPLES OF DEEP LEARNING

FALL 2020

(10/28) MID-TERM EXAM

DURATION: 100 MINUTES

Read the following instructions carefully before you begin.

- This is a closed book exam. You are allowed to use one A4/Letter paper cheat sheet (front and back). You may not use laptops, phones or the Internet during this exam. You may not discuss with your peers.
- The exam is designed to be completed in 75 minutes. You will have 100 minutes from the time you start the exam on Gradescope to upload your solutions. Late submissions will only receive 50% of the credit, so at least keep a buffer of 10-15 minutes for uploading your solutions into Gradescope.
- You should use pen and paper (do not use a pencil because it might not be readable when scanned) to write your solutions and click pictures and upload your solutions as a PDF. Please make sure to write in legible handwriting. You can also use a tablet to write your solutions.
- Begin each problem on a fresh page. Solutions that are not correctly annotated on the Gradescope outline will not receive credit.
- Each question mentions how short/long we would like your responses to be. DO NOT write excessively verbose answers.
- None of the questions require long derivations. If you find yourself going through lots of equations reconsider your approach or consider moving on to the next question.
- If you are stuck, explain your answers and what you are trying to do clearly. We will give partial credit if you are on the right track.
- The questions are NOT arranged in order of difficulty, try to attempt every question. **This exam has 7 problems**.
- In case of emergency, send the Instructor an email at pratikac@seas.upenn.edu.

Problem 1. (28 points)

(1) **(8 points)** How do the following affect the bias-variance tradeoff? Write "increases", "decreases" or "no effect" in the cells. No explanations necessary.

Answer:

	Bias	Variance
Get more data from the same distribution	No effect	Decreases
Dropout	Increases	Decreases
Too much data augmentation	Increases	Decreases
Remove all non-support samples in an SVM	No effect	No effect

(2) **(4 points)** Is the cross-entropy loss zero if the network makes the correct prediction on a datum? Explain your answer in 1-2 sentences.

Answer: The cross-entropy loss is

$$-\log(z_y).$$

where z is the softmax over the logits $\hat{y}$. A correct prediction is achieved when

$$y = \underset{i}{\text{argmax}}\, \hat{y}_i.$$

The cross-entropy loss is therefore not zero even when the network makes the correct prediction.

(3) **(4 points)** Explain in one sentence why surrogate losses are used for classification problems in machine learning. Give one example of a surrogate loss for binary classification and write the mathematical expression for it.

Answer: The zero-one loss which measures the mispredictions of the model is not differentiable. Losses such as the hinge loss (or exponential, logistic loss) are differentiable surrogates for the zero-one loss.

(4) **(4 points)** Answer if the following statements are True or False. No explanations necessary.

(i) A kernel SVM can only fit data that are linearly separable.

(ii) Given an input signal $x = [1, 1, 2, 1, 1]$ and a convolutional kernel $w = [-1, 1, 1]$, the output of the convolution operation with zero-padding is

$$x * w = [2, 2, 2, 0, 0].$$

(iii) Pooling builds translational invariance in a convolutional network.

(iv) We can reduce the variance of predictions of a model by using Dropout at test time and averaging the predictions of the model for each mask.

Answer: (i) False, (ii) False, (iii) True, (iv) True.

(5) **(4 points)** Explain Inverted Dropout in 2-3 sentences.

Answer: Inverted Dropout scales the activations of the Dropout layer by $1/(1-p)$ during training. This precludes the need to modify the weights or the activations when the model is used at test time without the Dropout mask.

(6) **(4 points)** Explain how autograd in PyTorch works in 2-3 sentences. You can use the function

```
def f(w, x):
    yh = numpy.dot(w, x)
    return yh
```

to explain. Focus on what are the inputs and outputs of the forward and backward for the function $\hat{y} = f(w, x)$ above.

Answer: Autograd creates a forward computational graph for the function $w, x \mapsto \hat{y}$. It writes a new backward function automatically that looks like

```
def df(dyh):
    dx = dyh * w
    dw = dyh * x
    return dx, dw
```

i.e., takes as input $\overline{\hat{y}}$ and returns as output the two variables

$$\overline{x} = \overline{\hat{y}}\, w,$$

$$\overline{w} = \overline{\hat{y}}\, x.$$

Problem 2. (15 points) We would like to build a deep network for the following classification problem. Given a video of length 10 seconds with 640× 480-sized RGB frames at 10 frames/second, we would like to predict different activities that people are performing in this video, e.g., playing a guitar, bouncing a basketball, cutting vegetables etc. We are given the true activity being performed as the label for each video in the training set.

Describe in 5-6 sentences what architecture you will use for this problem and why. Describe what loss you will use to train your architecture. Feel free to draw a sketch of the architecture and annotate different parts of it.

Answer: Sketch out the answer. The main points are as follows.

(a) Use a CNN to compute the features for each frame and combine it with a recurrent architecture to capture the temporal properties of the sequence. Some students used 3D convolutions which is a very good idea. Using recurrence is important because recognizing the activity being performed in the video depends on the past few frames, not just the current one.

(b) Each time-step can predict a the target activity using softmax and you can average the loss of each time-step (perhaps with more weight given to the later time-steps because they have more context) to train the model. Most students answered that they will use the prediction of the final time-step as the loss, which is also fine. A large fraction of solutions were not clear as to how they compute the loss; merely saying that "we should use the cross-entropy loss" is not enough, you should also mention what the input to the loss is exactly.

(c) It is important to realize that the sequence has a length of 100 so we want a more powerful recurrent model like GRU/LSTM/attention to tackle gradient vanishing.

(d) Some students designed a new architecture that incorporates convolutions inside the recurrent layers, e.g.,

$$h_{t+1} = \sigma\left(w_h * h_t + w_x x_{t+1}\right)$$

which is also a neat idea.

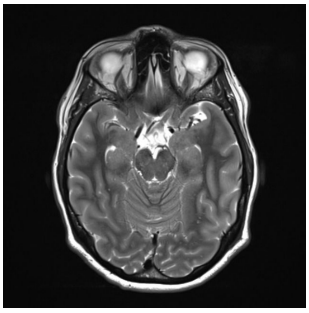
In general, students who attempted this question did well on it.

Problem 3. (15 points)

(1) **(4 points)** What is the Bayes error if the true labels y created by Nature are a deterministic function of the data x? Explain your answer in 1-2 sentences.

> **Answer:** The Bayes error is $\mathbb{E}_{x,y}\left[(f^*(x) - y)^2\right]$, is zero if the true labels y are a deterministic function of the data x. This is because the conditional expectation $f^*(x) = \mathbb{E}\left[y|x\right]$ is exactly equal to y if the true labels are generated without any randomness. Note: Most real datasets are created using high-quality annotations, so we expect Bayes error to be small, if not zero.

(2) **(4 points)** Let us say we are classifying a neurological disease using Magnetic Resonance Imaging (MRI) images of the brain. A typical image is shown below.



The sharpness of the image varies in images from different hospitals because they have different MRI machines, and also because people may not lie perfectly still while inside the machine. Different patients will have slightly different brain sizes (children have a smaller head, adults have a slightly shrunken brain than people in their twenties etc.). Suppose you are training a neural network using images from Penn's hospital but this model will be used in Harvard's hospital at test time.

Write down the two most important data augmentation techniques you will use with a one sentence explanation for each.

> **Answer:** (i) Blurring to make the network insensitive to sharpness of the image, or movement of the people. (ii) Small amounts of zoom or in general diffeomorphism deformations of the input image to generalize to the brain of different people. (iii) Augmentations with small translation and rotation of the image will counter the fact that people may move while the MRI machine takes an image. (iv) There are many other possible answers such as adding noise to the input images to mitigate sharpness, most of these alternate answers received credit.

(3) **(3 points)** Consider a three-layer linear neural network for the dataset $\{(x^i, y^i)\}_{i=1,\dots,n}$ with $x^i, y^i \in \mathbb{R}^d$ that is trained using the squared loss

$$A^*, B^*, C^* \in \operatorname*{argmin}_{A,B,C} \ell(A, B, C) := \frac{1}{n} \sum_{i=1}^{n} \|y^i - ABC\, x^i\|^2;$$

here $A, B, C \in \mathbb{R}^{d \times d}$ are the weights. Answer true or false for the following. No explanations necessary.

(i) The above optimization problem is convex.

(ii) The minimum takes a unique value.

(iii) The minimizer A^*, B^*, C^* is unique.

Answer: False: (i), (iii). True: (ii)

(4) **(4 points)** Explain in 1-2 sentences each. What are the benefits of using a GRU over an RNN for sequence prediction? What are the benefits of using an attention layer in recurrent architectures such as RNNs, GRUs or LSTMs.

Answer: GRU has a zero and reset gate that mitigates the gradient vanishing problem in RNNs; this is done by updating the hidden state in an RNN only at specific time-steps, based on whether the zero active is active or not.

An attention layer computes the correlation of the features across multiple time-steps. This provides a much shorter path for the backpropagation gradient to flow from the loss to each time-step and essentially eliminates the gradient vanishing problem in recurrent architectures.

Problem 4. (10 points) Consider a dataset $\left\{(x^i, y^i)\right\}_{i=1,\dots,n}$ where $x^i \in \mathbb{R}^d$ and $y^i \in \mathbb{R}$. If we arrange all the data in a matrix $X \in \mathbb{R}^{n\times d}$ and targets as a vector $Y \in \mathbb{R}^n$ we can perform linear regression by solving

$$w^* \in \operatorname*{argmin}_{w\in\mathbb{R}^d} \|Y - Xw\|_2^2 \tag{1}$$

when we have $d \leq n$, i.e., the number of data is more than the number of parameters to be estimated. If we have fewer data than the number of parameters $d > n$, the solution to the above problem is not unique. In this case, we pick one of the many solutions, say the one with the smallest ℓ_2 norm, using

$$\begin{aligned} w^* = \min_{w\in\mathbb{R}^d} \frac{1}{2}\|w\|_2^2 \\ \text{such that } Y = Xw. \end{aligned} \tag{2}$$

(8 points) Derive the solution w^* for the problem in (2). Assume $XX^\top$ is invertible.

(2 points) Write down all solutions of (1). Hint: think of the null-space of the matrix X.

Answer: We write down the Lagrangian as

$$L = \frac{1}{2}\|w\|_2^2 + \lambda^\top (Y - Xw);$$

note that we have n constraints $Y = Xw$, so $\lambda \in \mathbb{R}^n$. Now set the derivatives to zero

$$\frac{\mathrm{d}L}{\mathrm{d}w} = 0 \Rightarrow w^* - X^\top\lambda = 0$$
$$\frac{\mathrm{d}L}{\mathrm{d}\lambda} = 0 \Rightarrow Y = Xw^*.$$

Solve this to get (assuming $XX^\top$ is invertible)

$$\lambda = \left(XX^\top\right)^{-1} Y$$
$$w^* = X^\top \left(XX^\top\right)^{-1} Y.$$

Notice that this is different from the standard least-squares solution $\left(X^\top X\right)^{-1} X^\top Y$. Any weight vector

$$w^* = X^\top \left(XX^\top\right)^{-1} Y + v$$

where $v \in$ null-space(X), i.e., v is such that $Xv = 0$ is a solution of (1).

Problem 5. (16 points) Consider the following neural network:

$$32 \times 32 \text{ RGB images} \rightarrow \text{nn.Conv2d(3, 10, 5)} \rightarrow \text{nn.BatchNorm2d(C)},$$

i.e., we perform a convolution using a 5×5 kernel on the input images to get 10 output channels. Let us focus on the batch-normalization (BN) layer.

(1) **(2 points)** Suppose that the stride of the convolution layer is 1. How many pixels should we use for padding if we want the size of the input image (width and height) to remain unchanged after convolution? Explain in 1-2 sentences.

> **Answer:** The padding for each side (width and height) should be 2. This should be half the kernel-width which is 5.

(2) **(4 points)** Explain how batch-normalization works in 1-2 sentences. No need to write any formulae. Focus on (i) what parameters/buffers BN has, (ii) how it works at training time, and (iii) how it works at test time.

> **Answer:** For each mini-batch, BN subtracts the mean of the activations and divides by the standard deviation. It applies an affine transformation on the normalized activations. It maintains the running mean/std-dev statistics of the training data and uses these stored statistics to normalize the test data.

(3) **(2 points)** The BN layer above in PyTorch has a parameter C for the number of features that we would like you to fill in. What is the value of C? No need for an explanation.

> **Answer:** The batch-norm layer should have the same number of channels as the output of the convolutional layer, so $C = 10$.

(4) **(2 points)** How many backprop-updatable parameters does the BN layer in the above network have? No need for an explanation.

> **Answer:** Affine parameters are updated using back-prop. There are 10 output channels: this gives 10 weights and 10 bias parameters for the BN layer, one for every channel; this totals to 20 parameters.

(5) **(2 points)** Explain in 1-2 sentences how the running mean and running variance are updated. How many such parameters does the above network have?

Answer: Running mean and variance are exponential averages of the mini-batch mean and variance. There are 10 parameters for running mean and 10 more for running variance, one for each channel.

(6) **(4 points)** If you trained your network on one dataset and would like to use the model on a new dataset with slightly different image statistics, how can you update the running mean and running variance buffers of the BN layer without doing any backpropagation updates? Assume that weights and biases of convolutional and fully-connected layers are going to work as-is on the new data.

Answer: You can set the model in training mode using "model.train()" and forward propagation the new dataset through the model without any backpropagation updates. The running mean and variance parameters will be updated as usual using exponential averaging. We can then run this adapted model for inference on the new dataset.

Problem 6. (6 points) Consider a matrix factorization problem of the form

$$A^*, B^* = \underset{A,B}{\operatorname{argmin}} \|X - A^\top B\|_F^2 + \frac{\lambda}{2} \sum_{i=1}^{p} \left(\|A_i\|_2^2 + \|B_i\|_2^2 \right).$$

Here $X \in \mathbb{R}^{d \times d}$ and $A \in \mathbb{R}^{p \times d}$, $B \in \mathbb{R}^{p \times d}$ with $d > p$. The first term has the squared Frobenius norm $\|Z\|_F^2$ which is simply the sum of squares all entries in the matrix Z. The second term is the sum of the ℓ_2 norms of the columns of the matrices A, B; it is similar to weight decay.

We know from the homework that the solution A^* and B^* with $\lambda = 0$ is given by the first p columns of the singular value decomposition (SVD) of $X = U\Sigma V^\top$ where $U, V \in \mathbb{R}^{d \times d}$ and Σ is a diagonal matrix of the singular values of X with the singular values arranged in descending order. Say,

$$A^* = \Sigma_{1:p} U_{1:p}^\top, \quad B^* = V_{1:p}^\top,$$

where $U_{1:p}$ are the first p columns of U and likewise for the other matrices.

What is the new solution A^* and B^* if the rank of X is $p - 1$. In other words, what happens when A, B that we are fitting have one more column than necessary to fit the data. **You do not need to do any calculations for this problem, think logically.**

Answer: The data matrix X has rank $p - 1$, so it has $p - 1$ non-zero eigenvalues. As before, the first $p - 1$ columns of A^* and B^* are obtained from the solution of the SVD. The size of A^*, B^* is however $p \times d$, so we need some way to fill in the last column. What should the last column be? The weight-decay term makes sure that the magnitude of the entires of A_p and B_p is as small as possible. Since with the first $p - 1$ columns we have already perfectly minimized the Frobenius norm term, the last column of both A and B has to be zero.

Note: If the value of λ is very large, it may happen that the optimal solution will pick only some $m < p - 1$ columns from the SVD solution and set all other $p - m$ columns of A and B to zero.

Problem 7. (10 points)

(1) **(4 points)** Recall that a function $\ell(w)$ is strongly convex if we can find an $m > 0$ such that

$$\ell(w) - \frac{m}{2}\|w\|_2^2$$

is a convex function. For the least-squares loss function

$$\ell(w) = \frac{1}{2}\|Y - Xw\|_2^2$$

where $X \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^n$ are the input and targets respectively, find the largest strong-convexity parameter m.

Answer: The least-squares objective is twice-differentiable, so we have

$$\nabla^2 \ell(w) = X^\top X \in \mathbb{R}^{d \times d}.$$

This is a positive-definite matrix so the loss function is indeed convex. The best m that we can use is the smallest eigenvalue of the Hessian

$$m = \lambda_{\text{min}} \left(X^\top X \right).$$

(2) **(2 points)** Consider the two-layer network

$$\mathbb{R} \ni \hat{y} = v^\top \text{relu}(w^\top x).$$

and the regression loss

$$\ell(w, v; x, y) = \left(y - v^\top \text{relu}(w^\top x) \right)^2$$

where $x \in \mathbb{R}^d, y \in \mathbb{R}$ are the input and target respectively. The ReLU nonlinearity is a convex function and so is the squared loss. Why is fitting this model still a non-convex optimization problem?

Answer: Fitting this model is a non-convex problem because of the multiplicative terms involving the two parameters v, w. Also a composition of $f \circ g$ of two convex functions f, g is convex only if f is a non-decreasing function. Since the squared loss is not non-decreasing, the composition is non-convex even if either of the parameters v, w are fixed.

(3) **(4 points)** Explain k-fold cross-validation in 2-3 sentences. Mention why you may want to use cross-validation instead of simply partitioning the data into the training, validation and test sets.

Answer: Cross-validation should be used when do not have a very large dataset to sequester it as training, validation and test data. In this case, we partition the data into two sets, training and test. The training data is further partitioned into multiple pairs of subsets with a fraction $(k-1)/k$ samples for training and fraction $1/k$ samples for validation. The validation error of each model corresponds to fitting k models on each of the k folds and averaging the error of the corresponding validation fold.

END OF EXAM