DS-UA 301
Advanced Topics in Data Science
Instructor: Parijat Dube

Midterm Exam - 100 points

Oct 28, 2022

Due: Oct 30 11:59pm
Instructions:

- The Midterm Exam has a total of 12 questions, 6 are 5 points questions and 5 are 10 points questions and one 20 point question.
- You can attempt the questions in any order but clearly mention the Problem number in your submission.
- Problems marked with asterisk require you to submit code. You need to solve them using PYTHON and submit the code. If no code is provided you will get zero credit.
- Problems not marked in asterisk can be done without any computational software (like PYTHON). However if you need to use them that should be fine.
- To get full points for any question you need to have the right final answer **AND** show all the steps. Just writing the final answer will give you zero credit even if the answer is correct.
- Your answers should be legible and clearly written. Please make sure all the pages are submitted and uploaded correctly. **Absolutely NO requests for missing submissions will be entertained after the exam is over.**

Problem 1 - 5 points

1. (**2.5**) Provide a sketch of typical (squared) bias, variance, training error, test error, and irreducible error curves, on a single plot, as model complexity increases. The x-axis should represent the model complexity, and the y-axis should represent the values for each curve. There should be five curves. Make sure to label each one.
2. (**2.5**) Explain why each of the five curves has the shape displayed in part 1.

Problem 2 - 5 points

Table 1 in Figure-1 provides a training data set containing six observations, three features, and one target label. Suppose we wish to use this data set to make a prediction for Y when $X_1 = X_2 = X_3 = 0$ using K-nearest neighbors.

1. (**2**) Compute the Euclidean distance between each observation and the test point, $X1 = X2 = X3 = 0$.
2. (**1.5**) What is our prediction with $K = 1$ for this test point? Why?
3. (**1.5**) What is our prediction with $K = 3$? Why?

Midterm Exam - 100 points

Obs.	X_1	X_2	X_3	Y
1	0	3	0	Red
2	2	0	0	Red
3	0	1	3	Red
4	0	1	2	Green
5	−1	0	1	Green
6	1	1	1	Red

Figure 1: Table 1

Problem 3 - 5 points

Suppose we collect data for a group of students in a statistics class with variables X_1 = hours studied, X_2 = undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficient, $\beta_0 = -6, \beta_1 = 0.05, \beta_2 = 1$.

1. (**1**) Write the expression for probability of getting an A using the logistic function.
2. (**2**) Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.
3. (**2**) How many hours would the student in part 2 need to study to have a 50% chance of getting an A in the class?

Problem 4 - 5 points

1. (**2**) While learning rate is decayed using a learning rate schedule as training progresses, it needs to be done wisely. If the decay is too aggressive, it may prevent any learning. To understand this an experiment was done in which four different decay schedules we used. Look at the code in Figure 2 that was used to decay the learning rate in this experiment.

```
# demonstrate the effect of decay on the learning rate
from matplotlib import pyplot

# learning rate decay
def decay_lrate(initial_lrate, decay, iteration):
    return initial_lrate * (1.0 / (1.0 + decay * iteration))

decays = [1E-1, 1E-2, 1E-3, 1E-4]
lrate = 0.01
n_updates = 200
for decay in decays:
    # calculate learning rates for updates
    lrates = [decay_lrate(lrate, decay, i) for i in range(n_updates)]
    # plot result
    pyplot.plot(lrates, label=str(decay))
pyplot.legend()
pyplot.show()
```

Figure 2: Code

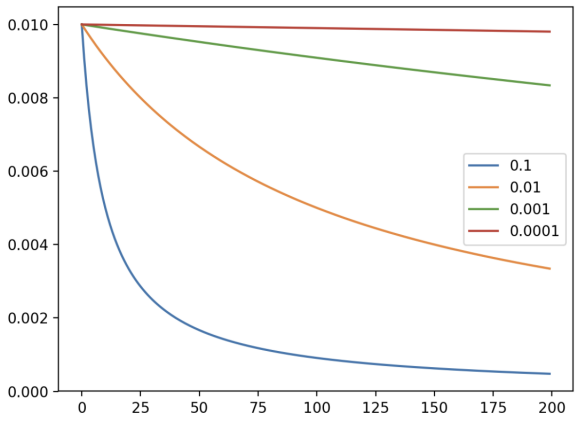

Figure 3: Learning Rate

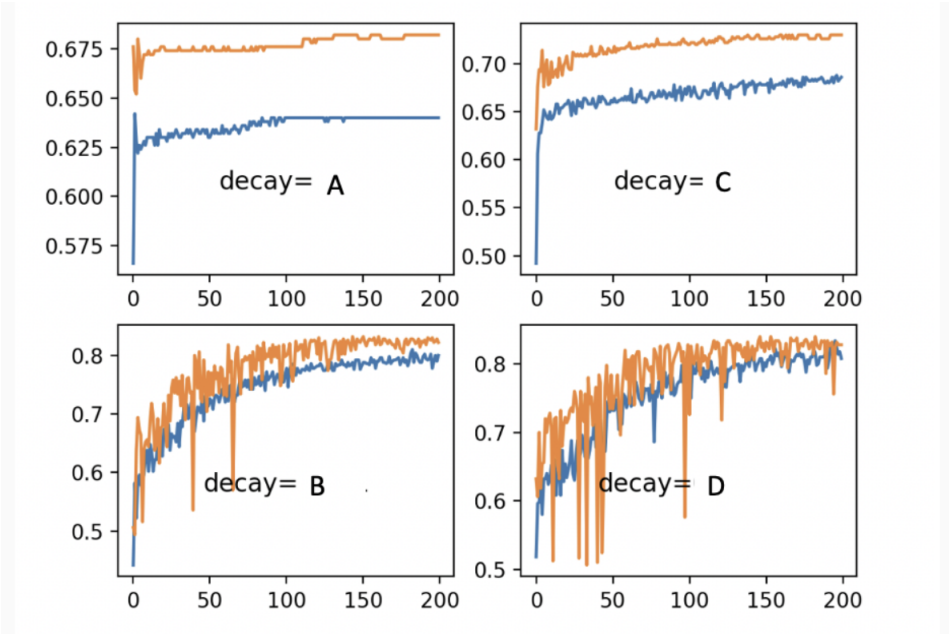

Figure 4: Train and test accuracies

Running the code in Figure 2 creates a line plot showing learning rates over updates for different decay values as shown in Figure 3.

Figure 4 shows the training and test accuracy for the 4 training jobs, each with one of the four learning rate schedules. Here each schedule is identified by the decay value. Each figure is labeled with an alphabet for the decay value.

The four decay values are:

0.1, 0.01, 0.001, 0.0001

You need to match the decay value with the right alphabet.

- A
- B
- C
- D

2. (**3**) An early stopping criterion is created using the generalization loss as: "Stop as soon as the generalization loss (GL) exceeds a certain threshold", where GL (expressed in percentage) is defined as:

$$GL(t) = 100 \left(\frac{E_{va}(t)}{E_{opt}(t)} - 1 \right)$$

where $E_{va}(t)$ is the validation error at epoch t, and $E_{opt}(t)$ is the minimum validation error till epoch t, which is defined as:

$$E_{opt}(t) = \min_{t' \leq t} E_{va}(t')$$

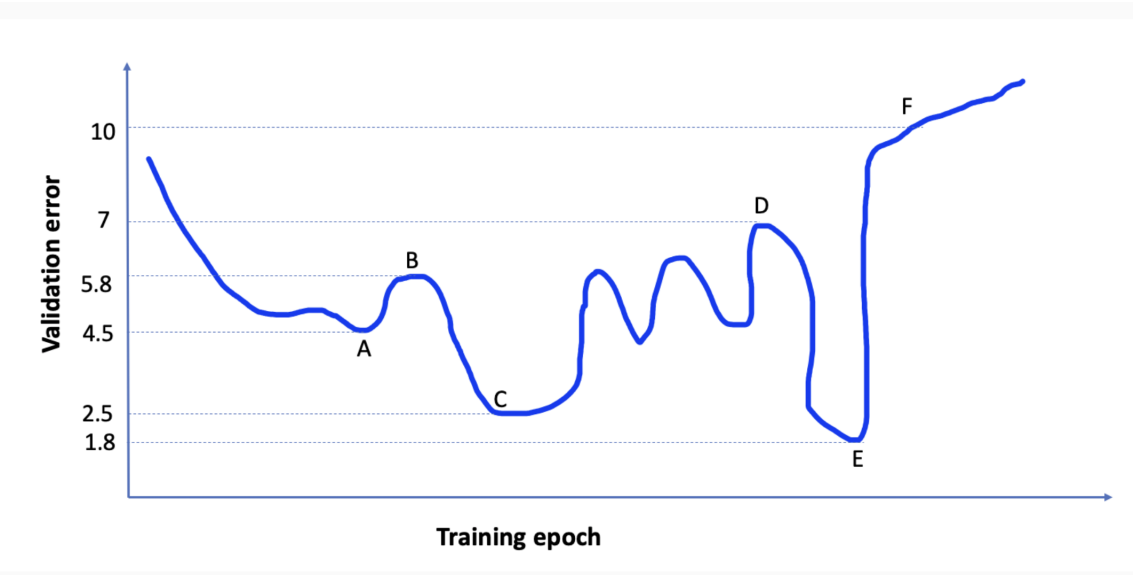

Figure 5: Early Stopping

From the validation error plot (Figure 5) identify the following:

(a) Early stopping epoch for 28% threshold
(b) Early stopping epoch for 150% threshold
(c) Epoch at which overfitting is correctly detected
(d) Epoch at which the model has the smallest generalization error

Your answers for the above should be in terms of epochs A, B, C, D, E, and F only. These epochs are marked in the validation plot.

Problem 5 - 5 points

Let there be four source datasets labeled $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$, each with an average feature representation given as a 4 dimensional vector:

$$\mathbf{A} : [1, 0, 0, 0] \quad \mathbf{B} : [0, 1, 0, 0] \quad \mathbf{C} : [1, 0, 1, 0] \quad \mathbf{D} : [0, 1, 0, 1]$$

Consider the following approach for creating pseudolabels for an image:

- First take cosine similarity between the image's feature representation and the feature representation of each of the four datasets. Recall that the cosine similarity between two vectors x and y is given by $\frac{x \bullet y}{\|x\|\|y\|}$, where $\bullet$ is for dot product between the two vectors.
- Concatenate the labels of nearest n datasets to create a pseudolabel for the image. Given two datasets, the one with which the cosine similarity of image is higher, is nearer than the other. For example if for an image, the cosine similarity with these 4 datasets is [A: -0.2, B: 0.5, C: 0.7, D: -0.3] then its pseudolabels under different Nearest-n schemes is as follows:

$$\text{Nearest-1} : \mathbf{C} \quad \text{Nearest-2} : \mathbf{CB} \quad \text{Nearest-3} : \mathbf{CBA} \quad \text{Nearest-4} : \mathbf{CBAD}$$

Using the approach given above, find the pseudolabels for the following two images under the Nearest-1, Nearest-2, Nearest-3 and Nearest-4 schemes. The image feature vectors are given as:

1. Image X : $[-1, 1, -1, 1]$
2. Image Y : $[1, -1, 1, -1]$

*Problem 6 - 5 points

This question is based on LeNet architecture as presented in the following paper:
LeCun et al. Gradient Based Learning Applied to Document Understanding.

1. (**1**) How many connections are there in the first convolutional layer?
2. (**1**) Why third convolutional layer is technically a fully connected layer?
3. (**1**) What type of layers are contributing the maximum number of trainable parameters?
4. (**2**) How did we get 1516 trainable parameters in the second convolution layer?

*Problem 7 - 10 points

Consider the case when $f(x) = x - 2x^2$ and $y(x) = f(x) + \mathcal{N}(0, 1)$, where $\mathcal{N}(0, 1)$ is normal distribution with mean 0 and standard deviation 1. Create a dataset of size 100 points by randomly sampling x from $\mathcal{N}(0, 1)$ and then using the expression for $y(x)$ to get the corresponding y.

1. (**2**) Display the dataset and $f(x)$. Use scatter plot for y and smooth line plot for $f(x)$.
2. (**4**) Set a random seed, and then compute the LOOCV (Leave One Out Cross Validation) errors that result from fitting the following four models using least squares:
   (a) $Y = \beta_0 + \beta_1 X + \epsilon$
   (b) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$
   (c) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$
   (d) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$
3. (**2**) Repeat 2 using another random seed, and report your results. Are your results the same as what you got in 2 ? Why?
4. (**2**) Which of the models in part 2 had the smallest LOOCV error? Is this what you expected? Explain your answer.

- We can set the seed using numpy.random.seed(x), where x is an integer
- Sklearn LOOCV link: LeaveOneOut

*Problem 8 - 10 points

You will create a Neural Network in PyTorch to classify a dataset of images. The dataset we are going to use is CIFAR10, which contains 50K 32x32 color images. The model we are going to build is ResNet-18, as described in https://arxiv.org/abs/1512.03385 . The reference code is at https://github.com/kuangliu/pytorch-cifar.

1. (**4**) Create a PyTorch program with a DataLoader that loads the images and the related labels from the torchvision CIFAR10 dataset. Import CIFAR10 dataset for the torchvision package, with the following sequence of transformations:
   (a) Random cropping, with size 32x32 and padding 4
   (b) Random horizontal flipping with a probability 0.5
   (c) Normalize each image's RGB channel with mean(0.4914, 0.4822, 0.4465) and variance (0.2023, 0.1994, 0.2010)

   The DataLoader for the training set uses a minibatch size of 128 and num_workers=2. The DataLoader for the testing set uses minibatch size of 100 and num_workers =2. You can refer to: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
2. (**4**) Create a main function that creates the DataLoaders for the training set and the neural network, then run 5 epochs with a complete training phase on all the minibatches of the training set.
3. (**2**) For each minibatch calculate the training loss value, the training accuracy of the predictions, measured on training data.

Note: (i) No need to write your own ResNet module, you can re-use as much code as you want from the above-mentioned github (ii) Typically, we would like to examine test accuracy as well, however, it is sufficient to just measure training loss and training accuracy for this question.

*Problem 9 - 10 points

Using the VGG19 as pretrained model (pretrained on Imagenet1K) architecture build a custom classifier for MNIST handwritten digits classification by following two approaches:

1. (**4**) Use VGG19 as a feature extractor and send them to dense layers which are trained according to our data set.
2. (**4**) Freeze the weights of first 10 layers of the VGG19 network, while retrain the subsequent layers.
3. (**2**) Which approach gives a better transferability and why?

Train for 10 epochs and use a learning rate of 0.001 with Adam optimizer with a batch size of 32. You can create models either in Tensorflow or Pytorch.

Problem 10 - 10 points

1. (**5**) Refer to VGG16 architecture on page 3 in Table 1 of the paper by Simonyan et al available at https://arxiv.org/pdf/1409.1556.pdf. You need to complete Table 1 below for calculating activation units and parameters at each layer in VGG16 (without counting biases).
2. The original Googlenet paper by Szegedy et al. (available at https://arxiv.org/pdf/1409.4842.pdf) proposes two architectures for Inception module, shown in Figure 2 on page 5 of the paper, referred to as naive and dimensionality reduction respectively.
   (a) (**2.5**) Assuming the input to inception module (referred to as "previous layer" in Figure 2 of the paper) has size 32x32x256, calculate the output size after filter concatenation for the naive and dimensionality reduction inception architectures with number of filters given in Figure 6.
   (b) (**2.5**) Next calculate the total number of convolutional operations for each of the two inception architecture again assuming the input to the module has dimensions 32x32x256 and number of filters given in Figure 6.

*Problem 11 - 10 points

This question focuses on batch normalization.

1. (**2**) Which of the following statements are true about batch normalization? (Select all that apply.)

Layer	Number of Activations (Memory)	Parameters (Compute)
Input	224*224*3=150K	0
CONV3-64		
CONV3-64		
POOL2		0
CONV3-128		
CONV3-128		
POOL2		0
CONV3-256		
CONV3-256		
CONV3-256		
POOL2		0
CONV3-512		
CONV3-512		
CONV3-512		
POOL2		0
CONV3-512		
CONV3-512		
CONV3-512		
POOL2		0
FC	4096	
FC	4096	4096*4096 = 16,777,216
FC	1000	
TOTAL		

Table 1: VGG16 memory and weights

(a) Batch normalization makes processing a single batch faster, reducing the training time while keeping the number of updates fixed. This allows the network to spend the same amount of time performing more updates to reach the minima.

(b) Batch normalization weakens the coupling between earlier/later layers, which allows for independent learning.

(c) Batch normalization normalizes the output distribution to be more uniform across dimensions.

(d) Batch normalization mitigates the effects of poor weight initialization and allows the network to initialize our weights to smaller values close to zero.

2. (**4**) In the code provided in Figure 7, complete the implementation of batch normalization's forward propagation in numpy code. The following formulas may be helpful:

$$\begin{aligned} z^{(i)}_{norm} &= \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}} \\ \tilde{z}^{(i)} &= \gamma z^{(i)}_{norm} + \beta \end{aligned}$$

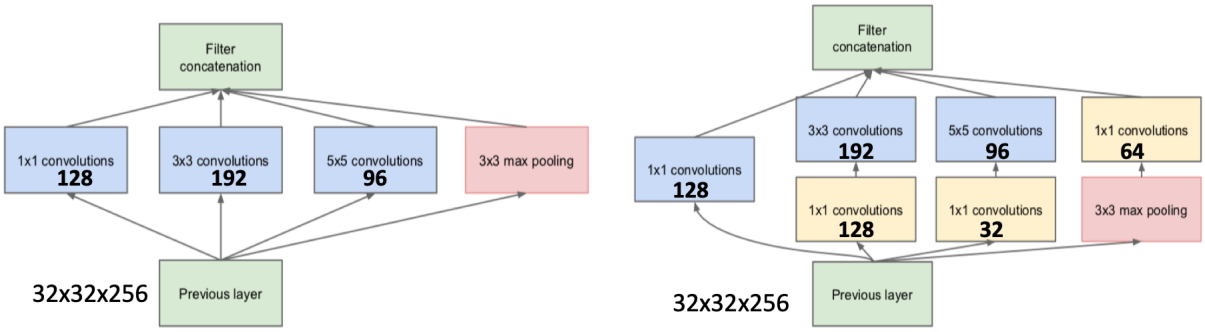

Figure 6: Inception blocks

3. (**1**) What is the role of the ϵ hyperparameter in batch normalization?
4. (**1**) What problem does using batch normalization have with a batch size of 1?
5. (**2**) You are applying batch normalization to a fully connected (dense) layer with an input size of 20 and output size of 30. How many training parameters does this layer have, including batch normalization parameters?

Problem 12 - 20 points

Assume you have an input of size $H \times W \times C$. You need to create a 2-D convolution layer with a receptive field of 7×7. The output size should be also $H \times W \times C$.

1. (**10**) Provide five different architectures that can achieve this. *Hint: Remember receptive field equivalence of stacked convolution layers and power of small filters.*
2. (**5**) Calculate the total number of trainable parameters for each of the five architectures. Which has the most number of trainable parameters and which has the least?
3. (**5**) Compute the total computations required for each of the five architectures. Which has the largest computational requirement and which has the smallest?

```python
def forward_batchnorm(Z, gamma, beta, eps, cache_dict, beta_avg, mode):
    """
    Performs the forward propagation through a BatchNorm layer.

    Arguments:
    Z -- input, with shape (num_examples, num_features)
    gamma -- vector, BN layer parameter
    beta -- vector, BN layer parameter
    eps -- scalar, BN layer hyperparameter
    beta_avg -- scalar, beta value to use for moving averages
    mode -- boolean, indicating whether used at 'train' or 'test' time

    Returns:
    out -- output, with shape (num_examples, num_features)
    """

    if mode == 'train':
        # TODO: Mean of Z across first dimension
        mu =

        # TODO: Variance of Z across first dimension
        var =

        # Take moving average for cache_dict['mu']
        cache_dict['mu'] = beta_avg * cache_dict['mu'] + \
            (1 - beta_avg) * mu

        # Take moving average for cache_dict['var']
        cache_dict['var'] = beta_avg * cache_dict['var'] + \
            (1 - beta_avg) * var
    elif mode == 'test':
        # TODO: Load moving average of mu
        mu =

        # TODO: Load moving average of var
        var =

    # TODO: Apply z_norm transformation
    Z_norm =

    # TODO: Apply gamma and beta transformation to get Z tilde
    out =

    return out
```

Figure 7: Problem 11