COMS 6998-015
PRAC DEEP LEARNING SYS
Instructor: Parijat Dube and Chen Wang
Due: Sept. 30, 2024

Homework 1

Problem 1 - *Bias Variance Tradeoff, Regularization* 35 points

1. **[5 points]** Derive the bias-variance decomposition for a regression problem, i.e., prove that the expected mean squared error of a regression problem can be written as

$$E[MSE] = Bias^2 + Variance + Noise$$

Hint: Let $y(x) = f(x) + \epsilon$ be the true (unknown) relationship and $\hat{y} = g(x)$ be the model predicted value of y. Then MSE over test instance x_i, $i = 1, \ldots, t$, is given by:

$$MSE = \frac{1}{t}\sum_{i=1}^{t}(f(x_i) + \epsilon - g(x_i))^2$$

2. **[4 points]** Consider the case when $y(x) = x + \sin(1.5x) + \mathcal{N}(0, 0.3)$, where $\mathcal{N}(0, 0.3)$ is normal distribution with mean 0 and variance 0.3. Here $f(x) = x + \sin(1.5x)$ and $\epsilon = \mathcal{N}(0, 0.3)$. Create a dataset of size 20 points by randomly generating samples from y. Display the dataset and $f(x)$. Use scatter plot for y and smooth line plot for $f(x)$.

3. **[8 points]** Use weighted sum of polynomials as an estimator function for $f(x)$, in particular, let the form of estimator function be:

$$g_n(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_n x^n$$

Consider three candidate estimators, g_1, g_3, and g_{10}. Estimate the coefficients of each of the three estimators using the sampled dataset and plot $y(x), f(x), g_1(x), g_3(x), g_{10}(x)$. Which estimator is underfitting? Which one is overfitting?

4. **[8 points]** Generate 100 datasets (each of size 50) by randomly sampling from y. Partition each dataset into training and test set (80/20 split). Next fit the estimators of varying complexity, i.e., $g_1, g_2, ...g_{15}$ using the training set for each dataset. Then calculate and display the squared bias, variance, and error on testing set for each of the estimators showing the tradeoff between bias and variance with model complexity. Can you identify the best model?

5. **[10 points]** One way to increase model bias is by using regularization. Lets take the order 10 polynomial and apply $\mathcal{L}_2$ regularization. You can work with any value of regularization rate. You don't need to tune it. Compare the bias, variance, and MSE of the regularized model with the unregularized order 10 polynomial model? Does the regularized model have a higher or lower bias ? What about MSE? Explain.

Note: For part 2 and 3 of this problem limit the range of x range for the 20 points generated to lie between some range, say 0 and 10, to observe overfitting and underfitting. Remember to use the same range for training and testing. Additionally, please note to sort the points (increasing x) before plotting. The graph must contain a scatter plot of the points and line plot of the functions.

For part 4 of this problem there are two different ways to sample x and y when creating 100 datasets.

- Follow the post https://dustinstansbury.github.io/theclevermachine/bias-variance-tradeoff. The idea is to keep the value of x same across all the 100 datasets. The y values will vary since it contains the noise (Normal distribution) component.
- Sample a test set (of size 10) before sampling any training dataset. Then sample training set (of size 40) for each 100 dataset but make sure that none of the 10 test set samples should show in any of the 100 datasets. So all the datasets share this common test set but their train set is different.

  The key is to have a fixed test set even though you have 100 independently sampled training set

Problem 2 - *Precision, Recall, ROC* 20 points

This question is based on two papers, one from ICML 2006 and other from NIPS 2015 (details below). ICML paper talks about the relationship between ROC and Precision-Recall (PR) curves and shows a one-to-one correspondence between them. NIPS paper introduces Precision-Recall-Gain (PRG) curves. You need to refer to the two papers to answer the following questions.

1. [**4 points**] Does true negative matter for both ROC and PR curve? Argue why each point on ROC curve corresponds to a unique point on PR curve.
2. [**10 points**] Select one OpenML dataset with 2 output classes. Use two binary classifiers (Adaboost and Logistic regression) and create ROC and PR curves for each of them. You will have two figures: one containing two ROC and other containing two PR curves. Show the point where an *all positive classifier* lies in the ROC and PR curves. An all positive classifier classifies all the samples as positive.
3. [**6 points**] NIPS paper defined PR Gain curve. Calculate AUROC (Area under ROC), AUPR (Area under PR), and AUPRG (Area under PRG) for two classifiers and compare. Do you agree with the conclusion of NIPS paper that practitioners should use PR gain curves rather than PR curves.

Related papers:

- Jesse Davis, Mark Goadrich, The Relationship Between Precision-Recall and ROC Curves, ICML 2006.
- Peter A. Flach and Meelis Kull, Precision-Recall-Gain Curves: PR Analysis Done Right, NIPS 2015.

Problem 3 - *Learning Rate, Batch Size, FashionMNIST* 15 points

Recall cyclical learning rate policy discussed in Lecture 3. The learning rate changes in cyclical manner between lr_{min} and lr_{max}, which are hyperparameters that need to be specified. For this problem you first need to read carefully the article referenced below. You can find references below in PyTorch with open source implementations of this policy which you can easily build over. You will work with FashionMNIST dataset and the small Inception model described in Figure 3 of https://arxiv.org/pdf/1611.03530.pdf. You would need to modify the PyTorch Implementation of GoogleNet (described in reference no. 3) according to the architecture described in the figure.

1. [**3 points**] Fix batch size to 64 and start with 10 candidate learning rates between 10^{-9} and 10^{1} and train your model for 5 epochs. Plot the training loss as a function of learning rate. You should see a curve like Figure 3 in reference below. From that figure identify the values of lr_{min} and lr_{max}.
2. [**4 points**] Use the cyclical learning rate policy (with exponential decay) and train your network using batch size 64 and lr_{min} and lr_{max} values obtained in part 1. Plot train/validation loss and accuracy curve (similar to Figure 4 in reference).
3. [**8 points**] We want to test if increasing batch size for a fixed learning rate has the same effect as decreasing learning rate for a fixed batch size. Fix learning rate to lr_{max} and train your network starting with batch size 32 and incrementally going upto 8192 (in increments of a factor of 2; like 32, 64...). You can choose a step size (in terms of number of iterations) to increment the batch size. Plot the training loss. Is the generalization of your final model similar or different than cyclical learning rate policy?

References:

Layer	Number of Activations (Memory)	Parameters (Compute)
Input	224*224*3=150K	0
CONV3-64	224*224*64=3.2M	(3*3*3)*64 = 1,728
CONV3-64	224*224*64=3.2M	(3*3*64)*64 = 36,864
POOL2	112*112*64=800K	0
CONV3-128		
CONV3-128		
POOL2	56*56*128=400K	0
CONV3-256		
CONV3-256	56*56*256=800K	(3*3*256)*256 = 589,824
CONV3-256		
CONV3-256		
POOL2		0
CONV3-512	28*28*512=400K	(3*3*256)*512 = 1,179,648
CONV3-512		
CONV3-512	28*28*512=400K	
CONV3-512		
POOL2		0
CONV3-512		
CONV3-512		
CONV3-512		
CONV3-512		
POOL2		0
FC	4096	
FC	4096	4096*4096 = 16,777,216
FC	1000	
TOTAL		

Table 1: VGG19 memory and weights

1. Leslie N. Smith Cyclical Learning Rates for Training Neural Networks. Available at https://arxiv.org/abs/1506.01186.
2. PyTorch implementation of cyclical learning rate policy. Available at PyTorch documentation.
3. GoogLeNet Implementation in PyTorch. Available at torchvision. https://arxiv.org/pdf/1611.03530.pdf

Problem 4 - *Convolutional Neural Networks Architectures* 20 points

In this problem we will study and compare different convolutional neural network architectures. We will calculate number of parameters (weights, to be learned) and memory requirement of each network. We will also analyze inception modules and understand their design.

1. [**6 points**] VGG (Simonyan et al.) has an extremely homogeneous architecture that only performs 3x3 convolutions with stride 1 and pad 1 and 2x2 max pooling with stride 2 (and no padding) from the beginning to the end. However VGGNet is very expensive to evaluate and uses a lot more memory and parameters. Refer to VGG19 architecture on page 3 in Table 1 of the paper by Simonyan et al. You need to complete Table 1 below for calculating activation units and parameters at each layer in VGG19 (without counting biases). Its been partially filled for you.

2. The original Googlenet paper (Szegedy et al.) 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) [**3 points**] What is the general idea behind designing an inception module (parallel convolutional filters of different sizes with a pooling followed by concatenation) in a convolutional neural network?

   (b) [**4 points**] 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 1.

   (c) [**3 points**] 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 1.

   (d) [**4 points**] Based on the calculations in part (c) explain the problem with naive architecture and how dimensionality reduction architecture helps (*Hint: compare computational complexity*). How much is the computational saving?

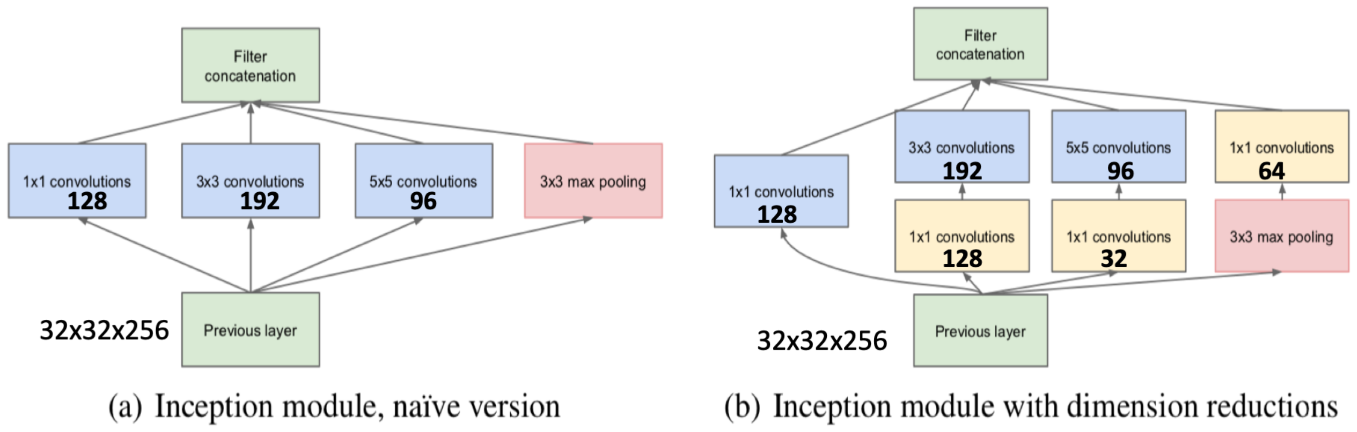

Figure 1: Two types of inception module with number of filters and input size for calculation in Question 3.4(b) and 3.4(c).

References:

- (Alexnet) Alex Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks.
  Paper available at https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- (VGG) Karen Simonyan et al. Very Deep Convolutional Networks for Large-scale Image Recognition. Paper available at https://arxiv.org/pdf/1409.1556.pdf
- (Googlenet) Christian Szegedy et al. Going deeper with convolutions.
  Paper available at https://arxiv.org/pdf/1409.4842.pdf

Problem 5 - 10 points

In a Parameter-Server (PS) based Asynchronous SGD training system, there are two learners. Assume a learner sends gradients to the PS, PS updates weights and a learner pulls the weights from the PS in zero amount of time (i.e. after learner sends gradients to the PS, it can receive updated weights from PS immediately). Let us assume that learner 1 runs at about 2.5x speed of learner 2. Learner 1 calculates gradients $g[L_1, 1]$ at second 1, $g[L_1, 2]$ at second 2, $g[L_1, 3]$ at second 3, $g[L_1, 4]$ at second 4. Learner 2 calculates gradients $g[L_2, 1]$ at second 2.5, $g[L_2, 2]$ at second 5. Updates to weights are instant once a gradient is available. Calculate the staleness (number of weight updates between reading and updating weights) of $g[L_1, 1], g[L_1, 2], g[L_1, 3] g[L_1, 4], g[L_2, 1], g[L_2, 2]$. ($g[L_i, j]$ means i-th learner's j-th calculated gradients).