Homework 3

Xuhang He

December 13, 2023

1 Question 1:

Suppose I build a binary decision tree for a dataset. The depth of the tree is k (i.e., for any given test data point, one performs at most k tests). Recall that leaf nodes in the decision tree correspond to decision regions in the feature space.

- a What is the minimum number of decision regions that are possible for such a tree? Explain your answer
- **Answer:** The minimum decision region is k+1.

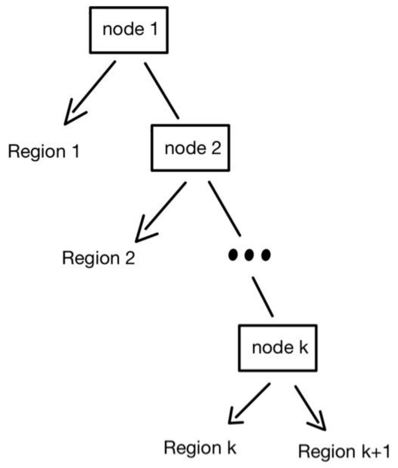

Figure 1: decision region of k+1

- b What is the maximum number of decision regions that are possible for such a tree? Explain your answer.
- **Answer:** The maximum decision region is 2^k where all the leaf nodes are the decision regions.

2 Question 2:

Suppose we are studying a language model over N tokens. For concreteness, let us imagine tokens as words in English. Represent this model by a probability distribution $P(x_1, x_2, ..., x_N)$.

- a. Write down a simplified approximation of this distribution, assuming a Markov dependency among the tokens where each token depends on the previous two tokens at most.
- **Answer:** The approximation is $P(x_1, x_2, ..., x_N) = P(x_1) \times P(x_2|x_1) \times P(x_3|x_2, x_1) \times ... \times P(x_N|x_{N-1}, x_{N-2})$.
- b. Argue that one can generate as many sentences (of length N) as needed using this simplified version, provided one can access the trigam probabilities, defined as follows: for any arbitrary triple of words (x_i, x_j, x_k) in English, the trigram probability is defined as the conditional probability $p(x_k|x_i, x_j)$.
- **Answer:** If all trigram combinations are given as well as their possibilities $p(x_k|x_i, x_j)$, then it is possible to generate as many sentences of length N as possible. First, randomly pick two English words as the initial state. Then, use the initial two words to find the third word that has the greatest probability among all other English words: $p(x_3) = p(x_3|x_1, x_2)$. Repeat the process of using the last two words x_{k-1}, x_k to guess x_{k+1} until k = N. Suppose there are M English words in the world, then $M \times M$ such sentences can be generated.
- c. Given a large enough corpus of text data (say, the entirety of Wikipedia), present a method to roughly estimate these trigram probabilities. You need not provide detailed pseudocode, just explain in words how this can be done.
- **Answer:** If x_i, x_j has never occurred in the text data, then just assign it with words that only appeared once in the text, each with a uniform probability of 1 over the number of unique words. If x_i, x_j has occurred in the text data, then for x_k, calculate $P(x_k|x_i, x_j) = \frac{numberoftrigram(x_i,x_j,x_k)}{numberofbigram(x_i,x_j))}$.

3 Question 3:

Recall from HW2 that you trained simple classifiers on the Fashion MNIST dataset. We will repeat this exercise, but with neural networks. For a quick refresher on how to load and manipulate this dataset in PyTorch, see https://github.com/chinmayhegde/ece6143-spring20/blob/master/demo08.ipynb.

- a. Load the dataset and display 10 representative images from each of the 10 classes.

- **Answer:** Please see `https://github.com/micheeeal/CSUY4563_Fall2023/blob/main/HW03/showclasses.py`.

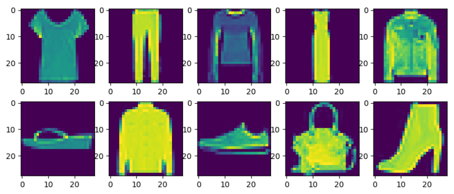

Figure 2: ten classes

- b. Implement a simple 3-layer neural network classifier for this dataset. The input is 784 features and the output is 10 classes, so the network should have three sets of weighs: 784x200, 200x200, and 200x10. Use ReLU activations for the hidden neurons.

- **Answer:** Please see `https://github.com/micheeeal/CSUY4563_Fall2023/blob/main/HW03/Neural_network.py`.

- c. Train this network for 30 epochs and plot your train and test curves. Report best test accuracy performance by tuning hyperparameters: pick learning rates (0.1, 0.5), batch size (16,64).

- **Answer:** The best hyperparameters are learning rate = 0.1 and batch size = 16. This set of hyperparameters gives the lowest final average training loss per batch of 0.147 and the final average test training loss per batch of 0.395. When learning rate = 0.5 and batch size = 16, the final average training loss and testing loss per batch are both around 1.7. The following diagrams show the change in training and testing loss with training loss shown in blue and testing loss shown in red.

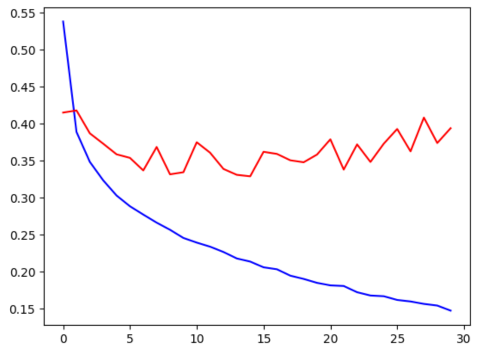

Figure 3: lr=0.1, batch size=16

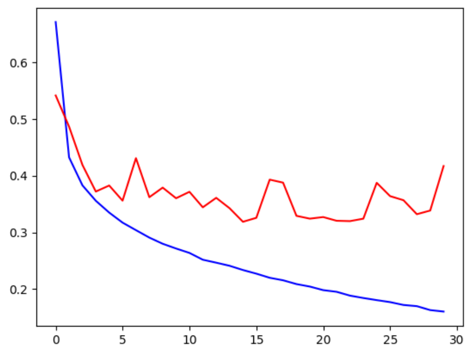

Figure 4: lr=0.1, batch size=64

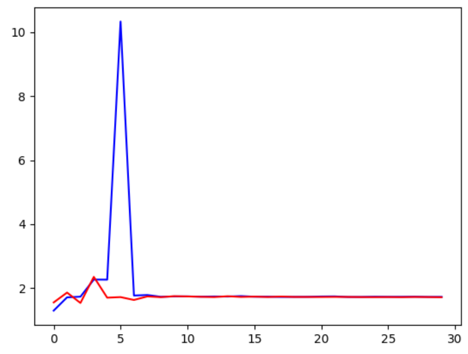

Figure 5: lr=0.5, batch size=16

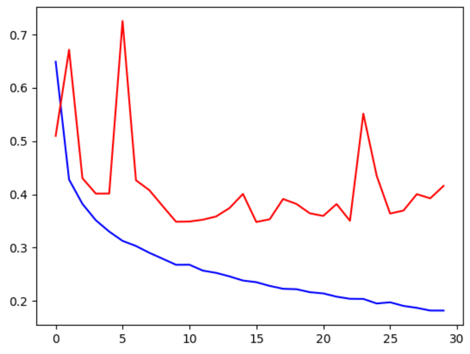

Figure 6: lr=0.5, batch size=64