COMP 432 Intro. to Machine Learning (Fall 2024)

Major Assignment #3

Due: 11:59PM, November 17th, 2024

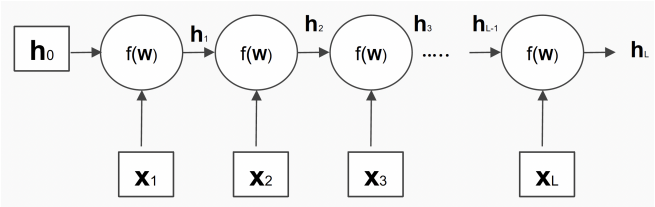
Note Your will be submitting two separate files from this assignment as follows:

(a) **One(1) .pdf file:** containing answers to Question as well as reported results from coding you develop. Include snapshots of the pieces of code you developed in the appendix.

(b) **One(1) .zip folder:** containing all developed Python codes including a README.txt file on explaining how to run your code.

Theoretical Questions

Question 1 **(20 Points)** Consider the following architecture of a simple Recurrent Neural Network (RNN):



In this network, all the inputs, hidden states, and weights are assumed to be 1D. This network is to be used for a many-to-one regression task. The network produces an output y at the last time step L, which is given as $y = f(w_{hy}h_L)$. For the following questions, assume that no nonlinearities are used (i.e. no activation functions).

(a) **(3 Points)** Write a general expression for h_t in terms of w_{in}, w_{hh}, x_t, and h_{t-1}, where t is the timestep.

(b) **(3 Points)** Given the sequential input $\mathbf{x} = [0.9, 0.8, 0.7]$, the initial hidden state $h_0 = 0$, and all weights initialized to 0.5, compute h_1, h_2, h_3, and y.

(c) **(10 Points)** You want to perform *Backpropagation Through Time (BPTT)*, which is used to update the weights of RNN networks. Given a target output y_r, Assume that you are using the function $l = \frac{1}{2}(y - y_r)^2$ to calculate the loss. The goal is to compute $\frac{\delta l}{\delta w_{in}}$, $\frac{\delta l}{\delta w_{hh}}$, and $\frac{\delta l}{\delta w_{hy}}$, and use them in the typical update rule given as $w_i \leftarrow w_i - \eta\frac{\delta l}{\delta w_i}$. Find the expressions for $\frac{\delta l}{\delta w_{in}}$, $\frac{\delta l}{\delta w_{hh}}$, and $\frac{\delta l}{\delta w_{hy}}$. Your expressions can include only the following: h_0, w_{in}, w_{hh}, w_{hy}, and x_i for $i \in [1, 2, 3]$. Show your work. (hint: for $\frac{\delta l}{\delta w_{in}}$ and $\frac{\delta l}{\delta w_{hh}}$, you need to consider the sum across all timesteps)

(d) **(4 Points)** Assume that the target for the previously given data sequence is $y_r = 0.8$, with a learning rate of $\eta = 0.1$. Calculate the updated value of each weight.

Question 2 **(10 Points)** Consider a Support Vector Machine (SVM) problem in 1D settings (with one weight and one bias). For the following questions, assume we have the dataset $D = \{(x_1, t_1), (x_2, t_2), (x_3, t_3)\} = \{(3, 1), (5, -1), (2, 1)\}$.

(a) **(5 Points)** Define the optimization problem (objective function and constraints) of SVM based on margin maximization.

(b) **(5 Points)** Solve the optimization problem using the graphical method. Explain your work.

Question 3 (15 Points) Suppose you are training a RNN to predict the next word in a sentence based on the previous words. Given the sentence: "The cat sat on the," you want the model to predict the next word, which should be "mat". The vocabulary consists of the following 8 words: {'The', 'cat', 'sat', 'on', 'the', 'mat', 'is', 'dog'}. Each word should be represented by a one-hot encoded vector based on the size of the vocabulary. The RNN has a hidden state size of 4 and processes the sequence one word at a time. The RNN cell computes the hidden state at each time step t as follows:

$$h_t = \tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$

The output is computed as:

$$y_t = \text{softmax}(W_{hy}h_t + b_y)$$

The initial hidden state h_0 is a zero vector.

You are given the following randomly initialized weights:

$$W_{hx} = \begin{bmatrix} 0.2 & -0.4 & 0.1 & 0.3 & 0.5 & 0.2 & -0.1 & 0.4 \\ 0.5 & 0.3 & -0.2 & 0.1 & -0.3 & 0.4 & 0.2 & 0.1 \\ 0.1 & 0.4 & 0.5 & 0.2 & 0.1 & -0.5 & 0.3 & 0.6 \\ 0.6 & -0.1 & -0.3 & 0.4 & 0.2 & 0.1 & 0.5 & -0.2 \end{bmatrix}$$

$$W_{hh} = \begin{bmatrix} 0.4 & -0.2 & 0.1 & 0.3 \\ 0.2 & 0.5 & -0.4 & 0.1 \\ -0.3 & 0.3 & 0.2 & 0.2 \\ 0.1 & -0.5 & 0.6 & 0.4 \end{bmatrix}$$

$$W_{hy} = \begin{bmatrix} 0.3 & 0.1 & -0.2 & 0.4 \\ 0.5 & -0.1 & 0.3 & -0.4 \\ 0.2 & 0.6 & -0.3 & 0.1 \\ 0.4 & -0.3 & 0.1 & 0.2 \\ 0.1 & 0.4 & 0.2 & -0.5 \\ 0.2 & 0.3 & -0.1 & 0.6 \\ -0.4 & 0.2 & 0.5 & 0.1 \\ 0.3 & -0.2 & 0.1 & 0.4 \end{bmatrix}$$

$$b_h = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}, \quad b_y = \begin{bmatrix} 0.05 \\ 0.1 \\ -0.05 \\ 0.15 \\ -0.1 \\ 0.05 \\ 0.2 \\ -0.15 \end{bmatrix}$$

(a) **(3 Points)** Using one-hot encoding (as per the given order of the words in the dictionary), encode each of the words in the input and output.

(b) **(5 Points)** Compute the hidden state value h_t after processing each word in the input sequence.

(c) **5 Points** Compute the final output after processing all the inputs.

(d) **(2 Points)** Based on your output, which word is the most probable to come next given the input? What does that tell you about your model?

Question 4 **(15 Points)** Consider a binary classification problem where the input data is linearly separable. Your task is to find the optimal hyperplane that separates the two classes with a maximum margin. This means the data can be perfectly divided into two classes without any misclassifications or margin violations.

The primal optimization problem for a hard-margin SVM is:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

subject to:

$$t_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, N$$

where:

- $\mathbf{x}_i$ is the input vector for the i-th training sample.
- $t_i \in \{-1, 1\}$ is the label for the i-th training sample.
- $\mathbf{w}$ is the weight vector that defines the separating hyperplane.
- b is the bias term.

Consider the following linearly separable training data:

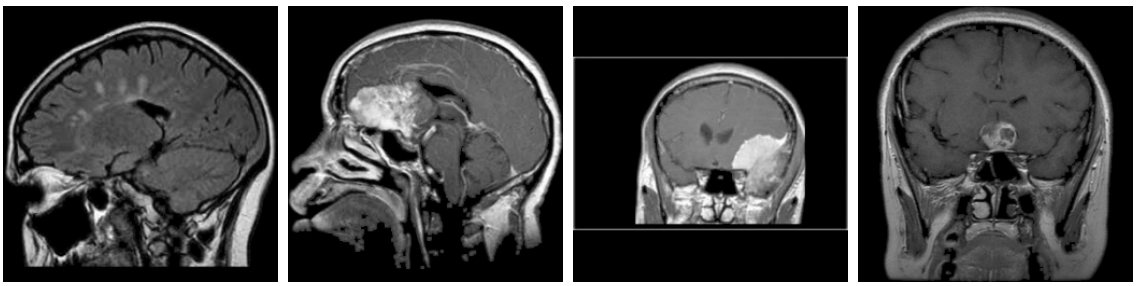
$\mathbf{x}_i$	t_i
$(1, 2)$	1
$(2, 3)$	1
$(3, 3)$	-1
$(1, 0)$	-1

(a) **(7 Points)** Write down the explicit form of the primal SVM optimization problem in terms of the weight vector $\mathbf{w}$ and bias term b, for the given dataset. Use scalar variables in your answer (do not use matrix notation) or combine the constraints into matrix form. Instead, write out each constraint explicitly.

(b) **(8 Points)** Derive the dual form of the optimization problem by introducing Lagrange multipliers with KKT constraints. Clearly list the objective function and the new constraints. Show your work.

Implementation Questions

Question 1 **(15 Points)** In this question, you will explore the concept of Transfer Learning (TL) using PyTorch. TL is a technique in which previously trained models can be used to help train new models. There are two possible ways to implemented TL, namely Fine-tuning and Feature Extraction. You can read more about these methods by referring to PyTorch documentations. For the following questions, you are to use the Brain Tumor classification dataset available on Kaggle. Sample images are shown below for the 4 different classes.



(a) Normal (b) Glioma Tumor (c) Meningioma Tumor (d) Pituitary Tumor

(a) **(3 Points)** Analyze and visualize the statistics of the dataset. Pre-process the data and prepare them for the training phase. Ensure that the images are resized to 224x224x3 and normalized. Split the data randomly into train and test sets, with a ratio of 7:3.

(b) **(6 Points)** Train a ResNet-18 model from scratch using the provided dataset for the classification task. You are free to choose the hyperparameters (batch size, learning rate, optimizer, loss function, etc).

(c) **(4 Points)** Train another ResNet-18 model using the fine-tuning TL method based on IMAGENET1K weights. You should use the same hyperparameters and same data used to train the previous ResNet-18 model.

(d) **(2 Points)** Report and compare the performance of both models in terms of training accuracy/loss and classification report. What are your conclusions?

Question 2 **(15 Points)** Build an LSTM-based model for time-series forecasting using PyTorch. Given a series of data points, the model should be able to predict the next data point. You should use the Amazon Stock dataset for this task, where the aim is to use the previous data points to predict the next stock value.

(a) **(3 Points)** We will only use the "close" column to train our model, hence you should remove the remaining columns from the dataset. Additionally, since our prediction is based on the historical trend, each data point (each row) in the dataset should be in the form of sequence -> prediction. Assume we want to use a sequence window of size 10 in this problem. Pre-process the dataset such that each sequence of 10 values is used to predict

the 11th value. You can refer to this article for some examples on how to preprocess the data into sequences. (hint: if your original dataset has 100 data points, then the preprocessed dataset would have 100-10=90 data points/sequences).

(b) **(2 Points)** Preprocess the data using the MinMaxScaler from scikit-learn. There is no need to split or shuffle the data for this question.

(c) **(7 Points)** Train a vanilla LSTM to forecast the price of amazon stock based on the last 10 values. You can refer to this article for some guidance. You are free to choose and optimize the hyperparameters (optimizer, batch size, etc), but you should use the MSE loss. Aim for better performance.

(d) **(3 Points)** Analyze the performance of your model by plotting the training loss. Using the available data, compare the original plot of the data points with the predicted plot (each data point on the predicted plot is obtained by feeding the previous 10 data points from the original data).

Question 3 **(10 Points)** Consider the lung cancer prediction dataset available on Kaggle, used for a classification task to determine if the patient has cancer or not.

(a) **(4 Points)** Preprocess the data by converting textual features into numerical values. Split the dataset into train and test sets with a ratio of 7:3.

(b) **(6 Points)** Using the sklearn library, train a Support Vector Machine (SVM) for the classification task. Study the performance on the training and testing sets using sklearn's classification report.