AI3001
Deep Learning Course
Assignment #0

Prof. Anupam Sobti

Instructions for Submission:

1. Assignment is due at 11:59:59 PM on Saturday, Feb 24, 2024.
2. Please follow the course policies.
3. A single .pdf report that contains your work. For Q1 and Q3 you can type out your responses directly on Google Docs, or via LATEX. No handwritten responses are allowed.
4. You also need to submit codes for Q2, Q4 and Q5 in the single zip folder. You can submit Python code in .ipynb format with the output clearly defined.
5. We reserve the right to take off points for not following submission instructions.

Question 1. In the classification problems the cost of a false positive (predicting positive when the true answer is negative) is different from the cost of a false negative (predicting negative when the true answer is positive). This might happen, for example, when the task is to predict the presence of a serious disease.

Let's say that the cost of a correct classification is 0, the cost of a false positive is 1, and the cost of a false negative (that is, you predict 0 when the correct answer was +1) is α. Recall that the usual logistic regression loss is:

$$\mathcal{L}_{\text{nll}}(g, y) = -(y \cdot \log g + (1 - y) \cdot \log(1 - g)).$$

(a) What is the usual loss, as a function of guess g, when the true label $y = 0$?

(b) What is the casual loss, as a function of guess g, when the true label $y = 1$?

(c) Dr Sobti suggests that using a simple modification of the usual logistic regression loss function is easy so, let us devise our loss function that penalizes false negatives α times more than the false positives.

Question 2. Implement a feedforward neural network and write the backpropagation code for training the network from scratch. We strongly recommend using numpy for all matrix/vector operations. You are not allowed to use any automatic differentiation packages. This network will be trained and tested using the Fashion-MNIST dataset. Specifically, given an input image (28 x 28 = 784 pixels) from the Fashion-MNIST dataset, the network will be trained to classify the image into 1 of 10 classes.

(a) Implement a feedforward neural network which takes images from the fashion-mnist data as input and outputs a probability distribution over the 10 classes.

(b) Implement the backpropagation algorithm with support for the following optimisation functions

i. SDG

ii. Momentum-based Gradient Descent

iii. Adam

Question 3. Consider the following vector function from $\mathbb{R}^3$ to $\mathbb{R}^3$:

$$f(x) = \begin{bmatrix} \sin(x_1 x_2 x_3) \\ \cos(x_2 + x_3) \\ \exp\left\{-\frac{1}{2}(x_3^2)\right\} \end{bmatrix}$$

(a) What is the Jacobian matrix of $f(x)$?

(b) Write the determinant of this Jacobian matrix as a function of x.

(c) Is the Jacobian a full rank matrix for all of $x \in \mathbb{R}^3$? Explain your reasoning.

Question 4. For this question, we are going to use the Stanford Dogs dataset link.

(a) Build a small CNN model consisting of 5 convolution layers. Each convolution layer would be followed by a ReLU activation and a max pooling layer.

i. What is the total number of computations done by your network? (assume m filters in each layer of size $k \times k \times k$ and n neurons in the dense layer)

ii. What is the total number of parameters in your network? (assume m filters in each layer of size $k \times k \times k$ and n neurons in the dense layer)

iii. Plot the training curve and report the test accuracy.

(b) For the previous part, now set aside 10% of your training data for hyperparameter tuning. Make sure each class is equally represented in the validation data and use the sweep feature in wandb to find the best hyperparameter configuration, as well as find the following plots.

i. accuracy v/s created plot (we would like to see the number of experiments you ran to get the best configuration)

ii. parallel co-ordinates plot

iii. correlation summary table

(c) Build a Logistic Regression model to classify the Dog Dataset and plot the training curve as well as report the test accuracy.

Question 5. We attempt to minimize some function $f(x)$ by starting with some initial x_0 in gradient descent and then iteratively modifying the parameters $x \in \mathbb{R}^n$ according to the following formula:

$$x_{t+1} \leftarrow x_t - \lambda(\nabla_x f(x_t))^T$$

for some small $\lambda \geq 0$ known as the learning rate or step size. This procedure modifies x to move in a direction proportional to the negative gradient.

Consider the simple function $f(x) = x^T Ax$ for a (constant) symmetric positive definite matrix $A \in \mathbb{R}^{n \times n}$.

(a) Implement a function $f(A, x)$ that takes as input an $n \times n$ numpy array A and a 1D array x of length n and returns the output for $f(x)$ above. Test it on the values

$$x = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

$$A = \begin{bmatrix} 3 & -1 \\ -1 & 2 \end{bmatrix}$$

.

(b) Implement a function $\text{grad}_f(A, x)$ that takes the same two arguments as above but returns $\nabla_x f(x)$ evaluated at x. Test it on the values x and A above.

(c) Now implement `grad_descent(A, x, lr, num_iters)` that takes the additional arguments `lr`, representing the learning rate λ above, and `num_iters` indicating the total number of iterations of gradient descent to perform. The function should plot the values of x_t and $f(x_t)$ at each iteration of gradient descent.