Assignment 1

Deep Learning
IIT-Hyderabad
Jan-May 2022

Max Marks: 25
Due: 23rd Feb 2022 11:59 pm

This homework is intended to cover programming exercises in the following topics:

- Pytorch, Feedforward neural networks, Backprop, Optimization in NNs

Instructions

- Please upload your submission on Google Classroom by the deadline mentioned above. Your submission should comprise of a single file (PDF/ZIP), named `<Your_Roll_No>_Assign1`, with all your solutions.
- For late submissions, 10% is deducted for each day (including weekend) late after an assignment is due. Note that each student begins the course with 12 grace days for late submission of assignments (of which atmost 7 can be used for a given submission). Late submissions will automatically use your grace days balance, if you have any left. You can see your balance on the CS5480 Marks and Grace Days document (soon to be shared).
- Please use PYTHON for the programming questions.
- Please read the department plagiarism policy. Do not engage in any form of cheating - strict penalties will be imposed for both givers and takers. Please talk to instructor or TA if you have concerns.

Learning PyTorch

Before you begin the questions in the next section, use `http://pytorch.org/tutorials/` to get started on PyTorch. A 1-hour blitz tutorial is also available at `https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html` for you to get introduced to PyTorch.

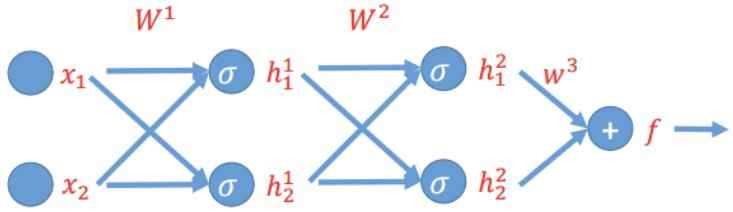
1 Questions

1. *(1+1=2 marks)* **Backpropagation:**

   (a) Consider a 3-layer network:

   $$h^1 = \sigma(W^1 x), h^2 = \sigma(W^2 h^1), f(x) =< w^3, h^2 >$$

   Compute $\frac{\partial f}{\partial W^1_{i,j}}$.



   (b) You are training a 3-layer neural network and would like to use backprop to compute the gradient of the cost function. In the backprop algorithm, one of the steps is to update $\Delta^{(2)}_{ij} := \Delta^{(2)}_{ij} + \delta^{(3)}_i * (a^{(2)})_j$ for every i, j. Can you rewrite the above equation for all weights in layer 2 in vector form? *(HINT: $\Delta^{(2)} := \Delta^{(2)} + \cdots$??)*

2. *(1+1=2 marks)* **Matrix Calculus:** Recall that the gradient of a scalar function $f : \mathbb{R}^n \to \mathbb{R}$ is the vector whose components are the partial derivatives of f.

   $$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right) \in \mathbb{R}^n$$

   where $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$. Consider the function $f(\mathbf{x}) = \log\left(\sum_{i=1}^n e^{x_i}\right)$:

   (a) Compute the partial derivatives $\frac{\partial f}{\partial x_i}, i = 1, \cdots, n$.

   (b) Based on the solution to the above problem, do you think there is a matrix calculus equivalent of the "chain rule" from calculus? Formulate this rule.

3. *(3 + 4 + 4 = 11 marks)* **Regression with linear neurons:** Take the demo code for linear regression available at `https://github.com/pytorch/examples/blob/master/regression/main.py`. This demo implements Stochastic Gradient Descent for estimating the parameters of the linear model. Your task is to familiarize yourself with every step of the code, and understand what it is doing.

   (a) Modify the simplistic implementation of Gradient Descent to use SGD class in the torch.optim package (`http://pytorch.org/docs/master/optim.html#torch.optim.SGD`). Modify the learning rate and comment on its effect.

   (b) You are provided with a toy dataset in `qn2_data.csv`. The data relates to the amount of corn produced (column 3) with respect to the amount of fertilizers and insecticides that are used (column 1 and column 2). Fit a linear regression model on this data, trained with SGD. Report the parameters of the trained model(weights[w1, w2] and the bias). Report the values of the number of corn produced with the following test set:

`test_set = torch.Tensor([[6,4],[10,5],[14,8]])`

Submit your code along with the weight, biases and the predicted values.

(c) Implement the least squares solution $\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ using the same dataset. What are the predictions for the above test set? How do they compare to the predictions of the linear neuron trained with SGD? How do the parameters compare?

Submit your code along with a report (iPython notebook highly recommended).

4. *(3+4+3=10 marks)* **MNIST Classification:** The MNIST dataset consists of 1024-dimensional inputs corresponding to pixels of a 32×32 image showing a handwritten digit from 0 to 9, and corresponding labels stating what digit the image shows. The goal is to learn a classifier that predicts the class (label) of several test-set inputs. You have been provided with a starter code (`mnist.py`) which downloads the dataset, defines a network and trains it. Your main task is to find a configuration (learning rates, mini-batch size, momentum, etc.) and corresponding meta-parameter values for an optimizer algorithm (SGD) so as to minimize the number of errors on the test set. To find out all the configuration options for the optimization algorithms, you may have to read the documentation here: `http://pytorch.org/docs/0.3.0/optim.html` (you can choose the appropriate version as required). To help you, we have provided some initial configuration options for SGD. However, more options are possible and the values provided are not necessarily optimal. With a better choice of parameters, you will converge much faster. Additionally, you can tweak other options such as the minibatch size. Answer the following questions (keep all your answers concise):

   (a) Modify the code so that it evaluates its performance on the test set after every epoch, then plot both the test loss and training loss on the same plot. Submit your code, the plot and a brief explanation of the code required to do this.

   (b) Find a configuration that works well for SGD (for the optimizer and perhaps mini-batch size and others). The classification error is the percentage of instances that are misclassified. Report your training set and test set classification error as just defined. Very briefly, explain why the solution for the final model you pick is "good". Show your code for computing the classification error, which should be short.

   (c) Attempt the same with two other optimizers, say Adagrad and L-BFGS (a 2nd-order method), instead of SGD. Explain your findings about which optimizers were easier to configure than others.

   Submit your code along with a report (iPython notebook highly recommended).

2 Practice Questions

No submission required; this is to practice for the first exam to be held in early February.

1. **Learning PyTorch:**

   (a) Assuming we have defined `t` as: `t = torch.Tensor([[1,2,3],[4,5,6],[7,8,9]])`. List 3 expressions that can replace the first line below to slice (extract) the middle column from `t`:

   `col = ···` *# extract the middle column from t*
   `print col` *# should print the 1-D tensor: 2,5,8*

(b) What is the difference between a `Tensor`, a `Storage` and a `Variable`?

2. Given the following training data/label inputs:

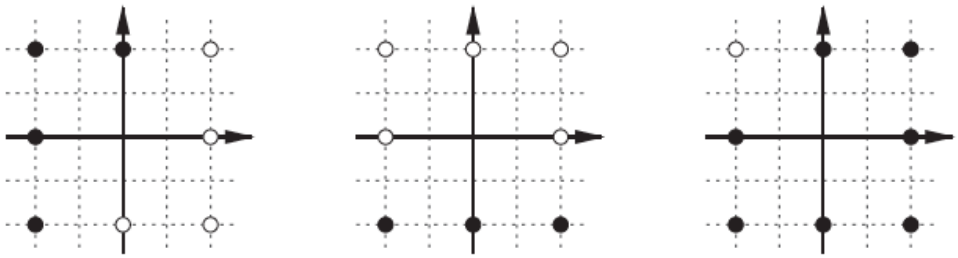
$$\{\mathbf{x}_1 = [1; 1]; d_1 = 1\}, \{\mathbf{x}_2 = [1; -1]; d_1 = -1\}$$

and that these inputs are used to train a perceptron with no bias using the LMS (Widrow-Hoff/Delta learning) rule, answer the following questions:

(a) What does the mean error surface look like? In particular, comment on its curvature along the $\mathtt{w}_1$ and $\mathtt{w}_2$ axes, as well as where the surface is centered on (the mimumum).

(b) What can you say about the eigenvalues of the Hessian of the error function?

3. Solve the three simple classification problems shown in the figure below by drawing a decision boundary. Find weight and bias values that result in single-neuron perceptrons with the chosen decision boundaries:



4. The points $(0, 0)$, $(\pi/2, \pi/2)$, and $(\pi/2, -\pi/2)$ are critical points of $f(x, y) = \sin x \sin y$.

(a) Calculate the local quadratic approximation to f at each of the three given critical points.

(b) Graph the surface $z = f(x, y)$ together with the local quadratic approximations at the three given critical points. Feel free to use any tool of your choice for this purpose.

(c) What does the concavity of the local quadratic approximation at a given point have to do with the concavity of the surface at that point?

(d) Calculate the Hessian matrix $H_f(x, y)$.

(e) Fill in the following table. (This will require you to evaluate H_f and find its eigenvalues at each of the three given critical points. You can figure out the concavity—whether up, down, or inconsistent—from the work you did for Part (c).)

Critical Point (x_0,y_0)	$H_f(x_0, y_0)$	Eigenvalues of $H_f(x_0, y_0)$	Concavity at (x_0,y_0)
$(0, 0)$			
$(\pi/2, \pi/2)$			
$(\pi/2, -\pi/2)$			

(f) Examine the table and the graph you made in Part (b). How can the eigenvalues help you classify the concavity of the surface at each critical point?