CS 599: Foundations of Deep Learning
Assignment #00001

December 17, 2024

Course Policy

Carefully read all the instructions below before you start working on the assignment, and before you make a submission.

- All Problems should be coded in Tensorflow 2 or pytorch
- Please typeset your submissions in LaTeX, give maximum explanation for each subproblems.
- Assignments are due at the end of the day at 11:59 pm on the due date given on the web Portal.
- No single line answers are accepted in the submission.
- Late assignments will suffer 50 percent loss after the first day and all loss after the second.
- All source materials must be cited. The University Academic Code of Conduct will be strictly enforced.
- We will be creating Canvas submission page for this. You have to submit python file[no ipython allowed] given in gitrepo along with your response pdf.

Problem 1: Linear Regression (1 + 1 + 1 = 3 points)

To get you to think in terms of neural architectures, we will approach the problem of estimating good regression models from the perspective of incremental learning. In other words, instead of using the normal equations to directly solve for a closed-form solution, we will search for optima (particularly minima) by iteratively calculating partial derivatives (of some cost function with respect to parameters) and taking steps down the resultant error landscape. The ideas you will develop and implement in this assignment will apply to learning the parameters of more complex computation graphs, including those that define convolutional neural networks and recurrent neural net.

We create a complete example of using linear regression to predict the paramters of the function

$$f(x) = 3x + 2 + noise$$

Given a point x we want to predict the value of y. We train the model on 10000 data pairs $(x, f(x))$. The model to learn is a linear model

$$\hat{y} = Wx + b$$

Note that, we use 'tf.GradientTape' to record the gradient with respect our trainable paramters. We use Mean Square Error(MSE) to calcuate the loss

$$g = (y - \hat{y})^2$$

Other loss function which can be used for eg L1

$$g = (y - \hat{y})$$

We use Gradient Descent to update the paramters

$$W = W - \alpha \frac{\partial g}{\partial W}$$

$$b = b - \alpha \frac{\partial g}{\partial b}$$

Things to Report

NOTE:- All submissions should use NIPS latex template.
Pdf generated from NIPS template would only be accepted rest all would lead to zero points.

- Fork repo [`https://github.com/AnkurMali/IST597_Fall2019_TF2.0`] and modify file lin_reg.py.
- Change the loss function, which loss function works better and why? Write mathematical formulation for each loss function.
- Create hybrid loss function(For eg. L1 + L2)
- Change the learning rate.
- Use patience scheduling[Whenever loss do not change, divide the learning rate by half].
- Train for longer Duration. Change the initial value for W and B. What effect it has on end result?
- Change the level of noise.
- Use various type of noise.
- Add noise in data.
- Add noise in your weights.
- Add noise in your learning rate[For all above: Scheme can be per epoch or per N epochs]
- How do these changes effect the performance?
- Do you think these changes will have the same effect (if any) on other classification problems and mathematical models?
- Plot the different result.
- Do you get the exact same results if you run the Notebook multiple times without changing any parameters? Why or why not?[Explain significance of seed].
- Use unique seed for each experiment[Note:- Convert your first name into decimal].
- Later report per epoch GPU vs CPU Time.
- Can you get an model which is robust to noise? Does model lead to faster convergence? Do you get better local minima? Is noise beneficial?
- Collect everything and report your findings.

Problem 2: Logistic Regression (2+2+1=5 points)

Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. In this you will be using Fashion Mnist which is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes[`https://github.com/zalandoresearch/fashion-mnist`]. FashionMnist is much harder than MNIST so getting accuracy in 90's is difficult. You can use whatever loss functions, optimizers, even models that you want, as long as your model is built in TensorFlow using eager execution[Remember no keras is allowed]

Things to Report

- Fork repo [`https://github.com/AnkurMali/IST597_Fall2019_TF2.0`] and modify file log_reg.py.
- TODO: This keyword means you have to implement specific section/function/formula.
- Report should contain matplotlib plots from function plot_images and plot_weights.
- Change the optimizer and report which one converges faster and which one reaches better local minima/generalizes better.[Now you can use tensorflow optimizer,but no keras]
- Train for longer epochs.
- Change Train/Val split. Report if you observe any performance drop/gain.
- Report Train/Val accuracy over time.
- Does batch size have any effect on performance.
- Report GPU vs CPU per epoch performance.
- Do Model overfit? If so why and also report measures you took to avoid overfitting.
- Compare performance with random forest and svm(you can use any built in library for only this)
- Cluster the weights for each class using any clustering mechanism. Show t-sne or k-means clusters.