IST 597: Foundations of Deep Learning **(Due: 11/06/19)**

Assignment #01000

Instructor: **Dr. Lee Giles** *TAs:* Ankur Mali , K.Zhang *Name: , PSUID:*

Created by Ankur Mali

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 1.14 or above
- Please typeset your submissions in provided LaTeXtemplate ,give maximum explanation for each sub-problems . Please include your name and PSUID with submission along with date and time on the first page.
- Assignments are due at the end of the day at 11:59 pm on the due date given on the webportal 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.
- All queries related to Assignment should have a subject line IST597:Assignment01000 Queries

Problem 1:Implementing various Normalization using basic tensorflow ops (3+3+2=8 points)

1 Batch Normalization [1]:The basic Idea

Batch normalization(BN) normalizes activations in our model across the mini-batches.For each input feature available in the model, BN then computes the mean and variance of those features within mini-batch. Then it subtracts the mean and divides the input features by mini-batch standard deviations. **So what does this do?** BN restricts the activations to have 0 mean and unit variance/std. Now let's think about few problems associated with this?

How will these approach work, when we have relu?

If increase in weight magnitude leads to better convergence. To address this BN adds two learnable parameters:First is mean of activation and second is magnitude of activation. Then BN rescales the normalized activations/post-activation and then goes on adding a constant.This ensures that expressiveness ability of neural network holds.

So what we need to implement? We will be working on forward pass and backward pass of batch normalization. You can pass parameters to gradient tape and let tensorflow handle it. But for forward pass you will be using basic tensorflow ops.

Let's get into details

x_i = input features within a mini-batch(MB) of size N

z_i = output after applying BN operations

γ and β are learnable parameters

$\mu_{MB} \leftarrow 1/N \sum_{i=1}^{N}(x_i)$ //calculate mini-batch mean

$\sigma^2_{MB} \leftarrow 1/N \sum_{i=1}^{N}(x_i - \mu_{MB})^2$ //calculate mini-batch variance

$\hat{x_i} \leftarrow (x_i - \mu_{MB})/(\sqrt{\sigma^2_{MB} + \epsilon}$ //normalize data

$z_i \leftarrow \gamma \hat{x_i} + \beta$ //Finally scale and shift.

You will be implementing above formulation for each mini-batch in your forward pass function.

2 Weight Normalization [3]

Standard way of calculating post-activation for any given input is given by: $Y = \phi(W.x + b)$ Y is output , W is weight matrix ,b is biases and x is input. Now we will do reparameterizing of each weight w in terms of paramater v and a scalar parameter g. **How to achieve this**

$w = (g/||v||)v$.

$||v||$ denotes the euclidean norm of v, and v is k-dimensional vector. If we fix the euclidean norm of w then we get $||w|| = g$.

[Note:- SGD step is now performed on new parameters v and g]

3 Layer Normalization[2]

Layer norm normalizes the inputs across the features. Whereas batch norm normalizes the input feature across mini batches.

Let's modify layer norm original equation and compare with batch norm to understand exact difference.

Batch Norm

Steps for batchnorm

Step1: $\mu_j \leftarrow 1/N \sum_{i=1}^{N}(x_{ij})$ //calculate mini-batch mean

Step2: $\sigma_j^2 \leftarrow 1/N \sum_{i=1}^{N}(x_{ij} - \mu_j)^2$ //calculate mini-batch variance

Step3: $\hat{x_{ij}} \leftarrow (x_{ij} - \mu_j)/(\sqrt{\sigma_j^2 + \epsilon})$ //normalize data

Step4: $z_{ij} \leftarrow \gamma \hat{x_{ij}} + \beta$ //Finally scale and shift.

x_{ij} is the i,jth element of input, which also represents batch and feature respectively.

Now lets look at layer norm

LayerNorm

Steps for LayerNorm

Step1: $\mu_i \leftarrow 1/N \sum_{j=1}^{N}(x_{ij})$

Step2: $\sigma_i^2 \leftarrow 1/N \sum_{j=1}^{N}(x_{ij} - \mu_i)^2$

Step3: $\hat{x_{ij}} \leftarrow (x_{ij} - \mu_i)/(\sqrt{\sigma_i^2 + \epsilon})$

Step4: $z_{ij} \leftarrow \gamma \hat{x_{ij}} + \beta$

Things to do:

- Modify the file provided[You can include functions but can't delete any]
- You will be working with Fashion MNIST or cifar10[Depends on resources you have]. I provide you the flexibility to chose the dataset for this problem[either fashion or cifar]
- You can use keras or tf.layers to declare CNN and maxpooling layer. I am giving freedom for this.
- Backward Pass must use gradient tape method.
- Create forward pass for all normalization approaches using basic tensorflow ops and formulation provided.
- You should have WeightNorm,BatchNorm and LayerNorm three different functions. Use them in your original forward pass.
- Compare Results with and without Normalization.
- Compare your normalization function with tensorflow Norm functions.Is there any difference in performance beside floating point error? If so check your backward pass.[You should get comparable gradients].Calculate the difference between your Normalization function and one with tensorflow function.
- Report your findings, based on your experiments which one is good and why?.Also explain why LayerNorm is better than batchNorm?

References

[1] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[2] Lei Ba, J., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[3] Salimans, T., and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems* (2016), pp. 901–909.