IST 597: Assignment 01000

Jooyoung Lee
Information Sciences and Technology
Pennsylvania University
920172979

1 Problem1. Implementing various Normalization using basic tensorflow ops

1. Batch Normalization

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.

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

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

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

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

$$z_i \leftarrow \gamma * \hat{x}_i + \beta \text{ //Finally scale and shift}$$

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

2. Weight Normalization

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 parameter 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.

3. Layer Normalization

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

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

$$\mu_{MB} \leftarrow 1_{\overline{N}} \sum_{j=1}^{N} x_{ij} \text{ //calculate mini-batch mean}$$

$$\sigma^2_{MB} \leftarrow 1_{\overline{N}} \sum_{j=1}^{N} (x_{ij} - \mu_{MB})^2 \text{ //calculate mini-batch variance}$$

$$\hat{x}_i \leftarrow (x_{ij} - \mu_{MB}) / \sqrt{(\sigma^2_{MB} + \epsilon)} \text{ //normalize data}$$

$$z_{ij} \leftarrow \gamma * \hat{x}_{ij} + \beta \text{ //Finally scale and shift}$$

1.1 Compare the results

In order to examine the effectiveness of different normalization methods on a CNN model using Mnist dataset, I retrieved the loss and accuracy of the model with three different normalization approaches. In terms of hyperparameters, I set minibatch size as 64, learning rate as 0.01 and hidden units as 100. Also, I implemented GradientDescentOptimizer and applied normalization methods before the activation function, Relu.

	w/o batch norm	batch norm	weight norm	layer norm
epoch 1	0.0087	0.006	0.0105	0.006
epoch 2	0.0043	0.0027	0.0058	0.0028
epoch 3	0.0034	0.0021	0.0051	0.0021
epoch 4	0.0029	0.0017	0.0046	0.0017

[Fig1] loss values using 3 different normalization functions

	w/o batch norm	batch norm	weight norm	layer norm
accuracy	94.61%	96.84%	92.22%	96.41%
total time taken	698.54s	677.64s	656.41s	693.06s

[Fig2] accuracy and total time taken reports using 3 different normalization functions

The results show that loss is minimum when using batch normalization compared to other two normalization functions. Also, the accuracy is maximum when using batch normalization as well. Even though weight normalization results in the fastest computational time, its performance seems to be weaker than with or without normalization methods.

1.2 Compare your normalization function with tensorflow Norm functions

	my batch norm	tf batch norm
accuracy	96.84%	96.44%
loss (epoch1)	0.0060	0.0059
loss (epoch2)	0.0027	0.0027
loss (epoch3)	0.0020	0.0021
loss (epoch4)	0.0017	0.0017

[Fig3] accuracy and loss values using my own batch normalization function and tensorflow function

When I compare my own batch normalization function with tensorflow batch norm functions, the loss and accuracy values are nearly the same beside floating point error. Therefore, we can conclude that there is no significant difference between using the equations given in the assignment and tensorflow norm functions.

1.3 Batch Normalization vs Layer Normalization

Based on my experiment, layer normalization offers a speedup over the baseline model without normalization, but batch normalization outperforms the other methods.