Homework 3

Statistical Machine Learning STAT 613

Rice University

Department of Statistics

Sent on: Tuesday, March 24, 2020

Due on Friday, April 10, 2020 (midnight)

Designed by Zhenwei Dai

1 Pytorch Coding Task

Batch normalization is critical when training deep neural networks. This assignment is designed to help you understand the effects of normalization when training convolutional neural networks.

Consider a convolutional neural network whose input is a n-dimensional vector $\mathbf{x} \in \mathbb{R}^n$ (we are not working with images as input but rather simple vectors). Assume each convolutional layer includes only one filter with size $= k$, followed by a ReLU activation function. Note that in this case *padding* is necessary to ensure the output of each convolutional layer is also an n-dimensional vector. For example, if the network has one convolutional layer, the network output is $f(\mathbf{x}, \mathbf{w}) = \text{relu}(\text{conv}(\mathbf{x})) = \text{relu}(\mathbf{W}\mathbf{x})$, where $\mathbf{W} \in \mathbb{R}^{n\times n}$ is a circulant matrix. Note that adding a convolutional layer to $\mathbf{x}$ is equivalent to multiplying $\mathbf{x}$ by a circulant matrix.

Normalization of a layer takes an n-dimensional vector and normalizes the mean and variance of the vector. It can be represented as $\text{norm}(\mathbf{z}) = (\mathbf{z}'_1, \mathbf{z}'_2, \cdots, \mathbf{z}'_n)$, and

$$\mathbf{z}'_i = \frac{\mathbf{z}_i - \text{mean}(\mathbf{z})}{\sqrt{\text{var}(\mathbf{z}) + \epsilon}}\gamma + \beta,$$

where $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n)$ is a n-dimensional input vector. $\text{mean}(\mathbf{z}) = \frac{1}{n}\sum_{i=1}^n \mathbf{z}_i$; $\text{var}(\mathbf{z}) = \frac{1}{n}\sum_{i=1}^n (\mathbf{z}_i - \text{mean}(\mathbf{z}))^2$. ϵ is a small constant (set $\epsilon = 0.01$ in this assignment), and γ and β are parameters of the normalization layer.

Now, with normalization layers, the one hidden layer convolutional neural network becomes $f(\mathbf{x}, \mathbf{w}) = \text{norm}(\text{relu}(\text{conv}(\mathbf{x}))) = \text{norm}(\text{relu}(\mathbf{W}\mathbf{x}))$. If there are multiple convolutional layers, the network output is $f(\mathbf{x}, \mathbf{w}) = \mathbf{W}_d \text{norm}(\text{relu}(\mathbf{W}_{d-1} \cdots \text{norm}(\text{relu}(\mathbf{W}_1 \mathbf{x}))) + \mathbf{b}$, which just replicates the one hidden layer structure. The last layer convolution layer $\mathbf{W}_d$ directly connect to the output (no relu activation or normalization layer). $\mathbf{b}$ is the shift term of the last convolution layer.

In this assignment, you need to use **Pytorch** to build the convolutional neural network described above. Important note: you do not need to run the code with a graphics card, it is fine to run it on a normal CPU. The input and output vectors can be sampled from Unif$(0, 1)$ (the network's input and output are two random uniform vectors). Set 1) input and output vector dimensions $n = 64$; 2) network depth (number of convolution layers) $d = 10$; 3) number of filters (on each convolutional layer) $f = 1$, filter size $n = 15$, and stride $s = 1$. Your network output should have the form $f(\mathbf{x}, \mathbf{w}) = \text{norm}(\text{relu} \cdots \mathbf{W}_2 \text{norm}(\text{relu}(\mathbf{W}_1 \mathbf{x})))$ where $\mathbf{W}_i \in \mathbb{R}^{64 \times 64}$.

Part i: Use gradient descent to train the network. Can you achieve zero-training error? Why? Please plot the change of the training error vs iteration steps.

Part ii: Now, remove the normalization layers and use gradient descent for training. What do you observe? Plot the change of training error with iteration steps.

Part iii: Compare the change in training error in parts i and ii, what's your conclusion about the effects of normalization of layers?

Part iv: Set $\gamma = 1$, can you still achieve zero-training error?

Part v: On part iv, remove the activation functions (remove ReLU functions). Is the removal of activation functions affecting your conclusions about the effect of normalization of layers?

Part vi: (Open question) Could you explain why the normalization layers are critical in training our simple convolutional neural network?

Final note: Some terms in this assignment were not covered in class. In case of doubt please send an email to Zhenwei Dai: daizwhao@gmail.com