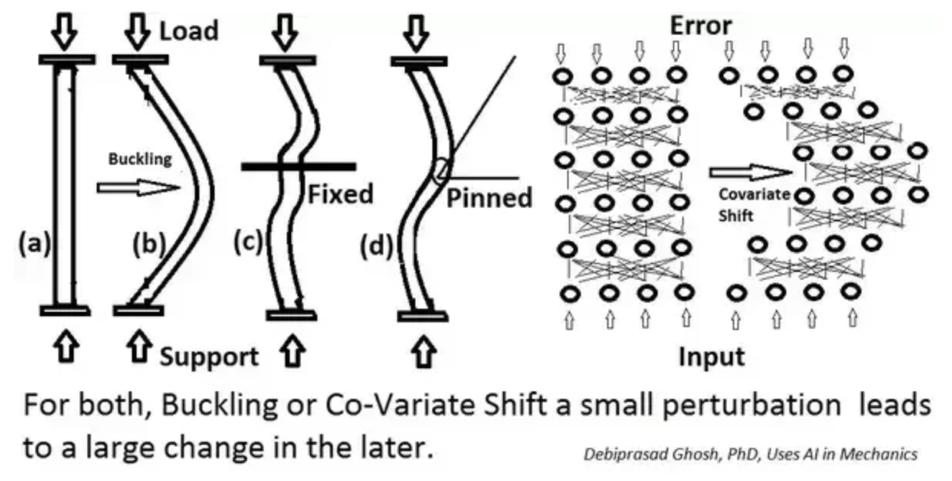
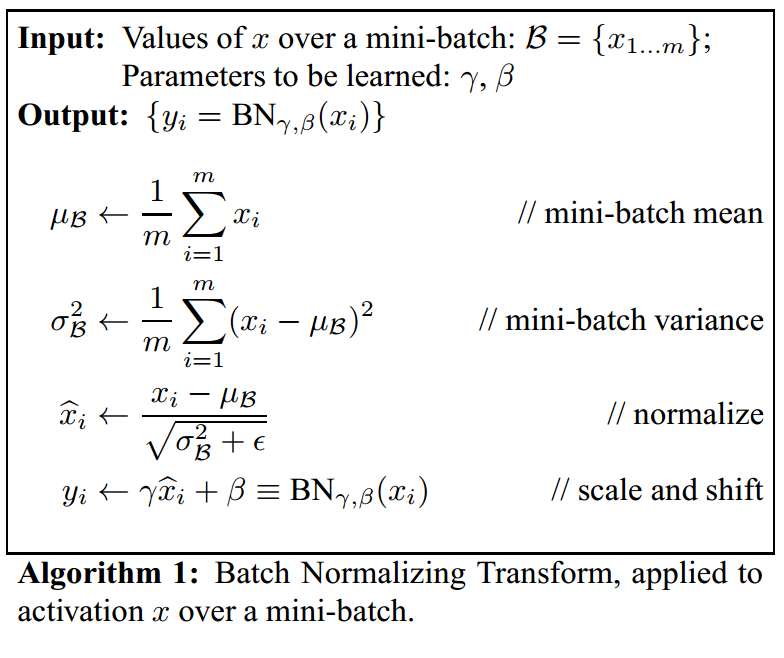
Batch normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Jo MinKi

1. Introduction   
   This is the famous batch norm. Deep learning was having unsolved annoying problem which called vanishing gradient. When the network use sigmoid or tanh like function as a activation function, the gradient value gets too small so you can’t train the network well. For this problem, people started to use ReLU function as a activation function but this is just temporary solution, not the fundamental one. And so do drop-out or regularization.   
   Then 2015, two big solution appear for this problem. First one is residual network from Microsoft and I will discuss about this at the later paper summary. Second one is BN, Batch Normalization. This paper published at 2015 but now 2 years later, people are almost thinking that BN is the truth that older than time. (Of course, I’m exaggerating but the fact is that in BN is now accepted as a general knowledge in this field. Over 80% of research are using this)
2. These days in deep learning field, 32~256 size of SGD (stochastic gradient descent) are used a lot, but this way needs detail modification of hyper parameter. And controlling learning rate is also very important, because the inputs for current layer are influenced by every former layer. This phenomenon is called ‘Covariate Shift’. This problem is explained well at Quara (https://www.quora.com/Why-does-batch-normalization-help)  
   When the buckling or covariate shift occur, we should recover the buckle by doing a process like (c) or (d). For example, Batch normalization or whitening.  
   1. Whitening   
      The whitening is that setting (or projecting) each input to so the set of data get normalized. But only naïve whitening process progress independently without influenced by backpropagation, so it could prone to monotone increasing for some certain parameter when the loss function never changes during the whitening. And controlling the mean and variance manually is necessary but inefficient. To make this clear, BN comes up.
   2. Batch Normalization.   
      If this is a naïve way, we can normalize inputs about total data, but since we are using the mini-batch SGD, we should normalize about the batch. So we should try to select the batch that represent the total data as possible.
3. Normalization via Mini-Batch statistics  
   BN has two variable which will be trained, scale and shift .   
   BN algorithm is placed right before the activation function layer, inside the network. So this process included in the backpropagation. The derivative of back propagation is described precisely in following website.   
   <https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html>  
   BN has different way for training and testing.   
   1. Training and Inference with batch normalized network   
      When BN applied for training, calculate for each mini-batch and store the value. When BN applied for testing, it uses the average value of total that already calculated before.
4. Benefit of this process  
   Paper claims that: For current (2015) deep network, if the running rate gets too high then gradient gets explode or vanish or fall into bad local optima. Usually the reason of this is the scale of parameter, and BN can solve this problem. The scale of parameter can ignored during the training.
5. Why it works?   
   According to michael’s brief explanation, network trained using some cat data and slightly different looking cat data will be putted at the inference stage. If we don’t do anything on this, the later data will get similar looking distribution but having different mean and var. To make it right, we should shift the mean and var so it have same env. BN does this process.

Question.

1. Detail about the whitening – setting it to does not make sense.
2. Should we try to select the batch so that represent the total data? Wasn’t it just random select to put the stochastic factor?