100

1.0: Introduction

In the field of supervised learning, a mathematical model is fit to a training dataset using various optimization schemes such that the model can be used to make predictions on unseen data. However, due to the nature of most training data, optimization is often made more difficult. In many multivariate learning problems, the input features are often on completely different scales. This leads to a highly eccentric cost function where gradients with respect to certain weights are much steeper than with respect to other weights. This requires learning rates to be very small in order not to overshoot in any dimension.

To solve this problem, many feature scaling techniques were developed to ensure that all input features are roughly on the same scale. One example is **standardization** where raw features are processed by subtracting by the mean of the data and scaling by the standard deviation:

$$\hat{x}_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

Here, each training feature j of training example i is standardized separately, where $\mu_j = \frac{1}{M}\sum_i x_j^{(i)}$ and $\sigma_j^2 = \frac{1}{M}\sum_i (x_j^{(i)} - \mu_j)^2$. Note that M represents the number of examples in the training set. This method is often considered to be a sufficient solution for linear models to mitigate highly eccentric cost functions. However, standardization is much less effective for neural networks.

To understand its shortcomings when applied to neural networks, consider the following example: suppose we model the linear transformation of a single layer within the NN:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

We may assume that $\mathbf{x} \sim \mathcal{N}(0, 1)$, however this by no means guarantee that $\mathbf{z} \sim \mathcal{N}(0, 1)$. This is problematic because $\mathbf{z}$ (after passing it through a nonlinearity) becomes the input to the next layer. A non-standardized $\mathbf{z}$ introduces different behaviours on the succeeding layer's inputs depending on the nonlinearity used – ReLU functions will preserve all features of $\mathbf{z}$ that are positive, of which features could be on vastly different scales, and Sigmoid and Tanh functions will saturate on certain features of $\mathbf{z}$ if the magnitudes are large.

1.1: Batch Normalization

In the 2015 paper "*Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*" (S. Ioffe, C. Szegedy, 2015), the researchers propose a method to standardize the weighted sum of each layer, ensuring that their distributions remain stable during training. This elimination of "*Internal Covariate Shift*" allows the use of much larger learning rates without the gradients diverging, thus greatly accelerating convergence.

Since Batch Normalization (BN) is applied layer-wise in a neural network, there needs to be a way to backpropagate through each BN layer, which was not needed in prior linear models using input feature standardization. While the aforementioned paper presented the equations for the backpropagated gradients, in this paper we will discuss the derivation of backpropagation in greater detail. This paper will also present qualitative and quantitative differences between experiments carried out on neural networks with and without BN.

2.0: Formulation

Batch Normalization applies feature standardization to the weighted sum of each layer, which then the normalized outputs are fed into the nonlinearity. A concrete description is shown below:

$$\mathbf{z}^{(i)} = \mathbf{W}\mathbf{x}^{(i)}$$

$$\hat{\mathbf{y}}^{(i)} = \mathrm{BN}(\mathbf{z}^{(i)})$$

$$\mathbf{a}^{(i)} = f(\hat{\mathbf{y}}^{(i)}) \tag{1}$$

In the above equations, we use $\mathbf{x}$ to denote the inputs to the layer, $\mathbf{z}$ to denote the output of the weighted sum, $\hat{\mathbf{y}}$ to denote the standardized weighted sum, $\mathbf{a}$ to denote the nonlinearity output, and (i) represents index of the training element. It's worth mentioning that a common area of contention is the placement of BN with respect to the activation function. This topic by itself can form an entire area of research, so for this paper we will stick with the implementation of the original 2015 paper, placing BN **before** the nonlinearity.

Applying feature standardization on the weighted sum, we get the following:

$$\hat{\mathbf{z}}^{(i)} = \frac{\mathbf{z}^{(i)} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \tag{2}$$

Note that we add a small ϵ in the denominator for numeric stability. Here, $\boldsymbol{\mu} = \mathrm{E}[\mathbf{z}]$ and $\boldsymbol{\sigma}^2 = \mathrm{Var}[\mathbf{z}]$. BN

computes the standardization of every example within a batch, so we should specify that $\mathbf{z}$ represents the output of the weighted sum with dimensions $M_{\text{batch}} \times N_x$, Where M_{batch} represents the batch size and N_x represents the number of features in $\mathbf{x}$.

Once we get the standardized output $\hat{\mathbf{z}}^{(i)}$, we observe that directly using this result can limit the hypothesis capacity of the neural network. Suppose we compare two weighted sums: $\mathbf{z}^{(i)} = \mathbf{W}\mathbf{x}^{(i)}$ and $\mathbf{z}'^{(i)} = \mathbf{W}\mathbf{x}^{(i)} + \mathbf{b}$. We can compute the means...

$$\boldsymbol{\mu} = \mathrm{E}[\mathbf{z}]$$

$$\boldsymbol{\mu}' = \mathrm{E}[\mathbf{z}'] = \mathrm{E}[\mathbf{z} + \mathbf{b}] = \boldsymbol{\mu} + \mathbf{b}$$

... As well as the variances:

$$\boldsymbol{\sigma}^2 = \mathrm{E}[\mathbf{z} - \boldsymbol{\mu}]^2$$

$$\boldsymbol{\sigma'}^2 = \mathrm{E}[\mathbf{z}' - \boldsymbol{\mu'}]^2$$

$$= \mathrm{E}[\mathbf{z} + \mathbf{b} - \boldsymbol{\mu} - \mathbf{b}]^2$$

$$= \boldsymbol{\sigma}^2$$

We can then show that:

$$\hat{\mathbf{z}}'^{(i)} = \frac{\mathbf{z}'^{(i)} - \boldsymbol{\mu}'}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}$$

$$= \frac{\mathbf{z}^{(i)} + \mathbf{b} - \boldsymbol{\mu} - \mathbf{b}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}$$

$$= \hat{\mathbf{z}}^{(i)}$$

Here we observe that standardizing the weighted sum will ignore the bias term. BN can be made more flexible by scaling the normalized sum and reintroducing the bias term:

$$\hat{y}_j^{(i)} = \gamma_j \hat{z}_j^{(i)} + \beta_j$$

$$\hat{\mathbf{y}}^{(i)} = \boldsymbol{\gamma} \odot \hat{\mathbf{z}}^{(i)} + \boldsymbol{\beta} \tag{3}$$

The rescaling factor $\boldsymbol{\gamma}$ and bias term $\boldsymbol{\beta}$ are parameters learnable through backpropagation. One example of the improved flexibility is the ability to learn the identity function. Suppose $\boldsymbol{\gamma} = \sqrt{\boldsymbol{\sigma}^2 + \epsilon}$ and $\boldsymbol{\beta} = \boldsymbol{\mu}$, then we observe that $\hat{\mathbf{y}}^{(i)} = \mathbf{z}^{(i)}$.

2.1: Backpropagation

In the beginning of **section 2.0**, we formulated the order of which operations are performed in forward propagation through a single layer (1). Backpropagation involves propagating the gradients in the reverse order, which requires computing several gradients of BN. This includes calculating $\frac{\partial J}{\partial \mathbf{z}^{(i)}}$ for the i^{th} input to BN, as well as $\frac{\partial J}{\partial \boldsymbol{\gamma}}, \frac{\partial J}{\partial \boldsymbol{\beta}}$ to learn the scaling and shifting parameters.

We will refer extensively to *Figure 2.1.0* in the appendix to explain the process. This dependence graph shows the relations between each variable. For example, J directly dependent on all $\hat{\mathbf{y}}^{(i)}, 1 \leq i \leq M$. The three bolded vectors points to all variables that $\mathbf{z}^{(i)}$ has direct contributions to, namely $\boldsymbol{\mu}, \boldsymbol{\sigma}^2$, and $\hat{\mathbf{z}}^{(i)}$. Therefore, we can formulate the gradients from here. Note that for this portion of the paper, all all multiplication operations are assumed to be pointwise rather than matrix, in order to reduce clutter. Also, it's worth mentioning that $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are intentionally left out of this particular diagram for the same reason.

$$\frac{\partial J}{\partial \mathbf{z}^{(i)}} = \frac{\partial J}{\partial \hat{\mathbf{z}}^{(i)}} \frac{\partial \hat{\mathbf{z}}^{(i)}}{\partial \mathbf{z}^{(i)}} + \frac{\partial J}{\partial \boldsymbol{\sigma}^2} \frac{\partial \boldsymbol{\sigma}^2}{\partial \mathbf{z}^{(i)}} + \frac{\partial J}{\partial \boldsymbol{\mu}} \frac{\partial \boldsymbol{\mu}}{\partial \mathbf{z}^{(i)}} \tag{4}$$

Starting from the first term, we first compute $\frac{\partial J}{\partial \hat{\mathbf{z}}^{(i)}}$:

$$\frac{\partial J}{\partial \hat{\mathbf{z}}^{(i)}} = \frac{\partial J}{\partial \hat{\mathbf{y}}^{(i)}} \frac{\partial \hat{\mathbf{y}}^{(i)}}{\partial \hat{\mathbf{z}}^{(i)}}$$

Referring to equation (3),

$$\frac{\partial \hat{\mathbf{y}}^{(i)}}{\partial \hat{\mathbf{z}}^{(i)}} = \boldsymbol{\gamma}$$

Therefore,

$$\frac{\partial J}{\partial \hat{\mathbf{z}}^{(i)}} = \frac{\partial J}{\partial \hat{\mathbf{y}}^{(i)}} \boldsymbol{\gamma}$$

Referring to equation (2), we can compute $\frac{\partial \hat{\mathbf{z}}^{(i)}}{\partial \mathbf{z}^{(i)}}$:

$$\frac{\partial \hat{\mathbf{z}}^{(i)}}{\partial \mathbf{z}^{(i)}} = \frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}$$

The first term of equation (4) can be put together:

$$\frac{\partial J}{\partial \hat{\mathbf{z}}^{(i)}} \frac{\partial \hat{\mathbf{z}}^{(i)}}{\partial \mathbf{z}^{(i)}} = \frac{\partial J}{\partial \hat{\mathbf{y}}^{(i)}} \boldsymbol{\gamma} \frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \tag{5}$$

Moving onto the second term, we compute $\frac{\partial J}{\partial \boldsymbol{\sigma}^2}$. Note that J's dependency on $\boldsymbol{\sigma}^2$ is split among many paths, so

the gradient through each path must be accounted for.

$$\frac{\partial J}{\partial \boldsymbol{\sigma}^2} = \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \frac{\partial \hat{\mathbf{y}}^{(m)}}{\partial \hat{\mathbf{z}}^{(m)}} \frac{\partial \hat{\mathbf{z}}^{(m)}}{\partial \boldsymbol{\sigma}^2}$$

Recall that $\frac{\partial \hat{\mathbf{y}}^{(m)}}{\partial \hat{\mathbf{z}}^{(m)}} = \boldsymbol{\gamma}$. Now we calculate $\frac{\partial \hat{\mathbf{z}}^{(m)}}{\partial \boldsymbol{\sigma}^2}$, referring to equation (2):

$$\frac{\partial \hat{\mathbf{z}}^{(m)}}{\partial \boldsymbol{\sigma}^2} = -\frac{1}{2}(\mathbf{z}^{(m)} - \boldsymbol{\mu})(\boldsymbol{\sigma}^2 + \epsilon)^{-3/2}$$

Now we can complete $\frac{\partial J}{\partial \boldsymbol{\sigma}^2}$:

$$\frac{\partial J}{\partial \boldsymbol{\sigma}^2} = -\frac{1}{2} \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma}(\mathbf{z}^{(m)} - \boldsymbol{\mu})(\boldsymbol{\sigma}^2 + \epsilon)^{-3/2} \tag{6}$$

We will now compute the second half of the second term in equation (4), namely $\frac{\partial \boldsymbol{\sigma}^2}{\partial \mathbf{z}^{(i)}}$. From the dependencies graph we note that there are two paths that lead from $\mathbf{z}^{(i)}$ to $\boldsymbol{\sigma}^2$, so the gradient must account for both contributions. The point-wise variance $\boldsymbol{\sigma}^2$ is represented by the following equation:

$$\boldsymbol{\sigma}^2 = \frac{1}{M} \sum_{m=1}^{M} (\mathbf{z}^{(m)} - \boldsymbol{\mu})^2 \tag{7}$$

And below is the gradient $\frac{\partial \boldsymbol{\sigma}^2}{\partial \mathbf{z}^{(i)}}$:

$$\frac{\partial \boldsymbol{\sigma}^2}{\partial \mathbf{z}^{(i)}} = \left[\frac{\partial \boldsymbol{\sigma}^2}{\partial \mathbf{z}^{(i)}}\right]_{\text{Direct}} + \frac{\partial \boldsymbol{\sigma}^2}{\partial \boldsymbol{\mu}} \frac{\partial \boldsymbol{\mu}}{\partial \mathbf{z}^{(i)}} \tag{8}$$

Here $\left[\frac{\partial \boldsymbol{\sigma}^2}{\partial \mathbf{z}^{(i)}}\right]_{\text{Direct}}$ is the gradient resulting from the direct contribution of $\mathbf{z}^{(i)}$ to $\boldsymbol{\sigma}^2$, and the second term represents the contribution of $\mathbf{z}^{(i)}$ chained through $\boldsymbol{\mu}$. First, we compute $\left[\frac{\partial \boldsymbol{\sigma}^2}{\partial \mathbf{z}^{(i)}}\right]_{\text{Direct}}$ from equation (7):

$$\left[\frac{\partial \boldsymbol{\sigma}^2}{\partial \mathbf{z}^{(i)}}\right]_{\text{Direct}} = \frac{2}{M}(\mathbf{z}^{(i)} - \boldsymbol{\mu})$$

Now we can compute the second term of equation (8), starting with $\frac{\partial \boldsymbol{\sigma}^2}{\partial \boldsymbol{\mu}}$:

$$\frac{\partial \boldsymbol{\sigma}^2}{\partial \boldsymbol{\mu}} = -\frac{2}{M} \sum_{m=1}^{M} (\mathbf{z}^{(m)} - \boldsymbol{\mu})$$

Then, we compute $\frac{\partial \boldsymbol{\mu}}{\partial \mathbf{z}^{(i)}}$, knowing that the pointwise mean $\boldsymbol{\mu}$ is represented by the equation $\boldsymbol{\mu} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{z}^{(m)}$:

$$\frac{\partial \boldsymbol{\mu}}{\partial \mathbf{z}^{(i)}} = \frac{1}{M} \tag{9}$$

Now, we can complete equation (8) by piecing together the gradients we just calculated:

$$\frac{\partial \boldsymbol{\sigma}^2}{\partial \mathbf{z}^{(i)}} = \frac{2}{M}(\mathbf{z}^{(i)} - \boldsymbol{\mu}) - \frac{2}{M^2} \sum_{m=1}^{M} (\mathbf{z}^{(m)} - \boldsymbol{\mu})$$

Combining equation (8), we complete the second term of equation (4). In other words, we computed the change in cost J with respect of $\mathbf{z}^{(i)}$ chaining through $\boldsymbol{\sigma}^2$.

$$\frac{\partial J}{\partial \boldsymbol{\sigma}^2} \frac{\partial \boldsymbol{\sigma}^2}{\partial \mathbf{z}^{(i)}} = \left[-\frac{1}{2} \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma}(\mathbf{z}^{(m)} - \boldsymbol{\mu})(\boldsymbol{\sigma}^2 + \epsilon)^{-3/2} \right] \left[\frac{2}{M}(\mathbf{z}^{(i)} - \boldsymbol{\mu}) - \frac{2}{M^2} \sum_{m=1}^{M} (\mathbf{z}^{(m)} - \boldsymbol{\mu}) \right]$$

$$= \frac{2}{M}(\mathbf{z}^{(i)} - \boldsymbol{\mu}) \left[-\frac{1}{2} \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma}(\mathbf{z}^{(m)} - \boldsymbol{\mu})(\boldsymbol{\sigma}^2 + \epsilon)^{-3/2} \right]$$

$$- \frac{2}{M^2} \sum_{m=1}^{M} (\mathbf{z}^{(m)} - \boldsymbol{\mu}) \left[-\frac{1}{2} \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma}(\mathbf{z}^{(m)} - \boldsymbol{\mu})(\boldsymbol{\sigma}^2 + \epsilon)^{-3/2} \right]$$

Here we observe that $(\boldsymbol{\sigma}^2 + \epsilon)^{-3/2}$ is invariant with respect to m, so we can factor that out of the sums. We also observe that $(\boldsymbol{\sigma}^2 + \epsilon)^{-3/2} = \frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}$. Taking this into consideration, we continue the simplification:

$$= \frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \left[-\frac{1}{M} \frac{(\mathbf{z}^{(i)} - \boldsymbol{\mu})}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \left[\sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma} \frac{(\mathbf{z}^{(m)} - \boldsymbol{\mu})}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \right] + \frac{1}{M^2} \sum_{m=1}^{M} \frac{(\mathbf{z}^{(m)} - \boldsymbol{\mu})}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \left[\sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma} \frac{(\mathbf{z}^{(m)} - \boldsymbol{\mu})}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \right] \right]$$

Here, we notice that $\hat{\mathbf{z}}^{(i)} = \frac{\mathbf{z}^{(i)} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}$, so we can substitute that in, completing the simplification.

$$\frac{\partial J}{\partial \boldsymbol{\sigma}^2} \frac{\partial \boldsymbol{\sigma}^2}{\partial \mathbf{z}^{(i)}} = \frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \left[-\frac{1}{M} \hat{\mathbf{z}}^{(i)} \left[\sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma} \hat{\mathbf{z}}^{(m)} \right] + \frac{1}{M^2} \sum_{m=1}^{M} \hat{\mathbf{z}}^{(m)} \left[\sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma} \hat{\mathbf{z}}^{(m)} \right] \right] \tag{10}$$

Finally, we calculate the final term of equation (4). Starting off with $\frac{\partial J}{\partial \boldsymbol{\mu}}$, we observe that $\boldsymbol{\mu}$ is a dependency of $\hat{\mathbf{z}}^{(1)}, ..., \hat{\mathbf{z}}^{(M)}$ as well as $\boldsymbol{\sigma}^2$. Note that we have already accounted for the contribution of $\boldsymbol{\mu}$ to J through $\boldsymbol{\sigma}^2$ in

equation (8), so we do not need to compute that a second time.

$$\frac{\partial J}{\partial \boldsymbol{\mu}} = \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma} \frac{\partial \hat{\mathbf{z}}^{(m)}}{\partial \boldsymbol{\mu}}$$

Next, we calculate $\frac{\partial \hat{\mathbf{z}}^{(m)}}{\partial \boldsymbol{\mu}}$ referencing equation (2):

$$\frac{\partial \hat{\mathbf{z}}^{(m)}}{\partial \boldsymbol{\mu}} = -\frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}$$

Thus, we get the full equation for $\frac{\partial J}{\partial \boldsymbol{\mu}}$:

$$\frac{\partial J}{\partial \boldsymbol{\mu}} = -\frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma}$$

The latter half of the third term $\frac{\partial \boldsymbol{\mu}}{\partial \mathbf{z}^{(i)}}$ is already calculated in equation (9), so we will reuse that calculation. The last term of equation (4) is as follows:

$$\frac{\partial J}{\partial \boldsymbol{\mu}} \frac{\partial \boldsymbol{\mu}}{\partial \mathbf{z}^{(i)}} = -\frac{1}{M} \frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma} \tag{11}$$

putting together $\frac{\partial J}{\partial \mathbf{z}^{(i)}}$, we sum together equations (5), (10), and (11) and simplify:

$$\begin{aligned}
\frac{\partial J}{\partial \mathbf{z}^{(i)}} &= \frac{\partial J}{\partial \hat{\mathbf{y}}^{(i)}} \boldsymbol{\gamma} \frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \\
&\quad + \frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \left[-\frac{1}{M} \hat{\mathbf{z}}^{(i)} \left[\sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma} \hat{\mathbf{z}}^{(m)} \right] \right. \\
&\quad \left. + \frac{1}{M^2} \sum_{m=1}^{M} \hat{\mathbf{z}}^{(m)} \left[\sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma} \hat{\mathbf{z}}^{(m)} \right] \right] \\
&\quad - \frac{1}{M} \frac{1}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma} \\
&= \frac{1}{M\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \left(M \frac{\partial J}{\partial \hat{\mathbf{y}}^{(i)}} \boldsymbol{\gamma} \right. \\
&\quad + \left[\sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma} \hat{\mathbf{z}}^{(m)} \right] \left[\left[\frac{1}{M} \sum_{m=1}^{M} \hat{\mathbf{z}}^{(m)} \right] - \hat{\mathbf{z}}^{(i)} \right] \\
&\quad \left. - \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \boldsymbol{\gamma} \right)
\end{aligned} \tag{12}$$

This gradient calculation for $\frac{\partial J}{\partial \mathbf{z}^{(i)}}$ is required to propagate gradients through the BN layer to earlier layers. There are two other gradients to calculate, namely $\frac{\partial J}{\partial \boldsymbol{\gamma}}$ and $\frac{\partial J}{\partial \boldsymbol{\beta}}$. Luckily, these two gradients are much easier to compute, and do not require pages of derivation. Referring to *Figure 2.1.1* in the appendix (which is modelled after equation (3)), scaling and shifting parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are dependencies of every $\hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)}$, therefore when computing these gradients, we need to sum every contribution:

$$\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{\gamma}} &= \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \frac{\partial \hat{\mathbf{y}}^{(m)}}{\partial \boldsymbol{\gamma}} \\
&= \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \hat{\mathbf{z}}^{(m)}
\end{aligned} \tag{13}$$

$$\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{\beta}} &= \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}} \frac{\partial \hat{\mathbf{y}}^{(m)}}{\partial \boldsymbol{\beta}} \\
&= \sum_{m=1}^{M} \frac{\partial J}{\partial \hat{\mathbf{y}}^{(m)}}
\end{aligned} \tag{14}$$

2.2: Inference

Batch Normalization learns the scaling and shifting factors $\boldsymbol{\gamma}, \boldsymbol{\beta}$ during the training process and uses the learned parameters at inference time. While these are the only parameters in BN that are learned via the backpropagation process, these are not the only parameters carried forward from training. During the inference process, we use the population mean and variance of the training set for the standardization calculations.

$$\hat{\mathbf{z}}^{(i)} = \frac{\mathbf{z}^{(i)} - \mathrm{E}[\mathbf{z}_{train}]}{\sqrt{\mathrm{Var}[\mathbf{z}_{train}] + \epsilon}} \tag{15}$$

The original papers mention why the population statistics were used rather than the inference mini-batch statistics – we want the inference output to depend only on the input, rather than other inputs in the mini-batch. This is especially apparent if mini-batch sample is small. The sample statistics of batch sizes smaller than 30 can vary wildly from the population statistics. There are also applications of inference where the mini-batch size of 1 is desired. In these cases, sample variance is unobtainable.

The population mean and variance are computed at train time, where an exponentially weighted average of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are kept. The exponentially weighted average of $\boldsymbol{\mu}$ is shown below:

$$\mathrm{E}[\mathbf{z}_{train}]_t = \alpha \mathrm{E}[\mathbf{z}_{train}]_{t-1} + (1 - \alpha)\boldsymbol{\mu}_t$$

We chose $\alpha = 0.97$, which roughly translates to averaging over the last 30 sample means. The same is done to keep track of the second moment.

3.0: Batch Normalization in Practice

In the previous section, we went through pages of derivations to formulate the forward propagation, backpropagation, and inference behaviours of Batch Norm. This section goes through the experimental implementation of BN, discussing about training performance, validation and testing performance, as well as inference computational penalties. All metrics will be compared to the same fully-connected neural network without BN.

3.1: Testing Methodology

Firstly it's important to talk about the dataset used to conduct the tests. The Higgs Dataset (Baldi, P., P. Sadowski, and D. Whiteson, 2014) is produced by Monte Carlo simulations of particles in a particle accelerator. The dataset contains 11 million datapoints with 28 features each, as well as a binary task representing whether the signal produced a Higgs boson, or if its simply a background signal. The full Higgs Dataset contains 11 million data points and takes up 7.5 GB of storage, however due to the computational deficiencies of training the model on CPU, we took a subset of the full dataset containing only 495,000 data points.

We implemented two 5-layer neural networks, one with batch normalization, and one without as a control. Other than the BN layers, the model architecture are identical, with 4 hidden layers with 1000 neurons each, terminating with a binary classifier. For the full model diagrams, refer to *Figure 3.1.0* in the appendix.

The training heuristics are identical for each model. The hyperparameter search process used a genetic algorithm (M. Gen, R. Cheng, 2000) to find the best performing hyperparameters for each model. While the hyperparameters will not be identical between the control and batch normalized models, we are still subjecting both models to the same heuristics, thus both models are given equal opportunities to search for ideal hyperparameters that maximize validation accuracy.

For each generation, the genetic algorithm trains 10 models with randomly selected hyperparameters within a set of bounds, and sorts the models by validation accuracy. The two hyperparameters being optimized are the L2 regularization coefficient and the learning rate. Then, the top 3 models within each generation are selected and using gaussian mutation, the bounds of each hyperparameter are updated. We train for 10 generations total, with 10 models per generation, where each model is trained for 10 epochs. Below is the hyperparameter search algorithm:

x.0: References

Backpropagation website: https://kevinzakka.github.io/2016/09/14/batch_normalization/
Original Batch Norm Paper: https://arxiv.org/pdf/1502.03167.pdf
Batch Norm Explained Visually: https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739:
Placement of Batch Norm: https://arxiv.org/pdf/1912.04259.pdf
Higgs Dataset: Baldi, P., P. Sadowski, and D. Whiteson. "Searching for Exotic Particles in High-energy Physics with Deep Learning." Nature Communications 5 (July 2, 2014).
Genetic Algorithms (page 34, gaussian mutation): https://books.google.ca/books?hl=en&lr=&id=U7MuV1q6P1oC&oi=fnd&pg=PR13&ots=51xsEryrpq&sig=y3vgWu0h7DZ4CGBcRlhXdc2ClCE&redir_esc=y#v=onepage&q&f=false

x.0: Appendix

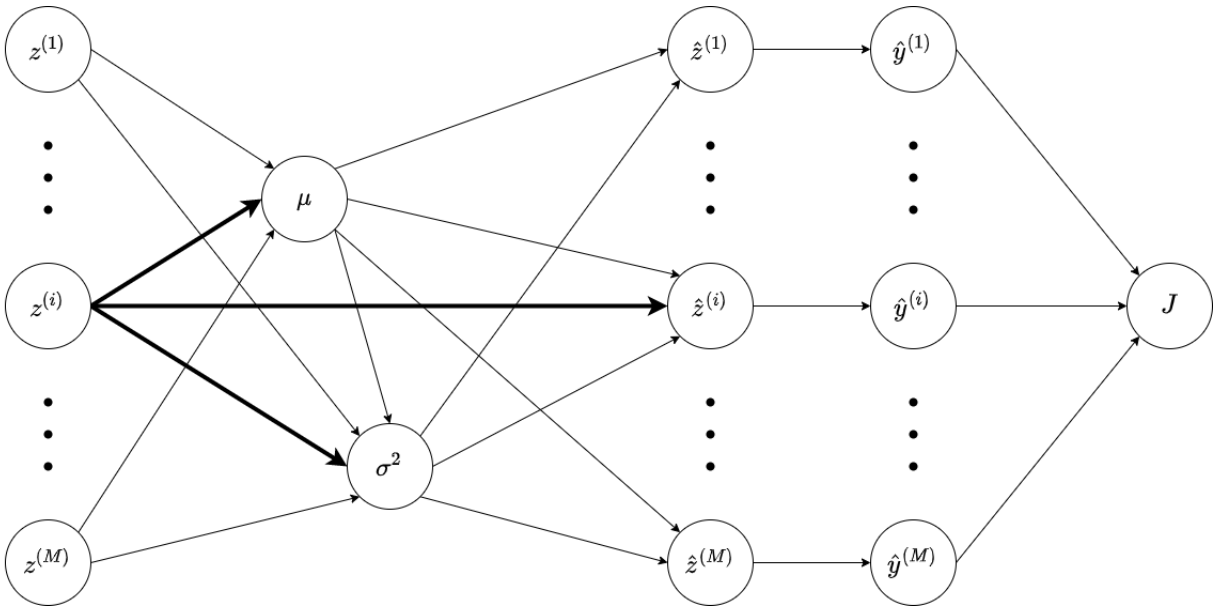


Figure 2.1.0: Dependencies graph of J with respect to $z^{(i)}$

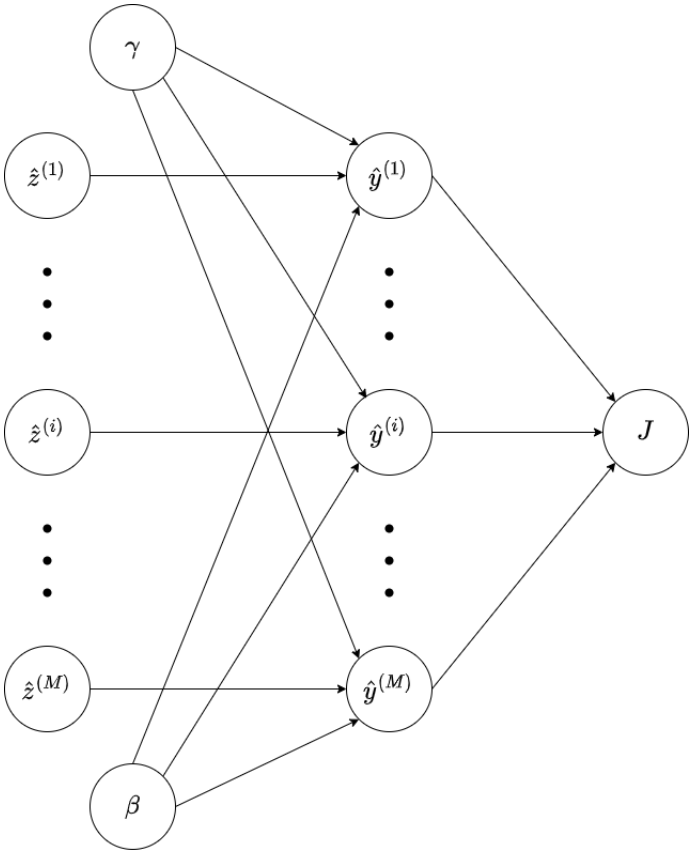
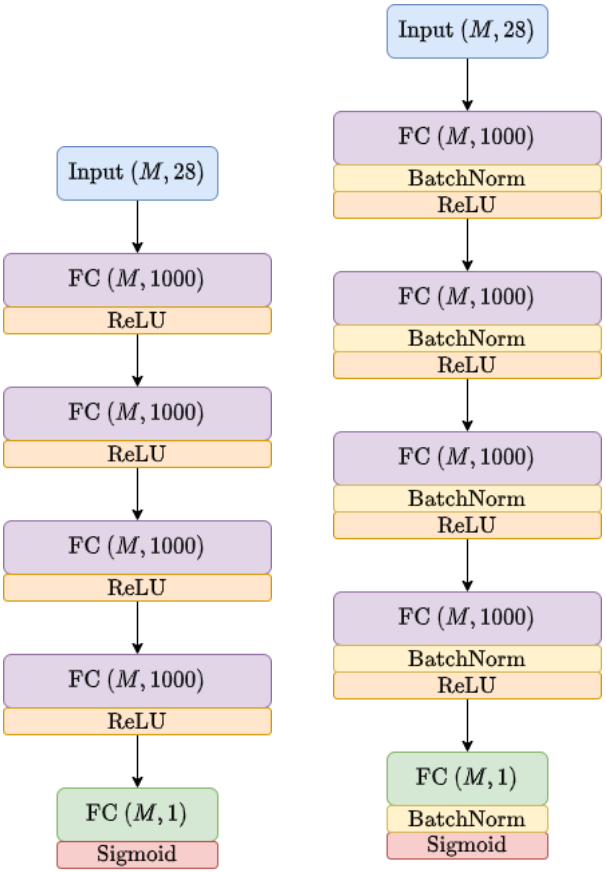


Figure 2.1.1: Dependencies graph of J with respect to $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$



(a) NN control (b) NN with Batch Norm

Figure 3.1.0: The control model and the Batch Normalized model