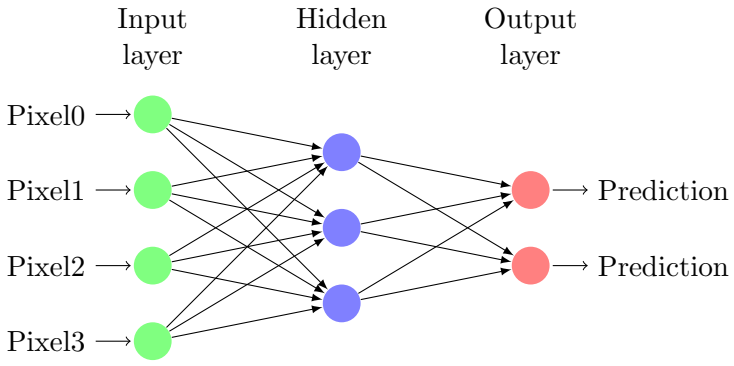
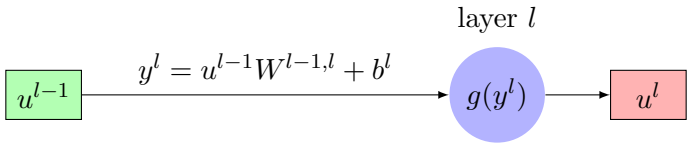
1 Baseline:

Network setup (schematic):



Notations and flows:



u^l is the lth layer's output, $W^{l-1,l}$ is the weight connecting layer $l-1$ and layer l, b^l is the bias at layer l. $g(y)$ is an activation function. The cost function for training can be found in many forms, cross-Entropy, regression etc. In my code, cross-Entropy is used:

$$C(\{\psi_s\},\{\phi_s\},\theta) = -\frac{1}{M}\left[\sum_{s=1}^{M} \phi_s \log[\mathcal{N}_\theta(\psi_s)] + (1-\phi_s)\log[1-\mathcal{N}_\theta(\psi_s)]\right]$$

ψ_s is a set of inputs, ϕ_s is the corresponding labels, $\theta = \{W, b, \hat{\theta}\}$ contains the weights W and bias b to be learned, as well as hyperparameters $\hat{\theta}$ (such as learning rate η). $\mathcal{N}_\theta$ is a network specified by parameters θ. Given an input ψ_s, its output $\mathcal{N}_\theta(\psi_s) = g(y_s^L) \equiv u_s^L$.

Stochastic Gradient Descent:

The workflow of SGD goes as the following:

- Shuffle the training sets, partition the shuffled set into m batches, each contains M_B samples
- For each batch:
  1. Feedforward, crossing all layers.
  2. BackPropagate, computing the the direction for descent: $\mathbf{v} = \left[\frac{\partial C}{\partial W^{i-1,i}}, \frac{\partial C}{\partial b^i}\right], i = 2 \cdots N_L$, N_L being number of layers.
  3. Update the parameters: $W, b \leftarrow W, b - \boldsymbol{\eta} \cdot \mathbf{v}$, in BaseLine, the learning rate is isotropic.

Back-Propagation:

Introduce error rate/sensitivity at layer l for sample s: $\Delta_s^l = \frac{\partial C}{\partial y_s^l}$. This quantity is crucial for BP algorithm. At the last layer, the error rate is:

$$\Delta_s^L \equiv \frac{\partial C}{\partial y_s^L} = \frac{\partial C}{\partial u_s^L} g'(y_s^L)$$

If activation function is sigmoid, then the error rate at the last layer becomes:

$$\Delta_s^L = \frac{1}{M_B}(u_s^L - \phi_s),$$

simply proportional to the difference between network's prediction and the labels. For 2 neighboring layers$\{l, l+1\}$,

$$\begin{aligned}\Delta_s^l = \frac{\partial C}{\partial y_s^l} &= \sum_{s'} \frac{\partial C}{\partial y_{s'}^{l+1}} \frac{\partial y_{s'}^{l+1}}{\partial y_s^l} = \sum_{s'} \Delta_{s'}^{l+1} \frac{\partial}{\partial y_s^l}\left(W^{l,l+1} g(y_{s'}^l) + b^l\right) \\ &= \sum_{s'} W^{l,l+1} \Delta_{s'}^{l+1} g'(y_s^l)\delta_{s,s'} \\ &= W^{l,l+1}\Delta_s^{l+1} g'(y_s^l)\end{aligned}$$

$\delta_{s,s'} = \{1(s = s'), 0(s \neq s')\}$. For the adjustment of weight:

$$\frac{\partial C}{\partial W^{l-1,l}} = \sum_s \frac{\partial C}{\partial y_s^l}\frac{\partial y_s^l}{\partial W^{l-1,l}} = \sum_s \Delta_s^l u_s^{l-1}$$

For the adjustment of bias:

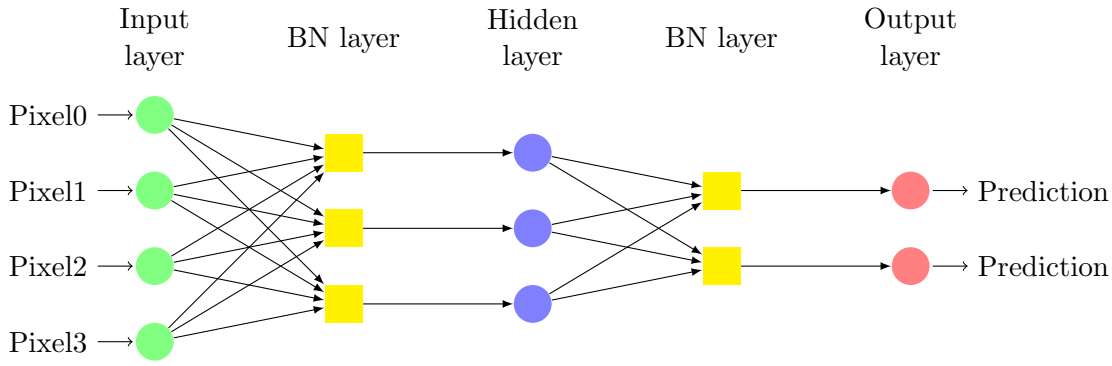
$$\frac{\partial C}{\partial b^l} = \sum_s \frac{\partial C}{\partial y_s^l}\frac{\partial y_s^l}{\partial b^l} = \sum_s \Delta_s^l$$

All in all, the BP algorithm for Baseline is:

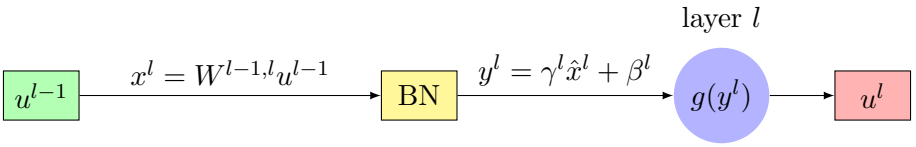
$$\begin{aligned}\Delta_s^l &= W^{l,l+1}\Delta_s^{l+1} g'(y_s^l) \\ \frac{\partial C}{\partial W^{l-1,l}} &= \sum_s \Delta_s^l u_s^{l-1} \\ \frac{\partial C}{\partial b^l} &= \sum_s \Delta_s^l \\ \Delta_s^L &= \frac{1}{M_B}(u_s^L - \phi_s)\end{aligned}$$

2 Batch Normalization:

Network setup (schematic):



Notations and flows:



One important thing about BN algorithm is that it normalize the input right **before** activation:

$$y^l \leftarrow \mathcal{BN}(W^{l-1,l}u^{l-1})$$
$$u^l = g(y^l)$$

$g(u)$ is activation function. It can be *sigmoid* ,*ReLu* etc. $W^{l-1,l}$ is the weights linking layers $\{l-1, l\}$, u^l is the nonlinear activation of layer l. To summarize the normalization in B.N.:

$$x^l = W^{l-1,l}u^{l-1}$$
$$\hat{x}^l = \frac{x^l - \mu_B^l}{\sqrt{\sigma_B^{2,l}}}$$
$$y^l = \gamma^l \hat{x}^l + \beta^l$$
$$u^l = g(y^l)$$

Back-Propagation:

Cost function $C(\gamma, \beta, W, \hat{\theta}, \psi, \phi)$ is a function of hyper parameter $\hat{\theta}$, training parameter$\{\gamma, \beta, W\}$, input ψ and label ϕ. The whole business is centered at minimizing this beast. Define growth rate at layer l: $\Delta_s^l \equiv \frac{\partial C}{\partial y_s^l}$. Here y_s^l is the linear combination of sample s that waits to be activated. Using chain rule, BP for BN can be derived as below. It differs with the baseline BP slightly.

$$\frac{\partial C}{\partial W^{l-1,l}} = \sum_s \frac{\partial C}{\partial y_s^l}\frac{\partial y_s^l}{\partial W^{l-1,l}} \equiv \sum_s \Delta_s^l \frac{\partial y_s^l}{\partial W^{l-1,l}}$$
$$\frac{\partial y_s^l}{\partial W^{l-1,l}} = \frac{\gamma^l}{\sqrt{\sigma_B^2}^l}\left[u_s^{l-1} - \left\langle u^{l-1}\right\rangle - \hat{x}_s^l \left\langle u^{l-1}\hat{x}^l\right\rangle\right] \tag{1}$$
$$\Delta_s^l = \frac{\gamma^{l+1}}{\sqrt{\sigma_B^2}^{l+1}} W^{l,l+1} g'(y_s^l)\left[\Delta_s^{l+1} - \left\langle \Delta^{l+1}\right\rangle - \hat{x}_s^{l+1}\left\langle \Delta^{l+1}\hat{x}^{l+1}\right\rangle\right] \tag{2}$$
$$\frac{\partial C}{\partial \beta^l} = \sum_s \frac{\partial C}{\partial y_s^l}$$
$$\frac{\partial C}{\partial \gamma^l} = \sum_s \frac{\partial C}{\partial y_s^l}\hat{x}_s^l$$

$\langle \cdots \rangle$ is the sample average. s is the label of sample in a mini-batch. l labels layer. $g'(x)$ is the derivative of activation function. At the boundary:

$$\Delta_s^L = \frac{\partial C}{\partial u_s^L} g'(y_s^L)$$

L labels the last layer in network. Derivation of $\frac{\partial y_s^l}{\partial W^{l-1,l}}$ and Δ_s^l both involves computing quantity $\frac{\partial y_s^l}{\partial x_{s'}^l}$, which exemplifes the mixing of training samples:

$$
\begin{aligned}
\frac{\partial}{\partial x^l_{s'}}(\sigma^l)^2 &= \frac{\partial}{\partial x^l_{s'}}\left[\frac{1}{M_B}\sum_s^{M_B}(x^l_s-\mu^l_B)^2\right] = \frac{2}{M_B}\sum_s(x^l_s-\mu^l_B)(\delta_{s,s'}-\frac{1}{M_B}) \\
&= \frac{2}{M_B}(x^l_{s'}-\mu^l_B-\mu^l_B+\mu^l_B) = \frac{2}{M_B}(x^l_{s'}-\mu^l_B) \\
\frac{\partial y^l_s}{\partial x^l_{s'}} &= \frac{\partial}{\partial x^l_{s'}}\left[\gamma^l\frac{x^l_s-\mu^l_B}{\sigma^l}+\beta_l\right] \\
&= \frac{\gamma^l}{\sigma^l}\left[\delta_{s,s'}-\frac{1}{M_B}\right]-\frac{\gamma^l}{2(\sigma^l)^3}\left(x^l_s-\mu^l_B\right)\frac{\partial}{\partial x^l_{s'}}(\sigma^l)^2 \\
&= \frac{\gamma^l}{\sigma^l}\left[\delta_{s,s'}-\frac{1}{M_B}\right]-\frac{\gamma^l}{2(\sigma^l)^3}\left(x^l_s-\mu^l_B\right)\frac{2}{M_B}(x^l_{s'}-\mu^l_B) \\
&= \frac{\gamma^l}{\sigma^l}\left[\delta_{s,s'}-\frac{1}{M_B}-\frac{1}{M_B}\frac{x^l_s-\mu^l_B}{\sigma^l}\frac{x^l_{s'}-\mu^l_B}{\sigma^l}\right] \\
&= \frac{\gamma^l}{\sigma^l}\left[\delta_{s,s'}-\frac{1}{M_B}-\frac{1}{M_B}\hat{x}^l_s\hat{x}^l_{s'}\right] \qquad (3)
\end{aligned}
$$

With the above identity, one can compute the errors Δ and the weight adjustment $\frac{\partial y}{\partial W}$.

We first look into the errors. When the input to activation function at layer l changes: $y^l \to y^l + \delta y^l$, it will cause chain correction for all the layers $l' \geq l$. E.g. the change of cost δC caused by layer l equals to summing all the changes that happen at layer $l+1$ caused by δy^l:

$$
\begin{aligned}
\Delta^l_s \equiv \frac{\partial C}{\partial y^l_s} &= \sum_{s'}\frac{\partial C}{\partial y^{l+1}_{s'}}\frac{\partial y^{l+1}_{s'}}{\partial y^l_s} \\
&= \sum_{s'}\Delta^{l+1}_{s'}\frac{\partial y^{l+1}_{s'}}{\partial x^{l+1}_s}\frac{\partial x^{l+1}_s}{\partial y^l_s} = \sum_{s'}\Delta^l_{s'}\frac{\partial y^{l+1}_{s'}}{\partial x^{l+1}_s}\frac{\partial}{\partial y^l_s}[W^{l,l+1}u^l_s] \\
&= \sum_{s'}\Delta^{l+1}_{s'}W^{l,l+1}g'(y^l_s)\frac{\partial y^{l+1}_{s'}}{\partial x^{l+1}_s}
\end{aligned}
$$

Plug the kernel term (3) in,

$$
\begin{aligned}
\Delta^l_s &= \sum_{s'}\Delta^{l+1}_{s'}W^{l,l+1}g'(y^l_s)\frac{r^l}{\sigma^l}\left[\delta_{s,s'}-\frac{1}{M_B}-\frac{1}{M_B}\hat{x}^l_s\hat{x}^l_{s'}\right] \\
&= \frac{\gamma^l W^{l,l+1}g'(y^l_s)}{\sigma^l}\left[\Delta^{l+1}_s-\left\langle\Delta^{l+1}\right\rangle-\hat{x}^l_s\left\langle\Delta^{l+1}\hat{x}^l\right\rangle\right]
\end{aligned}
$$

Similarly, one can also obtain the term:

$$
\begin{aligned}
\frac{\partial y^l_s}{\partial W^{l-1,l}} &= \sum_{s'}\frac{\partial y^l_s}{\partial x^l_{s'}}\frac{\partial x^l_{s'}}{\partial W^{l-1,l}} \\
&= \sum_{s'}u^{l-1}_{s'}\frac{\gamma^l}{\sigma^l}\left[\delta_{s,s'}-\frac{1}{M_B}-\frac{1}{M_B}\hat{x}^l_s\hat{x}^l_{s'}\right] \\
&= \frac{\gamma^l}{\sigma^l}\left[u^{l-1}_s-\left\langle u^{l-1}\right\rangle-\hat{x}^l_s\left\langle\hat{x}^l u^{l-1}\right\rangle\right]
\end{aligned}
$$

When one layer contains more than one neuron, one can just re-arrange the multiplication and addition/subtraction accordingly through dimension analysis. For example, when layer $l-1$ contains n_{l-1} neuron and layer

l contains n_l neurons, $\frac{\partial y_s^l}{\partial W^{l-1,l}}$ is a $n_{l-1} \times n_l$ matrix, the averaging term $\left\langle \hat{x}_s^l u^{l-1} \right\rangle$ can be written as the sample average of outer product $\frac{1}{M_B}\sum_s u_s^{l-1}\hat{x}_s^l$, which satisfies the dimension requirement.

3 Insights:

The key difference between BN and BL is the mixing among different samples in a minibatch. It can be clearly seen from the kernel expression (3). Its effect can is the modification of weights change and (1) and the error change(2). The weights adjustment in BP can be expressed in a matrix form in sample space:

$$\begin{aligned}
\frac{\partial C}{\partial W^{l-1,l}} &= \frac{\gamma^l}{\sigma^l}\boldsymbol{\Delta^l}\cdot\boldsymbol{M}\cdot\boldsymbol{u^{l-1}} \\
\boldsymbol{\Delta^l} &= (\Delta_1^l, \Delta_2^l, \cdots, \Delta_{M_B}^l) \\
\boldsymbol{u^{l-1}} &= (u_1^{l-1}, u_2^{l-1}, \cdots, u_{M_B}^{l-1})^T \\
\boldsymbol{M} &= \boldsymbol{I} - \frac{1}{M_B}\left[\boldsymbol{O} + \hat{\boldsymbol{X}}^{\boldsymbol{l}}\right]
\end{aligned}$$

$\boldsymbol{I}$ is an $M_B \times M_B$ identity matrix, $\boldsymbol{O}$ is an $M_B \times M_B$ matrix with $\boldsymbol{O}_{i,j} = 1$, $\forall i, j$; and $\hat{\boldsymbol{X}}^{\boldsymbol{l}}_{i,j} = \hat{x}_i^l \hat{x}_j^l$.

To see what (3) does, it is helpful to simplify the neural network such that each layer only contains 1 neuron. Then the weights W between each layer becomes a scalar. Under such simplification, $\hat{x}_s^l = \frac{\cancel{W}u_s^{l-1} - \langle \cancel{W}u^{l-1}\rangle}{STD(\cancel{W}u^{l-1})} \equiv \hat{u}_s^{l-1}$. Keep in mind that the STDs for x and u are different: $\sigma^l \equiv STD(Wu^{l-1})$, $\tilde{\sigma}^{l-1} \equiv STD(u^{l-1}) = W^{-1}\sigma^l$. Since $\hat{x}_s^l$ is already normalized, $\sum_s \hat{x}_s^l = 0$, $\sum_s (\hat{x}_s^l)^2 = M_B$, $\forall l$. Now examine the weights update.

$$\begin{aligned}
\boldsymbol{M}\cdot\boldsymbol{u^{l-1}} &= \boldsymbol{u^{l-1}} - \frac{1}{M_B}\left[\boldsymbol{O} + \hat{\boldsymbol{X}}^{\boldsymbol{l}}\right]\cdot\left(\boldsymbol{u^{l-1}} - \left\langle u^{l-1}\right\rangle\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix} + \left\langle u^{l-1}\right\rangle\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix}\right)\frac{\tilde{\sigma}^{l-1}}{\tilde{\sigma}^{l-1}} \\
&= \boldsymbol{u^{l-1}} - \frac{1}{M_B}\left[\hat{\boldsymbol{O}} + \hat{\boldsymbol{X}}^{\boldsymbol{l}}\right]\cdot\left(\hat{\boldsymbol{u}}^{\boldsymbol{l-1}} + \frac{\langle u^{l-1}\rangle}{\tilde{\sigma}^{l-1}}\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix}\right)\tilde{\sigma}^{l-1} \\
&= \boldsymbol{u^{l-1}} - \frac{1}{M_B}\left(M_B\frac{\langle u^{l-1}\rangle}{\tilde{\sigma}^{l-1}}\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix} + M_B\frac{\hat{\boldsymbol{u}}^{\boldsymbol{l-1}}\cdot\hat{\boldsymbol{u}}^{\boldsymbol{l-1}}}{M_B}\hat{\boldsymbol{u}}^{\boldsymbol{l-1}}\right)\tilde{\sigma}^{l-1} \\
&= \boldsymbol{u^{l-1}} - \left\langle u^{l-1}\right\rangle\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix} - \frac{\hat{\boldsymbol{u}}^{\boldsymbol{l-1}}\cdot\hat{\boldsymbol{u}}^{\boldsymbol{l-1}}}{M_B}\hat{\boldsymbol{u}}^{\boldsymbol{l-1}}\tilde{\sigma}^{l-1} \\
&= \tilde{\sigma}^{l-1}\left(\hat{\boldsymbol{u}}^{\boldsymbol{l-1}} - \frac{\hat{\boldsymbol{u}}^{\boldsymbol{l-1}}\cdot\hat{\boldsymbol{u}}^{\boldsymbol{l-1}}}{M_B}\hat{\boldsymbol{u}}^{\boldsymbol{l-1}}\right) \\
&= \boldsymbol{0}\,\boldsymbol{!!!}
\end{aligned}$$

Weights Update

Now we consider a more general case when each layer has more than one neuron. Use subscripted latin letter i, j, k to denote the neuron. The BN procedure can be written as:

$$
\begin{aligned}
x^l_{s,i} &= \sum_j W^{l-1,l}_{i,j} u^{l-1}_{s,j} \\
\hat{x}^l_{s,i} &= \frac{x^l_{s,i} - \langle x^l_i \rangle}{\sigma^l_i} \\
y^l_{s,i} &= \gamma^l_i \hat{x}^l_{s,i} + \beta^l_i \\
u^l_{s,i} &= g(y^l_{s,i})
\end{aligned}
$$

Since the ith neuron's BN input $y^l_{s,i}$ is only related with the ith neuron's $x^l_{s,i}$, the kernel function for each neuron is then :

$$
\frac{\partial y^l_{s,i}}{\partial x^l_{s',i}} = \frac{\gamma^l_i}{\sigma^l_i}\left[\delta_{s,s'} - \frac{1}{M_B} - \frac{1}{M_B}\hat{x}^l_{s,i}\hat{x}^l_{s',i}\right]; \quad \frac{\partial y^l_{s,i}}{\partial x^l_{s',j}} = 0,\ i \neq j
$$

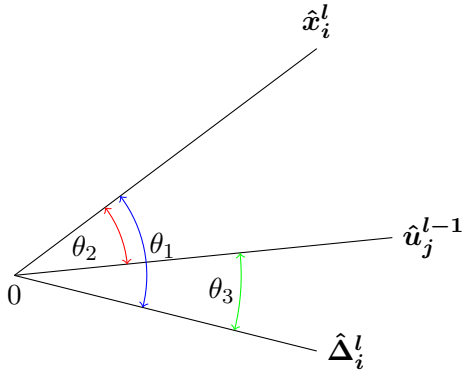
With this setup, the weight update rule in back-propagation becomes:

$$
\begin{aligned}
\frac{\partial C}{\partial W^{l-1,l}_{i,j}} &= \sum_{s,k} \frac{\partial C}{\partial y^l_{s,k}} \frac{\partial y^l_{s,k}}{\partial W^{l-1,l}_{i,j}} = \sum_{s,k} \Delta^l_{s,k} \frac{\partial y^l_{s,k}}{\partial W^{l-1,l}_{i,j}} \\
\frac{\partial y^l_{s,k}}{\partial W^{l-1,l}_{i,j}} &= \delta_{i,k} \frac{\gamma^l_i}{\sigma^l_i}\left[u^{l-1}_{s,j} - \left\langle u^{l-1}_j \right\rangle - \hat{x}^l_{s,i}\left\langle u^{l-1}_j \hat{x}^l_i \right\rangle\right] \\
\Rightarrow \frac{\partial C}{\partial W^{l-1,l}_{i,j}} &= M_B \frac{\gamma^l_i}{\sigma^l_i}\left(\left\langle \Delta^l_i u^{l-1}_j \right\rangle - \left\langle \Delta^l_i \right\rangle \left\langle u^{l-1}_j \right\rangle - \left\langle \Delta^l_i \hat{x}^l_i \right\rangle \left\langle u^{l-1}_j \hat{x}^l_i \right\rangle\right)
\end{aligned}
$$

Introduce two new variance, the variance of error term $\sigma(\Delta^l_i)$ and the variance of output $\sigma(u^{l-1}_j)$. The normalized error $\hat{\Delta}^l_i$ and normalized output $\hat{u}^{l-1}_j$ can be written as vectors in sample space: $\boldsymbol{\hat{\Delta}^l_i} = (\hat{\Delta}^l_{1,i}, \hat{\Delta}^l_{2,i}, \cdots, \hat{\Delta}^l_{M_B,i})$, $\boldsymbol{\hat{u}^{l-1}_j} = (\hat{u}^{l-1}_{1,j}, \hat{u}^{l-1}_{2,j}, \cdots, \hat{u}^{l-1}_{M_B,i})$ The above equation can be rewritten as:

$$
\begin{aligned}
\frac{\partial C}{\partial W^{l-1,l}_{i,j}} &= M_B \gamma^l_i \frac{\sigma(\Delta^l_i)\sigma(u^{l-1}_j)}{\sigma(x^l_i)}\left(Corr(\Delta^l_i, u^{l-1}_j) - \frac{\boldsymbol{\hat{\Delta}^l_i} \cdot \boldsymbol{\hat{x}^l_i}}{M_B}\frac{\boldsymbol{\hat{u}^{l-1}_j} \cdot \boldsymbol{\hat{x}^l_i}}{M_B}\right) \\
Corr(\Delta^l_i, u^{l-1}_j) &= \left\langle \frac{\Delta^l_{s,i} - \langle \Delta^l_i \rangle}{\sigma(\Delta^l_i)} \frac{u^{l-1}_{s,j} - \left\langle u^{l-1}_j \right\rangle}{\sigma(u^{l-1}_j)} \right\rangle = \frac{\boldsymbol{\hat{\Delta}^l_i} \cdot \boldsymbol{\hat{u}^{l-1}_j}}{M_B} \qquad (4)
\end{aligned}
$$

Recall that all the normalized quantity has fixed length in sample space $|\hat{\boldsymbol{a}}| = \sqrt{M_B}$, the above equation has a geometrical interpretation.



$$
\begin{aligned}
BN: \quad & \frac{\partial C}{\partial W_{i,j}^{l-1,l}} = M_B \gamma_i^l \frac{\sigma(\Delta_i^l)\sigma(u_j^{l-1})}{\sigma(x_i^l)} [\cos(\theta_3) - \cos(\theta_1)\cos(\theta_2)] \\
BL: \quad & \frac{\partial C}{\partial W_{i,j}^{l-1,l}} = \sum_s \left[\sigma(\Delta_i^l)\hat{\Delta}_{s,i}^l + \left\langle \Delta_i^l \right\rangle\right] \left[\sigma(u_j^{l-1})\hat{u}_{s,j}^{l-1} + \left\langle u_j^{l-1} \right\rangle\right] \\
& = M_B \sigma(\Delta_i^l)\sigma(u_j^{l-1})\cos(\theta_3) + M_B \left\langle \boldsymbol{\Delta_i^l} \right\rangle \left\langle \boldsymbol{u_j^{l-1}} \right\rangle
\end{aligned} \tag{5}
$$

Compare BN and BL, one can see 2 major differences. The first one being the amplitude of the adjustment. BN performs a more precise micro-tuning: the adjustment is proportional to variance. While BL also take the sample average into count, which can cause orders of difference when it comes to adjustment.
Secondly, the stopping condition is different. Assume that the layer's output u_i^l is normalized. For BL, the learning continues until the normalized error of current layer $\hat{\Delta}_i^l$ and normalized output of previous layer $\hat{u}_j^{l-1}$ is not correlated anymore.
For BN, the weights has 2 functionality. (1) it rotates the previous layer's output so the rotated quantity $\hat{x}^l$ can strongly correlate with the error term propagated back. This is reflected by the fact that the learning will stop if $\sin(\theta_1) = 0$. Once $\sin(\theta_1) = 0$, the learning is delegated to γ as shown in the next subsection. (2) Assume $\sin(\theta_1) \neq 0$. When the correlation between $\hat{x}_i^l$ and $\hat{u}_j^{l-1}$ is maximized, e.g. $\sin(\theta_2) = 0$, the update of weight $W_{i,j}^{l-1,l}$ stops. For another neuron in layer $l-1$, say neuron k, if its sample space vector is not aligned with neuron j in layer $l-1$, it will be updated. However, if $\hat{u}_k^{l-1}$ is aligned with $\hat{x}_i^l$, its update will stop too. The net effect is that if $\hat{u}_k^{l-1}$ and $\hat{u}_j^{l-1}$ are strongly correlated in sample space (e.g. their vectors are aligned), the update of both $W_{i,j}^{l-1,l}$ and $W_{i,k}^{l-1,l}$ stop. In essence, BN could make all neurons of layer $l-1$ connecting to ith neuron in layer l to be synced.

γ Update

The update of γ can be analyzed in the similar way.

$$
\frac{\partial C}{\partial \gamma_i^l} = \sigma(\Delta_i^l) \sum_s \hat{\Delta}_i^l \hat{x}_i^l = \sigma(\Delta_i^l) M_B \cos(\theta_1)
$$

Since correlation is the same as dot product of two normalized quantity divided by sample number: $Corr(\hat{A}, \hat{B}) = \hat{\boldsymbol{A}} \cdot \hat{\boldsymbol{B}}/M_B$, the above equation can also be expressed as:

$$
\frac{\partial C}{\partial \gamma_i^l} = \sigma(\Delta_i^l) M_B \, Corr(\hat{\Delta}_i^l, \hat{x}_i^l) \tag{6}
$$

e.g., when normalized error and input x is highly correlated across the samples, the most probable cause will be γ hence γ will be adjusted most. If the correlation is 0, which indicates the errors have nothing to do with inputs, γ doesn't need to be adjusted.

Error Propagation

For the update of γ, β, the error propagation is crucial. With each layer having more than one neuron, the BP for Δ_s^l is:

$$\begin{aligned}
\Delta_{s,i}^l = \frac{\partial C}{\partial y_{s,i}^l} &= \sum_{s',k} \frac{\partial C}{\partial y_{s',k}^{l+1}} \frac{\partial y_{s',k}^{l+1}}{\partial y_{s,i}^l} = \sum_{s',s'',k} \Delta_{s',k}^{l+1} \frac{\partial y_{s',k}^{l+1}}{\partial x_{s'',k}^{l+1}} \frac{\partial x_{s'',k}^{l+1}}{\partial y_{s',i}^l} = \sum_{s',s'',k} \Delta_{s',k}^{l+1} \frac{\partial y_{s',k}^{l+1}}{\partial x_{s'',k}^{l+1}} W_{k,i}^{l,l+1} g'(y_{s,i}^l) \delta_{s,s''} \\
&= g'(y_{s,i}^l) \sum_{s',k} \Delta_{s',k}^{l+1} W_{k,i}^{l,l+1} \frac{\partial y_{s',k}^{l+1}}{\partial x_{s,k}^{l+1}} \\
&= g'(y_{s,i}^l) \sum_{s',k} \Delta_{s',k}^{l+1} W_{k,i}^{l,l+1} \frac{\gamma_k^{l+1}}{\sigma_k^{l+1}} \left[\delta_{s',s} - \frac{1}{M_B} - \frac{1}{M_B} \hat{x}_{s',k}^{l+1} \hat{x}_{s,k}^{l+1} \right] \\
&= g'(y_{s,i}^l) \sum_{k} W_{k,i}^{l,l+1} \frac{\gamma_k^{l+1}}{\sigma(x_k^{l+1})} \left[\Delta_{s,k}^{l+1} - \left\langle \Delta_k^{l+1} \right\rangle - \hat{x}_{s,k}^{l+1} \left\langle \Delta_k^{l+1} \hat{x}_k^{l+1} \right\rangle \right] \\
&= g'(y_{s,i}^l) \sum_{k} W_{k,i}^{l,l+1} \frac{\gamma_k^{l+1}}{\sigma(x_k^{l+1})} \sigma(\Delta_k^{l+1}) \left[\hat{\Delta}_{s,k}^{l+1} - \hat{x}_{s,k}^{l+1} \left\langle \hat{\Delta}_k^{l+1} \hat{x}_k^{l+1} \right\rangle \right]
\end{aligned}$$

Again, in matrix form (Sample Space):

$$\boldsymbol{\Delta_i^{l-1}} = g'(\boldsymbol{y_i^{l-1}}) \odot \left[\sum_k W_{k,i}^{l-1,l} \frac{\gamma_k^l}{\sigma(x_k^l)} \sigma(\Delta_k^l) \left(\hat{\boldsymbol{\Delta}}_{\boldsymbol{k}}^{\boldsymbol{l}} - \cos(\theta_1) \hat{\boldsymbol{x}}_{\boldsymbol{k}}^{\boldsymbol{l}} \right) \right]$$

where θ_1 is the angle between sample space vector $\hat{\boldsymbol{\Delta}}_{\boldsymbol{k}}^{\boldsymbol{l}}$ and $\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{\boldsymbol{l}}$. Given all other things equal, if $\hat{\boldsymbol{\Delta}}_{\boldsymbol{k}}^{\boldsymbol{l}}$ and $\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{\boldsymbol{l}}$ is aligned in sample space, the error won't propagate from this node backwards. If these 2 vectors are opposite, the error back-propagates with largest magnitude.