Batch Normalization

CS 682 Optional Discussion - Oct 4

Feature Scaling

- The range of values of raw training data often varies widely
- In machine learning algorithms, the functions involved in the optimization process are sensitive to normalization
    - For example: Distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by this particular feature.
    - After, normalization, each feature contributes approximately proportionately to the final distance.
- In general, Gradient descent converges much faster with feature scaling than without it.
- Normalization ensures numerical stability which lets you use higher learning rates.

Normalization Techniques

Two methods are usually used for rescaling or normalizing data:

- Scaling data all numeric variables to the range [0,1]. One possible formula is given below:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- To have zero mean and unit variance:

$$x_{new} = \frac{x - \mu}{\sigma}$$

- In the NN community this is called *Whitening*

Batch Normalization

- Batch Normalization (BN) is a normalization method/layer for neural networks.
- Usually inputs to neural networks are normalized to either the range of [0, 1] or [-1, 1] or to mean=0 and variance=1
- BN essentially performs Whitening to the intermediate layers of the networks.

Why Batch Normalization is good?

- BN reduces *Covariate Shift*. That is the change in distribution of activation of a component. By using BN, each neuron's activation becomes (more or less) a Gaussian distribution.
- Covariate Shift is undesirable, because the later layers have to keep adapting to the change of the type of distribution.
- BN reduces effects of exploding and vanishing gradients, because every becomes roughly normal distributed. Without BN, low activations of one layer can lead to lower activations in the next layer, and then even lower ones in the next layer and so on.

The BN transformation is scalar invariant

- Batch Normalization also makes training more resilient to the parameter scale.
- Normally, large learning rates may increase the scale of layer parameters, which then amplify the gradient during backpropagation and lead to the model explosion
- However, with Batch Normalization, backpropagation through a layer is unaffected by the scale of its parameters. Indeed, for a scalar a,

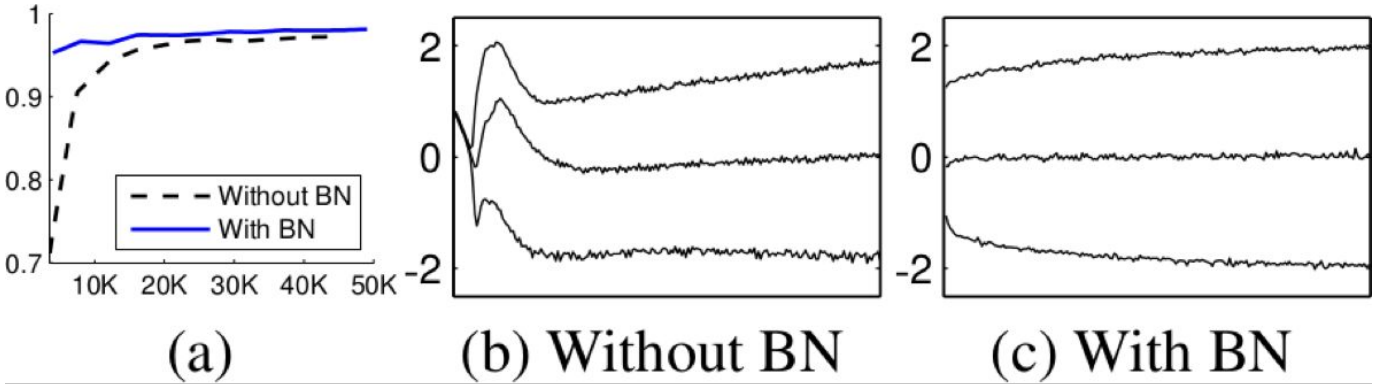
$$\mathrm{BN}(W\mathrm{u}) = \mathrm{BN}((aW)\mathrm{u})$$

and thus $\frac{\partial \mathrm{BN}((aW)\mathrm{u})}{\partial \mathrm{u}} = \frac{\partial \mathrm{BN}(W\mathrm{u})}{\partial \mathrm{u}}$, so the scale does not affect the layer Jacobian nor, consequently, the gradient propagation. Moreover, $\frac{\partial \mathrm{BN}((aW)\mathrm{u})}{\partial (aW)} = \frac{1}{a} \cdot \frac{\partial \mathrm{BN}(W\mathrm{u})}{\partial W}$, so larger weights lead to *smaller* gradients, and Batch Normalization will stabilize the parameter growth.

Batch normalization: Other benefits in practice

- BN reduces training times. (Because of less Covariate Shift, less exploding/vanishing gradients.)
- BN reduces demand for regularization, e.g. dropout or L2 norm.
  - Because the means and variances are calculated over batches and therefore every normalized value depends on the current batch. I.e. the network can no longer just memorize values and their correct answers.)
- BN allows higher learning rates. (Because of less danger of exploding/vanishing gradients.)
- BN enables training with saturating nonlinearities in deep networks, e.g. sigmoid. (Because the normalization prevents them from getting stuck in saturating ranges, e.g. very high/low values for sigmoid.)
- Less sensitive to careful initialization

Batch normalization: Better accuracy , faster.

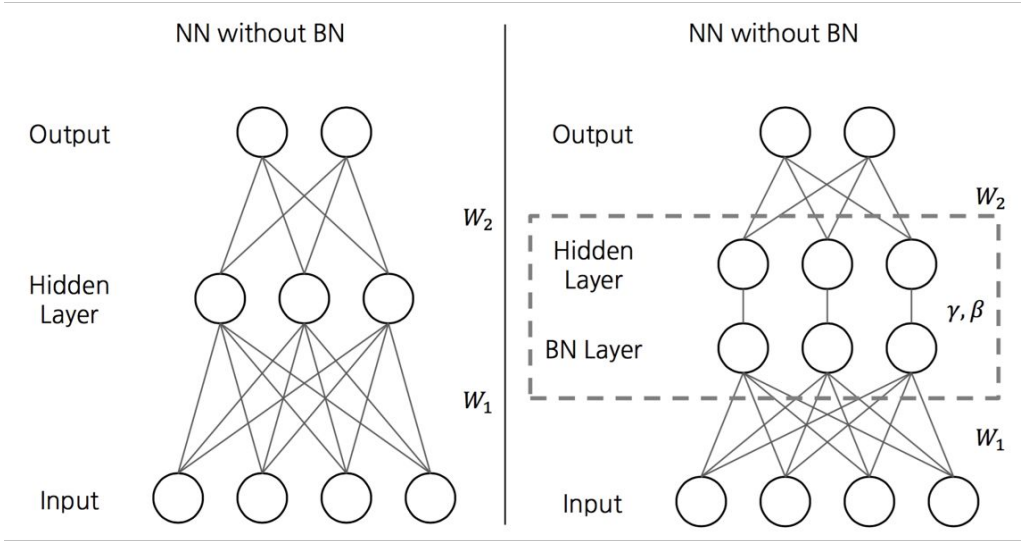


BN applied to MNIST (a), and activations of a randomly selected neuron over time (b, c), where the middle line is the median activation, the top line is the 15th percentile and the bottom line is the 85th percentile.

How Batch Normalization is added

we introduce, for each activation $x^{(k)}$, a pair of parameters $\gamma^{(k)}, \beta^{(k)}$, which scale and shift the normalized value:

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}.$$



A new layer is added so the gradient can "see" the normalization and make adjustments if needed.

Delve into details

Batch Normalization

Instead of whitening, normalize each scalar element individually:

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Estimate $\mathrm{E}[x^{(k)}]$ and $\mathrm{Var}[x^{(k)}]$ with mini-batch statistics

$$y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)}$$

Forward Pass

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Backward Pass

Inputs: $\frac{\partial l}{\partial y_i}$ and all things cached in forward pass (x_i, μ_B, σ_B^2,...)

Output: $\frac{\partial l}{\partial x_i}, \frac{\partial l}{\partial \gamma}, \frac{\partial l}{\partial \beta}$

Backward Pass

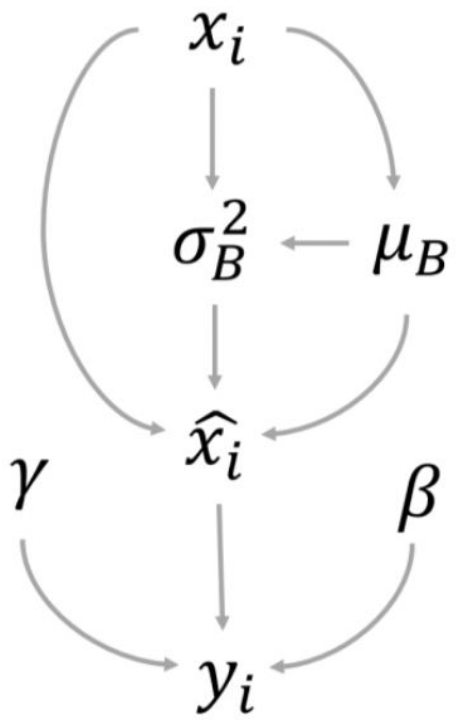
Foward pass:

$$\mu_B = \frac{1}{m}\sum x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum (x_i - \mu_B)^2$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i = \gamma \hat{x}_i + \beta$$

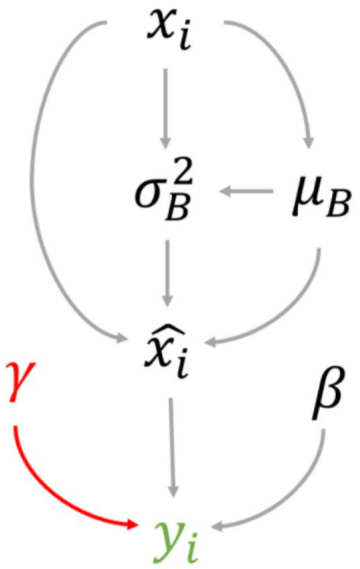


Backward Pass

Computing $\frac{\partial l}{\partial \gamma}$

$$\mu_B = \frac{1}{m}\sum x_i\,,\quad \sigma_B^2 = \frac{1}{m}\sum (x_i - \mu_B)^2\,,\quad \hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\,,\quad y_i = \gamma \hat{x}_i + \beta$$

$$\frac{\partial l}{\partial \gamma} = \sum \frac{\partial l}{\partial y_i}\hat{x}_i$$

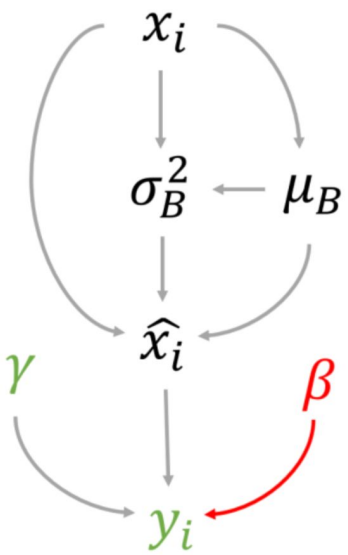


Backward Pass

Computing $\frac{\partial l}{\partial \beta}$

$$\mu_B = \frac{1}{m}\sum x_i\,,\quad \sigma_B^2 = \frac{1}{m}\sum (x_i - \mu_B)^2\,,\quad \hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\,,\quad y_i = \gamma \hat{x}_i + \beta$$

$$\frac{\partial l}{\partial \beta} = \sum \frac{\partial l}{\partial y_i}$$

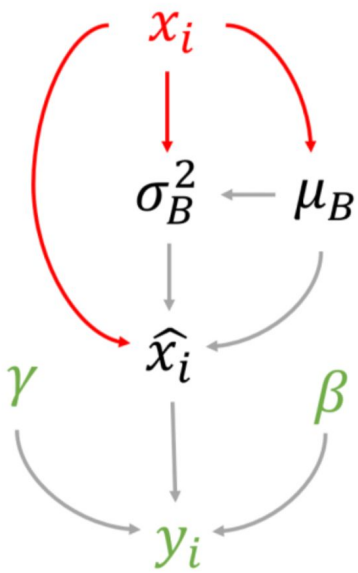


Backward Pass

Computing $\frac{\partial l}{\partial x_i}$

$$\mu_B = \frac{1}{m}\sum x_i\,,\quad \sigma_B^2 = \frac{1}{m}\sum (x_i - \mu_B)^2\,,\quad \hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\,,\quad y_i = \gamma \hat{x}_i + \beta$$

Not ready yet ...

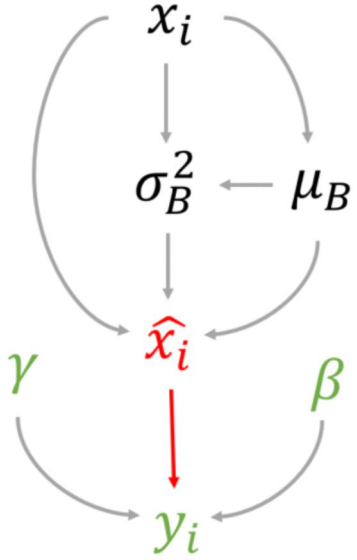


Backward Pass

Computing $\frac{\partial l}{\partial \hat{x}_i}$

$$\mu_B = \frac{1}{m}\sum x_i\,,\quad \sigma_B^2 = \frac{1}{m}\sum (x_i - \mu_B)^2\,,\quad \hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\,,\quad y_i = \gamma \hat{x}_i + \beta$$

$$\frac{\partial l}{\partial \hat{x}_i} = \frac{\partial l}{\partial y_i}\gamma$$



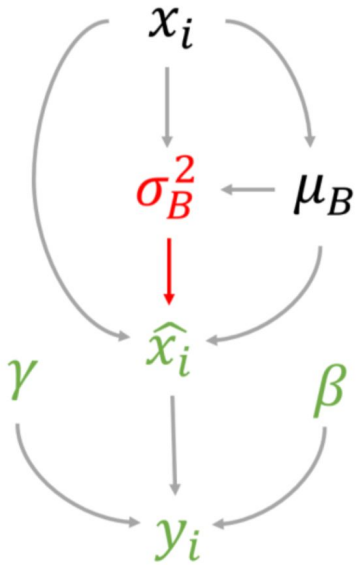
Backward Pass

Computing $\frac{\partial l}{\partial \sigma_B^2}$

$$\mu_B = \frac{1}{m}\sum x_i\,,\quad \sigma_B^2 = \frac{1}{m}\sum (x_i - \mu_B)^2\,,\quad \hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\,,\quad y_i = \gamma \hat{x}_i + \beta$$

$$\frac{\partial l}{\partial \sigma_B^2} = \sum \frac{\partial l}{\partial \hat{x}_i}\frac{\partial \hat{x}_i}{\partial \sigma_B^2}$$

$$= \sum \frac{\partial l}{\partial \hat{x}_i}\frac{-1}{2}(x_i - \mu_B)(\sigma_B^2 + \epsilon)^{-3/2}$$

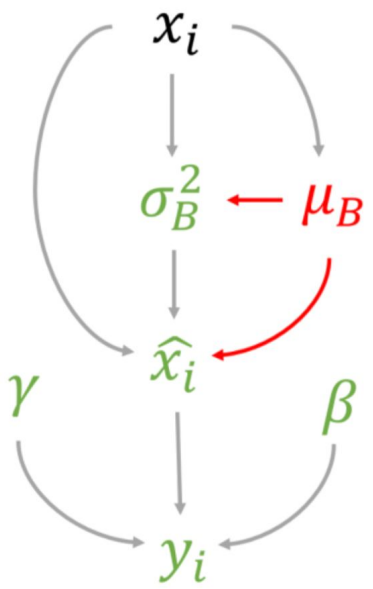


Backward Pass

Computing $\frac{\partial l}{\partial \mu_B}$

$$\mu_B = \frac{1}{m}\sum x_i\,,\quad \sigma_B^2 = \frac{1}{m}\sum (x_i-\mu_B)^2\,,\quad \hat{x}_i = \frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\epsilon}}\,,\quad y_i = \gamma\hat{x}_i + \beta$$

$$\frac{\partial l}{\partial \mu_B} = \frac{\partial l}{\partial \sigma_B^2}\frac{\partial \sigma_B^2}{\partial \mu_B} + \sum \frac{\partial l}{\partial \hat{x}_i}\frac{\partial}{\partial \mu_B}\left(\frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\epsilon}}\right)$$
$$= \frac{\partial l}{\partial \sigma_B^2}\frac{\sum -2(x_i-\mu_B)}{m} + \sum \frac{\partial l}{\partial \hat{x}_i}\frac{-1}{\sqrt{\sigma_B^2+\epsilon}}$$

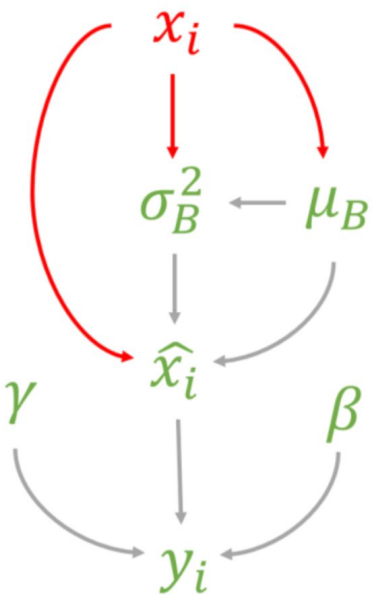


Backward Pass

Computing $\frac{\partial l}{\partial x_i}$

$$\mu_B = \frac{1}{m}\sum x_i\,,\quad \sigma_B^2 = \frac{1}{m}\sum (x_i-\mu_B)^2\,,\quad \hat{x}_i = \frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\epsilon}}\,,\quad y_i = \gamma\hat{x}_i + \beta$$

$$\begin{aligned}\frac{\partial l}{\partial x_i} &= \frac{\partial l}{\partial \hat{x}_i}\frac{\partial}{\partial x_i}\left(\frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\epsilon}}\right)\\ &+\frac{\partial l}{\partial \sigma_B^2}\frac{\partial}{\partial x_i}\left(\frac{1}{m}\sum(x_i-\mu_B)^2\right)+\frac{\partial l}{\partial \mu_B}\frac{\partial \mu_B}{\partial x_i}\\ &= \frac{\partial l}{\partial \hat{x}_i}\frac{1}{\sqrt{\sigma_B^2+\epsilon}}+\frac{\partial l}{\partial \sigma_B^2}\frac{2(x_i-\mu_B)}{m}+\frac{\partial l}{\partial \mu_B}\frac{1}{m}\end{aligned}$$



Backward Pass : Summary

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

References

[1502.03167] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Batch Normalization: Accelerating Deep Network Training ...UW-Madisonhttps://pages.cs.wisc.edu › ~shavlik › lectureNotes