Batch Normalization

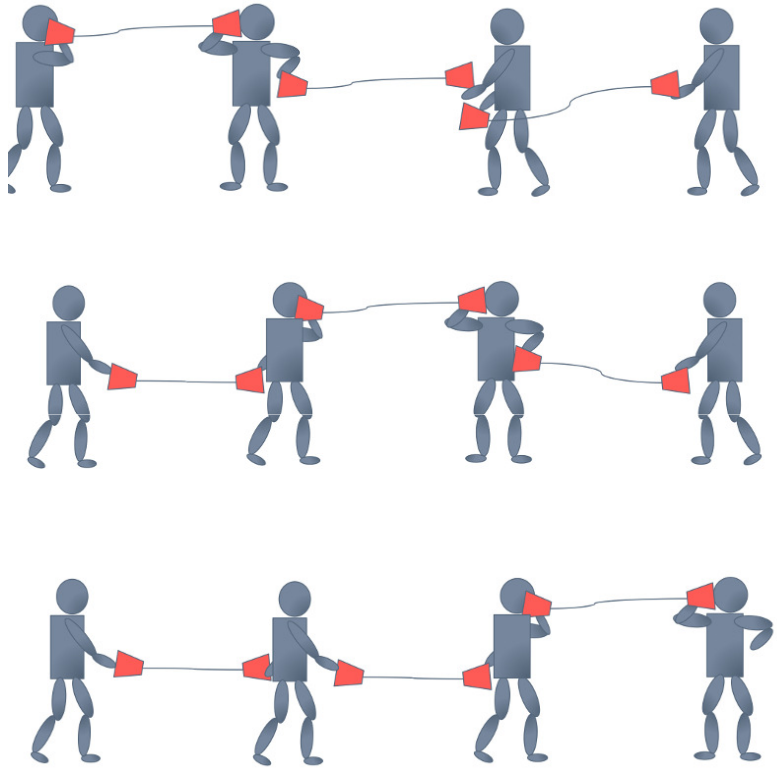
Some slides were adated/taken from various sources, including Andrew Ng’s Coursera Lectures, CS231n: Convolutional Neural Networks for Visual Recognition lectures, Stanford University CS Waterloo Canada lectures, Aykut Erdem, et.al. tutorial on Deep Learning in Computer Vision, Ismini Lourentzou's lecture slide on "Introduction to Deep Learning", Ramprasaath's lecture slides, and many more. We thankfully acknowledge them. Students are requested to use this material for their study only and NOT to distribute it.

Internal Covariate Shift

- The first guy tells the second guy, “go water the plants”, the second guy tells the third guy, “got water in your pants”, and so on until the last guy hears, “kite bang eat face monkey” or something totally wrong.

- Let’s say that the problems are entirely systemic and due entirely to faulty red cups. Then, the situation is analogous to forward propagation

- If can get new cups to fix the problem by trial and error, it would help to have a consistent way of passing messages in a more controlled and standardized (“normalized”) way. e.g. Same volume, same language, etc.



“First layer parameters change and so the distribution of the input to your second layer changes”

Motivation: Batch Normalization

In machine learning algorithms, the functions involved in the optimization process are sensitive to normalization

Example: number of bedrooms {range: 1-5}, size of the house: {500 – 2000 sq. ft.)

- 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.

Good practice for numerical stability for numerical calculations, and to avoid ill-conditioning when solving systems of equations.

Feature Scaling

Feature Scaling

Idea: Make sure features are on a similar scale.

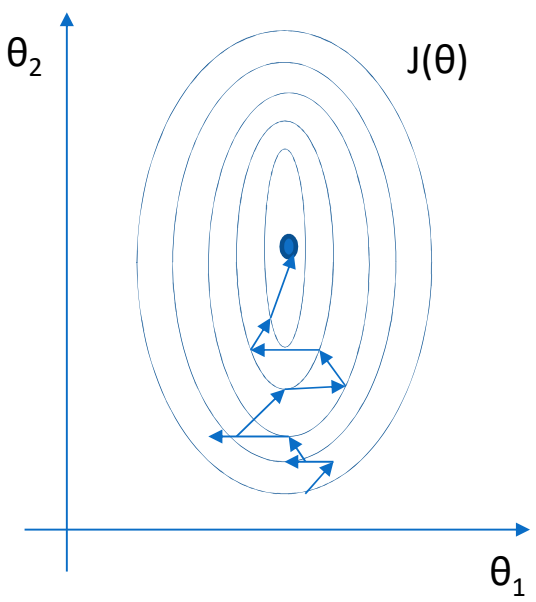
E.g. x_1 = size (0-2000 feet^2)

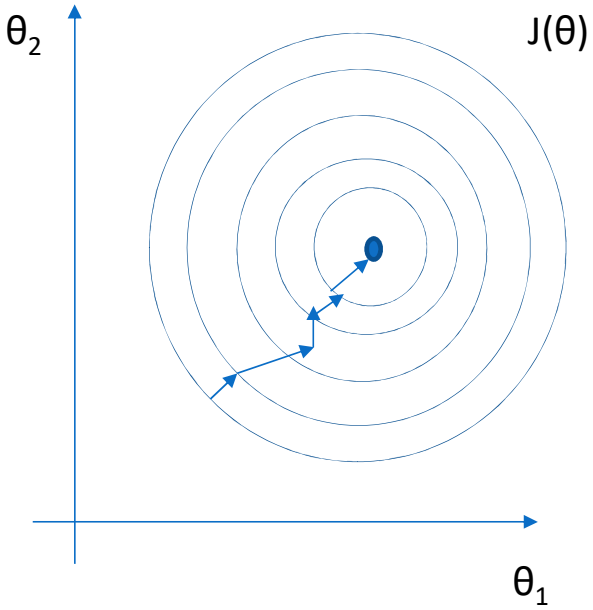
x_2 = number of bedrooms (1-5)

$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000} \qquad 0 \le x_1 \le 1$$

$$x_2 = \frac{\text{number of bedrooms}}{5} \qquad 0 \le x_2 \le 1$$

Get every feature into approximately a $-1 \le x_i \le 1$ range.





Gradient descent converges much faster with feature scaling than without it.

Common Normalization

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:

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

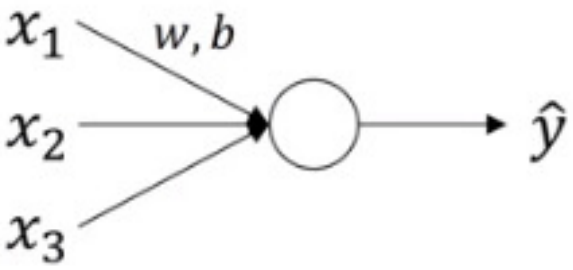
- To have zero mean and unit variance:

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

- In the NN community, this is call *Whitening*

Batch Norm

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$

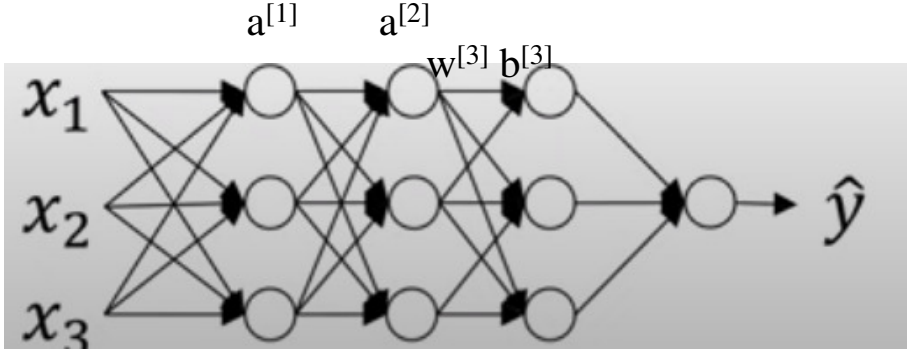


$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$X = X - \mu_B$$

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

$$X = X/\sigma_B^2$$



$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$$

$$a^{[l]} = g^{[l]}(z^{[l]})$$

Normalize Before or after activation

Can we normalize $a^{[2]}$ so as to train $w^{[3]}$ and $b^{[3]}$ faster; Normalize $z^{[2]}$

Implementing Batch Norm

Given some intermediate values in NN $z^{(1)}, z^{(2)}, \ldots \ldots z^{(n)}$

$$\mu = \frac{1}{m}\sum_i z^{(i)} \qquad where\ z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$$

$$\sigma^2 = \frac{1}{m}\sum_i (z_i - \mu)^2$$

$$z^i_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

Normalization step

$$\tilde{z}^i = \gamma\, z^i_{norm} + \beta$$

where γ and β are learnable parameters which can control the means and variance of the normalized z

Batch Nornalization

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

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

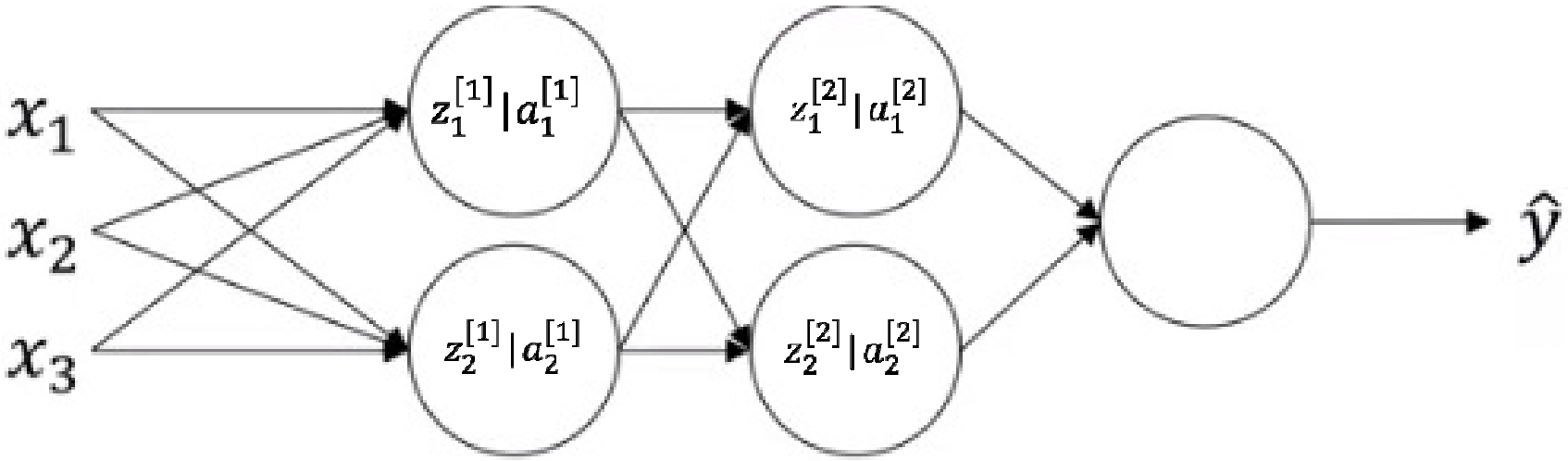
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

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

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

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

Adding BN to NN



$$X \xrightarrow{W^{[1]}, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Normalization}]{\beta^{[1]}, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow a^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]}, b^{[2]}} Z^{[2]} \xrightarrow[\text{Batch Normalization}]{\beta^{[2]}, \gamma^{[2]}} \tilde{Z}^{[2]} \rightarrow a^{[2]}$$

Parameter Set for batch Normalization

$$\beta^{[1]}, \gamma^{[1]} \beta^{[2]}, \gamma^{[2]} \beta^{[3]}, \gamma^{[3]} \dots \dots \dots$$

$$W^{[1]}, b^{[1]} W^{[2]}, b^{[2]} W^{[3]}, b^{[3]} \dots \dots \dots \; W^{[l]}, b^{[l]}$$

Working with Mini Batches

$$X^{\{1\}} \xrightarrow{W^{[1]}, b^{[1]}} Z^{[1]} \xrightarrow{\beta^{[1]}, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow a^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]}, b^{[2]}} Z^{[2]} \xrightarrow{\beta^{[2]}, \gamma^{[2]}} \tilde{Z}^{[2]} \rightarrow a^{[2]}$$

Batch Normalization — Batch Normalization

$$X^{\{2\}} \xrightarrow{W^{[1]}, b^{[1]}} Z^{[1]} \xrightarrow{\beta^{[1]}, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow a^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]}, b^{[2]}} Z^{[2]} \xrightarrow{\beta^{[2]}, \gamma^{[2]}} \tilde{Z}^{[2]} \rightarrow a^{[2]}$$

Batch Normalization — Batch Normalization

$$X^{\{3\}} \xrightarrow{W^{[1]}, b^{[1]}} Z^{[1]} \xrightarrow{\beta^{[1]}, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow a^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]}, b^{[2]}} Z^{[2]} \xrightarrow{\beta^{[2]}, \gamma^{[2]}} \tilde{Z}^{[2]} \rightarrow a^{[2]}$$

Batch Normalization — Batch Normalization

Working with Mini Batches

$$z^i_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z}^i = \gamma\, z^i_{norm} + \beta$$

Parameters: $W^{[l]}, b^{[l]}, \beta^{[l]}, \gamma^{[l]}$

$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$$

$$z^{[l]} = W^{[l]} a^{[l-1]}$$

$$z^i_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z}^i = \gamma\, z^i_{norm} + \beta$$

Batch Normalization: why 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, i.e. its usually not active, sometimes a bit active, rare very active.

Covariate Shift is undesirable, because the later layers have to keep adapting to the change of the type of distribution (instead of just to new distribution parameters, e.g. new mean and variance values for Gaussian distributions).

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.

to continue…