Batch Normalization

Fitting Batch Norm into a neural network

deeplearning.ai

Adding Batch Norm to a network
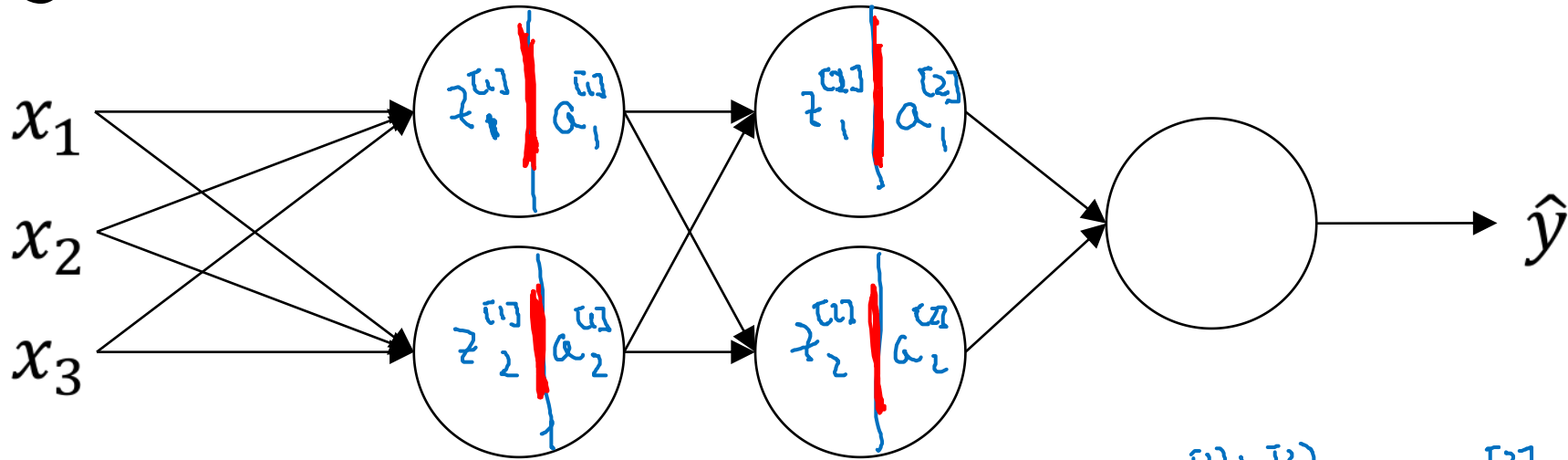
$$X \xrightarrow{W^{[1]}, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm (BN)}]{\beta^{[1]}, \gamma^{[1]}} \tilde{Z}^{[1]} \rightarrow a^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]}, b^{[2]}} Z^{[2]} \xrightarrow[\text{BN}]{\beta^{[2]}, \gamma^{[2]}} \tilde{Z}^{[2]} \rightarrow a^{[2]} \rightarrow \cdots$$

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, \ldots, W^{[L]}, b^{[L]},$

$\rightarrow \beta^{[1]}, \gamma^{[1]}, \beta^{[2]}, \gamma^{[2]}, \ldots, \beta^{[L]}, \gamma^{[L]}$

$d\beta^{[L]}$

$\beta^{[L]} = \beta^{[L]} - \alpha \, d\beta^{[L]}$

$\rightarrow \beta$

tf.nn.batch-normalization $\leftarrow$

Andrew Ng

Working with mini-batches

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

$X^{\{2\}} \rightarrow Z^{[1]} \xrightarrow[\boxed{BN}]{\beta^{[1]}, \gamma^{[1]}} \tilde{Z}^{[1]} \rightarrow \dots$

$X^{\{3\}} \rightarrow \dots$

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

$(n^{[l]}, 1) \quad (n^{[l]}, 1) \quad (n^{[l]}, 1)$

$Z^{[l]}$

$(n^{[l]}, 1)$

$Z^{[l]} = W^{[l]} a^{[l-1]} + \cancel{b^{[l]}}$

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

$Z^{[l]}_{norm}$

$\tilde{Z}^{[l]} = \gamma^{[l]} Z^{[l]}_{norm} + \beta^{[l]}$

Andrew Ng

Implementing gradient descent

for $t = 1$ .... num MiniBatches

    Compute forward prop on $X^{\{t\}}$.

      In each hidden layer, use BN to replace $Z^{[l]}$ with $\tilde{Z}^{[l]}$.

    Use backprop to compute $dW^{[l]}, \cancel{db^{[l]}}, d\beta^{[l]}, d\gamma^{[l]}$

    Update parameters
$$\left.\begin{array}{l} W^{[l]} := W^{[l]} - \alpha \, dW^{[l]} \\ \beta^{[l]} := \beta^{[l]} - \alpha \, d\beta^{[l]} \\ \gamma^{[l]} := \cdots \end{array}\right\} \longleftarrow$$

Works w/ momentum, RMSprop, Adam

Andrew Ng