C2W3- Batch normalization

Normalization makes learning faster as it modified the contours line of cost function as a more round shape

$$\mu = \frac{1}{m}\sum_i x^{(i)}$$
$$x = X - \mu$$
$$\sigma^2 = \frac{1}{m}\sum_i (x^{(i)})^2$$
$$X = X/\sigma^2$$

not normalized

normalized

Batch normalization:- previously we only normalized the input feature. But in batch normalization we will also normalize the activation unit (Z) of each neural node.

Implementation:-

For each hidden unit we will calculate $\tilde{Z}$ as follows:-

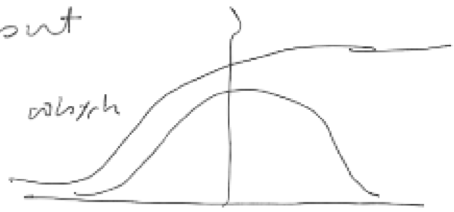
$$\mu = \frac{1}{m}\sum_i z^{(i)}$$
$$\sigma^2 = \frac{1}{m}\sum_i (z_i - \mu)^2$$
$$z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$
$$\tilde{z}^{(i)} = \gamma z_{norm} + \beta$$ | γ, β = learnable parameter

for next computation we will use $\tilde{z}^{(i)}$ instead of $z^{(i)}$

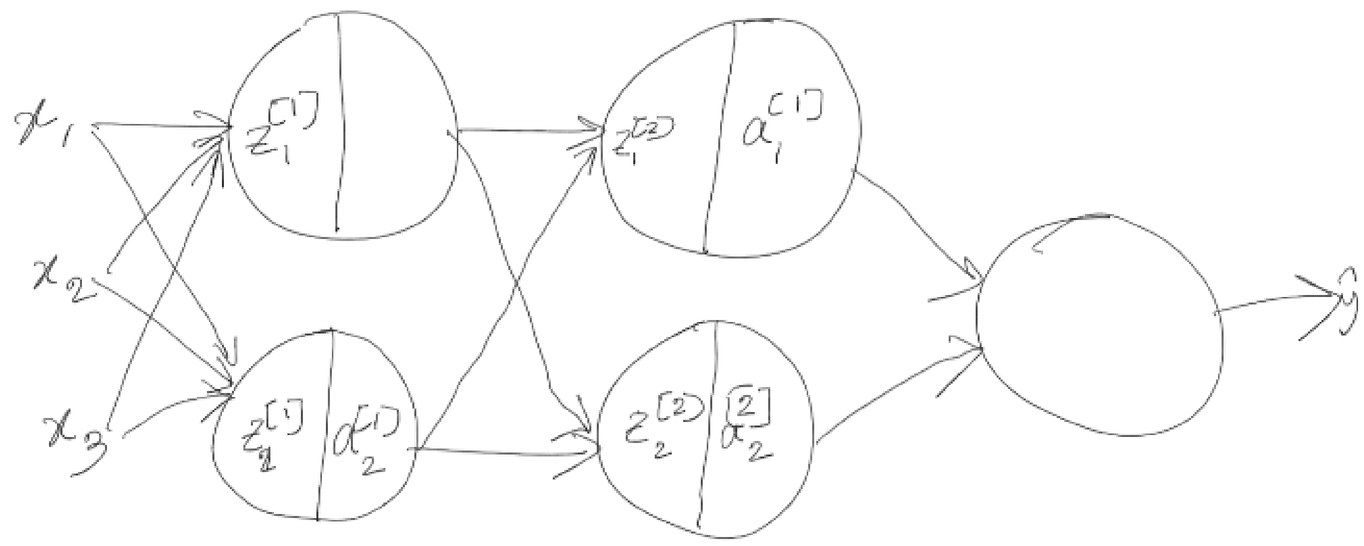
the reason of using γ & β :-

we don't want to force the mean of $\tilde{z}^{(i)}$ to be zero, we might loose the nonlinearity of the sigmoid function

γ & β doesn't set the mean=0 but it standarize the mean & variance which we can control now



Adding batch norm to a network:-



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

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

Working with minibatch:-

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

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

→ this BN takes the mean & variance from this $z^{[1]}$ only.

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

⇒ During normalization of minibatch all constant will be canceled out. so this b will have no effect.

if we implement batch norm we can elemenate the parameter b.

implementing Gradient descend:-

for t = 1 .... num of 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 parameter:-

$$w^{[l]} := w^{[l]} - \alpha\, dw^{[l]}.$$
$$\beta^{[l]} := \beta^{[l]} - \alpha\, d\beta^{[l]}.$$
$$\gamma^{[l]} := \gamma^{[l]} - \alpha\, d\gamma^{[l]}$$

This also works with momentum, adam/RMSprop.

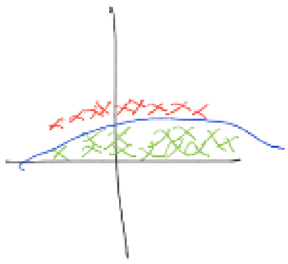
Why batch norm works:-

① Makes the mean zero and provide a better shape for the contour line of cost function

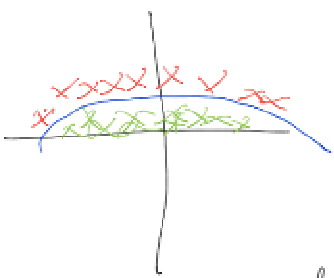
② Reduce the effect of covariate shift.

③ Get slight regularization effect as a side-effect

if we trained our network to identify only black cat then if we try to identify the non black cat then it will not perform better.



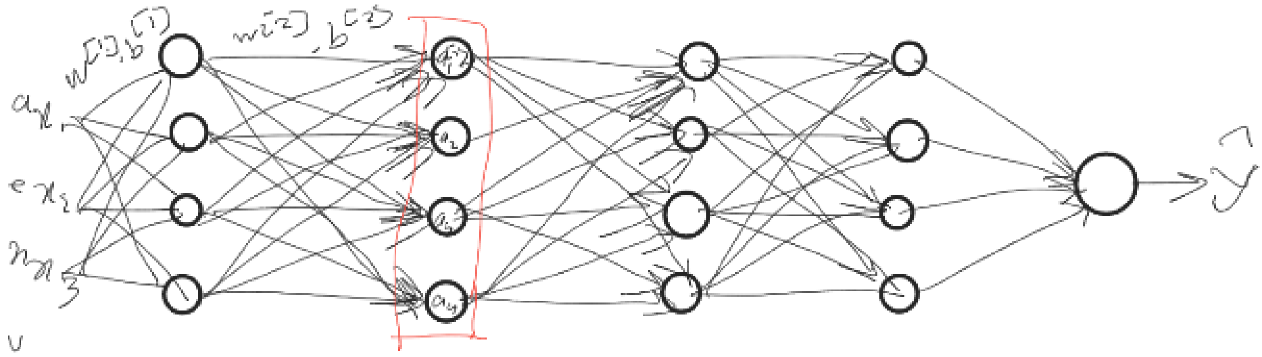
black cat



non-black cat

Even if black cat & non black cat use almost same function for decision boundary. We cannot use black cat model to non black cat model. because of covariate shift. The idea of co-variate shift is if if we use mapping $(x \rightarrow y)$ & if the distribution of x changes then we might need to retrain algorithm

Why co-variate shift is a problem of NN:-



if $w_{1/2}$ & $b_{1/2}$ changes then value of a_i also changes that's covariate shift. But if we use batch normalization then effect of covariate shift doesn't effect that much. Doesn't matter how a_i changes batch norm makes sure it's mean & variances stays the same

Batch norm as regularization effects:-

- Each mini batch is scaled by mean/variance computed on just that minibatch.
- This adds some noise to the value of $z^{[l]}$ within that minibatch. So similar to dropout it adds some noise to each hidden layer activations.
- This has a slight regularization effects.

Regularization is just a side effect of batch norm.

its main job is the normalizing the data of hidden unit for faster learning.

Batch norm at test time:-

in train time we have multiple train data so we can calculate mean, standard deviation thus normalize the data. But in test time we have only one example. so to calculate, μ, σ^2 we do:-

Use the exponentially weighted average of μ, σ^2 from minibatches $\mu^{[l]}$ = exponentially weighted average of $\{\mu^{\{1\}[l]}, \mu^{\{2\}[l]}, \ldots\}$

$\sigma^{2[l]}$ = exponentially weighted average of $\{\sigma^{2\{1\}[l]}, \sigma^{2\{2\}[l]}, \ldots\}$