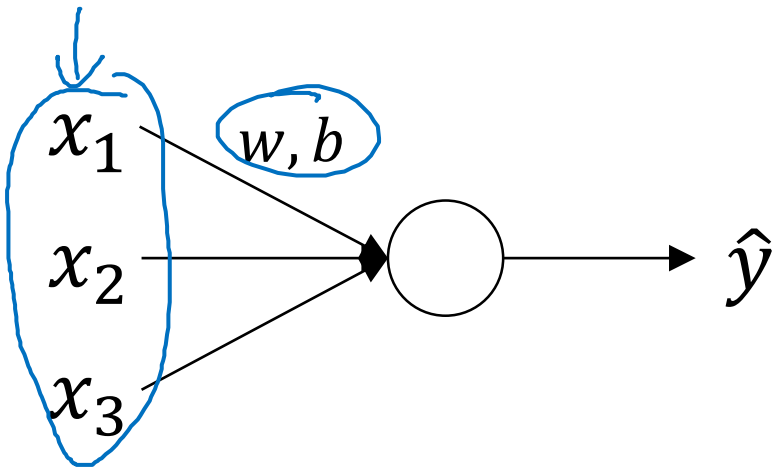
deeplearning.ai

Batch Normalization

Normalizing activations in a network

Batch normalization makes your hyperparameter search problem much easier, makes the neural network much more robust to the choice of hyperparameters, there's a much bigger range of hyperparameters that work well and also enable you to much more easily train even very deep networks.

Normalizing inputs to speed up learning



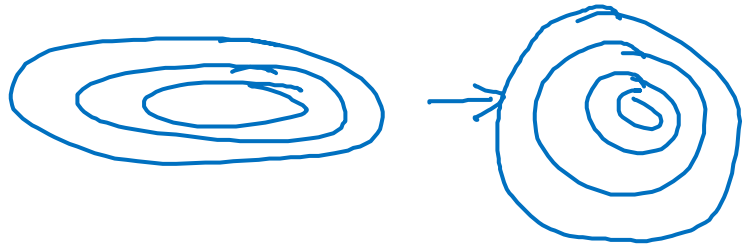
Normalizing the input features can speed up learnings

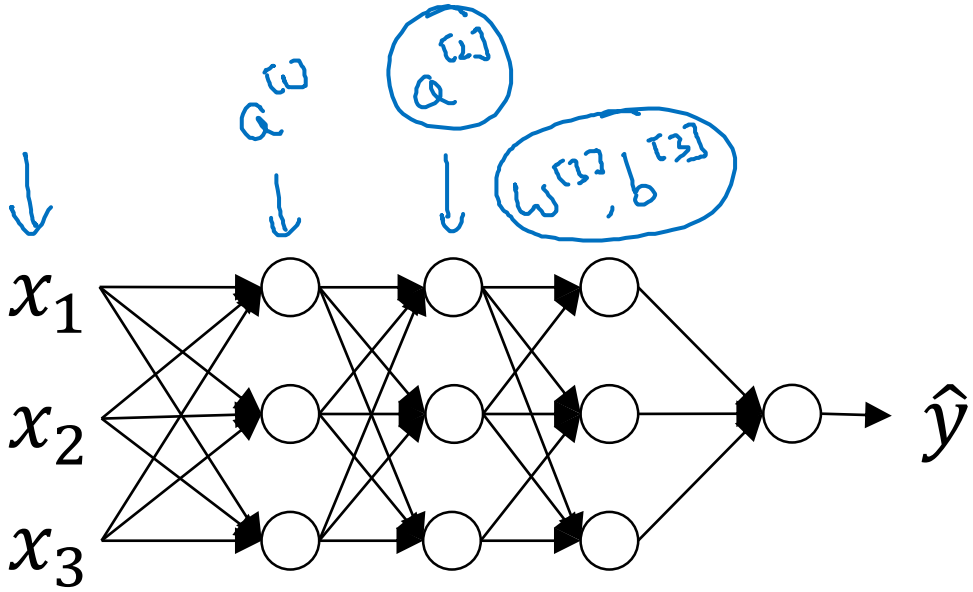
$$\mu = \frac{1}{m}\sum_i x^{(i)}$$

$$X = X - \mu \quad \text{element-wise}$$

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

$$X = X / \sigma^2$$





Can we normalize $a^{[2]}$ so as to train $w^{[3]}, b^{[3]}$ faster

Normalize $z^{[2]}$

Whether you should normalize the value after applying the activation function a[2] or not.

In practice, normalizing Z[2] is done much more often, so that's the version I presented what I would recommend you use as the default choice.

Andrew Ng

Implementing Batch Norm

Now, notice that the effect of gamma and beta is that it allows you to set the mean of Ztilde to be whatever you want it to be.

Given some intermediate values in NN

$z^{(1)}, \ldots, z^{(m)}$

$z^{[l](i)}$

$\mu = \frac{1}{m} \sum_i z^{(i)}$

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

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

for numerical stability, just in case sigma squared turns to be zeros

$\tilde{z}^{(i)} = \gamma z^{(i)}_{norm} + \beta$

learnable parameters of model.

You would update the parameters gamma and beta just as you update weights of the neural network.

If

$\gamma = \sqrt{\sigma^2 + \varepsilon}$

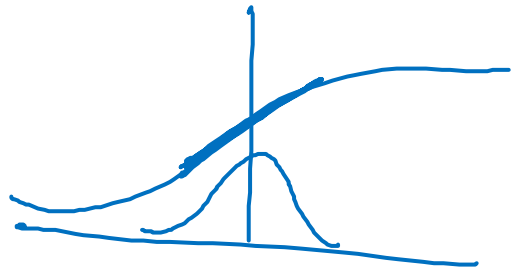
$\beta = \mu$

then $\tilde{z}^{(i)} = z^{(i)}$

If gamma were equal to this denominator term and if beta were equal to mu. Then the effect of gamma Znorm plus beta is that it would exactly invert this equation.

$X \leftarrow$

$z^{(i)} \leftarrow$



Use $\tilde{z}^{[l](i)}$ instead of $z^{[l](i)}$.

And so now we've taken these values Z and normalized them to have mean zero and standard unit variance. So every component of Z has mean zero and variance one. But we don't want the hidden units to always have mean zero and variance one. Maybe it makes sense for hidden units to have a different distribution.

Andrew Ng

And so by an appropriate setting of the parameters gamma and beta, this normalization step, that is these four equations, is just computing essentially the identity function. By choosing other values of gamma and beta, this allows you to make the hidden unit values of other means and variance as well.

And what Batch Norm does is it applies that normalization process not just to the input layer, but to the values even deep in some hidden layer in the neural networks.
So you apply this type of normalization to normalize the mean and variance of some of your hidden units values Z.
But one difference between the training input and these hidden unit values is you might not want your hidden unit values to be forced to mean zero and variance one.

Or what it does really is that it ensures that your hidden units have standardized mean and variance, where the mean and variance are controlled by two explicit parameters gamma and beta, which the learning algorithm can set to whatever it wants.

So what it really does is it normalizes the mean and variance of these hidden unit values, really, the Z[i], to have some fixed mean and variance. And that mean the variance can be zero and one or it could be some other value and it's controlled by these parameters gamma and beta.