

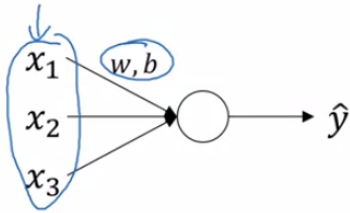
Week 3-2 Batch Normalization

笔记本: DL 2 - Deep NN Hyperparameter Tuning, Regularization & Optimization

创建时间: 2021/1/9 13:57

更新时间: 2021/1/9 14:33

Normalizing inputs to speed up learning



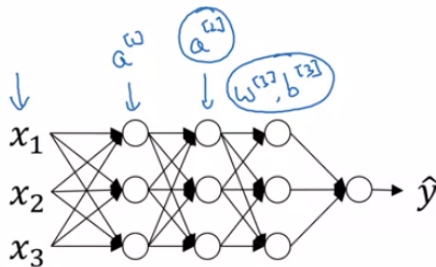
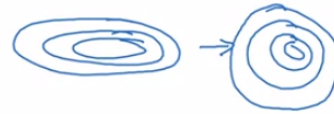
$$\mu = \frac{1}{m} \sum_i x^{(i)}$$

$$X = X - \mu$$

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

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

← element-wise



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

Normalize $z^{[2]}$

Implementing Batch Norm

Given some intermediate values in NN

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

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

$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

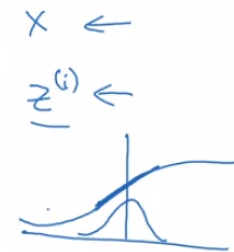
$$\hat{z}^{(i)} = \gamma z_{\text{norm}}^{(i)} + \beta$$

If $\gamma = \sqrt{\sigma^2 + \epsilon}$

$\beta = \mu$

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

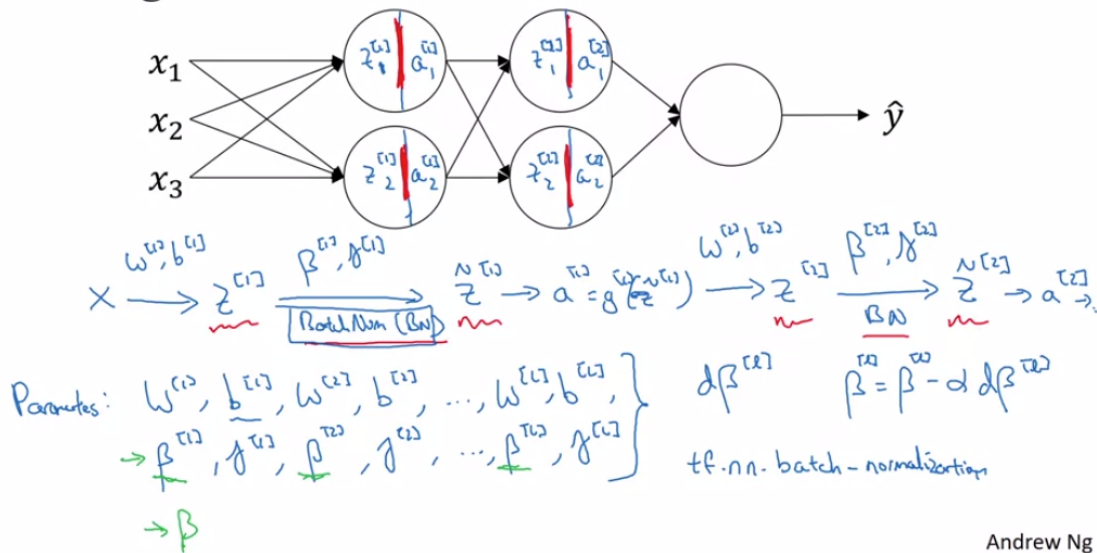
learnable parameters of model.



Use $\hat{z}^{(i)}$ instad of $z^{(i)}$.

So what it really does is it normalizes in mean and variance of these hidden unit values, really the Zs, to have some fixed mean and variance.

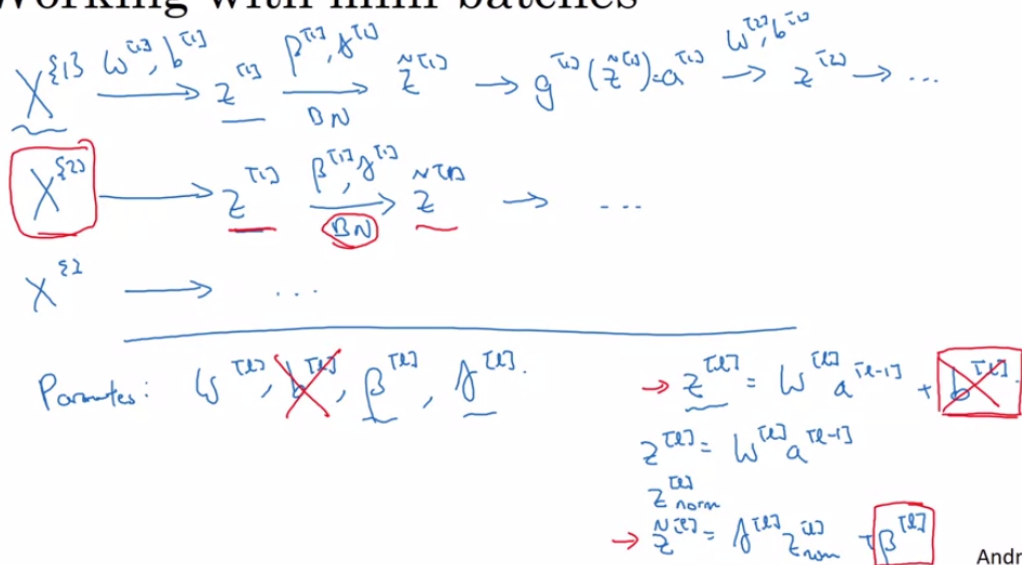
Adding Batch Norm to a network



Andrew Ng

Always work with mini-batches

Working with mini-batches



Andrew Ng

normalization zeroes out the mean,
so there is no point to have b

Implementing gradient descent

for $t = 1 \dots \text{num Mini Batches}$
Compute forward pass on X^{set} .
In each hidden layer, use BN to rep $\underline{z}^{\text{set}}$ with $\underline{\hat{z}}^{\text{set}}$.
Use backprop to compute $\underline{dw}^{\text{set}}$, ~~$\underline{d\beta}^{\text{set}}$~~ , $\underline{d\beta}^{\text{set}}$, $\underline{d\gamma}^{\text{set}}$
Update params $\left. \begin{aligned} W^{\text{set}} &:= W^{\text{set}} - \alpha \underline{dw}^{\text{set}} \\ \beta^{\text{set}} &:= \beta^{\text{set}} - \alpha \underline{d\beta}^{\text{set}} \\ \gamma^{\text{set}} &:= \dots \end{aligned} \right\} \leftarrow$
Works w/ momentum, RMSprop, Adam.

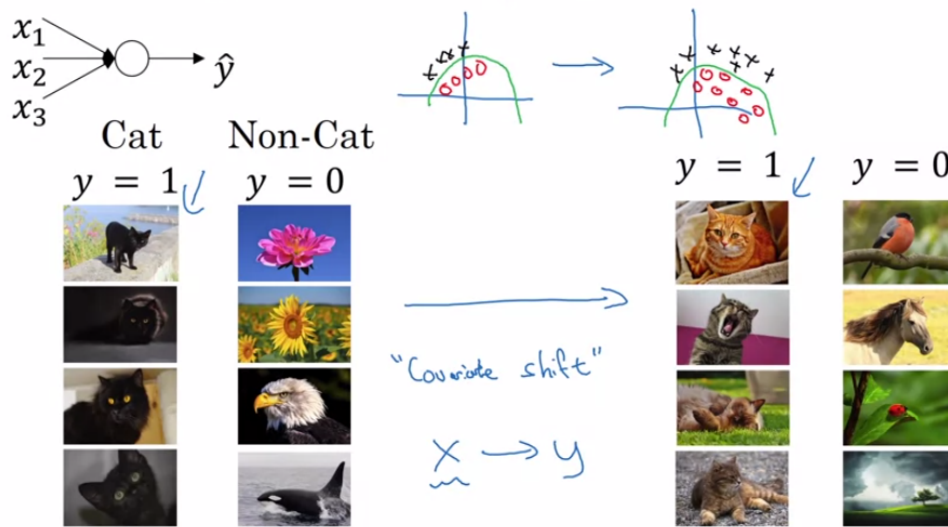
Why batch normalization works?

Here's one reason, you've seen how normalizing the input features, the X 's, **to mean zero and variance one**, how that can speed up learning. So, one intuition behind why batch norm works is, this is doing a similar thing, but further values in your hidden units and not just for your input there.

A second reason why batch norm works, is it makes **weights**, later or

deeper than your network, say the weight on layer 10, more **robust to changes to weights in earlier layers** of the neural network, say, in layer one.

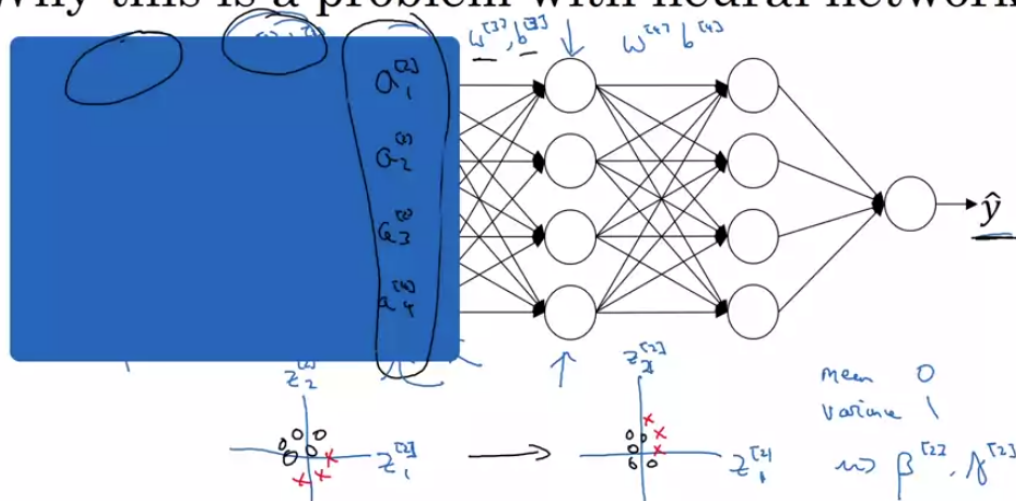
Learning on shifting input distribution



Andrew Ng

covariate shift (if distribution of training shifts, we need to re-train)

Why this is a problem with neural networks?



even as the earlier layers keep learning, the amounts that this forces the later layers to adapt to as early as layer changes is reduced

Regularization

Batch Norm as regularization

- Each mini-batch is scaled by the mean/variance computed on just that mini-batch. $\tilde{z}^{[L]}$
- This adds some noise to the values $z^{[l]}$ within that minibatch. So similar to dropout, it adds some noise to each hidden layer's activations. $\{x_i\}$
 μ, σ^2
- This has a slight regularization effect.

mini-batch: 64 \longrightarrow 512

(since mean and variance are approximated by only a mini-batch of data), larger batch size \rightarrow regularization worse

Test time

Batch norm handles data one mini-batch at a time. It computes mean and variances on mini-batches. So at test time, you try and make predictors, try and evaluate the neural network, you might be processing one single example at the time. So, at test time you need to do something slightly differently to make sure your predictions make sense.

Batch Norm at test time

→ $\mu = \frac{1}{m} \sum_i z^{(i)}$

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

→ $z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$ ←

→ $\tilde{z}^{(i)} = \gamma z_{\text{norm}}^{(i)} + \beta$

μ, σ^2 : estimate using exponentially weighted average (across mini-batches).

$X^{(1)}, X^{(2)}, X^{(3)}, \dots$

↓

$\mu^{(1)}[1], \mu^{(2)}[1], \mu^{(3)}[1] \rightarrow \mu$

$\sigma^{(1)}[1], \sigma^{(2)}[1], \sigma^{(3)}[1] \rightarrow \sigma^2$

$\tilde{z}_{\text{norm}} = \frac{z - \mu}{\sqrt{\sigma^2 + \epsilon}}$

$\tilde{z} = \gamma \tilde{z}_{\text{norm}} + \beta$

Andrew Ng

using exponentially weighted average for μ and σ^2 at test time

