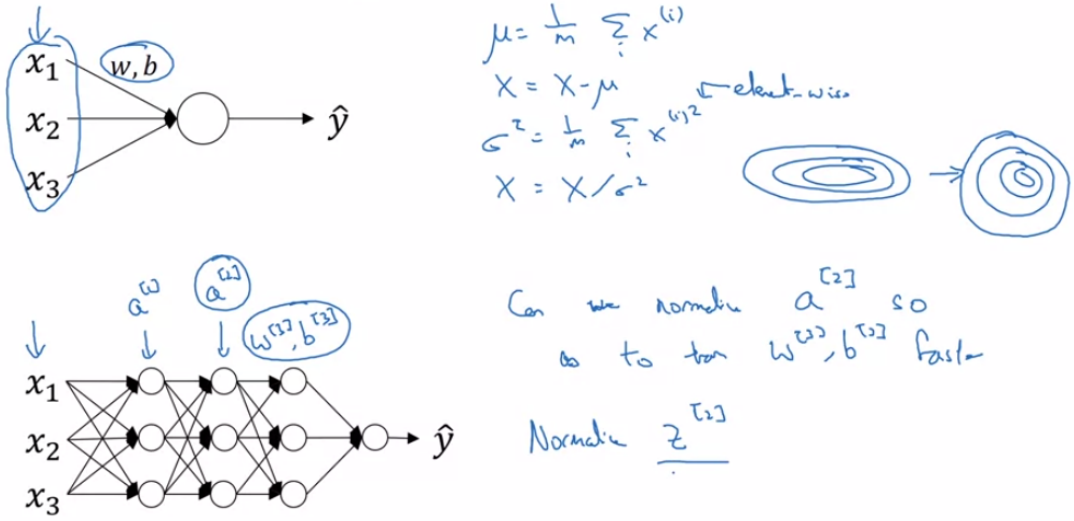
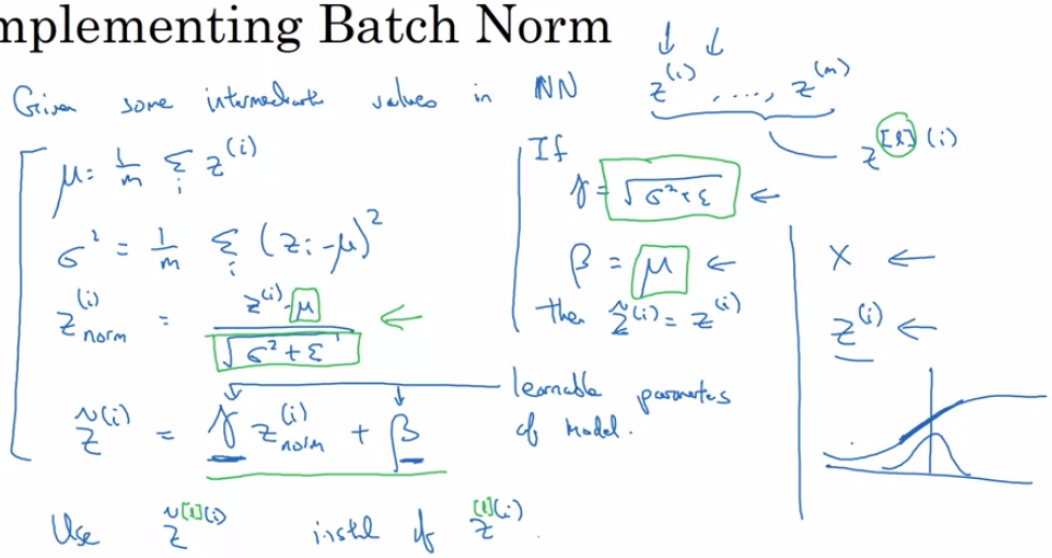
Week 3-2 Batch Normalization

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

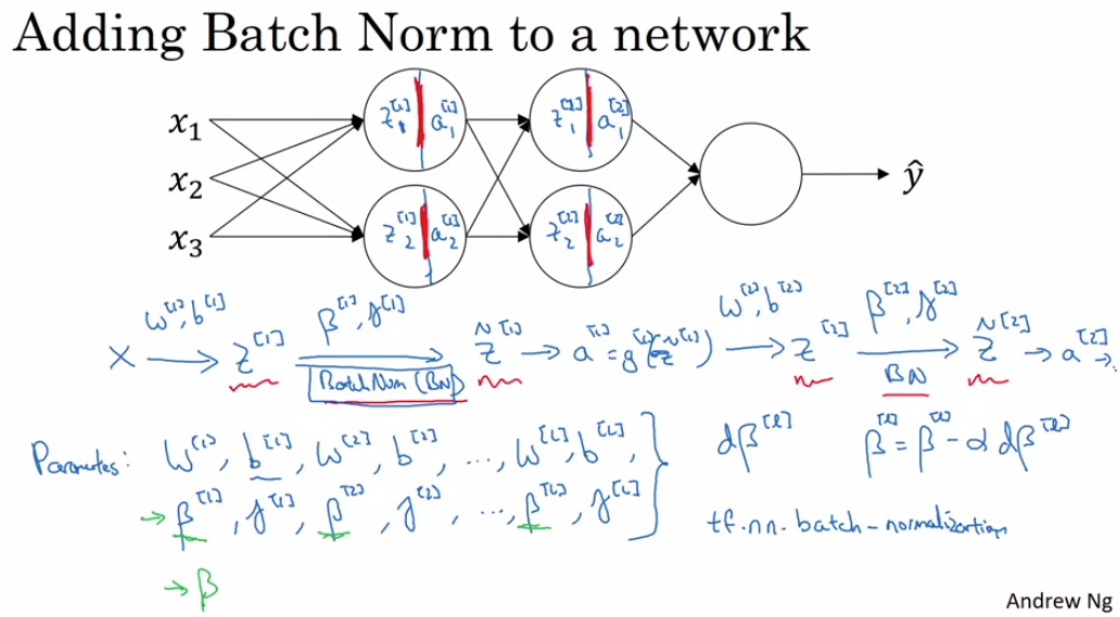
Normalizing inputs to speed up learning



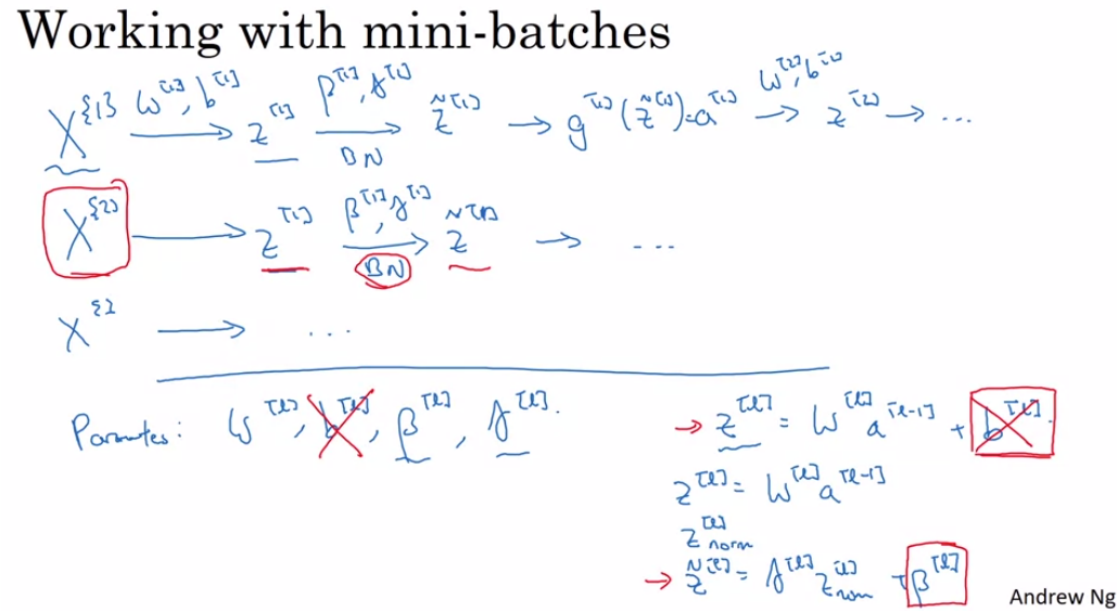
Implementing Batch Norm



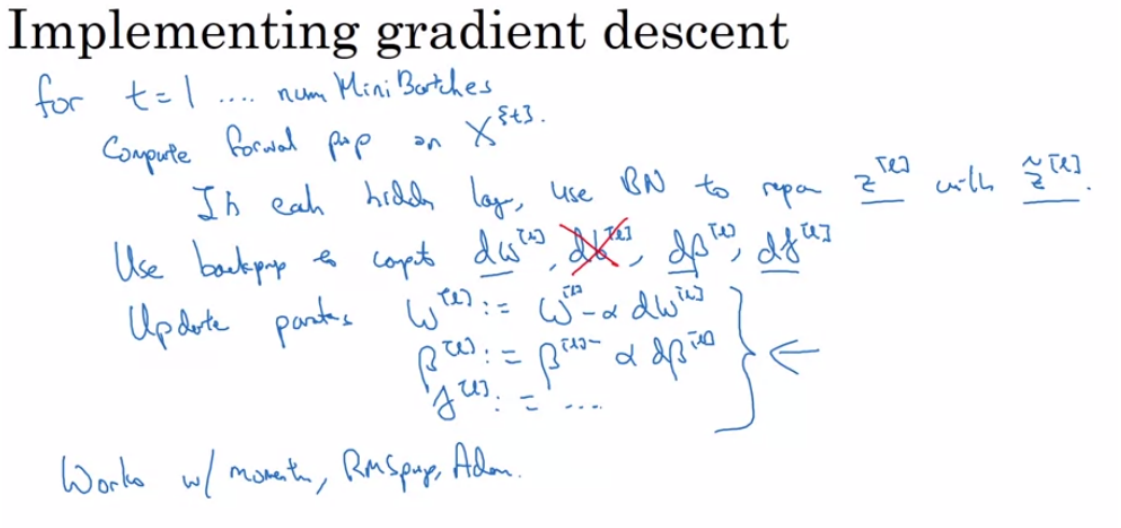
So what it really does is it normalizes in mean and variance of these hidden unit values, really the Zs, to have some fixed mean and variance.



Always work with mini-batches



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

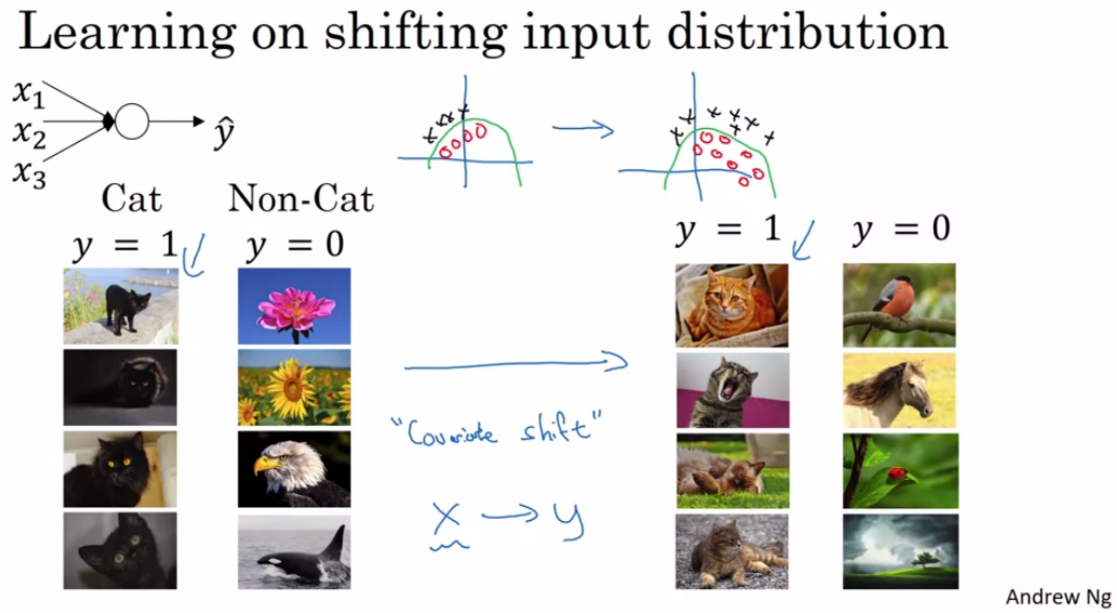


Why batch normalization works?

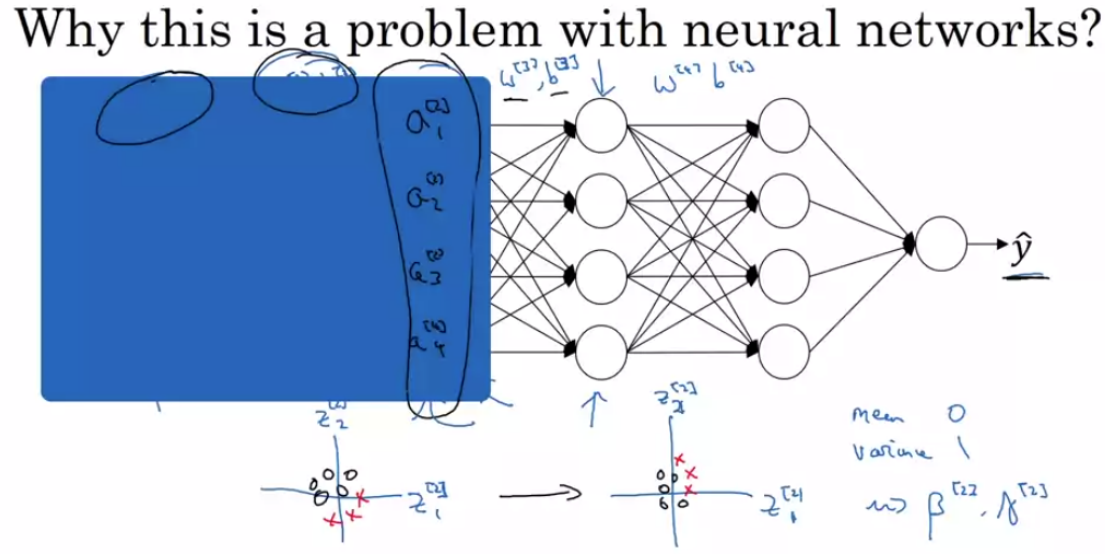
Here's one reason, you've seen how normalizing the input features, the X's, **to mean zero and variance one**, how that can speed up learning. So, one intuition behind why batch norm works is, this is doing a similar thing, but further values in your hidden units and not just for your input there.

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.



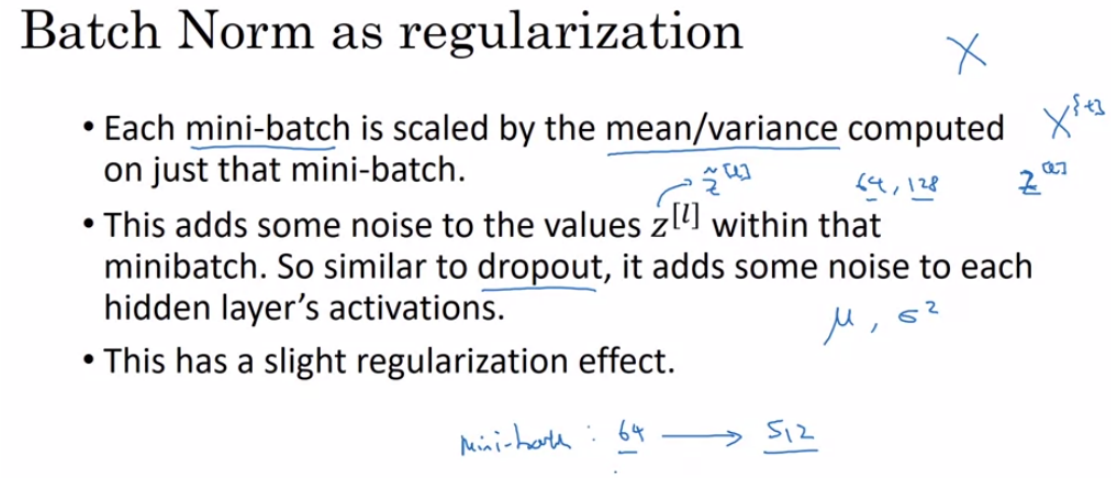
covariate shift (if distribution of training shifts, we need to re-train)



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

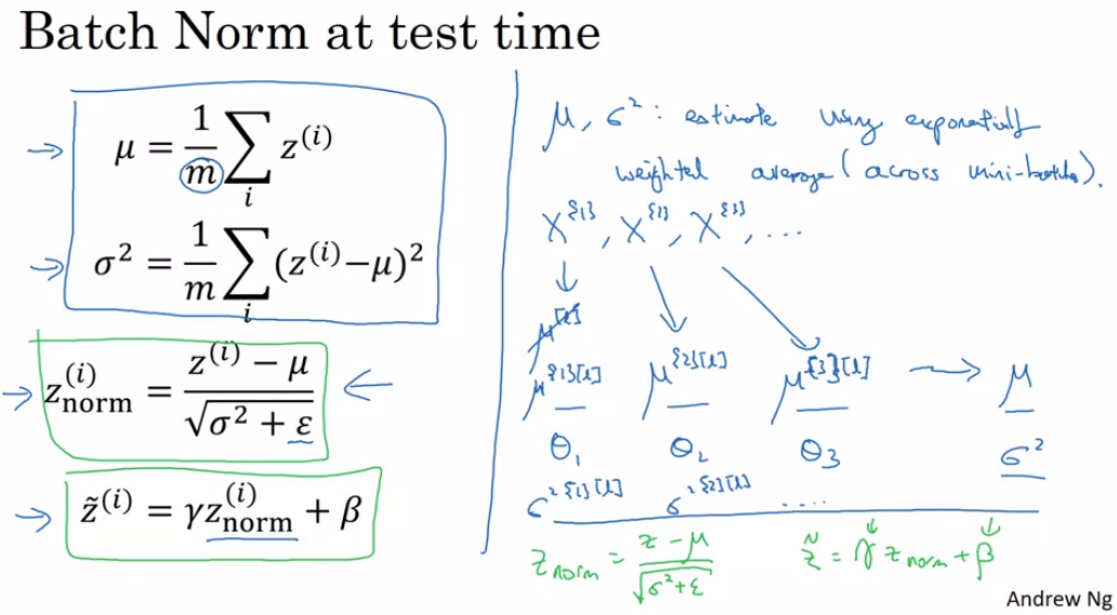


(since mean and variance are approximated by only a mini-batch of data), larger batch size -> regularization worse

---

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.



using exponentially weighted average for u and sigma squared at test time