Coursera Notes

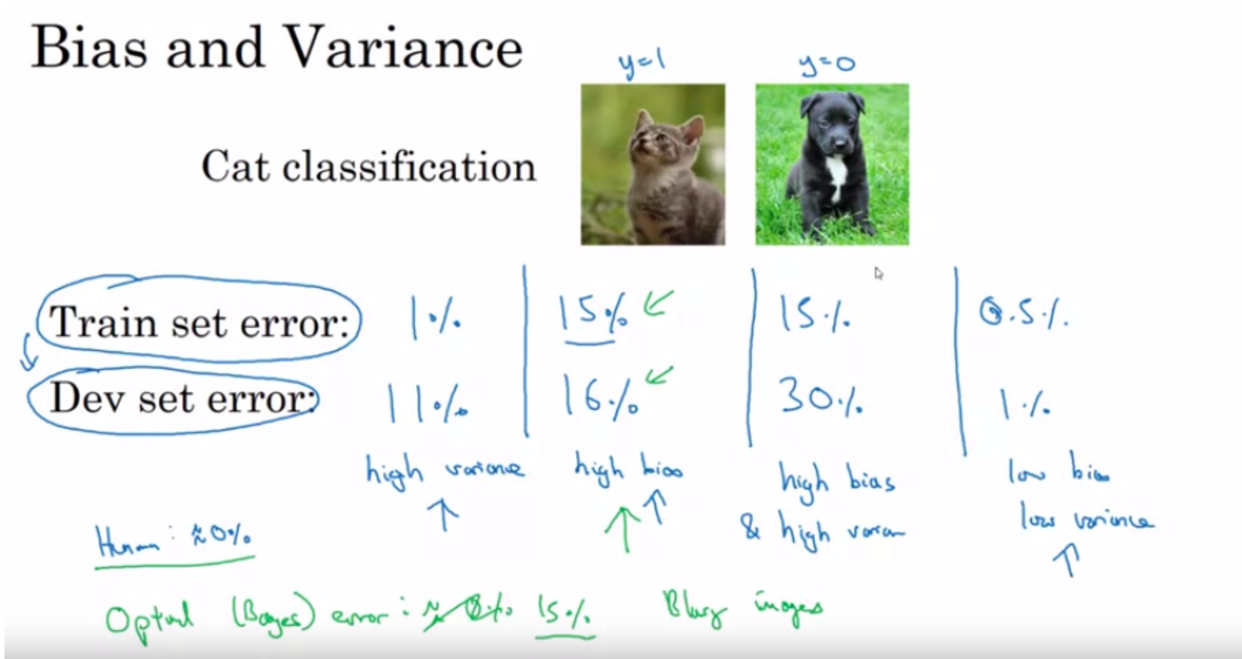
Improving Deep neural Networks: Hyperparameter Tuning, Regularization and Optimization Methods

Setting up an ML application:

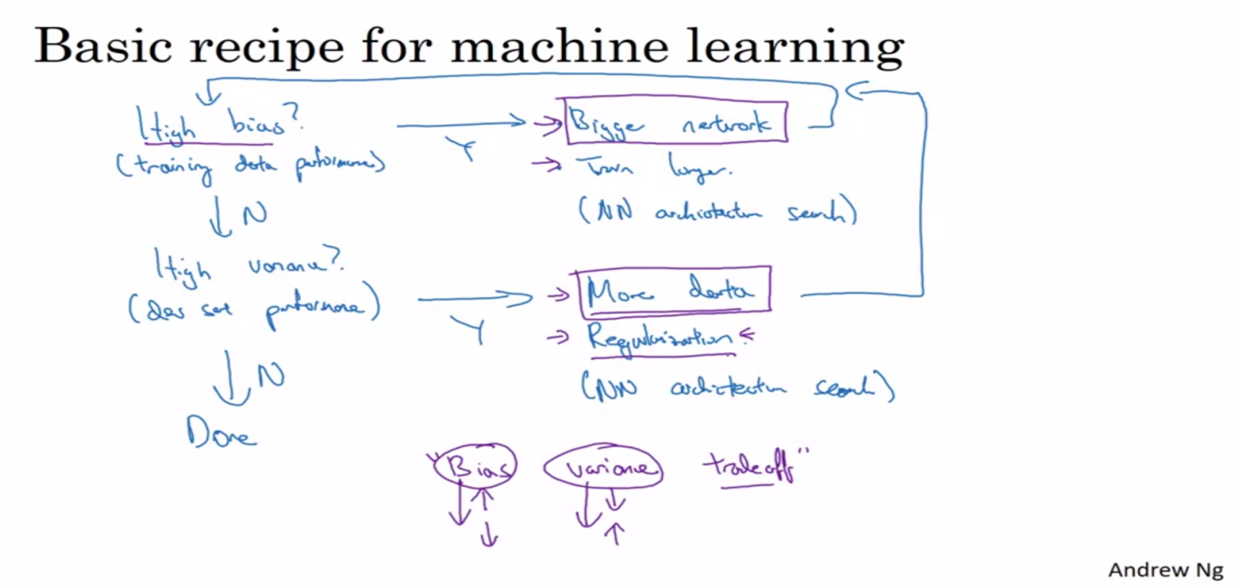
W1L1: train/dev/test sets

- Smaller datasets: 60/20/20 or 70/15/15
- Big datasets: 98/1/1
- Make sure that your dev and test sets come from the same data distributions or source of inputs

W1L2: bias/variance



W1L3:

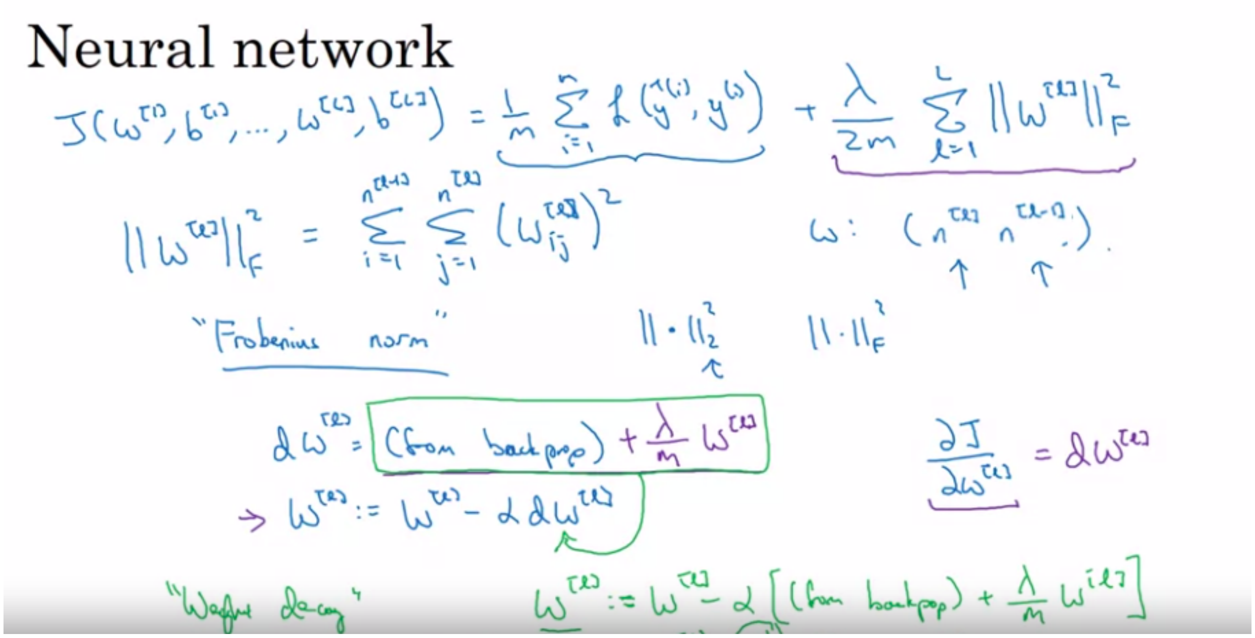


The vector norm $|\mathbf{x}|_p$ for $p = 1$, 2, ... is defined as

$$|\mathbf{x}|_p \equiv \left(\sum_i |x_i|^p\right)^{1/p}.$$

W1L4: Regularization:

- To prevent high variance problem & overfitting: use regularization
- L2 - generally used - also called as weight decay
- L1 - weights will be sparse



W1L5: Why regularization reduces overfitting?

Regularization in machine learning is the process of regularizing the parameters that constrain, regularizes, or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, avoiding the risk of **Overfitting**.

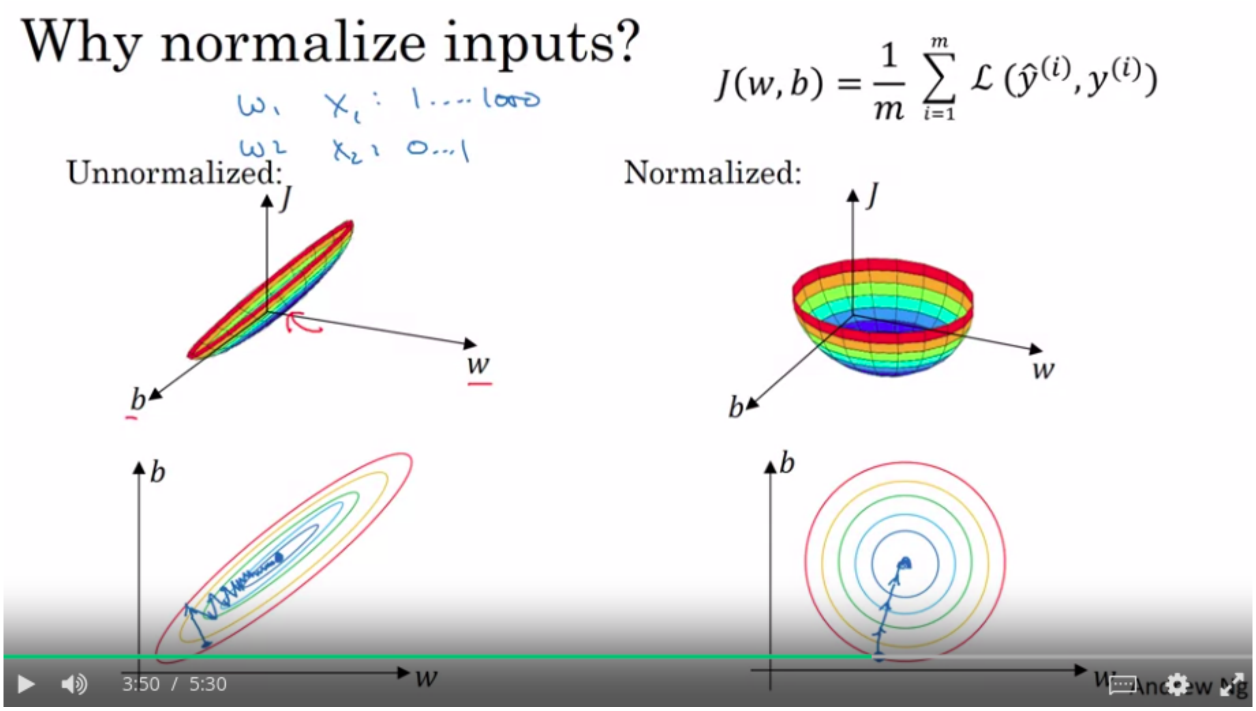
W1L6/7: Dropout regularization

- It helps prevent overfitting as the weights have to shrink when some features are dropped.
- In computer vision applications: dropout is used most often
- Downside: cost function J is not well defined, harder to double-check the performance of the gradient descent model.
- To avoid this: Ng runs the code with dropouts off, keep prob = 1, and then turns the dropout on to avoid introducing bugs.

W1L8: few more regularization techniques:

- Data Augmentation:
- Early stopping:

W1L9: Normalizing the inputs



W1L10: vanishing/exploding gradients:

for a deep neural network with L layers (L is large integer):

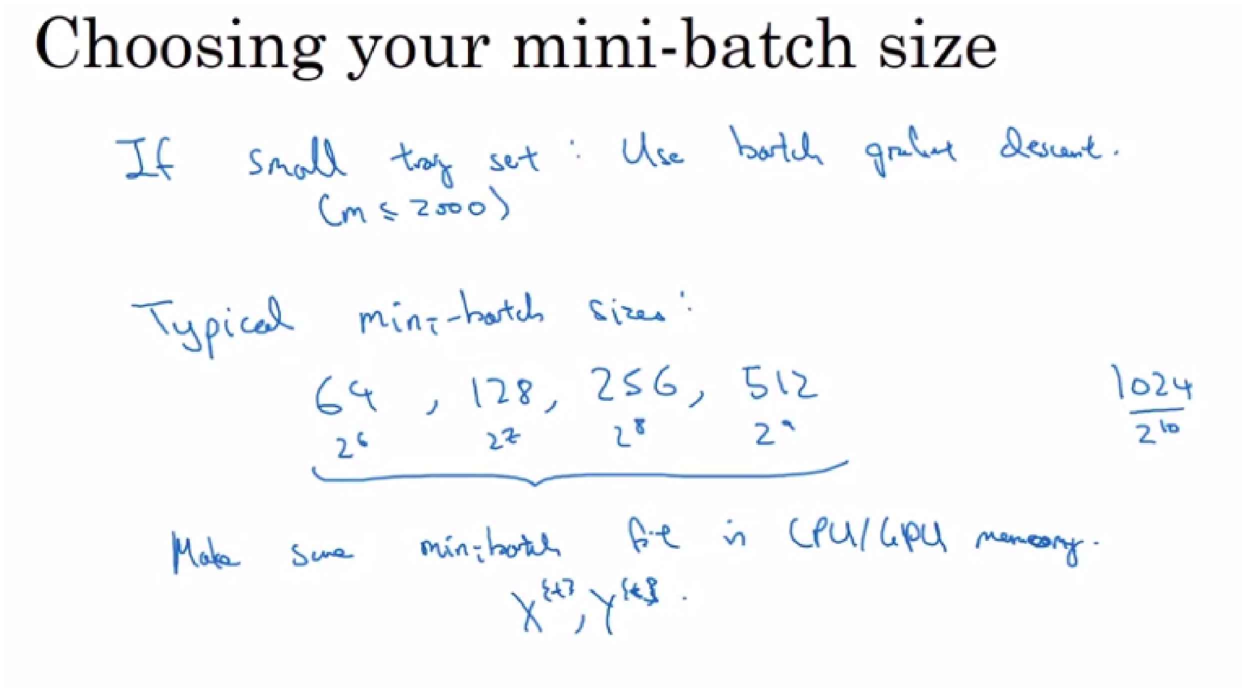
- If W < Identity matrix : vanishing gradients
- If W > identity matrix : exploding gradients as
  W raised to L-1 gives the hypothesis

W1L11: Weight Initialization: To avoid vanishing and exploding gradients

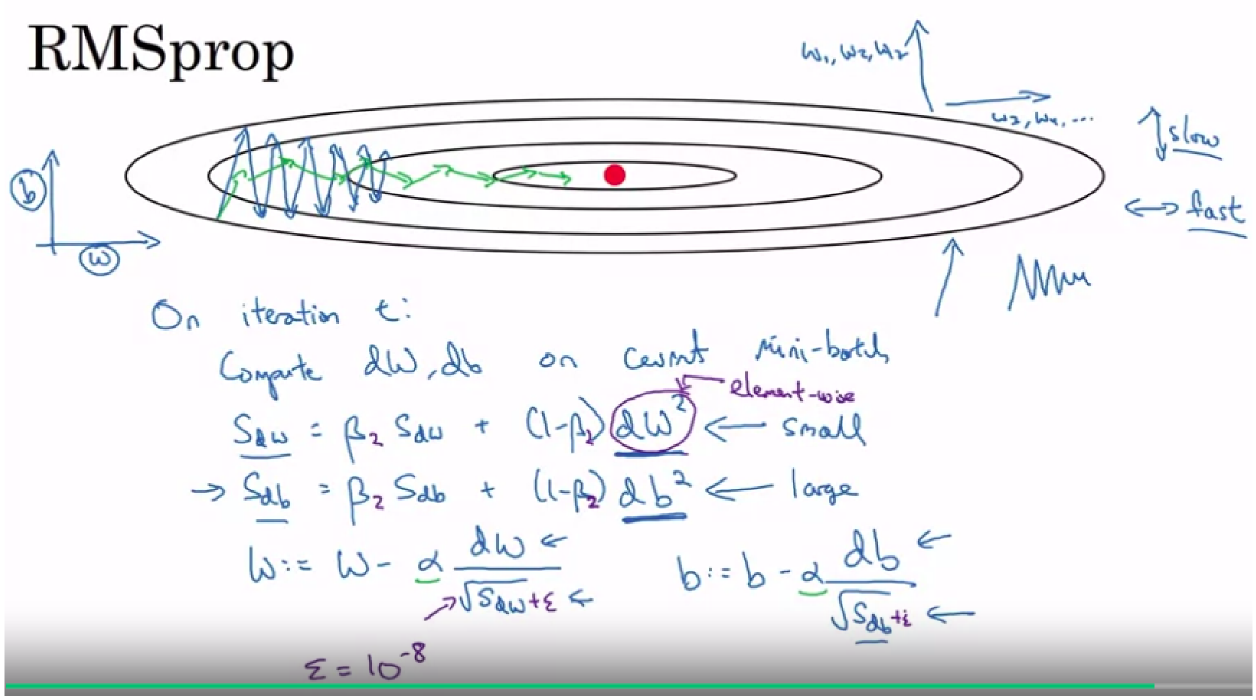
- Relu : W = np.random.rand(shape) * np.sqrt(1/(n^(L-1))) where n: number of hidden units
- Tanh: Xavier Initialiation

W2L1/2: Mini Batch Gradient Descent

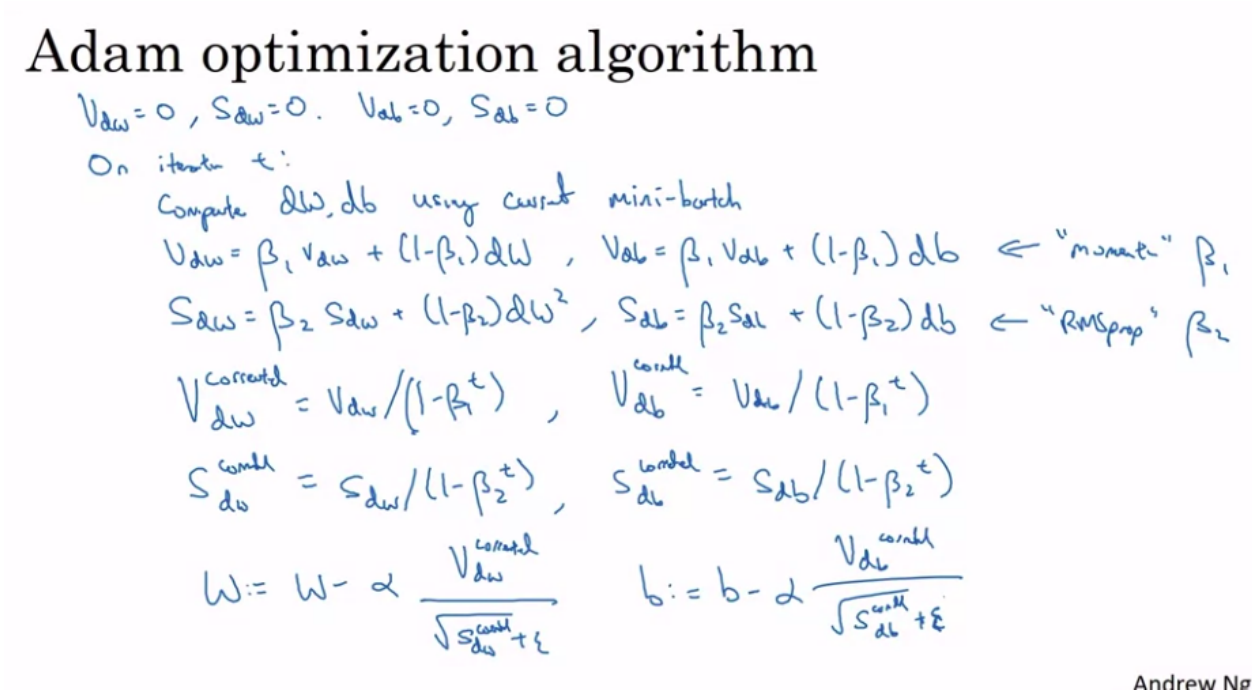
- Mini batch is faster than batch GD.
- Dividing the whole dataset into small batches and update weights for every batch.
- Choosing the size of mini-batch:



W2L4: RMSprop

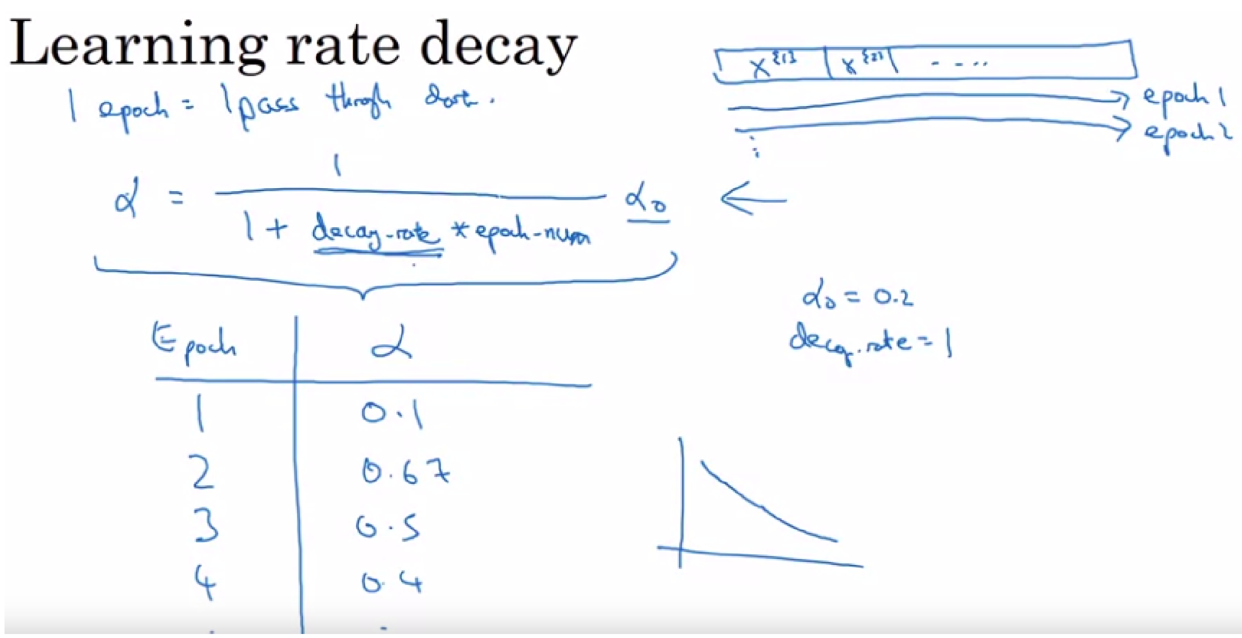


W2L5: Adam
Combination of exponential weighted avg and RMSprop



W2L6: learning rate decay

method1:

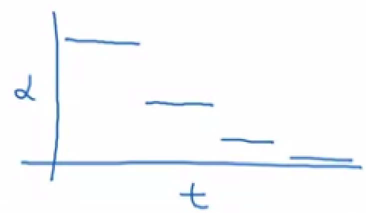


Other learning rate decay methods

formula

$$\alpha = 0.95^{\text{epoch-num}} \cdot \alpha_0$$ – exponentially decay.

$$\alpha = \frac{k}{\sqrt{\text{epoch-num}}} \cdot \alpha_0 \quad \text{or} \quad \frac{k}{\sqrt{t}} \cdot \alpha_0$$



discrete staircase

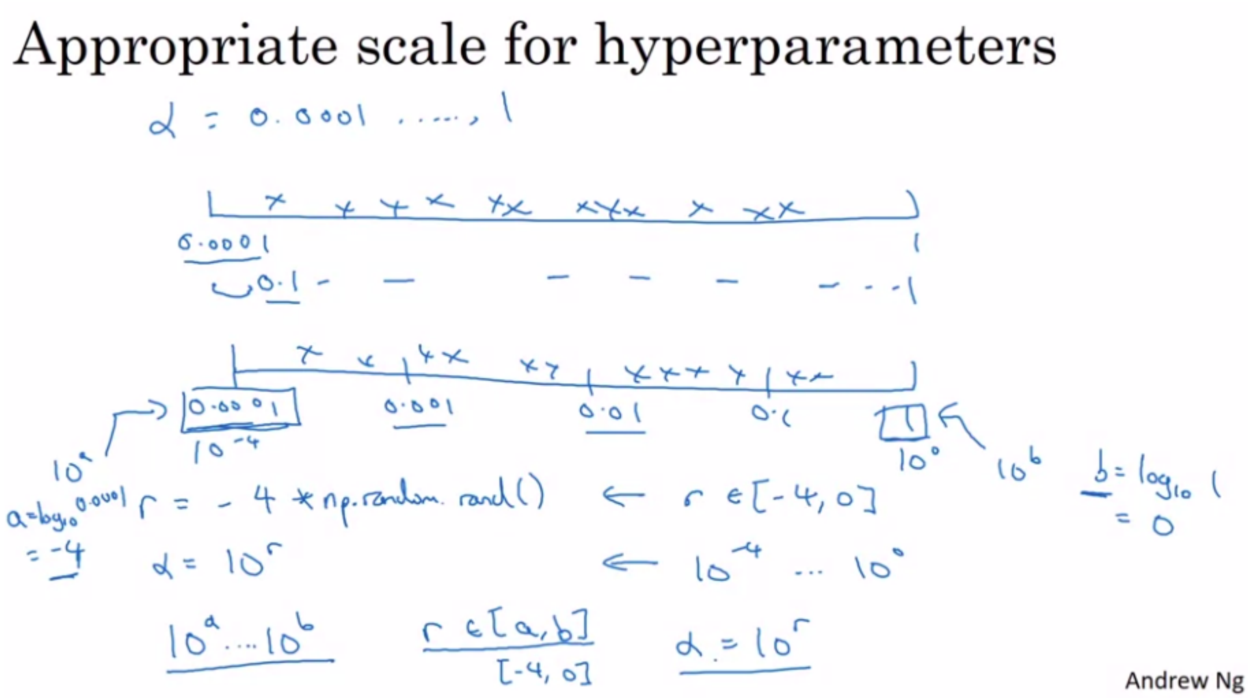
Manual decay.

W3L1: tuning process

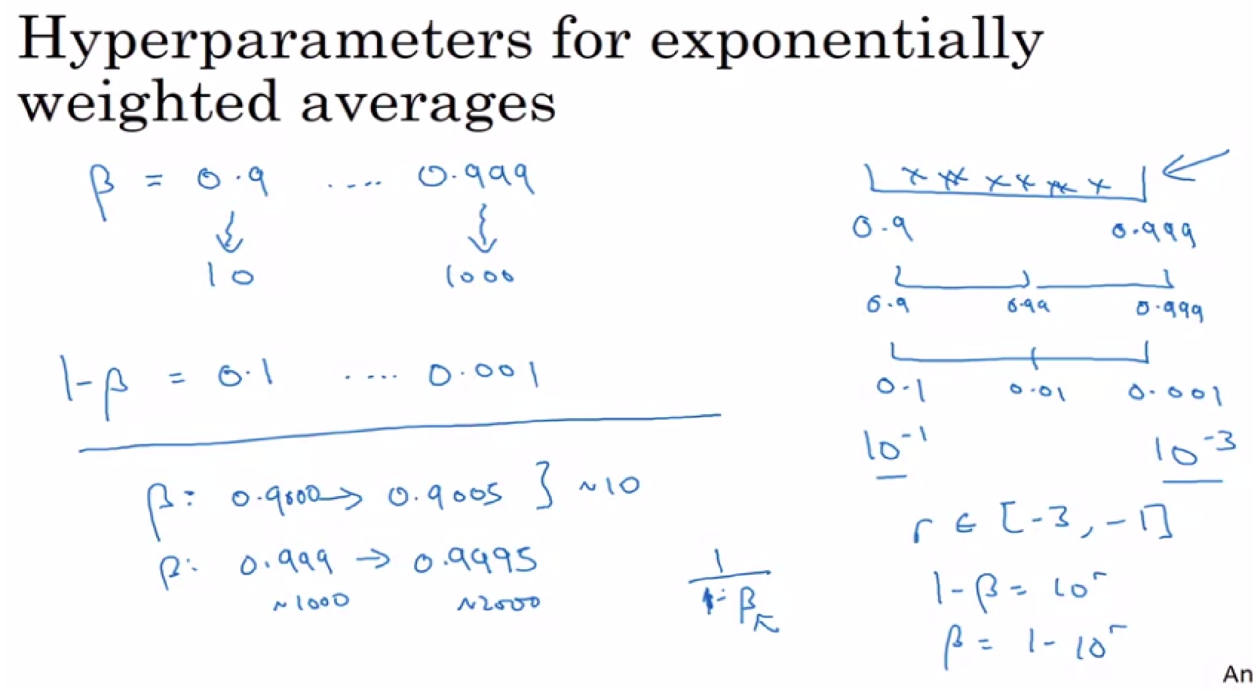
- Order of importance of tuning the hyperparameters:
  learning rate
  momentum
  # hidden units
  mini-batch size
  #layers
  learning rate decay
  adam optimizer parameters: beta1, beta2, epsilon
- Do not use a grid, try random samples

W3L2: picking the right scale for tuning: use logarithmic scale

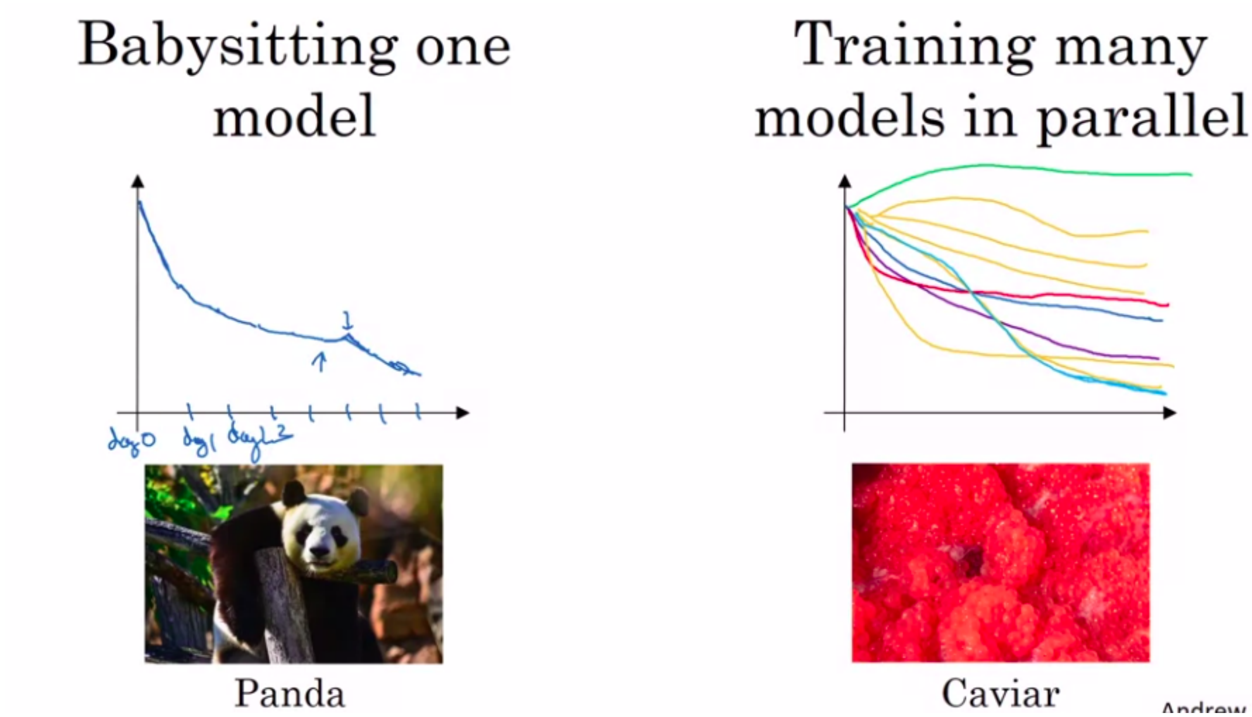
- Learning rate:



- Beta for exponentially weighted averages

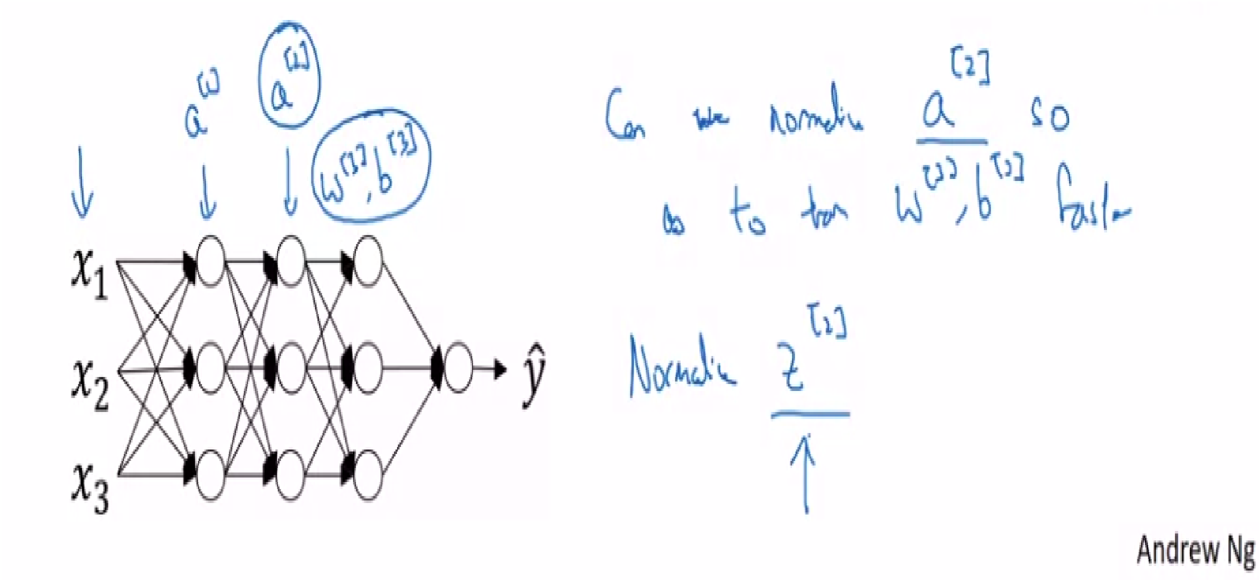


W3L3: tuning in practice: Application dependent

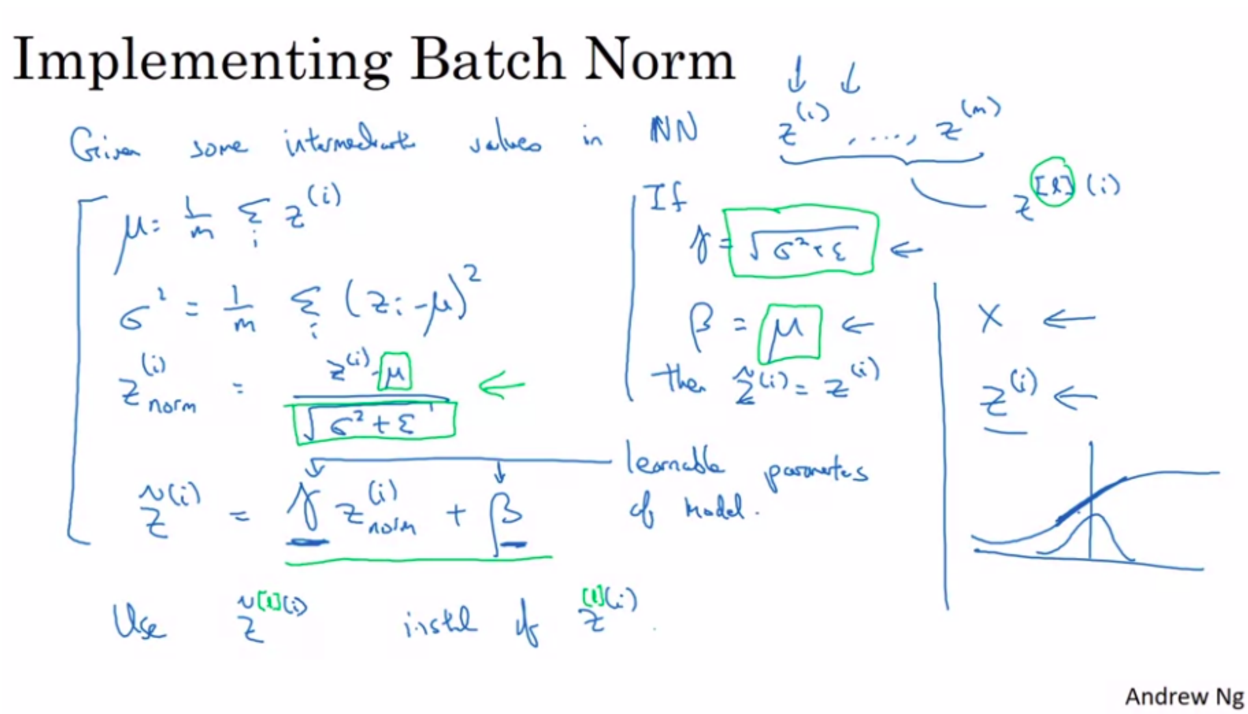


W3L4: Batch Normalization

- Normalizing the values of the parameters before applying the activation function.



- Implementing Batch Norm:



Working with mini-batches

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

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

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

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

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

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

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

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

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

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

Andrew Ng

Implementing gradient descent

for $t = 1 \dots$ numMiniBatches

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

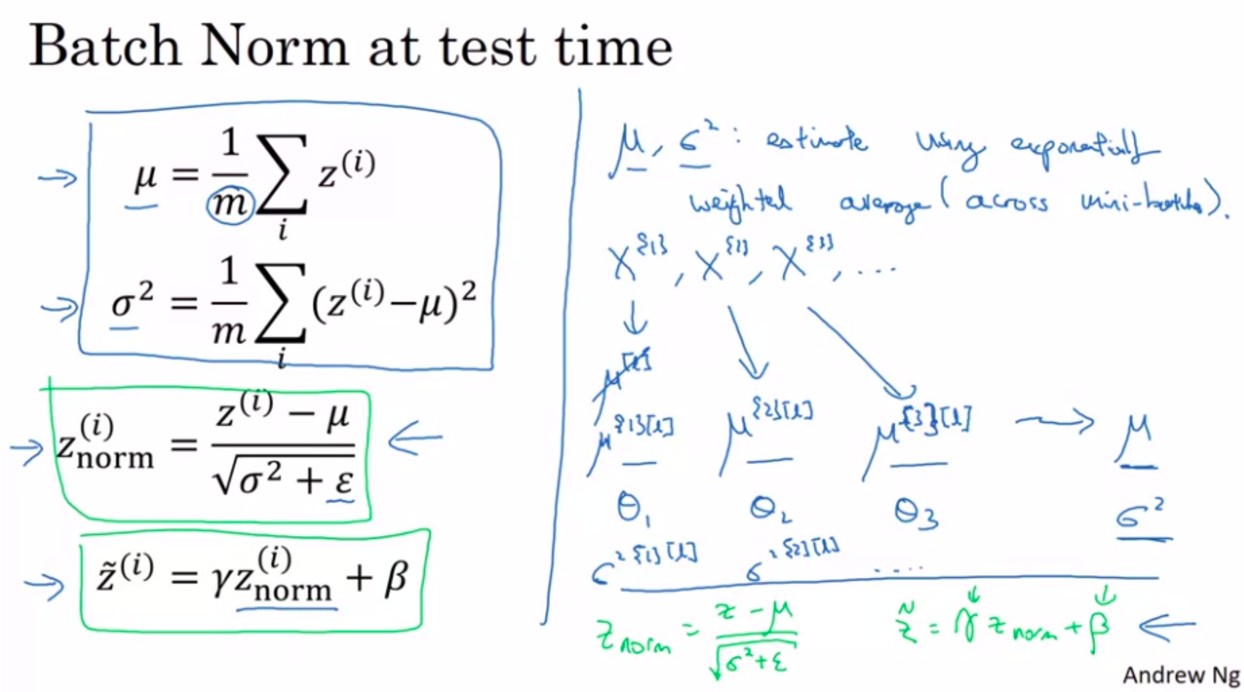
In each hidden layer, use BN to replace $z^{[l]}$ with $\tilde{z}^{[l]}$.

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

Update params
$$\left.\begin{aligned} W^{[l]} &:= W^{[l]} - \alpha\, dW^{[l]} \\ \beta^{[l]} &:= \beta^{[l]} - \alpha\, d\beta^{[l]} \\ \gamma^{[l]} &:= \dots \end{aligned}\right\}$$

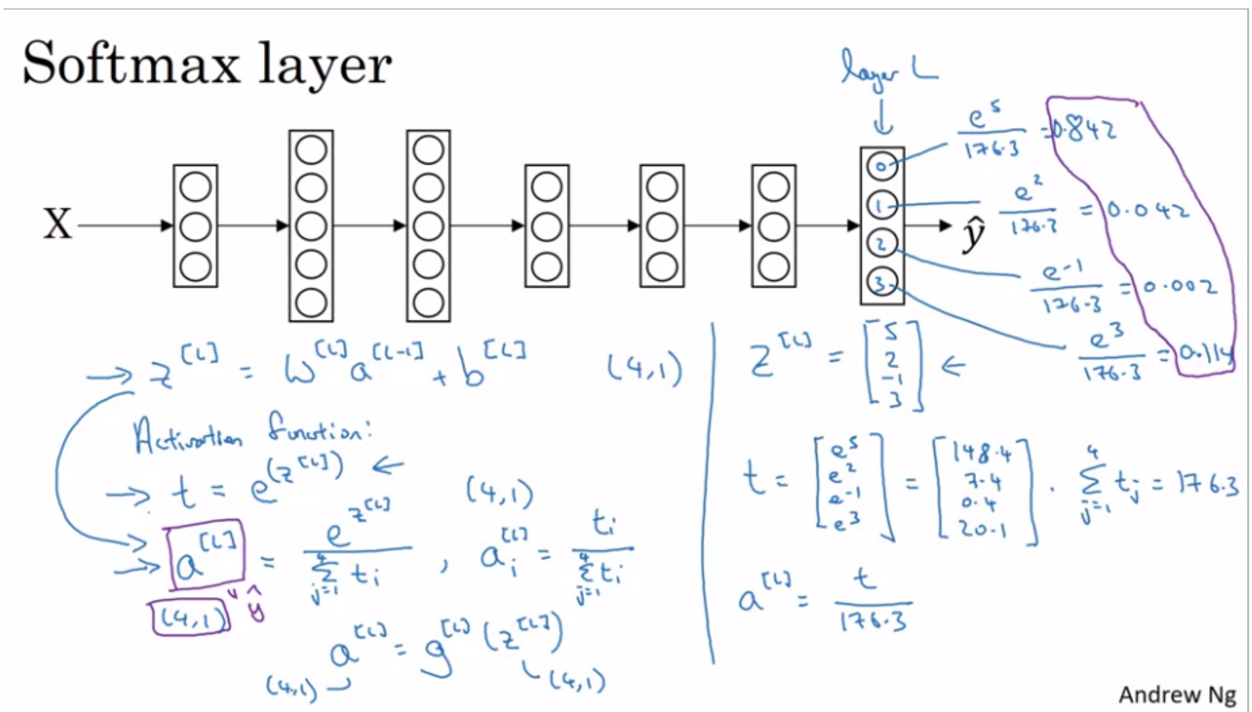
Works w/ momentum, RMSprop, Adam.

W3L5: BN at the test time



W3L6: Multiclass classification:

- Softmax Layer: takes a vector as an input and outputs a vector with the same dimensions. It computes the chance of occuring a particular class in the multiclass classification.
- In the example below: there is 84% chance of the predicted class to be class 0.



W3L7: training a softmax classifier