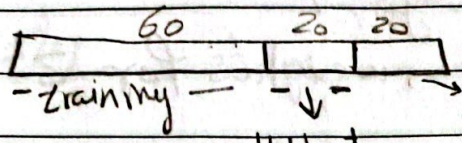
(Corvse 2) week

PAGE
DATE

* improving DNN, (Hyperparameters, tuning, reg. & optimization) (week 1)

→ Train/Dev/test sets

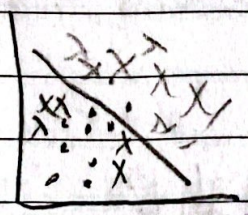
• when training NN, we have alot of decisions ex: num of layers
• num of hidden units • learning rate • act. func.



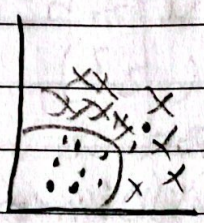
- Holdout Cross validation
- Development set "Dev"

• first training algorithms on training set then use dev to see which of many diff models performs best on your dev set & then after having final model we evaluate the test set

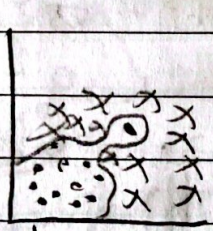
→ Bias / Variance



• high bias
• underfitting



• Just right



• high variance
• overfitting.

• train set: 1%	15%	15% → high bias
• Dev set: 11%	16%	30% → high variance
high variance	high bias	

↳ when algorithm doesn't do well on the training set.

0.5% → low bias
1% → low variance
} → we search for this

PAGE
DATE

Basic recipe for ML

- High bias? (Training data performance) —try→ • Bigger Network • train longer or • NN architecture search
- after fitting
- High variance? (to evaluate that I look at (dev set performance)) —→ • more data (to train) • try regularization • NN arch. search.

That I have good train & dev sets

- Bias variance trade off : reduce or inc. bias or variance without affecting the other one.

→ regularization.

for ex: in logistic regression we try to minimize the cost fun J.

$$J(w,b) = \frac{1}{m} \sum \mathcal{L}(\hat{y}^i, y^i) + \frac{\lambda}{2m} \|w\|_2^2$$

↳ regularization parameter

$$/2 \quad \|w\|^2 = \sum w_j^2 = w^T w$$

$$/1 \quad \frac{\lambda}{2m} \sum |w_j| = \frac{\lambda}{2m} |w|_1$$

→ Drop out regulization:

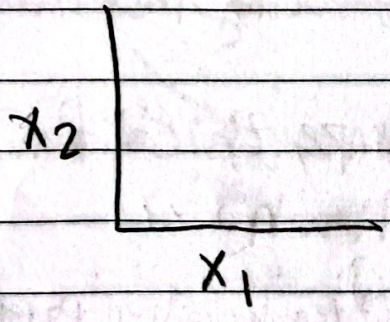
→ Other regulization techs.
- early stopping & Data augmentation

→ Normalizing inputs.
- normalizing the inputs is one of the techniques that is used to speed up the training
- normalizing inputs corresponds to two steps
  ① to subtract out ② to zero out the mean

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

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

$x = x - \mu$



② Normalize the variance

$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} **2 \rightarrow x /= \sigma$

- when features are on similar scales not from 1-1000 & 0-1 but mostly from -1..1 or about similar variance that just makes your cost function j easier & faster to optimize

ex: x_1: 0 ... 1
x_2: -1 ... 1
x_3: 1 ... 2
→ similar ranges ✓

x_1: 1 ... 1000 → hurts optimization
x_2: 0 ... 1

PAGE
DATE

→ Vanishing / exploding gradients

. is one of the problems of training neural network & it is when you're training very deep NN you derivatives or slope can sometimes get either very big or very small & this makes training difficult. & the random weight initialization to significantly reduce this problem.

→ weight initialization for deep networks

. $z = w_1 x_1 + \cdots + w_n x_n + b$

The largen $n \rightarrow$ smaller w_i

. The problem of vanishing or exploding gradients as well as choosing resonable scaling for how you initialize the weights That makes weights doesn't explode quickly or not decay to zero quickly

→ numerical approximation of gradients

. in order to make sure you've got all the details right & internal backprop ✗

→ gradient checking.

. is a technique That helps to find bugs in implementations of back propagations many times