Training a neural network

1) Randomly initialize weights. (≈ 0)
2) Implement forward propagation to get $h_\Theta(x^{(i)})$ for any $x^{(i)}$
3) Implement code to compute cost function $J(\Theta)$
4) Implement backdrop to compute partial derivatives $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$

for $i = 1 : m$

perform forward prop & back prop using $(x^{(i)}, y^{(i)})$

(get activations $a^{(l)}$ & $\delta^{(l)}$ for $l = 2, \ldots, L$)

5) Use gradient checky to compute $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$ computed using back propagation vs using numerical estimate of gradient of $J(\Theta)$. Then disable gradient checking code.

6) Use 'gradient descent or advanced optimization method with back propagation to try to minimize $J(\Theta)$ as a function of parameters Θ.

How to debug a learning algorithm

- Get more training example - fixes high variance.
- Try smaller set of features - fixes high variance
- Try getting additional features - fixes high bias
- Try adding polynomial features - fixes high bias
  (fixes high bias) ($x_1^2, x_2^2, x_1 x_2$, etc)
- Try decreasing λ (Regularized expression)
- Try increasing λ. (fixes high variance)

Machine learning diagnostic

: A test you can run to gain insight about the learning algorithms

→ 70% train

→ 30% test

To do,

→ learn parameter θ from training data (minimizing training error) J(θ).

→ compute test set error. (linear regression)

$$J_{test}(\theta) = \frac{1}{2m} \sum_{i=1}^{m_{test}} \left[h_\theta(x_{test}^{(i)}) - y_{test}^{(i)} \right]^2$$

Model selection

60% train set

20% cross validation set

20% test set.

* when we have a lot of features, we use regularization ~~for overfit~~

high bias → λ is large ∴ all other features of hypothesis are ≈ 0
∴ $h_\theta(x) \approx \theta_0$

high variance → $\lambda \approx 0$.

Learning curves

high bias won't help if you increase no. of examples

high variance, get more data

Building a spam classifier,

x = features of email.
y = spam (1) or not spam (0).

x = look for words that are used in spam i.e, deal, buy, now, discount. (or Andrew if we think it is not a spam).

How to go about it?

- Collect data
- Develop algorithm based on the email routing (contained in the header)
- Develop sophisticated algo - i.e., 'discount' & 'discounted' are same or not, 'deal' & 'dealer' are same or not.
- Develop sophisticated algorithm to detect misspellings (w4tches)

Approach

- Start simple
- plot learning curve & decide whether more data or more features.
- Error analysis - Manually examine the examples

Numerical evaluation - Single number for checking the error.

skewed class :- If you have only examples which point to one class i.e, patient doesn't have cancer.

		Actual class 1	Actual class 0
Predicted class	1	True positive	False positive
	0	False negative	True Negative

$\frac{85}{85+890}$

Precision - of all patients predicted, who actually have cancer

$$\text{precision} = \frac{\text{True pos.}}{\text{True pos + false pos.}}$$

Recall - of all patients that actually have cancer, how much did we predict

$$\text{recall} = \frac{\text{True pos}}{\text{True pos + false neg.}}$$

for precision recall ~~weight~~ trade off

$$f_1 \text{ score} = \frac{2PR}{P+R}$$