## 1 Rules of thumb

| Rule of Thumb | source | notes |
|---|---|---|
| If you have more than 1M datapoints - use deep learning dealing with images / audio / video), then you're probably better off learning deep learning. | link | |
| “I've seen over and over that one of the most reliable ways to get a high performance machine learning system is to take a low bias learning algorithm and to train it on a massive training set.” | Andrew Ng | |
| “if you don't have a low bias algorithm, try to add more features” | Andrew Ng | |
| ”the process of artificial data synthesis it is .. a little bit of an art as well and sometimes you just have to try it and see if it works.” | Andrew Ng | |
| ”try to calculate how much time it would take to collect and label data yourself” | Andrew Ng | |
| “It's not who has the best algorithm that wins. It's who has the most data ” | | |

## 2 Resources

| Topic | link | notes |
|---|---|---|
| Data science project ideas | link | |

## 3 Week 6 - Machine Learning Diagnostics

### 3.1 Learning Curves - figuring out if you have high bias or high variance

Plot error versus m (training set size).

High bias looks like this:

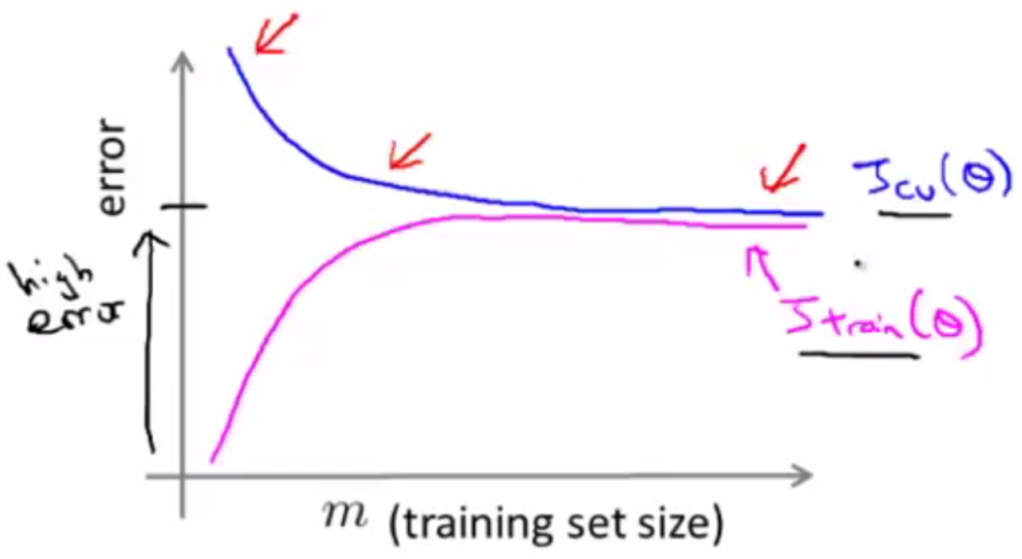
Getting more data won't help.

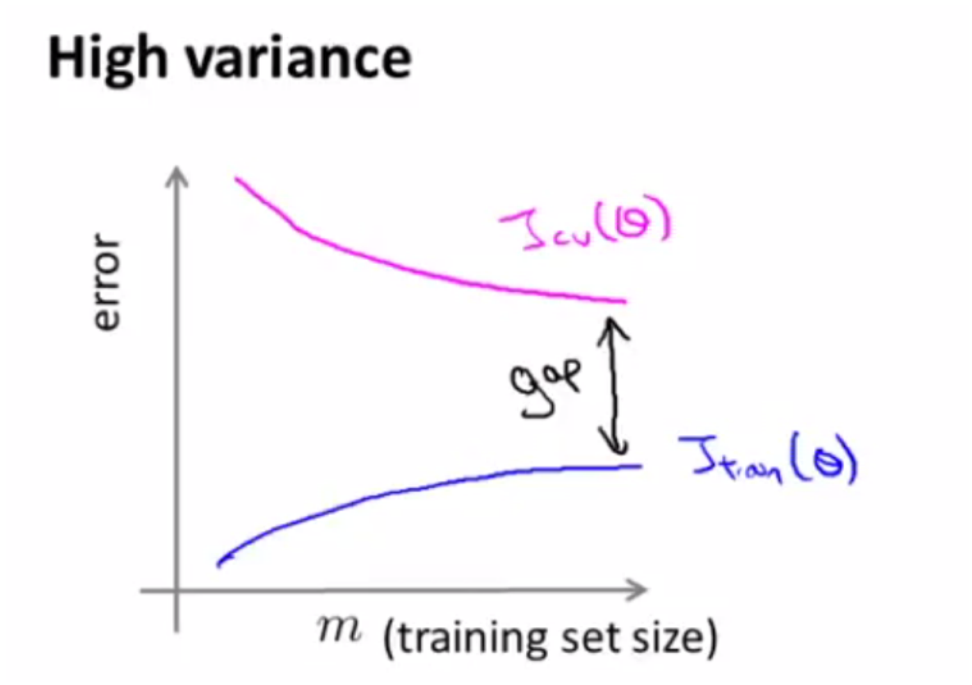
High variance looks like this:

Getting more data is likely to help.

To plot a learning curve, $J_{\text{train}}(\theta)$ average squared error on my training set or Jcv which is the average squared error on my cross validation set. And I'm going to plot that as a function of m, that is as a function of the number of training examples I have. And so m is usually a constant like maybe I just have, you know, a 100 training examples but what I'm going to do is artificially with use my training set exercise. So, I deliberately limit myself to using only, say, 10 or 20 or 30 or 40 training examples and plot what the training error is and what the cross validation is for this smallest training set exercises.

## High bias





## 3.2 Debugging a learning algorithm

- Get more training examples - fixes high variance
- try smaller sets of features - fixes high variance
- Try getting additional features - fixes high bias
- Try adding polynomial features - fixes high bias
- try decreasing $\lambda$ - fixes high bias
- try increasing $\lambda$ - fixes high variance

Neural networks: Often the larger the better (i.e. more hidden units the better), and use regularization to address overfitting.

How to choose how many hidden layers:

use one, hidden layer (often this is enough) measure error,

use 2 measure error etc.

$\vdots$

## 3.3 Skewed Classes

We have a lot more examples from one class compared to another.

If you have skewed classes, don't simply use classification accuracy as an evaluation metric. Instead use Precision/Recall.

**Precision** of all the patients where we predicted y=1, what fraction actually has cancer?

$$\frac{\text{True Positives}}{\text{\# predicted postive}} = \frac{\text{True Positives}}{\text{True Positives} + \text{False pos}}$$

**Recall** Of all patients that actually have cancer, what fraction did we correctly detect as having cancer?

$$\frac{\text{True Positives}}{\text{\#actual postives}} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Neg}}$$

y=1 denotes presence of rare class, e.g. patient has cancer.

High precision and high recall = good classifier

**F score** How to use precision and recall to come up with a single real number evaluation metric

Could use average, but not really a good idea, you'll get a number which is biased towards extreme values (e.g recall = 1, precision = 0.02).

$$F_1 = 2\frac{PR}{P+R}$$

Both P & R have to be high for $F_1$ to be high There are a lot of ways to combine precision and recall, but this is a very common one.

## 3.4 very large datasets

Assumption: If features have enough information to accurately predict $y$

If a human expert had the information contained in the features, would they have enough information to predict y? If yes, then the above assumption is true.

Then use a low bias algorithm, like a neural network with lots of hidden nodes or linear regression with lots of features.

If assumption 1 holds and you have a low bias algorithm, then getting lots of data is likely to really help your dataset.

# 4 Week 7: Unsupervised Learning

# 5 Week 9:Anomaly Detection

Anomaly Detection Example: $p(x_{test}) < \epsilon$ flag anomaly $p(x_{test}) \geq \epsilon$ flag anomaly

Most frequent use:

- Fraud Detection: $x^{(i)}$ = features of user $i$'s activities Model $p(x)$ from data Identify usual users by checking which have $p(x) < \epsilon$
- Manufacturing
- Monitoring computers in a data center $x^{(i)}$ features of machine $i$ $x_1$ memory use,$x_2$ number of disk accesses .

## 5.1 Normal distribution

## 5.2 Parameter Estimation

For Normal Distribution

(These are Maximum likelihood estimates - they are just averages)

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

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

## 5.3 Density estimation

training set: $\{x^{(i)}, \ldots, x^{(m)}\}$ each example is $x \in R^n$

$$\begin{aligned} p(x) &= p(x_1; \mu_1, \sigma_1^2)p(x_2, \mu_2, \sigma_2^2)\ldots p(x_n, \mu_n, \sigma_n^2) \\ &= \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) \end{aligned}$$

The above corresponds to assumption that:

$$\begin{aligned} x_1 &\backsim \mathcal{N}(\mu_1, \sigma_1^2) \\ x_2 &\backsim \mathcal{N}(\mu_2, \sigma_2^2) \\ &\vdots \end{aligned}$$

which corresponds to an independence assumption on the input features.

### 5.3.1 Anomaly detection algorithm

1. Choose features $x_i$ which you think might be indicative of anomalous examples, where each $x_i \in R^n$
2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2,$, i.e. compute:

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

   where $\mu_j$ is the average value of the $j$th feature.

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2$$

3. Given new example $x$, compute $p(x)$:

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

   anomaly if $p(x) < \epsilon$

# 6 Notes Coursera Week 10: Large Scale Machine Learning

## 6.1 Vid1: Learning with Large Datasets

motivating example: m = 100,000,000

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)} - y^{(i)}))x_j^{(i)}$$

That's 100,000,000 summations for 1 step of gradient descent.

Idea: train on 1000 samples, maybe it will do just as well.

Quiz question: Suppose you have m=100,000, 000; how can you tell if using all of the data is likely to perform much better than using a small subset of the data (m= 1000).

Answer: Plot a learning curve for a range of values of m and verify that the algorithm has high variance when m is small.

## 6.2 Vid2: Computationally efficient methods: Stochastic Gradient Descent

Gradient descent not efficient for very large datasets. It's called 'Batch gradient descent'

| $J_{train}(\theta)$ | stuff |
|---|---|

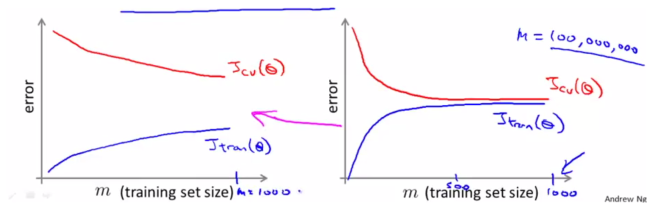


Figure 1: Left: High variance, should add more data, Right: high bias, more data won't help, should add more features, or more hidden units to neural networks

### 6.3 Vid3: Mini-Batch Gradient Descent

### 6.4 Vid4: Stochastic Gradient Descent Convergence

### 6.5 Vid5: Online Learning

### 6.6 Vid6: Map reduce and Data Parallelism

## 7 Week 11: Where to allocate scarce resources?

Video title: Ceiling analysis: what part of the pipeline to work on next

Suppose you have pipeline of tasks in your machine learning project. And your current overall accuracy is 72%. To figure out where to allocate resources to improve the system, break the pipeline and feed in perfectly correct information (by manually correcting the label outputs of the previous component for example) to one component, and recalculate the performance improvement of the entire pipeline. Then do the same with the next component, etc. etc. This will show you the 'upside potential' for each component. Allocate the resources to the component with the biggest upside potential.