ML COURSE

WEEK 1 - Introduction

Tom Mitchell's definition of Machine Learning:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if in its performance at tasks in T, as measured by P, improves with experience E".

SUPERVISED LEARNING

Regression	Classification
Trying to predict results within a continuous output	TRYING TO PREDICT RESULTS IN A DISCRETE OUTPUT
↓	↓
Trying to map input variables to some continuous function	Map input variables into discrete categories.
Example: house prices, predicting age.	Example: probability of tumor (benign or malignant)

You can consider an infinite number of features through something called the Support Vector Machine.

UNSUPERVISED LEARNING

Allows us to approach problems with little or no idea what our results should look like

Clustering	Non-clustering
Find a way to group elements (example: genes) into groups that are similar or somehow related (lifespan, location etc) (Google News!)	"Cocktail party algorithm" Allows you to find structure in a chaotic environment. (Example: identifying individual voices or sounds at a cocktail party from diff microphones)

WEEK 1 – MODEL AND COST FUNCTION

Cost function → $J(\theta_0, \theta_1) = \ldots$

WEEK 2

Linear Regression with multiple features

n = number of features

m = number of training examples

$x_j^{(i)}$ = value of feature j in ith training example

$x^{(i)}$ = vector of all values of all features of ith tr. example.

$h_\theta(x)$ becomes $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

$(x_0^{(i)} = 1)$

For convenience of notation, define $x_0 = 1$ (so they have same number of elements)

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$h_\theta(x) = \underbrace{\theta_0 x_0}_{1} + \theta_1 x_1 + \dots + \theta_n x_n =$$

$$= \theta^T x$$

Called multivariate linear regression

Cost Function $J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

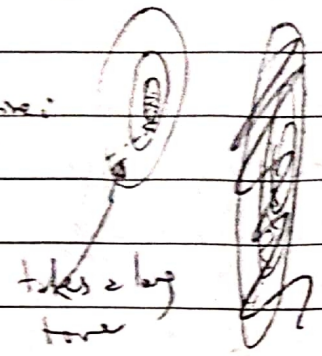
Gradient descent: $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n)$

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

Feature scaling:

if you make sure the values (numerical) of all features are similar, gradient descent will be much quicker.

image:



We can multiply/divide the features by whatever number we want.

Get every feature approximately in a $-1 \leq x \leq 1$ range.

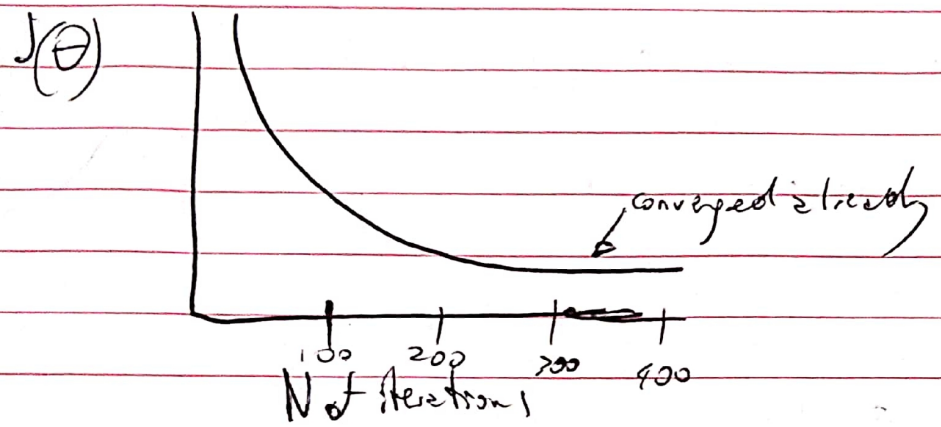
Sometimes people will also do:

Mean Normalisation:

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean ($x_0 \neq 1$)

WEEK 2: Making sure gradient descent works properly

$J(\theta)$ should decrease after every iteration



Example automatic convergence test:

- Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration

this is okay, but it's better to always be looking at the graph

IF the gradient descent is not working, try a smaller value of α.

It has been shown mathematically if α is small enough it will decrease on each step.

WEEK 2: Polynomial regression.

Suppose we have two features frontage & depth of a house.

Polynomials

More features, need feature scaling becomes more important.

$h_\theta(x) = x + x^2 + x^3 \ldots$

↓ if x ranges from 1 to 10, we'll be considering ranges of 1 to 10, 100, 1000 etc.

WK 2: Normal equation

If 1D: $J(\theta) = a\theta^2 + b\theta + c$

$\frac{d}{d\theta} J(\theta) = 0 \Longrightarrow$ Solve for θ

in the case of a real number.

for our case, it's an $n+1$ dimensional vector.

$$\frac{\partial}{\partial \theta_j} J(\theta) = \ldots \overset{\text{set}}{=} 0.$$

If you solve this, you found the θ! ($\theta_0, \theta_1, \theta_2, \ldots, \theta_n$)

Example:

x_0	Size	# of bedrooms	# floors	age	Price
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

(design matrix) $X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$ $\begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} = y$ $\rightarrow \theta = (X^T X)^{-1} X^T y$

Why do we even bother with gradient descent when we can just use normal equations?

Gradient descent	Normal equation
n iterations	nxn matrix
	inverting a matrix takes $O(n^3)$ time

if you have $n=100$, no problem, use neq.
if you have $n=1000$, still meh use neq.
if $n=10000$, maybe but not really
for $n=1000000$, definitely use gradient descent.

WEEK 2: Normal equation noninvertibility. $\theta = (X^T X)^{-1} X^T y$

What if $X^T X$ is non-invertible?

Octave: pinv$(X' \cdot X) \cdot X' \cdot y$

as long as you use pinv(pseudo inverse), it's all good.

- Redundant features (linearly dependent): try deleting one
- Too many features:
  1. Delete
  2. regularization