WEEK #1

1) What is ML.
2) Introductn
3) Quiz #1
4) Other materials
5) Model and Cost Fxn
6) Paramtr Learning
7) Review Quiz
8) Linear Algebra review
9) Review Quiz

WEEK #2

1) Environment Setup Instr.
2) Multivariate Linear Regr.
3) Computng Paramtrs Analtc.
4) Submitting Progr. Assignts.
5) Review Quiz
6) Octave/MATLAB Tutorial
7) Review Quiz

Programming Assignment

WEEK #3

1) Classificatn & Representn.
2) Logistic Regression Model
3) Multiclass Classificatn
4) Review Quiz
5) Solving the Prob. Overfitting
6) Review Quiz

Programming Assignment.

WEEK #4

1) Motivatns
2) NN
3) Applicatns
4) Review (Quiz+PA)

WEEK #5

1) Cost fxn & Backpropagatn
2) Backpropagatn in practice
3) Applicatn of NN
4) Review (QUIZ + P.A.)

WEEK #6

1) Evaluating a Learn. Algor.
2) Bias vs. Variance
3) Review (QUIZ + P.A.)
4) Building a Spam Classifier
5) Handling Skewed Data
6) Using Large Data Sets
7) QUIZ

WEEK #7

1) Large Margin Class'n.
2) Kernels
3) SVM's in Practice
4) Review (QUIZ+ P.A.)

WEEK #8

1) Clustering
2) Review (QUIZ)
3) Motivatn
4) PCA Analysis
5) Applying PCA
6) Review (QUIZ+ P.A.)

WEEK #9

1) Density Estimatn
2) Building a Anomaly Det.S.
3) Multivariate Gaus. Dist.
4) Review (QUIZ)
5) Predicting Movie Ratings
6) Collaborate Filtering
7) Low Rank Mtx Factorztn
8) Review (QUIZ+ P.A.)

WEEK #10

1) Grad. Desc. with Large Datasets
2) Advanced Topics
3) Review (QUIZ)

WEEK #11

1) Photo OCR
2) Review (QUIZ)
3) Conclusion.

WEEK #1

1. What is Machine Learning

- Science of getting computers to learn without being explicitly programmed.
- NN mimics the human brain.

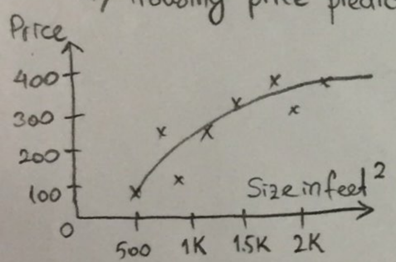
2. INTRODUCTION

2.1. Welcome

- Web search engine: Google, Bing
- Facebook, Apple face recognitn
- Spam filters
- Autonomous robots
- Computatnl biology
- Handwriting recognitn.
- NLP: Natural Language Processing
- Computer Vision
- Self-customizing programs
- In ∀ fields of engineering there is a huge amount of data sets, that we are trying to understand using learning algorithms.

2.2. Supervised Learning.

ex/ Housing price predictn.



ex/ Breast cancer (malignant, benign)

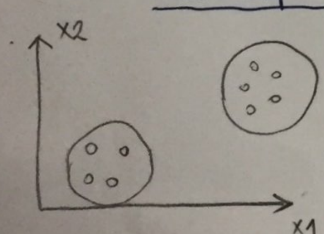
⇒ Supervised Learning: "right answers" given

⇒ Regression: Predict cont^s valued outputs

⇒ Classificatn: Descrete valued outputs (0 or 1) / (0, 1, 2, 3)

⇒ # of features may vary by problem

2.3. Unsupervised Learning:



ex/ Google News, Social network analysis, Astronomical data analysis, Market segmentatn.

⇒ Unsupervised Learning: we are given a data that does not have any labels / that ∀ has the same label.

Cocktail Party Problem Algorithm:

```
[W,s,v] = svd((repmat(sum(x.*x,1), size(x,1),1).*x)*x');
```

3. QUIZ #1 (100% ✓)

4. OTHER MATERIALS:

The Course Wiki

5. MODEL & COST FUNCTION

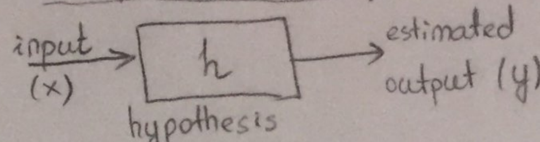
5.1. Model Representatn:

m = # of training examples

x's = "input" variable/features

y's = "output"/"target" variable.

h maps from x's to y's:



$h_\theta(x) = \theta_0 + \theta_1 x$ (univariant linear regression)

5.2. Cost Functn:

θ_0, θ_1: parameters.

→ Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples

$$\boxed{\underset{\theta_0,\theta_1}{\text{minimize}} \sum_{i=1}^{M} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 \cdot \frac{1}{2m} = J(\theta_0,\theta_1)}$$

Cost func.

(squared error func.n)

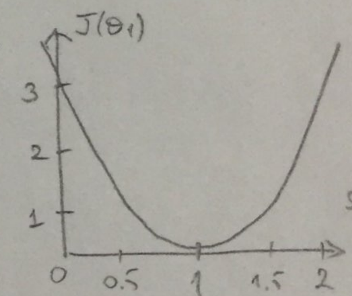
5.3. Cost Func'n - Intuitn 1:

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Paramtrs: θ_0, θ_1

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

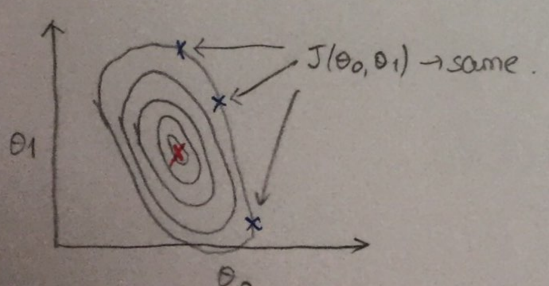
Goal: $\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0,\theta_1)$



⇒ Each value of θ_1 corresponds to a different hypothesis and we can get different $J(\theta_1)$

for $x^{(i)} = y^{(i)} \rightarrow J(\theta_1) = 0$

5.4 Cost Func:n - Intuitn 2:



6. PARAMETER LEARNING

6.1. Gradient Descend:

- for minimizing some arbitrary fxn J. ✓ Have some fxn $J(\theta_0,\theta_1)$ ✓ Want $\min_{\theta_0,\theta_1} J(\theta_0,\theta_1)$

Scanned by CamScanner

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum.

⇒ If you are standing at the pnt on the hill, look around and find that the best drn is to take a little step downhill is roughly that drn.

Gradient Descend Algorithm:

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j=0 \ \& \ j=1)$$

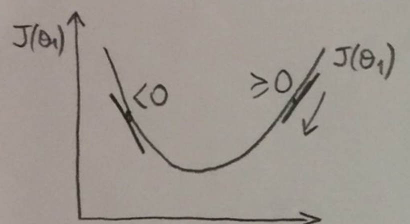
}

α: learning rate (step size)

⇒ simultaneous update:

$$\boxed{\begin{aligned} temp0 &:= \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\ temp1 &:= \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\ \theta_0 &:= temp0 \\ \theta_1 &:= temp1 \end{aligned}}$$

6.2. Gradient descend Intuitn:



if derive ≥ 0

$\theta_1 := \theta_1 - \alpha$ (pos've #) ⟶ decrease θ_1

if derive < 0

$\theta_1 := \theta_1 - \alpha$ (neg've #) ⟶ increase θ_1

⇒ if α is too small, gradient descend can be slow

⇒ if α is too large, gradient descend can overshoot the minimum. It may fail to converge or even diverge.

⇒ Gradient descend can converge to a local minimum, even with the learning rate α fixed

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

As we approach a local minimum, gradient descend will automatically take smaller steps. So, no need to decrease α over time.

6.3. Gradient descend for Linear Regression:

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 = \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \quad \underline{j=0}$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)} \quad \underline{j=1}$$

"Batch" Gradient descend: Each step of gradient descend uses ∀ the training examples.

⇒ Normal eq'n method can solve this minimizatn problem numerically, but Gradient descend can scale better to larger data sets than that normal eqn method.