Contents

1 Introduction **3**

2 Linear Regression With One Variable **6**

3 Linear Regression With Multiple Variable **11**

4 Logistic Regression **18**

5 Regularization **24**


Machine Learning
(Andrew NG)
classmate
Enduring Value
ALL THAT
JAZZ

CHAPTER 1

Introduction

1 Introduction

1.1) Welcome

→ Machine learning grew out of work in AI.

1.2) What is machine Learning

→ Field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel (1959)

Machine learning algorithms

Supervised learning

Unsupervised Learning

1.3) Supervised learning

→ (We gave the algorithm the data set with right answers)

↓

(and the task of the algorithm was to predict the right answer for the next question)

⇒ Regression :: Predict continuous value output.

⇒ Classification :: Discrete valued output (0 or 1)

(Support Vector machine)

→ Can have more than two values.

1.4) Unsupervised - Learning

Algorithm ~~the~~ that learns from test data that has not been labeled, classified or categorized

Clustering

Give the algorithm a ton of data ask to find structure in it.

CHAPTER 2

Linear Regression With One Variable

Linear regression with one variable

2.1) Model representation

⇒ Data set is called training set.

Notation:

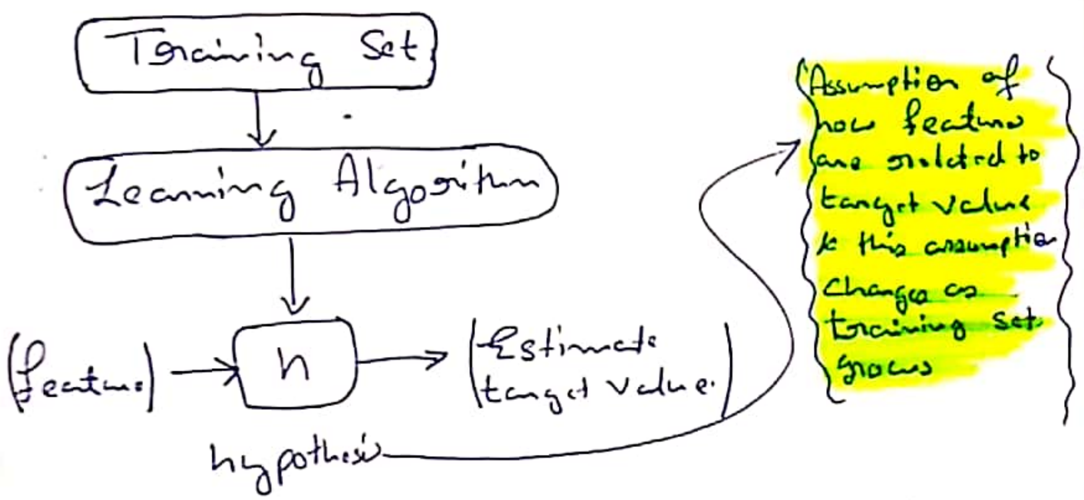
m ⇒ Number of training example

x's ⇒ "input" Variables / features

y's ⇒ "Output" Variable / "target" Variable

(x, y) ⇒ To denote single training example.

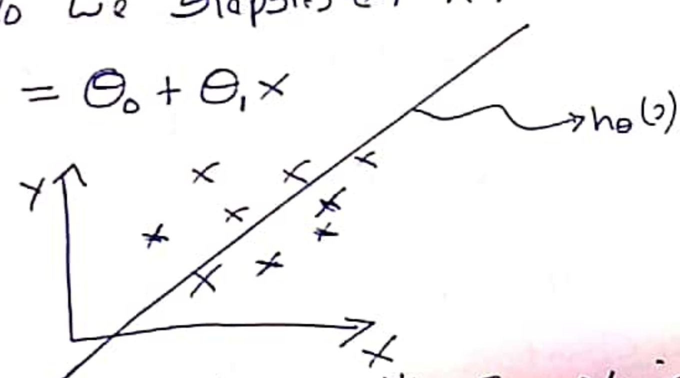
$(x^{(i)}, y^{(i)})$ ⇒ To denote i^{th} training example.



⇒ h is a function which maps x's to y's

⇒ How do we represent h?

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



⇒ Linear regression with one variable.
(i.e. Univariate linear regression)

2.2> Cost function

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

{ θ_i's ⇒ Parameters }

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

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

Cost function

{ Squared error cost function }

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

{ Minimize $J(\theta_0, \theta_1)$ to obtain value of θ_0 & θ_1 }

Contour line ⇒ A contour line of a function of two variables is a curve along which the function has a constant value, so that the curve joins points of equal value.

→ different colors are used to represent contour lines at different height.

2.5 Gradient descent

{An algorithm for minimizing the cost function J}

$$\boxed{\min_{\theta_0 \cdots \theta_n} J(\theta_0, \cdots \theta_n)}$$

⇒ First start from a random value,

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta_0, \theta_1) \quad (\forall\ j=0 \text{ and } j=1)$$

}

⇓ {Correct simultaneous update}

$$temp0 := \theta_0 - \alpha \frac{\delta}{\delta \theta_0} J(\theta_0, \theta_1)$$

$$temp1 := \theta_1 - \alpha \frac{\delta}{\delta \theta_1} J(\theta_0 . \theta_1)$$

$$\theta_0 := temp0$$

$$\theta_1 := temp1$$

{Assignment operator}

α ⇒ learning rate (i.e. how big the step is)

{= → {Truth assertion}}

Issue: It can be susceptible to local optima.

⇒ The algorithm is also called ("Batch") Gradient Descent.

Each step of gradient descent uses all the training examples

Two extension

Exact method

Larger number of features

x_1 x_2 - - - x_n can be different feature

CHAPTER 3

Linear Regression With Multiple Variable

Linear regression with Multiple Variables

4.1) Multiple features

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example

$x_j^{(i)}$ = Value of feature j in i^{th} training example

{Will be a number}

{Will be n dimentional vector}

$x^{(i)} \in \mathbb{R}^n$

⇒ New hypothesis function : $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \theta_n x_n$

⇒ For convenience of notation define $x_0^{(i)} = 1$

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

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$h_\theta(x) = \theta^T x \qquad \left\{ x = \begin{bmatrix} x_0 \\ x^{(i)} \end{bmatrix} \right\}$$

⇒ Multivariate linear regression

4.2> Gradient descent for multiple Variables

Cost function

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Vector Vector

Gradient descent

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\delta}{\delta\theta_j} J(\theta) \quad \{\forall\ j = 0, 1 \dots n\}$$

}

$$\Downarrow$$

Repeat {

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

(Simultaneously update θ_j for $j = 0 \dots n$)

}

4) Feature Scaling

If different features takes different range of values then it becomes very difficult for algorithm to find minimum or it takes way more time for algorithm to find minimum.

→ To avoid this we scale every feature so that they are in similar range.

⇒ Get every feature into approximately a $-1 \leq x_i \leq 1$ range.

Mean normalization

⇒ Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean.

General rule $x_1 \longleftrightarrow \dfrac{x_1 - \mu_1}{s_1}$

μ_1 → avg value of x_i in training set

s_1 → range of x_i (i.e. $\max x_i - \min x_i$)

⇒ To make sure gradient descent is working correctly.

→ Plot $J(\theta)$ Vs (Number of iteration)

↓

{If it is continuously decreasing then its going well.}

↓

{If it is almost flat then you can assume that it has converged}

$\Rightarrow$ To choose α, try

... 0.001 , 0.01 , 0.1 , 1 ...

4.5> Polynomial regression

$h_\theta(x) = \theta_0 + \theta_1 x$ { ✗ dosent give good fit }

So, $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

{Very good fit}

So $x_1 = x$
$x_2 = x^2$
$x_3 = x^3$
{ define three new features }

4.6> Normal Equations

Method to solve for θ analytically.

Let $X = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \cdots & x_n^1 \\ 1 & x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1^m & x_2^m & \cdots & x_n^m \end{bmatrix}_{m \times (n+1)}$

m → number of training set

n → number of feature

X → Matrix that contain all the training example

Similarly $Y = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix}_{m \times 1}$ Collection of all the target variable in training example

$$\boxed{\theta = (X^T X)^{-1} X^T y}$$

This gives you the value of θ that minimizes the cost function

Note

$$J(\theta) = \frac{1}{2m} (X\theta - Y)^T (X\theta - Y)$$

Gradient Descent	Normal Equation
⇒ Need to choose α.	⇒ No need to choose α.
⇒ Need many iterations.	⇒ Don't need to iterate.
⇒ Works well even when n is large	⇒ Slow if n is very large. $O(n^3)$
$n > 10000$	$n = 100$ OK $n = 1000$ OK $n = 10000$ ← Not greater than this

CHAPTER 4

Logistic Regression

6

Logistic regression

6.1> Classification

⇒ Logistic Regression is a classification algorithm

$$0 \leq h_\theta(x) < 1$$

6.2> hypothesis - representation

$$h_\theta(x) = g(\theta^T x)$$

$$\text{where } g(z) = \frac{1}{1+e^{-z}}$$

→ Sigmoid function or logistic function

$$\Rightarrow h_\theta(x) = \frac{1}{1+e^{\theta^T x}}$$

→ estimated probability that y = 1 on input x

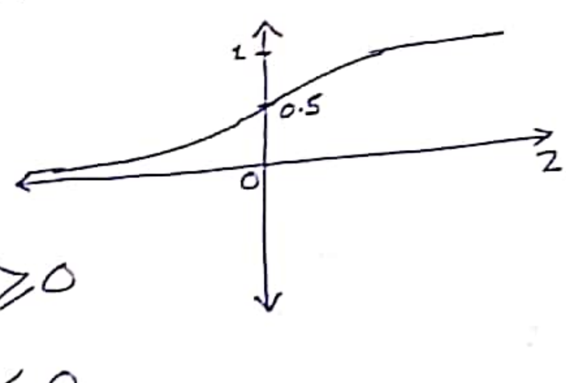
6.3> Decision boundary

Predict "y = 1" if $h_\theta(x) \geqslant 0.5$

& Predict "y = 0" if $h_\theta(x) < 0.5$

$$h_\theta(z) = \frac{1}{1-e^{-z}}$$

when $z = \theta^T x$

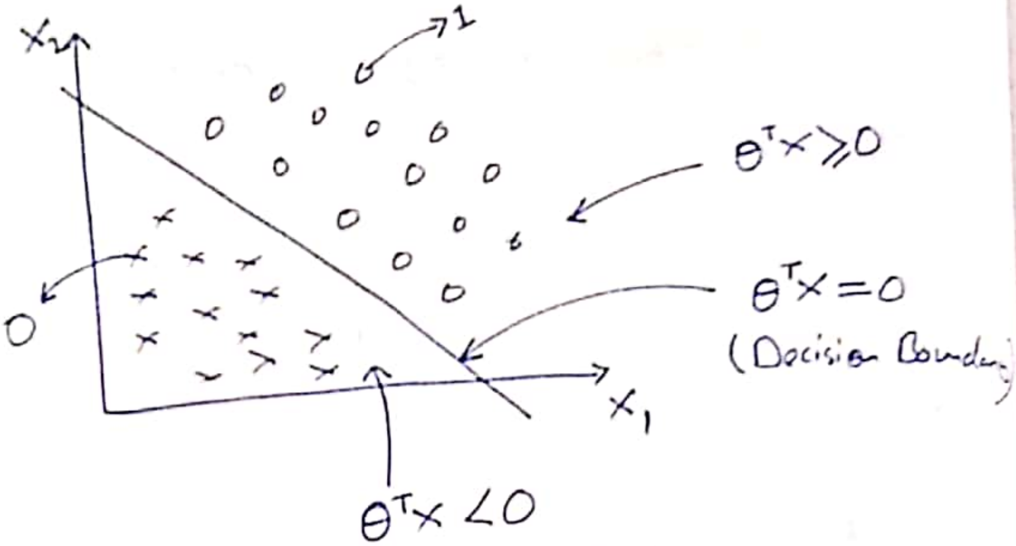


$$h_\theta(z) \geqslant 0.5 \Rightarrow z \geqslant 0$$

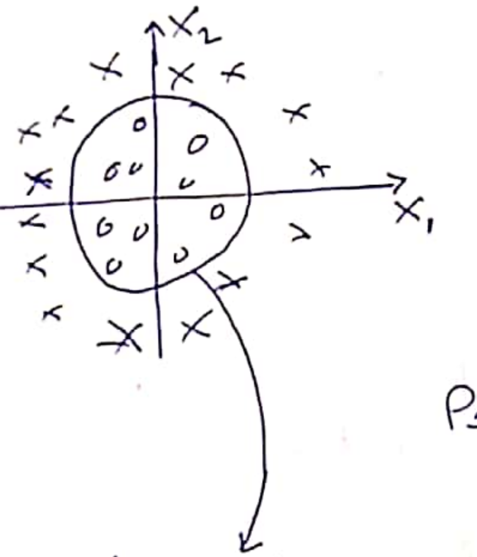
&

$$h_\theta(z) < 0.5 \Rightarrow z < 0$$

Scanned by CamScanner



Non linear decision boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Predict "$y=1$" if $-1 + x_1^2 + x_2^2 \geq 0$

$\Rightarrow x_1^2 + x_2^2 \geq 0$

decision boundario

$$\boxed{x_1^2 + x_2^2 = 1}$$

6.4) Cost function

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots (x^{(m)}, y^{(m)})\}$

{m examples}

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad x_0 = 1 \quad y \in \{0, 1\}$$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

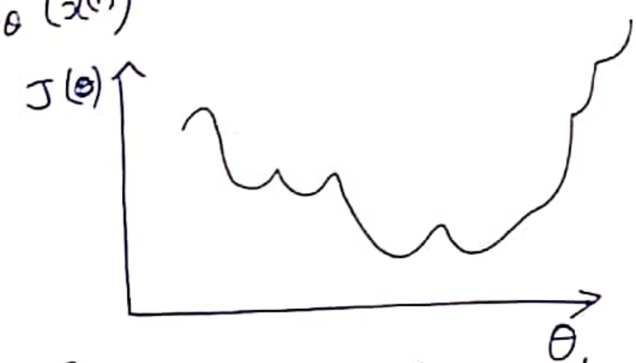
* **Cost function**

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_\theta(x^{(i)}), y\right)$$

⇒ If $Cost\left(h_\theta(x^{(i)}), y\right) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y\right)^2$

{Same as Linear regression}

⇒ then, $J(\theta)$ is not a convex function because of highly non linear term $h_\theta(x^{(i)})$



⇒ So using gradient descent will not guarantee global minima.

⇒ So to avoid this problem we will use the following as cost function.

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$

→ It give very high penalty if prediction is wrong and zero penalty if prediction is write.

→ It guarantee a convex function.
(Proving this is out of scope of this course)

6.5) Simplified Cost function and gradient descent

⇒ Simplified Cost function:

$$\text{Cost}(h_\theta(x), y) = -y(\log(h_\theta(x))) - (1-y)\log(1-h_\theta(x))$$

&

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

* **Gradient Descent**

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

Want $\min_\theta J(\theta)$:

Repeat {

$$\theta_j : \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta)$$

}

{Simultaneously update all θ_j}

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

6.6> Advanced Optimization

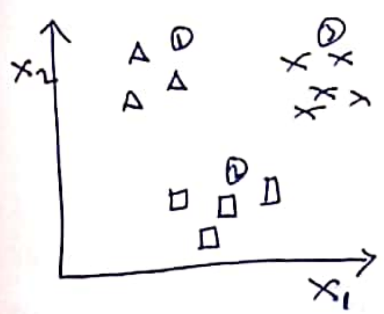
⇒ Other optimization algorithms: {other than gradient descent}

- Conjugate gradient
- BFGS
- L-BFGS

Advantages	Disadvantages
→ No need to manually pick α.	→ More complex
→ Often faster than gradient descent	

6.7> Multiclass-classification (one-vs-all)

⇒ Classification problem with more than one classes.



⇒ Turn this into three separate binary classification problem.

$h_\theta^{(1)}(x) \rightarrow \triangle$ Vs rest

$h_\theta^{(2)}(x) \rightarrow \square$ Vs rest

$h_\theta^{(3)}(x) \rightarrow \times$ Vs rest

① Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y=i$.

② On a new input x, to make a prediction, pick the class i that maximizes

$\max h_\theta^{(i)}(x)$

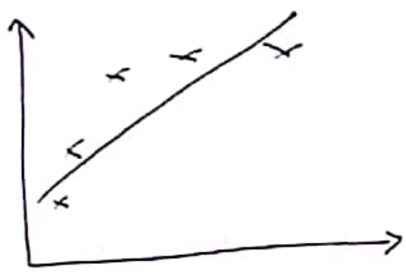
Scanned by CamScanner

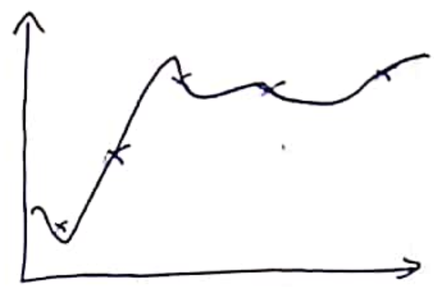
CHAPTER 5

Regularization

Regularization

7.1) Problem of Overfitting





$\theta_0 + \theta_1 x$

⇒ "Under fit"

⇒ It has "high bias"

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

⇒ "Overfit"

⇒ It has high variance

Overfitting

↳ If we have too many features, the learned hypothesis may fit the training set very well, but fails to generalize to example.

⇒ Overfitting can also be found in Classification.

* **Addressing Overfitting**

1. **Reduce number of features**
   - ⇒ Manually select which features to keep.
   - ⇒ Model selection algorithm
2. **Regularization**
   - ⇒ Keep all the features, but reduce magnitude of parameters θ_j.

7.2) Cost Function

⇒ Small values for parameters $\theta_0, \theta_1, \theta_2 \cdots \theta_n$

→ "Simpler" hypothesis.

→ Less prone to overfitting.

⇒ As we dont know which parameter is less important, so we will penalise all the parameters:-

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

regularization Parameter

{Objective :: fit the training data well}

{Objective : Keep the parameter small}

{It controls the trade off between the two Objective}

7.3) Regularized - Normal equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix}_{m \times (n+1)} \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}_{m \times 1}$$

$$\theta = \left(X^T X + \lambda \underbrace{\begin{bmatrix} 0 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{bmatrix}}_{(n+1)\times(n+1)} \right)^{-1} X^T y$$

7.4) Regularized - logistic regression

$$J(\theta) = -\left[\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1-h_\theta(x^{(i)})) \right]$$

$$+ \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Scanned by CamScanner