ML

Andrew Ng

Coursera

Week 3

Logistic Regression

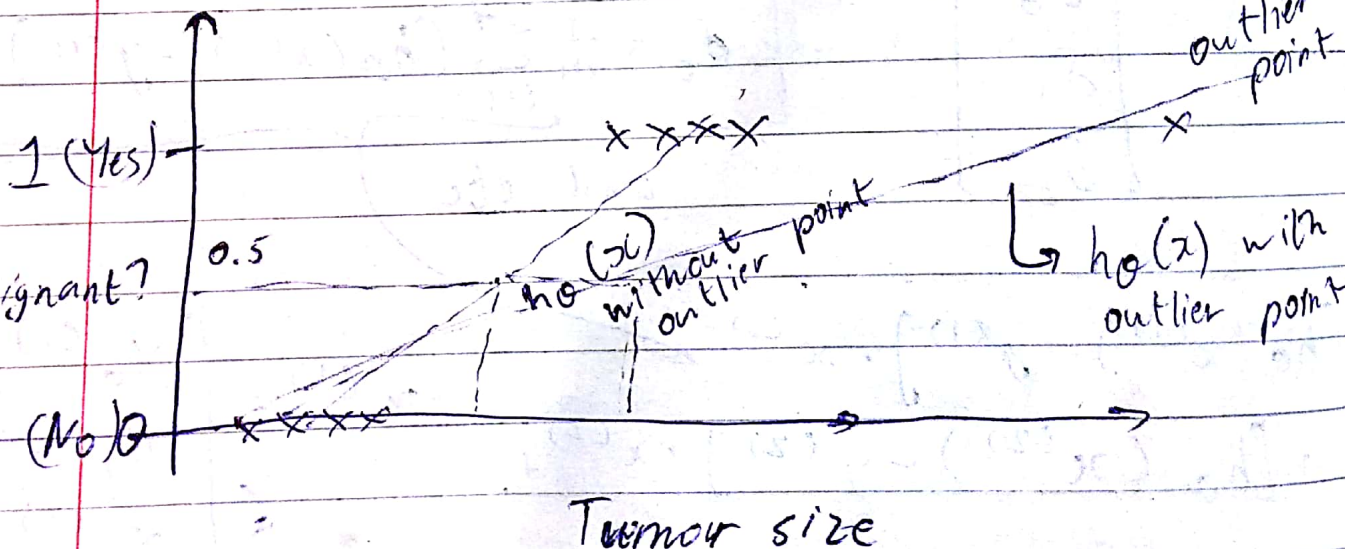
output y is a discrete variable (classification)

$y \in \{0, 1\}$ — negative class, positive class

y can be either (binary classification)

$y \in \{0, 1, 2, 3\}$ possible as well

multiclass classification



Assume we use linear ~~Logistic~~ regression to classify as 1 or 0. If $h_\theta(x) \geq 0.5$ then $y = 1$ ~~else~~, if $h_\theta(x) < 0.5$ then $y = 0$. It seems that linear regression

is predicting properly. When we add an outlier point & run linear regression the result turn to be wrong. (wrong predictions)

↳ regression does not work for a classification problem

logistic regression:

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

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

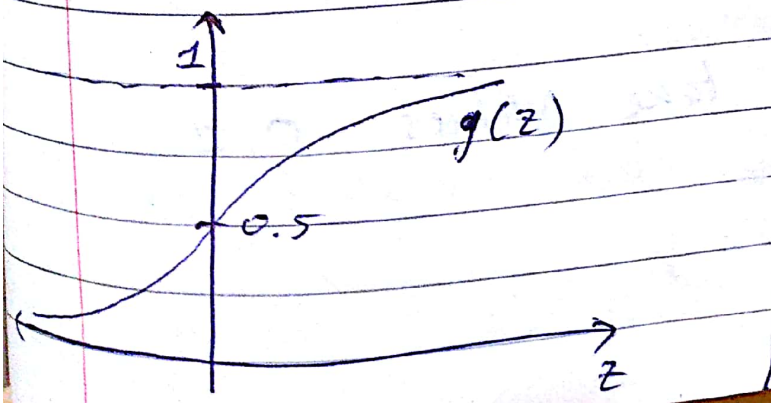
where $g(z) = \frac{1}{1+e^{-z}}$

sigmoid function

OR

Hence called logistic regression ←- - - - logistic function

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



Asymptotes at $y=1$ & $y=0$

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

fit θ to data now given training set

Interpretation of hypothesis output

$h_\theta(x)$ = estimated probability that $y = 1$ on input x

Example: if $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumor size} \end{bmatrix}$

then $h_\theta(x) = 0.7$

↳ Tells patient that there is a 70% chance of tumor being malignant on input x

~~$h_\theta(x) =$~~

Mathematically (more formally):

$h_\theta(x) = P(y = 1 \mid x; \theta)$

given x & given θ

"Probability that $y = 1$, given x parameterized by θ"

y can only take values 0, 1
hence:

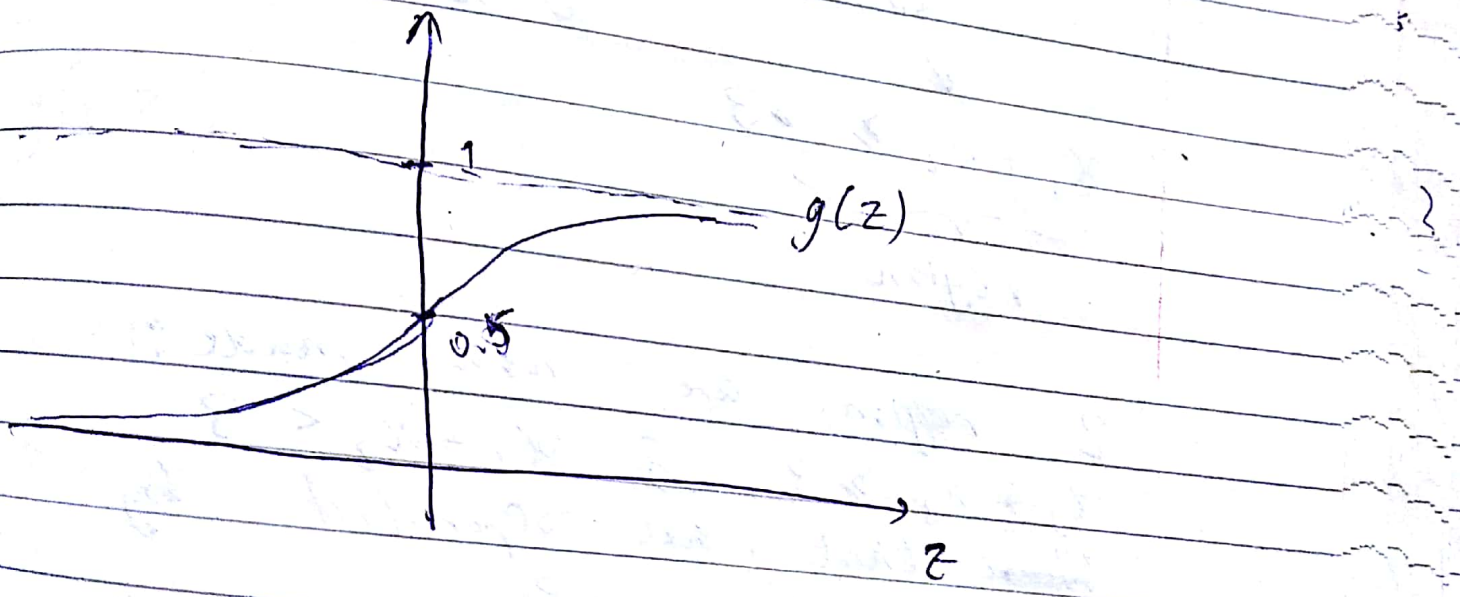
$P(y=0|x;\theta) + P(y=1|x;\theta) = 1$

$\hookrightarrow P(y=0|x;\theta) = 1 - P(y=1|x;\theta)$

Decision Boundary

Suppose predict "$y=1$" if $h_\theta(x) \geq 0.5$

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



$g(z) \geq 0.5$ when $z \geq 0$

$h_\theta(x) = g(\theta^T x) \geq 0.5$

whenever $\theta^T x \geq 0$

$g(z) < 0.5$ when $z < 0$

$h_\theta(x) = g(\theta^T x) < 0.5$ whenever

$\theta^T x < 0$

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

Suppose that we have our optimal values of θ such that:
$\theta_0 = -3, \theta_1 = 1, \theta_2 = 1$ hence:

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

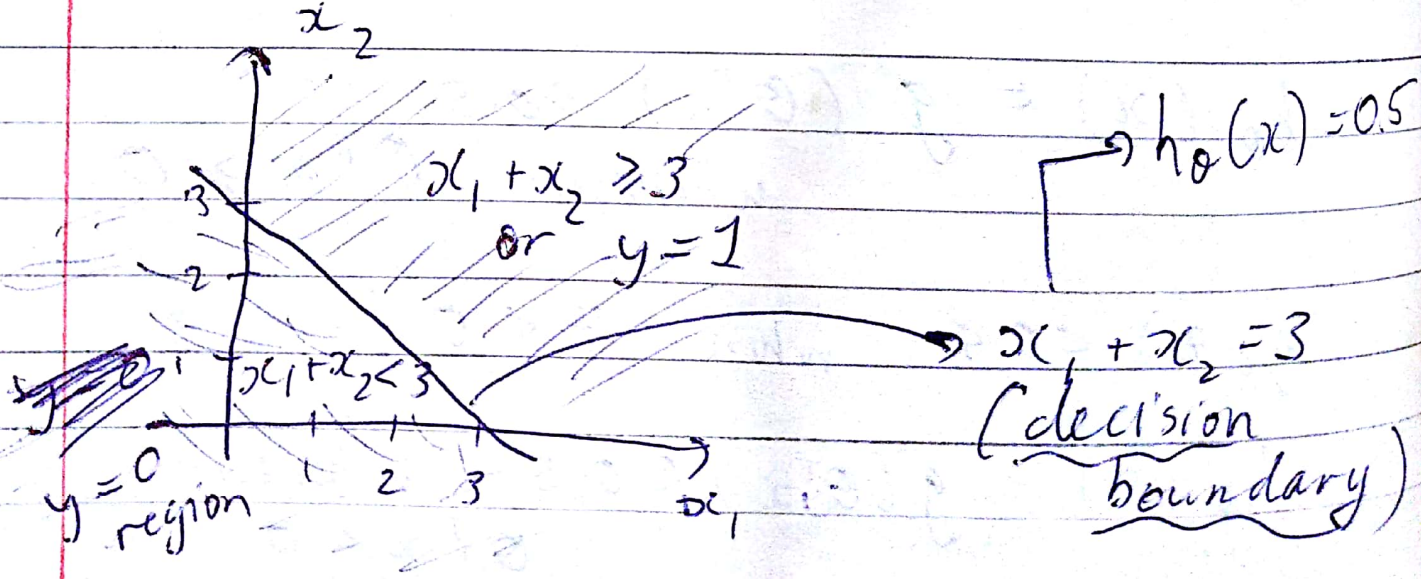
thus:
predict "$y = 1$" if
$\underbrace{-3 + x_1 + x_2}_{\theta^T x} \geq 0$

same ↗ ↙

$x_1 + x_2 \geq 3$

region

2 regions are now made:
$x_1 + x_2 \geq 3$ & $x_1 + x_2 < 3$
~~there~~ that are seperated by the line: $x_1 + x_2 = 3$



You can add higher order polynomial terms to logistic regression just as how it was done in ~~tic~~ polynomial regression

Example:

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

The decision boundary is a property of the hypothesis, not of the dataset

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

$z = 0,\ e^0 = 1 \rightarrow g(z) = \frac{1}{2}$

$z = \infty,\ e^{-\infty} \rightarrow 0,\ g(z) = 1$

$z = -\infty,\ e^{\infty} \rightarrow \infty,\ g(z) = 0$

Cost function

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

with m examples

$$x \in \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}}_{\mathbb{R}^{n+1}} \quad x_0 = 1, \quad y \in \{0, 1\}$$

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

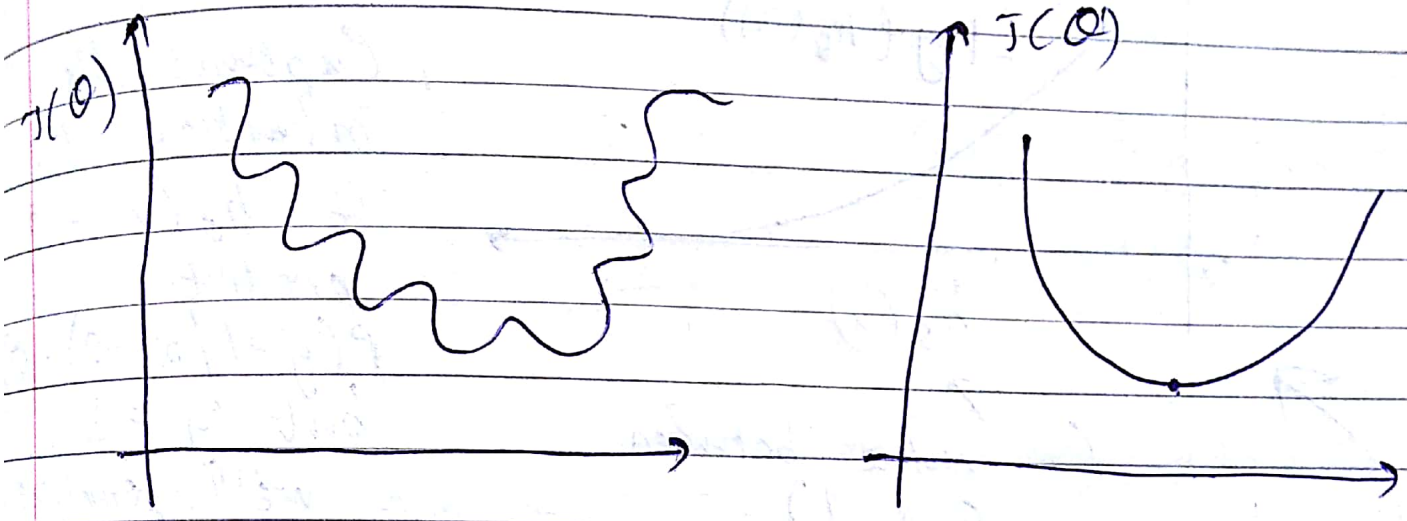
The cost function can generally be described as such:

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

for linear regression:

$$cost(h_\theta(x), y) = \frac{1}{2}(h_\theta(x) - y)^2$$

Using the same cost function as used in linear regression for logistic regression is possible but it is non-convex due to the non-linearity property of $h_\theta(x)$ in logistic regression



non-convex

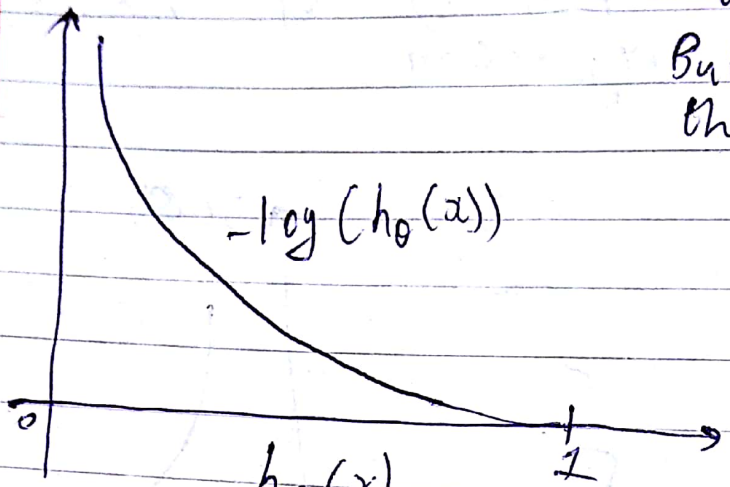
↳ has many local optima & gradient descent does not work well for it (does not necessarily converge to global minimum)

convex

convex for logistic regression (gradient descent can be applied to find global minimum)

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

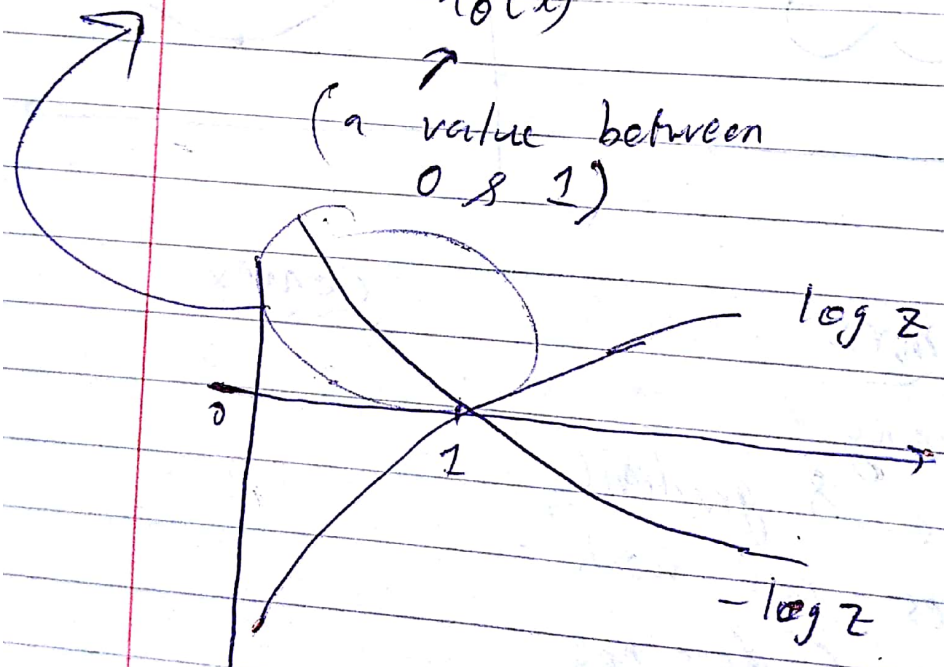
If $y = 1$:



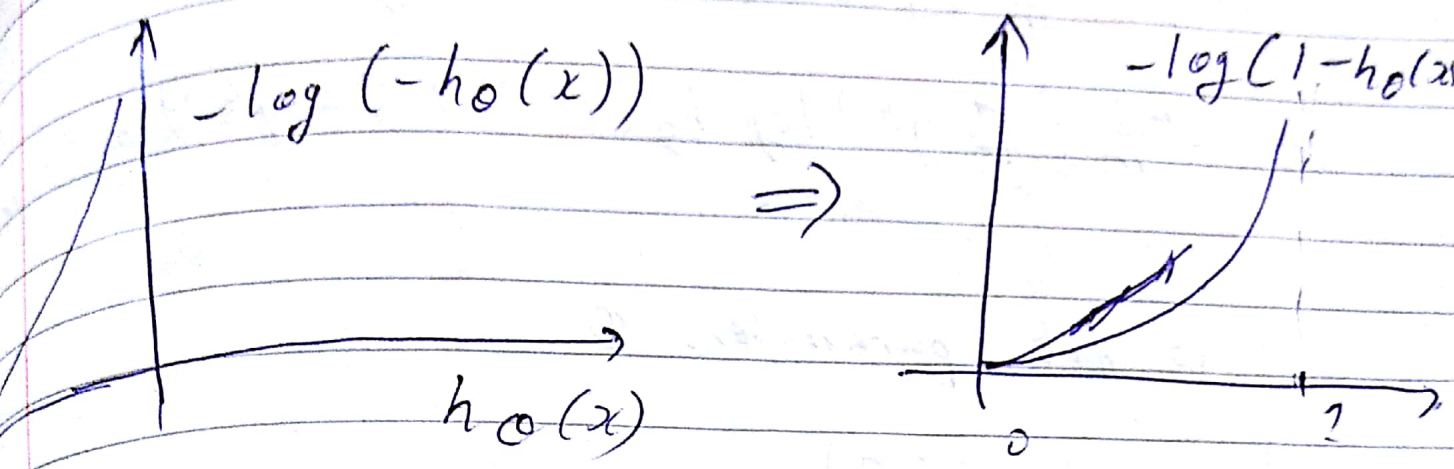
(a value between 0 & 1)

$Cost = 0$ if $y = 1$, $h_\theta(x) = 1$
But as $h_\theta(x) \to 0$
then $Cost \to \infty$

Captures the intuition that if $h_\theta(x) = 0$ (predict $P(y = 1 | x; \theta) = 0$) but $y = 1$, we'll penalize the learning algorithm by a very large cost



If $y = 0$:



Logistic Regression Cost Function

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

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

compact way of expressing equations

$$\text{cost}(h_\theta(x), y) =$$

$$= -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x))$$

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

If $y = 0$: $-\log(1-h_\theta(x))$

$\text{cost}(h_\theta(x), y)$

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

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

To fit parameters θ:

$$\min_\theta J(\theta)$$

Repeat {

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

}

(simult update for all θ_j)

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

exactly the same as differential of linear regression!

The value of $h_\theta(x^{(i)})$ is different hence not exactly the same formulas as linear regression

Feature scalling applies to logistic regression as well as gradient descent is used here

Vectorised implementation of cost func.

$h = g(X\theta)$ } $\mathbb{R}^m$ of predictions

$$J(\theta) = \frac{1}{m}\left(\underbrace{-y^T}_{1\times m}\underbrace{\log(h)}_{m\times 1} - \underbrace{(1-y)^T}_{1\times m}\underbrace{\log(1-h)}_{m\times 1}\right)$$

Vectorised implementation of gradient descent

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - y)$$

$n+1\times m \quad m\times 1$

$n+1\times 1$

Advanced Optimisation

Cost function $J(\theta)$ & we wish to find out $\min_\theta J(\theta)$

We need the following :

- $J(\theta)$ → to figure out how $J(\theta)$ converges or diverges
- $\frac{\partial}{\partial \theta_j} J(\theta)$ for $j = 0, 1, \ldots, n$

↳ used in gradient descent

↳ This is an optimisation algo

Other optimisation algorithms that require $J(\theta)$ & $\frac{\partial}{\partial \theta_j} J(\theta)$ to run:

1) Gradient descent
2) Conjugate gradient
3) BFGS
4) L-BFGS

} more sophisticated algos

Advantages & Disadvantages of using the other 3 algos:

Advantages	Disadvantages
- No need to manually pick α - faster than gradient descent	- more complex

Using the more complex algorithms

Suppose we are trying to find out the parameter $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

minimised for $\min_{\theta} J(\theta)$

where $J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$

We must know the gradients as well:

$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$

$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$

We can create a function that defines all the properties needed :

```
>> function [jVal, gradient] = costFunction(theta)
    jVal = ... ;
    gradient = zeros(2,1);
    gradient(1) = ...;
    gradient(2) = ...;
```

Then pipe this function into a function that finds the minimum of :

```
>> options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] = fminunc(@costFunction, initialTheta, options);
```

fminunc → function minimization unconstrained

functionVal → cost value at optTheta

@costFunction → pointer to the cost function

~~s~~ data structure that stores the options needed

'GradObj' : ~~set~~ 'on' ← means that gradient is being provided

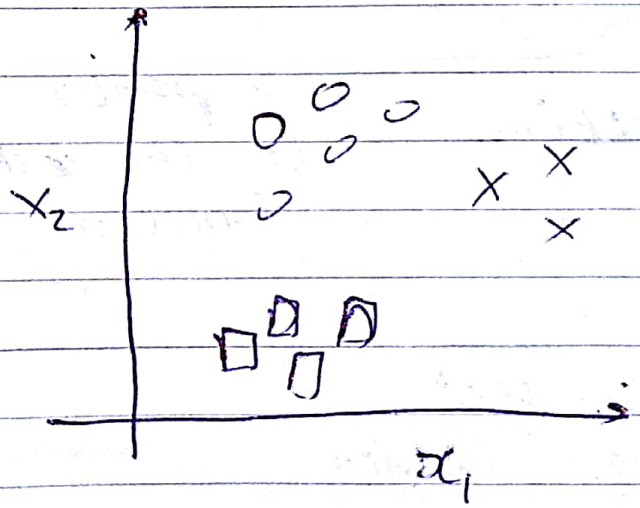
fminunc simply uses advanced optimisation algorithms to find the most minimal value

fminunc may not work with a single dimentional theta

Multiclass Classification
One-vs-all

Medical dignosis: Not ill, cold, flu
$y=1$ $y=2$ $y=3$

Weather: Sunny, Cloudy, Rain, Snow
$y=1$ $y=2$ $y=3$ $y=4$

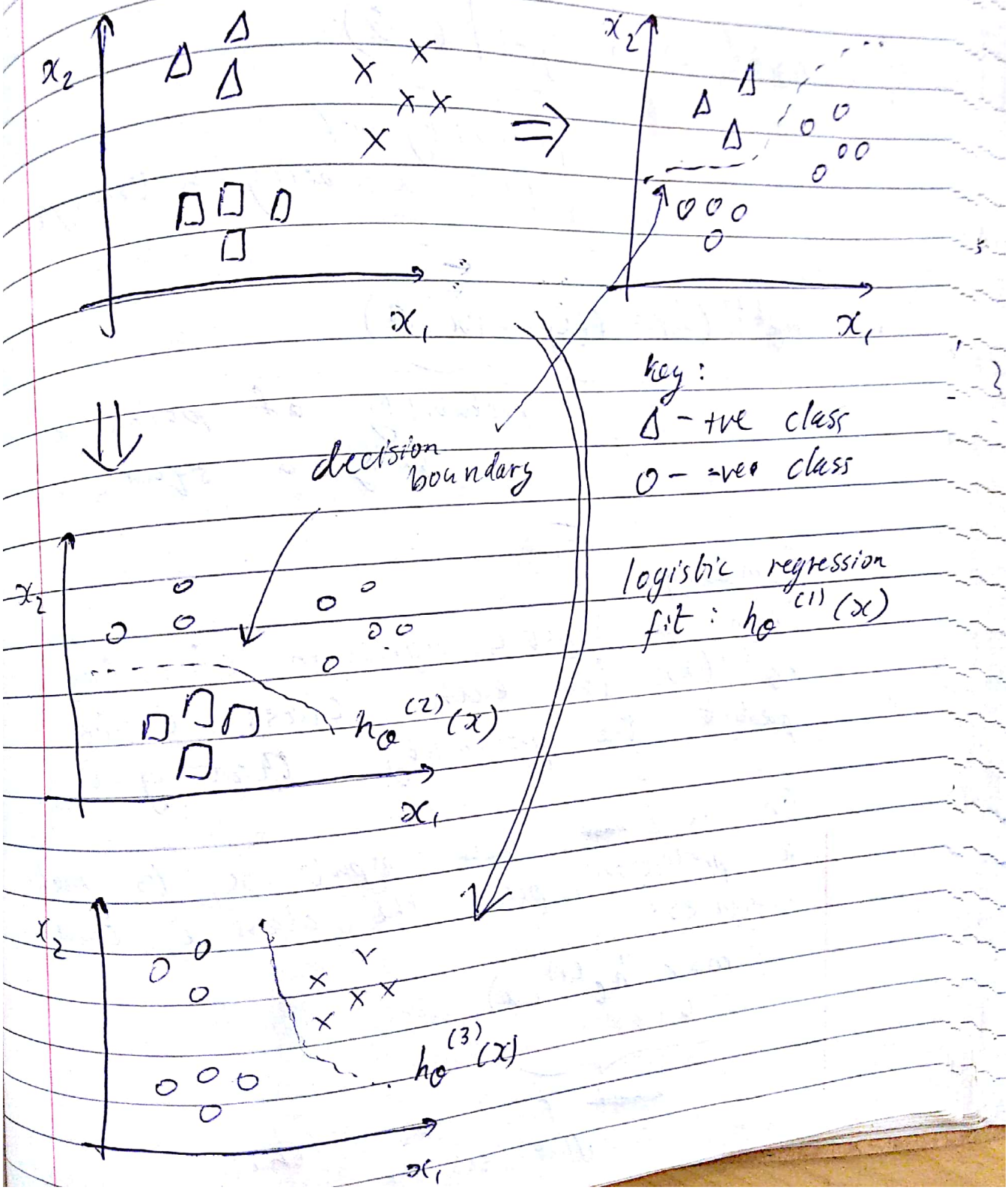


Multiple classes (3 to be exact) are shown

Use one-vs-all to classify multiple classes

One-vs-all / One-vs-rest:

Le break down multiclassification into 3 binary classifications



$h_\theta^{(i)}(x) = P(y = i \mid x; \theta)$

(for $i = 1, 2, 3$)

For $i = 1$:

$h_\theta^{(1)}(x) = \underbrace{P(y = 1 \mid x; \theta)}$

probability of point x being a triangle

$h_\theta^{(2)}(x) = \underbrace{P(y = 2 \mid x; \theta)}$

probability of point x being a square

Summary:

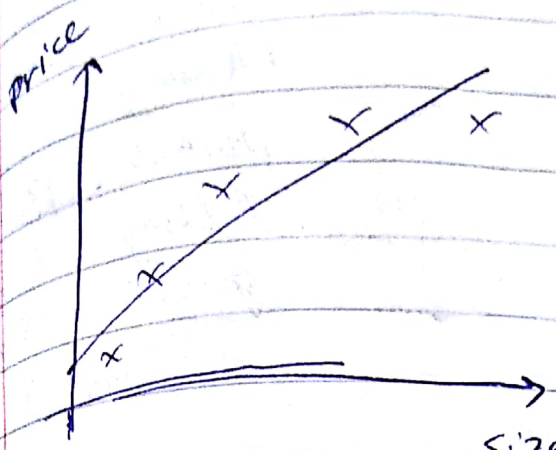
Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$

On a ~~new~~ new input x, to make a prediction, pick the class i that maximises:

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

~~which~~ find the i that gives us the greatest probability

Problem of overfitting

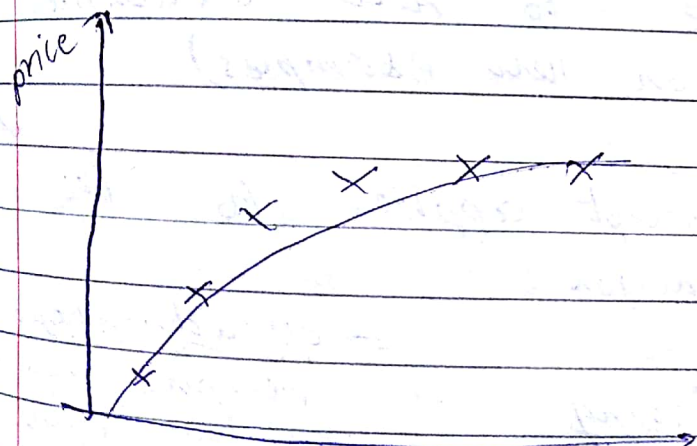


$\theta_0 + \theta_1 x$

"underfit" or "high bias" - the model does not fit the data well

Bias in this case is illustrated in the linear model - that as size increases, the house price will increase proportionally.

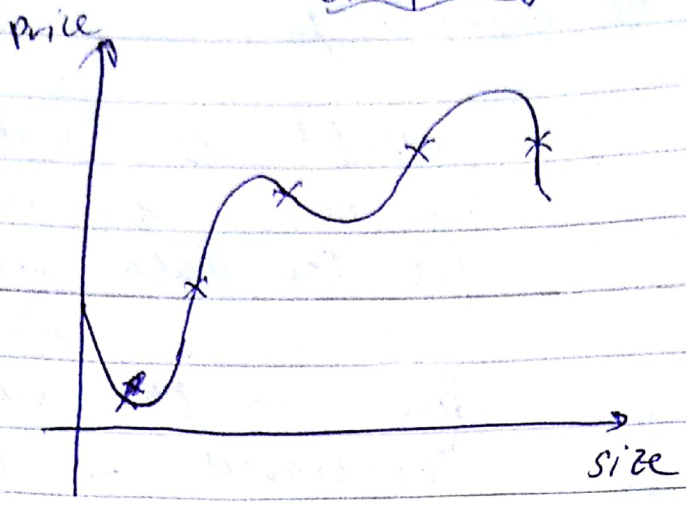
The data however shows that the prices tend to platea as prices keep on increasing.



"Just right"

$\theta_0 + \theta_1 x + \theta_2 x^2$

"overfitting" or "high variance"



- Curve passes through all the points. The fitting is really great (there is no cost)

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

"Overfitting" - if we have too many features, the learned hypothesis may fit the training set very well $\left[J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0 \right]$ but fail to generalise to new examples (predict prices on new examples)

This same concept applies to the ~~ap~~ logistic function

Addressing overfitting

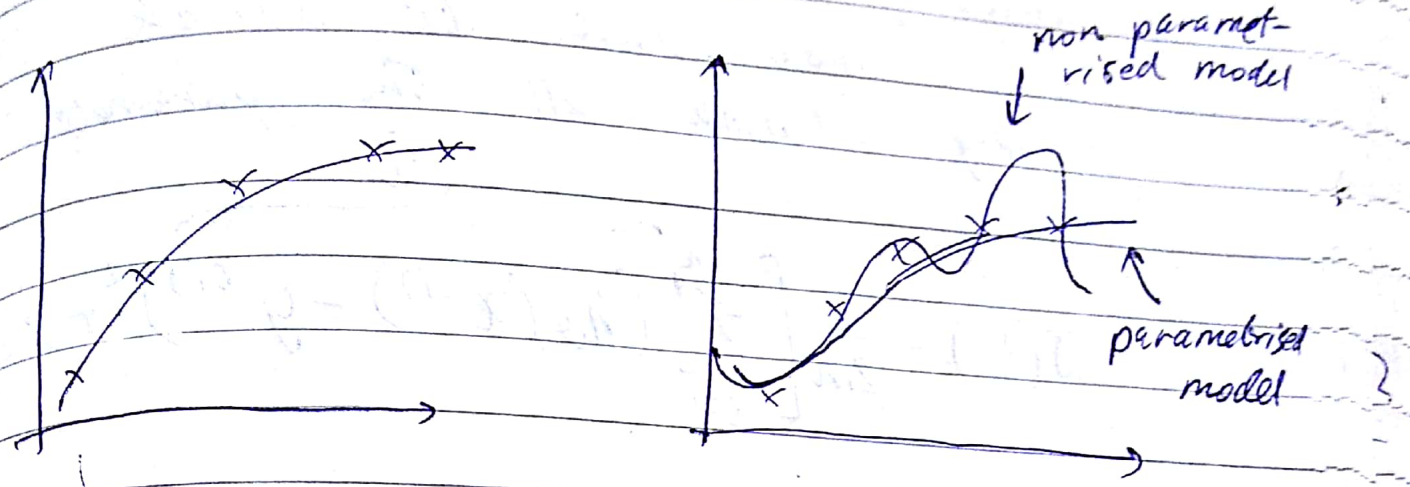
→ disadvantage: removes information you have about the problem

Options:

1. Reduce # of features
   - manually select which features to keep
   - model selection algorithms
     ↳ algo that determines which features to keep

2. Regularization

- keep all of the features, but reduce magnitude/values of parameters θ_j
- works well when we have a lot of features, each of which contributes a bit to predicting y



$\theta_0 + \theta_1 x_1 + \theta_2 x_2^2$

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

Suppose we penalise & make θ_3, θ_4 really small:

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\theta_3^2 + 1000\theta_4^2$$

↳ $\theta_3 \approx 0, \quad \theta_4 \approx 0$ ← almost

the same as getting rid of θ_3, θ_4 then the overall line becomes quite similar to a quadratic eqn

② → regularization term

with small contributions from θ_3 & θ_4

Small values for parameters $\theta_0, \theta_1, \ldots, \theta_n$ lead to:
- "simpler" hypothesis / smoother ~~func~~ graphs
- a hypothesis less prone to overfitting

When presented with situation where we don't know what to shrink then simply shrink all the parameters

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

↳ θ_0 is not penalised by convention

λ - regularisation parameter

↳ controls a trade off between 2 different goals:
① fitting the data well
② keeping the parameters low in value to avoid overfitting

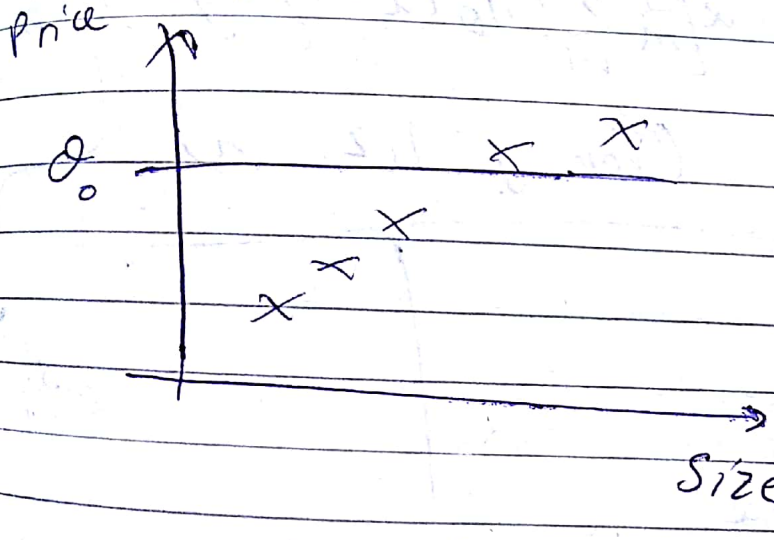
If λ is set to an extremely large value (example $\lambda = (0^{10})$ then all the parameters will be close to 0:

$\theta_1 \approx 0, \theta_2 \approx 0, \ldots, \theta_n \approx 0$

and hence it will close to eliminating terms in the hypothesis equation:

$h_\theta(x) = \theta_0 + \cancel{\theta_1 x} + \ldots + \cancel{\theta_n x}$

$\hookrightarrow h_\theta(x) \approx \theta_0$ thus making the hypothesis a horizontal line



This model is underfit for the data

Bias is high for this model

It can be said that λ is the value that pushes the model from ~~under~~ overfit to underfit. With a good enough value of λ, it will bring us in the middle but if the value ~~is~~ is too high then the model becomes underfit

Regularised linear regression

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

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

Regularised gradient descent:

Repeat {

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

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

(for $j = 1, 2 \ldots n$)

}

$x_j^{(i)}$

Note: θ_0 is not regularised hence $\frac{\partial}{\partial\theta_0}J(\theta)$ is same as before

can be simplified to

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

$1 - \alpha \frac{\lambda}{m} < 1$ because m is usually a large number & α is a small as well, $1 - \alpha \frac{\lambda}{m}$ is usually a little bit less than 1 example: 0.99
Thus θ_j is shrunk ~~with~~ a smaller ~~value~~ on each iteration

Regularising ~~linear~~ normal equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^m$$

$m \times (n+1)$

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

$(n+1) \times (n+1)$

Previously it was mentioned that if $m \leq n$ (# examples) $\leq$ (# features) then $\theta = (X^T X)^{-1} X^T y$

This part is singular/non-invertible

With regularization in the equation we can prove that if $\lambda > 0$ then:

$$\theta = \underbrace{\left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \right)^{-1}}_{\text{This part will always be non-singular}} X^T y$$

Thus regularization takes care of the singularity issue

Regularising logistic regression

Cost function without regularisation:

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

With regularization:

$$J(\theta) = -\left[\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}) \right.$$
$$\left. + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 \right. \quad (\text{for } \theta_1, \theta_2 \dots \theta_n)$$

Expressions are identical to those used in linear regression but $h_\theta(x)$ is different in ~~logistic~~ logistic & linear regression, hence evaluate to different expressions eventually

Gradient descent:

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

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

$$\frac{\partial}{\partial \theta_j} J(\theta)$$

}

θ_j is evaluated for $j = 1, 2, \ldots, n$