# Notes - Week 3

## Logistic Regression — classification

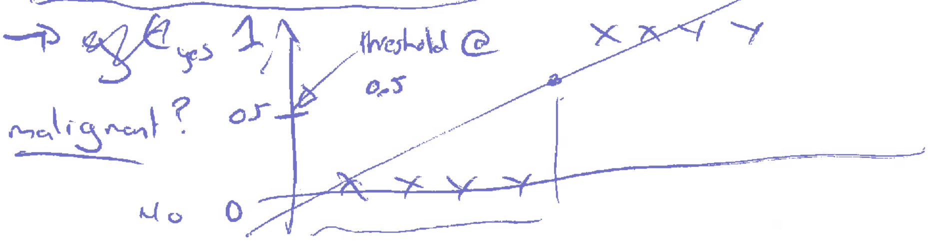
Classification problems
- email
- Online Transactions / fradulent etc
- Tumor: Malignant / benign

$y \in (0, 1)$ 0: "negative class" (benign tumor)
1: "positive class" (malignant tumor)

"absence of something"

"presence of something"

$h_\theta(x) = \theta^T x$ ← linear regression (hypothesis function)




Threshold classifier output $h_\theta(x)$ at 0.5

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

$h_\theta(x) \leq 0.5$, " "$y=0$"

Linear regression for classification

classification: $y = 0$ or $1$

$h_\theta(x)$ can be $\gg 1$ or $\ll 0$

so lets develop a new algorith such that $0 \leq h_\theta(x) \leq 1$

logistic regression. → classification algrotim (discrete value 0 or 1)

↑ historical (confusing)

# ~~Week 2~~ Notes – Week ~~2~~ 3

## Logistic Regression – hypothesis representation

what is the function that we are going to use to represent our hypothesis when we have a classification problem?

### Logistic Regression model

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

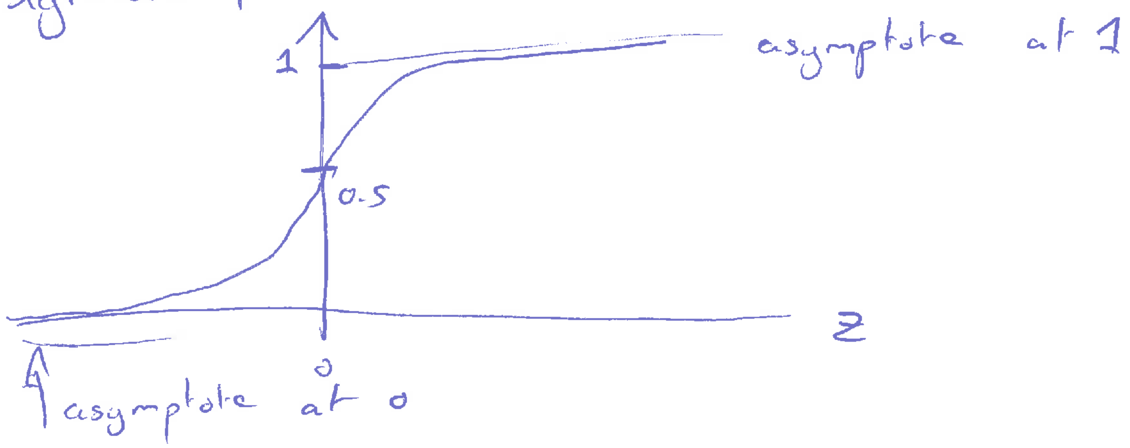
$h_\theta(x) = (\theta^T x$ ← linear regression

logistic regression

$h_\theta(x) = g(\overset{z}{\theta^T x}) \rightarrow g(z) = \frac{1}{1+e^{-z}}$ ← sigmoid function (logistic function)

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

sigmoid function.




### Interpretation of hypothesis output $h_\theta(x)$

~~check~~ how we are going to treat the ~~data~~ output of our hypothesis function using the sigmoid function.

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

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

$h_\theta(x) = 0.7$.

Tell patient 70% chance of tumor being malignant.

$h_\theta(x) = P(y=1 \mid x; \theta)$ → "probability that $y=1$, given $x$, parameterized by $\theta$"

Notes - Week ~~2~~ 3

Logist Regression - hypothesis function contd.

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

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

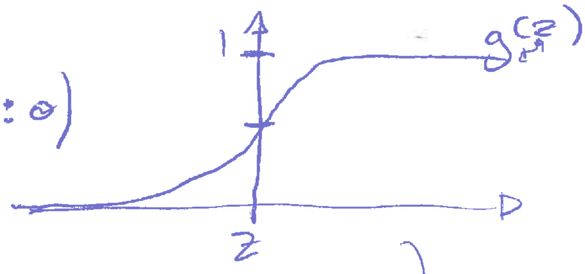
# Notes - Week 3

## Logistic Regression - Decission bandary

Logistic regressio

$h_\theta(x) = g(\theta^T x) = P(y=1|x;\theta)$

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




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

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

$g(z) \geq 0.5$

whenever

$\underset{\theta^T x}{\underset{\parallel}{z}} \geq 0$

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

whenever $\theta^T x \geq 0$

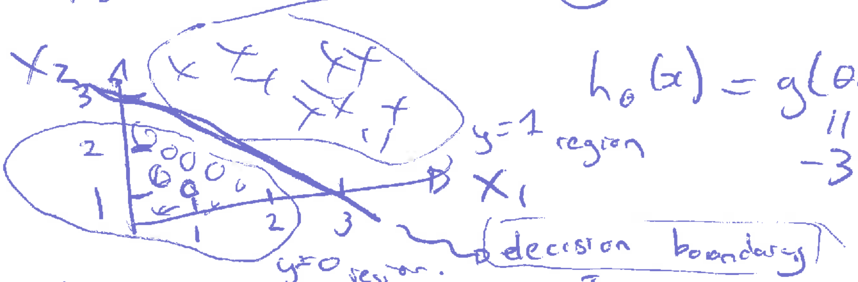
Similarly

If: $g(z) < 0.5$ then.

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

$\theta^T x < 0$

## Decision Bandary




$h_\theta(x) = g(\underset{-3}{\underset{\parallel}{\theta_0}} + \underset{1}{\underset{\parallel}{\theta_1}} x_1 + \underset{1}{\underset{\parallel}{\theta_2}} x_2) \quad \therefore \theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

ok so....

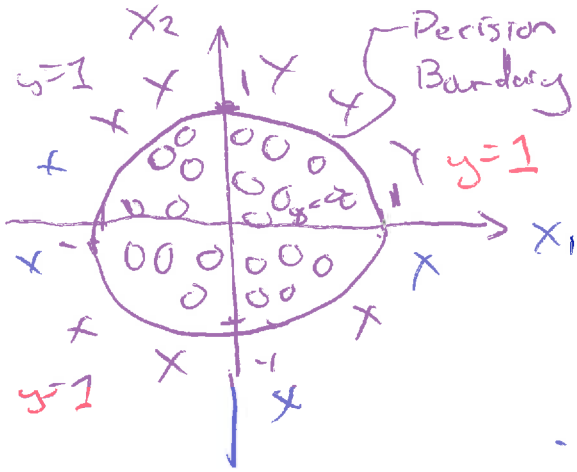
Predict "$\underline{y=1}$" if $\overbrace{-3 + x_1 + x_2}^{\theta^T x} \geq 0$

$\rightarrow x_1 + x_2 \geq 3$

$\boxed{\text{decision boundary}}$ is a property of the hypothesis (and the parameters there of) and **not** of the data set.

Notes - Week3

## Logistic Regression - Decision Boundary Cont...




Non linear decision boundaries

consider our hypothesis looks like this

$$h_\theta(x) = g(\underset{-1}{\theta_0} + \underset{0}{\theta_1} x_1 + \underset{0}{\theta_2} x_2 + \underset{1}{\theta_3} x_1^2 + \underset{1}{\theta_4} x_2^2)$$

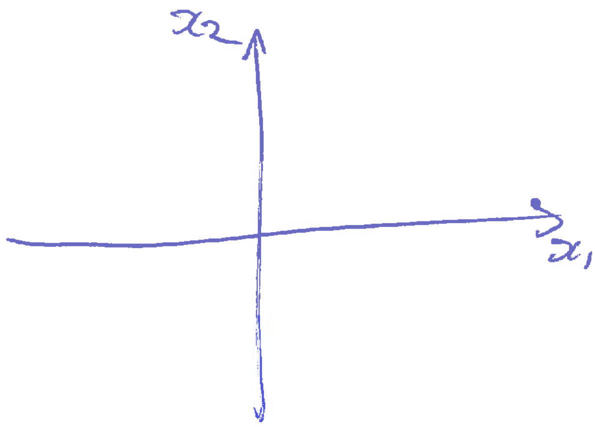
$\therefore$ our parameters matrix $\theta$ looks like

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

**problem:** predict "y=1" if $\underbrace{-1 + x_1^2 + x_2^2}_{} \geq 0$.

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

more complex example:




suppose

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$

# Week 3 - Notes

## Logistic Regression Model — Cost function

training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

m examples

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

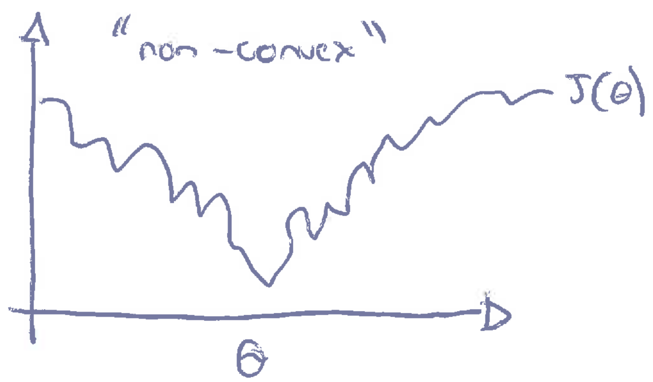
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

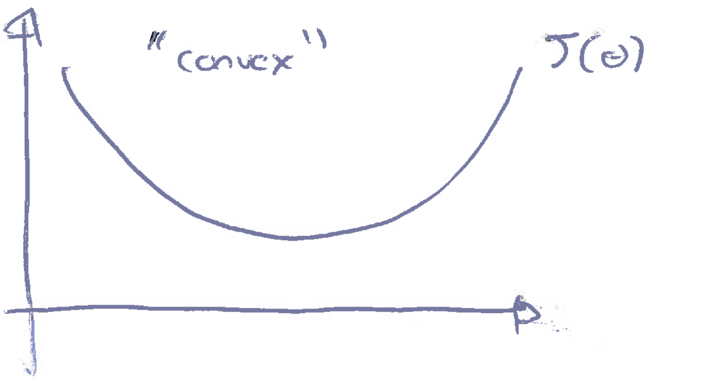
How to choose parameters $\theta$?

Linear regression:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}_{\text{squared error term}}$$

$$\mathrm{cost}\left(h_\theta(x^{\cancel{(i)}}), y^{\cancel{(i)}}\right) = \frac{1}{2}\left(h_\theta(x^{\cancel{(i)}}) - y^{\cancel{(i)}}\right)^2$$



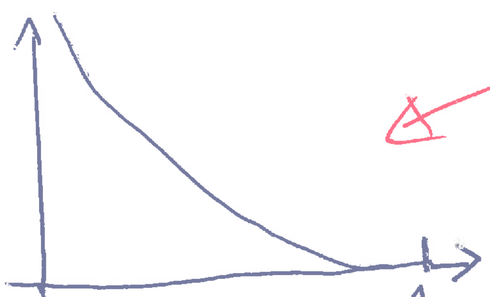
# Week 3 – Notes

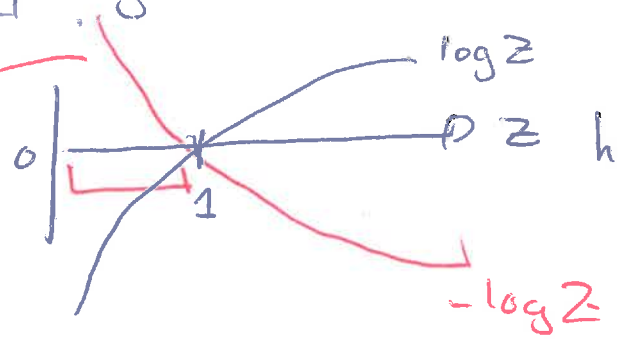
## Logistic Regression Model – Cost function

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

if $y=1$

cost function $y=1$ $\boxed{-\log(h_\theta(x))}$ $\log z$




Cost $= 0$ if $y=1$, $h_\theta(x) = 1$

but as $h_\theta(x) \to 0$

cost $\to \infty$

Captures intuition that if $h_\theta(x) = 0$

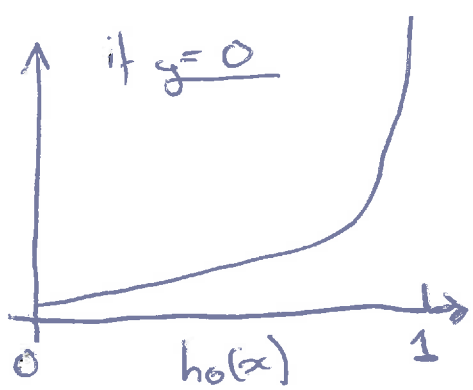
(predict $P(y=1 \mid x;\theta) = 0$ but $y=1$

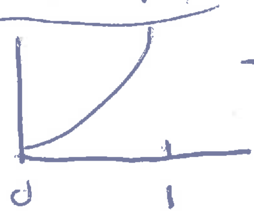
probability

we'll penalize learning algorithm by a very large cost.

If $y=0$ → $\boxed{-\log(1-h_\theta(x))}$ – cost

$-\log(1-z)$



# Week 3 - Notes

## Logistic Regression - Simplified cost function and gradient descent

logistic regression cost function

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

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

this is an easier way of writing the following

$\boxed{y = 0 \text{ or } 1 \text{ always}}$

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

switches terms on and off when needed

derived from statistics using the principal of maximum likelyhood estimation.

To fit parameter $\theta$:

$$\min_\theta J(\theta) \rightarrow \text{Get } \theta$$

To mak a prediction given new $x$:

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

$p(y=1 \mid x; \theta)$

{probability that $y=1$, given input $x$ and parameterized by $\theta$)

# Week 3 - Notes

## Logistic Regression - Simplified cost function and gradient descent

so using gradient descent on our cost function:

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

want min $J(\theta)$:

repeat {

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

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

(simultaneously update all $\theta_j$

---

feature scaling works for logistic regression too

# Week 3 - Notes

## Logistic Regression - Advanced optimization

ok so we have the situation

cost function $J(\theta)$ Want $\min_\theta J(\theta)$

Given $\theta$, we have code that can compute

- $J(\theta)$ ✓
- $\boxed{\frac{\partial}{\partial \theta_j} J(\theta)}$ (for $j = 0, 1, \dots, n$)

Gradient descent. (again)

Repeat {

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

}

---

**Optimization algorithms**

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

↳ do not need to understand to apply...

**Advantages**

- No need to manually pick $\alpha$ ✓
- often faster than gradient descent.

**Disadvantages**

- More complex.

Week 3 - Notes

## Logistic Regression - Advanced Optimization

Cont'd

Advanced optimization example...

$\Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ ∴ values that minimise $J(\theta) =$ $\theta_1 = 5$, $\theta_2 = 5$

Matlab code

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

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

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

```
function [jVal, gradient]
   = costFunction (theta)
jVal = (theta(1)-5)^2 + ....
       (theta(2)-5)^2;

gradient = zeros (2, 1);
gradient (1) = 2*(theta(1)-5);
gradient (2) = 2*(theta(2) -5);
```

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

octave notation to reference value is:

$$\text{theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \begin{matrix} \rightarrow \text{theta}(1) \\ \rightarrow \text{theta}(2) \\ \\ \rightarrow \text{theta}(n+1) \end{matrix}$$

function [jVal, gradient] = costFunction (theta)

  jVal = [ code to compute $J(\theta)$ ]

  gradient (1) = [ " " " $\frac{\partial}{\partial \theta_0} J(\theta)$ ]

  gradient (2) = [ " " " $\frac{\partial}{\partial \theta_1} J(\theta)$ ]

  ⋮

  gradient(n+1) = [code to compute $\frac{\partial}{\partial \theta_n} J(\theta)$ ];

# Week 3 - Notes

## Multiclass classification: One vs all

example:
let's say you want to automatically tag your email with

Email tagging: Work, Friends, Family, Hobby

$y=1$, $y=2$, $y=3$, $y=4$

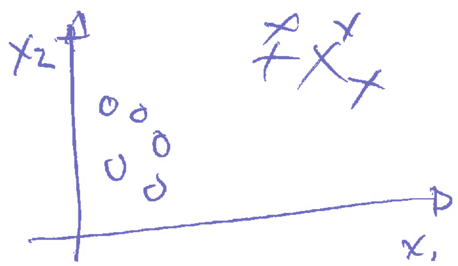
Medical diagrams: Not ill, cold, flu

$y=1$, 2, 3

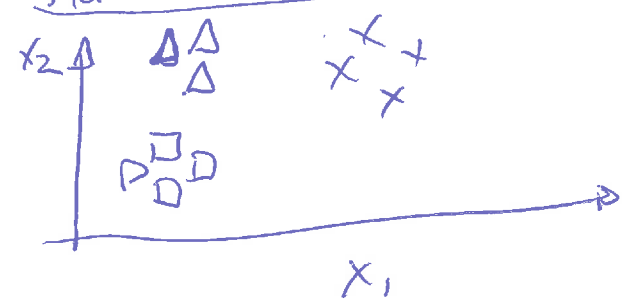
Weather: Sunny, Cloudy, Rain, Snow

$y=$ 1, 2, 3, 4

### Binary Classification

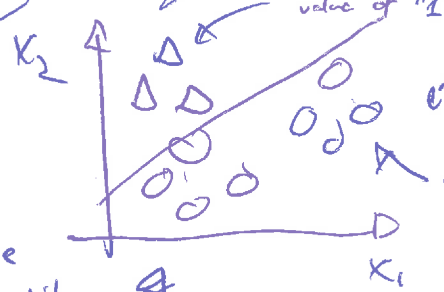



### Multi-Class classification

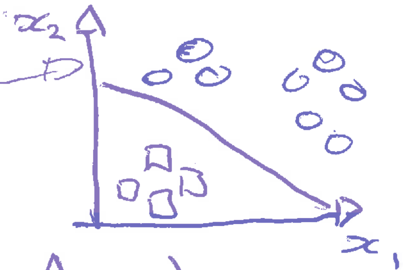



### One vs. All

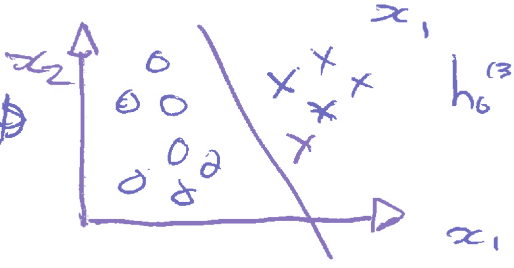

essentially mapping this data to a fake data set like so




essentially $\boxed{h_\theta^{(1)}(x)}$ → $P(y=1|x;\theta)$.




$h_\theta^{(2)}(x)$




$h_\theta^{(3)}(x)$.

class 1: △

class 2: □

class 3: ✕

$h_\theta^{(i)}(x) = P(y=i|x;\theta)$ $(i=1,2,3$

that is the probability that $y=i$ given $x$ and parameterized by $\theta$

## One-vs-all

The basic idea is :

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

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

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

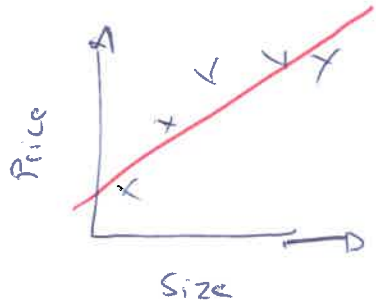
~~here~~ for example run all 3 classifiers on the input $x$, then pick class $i$ that maximises the 3.

# Week 3 - Notes

## Regularization - the problem of overfitting

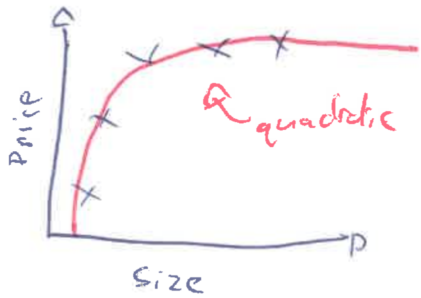
"tries' too hard to fit the training set"

example: linear regression (housing prices)



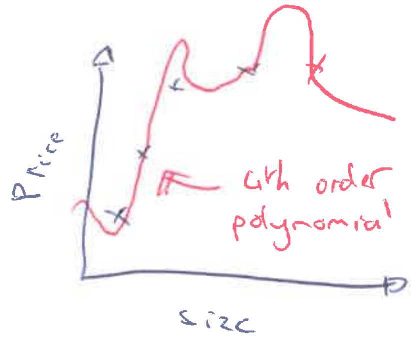

$\theta_0 + \theta_1 x$

→ This (the red line) is a bad ~~mod~~ model of the housing price data, this is known as "underfitting" or "high bias"




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

"Just right"




$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$ ← 4th order polynomial

This example shows what is known as "overfitting" or "high variance"

$\boxed{\text{Overfitting}}$: If we have too many features, the learned hypothesis may fit the training set very well

$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right)^2 \simeq 0$), but fail to generalize to new examples (predict prices on new examples)

# Week 3 - Notes

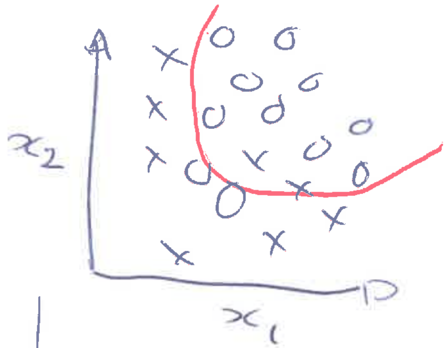
## Regularization - the problem of overfitting

example: logistic regression

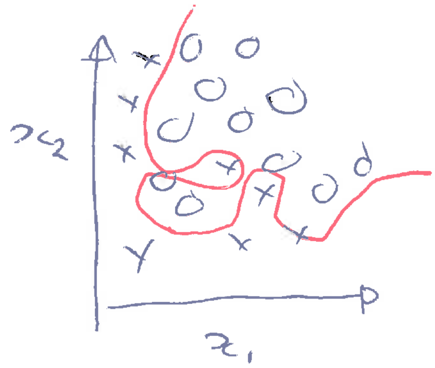

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

($g$ = sigmoid function)

"underfit"




$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$




$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots$

"overfit"

---

## Addressing overfitting

$x_1$ = size of house

$x_2$ = # of bedrooms

$x_3$ = # of floors

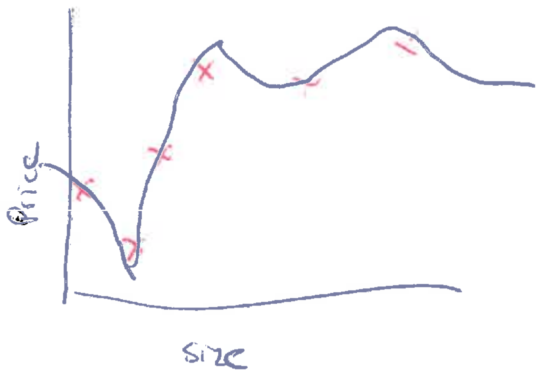
$x_4$ = age of house

$x_5$ = average income of neighborhood

$x_6$ = kitchen size

⋮

$x_{100}$




~~too~~ too many features, not enough training data, overfitting can become a problem.

---

## Addressing overfitting:

Options:

1. Reduce number of features
   - → manually reduce which features to keep
   - → model selection algorithm.

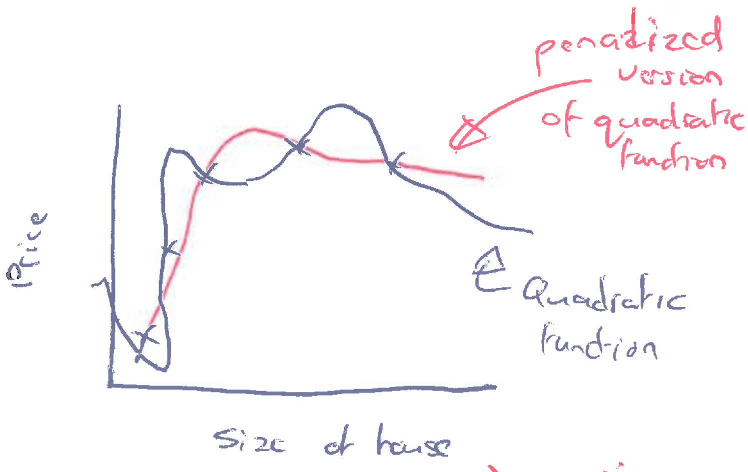
2. Regularization
   - Keep all the features but reduce magnitude / values of parameters $\theta_j$
   - works well when we have a lot of features, each of which contributes a bit to predicting $y$

# Week 3 - Notes

## Regularization - Cost function

### Intuition


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




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

Suppose we penalize and make $\theta_3, \theta_4$ really small

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

$\therefore \quad \theta_3 \approx 0 \qquad \theta_4 \approx 0$

---

### Regularization

Small values for parameters $\theta_0, \theta_1, \ldots, \theta_n$ ←

→ "Simpler" hypothesis ←

→ Less prone to overfitting. ←

$\theta_3, \theta_4 \approx 0$

Housing:

- Features: $x_1, x_2, \ldots, x_{100}$
- Parameters: $\theta_0, \theta_1, \theta_2, \ldots, \theta_{100}$

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

modify cost function to shrink all parameters

$\theta_1, \ldots \theta_{100}$

with the exception of

$\boxed{\theta_0}$

# Week 3 - Notes

## Regularization - Cost function

regularization term

2nd goal

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

regularization parameter

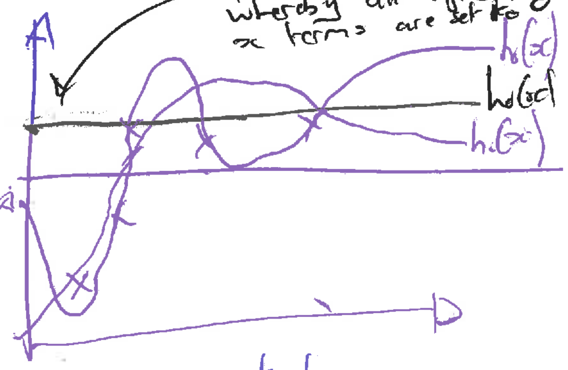
$\min_\theta J(\theta)$

$\lambda$ controls a trade off between two different goals.

1st goal: We would like to fit the the training data well, by minimizing the error.

2nd goal: We want to keep the parameters small

$\boxed{\lambda}$ controls the trade off between these two goals




if $\lambda$ is very very large

$\theta_1, \theta_2, \theta_3, \theta_4 \approx 0$ (approach)

$\therefore$ for ~~an~~ a hypothesis e.g

$h_\theta(x) = \theta_0 + \boxed{\cancel{\theta_1 x} + \cancel{\theta_2 x^2} + \cancel{\theta_3 x^3}}$

$\hookrightarrow$ approximates a straight line ie this is an example of "underfitting"

Week 3 - Notes

## Regularization - Linear Regression

Gradient descent

Repeat {

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

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

$$(j = \cancel{0}, 1, 2, 3, \ldots n)$$

Combining the two yields:

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

remains relatively the same

$$\rightarrow 1 - \alpha \frac{\lambda}{m} < 1$$

→ So here we are multiplying $\theta_j$ by a number a little bit less than one (for all $\theta_j$)...

---

## Normal equation

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

$m \times (n+1)$

m - dimensional vector $\mathbb{R}^m$

$$\rightarrow \min_\theta J(\theta) \qquad \frac{\partial}{\partial \theta_j} J(\theta) \overset{set}{=} 0$$

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

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

minimises $\theta$ when we are not using regularization

e.g. $n=2$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# Week 3 - Notes

## Non-invertibility (optional/advanced)

~~Non-inverti~~

normal equation, suppose $m \leq n$
(#examples) (#features)

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

↳ non-invertible/singular

pinv inv.

if $\lambda > 0$,

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

(don't get this)

# Week 3 – Notes

## Regularized Logistic Regression


$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \ldots)$$

### Cost function

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

$\theta_1, \theta_2, \ldots \theta_n$.

### implementing regularized gradient descent (algorithm for regularized logistic regression)

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

$$\theta_j = \theta_j - \alpha\underbrace{\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]}_{j = \cancel{0}, 1, 2, 3, \ldots, n.} \rightarrow \frac{\partial}{\partial\theta_j}J(\theta)$$

hypothesis is different → using sigmoid function...

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

Week 3 - Notes

## Regularized Logistic Regression

### Advanced optimization - Matlab code.

function [jVal, gradient] = costFunction(theta)

jVal = [code to compute $J(\theta)$
(cost function with regularized term)

gradient(1) = [code to compute $\frac{\partial}{\partial \theta_0} J(\theta)$];
↳ algorithm for $\theta_0$

gradient(2) = [code to compute $\frac{\partial}{\partial \theta_1} J(\theta)$];

⋮

gradient(n+1) = [code to compute $\frac{\partial}{\partial \theta_n} J(\theta)$];

need to create a vector:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \begin{array}{l} \text{theta(1)} \\ \text{theta(2)} \\ \\ \text{theta(n+1)} \end{array}$$

cost function needs to return jVal.

fminunc(@costfunction)