Contents

[Week 1 3](#_Toc473193089)

[What is Machine learning 3](#_Toc473193090)

[Linear Regression with One Variable 3](#_Toc473193091)

[Model and Cost Function 3](#_Toc473193092)

[Parameter Learning 6](#_Toc473193093)

[Week 2 7](#_Toc473193094)

[Environment Setup Instructions 7](#_Toc473193095)

[Multivariate Linear Regression 7](#_Toc473193096)

[Multiple Features; Using matrix to calculate hypothesis function 7](#_Toc473193097)

[Gradient Descent for Multiple Variables 8](#_Toc473193098)

[Gradient Descent in Practice 8](#_Toc473193099)

[Feature scaling 8](#_Toc473193100)

[Features and Polynomial Regression 9](#_Toc473193101)

[Additional Topics From Stanford own Course Materials 13](#_Toc473193102)

[Probabalistic Interpretation 13](#_Toc473193103)

[Locally weighted linear regression 14](#_Toc473193104)

[Week 3 14](#_Toc473193105)

[Classification and Representation 14](#_Toc473193106)

[Classification 14](#_Toc473193107)

[Hypothesis Representation 14](#_Toc473193108)

[Decision Boundary 15](#_Toc473193109)

[Logistic Regression Model 16](#_Toc473193110)

[Cost function 16](#_Toc473193111)

[Simplified Cost function and Gradient Descent 17](#_Toc473193112)

[Advanced Optimization 18](#_Toc473193113)

[Multiclass classification 18](#_Toc473193114)

[Solving The Problem of Overfitting. 18](#_Toc473193115)

[The Problem OF oVerfitting 18](#_Toc473193116)

[Cost function 19](#_Toc473193117)

[Regularized Linear Regression 19](#_Toc473193118)

[Regularized Logistic regression 20](#_Toc473193119)

[Another algorithm for maximizing l(theta) 20](#_Toc473193120)

[Generalized Linear Models 20](#_Toc473193121)

[Remarks from solving the Home Assignement 21](#_Toc473193122)

[Week 4 22](#_Toc473193123)

[Motivations 22](#_Toc473193124)

[Non Linear Hypothesis 22](#_Toc473193125)

[Neurons and the brain 22](#_Toc473193126)

[Neural Networks 22](#_Toc473193127)

[Model Representation I 22](#_Toc473193128)

[Model Representation II 23](#_Toc473193129)

[Applications 24](#_Toc473193130)

[Examples and Intuitions I 24](#_Toc473193131)

[Examples and Intuitions II 24](#_Toc473193132)

[Multiclass Classification 24](#_Toc473193133)

ML

26.12.2016

12:51

# Week 1

## What is Machine learning

Machine Learning – Learning from experience E to improve the perfomance P at task T.

**Supervised Learning**

Classification – map output to discrete function; Regression – map output to continuous function.

**Unsupervised learning**

We dont know the „supposed to be resul“ before. We try to find structure in data. Example: Clustering.

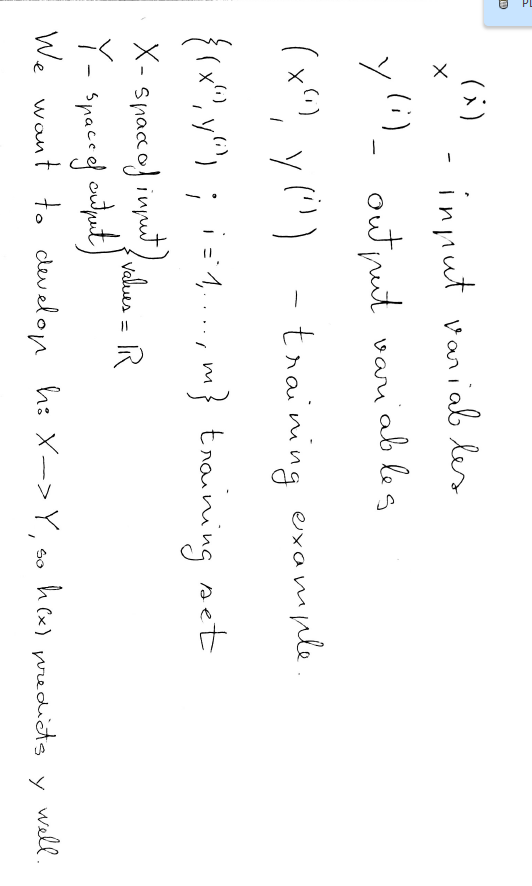
## Linear Regression with One Variable

### Model and Cost Function

**Model Representation**

<https://www.coursera.org/learn/machine-learning/supplement/cRa2m/model-representation>

i: i-th training example.

****

**Cost function**

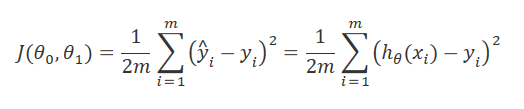
https://www.coursera.org/learn/machine-learning/supplement/nhzyF/cost-function

By Definition:

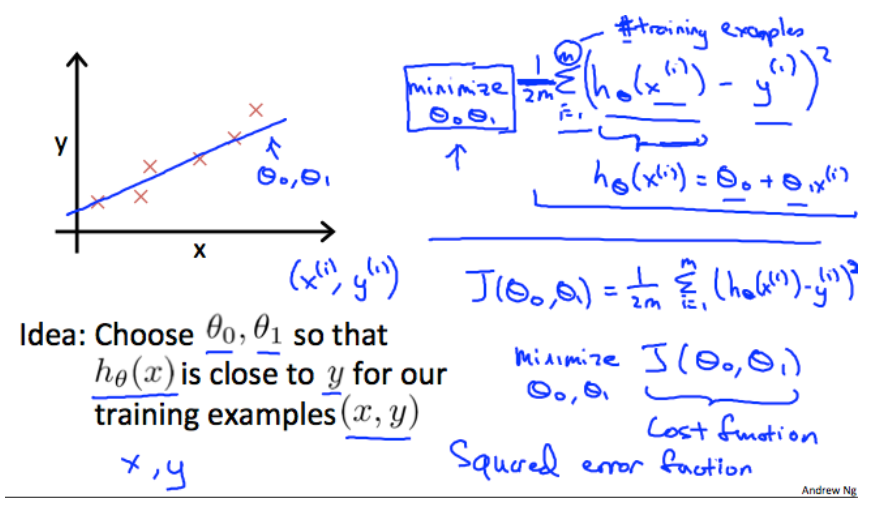
Overally it is the mean squared function from the hypothesis to the training example.

**m: number of taining samples**

**i: i-th training example**



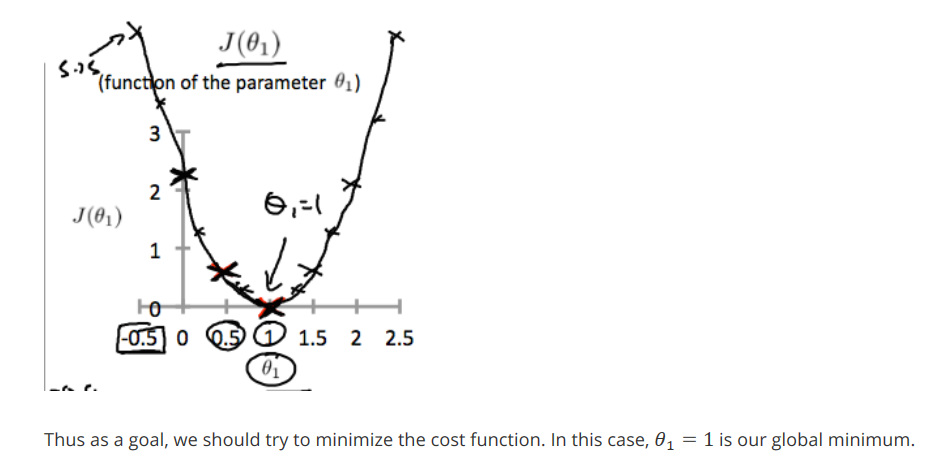
Overal purpose of the cost function:

****

**Cost function – Intuition I**

https://www.coursera.org/learn/machine-learning/supplement/u3qF5/cost-function-intuition-i

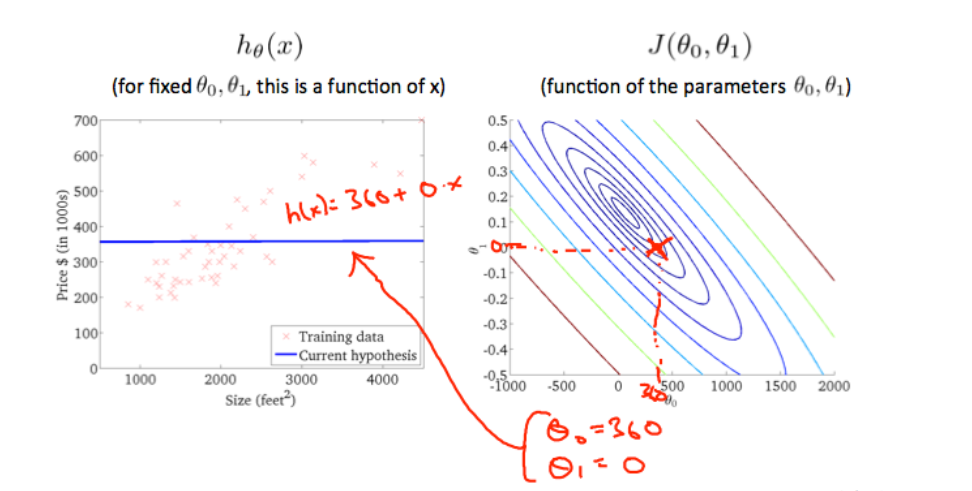
Cost function depends on hypothesis function matching with the training examples. Different hyphothesis functions naturally have different mappings, therefore also different matchings.

****

**Cost function – Intuition II** Multiple parameters.

https://www.coursera.org/learn/machine-learning/supplement/9SEeJ/cost-function-intuition-ii

The concept is not so much different from the previous „Intuition I“ but we use a contour plot to demonstrate te cost function with multiple input parameters: J(θ0,θ1) an example:

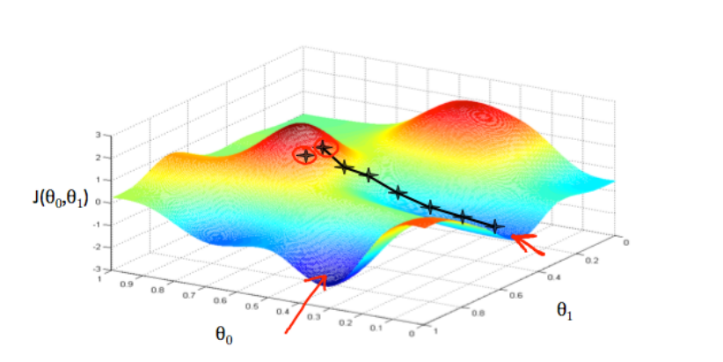
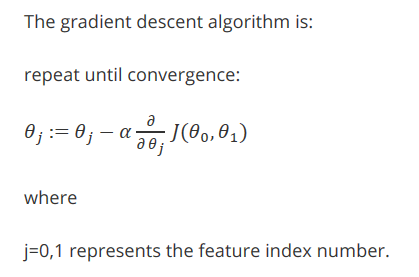
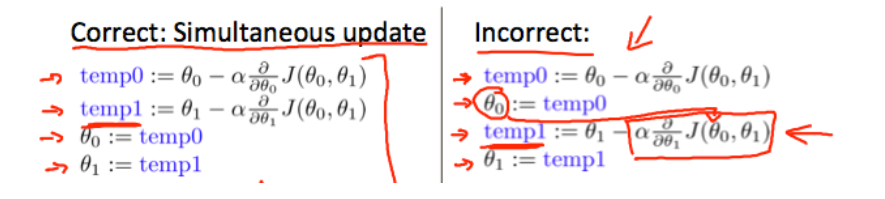
****

### Parameter Learning

**Gradient Descent:**

<https://www.coursera.org/learn/machine-learning/supplement/2GnUg/gradient-descent>

Convergence: Minimum. That means in real life, where the bettering of a certain function in every iteration is less than some convergence variable.

Additional Simplistic Example of Gradient Descent

28.12.2016

**Gradient Descent Intuition**

<https://www.coursera.org/learn/machine-learning/supplement/QKEdR/gradient-descent-intuition>

Convex function – No local minimas. Reference to week 1 folder



<http://mathcircle.berkeley.edu/BMC4/Handouts/inequal/>

Viimase pdf-I link. Päris kasulik. Listib regulaarsed convex functionid.

Gradient descent converges to minimum even when using a fixed step.

**Gradient Descent For Linear Regression**

<https://www.coursera.org/learn/machine-learning/supplement/U90DX/gradient-descent-for-linear-regression>

Here I had a moment for thinking: Gradient Descent acts through taking the derivative of cost function to get the **direction** and **amplitude** of ascent and moves the hypothesis function multiplicative constants opposite to that **direction** with a step based on the **amplitude**. (e.g. minimizes the cost function). The convergence is guaranteed due to the **amplitude** getting smaller and smaller when the cost function gets closer to minimum.

# Week 2

https://www.coursera.org/learn/machine-learning/supplement/WKgbA/multiple-features

## Environment Setup Instructions

Kasutan Octave, aga MatLab syntaxiga.

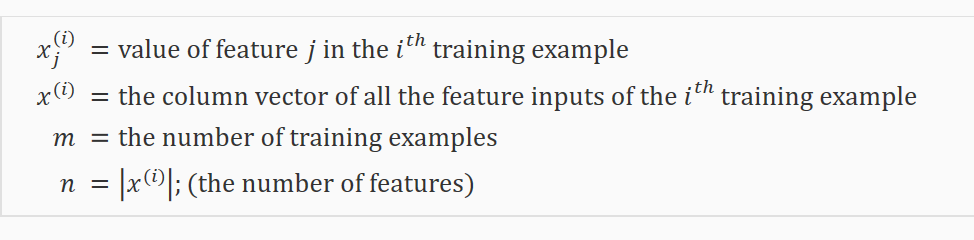
29.12.2016

## Multivariate Linear Regression

### Multiple Features; Using matrix to calculate hypothesis function

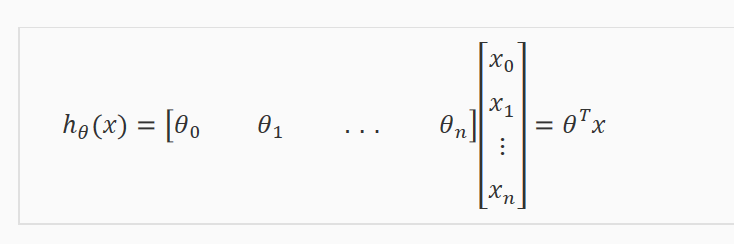
<https://www.coursera.org/learn/machine-learning/supplement/WKgbA/multiple-features>

In this chapter, we convert the training examples and the variable for hypothesis function into vectors and matrixes, so we could use matrix multiplication for computation simplicity.



X0 = 1

We create 1 dimensional vector with n columns from the hypothesis function parameters. And we convert our training examples into matrixes with m rows and n+1 columns.



### Gradient Descent for Multiple Variables

<https://www.coursera.org/learn/machine-learning/supplement/aEN5G/gradient-descent-for-multiple-variables>

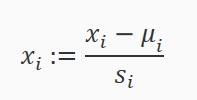
Pretty much the same logic as with one variable, but here I just add parameters. For every parameter I calculate the partial cost function derivative. And take it from the previous theta to minimize cost function using gradient directions.

## Gradient Descent in Practice

### Feature scaling

***μi* is mean**

***si* is std or max-min range**

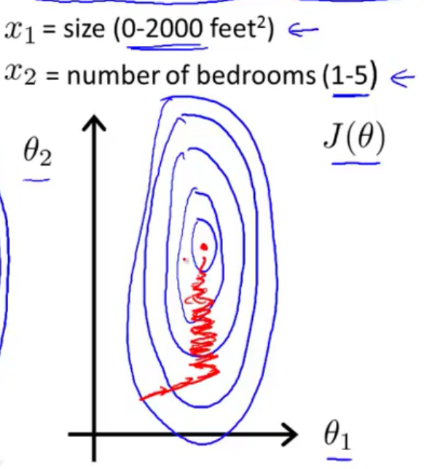


https://www.coursera.org/learn/machine-learning/supplement/CTA0D/gradient-descent-in-practice-i-feature-scaling

Why in one case the iteration number is larger than in other? As the starting point is random, without feature scaling the iterations necessary to get to the correct location for theta j is different from theta I, as the correction step comes from a different scale. (The gradient **amplitude** balances the step size anyway) If we do feature scaling, then the number of iterations necessary for all the thetas will be more similar & we wont waste resources.

Additional From wiki:

Since the range of values of raw data varies widely, in some [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms, objective functions will not work properly without [normalization](https://en.wikipedia.org/wiki/Normalization_%28statistics%29). For example, the majority of [classifiers](https://en.wikipedia.org/wiki/Statistical_classification) calculate the distance between two points by the [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance). If one of the features has a broad range of values, the **distance will be governed by this particular feature**. Therefore, the range of all features should be normalized so that each fmueature contributes approximately proportionately to the final distance



**Learning Rate**

<https://www.coursera.org/learn/machine-learning/supplement/TnHvV/gradient-descent-in-practice-ii-learning-rate>

**Automatic convergence test.** Declare convergence if J(θ) decreases by less than E in one iteration, where E is some small value such as 10−3. However in practice it's difficult to choose this threshold value.

If α is too small: slow convergence.

If α is too large: ￼may not decrease on every iteration and thus may not converge.

### Features and Polynomial Regression

**SETTING DATA (X) TO POLYNOMIAL WONT AFFECT THE CONVEXNESS OF THE FUNCTION – I KNOW THAT**

**I DON’T KNOW HOWEVER, ABOUT WHETHER SETTING THETA TO POLYNOMIAL WOULD AFFECT CONVEXNESS. SOME POLYNOMIALS CAN BE NON CONVEX it seems like, because polynomial complexity is a big topic. sssss**

**https://www.coursera.org/learn/machine-learning/supplement/ITznZ/features-and-polynomial-regression**

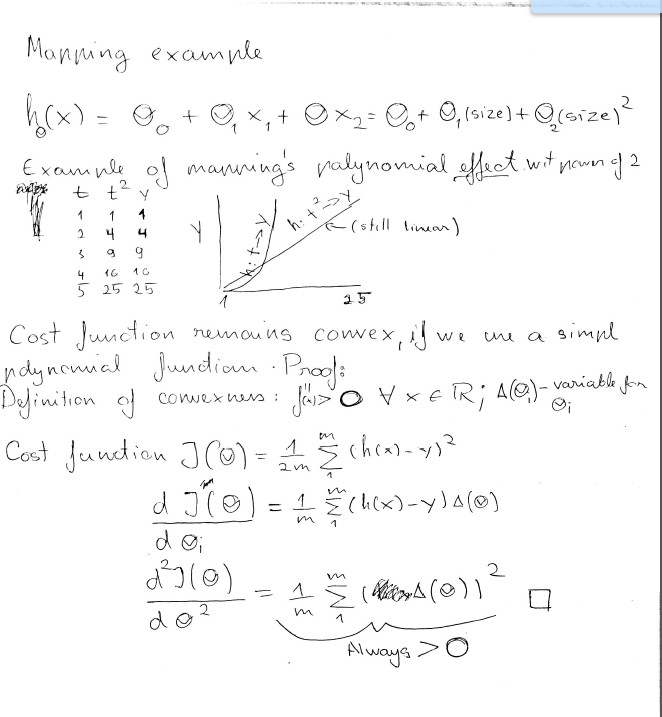
Spent 4 hours on this.

Polynomial regression – in essence it is writing down the relationship (hypothesis function) between vector x and y through a polynomial function instead of a linear function.

In Coursera video tutorial they use a mapping to do this procedure as shown below, but its not mandatory. It might simplify understanding, but in my case it actually slowed down the understanding process.

t = input. x

t\*\*2 = input mapping to x\*\*2 (elementwize)

****

It is possible to show, that Polynomial functions with same overall models than linear functions are also convex, therefore have a global minimum. As showed it by second derivative. ☺ *Read about convex function from the previous chapters*

Some ideas about the Polynomial Regression mapping that are good to keep in mind

* Mapping x to for example x\*\*2 changes to a more complex shape of the h(x) therefore also in highly likelihood changes the Cost function minimum.
* Using cost function takes place with having constant input parameters x and constant output parameters y and it tries to find the best kind of thetas based on these constant variables to form a minimum distance function to the learning data. If the thetas therefore are provided in the same shape, there is no principal difference in the cost function mathematically/analytically.
* If t and t\*\*2 differences in input are constant, t\*\*2 differences in output are not constant, whereas t output differences are constant.

One important thing to keep in mind is, if you choose your features using polynomial regression way then feature scaling becomes very important.eg. if x1 has range 1 - 1000 then range of x12 becomes 1 - 1000000 and that of x13 becomes 1 – 1000000000

**COMPUTING PARAMETERS ANALYTICALLY**

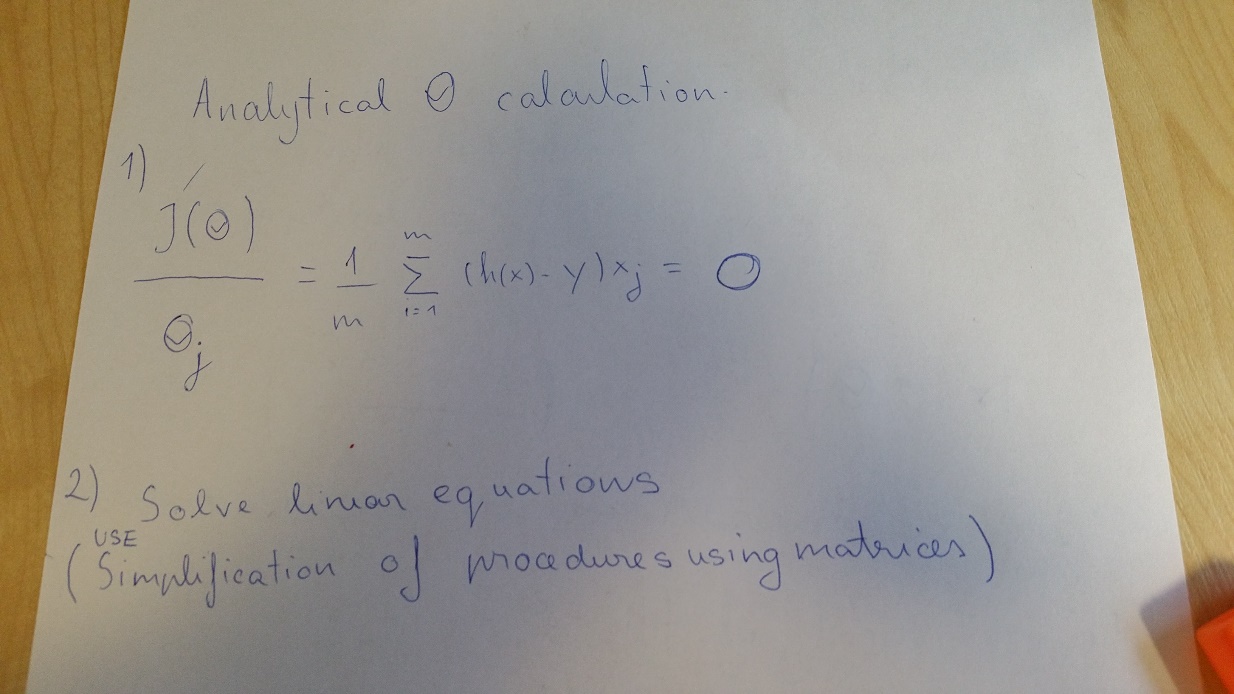
These Articles provide more underlying theory behind Normal Equation.

<http://eli.thegreenplace.net/2014/derivation-of-the-normal-equation-for-linear-regression/>



The main idea basically is, that we try to set the derivative of the full cost function to 0 and using matrixes, we want to calculate the appropriate thetas. There is no iteration in this solution. Only solving equations through matrix calculations.

The linear equation we get has n rows and n columns, and in very very high likelihood the rows and columns are independent, so the determinant != 0. So we have 1 solution accodring to Knoecker and Cappell theorem.



<http://eli.thegreenplace.net/2015/the-normal-equation-and-matrix-calculus/>



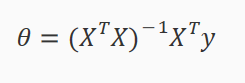
Clarifies, that matrix derivatives in essence dont diverge from normal calculus as we are taking the derivatives on basis of each separate variable in a matrix. EG matrix derivatives are not magic.

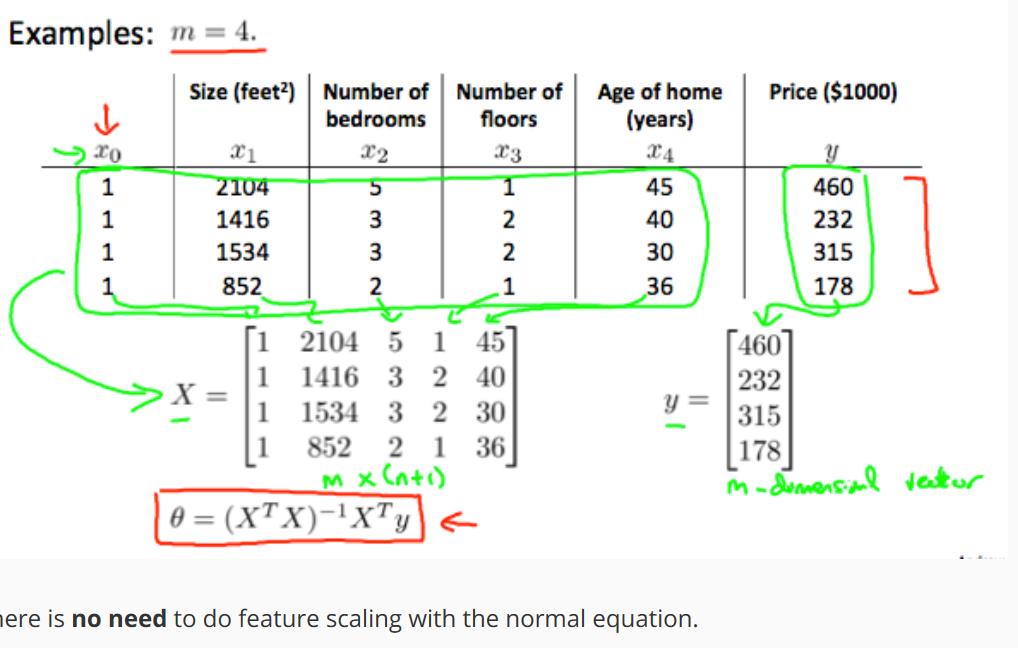
Stanford PDF # 1

Implementation

**Normal Equation**

https://www.coursera.org/learn/machine-learning/supplement/bjjZW/normal-equation

Pretty much compute 



Normal Equation Nonivertibility

In some cases you might have an issue with having singular marixes, which are noninvertable.

Solution: 1 Octave Pinv function solves this automatically 2 Get rid of dependent columns, they dont provide anything useful anyway. 3 Regularization – will talk about later.

## Additional Topics From Stanford own Course Materials

Stanford pdf 1

Probabalistic Interpretation

Why should we use linear regression and the least squares cost function to best face our linear regression problem? It can be shown that probabalistically these options should give us the best model. Basically we want to maximize the density function (1) according to our x|y model, maximize the likelihood of it working according to the given data. E.g we want to DO something called the maximum likelihood estimation.

1. WE make an assumption that error we get from our hypothesis is under normal dist.
2. WE try to find what parameters would with highest likelihood give me this data. So what thetas would give me the data from the training set, e.g. y|x. Think about this in a opposite way. I have Theta, I also have x. I want to get the probability of y happening. I want to choose a theta, which would make the probability of y happening with x the highest. *Why?* I want to do this because we are estimating the model, where x and y are included. I want to find a theta, that would make the agreement between the model and the data work.

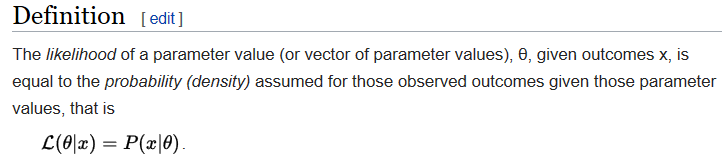
This is called maximum likelihood estimation.

The previous explanation also fits with the definition of likelihood.

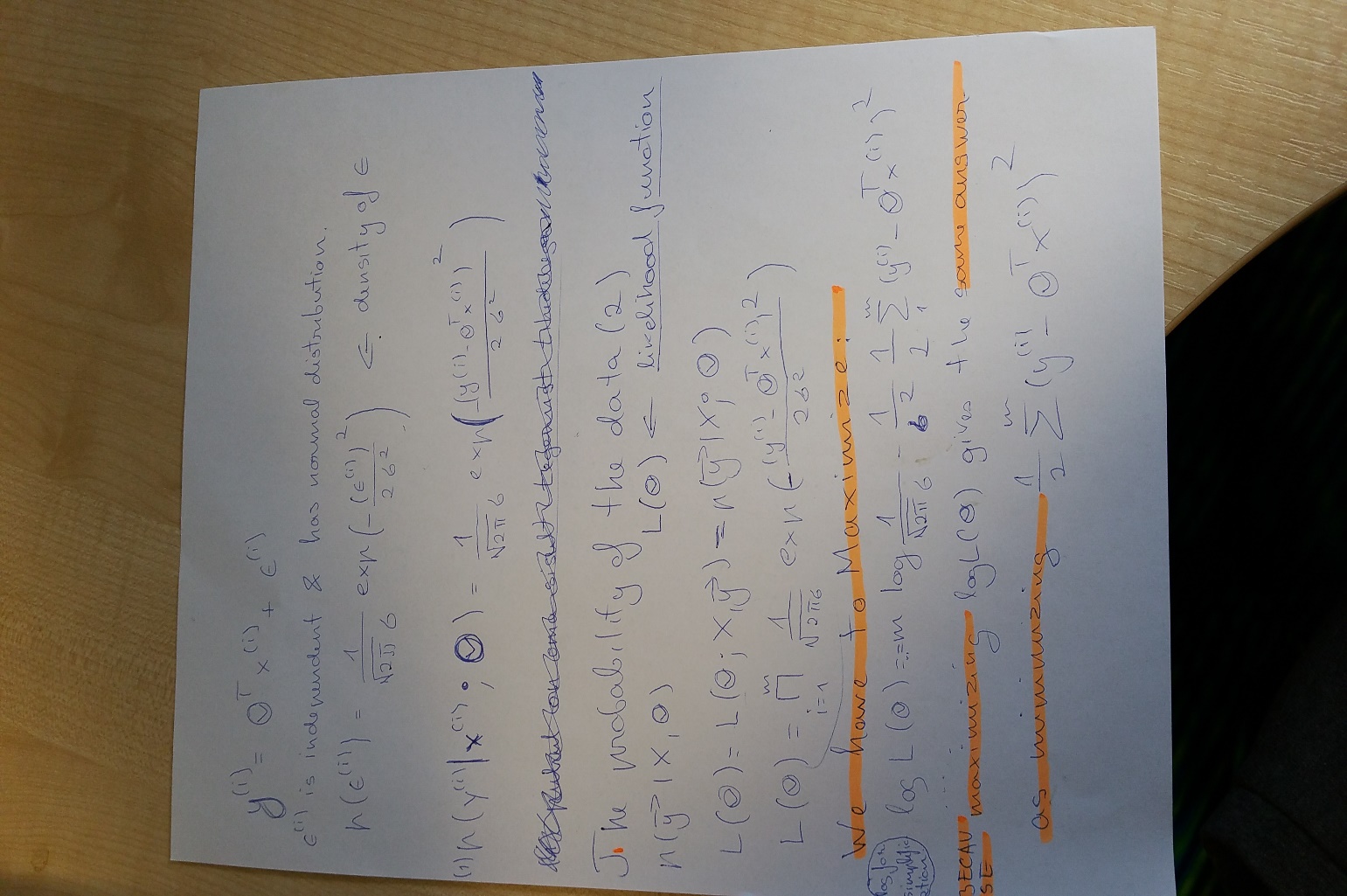
In here:

As a likelihood we can describe it from the perspective that if we have a model of random variable x, what is the likelihood of parameter theta

As a probability we can describe it from the perspective that if we have parameter theta, then what is the chance of having x



Maximum likelihood selects the set of values of the model parameters that maximizes the [likelihood function](https://en.wikipedia.org/wiki/Likelihood_function). Intuitively, this maximizes the **"agreement"** of the selected model with the observed data, and for discrete random variables it indeed maximizes the probability of the observed data under the resulting distribution.



(You can get a better view of this picture by saving it and then viewing it separately )

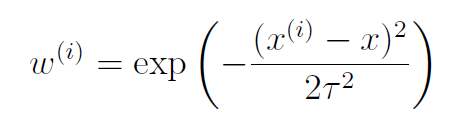
<https://www.youtube.com/watch?v=I_dhPETvll8> Ben Lampbert

Maximum Likelihood estimation - an introduction part 1

In [statistics](https://en.wikipedia.org/wiki/Statistics), a **likelihood function** (often simply the **likelihood**) is a function of the [parameters](https://en.wikipedia.org/wiki/Statistical_parameter) of a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) given data. <https://en.wikipedia.org/wiki/Likelihood_function>

### Locally weighted linear regression

Pretty much instead of calculating the whole function, we focus on calculating the function at specific values. This is best explained in stanf pdf # 1 Overally it uses weights to maximize the effect of training data close to x and minimizes effect of training data further away.



# Week 3

## Classification and Representation

### Classification

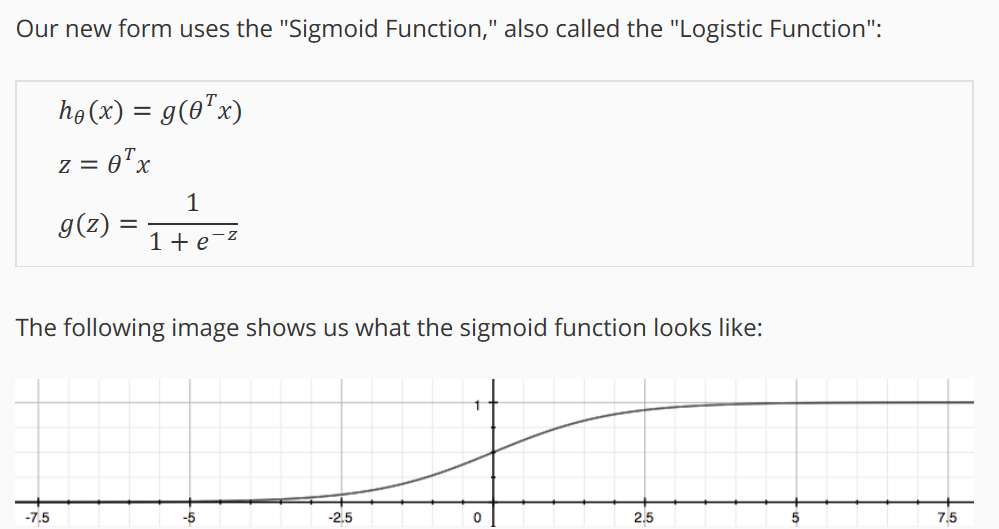
<https://www.coursera.org/learn/machine-learning/supplement/fDCQp/classification>

Classification – When output is discrete.

Why can’t we do classification with linear regression ? Because. The decision boundary will still be randomly chosen. Finding the output range and distribution is hard.

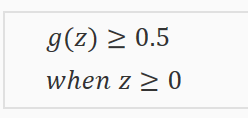
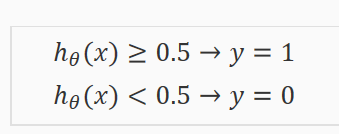
### Hypothesis Representation

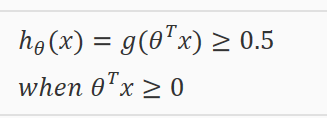
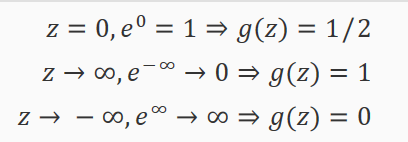
https://www.coursera.org/learn/machine-learning/supplement/AqSH6/hypothesis-representation



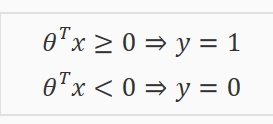
### Decision Boundary

<https://www.coursera.org/learn/machine-learning/supplement/N8qsm/decision-boundary>





Why then cant we just use the function below. A: This would give a bad cost function.

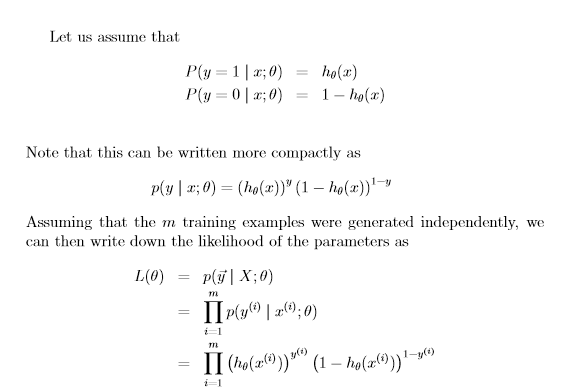


## Logistic Regression Model

### Cost function

Derivation from stanford pdf 1. x|y

We again assume some model for

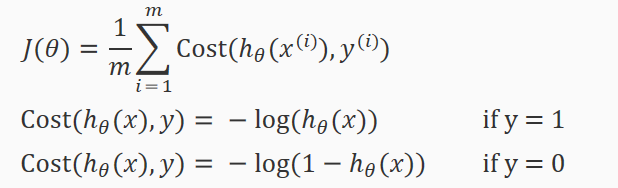


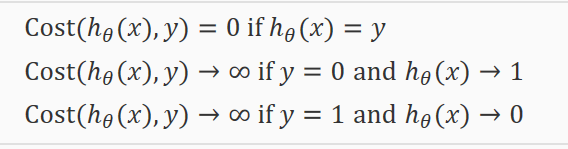
And then we maximize it !

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

We cannot use the same cost function that we use for linear regression because the Logistic Function will cause the cost function parameterized with theta to be wavy. , causing many local optima. In other words, it will not be a convex function.

Instead, our cost function for logistic regression looks like: Note that writing the cost function in this way guarantees that J(θ) is convex for logistic regression.





### Simplified Cost function and Gradient Descent

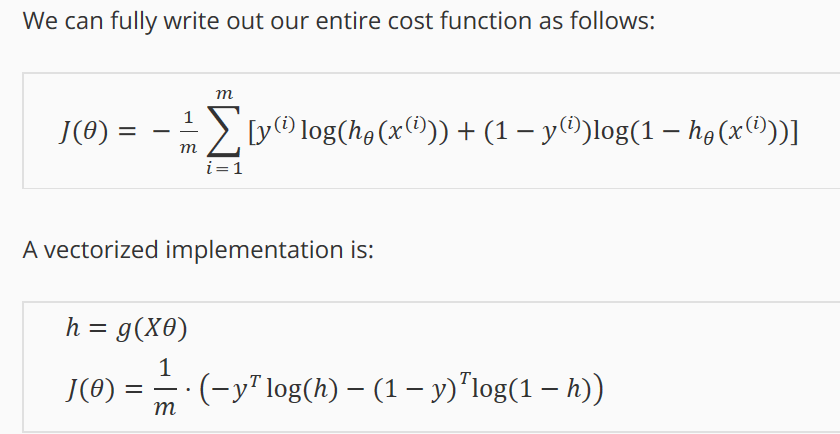
**Can we use normal equation on logistic regression: NO! The function does no behave that way.**

<http://stackoverflow.com/questions/37997253/can-we-use-normal-equation-for-logistic-regression>

**Answer:** So why it worked so well for linear regression? Because once you compute your derivatives you will notice, that resulting problem is set of **linear** equations, m equations with m variables, which we know can be directly solved through matrix inversions (and other techniques). When you differentiate logistic regression cost, resulting problem is no longer linear... it is convex (thus global optimum), but not linear, and consequently - current mathematics does not provide us with tools strong enough to find the optimum in closed form solution.

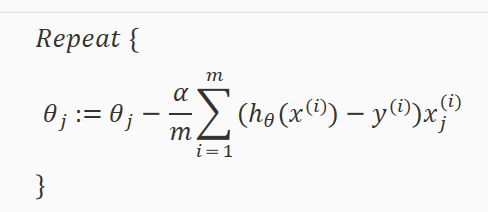
**Simplified Cost function**

Screen Clipping



**Gradient descent part**

Pretty much the same shape than in linear regression, but instead we use different hypothesis function.



Derivative(Log(x)) = 1/x

### Advanced Optimization

https://www.coursera.org/learn/machine-learning/supplement/cmjIc/advanced-optimization

**These algorithms can be both applied on linear and logistic regression.**

**Conjugate gradient**

**BFGS**

**L-BFGS**

Advantages:

No need to manually pick a – has „kind“ of inner loop that determines THE STEP a bit better.

Often faster than gradient descent

Disadvantages

More complex

I shouldnt implement these thing myself, because im not an expert of numerical computing.

Think of this as minor thing as finding matrix inverse. It calculates the minimum of the function. They also say, this is not important, so I trust them.

## Multiclass classification

https://www.coursera.org/learn/machine-learning/supplement/HuE6M/multiclass-classification-one-vs-all

1 vs all classification.

Divide your problem into n+1 (n starts at 0) classification problems. Do logistic regression for all.

## Solving The Problem of Overfitting.

### The Problem OF oVerfitting

<https://www.coursera.org/learn/machine-learning/supplement/VTe37/the-problem-of-overfitting>

Ways to Solve:

* Plotting wont always work
* Reduce number of features
  + Manually select which features to keep
  + Use algorithms to choose which features to keep
* **Regularization** – This is what I am going to study.

### Cost function

<https://www.coursera.org/learn/machine-learning/supplement/1tJlY/cost-function>

I had a question here: What is the difference between **Regularization** and **Feature Scaling:**

Feature scaling scales training data, but does not mean, that the effect of certain parameters will be small. It is rather used for gradient descent to converge faster.

Reqularization reduces the effect of Thetas and therefore the effect of some parameters in training data.

**Pretty much whe**n doing regularization we force the Cost function to optimize in a way, it would keep small thetas, in normalization, this does not happen .

**Why we dont to regularisation on theta 0 ?**

These will be the same for every input anyway, so they wont give some parameters an extra boosts or so ..

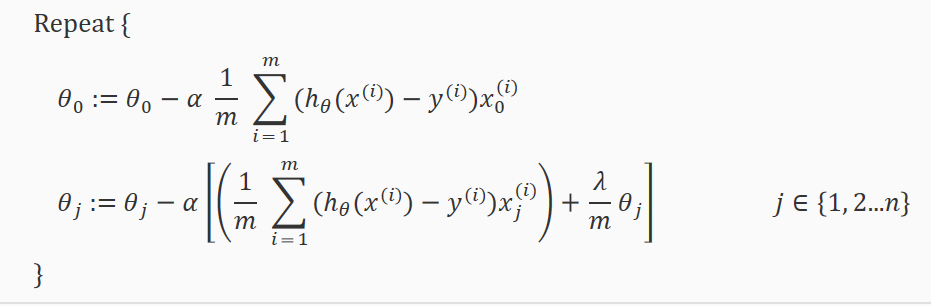
**Remark:**

Based on the Assignement, Regularization is really an awesome thing and avoids overfittign very well.

### Regularized Linear Regression

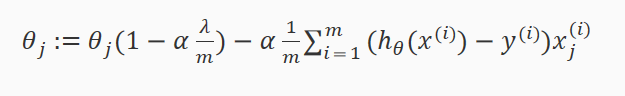
<https://www.coursera.org/learn/machine-learning/supplement/pKAsc/regularized-linear-regression>

The gradient descent still comes from partial derivative .



Add – on every iteration, theta j is being shrinked. It has to be shrinked, otherwise, the function can’t find its minimum.

We can algebraically modify the gradient descent function:



As [] is always slightly smaller than 1 , like 0.99 or something. Then on every iteration Theta is being shrinked. Which is the purpose of regularization.

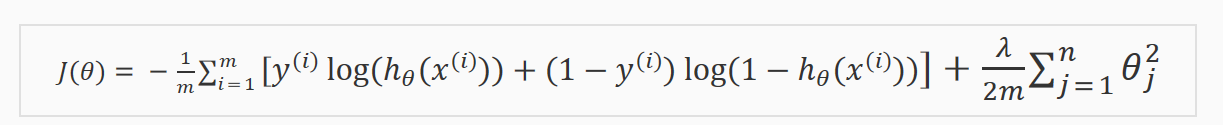
**Normal equation –** The principle still remains the same. We use the modified Cost function. Get its partial derivative, and set it equal to 0.

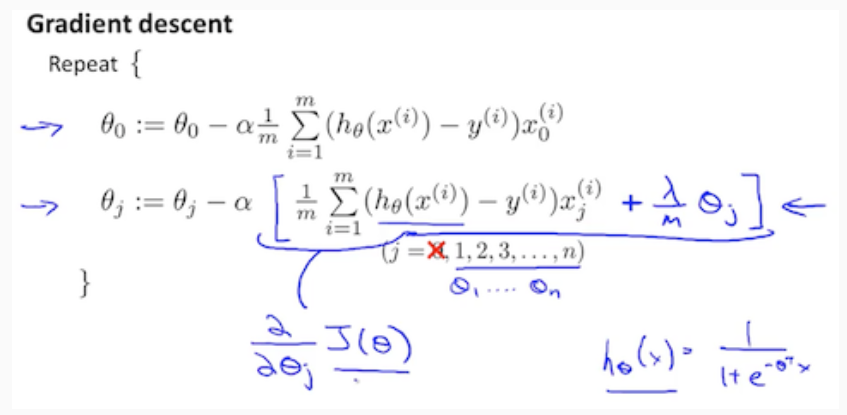
Adding regularization to normal equation also takes care of all the noninvertibility issues.

### Regularized Logistic regression

https://www.coursera.org/learn/machine-learning/supplement/v51eg/regularized-logistic-regression

Pretty much the same logic applies to this logic, than to all the other logics.





We also introduced using Advanced optimization usage.

### Another algorithm for maximizing Cost functin

STANFORD PDF 1 END

Newtons method.

It is known to converge with fewer iterations than gradient descent. At the same time, usez also mor computation powe per one iteration. As it has to calculate the hessian.

### Generalized Linear Models

STANFORD PDF 1 END

Pretty much both Linear Regression and Logistic regression can be generalized as a generalized linear model. This allows to use an extended family of distributions in approximating the error that our hypothesi functions obtain, like poissson….

This means for regression analysis we can use more models ….

I havent fully done the theory part here…..

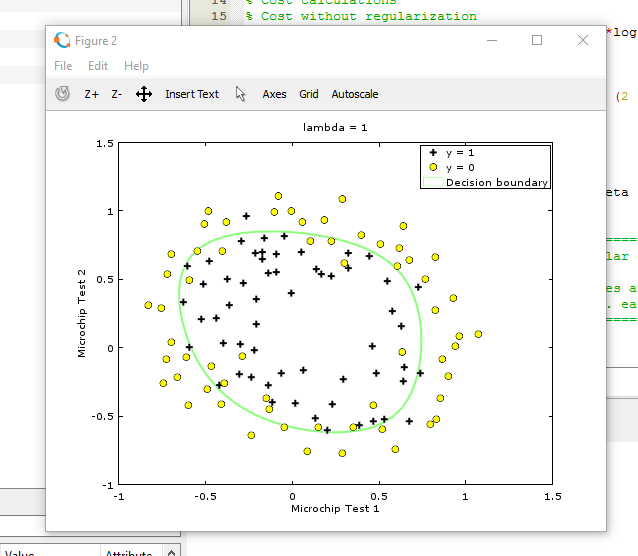
## Remarks from solving the Home Assignement

Decision boundary in Logistic regression is a wonderful thing. It allows us to pretty much derive Any type of shapes for our boundary from the function.

Pretty much the situation is like that:

1. we have our hypothesis function h(theta\*X)
2. we define a boundary h =< 0
3. The boundary is at the same time an equation, which does not have the limits of a function and therefore can form random shapes AND adapt **to any data.**

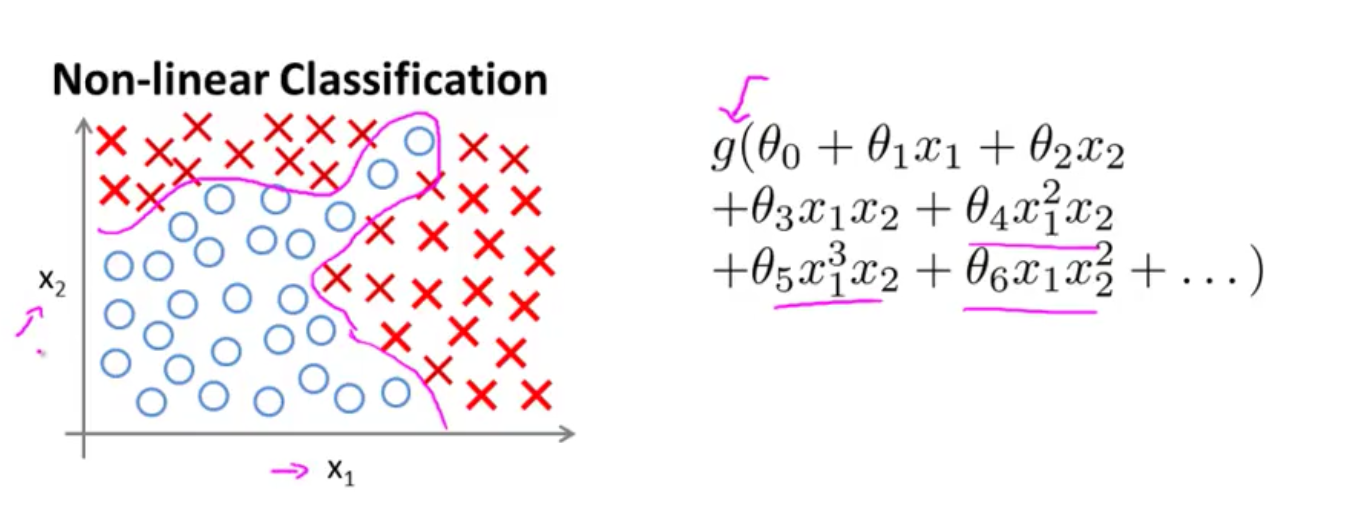
Example



# Week 4

## Motivations

### Non Linear Hypothesis

From homework and previous examples it was seen, that when data becomes more complex, it is increasingly difficult to do polynomial mapping, as the number of features becomes exremely large.

For example to have all the parameters in second power form we need n\*\*2/2 features. If n = 100, we will have 5000 parameters. Neural networks deal with this issue, because they allow us to form increasingly complex functions, but at the same time the number of features does not grow extremely large. Pretty much only features that matter are chosen based on cost function based analysis“

### Neurons and the brain

https://www.coursera.org/learn/machine-learning/lecture/OAOhO/non-linear-hypotheses

Neural networks is an old thing. The reason it has become popular again lies behing **us having increasing capapilities regarding technology. Thus I should do well on cluster computing.**

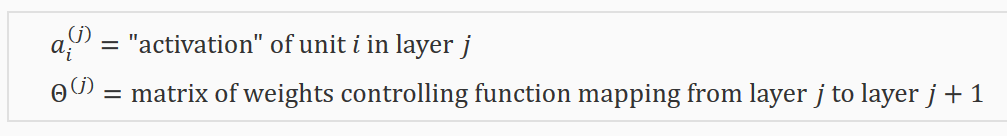
## Neural Networks

### Model Representation I

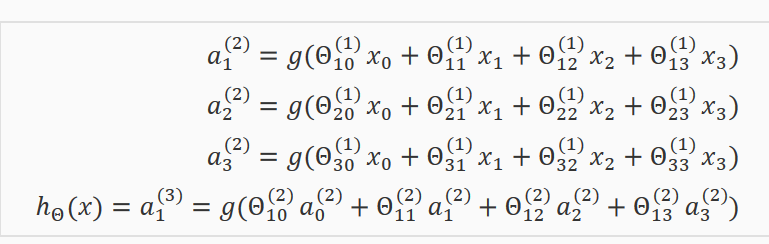
**Neuron description:**

https://www.coursera.org/learn/machine-learning/supplement/Bln5m/model-representation-i

* Inputs dendrites
* Outputs axons
* Activation function = sigmoid function



The values for each a activation nodes is obtained as follows: (a0 = 1)

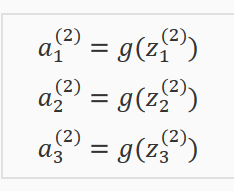


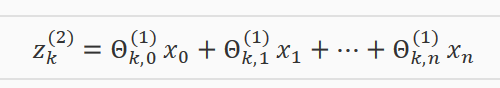
**MATRIX DIMENSIONS**

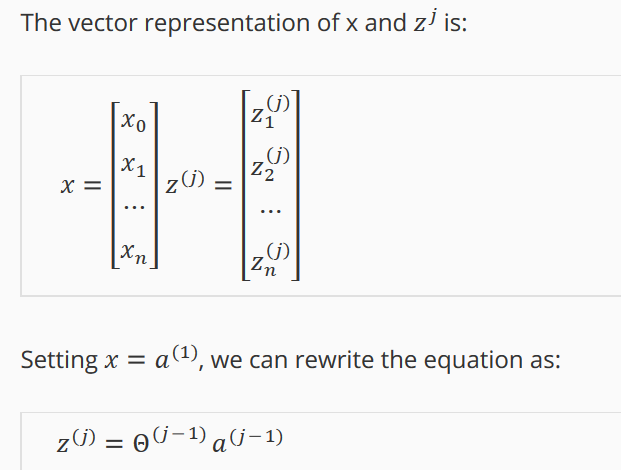
### Screen Clipping

### Model Representation II

<https://www.coursera.org/learn/machine-learning/supplement/YlEVx/model-representation-ii>







Now we get:

aj=g(z(j))

Where our function g can be applied element-wise to our vector zj.

The reason why Neural networks are better than just choosing all the features for polyniomial mapping, is the fact that in Neural networks, the features are chosen through some analytic measure instead of random. So they can be balanced in regards to their importance <- My understanding.

## Applications

### Examples and Intuitions I

<https://www.coursera.org/learn/machine-learning/supplement/kivO9/examples-and-intuitions-i>

A simple example of **predicting x1 AND x2**

We do this through the hypothesis function. We have a 2 layer network. 1 hidden layer inside.

Firs we will only have our input: vector of x. We multiply it with our theta matrix , which has sj + 1 columns and s(j+1) rows. We multiply the x-s with theta matrix j to get the input for the hidden layer. The output will be between 0 and 1. When we design our input thetas in a way that g(z) will be only if both inputs are 1-s

(1-s because our input is discrete {0,1})

### Examples and Intuitions II

https://www.coursera.org/learn/machine-learning/supplement/5iqtV/examples-and-intuitions-ii

Here we form a multiple layer network from multiple logical operations :{OR, AND, XOR} to get XNOR.

This show us how rather easy an intuitive rules can become extremely complicated when applied as a multilayer network

### Multiclass Classification

https://www.coursera.org/learn/machine-learning/supplement/xSUml/multiclass-classification

Pretty much instead of having output for 1 variable , we will have output for multiple variables of the network and based on what achieves the highest value, we do the classification.

# Week 5

IT IS IMPORTANT TO KNOW, THAT THIS KIND OF COST FUNCTION IS NON CONVEX , BECAUSE IT HAS THETA MULTIPLIED MANY TIMES WITH EACH OTHRE

11.05

Well, this topic took me quite a lot of time to get through only a part of it. However this is only because I really wanted to understand how a multilayer network actually improves itself over gradient descent.

**SIMPLE CALCULATION WITH 1 TRAINING EXAMPLE**

1. In laymans terms ( Which may or may not improve the understanding) What a neural network does:
2. Lets assume we have L layers.
3. Lets assume we have only one training example.
4. Neural Network initializes first of all L-1 theta matrixes to some certain value. Usually 0.
5. Then it runs the training example through the neural network
6. Then it calculates the Error
7. Now on every level 2….L we calculate the de/dz – derivative according to z.
8. When we have the De/Dz for all levels , its easy ( say thanks to the derivative rule called „chain rule“) to calculate the derivatives according to each theta separately.
9. We now have the derivatives for all the separate thetas. Its easy to calculate the the gradient for all of them, which pretty much = theta – De/Dtheta for all theta
10. We update the thetas and go back to step 5. If the improvement is less than our specified value, we stop.

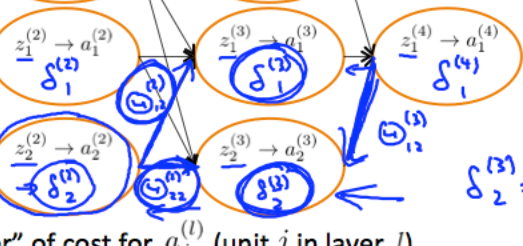
**EXTENDING THIS TO MULTIPLE TRAINING EXAMPLES**

Multiple training examples is pretty much equal to the sum of steps 5 for each particular training example divided by the total number of training examples. It does not have a big effect for the overall calculation. Because 1. the divisor is a constant and 2. for each sum we calculate the derivative separately. Otherwise the shape stays the same. The derivative De/Dtheta for allt theta is just gonna be sum of derivatives for each training example.

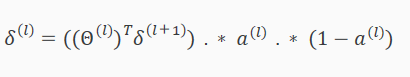
**DERIVATIVE CALCULATION SPECIFICS.**

gonna do this procedure tomorrow, because I dont have my notebook.

**CLAIM in backpropagation we follow the blue lines**

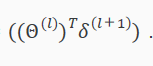
****

**The overall formula for DE/Dz=**

****

**Where**

**Screen Clipping=Da/Dz**

**= dE/da**

**So** ALLTOGETHER DE/DZ **= Da/Dz \* De/Da**

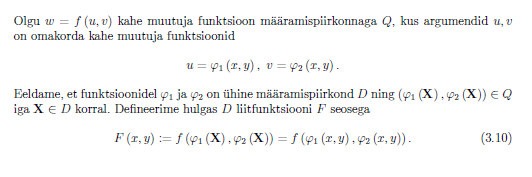
**,** *this was a bit more tricy part for me to understand, because it involves calculating the derivative for a variable, which influences the end function through multiple*

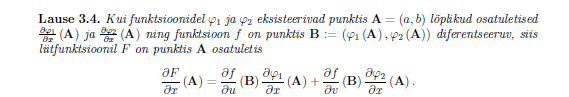
*functions in the chain ( look for the picture above to understand****)***

**In the last term we get from the original theta the column which was used together with the particular a to calculate all the different zL+1-s**

*In the last example a2 in level 2 influences the final solution and thus the error function through z1 and z2 which are located at level 3. Because of that, when we take the derivative from E according to the change of a2 in level two, we have to take z2 and z3 in level 3 into account****.***

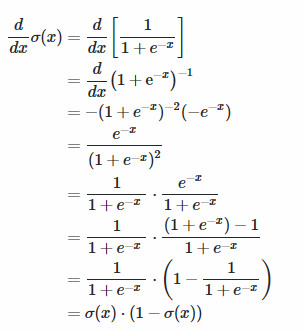
**This is no problem, because there is simple formula to take a derivative of a function with shape g(f1(x),f2(x)). Dg/Dx = Dg/Df1\*Df1/Dx+ Dg/Df2\*Df2/Dx**

****

****

**EXPLANATION OF LEVEL L** DE/Dz calculation. This is different from the rest of the cases.

**Da/dZ**

ss

**DE/Da =-y/a+(1-y)/(1-a)**

DE/Dz=DE/Da\*Da/Dz=a-y