Machine Learning Note (Matlab/Octave, Andrew Ng, Stanford U.)

1. ML: Introduction

Definition:

1. the field of study that gives computers the ability to learn without being explicitly programmed
2. A computer program P is said to learn from Experience E with respect to some class of tasks T and performance measure P, if its performance at task in T, as measured by P, improves with experience E

* Supervised Learning
* Regression – predict result in continuous output
* Classification – predict in discrete output/categories
* Unsupervised Learning – no idea on how result will look like
* Clustering
* Non-Clustering

1. Linear regression with one variable
2. Hypothesis function:
3. Random guess and
4. Cost function (mean of squared error, ½ is only for convenience):
5. Gradient Descent

Iteration:

* Measure the accuracy
* Automatically improve

(for j = 0, 1)

for linear regression:

=

=

=

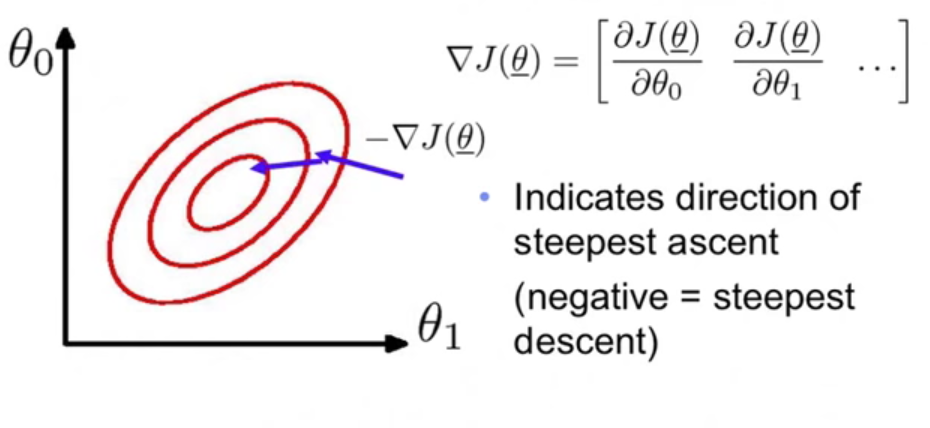
=

=

=

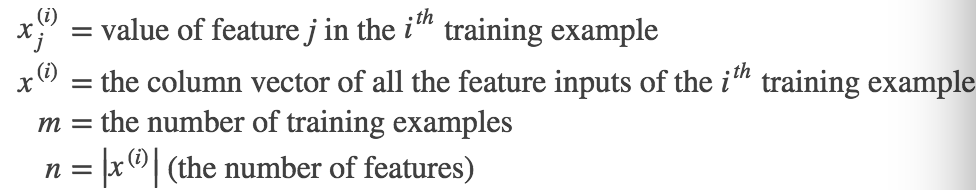
thus, repeat till convergence: {

}



Finally, get the for the final linear hypothesis equation, it will be with the lowest error for existing dataset.

1. Linear regression with multiple variables



1. Hypothesis function:

Or vectorized:

or, if X stored with features varied by columns: X is m x (n+1), theta will be (n+1) x 1

a m x 1 matrix of predictions in all X cases

1. Random guess
2. Cost function (mean of squared error, ½ is only for convenience):

or vectorized:

**- a number of total cost**

1. Gradient Descent

Iteration:

* Measure the accuracy
* Automatically improve

(for j = 0, 1,2,3…n)

for linear regression:

=

=

thus, repeat till convergence: {

from 0 to n

}

or vectorized:

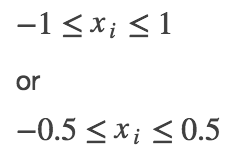
repeat: {

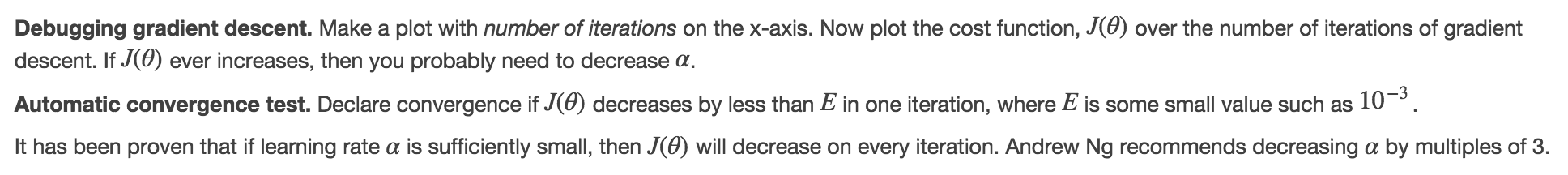
}

thus, gradient descent could be written as:

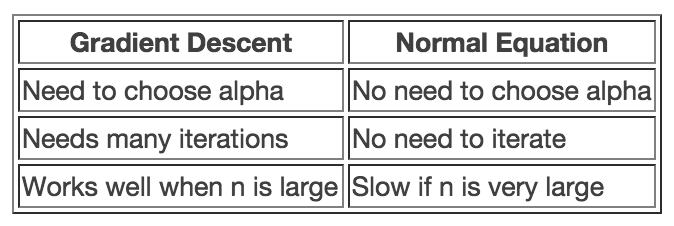
(by dimension, 1 X: m\*(n+1) ( )

In practice, we use:





If without iteration, we use Normal Equation



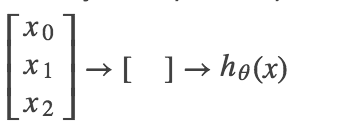
1. Logistic Regression

…

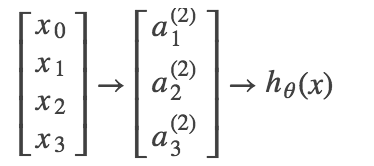
1. Neuron Networks
2. Model representation

At a very simple level, neurons are basically computational units that take input (**dendrites**) as electrical input (called "spikes") that are channeled to outputs (**axons**).

1 layer:



In neural networks however we sometimes call it a sigmoid (logistic) **activation** function.





or

X – 4x1 vertical a – 3x1

Steps of NN:

S1 – Choose arch:

1. Number of input units = dimension of features x(i)
2. Number of output units = number of classes
3. Number of unites per layer (the more, the better when computation cost allows)
4. Usually 1 hidden layer, If 1+, then same number in each layer

S2 – Training NN:

1. Randomly initialize weights
2. Implement the forward propagation to get
3. Implement the cost function based on theta above

* K: number of classes, for y(k)
* j has different meanings in 2 terms.

1. Implement back propagation to compute partial derivatives

* Goal: minimize

1. Use gradient descent or a built-in optimization function to minimize the cost function with the weights in theta

Iterate to loop on every training example:

for i = 1:m

Perform forward propagation and back propagation using example x(i), y(i)

(Get activations a(l) and delta terms d(l) for l = 2, ….L)