**Introduction – Supervise vs Unsupervised Learning**

**Machine Learning** – The Computer learn from experience E respect to task T and some performance measure P, P improves with E.

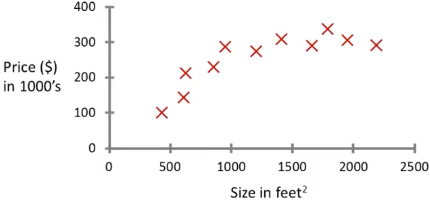
Example: Filter Spam Email

1. Classifying email as spam or not (T)
2. Watch you label emails as spam or not (E)
3. The number of emails correctly classified as spam / not spam (P)

**Supervised learning** – produce right answers

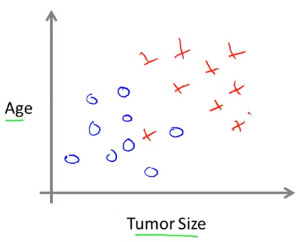
**Regression** – predict continuous valued output

Example: predict housing price (regression problem)



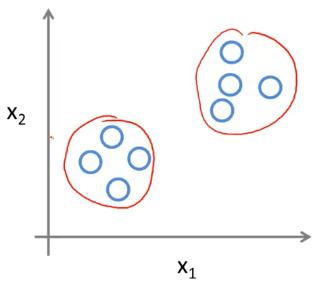
**Classification** – given labeled data, classifies input data in one category or another

Example: Predict if tumor is harmful (Classification Problem)



**Unsupervised learning** – Given unlabeled data, uninstructed, find structure from the data and approach problems with little or no idea what our results should look like.

**Clustering Algorithm** – Break data into clusters. Ex(google news, cluster organisms into its closely related by comparing their genes, Separating voices from an audio)



**Model and Cost Function**

**Training set** – Data input to a supervised learning algorithm

**Notation:**

m = #traning examples

x’s = “input” variable / features

y’s “output” variable / “target” variable

Example: Predicting housing price

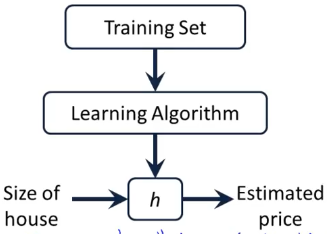
x’s = Size in feet2

y’s = Price

m = number of inputs

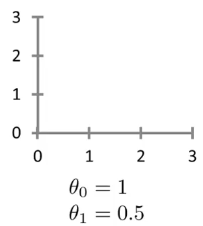
(x(i),y(i)) = ith training example

**Model**



**h (hypothesis)** - maps from x’s to y’s

Ex. Univariate Linear Regression - hθ(x) = θ0 + θ1x; θ0 and θ1 are called Parameters

h(x) = 0.5x + 1

**Idea:**

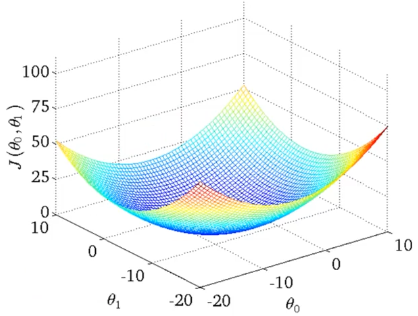
Chose parameters (θ0 and θ1) | hθ(x) is close to y for training examples (x, y)

**Why square the difference?** Squared error cost function is most commonly used for regression problems.

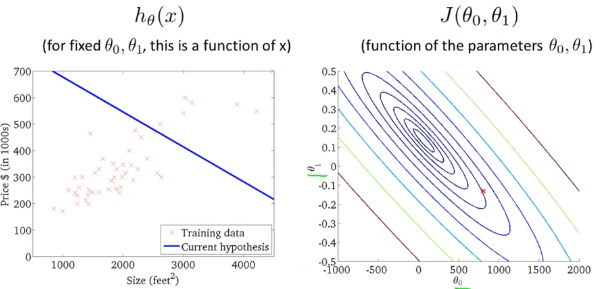
|  |  |
| --- | --- |
| Assume θ0 is constant 0 | |
| Function: hθ(x) | Function: J(θ1) |
|  |  |
| Result of J()    The vertex of J() is the optimum assignment to the parameters | |

With 2 parameters, it will look like this:

3D version



Contour Figure



**Gradient Descent (Minimize Cost Function of Univariant Linear Regression)**

**Idea:**

Start with some θ0 + θ1

Keep changing θ0 + θ1 to reduce until end up with minimum

Note: Different initialization might result in different local optimal minimum

**Gradient Decent Algorithm Formula: Repeat until convergence**

**Simultaneous Update**

*an assignment notation. Ex. a:= a+1 means let a+1 overwrite a*

*an assertion notation. Ex. a = b means a has same value as b, a = a+1 is never correct.*

*learning rate, determine the size of each changing parameters*

*derivative term. slope of tangent line on the point:*

1. Derivative term will always reach 0
2. always 🡪 0
3. As move toward local minimum,

If tangent line start with positive slope, the derivative term decrease.

Else, if tangent line start with negative slope, derivative term increase

**Why Derivative term exist?**

So that the size of the steps do not need to be manually changed, instead it will decrease automatically by derivative term as approaching local minimum

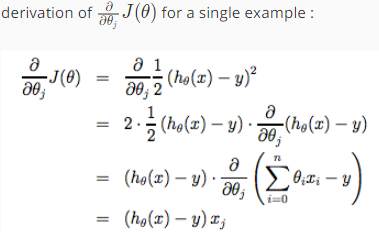
|  |  |
| --- | --- |
| If learning rate is too small, infinite steps may require to reach minimum |  |
| If learning rate is too large, it will overshoot the minimum and fail to converge |  |
| As , decreases, meaning that the steps will decrease as approaches local minimum |  |
| Example:  If is right on local minimum, the value of the parameters remain unchanged, since and | |

**Apply Gradient Decent Algorithm to Linear Regression Model**

**Linear Regression Model:**

**Gradient Decent Algorithm**

**Combined**

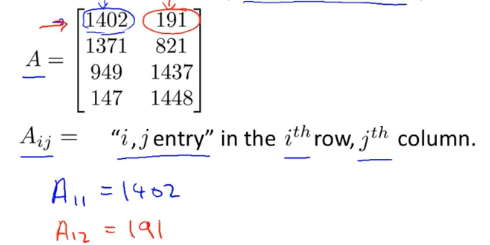


**Repeat until Convergence: Combined**

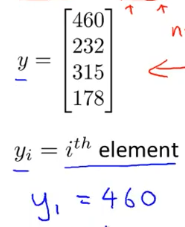
**“Batch” Gradient Descent** - Each step of gradient descent uses all the training data.

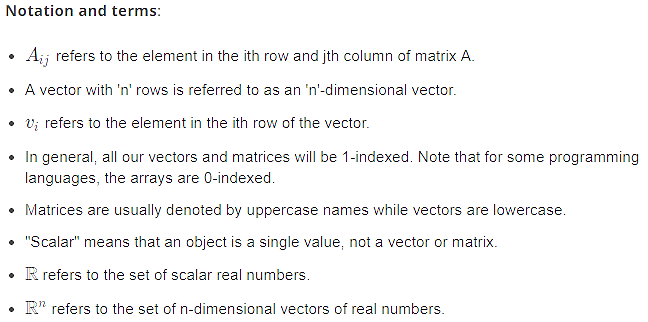
**Matrices (2D Array) and Vector**

**Matrix Representation (refer by capital):**



**Vector (refer by lower case)** – an n\*1 matrix





**Manipulation on Matrix:**

|  |
| --- |
| **Matrix Addition** |
|  |
| **Scalar Multiplication** |
|  |
| **Matrix-Vector Multiplication** |
| **General** |
|  |
| **Example** |
|  |
| **Detail** |
|  |
| **Example Housing Price**   |  |  | | --- | --- | |  |  | |  |  | |  | | | **Prediction = Data Matrix \* Parameters** | | |

**Matrix-Matrix Multiplication**

|  |
| --- |
| **Example** |
|  |
| **Detail** |
|  |
| **Example Housing Price** |
|  |
| Pack all computations into 1 matrix |
| First column of multiplied matrix correspond to first hypotheses, and so on |
| Different programing languages may have optimized algorithm to multiply matixs |

**Properties of Multiplication of Matrices**

|  |
| --- |
| **Property #1: Commutativity** |
| The order of multiplication is **not commutative**  **Matrix A and B; A \* B != B \* A** |
| **Example** |
|  |
| **Proofs** |
|  |
| **Property #2: Associativity** |
| The order of multiplication is **not associative**  **Matrix A, B, C; (A \* B) \* C != A \* (B \* C)** |
| **Property #3: Identity Matrix** |
| **Identity Matrix (denoted I):**  Top right and bottom left are summetric |
| For any matrix A;  **A \* I = I \* A = A ( I = Identity Matrix)** |
| **Proofs** |
|  |

**Matrix Inverse and Transpose**

|  |
| --- |
| **Facts** |
| Real number’s Property: 1 = “Identity”  Inverse of real numbers: 3 \* (3-1) = 1, number \* its inverse = 1  0 does not have an inverse |
| **Matrix inverse** |
| If A is a square matrix, and it has an inverse (if A does not contain zero):  A \* (A-1) = (A-1) \* A = I (Identitty Matrix) |
| **Example** |
|  |
| **Terminology** |
| **Singular / Degenerate** – Matrix who doesn’t have a matrix |
|  |
| **Matrix Transpose** |
| **Example** |
| **Definition**  Let A be m\*n matrix, B = AT  B is an n\*m matrix and |