Quiz

You are running a company and you want to develop learning algorithms to address each of the following two problems:

Problem 1: You have a large inventory of identical items. You want to predict how many of these items will sell over the next three months.

Problem 2: You would like software to examine individual customer accounts and for each account decide if it has been hacked/compromised.

Should you treat these as classification or as regression problems?

a) Treat both as classification problems
b) Treat problem 1 as classification problem, problem 2 as regression problem.
c) Treat problem 1 as regression problem, problem 2 as classification problem.
d) Treat both as regression problems

Answer: c)

Quiz

Of the following examples, which would you address using an <u>unsupervised</u> learning algorithm? (Check all that apply.)

- [ ] Given email labeled as spam/not spam, learn a spam filter.
- [x] Given a set of news articles found on the web, group them into set of articles about the same story.
- [x] Given a database of customer data, automatically discover market segments and group customers into different market segments.
- [ ] Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not.

The Supervised Learning Problem

- Outcome measurement Y (also called as dependent variable, response, target)
- Vector of p predictor measurements X (also called inputs, regressors, covariates, features, independent variables).
- In the regression problem, Y is quantitative (e.g. price, blood pressure)
- In the classification problem, Y takes values in a finite, unordered set (survived/died, digit 0-9, cancer class)
- We have training data $(x_1,y_1), \ldots, (x_N,y_N)$. These are observations (examples, instances) of these measurements.
- On the basis of training data we would like to:
  - Accurately predict unseen test cases
  - Understand which input affect the outcome
  - Asses the quality of our predictions and inferences.

Unsupervised Learning

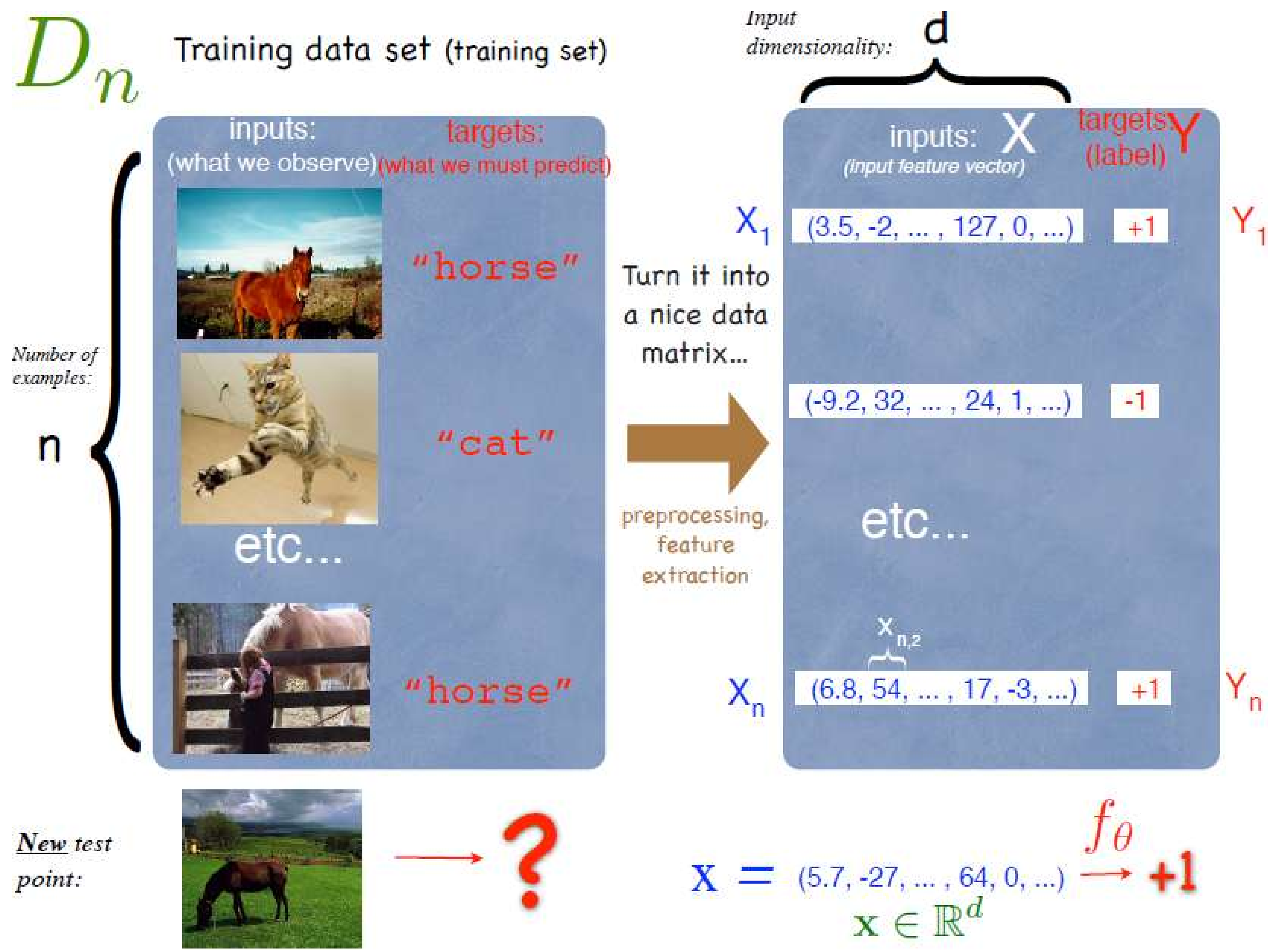
- No outcome variable, just a set of predictors (features) measured on a set of examples.
- Objective is more fuzzy – find group of samples that behave similarly, find features that behave similarly, find linear combination of features with the most variation.
- Difficult to know how well you are doing.
- Can be useful as a pre-processing step for supervised learning.

The key ingredient of machine learning is...

DATA

- Collected from nature or industrial processes.
  - Comes stored in many forms (and formats...):
    - structured, unstructured, occasionally clean, usually messy, ...
  - In ML we like to view data as a **list of examples** (or we'll turn it into one)
    - ideally ***many examples of the same nature.***
    - preferably with each example a *vector of numbers* (or we'll first turn it into one!)

Understanding the ML Task



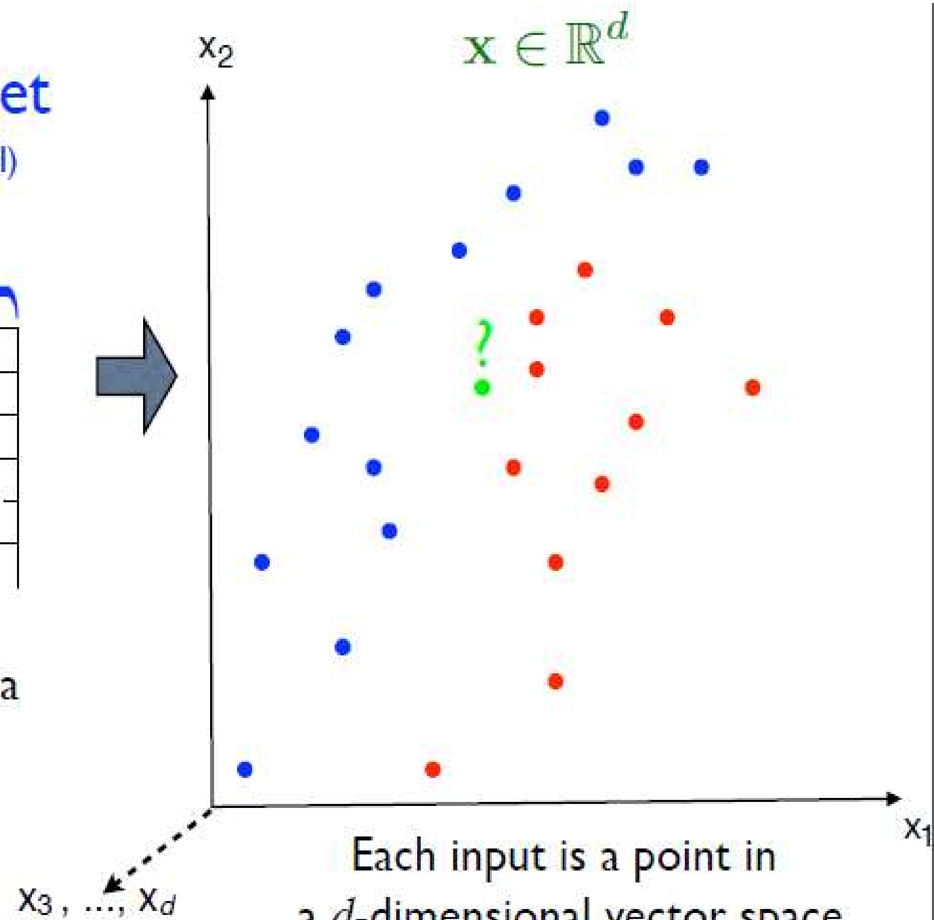
Dataset imagined as a point cloud in a high-dimensional vector space

input $x \in \mathbb{R}^d$ target (label) y

n examples

x_1	x_2	x_3	x_4	x_5	
0.32	-0.27	+1	0	0.82	1
-0.12	0.42	-1	1	0.22	0
0.06	0.35	-1	1	-0.37	1
0.91	-0.72	+1	0	-0.63	1
...	...	...	...	...	...

Each example (row) is now a d+1-dimensional vector



Each input is a point in a d-dimensional vector space

Machine Learning Tasks (Problem Types)

Supervised learning = predict a target y from input x
(and *semi*-supervised learning)

- y represents a category or "class"
  - ➠ classification — binary : $y \in \{-1, +1\}$ or $y \in \{0, 1\}$; multiclass : $y \in \{1, m\}$ or $y \in \{0, m-1\}$
- y is a real-value number
  - ➠ regression — $y \in \mathbb{R}$ or $y \in \mathbb{R}^m$

Predictive models

Unsupervised learning: no explicit prediciton target y

- model the probability distribution of x
  - ➠ density estimation
- discover underlying structure in data
  - ➠ clustering
  - ➠ dimensionality reduction
  - ➠ (unsupervised) representation learning

Descriptive modeling

Reinforcement learning: taking good sequential decisions to maximize a reward in an environment influenced by your decisions.

Understanding the ML Model

Supervised task:

predict y from x

input $\mathbf{x} \in \mathbb{R}^d$

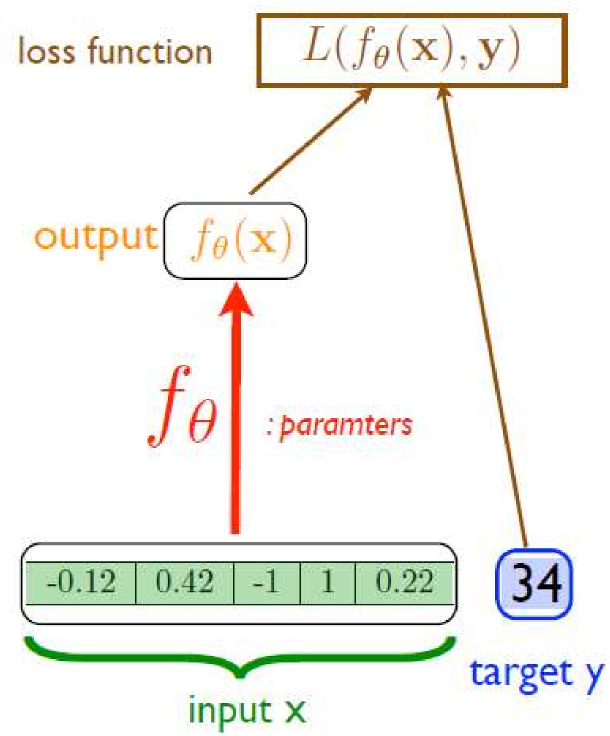
target (label) y

n examples

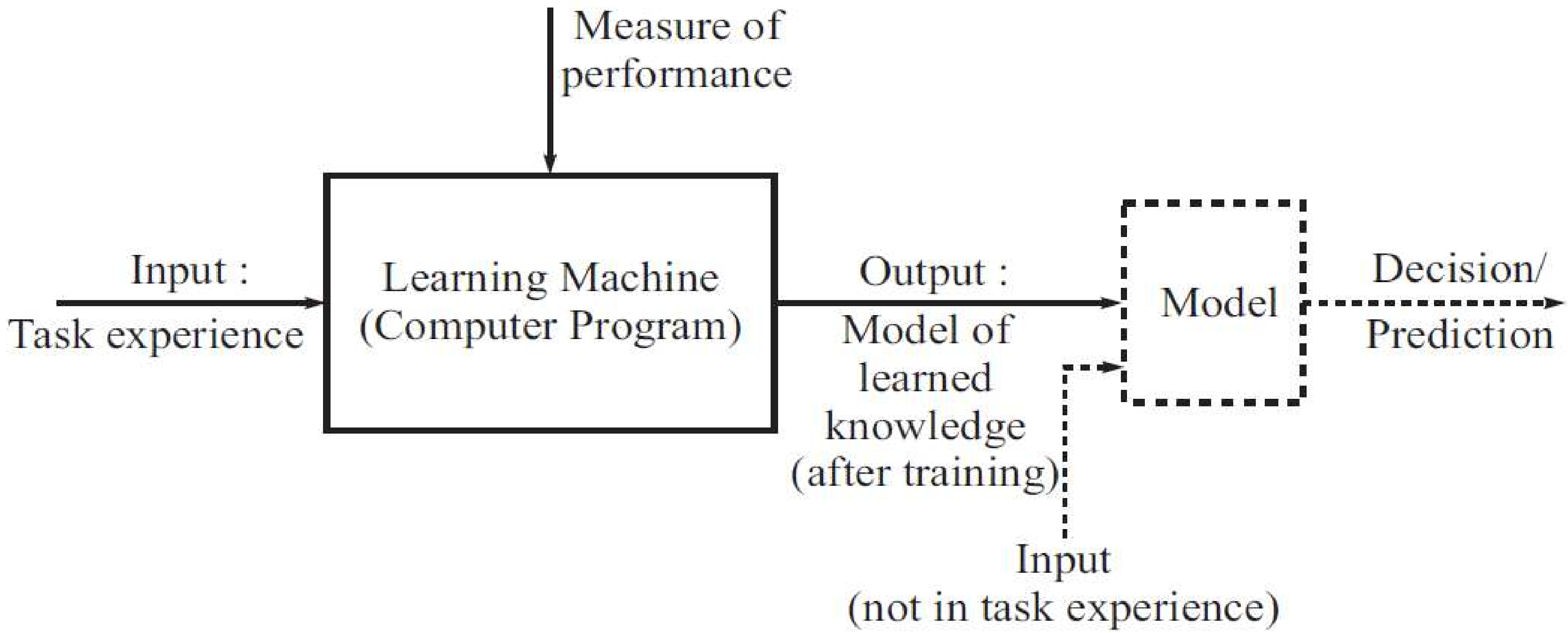
x_1	x_2	x_3	x_4	x_5	t
0.32	-0.27	+1	0	0.82	113
-0.12	0.42	-1	1	0.22	34
0.06	0.35	-1	1	-0.37	56
0.91	-0.72	+1	0	-0.63	77
…	…	…	…	…	…

Training set D_n

Learn a function f_θ that will minimize prediciton errors as measured by cost (loss) L.



Block Diagram of Learning Machine



ML Problem Phases

The following are the various steps involved in problem solving using ML:

- Define your task
- Collect Data
- Pre-processing of Data
- Dimensionality Reduction/Feature Selection
- Choose ML Algorithm
- Experimental Design
- Test and Validate
- Run System

Steps used in Machine Learning

1. **Collecting data:** Be it the raw data from excel, access, text files etc., this step (gathering past data) forms the foundation of the future learning.
   - The better the variety, density and volume of relevant data, better the learning prospects for the machine becomes.
2. **Preparing the data:** Any analytical process thrives on the quality of the data used.
   - One needs to spend time determining the quality of data and then taking steps for fixing issues such as missing data and treatment of outliers.
3. **Training a model:** This step involves choosing the appropriate algorithm and representation of data in the form of the model.
   - The cleaned data is split into two parts – train and test (proportion depending on the prerequisites);
   - the first part (training data) is used for developing the model.
   - The second part (test data), is used as a reference.

Steps used in Machine Learning

4. **Evaluating the model:** To test the accuracy, the second part of the data (holdout / test data) is used.
   - This step determines the precision in the choice of the algorithm based on the outcome.
   - A better test to check accuracy of model is to see its performance on data which was not used at all during model build.
5. **Improving the performance:** This step might involve choosing a different model altogether or introducing more variables to augment the efficiency.
   - That's why significant amount of time needs to be spent in data collection and preparation.

Gradient Descent

- **Gradient Descent** is an optimization algorithm commonly used in machine learning and deep learning to minimize a function, typically a loss or cost function.
- The main idea behind gradient descent is to iteratively adjust the parameters of a model to find the values that minimize the function.
- **How Gradient Descent Works:**
  - **Initialization**: Start with random or pre-defined values for the parameters (weights, biases, etc.) of the model.
  - **Compute the Gradient**: Calculate the gradient of the loss function with respect to the parameters. The gradient is a vector of partial derivatives that points in the direction of the steepest descent.
  - **Update the Parameters**: Move the parameters in the opposite direction of the gradient by a small step (learning rate). This is mathematically expressed as:

$$w^{new} = w^{old} - \alpha \nabla_w J$$

where w: parameters (weights, biases, etc.)

α: Learning Rate (step size)

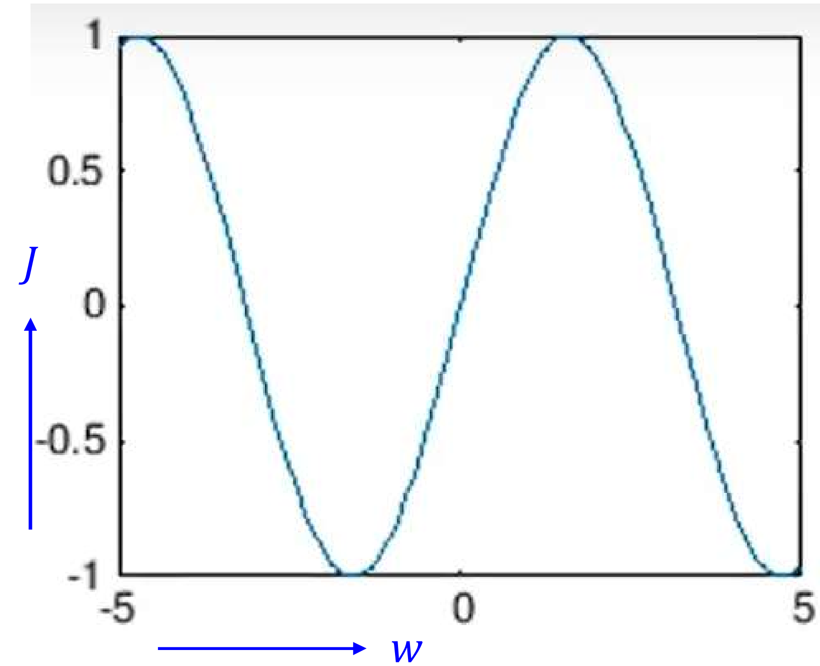
∇_w J: Gradient of the loss function with respect to the parameters.

  - **Repeat**: Continue updating the parameters until the loss function converges (i.e., stops decreasing significantly or reaches a predefined tolerance).

Gradient Descent (Scalar Case)

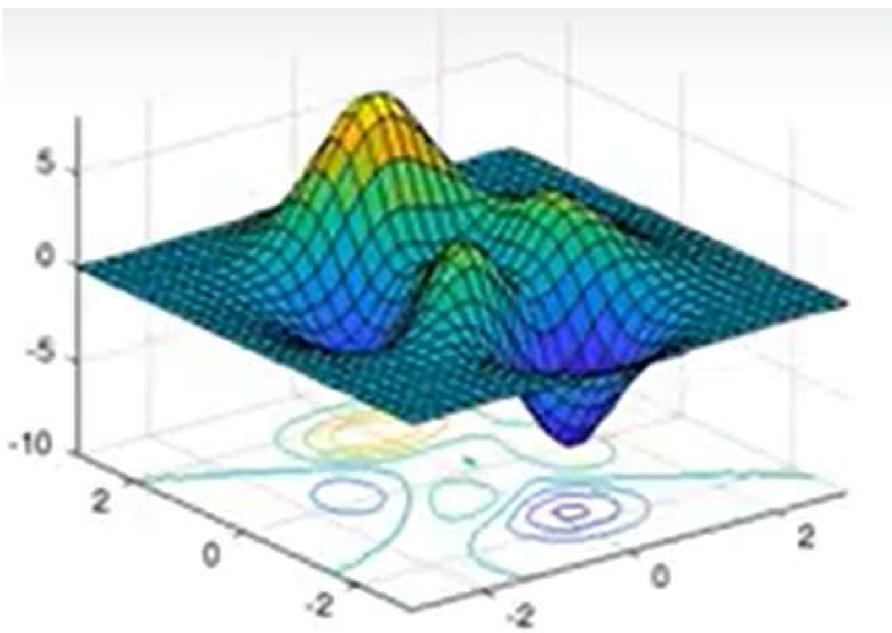
- Our task is to improve our guess for w such that we move from a region of higher gradient to a region of lower gradient.
- For scalar (i.e. one component) w, this is easy

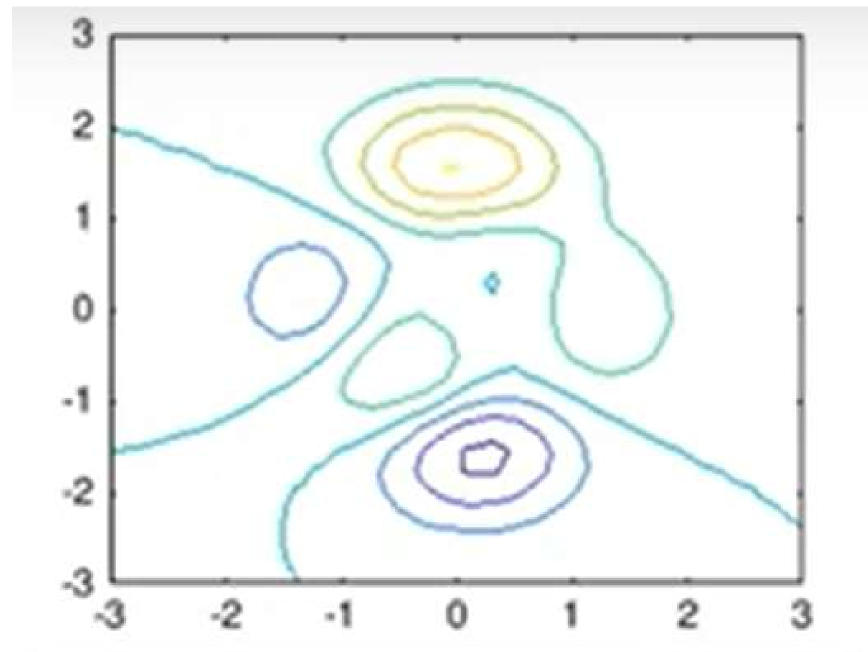
$$w^{new} = w^{old} - \alpha \frac{dJ}{dw}$$



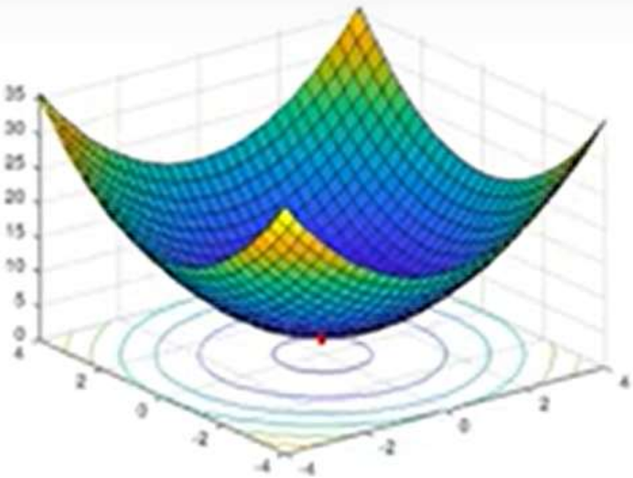
Gradient Descent (Vector Case)

- For the vector case we rely on a theorem that says "At any given point the gradient gives the direction of steepest descent".



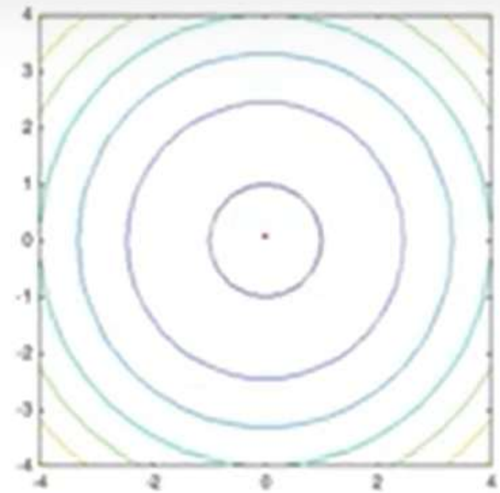


Gradient Descent: Example



$$J(w) = w_1^2 + w_2^2 + 4$$

$$\nabla_w J(w) = \begin{bmatrix} 2w_1 \\ 2w_2 \end{bmatrix}$$



- Gradient Descent gives the iterative formula:

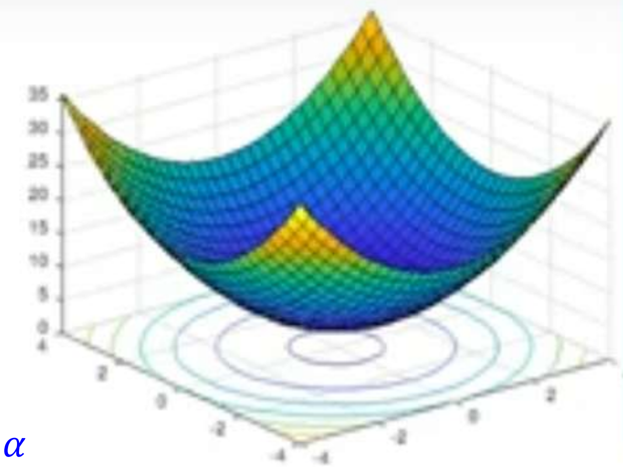
$$w_1^{k+1} = w_1^k - \alpha 2 w_1^k$$
$$w_2^{k+1} = w_2^k - \alpha 2 w_2^k$$

- We know that the actual minimum is at $w = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.
- Let us guess the initial guess is at $w^0 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$.
- Let us see different cases of $\alpha = 2, 1, 0.1, 0.5$

Gradient Descent: Example

$w^0 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, $\alpha = 2$

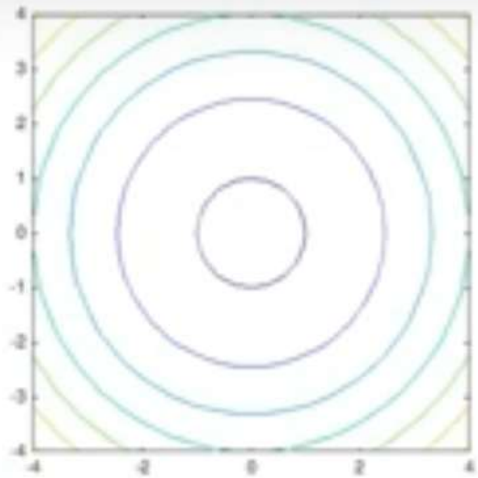
Divergent case of α



$$J(w) = w_1^2 + w_2^2 + 4$$

$$\nabla_w J(w) = \begin{bmatrix} 2w_1 \\ 2w_2 \end{bmatrix}$$

$$w_1^{k+1} = w_1^k - \alpha\left(2w_1^k\right)$$
$$w_2^{k+1} = w_2^k - \alpha\left(2w_2^k\right)$$



Iteration	w^k	$\nabla_w J = 2\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$	J	$w^{k+1} = w^k - \alpha \nabla_w J$
0	$\begin{bmatrix} 3 \\ 4 \end{bmatrix}$	$\begin{bmatrix} 6 \\ 8 \end{bmatrix}$	29	$\begin{bmatrix} -9 \\ -12 \end{bmatrix}$
1	$\begin{bmatrix} -9 \\ -12 \end{bmatrix}$	$\begin{bmatrix} -18 \\ -24 \end{bmatrix}$	229	$\begin{bmatrix} 27 \\ 36 \end{bmatrix}$
2	$\begin{bmatrix} 27 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 54 \\ 72 \end{bmatrix}$	2029	$\begin{bmatrix} -84 \\ -108 \end{bmatrix}$

Gradient Descent: Example

$\alpha = 1$

Oscillates

Iteration	w^k	$\nabla_w J = 2\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$	J	$w^{k+1} = w^k - \alpha \nabla_w J$
0	$\begin{bmatrix} 3 \\ 4 \end{bmatrix}$	$\begin{bmatrix} 6 \\ 8 \end{bmatrix}$	29	$\begin{bmatrix} -3 \\ -4 \end{bmatrix}$
1	$\begin{bmatrix} -3 \\ -4 \end{bmatrix}$	$\begin{bmatrix} -6 \\ -8 \end{bmatrix}$	29	$\begin{bmatrix} 3 \\ 4 \end{bmatrix}$
2	$\begin{bmatrix} 3 \\ 4 \end{bmatrix}$	$\begin{bmatrix} 6 \\ 8 \end{bmatrix}$	29	$\begin{bmatrix} -3 \\ -4 \end{bmatrix}$

$\alpha = 0.1$

Converges slowly

Iteration	w^k	$\nabla_w J = 2\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$	J	$w^{k+1} = w^k - \alpha \nabla_w J$
0	$\begin{bmatrix} 3 \\ 4 \end{bmatrix}$	$\begin{bmatrix} 6 \\ 8 \end{bmatrix}$	29	$\begin{bmatrix} 2.4 \\ 3.2 \end{bmatrix}$
1	$\begin{bmatrix} 2.4 \\ 3.2 \end{bmatrix}$	$\begin{bmatrix} 4.8 \\ 6.4 \end{bmatrix}$	20	$\begin{bmatrix} 1.92 \\ 2.56 \end{bmatrix}$
2	$\begin{bmatrix} 1.92 \\ 2.56 \end{bmatrix}$	$\begin{bmatrix} 3.84 \\ 5.12 \end{bmatrix}$	14.24	$\begin{bmatrix} 1.536 \\ 2.048 \end{bmatrix}$
30	$\begin{bmatrix} 0.0037 \\ 0.005 \end{bmatrix}$	$\dots$	4.000	$\dots$

$\alpha = 0.5$

Converges rapidly

Iteration	w^k	$\nabla_w J = 2\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$	J	$w^{k+1} = w^k - \alpha \nabla_w J$
0	$\begin{bmatrix} 3 \\ 4 \end{bmatrix}$	$\begin{bmatrix} 6 \\ 8 \end{bmatrix}$	29	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$
1	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	4	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Some Lessons from the example

- It is possible for gradient descent algorithm to:
  - Diverge ($\alpha = 2$)
  - Oscillates without diverging or converging ($\alpha = 1$)
  - Converge slowly ($\alpha = 0.1$)
  - Converge rapidly ($\alpha = 0.5$)
- The choice of α is important and is part of algorithm design.
- α is a hyper parameter (a parameter that must be set before learning begins).

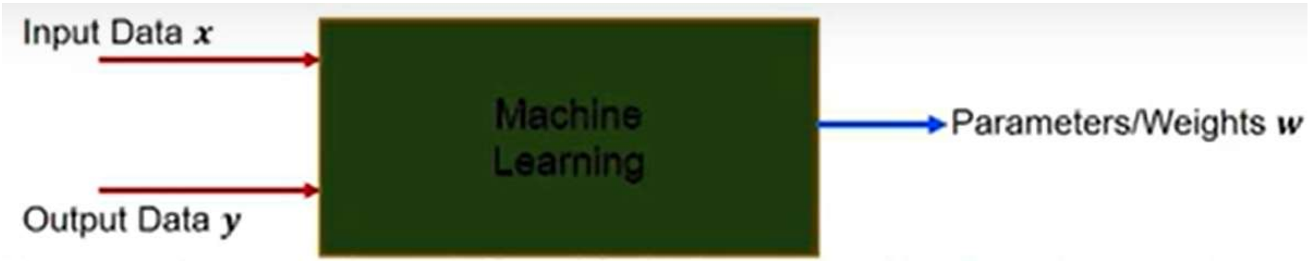
Stopping Criteria of Gradient Descent

- Ideally, we should stop when $\nabla_w J = 0$.
- This almost never happens in practice as the number of iterations required could be infinite.
- In practice, we decide on some precision ϵ (say $\epsilon \approx 10^{-5}$)
- Multiple options for stopping criteria. Stop when
  - $\|w^{k+1} - w^k\| \leq \epsilon$
  - $\| \nabla_w J(w^k)\| \leq \epsilon$
  - $\|J(w^{k+1}) - J(w^k)\| \leq \epsilon$

Summary of Gradient Descent Procedure

1. Decide on α, ϵ and stopping criterion.
2. Make an initial guess for $w = w^0$.
3. Calculate $w^{k+1} = w^k - \alpha \nabla_w J$
   a) Calculate gradient numerically, if required.
4. Calculate stopping criterion.
   a) If satisfied, stop
   b) If not satisfied, go to Step 3.

The Learning Paradigm

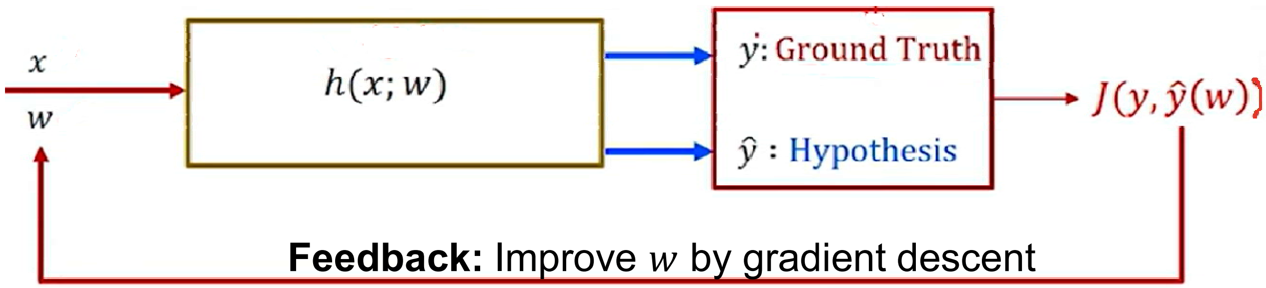


- We wish to learn the relationship between the input and output data.
- For now, we will think of this relationship as a function (which relates $x \rightarrow y$).
  - We call this function the model or hypothesis function.
- The function has two parts:
  - Form of the function [e.g. $h(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2$]
  - Parameter of the function
- Typically machine learning learns only the parameters.
- The form is provided by ML Engineer, which requires domain knowledge.

Forward Modeling

- A model or hypothesis is an educated guess at what the relationship between input and output is.
- As mentioned before, it has two pieces:
  - Form of the function: Linear, Quadratic, Exponential etc.
  - Parameters of the function
- We sometimes use the notation $y = f(x; w)$
  - Given x and choice of w, we can find a corresponding y.
- The function f going from x to y is called the forward model.
  - The process is sometimes called feedforward

Learning the parameters via feedback



To learn the parameters we follow this paradigm.

- Collect lots of data pairs (Input Vector, Output Vector) = (x, y)
- Guess for the form of the hypothesis function $h(x; w)$
  - Example: $h(x;w) = w_0 + w_1x_1 + w_2x_2$
- For an arbitrary guess for w
  - We will get some $\hat{y} = h(x; w)$ which will not match the ground truth y.
- Define a cost function $J(y, \hat{y}(w))$ depending on the difference.
- Find the optimal w by minimizing $J(w)$.
  - By using some optimization procedure such as gradient descent

What the ML Engineer must provide

- Appropriate decisions for input and output vectors (x, y)
  - All problems are data, all solutions are functions/maps
- Choosing appropriate datasets (should be usually large).
- Some appropriate form of the forward model $\hat{y} = h(x; w)$
  - Example: Linear Model $\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$
- Form the loss function
  - Example: Least Squares $J(y, \hat{y}) = (y - \hat{y})^2$
- Optimization Algorithm – Example: Gradient Descent
  - And associated hyper parameters such as α.

Machine Learning is not magic! It requires a lot of input from engineers.

A look ahead in this course

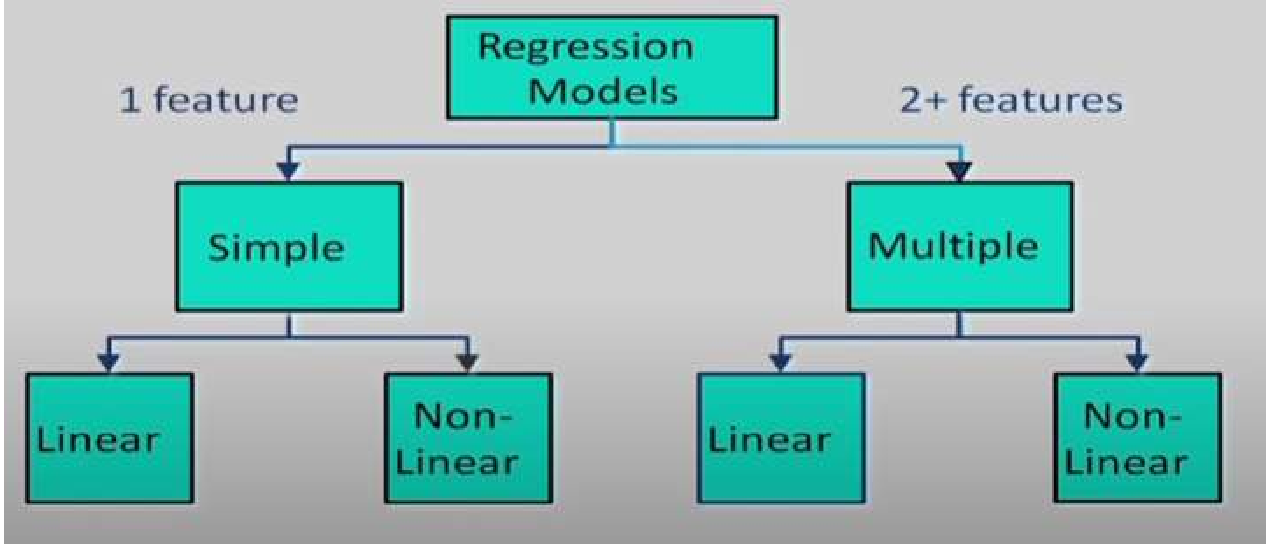
- We will be looking in the next few weeks at various types of hypothesis function $\hat{y} = h(x; w)$.
- Each of these have their own purpose and domain where they work well:
  - **Linear Regression**: For simple polynomial regression problems
  - **Logistic Regression**: For two-class (binary) classification problems
  - **Deep Neural Networks**: For any general non-linear problems
    - There is a theorem that assures us that sufficiently large neural network will approximate any function.
- There are also possible loss functions for each. We will see what is the loss function for each hypothesis function.
- It is possible you might have a better model than these for your problems. However, the general procedure outlined here remains the same.

Regression

- Regression is a statistical method used for predicting continuous outcome (dependent variable) based on one or more input features (independent variables).
- Purpose of regression is to identify the relationship between variables, understand patterns and make future predictions.
- **Goal:** Given a training set comprising of N observations $\{x_i\}, i = 1, 2, \ldots, N$, together with corresponding target values, the goal is to predict the value of y for a new value of x.

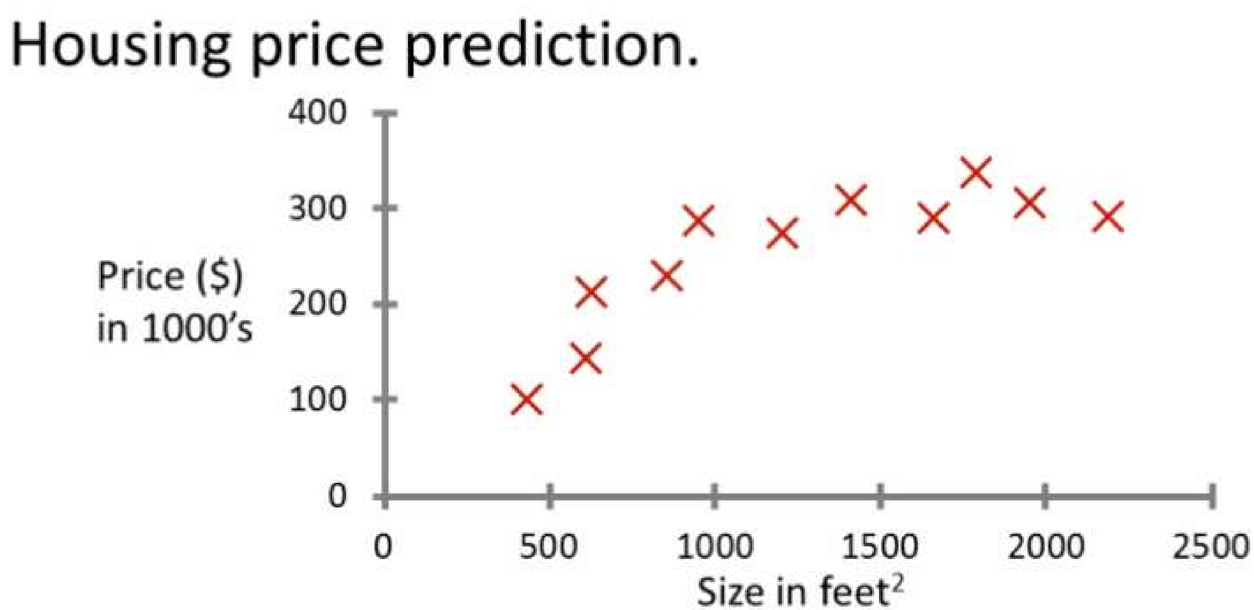
Types of Regression Models

1. **Simple Linear Regression**: Relationship between two variables.
2. **Multiple Linear Regression**: Relationship between one dependent and multiple dependent variables.
3. **Polynomial Regression**: Extension of Linear Regression by adding polynomial terms.



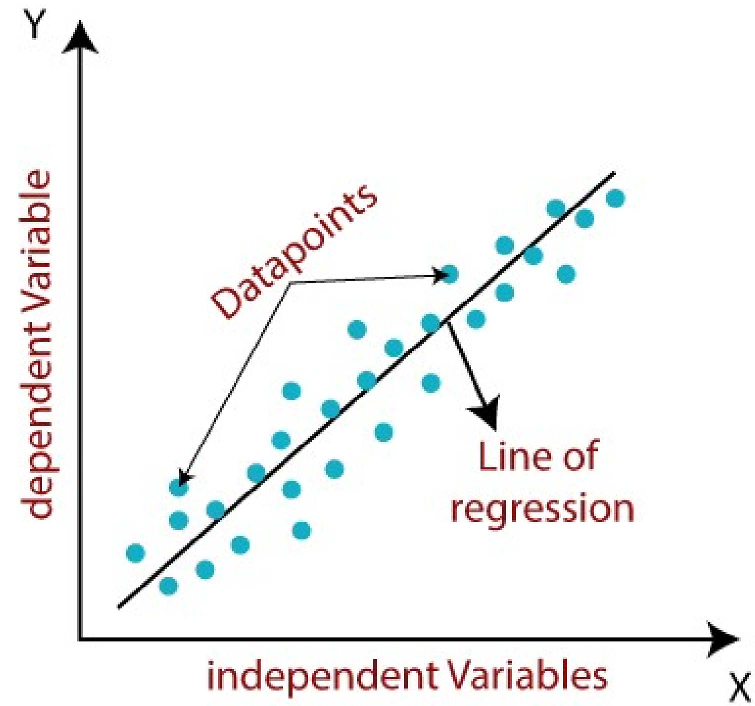
Introduction to Linear Regression

- Problem: Given m number of data points (x_i, y_i), where x_i is the **size of a house** in square feet and y_i is its **price** in 1000 $s. We are asked to predict the price of a house (not in the data set), given its size x.



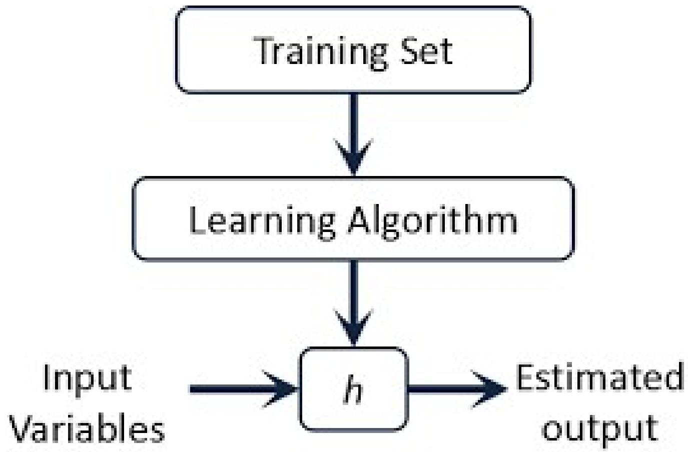
Linear Regression

- Regression is a method of modelling a target value based on independent predictors.
- Simple linear regression is a type of regression analysis where the number of independent variables is one and there is a linear relationship between the independent (x) and dependent (y) variable.
- Based on the given data points, we try to plot a line that models the points the best.
- The line can be modelled based on the linear equation: $y = mx + c$
- The motivation of the linear regression algorithm is to find the best values for m and c.



Linear Regression

- Let us write the equation $y = mx + c$ as
$$h_\theta(x) = \theta_0 + \theta_1 x$$
- In short $h: x \rightarrow y$, where h is known as the hypotheses and θ_0, θ_1 are the parameters.
- So, we are interested to find the best values of the parameters θ_0, θ_1 so that the hypothesis fits the data points the best.

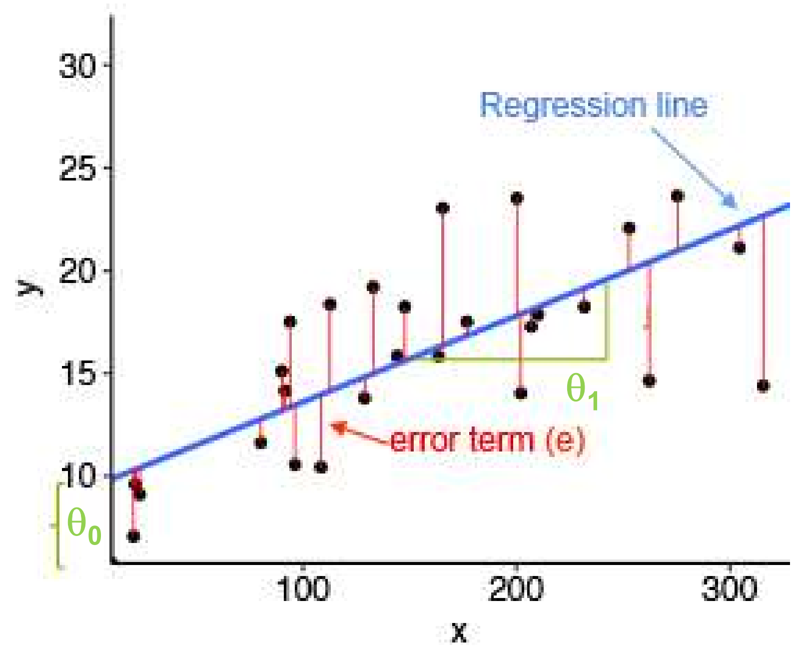


Cost Function

- The cost function helps us to figure out the best possible values for θ_0 and θ_1 which would provide the best fit line for the data points.
- Choose θ_0, θ_1 such that $h_\theta(x)$ is close to y for our training examples (x, y).
- $\underset{\theta_0,\theta_1}{\text{Minimize}} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
  where m: number of training examples and $h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$
- Formally the cost function (least squared error function) is:
  $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

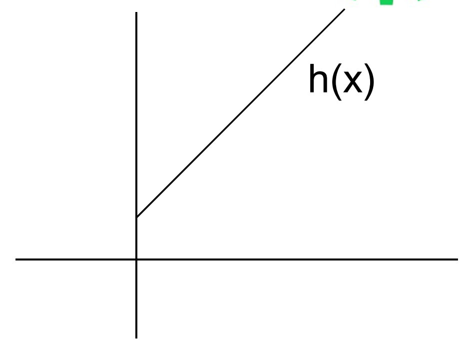
Cost Function

- We choose the above function to minimize.
- The difference between the predicted values and ground truth measures the error difference.
- We square the error difference and sum over all data points and divide that value by the total number of data points.
- This cost function is also known as the Mean Squared Error (MSE) function as it provides the average squared error over all the data points.
- Now, using this MSE function we are going to change the values of θ_0 and θ_1 such that the MSE value settles at the minima.

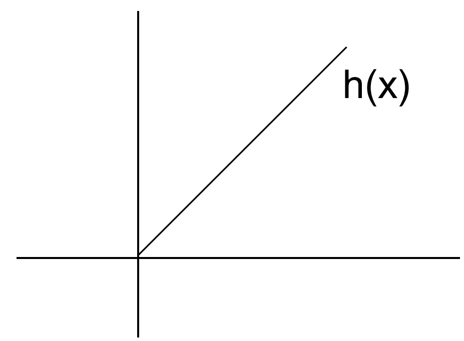


Cost Function: Intuition I

- In formal way
  - Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$
  - Parameters: θ_0, θ_1
  - Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
    $where\ h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$
  - Goal: $\underset{\theta_0,\theta_1}{\text{Minimize}}\ J(\theta_0, \theta_1)$

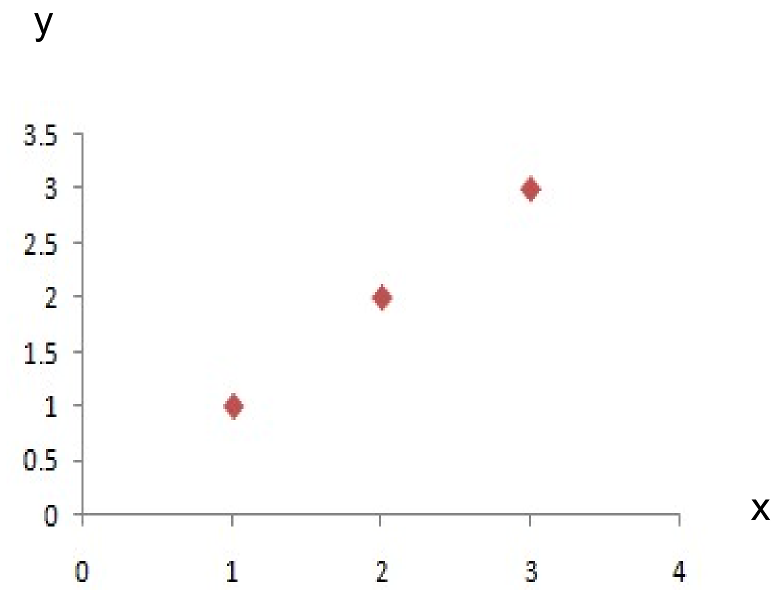


- A simplified Hypothesis Function
  - Hypothesis: $h_\theta(x) = \theta_1 x\ [\theta_0 = 0]$
  - Parameters: θ_1
  - Cost Function: $J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2\ where\ h_\theta(x^{(i)}) = \theta_1 x^{(i)}$
  - Goal: $\underset{\theta_1}{\text{Minimize}}\ J(\theta_1)$



Cost Function: Intuition I

- Comparison of Hypothesis ($h_\theta(x)$) with Cost Function ($J(\theta_1)$):
  - The hypothesis is a function of x for fixed θ_1 and cost function is a function of the parameter θ_1
- Let us assume the training set contains training data $(1,1)$, $(2,2)$ and $(3,3)$. So $m = 3$.



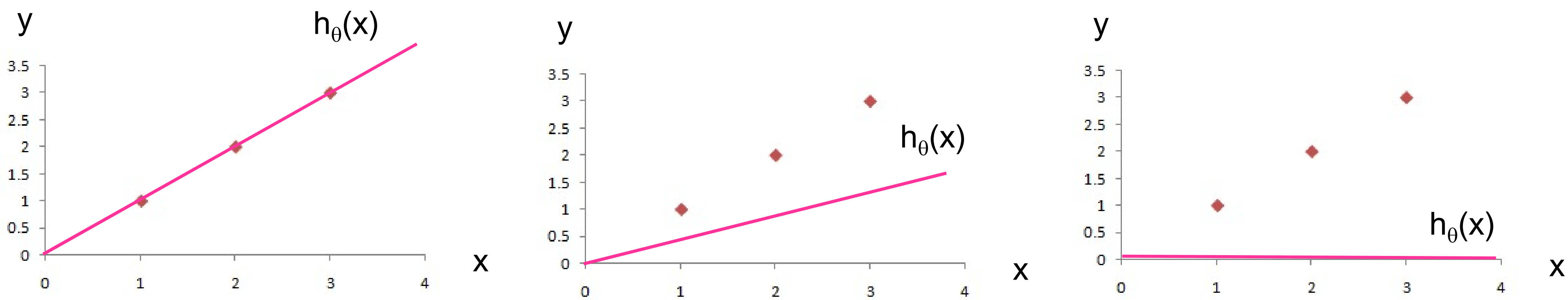
Cost Function: Intuition I

$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 = \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_1 x^{(i)} - y^{(i)}\right)^2$$

$$J(1) = \frac{1}{2 \times 3}[(1-1)^2 + (2-2)^2 + (3-3)^2] = 0$$

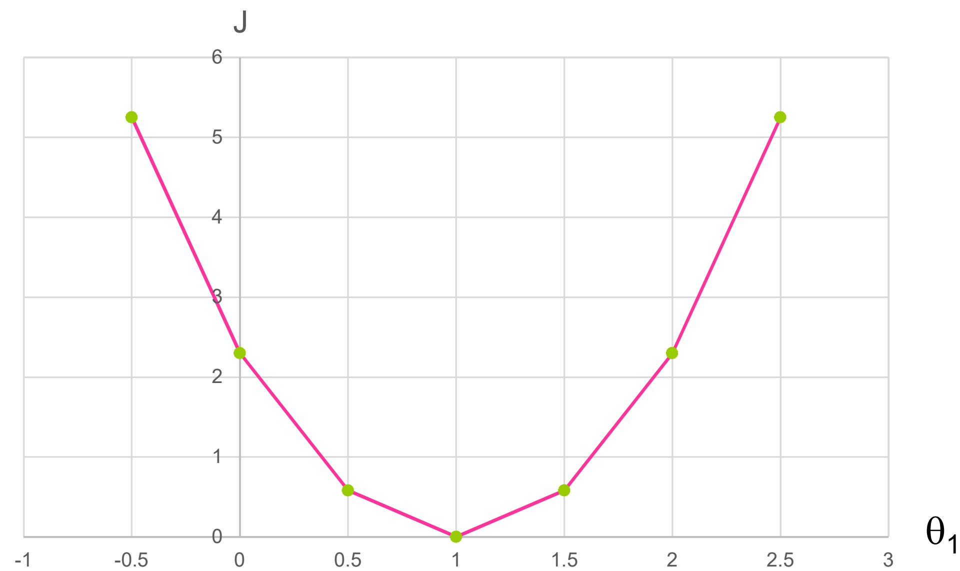
$$J(0.5) = \frac{1}{2 \times 3}[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2] = 0.58$$

$$J(0) = \frac{1}{2 \times 3}[(0-1)^2 + (0-2)^2 + (0-3)^2] = 2.3$$



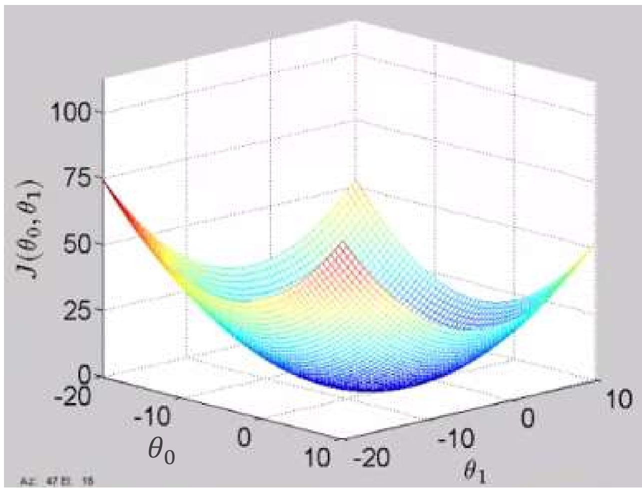
Cost Function: Intuition I

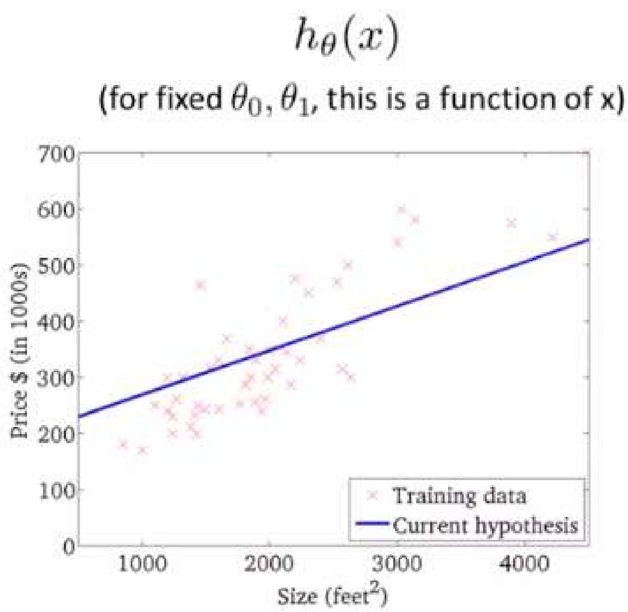
- We have computed $J(1) = 0$, $J(0.5) = 0.58$, $J(0) = 2.3$.
- In the similar way we can compute $J(-0.5) = 5.25$, $J(1.5) = 0.58$, $J(2) = 2.3$ and $J(2.5) = 5.25$.
- Plotting $(\theta_1, J(\theta 1))$ we have the following graph:
- Optimal value of θ_1 is 1.

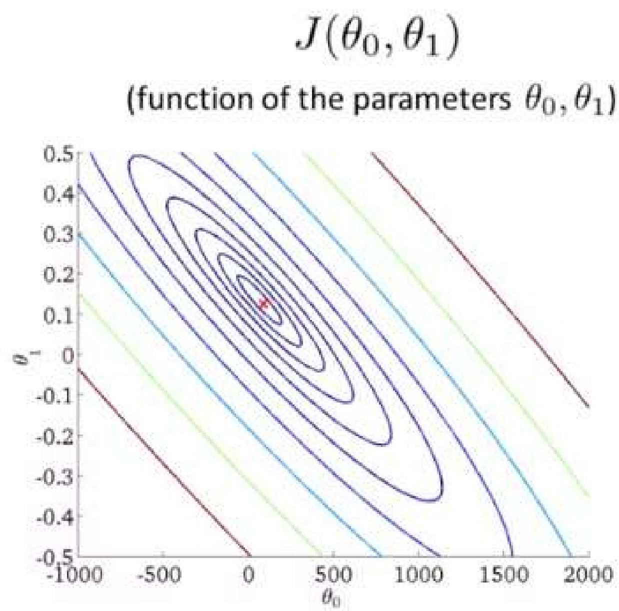


Cost Function: Intuition II

Contour Plots: Each of the ovals shows the set of points takes the same value of $J(\theta_0, \theta_1)$







Least Square Method of Solving Simple Linear Regression

- For i[th] data point the predicted value $\hat{y}^{(i)} = \theta_0 + \theta_1 x^{(i)}$.
- The error (residual) for i[th] data point: $Error_i = y^{(i)} - \hat{y}^{(i)} = y^{(i)} - (\theta_0 + \theta_1 x^{(i)})$.
- The sum of squared error of the model,

$$\mathrm{SSE}(\theta_0, \theta_1) = \sum_{i=1}^{N}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \sum_{i=1}^{N}\left(y^{(i)} - (\theta_0 + \theta_1 x^{(i)})\right)^2$$

- Partial derivative of SSE w.r.t, θ_1:

$$\frac{\partial(SSE)}{\partial\theta_1} = \frac{\partial}{\partial\theta_1}\sum_{i=1}^{N}\left(y^{(i)} - (\theta_0 + \theta_1 x^{(i)})\right)^2 = -2\sum_{i=1}^{N} x^{(i)}\left(y^{(i)} - \theta_0 - \theta_1 x^{(i)}\right)$$

$$\frac{\partial(SSE)}{\partial\theta_1} = 0 \Rightarrow -2\sum_{i=1}^{N} x^{(i)}\left(y^{(i)} - \theta_0 - \theta_1 x^{(i)}\right) = 0 \Rightarrow \sum_{i=1}^{N} x^{(i)} y^{(i)} = \theta_1 \sum_{i=1}^{N}\left(x^{(i)}\right)^2 + \theta_0 \sum_{i=1}^{N} x^{(i)} \quad \text{-------------- } (1)$$

Least Square Method of Solving Simple Linear Regression

- Partial derivative of SSE w.r.t, θ_0:

$$\frac{\partial(SSE)}{\partial\theta_0} = \sum_{i=1}^{N}(-1)2\left(y^{(i)} - \theta_0 - \theta_1 x^{(i)}\right) = -2\sum_{i=1}^{N}\left(y^{(i)} - \theta_0 - \theta_1 x^{(i)}\right)$$

$$\frac{\partial(SSE)}{\partial\theta_0} = 0 \Longrightarrow -2\sum_{i=1}^{N}\left(y^{(i)} - \theta_0 - \theta_1 x^{(i)}\right) = 0 \Longrightarrow \sum_{i=1}^{N} y^{(i)} = \theta_1 \sum_{i=1}^{N} x^{(i)} + \theta_0 \sum_{i=1}^{N} 1$$

$$\Longrightarrow \sum_{i=1}^{N} y^{(i)} = \theta_1 \sum_{i=1}^{N} x^{(i)} + \theta_0 N \qquad ------------- -(2)$$

- From equation (2) we have: $\theta_0 = \frac{\sum y^{(i)} - \theta_1 \sum x^{(i)}}{N}$ $\qquad ------------- -(3)$

- Substituting the value of θ_0 in equation (1): $\sum x^{(i)} y^{(i)} = \theta_1 \sum (x^{(i)})^2 + \left(\frac{\sum y^{(i)} - \theta_1 \sum x^{(i)}}{N}\right) \sum x^{(i)}$

$$\Longrightarrow \theta_1 = \frac{\sum x^{(i)} \sum y^{(i)} - N \sum x^{(i)} y^{(i)}}{(\sum x^{(i)})^2 - N \sum (x^{(i)})^2} \Longrightarrow \theta_1 = \frac{N \sum x^{(i)} y^{(i)} - \sum x^{(i)} \sum y^{(i)}}{N \sum (x^{(i)})^2 - (\sum x^{(i)})^2}$$

Least Square Method of Solving Simple Linear Regression

Example:

Consider the following dataset of points:

x	y
1	2
2	3
3	5
4	7
5	8

Find the best fit line using the list square method.

$$\theta_1 = \frac{N\sum x^{(i)}y^{(i)} - \sum x^{(i)}\sum y^{(i)}}{N\sum(x^{(i)})^2 - (\sum x^{(i)})^2}$$

$$\theta_0 = \frac{\sum y^{(i)} - \theta_1 \sum x^{(i)}}{N}$$

Solution:

$$\theta_1 = \frac{5 \times 91 - 15 \times 25}{5 \times 55 - (15)^2} = 1.6$$

$$\theta_0 = \frac{25 - 1.6 \times 15}{5} = 0.2$$