"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. What is the task T in this setting?

Classify emails as spam or not spam. (A) (Task T)

Watching you label emails as spam or not spam. (Experience E)

The number (or fraction) of emails correctly classified as spam/not spam. (Performance P)

None of the above, this is not a machine learning algorithm.

---------------------------------------------------------------------------------------------------------------------------------------------------------------------

What is Machine Learning?

Two definitions of Machine Learning are offered. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed." This is an older, informal definition.

Tom Mitchell provides a more modern definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Example: playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

In general, any machine learning problem can be assigned to one of two broad classifications:

Supervised learning and Unsupervised learning.

---------------------------------------------------------------------------------------------------------------------------------------------------------------------

You’re running a company, and you want to develop learning algorithms to address each of 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 3 months.

Problem 2: You’d 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?

Treat both as classification problems.

Treat problem 1 as a classification problem, problem 2 as a regression problem.

Treat problem 1 as a regression problem, problem 2 as a classification problem. (A)

Treat both as regression problems.

---------------------------------------------------------------------------------------------------------------------------------------------------------------------

Supervised Learning

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output.

Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

Example 1:

Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output, so this is a regression problem.

We could turn this example into a classification problem by instead making our output about whether the house "sells for more or less than the asking price." Here we are classifying the houses based on price into two discrete categories.

Example 2:

(a) Regression - Given a picture of a person, we have to predict their age on the basis of the given picture

(b) Classification - Given a patient with a tumor, we have to predict whether the tumor is malignant or benign.

---------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

Given email labeled as spam/not spam, learn a spam filter. (Supervised)

Given a set of news articles found on the web, group them into sets of articles about the same stories. (Un Supervised)

Given a database of customer data, automatically discover market segments and group customers into different market segments. (Un Supervised)

Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not. (Supervised)

---------------------------------------------------------------------------------------------------------------------------------------------------------------------

Unsupervised Learning

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data.

With unsupervised learning there is no feedback based on the prediction results.

Example:

Clustering: Take a collection of 1,000,000 different genes, and find a way to automatically group these genes into groups that are somehow similar or related by different variables, such as lifespan, location, roles, and so on.

Non-clustering: The "Cocktail Party Algorithm", allows you to find structure in a chaotic environment. (i.e. identifying individual voices and music from a mesh of sounds at a cocktail party).

---------------------------------------------------------------------------------------------------------------------------------------------------------------------

Consider the training set shown below. (x^{(i)}, y^{(i)}) is the i^{th} training example. What is y^{(3)}

Size in feet^2

2

(xx) Price ($) in 1000's (yy)

2104 460

1416 232

1534 315

852 178

... ...

( Q is what's y superscript 3 (i.e. 3rd row y value))

1416

1534

315 (A)

0

---------------------------------------------------------------------------------------------------------------------------------------------------------------------

# Model Representation

To establish notation for future use, we’ll use x(i) to denote the “input” variables (living area in this example), also called input features, and y(i) to denote the “output” or target variable that we are trying to predict (price). A pair (x(i),y(i)) is called a training example, and the dataset that we’ll be using to learn—a list of m training examples (x(i),y(i));i=1,...,m—is called a training set. Note that the superscript “(i)” in the notation is simply an index into the training set, and has nothing to do with exponentiation. We will also use X to denote the space of input values, and Y to denote the space of output values. In this example, X = Y = ℝ.

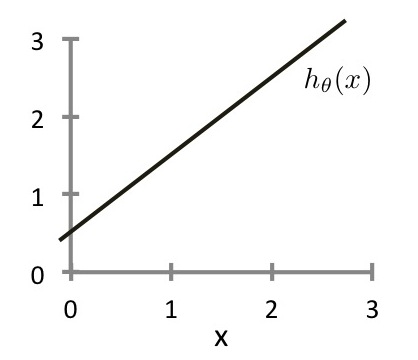
To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function h : X → Y so that h(x) is a “good” predictor for the corresponding value of y. For historical reasons, this function h is called a hypothesis. Seen pictorially, the process is therefore like this:



When the target variable that we’re trying to predict is continuous, such as in our housing example, we call the learning problem a regression problem. When y can take on only a small number of discrete values (such as if, given the living area, we wanted to predict if a dwelling is a house or an apartment, say), we call it a classification problem.

Q :

Consider the plot below of hθ​(x)=θ0​+θ1​x. What are θ0​ and θ1​?



θ0​=0,θ1​=1

θ0​=0.5,θ1​=1 (A)

θ0​=1,θ1​=0.5

θ0​=1,θ1​=1

# Cost Function

We can measure the accuracy of our hypothesis function by using a cost function. This takes an average difference (actually a fancier version of an average) of all the results of the hypothesis with inputs from x's and the actual output y's.

J(θ0,θ1)=12m∑i=1m(y^i−yi)2=12m∑i=1m(hθ(xi)−yi)2

To break it apart, it is 21​ x¯ where x¯ is the mean of the squares of hθ​(xi​)−yi​ , or the difference between the predicted value and the actual value.

This function is otherwise called the "Squared error function", or "Mean squared error". The mean is halved (21​) as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the 21​term. The following image summarizes what the cost function does:



# Cost Function - Intuition I

If we try to think of it in visual terms, our training data set is scattered on the x-y plane. We are trying to make a straight line (defined by h\_\theta(x)*hθ*​(*x*)) which passes through these scattered data points.

Our objective is to get the best possible line. The best possible line will be such so that the average squared vertical distances of the scattered points from the line will be the least. Ideally, the line should pass through all the points of our training data set. In such a case, the value of J(\theta\_0, \theta\_1)*J*(*θ*0​,*θ*1​) will be 0. The following example shows the ideal situation where we have a cost function of 0.



When \theta\_1 = 1*θ*1​=1, we get a slope of 1 which goes through every single data point in our model. Conversely, when \theta\_1 = 0.5*θ*1​=0.5, we see the vertical distance from our fit to the data points increase.



This increases our cost function to 0.58. Plotting several other points yields to the following graph:



Thus as a goal, we should try to minimize the cost function. In this case, \theta\_1 = 1*θ*1​=1 is our global minimum.

# Cost Function - Intuition II

A contour plot is a graph that contains many contour lines. A contour line of a two variable function has a constant value at all points of the same line. An example of such a graph is the one to the right below.



Taking any color and going along the 'circle', one would expect to get the same value of the cost function. For example, the three green points found on the green line above have the same value for J(\theta\_0,\theta\_1)*J*(*θ*0​,*θ*1​) and as a result, they are found along the same line. The circled x displays the value of the cost function for the graph on the left when \theta\_0*θ*0​ = 800 and \theta\_1*θ*1​= -0.15. Taking another h(x) and plotting its contour plot, one gets the following graphs:



When \theta\_0*θ*0​ = 360 and \theta\_1*θ*1​ = 0, the value of J(\theta\_0,\theta\_1)*J*(*θ*0​,*θ*1​) in the contour plot gets closer to the center thus reducing the cost function error. Now giving our hypothesis function a slightly positive slope results in a better fit of the data.



The graph above minimizes the cost function as much as possible and consequently, the result of \theta\_1*θ*1​ and \theta\_0*θ*0​ tend to be around 0.12 and 250 respectively. Plotting those values on our graph to the right seems to put our point in the center of the inner most 'circle'.

# Gradient Descent

So we have our hypothesis function and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in the hypothesis function. That's where gradient descent comes in.

Imagine that we graph our hypothesis function based on its fields \theta\_0*θ*0​ and \theta\_1*θ*1​ (actually we are graphing the cost function as a function of the parameter estimates). We are not graphing x and y itself, but the parameter range of our hypothesis function and the cost resulting from selecting a particular set of parameters.

We put \theta\_0*θ*0​ on the x axis and \theta\_1*θ*1​ on the y axis, with the cost function on the vertical z axis. The points on our graph will be the result of the cost function using our hypothesis with those specific theta parameters. The graph below depicts such a setup.



We will know that we have succeeded when our cost function is at the very bottom of the pits in our graph, i.e. when its value is the minimum. The red arrows show the minimum points in the graph.

The way we do this is by taking the derivative (the tangential line to a function) of our cost function. The slope of the tangent is the derivative at that point and it will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter α, which is called the learning rate.

For example, the distance between each 'star' in the graph above represents a step determined by our parameter α. A smaller α would result in a smaller step and a larger α results in a larger step. The direction in which the step is taken is determined by the partial derivative of J(\theta\_0,\theta\_1)*J*(*θ*0​,*θ*1​). Depending on where one starts on the graph, one could end up at different points. The image above shows us two different starting points that end up in two different places.

The gradient descent algorithm is:

repeat until convergence:

*θj*:=*θj*−*α*∂∂*θjJ*(*θ*0,*θ*1)

where

j=0,1 represents the feature index number.

At each iteration j, one should simultaneously update the parameters \theta\_1, \theta\_2,...,\theta\_n*θ*1​,*θ*2​,...,*θn*​. Updating a specific parameter prior to calculating another one on the j^{(th)}*j*(*th*) iteration would yield to a wrong implementation.



Suppose \theta\_0= 1, \theta\_1= 2*θ*0​=1,*θ*1​=2, and we simultaneously update \theta\_0*θ*0​ and \theta\_1*θ*1​ using the rule:\theta\_j := \theta\_j + \sqrt{\theta\_0 \theta\_1}*θj*​:=*θj*​+*θ*0​*θ*1​​ (for j = 0 and j=1) What are the resulting values of \theta\_0*θ*0​ and \theta\_1*θ*1​?



\theta\_0 = 1, \theta\_1 =2*θ*0​=1,*θ*1​=2



\theta\_0 = 1+\sqrt{2}, \theta\_1 =2 + \sqrt{2}*θ*0​=1+2​,*θ*1​=2+2​



\theta\_0 = 2 + \sqrt{2}, \theta\_1 =1 + \sqrt{2}*θ*0​=2+2​,*θ*1​=1+2​

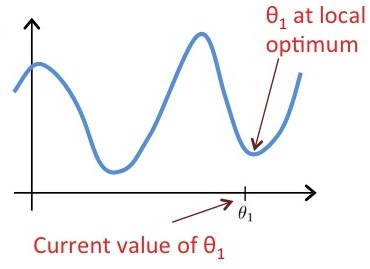


\theta\_0 = 1+\sqrt{2}, \theta\_1 =2 + \sqrt{(1 + \sqrt{2})\cdot 2}*θ*0​=1+2​,*θ*1​=2+(1+2​)⋅2​

Suppose θ1\theta\_1θ1​ is at a local optimum of J(θ1)J(\theta\_1)J(θ1​), such as shown in the figure.

What will one step of gradient descent θ1:=θ1−αddθ1J(θ1)\theta\_1 := \theta\_1 -\alpha \frac{d}{d \theta\_1} J(\theta\_1)θ1​:=θ1​−αdθ1​

d​J(θ1​) do?





Leave θ1\theta\_1θ1​ unchanged (A)



Change θ1\theta\_1θ1​ in a random direction



Move θ1\theta\_1θ1​ in the direction of the global minimum of J(θ1)J(\theta\_1)J(θ1​)



Decrease θ1\theta\_1θ1​

# Gradient Descent Intuition

In this video we explored the scenario where we used one parameter θ1\theta\_1θ1​ and plotted its cost function to implement a gradient descent. Our formula for a single parameter was :

Repeat until convergence:

|  |
| --- |
| θ1:=θ1−αddθ1J(θ1)\theta\_1:=\theta\_1-\alpha \frac{d}{d\theta\_1} J(\theta\_1)θ1​:=θ1​−αdθ1​ |

|  |
| --- |
| d​J(θ1​) |

Regardless of the slope's sign for ddθ1J(θ1)\frac{d}{d\theta\_1} J(\theta\_1)dθ1​

d​J(θ1​), θ1\theta\_1θ1​ eventually converges to its minimum value. The following graph shows that when the slope is negative, the value of θ1\theta\_1θ1​ increases and when it is positive, the value of θ1\theta\_1θ1​ decreases.



On a side note, we should adjust our parameter α\alphaα to ensure that the gradient descent algorithm converges in a reasonable time. Failure to converge or too much time to obtain the minimum value imply that our step size is wrong.



### How does gradient descent converge with a fixed step size α\alphaα?

The intuition behind the convergence is that ddθ1J(θ1)\frac{d}{d\theta\_1} J(\theta\_1)dθ1​

d​J(θ1​) approaches 0 as we approach the bottom of our convex function. At the minimum, the derivative will always be 0 and thus we get:

|  |
| --- |
| θ1:=θ1−α∗0\theta\_1:=\theta\_1-\alpha \* 0θ1​:=θ1​−α∗0 |

