What is machine learning?

A program learns from experience E with respect to tasks T and some performance measure P if it's performance at tasks in T, as measured by P, improves with experience E.

- supervised learning: teach machine what it is and how to do.
- unsupervised learning: learn by itself.

Supervised learning:

Given a data set and already know what our correct output should look like, having the idea that there is a relationship between input and output.

- Regression: predict results within a continuous output, meaning that we are trying to map input variables to some continuous function.
  - predict house price.
  - predict human's age given an image.
- Classification: predict results in a discrete output (discrete categories).
  - given a patient with tumor, predict it is malignant or benign.

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.

- Clustering: k-means.
- non-clustering: "Cocktail Party Algorithm".

Traing dataset :

m : Number of traing examples

x's : input variable / features. X : input space.

y's : output variable / features. Y : output space.

(x, y) : single traing example.

$(x^{(i)}, y^{(i)})$: refere to the i^{th} traing example.

$y^{(i)}$: we are trying to predict.

Traing set.

↓

Learning Algorithm

↓

size of house (x) → h (hypothesis) → Estimated price. (estimated value of y).

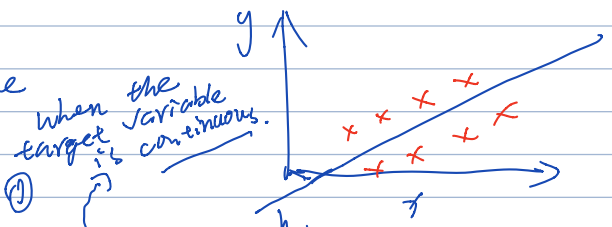
h maps from x's to y's.

$h : X \to Y$

How do we represent h ?

$h_\theta(x) = \theta_0 + \theta_1(x)$. shorthand $h(x)$.

a simple example : h is a linear function.



① when the target variable is continuous.

linear regression with one variable.

or univariable linear regression.

↓

a fancy way of saying one variable.

② when the target variable is only a small number of discrete values.

called classification problem.

Cost Function : $h_\theta(x) = \theta_0 + \theta_1 x$.

how to choose θ_0 & θ_1.

Idea : choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y).

minimize $\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$ ← # of traing examples.

$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$

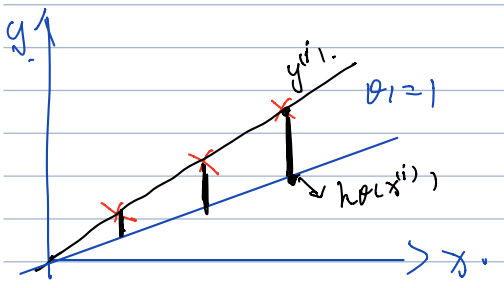
$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$

minimize $J(\theta_0,\theta_1)$ over θ_0,θ_1

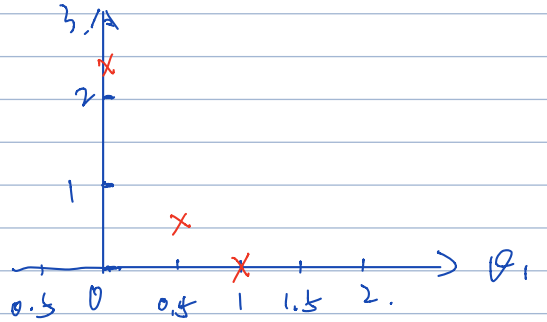
(or). Squared error function ⟵ cost function

A simple example: $h_\theta(x) = \theta_1 x$.

$h_\theta(x)$ for fixed θ_1, h is the function of x.



$J(\theta_1)$. a function of parameter θ_1.



for $\theta_1 = 1$

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

$= \frac{1}{2m}\sum_{i=1}^{m}(\theta_1 x^{(i)} - y^{(i)})^2$

$= \frac{1}{2m}(0^2+0^2+0^2) = 0.$

So, for $\theta_1 = 1$, $J(\theta_1) = 0$.

for $\theta_1 = 0$. $h_\theta(x) = 0$.

$J(0) = \frac{1}{2m}[1^2+2^2+3^2] = \frac{14}{6}$.

for $\theta_1 = 0.5$.

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

$= \frac{1}{6}(\frac{1}{4}+1+\frac{9}{4}) = \frac{3}{2}$

Objective: choose θ_1 to minimize the $J(\theta_1)$.

find the parameters to fit the data well.

Contour plot / figure.: contains many contour lines of a two variable function has a constant value at all points of the same line.

When the value in the contour plot get closer to the centre thus reducing the cost function error.

Gradient descent.

Have a function $J(\theta_0, \theta_1)$, minimize $J(\theta_0, \theta_1)$

Outline:

① start with some θ_0, θ_1, initial.

② keep changing θ_0, θ_1. to reduce $J(\theta_0, \theta_1)$.

algorithm:

repeat until convergence {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ (j = 0 and j = 1). }

$a := b$ set b equal to a no matter what is a.

:= assignment

α learning rate. how big step (fast) updating parameters.

$\frac{\partial}{\partial \theta_j}$ partial derivative. derivative

tangent 切线

Correct simultaneous update:

$temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$\theta_0 := temp0$

$\theta_1 := temp1$

Incorrect:

$temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

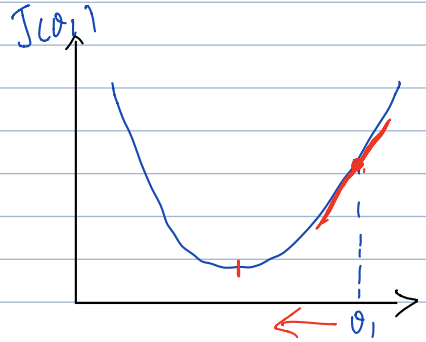
$\theta_0 := temp0$ — after updateing, when you calculate temp1, which use the new θ_0,

$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$ ←

$\theta_1 := temp1$

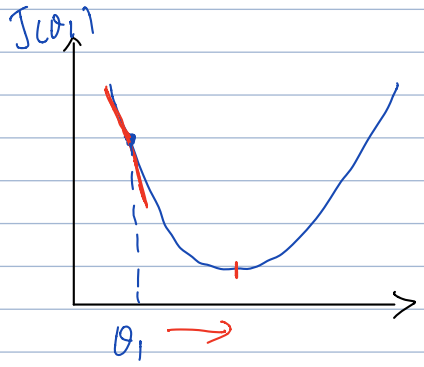
Simple examples.

$\theta_1 := \theta_1 - \alpha \left[\frac{\partial}{\partial \theta_1} J(\theta_1)\right]$ → the slope of the tangent.



$\theta_1 := \theta_1 - \alpha$ (slope/positive number).

θ_1 ↓. decreasing.

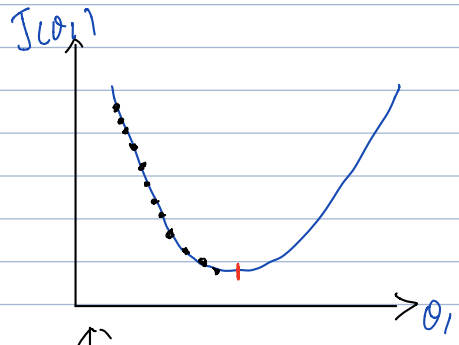


$\theta_1 := \theta_1 - \alpha$ (negative slope)

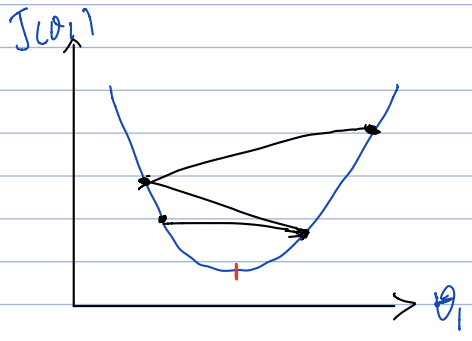
$\theta_1 \uparrow$ increasing.

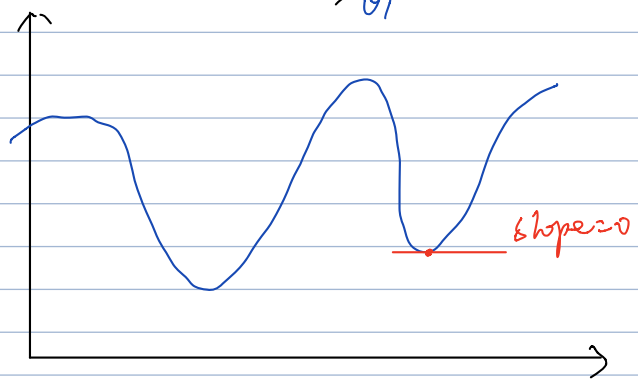
learning rate α:

If α is too small, gradient descent can be too slow.



If α is too large, GD can overshoot the minimum. It may fail to converge, or even diverge.

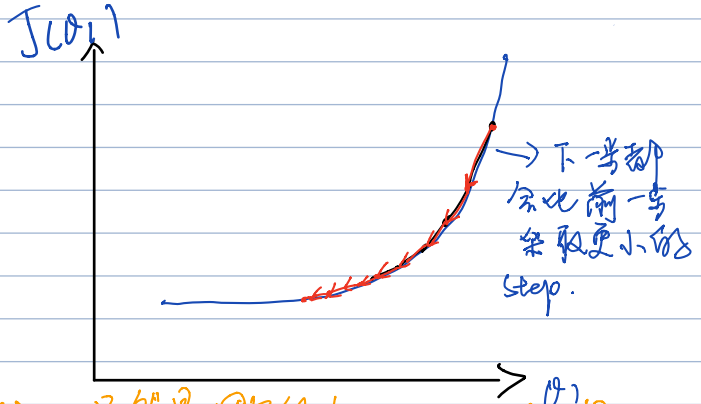




If the initial parameter is optimal, leave it unchanged.

$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_0} J(\theta_0)$

$= 0$



→ 下一步都会比前一步采取更小的 step.

Gradient descent can converge to a local minimum, even with the fixed α.

As we approach a local minimum, gradient descent will automatically take smaller steps. So no need to decrease α over time.

Notice: 虽然是固定的 learning rate α, 但下一步的 slope 总是比上一步更小，更趋近零，所以每一次更新都会比前一次小.

Gradient descent for linear regression.

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

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

$j=0$: $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)$

$j=1$: $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right) \cdot x^{(i)}$.

Notice:
for the specific choice of cost function $J(\theta_0, \theta_1)$ used in linear regression, there are no local optima. only one global optimum. so the cost function always converges to the global optimum.

repeat until convergence {

$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$

$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$.

"Batch" Gradient Descent

Each step of gradient descent uses all training samples.