Strojové učení: úvod; regrese (4. přednáška)

March 9, 2016

Machine Learning — Naučit stroje se učit

O co jde?

Máme **model výpočtu** (t.j. výpočetní postup jednoznačně daný vstupy a nějakými parametry), chceme najít vhodné nastavení parametrů, aby postup (model) dával řešení našeho problému.

Příklad

Chceme převést stupně Celsia na stupně Farenheita. Modelem výpočtu bude lineární funkce $f(x) = \alpha x + b$, která je daná dvěma parametry — α, b. Vhodné nastavení parametrů v tomto případě je $\alpha = 9/5$ a $b = 32$.

Při strojovém učení zvolíme dostatečně silný model a nastavení parametrů chceme nalézt automaticky na základě vzorových řešení daného problému (např. víme, že $0^oC = 32^oF$ a $100^oC = 212^oF$).

Strojové učení — Proč

Chceme, aby se stroj naučil řešit zadaný problém na základě vzorových řešení.

- řešení je příliš komplikované
- problém se často mění, vyvíjí
- lidská práce je drahá (v porovnání se strojovou)
- máme k dispozici tolik dat, že je není možné zpracovat "ručně"

Strojové učení — typické aplikace

- Rozpoznávání vzorců
  - věci/osoby/výrazy tváře na fotkách
  - mluvená slova
  - spam
  - medicínská diagnóza
- Rozpoznávání anomálií
  - netypické sekvence finančních transakcí
  - netypická data přicházející ze senzorů v atomové elektrárně
- Předpovídání
  - vývoj ceny akcií na burze / vývoj měnového kurzu
  - jaké filmy bude mít daný člověk rád
  - věk osoby na fotografii
- Shlukování
  - vyhledávání zpráv s podobným obsahem
  - vyhledání skupin zákazníků s podobnými vlastnostmi

Strojové učení — příklad

Rozpoznávání věcí na fotkách

(zdroj: G. Hinton, Neural Networks for ML)

40 vydra
15 křepelka
7 tetřev
6 koroptev

85 sněžný pluh
6 vrtná plošina
6 záchranný člun
2 popelářské auto

15 žížala
12 gilotina
7 orangutan
6 koště

Strojové učení — dělení

- učení s učitelem (supervised learning)
  - klasifikace
  - regrese
- učení bez učitele (unsupervised learning)
  - shluková analýza (clustering, cluster analysis)
  - latentní a faktorová analýza
- kombinace učení s učitelem a bez učitele (semi-supervised learning)
- zpětnovazebné učení, učení posilováním (reinforcement learning)
  - pasivní (postupuje podle předem dané strategie a učí se jednak zákonitosti prostředí, tj. predikovat, do jakého stavu jeho akce povede, jednak ohodnocení stavů)
  - aktivní (učí se navíc také určit svou další akci)

Regrese

Regrese je formou učení s učitelem, při které se stroj učí na základě vstupních dat (vektor příznaků) určit výstupní hodnotu (reálné číslo).

Vstupní data: množina dvojic $\{(x^j, y^j), j = 1, \ldots N\}$.
Hledáme funkci $h(x)$, která pro dané x co nejlépe aproximuje hodnotu y.

Příznaky mohou být

- spojité (reálná čísla)
- kategoriální (prvky nějaké konečné množiny)

Některé algoritmy si umí poradit s chybějícími hodnotami příznaků.

Lineární regrese

Lineární regrese je metoda proložení souboru bodů přímkou.

Předpokládáme, že závislost y na x má tvar

$$y = w_0 + w_1 x.$$

O lineární regresi mluvíme i tehdy, když je vstupních příznaků více, tj. $x = (x_1, x_2, \ldots, x_n)$ je vektor vstupních příznaků a předpokládáme, že

$$y = w_0 + w_1 x_1 + \cdots + w_n x_n.$$

$$y = w_0 + \sum_{i=1}^{n} w_i x_i$$

Lineární regrese — optimalizace parametrů

O bodech reprezentujících měřená data se předpokládá, že jejich x-ové souřadnice jsou přesné, zatímco y-ové souřadnice mohou být zatíženy náhodnou chybou.
V tomto případě lze nejlepší přímku prokládající danou množinu bodů určit **metodou nejmenších čtverců**:

$$Loss(w) = \sum_{j=1}^{N} (h_w(x_j) - y_j)^2 = \sum_{j=1}^{N} (w_0 + w_1 x^j - y^j)^2$$

Metoda nejmenších čtverců pochází od Gausse (1795).

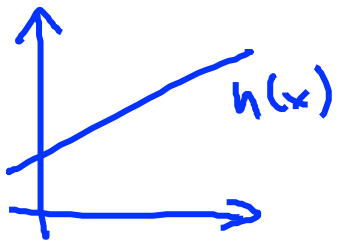
Simplified

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Parameters:

θ_0, θ_1



Cost Function:

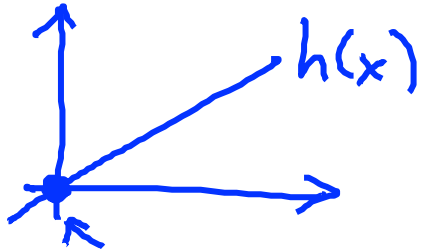
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

$h_\theta(x) = \theta_1 x$

$\theta_0 = 0$

θ_1



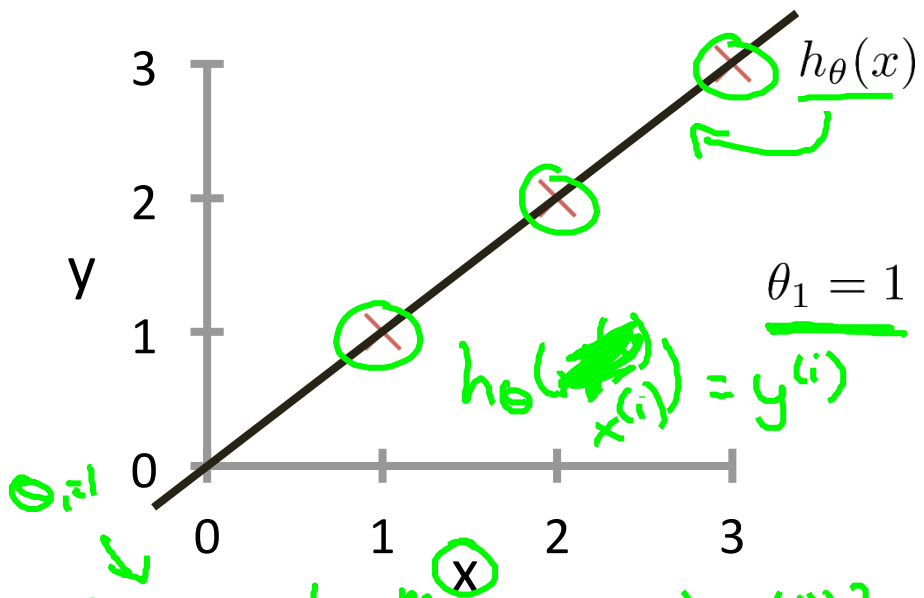
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$\theta_1 x^{(i)}$

$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

$h_\theta(x)$

(for fixed θ_1, this is a function of x)



$h_\theta(x^{(i)}) = y^{(i)}$

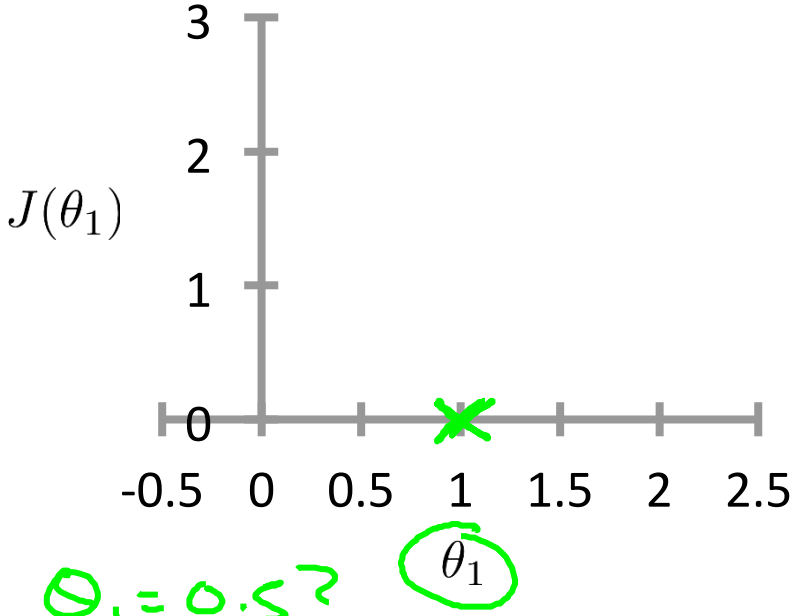
θ_1

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

$= \frac{1}{2m} \sum_{i=1}^{m} (\theta_1 x^{(i)} - y^{(i)})^2 = \frac{1}{2m}(0^2 + 0^2 + 0^2) = 0^2$

$J(\theta_1)$

(function of the parameter θ_1)

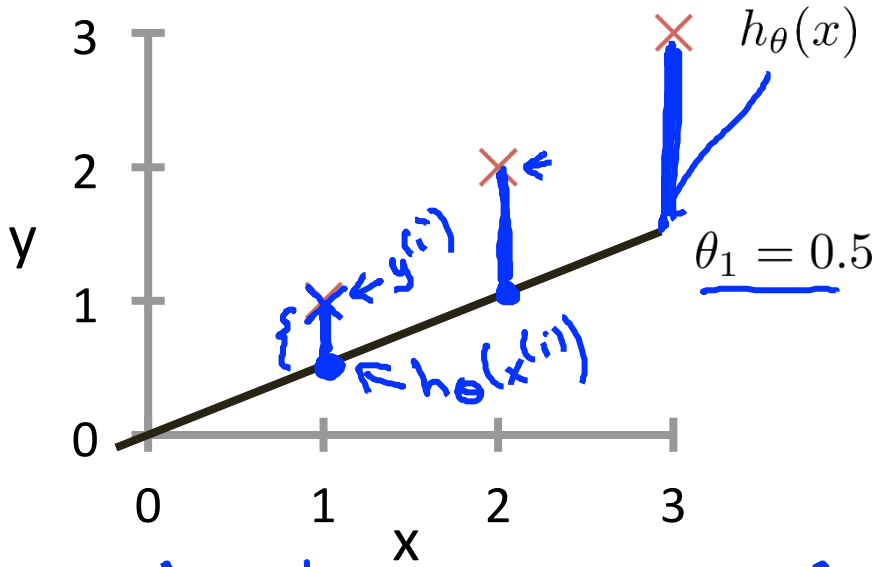


$\theta_1 = 0.5$?

$J(1) = 0$

$h_\theta(x)$

(for fixed θ_1, this is a function of x)

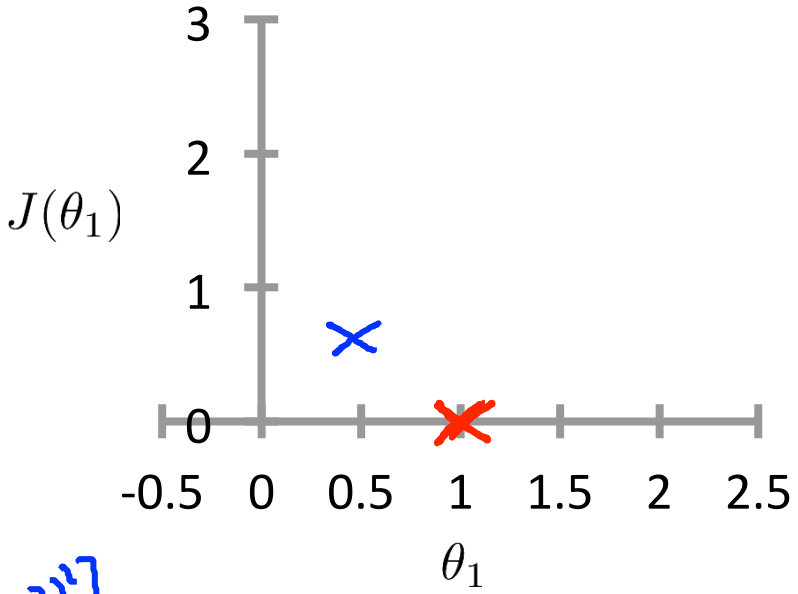


$$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$$

$$= \frac{1}{2\times 3}(3.5) = \frac{3.5}{6} \approx 0.58$$

$J(\theta_1)$

(function of the parameter θ_1)

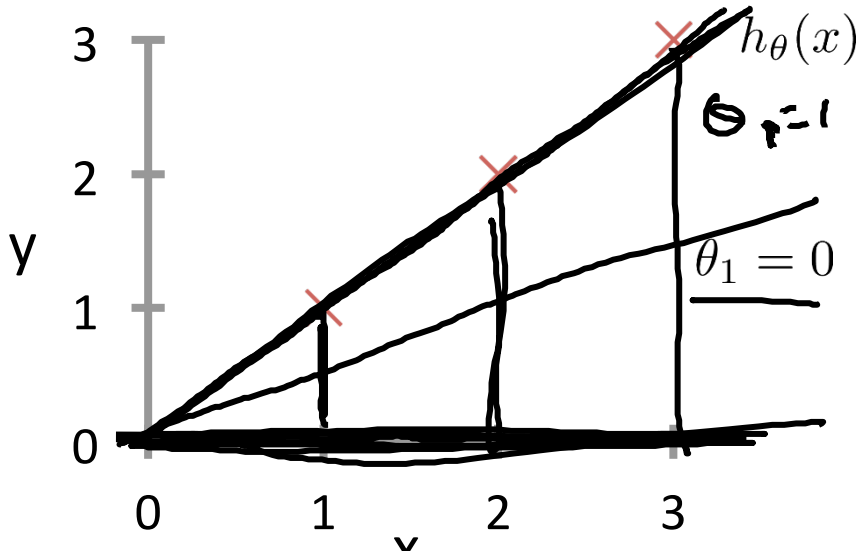


$\theta_1 = 0$?

$J(0) = ?$

$h_\theta(x)$

(for fixed θ_1, this is a function of x)

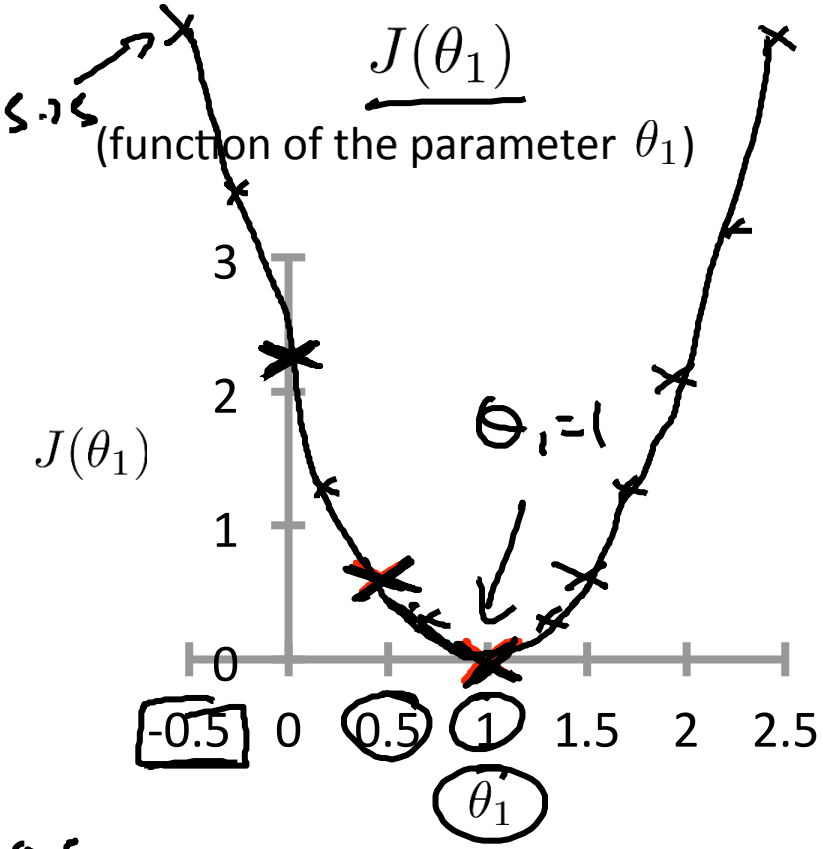


$J(\theta_1) = \frac{1}{2m}(1^2 + 2^2 + 3^2)$

$= \frac{1}{6} \cdot 14 \approx 2.3$

$J(\theta_1)$

(function of the parameter θ_1)



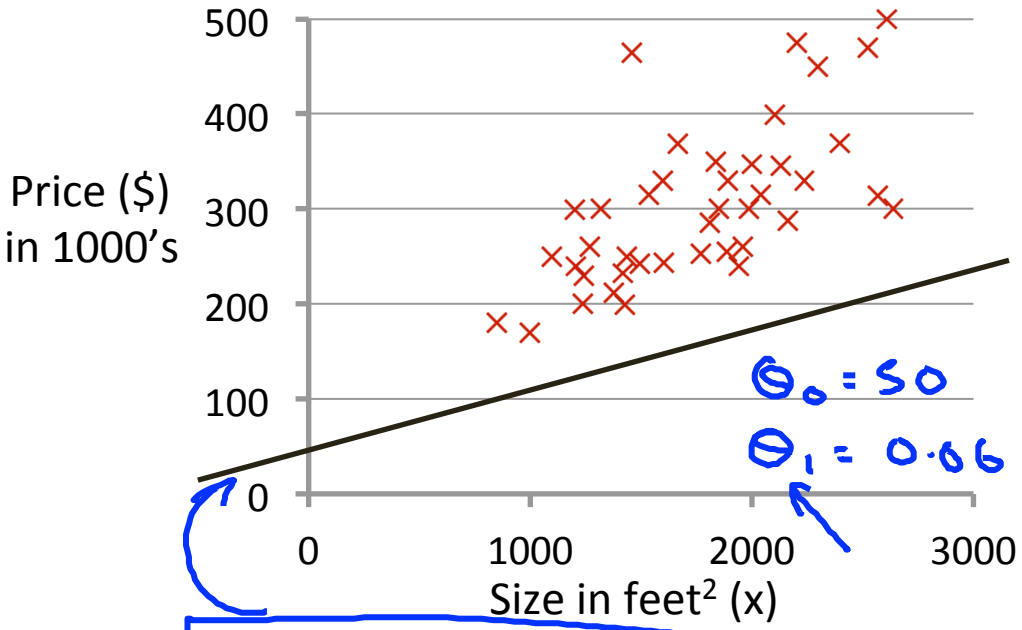
$h(x) = -0.5 \times$

minimize $J(\theta_1)$

θ_1

$h_\theta(x)$

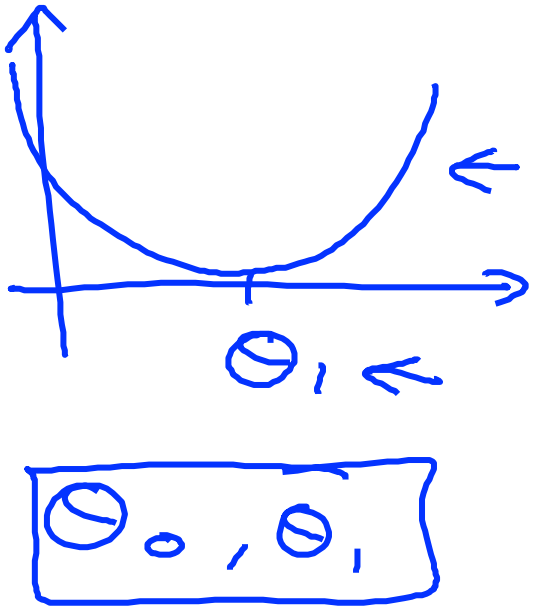
(for fixed θ_0, θ_1, this is a function of x)



$$h_\theta(x) = 50 + 0.06x$$

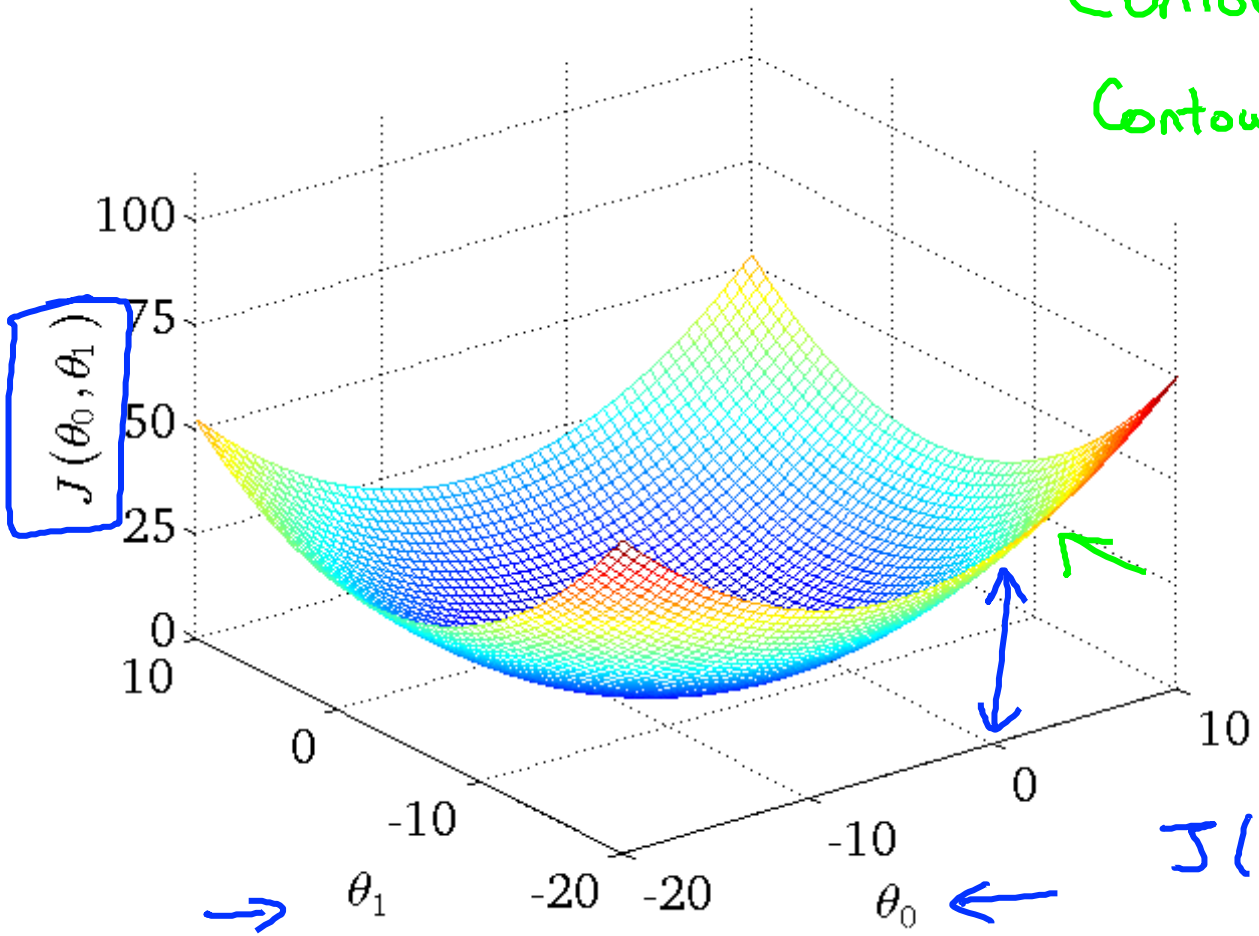
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Contour plots

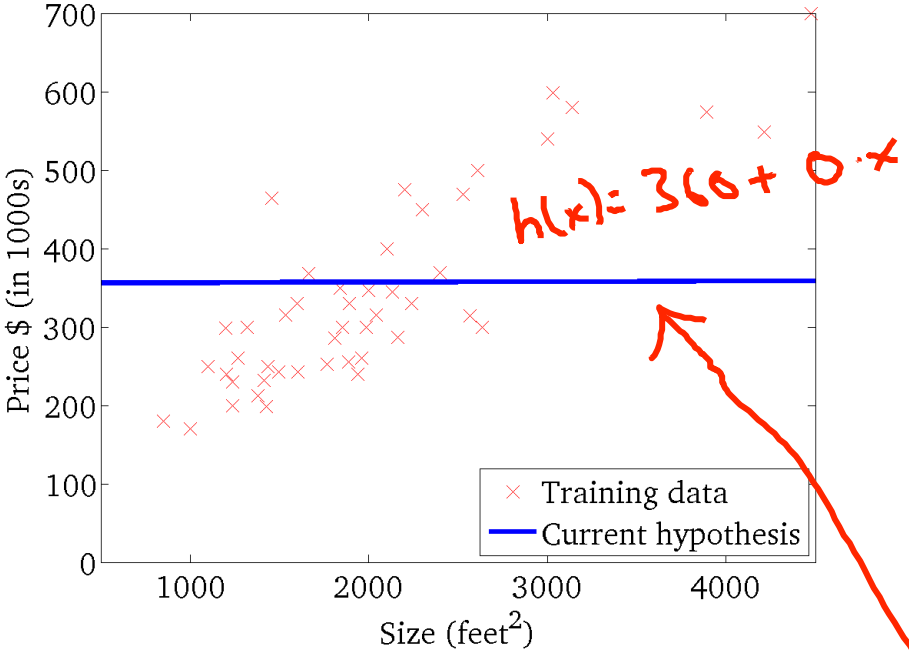
Contour figures -



$J(\theta_0, \theta_1)$

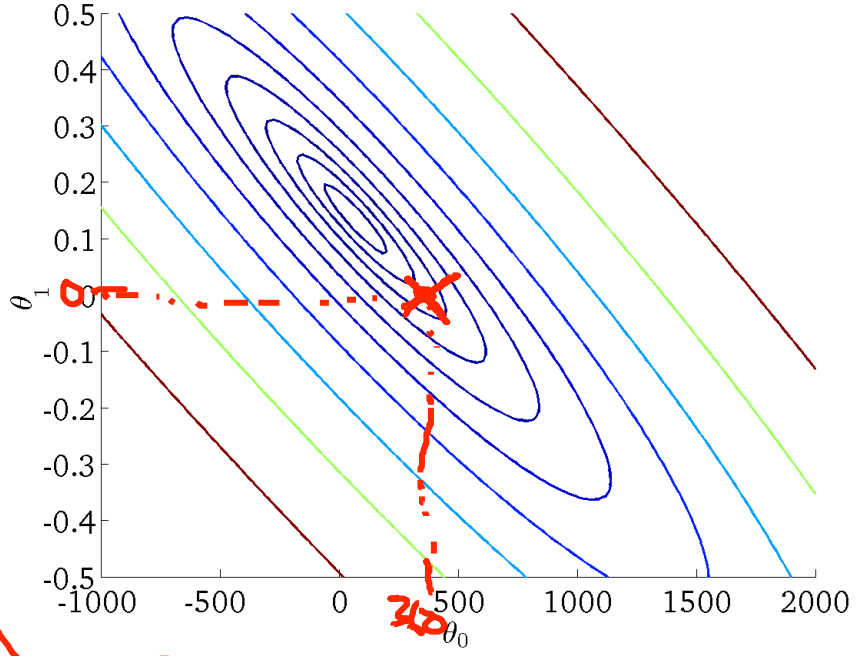
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

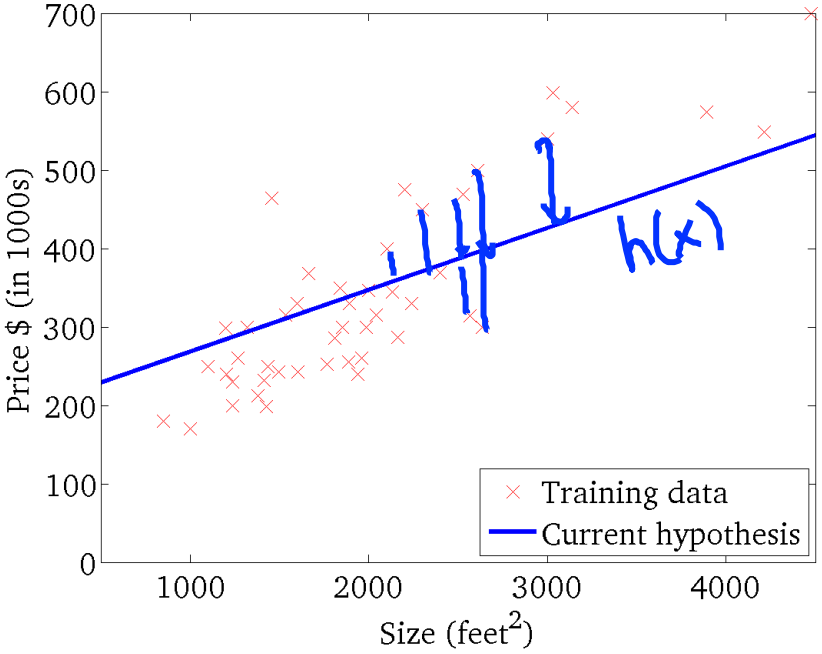


$\theta_0 = 360$

$\theta_1 = 0$

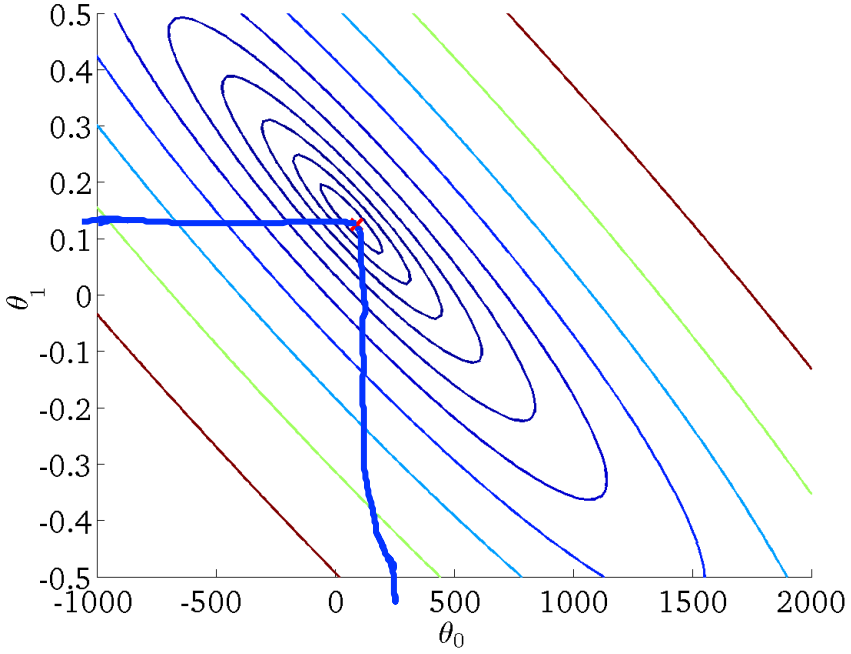
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



$$\text{argmin}_w \, Loss(w) = \text{argmin}_w \sum_{j=1}^{N} (w_0 + w_1 x^j - y^j)^2$$

$$0 = \frac{\partial}{\partial w_0} Loss(w_0, w_1) = 2 \sum_{j=1}^{N} (w_0 + w_1 x^j - y^j)$$

$$0 = \frac{\partial}{\partial x_1} Loss(w_0, w_1) = 2 \sum_{j=1}^{N} (w_0 + w_1 x^j - y^j) * x^j$$

$$w_0 = \frac{\sum y^j - w_1 \sum x^j}{N}$$

$$w_1 = \frac{N \sum x^j y^j - \sum x^j \sum y^j}{N \sum {x^j}^2 - (\sum x^j)^2}$$

Klesání podle gradientu

Klesání podle gradientu je metoda minimalizace funkce, pro niž v kterémkoli daném bodě umíme spočítat (nebo aproximovat) parciální derivaci.

Začneme s nějakými hodnotami (w_0, w_1).
V každém kroku současně provedeme tyto úpravy:

$$temp0 := w_0 - \alpha \frac{\partial}{\partial w_0} Loss(w_0, w_1)$$

$$temp1 := w_1 - \alpha \frac{\partial}{\partial w_1} Loss(w_0, w_1)$$

$$w_0 := temp0 \qquad w_1 := temp1$$

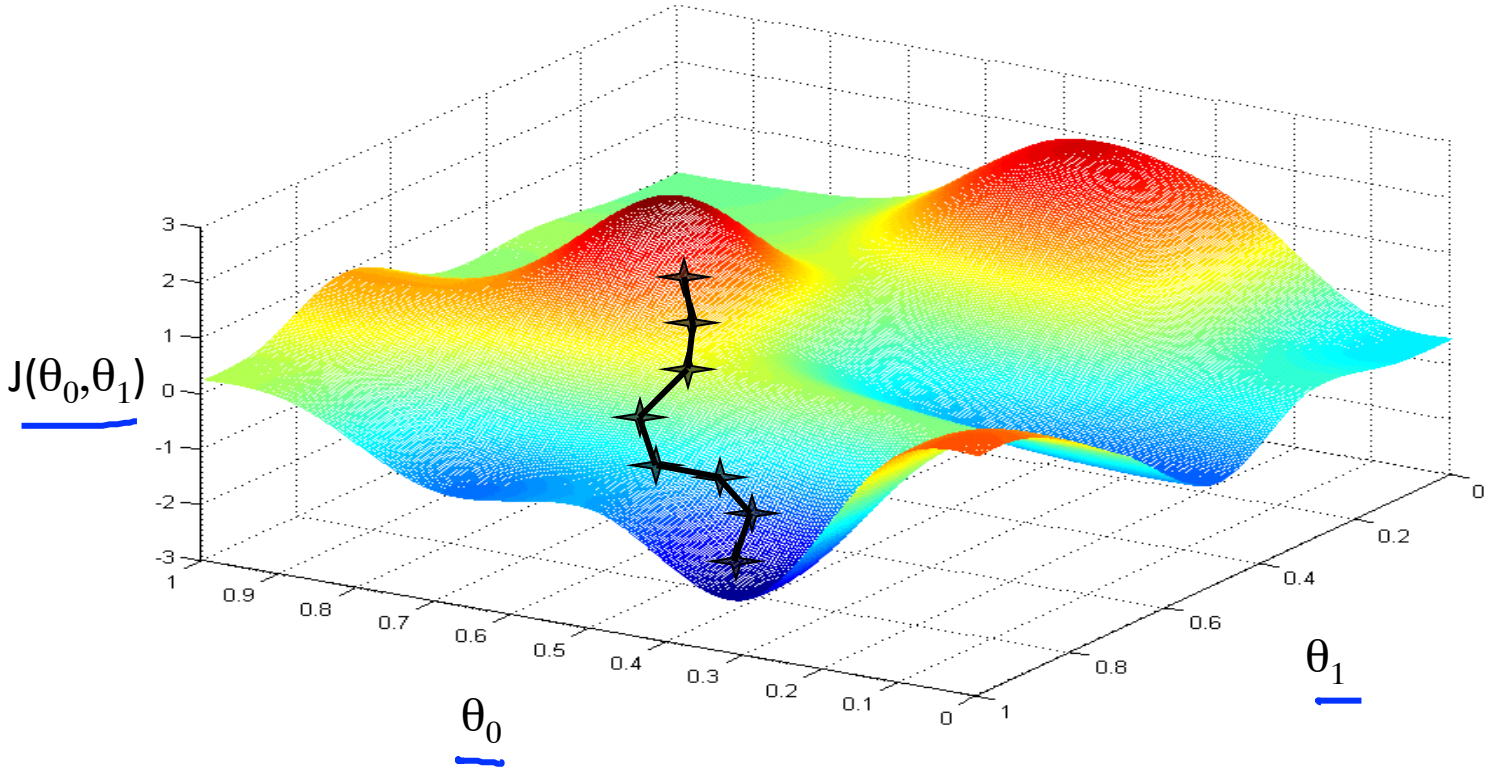
`http://www.onmyphd.com/?p=gradient.descent&ckattempt=1`

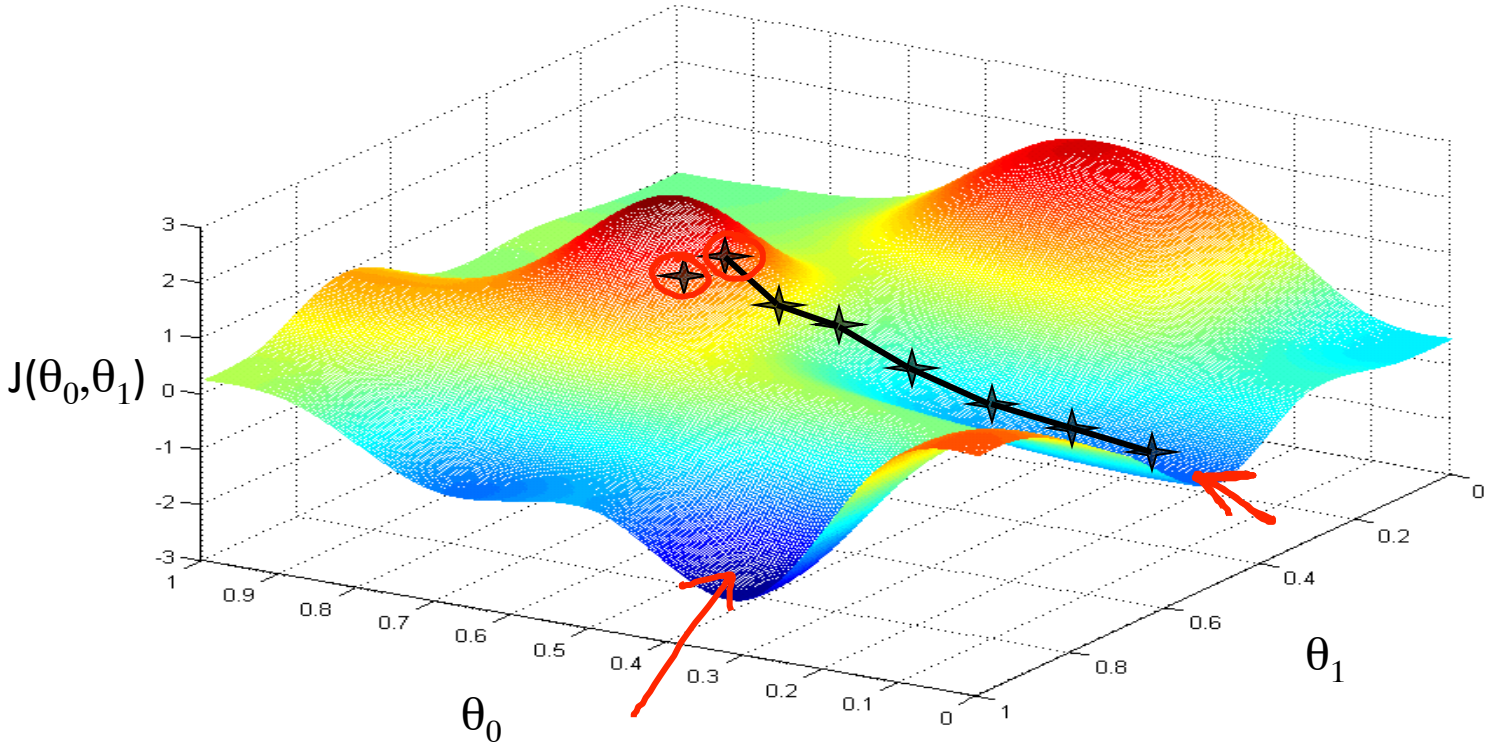
Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

Gradient descent algorithm

Assignment: $a := b$, $a := a+1$

Truth assertion: $a = b$, $a = a+1$ ✗

repeat until convergence {

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

}

θ_0, θ_1

learning rate

Simultaneously update θ_0 and θ_1

Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Incorrect:

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$

Gradient descent algorithm

repeat until convergence {

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

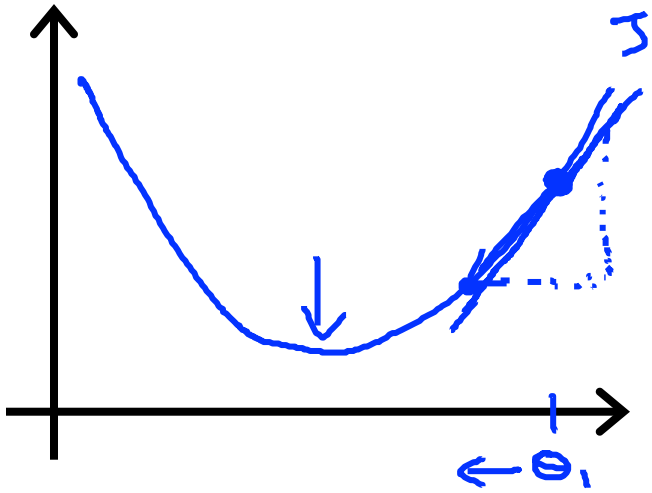
(simultaneously update $j = 0$ and $j = 1$)

}

learning rate

derivative

$\min_{\theta_1} J(\theta_1)$ $\quad \theta_1 \in \mathbb{R}$.

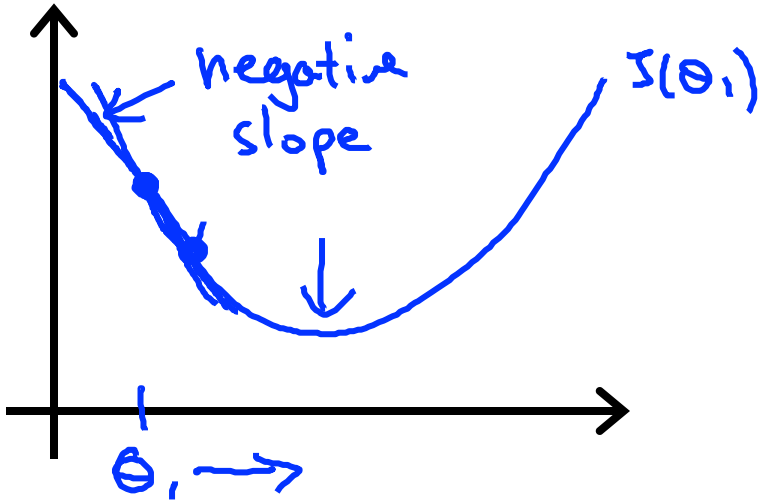


$(\theta_1 \in \mathbb{R})$

$\frac{d}{d\theta_1} J(\theta_1) \geq 0$

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

$\theta_1 := \theta_1 - \alpha \cdot (\text{positive number})$

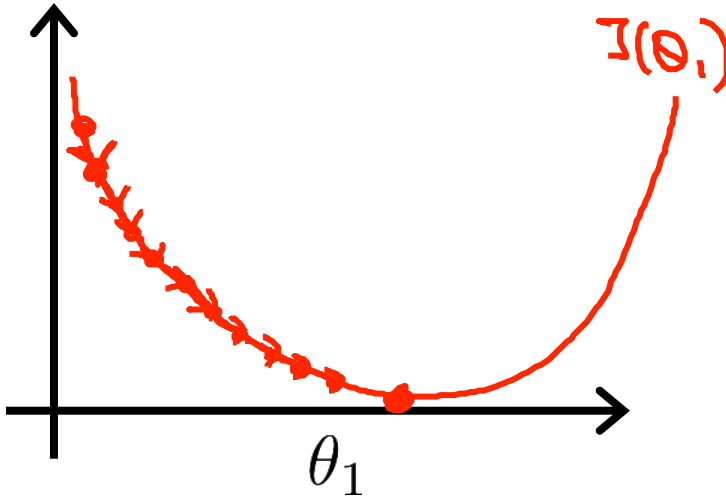


$\frac{d}{d\theta_1} J(\theta_1) \leq 0$

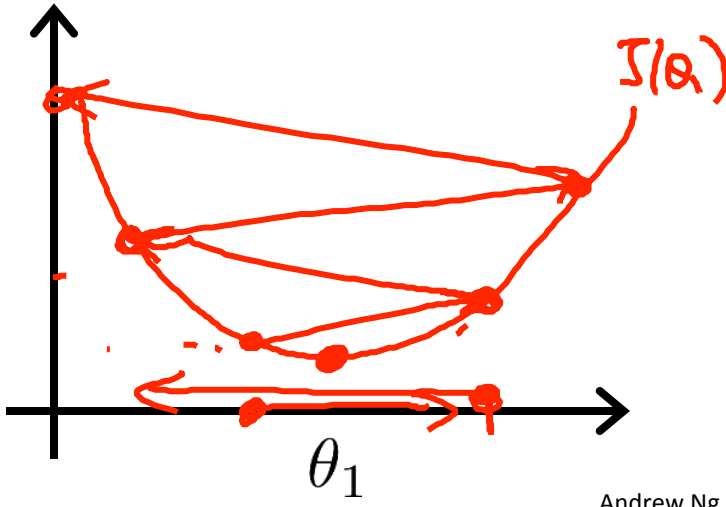
$\theta_1 := \theta_1 - \alpha \, (\text{negative number})$

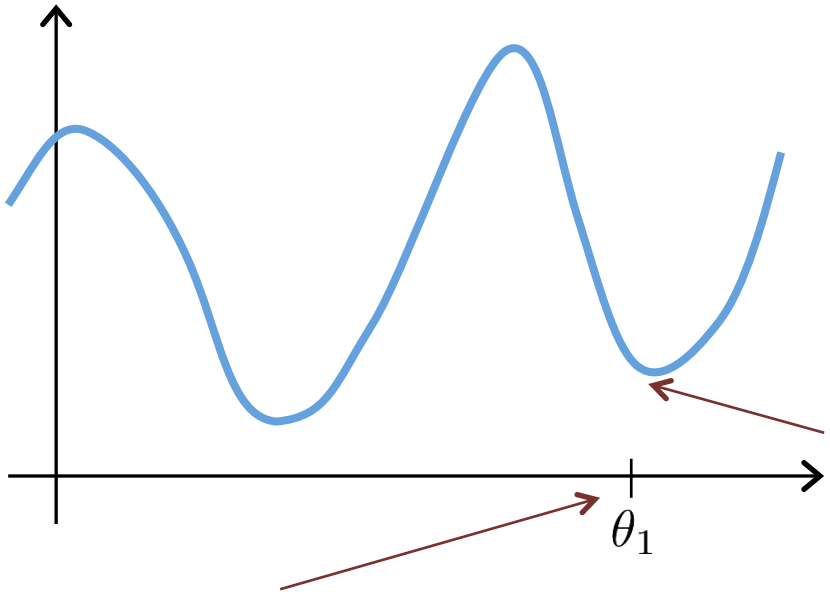
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

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



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





θ_1 at local optima

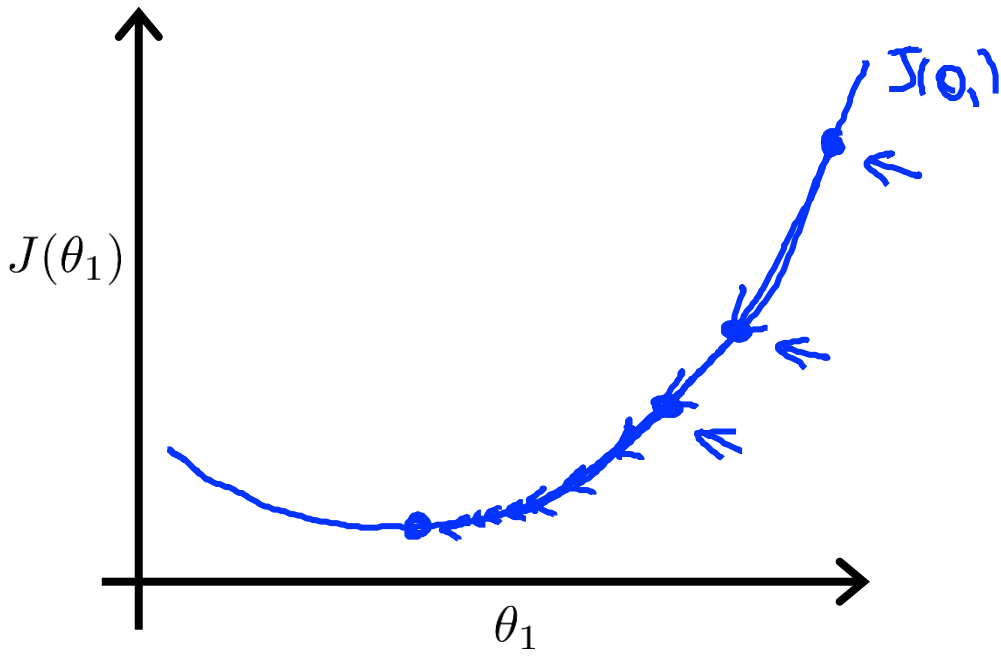
Current value of θ_1

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

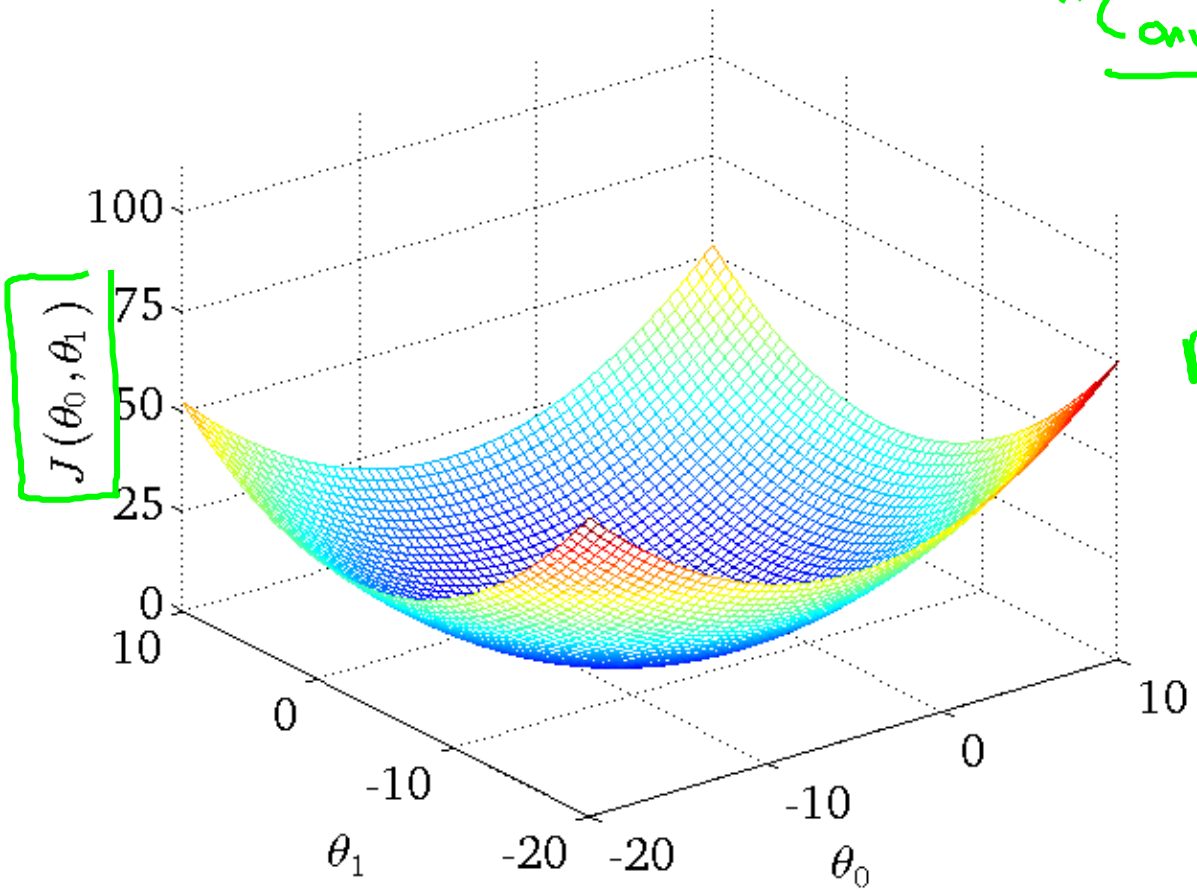
Gradient descent can converge to a local minimum, even with the learning rate α fixed.

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

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



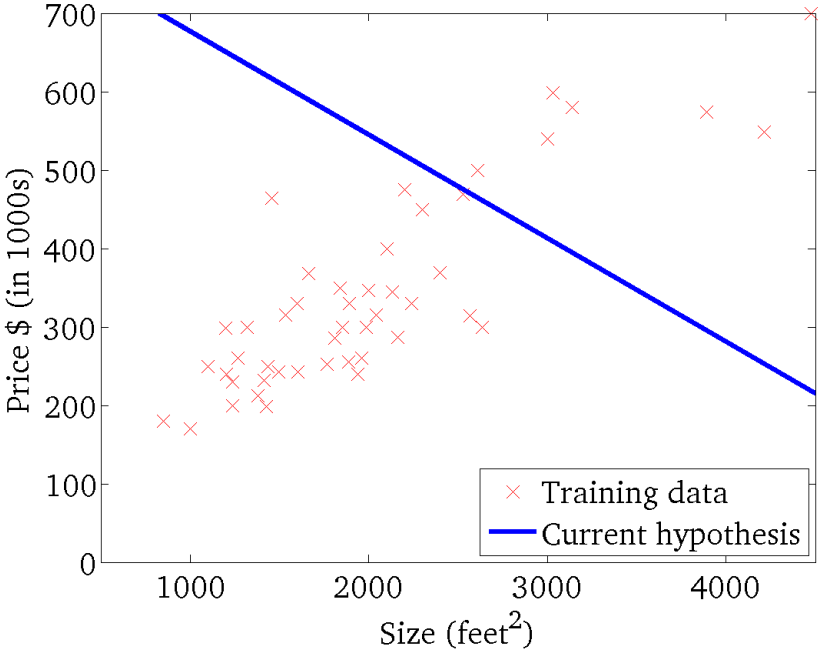
"Convex function"



Bowl-shaped

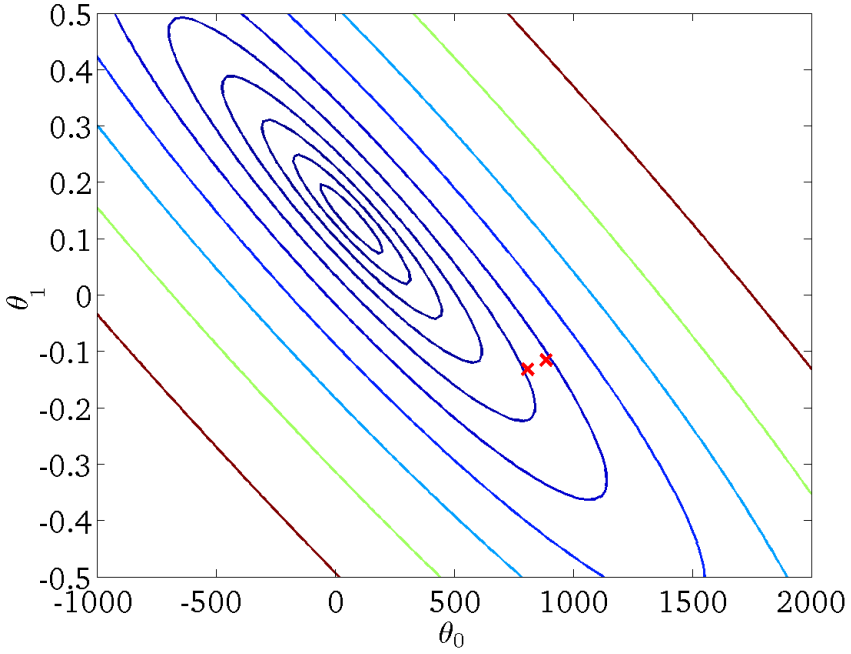
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



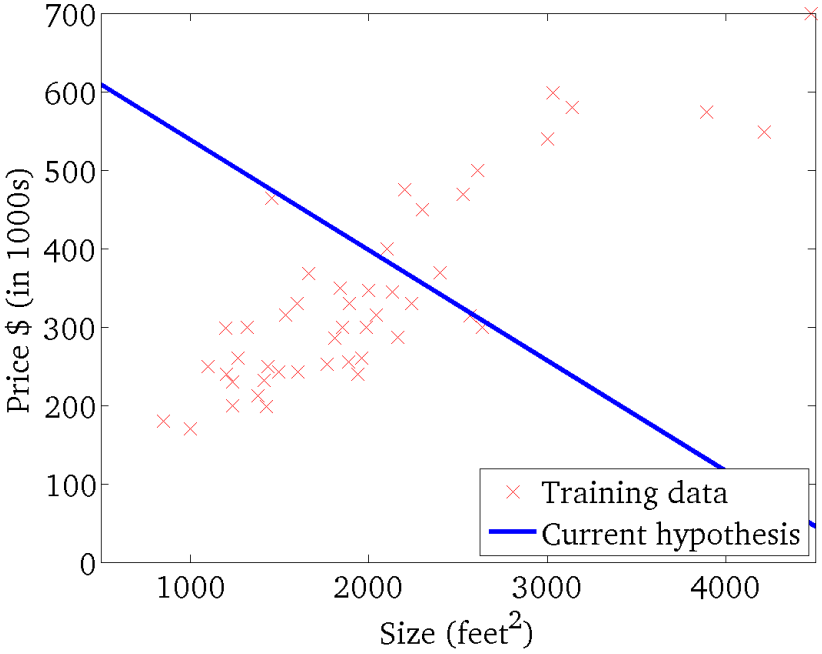
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



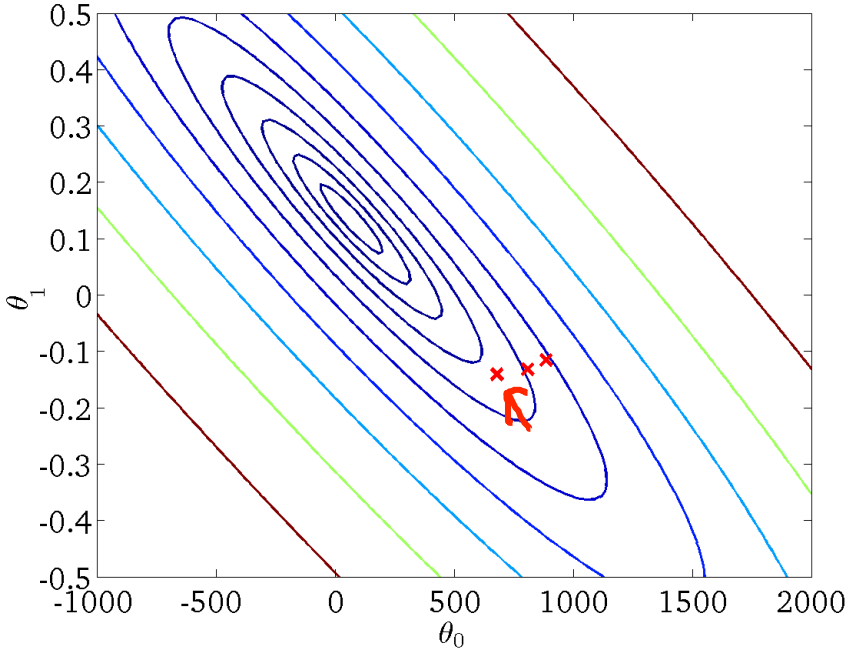
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



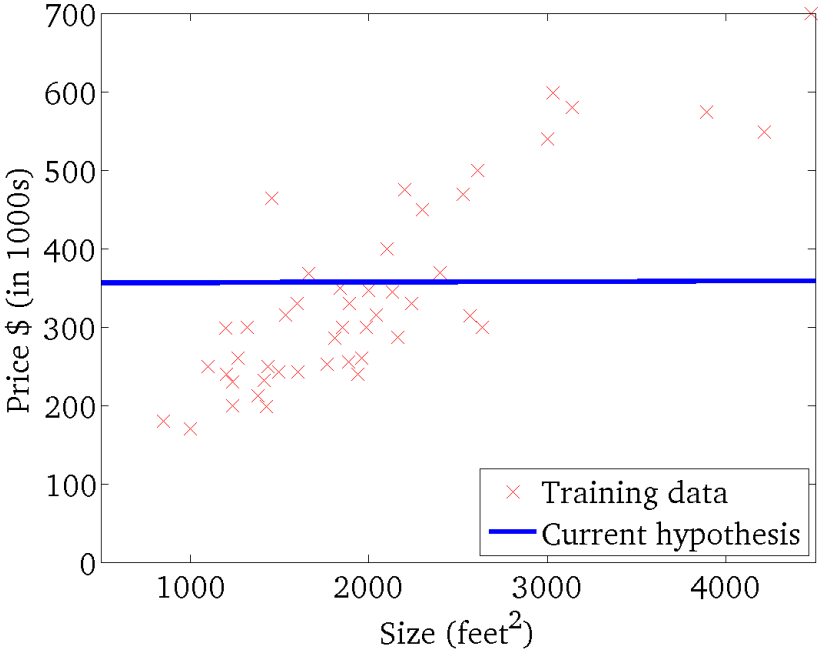
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



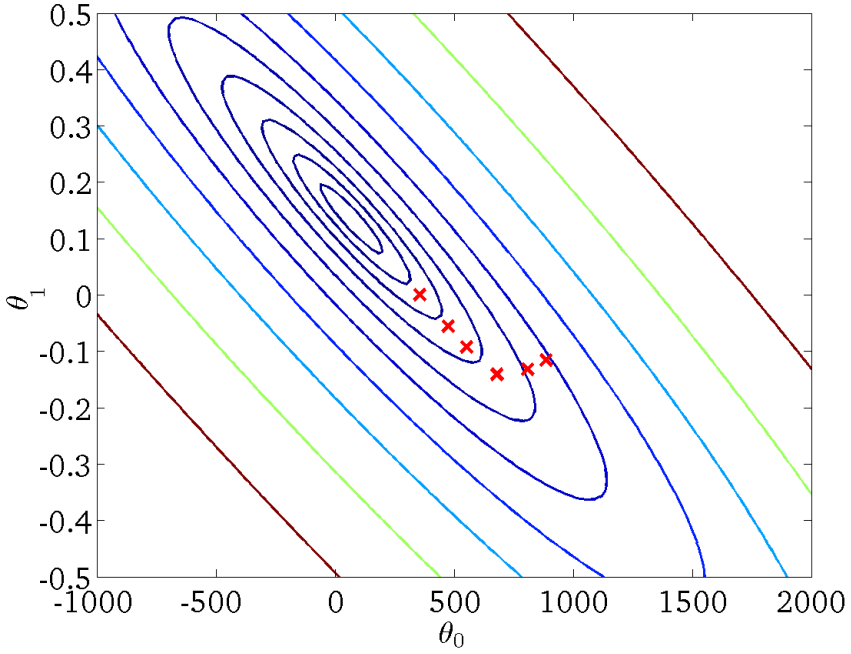
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



“Batch” Gradient Descent

“Batch”: Each step of gradient descent uses all the training examples.

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

Přeučení

- přeučení (overfitting)
  - příliš silný model sice má velmi malou chybu na trénovacích datech, ale velkou chybu na datech, která předtím neviděl
  - např. máme-li 18 trénovacích instancí, umíme jimi proložit polynom stupně 17 tak, že chybová funkce bude 0, ale pokud se ve skutečnosti jedná o data s lineární závislostí (a chybou měření), bylo by lepší použít lineární funkci
- nedoučení (underfitting)
  - příliš slabý model nemá dostatečnou expresivitu na to, aby vystihl zákonitosti v datech

Volba typu/síly modelu: křížová validace

problém: jak určit, jak silný model mám zvolit?

- hodnota chybové funkce na trénovacích datech s rostoucí expresivitou modelu klesá
- $\rightarrow$ proto si část dat necháme stranou (**testovací data**); konkrétní parametry modelu trénujeme na zbylých trénovacích datech, natrénované modely testujeme na testovacích datech$\rightarrow$ vybereme model s nejmenší chybou na testovacích datech, čímž máme jistotu, že nedošlo k přeučení
- **křížová validace** (cross-validation): rozdělím data na desetiny, vždy na devíti desetinách natrénuji konkrétní parametry modelu a na poslední desetině testuji, jak dobře si natrénovaný model vede na datech, které ve fázi trénování neviděl$\rightarrow$ zvolím takovou sílu modelu, která dává nejnižší průměrnou chybu na testovacích datech

Metodologická poznámka

Pokud se chceme chlubit tím, jak dobře si náš model vede na neznámých datech, nesmíme k určení úspěšnosti modelu použít ani trénovací data, ani testovací data, na základě kterých jsme vybrali konkrétní model.Proto si ještě před započetím experimentů malou část dat, která mám k dispozici, dáme stranou, a na těch testujeme až náš nejlepší model (ten, kterým se chceme chlubit).