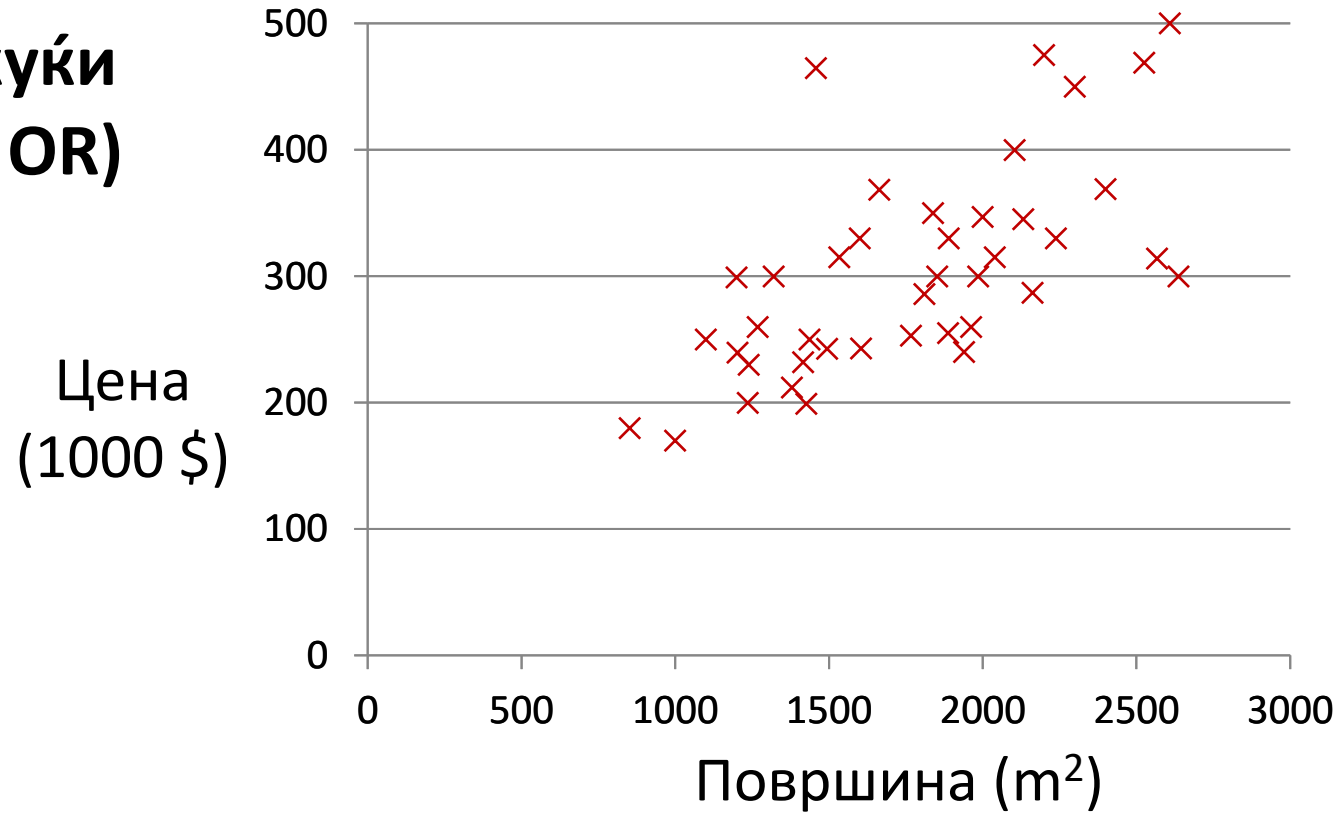
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

ЛИНЕАРНА РЕГРЕСИЈА

Машинско учење

Цени на куќи (Portland, OR)



Надгледувано учење

Даден е „точниот одговор“ за секој примерок од податоците

Проблем на регресија

Предвидување на излез со реална вредност

Множество за тренирање за цените на куќите (Portland, OR)

Површина во m^2 (x)	**Цена во 1000 $ (y)**
2104	460
1416	232
1534	315
852	178
…	…

Нотација:

m = Број на примероци за тренирање

x = влезни променливи (карактеристики)

y = излезни (целни) променливи

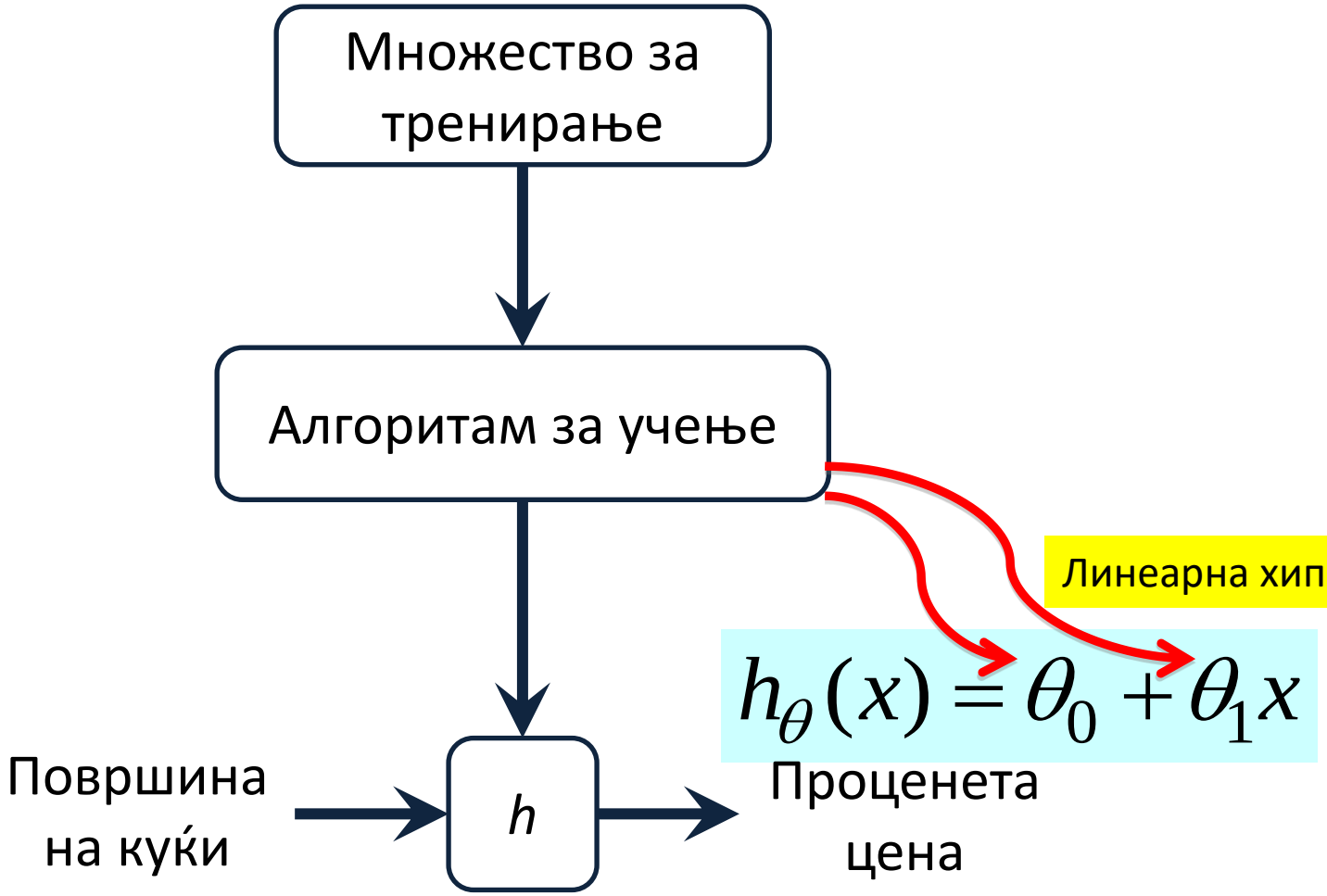
Множество за тренирање

Површина во m² (x)	Цена во 1000 $ (y)
2104	460
1416	232
1534	315
852	178
…	…

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

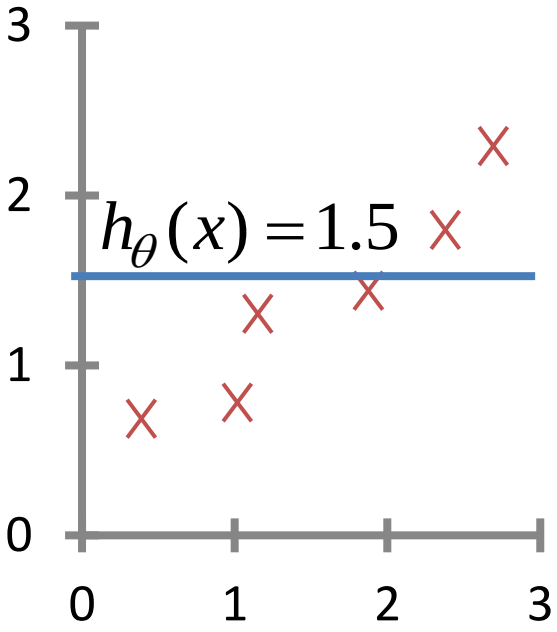
θ_i : Параметри

Како да се изберат θ_i ?



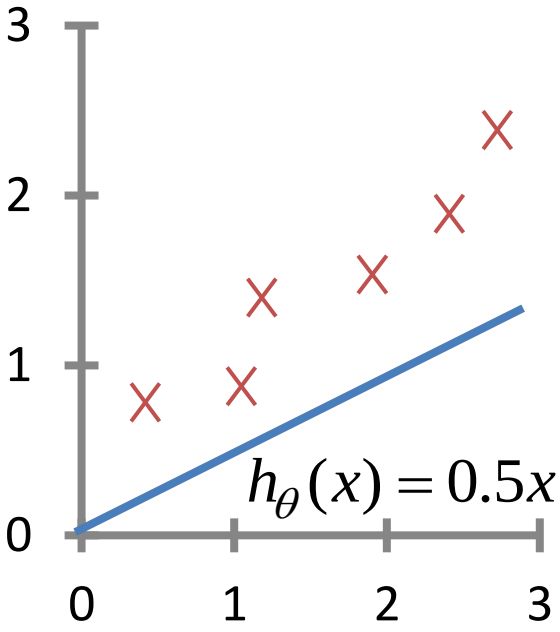
Линеарна регресија со една променлива (универијантна)

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



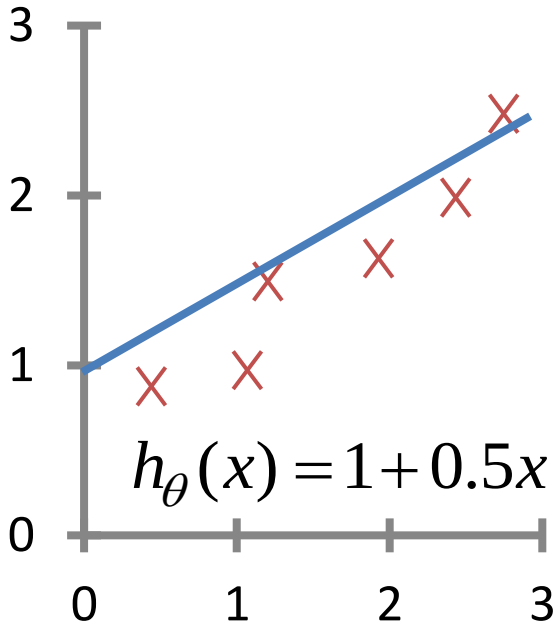
$\theta_0 = 1.5$

$\theta_1 = 0$



$\theta_0 = 0$

$\theta_1 = 0.5$



$\theta_0 = 1$

$\theta_1 = 0.5$

Идеја:

- Избери ги θ_0 ,θ_1 така што $h_\theta(x)$ е блиску до y за нашите примероци од множеството за тренирање
- Функција на цена на чинење (средна квадратна грешка)

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

$x^{(i)}, y^{(i)}$: Вредности на влезот и излезот на i-тиот примерок, соодветно

- Цел

$$\underset{\theta_0,\theta_1}{минимизирај}\ J(\theta_0,\theta_1)$$

Општ облик на линеарна регресија со една променлива

Хипотеза:

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

Параметри:

$$\theta_0, \theta_1$$

Функција на цена на чинење:

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

Цел:

$$\underset{\theta_0, \theta_1}{\textit{минимизирај}}\ J(\theta_0, \theta_1)$$

Поедноставен облик

$$\theta_0 = 0$$

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

$$\theta_1$$

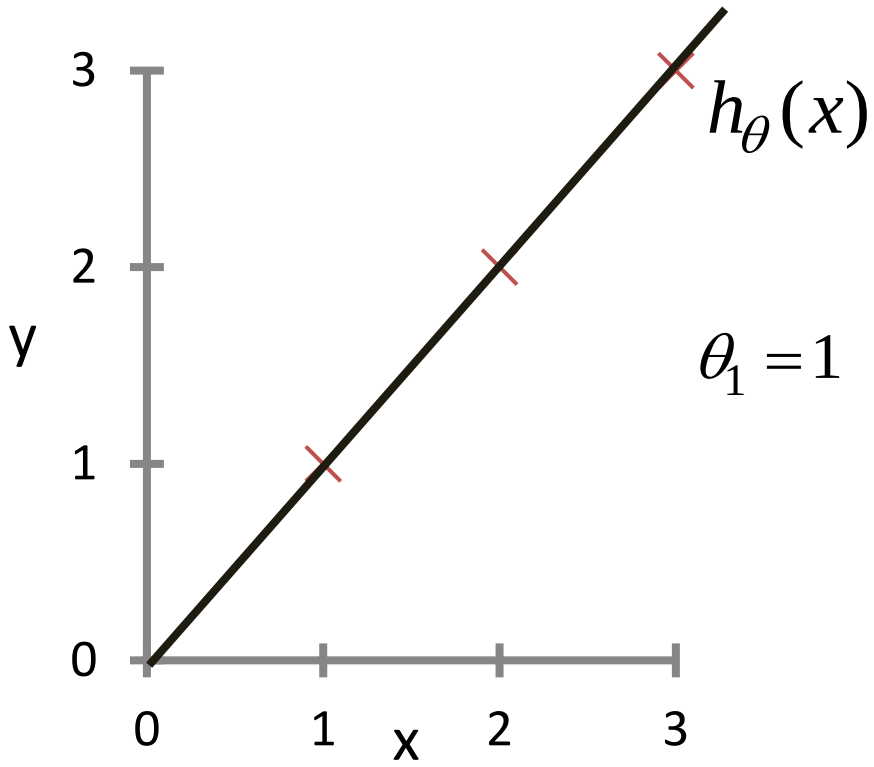
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\underset{\theta_1}{\textit{минимизирај}}\ J(\theta_1)$$

$\theta_0 = 0$

$h_\theta(x)$

(за фиксно θ_1 ова е функција од x)



$h_\theta(x) = x$

$J(\theta_1)$

(функција од параметарот θ_1)

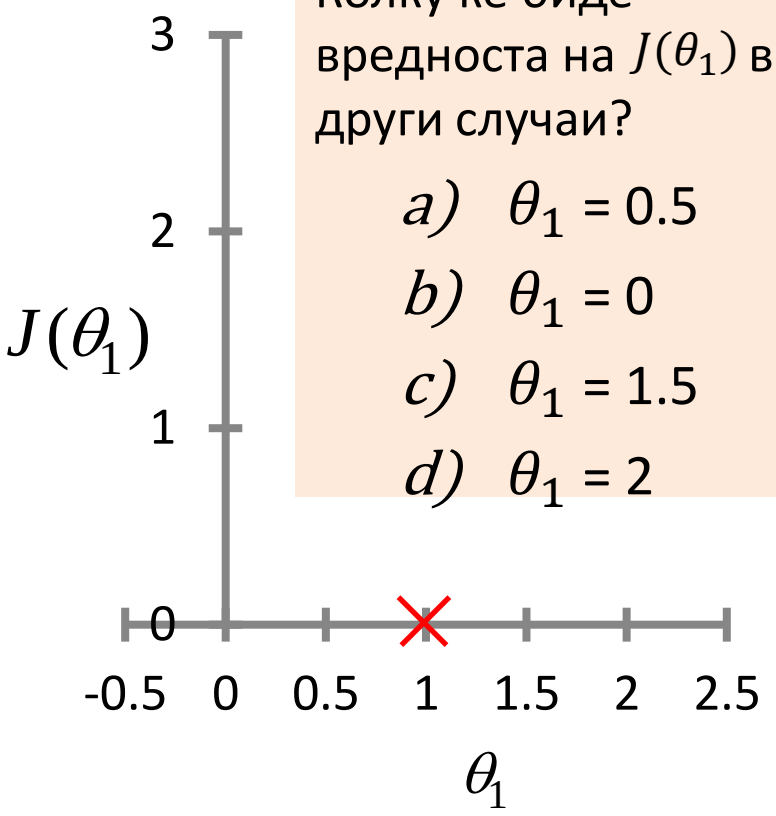
Колку ќе биде вредноста на $J(\theta_1)$ во други случаи?

a) $\theta_1 = 0.5$

b) $\theta_1 = 0$

c) $\theta_1 = 1.5$

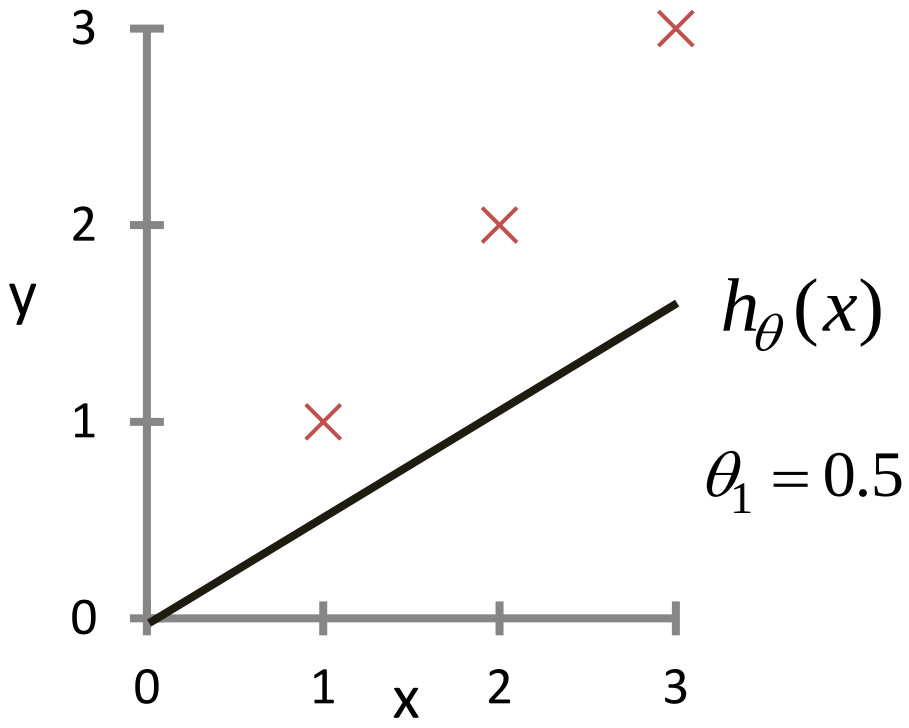
d) $\theta_1 = 2$



$\theta_0 = 0$

$h_\theta(x)$

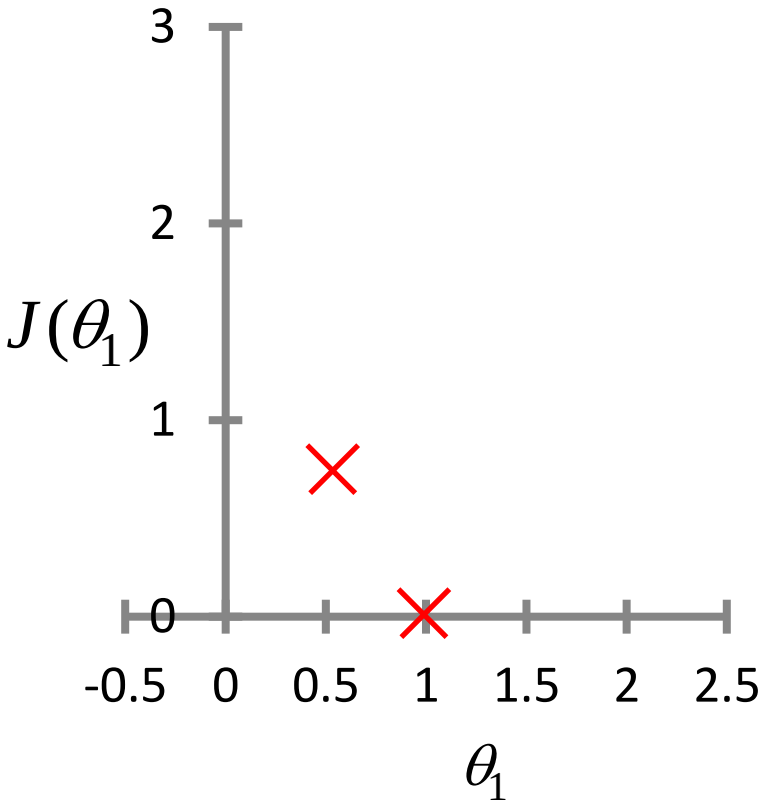
(за фиксно θ_1 ова е функција од x)



$h_\theta(x) = 0.5x$

$J(\theta_1)$

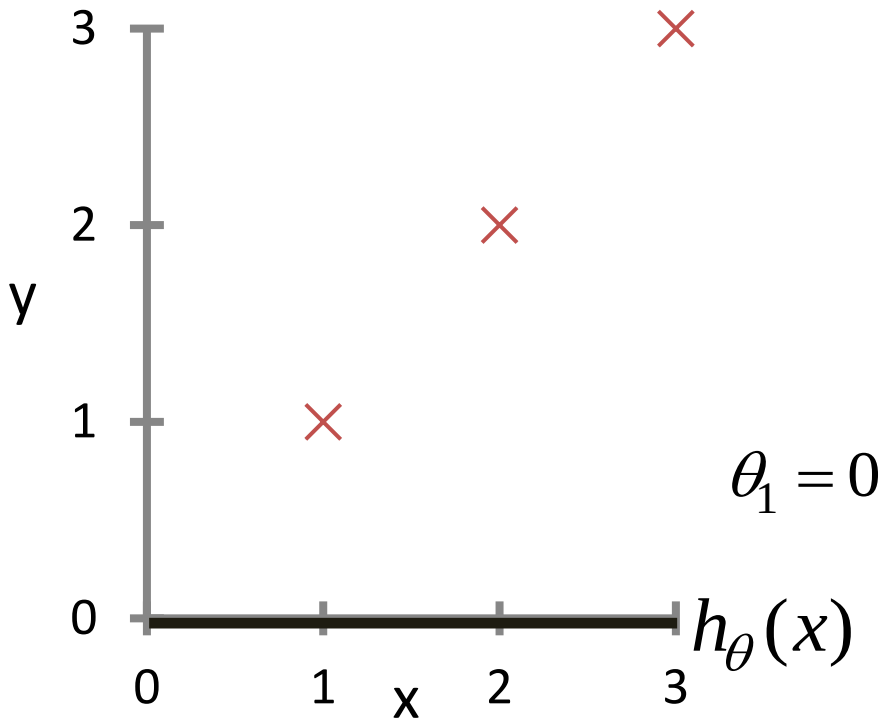
(функција од параметарот θ_1)



$\theta_0 = 0$

$h_\theta(x)$

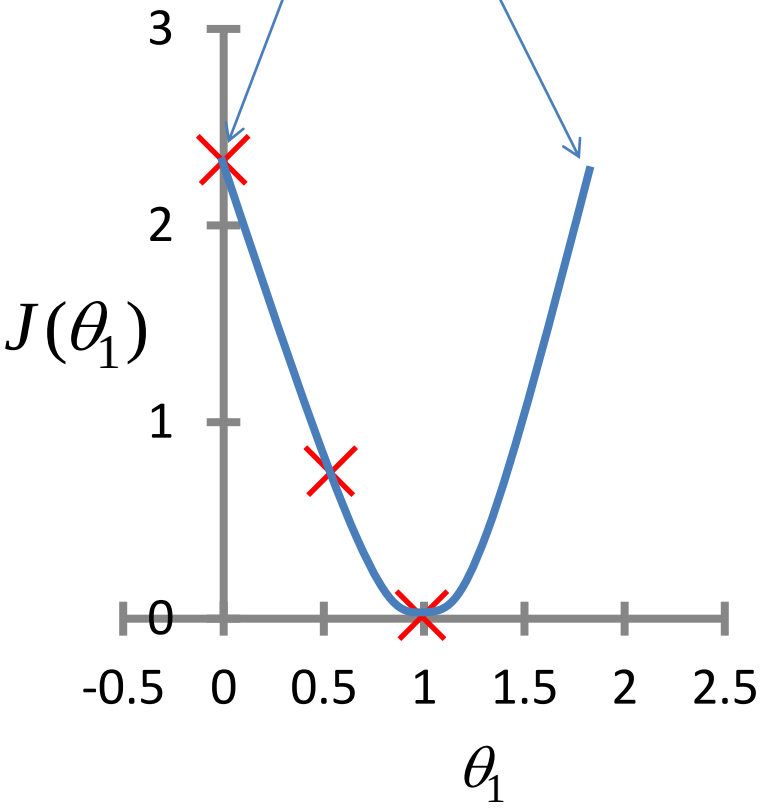
(за фиксно θ_1 ова е функција од x)



$h_\theta(x) = 0$

$J(\theta_1)$

(функција од параметарот θ_1)



Општ облик на линеарна регресија со една променлива

Хипотеза:

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

Параметри:

$$\theta_0, \theta_1$$

Функција на цена на чинење:

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

Цел:

$$\underset{\theta_0, \theta_1}{\textit{минимизирај}}\ J(\theta_0, \theta_1)$$

Поедноставен облик

$$\theta_0 = 0$$

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

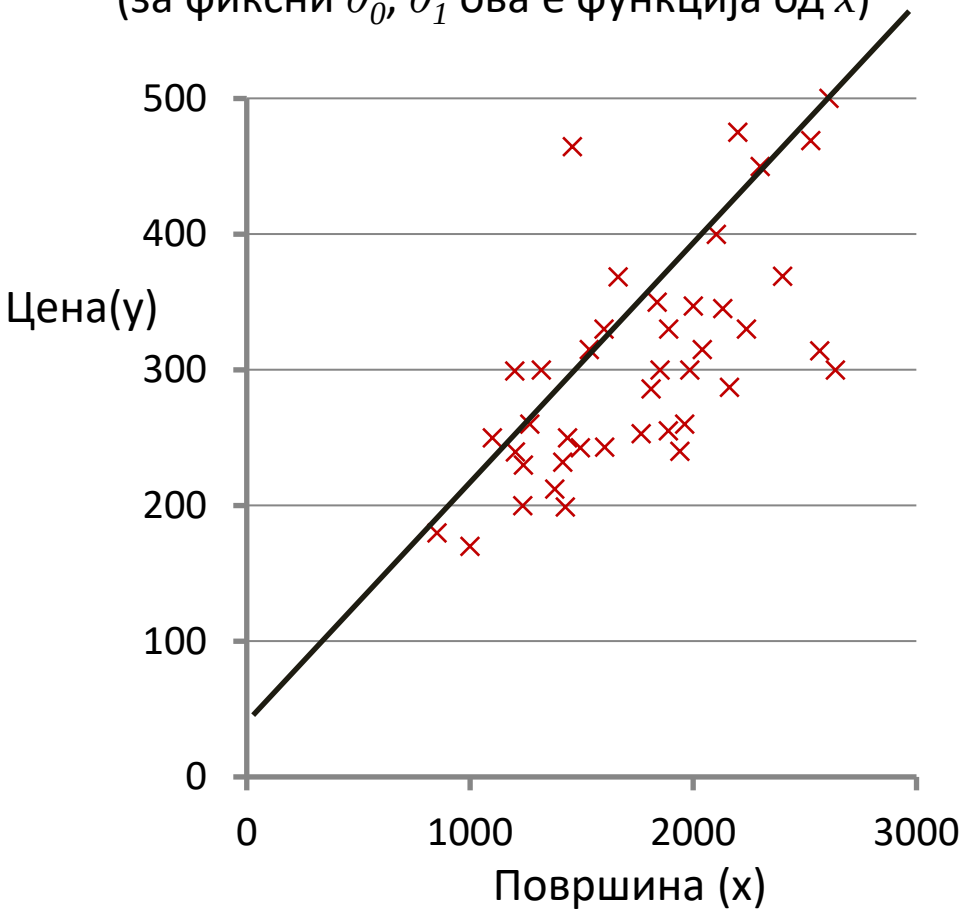
$$\theta_1$$

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

$$\underset{\theta_1}{\textit{минимизирај}}\ J(\theta_1)$$

$h_\theta(x)$

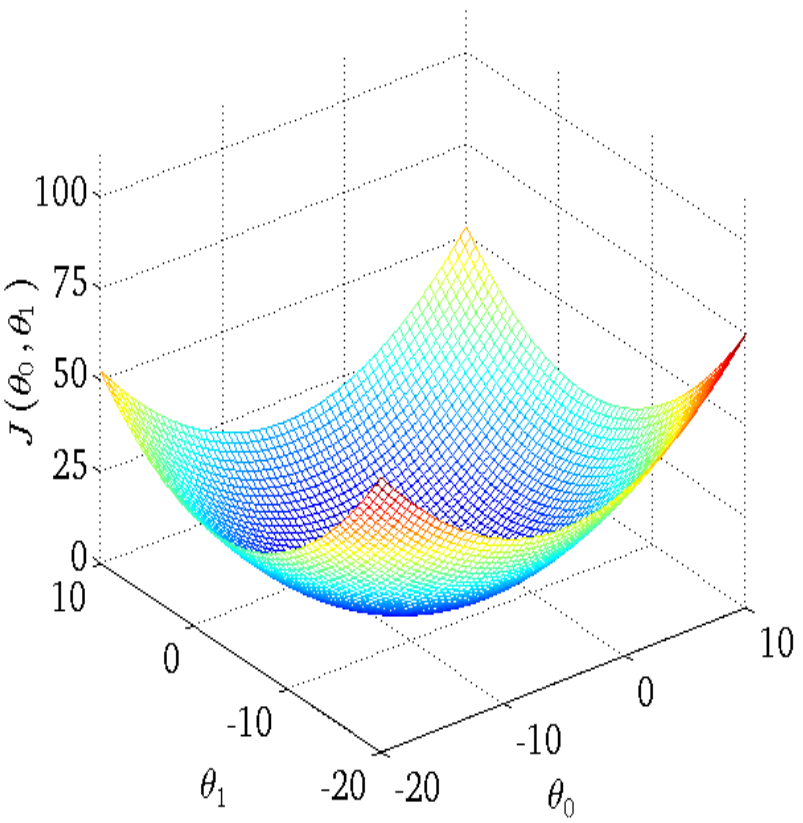
(за фиксни θ_0, θ_1 ова е функција од x)



$$h_\theta(x) = 50 + \frac{1}{6}x$$

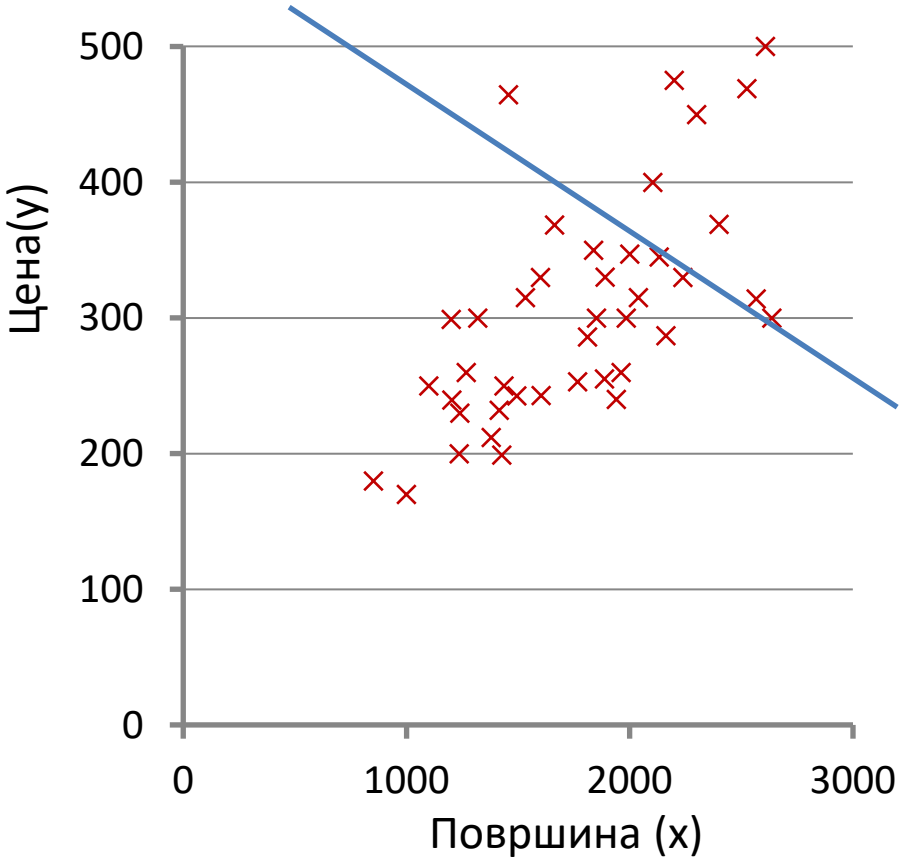
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



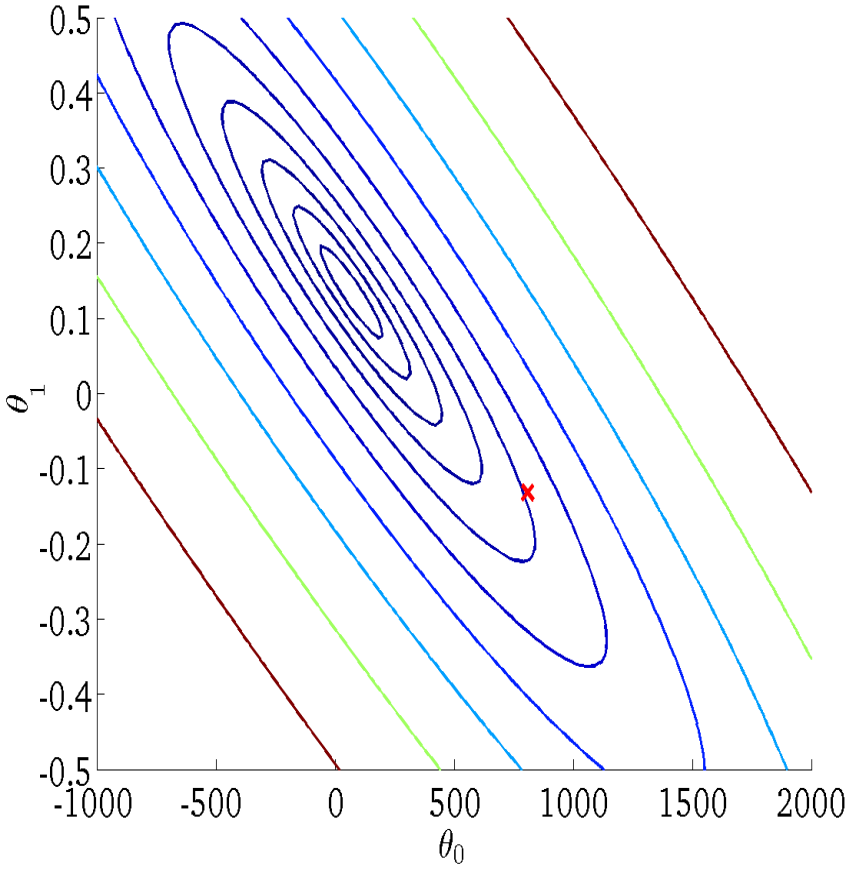
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



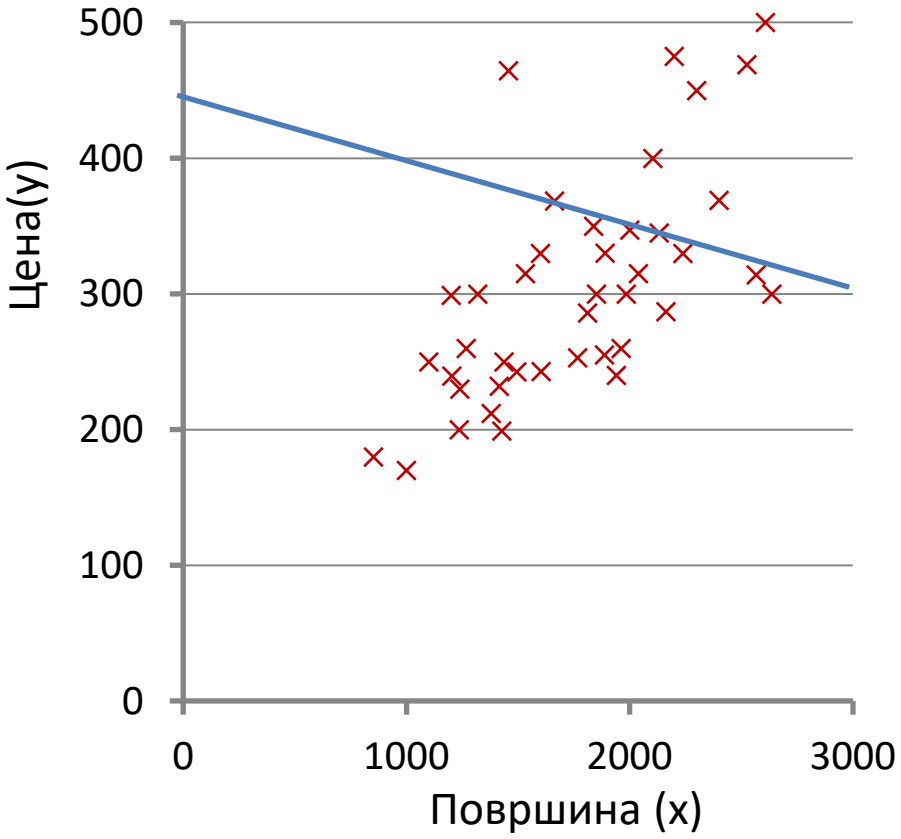
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



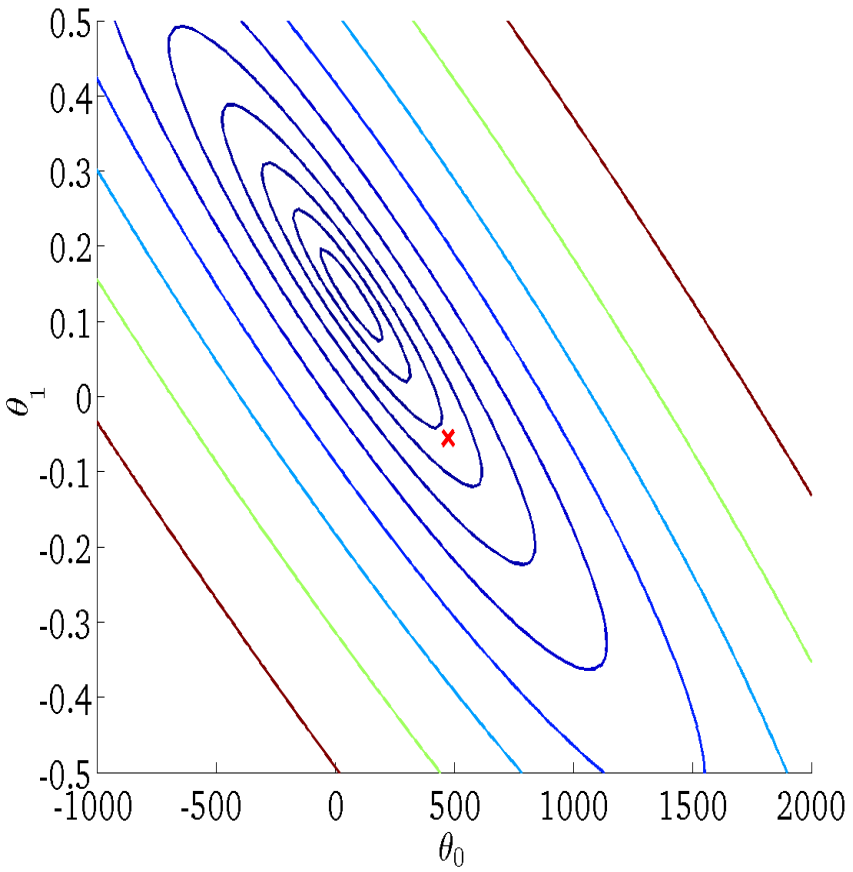
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



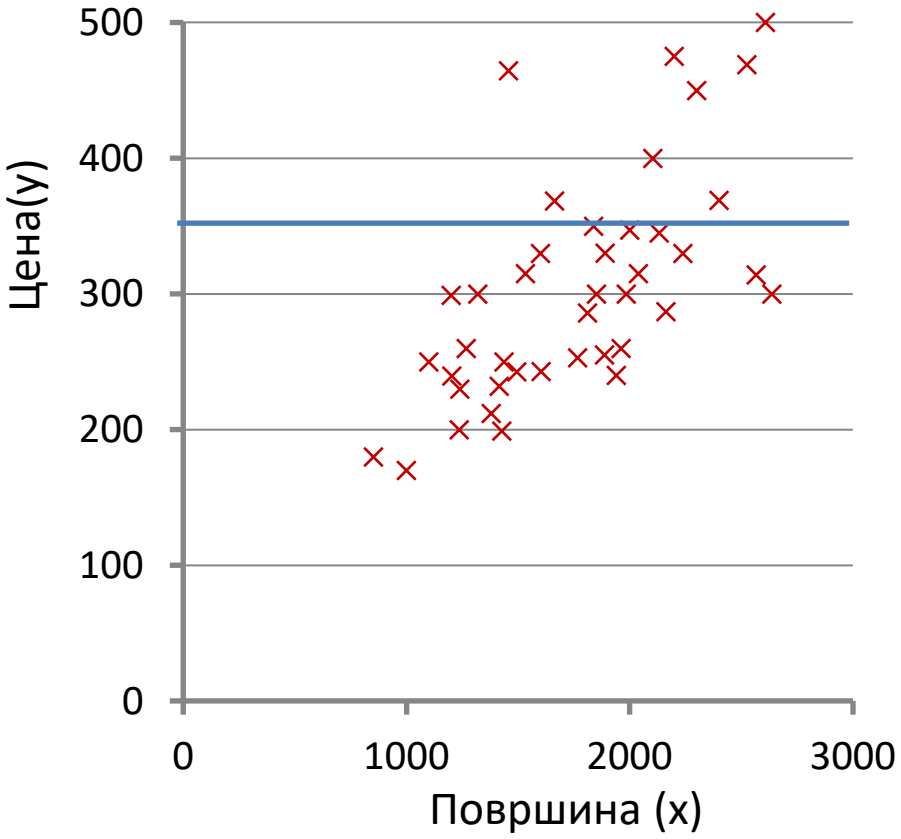
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



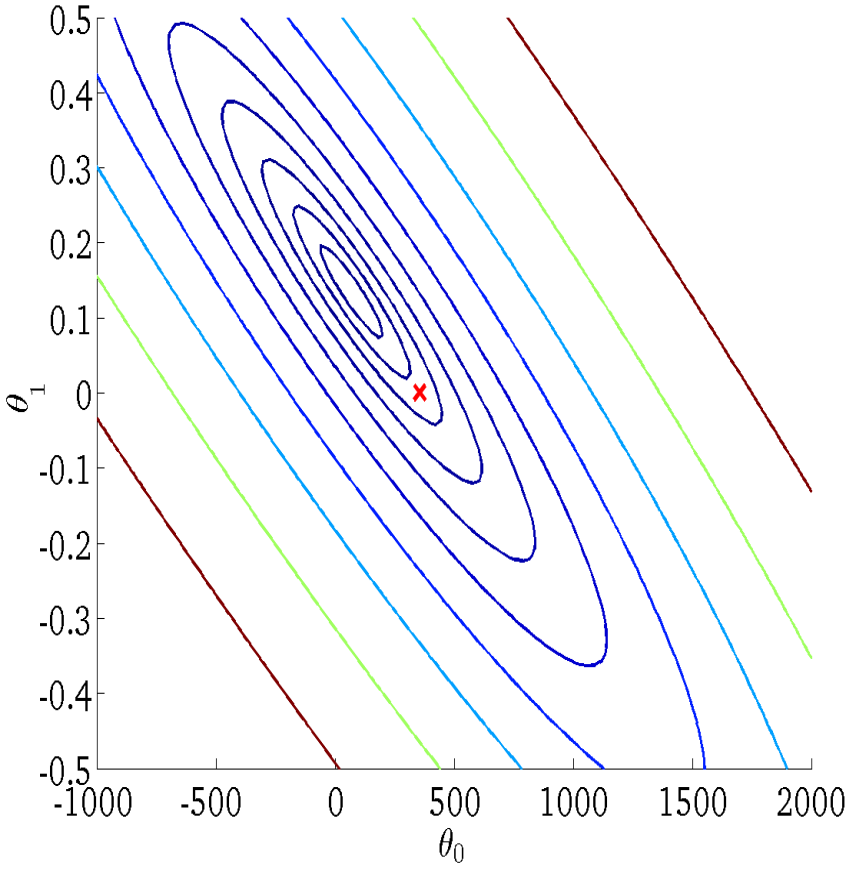
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



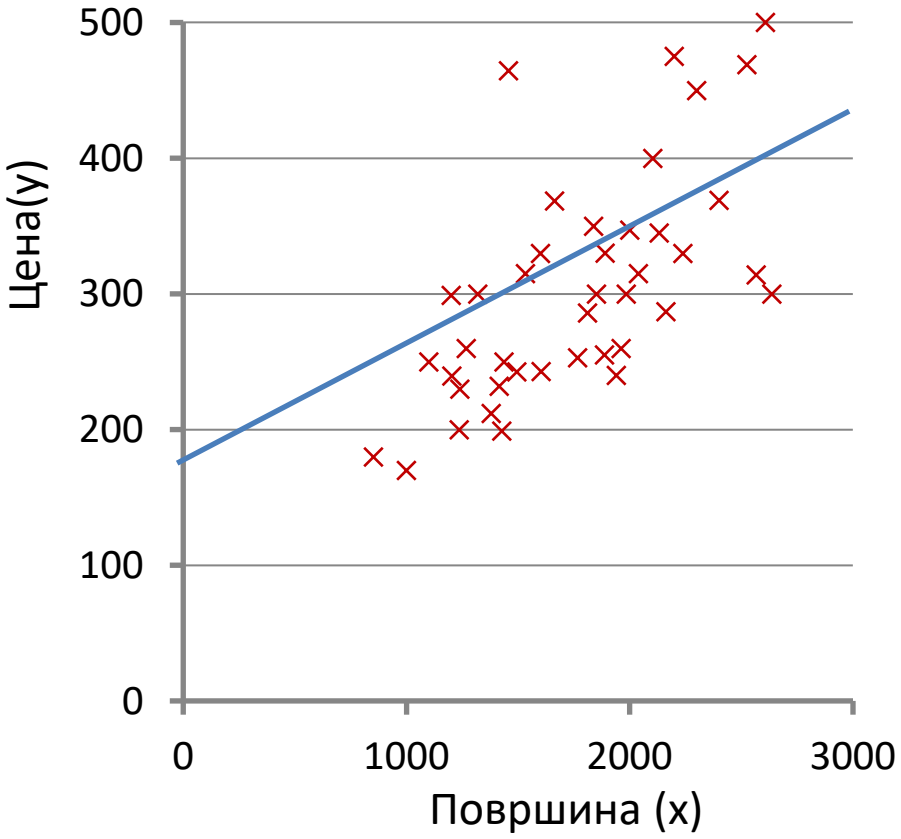
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



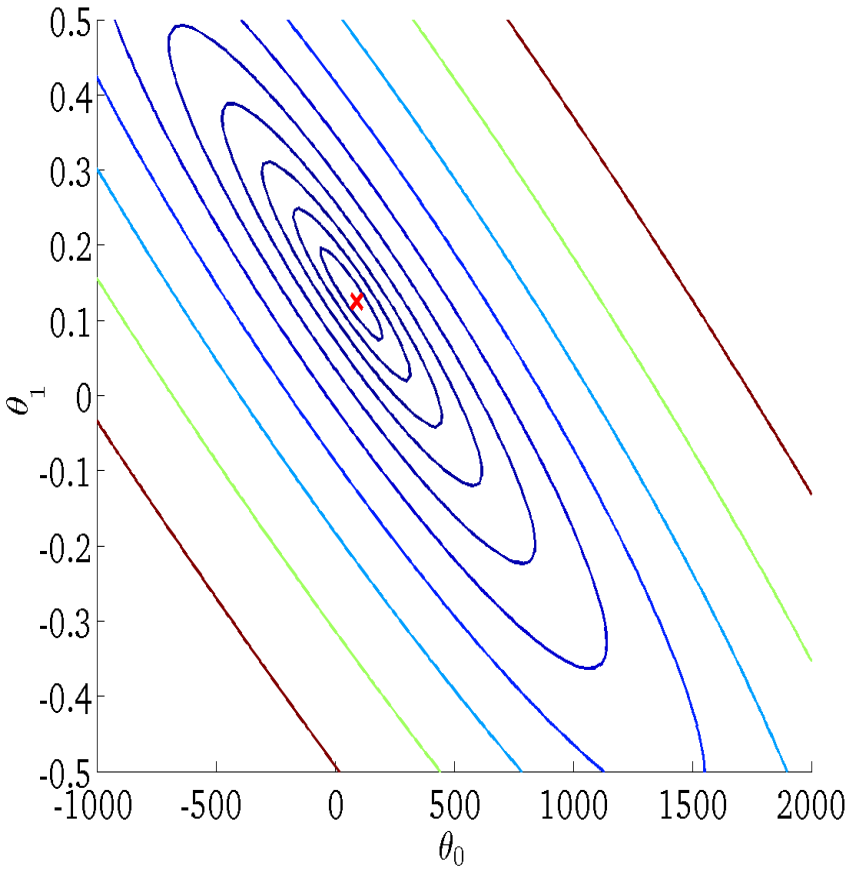
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)

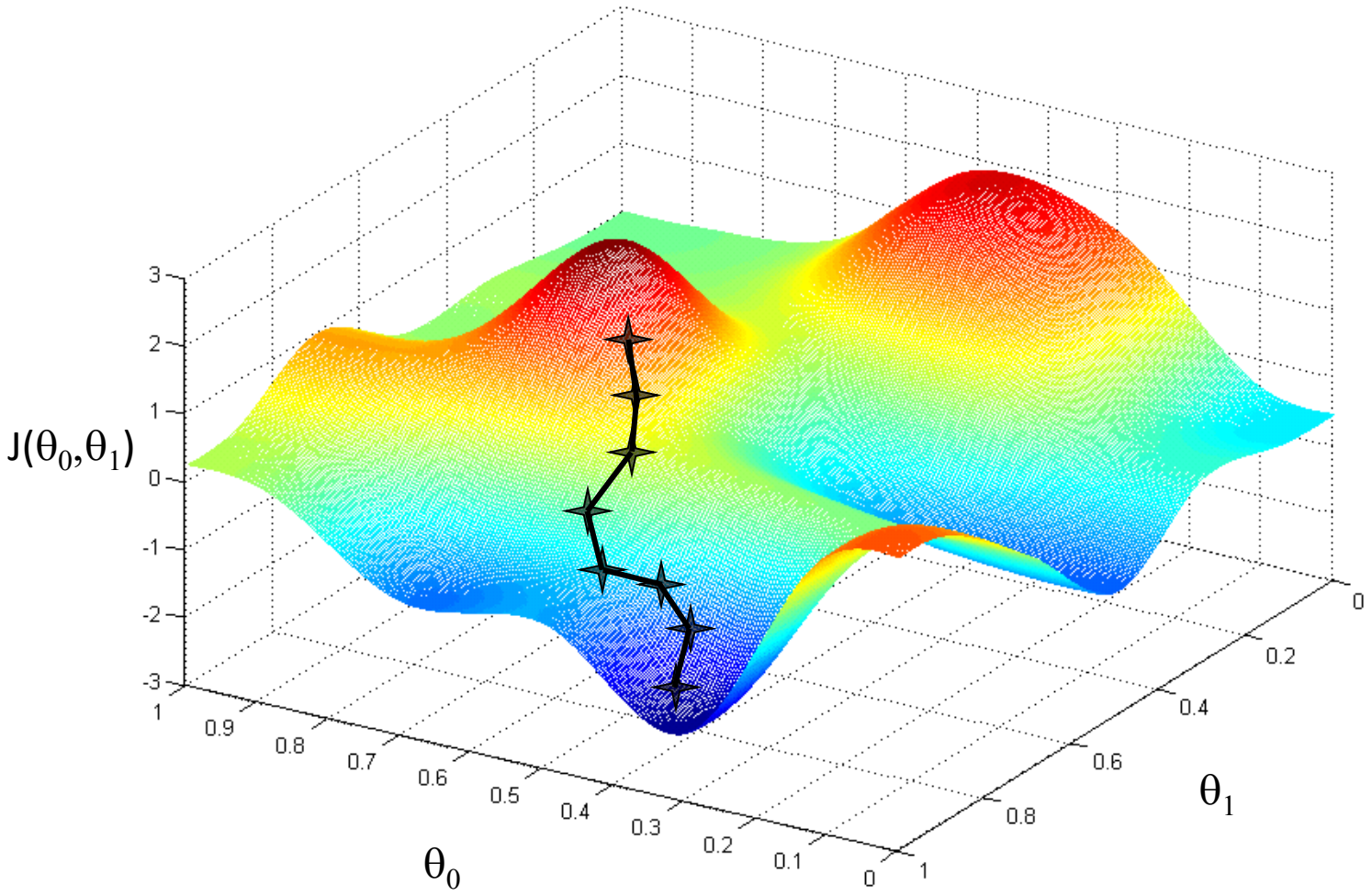


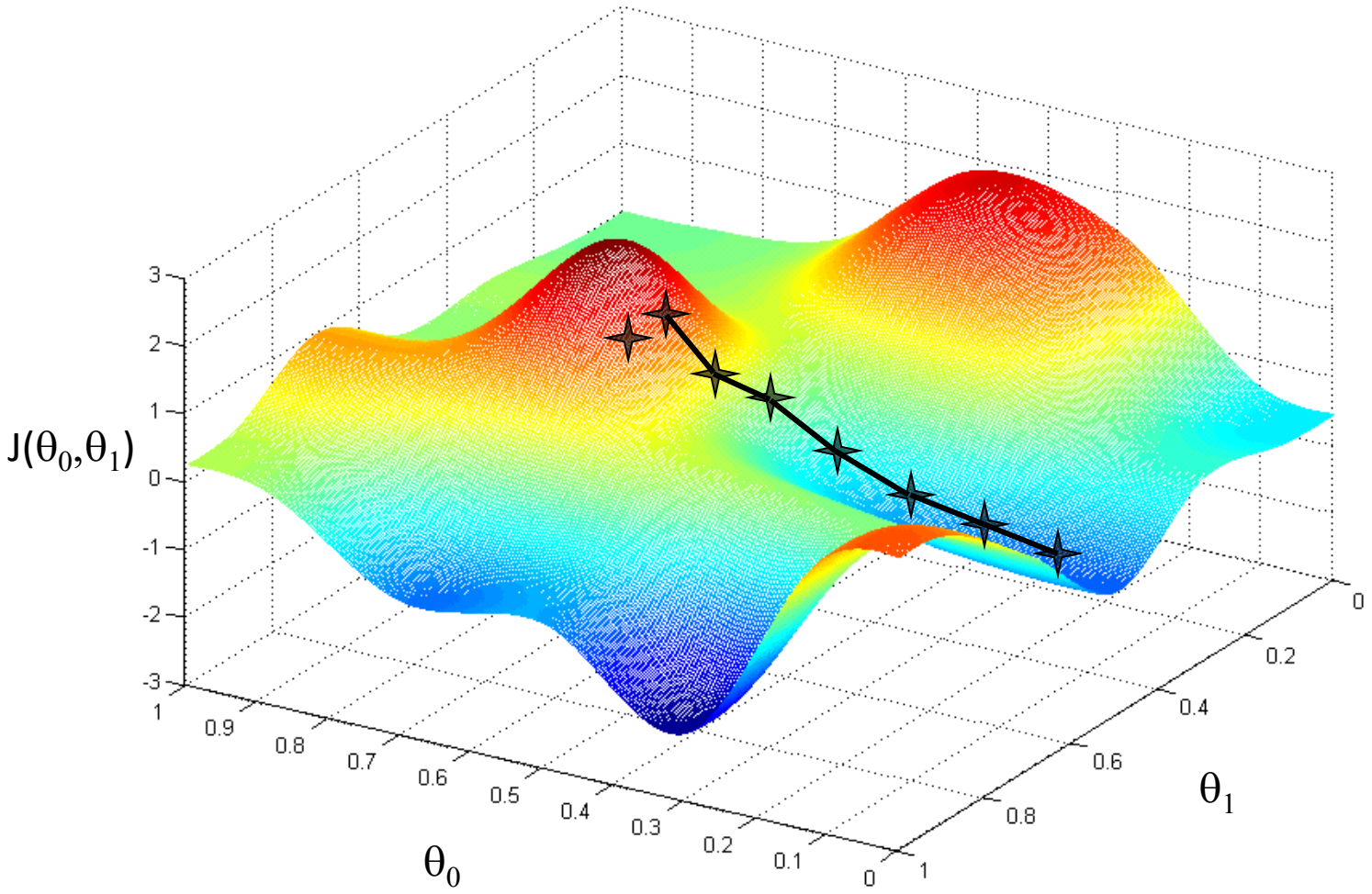
Имаме некоја функција $J(\theta_0, \theta_1)$

Сакаме да најдеме $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Пристап:

- Започни со некои иницијални вредности за θ_0, θ_1
- Менувај ги θ_0, θ_1 за да се намали $J(\theta_0, \theta_1)$ се додека не дојдеш до минимум


J(θ0,θ1)
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
θ0
0
0.2
0.4
0.6
0.8
1
θ1


J(θ0,θ1)
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
θ0
0
0.2
0.4
0.6
0.8
1
θ1

Gradient descent алгоритам

Повторувај додека не конвергира {

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

}

рата на учење

ТОЧЕН ПРИСТАП:
ИСТОВРЕМЕНО АЖУРИРАЊЕ

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

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

$$\theta_0 := temp0$$

$$\theta_1 := temp1$$

НЕТОЧЕН ПРИСТАП:

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

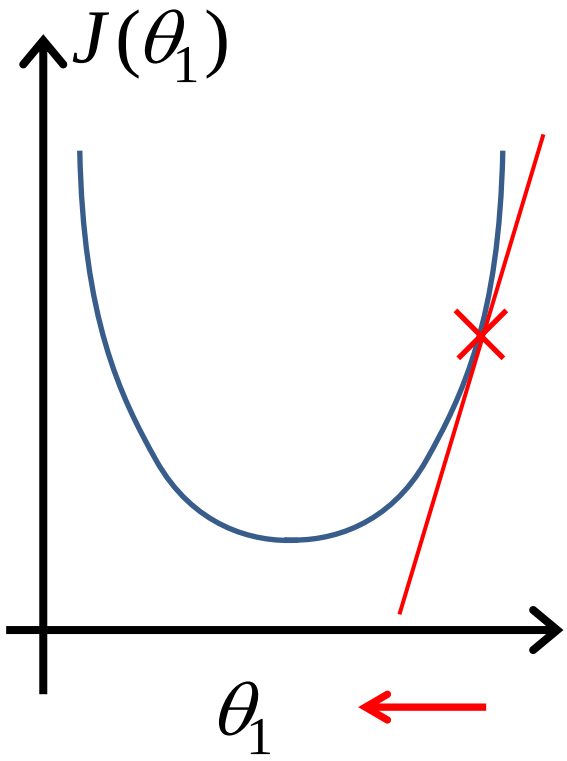
$$\theta_0 := temp0$$

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

$$\theta_1 := temp1$$

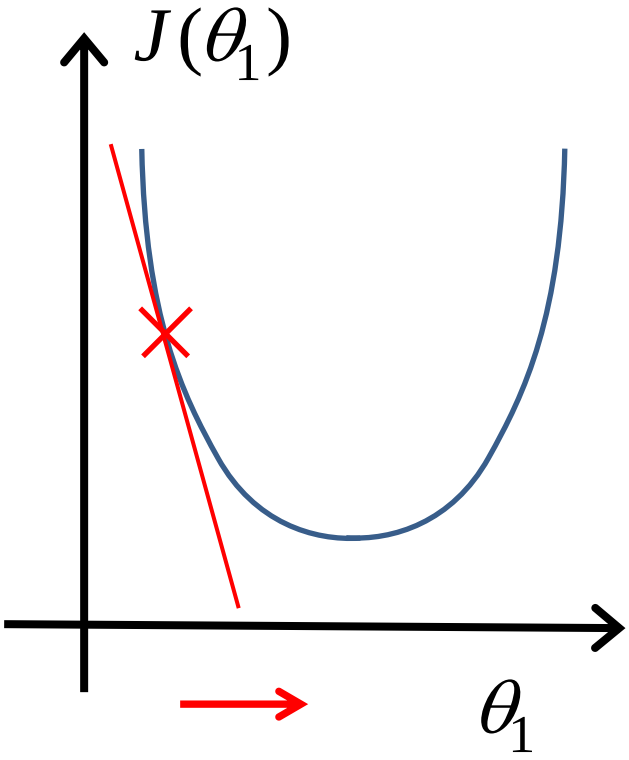
$\theta_0 = 0$

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



наклонот е позитивен

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



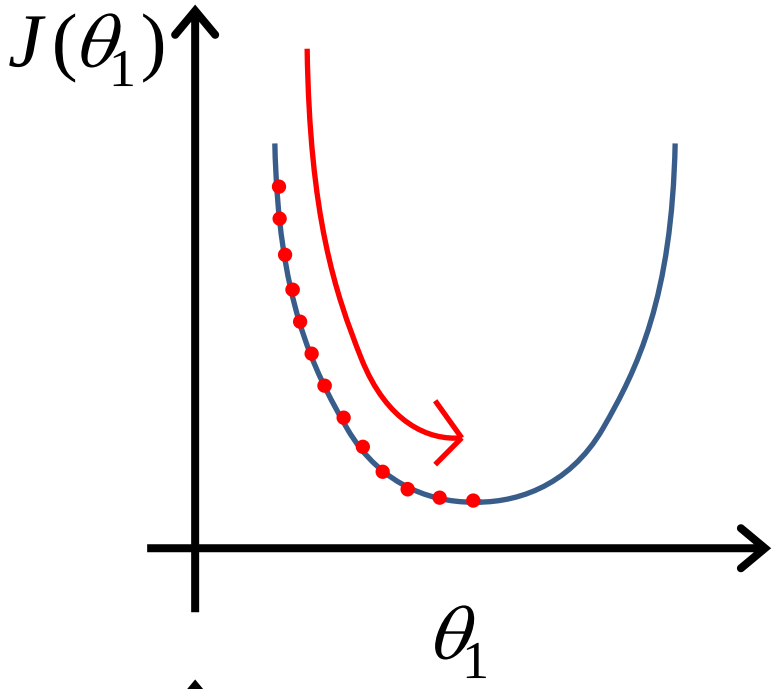
наклонот е негативен

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

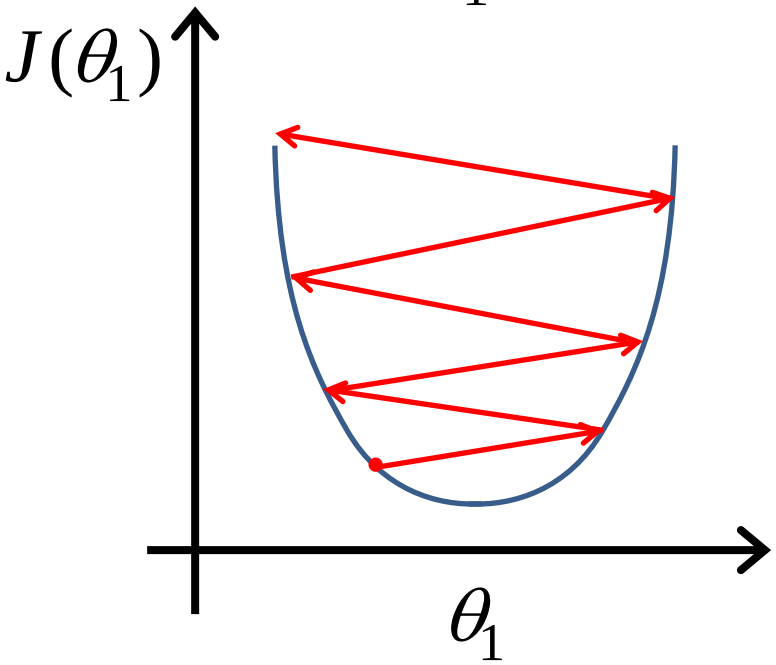
$\theta_0 = 0$

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

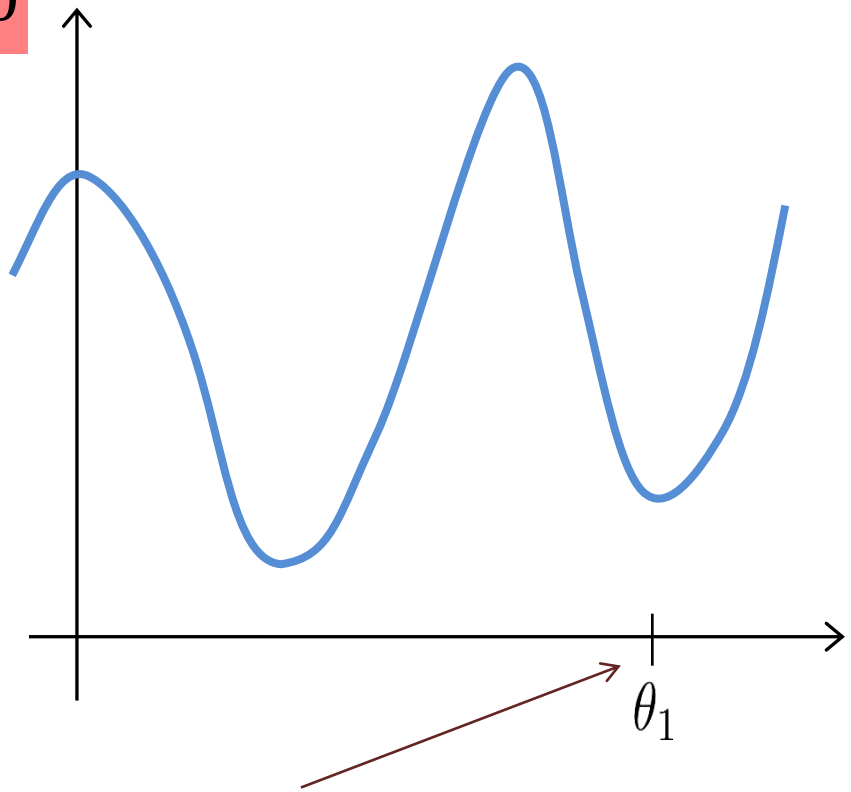
Ако α е премногу мало, алгоритамот може да биде премногу бавен.



Ако α е премногу големо, алгоритамот може да го промаши минимумот. Може да не конвергира, па дури и да дивергира.



$\theta_0 = 0$



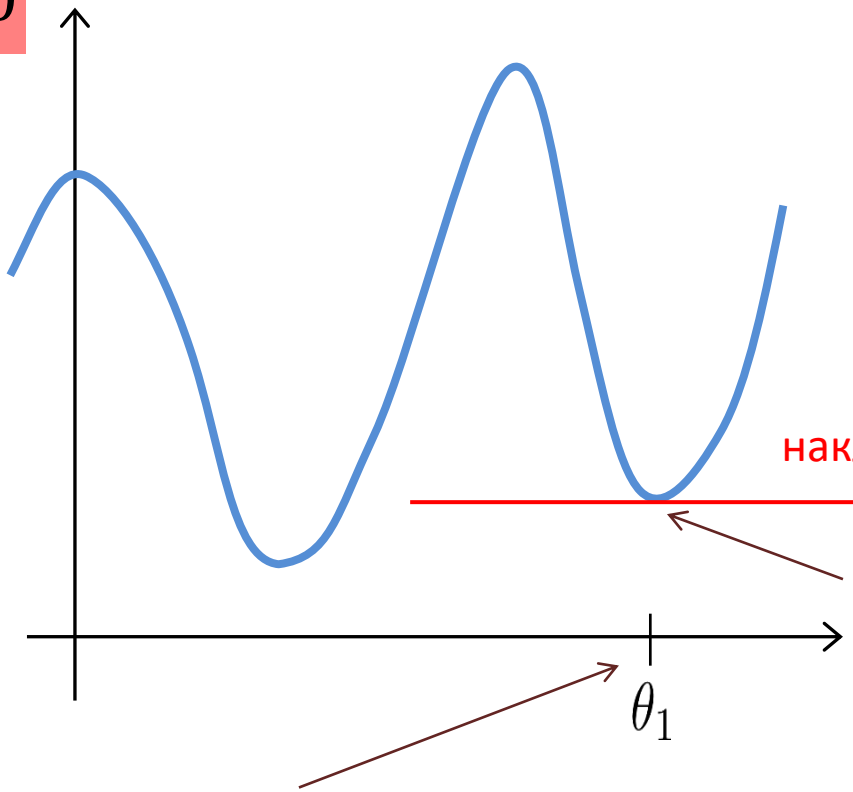
Моментална вредност на θ_1

Колку ќе биде вредноста на θ_1?

a) Ќе остане иста
b) Ќе се зголеми
c) Ќе се намали
d) Ќе биде бесконечност

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

$\theta_0 = 0$



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

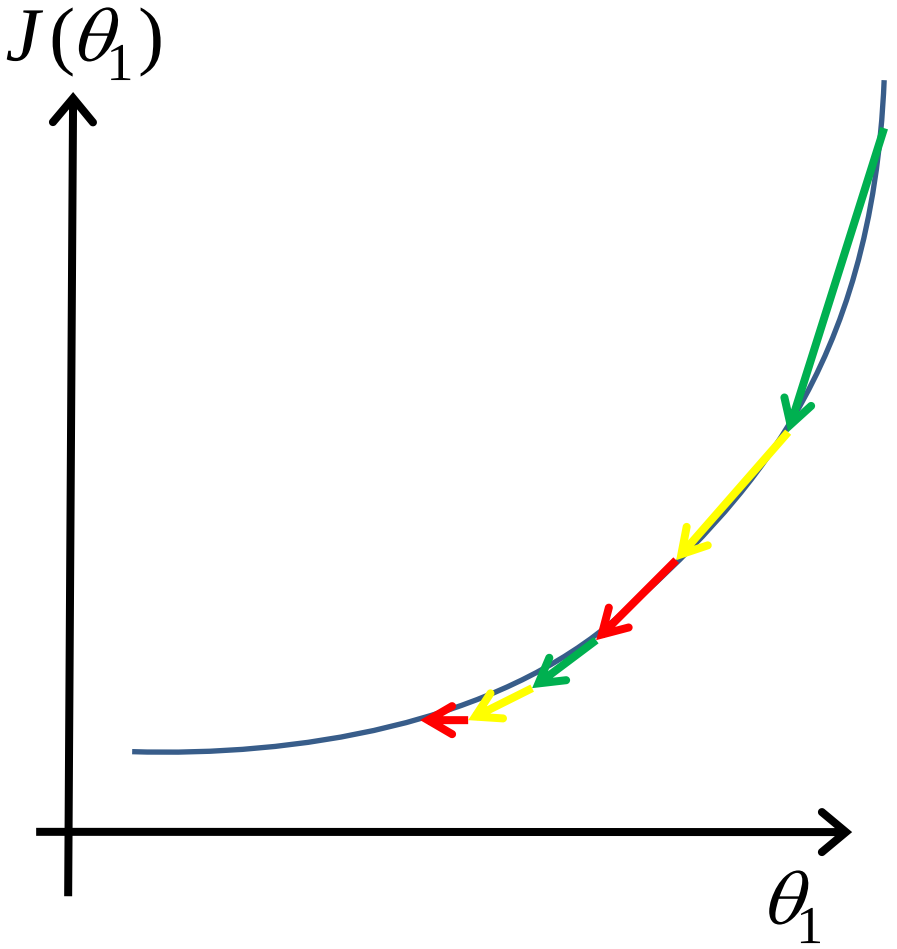
$$\theta_1 := \theta_1$$

$\theta_0 = 0$

Gradient descent може да конвергира во локалниот минимум, дури и кога ратата на учење α е фиксна.

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

Како што се приближуваме кон локалниот минимум, алгоритамот автоматски ќе превзема помали чекори. Што значи, нема потреба да се намалува α во текот на времето.



Gradient descent алгоритам

Повторувај до конвергенција {

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

(за j=0 и j=1)

}

Модел на линеарна регресија

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

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

Gradient descent за линеарна регресија:

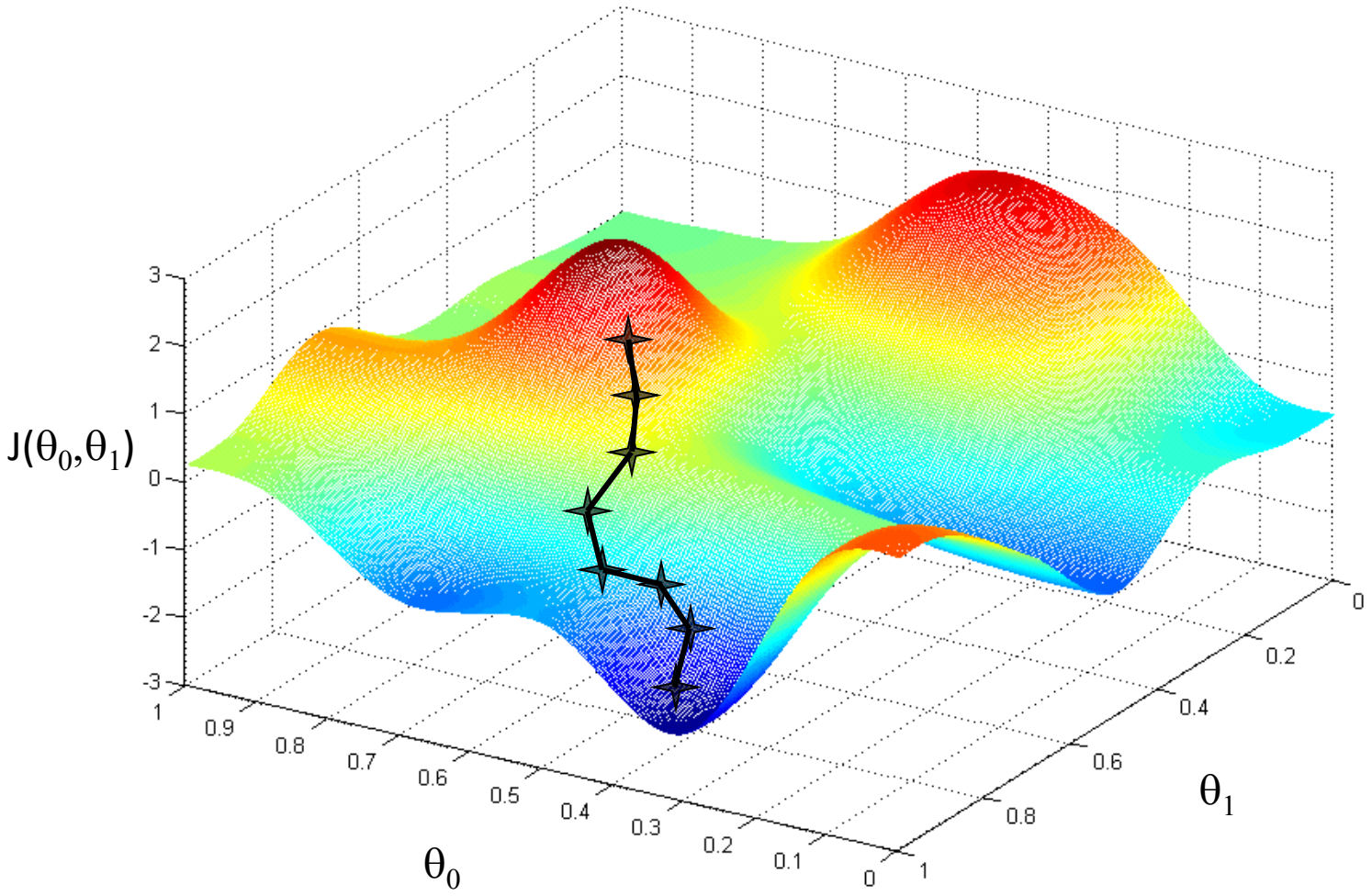
Повторувај до конвергенција {

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

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

}

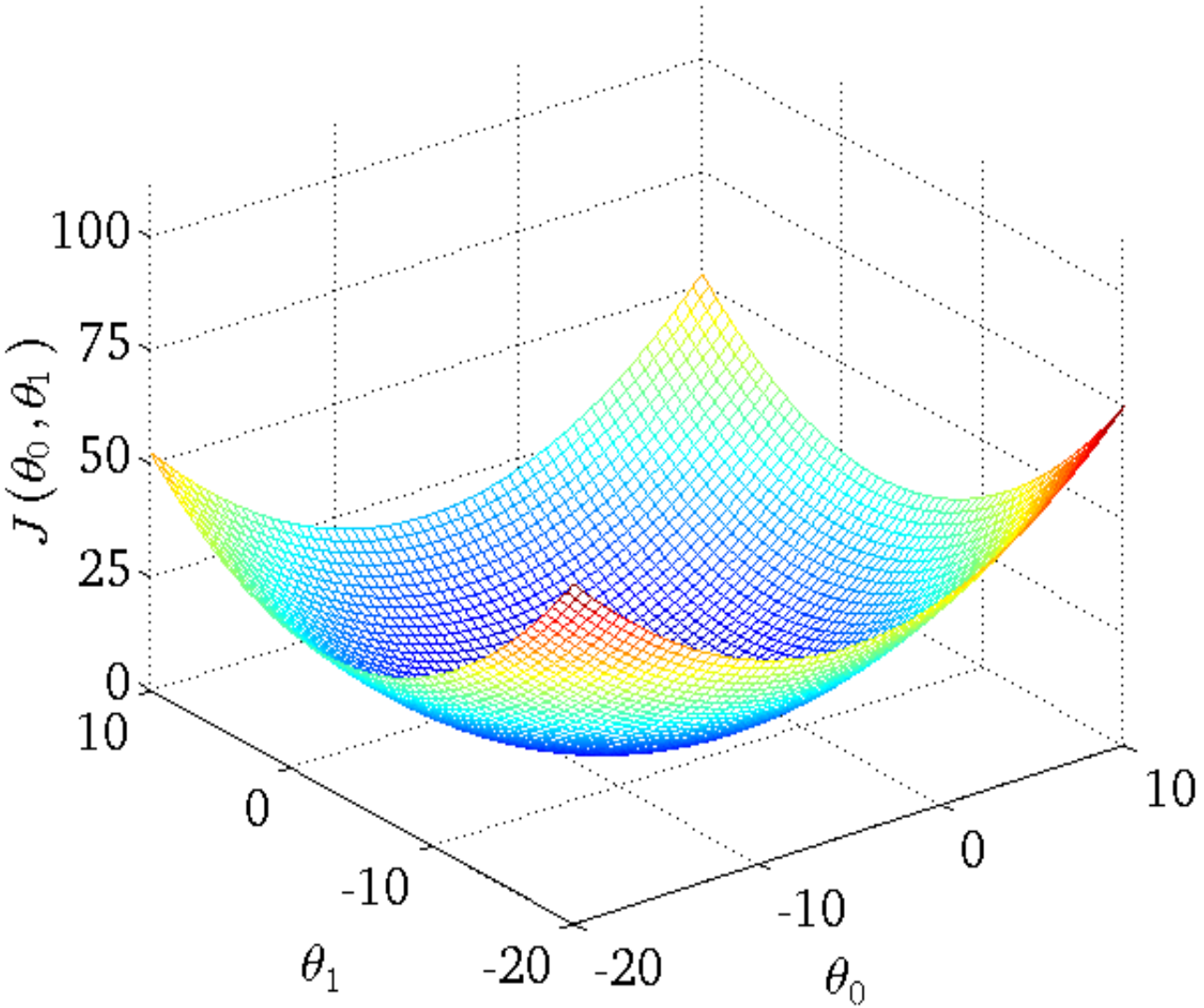
Симултано ажурирање



Наоѓа локален минимум во зависност од почетните услови

Прашање

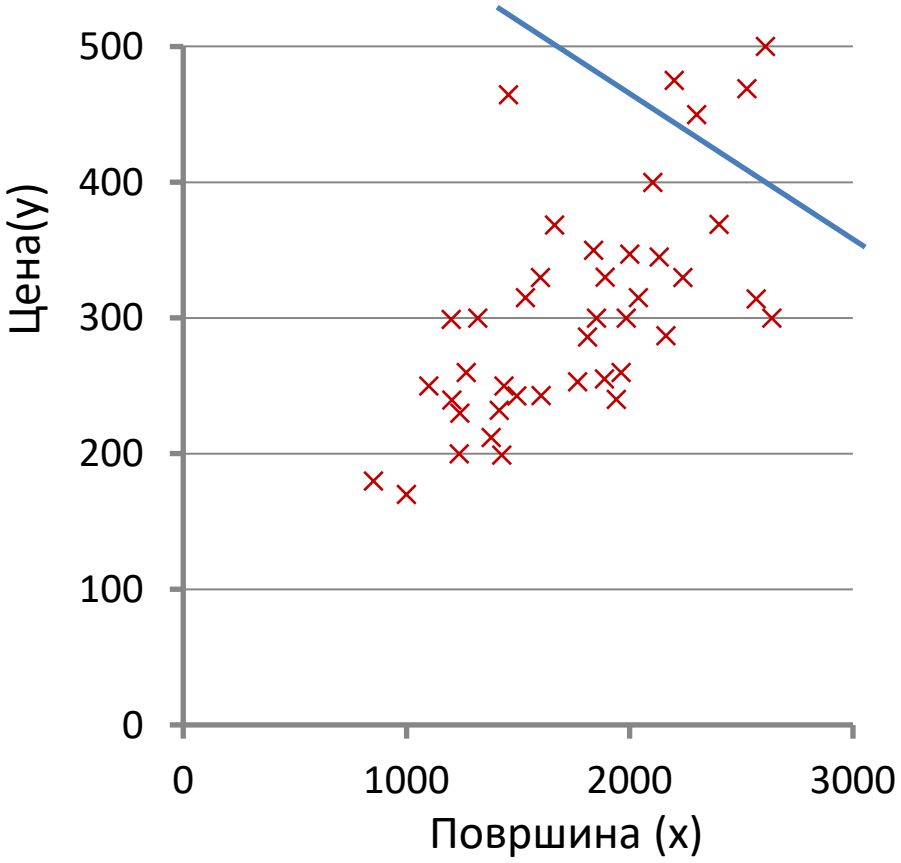
- Што мислите, дали кај линеарна регресија алгоритамот може да заглави во локален минимум (во зависност од тоа како изгледа функцијата на грешка)?
  a) Да
  b) Не



Кај линеарната регресија има само еден минимум (цената на чинење е конвексна функција) и нема никогаш да го имаме проблемот на локални минимуми

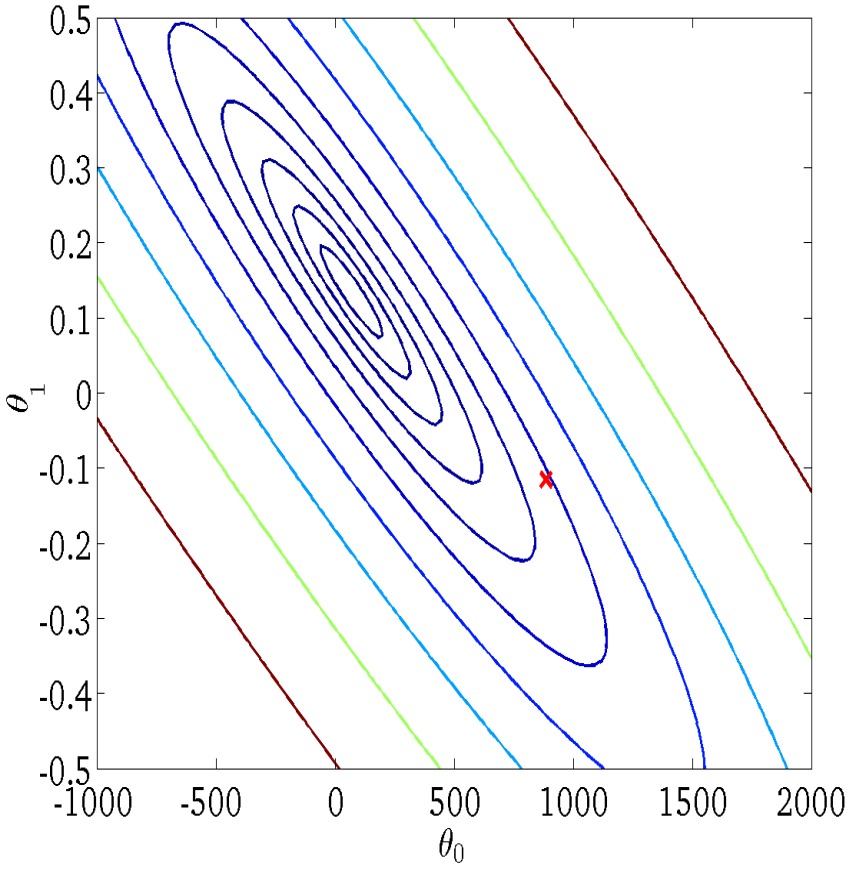
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



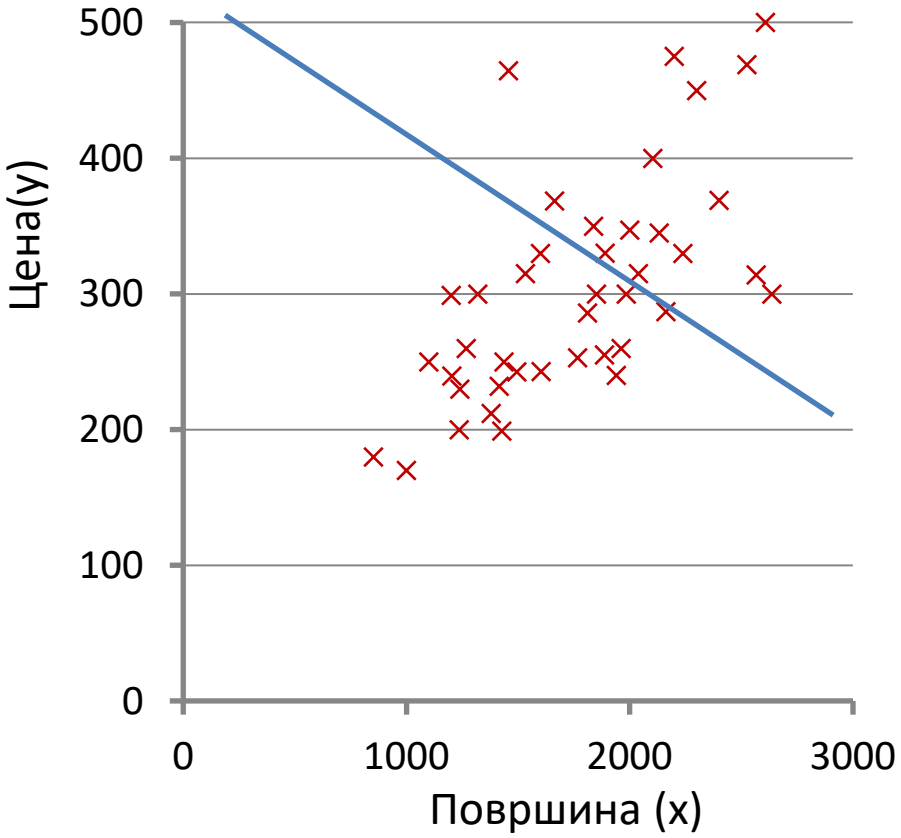
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



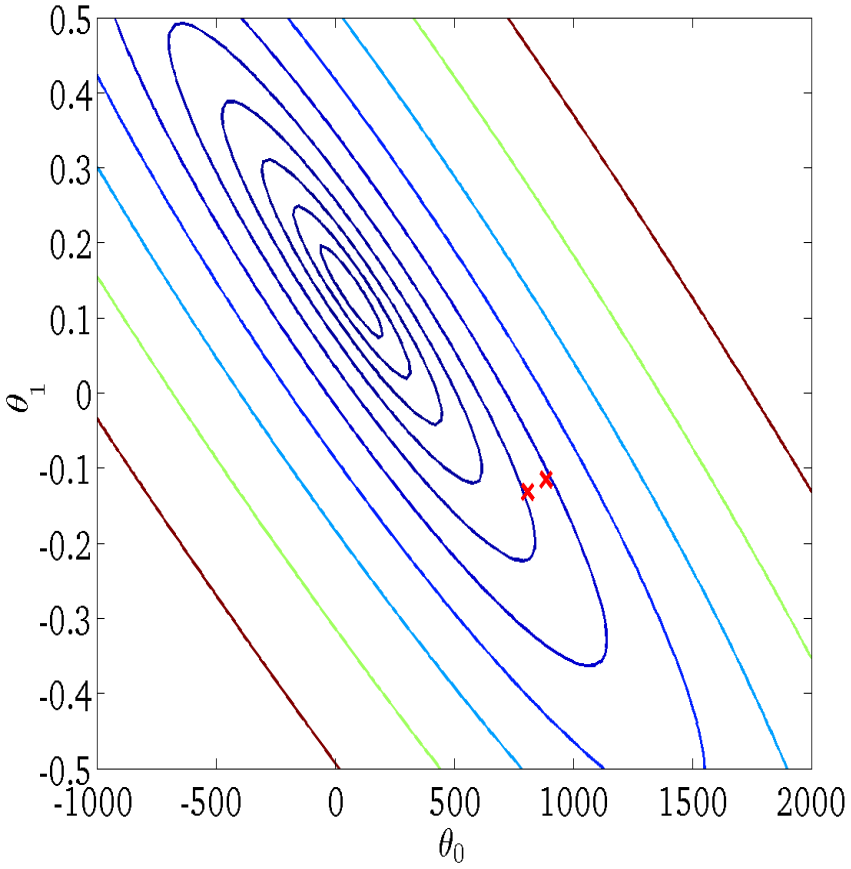
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



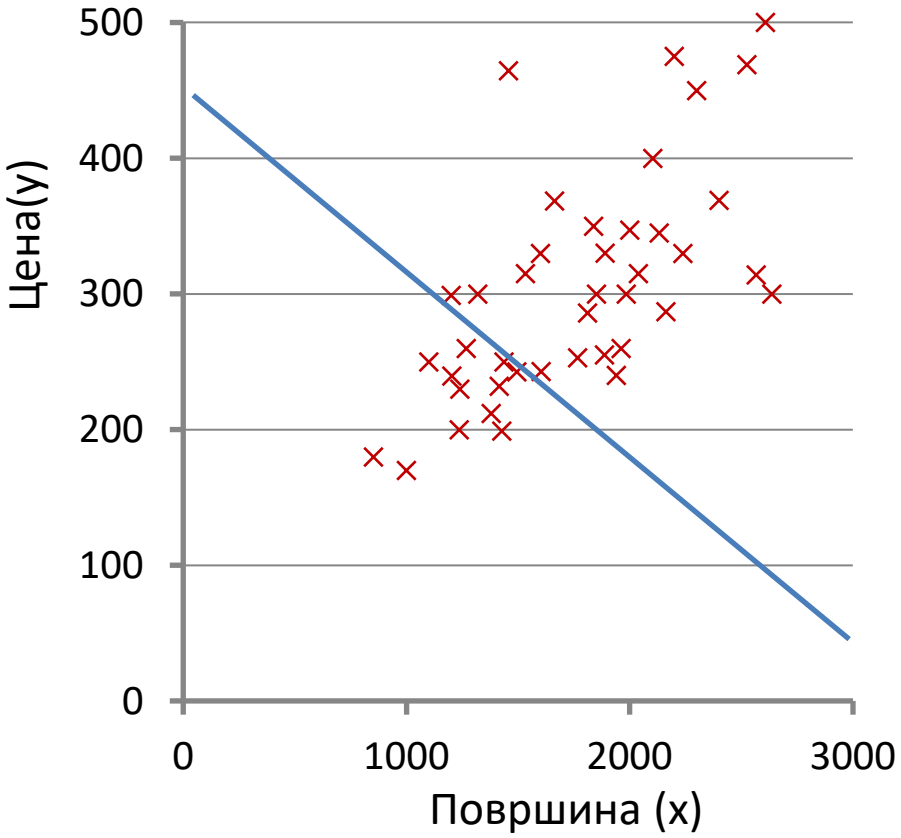
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



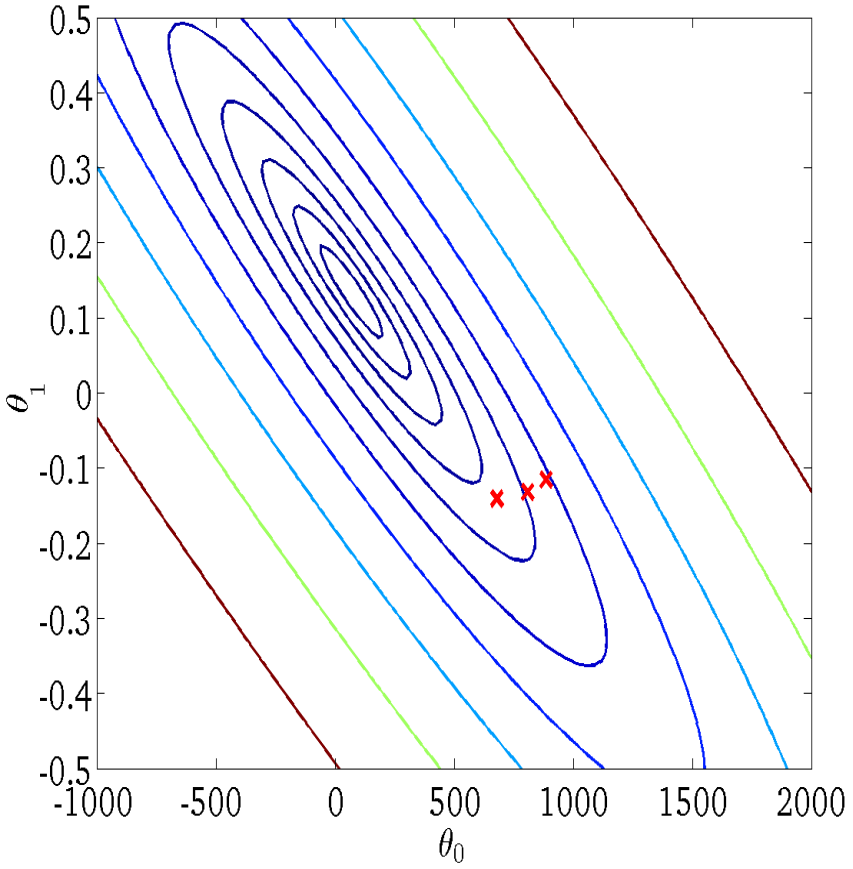
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



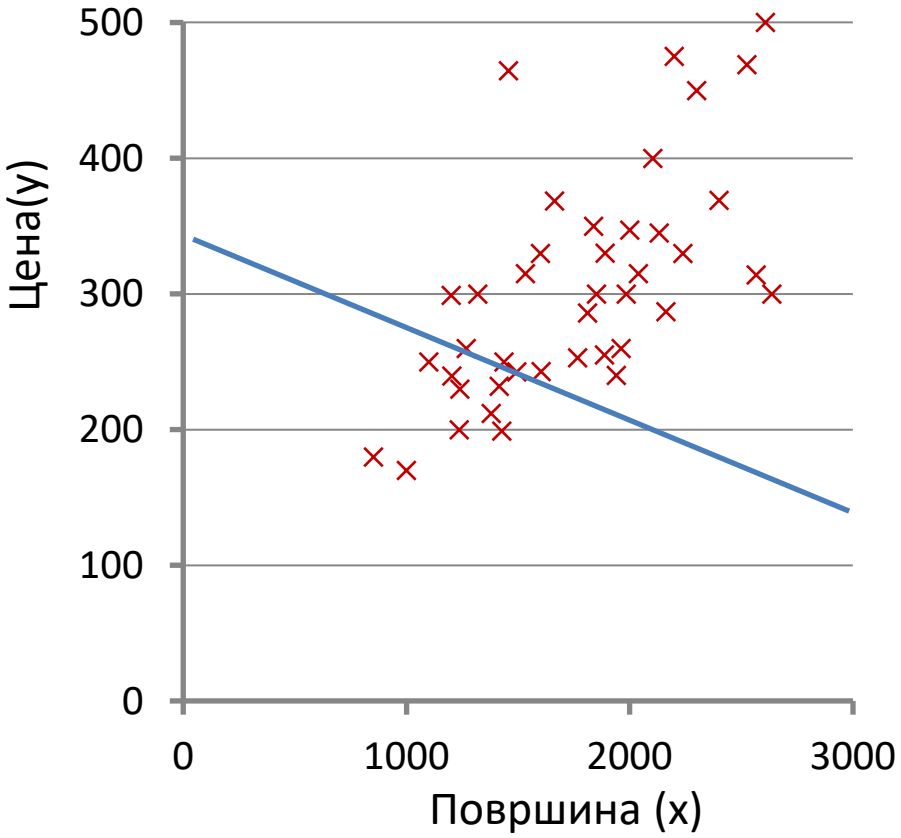
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



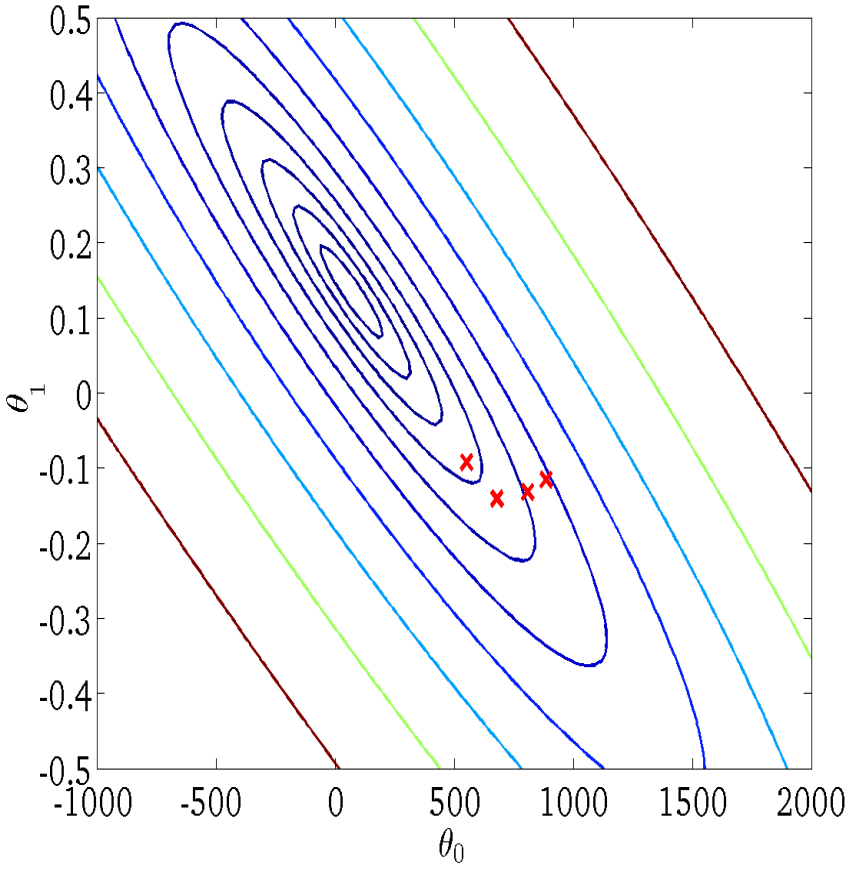
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



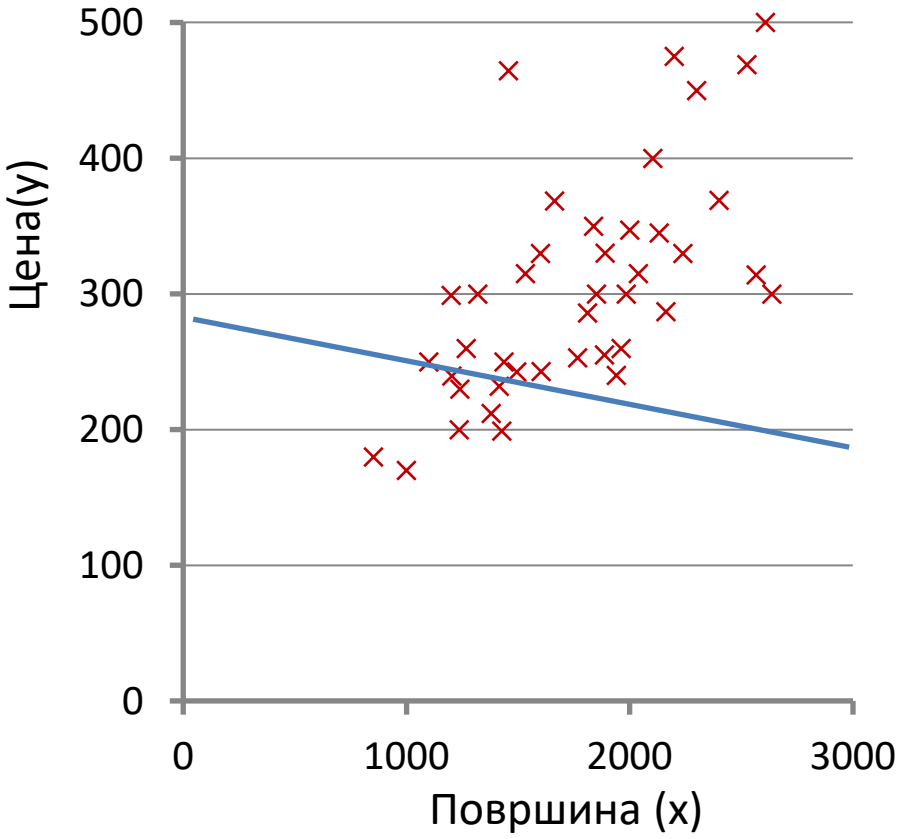
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



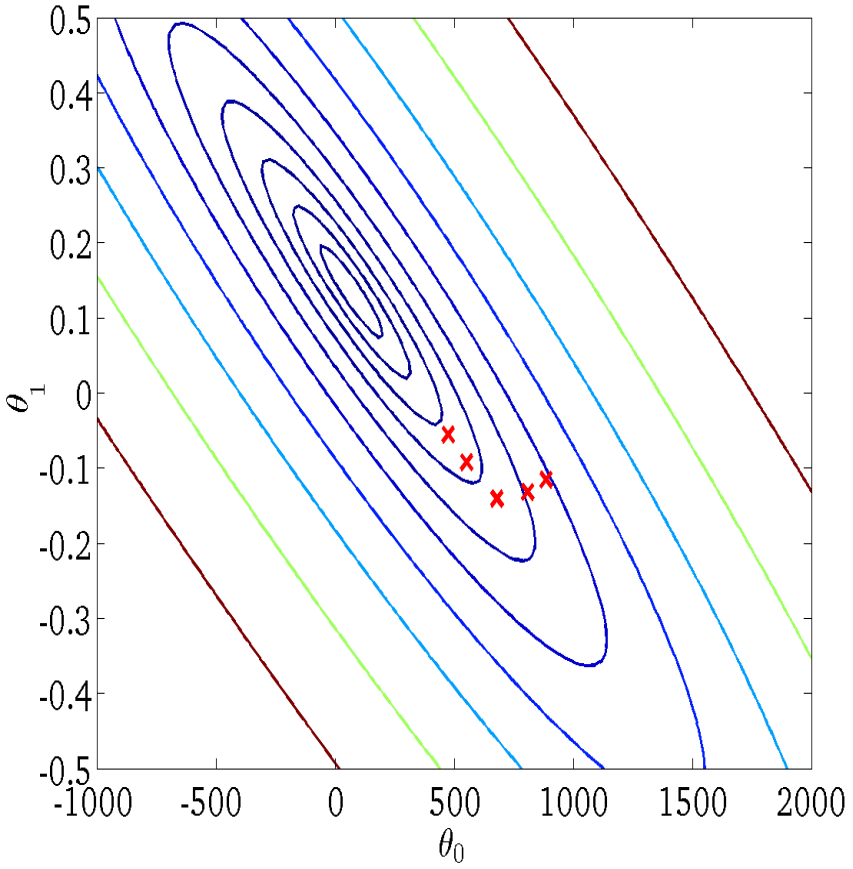
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



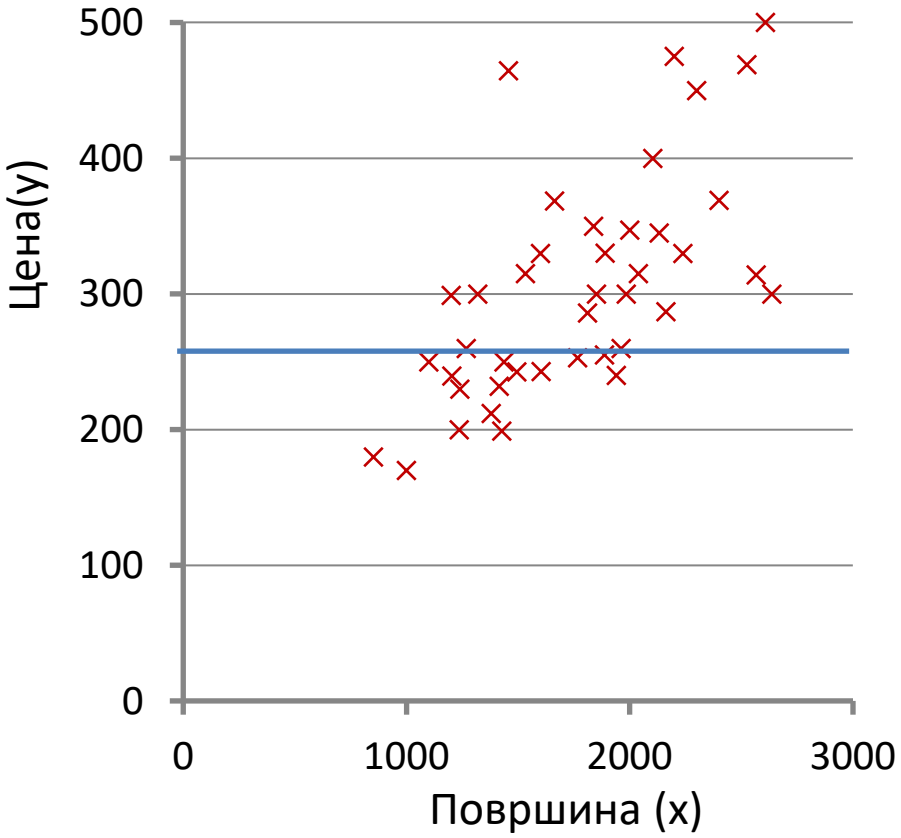
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



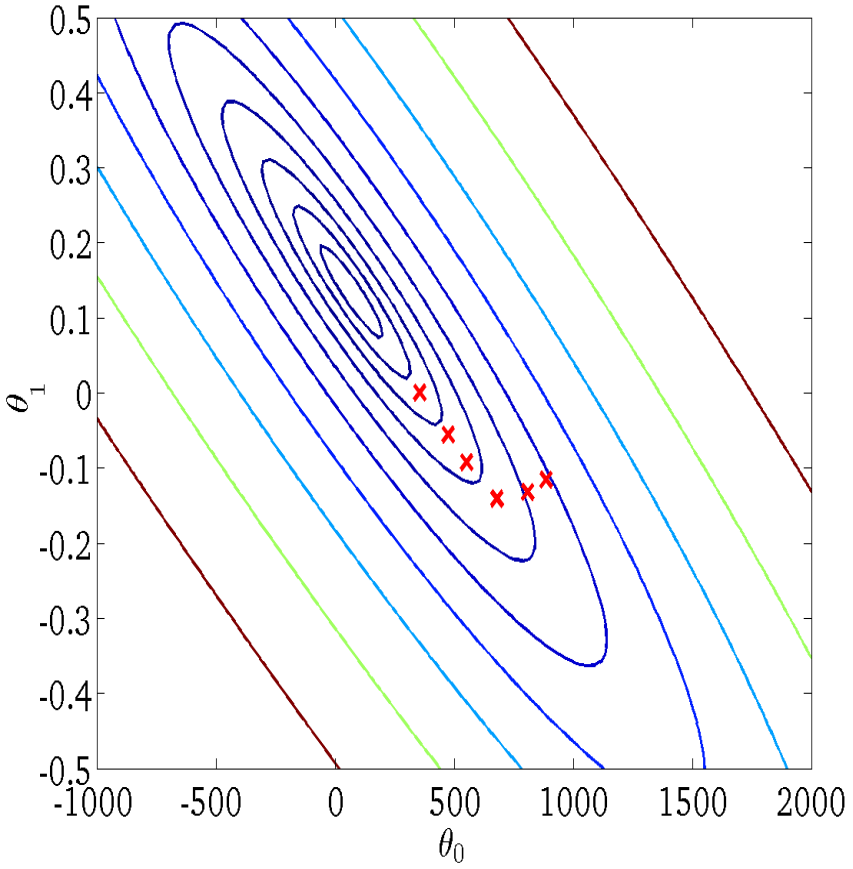
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



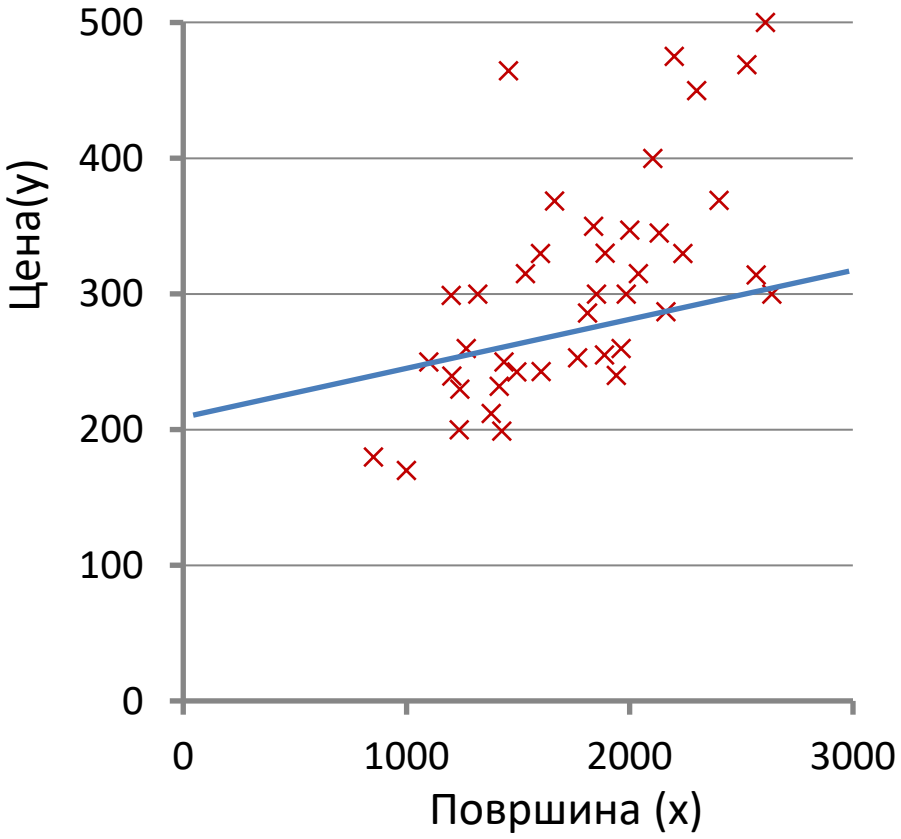
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



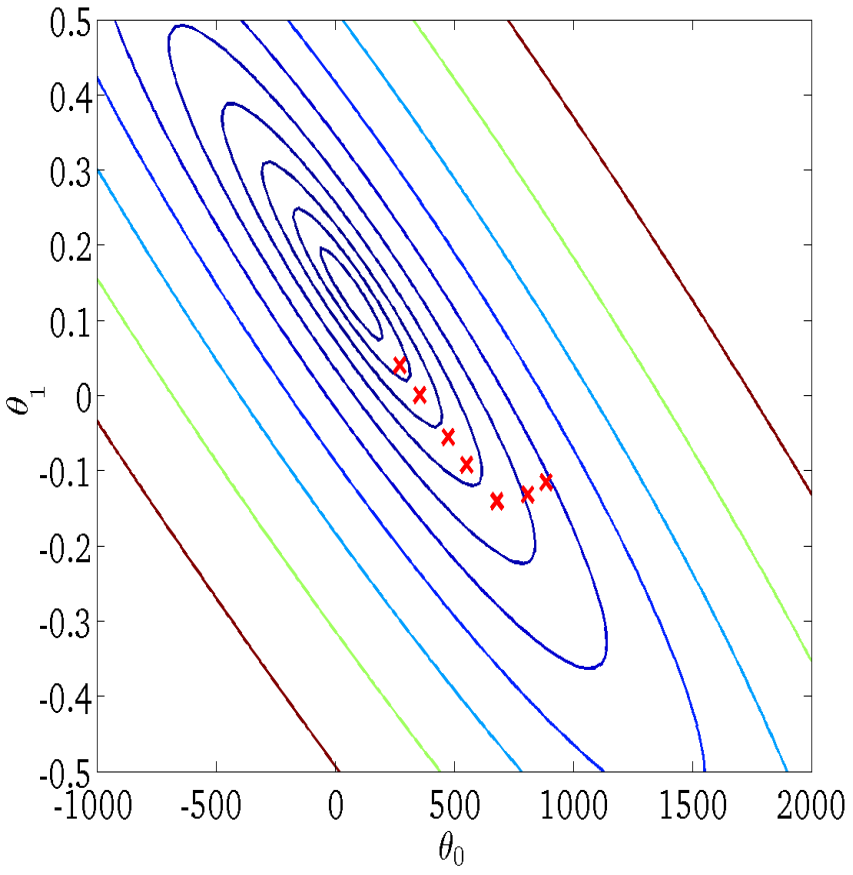
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



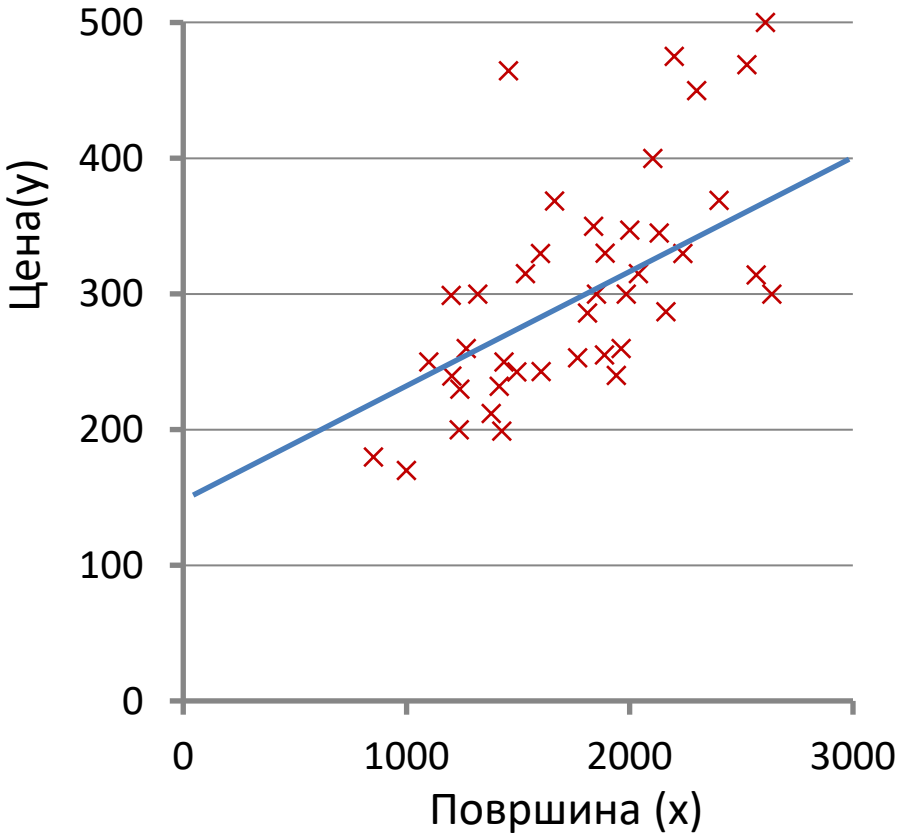
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



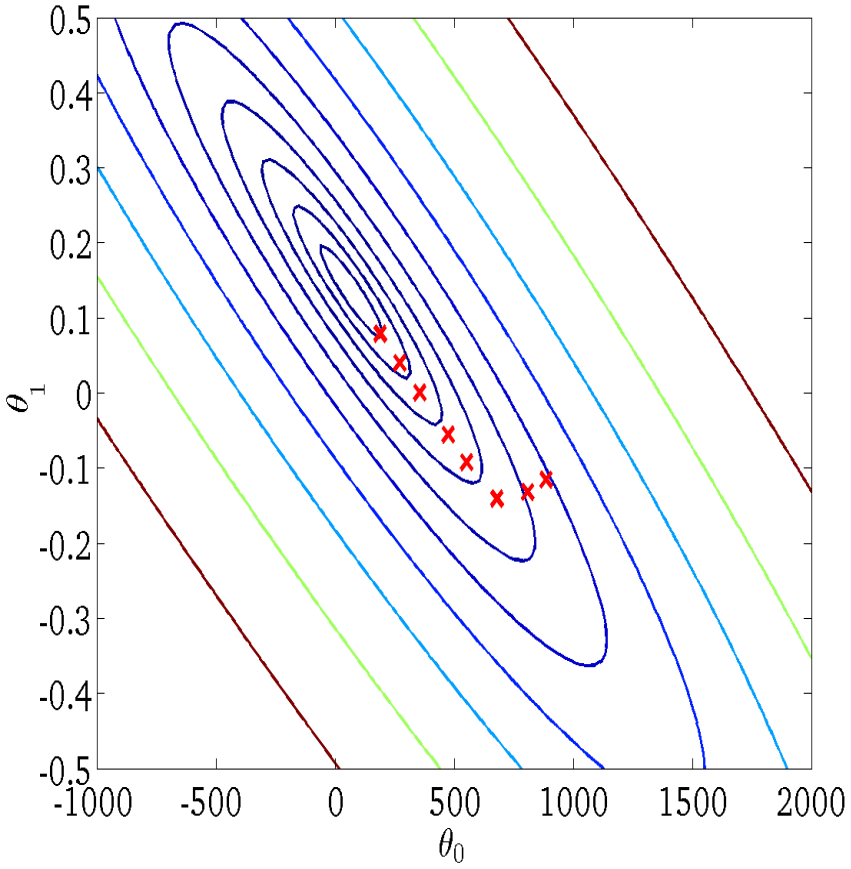
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



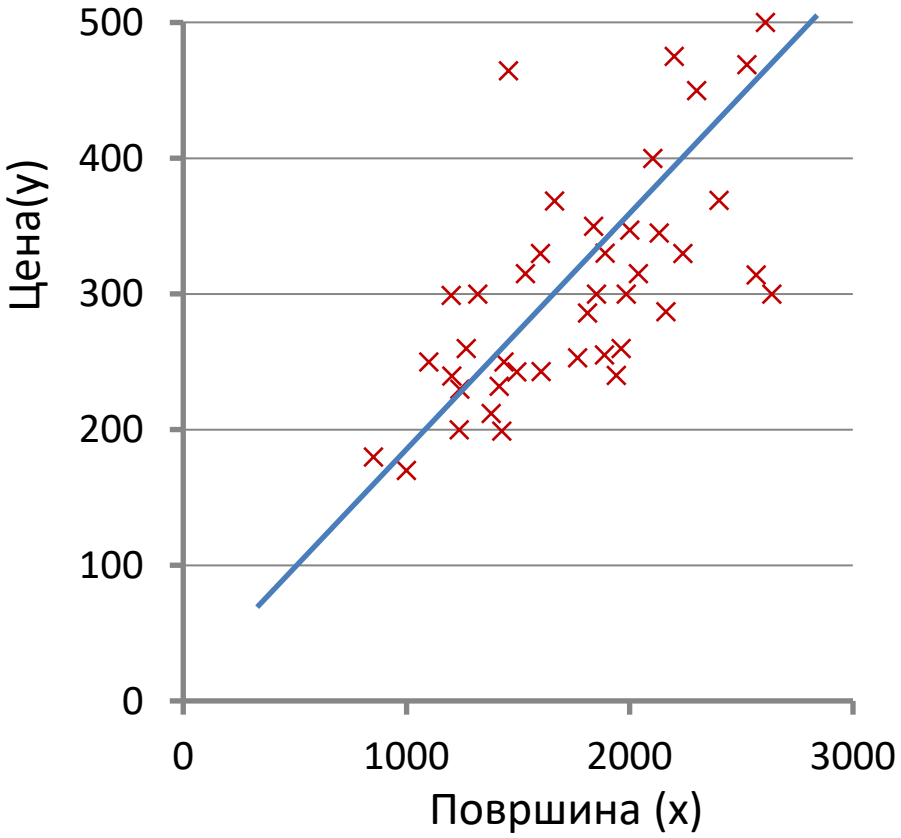
$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



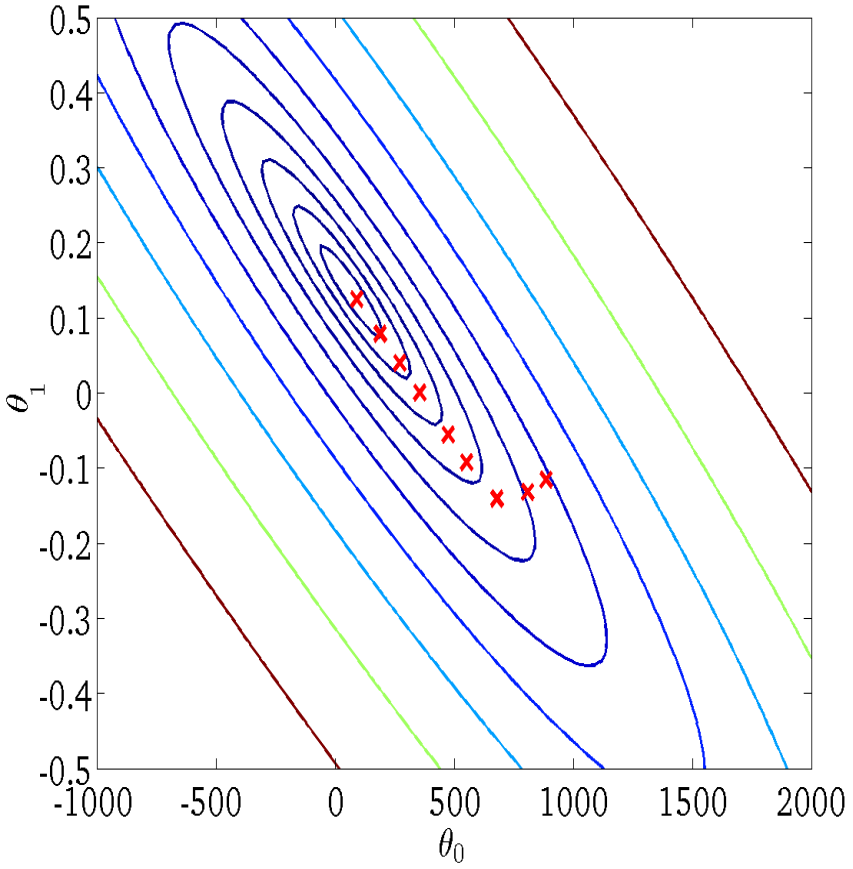
$h_\theta(x)$

(за фиксни θ_0, θ_1 ова е функција од x)



$J(\theta_0, \theta_1)$

(функција од параметрите θ_0, θ_1)



“Batch” Gradient Descent

- “Batch”: Секој чекор на алгоритамот ги користи сите примероци од множеството за тренирање
  - Одлично за конвексни или релативно “smooth” функции на грешка.
- Алтернатива: податоците да се процесираат по делови во секој чекор од алгоритамот.

Што ако имаме повеќе од една влезна променлива?

Површина (m^2)	Број на спални	Број на катови	Старост на куќата (години)	Цена($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Нотација:

n = број на карактеристики

$x^{(i)}$ = влез (карактеристики) на i-тиот примерок

$x_j^{(i)}$ = вредност на карактеристиката j во i-тиот примерок

Хипотеза:

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

Сега: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \ldots + \theta_n x_n$

мултиваријантна линеарна регресија

Заради полесно запишување ќе земеме дека $x_0 = 1$.

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \qquad X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad h_\theta(x) = \theta^T X$$

Хипотеза: $h_\theta(x) = \theta^T X$

Параметри: $\theta_0, \theta_1, \theta_2, \ldots, \theta_n$

Функција на цена на чинење:

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

Gradient descent:

Повторувај $\{$

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

$\}$ (симултано ажурирање за секое $j = 0, \ldots, n$)

Gradient Descent

Претходно ($n = 1$):

Повторувај $\{$

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

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

$\}$

Нов алгоритам $n \geq 1$:

Повторувај $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

$\ldots$

$\}$

Прашање

Кога има n карактеристики, ја дефинираме функцијата на цена на чинење како

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

За линеарна регресија, која друга нотација исто така е точна:

1. $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2$
2. $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}((\sum_{j=0}^{n}\theta_j x_j^{(i)}) - y^{(i)})^2$
3. $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}((\sum_{j=1}^{n}\theta_j x_j^{(i)}) - y^{(i)})^2$
4. $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}((\sum_{j=0}^{n}\theta_j x_j^{(i)}) - (\sum_{j=0}^{n} y_j^{(i)}))^2$

a) 1 и 2

b) 1 и 3

c) 2 и 4

d) 1 и 4

Скалирање (нормализација) на карактеристики

Идеја: Да се осигураме дека карактеристиките се во сличен опсег.

x_1 = површина (0-2000 m^2)

x_2 = број на спални (1-5)

Постојат многу варијанти за решавање на скалирањето

Подели ја секоја вредност со максималната

$$x_1 = \frac{површина(m^2)}{2000}$$

$$x_2 = \frac{\#спални}{5}$$

Од секоја вредност одземи ја средната и подели со максимум (девијацијата)

$$x_1 = \frac{површина - 1000}{2000}$$

$$x_2 = \frac{\#спални - 2}{5}$$

Mean normalization - Од секоја вредност одземи ја средната и подели со опсегот

$$x_1 = \frac{површина - 1000}{2000}$$

$$x_2 = \frac{\#спални - 2}{4}$$

Прашање

- Да претпоставиме дека користите алгоритам за учење за да ја процените цената на куќите во еден град. Сакате една од вашите одлики x_i да ја долови староста на куќата. Во вашето тренинг множество, сите куќи имаат старост од 30 до 50 години, со просечна старост од 38 години.
- Со кој од следниве изрази би направиле mean normalization на влезните карактеристики?

a) $x_i = \text{age of house}$
b) $x_i = \text{age of house/50}$
c) $x_i = \text{(age of house-38)/50}$
d) $x_i = \text{(age of house-38)/20}$

Gradient descent дебагирање

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

- Како да се осигураме дека алгоритамот работи точно.

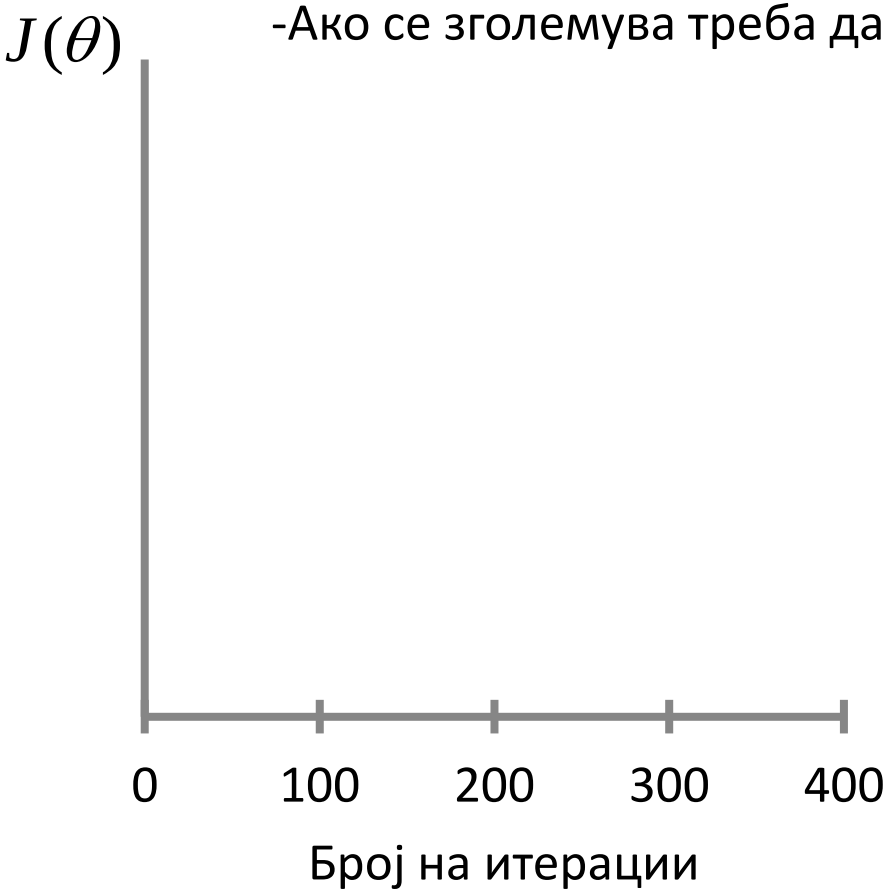
- Како да ја избереме ратата на учење α.

Да се осигураме дека gradient descent работи точно.

-Исцртај крива за вредност на J(θ) во секоја итерација

-Ако вредноста се намалува алгоритамот е добар

-Ако се зголемува треба да се намали ратата на учење



- За доволно мало α, J(θ) треба да се намалува во секоја итерација
- Но ако α е премногу мало, конвергенцијата ќе биде бавна.

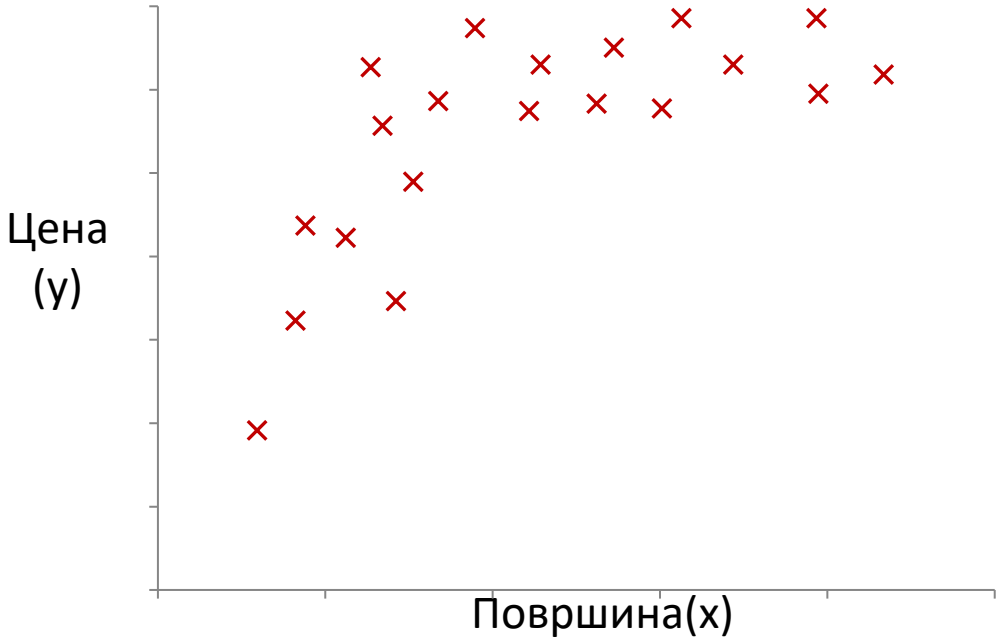
Избирање на вредност за α

- Ако α е премногу мало: бавна конвергенција.
- Ако α е премногу големо: $J(\theta)$ може да не се намали со секоја итерација; може и воопшто да не конвергира.

За избирање на α, пробајте

..., 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, ...

Полиномна регресија



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 =$$

$$\theta_0 + \theta_1(површина) + \theta_2(површина)^2 + \theta_3(површина)^3$$

$$x_1 = (површина) \qquad x_2 = (површина)^2 \qquad x_3 = (површина)^3$$

Аналитичко решение на линеарната регресија

x_0	Површина (m²) x_1	Број на спални x_2	Број на катови x_3	Старост на куќата (год) x_4	Цена ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
1	3000	4	1	38	540

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \\ 1 & 3000 & 4 & 1 & 38 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \\ 540 \end{bmatrix}$$

Нормална равенка

$$\Theta = (X^T X)^{-1} X^T y$$

Можен проблем со инверзната матрица?

m примероци за тренирање, n карактеристики.

Gradient Descent

- Треба да се избере α.
- Потребни се повеќе итерации.
- Работи добро кога n е големо.

Нормална равенка

- Не треба да се избере α.
- Не е потребно итерирање.
- Треба да се пресмета $(X^T X)^{-1}$
- Бавно ако n е големо.

Linear regression

- Using probabilistic notation, linear regression can be represented as

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T\mathbf{x}, \sigma^2)$$

- It can also model non-linear relationships by replacing x with some non-linear function ϕ(x)
  - Known as **basis function expansion**

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}), \sigma^2)$$

  - Polynomial regression is just one such expansion

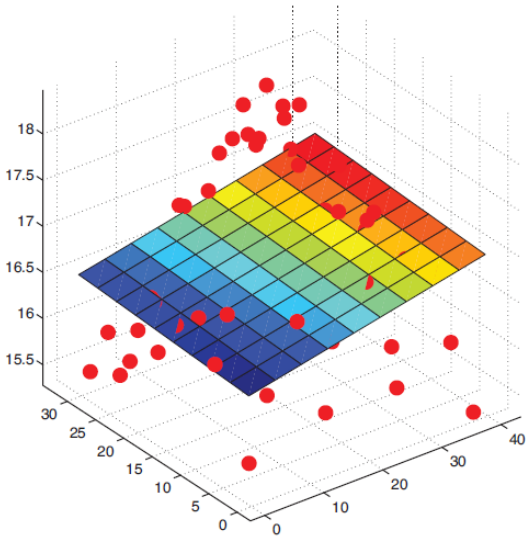
$$\boldsymbol{\phi}(x) = [1, x, x^2, \ldots, x^d]$$

Linear regression

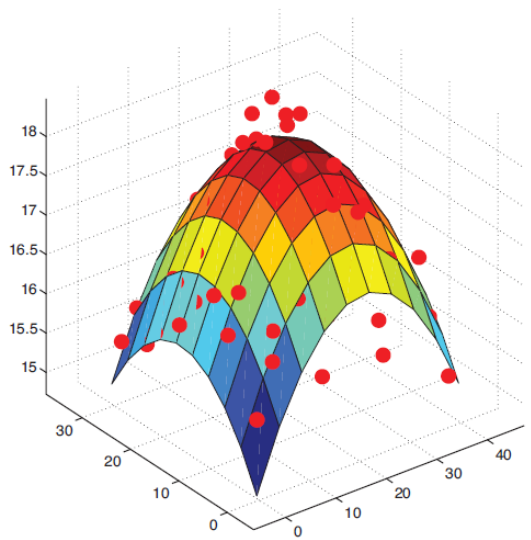
- Linear regression applied to 2D data. Vertical axis is temperature, horizontal axes are location within a room. Data was collected by some remote sensing motes at Intel's lab in Berkeley, CA.
- Temperature data is fitted using two forms of $\hat{f}(\boldsymbol{x})$

  a) $\hat{f}(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2$

  b) $\hat{f}(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2$



(a)



(b)

Maximum Likelihood Estimation

- A common way to estimate the parameters of a statistical model is to compute the MLE

$$\hat{\boldsymbol{\theta}} \triangleq \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta})$$

- If we assume that the training examples are iid the log-likelihood is given as

$$\ell(\boldsymbol{\theta}) \triangleq \log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$$

- One can then maximize the likelihood or minimize the negative log-likelihood (NLL)

$$\mathrm{NLL}(\boldsymbol{\theta}) \triangleq -\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$$

- By inserting the definition of the Gaussian, we have

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \left[\left(\frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 \right) \right] = \frac{-1}{2\sigma^2} RSS(\mathbf{w}) - \frac{N}{2}\log(2\pi\sigma^2)$$

Residual sum of squares

- RSS in the previous equation stands for **Residual sum of squares** and is defined by:

$$\mathrm{RSS}(\mathbf{w}) \triangleq \sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

- It can also be written as the square of the vector of the residual errors:

$$\mathrm{RSS}(\mathbf{w}) = ||\boldsymbol{\epsilon}||_2^2 = \sum_{i=1}^{N}\epsilon_i^2 \qquad \epsilon_i = (y_i - \mathbf{w}^T\mathbf{x}_i)$$

- MLE for $\mathbf{w}$ is the one that minimizes the RSS, a method known as **least squares**

Ordinary Least Squares

- If we use the following form for the objective function

$$\mathrm{NLL}(\mathbf{w}) = \frac{1}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w}) = \frac{1}{2}\mathbf{w}^T(\mathbf{X}^T\mathbf{X})\mathbf{w} - \mathbf{w}^T(\mathbf{X}^T\mathbf{y})$$

its gradient is $\mathbf{g}(\mathbf{w}) = [\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y}] = \sum_{i=1}^{N}\mathbf{x}_i(\mathbf{w}^T\mathbf{x}_i - y_i)$

and by $\mathbf{g}(\mathbf{w}) = 0$ we get the **normal equation** $\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y}$

- The corresponding solution $\hat{\mathbf{w}}$ to this linear system of equations is called the **ordinary least squares** (OLS) solution

$$\boxed{\hat{\mathbf{w}}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}}$$

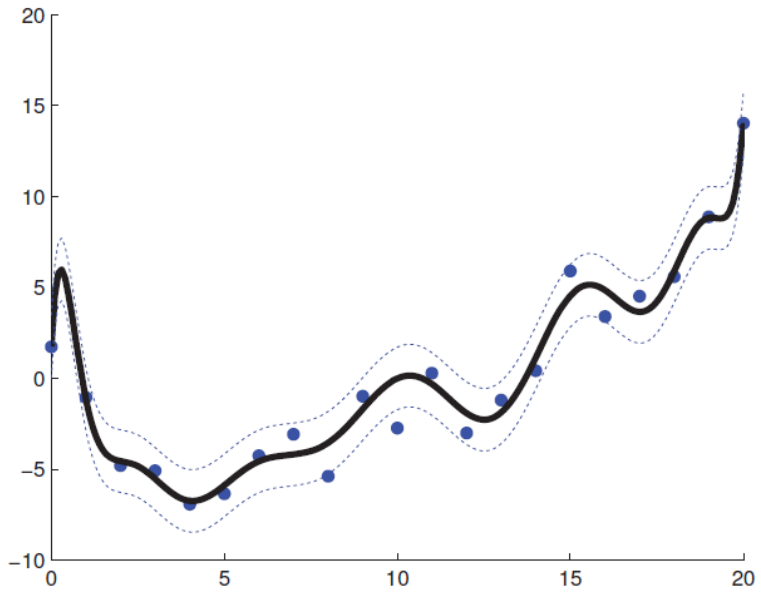
- Cons:
  - Calculation of $(X^TX)^{-1}$ is slow for large N
  - Calculation of $(X^TX)^{-1}$ may be impossible due to singularity (if parameters are highly correlated, there's not enough data…)

Ridge regression

- One problem with ML estimation is that it can result in overfitting.
- We discuss a way to ameliorate this problem by using MAP estimation with a prior.

Ridge regression

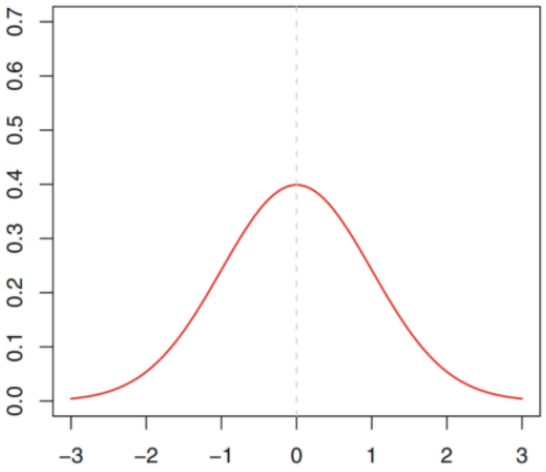
- The reason that the MLE can overfit is that it is picking parameter values that are the best for modeling the training data; but if the data is noisy, such parameters often result in complex functions.
- The figure illustrates a fit of degree 14 polynomial to N=21 data points using least square, where the obtained parameters (except w_0) are:
  - 6.560, -36.934, -109.255, 543.452, 1022.561, -3046.224, -3768.013, 8524.540, 6607.897, -12640.058, -5530.188, 9479.730, 1774.639, -2821.526
- There are many large positive and negative numbers
- This situation is unstable -> if we change the data a little, the coefficients will change a lot



Ridge regression

- We can encourage the parameters to be small, thus resulting in a smother curve, by using a zero-mean Gaussian prior, where $1/\tau^2$ controls its strength

$$p(\mathbf{w}) = \prod_j \mathcal{N}(w_j|0,\tau^2)$$



- The corresponding MAP estimation problem becomes:

$$\underset{\mathbf{w}}{\operatorname{argmax}} \sum_{i=1}^{N} \log \mathcal{N}(y_i|w_0 + \mathbf{w}^T\mathbf{x}_i, \sigma^2) + \sum_{j=1}^{D} \log \mathcal{N}(w_j|0,\tau^2)$$

Ridge regression

- This is equivalent to minimizing:

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + \mathbf{w}^T\mathbf{x}_i))^2 + \lambda||\mathbf{w}||_2^2$$

where $\lambda \triangleq \sigma^2/\tau^2$ and $||\mathbf{w}||_2^2 = \sum_j w_j^2 = \mathbf{w}^T\mathbf{w}$ is the squared two-norm

- The first term is MSE/NLL, and the second is a complexity penalty ($\lambda \geq 0$)
- The corresponding solution is

$$\boxed{\hat{\mathbf{w}}_{ridge} = (\lambda\mathbf{I}_D + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}}$$

- This is called **ridge regression** or penalized least squares.

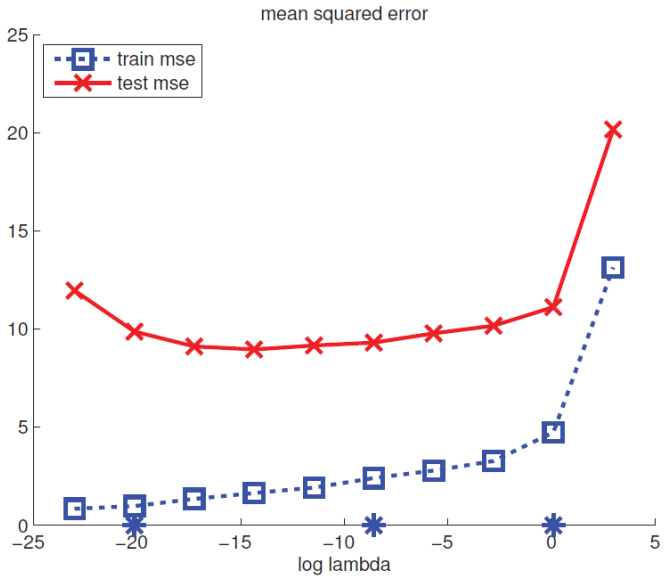
Ridge regression

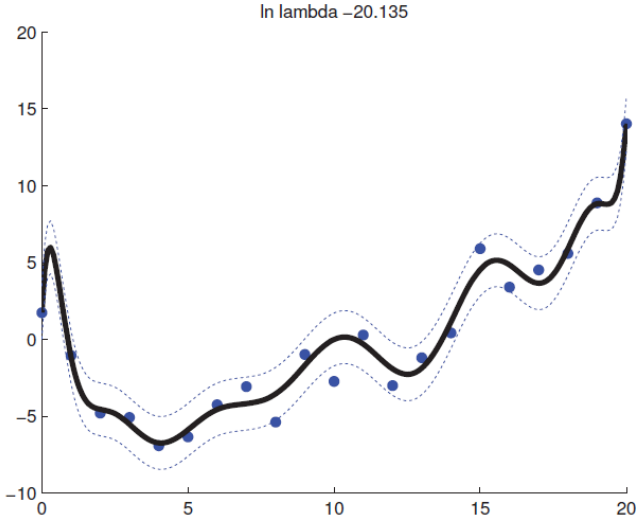
- In general, adding a Gaussian prior to the parameters of a model to encourage them to be small is called ℓ_2 **regularization**.
- Note that the offset term w_0 is not regularized, since it just affects the height of the function, not its complexity.
- By penalizing the sum of the magnitudes of the weights, we ensure the function is simple
  - since **w** = **0** corresponds to a straight line, which is the simplest possible function, corresponding to a constant
- In the previous example if $\lambda = 10^{-3}$

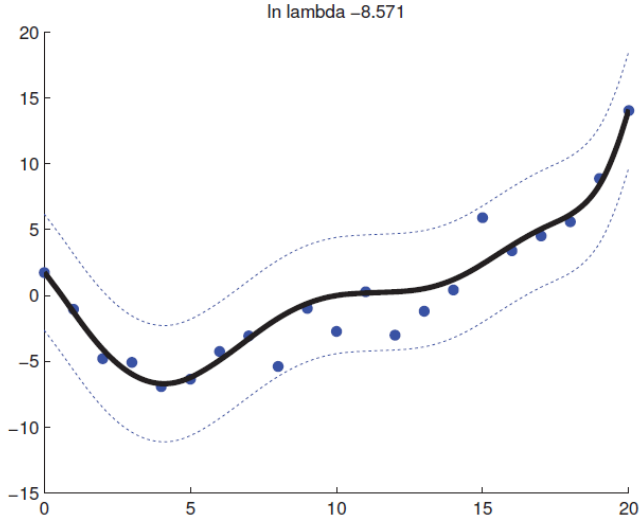
  2.128, 0.807, 16.457, 3.704, -24.948, -10.472, -2.625,
  4.360, 13.711, 10.063, 8.716, 3.966, -9.349, -9.232

Ridge regression - Example

- (right) Training and test errors for a degree 14 polynomial fit plotted vs $\log(\lambda)$
  - Data was generated from noise with variance $\sigma^2 = 4$
- Two examples of fitted curves for different λ values





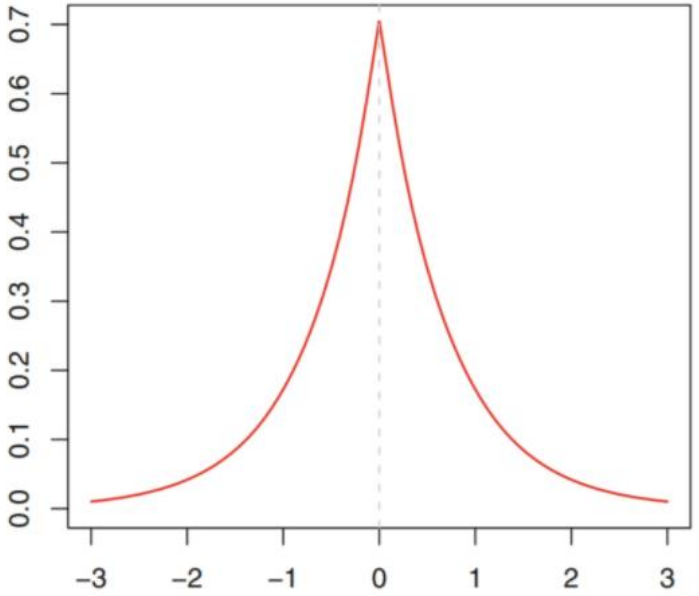


Lasso Regression

- Another form of regularization is L_1 **regularization** – which encourages parameters to be small (or even zero)
- The prior is a Laplace distribution with mean zero, so the resulting error function becomes

$$J(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + w^T x_i))^2 + \lambda||w||$$

- For LASSO, the prior distribution peaks at zero, therefore expects (a priori) many of the coefficients to be exactly equal to zero
- Alternatively, for ridge regression, the prior distribution is flatter at zero



Simple linear regression

- When the input is one dimensional, we call the linear regression simple
- The goal is just to find an estimate of the scalar function $y = f(x)$
- The solution can be calculated as:

$$w_1 = \frac{\sum_i (x_i - \overline{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{\sum_i x_i y_i - N\overline{x}\,\overline{y}}{\sum_i x_i^2 - N\overline{x}^2} \approx \frac{\mathrm{cov}\,[X, Y]}{\mathrm{var}\,[X]}$$

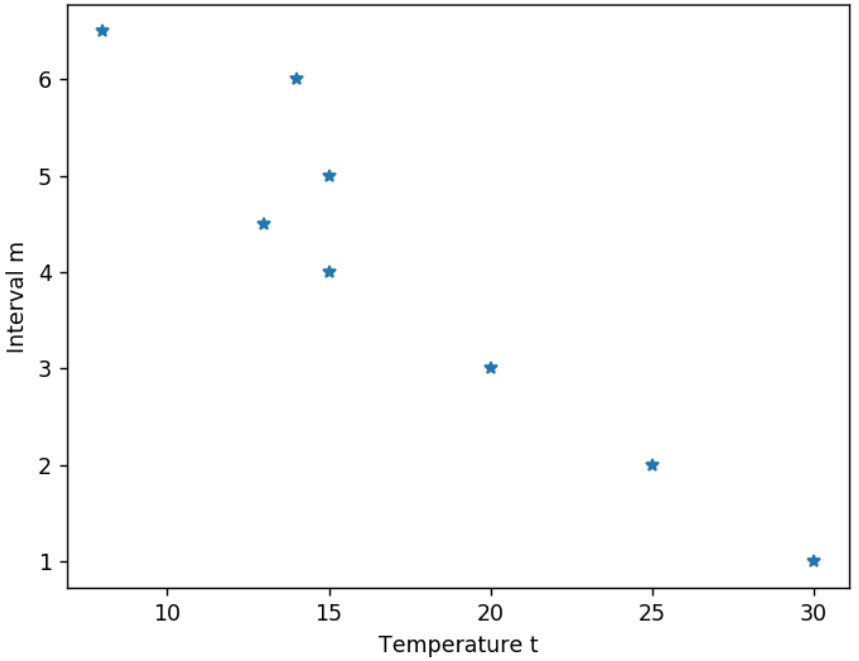
$$w_0 = \bar{y} - w_1 \bar{x} \approx \mathbb{E}\,[Y] - w_1 \mathbb{E}\,[X]$$

Exercise 1

- A biologist studies the intervals (*m secs*) between the mating calls of a certain species of tree frog and the surrounding temperature (*t °C*), and have obtained these observations:

t °C	8	13	14	15	15	20	25	30
m secs	6.5	4.5	6	5	4	3	2	1

- The temperature t: x
  - Input (independent) variable
- The interval m: y
  - Target (dependent) variable
- Find $y = w_0 + w_1 x$ (find w_0, w_1)



Exercise 1

- The first approach to calculate this is to first calculate the summary statistics of the data and use them in the simple linear regression model solution

$$
\begin{aligned}
w_1 &= \frac{\sum_i (x_i - \overline{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{\sum_i x_i y_i - N\overline{x}\,\overline{y}}{\sum_i x_i^2 - N\overline{x}^2} \approx \frac{\operatorname{cov}[X, Y]}{\operatorname{var}[X]} \\
w_0 &= \bar{y} - w_1 \bar{x} \approx \mathbb{E}[Y] - w_1 \mathbb{E}[X]
\end{aligned}
$$

- Therefore, we have:

$$\bar{x} = 17.5 \qquad \bar{y} = 4.0 \qquad \bar{x}^2 = 306.25 \qquad \sum_i x_i y_i = 469.5 \qquad \sum_i x_i^2 = 2804$$

- Using this we can calculate the upper equations to get

$$w_1 = -0.2556 \qquad w_0 = 8.4739 \qquad \hat{y} = -0.2556x + 8.4739$$

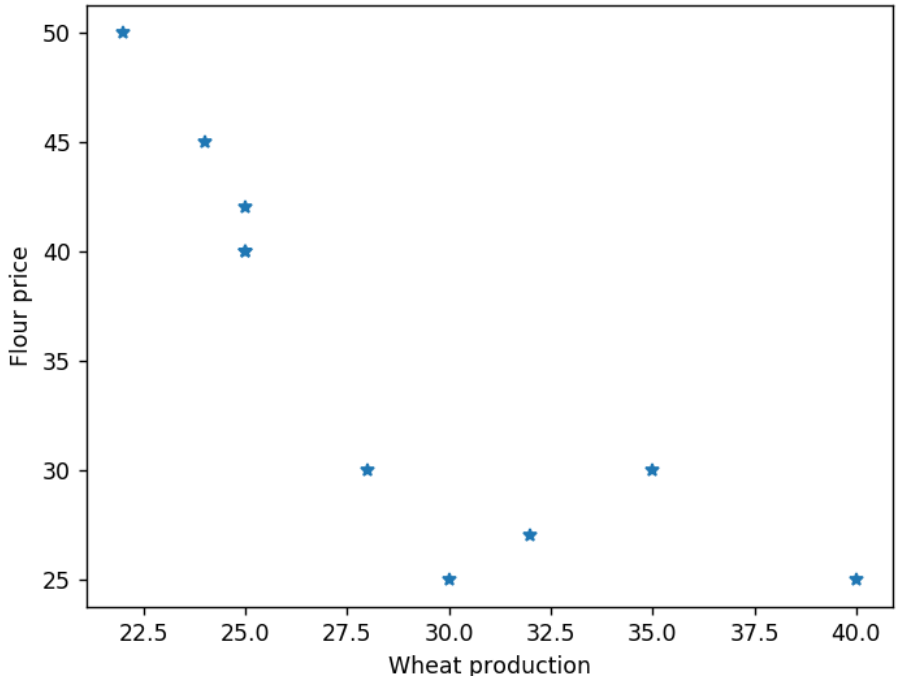
- Once we have the model parameters, we can use them to make predictions, like what is the interval when the t = 17 ℃

$$\hat{y} = -0.2556 * 17 + 8.4739 = 4.1287$$

Exercise 2

- Data represents the production of wheat in tons (X) and the price of the kilo of flour (Y):

Wheat production	30	28	32	25	25	25	22	24	35	40
Flour price	25	30	27	40	42	40	50	45	30	25



Exercise 2

- The sufficient statistics are:

$$\bar{x} = 28.6 \quad \bar{y} = 35.4 \quad \bar{x}^2 = 817.96 \quad \sum_i x_i y_i = 9734 \quad \sum_i x_i^2 = 8468$$

- Using those we can calculate the coefficients and the variance

$$\hat{w}_1 = \frac{\sum_{i=1}^{10} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{10} x_i^2 - n\bar{x}^2} = \frac{9734 - 10 * 28.6 * 35.4}{8468 - 10 * 28.6^2} = -1.3537$$

$$\hat{w}_0 = \bar{y} - \hat{w}_1 \bar{x} = 35.4 + 1.3537 * 28.6 = 74.116$$

- Predict y for x = 45

$$\hat{y} = -1.3537 * 45 + 74.116 = 13.1995$$

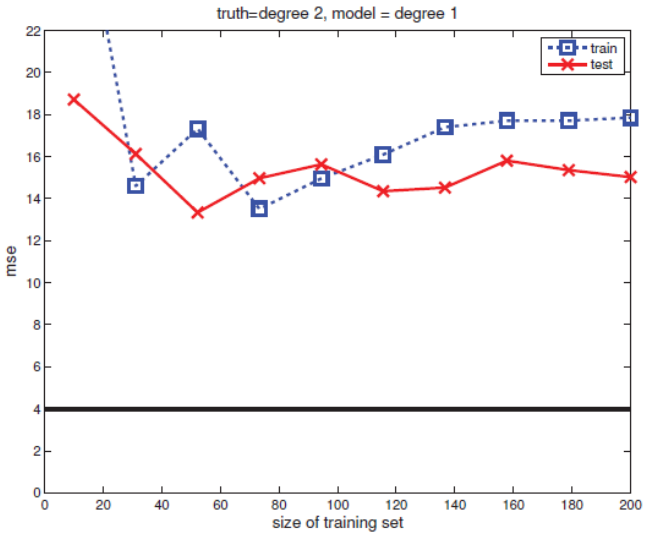
Regularization effects of big data

- Regularization is the most common way to avoid overfitting
- Another effective approach is to use lots of data
  - The more training data we have, the better we will learn
  - We expect the test set error to decrease to some plateau as N increases.
  - It will typically go to zero faster for simpler models, since there are fewer parameters to estimate.
- In domains with lots of data, simple methods can work very well, however, more sophisticated learning methods are required for problems with little available data
  - Even in data-rich domain as web search, for personalized results there are not much data available

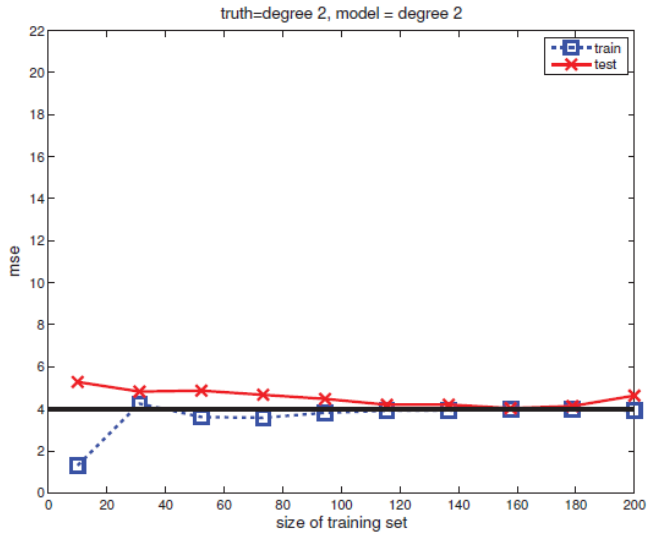
Polynomial fitting - example

The **truth** is a polynomial of degree 2

The **models** are of degrees 1, 2, 10 and 25



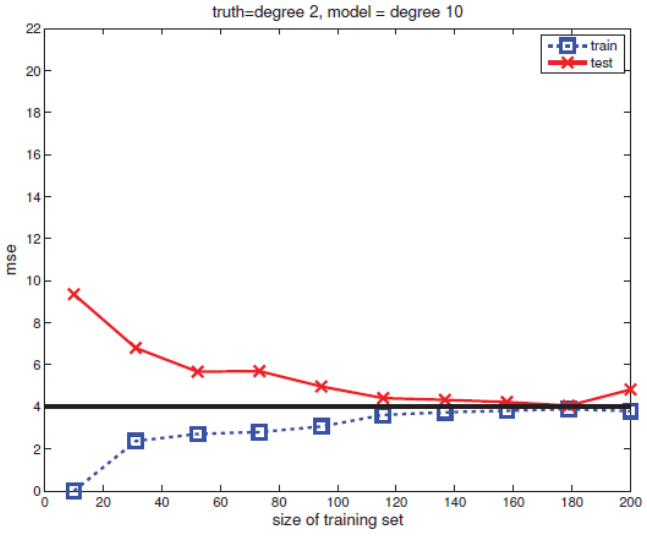
(a)



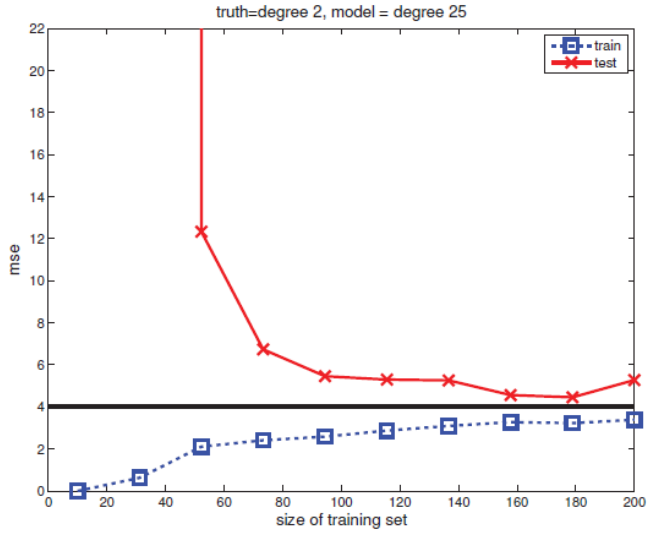
(b)

Degrees 1 and 25 give **poor** fit

Degrees 2 and 10 give **good fit**



(c)



(d)

Bayesian linear regression

- Instead of a point estimate, sometimes we want to find the full posterior over $\mathbf{w}$ and σ^2
  - For simplicity, we will assume the noise variance σ^2 is known, so we focus on finding $p(\mathbf{w}|D, \sigma^2)$
  - We assume a Gaussian likelihood model and a Gaussian conjugate prior, so the posterior is

$$
\begin{aligned}
p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma^2) &\propto \mathcal{N}(\mathbf{w}|\mathbf{w}_0, \mathbf{V}_0)\mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I}_N) = \mathcal{N}(\mathbf{w}|\mathbf{w}_N, \mathbf{V}_N) \\
\mathbf{w}_N &= \mathbf{V}_N\mathbf{V}_0^{-1}\mathbf{w}_0 + \frac{1}{\sigma^2}\mathbf{V}_N\mathbf{X}^T\mathbf{y} \\
\mathbf{V}_N^{-1} &= \mathbf{V}_0^{-1} + \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X} \\
\mathbf{V}_N &= \sigma^2(\sigma^2\mathbf{V}_0^{-1} + \mathbf{X}^T\mathbf{X})^{-1}
\end{aligned}
$$

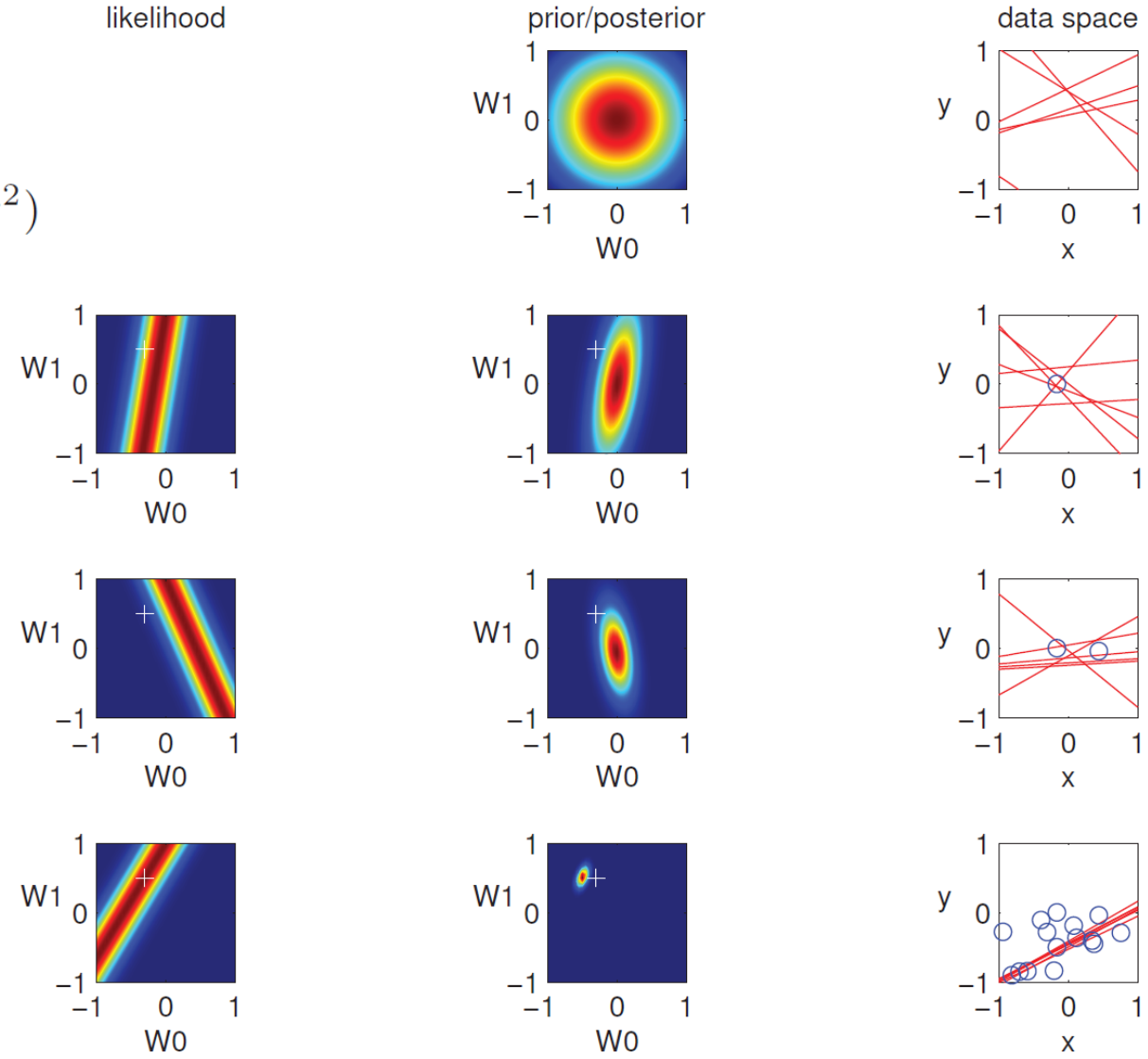
Bayesian linear regression

- 1D example

$$y(x, \mathbf{w}) = w_0 + w_1 x + \epsilon$$

$$p(y|\mathbf{x}) = \mathcal{N}(y|w_0 x_0 + w_1 x_1, \sigma^2)$$

- Sequential Bayesian updating after seeing 1, 2 and 20 data points



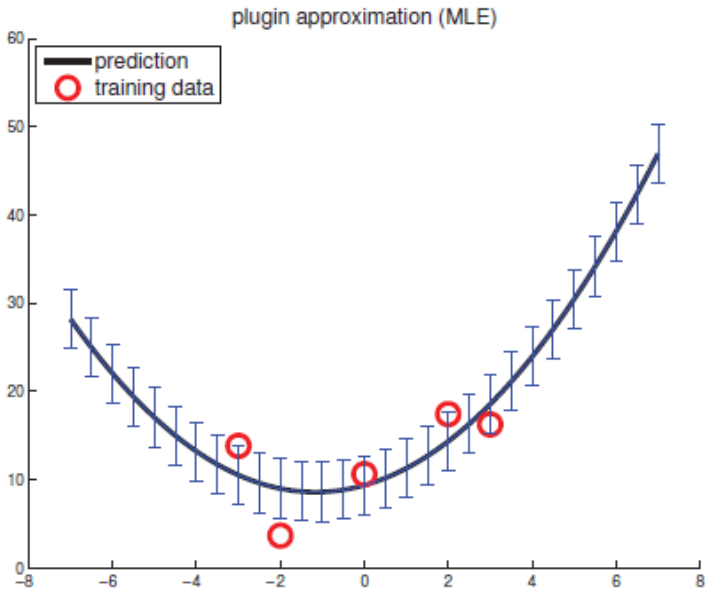
Bayesian linear regression

- The **posterior predictive** can be expressed as

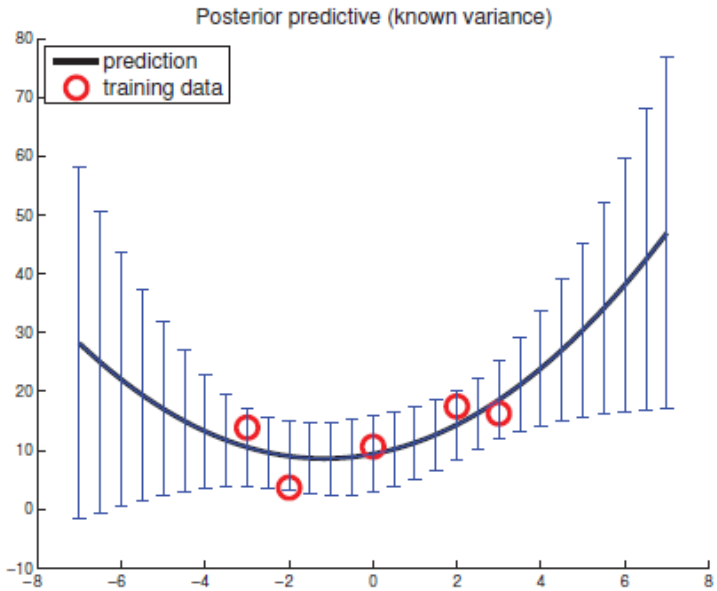
$$\begin{aligned} p(y|\mathbf{x}, \mathcal{D}, \sigma^2) &= \int \mathcal{N}(y|\mathbf{x}^T\mathbf{w}, \sigma^2)\mathcal{N}(\mathbf{w}|\mathbf{w}_N, \mathbf{V}_N)d\mathbf{w} \\ &= \mathcal{N}(y|\mathbf{w}_N^T\mathbf{x}, \sigma_N^2(\mathbf{x})) \\ \sigma_N^2(\mathbf{x}) &= \sigma^2 + \mathbf{x}^T\mathbf{V}_N\mathbf{x} \end{aligned}$$

- On the other hand, with **MLE** we have

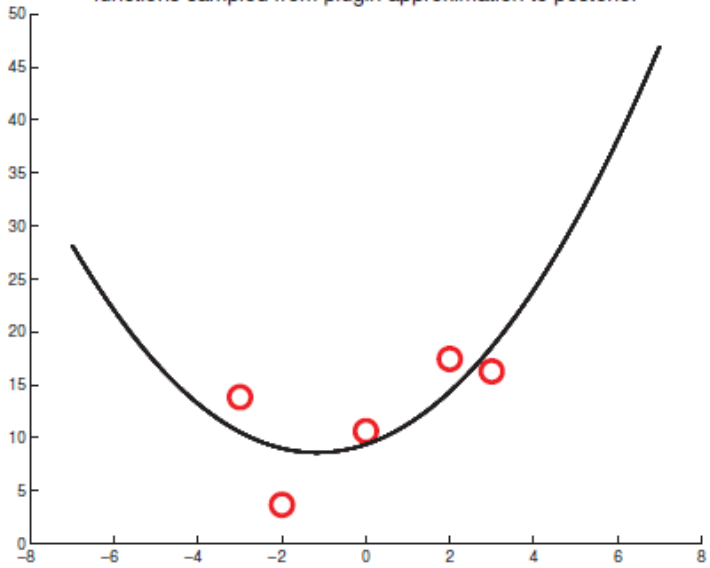
$$p(y|\mathbf{x}, \mathcal{D}, \sigma^2) \approx \int \mathcal{N}(y|\mathbf{x}^T\mathbf{w}, \sigma^2)\delta_{\hat{\mathbf{w}}}(\mathbf{w})d\mathbf{w} = p(y|\mathbf{x}, \hat{\mathbf{w}}, \sigma^2)$$


plugin approximation (MLE)
prediction
training data

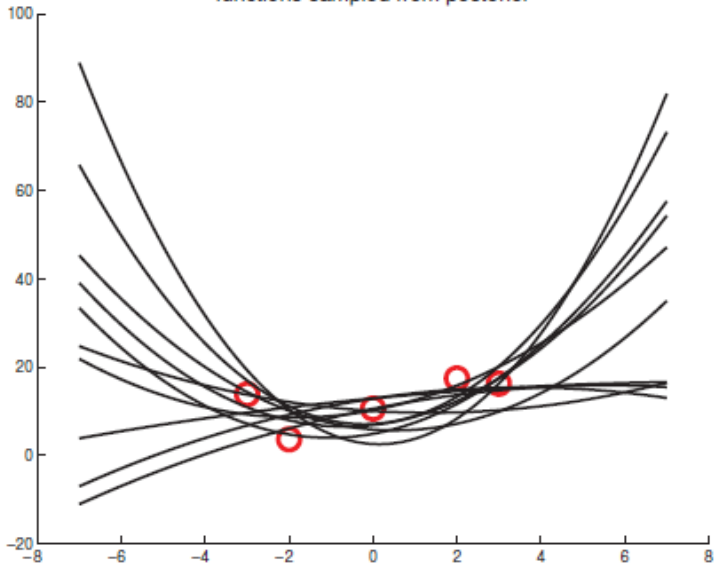
(a)


Posterior predictive (known variance)
prediction
training data

(b)


functions sampled from plugin approximation to posterior

(c)


functions sampled from posterior

(d)