

Bsc Data Science for Responsible Business

Deep Learning Course

Linear and Logistic Regression

Emmanuel Dellandrea - emmanuel.dellandrea@ec-lyon.fr

Introduction

Linear and Logistic regressions are both linear models

- Linear regression is dedicated to regression problems
- Logistic regression is dedicated to classification problems

Goal: Find a prediction function such that $y = f(x)$

y : target variable

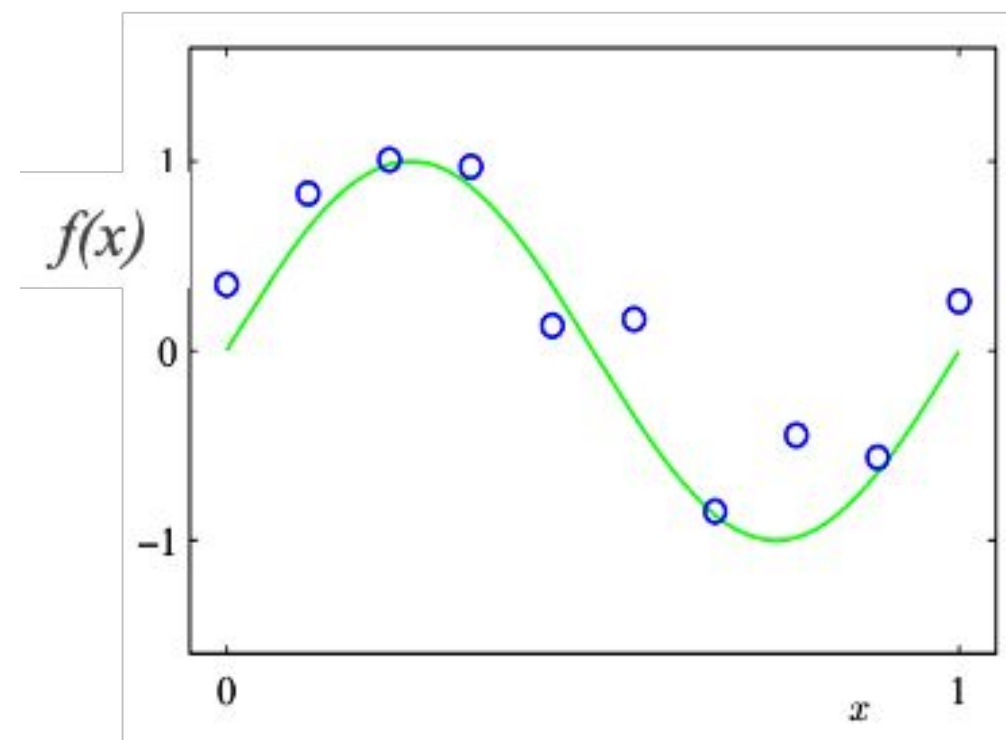
x : predictor variable (or explanatory variable)

f : prediction function

Regression vs classification

Regression: the target variable is quantitative (continuous)

Classification: the target variable is qualitative (discrete)



→ Regression

$f(\text{🍏}) = \text{"apple"}$
 $f(\text{🍅}) = \text{"tomato"}$
 $f(\text{🐮}) = \text{"cow"}$

→ Classification

$$y = f(x)$$

Training: Use a labeled dataset $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}$ in order to estimate the prediction function f by minimizing the prediction error

Test: Apply f function on x data that were not used during training to obtain predicted values

Prediction error:

- Compute the difference between the predicted values $f(x^{(i)})$ and the true values $y^{(i)}$
- This error computed on the whole test set allows to evaluate the model performance



Linear Regression

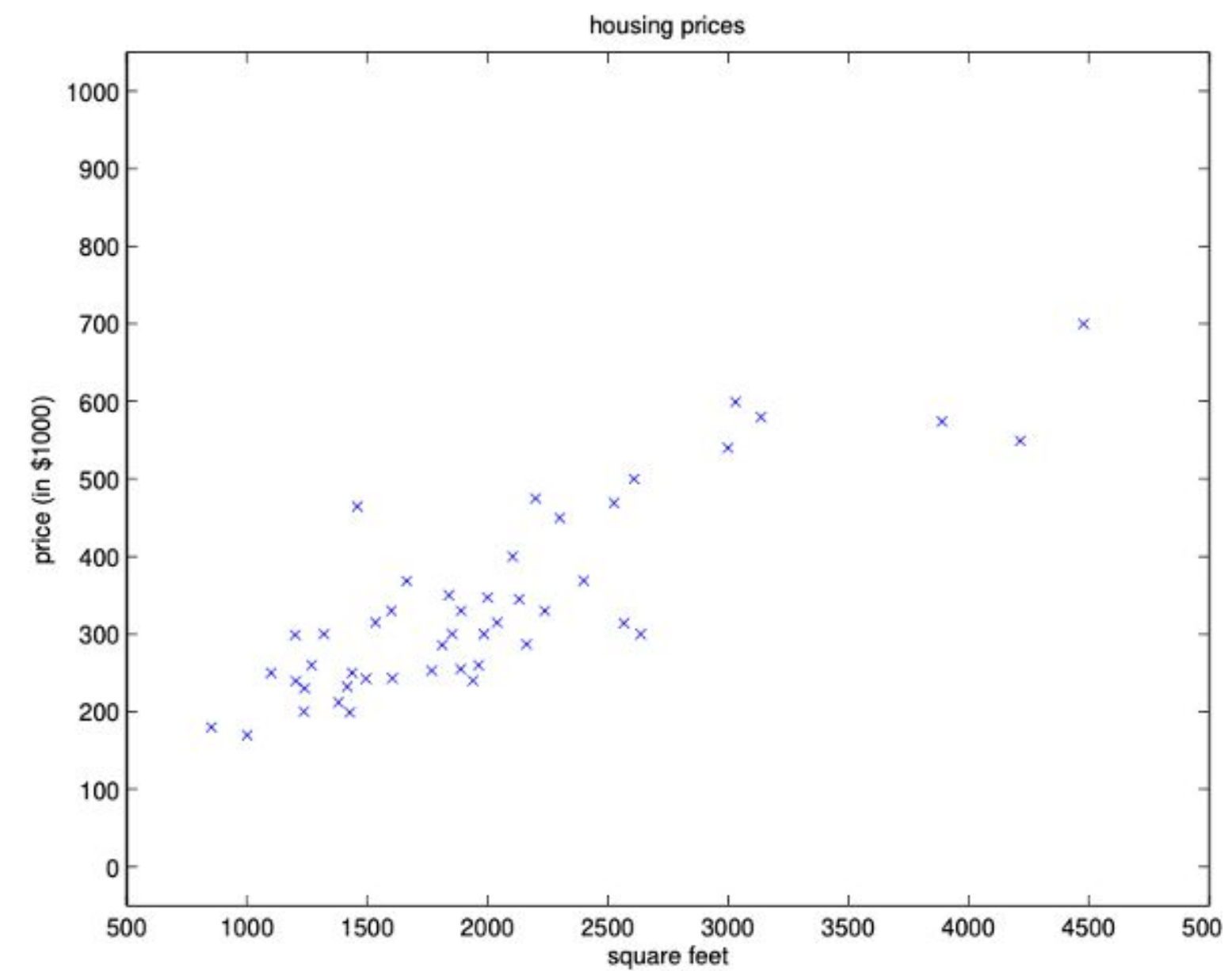
Some materials are borrowed from Andrew Ng's course on Machine Learning

Linear Regression

Goal: For a given example, predict the value of a the target variable using the known values of the predictor variable(s)

Living area (feet ²)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

$$f(2500) = ??$$



Linear Regression

Linear prediction function f

Living area (feet ²)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Predictor variables

Target variable

$$f_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Expression of the prediction function

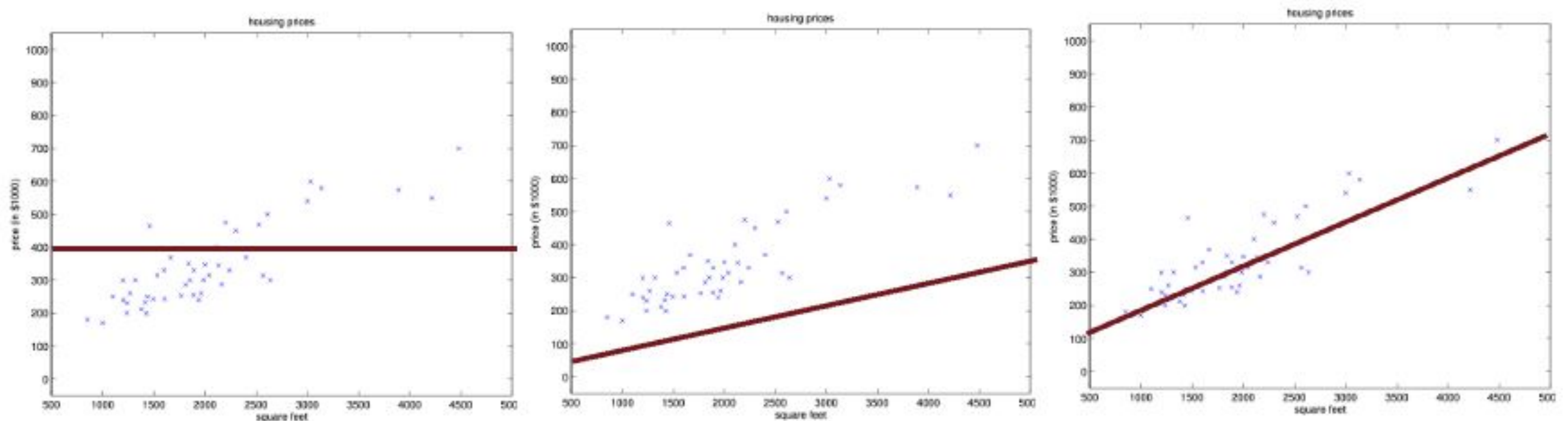
Notation:

$$\left| f(x) = \sum_{i=0}^d \theta_i x_i = \theta^T x \right|$$

With θ and x being vectors, d the number of predictor variables (without x_0) and $x_0 = 1$

Linear regression

How to estimate the value of the parameters θ ?



$$\theta_0 = 400, \theta_1 = 0$$

$$\theta_0 = 0, \theta_1 = 0.05$$

$$\theta_0 = 70, \theta_1 = 0.14$$

Linear regression

- Use of a training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}$ and choose θ such that $f(x)$ is as close as possible to y
- Definition of a loss function that measures the prediction error

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

Goal: find θ that minimizes $J(\theta)$

Gradient descent

Principle: Let $J(\theta)$ be a loss function. Find θ that minimizes $J(\theta)$.

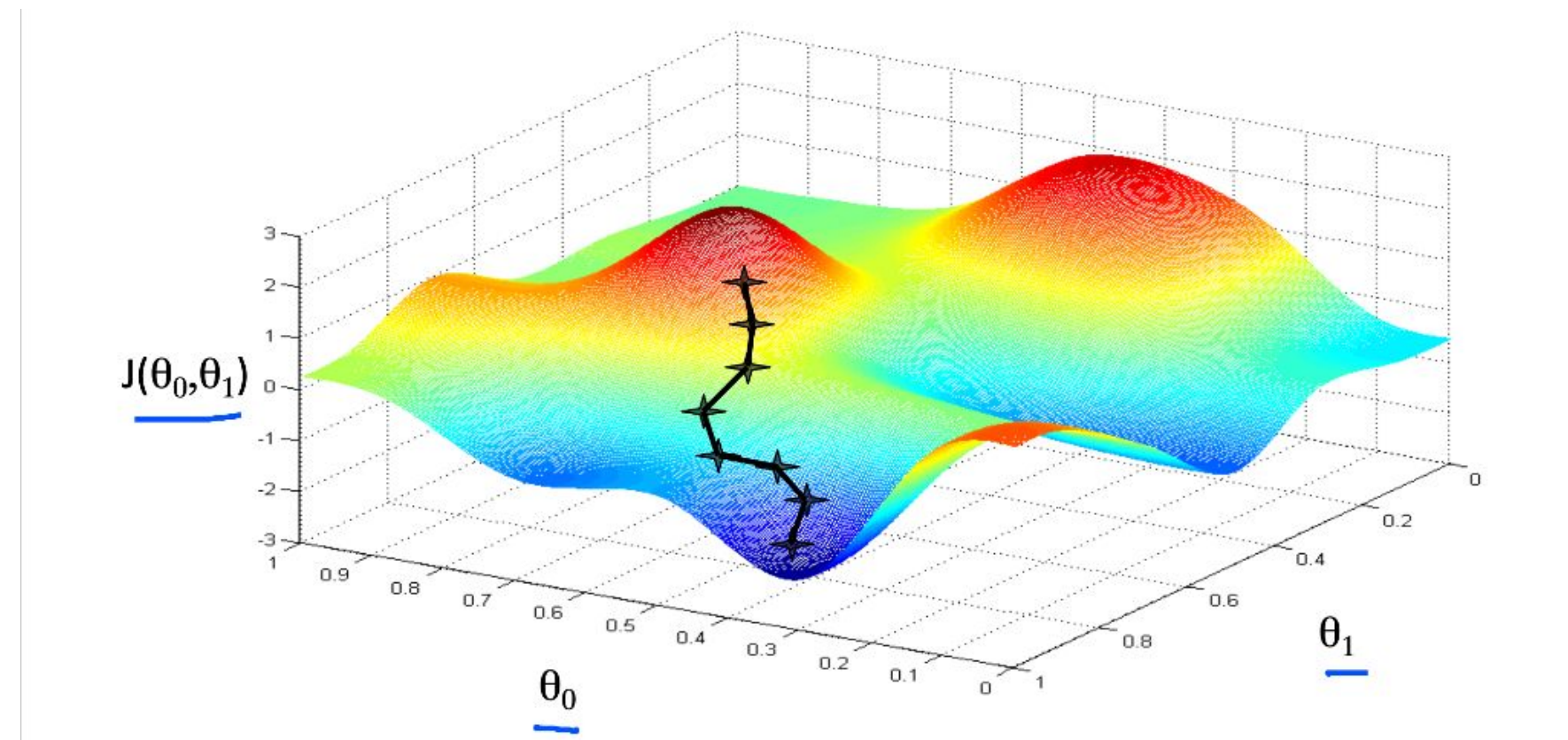
Algorithm:

Start with an arbitrary initialization of θ .

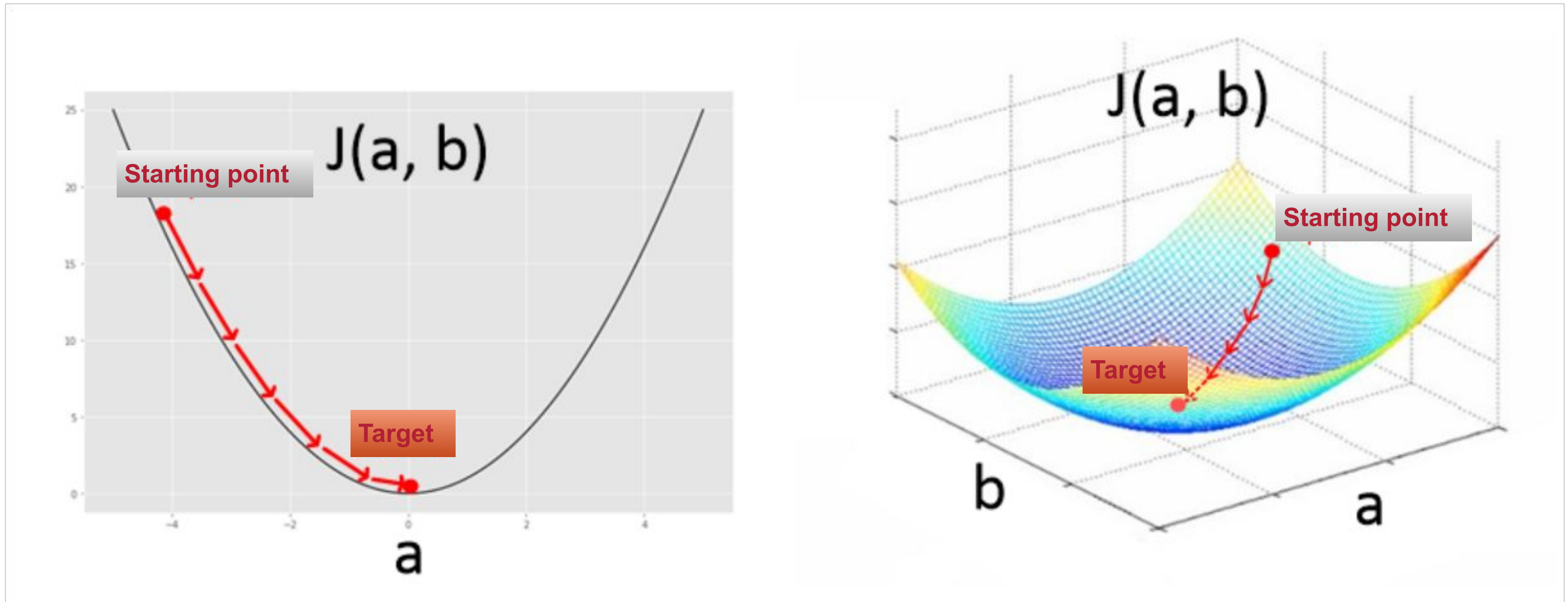
Gradually adjust θ to reduce the value of $J(\theta)$

How to adjust θ to reduce the value of $J(\theta)$?

→ Move in the opposite direction of the gradient (slope)



Gradient descent



Gradient descent : Algorithm

Iteratively repeat until convergence (simultaneously for each j)

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

Where α is the learning rate

Gradient computation

For one example (x,y)

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (f_\theta(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (f_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (f_\theta(x) - y) \\ &= (f_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^d \theta_i x_i - y \right) \\ &= (f_\theta(x) - y) x_j\end{aligned}$$

Gradient descent

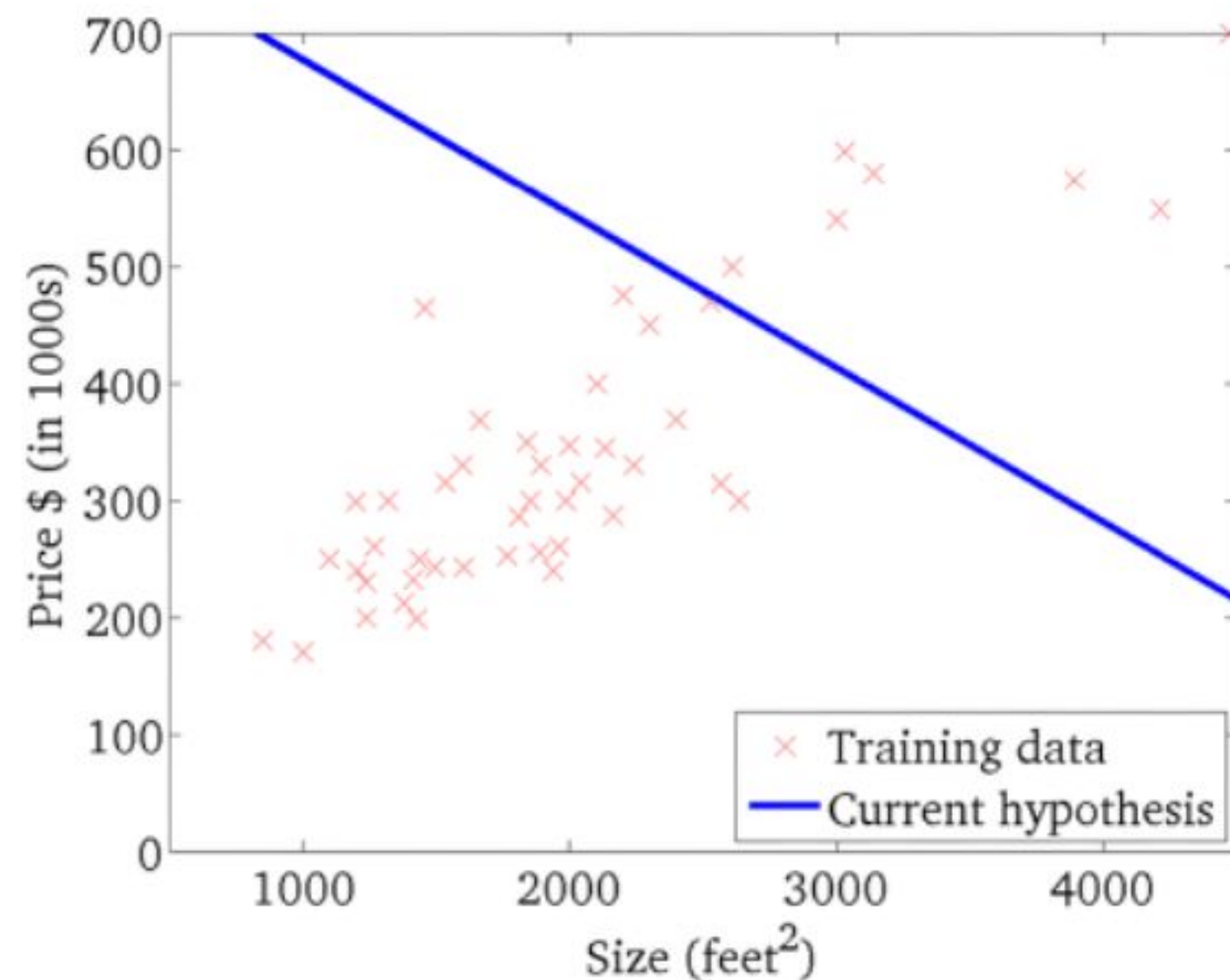
Iteratively repeat until convergence (simultaneously for each j)

$$\theta_j = \theta_j - \alpha \frac{1}{N} \sum_{i=1}^N (f_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Gradient descent: an example

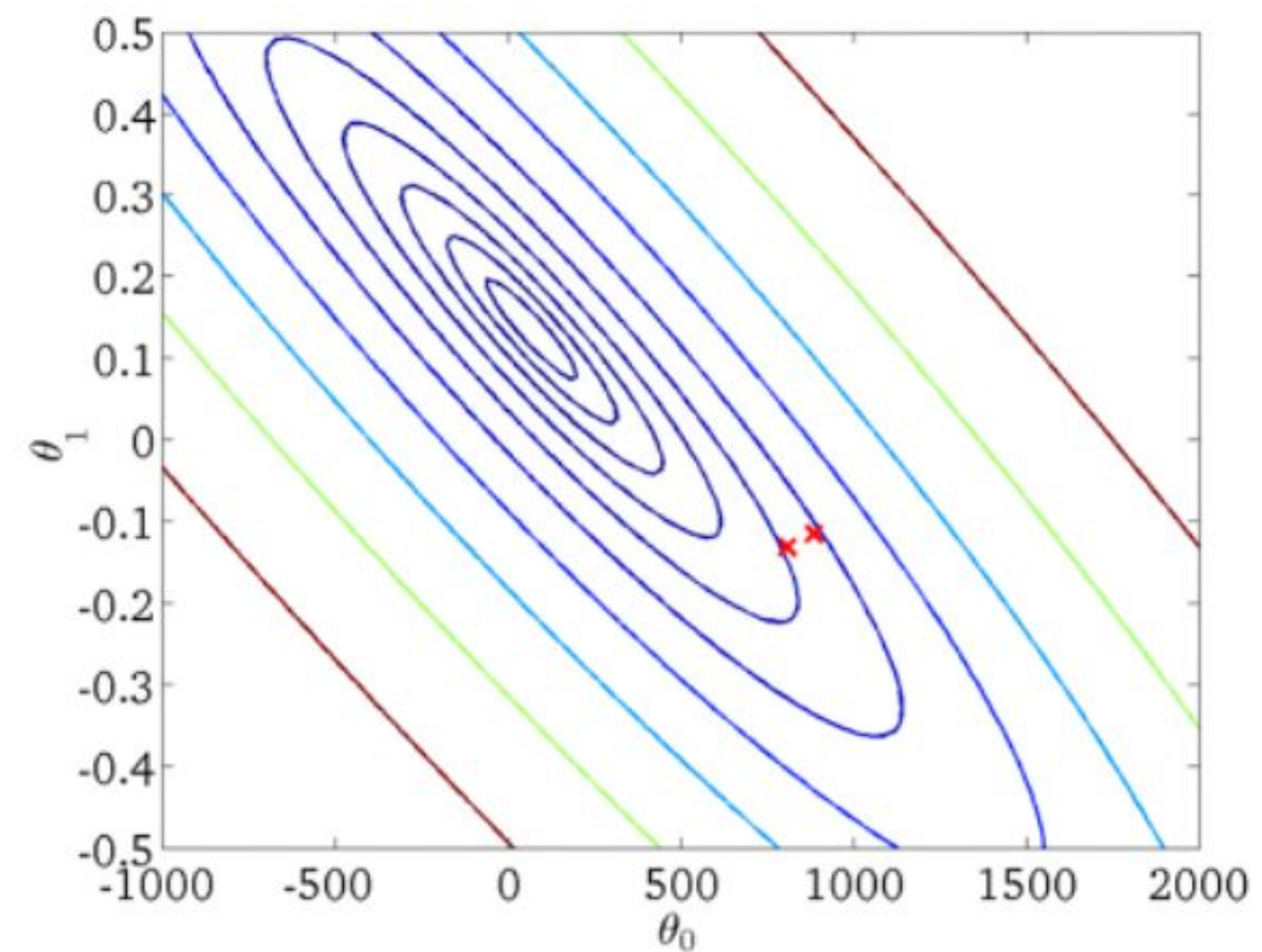
$$f_{\theta}(x)$$

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



$$J(\theta_0, \theta_1)$$

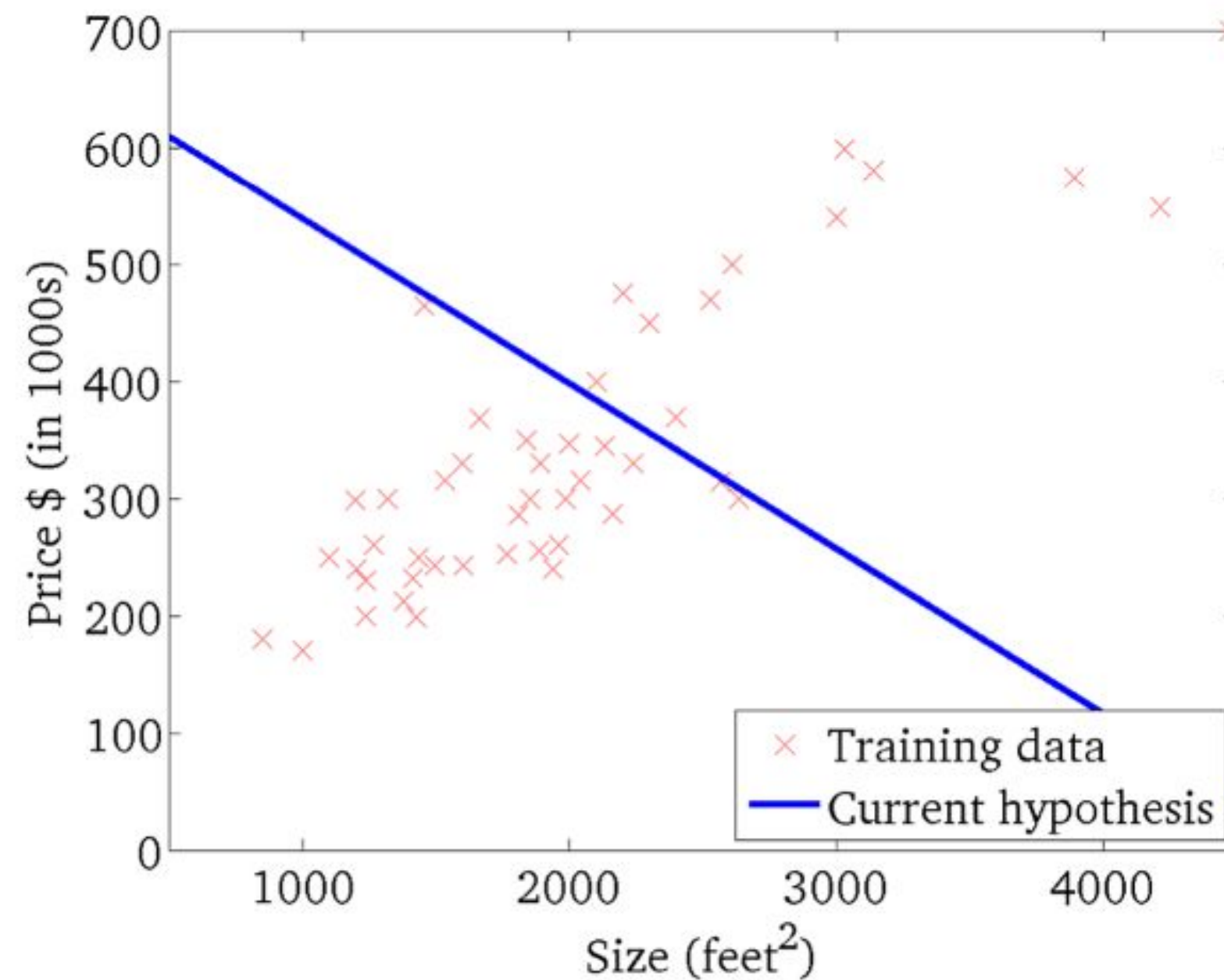
(function of the parameters θ_0, θ_1)



Gradient descent: an example

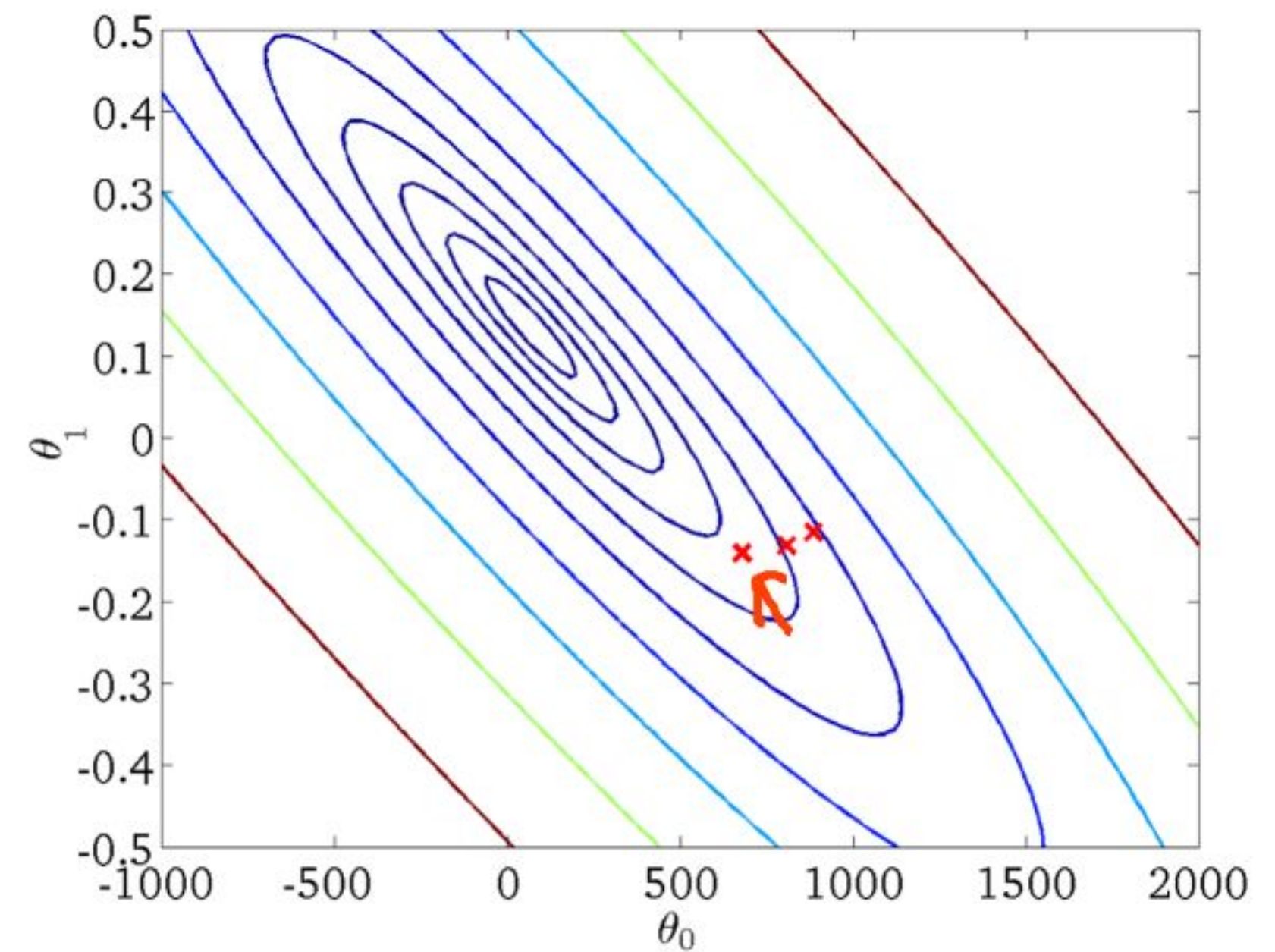
$$f_{\theta}(x)$$

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



$$J(\theta_0, \theta_1)$$

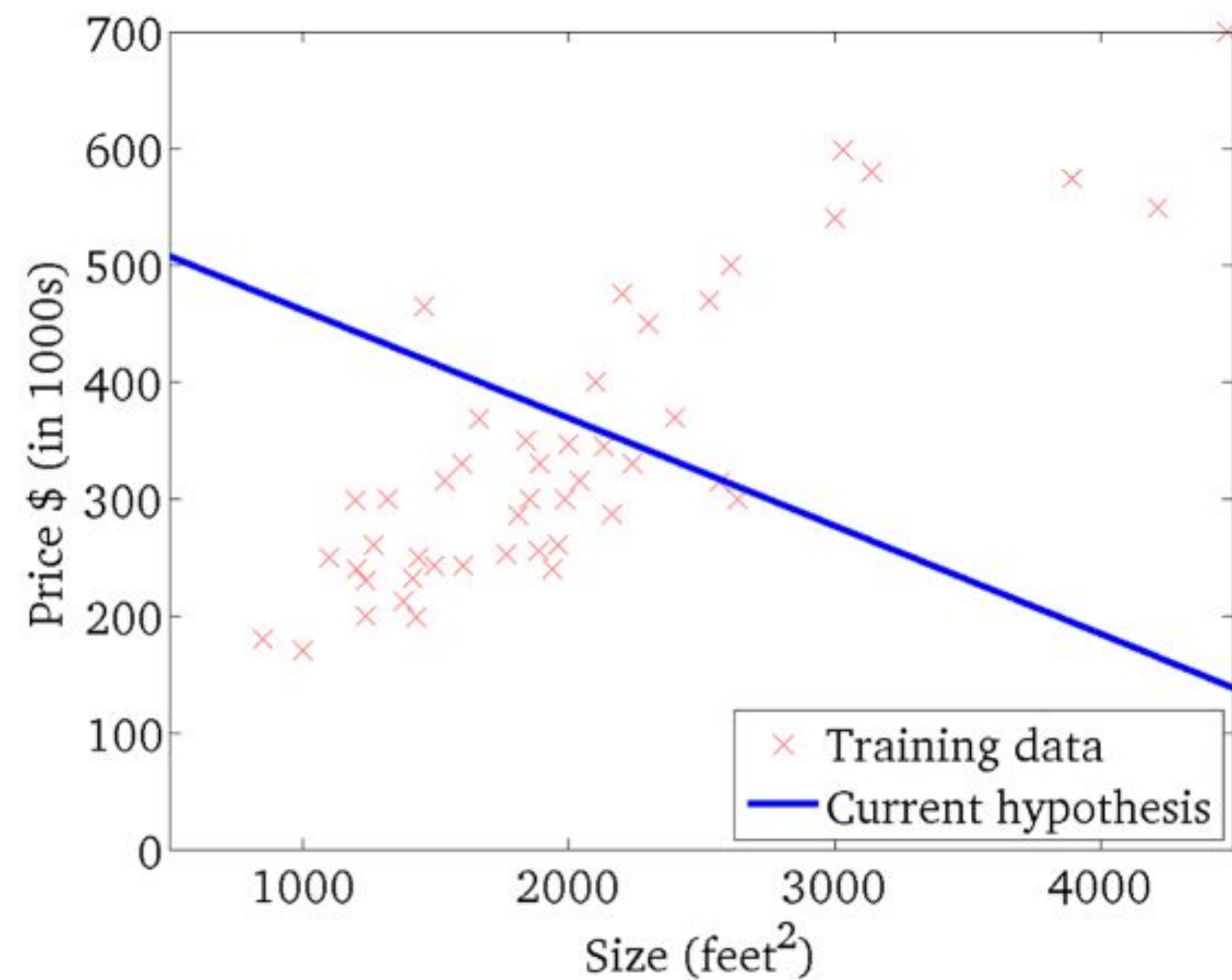
(function of the parameters θ_0, θ_1)



Gradient descent: an example

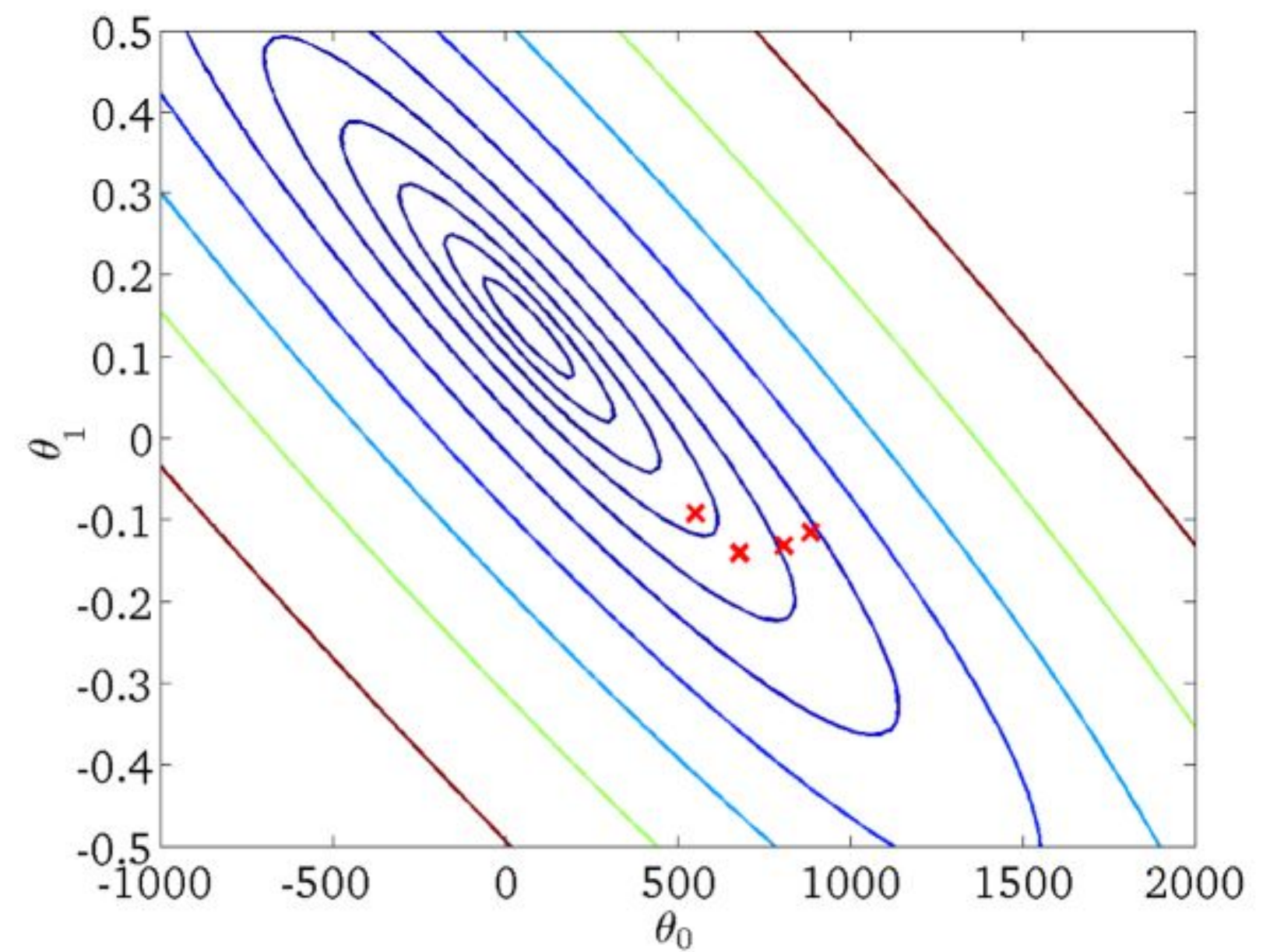
$$f_{\theta}(x)$$

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



$$J(\theta_0, \theta_1)$$

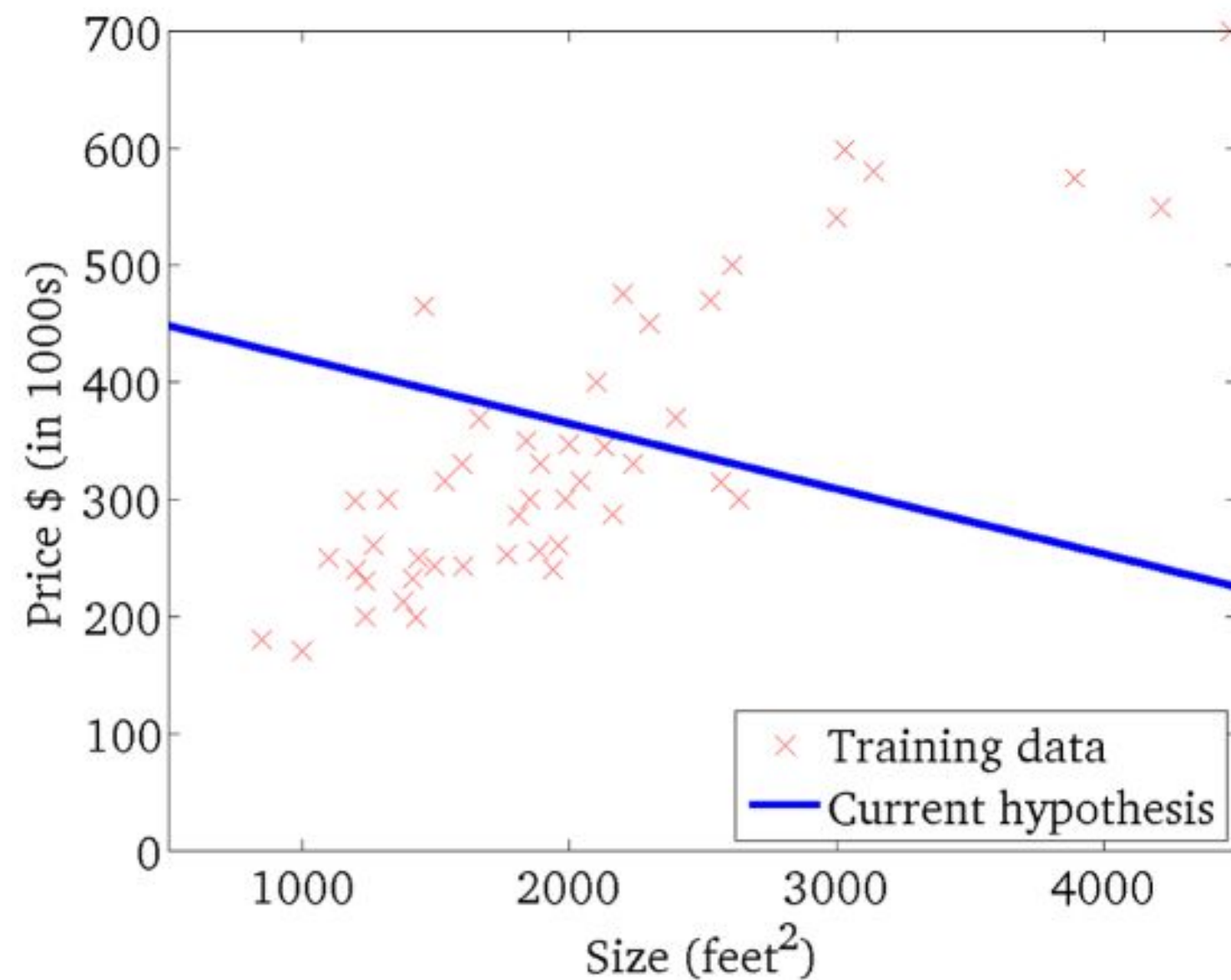
(function of the parameters θ_0, θ_1)



Gradient descent: an example

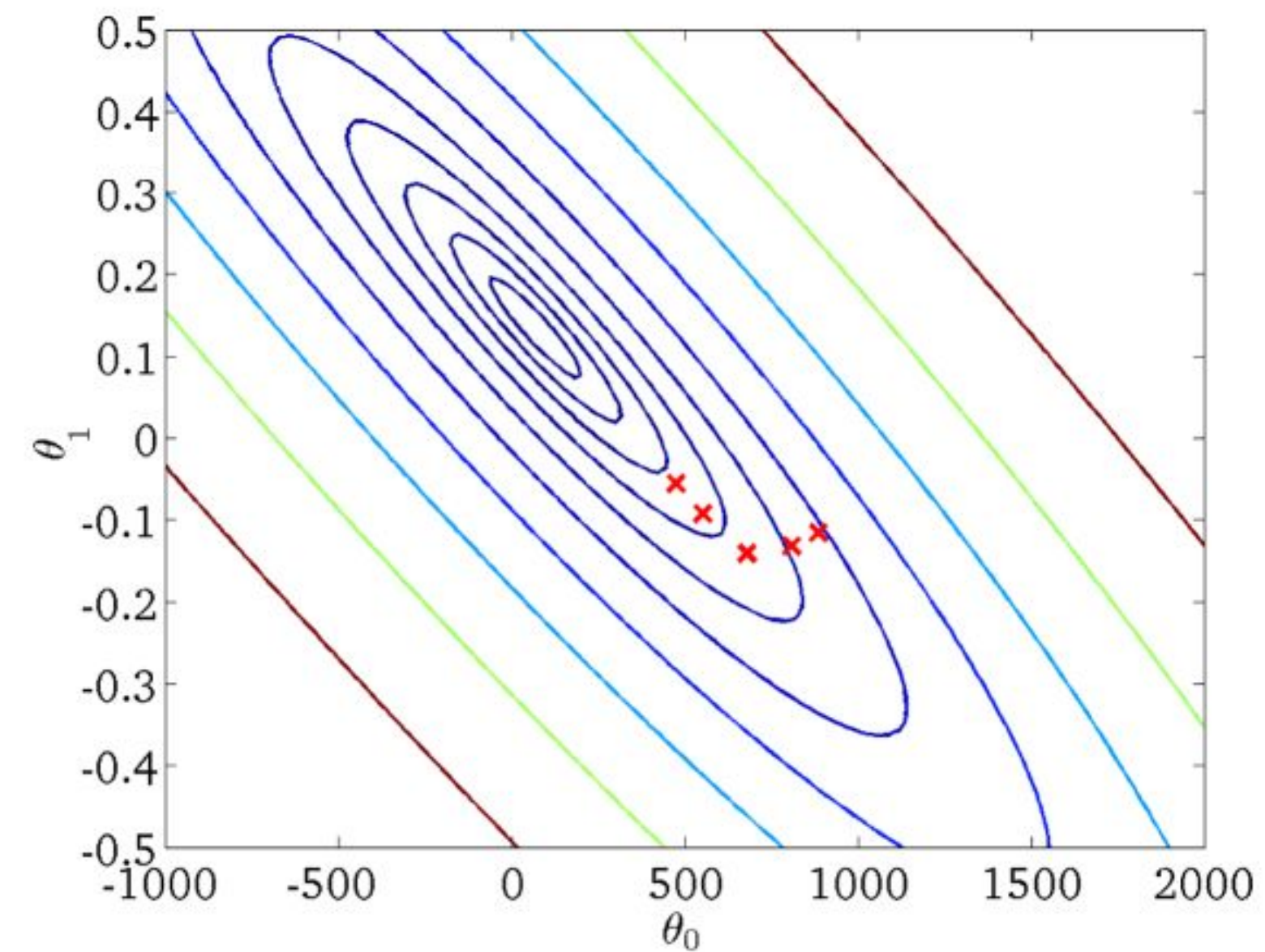
$$f_{\theta}(x)$$

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



$$J(\theta_0, \theta_1)$$

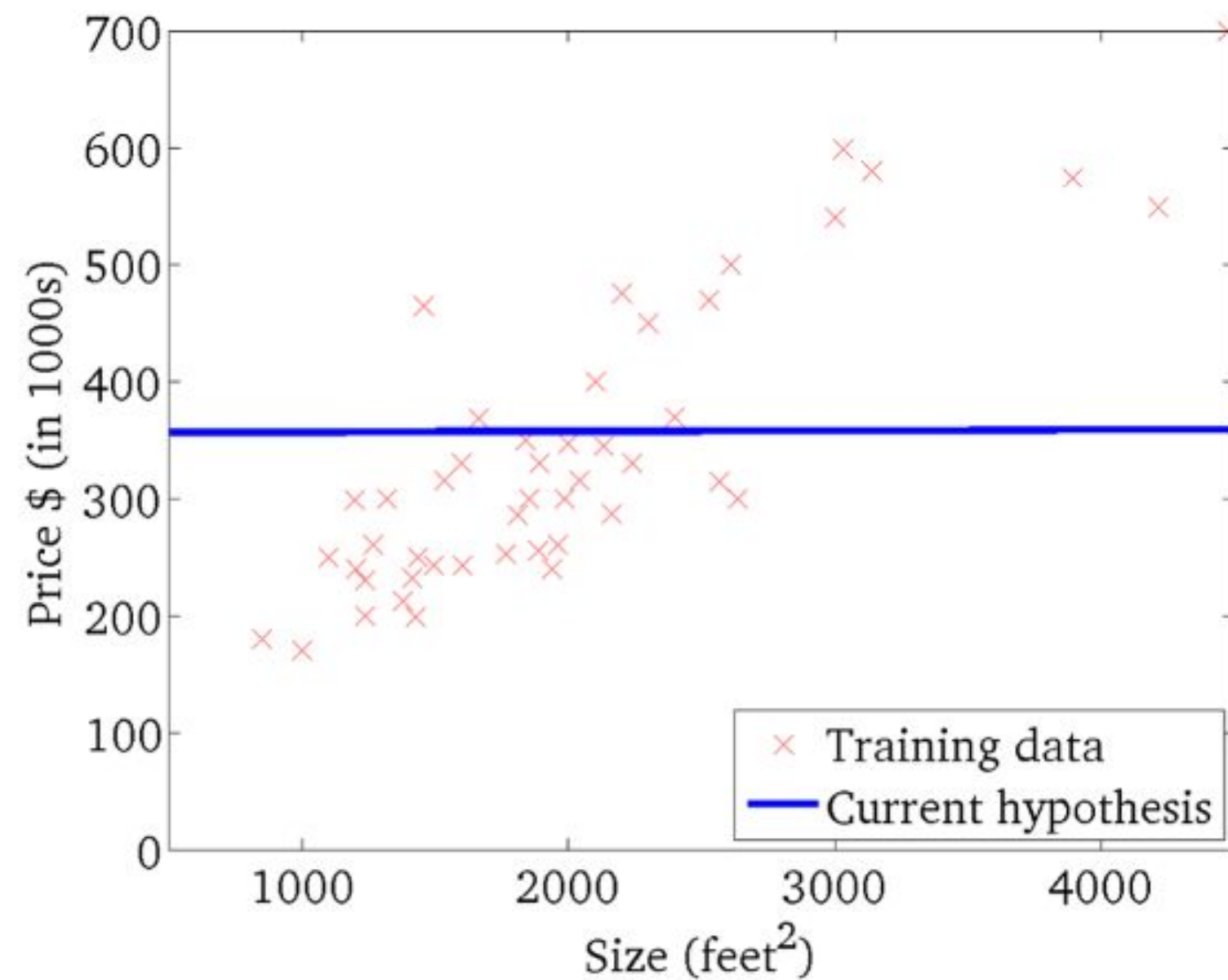
(function of the parameters θ_0, θ_1)



Gradient descent: an example

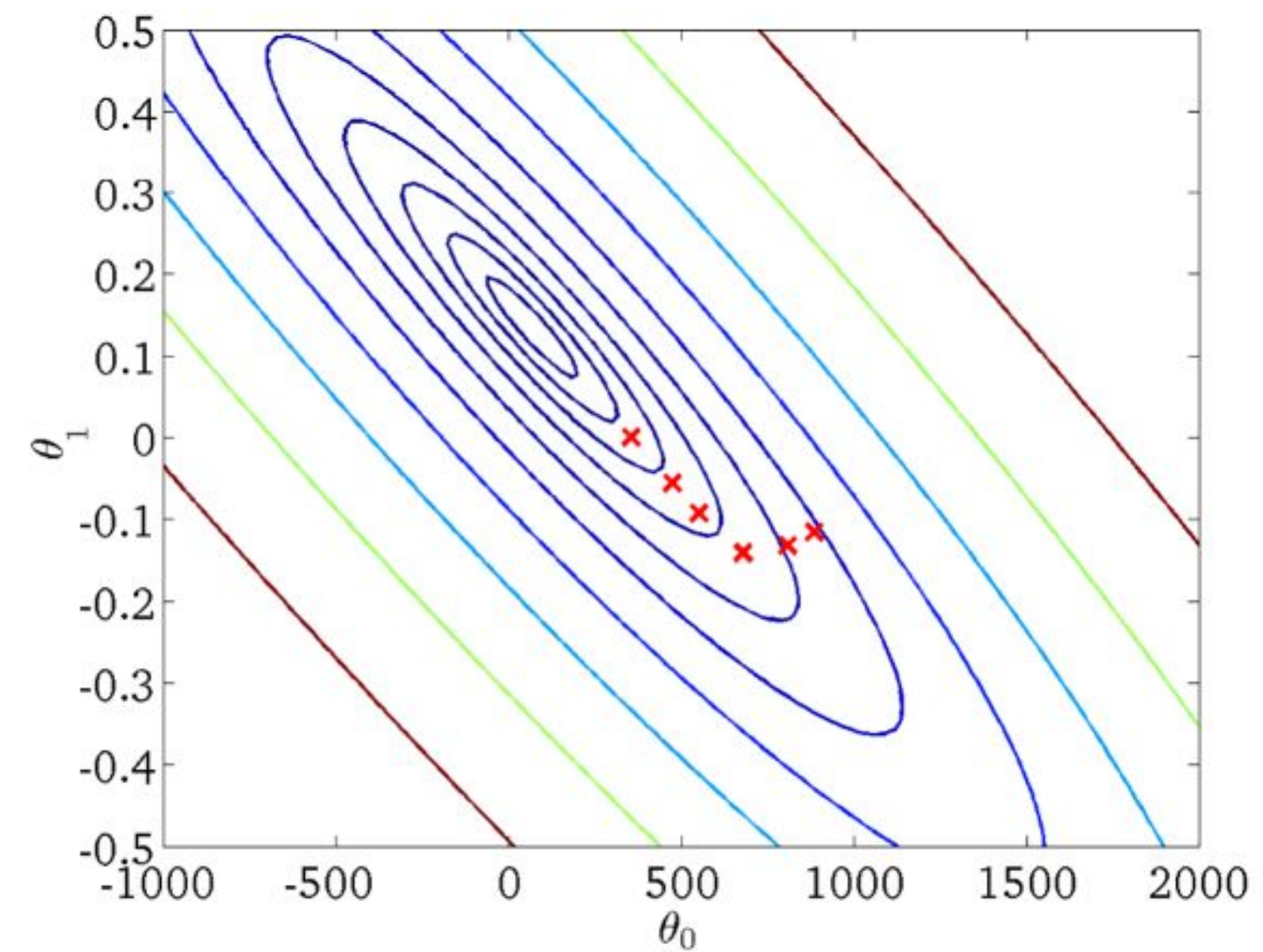
$$f_{\theta}(x)$$

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



$$J(\theta_0, \theta_1)$$

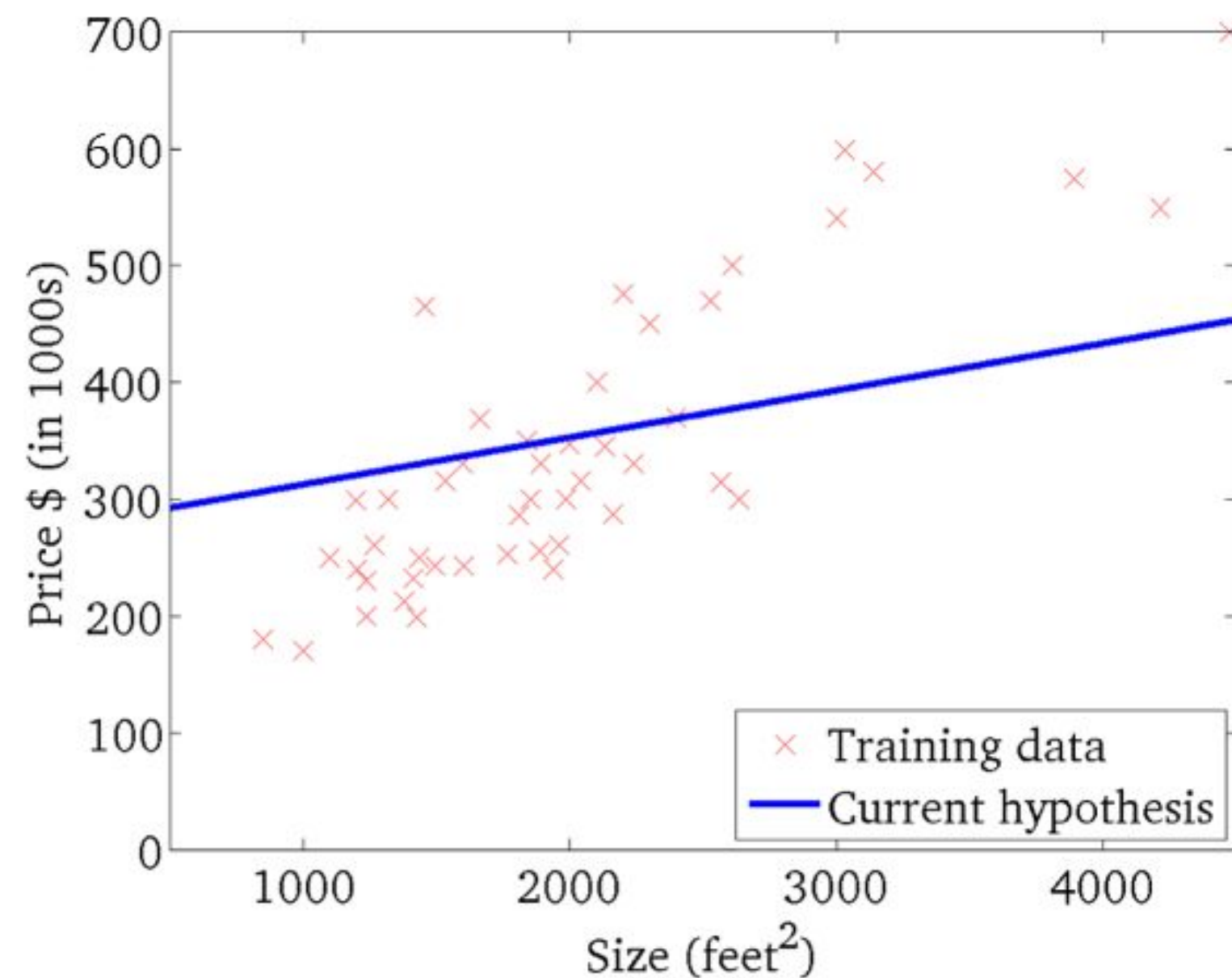
(function of the parameters θ_0, θ_1)



Gradient descent: an example

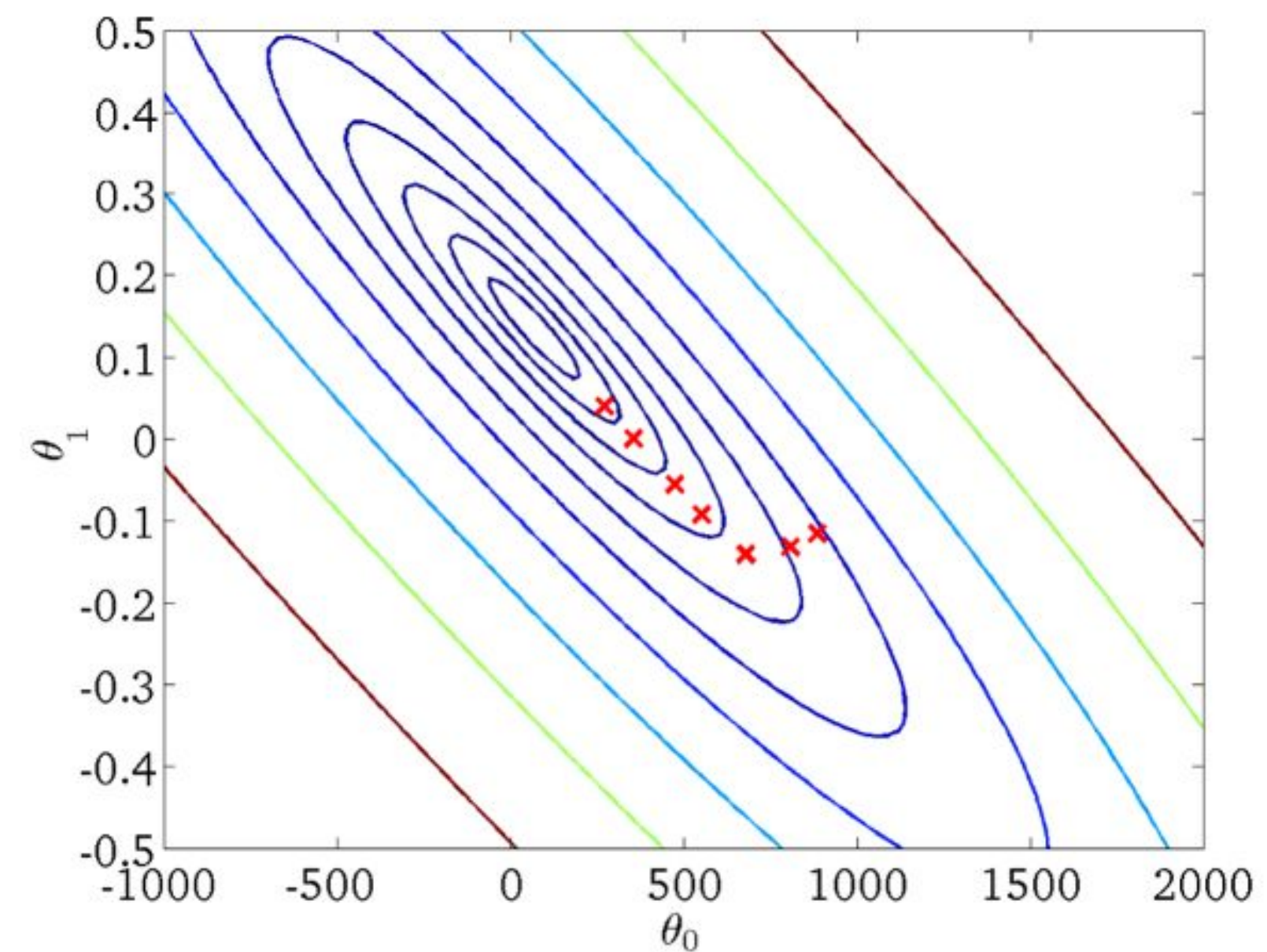
$$f_{\theta}(x)$$

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

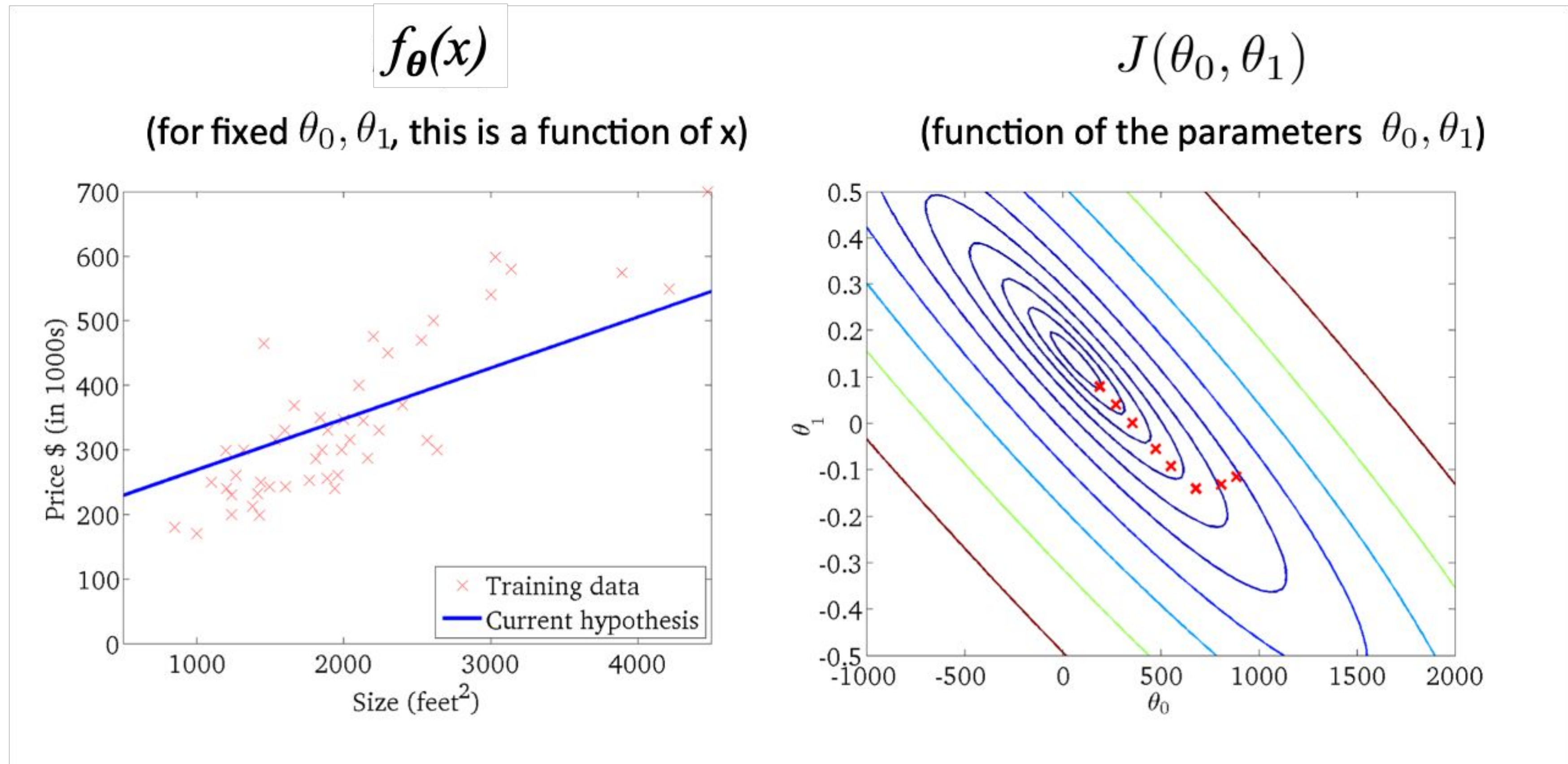


$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



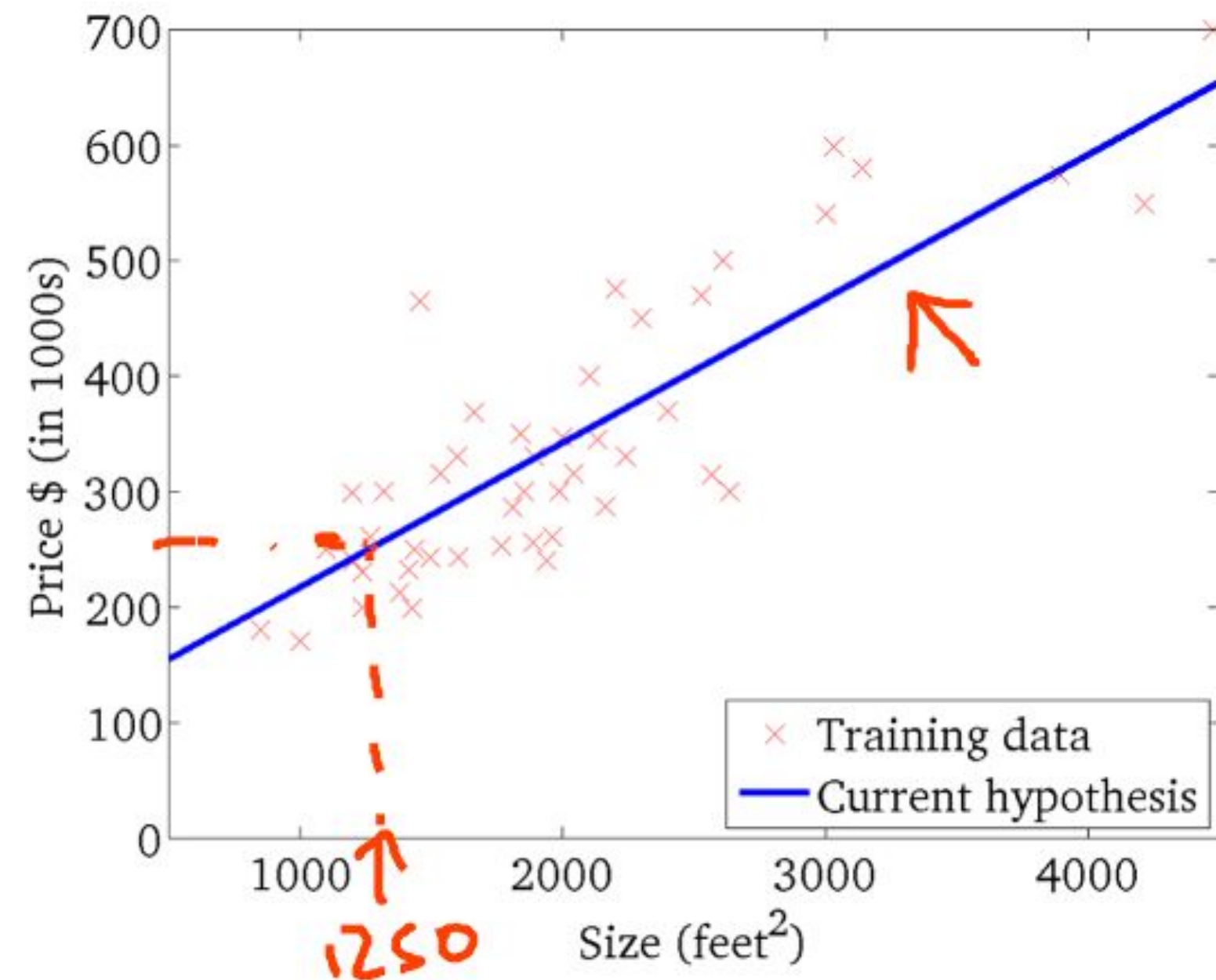
Gradient descent: an example



Gradient descent: an example

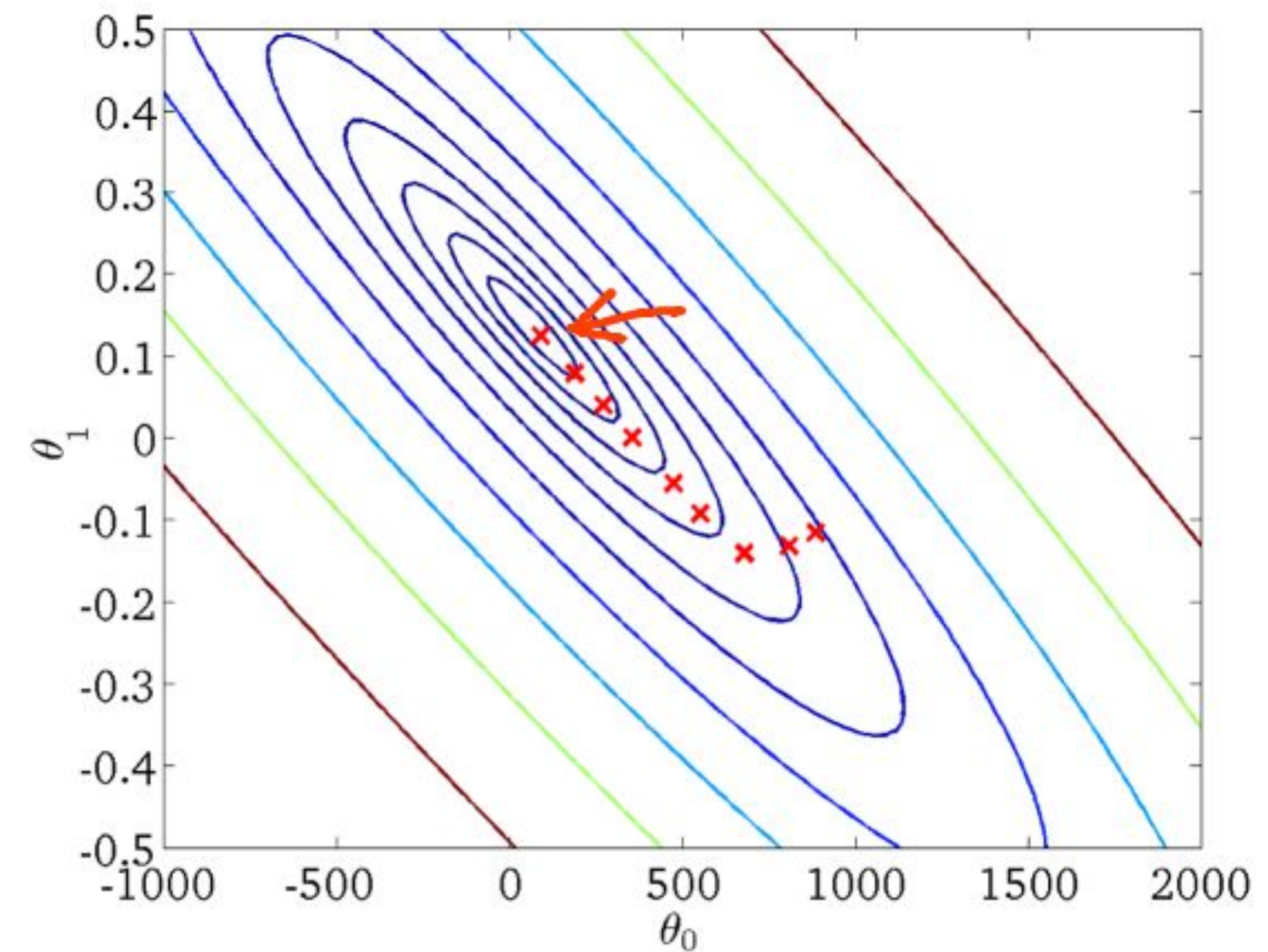
$$f_{\theta}(x)$$

(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)



Gradient descent



Recommendations:

- Normalize the predictor variables
- Choose the appropriate strategy for using training examples to update the gradient
- Select the learning rate carefully

Normalization of the predictor variables

Goal: Ensure that the magnitudes of all variables are similar

→ Normalized values:

$$x_i^k = \frac{x_i^k - \mu_i}{\sigma_i}$$

Where μ_i is the mean value of x_i variable over the training set, and σ_i the standard deviation

Strategy for gradient descent

Batch Gradient Descent (BGD): Computes the gradient of the entire dataset at once

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{N} \sum_{i=1}^N (f_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Stochastic Gradient Descent (SGD): Computes the gradient for a single training example at a time

$$\frac{\partial}{\partial \theta_j} J(\theta) = (f_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Mini-Batch Gradient Descent (MBGD): Computes the gradient over a small subset of b training examples of the dataset, with $b \ll N$

Strategy for gradient descent

Choose the right strategy:

- Batch Gradient Descent: When the dataset is small or can fit into memory, and stable convergence is critical.
- Stochastic Gradient Descent: When the dataset is very large and quick updates are required.
- Mini-Batch Gradient Descent: When the dataset is large, and a balance between convergence speed and computational efficiency is desired.

Choice of the learning rate

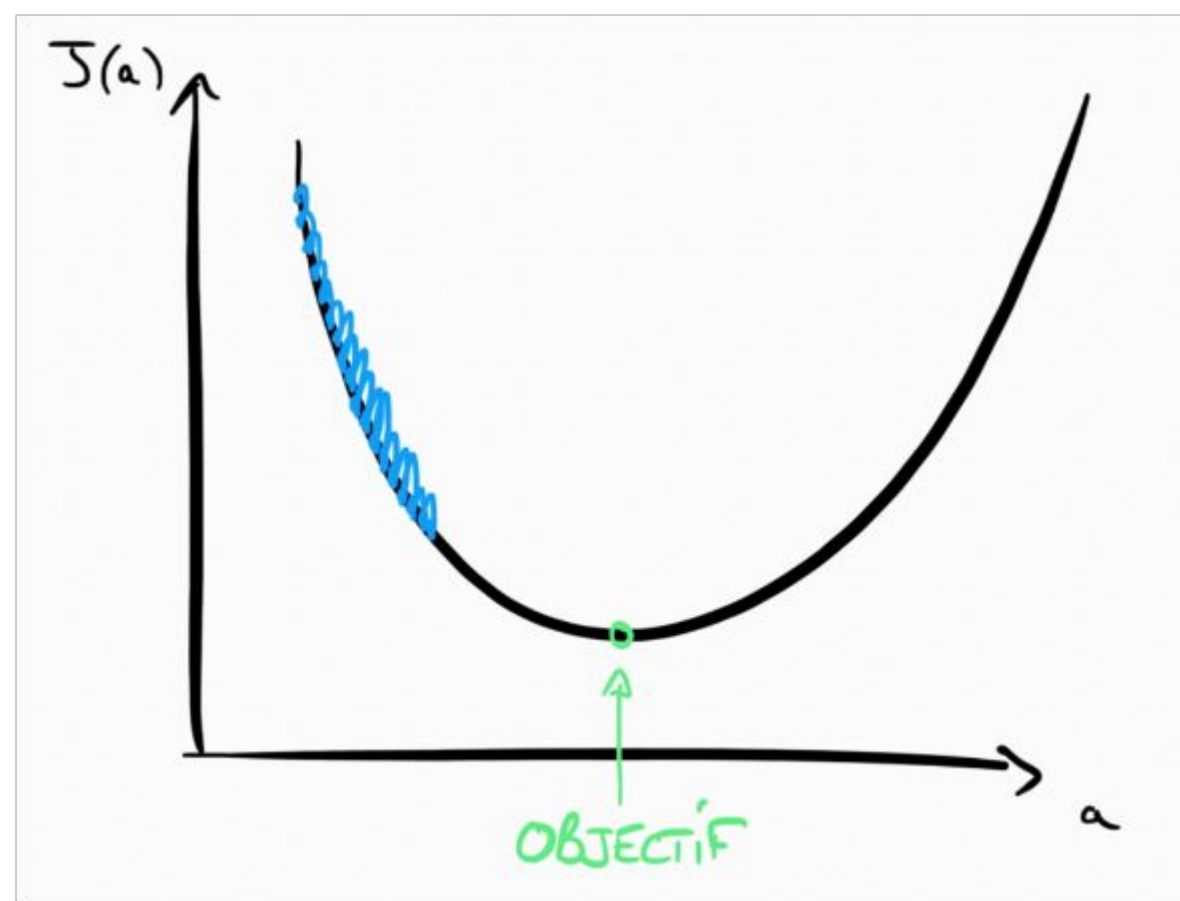
Remember: the parameters are updated according to

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

→ How to choose the learning rate α ?

Choice of the learning rate

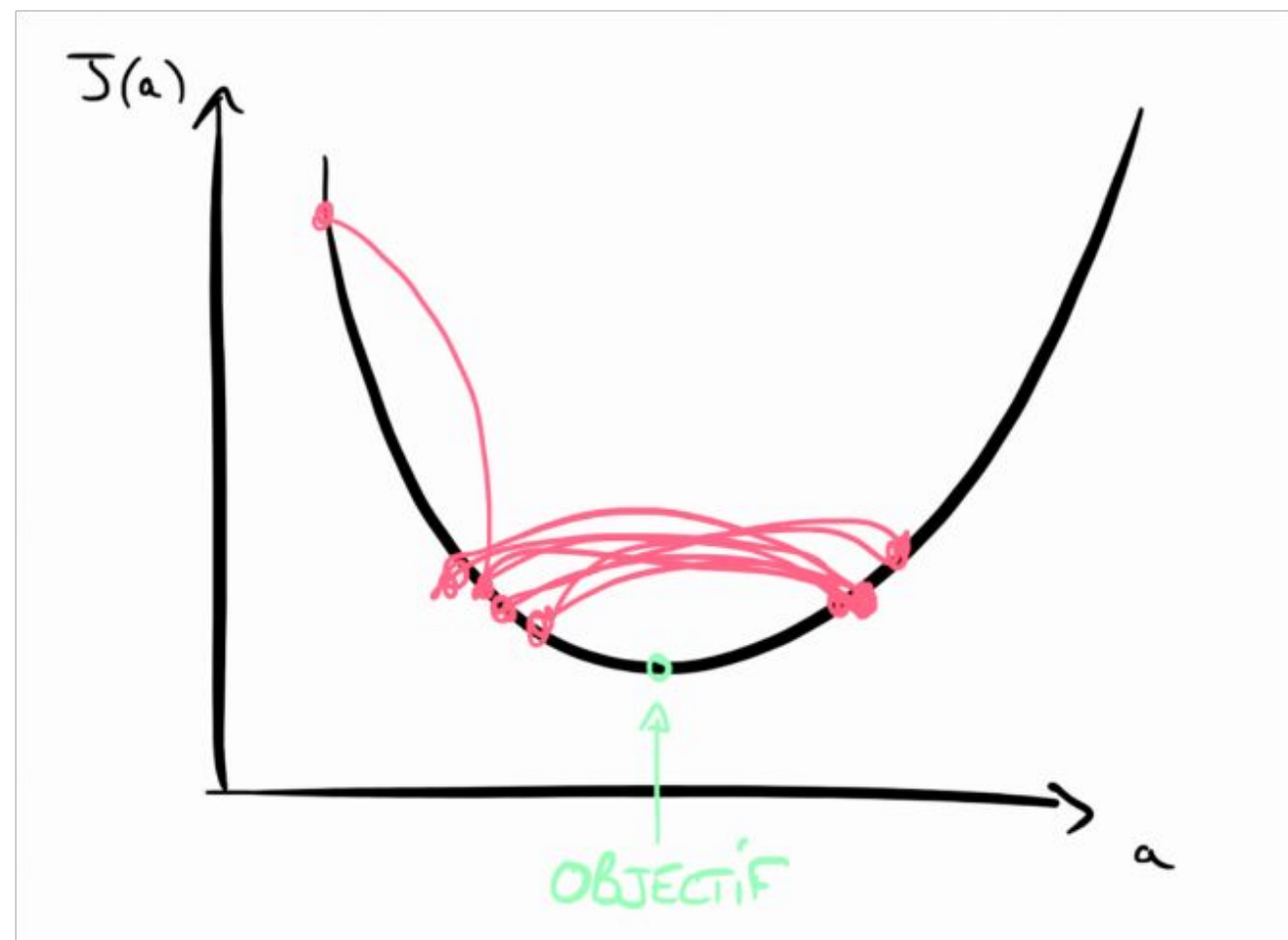
α too small:



→ Very slow convergence

Choice of the learning rate

α too big:



→ The loss function may not decrease at each iteration. The algorithm may then fail to converge.

Choice of the learning rate

→ It is a hyperparameter that must be set empirically through trials

- To choose the learning rate, try (multiplying by 3 at each step): ..., 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, ...
- There are strategies to gradually change the learning rate during gradient descent (to be discussed later)



Logistic Regression

Some materials are borrowed from Andrew Ng's course on Machine Learning

Regression vs Classification

Regression: The target variable to predict is continuous (quantitative variable).

Classification: The target variable to predict is discrete (qualitative variable).

→ Objective: Predict a small number of discrete values.

Multi-class problems: Dog / cat / horse / cow / ...

Binary problems: Malignant tumor / benign tumor
Good client / bad client

→ Focus here on **binary classification with logistic regression**

Binary classification



An achievable solution: Consider the linear prediction function f_{θ} as seen previously on regression and a threshold value (e.g. 0.5)

If $f_{\theta}(x) \geq 0.5$ then predict $y=1$

else if $f_{\theta}(x) < 0.5$ then predict $y=0$

Problem: $f_{\theta}(x)$ can take values much greater than 1 or much less than 0, while y belongs to $\{0, 1\}$.

→ **Logistic regression:** $0 \leq f_{\theta}(x) \leq 1$

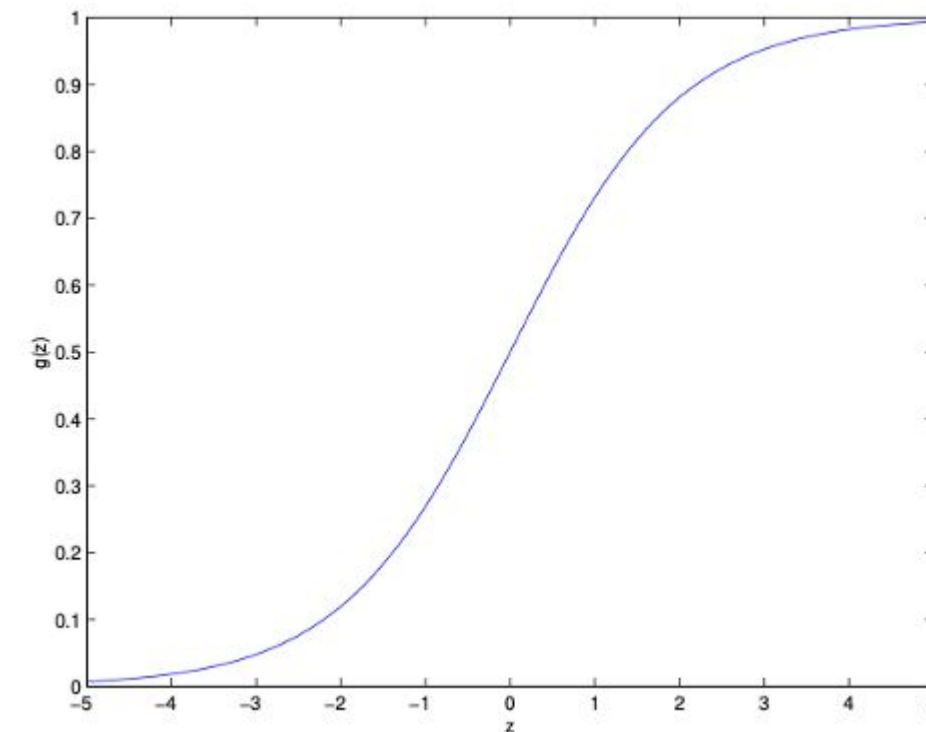
Logistic regression

Express f_{θ} using sigmoid function (logistic function):

$$f_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

where g is the sigmoid function:

$$g(z) = \frac{1}{1 + e^{-z}}$$



Logistic regression

Predict 1 if $f_{\theta}(x) \geq 0.5$ (which corresponds to $\theta^T x \geq 0$)

Predict 0 if $f_{\theta}(x) < 0.5$ (which corresponds to $\theta^T x < 0$)

So, we consider:

$$P(y = 1/x; \theta) = f_{\theta}(x)$$

$$P(y = 0/x; \theta) = 1 - f_{\theta}(x)$$

which can be summed up as: $p(y/x; \theta) = (f_{\theta}(x))^y (1 - f_{\theta}(x))^{1-y}$

Evaluation of the success by the **likelihood**:

measures of how well the model explains the observed data

it is the probability of the observed data given the model parameters

$$L(\theta) = p(\vec{y}/X; \theta)$$

Which can be expressed as (independence of the training examples):

$$L(\theta) = \prod_{i=1}^N p(y^{(i)} / x^{(i)}; \theta)$$

Or finally:

$$L(\theta) = \prod_{i=1}^N (f_{\theta}(x^{(i)}))^{y^{(i)}} (1 - f_{\theta}(x^{(i)}))^{1-y^{(i)}}$$

Log-likelihood

We use log to simplify:

$$l(\theta) = \log L(\theta)$$

And we finally obtain:

$$l(\theta) = \sum_{i=1}^N y^{(i)} \log(f_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - f_{\theta}(x^{(i)}))$$

Computation of the gradient

Remember:

$$f_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

With:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Derivative of g :

$$\begin{aligned} g'(z) &= \frac{d}{dz} \frac{1}{1 + e^{-z}} \\ &= \frac{1}{(1 + e^{-z})^2} (e^{-z}) \\ &= \frac{1}{(1 + e^{-z})} \cdot \left(1 - \frac{1}{(1 + e^{-z})} \right) \\ &= g(z)(1 - g(z)). \end{aligned}$$

Computation of the gradient

As the likelihood is:

$$l(\theta) = \sum_{i=1}^N y^{(i)} \log(f_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - f_{\theta}(x^{(i)}))$$

For one example (x, y) , we have:

$$\begin{aligned} \frac{\partial}{\partial \theta_j} \ell(\theta) &= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\ &= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) g(\theta^T x)(1 - g(\theta^T x)) \frac{\partial}{\partial \theta_j} \theta^T x \\ &= (y(1 - g(\theta^T x)) - (1 - y)g(\theta^T x)) x_j \end{aligned}$$

So the gradient of the likelihood is given by:

$$\frac{\partial}{\partial \theta_j} l(\theta) = \sum_{i=1}^N (y^{(i)} - f_{\theta}(x^{(i)})) x_j^{(i)}$$

Gradient descent

Loss function to minimize:

$$J(\theta) = -l(\theta)$$

So the algorithm for the gradient descent is:

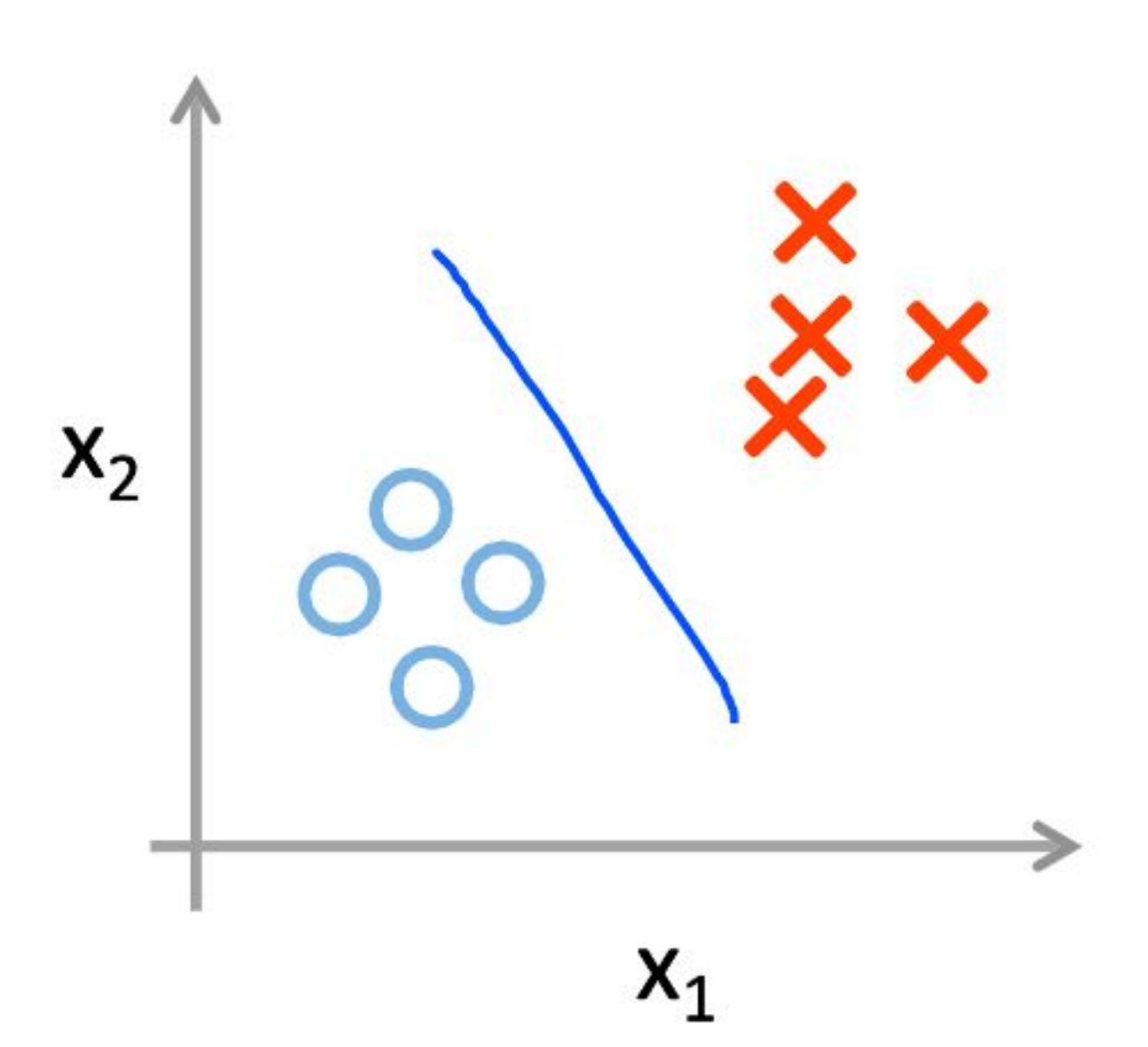
Iteratively repeat until convergence (simultaneously for each j)

$$\theta_j = \theta_j - \alpha \frac{1}{N} \sum_{i=1}^N (f_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

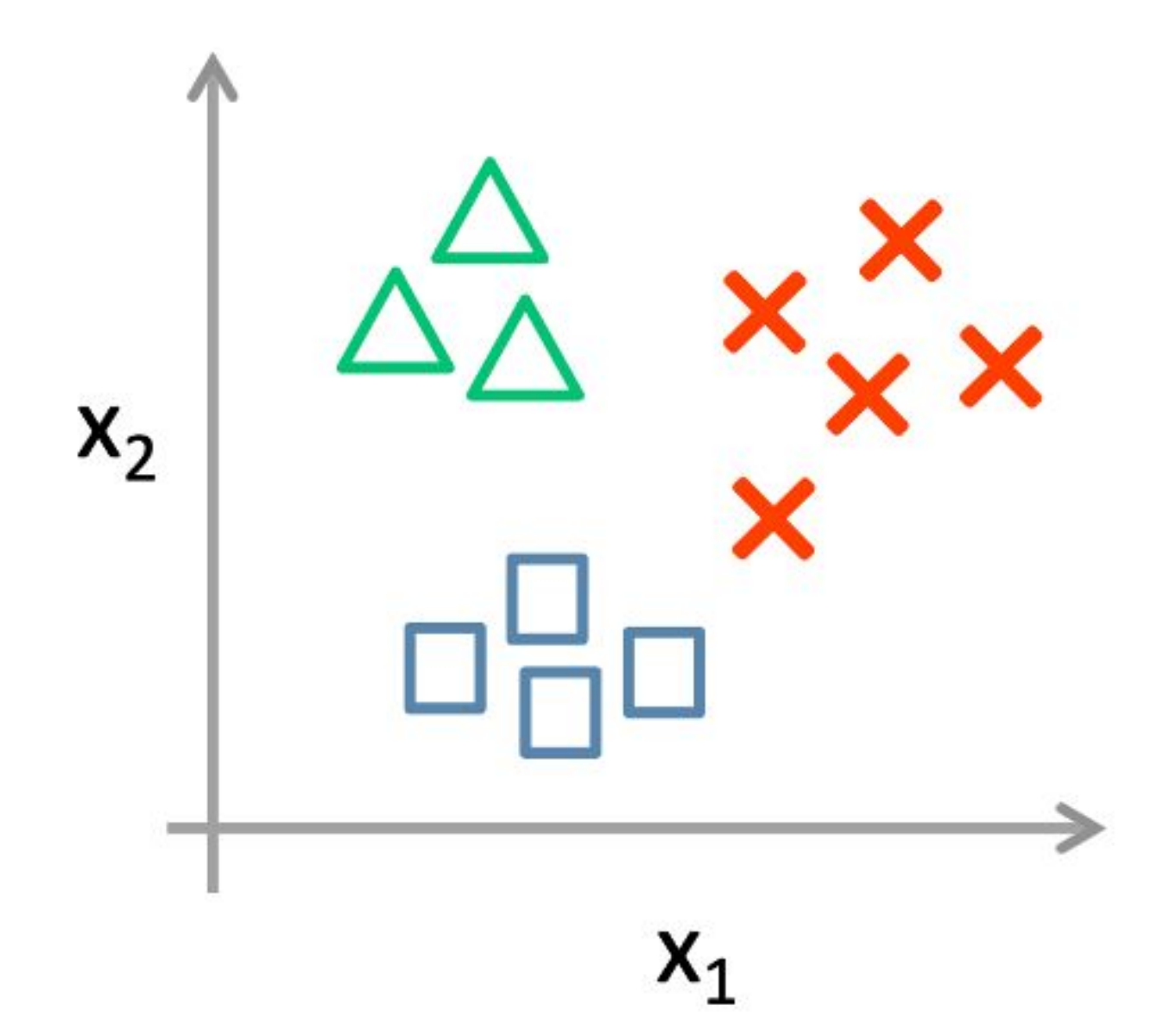
→ same expression as the one of linear regression, but with different f_{θ} (sigmoid function)

Classification with multiple classes

Binary classification



Multiclass classification



Multiclass classification

→ One against all

- Train a logistic regression classifier $f_{\theta}^{(k)}(x)$ for each class k to predict the probability of $y=k$
- For a new example x , choose the class k that maximizes $f_{\theta}^{(k)}(x)$

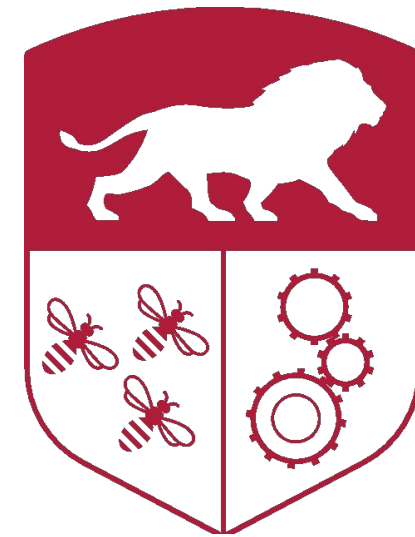
Some useful references

A. Ng, Machine Learning Course, Stanford University.

R.Duda, P. Hart, “Pattern Classification and Scene Analysis”, John Wiley & Sons, 2001.

C. M. Bishop, “Pattern recognition and machine learning”, Springer, 2006.

Numpy Library, <https://numpy.org/>



**CENTRALE
LYON**

36, avenue Guy de Collongue 69130 Écully
www.ec-lyon.fr | [@centralelyon](https://twitter.com/centralelyon)