Linear regression

CE-477: Machine Learning - CS-828: Theory of Machine Learning
Sharif University of Technology
Fall 2024

Fatemeh Seyyedsalehi

References of the lecture are mentioned in the last slide

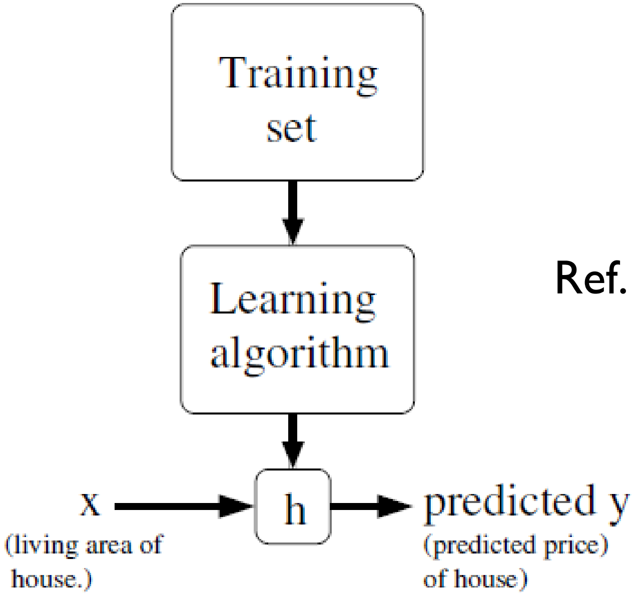
A supervised problem

- Predict the residential electrical usage as a function of living area and house members.
- Input feature $\boldsymbol{x}^{(i)} \in \mathcal{X}$
- Output or target $y^{(i)} \in \mathcal{Y}$
- Training set $D = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$

Living area (m^2)	#House members	Residential Electrical usage(kwh)
40	2	4
70	2	5
70	4	7
200	3	8
150	4	7
.	.	.
.	.	.

A supervised problem

- Our goal is to learn a function $h : \mathcal{X} \to \mathcal{Y}$
- $h(x)$ should be a good predictor for the corresponding y
- h is called a **hypothesis**



Ref. [2]

Linear Regression

- Regression problem: The target is continuous

- We begin by the class of linear functions as a hypothesis
  - Easy to extend to generalized linear and so cover more complex regression functions

Linear Regression

- Coming back to our problem
- $\boldsymbol{x}$ is a two dimensional vector in $\mathbb{R}^2$
- We first decide how to represent the hypothesis h
  - A function family

Living area (m^2)	#House members	Residential Electrical usage(kwh)
40	2	4
70	2	5
70	4	7
200	3	8
150	4	7
.	.	.
.	.	.

Linear Regression: the hypothesis space

- $h_w(\boldsymbol{x})$ as a linear function of $\boldsymbol{x}$

$$h_w(\boldsymbol{x}) = w_0 + w_1\, x_1 + w_2 x_2$$

- Where w_is are called parameters or weights of the model which parameterize the space of linear functions mapping from $\mathcal{X}$ to $\mathcal{Y}$.

Linear Regression: the hypothesis space

- To have a better notation we consider $x_0 = 1$ (the intercept term),

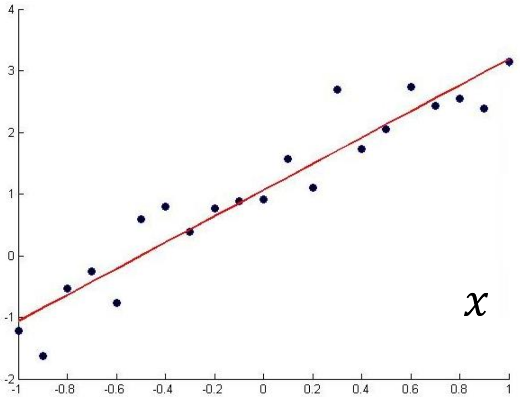
$$h_w(\boldsymbol{x}) = \sum_{i=0}^{d} w_i x_i$$

- How to learn parameters w_is, given a training set,
  - Make $h(\boldsymbol{x})$ close to y

Linear Regression: the hypothesis space

- Univariate

$$h: \mathbb{R} \to \mathbb{R} \qquad h_w(\boldsymbol{x}) = w_0 + w_1 x$$



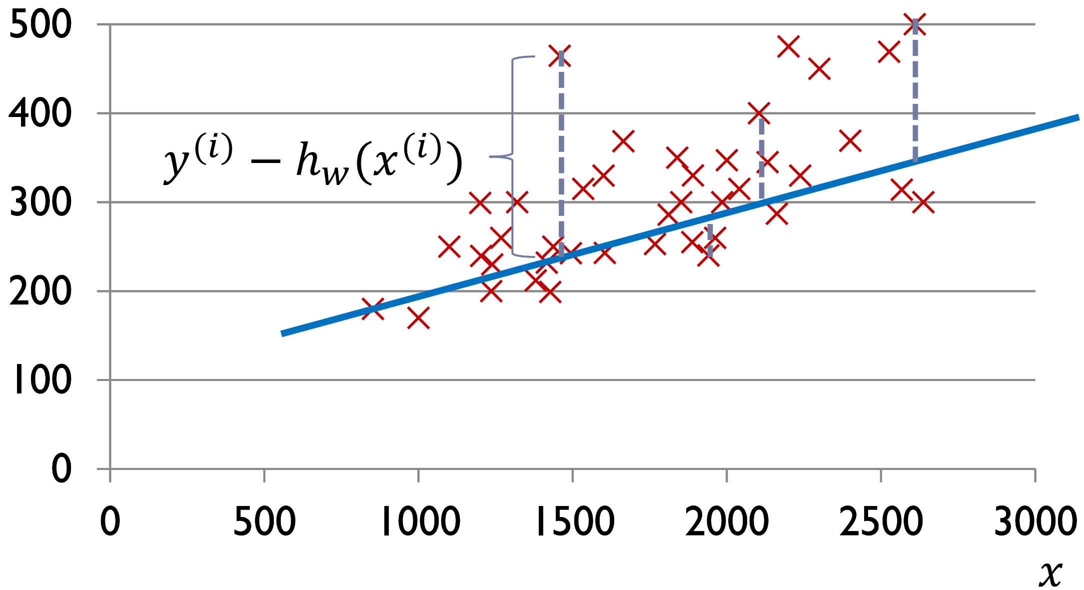
Ref. [1]

- Multivariate

$$h: \mathbb{R}^d \to \mathbb{R} \qquad h_w(\boldsymbol{x}) = w_0 + w_1 x_1 + \ldots w_d x_d$$

$\boldsymbol{w} = [w_0, w_1, \ldots, w_d]^T$ are parameters we need to set.

How to measure the error

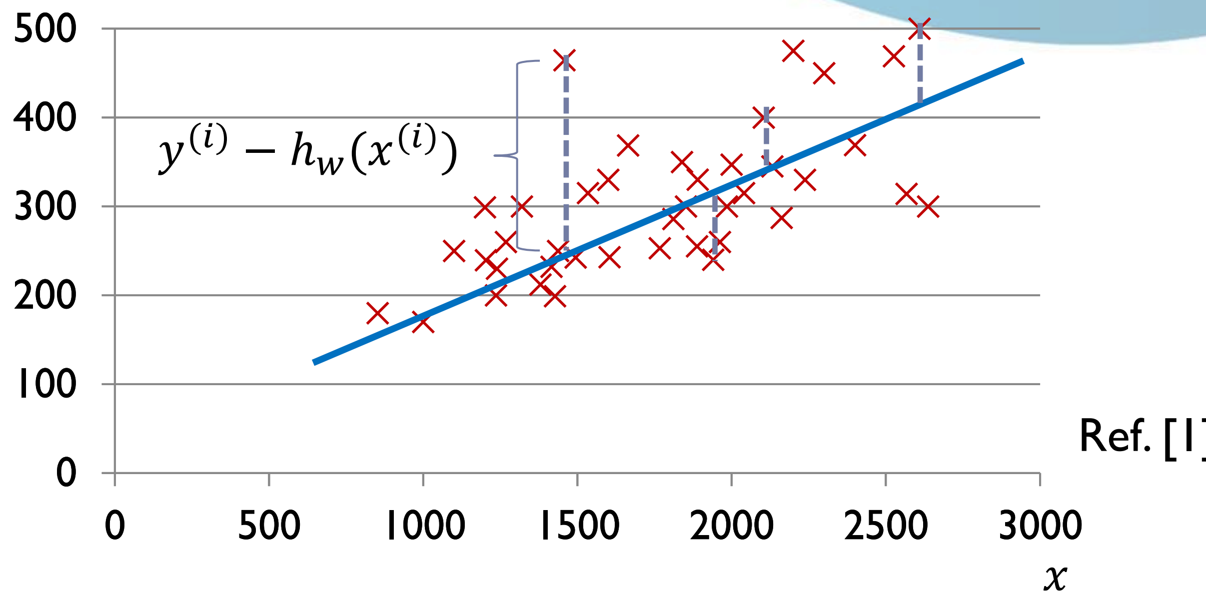


Ref. [1]

- A loss function between the ground truth and the estimated output

$$\text{Loss} = (y - h_w(x))^2$$

How to measure the error



Cost function (sum of squared error(SSE)):

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} \left(y^{(i)} - h_w(x^{(i)})\right)^2$$
$$= \sum_{i=1}^{n} \left(y^{(i)} - w_0 - w_1 x^{(i)}\right)^2$$

Linear Regression: the learning algorithm

- Choose w so as to minimize the $J(w)$

$$J(\boldsymbol{w}) = \sum_{i=1}^{n}\left(y^{(i)} - h_w(\boldsymbol{x}^{(i)})\right)^2$$

- The learning algorithm: optimization of the cost function
  - Explicitly taking the cost function derivative with respect to the w_is, and setting them to zero.

- Parameters of the best hypothesis for the training set:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, J(\boldsymbol{w})$$

Cost function optimization: univariate

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} \left(y^{(i)} - w_0 - w_1 x^{(i)}\right)^2$$

- Necessary conditions for the "optimal" parameter values:

$$\frac{\partial J(\boldsymbol{w})}{\partial w_0} = 0$$

$$\frac{\partial J(\boldsymbol{w})}{\partial w_1} = 0$$

Optimality conditions: univariate

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} \left(y^{(i)} - w_0 - w_1 x^{(i)}\right)^2$$

$$\frac{\partial J(\boldsymbol{w})}{\partial w_1} = \sum_{i=1}^{n} 2\left(y^{(i)} - w_0 - w_1 x^{(i)}\right)\left(-x^{(i)}\right) = 0$$

$$\frac{\partial J(\boldsymbol{w})}{\partial w_0} = \sum_{i=1}^{n} 2\left(y^{(i)} - w_0 - w_1 x^{(i)}\right)(-1) = 0$$

- A systems of 2 linear equations

Cost function optimization: multivariate

$$J(\boldsymbol{w}) = \sum_{i=1}^{n}\left(y^{(i)} - h_w(\boldsymbol{x}^{(i)})\right)^2 = \sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{w}^T\boldsymbol{x}^{(i)}\right)^2$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_d^{(n)} \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Cost function optimization: multivariate

- Explicitly taking the cost function derivative with respect to the $\boldsymbol{w}$s, and setting them to zero.

$$J(\boldsymbol{w}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = -2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \boldsymbol{0} \Rightarrow \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$$

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

- Is $(\boldsymbol{X}^T\boldsymbol{X})$ invertible?

Cost function optimization

- Another approach,
  - Start from an initial guess and iteratively change $\boldsymbol{w}$ to minimize $J(\boldsymbol{w})$.
    - The gradient descent algorithm

- Steps:
  - Start from $\boldsymbol{w}^0$
  - Repeat
    - Update $\boldsymbol{w}^t$ to $\boldsymbol{w}^{t+1}$ in order to reduce J
    - $t \leftarrow t + 1$
  - until we hopefully end up at a minimum

Review: Gradient descent

- In each step, takes steps proportional to the negative of the gradient vector of the function at the current point $\boldsymbol{w}^t$:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta \nabla J(\boldsymbol{w}^t)$$

  - $J(\boldsymbol{w})$ decreases fastest if one goes from $\boldsymbol{w}^t$ in the direction of $-\nabla J(\boldsymbol{w}^t)$
  - Assumption: $J(\boldsymbol{w})$ is defined and differentiable in a neighborhood of a point $\boldsymbol{w}^t$

Gradient ascent takes steps proportional to (the positive of) the gradient to find a local maximum of the function

- Continue to find

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}} J(\boldsymbol{w})$$

Review: Gradient descent

- Minimize $J(\boldsymbol{w})$

Step size
(Learning rate parameter)

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \eta \nabla_{\boldsymbol{w}} J(\boldsymbol{w}^{t})$$

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial J(\boldsymbol{w})}{\partial w_1} \\ \vdots \\ \frac{\partial J(\boldsymbol{w})}{\partial w_d} \end{bmatrix}$$

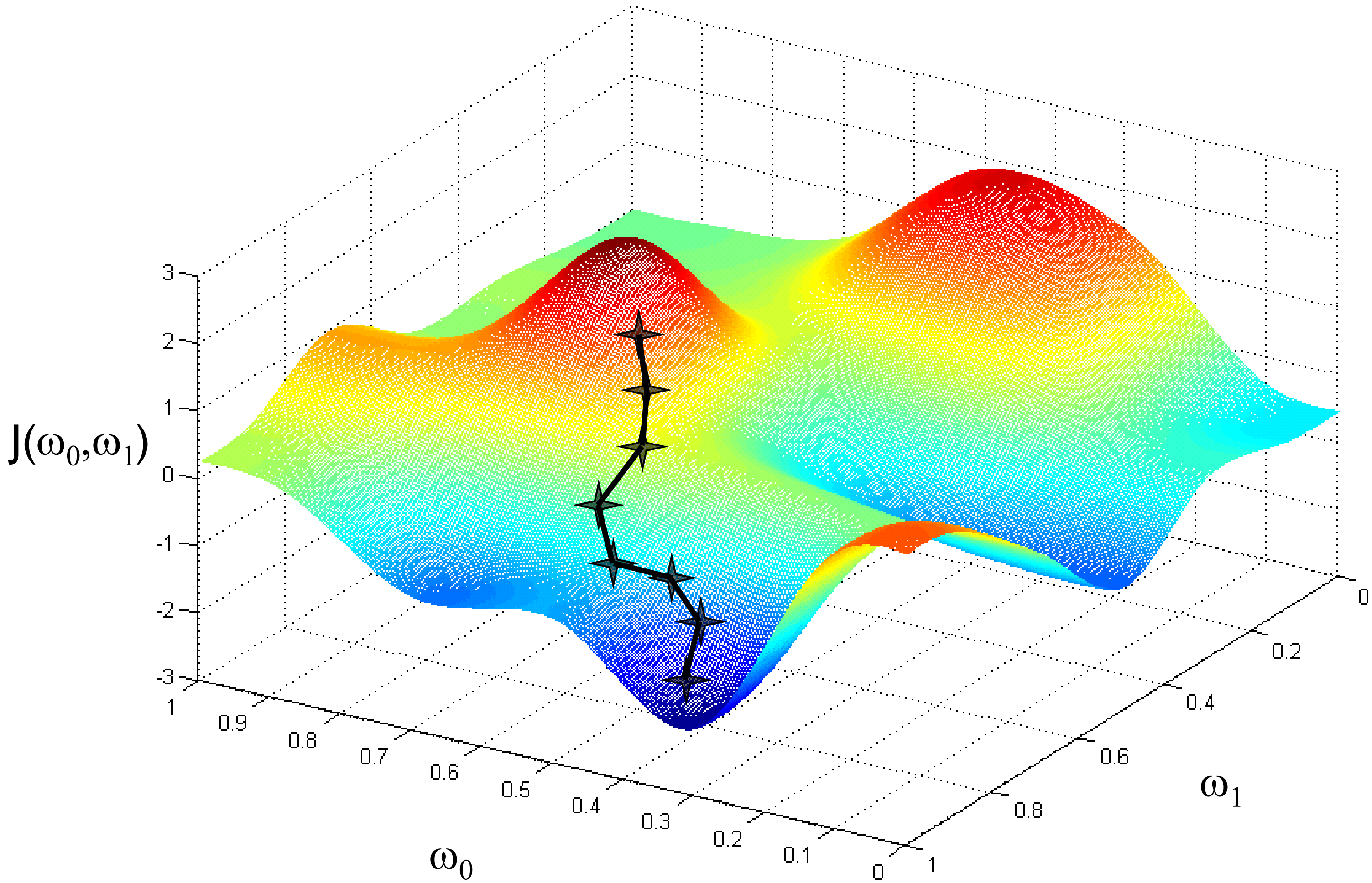
- If η is small enough, then $J(\boldsymbol{w}^{t+1}) \leq J(\boldsymbol{w}^{t})$.
- η can be allowed to change at every iteration as η_t.

Review:
Gradient descent disadvantages

- Local minima problem

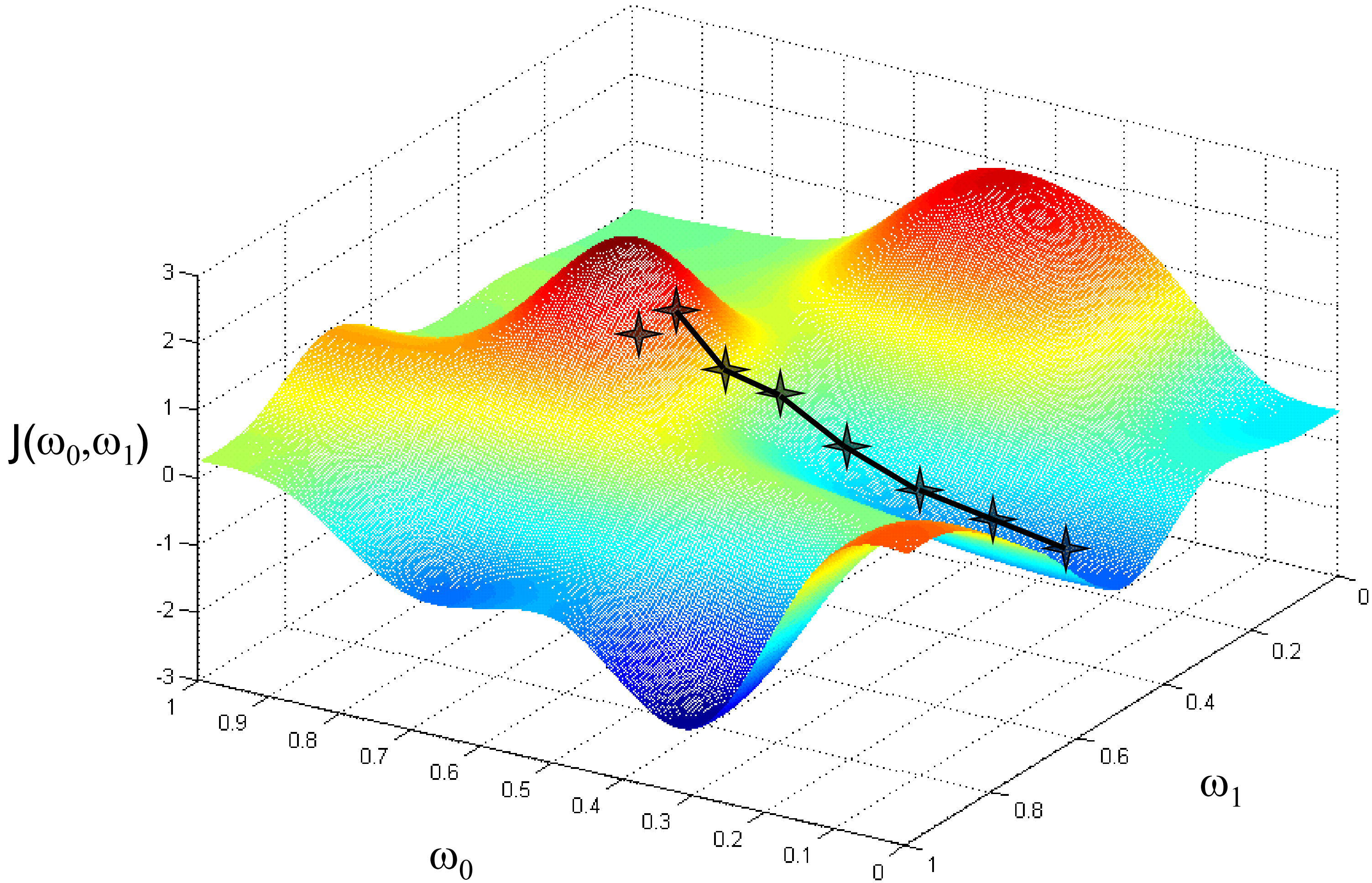
- However, when J is convex, all local minima are also global minima $\Rightarrow$ gradient descent can converge to the global solution.

Review: Problem of gradient descent with non-convex cost functions



Ref. [2]

Review: Problem of gradient descent with non-convex cost functions



Ref. [2]

Cost function optimization

- Minimize $J(\boldsymbol{w})$

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta \nabla_{\boldsymbol{w}} J(\boldsymbol{w}^t)$$

- $J(\boldsymbol{w})$: Sum of squares error

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} \left(y^{(i)} - h_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) \right)^2$$

- Weight update rule for $h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \eta \sum_{i=1}^{n} \left(y^{(i)} - {\boldsymbol{w}^t}^T \boldsymbol{x}^{(i)} \right) \boldsymbol{x}^{(i)}$$

Cost function optimization

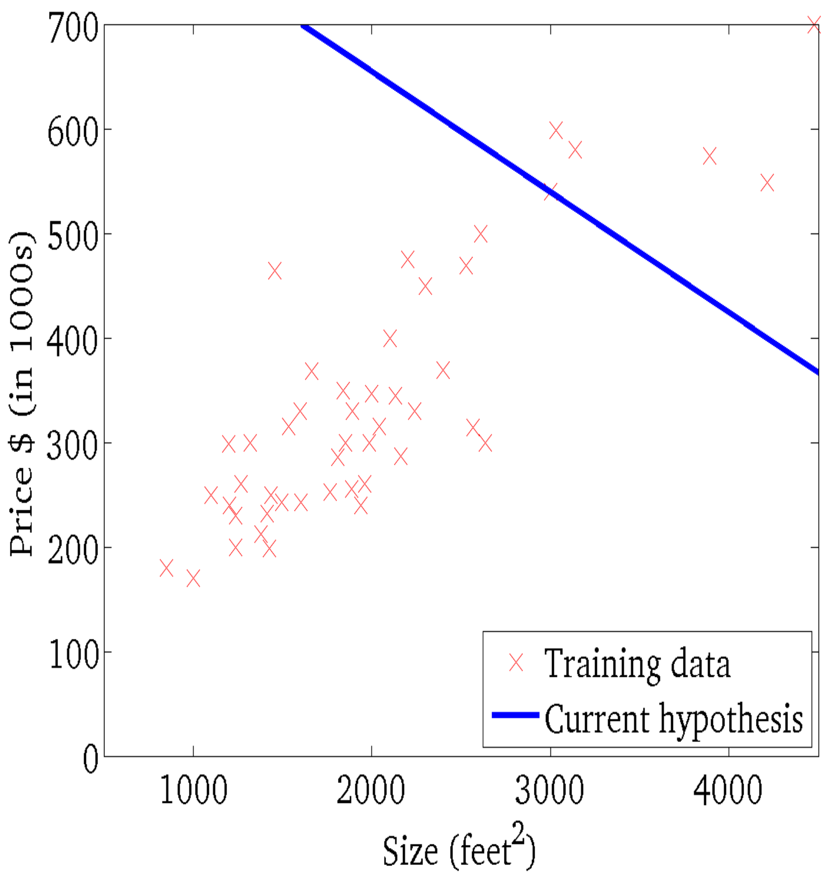
- Weight update rule: $h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} + \eta \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)} \right) \boldsymbol{x}^{(i)}$$

Batch mode: each step considers all training data

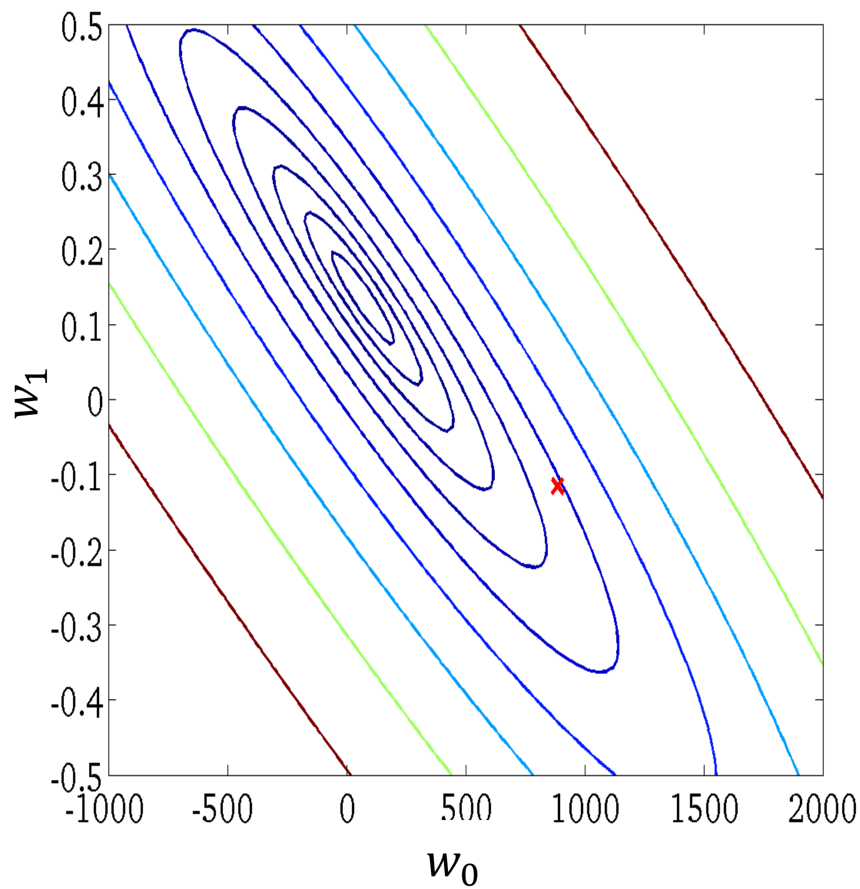
- η: too small → gradient descent can be slow.
- η: too large → gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

$h_w(x) = w_0 + w_1 x$



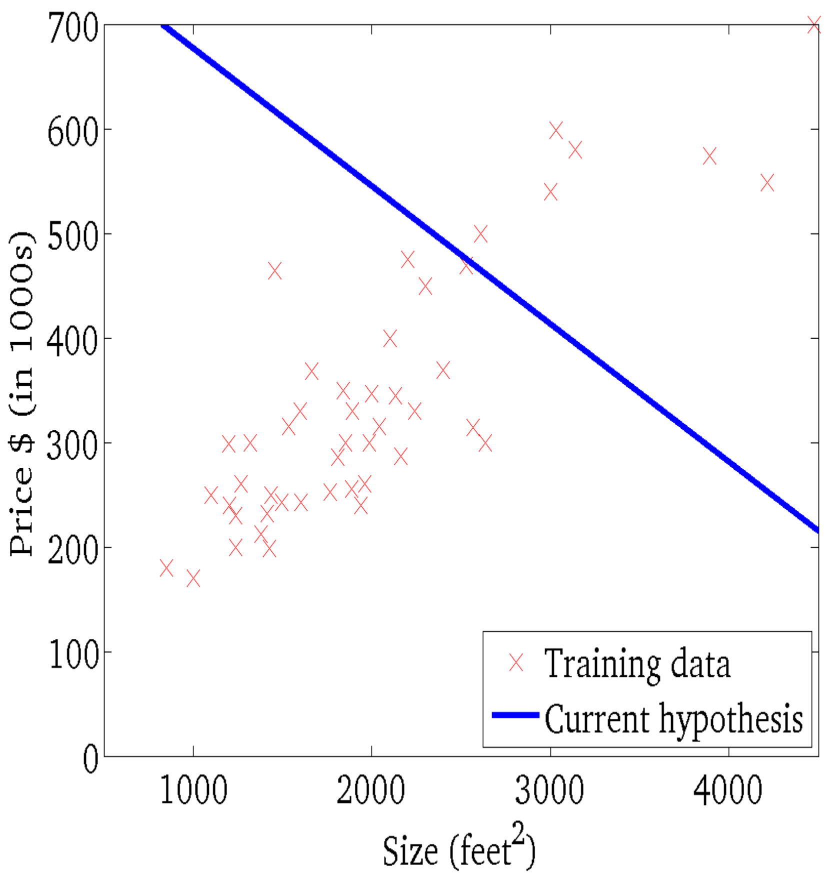
$J(w_0, w_1)$

(function of the parameters w_0,w_1)



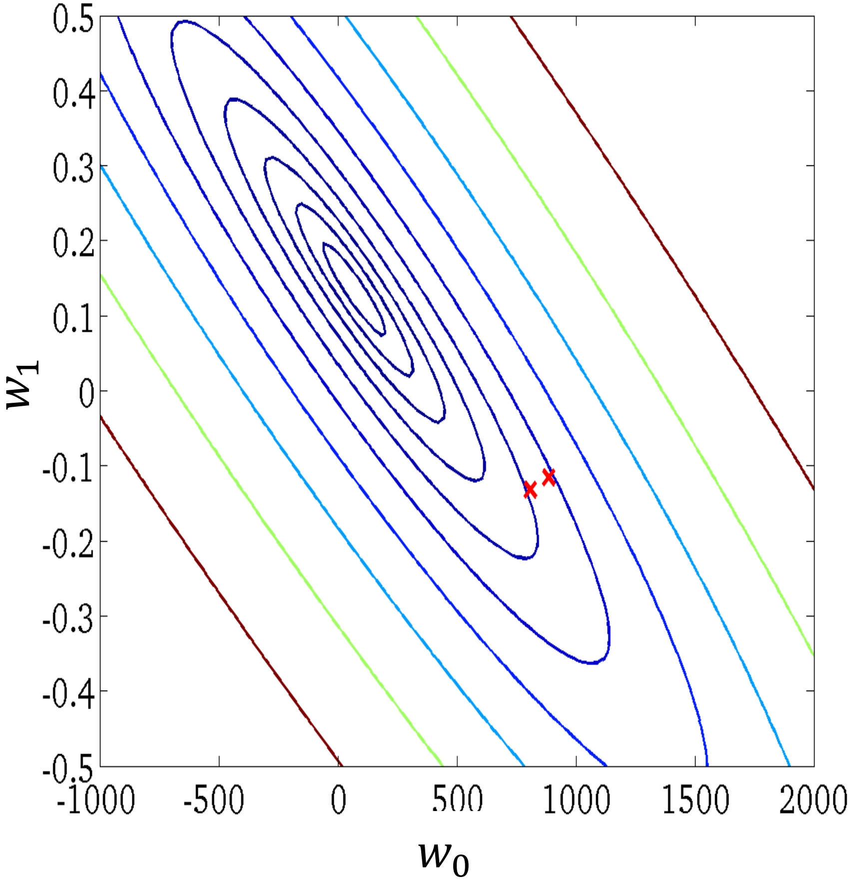
Ref. [2]

$$h_w(x) = w_0 + w_1 x$$



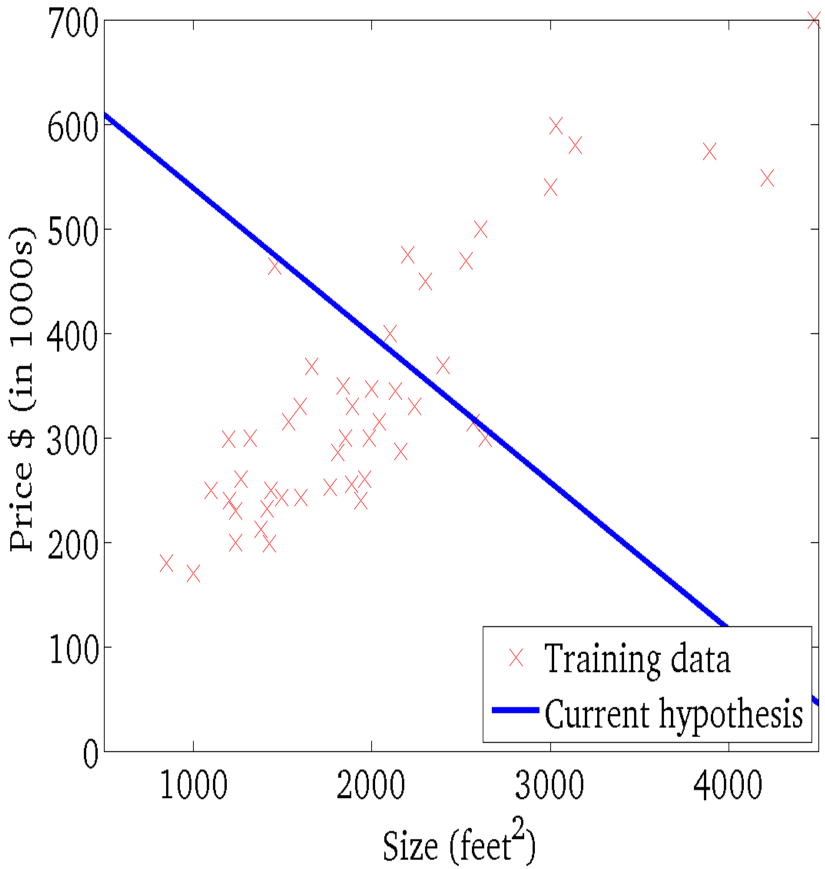
$$J(w_0, w_1)$$

(function of the parameters w_0,w_1)



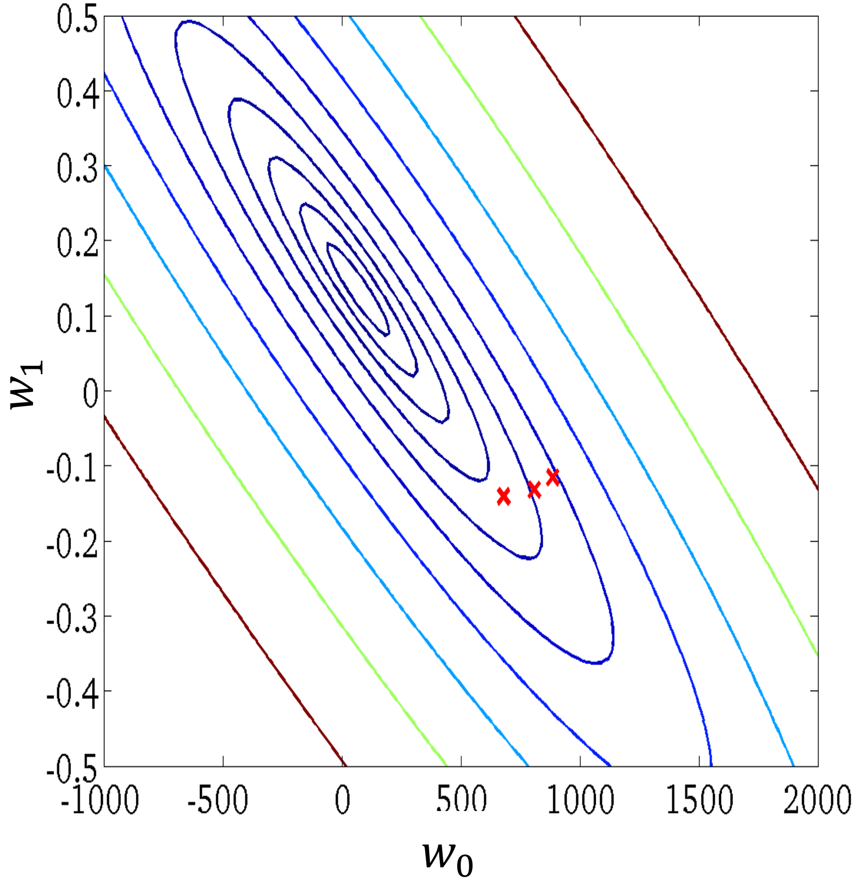
Ref. [2]

$h_w(x) = w_0 + w_1 x$



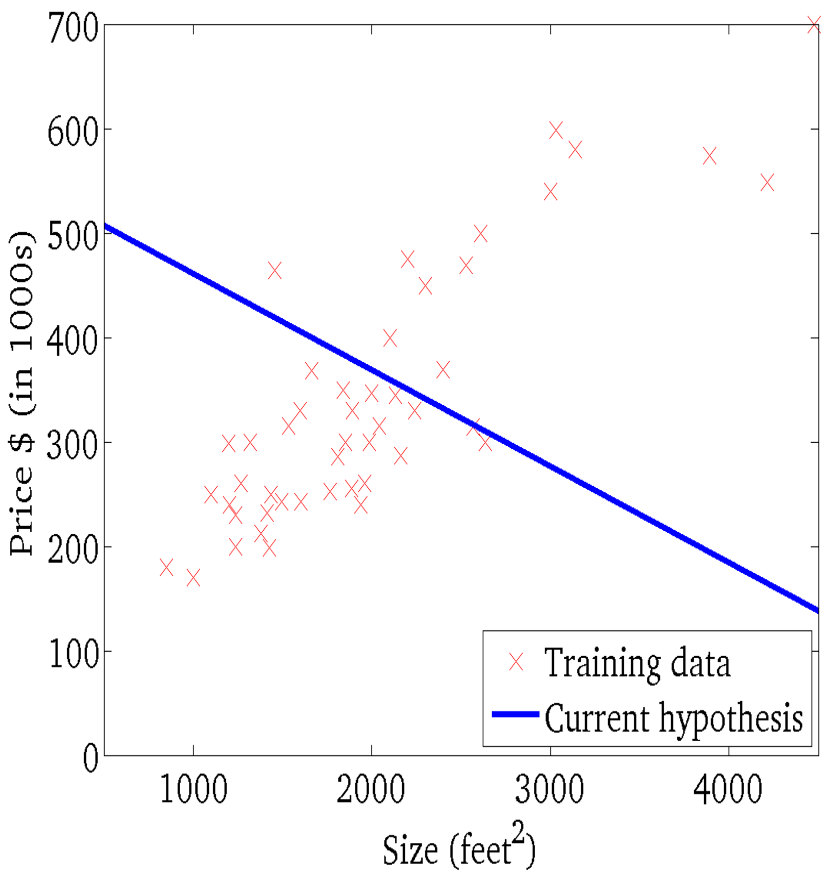
$J(w_0, w_1)$

(function of the parameters w_0,w_1)



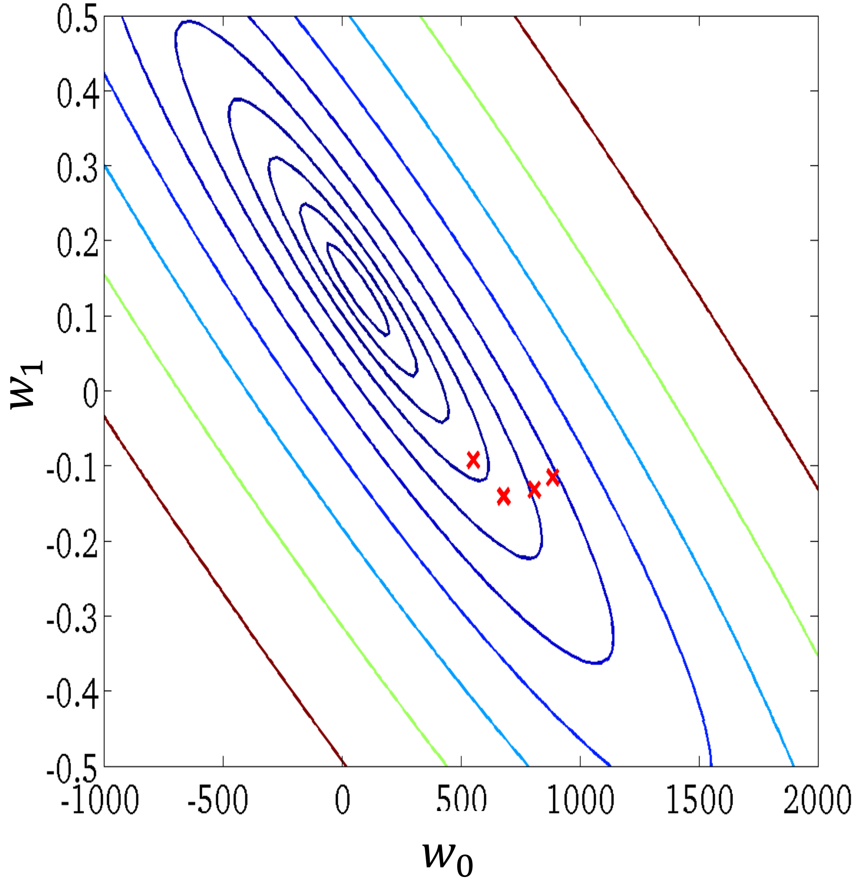
Ref. [2]

$$h_w(x) = w_0 + w_1 x$$



$J(w_0, w_1)$

(function of the parameters w_0,w_1)

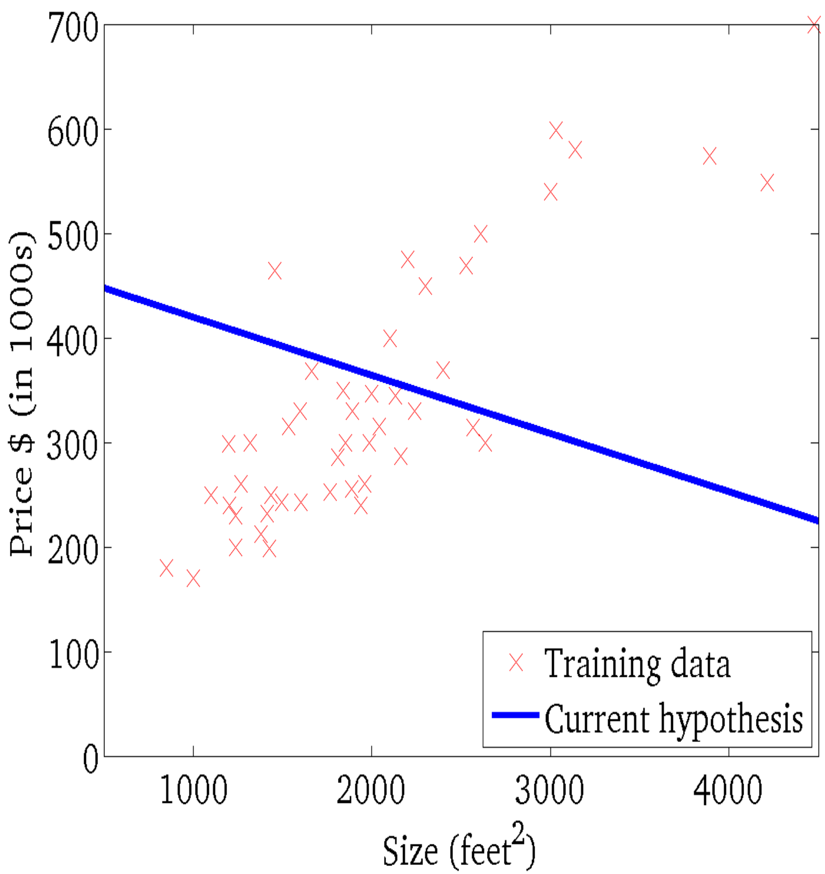


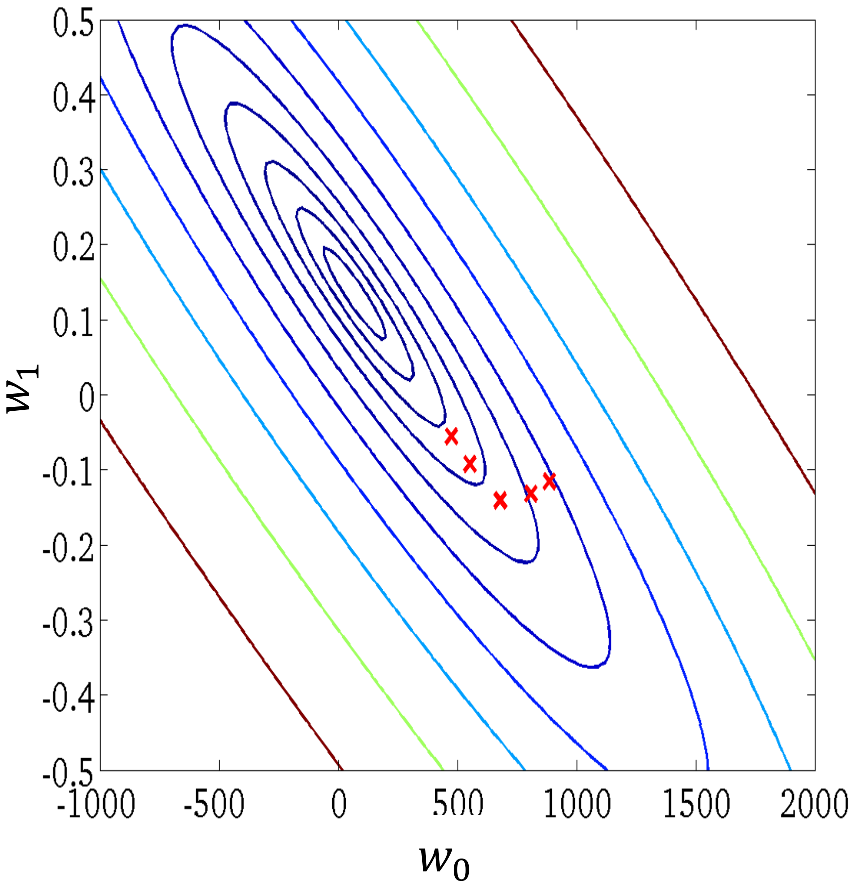
Ref. [2]

$h_w(x) = w_0 + w_1 x$

$J(w_0, w_1)$

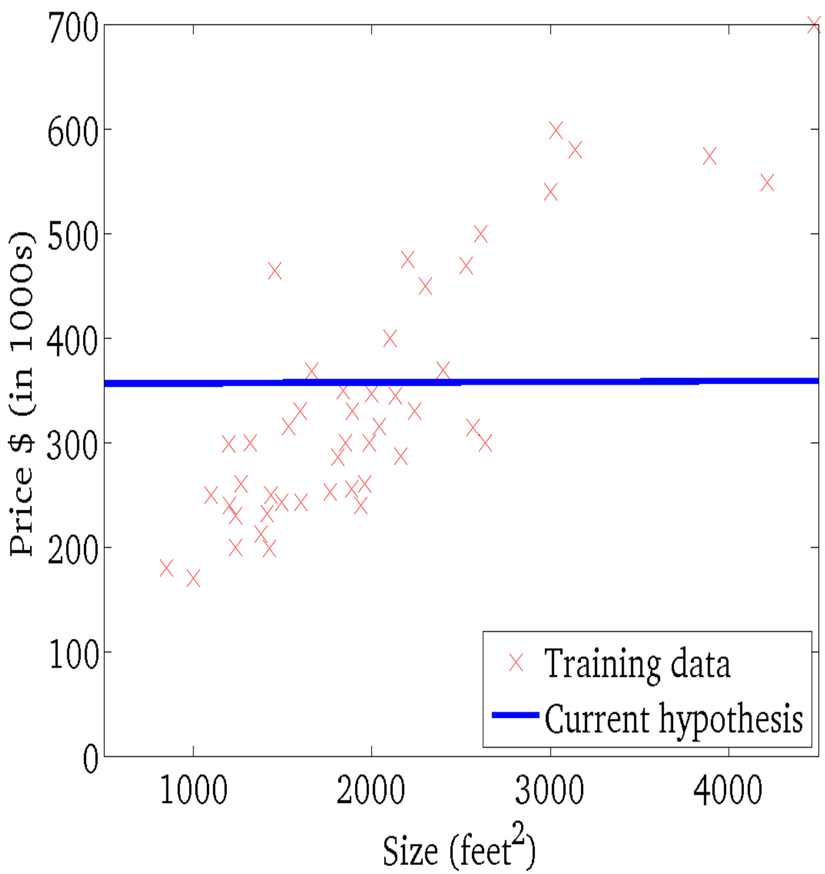
(function of the parameters w_0,w_1)





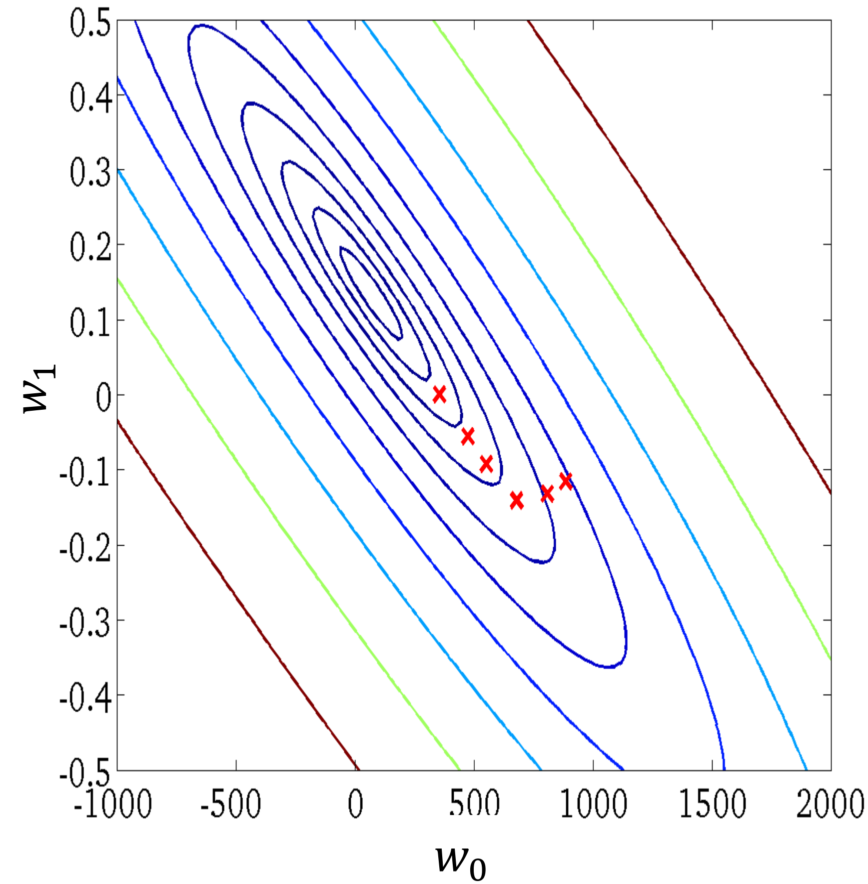
Ref. [2]

$$h_w(x) = w_0 + w_1 x$$



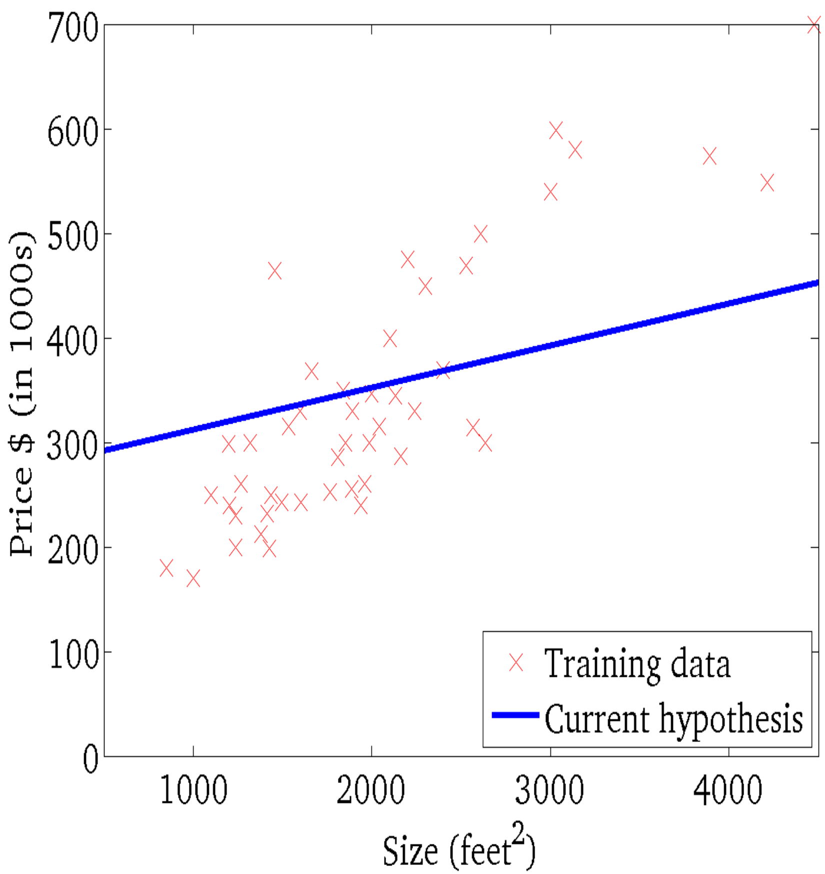
$$J(w_0, w_1)$$

(function of the parameters w_0,w_1)



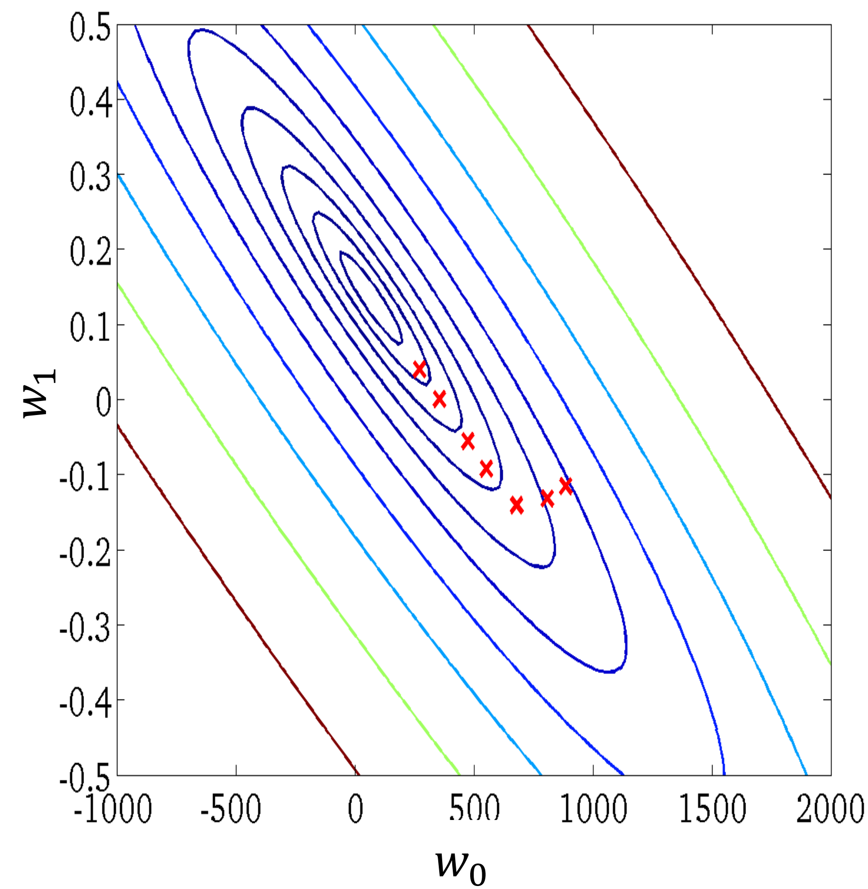
Ref. [2]

$$h_w(x) = w_0 + w_1 x$$



$$J(w_0, w_1)$$

(function of the parameters w_0,w_1)

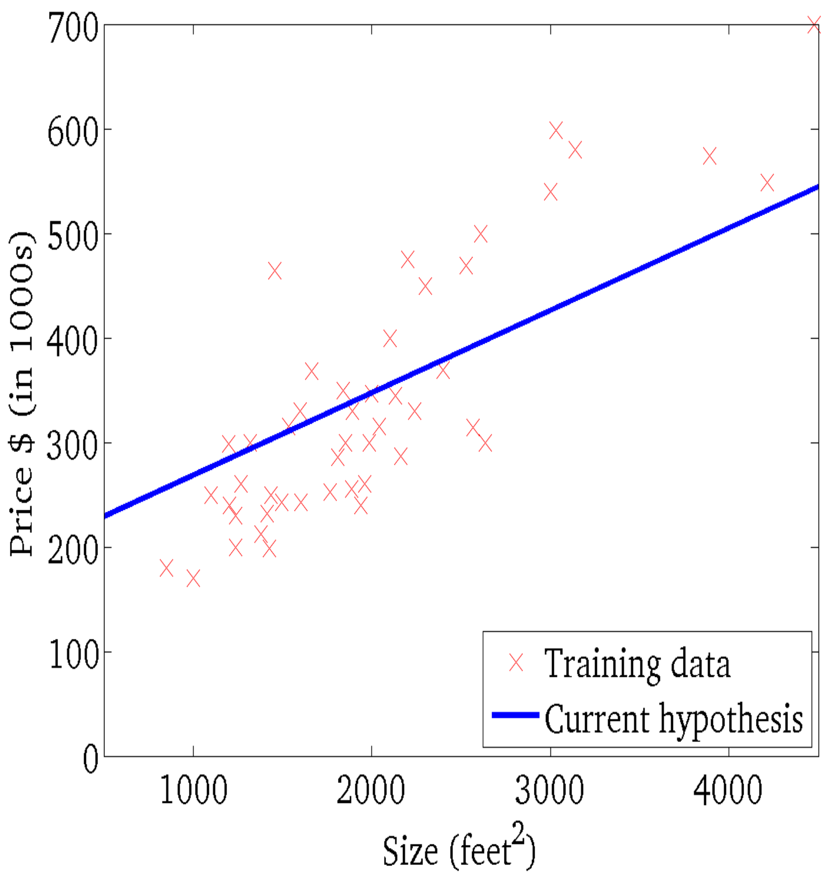


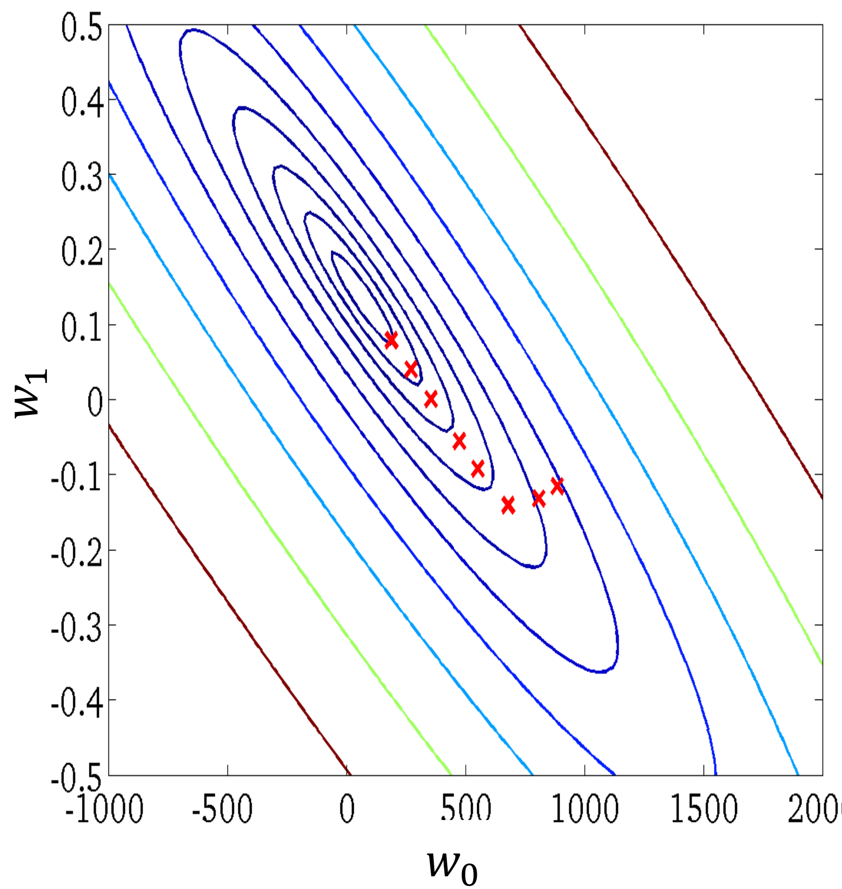
Ref. [2]

$$h_w(x) = w_0 + w_1 x$$

$J(w_0, w_1)$

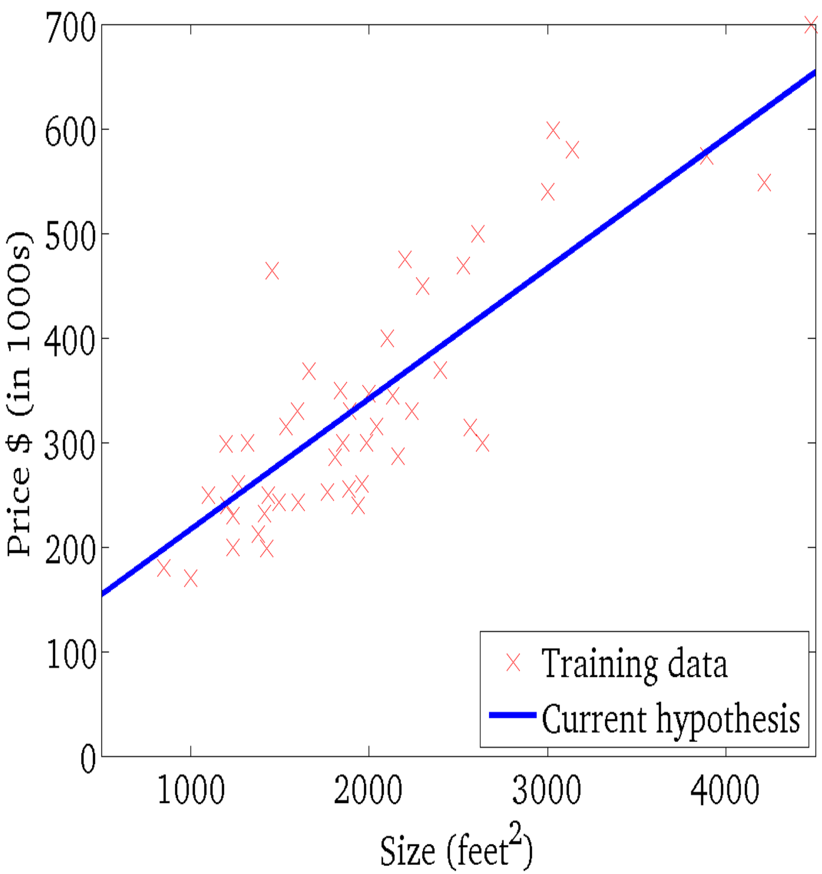
(function of the parameters w_0,w_1)





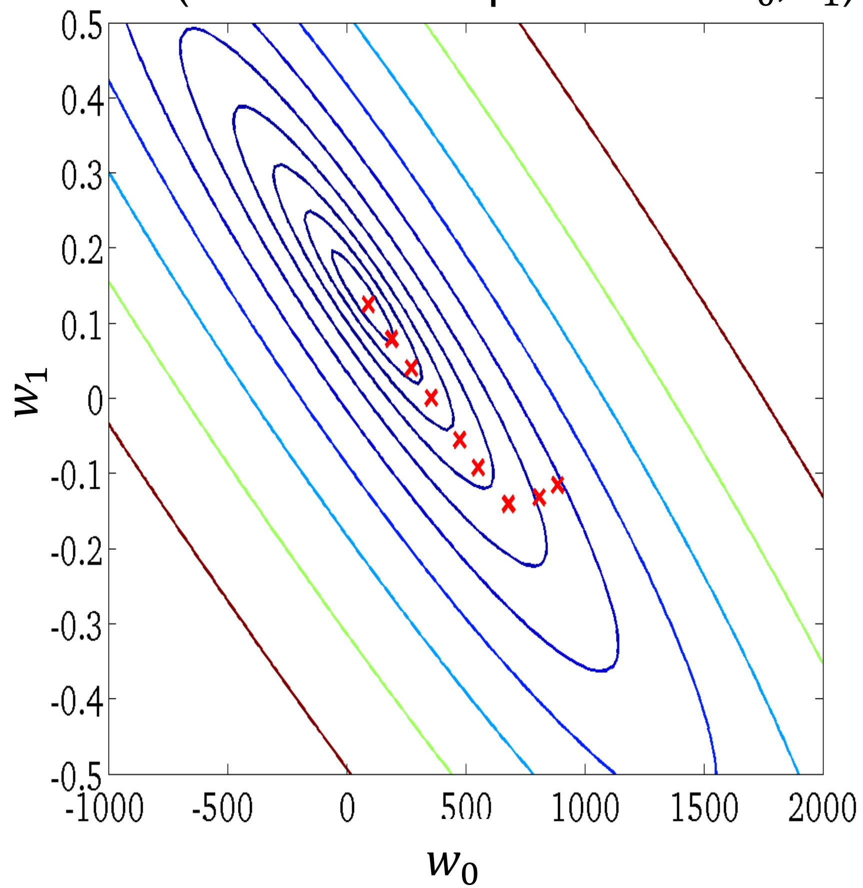
Ref. [2]

$h_w(x) = w_0 + w_1 x$



$J(w_0, w_1)$

(function of the parameters w_0,w_1)



Ref. [2]

Stochastic gradient descent

- Batch techniques process the entire training set in one iteration
  - Thus they can be computationally costly for large data sets.

- Stochastic gradient descent: when the cost function can comprise a sum over data points:

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} J^{(i)}(\boldsymbol{w})$$

Stochastic gradient descent

- Stochastic gradient descent: when the cost function can comprise a sum over data points:

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} J^{(i)}(\boldsymbol{w})$$

- Update after presentation of $(\boldsymbol{x}^{(i)}, y^{(i)})$:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \eta \nabla_{\boldsymbol{w}} J^{(i)}(\boldsymbol{w})$$

Stochastic gradient descent

- Example: Linear regression with SSE cost function

$$J^{(i)}(\boldsymbol{w}) = \left(y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)}\right)^2$$

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta \nabla_{\boldsymbol{w}} J^{(i)}(\boldsymbol{w})$$

$$\boxed{\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \eta\left(y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)}\right)\boldsymbol{x}^{(i)}}$$

Stochastic gradient descent: online learning

- Sequential learning is also appropriate for real-time applications
  - Data observations are arriving in a continuous stream
  - Predictions must be made before seeing all of the data

Stochastic gradient descent: online learning

- Often, stochastic gradient descent gets close to the minimum much faster than batch gradient descent.

- Note however that it may never converge to the minimum, and the parameters will keep oscillating around the minimum of cost function;
  - In practice most of the values near the minimum will be reasonably good approximations to the true minimum.

References

- [1]: Mahdieh Soleymani, Machine learning, Sharif university of technology
- [2]: Andrew Ng, Machine learning, Stanford