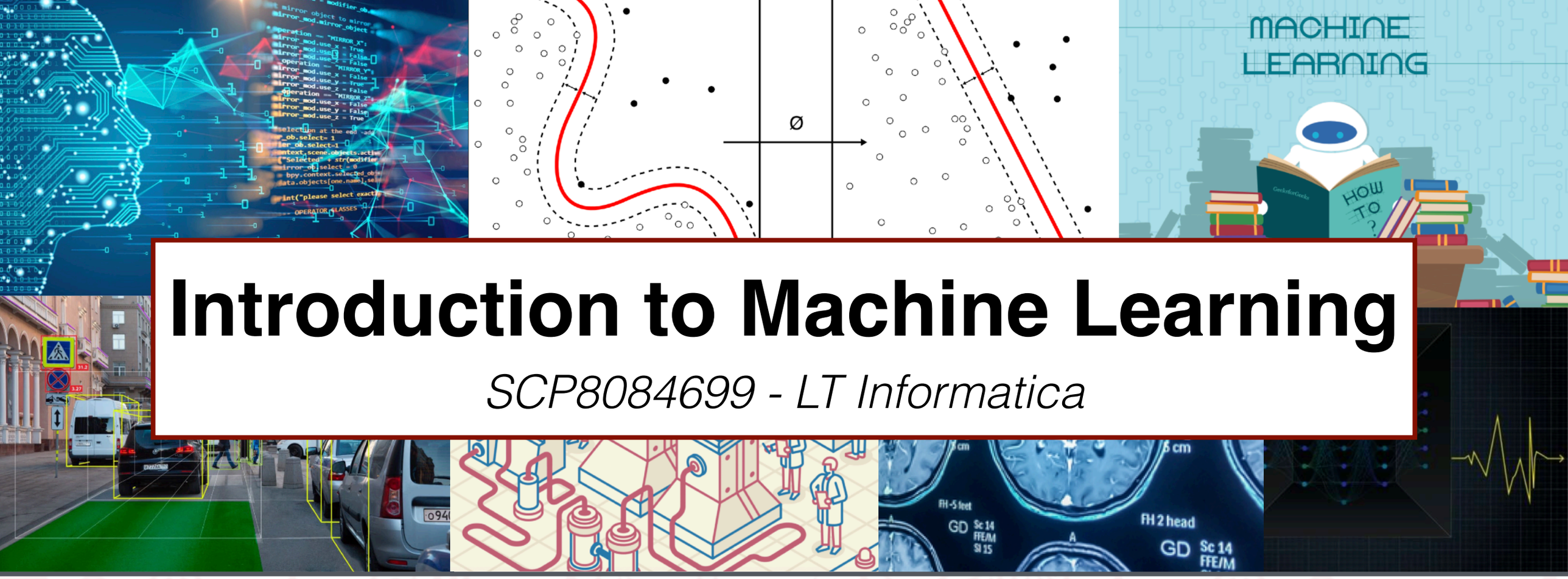


Introduction to Machine Learning

SCP8084699 - LT Informatica

Linear Regression and Classification, Regularization

Prof. Lamberto Ballan

University of Padova, A.Y. 2022/23 March 14, 2023

DIPARTIMENTO MATEMATICA
Department of Mathematics “Tullio Levi-Civita”

Linear Regression & Gradient Descent

Linear Regression Model

- Hypothesis:

$$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\, \mathrm{x}$$

- Cost Function:

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

- Objective: $\underset{\theta_0,\theta_1}{\text{minimize}}\ \mathrm{J}(\theta_0, \theta_1)$

Gradient Descent

```
repeat until convergence{
```

$$\theta_\mathrm{j} := \theta_\mathrm{j} - \eta \frac{\partial}{\partial \theta_\mathrm{j}} \mathrm{J}(\theta_0, \theta_1)$$

(for j = 0 and j = 1)

}

Linear Regression & Gradient Descent

- We need to figure out what is this derivative term…

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

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

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

Linear Regression & Gradient Descent

- We can now plug them back into our GD algorithm:

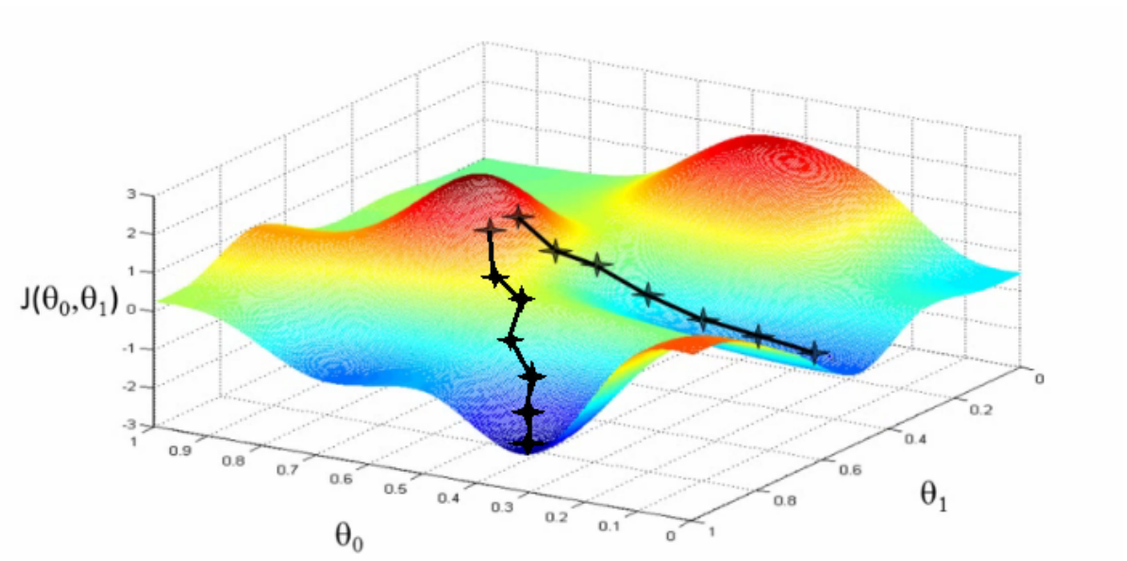
```
repeat until convergence{
```

$$\theta_0 := \theta_0 - \eta \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})$$

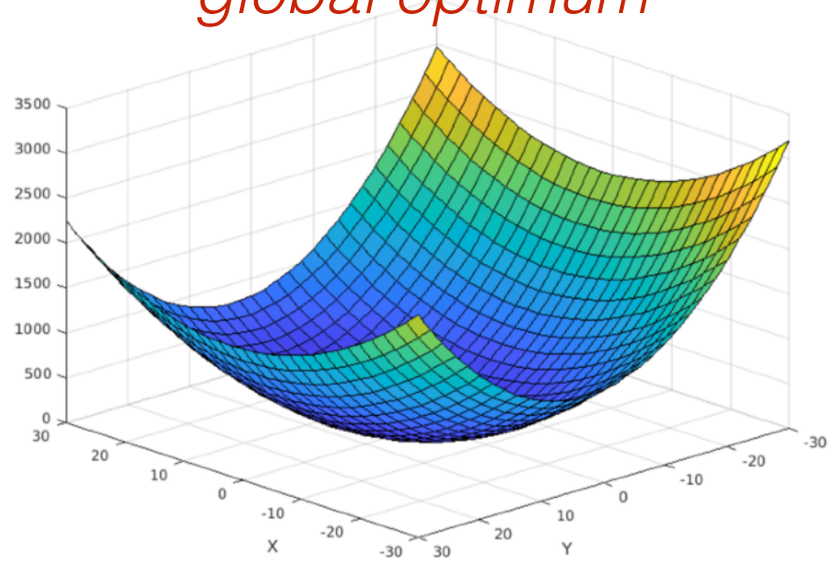
$$\theta_1 := \theta_1 - \eta \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})\, x^{(i)}$$

simultaneously update θ_0 and θ_1

```
}
```



global optimum

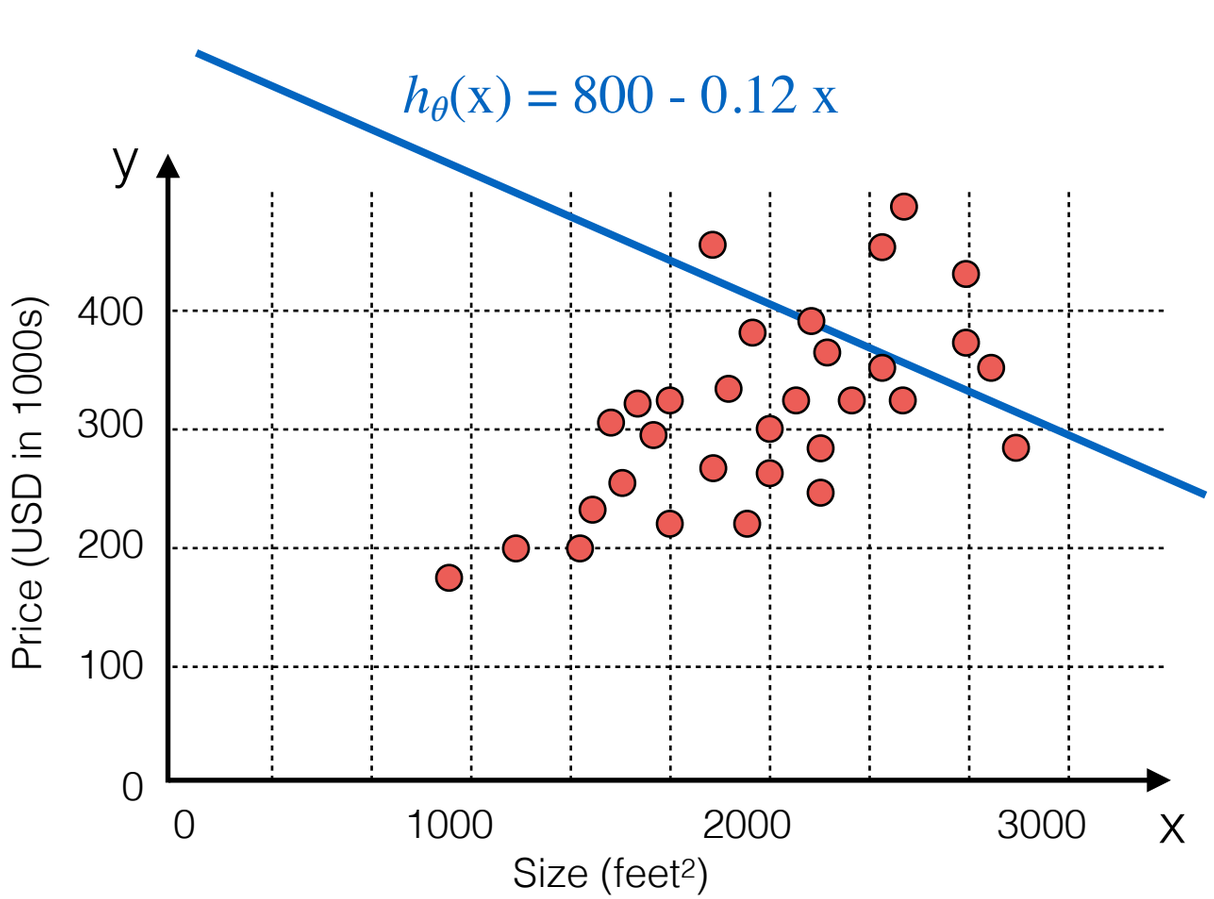


Linear Regression & Gradient Descent

- Let's see the gradient descent algorithm in action:

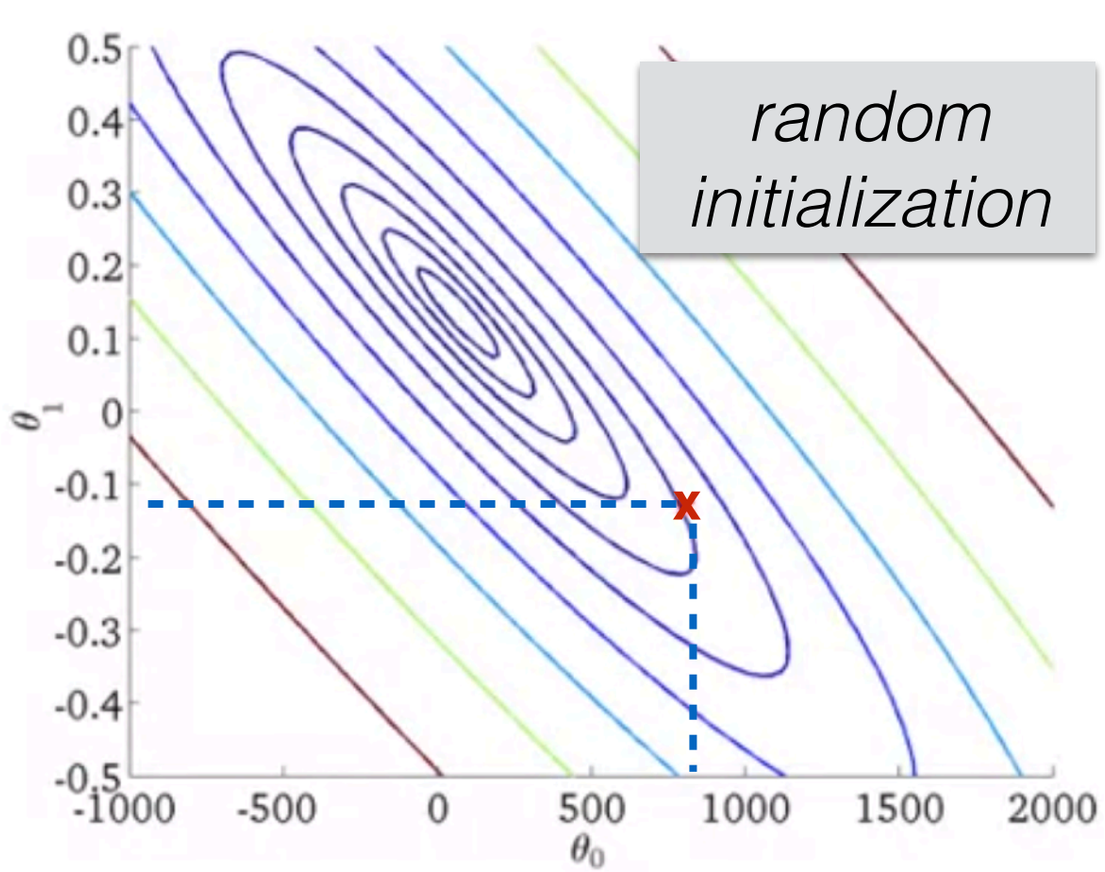
$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\ \mathrm{x}$

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



$\mathrm{J}(\theta_0,\theta_1)$

(function of the parameters θ_0,θ_1)

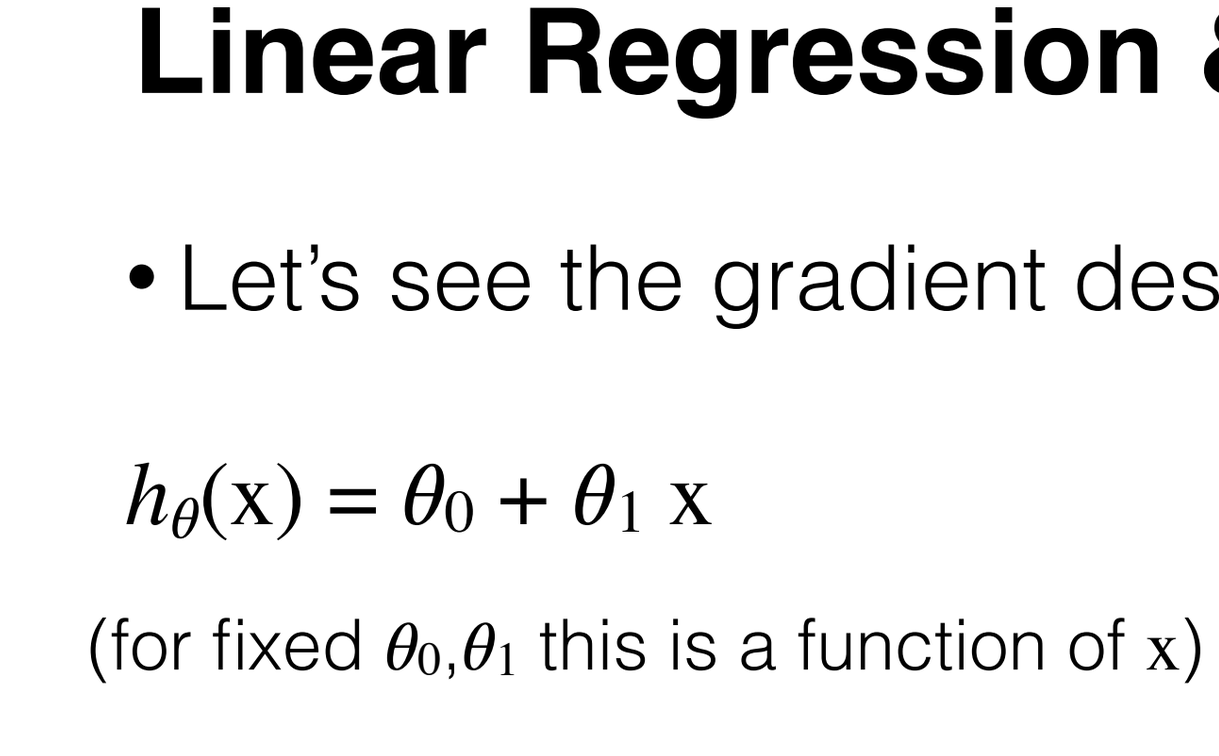


Linear Regression & Gradient Descent

- Let's see the gradient descent algorithm in action:

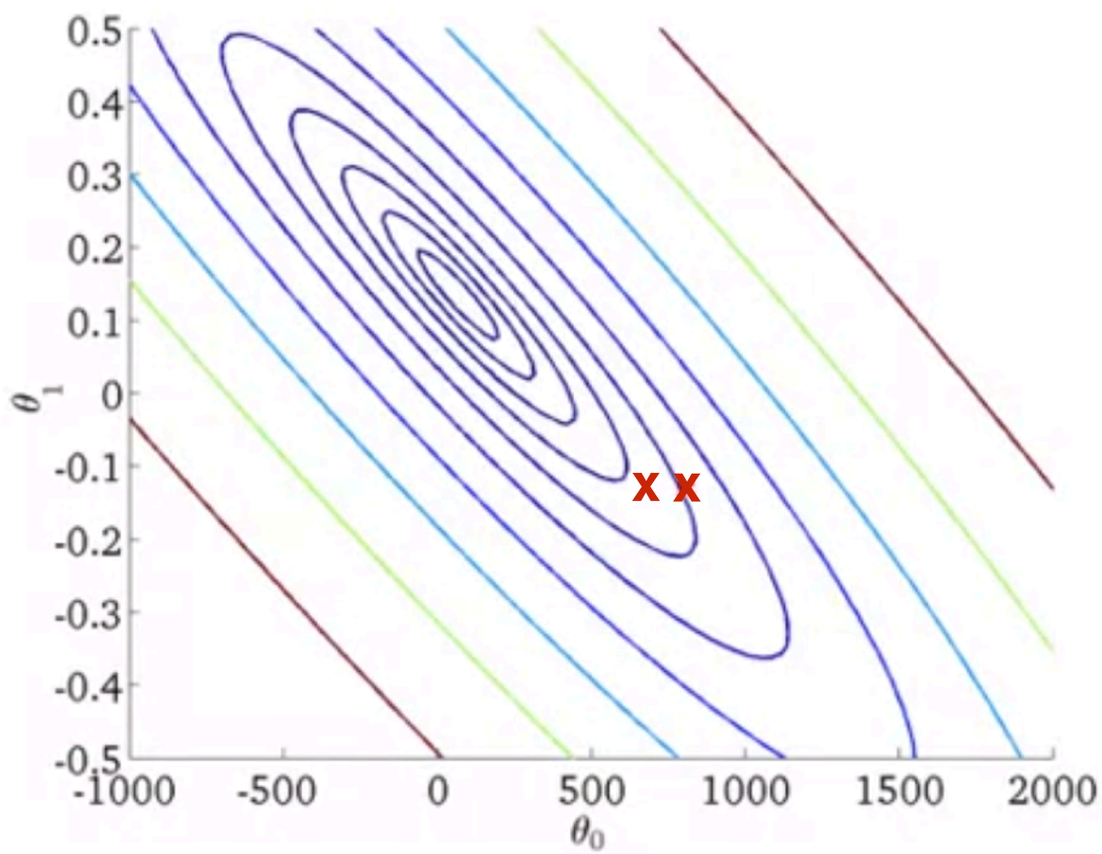
$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\ \mathrm{x}$

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



$\mathrm{J}(\theta_0,\theta_1)$

(function of the parameters θ_0,θ_1)

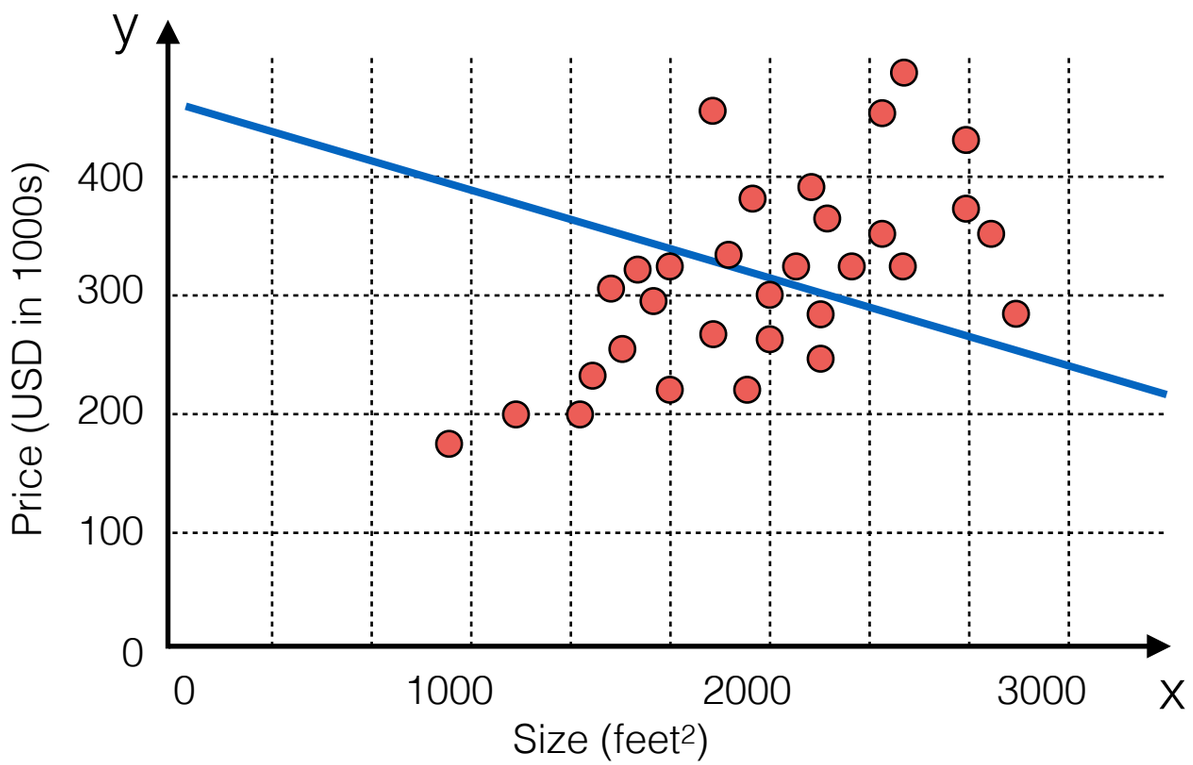


Linear Regression & Gradient Descent

- Let's see the gradient descent algorithm in action:

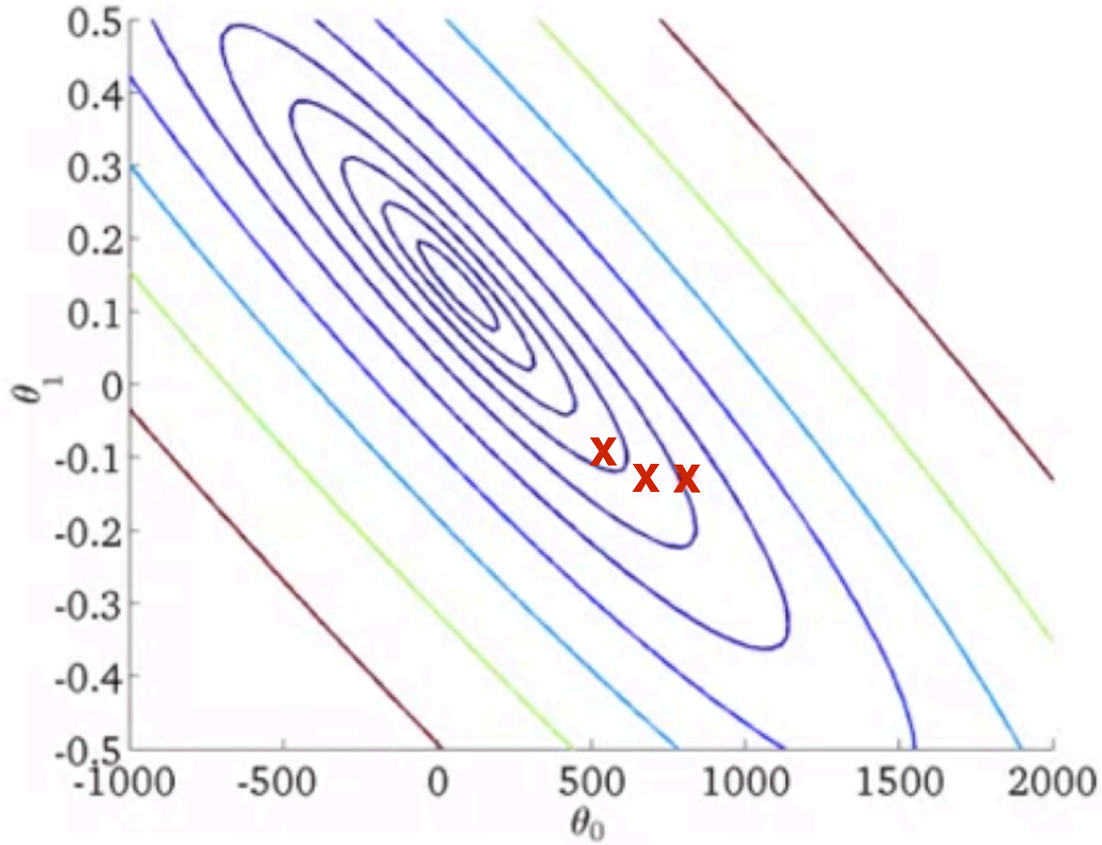
$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\ \mathrm{x}$

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



$\mathrm{J}(\theta_0,\theta_1)$

(function of the parameters θ_0,θ_1)

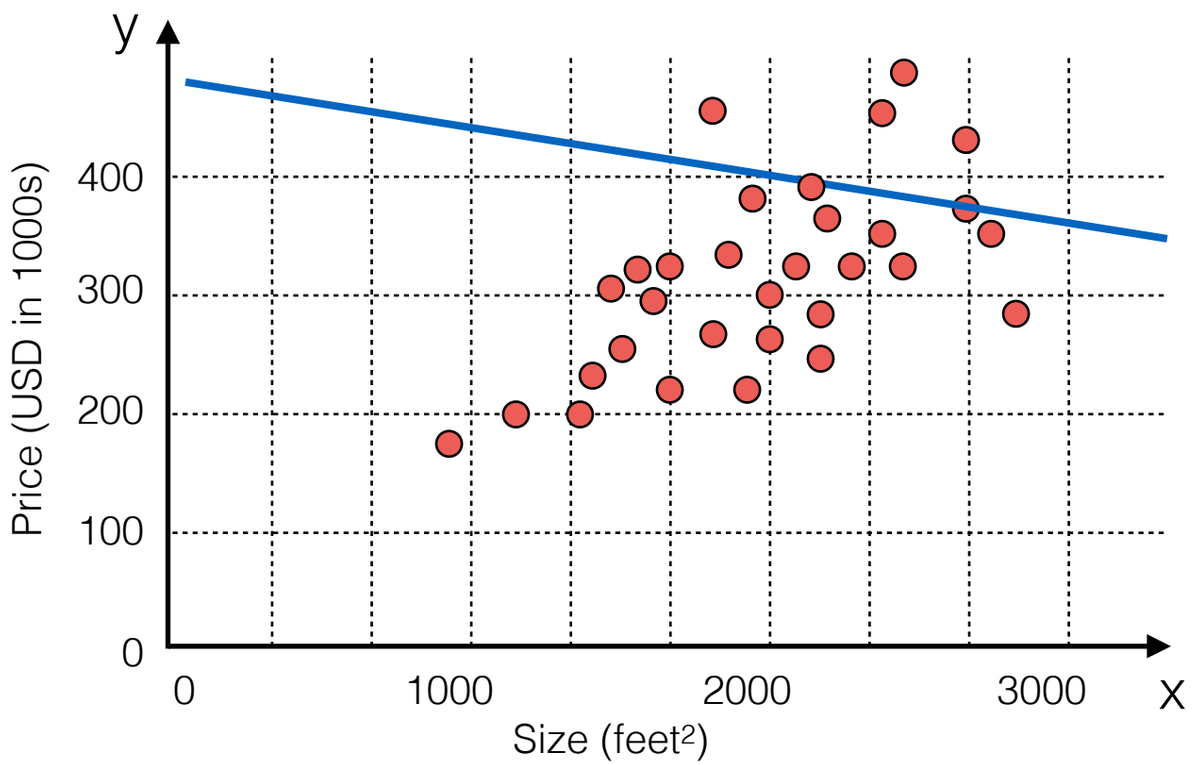


Linear Regression & Gradient Descent

- Let's see the gradient descent algorithm in action:

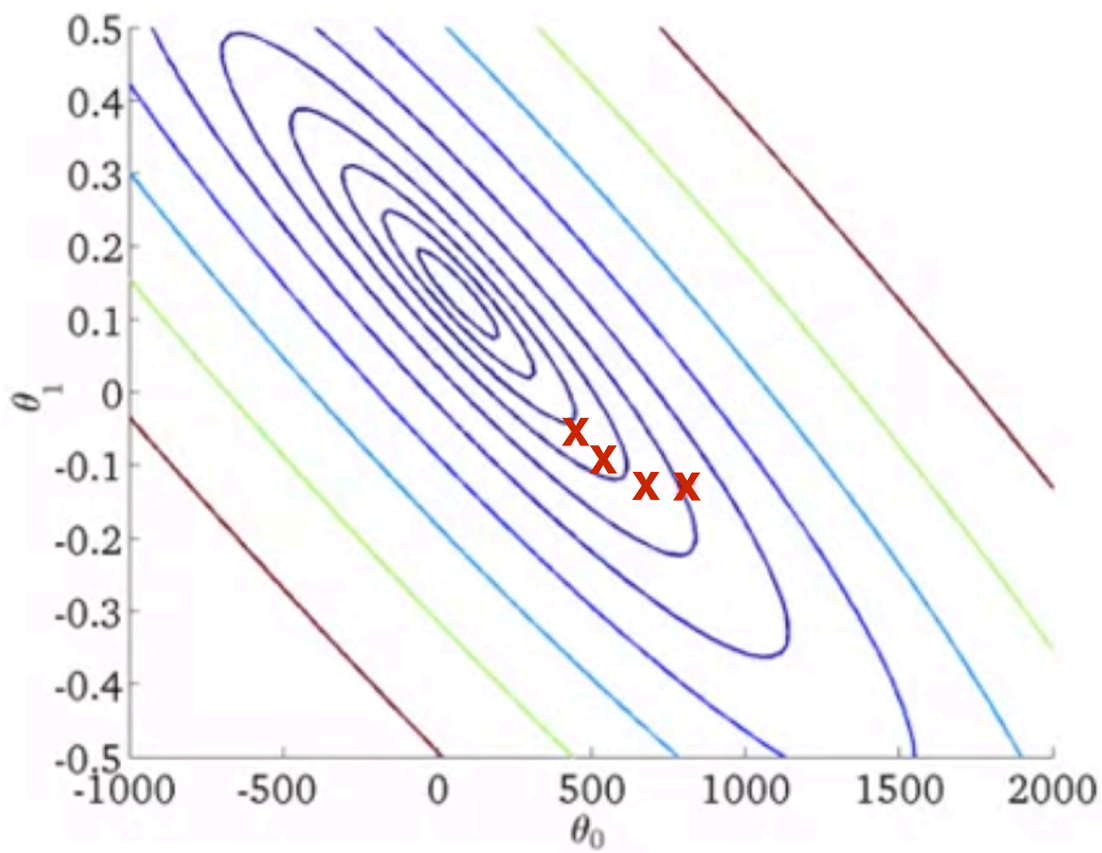
$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\ \mathrm{x}$

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



$\mathrm{J}(\theta_0,\theta_1)$

(function of the parameters θ_0,θ_1)

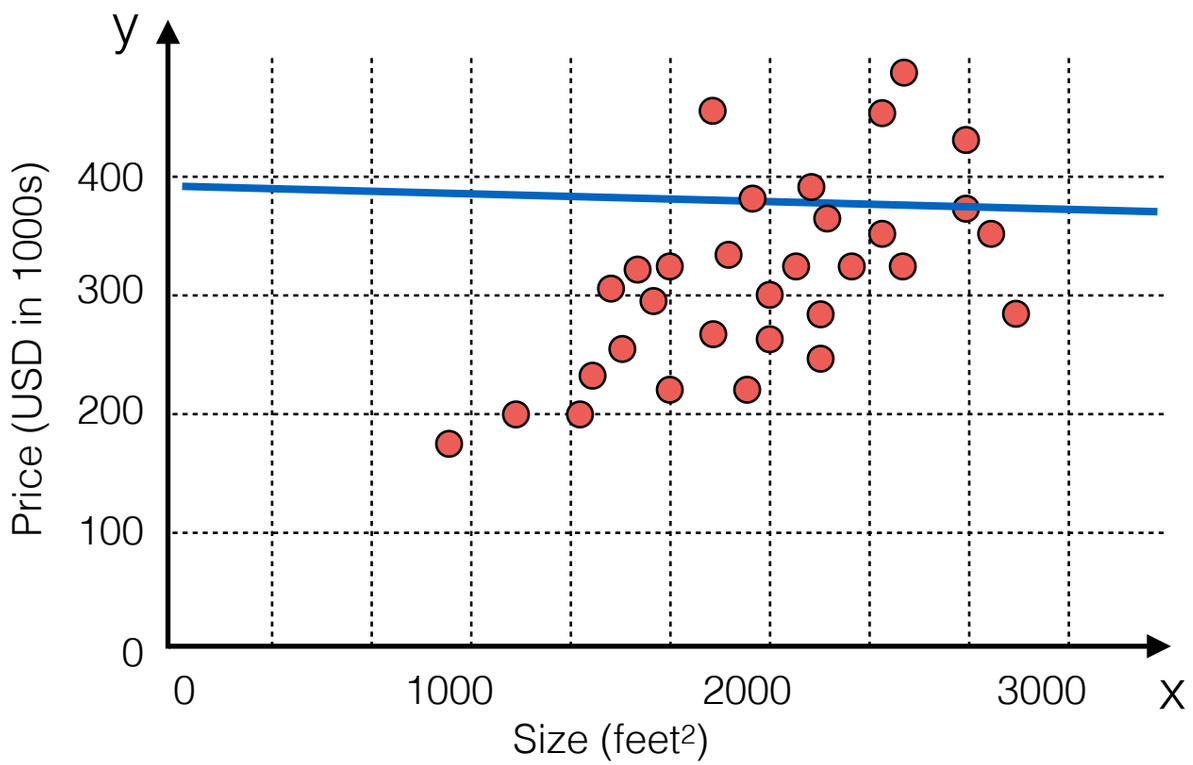


Linear Regression & Gradient Descent

- Let's see the gradient descent algorithm in action:

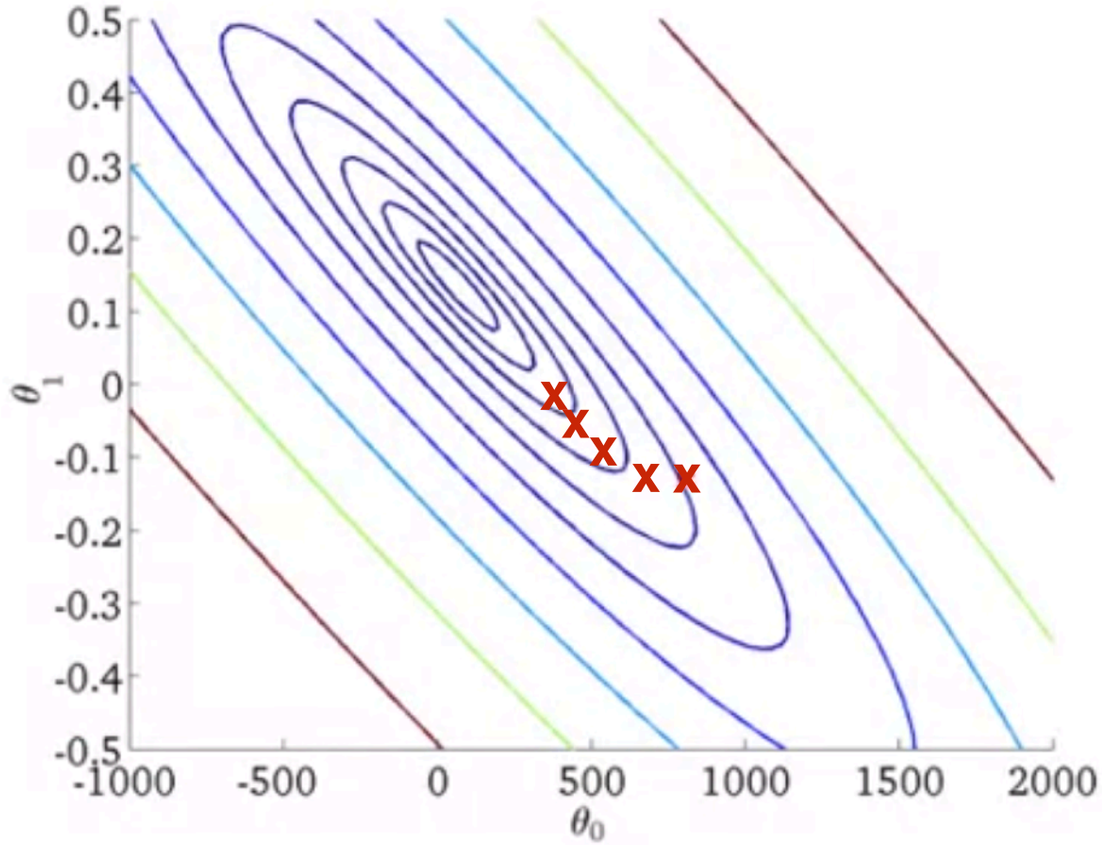
$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\ \mathrm{x}$

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



$\mathrm{J}(\theta_0,\theta_1)$

(function of the parameters θ_0,θ_1)

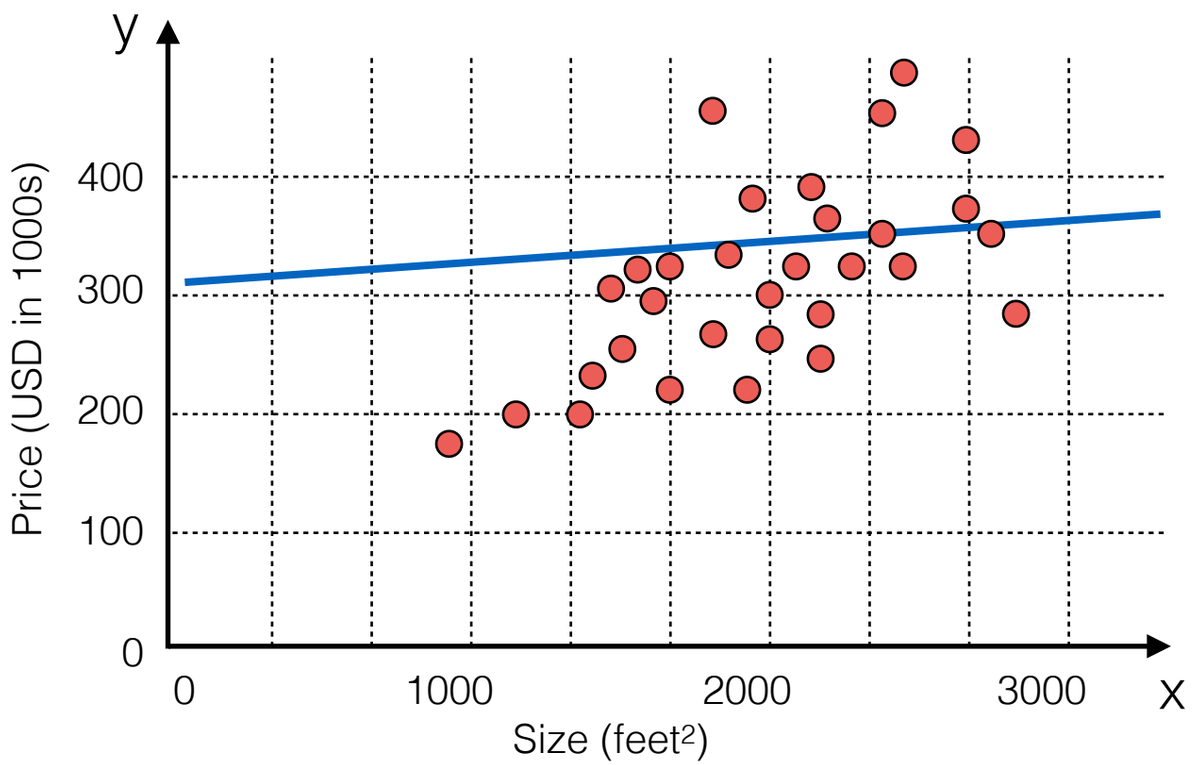


Linear Regression & Gradient Descent

- Let's see the gradient descent algorithm in action:

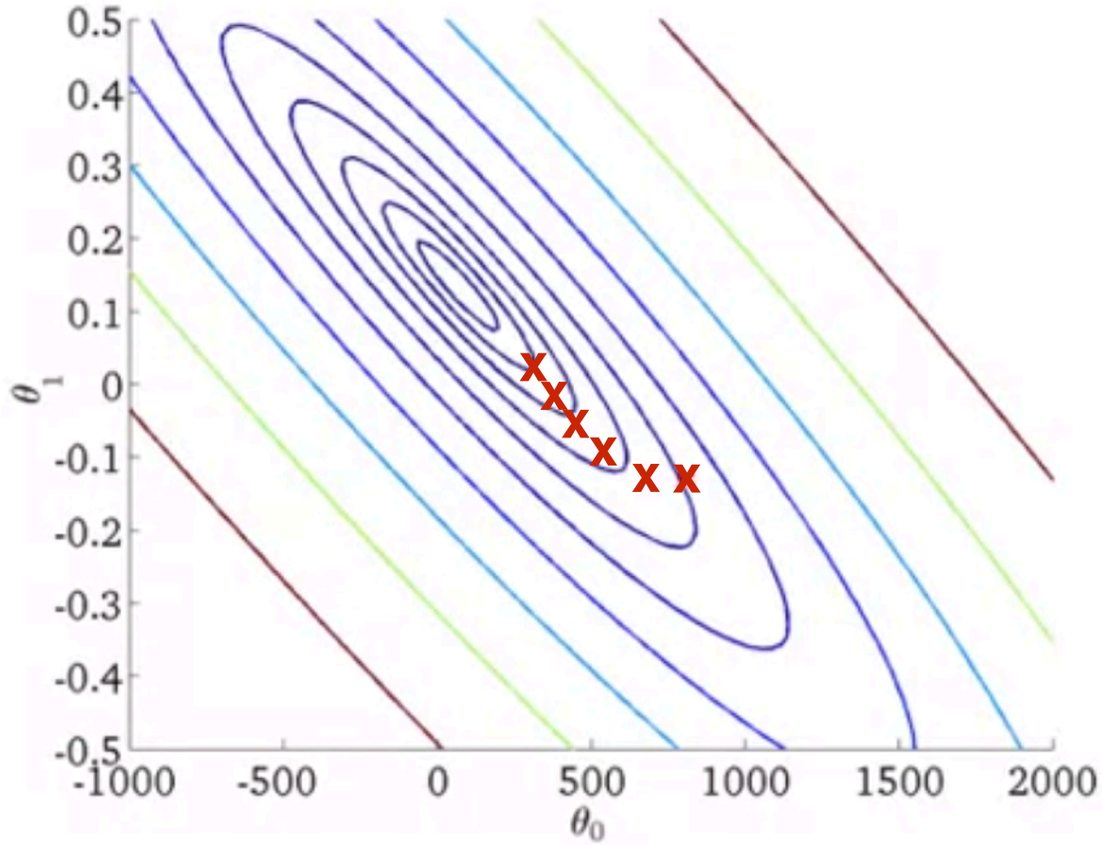
$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\ \mathrm{x}$

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



$\mathrm{J}(\theta_0,\theta_1)$

(function of the parameters θ_0,θ_1)

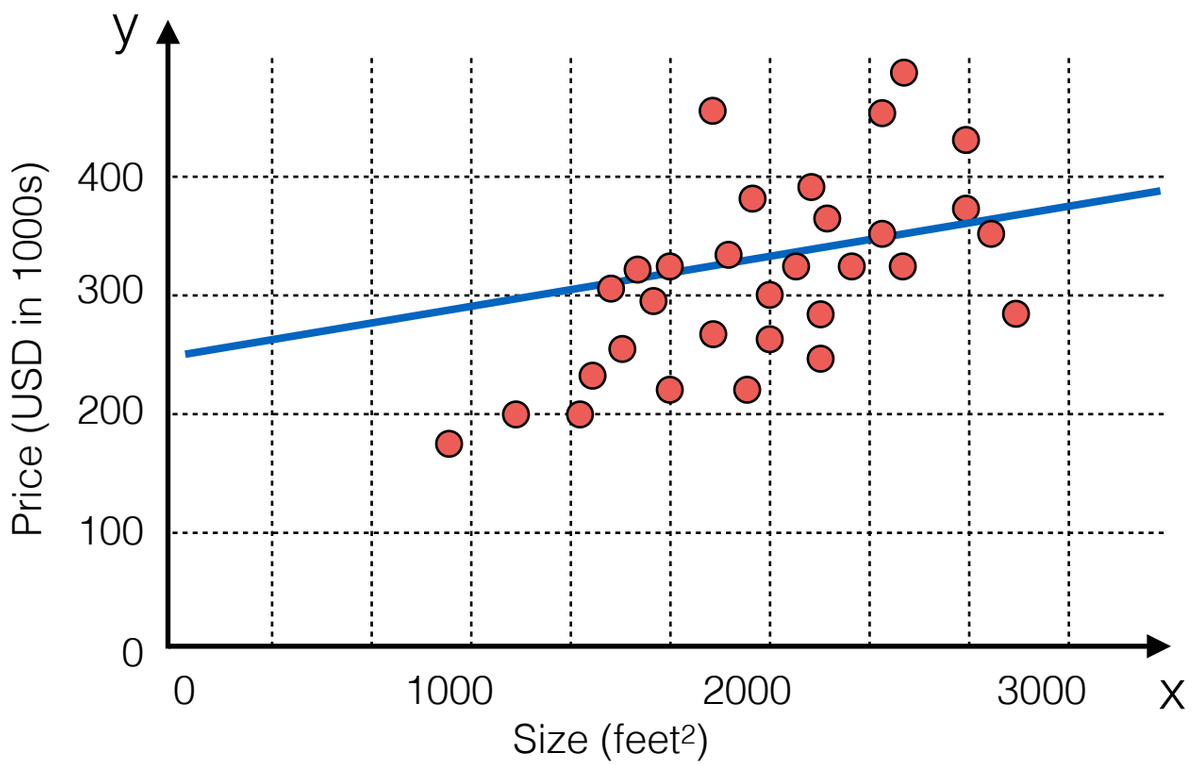


Linear Regression & Gradient Descent

- Let's see the gradient descent algorithm in action:

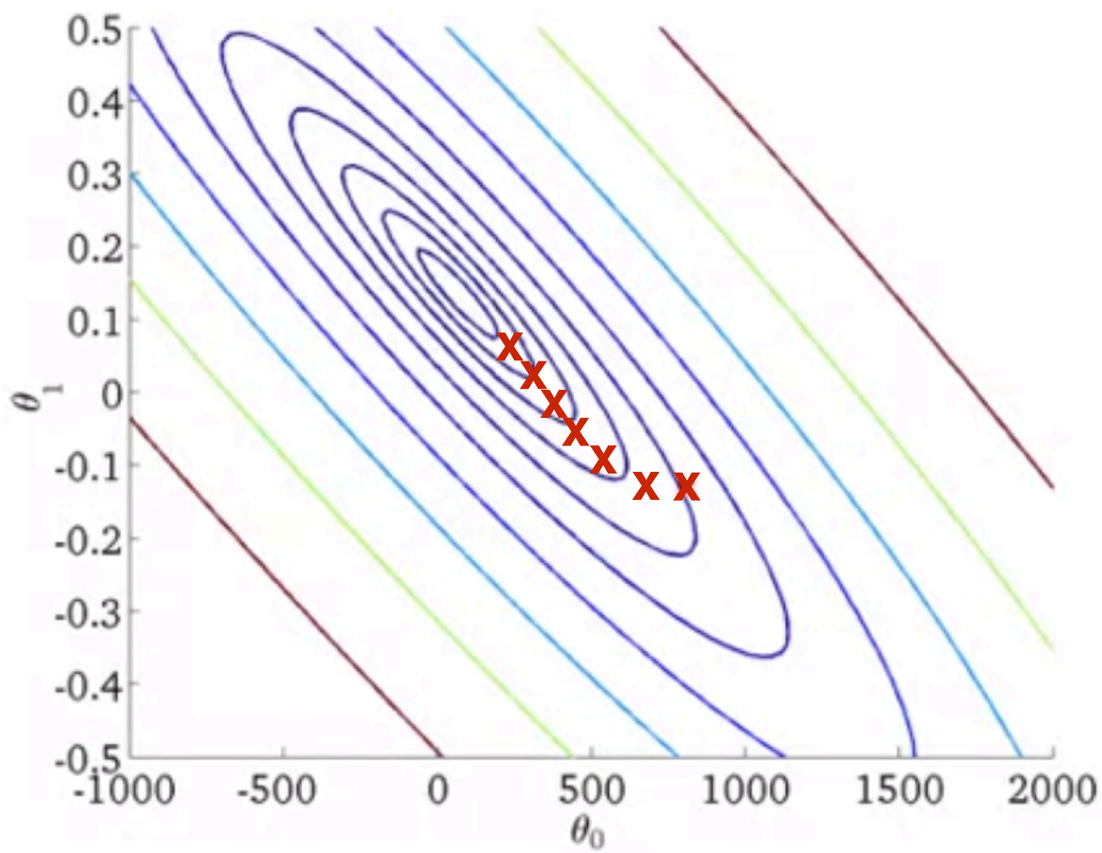
$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\ \mathrm{x}$

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



$\mathrm{J}(\theta_0,\theta_1)$

(function of the parameters θ_0,θ_1)

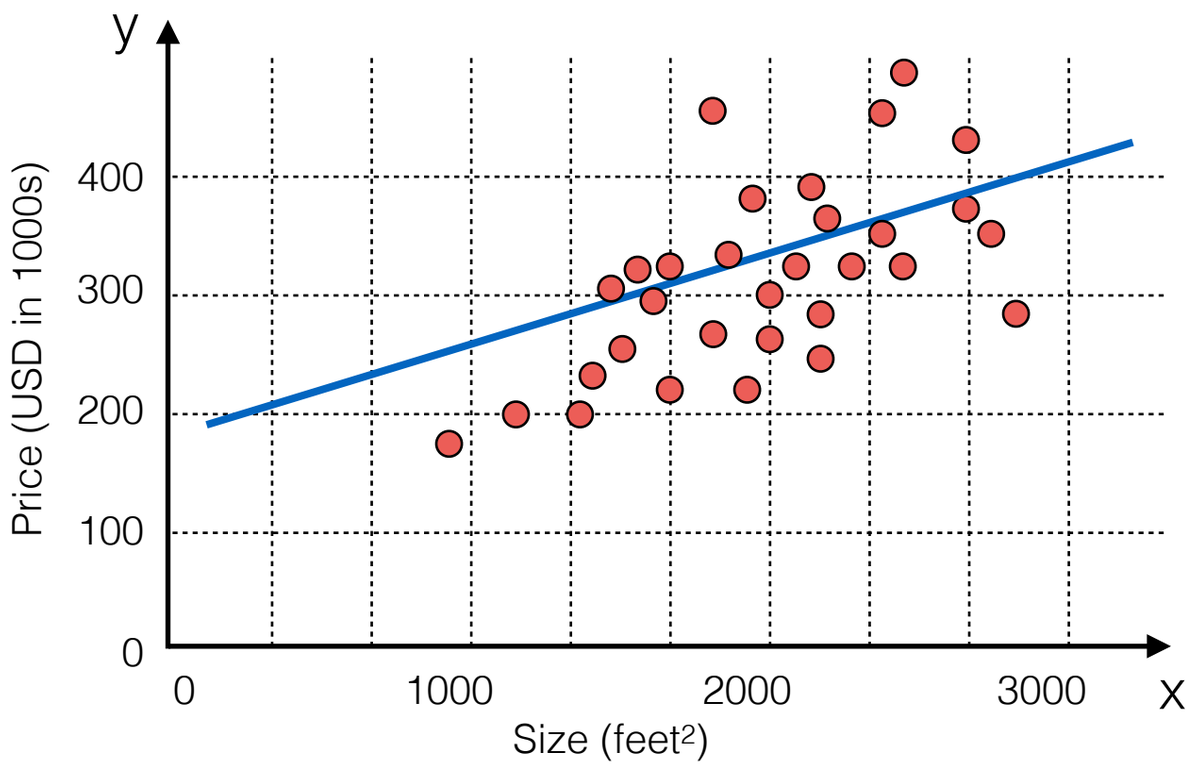


Linear Regression & Gradient Descent

- Let's see the gradient descent algorithm in action:

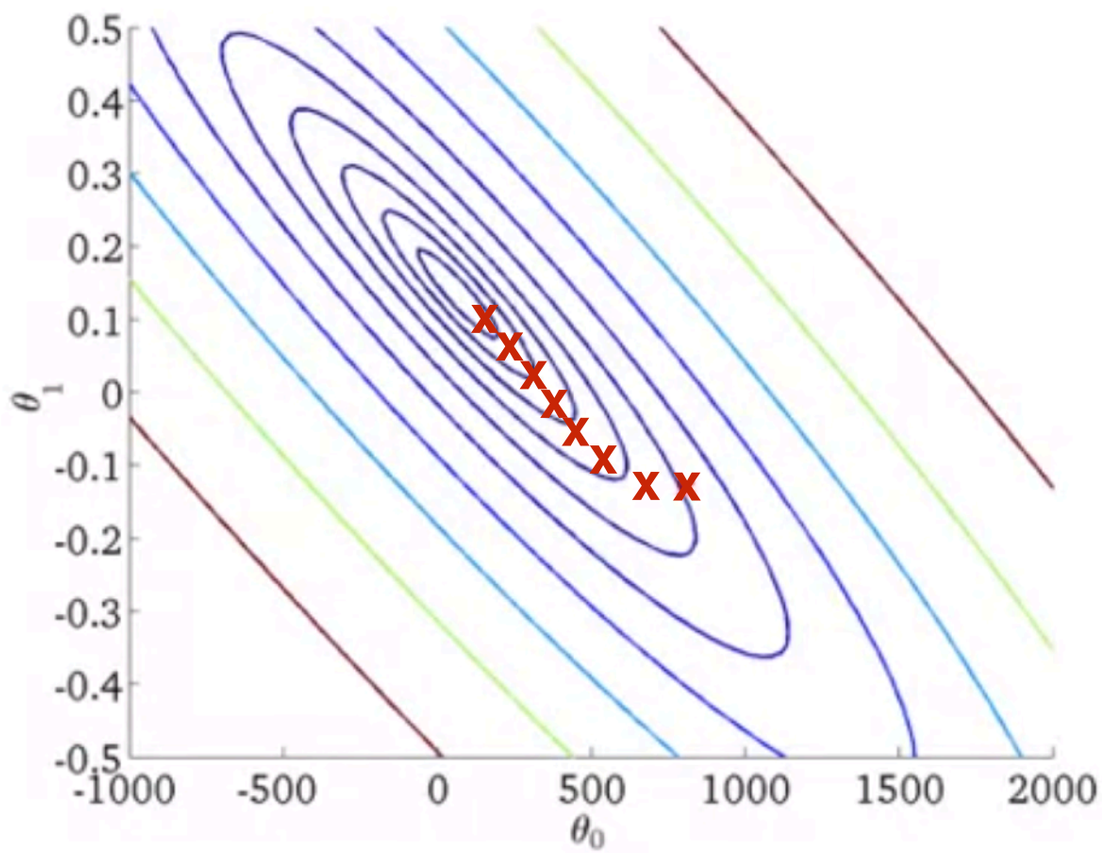
$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\ \mathrm{x}$

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



$\mathrm{J}(\theta_0,\theta_1)$

(function of the parameters θ_0,θ_1)

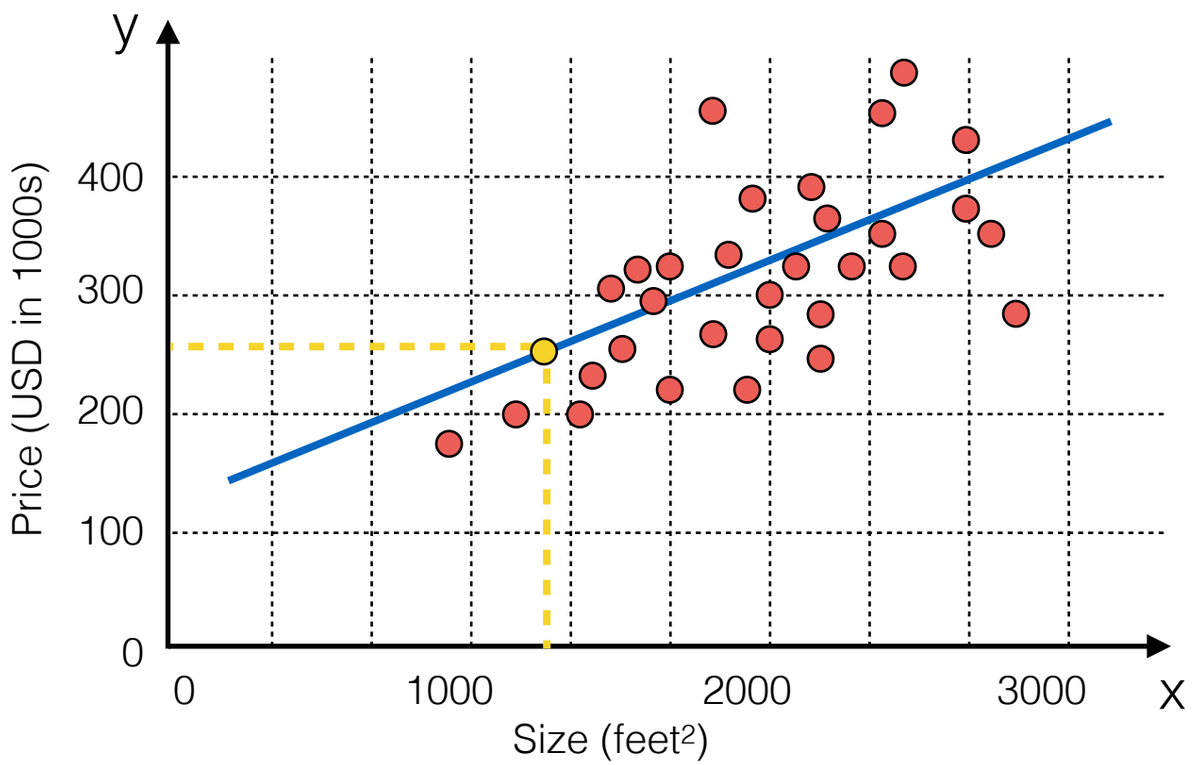


Linear Regression & Gradient Descent

- Let's see the gradient descent algorithm in action:

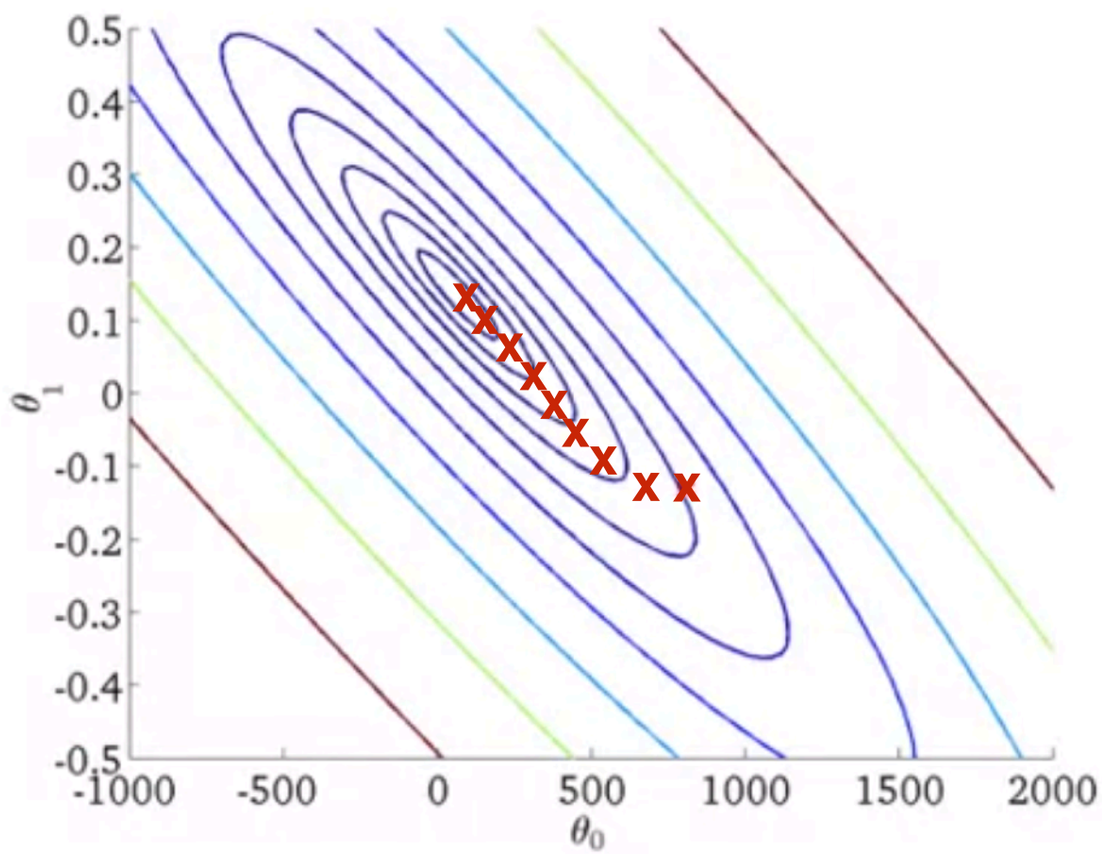
$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\ \mathrm{x}$

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



$\mathrm{J}(\theta_0,\theta_1)$

(function of the parameters θ_0,θ_1)



A bit more on Gradient Descent

- We have introduced *batch gradient descent*
  (i.e. each step of gradient descent uses all training examples)
  - There is another way to optimize across the training set…

- Stochastic Gradient Descent: update the parameters for each training case in turn, according to its own gradients

```
Randomly shuffle examples in the training set
for i=1 to m do{
```

$$\theta_0 := \theta_0 - \eta \ (\theta_0 + \theta_1 x^{(i)} - y^{(i)})$$

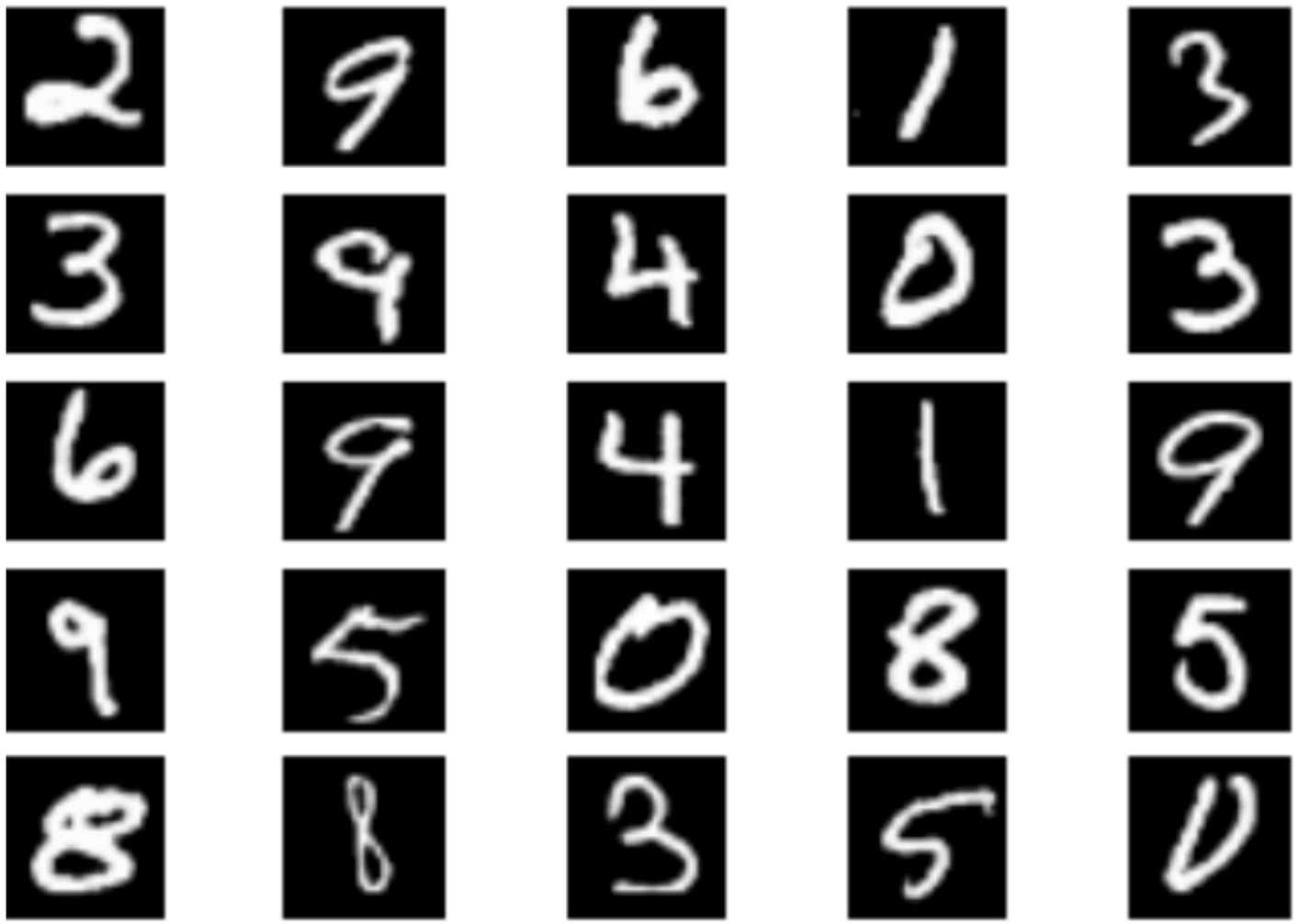
$$\theta_1 := \theta_1 - \eta \ (\theta_0 + \theta_1 x^{(i)} - y^{(i)}) \ x^{(i)}$$

```
}
```

Underlying assumption:
samples are independent and identically distributed (i.i.d.)

Learning is useful in many tasks

- **Classification:** determine to which discrete category a specific example belongs to



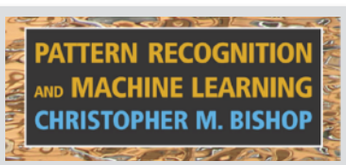
Example 1

What digit is this?

Linear classifiers

- **Classification:** determine to which discrete category a specific example belongs to

- Other examples:
  - Email: spam *vs* not spam (*ham*)
  - Online transactions: fraudulent *vs* not fraudulent
  - Tumor: malignant *vs* benign



Pages 179-195

Classification *vs* Regression

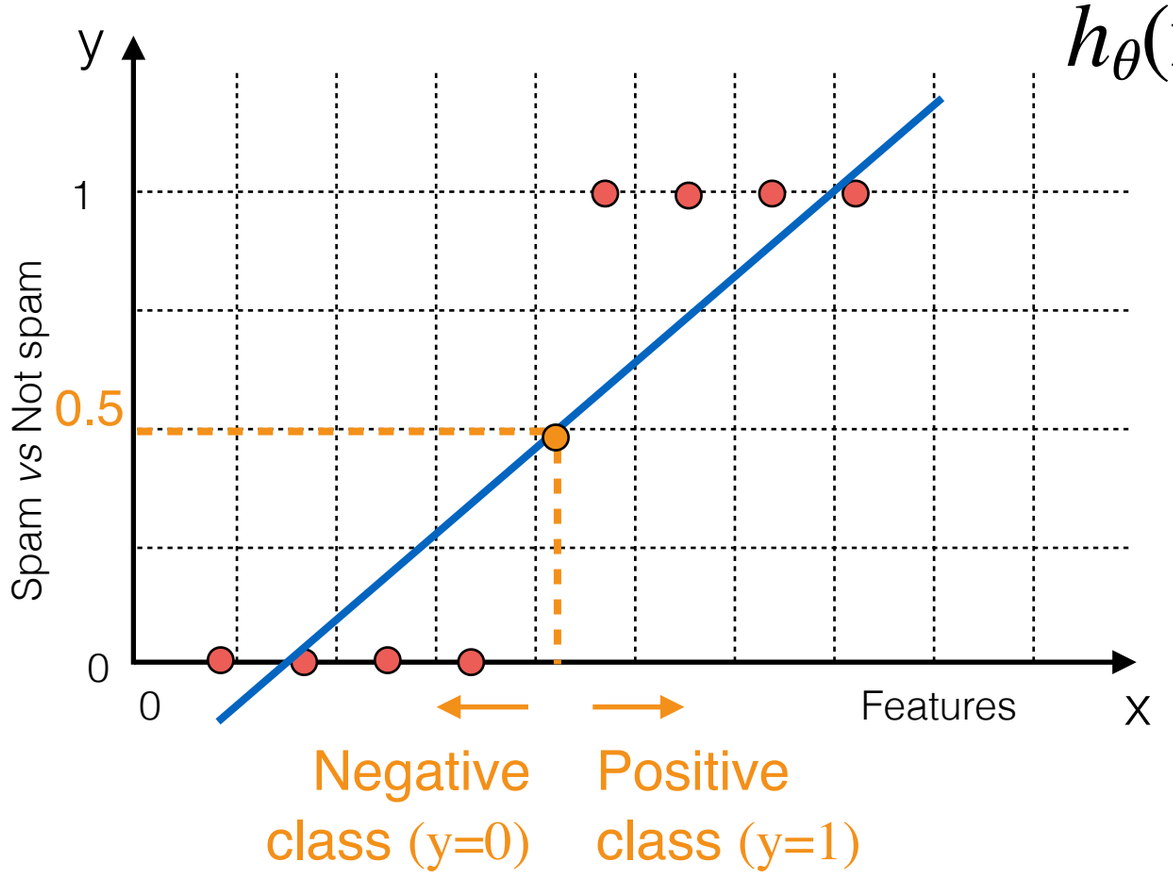
- Categorical outputs called labels (or classes)
  - e.g. yes/no, 1/2/3/…/9, cat/dog/person/…
  - Then we are interested in: $h \sim f: X \rightarrow Y$, where Y is categorical (while in regression typically $Y=R$)

- Binary classification: two possible labels
- Multi-class classification: multiple possible labels

We will first look at binary problems and then discuss multi-class problems

Classification as Regression

- Can we do (binary) classification using what we have learned until now?



$$h_\theta(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}}\,\mathbf{x}$$

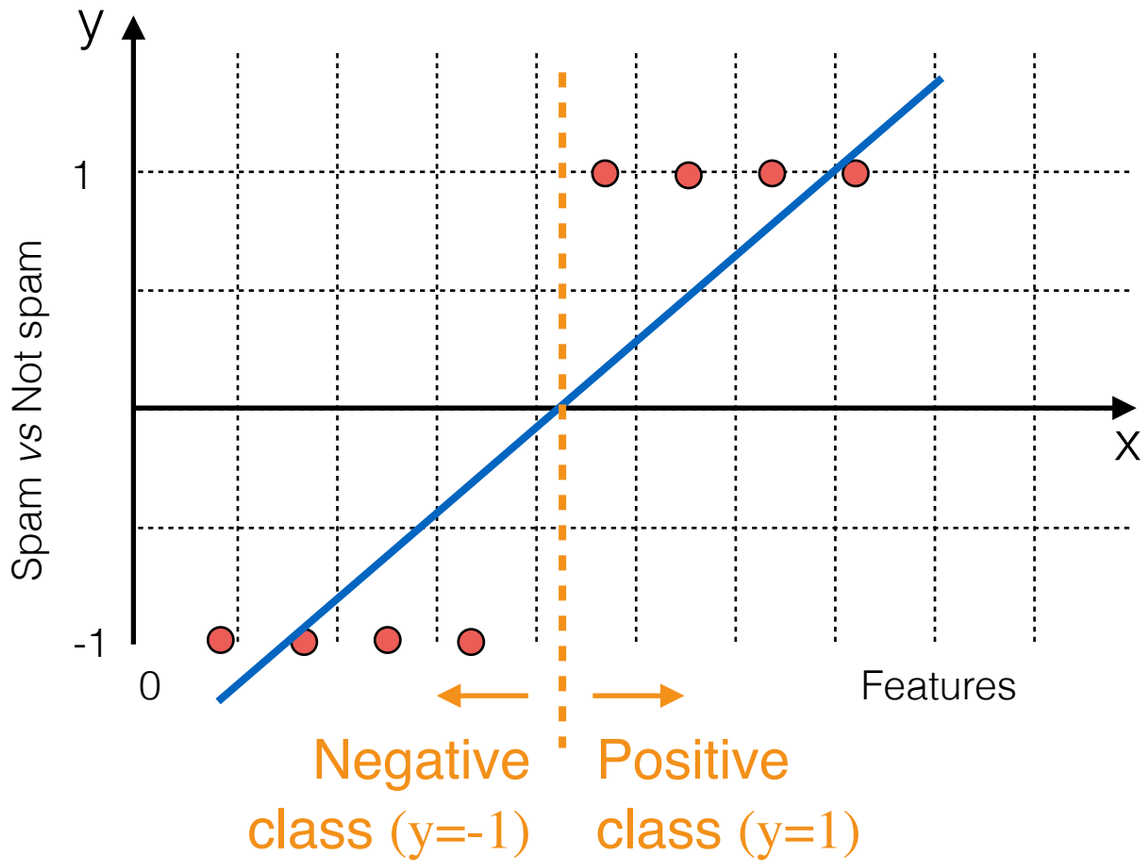
Threshold classifier:

- If $h_\theta(\mathbf{x}) \geq 0.5$, predict $\mathrm{y} = 1$
- If $h_\theta(\mathbf{x}) < 0.5$, predict $\mathrm{y} = 0$

Classification as Regression

- Let's use a slightly different notation

$$h_\theta(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}}\,\mathbf{x}$$



Threshold classifier:

- If $h_\theta(\mathbf{x}) \geq 0$, predict $\mathrm{y} = 1$
- If $h_\theta(\mathbf{x}) < 0$, predict $\mathrm{y} = -1$

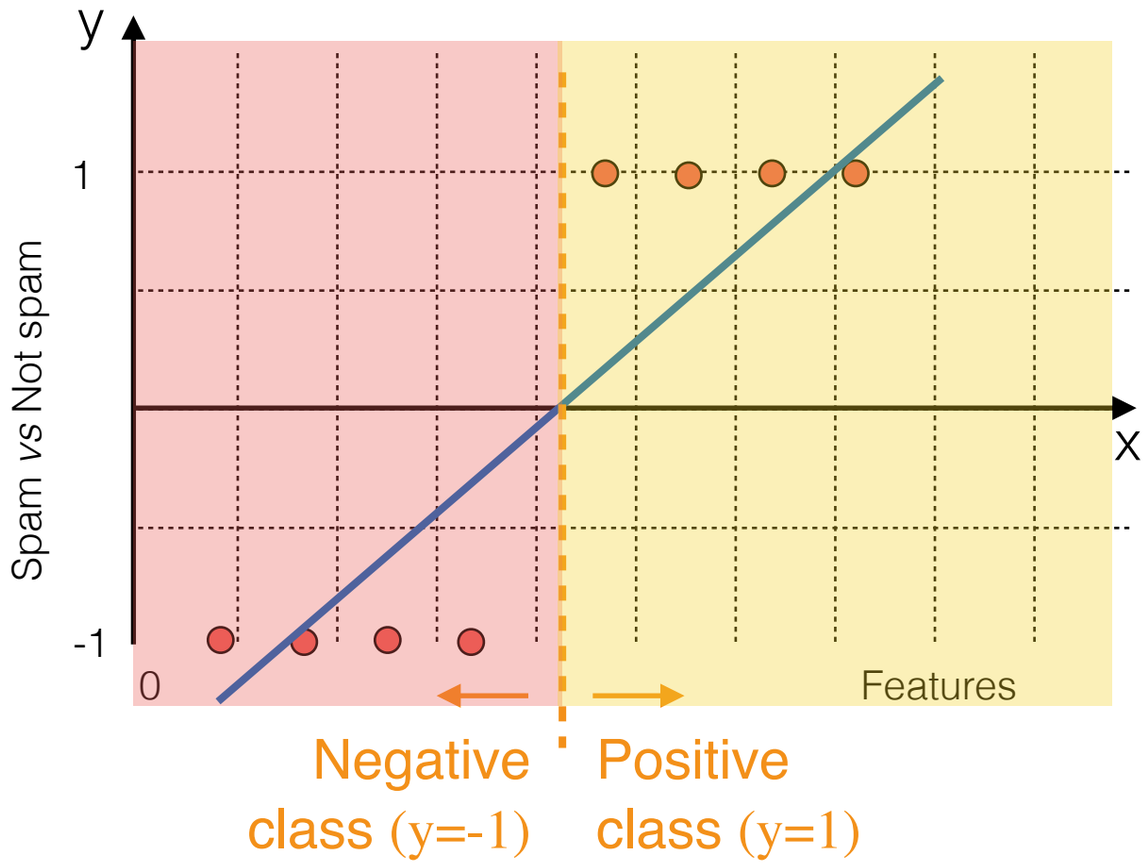
Decision rule *(mathematically)*:

- $\mathrm{y} = \mathrm{sign}(h_\theta(\mathbf{x}))$

Linear Classification

- This specifies a *linear classifier*: it has a linear boundary (hyperplane) which separates the space

$$h_\theta(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}}\,\mathbf{x}$$



Decision rule:

- $\mathrm{y} = \mathrm{sign}(h_\theta(\mathbf{x}))$

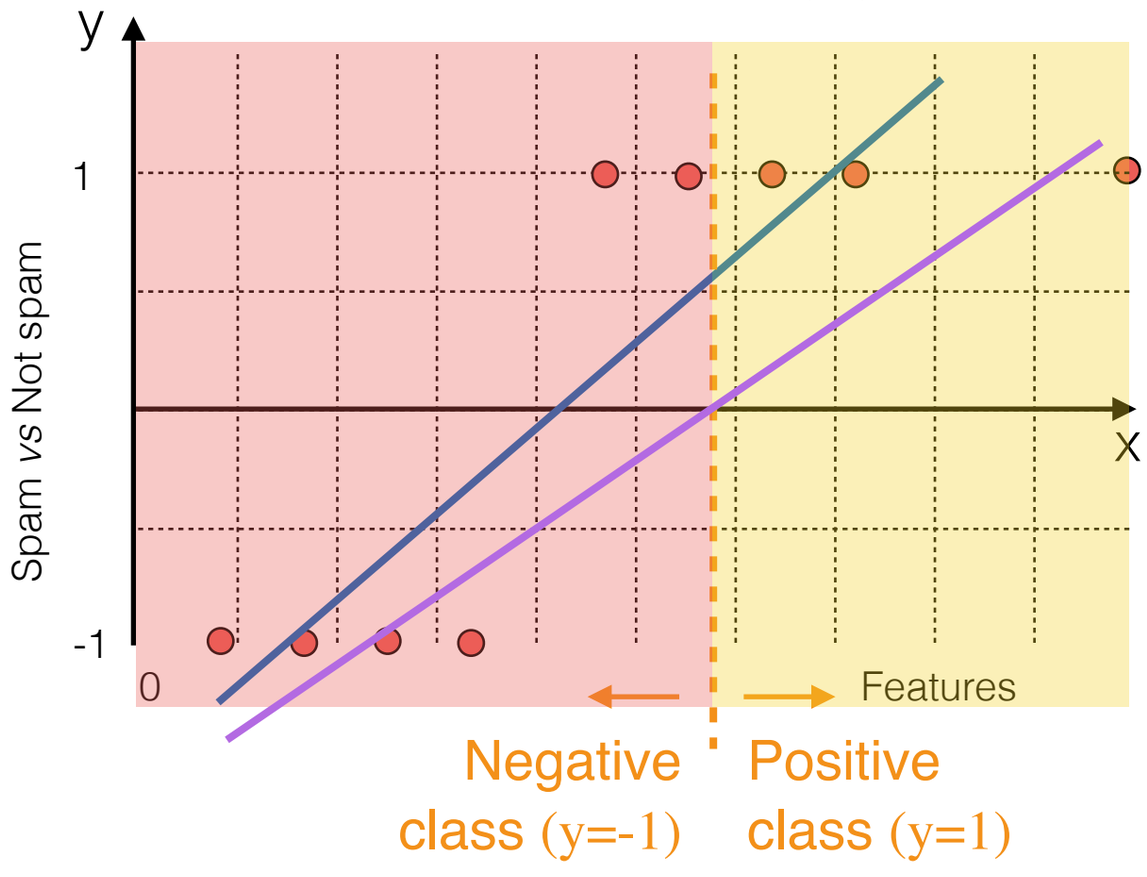
The linear boundary separates the space into two "half-spaces"

In 1D this is simply a threshold

Linear Classification

- Applying linear regression to classification tasks is not always a great idea…

$$h_\theta(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}}\,\mathbf{x}$$



Decision rule:

- $y = \mathrm{sign}(h_\theta(\mathbf{x}))$

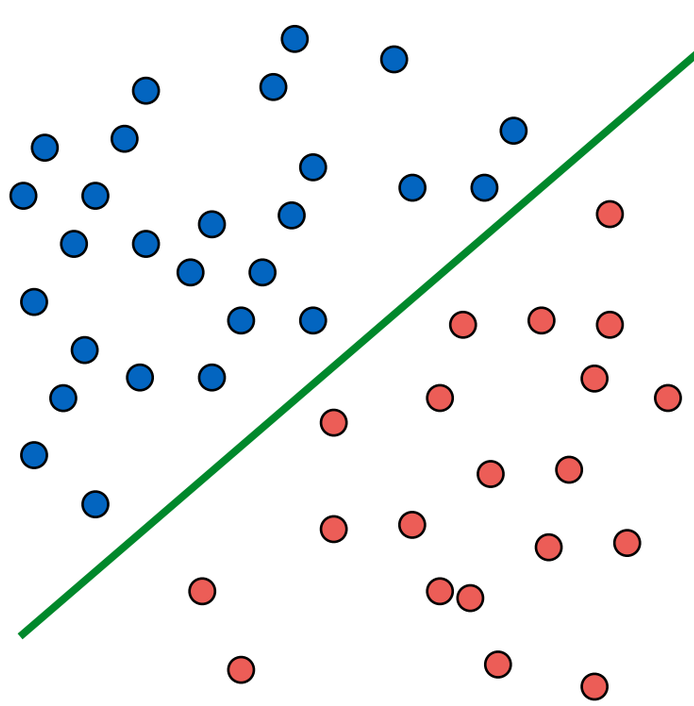
The linear boundary separates the space into two "half-spaces"

In 1D this is simply a threshold

Linear Classification

- This specifies a *linear classifier*: it has a linear boundary (hyperplane) which separates the space

$$h_\theta(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}}\,\mathbf{x}$$



Decision rule:

- $\mathrm{y} = \mathrm{sign}(h_\theta(\mathbf{x}))$

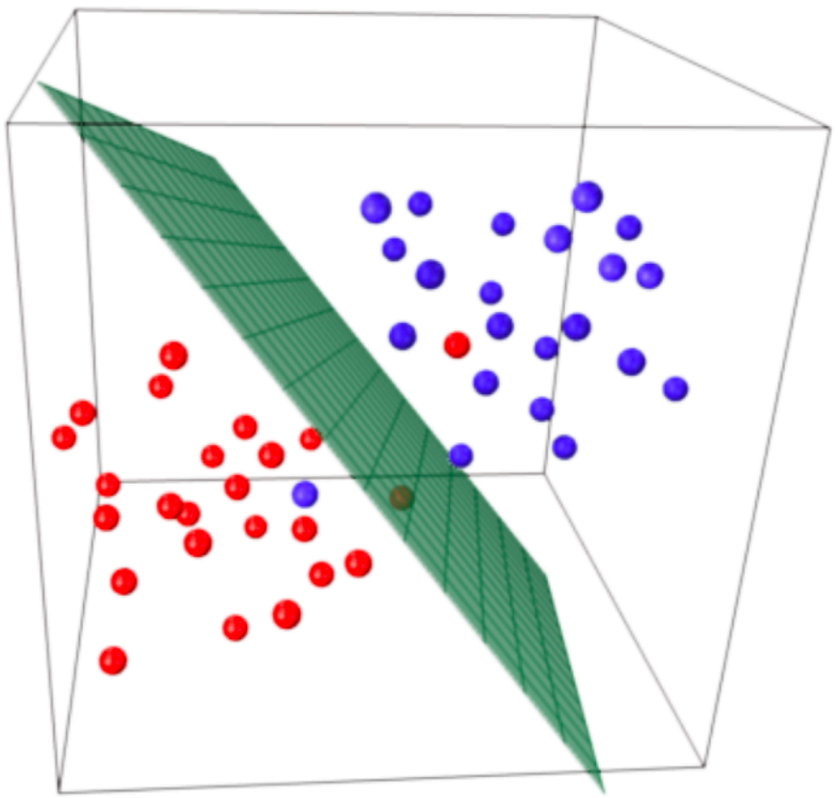
The linear boundary separates the space into two “half-spaces”

In 2D this is a line

Linear Classification

- This specifies a *linear classifier*: it has a linear boundary (hyperplane) which separates the space

$$h_\theta(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}}\,\mathbf{x}$$



Decision rule:

- $\mathrm{y} = \mathrm{sign}(h_\theta(\mathbf{x}))$

The linear boundary separates the space into two "half-spaces"

In 3D this is a plane

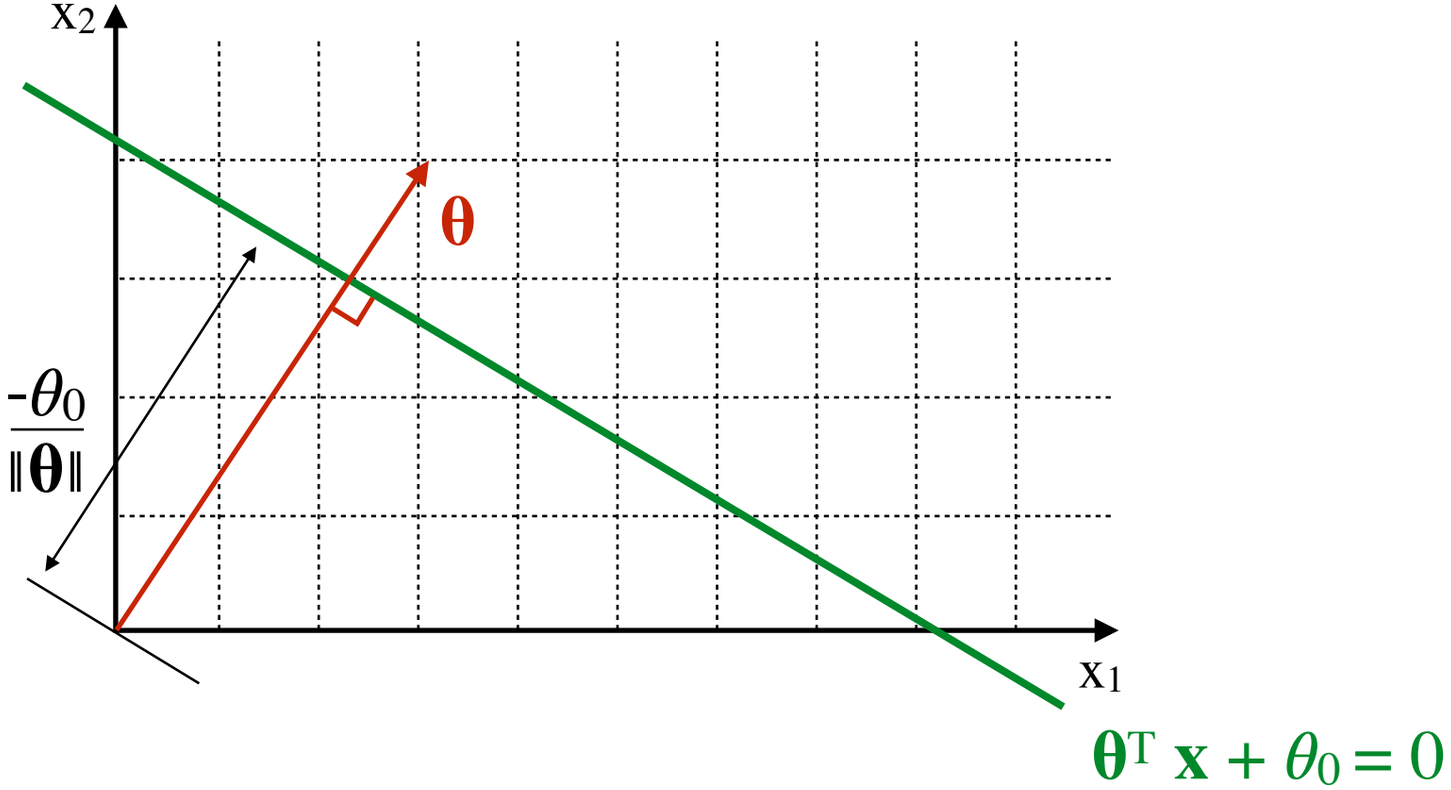
Image credit: S. Fidler

Geometric Interpretation

- What about higher-dimensional spaces?

$\boldsymbol{\theta}^T \mathbf{x} = 0$ a line passing through the origin and orthogonal to $\boldsymbol{\theta}$

$\boldsymbol{\theta}^T \mathbf{x} + \theta_0 = 0$ shifts it by θ_0 ← *Note: this is usually referred as to the "bias term"*

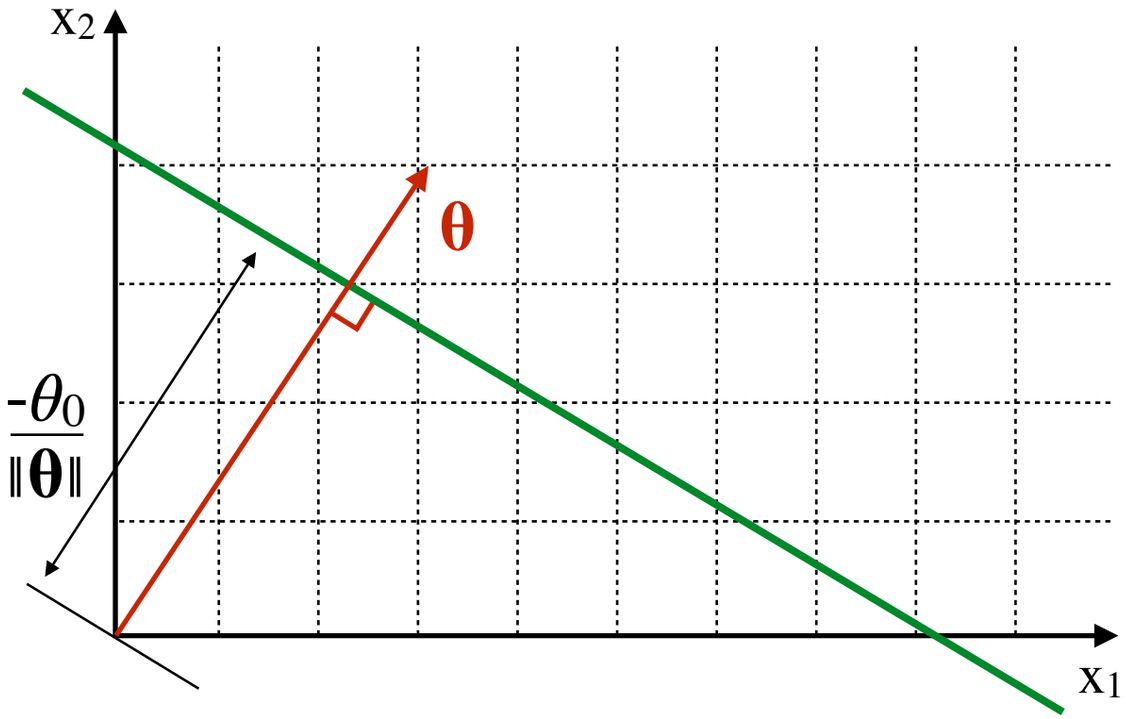


Geometric Interpretation

- What about higher-dimensional spaces?

$\boldsymbol{\theta}^T \mathbf{x} = 0$ a line passing through the origin and orthogonal to $\boldsymbol{\theta}$

$\boldsymbol{\theta}^T \mathbf{x} + \theta_0 = 0$ shifts it by θ_0 ← *Note: this is usually referred as to the "bias term"*



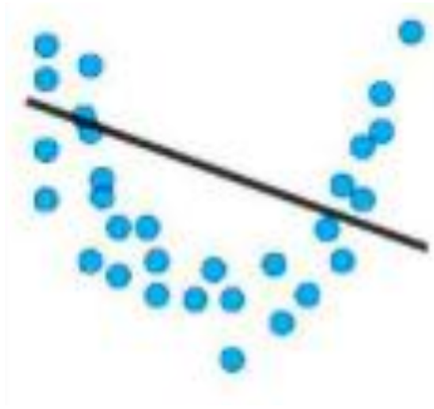
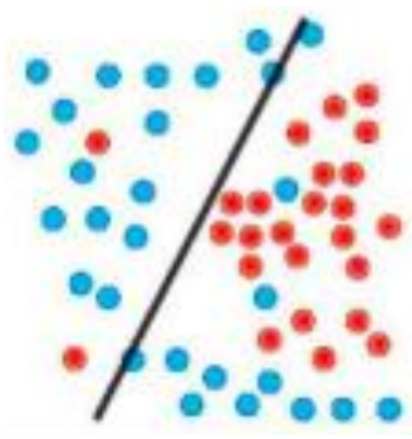
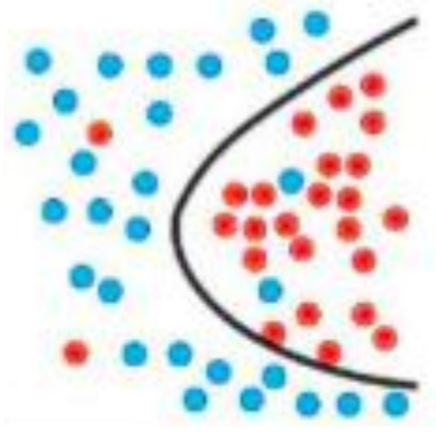
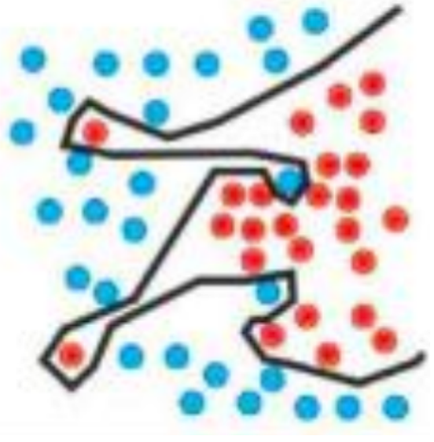
$\boldsymbol{\theta}^T \mathbf{x} + \theta_0 = 0$

A bit more about the notation

We are using this trick/assumption:

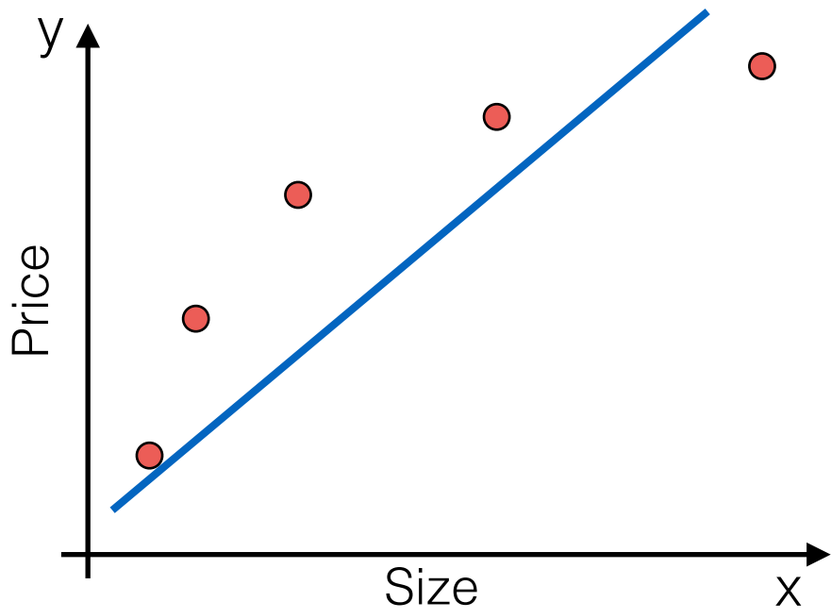
$$h_\theta(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}^T \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

Recap: Bias-Variance Tradeoff

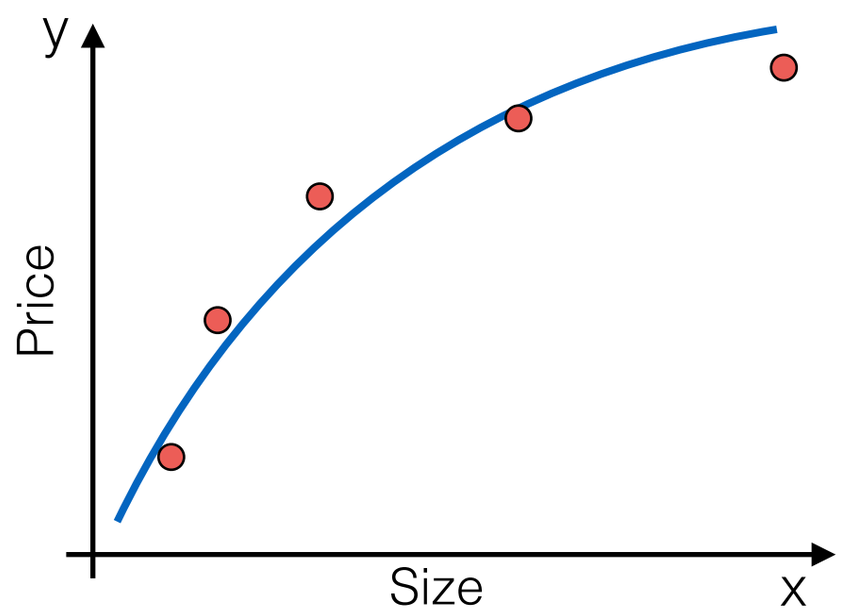
	Underfitting	Optimal	Overfitting
Regression			
Classification			

A bit more on Linear Regression

Underfitting

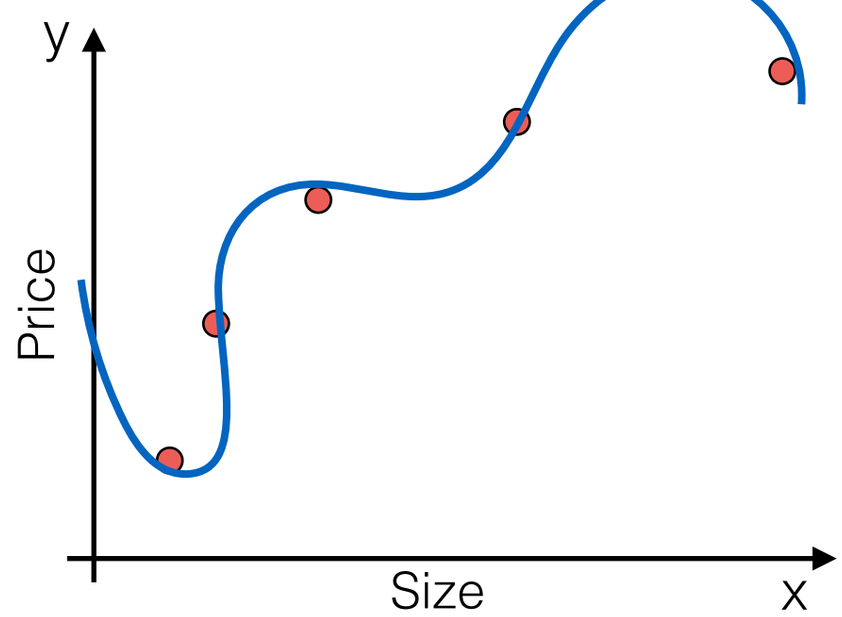


$h_\theta(x) = \theta_0 + \theta_1 x$



$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

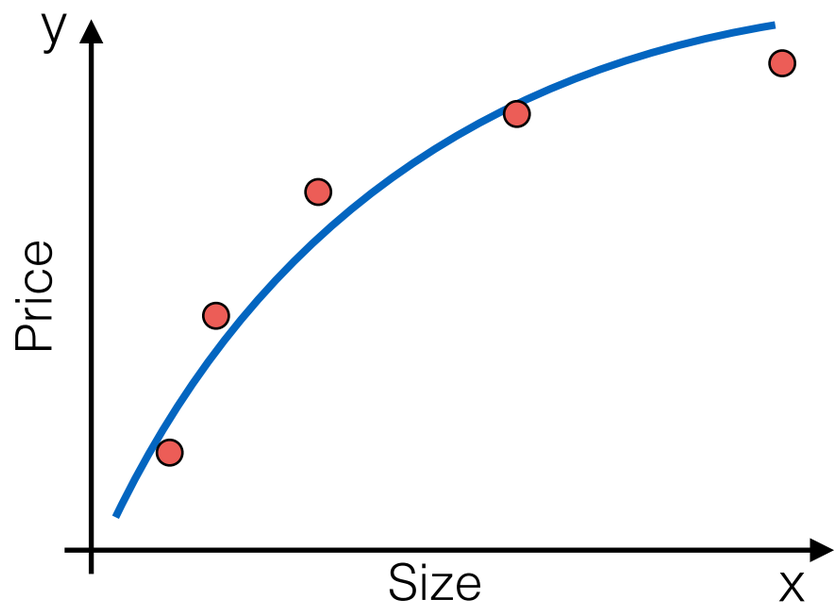
Overfitting



$\theta_0+\theta_1 x+\theta_2 x^2+\theta_3 x^3+\theta_4 x^4$

A bit more on Linear Regression

- Features and Polynomial Regression



$$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\,\mathrm{x} + \theta_2\,\mathrm{x}^2$$

How to get this $h_\theta(\mathrm{x})$ from our linear regression model $\boldsymbol{\theta}^{\mathrm{T}}\,\mathbf{x}$?

$$\Rightarrow \quad h_\theta(\mathrm{x}) = \begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}\mathrm{x}_0\\ \mathrm{x}_1\\ \mathrm{x}_2\end{bmatrix} = \begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}1\\ \mathrm{x}\\ \mathrm{x}^2\end{bmatrix}$$

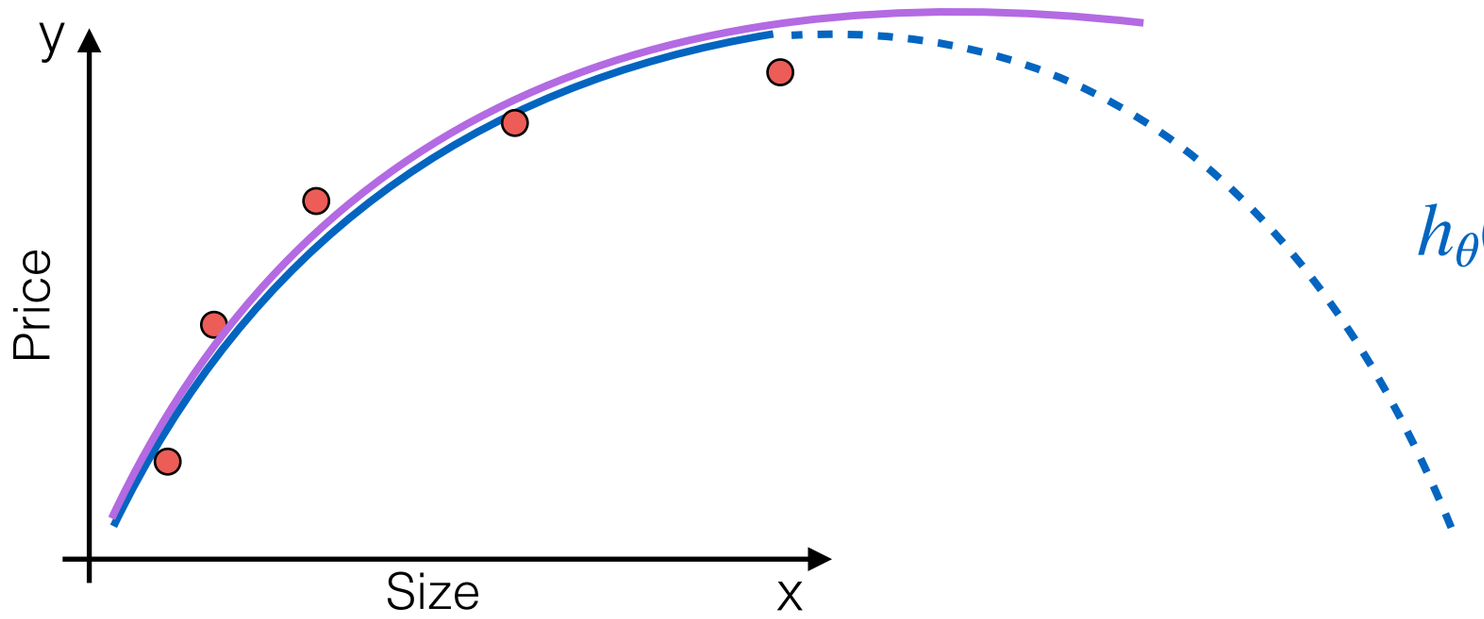
x_1: size of house

x_2: (size of house)^2

Note: if you choose your features this way, then feature scaling becomes very important

A bit more on Linear Regression

- Features and Polynomial Regression



$$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\,\mathrm{x} + \theta_2\,\mathrm{x}^2$$

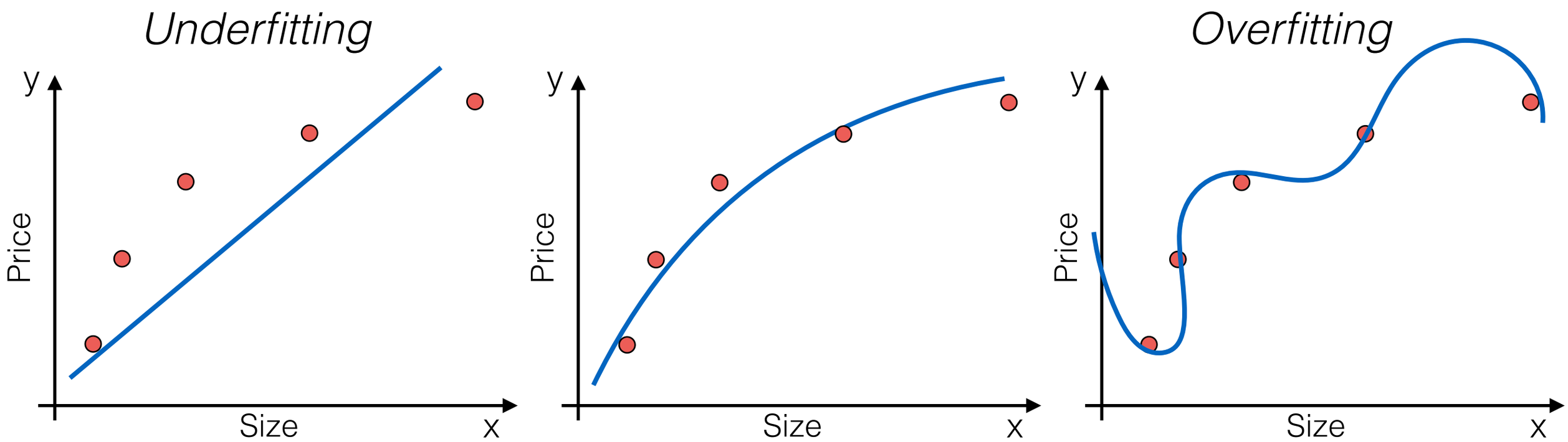
$$h_\theta(\mathrm{x}) = \begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}\mathrm{x}_0\\ \mathrm{x}_1\\ \mathrm{x}_2\end{bmatrix} = \begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}1\\ \mathrm{x}\\ \sqrt{\mathrm{x}}\end{bmatrix}$$

x_1: size of house

x_2: $\sqrt{\text{size of house}}$

Note: if you choose your features this way, then feature scaling becomes very important

A bit more on Linear Regression

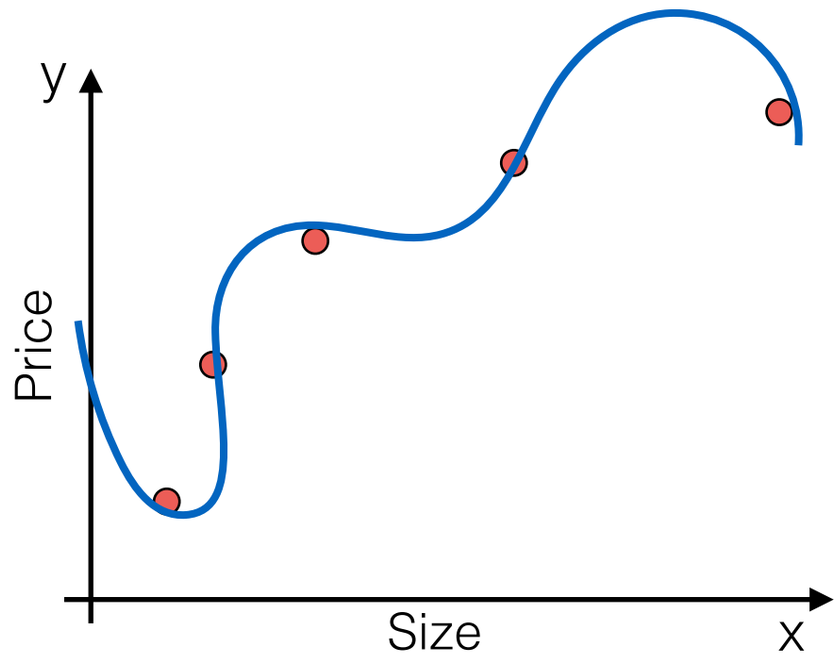


$h_\theta(\mathrm{x}) = \theta_0 + \theta_1\,\mathrm{x}$ $\quad$ $h_\theta(\mathrm{x}) = \theta_0 + \theta_1\,\mathrm{x} + \theta_2\,\mathrm{x}^2$ $\quad$ $\theta_0+\theta_1\mathrm{x}+\theta_2\mathrm{x}^2+\theta_3\mathrm{x}^3+\theta_4\mathrm{x}^4$

- **Overfitting**: if we have too many features, the learned $h_\theta(\mathrm{x})$ may fit the training set very well (i.e. $\mathbf{J(\theta) \approx 0}$), but fail to generalize to new samples (predict prices on new houses)

Addressing Overfitting

- Usually we have a lot of features
  - x_1: size of house
  - x_2: no. of bedrooms
  - x_3: no. of floors
  - x_4: age of house
  - x_5: kitchen size
  - x_6: avg income in neighborhood
  - …
  - x_{100}



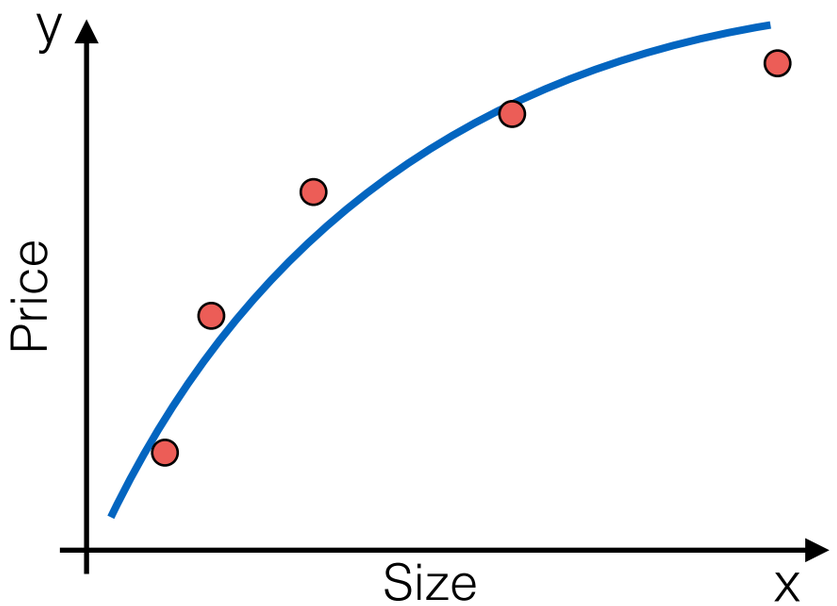
If we have many features and little training data, overfitting could be a problem

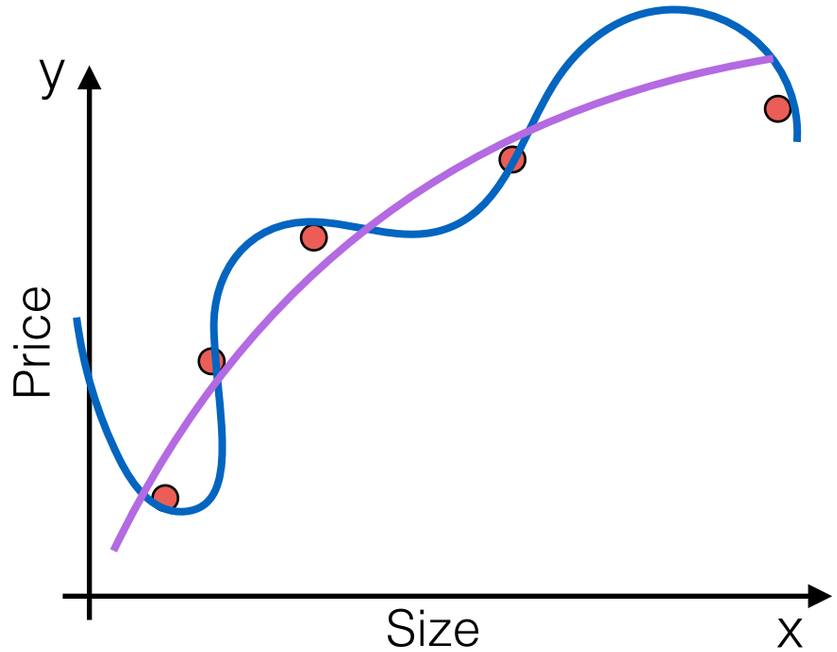
Adapted from slide by A. Ng

Addressing Overfitting

- We have two main options:
  - Reduce the number of features through **model selection** (we will talk about model selection later on)
  - **Regularization**
    - Keep all the features, but reduce the influence of parameters θ_j
    - It works well when we have many features, each of which contributes a bit to predicting y

Regularization - Intuition





$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \qquad h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Suppose we could penalize and make θ_3, θ_4 very small. *But how?*

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

Regularized Linear Regression

- Linear Regression model:

Hypothesis representation $h_\theta(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}}\,\mathbf{x}$ *Parameters* $\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix}$

Objective $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{2m}\left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$

This extra regularization term will shrink every single parameter

⇨ "Simpler" hypothesis, less prone to overfitting

Regularized Linear Regression

- We can learn our parameters with gradient descent

$$\min_{\boldsymbol{\theta}} \mathrm{J}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{2\mathrm{m}} \left[\sum_{\mathrm{i}=1}^{\mathrm{m}} (h_\theta(\mathrm{x}^{(\mathrm{i})}) - \mathrm{y}^{(\mathrm{i})})^2 + \lambda \sum_{\mathrm{j}=1}^{\mathrm{n}} \theta_{\mathrm{j}}^2 \right]$$

repeat until convergence{

$$\theta_{\mathrm{j}} := \theta_{\mathrm{j}} - \eta \frac{\partial}{\partial \theta_{\mathrm{j}}} \mathrm{J}(\boldsymbol{\theta}) = \theta_{\mathrm{j}} - \frac{\eta}{\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}} (h_\theta(\mathrm{x}^{(\mathrm{i})}) - \mathrm{y}^{(\mathrm{i})}) \, \mathrm{x}_{\mathrm{j}}^{(\mathrm{i})}$$

(simultaneously update all θ_{j})

}

Regularized Linear Regression

- We can learn our parameters with gradient descent

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{2m}\left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

repeat until convergence{

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

(simultaneously update all θ_j)

$$\theta_j := \theta_j - \eta \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\, x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

$(j = 1, \ldots, n)$

}

Here there's something pretty interesting

Regularized Linear Regression

- We can learn our parameters with gradient descent

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{2m}\left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

repeat until convergence{

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

(simultaneously update all θ_j)

$$\theta_j := \theta_j \boxed{(1 - \eta \frac{\lambda}{m})} - \boxed{\eta \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\, x_j^{(i)}}$$

$(j = 1, \ldots, n)$

}

This is going to be <1 (≈0.99)

This is exactly the same term we had before introducing regularization

Contact

- **Office:** Torre Archimede, room 6CD3
- **Office hours** (ricevimento)**:** Friday 9:00-11:00

✉ lamberto.ballan@unipd.it

🏠 http://www.lambertoballan.net

🏠 http://vimp.math.unipd.it

@ twitter.com/lambertoballan