Linear Regression

Housing Dataset

Sales of individual residential property in Ames, Iowa from 2006 to 2010

- **Instances:** 1,022
- **Features:** 79 total (19 ordinal, 25 nominal, 35 numeric)
- **Target Variable**: Sales price

MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	...	MoSold	YrSold	SaleType	SaleCondition	SalePrice
20	RL	80.0	10400	Pave	NaN	Reg	...	5	2008	WD	Normal	174000
180	RM	35.0	3675	Pave	NaN	Reg	...	5	2006	WD	Normal	145000
60	FV	72.0	8640	Pave	NaN	Reg	...	6	2010	Con	Normal	215200
20	RL	84.0	11670	Pave	NaN	IR1	...	3	2007	WD	Normal	320000
60	RL	43.0	10667	Pave	NaN	IR2	...	4	2009	ConLw	Normal	212000
80	RL	82.0	9020	Pave	NaN	Reg	...	6	2008	WD	Normal	168500
60	RL	70.0	11218	Pave	NaN	Reg	...	5	2010	WD	Normal	189000
80	RL	85.0	13825	Pave	NaN	Reg	...	12	2008	WD	Normal	140000
60	RL	NaN	13031	Pave	NaN	IR2	...	7	2006	WD	Normal	187500

Data from: De Cock. Journal of Statistics Education 19(3), 2011

Dataframe.describe()

- As with all datasets, the first thing we do is examine it…

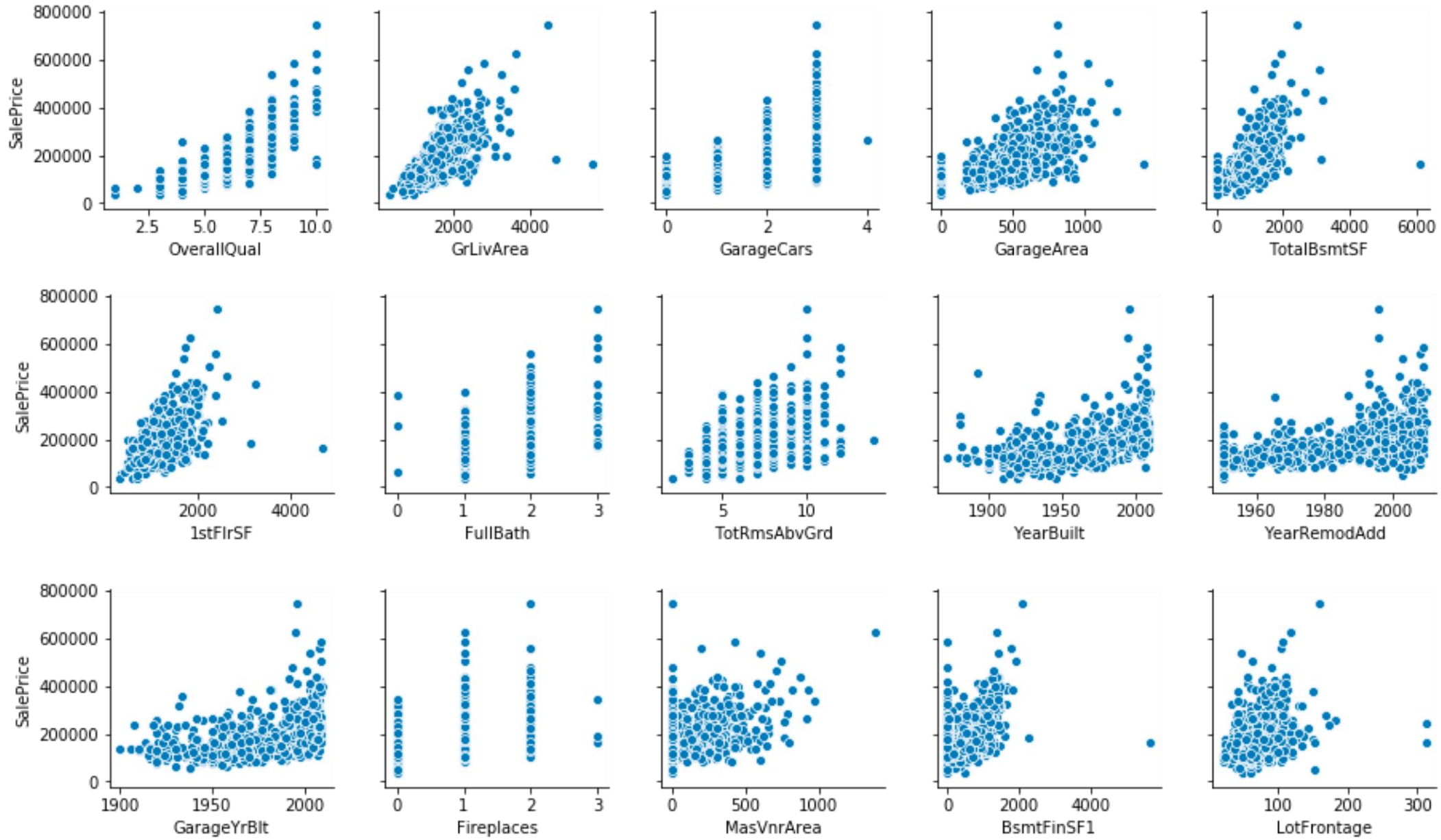
Note the missing values

No missing target values

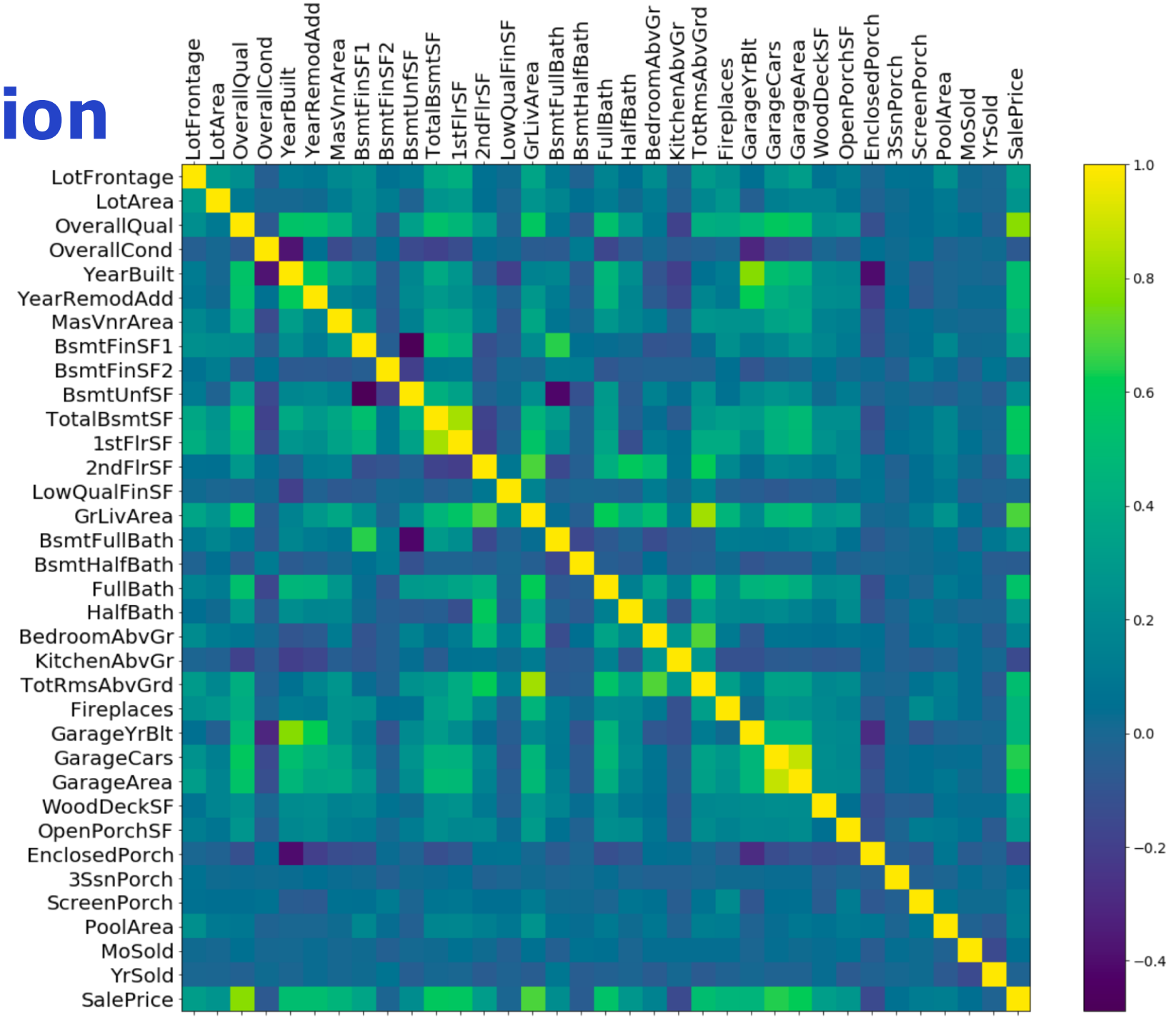
	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	...	SalePrice
count	1022.000000	1022.000000	832.000000	1022.000000	1022.000000	1022.000000	1022.000000	1022.000000	1019.000000		1022.000000
mean	732.338552	57.059687	70.375000	10745.437378	6.128180	5.564579	1970.995108	1984.757339	105.261040		181312.692759
std	425.860402	42.669715	25.533607	11329.753423	1.371391	1.110557	30.748816	20.747109	172.707705		77617.461005
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000		34900.000000
25%	367.500000	20.000000	59.000000	7564.250000	5.000000	5.000000	1953.000000	1966.000000	0.000000		130000.000000
50%	735.500000	50.000000	70.000000	9600.000000	6.000000	5.000000	1972.000000	1994.000000	0.000000		165000.000000
75%	1100.500000	70.000000	80.000000	11692.500000	7.000000	6.000000	2001.000000	2004.000000	170.000000		215000.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1378.000000		745000.000000

Potential outliers

Plot Features Most Correlated with Target Variable



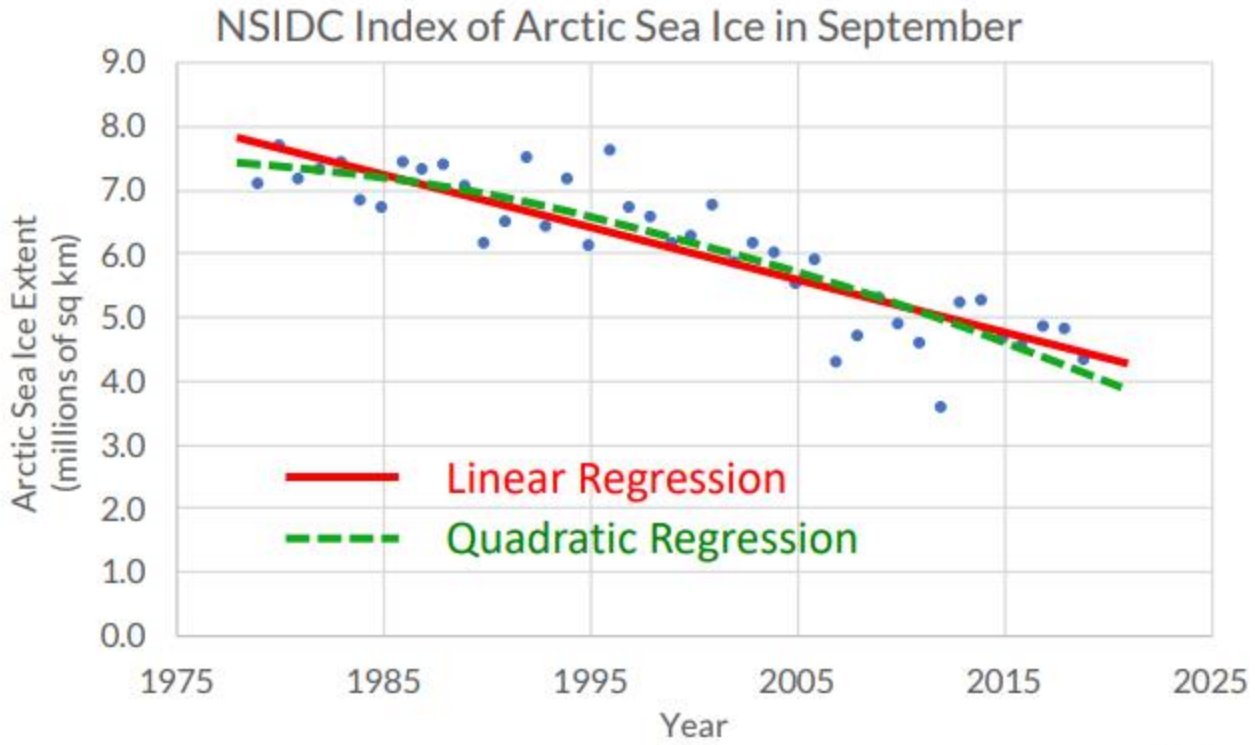
Feature Correlation Matrix



Regression

Given:

- Data $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ where $\boldsymbol{x}_i \in \mathbb{R}^d$
- Corresponding labels $\boldsymbol{y} = \{y_1, \ldots, y_n\}$ where $y_i \in \mathbb{R}$



Data from https://nsidc.org/arcticseaicenews/sea-ice-tools/

Ordinary Least Squares

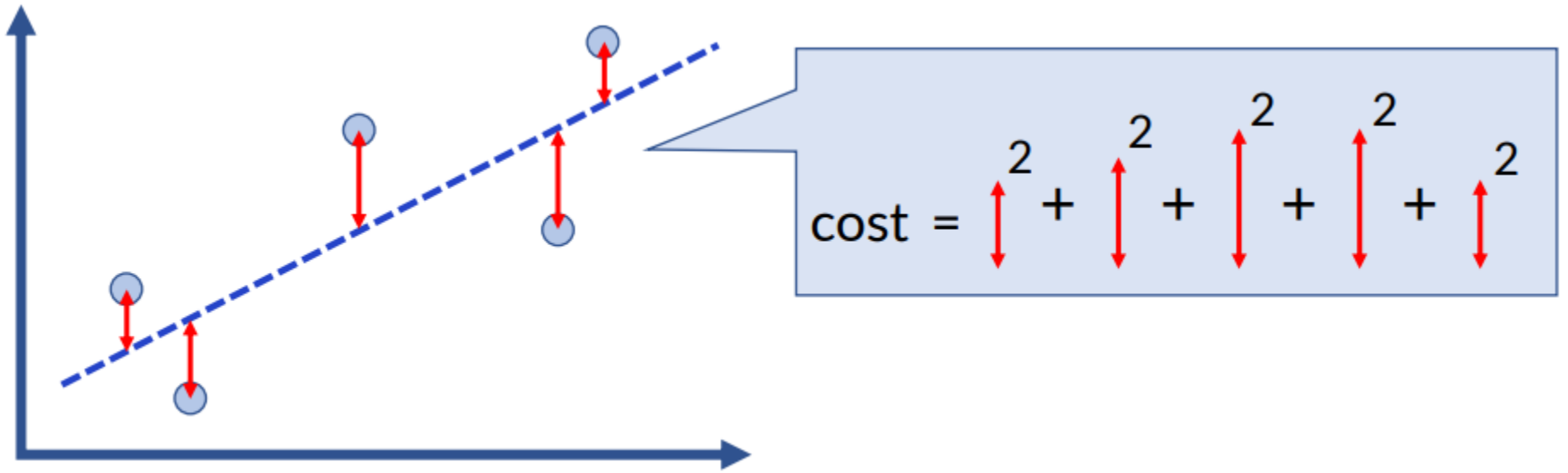
Least Squares Linear Regression

- Hypothesis:

$$y = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \ldots + \theta_d x_{id} = \sum_{j=0}^{d} \theta_j x_{ij}$$

Assume $x_{i0} = 1$

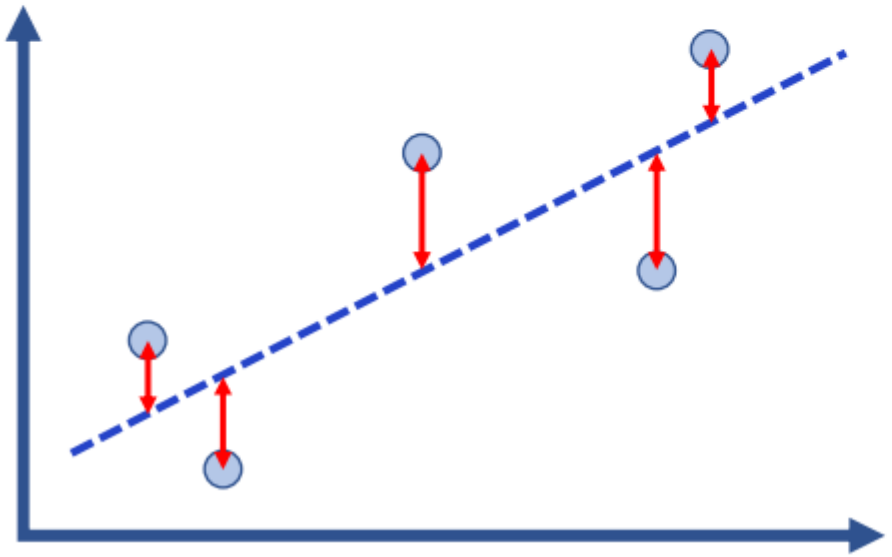
- Fit model by minimizing sum of squared errors

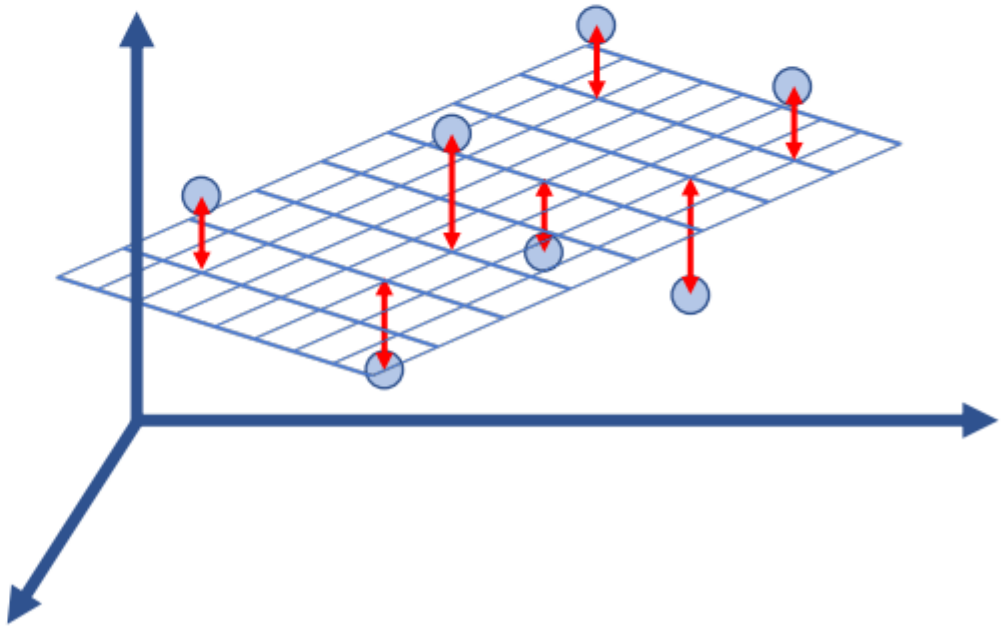


Least Squares Linear Regression

Cost Function $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2$

Fit by solving $\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$





Intuition Behind the Cost Function

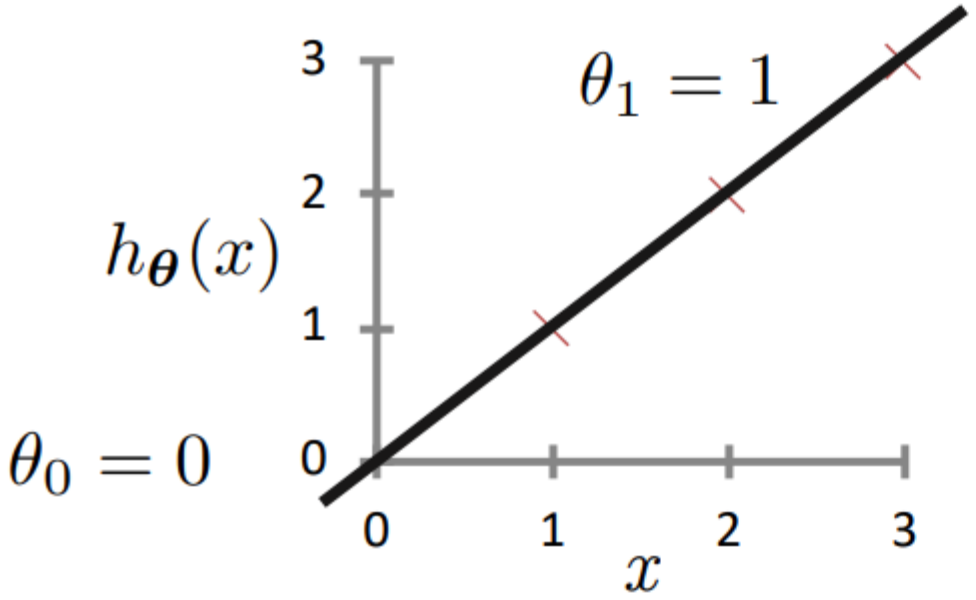
Cost function: $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2$

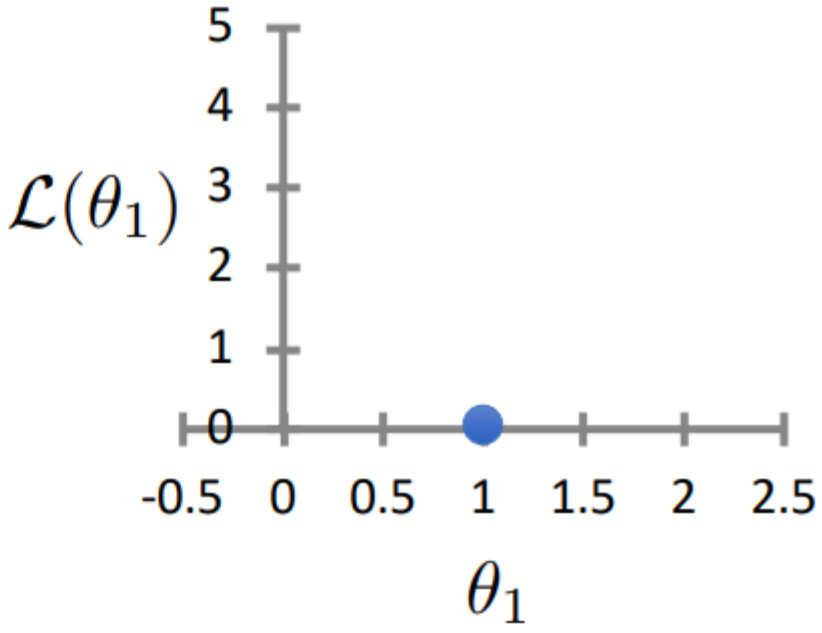
For insight on $\mathcal{L}(\boldsymbol{\theta})$, let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

Intuition Behind the Cost Function

Cost function: $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2$

For insight on $\mathcal{L}(\boldsymbol{\theta})$, let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

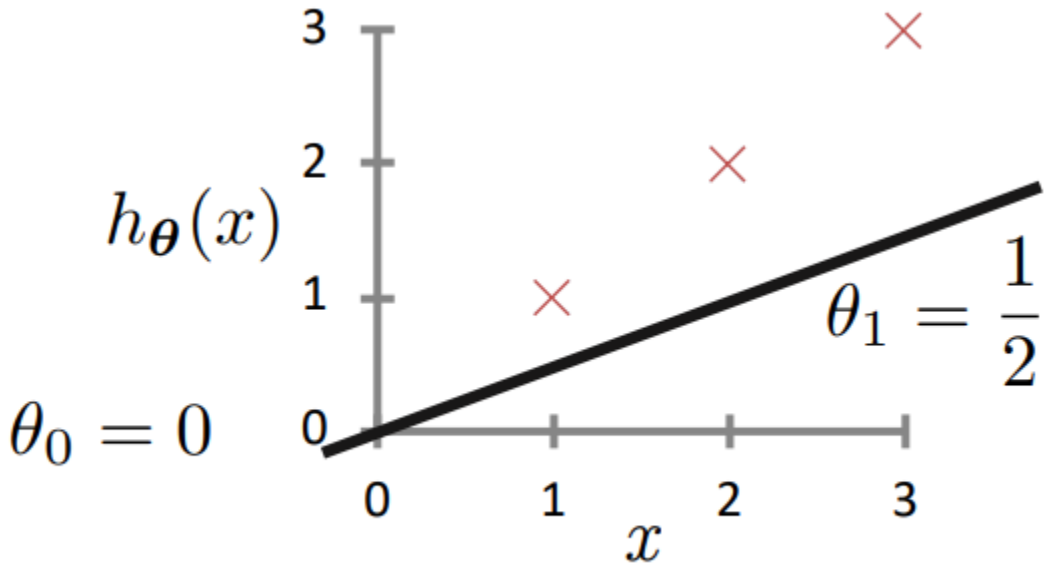




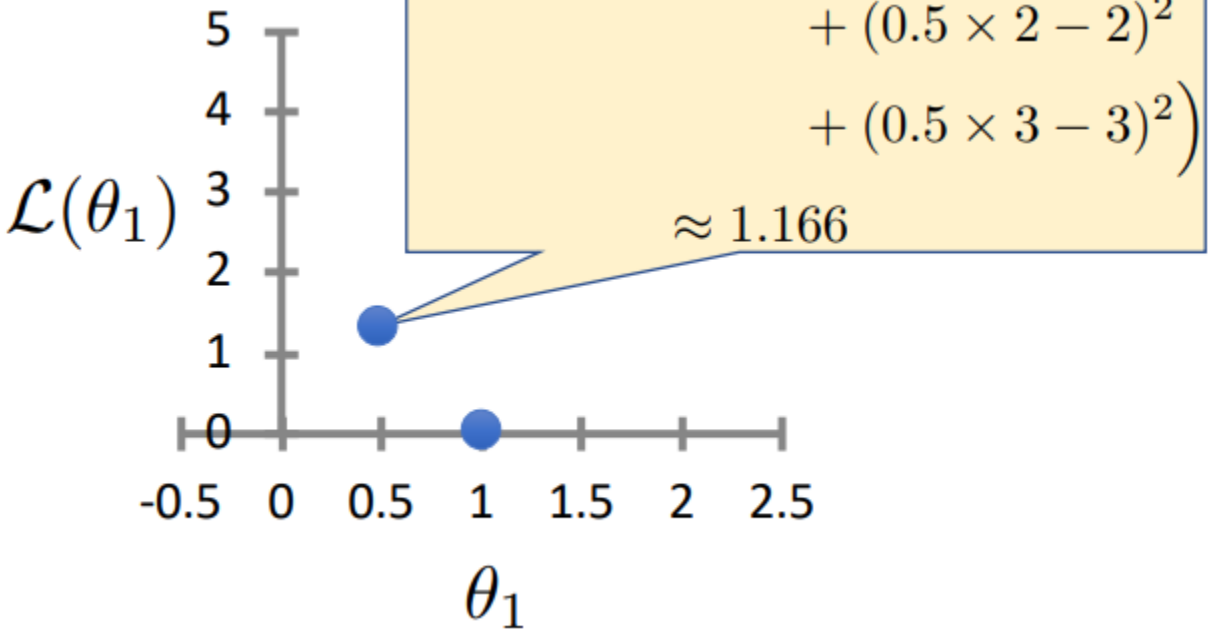
Intuition Behind the Cost Function

Cost function: $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2$

For insight on $\mathcal{L}(\boldsymbol{\theta})$, let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$



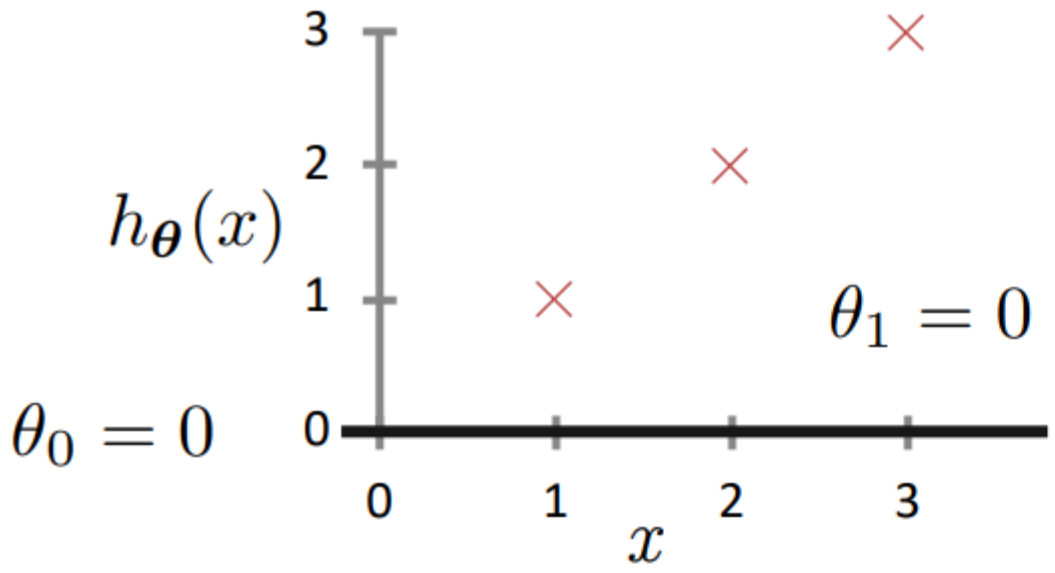
$$\mathcal{L}([0, 0.5]) = \frac{1}{3}\Big((0.5 \times 1 - 1)^2 + (0.5 \times 2 - 2)^2 + (0.5 \times 3 - 3)^2\Big) \approx 1.166$$



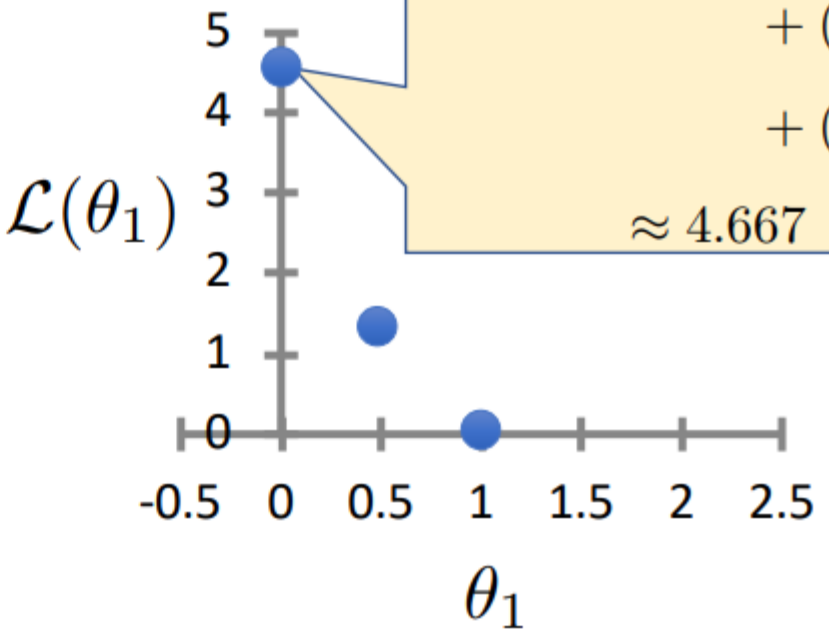
Intuition Behind the Cost Function

Cost function: $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2$

For insight on $\mathcal{L}(\boldsymbol{\theta})$, let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$



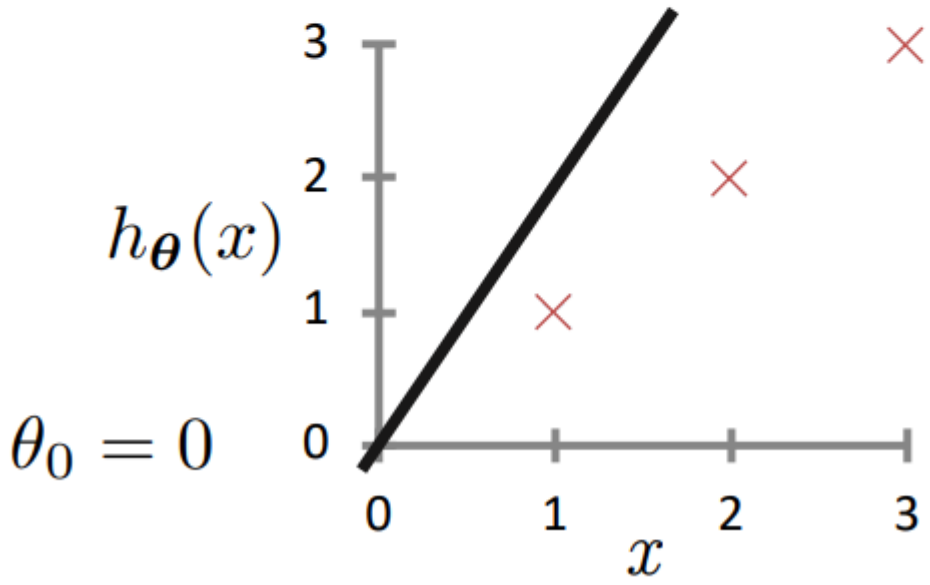
$$\mathcal{L}([0,0]) = \frac{1}{3}\Big((0\times 1 - 1)^2 + (0\times 2 - 2)^2 + (0\times 3 - 3)^2\Big) \approx 4.667$$

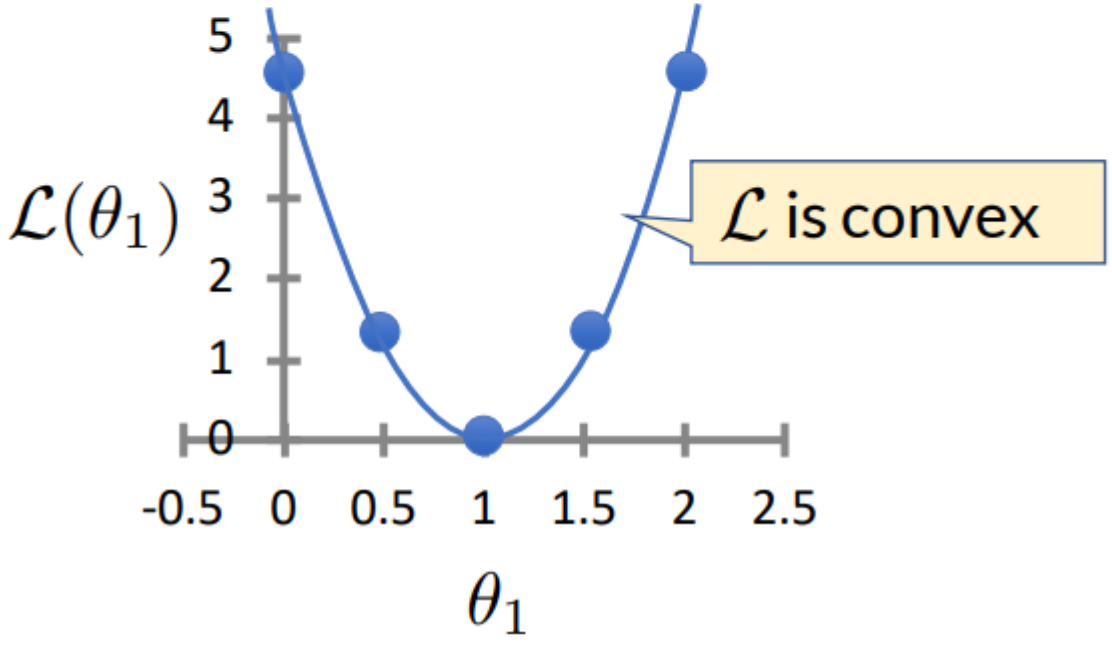


Intuition Behind the Cost Function

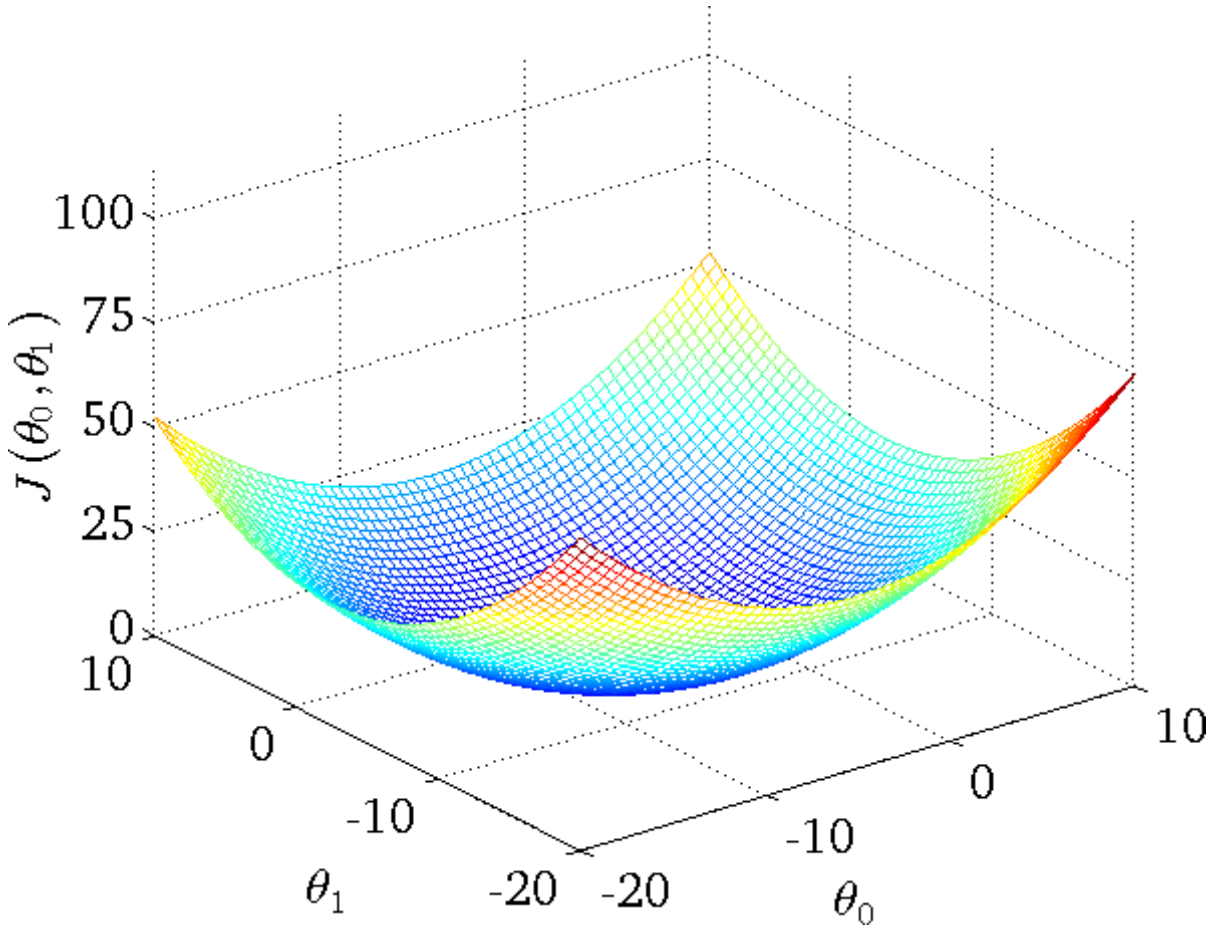
Cost function: $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2$

For insight on $\mathcal{L}(\boldsymbol{\theta})$, let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$





Intuition Behind the Cost Function

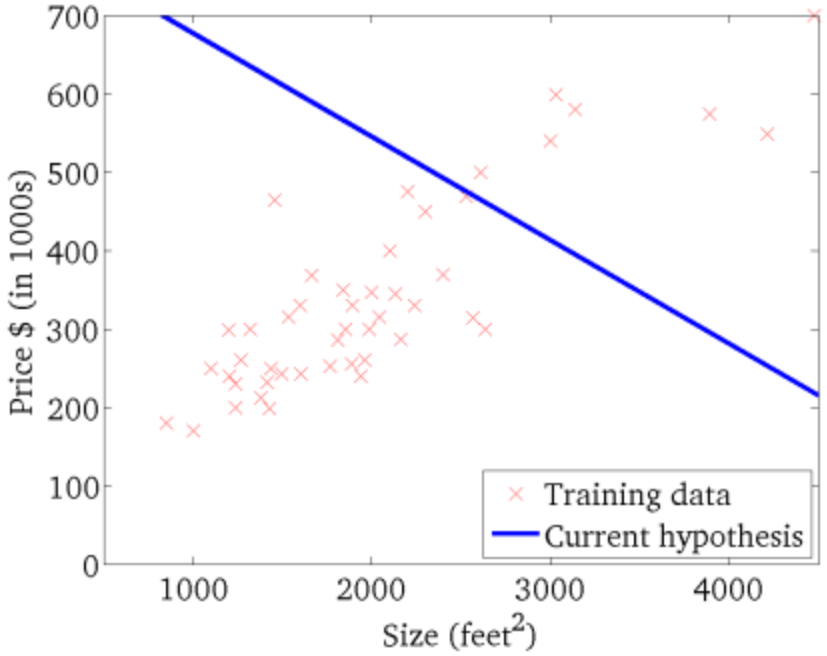


Slide by Andrew Ng

Intuition Behind the Cost Function

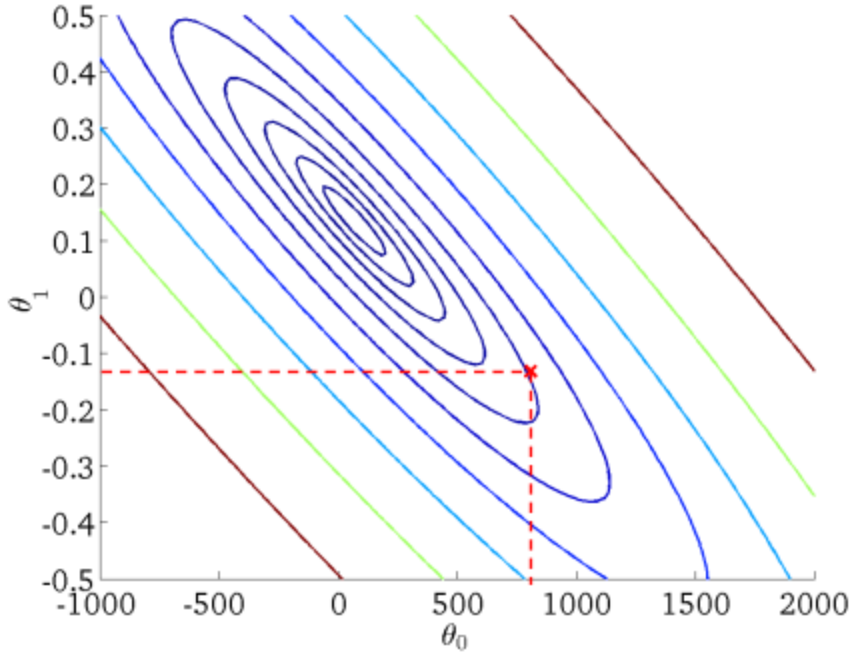
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

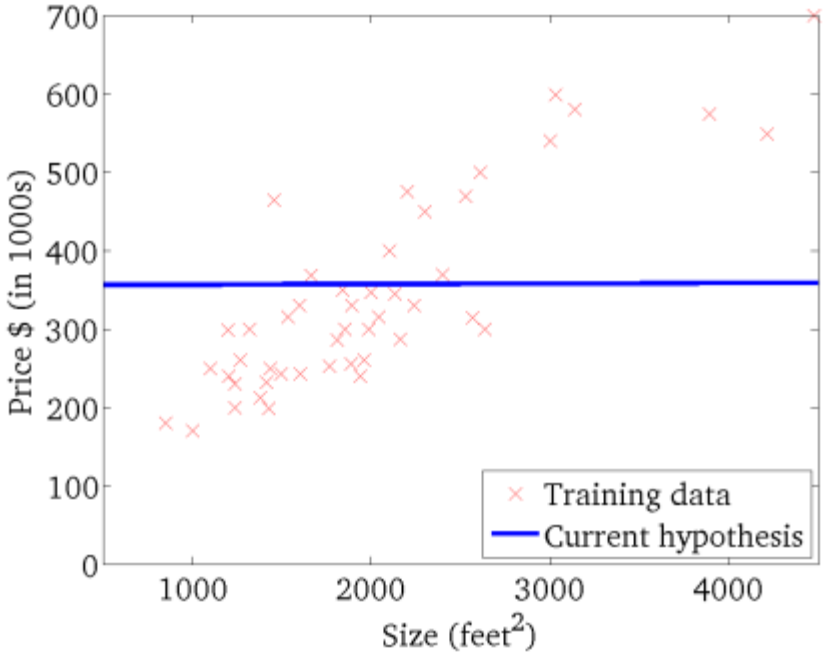
(function of the parameters θ_0, θ_1)



Intuition Behind the Cost Function

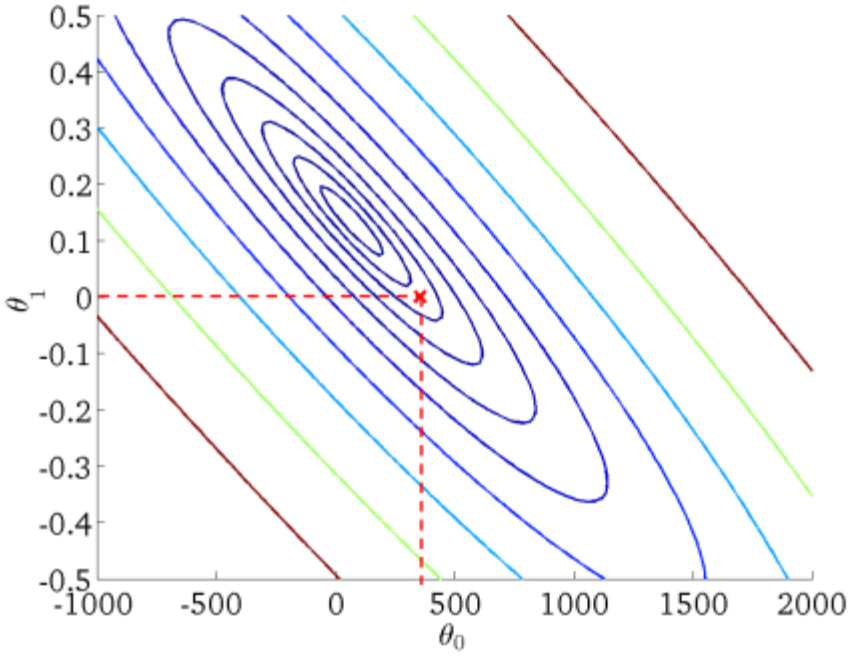
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

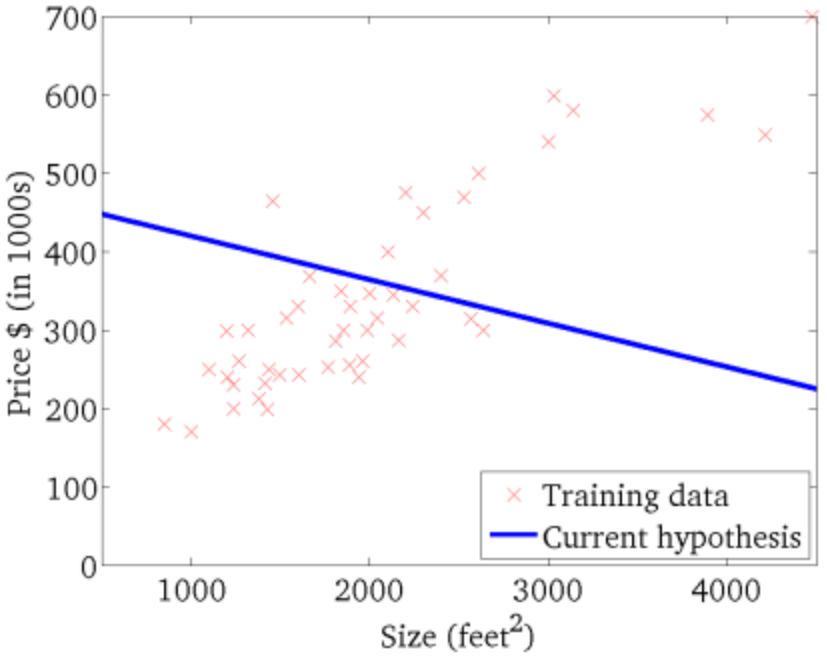
(function of the parameters θ_0, θ_1)



Intuition Behind the Cost Function

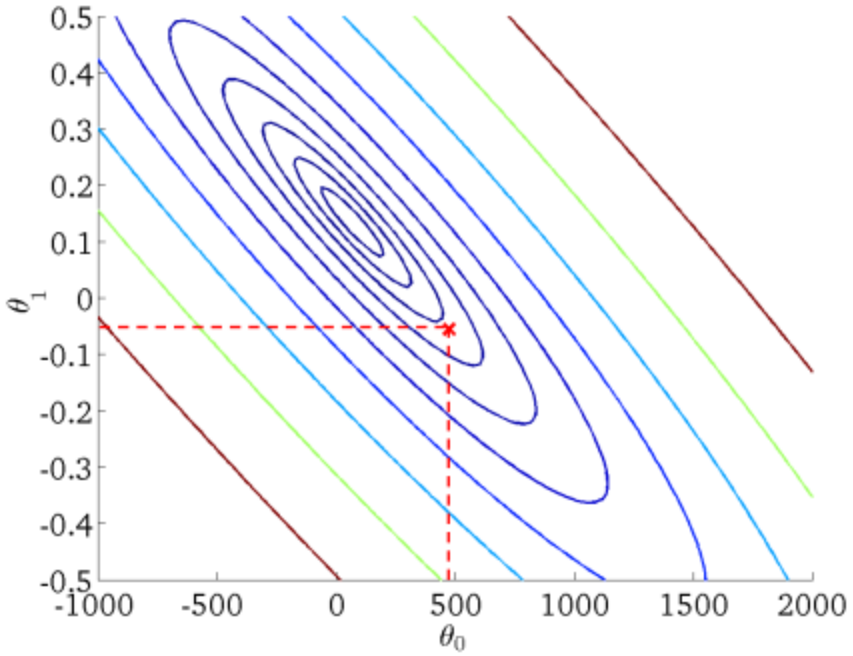
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

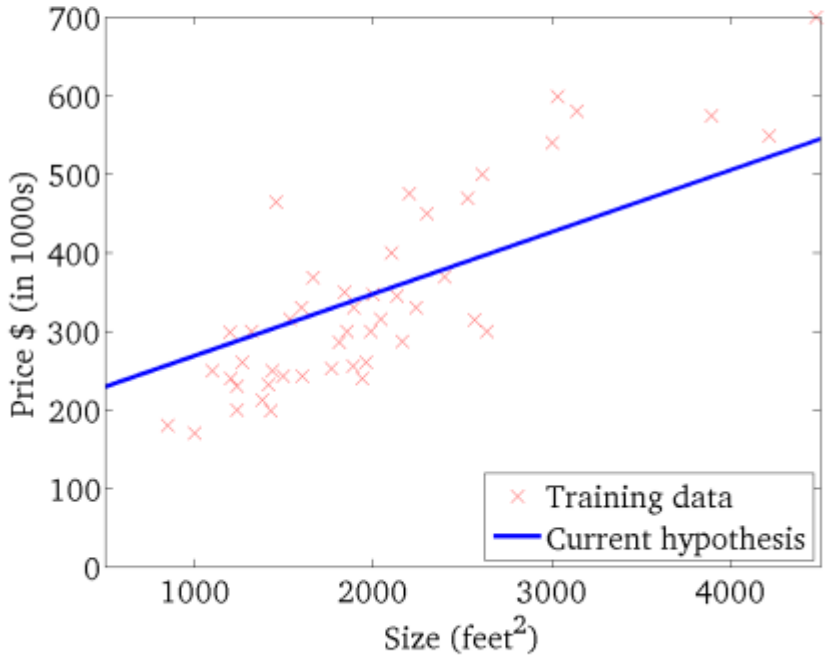
(function of the parameters θ_0, θ_1)



Intuition Behind the Cost Function

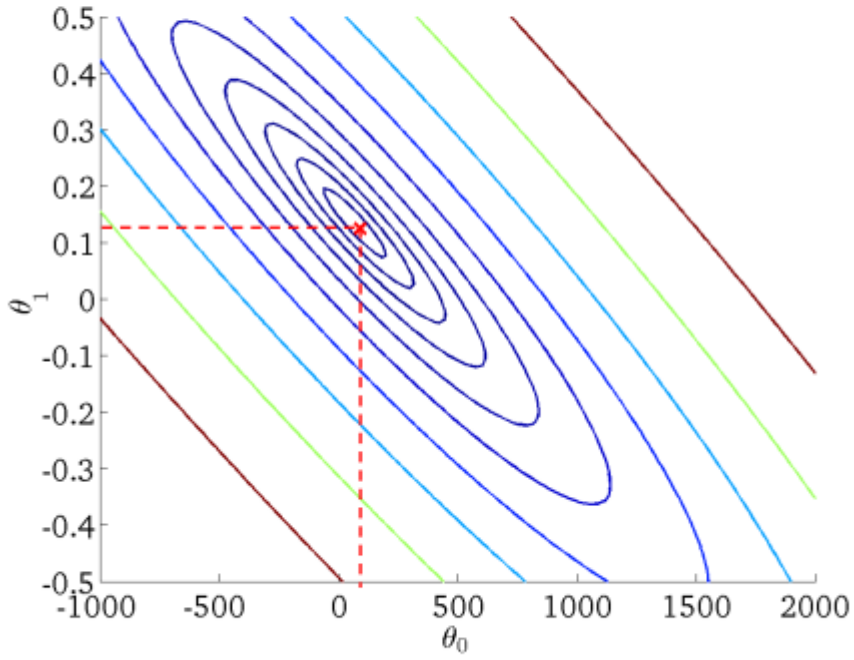
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

(function of the parameters θ_0, θ_1)

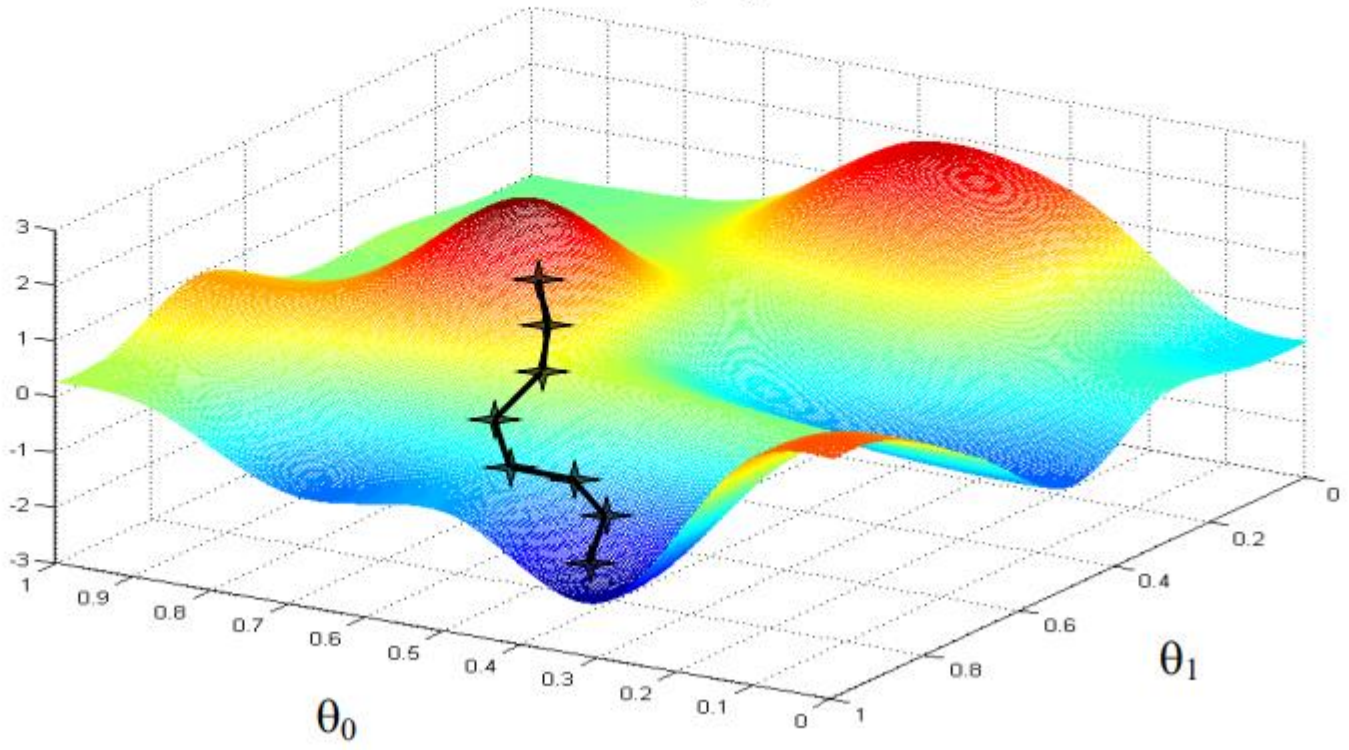


Gradient Descent: Intuition

Basic Search Procedure

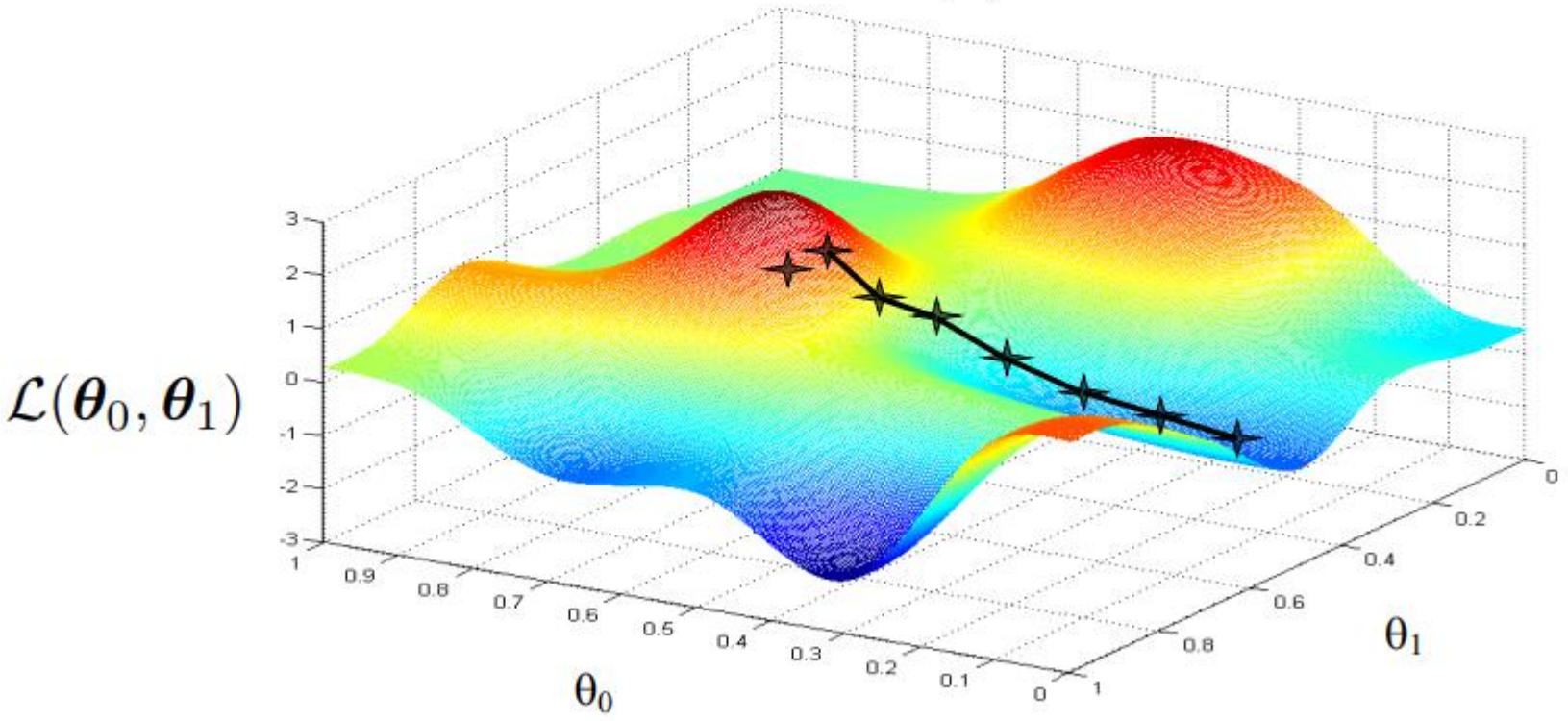
- Choose initial value for $\boldsymbol{\theta}$
- Until we reach a minimum:
  - Choose a new value for $\boldsymbol{\theta}$ to reduce $\mathcal{L}(\boldsymbol{\theta})$

$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$



Basic Search Procedure

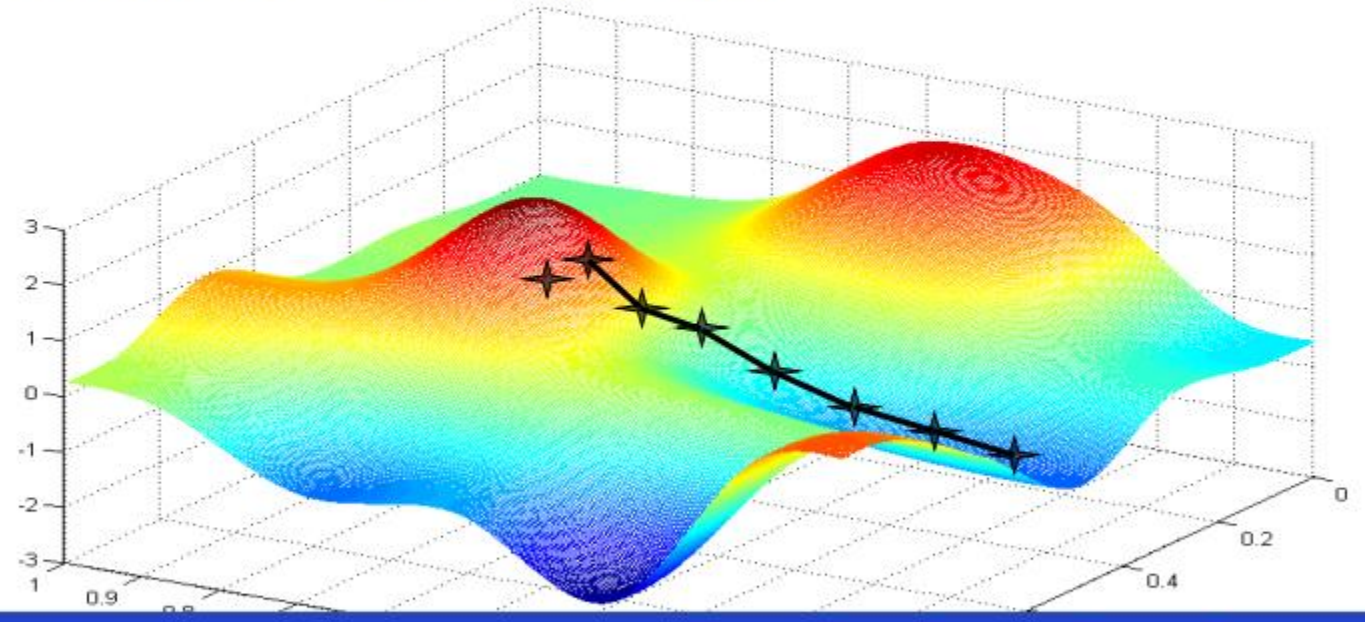
- Choose initial value for $\boldsymbol{\theta}$
- Until we reach a minimum:
  - Choose a new value for $\boldsymbol{\theta}$ to reduce $\mathcal{L}(\boldsymbol{\theta})$



Basic Search Procedure

- Choose initial value for $\boldsymbol{\theta}$
- Until we reach a minimum:
  - Choose a new value for $\boldsymbol{\theta}$ to reduce $\mathcal{L}(\boldsymbol{\theta})$

$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$



Since the least squares objective function is convex, we don't need to worry about local minima

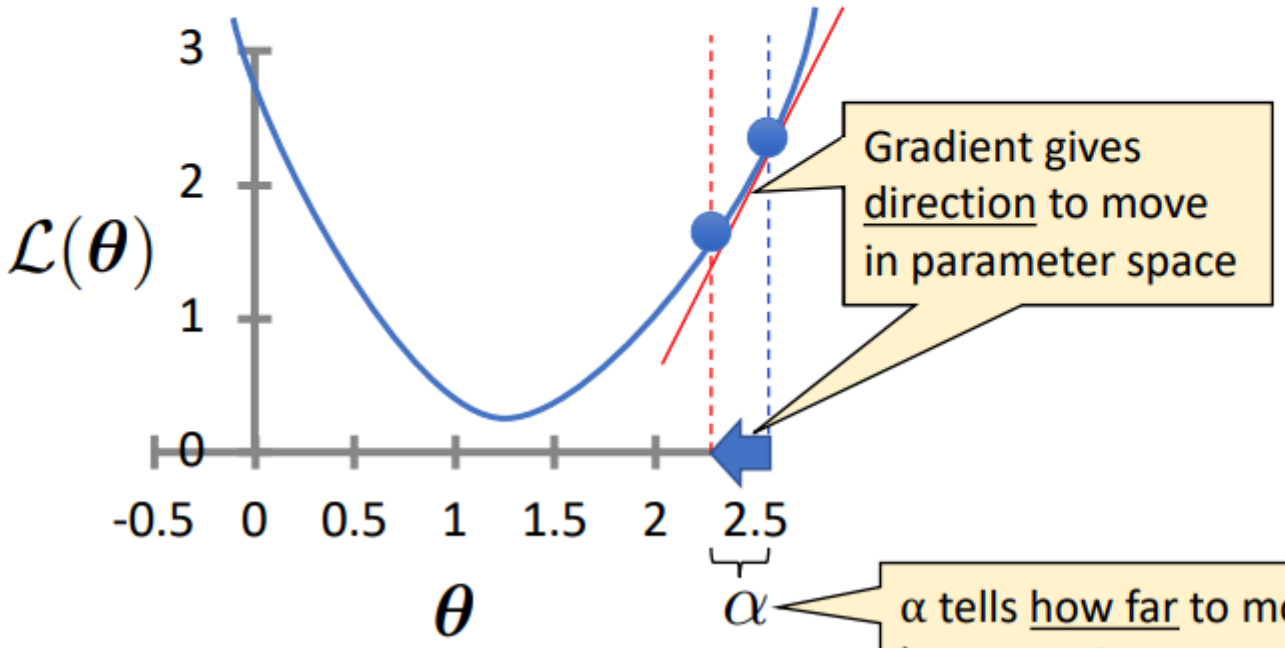
Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathcal{L}(\boldsymbol{\theta})$$

simultaneous update for $j = 0 \ldots d$

learning rate (small)
e.g., α = 0.05



Computing the Gradient for Ordinary Least Squares

Computing the Gradient for OLS

- Need to find $\frac{\partial}{\partial \theta_j} \mathcal{L}(\boldsymbol{\theta})$ for ordinary least squares:

$$\frac{\partial}{\partial \theta_j} \mathcal{L}(\boldsymbol{\theta}) = \frac{\partial}{\partial \theta_j} \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i \right) - y_i \right)^2$$

$$= \frac{\partial}{\partial \theta_j} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=0}^{d} \theta_j x_{ij} - y_i \right)^2$$

$$= \frac{2}{n} \sum_{i=1}^{n} \left(\sum_{j=0}^{d} \theta_j x_{ij} - y_i \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{j=0}^{d} \theta_j x_{ij} - y_i \right)$$

$$= \frac{2}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i \right) - y_i \right) x_{ij}$$

Gradient Descent for Ordinary Least Squares

Gradient Descent for Linear Regression

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathcal{L}(\boldsymbol{\theta})$$

simultaneous update for $j = 0 \ldots d$

Plug in $\frac{\partial}{\partial \theta_j} \mathcal{L}(\boldsymbol{\theta}) = \frac{2}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i \right) - y_i \right) x_{ij}$ from our previous step

Gradient Descent for Linear Regression

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{2}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i \right) - y_i \right) x_{ij} \quad \text{simultaneous update for } j = 0 \ldots d$$

- To achieve simultaneous update
  - At the start of each GD iteration, compute $h_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i \right)$
  - Use this stored value in the update step loop

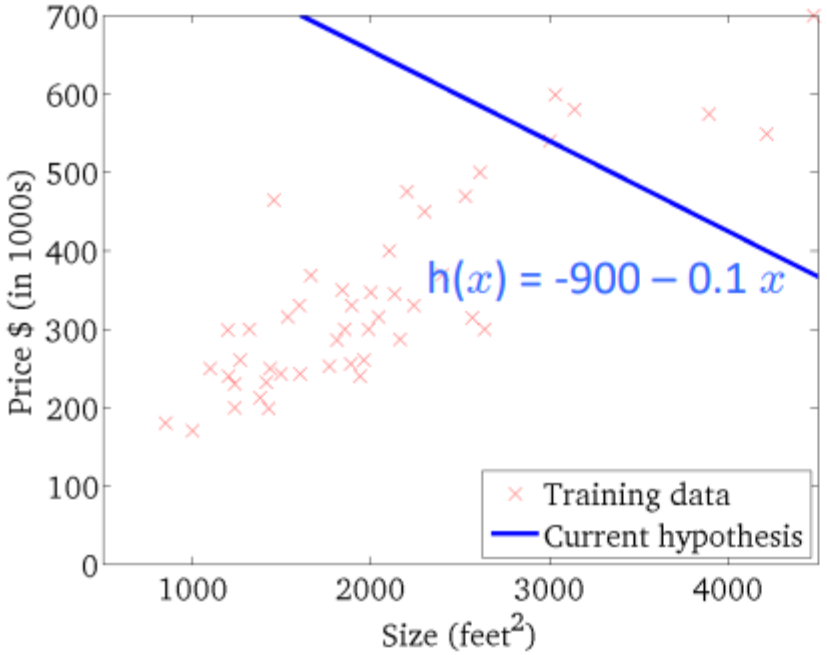
- Assume convergence when $\| \boldsymbol{\theta}_{new} - \boldsymbol{\theta}_{old} \|_2 < \epsilon$

L_2 norm: $\| \boldsymbol{v} \|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \ldots + v_{|v|}^2}$

Gradient Descent

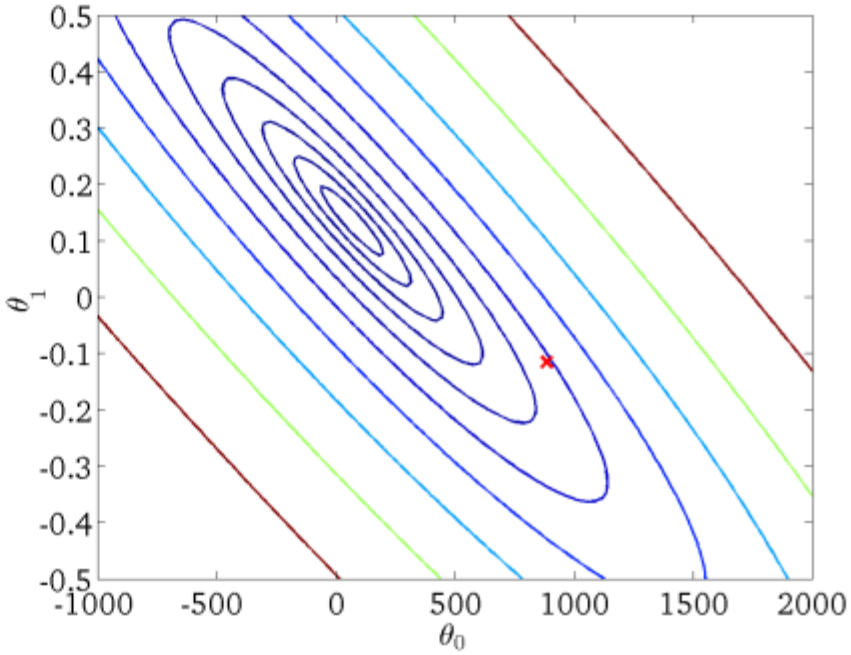
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

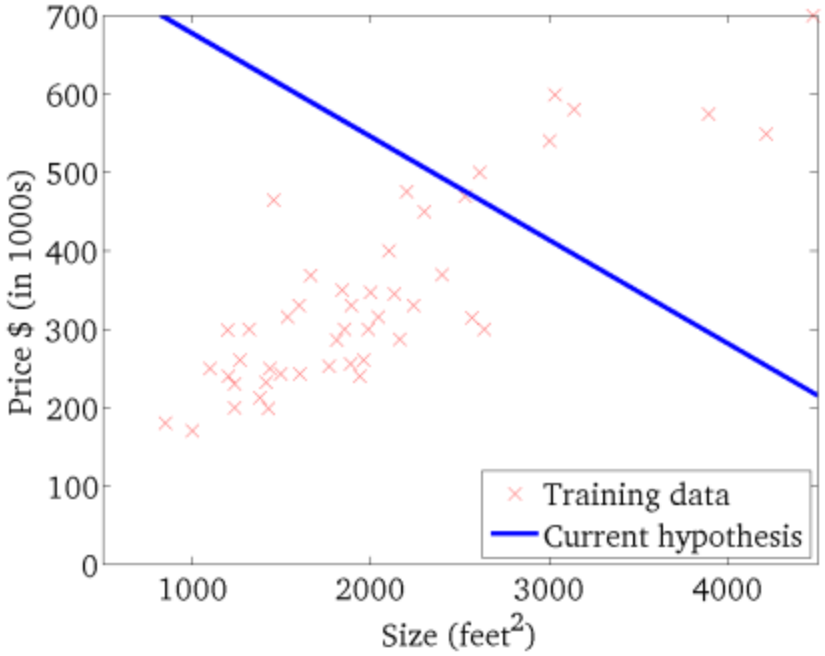
(function of the parameters θ_0, θ_1)



Gradient Descent

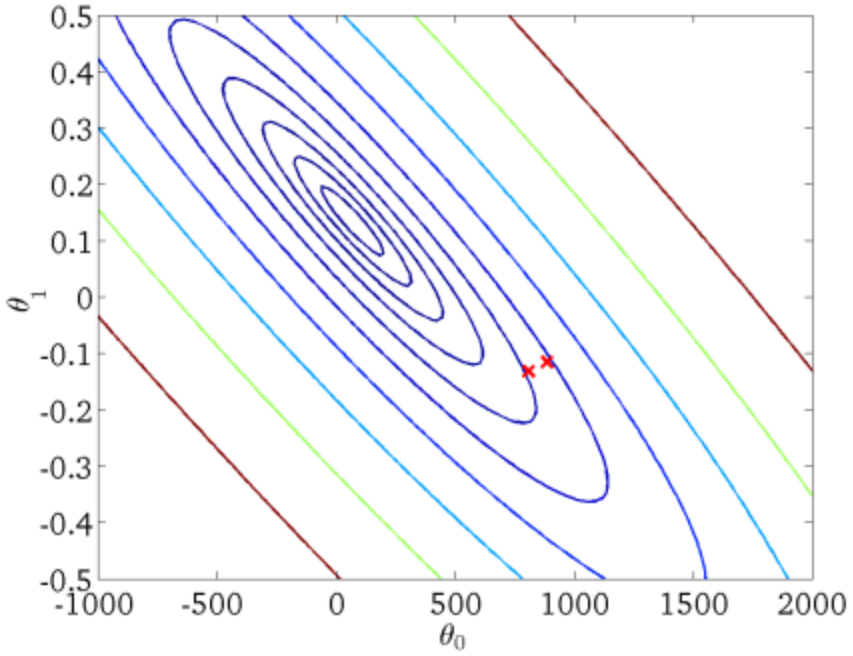
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

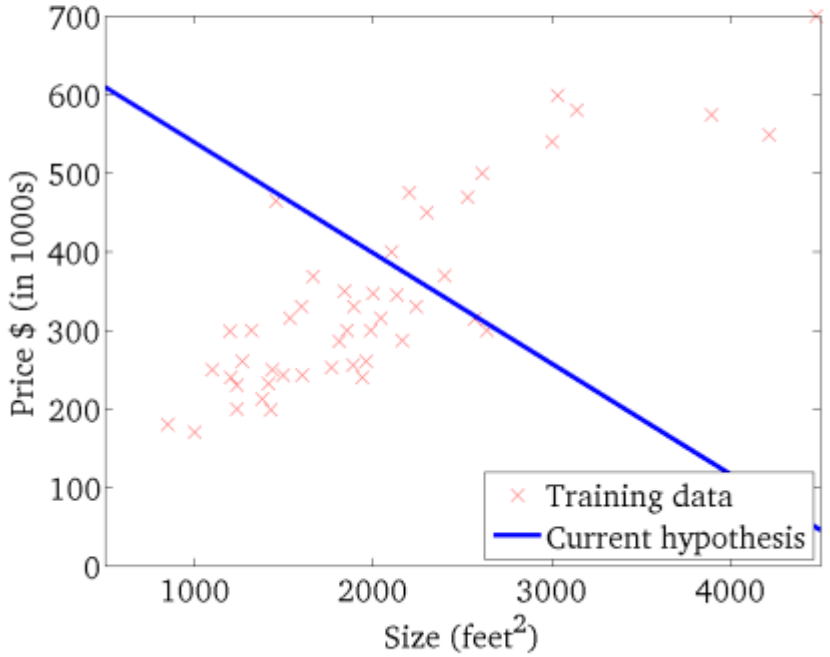
(function of the parameters θ_0, θ_1)



Gradient Descent

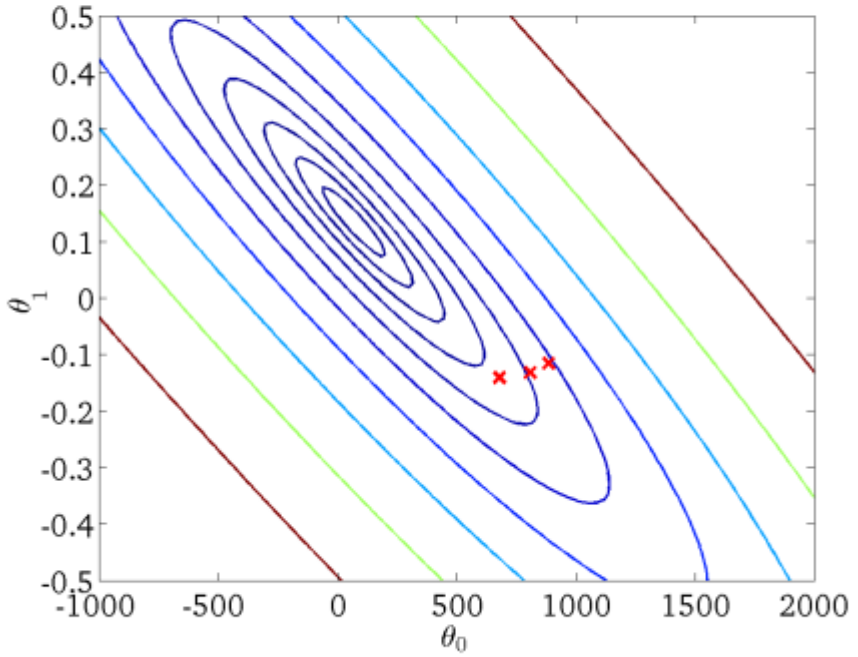
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

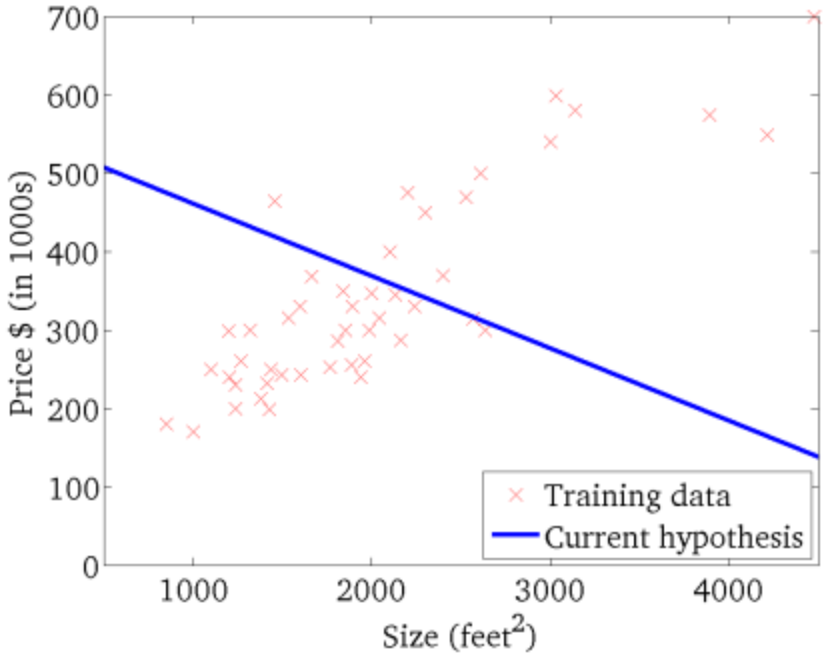
(function of the parameters θ_0, θ_1)



Gradient Descent

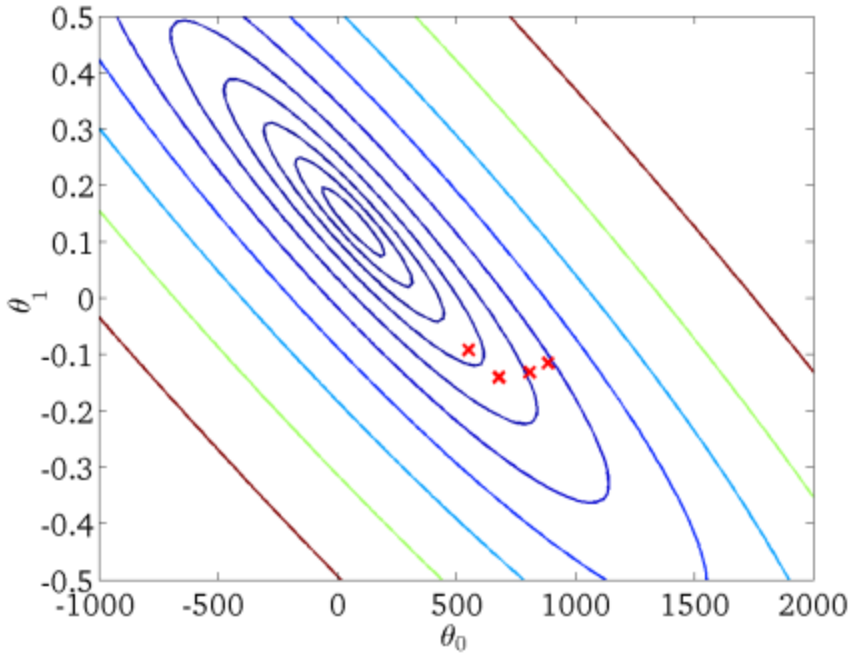
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

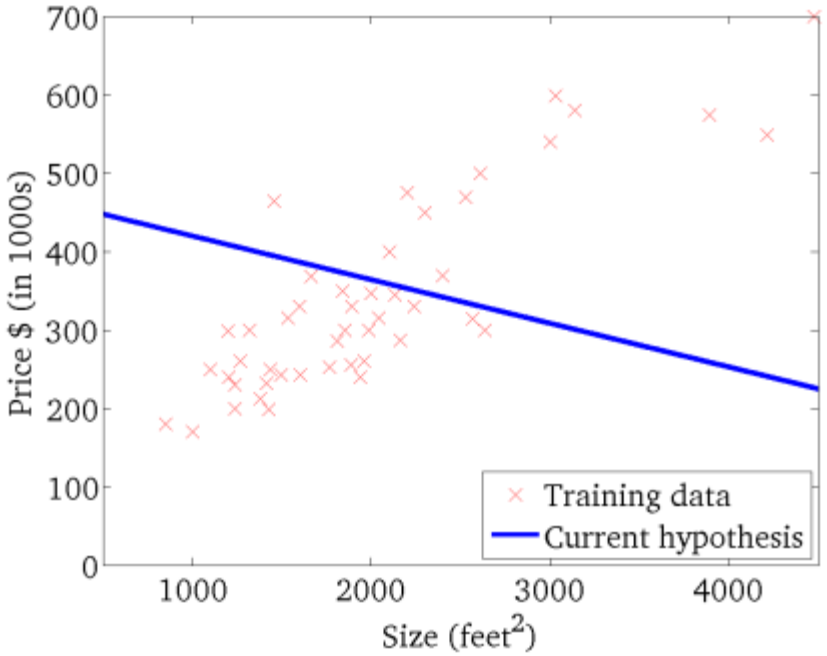
(function of the parameters θ_0, θ_1)



Gradient Descent

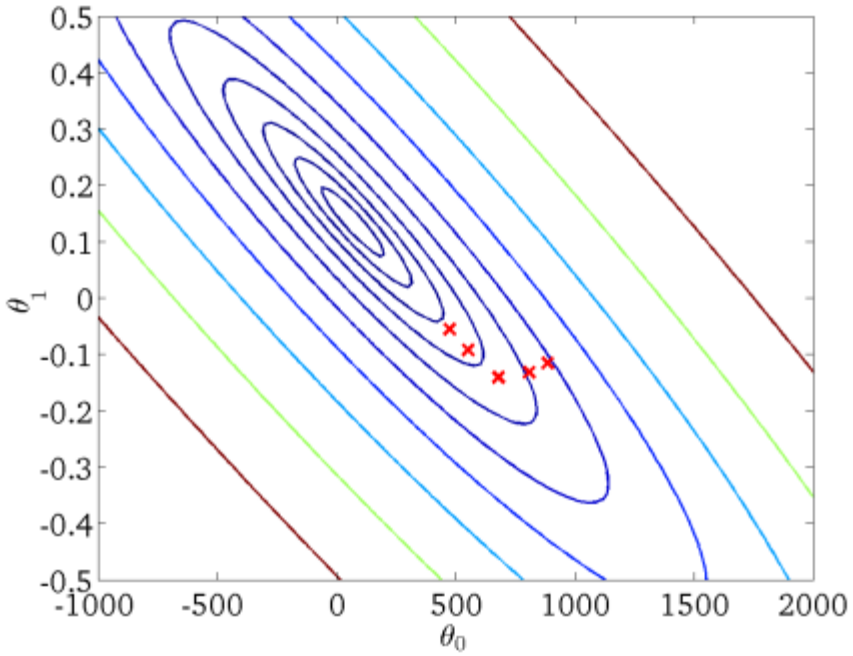
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

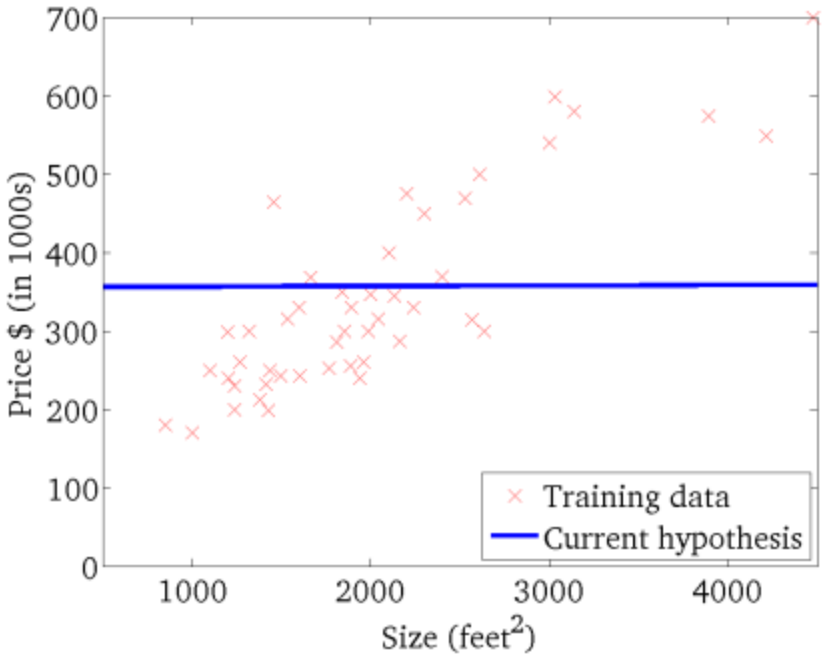
(function of the parameters θ_0, θ_1)



Gradient Descent

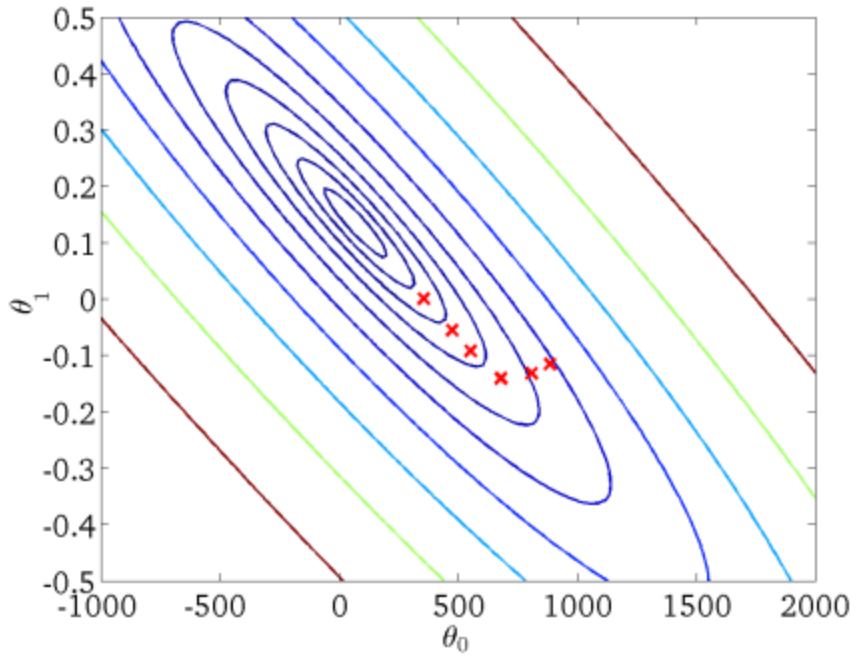
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

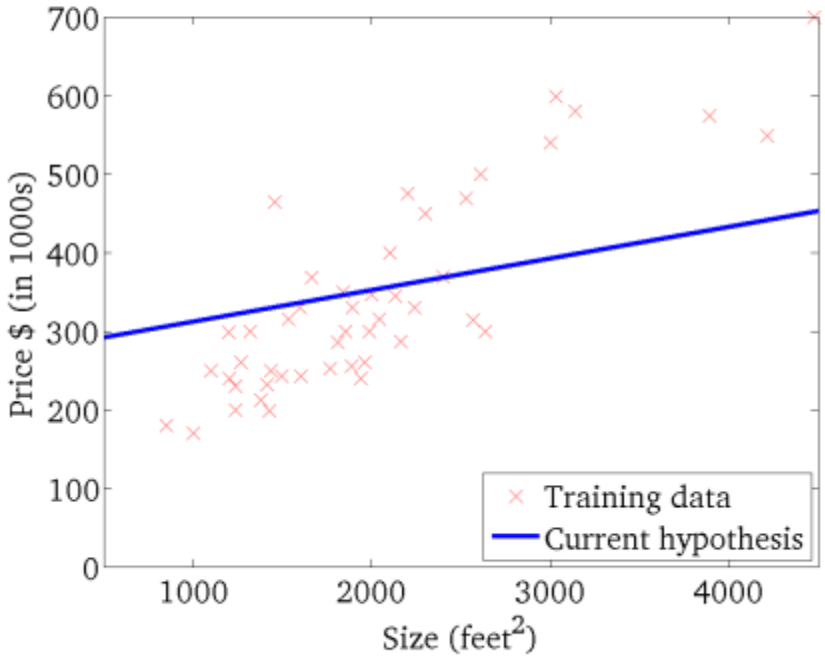
(function of the parameters θ_0, θ_1)



Gradient Descent

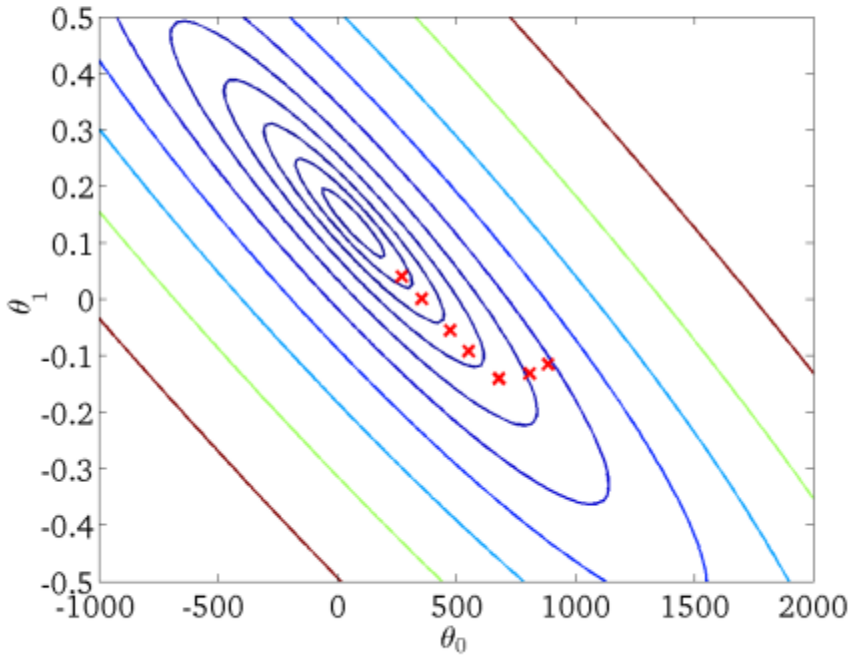
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

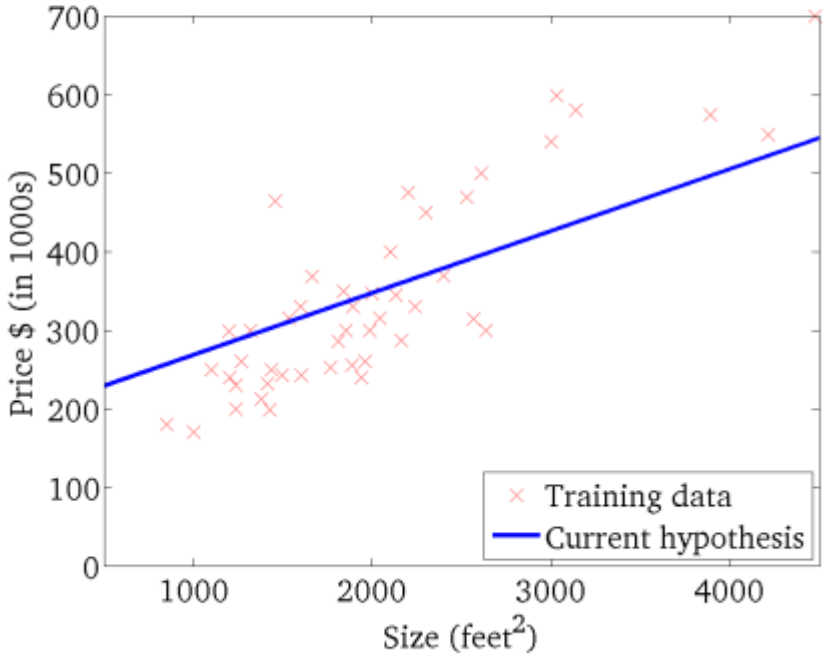
(function of the parameters θ_0, θ_1)



Gradient Descent

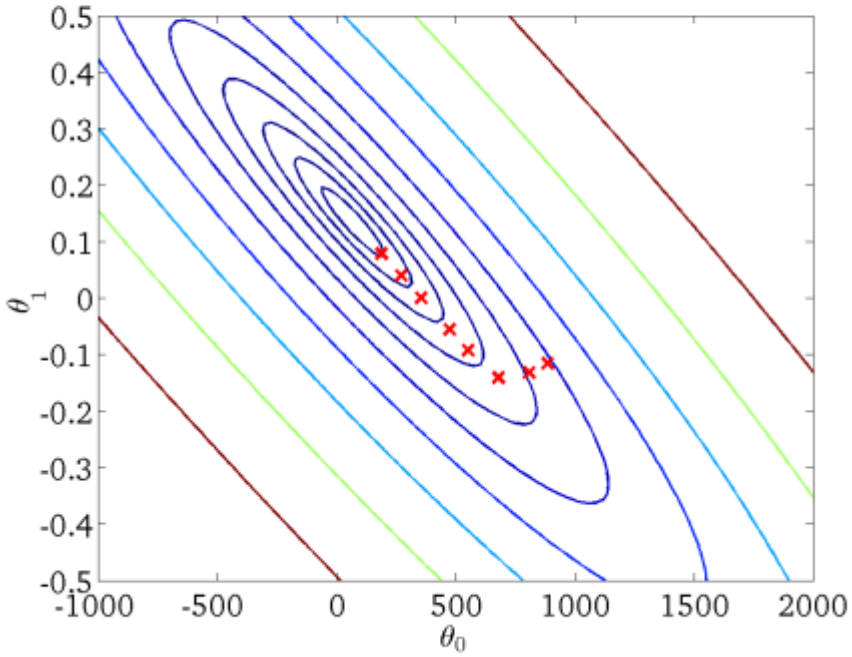
$h_{\boldsymbol{\theta}}(x)$

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



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

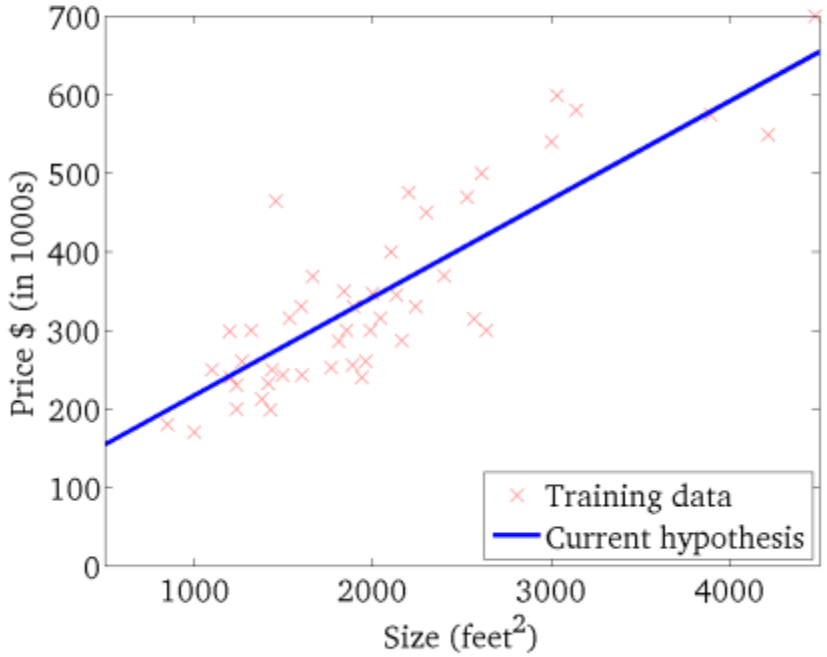
(function of the parameters θ_0, θ_1)



Gradient Descent

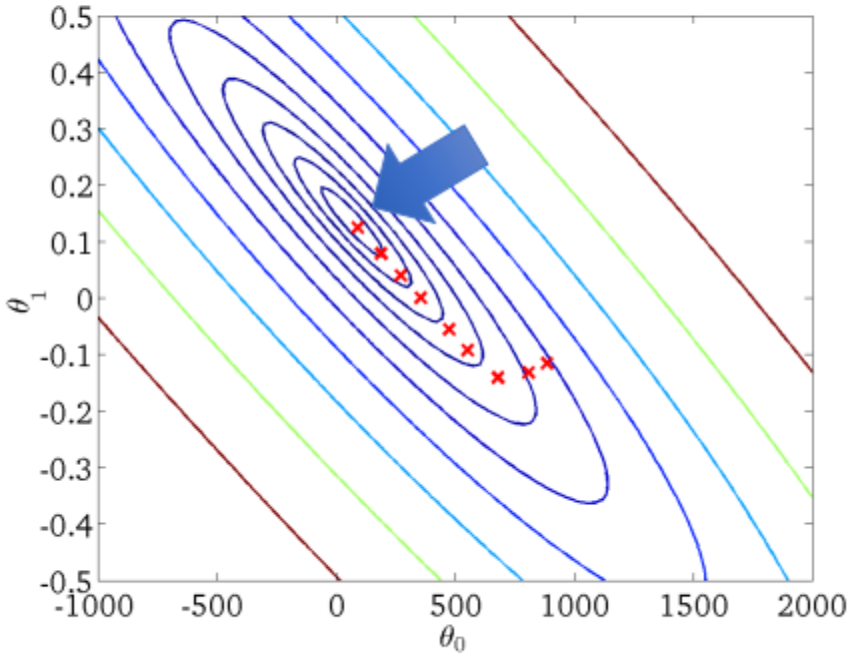
$h_{\boldsymbol{\theta}}(x)$

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



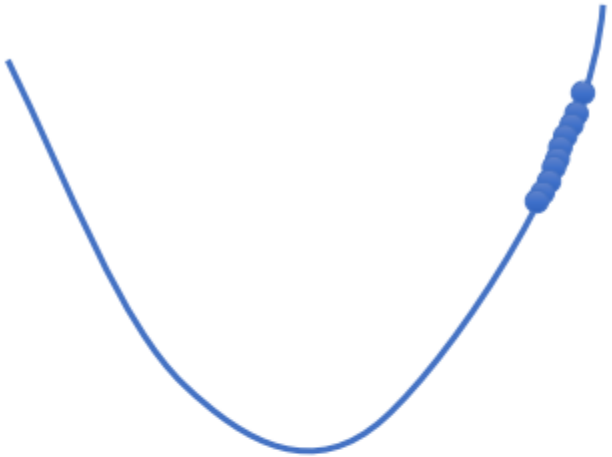
$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

(function of the parameters θ_0, θ_1)



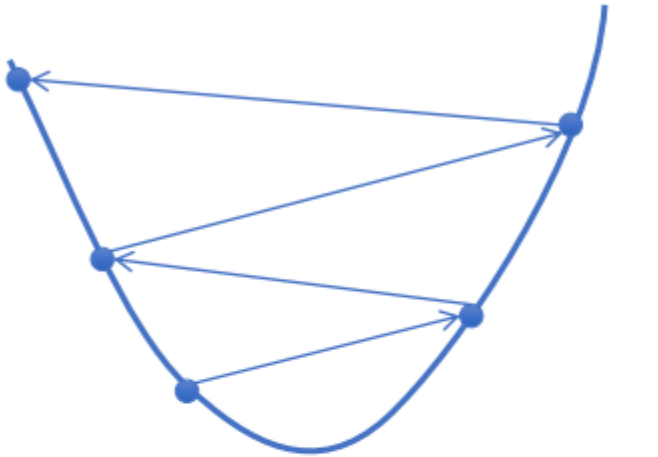
Choosing the Learning Rate (α)

slow convergence



Cause: α too small

increasing value for $\mathcal{L}(\boldsymbol{\theta})$

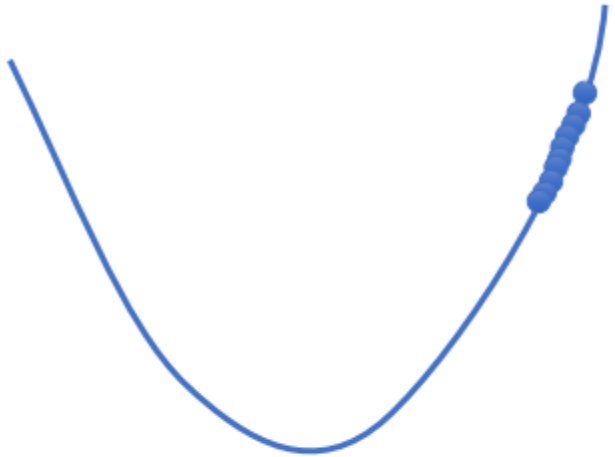


Cause: α too large

- May overshoot the minimum
- May fail to converge
- May even diverge

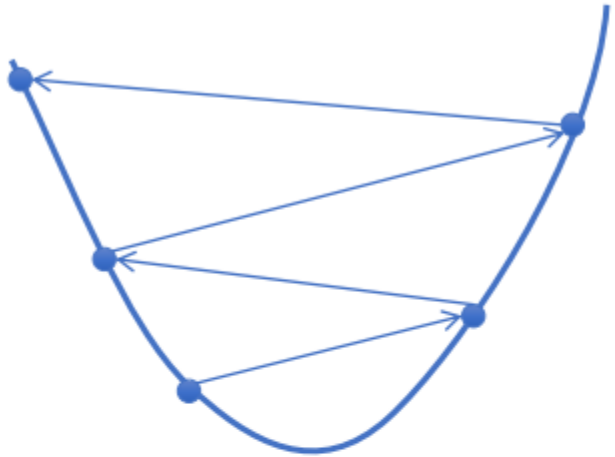
Choosing the Learning Rate (α)

slow convergence



Cause: α too small

increasing value for $\mathcal{L}(\boldsymbol{\theta})$



Cause: α too large

- To see if gradient descent is working, print out $\mathcal{L}(\boldsymbol{\theta})$ each iteration
  - The value should decrease at each iteration
  - If it doesn't, adjust α

Predicting Housing Prices

Housing Dataset

- **Instances:** 1,022
- **Features:** 79 total (19 ordinal, 25 nominal, 35 numeric)
- **Target Variable**: Sales price

MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	...	MoSold	YrSold	SaleType	SaleCondition	SalePrice
20	RL	80.0	10400	Pave	NaN	Reg	...	5	2008	WD	Normal	174000
180	RM	35.0	3675	Pave	NaN	Reg	...	5	2006	WD	Normal	145000
60	FV	72.0	8640	Pave	NaN	Reg	...	6	2010	Con	Normal	215200
20	RL	84.0	11670	Pave	NaN	IR1	...	3	2007	WD	Normal	320000
60	RL	43.0	10667	Pave	NaN	IR2	...	4	2009	ConLw	Normal	212000
80	RL	82.0	9020	Pave	NaN	Reg	...	6	2008	WD	Normal	168500
60	RL	70.0	11218	Pave	NaN	Reg	...	5	2010	WD	Normal	189000
80	RL	85.0	13825	Pave	NaN	Reg	...	12	2008	WD	Normal	140000
60	RL	NaN	13031	Pave	NaN	IR2	...	7	2006	WD	Normal	187500

Categorical features

Ordinal values

Missing values

Data from: De Cock. Journal of Statistics Education 19(3), 2011

Many Missing Values

Feature	%Missing Values
PoolQC	99.5108
MiscFeature	96.0861
Alley	93.5421
Fence	80.2348
FireplaceQu	47.6517
LotFrontage	18.5910
GarageCond	05.2838
GarageType	05.2838
GarageYrBlt	05.2838
GarageFinish	05.2838
GarageQual	05.2838
BsmtFinType1	02.5440
…	

Possible way to handle missing values

- Categorical: impute with mode
- Numerical: impute with mean
- Ordinal: impute with mode

Other Preprocessing

Encode categorical features

- Use one-hot encoding

Encode ordinal features

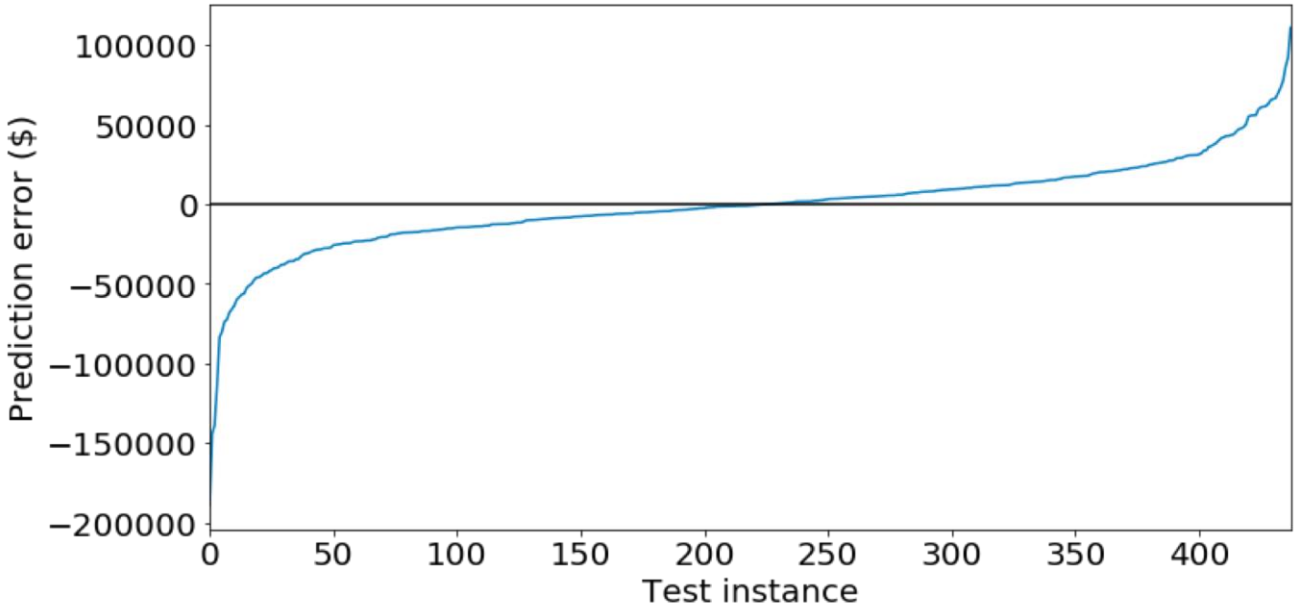
- Convert to a number, preserving the order (e.g. [low, medium, high] → [1, 2, 3])
- Encoding may not capture relative differences

HouseStyle	FullBath	RoofMatl	BsmtCond	KitchenQual
1Story	2	CompShg	TA	TA
SLvl	1	CompShg	TA	TA
2Story	2	CompShg	TA	Gd
1Story	2	CompShg	Gd	Ex
2Story	2	CompShg	TA	Gd
SLvl	1	WdShngl	TA	TA
2Story	2	CompShg	TA	Gd
SLvl	1	CompShg	TA	TA
2Story	2	CompShg	TA	TA
2Story	2	CompShg	TA	Gd

HouseStyle	FullBath	RoofMatl	BsmtCond	KitchenQual
1Story	2	CompShg	3	3
SLvl	1	CompShg	3	3
2Story	2	CompShg	3	4
1Story	2	CompShg	4	5
2Story	2	CompShg	3	4
SLvl	1	WdShngl	3	3
2Story	2	CompShg	3	4
SLvl	1	CompShg	3	3
2Story	2	CompShg	3	3
2Story	2	CompShg	3	4

Evaluation on Held-Out Test Data

- 438 test instances, preprocessed same as the training data



Not terrible, but we can do better...

Feature Transformations and Derived Features

Making Models Non-linear

Extending OLS to More Complex Models

We can alter the input $\boldsymbol{X}$ in many ways:

Feature encodings	Feature transformations	Derived features

• One-hot-encoding of categorical features
• Numeric encoding of ordinal features
	• Basic functions
	  ○ log, exp, sqrt, square, etc.
	• Polynomial functions
	  ○ $x_{ij} = \beta_0 + \beta_1 \cdot x_{ij} + \beta_2 \cdot x_{ij}^2$
	• Basis expansions
		• Combination features
		  ○ $x_{i,new} = x_{i1} \cdot x_{i2}$
		• Outputs of other ML models
		  ○ $x_{i,new}$ = tree.predict(x_{i1}, x_{i2})

This allows linear techniques to fit non-linear models

Linear Basis Function Models

- Generally,

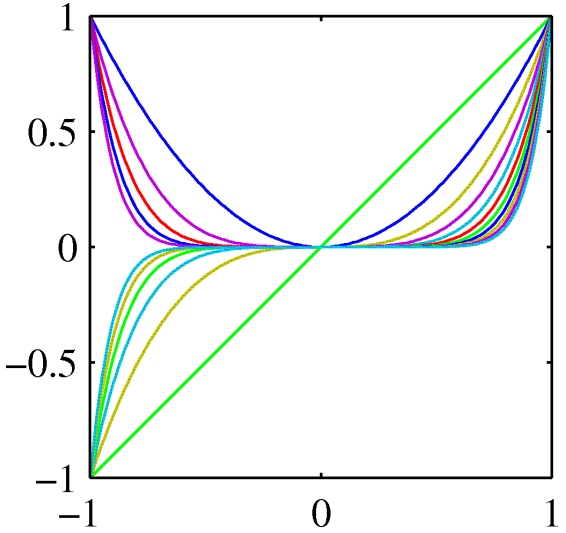
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j \underbrace{\phi_j(\boldsymbol{x})}_{\text{basis function}}$$

- Typically, $\phi_0(\boldsymbol{x}) = 1$ so that θ_0 acts as a bias
- In the simplest case, we use linear basis functions : $\phi_j(\boldsymbol{x}) = x_j$

Based on slide by Christopher Bishop (PRML)

Linear Basis Function Models

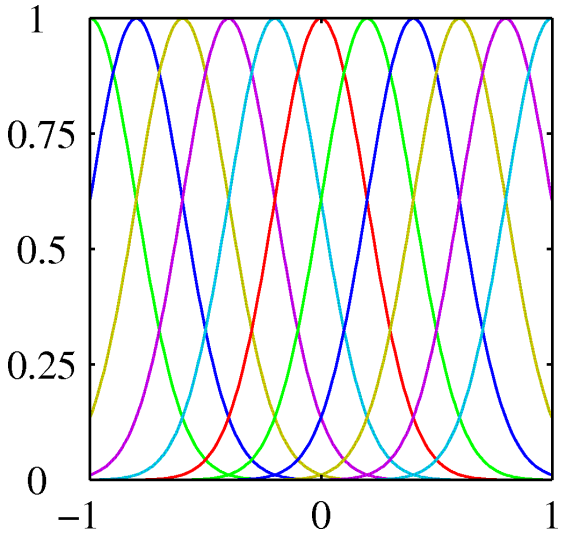
- Polynomial basis functions: $\phi_j(x) = x^j$
  - These are global features
  - A small change in $\boldsymbol{x}$ affects all basis functions



- Gaussian basis functions:

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

  - These are local features
  - A small change in $\boldsymbol{x}$ only affect nearby basis functions
  - μ_j and s control location and scale (width)



Based on slide by Christopher Bishop (PRML)

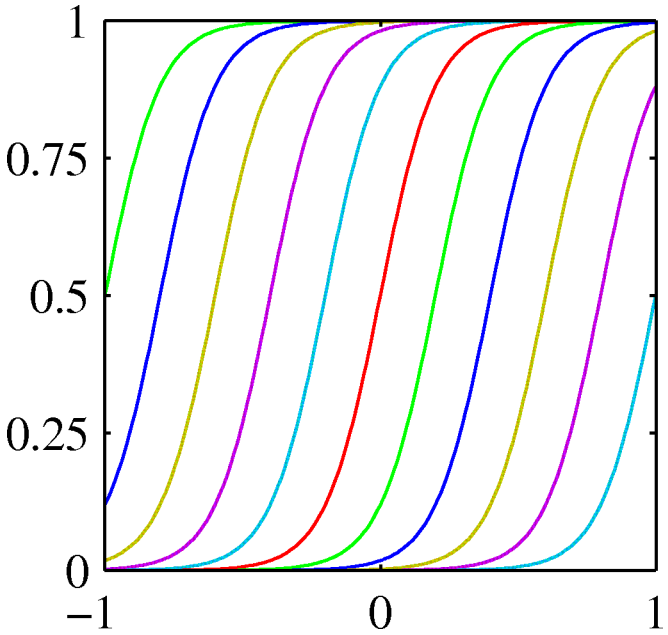
Linear Basis Function Models

- Sigmoidal basis functions:

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

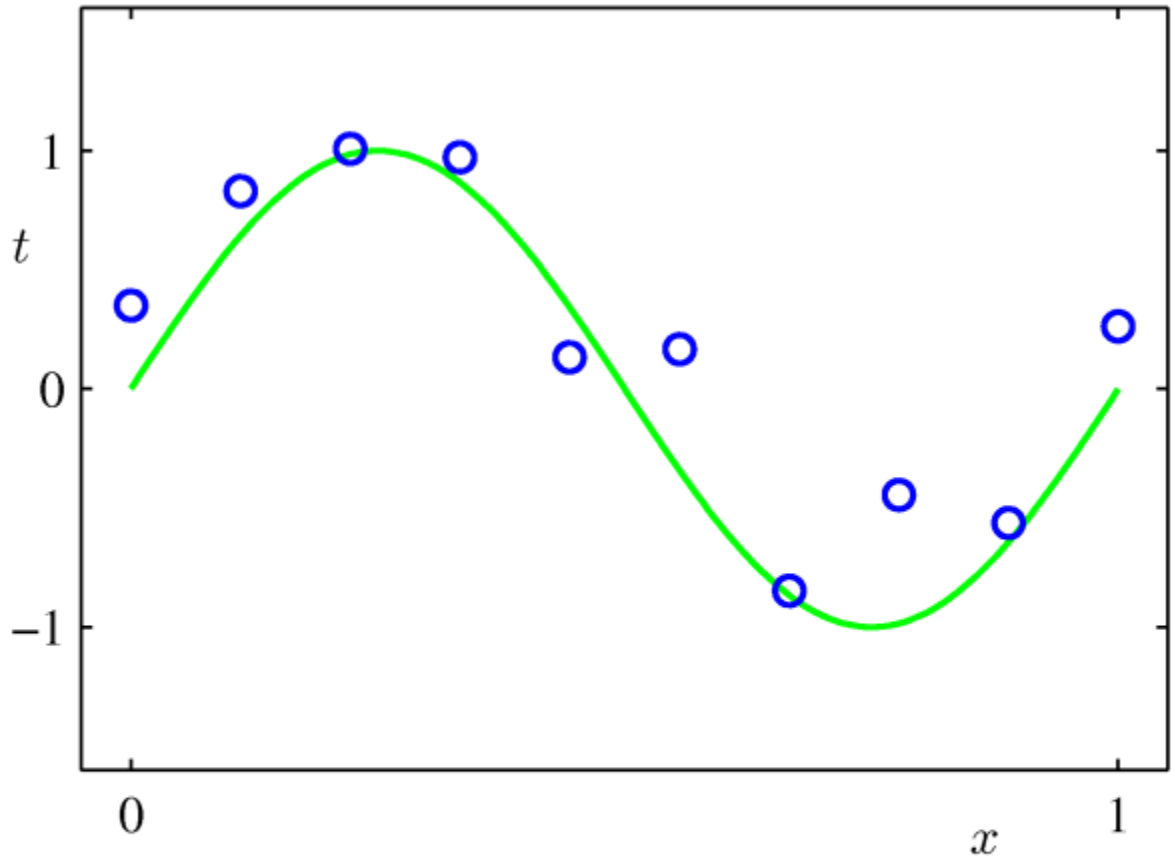
where

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$



  - These are also local
  - A small change in $\boldsymbol{x}$ only affects nearby basis functions
  - μ_j and s control location and scale (slope)

Example of Fitting a Polynomial Curve with a Linear Model



$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_p x^p = \sum_{j=0}^{p} \theta_j x^j$$

Linear Basis Function Models

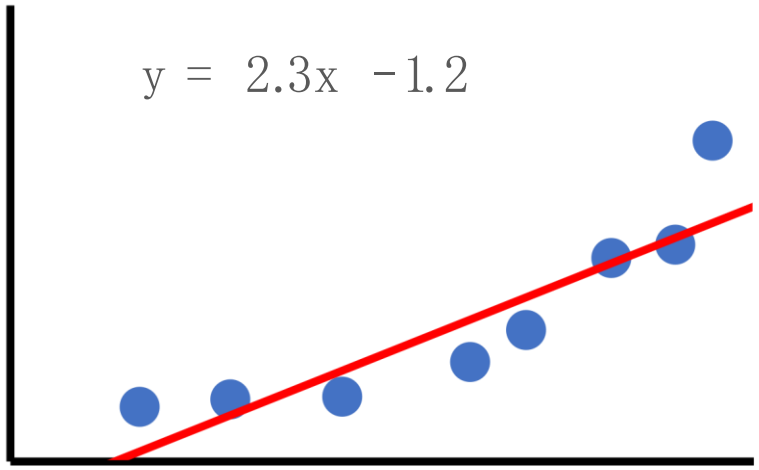
- Basic Linear Model: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j$
- Generalized Linear Model: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j \phi_j(\boldsymbol{x})$
- Once we have replaced the data by the outputs of the basis functions, fitting the generalized model is exactly the same problem as fitting the basic model
  - Unless we use the kernel trick – more on that with support vector machines
  - Therefore, there is no point in cluttering the math with basis functions

Based on slide by Geoff Hinton

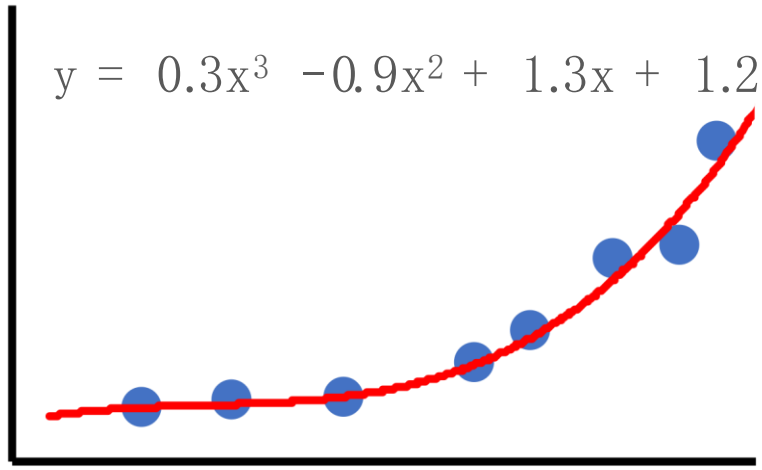
Regularized Linear Regression

(also called "Ridge Regression")

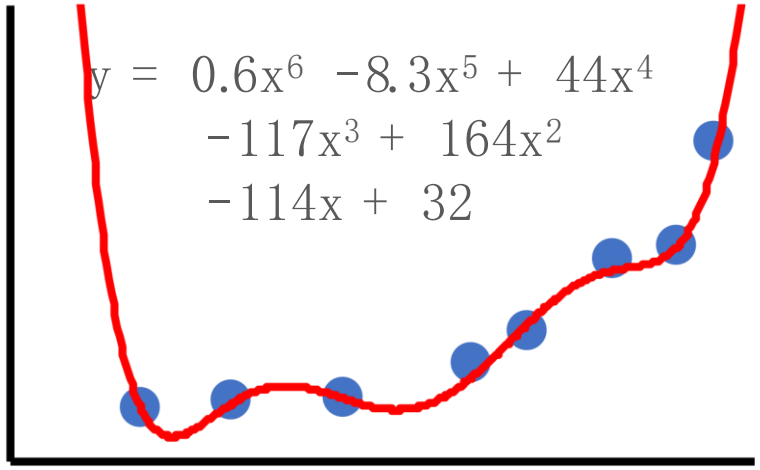
Quality of Fit



Underfitting
(high bias)



Correct fit

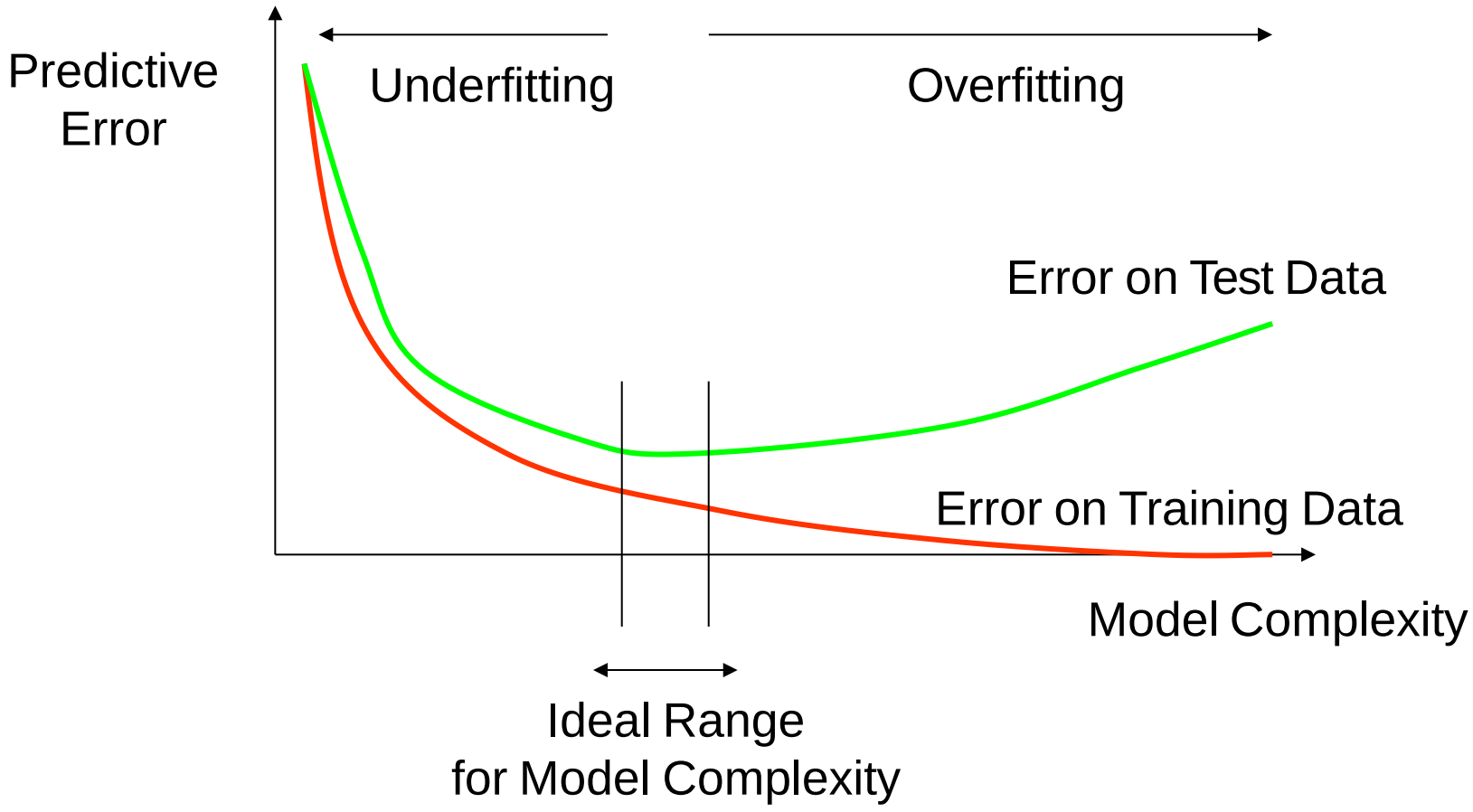


Overfitting
(high variance)

Overfitting occurs when:

- The learned hypothesis fits the training set very well ($\mathcal{L}(\boldsymbol{\theta}) \approx 0$)
- But fails to generalize to new examples

How Overfitting Affects Prediction



Regularization

- A method for automatically controlling the complexity of the learned hypothesis

- **Idea**: penalize for large values of θ_j
  - Incorporate penalty into the cost function
  - Works well when we have a lot of features, each that contributes a bit to predicting the label

- Can also address overfitting by eliminating features (either manually or via model selection)

L_2 Regularization

Regularized linear regression objective function:

$$\mathcal{L}(\boldsymbol{\theta}) = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2}_{\text{model fit to data}} + \underbrace{\lambda\sum_{j=1}^{d}\theta_j^2}_{\text{regularization}}$$

- λ is the regularization parameter ($\lambda \geq 0$)
- No regularization on θ_0!

Understanding L_2 Regularization

Cost Function: $$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$

- Note that $\sum_{j=1}^{d}\theta_j^2 = \|\boldsymbol{\theta}_{1:d}\|_2^2$

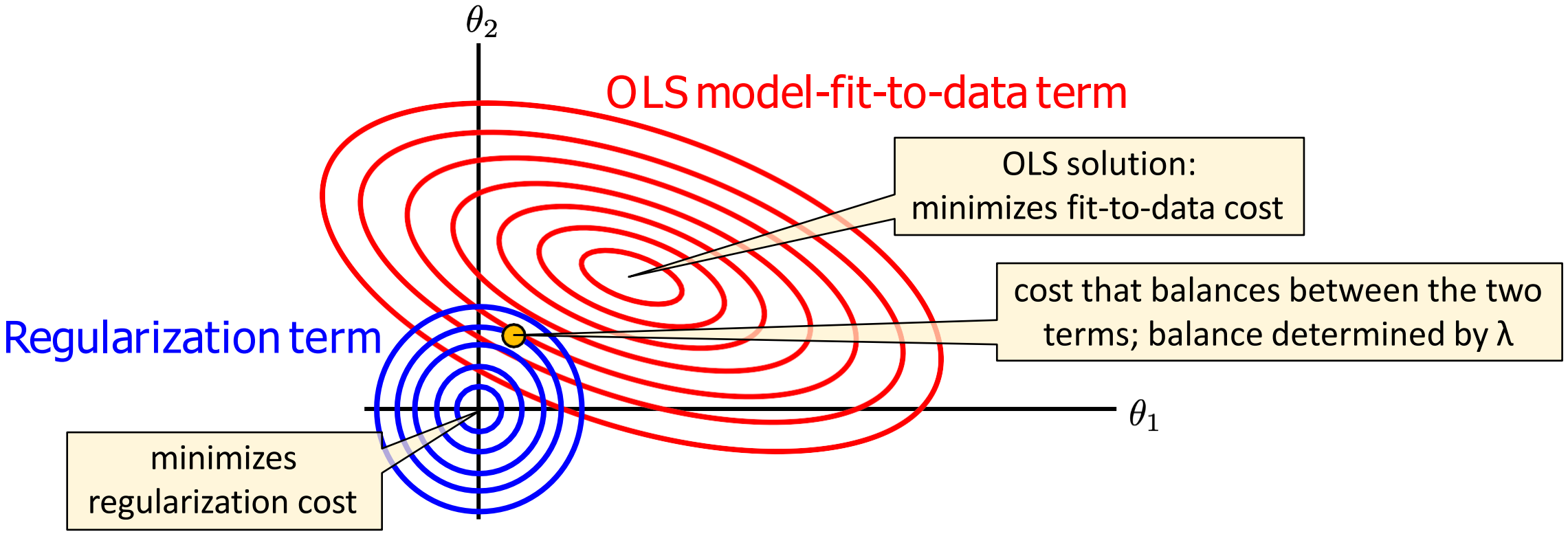
This is the magnitude of the feature coefficient vector squared!

- We can also think of this as: $\sum_{j=1}^{d}(\theta_j - 0)^2 = \|\boldsymbol{\theta}_{1:d} - \vec{\mathbf{0}}\|_2^2$

L_2 regularization pulls coefficients toward 0

Understanding L_2 Regularization

Cost Function: $$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$



Understanding L_2 Regularization

Cost Function: $$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$

- What happens as $\lambda \to \infty$?

$$\theta_0 + \cancel{\theta_1}^{0} x + \cancel{\theta_2}^{0} x^2 + \cancel{\theta_3}^{0} x^3 + \cancel{\theta_4}^{0} x^4$$

Coefficients approach 0

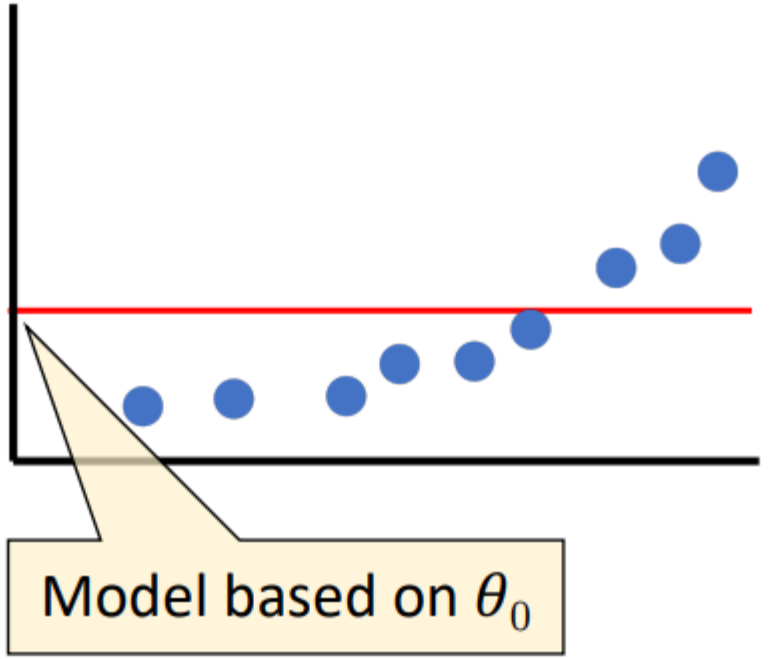


Illustration of Bias-Variance

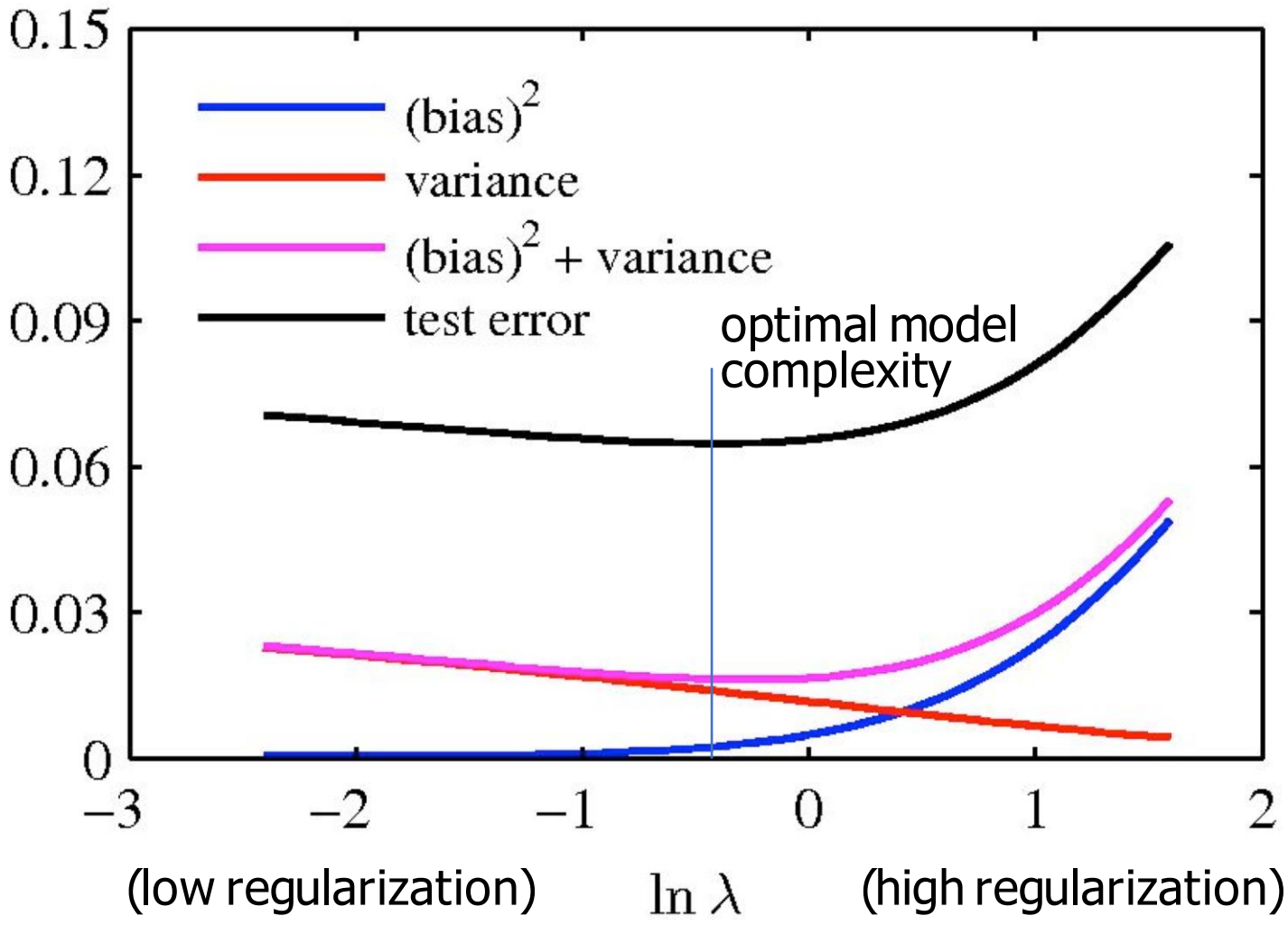


Figure provided by by Max Welling

Regularized Linear Regression

Cost Function: $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2$

- Fit by solving $\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$
- Gradient update:

$\frac{\partial}{\partial \boldsymbol{\theta}_0}\mathcal{L}(\boldsymbol{\theta})$ $\quad \theta_0 \leftarrow \theta_0 - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)$

No regularization on θ_0

$\frac{\partial}{\partial \boldsymbol{\theta}_j}\mathcal{L}(\boldsymbol{\theta})$ $\quad \theta_j \leftarrow \theta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)x_{ij} - \alpha\lambda\theta_j$

Extra factor of 2 in derivatives absorbed into α

regularization

Regularized Linear Regression

Cost Function: $$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$

Gradient update:

$$\theta_0 \leftarrow \theta_0 - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)$$

$$\theta_j \leftarrow \theta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)x_{ij} - \alpha\lambda\theta_j$$

- We can rewrite the gradient step for θ_j as:

$$\theta_j \leftarrow \theta_j\left(1 - \alpha\lambda\right) - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)x_{ij}$$

Regularization decreases the θ_j from the previous GD step

Regularized Closed Form Linear Regression

- To incorporate regularization into the closed form solution:

$$\theta = \left(\quad \right)^{-1} X^\intercal y$$

Regularized Closed Form Linear Regression

- To incorporate regularization into the closed form solution:

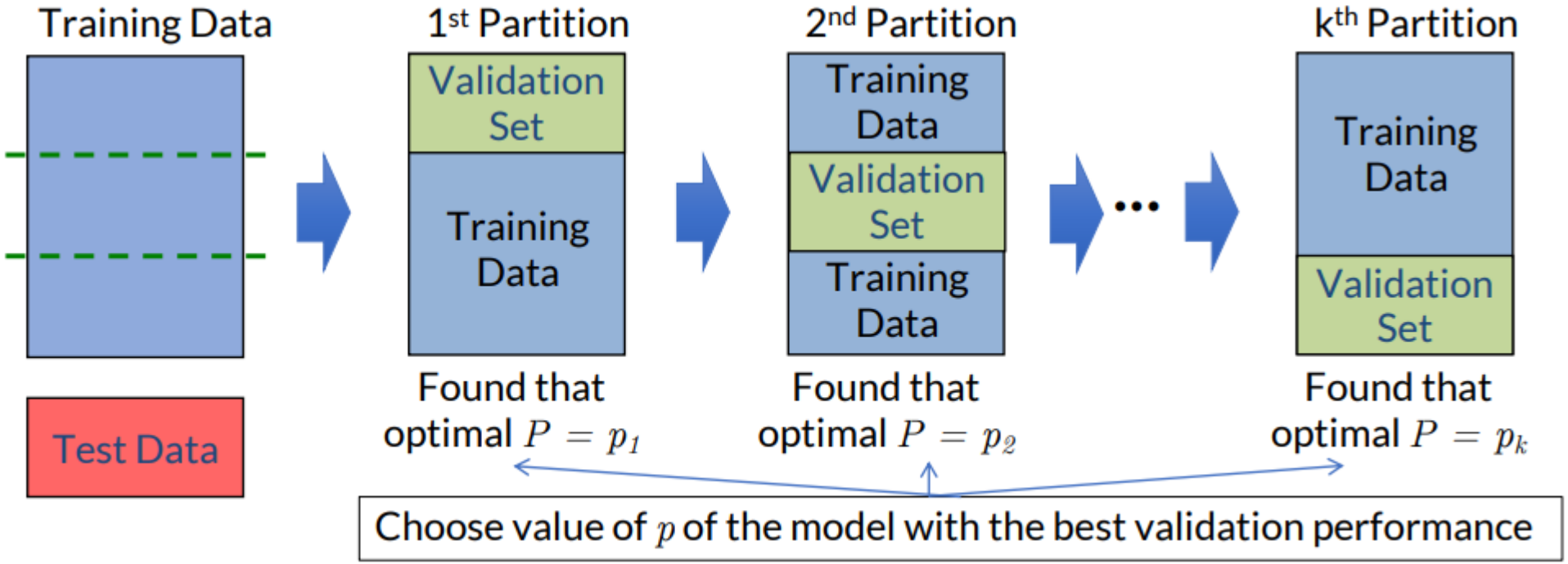
$$\boldsymbol{\theta} = \left(\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \right)^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

- Can derive this the same way, by solving $\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = 0$

- Can prove that for $\lambda > 0$, inverse exists in the equation above

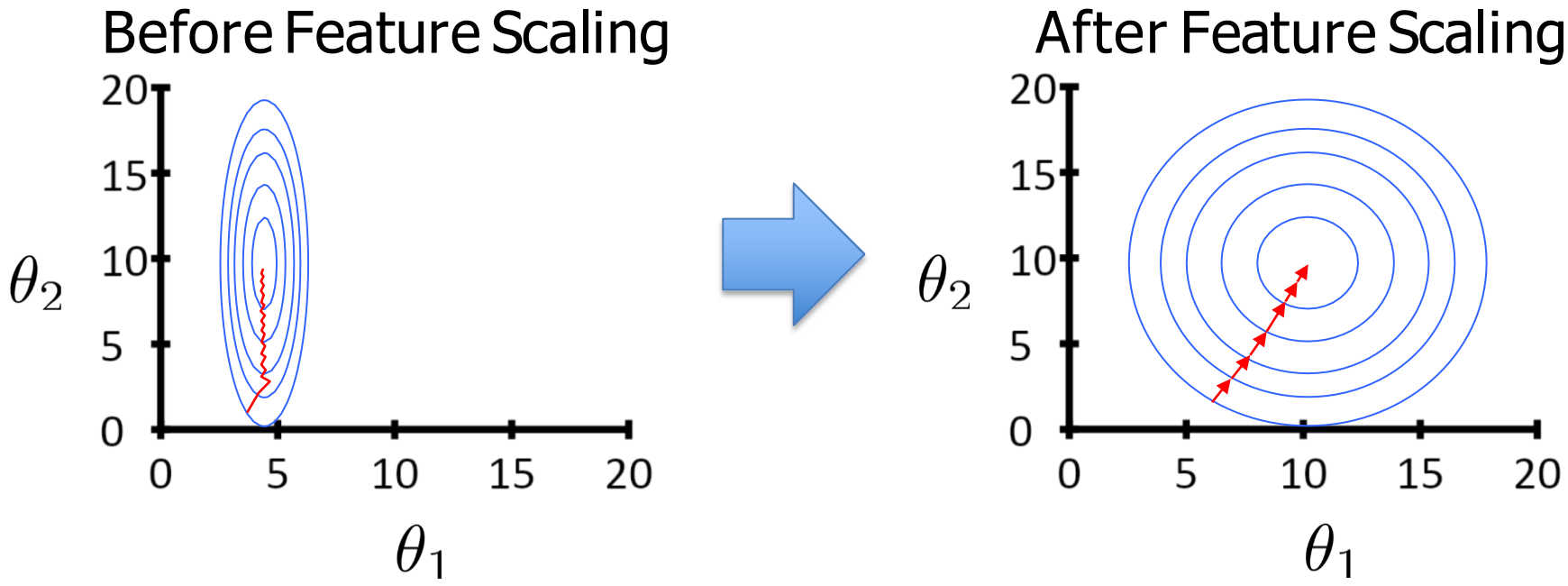
Optimizing Model Parameters

- Using CV to choose value of model parameter P:
- Search over space of parameter values $p \in values(P)$
  - Evaluate model with $P = p$ on validation set
- Choose value p' with highest validation performance
- Learn model on full training set with $P = p'$



Improving Learning: Feature Scaling

Idea: Ensure that feature have similar scales



- Makes gradient descent converge much faster
- Important for ridge regression, since it preferentially shrinks large coefficients

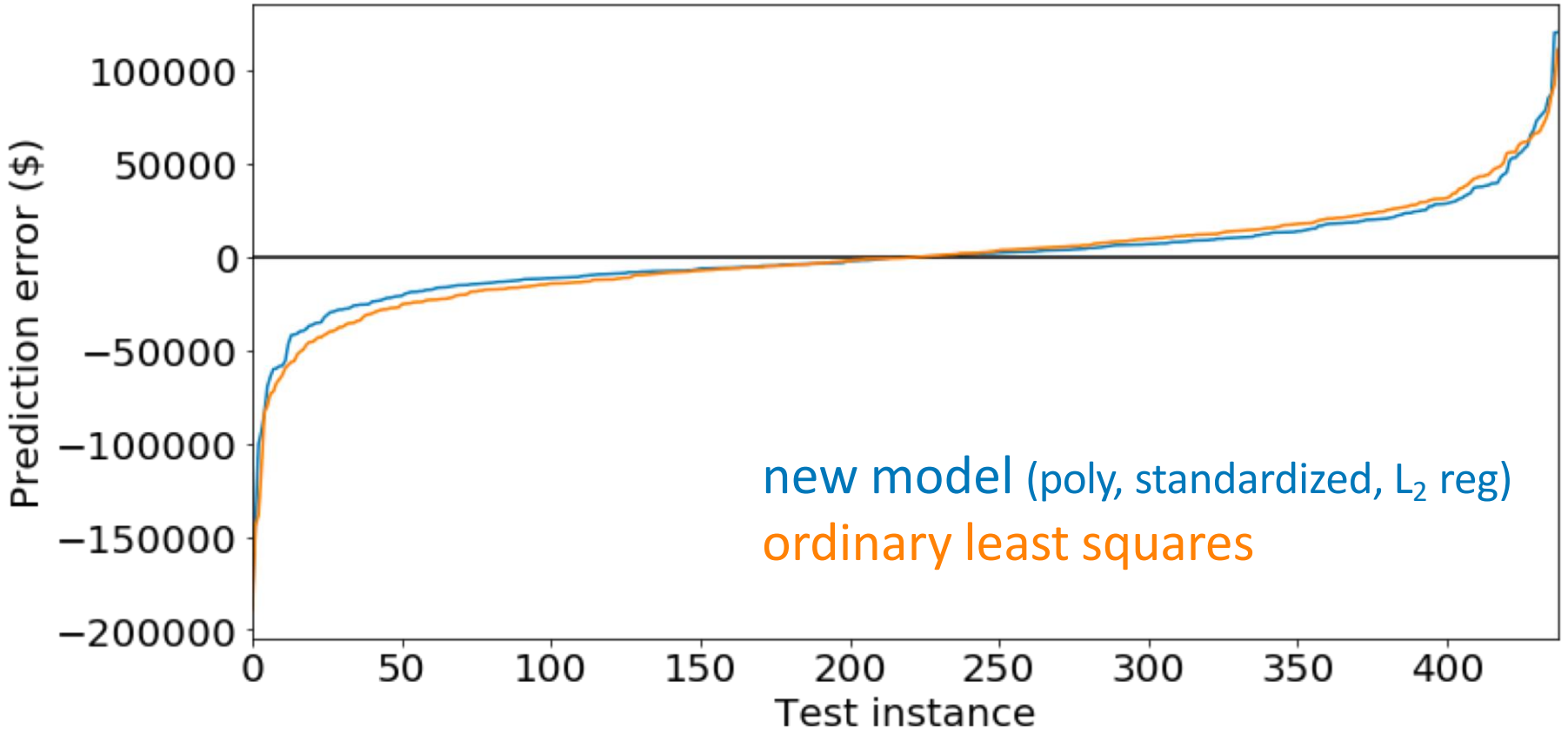
Feature Standardization

- Rescales features to have zero mean and unit variance
  - Let μ_j be the mean of feature j : $\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$
  - Replace each value with: $x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{s_j}$ for $j = 1 \ldots d$ (not x_0!)
    - s_j is the standard deviation of feature j
    - Could also use the range of feature j (max value – min value) for s_j
  - Outliers can cause problems

Must use the same transformation for both training and prediction
(μ_j and s_j are computed on training data and also used on the test data)

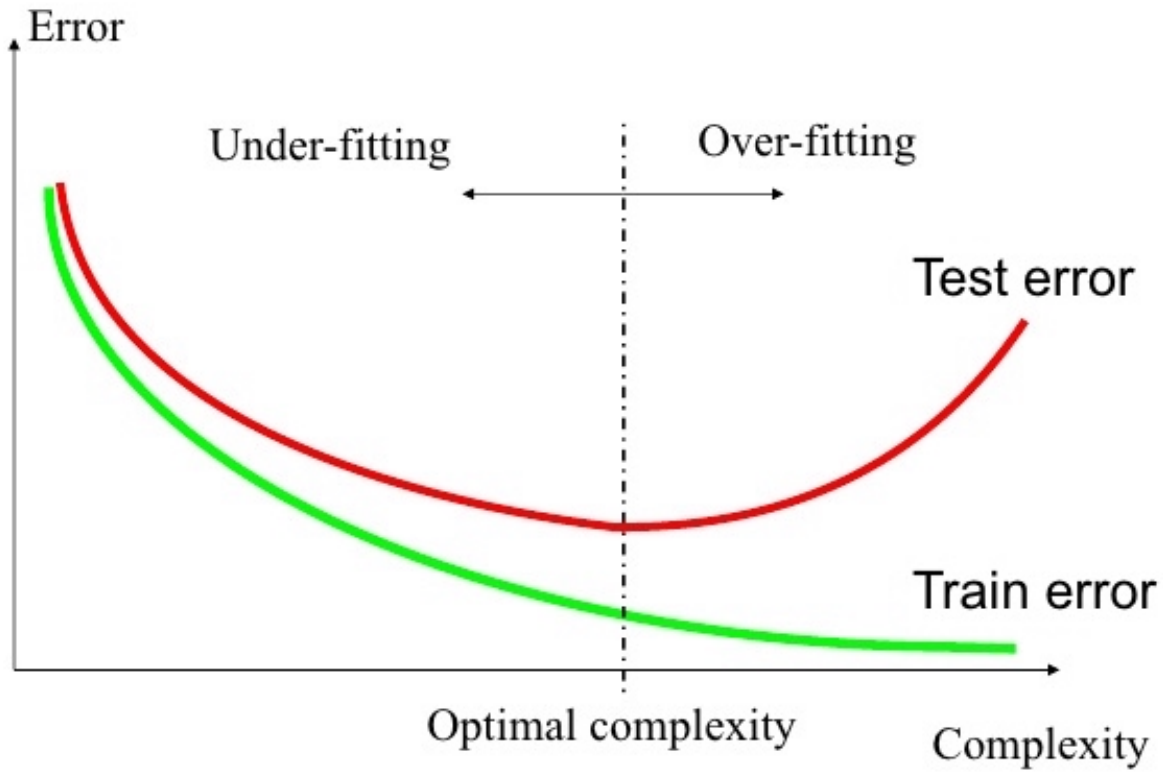
Housing Data

- Polynomial features of degree 2, feature standardization, L2 regularization



Underfitting versus overfitting

- All about improving generalization capacity



Splitting the data into training, validation and test

- In order to build a model with good generalization capacity, we have to test it on unseen data
- First approach (works well when we have enough data):

➢ 50% samples for training

➢ 25% samples for validation

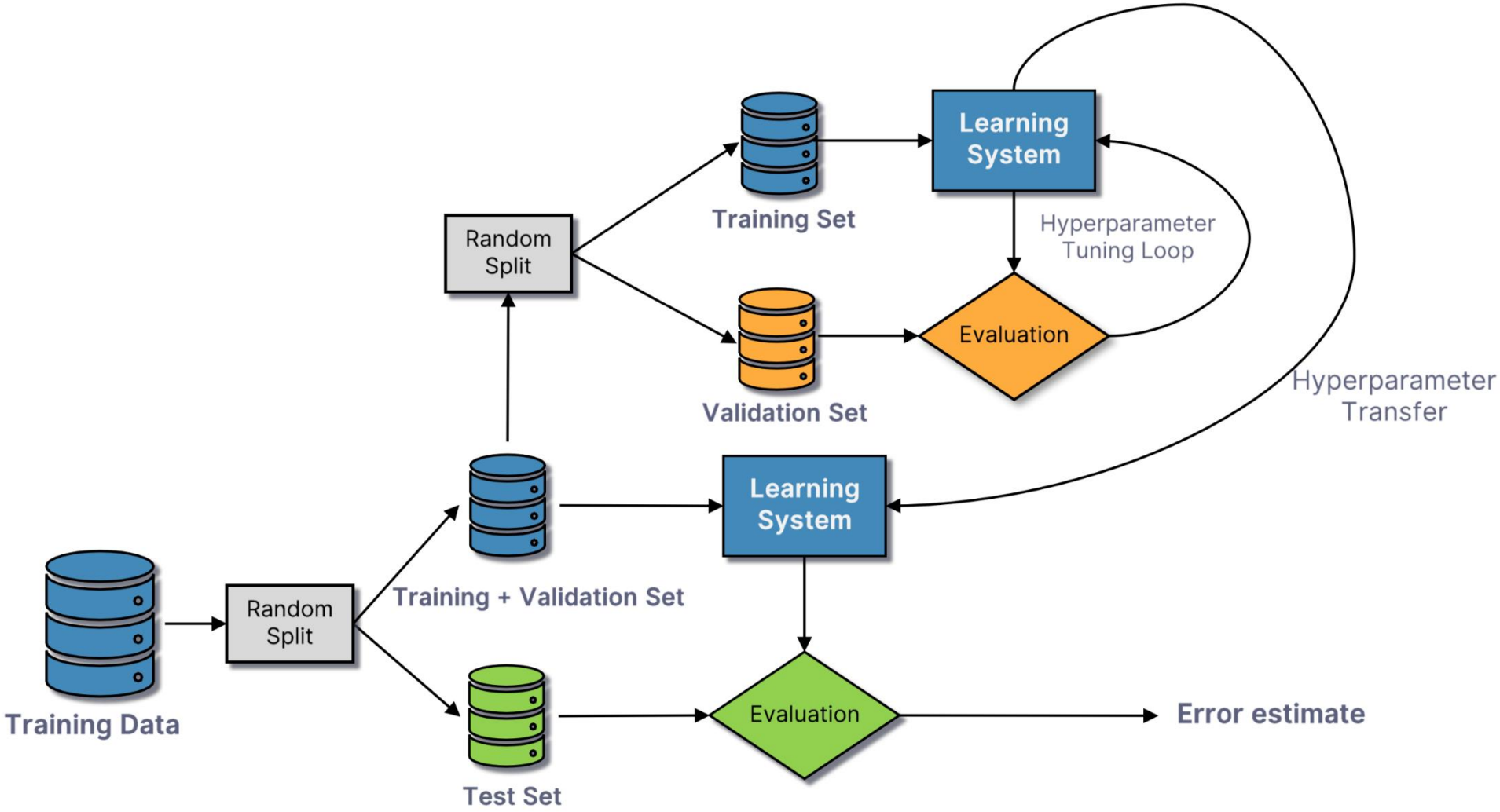
➢ 25% samples for test

(percentages can vary)

Why not just split the data into training and test?

- Repeatedly using the same split when trying different hyperparameters can “wear out” the test set:
  - ➢ We are overfitting in hyperparameter space!
- We obtain a better error estimate by tuning the hyperparameters on a (different) validation set

Training, validation, test

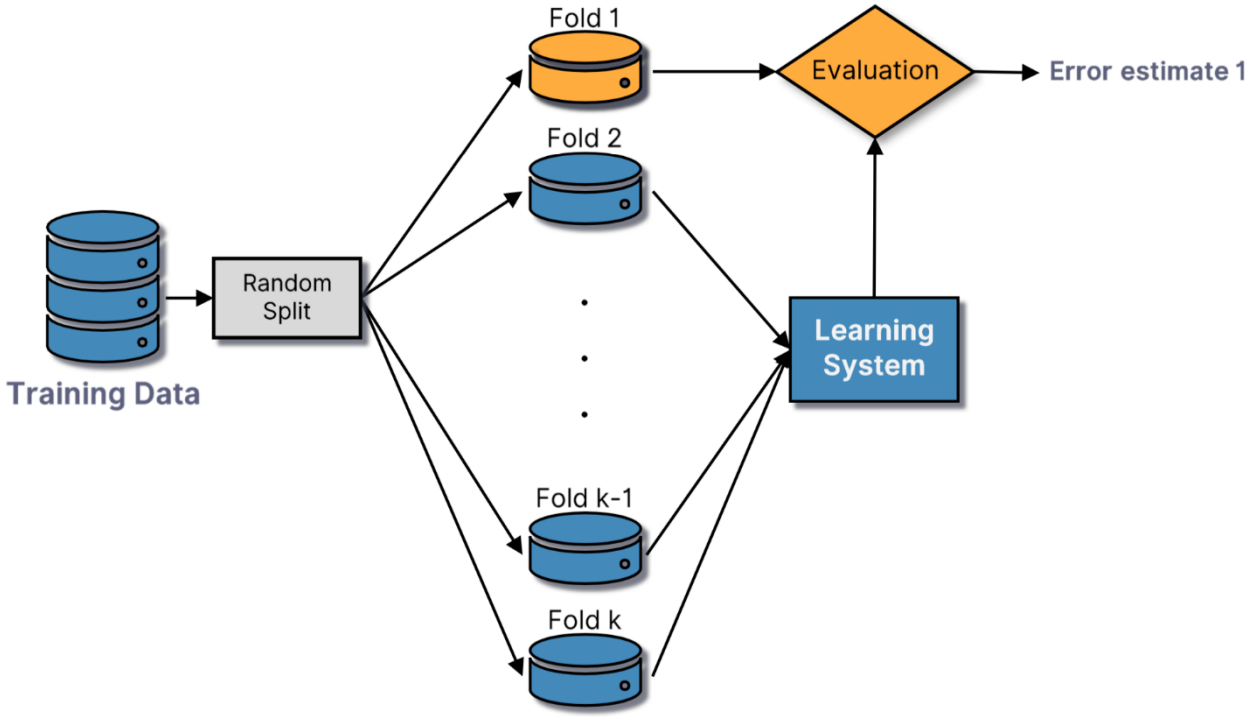


Cross-validation

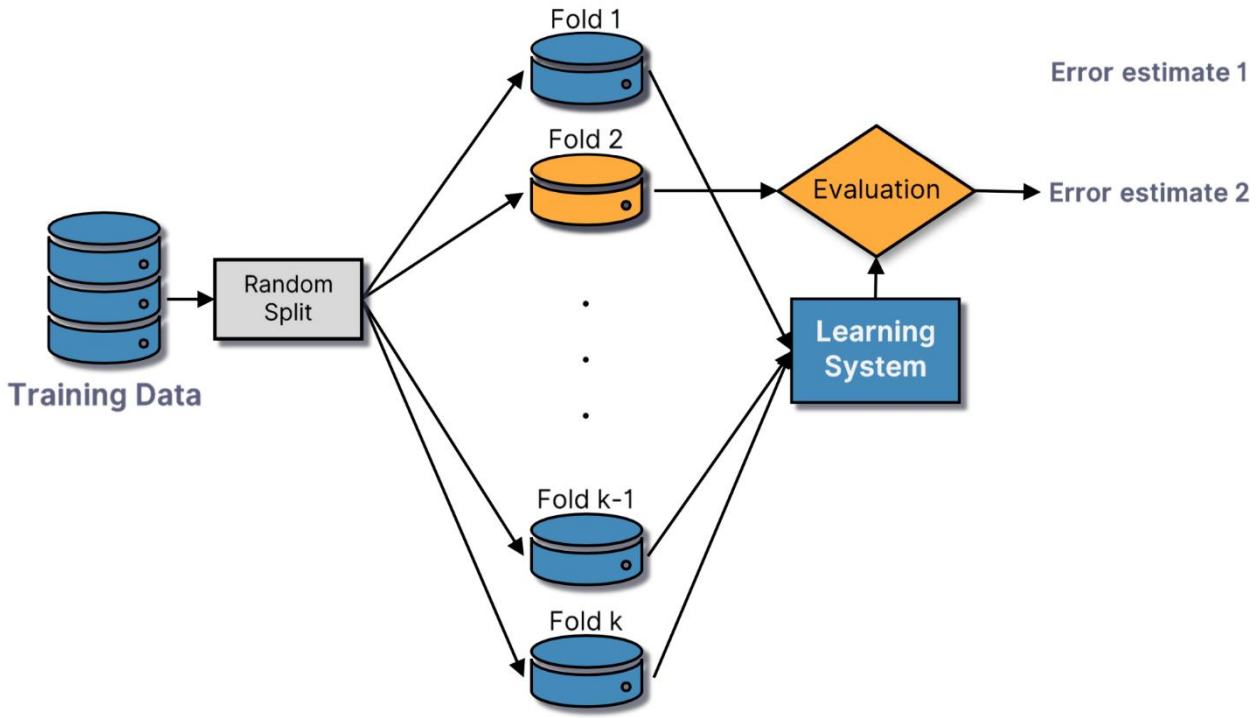
- Another approach (works well when we have limited data):
  - Split the data into k equal parts (folds)
  - Train on k-1 folds and test on the left out fold
  - Repeat for k times
  - Average the results

- When the number of folds is equal to the number of samples:
  - Leave-one-out cross-validation

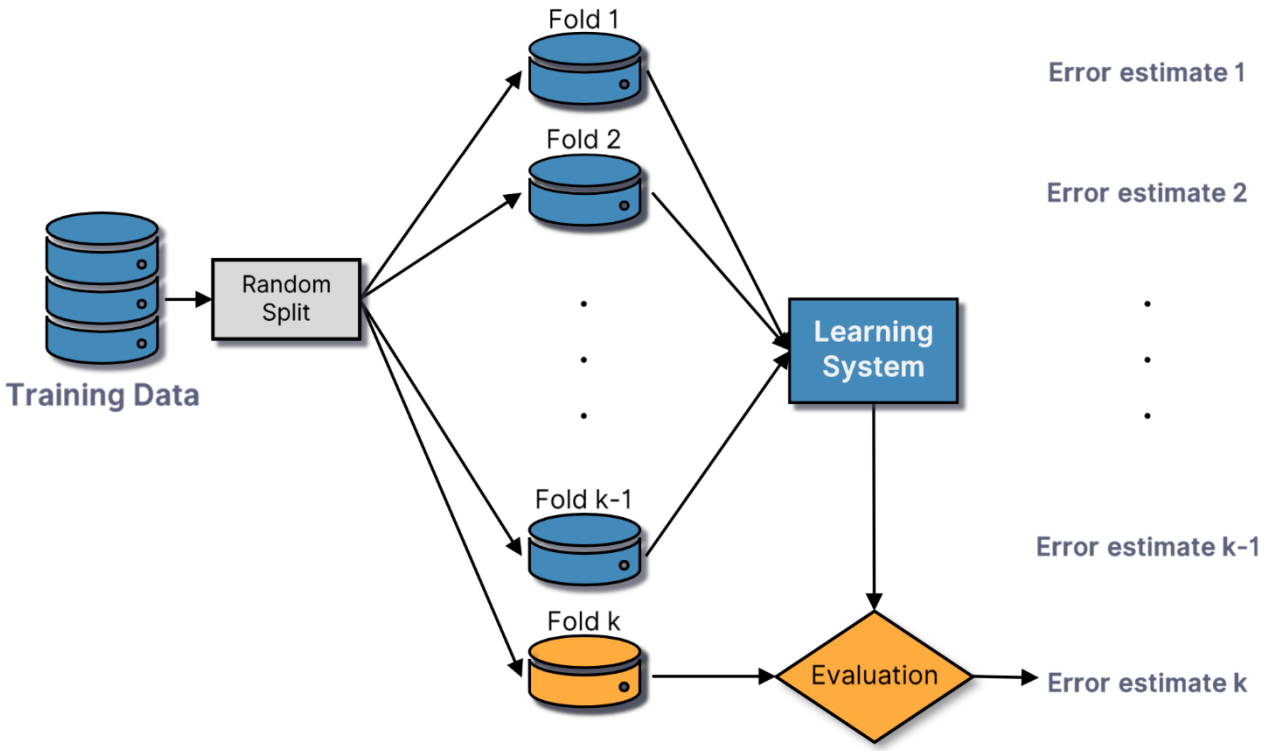
Cross-validation



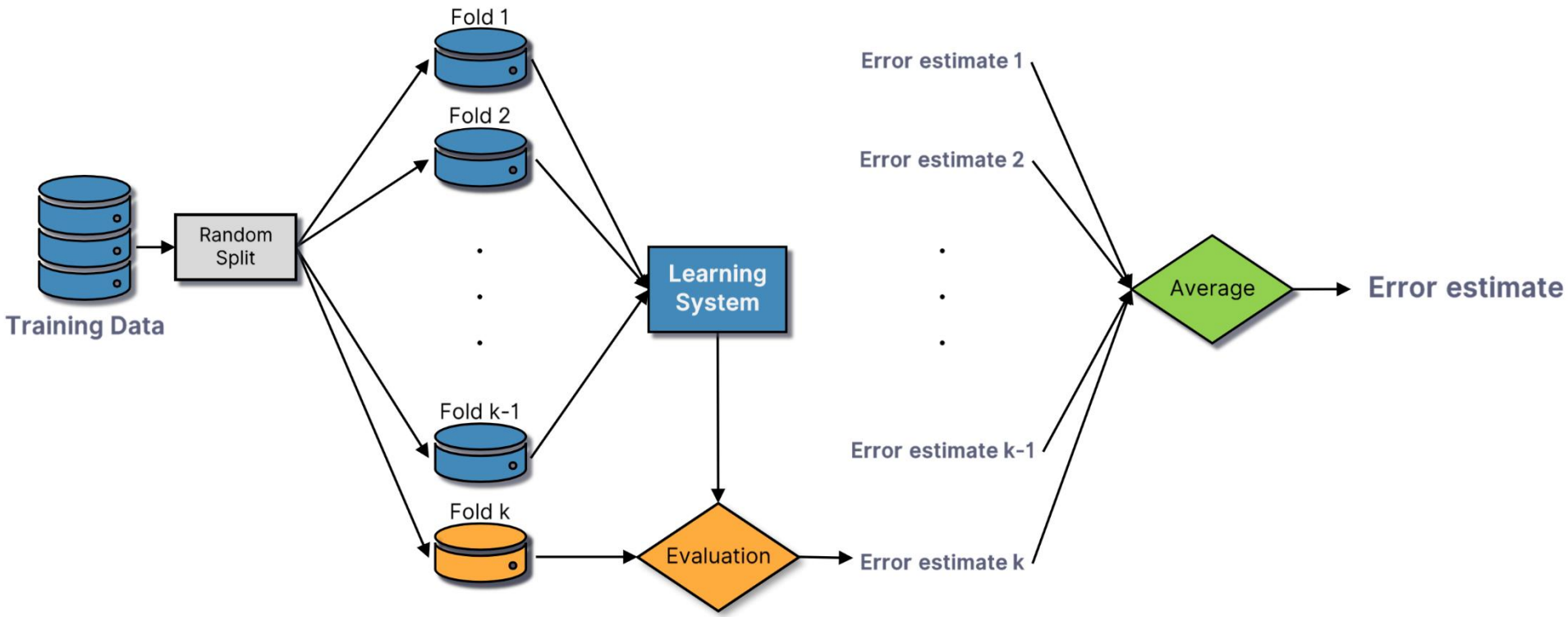
Cross-validation



Cross-validation



Cross-validation



Improving generalization

- Early Stopping
  - ➢ Stop the learning process when we notice that the validation error starts to increase

- Regularization
  - ➢ Add a penalty term to the loss function to reduce model complexity, by imposing smoothing restrictions or a limit on the weight vector norm:

$$\min_f \sum_{i=1}^{n} V(f(\hat{x}_i), \hat{y}_i) + \lambda R(f)$$