MACHINE LEARNING

MODEL SELECTION I

AGENDA

01 Overfit vs underfit

02 LR by local weighting

03 Regularization

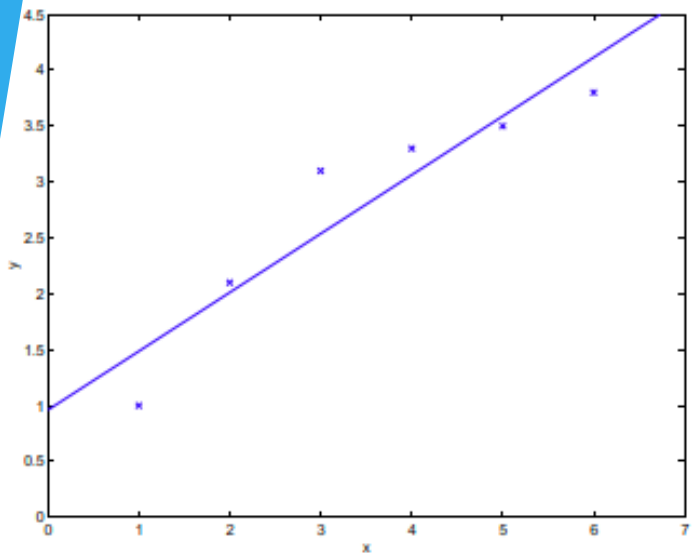
04 K-fold cross validation

Overfit Vs Underfit

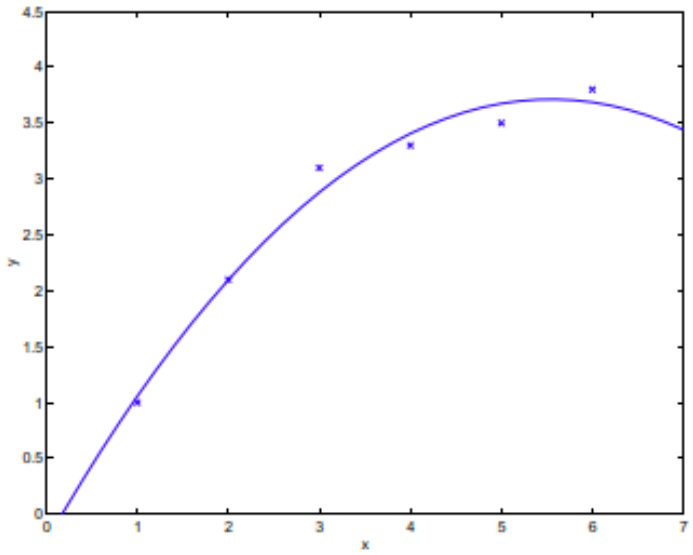
OVERFIT VS UNDERFIT

INTRODUCTION

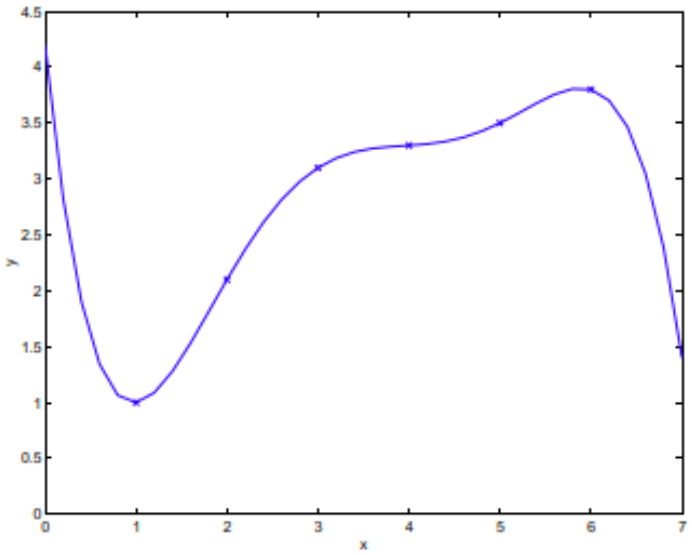
Addressing back to linear regression, a **problem** arises when establishing a **high polynomial degree** or defining a **large set** of basis functions:



“under-fitting”



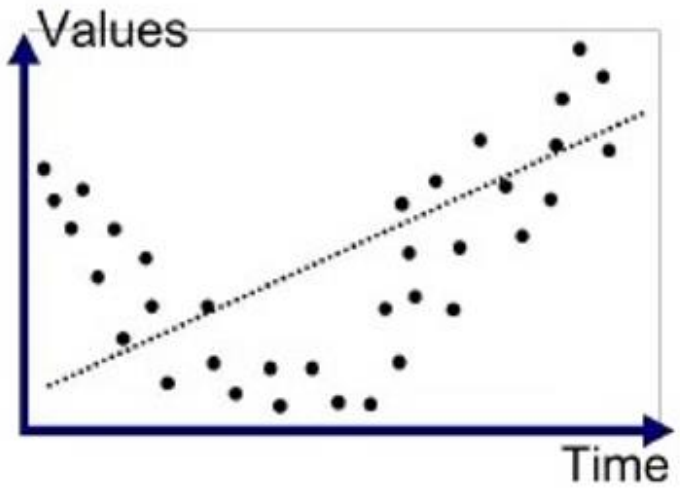
“good fit”



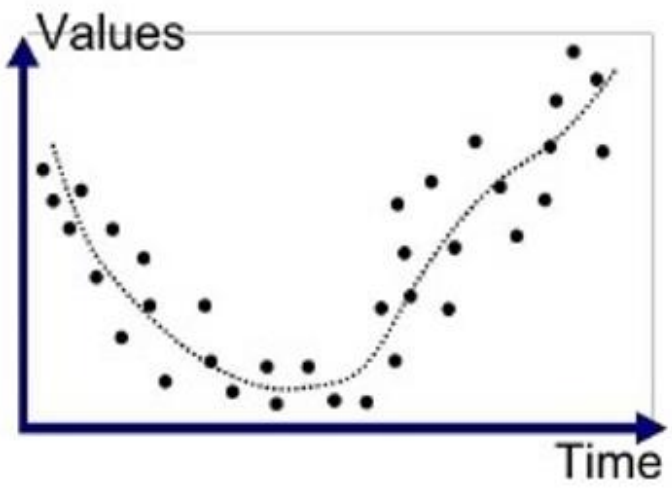
“over-fitting”

OVERFIT VS UNDERFIT

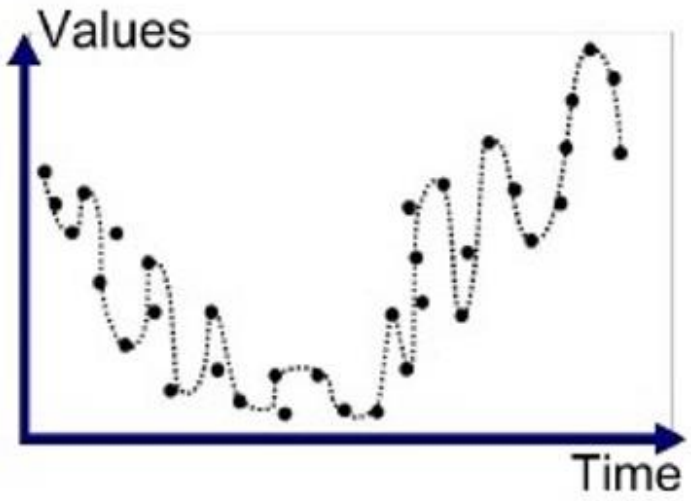
POSSIBILITIES



Underfitted



Good Fit/Robust



Overfitted

OVERFIT VS UNDERFIT

CROSS VALIDATION

To **evaluate a model**, it is **not enough** to just **fit the hypothesis** to the **training data**. The **generalization power** of the model should be seen when facing **new data**.

For this, the **cross-validation method** is used, where the **data** is **divided** into **two sets**:

- **Training data: data** that will be used to **train** the **model**. For a set with limited data, **70%** of the data is **added**.
- **Validation data: data** that will be used to **make predictions** based on the **parameters obtained** with the **model training**. For a limited set of data, **30%** of the data is **added**.

OVERFIT VS UNDERFIT

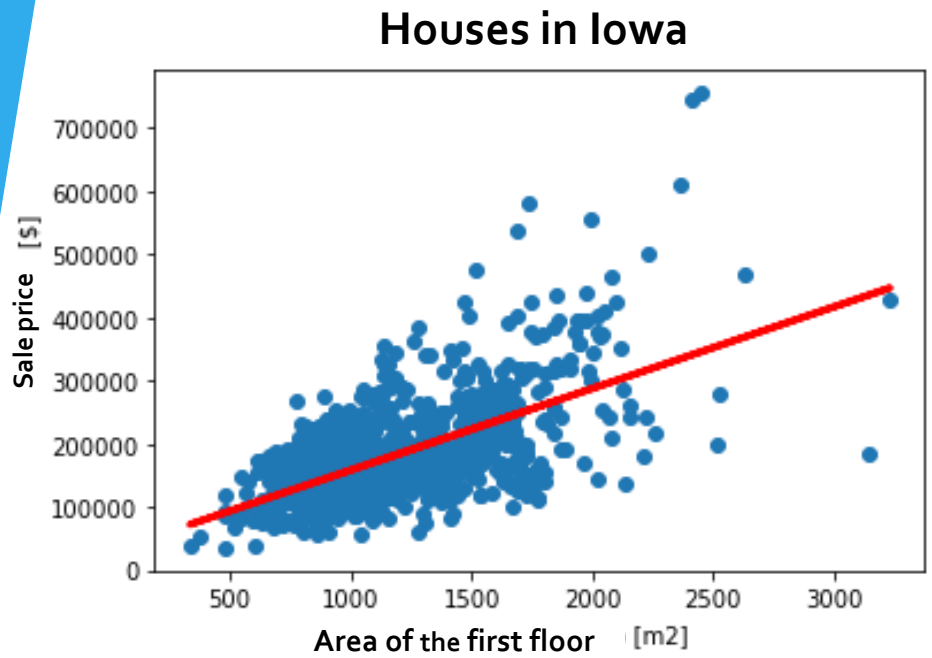
CROSS VALIDATION PROCESS

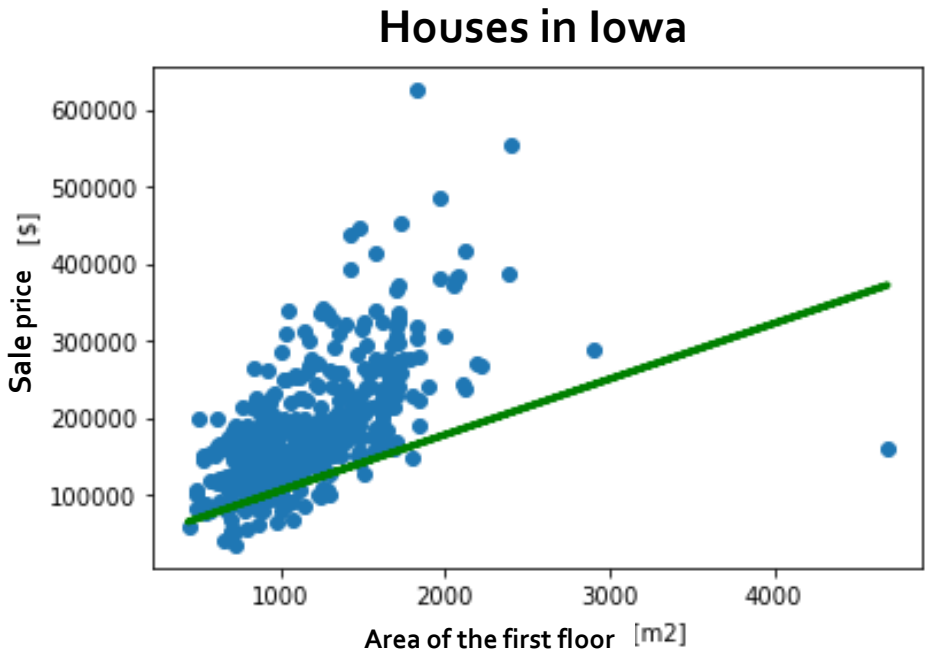
The **process** would be as follows:

1. **Divide** the data set into **two subsets**: **70**% for **training** and **30**% for **validating**.
2. **Train** the **model** with **70**% of the data and **obtain** the **best weights** $\vec{w}$.
3. With the **weights** $\vec{w}$ **calculate predictions** for the remaining **30**% of the **data**.
4. **Compute** the **cost functions** for steps **2** and **3.**
5. **Repeat** the process with **another hypothesis** (model) until the **model** with the **best validation** and **training errors** is **selected.**

OVERFIT VS UNDERFIT

NORMAL EQUATIONS



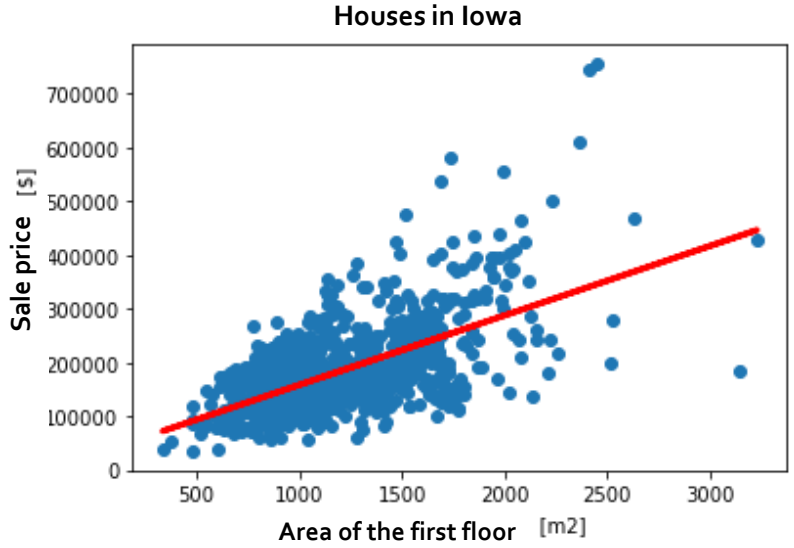


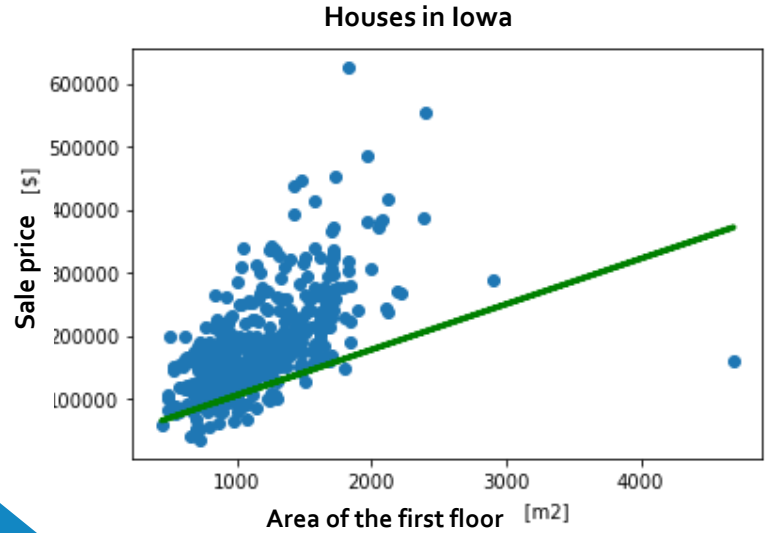
Training data
70%

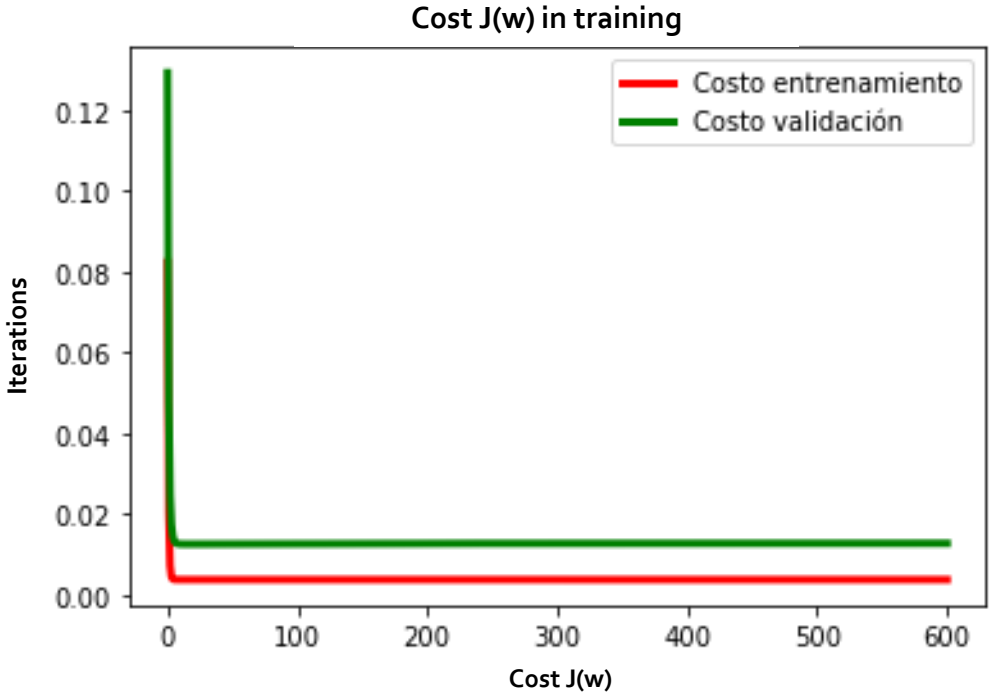
Validation data
30%

OVERFIT VS UNDERFIT

GRADIENT DESCENT

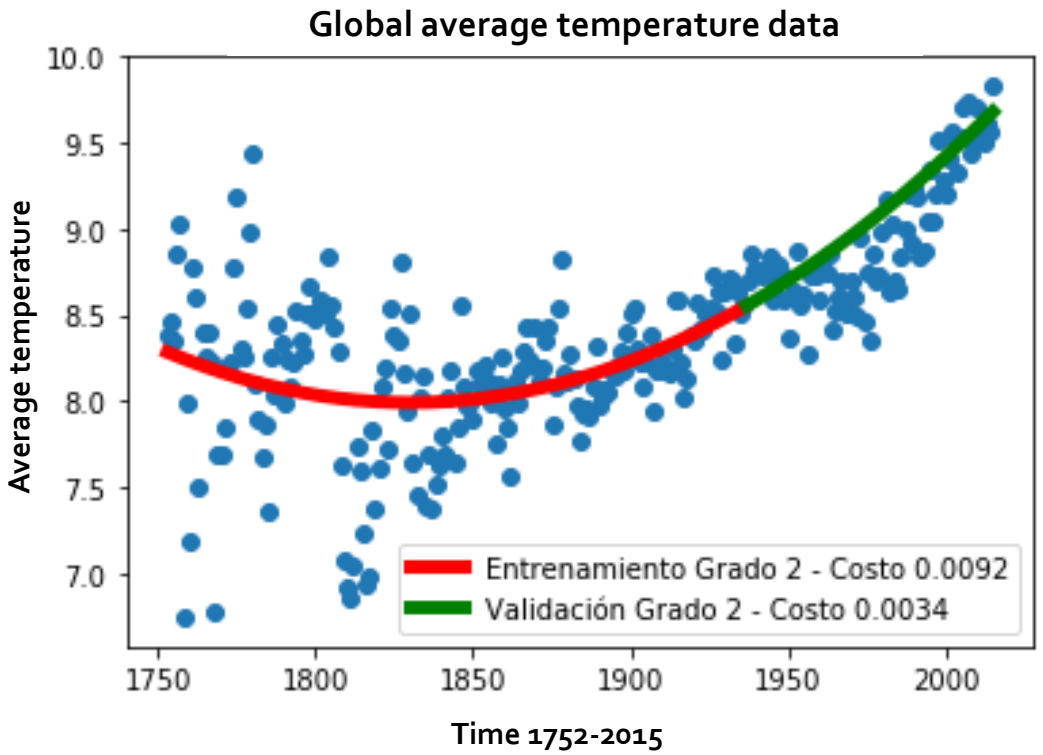


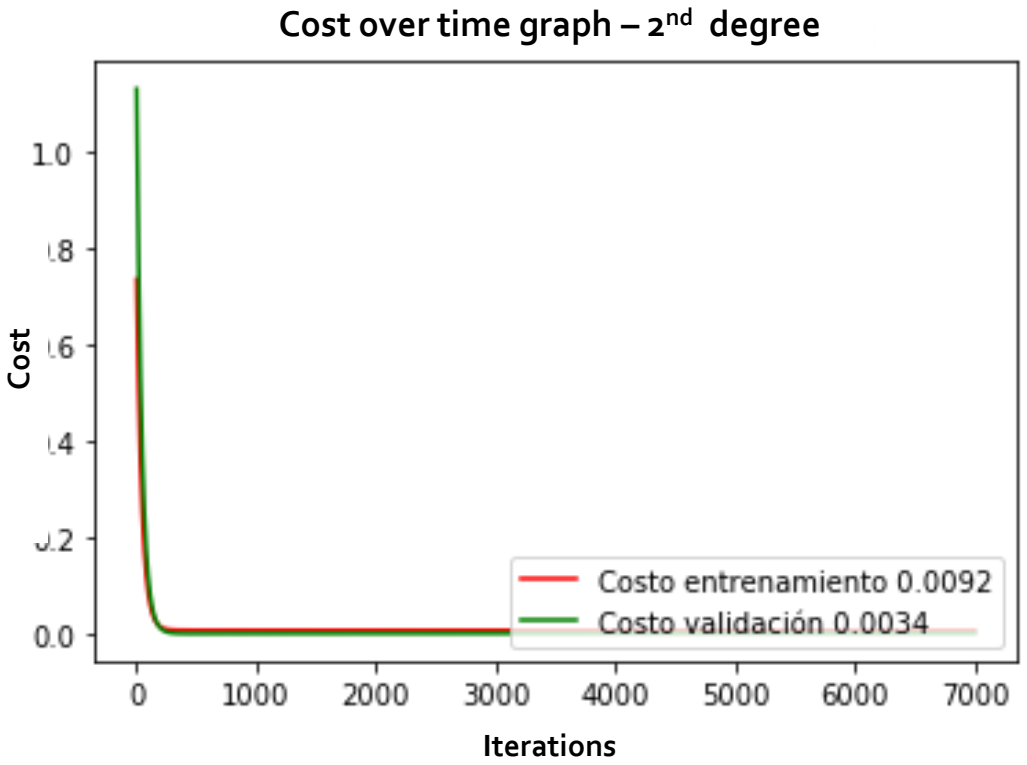




OVERFIT VS UNDERFIT

IDEAL TRAINING

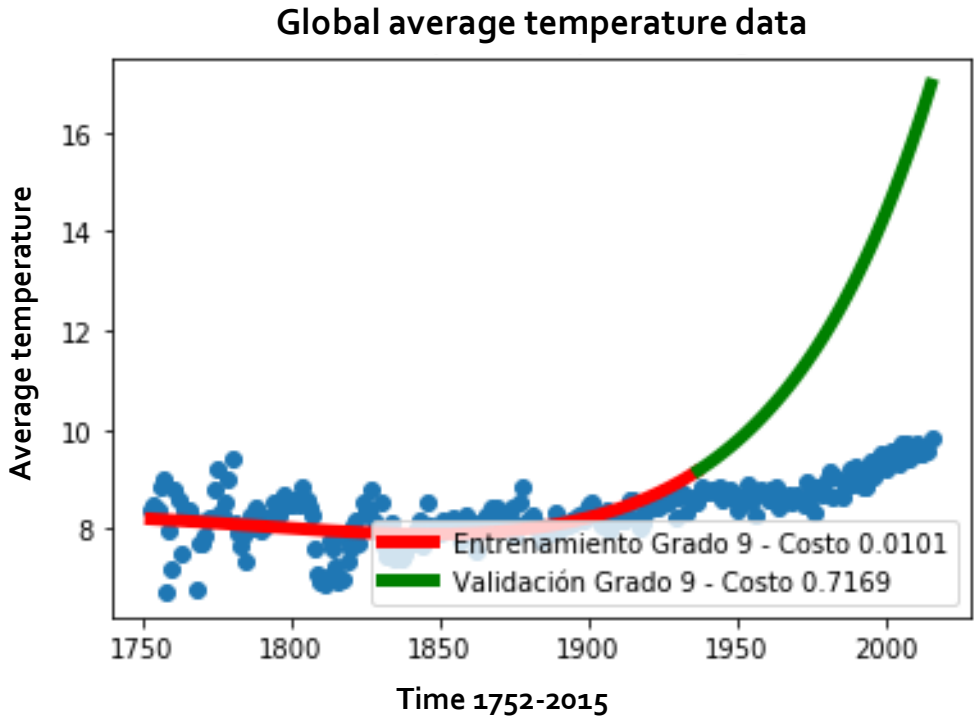


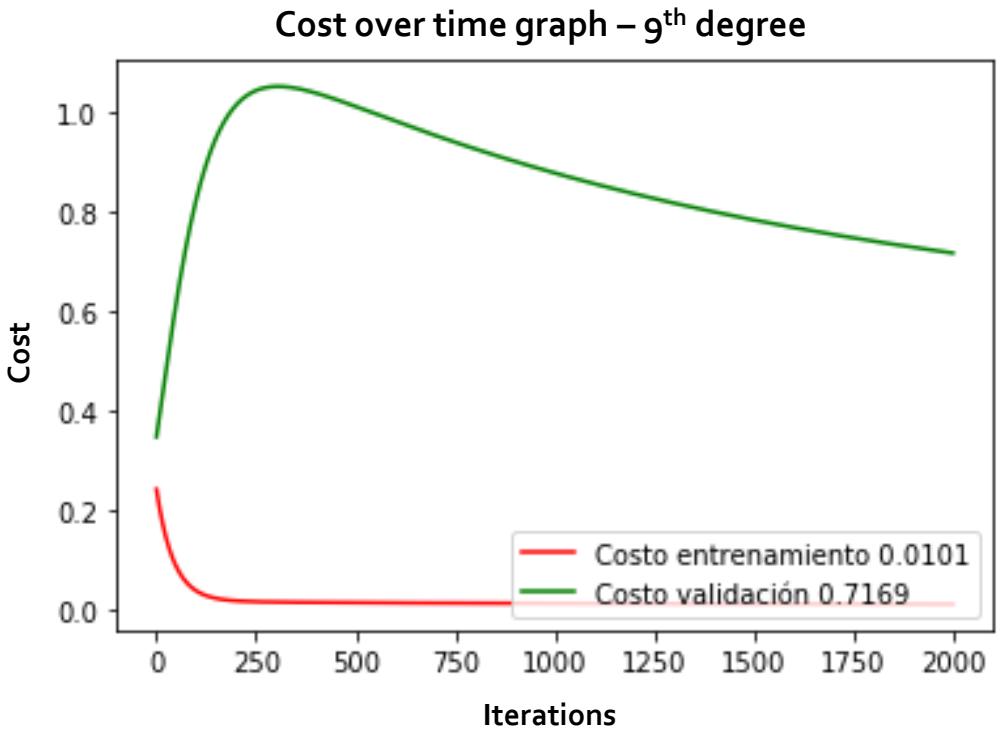


2nd DEGREE

OVERFIT VS UNDERFIT

OVERFITTING





9th DEGREE

OVERFIT VS UNDERFIT

DEFINITIONS

In a **formal** way, the previous concepts can be **defined** as:

- **Underfit:** the **hypothesis space** is too **restricted** to capture the **relevant structure** of the **function being modeled**.
- **Overfit:** the **hypothesis space** is **large** enough that it is possible to **model noise** in the **training data**, which results in a **non-generalizable structure**.

Adams, R. (2018). *Overfitting and Regularization*. COS324 – Introduction to Machine Learning, Princeton

OVERFIT VS UNDERFIT

OVERFITTING CAUSES

There are different **causes for overfitting**:

1. **Limited data**: there may be **false patterns** that would not exist with a larger data set.

2. **Noise** in the **data**: poor **signal-to-noise ratio** in the data may complicate the detection of the true structure that needs to be learned.

3. The **hypothesis space** is too large: the more **flexibility** the **hypothesis** has, the **greater** the **possibility** of **fitting false patterns**.

4. The **input space** is **highly dimensional**: adding features (more **base functions**) that are not important can introduce **unwanted noise**. Hence the **importance of feature selection**.

OVERFIT VS UNDERFIT

SUMMARY

The problem is **summarized** as follows:

ADJUST TRAINING DATA IN THE BEST WAY POSSIBLE, WITHOUT LOSING THE POWER OF GENERALIZATION

LR BY LOCAL WEIGHTING

LR BY WEIGHTING

WEIGHTING FUNCTION

Therefore, the selection of the $\boldsymbol{X}$ characteristics is very important to make an **adequate representation** of the data.

To **alleviate** the **selection problem** a bit, a **local weight regression model** is proposed. A **nonparametric** method (the **number** of **parameters** grows with the **number** of **input data** $\boldsymbol{m}$).

$$\arg\min_{\vec{w}} \boldsymbol{MSE} = \arg\min_{\vec{w}} \frac{1}{2m} \sum_{i=1}^{m} \boldsymbol{\theta}^{(i)} \left(\boldsymbol{y}^{(i)} - \boldsymbol{w}^{T} \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)\right)^{2}$$

Where $\boldsymbol{\theta}^{(i)}$ represents a **local weighting function**.

LR BY WEIGHTING

WEIGHTING FUNCTION

The **local weight function** is an **arbitrary choice**. Usually the following function is used, where τ^2 is referred to as **bandwidth**:

$$\theta^{(i)} = e^{-\frac{\left\|x^{(i)} - x\right\|_2^2}{2\tau^2}}$$

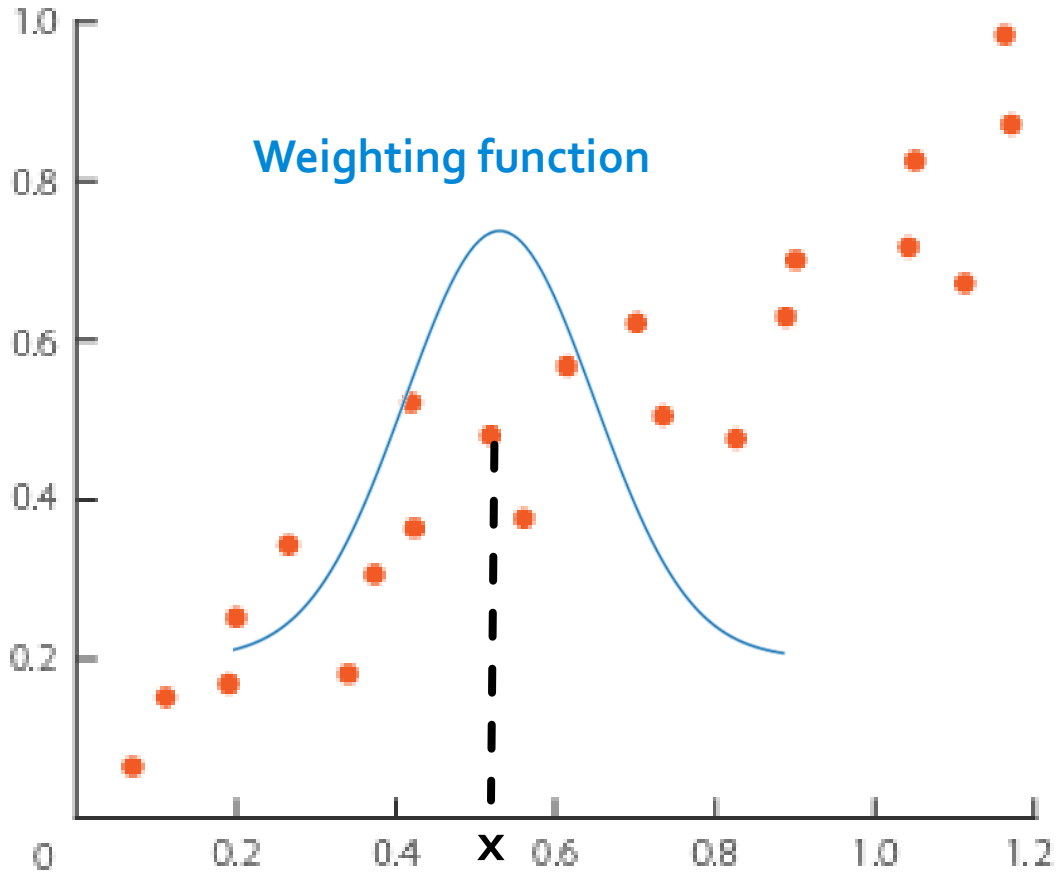
If $\left|x^{(i)} - x\right| \to 0$, then $\theta^{(i)} \to 1$

If $\left|x^{(i)} - x\right| \to \infty$, then $\theta^{(i)} \to 0$

In other words, as the data $x^{(i)}$ gets **further away** from the x datapoint, the **least importance** it'll have for the **regression**.

LR BY WEIGHTING

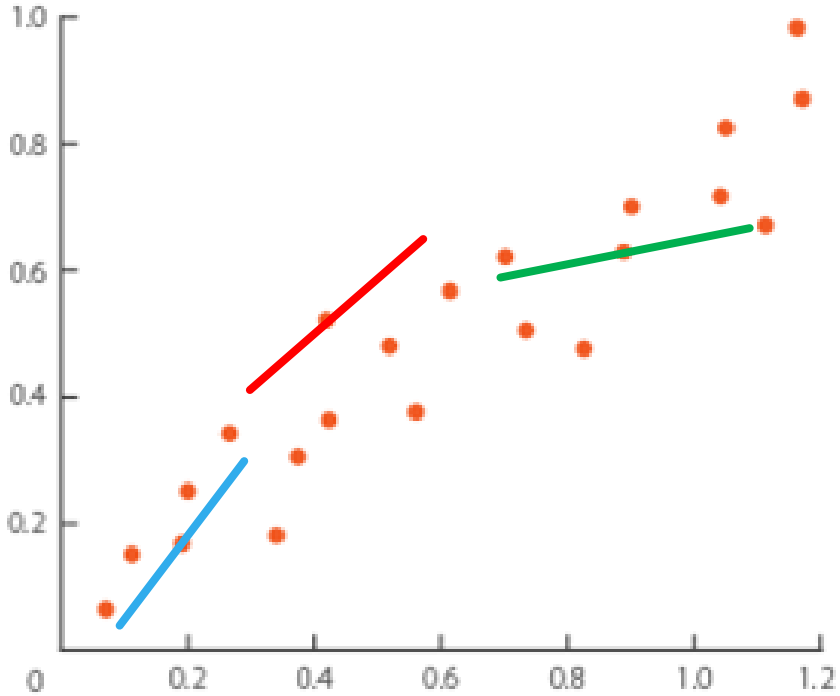
WEIGHTING FUNCTION



L R B Y W E I G H T I N G

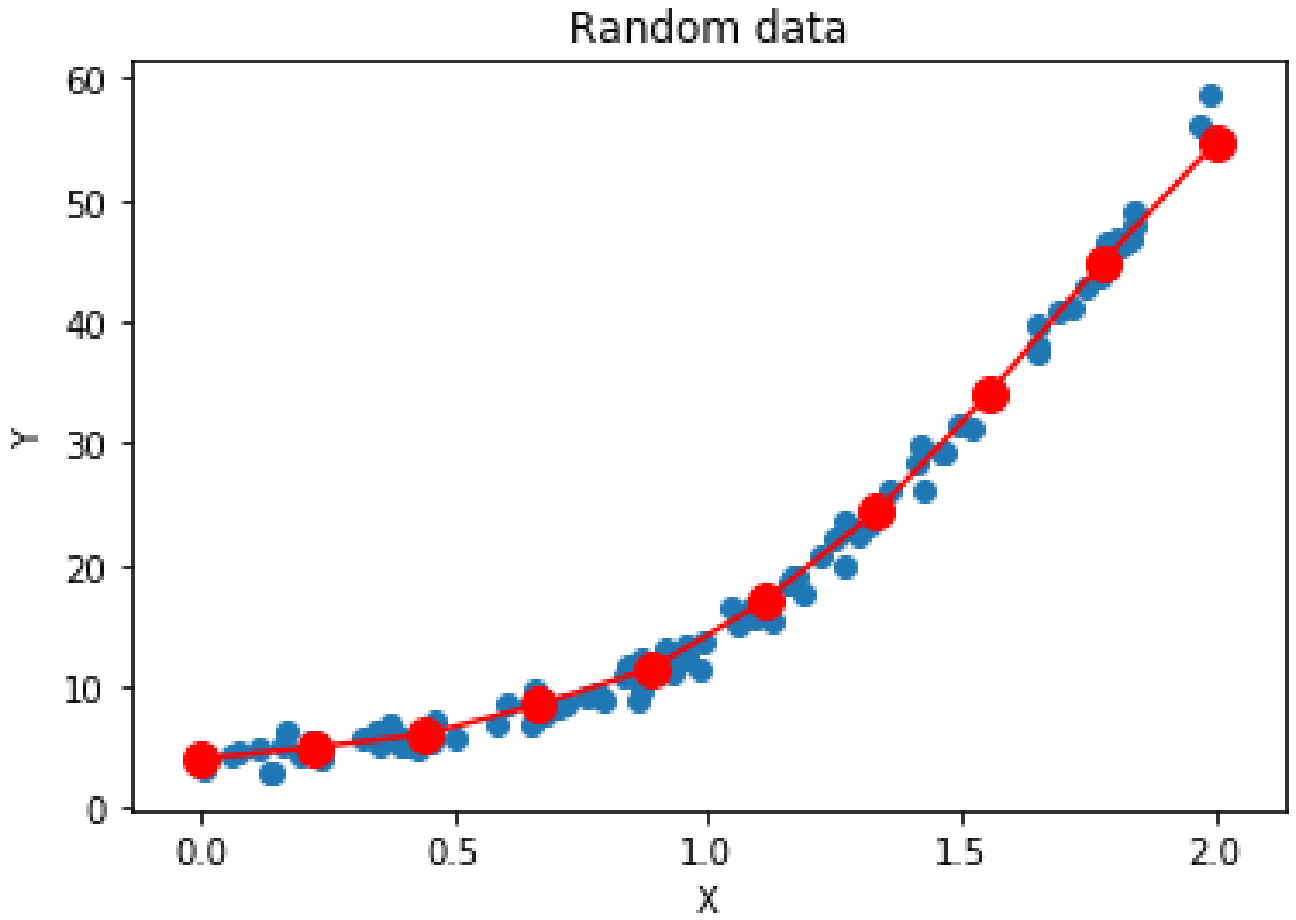
W E I G H T I N G I S S U E S

The **problem** with applying **weighted linear regression** is that each time you evaluate the **hypothesis** at a **new point** you need to run the **gradient descent** or **MLE**. Therefore, it is **very computationally expensive**.



L R B Y W E I G H T I N G

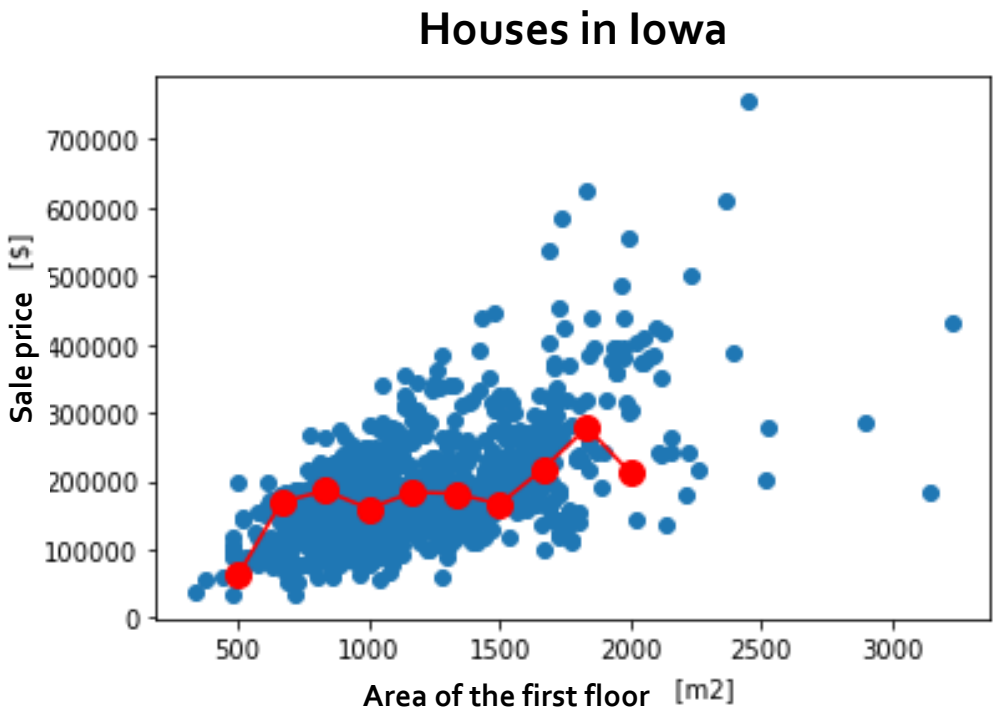
P R A C T I C A L E X A M P L E



$$\tau = 0.1$$

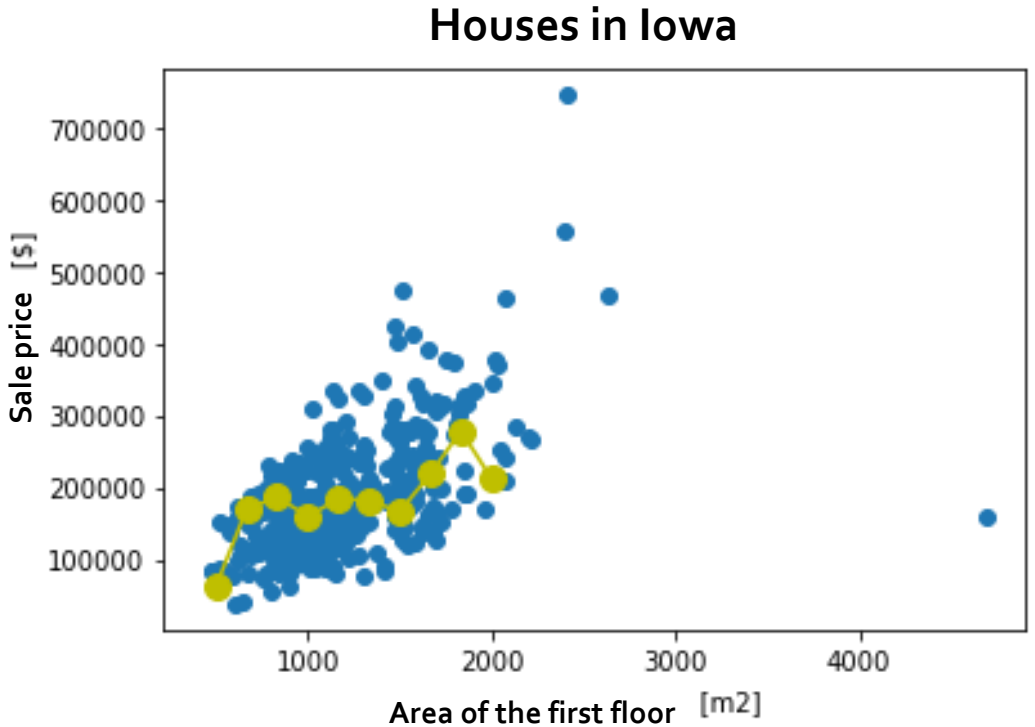
L R B Y W E I G H T I N G

H O U S E E X A M P L E



Training data

70%

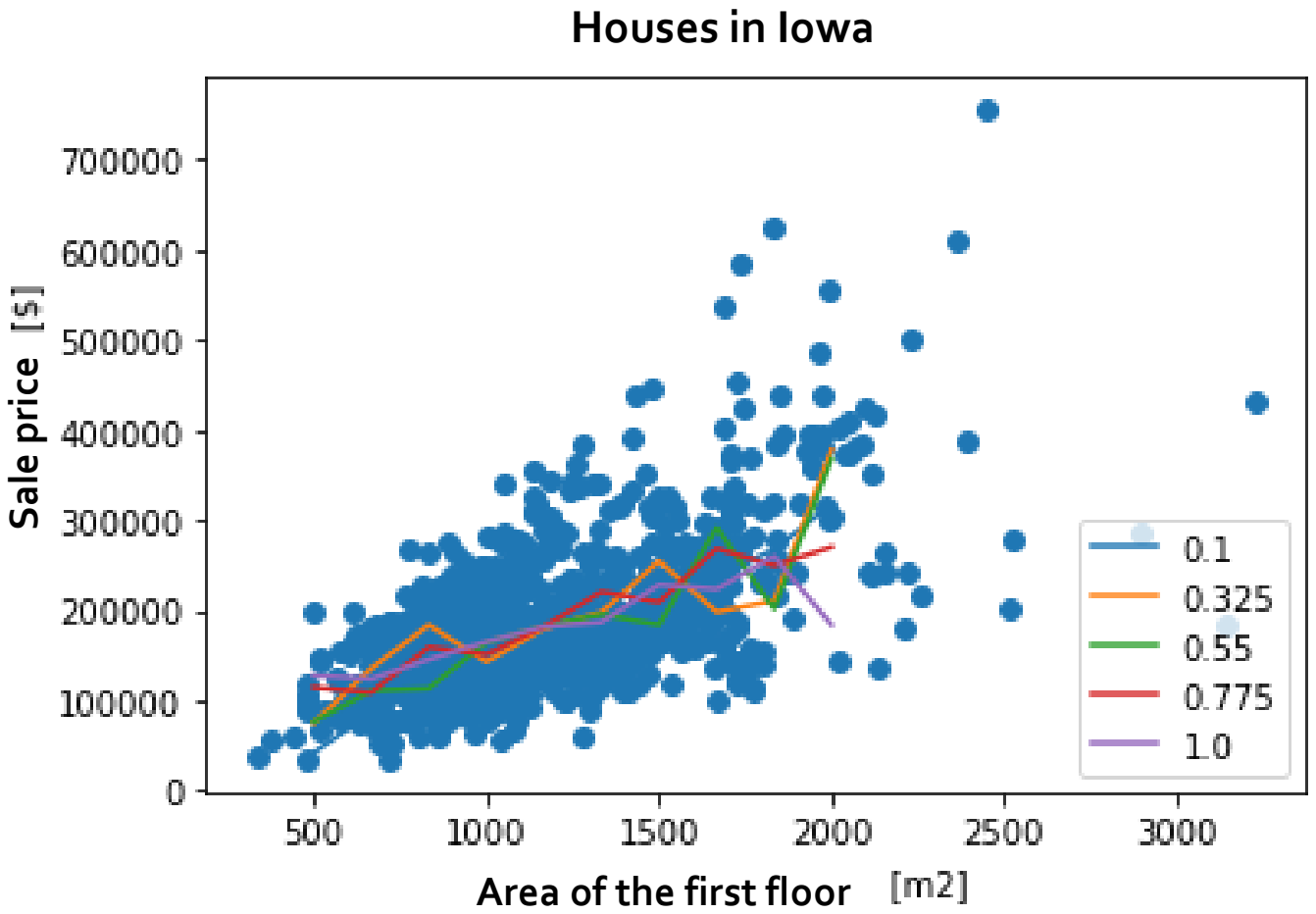


Validation data

30%

L R B Y W E I G H T I N G

H O U S E E X A M P L E



REGULARIZATION

REGULARIZATION

WEIGHT PENALIZATION

Even though **weighted local linear regression** helps us **avoid over-training** the model, it has the **disadvantage** of being an **expensive** method (**non-parametric method**).

There is another way to **reduce overtraining**: **penalize** the model for having a **very flexible hypothesis** space (many parameters $\boldsymbol{w}$). The following **cost function** is defined:

$$\boldsymbol{J(\vec{w}) = \frac{1}{2m}(X\vec{w} - \vec{y})^T(X\vec{w} - \vec{y}) + \frac{\lambda}{2m}\left(\|\vec{w}\|_q\right)^q}$$

Where $\|\vec{w}\|_q$ represents a norm $\boldsymbol{q}$ from the **vector of weights** $\vec{w}$.

Where $\boldsymbol{\lambda \geq 0}$ represents the **regularization constant** that **controls** the amount of **penalty**.

REGULARIZATION

VECTOR NORM

Remembering that:

$$\|\vec{w}\|_q = \left(\sum_{j=1}^{n} |w_j|^q \right)^{1/q}$$

We can infer:

$$\left(\|\vec{w}\|_q \right)^q = \sum_{j=1}^{n} |w_j|^q$$

REGULARIZATION

REGULARIZATION TYPES

Two types of **regularization** (decay of weights) widely used in practice for being **computationally efficient** are:

1. Lasso regularization (L1 norm)

$$\boldsymbol{J(\vec{w}) = \frac{1}{2m}(X\vec{w} - \vec{y})^T(X\vec{w} - \vec{y}) + \frac{\lambda}{2m}\|\vec{w}\|}$$

2. Tikhonov – Ridge regularization (L2 norm):

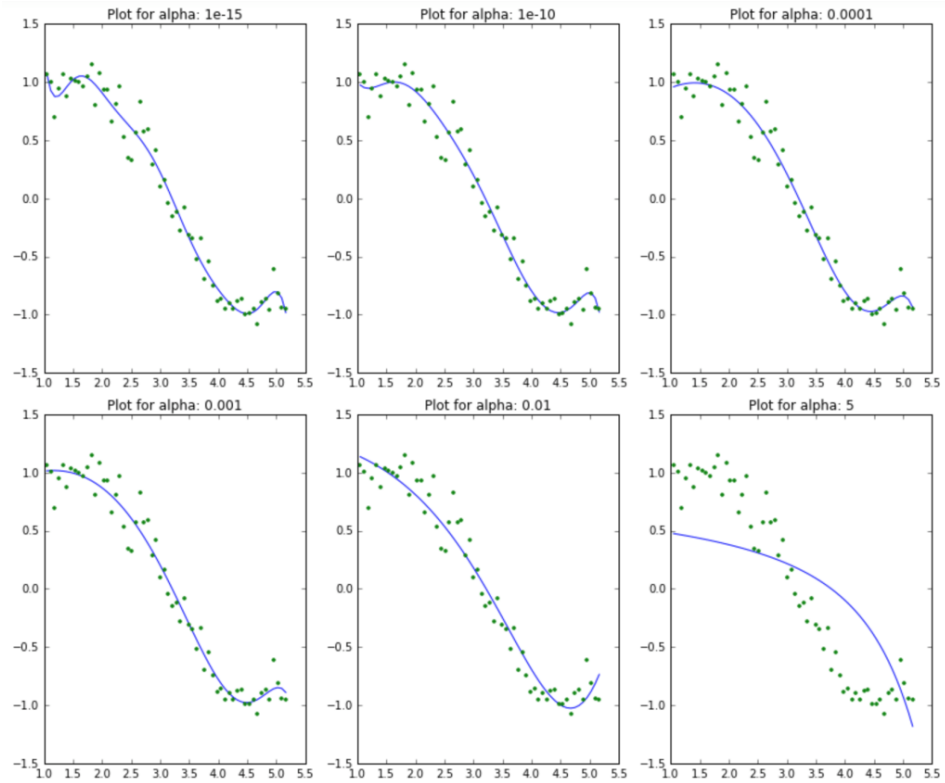
$$\boldsymbol{J(\vec{w}) = \frac{1}{2m}(X\vec{w} - \vec{y})^T(X\vec{w} - \vec{y}) + \frac{\lambda}{2m}\vec{w}^T\vec{w}}$$

REGULARIZATION

REGULARIZED MODEL OPTIMIZATION

The **normal equation** and **gradient descent optimization methods** can be applied to the **new regularized models**.

Therefore, its **implementation** in **code** is very **easy** to carry out.



K FOLD CROSS VALIDATION

K CROSS VALIDATION

REDUCED TRAINING DATA

When you have a **smaller number of data**, it is better to implement another type of cross-validation, where more data is retained:

1. **Divide** the **training data set into k sets**, each one with $\frac{m}{k}$ **training data**.

2. **Train** the model with $k-1$ **sets** and **validate** (make predictions) with the **last** k set

3. Repeat step 1 and 2**,** but the **validation set** will be **different.** In this way it is **validated** with **all sets k** and **trained** with all **sets $k-1$.**

4. All k **validation errors** are **averaged** to give a final metric.

K CROSS VALIDATION

K VALIDATION EXAMPLE