# APRENDIZAJE DE MÁQUINA

## SELECCIÓN DE MODELOS I

# AGENDA

**01** Sobreajuste vs subajuste

**02** RL por ponderación local

**03** Regularización

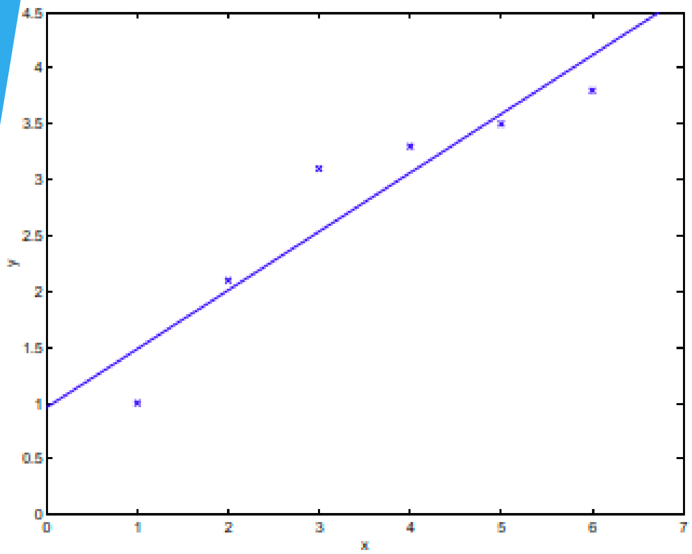
**04** Validación cruzada por k iteraciones

# SOBREAJUSTE VS SUBAJUSTE

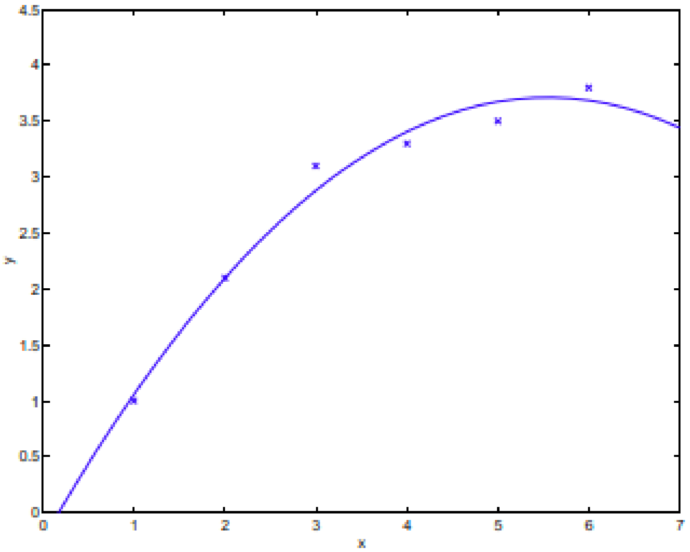
# SOBREAJUSTE VS SUBAJUSTE

## INTRODUCCIÓN

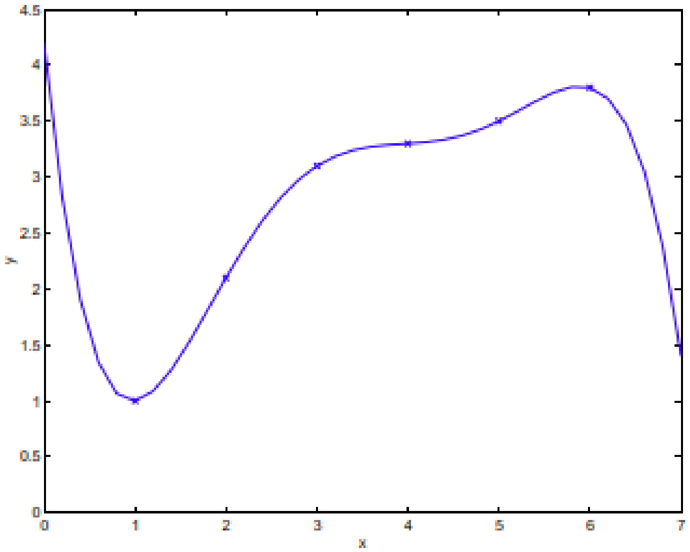
Regresando a la regresión lineal, existe el **problema** de establecer un **grado alto polinómico** o de definir un **conjunto grande** de **funciones** de **base**:



**"under-fitting"**



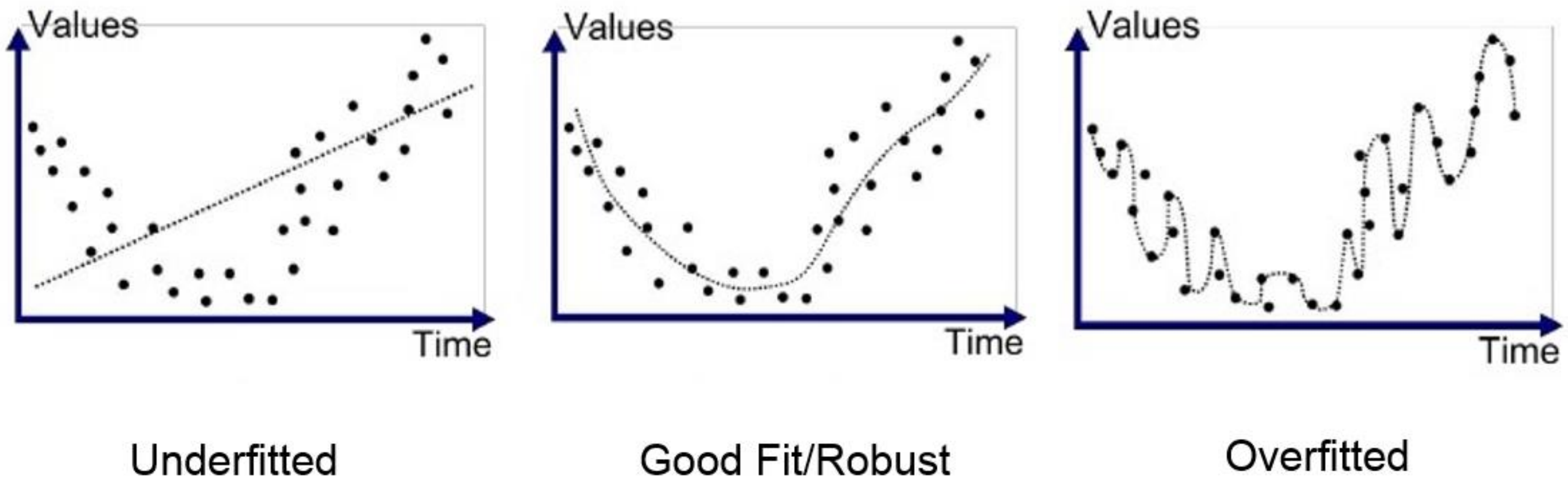
**"bien"**



**"over-fitting"**

# SOBREAJUSTE VS SUBAJUSTE

## P O S I B I L I D A D E S



# SOBREAJUSTE VS SUBAJUSTE

## VALIDACIÓN CRUZADA

Para **evaluar** un **modelo no solo basta** con **ajustar** la **hipótesis** a los **datos** de **entrenamiento**. Se debe **ver** el **poder** de **generalización** del modelo ante **nuevos datos**.

Para esto se utiliza el **método** de **validación cruzada** donde se **dividen** los **datos** en **dos conjuntos**:

- **Datos de entrenamiento: datos** que servirán para **entrenar** al **modelo**. Para un conjunto de pocos datos se añade el **70**% de los **datos.**
- **Datos de validación: datos** que servirán para **calcular predicciones** con los **parámetros obtenidos** en el **entrenamiento** del **modelo**. Para un conjunto de pocos datos se añade el **30**% de los **datos**.

# SOBREAJUSTE VS SUBAJUSTE

## PROCESO VALIDACIÓN CRUZADA

El **proceso** sería el **siguiente**:

1. **Dividir** el conjunto de **datos** en dos: **70**% para **entrenamiento** y **30**% para **validación**.

2. **Entrenar** el **modelo** con el **70**% de los datos y **obtener** los **mejores pesos** $\vec{w}$.

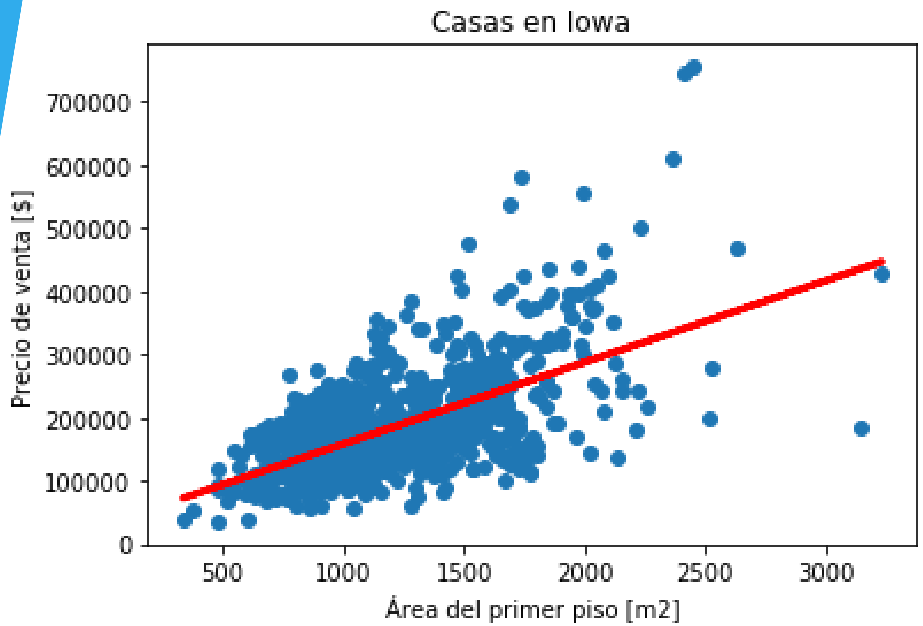
3. Con los **pesos** $\vec{w}$ **calcular** las **predicciones** para el **30**% de los **datos** restantes.

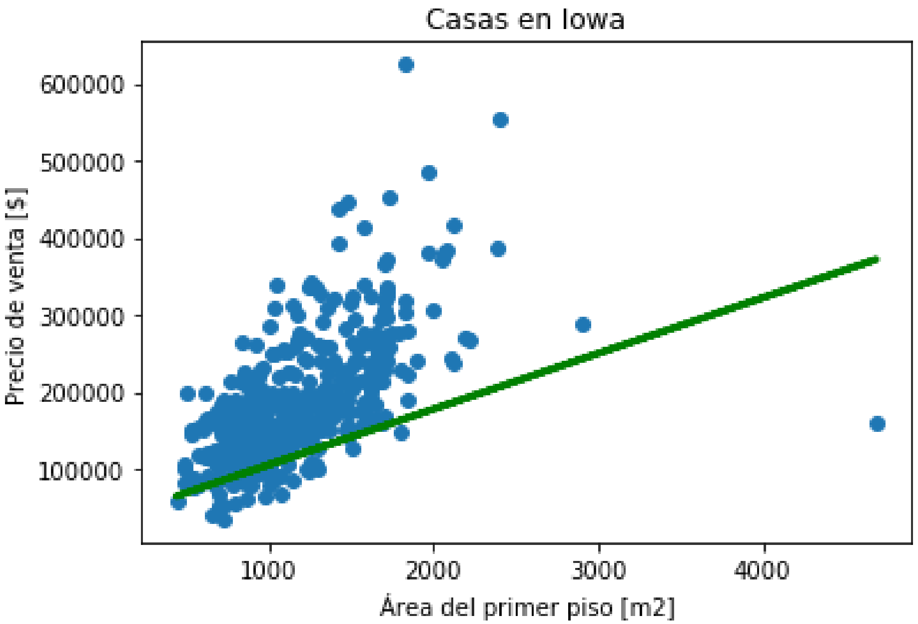
4. **Computar** las **funciones** de **costo** para los pasos **2** y **3.**

5. **Repetir** el **proceso** con **otra hipótesis** (modelo) **hasta seleccionar** el **modelo** con los **mejores errores** de **validación** y **entrenamiento**.

# SOBREAJUSTE VS SUBAJUSTE

## ECUACIONES NORMALES



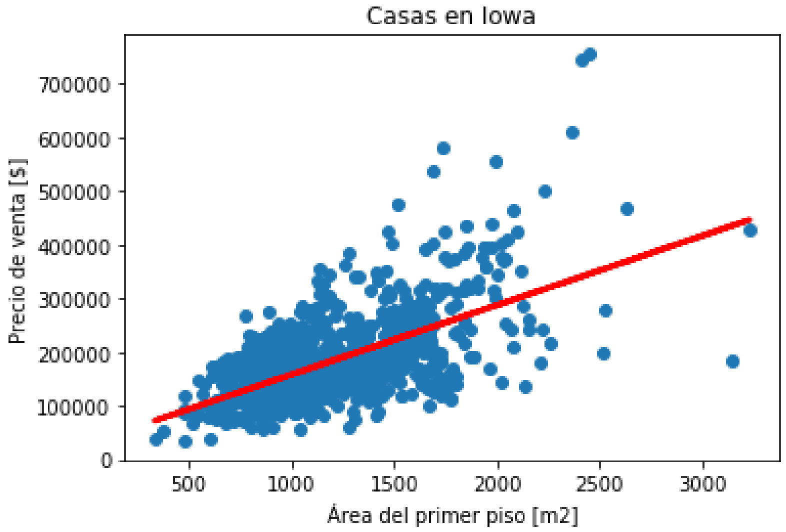



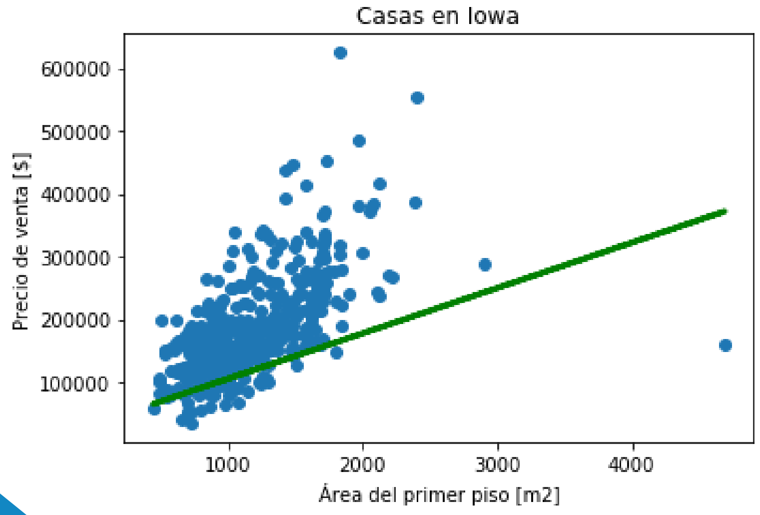

**Datos de entrenamiento 70%**

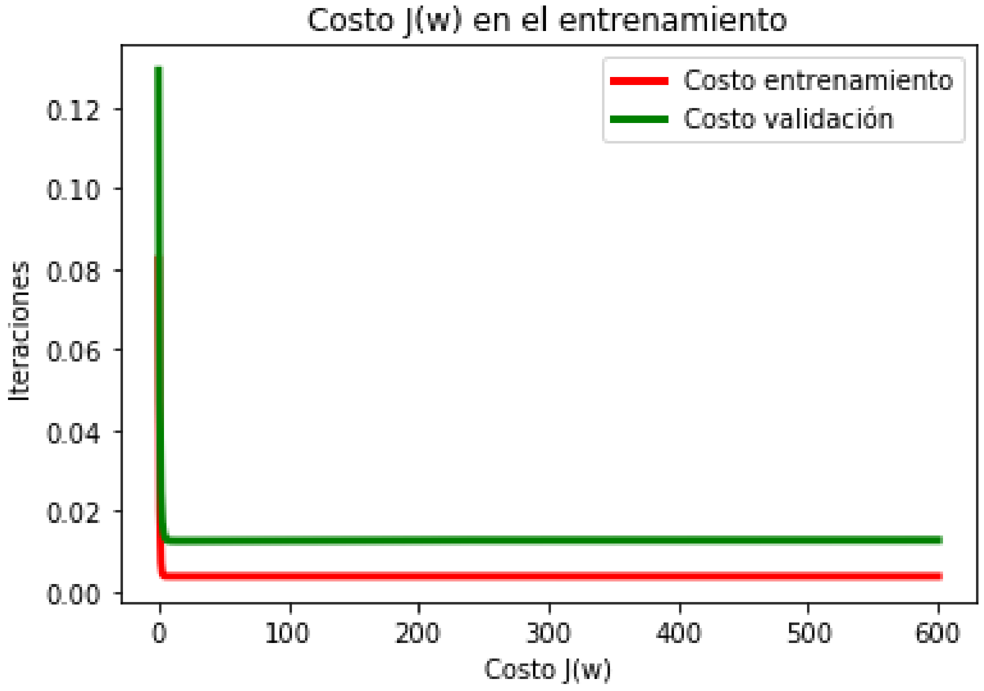
**Datos de validación 30%**

# SOBREAJUSTE VS SUBAJUSTE

## DESCENSO POR GRADIENTE

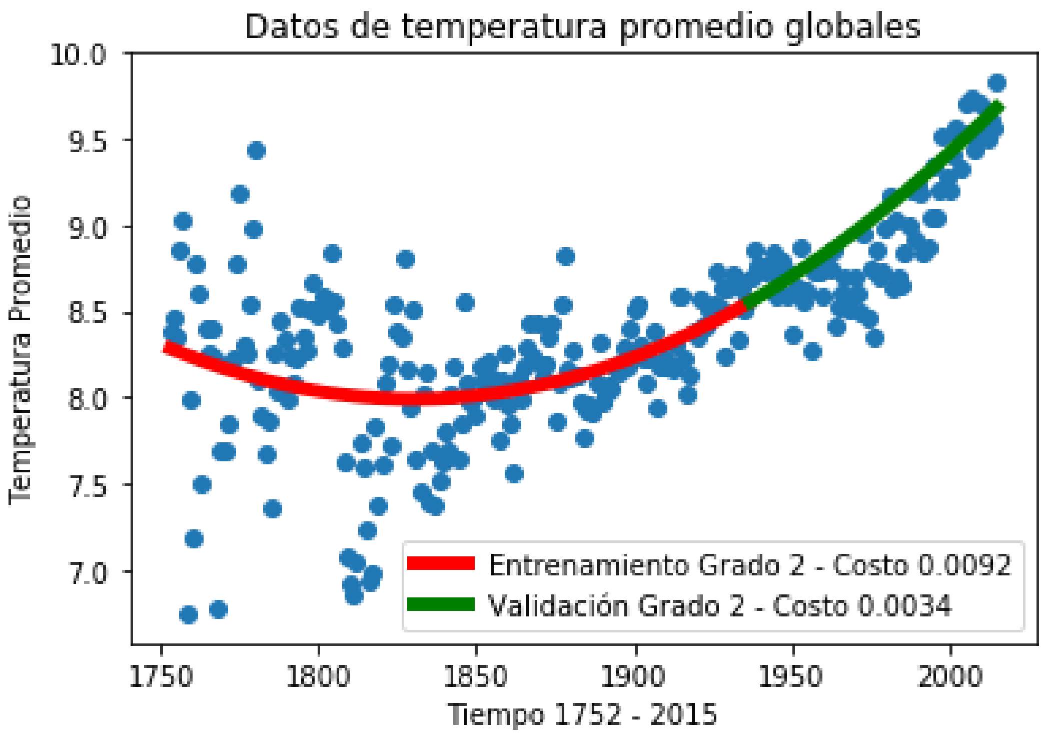


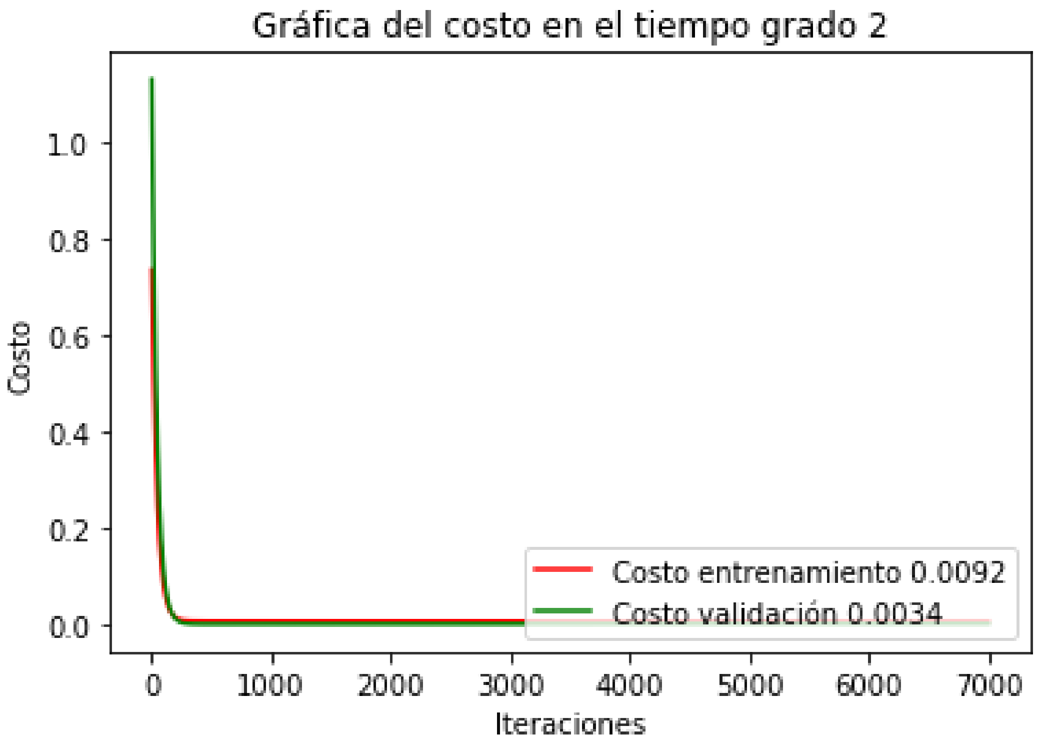



# SOBREAJUSTE VS SUBAJUSTE

## ENTRENAMIENTO IDEAL

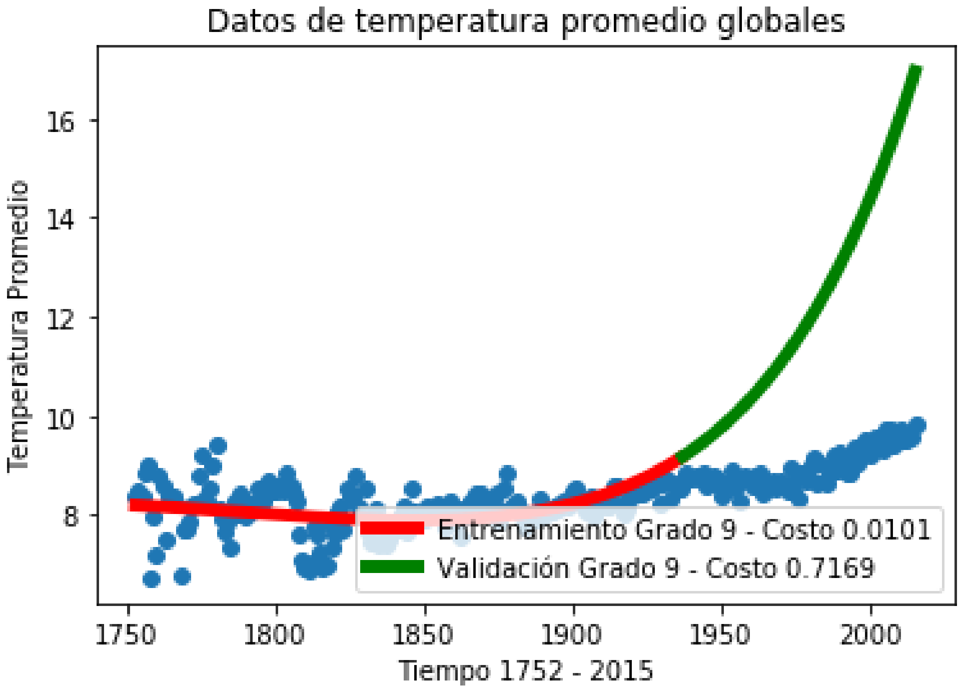


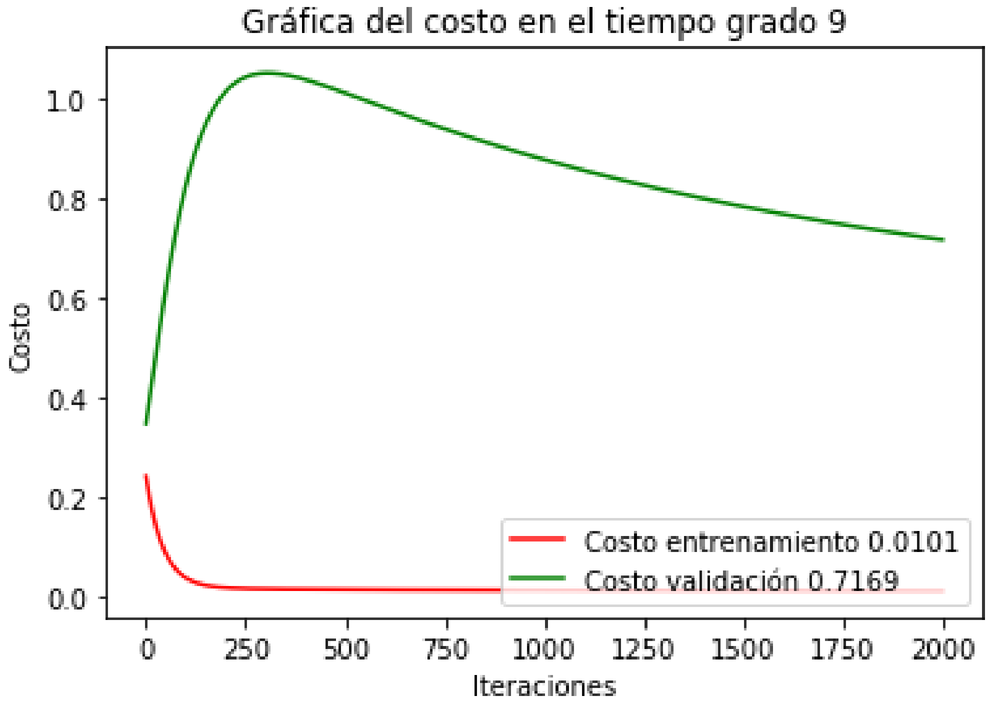




**GRADO 2**

# SOBREAJUSTE VS SUBAJUSTE

## SOBRE-ENTRENAMIENTO




**GRADO 9**

# SOBREAJUSTE VS SUBAJUSTE

## D E F I N I C I O N E S

De manera **formal** se puede **definir** lo siguiente:

- **Subajuste:** el **espacio** de la **hipótesis** está **demasiado restringido** para capturar la **estructura relevante** de la **función** que está siendo **modelada**.

- **Sobreajuste:** el espacio de la **hipótesis** es lo **suficientemente grande** que es posible **modelar ruido** en los datos de entrenamiento lo que **deriva** en una **estructura no generalizable**.

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

# SOBREAJUSTE VS SUBAJUSTE

## CAUSAS DEL SOBREAJUSTE

**Existen** diferentes **causas** para un **sobre entrenamiento**:

1. **Pocos datos**: puede haber **patrones falsos** que no existirían con un mayor conjunto de datos.

2. **Ruido** en los **datos**: la **proporción señal-ruido** es **muy mala** en los datos, por lo que es **difícil encontrar** la **verdadera estructura** que se quiere **aprender**.

3. El **espacio** de la **hipótesis** es muy **grande**: entre más **flexibilidad** tenga la **hipótesis**, existe **mayor posibilidad** de ajustar **patrones falsos**.

4. El **espacio** de **entradas** es **altamente dimensional**: añadir características (**más funciones** de **base**) que **no son importantes** puede introducir **ruido no deseado**. De aquí la importancia de la **selección** de **características**.

# SOBREAJUSTE VS SUBAJUSTE

## R E S U M E N

El **problema** se **resume** así:

**AJUSTAR DE LA MEJOR MANERA LOS DATOS DE ENTRENAMIENTO SIN PERDER EL PODER DE GENERALIZACIÓN**

# RL POR PONDERACIÓN LOCAL

# RL POR PONDERACIÓN

## FUNCIÓN DE PONDERACIÓN

Por lo tanto, la **selección** de las **características** $\boldsymbol{X}$ es muy **importante** para hacer una **representación** adecuada de los **datos**.

Para **aliviar** un poco el **problema** de **selección**, se propone un **modelo** de **regresión** por **ponderación local**. Un método **no paramétrico** (el **número** de **parámetros** crece con el **número** de **datos** de entrada $\boldsymbol{m}$**).**

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

Donde $\boldsymbol{\theta}^{(i)}$ representa una **función** de **ponderación local**.

# RL POR PONDERACIÓN

## FUNCIÓN DE PONDERACIÓN

La **función** de **ponderación local** es una **elección arbitraria**. Por lo general, se utiliza la siguiente **función,** donde $\boldsymbol{\tau}^2$ se le denomina como **ancho** de **banda**:

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

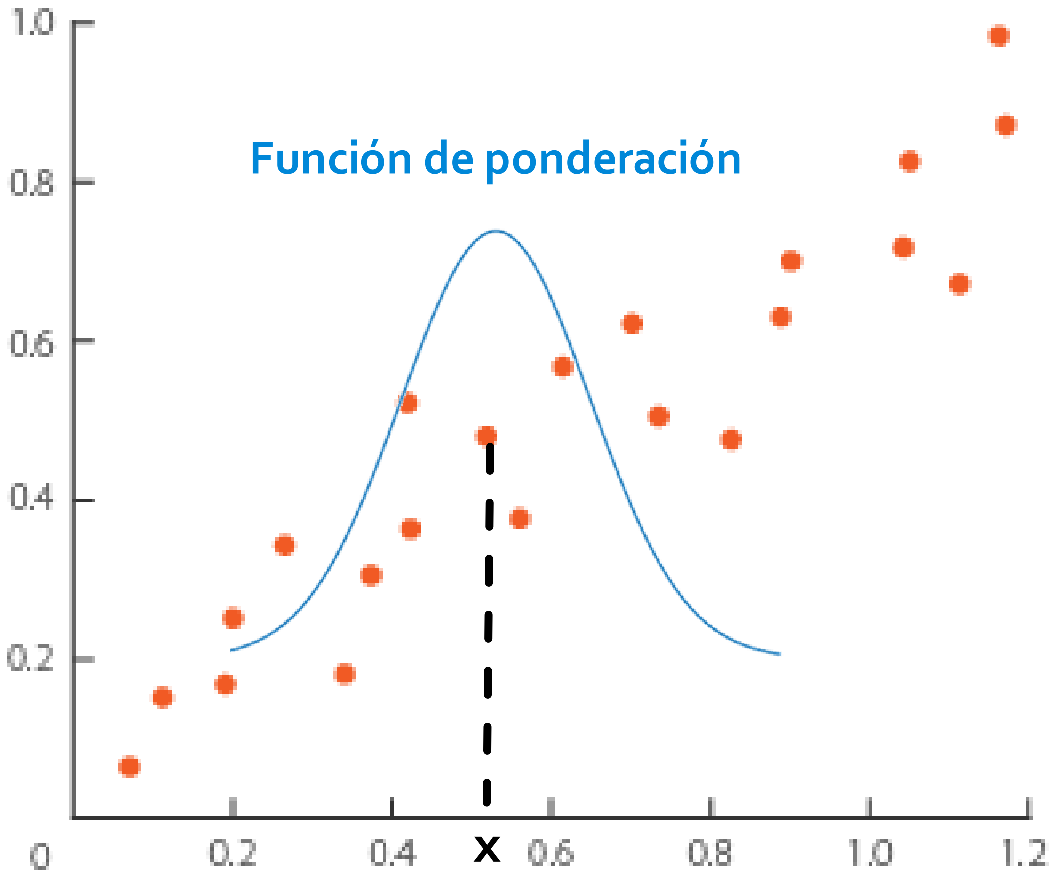
Sí $\left|\boldsymbol{x}^{(i)} - \boldsymbol{x}\right| \rightarrow 0$ es pequeño $\boldsymbol{\theta}^{(i)} \rightarrow \mathbf{1}$

Sí $\left|\boldsymbol{x}^{(i)} - \boldsymbol{x}\right| \rightarrow \infty$ es pequeño $\boldsymbol{\theta}^{(i)} \rightarrow \mathbf{0}$

En otras palabras, entre los datos $\boldsymbol{x}^{(i)}$ se alejen más del dato $\boldsymbol{x}$ menos importancia van a tener para la regresión.

# RL POR PONDERACIÓN

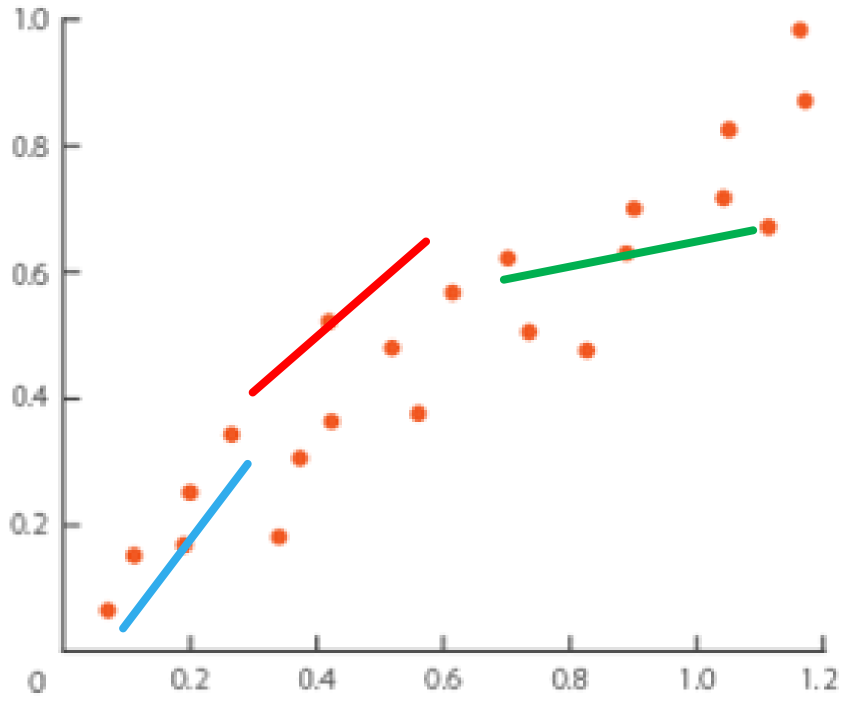
## FUNCIÓN DE PONDERACIÓN



# RL POR PONDERACIÓN

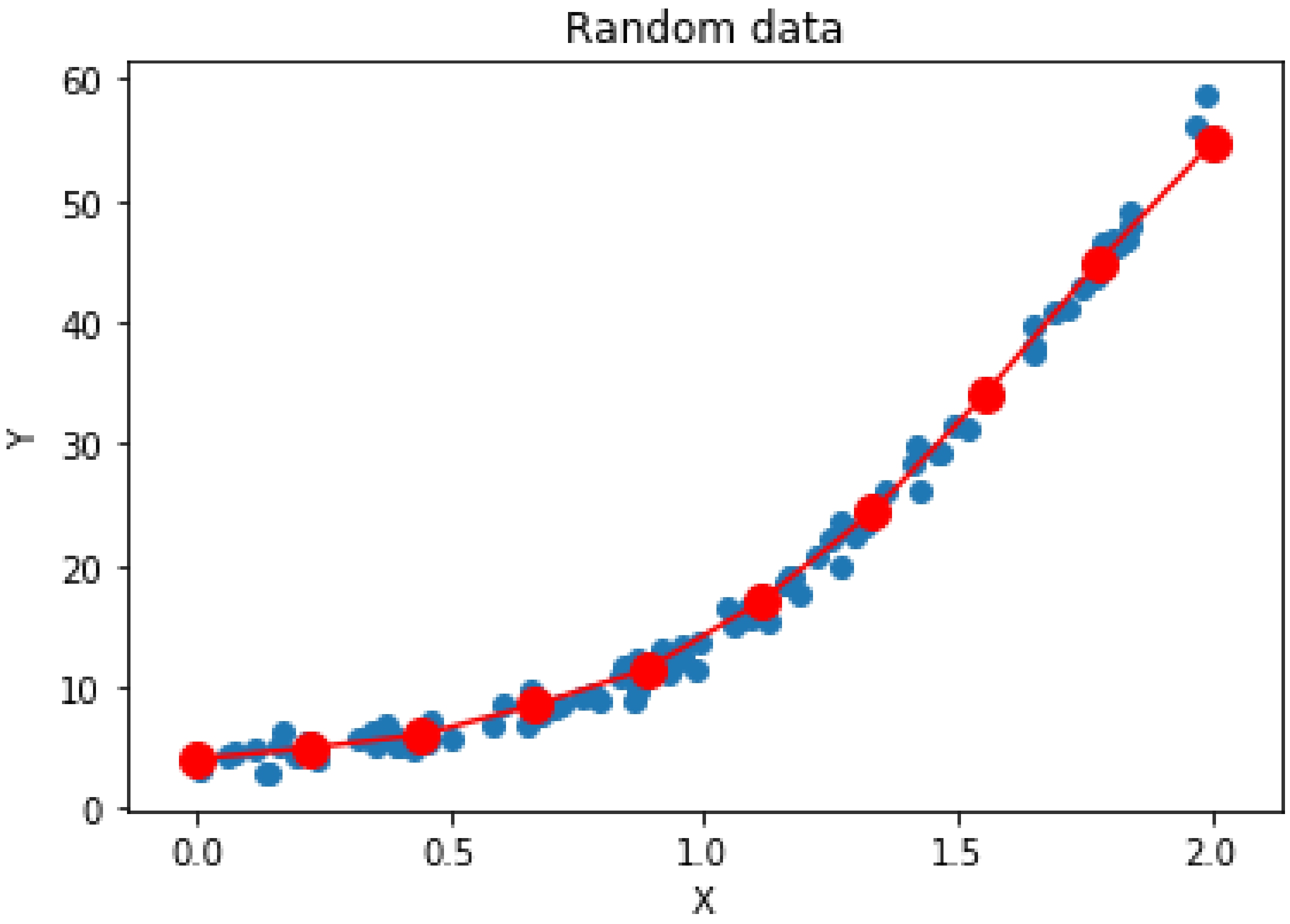
## PROBLEMA CON LA PONDERACIÓN

El **problema** al **aplicar** la **regresión lineal ponderada**, es que **cada vez** que **evalúas** la **hipótesis** en un **nuevo punto** necesitas **correr** el **descenso por gradiente** o **MLE**. Por lo tanto, es **muy costoso computacionalmente**.



# RL POR PONDERACIÓN

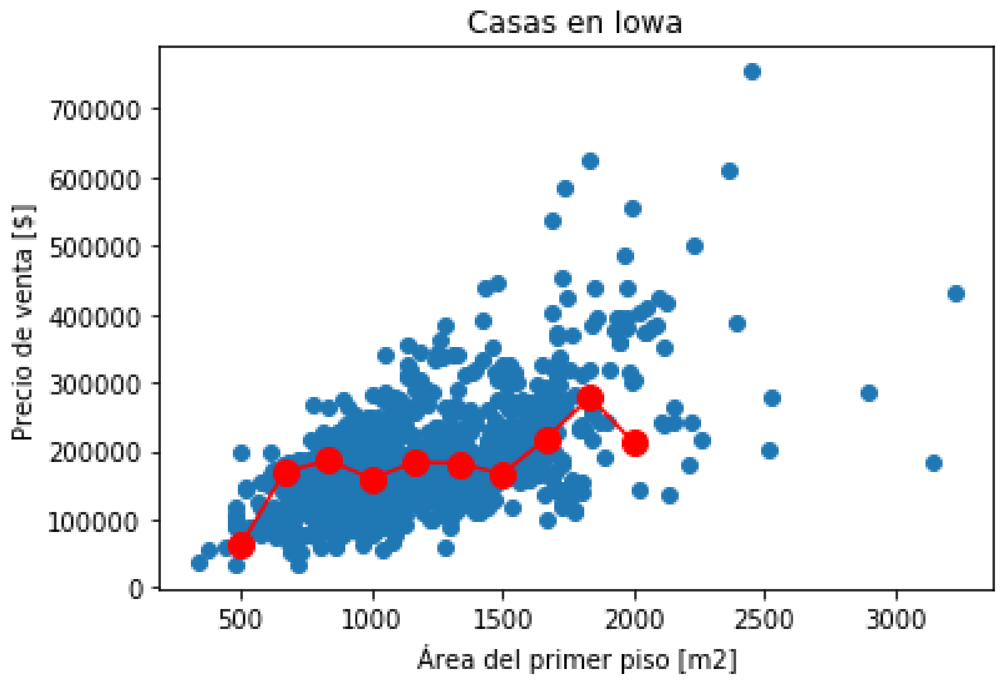
## EJEMPLO PRÁCTICO



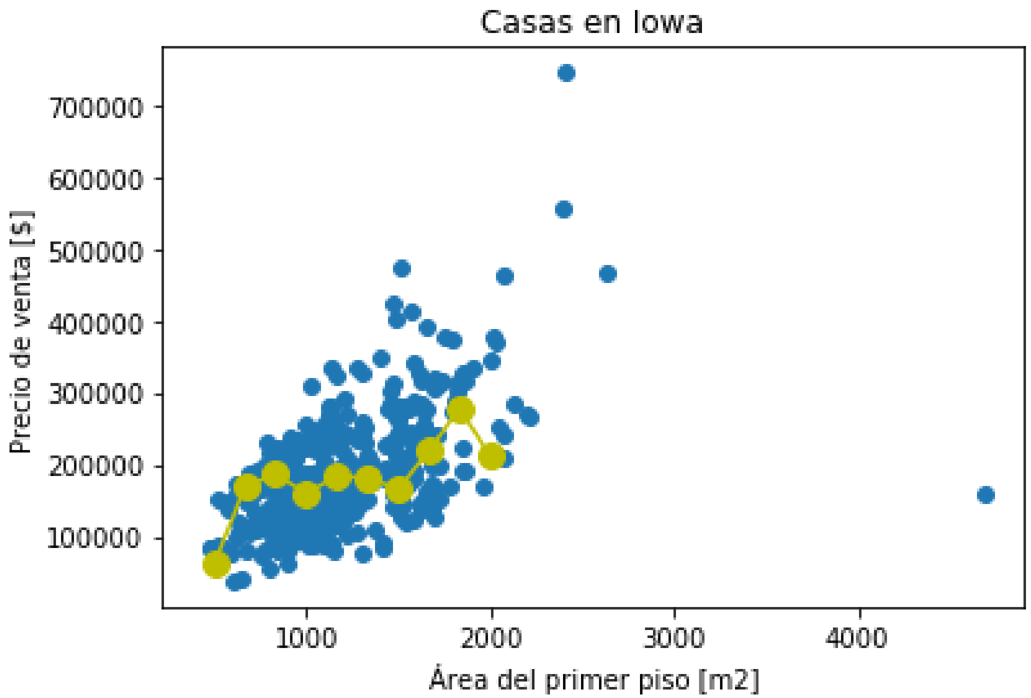

$$\tau = 0.1$$

# RL POR PONDERACIÓN

## EJEMPLO CASAS



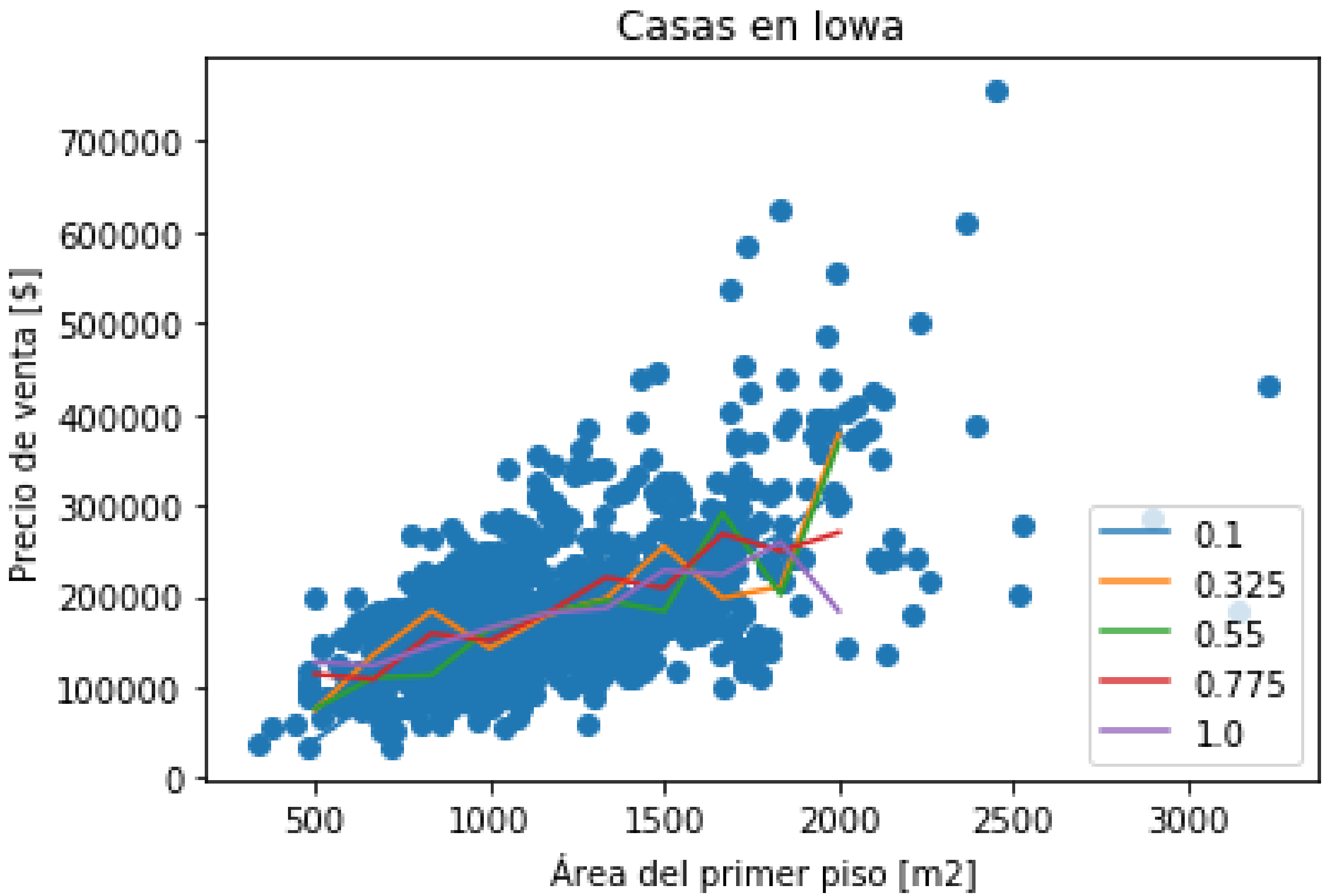

**Datos de entrenamiento 70%**




**Datos de validación 30%**

# RL POR PONDERACIÓN

## EJEMPLO CASAS



AI

# REGULARIZACIÓN

# R E G U L A R I Z A C I Ó N

## P E N A L I Z A C I Ó N  D E  P E S O S

Aún cuando la **regresión lineal local ponderada** nos ayuda a evitar el **sobre-entrenamiento** del modelo, pero tiene la desventaja de ser un **método costoso** (método **no parámetrico**).

Existe otra forma de **reducir** el **sobre entrenamiento**: **penalizar** al **modelo** por tener un **espacio** de **hipótesis** muy **flexible** (muchos **parámetros** $\boldsymbol{w}$). Se define la siguiente **función** de **costo**:

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

Donde $\|\vec{\boldsymbol{w}}\|_{\boldsymbol{q}}$ representa una la **norma $\boldsymbol{q}$ del vector de pesos $\vec{\boldsymbol{w}}$**.

Donde $\boldsymbol{\lambda} \geq \boldsymbol{0}$ representa la **constante** de **regularización** que **controla** la cantidad de **penalización**.

# REGULARIZACIÓN

## NORMA VECTORIAL

Recordando que:

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

Por lo que:

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

# R E G U L A R I Z A C I Ó N

## TIPOS DE REGULARIZACIÓN

Dos tipos de **regularización** (decaimiento de pesos) muy **usadas** en la **práctica** por ser **eficientes computacionalmente** son:

1. Regularización Lasso (norma L1)

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

2. Regularización de Tikhnonov – Ridge (norma L2):

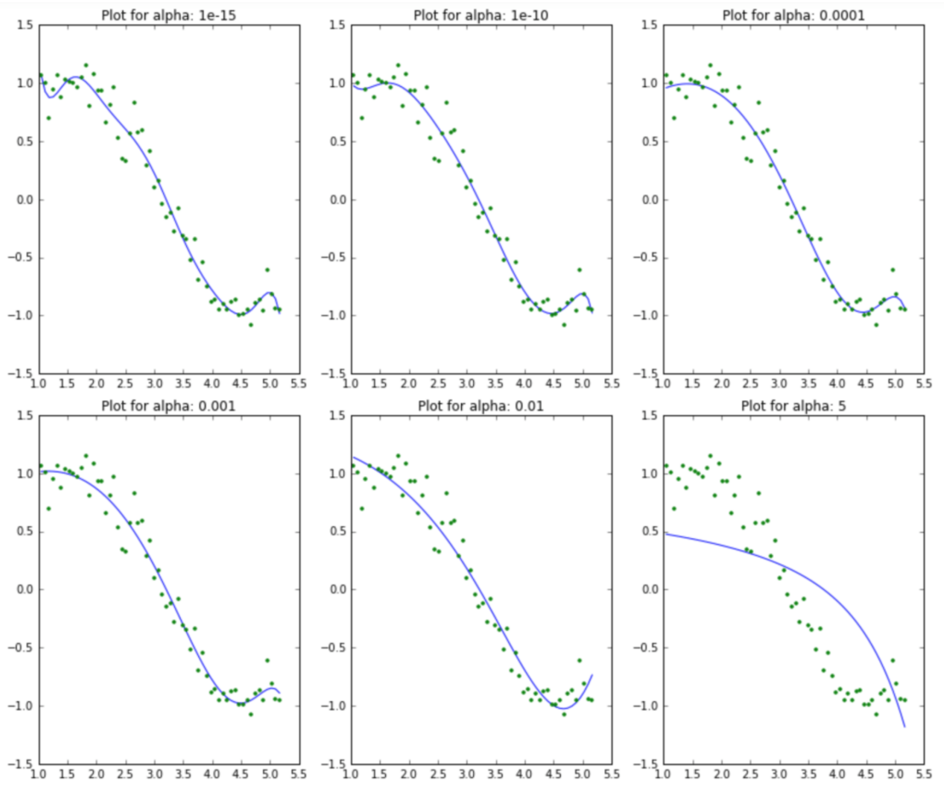
$$J(\vec{w}) = \frac{1}{2m}(X\vec{w} - \vec{y})^T(X\vec{w} - \vec{y}) + \frac{\lambda}{2m}\vec{w}^T\vec{w}$$

# R E G U L A R I Z A C I Ó N

## OPTIMIZACIÓN DE MODELOS REGUALRIZADOS

Los **métodos** de **optimización** de **ecuaciones normales** y **descenso por gradiente** se pueden **aplicar** a los **nuevos modelos regularizados**.

Por lo tanto, su **implementación** en **código** es **muy sencilla** de realizar.



# VALIDACIÓN CRUZADA POR K ITERACIONES

# VALIDACIÓN CRUZADA K

## POCOS DATOS DE ENTRENAMIENTO

Cuando se tienen un **número menor** de datos es mejor implementar otro tipo de validación cruzada, donde se retienen más los datos:

1. **Dividir** el **conjunto** de **datos** de entrenamiento en $\boldsymbol{k}$ **conjuntos**, cada uno con $\boldsymbol{\frac{m}{k}}$ **datos** de **entrenamiento**.

2. **Entrenar** el modelo con $\boldsymbol{k-1}$ **conjuntos** y **validarlo** (realizar predicciones) con el **último conjunto** $\boldsymbol{k}$.

3. **Repetir** el **paso 1** y **2**, pero el conjunto de validación será otro. De esta manera se **valida** con todos los **conjuntos** $\boldsymbol{k}$ y se **entrena** con **todos** los **conjuntos** $\boldsymbol{k-1}$.

4. Se **promedian todos** los $\boldsymbol{k}$ **errores** de **validación** para dar una **métrica final**.

# VALIDACIÓN CRUZADA K

## EJEMPLO DE VALIDACIÓN K