Introduction to Regression

Agenda

- Linear Regression
- Cost Functions
- Gradient Descent
- Hands-on Example
- Evaluating Regression Models
- Regularization

Agenda

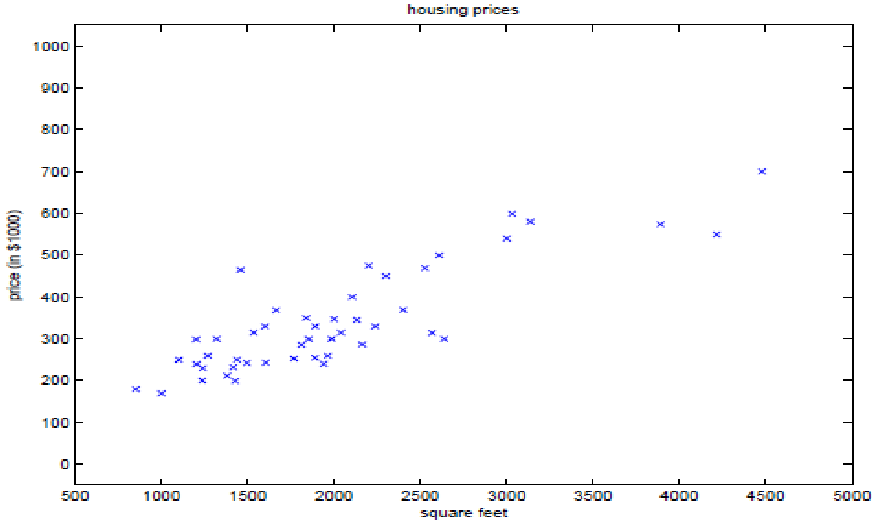
- **Linear Regression**
- Cost Functions
- Gradient Descent
- Hands-on Example
- Evaluating Regression Models
- Regularization

Regression vs Classification

- Regression: Target is continuous or ordinal
  - Example: Predict price, weight, height, temperature, ranking
- Classification : Target is discrete with finite value set
  - Example: Predict survived/dead, face/non-face, fraud/non-fraud, bot/non-bot

Example: Housing Prices

Living area ($feet^2$)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

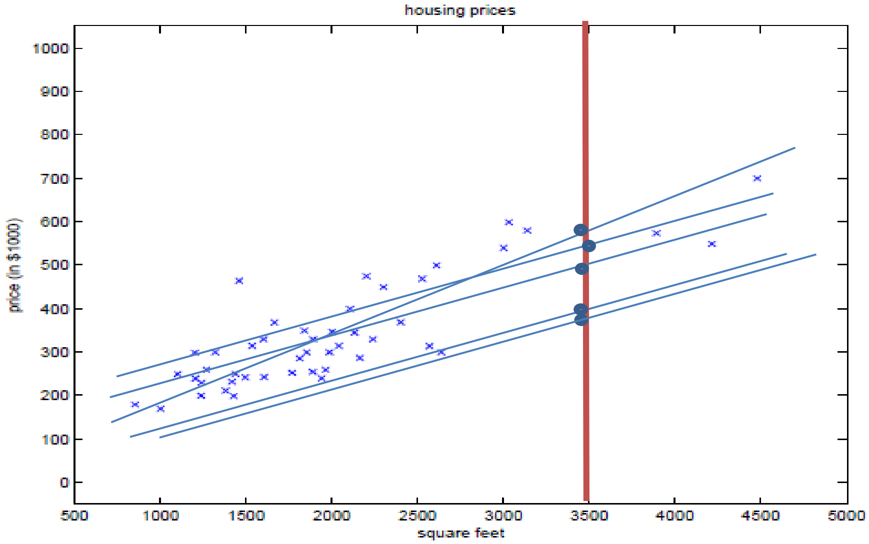


Can we learn to predict the price for a house not in the dataset?

Living Area = 3500 sq. ft. **Price = ?**

Example: Housing Prices

- Living Area = 3500 sq. ft.
- Price = ?
- Use the line that is somewhere in the middle
- How do we define "middle"?



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

datasciencedojo
unleash the data scientist in you

Notation

- x^i – The attribute vector for the ith data object
- X – The set of all x^i
- y^i – The true value for the ith data object
- Y – The set of all y^i
- m – Number of rows in the dataset
- n – Number of columns in the dataset

Notation

- h_θ – A specific hypothesis (estimate) for the map X → Y
- θ – A vector of model parameters which define a specific hypothesis

Linear Regression Hypothesis Function

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

- $h_\theta(x^i)$ is the prediction of the hypothesis h_θ for a specific object x^i
- y^i is the true target value (equivalent to label in classification)

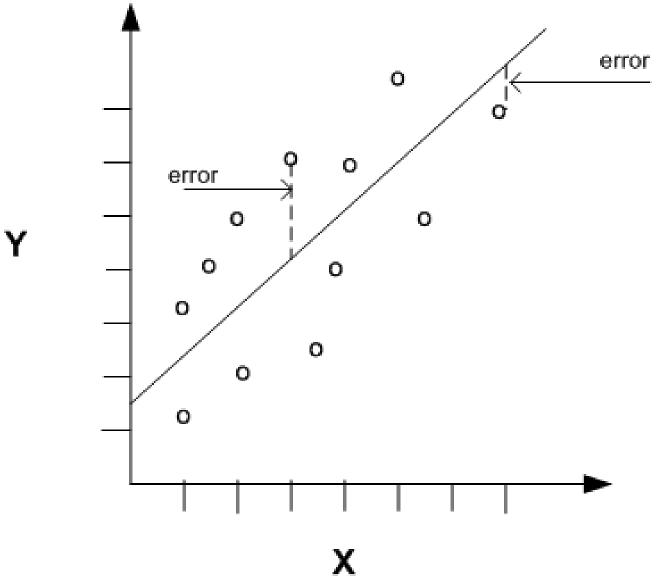
- We want the prediction $h_\theta(x^i)$ to be as close to the true label y^i as possible.
- How do we do that?

Agenda

- Linear Regression
- **Cost Functions**
- Gradient Descent
- Hands-on Example
- Evaluating Regression Models
- Regularization

Cost Function

- Define a "cost" or "loss" function $J(\theta)$
  - Smaller for lower error
  - Larger for higher error
- Many different types
- Most common: "Sum of squared residuals"

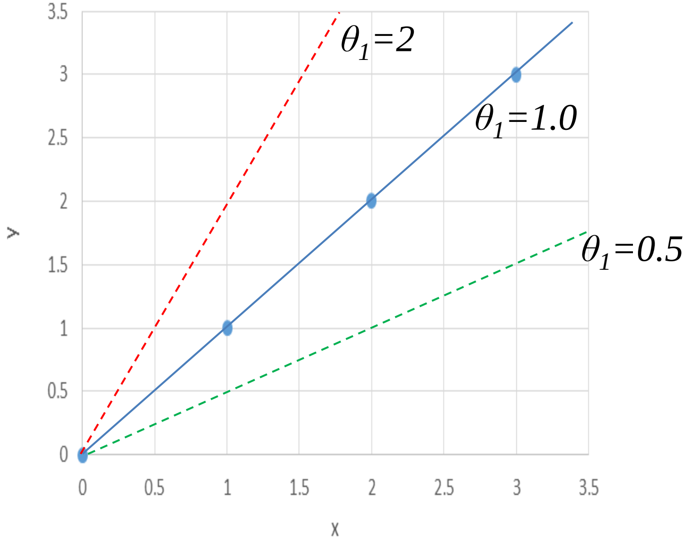


$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^i) - y^i\right)^2$$

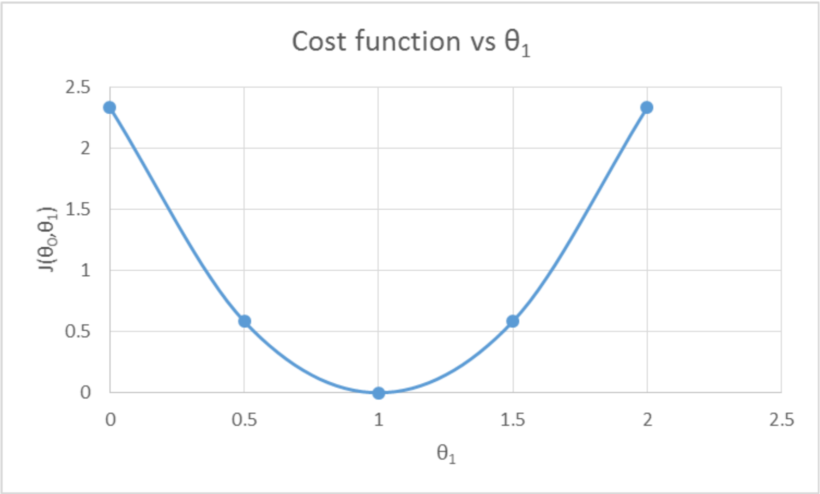
Cost Function – 2D

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

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^i) - y^i\right)^2$$



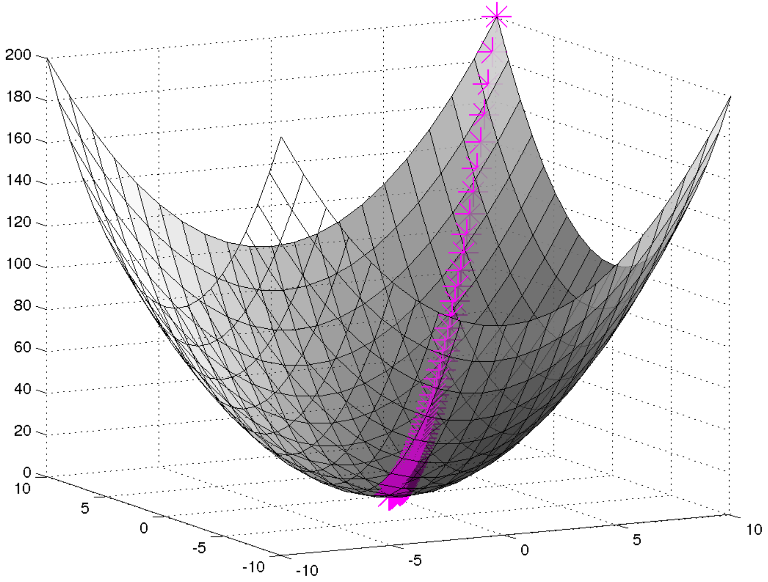
θ_0=0



The Cost Function – 3D

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

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^i) - y^i\right)^2$$



Agenda

- Linear Regression
- Cost Functions
- **Gradient Descent**
- Hands-on Example
- Evaluating Regression Models
- Regularization

Maximum/Minimum Problem

Find two nonnegative numbers whose sum is 9 and so that the product of one number and the square of the other number is a maximum.

Solution

Sum of number is 9

$$9 = x + y$$

Product of two numbers is

$$P = x\,y^2$$

$$= x\,(9 - x)^2$$

datasciencedojo
unleash the data scientist in you

Solution

$$P' = x\,(2)\,(9-x)(-1) + (1)\,(9-x)^2$$

$$= (9-x)\,[\,-2x + (9-x)\,]$$

$$= (9-x)\,[\,9-3x\,]$$

$$= (9-x)\,(3)[\,3-x\,]$$

$$= 0$$

$x=9$ or $x=3$

datasciencedojo

unleash the data scientist in you

Gradient Descent

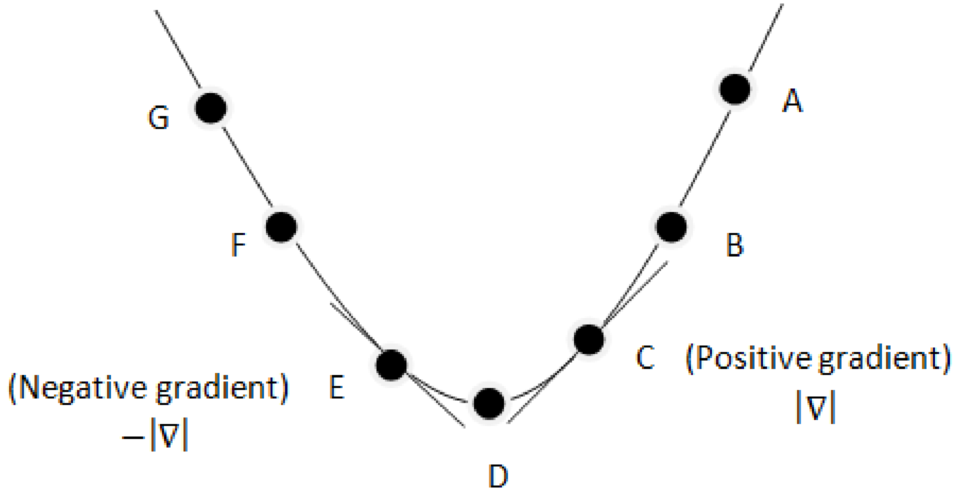
- Goal : minimize $J(\theta)$
- Gradient descent starts with some initial θ and then performs an update on each θ_j in turn:

$$\theta_j^{k+1} := \theta_j^k - \alpha \frac{\partial}{\partial \theta_j} J(\theta^k)$$

- Repeat until θ converges

Intuition

$$\theta_j^{k+1} \coloneqq \theta_j^k - \alpha \frac{\partial}{\partial \theta_j} J(\theta^k)$$



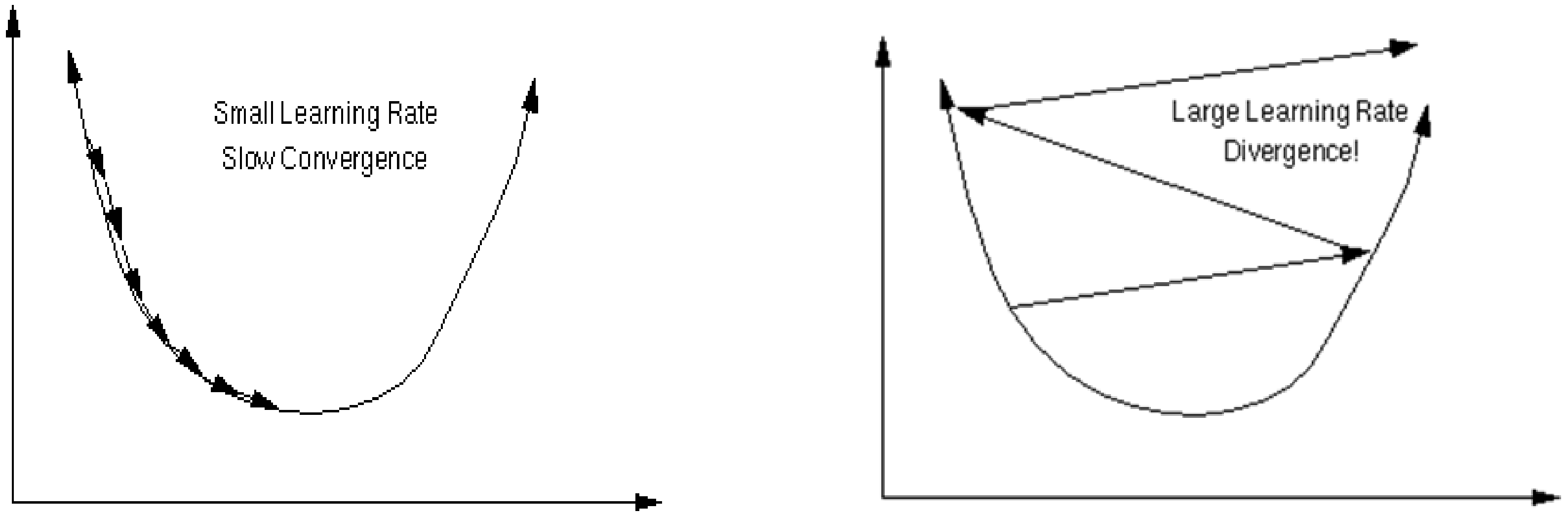
Gradient Descent Algorithm

$$\theta_j^{k+1} := \theta_j^k - \alpha \frac{\partial}{\partial \theta_j} J(\theta^k)$$

- α is known as the learning rate; set by user
- Each time the algorithm takes a step in the direction of the steepest descent and $J(\theta)$ decreases.
- α determines how quickly or slowly the algorithm will converge to a solution

Gradient Descent Algorithm

$$\theta_j^{k+1} := \theta_j^k - \alpha \frac{\partial}{\partial \theta_j} J(\theta^k)$$



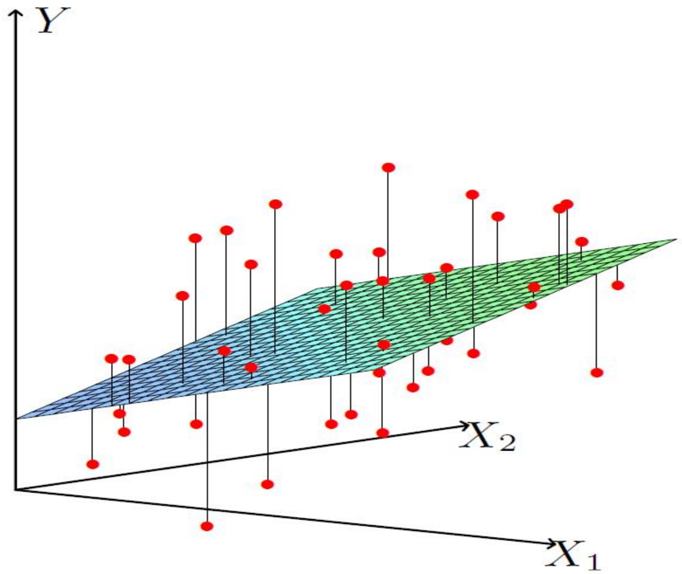
Regression with > 1 feature

$x^i{}_1$ Living area (feet^2)	$x^i{}_2$ #bedrooms	y^i Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

A linear function is just one of the choices to approximate the target variable!

Regression with > 1 feature



When we have two features, the linearity is in plane instead of line.

With three features, it's a hyperplane, and so on.

Batch Gradient Descent

For ONE training example, we get the following update rule:

$$\theta_j^{k+1} \coloneqq \theta_j^k + \alpha\big(y - h_\theta(x)\big)x_j$$

What do we observe here about the magnitude of the update?

For ALL training examples, we get the following update rule:

Repeat until convergence {

For j from 0 to n:

$$\theta_j^{k+1} \coloneqq \theta_j^k + \frac{\alpha}{m}\sum_{i=1}^{m}\left(y^i - h_\theta\left(x^i\right)\right)x_j^i$$

}

datasciencedojo

unleash the data scientist in you

Stochastic Gradient Descent

- Consider an alternative algorithm:

Repeat until convergence {
for i from 1 to m:
for j from 0 to n:

$$\theta_j^{k+1} := \theta_j^k + \alpha\left(y^i - h_\theta(x^i)\right)x_j^i$$

}

- This algorithm updates the parameters θ_j using each training example instead of all training examples.
- If the training set is big i.e., m is large, this technique converges quicker than batch gradient descent.
- Stochastic gradient descent may oscillate around the minimum of $J(\theta)$ and may not completely converge

Batch vs. Stochastic Gradient Descent

Batch Gradient Descent

Repeat until convergence {

For j from 0 to n:

$$\theta_j^{k+1} := \theta_j^k + \frac{\alpha}{m}\sum_{i=1}^{m}\left(y^i - h_\theta\left(x^i\right)\right)x_j^i$$

}

- To update each parameter value, scan through the whole training data
- Converges to the minimum value slowly
- Preferred for small datasets

Stochastic Gradient Descent

Repeat until convergence {

for i from 1 to m:

for j from 0 to n:

$$\theta_j^{k+1} := \theta_j^k + \alpha\left(y^i - h_\theta\left(x^i\right)\right)x_j^i$$

}

- Update the parameter values with one training example at a time
- Converges to the ‘proximity’ of minimum value fast but may keep oscillating near the minimum
- Preferred for large datasets

datasciencedojo
unleash the data scientist in you

Agenda

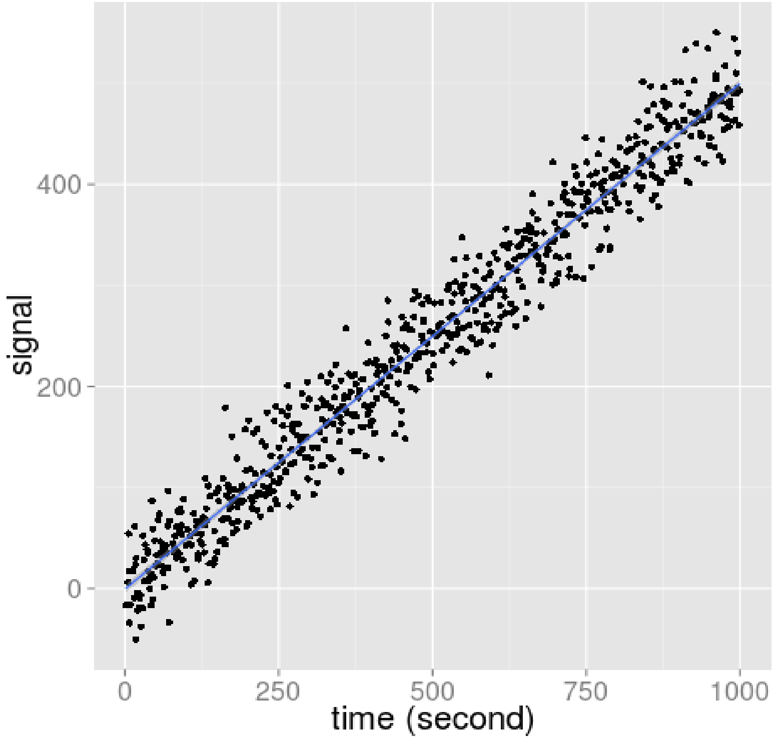
- Linear Regression
- Cost Functions
- Gradient Descent
- **Hands-on Example**
- Evaluating Regression Models
- Regularization

Agenda

- Linear Regression
- Cost Functions
- Gradient Descent
- Hands-on Example
- **Evaluating Regression Models**
- Regularization

Toy Example

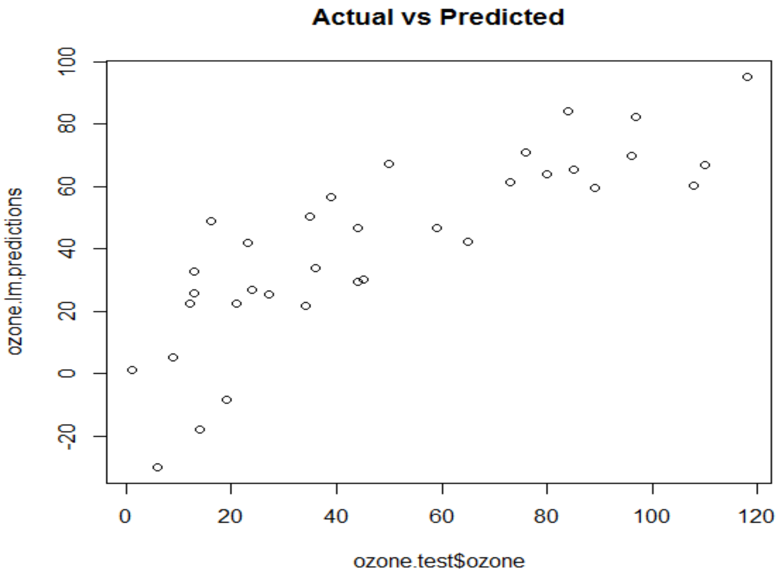
- Suppose we have a model:
  - $h(t) = -0.498 + 0.5 * t$
- How do we evaluate?
  - Train/Test Split
- What metrics to use?

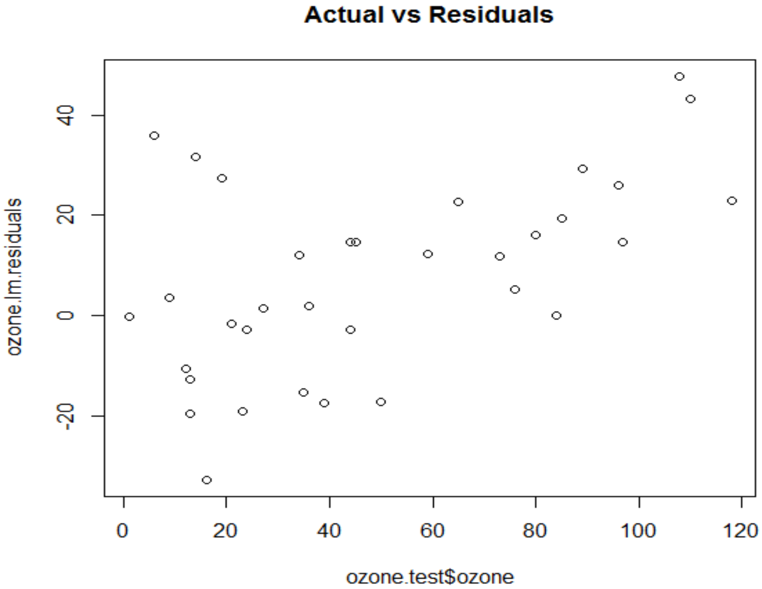


Graphical Evaluation Methods

Ideal: Straight line with slope 1

Ideal: Randomly distributed about 0





datasciencedojo
unleash the data scientist in you

Common Metrics

- Mean Absolute Error (MAE)
- Root-Mean-Square Error (RMSE)
  - Root-Mean-Square Deviation
- Coefficient of Determination (R^2)

Mean Absolute Error

$$MAE(\theta) = \frac{\sum_{i=1}^{m} \left| h_\theta\left(x^i\right) - y^i \right|}{m}$$

- Mean of residual values
- "Pure" measure of error

Mean Absolute Error - Example

$$y = \{-5.9, 48.3, 4.1, -8.4, 42.2, 10.3\}$$

$$h_\theta(x) = \{0.5, 1.0, 1.5, 4.5, 5.0, 6.5\}$$

$$|h_\theta(x) - y| = \{6.4, 49.3, 2.6, 12.9, 37.2, 3.8\}$$

$$MAE(\theta) = \frac{112.2}{6} = 18.7$$

Root-Mean-Square Error

$$RMSE(\theta) = \sqrt{\frac{\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2}{m}}$$

- Square root of mean of squared residuals
- Penalizes large errors more than small
- Good when large errors particularly bad

RMSE - Example

$$y = \{-5.9, 48.3, 4.1, -8.4, 42.2, 10.3\}$$

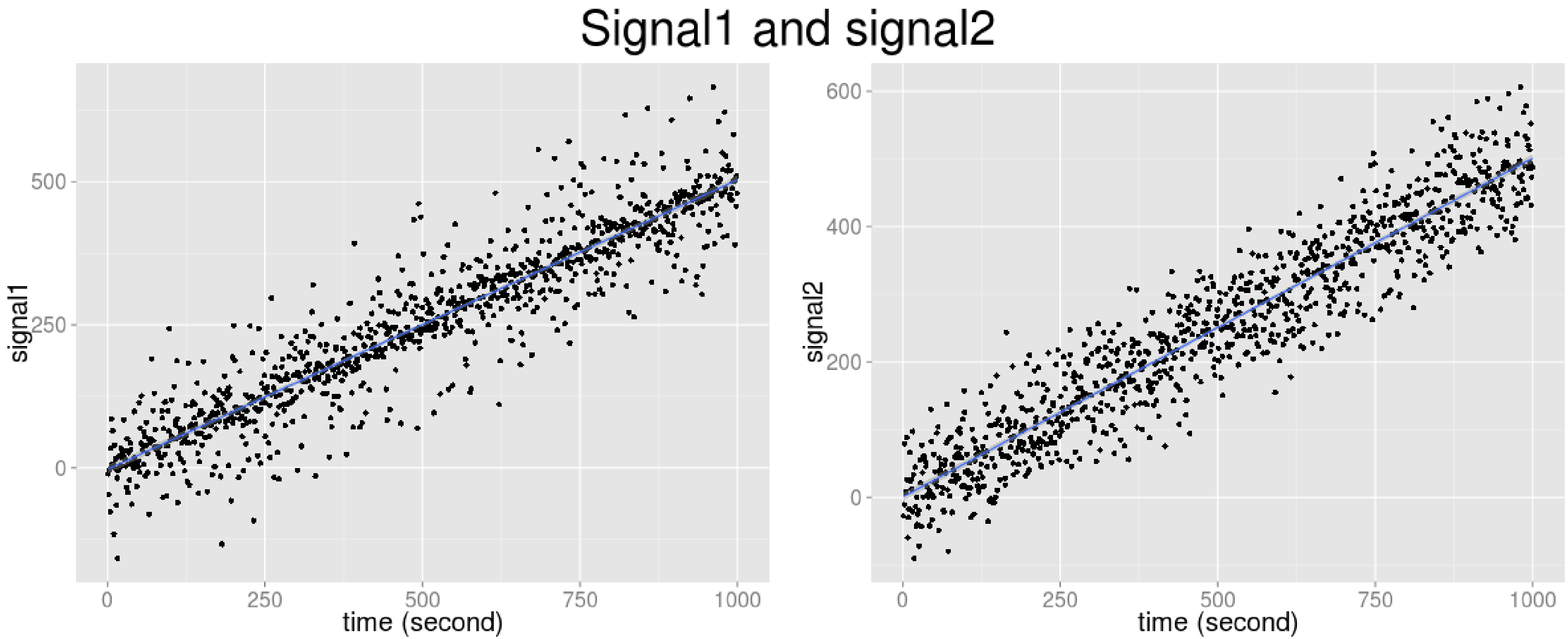
$$h_\theta(x) = \{0.5, 1.0, 1.5, 4.5,5.0,6.5\ \}$$

$$\left(h_\theta(x) - y^i\right)^2 = \{40.96,2237,6.76,166,1383,14.4\}$$

$$RMSE(\theta) = \sqrt{\frac{3848.12}{6}} = 25.3$$

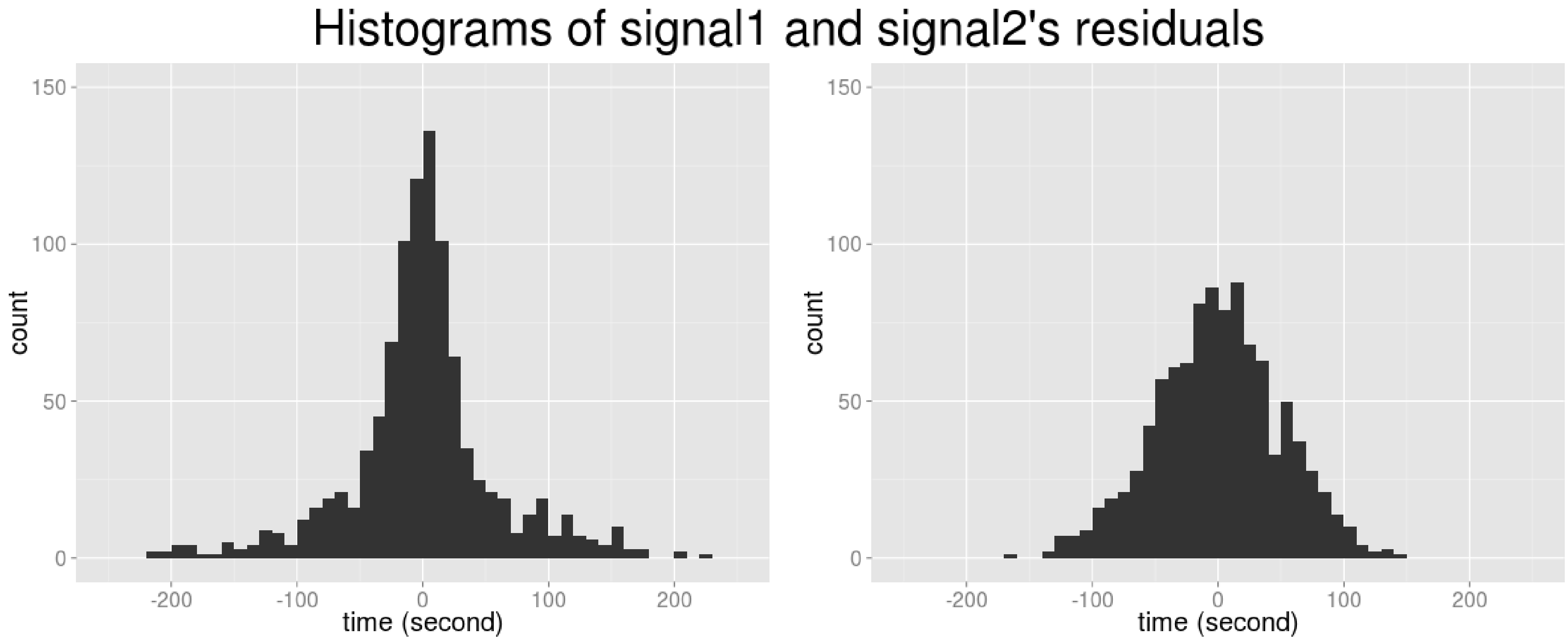
datasciencedojo
unleash the data scientist in you

MAE vs RMSE



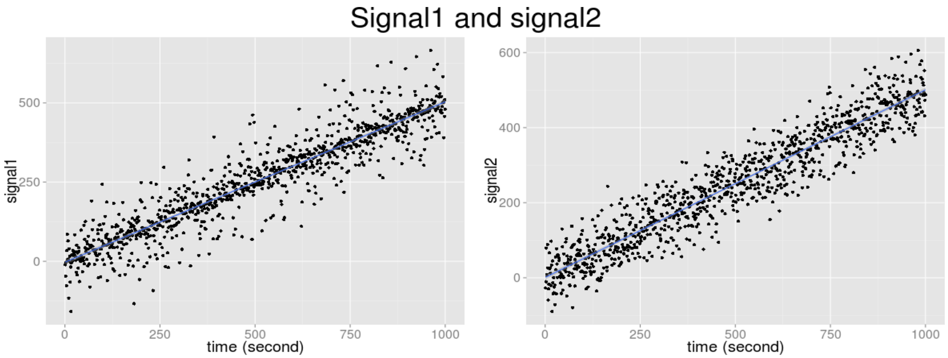
datasciencedojo
unleash the data scientist in you

MAE vs RMSE



datasciencedojo
unleash the data scientist in you

MAE vs RMSE



- MAE: **41.926** < 43.199
- RMSE: 64.458 > **54.516**
- Large deviation is penalized more by RMSE

Coefficient of Determination (R^2)

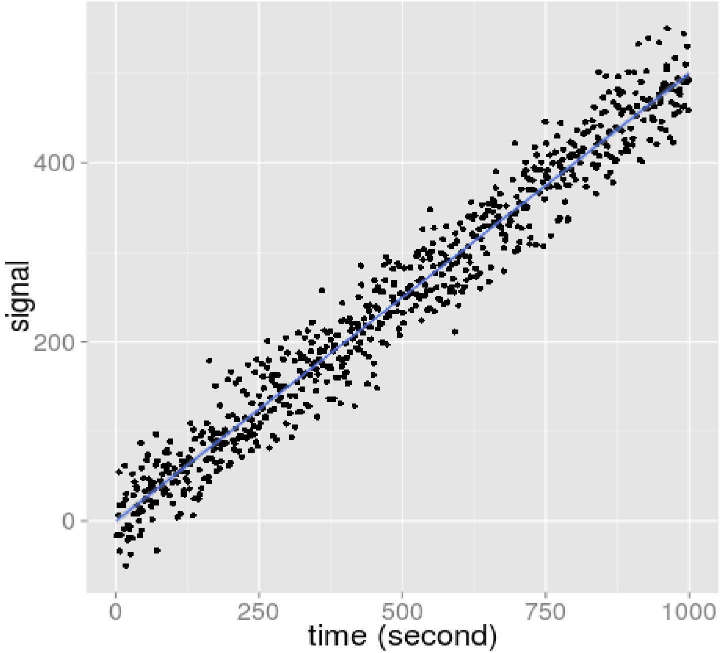
$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

where $$SS_{res} = \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i\right)^2 \quad , SS_{tot} = \sum_{i=1}^{m} \left(y^i - \bar{y}\right)^2$$

- SS_{res} – Sum of squared residuals (i.e. total squared error)
- SS_{tot} – Sum of squared differences from mean (i.e. total variation in dataset)
- Result: Measure of how well the model explains the data
  - "Fraction of variation in data explained by model"

R^2 Example

- $R^2 = 0.958$
- Real models usually much worse
- $R^2 = 0.6$ can be a good model depending on application



datasciencedojo
unleash the data scientist in you

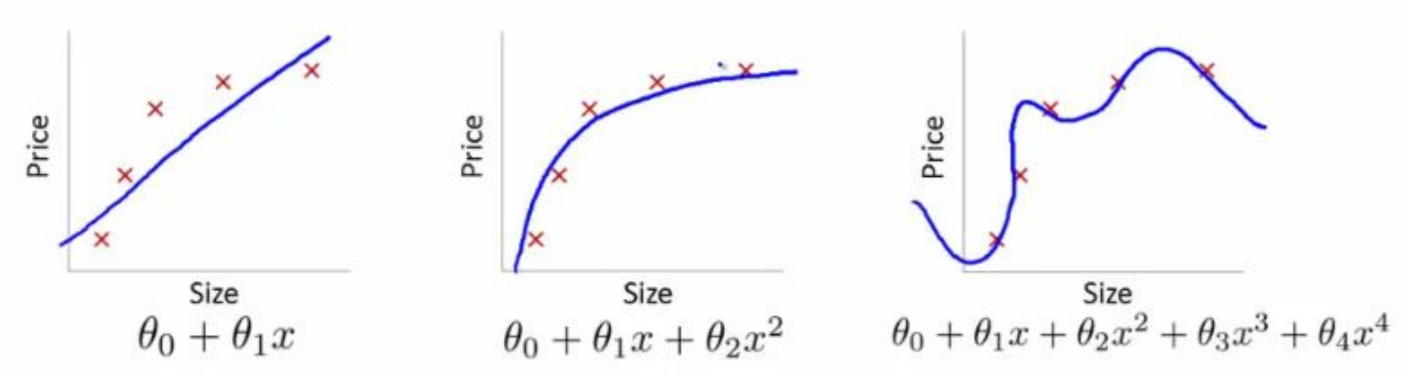
Agenda

- Linear Regression
- Cost Functions
- Gradient Descent
- Hands-on Example
- Evaluating Regression Models
- **Regularization**

Overfitting

- Want to extract general trends
- Danger: "memorizing" the training set
- A model is **overfit** when model performance on test set is much worse than on training set.

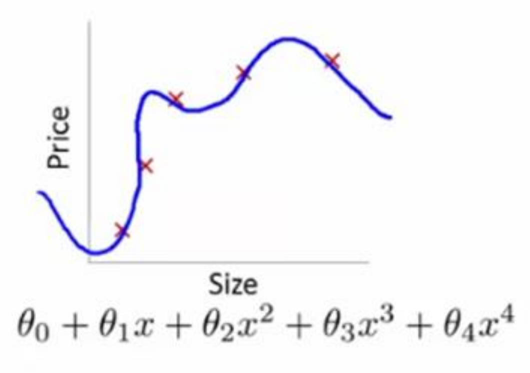
Overfitting

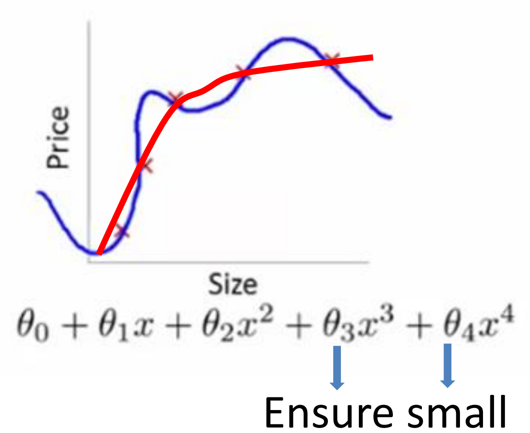


Complexity

- What makes a good model overfit?
  - Nature of training data
  - **Complexity of model**
- How do we handle these?
  - Cross validation
  - Manual model constraint
  - Regularization

Intuition





- Want to discourage complex models – How?
- Adjust the cost function!
  - Penalize models with large high-order θ terms

$$J'(\theta) = J(\theta) + Penalty$$

Definitions

- Two most common
  - L1 regularization
    - lasso regression

$$J_{L1}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)^2 + \lambda\sum_{j=1}^{d}|\theta_j|$$

  - L2 regularization
    - ridge regression
    - weight decay

$$J_{L2}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$

Regularized Regression

$$J_{L1}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)^2 + \lambda\sum_{j=1}^{d}\left|\theta_j\right| \qquad J_{L2}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$

- Find the best fit
- Keep the θ_j terms as small as possible.
- λ is a user-set parameter which controls the trade off

Regularized Regression

$$J_{L1}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)^2 + \lambda\sum_{j=1}^{d}\left|\theta_j\right| \qquad J_{L2}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$

- Size of λ important
  - λ high => higher penalty
  - λ low => lower penalty.

datasciencedojo
unleash the data scientist in you

QUESTIONS