Regression

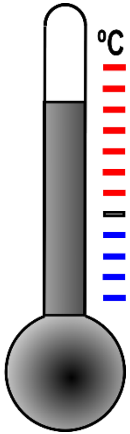
Agenda

- Introduction
- Cost Function & Gradient Descent
  - Minimization
  - Implementation
- Hands-on Example
- Evaluating Regression Models
- Regularization

Regression

Sales Forecasts

Housing Price Predictions



Daily Temperature Highs & Lows

Regression vs Classification

- Classification
  - Target is discrete with finite value set
  - **Examples:** survived/dead, face/non-face, fraud/non-fraud
- Regression
  - Target is continuous
  - **Examples:** price, weight, height, temperature,

Input Notation Summary

x^i – Each row of features
x_i – Each column of features
X – Set of all the feature columns

Features

y^i – Each row of the target(s)
Y – Set of all the target columns

Target

n – Number of rows in the dataset
m – Number of columns in the dataset

Example: Titanic Dataset

Passenger Id	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S

$$x_4^5$$

5: The passenger is in the 5th row
4: The passenger's name is the 4th column

Example: Ozone Dataset

The ozone dataset uses radiation, temperature and wind to predict ozone levels.

	ozone	radiation	temperature	wind	
		x_1	x_2	x_3	
Y	41	190	67	7.4	X
	36	118	72	8.0	
	12	149	74	12.6	
	18	313	62	11.5	
	23	299	65	8.6	
	19	99	59	13.8	

Using this notation, we can describe all the columns of the dataset.

Example: Ozone Dataset

So how do we describe all the rows?

	ozone	radiation	temperature	wind
Row 1	41	190	67	7.4
Row 2	36	118	72	8.0
Row 3	12	149	74	12.6
	18	313	62	11.5
	23	299	65	8.6
	19	99	59	13.8

$x^1 = [190, 67, 7.4]$

$x^2 = [118, 72, 8.0]$

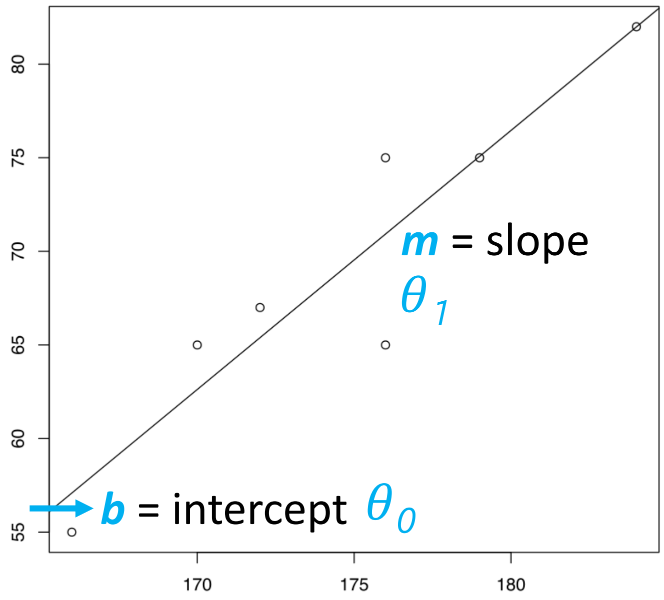
$x^3 = [149, 74, 12.6]$

COST FUNCTION AND GRADIENT DESCENT

Defining a line

How do we define a line in slope-intercept notation?

- $y = \boldsymbol{m}x + \boldsymbol{b}$

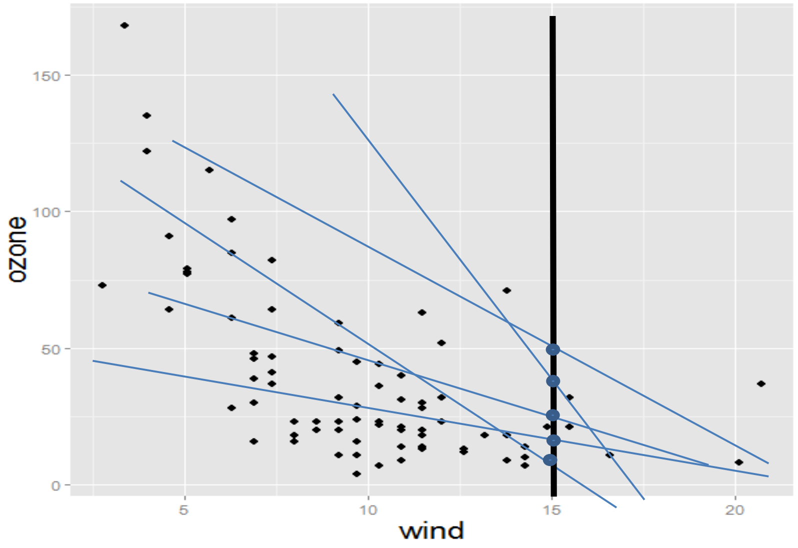


In θ notation?

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

What is a good regression line?

- Wind Speed=15 mph
- Ozone = ?
- Use the line that is **somewhere in the middle**
- How do we define "middle"?

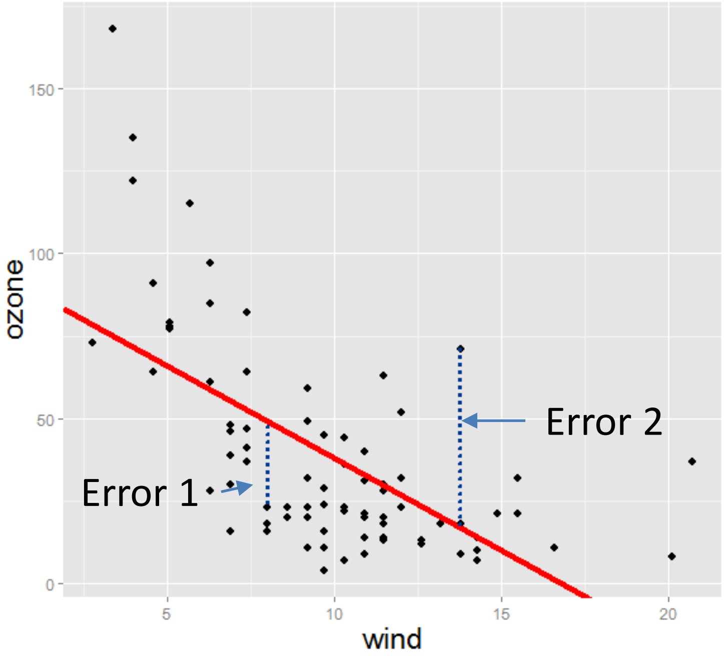


$$h_\theta(x) = \theta_0 + \theta_1 x$$

Cost Function

Residuals

- A measure of error
- Difference between hypothesis $h_\theta(x)$ (predicted value) and true value (known target)



datasciencedojo
unleash the data scientist in you

Cost Function

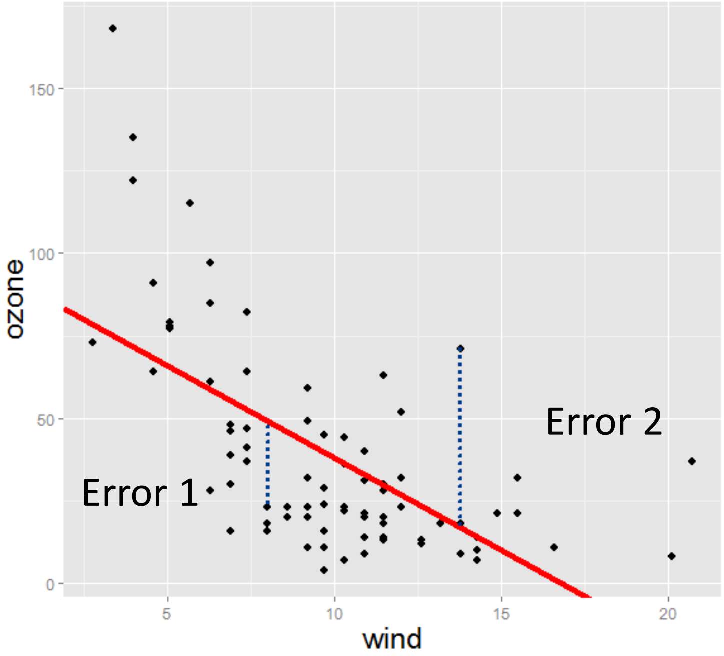
We want to minimize residuals

By "cost" or "loss" function – $J(\theta)$

- Smaller for lower error
- Larger for higher error

Residuals – a measure of error

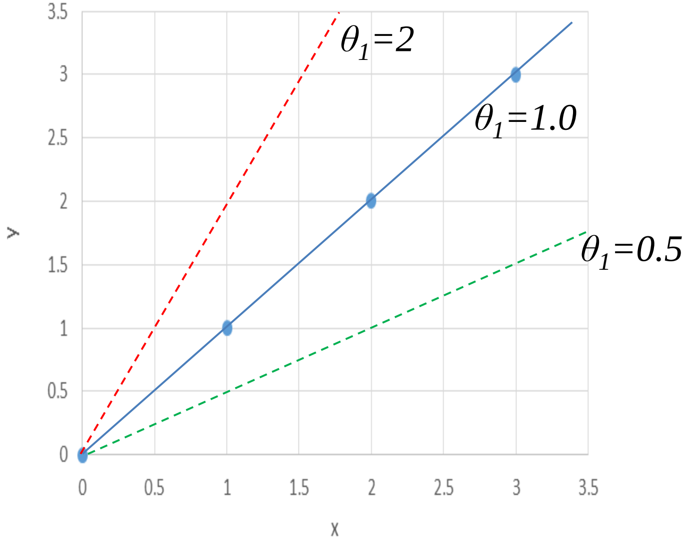
$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_\theta(x^i) - y^i\right)^2$$



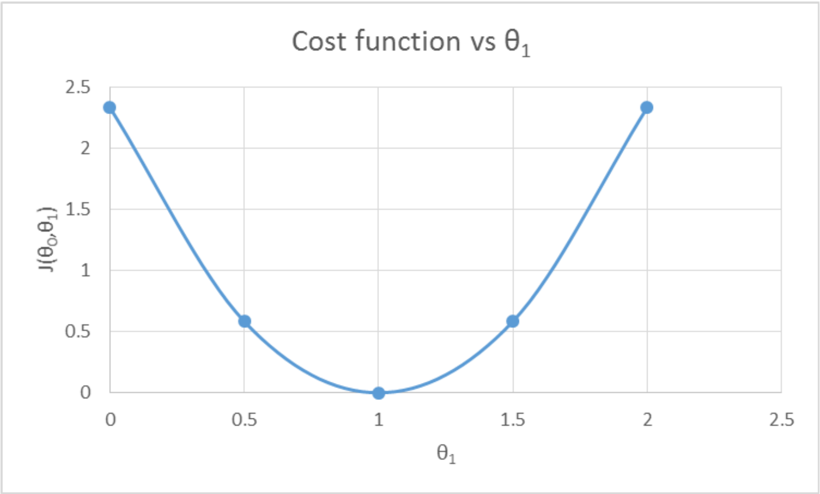
Mean Square Error

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

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



θ_0=0

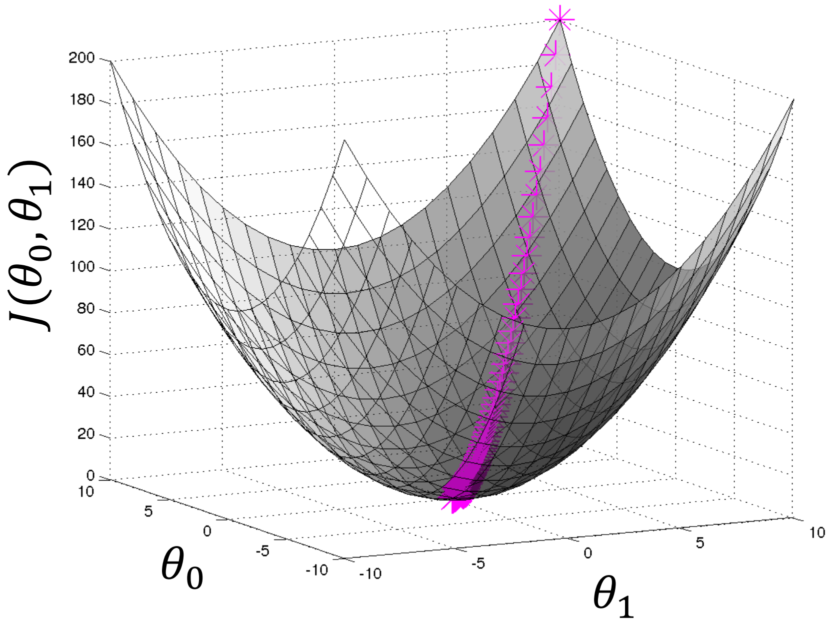


datasciencedojo
unleash the data scientist in you

Cost function in two dimensions

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

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



HOW DO WE FIND OUT THE MINIMUM OF THE COST FUNCTION

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

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

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

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 $x=3$

Gradients

- **Derivative:** slope in one direction
- What about more features?
- **Gradient:** a multi-dimensional derivative

Gradient Descent

- Goal : minimize $J(\theta)$
- Start with some initial θ and then perform 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

Gradient Descent

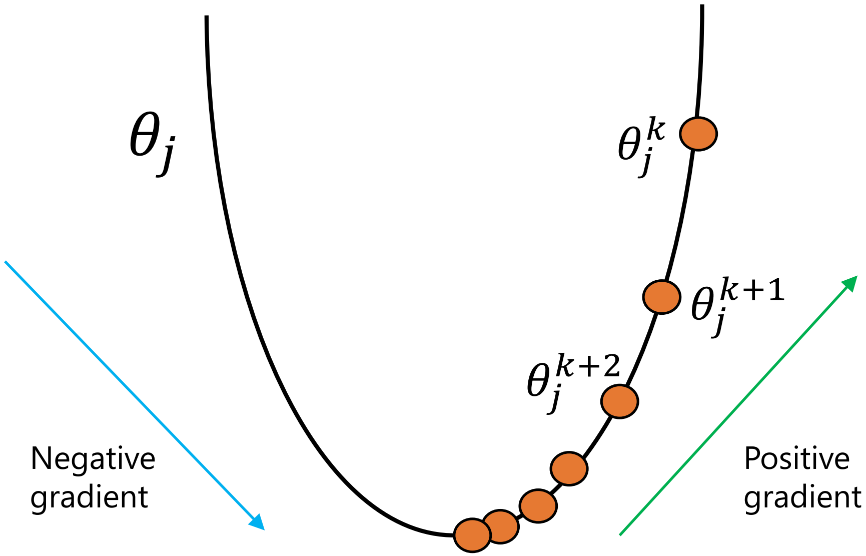
$$\theta_j^{k+1} := \theta_j^k - \alpha \frac{\partial}{\partial \theta_j} J(\theta^k)$$

- 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

Intuition

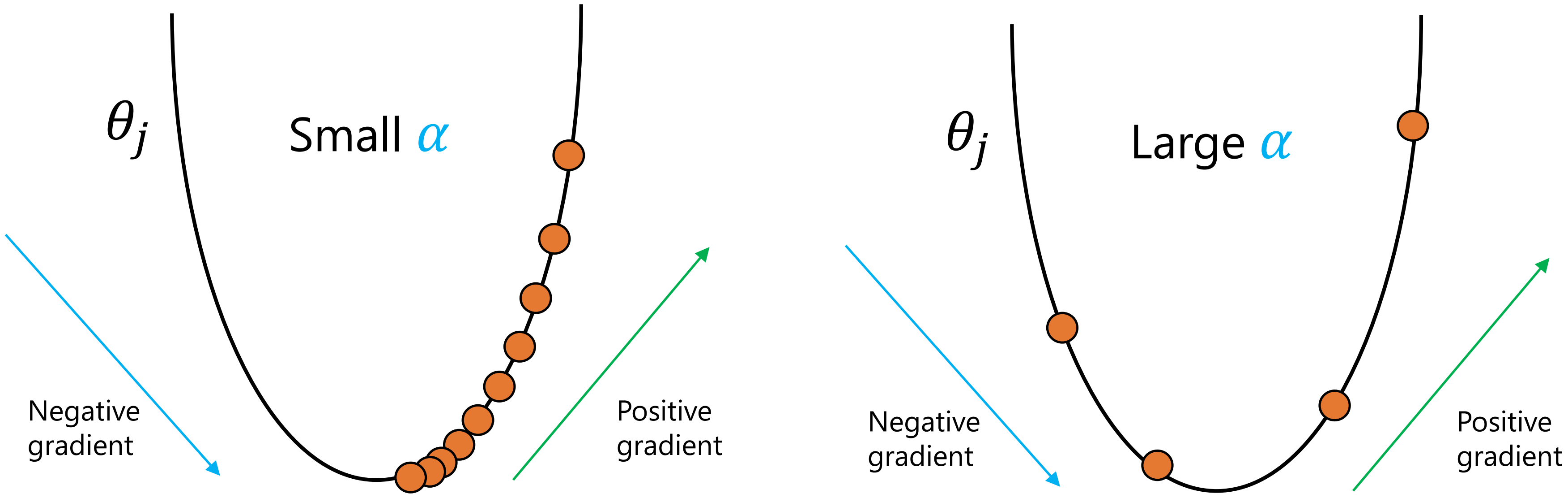


$$\theta_j^{k+1} := \theta_j^k - \alpha \frac{\partial}{\partial \theta_j} J(\theta^k)$$

datasciencedojo
unleash the data scientist in you

Learning Rate Effects

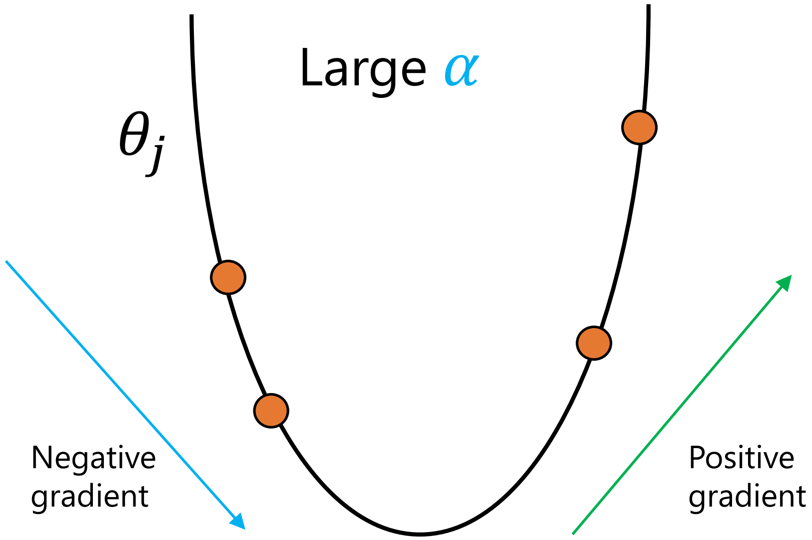
$$\theta_j^{k+1} := \theta_j^k - \alpha \frac{\partial}{\partial \theta_j} J(\theta^k)$$



datasciencedojo
unleash the data scientist in you

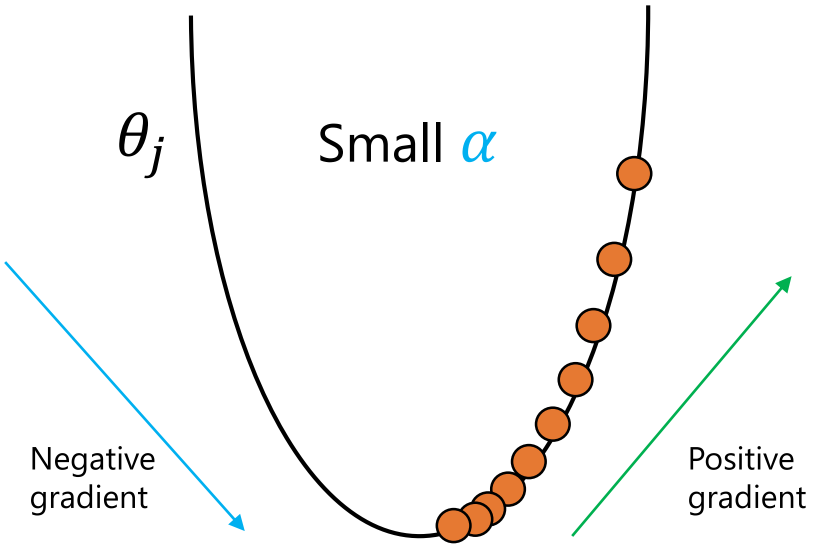
Learning Rate Effects

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



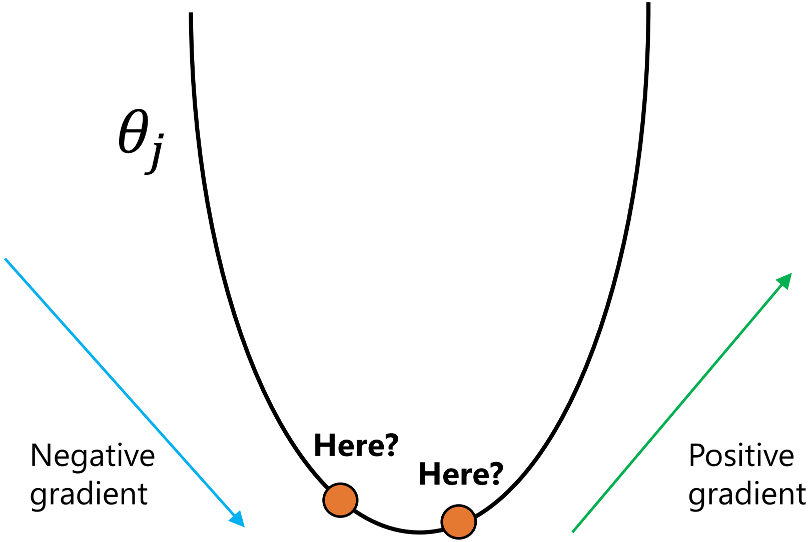
Learning Rate Effects

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



Gradient Descent Implementation

When do we stop updating?



- When θ_j^{k+1} is close to θ_j^k
- When $J(\theta^{k+1})$ is close to $J(\theta^k)$ [Error does not change]

Batch Gradient Descent

- How do we incorporate all our data?
- Loop!

For j from 0 to m:

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

- h_θ is updated only once the loop has completed
- Weaknesses?

Stochastic Gradient Descent

- Consider an alternative approach:

for i from 1 to n:

for j from 0 to m:

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

- h_θ is updated when inner loop is complete
- If the training set is big, converges quicker than batch
- May oscillate around a minimum of $J(\theta)$ and never converge

Batch vs. Stochastic

Which is the best to use? It depends.

	Batch Gradient Descent	Stochastic Gradient Descent
Function	Updates hypothesis by scanning whole dataset	Updates hypothesis by scanning one training sample at a time
Rate of convergence	Slowly	Quickly (but may oscillate at minimum)
Appropriate Dataset Size	Small	Large

Agenda

- Introduction
- Cost Functions & Gradient Descent
  - Minimization
  - Implementation
- Hands-on Example
- Evaluating Regression Models
- Regularization

Non-Parametric Algorithms

- Uses flexible number of parameters that can grow as it learns from more data
- Slow computation
- Ex: Decision Trees, Neural Nets

Parametric Algorithms

- Uses fixed number of parameters and makes strong assumptions about the data
- Fast computation
- Ex: Traditional scientific modeling, linear regression

Common Metrics

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

Mean Absolute Error

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

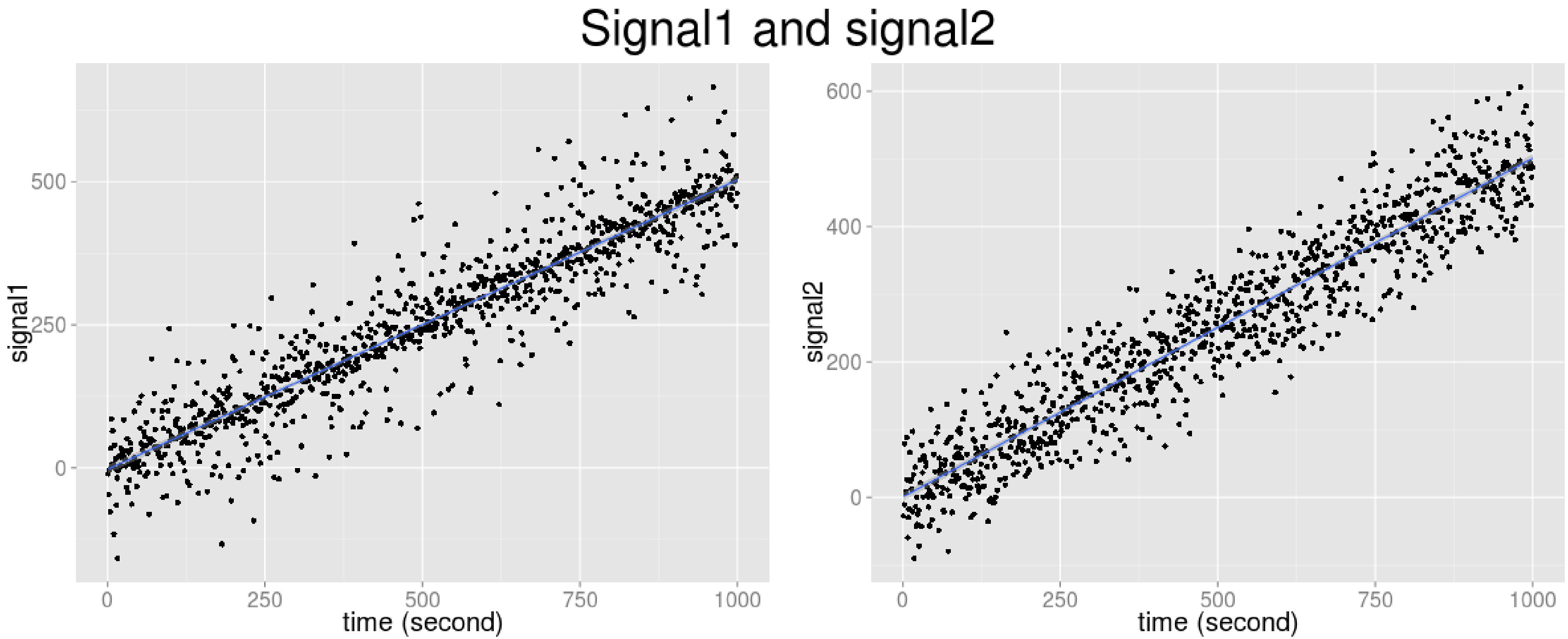
- Mean of residual values
- "Pure" measure of error

Root-Mean-Square Error

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

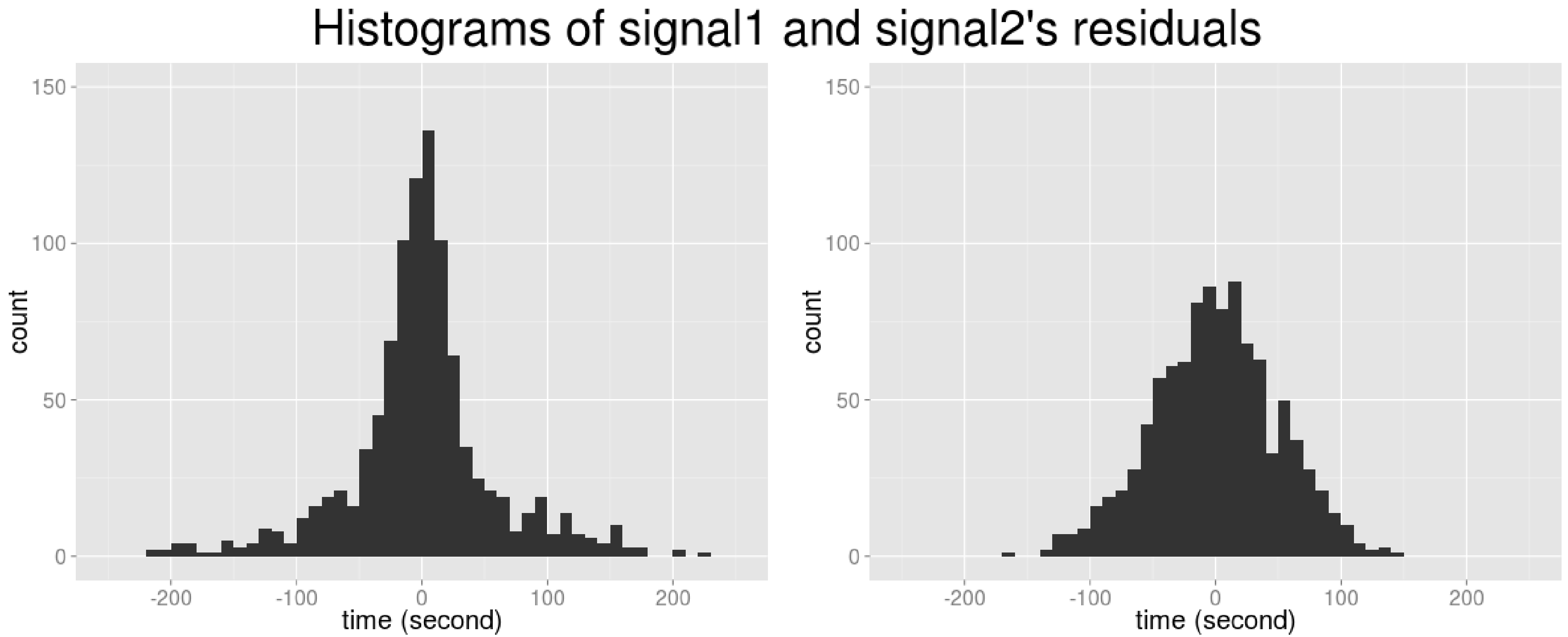
- Square root of mean of squared residuals
- Penalizes large errors more than small
- Good when large errors particularly bad

MAE vs RMSE



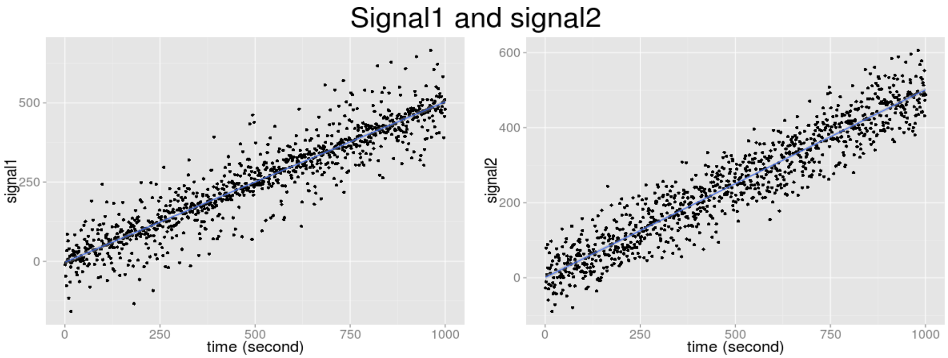
Signal1 and signal2
signal1
500
250
0
0
250
500
750
1000
time (second)
signal2
600
400
200
0
0
250
500
750
1000
time (second)

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

datasciencedojo
unleash the data scientist in you

Coefficient of Determination (R^2)

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

where

$$SS_{res} = \sum_{i=1}^{n} \left(h_\theta\left(x^i\right) - y^i\right)^2 \qquad SS_{tot} = \sum_{i=1}^{n} \left(y^i - \bar{y}\right)^2$$

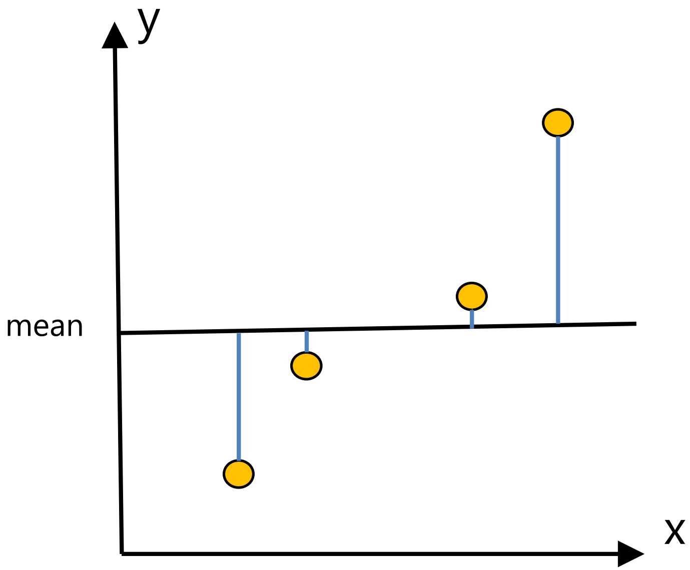
SS_{res} – Sum of squared residuals (i.e. total squared error)

SS_{tot} – Sum of squared differences from mean (i.e. total variation in dataset)

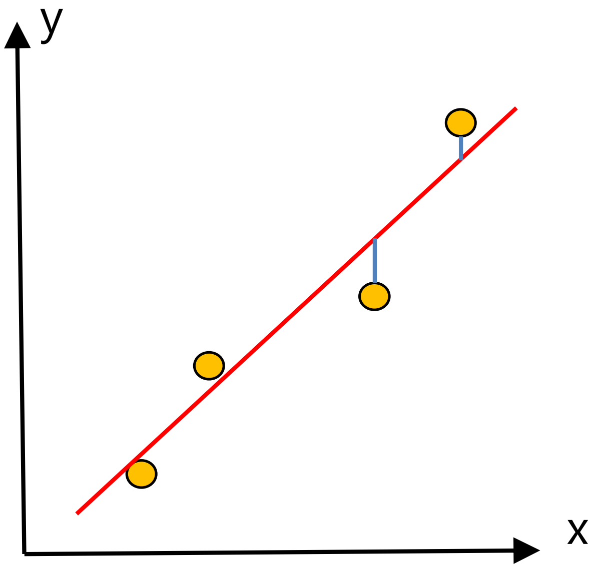
Result: Measure of how well the model explains the data

- "Fraction of variation in data explained by model"

Coefficient of Determination



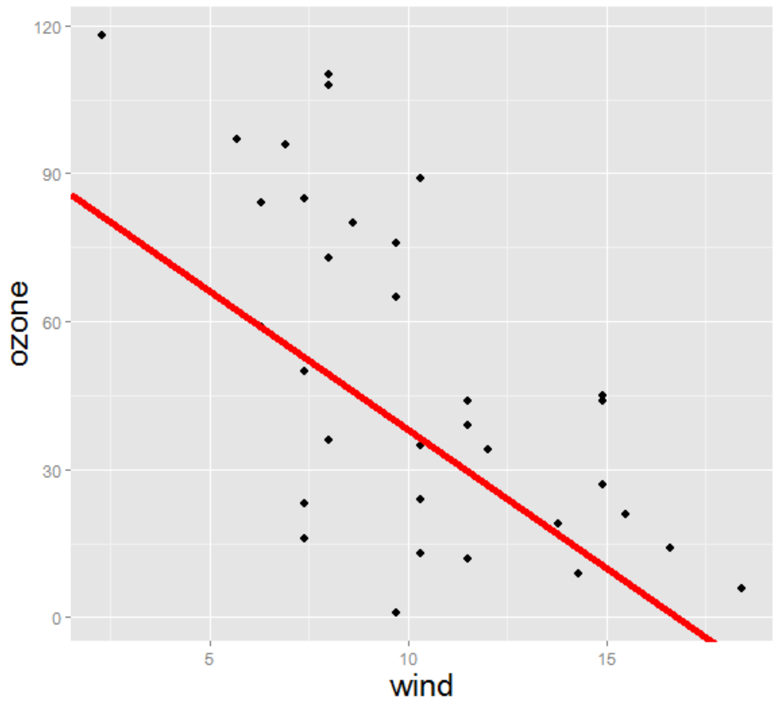
SS_{tot}



SS_{res}

R^2 Example

- $R^2 = 0.277$
- Want a much better model for real application
- $R^2 = 0.6$ can be a good model



datasciencedojo
unleash the data scientist in you

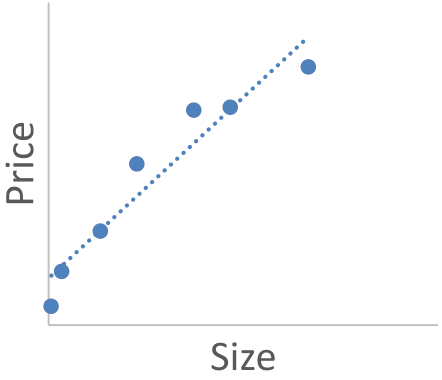
Agenda

- Introduction
- Cost Functions & Gradient Descent
  - Minimization
  - Implementation
- Hands-on Example
- Evaluating Regression Models
- Regularization

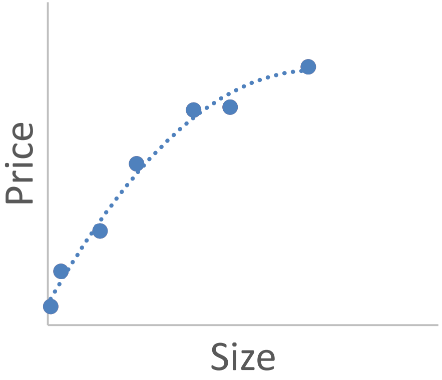
Overfitting

- 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.

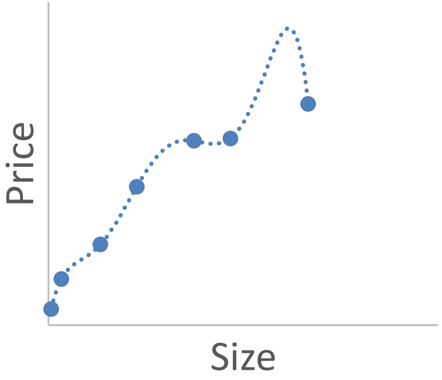
Overfitting



$\theta_0 + \theta_1 x$



$\theta_0 + \theta_1 x + \theta_2 x^2$

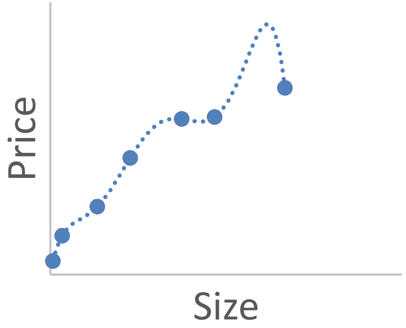


$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

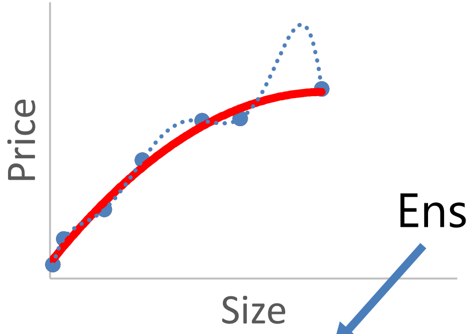
Complexity

- 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



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



Ensure Small

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

- Want to discourage complex models automatically – 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

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

  - L2 regularization
    - ridge regression
    - weight decay

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

Regularized Regression

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

- 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

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

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

- Size of λ important
  - λ too high => no fitting
  - λ too low => no regularization

QUESTIONS