

Introduction to Regression

Agenda

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

Agenda

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

Notation

- x^i – The feature vector for the i th data object
 - $x^i = [x_1^i, x_2^i, \dots, x_m^i]$
- X – The set of all x^i
- y^i – The true target value for the i th data object
- Y – The set of all y^i
- n – Number of rows in the dataset
- m – Number of columns in the dataset

Example: Ozone Levels

- Daily measurements of weather data
- Predict ozone level for public awareness

```
ozone <- read.table('Datasets/Ozone/ozone.data')
```

```
head(ozone)
```

	y	x_1	x_2	x_3	
	ozone	radiation	temperature	wind	
y^1	41	190	67	7.4	$x^1 = [190, 67, 7.4]$
y^2	36	118	72	8.0	$x^2 = [118, 72, 8.0]$
y^3	12	149	74	12.6	$x^3 = [149, 74, 12.6]$
	18	313	62	11.5	
	23	299	65	8.6	
	19	99	59	13.8	

Example: Ozone Levels

```
ozone <- read.table('Datasets/Ozone/ozone.data')  
head(ozone)
```

	y	x_1	x_2	x_3	
	ozone	radiation	temperature	wind	
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	

Regression vs Classification

- Classification

- Target is discrete with finite value set
- Ex: survived/dead, face/non-face, fraud/non-fraud, product categories, ranking

- Regression

- Target is continuous or ordinal
- Ex: price, weight, height, temperature, ranking

Non-Parametric Algorithms

- Cannot be represented as a single closed form function
- Many different assumptions about underlying structure of data
- Ex: Decision Trees, Neural Nets

Parametric Algorithms

- Assumption: Relationship between features and target can be represented as a closed form function
- This function "maps" $X \rightarrow Y$
- Used in traditional scientific modeling
- Ex: $y = 1 + 2x$
 - Parameters: [1, 2]

Parametric Notation

- h – A specific functional form (line, exponential, Poisson distribution, logit, etc.)
- θ – A vector of function parameters which define a specific hypothesis
- h_{θ} – A specific hypothesis (estimate) for the mapping $X \rightarrow Y$

Regression Example

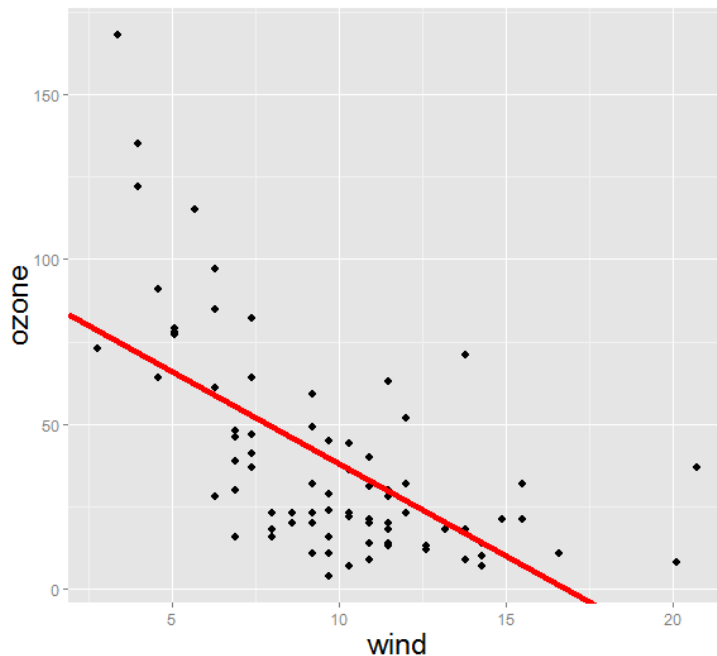
- How do we define a line?

- $y = mx + b$

- What is...

- θ ?

- h_{θ} ?



Regression Example

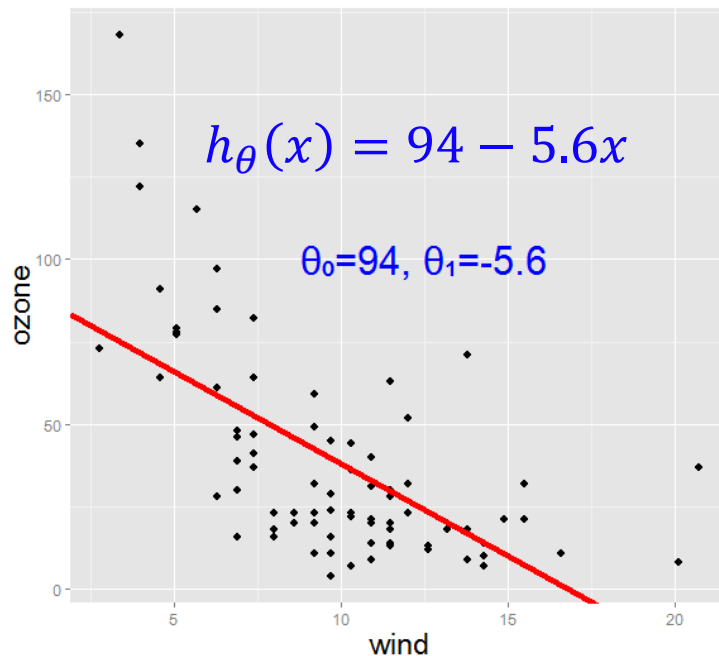
- How do we define a line?

- $y = mx + b$

- What is...

- θ ?

- h_{θ} ?



Regression Example

- What about with three features?

- What is...

- θ ?

- h_{θ} ?

y	x_1	x_2	x_3
ozone	radiation	temperature	wind
41	190	67	7.4
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

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

Regression Example

- What about with more features?
- What is...

- $\theta?$ $\theta = [\theta_0, \theta_1, \theta_2, \dots, \theta_m]$

- $h_\theta?$

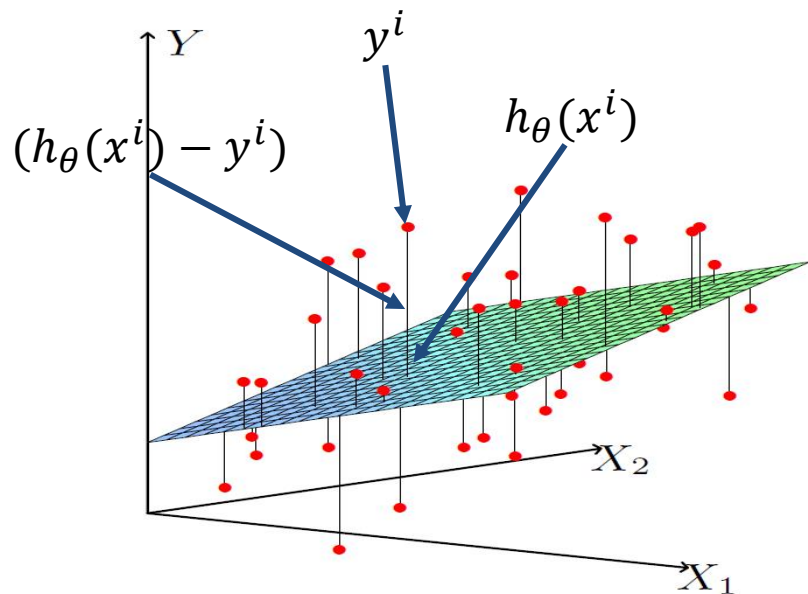
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_m x_m$$

Regression Algorithms

- $\theta = [\theta_0, \theta_1, \theta_2, \dots, \theta_m], x = [x_1, x_2, \dots, x_m]$
- Linear Regression
 - h – a line function
 - $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_m x_m = \theta^T x$
- Logistic Regression
 - h – a logit function
 - $h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$

Note: Logistic Regression is a classification algorithm

Regression Errors



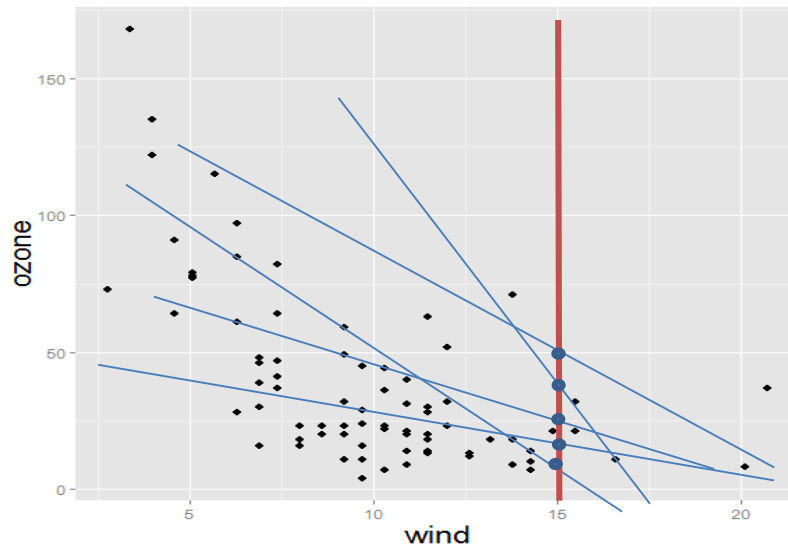
- How do we assess the error of a model?
- "Residual" – difference between predicted and actual target value
 - $res = (h_\theta(x^i) - y^i)$

Agenda

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

Regression Training

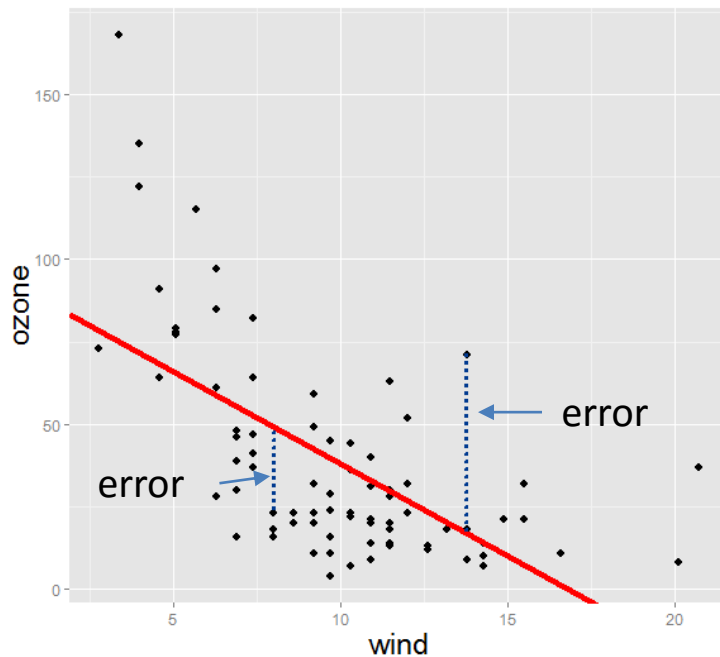
- Wind Speed=15 mph
- Ozone = ?
- Use the line that is somewhere in the middle
- How do we define "middle"?



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

Cost Function

- Want: a "cost" or "loss" function – $J(\theta)$
 - Smaller for lower error
 - Larger for higher error
- Residuals – a measure of error
 - Simple sum doesn't work – why?
- Solution: Square them!
 - "Mean Square Error"

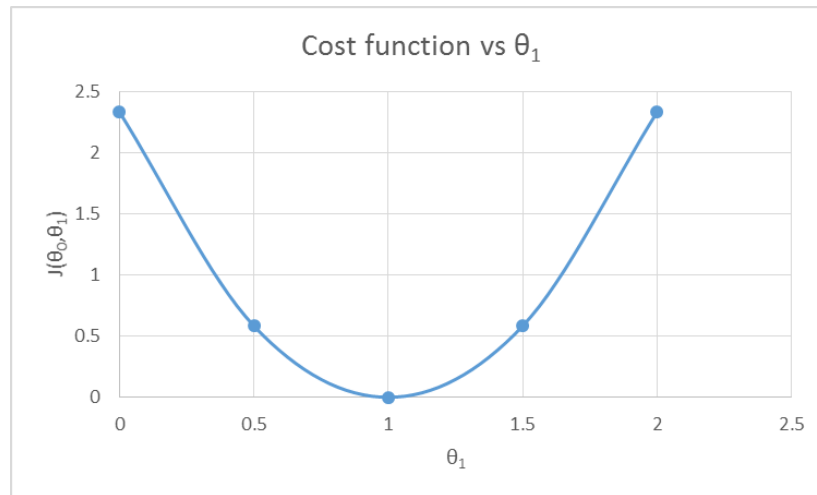
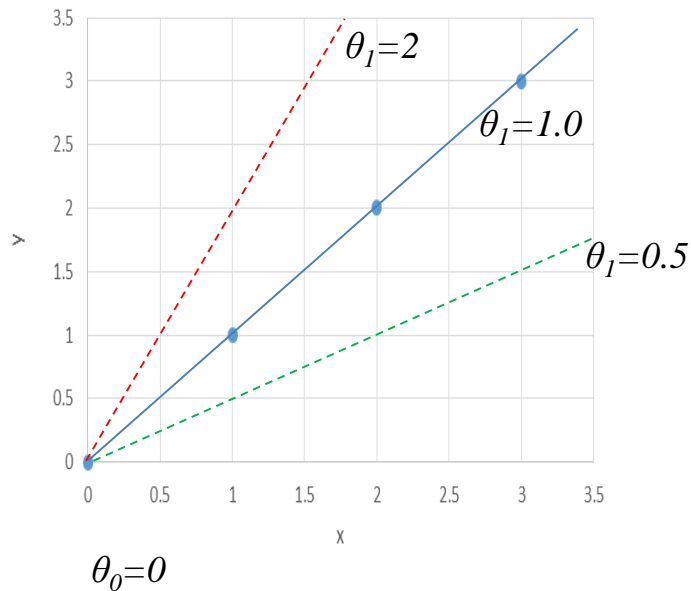


$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n (h_{\theta}(x^i) - y^i)^2$$

Mean Square Error – 1D

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

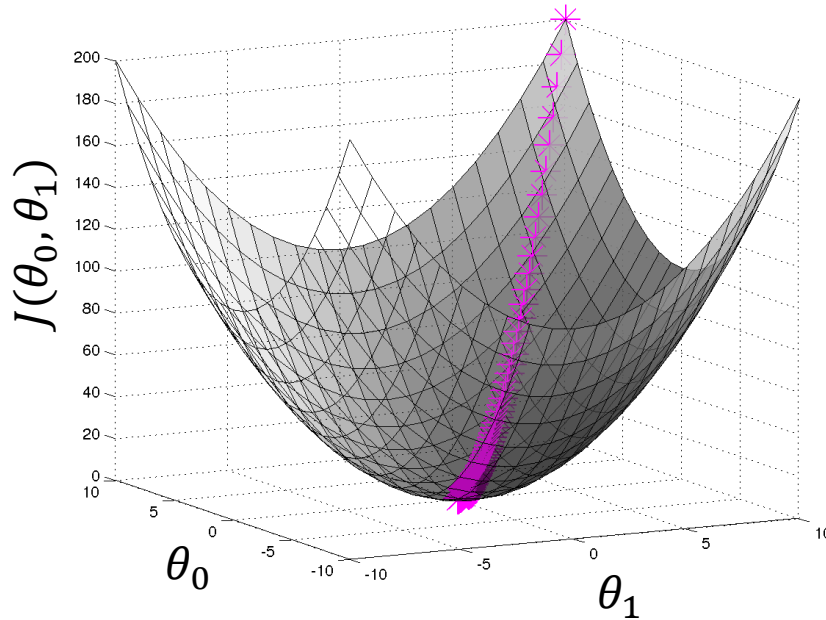
$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n (h_{\theta}(x^i) - y^i)^2$$



Mean Square Error – 2D

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

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



Agenda

- Introduction
- **Cost Functions & Gradient Descent**
 - **Minimization**
 - Implementation
- 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

$$\begin{aligned} P &= x y^2 \\ &= x (9-x)^2 \end{aligned}$$

Solution

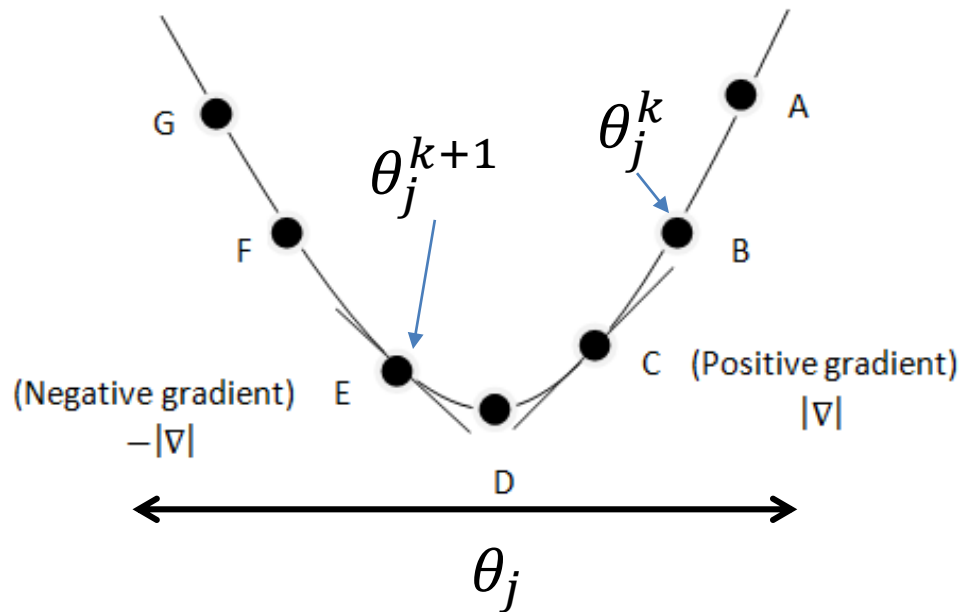
$$\begin{aligned} P' &= x(2)(9-x)(-1) + (1)(9-x)^2 \\ &= (9-x)[-2x + (9-x)] \\ &= (9-x)[9-3x] \\ &= (9-x)(3)[3-x] \\ &= 0 \end{aligned}$$

$$x=9 \text{ or } x=3$$

Gradients

- Derivatives: slope in one direction
- What about more features?
- Gradient: a multi-dimensional derivative
 - $\nabla J(\theta) = [\frac{\partial J}{\partial \theta_0}, \frac{\partial J}{\partial \theta_1}, \dots, \frac{\partial J}{\partial \theta_m}]$
 - Treat each dimension independently

Intuition



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

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

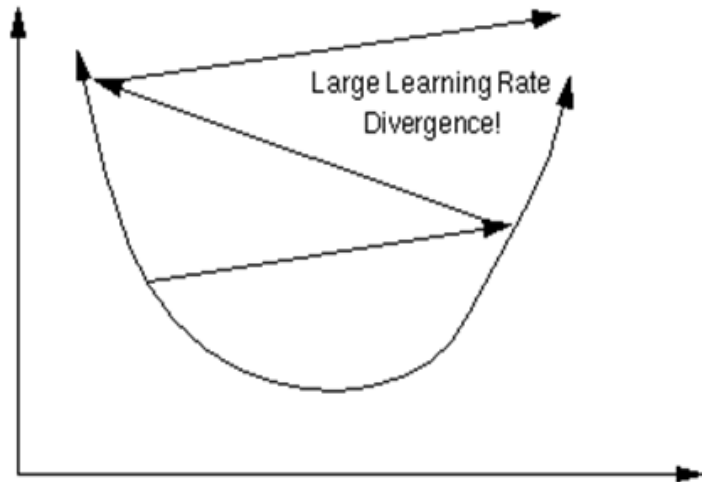
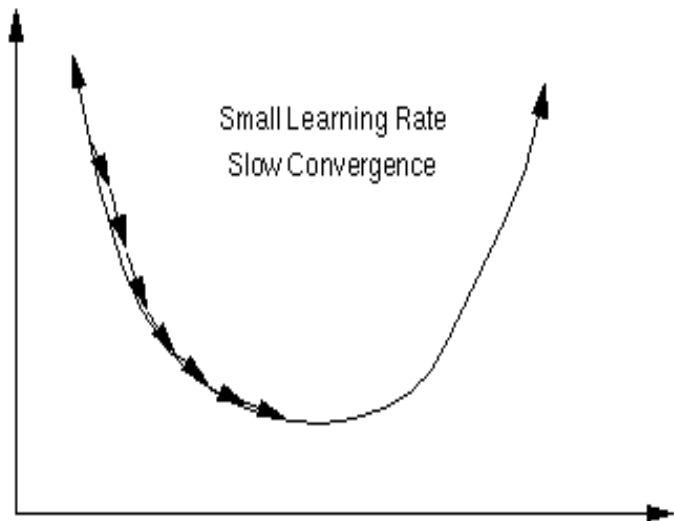
Gradient Descent

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

- 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

Learning Rate Effects

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



Gradient Descent Animations

Agenda

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

Gradient Descent Implementation

- Ordinary least squares, linear regression

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

- When do we stop updating ("converge")?
 - When θ stops changing
 - i.e. θ^{k+1} very close to θ^k
 - When error stops dropping
 - i.e. $J(\theta^{k+1})$ very close to $J(\theta^k)$
 - "close" can be absolute or relative

Batch Gradient Descent

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

For j from 0 to m :

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

- 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:

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

- 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

Batch Gradient Descent

Repeat until convergence {

For j from 0 to m:

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

}

- 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 n:

for j from 0 to m:

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

}

- 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

Agenda

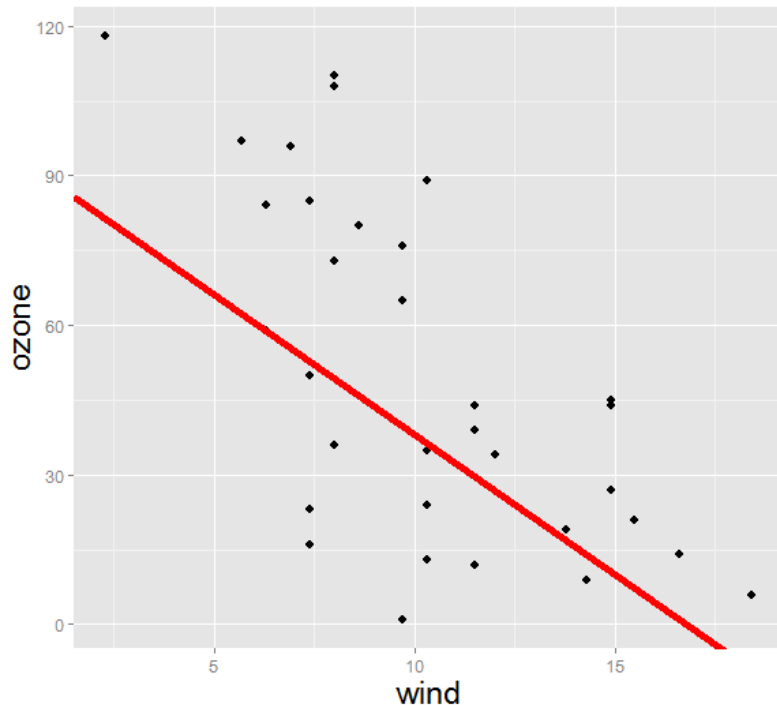
- Introduction
- Cost Functions & Gradient Descent
 - Minimization
 - Implementation
- **Hands-on Example**
- Evaluating Regression Models
- Regularization

Agenda

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

Regression Example

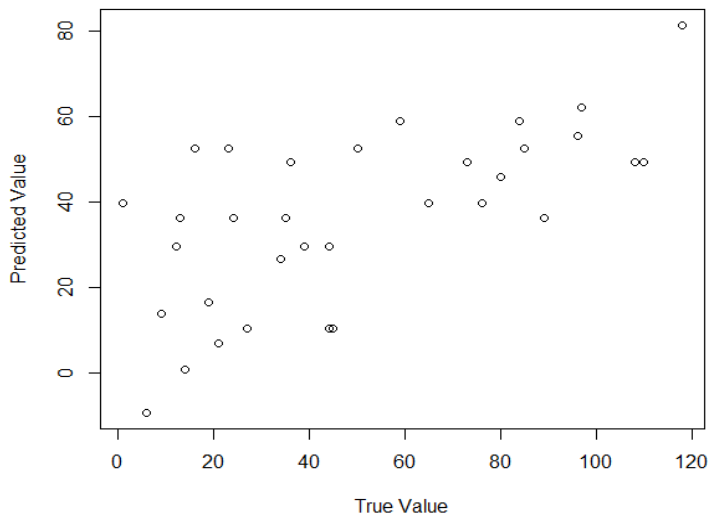
- From earlier:
 - $h_{\theta}(x) = 94 - 5.6 * x$
 - How do we evaluate?
 - Train/Test Split
- What metrics to use?
 - $J(\theta)$?



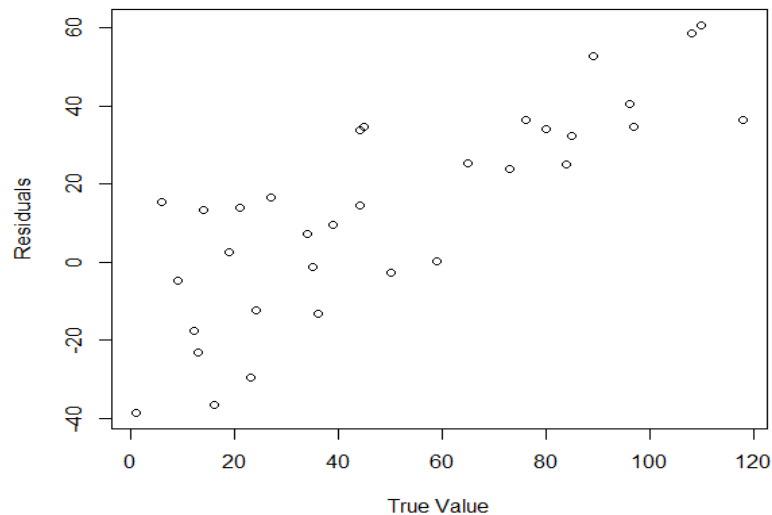
Graphical Evaluation Methods

Ideal: Straight line with slope 1 Ideal: Randomly distributed about 0

Ozone Test True vs Predicted Values



Ozone Test True Value vs Residuals



Common Metrics

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

Mean Absolute Error

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

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

Mean Absolute Error - Example

$$y = \{36, 19, 34, 6, 1, 45 \dots\}$$

$$h_{\theta}(x) = \{27, -2.6, 13, -7.3, -2.6, 48 \dots\}$$

$$|h_{\theta}(x) - y| = \{9, 21.6, 21, 13.3, 3.6, 3 \dots\}$$

$$MAE(\theta) = \frac{71.5}{6} = 11.9$$

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

RMSE - Example

$$y = \{36, 19, 34, 6, 1, 45, \dots\}$$

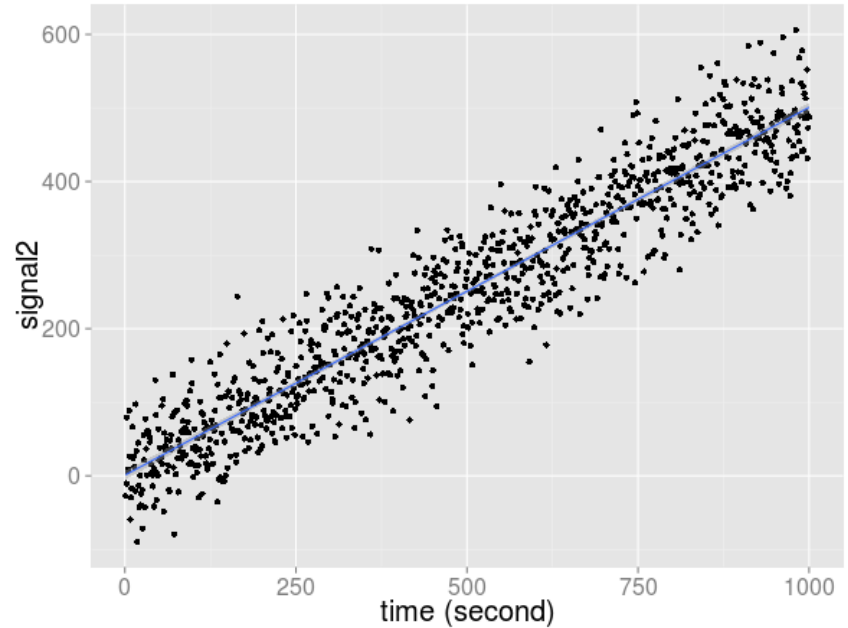
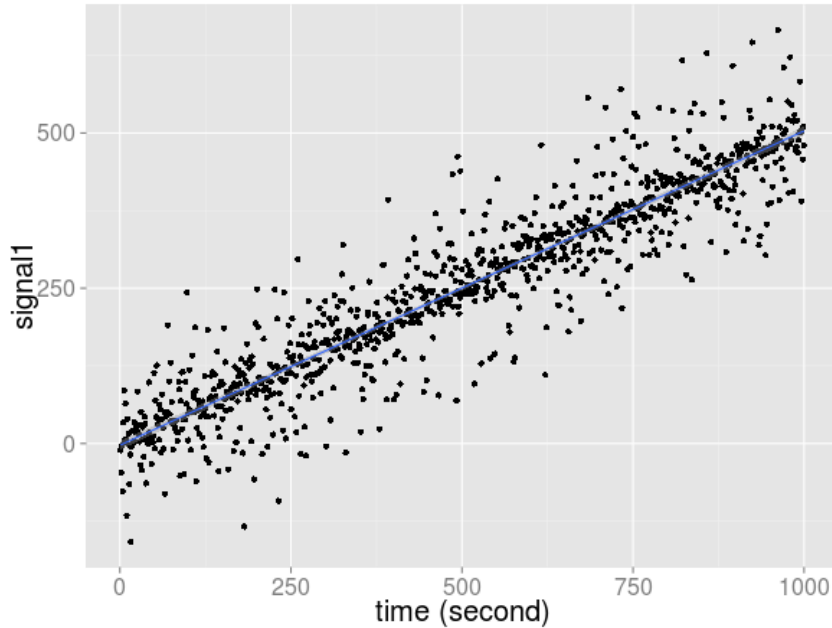
$$h_{\theta}(x) = \{27, -2.6, 13, -7.3, -2.6, 48, \dots\}$$

$$(h_{\theta}(x) - y)^2 = \{81, 467, 441, 177, 13, 9, \dots\}$$

$$MAE(\theta) = \sqrt{\frac{1218}{6}} = 14.2$$

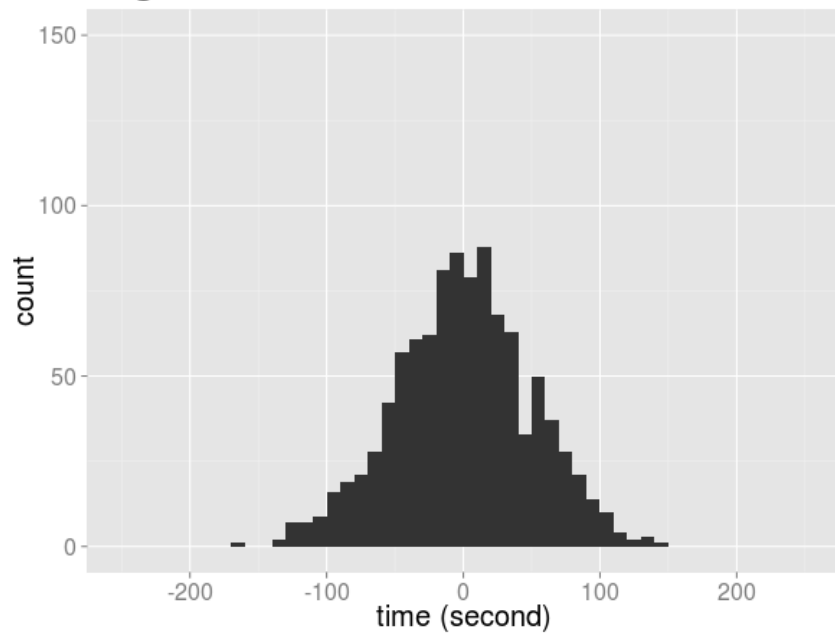
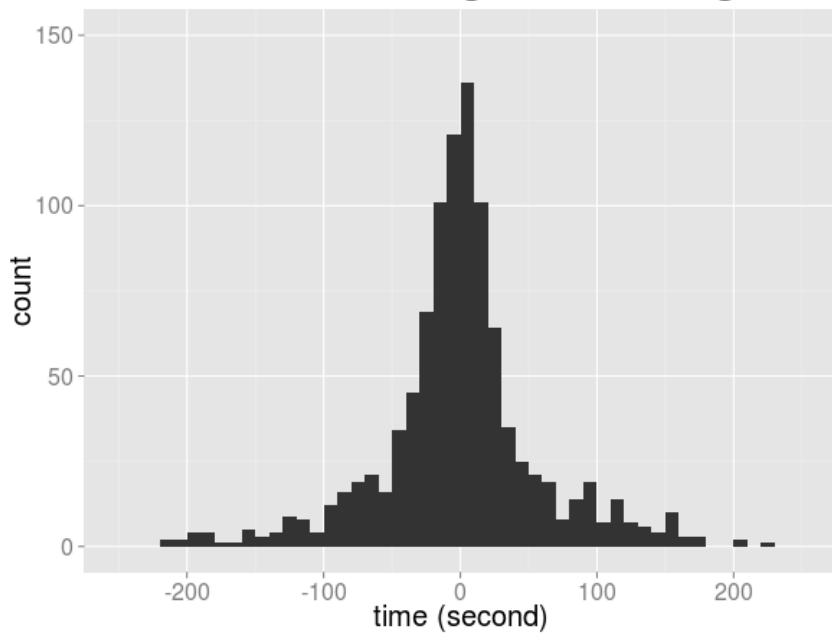
MAE vs RMSE

Signal1 and signal2

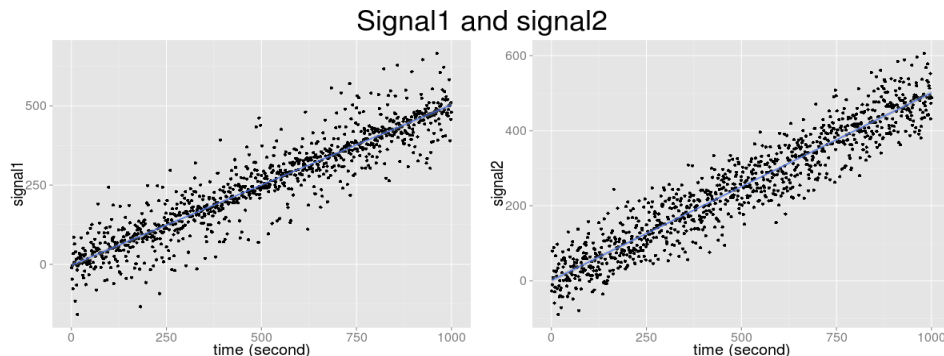


MAE vs RMSE

Histograms of signal1 and signal2's residuals



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)

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

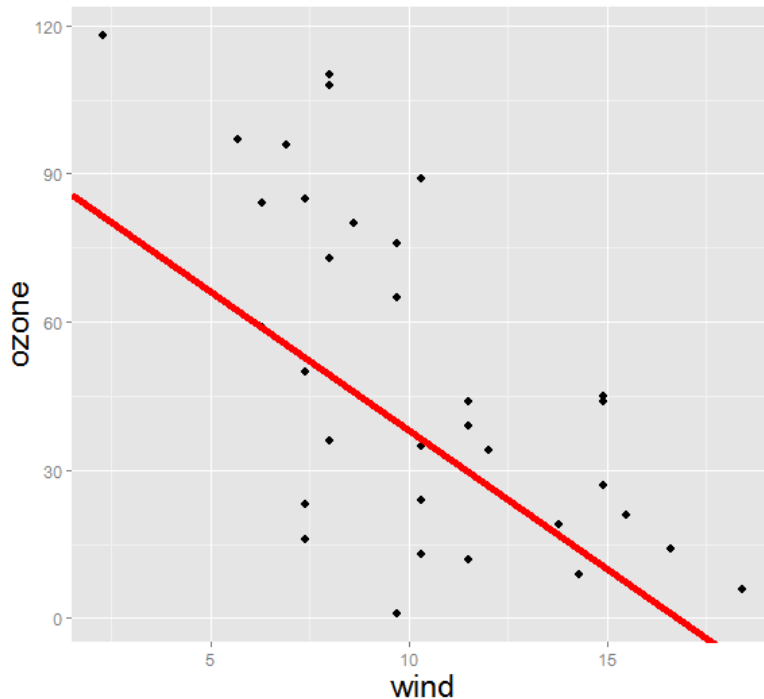
where

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

- 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"

R² Example

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



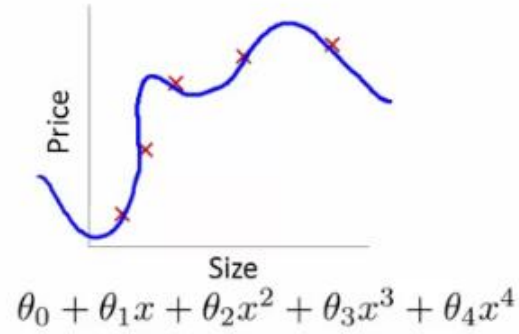
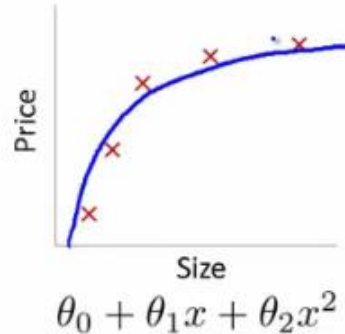
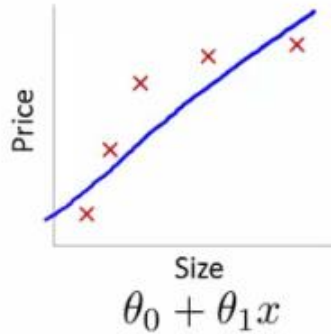
Agenda

- 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.

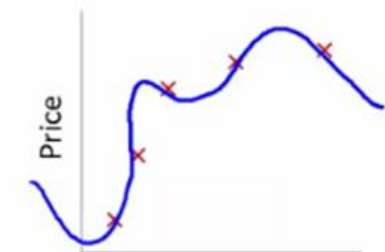
Overfitting



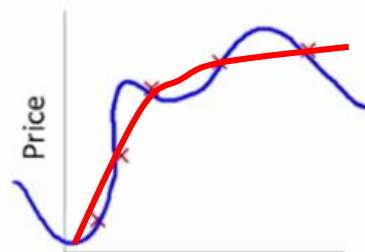
Complexity

- 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



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



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

Ensure small

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

$$J'(\theta) = J(\theta) + \text{Penalty}$$

Definitions

- Two most common

- L1 regularization

- lasso regression

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

- L2 regularization

- ridge regression
- weight decay

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

Regularized Regression

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

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

- 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

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

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

- Size of λ important
 - λ too high \Rightarrow no fitting
 - λ too low \Rightarrow no regularization

QUESTIONS