

Introduction to Regression

Agenda

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

Agenda

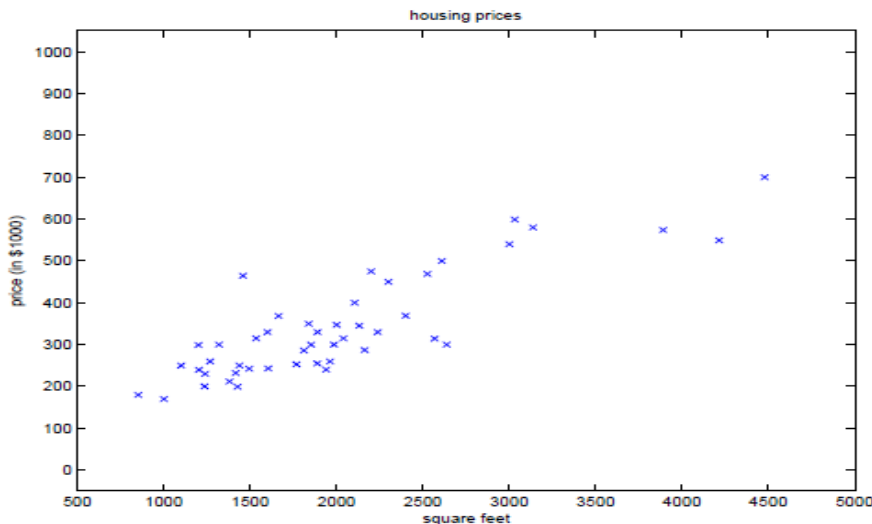
- **Linear Regression**
- Cost Functions
- Gradient Descent
- Hands-on Example
- Evaluating Regression Models
- Regularization

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\$)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



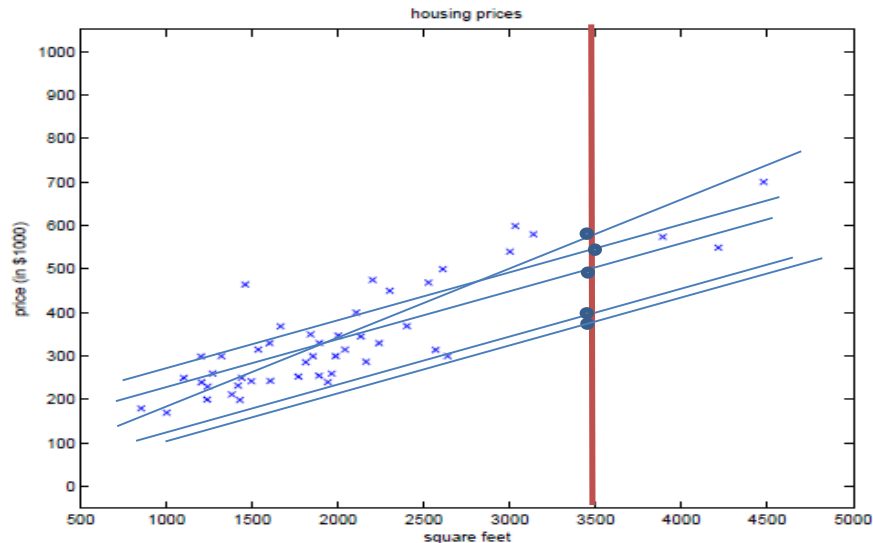
Can we learn to predict the price for a house not in the dataset?

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"?



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

Sidebar: Notation

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

Sidebar: Notation

- h_{θ} – A specific hypothesis (guess) for the map $X \rightarrow Y$
- θ – A vector of model parameters which define a specific hypothesis

Linear Regression Hypothesis Function

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

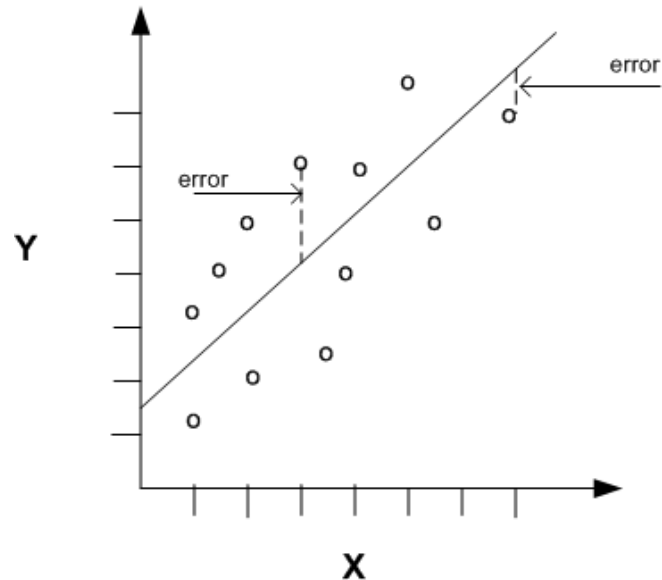
- $h_{\theta}(x^i)$ is the prediction of the hypothesis h_{θ} for a specific object x^i
- y^i is the true target value (equivalent to label in classification)
- We want the prediction $h_{\theta}(x^i)$ to be as close to the true label y^i as possible.
- How do we do that?

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"

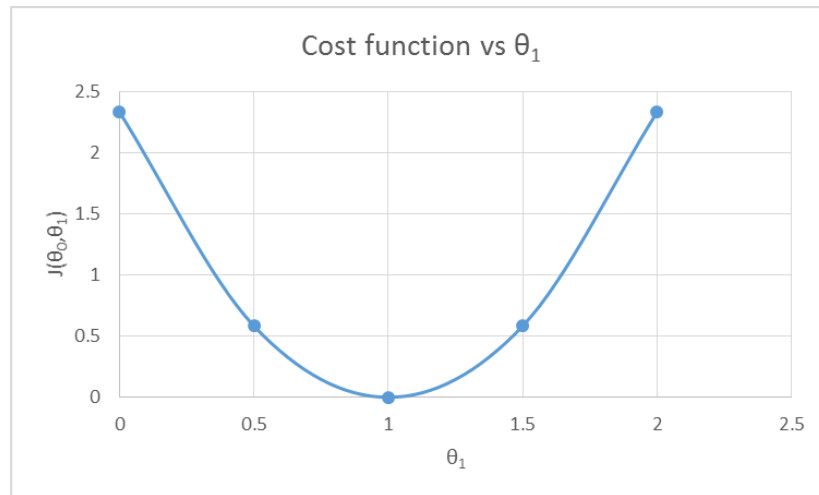
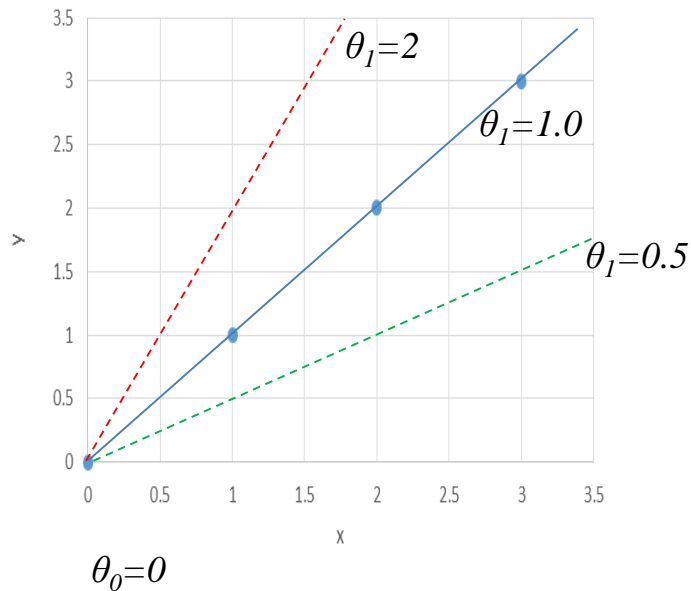


$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^i) - y^i)^2$$

Cost Function – 2D

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

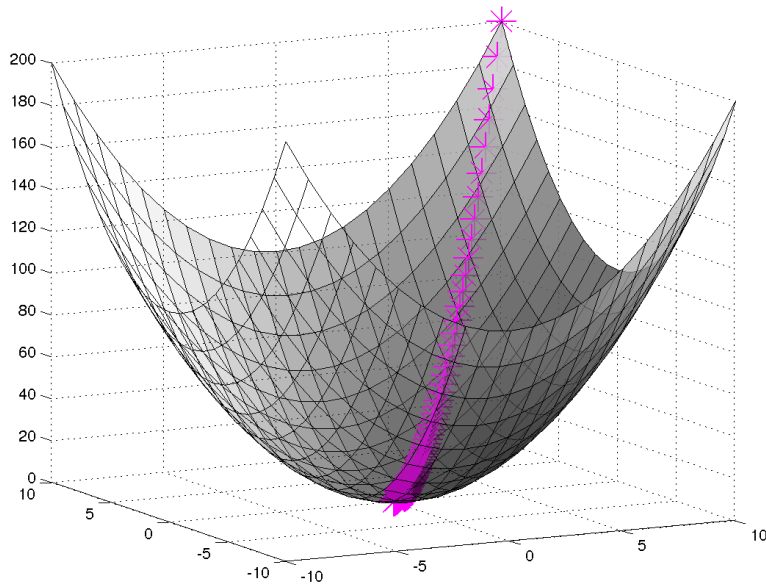
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^i) - y^i)^2$$



The Cost Function – 3D

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

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



Agenda

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

Gradient Descent

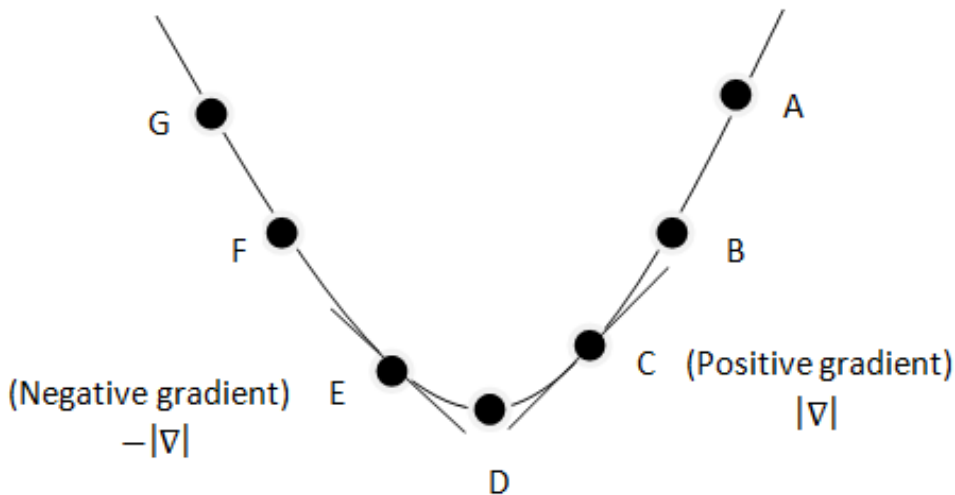
- Goal : minimize $J(\theta)$
- Gradient descent starts with some initial θ and then performs an update on each θ_j in turn:

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

- Repeat until θ converges

Intuition

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



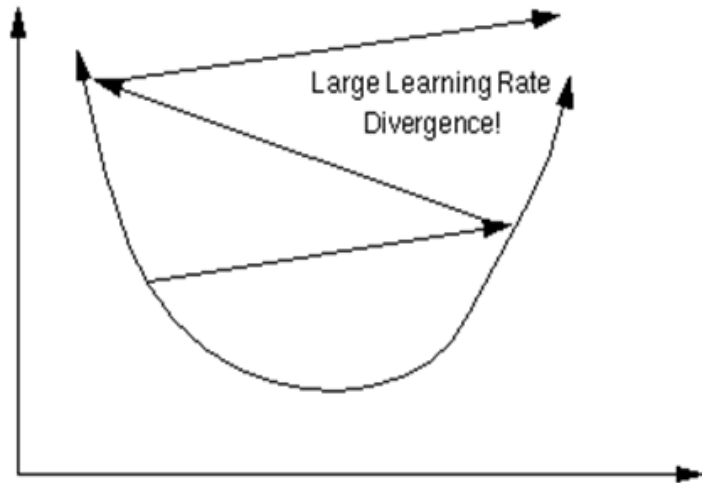
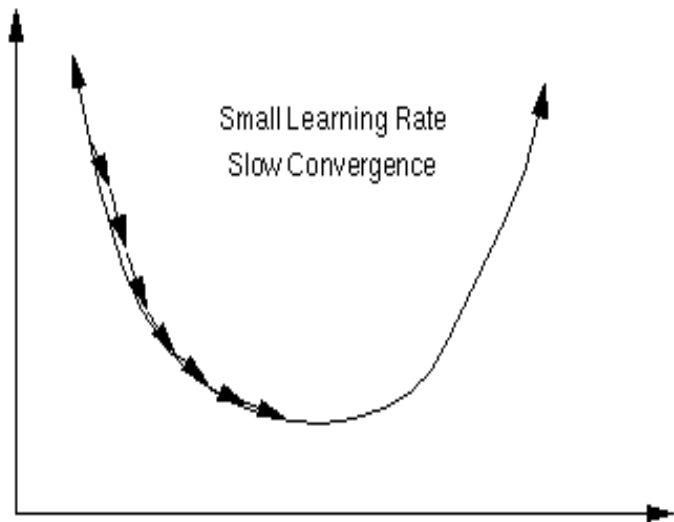
Gradient Descent Algorithm

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

- 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

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



Gradient Descent Animations

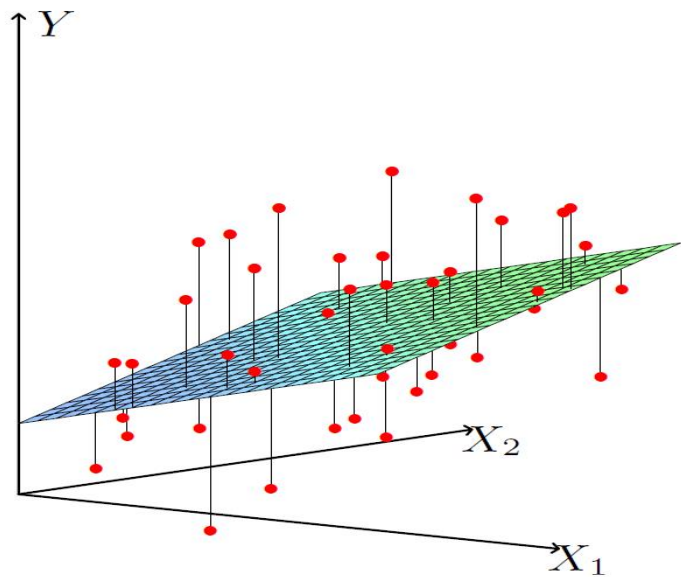
Sidebar: Regression with > 1 feature

x_1^i Living area (feet ²)	x_2^i #bedrooms	y^i Price (1000\$)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
\vdots	\vdots	\vdots

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

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

Sidebar: 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:

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

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:

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

}

Stochastic Gradient Descent

- Consider an alternative algorithm:

Repeat until convergence {

for i from 1 to m:

for j from 0 to n:

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

}

- 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:

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

}

- 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 m:

for j from 0 to n:

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

}

- 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

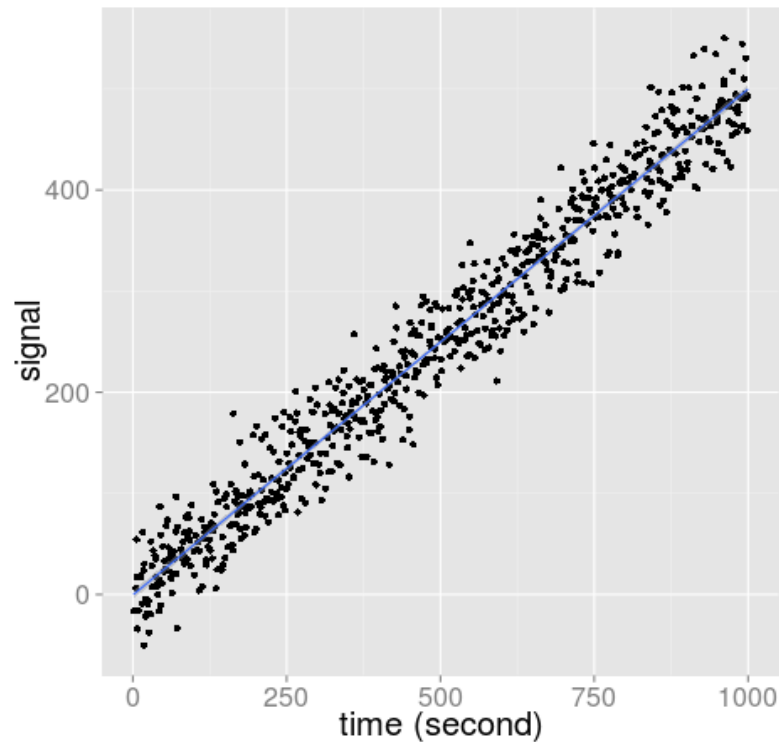
- Linear Regression
- Cost Functions
- Gradient Descent
- **Hands-on Example**
- Evaluating Regression Models
- Regularization

Agenda

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

Toy Example

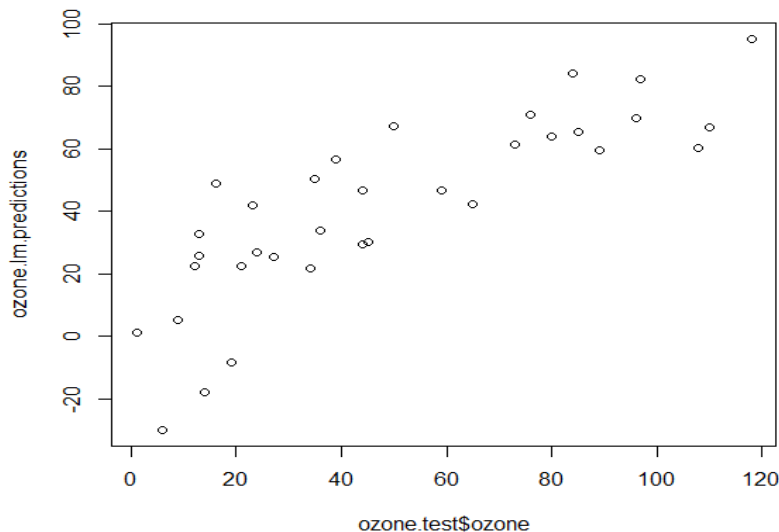
- Suppose we have a model:
 - $h(t) = -0.498 + 0.5 * t$
- How do we evaluate?
 - Train/Test Split
- What metrics to use?



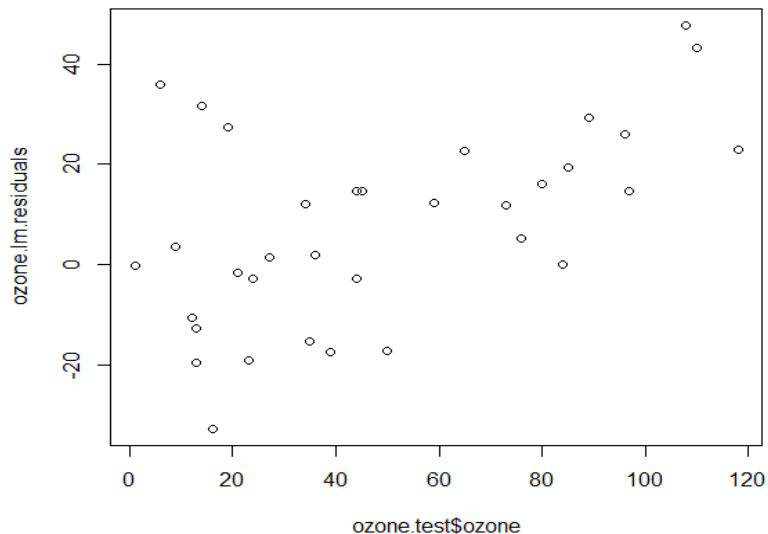
Graphical Evaluation Methods

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

Actual vs Predicted



Actual 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

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

- Mean of residual values
- "Pure" measure of error
- Somewhat resistant to outliers

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

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

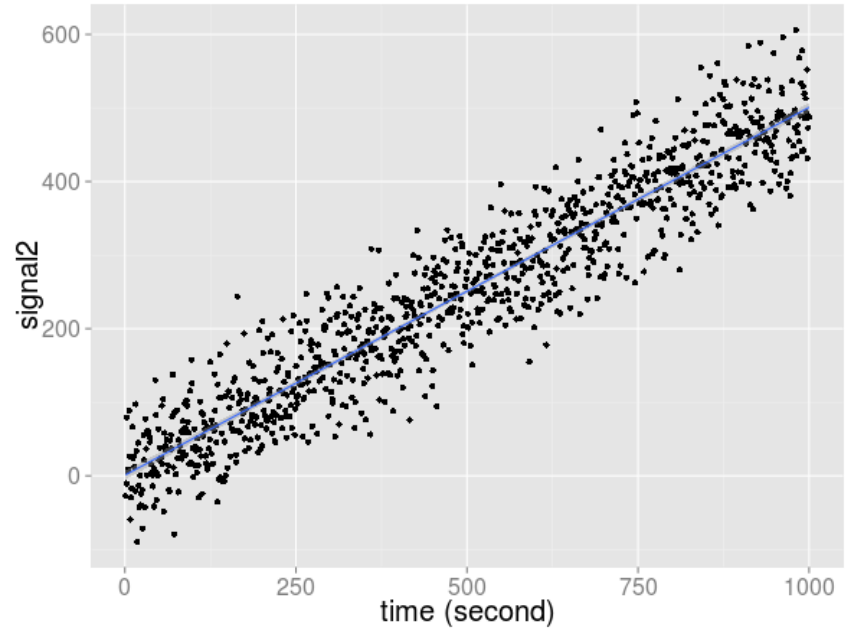
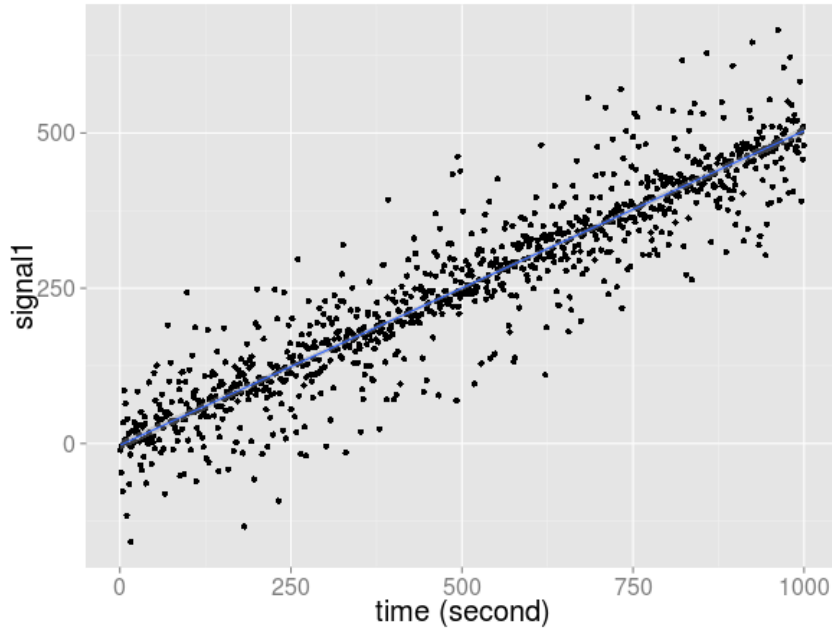
$$h_{\theta}(x) = \{0.5, 1.0, 1.5, 4.5, 5.0, 6.5\}$$

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

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

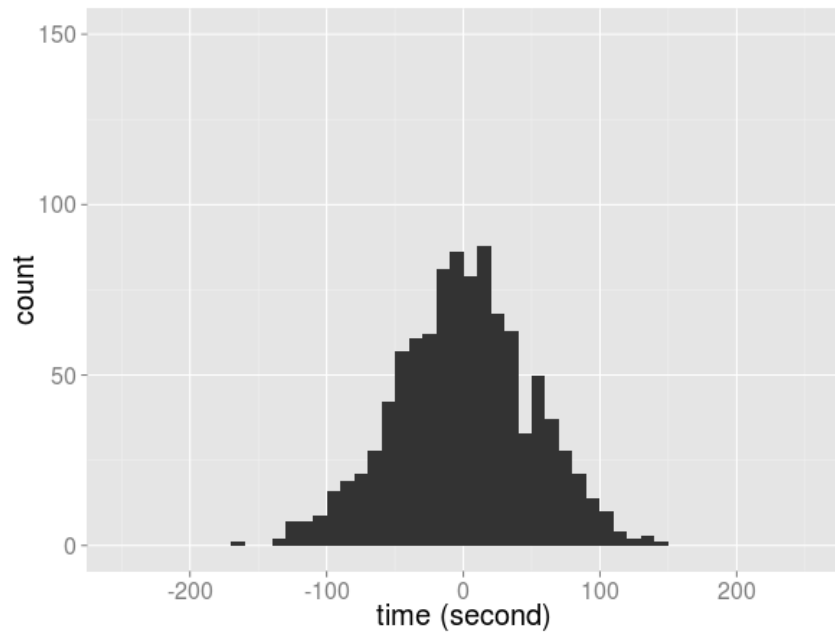
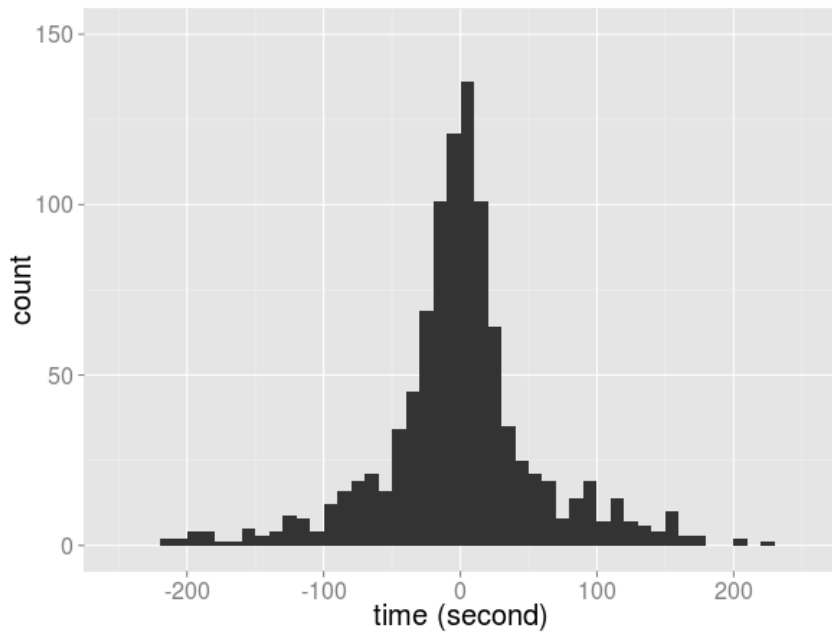
MAE vs RMSE

Signal1 and signal2

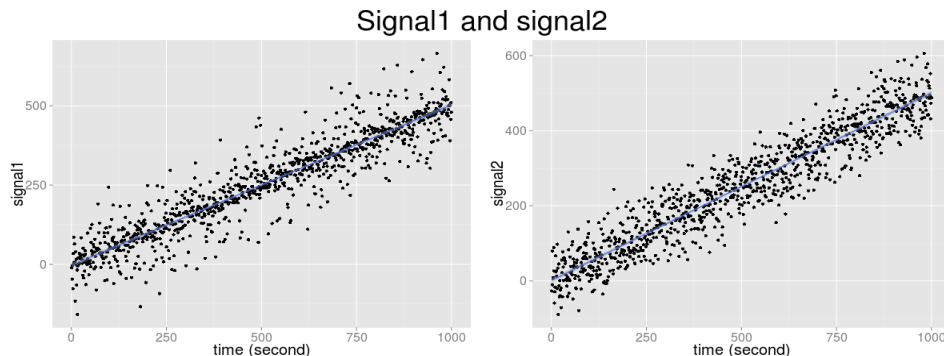


MAE vs RMSE

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)

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

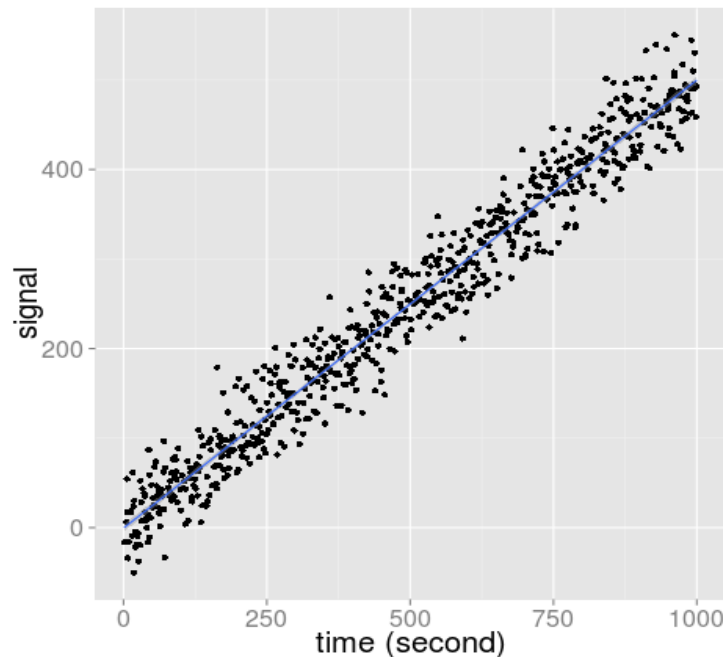
where

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

- 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

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



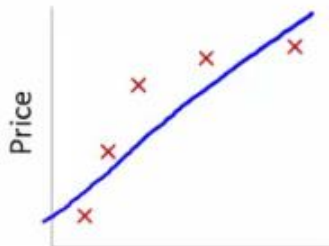
Agenda

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

Overfitting

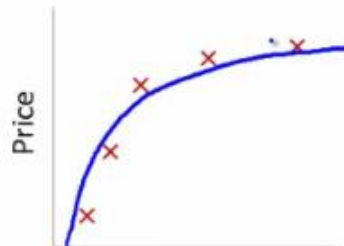
- Want to extract general trends
- Danger: "memorizing" the training set
- A model is **overfit** when model performance on test set is much worse than on training set.

Overfitting



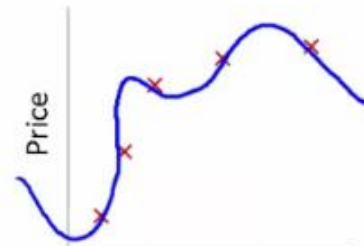
$$\theta_0 + \theta_1 x$$

High bias
(underfit)



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

"Just right"



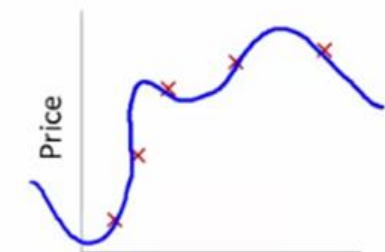
$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

High variance
(overfit)

Complexity

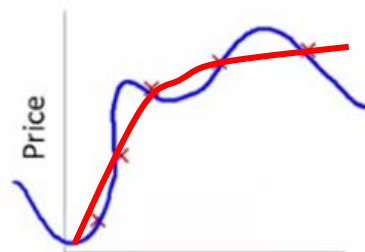
- What makes a good model overfit?
 - Training set limitations
 - **Complexity**
- How do we handle these?
 - Cross validation, etc
 - Manual model constraint
 - Regularization

Intuition



Size

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



Size

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

Ensure small

- Want to discourage complex models – 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
 - L2 regularization
 - ridge regression
 - weight decay

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

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

Regularized Regression

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

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

- 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

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

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

- Size of λ important
 - Too large: overwhelm affect of residuals
 - Too small: effectively no regularization

QUESTIONS