Linear Regression II

Cornelia Caragea

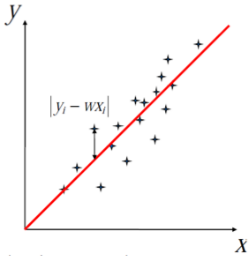
Department of Computer Science and Engineering
University of North Texas

Acknowledgments: Piyush Rai, Andrew Ng

June 22, 2016

Recap: Linear Regression: One-Dimensional Case

- **Given:** a set of N input-target pairs
- **Goal:** Model the relationship between x and y
- **Assumption:** the relationship between x and y is linear
  - Linear relationship - defined by a straight line with parameter w

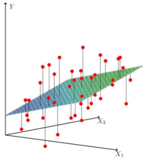


- The line may not fit the data *exactly*, but look for a reasonable approximation
- The total squared error: $E = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N}(y_i - wx_i)^2$
- The best fitting line is defined by w minimizing the total error E

Multivariate Linear Regression

Linear Regression: In Higher Dimensions

- **Analogy to line fitting**: In higher dimensions, fit hyperplanes
- For 2-dim. inputs, linear regression fits a 2-dim. plane to the data



- Many planes are possible. Which one is the best?
- **Intuition**: Choose the one closest to the targets y
  - Linear regression uses the sum-of-squared error notion of closeness
- Similar intuition carries over to higher dimensions too
  - Fitting a D-dimensional hyperplane to the data (hard to visualize)
- The hyperplane is defined by parameters $\mathbf{w}$ (a $D \times 1$ weight vector)

Example - House Price Prediction

Multiple features (variables):

Size ($feet^2$) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Notation:

- $n =$ number of features
- $x^{(i)} =$ input (features) of the i^{th} training example
- $x_j^{(i)} =$ value of feature j in the i^{th} training example

Model Representation

Previously: $h_w(x) = w_0 + w_1 x$

$$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$

Model Representation

Previously: $h_w(x) = w_0 + w_1 x$

$$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$
$$h_w(x) = 80 + 0.1 x_1 + 0.001 x_2 + 3 x_3 + 2 x_4$$

Model Representation

Previously: $h_w(x) = w_0 + w_1 x$

$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$
$h_w(x) = 80 + 0.1 x_1 + 0.001 x_2 + 3 x_3 + 2 x_4$

More generally,

$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$
For convenience of notation, define $x_0 = 1$. Hence,

$$h_w(x) = \sum_{j=0}^{n} w_j x_j = \mathbf{w}^T \mathbf{x} = [w_0 w_1 \cdots w_n] \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$$

Linear Regression: In Higher Dimensions (Formally)

- Given training data $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$
- Inputs $\mathbf{x}^{(i)}$: n-dimensional vectors (R^n), targets $y^{(i)}$: scalars (R)
- In the linear model: target is a linear function of the model parameters

$$y = h_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j x_j = w_0 + \mathbf{w}^T \mathbf{x}$$

- $\mathbf{x} = [x_1, \cdots, x_n]$
- w_j's and w_0 are the model parameters (w_0 is an offset)
    - $\mathbf{w} = [w_1, \cdots, w_n]$, the **weight vector** (to be learned from $\mathcal{D}$)
    - Parameters define the mapping from the inputs to targets

Linear Regression: In Higher Dimensions (Formally)

- Given training data $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$
- Inputs $\mathbf{x}^{(i)}$: n-dimensional vectors (R^n), targets $y^{(i)}$: scalars (R)
- In the linear model: target is a linear function of the model parameters

$$y = h_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j x_j = w_0 + \mathbf{w}^T \mathbf{x}$$

- $\mathbf{x} = [x_1, \cdots, x_n]$
- w_j's and w_0 are the model parameters (w_0 is an offset)
  - $\mathbf{w} = [w_1, \cdots, w_n]$, the **weight vector** (to be learned from $\mathcal{D}$)
  - Parameters define the mapping from the inputs to targets

How to choose $\mathbf{w}$?

Linear Regression: Gradient Descent Solution

- One solution: *Iterative* minimization of the cost function

  Cost Function:

  $$E(w_0, w_1, \cdots, w_n) = \frac{1}{2N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)} \right)^2,$$

  Goal:

  $$\min_{w_0, w_1, \cdots, w_n} E(w_0, w_1, \cdots, w_n),$$

- How: Using Gradient Descent (GD)
- A general recipe for iteratively optimizing similar loss functions
- Gradient Descent rule:
  - Initialize the weight vector $\mathbf{w} = \mathbf{w}^0$
  - Update $\mathbf{w}$ by moving along the direction of negative gradient $-\frac{\partial E}{\partial \mathbf{w}}$

Linear Regression: Gradient Descent Solution

- Initialize $\mathbf{w} = \mathbf{w}^0$
- Repeat until convergence:

$$
\begin{aligned}
w_j &:= w_j - \alpha \frac{\partial E}{\partial w_j} \\
&:= w_j - \alpha \frac{1}{N} \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}_j^{(i)}
\end{aligned}
$$

- Simultaneously update for every $j = 0, \cdots, n$
- α is the learning rate
- **Stop:** When some criteria is met (e.g., max. # of iterations), or the rate of decrease of E falls below some threshold.

Gradient Descent for Linear Regression

Gradient Descent for Univariate Linear Regression

$n = 1$

repeat until convergence {
$w_0 := w_0 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)} \right)$
$w_1 := w_1 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)} \right) x^{(i)}$
}

Update w_0 and w_1 simultaneously.

Gradient Descent for Multivariate Linear Regression

$n > 1$

repeat until convergence {
$w_0 := w_0 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)} \right) x_0^{(i)}$
$w_1 := w_1 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)} \right) x_1^{(i)}$
$w_2 := w_2 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)} \right) x_2^{(i)}$
$\vdots$
$w_n := w_n - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)} \right) x_n^{(i)}$
}

Update $w_0, w_1, \cdots, w_n$ simultaneously.

- To choose α, try:
  - $\cdots, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \cdots$

Polynomial Regression

Regression In Higher Dimensions - The More General Case

Nonlinear relationships between $\mathbf{x}$ and y can be modeled using some suitably chosen functions ϕ_j

- Given training data $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$
- Inputs $\mathbf{x}^{(i)}$: n-dimensional vectors ($\mathbb{R}^n$), targets $y^{(i)}$: scalars ($\mathbb{R}$)
- In the linear model: target is a linear function of the model parameters

$$y = h_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{j=1}^{m} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

- w_j's and w_0 are the model parameters (w_0 is an offset)
    - Parameters define the mapping from the inputs to targets
- $\phi = [\phi_0, \phi_1, \cdots, \phi_m]$
- Each ϕ_j is called a basis function
    - Allows change of representation of the input $\mathbf{x}$ (often desired)

Basis Function Expansions

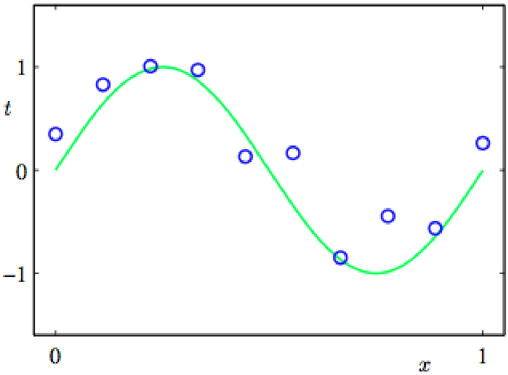
- Polynomial basis: $1, x, x^2, \cdots, x^n$

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Note that h is nonlinear in $\mathbf{x}$, but linear in $\mathbf{w}$

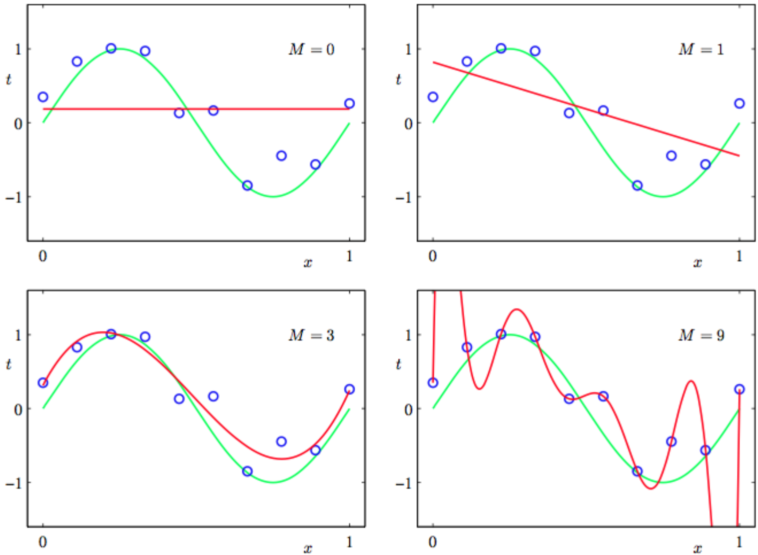
Regression

Plot of a training data set of $N = 10$ points, shown as blue circles, each comprising an observation of the input variable x along with the corresponding target variable t. The green curve shows the function $\sin(2\pi x)$ used to generate the data. Our goal is to predict the value of t for some new value of x, without knowledge of the green curve.



Fit a polynomial to the training data to determine the values of $\mathbf{w}$.

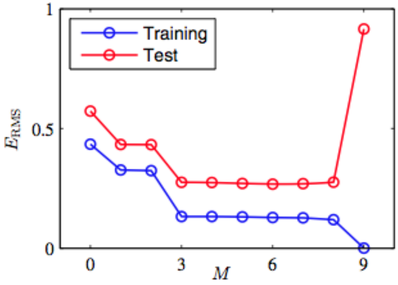
Polynomial Curve Fitting



Overfitting or Generalization

Generalization: the ability to correctly predict new examples that differ from those used for training

Overfitting: poor generalization
Performance on Test:



Summary

Linear Regression Model with Multiple Variables

- Model Representation
- How to choose a hypothesis?
- Polynomial Curve Fitting