Lecture 5 – Multivariate Linear Regression

Dan Sheldon

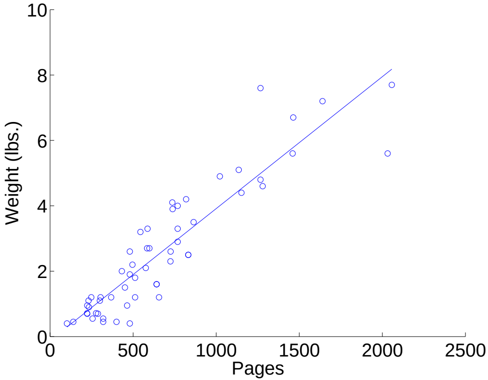
September 23, 2014

Topics

Multivariate linear regression

- Model
- Cost function
- Normal equations
- Gradient descent
- Features

Book Data



$$y = 0.004036x - .122291$$

Book Data

Can we predict better with multiple features?

Width	Thickness	Height	# Pages	Hardcover	Weight
8	1.8	10	1152	1	4.4
8	0.9	9	584	1	2.7
7	1.8	9.2	738	1	3.9
6.4	1.5	9.5	512	1	1.8

Book Data

Can we predict better with multiple features?

Width	Thickness	Height	# Pages	Hardcover	Weight
8	1.8	10	1152	1	4.4
8	0.9	9	584	1	2.7
7	1.8	9.2	738	1	3.9
6.4	1.5	9.5	512	1	1.8

Training data

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})$$

Book Data

Can we predict better with multiple features?

Width	Thickness	Height	# Pages	Hardcover	Weight
8	1.8	10	1152	1	4.4
8	0.9	9	584	1	2.7
7	1.8	9.2	738	1	3.9
6.4	1.5	9.5	512	1	1.8

Training data

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})$$

$\mathbf{x}^{(i)}$ is a **feature vector**

Multivariate Linear Regression

- Input: $\mathbf{x} \in \mathbb{R}^n$
- Output: $y \in \mathbb{R}$
- Model (hypothesis class): ?
- Cost function: ?

Model

$$h_{\boldsymbol{\theta}}(\mathbf{x}) =$$

Model

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Model

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{bmatrix} \theta_0 & \theta_1 & \ldots & \theta_n \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Model

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{bmatrix} \theta_0 & \theta_1 & \ldots & \theta_n \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} = \mathbf{x}^T \boldsymbol{\theta}$$

(Augment feature vector with 1)

Geometry of high dimensional linear (affine) functions

n-dimensional function $h_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \qquad \text{(linear)}$$
$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \qquad \text{(affine)}$$

Three facts on board

1. Contours = hyperplanes
2. Gradient = $\boldsymbol{\theta}$ (a vector, orthogonal to contours)
3. The norm $\|\boldsymbol{\theta}\|$ can be interpreted as slope

The Problem

Find $\boldsymbol{\theta}$ such that

$$y^{(i)} \approx h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), \qquad i = 1, \ldots, m$$

The Problem

Find $\boldsymbol{\theta}$ such that

$$y^{(i)} \approx h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), \qquad i = 1, \dots, m$$

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{bmatrix} \approx \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \dots & & & & \\ 1 & x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix}$$

The Problem

Find $\boldsymbol{\theta}$ such that

$$y^{(i)} \approx h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), \qquad i = 1, \dots, m$$

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{bmatrix} \approx \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \dots & & & & \\ 1 & x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix}$$

$$\mathbf{y} \approx X\boldsymbol{\theta}$$

Inputs: Data Matrix and Label Vector

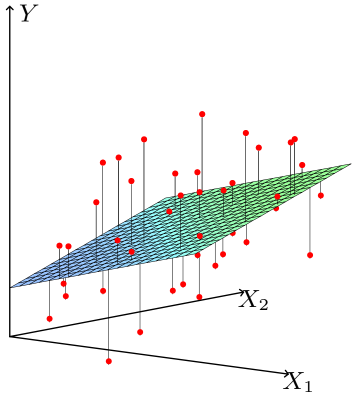
$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \dots & & & & \\ 1 & x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

Data matrix **Label vector**

Width	Thickness	Height	# Pages	Hardcover	Weight
8	1.8	10	1152	1	4.4
8	0.9	9	584	1	2.7
7	1.8	9.2	738	1	3.9
6.4	1.5	9.5	512	1	1.8

Illustration

Find $\boldsymbol{\theta}$ such that $y^{(i)} \approx h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), \qquad i = 1, \ldots, m$



Elements of Statistical Learning (2nd Ed.) ©Hastie, Tibshirani & Friedman 2009 Chap 3

Cost Function

$$J(\boldsymbol{\theta}) =$$

Cost Function

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

Exercise: write this succinctly in matrix-vector notation

Cost Function

Answer:

$$J(\boldsymbol{\theta}) = \frac{1}{2}(X\boldsymbol{\theta} - \mathbf{y})^T(X\boldsymbol{\theta} - \mathbf{y})$$

The Problem

Given training data X and $\mathbf{y}$, find $\boldsymbol{\theta}$ to minimize cost function:

$$J(\boldsymbol{\theta}) = \frac{1}{2}(X\boldsymbol{\theta} - \mathbf{y})^T(X\boldsymbol{\theta} - \mathbf{y})$$

Solution 1: Normal Equations

Normal equations

$$\boldsymbol{\theta} = (X^T X)^{-1} X^T \mathbf{y}$$

Heuristic derivation:

Proper Approach

- Set all partial derivatives to zero

$$0 = \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

- Solve a system of $n+1$ linear equations for $\theta_0, \ldots, \theta_n$
- Tedious, but leads to normal equations

Matrix Calculus

Succinct (and cool!) way to solve for normal equations:

$$0 = \nabla J(\boldsymbol{\theta}) \quad = \quad \frac{d}{d\boldsymbol{\theta}} \, \frac{1}{2}(X\boldsymbol{\theta} - \mathbf{y})^T(X\boldsymbol{\theta} - \mathbf{y})$$

Matrix Calculus

Succinct (and cool!) way to solve for normal equations:

$$
\begin{aligned}
0 = \nabla J(\boldsymbol{\theta}) &= \frac{d}{d\boldsymbol{\theta}} \, \frac{1}{2}(X\boldsymbol{\theta} - \mathbf{y})^T(X\boldsymbol{\theta} - \mathbf{y}) \\
0 &= (X\boldsymbol{\theta} - \mathbf{y})^T X
\end{aligned}
$$

Matrix Calculus

Succinct (and cool!) way to solve for normal equations:

$$
\begin{aligned}
0 = \nabla J(\boldsymbol{\theta}) &= \frac{d}{d\boldsymbol{\theta}} \frac{1}{2}(X\boldsymbol{\theta} - \mathbf{y})^T (X\boldsymbol{\theta} - \mathbf{y}) \\
0 &= (X\boldsymbol{\theta} - \mathbf{y})^T X \\
0 &= X^T (X\boldsymbol{\theta} - \mathbf{y})
\end{aligned}
$$

Matrix Calculus

Succinct (and cool!) way to solve for normal equations:

$$\begin{aligned}
0 = \nabla J(\boldsymbol{\theta}) &= \frac{d}{d\boldsymbol{\theta}} \frac{1}{2}(X\boldsymbol{\theta} - \mathbf{y})^T (X\boldsymbol{\theta} - \mathbf{y}) \\
0 &= (X\boldsymbol{\theta} - \mathbf{y})^T X \\
0 &= X^T (X\boldsymbol{\theta} - \mathbf{y}) \\
X^T X\boldsymbol{\theta} &= X^T \mathbf{y}
\end{aligned}$$

Matrix Calculus

Succinct (and cool!) way to solve for normal equations:

$$
\begin{aligned}
0 = \nabla J(\boldsymbol{\theta}) &= \frac{d}{d\boldsymbol{\theta}}\ \frac{1}{2}(X\boldsymbol{\theta} - \mathbf{y})^T(X\boldsymbol{\theta} - \mathbf{y}) \\
0 &= (X\boldsymbol{\theta} - \mathbf{y})^T X \\
0 &= X^T(X\boldsymbol{\theta} - \mathbf{y}) \\
X^T X\boldsymbol{\theta} &= X^T\mathbf{y} \\
\boldsymbol{\theta} &= (X^T X)^{-1}X^T\mathbf{y}
\end{aligned}
$$

(Note: not responsible vector derivative in first line, but should understand rest of derivation.)

Solution 2: Gradient Descent

1. Initialize $\theta_0, \theta_1, \ldots, \theta_n$ arbitrarily
2. Repeat until convergence

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}), \qquad j = 0, \ldots, n.$$

Solution 2: Gradient Descent

1. Initialize $\theta_0, \theta_1, \ldots, \theta_n$ arbitrarily
2. Repeat until convergence

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}), \qquad j = 0, \ldots, n.$$

Partial derivatives:

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

Vectorized Gradient Descent

1. Initialize $\boldsymbol{\theta}$ arbitrarily
2. Repeat until convergence

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \underbrace{X^T(X\boldsymbol{\theta} - \mathbf{y})}_{\nabla J(\boldsymbol{\theta})}$$

Feature Normalization

- Demo: Problem 3 from HW0
- Advice: normalize your features so they have the similar numeric ranges!

Feature Normalization

For each feature j, compute the mean μ_j and standard deviation σ_j of that feature over training set.

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}, \qquad \sigma_j = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2}$$

Feature Normalization

For each feature j, compute the mean μ_j and standard deviation σ_j of that feature over training set.

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}, \qquad \sigma_j = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2}$$

Then, subtract mean and divide by standard deviation:

$$x_j^{(i)} \leftarrow (x_j^{(i)} - \mu_j)/\sigma_j$$

Feature Normalization

For each feature j, compute the mean μ_j and standard deviation σ_j of that feature over training set.

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}, \qquad \sigma_j = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2}$$

Then, subtract mean and divide by standard deviation:

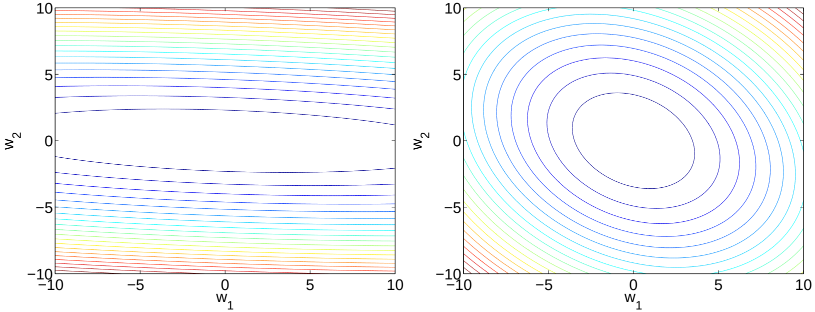
$$x_j^{(i)} \leftarrow (x_j^{(i)} - \mu_j)/\sigma_j$$

Effect: adjust columns of data matrix to have mean zero and standard deviation equal to one. E.g.

$$\begin{bmatrix} 94 & .99 \\ 116 & 1 \\ 83 & 1.01 \end{bmatrix} \mapsto \begin{bmatrix} -0.22 & -1 \\ 1.09 & 0 \\ -0.87 & 1 \end{bmatrix}$$

Feature Normalization

Example: cost function contours before and after normalization



Feature Design

It is possible to fit *nonlinear* functions using linear regression:

$$(x_1, x_2, x_3) \mapsto (x_1, x_2, x_3, x_1^2, \log(x_2), x_1 + x_3)$$

Approaches

- Try standard transformations
- Design features you think will work

Polynomial Regression

$$x \mapsto (1, x, x^2, x^3, \ldots)$$