Lecture 4-5 – Multivariate Linear Regression

ECE597ML-697ML

Mario Parente

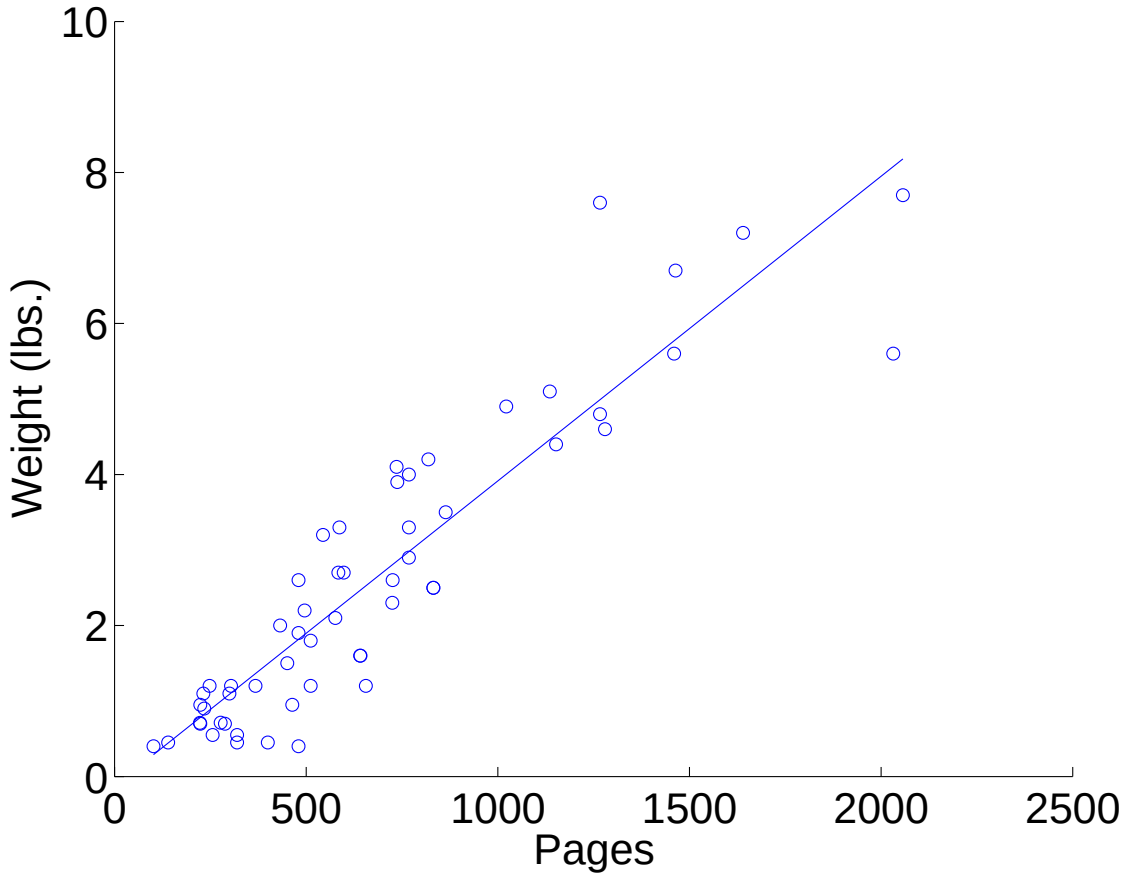
Linear algebra wrap-up

- Inverse (other slides)

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: $\mathbf{x} \in \mathbb{R}^d$
- 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

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

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

Model

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

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

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

(Augment feature vector with 1)

Geometry of high dimensional linear functions

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

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

MATLAB demo

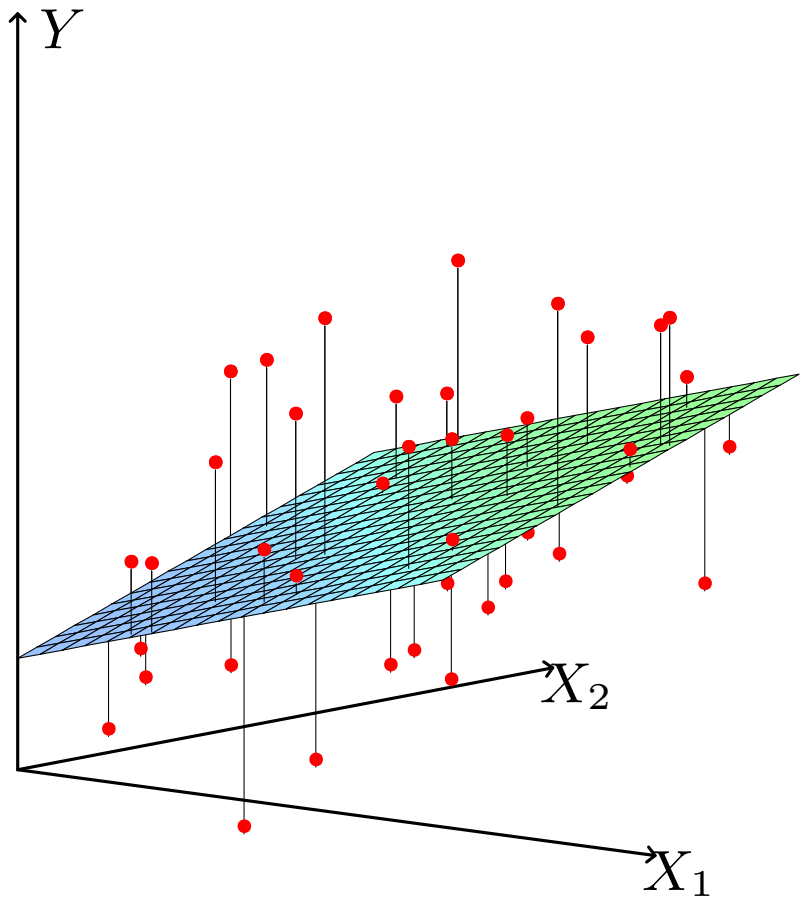
- Contours = parallel hyperplanes

$$\theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n = \text{constant}$$

- Gradient = $\boldsymbol{\theta}$ (a vector, orthogonal to contours)
- The norm $\|\boldsymbol{\theta}\|$ can be interpreted as slope

Illustration

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

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

The Problem

Find $\boldsymbol{\theta}$ such that

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

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

The Problem

Find $\boldsymbol{\theta}$ such that

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

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

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

Inputs: Data Matrix and Label Vector

Data matrix, label vector

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

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

Cost Function

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

Cost Function

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

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:

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

Cost Function

Answer:

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

Easy knowing that if $\mathbf{z} = [z_1, \ldots, z_m]$ then $||\mathbf{z}||^2 = \sum_{i=1}^{m} z_i^2 = \mathbf{z}^T\mathbf{z}$ and set $z_i = h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} = {\mathbf{x}^{(i)}}^T\boldsymbol{\theta} - y^{(i)}$

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

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

Derivation:

- 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

Gradient!

Definition: gradient = vector of partial derivatives:

$$\nabla J(\boldsymbol{\theta}) := \begin{bmatrix} \frac{\partial}{\partial \theta_0} J(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} J(\boldsymbol{\theta}) \end{bmatrix}$$

Gradient!

Definition: gradient = vector of partial derivatives:

$$\nabla J(\boldsymbol{\theta}) := \begin{bmatrix} \frac{\partial}{\partial \theta_0} J(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} J(\boldsymbol{\theta}) \end{bmatrix}$$

- Also written as $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ or $\frac{d}{d\boldsymbol{\theta}} J(\boldsymbol{\theta})$

Gradient!

Definition: gradient = vector of partial derivatives:

$$\nabla J(\boldsymbol{\theta}) := \begin{bmatrix} \frac{\partial}{\partial \theta_0} J(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} J(\boldsymbol{\theta}) \end{bmatrix}$$

- Also written as $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ or $\frac{d}{d\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Fact: the gradient is a *vector* that points in the steepest uphill direction

Aside: Matrix Calculus

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

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

Aside: Matrix Calculus

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

$$\begin{aligned} J(\theta) &= \frac{1}{2}(X\boldsymbol{\theta} - \mathbf{y})^T(X\boldsymbol{\theta} - \mathbf{y}) \\ &= \frac{1}{2}\sum_{i=1}^{m}\left((X\boldsymbol{\theta})_i - y^{(i)}\right)^2 \end{aligned}$$

Aside: Matrix Calculus

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

$$\begin{aligned} J(\theta) &= \frac{1}{2}(X\boldsymbol{\theta} - \mathbf{y})^T(X\boldsymbol{\theta} - \mathbf{y}) \\ &= \frac{1}{2}\sum_{i=1}^{m}\left((X\boldsymbol{\theta})_i - y^{(i)}\right)^2 \\ &= \frac{1}{2}\sum_{i=1}^{m}\left(\sum_{j=1}^{n+1} X_{ij}\tilde{\theta}_j - y^{(i)}\right)^2 \end{aligned}$$

where $\tilde{\theta}_j = \theta_{j-1}$ and $X_{i\cdot} = [1\ x_1^{(i)}\ \ldots\ x_n^{(i)}]$.

Aside: Matrix Calculus (cont)

Then by normal calculus, the kth partial derivative can be derived as below.

$$\frac{\partial J}{\partial \tilde{\theta}_k} = \frac{1}{2}\sum_{i=1}^{m} 2\left(\sum_{j=1}^{n+1} X_{ij}\tilde{\theta}_j - y^{(i)}\right)\frac{\partial}{\partial \tilde{\theta}_k}\left(\sum_{j=1}^{n+1} X_{ij}\tilde{\theta}_j - y^{(i)}\right)$$

Aside: Matrix Calculus (cont)

Then by normal calculus, the kth partial derivative can be derived as below.

$$\begin{aligned}
\frac{\partial J}{\partial \tilde{\theta}_k} &= \frac{1}{2}\sum_{i=1}^{m} 2\left(\sum_{j=1}^{n+1} X_{ij}\tilde{\theta}_j - y^{(i)}\right)\frac{\partial}{\partial \tilde{\theta}_k}\left(\sum_{j=1}^{n+1} X_{ij}\tilde{\theta}_j - y^{(i)}\right) \\
&= \sum_{i=1}^{m}\left(\sum_{j=1}^{n+1} X_{ij}\tilde{\theta}_j - y^{(i)}\right) X_{ik}
\end{aligned}$$

Aside: Matrix Calculus (cont)

Then by normal calculus, the kth partial derivative can be derived as below.

$$\begin{aligned}
\frac{\partial J}{\partial \tilde{\theta}_k} &= \frac{1}{2}\sum_{i=1}^{m} 2\left(\sum_{j=1}^{n+1} X_{ij}\tilde{\theta}_j - y^{(i)}\right)\frac{\partial}{\partial \tilde{\theta}_k}\left(\sum_{j=1}^{n+1} X_{ij}\tilde{\theta}_j - y^{(i)}\right) \\
&= \sum_{i=1}^{m}\left(\sum_{j=1}^{n+1} X_{ij}\tilde{\theta}_j - y^{(i)}\right) X_{ik} \\
&= X_{.k}^{T}\left(X\boldsymbol{\theta} - \mathbf{y}\right)
\end{aligned}$$

where $X_{.k}^{T}$ is the k-th row of X^T

By combining all k into a vector, we get the gradient to be

$$\nabla J(\boldsymbol{\theta}) = X^{T}\left(X\boldsymbol{\theta} - \mathbf{y}\right)$$

Aside: Matrix Calculus (cont)

if I set the grad equal to the zero vector

$$\mathbf{0} \quad = \quad \nabla J(\boldsymbol{\theta})$$

Aside: Matrix Calculus (cont)

if I set the grad equal to the zero vector

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

Aside: Matrix Calculus (cont)

if I set the grad equal to the zero vector

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

Aside: Matrix Calculus (cont)

if I set the grad equal to the zero vector

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

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

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

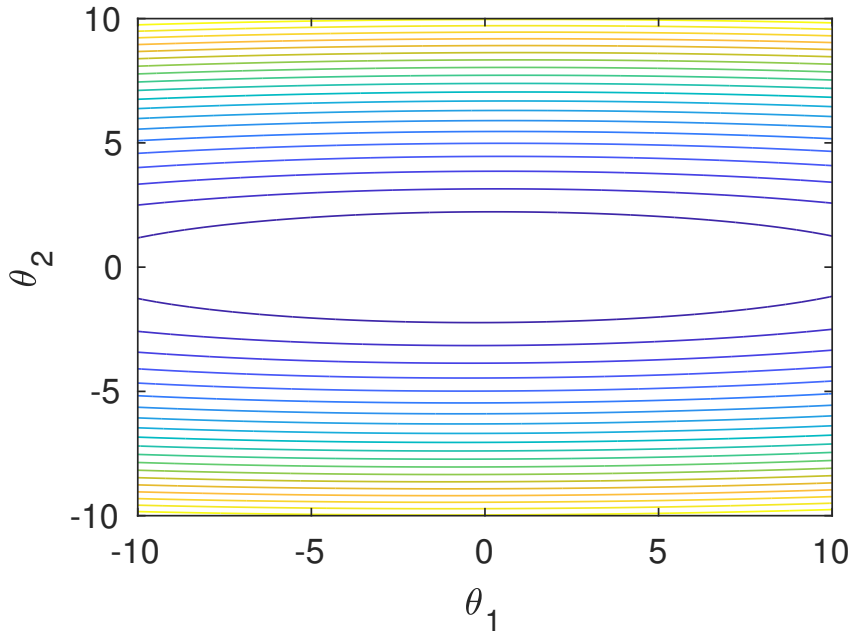
Feature Normalization

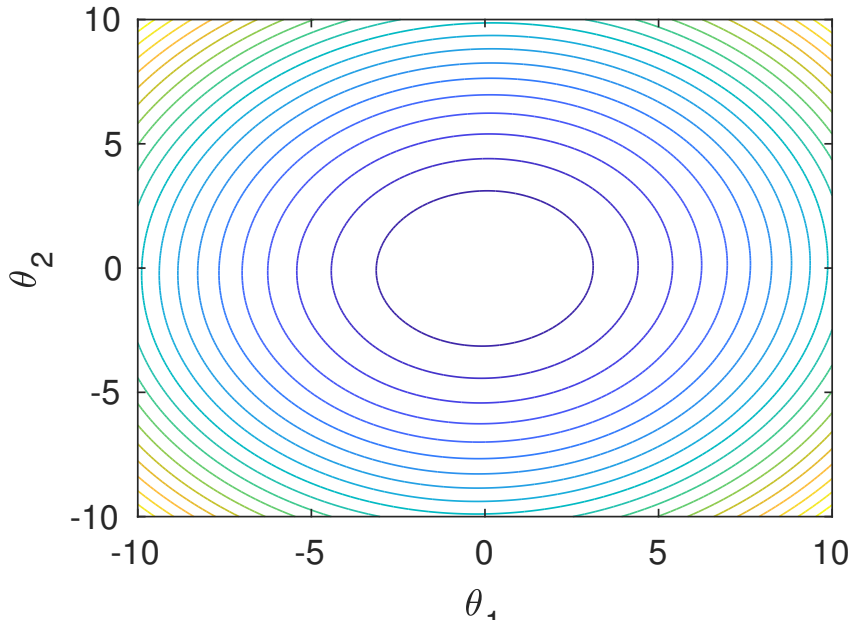
- Advice: normalize your features so they have the similar numeric ranges!

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

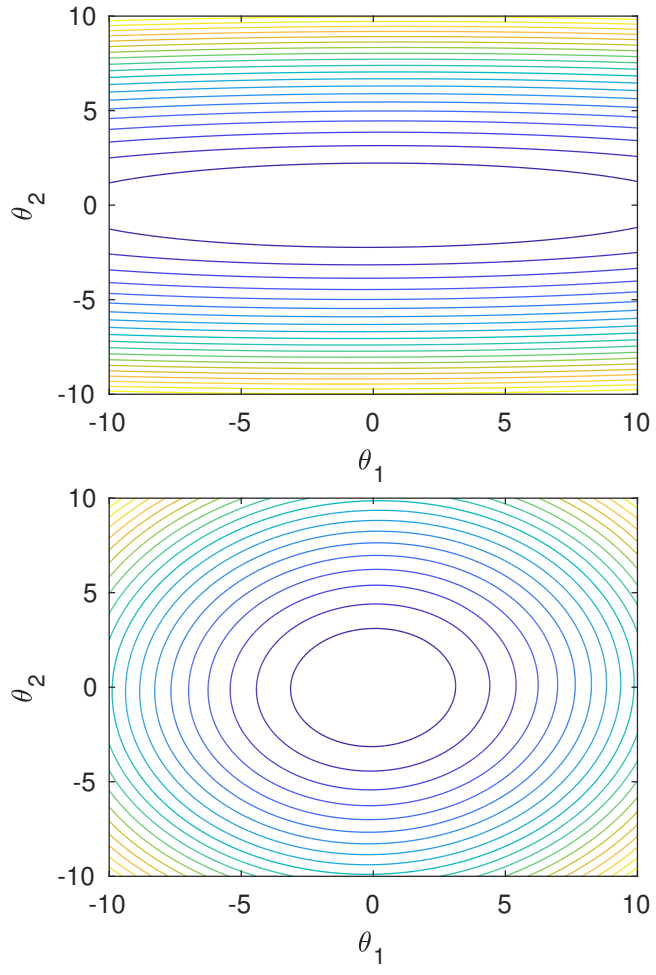
Feature Normalization: why?

Example: cost function contours before and after normalization





Feature Normalization: why?



Why is the bottom cost function better?

1. It has a smaller minimum value, which allows you to find a better hypothesis.
2. It is easier to optimize (e.g. can use bigger step size)
3. It is not better. The top one is better (explain why).

Feature Normalization: how?

Adjust columns of data matrix to have mean zero and standard deviation equal to one.

Feature Normalization: how?

Adjust columns of data matrix to have mean zero and standard deviation equal to one.

- 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: how?

Adjust columns of data matrix to have mean zero and standard deviation equal to one.

- 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 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

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