Linear Regression

Instructor: Jessica Wu -- Harvey Mudd College

The instructor gratefully acknowledges Andrew Ng (Stanford), Eric Eaton (UPenn), David Kauchak (Pomona), and the many others who made their course materials freely available online.

Linear Regression Setup

Learning Goals

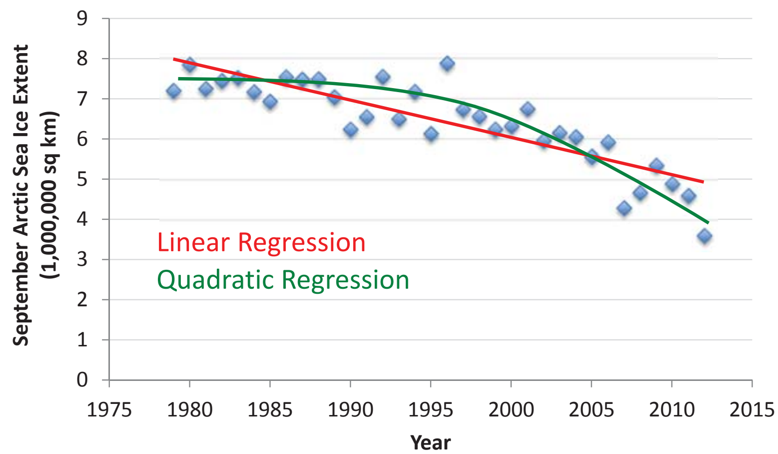
- Describe how regression differs from classification
- Describe the goal of linear regression

Regression

Given:

– Data $\mathbf{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$ where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$

– Corresponding labels $\boldsymbol{y} = \{y^{(1)}, \ldots, y^{(n)}\}$ where $y^{(i)} \in \mathbb{R}$



Based on slide by Eric Eaton
[Data from G. Witt. Journal of Statistics Education, Volume 21, Number 1 (2013)]

Linear Regression

Hypothesis

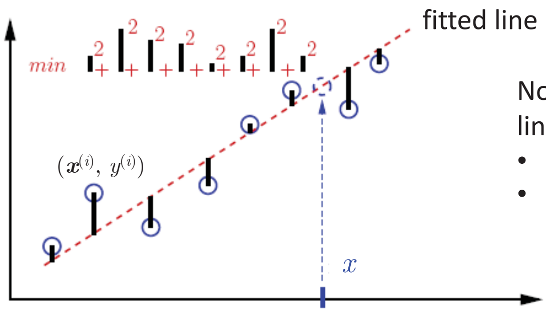
assume $x_0 = 1$

$$y = h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d$$

$$= \sum_{j=0}^{d} \theta_j x_j = \boldsymbol{\theta}^T \boldsymbol{x}$$

θ_j's are parameters (weights)

Fit model (find $\boldsymbol{\theta}$) by minimizing sum of squared errors



fitted line used as predictor

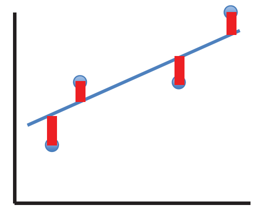
Note:
linear regression is a **parametric** method
- makes inferences about model
- fixed number of parameters (d+1)

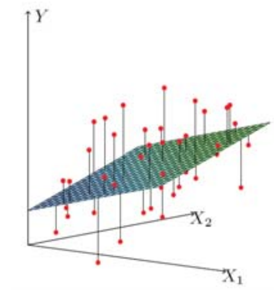
Based on slide by Eric Eaton [Figures courtesy of Greg Shakhnarovich]

Cost Function

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

Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$





More generally…

$$R_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\mathrm{Loss}\left(y^{(i)}, h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)$$

empirical risk / training error

$$z = y^{(i)} - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})$$

Q1: Why use squared error?
Q2: How do we optimize $\boldsymbol{\theta}$?

$$\mathrm{Loss}(z) = \frac{z^2}{2}$$

squared error

different people use $1/(2n)$ vs $1/2$

Based on slide by Eric Eaton

(This slide intentionally left blank.)

Probabilistic Interpretation

Learning Goals

- Describe why we minimize squared error

Probabilistic Interpretation

Assume $y^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} + \epsilon^{(i)}$

What is $\epsilon^{(i)}$?

- error term
- captures un-modeled effects (e.g. missing features)
- captures random noise

How can we model $\epsilon^{(i)}$?

Probabilistic Interpretation

(This slide intentionally left blank.)

Maximum Likelihood Estimate (MLE)

(This slide intentionally left blank.)

Solving Linear Regression

Learning Goals

- Describe shape of $J(\boldsymbol{\theta})$
- Describe two approaches for optimizing $\boldsymbol{\theta}$
  - Gradient Descent (stochastic and batch version)
  - Normal Equations
- Compare tradeoffs of GD vs Normal Equations

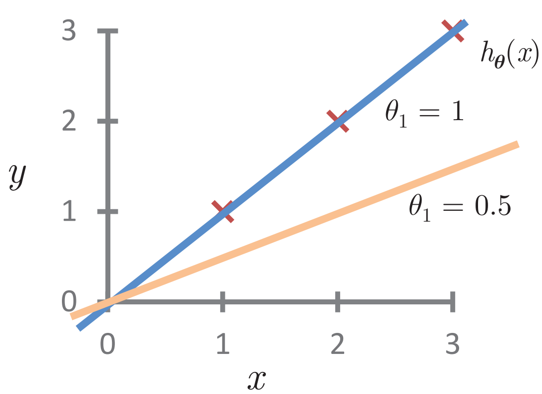
Intuition behind Cost Function

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

For insight on $J(\boldsymbol{\theta})$, assume $x^{(i)} \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]^T$. Also fix $\theta_0 = 0$.

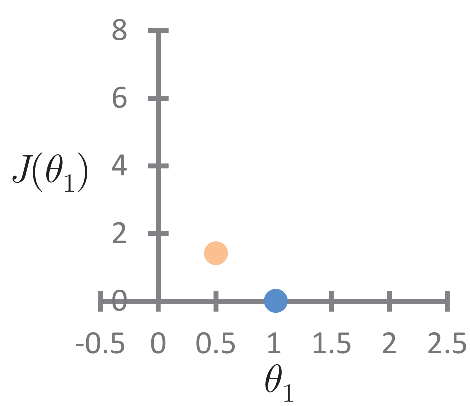
$h_{\boldsymbol{\theta}}(x)$

(for fixed θ_1, this is a function of x)



$J(\theta_1)$

(function of the parameter θ_1)

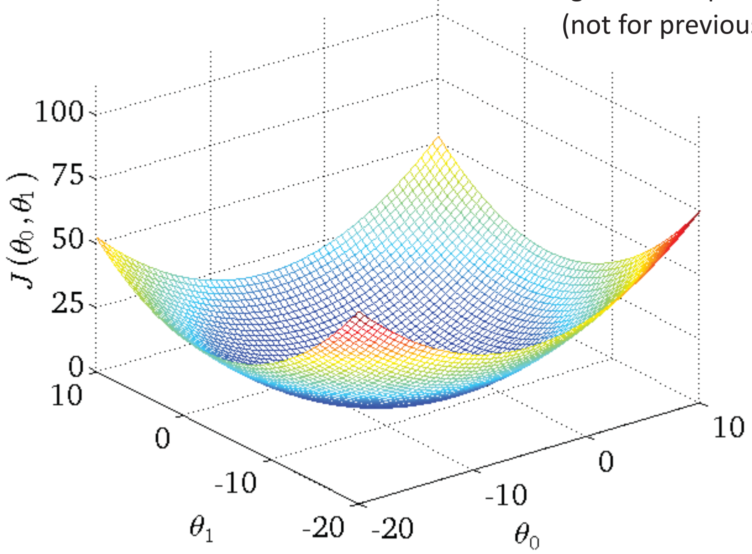


Based on slide by Eric Eaton
[Example by Andrew Ng]

$$J(0.5) = \tfrac{1}{2}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right] = 1.75$$

Intuition behind Cost Function

general shape of $J(\theta_0,\theta_1)$
(not for previous example)



Based on slide by Andrew Ng

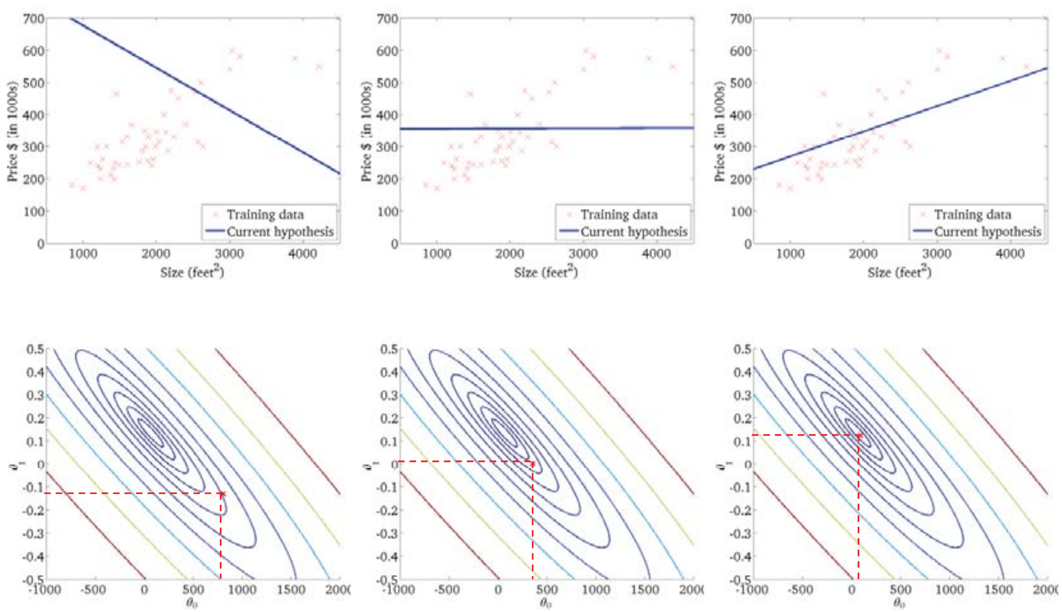
Intuition behind Cost Function

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

(for fixed (θ_0,θ_1), this is a fcn of x)

$J(\theta_0,\theta_1)$

(fcn of the parameters (θ_0,θ_1))



contour map

Based on slide by Andrew Ng

Solving Linear Regression

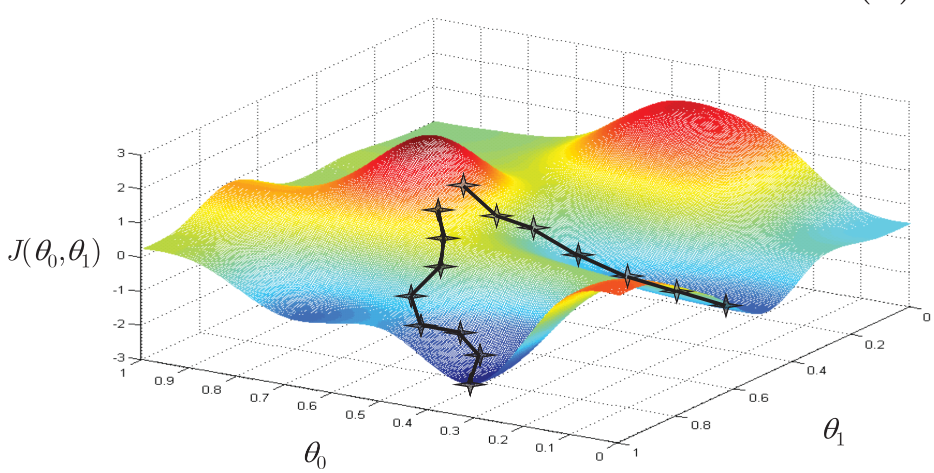
Learning Goals

- ✓ Describe shape of $J(\boldsymbol{\theta})$
- Describe two approaches for optimizing $\boldsymbol{\theta}$
  - Gradient Descent (stochastic and batch version)
  - Normal Equations
- Compare tradeoffs of GD vs Normal Equations

Basic Search Procedure

Idea

- Choose initial value for $\boldsymbol{\theta}$
- Until we reach minimum
  - Choose new value for $\boldsymbol{\theta}$ to reduce $J(\boldsymbol{\theta})$



since the least squares objective function is convex, we do not need to worry about local minima

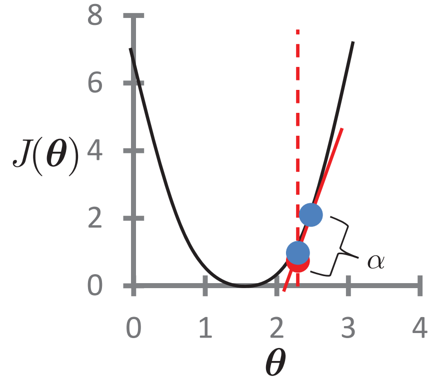
Based on slide by Andrew Ng

Approach 1: Gradient Descent

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

learning rate
(small, e.g., $\alpha = 0.05$)

update simultaneously for all $j = 0, \ldots, d$



For one training example $(\boldsymbol{x}, y)$

Based on slide by Eric Eaton

(This slide intentionally left blank.)

Batch versus Stochastic Mode

Batch Gradient Descent (BGD)

repeat until convergence {

$$\theta_j \leftarrow \theta_j - \alpha \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

}

converged when $||\boldsymbol{\theta}_{\text{new}} - \boldsymbol{\theta}_{\text{old}}||_2 < \epsilon$

for every *j* **simultaneously**

To achieve simultaneous update:

- At start of each GD iteration, compute $h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})$
- Use this stored value in update step

Stochastic Gradient Descent (SGD)

repeat until convergence {

for $i = 1$ to n {

$$\theta_j \leftarrow \theta_j - \alpha \left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

}

}

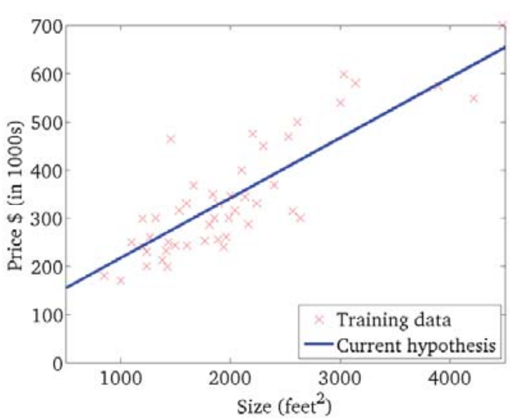
often will randomly shuffle points

for every *j* **simultaneously**

Gradient Descent Example

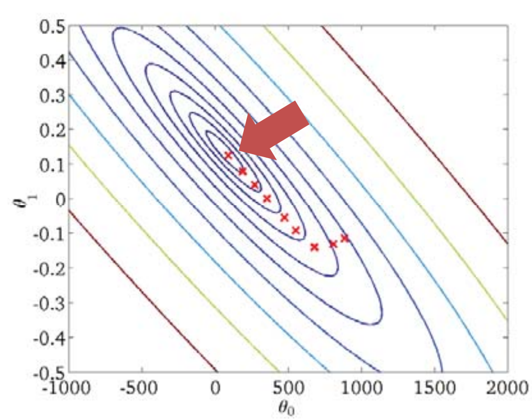
$h_{\boldsymbol{\theta}}(x)$

(for fixed (θ_0,θ_1), this is a function of x)



$J(\theta_0,\theta_1)$

(function of the parameter (θ_0,θ_1))



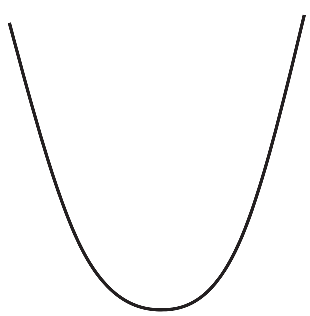
Based on slide by Andrew Ng

Batch versus Stochastic GD

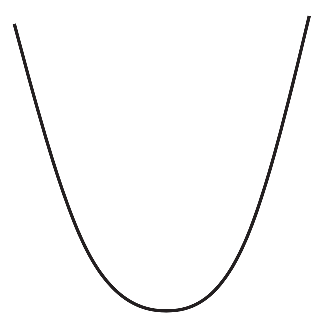
- Batch has to scan through entire training set before taking single step (costly if n is large)
- Stochastic makes progress "right away"
  - often $\boldsymbol{\theta}$ gets "close" to minimum much faster
    - but $\boldsymbol{\theta}$ may never converge if it oscillates around $\arg\min J(\boldsymbol{\theta})$
  - in practice, most values near minimum will be good enough approximation to true minimum
    - alternatively, we can guarantee convergence if we allow step size α to decrease, e.g. $\alpha_k = \frac{1}{1+k}$, where k = outer loop iteration

Choosing step size α

α too small



α too large



To check if GD is working, print out $J(\boldsymbol{\theta})$ each iteration
- value should decrease each iteration
- if it does not, adjust α

Based on slide by Eric Eaton
[Originally by Andrew Ng]

Solving Linear Regression

Learning Goals

✓ Describe shape of $J(\boldsymbol{\theta})$

- Describe two approaches for optimizing $\boldsymbol{\theta}$
  - ✓ Gradient Descent (stochastic and batch version)
  - – Normal Equations
- Compare tradeoffs of GD vs Normal Equations

Approach 2: Normal Equations

(closed form solution)

Vectorization

- more compact equations
- faster code (using optimized matrix libraries)

Let us consider our model

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j$$

Let $\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$ and $\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$. Then $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}$.

Matrix-Vector Form

For n instances:

$$h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) = \sum_{j=0}^{d} \theta_j x_j^{(i)}$$

Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} - & \left(\boldsymbol{x}^{(1)}\right)^T & - \\ & \vdots & \\ - & \left(\boldsymbol{x}^{(i)}\right)^T & - \\ & \vdots & \\ - & \left(\boldsymbol{x}^{(n)}\right)^T & - \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}$$

Then

$$\underset{\mathbb{R}^{n\times 1}}{h_{\boldsymbol{\theta}}(\boldsymbol{X})} = \underset{\mathbb{R}^{n\times(d+1)}}{\boldsymbol{X}}\,\underset{\mathbb{R}^{(d+1)\times 1}}{\boldsymbol{\theta}}$$

predictions for all instances

Matrix-Vector Form of Cost Function

Let

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Solve for θ analytically

$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ attained when $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = 0$

Expanding $\mathrm{J}(\boldsymbol{\theta})$:

Solve for θ analytically

Take gradient, set equal to 0, then solve for θ:

Closed-Form Solution

Can obtain $\boldsymbol{\theta}$ by plugging in $\boldsymbol{X}$ and $\boldsymbol{y}$ into

$$\boldsymbol{\theta}^* = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

- Cost of inverting $\boldsymbol{A} = \boldsymbol{X}^T\boldsymbol{X} \sim O(d^3)$
  – expensive for large d
- If $\boldsymbol{A}$ is not invertible (i.e. $\boldsymbol{A}$ is singular), may need to
  – use pseudo-inverse (`np.linalg.pinv`)
    - Cholesky decomposition better
  – remove redundant (not linearly independent) features
  – remove extra features to ensure $d \leq n$

Based on slide by Eric Eaton

Solving Linear Regression

Learning Goals

✓ Describe shape of $J(\boldsymbol{\theta})$

✓ Describe two approaches for optimizing $\boldsymbol{\theta}$

  ✓ Gradient Descent (stochastic and batch version)

  ✓ Normal Equations

- Compare tradeoffs of GD vs Normal Equations

Gradient Descent vs Normal Equation

Gradient Descent

- requires multiple iterations
- need to choose α
- works well when d is large
- can support online learning

Normal Equations

- non-iterative
- no need for α
- slow if *d* is large
  - matrix inversion is $O(d^3)$

Based on slide by Eric Eaton

What Should You Be Able To Do?

- Name cost function for linear regression and provide rationale
- Given model, find Maximum Likelihood Estimator (MLE)
- Optimize functions by taking derivative / gradient
- State tradeoffs of gradient descent vs normal equations
- Homework
  - Extend linear regression to weighted linear regression
  - Implement (regularized polynomial) regression using SGD and normal equations

(Extra Slides)

Linear Regression Extensions

Learning Goals

- Describe how to extend linear regression to more complex models using basis functions
- Describe why we might scale features

Basis Functions

So far,

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j$$

transform features by ϕ

Generally,

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d'} \theta_j \underbrace{\phi_j(\boldsymbol{x})}_{\text{basis function}}$$

new dimension

Typically, $\phi_0(\boldsymbol{x}) = 1$ so that θ_0 acts as bias.

In the simplest case, we use linear basis functions:

$$\phi_j(\boldsymbol{x}) = x_j$$

More complex basis functions allow use of linear regression techniques to **fit non-linear datasets**.

Based on slide by Eric Eaton
(originally by Chrisopher Bishop [PRML])

Extending Linear Regression to More Complex Models

The inputs $\boldsymbol{X}$ for linear regression can be …

- original quantitative inputs
- transformation of quantitative inputs
  - e.g. $\log$, $\exp$, square root, square, etc
- polynomial transformation
  - e.g. $y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$
- dummy coding of categorical inputs
- interactions between variables
  - e.g. $x_3 = x_1 x_2$

Based on slide by Eric Eaton

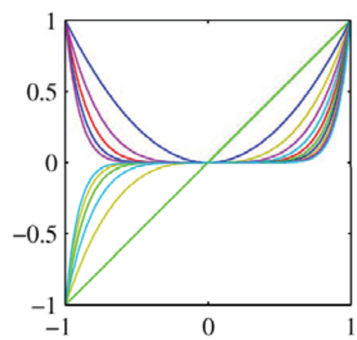
Polynomial Regression

Let $x \in \mathbb{R}$.

Polynomial basis functions:

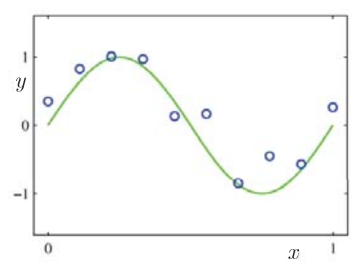
$$\phi_j(x) = x^j$$

These are "global" (a small change in x affects all basis functions).



Fit a polynomial curve with a linear model

$$h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_p x^p = \sum_{j=0}^{p} \theta_j x^p$$



Based on slide by Eric Eaton
(originally by Chrisopher Bishop [PRML])

Summary

Basic Linear Model

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j$$

Generalized Linear Model

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d'} \theta_j \phi_j(\boldsymbol{x})$$

Once we have replaced the data by the outputs of the basis functions, fitting the generalized linear model is same problem as fitting the basic model, e.g.

$$\boldsymbol{\Phi} = \begin{bmatrix} - & \left(\boldsymbol{\phi}(\boldsymbol{x}^{(1)})\right)^T & - \\ & \vdots & \\ - & \left(\boldsymbol{\phi}(\boldsymbol{x}^{(i)})\right)^T & - \\ & \vdots & \\ - & \left(\boldsymbol{\phi}(\boldsymbol{x}^{(n)})\right)^T & - \end{bmatrix}$$

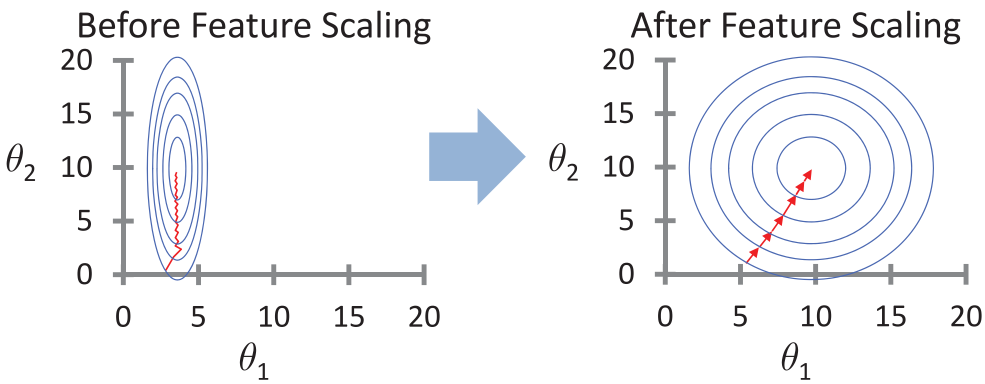
$$\boldsymbol{\theta}^* = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{y}$$

Based on slide by Eric Eaton
(originally by Geoff Hinton)

(This slide intentionally left blank.)

Feature Scaling

Ensure features have similar scales



Why?

- Make GD converge much faster
- Avoid numerical issues in matrix-vector products

Based on slide by Eric Eaton

Feature Standardization

Rescale features to have zero-mean, unit-variance

Let
$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)} \qquad \sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_j^{(i)} - \mu_j\right)^2}$$

Replace each value with
$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$
for $j = 1, \ldots, d$
do not rescale x_0

Remember to apply same transformation to both training and prediction

- learn parameters (μ, σ) using only training set
- see `sklearn.preprocessing`
- outliers can cause problems

Based on slide by Eric Eaton