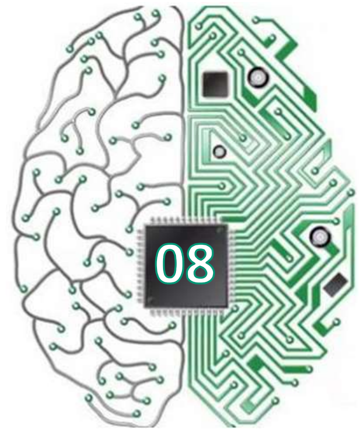
Open Elective Course [OE]
Course Code: CSO507
Winter 2023-24

Lecture#



Deep Learning

Unit-2: Linear and Logistic Regression (Part-I)

Course Instructor:

Dr. Monidipa Das
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkhand 826004, India

Supervised Learning

2

- Given a set of data points $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots., \boldsymbol{x}^{(n)}\}$ associated to a set of outcomes $\{y^{(1)}, y^{(2)}, \ldots., y^{(n)}\}$, we want to build a model that learns how to predict y from $\boldsymbol{x}$.

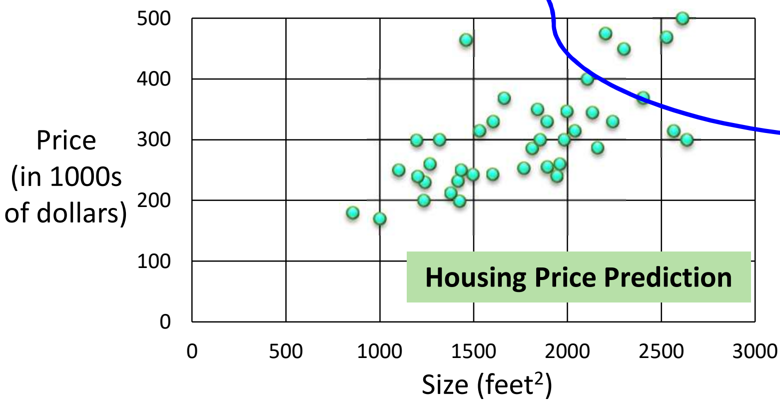
Type of prediction — The different types of predictive models are summed up in the table below:

	Regression	**Classification**
Outcome	Continuous	Class
Examples	Linear regression	Logistic regression, SVM, Naive Bayes

Regression: Task Description

Given:

- Data $\boldsymbol{X} = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\right\}$ where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$
- Corresponding labels $\boldsymbol{y} = \left\{y^{(1)}, \ldots, y^{(n)}\right\}$ where $y^{(i)} \in \mathbb{R}$

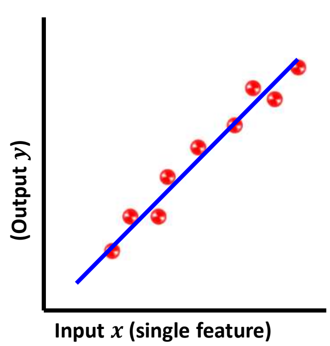


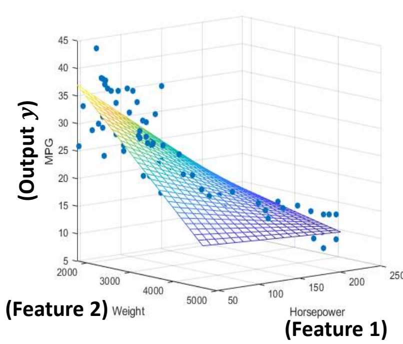
Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

Linear Regression Model

- **Linear regression is like fitting a line or (hyper)plane to a set of points**

- **Univariate linear regression:** A single independent variable is used to predict
- **Multivariate linear regression:** Two or more independent variables are used to predict



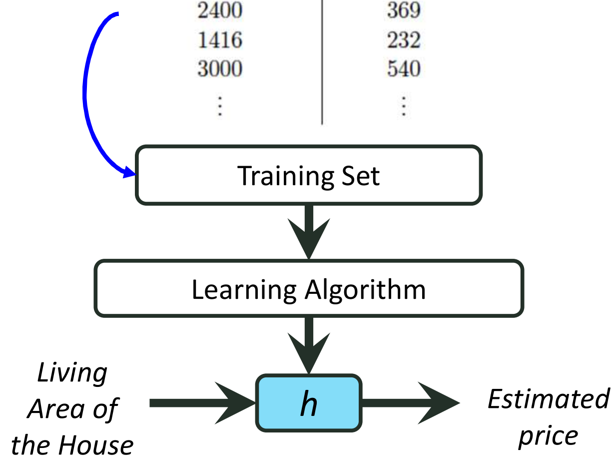


However, we can even fit a curve using a linear model after suitably transforming the inputs
$y \approx w^\top \phi(x)$

The transformation $\phi(.)$ can be predefined or learned (e.g., using kernel methods or a deep neural network based feature extractor).

Model Representation: Univariate Linear Regression

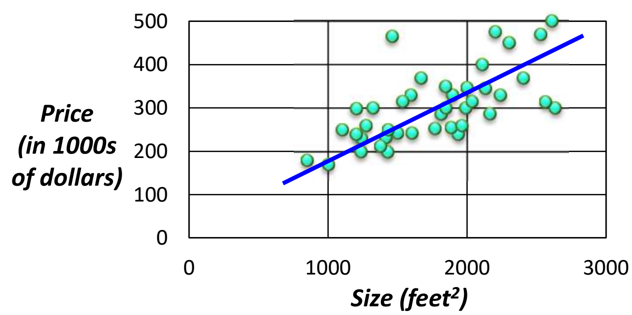
Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



How do we represent *h* ?

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

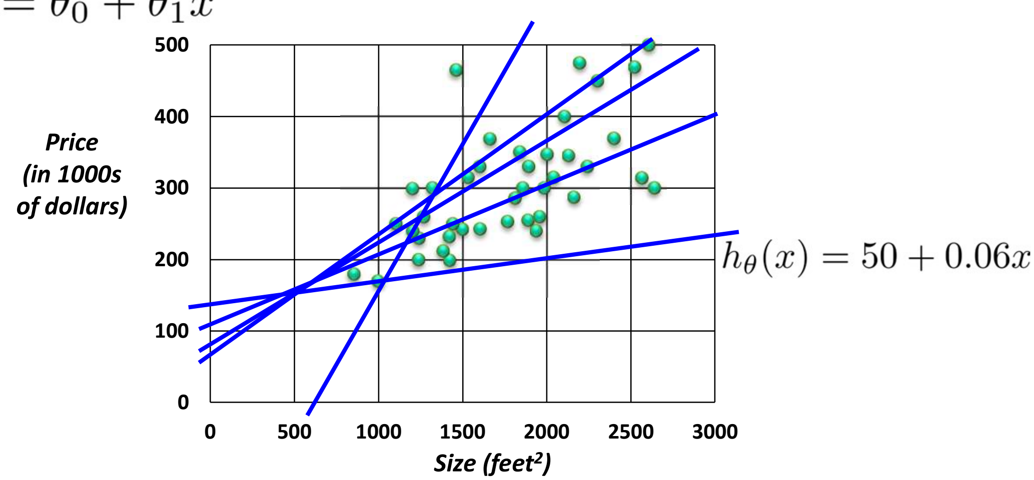
θ_i's: **Parameters**



Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

How to choose θ_i's ?

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



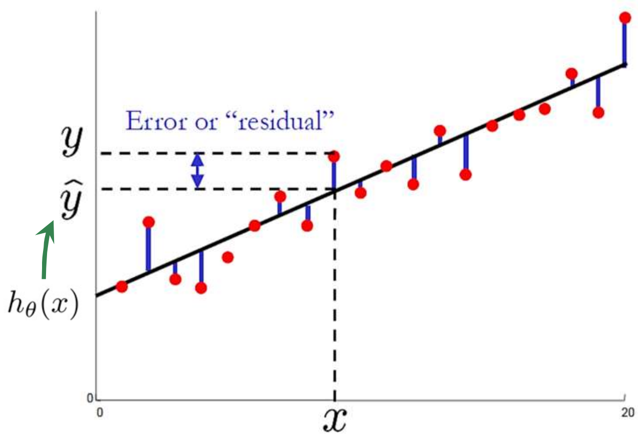
Idea: **Choose** θ_0, θ_1 **so that** $h_\theta(x)$ **is close to** y **for our training examples** (x, y)

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Cost Function

7

How good is the prediction given by the straight line?



Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function:

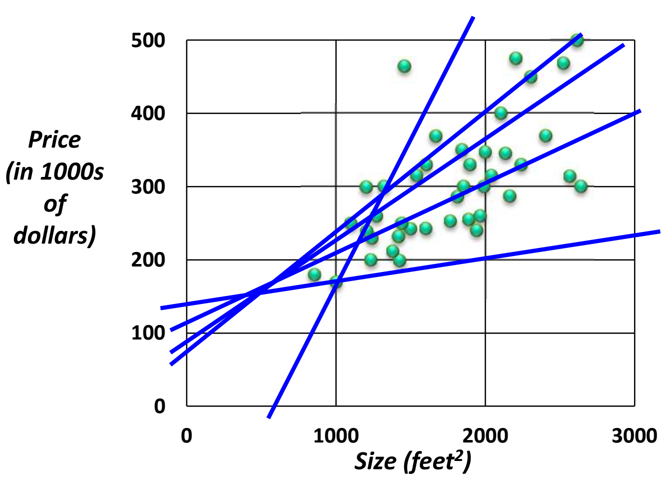
$\widehat{y}^{(i)}$

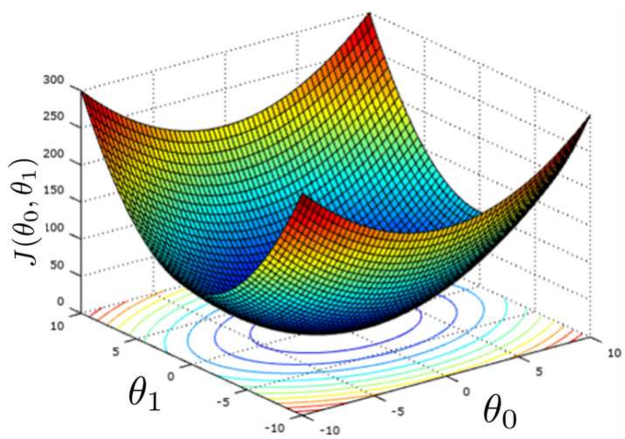
$$J(\theta_0, \theta_1) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Cost Function: Contour Plot

8





Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Optimization using Gradient Descent

9

Gradient descent algorithm

$$\text{repeat until convergence } \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$
$$(\text{for } j = 1 \text{ and } j = 0)$$
$$\}$$

Univariate Linear Regression Model

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

$$J(\theta_0, \theta_1) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

GD for Univariate Linear Regression Model

10

Gradient descent algorithm

$$\text{repeat until convergence } \{$$
$$\left.\begin{aligned} \theta_0 &:= \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \\ \theta_1 &:= \theta_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)} \end{aligned}\right\}$$
$$\}$$

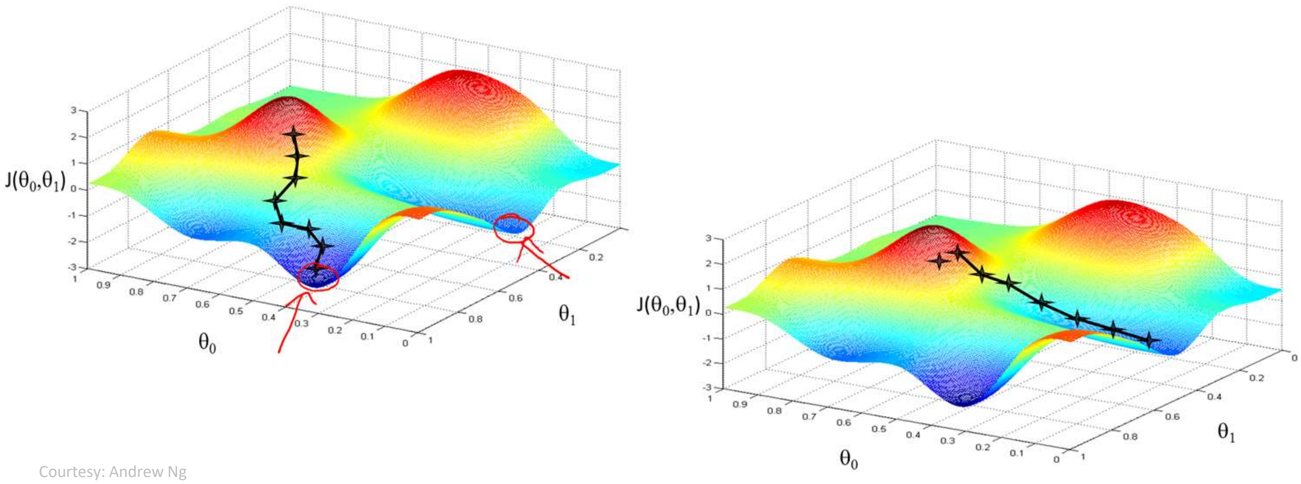
update θ_0 and θ_1 simultaneously

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Non-Convex Cost Function

11

- **Initialization of θ matters**



Courtesy: Andrew Ng

Multiple Features (variables)

12

$x_3^{(2)}$

Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

$y^{(2)}$

Notations:

n : Number of training samples
d : Number of features. E.g. $d = 4$ in the above example
$x^{(i)}$: training example i.
$y^{(i)}$: Label/target for training example i.
$x_j^{(i)}$: value of feature j in training example i.

Multivariate Linear Regression: **Hypothesis**

13

- Hypothesis for univariate linear regression:

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

- Hypothesis for **multivariate linear regression**:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$$

For convenience of notation, define $x_0 = 1$ for all sample i $\quad h_\theta(x^{(i)}) = \sum_{k=0}^{d} \theta_k x_k^{(i)}$

Multivariate Linear Regression: **Cost Function**

14

Hypothesis: $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$

Parameters: $\theta_0, \theta_1, \ldots, \theta_d$

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_d) = \frac{1}{2n} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Multivariate Linear Regression: **Gradient Descent**

15

Gradient descent:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_d)$$

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

} (simultaneously update for every $j = 0, \ldots, d$)

Repeat {

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

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

}

For Linear Regression

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

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)}$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

$\ldots$

Linear Regression: More Complex Models

16

- The inputs X for linear regression can be:
  - Original quantitative inputs: x
  - Transformation of quantitative inputs
    - **example:** $\log(x)$, $\exp(x)$, $\sqrt{x}$, x^2 etc.
  - Polynomial transformation
    - **example:** $y = \beta 0 + \beta 1 \cdot x + \beta 2 \cdot x^2 + \beta 3 \cdot x^3$
  - Basis expansions
  - Dummy coding of categorical inputs:
    - **example:** R: [1 0 0], G: [0 1 0], B: [0 0 1]
  - Interactions between variables
    - **example:** x3 = x1 · x2

- **These allows use of linear regression techniques to fit non-linear datasets.**

Linear Basis Function Models

17

- Generally,

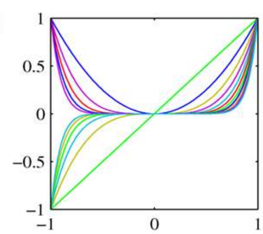
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j \underbrace{\phi_j(\boldsymbol{x})}_{\text{basis function}}$$

- Typically, $\phi_0(\boldsymbol{x}) = 1$ so that θ_0 acts as a bias
- In the simplest case, we use linear basis functions :

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

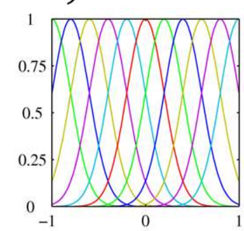
- Polynomial basis functions:

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



- Gaussian basis functions:

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

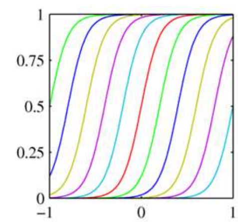


- Sigmoidal basis functions:

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$$

where

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$



Linear Basis Function Models

18

- 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 model is exactly the same problem as fitting the basic model
  - Unless we use the kernel trick
  - Therefore, there is no point in cluttering the math with basis functions

19

Model Representation through Vectorization

Vectorization

20

- Benefits of vectorization
  - More compact equations
  - Faster code (using optimized matrix libraries)
- Consider our model:

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

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$

- Can write the model in vectorized form as $h(\boldsymbol{x}) = \boldsymbol{\theta}^\intercal \boldsymbol{x}$

Vectorization

21

- Consider our model for n instances:

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

- Let

$$\underset{\mathbb{R}^{(d+1)\times 1}}{\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}} \qquad \underset{\mathbb{R}^{n\times(d+1)}}{\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}}$$

- Can write the model in vectorized form as $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\theta}$

Vectorization

22

- For the linear regression cost function:

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

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

$$= \frac{1}{2n}\underbrace{(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})^{\mathsf{T}}}_{\mathbb{R}^{1\times n}}\underbrace{(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})}_{\mathbb{R}^{n\times 1}}$$

$\mathbb{R}^{n\times(d+1)}$

$\mathbb{R}^{(d+1)\times 1}$

Let:

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

23

GD vs. Closed Form Solution for Linear Regression

Closed Form Solution

24

- Instead of using GD, solve for optimal $\boldsymbol{\theta}$ analytically
  - Notice that the solution is when $\frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 0$
- Derivation:

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2n} (\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})^\intercal (\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})$$

1 x 1

$$\propto \boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - \boxed{\boldsymbol{y}^\intercal \boldsymbol{X}\boldsymbol{\theta}} - \boxed{\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y}} + \boldsymbol{y}^\intercal \boldsymbol{y}$$

$$\propto \boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y} + \boldsymbol{y}^\intercal \boldsymbol{y}$$

Take derivative and set equal to 0, then solve for $\boldsymbol{\theta}$:

$$\frac{\partial}{\partial \boldsymbol{\theta}} (\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y} + \cancel{\boldsymbol{y}^\intercal \boldsymbol{y}}) = 0$$

$$(\boldsymbol{X}^\intercal \boldsymbol{X})\boldsymbol{\theta} - \boldsymbol{X}^\intercal \boldsymbol{y} = 0$$

$$(\boldsymbol{X}^\intercal \boldsymbol{X})\boldsymbol{\theta} = \boldsymbol{X}^\intercal \boldsymbol{y}$$

Closed Form Solution vs. GD

25

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

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

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Gradient Descent	**Closed Form Solution**
• Requires multiple iterations • Need to choose α • Works well when n is large • Can support incremental learning	• Non-iterative • No need for α • Slow if n is large – Computing $(\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}$ is roughly $O(n^3)$

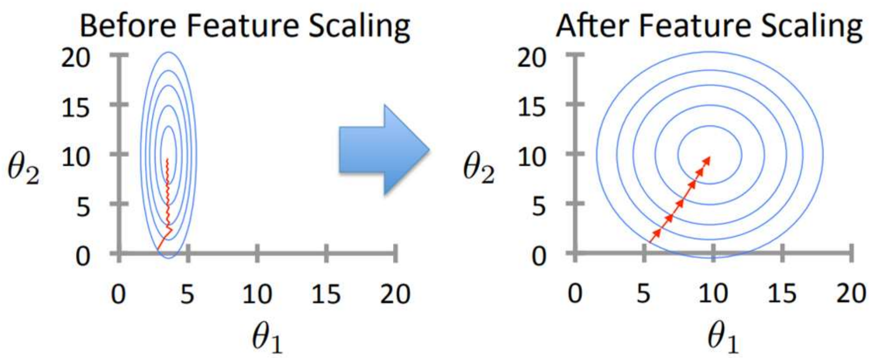
26

Improving Learning

Feature Scaling

27

- **Idea:** Ensure that feature have similar scales



- Makes gradient descent converge much faster

Feature Standardization

28

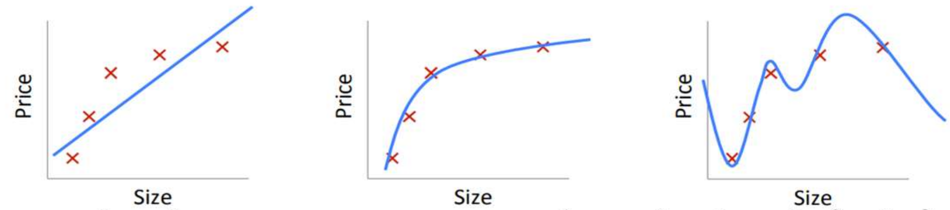
- Rescales features to have zero mean and unit variance
  - Let μ_j be the mean of feature j: $\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)}$
  - Replace each value with:

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$ for j = 1...d (not x_0!)

- s_j is the standard deviation of feature j
- Could also use the range of feature j ($\max_j - \min_j$) for s_j

- Must apply the same transformation to instances for both training and prediction
- Outliers can cause problems

Quality of Fit



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

Underfitting (high bias) | Correct fit | Overfitting (high variance)

Overfitting:

- The learned hypothesis may fit the training set very well ($J(\boldsymbol{\theta}) \approx 0$)
- ...but fails to generalize to new examples

Regularization

- A method for automatically controlling the complexity of the learned hypothesis
- **Idea**: penalize for large values of θ_j
  - Can incorporate into the cost function
  - Works well when we have a lot of features, each that contributes a bit to predicting the label
- Can also address overfitting by eliminating features (either manually or via model selection)

Regularization

31

- Linear regression objective function

$$J(\boldsymbol{\theta}) = \underbrace{\frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2}_{\text{model fit to data}} + \underbrace{\lambda\sum_{j=1}^{d}\theta_j^2}_{\text{regularization}}$$

  – λ is the regularization parameter ($\lambda \geq 0$)

  – No regularization on θ_0!

Understanding Regularization

32

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

- Note that $\sum_{j=1}^{d}\theta_j^2 = \|\boldsymbol{\theta}_{1:d}\|_2^2$
  – This is the magnitude of the feature coefficient vector!

- We can also think of this as:

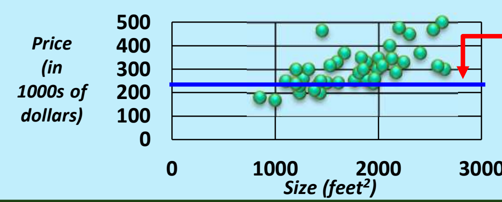
$$\sum_{j=1}^{d}(\theta_j - 0)^2 = \|\boldsymbol{\theta}_{1:d} - \vec{\mathbf{0}}\|_2^2$$

  - L_2 regularization pulls coefficients toward 0

Let $d = 4$

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

What happens if we set λ to be huge (e.g., 10^{10})? **$\theta_1, \theta_2, \theta_3, \theta_4$ becomes 0**



Regularized Linear Regression

33

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\left[\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2\right]$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Gradient update:

$$\frac{\partial}{\partial\theta_0}J(\theta) \qquad \theta_0 \leftarrow \theta_0 - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)$$

$$\frac{\partial}{\partial\theta_j}J(\theta) \qquad \theta_j \leftarrow \theta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)} \underbrace{- \alpha\frac{\lambda}{n}\theta_j}_{\text{regularization}}$$

- We can rewrite the gradient step as:

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

Various Ways of Regularization

34

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

$\lambda||\theta||_2^2$

ℓ_2 regularization

Ridge Regression

- $$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{d}|\theta_j|$$

$\lambda||\theta||_1$

ℓ_1 regularization

Lasso Regression

LASSO	Ridge
• Shrinks coefficients to 0 • Good for variable selection	Makes coefficients smaller

Least Absolute Shrinkage and Selection Operator (LASSO)

Non-regularization Approaches for Overfitting

35

- ***Early-stopping*** (stopping training just when we have a decent validation set accuracy)
- ***Injecting noise*** in the inputs
- ***Dropout*** (in each iteration, don't update some of the weights) [e.g. used in deep network-based models]

36

Questions?