Machine Learning

Lecture 2: Linear Regression

Feng Li

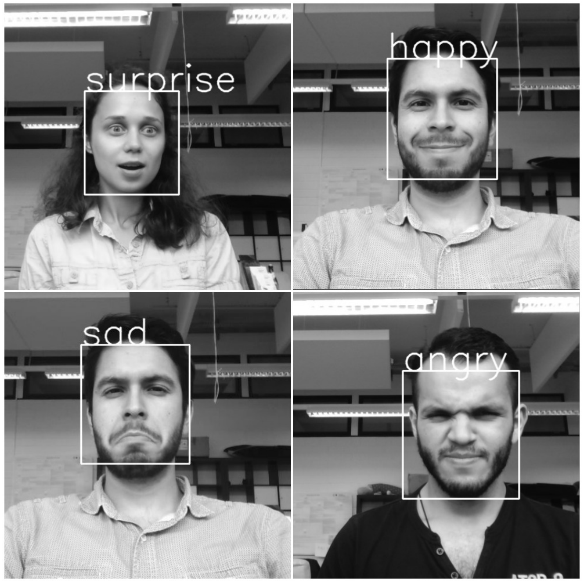
fli@sdu.edu.cn
https://funglee.github.io

School of Computer Science and Technology
Shandong University

Supervised Learning

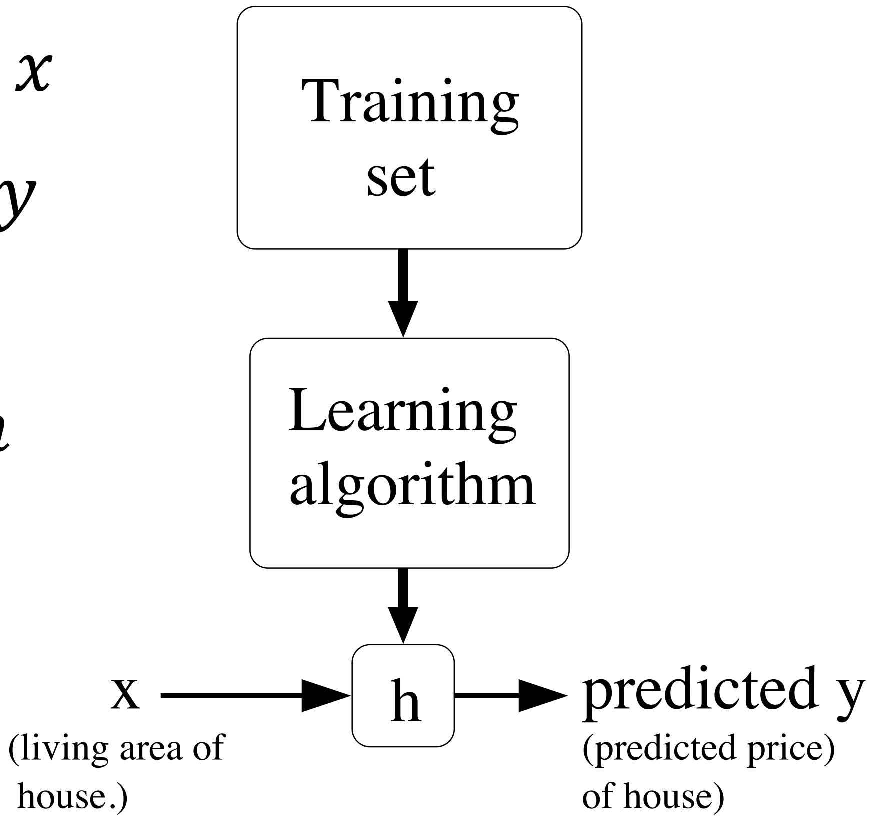
- Regression: Predict a continuous value
- Classification: Predict a discrete value, the class

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



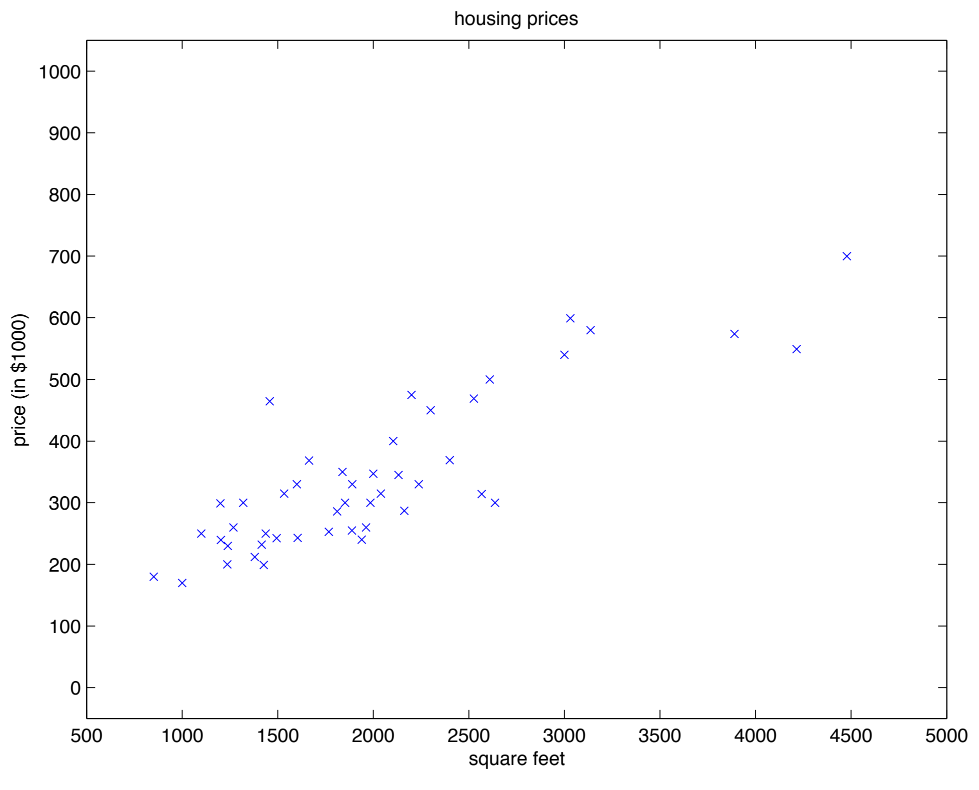
Supervised Learning (Contd.)

- Features: Input variables, x
- Target: Output variables, y
- Training examples: $\left(x^{(i)}, y^{(i)}\right), i = 1, 2, \cdots, m$
- Hypothesis: $h: \mathcal{X} \rightarrow \mathcal{Y}$



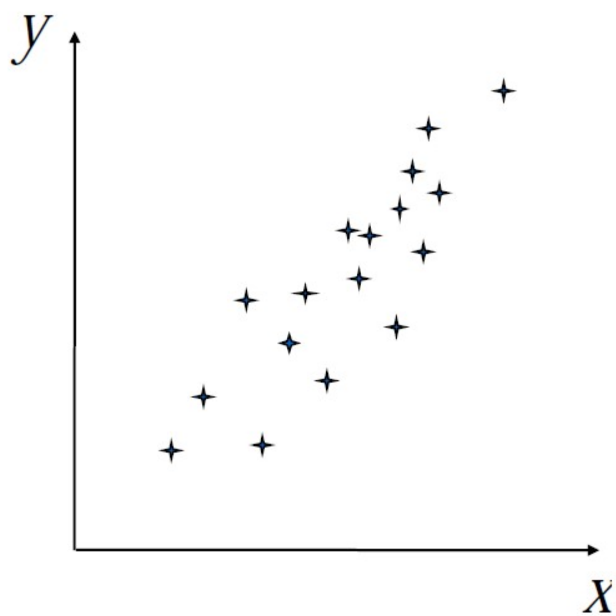
Linear Regression

- Linear hypothesis: $h(x) = \theta_1 x + \theta_0$
- θ_i ($i = 1, 2$ for 2D cases): Parameters to estimate
- How to choose θ_i ’s ?



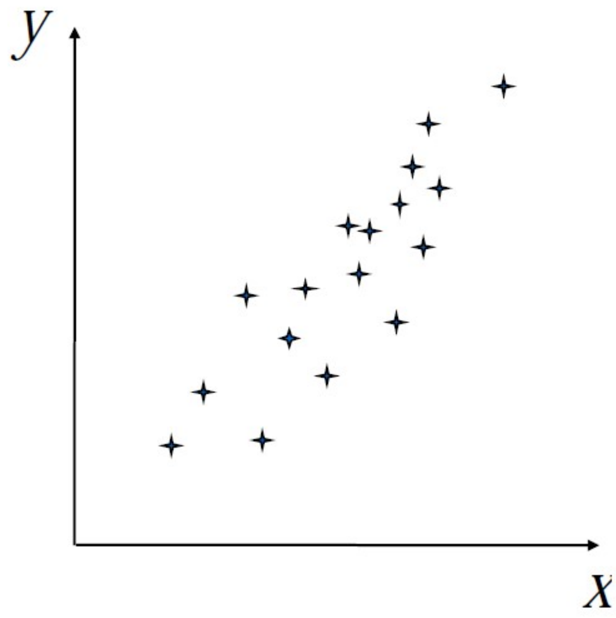
Linear Regression (Contd.)

- Input: Training set $\left(x^{(i)}, y^{(i)}\right) \in \mathbb{R}^2$ with $i = 1,2,\cdots m$
- Goal: Model the relationship between x and y such that we can predict the target $y = h(x)$ according to a given new feature input x



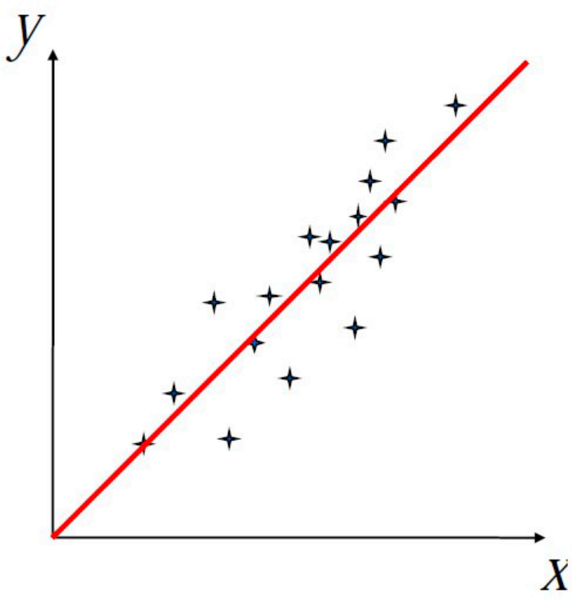
Linear Regression (Contd.)

- Input: Training set $\left(x^{(i)}, y^{(i)}\right) \in \mathbb{R}^2$ with $i = 1,2,\cdots m$
- Goal: Model the relationship between x and y such that we can predict the target $y = h(x)$ according to a given new feature input x



Linear Regression (Contd.)

- The relationship between x and y is modeled as a linear function (with respect to θ).
- The linear function in the 2D plane is a straight line
- Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$



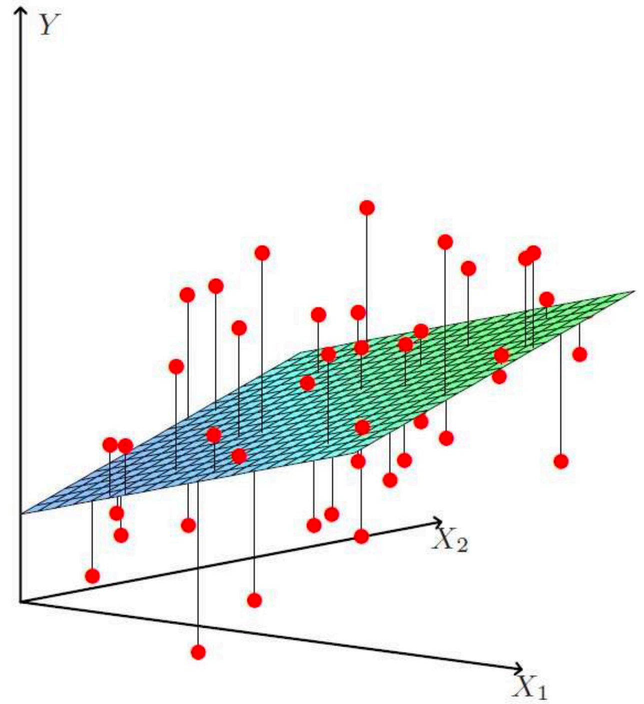
Linear Regression (Contd.)

- Given data $x \in \mathbb{R}^n$, we then have $\theta \in \mathbb{R}^{n+1}$
- Hence, $h_\theta(x) = \sum_{i=0}^{n} \theta_i \, x_i = \theta^T x$, where $x_0 = 1$
- What is the best choice of θ

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

where $J(\theta)$ is the so-called *cost function*

Linear Regression (Contd.)



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

Gradient Descent (GD) Algorithm

- If the multi-variable function $J(\theta)$ is differentiable in a neighborhood of a point θ, then $J(\theta)$ decreases fastest if one goes from θ in the direction of the negative gradient of J at θ
- Find a local minimum of a differentiable function using gradient descent

---

Algorithm 1 Gradient Descent

1: **Given** a starting point $\theta \in \mathbf{dom}\, J$

2: **Repeat**

3: Calculate gradient $\nabla J(\theta)$

4: Update $\theta \leftarrow \theta - \alpha \nabla J(\theta)$

5: **until** convergence criterion is satisfied

Remarks: θ is usually initialized randomly, and α is so-called *learning rate*

GD Algorithm (Contd.)

- Stopping criterion
  - The gradient has its magnitude less than or equal to a predefined threshold (say ε), i.e.,

    $$\|\nabla f(x)\|_2 \leq \varepsilon$$

    where $\|\cdot\|_2$ is ℓ_2 norm, such that the values of the objective function differ very slightly across different iterations
  - Set a fixed value for the maximum number of iterations, such that the algorithm is terminated after the number of the iterations exceeds the threshold

GD Algorithm (Contd.)

- Specifically, we update each component of θ according to the following rule

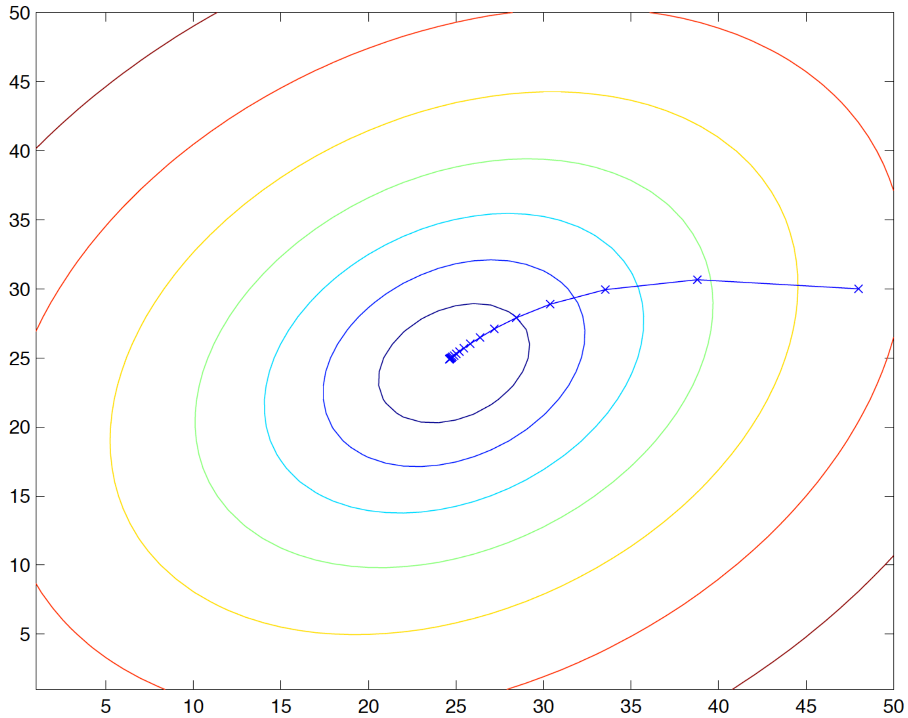
$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}, \qquad \forall j$$

- Calculating the gradient for linear regression

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \sum_{i=1}^{m} \left(\theta^{\mathrm{T}} x^{(i)} - y^{(i)}\right)^2 \\ &= \sum_{i=1}^{m} \left(\theta^{\mathrm{T}} x^{(i)} - y^{(i)}\right) \ x_j^{(i)}\end{aligned}$$

GD Algorithm (Contd.)

- An illustration of gradient descent algorithm



The objective function is decreased along the gradient

GD Algorithm (Contd.)

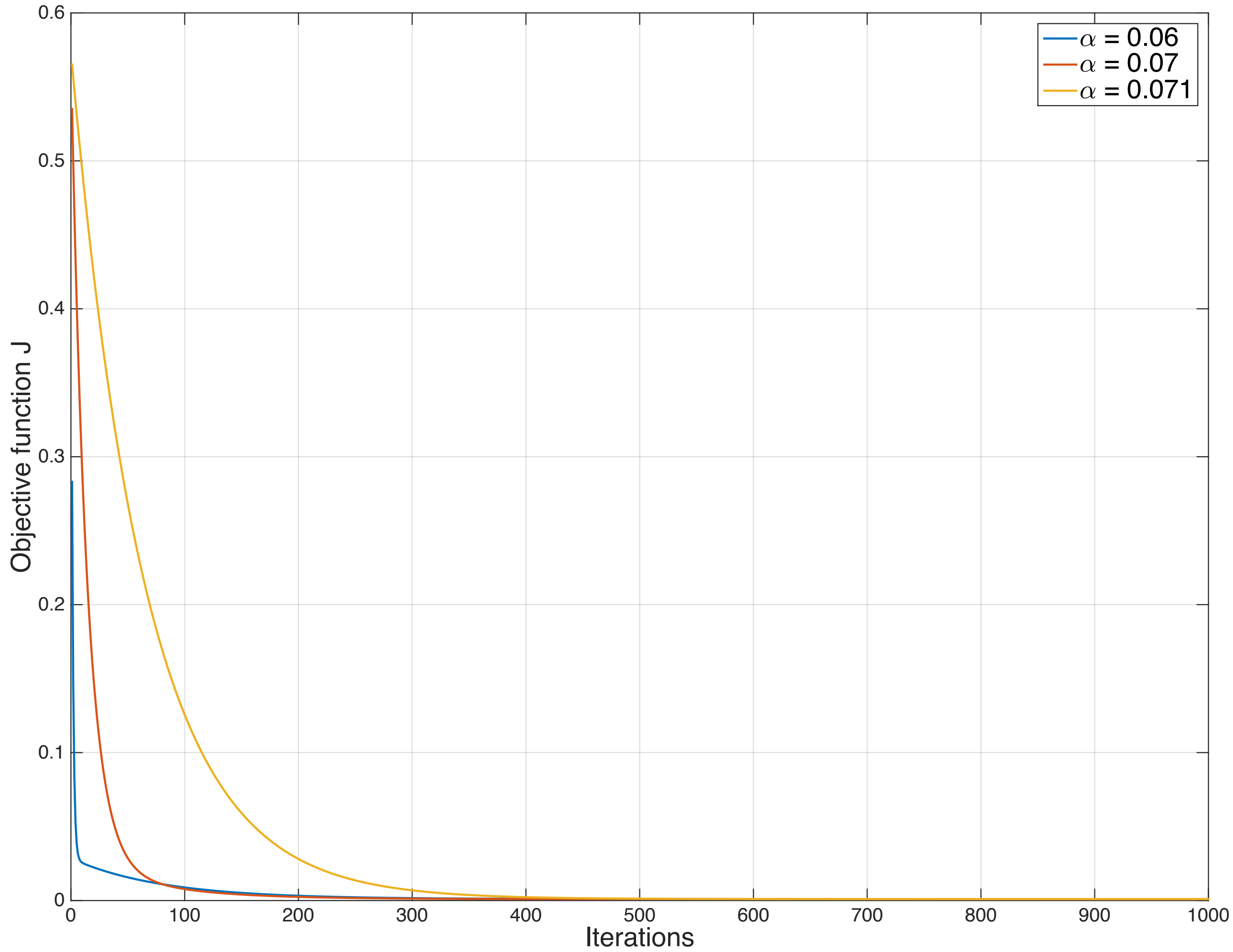
- Another commonly used form

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

- Gradient ascent algorithm
  - Maximize the differentiable function $J(\theta)$
  - The gradient represents the direction along which J increase fastest
  - Therefore, we have

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

Convergence under Different Step Sizes



GD Algorithm (Contd.)

- What if the training set is huge?
  - In the above batch gradient descent algorithm, we have to run through the entire training set in each iteration
  - A considerable computation cost is induced!
- Stochastic gradient descent (SGD), also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization method
  - In each iteration, the parameters are updated according to the gradient of the error with respect to one training sample only

Algorithm 2 Stochastic Gradient Descent for Linear Regression

```
1: Given a starting point θ ∈ dom J
2: repeat
3:     Randomly shuffle the training data;
4:     for i = 1, 2, ··· , m do
5:         θ ← θ − α∇J(θ; x^(i), y^(i))
6:     end for
7: until convergence criterion is satisfied
```

Stochastic Gradient Descent (SGD)

- Stochastic gradient descent (SGD), also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization method
  - In each iteration, the parameters are updated according to the gradient of the error with respect to one training sample only
  - For linear regression,

$$\nabla J\left(\theta; x^{(i)}, y^{(i)}\right) = \left(\theta^{\mathrm{T}} x^{(i)} - y^{(i)}\right) x_j^{(i)}$$

Algorithm 2 Stochastic Gradient Descent for Linear Regression

---

Given a starting point $\theta \in \mathbf{dom}\ J$
repeat
    Randomly shuffle the training data;
    for $i = 1, 2, \cdots, m$ **do**
        $\theta \leftarrow \theta - \alpha \nabla J(\theta; x^{(i)}, y^{(i)})$
    end for
until convergence criterion is satisfied

SGD (Contd.)

- Stochastic gradient descent (SGD), also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization method
  - In each iteration, the parameters are updated according to the gradient of the error with respect to one training sample only
  - For linear regression,

$$\nabla J\left(\theta; x^{(i)}, y^{(i)}\right) = \left(\theta^{\mathrm{T}} x^{(i)} - y^{(i)}\right) x_j^{(i)}$$

Algorithm 2 Stochastic Gradient Descent for Linear Regression

1: **Given** a starting point $\theta \in \mathbf{dom}\ J$
2: **repeat**
3:     Randomly shuffle the training data;
4:     **for** $i = 1, 2, \cdots, m$ **do**
5:         $\theta \leftarrow \theta - \alpha \nabla J(\theta; x^{(i)}, y^{(i)})$
6:     **end for**
7: **until** convergence criterion is satisfied

More about SGD

- The objective does not always decrease for each iteration
- Usually, SGD has approaching the minimum much faster than batch GD
- SGD may never converge to the minimum, and oscillating may happen
- A variants: Mini-batch, say pick up a small group of samples and do average, which may accelerate and smoothen the convergence

Matrix Derivatives[1]

- A function $f\colon \mathbb{R}^{m\times n} \to \mathbb{R}$
- The derivative of f with respect to A is defined as

$$\nabla f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

- For an $n\times n$ matrix, its trace is defined as $\mathrm{tr}A = \sum_{i=1}^{n} A_{ii}$

$$\mathrm{tr}ABCD = \mathrm{tr}DABC = \mathrm{tr}CDAB = \mathrm{tr}BCDA$$
$$\mathrm{tr}A = \mathrm{tr}A^T,\ \mathrm{tr}(A+B) = \mathrm{tr}A + \mathrm{tr}B,\ \mathrm{tr}aA = a\mathrm{tr}A$$
$$\nabla_A \mathrm{tr}AB = B^T,\ \nabla_{A^T} f(A) = (\nabla_A f(A))^T$$
$$\nabla_A \mathrm{tr}ABA^TC = CAB + C^TAB^T,\ \nabla_A|A| = |A|(A^{-1})^T$$
Funky trace derivative $\nabla_{A^T}\mathrm{tr}ABA^TC = B^TA^TC^T + BA^TC$

1 Details can be found in “Properties of the Trace and Matrix Derivatives” by John Duchi

Revisiting Least Square

- Assume

$$X = \begin{bmatrix} \left(x^{(1)}\right)^{\mathrm{T}} \\ \vdots \\ \left(x^{(m)}\right)^{\mathrm{T}} \end{bmatrix}, \qquad Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

- Therefore, we have

$$X\theta - Y = \begin{bmatrix} \left(x^{(1)}\right)^{\mathrm{T}}\theta \\ \vdots \\ \left(x^{(m)}\right)^{\mathrm{T}}\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} = \begin{bmatrix} h_\theta\left(x^{(1)}\right) - y^{(1)} \\ \vdots \\ h_\theta\left(x^{(m)}\right) - y^{(m)} \end{bmatrix}$$

- The objective function can be written as

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = \frac{1}{2}(X\theta - Y)^T(X\theta - Y)$$

Revisiting Least Square (Contd.)

- Minimize $J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) = \frac{1}{2}(X\theta - Y)^T(X\theta - Y)$
- Calculate its derivative with respect to θ

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(Y - X\theta)^{\mathrm{T}}(Y - X\theta) \\
&= \frac{1}{2}\nabla_\theta (Y^T - \theta^T X^T)(Y - X\theta) \\
&= \frac{1}{2}\nabla_\theta \mathrm{tr}(Y^T Y - Y^T X\theta - \theta^T X^T Y + \theta^T X^T X\theta) \\
&= \frac{1}{2}\nabla_\theta \mathrm{tr}(\theta^T X^T X\theta) - X^T Y \\
&= \frac{1}{2}(X^T X\theta + X^T X\theta) - X^T Y \\
&= X^T X\theta - X^T Y
\end{aligned}
$$

Revisiting Least Square (Contd.)

- **Theorem:**

The matrix $A^T A$ is invertible if and only if the columns of A are linearly independent. In this case, there exists only one least-squares solution

$$\theta = (X^T X)^{-1} X^T Y$$

- Prove the above theorem in Problem Set 1

Probabilistic Interpretation

- The target variables and the inputs are related

$$y = \theta^{\mathrm{T}} x + \epsilon$$

  - ϵ's denote the errors and are independently and identically distributed (i.i.d.) according to a Gaussian distribution $\mathcal{N}(0, \sigma^2)$

- The density of $\epsilon^{(i)}$ is given by

$$f(\epsilon) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\epsilon^2}{2\sigma^2}\right)$$

- The conditional probability density function of y

$$y \mid x; \theta \sim \mathcal{N}(\theta^{\mathrm{T}} x, \sigma^2)$$

Probabilistic Interpretation (Contd.)

- The training data $\{x^{(i)}, y^{(i)}\}_{i=1,\cdots,m}$ are sampled identically and independently

$$p\left(y^{(i)} \mid x^{(i)}; \theta \right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2} \right)$$

- Likelihood function

$$L(\theta) = \prod_i p(y^{(i)} \mid x^{(i)}; \theta)$$

$$= \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2} \right)$$

Probabilistic Interpretation (Contd.)

- Maximizing the likelihood $L(\theta)$
  - Choosing the optimal θ to make the data as high probability as possible
- Since $L(\theta)$ is complicated, we maximize an logarithmic function of $L(\theta)$ instead

$$
\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_{i}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_{i}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i} (y^{(i)} - \theta^T x^{(i)})^2
\end{aligned}
$$

Thanks !

Q & A