

# CIS526: Machine Learning

## Lecture 3 (Sept 16, 2003)

Preparation help: Xiaoying Huang

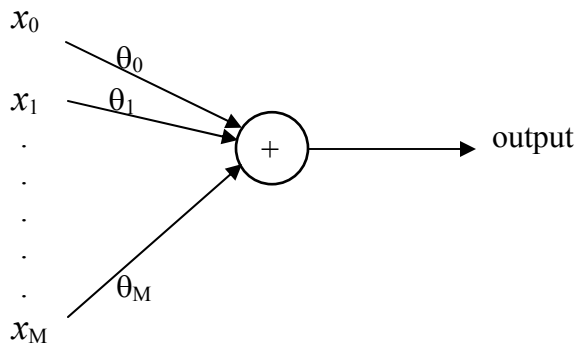
### Linear Regression

Linear regression can be represented by a functional form:

$$f(\mathbf{x}; \boldsymbol{\theta}) = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_M x_M = \sum_{j=0}^M \theta_j x_j$$

**Note:**  $x_0$  is a **dummy attribute** and its value is a constant equal to 1.

Linear regression can also be represented in a graphic form:



**Goal:** Minimize Mean Square Error (MSE):

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - f(\mathbf{x}_i; \boldsymbol{\theta}))^2$$

⇒ MSE is a quadratic function in parameters  $\boldsymbol{\theta}$

⇒ It is a convex function

⇒ There is only one minimum, it is the **global minimum**

**Solution:** Sufficient condition is  $\frac{\partial \text{MSE}}{\partial \theta_j} = 0, \forall \theta_j, j = 0, 1, \dots, M$ .

Therefore, find  $\theta_j$  such that

$$\frac{\partial \text{MSE}}{\partial \theta_j} = -\frac{2}{N} \sum_i \left( y_i - \sum_k \theta_k x_{ik} \right) \cdot x_{ij} = 0, \forall j$$

There are  $M+1$  linear equations with  $M+1$  unknown variables ⇒ we can get a closed-form solution.

**Special Case:** If some attribute is a linear combination of others, there is no unique solution.

$$\frac{\partial MSE}{\partial \theta_j} = 0 \Rightarrow \sum_{i=1}^N y_i x_{ij} = \sum_{i=1}^N \sum_{k=0}^M \theta_k x_{ik} x_{ij} \Rightarrow (\text{in matrix form}) X^T Y = X^T X \theta,$$

where:

$$X_{[N \times (M+1)]} = \{x_{ij}\}_{i=1:N, j=1:(M+1)}, \quad (x_{ij} \text{ is } j\text{th attribute of } i\text{th data point})$$

$$Y_{[N \times 1]} = \{y_i\}_{i=1:N},$$

$$\theta_{[(M+1) \times 1]} = \{\theta_j\}_{j=1:(M+1)}$$

**Note:**  $D = [X \ Y]$ , i.e.,  $[X \ Y]$  is what we defined previously as the data set.

The **optimal parameter choice** is then:

$$\theta = (X^T X)^{-1} X^T Y, \text{ which is a closed form solution.}$$

**Note:** the above solution exists if  $X^T X$  is invertible, i.e. if its rank equals  $M+1$ , i.e. no attribute is a linear combination of others (in Matlab, use function *rank*).

**Note:** using matrix derivations we can do the optimization in a more elegant way by defining

$$\begin{aligned} MSE &= \frac{1}{N} (Y - X\theta)^T (Y - X\theta) \Rightarrow \nabla_{\theta} MSE = -2X^T (Y - X\theta) = \mathbf{0}_{[(M+1) \times 1]} \\ &\Rightarrow \theta = (X^T X)^{-1} X^T Y \end{aligned}$$

## Statistical results:

**Assumption:** the true **data generating process** (DGP) is

$$y = \sum_{j=0}^M \beta_j x_j + e, \quad e \text{ is noise with } E(e) = 0, \text{ Var}(e) = \sigma^2$$

**Note:** This is a big assumption!

**Questions:** How close is the estimate  $\theta$  to the true value  $\beta$ ?

**Answer 1:**

$$E[\theta] = E[(X^T X)^{-1} X^T Y] = (X^T X)^{-1} X^T E[Y] \quad (\text{remember, } Y = X\beta + e_{[N \times 1]})$$

$$\Rightarrow E[\theta] = (X^T X)^{-1} X^T X \beta + (X^T X)^{-1} X^T E[e]$$

$$\Rightarrow E[\theta] = \beta + 0 = \beta$$

**Conclusion:** if we repeat linear regression on different data sets sampled according to the true DGP, the average  $\theta$  will equal  $\beta$  (i.e.,  $E[\theta] = \beta$ ), which are the true parameters. Therefore, the linear regression is an **unbiased predictor**.

**Answer 2:** The variance of parameter estimate  $\theta$  is

$$\text{Var}[\theta] = \dots (\text{after some calculation}) \dots = (X^T X)^{-1} \sigma^2$$

**Conclusion:**  $\text{Var}[\theta]$  is a measure of how different estimation  $\theta$  is from the true parameters  $\beta$ , i.e. how successful is the linear regression. Therefore, quality of linear

regression depends on the **noise level** (i.e.  $\sigma^2$ ) and on the **data size**. The variance increases linearly with  $\sigma^2$  and decreases as  $1/N^2$  with the size of the dataset  $N$ .

**More stringent assumption:** the true DGP is

$$y = \sum_{j=0}^M \beta_j x_j + e, \text{ and } e \sim N(0, \sigma^2) \text{ (i.e., } e \text{ is Gaussian additive noise)}$$

**If the assumption is valid we could:** Estimate  $\theta$  can be considered as a multi-dimensional Gaussian variable with  $\theta = N(\beta, (X^T X)^{-1} \sigma^2)$ . Therefore, we could do some nice thing such as test the hypothesis that  $\beta_j=0$  (i.e. that attribute  $j$  is not influencing the target  $y$ ).

## Nonlinear Regression

**Question:** What if we know that  $f(\mathbf{x};\theta)$  is a non-linear parametric function?

**For example:**  $f(\mathbf{x};\theta) = \theta_0 + \theta_1 x_1 x_2^2$ , this is a function nonlinear in parameters.

**Solution:** Minimize  $MSE = \frac{1}{N} \sum (y_i - f(\mathbf{x}_i; \theta))^2$

Start from the necessary condition for minimum:

$$\Rightarrow \frac{\partial MSE}{\partial \theta_j} = -\frac{2}{N} \sum (y_i - f(\mathbf{x}_i; \theta)) \frac{\partial f(\mathbf{x}_i; \theta)}{\partial \theta_j} = 0$$

$\Rightarrow$  Again, we have to solve  $M$  nonlinear equations with  $M$  unknowns.

$\Rightarrow$  But, this time closed-form solution is not easy to derive.

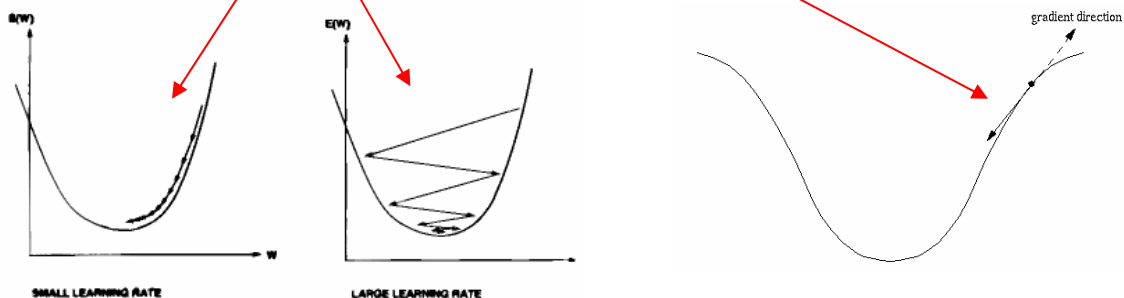
**Math Background: Unconstrained Optimization:**

**Problem:** Given  $f(\mathbf{x})$ , find its minimum.

**Popular Solution:** Use the **gradient descent** algorithm.

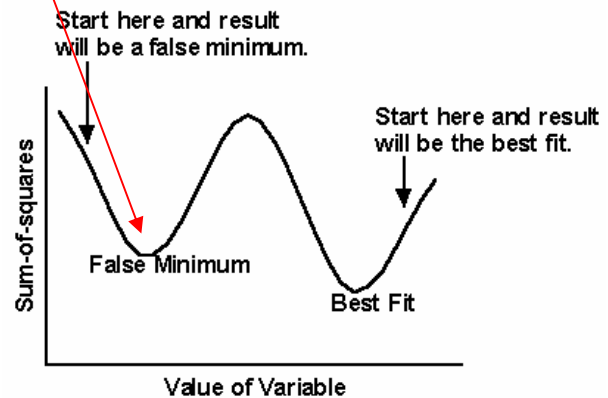
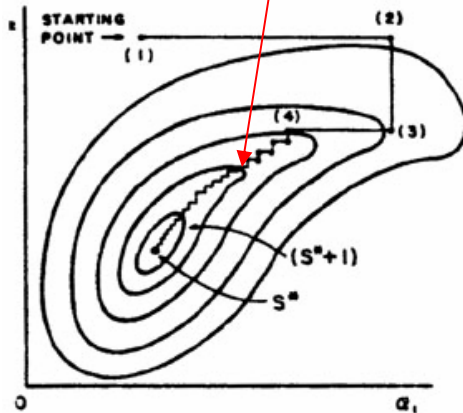
**Idea:** The gradient of  $f(\mathbf{x})$  at the minimum is zero vector. So,

1. start from an initial guess  $\mathbf{x}^0$ ;
2. calculate gradient  $\nabla f(\mathbf{x}^0)$ ;
3. move in the direction **opposite of the gradient**, i.e., generate new guess  $\mathbf{x}^1$  as  $\mathbf{x}^1 = \mathbf{x}^0 - \alpha \nabla f(\mathbf{x}^0)$ , where  $\alpha$  is a properly selected constant;
4. repeat this process until convergence to the minimum.



Two problems with gradient descent algorithm:

1. It accepts convergence to a **local minimum**. The simplest solution to avoid the local minimum is to repeat the procedure starting from multiple initial guesses  $\mathbf{x}^0$ .
2. Possible **slow convergence** to a minimum. There are a number of algorithms providing faster convergence (e.g. conjugate gradient; second order methods such as Newton or quazi-Newton; nonderivative methods)



**Back to solving nonlinear regression using gradient descent procedure:**

- Step 1: Start from an initial guess for parameters  $\theta^0$ .  
 Step k: Update the parameters as  $\theta^{k+1} = \theta^k - \alpha \nabla f(\theta^k)$

Special Case: For linear prediction the update step would be  $\theta^{k+1} = \theta^k + 2\alpha X^T(Y - X\theta^k)$

## Logistic Regression by MSE Minimization

**Remember:** Classification can be solved by MSE minimization methods ( $E[y|\mathbf{x}]$  can be used to derive posteriors  $P(y \in C_j | \mathbf{x})$ ).

**Question:** What functional form  $f(\mathbf{x}; \theta)$  can be an appropriate choice for representing posterior class probabilities?

**Option 1:** What about linear model  $f(\mathbf{x}; \theta) = \sum_{j=0}^M \theta_j x_j$ ? The range of the function goes beyond 0-1, so it is not a good choice.

**Option 2:** We can use sigmoid function to do squeeze the output of a linear model to the range between 0 and 1:  $f(\mathbf{x}; \theta) = g(\sum_{j=0}^M \theta_j x_j)$ . If  $g(z) = e^{-z}/(1+e^{-z})$ , optimizing  $f(\mathbf{x}; \theta)$  is called **logistic regression**.

**Solution:** Logistic regression can be solved by minimizing MSE. Derivative  $\partial \text{MSE} / \partial \theta_j$  is

$$\frac{\partial \text{MSE}}{\partial \theta_j} = -\frac{2}{N} \sum_{i=1}^N (y - f(\mathbf{x}; \boldsymbol{\theta})) x_{ij} g' \left( \sum_{k=0}^M \theta_k x_{ik} \right)$$

**Note:** Solving  $\nabla_{\boldsymbol{\theta}} \text{MSE} = 0$  results in  $(M+1)$  nonlinear equations with  $(M+1)$  unknowns  $\Rightarrow$  optimization can be done by using gradient descent algorithm.

## Maximum Likelihood (ML) Algorithm

**Basic Idea:** Given a data set  $D$  and a parametric model with parameters  $\boldsymbol{\theta}$  that describes the data generating process, the best solution  $\boldsymbol{\theta}^*$  is the one that maximizes  $P(D|\boldsymbol{\theta})$ , i.e.

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta}} P(D|\boldsymbol{\theta})$$

$P(D|\boldsymbol{\theta})$  is called the **likelihood**, so the name of the algorithm that finds the optimal solution  $\boldsymbol{\theta}^*$  is called the **maximum likelihood algorithm**. This idea can be applied for both unsupervised and supervised learning problems.

### ML for Unsupervised Learning: Density Estimation

Given  $D = \{\mathbf{x}_i, i=1, 2, \dots, N\}$ , and assuming the functional form  $p(\mathbf{x}|\boldsymbol{\theta})$  of the data generating process, the goal is to estimate the optimal parameters  $\boldsymbol{\theta}$  that maximize likelihood  $P(D|\boldsymbol{\theta})$ :

$$P(D|\boldsymbol{\theta}) = P(x_1, x_2, \dots, x_N|\boldsymbol{\theta})$$

By assuming that data points  $\mathbf{x}_i$  are **independent and identically distributed (iid)**

$$P(D|\boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{x}_i | \boldsymbol{\theta}) \quad (p \text{ is the probability density function.})$$

Since  $\log(x)$  is monotonically increasing function with  $x$ , maximization of  $P(D|\boldsymbol{\theta})$  is equivalent to maximization of  $l = \log(P(D|\boldsymbol{\theta}))$ .  $l$  is called the log-likelihood. So,

$$l = \sum_{i=1}^N \log(p(\mathbf{x}_i | \boldsymbol{\theta}))$$

**Example:** Data set  $D = \{\mathbf{x}_i, i=1, 2, \dots, N\}$  is drawn from a Gaussian distribution with mean  $\mu$  and standard deviation  $\sigma$ , i.e.,  $X \sim N(\mu, \sigma^2)$ . Therefore,

$$p(x_i | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \Rightarrow l = \sum_{i=1}^N \left( \log \frac{1}{\sqrt{2\pi\sigma}} - \frac{(x-\mu)^2}{2\sigma^2} \right)$$

Values  $\mu$  and  $\sigma$  that maximize the log-likelihood satisfy the necessary condition for local optimum:

$$\frac{\partial l}{\partial \mu} = 0 \Rightarrow \hat{\mu} = \frac{1}{N} \sum_{i=1}^N x_i, \quad \frac{\partial l}{\partial \sigma} = 0 \Rightarrow \hat{\sigma} = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{\mu})^2$$

## ML for Supervised Learning

Given  $D = \{(\mathbf{x}_i, y_i), i=1, 2, \dots, N\}$ , and assuming the functional form  $p(y|\mathbf{x}, \boldsymbol{\theta})$  of the data generating process, the goal is to estimate the optimal parameters  $\boldsymbol{\theta}$  that maximize likelihood  $P(D|\boldsymbol{\theta})$ :

$$\begin{aligned} P(D|\boldsymbol{\theta}) &= P(y_1, y_2, \dots, y_N | x_1, x_2, \dots, x_N, \boldsymbol{\theta}) = \quad \text{/if data is iid} \\ &= \prod_{i=1}^N p(y_i | \mathbf{x}_i, \boldsymbol{\theta}) \end{aligned}$$

## ML for Regression

Assume the data generating process corresponds to:

$$y = f(\mathbf{x}, \boldsymbol{\theta}) + e, \text{ where } e \sim N(\mu, \sigma^2)$$

Note: this is a relatively strong assumption!

$$\Rightarrow y \sim N(f(\mathbf{x}, \boldsymbol{\theta}), \sigma^2)$$

$$\Rightarrow p(y | \mathbf{x}, \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(y-f(\mathbf{x}, \boldsymbol{\theta}))^2}{2\sigma^2}}$$

$$\Rightarrow l = \log P(D | \boldsymbol{\theta}) = \sum_{i=1}^N \left( \log \frac{1}{\sqrt{2\pi\sigma}} - \frac{(y_i - f(\mathbf{x}_i, \boldsymbol{\theta}))^2}{2\sigma^2} \right)$$

Since  $\sigma$  is a constant, maximization of  $l$  is equivalent to minimization of  $\frac{1}{N} \sum_{i=1}^N (y_i - f(x_i, \boldsymbol{\theta}))^2$

**Important conclusion:** **Regression using ML** under the assumption of DGP with additive Gaussian noise is equivalent to **regression using MSE minimization!!**