

**CS 1675 Intro to Machine Learning**  
**Lecture 8**

**Linear regression**

Milos Hauskrecht  
[milos@cs.pitt.edu](mailto:milos@cs.pitt.edu)  
5329 Sennott Square

---

**Outline**

**Linear Regression**

- Linear model
  - Mean squared error function
  - Parameter estimation.
  - Gradient methods.
  - On-line regression techniques.
  - Linear additive models
  - Statistical model of linear regression
  - Regularized linear regression
-

## Supervised learning

**Data:**  $D = \{D_1, D_2, \dots, D_n\}$  a set of  $n$  examples

$$D_i = \langle \mathbf{x}_i, y_i \rangle$$

$\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,d})$  is an input vector of size  $d$

$y_i$  is the desired output (given by a teacher)

**Objective:** learn the mapping  $f : X \rightarrow Y$

$$\text{s.t. } y_i \approx f(\mathbf{x}_i) \text{ for all } i = 1, \dots, n$$

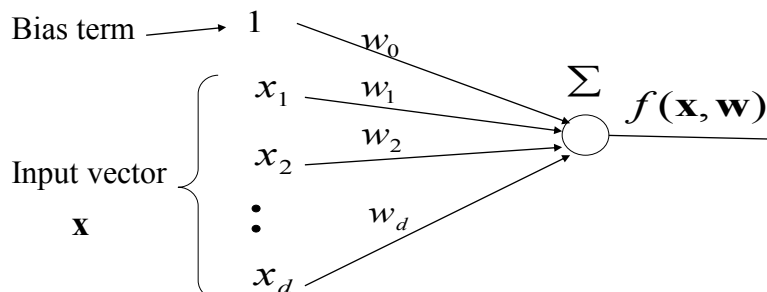
- **Regression:**  $Y$  is **continuous**  
Example: earnings, product orders  $\rightarrow$  company stock price
- **Classification:**  $Y$  is **discrete**  
Example: handwritten digit in binary form  $\rightarrow$  digit label

## Linear regression

- **Function**  $f : X \rightarrow Y$  is a linear combination of input components

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d = w_0 + \sum_{j=1}^d w_j x_j$$

$w_0, w_1, \dots, w_k$  - **parameters (weights)**



## Linear regression

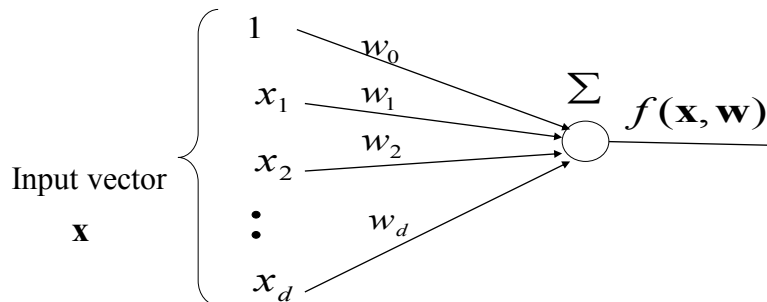
- **Shorter (vector) definition of the model**

- Include bias constant in the input vector

$$\mathbf{x} = (1, x_1, x_2, \dots, x_d)$$

$$f(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \dots w_d x_d = \mathbf{w}^T \mathbf{x}$$

$w_0, w_1, \dots, w_k$  - **parameters (weights)**



## Linear regression. Error.

- **Data:**  $D_i = \langle \mathbf{x}_i, y_i \rangle$
- **Function:**  $\mathbf{x}_i \rightarrow f(\mathbf{x}_i)$
- We would like to have  $y_i \approx f(\mathbf{x}_i)$  for all  $i = 1, \dots, n$

- **Error function**

- measures how much our predictions deviate from the desired answers

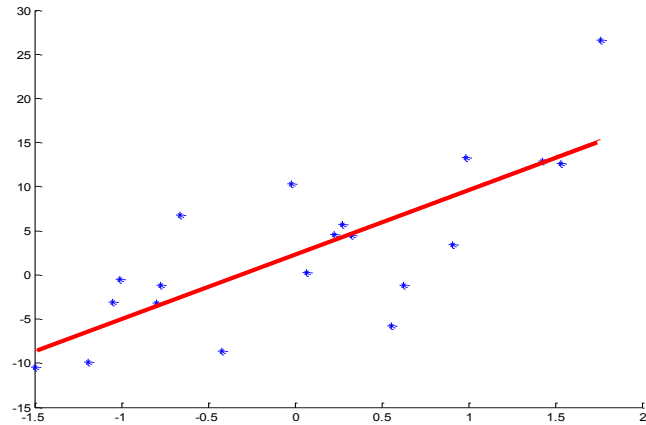
**Mean-squared error**  $J_n = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - f(\mathbf{x}_i))^2$

- **Learning:**

**We want to find the weights minimizing the error !**

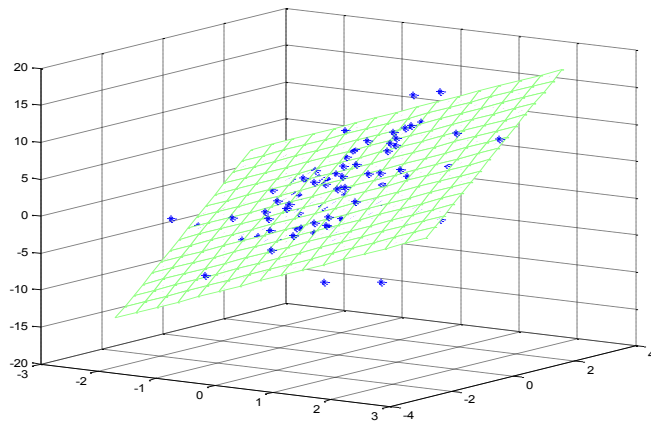
## Linear regression. Example

- 1 dimensional input  $\mathbf{x} = (x_1)$



## Linear regression. Example.

- 2 dimensional input  $\mathbf{x} = (x_1, x_2)$



## Linear regression. Optimization.

- We want the **weights minimizing the error**

$$J_n = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - f(\mathbf{x}_i))^2 = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

- For the optimal set of parameters, derivatives of the error with respect to each parameter must be 0

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) x_{i,j} = 0$$

- Vector of derivatives:**

$$\text{grad}_{\mathbf{w}}(J_n(\mathbf{w})) = \nabla_{\mathbf{w}}(J_n(\mathbf{w})) = -\frac{2}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i = \bar{\mathbf{0}}$$


---

## Linear regression. Optimization.

- $\text{grad}_{\mathbf{w}}(J_n(\mathbf{w})) = \bar{\mathbf{0}}$  defines a set of equations in  $\mathbf{w}$

$$\frac{\partial}{\partial w_0} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) = 0$$

$$\frac{\partial}{\partial w_1} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) x_{i,1} = 0$$

...

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) x_{i,j} = 0$$

...

$$\frac{\partial}{\partial w_d} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) x_{i,d} = 0$$


---

## Solving linear regression

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^n (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \dots - w_d x_{i,d}) x_{i,j} = 0$$

By rearranging the terms we get a **system of linear equations** with  $d+1$  unknowns

$$\mathbf{A}\mathbf{w} = \mathbf{b}$$

$$\begin{aligned} w_0 \sum_{i=1}^n x_{i,0} 1 + w_1 \sum_{i=1}^n x_{i,1} 1 + \dots + w_j \sum_{i=1}^n x_{i,j} 1 + \dots + w_d \sum_{i=1}^n x_{i,d} 1 &= \sum_{i=1}^n y_i 1 \\ w_0 \sum_{i=1}^n x_{i,0} x_{i,1} + w_1 \sum_{i=1}^n x_{i,1} x_{i,1} + \dots + w_j \sum_{i=1}^n x_{i,j} x_{i,1} + \dots + w_d \sum_{i=1}^n x_{i,d} x_{i,1} &= \sum_{i=1}^n y_i x_{i,1} \\ &\vdots \\ w_0 \sum_{i=1}^n x_{i,0} x_{i,j} + w_1 \sum_{i=1}^n x_{i,1} x_{i,j} + \dots + w_j \sum_{i=1}^n x_{i,j} x_{i,j} + \dots + w_d \sum_{i=1}^n x_{i,d} x_{i,j} &= \sum_{i=1}^n y_i x_{i,j} \\ &\vdots \end{aligned}$$

## Solving linear regression

- The optimal set of weights satisfies:

$$\nabla_{\mathbf{w}} (J_n(\mathbf{w})) = -\frac{2}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i = \bar{\mathbf{0}}$$

Leads to a **system of linear equations (SLE)** with  $d+1$  unknowns of the form

$$\mathbf{A}\mathbf{w} = \mathbf{b}$$

$$w_0 \sum_{i=1}^n x_{i,0} x_{i,j} + w_1 \sum_{i=1}^n x_{i,1} x_{i,j} + \dots + w_j \sum_{i=1}^n x_{i,j} x_{i,j} + \dots + w_d \sum_{i=1}^n x_{i,d} x_{i,j} = \sum_{i=1}^n y_i x_{i,j}$$

**Solution to SLE: ?**

## Solving linear regression

- The optimal set of weights satisfies:

$$\nabla_{\mathbf{w}}(J_n(\mathbf{w})) = -\frac{2}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i = \bar{\mathbf{0}}$$

Leads to a **system of linear equations (SLE)** with  $d+1$  unknowns of the form

$$\mathbf{Aw} = \mathbf{b}$$

$$w_0 \sum_{i=1}^n x_{i,0} x_{i,j} + w_1 \sum_{i=1}^n x_{i,1} x_{i,j} + \dots + w_j \sum_{i=1}^n x_{i,j} x_{i,j} + \dots + w_d \sum_{i=1}^n x_{i,d} x_{i,j} = \sum_{i=1}^n y_i x_{i,j}$$

**Solution to SLE:**

$$\mathbf{w} = \mathbf{A}^{-1} \mathbf{b}$$

- matrix inversion

## Gradient descent solution

**Goal:** the weight optimization in the linear regression model

$$J_n = \text{Error}(\mathbf{w}) = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

An alternative to SLE solution:

- Gradient descent**

**Idea:**

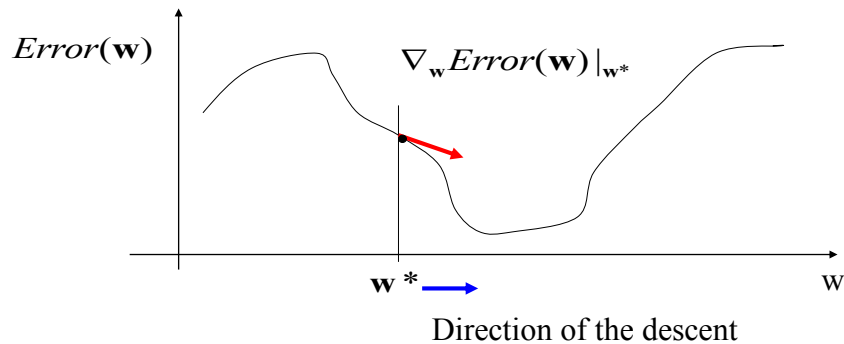
- Adjust weights in the direction that improves the Error
- The gradient tells us what is the right direction

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} \text{Error}_i(\mathbf{w})$$

$\alpha > 0$  - a **learning rate** (scales the gradient changes)

## Gradient descent method

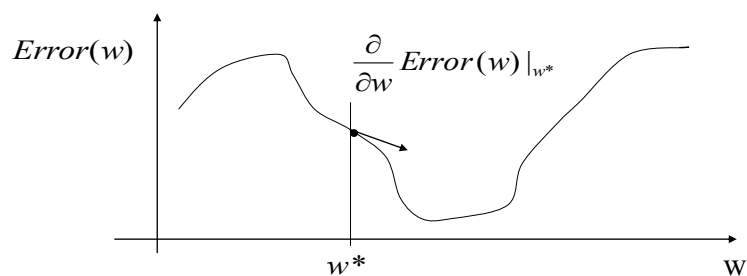
- Descend using the gradient information



- Change the value of  $w$  according to the gradient

$$w \leftarrow w - \alpha \nabla_w Error_i(w)$$

## Gradient descent method



- New value of the parameter

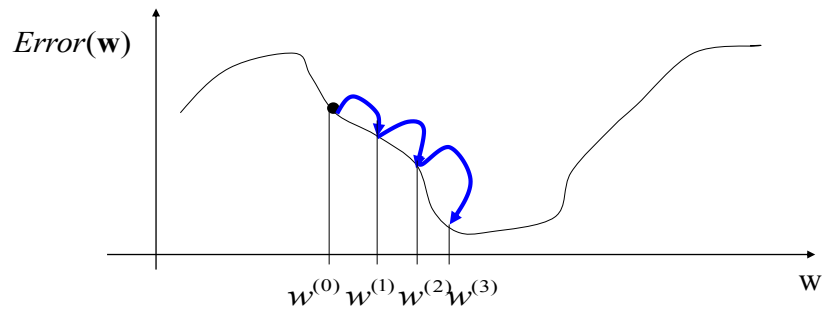
$$w_j \leftarrow w_j^* - \alpha \frac{\partial}{\partial w_j} Error(w)|_{w^*} \quad \text{For all } j$$

$\alpha > 0$  - a learning rate (scales the gradient changes)



## Gradient descent method

- Iteratively approaches the optimum of the Error function



## Online gradient algorithm

- The error function is defined for the whole dataset  $D$

$$J_n = Error(\mathbf{w}) = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

- error for a sample**  $D_i = \langle \mathbf{x}_i, y_i \rangle$

$$J_{\text{online}} = Error_i(\mathbf{w}) = \frac{1}{2} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

- Online gradient method: changes weights after every example**

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} Error_i(\mathbf{w})$$

- vector form:**

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} Error_i(\mathbf{w})$$

$\alpha > 0$  - Learning rate that depends on the number of updates

## Online gradient method

Linear model

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

On-line error

$$J_{online} = Error_i(\mathbf{w}) = \frac{1}{2}(y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

**On-line algorithm:** generates a sequence of online updates

**(i)-th update step with :**  $D_i = \langle \mathbf{x}_i, y_i \rangle$

**j-th weight:**

$$w_j^{(i)} \leftarrow w_j^{(i-1)} - \alpha(i) \frac{\partial Error_i(\mathbf{w})}{\partial w_j} \Big|_{\mathbf{w}^{(i-1)}}$$

$$w_j^{(i)} \leftarrow w_j^{(i-1)} + \alpha(i)(y_i - f(\mathbf{x}_i, \mathbf{w}^{(i-1)}))x_{i,j}$$

**Fixed learning rate:**  $\alpha(i) = C$

- Use a small constant

**Annealed learning rate:**  $\alpha(i) \approx \frac{1}{i}$

- Gradually rescales changes

## Online regression algorithm

**Online-linear-regression** (*stopping\_criterion*)

**Initialize weights**  $\mathbf{w} = (w_0, w_1, w_2 \dots w_d)$

**initialize i=1;**

**while** *stopping\_criterion* = FALSE

**select** the next data point  $D_i = (\mathbf{x}_i, y_i)$

**set** learning rate  $\alpha(i)$

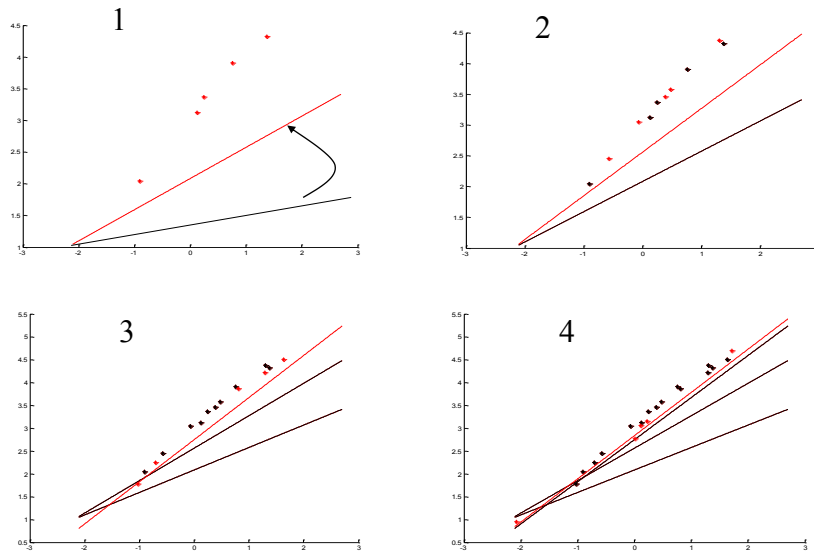
**update** weight vector  $\mathbf{w} \leftarrow \mathbf{w} + \alpha(i)(y_i - f(\mathbf{x}_i, \mathbf{w}))\mathbf{x}_i$

**end**

**return** weights

**Advantages:** very easy to implement, continuous data streams

## On-line learning. Example



## Practical concerns: Input normalization

### • Input normalization

- makes the data vary roughly on the same scale.
- Can make a huge difference in **on-line gradient learning**

**Example:** Assume on-line update rule for two weights  $j, k$ :

$$w_j \leftarrow w_j + \alpha(i)(y_i - f(\mathbf{x}_i))x_{i,j}$$

$$=$$

$$w_k \leftarrow w_k + \alpha(i)(y_i - f(\mathbf{x}_i))x_{i,k}$$

Change depends on the magnitude of the input

**Problem:** for inputs with a large magnitude the change in the weight is huge compared to the inputs with low magnitude: as if the input was more important

## Input normalization

- **Input normalization:**

- Solution to the problem of different scales
- Makes all inputs vary in the same range around 0

$$\bar{x}_j = \frac{1}{n} \sum_{i=1}^n x_{i,j} \quad \sigma_j^2 = \frac{1}{n-1} \sum_{i=1}^n (x_{i,j} - \bar{x}_j)^2$$

**New input:**  $\tilde{x}_{i,j} = \frac{(x_{i,j} - \bar{x}_j)}{\sigma_j}$

More complex normalization approach can be applied when we want to process data with correlations

**Similarly we can renormalize outputs  $y$**

---