CS 1571 Introduction to AI
Lecture 24

Linear regression.

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Announcements

Homeworks:

- Homework 9 due today
- Homework 10 out today (due November 26, 2002)
  - **Programming part:**
    - Learning classification of handwritten digits

Final:

- December 11, 2002 at 2:00-3:50pm
- Location: Sennott Square 5502

Supervised learning

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

$D_i =< \mathbf{x}_i, y_i >$

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

y_i is the desired output (given by a teacher)

Objective: learn the mapping $f : X \to Y$

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

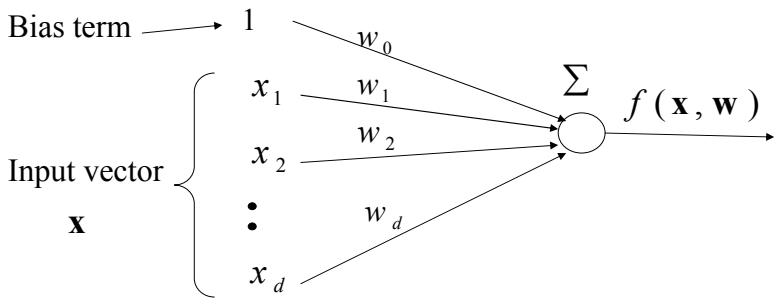
- **Regression:** Y is **continuous**
  Example: earnings, product orders → company stock price
- **Classification:** Y is **discrete**
  Example: handwritten digit in binary form → digit label

Linear regression

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

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

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



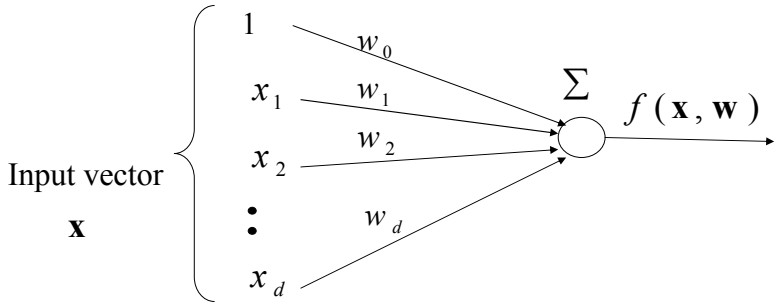
Linear regression

- **Shorter (vector) definition of the model**
  - Include bias constant in the input vector

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

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

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



Linear regression. Error.

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

- **Error function** measures how much our predictions deviate from the desired answers

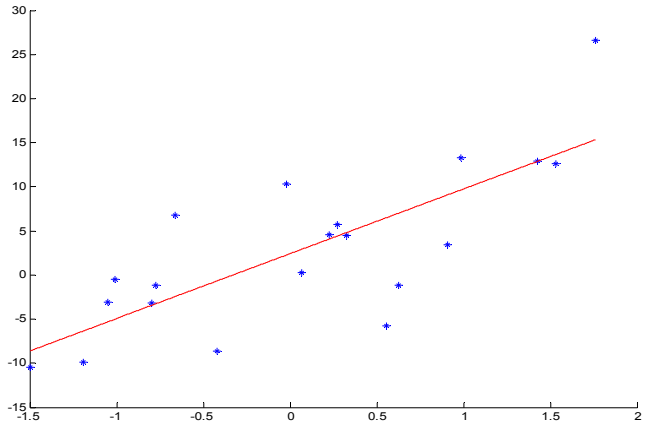
Mean-squared error $J_n = \frac{1}{n} \sum_{i=1,..n} (y_i - f(\mathbf{x}_i))^2$

- **Learning:**

  We want to find the weights minimizing the error !

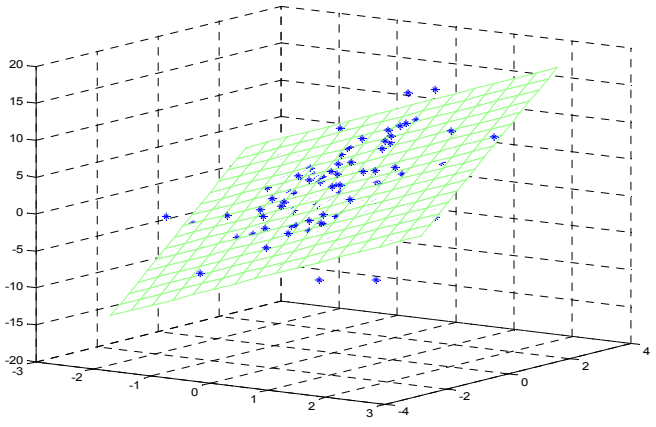
Linear regression. Example

- 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**

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

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

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

- **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.

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

$$J_n = \frac{1}{n}\sum_{i=1,..n}(y_i - f(\mathbf{x}_i))^2 = \frac{1}{n}\sum_{i=1,..n}(y_i - [w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \ldots w_k x^{(k)}])^2$$

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

$$\frac{\partial}{\partial w_0}J_n(w) = -\frac{2}{n}\sum_{i=1}^{n}[y_i - (w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \ldots w_k x^{(k)})] = 0$$

$$\frac{\partial}{\partial w_1}J_n(w) = -\frac{2}{n}\sum_{i=1}^{n}[y_i - (w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \ldots w_k x^{(k)})]x^{(1)} = 0$$

$$\ldots$$

$$\frac{\partial}{\partial w_j}J_n(w) = -\frac{2}{n}\sum_{i=1}^{n}[y_i - (w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \ldots w_k x^{(k)})]x^{(j)} = 0$$

$$\ldots$$

Solving linear regression

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

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

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

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

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

$$\bullet\bullet\bullet$$

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

$$\bullet\bullet\bullet$$

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

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

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

Solutions to SLE:

• e.g. matrix inversion (if the matrix is singular)

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

Gradient descent solution

- There are other ways to solve the weight optimization problem in the linear regression model

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

- A simple technique:
  - **Gradient descent**

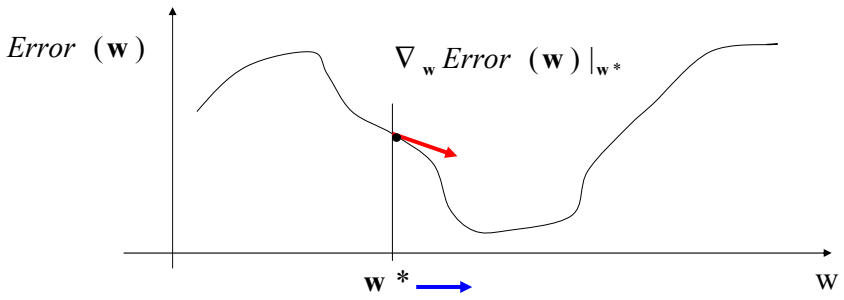
    Idea:
    - Adjust weights in the direction that improves the Error
    - The gradient tells us what is the right direction

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

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

Gradient descent method

- Descend using the gradient information



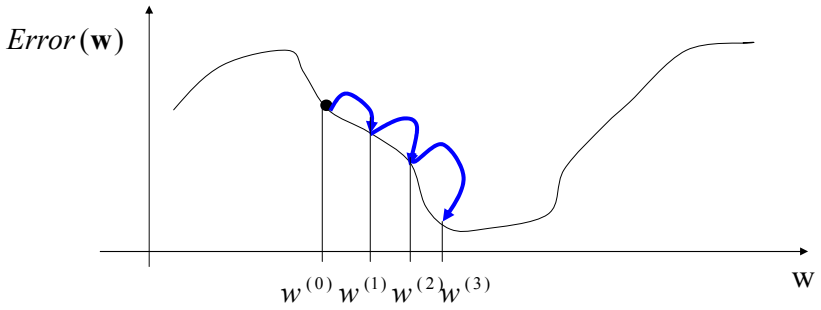
Direction of the descent

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

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

Gradient descent method

- Iteratively converge to the optimum of the Error function



Online gradient descent algorithm

- The error function defined for the whole dataset D

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

- Instead of the error for all data points we **use error for an individual sample** $D_i =< \mathbf{x}_i, y_i >$

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

- **Change regression weights after every example according to the gradient:**

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

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

Gradient for on-line learning

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: sequence of online updates

(i)-th update for the linear model: $D_i =< \mathbf{x}_i, y_i >$

Vector form:

$$\mathbf{w}^{(i)} \leftarrow \mathbf{w}^{(i-1)} - \alpha(i)\nabla_{\mathbf{w}} Error_i(\mathbf{w})|_{\mathbf{w}^{(i-1)}} = \mathbf{w}^{(i-1)} + \alpha(i)(y_i - f(\mathbf{x}_i, \mathbf{w}^{(i-1)}))\mathbf{x}_i$$

j-th weight:

$$w_j^{(i)} \leftarrow w_j^{(i-1)} - \alpha(i)\frac{\partial Error_i(\mathbf{w})}{\partial w_j}|_{\mathbf{w}^{(i-1)}} = w_j^{(i-1)} + \alpha(i)(y_i - f(\mathbf{x}_i, \mathbf{w}^{(i-1)}))x_{i,j}$$

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

- Gradually rescales changes in weights

Online regression algorithm

Online-linear-regression (*D, number of iterations*)
  Initialize weights $\mathbf{w} = (w_0, w_1, w_2 \ldots w_d)$
  for *i*=1:1*: number of iterations*
    do **select** a data point $D_i = (\mathbf{x}_i, y_i)$ from D
      set $\alpha = 1/i$
      update weight vector
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha(y_i - f(\mathbf{x}_i, \mathbf{w}))\mathbf{x}_i$
  end for
  return weights $\mathbf{w}$

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

On-line learning. Example