CS 1675 Intro to Machine Learning
Lecture 9

Linear regression

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

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)

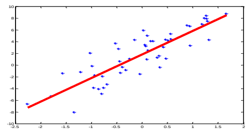
Objective: learn the mapping $f : X \rightarrow Y$

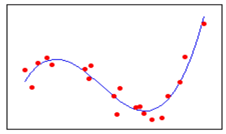
s.t. $y_i \approx f(\mathbf{x}_i)$ for all $i = 1,.., n$

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

Supervised learning examples

- **Regression:** Y is **continuous**





Debt/equity
Earnings
Future product orders → Stock price

Data:

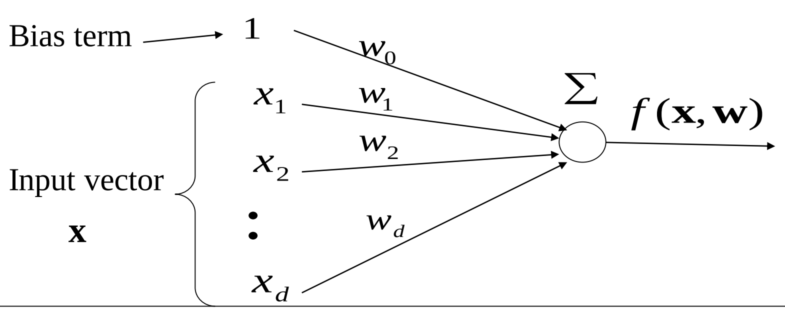
Debt/equity	Earnings	Future prod orders	Stock price
20	115	20	123.45
18	120	31	140.56
….			

Linear regression

- **Function** $f : X \to Y$
- 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)**



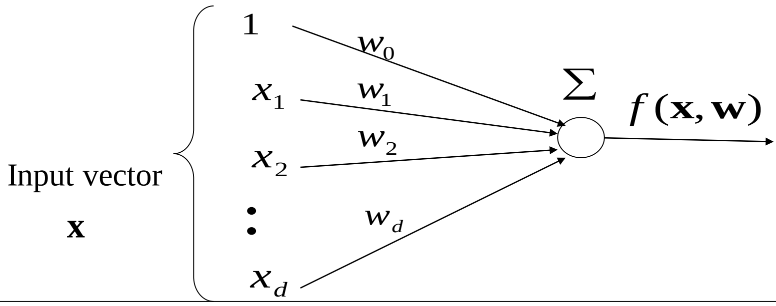
Linear regression

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

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

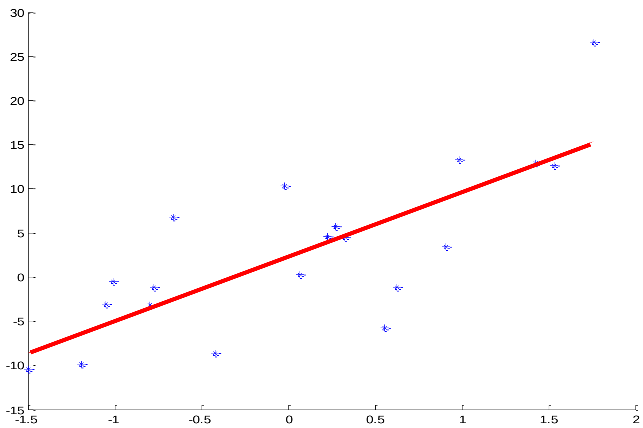
$$f(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots w_d x_d = \mathbf{w}^T \mathbf{x}$$

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



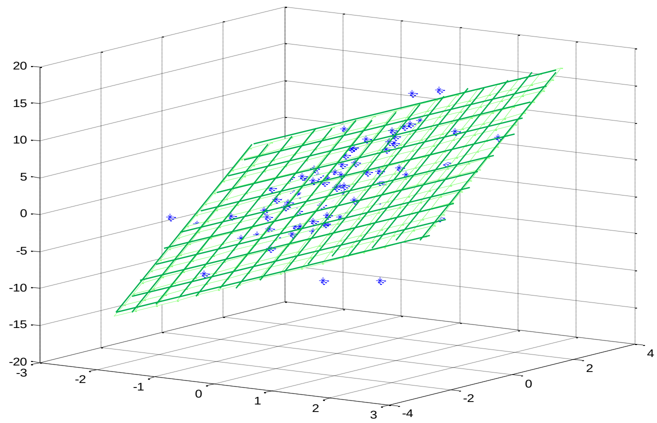
Linear regression. Example

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



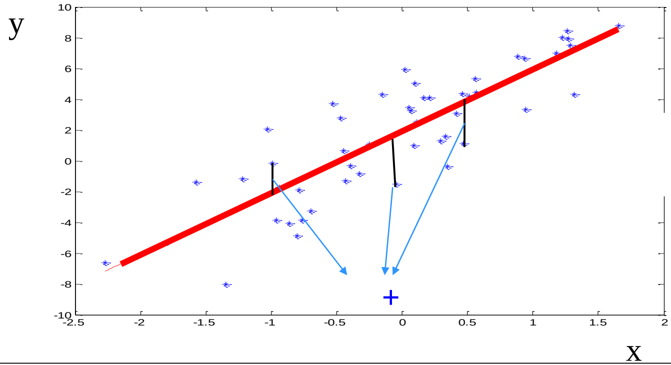
Linear regression. Example.

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



Linear regression: error

- **Data:** $D_i = < \mathbf{x}_i, y_i >$
- **Function:** $\mathbf{x}_i \rightarrow f(\mathbf{x}_i)$
- **Goal:** find the **best set** of model parameters
- **Error**: a measure of misfit of the model and the data



Linear regression: Error.

- **Data:** $D_i = < \mathbf{x}_i, y_i >$
- **Function:** $\mathbf{x}_i \rightarrow f(\mathbf{x}_i)$
- **Goal:** find the **best set** of model parameters
- **Error function**
  – a measure of misfit of the model and the data
  – in other words, it measures how much our predictions deviate from the desired answers

Mean-squared error: *Error(w, D)*

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

- **Learning:**
  We want to find the weights minimizing the error !

Linear regression: Optimization.

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

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

- 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:**

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

Linear regression: Optimization.

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

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

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

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

$$\ldots$$

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

$$\ldots$$

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

Solving linear regression

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

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

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

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

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

$$\cdots$$

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

$$\cdots$$

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

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

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

Solution to SLE: $\mathbf{w} = \mathbf{A}^{-1}\mathbf{b}$

Assuming $\mathbf{X}$ is an *nxd* data matrix with rows corresponding to examples and columns to inputs, and $\boldsymbol{y}$ is nx1 vector of outputs, then $\mathbf{w} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$

Gradient descent solution

Objective: optimize the weights in the linear regression model

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

An alternative to SLE solution:

- **Gradient descent**

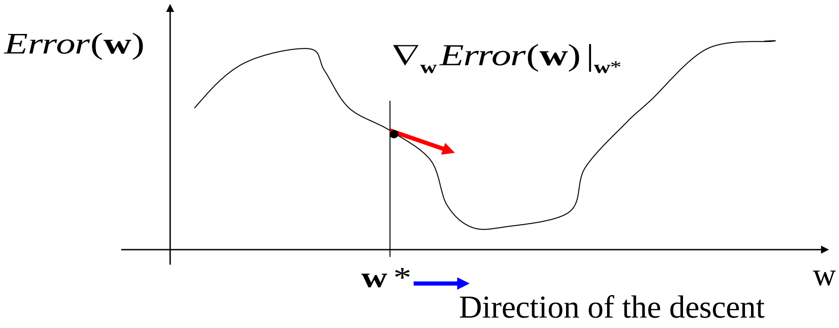
  Idea:
  - Adjust weights in the direction that improves the Error
  - The gradient tells us what is the right direction

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

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

Gradient descent method

- Descend using the gradient information



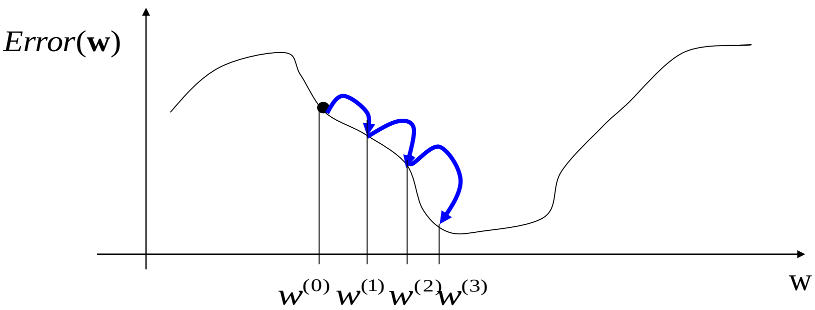
- Change the value of **w** according to the gradient

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

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

Gradient descent method

- Iteratively approaches the optimum of the Error function



Batch vs online gradient algorithm

- The error function defined on the complete dataset *D*

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

- We say we are learning the model in **the batch mode**:
  - All examples are available at the time of learning
  - Weights are optimizes with respect to all training examples

- An alternative is to learn the model in **the online mode**
  - Examples are arriving sequentially
  - Model weights are updated after every example
  - If needed examples seen can be forgotten

-

Online gradient algorithm

- **The error function** for the complete dataset *D*

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

- **Error for one example** $D_i = < \mathbf{x}_i, y_i >$

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

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

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

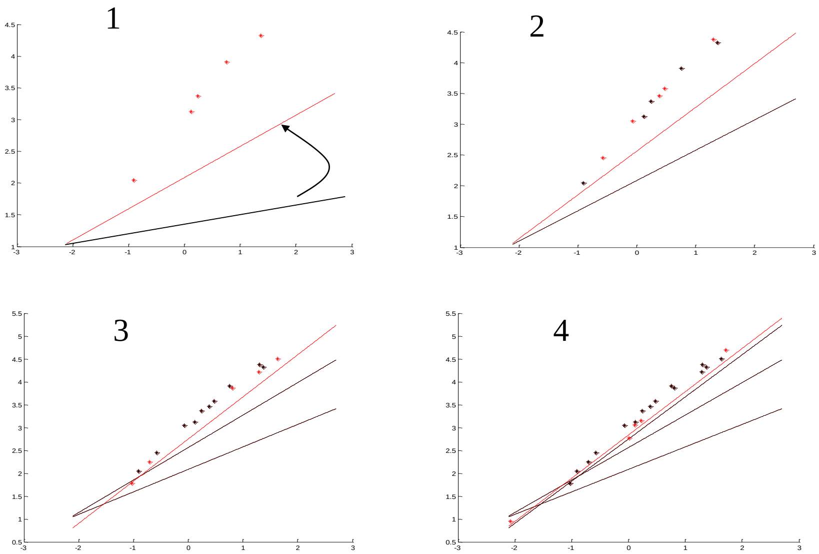
- Gradually rescales changes

Online regression algorithm

Online-linear-regression (*stopping_criterion*)
    Initialize weights $\mathbf{w} = (w_0, w_1, w_2 \ldots w_d)$
    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, works on continuous data streams

On-line learning. Example

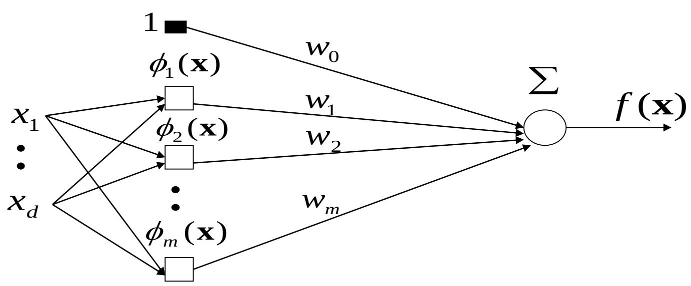


Extensions of simple linear model

Replace inputs to linear units with m **feature (basis) functions** to model **nonlinearities**

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^{m} w_j \phi_j(\mathbf{x})$$

$\phi_j(\mathbf{x})$ - an arbitrary function of $\mathbf{x}$



Original input ⇒ **New input** ⇒ **Linear model**

Extensions of simple linear model

- **Models linear in the parameters we want to fit**

$$f(\mathbf{x}) = w_0 + \sum_{k=1}^{m} w_k \phi_k(\mathbf{x})$$

$w_0, w_1 \ldots w_m$ - parameters

$\phi_1(\mathbf{x}), \phi_2(\mathbf{x}) \ldots \phi_m(\mathbf{x})$ - **feature or basis functions**

- **Basis functions examples:**
  - **a higher order polynomial, one-dimensional input** $\mathbf{x} = (x_1)$

    $\phi_1(x) = x \quad \phi_2(x) = x^2 \quad \phi_3(x) = x^3$

Extensions of simple linear model

- **Models linear in the parameters we want to fit**

$$f(\mathbf{x}) = w_0 + \sum_{k=1}^{m} w_k \phi_k(\mathbf{x})$$

$w_0, w_1 \ldots w_m$ - parameters

$\phi_1(\mathbf{x}), \phi_2(\mathbf{x}) \ldots \phi_m(\mathbf{x})$ - **feature or basis functions**

- **Basis functions examples:**
  - a higher order polynomial, one-dimensional input $\mathbf{x} = (x_1)$

    $\phi_1(x) = x \quad \phi_2(x) = x^2 \quad \phi_3(x) = x^3$

  - **Multidimensional quadratic** $\mathbf{x} = (x_1, x_2)$

$\phi_1(\mathbf{x}) = x_1 \quad \phi_2(\mathbf{x}) = x_1^2 \quad \phi_3(\mathbf{x}) = x_2 \quad \phi_4(\mathbf{x}) = x_2^2 \quad \phi_5(\mathbf{x}) = x_1 x_2$

Extensions of simple linear model

- **Models linear in the parameters we want to fit**

$$f(\mathbf{x}) = w_0 + \sum_{k=1}^{m} w_k \phi_k(\mathbf{x})$$

$w_0, w_1 \dots w_m$ - parameters

$\phi_1(\mathbf{x}), \phi_2(\mathbf{x}) \dots \phi_m(\mathbf{x})$ - **feature or basis functions**

- **Basis functions examples:**
  - a higher order polynomial, one-dimensional input $\mathbf{x} = (x_1)$

    $\phi_1(x) = x \quad \phi_2(x) = x^2 \quad \phi_3(x) = x^3$
  - Multidimensional quadratic $\mathbf{x} = (x_1, x_2)$

    $\phi_1(\mathbf{x}) = x_1 \quad \phi_2(\mathbf{x}) = x_1^2 \quad \phi_3(\mathbf{x}) = x_2 \quad \phi_4(\mathbf{x}) = x_2^2 \quad \phi_5(\mathbf{x}) = x_1 x_2$
  - **Other types of basis functions**

    $\phi_1(x) = \sin x \quad \phi_2(x) = \cos x$

Extensions of simple linear model

The same techniques as for the linear model to learn the weights

- **Error function** $\quad J_n = 1/n \sum_{i=1,..n} (y - f(\mathbf{x}_i, \mathbf{w}))^2$

Assume: $\quad \boldsymbol{\varphi}(\mathbf{x}_i) = (1, \phi_1(\mathbf{x}_i), \phi_2(\mathbf{x}_i), \dots, \phi_m(\mathbf{x}_i))$

$$\nabla_{\mathbf{w}} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1,..n} (y_i - f(\mathbf{x}_i)) \boldsymbol{\varphi}(\mathbf{x}_i) = \bar{\mathbf{0}}$$

- Leads to a **system of *m* linear equations**

$$w_0 \sum_{i=1}^{n} 1\phi_j(\mathbf{x}_i) + \dots + w_j \sum_{i=1}^{n} \phi_j(\mathbf{x}_i)\phi_j(\mathbf{x}_i) + \dots + w_m \sum_{i=1}^{n} \phi_m(\mathbf{x}_i)\phi_j(\mathbf{x}_i) = \sum_{i=1}^{n} y_i \phi_j(\mathbf{x}_i)$$

- Can be solved exactly like the linear case

Example. Regression with polynomials.

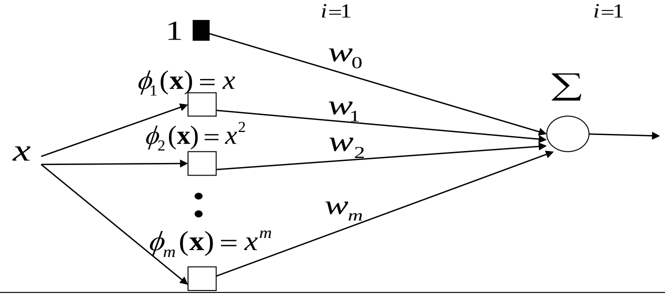
Regression with polynomials of degree m

- **Data instances:** pairs of $< x, y >$
- **Feature functions:** m feature functions

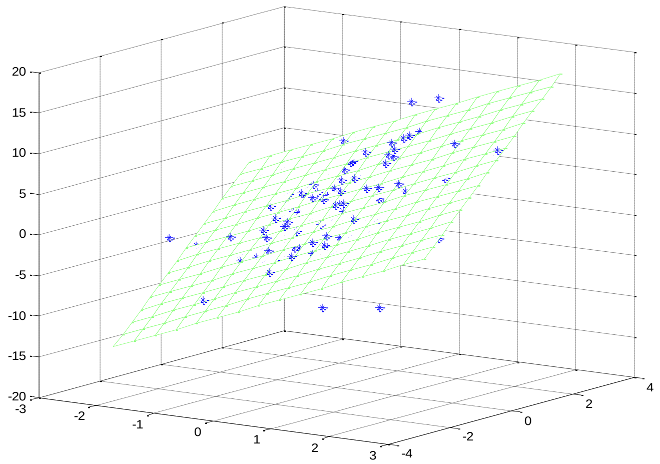
$$\phi_i(x) = x^i \qquad i = 1, 2, \ldots, m$$

- **Function to learn:**

$$f(x, \mathbf{w}) = w_0 + \sum_{i=1}^{m} w_i \phi_i(x) = w_0 + \sum_{i=1}^{m} w_i x^i$$



Linear model example



Non-linear (quadratic) model