

CS 2750 Machine Learning Lecture 8

Linear regression

Milos Hauskrecht

milos@cs.pitt.edu

5329 Sennott Square

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 $f : X \rightarrow Y$

s.t. $y_i \approx f(\mathbf{x}_i)$ 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
-

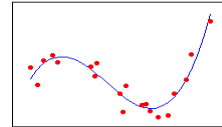
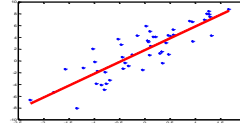
Supervised learning examples

- **Regression:** Y is continuous

Debt/equity
Earnings
Future product orders



Stock price



Data:

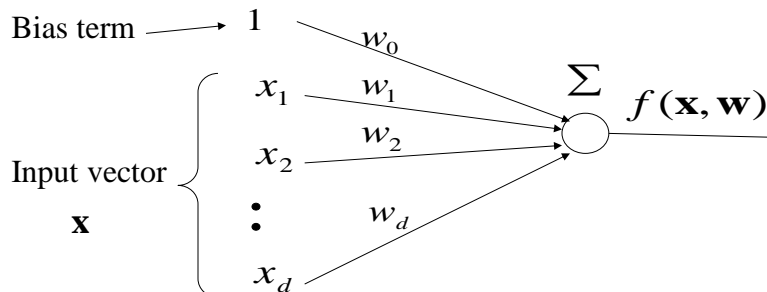
Debt/equity	Earnings	Future prod orders	Stock price
20	115	20	123.45
18	120	31	140.56
....			

Linear regression

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

$$f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d = w_0 + \sum_{j=1}^d w_jx_j$$

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



Linear regression

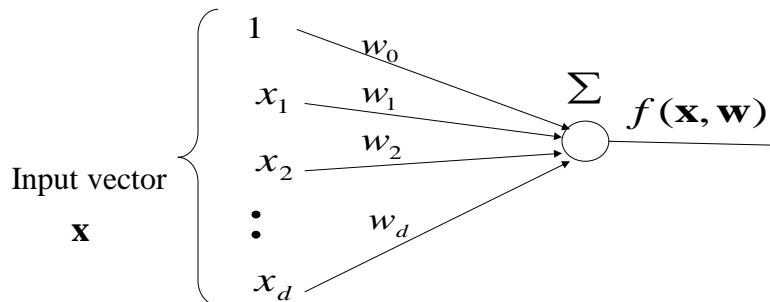
- **Shorter (vector) definition of the model**

- 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

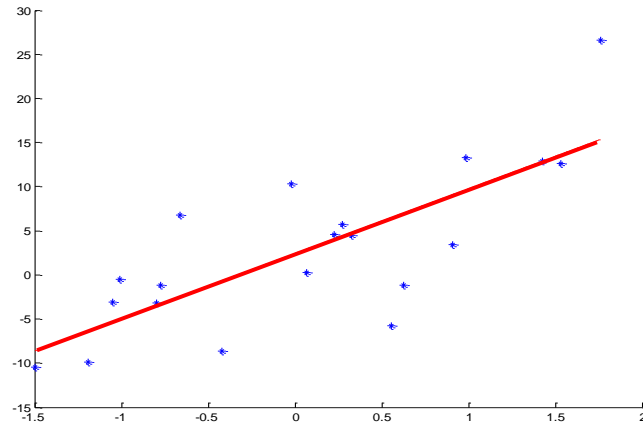
Mean-squared error $J_n = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - f(\mathbf{x}_i))^2$

- **Learning:**

We want to find the weights minimizing the error !

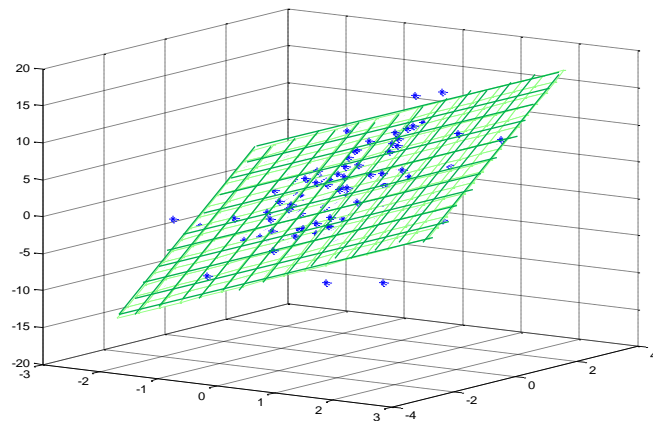
Linear regression. Example

- 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**

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

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

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

- **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}

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

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

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

...

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

...

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

Solving linear regression

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

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

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

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

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:

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

Assuming \mathbf{X} is an $n \times d$ data matrix with rows corresponding to examples and columns to inputs, and \mathbf{y} is $n \times 1$ vector of outputs, then

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

Gradient descent solution

Goal: the weight optimization in the linear regression model

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

An alternative to SLE solution:

- **Gradient descent**

Idea:

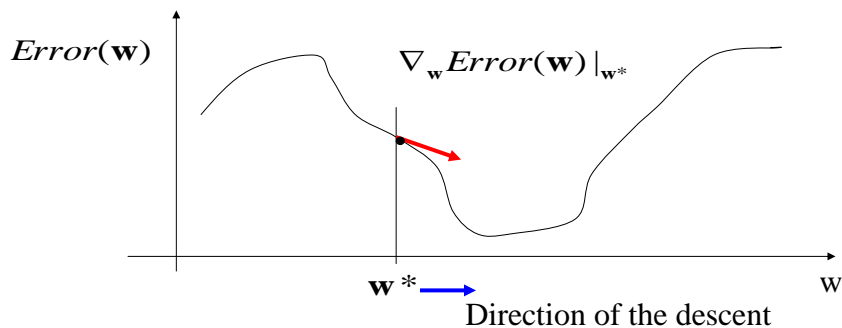
- Adjust weights in the direction that improves the Error
- The gradient tells us what is the right direction

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

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

Gradient descent method

- Descend using the gradient information



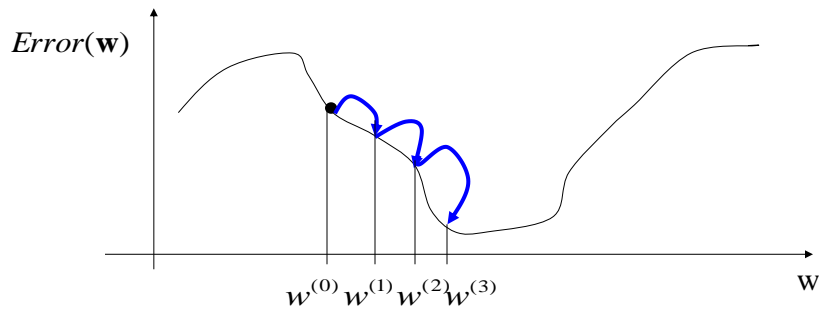
- Change the value of \mathbf{w} according to the gradient

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

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

Gradient descent method

- Iteratively approaches the optimum of the Error function



Batch vs Online regression 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 optimized 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 is defined for the complete dataset D

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

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

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

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

- vector form:** $w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} Error_i(\mathbf{w})$

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

$\alpha > 0$ - 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: $\alpha(i) = C$ **Annealed learning rate:** $\alpha(i) \approx \frac{1}{i}$

- Use a small constant

- 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 $\alpha(i)$

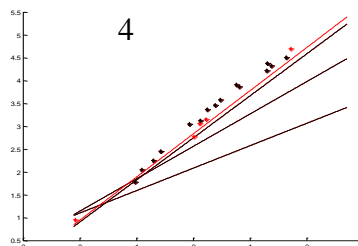
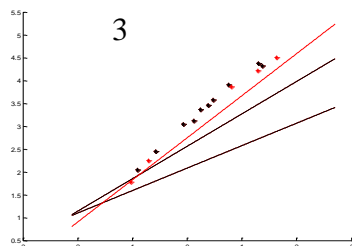
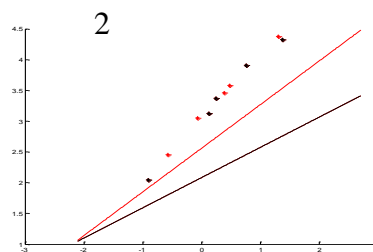
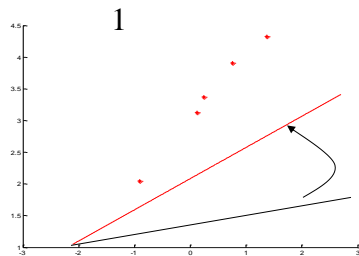
update weight vector $\mathbf{w} \leftarrow \mathbf{w} + \alpha(i)(y_i - f(\mathbf{x}_i, \mathbf{w}))\mathbf{x}_i$

end

return weights

Advantages: very easy to implement, continuous data streams

On-line learning. Example

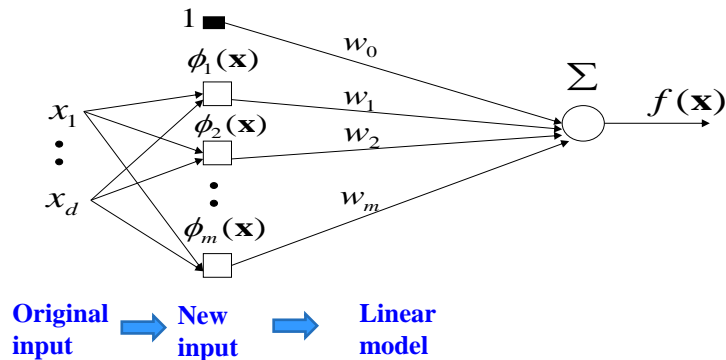


Extensions of simple linear model

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}



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 $\mathbf{x} = (x_1)$

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

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

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

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

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

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

- 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.

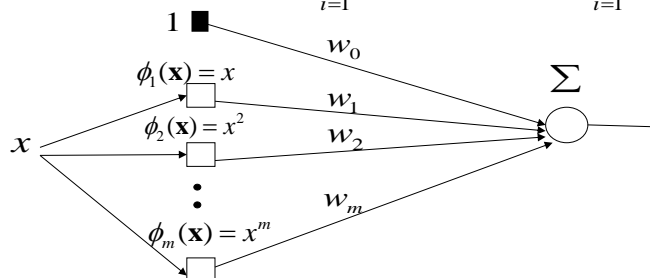
Regression with polynomials of degree m

- **Data instances:** pairs of $\langle x, y \rangle$
- **Feature functions:** m feature functions

$$\phi_i(x) = x^i \quad i = 1, 2, \dots, m$$

- **Function to learn:**

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



Learning with feature functions

Function to learn:

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

On line gradient update for the $\langle \mathbf{x}, y \rangle$ pair

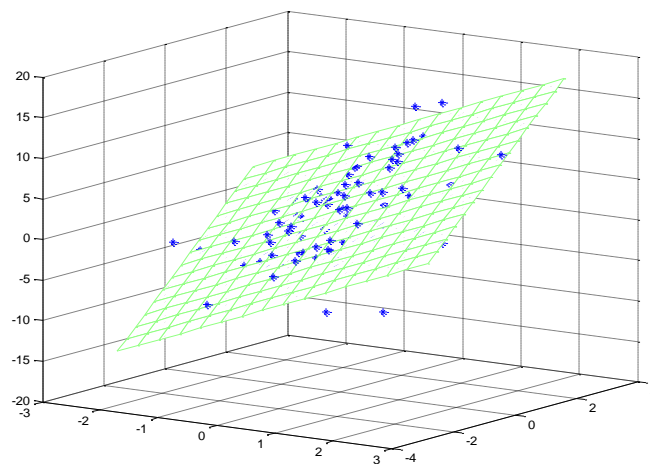
$$w_0 = w_0 + \alpha(y - f(\mathbf{x}, \mathbf{w}))$$

•

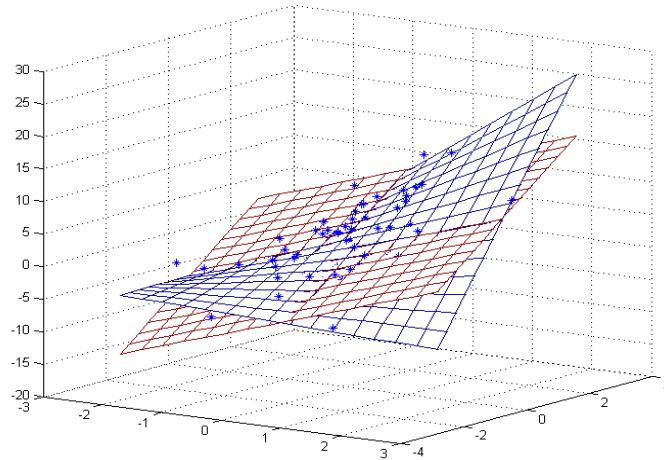
$$w_j = w_j + \alpha(y - f(\mathbf{x}, \mathbf{w}))\phi_j(\mathbf{x})$$

Gradient updates are of the same form as in the linear regression models

Linear model example

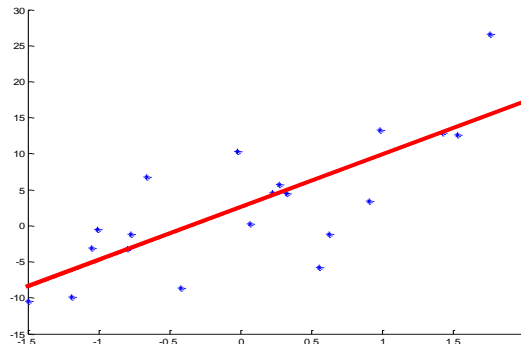


Non-linear model



Linear regression model

- **Linear model:** $y = f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x}$



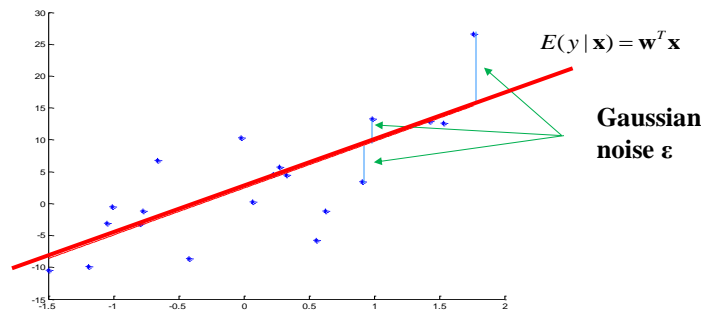
- **Notice:** the above model does not try to explain variation in observed y s for the data

Statistical model of regression

A statistical model of linear regression:

$$y = \mathbf{w}^T \mathbf{x} + \varepsilon \quad \varepsilon \sim N(0, \sigma^2)$$

ε is a random noise, represents things we cannot capture with $\mathbf{w}^T \mathbf{x}$



$$y \sim N(\mathbf{w}^T \mathbf{x}, \sigma^2)$$

Statistical model of regression

A statistical model of linear regression:

$$y = \mathbf{w}^T \mathbf{x} + \varepsilon \quad \varepsilon \sim N(0, \sigma^2)$$

$$y \sim N(\mathbf{w}^T \mathbf{x}, \sigma^2)$$

- The conditional distribution of y given \mathbf{x}

$$p(y | \mathbf{x}, \mathbf{w}, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left[-\frac{1}{2\sigma^2} (y - \mathbf{w}^T \mathbf{x})^2 \right]$$

$$E(y | \mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

ML estimation of the parameters

- **likelihood of predictions** = the probability of observing outputs y in D given \mathbf{w}, σ

$$L(D, \mathbf{w}, \sigma) = \prod_{i=1}^n p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma)$$

- **Maximum likelihood estimation of parameters \mathbf{w}**
 - parameters maximizing the likelihood of predictions

$$\mathbf{w}^* = \arg \max_{\mathbf{w}} \prod_{i=1}^n p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma)$$

- **Log-likelihood** trick for the ML optimization
 - Maximizing the log-likelihood is equivalent to maximizing the likelihood

$$l(D, \mathbf{w}, \sigma) = \log(L(D, \mathbf{w}, \sigma)) = \log \prod_{i=1}^n p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma)$$

ML estimation of the parameters

- **Using conditional density**

$$p(y | \mathbf{x}, \mathbf{w}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2\sigma^2} (y - f(\mathbf{x}, \mathbf{w}))^2\right]$$

- **We can rewrite the log-likelihood as**

$$\begin{aligned} l(D, \mathbf{w}, \sigma) &= \log(L(D, \mathbf{w}, \sigma)) = \log \prod_{i=1}^n p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma) \\ &= \sum_{i=1}^n \log p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma) = \sum_{i=1}^n \left\{ -\frac{1}{2\sigma^2} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 - c(\sigma) \right\} \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + C(\sigma) \end{aligned}$$

Did we see a similar expression before?

ML estimation of the parameters

- Using conditional density

$$p(y | \mathbf{x}, \mathbf{w}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2\sigma^2} (y - f(\mathbf{x}, \mathbf{w}))^2\right]$$

- We can rewrite the log-likelihood as

$$\begin{aligned} l(D, \mathbf{w}, \sigma) &= \log(L(D, \mathbf{w}, \sigma)) = \log \prod_{i=1}^n p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma) \\ &= \sum_{i=1}^n \log p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma) = \sum_{i=1}^n \left\{ -\frac{1}{2\sigma^2} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 - c(\sigma) \right\} \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + C(\sigma) \end{aligned}$$

- Maximizing the predictive log likelihood with regard to \mathbf{w} , is equivalent to minimizing the mean squared error function

ML estimation of parameters

- Criteria based on the mean squares error function and the log likelihood of the output are related

$$J_{online}(y_i, \mathbf{x}_i) = \frac{1}{2\sigma^2} \log p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma) + c(\sigma)$$

- We know how to optimize parameters w
 - the same approach as used for the least squares fit
- But what is the ML estimate of the variance of the noise?
- Maximize $l(D, \mathbf{w}, \sigma)$ with respect to variance

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

= mean square prediction error for the best predictor