Introduction to Machine Learning

Inas A. Yassine, PhD

Assoc. Prof. ,

Systems and Biomedical Engineering Department,

Cairo University

inas.yassine@eng.cu.edu.eg

Cairo University

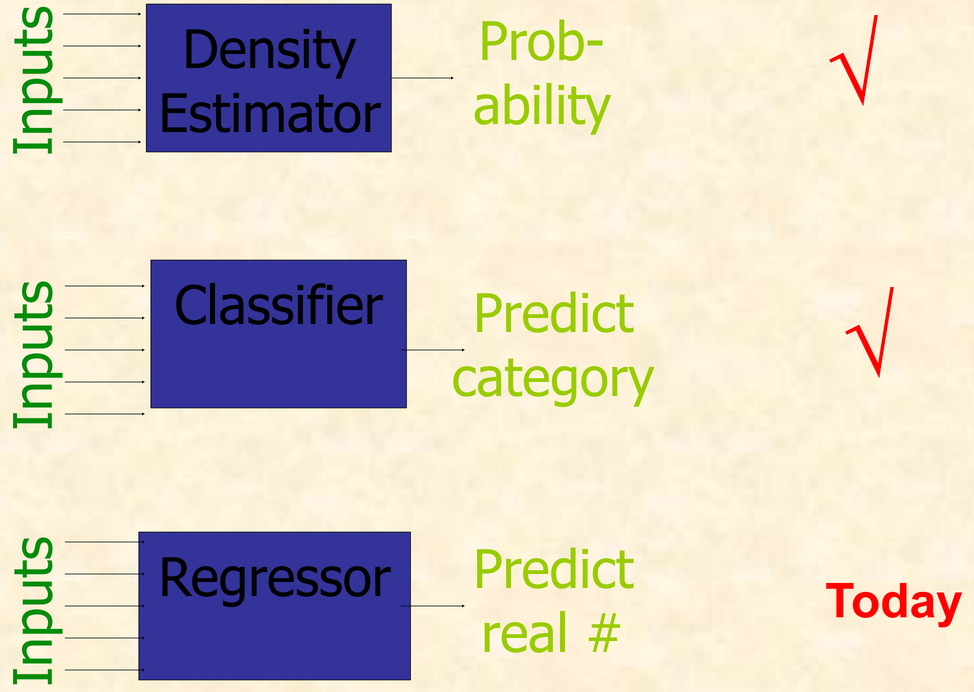
Regression in Machine Learning

Outline

- Regression *vs* Classification
- Linear regression – another discriminative learning method
  - As optimization ➔ Gradient descent
  - As matrix inversion (<u>O</u>rdinary <u>L</u>east <u>S</u>quares)
- Overfitting

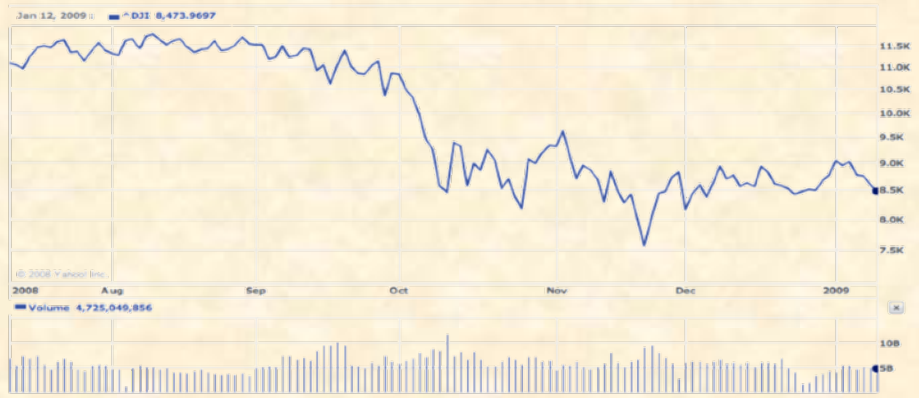
What is linear Regression

Regression/ Classification/ [Confidence]

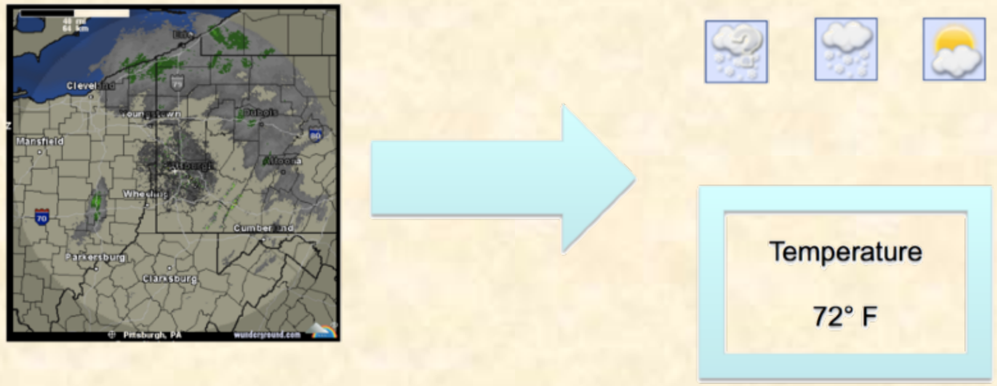


Regression examples

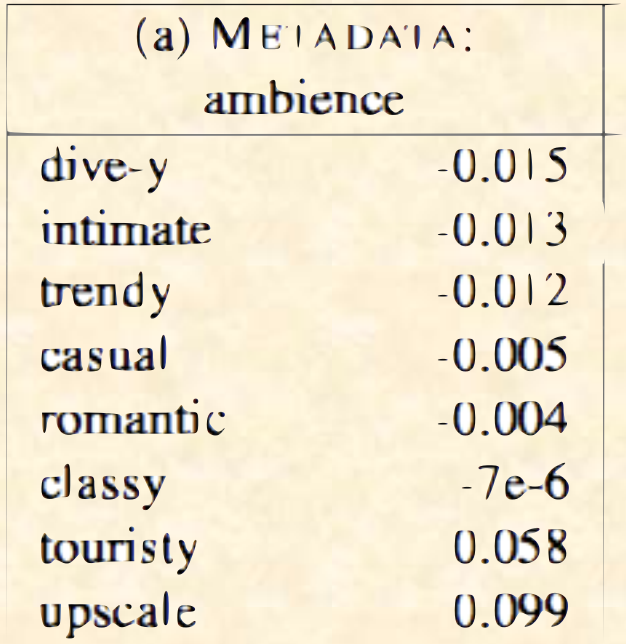
Stock Market Estimation



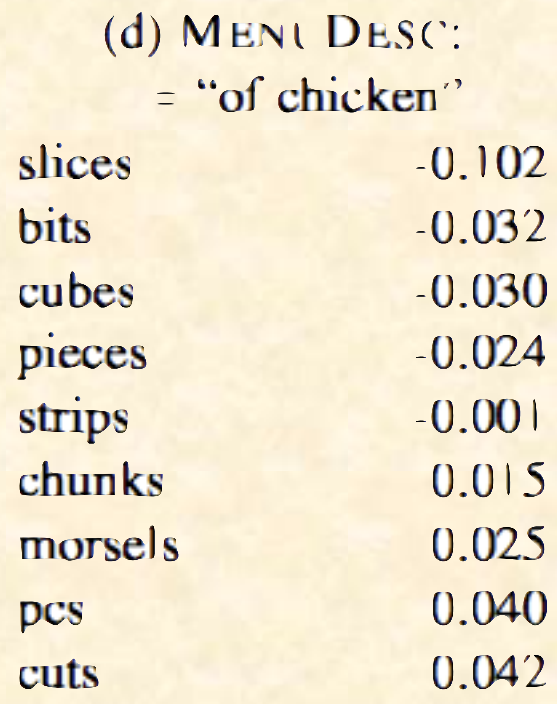
Temperature Estimation



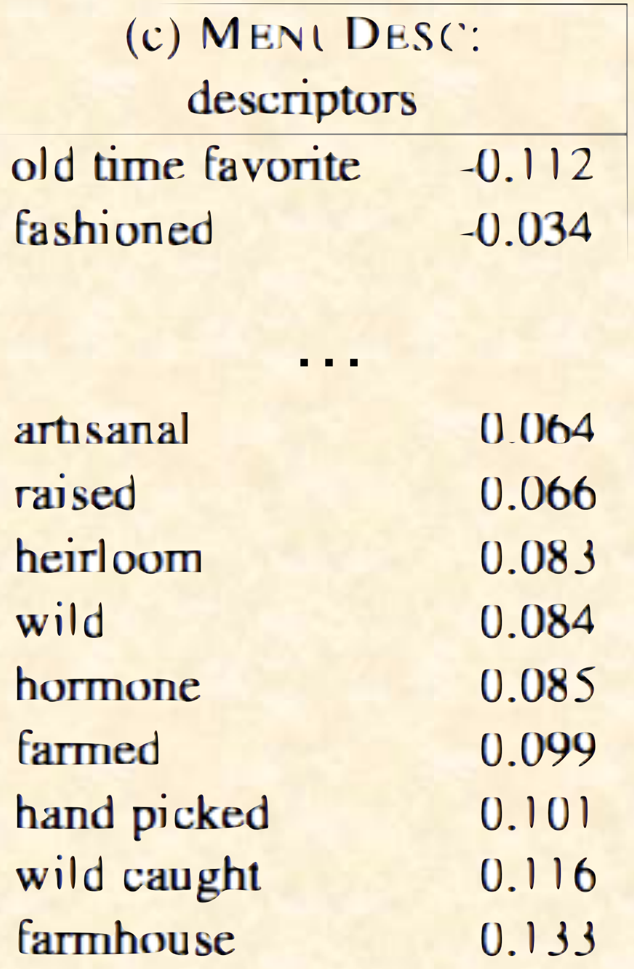
Prediction of menu prices



(a) Metadata: ambience	
dive-y	-0.015
intimate	-0.013
trendy	-0.012
casual	-0.005
romantic	-0.004
classy	-7e-6
touristy	0.058
upscale	0.099

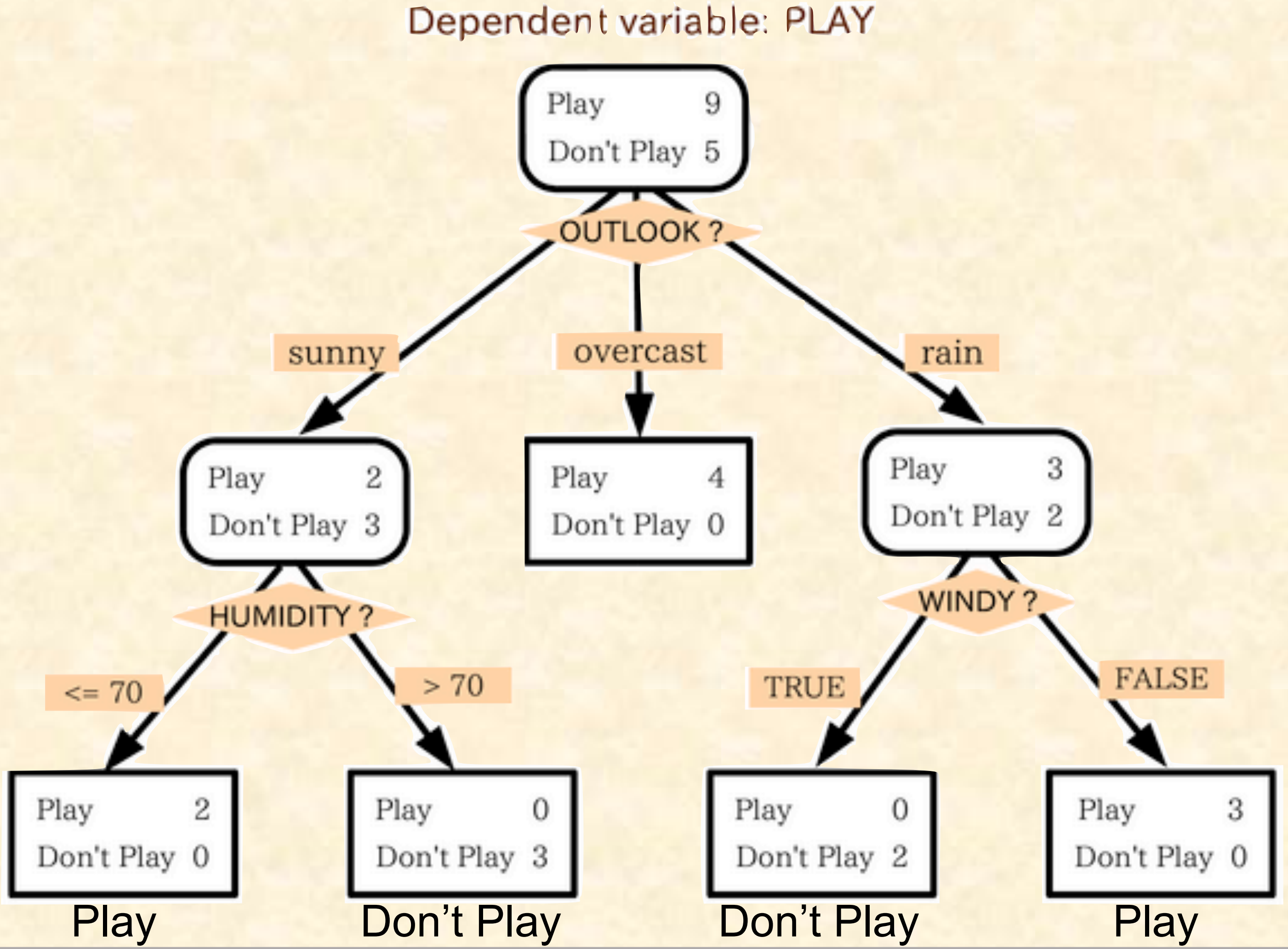


(d) Menu Desc: = "of chicken"	
slices	-0.102
bits	-0.032
cubes	-0.030
pieces	-0.024
strips	-0.001
chunks	0.015
morsels	0.025
pcs	0.040
cuts	0.042

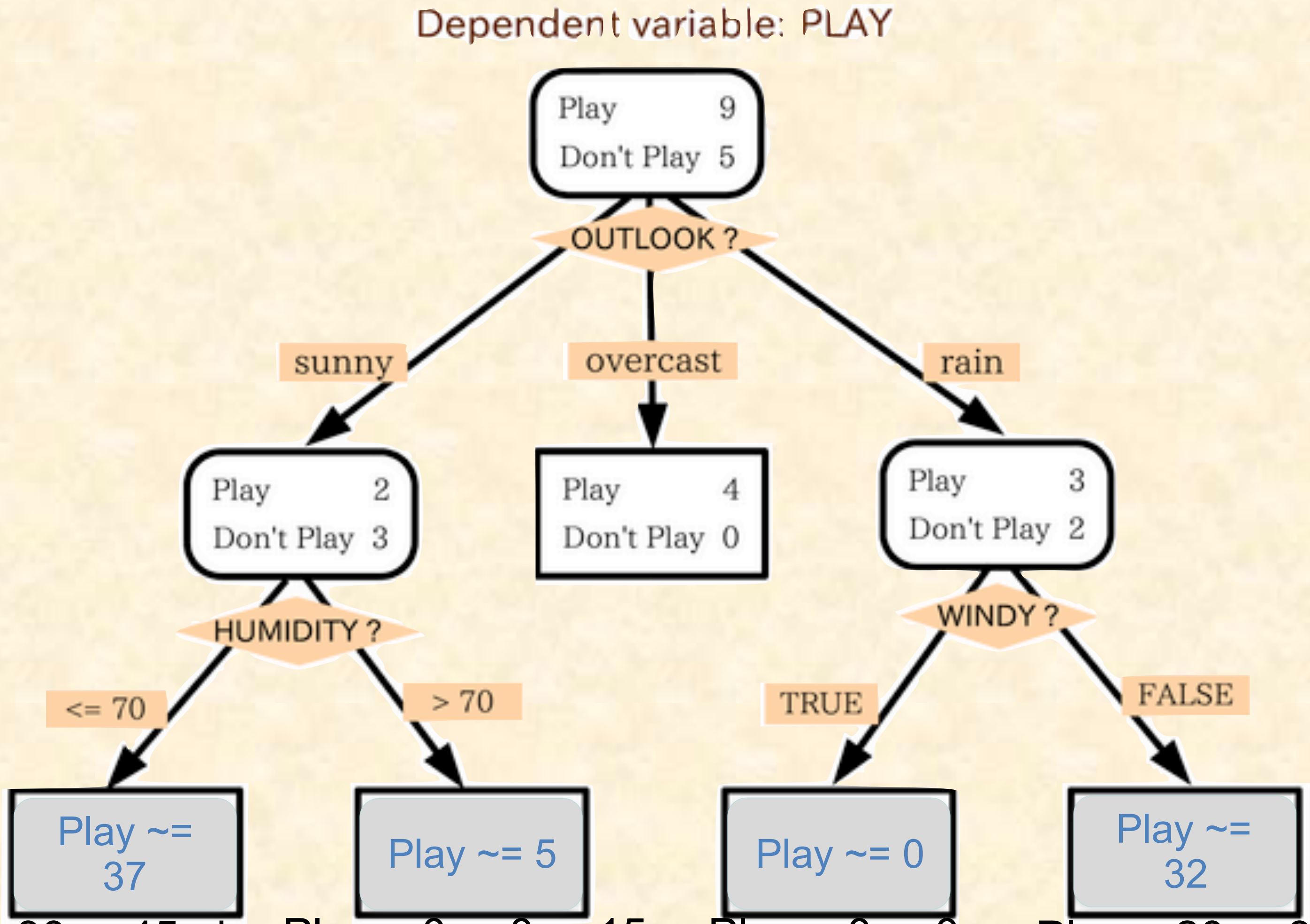


(c) Menu Desc: descriptors	
old time favorite	-0.112
fashioned	-0.034
...	
artisanal	0.064
raised	0.066
heirloom	0.083
wild	0.084
hormone	0.085
farmed	0.099
hand picked	0.101
wild caught	0.116
farmhouse	0.133

A decision tree: classification



A Regression tree



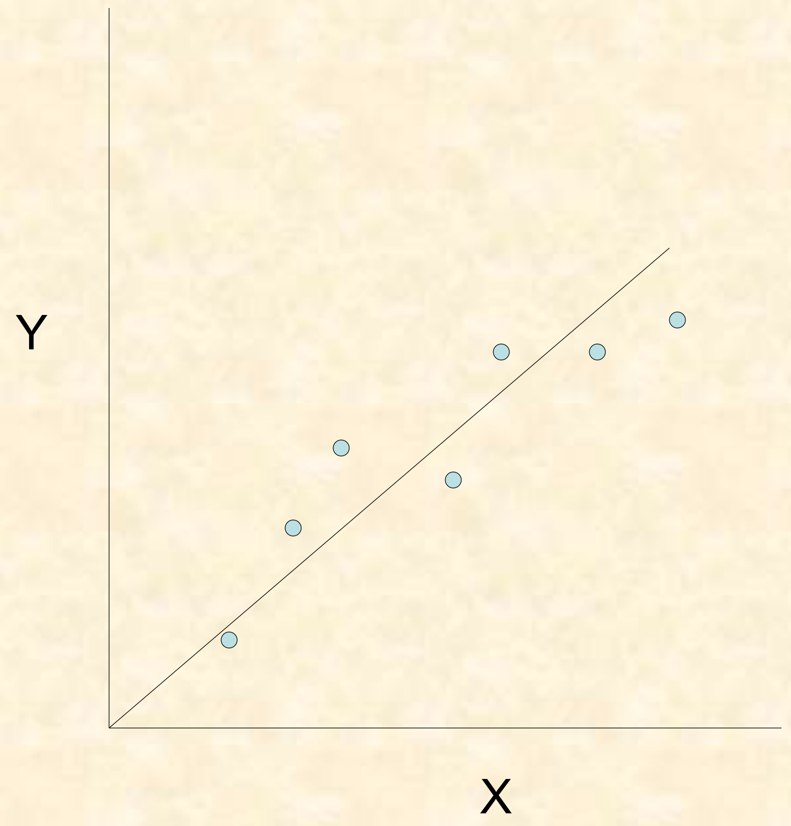
Play = 30m, 45min Play = 0m, 0m, 15m Play = 0m, 0m Play = 20m, 30m, 45m,

Least Mean Squares

Regression for LMS as optimization

Linear regression

- Given an input x we would like to compute an output y
- For example:
  - Predict height from age
  - Predict Google’s price from Yahoo’s price
  - Predict distance from wall from sensors



Linear regression

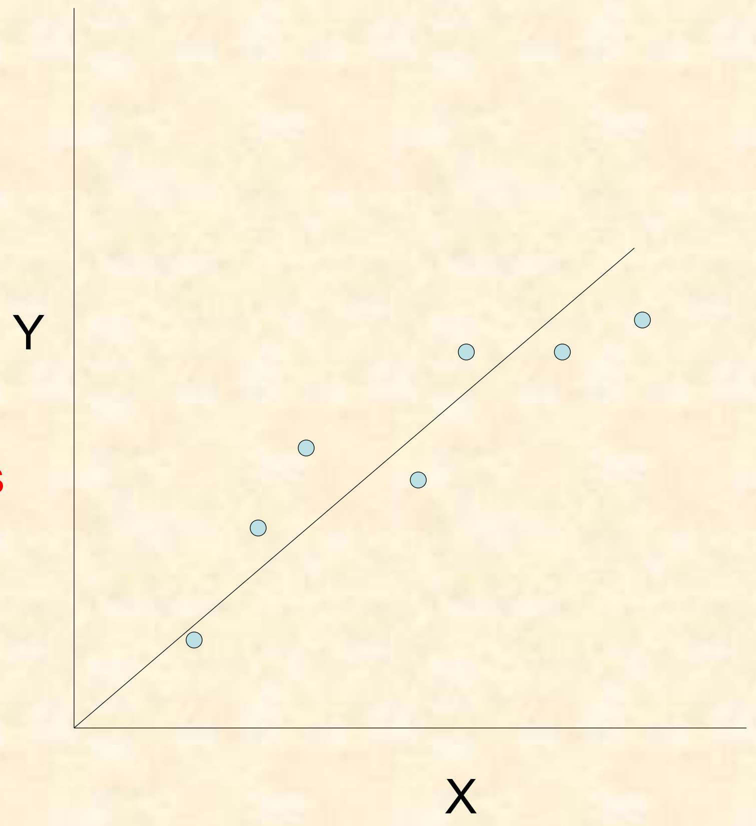
- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

What we are trying to predict

Observed values

$$y = wx + \varepsilon$$

where w is a parameter and ε represents measurement or other noise



Linear regression

- Our goal is to estimate *w* from a training data of $<x_i,y_i>$ pairs
- Optimization goal: minimize squared error (least squares):

$$\arg\min_w \sum_i (y_i - wx_i)^2$$

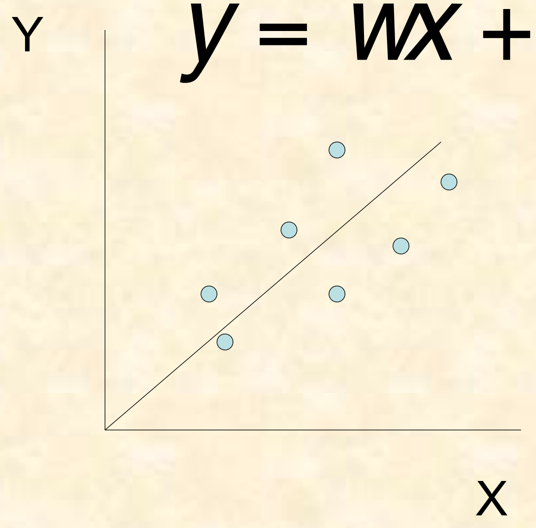
- Why least squares?

  - minimizes squared distance between measurements and predicted line

  - has a nice probabilistic interpretation

  - the math is pretty

$$y = wx + \varepsilon$$



Solving linear regression

- To optimize:
- We just take the derivative w.r.t. to w ….

prediction

$$\frac{\partial}{\partial w}\sum_i (y_i - wx_i)^2 = 2\sum_i -x_i(y_i - wx_i)$$

Compare to logistic regression…

prediction

$$\frac{\partial}{\partial w^j}\log P(Y = y|X = \mathbf{x}, \mathbf{w}) = (y - p)x^j$$

Solving linear regression

- To optimize – closed form:
- We just take the derivative w.r.t. to w and set to 0:

$$\frac{\partial}{\partial w}\sum_i (y_i - wx_i)^2 = 2\sum_i -x_i(y_i - wx_i) \Rightarrow$$

$$2\sum_i x_i(y_i - wx_i) = 0 \quad \Rightarrow \quad 2\sum_i x_i y_i - 2\sum_i wx_i x_i = 0$$

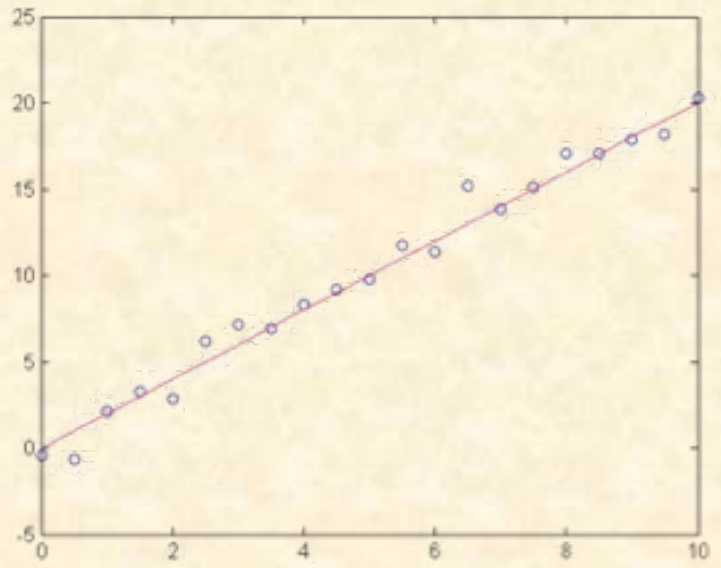
$$\sum_i x_i y_i = \sum_i wx_i^2 \quad \Rightarrow$$

$$w = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$$

covar(X,Y)/var(X)
if mean(X)=mean(Y)=0

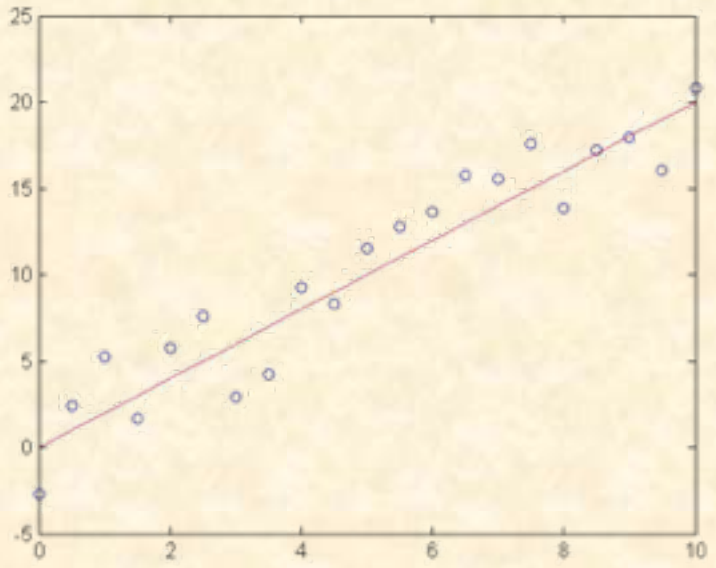
Regression example

- Generated: w=2
- Recovered: w=2.03
- Noise: std=1



Regression example

- Generated: w=2
- Recovered: w=2.05
- Noise: std=2

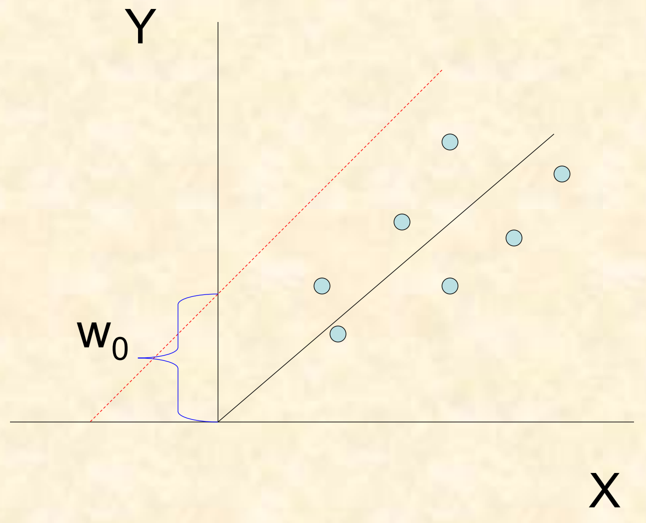


Bias term

- So far we assumed that the line passes through the origin
- What if the line does not?
- No problem, simply change the model to

  $y = w_0 + w_1 x + \varepsilon$

- Can use least squares to determine w_0 , w_1



$$w_0 = \frac{\sum_i y_i - w_1 x_i}{n} \qquad w_1 = \frac{\sum_i x_i (y_i - w_0)}{\sum_i x_i^2}$$

Multivariate regression

- What if we have several inputs?
  - Stock prices for Yahoo, Microsoft and Ebay for the Google prediction task
- This becomes a multivariate regression problem
- Again, its easy to model:

$$y = w_0 + w_1 x_1 + \dots + w_k x_k + \varepsilon$$

Google's stock price

Yahoo's stock price

Microsoft's stock price

Multivariate regression

- What if we have several inputs?
  - Stock prices for Yahoo, Microsoft and Ebay for the Google prediction task
- This becomes a multivariate regression problem
- Again, its easy to model:

$$y = w_0 + w_1x_1 + \ldots + w_kx_k + \varepsilon$$

Non-Linear basis function

- So far we only used the observed values $x_1, x_2, \ldots$
- However, linear regression can be applied in the same way to **functions** of these values
  - Eg: to add a term w x_1x_2 add a new variable $z=x_1x_2$ so each example becomes: x_1, x_2, …. z
- As long as these functions can be directly computed from the observed values the parameters are still linear in the data and the problem remains a multi-variate linear regression problem

$$y = w_0 + w_1 x_1^2 + \boxed{?} + w_k x_k^2 + \varepsilon$$

Non-Linear basis function

- How can we use this to add an intercept term?

Add a new “variable” z=1 and weight w_0

Non-linear basis functions

- What type of functions can we use?
- A few common examples:

- Polynomial: $\phi_j(x) = x^j$ for j=0 … n

- Gaussian: $$\phi_j(x) = \frac{(x-\mu_j)}{2\sigma_j^2}$$

- Sigmoid: $$\phi_j(x) = \frac{1}{1+\exp(-s_j x)}$$

- Logs: $$\phi_j(x) = \log(x+1)$$

Any function of the input values can be used. The solution for the parameters of the regression remains the same.

General linear regression problem

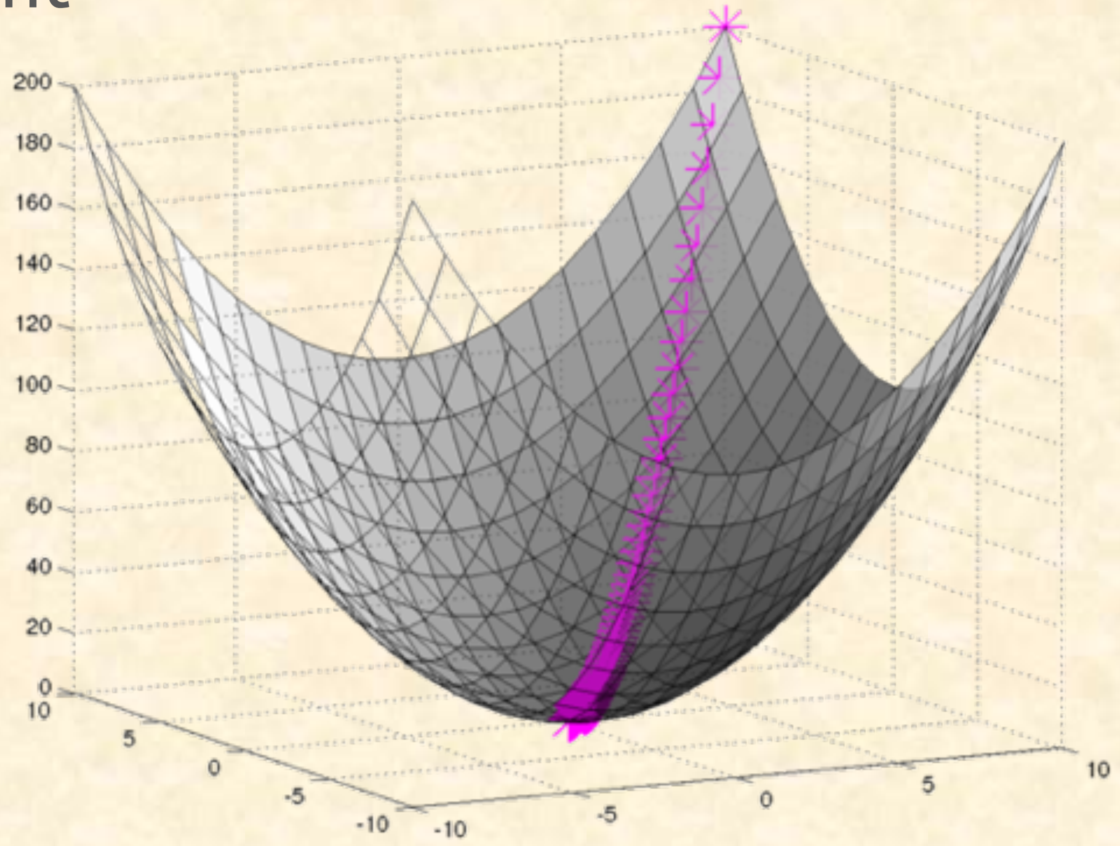
- Using our new notations for the basis function linear regression can be written as

$$y = \sum_{j=0}^{n} w_j \phi_j(x)$$

- Where $\phi_j(\mathbf{x})$ can be either x_j for multivariate regression or one of the non-linear basis functions we defined
- … and $\phi_0(\mathbf{x})=1$ for the intercept term

Learning/Optimizing Multivariate Least Squares

Approach 1: Gradient Descent



Gradient Descent for Linear Regression

Goal: minimize the following loss function:

predict with: $\hat{y}^i = \sum_j^n w_j \phi_j(\mathbf{x}^i)$

$$J_{\mathbf{X},\mathbf{y}}(\mathbf{w}) = \underbrace{\sum_i}_{\text{sum over } n \text{ examples}} \left(y^i - \hat{y}^i\right)^2 = \sum_i \left(y^i - \underbrace{\sum_j}_{\text{sum over } k+1 \text{ basis vectors}} w_j \phi_j(\mathbf{x}^i) \right)^2$$

Gradient Descent for Linear Regression

Goal: minimize the following loss function:

$$\text{predict with: } \hat{y}^i = \sum_j^n w_j \phi_j(\mathbf{x}^i)$$

$$J_{\mathbf{x},\mathbf{y}}(\mathbf{w}) = \sum_i \left(y^i - \hat{y}^i\right)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i)\right)^2$$

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = \frac{\partial}{\partial w_j} \sum_i \left(y^i - \hat{y}^i\right)^2$$

$$= 2\sum_i \left(y^i - \hat{y}^i\right) \frac{\partial}{\partial w_j} \hat{y}^i$$

$$= 2\sum_i \left(y^i - \hat{y}^i\right) \frac{\partial}{\partial w_j} \sum_j w_j \phi_j(\mathbf{x}^i)$$

$$= 2\sum_i \left(y^i - \hat{y}^i\right) \phi_j(\mathbf{x}^i)$$

Gradient Descent for Linear Regression

Learning algorithm:

- Initialize weights **w=0**
- For t=1,… until convergence:
  - Predict for each example $\boldsymbol{x}^i$ using **w:**

$$\hat{y}^i = \sum_{j=0}^{k} w_j \phi_j(\mathbf{x}^i)$$

  - Compute gradient of loss:

  This is a vector **g**

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = 2\sum_i \left(y^i - \hat{y}^i\right)\phi_j(\mathbf{x}^i)$$

  - Update: **w = w** – λ**g**

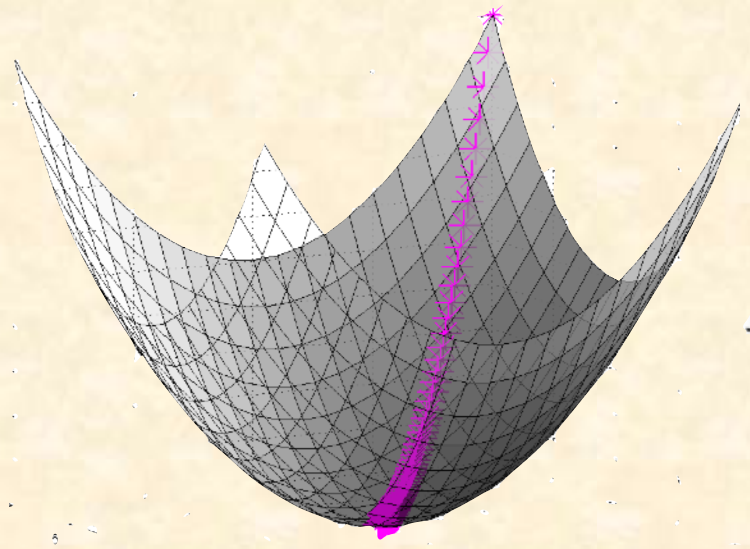
  λ is the learning rate.

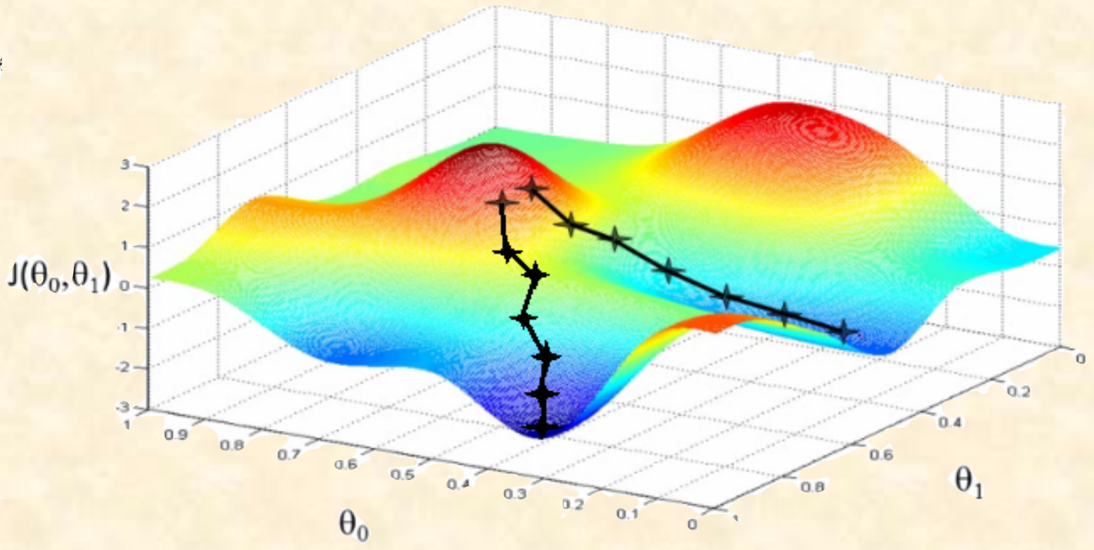
Gradient Descent for Linear Regression

- We can use any of the tricks we used for logistic regression:
  - stochastic gradient descent (if the data is too big to put in memory)
  - regularization
  - …

Linear regression is a *convex* optimization problem

gradient descent will reach a *global* optimum





proof: differentiate again to get the second derivative

Multivariate Least Squares

Approach 2: Matrix Inversion

Goal: minimize the following loss function:

predict with: $\hat{y}^i = \sum_j^n w_j \phi_j(\mathbf{x}^i)$

$$J_{\mathbf{x},\mathbf{y}}(\mathbf{w}) = \sum_i \left(y^i - \hat{y}^i\right)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i)\right)^2$$

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = 2\sum_i \left(y^i - \hat{y}^i\right)\phi_j(\mathbf{x}^i)$$

Multivariate Least Squares

Approach 2: Matrix Inversion

Goal: minimize the following loss function:

$$\text{predict with}: \hat{y}^i = \sum_j^n w_j \phi_j(\mathbf{x}^i)$$

$$J_{\mathbf{x},\mathbf{y}}(\mathbf{w}) = \sum_i \left(y^i - \hat{y}^i\right)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i)\right)^2$$

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = 2\sum_i \left(y^i - \hat{y}^i\right)\phi_j(\mathbf{x}^i)$$

Notation:

$$\Phi = \left.\overbrace{\begin{pmatrix} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \boxed{?} & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \boxed{?} & \phi_k(\mathbf{x}^2) \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \boxed{?} & \phi_k(\mathbf{x}^n) \end{pmatrix}}^{\textit{k+1 basis vectors}}\right\} \textit{n examples}$$

Multivariate Least Squares

Approach 2: Matrix Inversion

Goal: minimize the following loss function:

$$\text{predict with: } \hat{y}^i = \sum_j^n w_j \phi_j(\mathbf{x}^i)$$

$$J_{\mathbf{x},\mathbf{y}}(\mathbf{w}) = \sum_i \left(y^i - \hat{y}^i\right)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i)\right)^2$$

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = 2 \sum_i \left(y^i - \hat{y}^i\right) \phi_j(\mathbf{x}^i)$$

Notation:

$$\Phi = \overbrace{\left(\begin{array}{cccc} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \boxed{?} & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \boxed{?} & \phi_k(\mathbf{x}^2) \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \boxed{?} & \phi_k(\mathbf{x}^n) \end{array}\right)}^{\textit{k+1 basis vectors}} \left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} \textit{n examples}$$

$$\mathbf{w} = \begin{pmatrix} w_0 \\ .. \\ w_k \end{pmatrix}$$

Multivariate Least Squares

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \boxed{?} & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \boxed{?} & \phi_k(\mathbf{x}^2) \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \boxed{?} & \phi_k(\mathbf{x}^n) \end{pmatrix} = \begin{pmatrix} \phi^1 \\ \dots \\ \dots \\ \phi^n \end{pmatrix}$$

$$\left(\begin{array}{c} \dfrac{\partial}{\partial w_0} J(\mathbf{w}) = 2\sum_i \left(y^i - \hat{y}^i\right)\phi_0(\mathbf{x}^i) \\ \dots \\ \dfrac{\partial}{\partial w_k} J(\mathbf{w}) = 2\sum_i \left(y^i - \hat{y}^i\right)\phi_k(\mathbf{x}^i) \end{array} \right)$$

notation: $\phi_j^i \equiv \phi_j(\mathbf{x}^i)$

Multivariate Least Squares

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \boxed{?} & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \boxed{?} & \phi_k(\mathbf{x}^2) \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \boxed{?} & \phi_k(\mathbf{x}^n) \end{pmatrix} = \begin{pmatrix} \phi^1 \\ \dots \\ \dots \\ \phi^n \end{pmatrix}$$

$$\left(\begin{array}{c} \dfrac{\partial}{\partial w_0} J(\mathbf{w}) = 2\sum_i \left(y^i \phi_1^i - \hat{y}^i \phi_1^i \right) \\ \dots \\ \dfrac{\partial}{\partial w_k} J(\mathbf{w}) = 2\sum_i \left(y^i \phi_k^i - \hat{y}^i \phi_k^i \right) \end{array} \right)$$

$$\text{recall } \hat{y}^i = \sum_j^n w_j \phi_j^i$$

$$= \phi^{i\,\boxed{?}} \mathbf{w}$$

Multivariate Least Squares

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \boxed{?} & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \boxed{?} & \phi_k(\mathbf{x}^2) \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \boxed{?} & \phi_k(\mathbf{x}^n) \end{pmatrix} = \begin{pmatrix} \phi^1 \\ \dots \\ \dots \\ \phi^n \end{pmatrix}$$

$$\left(\begin{array}{c} \frac{\partial}{\partial w_0} J(\mathbf{w}) = 2\sum_i \left(y^i \phi_0^i - \phi^i \mathbf{w}\, \phi_0^i \right) \\ \dots \\ \frac{\partial}{\partial w_k} J(\mathbf{w}) = 2\sum_i \left(y^i \phi_k^i - \phi^i \mathbf{w}\, \phi_k^i \right) \end{array} \right)$$

$$= 2\Phi^T \mathbf{y} - 2\Phi^T \Phi \mathbf{w}$$

Multivariate Least Squares

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \boxed{?} & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \boxed{?} & \phi_k(\mathbf{x}^2) \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \boxed{?} & \phi_k(\mathbf{x}^n) \end{pmatrix} = \begin{pmatrix} \phi^1 \\ \cdots \\ \cdots \\ \phi^n \end{pmatrix}$$

$$\begin{pmatrix} \dfrac{\partial}{\partial w_0} J(\mathbf{w}) = 2\sum_i \left(\phi_0^i y^i - \phi_0^i \phi^i \mathbf{w}\right) \\ \cdots \\ \dfrac{\partial}{\partial w_k} J(\mathbf{w}) = 2\sum_i \left(\phi_k^i y^i - \phi_k^i \phi^i \mathbf{w}\right) \end{pmatrix} = 2\Phi^T \mathbf{y} - \ldots$$

Multivariate Least Squares

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \boxed{?} & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \boxed{?} & \phi_k(\mathbf{x}^2) \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \boxed{?} & \phi_k(\mathbf{x}^n) \end{pmatrix} = \begin{pmatrix} \phi^1 \\ \dots \\ \dots \\ \phi^n \end{pmatrix}$$

$$\begin{pmatrix} \dfrac{\partial}{\partial w_0} J(\mathbf{w}) = 2\sum_i \left(\phi_0^i y^i - \phi_0^i \phi^i \mathbf{w}\right) \\ \dots \\ \dfrac{\partial}{\partial w_k} J(\mathbf{w}) = 2\sum_i \left(\phi_k^i y^i - \phi_k^i \phi^i \mathbf{w}\right) \end{pmatrix} = \dots - 2\Phi^T \Phi \mathbf{w}$$

Multivariate Least Squares

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \boxed{?} & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \boxed{?} & \phi_k(\mathbf{x}^2) \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \boxed{?} & \phi_k(\mathbf{x}^n) \end{pmatrix} = \begin{pmatrix} \phi^1 \\ \dots \\ \dots \\ \phi^n \end{pmatrix}$$

$$\begin{pmatrix} \dfrac{\partial}{\partial w_0} J(\mathbf{w}) = 2\sum_i \left(\phi_0^i y^i - \phi_0^i \phi^i \mathbf{w}\right) \\ \dots \\ \dfrac{\partial}{\partial w_k} J(\mathbf{w}) = 2\sum_i \left(\phi_k^i y^i - \phi_k^i \phi^i \mathbf{w}\right) \end{pmatrix} = \dots - 2\Phi^T \Phi \mathbf{w}$$

Multivariate Least Squares

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \boxed{?} & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \boxed{?} & \phi_k(\mathbf{x}^2) \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \boxed{?} & \phi_k(\mathbf{x}^n) \end{pmatrix} = \begin{pmatrix} \phi^1 \\ \dots \\ \dots \\ \phi^n \end{pmatrix}$$

$$\left(\begin{array}{c} \frac{\partial}{\partial w_0} J(\mathbf{w}) = 2\sum_i \left(y^i \phi_0^i - \phi^i \mathbf{w}\, \phi_0^i \right) \\ \dots \\ \frac{\partial}{\partial w_k} J(\mathbf{w}) = 2\sum_i \left(y^i \phi_k^i - \phi^i \mathbf{w}\, \phi_k^i \right) \end{array} \right)$$

$$= 2\Phi^T \mathbf{y} - 2\Phi^T \Phi \mathbf{w} = 0$$

$$\mathbf{w} = \left(\Phi^T \Phi \right)^{-1} \Phi^T \mathbf{y}$$

LMS for general linear regression problem

Deriving w we get: $\mathrm{w} = (\Phi^T\Phi)^{-1}\Phi^T\mathrm{y}$

$$J(\mathrm{w}) = \sum_i (y^i - \mathrm{w}^{\mathrm{T}}\phi(x^i))^2$$

k+1 entries vector

n by k+1 matrix

n entries vector

This solution is also known as ‘pseudo inverse’

Another reason to start with an objective function: you can see when two learning methods are the same!

LMS versus gradient descent

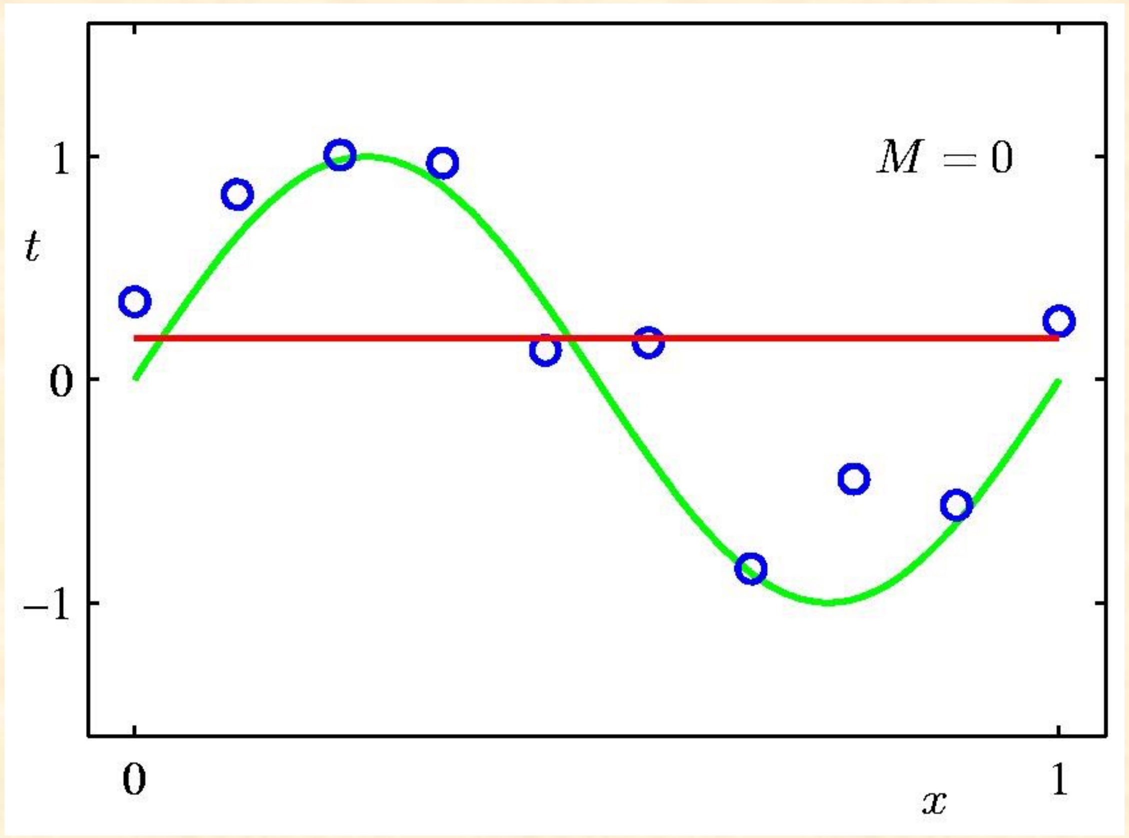
$$J(\mathrm{w}) = \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i))^2 \qquad \mathrm{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathrm{y}$$

LMS solution:
+ Very simple in Matlab or something similar
-Requires matrix inverse, which is expensive for a large matrix.

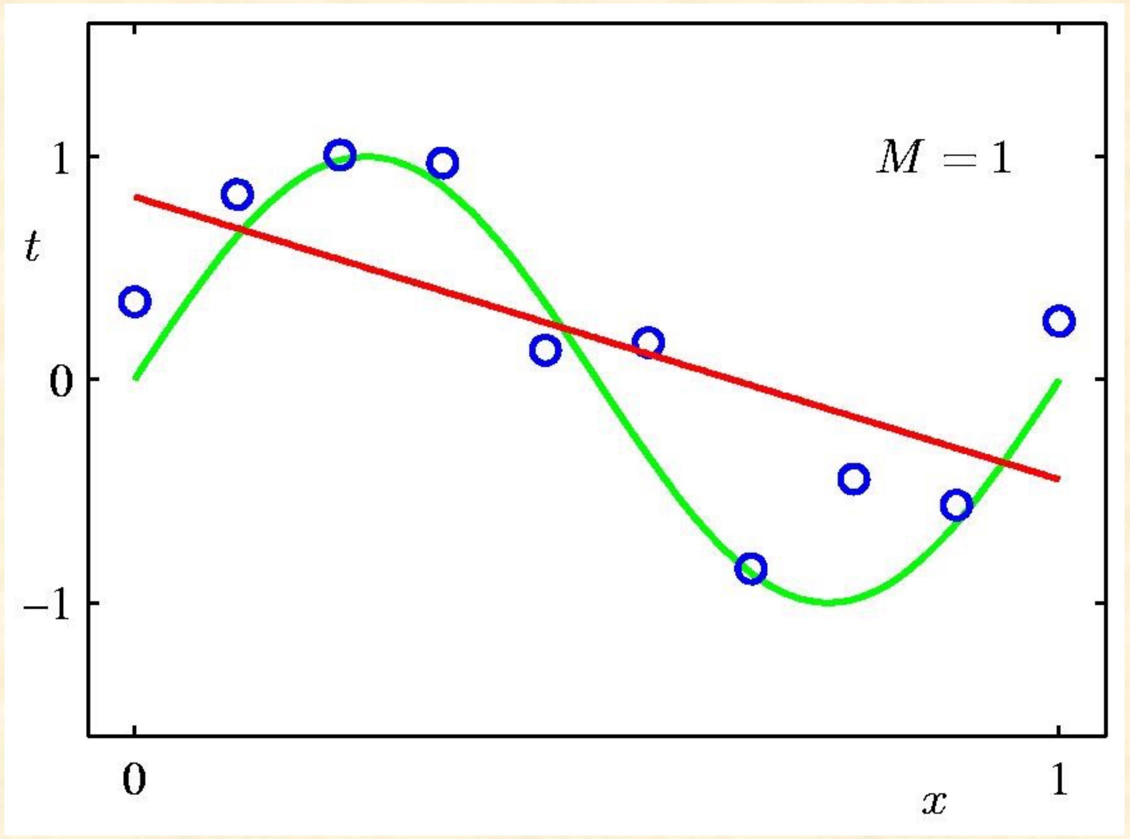
Gradient descent:
+ Fast for large matrices
+ Stochastic GD is very memory efficient
+ Easily extended to other cases
- Parameters to tweak (how to decide convergence? what is the learning rate? ….)

Overfitting in Regression

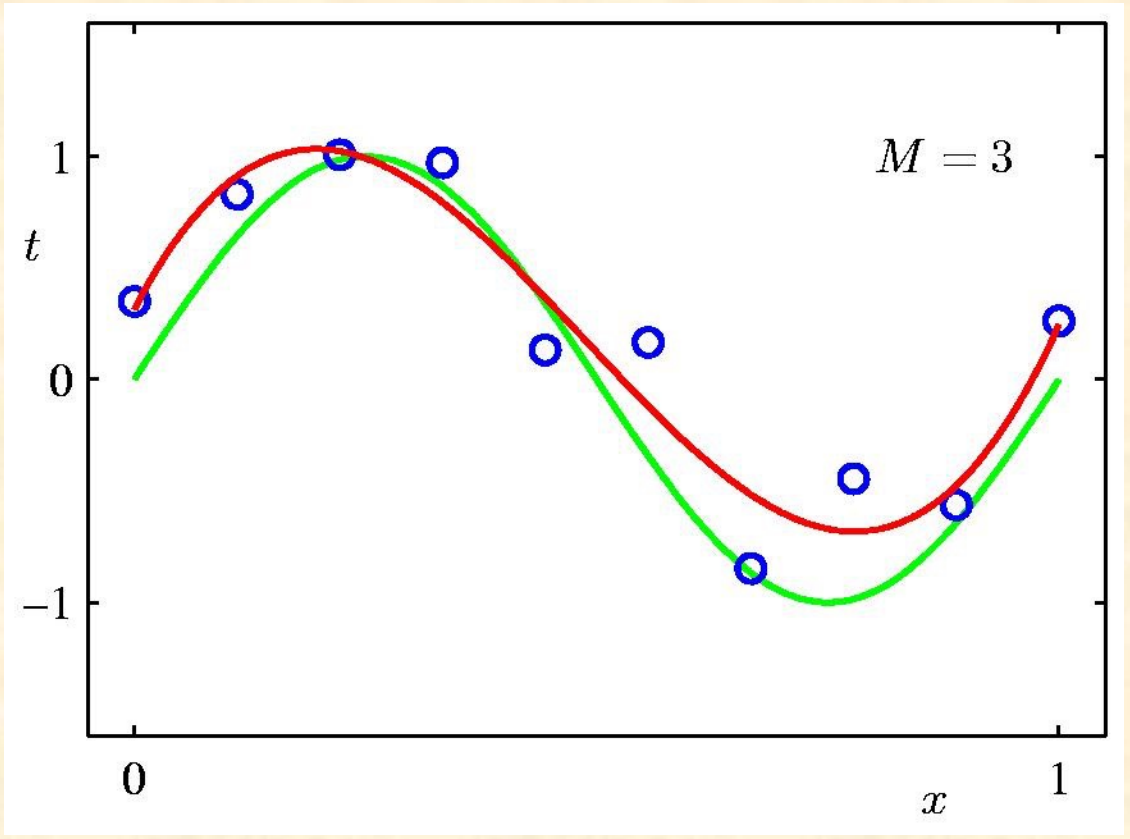
0th Order Polynomial



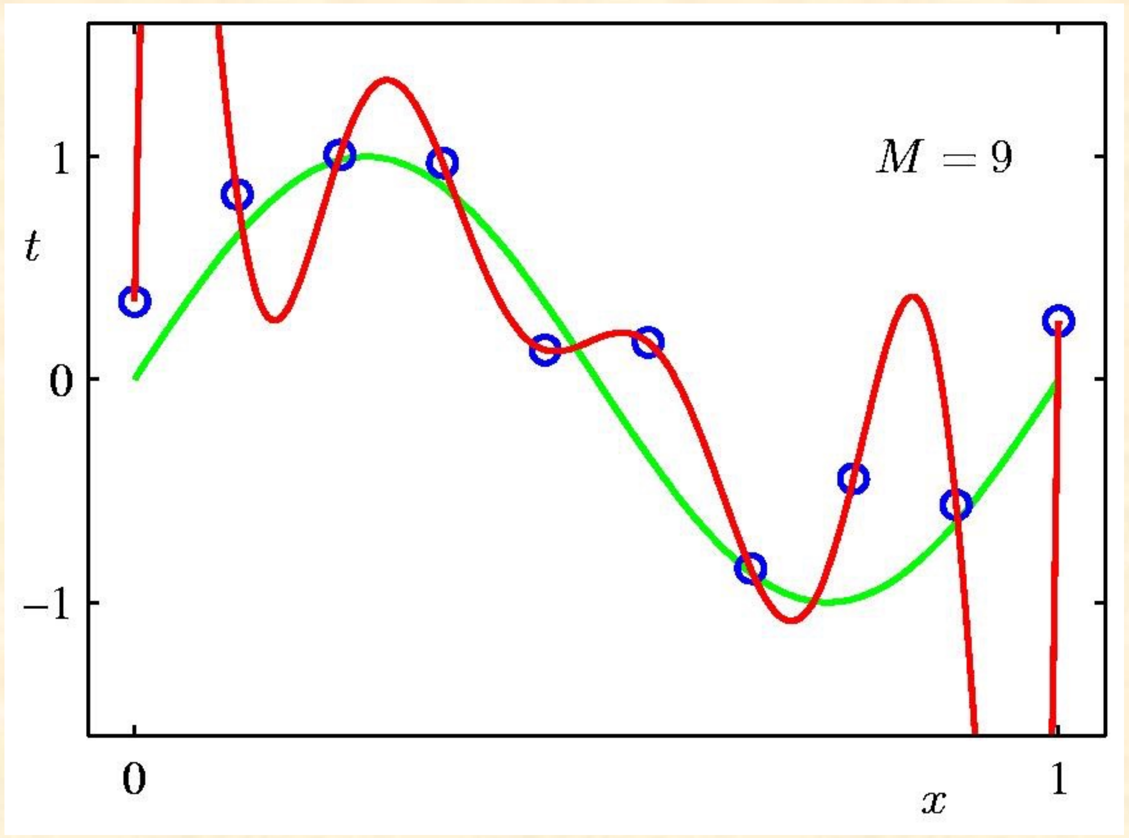
1st Order Polynomial



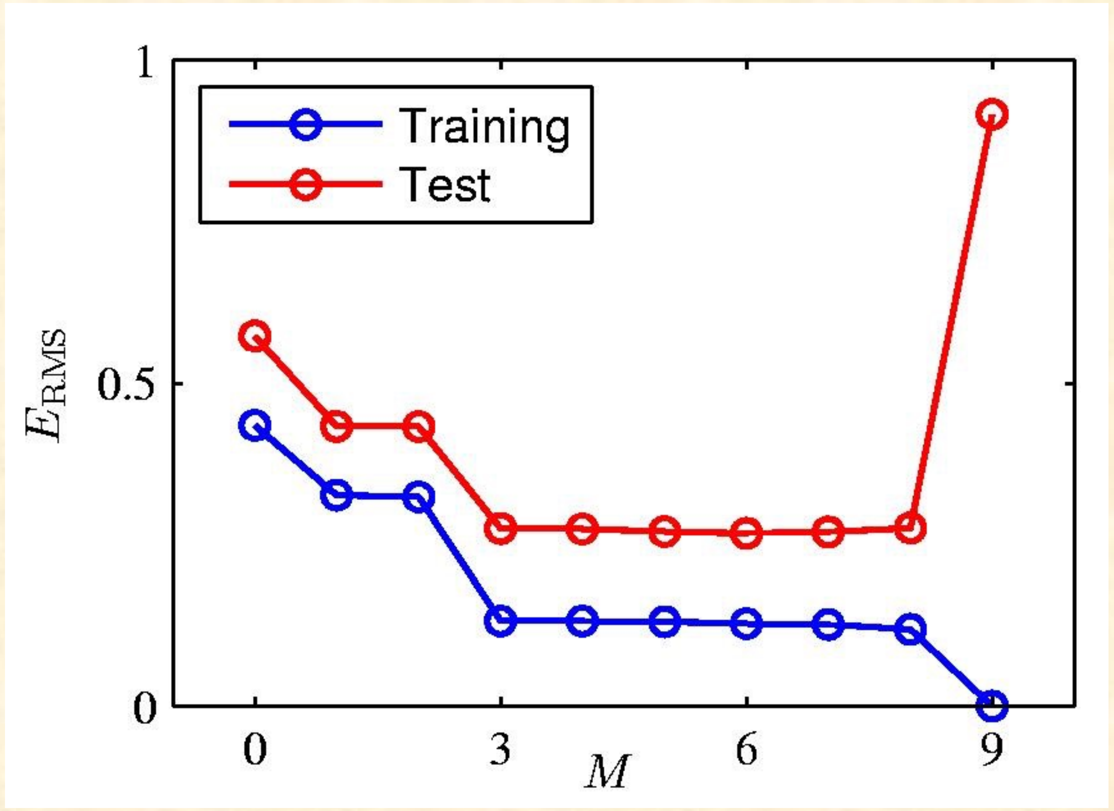
3rd Order Polynomial



9th Order Polynomial



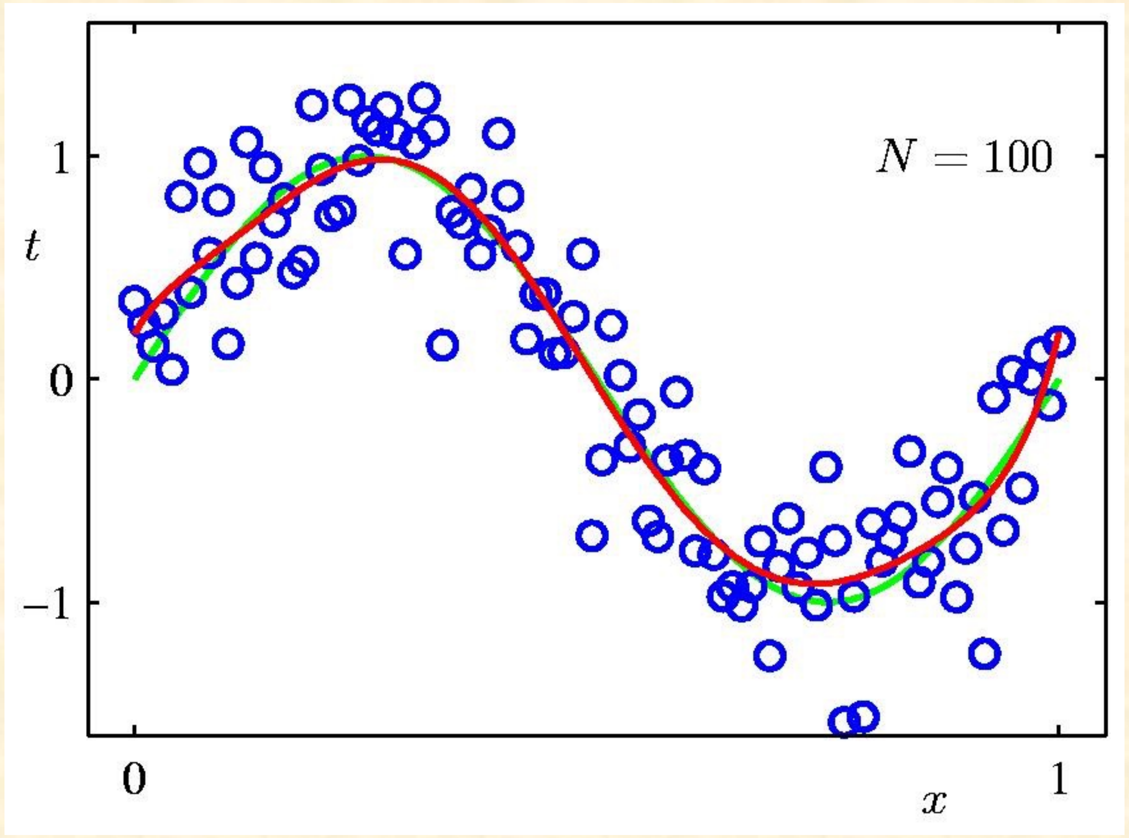
Over-fitting



Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^{\star})/N}$

Dataset Size

9th Order Polynomial



Cairo University

Thank You …