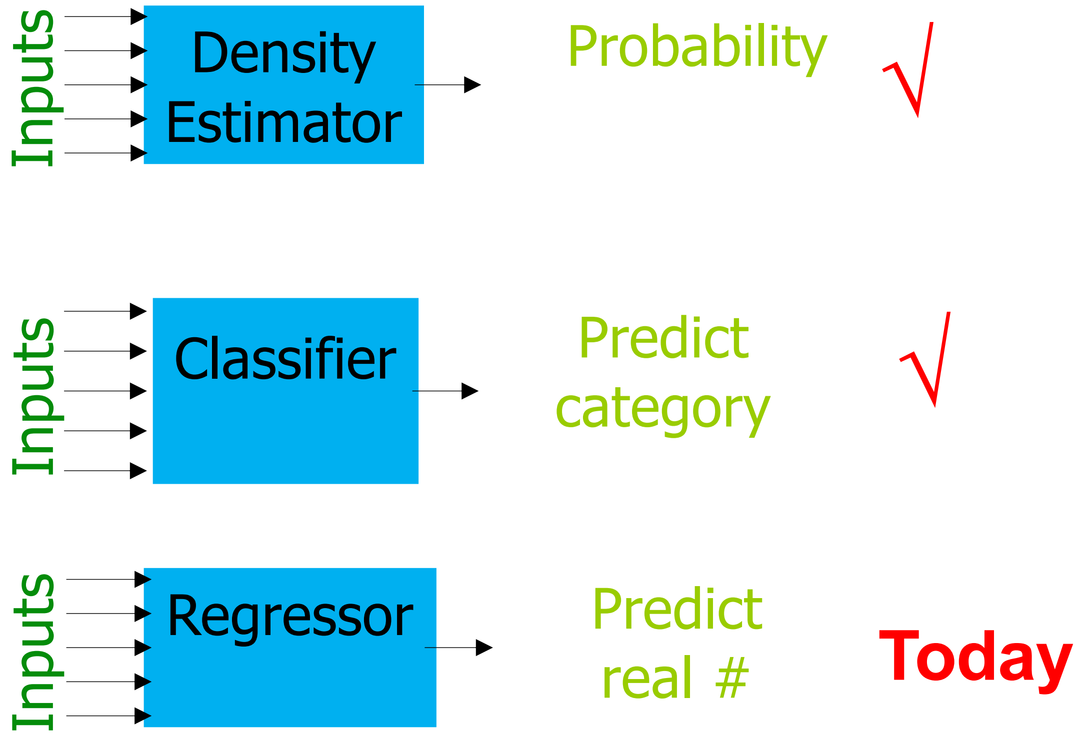
Data Mining

Regression Analysis

Outline

- Regression *vs* Classification



- Linear regression – another discriminative learning method
  - As optimization ➔ Gradient descent

Regression examples

Stock market

Weather prediction

Temperature
72° F

Predict the temperature at any given location

Prediction of menu price

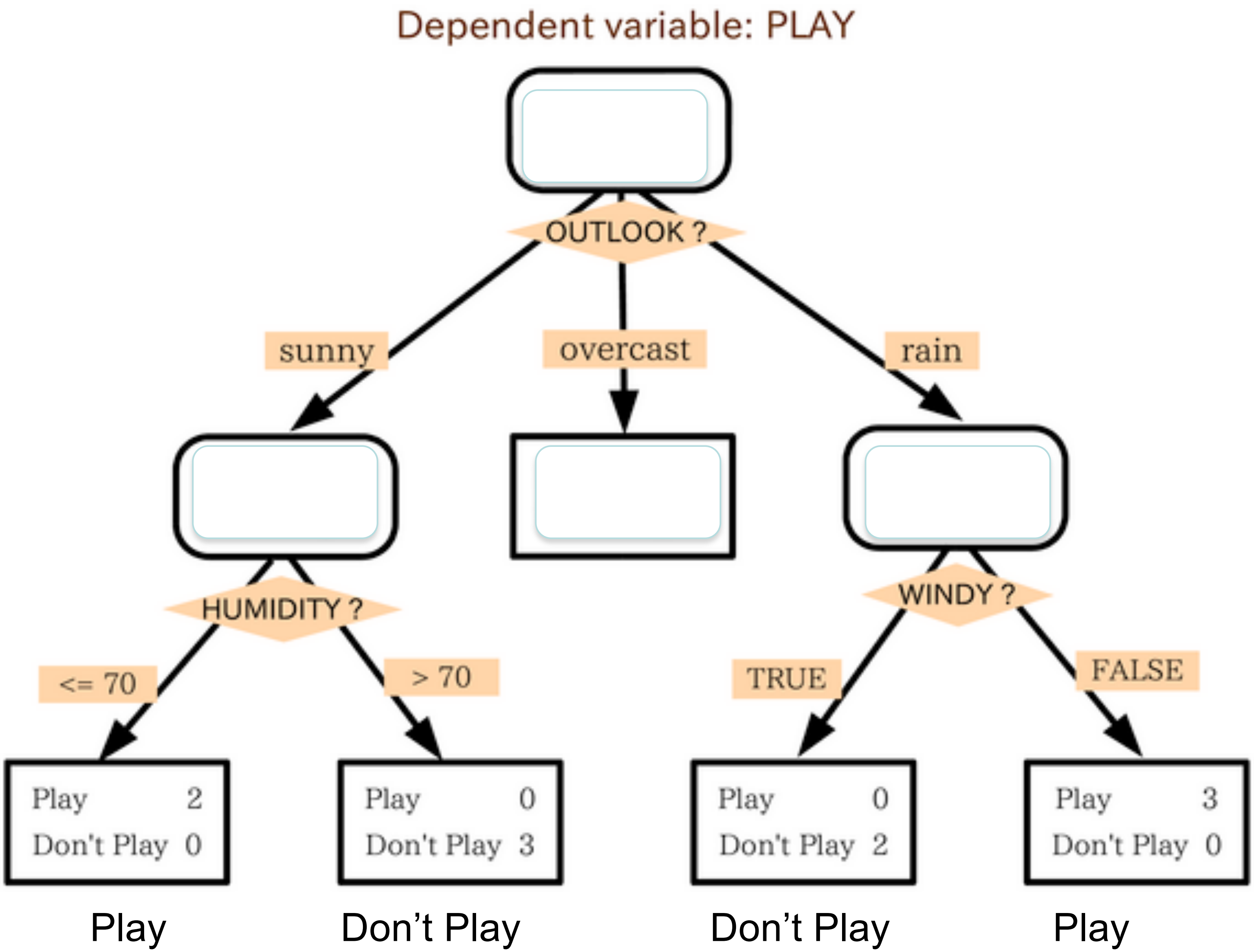
(a) METADATA: ambience	
dive-y	-0.015
intimate	-0.013
trendy	-0.012
casual	-0.005
romantic	-0.004
classy	-7e-6
touristy	0.058
upscale	0.099

(b) MENUDESC: cooking	
panfried	-0.094
chargrilled	-0.029
cooked	-0.012
boiled	-0.006
fried	-0.005
steamed	0.011
charbroiled	0.015
grilled	0.022
simmered	0.025
roasted	0.034
sauteed	0.034
broiled	0.053
seared	0.066
braised	0.068
stirfried	0.071
flamebroiled	0.106

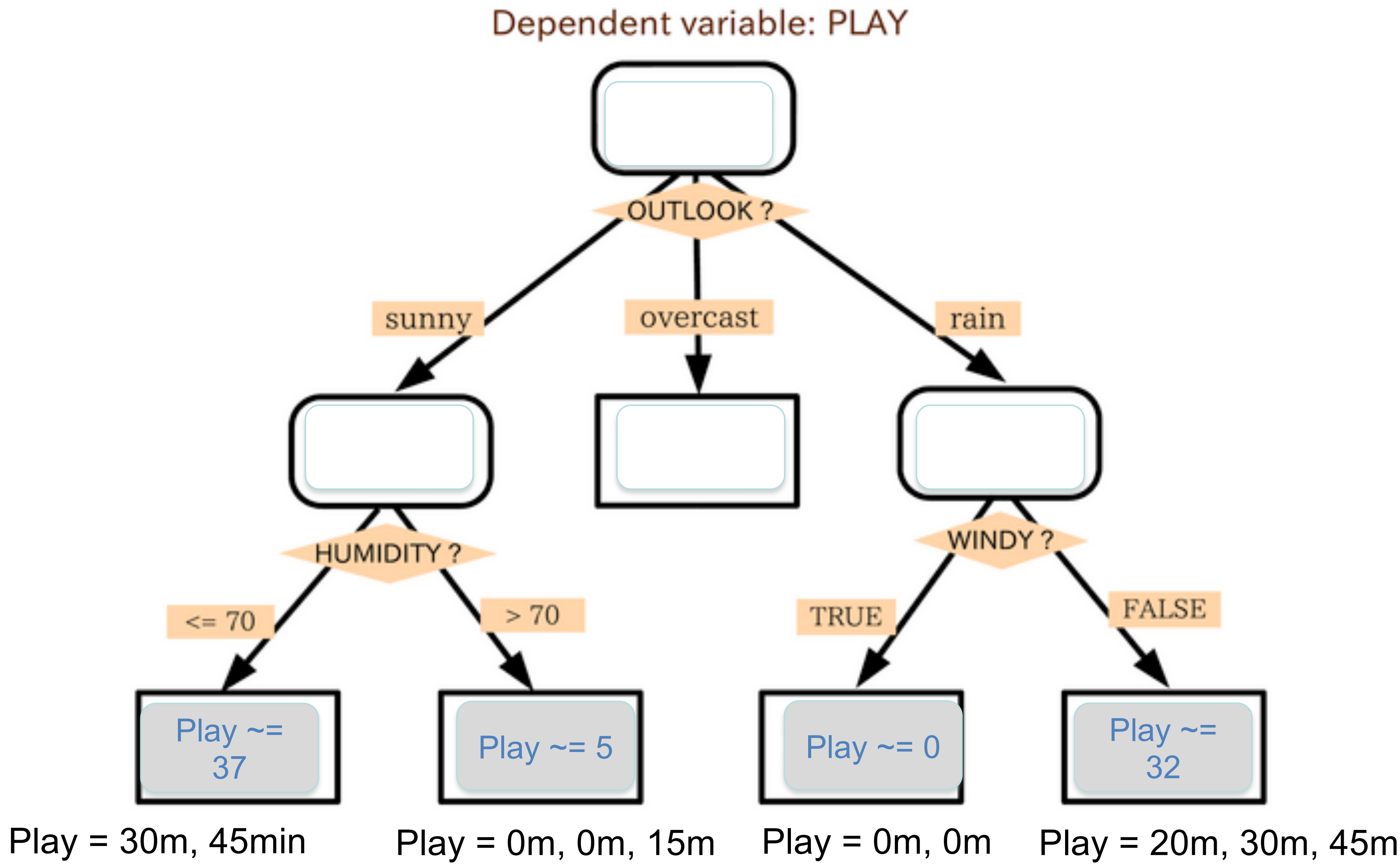
(c) MENUDESC: descriptors	
old time favorite	-0.112
fashioned	-0.034
line caught	-0.028
all natural	-0.028
traditional	-0.009
natural	3e-4
classic	0.002
free range	0.004
real	0.004
fresh	0.006
homemade	
authentic	
organic	

(d) MENUDESC: _ = "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

A decision tree: classification



A regression tree



Learning as optimization

- Two types of learners:

  1. Generative: make assumptions about how to generate data (given the class)
     - e.g., naïve Bayes

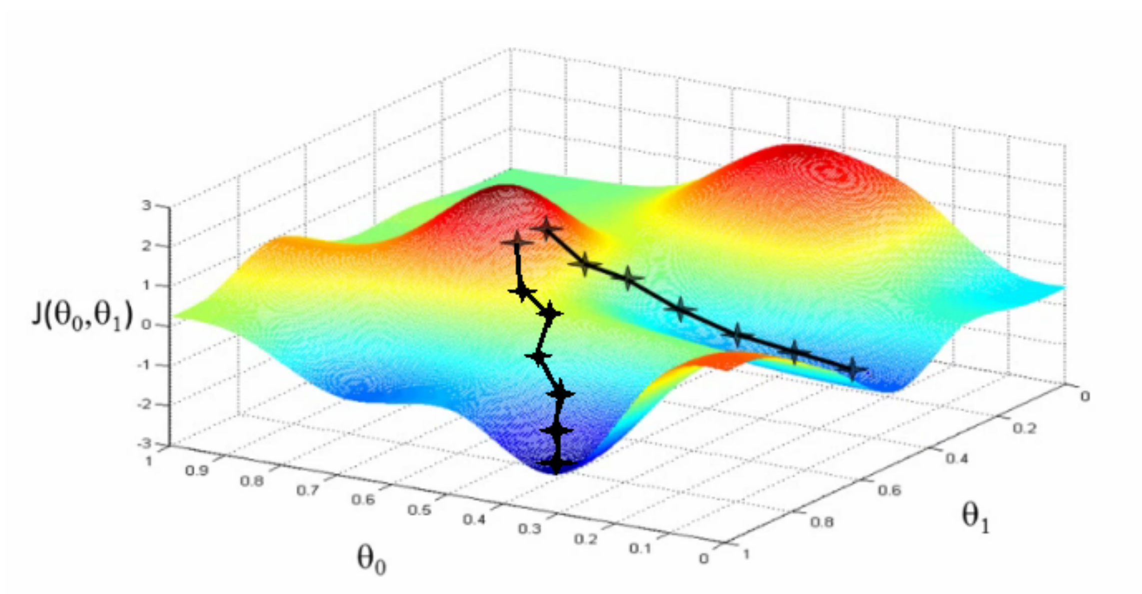
  2. Discriminative
     - directly estimate a decision rule/boundary
     - e.g., logistic regression

Today: another discriminative learner, but for regression tasks

Regression

Least Mean Squares

- Regression for LMS as optimization



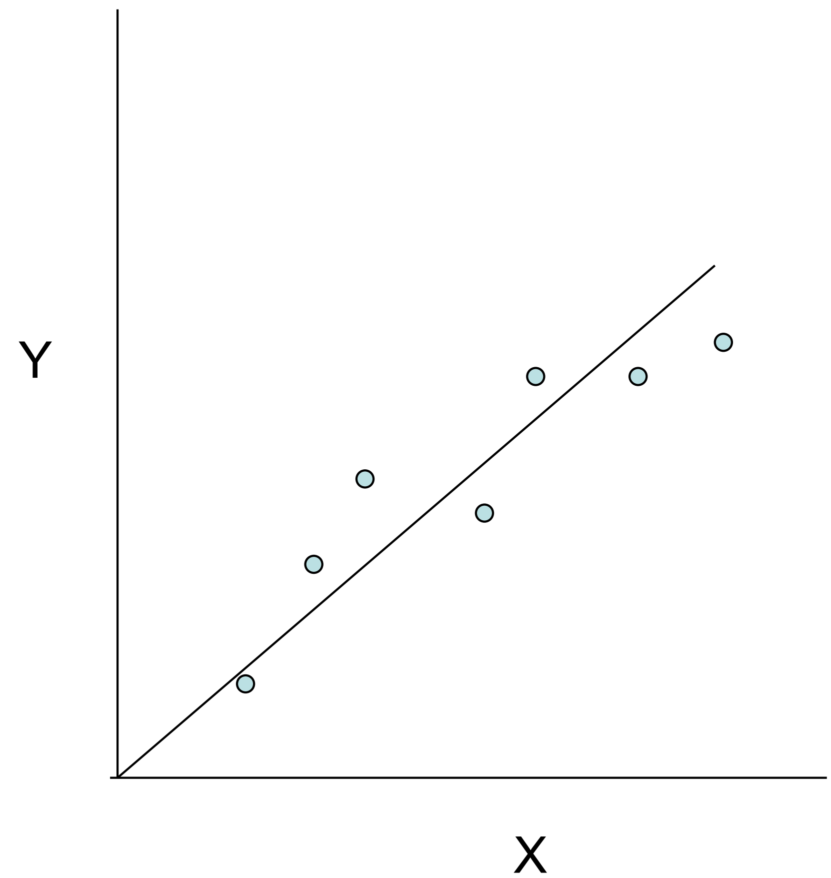
Linear regression

- 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

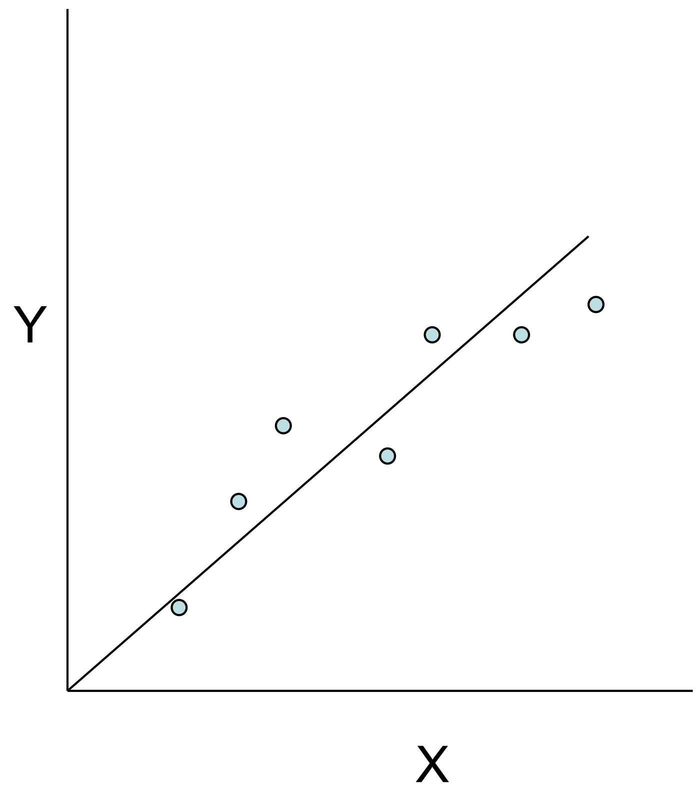
- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

What we are trying to predict

Observed values

$$y = wx + \varepsilon$$

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



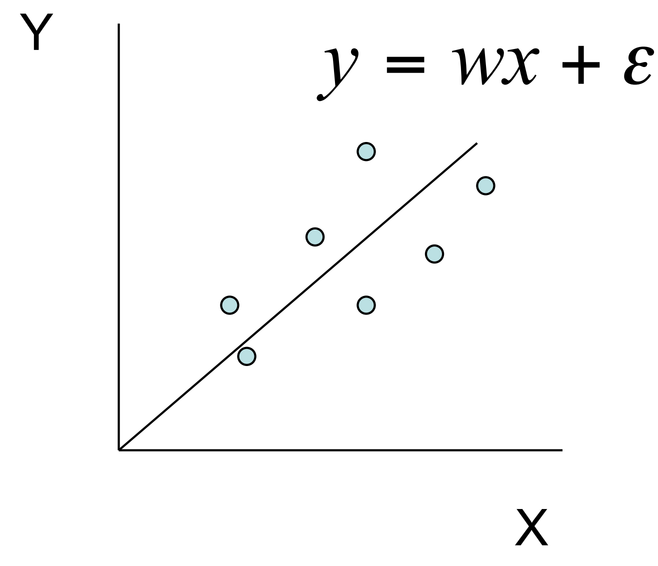
Linear regression

- 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



Solving linear regression

- To optimize:

We just take the derivative w.r.t. to w ….

prediction

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

Solving linear regression

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

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

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

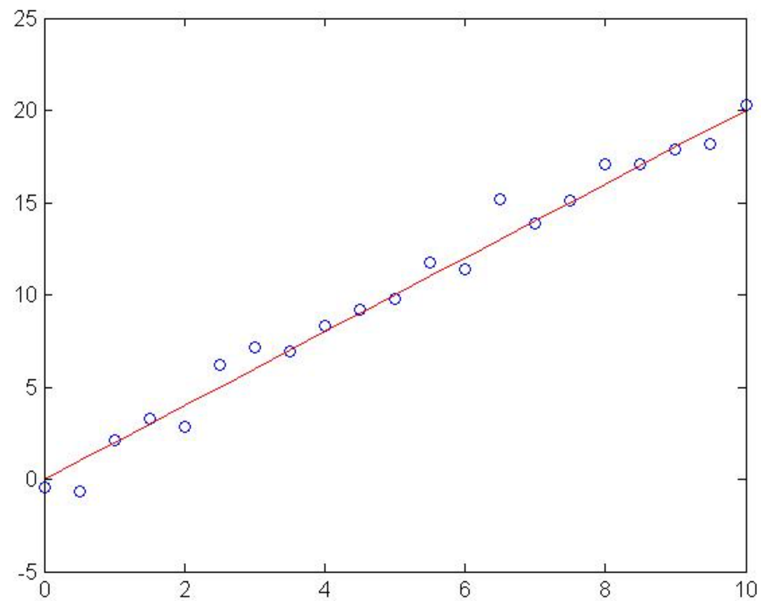
$$\sum_i x_i y_i = \sum_i w x_i^2 \quad \Rightarrow$$

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

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

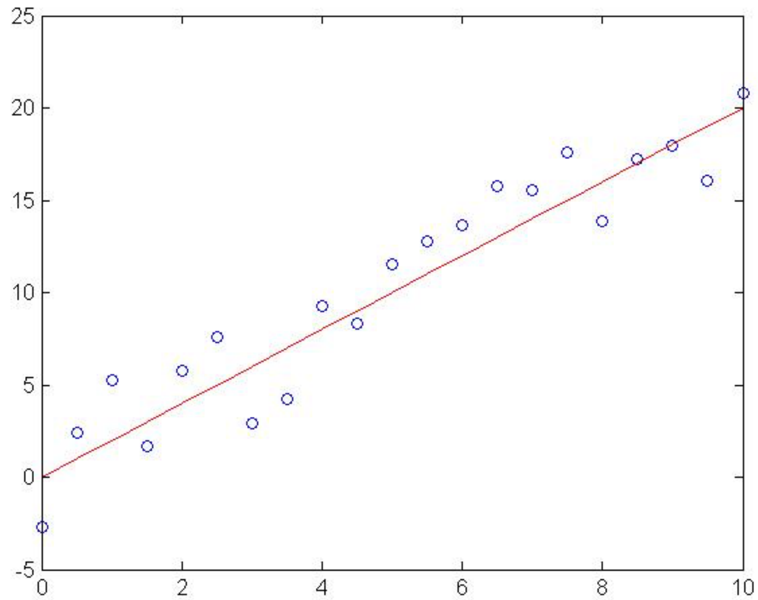
Regression example

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



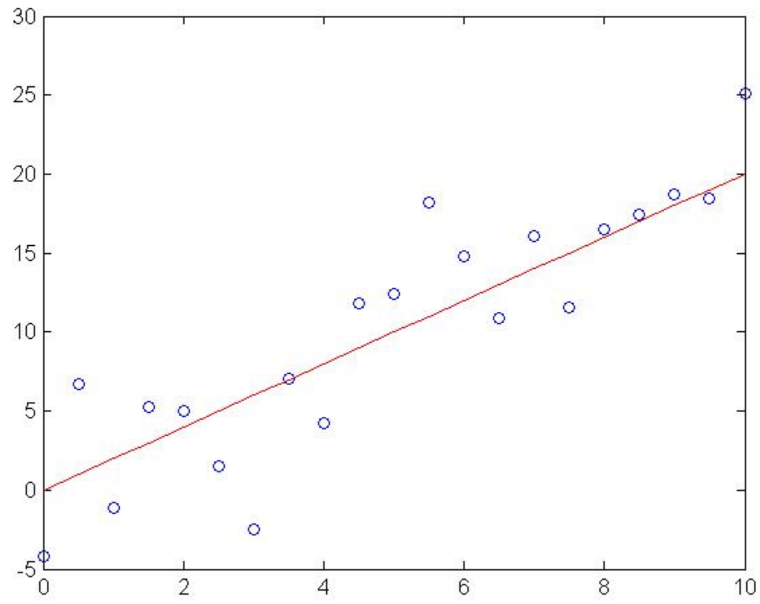
Regression example

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



Regression example

- Generated: w=2
- Recovered: w=2.08
- Noise: std=4

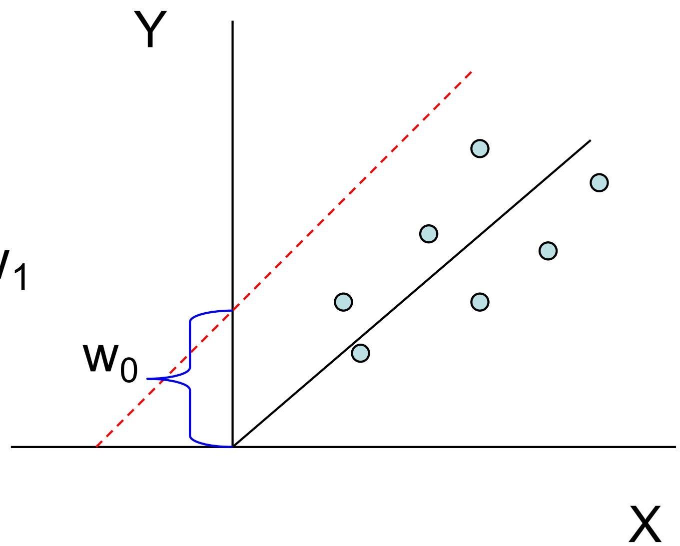


Bias term

- 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



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

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:

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

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:

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

Other functions of x

Not all functions can be approximated by a line/hyperplane…

$$y=10+3x_1^2-2x_2^2+\varepsilon$$

In some cases we would like to use polynomial or other terms based on the input data, are these still linear regression problems?

Yes. As long as the *coefficients* are linear the equation is still a linear regression problem!

Non-Linear basis function

- So far we only used the observed values x_1,x_2,…
- However, linear regression can be applied in the same way to **functions** of these values
  - Eg: to add a term w $x_1 x_2$ add a new variable $z=x_1 x_2$ 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

$$y = w_0 + w_1 x_1^2 + \ldots + w_k x_k^2 + \varepsilon$$

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

Gradient Descent Approach

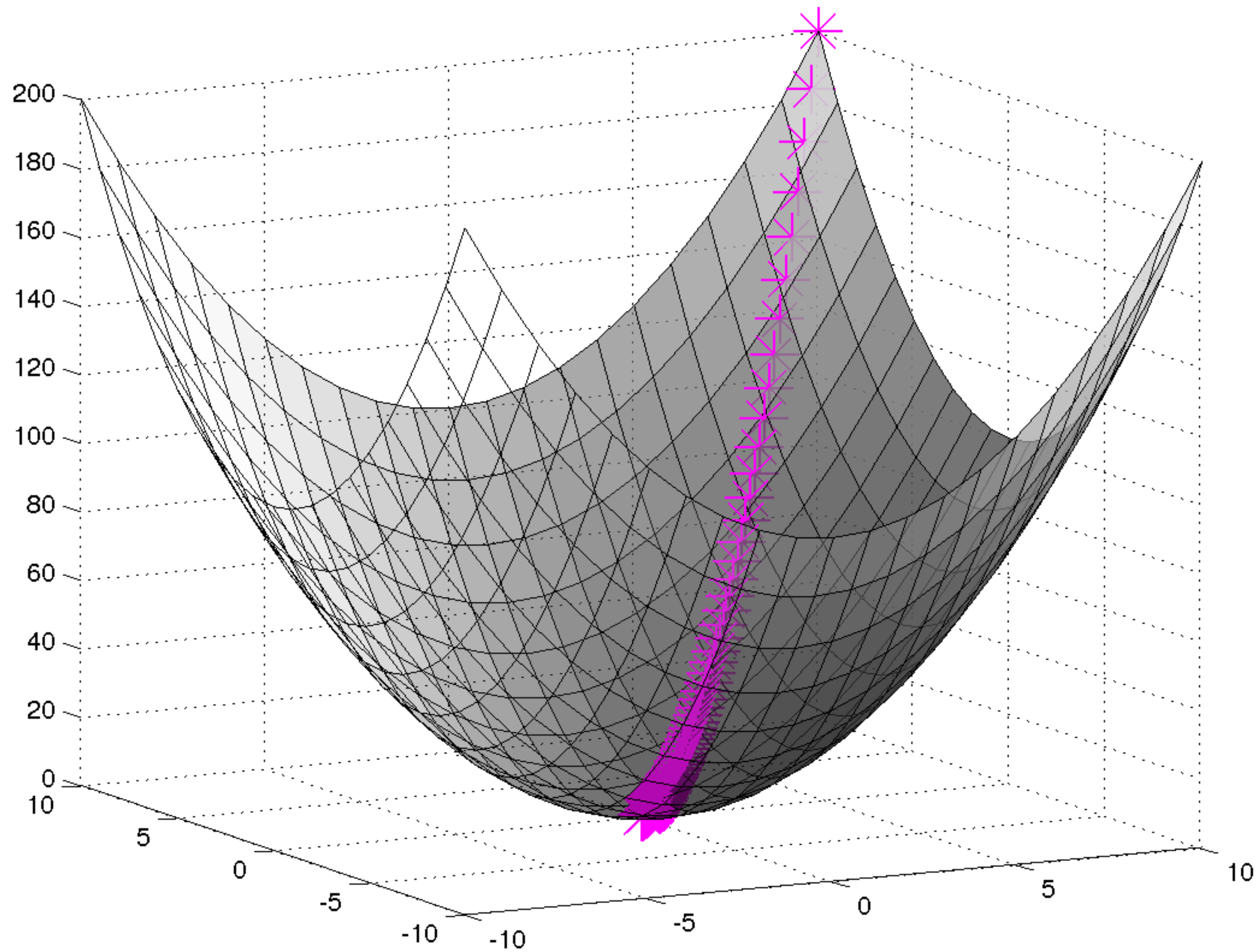
Gradient Descent for Linear Regression

Goal: minimize the following loss function:

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

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

Gradient descent



Gradient Descent for Linear Regression

Goal: minimize the following loss function:

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

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

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

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

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

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

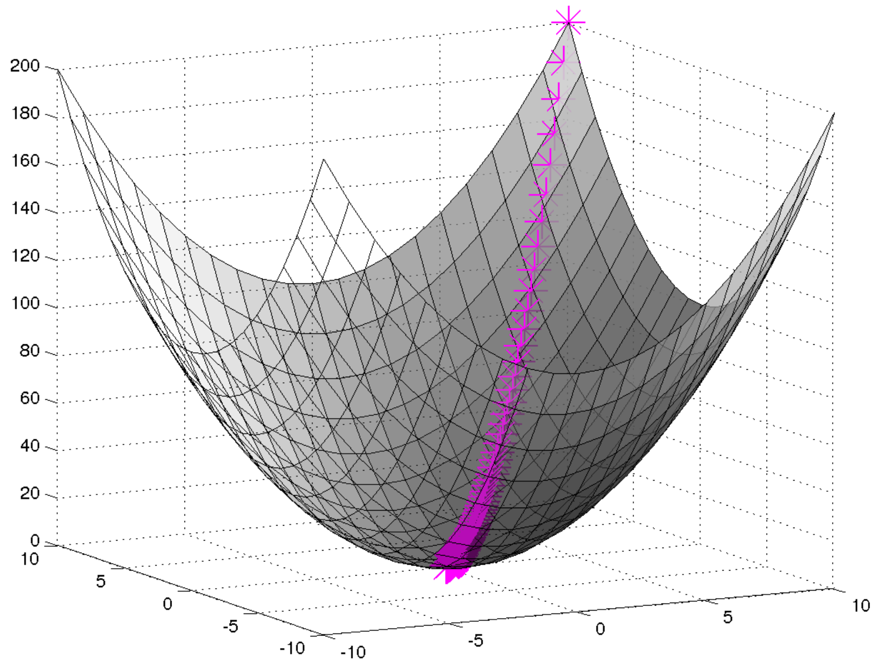
Gradient Descent for Linear Regression

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: $$\frac{\partial}{\partial w_j} J(\mathbf{w}) = 2\sum_i \left(y^i - \hat{y}^i\right)\, \phi_j(\mathbf{x}^i)$$
  - This is a vector **g**
  - Update: **w = w** – λ**g**
  - λ is the learning rate.

Linear regression is a *convex* optimization problem

so again gradient descent will reach a *global* optimum



proof: differentiate again to get the second derivative