Machine Learning

David Suendermann-Oeft

http://suendermann.com

Baden-Wuerttemberg Cooperative State University

Stuttgart, Germany

- **The most up-to-date version of this document as well as auxiliary material can be found online at**

  `http://suendermann.com`

- **For many of the exercises we will be using the programming language Octave available from**

  `http://www.gnu.org/software/octave/`

  Please install and bring your laptops to class.

- **Andrew Ng from the Standford Artificial Intelligence Lab will give an extensive online lecture starting in October. Details at**

  `http://www.ml-class.com/`

Outline

1. **introduction**
2. **probability and statistics**
3. **linear regression**
4. **neural networks**
5. **Bayesian networks**
6. **hidden-Markov models**
7. **decision trees**
8. **boosting**
9. **homework**

Outline

1. introduction
2. probability and statistics
3. linear regression
4. neural networks
5. Bayesian networks
6. hidden-Markov models
7. decision trees
8. boosting
9. homework

- **Machine learning is related to many disciplines in computer science inluding**
  - **artificial intelligence**
  - **pattern recognition**
  - **natural language processing (NLP)**
  - **speech and audio processing**
  - **image processing**
  - **robotics**
  - **financial modeling**
  - **planning**

- **ML is to automatically**
  - **identify patterns in data or**
  - **make decisions based on data.**
- **Technically speaking, the major tasks of ML are**
  - **classification,**
  - **regression, and**
  - **clustering.**

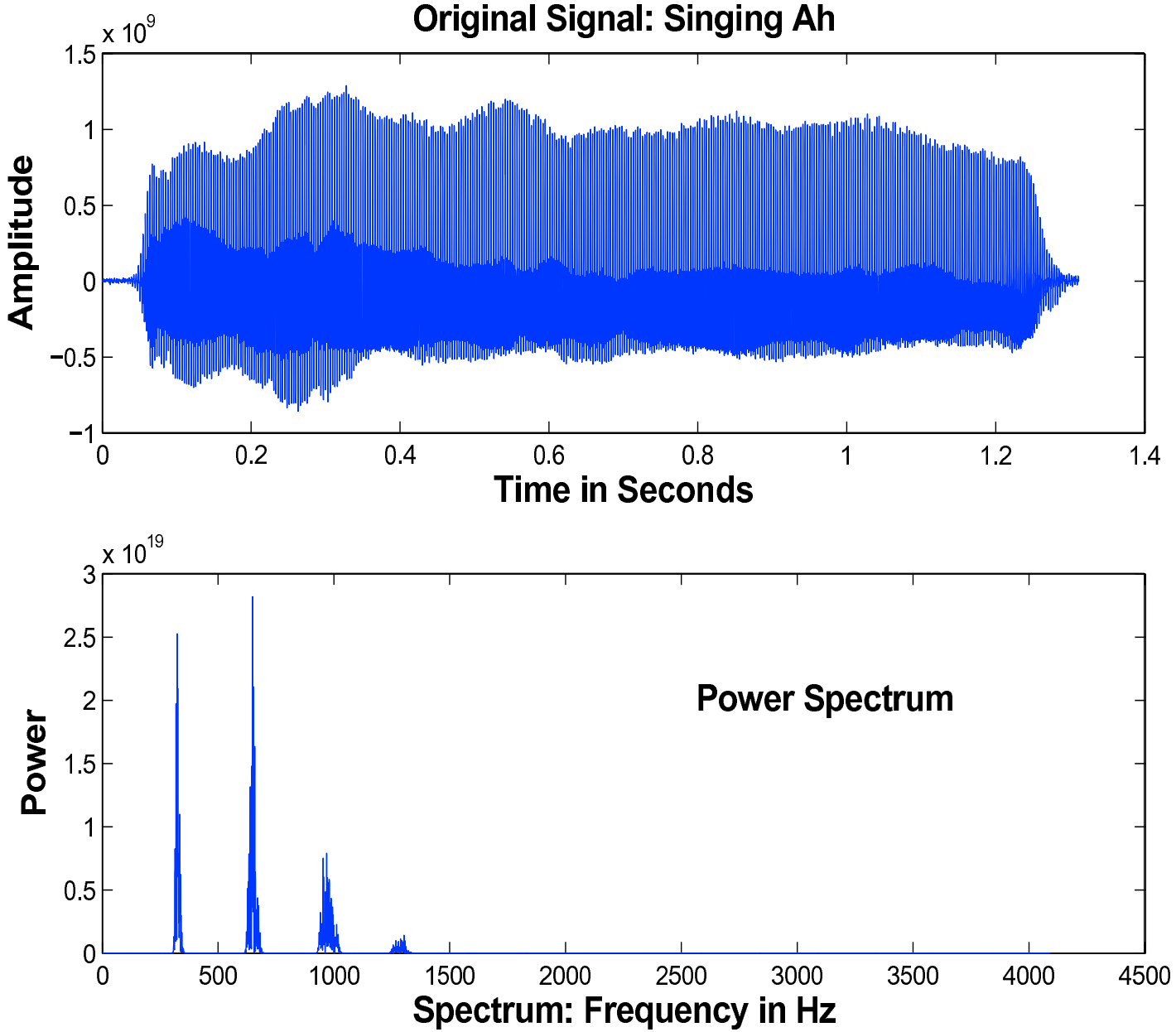
- **Classification is to identify which of N classes an entity x (a data point) belongs to.**
- **The feature vector $\vec{x}$ is composed of return values of feature functions that depend on x:**

$$\vec{x} = \begin{pmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{pmatrix} \tag{1}$$

- **In some cases, the entity x is a vector by itself, so $\vec{x}$ could be composed of its components:**

$$\vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \tag{2}$$

Example feature function in vowel classification: f_1, f_2

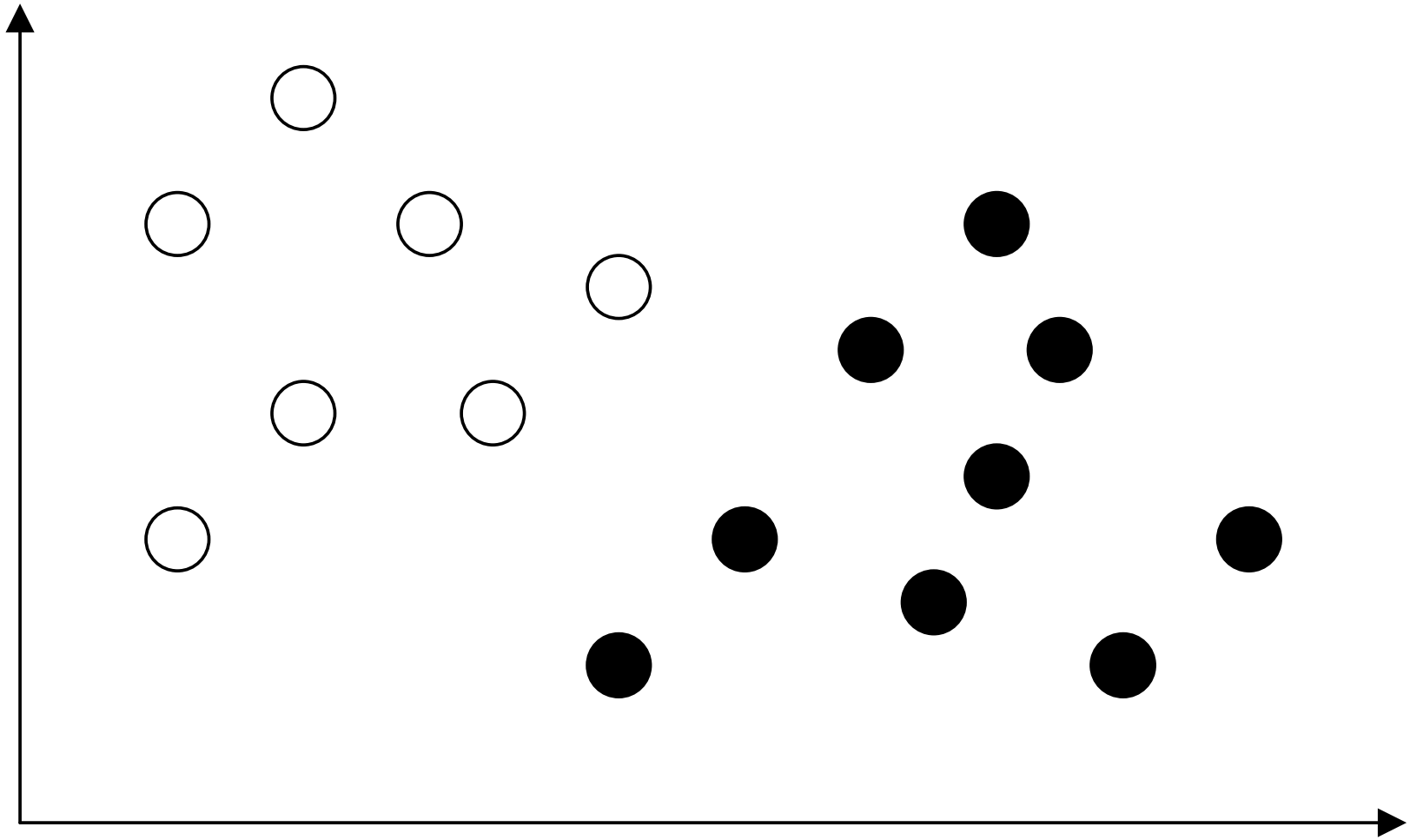


- In addition to the data point x, supervised techniques make use of a target value t.
- In classification, t represents the class of x.
- Accordingly, the goal of classification is to identify a function y such that

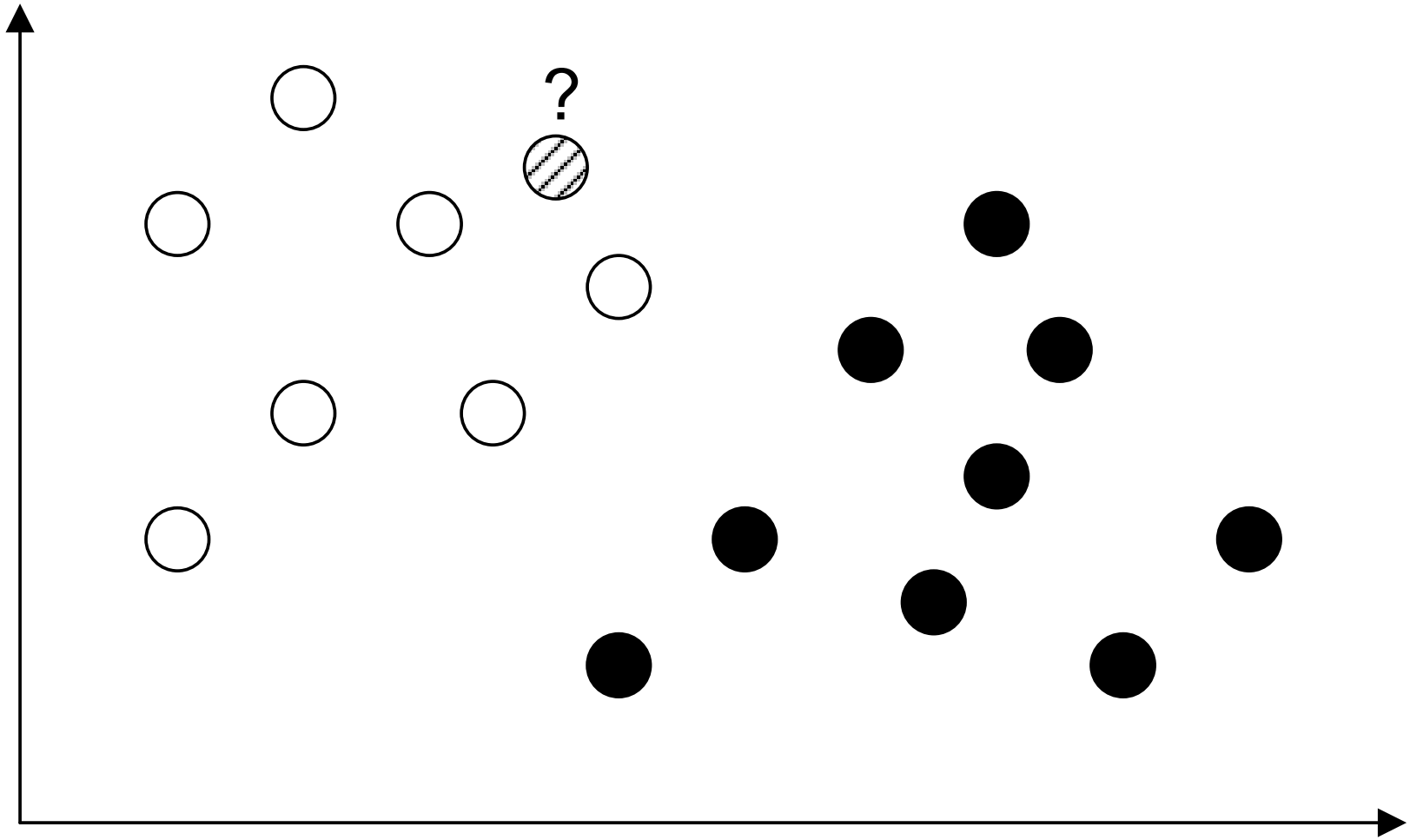
$$y(x) = t. \tag{3}$$

- To come up with a function y that returns the correct t most of the time, we have to find
  - powerful features turning x into $\vec{x}$ and
  - a powerful vector classification technique turning $\vec{x}$ into t.

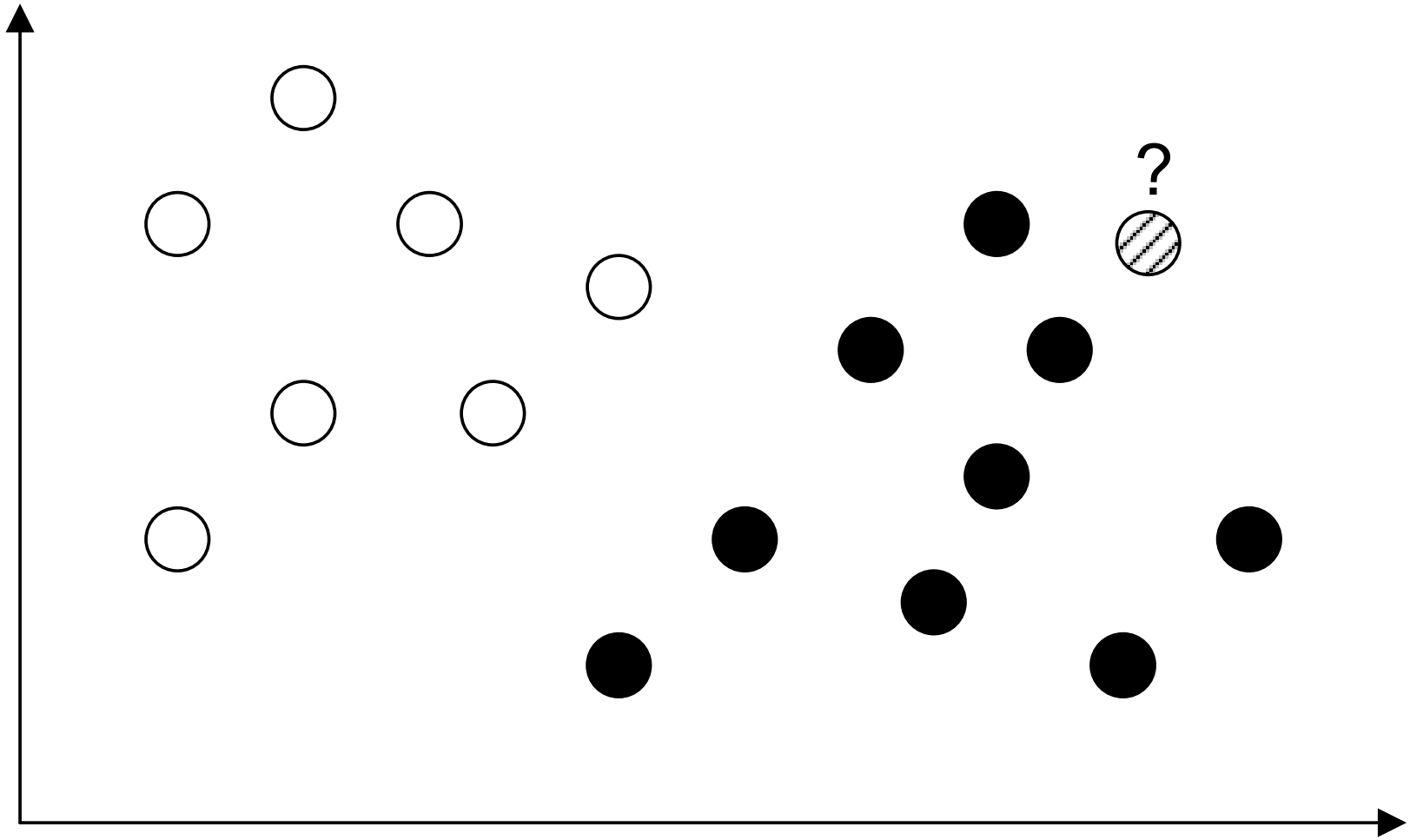
Classification: example



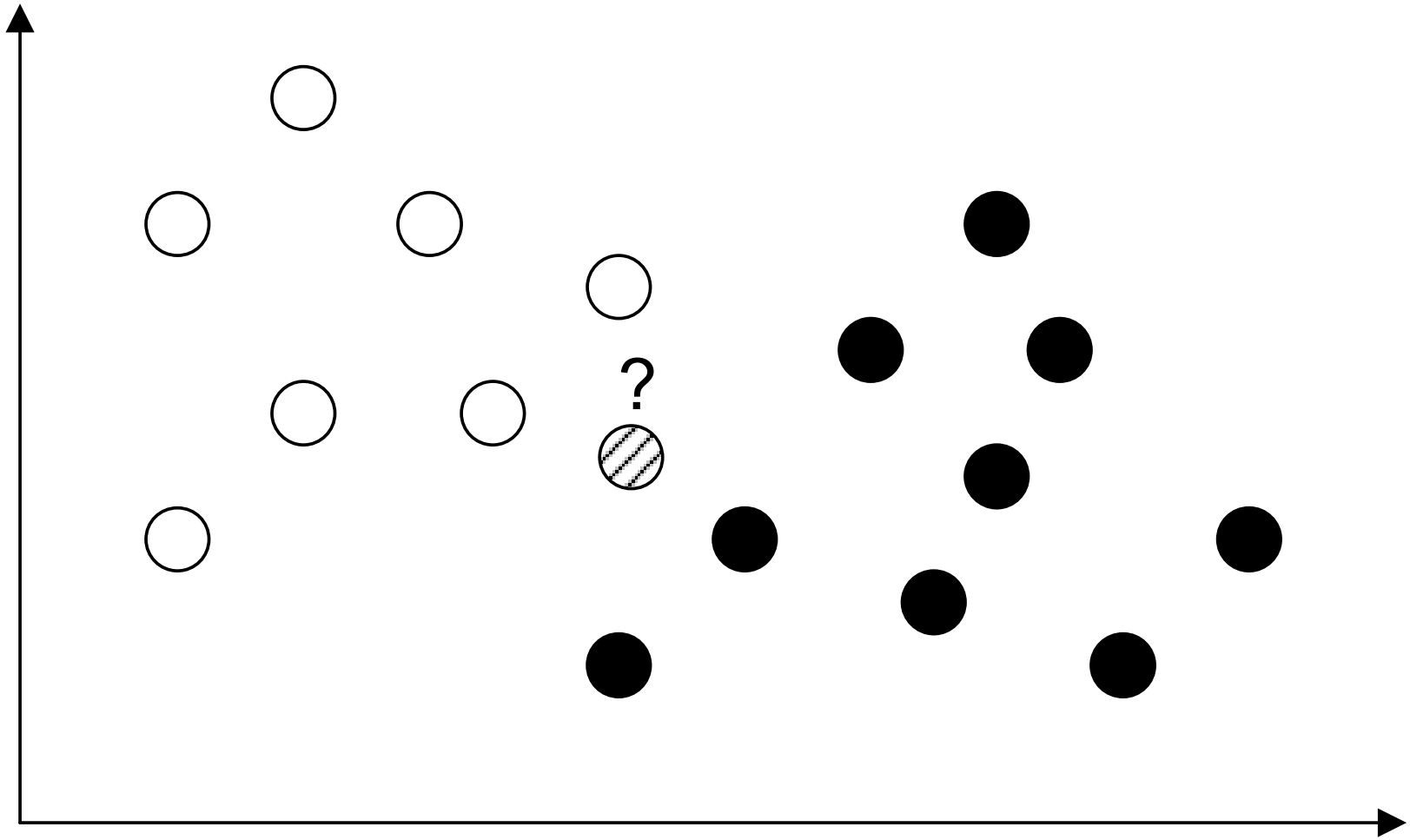
Classification: example



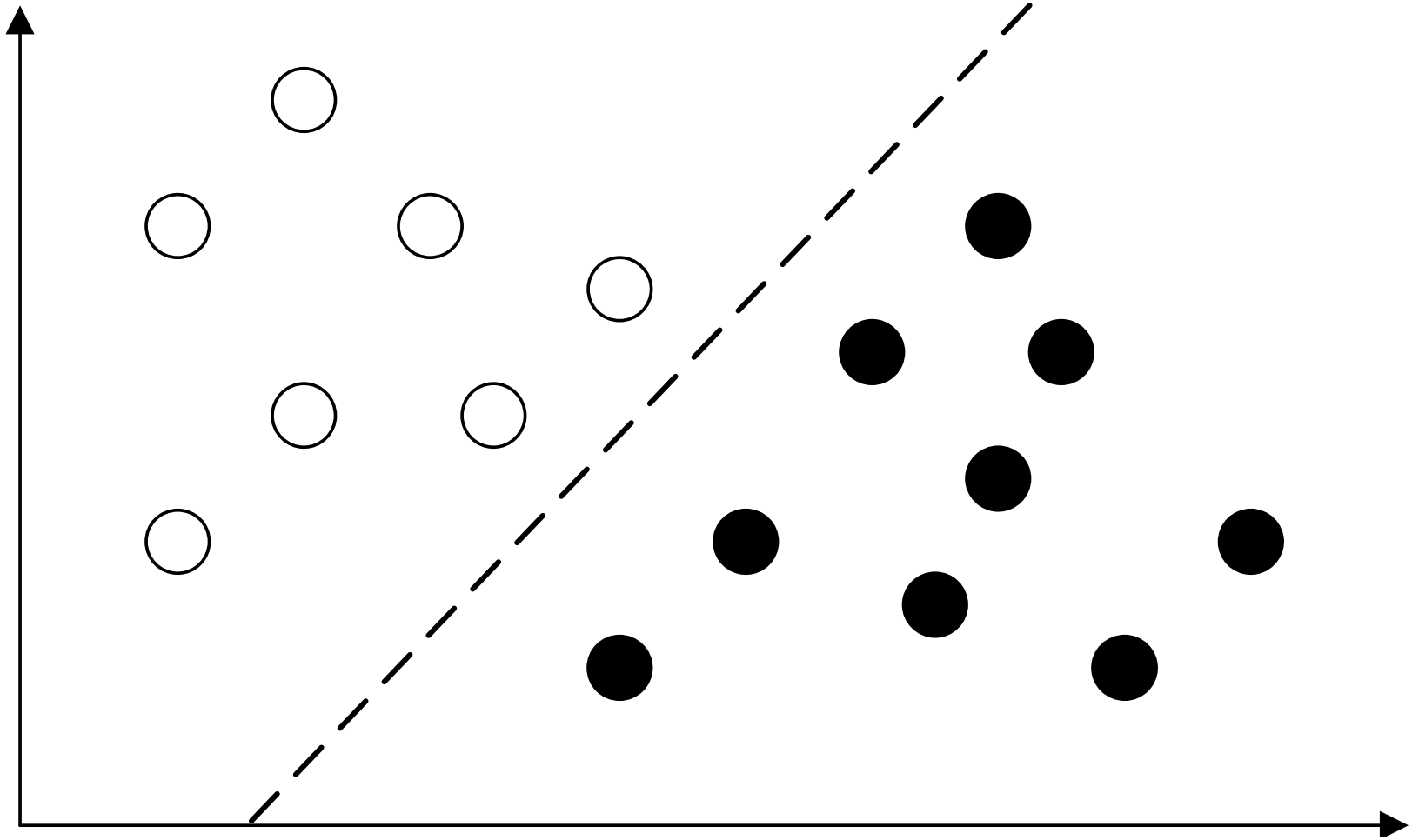
Classification: example



Classification: example



Classification: example

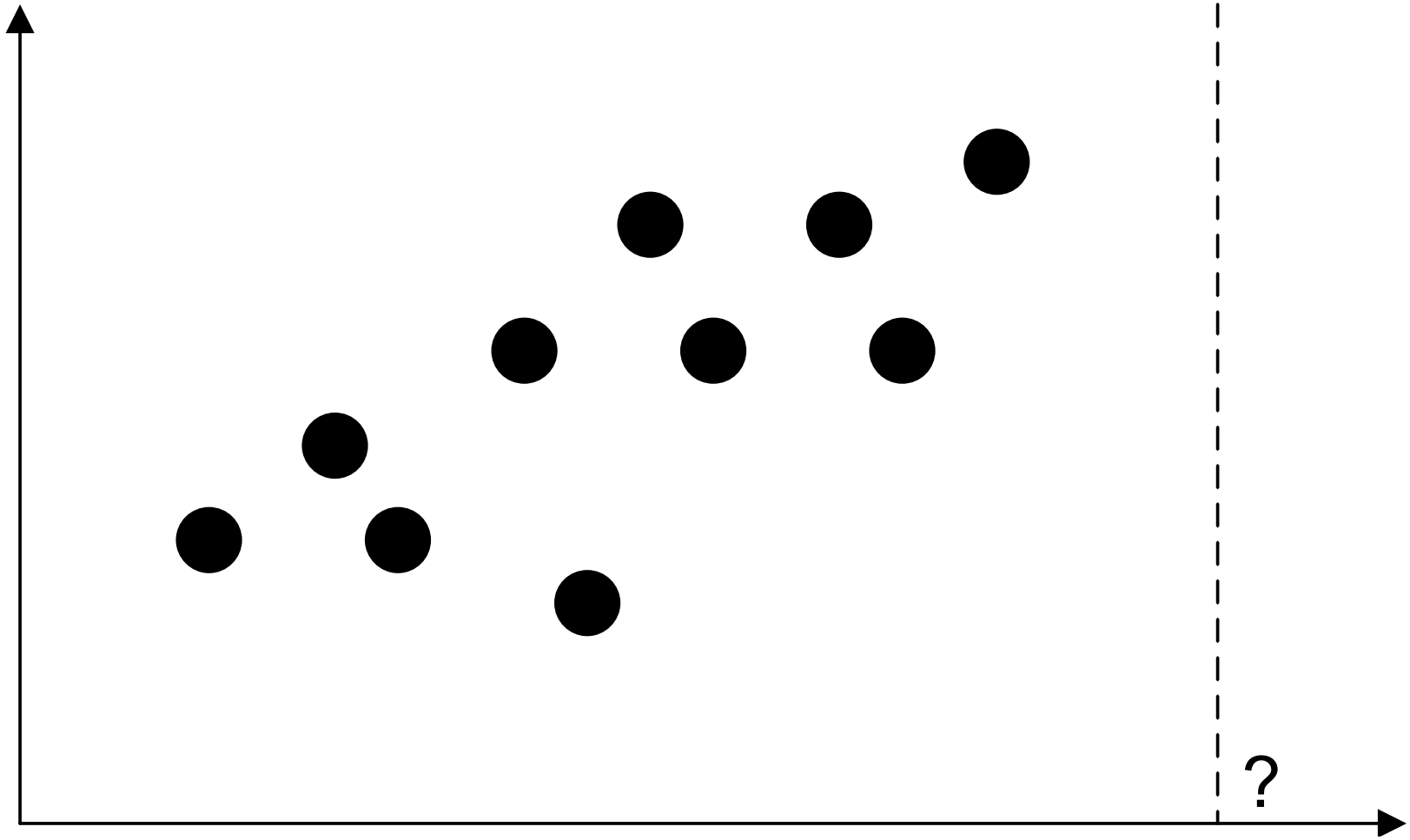


- Classification attempts to predict one class out of a finite and discrete set of classes given x.
- In contrast, regression attempts to predict a continuous variable given x.
- Accordingly, the goal of regression is to identify a function y such that

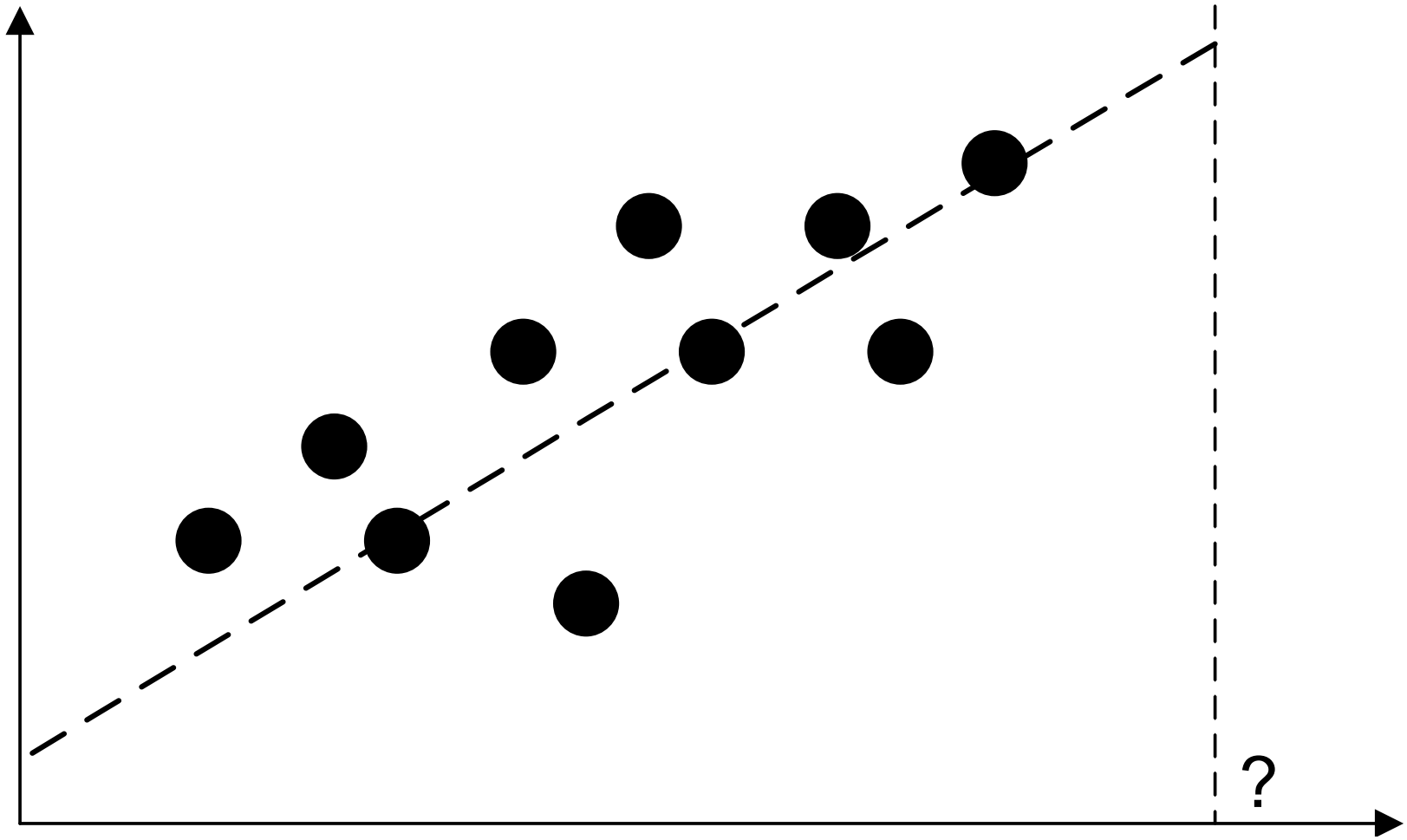
$$y(x) = t. \tag{4}$$

- The goals of classification and regression are identical, so, what is the difference?
- The main difference is how to evaluate performance.
- In classification, a prediction is either identical to the ground truth or not, i.e., it is correct or wrong.
- In regression, a prediction has a distance from the ground truth, i.e., it features a continuous degree of correctness.

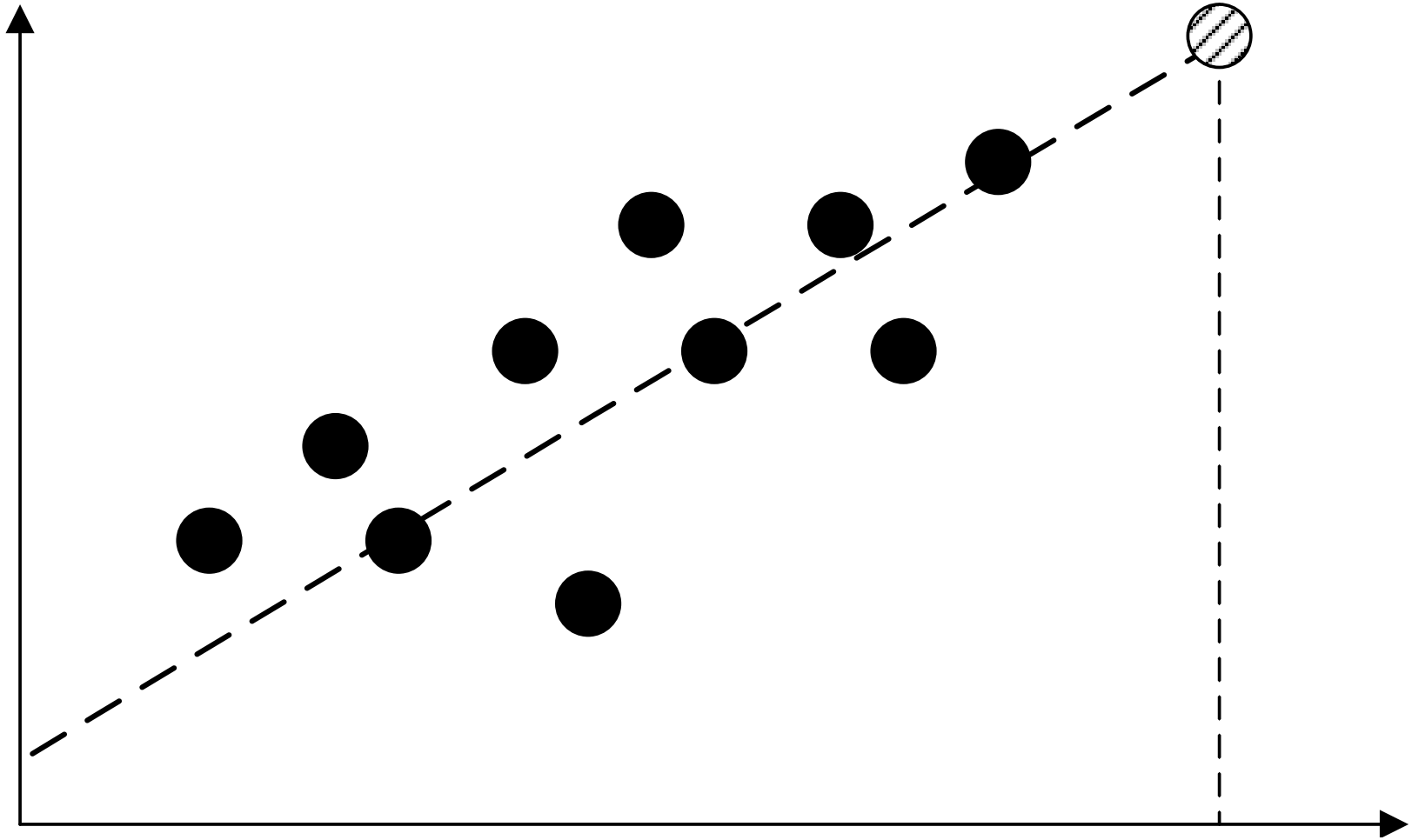
Regression: example



Regression: example

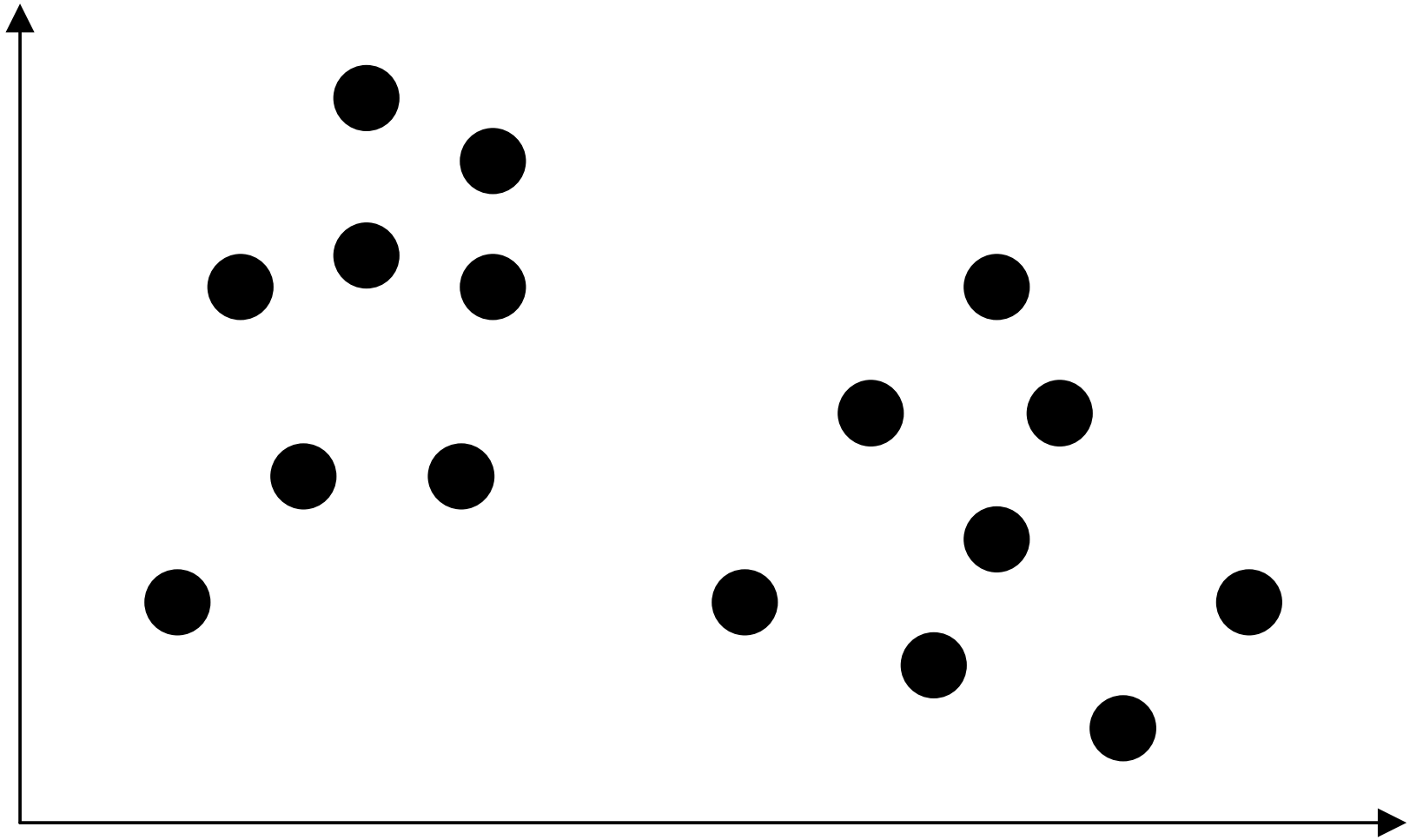


Regression: example

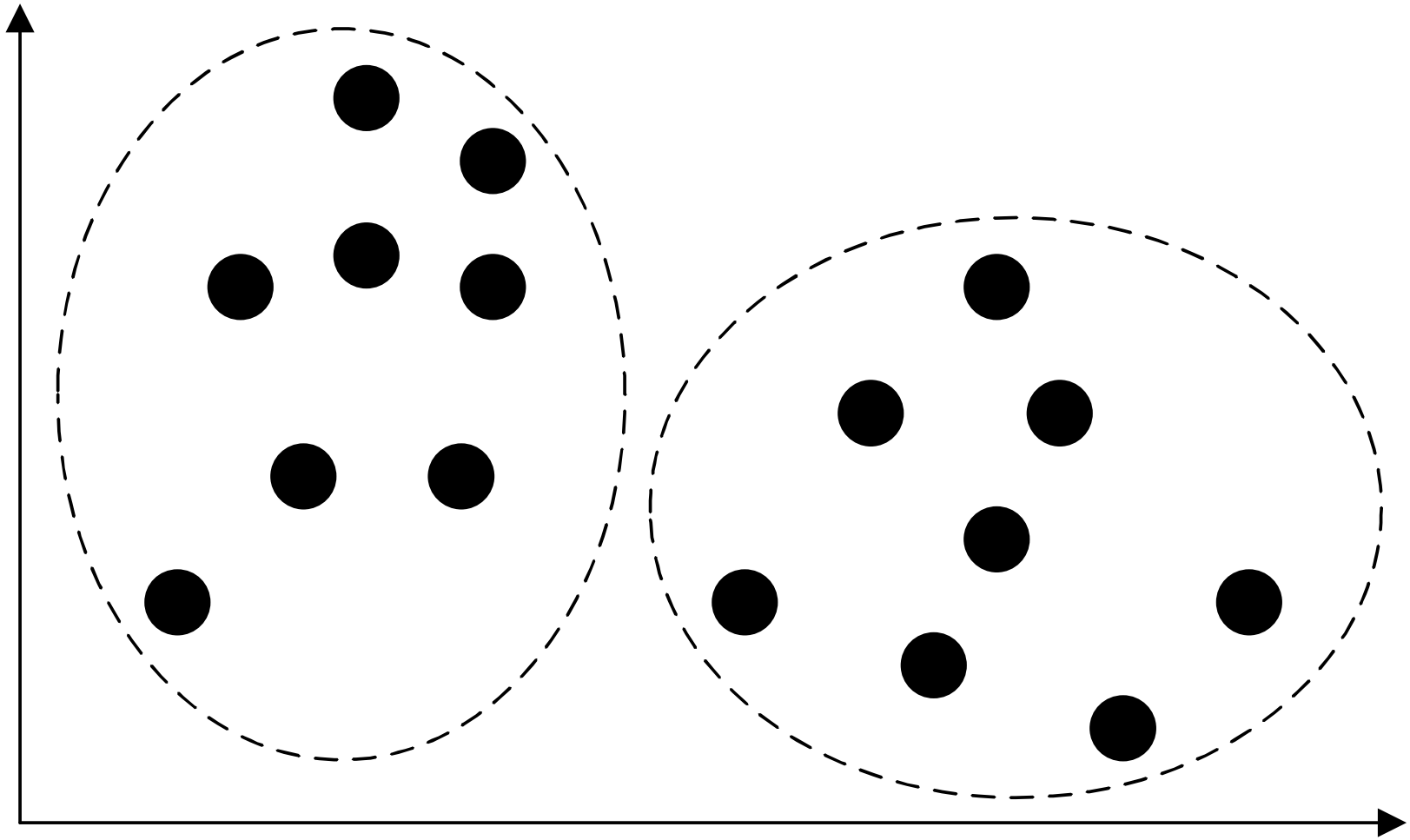


- **Clustering is an unsupervised task.**
- **Hence, we do not have target values.**
- **The goal is to identify groups of similar data points that are dissimilar to others.**
- **Technically speaking, we want to find a partition of the data such that**
  1. **Points in the same cluster are similar.**
  2. **Points in different clusters are dissimilar.**
- **The challenge is to define (dis)similarity for a given type of data.**

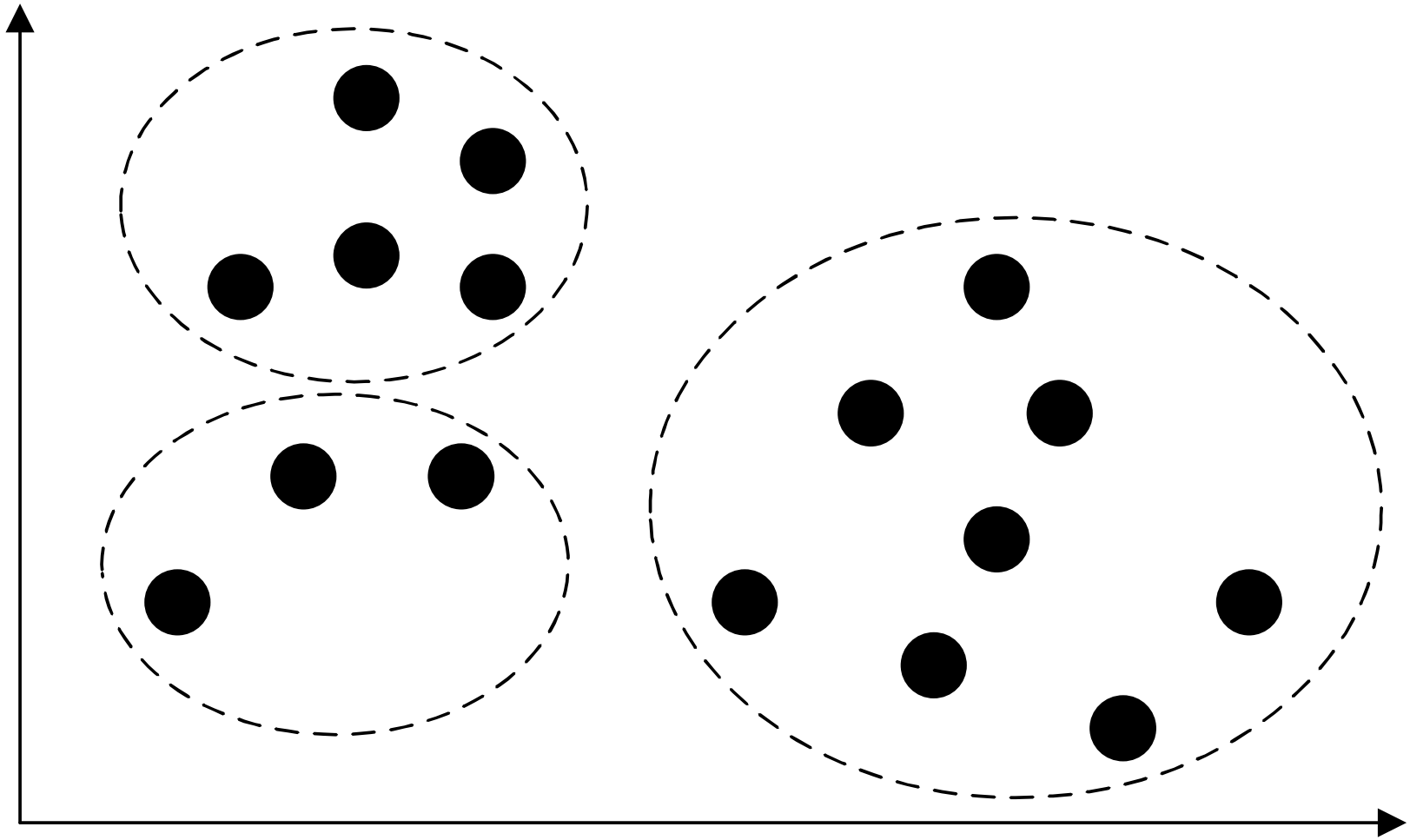
Clustering: example



Clustering: example



Clustering: example



- **Classical artificial intelligence (AI) is based in determinism (rules).**
- **Examples include**
  - **expert systems ($\longrightarrow$ Knowledge-Based Systems)**
  - **theorem provers ($\longrightarrow$ Logic; Knowledge-Based Systems)**
  - **Shakey the robot**
  - **Deep Blue**

- **first general-purpose robot able to reason about its actions**
- **developed between 1966 and 1972 at Standford Research Institute (SRI)**
- **combined research in robotics, image processing, and NLP**
- **programmed in LISP**
- **Sample task:**

  push the block off the platform

- **A research results was the A* search algorithm ($\longrightarrow$ Knowledge-Based Systems).**

- **pic:**
  - **source:** `http://www.flickr.com/photos/15965815@N00/352902842/`
  - **author: Marshall Astor**
  - **license: Creative Commons Attribution-Share Alike 2.0 Generic**

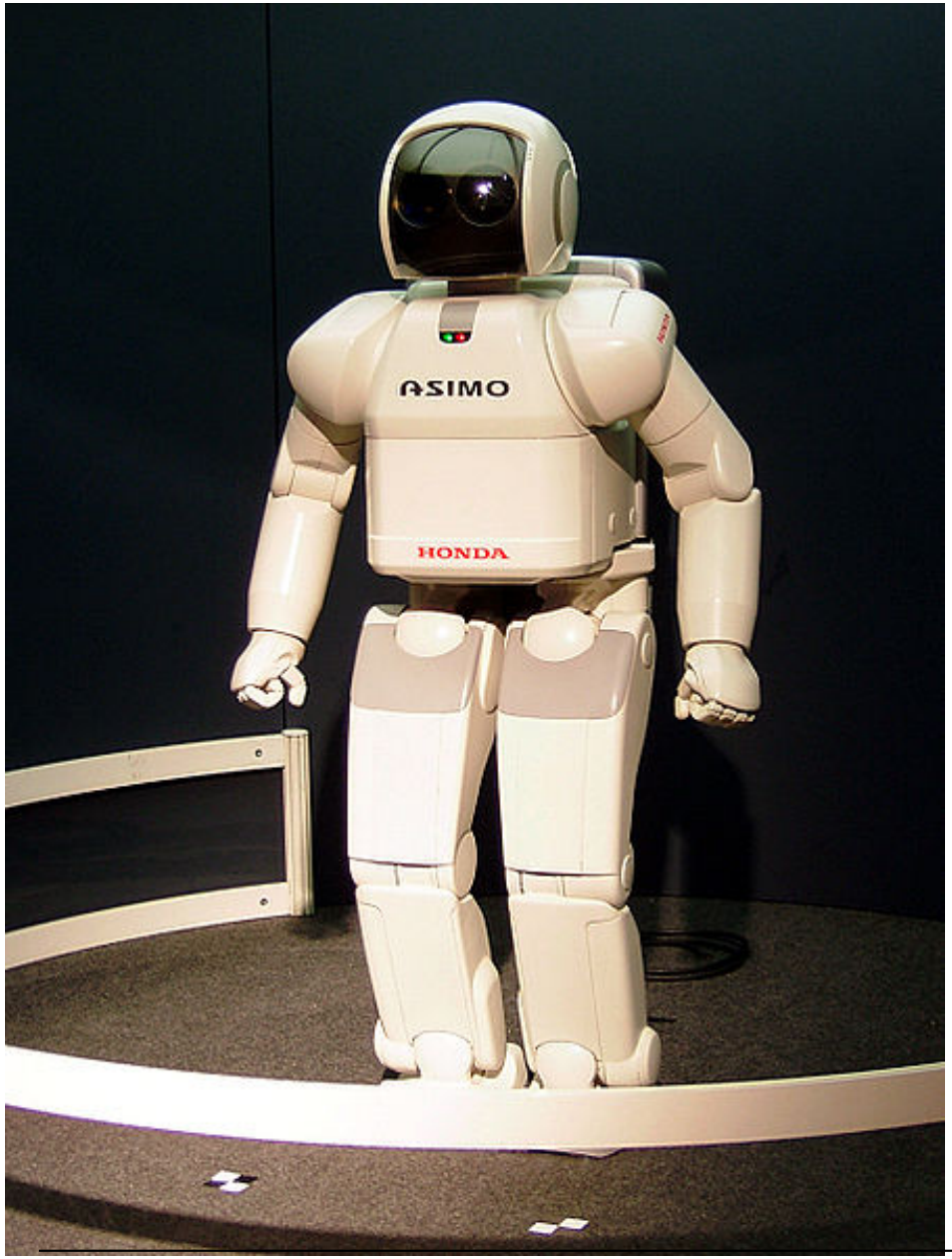
Deep Blue

- chess-playing computer by IBM
- On May 11, 1997, Deep Blue won a six-game match against Garry Kasparov.
- based on brute-force computing power (30 nodes with 480 VLSI chess chips)
- written in C under AIX
- The evaluation function contained multiple parameters tuned on 700,000 grandmaster games.

- pic:
  - source: `http://flickr.com/photos/22453761@N00/592436598/`
  - author: James the photographer
  - license: Creative Commons Attribution 2.0 Generic

- **Modern AI is based in statistical modeling (probabilities).**
- **Examples include**
  - **web search**
  - **speech recognition**
  - **machine translation (MT)**
  - **optical character recognition**
  - **ASIMO**
  - **Watson**

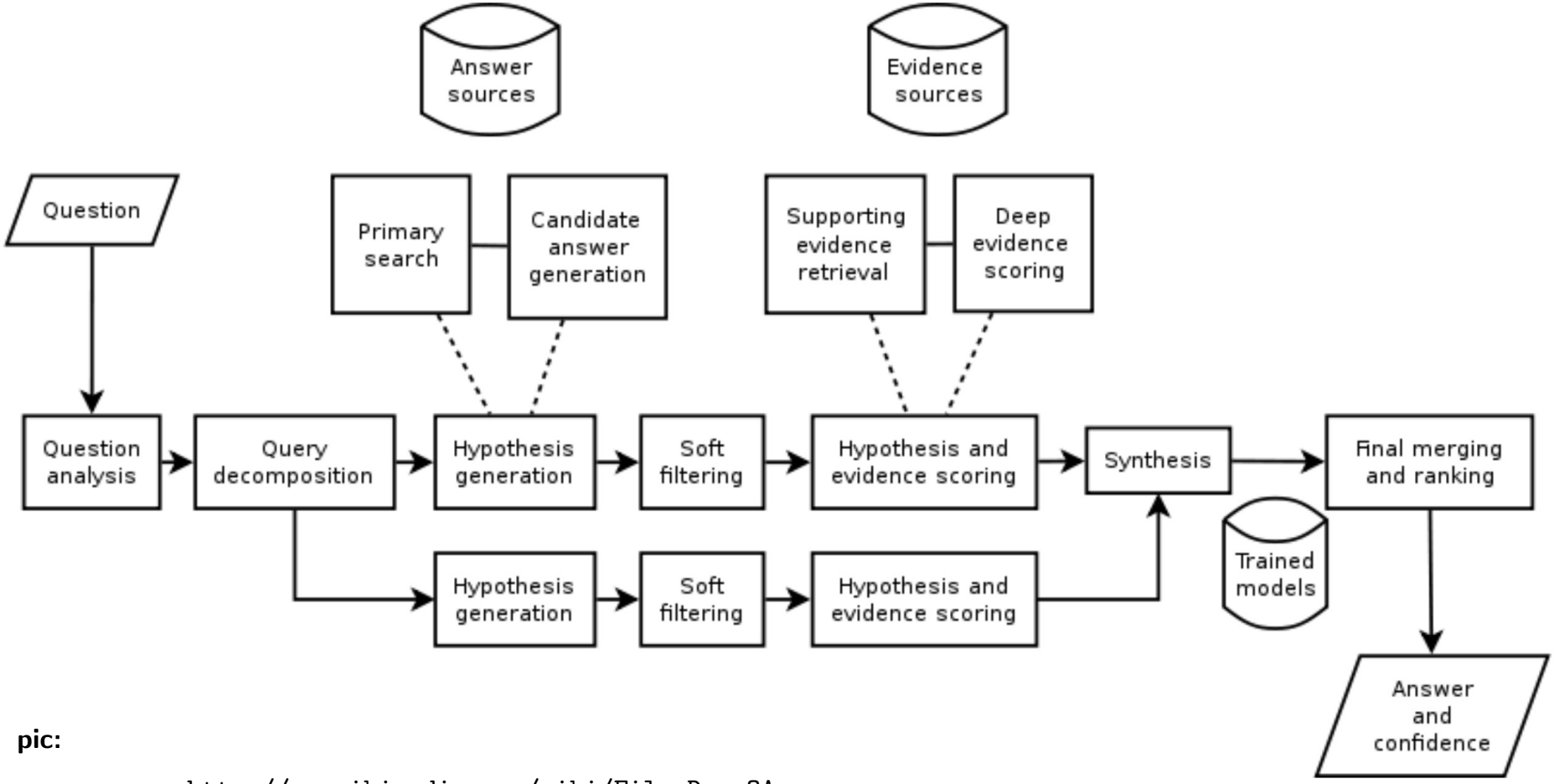
ASIMO



- a humanoid robot by Honda
- stands at 130cm and weighs 54kg
- runs @6km/h on two feet (2005)
- AI technology integrated:
  - detects moving objects
  - interprets postures, gestures, and voice commands
  - faces people when spoken to
  - facial recognition of $\leq$10 faces; addresses people by name
- pic:
  - source: http://upload.wikimedia.org/wikipedia/commons/0/05/HONDA_ASIMO.jpg
  - author: Gnsin
  - license: Creative Commons Attribution-Share Alike 3.0 Unported

- Watson is an AI computer system from IBM for question answering.
- It combines applications of
  - machine learning,
  - NLP,
  - information retrieval,
  - knowledge representation,
  - reasoning.
- To showcase its abilities, in February 2011, Watson competed on the show Jeopardy! against the human champions and won.
- During the quiz, Watson had no access to the Internet.
- It had access to 200M pages of structured and unstructured data (including a copy of the entire Wikipedia), amounting to 4TB.
- Hardware consisted of
  - 90 IBM Power 750 servers with 2880 processors and 16TB of RAM.

Watson: architecture



- **pic:**
  - **source:** `http://en.wikipedia.org/wiki/File:DeepQA.svg`
  - **author: Pgr94**
  - **license: Creative Commons CC0 1.0 Universal Public Domain Dedication**

1. introduction
2. probability and statistics
3. linear regression
4. neural networks
5. Bayesian networks
6. hidden-Markov models
7. decision trees
8. boosting
9. homework

- Suppose we have a blue (b) and a red (r) box.
- In the blue box, there are 2 apples (a) and 6 oranges (o).
- In the red box, there are 3 apples and 1 orange.
- Suppose, we draw each fruit token with the same probability.
- We consider box B and fruit F to be random variables.
- B can have the values b and r.
- F can have the values a and o.
- These are questions we want to answer:
  - What is the probability of picking an apple?
  - Given I chose an orange, what is the probability that it was drawn from the blue box?

- The probability p of an event is the fraction of time the event occurs out of some number of trials approaching infinity.
- $0 \leq p \leq 1$
- Events that cannot cooccur are called mutually exclusive.
- Probabilities of mutually exclusive events sum up to 1.
- If two events x and y are independent, we have

$$p(x, y) = p(x)p(y) \tag{5}$$

and

$$p(x|y) = p(x). \tag{6}$$

- **Count table**

from our example:

	a	o	Σ
b	2	6	8
r	3	1	4
Σ	5	7	12

generalized:

	x_i	
y_j	n_{ij}	$r_j = \sum_i n_{ij}$
	$c_i = \sum_j n_{ij}$	$N = \sum_i \sum_j n_{ij}$

- The joint probability describes how likely events occur simultaneously:

$$p(x_i, y_j) = \frac{n_{ij}}{N}. \tag{7}$$

- E.g., selecting an orange from the blue box:

$$p(b, o) = \frac{6}{12} = \frac{1}{2}. \tag{8}$$

- The probability of x_i irrespective of y_j is

$$p(x_i) = \frac{c_i}{N} = \frac{\sum_j n_{ij}}{N} = \sum_j p(x_i, y_i). \tag{9}$$

- E.g., selecting the blue box:

$$p(b) = p(b, a) + p(b, o) = \frac{1}{6} + \frac{1}{2} = \frac{2}{3}. \tag{10}$$

- We now limit our analysis only to events with x_i and want to analyze the fraction of these cooccurring with y_j.
- This case is the conditional probability of y_j given x_i:

$$p(y_j|x_i) = \frac{n_{ij}}{c_i}. \tag{11}$$

- E.g., the probability of having chosen the blue box when eating an orange:

$$p(b|o) = \frac{6}{7}. \tag{12}$$

- We can express the joint probability in terms of conditional probabilities:

$$p(x_i, y_j) = \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} \cdot \frac{c_i}{N} = p(y_j|x_i)p(x_i). \tag{13}$$

- Generalizing from the probability $p(x_i)$ of a particular value x_i, we get a probability distribution $p(X)$ of the random variable X.
- Now, we can derive
  - the sum rule (from Equation 9):

$$p(X) = \sum_{Y} p(X, Y), \tag{14}$$

  - the product rule (from Equation 13):

$$p(X, Y) = p(Y|X)p(X), \tag{15}$$

  - Bayes' rule (from double application of Equation 15):

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}. \tag{16}$$

- We want to look one more time at the question

  Given I chose an orange, what is the probability that it was drawn from the blue box?

- Before knowing which fruit I chose, the probability distribution of chosing boxes was $P(B)$ (prior probability).

- After chosing a fuit (F), the probability distribution of chosing boxes changed to $P(B|F)$ (posterior probability).

- In our example, we have the prior

$$p(b) = \frac{2}{3} \tag{17}$$

  and the posterior

$$p(b|o) = \frac{6}{7}. \tag{18}$$

- That is, the probability that the box was blue increased after observing o.

- A probabilistic MT system is to translate an Urdu (U) source sentence into English (E).
- We are given the following count tables:

translation model

	u_1	u_2	u_3	u_4
e_1	0	1	0	1
e_2	1	1	1	0
e_3	1	0	1	0
e_4	0	1	38	0
e_5	0	0	1	0

language model for U

u_1	u_2	u_3	u_4
15	632	52	23

language model for E

e_1	e_2	e_3	e_4	e_5
450	891	586	899	7638

- The MT system uses a smoothing strategy replacing zero counts in the count table by 0.5 to prevent unlikely events to be suppressed.

- **What is the best translation of the sentences**

  a) u_3 **and**

  b) u_4

  using

  I) simple posterior probabilities,

  II) Bayes' rule?

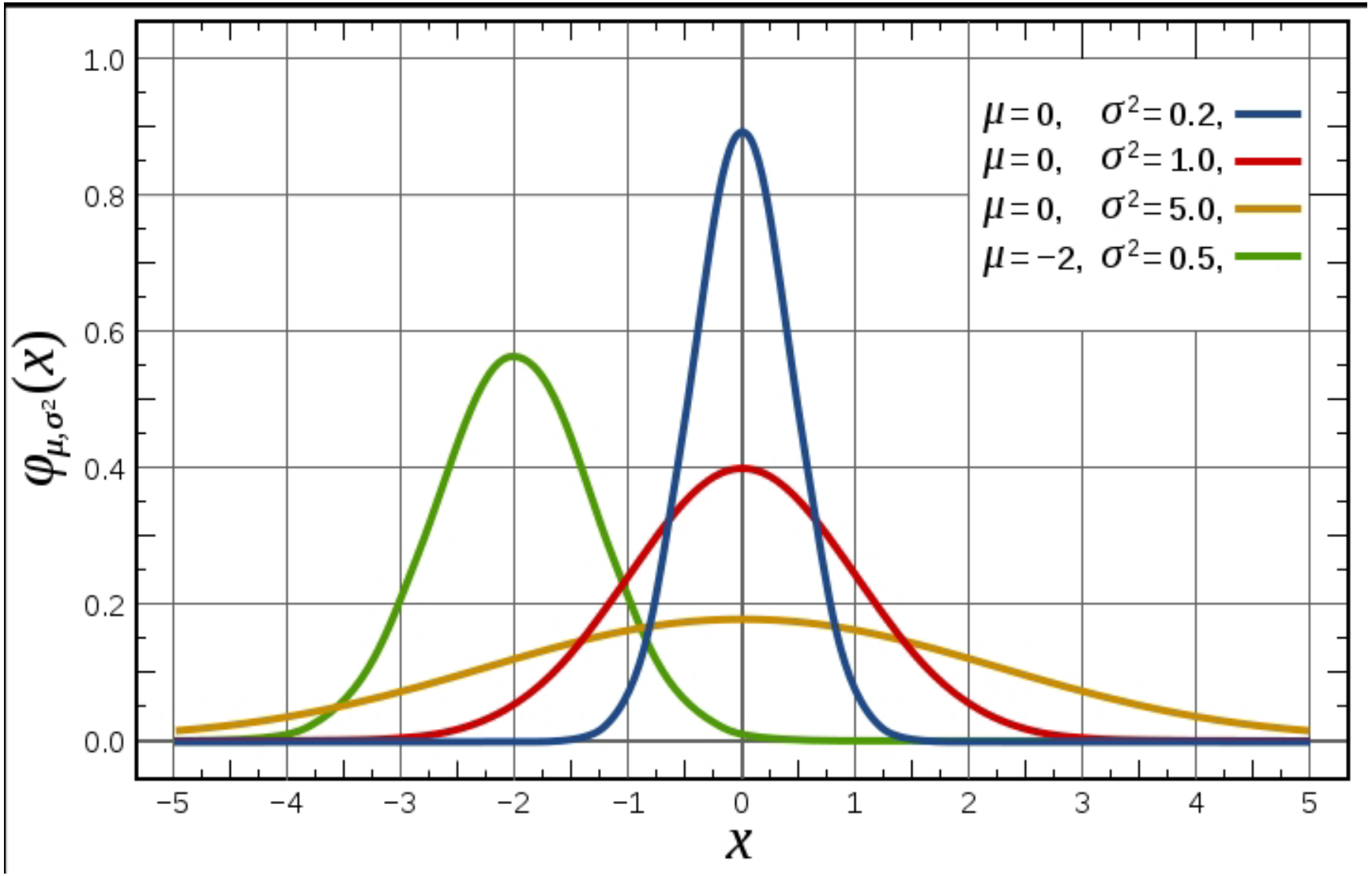
- So far, we talked about discrete probability.
- That is, X can take one out of a finite set of discrete values $(x_1, \ldots, x_I)$.
- If X becomes continuous, there are infinitely many different values, so, the probability of a specific value may approach zero.
- Hence, we introduce the notion of a probability density function (PDF) $p(x)$.
- We can calculate the probability that x falls in an interval (x_0, x_1) using the definite integral

$$p(x \in (x_0, x_1)) = \int_{x_0}^{x_1} p(x)\mathrm{d}x. \tag{19}$$

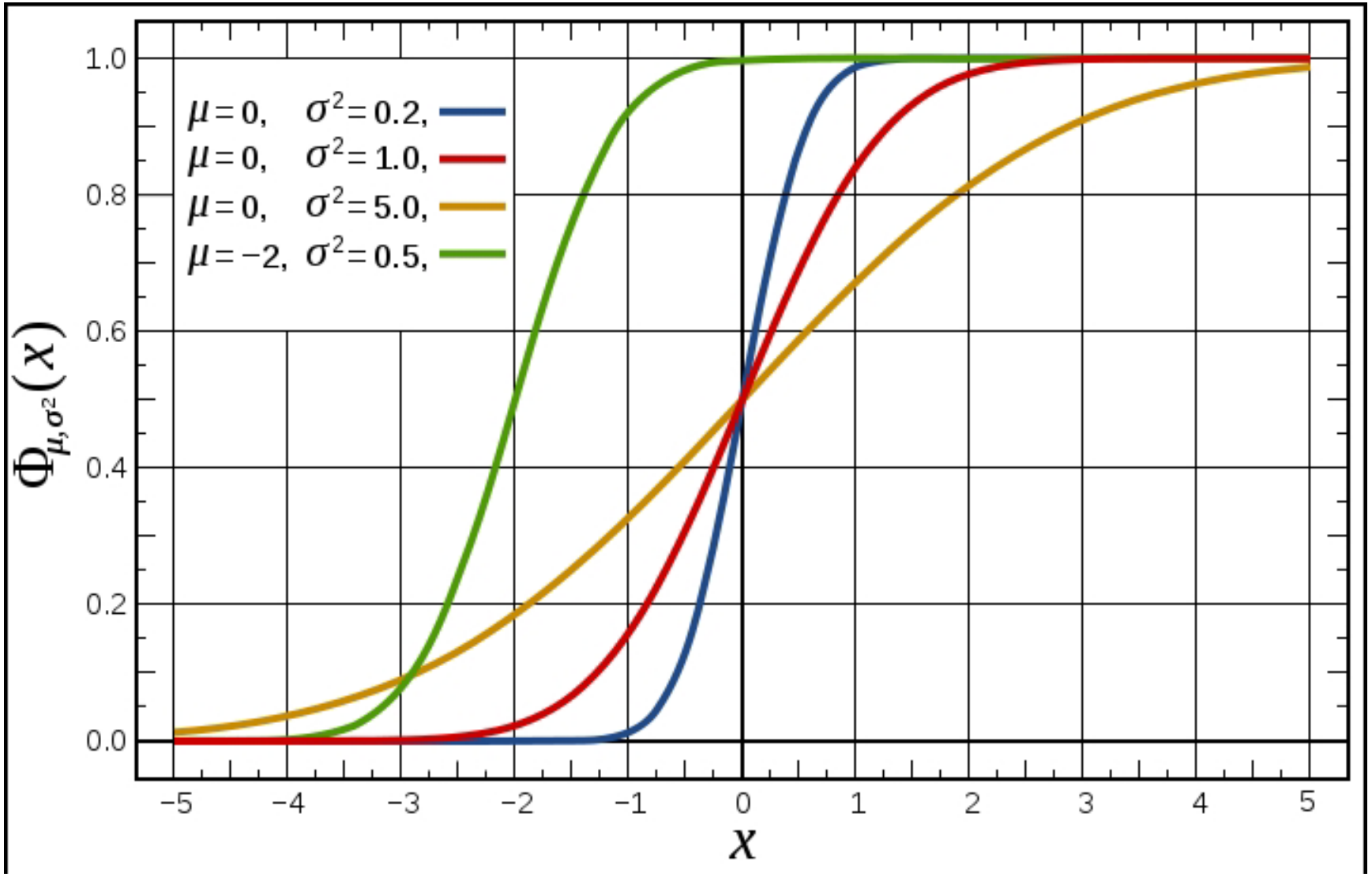
- A special case is the cumulative density function (CDF) $P(x)$:

$$P(x) = \int_{-\infty}^{x} f(u)\mathrm{d}u. \tag{20}$$

PDF: examples



CDF: examples



- **range of values:**

$$\int_{-\infty}^{\infty} p(x)\mathrm{d}x = P(\infty) = 1; p(x) \geq 0 \tag{21}$$

- **sum rule:**

$$p(x) = \int_{-\infty}^{\infty} p(x, y)\mathrm{d}y \tag{22}$$

- **product rule:**

$$p(x, y) = p(y|x)p(x) \tag{23}$$

- **Bayes' rule:**

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \tag{24}$$

- The expected value or expectation of a random variable is
  - for discrete distributions:

$$\mathbf{E}(X) = \sum_{x} xp(x) \tag{25}$$

  - for continuous distributions:

$$\mathbf{E}(X) = \int_{-\infty}^{\infty} xp(x)\mathrm{d}x \tag{26}$$

  - theorems (X and Y are independent random variables):

$$\mathbf{E}(XY) = \mathbf{E}(X)\mathbf{E}(Y) \tag{27}$$

$$\mathbf{E}(X+Y) = \mathbf{E}(X) + \mathbf{E}(Y) \tag{28}$$

- **We are rolling**
  - **a) one die,**
  - **b) two dice,**
  - **c) three dice.**
- **Assuming the random variable X is the sum of numbers obtained at each roll, calculate**
  - **I) $p(X)$,**
  - **II) $\mathbf{E}(X)$.**
- **Plot $p(X)$ for a), b), and c). How do you think $p(X)$ looks like for a larger number of rolls?**

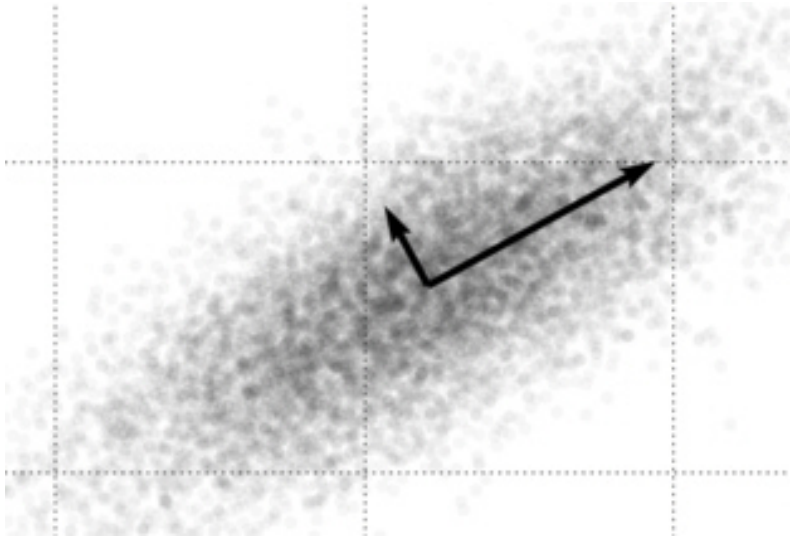
- **univariate (one-dimensional) Gaussian/normal distribution:**

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{29}$$

- **pic:**
  - **source:** `http://en.wikipedia.org/wiki/File:DEU-10m-anv.jpg`
  - **author: Deutsche Bundesbank (banknote), European Central Bank (photo)**
  - **permission: ECB/2003/4 and ECB/2003/5**

- **multivariate (D-dimensional) normal distribution:**

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\tfrac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)\right) \quad (30)$$



- **pic:**
  - **source:** http://en.wikipedia.org/wiki/File:GaussianScatterPCA.png
  - **author: Ben FrantzDale**
  - **license: GNU Free Documentation License 1.2**

- **The expectation of the univariate normal distribution is**

$$\begin{aligned}
\mathbf{E}(X) &= \mathbf{E}(Y+\mu) \quad \textbf{with} \quad Y = X - \mu && (31)\\
&= \mathbf{E}(Y) + \mu && (32)\\
&= \int_{-\infty}^{\infty} y \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} \mathrm{d}y + \mu && (33)\\
&= -\frac{\sigma^2}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} y \frac{y}{\sigma^2} e^{-\frac{y^2}{2\sigma^2}} \mathrm{d}y + \mu && (34)\\
&= -\frac{\sigma^2}{\sqrt{2\pi\sigma^2}} \left[e^{-\frac{y^2}{2\sigma^2}} \right]_{y=-\infty}^{\infty} + \mu && (35)\\
&= \mu. && (36)
\end{aligned}$$

- **The variance expresses how broad is a distribution around its mean (expectation):**

$$\begin{aligned}
\mathbf{Var}(X) &= \mathbf{E}((X - \mathbf{E}(X))^2) && (37)\\
&= \mathbf{E}((X - \mu)^2) && (38)\\
&= \mathbf{E}(X^2 - 2X\mu + \mu^2) && (39)\\
&= \mathbf{E}(X^2) - 2\mathbf{E}(X)\mu + \mu^2 && (40)\\
&= \mathbf{E}(X^2) - 2\mu^2 + \mu^2 && (41)\\
&= \mathbf{E}(X^2) - \mathbf{E}(X)^2 && (42)
\end{aligned}$$

- **Calculate the variance of a univariant normal distribution.**

- The covariance expresses how two random variables X and Y change together.
- A special case of the covariance is the case that $X = Y$

$$\mathrm{Cov}(X, Y) = \mathrm{Var}(X). \tag{43}$$

- Another special case is when X and Y are independent where

$$\mathrm{Cov}(X, Y) = 0. \tag{44}$$

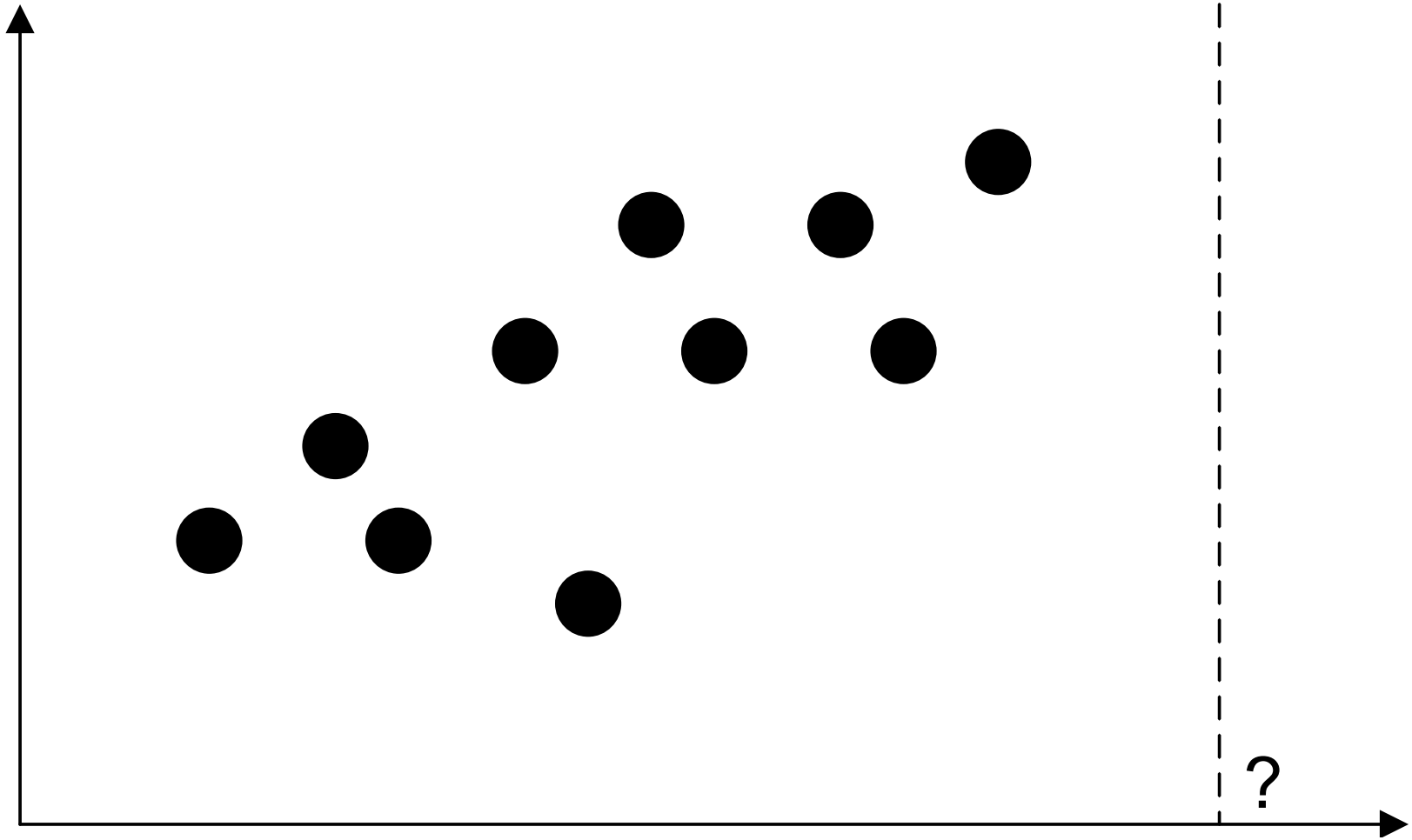
- Generally, we have

$$\begin{aligned}
\mathrm{Cov}(X,) &= \mathrm{E}((X - \mathrm{E}(X))(Y - \mathrm{E}(Y))) && (45)\\
&= \mathrm{E}((X - \mu_x)(Y - \mu_y)) && (46)\\
&= \mathrm{E}(XY - X\mu_y - \mu_x Y + \mu_x\mu_y) && (47)\\
&= \mathrm{E}(XY) - \mathrm{E}(X)\mu_y - \mu_x\mathrm{E}(Y) + \mu_x\mu_y && (48)\\
&= \mathrm{E}(XY) - \mu_x\mu_y - \mu_x\mu_y + \mu_x\mu_y && (49)\\
&= \mathrm{E}(XY) - \mathrm{E}(X)\mathrm{E}(Y) && (50)
\end{aligned}$$

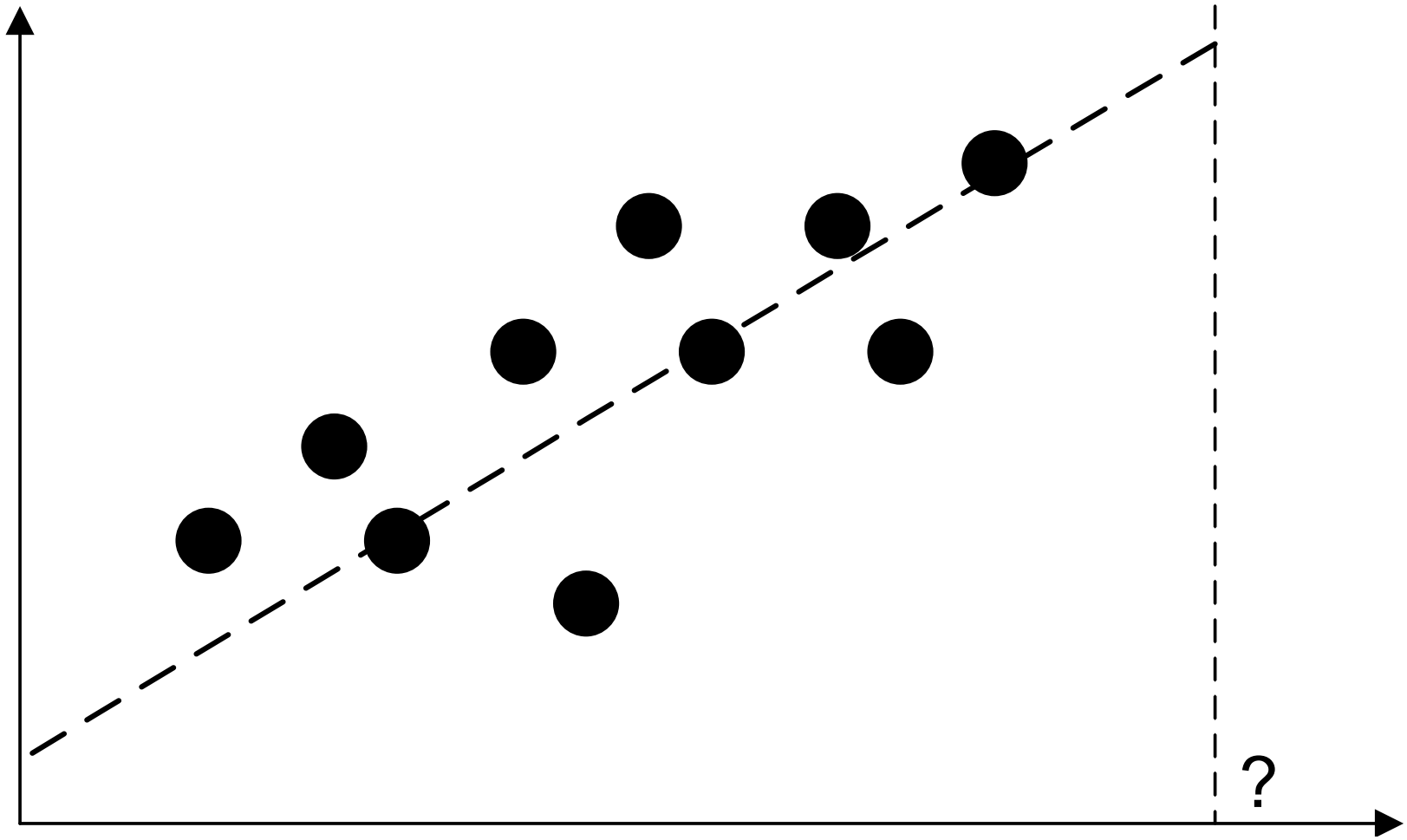
1. introduction
2. probability and statistics
3. linear regression
4. neural networks
5. Bayesian networks
6. hidden-Markov models
7. decision trees
8. boosting
9. homework

- The goal of regression is to learn a function $y(x)$
  - for one-dimensional x:
    $$y : \mathbb{R} \to \mathbb{R} \tag{51}$$
  - for D-dimensional x:
    $$y : \mathbb{R}^D \to \mathbb{R} \tag{52}$$
- Since regression is a supervised technique, we are given
  - the set of training data points $x_1, \ldots, x_N$ and
  - the respective targets $t_1, \ldots, t_N$.

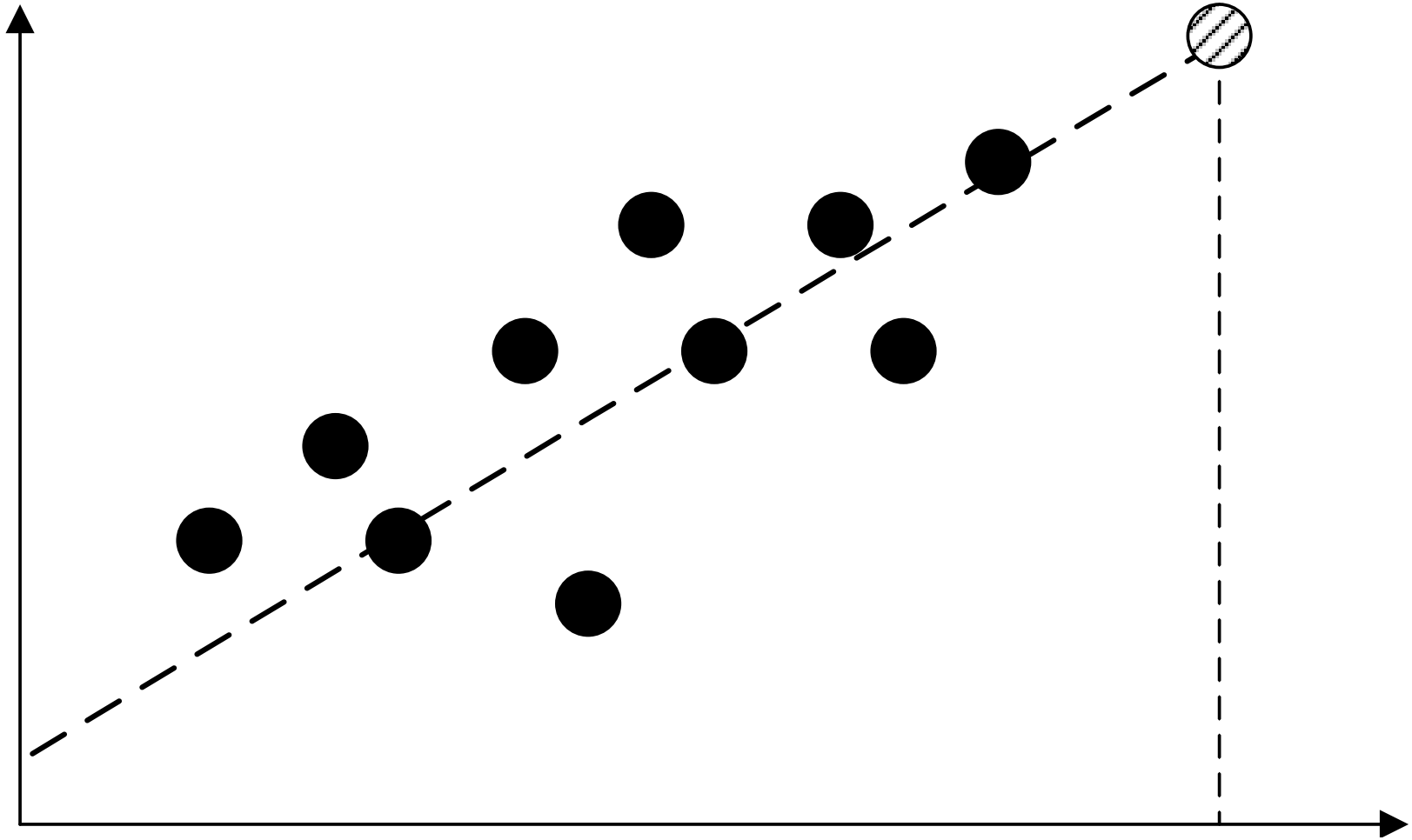
Regression: example



Regression: example



Regression: example



- In linear regression, we assume the model generating t is based on a linear combination of $\vec{x}$'s elements:

$$y(\vec{x}, \vec{w}) = w_o + w_1 x_1 + \ldots + w_D x_D \quad (53)$$

$$= w_0 + \sum_{d=1}^{D} w_d x_d \quad (54)$$

- Here, $\vec{w}$ is a vector of weights defining the parameters of the model.

- We want to evaluate the performance of a given y.
- To this end, we need an error function (aka loss function).
- Typical error functions include
  - square error:

$$E(t_i, y(\vec{x}_i, \vec{w})) = \frac{1}{2}(t_i - y(\vec{x}_i, \vec{w}))^2 \tag{55}$$

  - linear error:

$$E(t_i, y(\vec{x}_i, \vec{w})) = |(t_i - y(\vec{x}_i, \vec{w}))| \tag{56}$$

- The total (or mean) error is

$$E(t, y(\vec{x}, \vec{w})) = \frac{1}{N}\sum_{i=1}^{N} E(t_i, y(\vec{x}_i, \vec{w})). \tag{57}$$

- **Assume we were able to estimate the likelihood (probability) of a target t given the input vector $\vec{x}$ and some model parameters $\vec{w}$:**

$$p(t|\vec{x},\vec{w}). \tag{58}$$

- **Then, we could optimize our model by maximizing the likelihood over all possible parameterizations $\vec{w}$.**
- **Assume, we can partially differentiate p with respect the dimensions of $\vec{w}$, then we could find a closed-form solution of $\vec{w}$'s optimum by setting the gradient to zero.**
- **Assuming further that the likelihood follows a Gaussian distribution, i.e., if there were infinitely many targets for a given input vector x, they would be normally distributed around a mean with a standard deviation σ.**

- **So, the likelihood of a target given a data point and a model is**

$$p(t_i|\vec{x}_i, \vec{w}) = \mathcal{N}(t_i|y(\vec{x}_i, \vec{w}), \sigma^2). \tag{59}$$

- **Assuming data points are independent and identically distributed (iid), we can write the likelihood of the entire observed data given a model as**

$$\begin{aligned} p(t|\vec{x}, \vec{w}) &= \prod_{i=1}^{N} \mathcal{N}(t_i|y(\vec{x}_i, \vec{w}), \sigma^2). \\ &= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y(\vec{x}_i, \vec{w}) - t_i)^2}{2\sigma^2}\right). \end{aligned} \tag{60}$$

- For further considerations, we want to look at the log likelihood rather than the likelihood itself.
- This is to resolve the product of powers into a sum.
- Since the natural logarithm is a monotonous function, $\ln(p(x))$ has the same location of extrema as $p(x)$ given that $p(x) > 0$ (exercise: show this!).
- This yields

$$\begin{aligned} \ln(p(t|\vec{x},\vec{w})) &= \ln \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y(\vec{x}_i,\vec{w})-t_i)^2}{2\sigma^2}\right) \\ &= -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y(\vec{x}_i,\vec{w})-t_i)^2 - N\ln\sqrt{2\pi\sigma^2} \quad (61) \end{aligned}$$

- To determine $\ln(p)$'s gradient, we apply the del operator:

$$\begin{aligned}
\nabla_{\vec{w}} \ln(p(t|\vec{x},\vec{w})) &= \nabla_{\vec{w}} \left(-\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y(\vec{x}_i,\vec{w}) - t_i)^2 - N \ln \sqrt{2\pi\sigma^2} \right) \\
&= -\frac{1}{2\sigma^2} \nabla_{\vec{w}} \left(\sum_{i=1}^{N} (y(\vec{x}_i,\vec{w}) - t_i)^2 \right) \\
&= -\frac{1}{2\sigma^2} \nabla_{\vec{w}} R(\vec{w}). \qquad (62)
\end{aligned}$$

- $R(\vec{w})$ is called the empirical risk.
- To determine the maximum, we set the gradient to zero:

$$\nabla_{\vec{w}} \ln(p(t|\vec{x},\vec{w})) = \vec{0}. \qquad (63)$$

- This shows that maximizing likelihood (assuming Gaussian distribution of values) is identical to minimum mean square error (MMSE).

- **Assuming $\vec{x}$ to be one-dimensional Eq. 54 yields**

$$y(\vec{x}_i, \vec{w}) = w_0 + w_1 x_{1,i}. \tag{64}$$

- **Considering the first dimension of $\vec{w}$ in Eq. 63, we have**

$$\begin{aligned} 0 &= -\frac{1}{2\sigma^2} \sum_{i=1}^{N} (w_0 + w_1 x_{1,i} - t_i)^2 \frac{\partial}{\partial w_0}; \qquad (65) \\ 0 &= \sum_{i=1}^{N} (w_0 + w_1 x_{1,i} - t_i) \\ &= N w_0 + w_1 \sum_{i=1}^{N} x_{1,i} - \sum_{i=1}^{N} t_i. \qquad (66) \end{aligned}$$

- **So, we have**

$$\begin{aligned} w_0 &= -w_1 \frac{1}{N} \sum_{i=1}^{N} x_{1,i} + \frac{1}{N} \sum_{i=1}^{N} t_i \\ &= a_0 w_1 + b_0. \qquad (67) \end{aligned}$$

- **Considering the second dimension of $\vec{w}$, we have**

$$\begin{aligned} 0 &= -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(w_0 + w_1 x_{1,i} - t_i)^2 \frac{\partial}{\partial w_1}; \quad (68) \\ 0 &= \sum_{i=1}^{N}(w_0 + w_1 x_{1,i} - t_i)x_{1,i} \\ &= w_0 \sum_{i=1}^{N} x_{1,i} + w_1 \sum_{i=1}^{N} x_{1,i}^2 - \sum_{i=1}^{N} t_i x_{1,i} \quad (69) \end{aligned}$$

- **So, we have**

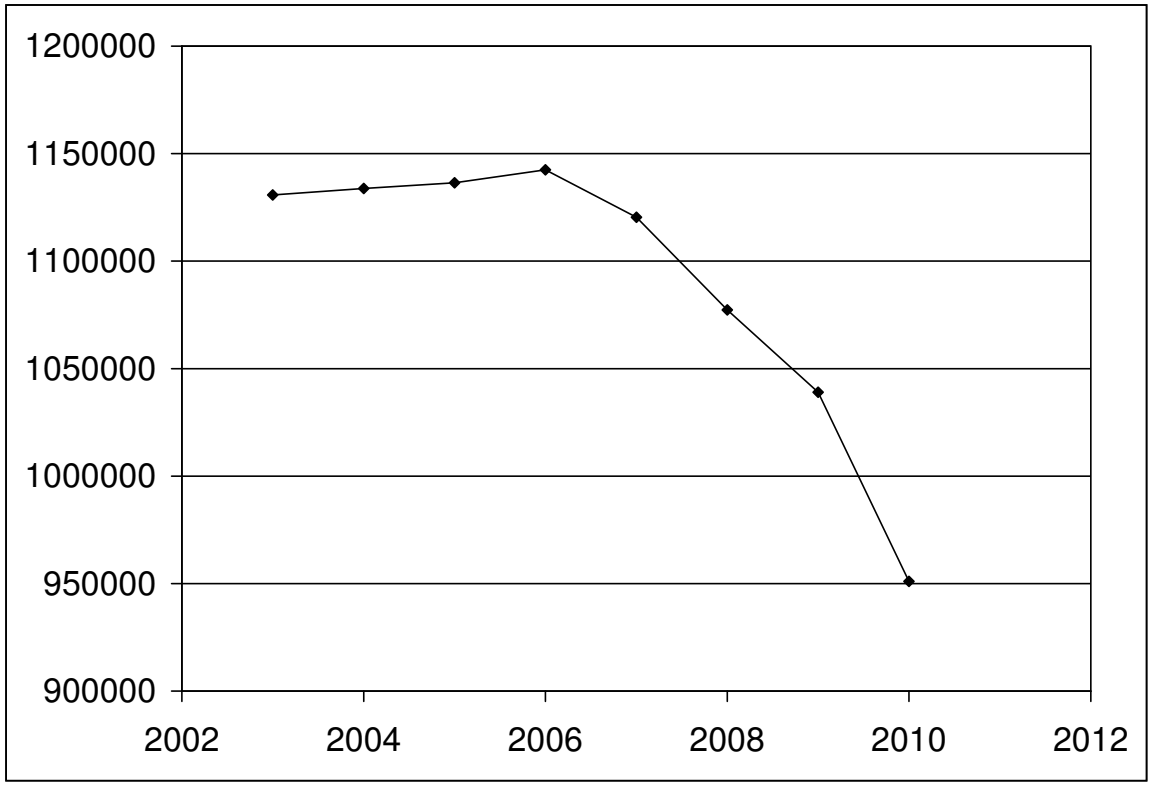
$$\begin{aligned} w_1 &= -w_0 \frac{\sum_{i=1}^{N} x_{1,i}}{\sum_{i=1}^{N} x_{1,i}^2} + \frac{\sum_{i=1}^{N} t_i x_{1,i}}{\sum_{i=1}^{N} x_{1,i}^2} \\ &= a_1 w_0 + b_1 \\ &= a_1(a_0 w_1 + b_0) + b_1 \end{aligned} \tag{70}$$

- **This finally leads to**

$$w_1 = \frac{a_1 b_0 + b_1}{1 - a_1 a_0}. \tag{71}$$

- **The six-month average daily circulation of the New York Times (period ending in March) for the last 8 years is**

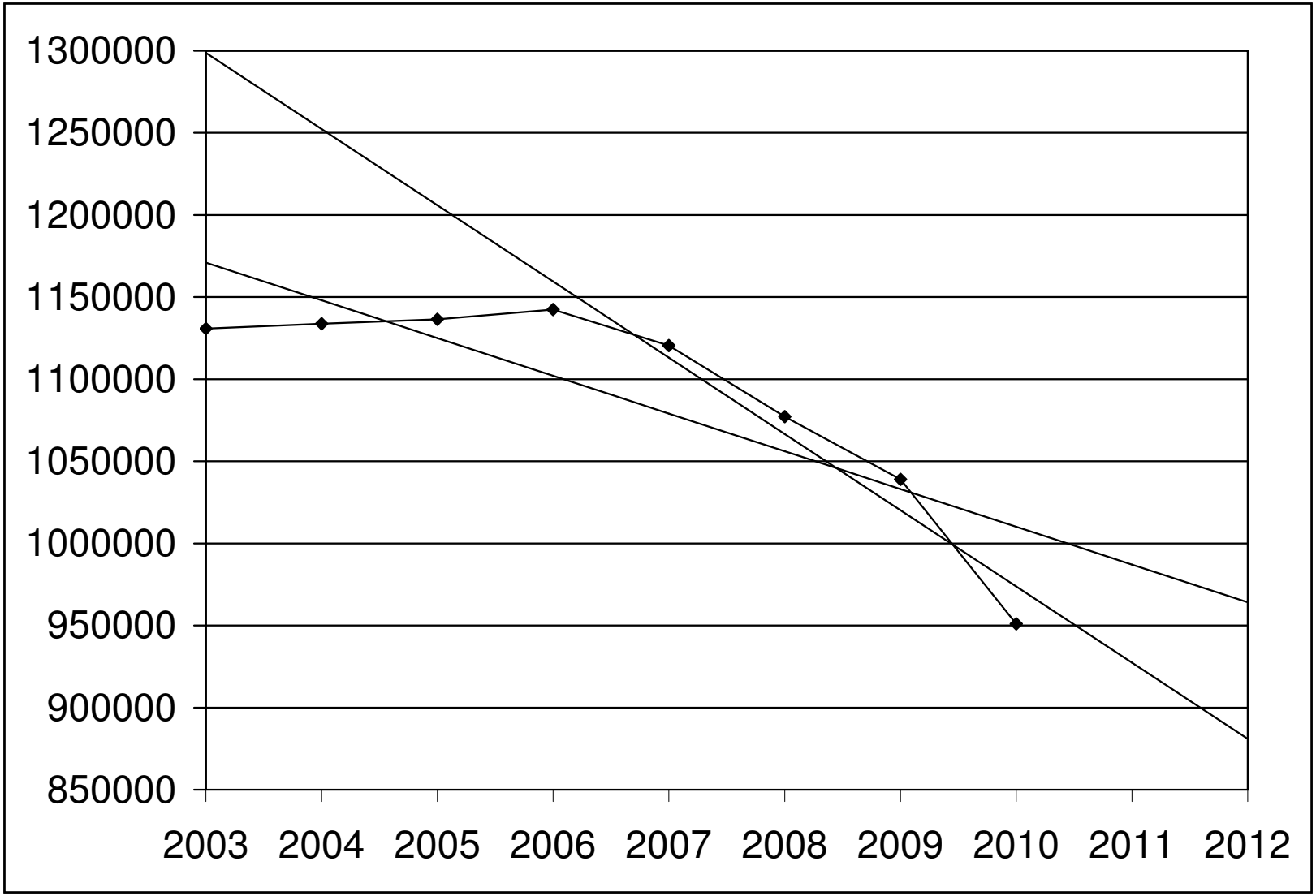
year	avg
2003	1,130,740
2004	1,133,763
2005	1,136,433
2006	1,142,464
2007	1,120,420
2008	1,077,256
2009	1,039,031
2010	951,063



a) What is the expected circulation for the same period in 2012?

b) How does the result change when you look only 5 years back?

One dimension: exercise (cont.)



- **Using Eqs. 62 and 54, we can write**

$$\begin{aligned} R(\vec{w}) &= \sum_{i=1}^{N} \left(t_i - w_0 - \sum_{d=1}^{D} w_d x_{d,i} \right)^2 \\ &= \left|\left| \begin{pmatrix} t_1 \\ \vdots \\ t_N \end{pmatrix} - \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{D,1} \\ \vdots & \vdots & & \vdots \\ 1 & x_{1,N} & \cdots & x_{D,N} \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_D \end{pmatrix} \right|\right|^2 \\ &= ||\vec{t} - \underline{X}\vec{w}||^2 \end{aligned} \tag{72}$$

- **According to Eq. 63, we set the gradient to zero:**

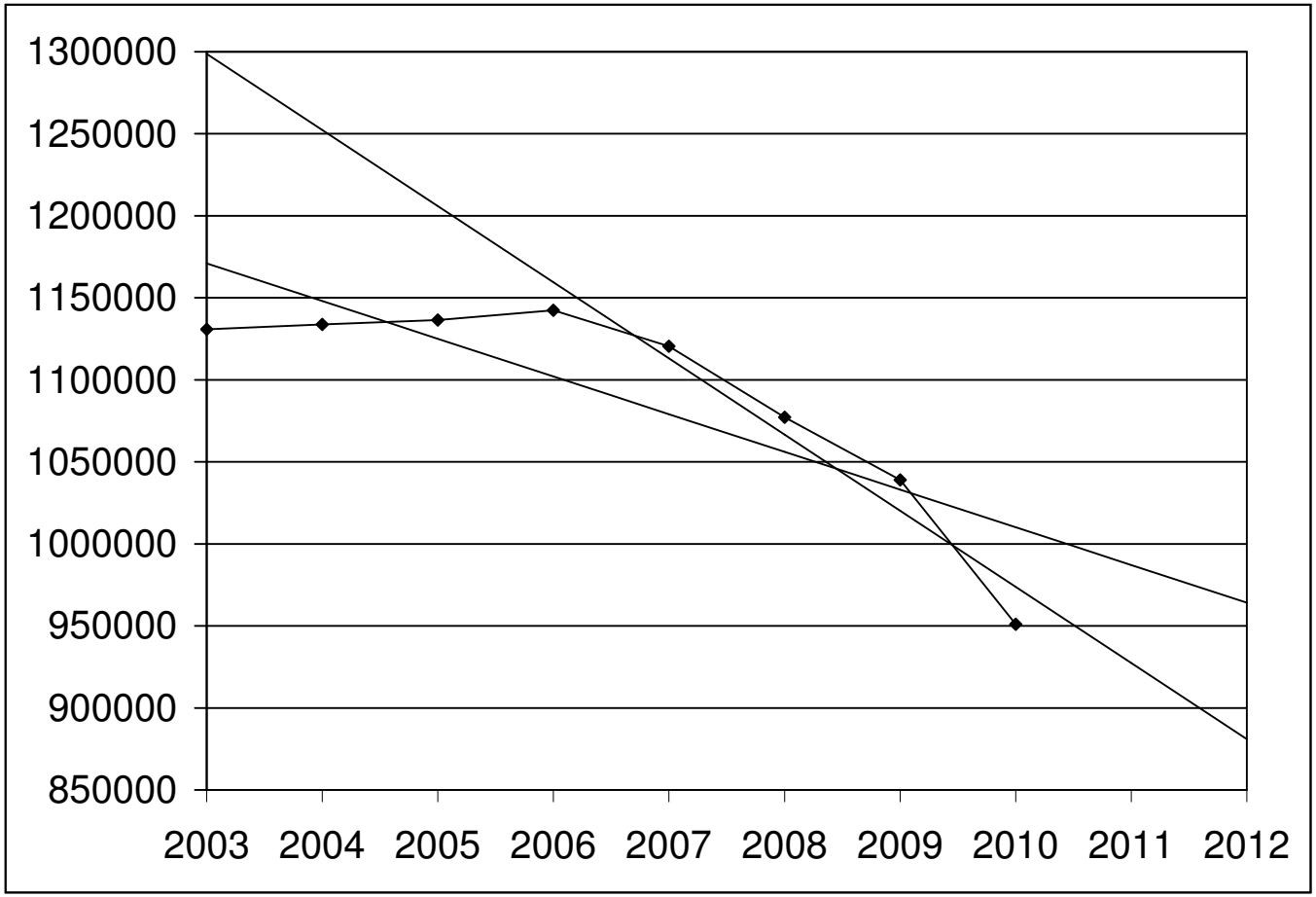
$$\begin{aligned}
\vec{0} &= \nabla_{\vec{w}} \ln(p(t|\vec{x}, \vec{w})); && (73)\\
\vec{0} &= \nabla_{\vec{w}} R(\vec{w}) \\
&= \nabla_{\vec{w}} ||\vec{t} - \underline{X}\vec{w}||^2 \\
&= \nabla_{\vec{w}} (\vec{t} - \underline{X}\vec{w})^\top (\vec{t} - \underline{X}\vec{w}) \\
&= \nabla_{\vec{w}} \vec{t}^{\,\top}\vec{t} - \vec{t}^{\,\top}\underline{X}\vec{w} - (\underline{X}\vec{w})^\top \vec{t} + (\underline{X}\vec{w})^\top \underline{X}\vec{w} \\
&= \nabla_{\vec{w}} \vec{t}^{\,\top}\vec{t} - 2\vec{w}^\top \underline{X}^\top \vec{t} + \vec{w}^\top \underline{X}^\top \underline{X}\vec{w} \\
&= -2\underline{X}^\top \vec{t} + 2\underline{X}^\top \underline{X}\vec{w}. && (74)
\end{aligned}$$

- **Solving this equation for $\vec{w}$ yields**

$$\vec{w} = (\underline{X}^\top \underline{X})^{-1} \underline{X}^\top \vec{t}. \quad (75)$$

- **Solve the one-dimensional linear regression exercise (a and b) using the generic approach.**

- **Often, the relationship between data points and target is not represented well by a linear function:**



- Here, a possible solution could be a polynomial regression function where the model depends on a linear combination of powers of a data point.
- In the case of one dimension, the regression model is (similar to Eq. 54)

$$y(x, \vec{w}) = w_0 + \sum_{d=1}^{D} w_d x^d = \sum_{d=0}^{D} w_d x^d. \tag{76}$$

- Accordingly, the empirical risk is

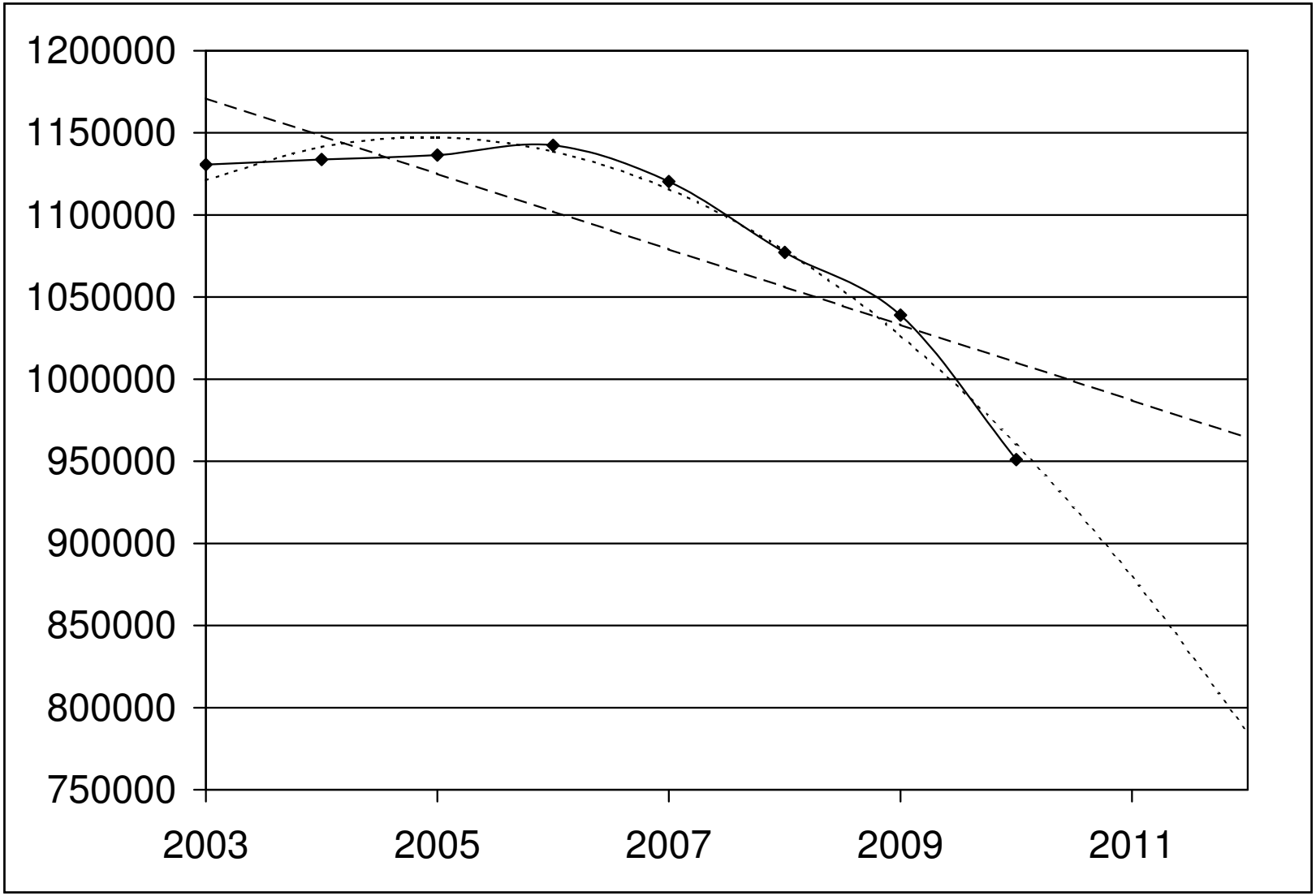
$$\begin{aligned} R(\vec{w}) &= \left\| \begin{pmatrix} t_1 \\ \vdots \\ t_N \end{pmatrix} - \begin{pmatrix} 1 & x_1 & \cdots & x_1^D \\ \vdots & \vdots & & \vdots \\ 1 & x_N & \cdots & x_N^D \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_D \end{pmatrix} \right\|^2 \\ &= ||\vec{t} - \underline{X}\vec{w}||^2 \end{aligned} \tag{77}$$

- Hence, to perform polynomial regression in one dimension, we set

$$x_{d,i} = x_{1,i}^d. \tag{78}$$

- **Solve the one-dimensional regression exercise (a) using the polynomial approach for $D = 2$.**

Polynomial regression: exercise (cont.)

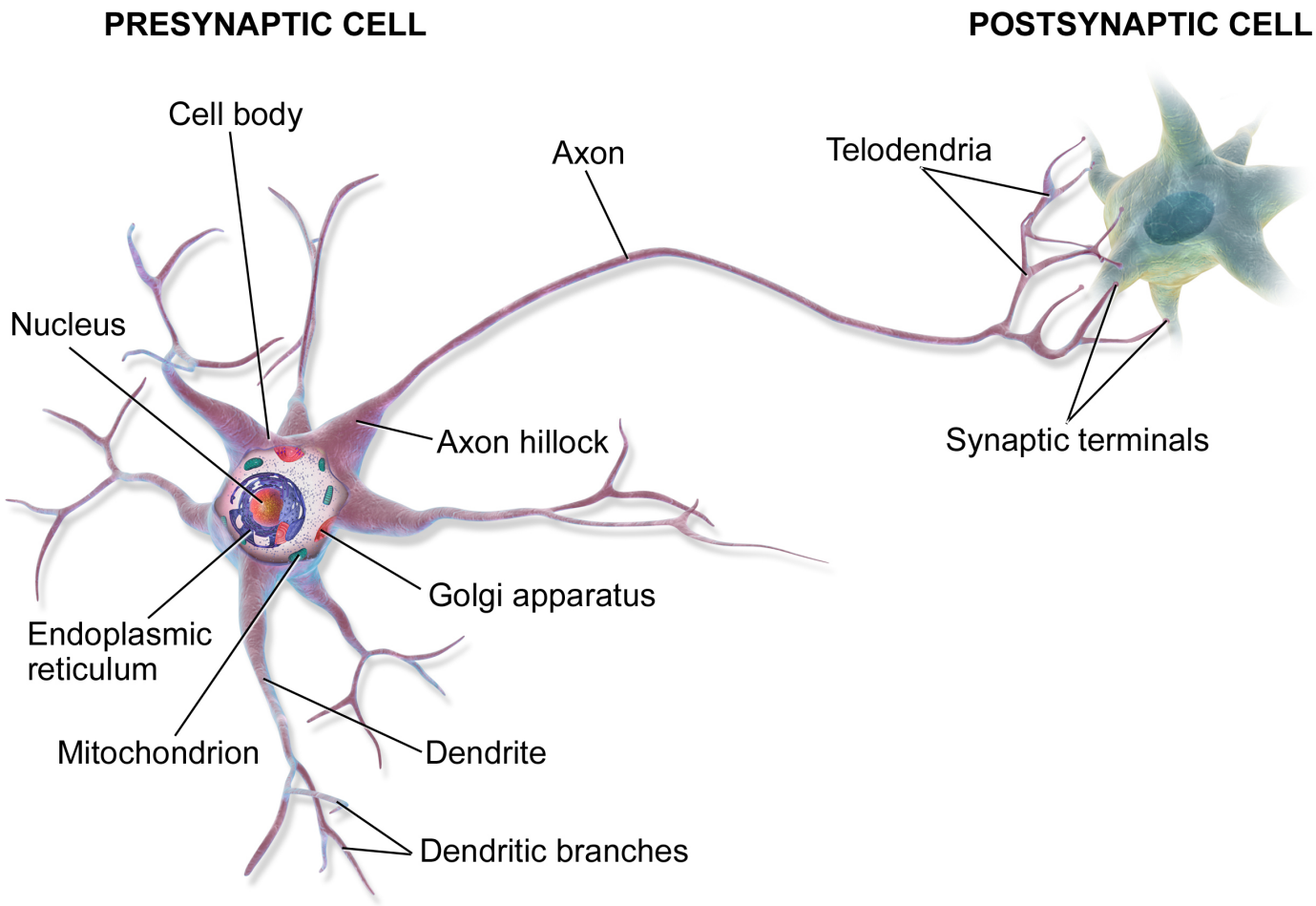


- Polynomial regression is also linear since it is linear in its parameters.
- Generally, we can use an arbitrary set of functions $f_d : \mathbb{R} \to \mathbb{R}$ such that

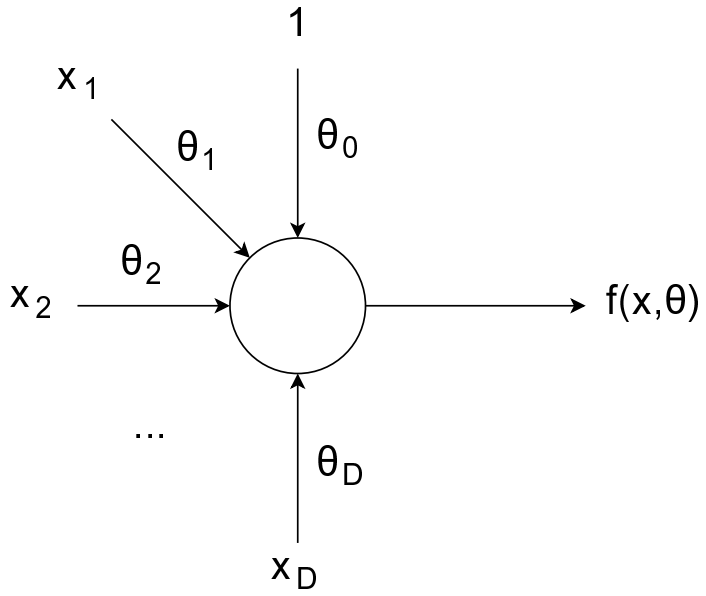
$$y(x, \vec{w}) = \sum_{d=0}^{D} w_d f_d(x). \tag{79}$$

- These function are called basis functions, defining the bases of the feature space.
- Popular basis functions include
  - polynomials,
  - Gaussians,
  - sigmoids,
  - sinusoids.

1. introduction
2. probability and statistics
3. linear regression
4. neural networks
5. Bayesian networks
6. hidden-Markov models
7. decision trees
8. boosting
9. homework

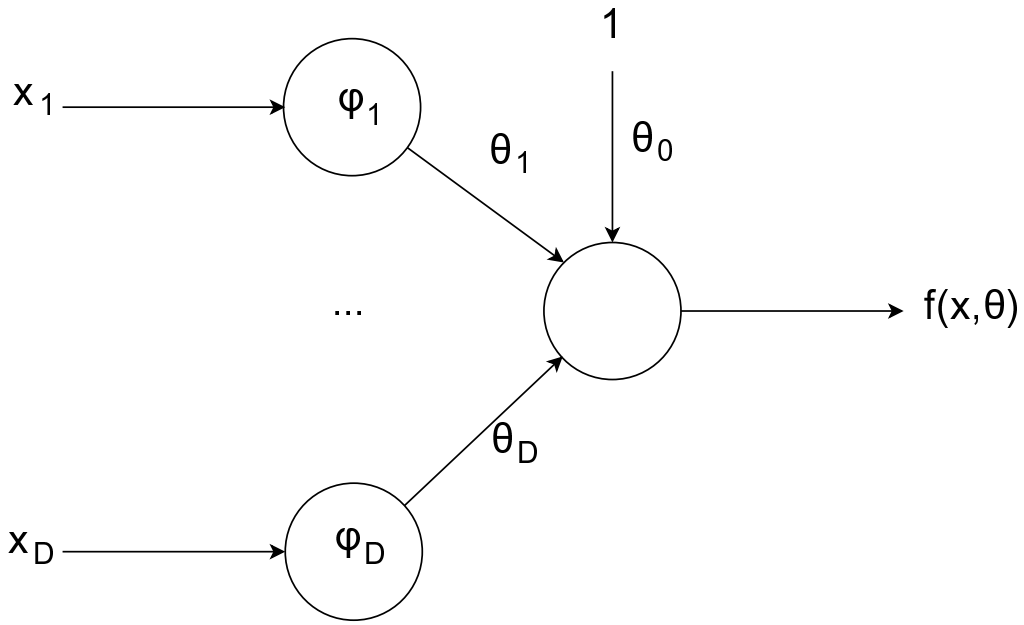


The Anatomy of a Multipolar Neuron



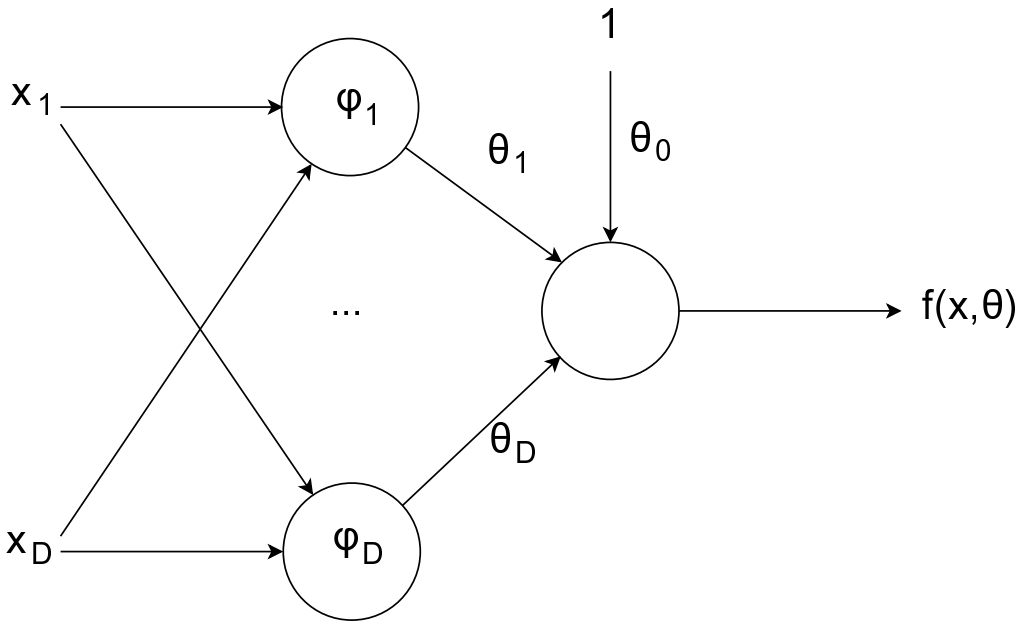
- **Dentrites weight the signal x_d with θ_d.**
- **Weighted signals are accumulated by the node.**
- **linear-regression-like sum formula:**

$$f(\vec{x}, \vec{\theta}) = \theta_0 + \sum_{d=1}^{D} \theta_d x_d \tag{80}$$



- **additional application of feature (basis) functions**
- **linear-regression-like sum formula:**

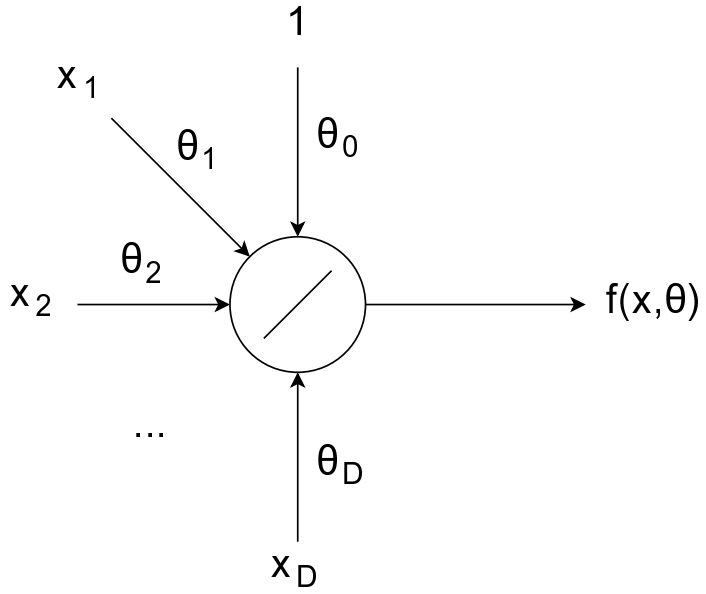
$$f(x,\vec{\theta}) = \theta_0 + \sum_{d=1}^{D} \theta_d \phi_d(x) \tag{81}$$



- **extension to multiple features**

$$f(\vec{x}, \vec{\theta}) = \theta_0 + \sum_{i=1}^{D} \sum_{d=1}^{D} \theta_d \phi_d(x_i) \tag{82}$$

- **The dimensionality of $\vec{x}$ can also be different from that of $\vec{\theta}$.**



- **The linear-regression-like sum formula can also be written as**

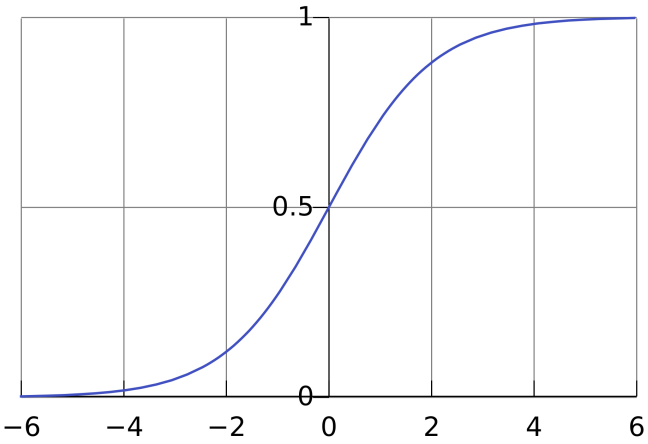
$$f(\vec{x}, \vec{\theta}) = \vec{\theta}^{\top} \vec{x}. \tag{83}$$

- **Artificial neurons often use a combining function g rather than the sum itself:**

$$f(\vec{x}, \vec{\theta}) = g(\vec{\theta}^{\top} \vec{x}). \tag{84}$$

- **In the original version of the neuron, we have $g(z) = z$.**

- **Very popular is the sigmoid, logistic, or squashing function for "soft" (binary) classification**



$$g(z) = \frac{1}{1+e^{-z}}. \tag{85}$$

- **Minimizing the empirical risk:**

$$R(\vec{\theta}) = \sum_{i=1}^{N} (t_i - g(\vec{\theta}^\top \vec{x}_i))^2 \tag{86}$$

$$\nabla_{\vec{\theta}} R = \sum_{i=1}^{N} 2(t_i - g(\vec{\theta}^\top \vec{x}_i))(-1) g'(\vec{\theta}^\top \vec{x}_i)\vec{x}_i = \vec{0}$$

$$g'(z) = g(z)(1-g(z)) \qquad \textbf{exercise: prove this!}$$

- Unfortunately, $R(\vec{\theta}) = \vec{0}$ has no closed-form solution.
- However, it is a convex function.
- I.e., there are no local minima.
- Hence, the gradient descent algorithm can be applied (show examples!) with

$$\begin{aligned} \vec{\theta_0} &= \text{rand} \\ \vec{\theta}_{i+1} &= \vec{\theta}_i - \eta \nabla_{\vec{\theta}} R|_{\vec{\theta}_i} \end{aligned} \tag{87}$$

- Possible convergence problems:
  - oszillation around minimum when η is too large,
  - stalling when gradient is $\vec{0}$ at a non-minimum location (plateaus, ridges, valleys)

- A perceptron uses the "hard" (binary) classification squashing function

$$g(z) = \begin{cases} -1 & : \quad z < 0 \\ 1 & : \quad z \geq 0 \end{cases} . \tag{88}$$

- The empirical classification risk is

$$R_2(\vec{\theta}) = \sum_{i=1}^{N} 1 - \delta(t_i, \mathbf{sign}(\vec{\theta}^\top \vec{x}_i)) \tag{89}$$

with the Kronecker function

$$\delta(x, y) = \begin{cases} 1 & : \quad x = y \\ 0 & : \quad x \neq y \end{cases} . \tag{90}$$

- As this risk function is not monotonous, it is useful to use the "confidence" of the misclassification:

$$R_3(\vec{\theta}) = \sum_{i=1}^{N} \mathbf{max}(0, -t_i \vec{\theta}^\top \vec{x}_i). \tag{91}$$

- **Instead of calculating gradients over the entire training data, update the gradient for each misclassified point:**

$$\begin{aligned} \vec{\theta}_{i+1} &= \vec{\theta}_i - \eta \nabla_{\vec{\theta}} R|_{\vec{\theta}_i} \\ &= \vec{\theta}_i - \eta \nabla_{\vec{\theta}} (-t_j \vec{\theta}^\top \vec{x}_j)|_{\vec{\theta}_i} \\ &= \vec{\theta}_i + \eta t_j \vec{x}_j. \end{aligned} \tag{92}$$

- **This update needs to be repeated until a certain threshold for R_2 or i is reached.**

- **It was shown that if classes are linearly separable in $\vec{x}$ space (show examples!) then gradient descent will converge to a solution producing zero error (in terms of R_2) on the training data.**

- **We are given the training data**

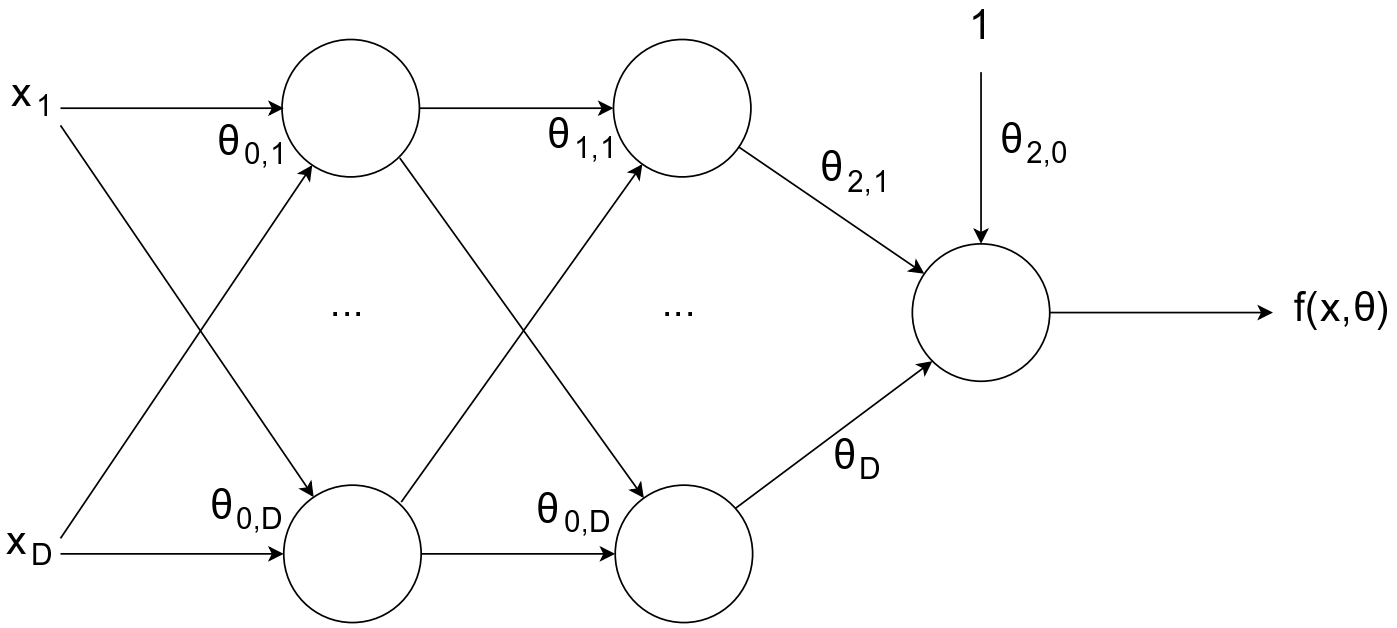
$$\begin{aligned} \vec{x}_1^2 &= (1\ 1)^\top, (3\ 3)^\top, \\ t_1^2 &= -1, 1. \end{aligned} \tag{93}$$

- **For online perceptron training, we assume that**

$$\begin{aligned} \theta_0 &= (0\ 0\ 0)^\top \\ \eta &= 1. \end{aligned} \tag{94}$$

- **These are the steps the algorithm takes before R_2 becomes zero:**

$$\begin{aligned}
\boldsymbol{\theta_0} &= (0\ 0\ 0)^\top & \boldsymbol{R_2(\theta_0)} &= 1 \\
\boldsymbol{\theta_1} &= (-1\ -1\ -1)^\top & \boldsymbol{R_2(\theta_1)} &= 1 \\
\boldsymbol{\theta_2} &= (0\ 2\ 2)^\top & \boldsymbol{R_2(\theta_2)} &= 1 \\
\boldsymbol{\theta_3} &= (-1\ 1\ 1)^\top & \boldsymbol{R_2(\theta_3)} &= 1 \\
\boldsymbol{\theta_4} &= (-2\ 0\ 0)^\top & \boldsymbol{R_2(\theta_4)} &= 1 \\
\boldsymbol{\theta_5} &= (-1\ 3\ 3)^\top & \boldsymbol{R_2(\theta_5)} &= 1 \\
\boldsymbol{\theta_6} &= (-2\ 2\ 2)^\top & \boldsymbol{R_2(\theta_6)} &= 1 \\
\boldsymbol{\theta_7} &= (-3\ 1\ 1)^\top & \boldsymbol{R_2(\theta_7)} &= 0
\end{aligned} \tag{95}$$

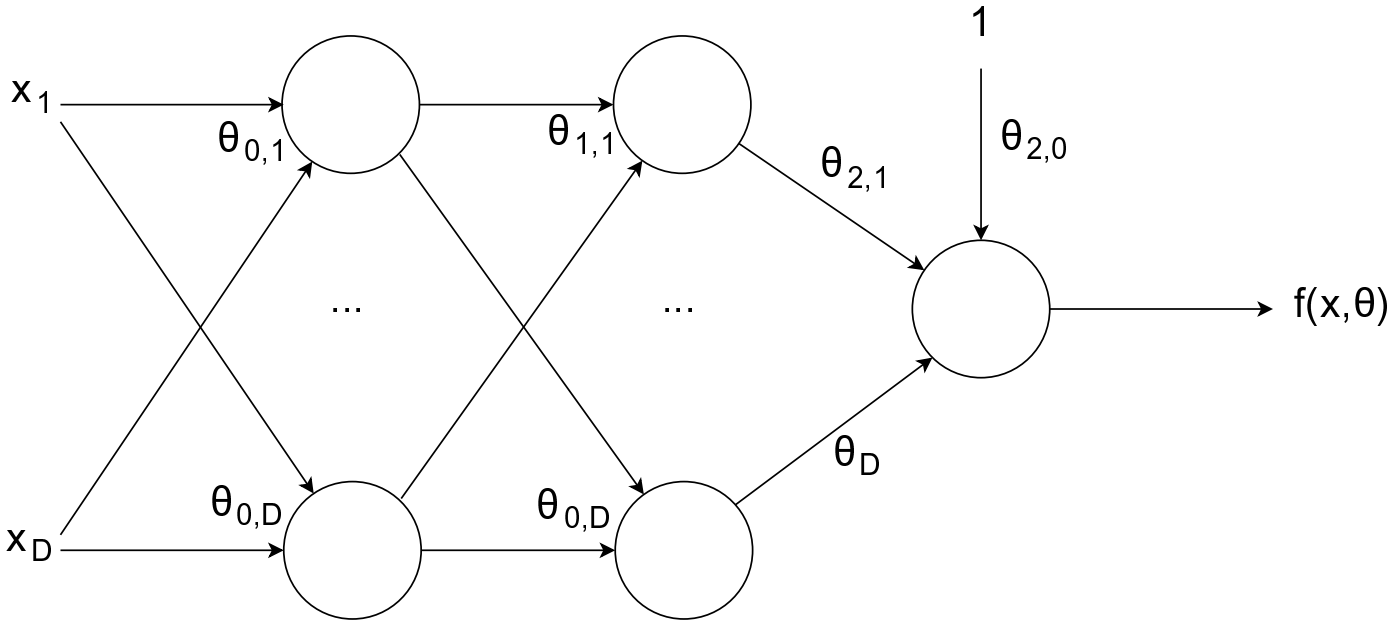


- **cascaded neurons**
- **linear-regression-like multilayer networks are no different from single neurons:**

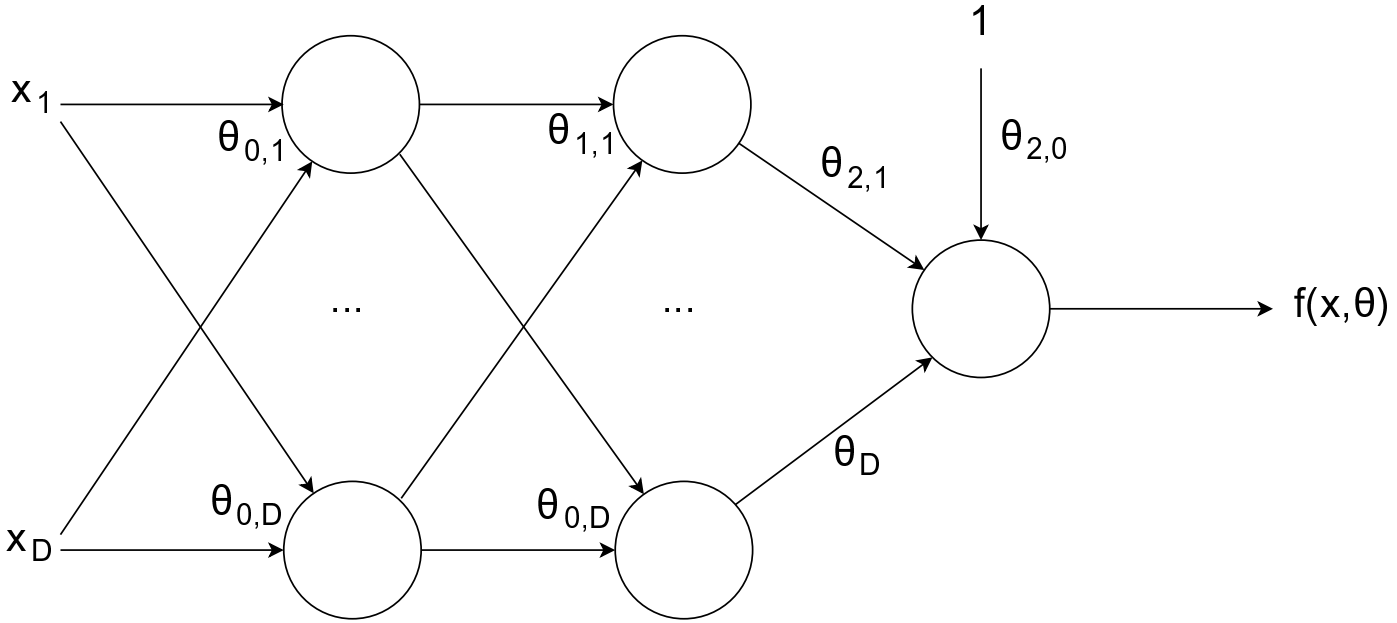
$$f(x, \vec{\theta}_{0,0}^{\,D}, \vec{\theta}_{1,0}^{\,D}, \vec{\theta}_2) = \sum_{i=1}^{D} \theta_{2,i} \sum_{j=1}^{D} \theta_{1,i,j} \sum_{k=1}^{D} \theta_{0,j,k} x_k \quad \dots \tag{96}$$

$$
\begin{aligned}
f(x, \vec{\theta}_{0,0}^{\,D}, \vec{\theta}_{1,0}^{\,D}, \vec{\theta}_2) &= \sum_{i=1}^{D} \theta_{2,i} \sum_{j=1}^{D} \theta_{1,i,j} \sum_{k=1}^{D} \theta_{0,j,k} x_k \\
&= \sum_{i=1}^{D} \theta_{2,i} \sum_{j=1}^{D} \theta_{1,i,j} (\vec{\theta}_{0,j}^{\top} \vec{x}) \\
&= \left(\sum_{i=1}^{D} \theta_{2,i} \sum_{j=1}^{D} \theta_{1,i,j} \vec{\theta}_{0,j}^{\top} \right) \vec{x} \\
&= \vec{\theta'}^{\top} \vec{x} \qquad (97)
\end{aligned}
$$

- Using a single perceptron allows for linear separation of data.
- Using a layer of sigmoids combined with single perceptron allows for representing convex hulls.
- More than two layers allow for representing more complex shapes.
- Gradient descent must be performed over the entire network.
- approach: error backpropagation



$$
\begin{aligned}
\boldsymbol{R} &= \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} (\boldsymbol{t_i} - \boldsymbol{f(\vec{x}_i)})^{\boldsymbol{2}} \qquad (98)\\
&= \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \left(\boldsymbol{t_i} - \boldsymbol{g} \left(\sum_{\boldsymbol{i=1}}^{\boldsymbol{D}} \boldsymbol{\theta_{2,i}} \boldsymbol{g} \left(\sum_{\boldsymbol{j=1}}^{\boldsymbol{D}} \boldsymbol{\theta_{1,i,j}} \boldsymbol{g} \left(\sum_{\boldsymbol{k=1}}^{\boldsymbol{D}} \boldsymbol{\theta_{0,j,k}} \boldsymbol{x_k} \right) \right) \right) \right)^{\boldsymbol{2}}
\end{aligned}
$$



- **Partial gradient update:**

$$\vec{\theta}^{i+1}_{j,k} = \vec{\theta}^{i}_{j,k} - \eta \nabla_{\vec{\theta}_{j,k}} R|_{\vec{\theta}^{i}_{j,k}} \tag{99}$$

1. introduction
2. probability and statistics
3. linear regression
4. neural networks
5. Bayesian networks
6. hidden-Markov models
7. decision trees
8. boosting
9. homework

- Taking a detour...
- Assume, we are looking at joint probabilities of D different (discrete) random variables (aka multinomial joint probability).

$$p(x_1, \ldots, x_D). \tag{100}$$

- E.g., in the introduction to probability theory, we looked at the variables B (box) and F (fruit), i.e., we had $D = 2$.
- Let us also assume, x_d can assume V values (e.g., the vocabulary size in an NLP application).
- The size of the count table in this case is V^D.
- For growing numbers of variables, the table becomes extremely large and sparse when estimated on training data (e.g. when estimating the probability of a sentence with D words).

- **If all variables were independent, we had**

$$p(x_1, \ldots, x_D) = p(x_1)p(x_2)\cdots p(x_D) = \prod_{d=1}^{D} p(x_d). \tag{101}$$

- **In this case, the count table could be reduced to a size of VD.**
- **Independence assumption makes things convenient and simple (naïve Bayes).**
- **However, in real world, some variables depend on each other, others do not.**

- **Furthermore, some variables are conditionally independent,**
- **Two events x and y are conditionally independent given z, iff given knowledge of whether z, knowledge of whether x provides no information on the probability of y and vice versa.**
- **Formally, we have:**

$$p(x, y|z) = p(x|z)p(y|z) \tag{102}$$

however,

$$p(x, y) \neq p(x)p(y). \tag{103}$$

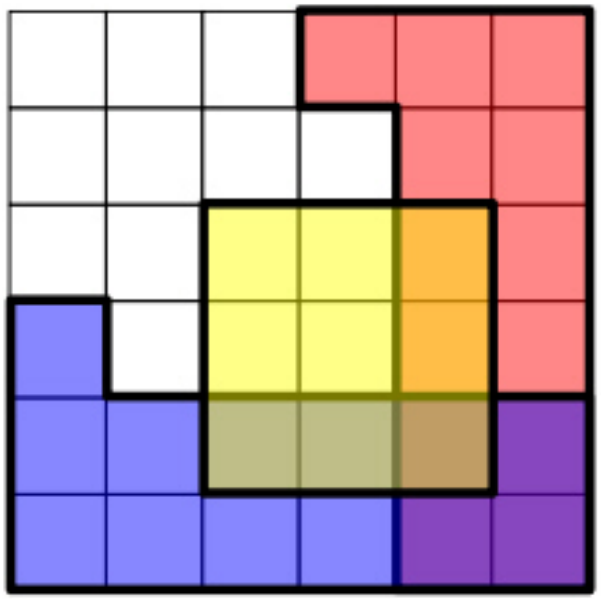
- **We also use the formalisms**

$$(x \perp y)|z \quad \text{or} \quad x \perp y|z \tag{104}$$

read as "(x is independent of y) given z".

- **To check for conditional independence means to prove Eq. 102 holds true.**

- **Example (binary/Boolean random variables):**
  - **$y$ellow: dry spell**
  - **$r$ed: sunburn**
  - **$b$lue: forest fire**



  - **pic:**
    - **source:** http://en.wikipedia.org/wiki/File:Conditional_independence.svg
    - **author: AzaToth**
    - **license: Creative Commons Attribution-Share Alike**

- **Determine if $r \perp b|y$.**

- **If**

$$x \perp y|z, \tag{105}$$

we can express the joint probability of x, y, and z as

$$\begin{aligned} p(x,y,z) &= p(x,y|z)p(z) \\ &= p(x|z)p(y|z)p(z). \end{aligned} \tag{106}$$

- **If**

$$x \not\perp y|z, \tag{107}$$

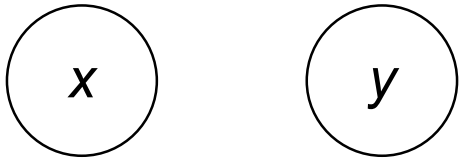
we can express the joint probability of x, y, and z as

$$\begin{aligned} p(x,y,z) &= p(x,y|z)p(z) \\ &= p(x|y,z)p(y|z)p(z). \end{aligned} \tag{108}$$

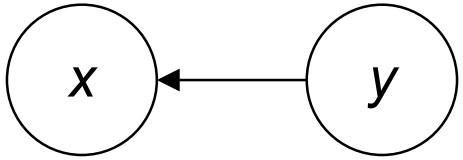
- **Control Eqs. 106 and 108 for $r \perp b|y$ of the above exercise.**

- **Bayesian networks (aka graphical models, belief networks) are a visual representation of dependences between random variables using graphs.**
- **Nodes are random variables.**
- **Arcs are dependences (i.e., missing arcs indicate independence).**
- **The arc's direction indicates causality:**
  - **source: trigger, parent;**
  - **destination: response.**

- **independence:** $x \perp y \quad \Longleftrightarrow \quad p(x,y) = p(x)p(y)$



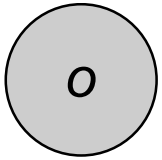
- **dependence:** $x \not\perp y \quad \Longleftrightarrow \quad p(x,y) = p(x|y)p(y)$



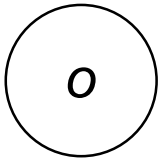
- **Referring to the parent(s) of a node x_i as $x_{\mathrm{pa}(i)}$, we can write the joint probability of all the variables in a Bayesian network as**

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i|x_{\mathrm{pa}(i)}). \tag{109}$$

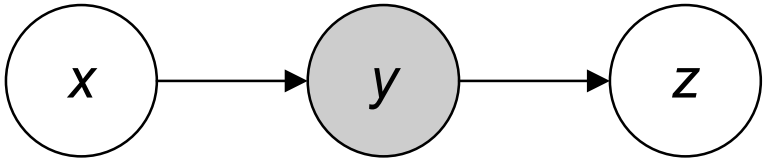
- In probability theory, we refer to conditioning as the idea that beliefs depend on the availability of information.
- In the example on boxes and fruit, we saw that the probability of the event b (blue box) depended on whether we had observed the event o (orange) [posterior] or not [prior].
- Accordingly, we say that o is
  - observed



  - hidden

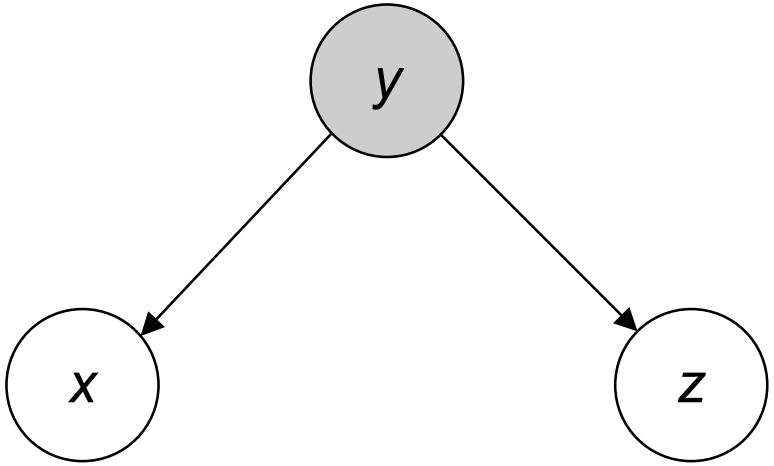


- We are given the following Markov chain



- Example: $x =$ overcast? $y =$ rain? $z =$ wet floor?
- Now, we want to determine whether $x \perp z|y$ using Eq. 102

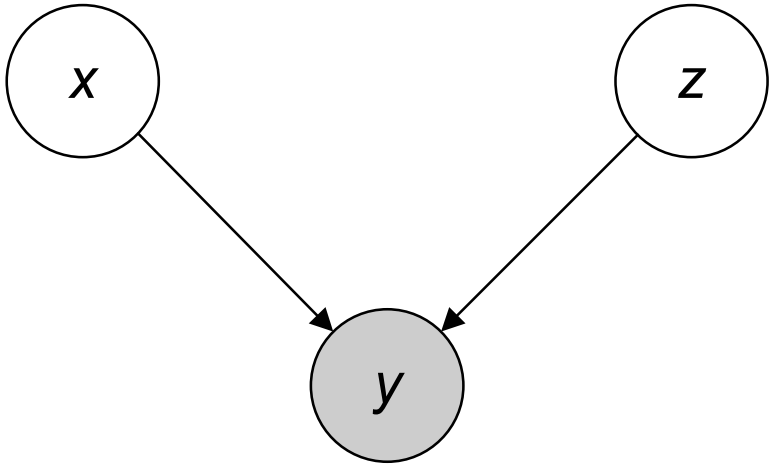
$$\begin{aligned} p(x,z|y) &= \frac{p(x,y,z)}{p(y)} \\ &= \frac{\prod_{x_i \in \{x,y,z\}} p(x_i|x_{\mathbf{pa}(i)})}{p(y)} \\ &= \frac{p(x)p(y|x)p(z|y)}{p(y)} \\ &= \frac{p(x)p(y,x)p(z,y)}{p(y)p(x)p(y)} \\ &= p(x|y)p(z|y) \quad \Longleftrightarrow \quad x \perp z|y. \end{aligned} \tag{110}$$



- Example: $y =$ sick? $x =$ fever? $z =$ headache?
- Determine whether $x \perp z|y$ using Eq. 102

$$\begin{aligned} p(x,z|y) &= \frac{p(x,y,z)}{p(y)} \\ &= \frac{\prod\limits_{x_i \in \{x,y,z\}} p(x_i|x_{\mathbf{pa}(i)})}{p(y)} \\ &= \frac{p(y)p(x|y)p(z|y)}{p(y)} \\ &= p(x|y)p(z|y) \quad \Longleftrightarrow \quad x \perp z|y. \end{aligned} \tag{111}$$

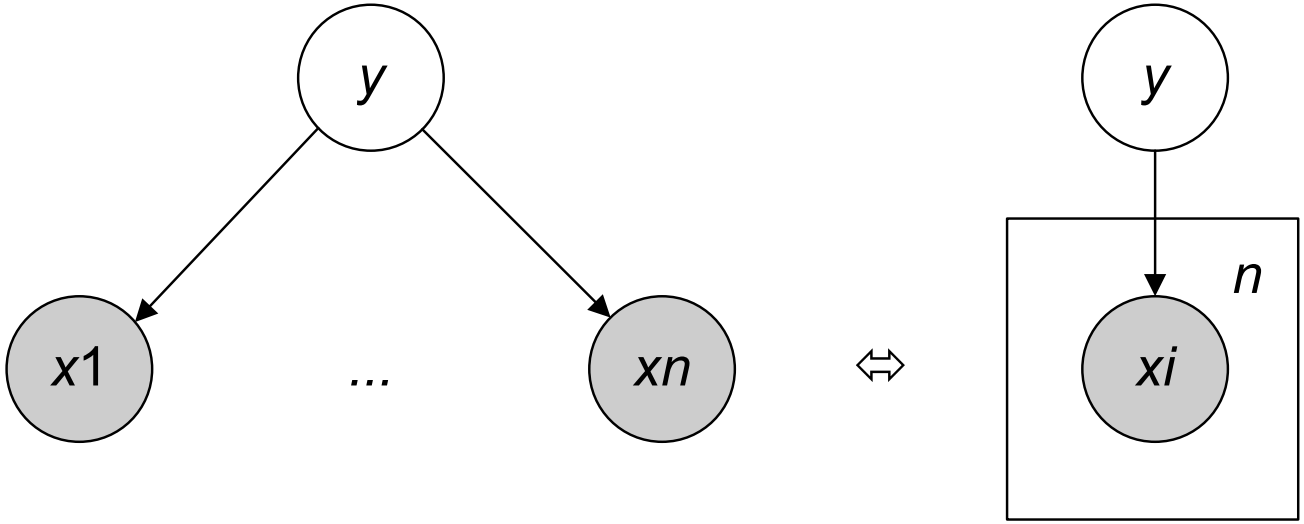
- **We are given the following graph**



- **Example: $x =$ spilled coffee? $z =$ rain? $y =$ wet floor?**
- **Determine whether $x \perp z|y$ using Eq. 102**

$$
\begin{aligned}
p(x,z|y) &= \frac{p(x,y,z)}{p(y)} \\
&= \frac{\prod\limits_{x_i \in \{x,y,z\}} p(x_i|x_{\mathbf{pa}(i)})}{p(y)} \\
&= \frac{p(x)p(y|x,z)p(z)}{p(y)} \\
&= \frac{p(x)p(y|x,z)p(z)}{p(y)} \cdot \frac{p(z,y)}{p(z,y)} \\
&= \frac{p(x)p(y|x,z)p(z)}{p(y)} \cdot \frac{p(z,y)}{\prod\limits_{x_i \in \{y,z\}} p(x_i|x_{\mathbf{pa}(i)})} \\
&= \frac{p(x)p(y|x,z)p(z)}{p(y)} \cdot \frac{p(z,y)}{p(y|x,z)p(z)} \\
&= p(x)p(z|y) \quad \Longleftrightarrow \quad x \not\perp z|y. \qquad (112)
\end{aligned}
$$

- **Graphical model of a naïve Bayes classifier:**



- **On the right: plate notation**
- **The observation variables x_i are independent given the class y (we showed this earlier).**

- **Determining the best class y given observations $x_1^n := x_1, \ldots, x_n$:**

$$
\begin{aligned}
\hat{y} &= \arg\max_y \; p(y|x_1^n) \\
&= \arg\max_y \; \frac{p(x_1^n, y)}{p(x_1^n)} \\
&= \arg\max_y \; p(x_1^n, y) \\
&= \arg\max_y \; \prod_{x_i \in \{x_1^n, y\}} p(x_i|x_{\mathbf{pa}(i)}) \\
&= \arg\max_y \; p(y) \prod_{i=1}^{n} p(x_i|y) \\
&= \arg\max_y \; \log\left(p(y) \prod_{i=1}^{n} p(x_i|y)\right) \\
&= \arg\max_y \; \log p(y) + \sum_{i=1}^{n} \log p(x_i|y)
\end{aligned}
$$

- **We are given 100 sample utterances and a semantic class for each utterance (describing its topic).**
- **For the sake of simplicity, the utterances are already converted into vector form (100 word presence vectors of dimensionality $D = 72$) stored in the Octave matrix `nb` (see *word.txt* for mapping to original data).**
- **The respective semantic classes are coded as integers between 1 and 13 and stored in the vector `nbc` (see *class.txt* for mapping to original data).**
- **All the data is available in the file `nb.m`. Using this data,**
  - **a) Build a naïve Bayes classifier using the entire data body.**
  - **b) Which class does the classifier return for the input utterance**
    i want to pay my bill?
  - **c) Test your classifier on the entire amount of available data.**
  - **d) Select 90 random utterances and train your classifier on this data. Test the classifier on the held-out test set. Repeat this exercise.**
  - **e) Perform ten-fold cross-validation.**

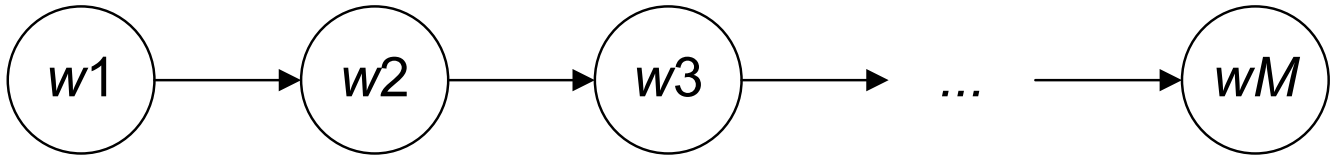
- A statistical language model (SLM) assigns a probability to a sequence of words $p(w_1^M)$.
- It is a crucial component in machine learning disciplines such as
  - speech recognition,
  - spoken language understanding,
  - machine translation,
  - syntactic tagging and parsing.
- Due to data sparseness, context is taken into account in a varying degree (unigram SLM, bigram SLM, trigram SLM, $n$gram SLM).

- A unigram SLM takes no context knowledge into consideration.
- That is, the probability of a word w_m is independent of its predecessors (w_{m-1}, etc.) and successors (w_{m+1}, etc.).
- The respective Bayesian network is

- Repectively, we have

$$p(w_1^M) = \prod_{w_m \in \{w_1,\ldots,w_M\}} p(w_m|w_{\mathbf{pa}(m)}) = \prod_{m=1}^{M} p(w_m) \quad (113)$$

- A bigram SLM takes knowledge about a word's predecessor into consideration.
- That is, the probability of a word w_m depends on its single predecessor w_{m-1}.
- The respective Bayesian network is

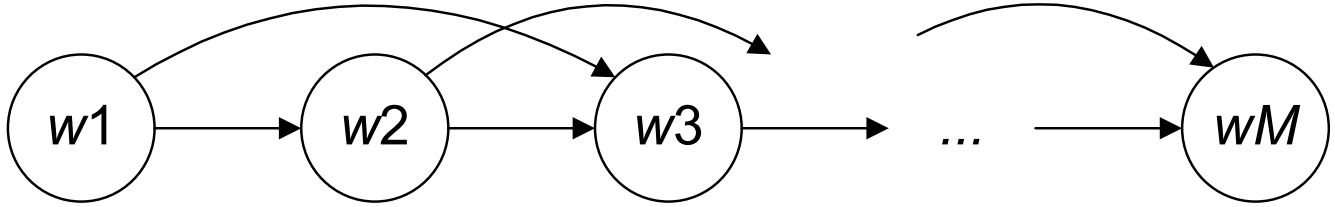


- Repectively, we have

$$p(w_1^M) = \prod_{w_m \in \{w_1,\ldots,w_M\}} p(w_m|w_{\mathbf{pa}(m)})$$

$$= p(w_1) \prod_{m=2}^{M} p(w_m|w_{m-1}) \quad (114)$$

- A trigram SLM takes knowledge about a word's two closest predecessors into consideration.
- That is, the probability of a word w_m depends on the predecessors w_{m-1} and w_{m-2}.
- The respective Bayesian network is



- Repectively, we have

$$p(w_1^M) = \prod_{w_m \in \{w_1,\ldots,w_M\}} p(w_m|w_{\mathbf{pa}(m)})$$

$$= p(w_1)p(w_2|w_1) \prod_{m=3}^{M} p(w_m|w_{m-2}, w_{m-1}) \quad (115)$$

- In our exercises on MT and naïve Bayes, we have made use of strategies coping with unlikely events.
- The fact that we have no encountered certain (combinations of) events does not mean they do not exist, it may just be that our data is too sparse.
- Assuming zero probability of these events often does not work due to the factorial combination of event probabilities.
- A technique to overcome this effect is smoothing which discounts some of the probability mass of observed events and assigns it to unobserved events.
- Popular smoothing techniques include
  - additive (Laplace) smoothing
  - absolute discounting ($n$grams)
  - leaving-one-out ($n$grams)

- Additive smoothing is based on the idea that, due to data sparseness, we have missed a fixed ratio of occurrences per event as expressed by the smoothing parameter α.
- Hence, we re-distribute our probability mass as follows:

$$p'(x_i) = \frac{p(x_i) + \alpha}{\sum_{j=1}^{n} (p(x_j) + \alpha)} = \frac{p(x_i) + \alpha}{1 + \alpha n}. \tag{116}$$

- In our exercise on naïve Bayes classifiers, α could be extremely small (e.g. 10^{-100}) since taking the logarithm converts this value into a number not colliding with the machine precision.

- This technique copes with sparseness of $n$gram counts.
- Due to the exponential explosion of possible $n$grams with growing order n, data gets sparser and sparser as well.
- Consequently, an approach is to back off the $n$gram order in case of zero probabilities.
- Absolute discounting discounts non-zero probabilities by an absolute value β_μ and redistributes the probability mass to unseen events backing off by one $n$gram order:

$$p'(w_m|w_{m-\mu+1}^{m-1}) = \frac{1}{F}\begin{cases} p(w_m|w_{m-\mu+1}^{m-1}) - \beta_\mu & \text{for } p(w_m|w_{m-\mu+1}^{m-1}) > 0 \\ \beta_\mu p'(w_m|w_{m-\mu+2}^{m-1}) & \text{otherwise} \end{cases} \quad (117)$$

  with the normalization constant F.
- β_μ can be determined based on heuristics or trained on a development set.

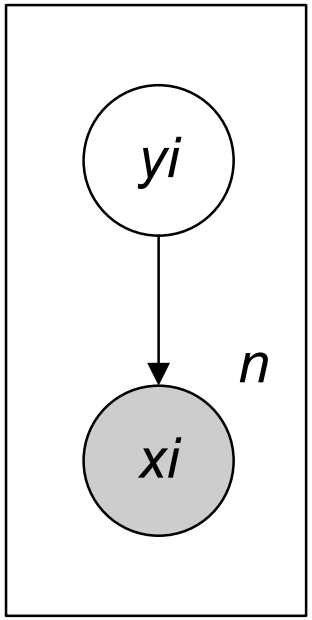
- **In contrast to absolute discounting, this technique linearly combines the probabilities of all $n$gram counts down to the unigram for the given history:**

$$p'(w_m|w_{m-\mu+1}^{m-1}) = \sum_{\mu=1}^{m} \lambda_\mu p(w_m|w_{m-\mu+1}^{m-1}) \quad \textbf{with} \quad \sum_{\mu=1}^{m} \lambda_\mu = 1. \tag{118}$$

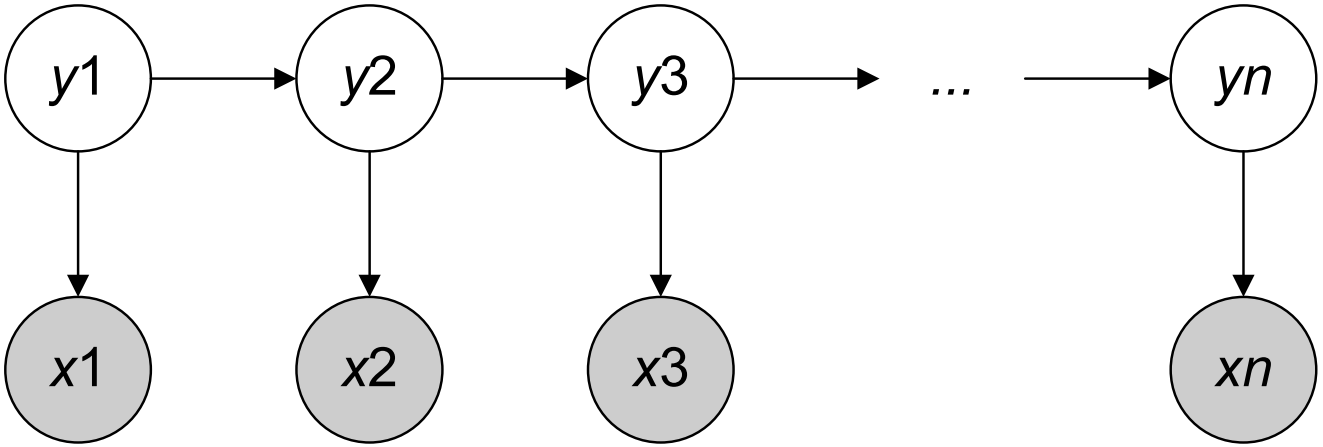
- **Again, λ_μ can be determined based on heuristics (e.g. by discounting non-zero counts of observed events by one–"leaving one out") or trained on a development set.**

1. introduction
2. probability and statistics
3. linear regression
4. neural networks
5. Bayesian networks
6. hidden-Markov models
7. decision trees
8. boosting
9. homework

- As a motivation, let us look at a weather prediction task.
- Here, we are looking at a number of random variables:
  - y_i: the air pressure (low, high) on day i
  - x_i: the fact whether it is raining on day i
- Intuitively, there is a causal relationship between y_i and x_i as expressed by the network on the right.
- It also shows that
  - x_i are observed variables (everybody can tell whether it is raining or not) while
  - y_i are hidden (assuming we do not have a manometer, the air pressure is unknown).



- Now, we want to take into account that the air pressure tomorrow is not independent of the air pressure today.
- That is, there is a relationship between y_i and y_{i+1}:



- This graph looks like a bigram SLM with additional observed variables generated by every hidden.

- Since x_1^n are observed variables, a typical task is to determine the most likely y_1^n given x_1^n:

$$\begin{aligned}
\hat{y}_1^n &= \arg\max_{y_1^n} p(y_1^n|x_1^n) \\
&= \arg\max_{y_1^n} \frac{p(x_1^n, y_1^n)}{p(x_1^n)} \\
&= \arg\max_{y_1^n} p(x_1^n, y_1^n) \\
&= \arg\max_{y_1^n} \prod_{x_i \in \{x_1^n, y_1^n\}} p(x_i|x_{\mathrm{pa}(i)}) \\
&= \arg\max_{y_1^n} \underbrace{\prod_{i=1}^{n} p(x_i|y_i)}_{\text{acoustic model}} \underbrace{p(y_1) \prod_{i=2}^{n} p(y_i|y_{i-1})}_{\text{language model}} \qquad (119)
\end{aligned}$$

- **We are given 962 sample tokens taken from the medical Part-of-Speech tagging corpus Genia available at**

  http://www-tsujii.is.s.u-tokyo.ac.jp/~genia/geniaform.cgi

- **For the sake of simplicity, the tokens are already converted into vector form (962 pairs (x_i, y_i) stored in the Octave matrix `hmm` (see `hmm.m`;** *wordHMM.txt* **and** *classHMM.txt* **maps to original data). Using this data,**

  a) Estimate the probability (1-gram, 2-gram, 3-gram) of the tag sequence

  NN CC NN VBP NN

  b) Build a unigram HMM classifier using the entire data body.

  c) Which class sequence does the classifier return for the input phrase

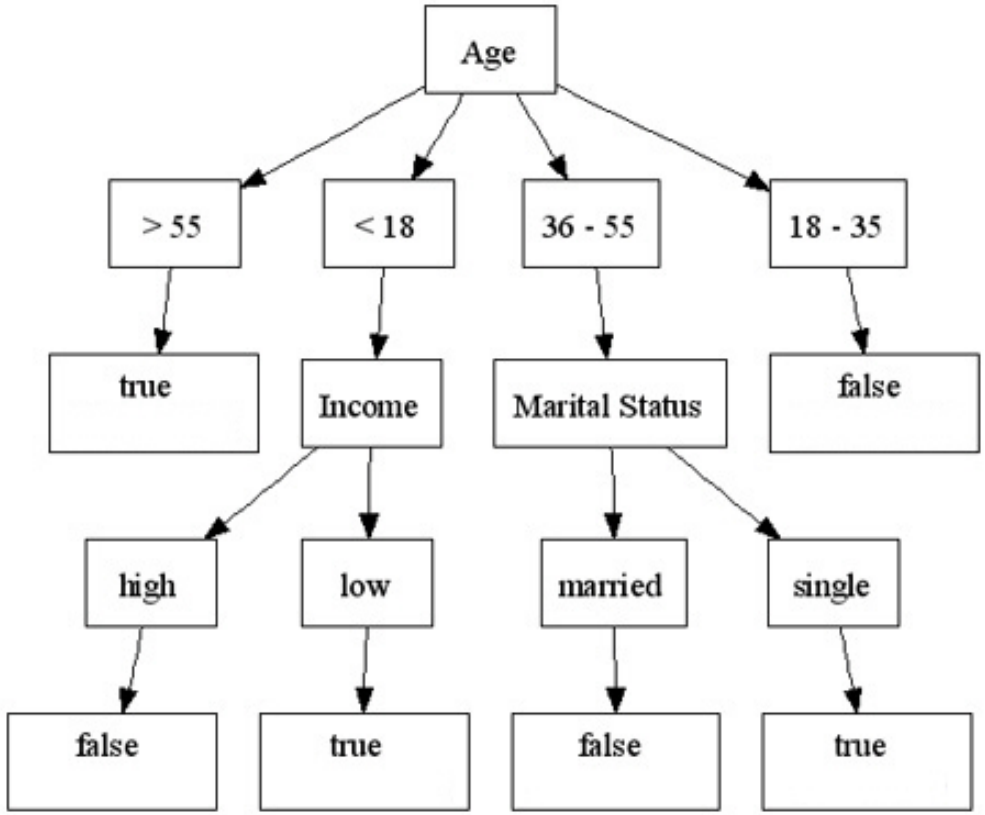
  induce 2 estrogens?

  d) Test your classifier on the entire amount of available data.

  e) Perform ten-fold cross-validation.

  f) How does the result of c) change when using a bigram classifier?

1. introduction
2. probability and statistics
3. linear regression
4. neural networks
5. Bayesian networks
6. hidden-Markov models
7. decision trees
8. boosting
9. homework



Decision trees have two kinds of nodes:

1. **Each leaf node has a class label. This label is determined by the majority vote of the training samples reaching that leaf.**

2. **Each internal node is a question on features. It branches out according to the answers.**

Advantages

- simple to understand, interpret, and implement
- have value even with little hard data (expert estimates can be used)
- robust against violations of model assumptions
- computationally cheap (both training and testing)
- able to handle both numerical and categorical features

Applications

- financial, economical decision making
- expert systems (e.g. for medical diagnosis)
- language processing and dialog systems (call flow, CEI, Engager, Escalator, SLU, verbatim)
- ...

Some approaches and their history

- A decision tree can be constructed manually by a knowledge engineer.
- It is more fun (and more accurate if enough data is available) to induce a tree from training data.

age	income	status	interested?
66	high	married	yes
34	high	single	no
...	...	...	...

- Early work was based on divide and conquer algorithms (e.g. Hoveland and Hunt in the 1950s and 1960s).
- classification and regression tree CART (Breimann et altres, 1984)
- ID3, C4.5, and others were developed by Ross Quinlan starting in 1978.

C4.5 (Quinlan 1993)

1. Check for base cases (e.g. all the targets are identical).
2. For each feature x,
   compute the information gain from splitting on x.
3. Let $\hat{x}$ be the feature with the highest information gain.
4. Create a decision node n that splits on $\hat{x}$.
5. Recur on the subsets obtained by splitting on $\hat{x}$ and add those nodes as children of n.

Why information gain? And what exactly is that?

- We want pure leaf nodes, i.e. all examples having (almost) the same class.
- A good question $\equiv$ a split resulting in a pure node
- How do we measure purity (or impurity)?
- the node's impurity $\equiv$ uncertainty of y in a random drawing

Entropy

$$H(Y) = -\sum_{i=1}^{k} p(y_i) \log_2 p(y_i) \tag{120}$$

- **Interpretation:**
  The number of yes/no questions (bits) needed on average to pin down the value of y in a random drawing

$p(\text{head}) = 0.5$

$p(\text{tail}) = 0.5$

$H = 1\,\text{bit}$

$p(\text{head}) = 0.6$

$p(\text{tail}) = 0.4$

$H = 0.97\,\text{bit}$

Conditional entropy (aka equivocation)

$$H(Y|X) = -\sum_{j=1}^{l} p(x_j) \sum_{i=1}^{k} p(y_i|x_j) \log_2 p(y_i|x_j) \tag{121}$$

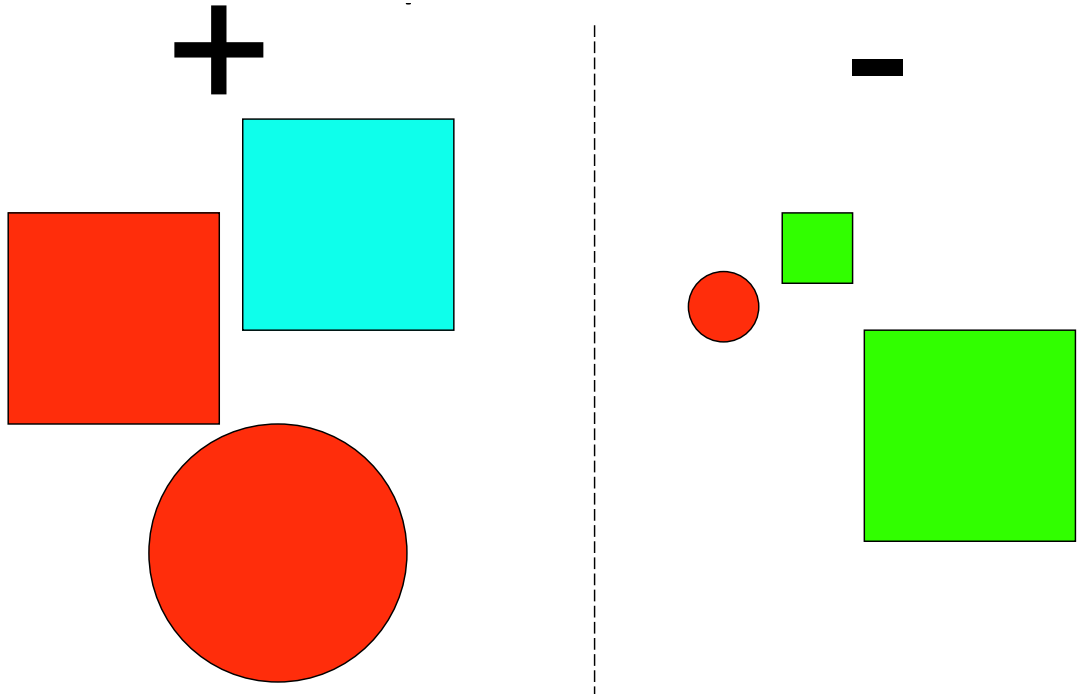
- **Interpretation:**
  Quantifies the remaining entropy of a random variable Y given that the value of another random variable X is known.

Information gain (aka mutual information)

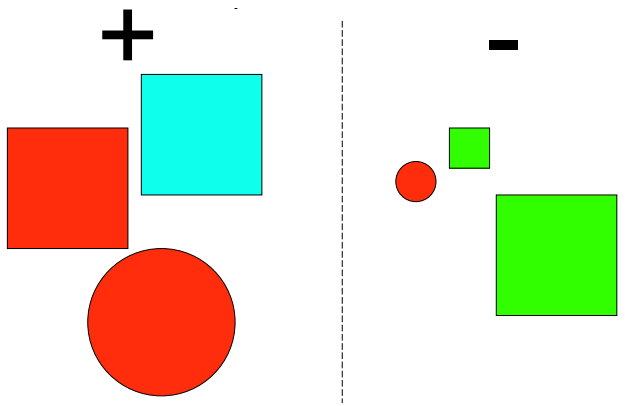
$$I(Y;X) = H(Y) - H(Y|X) \tag{122}$$

- **Choose the question that maximizes $I(Y;X)$.**

- **Features: color, shape, and size**
- **What is the best initial question (at the root node)?**



ID	color	shape	size	class
1	red	square	big	+
2	blue	square	big	+
3	red	circle	big	+
4	red	circle	small	-
5	green	square	small	-
6	green	square	big	-

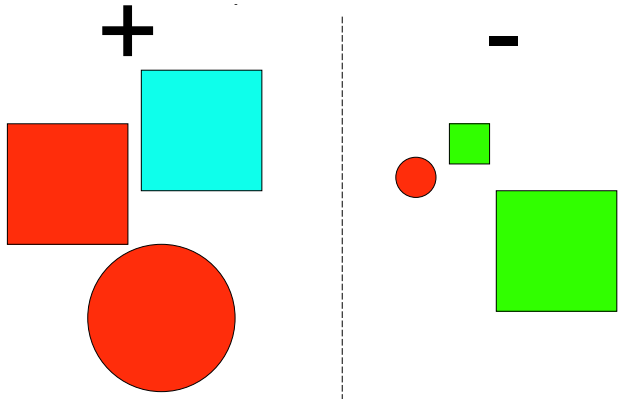


$$H(Y) = -\sum_{i=1}^{k} p(y_i) \log_2 p(y_i)$$

$$H(Y|X) = -\sum_{j=1}^{l} p(x_j) \sum_{i=1}^{k} p(y_i|x_j) \log_2 p(y_i|x_j)$$

$$I(Y;X) = H(Y) - H(Y|X)$$

ID	color	shape	size	class
1	red	square	big	+
2	blue	square	big	+
3	red	circle	big	+
4	red	circle	small	-
5	green	square	small	-
6	green	square	big	-

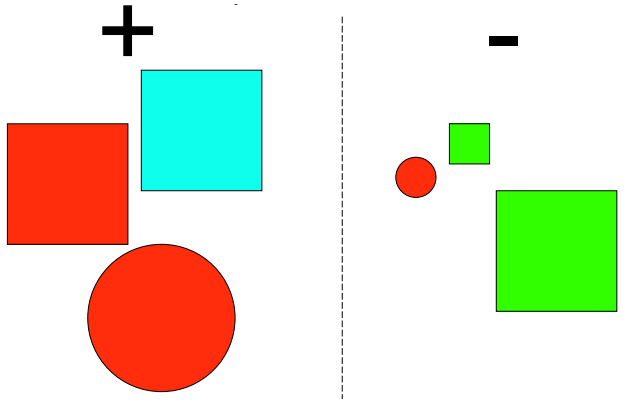


$$H(Y) = H\left(\frac{3}{6}, \frac{3}{6}\right) = 1\,\text{bit}$$

$$H(Y|X) = \frac{3}{6}H\left(\frac{2}{3}, \frac{1}{3}\right) + \frac{1}{6}H\left(1, 0\right) + \frac{2}{6}H\left(0, 1\right) = 0.46\,\text{bit}$$

$$I(Y;X) = \underline{0.54\,\text{bit}}$$

ID	color	shape	size	class
1	red	square	big	+
2	blue	square	big	+
3	red	circle	big	+
4	red	circle	small	-
5	green	square	small	-
6	green	square	big	-

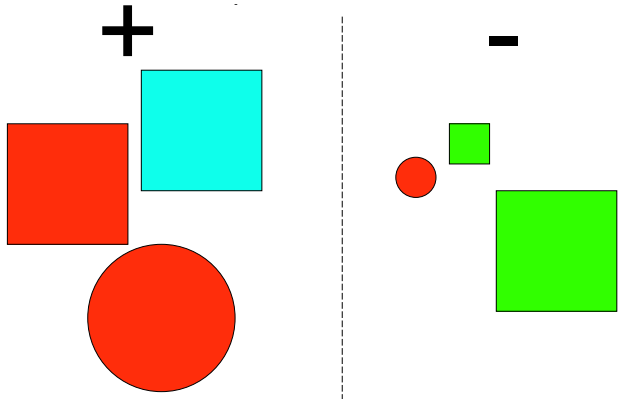


$$H(Y) = H\left(\frac{3}{6}, \frac{3}{6}\right) = 1\,\text{bit}$$

$$H(Y|X) = \frac{4}{6}H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{6}H\left(\frac{1}{2}, \frac{1}{2}\right) = 1\,\text{bit}$$

$$I(Y;X) = \underline{0\,\text{bit}}\ (!)$$

ID	color	shape	size	class
1	red	square	big	+
2	blue	square	big	+
3	red	circle	big	+
4	red	circle	small	-
5	green	square	small	-
6	green	square	big	-

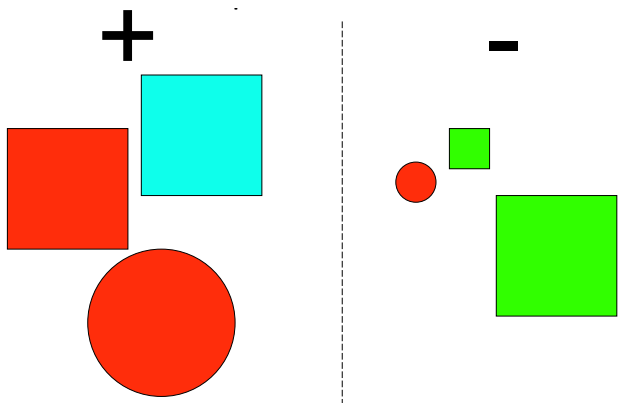


$$H(Y) = H\left(\frac{3}{6}, \frac{3}{6}\right) = 1\,\text{bit}$$

$$H(Y|X) = \frac{4}{6}H\left(\frac{3}{4}, \frac{1}{4}\right) + \frac{2}{6}H\left(0, 1\right) = 0.54\,\text{bit}$$

$$I(Y;X) = \underline{0.46\,\text{bit}}$$

ID	color	shape	size	class
1	red	square	big	+
2	blue	square	big	+
3	red	circle	big	+
4	red	circle	small	-
5	green	square	small	-
6	green	square	big	-



$$\begin{aligned} \text{color}: \quad I(Y;X) &= \underline{0.54\,\text{bit}} \\ \text{shape}: \quad I(Y;X) &= 0\,\text{bit} \\ \text{size}: \quad I(Y;X) &= 0.46\,\text{bit} \end{aligned}$$

- **Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.**
- **example: weather forecast**

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- **Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.**
- **example: weather forecast**

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- **What do you think is the best feature?**

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- $I(\text{year}) = 0\,\text{bit} \qquad I(\text{month}) = 0\,\text{bit}$

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- $I(\text{day}) = 0.86\,\text{bit}$ $I(\text{barometer}) = 0.58\,\text{bit}$

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: day

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: day

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: day

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: barometer

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: barometer

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: barometer

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: barometer

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: barometer

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: barometer

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: barometer

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: barometer

- Imagine we have a set of features whose combination uniquely identifies the corresponding class in the training data.
- example: weather forecast

year	month	day	barometer	weather
2010	12	11	fall	cloudy
2010	12	12	rise	cloudy
2010	12	13	high	sunny
2010	12	14	rise	sunny
2010	12	15	fall	cloudy
2010	12	16	low	cloudy
2010	12	17	low	cloudy

- What do you think is the best feature?
- Leaving-one-out: barometer

Techniques to avoid overfitting

- feature selection (just discussed)
- pruning
- example of a greedy pruning algorithm:

  Given a decision tree T and development data $\{X_d, Y_d\}$

  1. For every internal node in T:
     * Let T'_N be T with pruning the sub-tree under N.
     * N becomes a leaf node of T'_N.
     * N's class is the majority vote of all examples reaching N.
  2. $$T := \underset{t \in \{T, T'_1, \ldots\}}{\arg\max}\ \text{acc}(t) \tag{123}$$
  3. Repeat from step 1 until accuracy does not improve anymore.

Techniques to avoid overfitting

- feature selection (just discussed)
- pruning
- example of a greedy pruning algorithm: Why not exhaustive?

  Given a decision tree T and development data $Z_d = (X_d, Y_d)$

  1. For every internal node in T:
     * Let T'_N be T with pruning the sub-tree under N.
     * N becomes a leaf node of T'_N.
     * N's class is the majority vote of all Z_d examples reaching N.
  2. $$T := \arg\max_{t \in \{T, T'_1, \ldots\}} \text{acc}(t) \tag{124}$$
  3. Repeat from step 1 until accuracy does not improve anymore.

1. introduction
2. probability and statistics
3. linear regression
4. neural networks
5. Bayesian networks
6. hidden-Markov models
7. decision trees
8. boosting
9. homework

- **Michael Kearns (1988): "Can a set of weak learners create a single strong learner?"**
- **Freund and Shapire published a number of boosting algorithms, a very popular being AdaBoost.**
- **A string classifier is build as a linear combination of weak classifiers:**

$$H(x) = \text{sign}\left(\sum_i = 1^T \alpha_i h_i(x)\right) \tag{125}$$

- We are given the training data x_1^N and t_1^N.
- For each of the T available weak classifiers, the algorithm will determine a distribution $D_i(n)$ over the set of training data points.
- The first classifier is initialized with $D_1(n) = \frac{1}{N}$ for $n = 1, \ldots, N$.
- Find the classifier producing the lowest error

$$h_t = \underset{i \in \{1,\ldots,T\}}{\arg\max} |0.5 - \varepsilon| \quad \text{with} \quad \varepsilon = \sum_{n=1}^{N} D_t(n)\delta(h_t(x_n), t_n) \tag{126}$$

- Chose α_t, often as

$$\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon}{\varepsilon}. \tag{127}$$

- Update the distribution

$$D_{t+1} = \frac{D_t e^{2\delta(h_t(x_n), t_n) - 1}}{Z_t} \tag{128}$$

  with the normalization factor Z_t.

1. introduction
2. probability and statistics
3. linear regression
4. neural networks
5. Bayesian networks
6. hidden-Markov models
7. decision trees
8. boosting
9. homework

- **In order to be eligible to write the exam, you will have to satisfactorily complete a home project.**
- **Please prepare a slide deck as you would be using for a short presentation (about 10 slides) about one of the following topics:**
  - **reinforcement learning**
  - **fuzzy systems**
  - **evolutionary algorithms**
  - **support vector machines**
  - **maximum entropy classification**
  - **Gaussian mixture models**
  - **instance-based classification**
  - **deep neural networks**
  - **conditional random fields**
- **Hand in your slide deck to** `david@suendermann.com` **no later than October 31.**