Artificial Intelligence

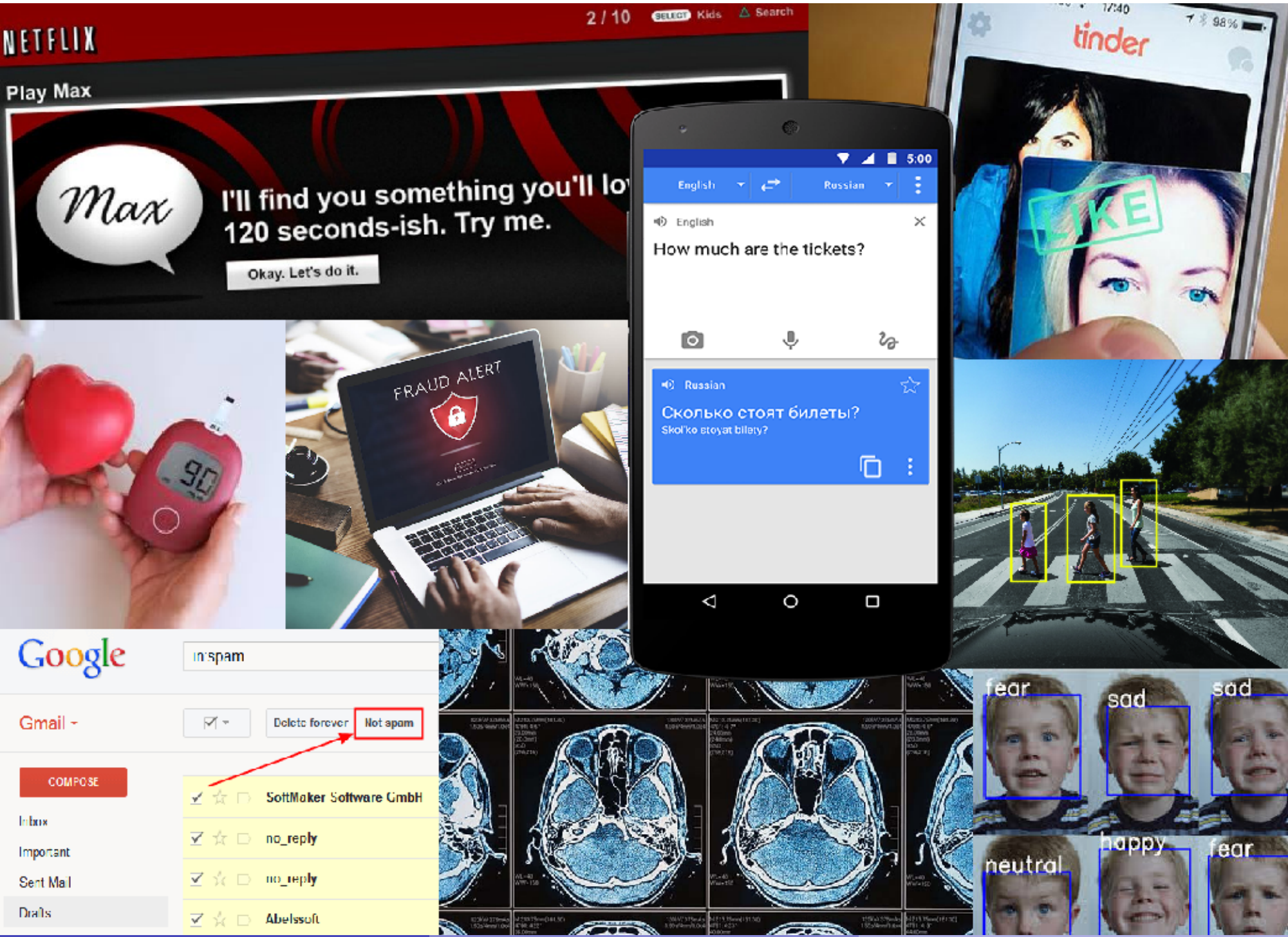
10. Machine Learning

Prof. Bojana Dalbelo Bašić
Assoc. Prof. Jan Šnajder

University of Zagreb
Faculty of Electrical Engineering and Computing

Academic Year 2019/2020

v1.5


NETFLIX
2 / 10
Kids
Search
Play Max
Max
I'll find you something you'll lo
120 seconds-ish. Try me.
Okay. Let's do it.
tinder
LIKE
5:00
English
Russian
English
How much are the tickets?
Russian
Сколько стоят билеты?
Skol'ko stoyat bilety?
FRAUD ALERT
Google
in:spam
Gmail
Delete forever
Not spam
COMPOSE
Inbox
Important
Sent Mail
Drafts
SoftMaker Software GmbH
no_reply
no_reply
Abelssoft
fear
sad
sad
neutral
happy
fear

Data Mining Cup 2010

Using the existing characteristics of a customer's initial order, such as order quantity per type of goods, title and delivery weight, a decision must be made on whether to send a voucher worth EUR 5.00. The customers who receive a voucher should be those who would not have decided to re-order by themselves.

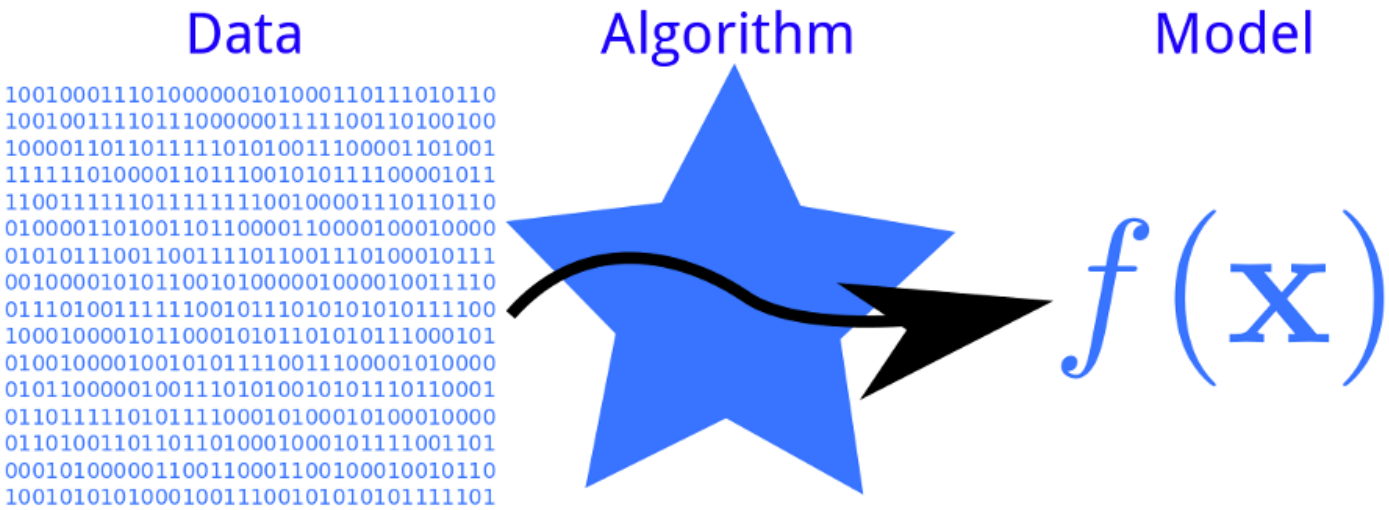
IEEE ICDM 2010 DM Competition

Modeling the process of traffic jams formation during morning peak in the presence of roadworks, based on initial information about jams broadcast by radio stations. Input data contain identifiers of road segments closed due to roadworks, accompanied by a sequence of segments where the first jams occurred. The algorithm should predict a sequence of segments where next jams will occur in the nearest future.

Outline

Outline

What is machine learning?



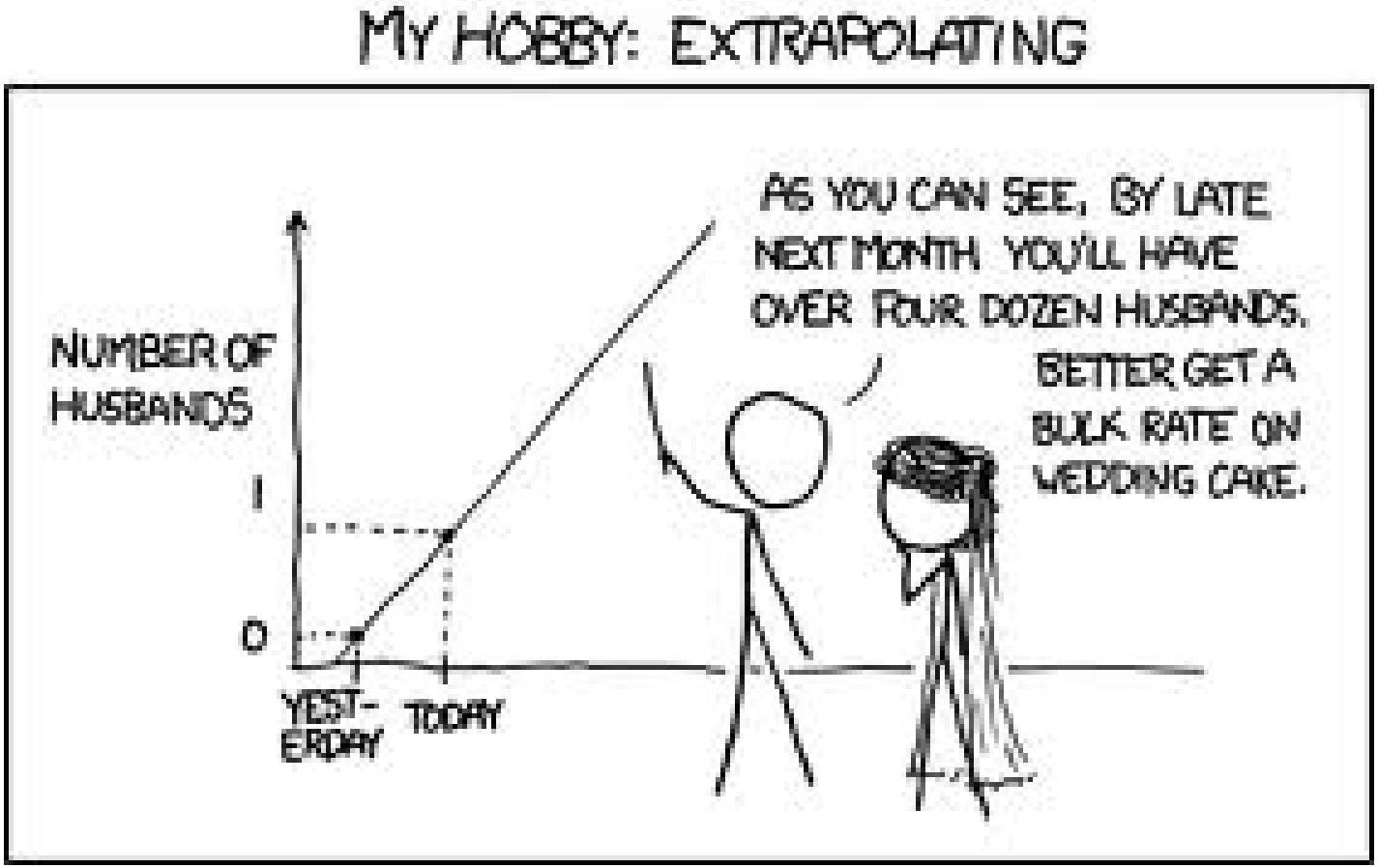
Machine learning (Alpaydin 2009)

Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameter, and learning is the execution of a computer algorithm to optimize the parameters of the model using the training data or past experience.

Prediction

- Based on seen data, the model should be able to predict properties of new, unseen data
- Otherwise said, the model should be able to generalize
- From a logical perspective, what the model does is induction: based on a limited sample of data, the model infers a general rule how the data behave
- Aim of machine learning: build models that generalize well!

Generalization requires some data!



Why machine learning?

At least three reasons:

1. Problems that are too hard to be solved algorithmically – we have no idea how to solve them, because we don't really know how we do it ourselves (e.g., speech recognition)
   - problems that cannot be solved in a traditional (algorithmic) manner $\Rightarrow$ AI-complete problems
2. Dynamically adaptable systems – e.g., robots, adaptable user interfaces
3. Vast amounts of data – vast data is out there, can we tap on the knowledge that it contains?
   - data science, big data, data mining

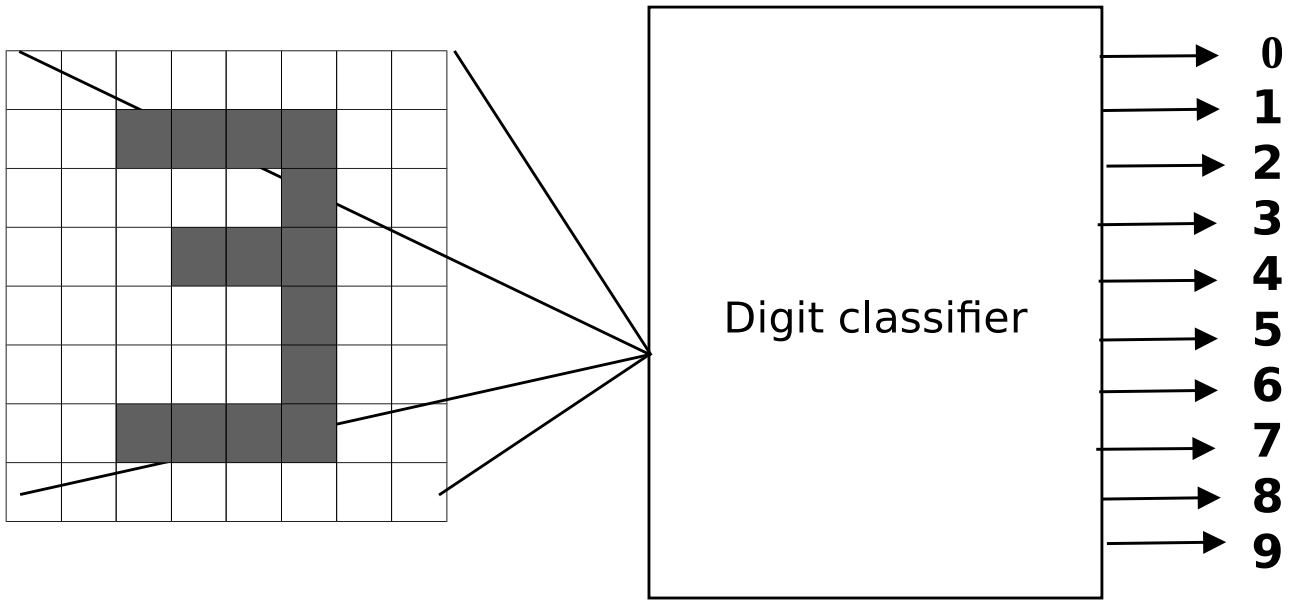
NB: Not all problems deserve an ML solution (e.g., payroll accounting software). Do not use a cannon to kill a fly.

Case 1: Hard problems

- Pattern recognition
- Computer vision
- Natural language understanding
- Speech recognition
- Robotics
- ...

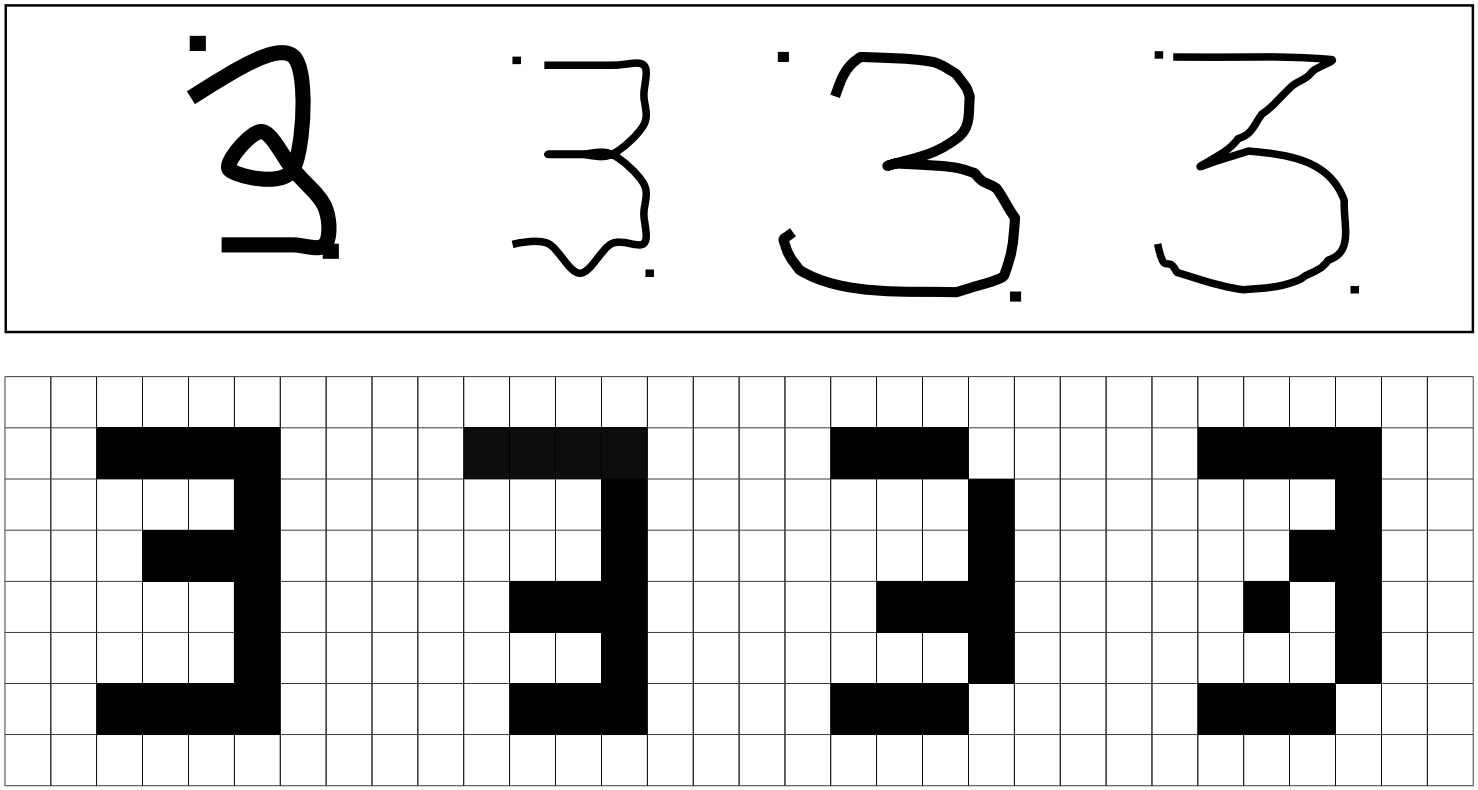
Example: Recognizing handwritten digits

- The system is presented patterns (digitized input)
- It needs to recognize to which of the predefined 10 classes the input pattern belongs to
- This is called pattern recognition or pattern classification



Example: Recognizing handwritten digits

- The problem is that no one writes perfect digits: there is a large number of variants of each digit



Example: Recognizing handwritten digits

- Naive approach:
  - store all possible variants of each digit in memory
  - when presented with a digit (8×8 binary matrix), look it up
- **Q:** Why is this not a good idea?
- **A:** Because there are too many different patterns: $2^{64} = 1.8 \cdot 10^{19}$

1. Memory issue
   - assume each pattern is 8 bytes $\Rightarrow 2^{(64+3)}$ bytes $=$ 157 exabytes
2. Time issue
   - assume comparing patterns takes 100 microseconds $\Rightarrow$ average look-up time is $\frac{1}{2} \cdot 2^{64} \cdot 10^{-4}$ seconds $=$ 29 million years
3. Conceptual issue
   - even if we could solve the above issues, how do we gather the data in the first place? Manual labeling of each pattern would obviously take much longer than 29 million years!

Case 2: Dynamically adaptable systems

- An intelligent systems should be capable of adapting to its environment – be capable of learning. If it is capable of learning, it can plan how to act in new situations.

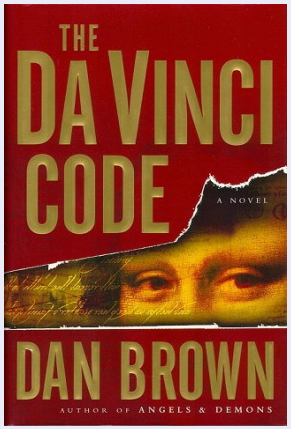
- Robotics
- Multi-agent systems
- Intelligent user interfaces

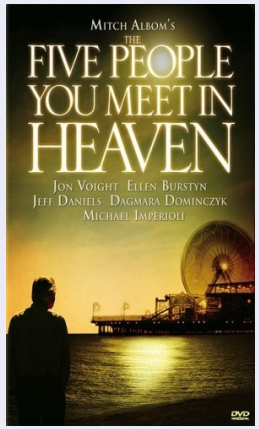
Case 3: Knowledge from data

- From data to information to knowledge to action
- Data exists in abundance (web, text, video, audio, experimental data, data warehouses, click data, deep web)
- On the other hand, knowledge is scarce and expensive
- Goal: build models that explain the data and allow us to make inferences/predictions

Example: Purchase transactions explain users' behavior

People who bought
"The da Vinci Code"
also bought
"The Five People You Meet in Heaven"



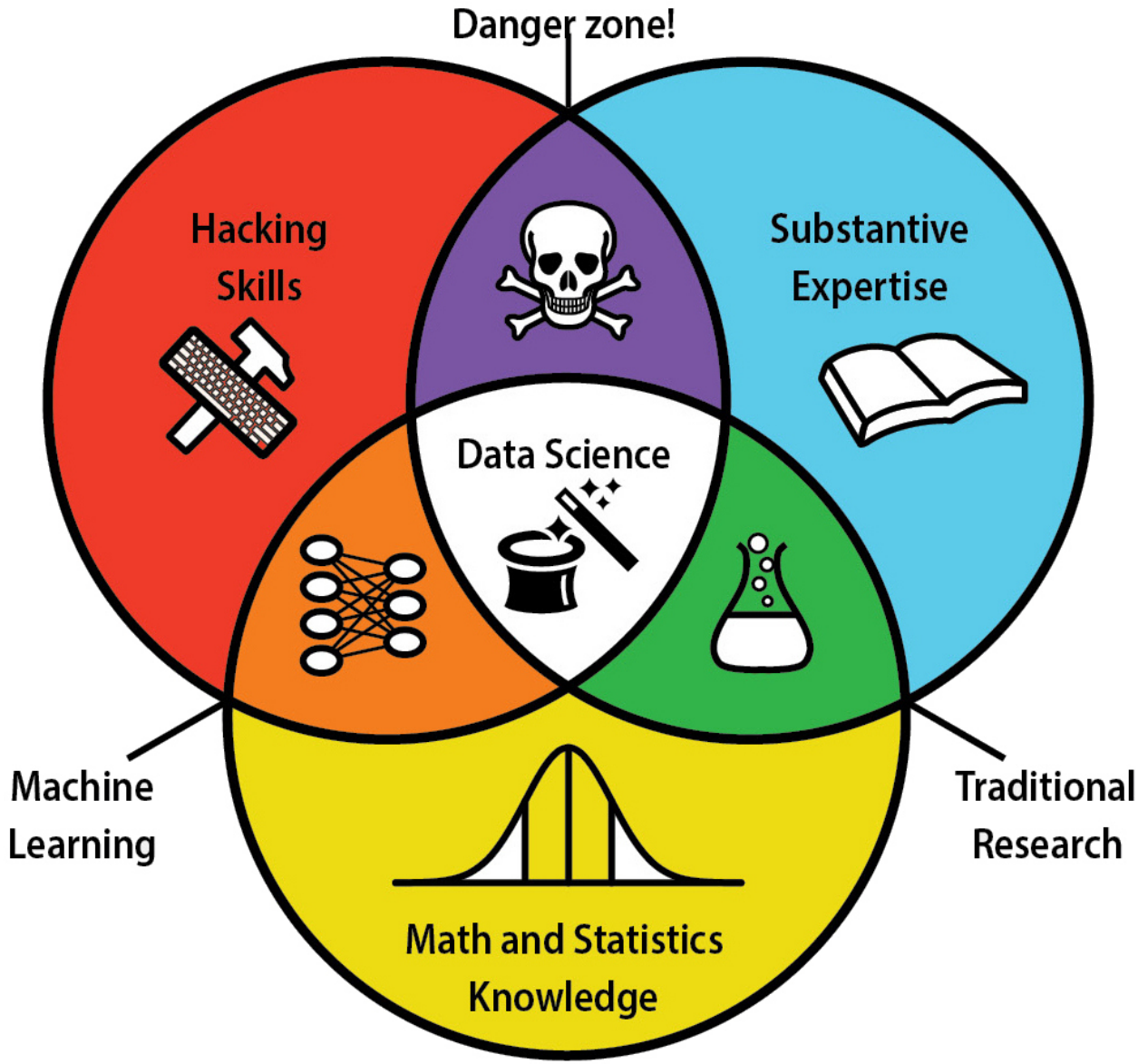


Data Mining

Data mining or knowledge discovery in data applies machine learning algorithms to structured and unstructured databases

- Commerce: market basket analysis, CRM
- Finances: credit risk assessment, credit card fraud
- Manufacturing: optimization, troubleshooting
- Medicine: diagnosis
- Telecommunications: service optimization, churn prediction
- Bioinformatics: gene expression analysis, sequence alignment
- Text mining: document classification, information extraction
- ...

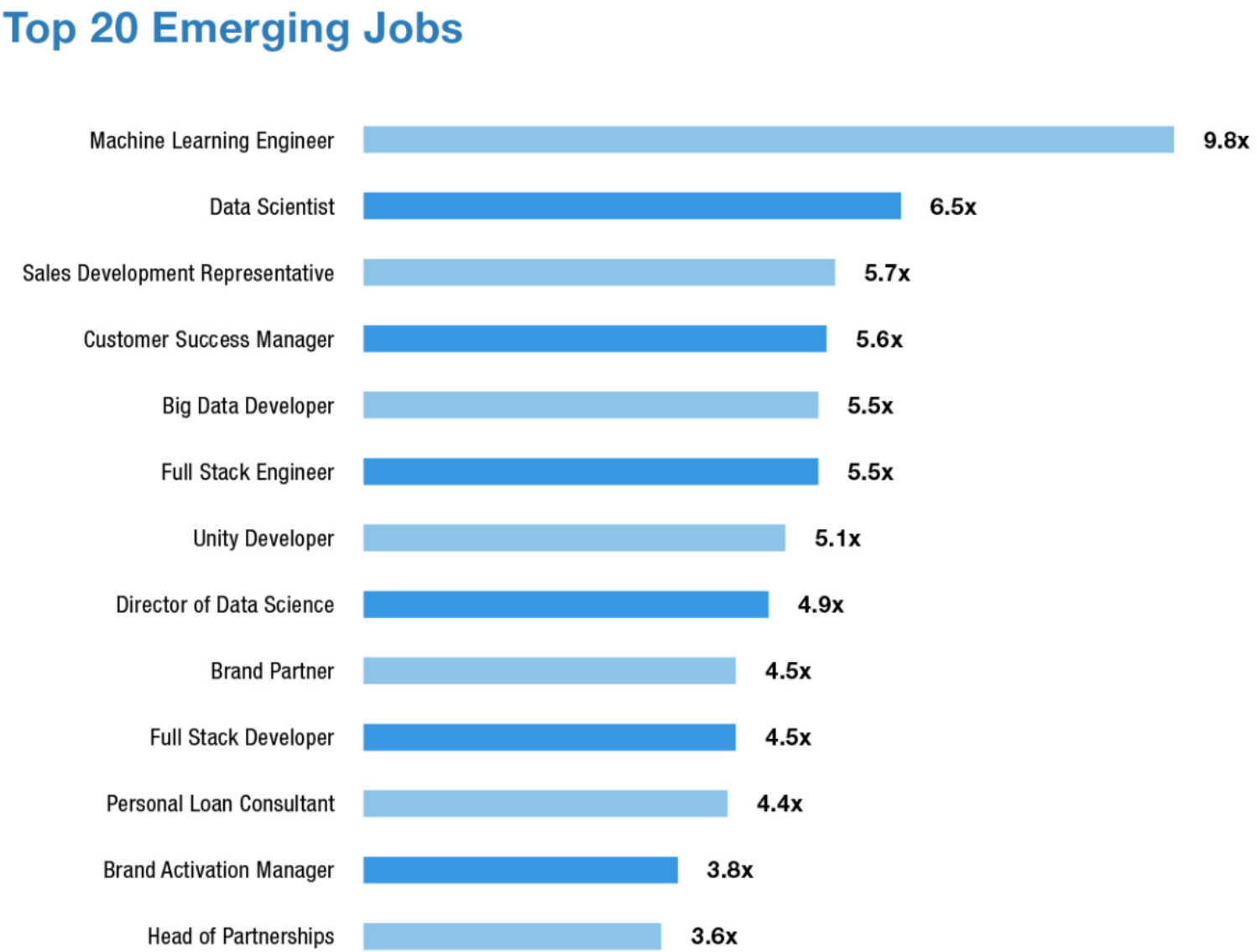
Data Science



http://berkeleysciencereview.com/how-to-become-a-data-scientist-before-you-graduate/

Data Science

LinkedIn: The fastest growing jobs in the US (2012–2017)



The promises and perils of Data Science

Most of the knowledge in the world in the future is going to be extracted by machines and will reside in machines.

– Yann LeCun, Director of AI Research, Facebook

If we used all our technology resources, we could actually give people personalized recommendations for every step of your life.

– Aneesh Chopra, former CTO of the U.S.

Machine learning and cognitive science

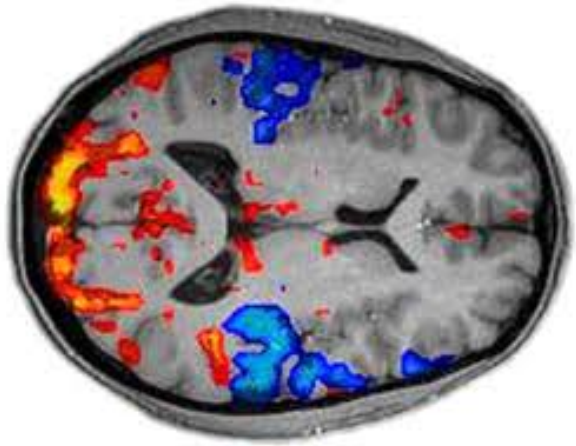
Cognitively plausible models

Understanding machine learning algorithms $\Leftrightarrow$ Understanding human learning abilities

Thought Reading Experiment:

`http://www.cs.cmu.edu/afs/cs/project/theo-73/www/index.html`

- Functional magnetic resonance (fMRI)
  - Detects blood flow changes in the brain: active brain areas consume more oxygen
  - Exploits the fact that molecules in the blood react to the magnetic field depending on the level of oxygen



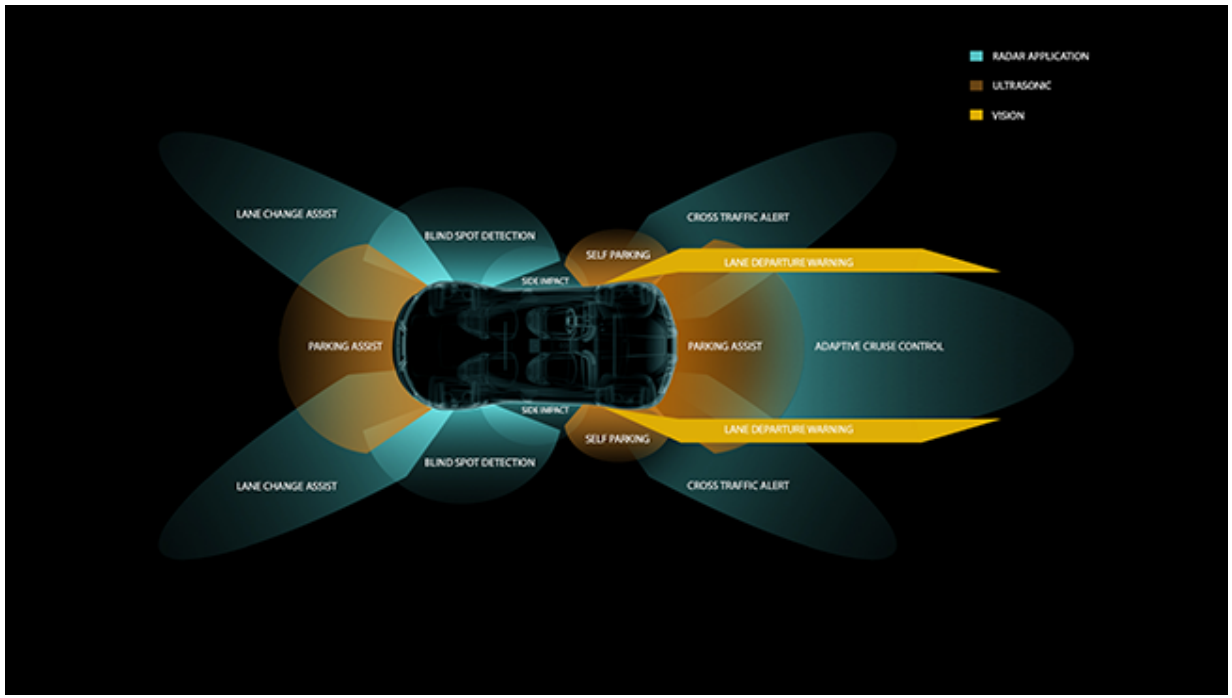
Deep learning: The current big thing in ML

- Combines cleverly engineered artificial neural networks and large amounts of data
- We have had neural networks before, but today we also have:
  - Much faster computer architectures (GPU SIMD)
  - Much more data
- Reaches *state-of-the-art* results on a broad range of machine learning tasks
- Google, Baidu, Paypal, NVIDIA, Facebook, ...

Deep learning: autonomous vehicles (NVIDIA)

http://blogs.nvidia.com/blog/2015/02/24/deep-learning-drive/

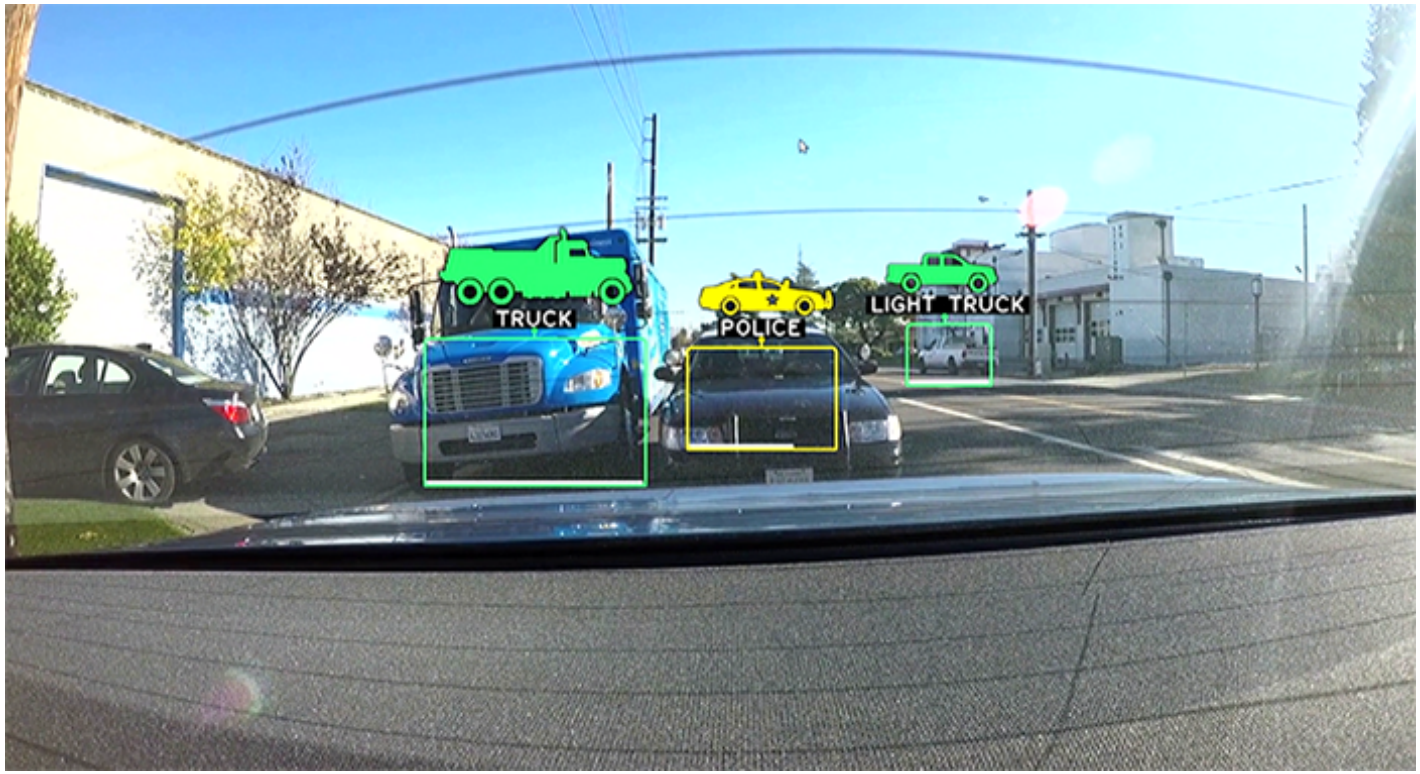
- Various sensors gather large amounts of data which a deep learning-based system uses to solve challenging tasks such as detecting pedestrians, automatic parking, changing lanes, ...



Deep learning: autonomous vehicles (NVIDIA)

http://blogs.nvidia.com/blog/2015/02/24/deep-learning-drive/

- A classifier recognizes various types of vehicles and traffic signs, even in cases that would be difficult for humans (bad weather conditions etc.)



Deep dish learning (Google)

http://www.popsci.com/google-using-ai-count-calories-food-photos

- Based on a still photo of a dish, the system estimates how many calories it has

Outline

Approaches

(1) Supervised learning

Data is labeled and comes in pairs *(input, output)*$=(\mathbf{x}, y)$. We need to find a mapping $f(\mathbf{x}) = y$

- If y is a discrete label: classification
- If y is a number: regression

(2) Unsupervised learning

Data comes without labels. We need to find regularities in the data.

- clustering
- novelty/outlier detection
- dimensionality reduction

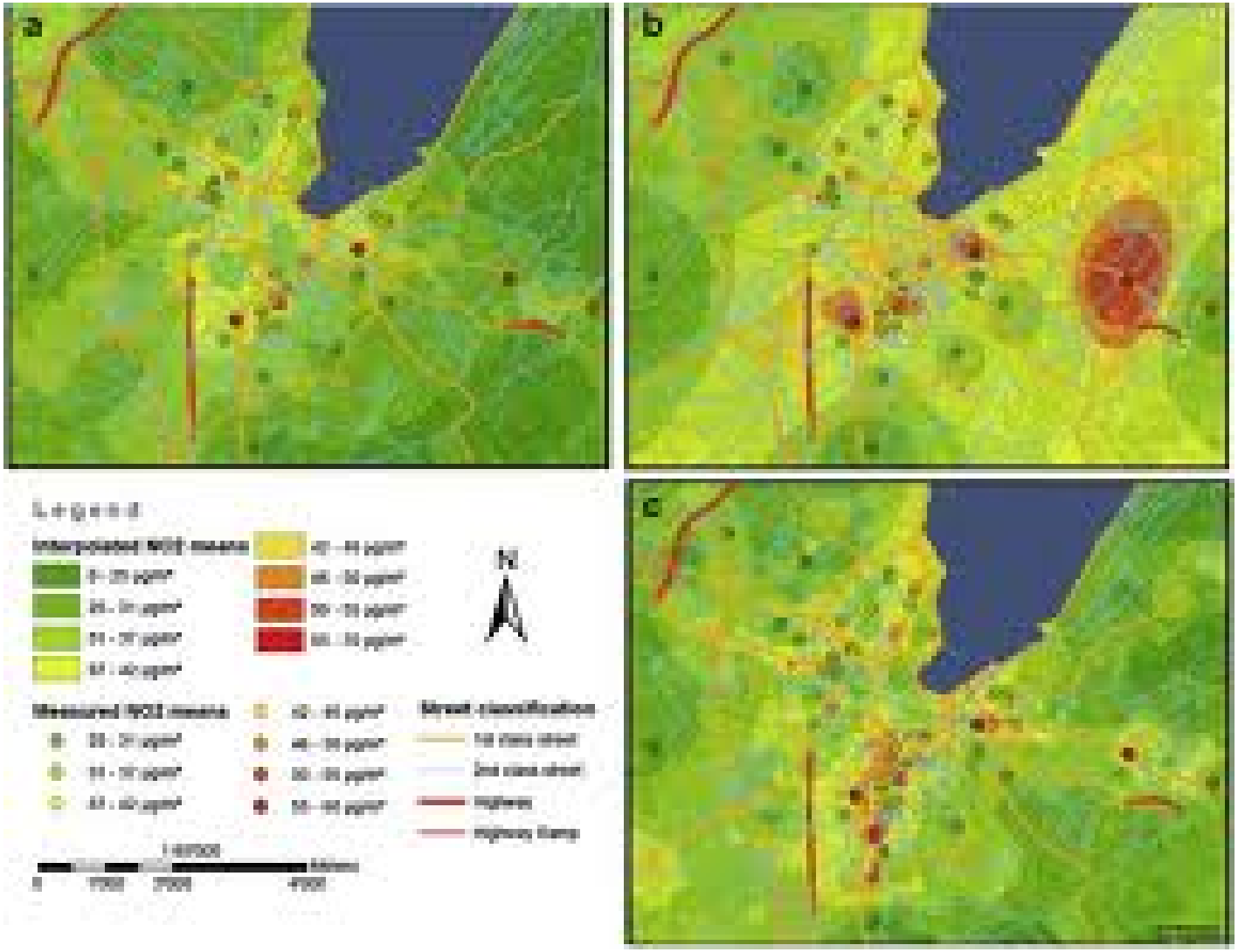
(3) Reinforcement learning

Learning an optimal strategy based on trials with a delayed credit.

Classification example: Predicting election outcomes

Election Analytics (`http://electionanalytics.cs.illinois.edu/index.html`)

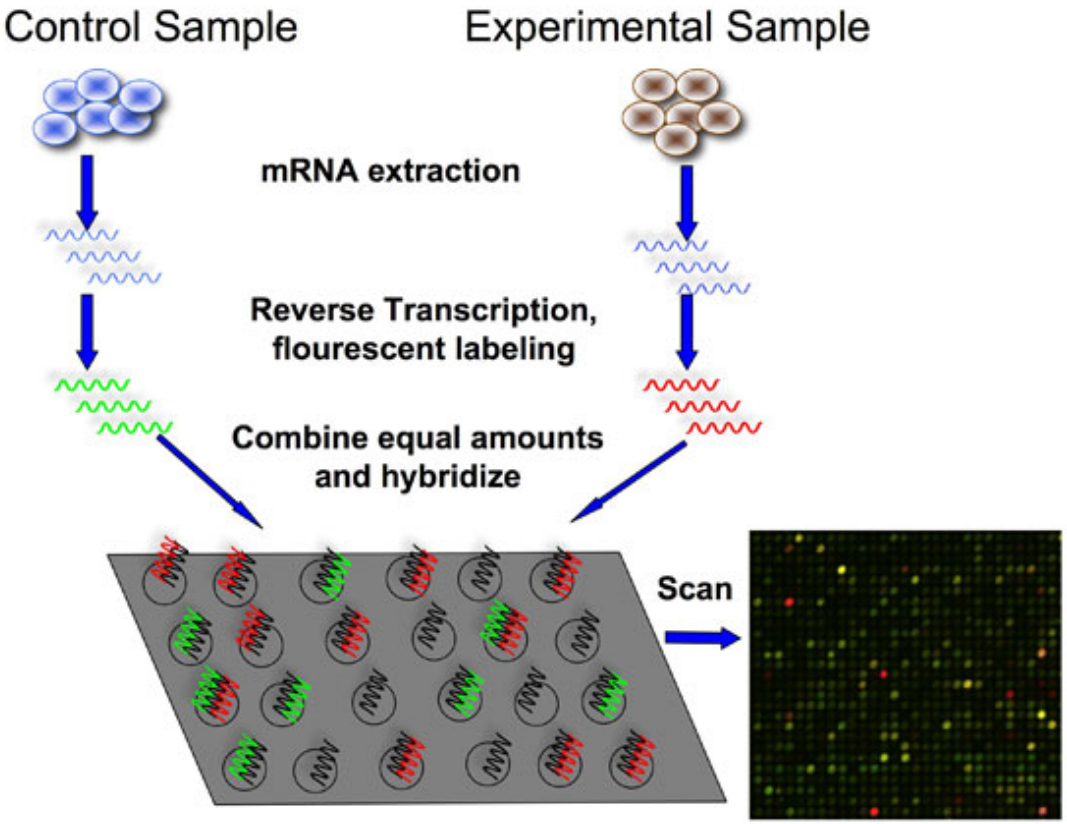
Regression example: predicting NO_2 level emissions

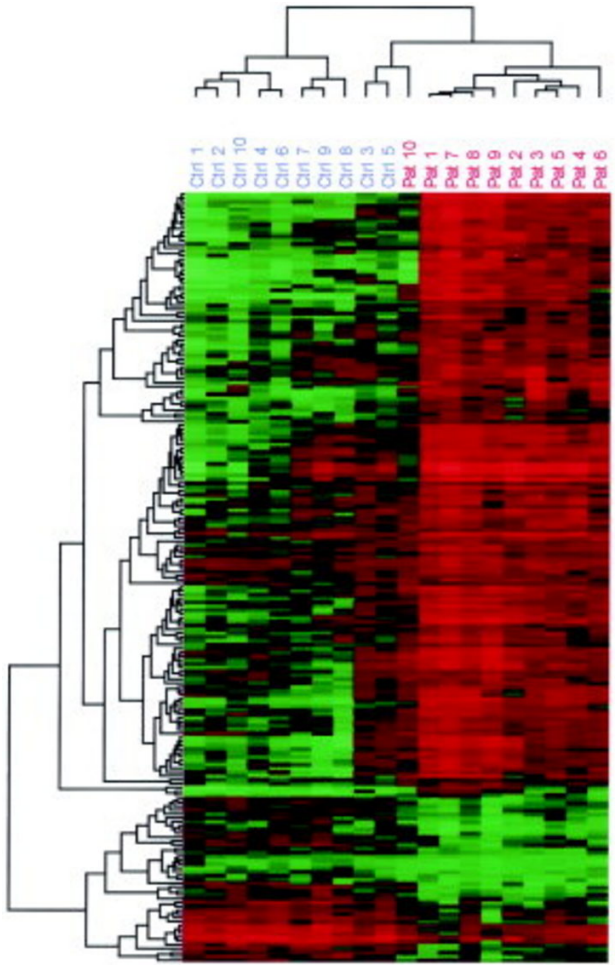


Exposure models for traffic related NO2 (`http://www.sciencedirect.com/science/article/pii/S1352231011009629`)

Clustering example: grouping DNA microarrays

Goal: grouping genes with similar expressivity
(similar expressivity $\leftrightarrow$ similar functionality)

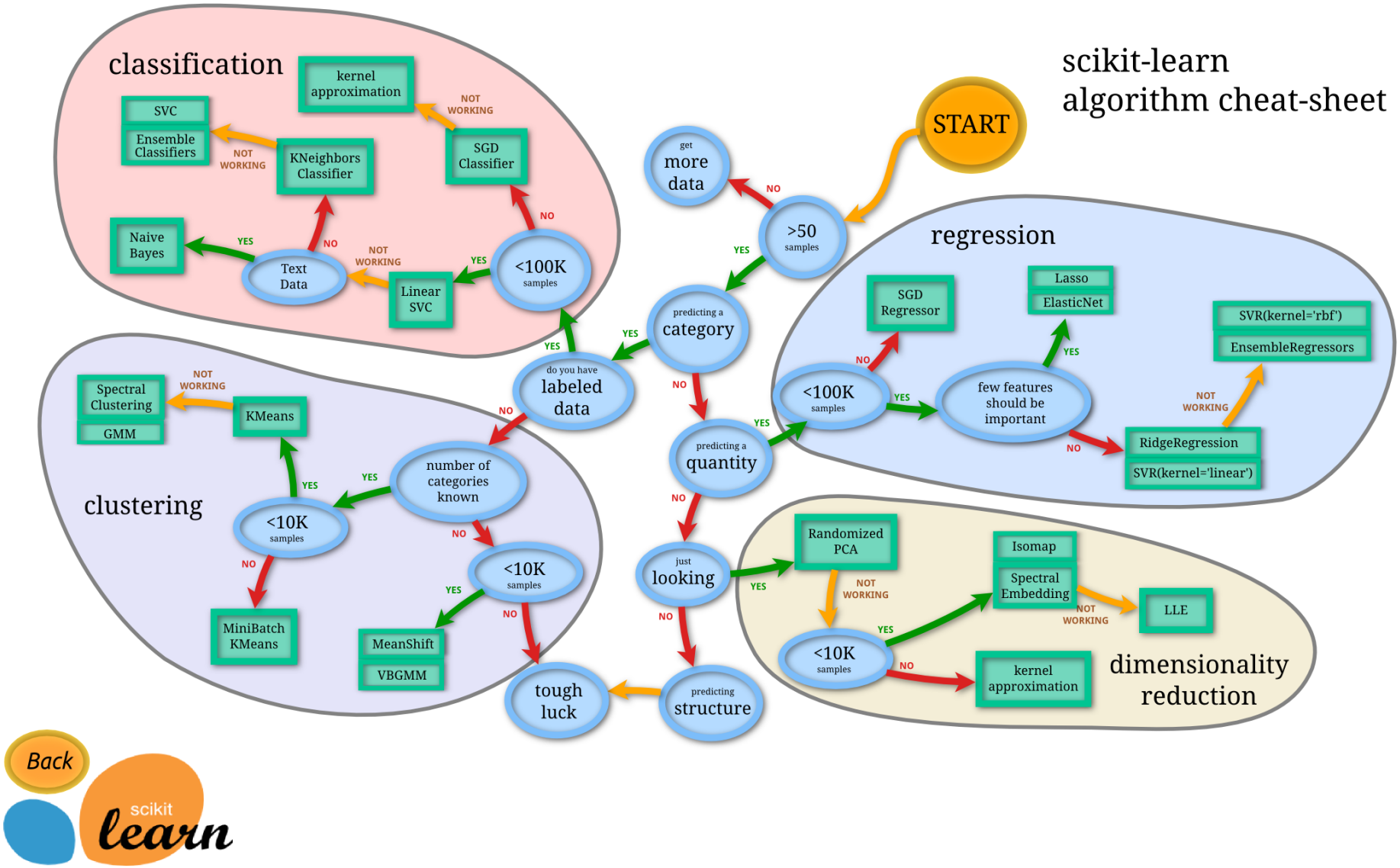




Reinforcement learning: AlphaGo

- Two deep neural networks (DNN) trained to predict the next move and reduce the search space
- Reinforcement learning on top of DNN, to learn a playing strategy, trained by playing parties against itself
- Uses 40 search threads, 48 CPUs, and 8 GPUs. Distributed version uses 40 search threads, 1,202 CPUs and 176 GPUs

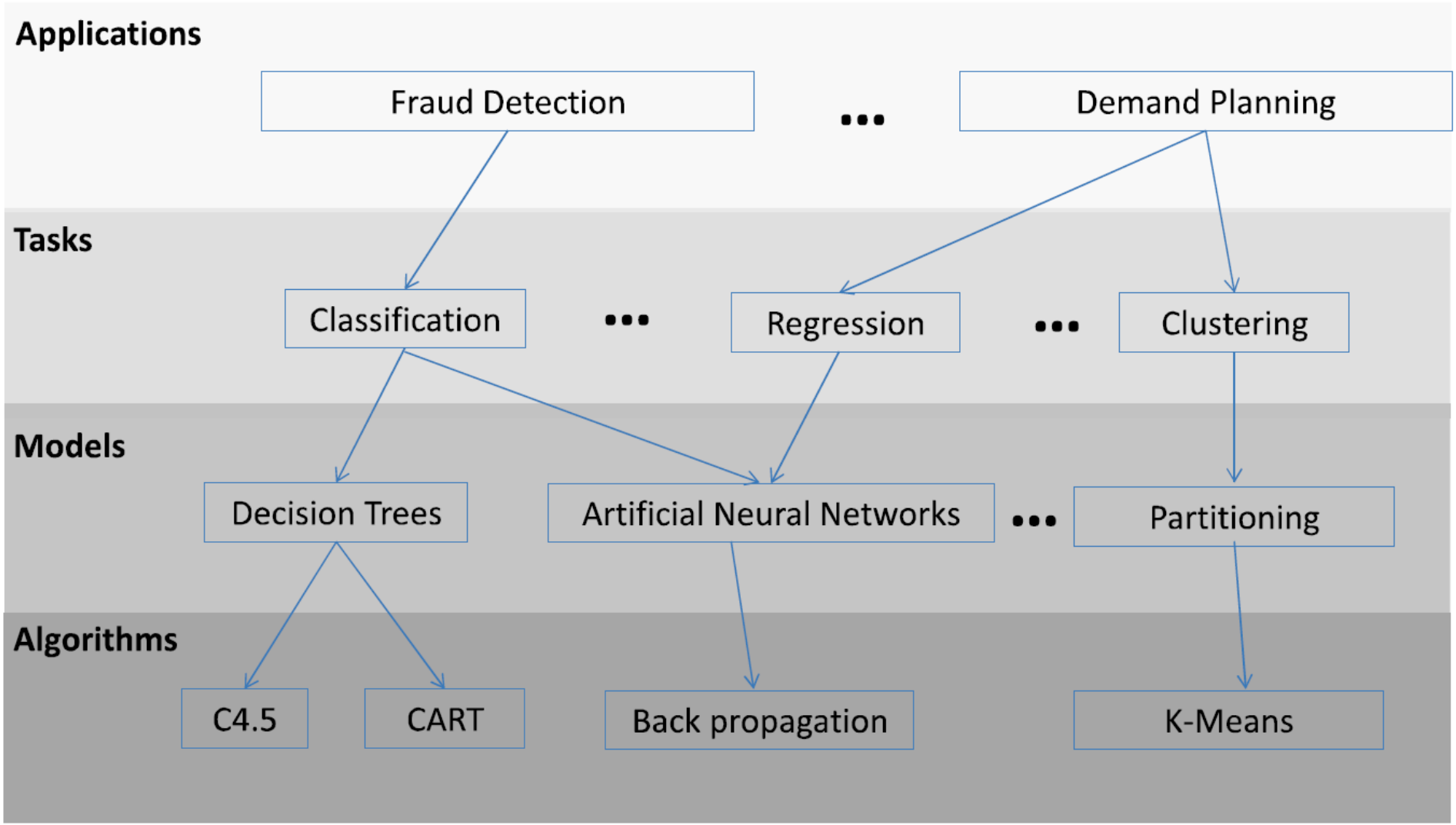
Machine learning map



Back

http://scikit-learn.org/stable/tutorial/machine_learning_map/

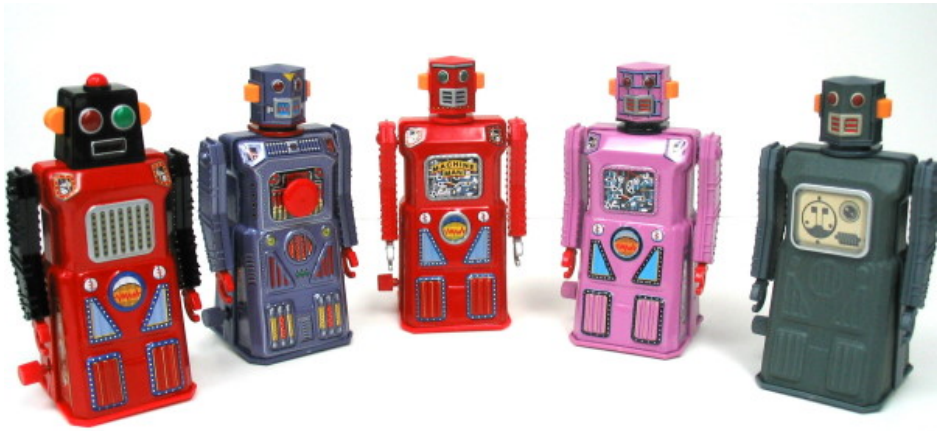
Applications, tasks, models, and algorithms



Rokach, L. and Maimon, O. Z. (2008). Data Mining with Decision Trees: Theory and Applications.

The Five Tribes of Machine Learning

Pedro Domingos: *The Master Algorithm*

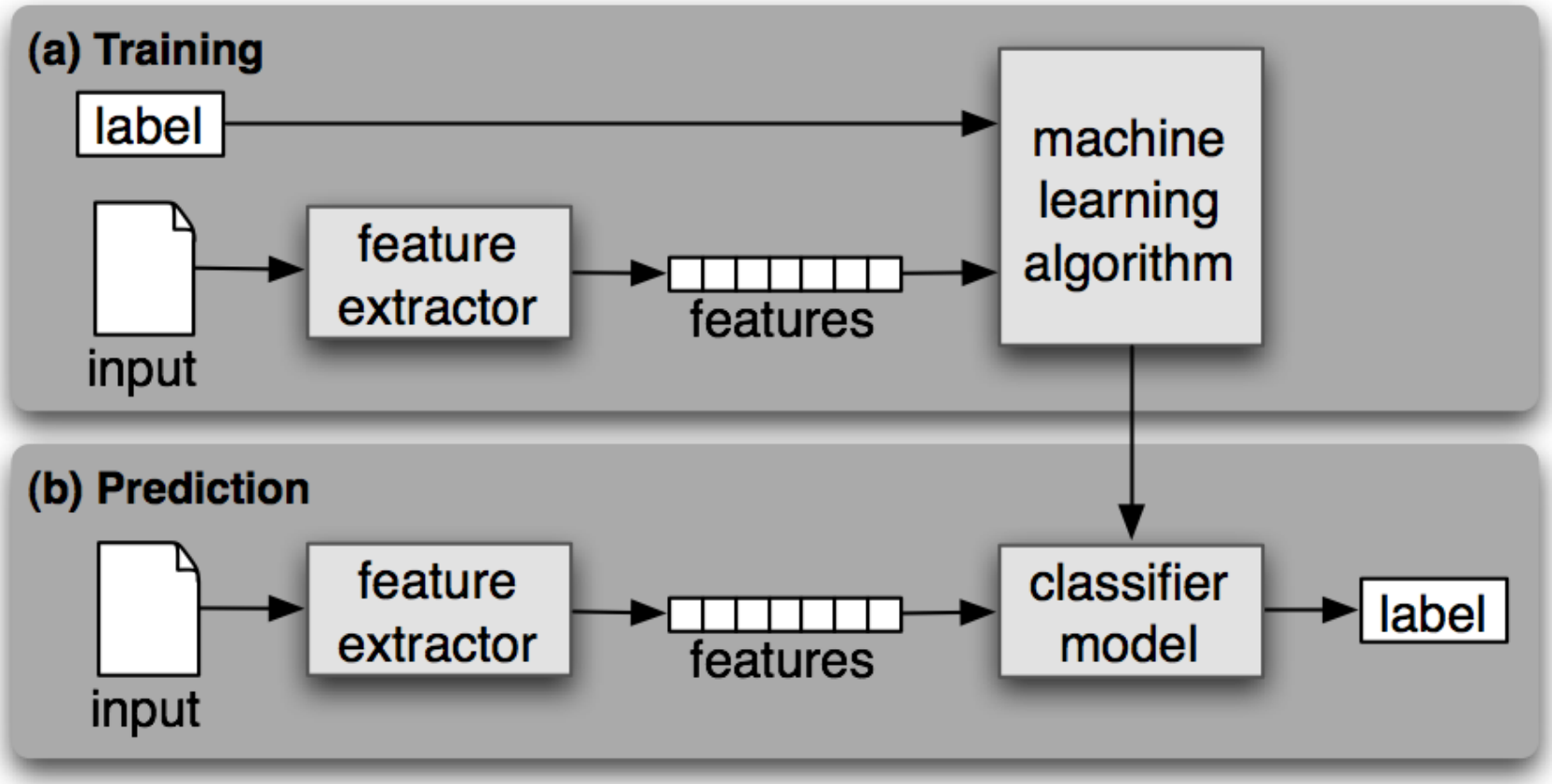


Tribe	Foundation	The main algorithm
Symbolists	Logic, philosophy	Induction
Connectionists	Neuroscience	Backpropagation
Evolutionaries	Evolutionary biology	Genetic programming
Bayesians	Statistics	Probabilistic inference
Analogizers	Psychology	Kernel machines

NB: We'll cover one Bayesian algorithm and one Symbolists algorithm

Outline

Training and prediction



Training and prediction

- A supervised model first needs to be trained on labeled data, and after that the model can be used for prediction (classification or regression) on previously unseen data
- Each data instance is represented as a vector of features $\mathbf{x}$, where each feature encodes one aspect of the input (e.g., height, weight, income, ...)
- Feature extractor generates the feature vector $\mathbf{x}$ for each data instance (this step is identical for both training and prediction)
- During training, the model is fed data instances $\mathbf{x}$ and their labels y, i.e., the pairs $(\mathbf{x}, y)$
- After the model has been trained, the same model is used for prediction
- During prediction, the model is presented only data instances $\mathbf{x}$ as input, and it generates y as output

Overfitting

- Our main aim is for the model to generalize well
- If the model is very complex, it will fit the data very well, and make (nearly) perfect predictions
- If the model is too complex, it will adapt to much to the data it has been trained on, but give poor predictions on unseen data $\Rightarrow$ overfitting
- An overfitted model generalizes poorly and is therefore useless!
- Overfitting is the biggest problem of machine learning
- To avoid overfitting, we have to test how well the model will work on unseen data $\Rightarrow$ cross-validation

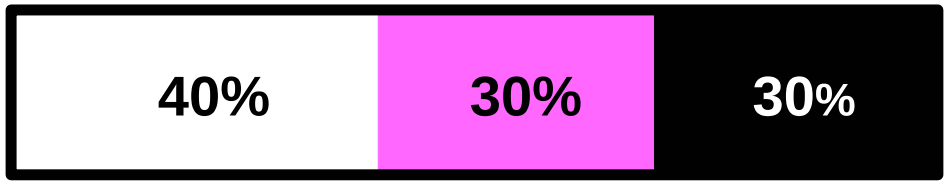
Cross-validation

- As unseen data are now available, we take out a portion of available data to act as unseen data
- We split the dataset into a training set and a test set (typically 70%–30%)

- We train the model on the train set, use this model to obtain predictions on the test set, and then compute the accuracy (or error)
- Model's accuracy/error on the test set indicates how well the model generalizes (the model has previously not seen the data from the test set!)

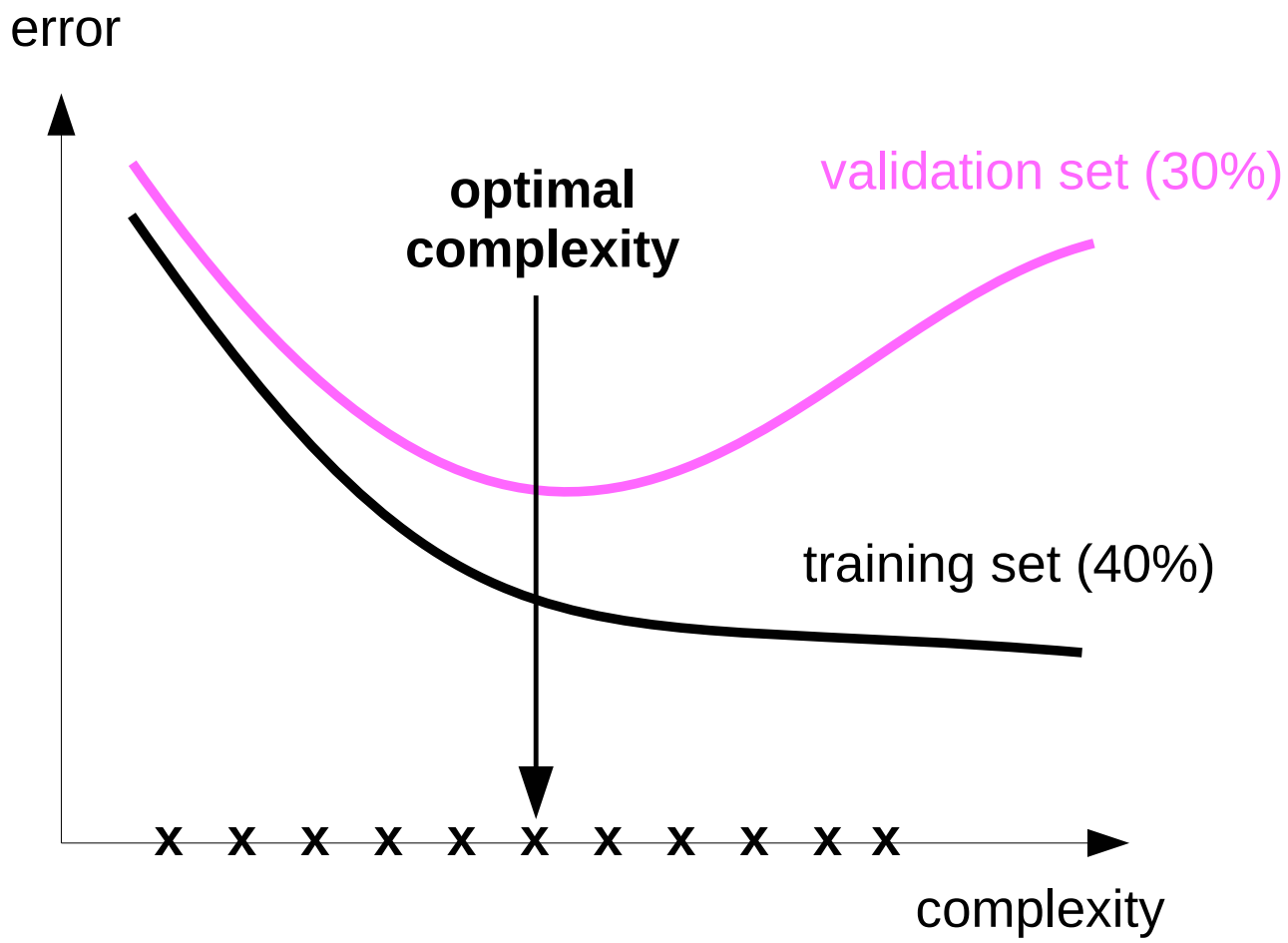
Cross-validation

- Some models allow their complexity to be fine-tuned (e.g., pruning of decision trees)
- To find the optimal complexity, we need a third dataset on which we can test the accuracy/error of the model: the validation set
- We split the dataset into a training set, a validation set, and a test set, e.g.:

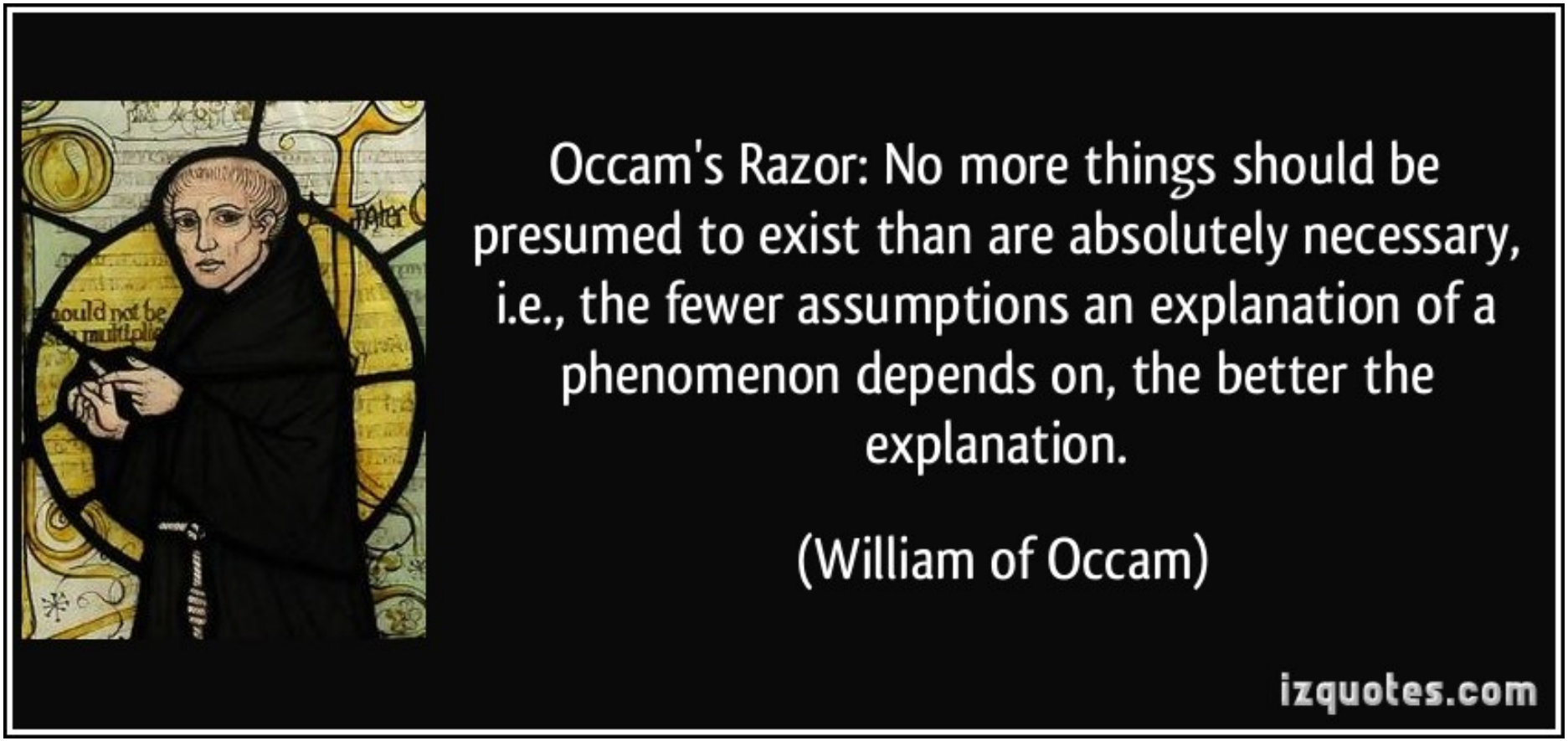


- We train models of varying complexity on the train set and test each on the validation set
- After we have found the best-performing model on the validation set, we compute the final accuracy/error of this model on the test set

Cross-validation



Occam's razor (Principle of parsimony)

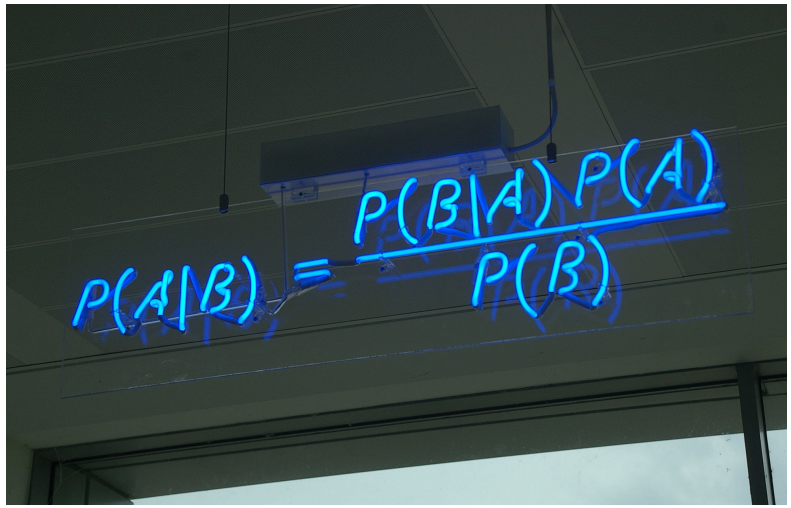


Outline

Bayes rule and Bayes classifier

- Bayes classifier is a simple supervised machine learning algorithm based on **Bayes rule** (theorem)

Thomas Bayes (1702–1761)



- Bayes classifier utilizes the same ideas we discussed in the context of modeling uncertain knowledge (evidences contribute to posterior probability of a hypothesis), but now we wish to obtain these probabilities automatically from data

Refresher: Bayes rule

- Bayes rule (probability of hypothesis H given evidence E):

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} = \frac{P(E|H)P(H)}{P(E|H)P(H) + P(E|\neg H)P(\neg H)}$$

- For multiple hypotheses (that make up a complete system of events):

$$P(H_i|E) = \frac{P(E|H_i)P(H_i)}{\sum_j P(E|H_j)P(H_j)}$$

- For multiple hypotheses and multiple (n) evidences:

$$P(H_i|E_1, E_2, \ldots, E_n) = \frac{P(E_1, E_2, \ldots, E_n|H_i)P(H_i)}{\sum_j P(E_1, E_2, \ldots, E_n|H_j)P(H_j)}$$

- Assumption of conditional independence of evidences for a hypothesis:

$$P(H_i|E_1, E_2, \ldots, E_n) = \frac{P(E_1|H_i)P(E_2|H_i)\cdots P(E_n|H_i)P(H_i)}{\sum_j P(E_1|H_j)P(E_2|H_j)\cdots P(E_n|H_j)P(H_j)}$$

Bayes rule as a classification model

- We can use the Bayes rule directly as a classification model
  - Hypothesis $H \rightarrow$ class y
  - Evidences $E_1, \ldots, E_n \rightarrow$ instance $\mathbf{x} = (x_1, \ldots, x_n)$
- Model:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y)P(y)}{\sum_{y'} P(\mathbf{x}|y')P(y')}$$

  where
  - $P(y)$ is the class prior
  - $P(\mathbf{x}|y)$ is the class likelihood
  - $P(y|\mathbf{x})$ is the class posterior
- We're interested in obtaining the posterior probability $P(y|\mathbf{x})$: **what is the probability that instance $\mathbf{x}$ belongs to class y** (i.e., has the corresponding class label)
- Note that this is the opposite of class likelihood $P(\mathbf{x}|y)$: the probability that class y contains instance $\mathbf{x}$

MAP hypothesis

- $P(y|\mathbf{x})$ gives us the probability that instance $\mathbf{x}$ belongs to class y
- This is useful, but often it suffices to merely classify the instance
- The optimal classification decision is the one which assigns the class for which the **posterior probability is maximal**
- Such a classification decision is termed maximum aposteriori hypothesis or MAP hypothesis
- The denominator of Bayes rule is the same for all classes y – for computing MAP, we can drop it and compute only the numerator
- Bayes classifier model returning the MAP hypothesis:

$$h_{\text{MAP}} = \underset{y}{\operatorname{argmax}}\, P(y|\mathbf{x}) = \underset{y}{\operatorname{argmax}}\, P(\mathbf{x}|y)P(y)$$

  where function $\operatorname{argmax}_x f(x)$ returns x that maximizes $f(x)$
- h_{MAP} is the class (label) to which the instance most probably belongs

Naïve Bayes classifier

- Same as in the case with multiple evidences, we introduce the conditional independence assumption: features x_j are conditionally independent given class y, i.e., it holds $P(x_j, x_k|y) = P(x_j|y)P(x_k|y)$
- Under this assumption, Bayes classifier model becomes:

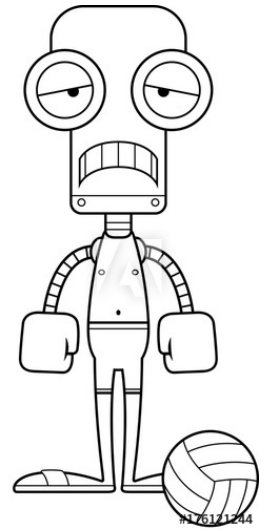
$$h_{\text{MAP}} = \operatorname*{argmax}_y P(y|\mathbf{x}) =$$

$$= \operatorname*{argmax}_y P(\mathbf{x}|y)P(y) \overset{c.i.}{=} \operatorname*{argmax}_y P(y) \prod_{j=1}^{n} P(x_j|y)$$

- This is the so-called **naïve Bayes classifier**
- It's "naive" because the conditional independence assumption generally does not hold (generally, there are interactions between features)
- Nonetheless, in practice, the model works quite well

Example: A day for beach volleyball

Day	x_1 Weather	x_2 Temp	x_3 Humidity	x_4 Wind	y Volleyball?
1	sunny	high	high	weak	no
2	sunny	high	high	strong	no
3	cloudy	high	high	weak	yes
4	rainy	medium	high	weak	yes
5	rainy	low	normal	weak	yes
6	rainy	low	normal	strong	no
7	cloudy	low	normal	strong	yes
8	sunny	medium	high	weak	no
9	sunny	low	normal	weak	yes
10	rainy	medium	normal	weak	yes
11	sunny	medium	normal	strong	yes
12	cloudy	medium	high	strong	yes
13	cloudy	high	normal	weak	yes
14	rainy	medium	high	strong	no



Example: A day for beach volleyball

- We wish to classify a new (fifteenth) instance:

$$\mathbf{x} = (\underbrace{\text{Weather}=\text{sunny}}_{x_1}, \underbrace{\text{Temp}=\text{low}}_{x_2}, \underbrace{\text{Humidity}=\text{high}}_{x_3}, \underbrace{\text{Wind}=\text{strong}}_{x_4})$$

- The MAP hypothesis is:

$$\begin{aligned} h_{\text{MAP}} &= \underset{y \in \{\text{yes},\text{no}\}}{\operatorname{argmax}} P(y|\mathbf{x}) \\ &\overset{c.i.}{=} \underset{y \in \{\text{yes},\text{no}\}}{\operatorname{argmax}} P(y) \prod_{j=1}^{n} P(x_j|y) \\ &= \underset{y \in \{\text{yes},\text{no}\}}{\operatorname{argmax}} P(y)P(\text{sunny}|y)P(\text{low}|y)P(\text{high}|y)P(\text{strong}|y) \end{aligned}$$

- So, we need to compute:

$$\begin{aligned} y = \text{yes}: &\quad P(\text{yes})P(\text{sunny}|\text{yes})P(\text{low}|\text{yes})P(\text{high}|\text{yes})P(\text{strong}|\text{yes}) \\ y = \text{no}: &\quad P(\text{no})P(\text{sunny}|\text{no})P(\text{low}|\text{no})P(\text{high}|\text{no})P(\text{strong}|\text{no}) \end{aligned}$$

Training a Bayes classifier

- Training a Bayes classifier amounts to **estimating probabilities** based on a set of training examples
- **Q**: How to estimate the **prior probabilities** $P(\text{yes})$ and $P(\text{no})$?
- The simplest estimate: **relative frequencies** (the ratio of examples labeled with $y = \text{yes}$ and $y = \text{no}$, respectively)

$$P(\text{yes}) = \frac{9}{14} = 0.64$$
$$P(\text{no}) = \frac{5}{14} = 0.36$$

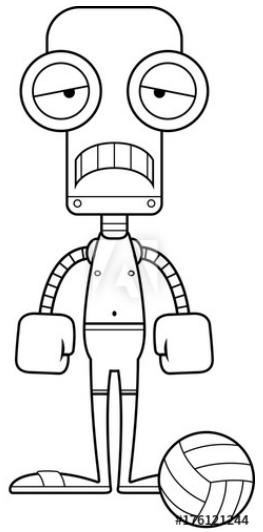
- We can estimate **likelihoods** in a similar way. E.g., $P(\text{sunny}|\text{yes})$ is the ratio of examples with $x_1 = \text{sunny}$ in the subset of examples labeled with $y = \text{yes}$

$$P(\text{sunny}|\text{yes}) = \frac{2}{9} = 0.22$$
$$P(\text{sunny}|\text{no}) = \frac{3}{5} = 0.6$$

Example: A day for beach volleyball

Day	x_1 Weather	x_2 Temp	x_3 Humidity	x_4 Wind	y Volleyball?
1	sunny	high	high	weak	no
2	sunny	high	high	strong	no
3	cloudy	high	high	weak	yes
4	rainy	medium	high	weak	yes
5	rainy	low	normal	weak	yes
6	rainy	low	normal	strong	no
7	cloudy	low	normal	strong	yes
8	sunny	medium	high	weak	no
9	sunny	low	normal	weak	yes
10	rainy	medium	normal	weak	yes
11	sunny	medium	normal	strong	yes
12	cloudy	medium	high	strong	yes
13	cloudy	high	normal	weak	yes
14	rainy	medium	high	strong	no

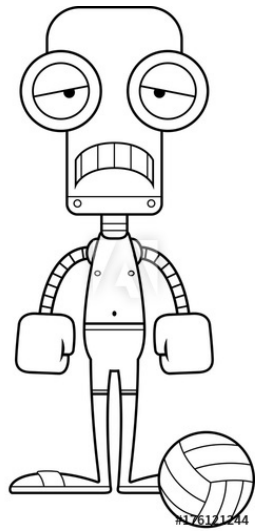
$$P(\text{yes}) = \frac{9}{14}$$



Example: A day for beach volleyball

Day	x_1 Weather	x_2 Temp	x_3 Humidity	x_4 Wind	y Volleyball?
1	sunny	high	high	weak	no
2	sunny	high	high	strong	no
3	cloudy	high	high	weak	yes
4	rainy	medium	high	weak	yes
5	rainy	low	normal	weak	yes
6	rainy	low	normal	strong	no
7	cloudy	low	normal	strong	yes
8	sunny	medium	high	weak	no
9	sunny	low	normal	weak	yes
10	rainy	medium	normal	weak	yes
11	sunny	medium	normal	strong	yes
12	cloudy	medium	high	strong	yes
13	cloudy	high	normal	weak	yes
14	rainy	medium	high	strong	no

$$P(\text{sunny}|\text{yes}) = \frac{2}{9}$$



Example: A day for beach volleyball

- Computing all the required probabilities:

$$P(y = \text{yes}) = 9/14 = 0.64 \qquad P(y = \text{no}) = 5/14 = 0.36$$
$$P(x_1 = \text{sunny}|y = \text{yes}) = 2/9 = 0.22 \qquad P(x_1 = \text{sunny}|y = \text{no}) = 3/5 = 0.6$$
$$P(x_2 = \text{low}|y = \text{yes}) = 3/9 = 0.33 \qquad P(x_2 = \text{low}|y = \text{no}) = 1/5 = 0.2$$
$$P(x_3 = \text{high}|y = \text{yes}) = 3/9 = 0.33 \qquad P(x_3 = \text{high}|y = \text{no}) = 4/5 = 0.8$$
$$P(x_4 = \text{strong}|y = \text{yes}) = 3/9 = 0.33 \qquad P(x_4 = \text{strong}|y = \text{no}) = 3/5 = 0.6$$

- Computing the MAP hypothesis:

$$y = \text{yes}: \quad P(\text{yes})P(\text{sunny}|\text{yes})P(\text{low}|\text{yes})P(\text{high}|\text{yes})P(\text{strong}|\text{yes}) = 0.64 \cdot 0.22 \cdot 0.33 \cdot 0.33 \cdot 0.33 = 0.0051$$
$$y = \text{no}: \quad P(\text{no})P(\text{sunny}|\text{no})P(\text{low}|\text{no})P(\text{high}|\text{no})P(\text{strong}|\text{no}) = 0.36 \cdot 0.6 \cdot 0.2 \cdot 0.8 \cdot 0.6 = 0.0207$$
$$\Rightarrow h_{\text{MAP}} = \text{no}$$

- If required, we can also compute the posterior class probabilities:

$$P(\text{yes}|\mathbf{x}) = \frac{0.0051}{0.0051 + 0.0207} = 0.198 \qquad P(\text{no}|\mathbf{x}) = \frac{0.0207}{0.0051 + 0.0207} = 0.802$$

Maximum likelihood estimation

- Estimating probabilities via relative frequencies is known as **maximum likelihood estimation (MLE)**, because such an estimate maximizes the likelihood of model's parameters (data probability under the model)

Maximum likelihood estimate (MLE)

Maximum likelihood estimate for prior class probabilities and class likelihoods of a Bayes classifier are:

$$P(y = v) = \frac{|D_{y=v}|}{|D|}$$

$$P(x_j = w | y = v) = \frac{|D_{y=v \wedge x_j=w}|}{|D_{y=v}|}$$

where D is a set of examples, and D_P is a subset of examples satisfying P

Overfitting a Bayes classifier

- MLE is the simplest parameter estimation method (parameters = probabilities of a Bayes classifier), but it's prone to **overfitting**
- **Overfitting $\Leftrightarrow$ the model got too attuned to the training set**
- In case of Bayes classifier: class likelihoods for combinations not occurring in train data will be equal to zero
- E.g., we wish to classify the instance:

$$\mathbf{x} = (\underset{x_1}{\text{Weather}=\text{cloudy}}, \underset{x_2}{\text{Temp}=\text{low}}, \underset{x_3}{\text{Humidity}=\text{high}}, \underset{x_4}{\text{Wind}=\text{strong}})$$

- The training set has no examples with Weather=cloudy and $y = \text{no}$, thus:

$$P(\text{cloudy}|\text{no}) = \frac{0}{5} = 0$$

and consequently:

$$y = \text{no}: \quad P(\text{no})P(\text{cloudy}|\text{no})P(\text{low}|\text{no})P(\text{high}|\text{no})P(\text{strong}|\text{no}) = 0.36 \cdot 0.0 \cdot 0.2 \cdot 0.8 \cdot 0.6 = 0.0$$

Smoothing

- Although some feature-label combinations have never occurred in the training set, we shouldn't consider these combinations as impossible
- The remedy is to slightly redistribute the probability mass from observed to unobserved combinations
- Doing so is called **smoothing**
- The simplest smoothing method is **Laplace smoothing**

Class likelihood estimates with Laplace smoothing

$$P(x_j = w|y = v) = \frac{|D_{y=v \wedge x_j=w}| + \alpha}{|D_{y=v}| + \alpha|V(x_j)|}$$

where $V(x_j)$ is the set of possible values of feature (attribute) x_j, and $\alpha \geq 0$ is the smoothing parameter

- Setting $\alpha = 0$ gives non-smoothed estimates, while setting $\alpha = 1$ gives the so-called **add-one smoothing**

Add-one smoothing – example

- Non-smoothed probability distribution:

$$P(\text{sunny}|\text{no}) = \frac{3}{5} = 0.6$$
$$P(\text{cloudy}|\text{no}) = \frac{0}{5} = 0$$
$$P(\text{rainy}|\text{no}) = \frac{2}{5} = 0.4$$

- Laplace smoothing with $\alpha = 1$ (where $|V(\text{Weather})| = 3$):

$$P(\text{sunny}|\text{no}) = \frac{3+1}{5+3} = 0.5$$
$$P(\text{cloudy}|\text{no}) = \frac{0+1}{5+3} = 0.125$$
$$P(\text{rainy}|\text{no}) = \frac{2+1}{5+3} = 0.375$$

$\Rightarrow$ probability redistribution from observed to unobserved events

$\Rightarrow$ the more training data we have, the smaller is the effect of smoothing

Add-one smoothing – example

$$\mathbf{x} = (\underset{x_1}{\text{Weather}=\text{cloudy}}, \underset{x_2}{\text{Temp}=\text{low}}, \underset{x_3}{\text{Humidity}=\text{high}}, \underset{x_4}{\text{Wind}=\text{strong}})$$

$$P(y = \text{yes}) = 9/14 = 0.64 \qquad P(y = \text{no}) = 5/14 = 0.36$$

$$P(x_1 = \text{cloudy}|y = \text{yes}) = \frac{3+1}{9+3} = 0.33 \qquad P(x_1 = \text{cloudy}|y = \text{no}) = \frac{0+1}{5+3} = 0.125$$

$$P(x_2 = \text{low}|y = \text{yes}) = \frac{3+1}{9+3} = 0.33 \qquad P(x_2 = \text{low}|y = \text{no}) = \frac{1+1}{5+3} = 0.25$$

$$P(x_3 = \text{high}|y = \text{yes}) = \frac{3+1}{9+2} = 0.36 \qquad P(x_3 = \text{high}|y = \text{no}) = \frac{4+1}{5+2} = 0.71$$

$$P(x_4 = \text{strong}|y = \text{yes}) = \frac{3+1}{9+2} = 0.36 \qquad P(x_4 = \text{strong}|y = \text{no}) = \frac{3+1}{5+2} = 0.57$$

$$y = \text{yes}: \quad P(\text{yes})P(\text{cloudy}|\text{yes})P(\text{low}|\text{yes})P(\text{high}|\text{yes})P(\text{strong}|\text{yes}) = 0.64 \cdot 0.33 \cdot 0.33 \cdot 0.36 \cdot 0.36 = 0.009$$

$$y = \text{no}: \quad P(\text{no})P(\text{cloudy}|\text{no})P(\text{low}|\text{no})P(\text{high}|\text{no})P(\text{strong}|\text{no}) = 0.36 \cdot 0.125 \cdot 0.25 \cdot 0.71 \cdot 0.57 = 0.005$$

$$\Rightarrow h_{\text{MAP}} = \text{yes}$$

Preventing underflow

- Computing h_{MAP} requires us to multiply the likelihoods in the numerator of the Bayes rule

$$h_{\text{MAP}} = \operatorname*{argmax}_y P(y) \prod_{j=1}^{n} P(x_j|y)$$

- This can result in an underflow when there are many features
- To avoid this, we work in the logarithmic domain (the logarithm is a monotonically increasing function and thus does not affect the argmax)

$$\begin{aligned} h_{\text{MAP}} &= \ln \left(\operatorname*{argmax}_y P(y) \prod_{j=1}^{n} P(x_j|y) \right) \\ &= \operatorname*{argmax}_y \ln \left(P(y) \prod_{j=1}^{n} P(x_j|y) \right) \\ &= \operatorname*{argmax}_y \left(\ln P(y) + \sum_{j=1}^{n} \ln P(x_j|y) \right) \end{aligned}$$

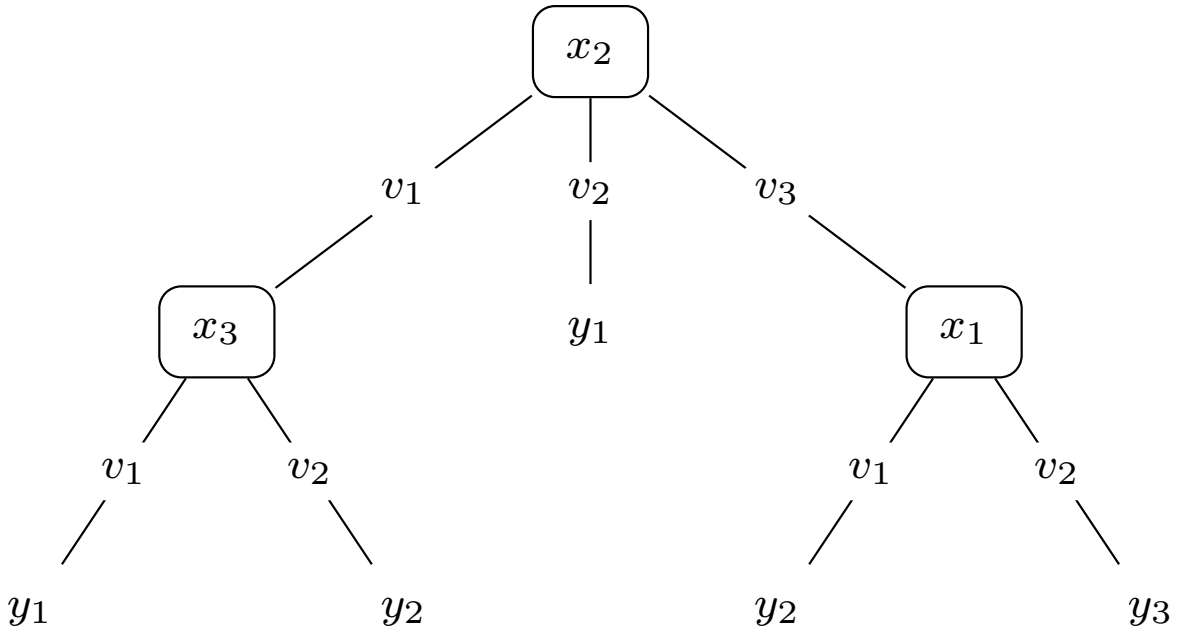
Naïve Bayes classifier – remarks

- **Strengths**:
  - a fast and simple algorithm (models are easy to train)
  - can work with multivalued features and multiclass problems
  - works well with a large number of features (scales linearly)
  - gives good predictions if feature independence assumption holds
- **Weaknesses**:
  - in reality, features are rarely conditionally independent. The more the data deviates from this assumption, the worse the predictions
- Typical applications: document classification, spam filtering
- If conditional independence assumptions don't hold, one can use the **semi-naïve Bayes classifier** (a more complex model)
- If features are continuous, class likelihoods are modeled as Gaussian distributions (**Gaussian Bayes classifier**)
- Bayes classifiers belongs to the broad family of machine learning models called **probabilistic graphical models**

Outline

Classification using a decision tree

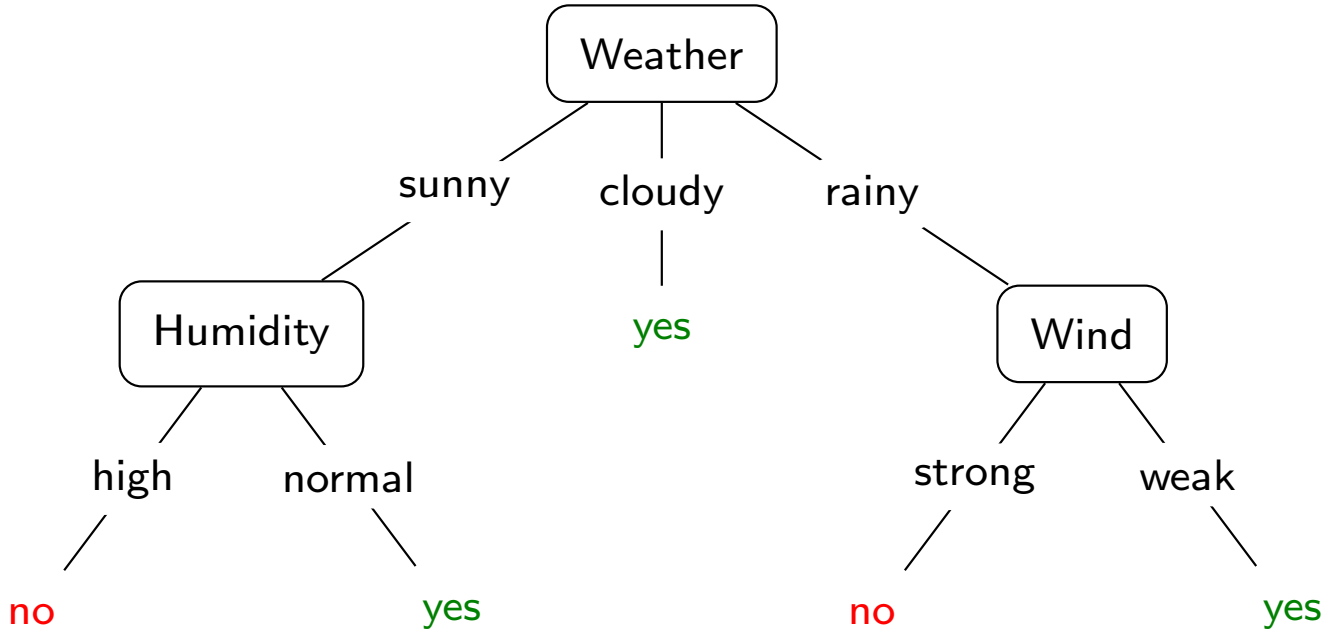
- **Inner nodes** correspond to features (attributes), **branches** beneath each node correspond to values of the corresponding features, while **leaves** correspond to classification decisions (class labels)



- An instance is classified by comparing its feature values against thee branches, starting from the root and traversing down to the leaves
- Upon reaching a leaf, the instance is assigned the label of the leaf

Decision tree – example

- Decision tree for "A day for beach volleyball":



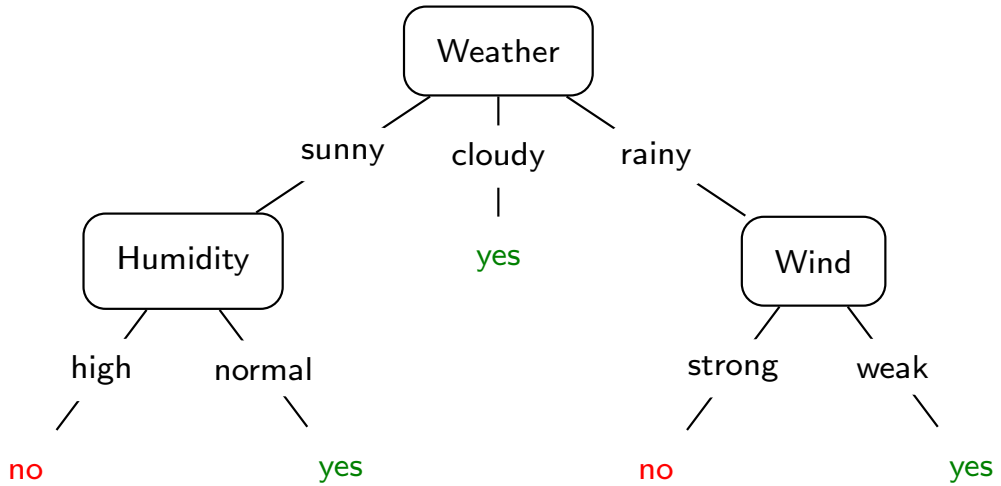
- Classifying a new instance:

$$\mathbf{x} = (\underset{x_1}{\text{Weather}=\text{sunny}}, \text{Temp}=\underset{x_2}{\text{low}}, \underset{x_3}{\text{Humidity}=\text{high}}, \underset{x_4}{\text{Wind}=\text{strong}})$$

$\Rightarrow$ no

Decision trees and rule-based classification

- Decision trees relate to **rule-based classification**
- A decision tree can be thought of as a compact representation of rules
- Every path from the root to a leaf corresponds to one rule in the form of conjunctions of conditions on feature values
- A set of rules for the same class label is a disjunction of conjunctions



$$\big((\text{Weather}{=}\text{sunny} \wedge \text{Humidity}{=}\text{normal}) \vee$$
$$(\text{Weather}{=}\text{cloudy}) \vee (\text{Weather}{=}\text{rainy} \wedge \text{Wind}{=}\text{strong})\big) \rightarrow (y = \text{yes})$$

ID3 algorithm

- **ID3 algorithm (Iterative Dichotomiser 3)** is a simple algorithm for building decision trees proposed by Ross Quinlan in 1986

Machine Learning 1: 81–106, 1986

Induction of Decision Trees

J.R. QUINLAN (munnari!nswitgould.oz!quinlan@seismo.css.gov)
Centre for Advanced Computing Sciences, New South Wales Institute of Technology, Sydney 2007, Australia

(Received August 1, 1985)

Key words: classification, induction, decision trees, information theory, knowledge acquisition, expert systems

Abstract. The technology for building knowledge-based systems by inductive inference from examples has been demonstrated successfully in several practical applications. This paper summarizes an approach to synthesizing decision trees that has been used in a variety of systems, and it describes one such system, ID3, in detail. Results from recent studies show ways in which the methodology can be modified to deal with information that is noisy and/or incomplete. A reported shortcoming of the basic algorithm is discussed and two means of overcoming it are compared. The paper concludes with illustrations of current research directions.

Quinlan, J. R. (1986). **Induction of decision trees**. Machine learning, 1(1), 81-106.

Building decision trees

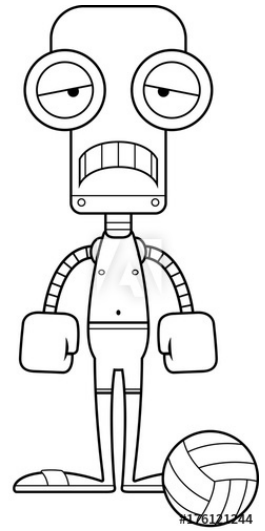
- Classification using decision trees is straightforward, the main question is **how to build decision trees** from data
- Every decision tree **recursively partitions** the set of training examples
- For every training set, there is a finite number of decision trees that perfectly classify each example from the training set (assuming no identical examples with distinct labels)
- If there are many features and possible values, the number of such trees becomes prohibitively large
- In practice, we prefer trees with as fewer number of nodes: simpler trees correspond to simpler hypotheses which **generalize better**, i.e., they are more prone to **overfitting**
- However, finding a minimum-size tree that perfectly classifies all training examples is a shown to be an **NP-hard problem** $\Rightarrow$ we need a heuristic approach

Building decision trees

- The tree structure is entirely determined by what feature is tested in what node
- For the root node we choose between n features, for first-level nodes between $n-1$ features, etc.
- We need a (numeric) **criterion for selecting the best feature** by which to split the data in the current node
- Intuitively, we prefer features that **discriminate well** between training examples according to label y (ideally so that all examples with a certain feature value belong to an identical class)

Example: A day for beach volleyball

Day	x_1 Weather	x_2 Temp	x_3 Humidity	x_4 Wind	y Volleyball?
1	sunny	high	high	weak	no
2	sunny	high	high	strong	no
3	cloudy	high	high	weak	yes
4	rainy	medium	high	weak	yes
5	rainy	low	normal	weak	yes
6	rainy	low	normal	strong	no
7	cloudy	low	normal	strong	yes
8	sunny	medium	high	weak	no
9	sunny	low	normal	weak	yes
10	rainy	medium	normal	weak	yes
11	sunny	medium	normal	strong	yes
12	cloudy	medium	high	strong	yes
13	cloudy	high	normal	weak	yes
14	rainy	medium	high	strong	no



Feature selection – example

Feature	Value	# yes	# no
Weather	sunny	2	3
	cloudy	4	0
	rainy	3	2
Temp	high	2	2
	medium	4	2
	low	3	1
Humidity	high	3	4
	normal	6	1
Wind	strong	3	3
	weak	6	2

- Which feature to select for the root node?
- A split by **Weather** might be good because Weather=cloudy gives only the positive examples, however Weather=sunny and Weather=rainy discriminate rather poorly

Entropy of a dataset

- The best split is by a feature for which all examples in each branch belong to the same class
- The worst split is by a feature for which in each branch examples are uniformly distributed across different classes
- This criterion can be expressed as **minimizing the entropy** of labels

Entropy of a dataset

Let D contain labeled examples from K classes. The entropy of D is:

$$E(D) = -\sum_{i=1}^{K} P(y=i) \log_2 P(y=i)$$

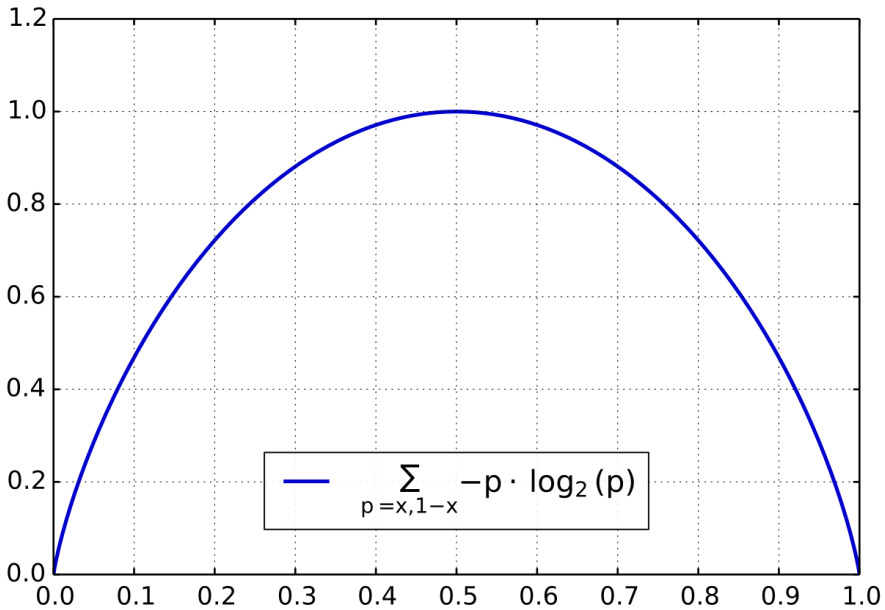
where $P(y=i)$ is the probability of class $y=i$. Per definition, $0 \log_2 0 \doteq 0$.

- a perfect split will have $E = 0$, whereas a non-discriminative split will have $E = \log_2(K)$

Entropy of a dataset

- In case of binary classification ($K = 2$):

$$E(D) = -\sum_{y=\{\text{yes},\text{no}\}} P(y) \log_2 P(y)$$



- The best split has $E = 0$, the worst has $E = 1$

Entropy of a dataset – example

Feature	Value	# yes	# no	$E(D)$
Weather	sunny	2	3	0.971
	cloudy	4	0	0
	rainy	3	2	0.971
Temp	high	2	2	1
	medium	4	2	0.918
	low	3	1	0.818
Humidity	high	3	4	0.985
	normal	6	1	0.592
Wind	strong	3	3	1
	weak	6	2	0.811

- **Weather** attains the lowest entropy for "cloudy", but we have to also consider its other two values
- Rarely will entropy of a feature be zero for all its values
- We select the feature that **reduces the entropy the most**, while splits in subsequent nodes will eventually reduce it to zero

Information gain

- **Information gain** criterion measures the expected reduction in entropy that is due to splitting the dataset by values of a certain feature

Information gain

Information gain (IG) of feature x on the set of examples D is:

$$IG(D,x) = \underbrace{E(D)}_{\substack{\text{entropy of the} \\ \text{initial dataset}}} - \underbrace{\sum_{v \in V(x)} \frac{|D_{x=v}|}{|D|} E(D_{x=v})}_{\substack{\text{expected entropy after splitting} \\ \text{the dataset by values of the feature}}}$$

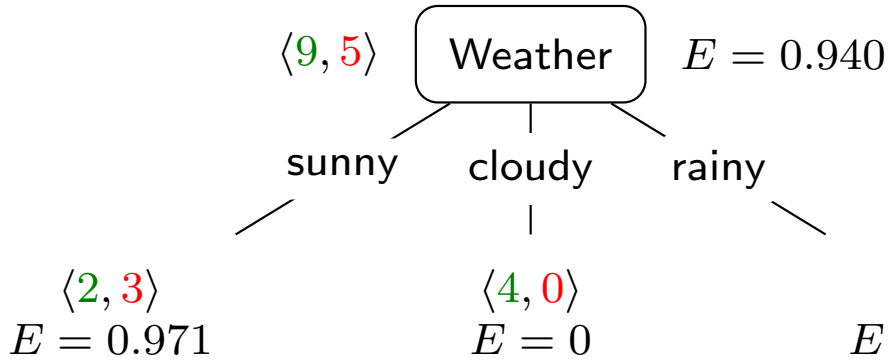
where $E(D)$ is the entropy of the dataset D, D_P is the subset of examples satisfying condition P, and $V(x)$ is the set of possible values of feature x.

Example: A day for beach volleyball

- Let $\langle m, n \rangle$ denote dataset D with m positive examples ($y = \text{yes}$) and n negative examples ($y = \text{no}$)
- Entropy of the initial dataset D is:

$$E(D) = E(\langle 9, 5 \rangle) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940$$

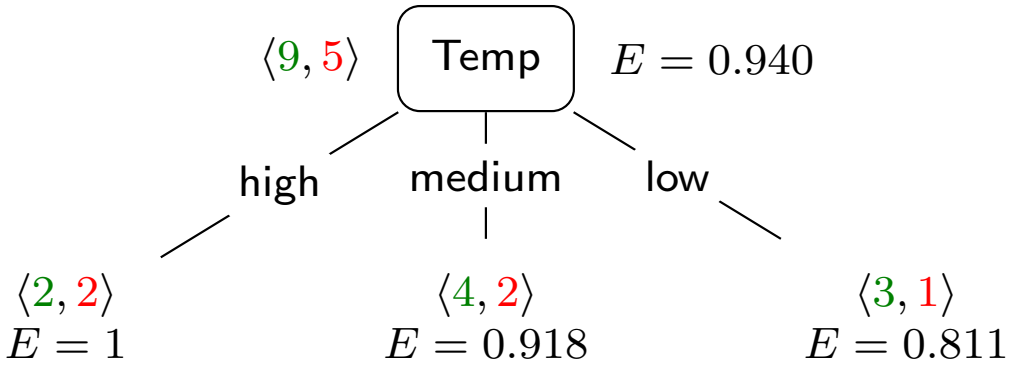
- We start by selecting the best feature (as per IG) for the root node
- IG for **Weather** feature on set D:



$$IG(D, \text{Weather}) = 0.940 - \frac{5}{14} \cdot 0.971 - \frac{4}{14} \cdot 0 - \frac{5}{14} \cdot 0.971 = 0.246$$

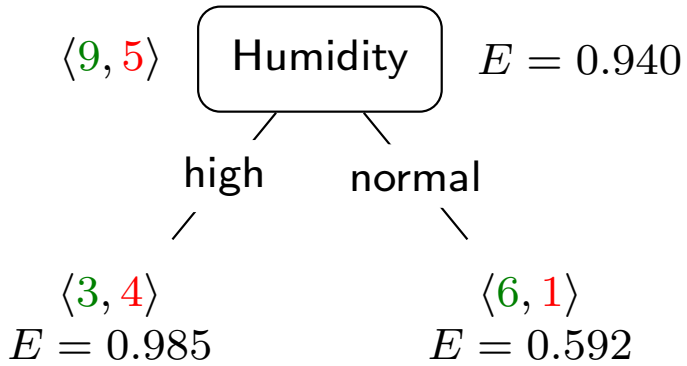
Example: A day for beach volleyball

- IG for **Temp** feature on set D:



$$IG(D, \text{Temp}) = 0.940 - \frac{4}{14} \cdot 1 - \frac{6}{14} \cdot 0.918 - \frac{4}{14} \cdot 0.811 = 0.029$$

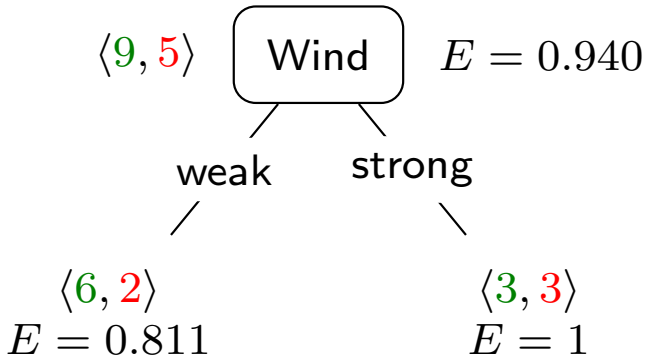
- IG for **Humidity** feature on set D:



$$IG(D, \text{Humidity}) = 0.940 - \frac{7}{14} \cdot 0.985 - \frac{7}{14} \cdot 0.592 = 0.151$$

Example: A day for beach volleyball

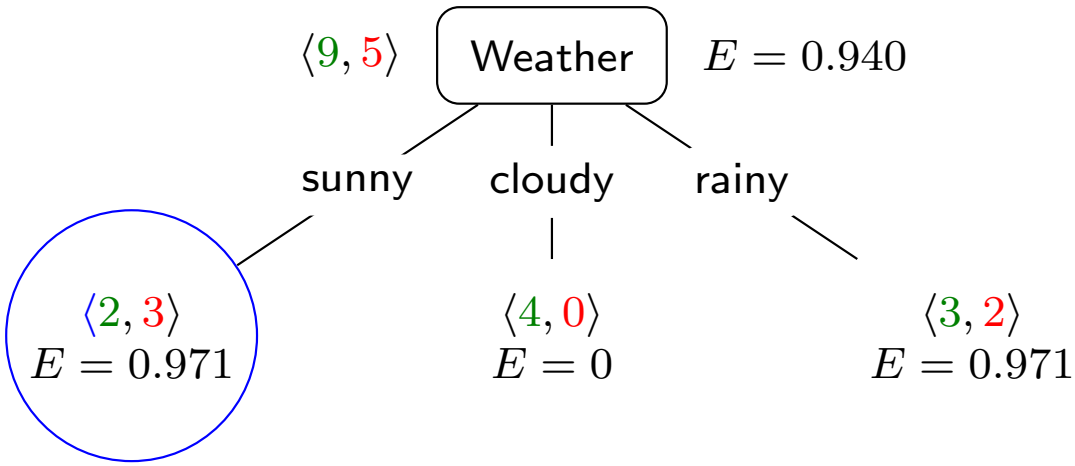
- IG for **Wind** feature on set D:



$$IG(D, \text{Wind}) = 0.940 - \frac{8}{14} \cdot 0.811 - \frac{6}{14} \cdot 1 = 0.048$$

- Information gain (expected reduction in entropy) is the largest for **Weather** feature, so this feature will be in the root node of our tree
- We recursively repeat this procedure on subsets of the three branches outgoing from the **Weather** node, adding children nodes and selecting among the remaining three features

Example: A day for beach volleyball

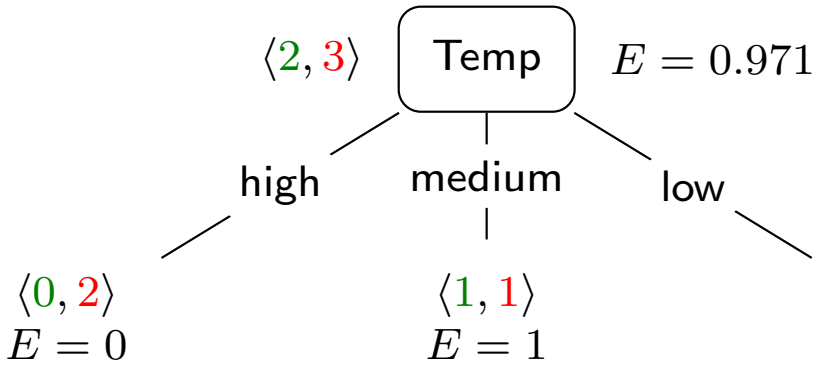


$D_{\text{Weather=sunny}}$

Day	x_2 Temp	x_3 Humidity	x_4 Wind	y Volleyball?
1	high	high	weak	no
2	high	high	strong	no
8	medium	high	weak	no
9	low	normal	weak	yes
11	medium	normal	strong	yes

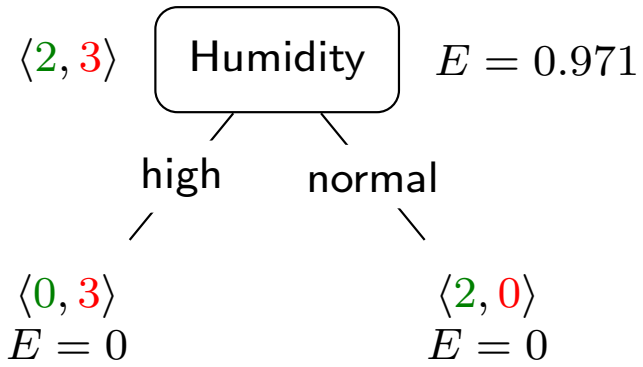
Example: A day for beach volleyball

- IG **Temp** feature on subset $D_{\text{Weather=sunny}}$:



$$IG(D_{\text{Weather=sunny}}, \text{Temp}) = 0.971 - \frac{2}{5} \cdot 0 - \frac{2}{5} \cdot 1 - \frac{1}{5} \cdot 0 = 0.571$$

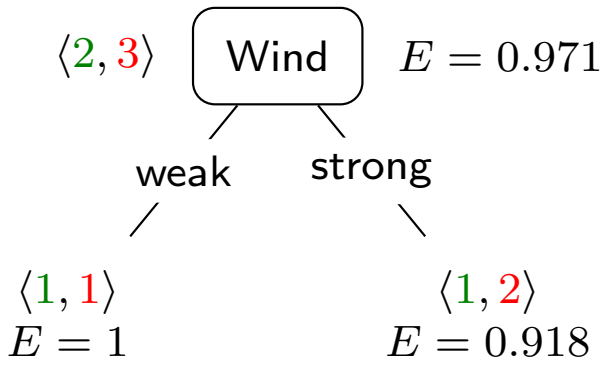
- IG for **Humidity** feature on subset D:



$$IG(D_{\text{Weather=sunny}}, \text{Humidity}) = 0.971 - \frac{2}{5} \cdot 0 - \frac{2}{5} \cdot 0 = 0.971$$

Example: A day for beach volleyball

- IG for **Wind** feature on subset $D_{\text{Weather=sunny}}$:

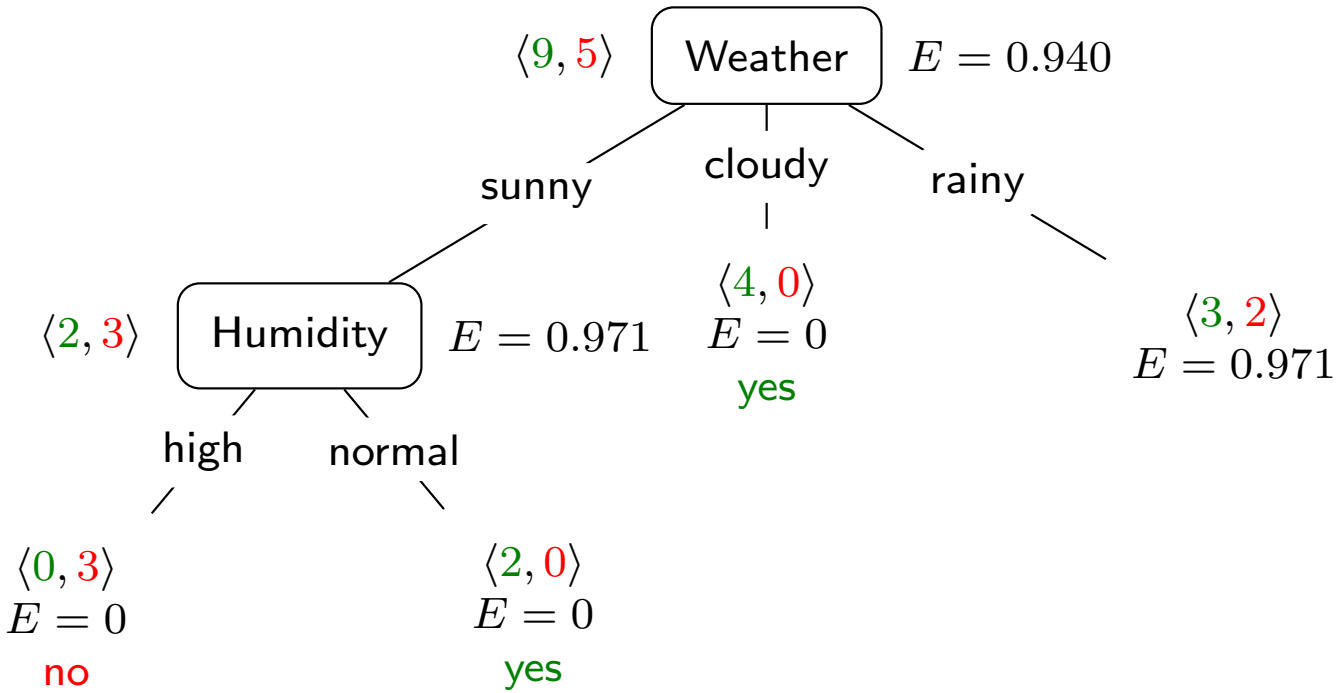


$$IG(D_{\text{Weather=sunny}}, \text{Wind}) = 0.971 - \frac{2}{5} \cdot 1 - \frac{3}{5} \cdot 0.918 = 0.02$$

- Information gain (expected reduction in entropy) is the largest for **Humidity** feature, so we select this feature for the node beneath the Weather=sunny branch

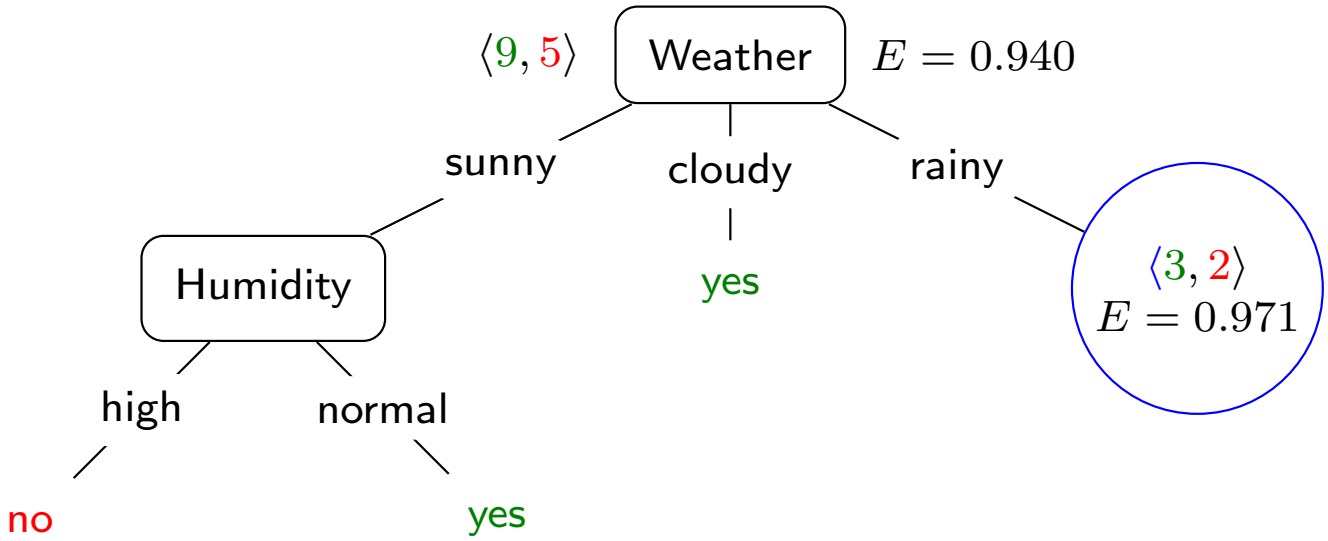
Example: A day for beach volleyball

- Decision tree after adding the second node:



- Nodes with data subsets $\langle m, 0 \rangle$ and $\langle 0, n \rangle$ have $E = 0$ and perfectly discriminate between examples, hence we can replace them with a leaf labeled yes and no, respectively

Example: A day for beach volleyball

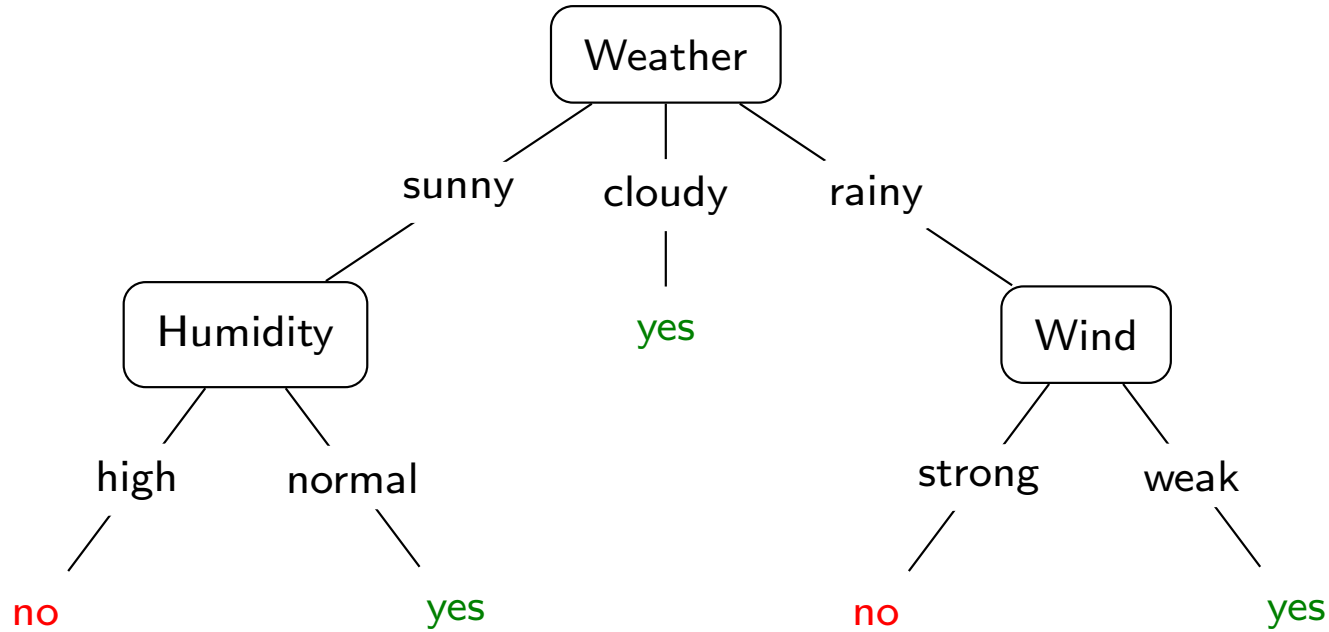


$D_{\text{Weather=rainy}}$

Day	x_2 Temp	x_3 Humidity	x_4 Wind	y Volleyball?
4	medium	high	weak	yes
5	low	normal	weak	yes
6	low	normal	strong	no
10	medium	normal	weak	yes
14	medium	high	strong	no

Example: A day for beach volleyball

- Computing information gain for the remaining three features on the subset $D_{\text{Weather=rainy}}$ gives **Wind** as the feature with the highest information gain
- The final decision tree looks like this:



ID3 algorithm

function $\mathrm{id3}(D, D^{parent}, X, y)$ -- *initially* $D^{parent} = D$
  if $D = \varnothing$ **then**
    $v \leftarrow \mathrm{argmax}_v \, |D^{parent}_{y=v}|$ -- *most frequent label of parent node*
    return $\mathrm{Leaf}(v)$
  $v \leftarrow \mathrm{argmax}_v \, |D_{y=v}|$ -- *most frequent label of current node*
  if $X = \varnothing$ **or** $D = D_{y=v}$ **then**
    return $\mathrm{Leaf}(v)$
  $x \leftarrow \mathrm{argmax}_{x \in X} \, IG(D, x)$ -- *most discriminative feature*
  $subtrees \leftarrow \varnothing$
  for $v \in V(x)$:
    $t \leftarrow \mathrm{id3}(D_{x=v}, D, X \setminus \{x\}, y)$
    $subtrees \leftarrow \mathrm{append}(subtrees, (v, t))$
  return $\mathrm{Node}(x, subtrees)$

X – set of all features, y – class label, D – set of labeled examples

ID3 algorithm – example

- Initial call:

$$\mathrm{id3}(D, D, \{x_1, x_2, x_3, x_4\}, y)$$

- After first iteration:

$$\begin{aligned}\mathrm{Node}\Big(x_1, \big\{&(\mathsf{sunny}, \mathrm{id3}(D_{x_1=\mathsf{sunny}}, D, \{x_2, x_3, x_4\}, y)),\\ &(\mathsf{cloudy}, \mathrm{id3}(D_{x_1=\mathsf{cloudy}}, D, \{x_2, x_3, x_4\}, y)),\\ &(\mathsf{rainy}, \mathrm{id3}(D_{x_1=\mathsf{rainy}}, D, \{x_2, x_3, x_4\}, y))\big\}\Big)\end{aligned}$$

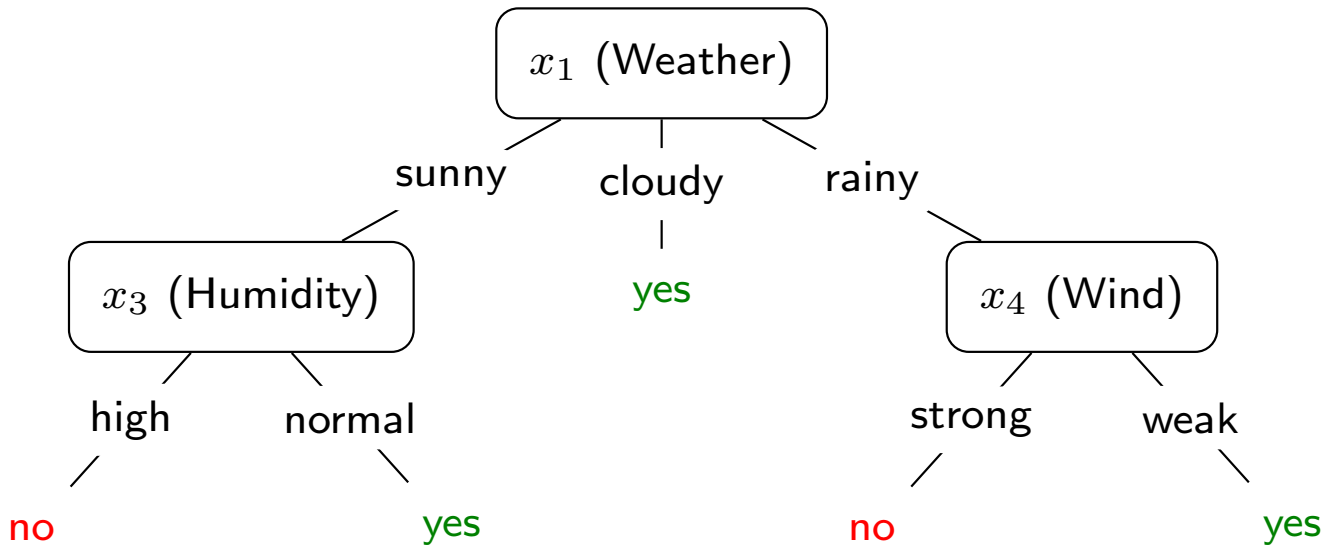
- After second iteration:

$$\begin{aligned}\mathrm{Node}\Big(x_1, \big\{&(\mathsf{sunny},\ \mathrm{Node}\big(x_3, \big\{(\mathsf{high}, \mathrm{id3}(D_{\substack{x_1=\mathsf{sunny}\\x_3=\mathsf{high}}}, D_{x_1=\mathsf{sunny}}, \{x_2, x_4\}, y)),\\ &\qquad (\mathsf{normal}, \mathrm{id3}(D_{\substack{x_1=\mathsf{sunny}\\x_3=\mathsf{normal}}}, D_{x_1=\mathsf{sunny}}, \{x_2, x_4\}, y))\big\}\big)),\\ &(\mathsf{cloudy}, \mathrm{Leaf}(\mathsf{yes})),\\ &(\mathsf{rainy},\ \mathrm{Node}\big(x_4, \big\{(\mathsf{strong}, \mathrm{id3}(D_{\substack{x_1=\mathsf{sunny}\\x_4=\mathsf{strong}}}, D_{x_1=\mathsf{sunny}}, \{x_2, x_3\}, y)),\\ &\qquad (\mathsf{weak}, \mathrm{id3}(D_{\substack{x_1=\mathsf{sunny}\\x_4=\mathsf{weak}}}, D_{x_1=\mathsf{sunny}}, \{x_2, x_3\}, y))\big\}\big))\big\}\Big)\end{aligned}$$

ID3 algorithm – example

- After third iteration:

$$\text{Node}\Big(x_1, \{(\text{sunny}, \text{Node}(x_3, \{(\text{high}, \text{Leaf}(\text{no})), (\text{normal}, \text{Leaf}(\text{yes}))\})), (\text{cloudy}, \text{Leaf}(\text{yes})), (\text{rainy}, \text{Node}(x_4, \{(\text{strong}, \text{Leaf}(\text{no})), (\text{weak}, \text{Leaf}(\text{yes}))\}))\}\Big)$$



Overfitting decision trees

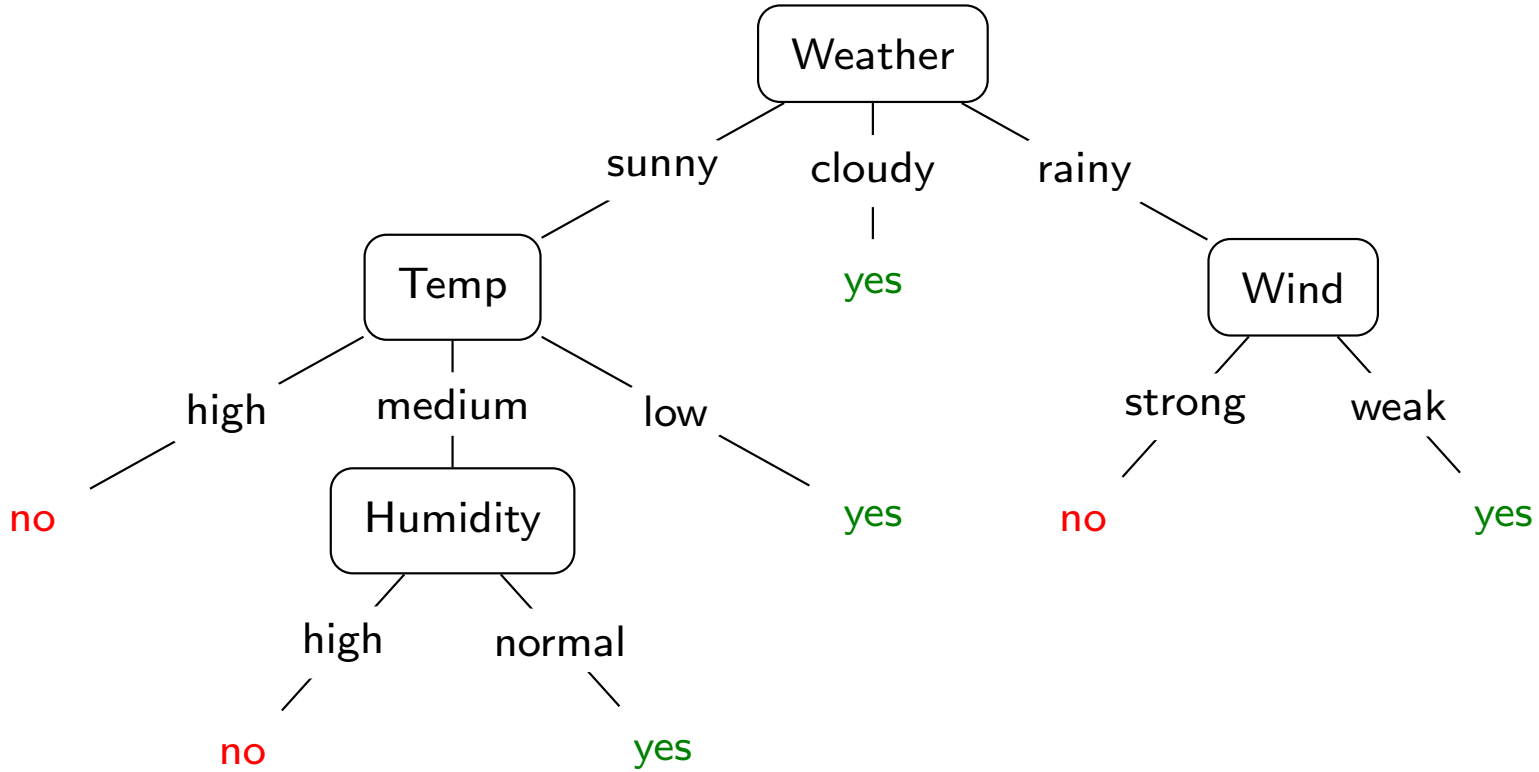
- Decision tree built by ID3 will always perfectly classify all examples from set D (assuming no identical examples have distinct labels)
- This easily leads to **overfitting**: the model classifies perfectly the examples from the training set, but does a lousy job on unseen examples $\Rightarrow$ **poor generalization**
- Overfitting occurs when training examples contain **noise** (incorrect feature values or class labels)
- E.g., assume set D is extended with a new, fifteenth examples:

$$\mathbf{x} = (\underbrace{\text{Weather}=\text{sunny}}_{x_1}, \underbrace{\text{Temp}=\text{high}}_{x_2}, \underbrace{\text{Humidity}=\text{normal}}_{x_3}, \underbrace{\text{Wind}=\text{strong}}_{x_4})$$

- Assume the correct label for this example is $y = \text{yes}$, but due to noise the label is flipped to $y = \text{no}$
- Note that the tree we built would correctly classify this examples

Overfitting decision trees

- However, if this incorrectly labeled example is added to the training set, the ID3 produces a different tree:

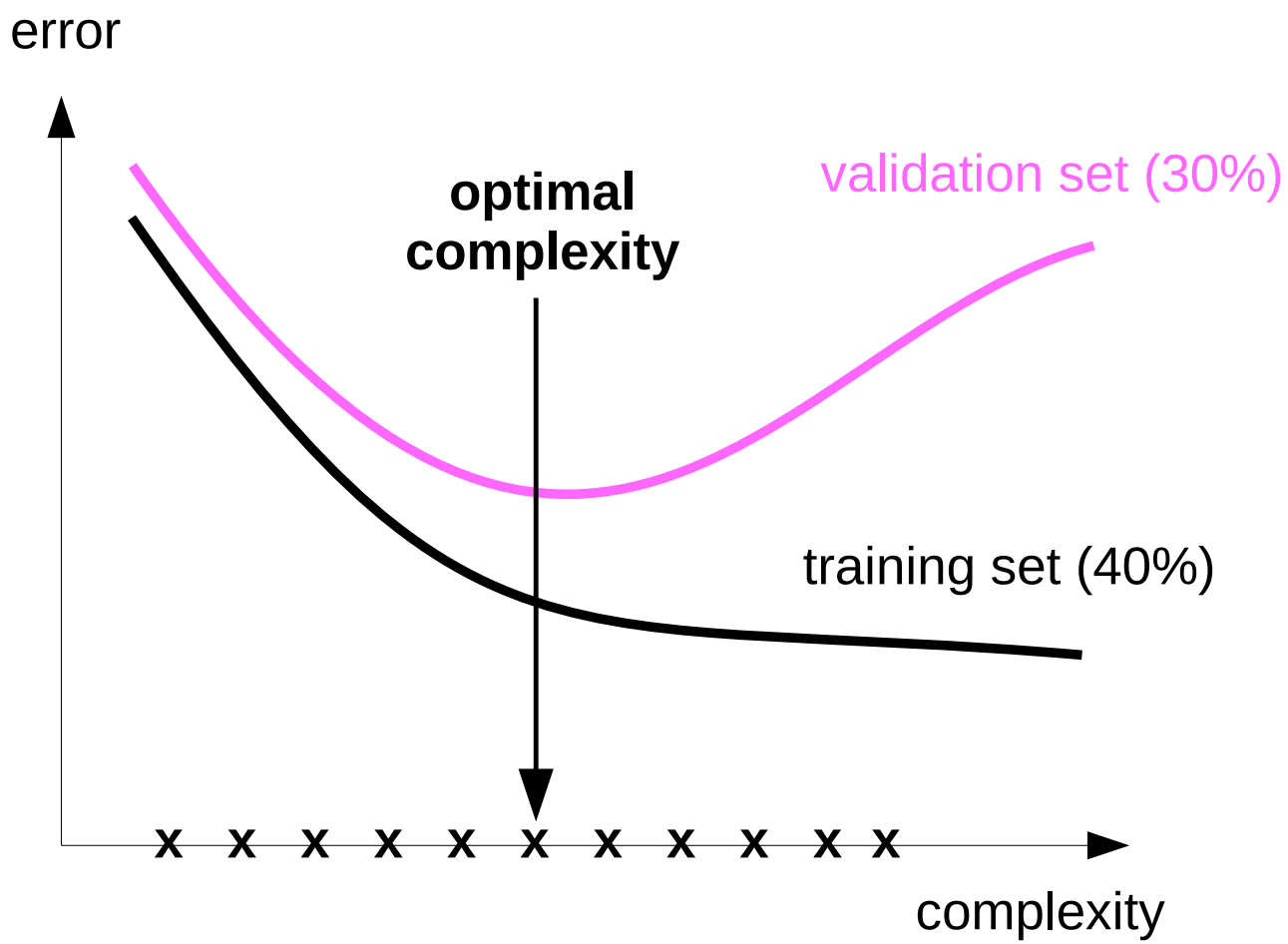


- This tree correctly classifies all fifteen examples as labeled in the training set, including the wrongly labeled fifteenth example

Overfitting decision trees

- Two approaches to preventing overfitting:
  1. **Stopping tree growth** before achieving perfect classification of all training examples
  2. **Subsequent pruning of an overfitted tree**
     - ★ replacing subtrees with leaves
     - ★ converting the tree into if-then rules and eliminating conditions from rule antecedents
- **Q:** How to determine the optimal size of a tree?
- **Intrinsic criteria** for stopping tree growth:
  - reaching a predefined maximum depth
  - number of examples covered by a node is smaller than a predefined number
  - reduction in entropy is smaller than a predefined threshold
- **Extrinsic criterion** for stopping tree growth or pruning:
  - accuracy drop (error increase) on a validation dataset

Cross-validation



E.g.., complexity = tree depth

Decision trees – remarks

- ID3 algorithm is the simplest tree building algorithm. There are more complex alternative (e.g., C4.5, QUEST)
- Various other feature selection measures can be used besides information gain (e.g., Gini index, χ^2-test)
- Decision trees can be used for numeric features: each node splits the data according to a threshold on the feature value
- Decision trees can also be used for regression (not only for classification): **CART** (classification and regression trees)
- Unlike Bayes classifier, decision trees...
  - don't produce classification probabilities
  - are easy to **interpret** (*black box* vs. *glass box* machine learning)
- Weakness: **prone to overfitting** (poor generalization), **high variance** (completely different trees for small changes in the training set)
- Can be alleviated using tree ensembles, e.g., the **random forest** model

Wrap-up

- Machine learning is about building **models** based on **data** that can **predict** properties of new, yet unseen data
- Main paradigms: **supervised**, **unsupervised**, **reinforcement** learning
- Supervised models are **trained** examples with class labels (**classification**) or numeric labels (**regression**)
- Machine learning models are prone to **overfitting**, which can be prevented by **cross-validation** on a validation set
- A **naïve Bayes classifier** classifies an example to a maximum posterior probability class (MAP-hypothesis), based on class likelihoods and prior probabilities estimated from training set
- A **decision tree** recursively partitions a set of examples by feature values, which in **ID3 algorithm** are selected by the **information gain** criterion

Next topic: Artificial neural networks