Introduction to machine learning & deep learning

- What ML is / isn't
- How ML works
- Where ML can be useful

Iftach Sadeh

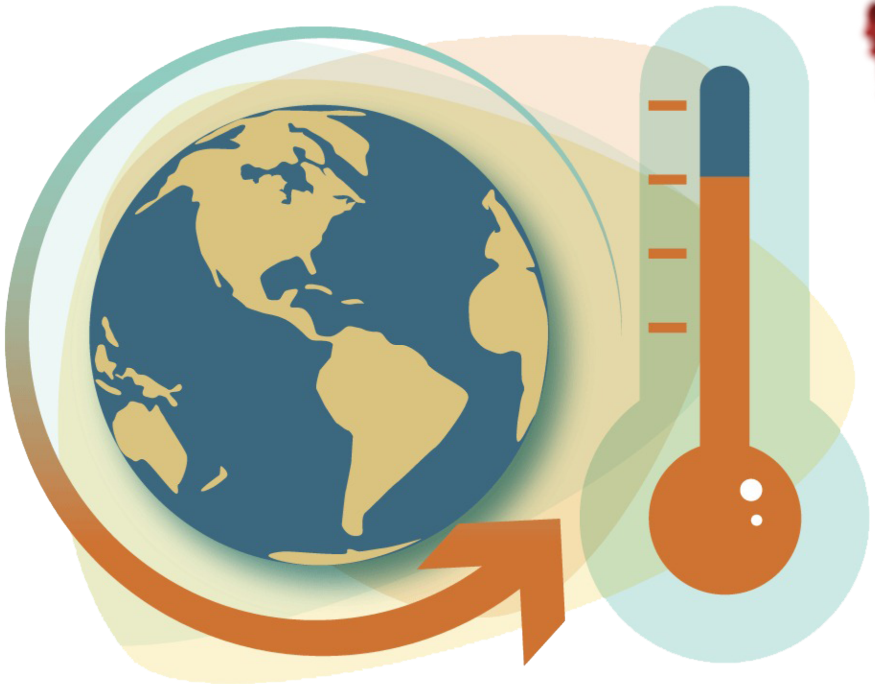
June 2020

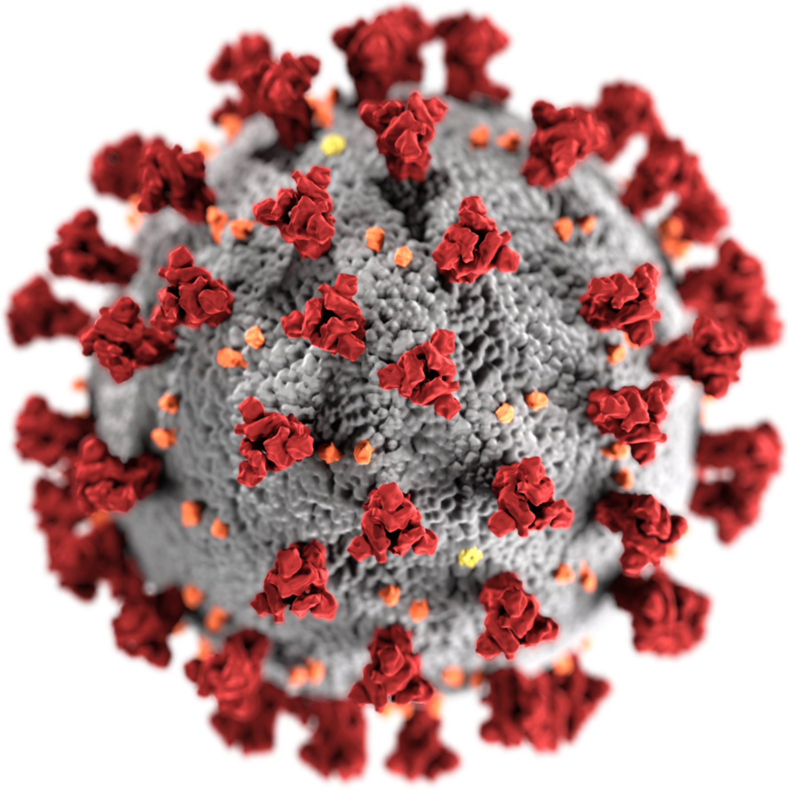
iftach.sadeh@desy.de

- Largely derived from:
  - University of Toronto CSC411 - Introduction to Machine Learning (Fall 2016). See: http://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/CSC411_Fall16.html
  - MIT's introductory course on deep learning - MIT 6.S191 - http://introtodeeplearning.com/

DESY

Modern challenges





What movie should I watch next?

- Undercover FBI agent
- Gang of surfers / bank robbers
- “Complex relationship” with Patrick Swayze

What movie should I watch next?

- Undercover FBI agent
- Gang of surfers / bank robbers
- “Complex relationship” with Patrick Swayze

- Undercover agent
- Keanu crossover

What movie should I watch next?

- Undercover FBI agent
- Gang of surfers / bank robbers
- “Complex relationship” with Patrick Swayze

- Action at sea
- Big storm

What movie should I watch next?

- Undercover FBI agent
- Gang of surfers / bank robbers
- “Complex relationship” with Patrick Swayze

- Emotional stakes
- Drowning

What movie should I watch next?

- Superficial correlations:
  - Anything with Keanu Reeves
  - Plot specifics - undercover cop; heist; criminal gangs; sports; action at sea; social clubs …
- Critical- / viewer-analysis (code words):
  - Directing ; cinematography ; sound track …
- Cultural / contextual significance:
  - 90s movies I only watch “ironically”
- User surveillance:
  - What time of day this is?
  - What did I have for dinner?
  - What did I Google two weeks ago?
- …

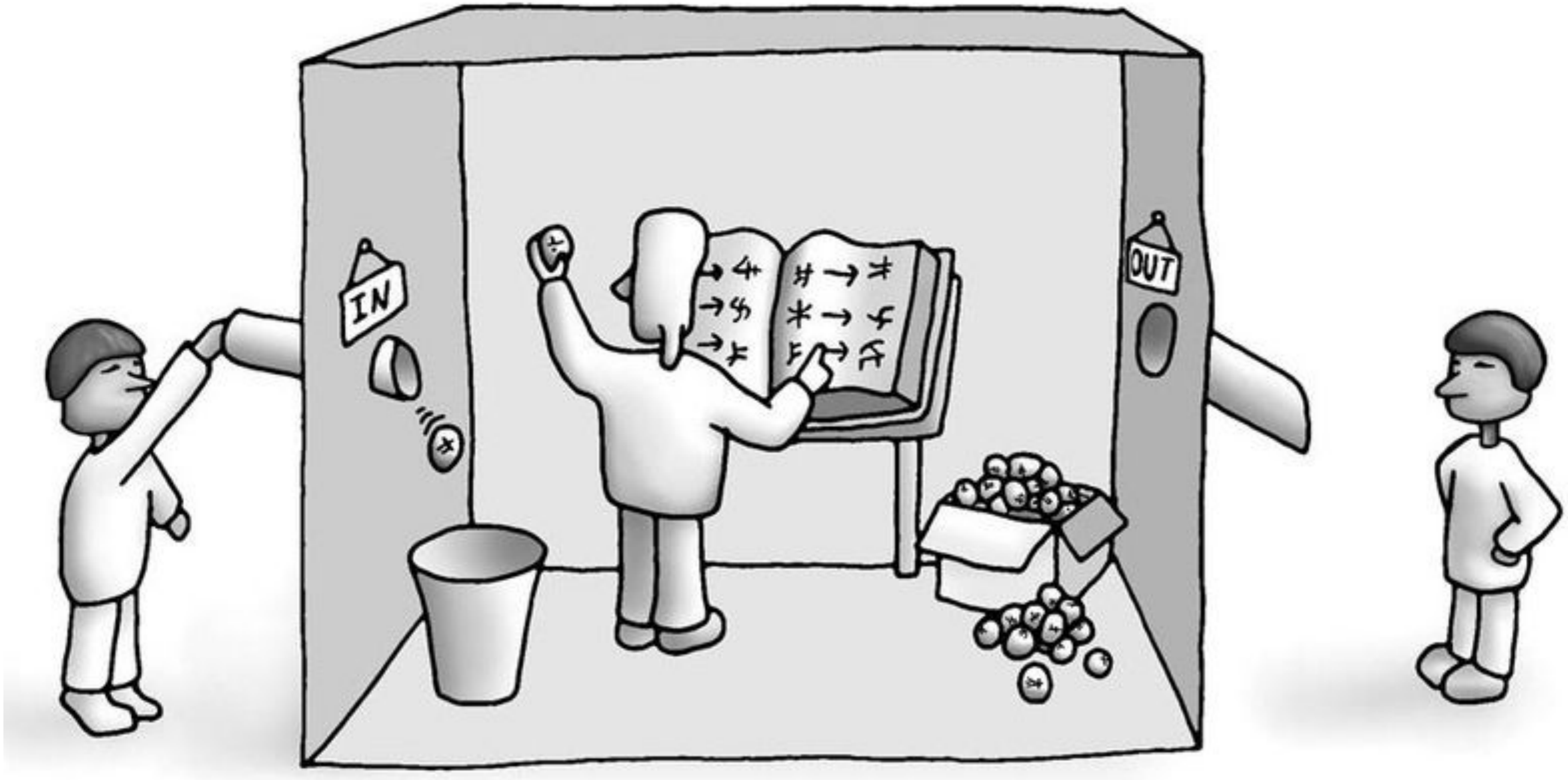
Enters machine learning…

- How can we solve a specific problem?
  - Program that encodes a set of rules that are useful to solve the problem.
  - Usually difficult to specify those rules, e.g., locate the cat in the image?

- On the other hand - learning systems are not directly programmed to solve a problem. Instead, develop own program based on:
  - Examples of how they should behave.
  - From trial-and-error experience trying to solve the problem.
- Different than standard computer science:
  - Want to implement unknown function, only have (training) sample input/output.
  - Learning ➜ incorporating information from the training examples into the system.

The Chinese room thought experiment

- John Searle, "*Minds, Brains, and Programs*" (1980).

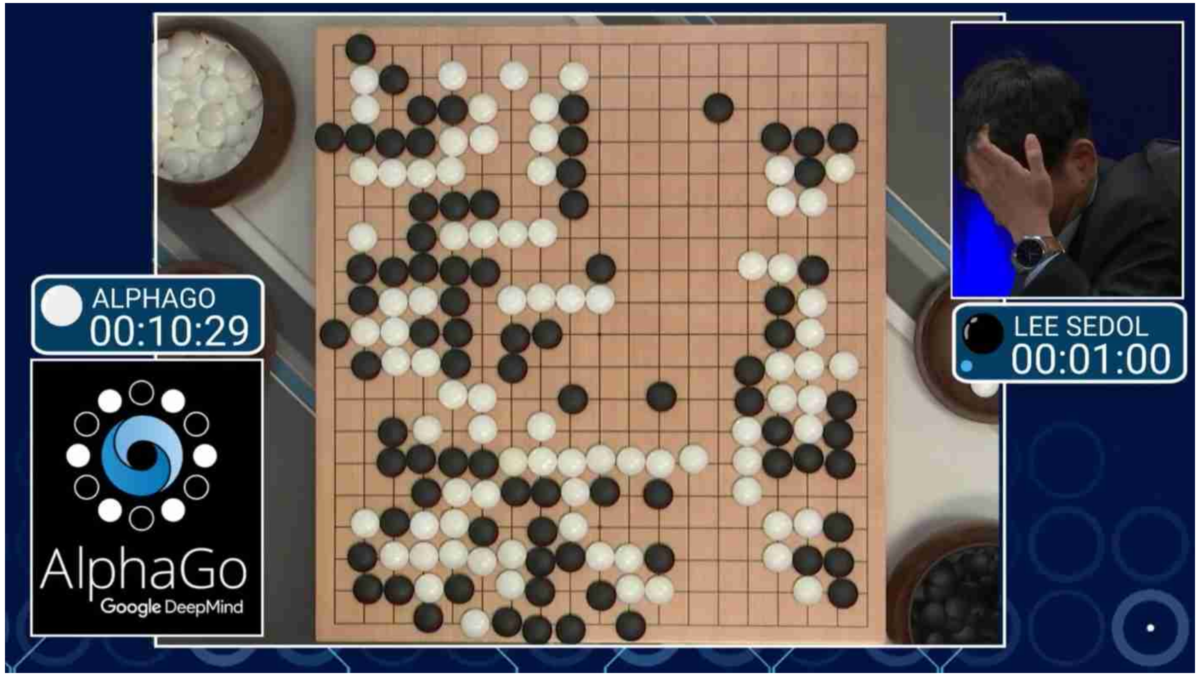


Our robot overlords

Artificial General Intelligence

- “Strong AI” >= human intelligence
- Ability to reason, solve puzzles, make judgments, plan, learn, and communicate
- Desire to bring about judgement day ?

Narrow AI



- “Weak AI” limited to a specific or narrow area

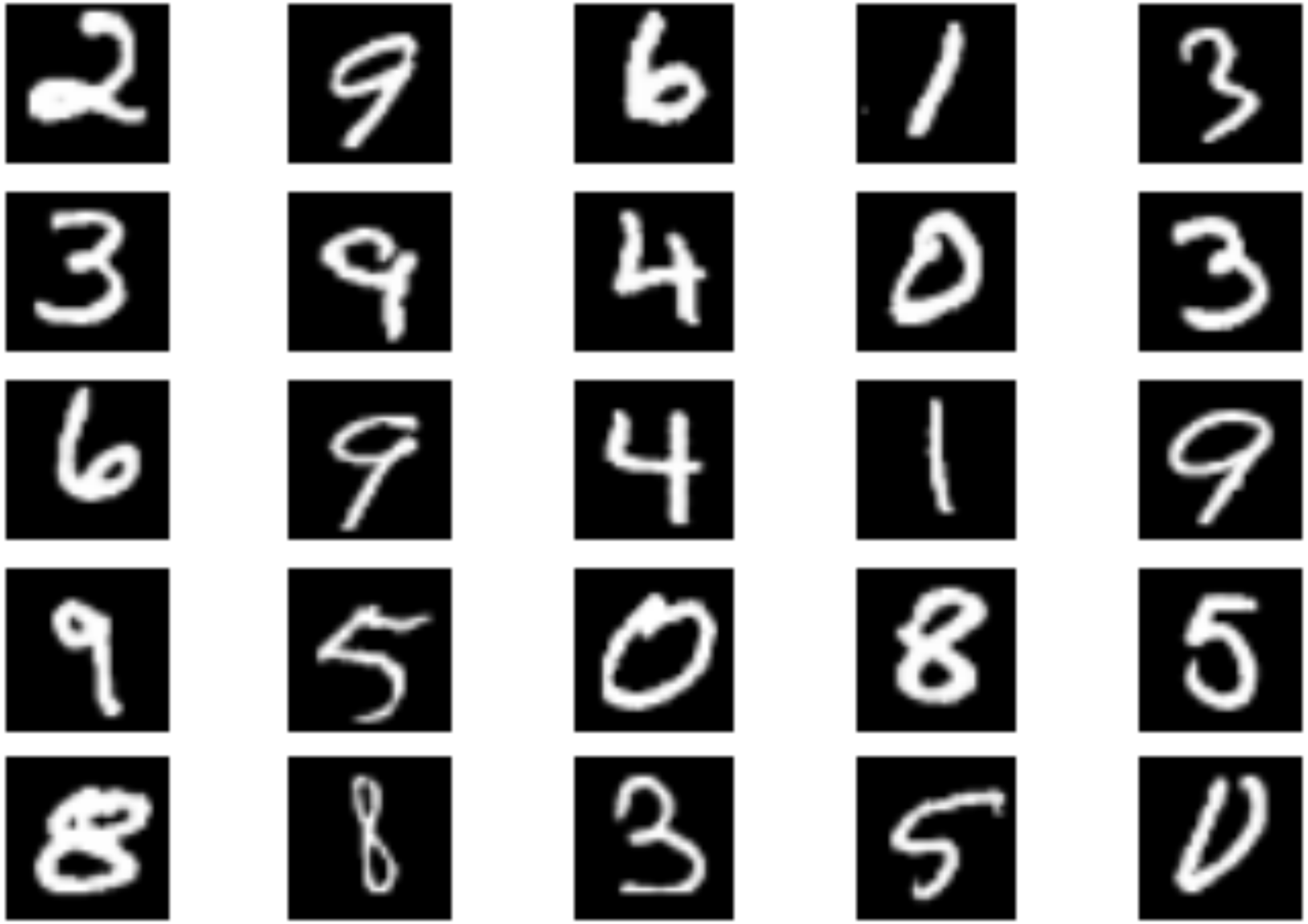
AlphaGo

Types of machine learning

Types of ML

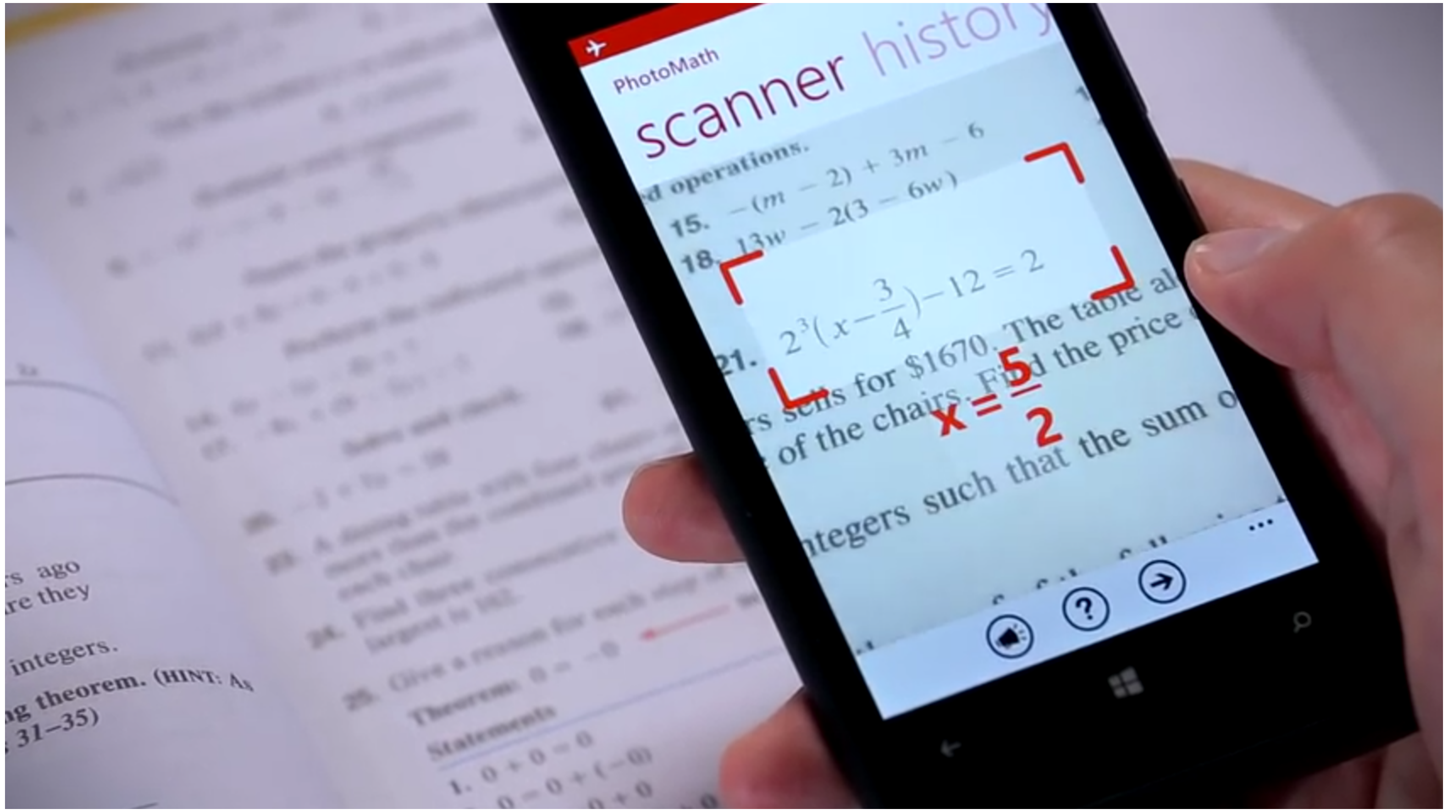
- **Classification**: determine which discrete category the example is
- **Recognising patterns**: speech recognition, facial identity, etc
- **Recognising anomalies**: unusual sequences of credit card transactions, panic situation at an airport
- **Recommender Systems**: noisy data, commercial pay-off (e.g., Amazon, Netflix).
- **Information retrieval**: find documents or images with similar content
- **Computer vision**: detection, segmentation, depth estimation, optical flow, etc
- **Robotics**: perception, planning, etc
- Learning to **play games**
- **Spam filtering, fraud detection**: the enemy adapts so we must adapt too
- Many more…

Classification

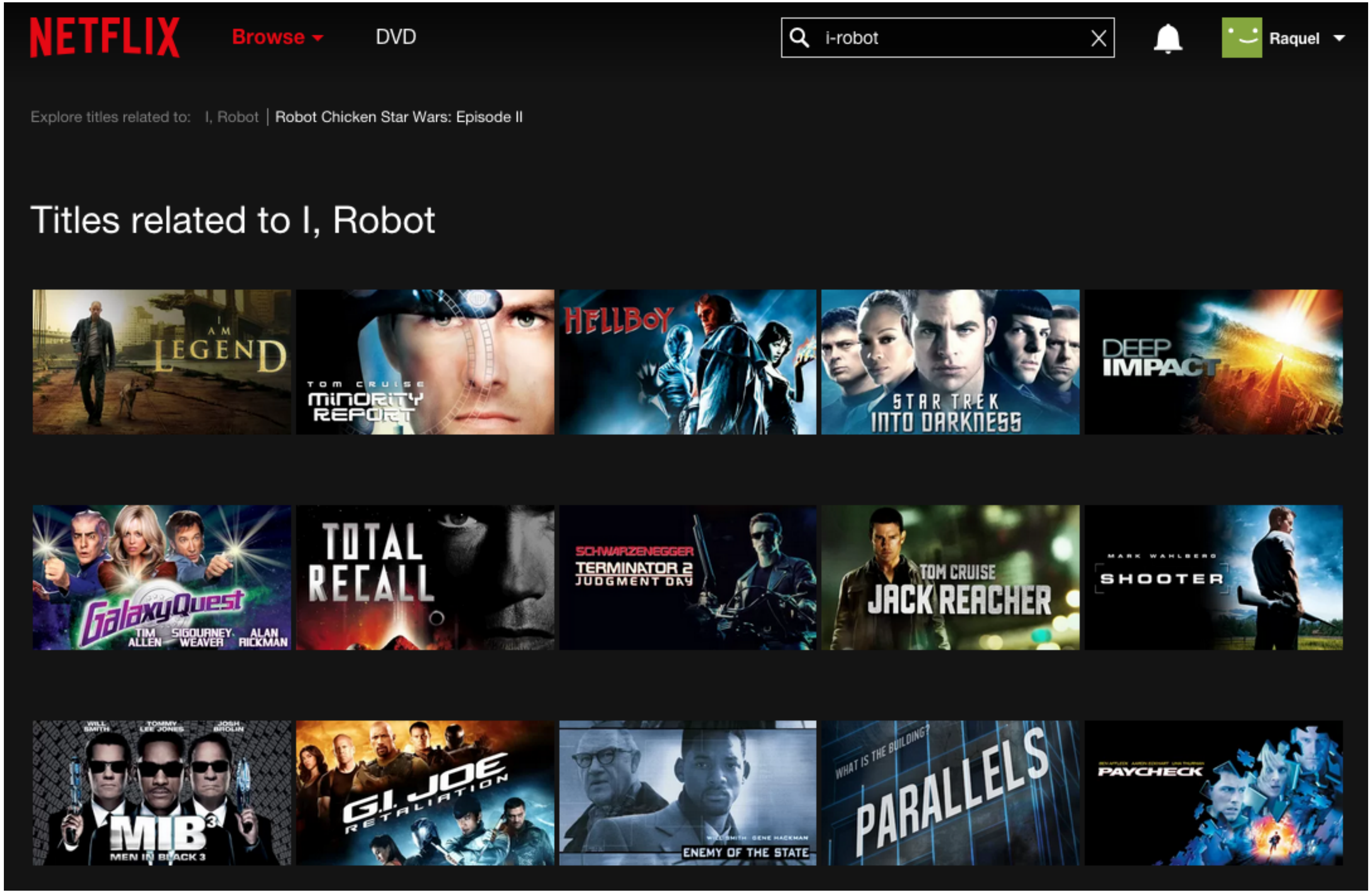


What digit is this?

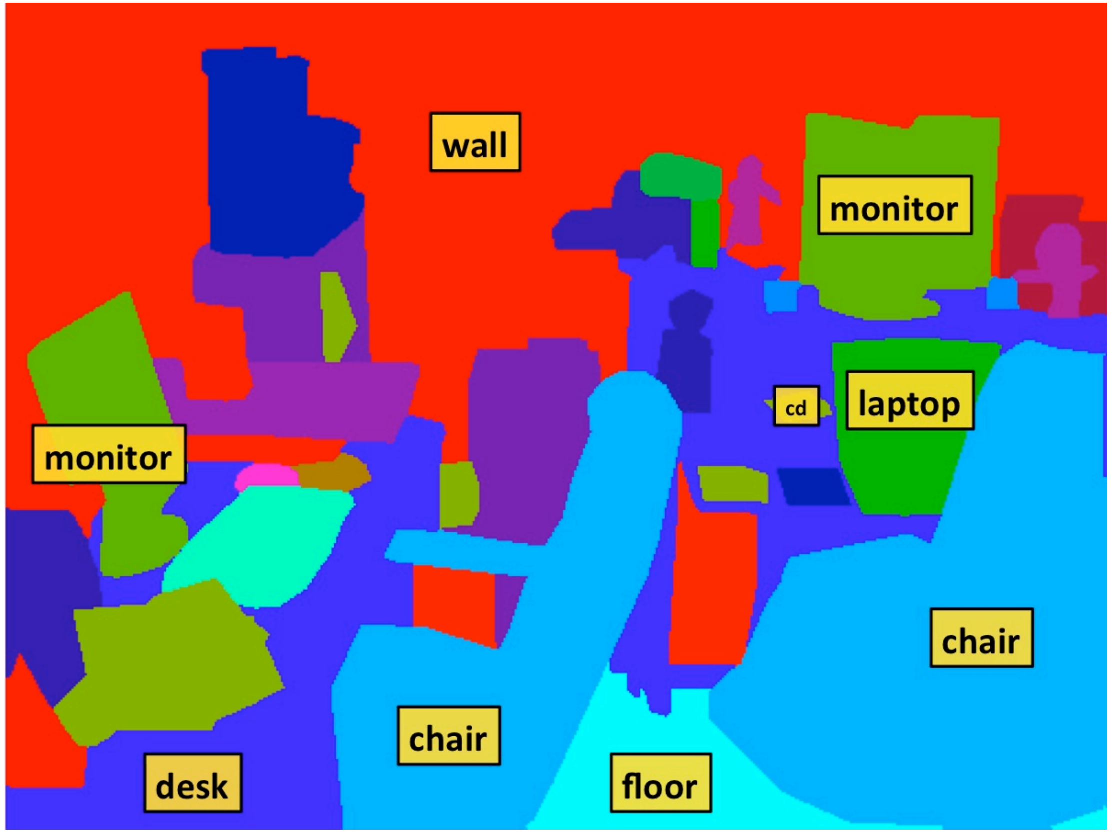
Pattern recognition



Recommendation system

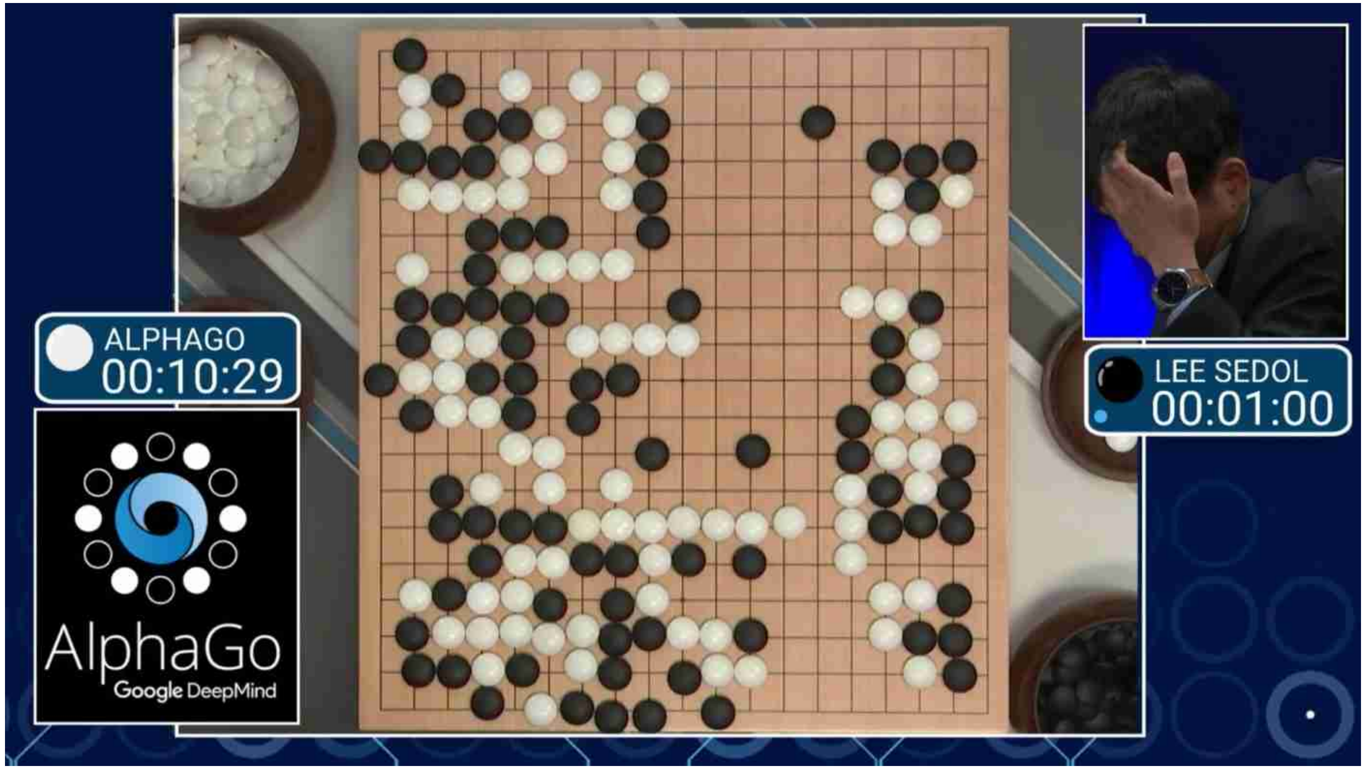


Computer vision



Playing games

- Learning to play Super Mario: https://www.youtube.com/watch?v=wfL4L_I4U9A
- Learning to play Alpha Go:



How does it actually work?

ML techniques

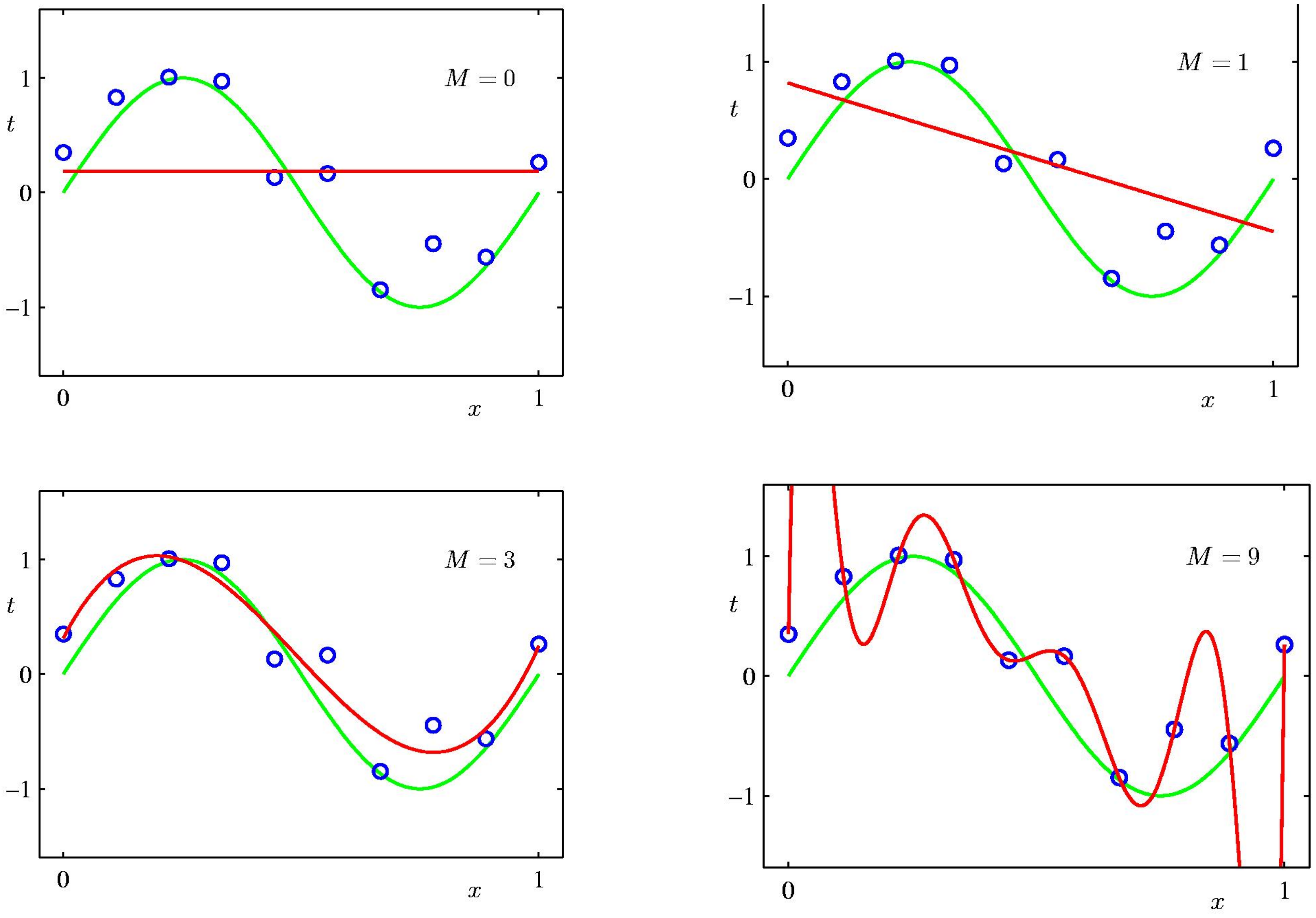
- **Supervised**: correct output known for each training example - Learn to predict output when given an input vector:
  - Classification: 1-of-N output (speech recognition, object recognition, medical diagnosis)
  - Regression: real-valued output (predicting market prices, customer rating)
- **Unsupervised learning**:
  - Create an internal representation of the input, capturing regularities/structure in data
  - Examples: form clusters; extract features ➜ How do we know if a representation is good?
- **Reinforcement learning:**
  - Learn action to maximize payoff
  - Not much information in a payoff signal
  - Payoff is often delayed

Components of supervised learning

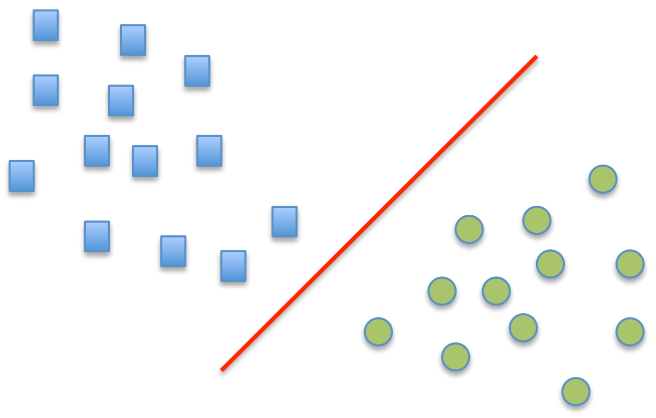
- **Continuous outputs** (regression), e.g., a rating, # of followers, house price, or **”integer” classes** (classification), e.g., animal breed.

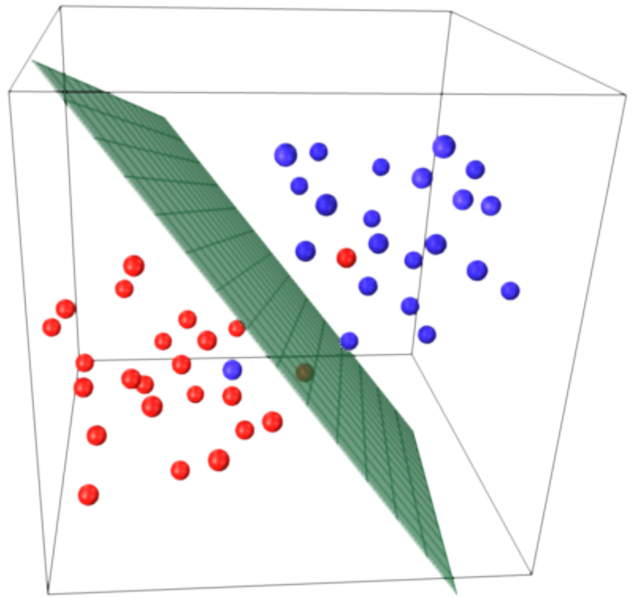
- What do I need in order to predict these outputs?
  - **Features** - inputs to the estimator & **labels** - known outputs.
  - **Training examples**, many input sets for which the output is known (e.g., many movies with ratings)
  - A **model**, a function that represents the relationship between inputs / outputs
  - A **loss / cost function**, which tells us how well our model approximates the training examples
  - **Optimization**, a way of finding the parameters of our model that minimizes the loss function

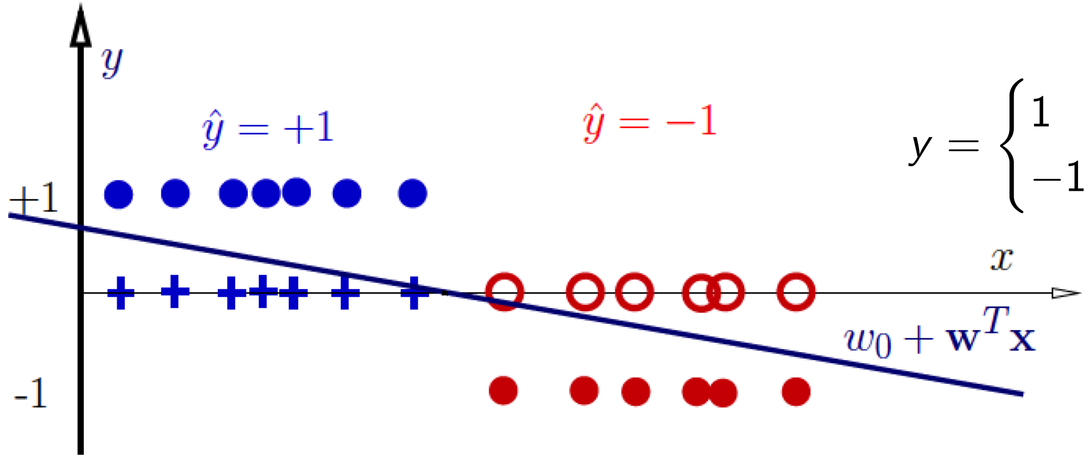
Regression



Classification



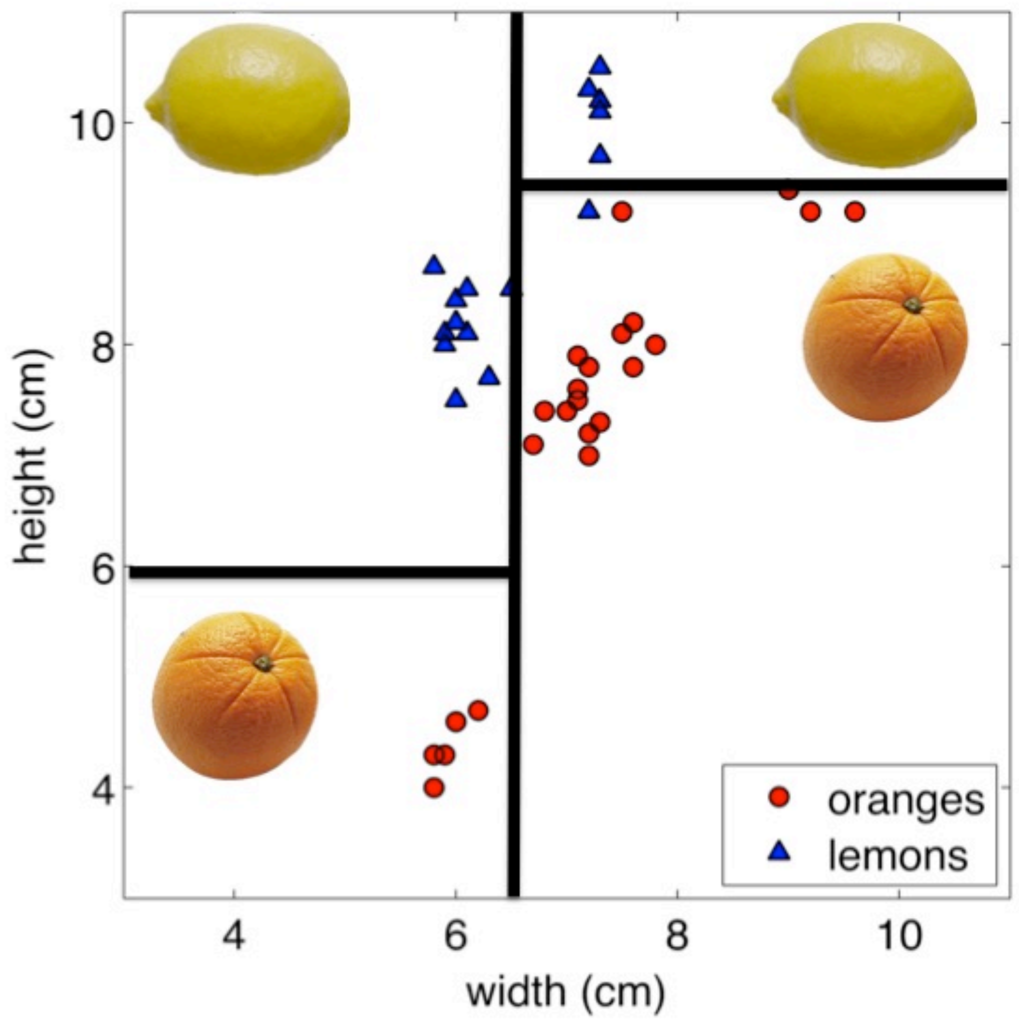


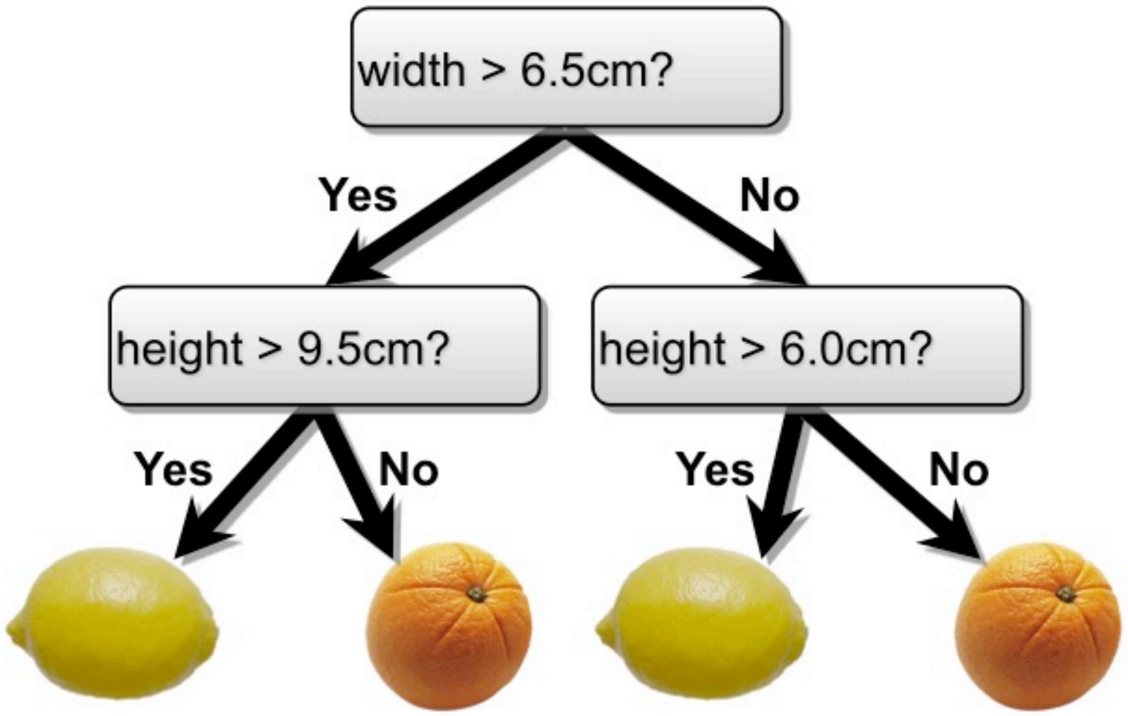


$$y = \begin{cases} 1 & \text{if } f(\mathbf{x}, \mathbf{w}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

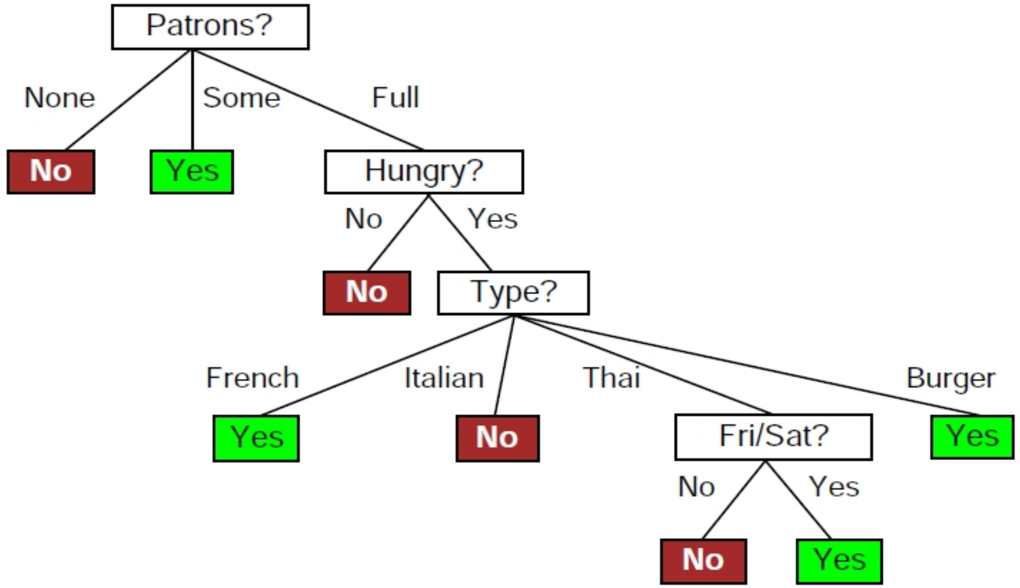
TP (True Positive) FP (False Positive) FN (False Negative)

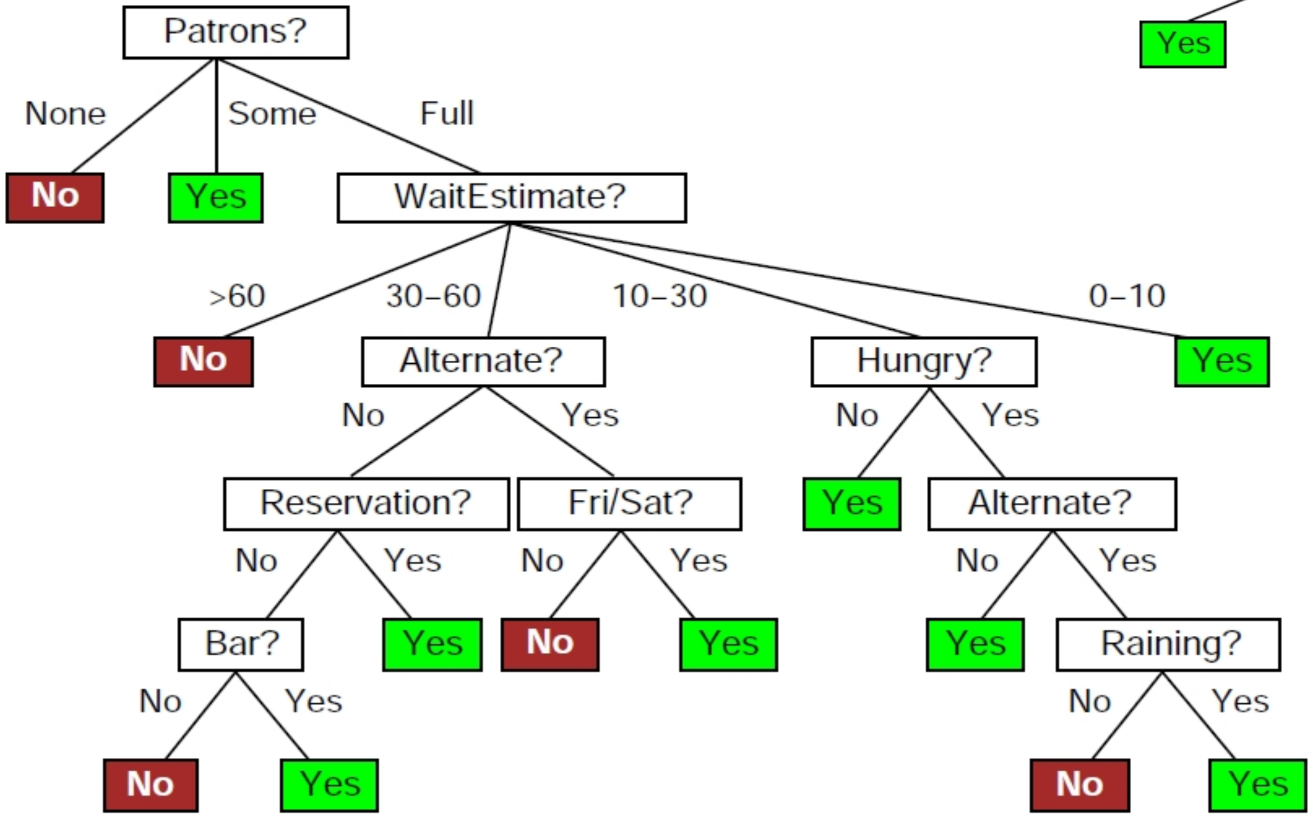
Decision Trees





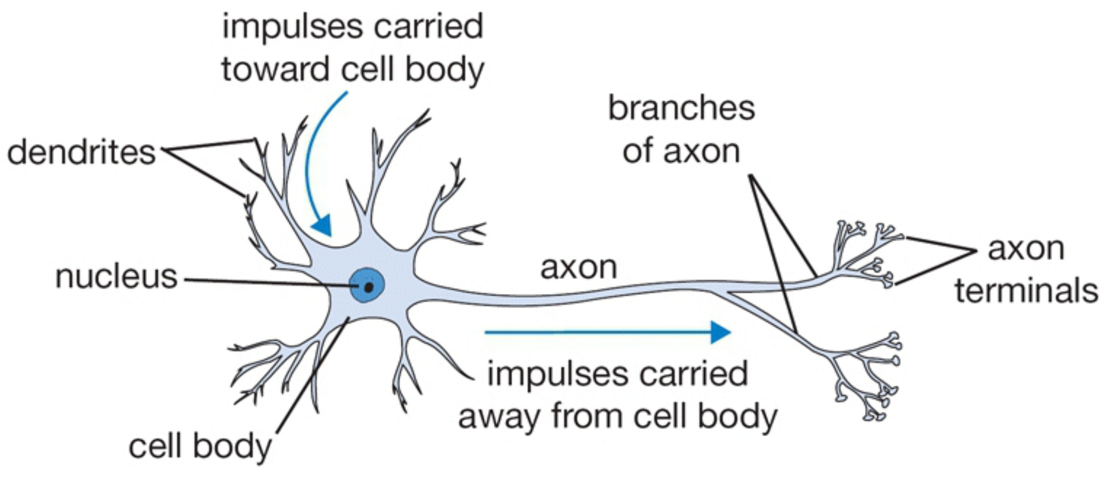
Which tree is better?

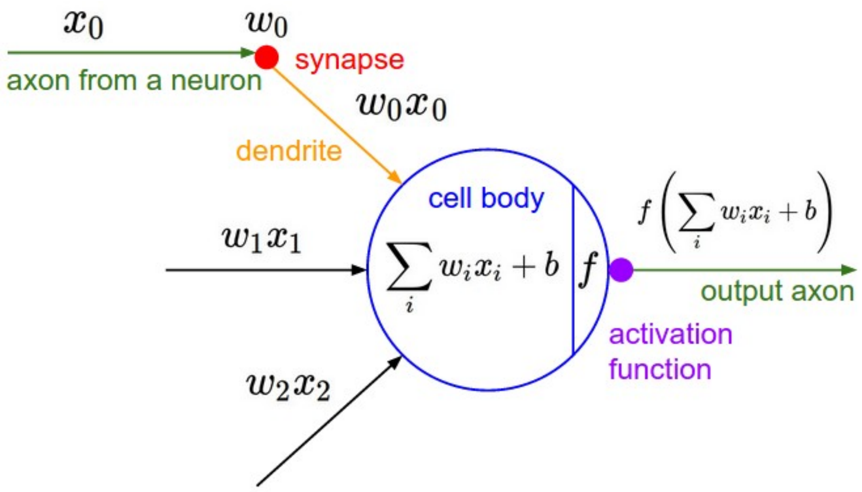


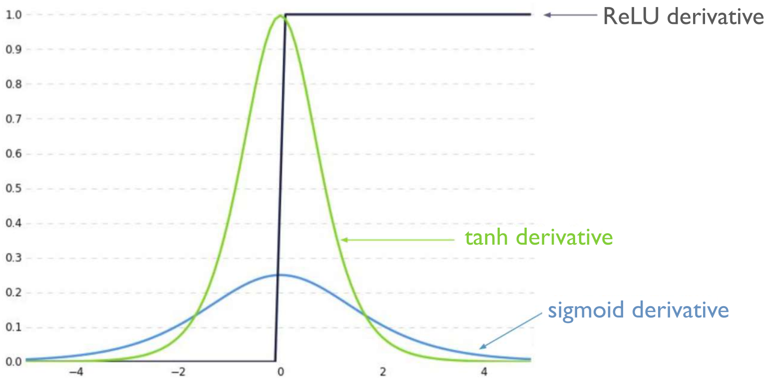


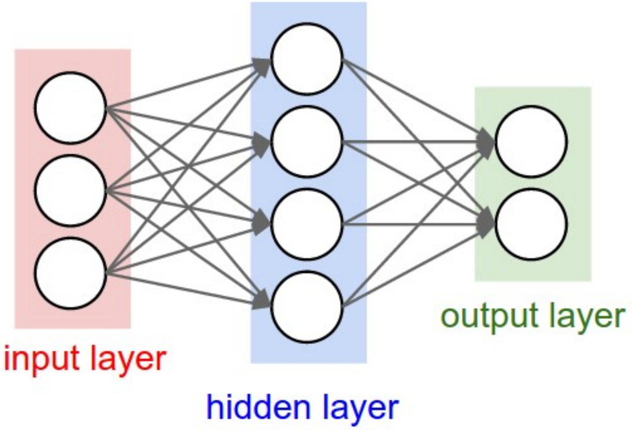
Artificial neural networks (ANN)

- Non-linear discriminative classifier that utilises functions of input variables
- Use a large number of simpler functions ➜ for fixed functions (Gaussian, sigmoid, polynomial basis functions), optimisation involves linear combinations of (fixed functions of) the inputs



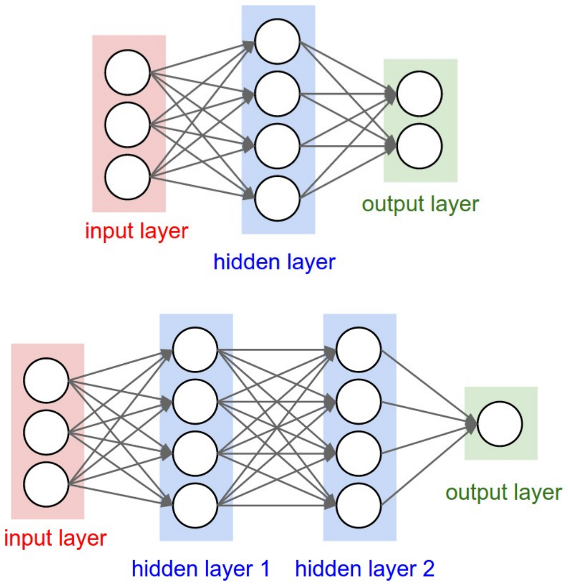


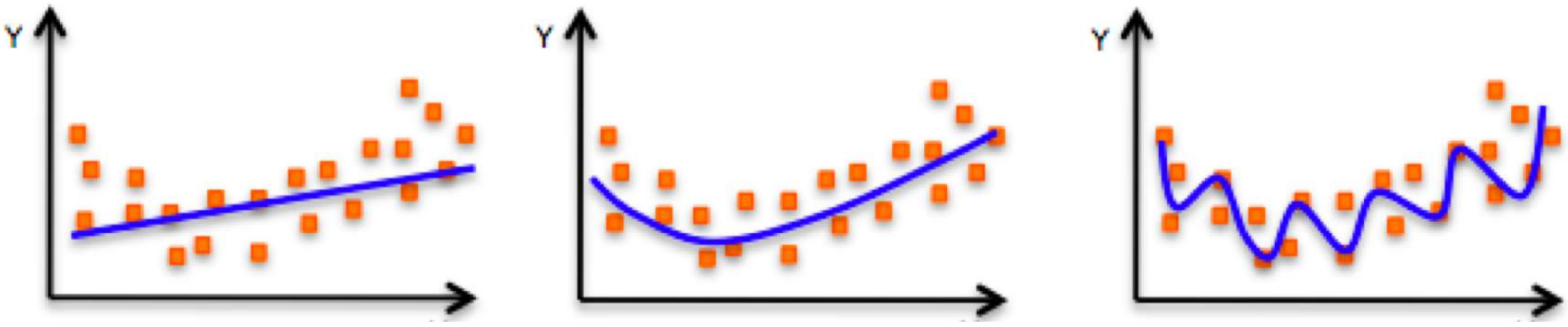




Overfitting

- Training data contain information about the true patterns in the mapping from input to output
- But they also contain statistical & systematic noise:
  - The target values may be unreliable
  - There are statistical fluctuations (there will be accidental patterns)
- Fit the model ➜ end up predicting both true and spurious properties





Underfitting
Model does not have capacity to fully learn the data

Ideal fit

Overfitting
Too complex, extra parameters, does not generalize well

Deep learning

Why deep learning?

https://www.youtube.com/watch?v=PoiHAiDHgDs

Deep learning

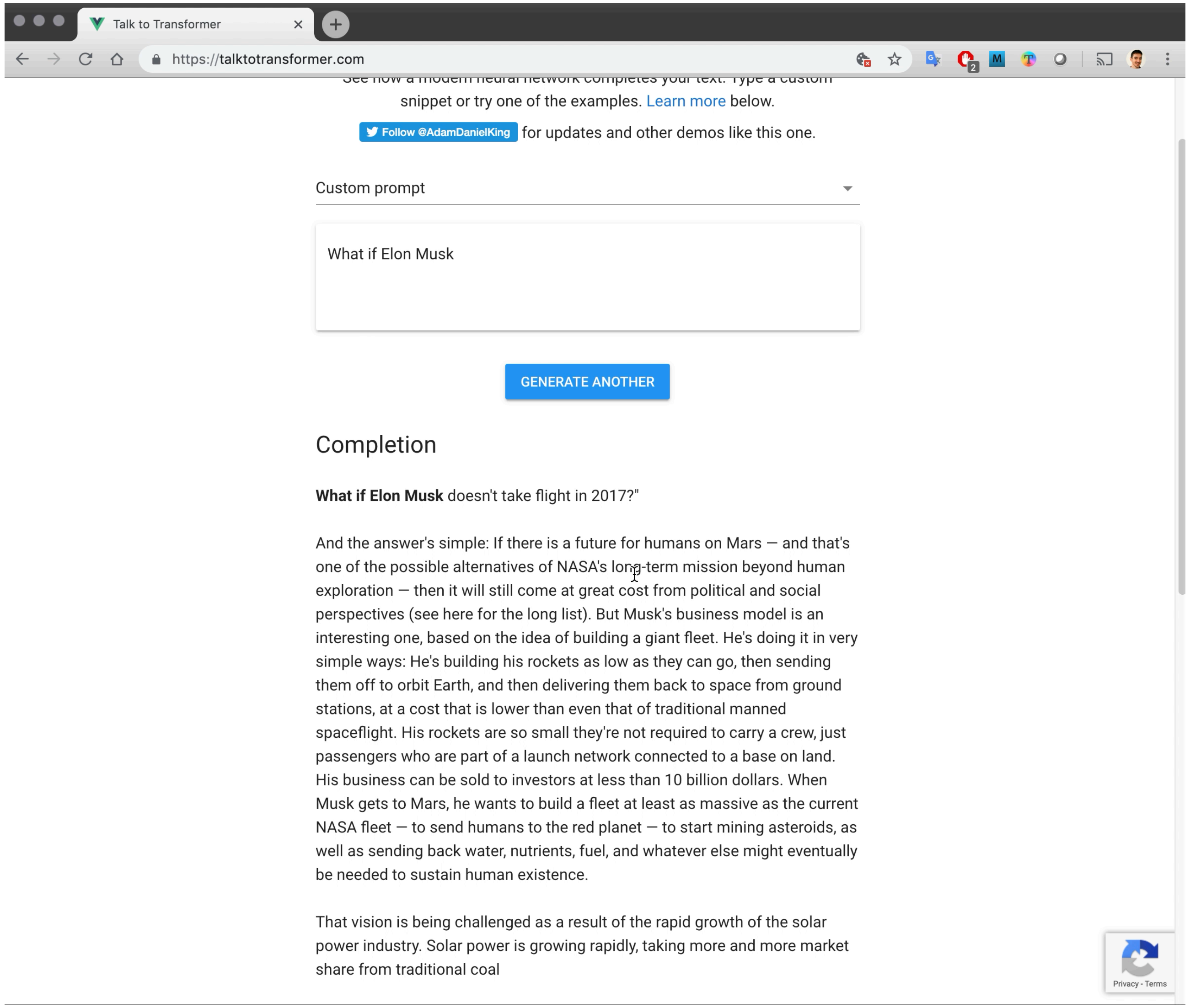
- **Challenges**:
  - Phase-space: huge number of classes, with lots of intra-class variation
  - Segmentation: real scenes are cluttered
  - Invariances: variations (or fluctuations) do not affect nominal shape
  - Deformations: natural shape classes allow variations (faces, letters, chairs)
  - A huge amount of computation

Deepfake

“Club Fight” from [Ctrl Shift Face](https://youtu.be/bFojG5jCoQA?t=20) :

https://youtu.be/bFojG5jCoQA?t=20

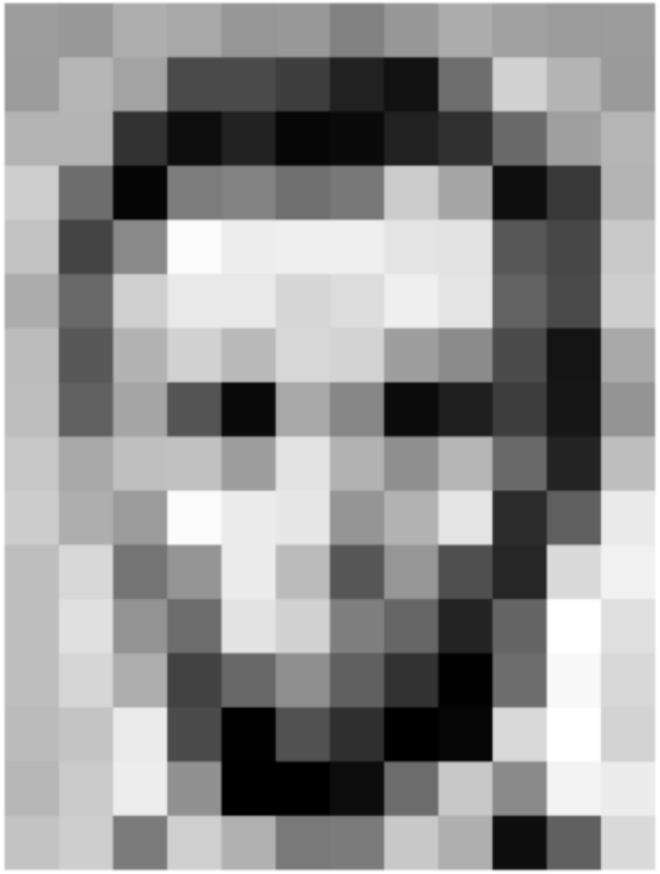
Text generation

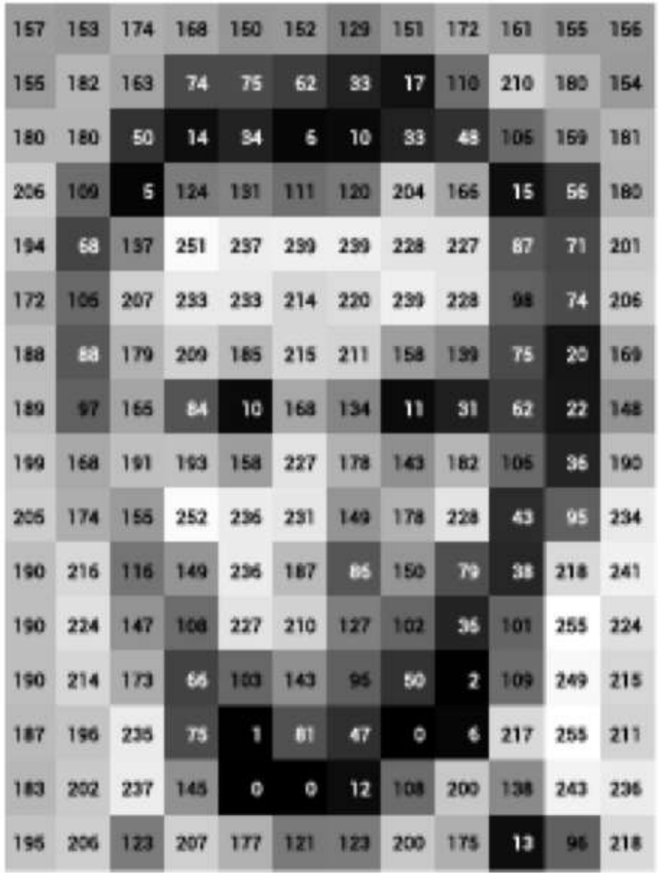


https://talktotransformer.com/

Computer vision

Computer vision



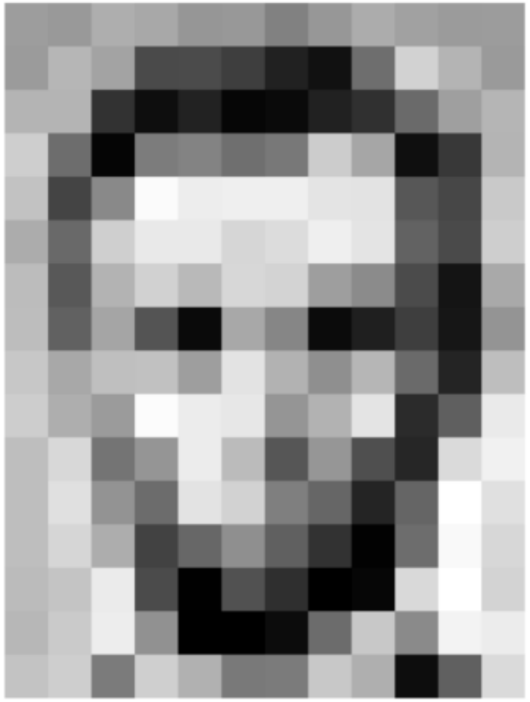


What the computer sees

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

Computer vision



Input Image

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Pixel Representation

classification

$$\begin{array}{l} \text{Lincoln} \\ \text{Washington} \\ \text{Jefferson} \\ \text{Obama} \end{array} \begin{bmatrix} 0.8 \\ 0.1 \\ 0.05 \\ 0.05 \end{bmatrix}$$

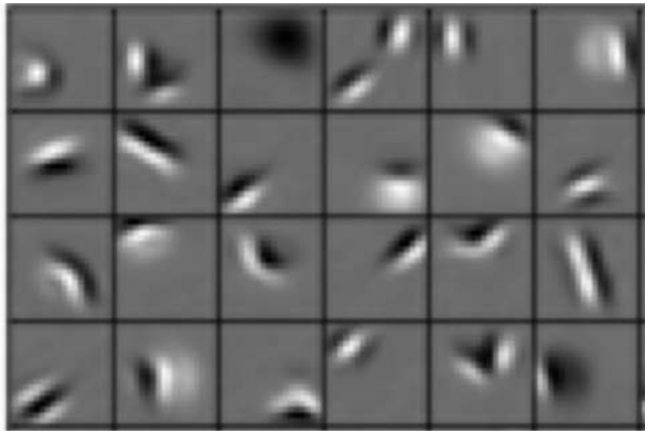
Hierarchy of features

- Try to design features for detection:

Domain knowledge → Define features → Detect features to classify

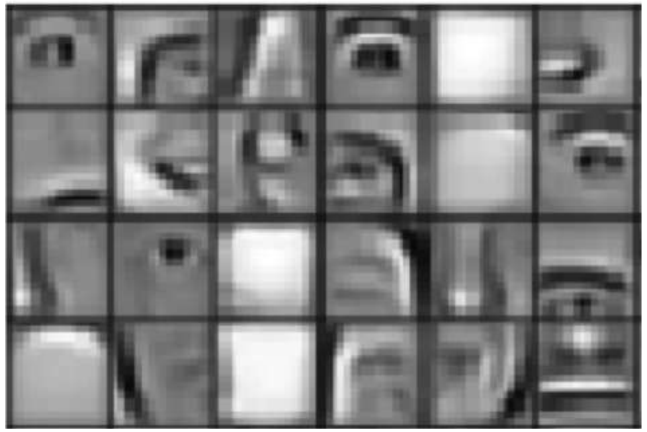
- Or … lean hierarchy of features directly from data:

Low level features



Edges, dark spots

Mid level features



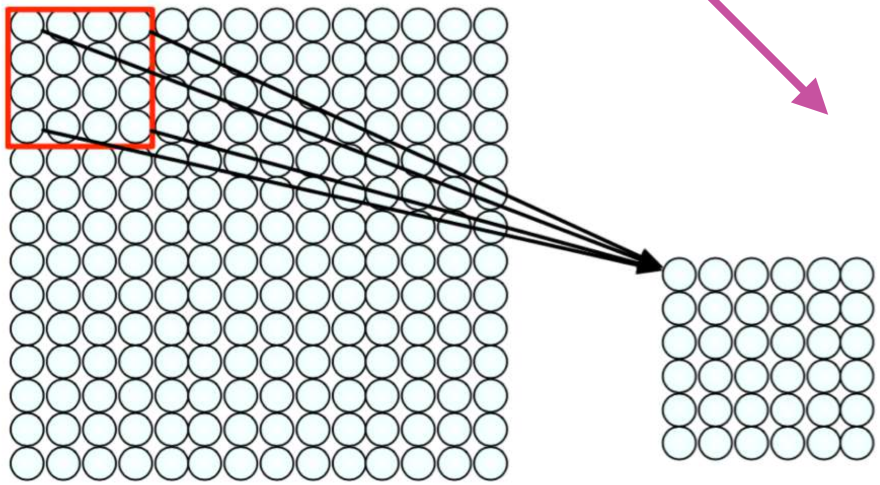
Eyes, ears, nose

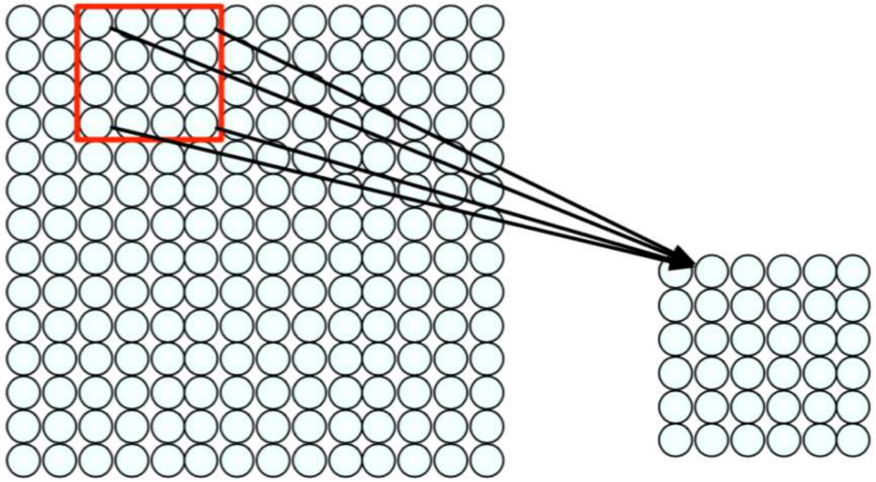
High level features

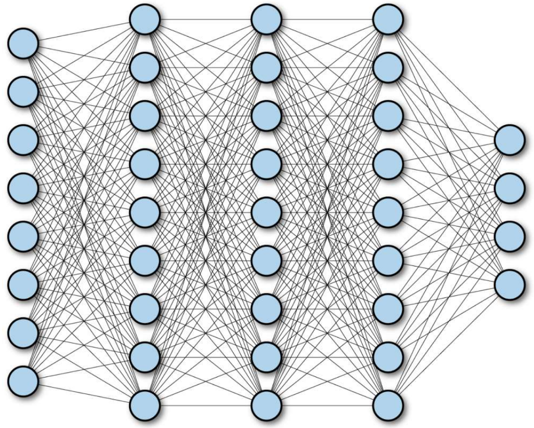
Facial structure

Use spatial structure

- Apply a set of weights (a filter) to extract local features
- Use multiple filters to extract different features
- Spatially share parameters of each filter
- Example:
  - Filter of size 4x4 : 16 different weights
  - Apply same filter to 4x4 patches (convolution) in input
  - Shift by 2 pixels for next patch

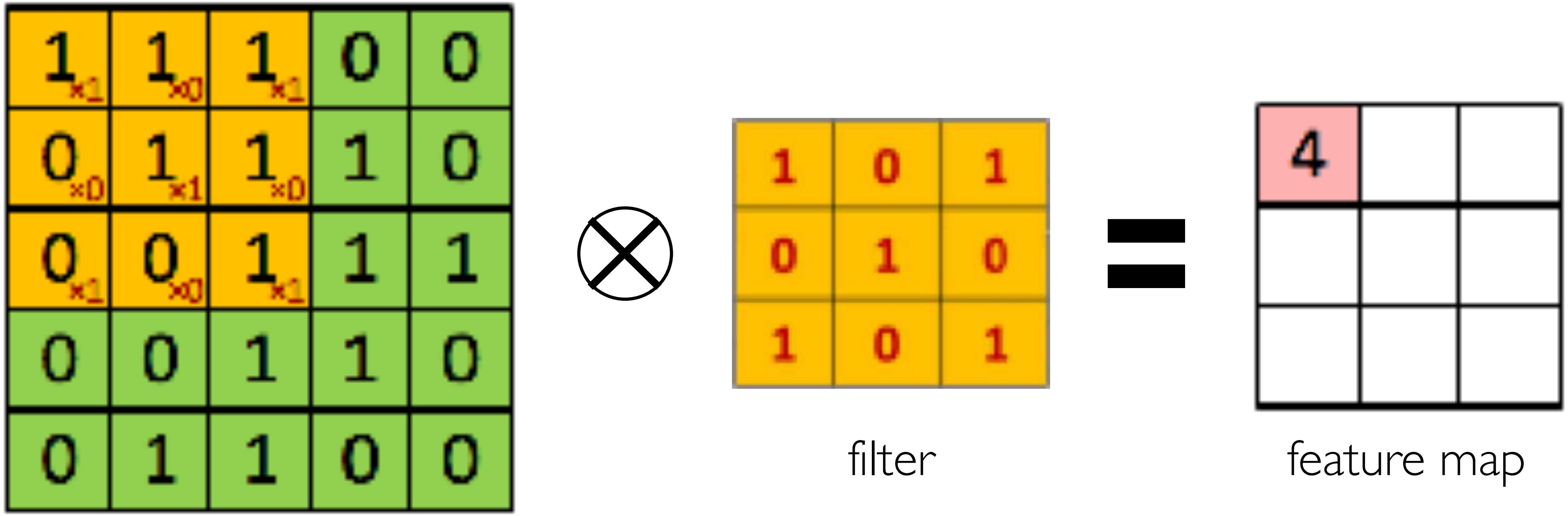




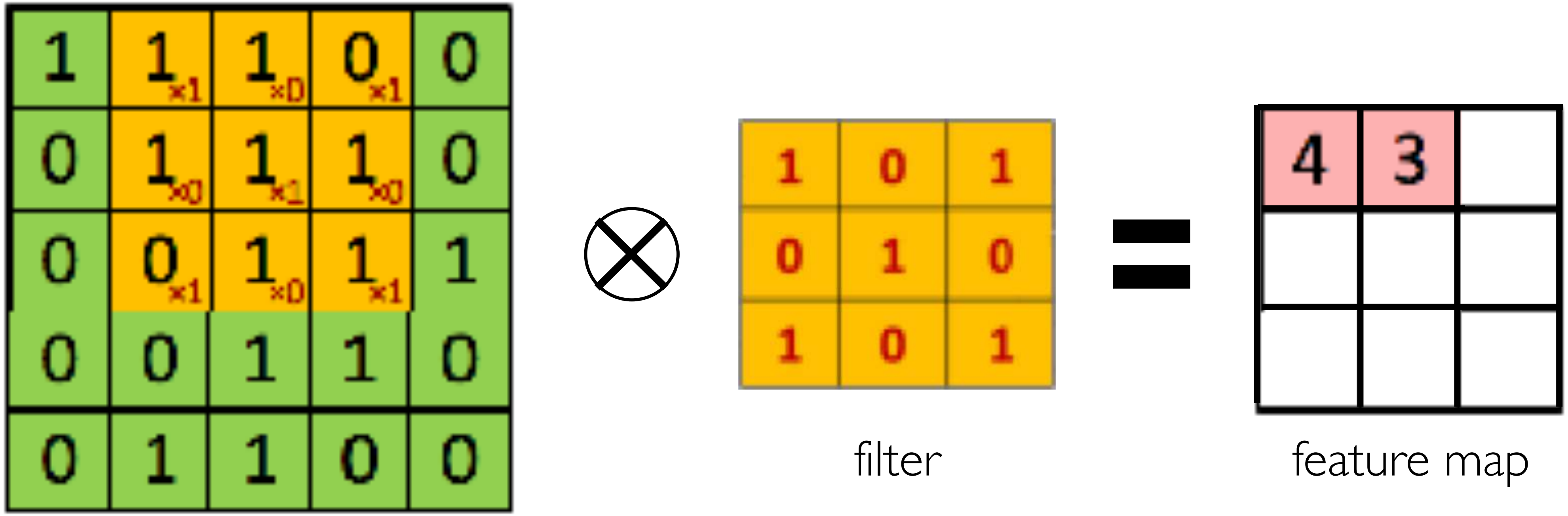


- No spatial info
- Many many parameters

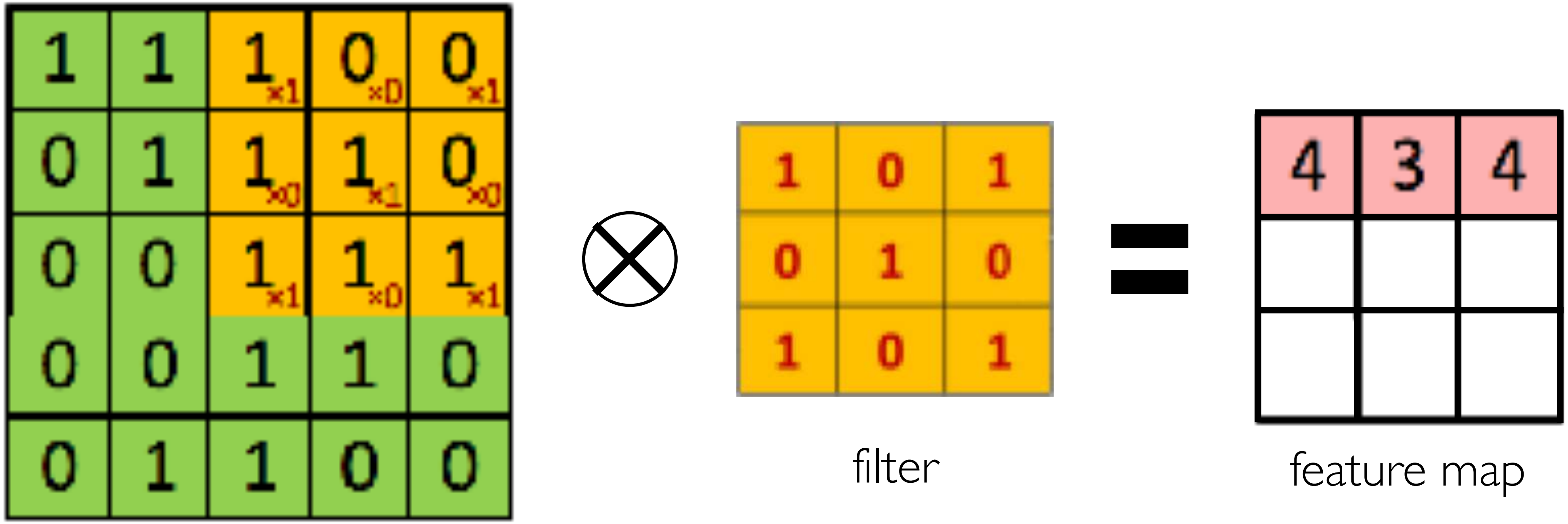
Convolution



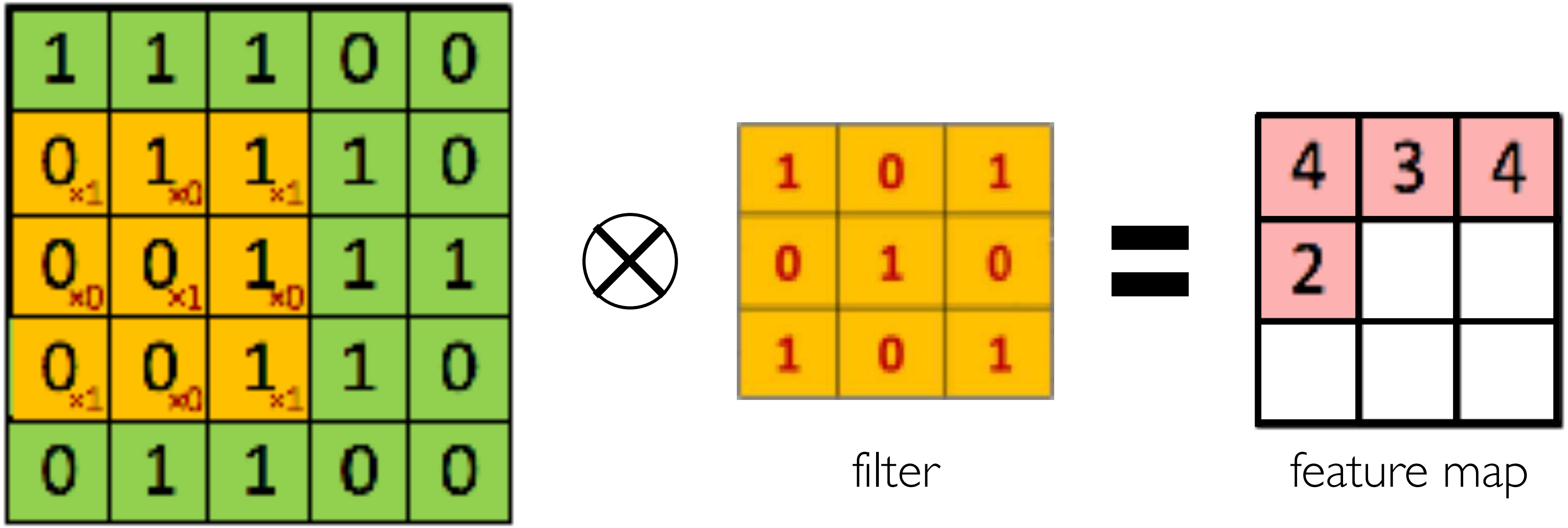
Convolution



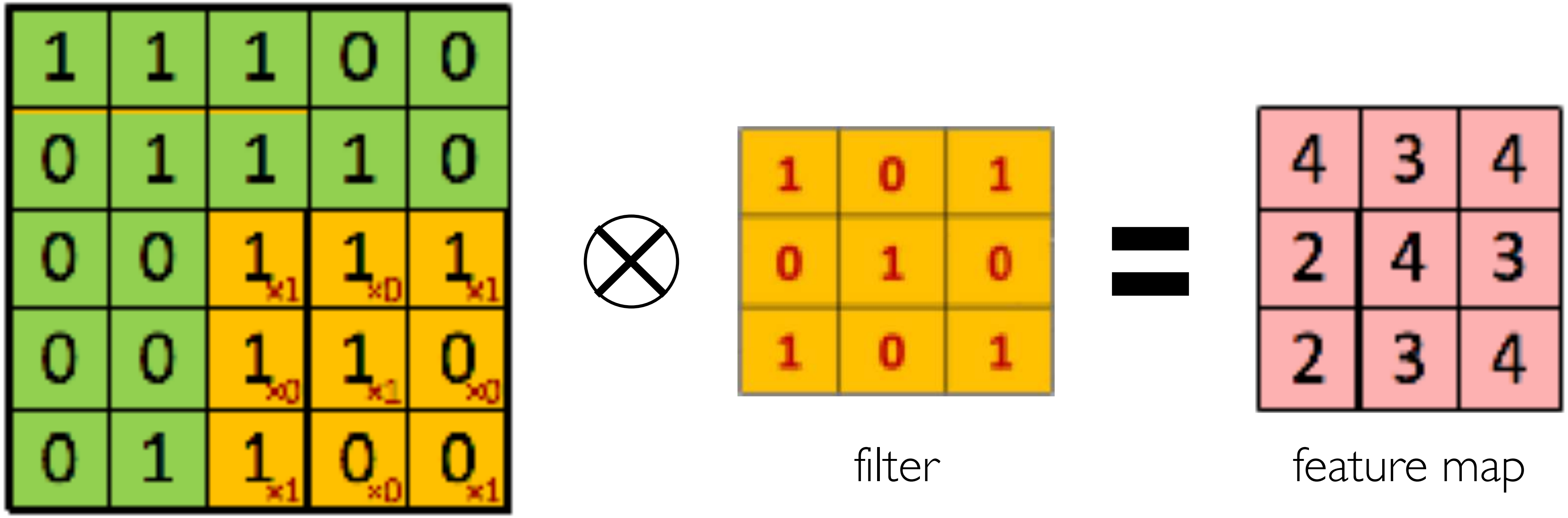
Convolution



Convolution

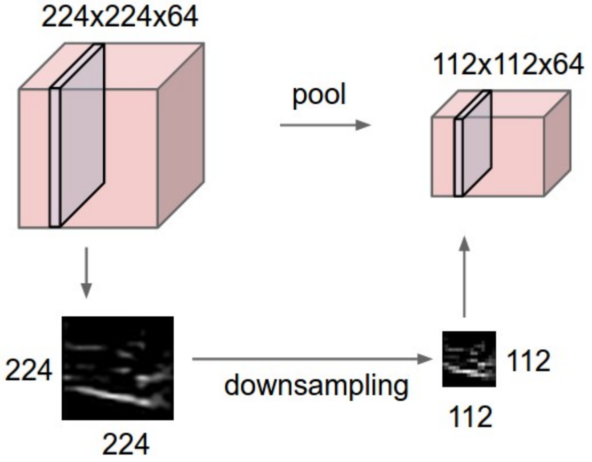


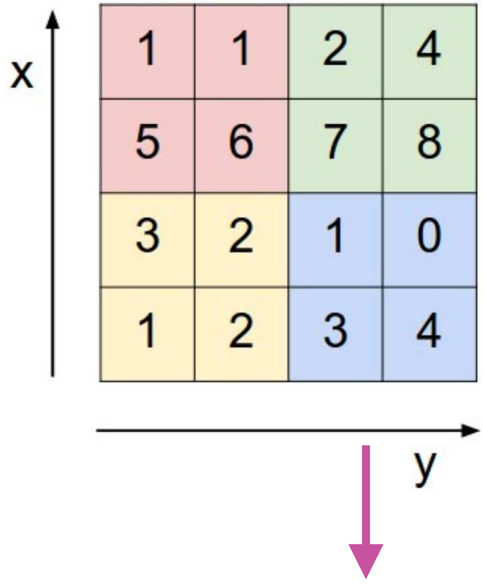
Convolution



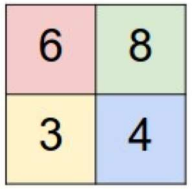
CNN components

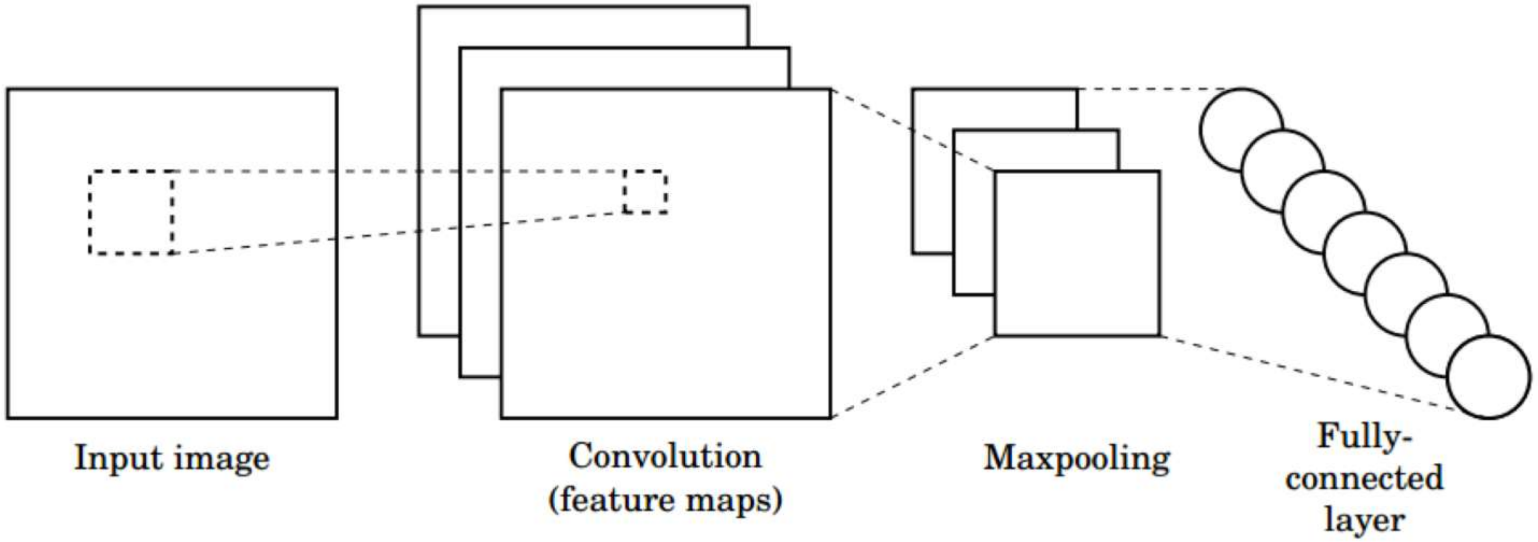
- **Convolution**: Apply filters with learned weights to generate feature maps from images
- **Non-linearity**: Often ReLU.
- **Pooling**: Down-sampling operation on feature maps to summarise the presence of features (e.g., averaging or taking the maximum) to increase translational invariance
- **Fully connected layers**: similar to classical ANN arch.



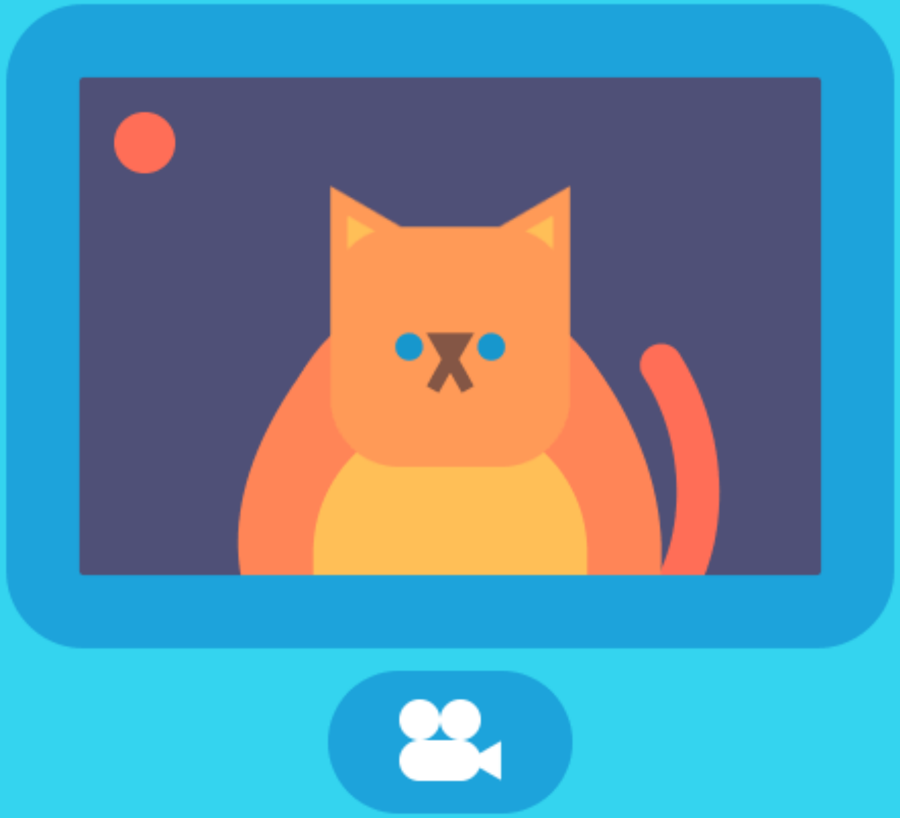


Max-pooling
➜ improve invariance to small-scale translations





Sequences / time-series analysis



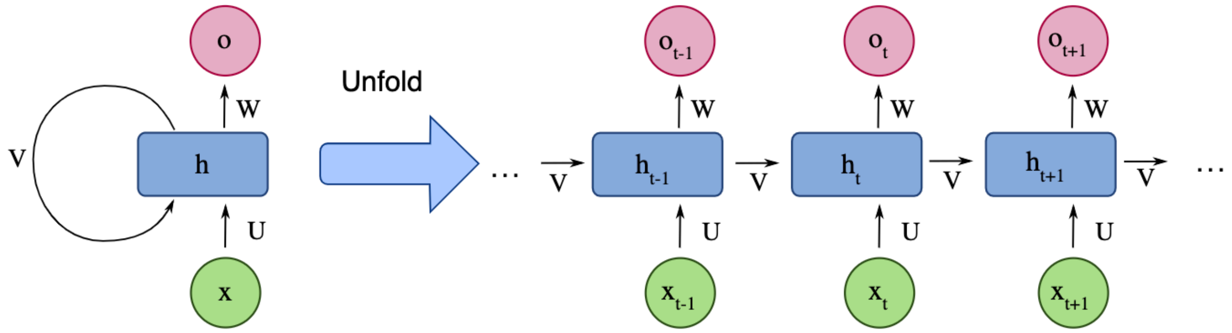
Sequences / time-series analysis

- Can you predict the next word?

"**France** is where I grew up, but I now live in Boston. I speak fluent ____."

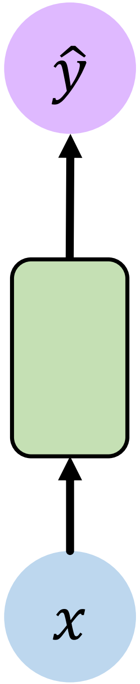
- Information from the distant past is required for robust predictions…
- In general…
  - Variable-length sequences
  - Track long- & sort-term trends
  - Ordered information
  - Shared parameters across sequences

- ➜ one solution is recurrent neural networks (RNNs) - nodes connected to form a directed graph along a temporal sequence.



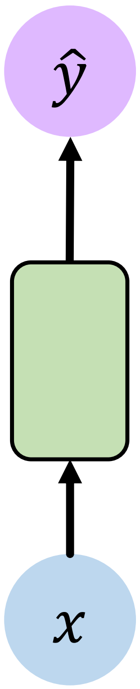
MIT - 6.S191 (2019)

Recurrent neural networks

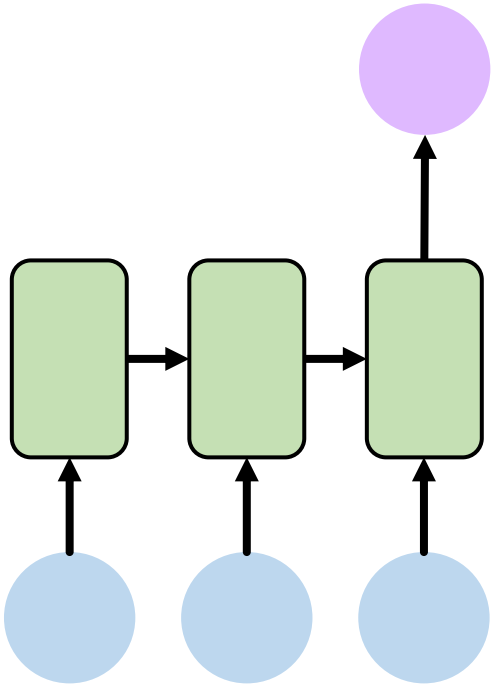


One to One
"Vanilla" neural network

Recurrent neural networks

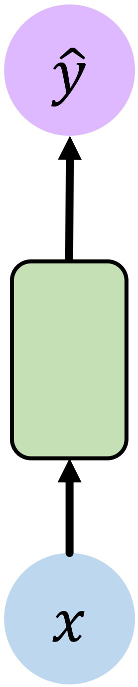


One to One
"Vanilla" neural network

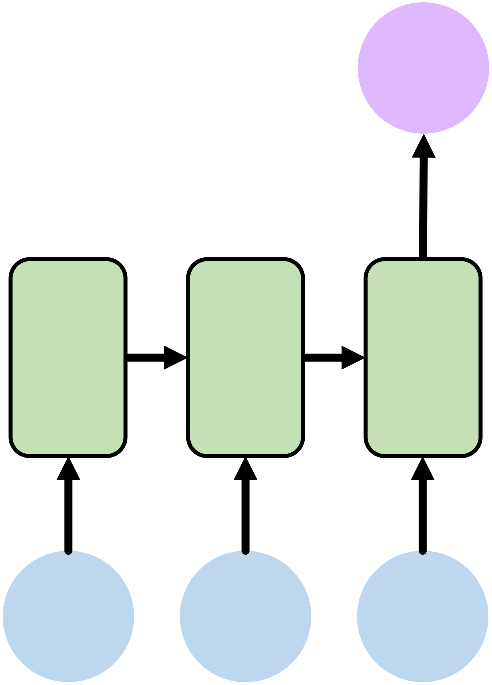


Many to One
Sentiment Classification

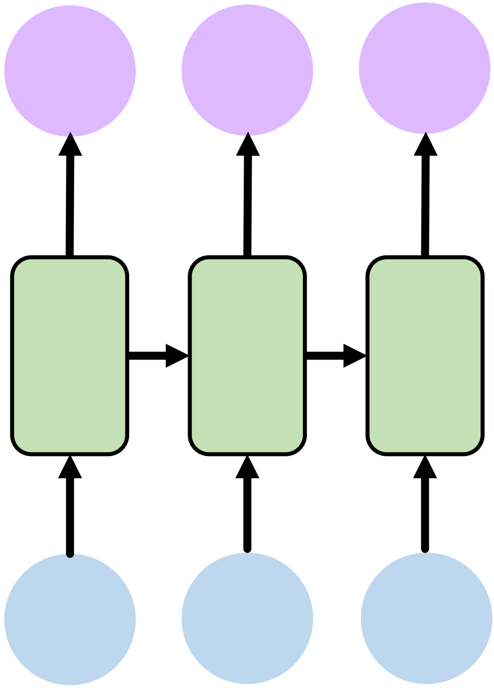
Recurrent neural networks



One to One
"Vanilla" neural network

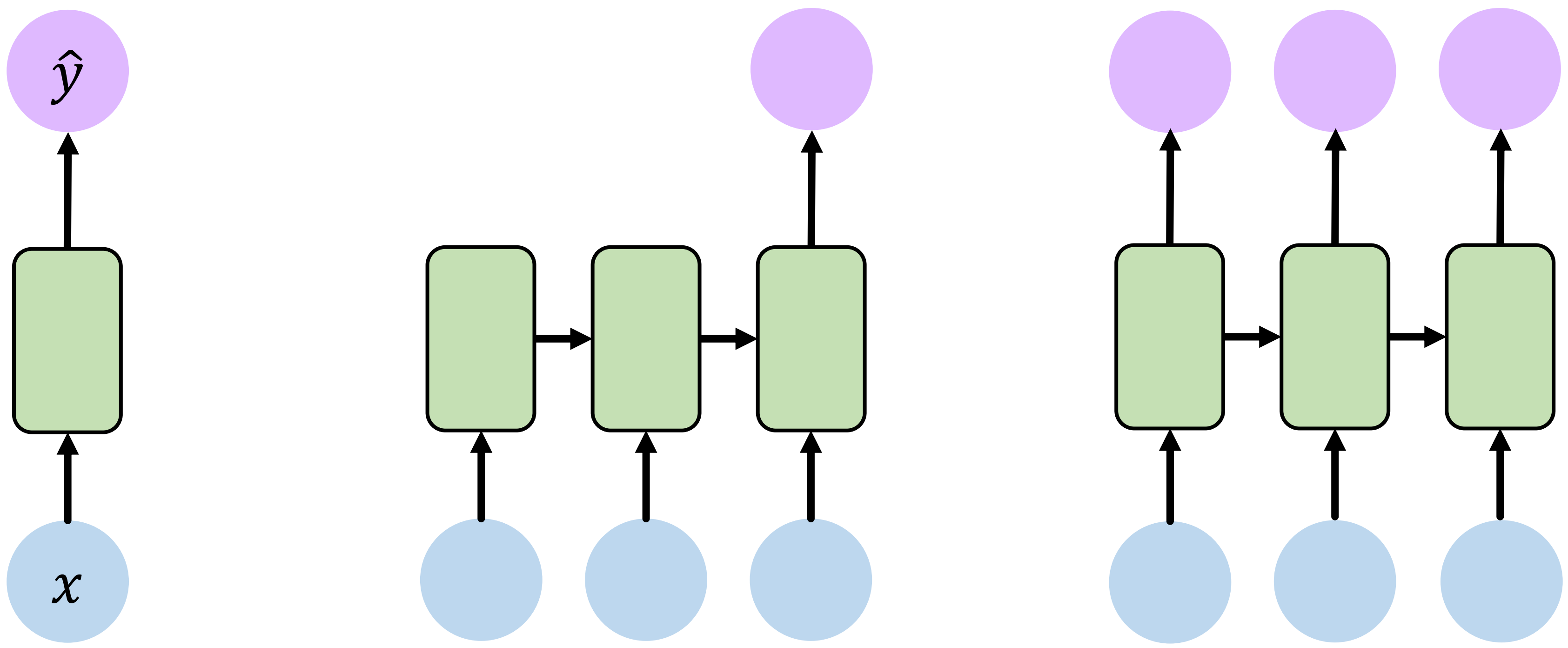


Many to One
Sentiment Classification



Many to Many
Music Generation

Recurrent neural networks



One to One
"Vanilla" neural network

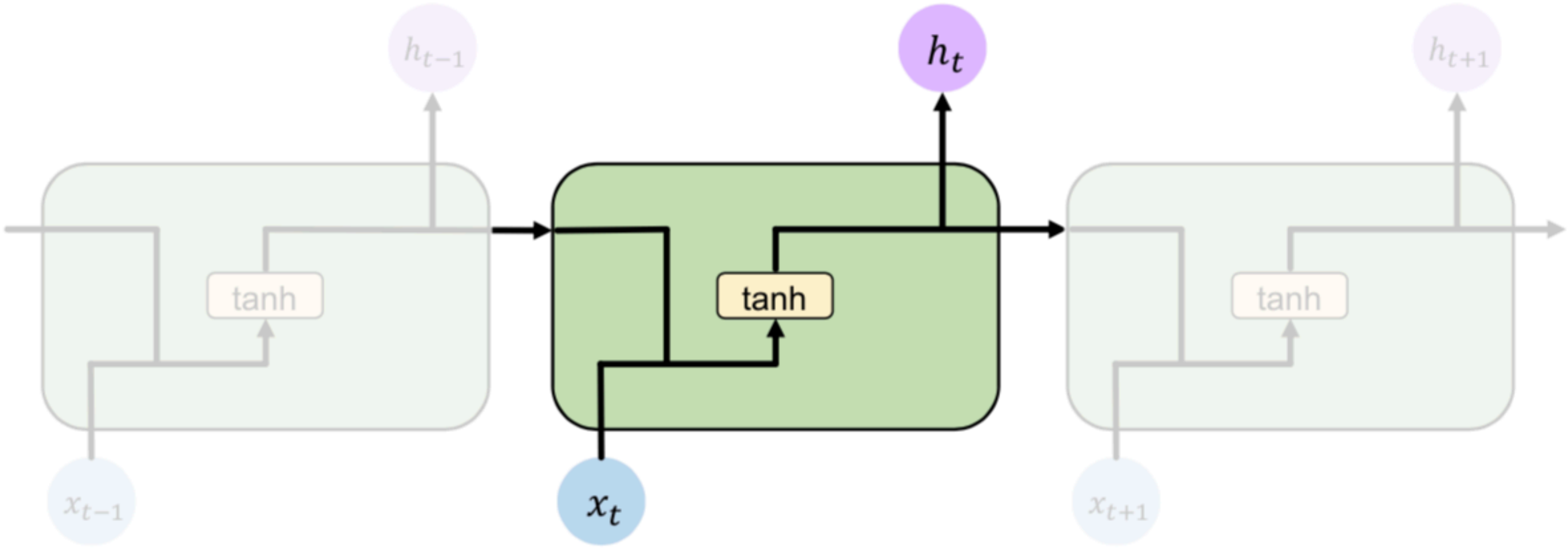
Many to One
Sentiment Classification

Many to Many
Music Generation

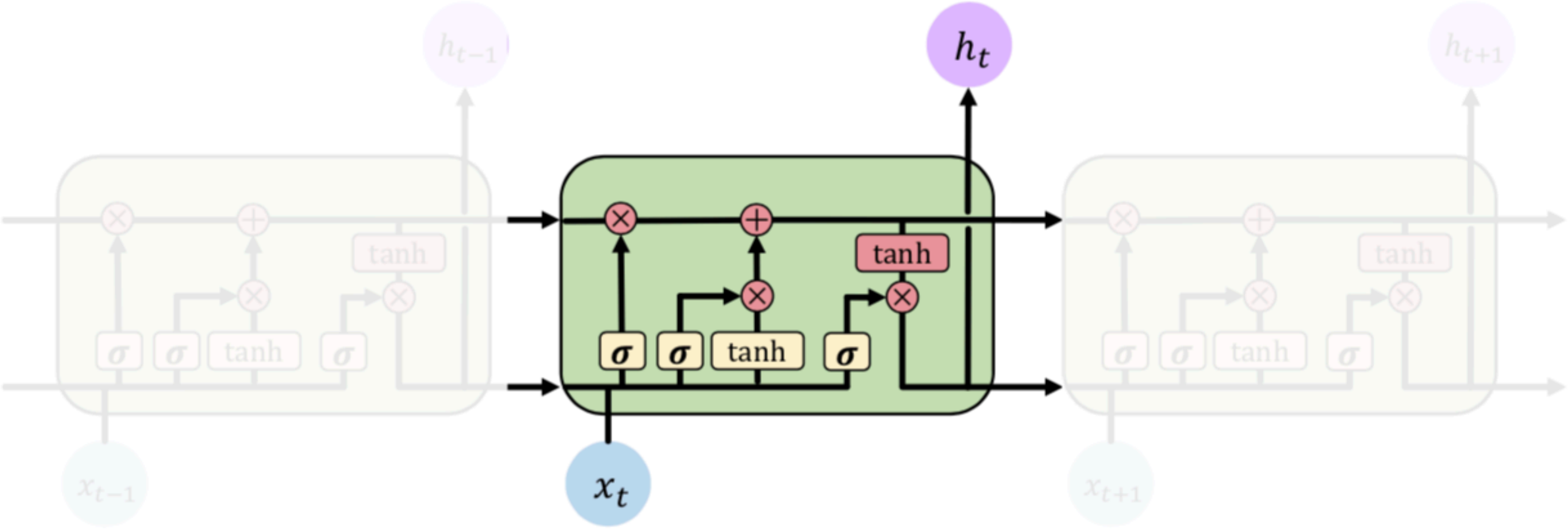
… and many other architectures and applications

Long short term memory (LSTMs)

- Standard RNN

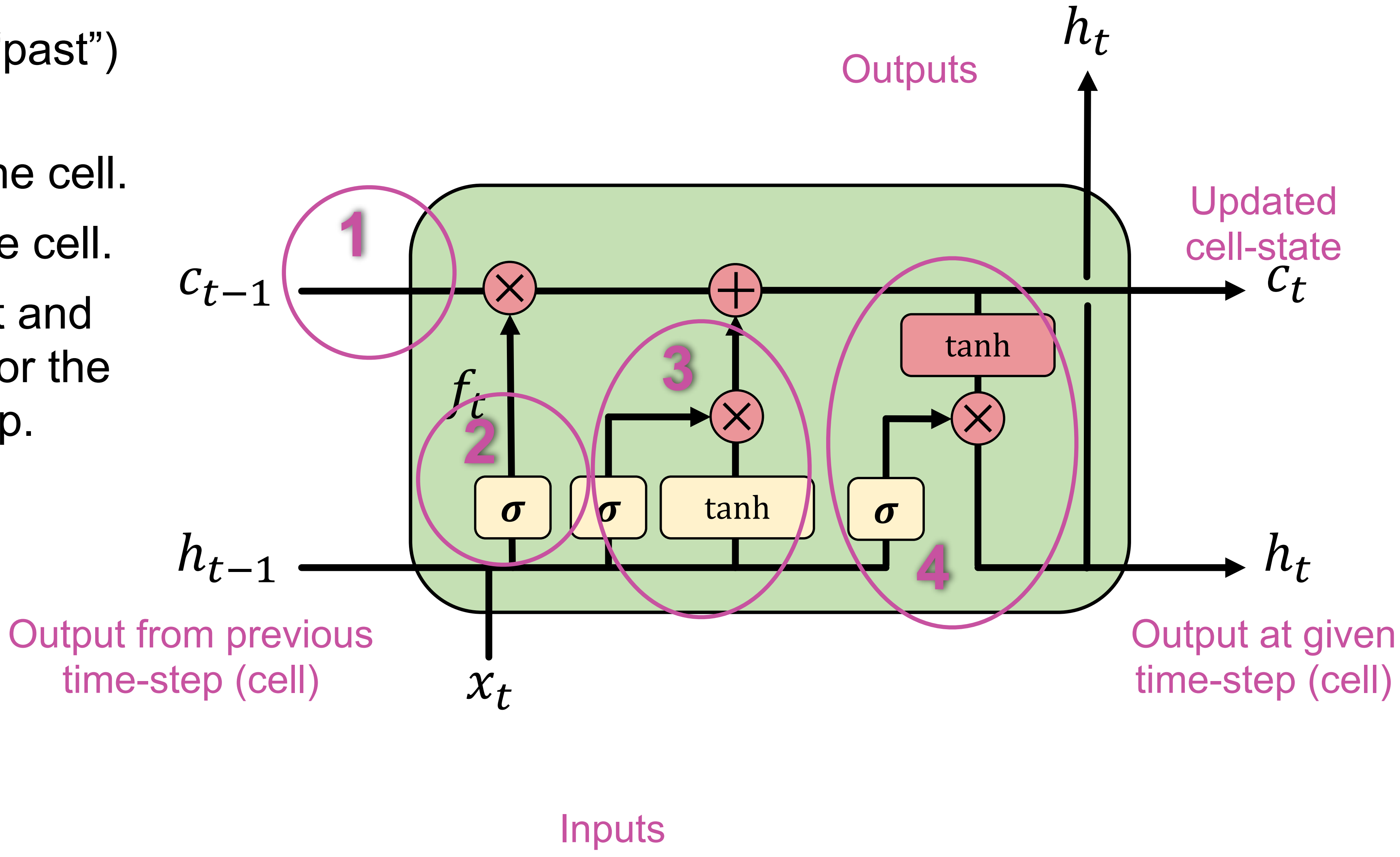


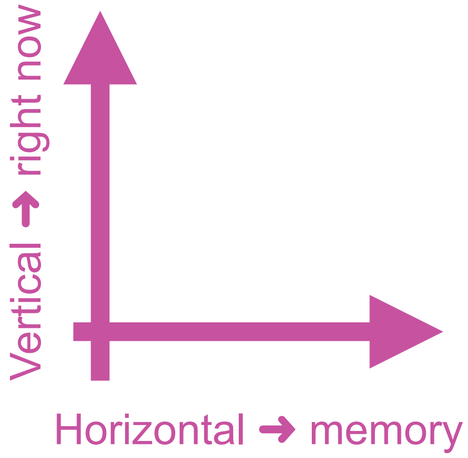
- LSTM



Vanishing gradient problem ➜ Long-short memory units (LSTM)

1. Pass-in the previous ("past") state for modification.
2. "Forget" a sub-set of the cell.
3. Update a sub-set of the cell.
4. Derive a filtered output and an updated cell-state for the next ("future") time-step.





Reinforcement learning

See also: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
(➜ Inc. video lectures: https://www.youtube.com/playlist?list=PLbWDNovNB5mqFBgq7i3MY6Ui4zudcvNFJ)

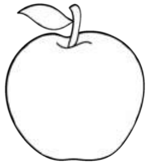
Types of learning problems

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn function to map
$$x \rightarrow y$$

Apple example:



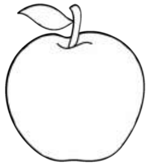
This thing is an apple.

Unsupervised Learning

Data: x
x is data, no labels!

Goal: Learn underlying structure

Apple example:

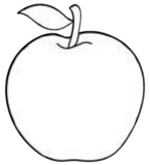
This thing is like the other thing.

Reinforcement Learning

Data: state-action pairs

Goal: Maximize future rewards over many time steps

Apple example:



Eat this thing because it will keep you alive.

Reinforcement learning

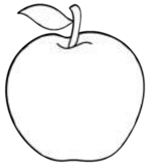
- **Strategy**: find policy of “behaviour” that maximises future rewards, R_t.
- Particularly well-suited to problems that include a long-term versus short-term reward trade-off.
- Balance exploration (of uncharted territory) and exploitation (of current knowledge).

Reinforcement Learning

Data: state-action pairs

Goal: Maximize future rewards over many time steps

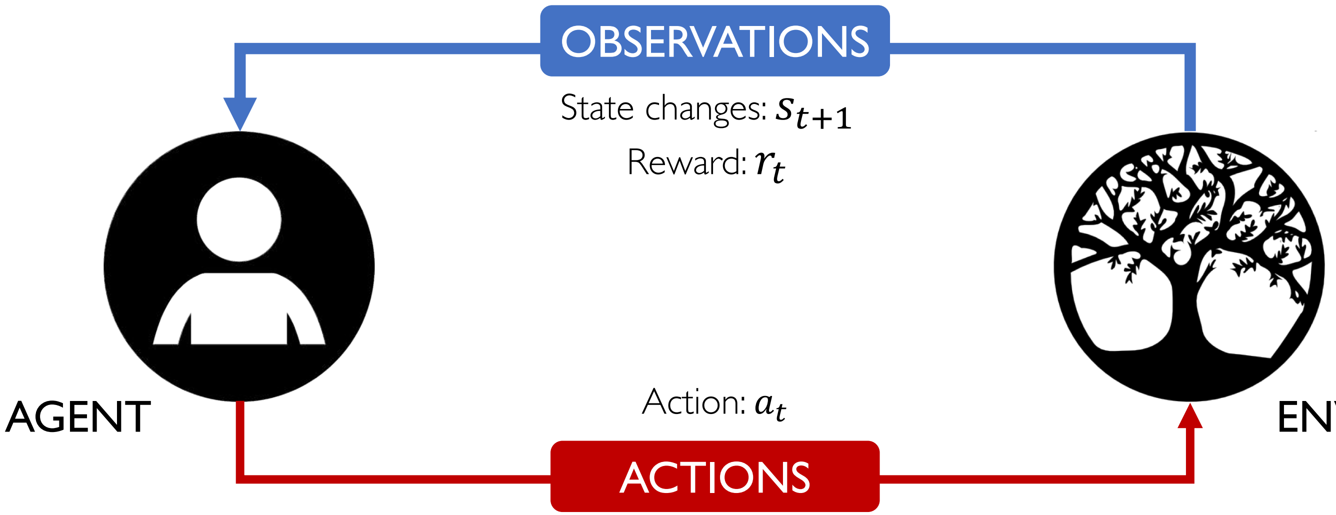
Apple example:



Eat this thing because it will keep you alive.

Agent vs. environment

- **Value learning**: maximise value function for optimal policy (optimises approximated average) rewards for all state-action pairs.
  ➜ Works for discrete / small action spaces.
- **Policy learning**: directly optimise the policy (e.g., with gradient-based methods).
  ➜ Models continuous action spaces



- R_t : Sum of rewards for time, t.
- Q (value function): total future reward (state, s, & action, a).
- π: policy to infer best action to take

$$Q(s, a) = \mathbb{E}[R_t]$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Discounted Total Reward

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i = \gamma^t r_t + \gamma^{t+1} r_{t+1} \ldots + \gamma^{t+n} r_{t+n} + \cdots$$

γ: discount factor

(My personal) important takeaways

Garbage in ➜ garbage out

- **Know your data:**
  - Use complete datasets
  - (For most cases) use representative samples to minimise systematic (sampling) bias
  - Transform input variables to reflect expected signals (e.g., use logarithm of energy for PL analyses)
  - Normalise inputs for homogenise numerical operations
  - Do not add unnecessary / noisy features (or data in general)
  - Fold-in known systematics on inputs & outputs to increase robustness

Garbage in ➜ garbage out

- **Keep it simple:**
  - Prefer feature-engineering over complex architectures
  - Don’t get too excited about bleeding edge technologies

- **Cross-check your results:**
  - Assume outliers will be catastrophic (can probabilistically be identified)
  - Reweight your training sample / experiment with different loss functions
  - Use control regions to test generalisability (extrapolation is notoriously difficult)
  - Check control plots through the pipeline ➜ verify intermediate performance metrics
  - Compare with classical methods where possible

Questions… ?