Terminology

In the lecture material, we use a few different names for closely related things.



- ➔ **Artificial Neural Networks (ANNs)**
- ➔ **Multilayer Perceptrons (MLPs)**
- ➔ **Feed-forward Neural Network**
- ➔ **Deep Neural Network (DNN)** (coming in a few weeks)

Are these all the same? How do they differ?

When should I use an Artificial Neural Network (AKA Multilayer Perceptron)?



I have a problem that I want to solve with machine learning.

Which of these three strategies would you recommend?

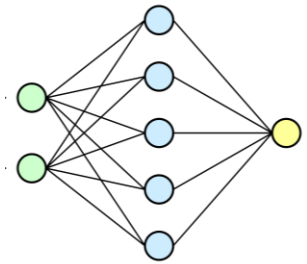
- Go from simple to increasingly more complex models (like an ANN) until I get satisfactory results
- Go from complex models (like an ANN) to increasingly simpler models until I get unsatisfactory results
- Go straight to a model like an ANN (which should work in most cases)

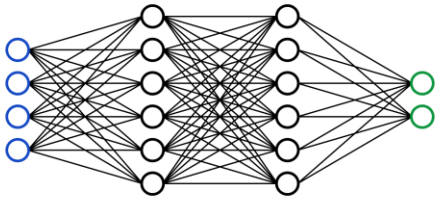
Or are there other options?

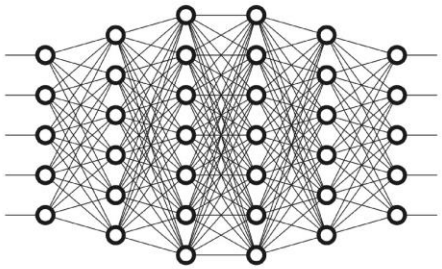
What type of neural network should I use?

There are many different ANN models, as we can vary:

- Number of hidden layers
- Number of nodes in each hidden layer
- Type of activation function (sigmoid, ReLU, ...)



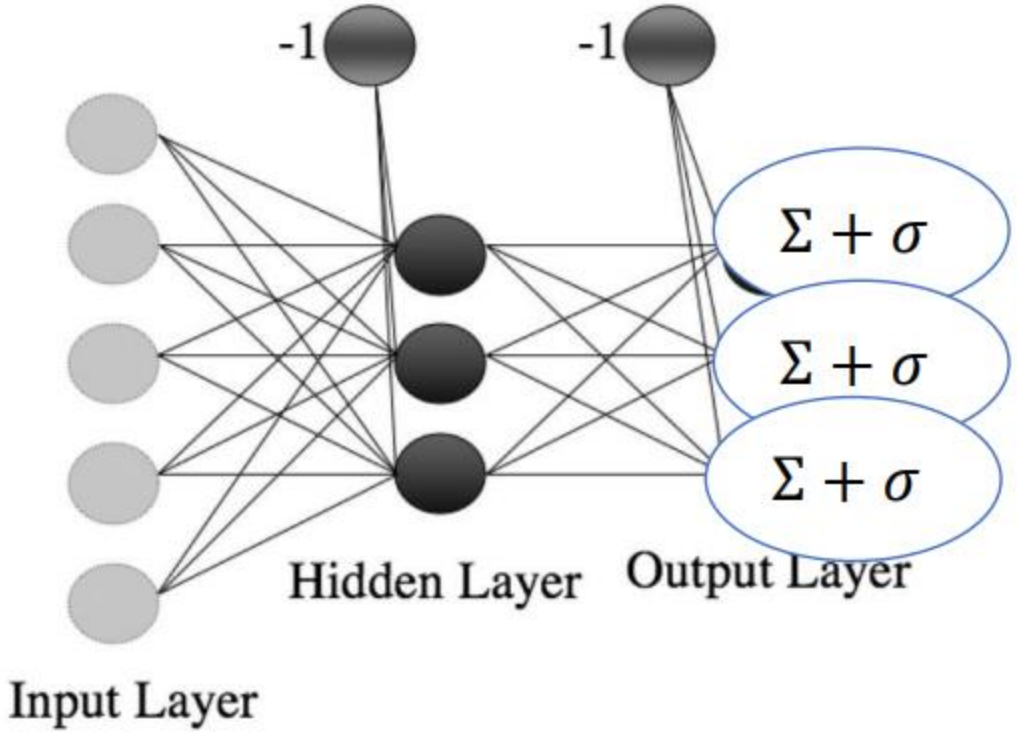


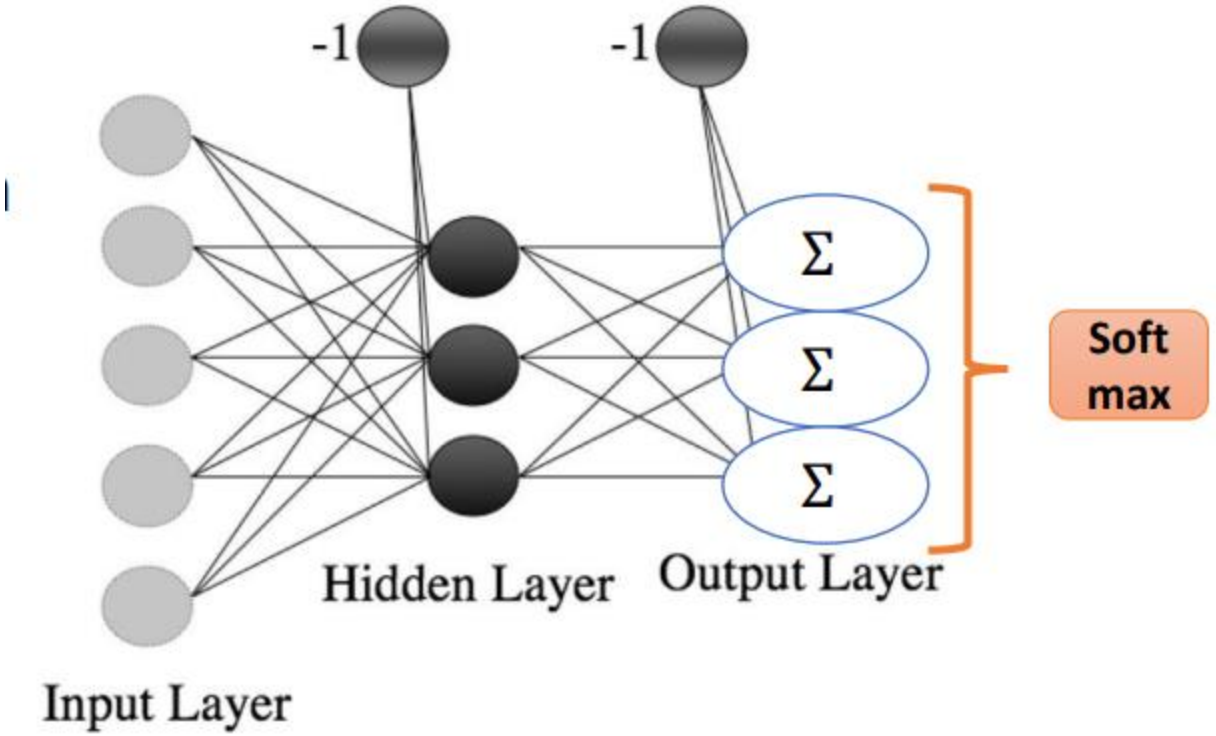


How do I go about to choose the right one?

- Any recommended standard choices?
- Any rules of thumb?
- Or should I try many different choices and compare?

Multi-class vs multi-label classification – what is the difference?





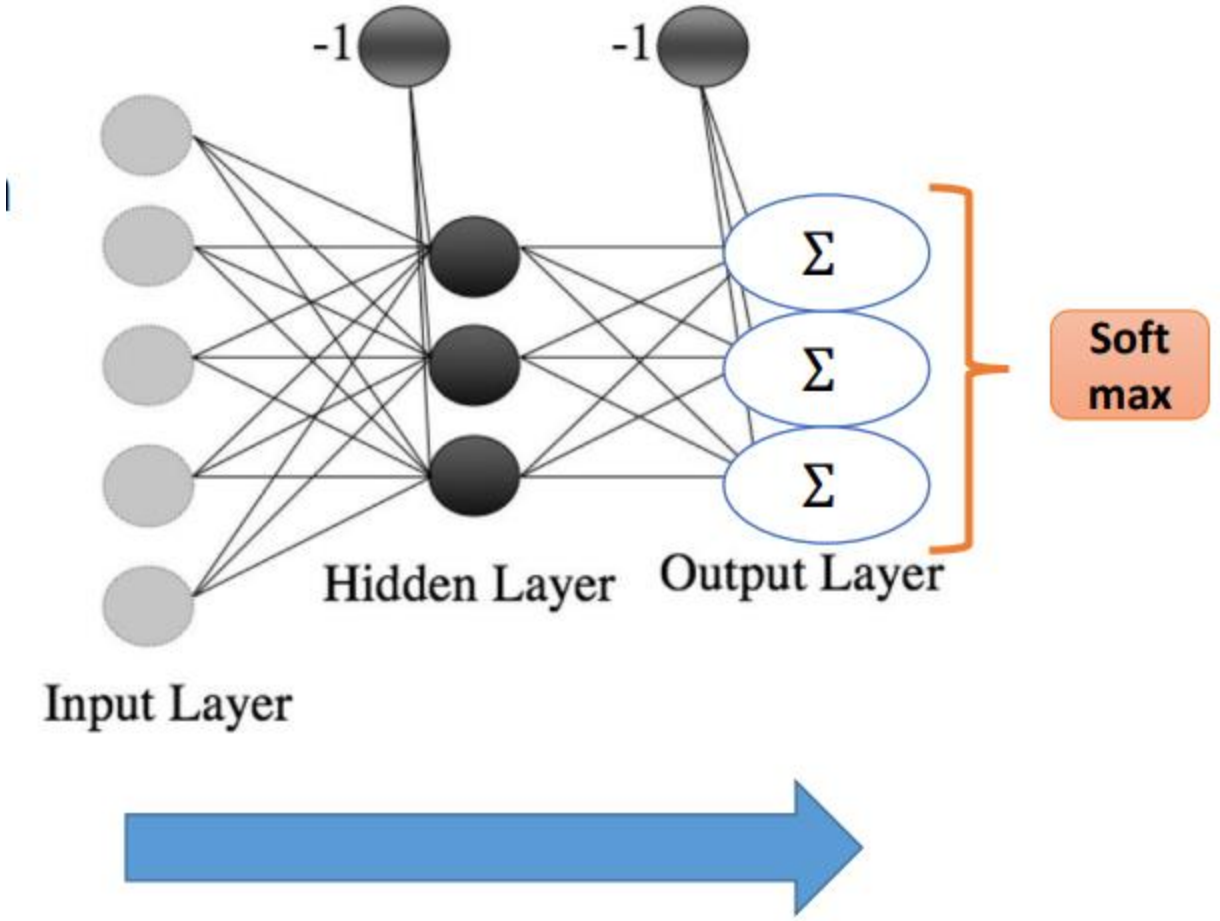
Multi-class vs multi-label classification – what is the difference?

- **Multi-class** usually stands for "*choose **one** class out of N classes for every instance*"
- **Multi-label** usually stands for "*choose **any number** of classes out of N for every instance*"

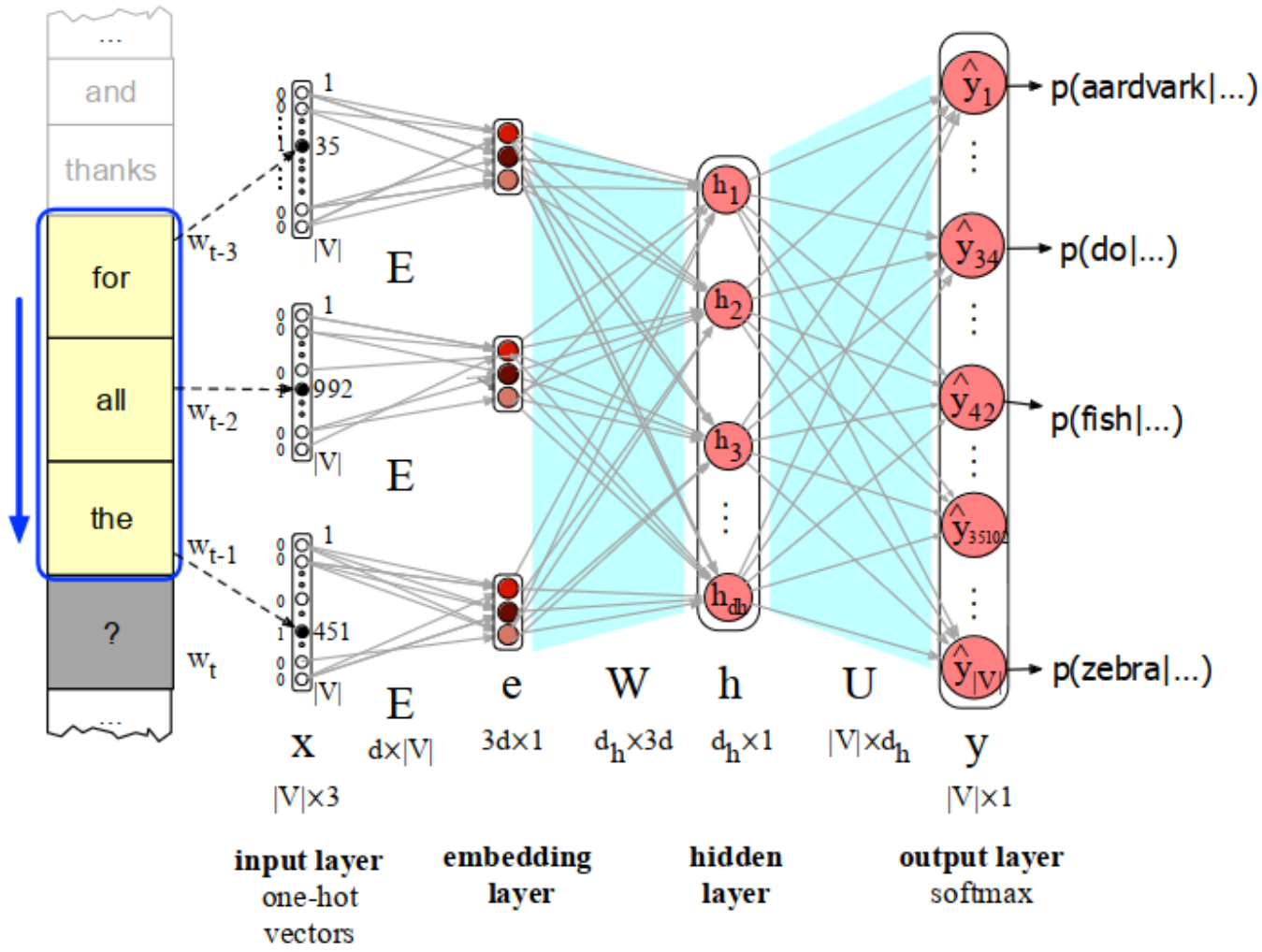
- Multi-label classification is a more complex task, obviously.

Multi-class classification vs ChatGPT – what is the difference?





Multinomial classification in a language model



(from Jurafsky and Martin, 2024)

Pros and cons of ANNs

The course introduces different machine learning methods, such as:

- k-nearest neighbor regression and classification
- linear regression and classification
- logistic regression and classification
- artificial neural networks

If we average performance over all imaginable problems, then no method is better than any other. So ANNs must be good for some problems and bad for others.

What type of problems are ANNs good for?

Any particular strong and weak sides of ANNs?



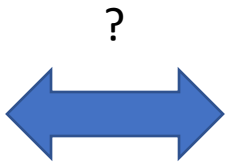
Is backpropagation inspired by the brain?

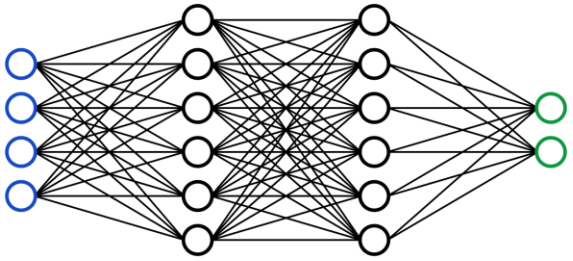
The structure of perceptrons and ANNs are inspired by the brain.

Learning algorithms for networks can also be inspired by the brain (Hebb rule!)

- **→ What about backpropagation?**
- **→ Any biological link there?**
- **→ And should we even care?**







Should artificial neural networks be like brain?



Do artificial neural networks capture the source of intelligence in the brain?

Modern ANNs abstract away many things that the brain does: Chemical properties, electrical signals, local learning rules

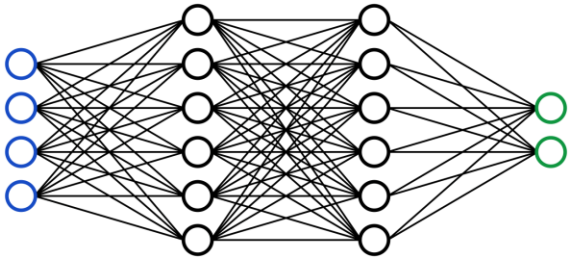
→ Have ANNs captured the essential parts that make brains intelligent?

→ Are chemical, electrical properties NOT important for intelligence? Just a bi-product of evolution?

→ Are there things the brain still does much better than ANNs?



?



Other learning algorithms for ANNs

The purpose of backpropagation is to find weights that minimize a given cost function. It uses gradient descent (or a variant thereof).

There are many other optimization methods, for example:

- Hill climbing
- Simulated annealing
- Evolutionary algorithms
- Newton's method
- Conjugate gradient
- Quasi-Newton

Are these used, too? Are they better?

Other learning algorithms for ANNs

The purpose of backpropagation is to find weights that minimize a given cost function. It uses gradient descent (or a variant thereof).

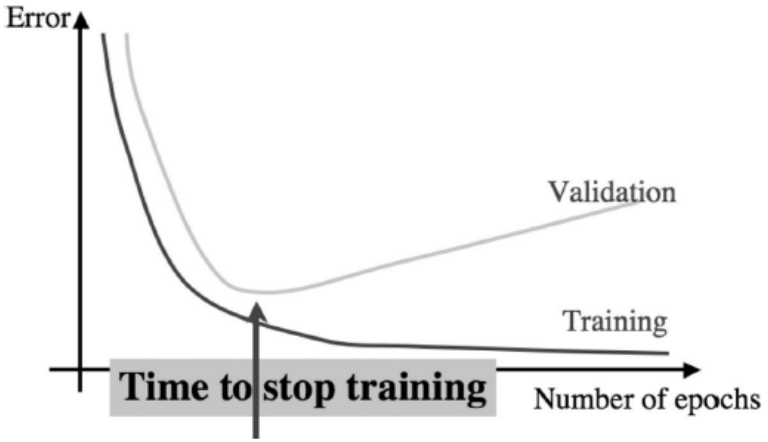
There are many other optimization methods, for example:

- Hill climbing
- Simulated annealing
- Evolutionary algorithms
- Newton's method
- Conjugate gradient
- Quasi-Newton

Are these used, too? Are they better?

Short answer: no, almost never (mostly because of computational efficiency)

Early stopping



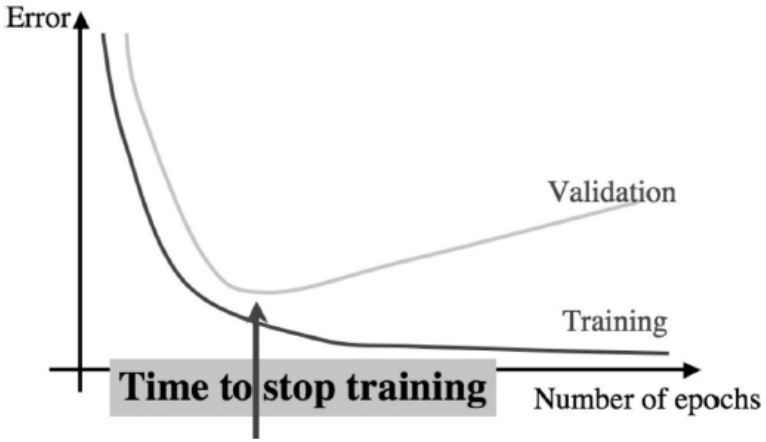
Principle:

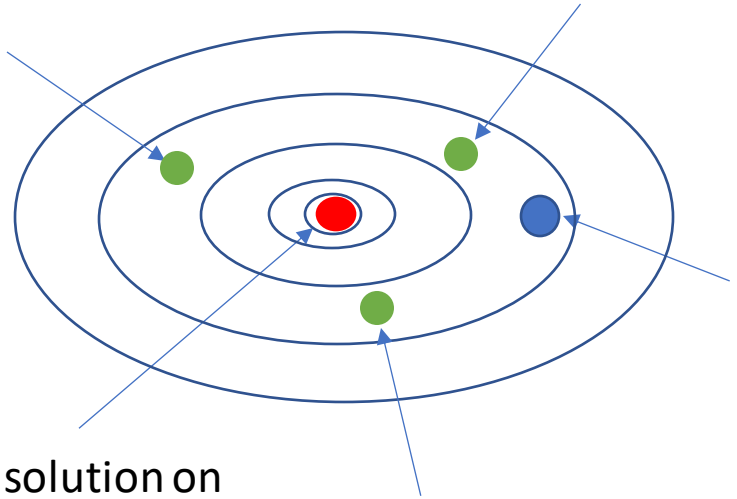
Stop the learning when the error starts increasing on the validation data set

Idea:

When the error starts increasing on the validation data, we are likely overfitting to the training data

Early stopping





True solution (optimal solution on future data sets)

Optimal solution on training data set

- It seems early stopping is very dependent on the start state?
- Can early stopping give a worse result than not stopping early?

Initializing weights

- The weights:
  - should not be initialized to 0
  - should be initialized to random numbers
  - should be initialized to numbers between -1 and 1
- In addition, Marsland recommends to multiply with $\frac{1}{\sqrt{m}}$
  - where m is the number of input nodes

- Why not initialize to zero?

- Why not initialize with a small number (e.g. all weights 0.1)? Is the randomness important?

Initializing weights

- Initialized with zeros, an ANN will not learn anything at all
- Initialized with the same value, all neurons will most probably learn exactly the same thing: not efficient

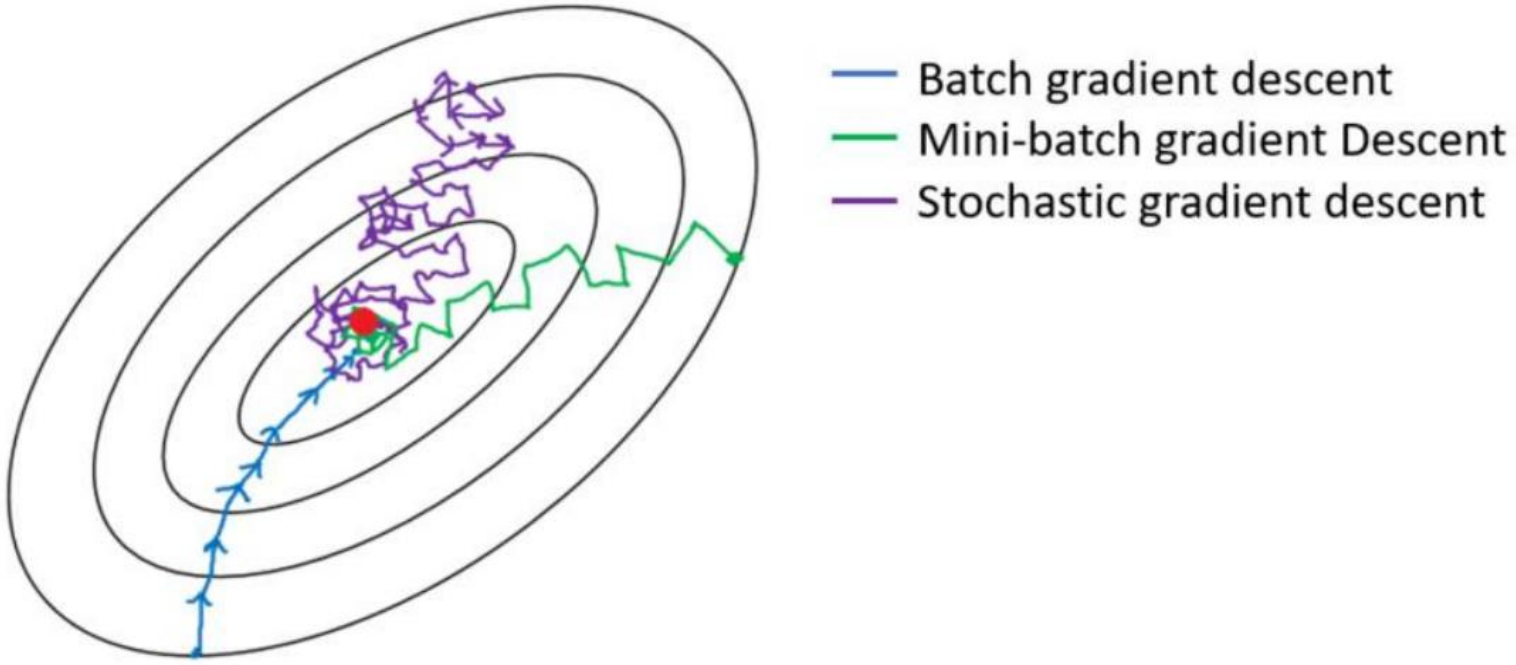
- torch.nn.init
  - calculate_gain()
  - uniform_()
  - normal_()
  - constant_()
  - ones_()
  - zeros_()
  - eye_()
  - dirac_()
  - xavier_uniform_()
  - xavier_normal_()
  - kaiming_uniform_()
  - kaiming_normal_()
  - trunc_normal_()
  - orthogonal_()
  - sparse_()

- Assortment of initialization methods in PyTorch

- https://pytorch.org/docs/stable/nn.init.html

Variants of gradient descent

- Reminder: What is the difference between these? What are their pros/cons?
- Which one is applied most in practice? Why?



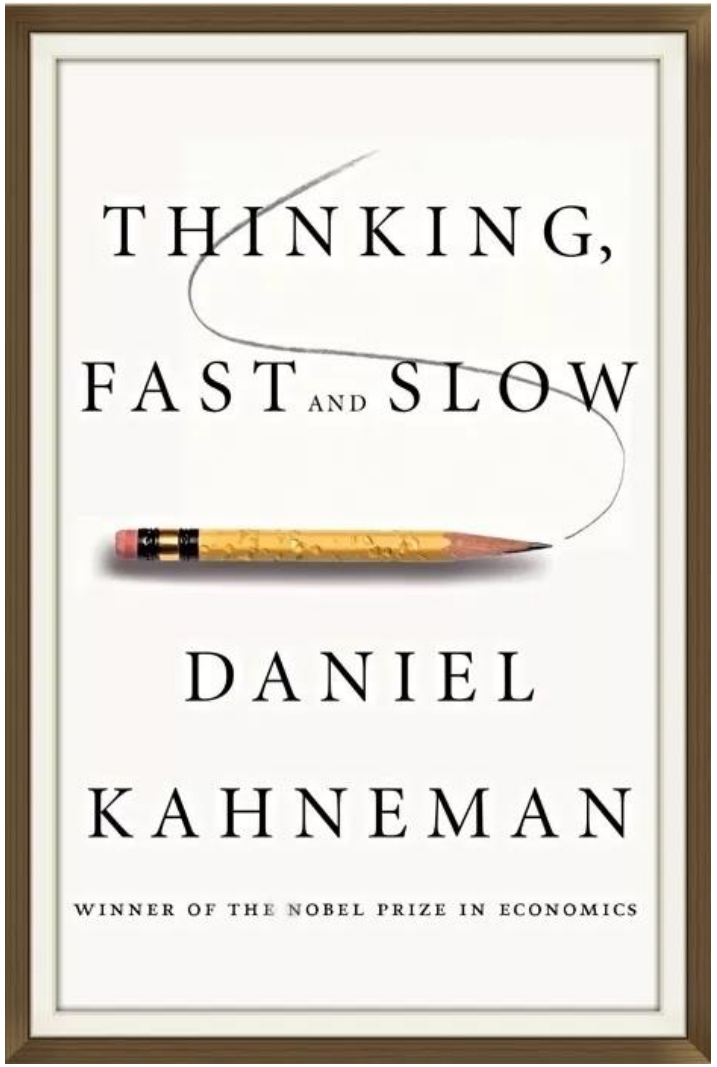
Fast and slow thinking

It has been hypothesized that humans rely on 2 different modes of thinking:

System 1: Fast, instinctive, emotional

System 2: Slow, deliberative, logical

The best way to make decisions depends on the context



Fast and slow thinking

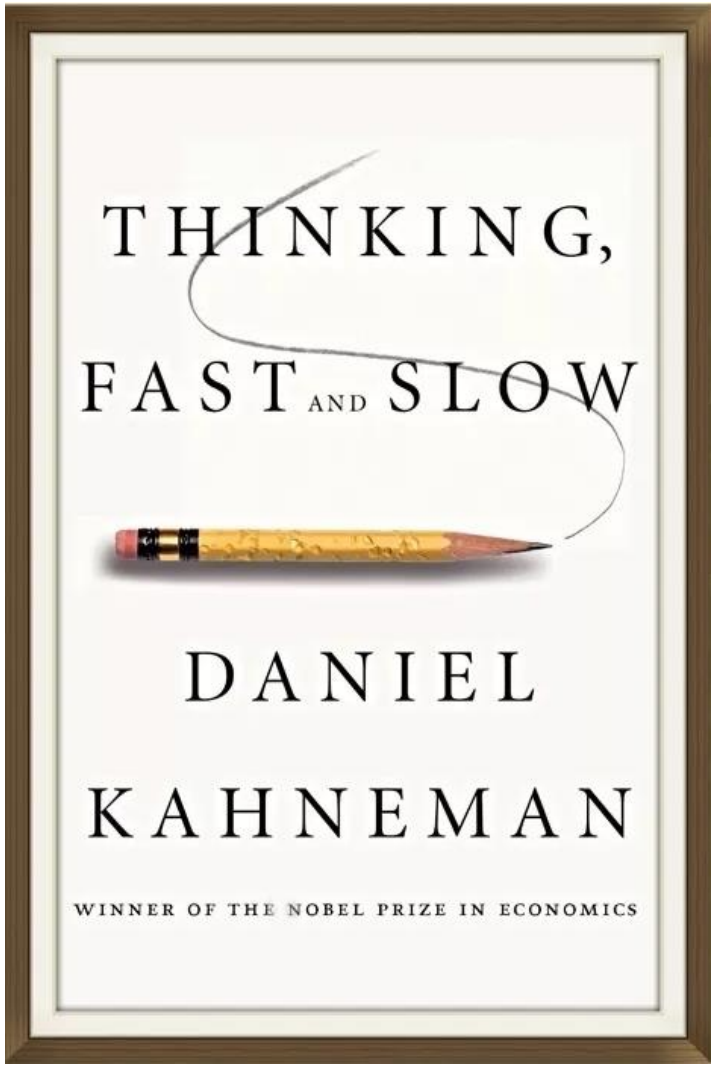
- Some researchers argue that ANNs can only do System 1 (fast) thinking, and that this is limiting:
- First, it makes them less reliable, as they don’t reason deeply before making a decision
- Second, it could mean there are things they can never do, such as long-horizon planning.



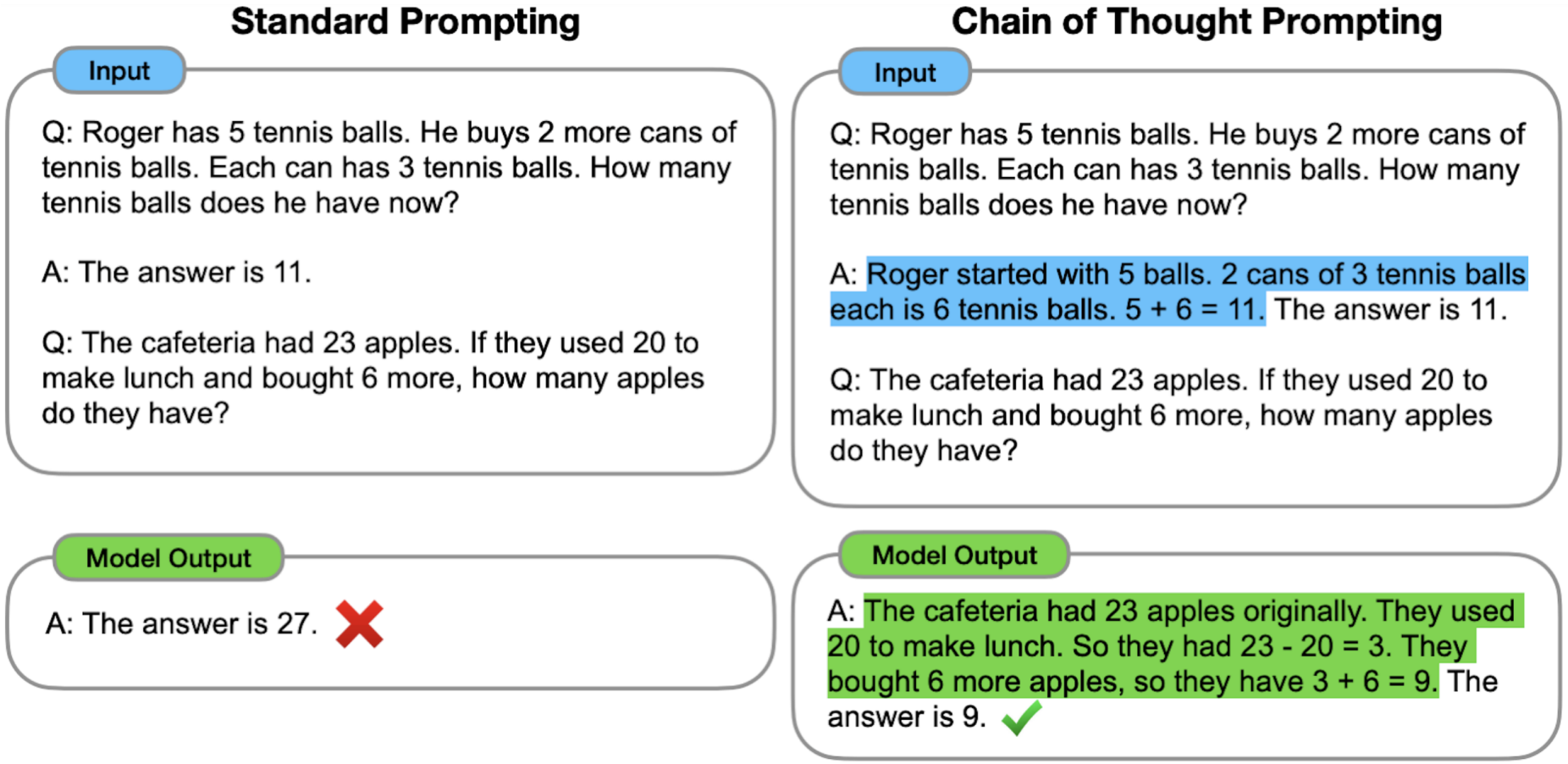
Sample of how an object in a noncanonical orientation and context fools many current object classification systems (Alcorn et al., 2018)

Fast and slow thinking

- Do you agree neural networks are mainly doing «system 1»-style thinking?
- Do AI agents need to have a «system 2»?
- If so, how could we enable reasoning, planning, logical thinking in future AI agents?



Fast and slow thinking



We do have the so called "chain-of-thought prompting" in large language models. But it is still very far away from real **reasoning**. *(image: Google Research)*

One possibility: Neuro-Symbolic AI

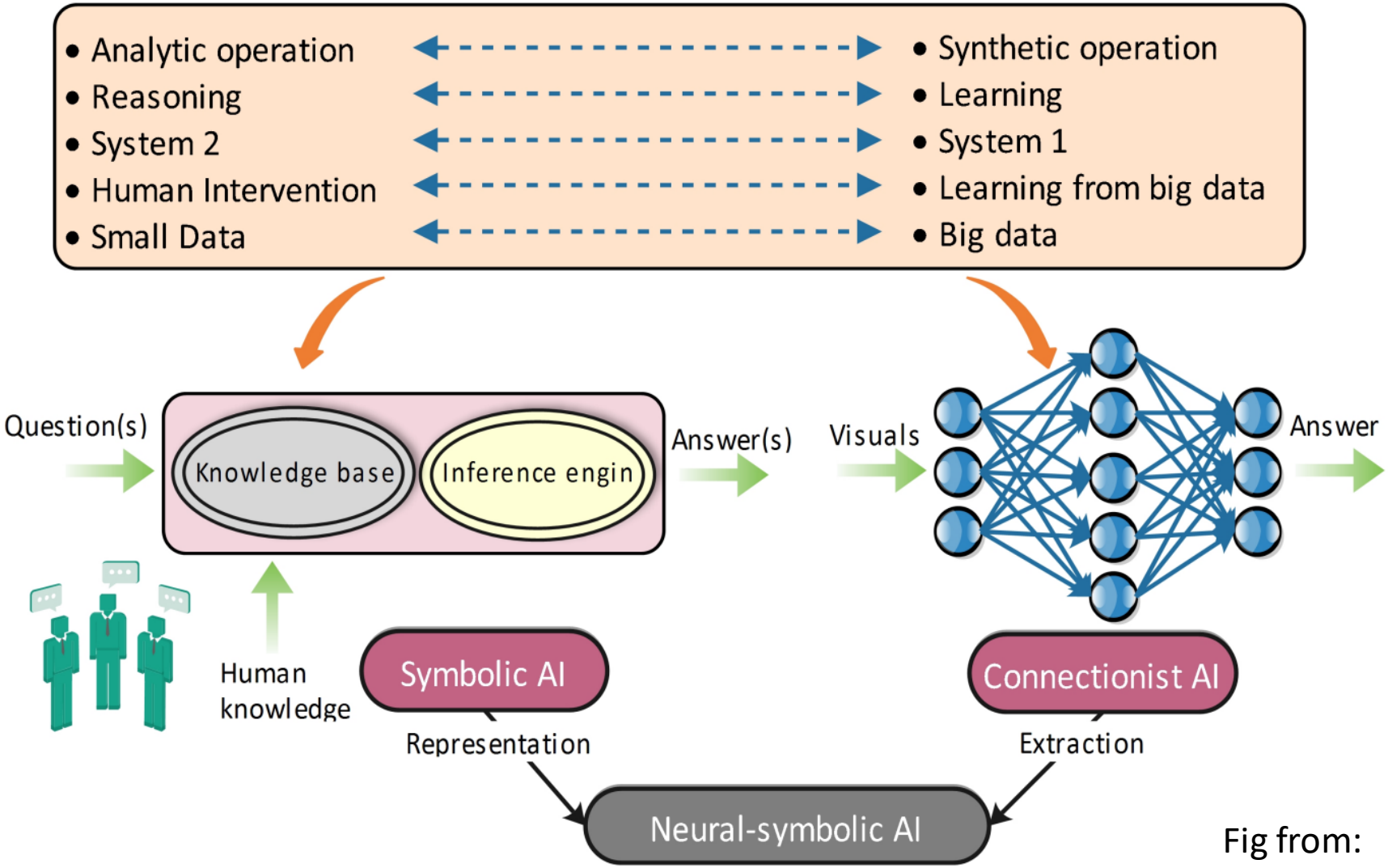


Fig from:
https://arxiv.org/pdf/2208.00374.pdf