Machine Learning

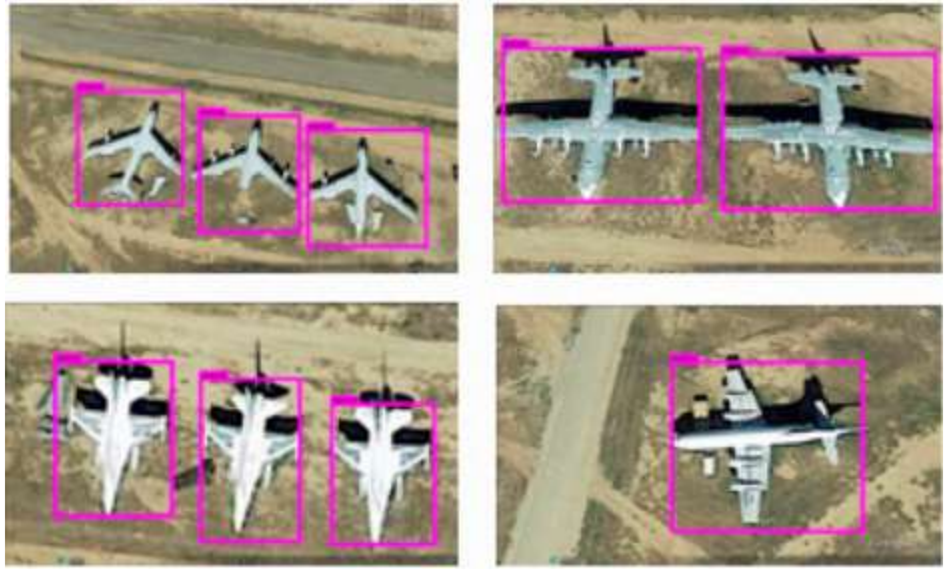
week 2, 2024

The Problem of Perception

Some problems are hard to solve by programming

Example: computer vision

```
    if image_pixels[223][437] == "grey":
        return "Airplane"
    else:
        return "Ground"
    if image_pixels[223][438] == "grey":
        return "Airplane"
```



Radovic, Matija, Offei Adarkwa, and Qiaosong Wang. "Object recognition in aerial images using convolutional neural networks." *Journal of Imaging* 3.2 (2017): 21.

Difficult Functions

AlphaStar playing StarCraft II:

Input: recent history $(t-1, t-2, \ldots)$ of 10000 input variables

Output: choose between 10^{26} possible actions

This extremely difficult function is computed by a neural network

Vinyals, Oriol, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning." *Nature* 575.7782 (2019): 350-354.

Motivation

For problems that:

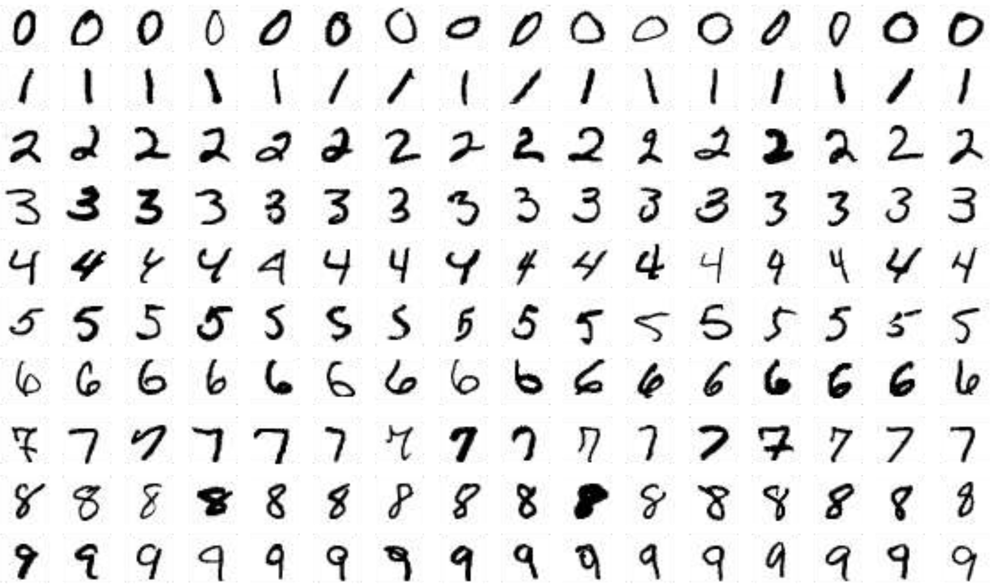
- are unreasonably difficult to compute
- or, we have no idea *how* to compute

Machine learning may provide an **approximate** solution

Usage Example

Problem: recognize handwritten digits automatically

You have some examples:



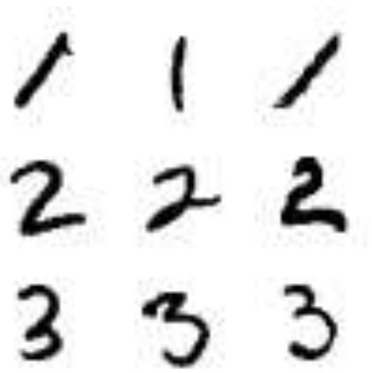
(Like the 60000 images

in the MNIST dataset;

https://en.wikipedia.org/wiki/MNIST_database)

Usage Example

Step 1: show the examples to the magic black box:



“These are 1-s”

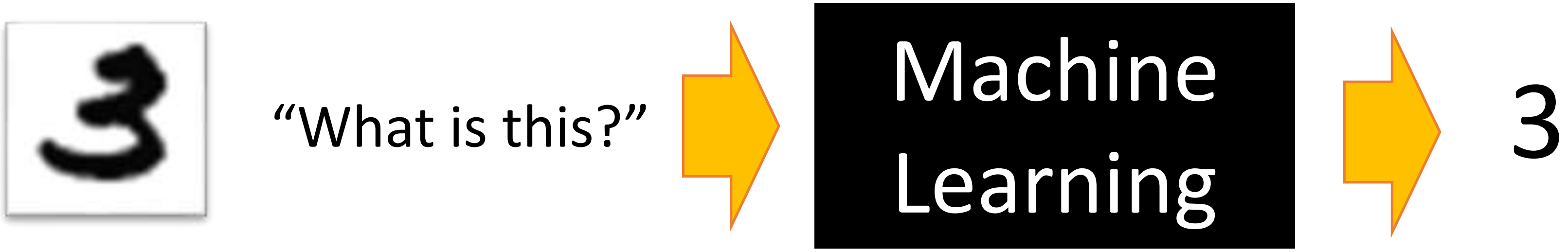
“These are 2-s”

“These are 3-s”

Machine Learning

Usage Example

Step 2: the magic box will start answering **based on the examples**



Problem solved!

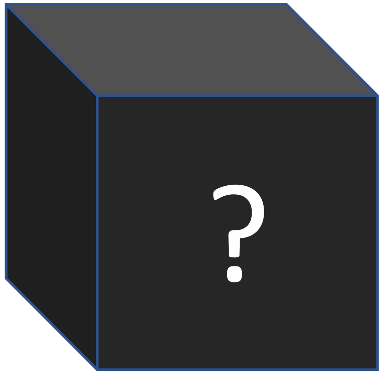
(Provided that it works reliably enough –

usually the data you needed this for should be similar to examples)

Making the magic box

There are many machine learning methods,
we will study only a few in this course

Starting focus: neural networks

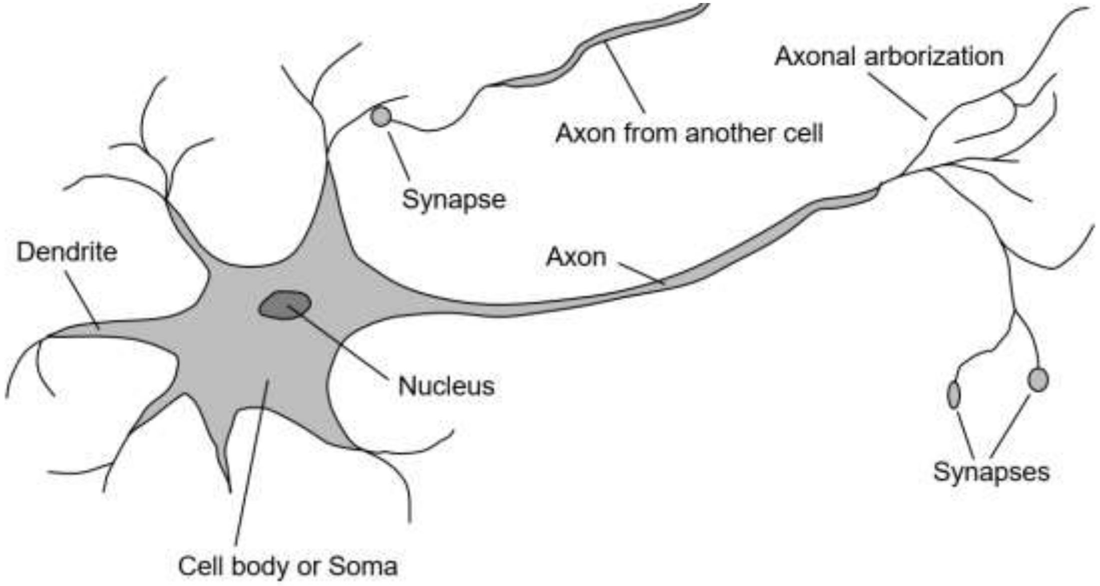


Artificial Neuron

Idea from 1950-s:

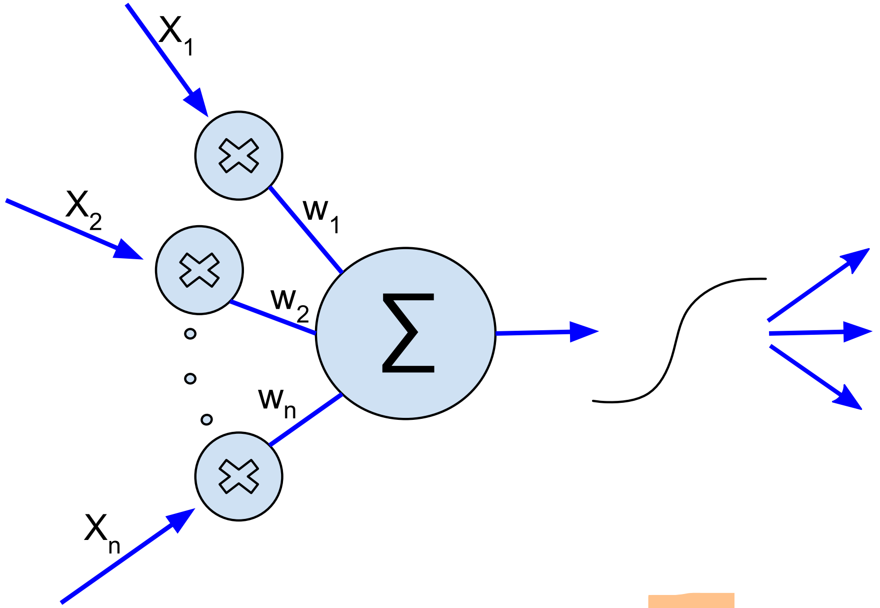
Can we imitate the brain to create AI?

consists of many similar units (neurons) that send signals to each other

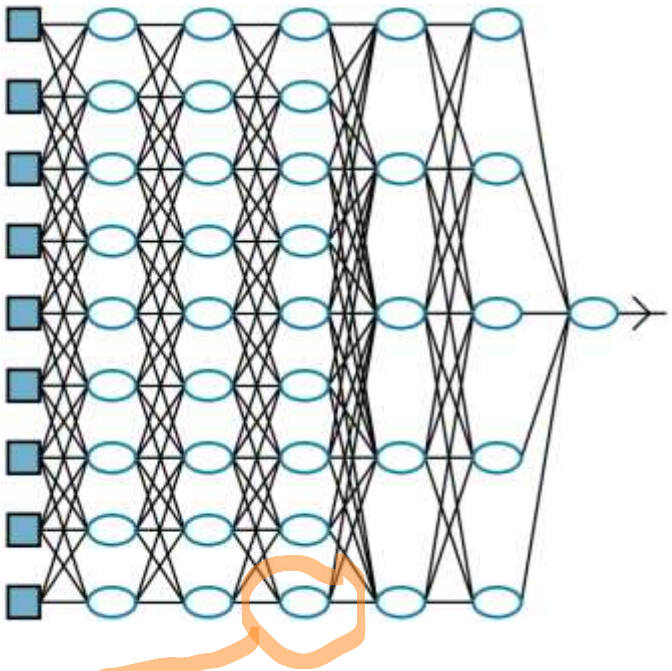


Artificial Neuron

Single unit:



Connect them like this:



Artificial Neuron

Single neuron is a decision making machine

Is it a good time to go pick blueberries?

it's blueberry season (season=1)

the weather is nice (weather=1)

no bear sightings (bear=0)

× **weights ?** = **Yes**

inputs

expected decision

Artificial Neuron

Is it a good time to go pick blueberries?

$$Decision = w_1 \times season + w_2 \times weather + w_3 \times bear$$

Exercise:

Try to find weights $w_1, \ldots, w_3$ to make decisions with this formula

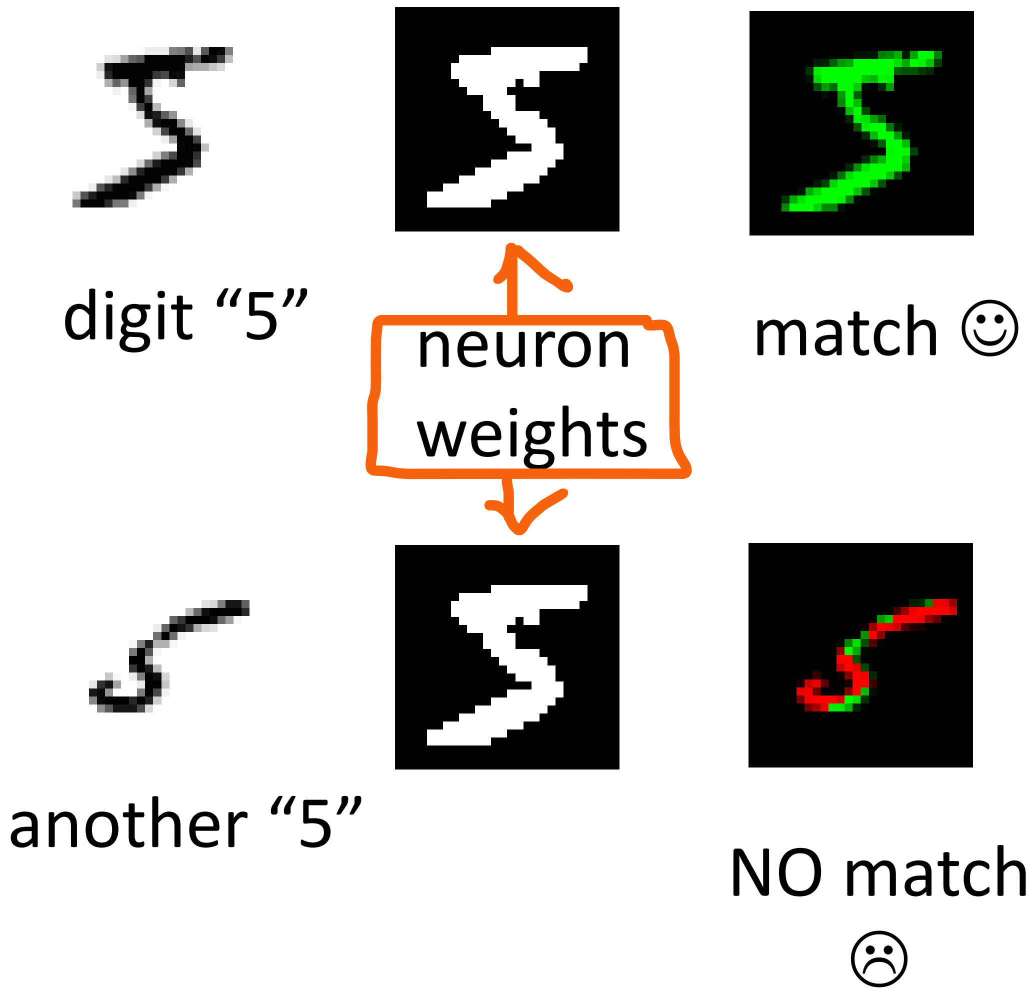
w_1	w_2	w_3

season	weather	bear
1	1	0
1	0	0
1	0	1
0	1	0

Neural Networks

Why do we need multiple layers?

Single neuron is not powerful enough for things like computer vision



Multi-layer Networks

Feature extraction layer

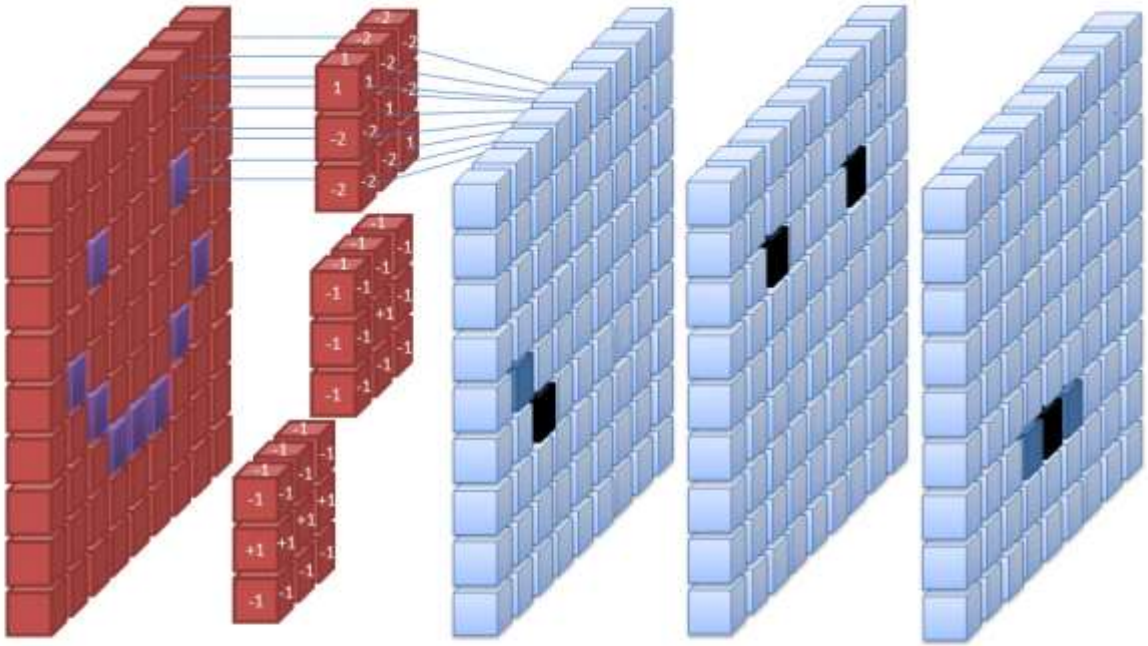


Image: Wikimedia Commons

Completing the Neuron

$$Decision = w_1 \times season + w_2 \times weather + w_3 \times bear$$

w_1	w_2	w_3
2.6	0.5	-1

season	weather	bear	"Decision"
1	1	0	3.1
1	0	0	2.6
1	0	1	1.6
0	1	0	0.5

TODO: discuss: add activation function, then need bias

Training Neural Networks

We’re classifying cats and dogs TODO: simplify

$\hat{y}_k$: output neuron k that says “this is a cat”

y_k : what we really wanted

$x_1 \ldots x_n$	$\hat{y}_k$	y_k	Error
Cat data	0.78	1	0.22
Dog data	0.33	0	-0.33
Cat data 2	0.96	1	0.04

Goal: try to make error close to 0

Training Neural Networks

some explanation based on optimizing the weights

maybe mention things like autograd

Model Size

Cost of training (very roughly)

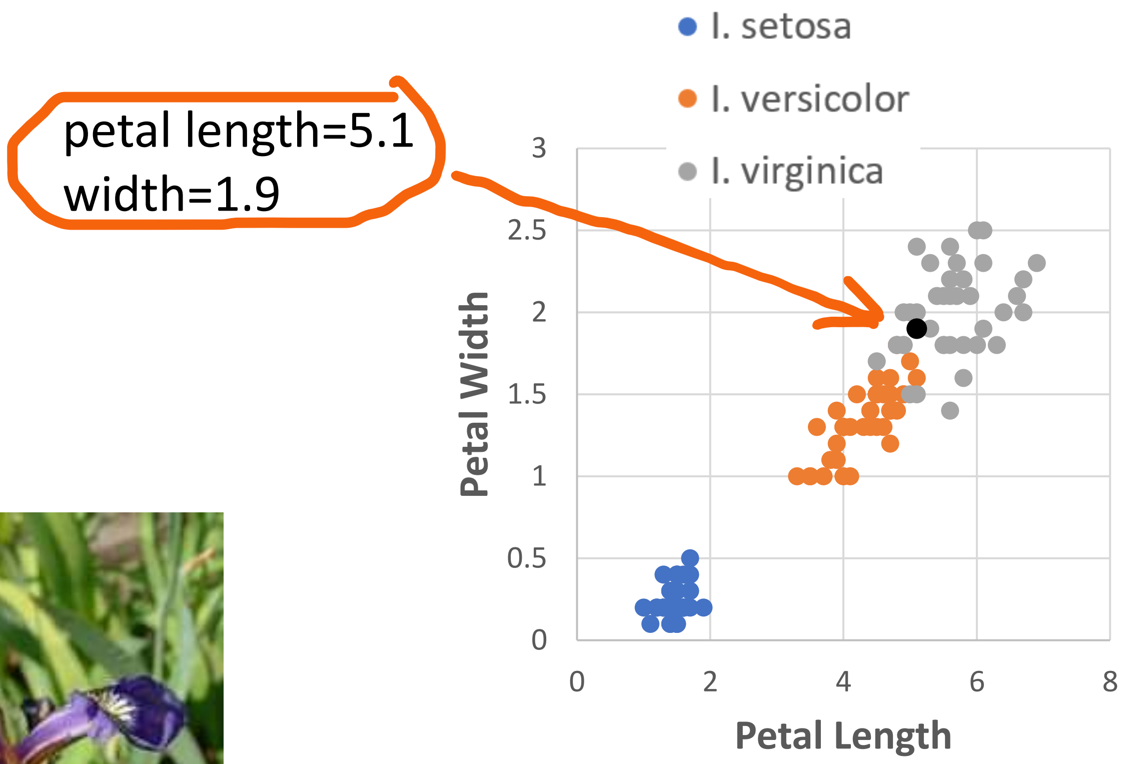
model	application	year	parameters	4090 time*
LeNet-5	handwritten digits	1998	44000	few seconds
AlexNet	image classification	2012	62 million	2 hours
BERT	natural language	2018	340 million	15 days
GPT-4	“general purpose”	2023	1 trillion	10000 years

*- computing time to do the needed FLOPS on a Nvidia RTX 4090

k-Nearest Neighbors

No model needed!

Which species is the black dot?



“The Iris Flower Dataset”

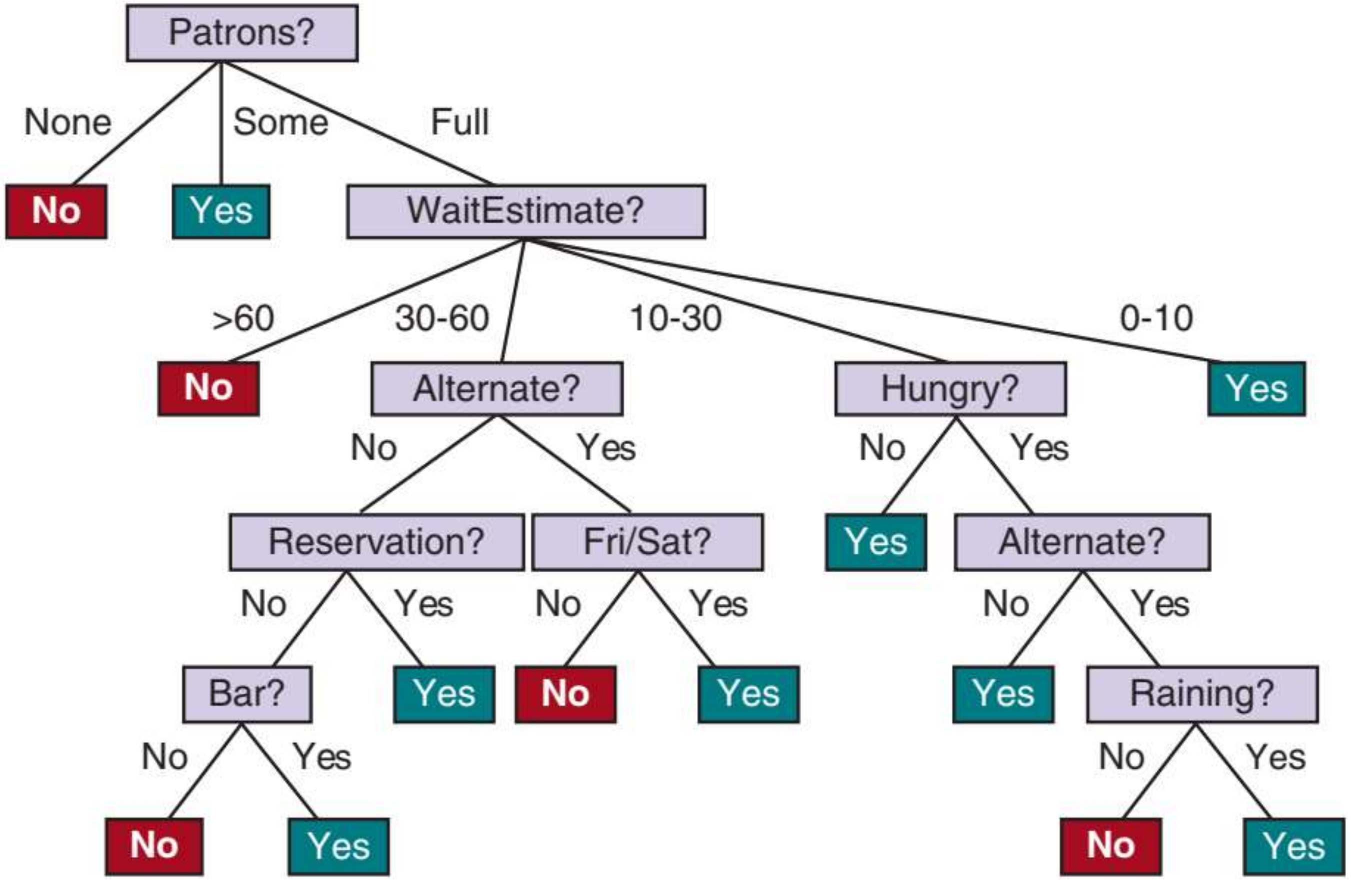
R. Fisher (1936)

Decision Trees

A human deciding

"Should I wait for a table at this restaurant?"

We can learn a tree like this **automatically**



Decision Trees: Learn from Data

Alt: alternative nearby

Res: have reservation

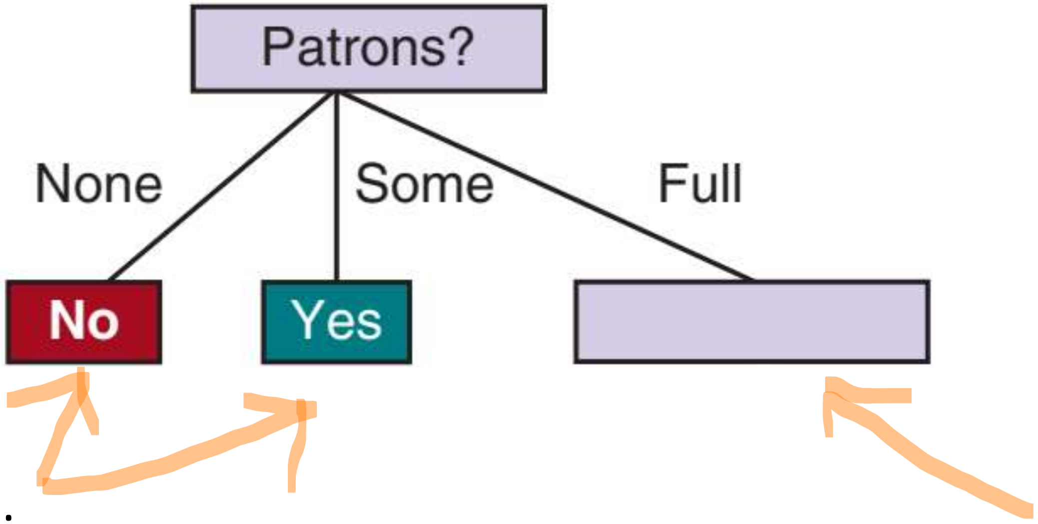
Est: estimated waiting time

Will Wait: the decision to learn

Alt	Bar	Friday	Hungry	Patrons	Price	Rain	Res	Type	Est	Will Wait
Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes
Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No
No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes
Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes
Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No
No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0-10	Yes
No	Yes	No	No	None	$	Yes	No	Burger	0-10	No
No	No	No	Yes	Some	$$	Yes	Yes	Thai	0-10	Yes
No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No
Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10-30	No
No	No	No	No	None	$	No	No	Thai	0-10	No
Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	Yes

Building a Decision Tree

Start somewhere: pick an attribute, look at possible values

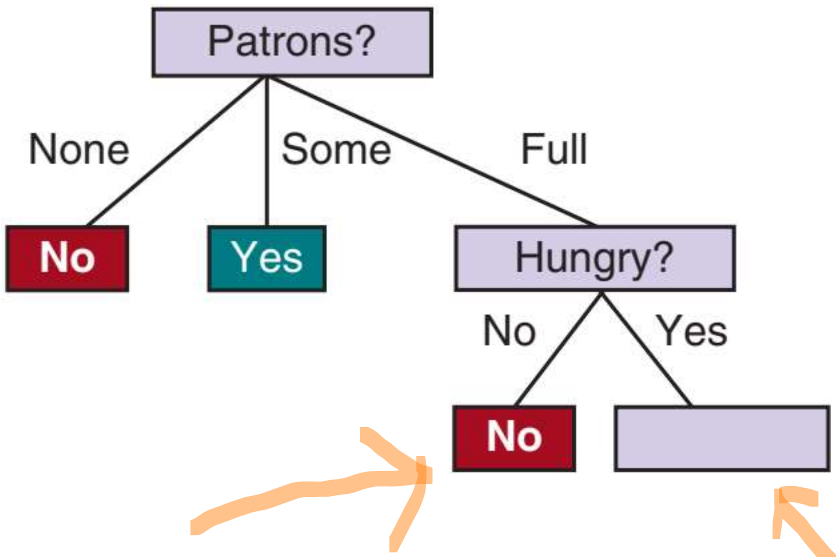


Can decide immediately:
all cases with “None” and “Some”
have the same “WillWait” value

Decision not clear yet –
need to check more attributes

Building a Decision Tree

Next attribute:



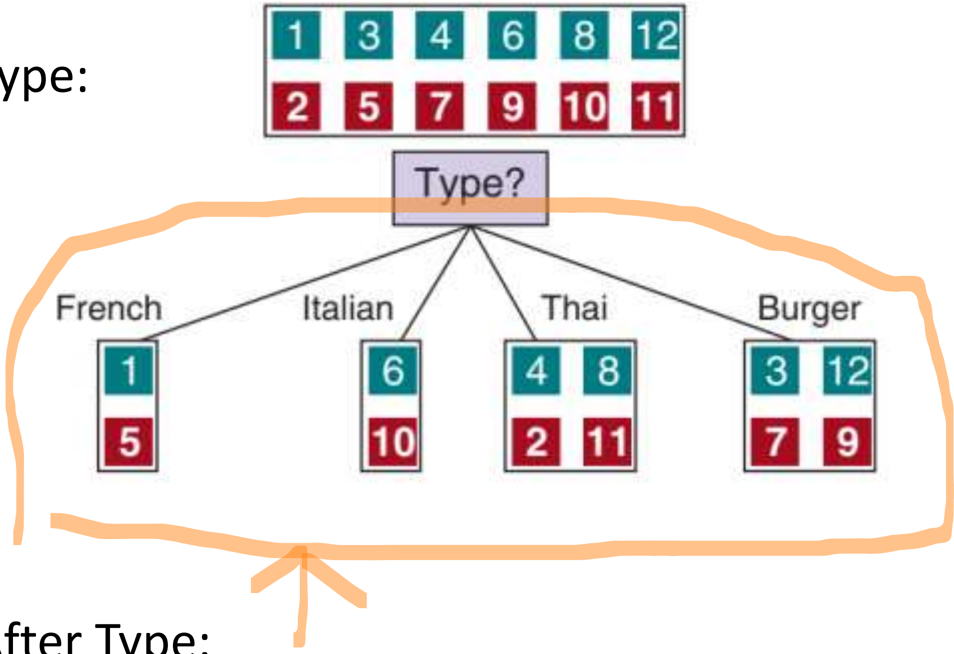
Patrons = "Full" and Hungry = "No": decision possible (WillWait = "No")

Need more attributes to decide

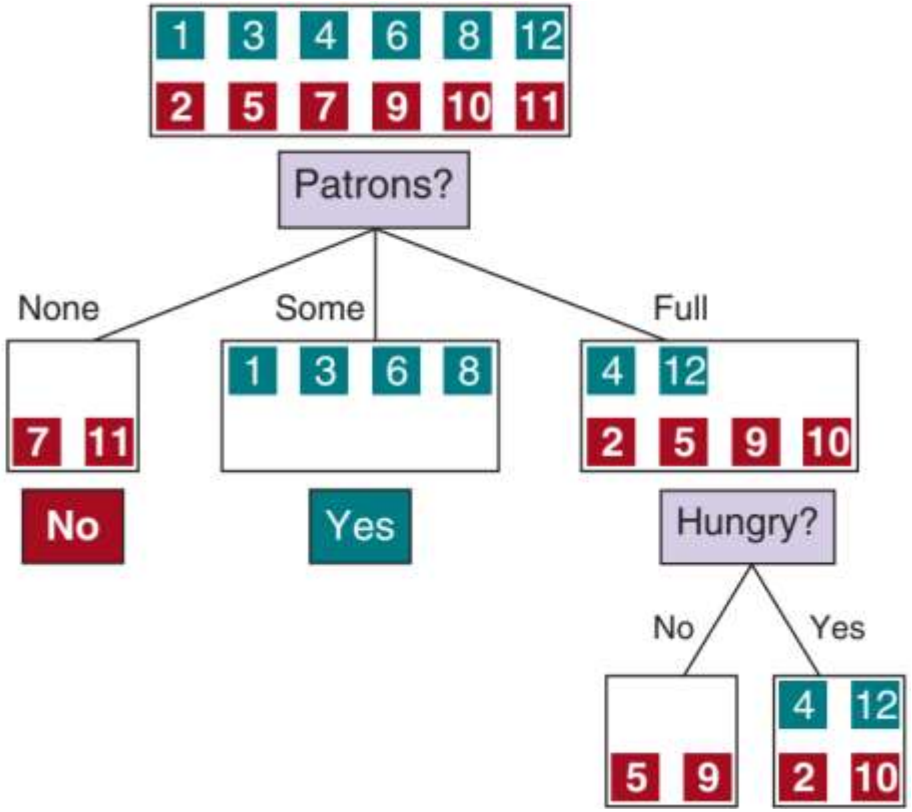
Building a Decision Tree

Prefer attributes that predict the answer better:

Before Type:
50/50

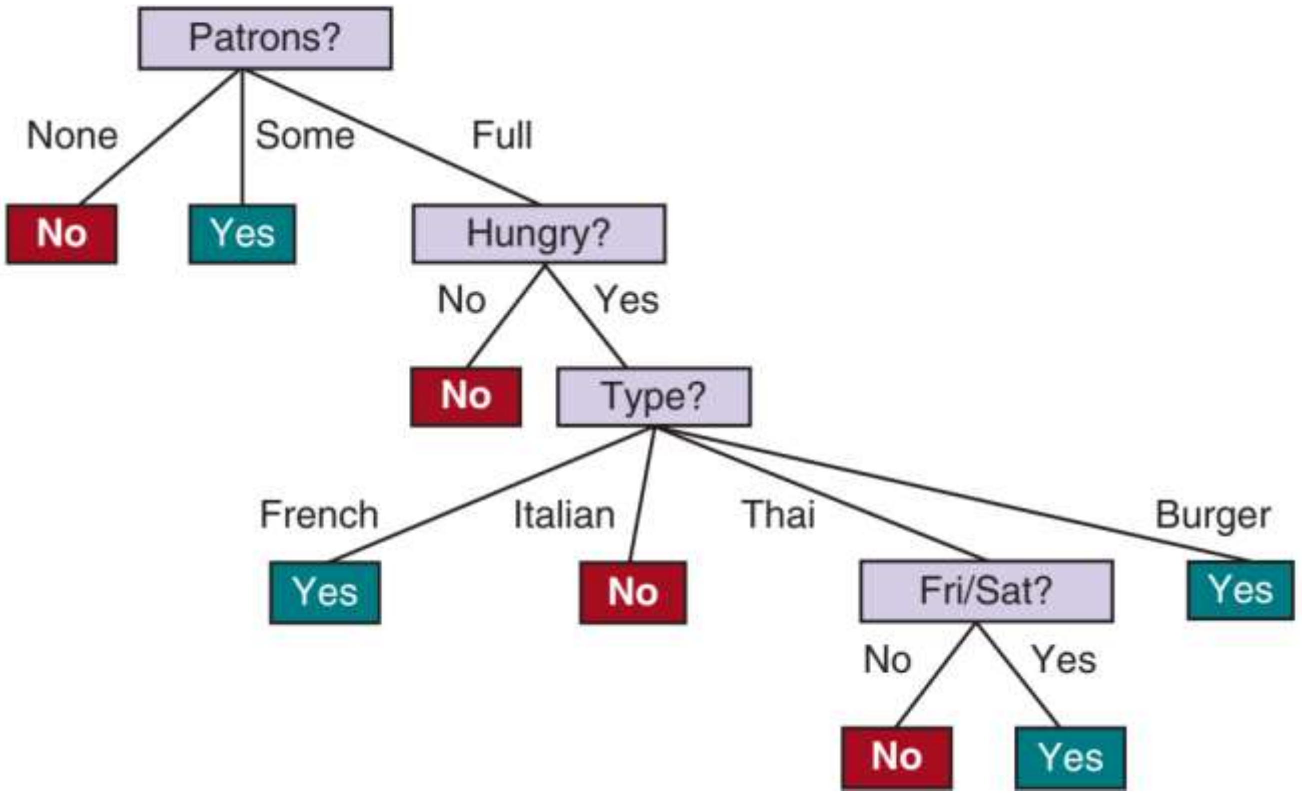


After Type:
each branch still 50/50,
no new information



The Restaurant Example

Automatically learned tree:

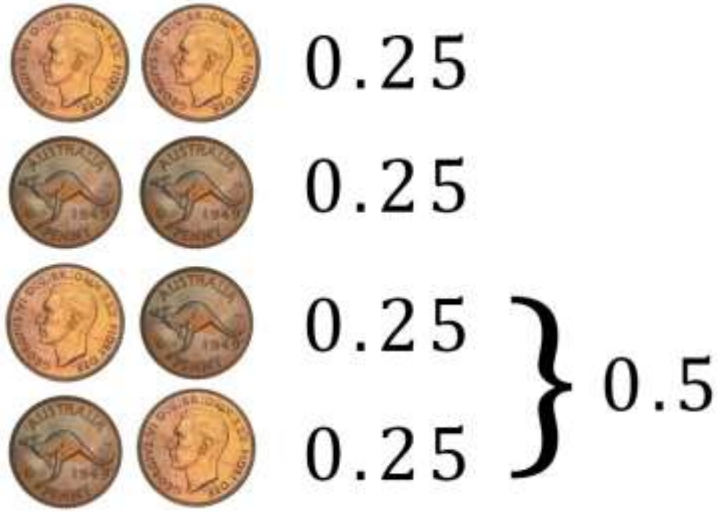


Ensemble learning

What is the probability of throwing two coins and getting “tails” twice?

Probability of three independent models with 90% accuracy, ALL being wrong?

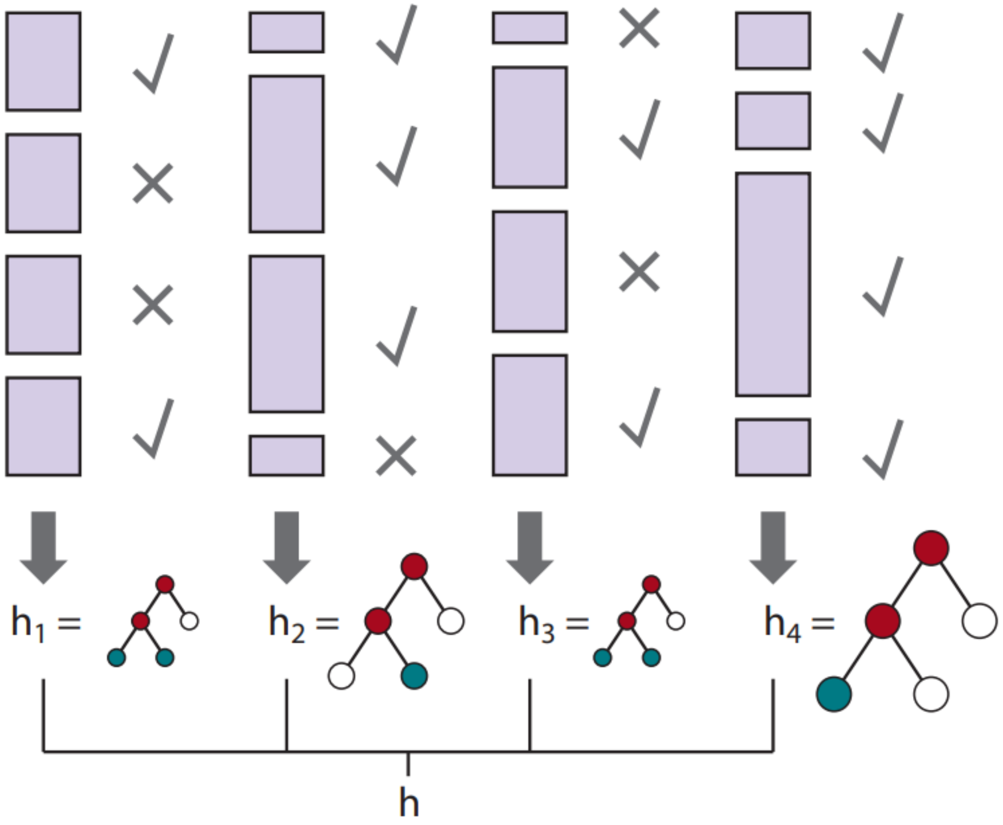
Three independent 90% models,
2 or more (majority) are correct? **0.972**



Ensemble learning

Example: **Boosting**

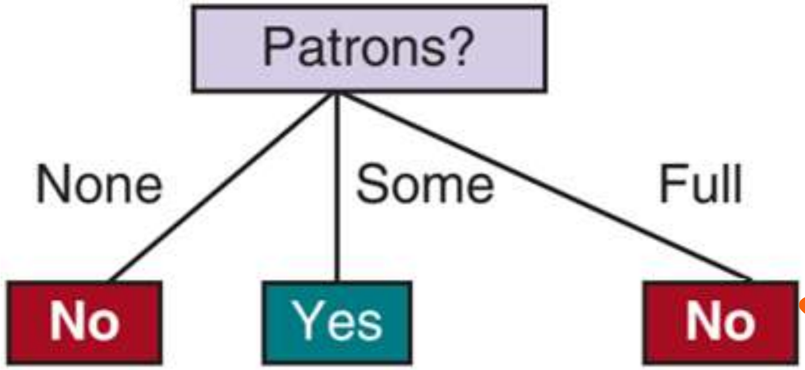
1. Train multiple different models h
2. Decisions by weighted majority vote



Ensemble learning

Individual models can be “dumb”

Decision stump for the restaurant example:



based on
weighted samples

Algorithms: AdaBoost, xgBoost, LightGBM

Machine Learning Basics

- Data should be representative
- Must have enough examples/feedback to train
- Training the model is NOT the goal

Example with these basics: face recognition

Maybe I should include other people?

What am I actually trying to do? And who will draw these boundary boxes?

99.99%
on training set

Awesome, does it work in my application now?

Images: Wikimedia Commons

Evaluating Models

Unbalanced data:
10000000 card transactions, 1000 fraudulent

No joke: a model rewarded by (optimized by) accuracy adopts such strategies

99.99% accuracy predictor

```python
def predict(data):
    return "Not fraud"
```

Fix: measure the right thing: $Recall = \frac{TP}{TP+FN}$

Constant "Not fraud" has $Recall = \frac{0}{0+1000} = 0$

TP – true positive (fraud)
FN – false negative (fraud)
$TP + FN$ – total fraud (detected and undetected)

Evaluating Models

How about this amazing 100% recall predictor?
(unlikely to happen with neural networks)

```python

def predict(data):
    return "Fraud"

```

$Precision = \frac{TP}{TP+FP}$ metric to the rescue (FP – false positive)

Constant "Fraud" predictor: $Precision = \frac{1000}{10000000} = 0.01\%$

Conclusion: use the **right metric(s)** for **your use case**
(Proper training with unbalanced data is a separate issue)