HPE DSI 311
Introduction to Machine Learning

Spring 2023

Instructor: Ioannis Konstantinidis

Overview

Assessing Model Capacity

- The Bias-Variance tradeoff

Supervised classification using deep learning models

- Neural Nets

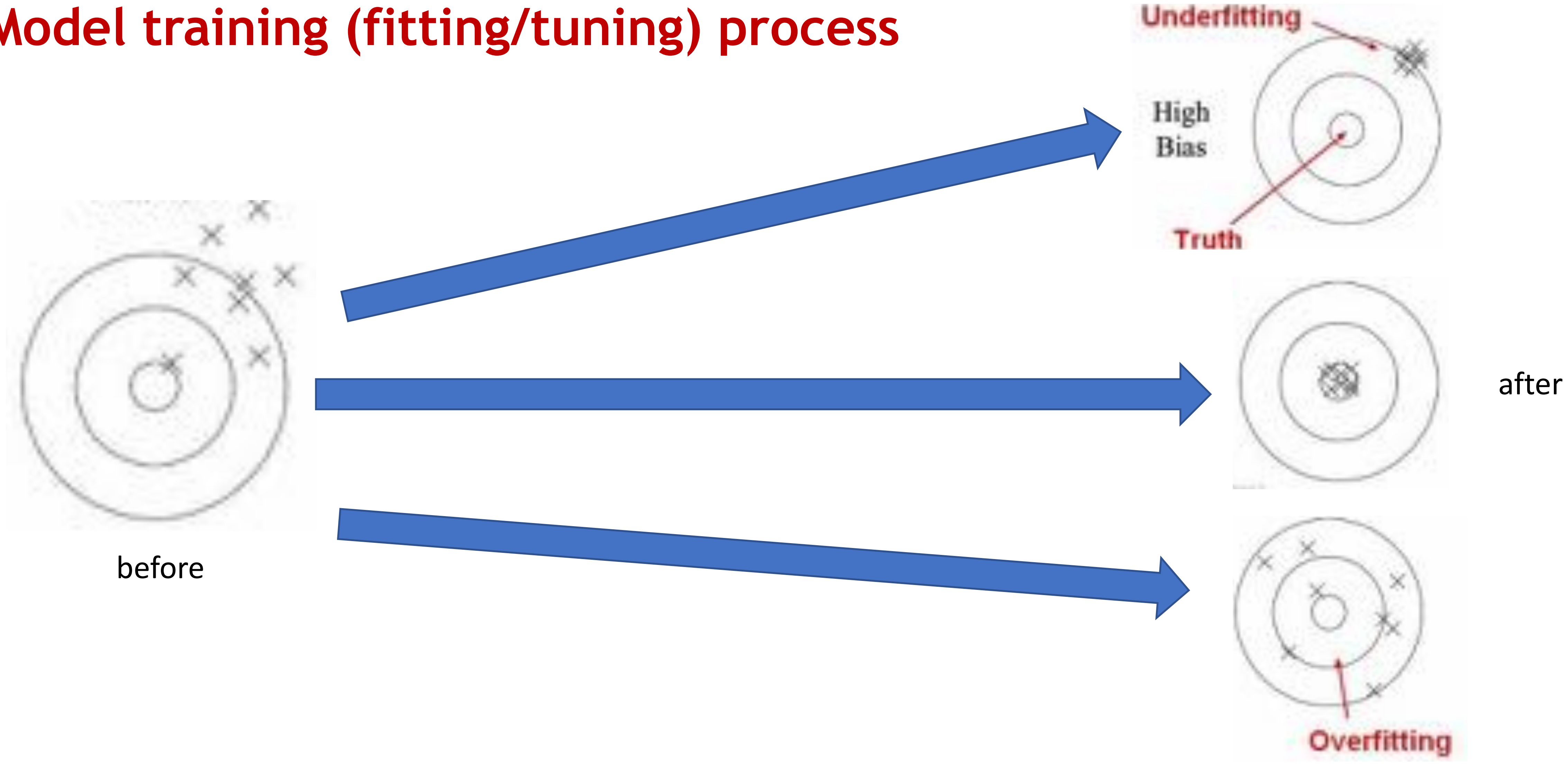
The bias-variance tradeoff

Model tuning: the over/under (fitting)

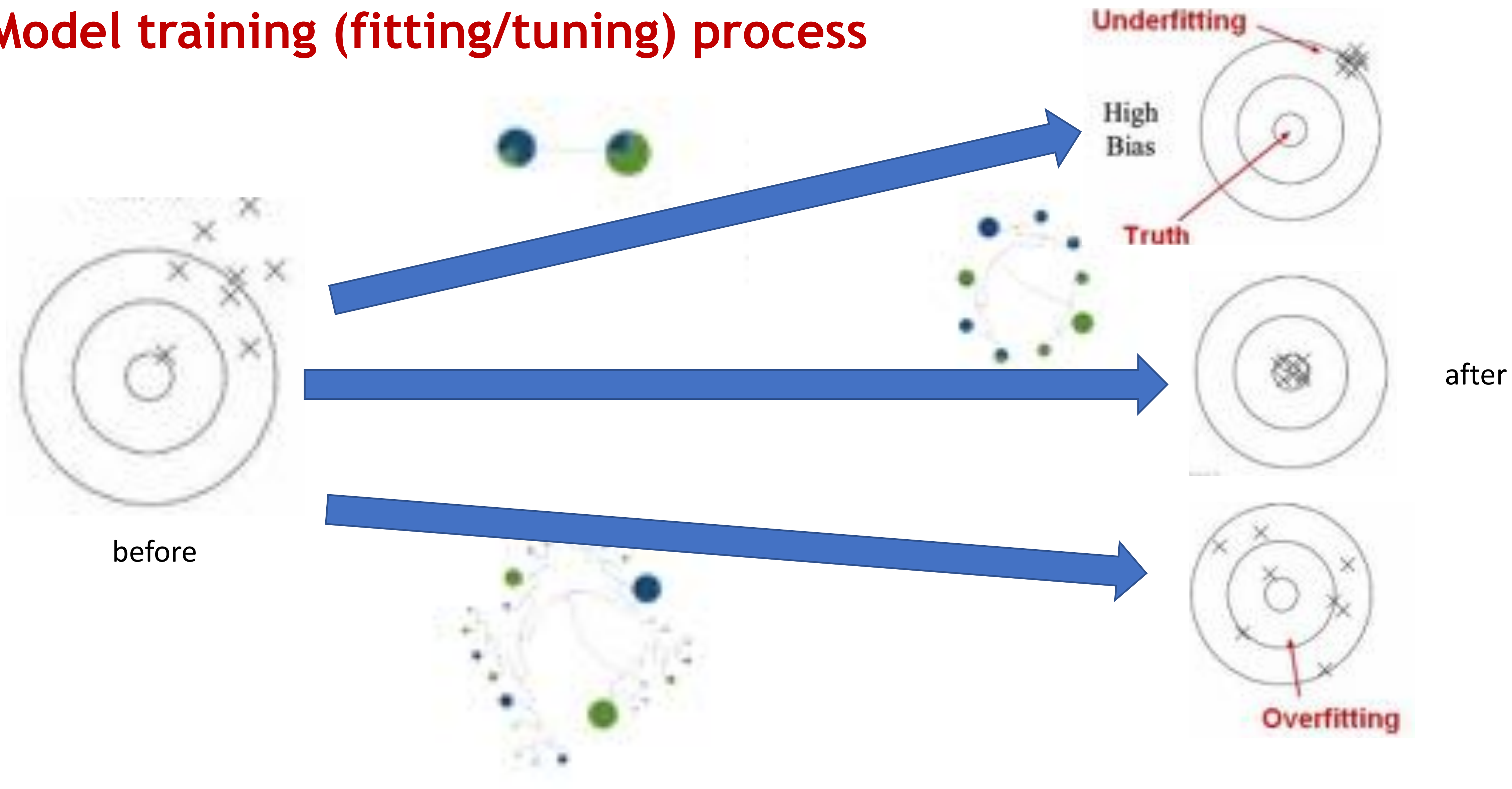
Interactive visualization of the main idea

http://www.r2d3.us/visual-intro-to-machine-learning-part-2/

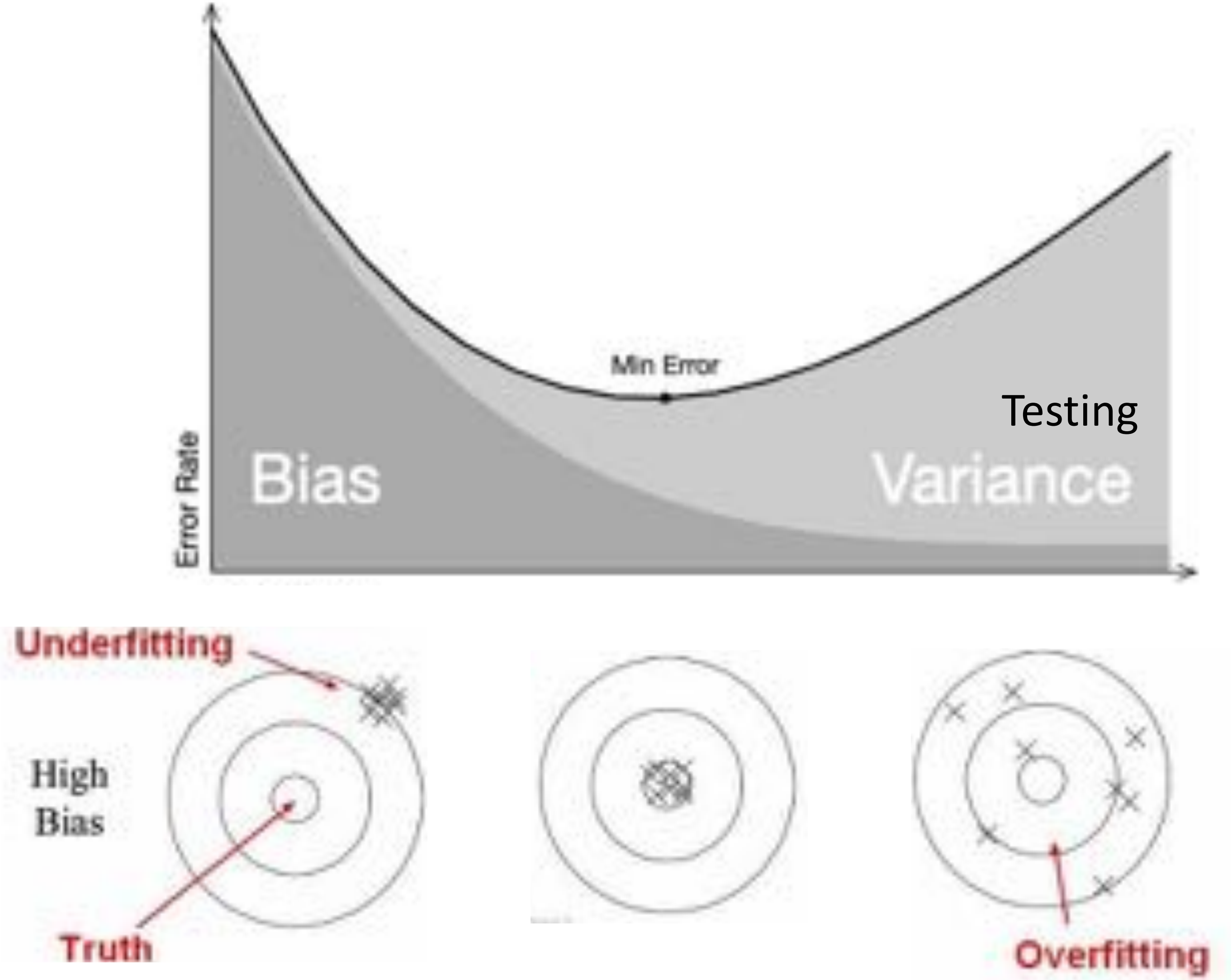
Model training (fitting/tuning) process



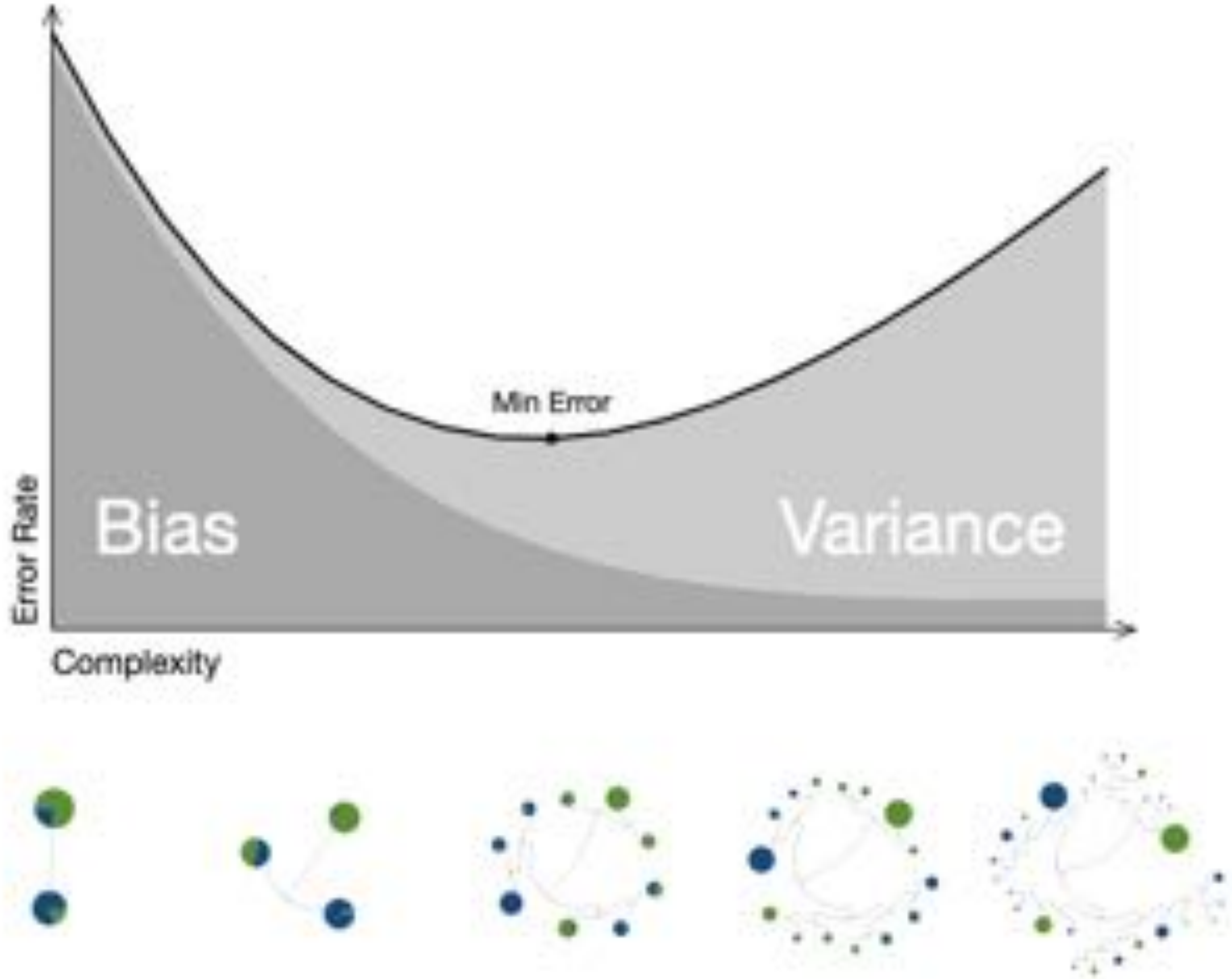
Model training (fitting/tuning) process



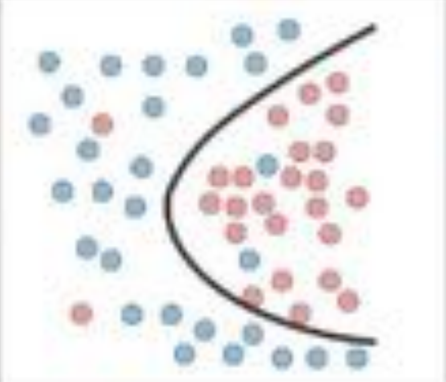
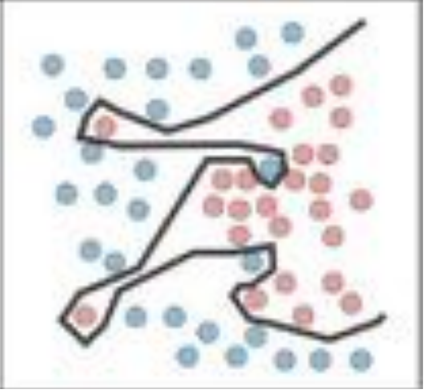
Some people think of it this way in ML



Some people think of it this way in ML



Some people think of it this way in ML

	Underfitting	Just right	Overfitting
Symptoms	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Regression			
Classification			

Hands-on Example

Ready to move on

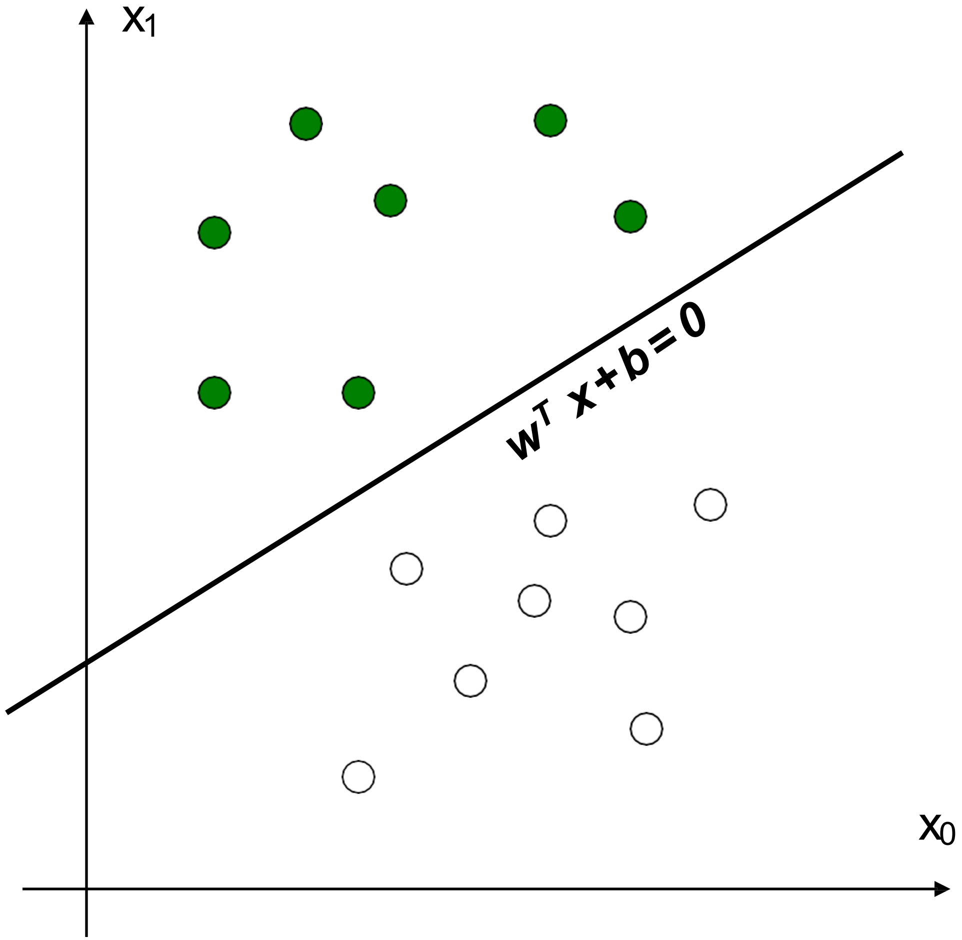
Remember Linear Discriminant Classifiers?

Linear Discriminant Classifier

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b =$$

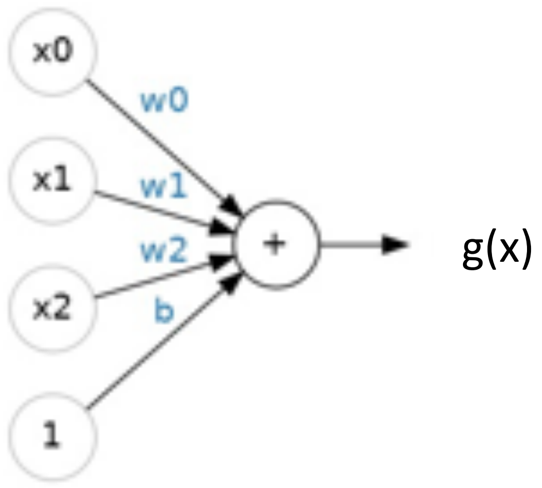
$$= \sum_{i \in SV} w_i \, x_i + b$$

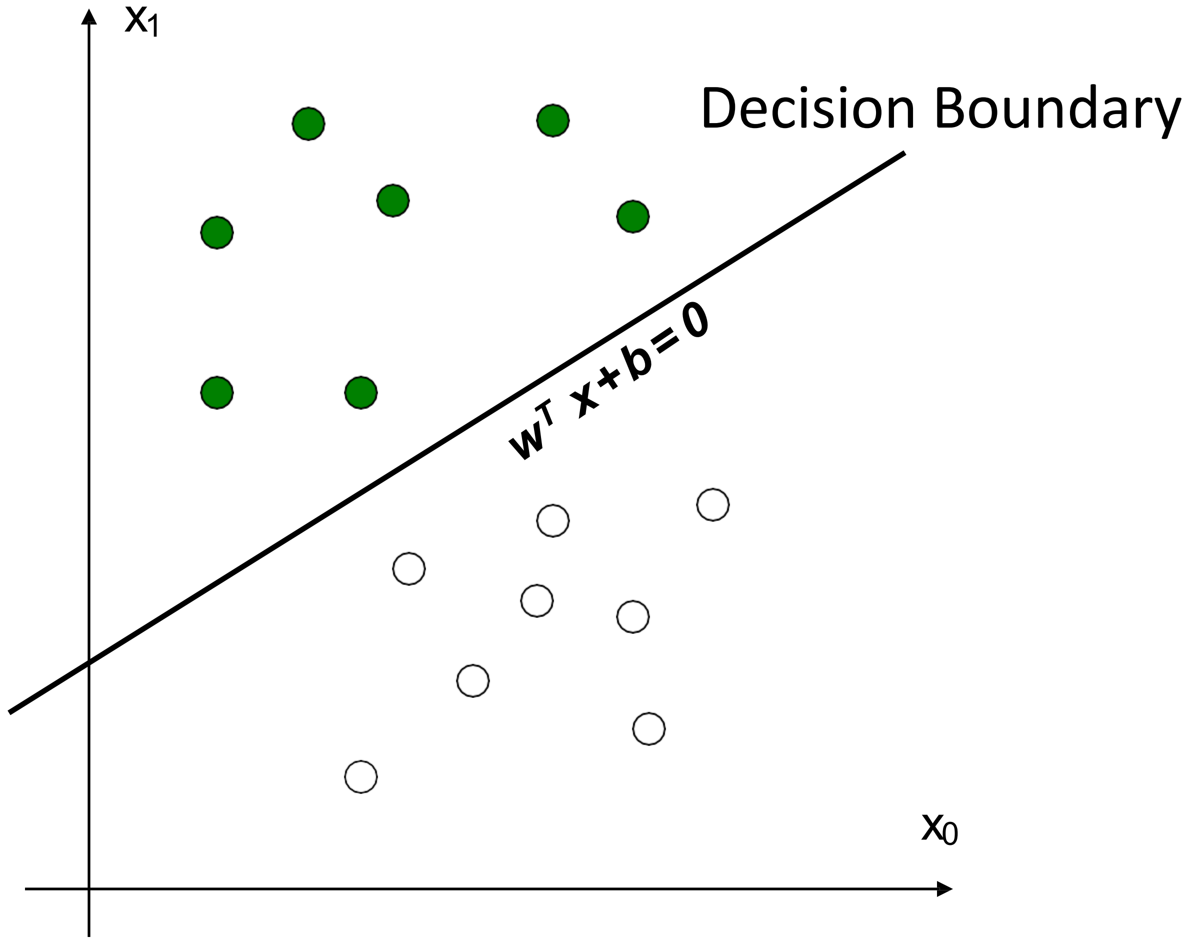
Decision function = sign(x)



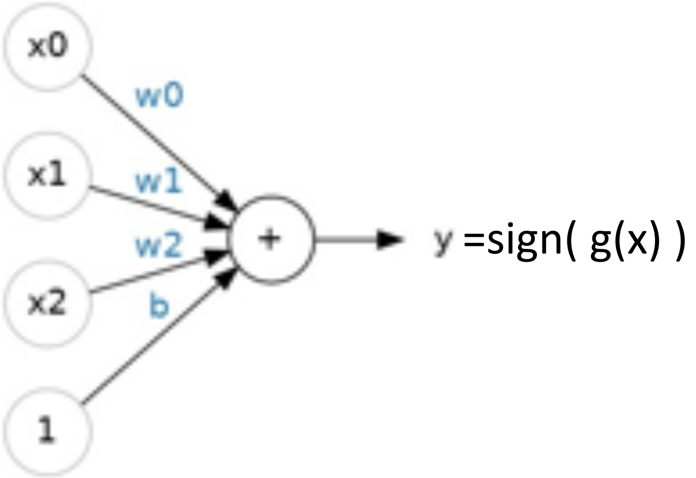
Linear units

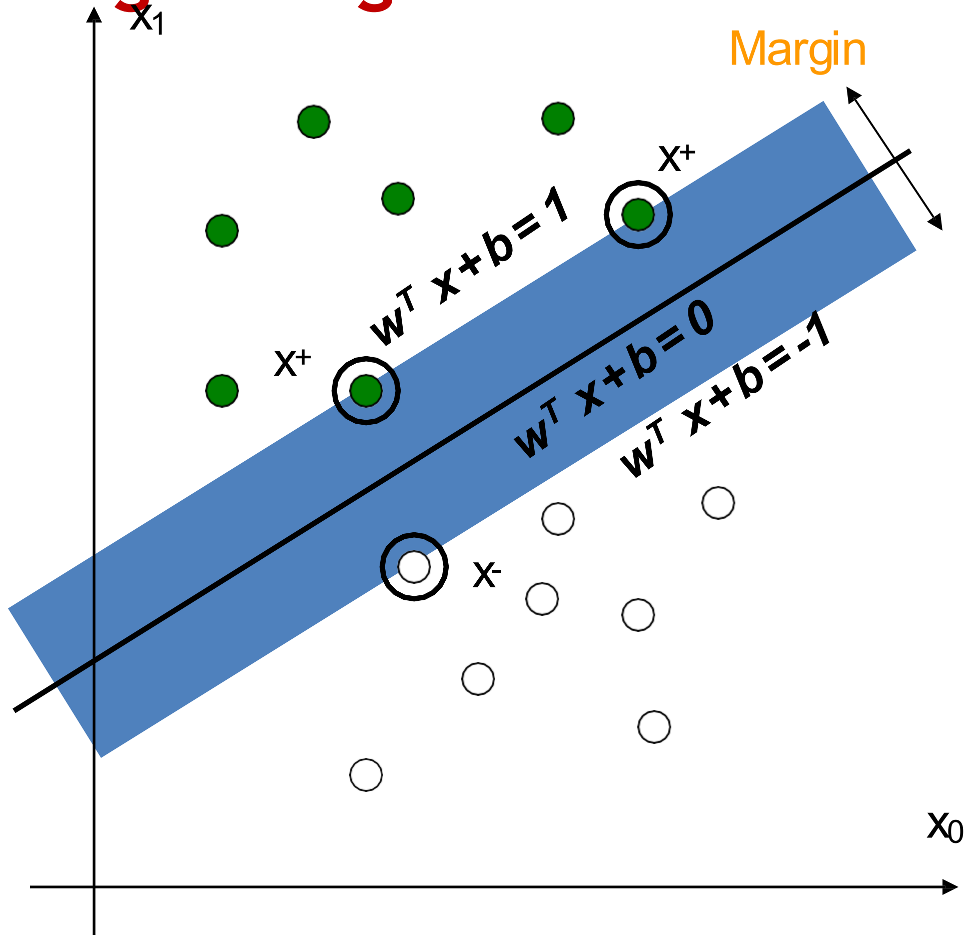
A linear unit:





SVM: linear unit with largest margin + sign





Speed vs. optimality

A 1080p digital image (most common screen size) comprises

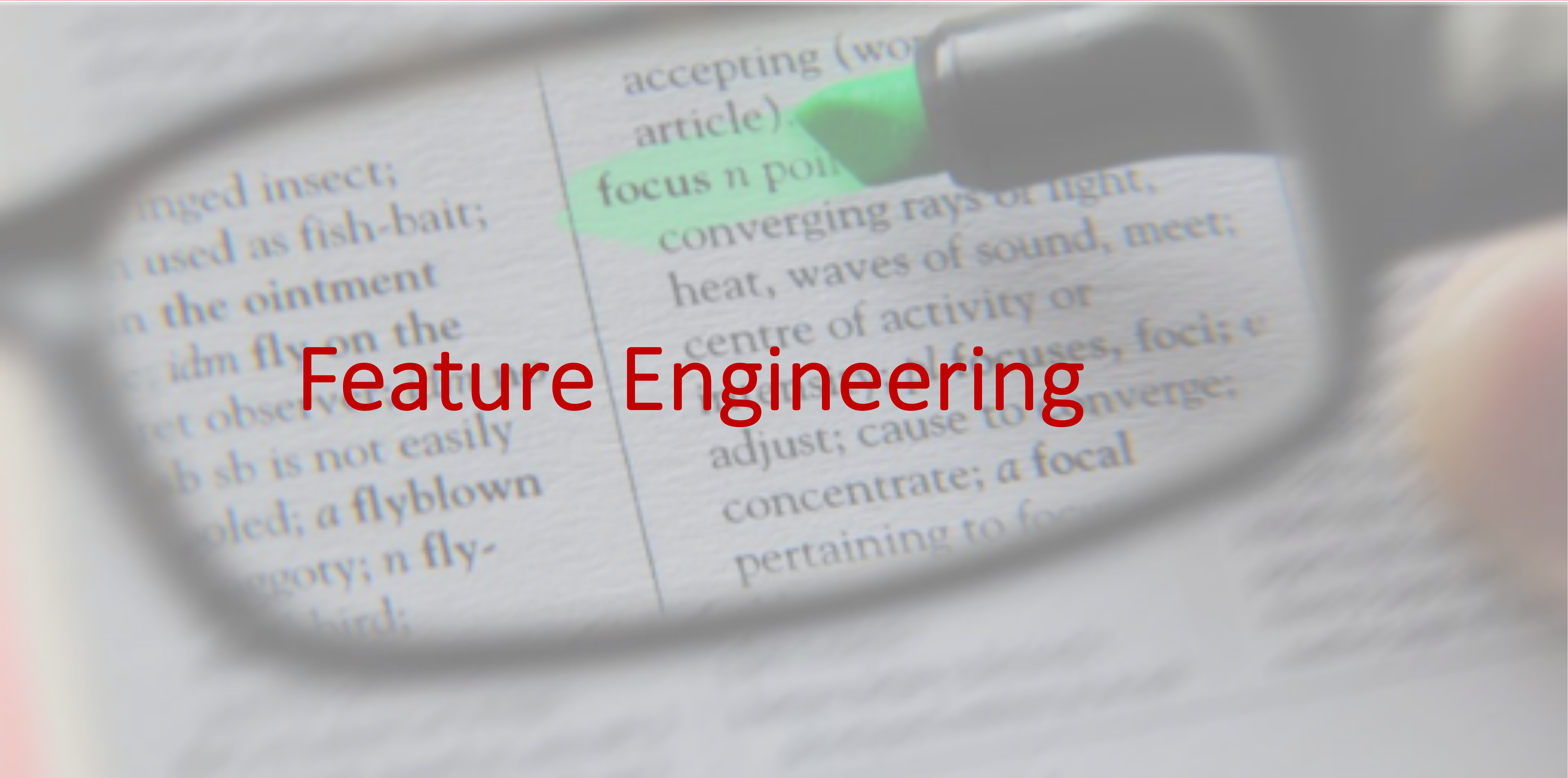
- 1920 x 1080 pixels (1080 lines of vertical resolution) and
- three color channels (RGB)

A total of more that 6 million variables!

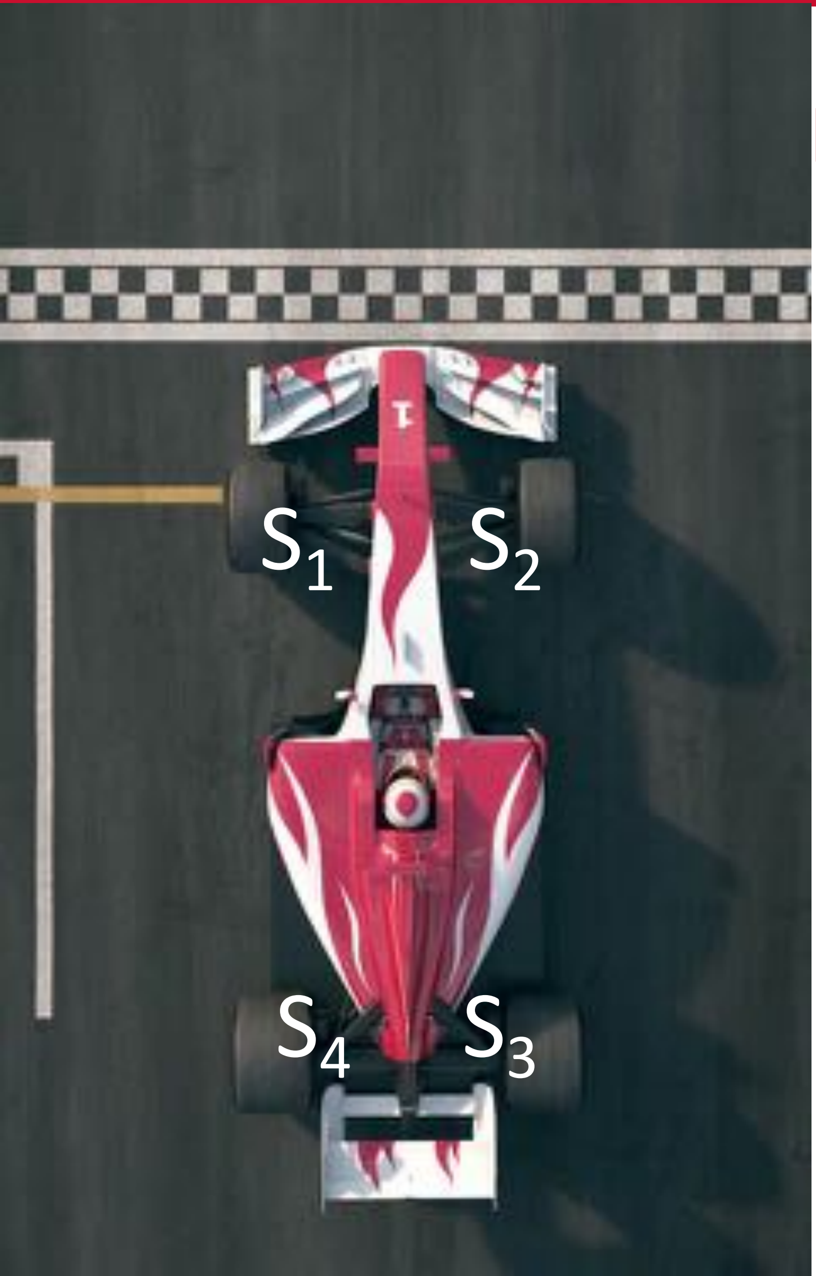
➢unlikely the points will line up nicely for linear class separation to work; also,

➢computing the optimal margin takes a lot of effort (quadratic programming)

Need feature extraction / dimension reduction



Feature Engineering



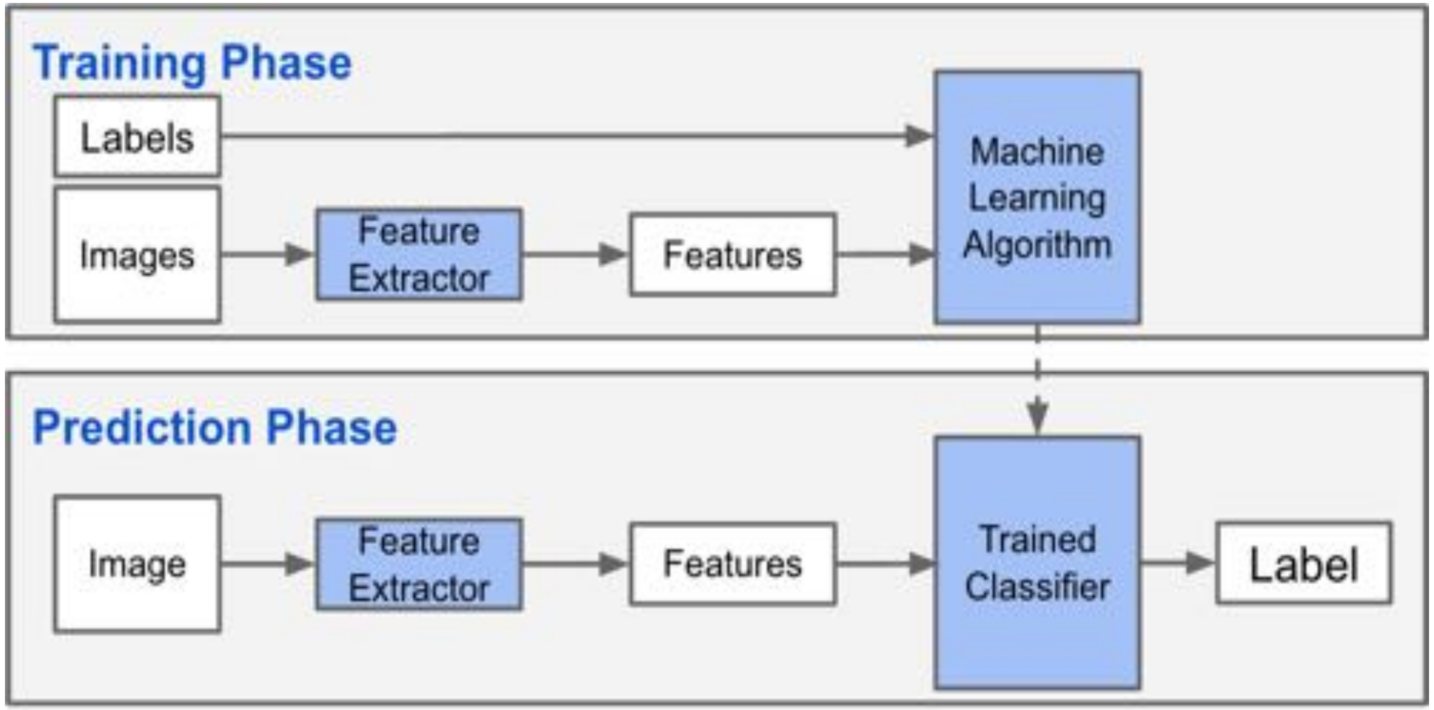
Features: domain knowledge

RAW DATA
Four sensors measuring rotation speed (spin) at each wheel: S_1, S_2, S_3, S_4

NEW FEATURES
$$T_1 = \{ (\frac{S_2 + S_3 + S_4}{3}) - S_1\} / 2 = -\frac{1}{2} S_1 + \frac{1}{6} S_2 + \frac{1}{6} S_3 + \frac{1}{6} S_4$$

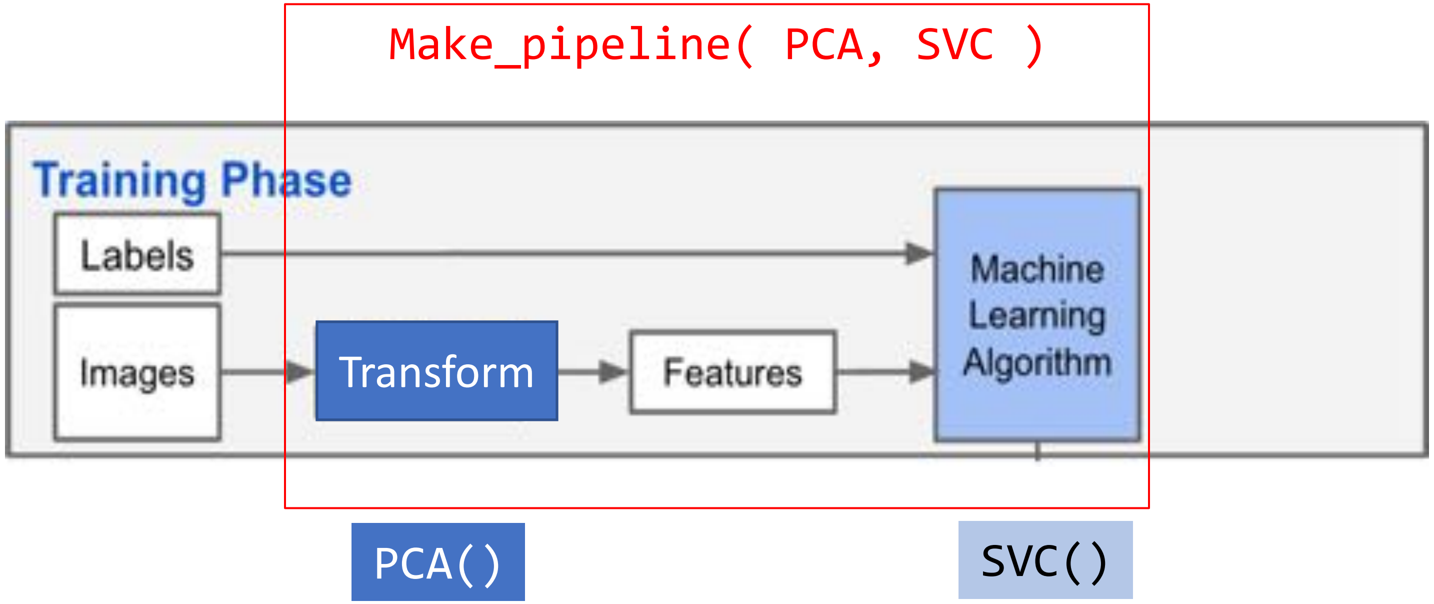
EXPERT KNOWLEDGE
If a feature starts to veer away from zero, then a tire is spinning faster than the others (possible flat)

Feature extractors help unscramble the features from the raw data, and prioritize features for selection



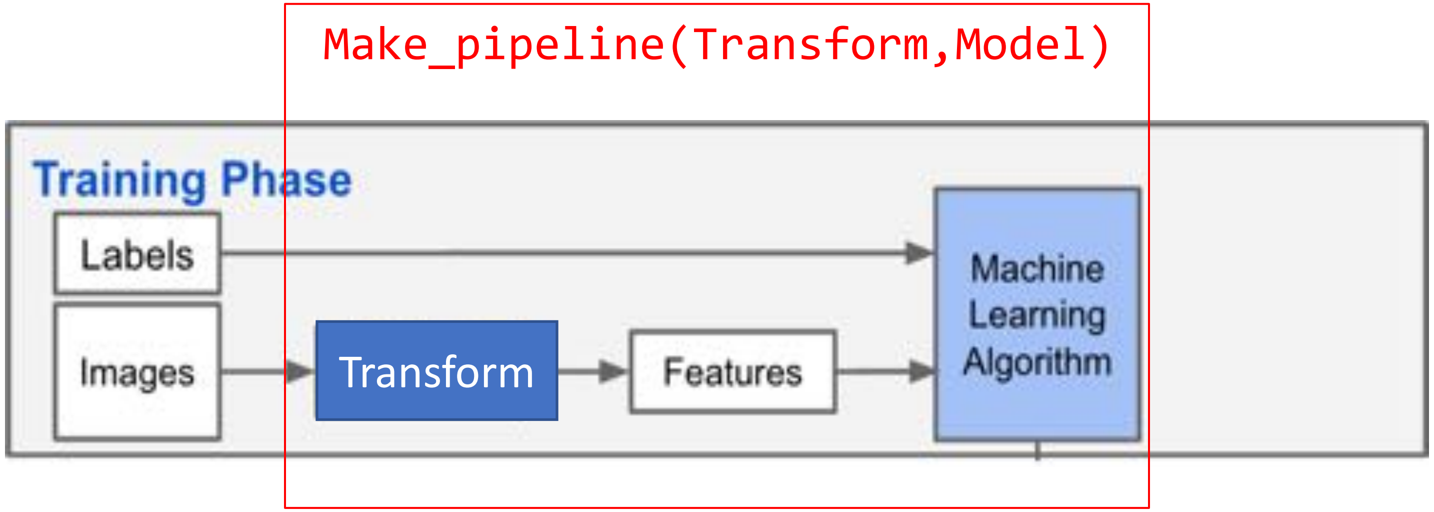
Machine Learning Phases

Feature engineering example: PCA



PCA is most commonly available data transform, because it is the most generic

Feature engineering: a lot of possibilities

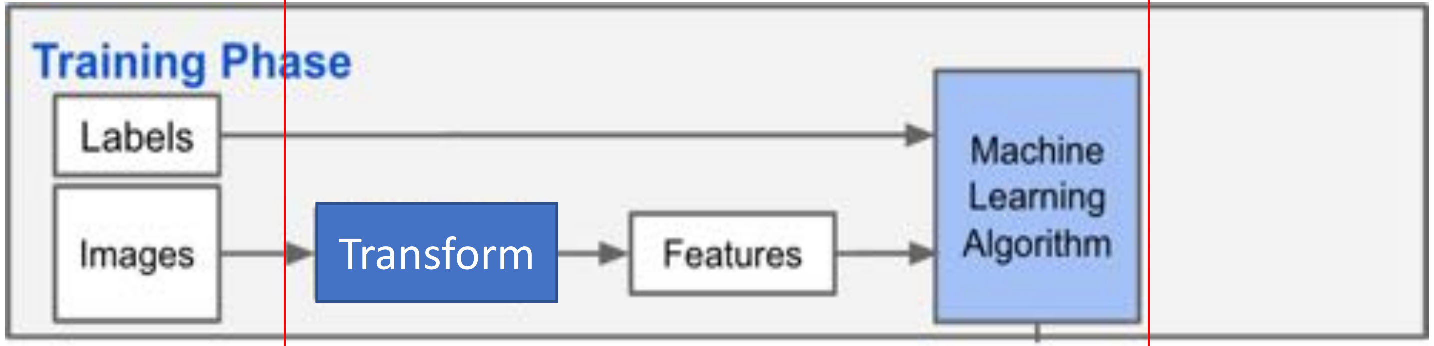


PCA is most commonly available data transform because it is the most generic

There are many other choices

Feature engineering: a lot of possibilities

```
Make_pipeline(Transform,Model)
```

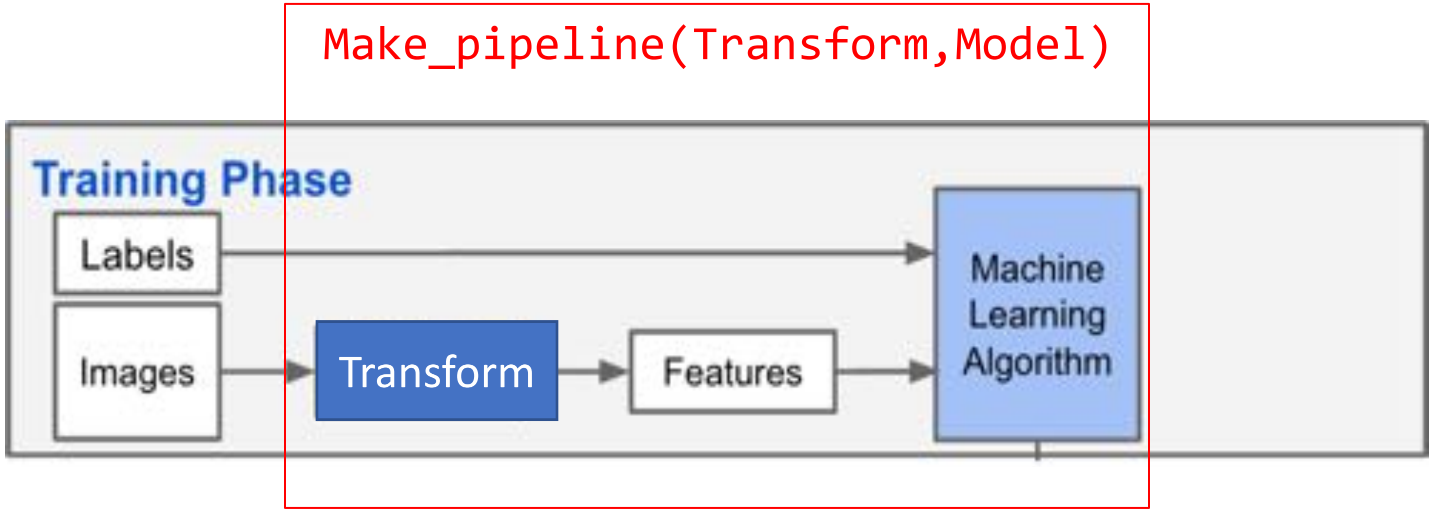


PCA is most commonly available data transform because it is the most generic

There are many other choices:

- Fourier Transform: extract frequencies from wave signals

Feature engineering: a lot of possibilities

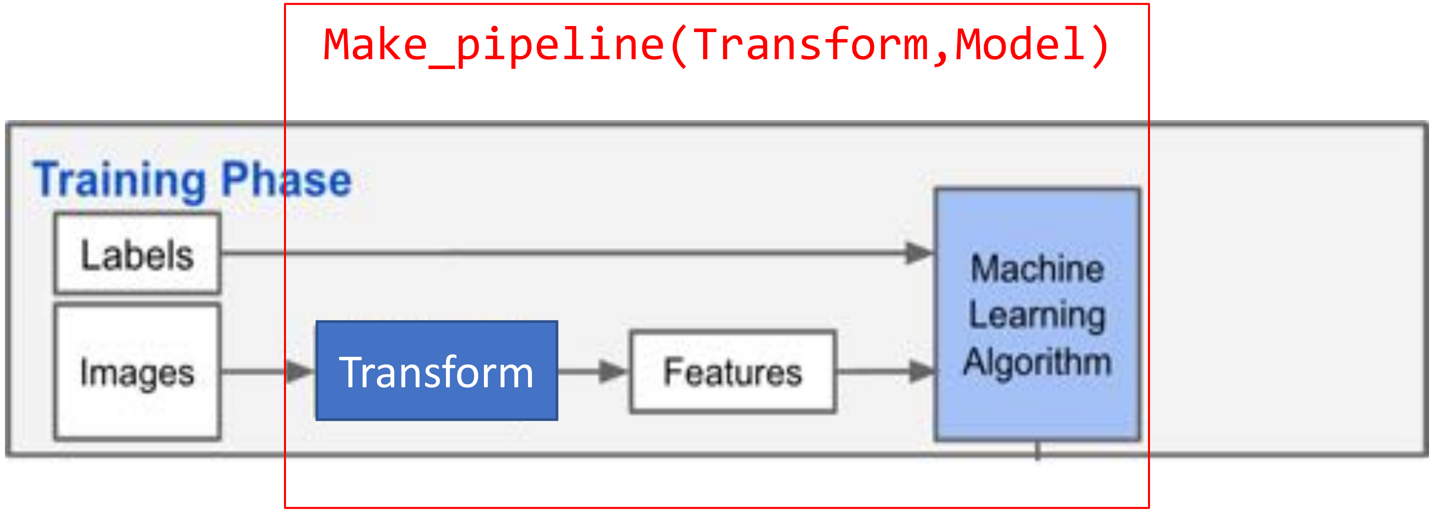


PCA is most commonly available data transform because it is the most generic

There are many other choices:

- Fourier Transform: extract frequencies from wave signals
- Wavelet Transform: extract levels of detail from images

Feature engineering: a lot of possibilities

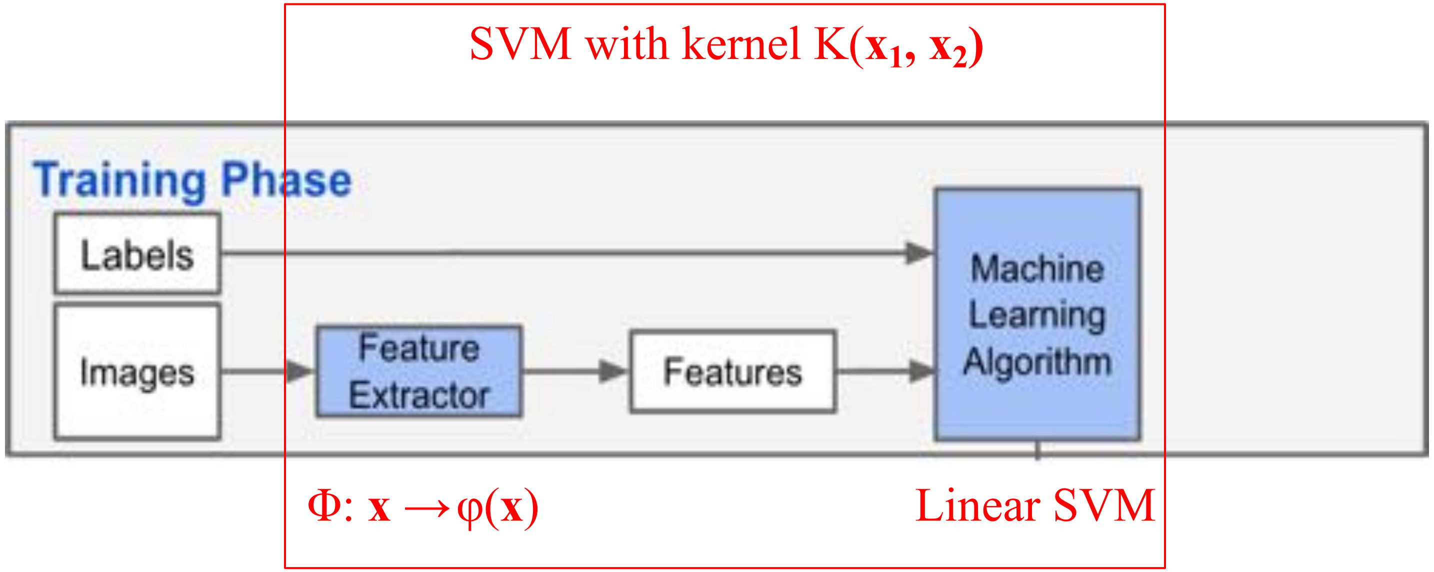


PCA is most commonly available data transform because it is the most generic

There are many other choices:

- Fourier Transform: extract frequencies from wave signals
- Wavelet Transform: extract levels of detail from images
- Kernel Trick!

The kernel trick masks a data transform



Think of

```
    SVC( kernel=‘rbf’ )
```

as being the same as

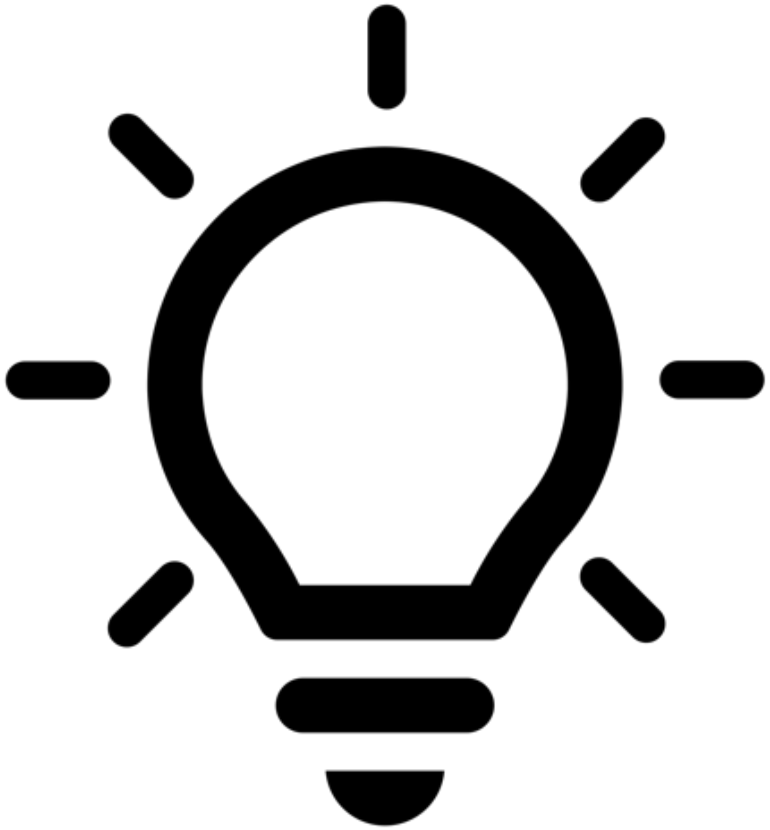
```
    make_pipeline( rbfTransform, SVC )
```

Feature engineering: drawbacks

- Feature engineering is difficult, time-consuming, and requires domain expertise.
- It is in the spirit of symbolic AI, instead of the modern connectionist paradigm

Can we try something different?

First Idea: Ensemble SVC

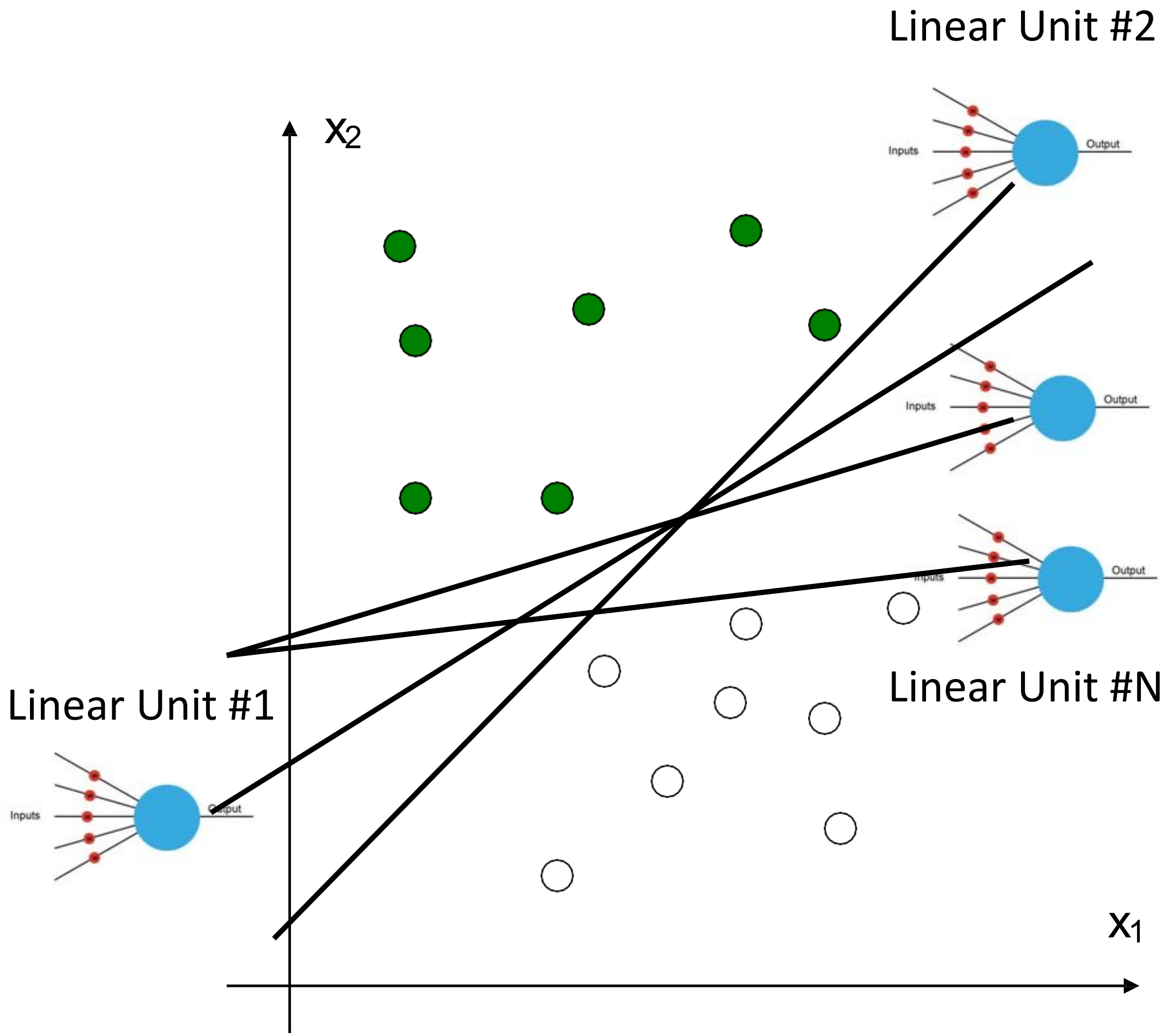


Aggregating activated linear units

Linear units

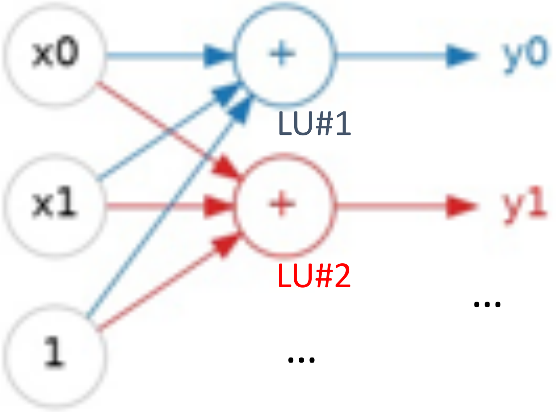
- Different hyperplanes correspond to different linear units

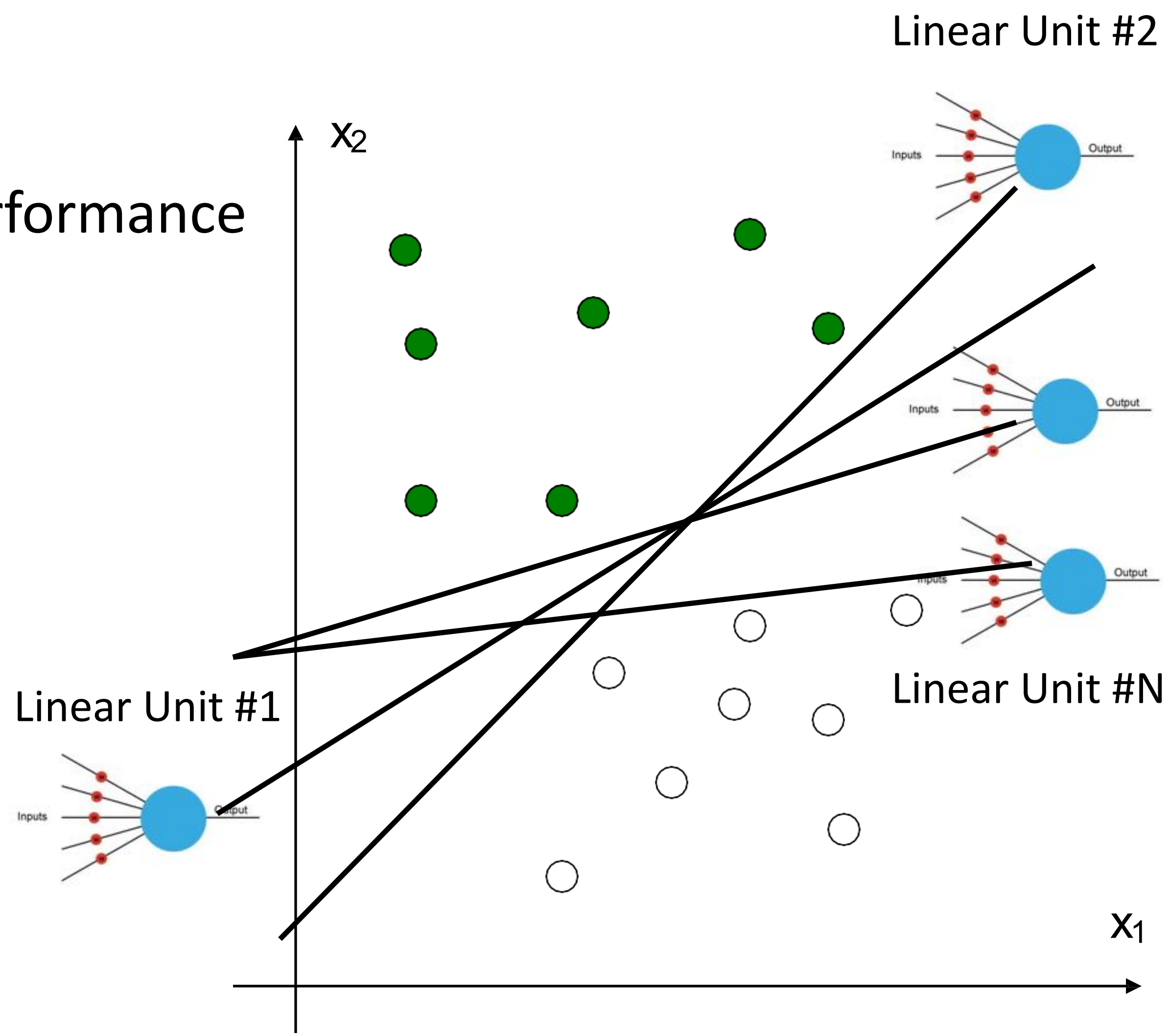
- They all classify the training set correctly, but are slightly suboptimal



Linear units

Aggregated as a group, their performance can be close to the SVM



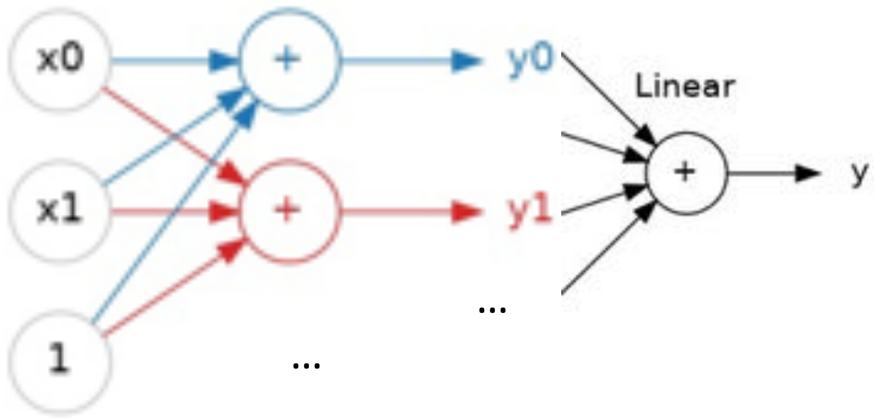


A simple ensemble

Decision function: Add the outcome of each unit to aggregate

Input Layer

Output Layer

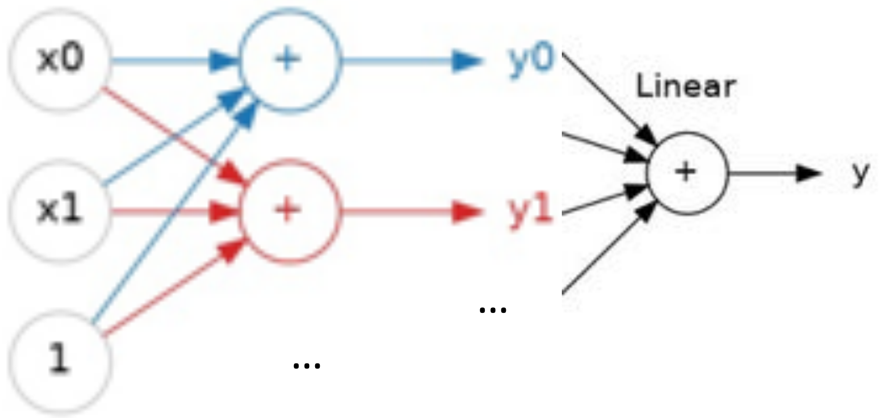


A simple ensemble

Decision function: Add the outcome of each unit to aggregate

Input Layer

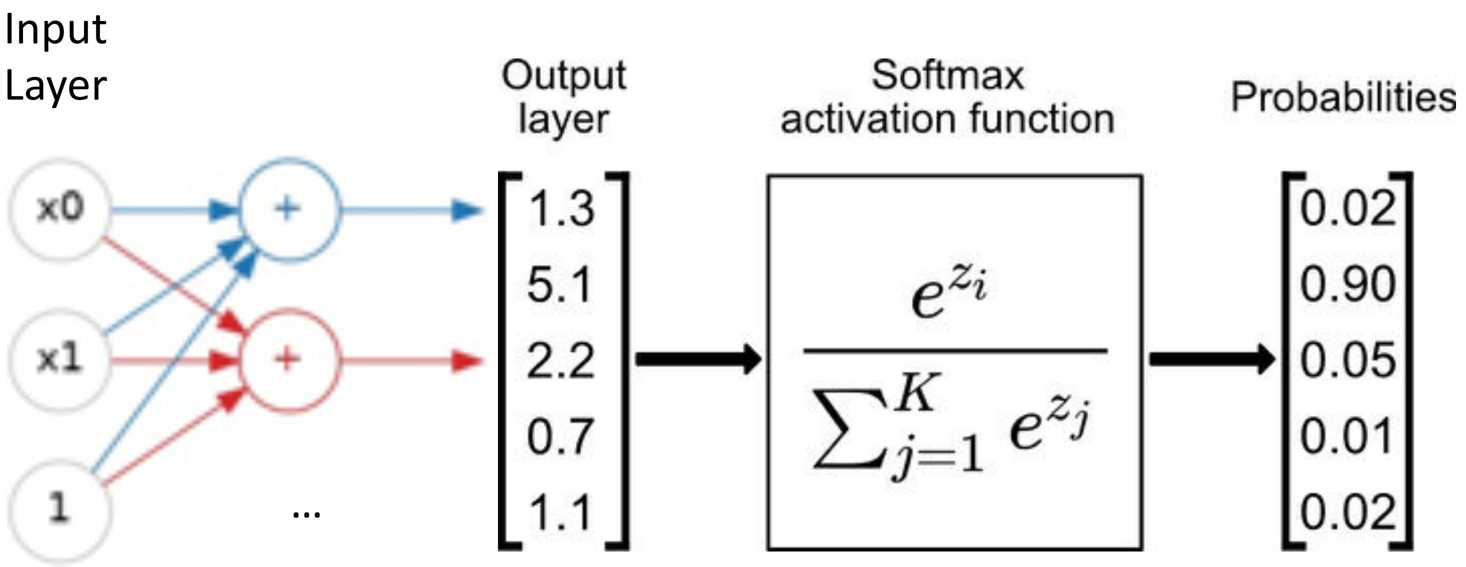
Output Layer



The final (output) layer is also a linear unit. That makes this network appropriate to a regression task, where we are trying to predict some arbitrary numeric value.

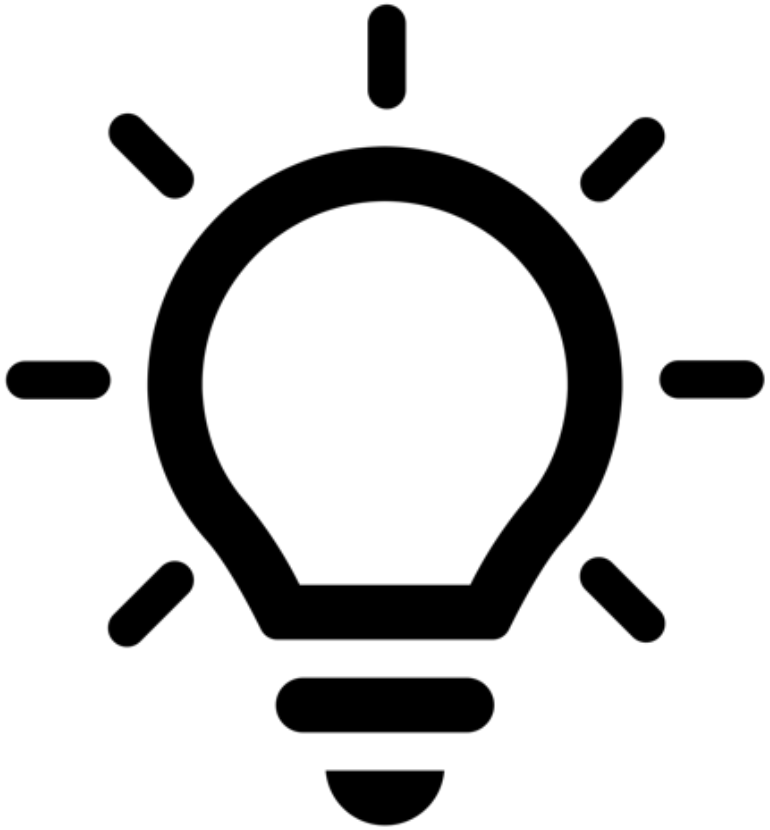
A simple ensemble

Decision function: Other tasks might require a different decision function on the output

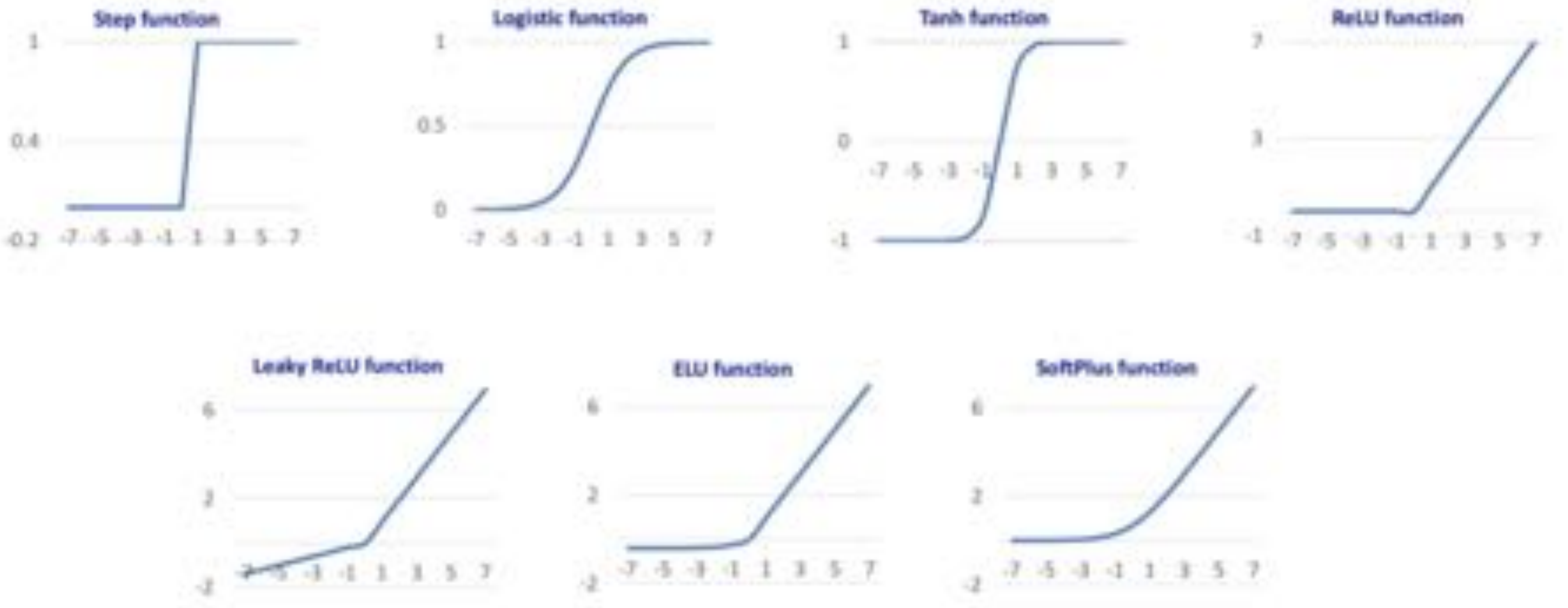


SoftMax is the most common for classification

Second Idea: sign, logit, … ?



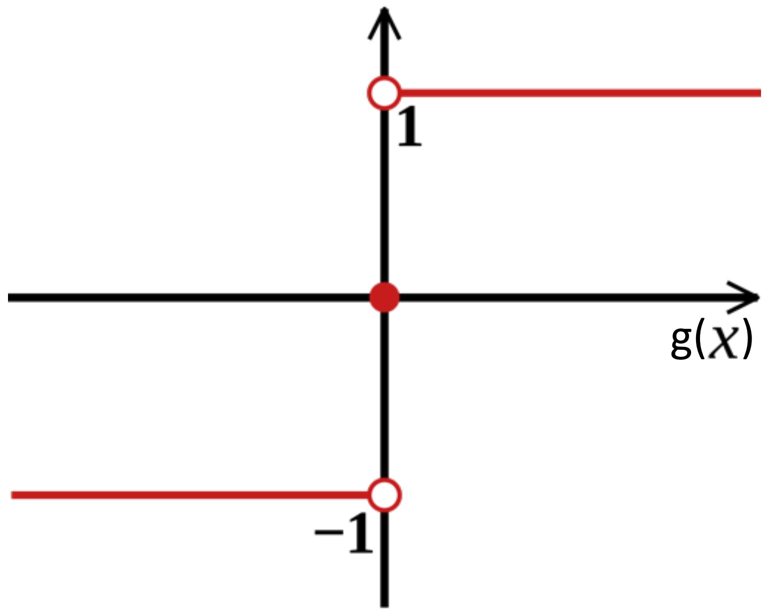
Many choices for the activation function



Logistic Regression: activate using the logistic function

$$\text{Probability(y)} = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

SVC: activate using the sign function

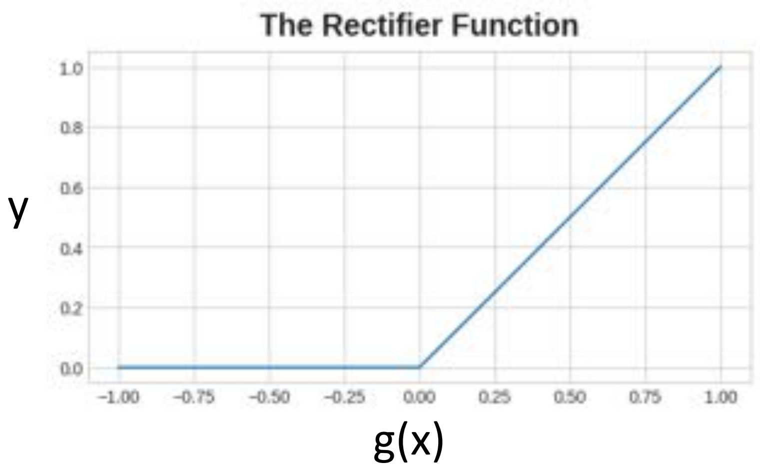


The output is sign(g(x))

decision = +1 if g(x) > 0

decision = -1 if g(x) < 0

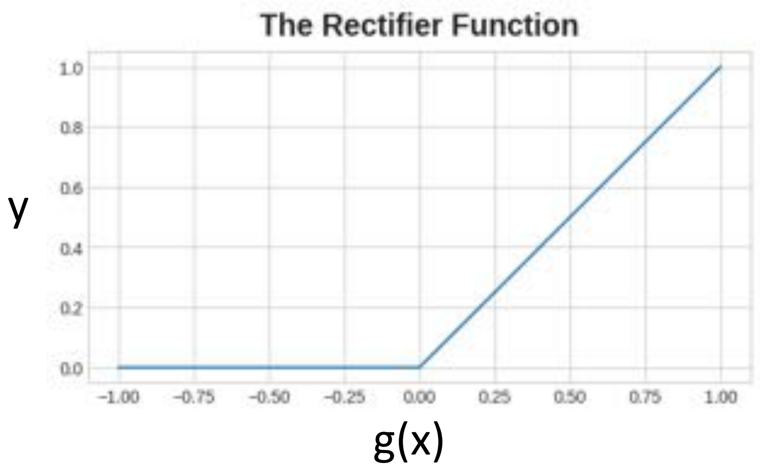
New: activate using the rectifier function

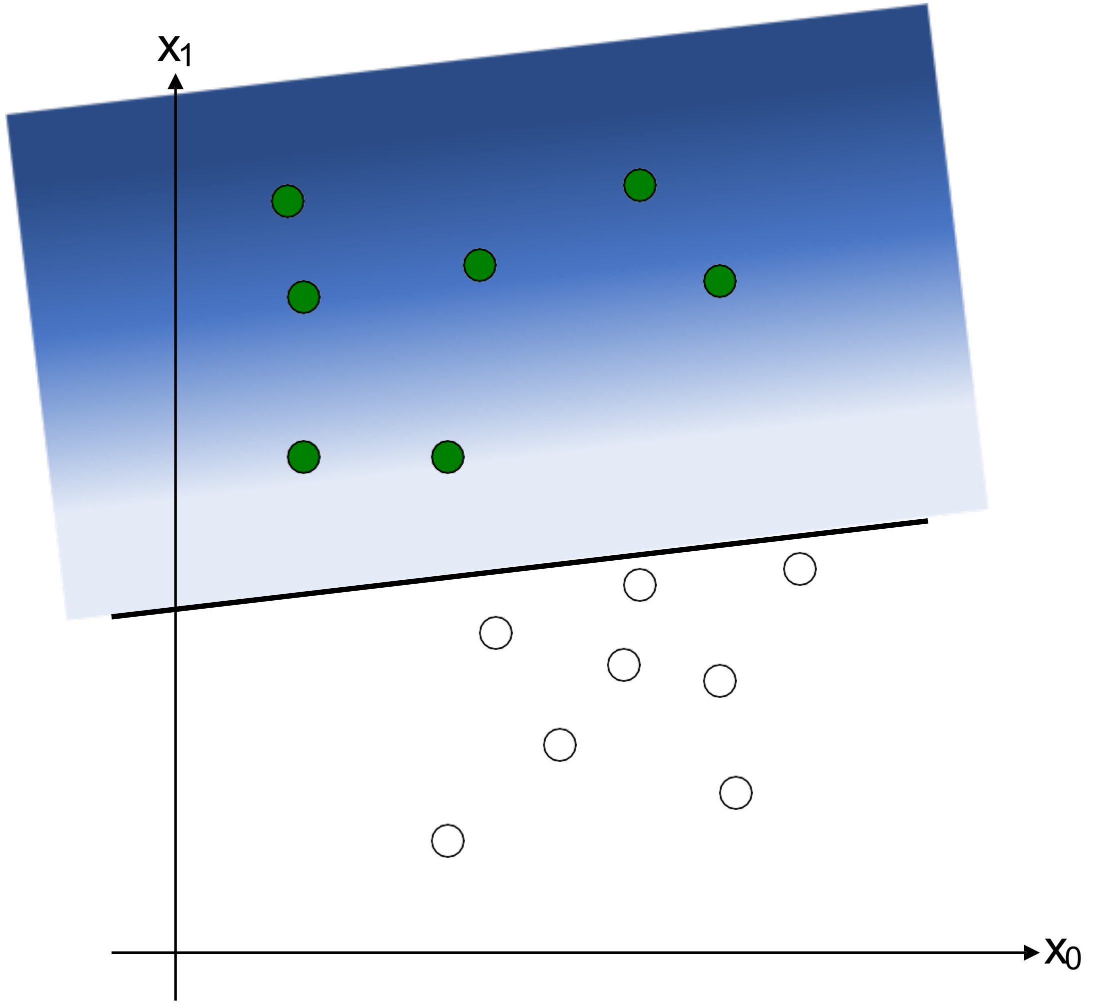


Instead of strict binary sign(g(x)), the output is max(0, g(x))

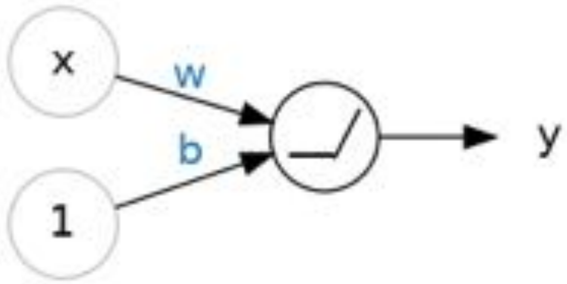
y = g(x) if g(x) > 0
y = 0 if g(x) < 0

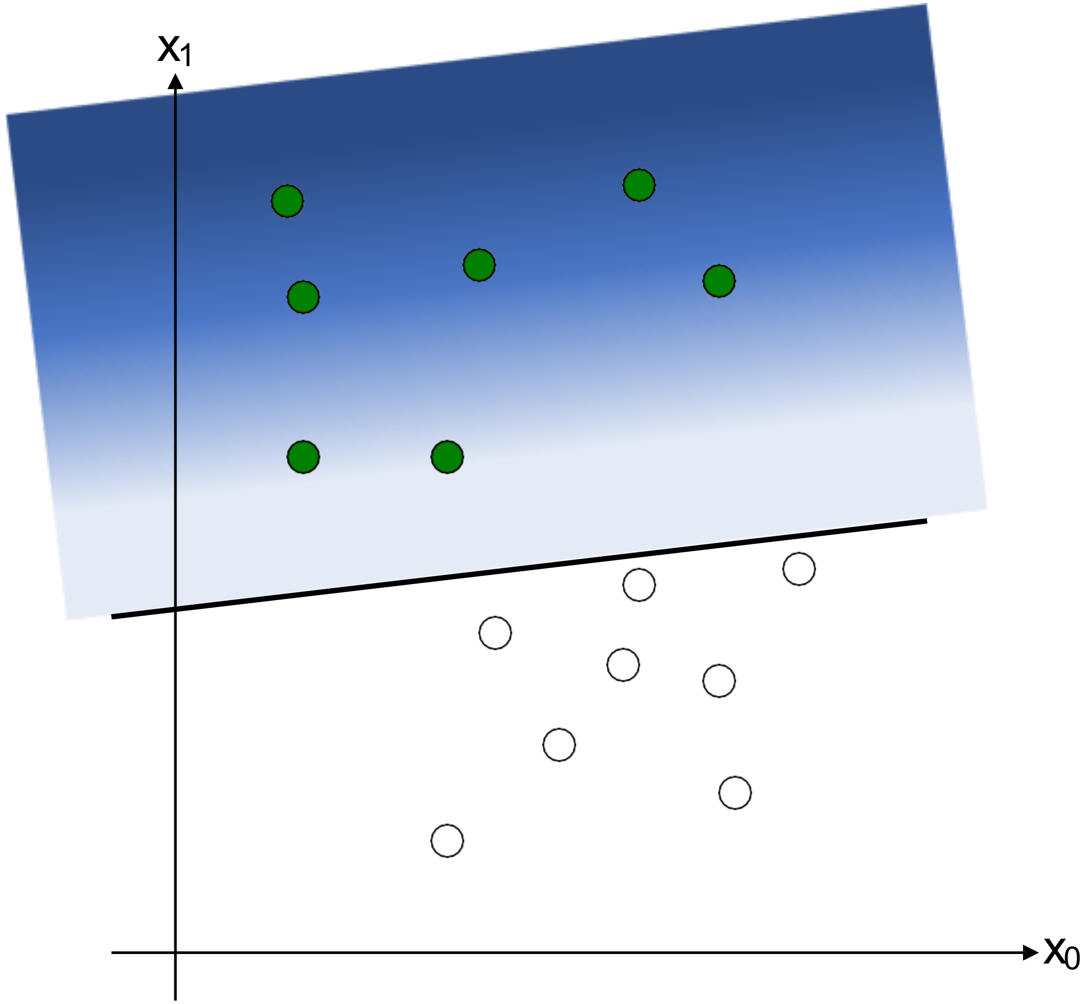
New: activate using the rectifier function





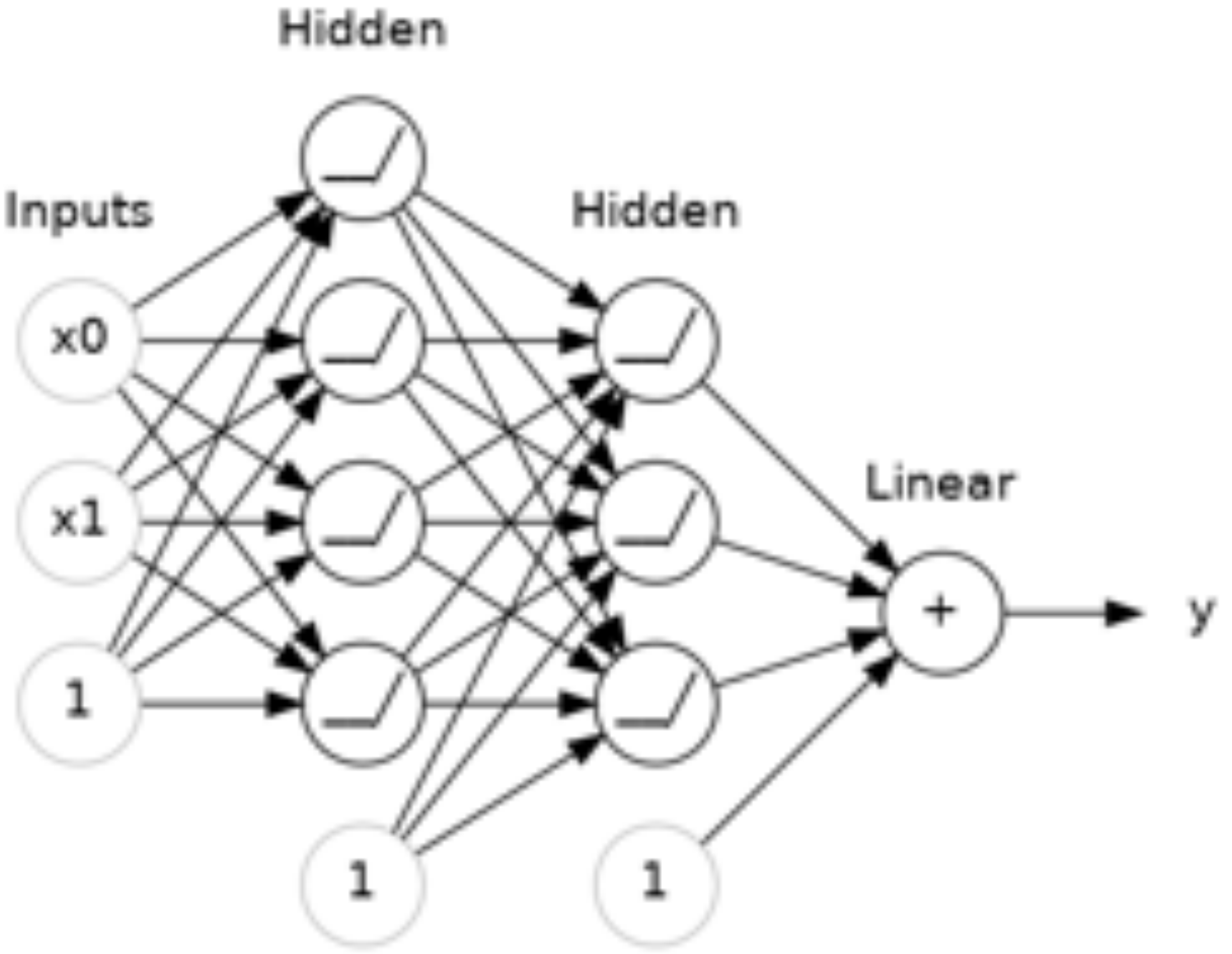
ReLU: a Linear Unit activated using the rectifier function





Putting it all together: Multi-Layer Perceptron

A fully-connected, feed-forward ReLU neural network with two hidden layers



Next time: Neural Nets

Supervised classification using deep learning models

https://playground.tensorflow.org/