Introduction to Machine Learning

Anoop Kunchukuttan

Center for Indian Language Technology
Indian Institute of Technology Bombay

anoop.kunchukuttan@gmail.com

Artificial Intelligence

Designing machines that can perform tasks that are characteristic of human intelligence

Learning

Planning

Reasoning

Problem Solving

Language Understanding & Generation

Object Recognition

Speech Recognition & Generation

Strong AI vs Weak AI

Building AI systems

Rule-based systems

Learning from Data

Machine Learning

Identify the class of Iris plant:

Iris Setosa OR Iris Versicolour OR Iris Virginica

Given features predict class label

Sepal Length	Sepal Width	Petal Length	Petal Width	Class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
7	3.2	4.7	1.4	Iris-versicolor
6.3	3.3	6	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica

Supervised Classification

Features/Signals/Hints

Class Label

Training Data

TRAINING

Sepal Length	Sepal Width	Petal Length	Petal Width	Class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.3	3.3	6.0	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica

X ***Y***

Model **Y = f(X)**

Test Data

DECODING/ INFERENCE

Sepal Length	Sepal Width	Petal Length	Petal Width	Class
6.4	2.8	5.6	2.1	?
7.2	3.0	5.8	1.6	?

What is the nature of the model f(X)?

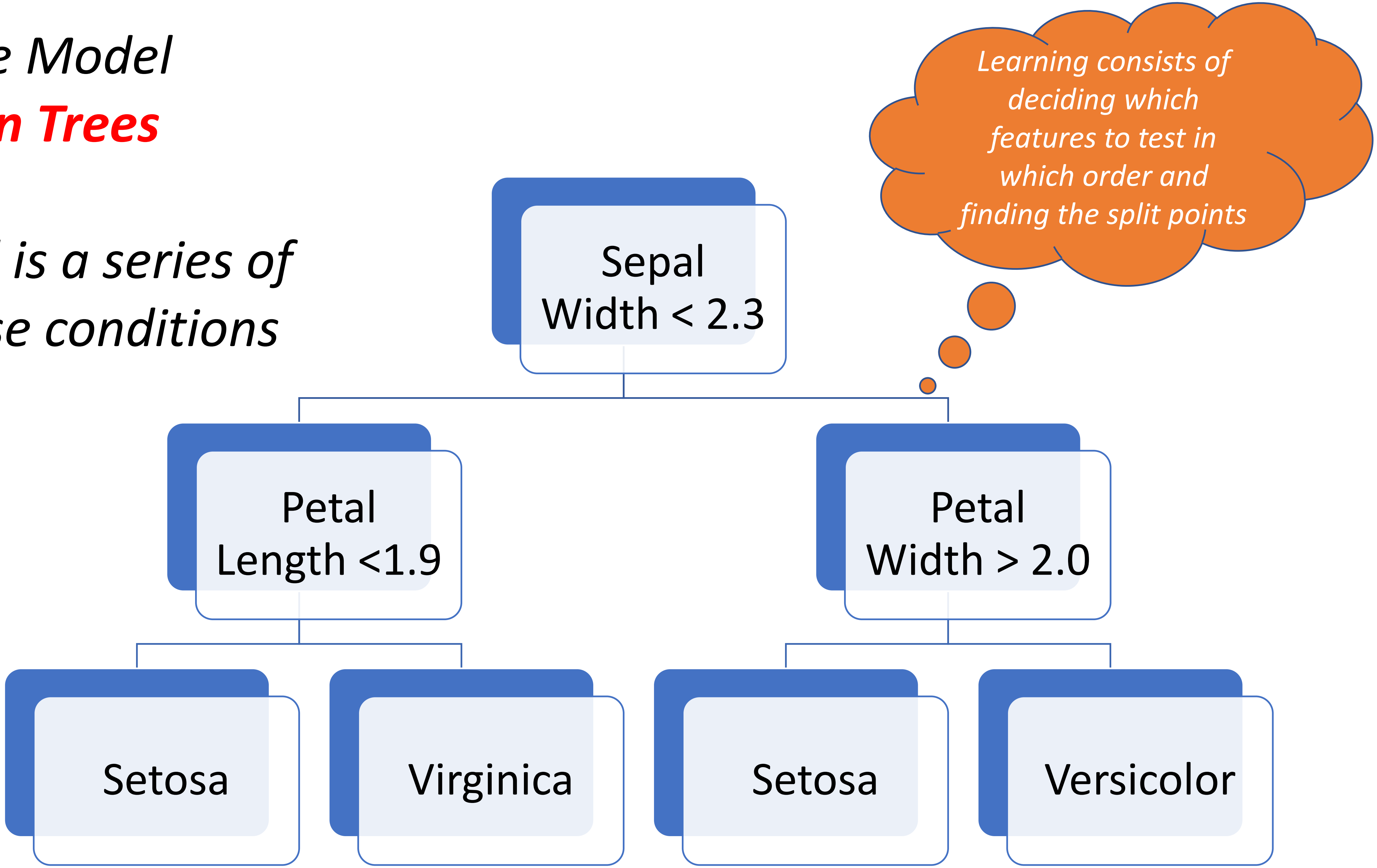
We want f(X) that generalizes well to test data

Obviously, f(X) cannot be a lookup table that memorizes

How do we learn the model $f(X)$?

A Simple Model
Decision Trees

The model is a series of if-then-else conditions

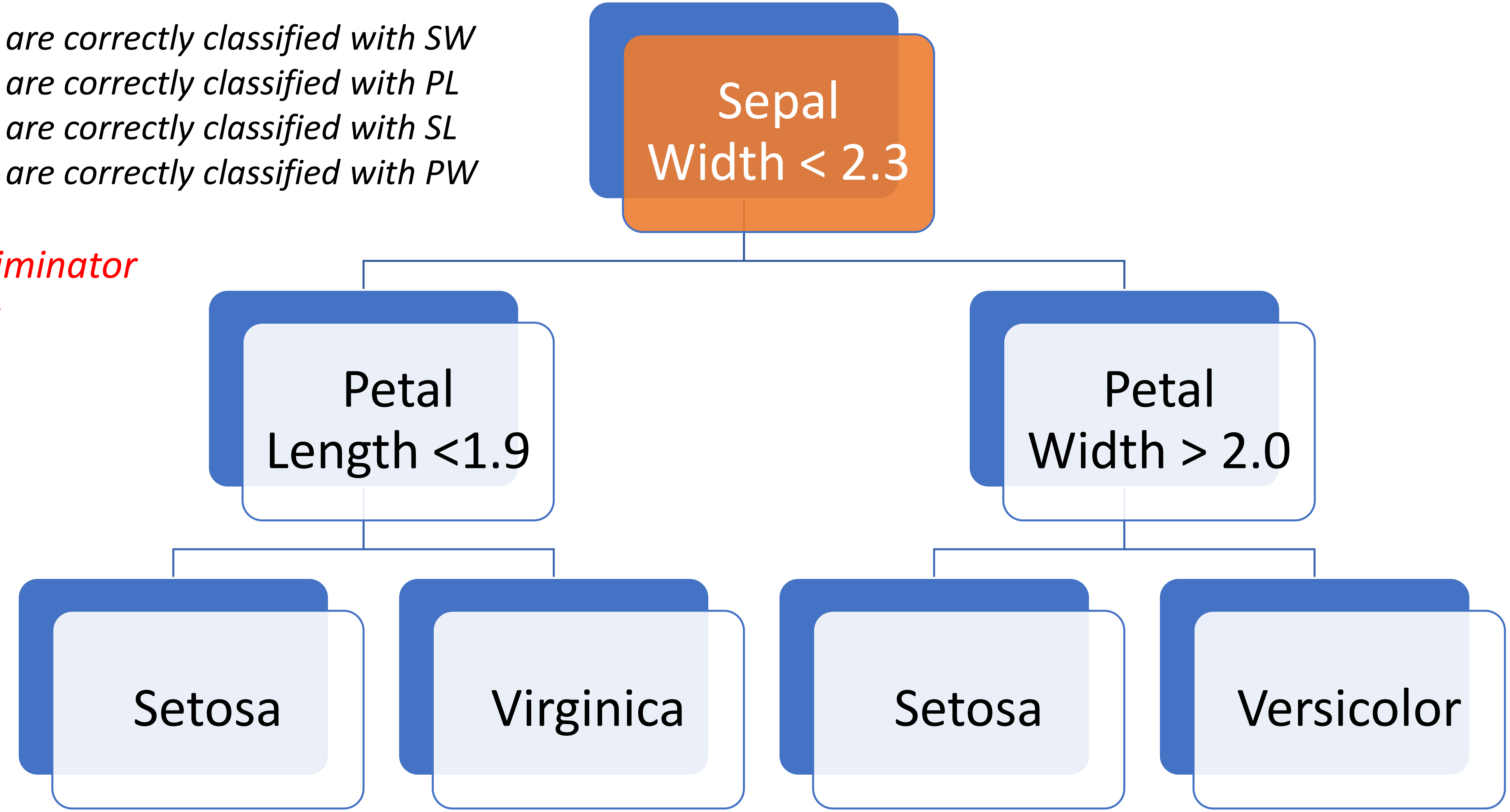


Decision Tree Learning Explained Simply

Say,

- *80% examples are correctly classified with SW*
- *60% examples are correctly classified with PL*
- *50% examples are correctly classified with SL*
- *20% examples are correctly classified with PW*

SW is best discriminator
Split on SW first



Probabilistic Models ➔ Let's look at a simple classifier

Simplified Bayes Decision Rule

$$Y^* = \underset{Y \in \boldsymbol{Y}}{\operatorname{argmax}} \, P(Y|X)$$

We would like to estimate the conditional distribution: P(Y|X)

$$P(Y|X) = \frac{P(X,Y)}{P(X)}$$

$$P(Y|X) \propto P(X,Y)$$
$$\propto P(X|Y)\, P(Y)$$

Likelihood

Prior

$$Y^* = \underset{Y \in \boldsymbol{Y}}{\operatorname{argmax}} \, P(X|Y)\, P(Y)$$

How do we find P(X|Y)?

Maximum Likelihood Estimation

$$P(D) = \prod_{i=1,M} P(X|Y)$$

We can assume P(X|Y) follows some distribution – say Gaussian

$$P(X|Y_1{=}setosa) = N(\mu_1, \sigma_1)$$
$$P(X|Y_2{=}virginica) = N(\mu_2, \sigma_2)$$
$$P(X|Y_3{=}versicolor) = N(\mu_3, \sigma_3)$$

Parameters(Θ)

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} P(D)$$

Objective Function

Training, Validation and Test Sets

Training Set → To learn model parameters

Validation/Development Set →

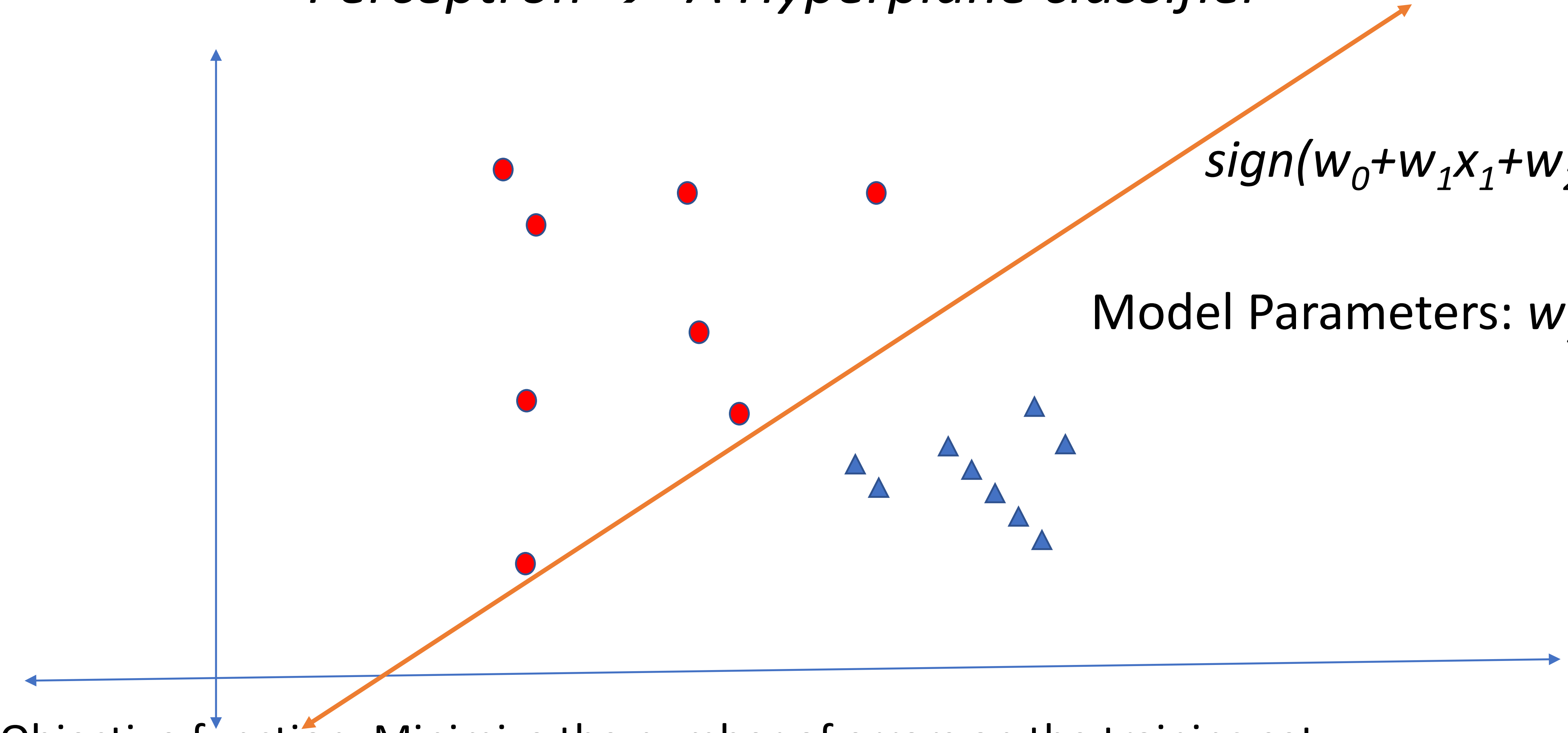
- To select model e.g. what features to use, what distribution to use
- To choose special values → the hyperparameters

Use the test set **only** to evaluate the results

Precision for class k = what fraction of the predictions were correct?
Recall = What fraction of examples of class k were correctly identified
F-1: A metric which combines precision and recall

Using test set for model selection will bias the model

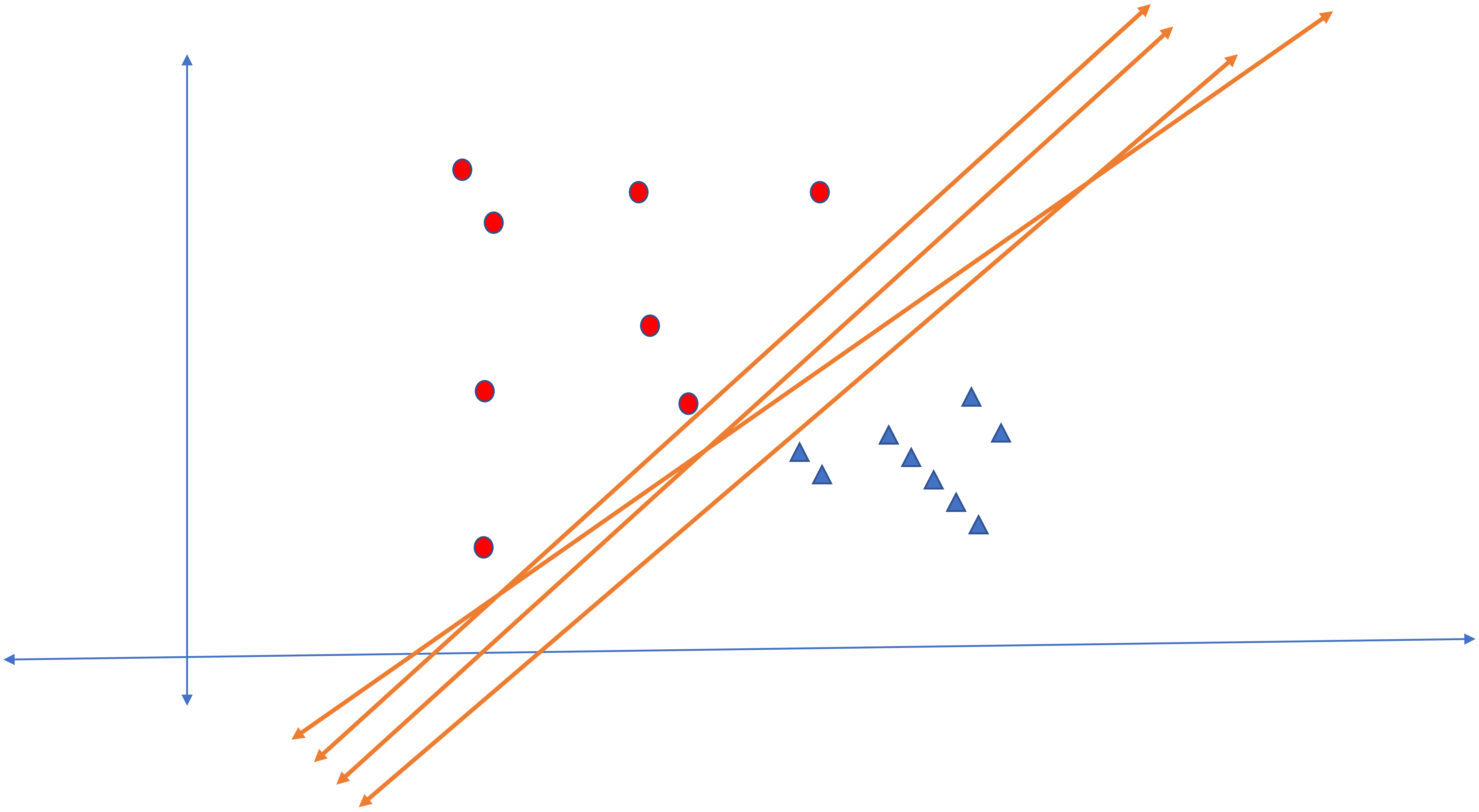
Perceptron → A Hyperplane classifier



$sign(w_0+w_1x_1+w_2x_2)$

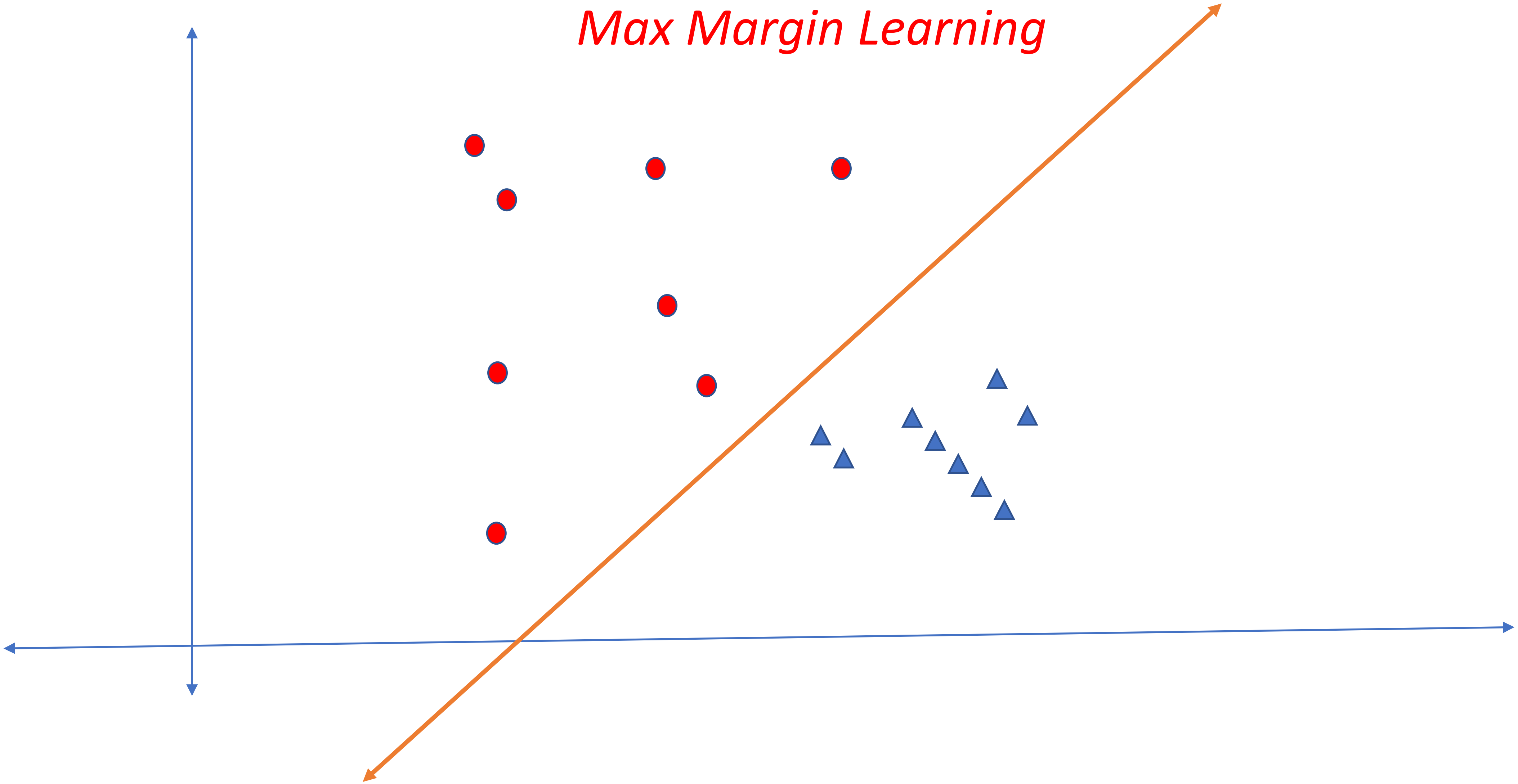
Model Parameters: w_1, w_2

- Objective function: Minimize the number of errors on the training set.
- Zero-one loss function, difficult to optimize
- Approximate with: Minimize $\sum_i^n max(0, -t_i * y_i)$ *where* $t_i = w_1x_{1i}+w_2x_{2i}$



Which line to choose?

Max Margin Learning

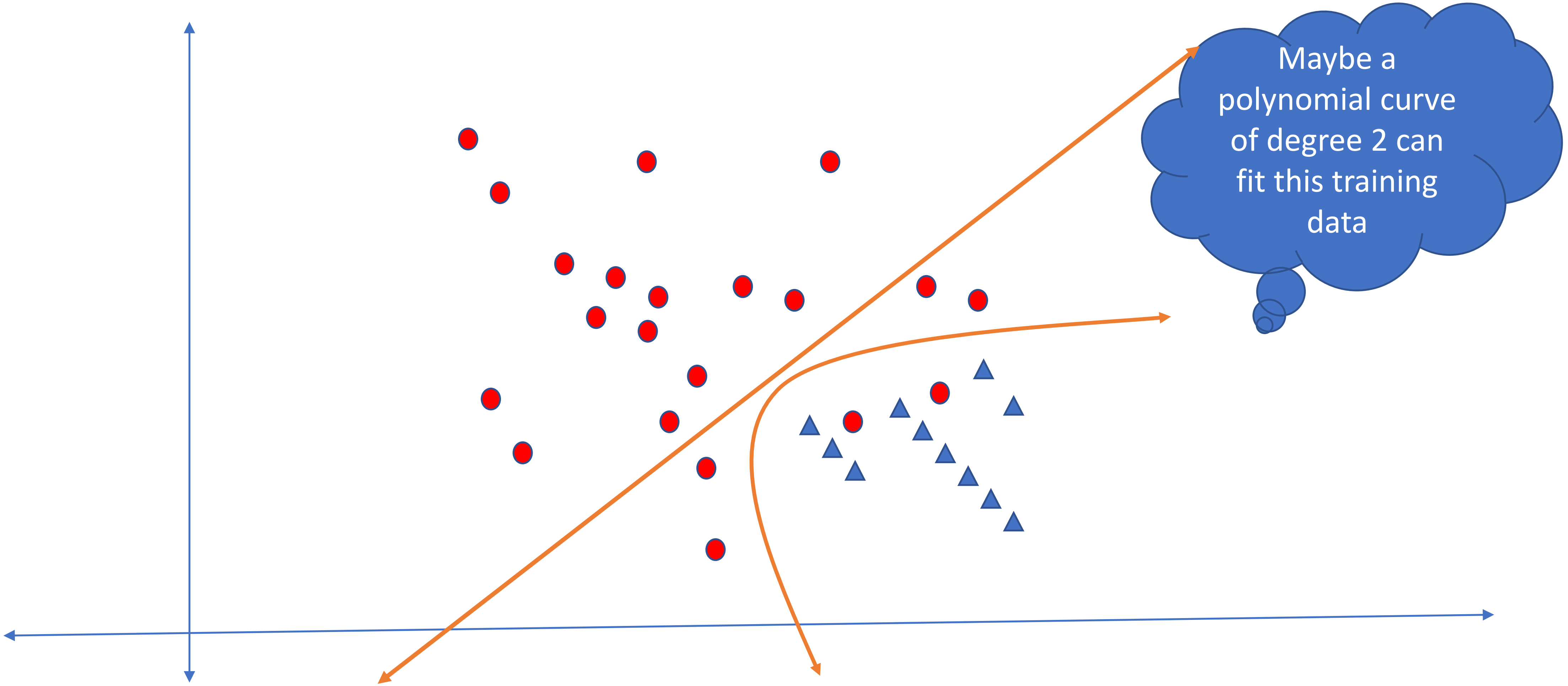


Choose the line which maximizes the margin → this is a more general solution

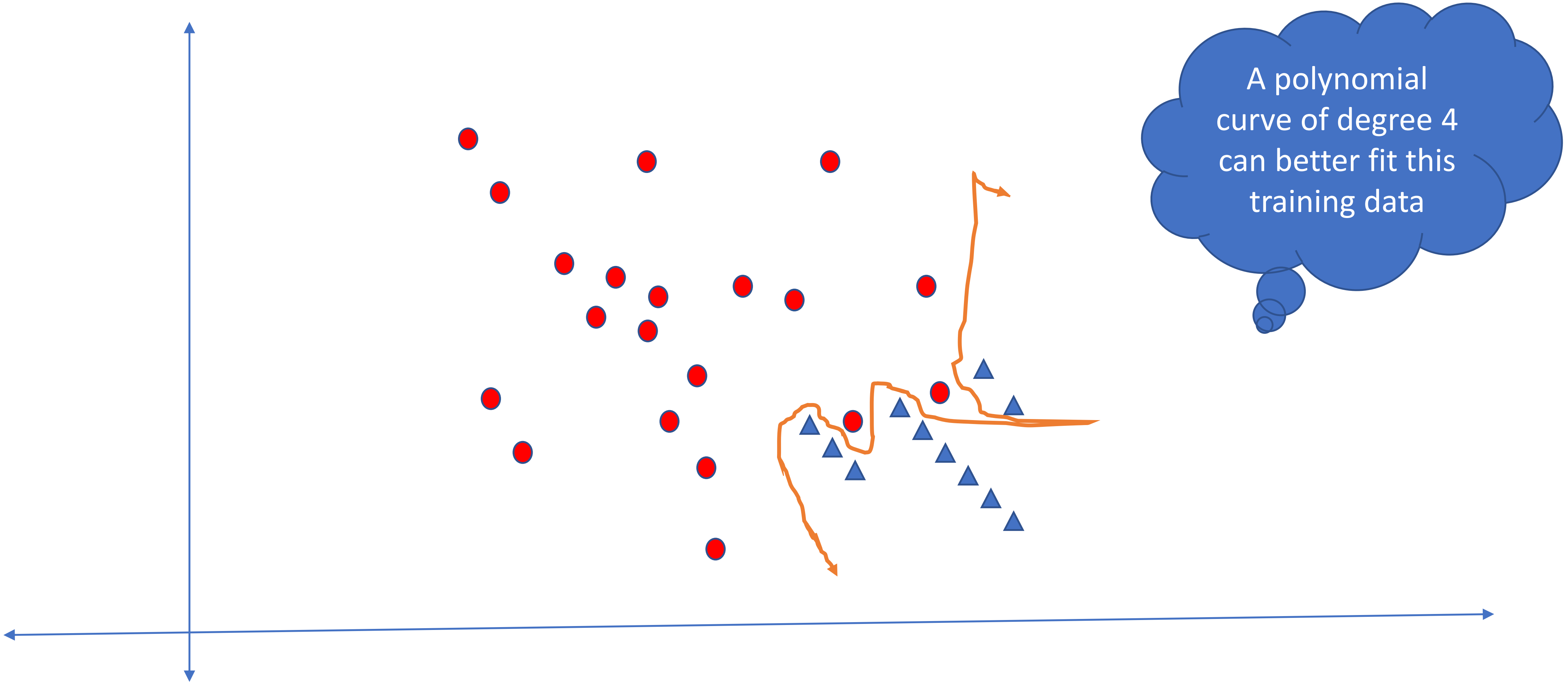
This is the Support Vector Machine classifier

Perceptron & Support Vector Machine are linear classifiers → the decision boundary is a line

In many cases, data is spread out in a more complicated



So, we need non-linear classifiers



But is it desirable? ➔ This could lead to OVERFITTING

Overfitting

- Model performs well on training data, but performs badly on test data
- The model could fit noise in training data
- The more complex the model, the more prone to overfitting
  - As a rule of thumb, more parameter → more complex models
- Occam's Razor: *When presented with competing hypothetical answers to a problem, one should select the answer that makes the fewest assumptions*
  - Prefer simpler models
  - Penalize complexity of the model

How to prevent overfitting?

- Try simpler models first
- There are ways to prevent overfitting
- Regularization is a popular method:
  - Controls range of parameter values
  - Smaller range → lesser variance in the learnt function
- Many model-specific methods exist
  - Tree pruning for decision trees
  - Drop-out for neural networks
- Feature Selection
- Domain knowledge

These represent model biases → Learning is not possible without a bias

We saw one kind of machine learning problem: Supervised Scalar Classification

Supervised → Expected output is known during training

Scalar → Single Output

Classification → Output is one of K possible outcomes (discrete)

But there are many other machine learning scenarios

Regression: Predict a real value instead of a discrete categorical value

e.g. Price Prediction

Sequence Labelling: Input is a sequence, output is a sequence of the same length

e.g. Part-of-Speech Tagging

Sequence-to-Sequence Learning: Input and Output are sequences of different lengths

e.g Machine Translation

English:	Water is necessary for life			(5 words)
Malayalam:	വെള്ളം	ജീവന്	അത്യാവശ്യമാണ്	(3 words)
	water	life+for	necessary+is	

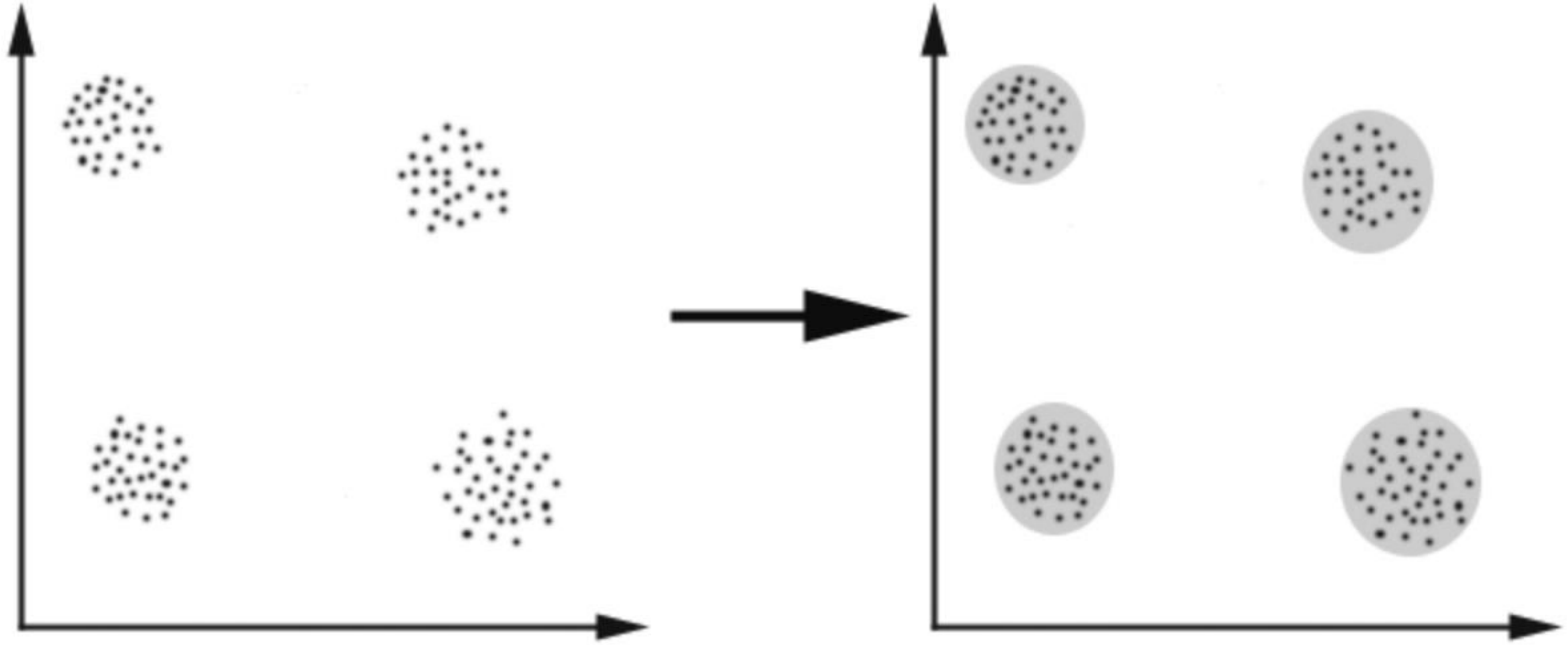
More complicated output structures: trees, graphs, *etc*

Reinforcement Learning: Weak supervision, just rewards and penalties on predictions

e.g. game-playing, car driving

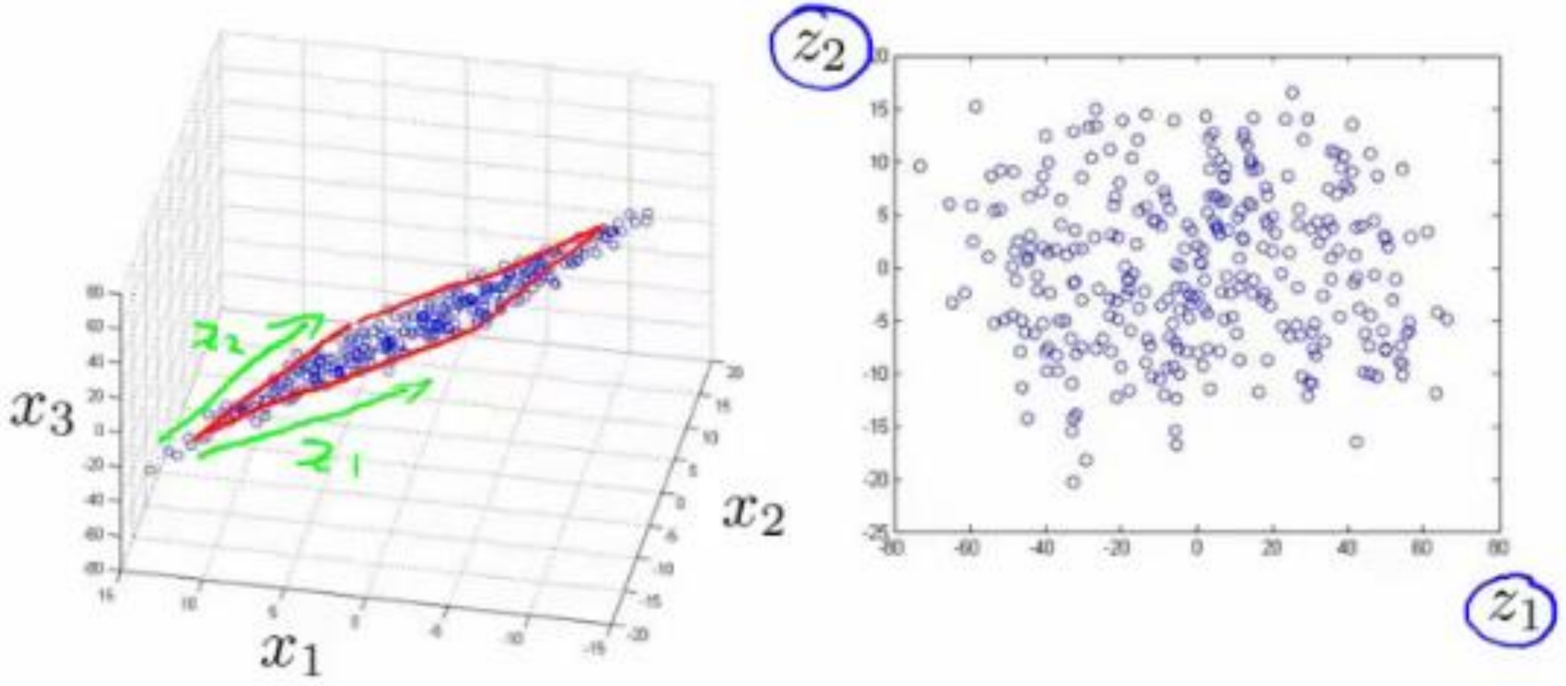
Unsupervised Learning: Given only data, no labels → let's know a little bit more about this

Clustering



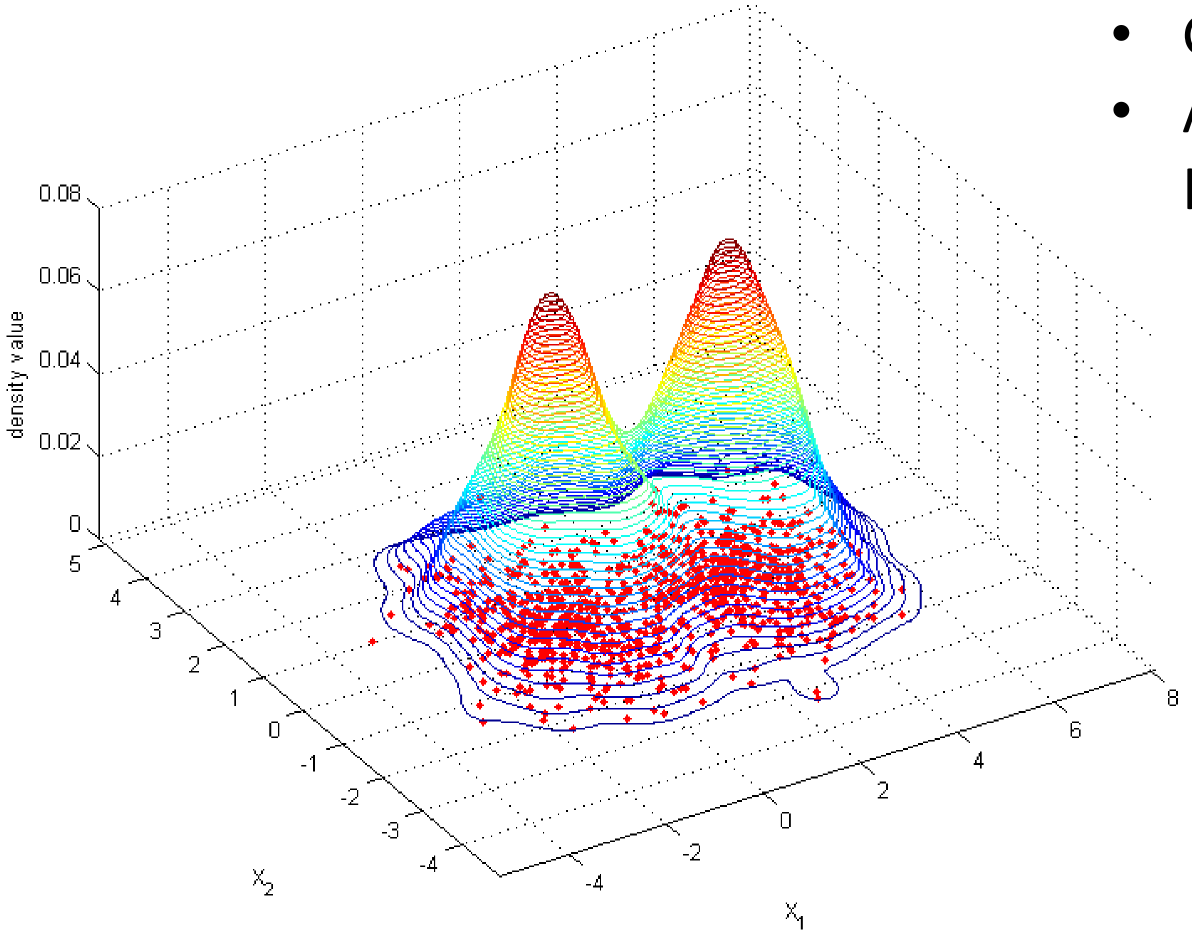
- Discover patterns in data, exploratory analysis
- Popular Algorithms: k-means, Expectation Maximization with Mixture Models, agglomerative hierarchical clustering

Lower Dimensional Representation



- Noise reduction, computational efficiency, avoiding overfitting, visualization
- Popular Algorithms: Principal Component Analysis, Singular Value Decomposition, Latent Dirichlet Allocation, Autoencoding

Density Estimation



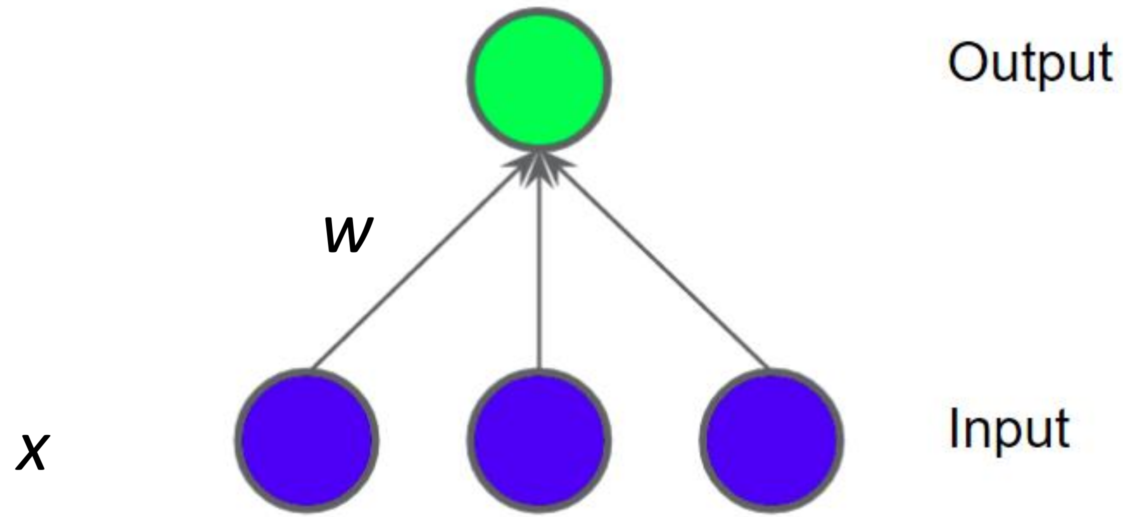
- Novely and anamoly detection
- Generating synthetic datasets
- Algorithms: Parzen windows, Kernel Density estimation

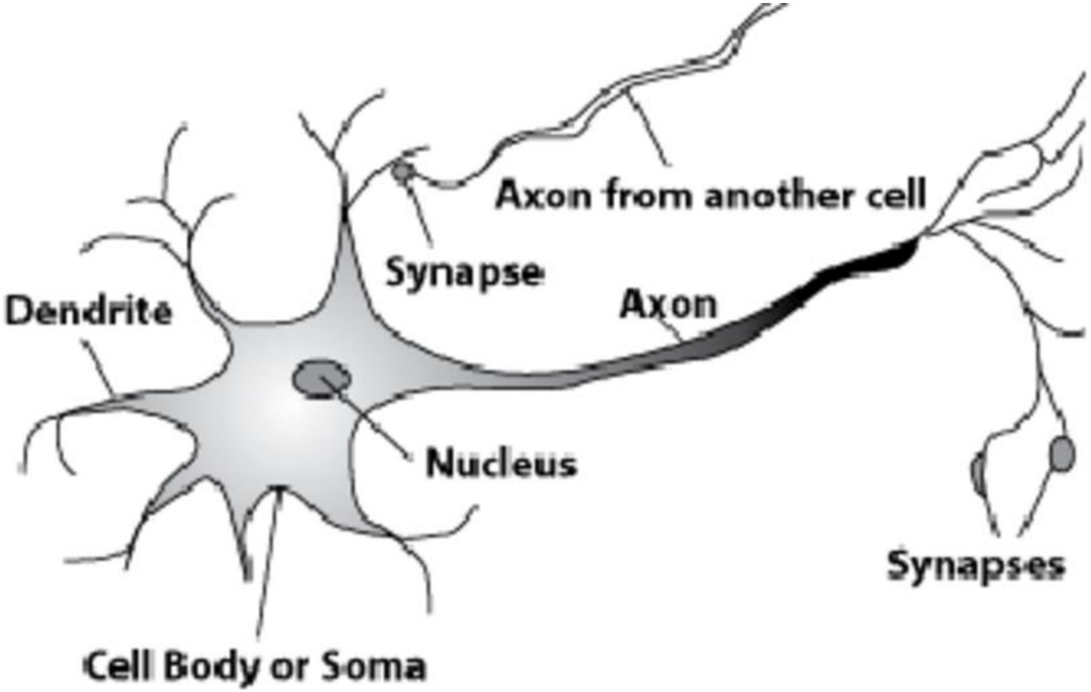
Deep Learning

The return of artificial neural networks

Perceptrons → The Simplest Network

$sign(w_0+w_1x_1+w_2x_2+w_3x_3)$



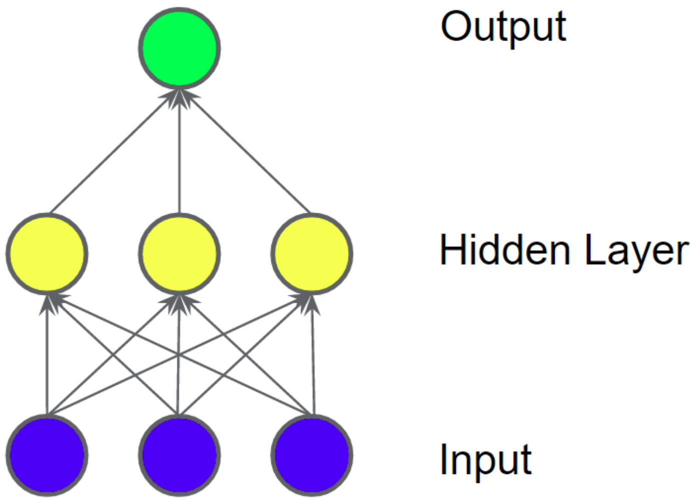


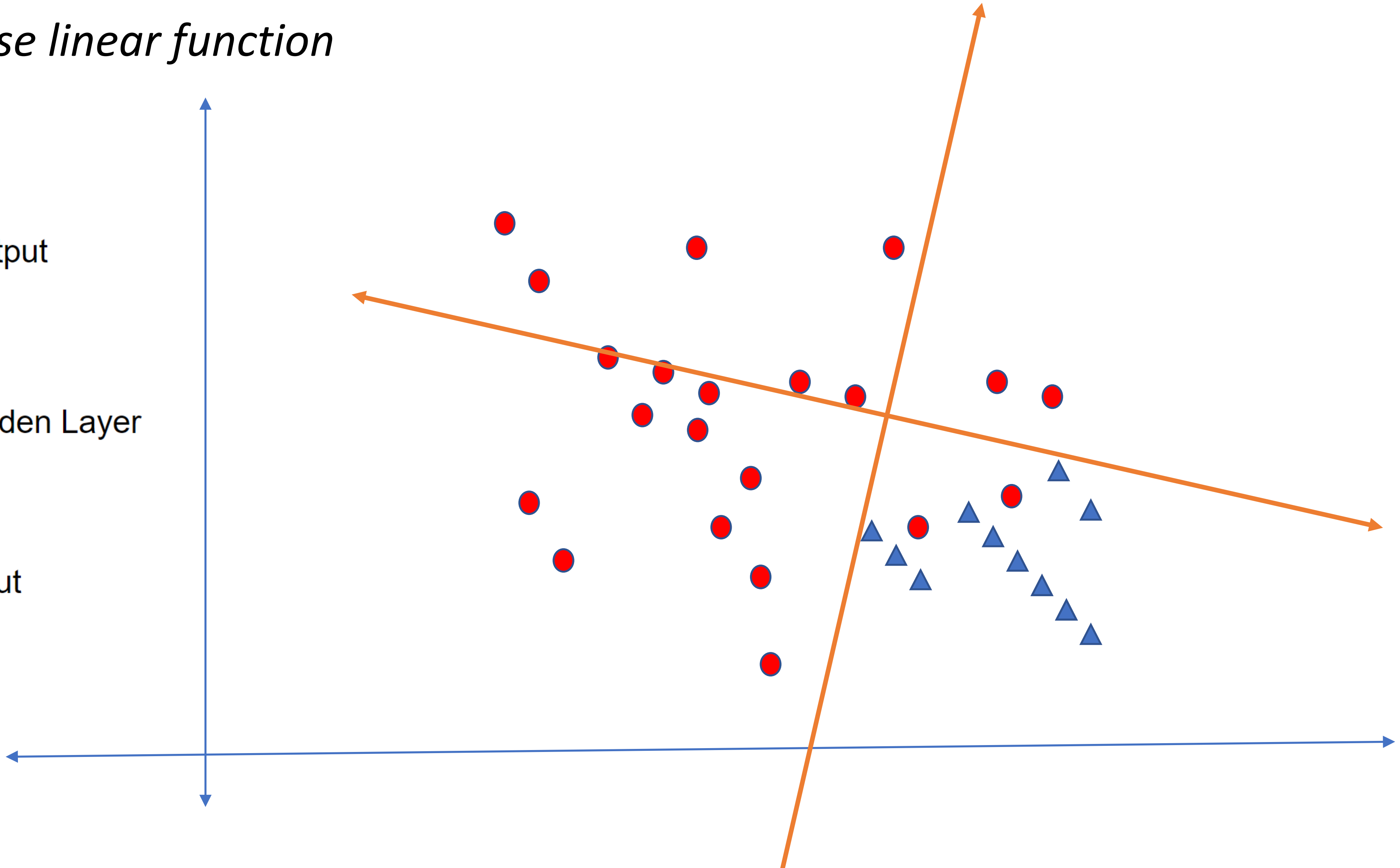
Input → Dendrites   Weighted sum of inputs

Outputs → Axons   Output only if weighted sum greater than a threshold (w_0)

Add multiple perceptron units

Learns a piece-wise linear function

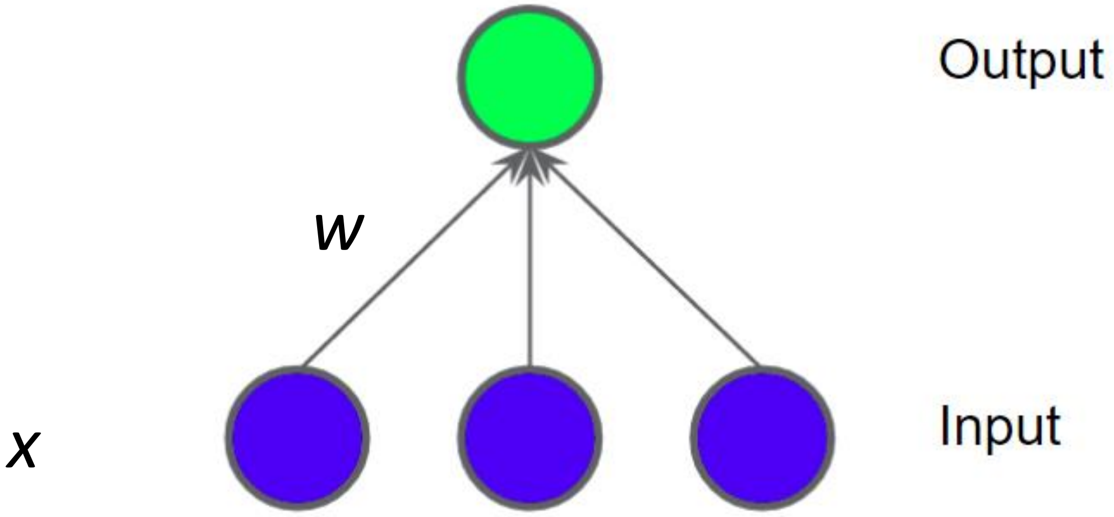




This is still not sufficient

Add a non-linear activation

$$\text{sigmoid}(w_0+w_1x_1+w_2x_2+w_3x_3)$$



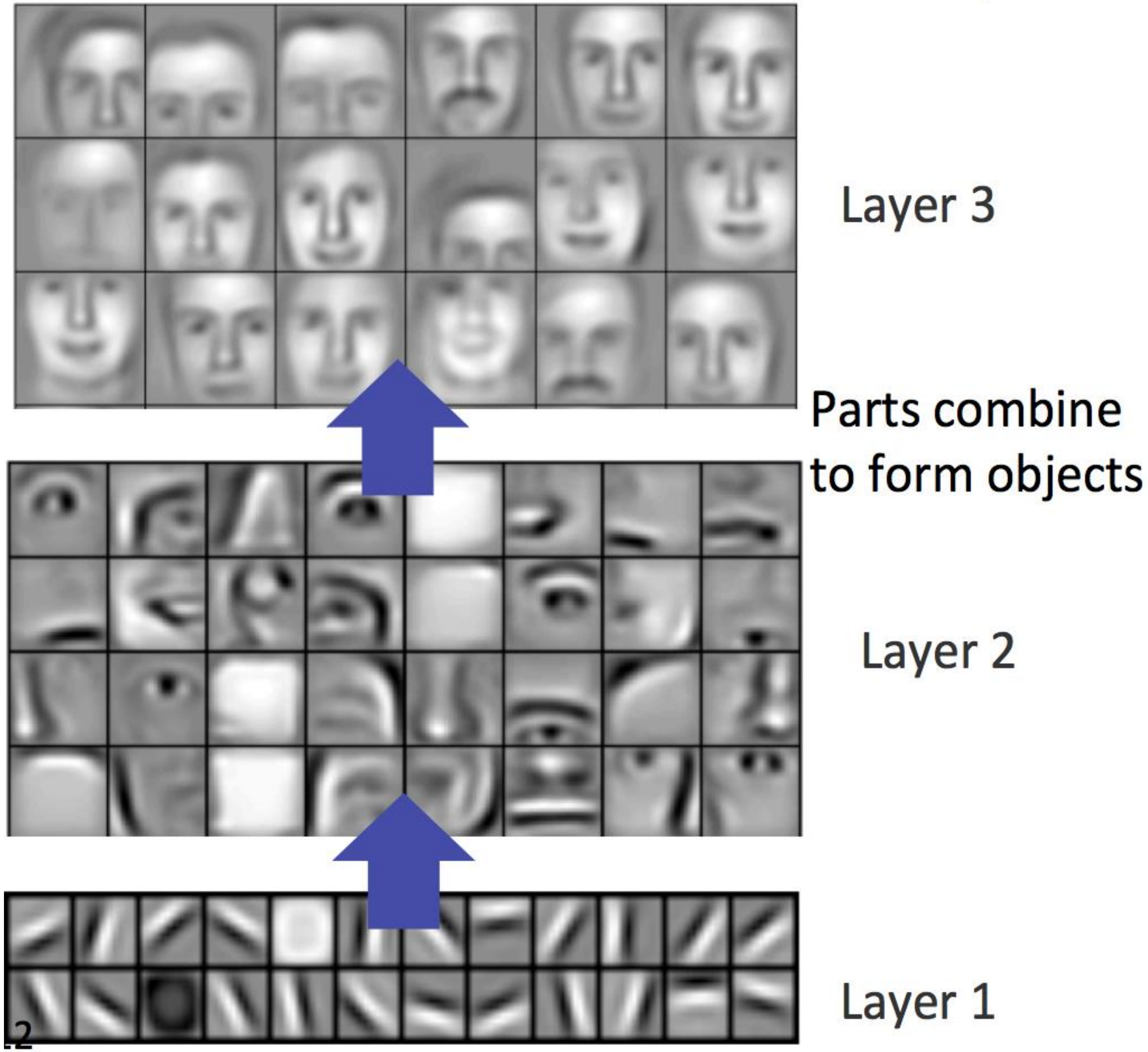
A network with non-linear activations and at least one hidden layer can approximate any function

The network is made of small simple learning units

Why have deep neural networks become popular?

Deep Neural Networks ➔ Many Hidden Layers

- Deep networks have millions of parameters → prone to overfitting
- Lots of data mitigates overfitting to some extent
- Modern hardware makes it possible to train deep networks
- Better initialization, optimization and activation methods
- Very developer friendly language and good toolkits
- Representation Learning → Features are automatically learnt



DNNs have achieved tremendous success in many perceptual task since as image recognition, vision, speech

Lesser success in language – to some extent in machine translation

Some Reading Material

- Google’s Crash Course (Basic)
- Andrew Ng’s Coursera Course (Intermediate)
- Andrew Moore’s course material (Intermediate)
- Andrew Ng’s Stanford Course (Advanced)
- Mathematics Pre-requisites for Advanced Learning
  - Probability and Statistics
  - Linear Algebra
  - Optimization

Some Tools and Software

- Weka Library (Java)
- Scikit Learn (Python)
- Deep learning frameworks like Tensorflow, Torch/PyTorch, MXNet

Data Sources

- UC Irvine Machine Learning Repository
- Kaggle
- .. Many other sources

Thank you!

https://www.cse.iitb.ac.in/~anoopk

anoop.kunchukuttan@gmail.com