Machine Learning
ITCS 5356

Introduction

Razvan C. Bunescu

Department of Computer Science @ CCI

rbunescu@uncc.edu

How to Solve Computational Problems?

- A **computational problem** is a task that can be solved by a computer, i.e. by the mechanical application of a sequence of steps, i.e. by computer code.

- Where does the **computer code** come from?
  - **Expert Systems (**aka **rule-based** or **traditional programming**):
    1. Experts write rules that capture **patterns** about the problem.
    2. Programmers implement the rule-based solution in **code**.
  - **Machine Learning** = program computers to *learn* from *experience* to improve performance on a given task.
    - Automatically discover **patterns** from solved problem instances (i.e. *experience*) that enable solving new instances of the problem.
    - Trained **ML model** is **code** that does pattern recognition.

We use ML to automate solutions to Computational Problems

- Why use a **Machine Learning (ML)** approach:
  - Because ML is **hot**? Because ML is **The solution**?
- **Traditional programming** may work very well:

What is the solution of $x^2 - 4x + 3$?

How do I get from UNCC campus to the Mint museum uptown?

Spam Filtering is a Computational Problem

From: Tammy Jordan	**From: UK National Lottery**
jordant@oak.cats.ohiou.edu	**edreyes@uknational.co.uk**
Subject: Spring 2015 Course	**Subject: Award Winning Notice**
--------------------------------------	**--------------------------------------**
CS690: Machine Learning Instructor: Razvan Bunescu Email: bunescu@ohio.edu Time and Location: Tue, Thu 9:00 AM , ARC 101 Website: http://ace.cs.ohio.edu/~razvan/courses/ml6830 Course description: Machine Learning is concerned with the design and analysis of algorithms that enable computers to automatically find patterns in the data. This introductory course will give an overview …	**UK NATIONAL LOTTERY. GOVERNMENT ACCREDITED LICENSED LOTTERY. REGISTERED UNDER THE UNITED KINGDOM DATA PROTECTION ACT;** **We happily announce to you the draws of (UK NATIONAL LOTTERY PROMOTION) International programs held in London , England Your email address attached to ticket number :3456 with serial number :7576/06 drew the lucky number 4-2-274, which subsequently won you the lottery in the first category …**

- Example rules or patterns for an expert systems approach:
  - ”MONEY” appears in the text => *Spam*.
    - What if email sent by grandmother?

How to Automate Solutions to Computational Problems?

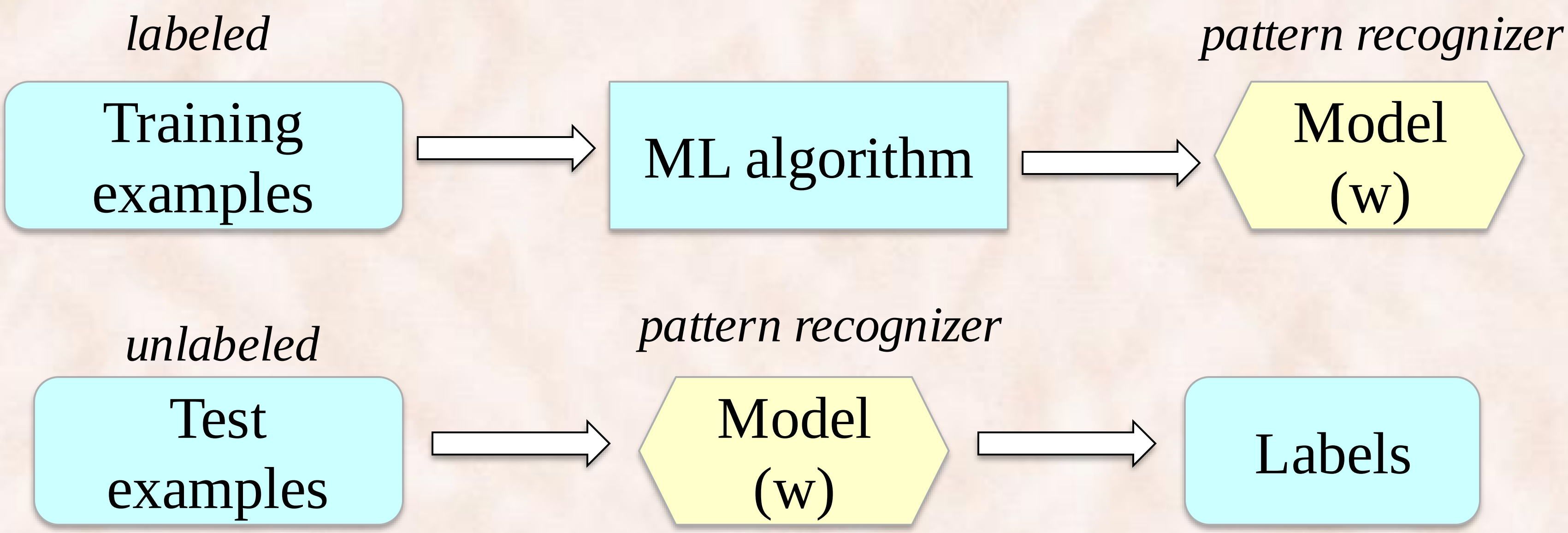
- **Expert Systems** approach:
  - **Cognitively demanding**:
    - Difficult for humans to reason with many useful but imprecise features that are indicative (signals) of spam or not spam:
      - Words, phrases, images, meta-data, time series, …
      - Need to *combine a large number of signals*, figure out their *relative importance* in determining spam vs. ham label.
  - **Brittle**: Always going to miss some useful features or patterns.
      - "*All grammars leak.*" (Edward Sapir).
      - Spam filtering is adversarial, new features need to be added over time.
  - \+ Often much more **interpretable** than ML approach!
    - \+ Often better at **systematic generalization** too …

How to Automate Solutions to Computational Problems?

- **Machine Learning (ML)** approach:
  1. **Data acquisition**: create a large enough dataset of *labeled examples*:
     - Email is the *example*, the *label* is spam (+1) vs. not spam (−1).
     - Collecting labels is easier than writing rules!
  2. **Feature engineering**: Represent examples as *feature vectors*, each feature has a *weight*.
  3. **Learn the weights** such that the model prediction (weighted combination of features) matches the labels of training examples.

What is Machine Learning?

- **Machine Learning** = constructing computer programs that *learn* from *experience* to perform well on a given task.
  - **Supervised Learning** i.e. discover **patterns** from labeled examples that enable predictions on unseen examples.

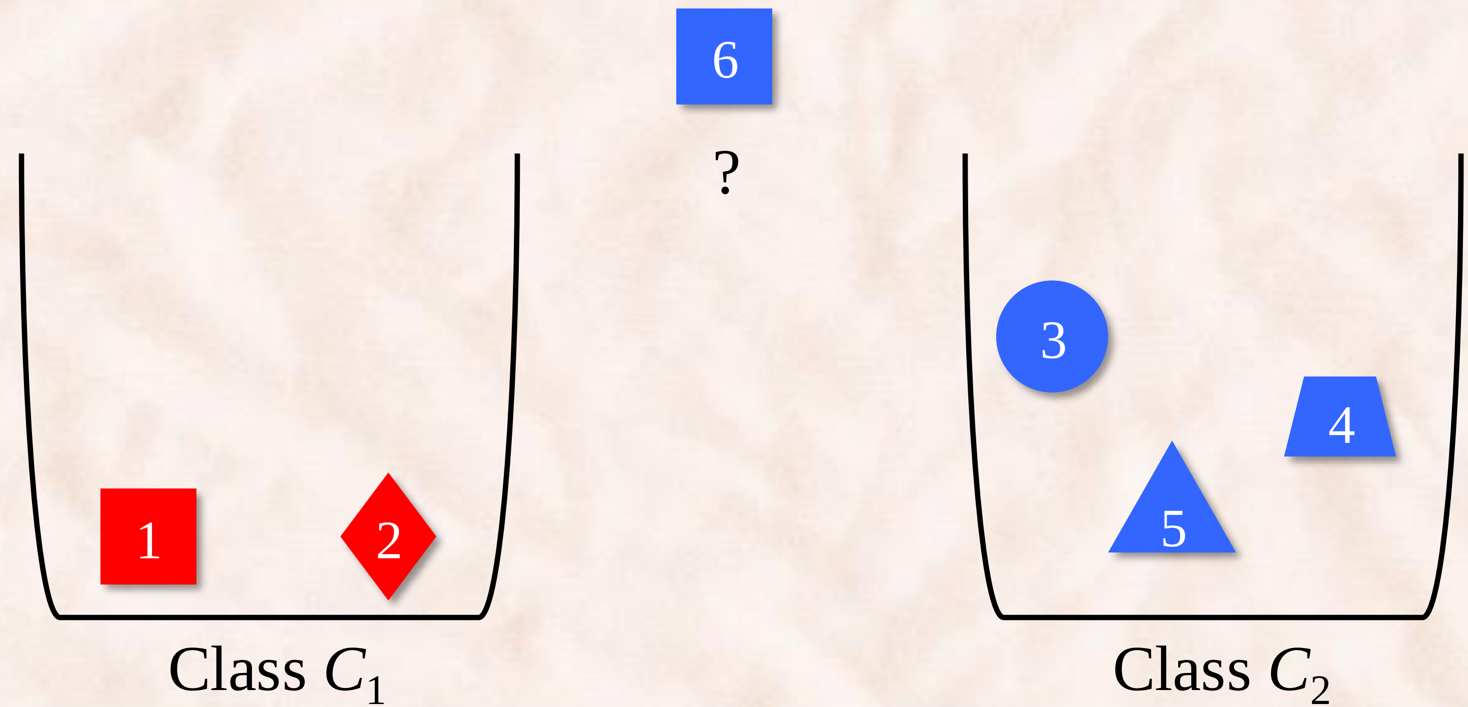


Human Learning

M_1: x is Red => $x \in C_1$

M_2: x is a Square or x is a Diamond => $x \in C_1$

M_3: x is Red and x is a Quadrilateral => $x \in C_1$



Occam's Razor

William of Occam (1288 – 1348)

English Franciscan friar, theologian and philosopher.

"*Entia non sunt multiplicanda praeter necessitatem*"

– Entities must not be multiplied beyond necessity.

i.e. Do not make things needlessly complicated.

i.e. Prefer the simplest hypothesis that fits the data.

Occam's Razor vs. Kolmogorov Complexity, Intelligence & Science

- **Kolmogorov Complexity** = the length of the shortest program that generates the data.
  - 1, 2, 4, 7, 11, 16, 22, 29, 37, 46, 56, …
  - 1, 3, 6, 11, 18, 29, …
  - 1, 2, 3, 5, 5, 8, 7, 11, 9, …

- **Intelligence** = the ability to apply Occam's Razor.
  - http://www.vetta.org/documents/Machine_Super_Intelligence.pdf

- **Science** = discover the simplest descriptions of our world.
  - https://doi.org/10.1111/nyas.15086

ML Objective

- Find a model **M**

  that is *simple* + that *fits the training data*.

$$\hat{\mathbf{M}} = \underset{\mathbf{M}}{\operatorname{argmin}} \; \mathit{Complexity}(\mathbf{M}) + \mathit{Error}(\mathbf{M}, \mathit{Data})$$

- **Inductive hypothesis**: Models that perform well on training examples are expected to do well on test (unseen) examples.

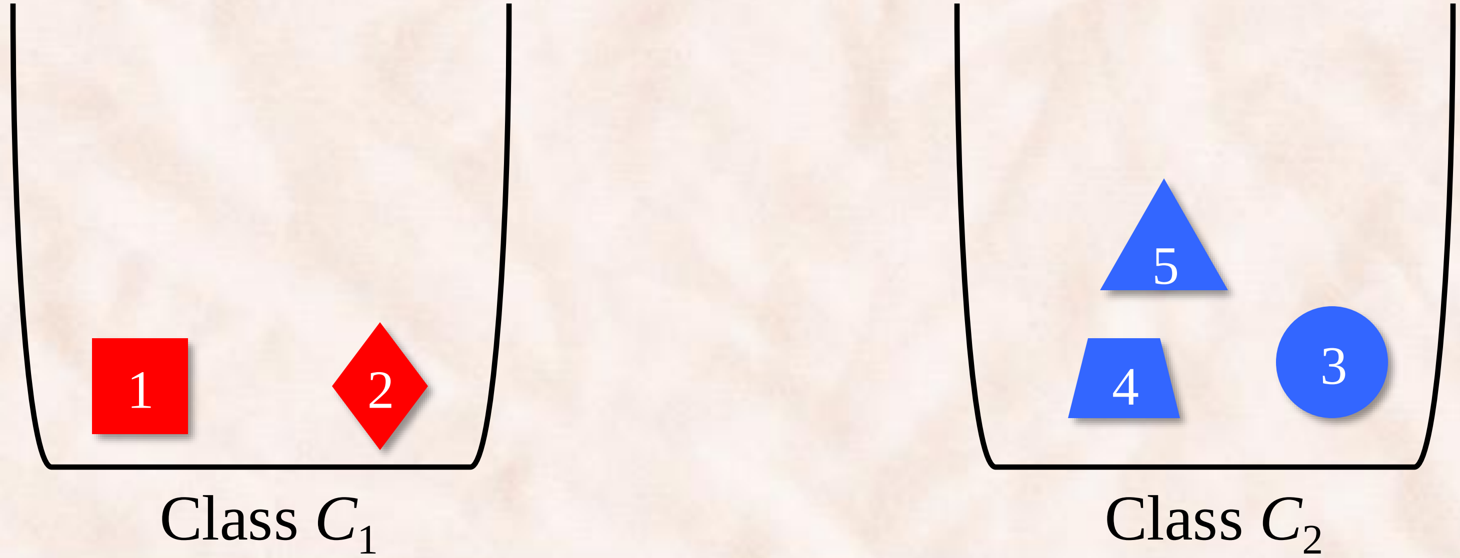
- **Occam's Razor**: Simpler models are expected to do better than complex models on test examples (assuming similar training performance).

Example

M_1: x is Red => $x \in C_1$

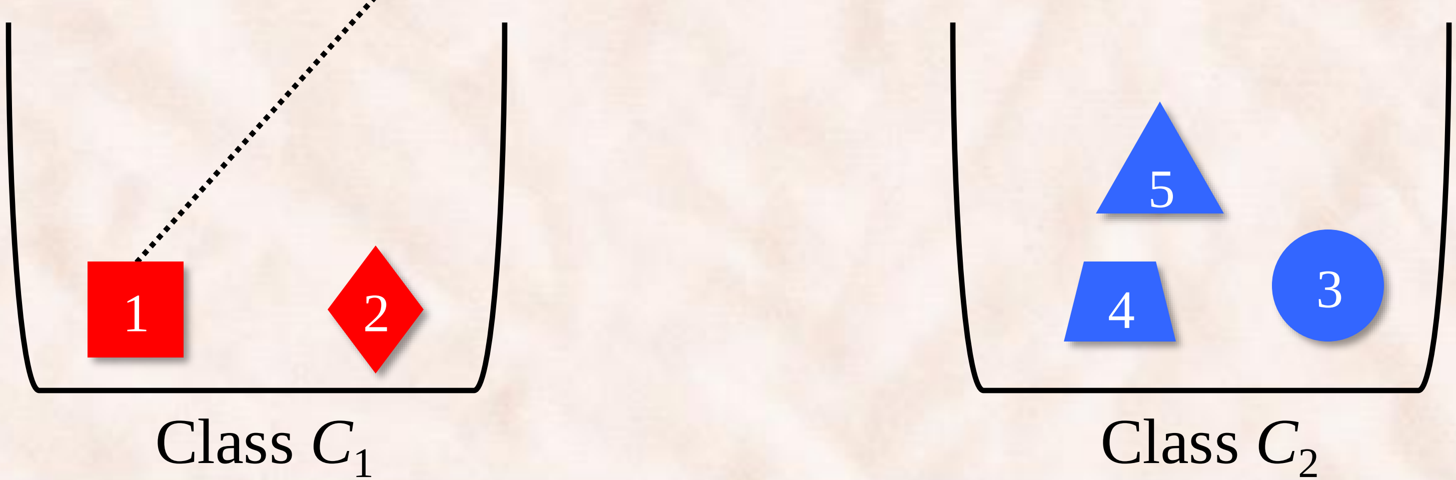
M_2: x is a Square or x is a Diamond => $x \in C_1$

M_3: x is Red and x is a Quadrilateral => $x \in C_1$



Feature Vectors

Features	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$
Red?	1	1	0	0	0
Quad?	1	1	0	1	0
Square?	1	0	0	0	0
Diamond?	0	1	0	0	0
Label (y)	$y_1 = +1$	$y_2 = +1$	$y_3 = -1$	$y_4 = -1$	$y_5 = -1$



Learning with a Linear Classifier

Features	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$
Red?	1	1	0	0	0
Quad?	1	1	0	1	0
Square?	1	0	0	0	0
Diamond?	0	1	0	0	0
Label (y)	$y_1 = +1$	$y_2 = +1$	$y_3 = -1$	$y_4 = -1$	$y_5 = -1$

$\mathbf{x}_1 = [1, 1, 1, 0]$ $\quad$ $\mathbf{x}_2 = [1, 1, 0, 1]$ $\quad$ $\mathbf{x}_3 = [0, 0, 0, 0]$ $\quad$ ...

$y_1 = +1$ $\quad$ $y_2 = +1$ $\quad$ $y_3 = -1$ $\quad$ ...

Learning = finding parameters $\mathbf{w}^T = [w_1, w_2, w_3, w_4]$ and τ such that:

- $\mathbf{w}^T \mathbf{x}_i \geq \tau$, if $y_i = +1$
- $\mathbf{w}^T \mathbf{x}_i < \tau$, if $y_i = -1$

where $\mathbf{w}^T \mathbf{x} = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$ $\quad$ $\mathbf{w} = [w_1, w_2, w_3, w_4]$ $\quad$ $\mathbf{x} = [x_1, x_2, x_3, x_4]$

Model M_1: $\mathbf{x}_i$ is Red => $y_i = +1$

Red? Quad? Square? Diamond?

$\mathbf{x}_1 = [1, 1, 1, 0]^T$ label $y_1 = +1$ => $\mathbf{w}^T\mathbf{x}_1 = 1 \geq 0.5$

$\mathbf{x}_2 = [1, 1, 0, 1]^T$ label $y_2 = +1$ => $\mathbf{w}^T\mathbf{x}_2 = 1 \geq 0.5$

$\mathbf{x}_3 = [0, 0, 0, 0]^T$ label $y_3 = -1$ => $\mathbf{w}^T\mathbf{x}_3 = 0 < 0.5$

$\mathbf{x}_4 = [0, 1, 0, 0]^T$ label $y_3 = -1$ => $\mathbf{w}^T\mathbf{x}_4 = 0 < 0.5$

$\mathbf{x}_5 = [0, 0, 0, 0]^T$ label $y_3 = -1$ => $\mathbf{w}^T\mathbf{x}_5 = 0 < 0.5$

w = [1, 0, 0, 0]

= > M_1 error is 0%

Learning = finding parameters $\mathbf{w}^T = [w_1, w_2, w_3, w_4]$ such that (τ =0.5):

- $\mathbf{w}^T \mathbf{x}_i \geq 0.5$, if $y_i = +1$
- $\mathbf{w}^T \mathbf{x}_i < 0.5$, if $y_i = -1$

$\mathbf{w} = [w_1, w_2, w_3, w_4]$

where $\mathbf{w}^T \mathbf{x} = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$

$\mathbf{x} = [x_1, x_2, x_3, x_4]$

M_2: $\mathbf{x}_i$ is Square or Diamond => $y_i = +1$

Red? Quad? Square? Diamond?

$\mathbf{x}_1 = [1, 1, 1, 0]^T$ label $y_1 = +1$ => $\mathbf{w}^T\mathbf{x}_1 = 1 \geq 0.5$

$\mathbf{x}_2 =[1, 1, 0, 1]^T$ label $y_2 = +1$ => $\mathbf{w}^T\mathbf{x}_2 = 1 \geq 0.5$

$\mathbf{x}_3 = [0, 0, 0, 0]^T$ label $y_3 = -1$ => $\mathbf{w}^T\mathbf{x}_3 = 0 < 0.5$

$\mathbf{x}_4 = [0, 1, 0, 0]^T$ label $y_3 = -1$ => $\mathbf{w}^T\mathbf{x}_4 = 0 < 0.5$

$\mathbf{x}_5 = [0, 0, 0, 0]^T$ label $y_3 = -1$ => $\mathbf{w}^T\mathbf{x}_5 = 0 < 0.5$

w = [0, 0, 1, 1]^T

= > **M_2 error is 0%**

Learning = finding parameters $\mathbf{w}^T = [w_1, w_2, w_3, w_4]$ such that (τ =0.5):

- $\mathbf{w}^T \mathbf{x}_i \geq 0.5$, if $y_i = +1$
- $\mathbf{w}^T \mathbf{x}_i < 0.5$, if $y_i = -1$

$\mathbf{w}^T = [w_1, w_2, w_3, w_4]$

where $\mathbf{w}^T \mathbf{x} = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$

$\mathbf{x} = [x_1, x_2, x_3, x_4]$

M_1 or M_2?

- Model M_1: **$\mathbf{x}_i$ is Red => $y_i = +1$**
  - **$\mathbf{w}^{(1)} = [1, 0, 0, 0]^T$**
  - **Error = 0%**

- Model M_2: **$\mathbf{x}_i$ is Square or Diamond => $y_i = +1$**
  - **$\mathbf{w}^{(2)} = [0, 0, 1, 1]^T$**
  - **Error = 0%**

- Which one should we choose?
  - Which one is expected to perform better on unseen (new) examples?

ML Objective

- Find a model **w** that is *simple* and that *fits the training data*.

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \ \text{Complexity}(\mathbf{w}) + \text{Error}(\mathbf{w}, \mathit{Data})$$

M_1 or M_2?

- Model M_1: $\mathbf{x}_i$ **is Red =>** $y_i = +1$
  - $\mathbf{w}^{(1)} = [1, 0, 0, 0]^T$
  - **Error = 0%**
- Model M_2: $\mathbf{x}_i$ **is Square or Diamond =>** $y_i = +1$
  - $\mathbf{w}^{(2)} = [0, 0, 1, 1]^T$
  - **Error = 0%**

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \ \text{Complexity}(\mathbf{w}) + \text{Error}(\mathbf{w}, \mathit{Data})$$

Complexity($\mathbf{w}$) = ?

- $\|\mathbf{w}\|_0$ i.e. # non-zero values
- $\|\mathbf{w}\|_1$ i.e. sum of absolute values
- $\|\mathbf{w}\|_2^2$ i.e sum of squared values

ML Objectives

- Find a model **w** that is *simple* and that *fits the training data*.

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \ \text{Complexity}(\mathbf{w}) + \text{Error}(\mathbf{w}, \mathit{Data})$$

Ridge Regression:

$$\operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

Logistic Regression:

$$\operatorname{argmin} \frac{\alpha}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} \ln p(t_n | x_n)$$

ML Objectives

Support Vector Machines:

Upper bound on the number of misclassified training examples

$$\underset{\mathbf{w}}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \xi_n$$

subject to:

$$t_n(\mathbf{w}^T\varphi(\mathbf{x}_n) + b) \geq 1 - \xi_n, \quad \forall n \in \{1, \ldots, N\}$$

$$\xi_n \geq 0$$

Bias w_0 = – Threshold τ

$\mathbf{w}^{\mathrm{T}}\mathbf{x} = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 \geq \tau$

$\Leftrightarrow w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 - \tau \geq 0$

Define the **intercept** or **bias** $w_0 = -\tau$.

$\Leftrightarrow w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_0 \geq 0$

$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 \geq 0$
where:
$\mathbf{w}^{\mathrm{T}} = [w_1\ w_2\ w_3\ w_4]$
$\mathbf{x} = [x_1\ x_2\ x_3\ x_4]$

$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} \geq 0$
where:
$\mathbf{w}^{\mathrm{T}} = [w_0\ w_1\ w_2\ w_3\ w_4]$
$\mathbf{x} = [1\ x_1\ x_2\ x_3\ x_4]$

Geometric Interpretation

- Example **x** is a feature vector **x** = [x_1 x_2 … x_K].
  - Example **x** is a point in a K-dimensional feature space.

- Parameters **w** form a vector $\mathbf{w}^T$ = [w_1 w_2 … w_K].
  - Parameters **w** are a point in a K-dimensional feature space.

- What does it mean that $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 > 0$?

Linear Discriminant Functions: Two classes (K = 2)

- Use a linear function of the input vector:

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

weight vector *bias = − threshold*

- Decision:

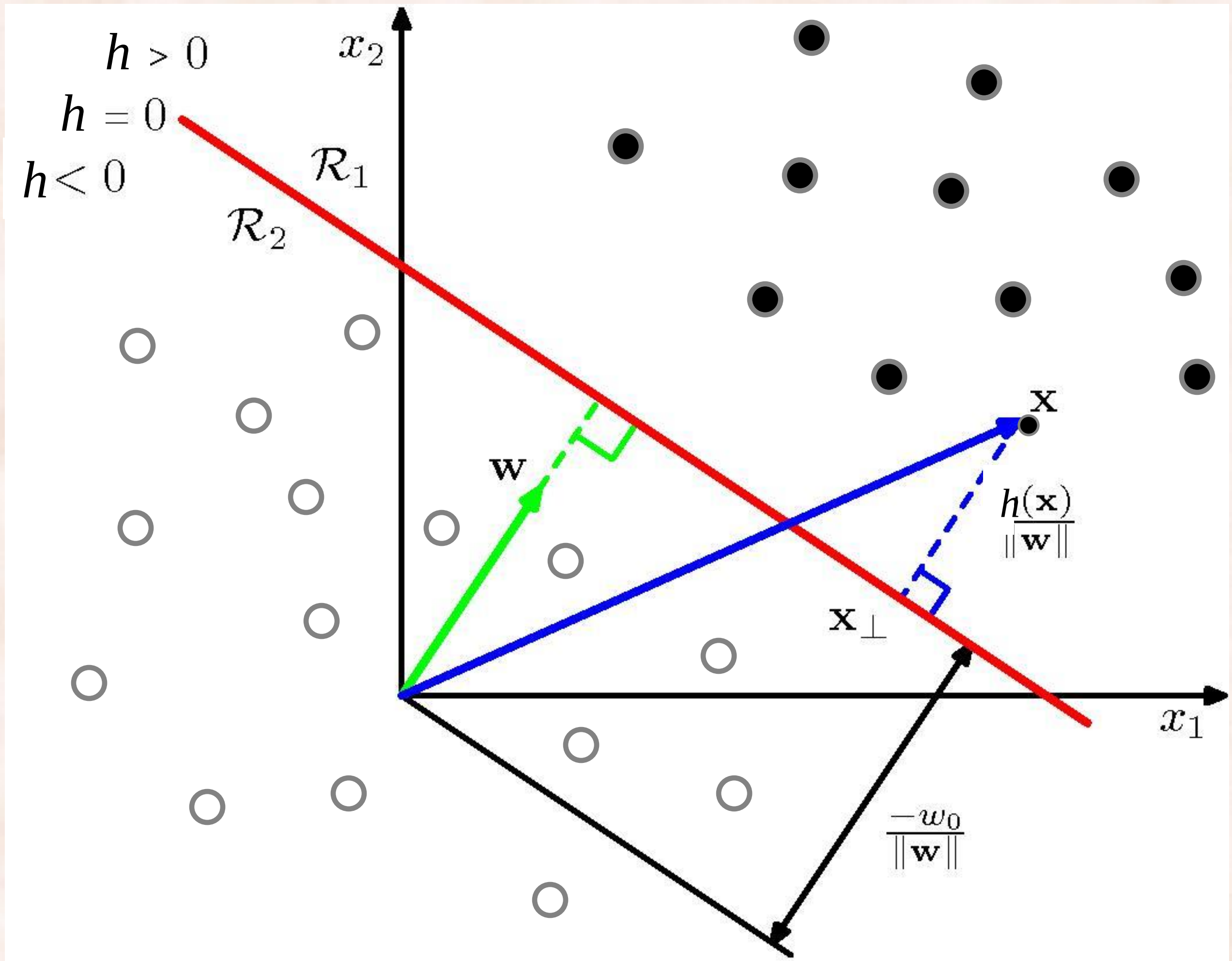
  $\mathbf{x} \in C_1$ if $h(\mathbf{x}) \geq 0$, otherwise $\mathbf{x} \in C_2$.

  $\Rightarrow$ **decision boundary** is hyperplane $h(\mathbf{x}) = 0$.

- Properties:
  - $\mathbf{w}$ is orthogonal to vectors lying within the decision surface.
  - w_0 controls the location of the decision hyperplane.

Geometric Interpretation

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = w_1x_1 + w_2x_2 + w_0$$



Linear Models for Classification

- We want to use a linear function of the feature vector:

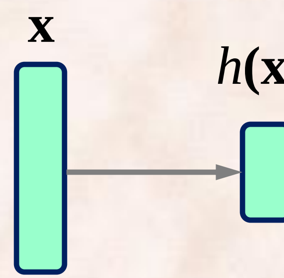
$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

- How to find **w** automatically? Use ML!
  - **Perceptron**.
  - **Logistic Regression**.

- What if the data is not linearly separable? Make it!
  - Engineer new features (LR) or use kernels (Perceptron).
  - Learn new features (**Neural Networks**).

Machine Learning (most of ML pre-2006)

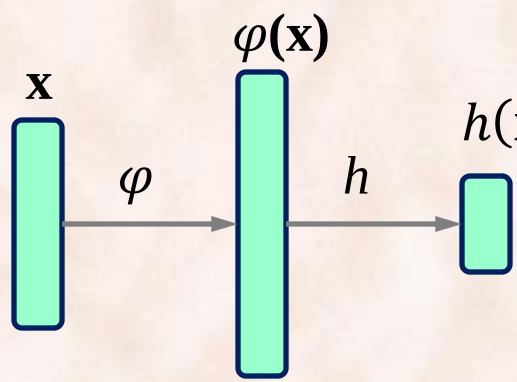
- Hope raw data **x** is linearly separable.



$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$

Use a Perceptron or LR or SVMs to learn **w**.

- Engineer features $\varphi(\mathbf{x})$, aim to make data linearly separable.

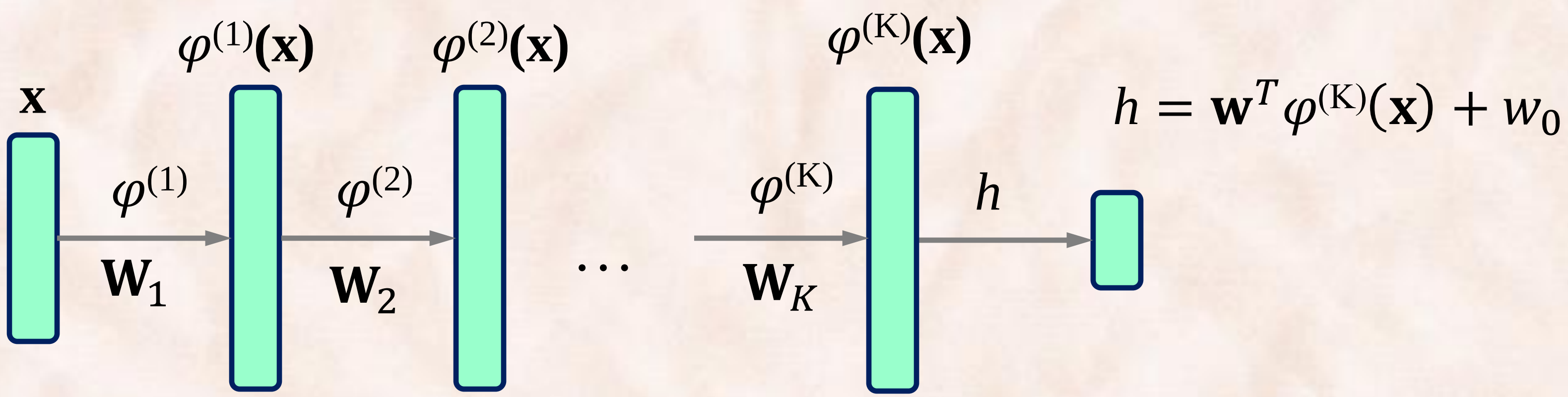


$h(\mathbf{x}) = \mathbf{w}^T\varphi(\mathbf{x}) + w_0$

Use a Perceptron or LR or SVMs to learn **w**.

Deep Learning

- A raw observation vector **x** is pre-processed and further transformed into a sequence of higher-level feature vectors $\varphi(\mathbf{x}) = [\varphi^{(1)}(\mathbf{x}), \varphi^{(2)}(\mathbf{x}), \ldots, \varphi^{(K)}(\mathbf{x})]^T$ that are **learned**.



Linear Models: $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$

- Given N training examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_N, y_N)$ where:
  - Labels $y_j \in \{-1, +1\}$.
  - Each example $\mathbf{x}_j$ is assumed to also contain a bias feature set to 1, corresponding to parameter w_0.

- Find parameter vector $\mathbf{w}$ such the model $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ *fits the training examples*:
  - $\mathbf{w}^T\mathbf{x}_n > 0$ for all positive examples $(y_n = +1)$
  - $\mathbf{w}^T\mathbf{x}_n < 0$ for all negative examples $(y_n = -1)$

The Perceptron Algorithm: Two Classes

1. **initialize** parameters $\mathbf{w} = 0$
2. **for** $n = 1 \ldots N$
3. $\quad h_n = \mathbf{w}^T \mathbf{x}_n$
4. $\quad$ **if** $h_n \geq 0$ and $y_n = -1$
5. $\qquad \mathbf{w} = \mathbf{w} - \mathbf{x}_n$
6. $\quad$ **if** $h_n \leq 0$ and $y_n = +1$
7. $\qquad \mathbf{w} = \mathbf{w} + \mathbf{x}_n$

Repeat:
a) until convergence.
b) for a number of epochs E.

What is the impact of the perceptron update on the score $\mathbf{w}^T \mathbf{x}_n$ of the misclassified example $\mathbf{x}_n$?

The Perceptron Algorithm: Two Classes

1. **initialize** parameters $\mathbf{w} = 0$
2. **for** $n = 1 \dots N$
3. $h_n = \mathbf{w}^{\mathrm{T}}\mathbf{x}_n$
4. **if** $h_n y_n \leq 0$ **then**
5. $\mathbf{w} = \mathbf{w} + y_n \mathbf{x}_n$

Repeat:

a) until convergence.

b) for a number of epochs E.

Loop invariant: **w** is a weighted sum of training vectors:

$$\mathbf{w} = \sum_n \alpha_n y_n \mathbf{x}_n \quad \Rightarrow \quad \mathbf{w}^T\mathbf{x} = \sum_n \alpha_n y_n \mathbf{x}_n^T \mathbf{x}$$

The Perceptron Algorithm: Two Classes

1. **initialize** parameters $\mathbf{w} = 0$
2. **for** $n = 1 \ldots N$
3. $\hat{y}_n = sgn(\mathbf{w}^T\mathbf{x}_n)$
4. **if** $\hat{y}_\text{n} \neq y_n$ **then**
5. $\mathbf{w} = \mathbf{w} + y_n\mathbf{x}_n$

$$sgn(h) = \begin{cases} +1 & \text{if } h > 0, \\ 0 & \text{if } h = 0, \\ -1 & \text{if } h < 0 \end{cases}$$

Repeat:

a) until convergence.

b) for a number of epochs E.

Theorem [Rosenblatt, 1962]:

If the training dataset is linearly separable, the perceptron learning algorithm is guaranteed to find a solution in a finite number of steps.

- see Theorem 1 (Block, Novikoff) in [Freund & Schapire, 1999].

The Perceptron Algorithm: Two Classes

$sgn(h) = +1$ if $h > 0$,
0 if $h = 0$,
-1 if $h < 0$

initialize parameters $\mathbf{w} = 0$

for epoch $e = 1 \ldots E$

  mistakes = 0

  for example $n = 1 \ldots N$

    $\hat{y}_n = sgn(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)$

    if $\hat{y}_n \neq y_n$ **then**

      $\mathbf{w} = \mathbf{w} + y_n\mathbf{x}_n$

      mistakes = mistakes + 1

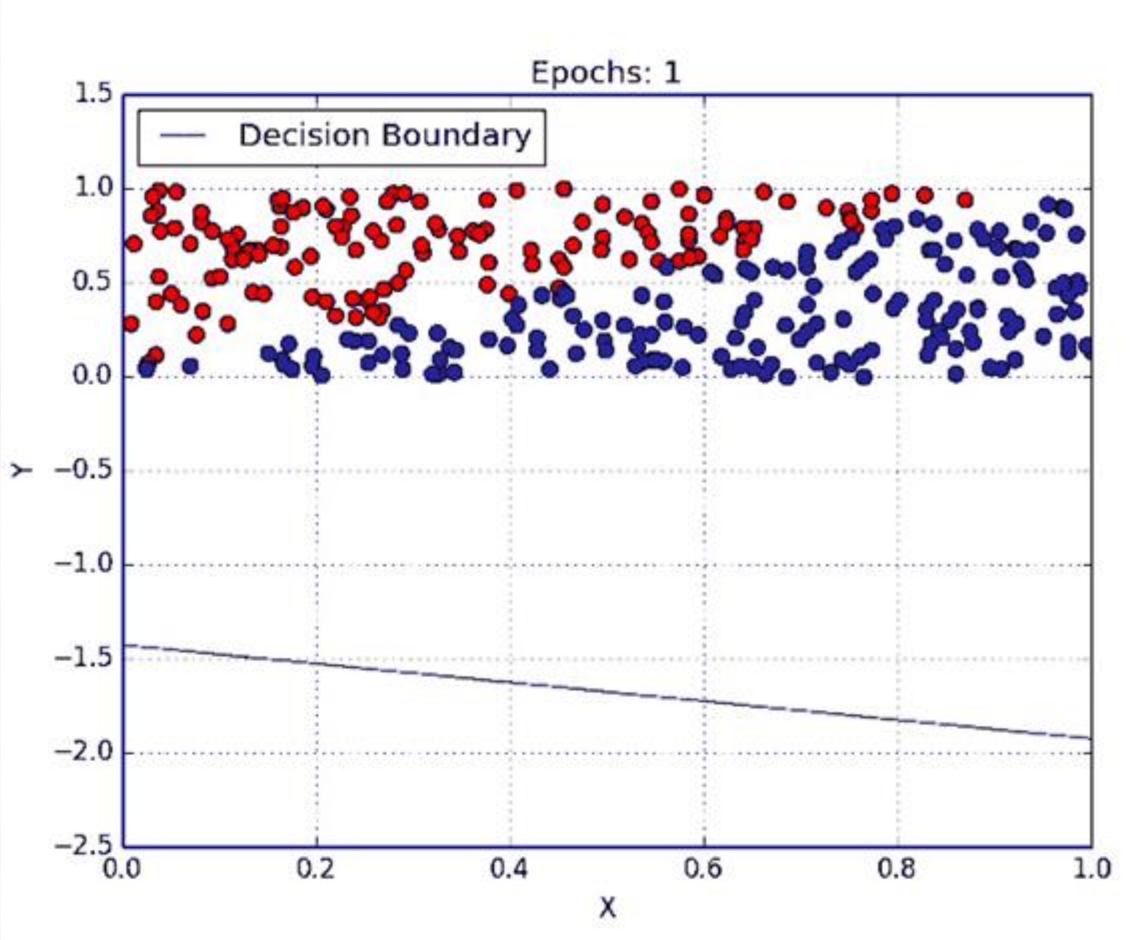
  if mistakes = 0

    break

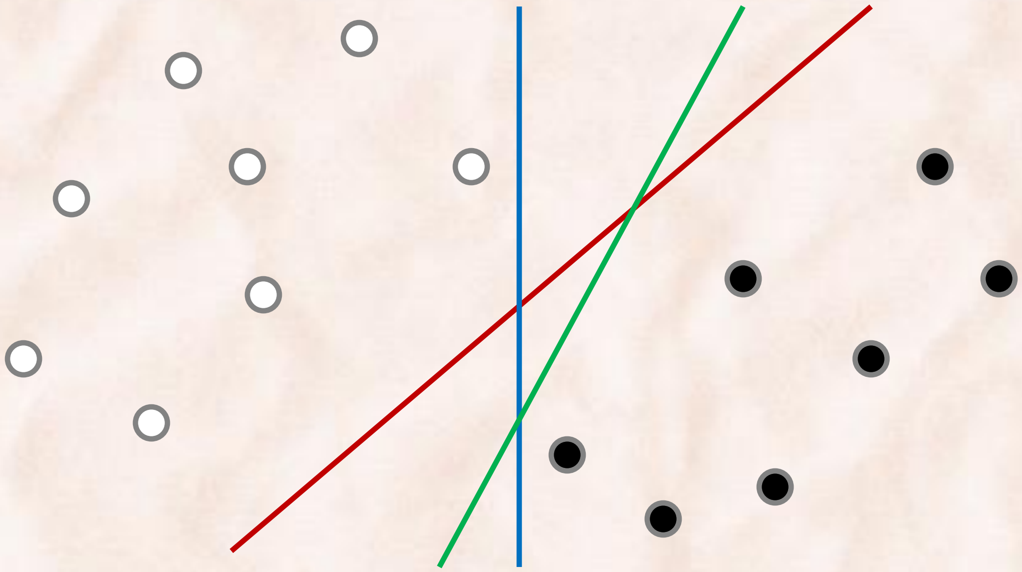
Converged!

1 epoch = one pass over all training examples.

The Perceptron Algorithm

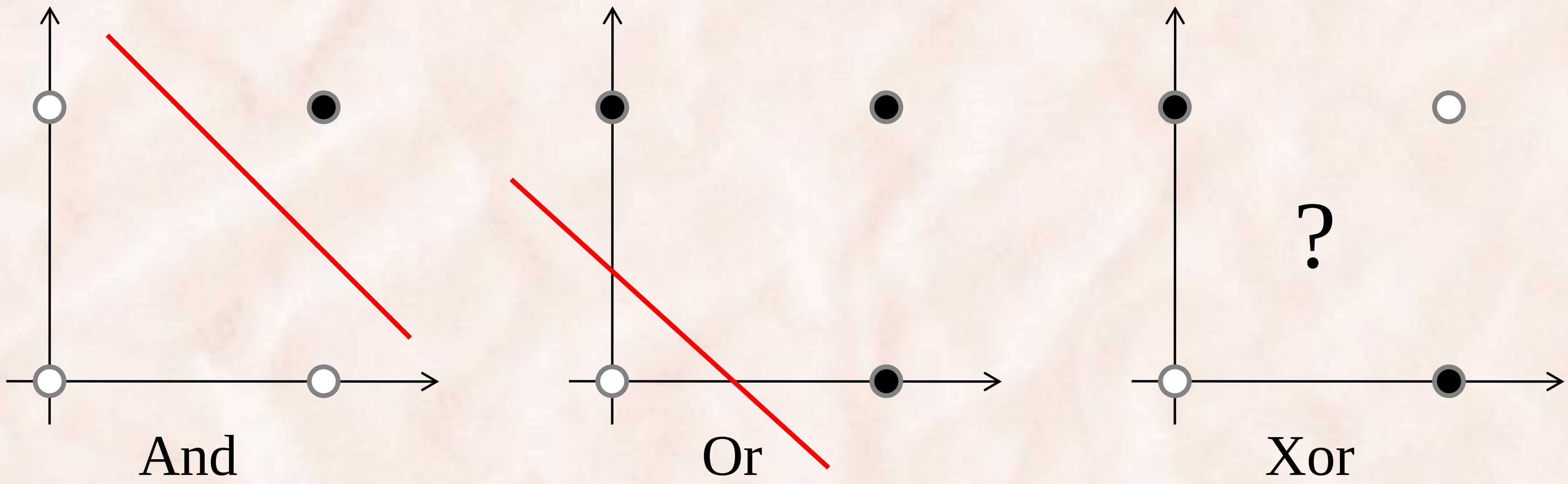


Classifiers & Margin



- Which classifier has the smallest generalization error?
  - The one that maximizes the margin [Computational Learning Theory]
    - **margin** = the distance between the decision boundary and the closest sample.

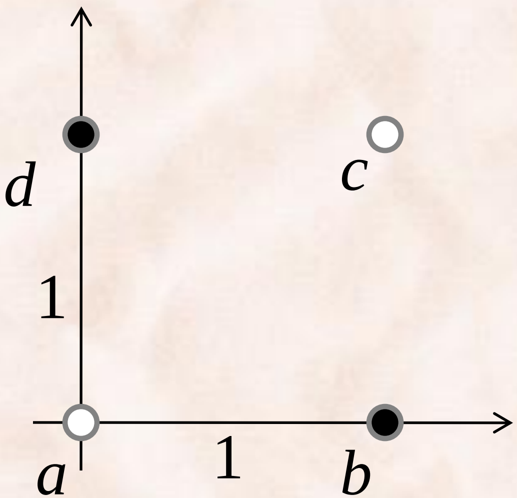
Linear vs. Non-linear Classifiers

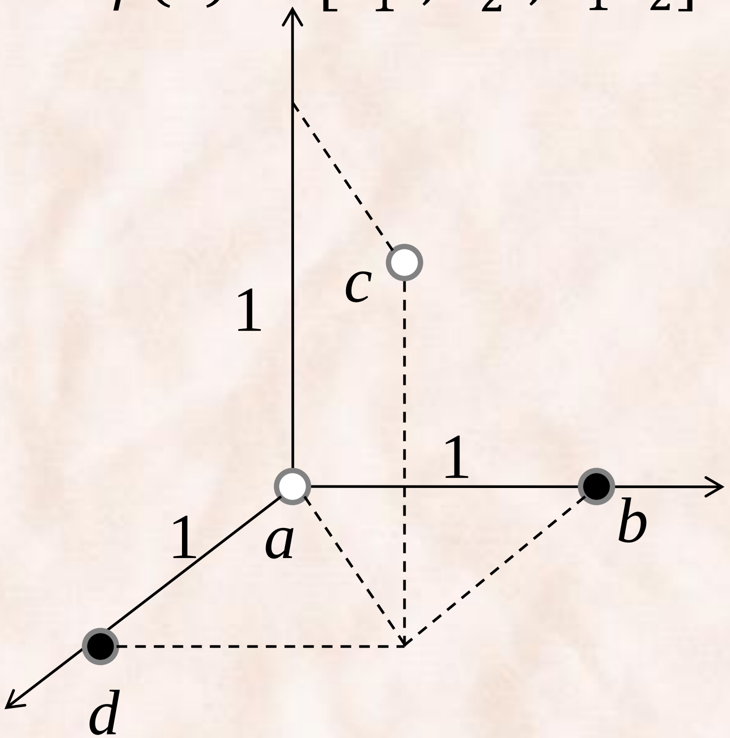


$$\left.\begin{array}{r}\varphi(\mathbf{x}) = [1, x_1, x_2]^T \\ \mathbf{w} = [w_0, w_1, w_2]^T\end{array}\right\} \Rightarrow \mathbf{w}^T\varphi(\mathbf{x}) = [w_1, w_2]^T[x_1, x_2] + w_0$$

ML with Manually Engineered Features

$\mathbf{x} = [x_1, x_2]$ $\varphi(\mathbf{x}) = [x_1, x_2, x_1x_2]$





ML Concepts & Notation

- A (labeled) example ($\mathbf{x}$, y) consists of:
  - Instance / observation / raw feature vector $\mathbf{x}$.
  - Label y.

- Examples:
  1. Image classification in **Computer Vision** (CV):

     instance $\mathbf{x}$ = ?

     label y = ?

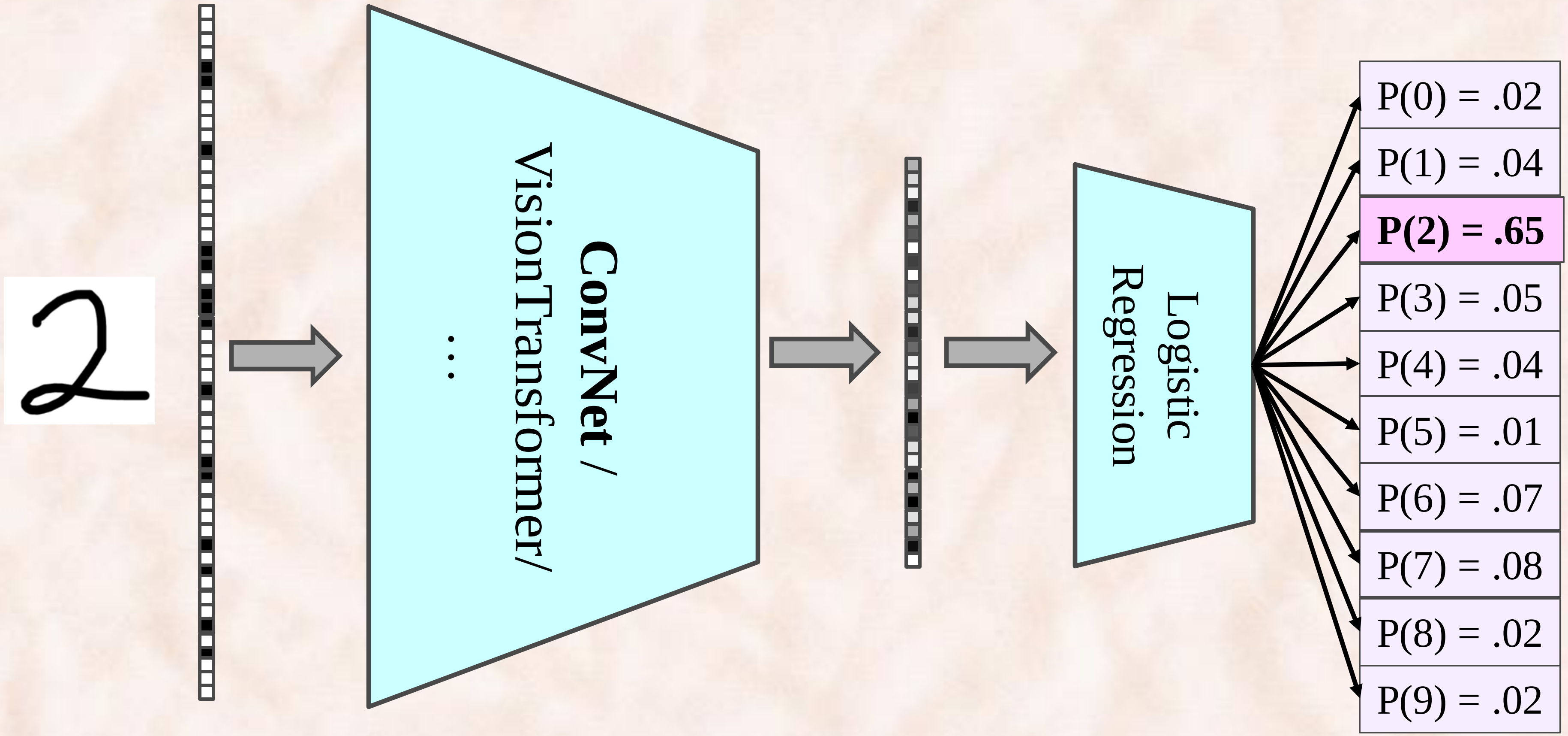
  2. Language Modeling (LM) in **Natural Language Processing** (NLP):
     - "I went to the Data Science International Summer ___?___"

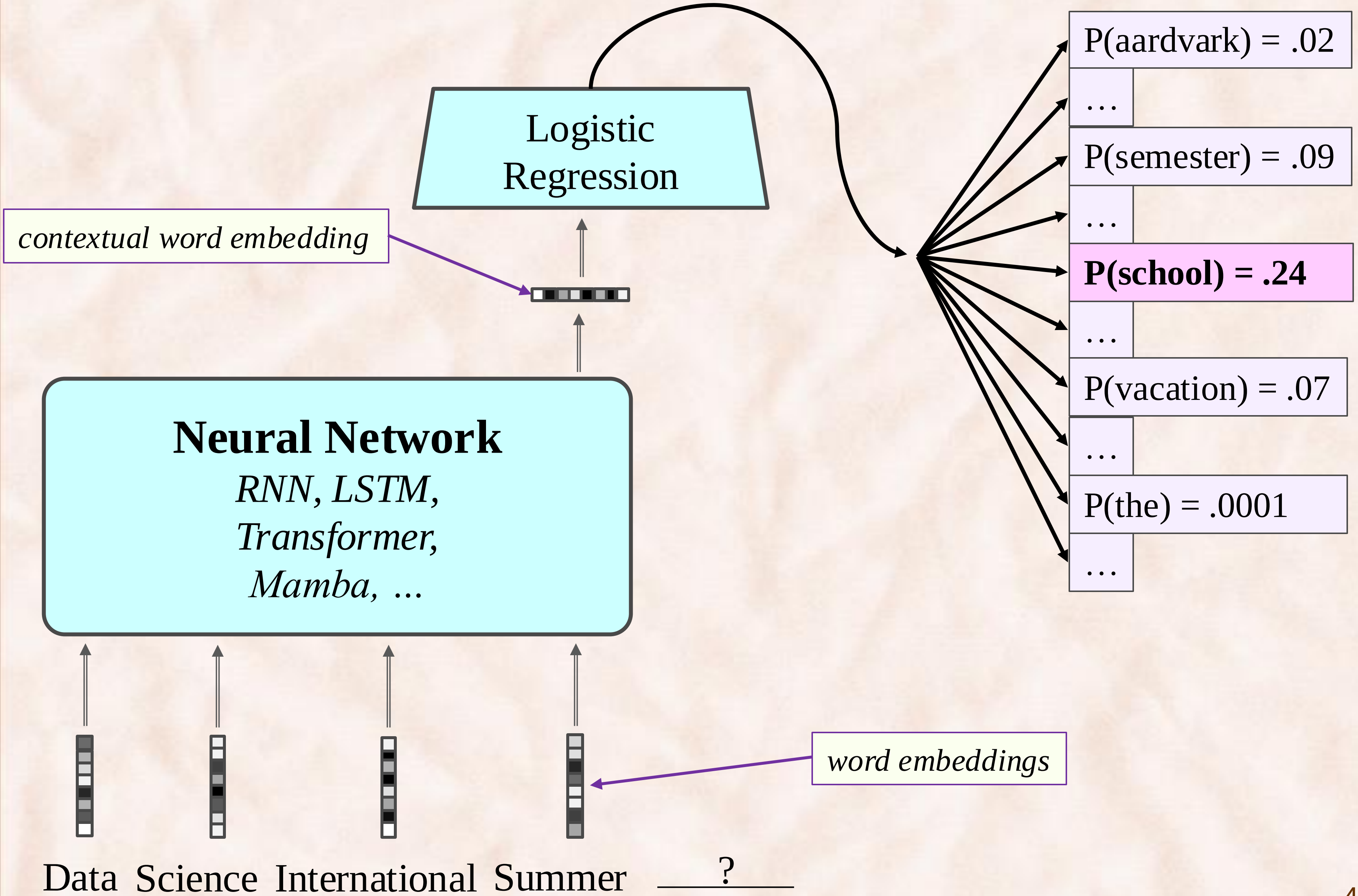
     instance $\mathbf{x}$ = ?

     label y = ?

     eat … school … camp

Image classification (CV)




Logistic Regression
contextual word embedding
Neural Network
RNN, LSTM, Transformer, Mamba, …
P(aardvark) = .02
…
P(semester) = .09
…
P(school) = .24
…
P(vacation) = .07
…
P(the) = .0001
…
word embeddings
Data Science International Summer ___?___

ML Concepts & Notation

- A training dataset is a set of (training) examples $(\mathbf{x}_1,t_1)$, $(\mathbf{x}_2,t_2)$, … $(\mathbf{x}_N,t_N)$:
  - The data matrix X contains all instance vectors $\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_N$ row-wise.
  - The label vector $\mathbf{y} = [y_1, y_2, \ldots, y_N]$.

- A test dataset is a set of (test) examples $(\mathbf{x}_{N+1},y_{N+1})$, …, $(\mathbf{x}_{N+\mathrm{M}},y_{N+\mathrm{M}})$:
  - **Must be unseen, i.e. new, i.e. different from the training examples**!

- A development dataset …

ML Concepts & Notation

- There is a function f that maps an instance $\mathbf{x}$ to its label $y = f(\mathbf{x})$.
  - f is unknown / not given.
  - But we observe samples from f : $(\mathbf{x}_1, y_1=f(\mathbf{x}_1))$, $(\mathbf{x}_2,y_2)$, … $(\mathbf{x}_N,y_N)$.

- Learning means finding a model h that maps an instance $\mathbf{x}$ to a label $h(\mathbf{x}) \approx f(\mathbf{x})$, i.e. close to the true label of $\mathbf{x}$.
  - Machine learning = finding a model h that approximates well the unknown function f.
  - Machine learning = function approximation.

ML Concepts & Notation

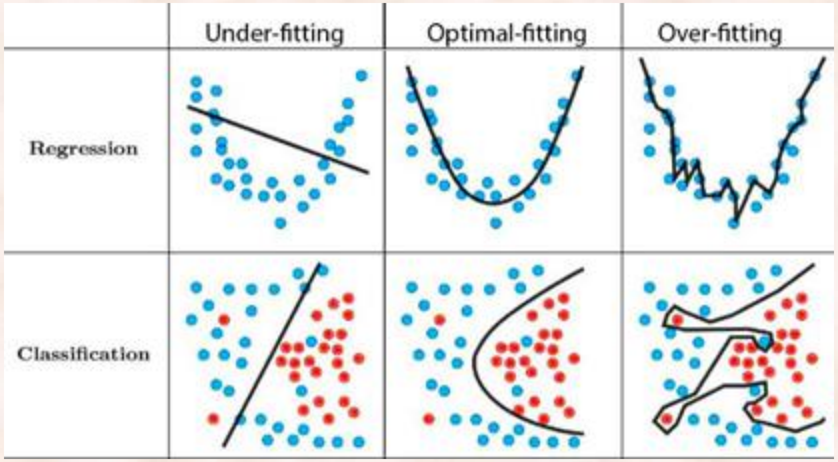
- Machine learning is inductive:
  - Inductive hypothesis: if a model performs well on training examples, it is expected to also perform well on unseen (test) examples.
    - Assume within-distribution test examples.

- The model h is often specified through a set of parameters $\mathbf{w}$:
  - $\mathbf{x}$ is mapped by the model to $h(\mathbf{x}, \mathbf{w})$.

- The objective function $J(\mathbf{w})$ captures how poorly the model does on the training dataset:
  - Want to find $\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} J(\mathbf{w})$
    - Machine learning = optimization.

Fitting vs. Generalization

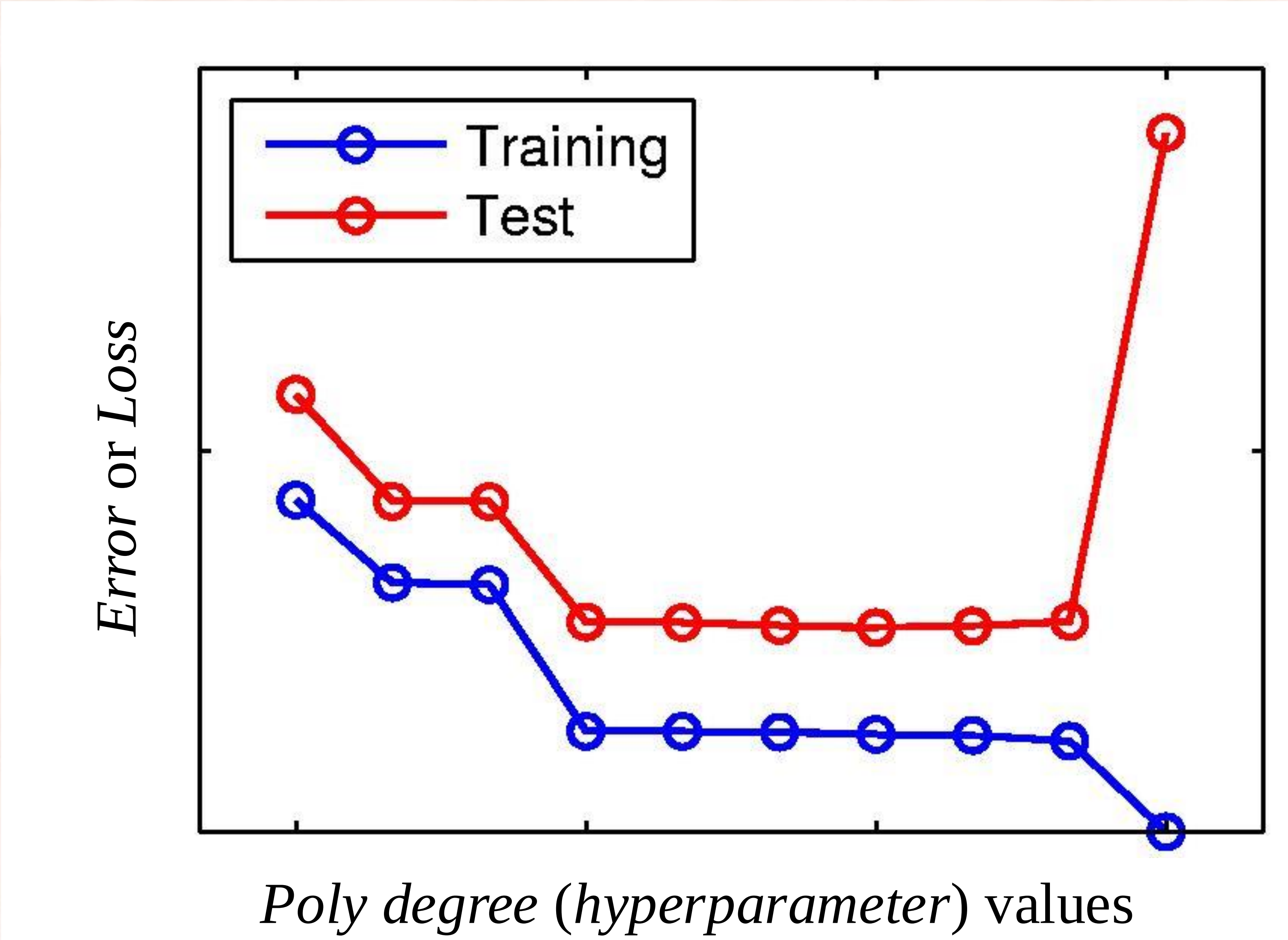
- Fitting performance = how well the model does on training examples.
- Generalization performance = how well the model does on unseen (test) examples.
- We prefer finding patterns to memorizing examples!
  - **Overfitting**:
    - Add Regularization.
  - **Underfitting**:
    - Increase Capacity.



Underfitting vs. Overfitting

- **Underfitting** = model does not do well on training data:
  - Low capacity (too few params) or Training issues (too little training).
- **Overfitting** = model does well on training, poorly on test.
  - Can be mitigated by **tuning hyper-parameters**.
    - **Perceptron**: E (number of epochs).
    - **Logistic regression**: λ (strength of L_2 regularization).
    - **Neural networks**: number of layers, number of neurons on each layer, number of CNN filters, λ, dropout rate, gradient descent hyper-parameters (momentum, learning rate cooling schedule), number of epochs, …

Overfitting with Polynomail Curve Fitting



Regularization = Any Method that Alleviates Overfitting

- **Parameter norm penalties** (strength λ of L_1 or L_2 term).
- Dataset augmentation.
- **Dropout** (dropout rate)
- Ensembles.
- Semi-supervised learning.
- **Early stopping** (limit number of epochs).
- Noise robustness.
- Sparse representations.
- Adversarial training.

Math and Machine Learning

- Formulating ML algorithms and understanding their basic behavior requires basic mathematical concepts.
  - **Linear algebra**.
  - **Calculus**.
  - **Statistics**.

- Basic math concepts so far:
  - Vector spaces:
    - Vectors, dot-products, L1 and L2 norms.
    - Orthogonal vectors, hyperplanes.
  - Functions, optimization problems.

Math and Machine Learning

- Basic math concepts in this course:
  - Linear Algebra:
  - Calculus:
  - Statistics:

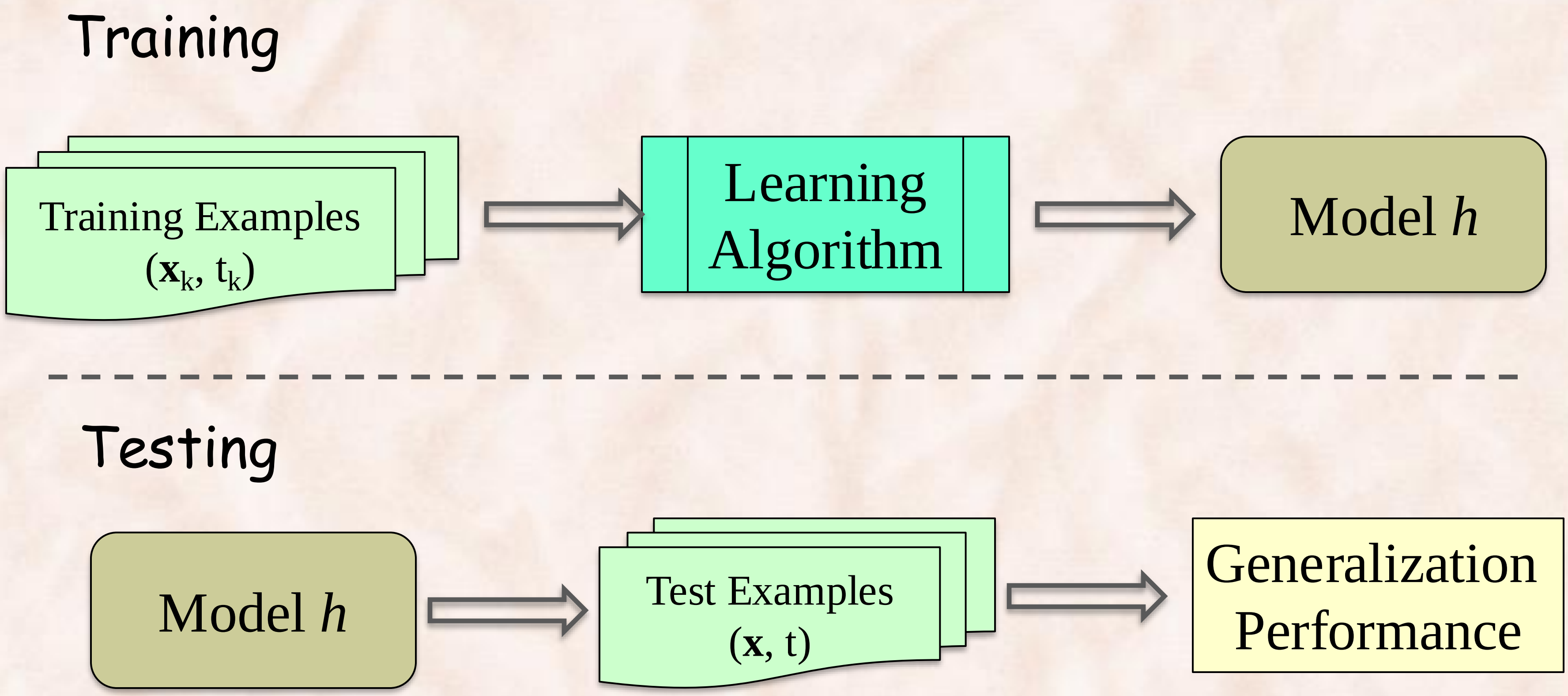
Mathematics for ML and Data Science

- Coursera has a really gentle math introduction for ML, organized into a sequence of 3 courses:

  https://www.coursera.org/specializations/mathematics-for-machine-learning-and-data-science#courses

  - Click on "Linear algebra for ML and Data Science" link.
  - Click on "Enroll for free", then click on the small "Audit the course" link in the popup window to see the videos for free.

Supervised Learning



Features

- Learning = finding parameters $\mathbf{w} = [w_1, w_2, w_3, w_4]$ and τ such that:

$$\mathbf{w}^T\varphi(x_i) \geq \tau, \text{ if } y_i = +1$$

$$\mathbf{w}^T\varphi(x_i) < \tau, \text{ if } y_i = -1$$

$$\text{where } \mathbf{w}^T\varphi(x) = w_1\times\varphi_1(x) + w_2\times\varphi_2(x) + w_3\times\varphi_3(x) + w_4\times\varphi_4(x)$$

Red? Quad? Square? Diamond?

Where do these features come from?

Object Recognition: Cats





Pixels as Features?

$\varphi(x)$ = [25, 63, 125, 32, 84, 257, ..., 13,
27, 39, 8, 213, 107, 54, 73, ..., 91,
..., 9,
..., 28,
93, 44, 69, 85, 68, 54, 87, ..., 11,
117, 59, 117, 210, 177, 54, 72, ...]^T

Poor recognition accuracy!

- Learning = finding parameters $\mathbf{w} = [w_1, w_2, w_3, \dots w_k]^T$ such that:

  $\mathbf{w}^T\varphi(x_i) \geq \tau$, if $y_i = +1$ (cat)

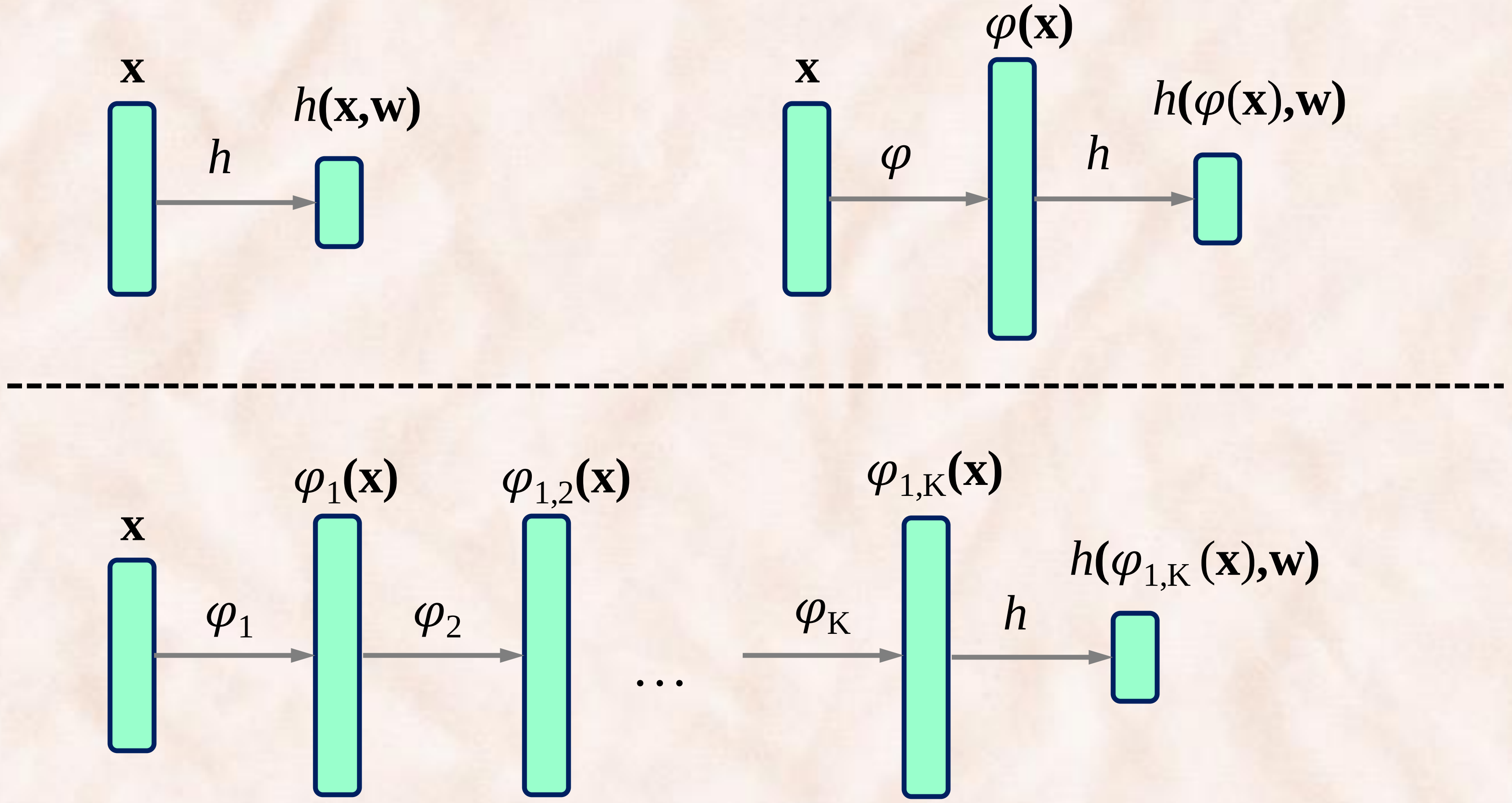
  $\mathbf{w}^T\varphi(x_i) < \tau$, if $y_i = -1$ (other)

  where $\mathbf{w}^T\varphi(x) = w_1 \times \varphi_1(x) + w_2 \times \varphi_2(x) + w_3 \times \varphi_3(x) + \dots w_k \times \varphi_k(x)$

ML Concepts & Notation

- Often, a raw observation $\mathbf{x}$ is pre-processed and further transformed into a feature vector $\varphi(\mathbf{x}) = [\varphi_1(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_K(\mathbf{x})]^T$.
  - Where do the features φ_k come from?
    - Feature engineering, e.g. in polynomial curve fitting:
      - manual, can be time consuming (e.g. SIFT).
    - (Self-supervised) feature learning, e.g. in modern computer vision:
      - automatic, used in deep learning models.

Machine Learning vs. Deep Learning



What is Machine Learning?

- **Machine Learning** = constructing computer programs that *automatically improve with experience*:
  - Supervised Learning i.e. learning from labeled examples:
    - Classification
    - Regression
  - Unsupervised Learning i.e. learning from unlabeled examples:
    - Clustering.
    - Dimensionality reduction (visualization).
    - Density estimation.
  - Reinforcement Learning i.e. learning with delayed feedback.

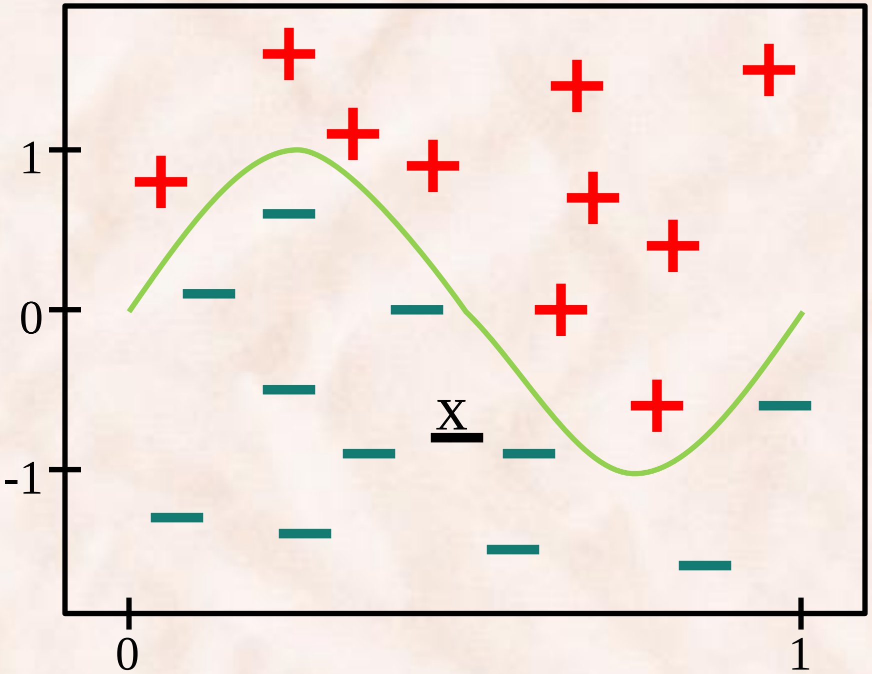
Supervised Learning

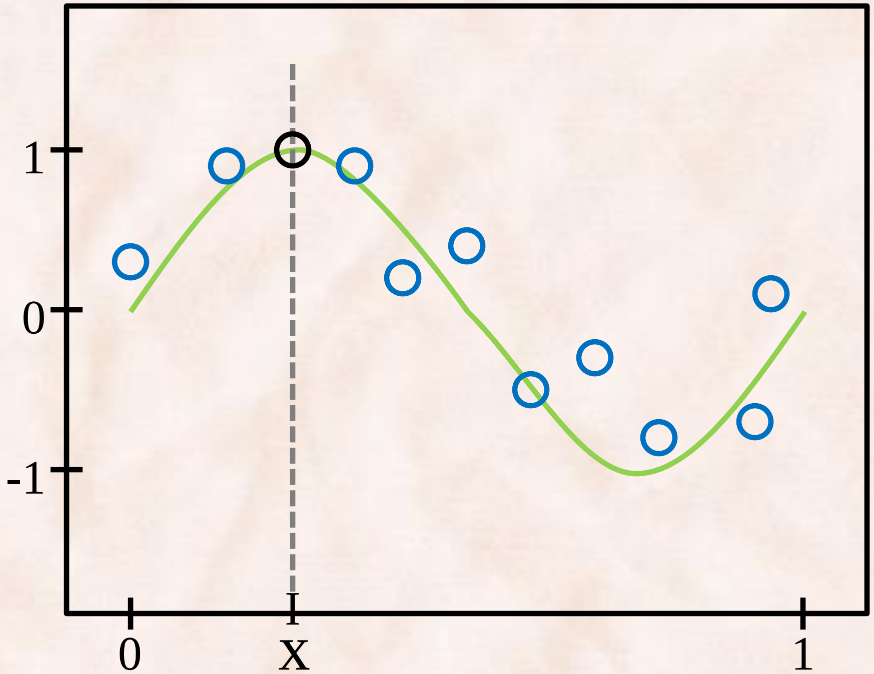
- Task = learn a function $f : X \rightarrow T$ that maps input instances $\mathbf{x} \in X$ to output targets $y \in Y$:
  - **Classification**:
    - The output $y \in Y$ is one of a finite set of discrete categories.
  - **Regression**:
    - The output $y \in Y$ is continuous, or has a continuous component.

- Supervision = set of training examples:

  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)$

Classification vs. Regression





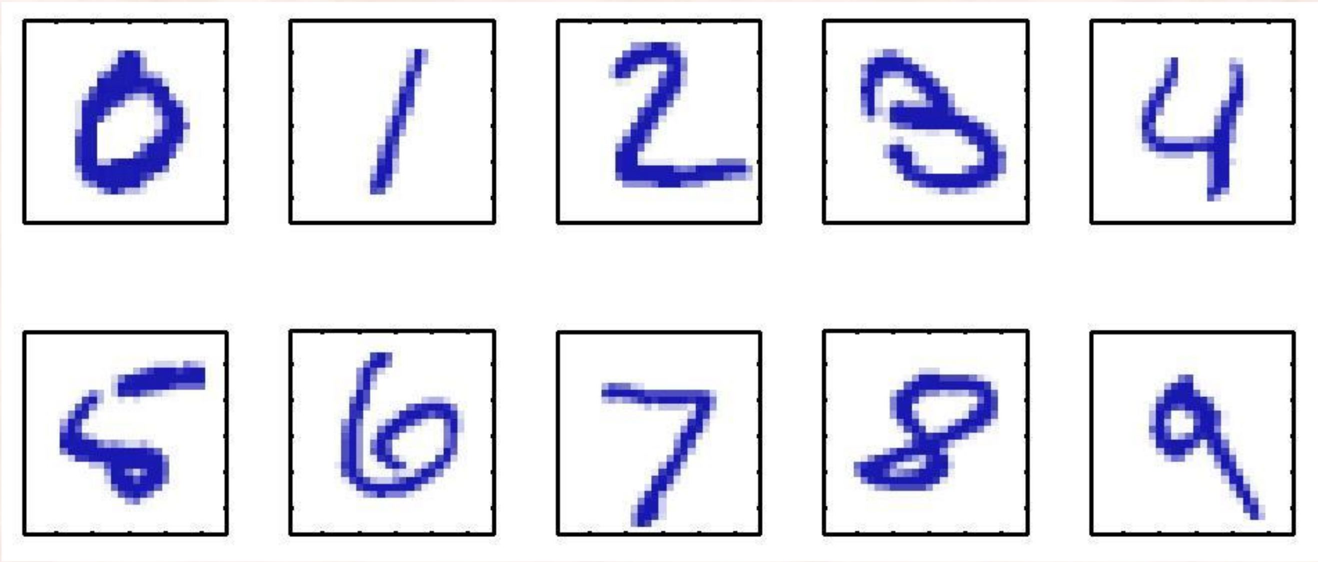
Classification: Junk Email Filtering

[Sahami, Dumais & Heckerman, AAAI'98]

From: Tammy Jordan jordant@oak.cats.ohiou.edu **Subject: Spring 2015 Course** **-------------------------------------** CS690: Machine Learning Instructor: Razvan Bunescu Email: bunescu@ohio.edu Time and Location: Tue, Thu 9:00 AM , ARC 101 Website: http://ace.cs.ohio.edu/~razvan/courses/ml6830 Course description: Machine Learning is concerned with the design and analysis of algorithms that enable computers to automatically find patterns in the data. This introductory course will give an overview …	**From: UK National Lottery** **edreyes@uknational.co.uk** **Subject: Award Winning Notice** **-------------------------------------** **UK NATIONAL LOTTERY. GOVERNMENT ACCREDITED LICENSED LOTTERY. REGISTERED UNDER THE UNITED KINGDOM DATA PROTECTION ACT;** **We happily announce to you the draws of (UK NATIONAL LOTTERY PROMOTION) International programs held in London , England Your email address attached to ticket number :3456 with serial number :7576/06 drew the lucky number 4-2-274, which subsequently won you the lottery in the first category …**

- Email filtering:
  - Provide emails labeled as *{Spam, Ham}*.
  - Train *Naïve Bayes* model to discriminate between the two.

Classification: Handwritten Zip Code Recognition [Le Cun et al., Neural Computation '89]



- Handwritten digit recognition:
  - Provide images of handwritten digits, labeled as {0, 1, …, 9}.
  - Train *Convolutional Neural Network* model to recognize digits.

Classification: Medical Diagnosis

[Krishnapuram et al., GENSIPS'02]

- Cancer diagnosis from gene expression signatures:
  - Create database of gene expression profiles (X) from tissues of known cancer status (Y):
    - Human accute leukemia dataset:
      - http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi
    - Colon cancer microarray data:
      - http://microarray.princeton.edu/oncology
  - Train *Logistic Regression* / *SVM* / *RVM* model to classify the gene expression of a tissue of unknown cancer status.

Regression: Examples

1. Stock market, oil price, GDP, income prediction:
   - Use the current stock market conditions ($x \in X$) to predict tomorrow's value of a particular stock ($y \in Y$).

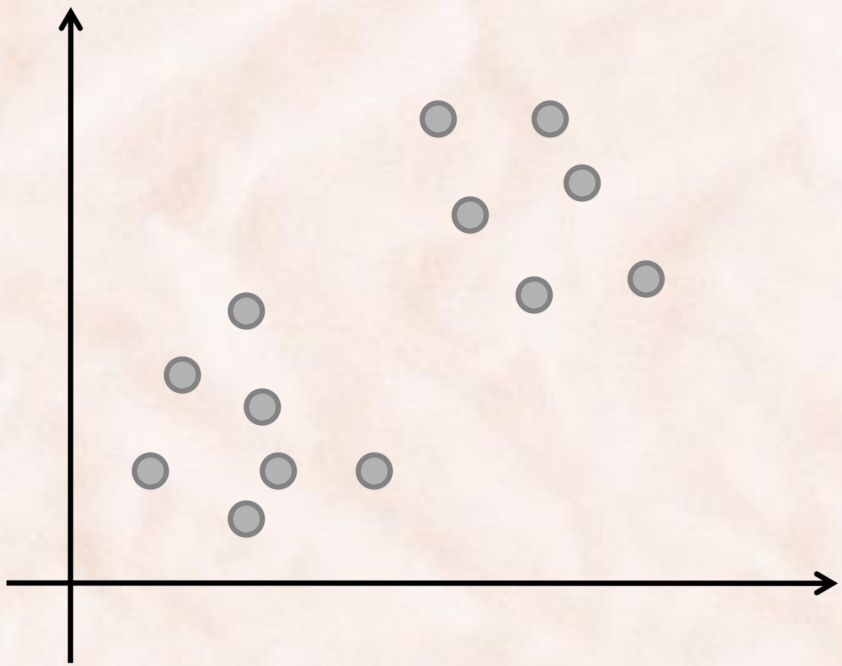
2. Blood glucose level prediction.

3. Chemical processes:
   - Predict the yield in a chemical process based on the concentrations of reactants, temperature and pressure.

- Algorithms:
  - *Linear Regression*, *Neural Networks*, *Support Vector Machines*, …

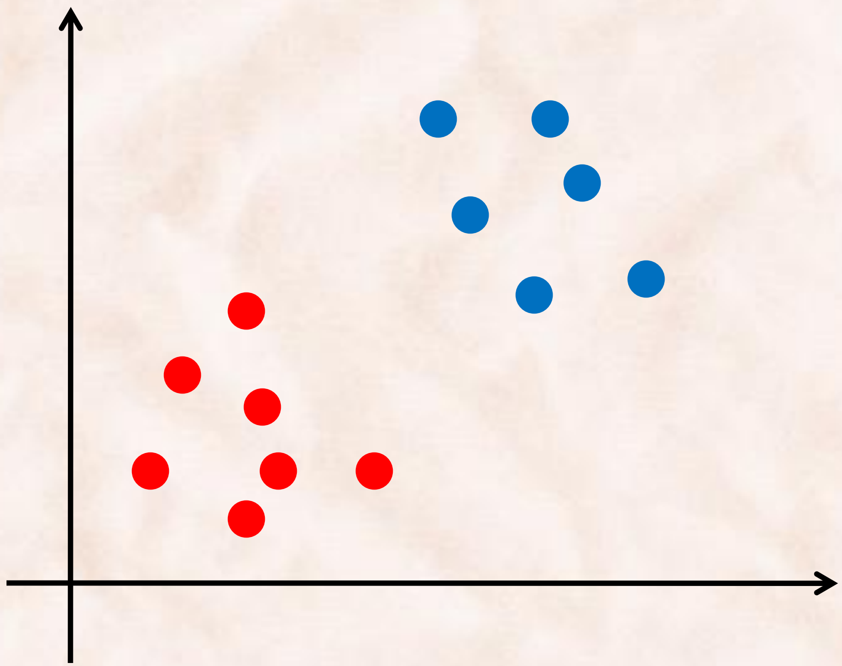
Unsupervised Learning: Clustering

- Partition unlabeled examples into disjoint clusters such that:
  - Examples in the same cluster are similar.
  - Examples in different clusters are different.



Unsupervised Learning: Clustering

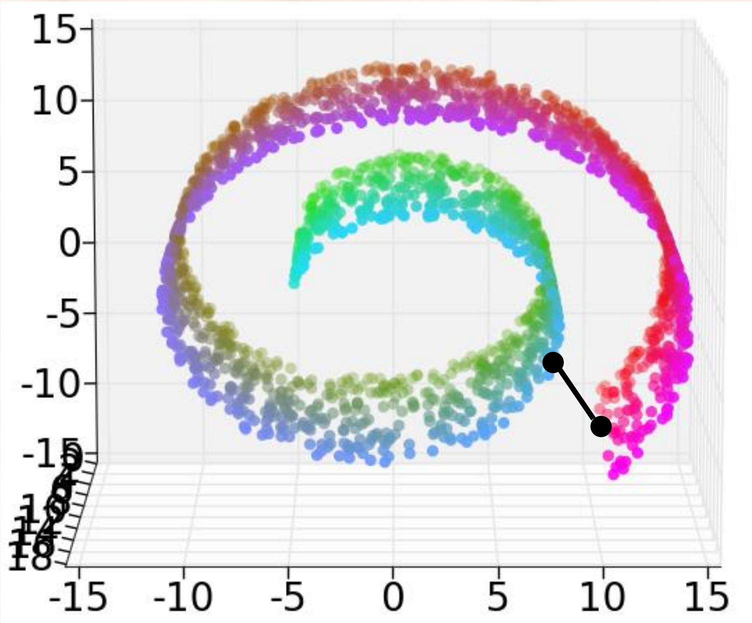
- Partition unlabeled examples into disjoint clusters such that:
  - Examples in the same cluster are similar.
  - Examples in different clusters are different.

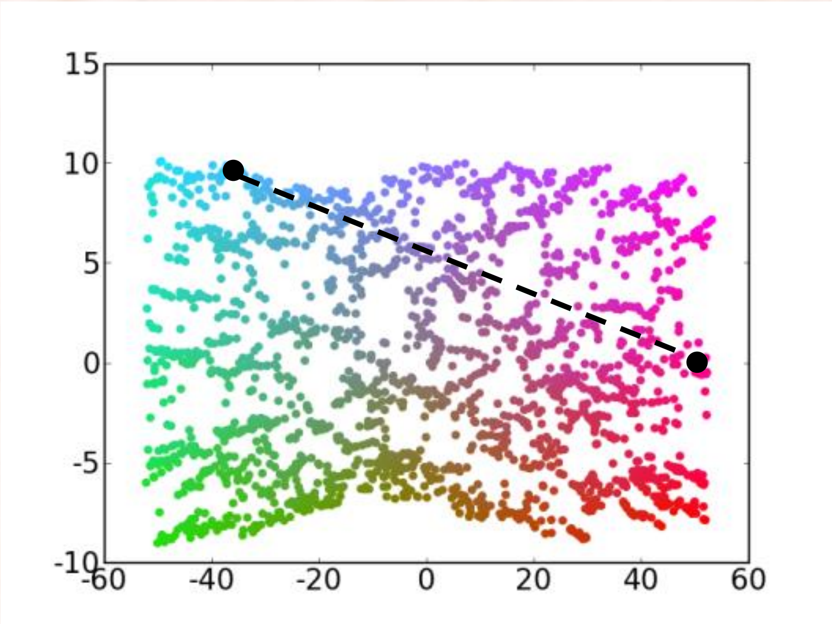


- k-Means, need to provide:
  - number of clusters (*k* = 2)
  - similarity measure (Euclidean)

Unsupervised Learning: Dimensionality Reduction

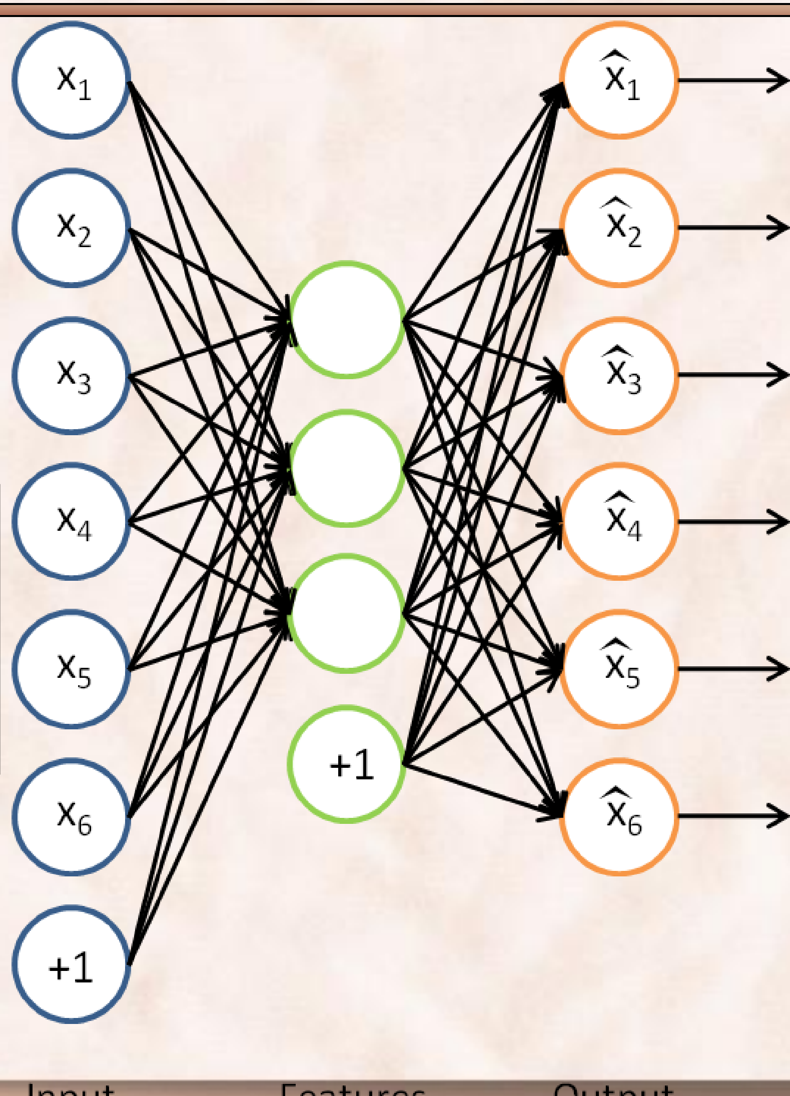
- Manifold Learning:
  - Data lies on a low-dimensional manifold embedded in a high-dimensional space.
  - Useful for *feature extraction* and *visualization*.





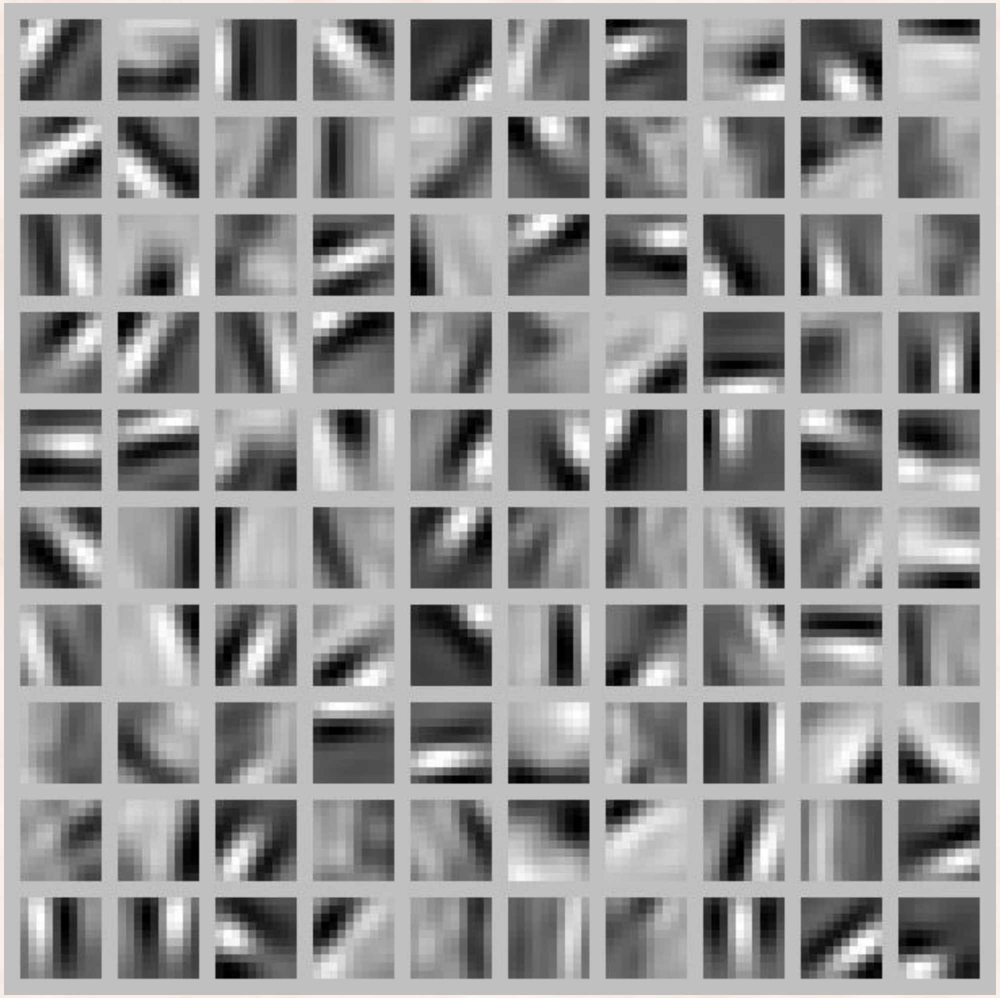
Self-supervised Feature Learning: Auto-encoders

[25, 63, 125, 32, 84, 257, ..., 13,
27, 39, 8, 213, 107, 54, 73, ..., 91
67, 59, 72, 33, 112, 54, 35, ..., 9
18, 37, 18, 142, 162, 54, 53, ..., 28
93, 44, 69, 85, 68, 54, 87, ..., 11,
117, 59, 117, 210, 177, 54, 72, ...]

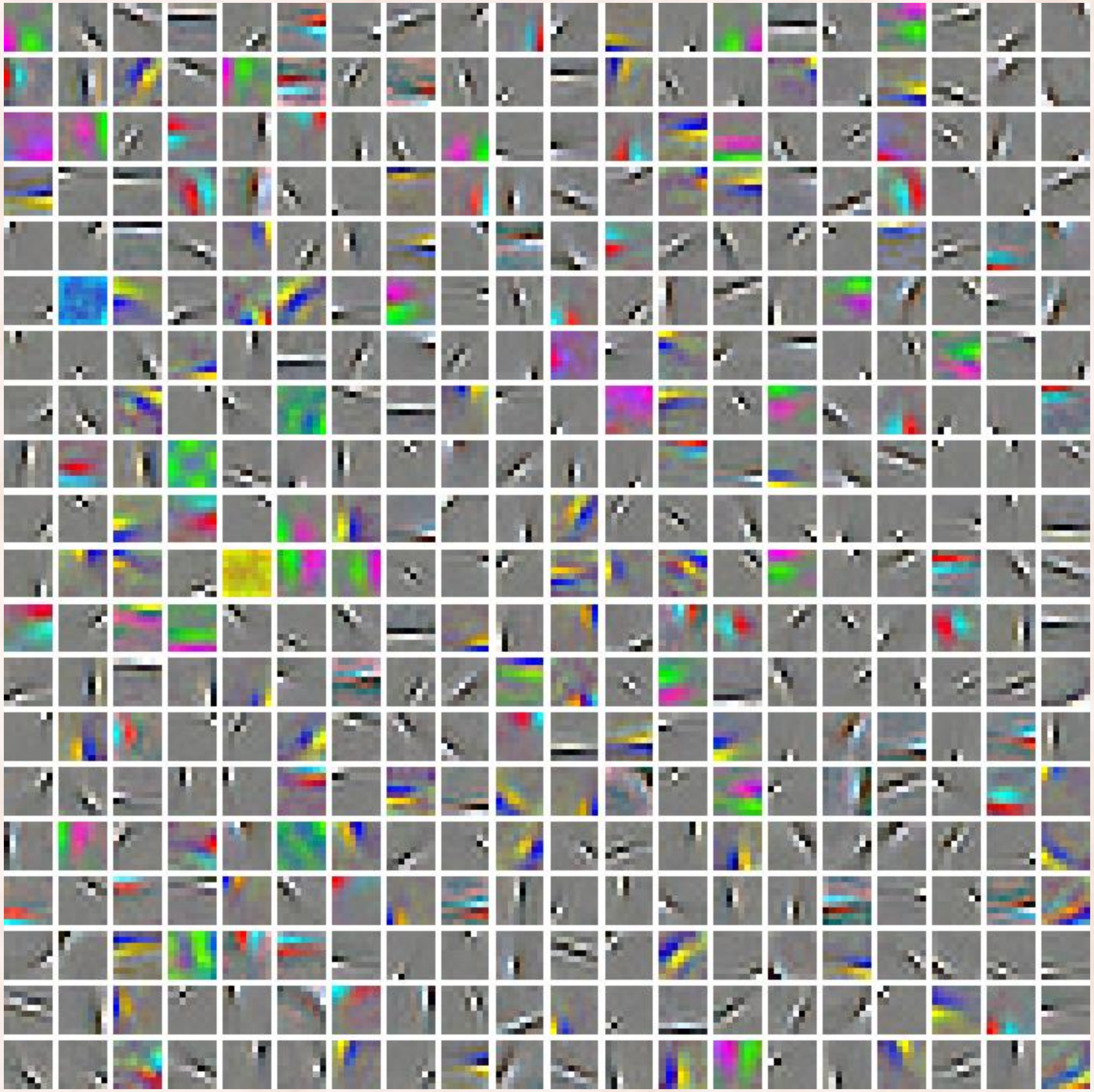


[25, 63, 125, 32, 84, 257, ..., 13,
27, 39, 8, 213, 107, 54, 73, ..., 91
67, 59, 72, 33, 112, 54, 35, ..., 9
18, 37, 18, 142, 162, 54, 53, ..., 28
93, 44, 69, 85, 68, 54, 87, ..., 11,
117, 59, 117, 210, 177, 54, 72, ...]

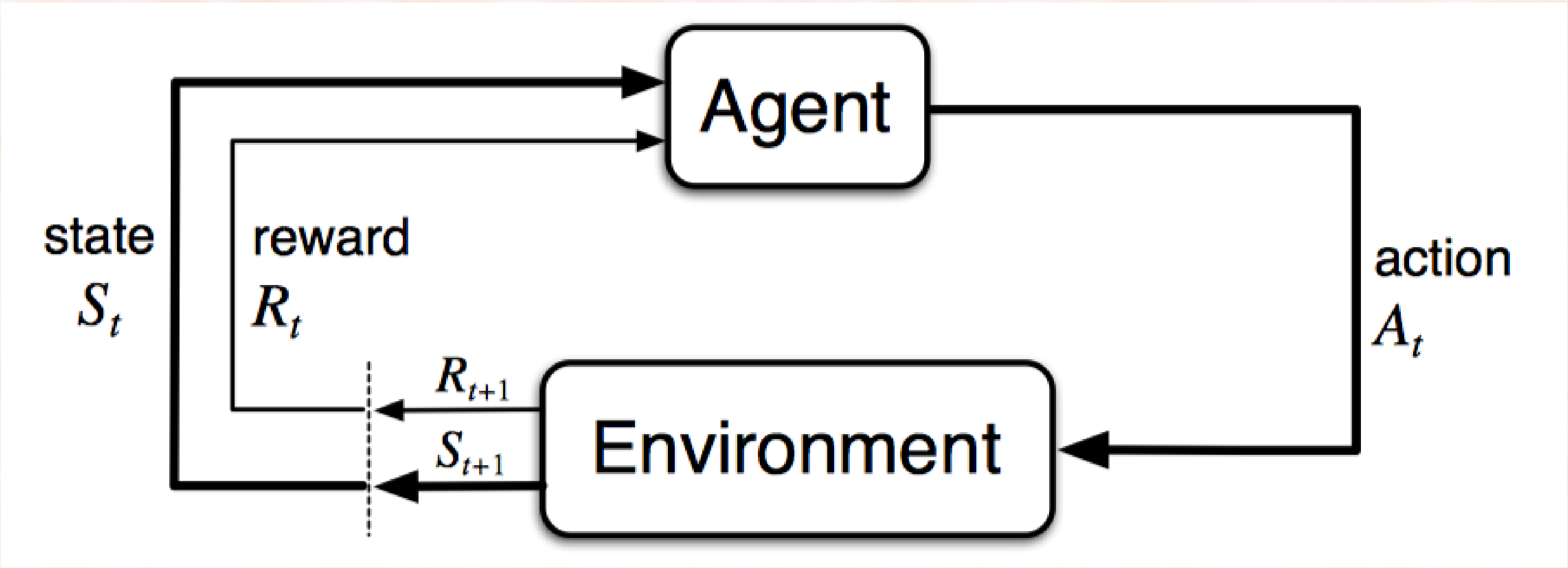
Learned Features (Representations)

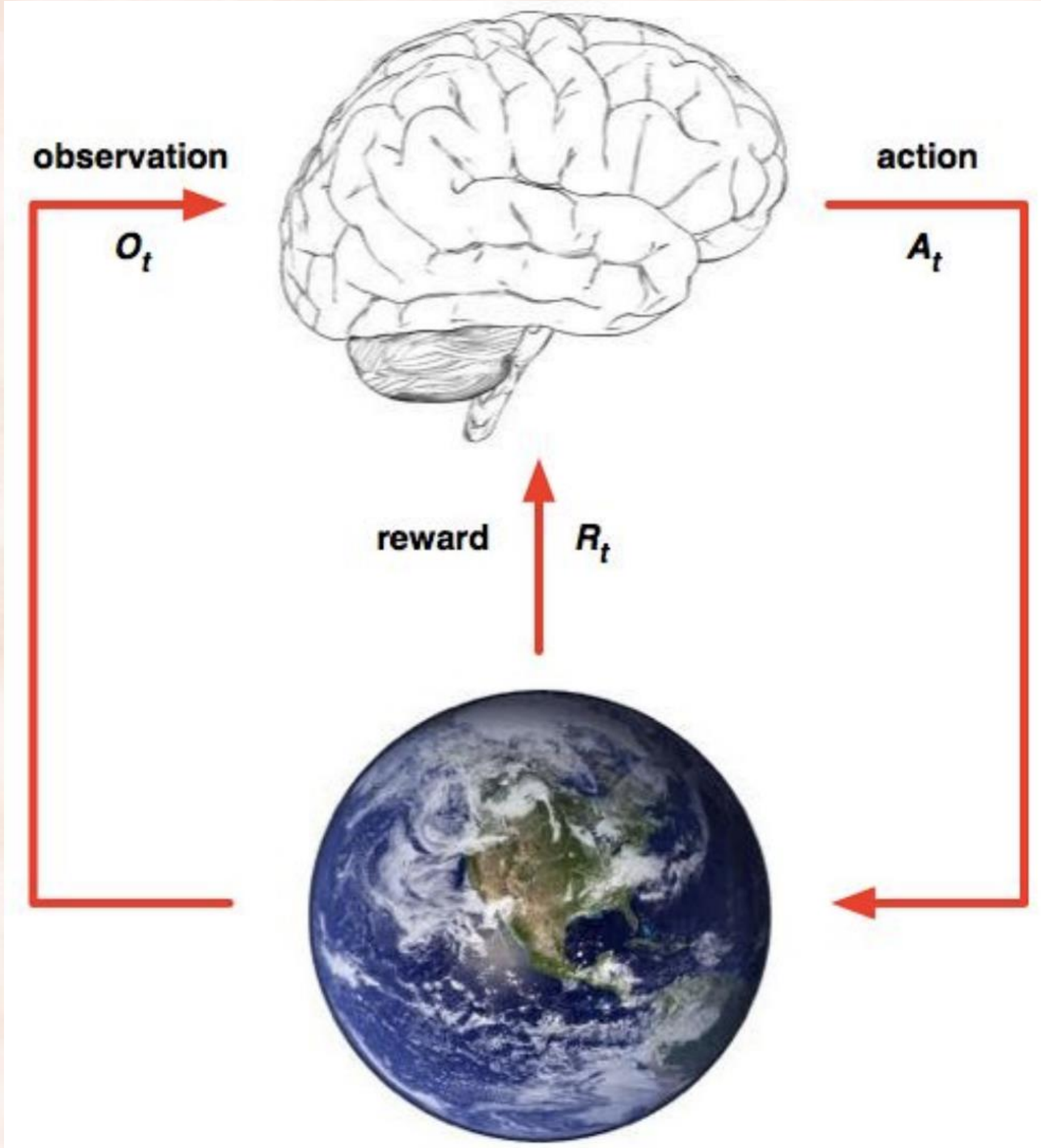


Learned Features (Representations)



Reinforcement Learning




observation
action
O_t
A_t
reward
R_t

Reinforcement Learning: TD-Gammon

[Tesauro, CACM'95]

- Learn to play Backgammon:
  - Immediate reward:
    - +100 if win
    - –100 if lose
    - 0 for all other states
  - *Temporal Difference Learning* with a *Multilayer Perceptron*.
  - Trained by playing 1.5 million games against itself.
  - Played competitively against top-ranked players in international tournaments.

Reinforcement Learning

- Interaction between agent and environment modeled as a sequence of *actions* & *states*:
  - Learn *policy* for mapping states to actions in order to maximize a *reward*.
  - Reward may be given only at the end state => *delayed reward*.
  - States may be only *partially observable*.
  - Trade-off between *exploration* and *exploitation*.
- Examples:
  - Backgammon [Tesauro, CACM'95], helicopter flight [Abbeel, NIPS'07].
  - 49 Atari games, using deep RL [Mnih et al., Nature'15].
  - AlphaGo [Silver et al., 2016], AlphaZero [Silver et al., 2017], …

Background readings

- **Python**:
  - Introductory Python lecture.
- **Probability theory**:
  - Basic probability theory (pp. 12-19) in Pattern Recognition and Machine Learning.
  - Chapter 3 in DL textbook on Probability and Information Theory.
- **Linear algebra**:
  - Chapter 2 in DL textbook on Linear Algebra.
  - Chapter 2 on Linear Algebra in Mathematics for Machine Learning.
- **Calculus**:
  - Basic properties for derivatives, exponentials, and logarithms.
  - Chapter 4.3 in DT textbook on Numerical Computation.