

Machine Learning

CS 4900/5900

Introduction

Razvan C. Bunescu

School of Electrical Engineering and Computer Science

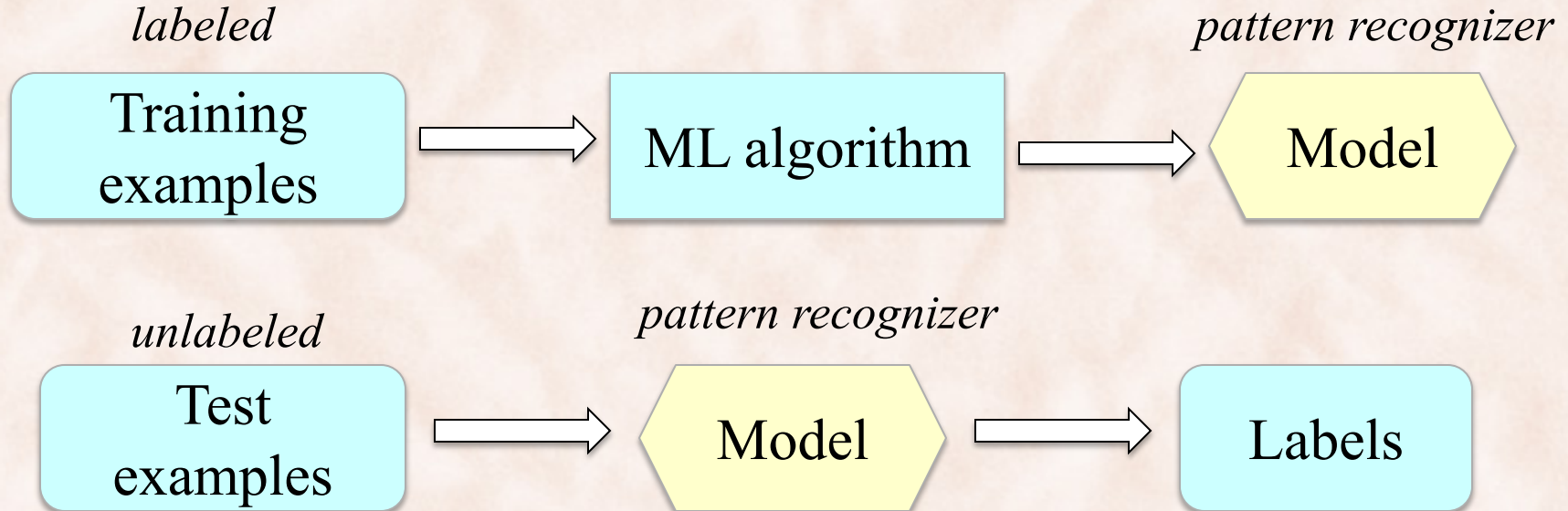
bunescu@ohio.edu

What is (Human) Learning?

- Merriam-Webster:
 - *learn* = to acquire knowledge, understanding, or skill ... by study, instruction, or *experience*.
- Why do we learn?
 - to *improve performance* on a given *task*.
- What (tasks) do we learn:
 1. categorize email, recognize faces, diagnose diseases, translate, ...
 2. clustering (fish, insects, birds, mice, humans), summarization, sound source separation, ...
 3. walk, play backgammon, ride bikes, drive cars, fly helicopters, ...

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 (previously unseen) unlabeled examples.



ML is Meta-Programming

- An ML **model** (e.g. a *neural network*) is a computer **program**:
 - We do not want to explicitly program (model) the computer for each particular task.
 - Use a general ML algorithm and task-specific data to automatically create the Program, i.e. the Model, that solves the task.
- ⇒ An ML algorithm (e.g. *gradient descent*) is a **meta-program**.

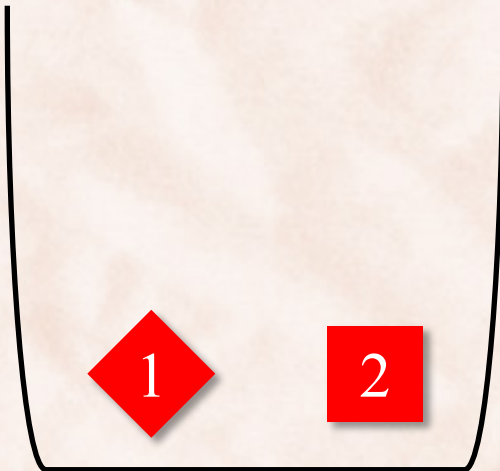
Example

$M_1: x \text{ is Red} \Rightarrow x \in C_1$

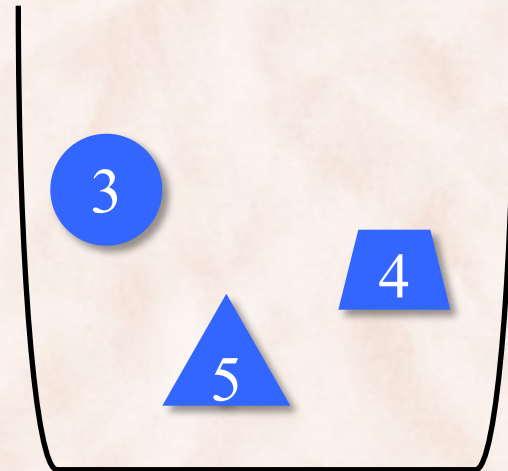
$M_2: x \text{ is a Square or } x \text{ is a Diamond} \Rightarrow x \in C_1$

$M_3: x \text{ is Red and } x \text{ is a Quadrilateral} \Rightarrow x \in C_1$

?



Class C_1



Class C_2

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.

ML Objective

- Find a model \mathbf{M}

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

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

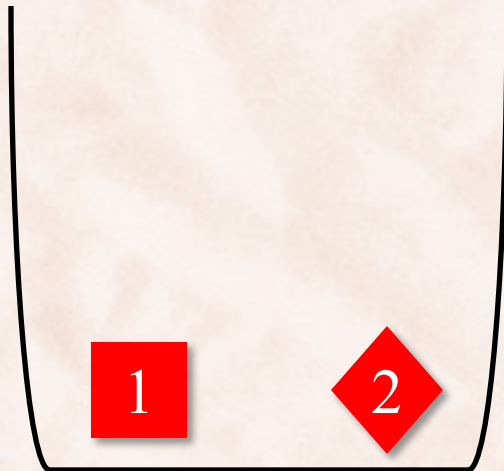
- **Inductive hypothesis:** Models that perform well on training examples are expected to do well on test (unseen) examples.
- **Occam's Razor:** Simpler models are expected to do better than complex models on test examples (assuming similar training performance).

Example

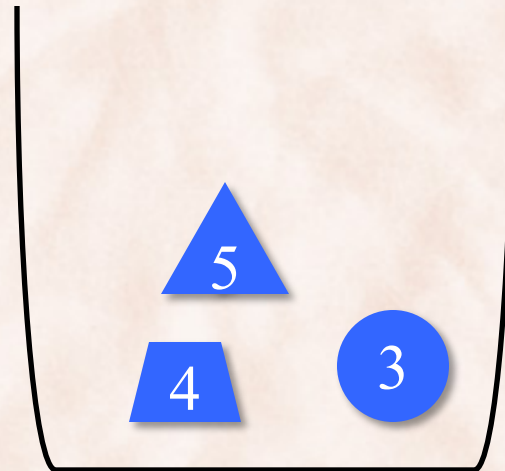
$M_1: x \text{ is Red} \Rightarrow x \in C_1$

$M_2: x \text{ is a Square or } x \text{ is a Diamond} \Rightarrow x \in C_1$

$M_3: x \text{ is Red and } x \text{ is a Quadrilateral} \Rightarrow x \in C_1$



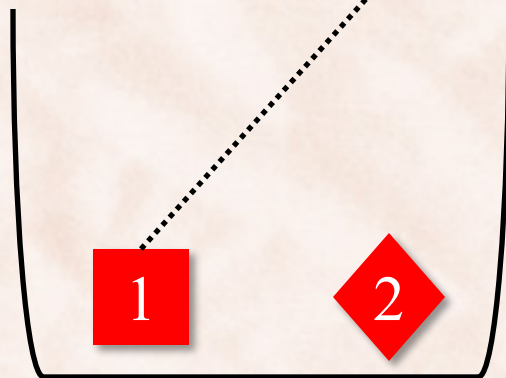
Class C_1



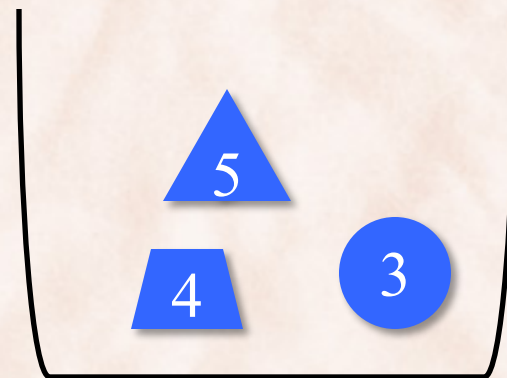
Class C_2

Feature Vectors

Features	$\varphi(x_1)$	$\varphi(x_2)$	$\varphi(x_3)$	$\varphi(x_4)$	$\varphi(x_5)$
(φ_1) Red?	1	1	0	0	0
(φ_2) Quad?	1	1	0	1	0
(φ_3) Square?	1	0	0	0	0
(φ_4) Diamond?	0	1	0	0	0
(y) Label	$y_1=+1$	$y_2=+1$	$y_3=-1$	$y_4=-1$	$y_5=-1$



Class C_1



Class C_2

Learning with Labeled Feature Vectors

Features	$\varphi(x_1)$	$\varphi(x_2)$	$\varphi(x_3)$	$\varphi(x_4)$	$\varphi(x_5)$
(φ_1) Red?	1	1	0	0	0
(φ_2) Quad?	1	1	0	1	0
(φ_3) Square?	1	0	0	0	0
(φ_4) Diamond?	0	1	0	0	0
(y) Label	$y_1=+1$	$y_2=+1$	$y_3=-1$	$y_4=-1$	$y_5=-1$

$$\begin{array}{llll} \varphi(x_1) = [1, 1, 1, 0]^T & \varphi(x_2) = [1, 1, 0, 1]^T & \varphi(x_3) = [0, 0, 0, 0]^T & \dots \\ y_1=+1 & y_2=+1 & y_3=-1 & \dots \end{array}$$

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

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

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

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

	Red?	Quad?	Square?	Diamond?		
$\phi(\mathbf{x}_1) = [1, 1, 1, 0]^T$					label $y_1 = +1$	$\Rightarrow \mathbf{w}^T \phi(\mathbf{x}_1) = 1 \geq 1$
$\phi(\mathbf{x}_2) = [1, 1, 0, 1]^T$					label $y_2 = +1$	$\Rightarrow \mathbf{w}^T \phi(\mathbf{x}_2) = 1 \geq 1$
$\phi(\mathbf{x}_3) = [0, 0, 0, 0]^T$					label $y_3 = -1$	$\Rightarrow \mathbf{w}^T \phi(\mathbf{x}_3) = 0 < 1$
$\phi(\mathbf{x}_4) = [0, 1, 0, 0]^T$					label $y_3 = -1$	$\Rightarrow \mathbf{w}^T \phi(\mathbf{x}_4) = 0 < 1$
$\phi(\mathbf{x}_5) = [0, 0, 0, 0]^T$					label $y_3 = -1$	$\Rightarrow \mathbf{w}^T \phi(\mathbf{x}_5) = 0 < 1$

$$\mathbf{w} = [1, 0, 0, 0]^T$$

$\Rightarrow M_1$ error is 0%

Learning = finding parameters $\mathbf{w} = [w_1, w_2, w_3, w_4]^T$ such that ($\tau = 1$):

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

where $\mathbf{w}^T \phi(\mathbf{x}) = w_1 \phi_1(\mathbf{x}) + w_2 \phi_2(\mathbf{x}) + w_3 \phi_3(\mathbf{x}) + w_4 \phi_4(\mathbf{x})$

M_2 : x_i is Square or Diamond $\Rightarrow y_i = +1$

	Red?	Quad?	Square?	Diamond?		
$\varphi(x_1) = [1, 1, 1, 0]^T$					label $y_1 = +1$	$\Rightarrow \mathbf{w}^T \varphi(x_1) = 1 \geq 1$
$\varphi(x_2) = [1, 1, 0, 1]^T$					label $y_2 = +1$	$\Rightarrow \mathbf{w}^T \varphi(x_2) = 1 \geq 1$
$\varphi(x_3) = [0, 0, 0, 0]^T$					label $y_3 = -1$	$\Rightarrow \mathbf{w}^T \varphi(x_3) = 0 < 1$
$\varphi(x_4) = [0, 1, 0, 0]^T$					label $y_3 = -1$	$\Rightarrow \mathbf{w}^T \varphi(x_4) = 0 < 1$
$\varphi(x_5) = [0, 0, 0, 0]^T$					label $y_3 = -1$	$\Rightarrow \mathbf{w}^T \varphi(x_5) = 0 < 1$

$$\mathbf{w} = [0, 0, 1, 1]^T$$

$\Rightarrow M_2$ error is 0%

Learning = finding parameters $\mathbf{w} = [w_1, w_2, w_3, w_4]^T$ such that ($\tau = 1$):

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

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

Linear Discriminant Functions: Two classes ($K = 2$)

- Use a linear function of the input vector:

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

weight vector

bias = - threshold

- Decision:

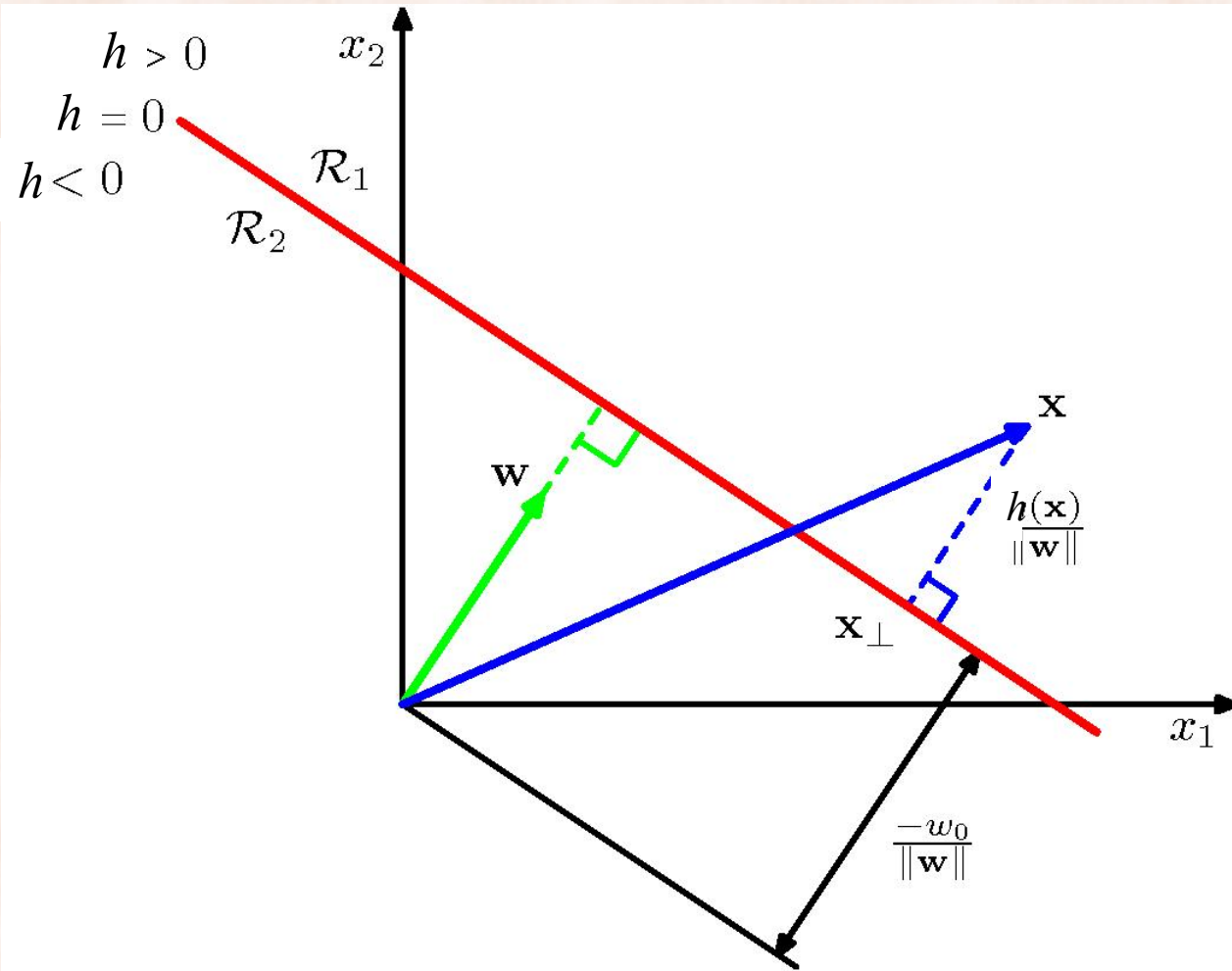
$\mathbf{x} \in C_1$ if $h(\mathbf{x}) \geq 0$, otherwise $\mathbf{x} \in C_2$.

\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



The Perceptron Algorithm: Two Classes

$$t_n \in \{+1, -1\}$$

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

Repeat:

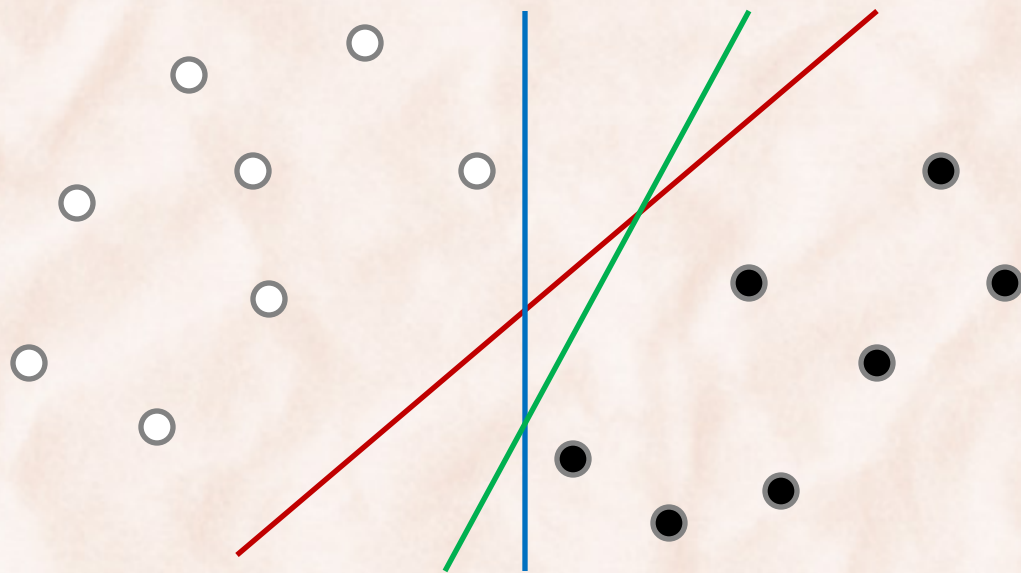
- a) until convergence.
- b) for a number of epochs E .

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](#)].

Classifiers & Margin



The Perceptron Algorithm: Two Classes

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

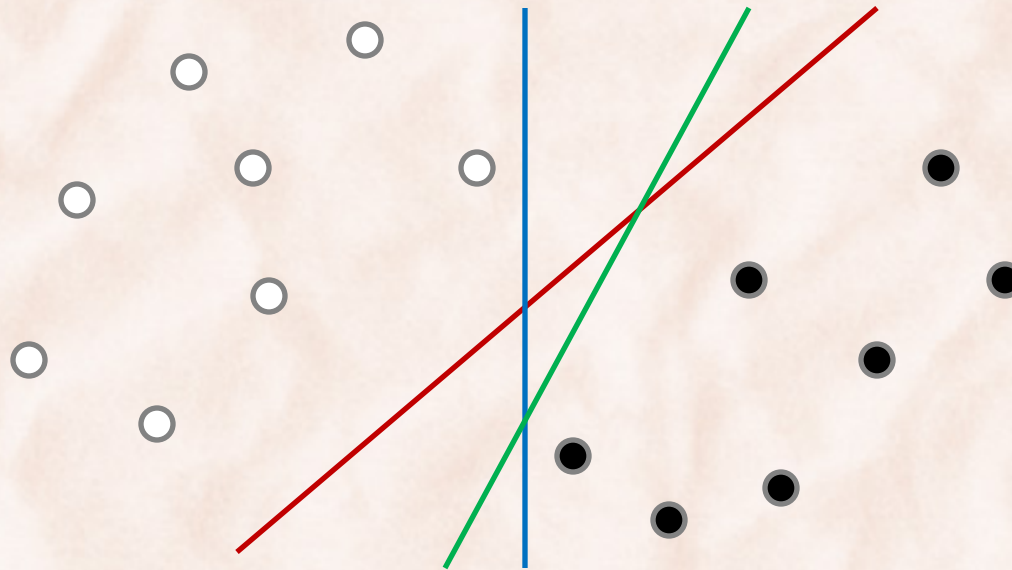
Repeat:

- a) until convergence.
- b) for a number of epochs E .

Loop invariant: \mathbf{w} is a weighted sum of training vectors:

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

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.

M_1 or M_2 ?

- Model M_1 : x_i is **Red** $\Rightarrow y_i = +1$
 - $\mathbf{w}^{(1)} = [1, 0, 0, 0]^T$
 - **Error = 0%**
- Model M_2 : x_i is **Square or Diamond** $\Rightarrow y_i = +1$
 - $\mathbf{w}^{(2)} = [0, 0, 1, 1]^T$
 - **Error = 0%**
- Which one should we choose?
 - Which one is expected to perform better on unseen (new) examples?

ML Objective

- Find a model \mathbf{w} that is *simple* and that *fits the training data*.

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

M_1 or M_2 ?

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

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

Complexity(\mathbf{w}) = ?

$\|\mathbf{w}\|_0$ i.e. # non-zero values

$\|\mathbf{w}\|_1$ i.e. sum of absolute values

$\|\mathbf{w}\|_2$ i.e. sum of squared values

ML Objectives

- Find a model \mathbf{w} that is *simple* and that *fits the training data*.

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \operatorname{Complexity}(\mathbf{w}) + \operatorname{Error}(\mathbf{w}, \text{Data})$$

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

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

ML Objectives

Support Vector Machines:

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

Upper bound on the number of misclassified training examples

subject to:

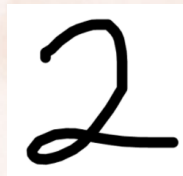
$$t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b) \geq 1 - \xi_n, \quad \forall n \in \{1, \dots, N\}$$
$$\xi_n \geq 0$$

ML Concepts & Notation

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

• Examples:

1. Digit recognition:

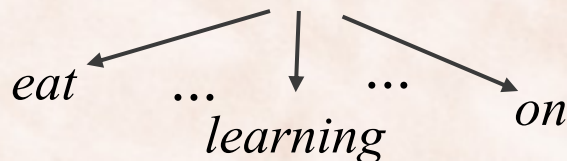


instance $\mathbf{x} = ?$

label $t = ?$

2. Language modelling:

- “machine is a hot topic in AI”



instance $\mathbf{x} = ?$

label $t = ?$

ML Concepts & Notation

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

ML Concepts & Notation

- A training dataset is a set of (training) examples $(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_N, t_N)$:
 - The data matrix X contains all instance vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ row-wise.
 - The label vector $\mathbf{t} = [t_1, t_2, \dots, t_N]^T$.
- A test dataset is a set of (test) examples $(\mathbf{x}_{N+1}, t_{N+1}), \dots, (\mathbf{x}_{N+M}, t_{N+M})$:
 - **Must be new/unseen/different from the training examples!**

ML Concepts & Notation

- There is a function f that maps an instance \mathbf{x} to its label $t = f(\mathbf{x})$.
 - f is unknown / not given.
 - But we observe samples from f : $(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_N, t_N)$.
- 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.
- The model y 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

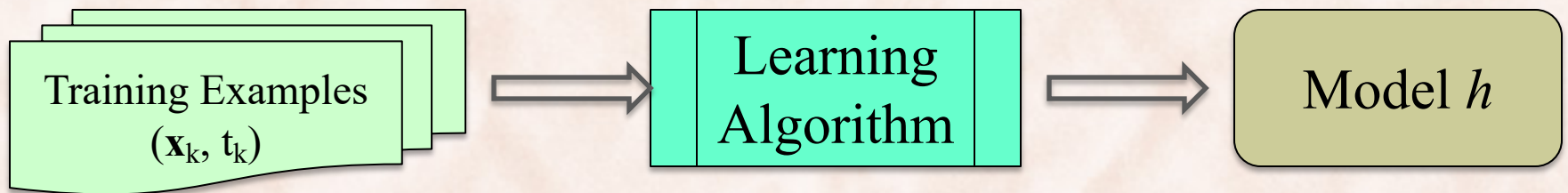
- Fitting performance = how well the model performs on training examples.
- Generalization performance = how well the model performs on unseen (test) examples.
- We are interested in **Generalization**:
 - Prefer finding patterns to memorizing examples!
 - **Overfitting**:
 - **Underfitting**:
 - **Regularization**:

Regularization = Any Method that Alleviates Overfitting

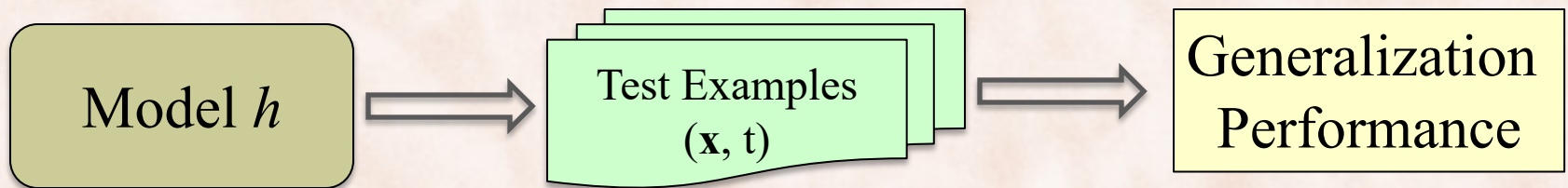
- **Parameter norm penalties** (term in the objective).
- Limit parameter norm (constraint).
- Dataset augmentation.
- **Dropout.**
- Ensembles.
- Semi-supervised learning.
- **Early stopping.**
- Noise robustness.
- Sparse representations.
- Adversarial training.

Supervised Learning

Training



Testing



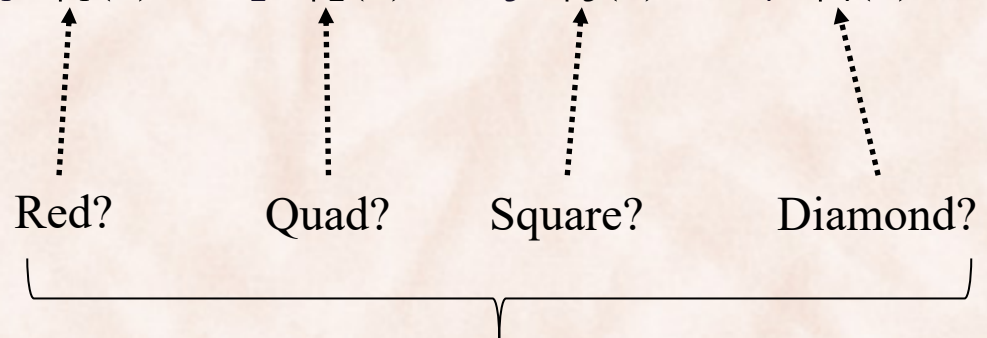
Features

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

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

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

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



Where do these features come from?

Object Recognition: Cats



Pixels as Features?

$\varphi(\mathbf{x}) = [25, 63, 125, 32, 84, 257, \dots, 13,$
 $27, 39, 8, 213, 107, 54, 73, \dots, 91,$
 $67, 59, 72, 22, 112, 54, 25, \dots, 9,$
Poor recognition accuracy! $\dots, 28,$
 $18, 57, 18, 21, 2, 2, 2, 2, 2, 2, \dots,$
 $93, 44, 69, 85, 68, 54, 87, \dots, 11,$
 $117, 59, 117, 210, 177, 54, 72, \dots]^T$

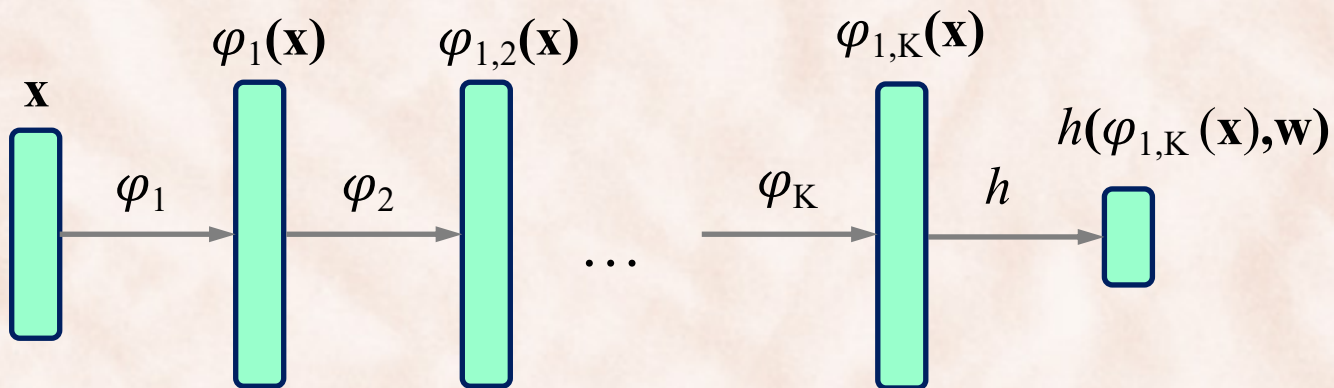
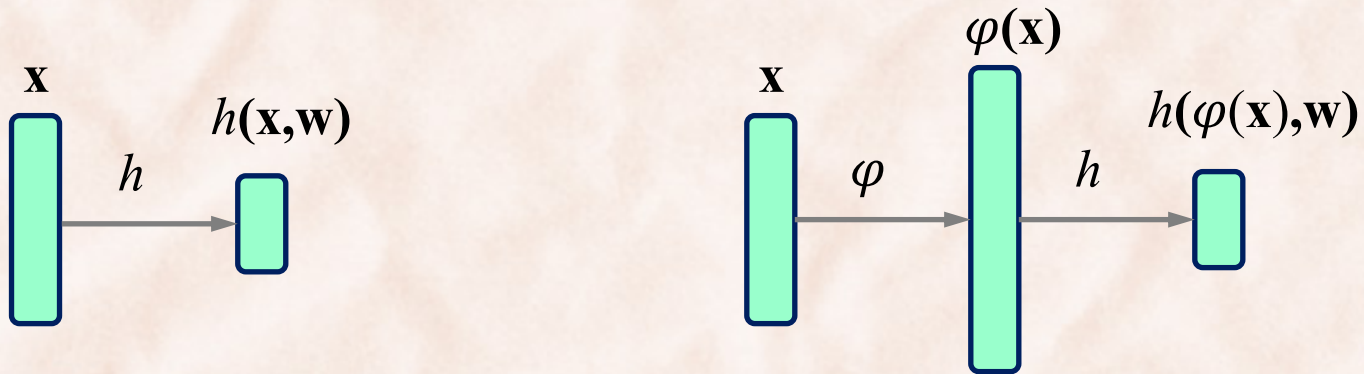


- Learning = finding parameters $\mathbf{w} = [w_1, w_2, w_3, \dots, w_k]^T$ such that:
 $\mathbf{w}^T \varphi(\mathbf{x}_i) \geq \tau$, if $y_i = +1$ (cat)
 $\mathbf{w}^T \varphi(\mathbf{x}_i) < \tau$, if $y_i = -1$ (other)
where $\mathbf{w}^T \varphi(\mathbf{x}) = w_1 \times \varphi_1(\mathbf{x}) + w_2 \times \varphi_2(\mathbf{x}) + w_3 \times \varphi_3(\mathbf{x}) + \dots + w_k \times \varphi_k(\mathbf{x})$

ML Concepts & Notation

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

Machine Learning vs. Deep Learning



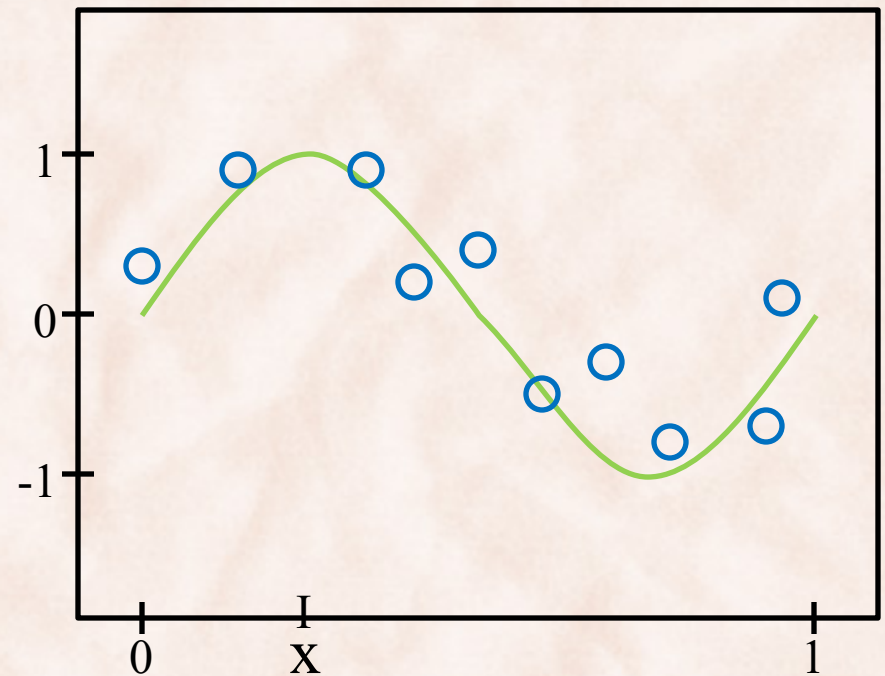
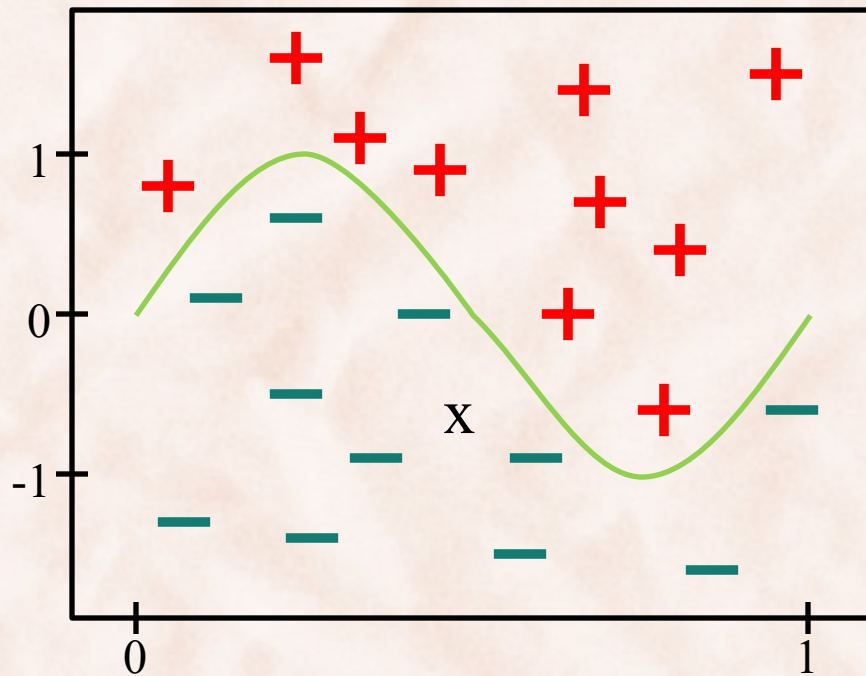
What is Machine Learning?

- **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 $t \in T$:
 - **Classification:**
 - The output $t \in T$ is one of a finite set of discrete categories.
 - **Regression:**
 - The output $t \in T$ is continuous, or has a continuous component.
- Supervision = set of training examples:
 $(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_n, t_n)$

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

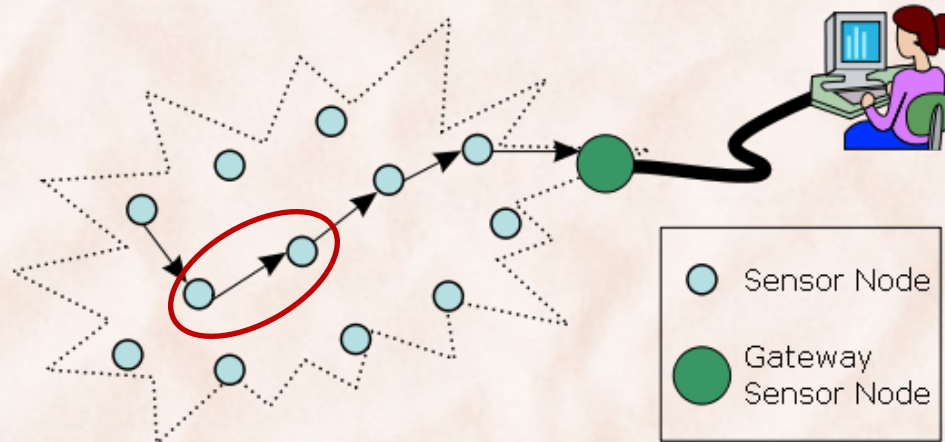
UK NATIONAL LOTTERY. GOVERNMENT ACCREDITED LICENSED LOTTERY. REGISTERED UNDER THE UNITED KINGDOM DATA PROTECTION ACT;

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: Routing in Wireless Sensor Networks

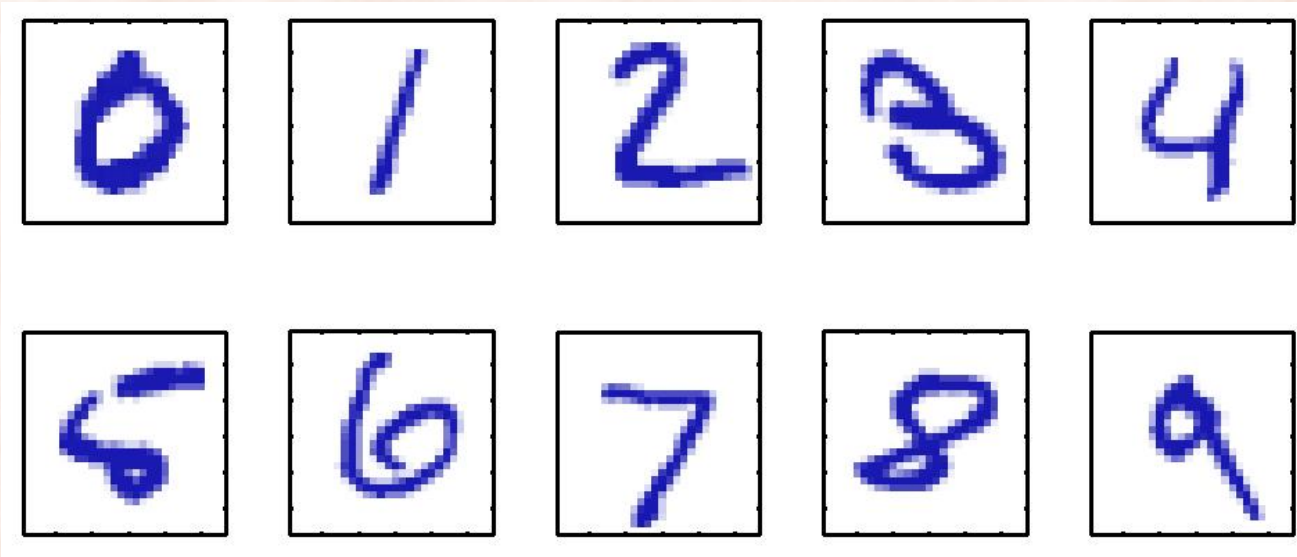
[Wang, Martonosi & Peh, SECON'06]



- Link quality prediction:
 - Provide a set of training links:
 - received signal strength, send/forward buffer sizes
 - node depth from base station, forward/backward probability
 - LQI = Link Quality Indication, binarized as {Good, Bad}
 - Train *Decision Trees* model to predict LQ using runtime features.

Classification: Handwritten Zip Code Recognition

[Le Cun et al., Neural Computation '89]



- Handwritten digit recognition:
 - Provide images of handwritten digits, labeled as $\{0, 1, \dots, 9\}$.
 - Train *Neural Network* model to recognize digits from input images.

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 acute 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.

ML for Software Verification / ATP

- Software verification requires theorem proving.
- Proving a mathematical theorem requires finding and using relevant previous theorems and definitions:
 - The space of existing theorems and definitions is huge.
 - Use machine learning to narrow the search space to relevant theorems and definitions:
 - “Premise Selection for Mathematics by Corpus Analysis and *Kernel Methods*”, Alama et al., JAR 2012.
 - “DeepMath – *Deep Sequence* Models for Premise Selection”, Alemi et al., NIPS 2016.
 - “Learning to Prove Theorems via Interacting with Proof Assistants”, *TreeLSTM encoder-decoder*, Yang & Deng, ICML 2019.

Classification: Other Examples

- Named Entity Recognition
- Named Entity Disambiguation
- Relation Extraction
- Word Sense Disambiguation
- Coreference Resolution
- Sentiment Analysis
- Chord Recognition
- Voice Separation
- Tone recognition
- Gesture Recognition
- Galaxy Morphology Recognition
- Dysarthria Prediction
- Tone Classification in Mandarin Chinese
- ...

Regression: Examples

1. Stock market prediction:

- Use the current stock market conditions ($x \in X$) to predict tomorrow's value of a particular stock ($t \in T$).

2. Oil price, GDP, income prediction.

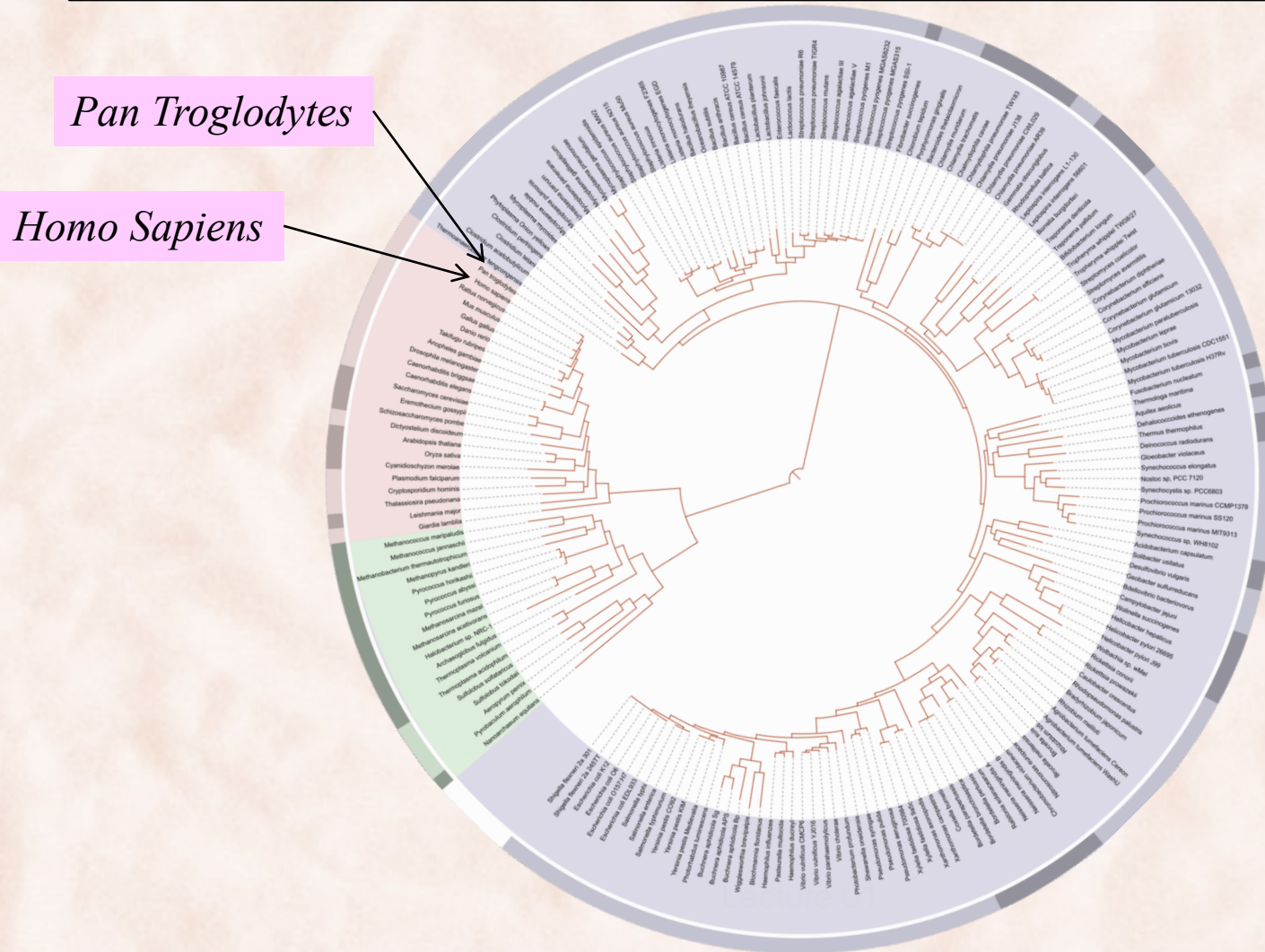
3. Chemical processes:

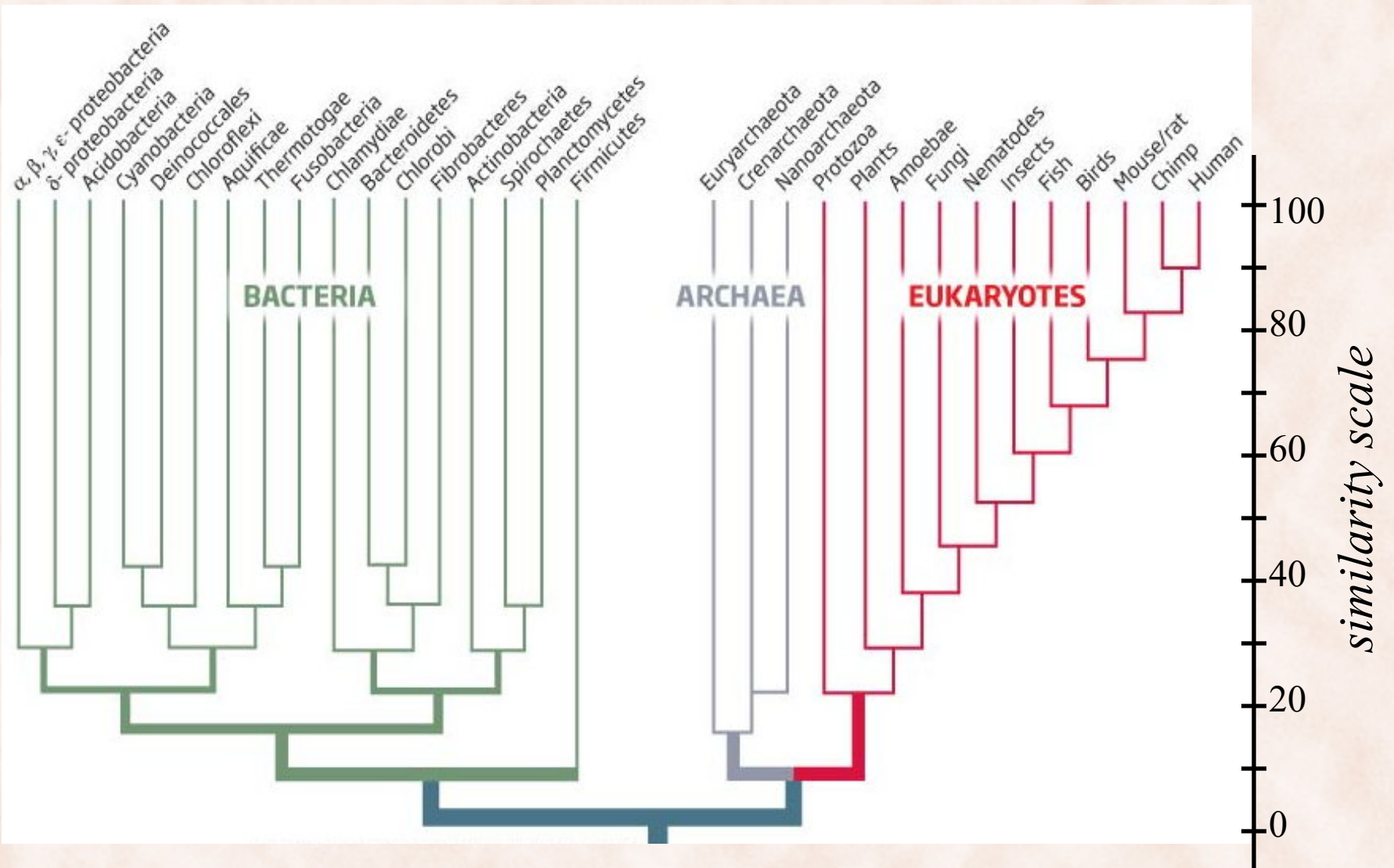
- Predict the yield in a chemical process based on the concentrations of reactants, temperature and pressure.

- Algorithms:

- *Linear Regression, Neural Networks, Support Vector Machines, ...*

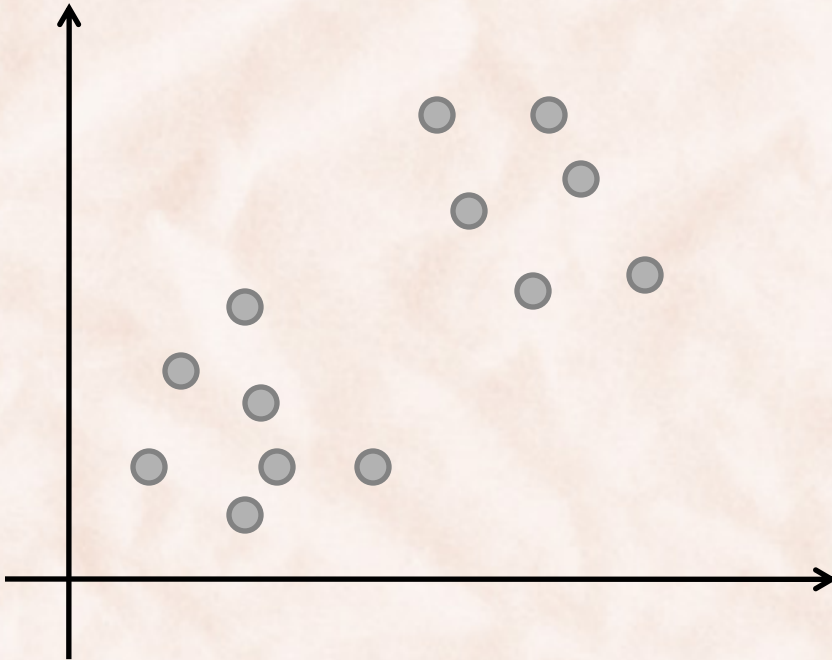
Unsupervised Learning: Hierarchical Clustering





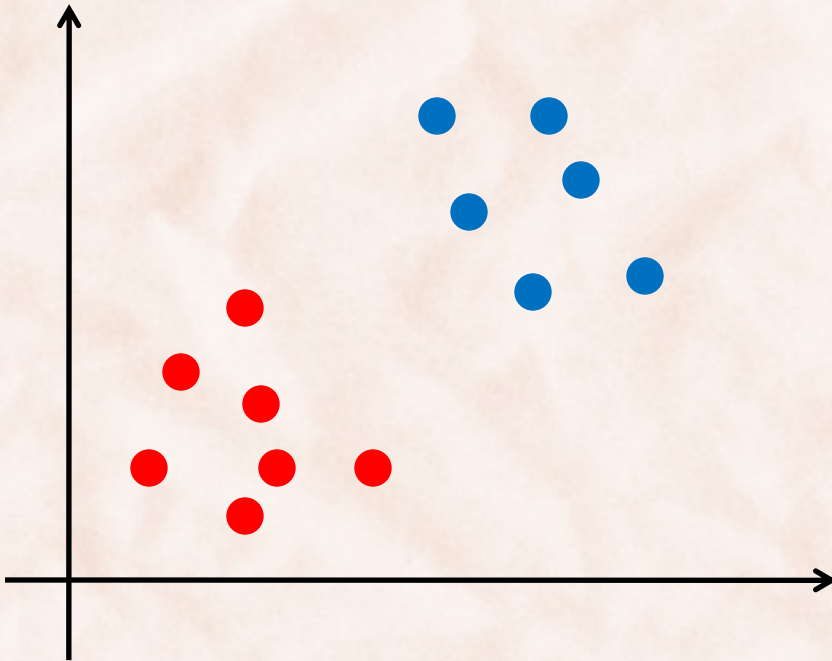
Unsupervised Learning: Clustering

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



Unsupervised Learning: Clustering

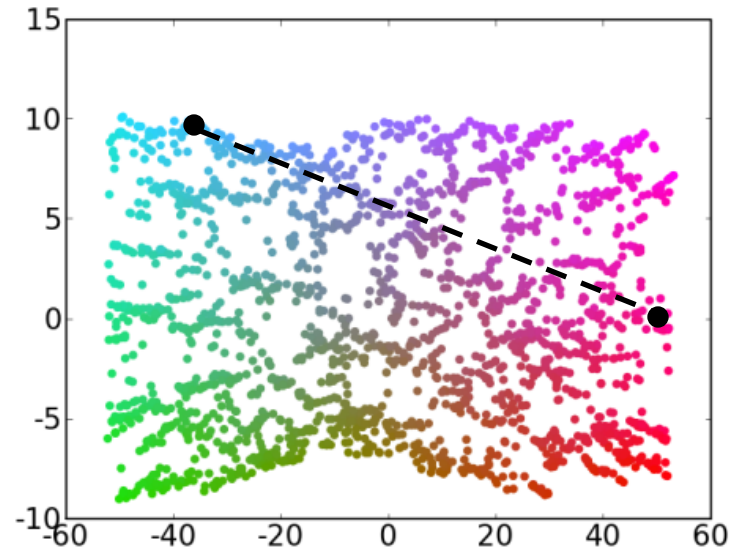
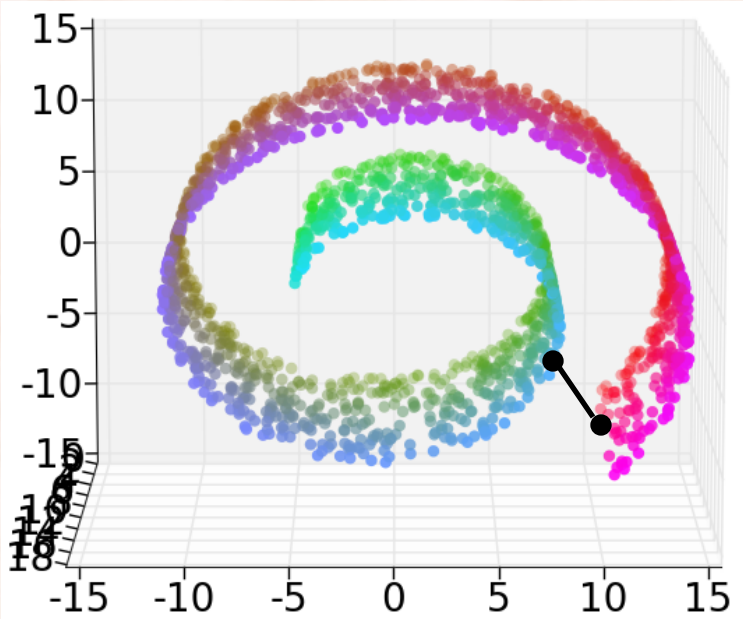
- Partition unlabeled examples into disjoint clusters such that:
 - Examples in the same cluster are very similar.
 - Examples in different clusters are very different.



- Need to provide:
 - number of clusters ($k = 2$)
 - similarity measure (Euclidean)

Unsupervised Learning: Dimensionality Reduction

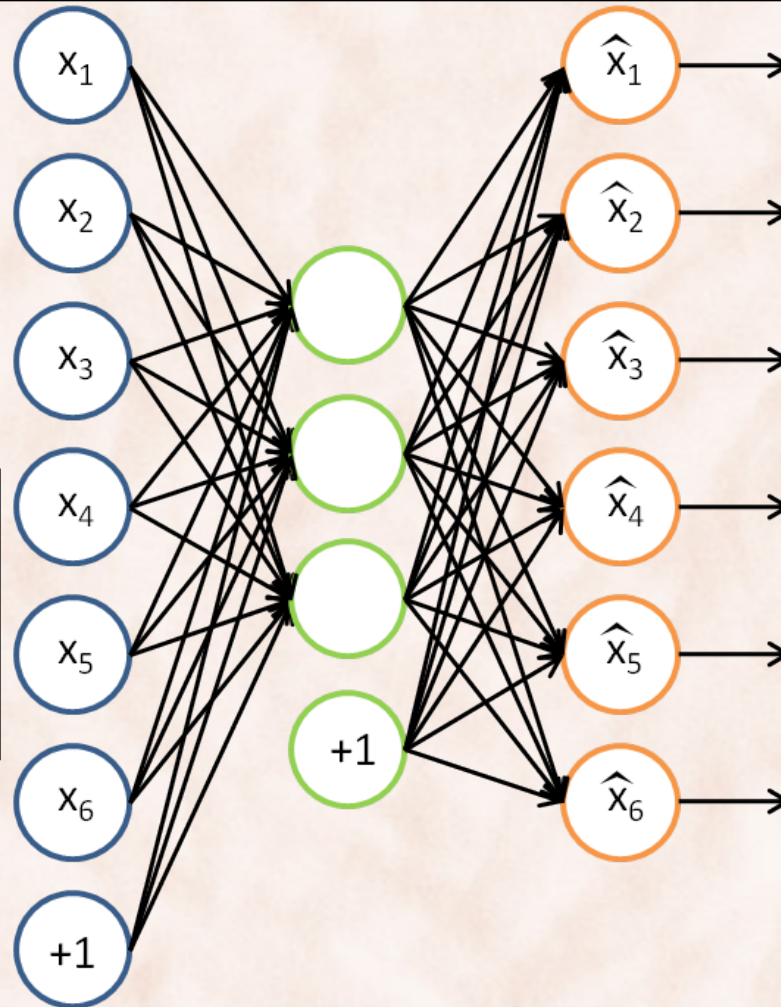
- Manifold Learning:
 - Data lies on a low-dimensional manifold embedded in a high-dimensional space.
 - Useful for *feature extraction* and *visualization*.



Unsupervised 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, ...]



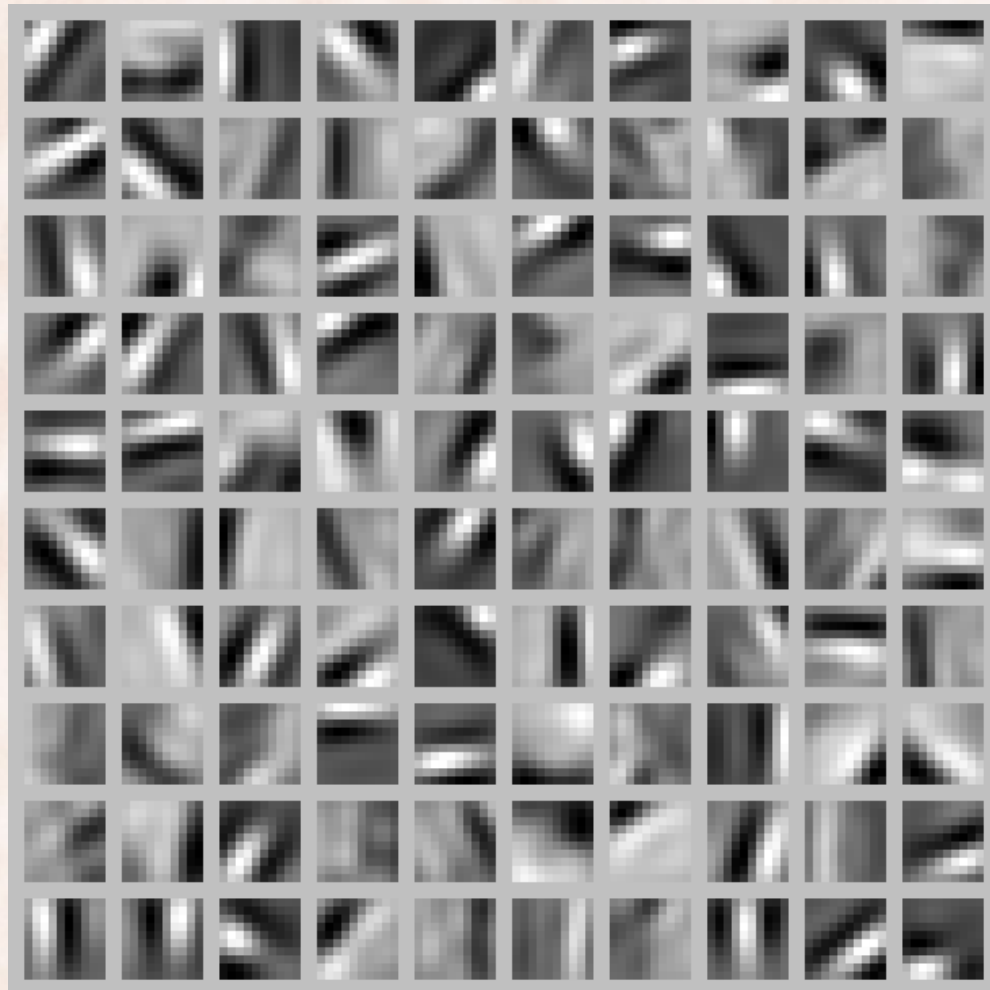
[25, 63, 125, 32, 84, 257, ..., 13,
27, 39, 8, 213, 107, 54, 73, ..., 91
67, 59, 72, 33, 112, 54, 35, ..., 9
18, 37, 18, 142, 162, 54, 53, ..., 28
93, 44, 69, 85, 68, 54, 87, ..., 11,
117, 59, 117, 210, 177, 54, 72, ...]

Input

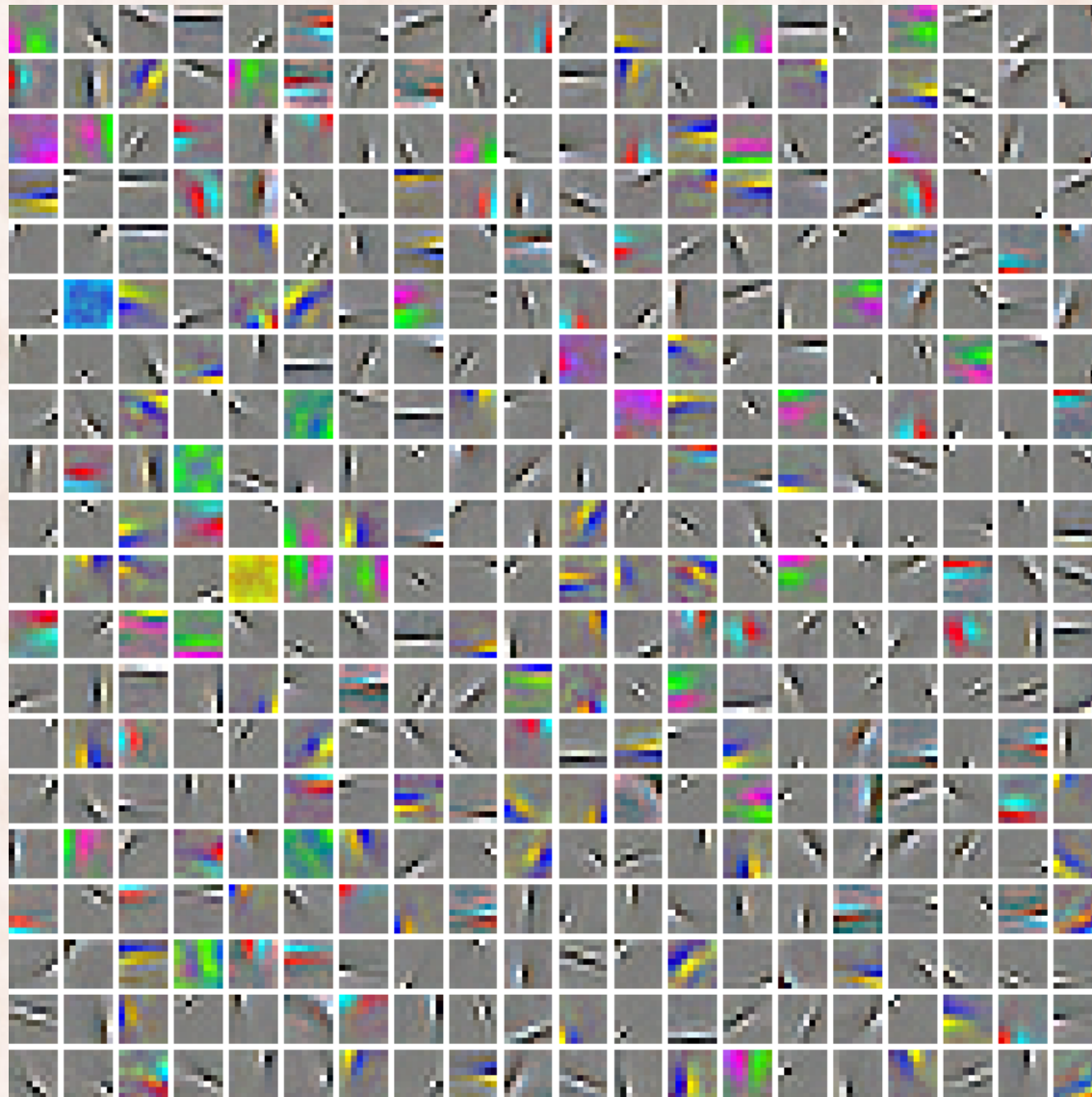
Features

Output

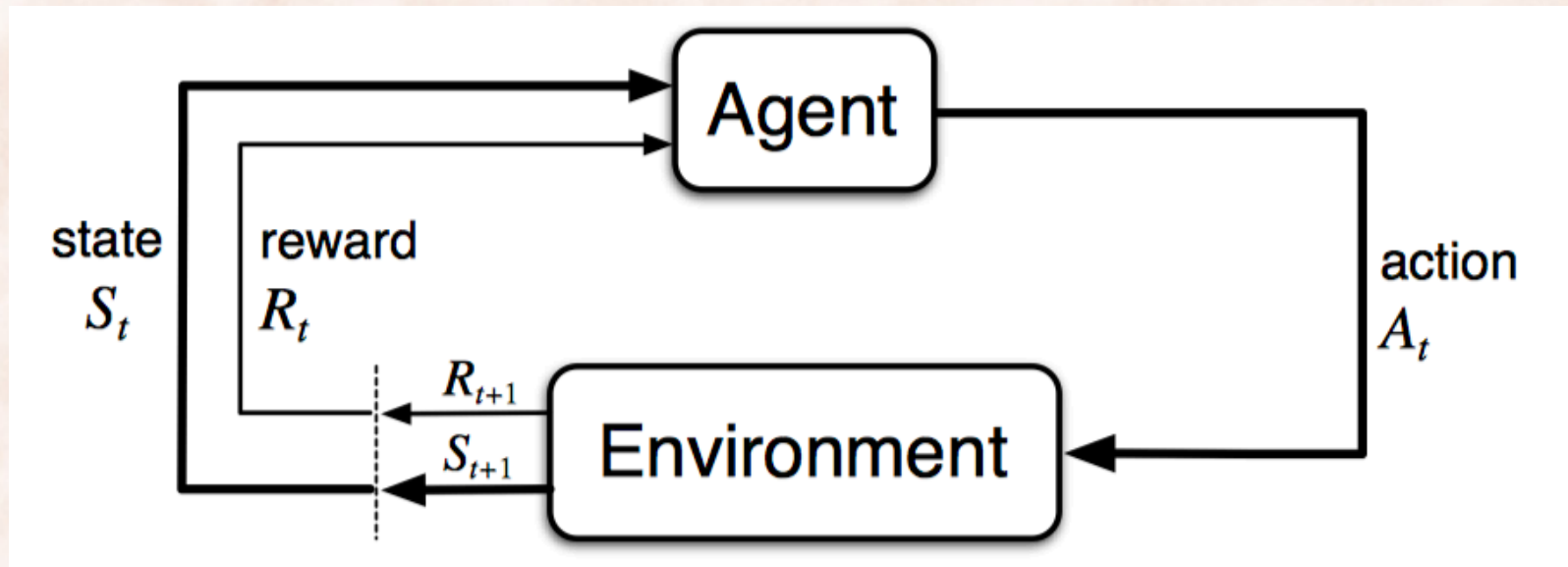
Learned Features (Representations)

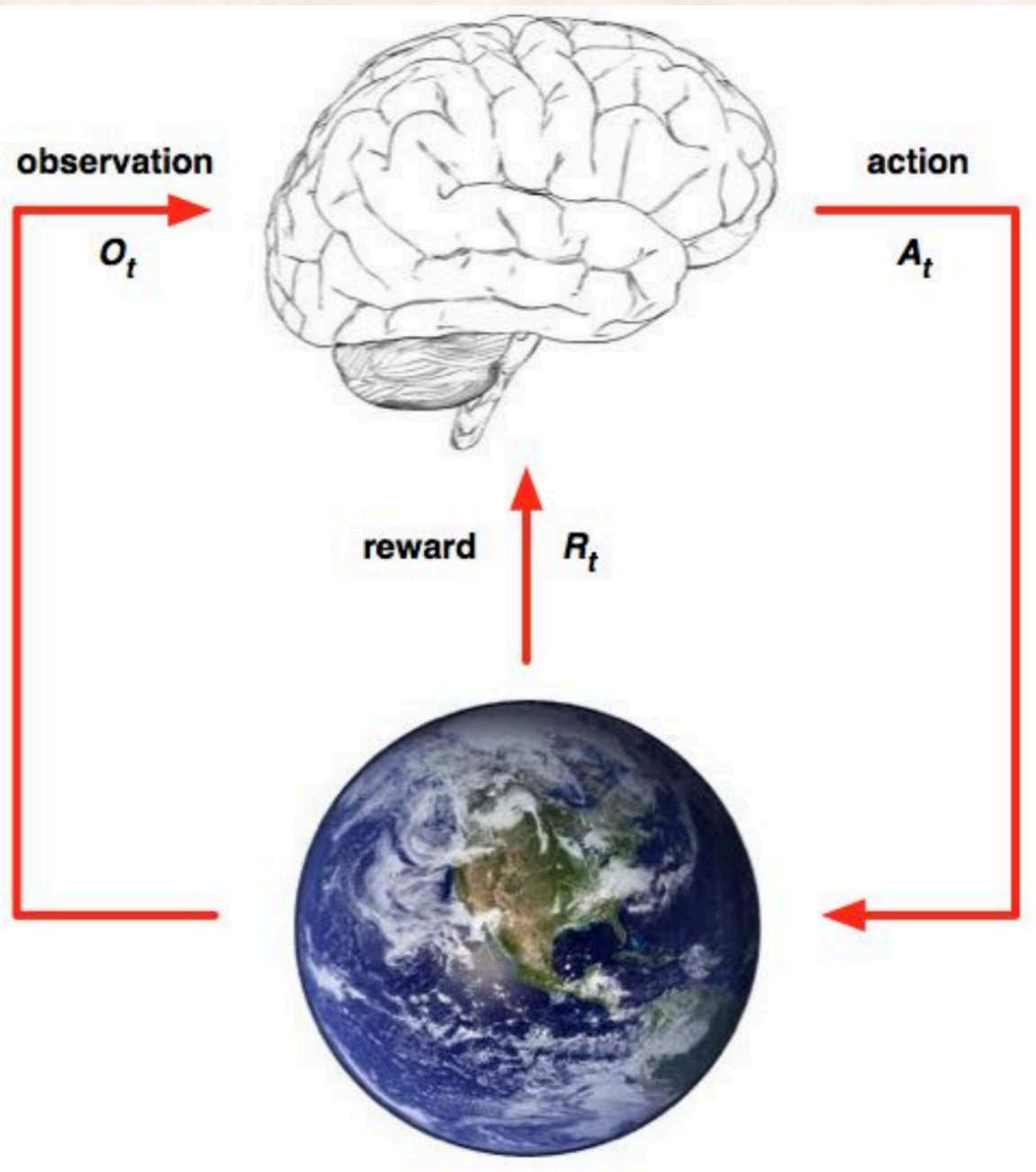


Learned Features (Representations)



Reinforcement Learning





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](#)].

Relevant Disciplines

- Mathematics:
 - Probability & Statistics
 - Information Theory
 - Linear Algebra
 - Optimization
- Algorithms:
 - Computational Complexity
 - Dynamic Programming
- Artificial Intelligence
 - Search
- Neurobiology

Supplemental Readings

- PRML 1.2, 2.1 – 2.1.1, 2.2 – 2.2.1, 2.3 (2.3.4, 2.3.9).
- PRML Appendix B and C.