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

\[ \hat{M} = \arg\min_M \text{Complexity}(M) + \text{Error}(M, \text{Data}) \]

• **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

$M_1$: $x$ is Red $\Rightarrow x \in C_1$

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

$M_3$: $x$ is Red and $x$ is a Quadrilateral $\Rightarrow x \in C_1$
Feature Vectors

<table>
<thead>
<tr>
<th>Features</th>
<th>$\varphi(x_1)$</th>
<th>$\varphi(x_2)$</th>
<th>$\varphi(x_3)$</th>
<th>$\varphi(x_4)$</th>
<th>$\varphi(x_5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\varphi_1)) Red?</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>((\varphi_2)) Quad?</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>((\varphi_3)) Square?</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>((\varphi_4)) Diamond?</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(y) Label</td>
<td>$y_1=+1$</td>
<td>$y_2=+1$</td>
<td>$y_3=-1$</td>
<td>$y_4=-1$</td>
<td>$y_5=-1$</td>
</tr>
</tbody>
</table>

Class $C_1$

Class $C_2$
Learning with Labeled Feature Vectors

<table>
<thead>
<tr>
<th>Features</th>
<th>( \phi(x_1) )</th>
<th>( \phi(x_2) )</th>
<th>( \phi(x_3) )</th>
<th>( \phi(x_4) )</th>
<th>( \phi(x_5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(( \phi_1 ))</td>
<td>Red?</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(( \phi_2 ))</td>
<td>Quad?</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(( \phi_3 ))</td>
<td>Square?</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(( \phi_4 ))</td>
<td>Diamond?</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(y) Label</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( y_1 = +1 )</td>
<td></td>
<td>( y_2 = +1 )</td>
<td>( y_3 = -1 )</td>
<td>( y_4 = -1 )</td>
<td>( y_5 = -1 )</td>
</tr>
</tbody>
</table>

\( \phi(x_1) = [1, 1, 1, 0]^T \) \( \phi(x_2) = [1, 1, 0, 1]^T \) \( \phi(x_3) = [0, 0, 0, 0]^T \) ...

\( y_1 = +1 \) \( y_2 = +1 \) \( y_3 = -1 \)

Learning = finding parameters \( w = [w_1, w_2, w_3, w_4]^T \) and \( \tau \) such that:
- \( w^T \phi(x_i) \geq \tau \), if \( y_i = +1 \)
- \( w^T \phi(x_i) < \tau \), if \( y_i = -1 \)

where \( w^T \phi(x) = w_1 \phi_1(x) + w_2 \phi_2(x) + w_3 \phi_3(x) + w_4 \phi_4(x) \)
Model $M_1$: $x_i$ is Red $\Rightarrow y_i = +1$

Red? Quad? Square? Diamond?

<table>
<thead>
<tr>
<th>$\phi(x_1)$</th>
<th>label $y_1 = +1$</th>
<th>$\Rightarrow w^T\phi(x_1) = 1 \geq 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[1, 1, 1, 0]^T$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\phi(x_2) = [1, 1, 0, 1]^T$</td>
<td>label $y_2 = +1$</td>
<td>$\Rightarrow w^T\phi(x_2) = 1 \geq 1$</td>
</tr>
<tr>
<td>$\phi(x_3) = [0, 0, 0, 0]^T$</td>
<td>label $y_3 = -1$</td>
<td>$\Rightarrow w^T\phi(x_3) = 0 &lt; 1$</td>
</tr>
<tr>
<td>$\phi(x_4) = [0, 1, 0, 0]^T$</td>
<td>label $y_3 = -1$</td>
<td>$\Rightarrow w^T\phi(x_4) = 0 &lt; 1$</td>
</tr>
<tr>
<td>$\phi(x_5) = [0, 0, 0, 0]^T$</td>
<td>label $y_3 = -1$</td>
<td>$\Rightarrow w^T\phi(x_5) = 0 &lt; 1$</td>
</tr>
</tbody>
</table>

$w = [1, 0, 0, 0]^T$ $\Rightarrow M_1$ error is 0%

Learning = finding parameters $w = [w_1, w_2, w_3, w_4]^T$ such that ($\tau = 1$):
- $w^T\phi(x_i) \geq 1$, if $y_i = +1$
- $w^T\phi(x_i) < 1$, if $y_i = -1$

where $w^T\phi(x) = w_1\phi_1(x) + w_2\phi_2(x) + w_3\phi_3(x) + w_4\phi_4(x)$
M₂: \( x_i \) is Square or Diamond \( \Rightarrow y_i = +1 \)

\[ \begin{align*}
\phi(x_1) &= [1, 1, 1, 0]^T \quad \text{label } y_1 = +1 \quad \Rightarrow w^T \phi(x_1) = 1 \geq 1 \\
\phi(x_2) &= [1, 1, 0, 1]^T \quad \text{label } y_2 = +1 \quad \Rightarrow w^T \phi(x_2) = 1 \geq 1 \\
\phi(x_3) &= [0, 0, 0, 0]^T \quad \text{label } y_3 = -1 \quad \Rightarrow w^T \phi(x_3) = 0 < 1 \\
\phi(x_4) &= [0, 1, 0, 0]^T \quad \text{label } y_3 = -1 \quad \Rightarrow w^T \phi(x_4) = 0 < 1 \\
\phi(x_5) &= [0, 0, 0, 0]^T \quad \text{label } y_3 = -1 \quad \Rightarrow w^T \phi(x_5) = 0 < 1
\end{align*} \]

\[ w = [0, 0, 1, 1]^T \quad \Rightarrow M₂ \text{ error is 0\%} \]

Learning = finding parameters \( w = [w_1, w_2, w_3, w_4]^T \) such that (\( \tau = 1 \)):
- \( w^T \phi(x_i) \geq 1 \), if \( y_i = +1 \)
- \( w^T \phi(x_i) < 1 \), if \( y_i = -1 \)

where \( w^T \phi(x) = w_1 \phi_1(x) + w_2 \phi_2(x) + w_3 \phi_3(x) + w_4 \phi_4(x) \)
Linear Discriminant Functions: Two classes \((K = 2)\)

- Use a linear function of the input vector:
  \[ h(x) = w^T \varphi(x) + w_0 \]

- Decision:
  \[ x \in C_1 \text{ if } h(x) \geq 0, \text{ otherwise } x \in C_2. \]
  \[ \Rightarrow \text{decision boundary is hyperplane } h(x) = 0. \]

- Properties:
  - \(w\) is orthogonal to vectors lying within the decision surface.
  - \(w_0\) controls the location of the decision hyperplane.
Geometric Interpretation

\[ h > 0 \]
\[ h = 0 \]
\[ h < 0 \]

\[ x_1 \]
\[ x_2 \]
The Perceptron Algorithm: Two Classes

\[ t_n \in \{+1, -1\} \]

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

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 \( w = 0 \)
2. **for** \( n = 1 \) \( \ldots \) \( N \)
3. \( h_n = sgn(w^T x_n) \)
4. **if** \( h_n \neq t_n \) **then**
5. \( w = w + t_n x_n \)

Loop invariant: \( w \) is a weighted sum of training vectors:

\[
w = \sum_n \alpha_n t_n x_n \quad \Rightarrow \quad w^T x = \sum_n \alpha_n t_n x_n^T x
\]

Repeat:
- a) until convergence.
- b) for a number of epochs \( E \).
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.
M₁ or M₂?

• Model M₁: *xᵢ is Red* => *yᵢ = +1*
  – *w⁽¹⁾ = [1, 0, 0, 0]ᵀ*
  – *Error = 0%*

• Model M₂: *xᵢ is Square or Diamond* => *yᵢ = +1*
  – *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.

\[
\hat{w} = \arg\min_w \text{Complexity}(w) + \text{Error}(w, Data)
\]
M₁ or M₂?

• Model M₁: \( x_i \) is Red \( \Rightarrow \) \( y_i = +1 \)
  – \( w^{(1)} = [1, 0, 0, 0]^T \)
  – Error = 0%

• Model M₂: \( x_i \) is Square or Diamond \( \Rightarrow \) \( y_i = +1 \)
  – \( w^{(2)} = [0, 0, 1, 1]^T \)
  – Error = 0%

\[ \hat{w} = \text{arg min}_w \text{ Complexity}(w) + \text{Error}(w, Data) \]

\[ \text{Complexity}(w) = ? \]

\[ ||w||_0 \text{ i.e. } \# \text{ non-zero values} \]

\[ ||w||_1 \text{ i.e. sum of absolute values} \]

\[ ||w||_2 \text{ i.e sum of squared values} \]
ML Objectives

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

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

Ridge Regression: \[
\arg\min_{\mathbf{w}} \frac{\lambda}{2} \| \mathbf{w} \|^2 + \frac{1}{2} \sum_{n=1}^{N} \{ y(x_n, \mathbf{w}) - t_n \}^2
\]

Logistic Regression: \[
\arg\min_{\mathbf{w}} \frac{\alpha}{2} \| \mathbf{w} \|^2 - \sum_{n=1}^{N} \ln p(t_n | x_n)
\]
Support Vector Machines:

$$\arg\min_w \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{N} \xi_n$$

subject to:

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

$$\xi_n \geq 0$$

Upper bound on the number of misclassified training examples
ML Concepts & Notation

• A (labeled) example \((x, t)\) consists of:
  – *Instance / observation / raw feature* vector \(x\).
  – *Label* \(t\).

• Examples:
  1. Digit recognition:
     - \( \text{instance } x = ? \)
     - \( \text{label } t = ? \)

  2. Language modelling:
     - “machine ............ is a hot topic in AI”
     - \( \text{instance } x = ? \)
     - \( \text{label } t = ? \)
Often, a raw observation \( x \) is pre-processed and further transformed into a feature vector \( \varphi(x) = [\varphi_1(x), \varphi_1(x), \ldots, \varphi_K(x)]^T \).

Where do the features \( \varphi_k \) 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 \((x_1,t_1), (x_2,t_2), \ldots, (x_N,t_N)\):
  – The **data matrix** \(X\) contains all instance vectors \(x_1, x_2, \ldots, x_N\) row-wise.
  – The label vector \(t = [t_1, t_2, \ldots, t_N]^T\).

• A **test dataset** is a set of (test) examples \((x_{N+1},t_{N+1}), \ldots, (x_{N+M},t_{N+M})\):
  – Must be new/unseen/different from the training examples!
ML Concepts & Notation

• There is a function $f$ that maps an instance $x$ to its label $t = f(x)$.
  – $f$ is unknown / not given.
  – But we observe samples from $f$: $(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)$.

• Learning means finding a model $h$ that maps an instance $x$ to a label $h(x) \approx f(x)$, i.e. close to the true label of $x$.
  – 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 $\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w})$
    - 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

Training Examples
\((x_k, t_k)\)

Learning Algorithm

Model \(h\)

Testing

Model \(h\)

Test Examples
\((x, t)\)

Generalization Performance
Features

- Learning = finding parameters $w = [w_1, w_2, w_3, w_4]$ and $\tau$ such that:
  $w^T \varphi(x_i) \geq \tau$, if $y_i = +1$
  $w^T \varphi(x_i) < \tau$, if $y_i = -1$

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

Where do these features come from?
Object Recognition: Cats
Pixels as Features?

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

- Learning = finding parameters \( \mathbf{w} = [w_1, w_2, w_3, \ldots w_k]^T \) such that:
  \[ w^T \phi(x_i) \geq \tau, \text{ if } y_i = +1 \text{ (cat)} \]
  \[ w^T \phi(x_i) < \tau, \text{ if } y_i = -1 \text{ (other)} \]

where \( \mathbf{w}^T \phi(x) = w_1 \times \phi_1(x) + w_2 \times \phi_2(x) + w_3 \times \phi_3(x) + \ldots w_k \times \phi_k(x) \)

Poor recognition accuracy!
Often, a raw observation $x$ is pre-processed and further transformed into a feature vector $\phi(x) = [\phi_1(x), \phi_1(x), \ldots, \phi_K(x)]^T$.

- Where do the features $\phi_k$ 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

\[ \varphi(x) \]

\[ h(x, w) \]

\[ \varphi_1(x) \]

\[ \varphi_2(x) \]

\[ \ldots \]

\[ \varphi_K(x) \]

\[ h(\varphi_1(x), w) \]

\[ h(\varphi_2(x), w) \]

\[ h(\varphi_K(x), w) \]
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 \( x \in X \) 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:
  
  \( (x_1, t_1), (x_2, t_2), \ldots, (x_n, t_n) \)
Classification vs. Regression
Classification: Junk Email Filtering

[ *Sahami, Dumais & Heckerman, AAAI’98* ]

**Email filtering:**

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

---

**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 …
Classification: Routing in Wireless Sensor Networks

- 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.

[Wang, Martonosi & Peh, SECON’06]
Classification: Handwritten Zip Code Recognition

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

[Le Cun et al., Neural Computation ‘89]
Classification: Medical Diagnosis

- 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](http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi)
    - Colon cancer microarray data:
      - [http://microarray.princeton.edu/oncology](http://microarray.princeton.edu/oncology)
  - Train *Logistic Regression* / *SVM* / *RVM* model to classify the gene expression of a tissue of unknown cancer status.

[Lecture 01](#)
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:
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

Pan Troglodytes
Homo Sapiens
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

\[ [25, 63, 125, 32, 84, 257, ..., 13, 27, 39, 8, 213, 107, 54, 73, ..., 91, 67, 59, 72, 33, 112, 54, 35, ..., 9, 18, 37, 18, 142, 162, 54, 53, ..., 28, 93, 44, 69, 85, 68, 54, 87, ..., 11, 117, 59, 117, 210, 177, 54, 72, ...] \]
Learned Features (Representations)
Learned Features (Representations)
Reinforcement Learning
Reinforcement Learning: TD-Gammon

• 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.

[Tesauro, CACM‘95]
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].
  – 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.