Machine Learning
ITCS 6156/8156

Introduction

Razvan C. Bunescu
Department of Computer Science @ CCI

rbunescu@uncc.edu
How to Automate Solutions to Computational Problems?

• Spam email:
  – Binary classification of emails into Spam vs. Ham.

• **Expert Systems** approach (also called rule-based):
  1. A group of experts write rules determining whether an email is spam or not.
  2. A programmer implement the rules into computer code.

• Example rules:
  – ”MONEY” appears in the text?
    • What if email sent by grandmother?
How to Automate Solutions to Computational Problems?

- **Expert Systems** approach (also called **rule-based**):
  - **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.
Why Machine Learning?

- Machine Learning (ML) approach:
  - Because ML is hot? No!
  - Rule-based (knowledge-based) may work very well.

<table>
<thead>
<tr>
<th>Input</th>
<th>=&gt;</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.2</td>
<td>2.1</td>
<td>63.3</td>
</tr>
<tr>
<td>65.4</td>
<td>3.5</td>
<td>229.1</td>
</tr>
<tr>
<td>46.2</td>
<td>0.5</td>
<td>22.9</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>25.3</td>
<td>4.0</td>
<td>?</td>
</tr>
</tbody>
</table>
Why Machine Learning?

- **Machine Learning (ML) approach:**
  1. Acquire a large enough dataset of *labeled examples*:
     - Email is the *instance*, the *label* is spam (+1) vs. not spam (-1).
  2. Represent emails as *feature vectors*:
     - Each feature has a *weight*, the sign of the weighted sum of features should match the label.
       - Traditional ML: engineer the features.
       - Deep ML: learn the features.
  3. **Learn the weights** s.t. the model (weighted combination of features) does well on labeled 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 (previously unseen) unlabeled examples.

---

**labeled**

Training examples → ML algorithm → Model (w)

**unlabeled**

Test examples → Model (w) → Labels

**pattern recognizer**
Example

\[ M_1: \text{x is Red} \Rightarrow x \in C_1 \]
\[ M_2: \text{x is a Square or x is a Diamond} \Rightarrow x \in C_1 \]
\[ M_3: \text{x is Red and x is a Quadrilateral} \Rightarrow x \in C_1 \]
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.
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

\[ M_1: x \text{ is Red} \implies x \in C_1 \]

\[ M_2: x \text{ is a Square or } x \text{ is a Diamond} \implies x \in C_1 \]

\[ M_3: x \text{ is Red and } x \text{ is a Quadrilateral} \implies x \in C_1 \]
### 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 $C_1$:
- 1
- 2

Class $C_2$:
- 3
- 4
- 5
Learning with Labeled Feature Vectors

<table>
<thead>
<tr>
<th>Features</th>
<th>φ(x₁)</th>
<th>φ(x₂)</th>
<th>φ(x₃)</th>
<th>φ(x₄)</th>
<th>φ(x₅)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(φ₁) Red?</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(φ₂) Quad?</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(φ₃) Square?</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(φ₄) Diamond?</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(y) Label</td>
<td>y₁=+1</td>
<td>y₂=+1</td>
<td>y₃=−1</td>
<td>y₄=−1</td>
<td>y₅=−1</td>
</tr>
</tbody>
</table>

φ(x₁) = [1, 1, 1, 0]ᵀ  φ(x₂) = [1, 1, 0, 1]ᵀ  φ(x₃) = [0, 0, 0, 0]ᵀ  ...

Learning = finding parameters w = [w₁, w₂, w₃, w₄]ᵀ and τ such that:
- wᵀφ(xᵢ) ≥ τ, if yᵢ = +1
- wᵀφ(xᵢ) < τ, if yᵢ = −1

where wᵀφ(x) = w₁φ₁(x) + w₂φ₂(x) + w₃φ₃(x) + w₄φ₄(x)
Model $M_1$: $x_i$ is Red $\Rightarrow y_i = +1$

Learning = finding parameters $w = [w_1, w_2, w_3, w_4]^T$ such that:

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

$\tau = 1$

$\varphi(x_1) = [1, 1, 1, 0]^T$ 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 $y_3 = -1$ $\Rightarrow \mathbf{w}^T \varphi(x_5) = 0 < 1$

$\mathbf{w} = [1, 0, 0, 0]^T$ $\Rightarrow M_1$ error is 0%
**M₂: xᵢ is Square or Diamond ⇒ yᵢ = +1**

<table>
<thead>
<tr>
<th>xᵢ</th>
<th>φ(xᵢ)</th>
<th>label</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>[1, 1, 1, 0]ᵀ</td>
<td>+1</td>
<td>wᵀφ(x₁) ≥ 1</td>
</tr>
<tr>
<td>x₂</td>
<td>[1, 1, 0, 1]ᵀ</td>
<td>+1</td>
<td>wᵀφ(x₂) ≥ 1</td>
</tr>
<tr>
<td>x₃</td>
<td>[0, 0, 0, 0]ᵀ</td>
<td>−1</td>
<td>wᵀφ(x₃) &lt; 1</td>
</tr>
<tr>
<td>x₄</td>
<td>[0, 1, 0, 0]ᵀ</td>
<td>−1</td>
<td>wᵀφ(x₄) &lt; 1</td>
</tr>
<tr>
<td>x₅</td>
<td>[0, 0, 0, 0]ᵀ</td>
<td>−1</td>
<td>wᵀφ(x₅) &lt; 1</td>
</tr>
</tbody>
</table>

**w = [0, 0, 1, 1]ᵀ**

⇒ wᵀφ(x₁) = 1 ≥ 1
⇒ wᵀφ(x₂) = 1 ≥ 1
⇒ wᵀφ(x₃) = 0 < 1
⇒ wᵀφ(x₄) = 0 < 1
⇒ wᵀφ(x₅) = 0 < 1

= > M₂ error is 0%

Learning = finding parameters **w = [w₁, w₂, w₃, w₄]ᵀ** such that (τ =1):
- wᵀφ(xᵢ) ≥ 1, if yᵢ = +1
- wᵀφ(xᵢ) < 1, if yᵢ = −1

where **wᵀφ(x) = w₁φ₁(x) + w₂φ₂(x) + w₃φ₃(x) + w₄φ₄(x)**
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*.

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

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

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

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

**Complexity**

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

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

\[ \|w\|_2^2 \text{ i.e sum of squared values} \]
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:**

$$\arg\min_{\mathbf{w}} \frac{\alpha}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^{N} \ln p(t_n | x_n)$$
ML Objectives

Support Vector Machines:

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

subject to:

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

$$\xi_n \geq 0$$

Upper bound on the number of misclassified training examples
Definition: *Bias* \( w_0 = -\text{Threshold } \tau \)

\[
w_1 \phi_1(x) + w_2 \phi_2(x) + w_3 \phi_3(x) + w_4 \phi_4(x) \geq \tau
\]

\[
w_1 \phi_1(x) + w_2 \phi_2(x) + w_3 \phi_3(x) + w_4 \phi_4(x) - \tau \geq 0
\]

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

\[
w_1 \phi_1(x) + w_2 \phi_2(x) + w_3 \phi_3(x) + w_4 \phi_4(x) + w_0 \geq 0
\]

\[
h(x) = w^T \phi(x) + w_0 \geq 0
\]

where:

\[
w = [w_1, w_2, w_3, w_4]
\]

\[
\phi(x) = [\phi_1(x), \phi_2(x), \phi_3(x), \phi_4(x)]
\]
Geometric Interpretation

• Often we drop the $\varphi$ and use bolded $\mathbf{x}$ itself to denote the feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_K]$.

• Example $\mathbf{x}$ is a point in a $K$-dimensional feature space.

• Parameters $\mathbf{w}$ form a vector.

• What does it mean that $\mathbf{w}^T \mathbf{x} + w_0 > 0$?
Linear Discriminant Functions: Two classes ($K = 2$)

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

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

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

\[ h(x) = w^T x + w_0 \]
Outline

• We want to use a linear function of the feature vector:
  \[ h(x) = \mathbf{w}^T \mathbf{x} + w_0 \]

• How to find \( \mathbf{w} \) automatically? Use ML!
  – Perceptron.
  – Logistic Regression.
  – SVMs.

• What if the data is not linearly separable? Make it!
  – Engineer new features or use kernels (Perceptron, SVMs).
  – Learn new features (Neural Networks).
Machine Learning (most of ML pre-2006)

- Hope raw data $x$ is linearly separable.

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

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

Use a Perceptron or LR or SVMs to learn $\mathbf{w}$. 

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

Use a Perceptron or LR or SVMs to learn $\mathbf{w}$. 

Deep Learning

- A raw observation vector $x$ is pre-processed and further transformed into a sequence of higher-level feature vectors $\varphi(x) = [\varphi_1(x), \varphi_2(x), \ldots, \varphi_K(x)]^T$ that are learned.

\[
h = w^T \varphi_K(x) + w_0
\]
Linear Models: \( h(x) = w^T x \)

- Given \( N \) training examples \((x_1, t_1), (x_2, t_2), \ldots (x_N, t_N)\) where:
  - Labels \( t_j \in \{-1, +1\} \).
  - Each example \( x_j \) is assumed to also contain a bias feature set to 1, corresponding to parameter \( w_0 \).

- Find parameter vector \( w \) such the the linear model \( h(x) = w^T x \) fits the training examples.
The Perceptron Algorithm: Two Classes

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

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].

$sgn(z) = +1$ if $z > 0$,
0 if $z = 0$,
$-1$ if $z < 0$
The Perceptron Algorithm: Two Classes

1. initialize parameters $\mathbf{w} = 0$
2. for $n = 1 \ldots N$
  3. $h_n = \mathbf{w}^T \mathbf{x}_n$
  4. if $h_n t_n \leq 0$ 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].
The Perceptron Algorithm: Two Classes

1. **initialize** parameters $w = 0$

2. **for** $n = 1 \cdots N$

3. $h_n = w^T x_n$

4. **if** $h_n \geq 0$ and $t_n = -1$

5. $w = w - x_n$

6. **if** $h_n \leq 0$ and $t_n = +1$

7. $w = w + x_n$

Repeat:

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

$s\text{gn}(z) = +1$ if $z > 0$,
$0$ if $z = 0$,
$-1$ if $z < 0$

What is the impact of the perceptron update on the score $w^T x_n$ of the misclassified example $x_n$?
The Perceptron Algorithm: Two Classes

initialize parameters $w = 0$

for epoch $e = 1 \ldots E$
  mistakes = 0
  for example $n = 1 \ldots N$
    $h_n = sgn(w^T x_n)$
    if $h_n \neq t_n$ then
      $w = w + t_n x_n$
      mistakes = mistakes + 1
  if mistakes = 0
    break Converged!

1 epoch = one pass over all training examples.
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. \( w = w + t_n x_n \)

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

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

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.
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 modeling:
     - “machine .......... is a hot topic in AI”
     - \(\text{instance } x = ?\)
     - \(\text{label } t = ?\)
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 unseen, i.e. new, i.e. 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 $f$: $(x_1, t_1 = f(x_1)), (x_2, t_2), \ldots (x_N, t_N)$.

- 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.
    • Assume **within-distribution** test examples.

• The **model** $h$ is often specified through a set of parameters $w$:
  – $x$ is mapped by the model to $h(x, w)$.

• The **objective function** $J(w)$ captures how poorly the model does on the training dataset:
  – Want to find $\hat{w} = \arg\min_w J(w)$
    • 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 $(x_k, t_k)$ \rightarrow \text{Learning Algorithm} \rightarrow \text{Model } h

Testing

Model $h$ \rightarrow Test Examples $(x, t)$ \rightarrow Generalization Performance
Features

- Learning = finding parameters $w = [w_1, w_2, w_3, w_4]$ 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 $w^T \phi(x) = w_1 \phi_1(x) + w_2 \phi_2(x) + w_3 \phi_3(x) + w_4 \phi_4(x)$

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

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

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

Poor recognition accuracy!
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.
Machine Learning vs. Deep Learning

\[ \phi(x) \]

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

\[ \phi_1(x) \quad \phi_1,2(x) \quad \ldots \quad \phi_{1,K}(x) \]

\[ h(\phi_{1,K}(x), w) \]
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

[Source: 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](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 …
Handwritten digit recognition:

- Provide images of handwritten digits, labeled as \{0, 1, \ldots, 9\}.
- Train **Convolutional Neural Network** model to recognize digits.

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

[Crishnapuram 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.
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, oil price, GDP, income prediction:
   – Use the current stock market conditions \( (x \in X) \) to predict tomorrow’s value of a particular stock \( (t \in T) \).

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.*
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

[• 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].
  – 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.

- **Linear algebra:**
  - Chapter 2 on Matrices in Mathematics for Machine Learning.
  - Sections 1-3 in Inderjit Dhillon's Linear Algebra Background.

- **Calculus:**
  - Basic properties for derivatives, exponentials, and logarithms.