BBM406 - Introduction to ML
Spring 2014

Week #1: Introduction

Asst. Prof. Aykut Erdem
Dept. of Computer Engineering
Hacettepe University

Logistics

• Instructor: Asst. Prof. Aykut ERDEM
  aykut@cs.hacettepe.edu.tr
  Office: 111
  Tel: 297 7500, 146

• Lectures: Mon 10:00 - 11:45_D9
  Fri 09:00 - 09:45_D9

• Office hour: By appointment

About BBM 406

• This is a undergraduate-level introductory course in machine learning (ML) which will give a broad overview of many concepts and algorithms in ML.

• The goal is to provide students with a deep understanding of the subject matter and skills to apply these concepts to real world problems.
Communication

- The course webpage will be updated regularly throughout the semester with lecture notes, programming and reading assignments and important deadlines.

- We will be using Piazza for course related discussions and announcements. Please enroll the class on Piazza by following the link
  [http://piazza.com/class#spring2014/bbm406](http://piazza.com/class#spring2014/bbm406)

Prerequisites

- Basic algorithms, data structures.
- Basic probability and statistics.
- Basic linear algebra and calculus.
- Good programming skills.

Reference Books

- Bayesian Reasoning and Machine Learning, Barber, Cambridge University Press, 2012. (online version available)
- Pattern Recognition and Machine Learning, Bishop, Springer, 2006

Grading Policy

- 30% Problem Sets
- 30% Midterm Exam
- 40% Final Exam
Problem Sets

- There will be 3 problem sets.
- Each one will involve both theoretical and practical exercises.
- Each assignment have to be done **individually** or in **pairs**.
- Start your problem sets early!

Course Outline

- **Week 1**  Overview of Machine Learning
- **Week 2**  Linear Regression, Least Squares  *PS1 out*
- **Week 3**  Machine Learning Methodology
- **Week 4**  Statistical Estimation
- **Week 5**  Classification  *PS1 due, PS2 out*
- **Week 6**  Support Vector Machines
- **Week 7**  Support Vector Machines (cont’d.)
- **Week 8**  **Midterm exam**
- **Week 9**  Decision Tree Learning  *PS2 due, PS3 out*
- **Week 10**  Ensemble Methods
- **Week 11**  Instance-based Methods
- **Week 12**  Neural Networks  *PS3 due*
- **Week 13**  Unsupervised Learning
- **Week 14**  Factor Analysis

Welcome to BBM 406!
This week

• An overview of Machine Learning

• Nearest-neighbor classifier

Machine Learning: An Overview

Two definitions of learning

(1) Learning is the acquisition of knowledge about the world.

Kupfermann (1985)

(2) Learning is an adaptive change in behavior caused by experience.

Shepherd (1988)
Empirical Inference

• Drawing conclusions from empirical data (observations, measurements)

• Example 1: Scientific inference

\[ y = \sum a_i k(x, x_i) + b \]

Leibniz, Weyl, Chaitin

Empirical Inference

• Example 2: Perception
What is machine learning? (by examples)

Collaborative Filtering

Don’t mix preferences on Netflix!

Amazon books

Imitation Learning in Games

Avatar learns from your behavior

Spam Filtering

Ham

Spam

Collaborative Filtering

Imitation Learning in Games
In May 1997, an IBM supercomputer known as Deep Blue beat then chess world champion Garry Kasparov, who had once bragged he would never lose to a machine.

Kasparov and other chess masters blamed the defeat on a single move made by the IBM machine. Either at the end of the first game or the beginning of the second, depending on who’s telling the story, the computer made a sacrifice that seemed to hint at its long-term strategy.

Kasparov and many others thought the move was too sophisticated for a computer, suggesting there had been some sort of human intervention during the game. “It was an incredibly refined move, of defending while ahead to cut out any hint of countermoves,” grandmaster Yasser Seirawan told Wired.com in 2001. “And it sent Garry into a tizzy.”

Fifteen years after the historical match, one of Big Blue’s designers says the move was the result of a bug in Deep Blue’s software.

http://www.wired.co.uk/news/archive/2012-10/01/deep-blue-bug
**Google Self-Driving Cars**

- Google's self-driving car passes 300,000 miles (Forbes, 8/15/2012)

**Search Ads**

- Why these ads?

**Weather Prediction**

- Weather prediction revisited
  - Temperature: 72° F

**Speech Recognition**

- Given an audio waveform, robustly extract & recognize any spoken words
  - Statistical models can be used to
    - Provide greater robustness to noise
    - Adapt to accent of different speakers
    - Learn from training
I need to hide a body
noun, verb, preposition, …

Visual Scene Understanding

sky
skyscraper
sky
dome

Learning - revisited

Learning knowledge
data
**Programming with Data**

- Want adaptive robust and fault tolerant systems
- Rule-based implementation is (often)
  - difficult (for the programmer)
  - brittle (can miss many edge-cases)
  - becomes a nightmare to maintain explicitly
  - often doesn’t work too well (e.g. OCR)
- Usually easy to obtain examples of what we want
  IF \( x \) THEN DO \( y \)
- Collect many pairs \( (x_i, y_i) \)
- Estimate function \( f \) such that \( f(x_i) = y_i \) (supervised learning)
- Detect patterns in data (unsupervised learning)

**Objectives of Machine Learning**

- **Algorithms:** design of efficient, accurate, and general learning algorithms to
  - deal with large-scale problems.
  - make accurate predictions (unseen examples).
  - handle a variety of different learning problems.
- **Theoretical questions:**
  - what can be learned? Under what conditions?
  - what learning guarantees can be given?
  - what is the algorithmic complexity?
Definitions and Terminology

• **Example:** an object, instance of the data used.

• **Features:** the set of attributes, often represented as a vector, associated to an example (e.g., height and weight for gender prediction).

• **Labels:** in classification, category associated to an object (e.g., positive or negative in binary classification); in regression real value.

• **Training data:** data used for training learning algorithm (often labeled data).

Supervised Learning Example - SPAM Detection

• **Input:** email
• **Output:** spam/ham
• **Setup:**
  – Get a large collection of example emails, each labeled “spam” or “ham”
  – Note: someone has to hand label all this data!
  – Want to learn to predict labels of new, future emails

  • **Features:** The attributes used to make the ham / spam decision
    – Words: FREE!
    – Text Patterns: $dd, CAPS
    – Non-text: SenderInContacts

  Dear Sir:
  First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidential and top secret.
  
  Ok, know this is blatantly OT but I’m beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use. I know it was working prev being stuck in the corner, but when I plugged it in, til the power nothing happened.

Definitions and Terminology (cont’d.)

• **Test data:** data used for testing learning algorithm (unlabeled data).

• **Unsupervised learning:** no labeled data.

• **Supervised learning:** uses labeled data.

• **Semi-supervised learning and transduction:** intermediate scenarios.

Unsupervised Learning Example - Segmentation
Important challenges in ML

• How important is the actual learning algorithm and its tuning
• Simple versus complex algorithm
• Overfitting
• Model Selection
• Regularization

The basic machine learning framework

\[ y = f(x) \]

• **Learning**: given a *training set* of labeled examples \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \), estimate the parameters of the prediction function \( f \)
• **Inference**: apply \( f \) to a never before seen *test example* \( x \) and output the predicted value \( y = f(x) \)

Concept Learning

• **Definition**: Acquire an operational definition of a general category of objects given positive and negative training examples.
• Also called binary classification, binary supervised learning

Your 1st Classifier: Nearest Neighbor Classifier
**Concept Learning Example**

<table>
<thead>
<tr>
<th>correct (complete, partial, guessing)</th>
<th>color (yes, no)</th>
<th>original (yes, no)</th>
<th>presentation (clear, unclear, cryptic)</th>
<th>binder (yes, no)</th>
<th>A+</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 complete</td>
<td>yes</td>
<td>yes</td>
<td>clear</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>2 complete</td>
<td>no</td>
<td>yes</td>
<td>clear</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>3 partial</td>
<td>yes</td>
<td>no</td>
<td>unclear</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4 complete</td>
<td>yes</td>
<td>yes</td>
<td>clear</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

- **Instance Space X**: Set of all possible objects describable by attributes (often called features).
- **Concept c**: Subset of objects from X (c is unknown).
- **Target Function f**: Characteristic function indicating membership in c based on attributes (i.e. label) (f is unknown).
- **Training Data S**: Set of instances labeled with target function.

---

**Concept Learning as Learning a Binary Function**

- **Task**: Learn (to imitate) a function \( f: X \rightarrow \{+1,-1\} \)
- **Training Examples**: Learning algorithm is given the correct value of the function for particular inputs \( \rightarrow \) training examples
  - An example is a pair \((x, y)\), where \(x\) is the input and \(y = f(x)\) is the output of the target function applied to \(x\).
- **Goal**: Find a function \( h: X \rightarrow \{+1,-1\} \)
  - that approximates \( f: X \rightarrow \{+1,-1\} \)
  - as well as possible.

---

**Supervised Learning**

- **Task**: Learn (to imitate) a function \( f: X \rightarrow Y \)
- **Training Examples**: Learning algorithm is given the correct value of the function for particular inputs \( \rightarrow \) training examples
  - An example is a pair \((x, f(x))\), where \(x\) is the input and \(y = f(x)\) is the output of the target function applied to \(x\).
- **Goal**: Find a function
  - that approximates \( h: X \rightarrow Y \)
  - as well as possible.

---

**Nearest Neighbors**

- **Table lookup**
  - For previously seen instance remember label
- **Nearest neighbor**
  - Pick label of most similar neighbor
  - Slight improvement - use k-nearest neighbors
  - Really useful baseline!
  - Easy to implement for small amounts of data. Why?
K-Nearest Neighbor (kNN)

- Given: Training data ( \( x_1, y_1 \), ..., \( x_n, y_n \) )
  - Attribute vectors: \( x_i \in X \)
  - Labels: \( y_i \in Y \)
- Parameter:
  - Similarity function: \( K : X \times X \rightarrow R \)
  - Number of nearest neighbors to consider: \( k \)
- Prediction rule
  - New example \( x' \)
  - K-nearest neighbors: \( k \) train examples with largest \( K(x_i, x') \)

\[
k(x', y')(\text{A+}) = \arg \max_{y \in \text{labels}} \sum_{i=1}^{k} 1_{\|x_i - x'\| < \|x_j - x'\|}
\]

<table>
<thead>
<tr>
<th>correct (complete, partial, guessing)</th>
<th>color (yes, no)</th>
<th>original (yes, no)</th>
<th>presentation (clear, unclear, cryptic)</th>
<th>binder (yes, no)</th>
<th>A+</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 complete</td>
<td>yes</td>
<td>yes</td>
<td>clear</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>2 complete</td>
<td>no</td>
<td>yes</td>
<td>clear</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>3 partial</td>
<td>yes</td>
<td>no</td>
<td>unclear</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4 complete</td>
<td>yes</td>
<td>yes</td>
<td>clear</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

- How will new examples be classified?
  - Similarity function?
  - Value of \( k \)?

Example - Document Classification

![Example - Document Classification](image1)

![Example - Document Classification](image2)
Example - Document Classification

Recall: Vector Space Representation

Each document is a vector, one component for each term (= word).

\[
\begin{align*}
\text{Doc 1} & \quad \text{Doc 2} & \quad \text{Doc 3} \\
\text{Word 1} & \quad 300 & \quad \ldots \\
\text{Word 2} & \quad 081 & \quad \ldots \\
\text{Word 3} & \quad 12 & \quad 1 & \quad 10 & \quad \ldots
\end{align*}
\]

... component for each term (= word).

Normalize to unit length.

High-dimensional vector space:

Terms are axes, 10,000+ dimensions, or even 100,000+.

Docs are vectors in this space.

Example - Document Classification

Test Document = ?

Sports

Science

Arts

If we get more data

- 1 Nearest Neighbor
  - Converges to perfect solution if clear separation
  - Twice the minimal error rate $2p(1-p)$ for noisy problems
- k-Nearest Neighbor
  - Converges to perfect solution if clear separation (but needs more data)
  - Converges to minimal error $\min(p, 1-p)$ for noisy problems if k increases
Weighted K-Nearest Neighbor

- Given: Training data ( \( (x_1, y_1), \ldots, (x_n, y_n) \))
  - Attribute vectors: \( x_i \in X \)
  - Target attribute \( y_i \in Y \)
- Parameter:
  - Similarity function: \( K : X \times X \rightarrow R \)
  - Number of nearest neighbors to consider: \( k \)
- Prediction rule
  - New example \( x' \)
  - K-nearest neighbors: \( k \) train examples with largest \( K(x_i, x') \)

\[
K(x', x) = \max_{i \leq n} \left( \sum_{j=1}^{k} l_{y_j = y} K(x_j, x') \right)
\]
• Torralba et al., 80 million tiny images: a large dataset for non-parametric object and scene recognition, IEEE TPAMI, 2008

Nearest neighbors on images

Overview of Nearest Neighbors

• Very simple method
• Retain all training data
  - Can be slow in testing
  - Finding NN in high dimensions is slow
• Metrics are very important
• Good baseline

Retain all training data
Finding NN in high dimensions is slow

Nearest neighbor search in higher dimensions

Linear Search:
e.g. scanning 4.5M images!

k-D trees:
axis parallel partitions of the data
Only effective in low-dimensional data

Large Scale Approximate Indexing
Locality Sensitive Hashing (LSH), Spill-Tree, NV-Tree
All above run on a single machine with all data in memory, and scale to big data

Web-scale Approximate Indexing
Parallel variant of Spill-tree, NV-tree on distributed systems,
Scale to Billions of images in disks on multiple machines