Class Overview and General Introduction to Machine Learning

Piyush Rai
www.cs.utah.edu/~piyush

CS5350/6350: Machine Learning

August 23, 2011
Course Logistics

- **Class webpage**: [http://www.eng.utah.edu/~cs5350/](http://www.eng.utah.edu/~cs5350/)

- **Class mailing lists**
  - cs5350@list.eng.utah.edu (SUBSCRIBE but don’t post)
  - teach-cs5350@list.eng.utah.edu (POST but don’t subscribe)

- **Requirements and Grading:**
  - Six homeworks worth a total of 60% (10% each)
    - Each consisting of some written problems and some programming assignments
    - Programming assignments to be done in MATLAB (or Octave)
    - Homeworks due *before* class time on the due date
  - Final exam worth 20%
  - Final Class Project worth 20%

- **Textbook**: No official textbook required
  - Required reading material provided on the class webpage
  - Some of the material is password-protected
Course Goals

By the end of the semester, you should be able to:

- **Understand** how various machine learning algorithms work
- **Implement** them (and, hopefully, their variants/improvements) on your own
- Look at a real-world problem and **identify** if ML is an appropriate solution
- If so, identify what types of algorithms might be applicable
- **Apply** those algorithms
- **Feel inspired** to work on and **learn more** about Machine Learning :-)

**This class is not** about:
- Introduction to machine learning tools/softwares
What is Machine Learning?

- **Machine Learning:**
  - Designing algorithms that can learn *patterns* from data (and exploit them)
  - **Approach:** human supplies training examples, the machine learns
  - **Example:** Show the machine a bunch of *spam* and *legitimate* emails and let it learn to predict if a new email is spam or not

Machine Learning primarily uses the *statistically motivated* approach
- No hand-crafted rules - subtle pattern nuances are often be difficult to specify
- Instead, let the machine figure out the rules on its own by looking at data
  - .. by building statistical models of the data

The statistical model helps uncover the process which generated the data

- **Desirable Property:** *Generalization*
  - The model shouldn’t *overfit* on the training data
  - It should *generalize* well on *unseen* (future) *test data*
Generalization (Pictorially)

Pictures below: The $X$ axis is the input. The $Y$ axis is the response.

- Which of the four red curves fits the data (blue dots) best?
- Which curve is expected to generalize the best?
- Are they both the same? If yes, why? If no, why not?

**Lesson:** Simple models should be preferred over complicated models

- Simple models can prevent overfitting
- **Caution:** Too simple a model can underfit (e.g., $M = 0$ above)
- **General guideline:** Choose a model not-too-simple, yet not-too-complex
Machine Learning in the real-world

Broadly applicable in many domains (e.g., finance, robotics, bioinformatics, vision, natural language, etc.). Some applications:

- Spam filtering
- Speech/handwriting recognition
- Object detection/recognition
- Weather prediction
- Stock market analysis
- Search engines (e.g., Google)
- Ad placement on websites
- Adaptive website design
- Credit-card fraud detection
- Webpage clustering (e.g., Google News)
- Machine Translation (e.g., Google Translate)
- Recommendation systems (e.g., Netflix, Amazon)
- Classifying DNA sequences
- Automatic vehicle navigation
- Performance tuning of computer systems
- Predicting good compilation flags for programs
- .. and many more

12 IT skills that employers can’t say no to (Machine Learning is #1)

http://www.computerworld.com/s/article/9026623/12_IT_skills_that_employers_can_t_say_no_to_
Major Machine Learning Paradigms

**Nomenclature:** $x$ denotes an input/example/instance, $y$ denotes a response/output/label/prediction

- **Supervised Learning:** learning with a teacher
  - Given: $N$ labeled training examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$
  - Goal: learn mapping $f$ that predicts label $y$ for a test example $x$
  - Example: Spam classification, webpage categorization

- **Unsupervised Learning:** learning without a teacher
  - Given: a set of $N$ unlabeled inputs $\{x_1, \ldots, x_N\}$
  - Goal: learn some intrinsic structure in the inputs (e.g., groups/clusters)
  - Example: Automatically grouping news stories (Google News)

- **Reinforcement Learning:** learning by interacting
  - Given: an agent acting in an environment (having a set of states)
  - Goal: learn a policy (state to action mapping) that maximizes agent’s reward
  - Example: Automatic vehicle navigation, (computer) learning to play Chess
Supervised Learning

- **Given:** \( N \) labeled training examples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \)

- **Goal:** learn a model that predicts the label \( y \) for a test example \( x \)

- **Assumption:** The training and the test examples are drawn from the same data distribution

- **Things to keep in mind:**
  - No single learning algorithm is universally good ("no free lunch")
  - Different learning algorithms work with different assumptions
  - Generalization is particularly important for supervised learning
Supervised Learning: Problem Settings

- \( f : x \rightarrow y \)

- **Classification:** when \( y \) is a discrete variable
  - Discrete variable: takes a value from a discrete set \( y \in \{1, \ldots, K\} \)
  - **Example:** Category of a webpage (sports, politics, business, science, etc.)

- **Regression:** when \( y \) is a real-valued variable
  - **Example:** Price of a stock
Problem Types:

- **Binary Classification**: $y$ is binary (two classes: 0/1 or -1/+1)
  - **Example**: Spam Filtering (tell whether this email is spam or legitimate)

- **Multi-class Classification**: $y$ is discrete with one of $K > 2$ possible values
  - **Example**: Predicting your CS5350 grade (e.g., $A, A-, B+, B, B-$, other)

- **Multi-label Classification**: When $y$ is a vector of discrete variables
  - Each input $x$ has multiple labels
  - Each element of $y$ is one label (individual labels can be binary/multi-class)
  - **Example**: Image annotation (each image can have multiple labels)

- **Structured Prediction**: When $y$ is a vector with a structure
  - Elements of $y$ are not independent but related to each-other
  - **Example**: Predicting parts-of-speech (POS) tags for a sentence

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>The man ate the really tasty sandwich</td>
<td></td>
</tr>
<tr>
<td>DET NOUN VERB DET ADV ADJ NOUN</td>
<td></td>
</tr>
</tbody>
</table>
Supervised Learning: Regression

Problem Types:

- **Univariate Regression**: \( y \) is a single real-valued number
  - **Example**: Predicting the future price of a stock

- **Multivariate Regression**: \( y \) is a real-valued vector
  - Each element of \( y \) tells the value of one response variable
  - **Example**: Torque values in multiple joints of a robotic arm
  - Akin to multi-label classification
Supervised Learning: Pictorially

Classification is about finding separation boundaries (linear/non-linear):

Regression is more like fitting a curve/surface to the data:
Unsupervised Learning

- **Unsupervised Learning**: learning without a teacher
  - Given: a set of unlabeled inputs \( \{x_1, \ldots, x_N\} \)
  - Goal: learn some intrinsic structure in the data

- **Some Examples**: Data Clustering, Dimensionality Reduction

- **Data Clustering**
  - Grouping a given set of inputs based on their similarities
  - **Example**: clustering new stories based on their topics (e.g., Google News)
  - Clustering sometimes is also referred to as (probability) density estimation

- **Dimensionality Reduction**
  - Often, real-world data is high dimensional
  - Reducing dimensionality helps in several ways
  - **Computational benefits**: speeding up learning algorithms
  - **Better input representations** for supervised learning tasks
  - Used for data visualization by reducing data to smaller dimensions
Unsupervised Learning: Data Clustering
Unsupervised Learning: Dimensionality Reduction

- Data high-dimensional in ambient space, but intrinsically lower dimensional
- 2-D data lying close to 1-D space

3-D data living on a manifold, intrinsically 2-D
Unlike supervised/unsupervised learning, **RL does not receive examples**

Rather, it learns (gathers experience) by interacting with the world

Defined by an **agent** and an **environment** the agent acts in

Agent has a set $\mathcal{A}$ of **actions**, environment has a set $\mathcal{S}$ of **states**

**Goal:** Find a **sequence of actions** by the agent that **maximizes** its **reward**

**Output:** A **policy** which maps states to actions

RL problems always include **time** as a variable

**Example problems:** Chess, Robot control, autonomous driving

In RL, the key trade-off is **exploration** versus **exploitation**
Other Paradigms: Semi-supervised Learning

- Supervised Learning requires labeled data (the more, the better!)

- Problem 1: Labeling is expensive (usually done by humans)

- Problem 2: Sometimes labels are really hard to get
  - Speech-analysis: transcribing an hour of speech can take several hundred hours!

- How can we learn well even with small amounts of labeled data?

- One answer: **Semi-supervised Learning**
  - Using small amount of labeled + plenty of (freely available) unlabeled data
Other Paradigms: Semi-supervised Learning

- Often unlabeled data can give a good idea about class separation

- One intuition: Class boundary is expected to lie in a low-density region
  - Low density region: region that has very few examples

from [Semi-Supervised Learning, ICML 2007 Tutorial; Xiaojin Zhu]
Other Paradigms: Active Learning

- Similar motivation as semi-supervised learning (saving data labeling cost)

- Standard supervised learning is **passive**
  
  - Learner has no choice for the data it has to learn from
  
  - Not all labeled examples are really informative
  
  - Spending labeling efforts on uninformative examples isn’t really worth it

- **Active Learning:** allows the learner to ask for specific labeled examples
  
  - .. the ones it considers the most informative

- Active Learning can lead to several benefits:
  
  - Less labeled data needed to learn
  
  - Better classifiers
Other Paradigms: Transfer Learning

- Let’s assume we have two related learning tasks ‘A’ and ‘B’
  - Plenty of labeled training data for ‘A’: Can learn ‘A’ well
  - Little or no labeled data for ‘B’: Little or no hope of learning ‘B’

- **Transfer Learning**: allows ‘B’ to leverage the data from task ‘A’
  - Under suitable task-relatedness assumptions, transfer learning may help
  - **Caution**: Incorrect/inappropriate assumptions can hurt learning

- Several variants/names of Transfer Learning
  - Multitask Learning
  - Domain Adaptation
  - Co-variate Shift
Bayesian Learning

- Not really a different learning paradigm
  - Rather, a way of doing machine learning (can be used for any learning paradigm - supervised, unsupervised, etc.)

- Most ML algorithms: Provide them data, get a model out of it
  - No way to know how confident your model parameters are
  - No way to know how confident your predictions are

- But in some problem domains, confidence estimates are important

- Bayesian Learning gives a way to quantify confidence/uncertainty
  - By maintaining a probability distribution over the parameters/predictions
  - So we also have mean and variance estimates of the parameters/predictions

- Another advantage: Incorporating prior knowledge about the problem, Bayesian methods can automatically control overfitting (and can learn well with small amounts of data)
Machine Learning vs Statistics

- Traditionally, Statistics mainly cares about fitting a model over the data
  - Main focus is on explaining the data
  - Issues such as generalization are typically ignored
  - Note: There may be some exceptions

- ML focuses more on the prediction aspect (generalization is important)
  - Although knowing about the data generating model can help prediction, such modeling can sometimes be expensive. ML therefore often goes easy on the modeling aspect and focuses directly on the prediction task

- Statistics traditionally does not focus much on computational issues

- Most ML algorithms nowadays consider the computational issues

- For some discussion, see:
  
Data Representation

Data has form: \{({x_1, y_1}, \ldots, (x_N, y_N))\} (labeled), or \{x_1, \ldots, x_N\} (unlabeled)

- What the label \(y\) looks like is task-specific (as we saw)

- What about \(x\) which denotes a real-world object (e.g., image or text document)?

  - Each example \(x\) is a set of (numeric) features/attributes/dimensions

  - Features encode properties of the object which \(x\) represents

  - \(x\) is commonly represented as a \(D \times 1\) vector

- Representing a 28 \times 28 image: \(x\) can be a 784 \times 1 vector of pixel values

- Representing a text document: \(x\) can be a vector of word-counts of words appearing in that document

- For some problems, non-vectorial representations may be more appropriate
Some Notations

- $\mathbb{R}^D$ denotes the set of all $D \times 1$ real-valued column vectors
- $\mathbf{x} \in \mathbb{R}^D$ denotes a $D \times 1$ real-valued column vector
- $\mathbf{x}^T$ denotes the transpose of $\mathbf{x}$, a $1 \times D$ row vector
- $\mathbb{R}^{N \times D}$ denotes the set of all $N \times D$ real-valued matrices
- $\mathbf{X} \in \mathbb{R}^{N \times D}$ denotes an $N \times D$ real-valued matrix

Supervised Learning: Often, we write $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ as $(\mathbf{X}, \mathbf{Y})$

- $\mathbf{X}$ is an $N \times D$ matrix
- Each row of $\mathbf{X}$ denotes an example, each column denotes a feature
- $x_{ij}$ denotes the $j$-th feature of the $i$-th example
- $\mathbf{Y}$ is an $N \times 1$ vector. Row $i$ denotes the label of the $i$-th example
Next class..

- Two supervised learning algorithms
  - *K*-Nearest Neighbors
  - Decision Trees
  - Both based more on intuition and less on maths :)