CS 2750 Machine Learning
Lecture 1

Machine Learning

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square, x4-8845

http://www.cs.pitt.edu/~milos/courses/cs2750/

Administration

Study material

• Handouts, your notes and course readings
• Primary textbook:
• Recommended book:
• Other books:
Administration

- **Lectures:**
  - Random short quizzes testing the understanding of basic concepts from previous lectures
- **Homeworks: weekly**
  - Programming tool: Matlab (CSSD machines and labs)
  - Matlab Tutorial: next week
- **Exams:**
  - Midterm (March)
- **Final project:**
  - Proposals (early March)
  - Written report + Oral presentation (end of the semester)

Tentative topics

- Concept learning.
- Density estimation.
- Linear models for regression and classification.
- Learning Bayesian networks.
- Clustering. Latent variable models.
- Dimensionality reduction. Feature extraction.
- Hidden Markov models.
- Reinforcement learning
Machine Learning

• The field of **machine learning** studies the design of computer programs (agents) capable of learning from past experience or adapting to changes in the environment

• The need for building agents capable of learning is everywhere
  – predictions in medicine,
  – text and web page classification,
  – speech recognition,
  – image/text retrieval,
  – commercial software

Learning

**Learning process:**
Learner (a computer program) processes data \( D \) representing past experiences and tries to either develop an appropriate response to future data, or describe in some meaningful way the data seen

**Example:**
Learner sees a set of patient cases (patient records) with corresponding diagnoses. It can either try:
  – to predict the presence of a disease for future patients
  – describe the dependencies between diseases, symptoms
Types of learning

- **Supervised learning**
  - Learning mapping between input $x$ and desired output $y$
  - Teacher gives me $y$’s for the learning purposes

- **Unsupervised learning**
  - Learning relations between data components
  - No specific outputs given by a teacher

- **Reinforcement learning**
  - Learning mapping between input $x$ and desired output $y$
  - Critic does not give me $y$’s but instead a signal (reinforcement) of how good my answer was

- **Other types of learning:**
  - explanation-based learning, etc.

---

**Supervised learning**

**Data:** $D = \{d_1, d_2, \ldots, d_n\}$ a set of $n$ examples

\[ d_i = \langle x_i, y_i \rangle \]

$x_i$ is input vector, and $y_i$ is desired output (given by a teacher)

**Objective:** learn the mapping $f : X \rightarrow Y$

\[ y_i = f(x_i) \quad \text{for all} \quad i = 1, \ldots, n \]

**Two types of problems:**

- **Regression:** $X$ discrete or continuous $\rightarrow$ $Y$ is **continuous**
- **Classification:** $X$ discrete or continuous $\rightarrow$ $Y$ is **discrete**
Supervised learning examples

• **Regression:** Y is **continuous**
  
  Debt/equity  
  Earnings  
  Future product orders  
  → company stock price

• **Classification:** Y is **discrete**
  
  Handwritten digit (array of 0,1s)  
  → Label “3”

Unsupervised learning

• **Data:** \( D = \{ d_1, d_2, \ldots, d_n \} \)
  
  \( d_i = x_i \)  
  vector of values  
  No target value (output) y

• **Objective:**
  – learn relations between samples, components of samples

Types of problems:

• **Clustering**
  Group together “similar” examples, e.g. patient cases

• **Density estimation**
  – Model probabilistically the population of samples
Unsupervised learning example.

• **Density estimation.** We want to build the probability model of a population from which we draw samples $d_j = x_j$

\begin{center}
\includegraphics[width=\textwidth]{density_estimation.png}
\end{center}

Unsupervised learning. Density estimation

• A probability density of a point in the two dimensional space
  – Model used here: **Mixture of Gaussians**

\begin{center}
\includegraphics[width=\textwidth]{mixture_of_gaussians.png}
\end{center}
Reinforcement learning

- We want to learn: \( f : X \rightarrow Y \)
- We see samples of \( x \) but not \( y \)
- Instead of \( y \) we get a feedback (reinforcement) from a critic about how good our output was

![Diagram of Reinforcement Learning]

- The goal is to select outputs that lead to the best reinforcement

Learning

- Assume we see examples of pairs \((x, y)\) and we want to learn the mapping \( f : X \rightarrow Y \) to predict future \( y \)s for values of \( x \)
- We get the data what should we do?

![Graph of Data Points]
Learning bias

- **Problem:** many possible functions $f : X \rightarrow Y$ exists for representing the mapping between $x$ and $y$
- Which one to choose? Many examples still unseen!

![](image1)

Learning bias

- Problem is easier when we make an assumption about the model, say, $f(x) = ax + b + \epsilon$
  \[ \epsilon = N(0, \sigma) \] - random (normally distributed) noise
- Restriction to a linear model is an example of learning bias

![](image2)
Learning bias

- **Bias** provides the learner with some basis for choosing among possible representations of the function.
- **Forms of bias:** constraints, restrictions, model preferences
- **Important:** There is no learning without a bias!

![Graph showing linear relationship between X and Y]

Learning bias

- Choosing a parametric model or a set of models is not enough. Still too many functions $f(x) = ax + b + \epsilon \quad \epsilon = \mathcal{N}(0, \sigma)$
  - One for every pair of parameters $a, b$
Fitting the data to the model

- We are interested in finding the **best set** of model parameters

**Objective:** Find the set of parameters that:
- reduces the misfit between the model and observed data
- Or, (in other words) that explain the data the best

**Error function:**
**Measure of misfit between the data and the model**

- **Examples of error functions:**
  - Average square error
    \[
    \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2
    \]
  - Average misclassification error
    \[
    \frac{1}{n} \sum_{i=1}^{n} 1_{y_i \neq f(x_i)}
    \]

  Average # of misclassified cases

---

**Fitting the data to the model**

- **Linear regression**
  - Least squares fit with the linear model
  - minimizes
    \[
    \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2
    \]
Typical learning

Three basic steps:

- **Select a model** or a set of models (with parameters)
  E.g. \( y = ax + b + \varepsilon \) \( \varepsilon = N(0, \sigma) \)

- **Select the error function** to be optimized
  E.g. \( \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 \)

- **Find the set of parameters optimizing the error function**
  - The model and parameters with the smallest error represent the best fit of the model to the data

But there are problems one must be careful about …

Learning

**Problem**

- We fit the model based on past experience (past examples seen)
- But ultimately we are interested in learning the mapping that performs well on the whole population of examples

**Training data:** Data used to fit the parameters of the model

**Training error:** \( \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 \)

**True (generalization) error** (over the whole unknown population):
\( E_{(x,y)}[(y - f(x))^2] \)  Mean squared error

**Training error tries to approximate the true error !!!**
Does a good training error imply a good generalization error ?
Overfitting

• Assume we have a set of 10 points and we consider polynomial functions as our possible models

Overfitting

• Fitting a linear function with the square error
• Error is nonzero
Overfitting

- Linear vs. cubic polynomial
- Higher order polynomial leads to a better fit, smaller error

Overfitting

- Is it always good to minimize the error of the observed data?
Overfitting

• For 10 data points, the degree 9 polynomial gives a perfect fit (Lagrange interpolation). Error is zero.
• Is it always good to minimize the training error?

More important: How do we perform on the unseen data?
**Overfitting**

*Situation* when the training error is low and the generalization error is high. Causes of the phenomenon:

- Model with a large number of parameters (degrees of freedom)
- Small data size (as compared to the complexity of the model)

---

**How to evaluate the learner’s performance?**

- **Generalization error** is the true error for the population of examples we would like to optimize

\[
E_{(x,y)}[(y - f(x))^2]
\]

- But it cannot be computed exactly
- **Sample mean only approximates the true mean**

- Optimizing (mean) training error can lead to the overfit, i.e. training error may not reflect properly the generalization error

\[
\frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2
\]

- So how to test the generalization error?
**How to evaluate the learner’s performance?**

- **Generalization error** is the true error for the population of examples we would like to optimize
  \[ E_{(x,y)}[(y - f(x))^2] \]
- Sample mean only approximates it
- How to measure the generalization error?
- **Two ways:**
  - **Theoretical:** Law of large numbers
    - statistical bounds on the difference between true and sample mean errors
  - **Practical:** Use a separate data set with \( m \) data samples to test
    - (Mean) test error
    \[
    \frac{1}{m} \sum_{j=1}^{m} (y_j - f(x_j))^2
    \]

---

**Basic experimental setup to test the learner’s performance**

1. Take a dataset \( D \) and divide it into:
   - Training data set
   - Testing data set
2. Use the training set and your favorite ML algorithm to train the learner
3. Test (evaluate) the learner on the testing data set

- The results on the testing set can be used to compare different learners powered with different models and learning algorithms
Solutions for overfitting

How to make the learner avoid overfitting?

• **Assure sufficient number of samples** in the training set
  – May not be possible

• **Hold some data out of the training set = validation set**
  – Train (fit) on the training set (w/o data held out);
  – Check for the generalization error on the validation set,
    choose the model based on the validation set error
    (cross-validation techniques)

• **Regularization (Occam’s Razor)**
  – Penalize for the model complexity (number of parameters)
  – Explicit preference towards simple models

---

Design of a learning system (first view)

```
Data

Model selection

Learning

Application or Testing
```
Design of a learning system.

1. **Data:** $D = \{d_1, d_2, \ldots, d_n\}$

2. **Model selection:**
   - **Select a model** or a set of models (with parameters)
     
     E.g. $y = ax + b + \epsilon \quad \epsilon = N(0, \sigma)$
   - **Select the error function** to be optimized
     
     E.g. $\frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$

3. **Learning:**
   - **Find the set of parameters optimizing the error function**
     
     – The model and parameters with the smallest error

4. **Application:**
   - **Apply the learned model**
     
     – E.g. predict $y$s for new inputs $x$ using learned $f(x)$