Machine Learning

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

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

Administration

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

TA:
   Zhipeng (Patrick) Luo
   ZHL78@pitt.edu
   6509 Sennott Square

Office hours: TBA
Administration

Study material

• Handouts, your notes and course readings
• Primary textbook:


Other books:

– J. Han, M. Kamber. Data Mining. Morgan Kauffman, 2011.
Administration

- **Homeworks:** weekly
  - **Programming tool:** Matlab (CSSD machines and labs)
  - **Matlab Tutorial:** next week
- **Exams:**
  - **Midterm** (March)
  - **Final** (April 13-17)
- **Final project:**
  - **Written report + Oral presentation**
    (April 20-24)
- **Lectures:**
  - **Attendance and Activity**

Tentative topics

- Introduction to Machine Learning
- **Density estimation.**
- **Supervised Learning.**
  - Linear models for regression and classification.
- **Unsupervised Learning.**
  - Learning Bayesian networks.
  - Latent variable models. Expectation maximization.
  - Clustering
Tentative topics (cont)

- **Dimensionality reduction.**
  - Feature extraction.
  - Principal component analysis (PCA)
- **Ensemble methods.**
  - Mixture models.
  - Bagging and boosting.
- **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:**
  - Concept learning, Active learning, Transfer learning, Deep learning
Supervised learning

**Data:** \( D = \{d_1, d_2, \ldots, d_n\} \) \hspace{1em} a set of \( n \) examples
\( d_i = \langle x_i, y_i \rangle \)
\( x_i \) is input vector, and \( y \) is desired output (given by a teacher)

**Objective:** learn the mapping \( f : X \rightarrow Y \)
\( s.t. \ \ y_i \approx f(x_i) \) \hspace{.5em} for all \( i = 1, \ldots, n \)

**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 \( \rightarrow \) company stock price

- **Classification:** \( Y \) is **discrete**

  Handwritten digit (array of 0,1s) \( \rightarrow \) Label “3”
Unsupervised learning

- **Data:**  
  \[ D = \{d_1, d_2, \ldots, d_n\} \]
  
  \[ d_i = x_i \quad \text{vector of values} \]
  
  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

- **Clustering.** Group together similar examples  
  \[ d_i = x_i \]
Unsupervised learning example

• **Clustering.** Group together similar examples \( d_i = x_i \)

![Image of clustering example]

Unsupervised learning example

• **Density estimation.** We want to build the probability model \( P(x) \) of a population from which we draw examples \( d_i = x_i \)

![Image of density estimation example]
Unsupervised learning. Density estimation

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

Reinforcement learning

- We want to learn: \( f : X \to Y \)
- 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

- The goal is to select outputs that lead to the best reinforcement
Learning: first look

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

Learning: first look

- **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!
Learning: first look

- Solution: make an assumption about the model, say,
  \[ f(x) = ax + b \]

Learning: first look

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

- We want the **best set** of model parameters

**Objective:** Find parameters that:
- reduce the misfit between the model $M$ and observed data $D$
- Or, (in other words) explain the data the best

**Objective function:**
- **Error function:** Measures the misfit between $D$ and $M$
- **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 problem**
  - Minimizes the squared error function for the linear model
  - minimizes $\frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$
Learning: summary

Three basic steps:

• **Select a model** or a set of models (with parameters)
  E.g. \( f(x) = ax + b \)

• **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 examples observed in \( D \)
• 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:

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

True (generalization) error (over the whole 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

- 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
- **Sample mean only approximates it**
- **Two ways to assess the generalization error is:**
  - **Theoretical:** Law of Large numbers
    * statistical bounds on the difference between the true and sample mean errors
  - **Practical:** Use a separate data set with \( m \) data samples to test the model
    * (Mean) test error \[
    \frac{1}{m} \sum_{j=1}^{m} (y_j - f'(x_j))^2
    \]

---

Testing of learning models

- **Simple holdout method**
  - Divide the data to the training and test data

  ![Diagram of the simple holdout method]

  - Typically 2/3 training and 1/3 testing
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
A learning system: basic cycle

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. \( y = ax + b \)

3. Choose the objective function
   - Squared error \( \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 \)

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

5. Testing:
   - Apply the learned model to new data
     - E.g. predict ys for new inputs \( x \) using learned \( f(x) \)
     - Evaluate on the test data