Designing a learning system

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

people.cs.pitt.edu/~milos/courses/cs2750/

---

Administrivia

- No homework assignment this week

- Please try to obtain a copy of Matlab:
  http://technology.pitt.edu/software/matlab-students

- A brief tutorial on Matlab next week
Learning: first look

Assume we get a dataset \( D \) that consists of pairs \((x, y)\)

**Goal:** learn the mapping \( f : X \rightarrow Y \) to be able to predict well \( y \) for some future \( x \).

**Question:** How do we learn \( f \) ?

---

**Learning: first look**

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 (error) function**
   - **Squared error** \( Error(D, a, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - ax_i - b)^2 \)
4. **Learning:**
   - Find the set of parameters \((a, b)\) optimizing the error function
   
   \( (a^*, b^*) = \arg \max_{(a,b)} Error(D, a, b) \)
5. **Application**
   - **Apply the learned model to new data** \( f(x) = a^* x + b^* \)
   - E.g. predict \( y \) for the new input \( x \)
Learning: first look

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 error function
   - Squared error

4. Learning:
   - Find the set of parameters \((a, b)\) optimizing the error function

5. Application
   - Apply the learned model to new data \( f(x) = ax + b \)
     E.g. predict \( y_s \) for the new input \( x \)
Learning: first look

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 (error) function
   - Squared error \[ Error(D, a, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - ax_i - b)^2 \]
4. Learning:
   - Find the set of parameters $(a, b)$ optimizing the error function
5. Application
   - Apply the learned model to new data
     - E.g. predict $y$s for the new input $x$
Learning: first look

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. 
3. Choose the objective (error) function
   
   - Squared error
4. Learning:
   
   - Find the set of parameters \((a, b)\) optimizing the error function
5. Application
   
   - Apply the learned model to new data \( f(x) = a^* x + b^* \)
   
   E.g. predict \(y\)s for the new input \(x\)

Looks straightforward, but there are problems ....
Learning: generalization error

We fit the model based on past examples observed in $D$.

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

**Training error:**

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

**Problem:** Ultimately we are interested in learning the mapping that performs well on the whole population of examples.

**True (generalization) error** (over the whole population):

$$\text{Error}(a,b) = E_{(x,y)}[(y - f(x))^2]$$

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. Why?

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

Assume in addition to a linear model: \( y = f(x) = ax + b \)
also: \( y = f(x) = a_3x^3 + a_2x^2 + a_1x + b \)
Which model would give us a smaller error for the least squares fit?

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?
Overfitting

- For 10 data points, degree 9 polynomial gives a perfect fit (Lagrange interpolation). Error is zero.
- Is it always good to minimize the training error? NO!!
- 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 the 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 assess 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 true and sample mean errors
  – **Practical:** Use a separate data set with \( m \) data samples to test the model
    • (Average) test error

\[ Error(D_{test}, f) = \frac{1}{m} \sum_{j=1}^{m} (y_j - f(x_j))^2 \]
Assessment of the generalization performance

- **Simple holdout method**
  - Divide the data into the disjoint training and test data

  ![Diagram of holdout method]

  - Typically 2/3 training and 1/3 testing

Testing of models: regression

- **Data set**
  - Training set
  - Test set

  ![Diagram of regression testing]

  - Learn on the training set
  - The model
  - Evaluate on the test set
Testing of models: classification

Data set

Training set

Learn on the training set

The model

Evaluate on the test set

Evaluation measures

Easiest way to evaluate the model:

- Error function used in the optimization is adopted also in the evaluation
- Advantage: may help us to see model overfitting. Simply compare the error on the training and testing data.

Evaluation of the models often considers:

- Other aspects or statistics of the model and its performance
- Moreover the Error function used for the optimization may be a convenient approximation of the quality measure we would really like to optimize
Evaluation measures

**Classification:**

<table>
<thead>
<tr>
<th></th>
<th>Actual</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Case</strong></td>
<td>TP 0.3</td>
<td>FP 0.1</td>
</tr>
<tr>
<td><strong>Control</strong></td>
<td>FN 0.2</td>
<td>TN 0.4</td>
</tr>
</tbody>
</table>

**Misclassification error:**

\[ E = FP + FN \]

**Sensitivity:**

\[ SN = \frac{TP}{TP + FN} \]

**Specificity:**

\[ SP = \frac{TN}{TN + FP} \]

A learning system: basic cycle

1. **Data:** \( D = \{d_1, d_2, ..., d_n\} \)
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/validation:**
   - **Evaluate on the test data**
6. **Application**
   - **Apply the learned model to new data** \( f(x) \)
A learning system: basic cycle

1. Data: $D = \{d_1, d_2, ..., d_n\}$
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
4. Learning:
   - Find the set of parameters optimizing the error function
   - The model and parameters with the smallest error
5. Testing/validation:
   - Evaluate on the test data
6. Application
   - Apply the learned model to new data $f(x)$
Steps taken when designing an ML system

- Data
- Model selection
- Choice of Error function
- Learning/optimization
- Evaluation
- Application

Add some complexity

- Data
- Data cleaning/preprocessing
- Feature selection/dimensionality reduction
- Model selection
- Choice of Error function
- Learning/optimization
- Evaluation
- Application
Data source and data biases

- Understand the data source
- Understand the data your models will be applied to
- Watch out for data biases:
  - Make sure the data we make conclusions on are the same as data we used in the analysis
  - It is very easy to derive “unexpected” results when data used for analysis and learning are biased

- Results (conclusions) derived for a biased dataset do not hold in general !!!

Data biases

**Example:** Assume you want to build an ML program for predicting the stock behavior and for choosing your investment strategy

**Data extraction:**
- pick companies that are traded on the stock market on January 2017
- Go back 30 years and extract all the data for these companies
- Use the data to build an ML model supporting your future investments

**Question:**
- Would you trust the model?
- Are there any biases in the data?
Steps taken when designing an ML system

- Data
- Data cleaning/preprocessing
- Feature selection/dimensionality reduction
- Model selection
- Choice of Error function
- Learning/optimization
- Evaluation
- Application

Data cleaning and preprocessing

Data you receive may not be perfect:
- Cleaning
- Preprocessing (conversions)

Cleaning:
- Get rid of errors, noise,
- Removal of redundancies

Preprocessing:
- Renaming
- Rescaling (normalization)
- Discretizations
- Abstraction
- Aggregation
- New attributes
Data preprocessing

• **Renaming** (relabeling) categorical values to numbers
  – dangerous in conjunction with some learning methods
  – numbers will impose an order that is not warranted


<table>
<thead>
<tr>
<th>Categorical Value</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>2</td>
</tr>
<tr>
<td>Normal</td>
<td>1</td>
</tr>
<tr>
<td>Low</td>
<td>0</td>
</tr>
<tr>
<td>True</td>
<td>2</td>
</tr>
<tr>
<td>False</td>
<td>1</td>
</tr>
<tr>
<td>Unknown</td>
<td>0</td>
</tr>
<tr>
<td>Red</td>
<td>✗</td>
</tr>
<tr>
<td>Blue</td>
<td>✗</td>
</tr>
<tr>
<td>Green</td>
<td>✗</td>
</tr>
</tbody>
</table>

• How to safely relabel the categorical values to numbers when no order exists?

• **Indicator vector (or one-hot) representation.**

• **Example: Red, Blue, Green colors**
  – Use a vector of binary (0,1) values of size 3 (= number of categories). E.g. Red is (1,0,0), Blue is (0,1,0) and Green is (0,0,1)

Data preprocessing

• **Rescaling (normalization):** continuous values transformed to some range, typically [-1, 1] or [0,1].

  ![Rescaling Diagram]

• Why normalization?
  – Some learning algorithms are sensitive to the values recorded in the specific input field and its magnitude
Data preprocessing

• **Discretizations (binning):** continuous values to a finite set of discrete values

• **Example:**

![Discretization Example](image)

• **Another Example:**

![Another Discretization Example](image)

Data preprocessing

• **Abstraction:** merge together categorical values

• **Aggregation:** summary or aggregation operations, such minimum value, maximum value, average etc.

• **New attributes:**
  – example: obesity-factor = weight/height