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, explanation-based learning, etc.
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?

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!
Learning bias

- Problem is easier when we make an assumption about the model, say, \( f(x) = ax + b \)
- Restriction to a linear model is an example of learning bias

![Graph showing learning bias](image)

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 learning bias](image)
Learning bias

- 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 \( a, b \)

![Graph showing fitting of data to the model](image)

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**:
  - Measures of misfit between the data and the model

- **Examples of error functions**:
  - Average squared 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. \( y = ax + b \)

- **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?

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?

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

- **Two ways to estimate generalization error:**
  
  - **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
    
    • **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

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. \( y = ax + b \)
   - **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 \( f(x) \)
Design cycle

Data
- Feature selection
- Model selection
- Learning
- Evaluation

Require prior knowledge

Data

Data may need a lot of:
- 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
  
  \[
  \begin{array}{cc}
  \text{High} & \rightarrow 2 \\
  \text{Normal} & \rightarrow 1 \\
  \text{Low} & \rightarrow 0
  \end{array}
  \quad \begin{array}{cc}
  \text{True} & \rightarrow 2 \\
  \text{False} & \rightarrow 1 \\
  \text{Unknown} & \rightarrow 0
  \end{array}
  \]

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

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

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
Data biases

• **Watch out for data biases:**
  – Try to understand the data source
  – 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 (pre-selected)

• **Results (conclusions) derived for biased data do not hold in general !!!**

---

Data biases

**Example 1: Risks in pregnancy study**

• Sponsored by DARPA at military hospitals

• Study of a large sample of pregnant woman who visited military hospitals

• **Conclusion:** the factor with the largest impact on reducing risks during pregnancy (statistically significant) is a pregnant woman being single

• a woman that is single \( \rightarrow \) the smallest risk

• What is wrong?
Data

Example 2: Stock market trading (example by Andrew Lo)

- Data on stock performances of companies traded on stock market over past 25 year
- **Investment goal:** pick a stock to hold long term
- **Proposed strategy:** invest in a company stock with an IPO corresponding to a Carmichael number
- **Evaluation result:** excellent return over 25 years
- Where the magic comes from?

Design cycle

Data

÷

Feature selection

Require prior knowledge

Model selection

Learning

Evaluation

CS 2750 Machine Learning
Feature selection

- **The size (dimensionality) of a sample** can be enormous
  \[ x_i = (x^1_i, x^2_i, \ldots, x^d_i) \quad d \quad \text{- very large} \]

- **Example: document classification**
  - 10,000 different words
  - Inputs: counts of occurrences of different words
  - Too many parameters to learn (not enough samples to justify the estimates the parameters of the model)

- **Dimensionality reduction: replace inputs with features**
  - **Extract relevant inputs** (e.g. mutual information measure)
  - **PCA** – principal component analysis
  - **Group (cluster) similar words** (uses a similarity measure)
    - Replace with the group label

---

Design cycle

- Data
- Feature selection
- Model selection
- Learning
- Evaluation

Require prior knowledge

---

CS 2750 Machine Learning
Model selection

- **What is the right model to learn?**
  - A prior knowledge helps a lot, but still a lot of guessing
  - Initial data analysis and visualization
    - We can make a good guess about the form of the distribution, shape of the function
  -Independences and correlations
- **Overfitting problem**
  - Take into account the **bias and variance** of error estimates
  - Simpler (more biased) model – parameters can be estimated more reliably (smaller variance of estimates)
  - Complex model with many parameters – parameter estimates are less reliable (large variance of the estimate)

---

Solutions for overfitting

How to make the learner avoid the overfit?

- **Assure sufficient number of samples** in the training set
  - May not be possible (small number of examples)
- **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 (random resampling validation techniques)
- **Regularization (Occam’s Razor)**
  - Penalize for the model complexity (number of parameters)
  - Explicit preference towards simple models
### Design cycle

1. **Data**
2. **Feature selection**
3. **Model selection**
4. **Learning**
5. **Evaluation**

- **Require prior knowledge**

### Learning

- **Learning = optimization problem.** Various criteria:
  - **Mean square error**
    \[ w^* = \arg \min_w Error(w) \quad Error(w) = \frac{1}{N} \sum_{i=1...N} (y_i - f(x_i, w))^2 \]
  - **Maximum likelihood (ML) criterion**
    \[ \Theta^* = \arg \max_\Theta P(D \mid \Theta) \quad Error(\Theta) = -\log P(D \mid \Theta) \]
  - **Maximum posterior probability (MAP)**
    \[ \Theta^* = \arg \max_\Theta P(\Theta \mid D) \quad P(\Theta \mid D) = \frac{P(D \mid \Theta)P(\Theta)}{P(D)} \]
Learning

Learning = optimization problem

- Optimization problems can be hard to solve. Right choice of a model and an error function makes a difference.

- Parameter optimizations (continuous space)
  - Linear programming, Convex programming
  - Gradient methods: grad. descent, Conjugate gradient
  - Newton-Rhapson (2nd order method)
  - Levenberg-Marquard

Some can be carried on-line on a sample by sample basis

- Combinatorial optimizations (over discrete spaces):
  - Hill-climbing
  - Simulated-annealing
  - Genetic algorithms

Parametric optimizations

- Sometimes can be solved directly but this depends on the error function and the model
  - Example: squared error criterion for linear regression

- Very often the error function to be optimized is not that nice.

  \[ Error(w) = f(w) \quad w = (w_0, w_1, w_2 \ldots w_k) \]
  - a complex function of weights (parameters)

  \[ \text{Goal: } \quad w^* = \arg \min_w f(w) \]

- One solution: iterative optimization methods

- Example: Gradient-descent method

  Idea: move the weights (free parameters) gradually in the error decreasing direction
Gradient descent method

- Descend to the minimum of the function using the gradient information

\[
\frac{\partial}{\partial w} \text{Error} (w) \big|_{w^*}
\]

- Change the parameter value of w according to the gradient

\[
w \leftarrow w^* - \alpha \frac{\partial}{\partial w} \text{Error} (w) \big|_{w^*}
\]

\[\alpha > 0\] - a learning rate (scales the gradient changes)

CS 2750 Machine Learning
Gradient descent method

- To get to the function minimum repeat (iterate) the gradient based update few times

\[ \text{Error}(w) \]

- **Problems:** local optima, saddle points, slow convergence
- More complex optimization techniques use additional information (e.g. second derivatives)

On-line learning (optimization)

- Error function looks at all data points at the same time
  
  \[ \text{Error}(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 \]
  
  - **On-line error** - separates the contribution from a data point
    
    \[ \text{Error}_{\text{ON-LINE}}(w) = (y_i - f(x_i, w))^2 \]
  
  - **Example:** On-line gradient descent

\[ \text{Error}(w) \]

- **Advantages:**
  1. simple learning algorithm
  2. no need to store data (on-line data streams)
Design cycle

- Data
- Feature selection
- Model selection
- Learning
- Evaluation

Require prior knowledge

Evaluation of learning models

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

- Typically 2/3 training and 1/3 testing
Evaluation

- **Other more complex methods (multiple train/test sets)**
  - Based on random re-sampling validation schemes
    - Cross-validation
    - Random subsampling
    - Bootstrap

Evaluation

- **Random subsampling**
  - Repeat a simple holdout method k times

```
Data
Split randomly into 70% Train, 30% Test
Train
Learning
Average Stats
Test
Classify/Evaluate
```

CS 2750 Machine Learning
**Evaluation**

**Cross-validation (k-fold)**
- Divide data into \( k \) disjoint groups, test on \( k \)-th group/train on the rest
- Typically 10-fold cross-validation
- Leave one out cross-validation
  \( (k = \text{size of the data } D) \)

**Bootstrap**
- The training set of size \( N = \text{size of the data } D \)
- Sampling with the replacement
Evaluation

- What if we want to compare the predictive performance on a classification or a regression problem for two different learning methods?
- **Solution:** compare the error results on the test data set or the average statistics on the same training/testing data splits
- **Answer:** the method with better (smaller) testing error gives a better generalization error.
- But we need to use statistics to validate the choice