CS 2750 Machine Learning

Lecture 2

Machine Learning

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

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

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.**

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

A learning system: basic cycle

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

2. Model selectio

– **Selec**

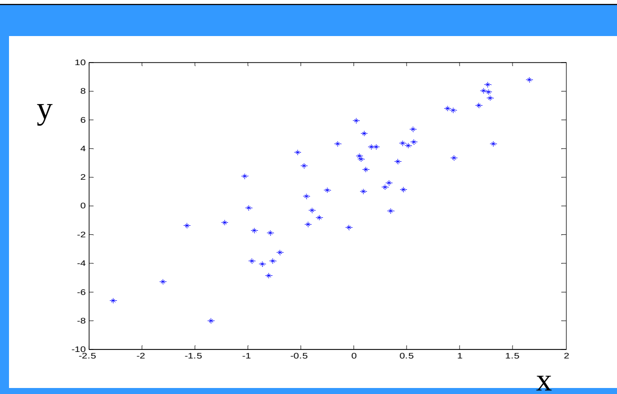
E.g.

3. Choose the

– **Squared e**

4. Learning:

- **Find the set o**
  – The model



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. $y = ax + b$

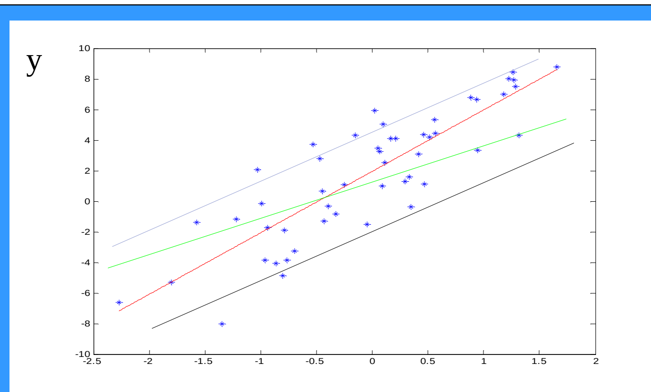
3. Choose the

– **Squa**

4. Learning:

• **Find the s**

– The mo



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. $y = ax + b$

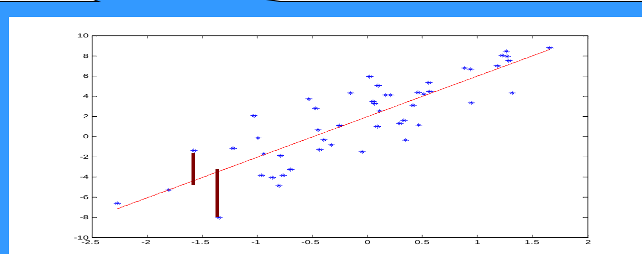
3. Choose the objective function

– **Squared error** $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$

4. Learning:

• **Find the se**

– The mo



A learning system: basic cycle

1. Data: $D =$

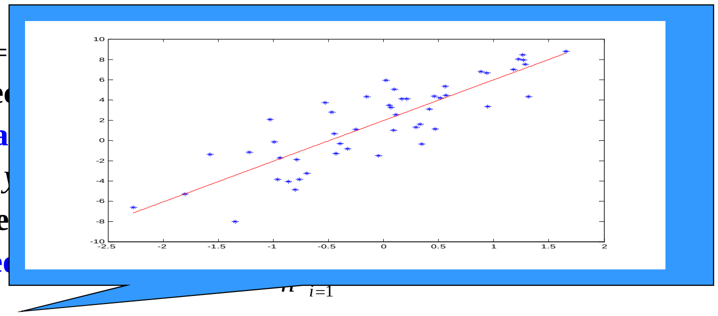
2. Model sele

- **Select a**

  E.g. y

3. Choose the

- **Square**



4. Learning:

- **Find the set of parameters optimizing the error function**
  - The model and parameters with the smallest error

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

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:

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

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

$$E_{(x,y)}[(y - f(x))^2] \qquad \text{Mean squared error}$$

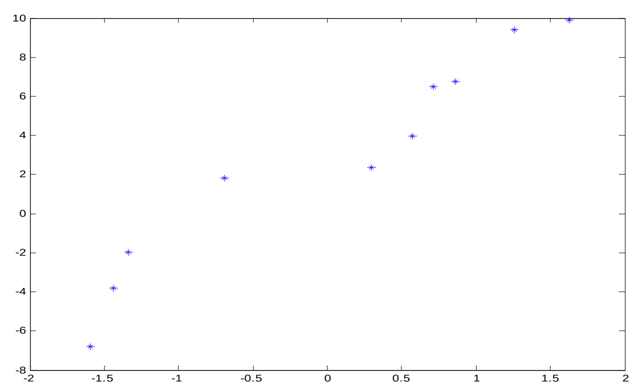
Training error tries to approximate the true error !!!!

Does a good training error imply a good generalization error ?

CS 2750 Machine Learning

Overfitting

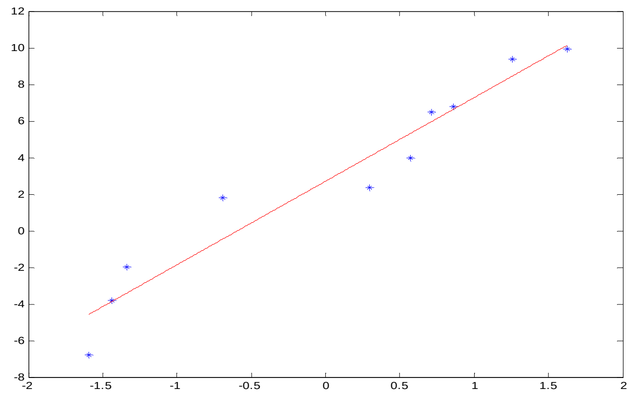
- Assume we have a set of 10 points and we consider polynomial functions as our possible models



CS 2750 Machine Learning

Overfitting

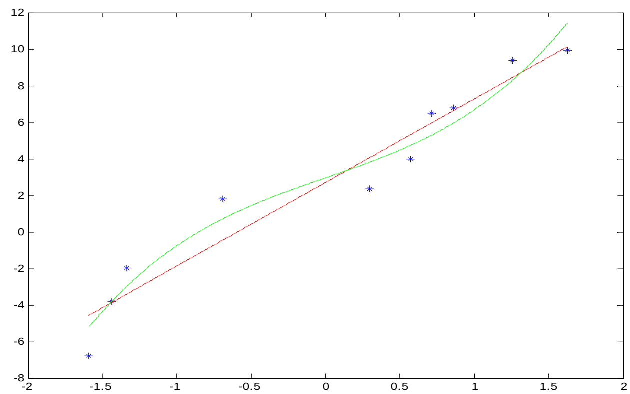
- Fitting a linear function with the square error
- Error is nonzero



CS 2750 Machine Learning

Overfitting

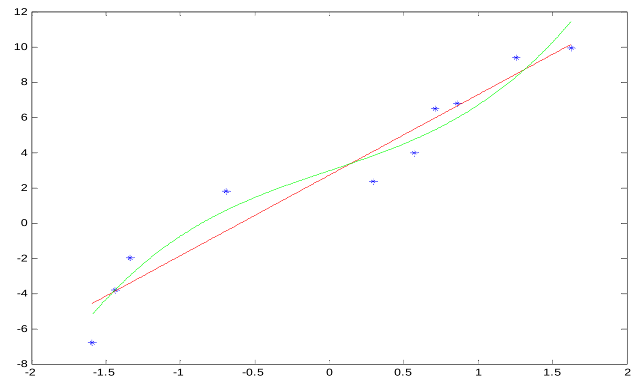
- Linear vs. cubic polynomial
- Higher order polynomial leads to a better fit, smaller error



CS 2750 Machine Learning

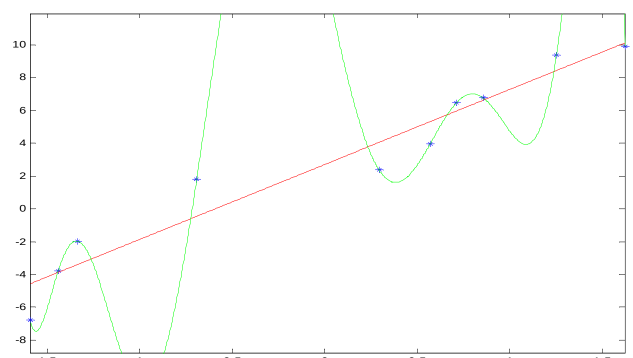
Overfitting

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



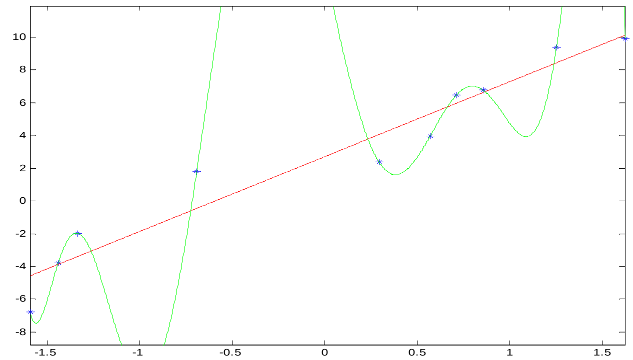
Overfitting

- 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

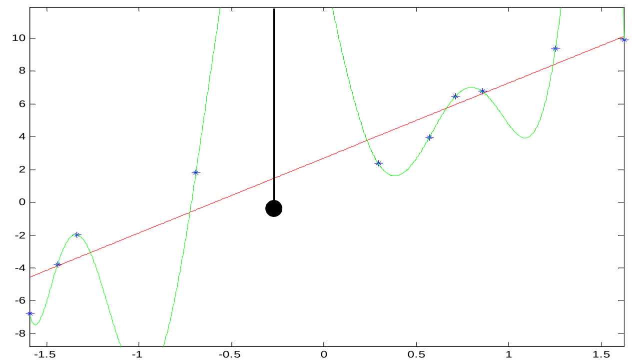
- For 10 data points, degree 9 polynomial gives a perfect fit (Lagrange interpolation). Error is zero.
- Is it always good to minimize the training error? NO !!
- **More important:** How do we perform on the unseen data?



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

$$\frac{1}{n}\sum_{i=1,..n}(y_i - f(x_i))^2$$

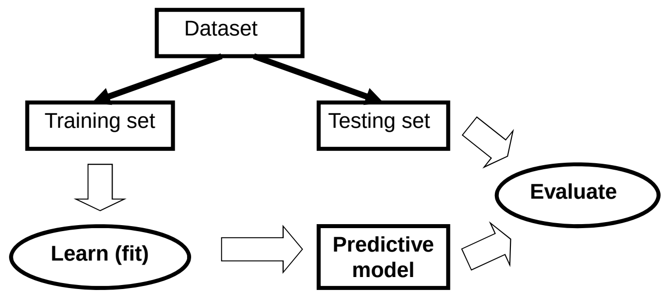
- So how to test the generalization error?

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 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



  – Typically 2/3 training and 1/3 testing

Basic experimental setup to test the learner's performance

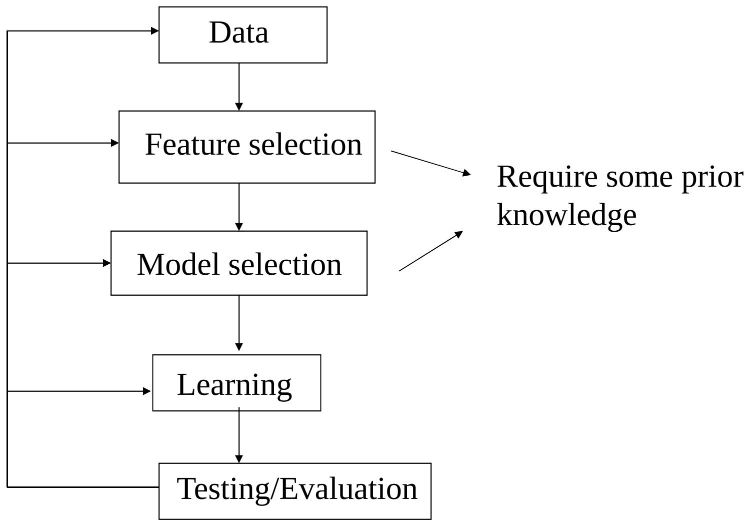
1. Take a dataset D and divide it into:
- **Training data set**
- **Testing data set**

2. Use the training set and your favorite ML algorithm to train the learner

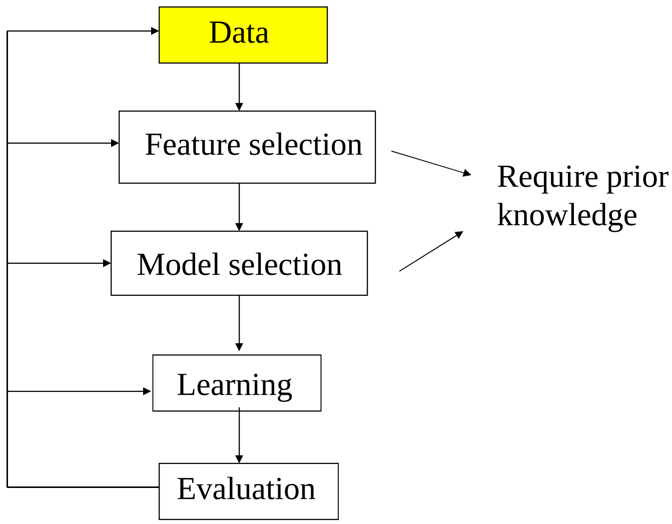
3. Test (evaluate) the learner on the testing data set

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

Design cycle



Design cycle



Data

Data may need a lot of:

- **Cleaning**
- **Preprocessing (conversions)**

Cleaning:

- **Get rid of errors, noise,**
- **Removal of redundancies**

Preprocessing:

- **Renaming**
- **Rescaling (normalization)**
- **Discretization**
- **Abstraction**
- **Aggregation**
- **New attributes**

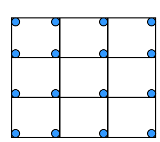
Data preprocessing

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

High → 2
Normal → 1
Low → 0

True → 2
False → 1
Unknown → 0

- **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

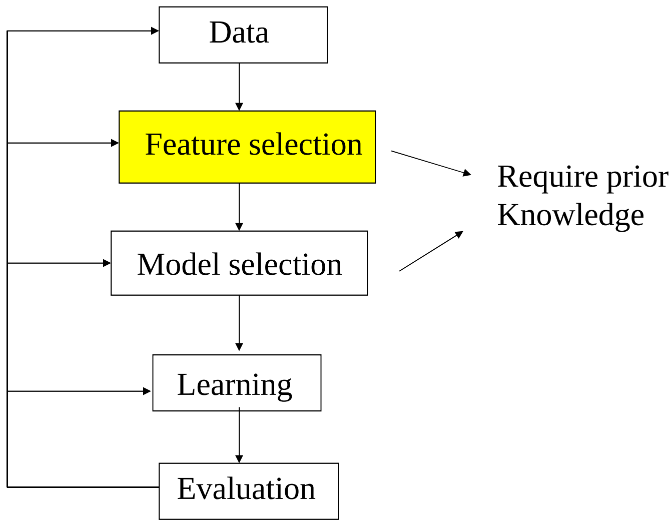
- Study of a large sample of pregnant woman who visited military hospitals
- **Conclusion:** the factor with the largest impact on reducing risks during pregnancy (statistically significant) is a pregnant woman being single
- a woman that is single → the smallest risk
- What is wrong?

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



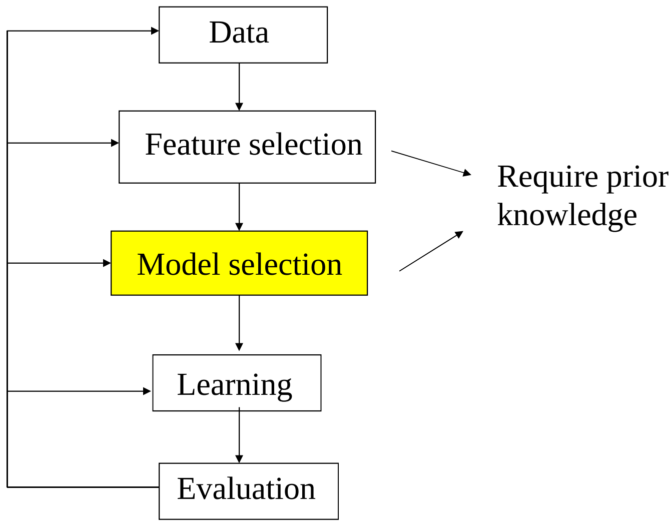
Feature selection

- **The size (dimensionality) of a sample** can be enormous

  $x_i = (x_i^1, x_i^2, .., x_i^d)$ $\quad d$ - very large
- **Example:** **document classification**
  - **thousands of documents**
  - 10,000 different words
  - **Features/Inputs:** counts of occurrences of different words
  - Overfit threat - too many parameters to learn, not enough samples to justify the estimates the parameters of the model

- **Feature selection: reduces the feature sets**
  - **Methods for removing input features**

Design cycle



Model selection

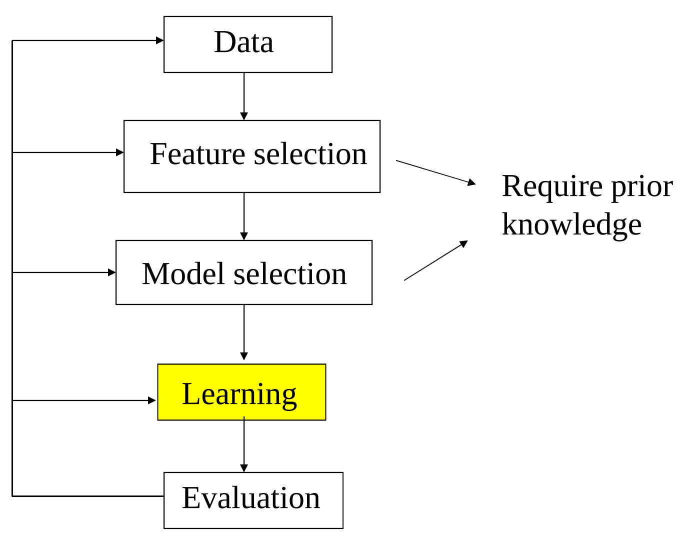
- **What is the right model to learn?**
  - A prior knowledge helps a lot, but still a lot of guessing
  - Initial data analysis and visualization
    - We can make a good guess about the form of the distribution, shape of the function
  - Independences and correlations
- **Overfitting problem**
  - 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 re-sampling validation techniques)
- **Regularization (Occam's Razor)**
  - Explicit preference towards simple models
  - Penalize for the model complexity (number of parameters) in the objective function

Design cycle



Learning

- **Learning = optimization problem**. Various criteria:
  - **Mean square error**

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} Error(\mathbf{w}) \qquad Error(\mathbf{w}) = \frac{1}{N}\sum_{i=1,..N}(y_i - f(x_i,\mathbf{w}))^2$$

  - **Maximum likelihood (ML) criterion**

$$\Theta^* = \arg\max_{\Theta} P(D \mid \Theta) \qquad Error(\Theta) = -\log P(D \mid \Theta)$$

  - **Maximum posterior probability (MAP)**

$$\Theta^* = \arg\max_{\Theta} P(\Theta \mid D) \qquad P(\Theta \mid D) = \frac{P(D \mid \Theta) P(\Theta)}{P(D)}$$

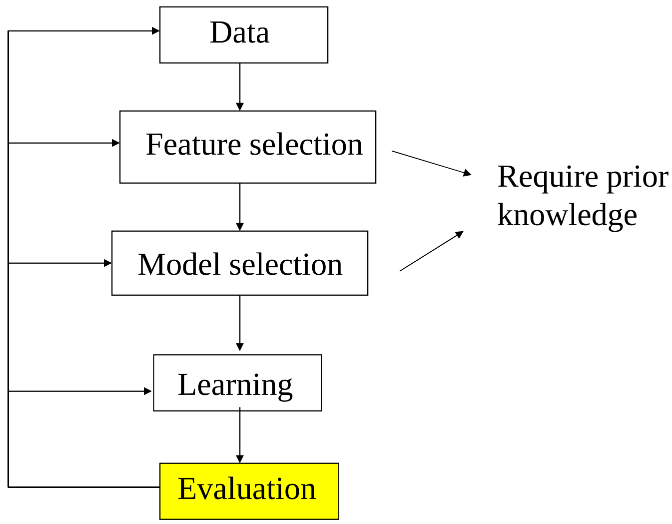
Learning

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

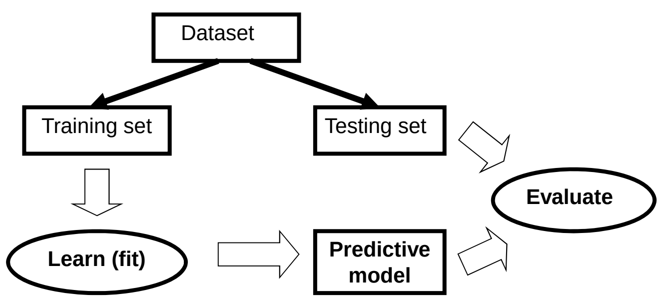
  Some can be carried **on-line** on a sample by sample basis
- **Combinatorial optimizations (over discrete spaces):**
  - Hill-climbing
  - Simulated-annealing
  - Genetic algorithms

Design cycle



Evaluation of learning models

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

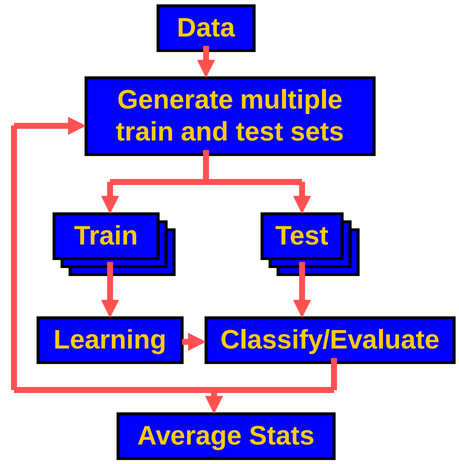


  - Typically 2/3 training and 1/3 testing

Evaluation

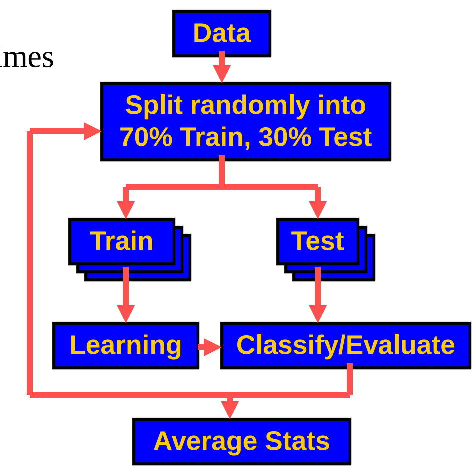
Other more complex methods

- Use multiple train/test sets
- Based on various random re-sampling schemes:
  - Random sub-sampling
  - Cross-validation
  - Bootstrap



Evaluation

- **Random sub-sampling**
  - Repeat a simple holdout method k times

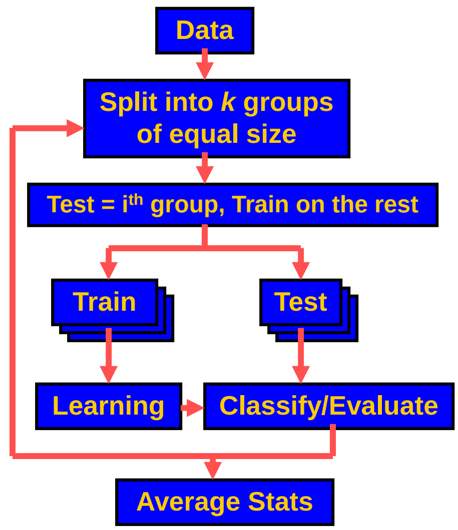


Evaluation

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 = size of the data D)



Evaluation

Bootstrap

- The training set of size N = size of the data D
- Sampling with the replacement