CS 2750 Machine Learning

Lecture 4

Designing a learning system

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

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

Homework assignment

Homework assignment 1 out today

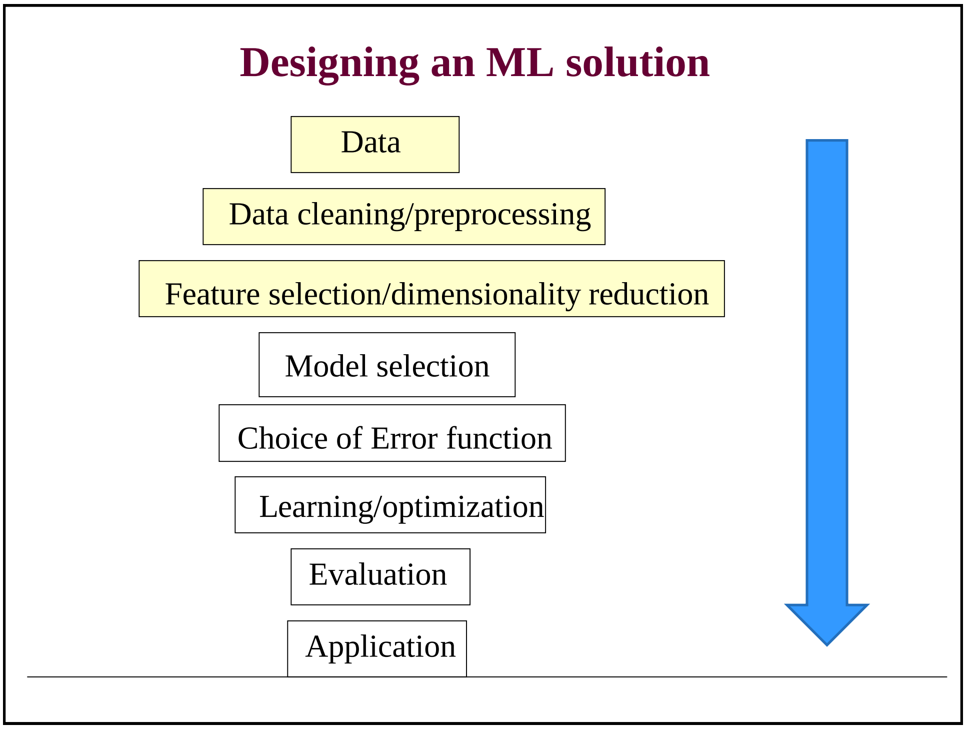
- **Due next week on Thursday**
- Two parts: **Report + Programs**

Submission:

- via Courseweb
- Report (submit in pdf)
- Programs (submit using the zip or tar archive)
- Deadline 1:00pm (prior to the lecture)

Rules:

- Strict deadline
- No collaboration on the programming and the report part



Designing an ML solution

Data

Data cleaning/preprocessing

Feature selection/dimensionality reduction

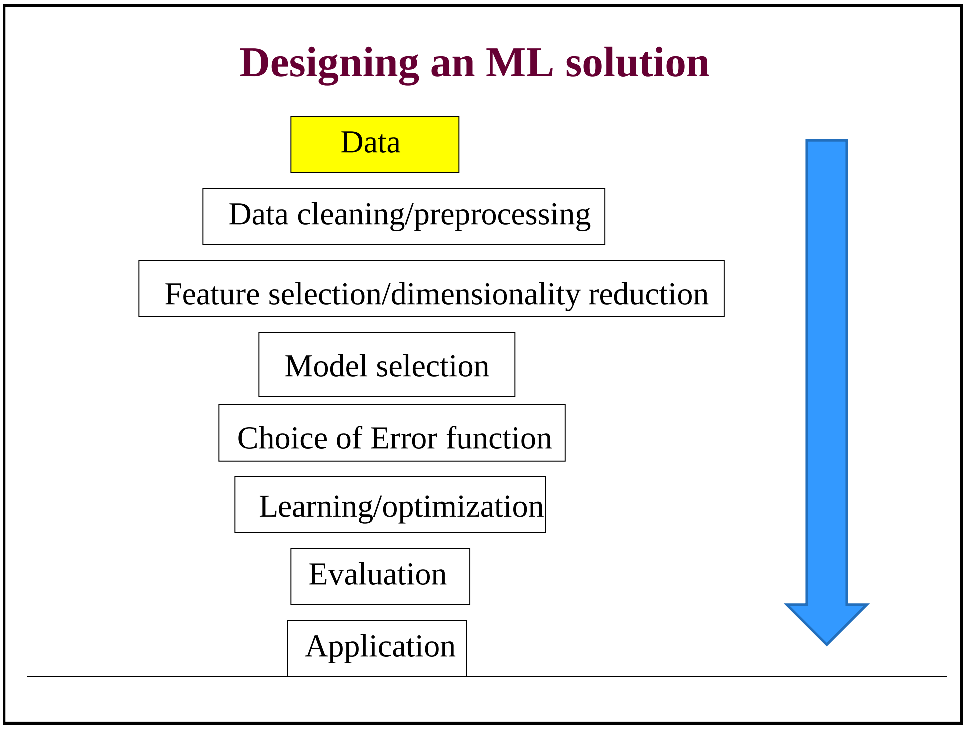
Model selection

Choice of Error function

Learning/optimization

Evaluation

Application



Designing an ML solution

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

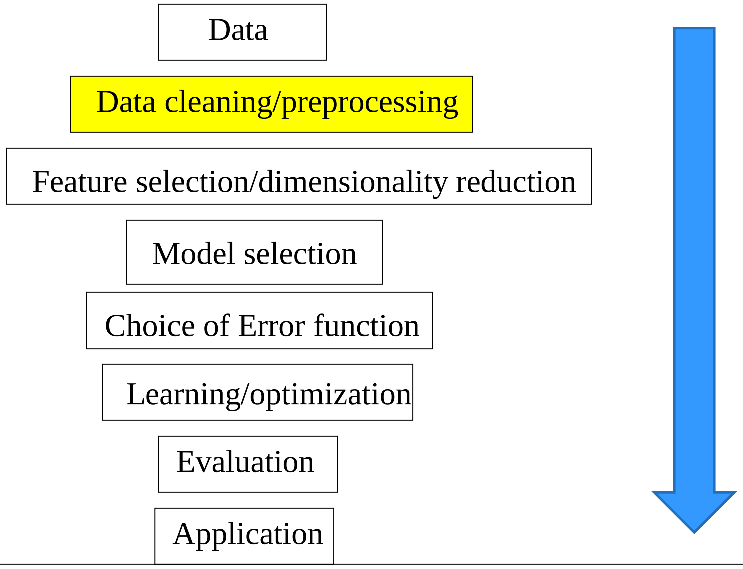
Data extraction:

- 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:

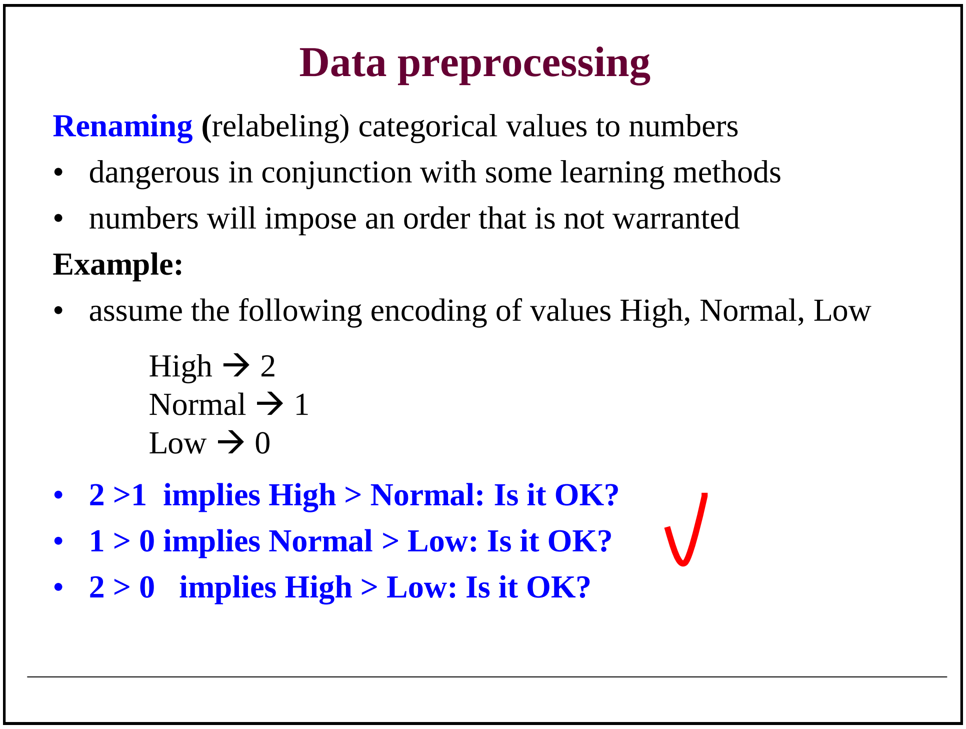
- **Cleaning**
- **Preprocessing (conversions)**

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

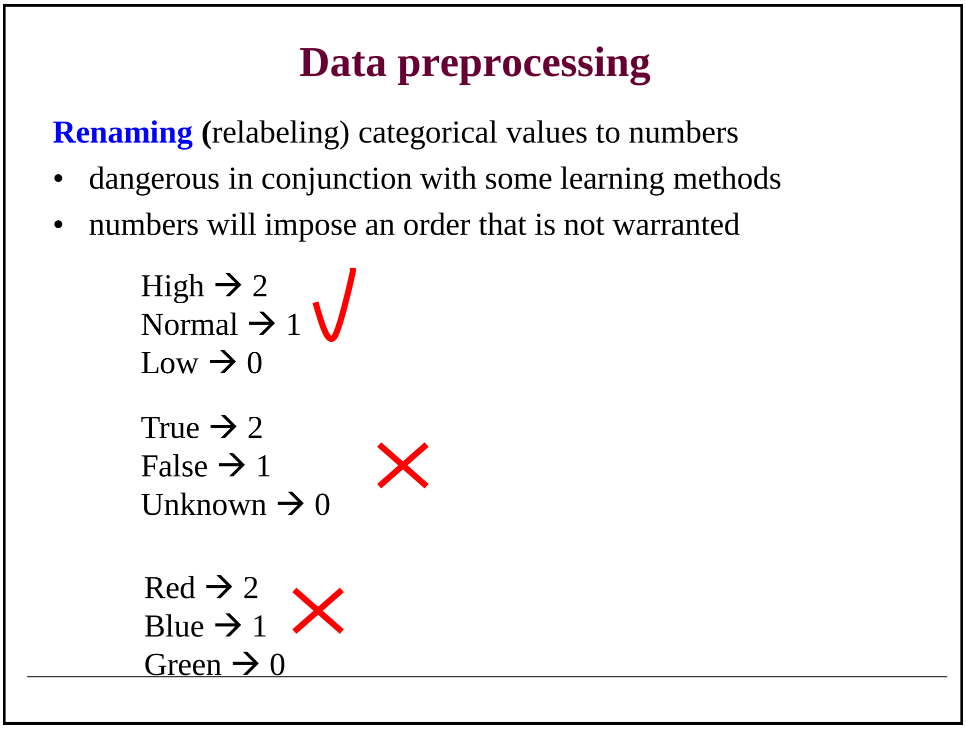
- dangerous in conjunction with some learning methods
- numbers will impose an order that is not warranted

Example:

- assume the following encoding of values High, Normal, Low

High → 2
Normal → 1
Low → 0

- **2 >1 implies High > Normal: Is it OK?**
- **1 > 0 implies Normal > Low: Is it OK?** ✓
- **2 > 0 implies High > Low: Is it OK?**



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

Red → 2
Blue → 1 ✗
Green → 0

Data preprocessing

Renaming **(**relabeling) categorical values to numbers

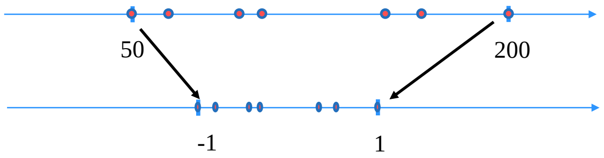
Problem: How to safely represent the different categories as numbers when no order exists?

Solution:

- **Use indicator vector (or one-hot) representation**.
- **Example: Red, Blue, Green colors**
  - **3 categories** → use a vector of size 3 with binary values
  - Encoding:
    - **Red:** (1,0,0);
    - **Blue:** (0,1,0);
    - **Green:** (0,0,1)

Data preprocessing

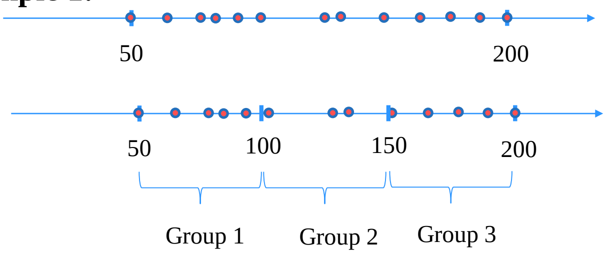
- **Rescaling (normalization):** continuous values are transformed to some range, typically [-1, 1] or [0,1].



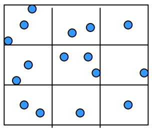
- Why normalization?
  - Some learning algorithms are sensitive to the values recorded in the specific input field and its magnitude

Data preprocessing

- **Discretization (binning):** continuous values to a finite set of discrete values
- **Example 1:**



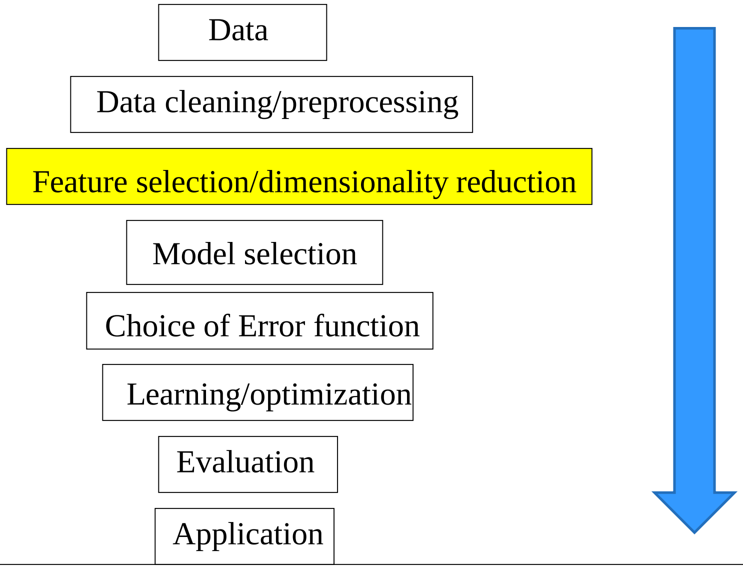
- **Example 2:**



Data preprocessing

- **Abstraction:** merge together categorical values
- **Aggregation:** summary or aggregation operations, such minimum value, maximum value, average over a set of values etc.
- **New attributes:**
  - example: obesity-factor = weight/height

Steps taken when designing an ML system


Data

Data cleaning/preprocessing

Feature selection/dimensionality reduction

Model selection

Choice of Error function

Learning/optimization

Evaluation

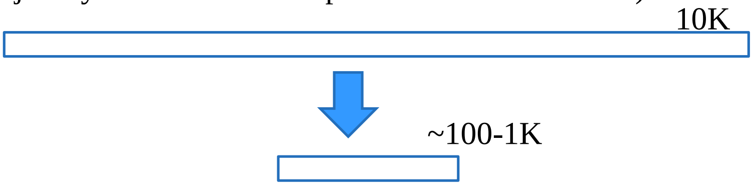
Application

Feature selection/dimensionality reduction

- **The size (dimensionality) of an instance** can be enormous

  $x_i = (x_i^1, x_i^2, .., x_i^d)$ d - very large

- **Problem:** Too many parameters to learn (not enough samples to justify the estimates the parameters of the model)

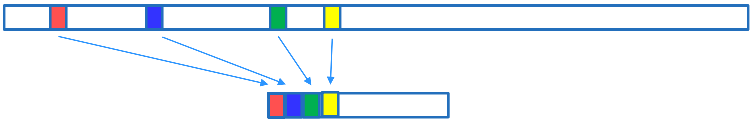


Example: document classification

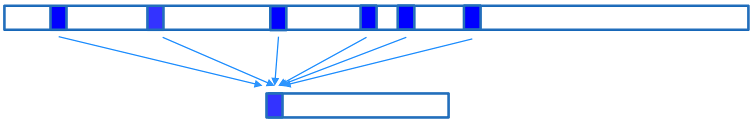
- 10,000 different words
- Big vector: counts of occurrences of different words

Feature selection/dimensionality reduction

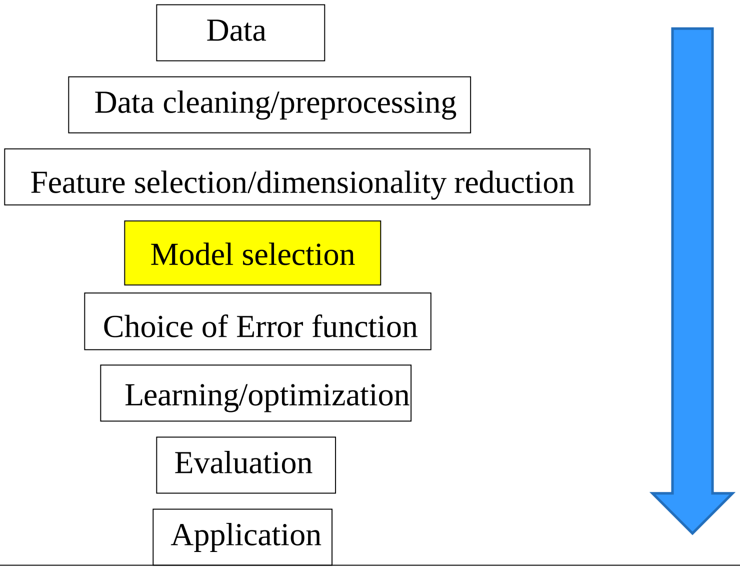
- **Dimensionality reduction solutions**
  - **Extract a small subset of original inputs**



  - **Project inputs into a lower dimensional vector:**
    - **PCA** – principal component analysis
    - **Latent variable models**
    - **Auto-encoders**



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

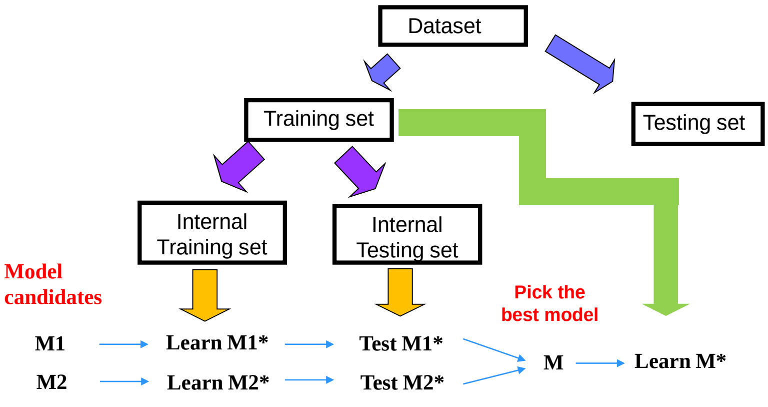
Model selection

- **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 by looking at data
  - 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

A. **Use internal train and test splitting.** Basically, hold some data out of the training set (called validation set) to decide on the model first, than train the picked model



Solutions for overfitting

B. Regularization

- Directly penalizes the model complexity (number of parameters) in the objective function used for finding the model parameters
- **Minimize:**
  - **Error + Penalty for model complexity**
- **Example: assume** $f(x) = w_1 x + w_0$

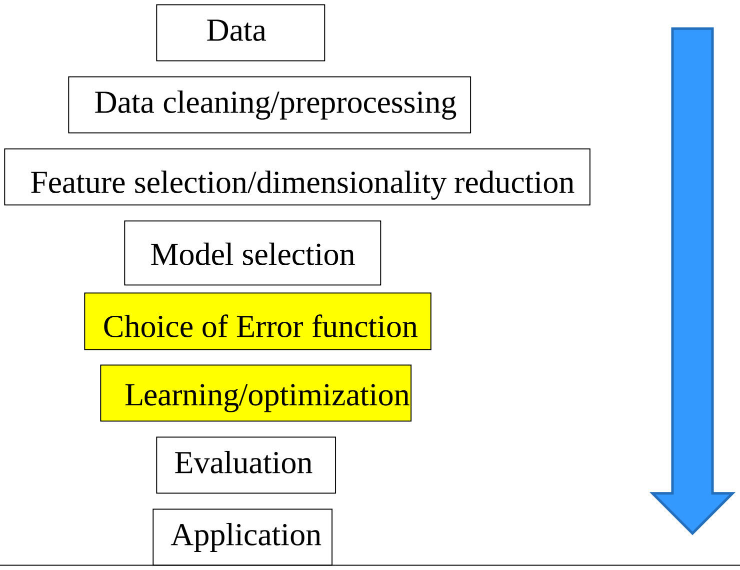
$$\frac{1}{n}\sum_{i=1}^{n}(y_i - w_1 x_i - w_0))^2 + ||[w_1, w_2]||_2^2$$

Error **+** **Penalty**

Regularization penalty reflects our preference towards simple models (Occam's Razor principle)

- – Lasso or Ridge regularization penalties

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

Learning: objective functions

- **Learning = optimization problem**.
  - Various error (loss) functions
  - Also various regularization penalties
- Assume: model $f(\mathbf{w})$ with parameters w and n data instances

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} Error(\mathbf{w}, D)$$

- Examples of loss functions
  - **Mean squared error**

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

  - **Negative loglikelihood**

$$Error(\mathbf{w}, D) = -\sum_{i=1}^{n} \log P(y_i \mid x_i, \mathbf{w})$$

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**
  - Gradient descent, Conjugate gradient
  - Newton-Rhapson
  - 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 objective function and the model
  - **Example:** squared error criterion for the linear regression
- Very often the objective function to be optimized is not that nice.

  $Error(\mathbf{w}, D) = G(\mathbf{w}) \qquad \mathbf{w} = (w_0, w_1, w_2 \dots w_k)$

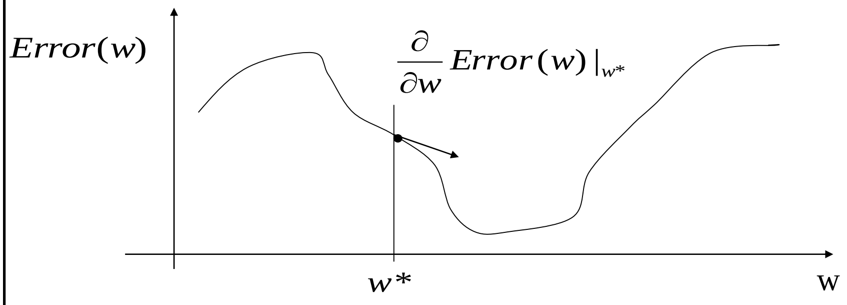
  - a complex function of weights (parameters)

  Goal: $\mathbf{w}^* = \arg\min_{\mathbf{w}} G(\mathbf{w})$
- One solution: **iterative optimization methods**
- **Example: Gradient-descent method**

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

Gradient descent method

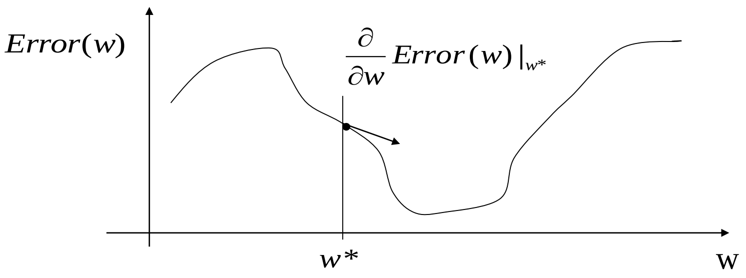
- Descend to the minimum of the function using the gradient information



- Change the parameter value of w according to the gradient

$$w \leftarrow w^* - \alpha \frac{\partial}{\partial w} Error(w)|_{w^*}$$

Gradient descent method



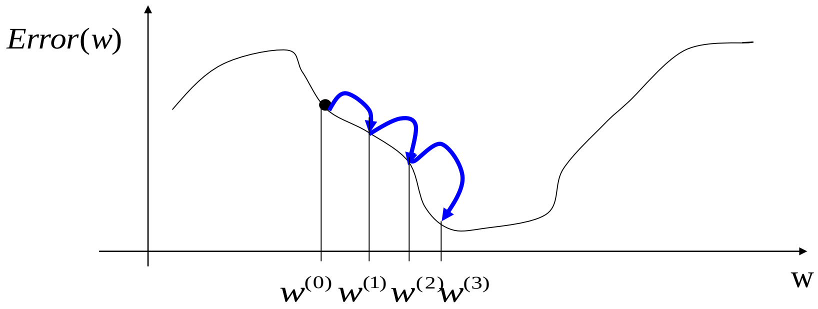
- New value of the parameter

$$w \leftarrow w^* - \boxed{\alpha} \frac{\partial}{\partial w} Error(w)|_{w^*}$$

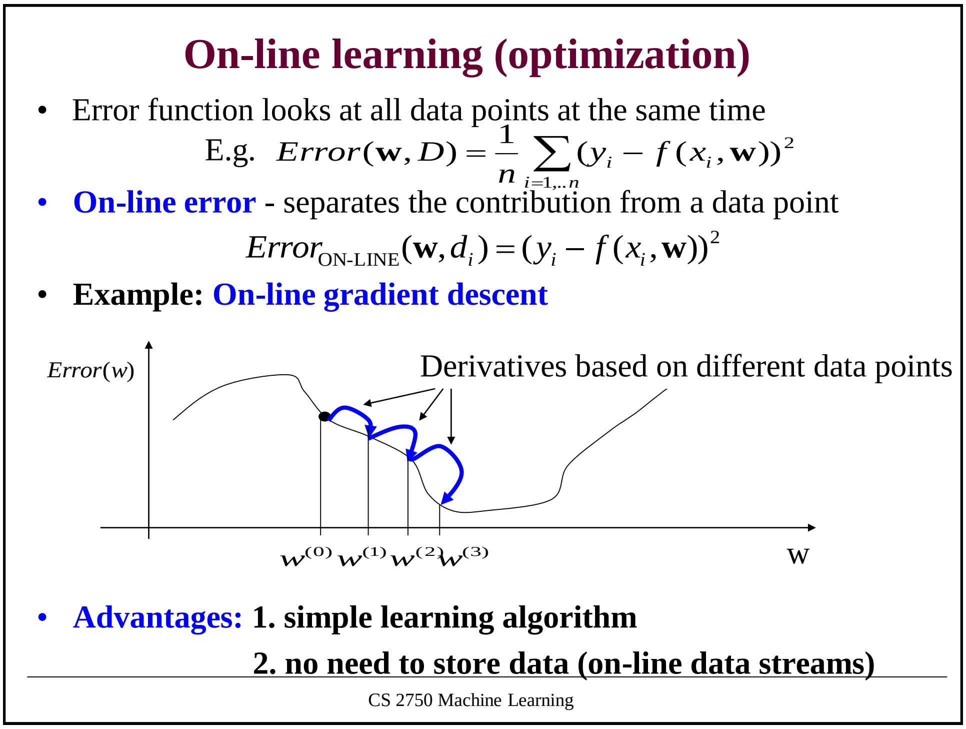
$\alpha > 0$ - a learning rate (scales the gradient changes)

Gradient descent method

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



- **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

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

- **On-line error** - separates the contribution from a data point

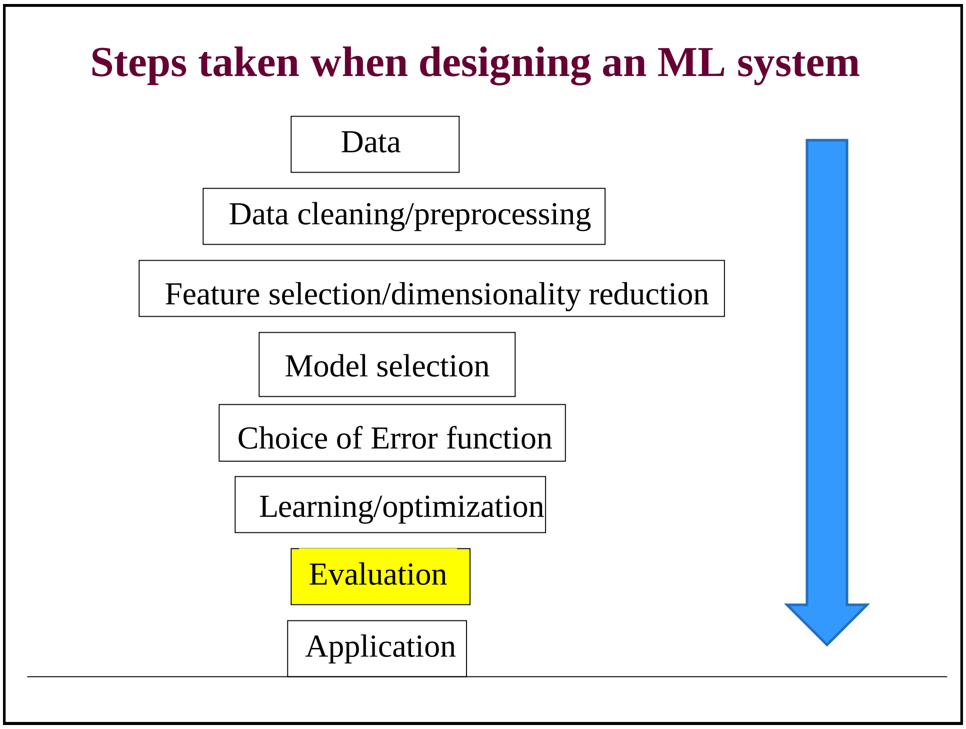
  $$Error_{\text{ON-LINE}}(\mathbf{w}, d_i) = (y_i - f(x_i, \mathbf{w}))^2$$

- **Example: On-line gradient descent**

$Error(w)$
Derivatives based on different data points
$w^{(0)}$
$w^{(1)}$
$w^{(2)}$
$w^{(3)}$
w

- **Advantages: 1. simple learning algorithm**

  2. no need to store data (on-line data streams)



Steps taken when designing an ML system

Data

Data cleaning/preprocessing

Feature selection/dimensionality reduction

Model selection

Choice of Error function

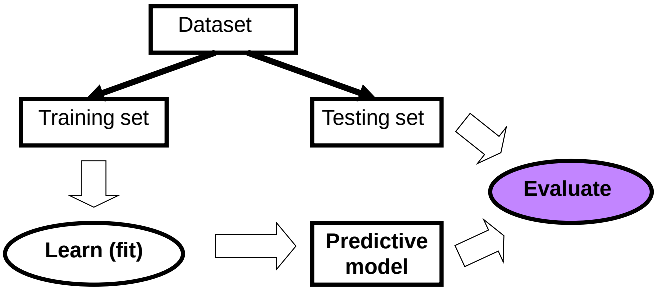
Learning/optimization

Evaluation

Application

Evaluation of models

- **Simple holdout method**



Dataset

Training set

Testing set

Evaluate

Learn (fit)

Predictive model

Evaluation measures

Regression model f: X → Y where Y is real valued

- Mean Squared Error

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

- Mean Absolute Error

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

- Mean Absolute Percentage Error

$$MAPE(D, f) = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - f(x_i)}{y_i}\right|$$

Evaluation measures

Regression model f: X → Y where Y is real valued

- The error is calculated on the test data D of size *n*:

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

- This is an estimate of the error for f on the complete population

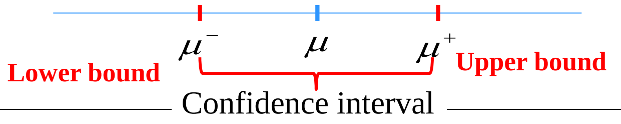
Important question:

- How close is our estimate on the test data to the true mean error?

To answer the question we need to resort to statistics:

- How confident we are the true error falls into interval around our estimate μ ?

Answer: with probability 0.95 the true error is in interval $[\mu^-, \mu^+]$



Lower bound μ^- μ μ^+ **Upper bound**

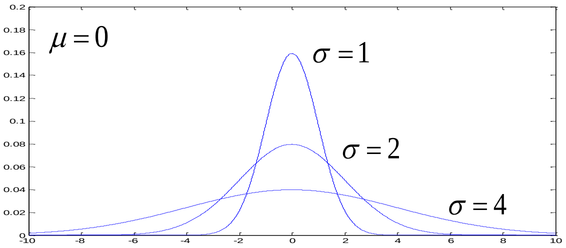
Confidence interval

Evaluation

- **Central limit theorem:**

Let random variables $X_1, X_2, \cdots X_n$ form a random sample from **a distribution** with mean μ and variance σ. Then if the sample *n* is large, the sum of the samples follows the normal distribution:

$$\sum_{i=1}^{n} X_i \approx N(n\mu, n\sigma^2) \quad \text{or} \quad \frac{1}{n}\sum_{i=1}^{n} X_i \approx N(\mu, \sigma^2/n)$$



Statistical significance test

- **Statistical tests for the mean**
  - **H0 (null hypothesis)** $E[X] = \mu^0$
  - **H1 (alternative hypothesis)** $E[X] \neq \mu^0$
- **Basic idea:**
  we use the sample mean $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$

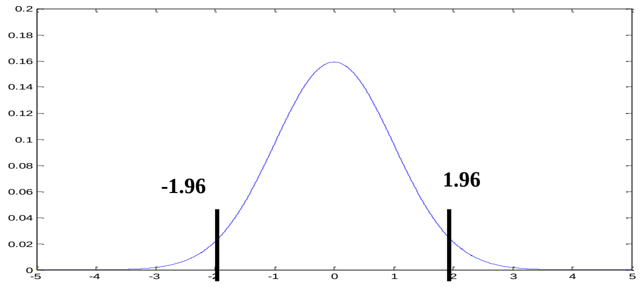
  and check how probable it is that $E[X] = \mu^0$ holds

If the probability that $\overline{X}$ comes from the normal distribution with mean μ^0 is small – we reject the null hypothesis on that probability level

Statistical significance test

- **Statistical tests for the mean**
  - **H0 (null hypothesis)** $E[X] = \mu^0$
  - **H1 (alternative hypothesis)** $E[X] \neq \mu^0$
- **Assume we know the standard deviation σ for the sample**

$$z = \frac{\overline{X} - \mu^0}{\sigma}\sqrt{n} \approx N(0,1) \quad \text{with} \quad P{=}0.95 \quad z \in [-1.96, 1.96]$$



Statistical significance test

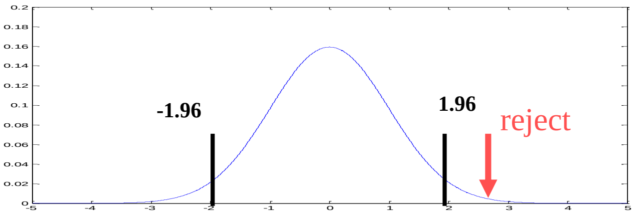
- **Statistical tests for the mean**
  - **H0 (null hypothesis)**

$$E[X] = \mu^0$$

- **Assume we know the standard deviation** σ

$$z = \frac{\bar{X} - \mu^0}{\sigma}\sqrt{n} \approx N(0,1) \qquad \text{with} \qquad \text{P=0.95} \qquad z \in [-1.96, 1.96]$$

- **Z-test: If z is outside of the interval – reject the null hypothesis at significance level (1- P)** if P=0.95 it is 0.05



Statistical significance test

- **Statistical tests for the mean**
  - **H0 (null hypothesis)**

$$E[X] = \mu^0$$

- **Problem: we do not know the standard deviation** σ
- **Solution:**

$$t = \frac{\bar{X} - \mu^0}{s}\sqrt{n} \approx \text{t} - \text{distributi on} \qquad \text{(Student distribution)}$$

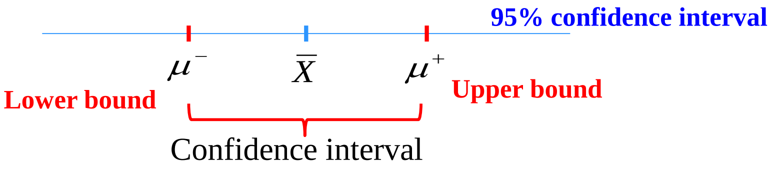
$$s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}}$$

- Estimate of the standard deviation

- **T-test: If t is outside of the tabulated interval reject the null hypothesis at the corresponding significance level**

Confidence interval

- Assume we have calculated the average error $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$
- There are many values of μ^0 around it that are not rejected at some significance level (say 0.05)
- These values form a confidence interval around it



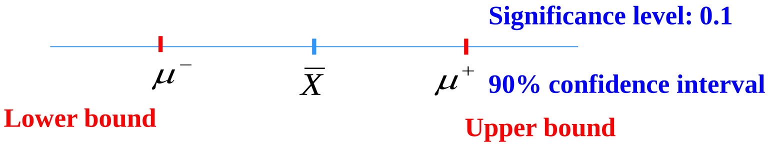
95% confidence interval

μ^- $\overline{X}$ μ^+

Lower bound

Upper bound

Confidence interval



Significance level: 0.1

μ^- $\overline{X}$ μ^+

90% confidence interval

Lower bound

Upper bound

Statistical tests

The statistical tests lets us answer:

- The interval to which the true error falls to with some high probability (confidence) :

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

- Compare two models M1 and M2 and determine based on the error on the data entries the probability with which model M1 is different (or better) than M2

$$MSE(D, f_1) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_1(x_i))^2 \qquad MSE(D, f_2) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_2(x_i))^2$$

Trick:

$$MSE(D, f_1) - MSE(D, f_2) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_1(x_i))^2 - \frac{1}{n}\sum_{i=1}^{n}(y_i - f_2(x_i))^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - f_1(x_i))^2 - (y_i - f_2(x_i))^2$$

Evaluation measures for classification

Assume binary classification:

- **Confusion matrix represents all possible combination of true and predicted values**

		Actual	
		Case	Control
Prediction	Case	TP 0.3	FP 0.1
	Control	FN 0.2	TN 0.4

TP – true positive
FP – false positive
TN – true negative
FN – false negative

Evaluation measures for classification

Evaluation stats calculated from the confusion matrix:

		Actual	
		Case	Control
Prediction	Case	TP 0.3	FP 0.1
	Control	FN 0.2	TN 0.4

TP – true positive
FP – false positive
TN – true negative
FN – false negative

Misclassification error:

$$E = FP + FN$$

Accuracy:

$$Accuracy = TP + TN$$

Sensitivity:

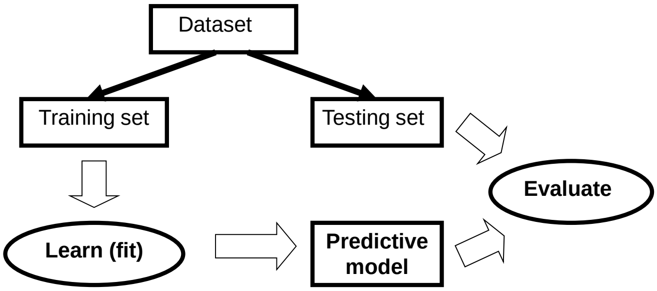
$$SN = \frac{TP}{TP + FN}$$

Specificity:

$$SP = \frac{TN}{TN + FP}$$

Evaluation of models

- **We started with a simple holdout method**



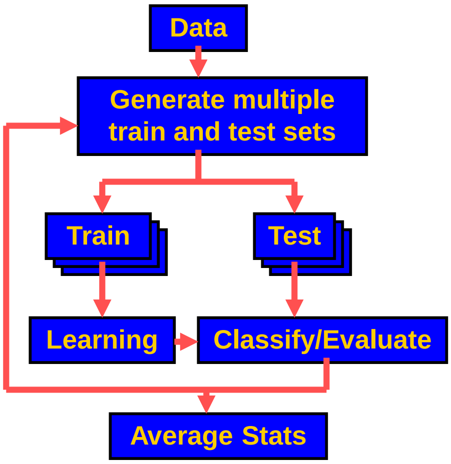
Problem: the mean error results may be influenced by a lucky or an unlucky **training and testing** split especially for a small size D

Solution: try multiple train-test splits and average their results

Evaluation of models via random resampling

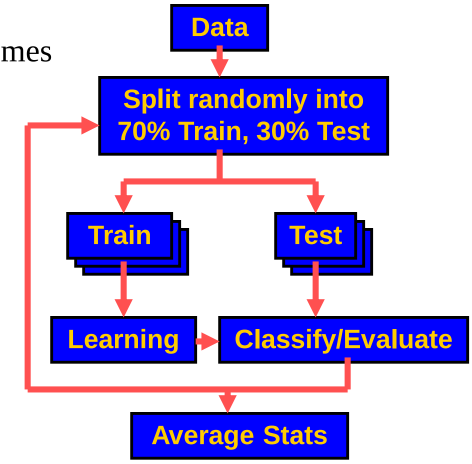
Other more complex methods

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



Evaluation of models using random subsampling

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



Data

Split randomly into 70% Train, 30% Test

Train

Test

Learning

Classify/Evaluate

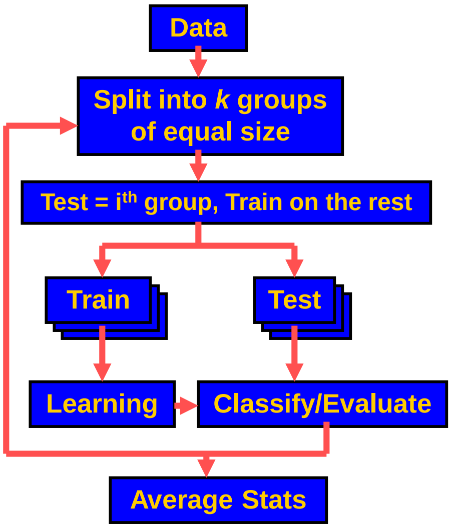
Average Stats

Evaluation of models using k-fold cross-validation

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)



Data

Split into *k* groups of equal size

Test = ith group, Train on the rest

Train

Test

Learning

Classify/Evaluate

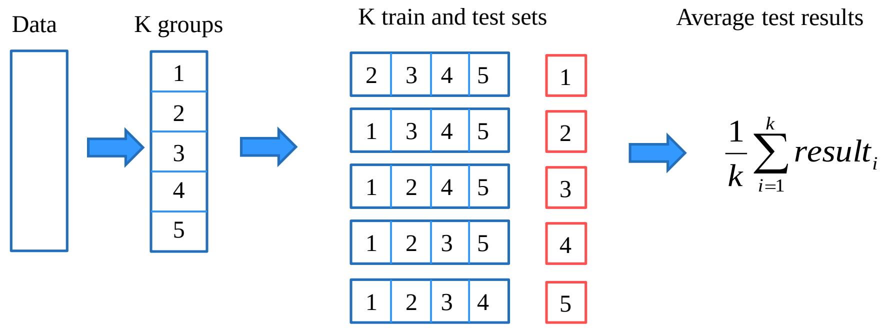
Average Stats

Evaluation of models using k-fold cross-validation

Cross-validation (k-fold)

- Divide data into k disjoint groups,
- For every group i, test on i-th group and train on the rest
- Gives k models and k test results

Example: k=5 (5-fold crossvalidation)



Data

K groups

1
2
3
4
5

K train and test sets

2 3 4 5	1
1 3 4 5	2
1 2 4 5	3
1 2 3 5	4
1 2 3 4	5

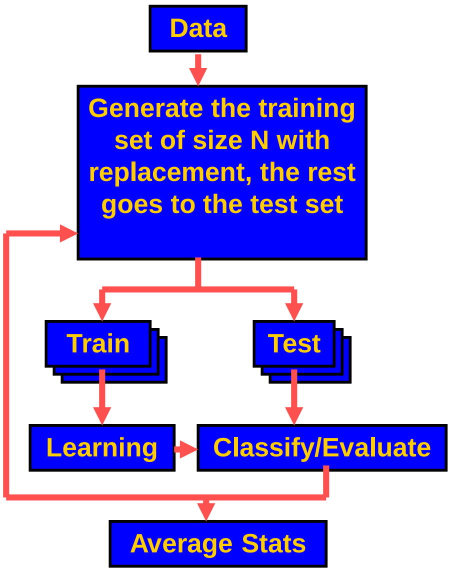
Average test results

$$\frac{1}{k}\sum_{i=1}^{k} result_i$$

Evaluation of models using bootstrap

Bootstrap

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



Data

Generate the training set of size N with replacement, the rest goes to the test set

Train

Test

Learning

Classify/Evaluate

Average Stats