Model Configuration and Experiments

Introduction à l'apprentissage automatique – GIF-4101 / GIF-7005

Professor: Christian Gagné

Week 14

UNIVERSITÉ LAVAL

14.1 Algorithms evaluation and comparison

Algorithms evaluation and comparison

- Performance evaluation problem
  - How to evaluate the performance of a classification algorithm for a given problem (for generalization)?
  - Big difference between the performance on the training and testing dataset
- Performance comparison problem
  - How to evaluate if an algorithm performs better than another for a given problem?
  - Different types of comparisons are possible
    - Different algorithms
    - Same algorithm, different hyperparameters
    - Same algorithm, different data representations
- Repetition of the necessary measures for statistical validity
  - Random partitioning for training/validation
  - Learning process with variable results
    - Stochastic algorithm
    - Algorithm sensitive to the choice of hyperparameters (e.g. σ and C values of the SVM)

Charlatans example (Jensen and Cohen, 2000)

- Evaluation of an investment advisor
  - Each day, the advisor must predict whether stock prices will rise or fall.
  - Test: predict stock prices for 14 days
  - Selection criteria: correct prediction for 11 days or more
    - Charlatan makes random predictions (0.5/0.5)
    - Charlatan therefore has a probability of 0.0287 of passing the test.
  - Good test to evaluate an advisor's performance
- But is not suitable for choosing an advisor from n candidates.
  - Probability that a charlatan among n passes the test: $1 - (1 - 0.0287)^n$.
  - For $n = 10$, probability ≈ 0.253; for $n = 30$, probability ≈ 0.583.
  - For a high value of n, it is almost certain that charlatans will pass the test, even if they do not do better than chance!

D. Jensen, P. Cohen, *Multiple Comparisons in Induction Algorithms*, Machine Learning, n^o 38, p. 309–338, 2000.

Pathologies in learning

- Overfitting
  - Add superfluous elements to the model (learn by heart)
    - Low value of C with SVM, too many support vectors
  - Discovering non-existent relationships between data
    - Overtraining: learning false links between data
  - Making more complex models that offer no advantage
- Errors in the selection of discriminating information
  - Bias in the algorithm favors certain types of data
    - Parametric classification with multivariate normal distribution and diagonal covariance matrix: bias towards discrimination of independent variables
  - Sensitivity to the prior probabilities of the data (classes balance)
  - Sensitivity to feature selection
- Oversearching
  - Searching in very large model spaces
    - Solution: first, simple model spaces, then, increase complexity
  - Similar to increasing the value of n with the example of charlatans
    - Solution: tighten the selection criteria when n increases

Factors to consider (1/2)

- Difficult to generalize any conclusions made on a particular problem to other problems
  - *No Free Lunch* theorem!
  - Good algorithm for a problem: compatibility between the inductive bias and the problem
- Partitioning the dataset into training/validation subsets for testing only
  - Good for evaluation/comparison of performance in algorithm generalization
  - Good for choosing hyperparameters
  - Once the choice of algorithms/hyperparameters is made: use of the entire dataset for training

Factors to consider (2/2)

- Validation subset is part of the inference data
  - Choice of hyperparameter or stopping criteria
    - Each use of the validation set integrates information into the learning algorithm
  - Final performance evaluation on a separate test set, **never** used in the learning phase
- Other criteria for evaluation and comparison of algorithms
  - Other risk measures, other loss functions
  - Complexity of training (time and space)
  - Complexity of the evaluation (time and space)
  - Interpretability of results
  - Ease of programming

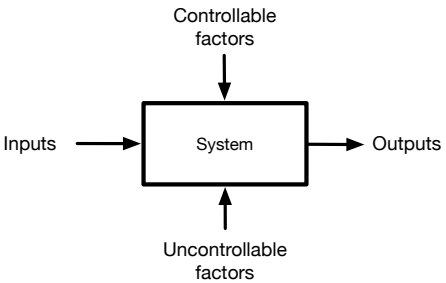
14.2 Design of experiments

Experiments

- Experimentation: test or series of tests where we play with factors modifying the output
  - Choice of the learning algorithm
  - Training dataset
  - Data characteristics
- General objectives
  - Identify the most influential factors
  - Eliminate the least important factors
  - Determine the configuration of the factors giving the best results
- Learning objectives
  - Statistically significant results (eliminate effect of chance)
  - Better performance for generalization
  - Reduced complexity (time and space)
  - Robustness

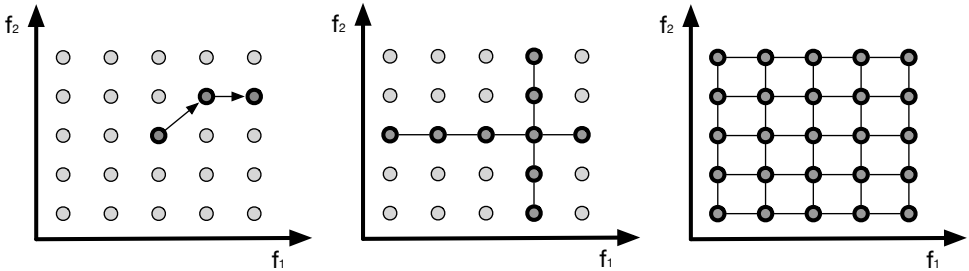
Experimental process

- Controllable factors: elements we want to study
- Uncontrollable factors: elements over which we do not have control, but for which we want to minimize impact on decisions



Experimentation strategies

- Possible experimentation strategies
  - By intuition: experimentation based on the operator's intuition
  - One factor at a time: starting configuration, testing all values of one factor separately
  - Grid search: test all combinations



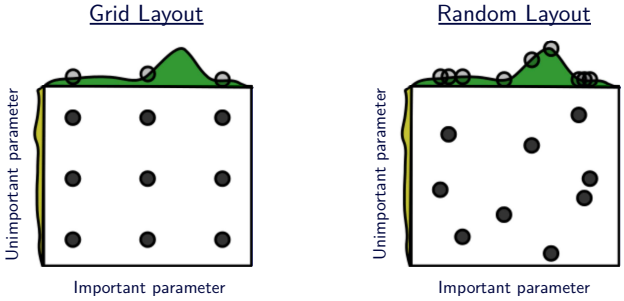
Best guess | One factor at the time | Grid search

Grid search

- Grid search: adjustment of pairs (or triplets) of hyperparameters, with measurement on validation set
  1. Partition the dataset $\mathcal{X}$ into two subsets, $\mathcal{X}_T$ and $\mathcal{X}_V$ (usually 50%-50%)
  2. Train summarily the classifier with $\mathcal{X}_T$ for each pair of hyperparameters considered
  3. Select the pair of hyperparameters for which the error is minimal on $\mathcal{X}_V$
  4. Use this pair of hyperparameters for training on the whole set $\mathcal{X}$
- Applicable for all pairs of hyperparameters for which the combined effect is important for the training of classifiers

Random search

- Select hyperparameter values randomly
  - Allows a better exploration of space in the presence of variables with no influence



From *J. Bergstra and Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research, vol. 13, 2012.*
Available online at `https://www.jmlr.org/papers/v13/bergstra12a.html`.

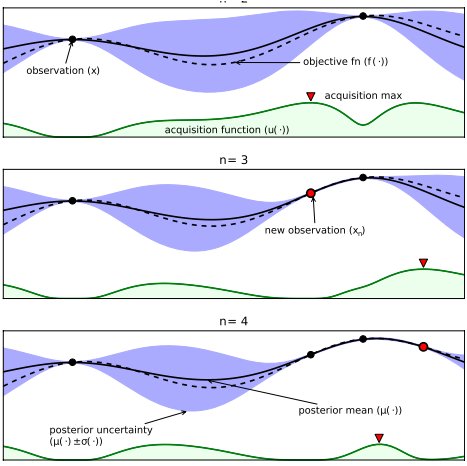
- Possible refinement: use of quasi-random numbers
  - Deterministic sequence with uniformly distributed values for each dimension

14.3 Optimization for hyperparameter adjustment

Model-based sequential optimization

- Idea: Build learning models to estimate performance
  - Regression of a function $\mathrm{f}(\mathbf{x})$ which gives the estimated performance according to hyperparameters $\mathbf{x}$
  - Estimating the uncertainty of predictions in hyperparameter space
  - Commonly used model: Gaussian processes
    - Random process generating a normal distribution for each value of $\mathbf{x}$
- Exploration-exploitation compromise: selection of future hyperparameters $\mathbf{x}$ to be evaluated
  - Exploitation: select value of $\mathbf{x}$ with good performance
  - Exploration: test new value of $\mathbf{x}$ to acquire more information on the function to be optimized
- Acquisition function to determine the next value of $\mathbf{x}$
  - Typical function *Upper Confidence Bound*: $\mathrm{argmax}_{\mathbf{x}}\, \mu(\mathbf{x}) + \sigma(\mathbf{x})$
- Re-estimate regression function with evaluation of the next value

Bayesian optimization



From *B. Shahriari, K. Swersky, Z. Wang, R.P. Adams and N. De Freitas, Taking the human out of the loop: A review of bayesian optimization, Proceedings of the IEEE, vol. 104, no. 1, 2016.* Available online at `https://doi.org/10.1109/JPROC.2015.2494218`.

AutoML

- AutoML: automate machine learning
  - Allow the use of these techniques by non-experts
  - Allow deployment in unknown situations, with minimal intervention
  - Enable adaptation of models to operating conditions
- Choice of models and pre-processing
  - Beyond the choice of hyperparameters, which model to use?
    - SVM, neural networks, k-nearest neighbours, linear models, AdaBoost, random forests, etc.
  - Refine model configuration
    - Number of hidden layers, core function, distance measurement, etc.
  - What pre-processing to do with the data?
    - Normalization, standardization, feature selection, etc.
- Apart from hyperparameter optimization, another research topic
  - No universal models
  - Computing resources required can be very large
  - Dataset size limits the possible scope of model search

14.4 Organization of experimental plans

Basic principles for planning experiments

- Randomize: the order of execution of the experiments must be randomized, in order to ensure independence in the results
  - E.g.: a machine that requires a certain time to be at the right temperature
  - Generally not a problem when experimenting with software
- Reproduce: average the results of several experiments with the same values of controllable factors, to eliminate the effect of uncontrollable factors
  - For learning: run the same algorithm with different samples of the dataset (e.g. cross-validation)
- Block: reduce or avoid nuisance factors, which influence the output results, without being of interest
  - For learning: compare algorithms using the same data samples (same subsets)

Directives for experimentation with learning

1. Setting the study objective
   - Estimate the error of a method on a particular problem (error below a given value)
   - Comparing two algorithms on the same problem (is one algorithm better than the other?)
2. Select the response variable
   - Classification error or quadratic error in regression
   - Arbitrary loss function, risk measurement, accuracy, recall, complexity, etc
3. Choice of factors and levels
   - Hyperparameter values
   - Learning algorithms
   - Datasets
4. Choice of the experimentation plan
   - Make a factorial design, unless you are sure there are no interactions
   - Number of replications for the experiments is inversely proportional to the size of the datasets (variance of results according to size)
   - Avoid synthetic datasets to assess performance

Directives for experimentation with learning

5. Performing the experiments
   - Do some preliminary executions to make sure everything is going as planned
   - For resource-intensive experiments, backup intermediate states (*checkpoints*)
   - Experiments must be reproducible
   - Make honest comparisons, being fair towards the different approaches tested
6. Perform a statistical analysis of the data
   - Ensure that results are not subjective or a product of chance
   - Testing statistical hypotheses: is the error of A significantly lower than B?
7. Conclusions and recommendations
   - Once data has been obtained and analyzed, draw objective conclusions
   - Frequent conclusions: need to do more experiments!
   - Proceed iteratively: don't invest all the resources for making a single set of experiments

14.5 Manipulating datasets

Partitioning and stratification

- Ideal case: partitioning dataset $\mathcal{X}$ into K separate pairs of training and validation datasets
  - Requires huge datasets
- Solution: make several subsets of the same dataset

$$\{\mathcal{T}_i, \mathcal{V}_i\}_{i=1}^{K}$$

  - Trade-off between datasets size and overlap
    - Big datasets allow better inference of classifiers
    - Big overlap between datasets gives non-statistically independent measures
- Partitioning with stratification
  - Respecting the prior probabilities when partitioning into training/validation datasets
  - Avoids variations due to algorithm bias related to proportions between classes

Effect of the training dataset size

- For real problems, it is common that the error rates in training and testing follow power laws

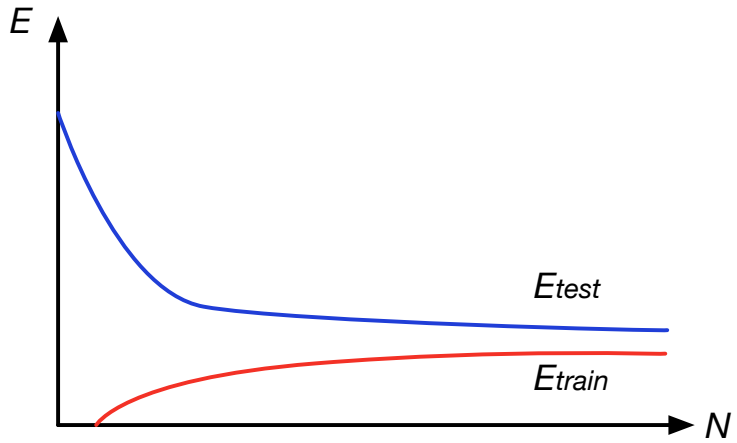
$$\begin{aligned} E_{train} &= E_{Bayes} - \frac{b}{N^{\beta}} \\ E_{test} &= E_{Bayes} + \frac{a}{N^{\alpha}} \end{aligned}$$

  where a, b, $\alpha \geq 1$ and $\beta \geq 1$ depend on the classifier and the problem

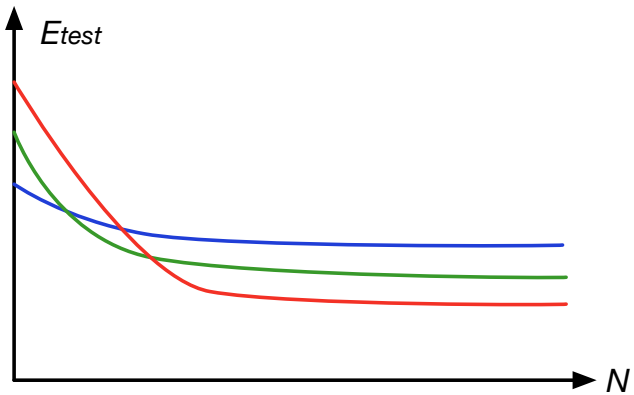
- With large datasets, error rates tend to be towards the optimal Bayesian rate.

$$\begin{aligned} \lim_{N \to \infty} E_{train} &= E_{Bayes} \\ \lim_{N \to \infty} E_{test} &= E_{Bayes} \end{aligned}$$

Training and testing rate as a function of N



Rate under test as a function of N



K-fold cross-validation

- K-fold cross-validation
  - Training dataset divided into K disjointed subsets, $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_K = \mathcal{X}$
  - K training on $\mathcal{T}_i$ and evaluation on $\mathcal{V}_i$, $i = 1, \ldots, K$

$$\begin{array}{ll} \mathcal{V}_1 = \mathcal{X}_1 & \mathcal{T}_1 = \mathcal{X}_2 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_K \\ \mathcal{V}_2 = \mathcal{X}_2 & \mathcal{T}_2 = \mathcal{X}_1 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_K \\ \vdots & \vdots \\ \mathcal{V}_K = \mathcal{X}_K & \mathcal{T}_K = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_{K-1} \end{array}$$

  - Average performance over $\mathcal{V}_i$, $i = 1, \ldots, K$
  - $(K-2)/K$ of data shared by each pair of training sets (statistical non-independence of the results)
- *Leave-one-out*: $K = N$
  - Training on $N-1$ data, performance on one data (repeated N times)
  - Useful for algorithms with reduced or no training times (e.g. k-PPV), or very small datasets

5×2 cross-validation

- 5×2 cross-validation
  - Divide dataset $\mathcal{X}$ into two equal disjoint subsets $\mathcal{X}_1^{(1)}$ et $\mathcal{X}_1^{(2)}$
  - Train on $\mathcal{T}_1 = \mathcal{X}_1^{(1)}$ and evaluate on $\mathcal{V}_1 = \mathcal{X}_1^{(2)}$
  - Repeat with training on $\mathcal{T}_2 = \mathcal{X}_1^{(2)}$ and evaluation on $\mathcal{V}_2 = \mathcal{X}_1^{(1)}$
  - Repeat five times for a total of 10 trainings/evaluations

$$
\begin{array}{ll}
\mathcal{T}_1 = \mathcal{X}_1^{(1)} & \mathcal{V}_1 = \mathcal{X}_1^{(2)} \\
\mathcal{T}_2 = \mathcal{X}_1^{(2)} & \mathcal{V}_2 = \mathcal{X}_1^{(1)} \\
\mathcal{T}_3 = \mathcal{X}_2^{(1)} & \mathcal{V}_3 = \mathcal{X}_2^{(2)} \\
\mathcal{T}_4 = \mathcal{X}_2^{(2)} & \mathcal{V}_4 = \mathcal{X}_2^{(1)} \\
\vdots & \vdots \\
\mathcal{T}_9 = \mathcal{X}_5^{(1)} & \mathcal{V}_9 = \mathcal{X}_5^{(2)} \\
\mathcal{T}_{10} = \mathcal{X}_5^{(2)} & \mathcal{V}_{10} = \mathcal{X}_5^{(1)}
\end{array}
$$

- More than five repetitions: too many dependencies between datasets
- Less than ten results: not enough samples to estimate a distribution and do statistical tests

Bootstrapping

- *Bootstrapping*: sampling **with replacement**
  - Generate training set by sampling N data with replacement among N data of the original set
  - Validation on a different training set, generated in the same way
  - Repeat as many times as necessary to evaluate performance
  - Probability to sample a data is $1/N$
    - For dataset of N data, probability that a given data is not drawn

$$\left(1 - \frac{1}{N}\right)^N \approx e^{-1} = 0.368$$

  - Approximately 63.2 % of original data present in sampled set
- Greater dependency between sampled datasets than with cross-validation
  - Still excellent for evaluating performance with small datasets
  - Also good for evaluating the stability of an algorithm

14.6 Error measurements and ROC curves

Error measurement and confusion matrix

- Confusion matrix: explanation of the errors made

	Decision	
Truth	1	0
1	$\|TP\|$	$\|FN\|$
0	$\|FP\|$	$\|TN\|$

- Error rate redefinition: $E = \frac{|FN|+|FP|}{N}$
  - With $N = |TP| + |FP| + |TN| + |FN|$
- Weighting by type of error (variable costs)

$$E = \frac{c_{FN}|FN| + c_{FP}|FP|}{N}$$

- Direct generalization to K classes

ROC curves

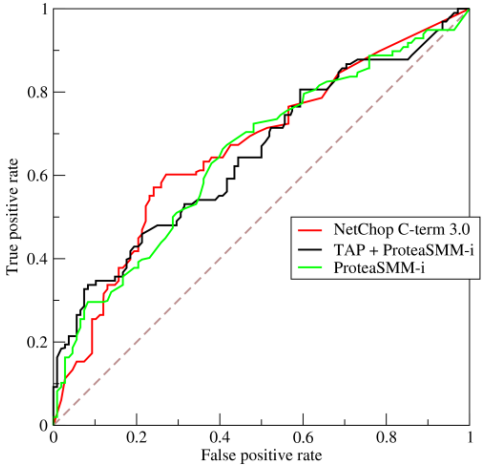
- ROC curve (*receiver operator characteristics*)
  - Rate of correct decisions

$$\frac{|TP|}{|TP| + |FN|}$$

  - False alarm rate

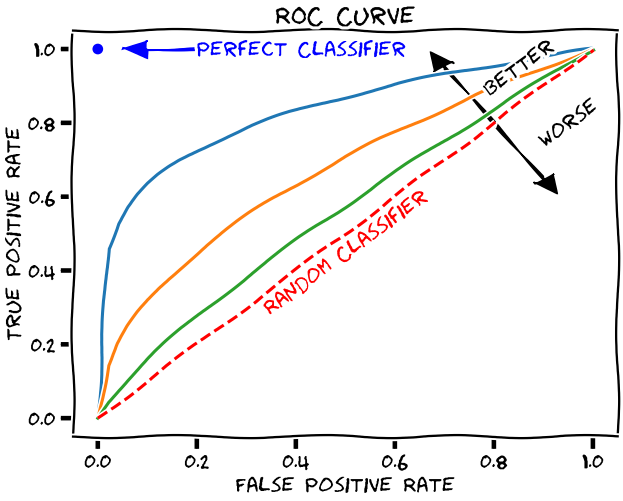
$$\frac{|FP|}{|FP| + |TN|}$$

  - Different acceptance thresholds give different operation points on the curve



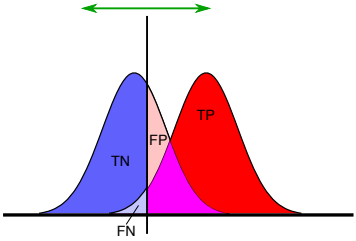
By BOR, CC-BY-SA 3.0,
https://commons.wikimedia.org/wiki/File:Roccurves.png.

ROC curves for classification

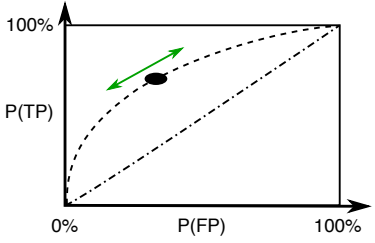


By MartinThoma, public domain, https://commons.wikimedia.org/wiki/File:Roc-draft-xkcd-style.svg.

ROC curve decision threshold



TP	FP
FN	TN



By Sharpr, CC-BY-SA 3.0, https://commons.wikimedia.org/wiki/File:ROC_curves.svg.

AUC-ROC, sensitivity and specificity

- Area under the ROC (AUC-ROC) curve: threshold-independent performance measurement
  - Ability of the classifier to properly discriminate two classes for all thresholds
  - Similarity with nonparametric Wilcoxon-Mann-Whitney test
- Sensitivity: number of correctly identified positives

$$\text{sensibility} = \frac{|TP|}{|TP| + |FP|}$$

- Specificity: number of correctly identified negatives

$$\text{specificity} = \frac{|TN|}{|TN| + |FN|} = 1 - \frac{|FP|}{|TN| + |FN|}$$

Precision and recall

- Searching for information in databases
  - Extracted entries following a query: positive
  - Relevant entries for a query: true positives + false negatives
- Accuracy: # relevant extracted entries by # extracted entries

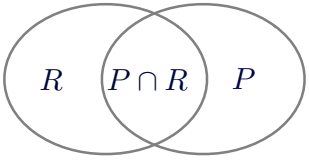
$$\text{precision} = \frac{|TP|}{|TP| + |FP|}$$

  - Accuracy of 1: extracted entries all relevant, but may remain false negatives
  - Equivalent to sensitivity
- Recall: # elevant entries extracted by # relevant entries

$$\text{recall} = \frac{|TP|}{|TP| + |FN|}$$

  - Recall of 1: all relevant entries are retrieved, but there may be irrelevant (false positive) entries retrieved.

Precision and recall

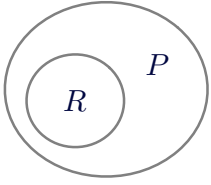


Retrieve items

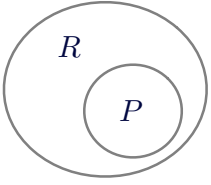
Relevant items

$$\text{Precision} = \frac{|R|}{|P \cap R|}$$

$$\text{Recall} = \frac{|P \cap R|}{|P|}$$



Precision = 1



Recall = 1

14.7 Confidence interval and statistical laws

Confidence interval

- Estimator (e.g. maximum likelihood): a value of a parameter
- Confidence interval: the range of plausible values of a parameter, for a given confidence threshold.
  - Based on the underlying probability density of the estimator
- Example: estimation of mean μ of a normal distribution from samples $\mathcal{X} = \{x^t\}_{t=1}^N$
  - Estimation by average of samples: $m = \sum_t x^t / N$
  - m is a sum of normal variables, and thus also normal, $m \sim \mathcal{N}(\mu, \sigma^2/N)$
  - According to the normal law, we therefore have confidence at 95 % that $\mu \in [m - 1.96\sigma/\sqrt{N}, m + 1.96\sigma/\sqrt{N}]$

$$P\left(m - 1.96\frac{\sigma}{\sqrt{N}} < \mu < m + 1.96\frac{\sigma}{\sqrt{N}}\right) = 0.95$$

Confidence interval

- Law $\mathcal{Z}$: normal law of null mean and unit variance, $\mathcal{Z} \equiv \mathcal{N}(0,1)$
- General formalization of confidence interval for normal law:
  $Z \sim \mathcal{Z}$, $P(Z > z_\alpha) = \alpha$, $\alpha \in [0,1]$
  - Normal law of null mean is symmetrical
    - Single bound: $P(-z_\alpha < Z) = 1-\alpha$, $P(Z < z_\alpha) = 1-\alpha$, $\alpha \in [0,1]$
    - Double bounds: $P(-z_{0.5\alpha} < Z < z_{0.5\alpha}) = 1-\alpha$, $\alpha \in [0,1]$
- Estimation of sample mean $m \sim \mathcal{N}(\mu,\sigma^2/N)$, implies

$$\begin{aligned} \sqrt{N}\frac{m-\mu}{\sigma} &\sim \mathcal{Z} \\ P\left(m - z_\alpha \frac{\sigma}{\sqrt{N}} < \mu\right) &= 1-\alpha \\ P\left(\mu < m + z_\alpha \frac{\sigma}{\sqrt{N}}\right) &= 1-\alpha \end{aligned}$$

χ^2 Law

- If $Z_i \sim \mathcal{Z}$ are independent random variables, and

$$X = Z_1^2 + Z_2^2 + \cdots + Z_n^2$$

  then X follows a law from χ^2 with n degrees of freedom, $X \sim \chi_n^2$
- Expected value of $\mathbb{E}[X] = n$ and variance $\mathrm{Var}(X) = 2n$
- For a sampling $x^t \sim \mathcal{N}(\mu, \sigma^2)$
  - Variance estimate: $s^2 = \frac{\sum_t (x^t - m)^2}{N-1}$
  - $(N-1)\frac{s^2}{\sigma^2} \sim \chi_{N-1}^2$
- χ^2 Law is excellent for performing statistical tests on several random variables according to normal laws
  - For example, several estimates of a classification rate

Student's Law

- Student's Law: suitable for testing on normal distributions where there are few samples.
- If $Z \sim \mathcal{Z}$ and $X \sim \chi^2_n$ are independent, then $T_n \sim t_n$, follows a Student's Law with n degrees of freedom

$$T_n = \frac{Z}{\sqrt{X/n}}$$

  - With large n, the distribution has a shape similar to a normal distribution of mean equal to 0
  - Expected value $\mathbb{E}[T_n] = 0$, variance $\mathrm{Var}(T_n) = \frac{n}{n-2}$, pour $n > 2$

14.8 Statistical tests

Hypothesis testing

- Hypothesis testing: classic method for testing the statistical validity of results
  - Assuming that a random variable follows a certain density law
  - Estimate the probability that the variable meets the hypothesis based on the statistics obtained from the measurements
  - If the probability is sufficiently high, the test is positive (null hypothesis verified)
- t-test (Student's Law)
  - Difference between true mean μ_0 and mean m from N samples, having a variance s, follows a Student's Law with $N-1$ degrees of freedom

$$\frac{\sqrt{N}(m-\mu_0)}{s} \sim t_{N-1}$$

  - Hypothesis verified with probability $1-\alpha$ when:

$$\frac{\sqrt{N}(m-\mu_0)}{s} \in [-t_{0.5\alpha,\,N-1},\, t_{0.5\alpha,\,N-1}]$$

Paired t-test

- Using the t-test for K-fold cross-validation
  - K error percentages p_i on validation sets $\mathcal{V}_i$, $i = 1, \ldots, K$

$$p_i = \frac{\sum_{\mathbf{x}^t \in \mathcal{V}_i} \mathbb{I}(r^t, \mathrm{h}(\mathbf{x}^t | \mathcal{T}_i))}{N}$$

  - Mean and variance of results with K-fold cross-validation

$$m = \frac{\sum_{i=1}^{K} p_i}{K}, \quad s^2 = \frac{\sum_{i=1}^{K} (p_i - m)^2}{K - 1}$$

  - Paired t-test performed according to

$$\frac{\sqrt{K}(m - p_0)}{s} \sim t_{K-1}$$

    where p_0 is the error rate verified by the hypothesis test

- So, error rate less than p_0 with probability $1 - \alpha$ if next test is positive

$$\frac{\sqrt{K}(m - p_0)}{s} < t_{\alpha, K-1}$$

Paired t-test for results comparison

- Comparison of two algorithms trained with K-fold cross-validation
  - p_i^1: classification error on $\mathcal{V}_i$ of the first algorithm trained on $\mathcal{T}_i$
  - p_i^2: classification error on $\mathcal{V}_i$ of the second algorithm trained on $\mathcal{T}_i$
  - Difference of the classification error on fold i: $p_i = p_i^1 - p_i^2$
  - Hypothesis test: mean value of p_i is null
  - Mean and variance of the error difference

$$m = \frac{\sum_{i=1}^{K} p_i}{K}, \quad s^2 = \frac{\sum_{i=1}^{K} (p_i - m)^2}{K - 1}$$

  - The error difference p_i follows a Student's Law with $K - 1$ degrees of freedom

$$\frac{\sqrt{K}(m - 0)}{s} = \frac{\sqrt{K}m}{s} \sim t_{K-1}$$

- Algorithm with statistically identical performance, with probability $1 - \alpha$, if next test is positive

$$\frac{\sqrt{K}m}{s} \in [-t_{0.5\alpha,\, K-1},\, t_{0.5\alpha,\, K-1}]$$

Analysis of variance (ANOVA)

- ANOVA: comparing several classification algorithms
  - How to compare L algorithms, each trained and tested on K pairs of different subsets?
  - Assuming that each result $E_{i,j}$ follows a normal distribution of mean

$$E_{i,j} \sim \mathcal{N}(\mu_j, \sigma^2),\ i = 1, \ldots, K,\ j = 1, \ldots, L$$

    - Average μ_j unknown and different for each algorithm
    - Variance σ^2 shared by all folds/algorithms
  - Hypothesis H_0: all averages μ_j are equal

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_L$$

- ANOVA approach: two different estimators of σ^2
  - First estimator of σ^2 valid only when H_0 is true
  - Second estimator of σ^2 valid no matter how valid H_0 is

First estimator of σ^2 with ANOVA

- First estimator of σ^2: H_0 is valid
  - Average by algorithm on K folds: $m_j = \frac{\sum_{i=1}^{K} e_{i,j}}{K}$
  - Mean and variance of the m_j

$$m = \frac{\sum_{j=1}^{L} m_j}{L}, \quad s^2 = \frac{\sum_{j=1}^{L}(m_j - m)^2}{L-1}$$

  - Estimator of σ^2

$$\hat{\sigma}^2 = Ks^2 = K\frac{\sum_{j=1}^{L}(m_j - m)^2}{L-1}$$

  - As each m_j follows a normal law, we can say

$$\frac{(L-1)s^2}{\sigma^2/K} = \frac{K\sum_{j=1}^{L}(m_j - m)^2}{\sigma^2} \sim \chi^2_{L-1}$$

  - By posing $S_b \equiv K\sum_{j=1}^{L}(m_j - m)^2$, we get H_0 is valid when

$$\frac{S_b}{\sigma^2} \sim \chi^2_{L-1}$$

Second estimator of σ^2 with ANOVA

- Second estimator of σ^2: independent of validity of H_0
  - σ^2: mean of the variance s_j^2 of the algorithms

$$
\begin{aligned}
s_j^2 &= \frac{\sum_{i=1}^{K}(e_{i,j} - m_j)^2}{K-1} \\
\hat{\sigma}^2 &= \sum_{j=1}^{L}\frac{s_j^2}{L} = \sum_{j=1}^{L}\sum_{i=1}^{K}\frac{(e_{i,j} - m_j)^2}{L(K-1)}
\end{aligned}
$$

  - By posing $S_w \equiv \sum_{j=1}^{L}\sum_{i=1}^{K}(e_{i,j} - m_j)^2$

$$
(K-1)\sum_{j=1}^{K}\frac{s_j^2}{\sigma^2} = (K-1)\sum_{j=1}^{K}\frac{\sum_{i=1}^{K}(e_{i,j} - m_j)^2}{(K-1)\sigma^2} = \frac{S_w}{\sigma^2} \sim \chi^2_{L(K-1)}
$$

ANOVA

- Fisher's law: ratio of two independent χ^2 laws

$$F_{n,m} = \frac{X_1/n}{X_2/m}, \quad \text{où } X_1 \sim \chi^2_n \text{ et } X_2 \sim \chi^2_m$$

- ANOVA: reject hypothesis H_0 if the two estimators of σ^2 differ significantly

$$\begin{aligned} H_0 : \mu_1 &= \mu_2 = \cdots = \mu_L \\ \frac{\frac{S_b/\sigma^2}{L-1}}{\frac{S_w/\sigma^2}{L(K-1)}} &= \frac{S_b/(L-1)}{S_w/(L(K-1))} = \frac{L(K-1)}{L-1}\frac{S_b}{S_w} \sim F_{L-1,L(K-1)} \end{aligned}$$

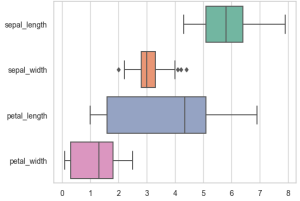
- Therefore, hypothesis that average classification rates are equal for all algorithms is valid at a probability $1-\alpha$ when

$$\frac{L(K-1)}{L-1}\frac{S_b}{S_w} < F_{\alpha,L-1,L(K-1)}$$

14.9 Python tools for experimentation

Python tools for experimentation

- `sklearn.model_selection.cross_val_score`: K-fold cross-validation
- `scipy.stats.ttest_rel` and `scipy.stats.ttest_ind`: t-test, paired or independent
- `scipy.stats.f_oneway`: analysis of variance (ANOVA)
- `seaborn.boxplot`: graphical comparison of several results (requires Seaborn library)



From `https://seaborn.pydata.org/generated/seaborn.boxplot.html`.

- Auto-sklearn: AutoML with scikit-learn
  `https://automl.github.io/auto-sklearn/master/`