Introduction to Supervised Learning

Performance Evaluation

Marcelo S. Lauretto

Escola de Artes, Ciências e Humanidades,
Universidade de São Paulo
marcelolauretto@usp.br

Lima - Peru

Performance Evaluation

- Issues: training, testing
- Confusion matrix
- Performance indicators
- Holdout, cross-validation, bootstrap
- Predicting performance: confidence limits
- Comparing classification algorithms: t-test, non-parametric test
- Parameter tuning

Performance Evaluation

- How good is the classifier?
- Natural performance measure for classification problems: *error rate*
  - *Success*: instance's class is predicted correctly
  - *Error*: instance's class is predicted incorrectly *Error rate*: proportion of errors made over the whole set of instances
- *Resubstitution error*: error rate obtained from training data
  - Extremely optimistic – particularly if the classifier overfits

Training & Test Sets

- *Training set*: instances used to train (induce) the classifier
- *Test set*: independent instances that have played no part in formation of classifier
  - Assumption: both training data and test data are representative samples of the underlying problem
- Generally, the larger the training data the better the classifier
- The larger the test data the more accurate the error estimate
- *Holdout* procedure: method of splitting original data into training and test set
  - Dilemma: ideally both training set and test set should be large!

Predicting Performance

- Assume the estimated error rate is 25%. How close is this to the true error rate?
  - Depends on the amount of test data
- Prediction is just like tossing a (biased!) coin
  - “Head” is a “success”, “tail” is an “error”
- In statistics, a succession of independent events like this is called a Bernoulli process
  - Statistical theory provides us with confidence intervals for the true underlying proportion

Confidence Intervals for Success Rate

- We can say: the true success rate (denote by p) lies within a certain specified interval with a certain specified confidence
- Example: $S = 750$ successes in $N = 1000$ trials
  - Estimated success rate: 75%
  - How close is this to true success rate p?
    - Answer: with 80% confidence p in [73.2,76.7]
- Another example: $S = 75$ and $N = 100$
  - Estimated success rate: 75%
  - With 80% confidence p in [69.1,80.1]

Confidence Intervals for Success Rate

- Mean and variance for a Bernoulli trial:
  - $\mu = p, \; V = p(1-p)$
- Expected success rate: sample mean $\hat{p} = S/N$
- *Central Limit Theorem*: For large enough N, $\hat{p}$ follows a Normal Distribution with mean $\mu = p$ and variance $V/N = p(1-p)/N$
- A c% probability interval $[-z \leq X \leq z]$ for a random variable with mean 0 is given by:

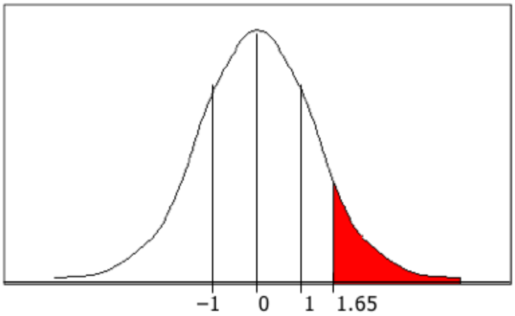
$$\Pr[-z \leq X \leq z] = c\%$$

- For a symmetric distribution:

$$\begin{aligned} c &= \Pr[-z \leq X \leq z] = 1 - 2\Pr[X \geq z] \\ \Longrightarrow & \qquad \Pr[X \geq z] = \frac{1-c}{2} \end{aligned}$$

Confidence Intervals for Success Rate – Bernoulli Process

- Confidence limits for a variable X with standard normal distribution (mean 0 and variance 1):



$\Pr[X \geq z]$	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
40%	0.25

- Thus:

$$\Pr[-1.65 \leq X \leq +1.65] = 90\%$$

- To use this we have to *standardize* $\hat{p}$ to have 0 mean and unit variance

Confidence Intervals – Standard Normal Distribution

- Transformed value for $\hat{p}$:
$$\frac{\hat{p}-p}{\sqrt{p(1-p)/N}}$$
(i.e. subtract the mean and divide by the standard deviation)
- Resulting equation:
$$\Pr\left[-z \leq \frac{\hat{p}-p}{\sqrt{p(1-p)/N}} \leq z\right] = c$$
- Transforming inequalities in equalities and solving for p:
$$p \in \left[\hat{p} + \frac{z^2}{2N} \pm z\sqrt{\frac{\hat{p}}{N} - \frac{\hat{p}^2}{N} + \frac{z^2}{4N^2}} \bigg/ \left(1 + \frac{z^2}{N}\right)\right]$$

Confidence Intervals – Standard Normal Distribution

- Examples
  - $\hat{p} = 75\%, N = 1000, c = 80\%$ (so that z = 1.28):
    $p \in [0.732, 0.767]$
  - $\hat{p} = 75\%, N = 100, c = 80\%$ (so that z = 1.28):
    $p \in [0.691, 0.801]$
  - $\hat{p} = 75\%, N = 10, c = 80\%$ (so that z = 1.28):
    $p \in [0.549, 0.881]$!!
- Normal approximation for Bernoulli processes is only valid for large N (i.e. N > 100)

Holdout estimation

- What to do if the amount of data is limited?
- The *holdout* method reserves a certain amount for testing and uses the remainder for training
    - Usually: one third for testing, the rest for training
- Problem: the samples might not be representative
    - Example: class might be missing in the test data
- Advanced version uses stratification
    - Ensures that each class is represented with approximately equal proportions in both subsets

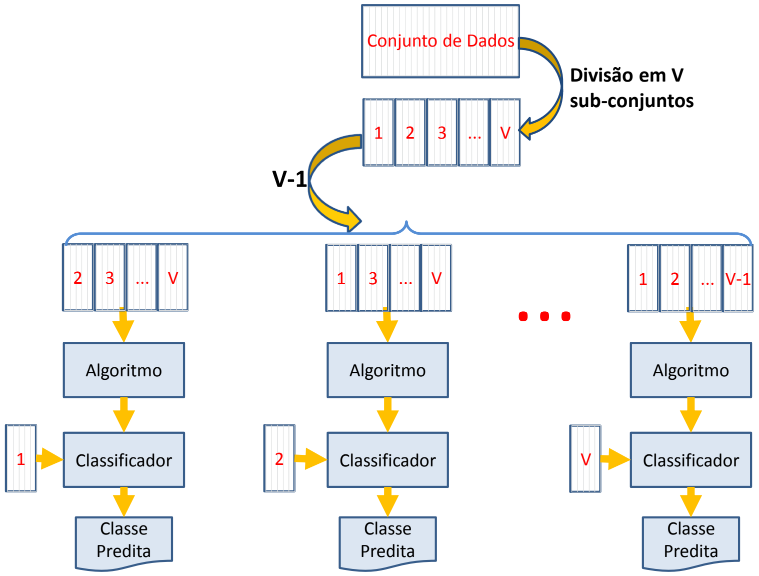
Repeated holdout method

- Holdout estimate can be made more reliable by repeating the process with different subsamples
  - In each iteration, a certain proportion is randomly selected for training (possibly with stratificiation)
  - The error rates on the different iterations are averaged to yield an overall error rate
- This is called the repeated holdout method
- Still not optimum: the different test sets overlap
  - Can we prevent overlapping?
- Problem: the samples might not be representative
  - Example: class might be missing in the test data
- Advanced version uses stratification
  - Ensures that each class is represented with approximately equal proportions in both subsets

Cross-validation

- *Cross-validation* avoids overlapping test sets
    - First step: split data into *k* subsets of equal size
    - Second step: use each subset in turn for testing, the remainder for training
- Called *k-fold cross-validation*
- Subsets may be stratified
- The error estimates are averaged to yield an overall error estimate
- Standard method for evaluation: stratified ten-fold cross-validation
- Best variant: Repeated stratified cross-validation
    - E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

Cross-validation – Diagram

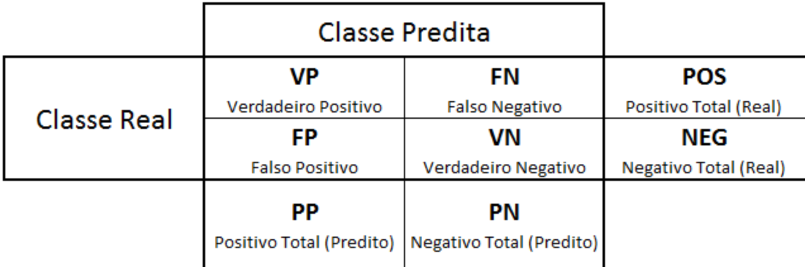


Leave-One-Out Cross-validation

- *Leave-One-Out*: particular form of cross-validation
  - Set number of folds to number of training instances
  - I.e., for n training instances, build classifier n times
- Pros:
  - Makes best use of the data for training
  - Involves no random subsampling
- Cons:
  - Very computationally expensive
  - Cannot be stratified
    - There is only one instance in each test set!
  - Extreme (artificial) illustration: random dataset with the same number of instances of each of two classes
    - Best inducer predicts majority class
    - 50% accuracy on fresh data
    - Leave-One-Out yield an estimated error of 100%!

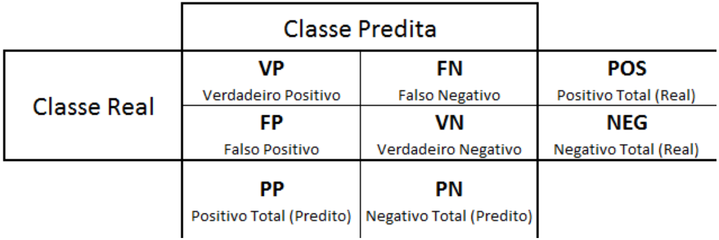
Confusion Matrix

- Applying the classifier on a test set yields a *confusion matrix*, a bi-dimensional contingency table formed by the absolute frequencies of real and predicted classes of test instances
- For binary classification (two classes):



	Classe Predita		
Classe Real	**VP** Verdadeiro Positivo	**FN** Falso Negativo	**POS** Positivo Total (Real)
	FP Falso Positivo	**VN** Verdadeiro Negativo	**NEG** Negativo Total (Real)
	PP Positivo Total (Predito)	**PN** Negativo Total (Predito)	

Confusion Matrix

- 

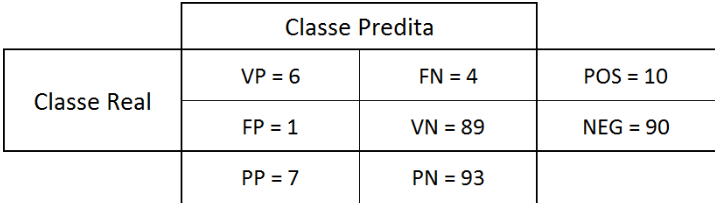
	Classe Predita		
Classe Real	**VP** Verdadeiro Positivo	**FN** Falso Negativo	**POS** Positivo Total (Real)
	FP Falso Positivo	**VN** Verdadeiro Negativo	**NEG** Negativo Total (Real)
	PP Positivo Total (Predito)	**PN** Negativo Total (Predito)	

- TP (True Positives), TN (True Negatives): instances classified correctly
- FN (False Negatives), FP (False Positives): misclassified instances
- POS: positive instances: $POS = TP + FN$;
- NEG: negative instances $NEG = FP + TN$;
- PP (Prediced Positive): $PP = TP + FP$
- PN (Prediced Negative): $PN = TN + FN$.

Performance Measures

- *Total error rate, total accuracy rate*: The most used measures
  $E_r = (FN + FP)/(NEG + POS)$; $Acc = (1 - E_r)$
- *True Positive Rate* (also called *sensitivity* or *recall*): $TP_r = TP/POS$
  True Negative Rate (also called *specificity*): $TN_r = TN/NEG$
  False Negative Rate: $FN_r = FN/POS$
  False Positive Rate: $FP_r = FP/NEG$
- *Precision rate:* Proportion of positive class instances among those predicted as positive.
  $Prec_r = TP/PP$
  - Good measure for high-cost misclassification of negative cases:
    - In stock markets, if a trader decides to start a buy & hold operation, its success rate must be high
  - Ineffective for very low predicted positive rates
    - Usually, low $PP \Rightarrow$ high FN
    - It is not defined if $PP = 0$

Performance Measures – Example



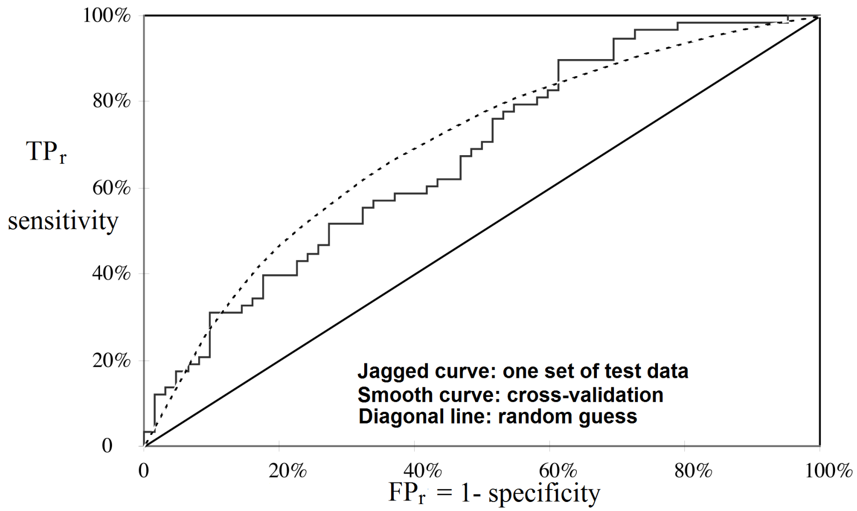
	Classe Predita		
Classe Real	VP = 6	FN = 4	POS = 10
	FP = 1	VN = 89	NEG = 90
	PP = 7	PN = 93	

- $TP_r = 6/10 = 60\%$; $\quad TN_r = 89/90 = 98.9\%$
  $FN_r = 4/10 = 40\%$; $\quad FP_r = 01/90 = 1.1\%$
- Total error rate: $ET_r = (4+1)/(10+90) = 5\%$
- Precision rate: $Prec_r = 6/7 = 85.7\%$
- Excellent for predicting negative class; very bad for predicting positive class

ROC Curve

- *ROC*: Receiver Operating Characteristic
  - Used in signal detection to show tradeoff between hit rate and false alarm rate over noisy channel http://psych.hanover.edu/JavaTest/SDT/index.html
  - Common use for calibrating medical diagnostic tests http://gim.unmc.edu/dxtests/Default.htm
- ROC curve is obtained by plotting the FP_r (or $1-$specificity) on horizontal axis and TP_r (sensitivity) on vertical axis
- Suitable for
  - tuning parameters of algorithms for the adequate trade-off between sensitivity and specificity
  - comparing algorithms performances

ROC Curve



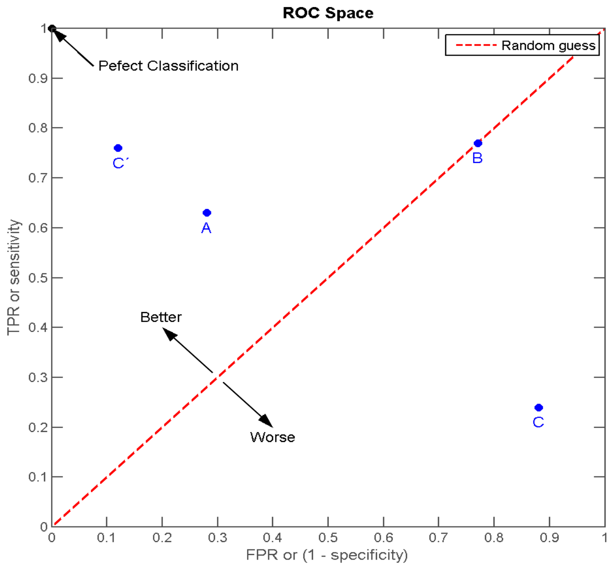
Jagged curve: one set of test data
Smooth curve: cross-validation
Diagonal line: random guess

ROC Curve

- Optimum point: Perfect classification
  - 100% true positives, 0% false positives
- The closer the point (FP_r, TP_r) to (0%, 100%), the better the algorithm
- A completely random guess (with variable probability of positive assignment) would give a point along a diagonal line (*line of no-discrimination*)

```
Source of image in next slide:
"ROC space-2" by ROC_space.png: Indonderivative work:
Kai walz (talk) - ROC_space.png.
Licensed under CC BY-SA 3.0 via Wikimedia Commons
https://commons.wikimedia.org/wiki/File:ROC_space-2.png#/media/
       File:ROC_space-2.png
```

ROC – Examples



ROC – historical note

- ROC analysis is part of a field called "Signal Dectection Theory"developed during World War II for the analysis of radar images.
    - Radar operators had to decide whether a blip on the screen represented an enemy target, a friendly ship, or just noise.
    - Signal detection theory measures the ability of radar receiver operators to make these important distinctions.
    - Their ability to do so was called the Receiver Operating Characteristics.
    - It was not until the 1970's that signal detection theory was recognized as useful for interpreting medical test results.

Area under the ROC Curve

- Area under the curve (AUC): one of ROC summary statistics
- Corresponds to the integral

$$\int_{-\infty}^{\infty} TP_r(t) FP_r(t) dt$$

  where t is a (continuous) sensitivity-related parameter
- Evaluation:
  - AUC = 1: perfect classifier
  - AUC=0.5: worthless classifier (random guess)
  - In Medicine: criterion for classifying the accuracy of a diagnostic test:
    - .90 – 1 = excellent (A)
    - .80 – .90 = good (B)
    - .70 – .80 = fair (C)
    - .60 – .70 = poor (D)
    - .50 – .60 = fail (F)

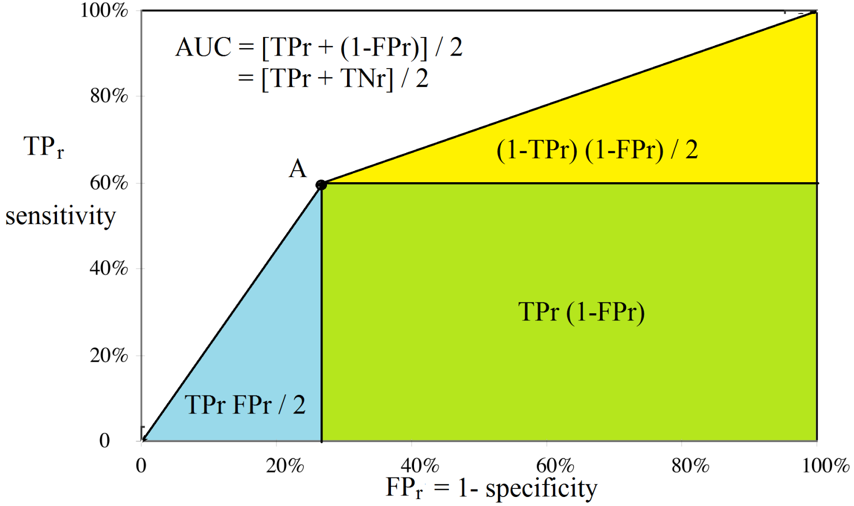
Area under the ROC Curve

- AUC measures *discrimination*, that is, the ability of a classifier to correctly classify instances of positive and negative classes
    - Probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative')
- Computation:
    - For a single test point (corresponding to a unique test sample and unique t value), the AUC may be estimated by the mean and sensitivity and specificity:

      $$AUC = (TP_r + TN_r)/2$$

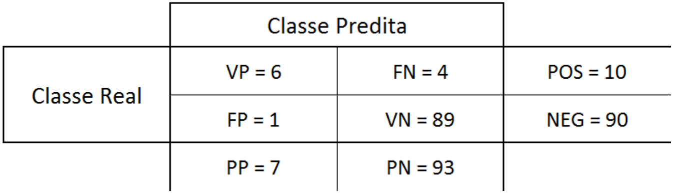
      (Figure in the next slide)
    - Given several test points, build trapeziods under the curve as an approximation of area (extension of the single point case above)

Area under the ROC Curve – Estimation



Unbalanced Datasets: Accuracy vs AUC

- Unbalanced datasets:
  - High prevalence of one class
- In such cases, Accuracy measure may be (optimistically) misleading
- Taking the previous example:



	Classe Predita		
Classe Real	VP = 6	FN = 4	POS = 10
	FP = 1	VN = 89	NEG = 90
	PP = 7	PN = 93	

  - $TP_r = 60\%$; $TN_r = 98.9\%$; $FN_r = 40\%$; $FP_r = 1.1\%$
  - $Acc = 95\%$
    High accuracy rate, but... an error rate of 40% in positive class!

Unbalanced Datasets: Accuracy vs AUC

- Taking the previous example:
  - $TP_r = 0.6$; $TN_r = 0.989$; $FN_r = 0.4$; $FP_r = 0.011$
  - $Acc = 95\%$
    High accuracy rate, but... an error rate of 40% in positive class!
  - If positive class corresponded to a disease, 40% of ill patients would be classified as healthy!
- AUC estimate:

$$AUC = (TP_r + TN_r)/2 = (0.6 + 0.989)/2 = 0.795$$

According to AUC reference table, just *fair*!

Comparing Learning Algorithms

- Frequent question: which of two learning schemes performs better?
- Note: this is domain dependent!
- Obvious way: compare 10-fold CV estimates
- Generally sufficient in applications (we don't loose if the chosen method is not truly better)
- However, what about machine learning research?
  - Need to show convincingly that a particular method works better
- A possible answer for this question is to use statistical techniques
  - Confidence intervals and significance tests

Comparing Algorithms – Confidence Intervals

- Notation:
  - ψ: classification algorithm
  - $\psi_{\mathcal{L}_{tr}}(\bullet)$: a classifier inducted by algorithm ψ using training set $\mathcal{L}_{tr}$
  - $\psi_{\mathcal{L}_{tr}}(\mathcal{L}_{ts})$: classes predicted by $\psi_{\mathcal{L}_{tr}}(\bullet)$ for instances of set $\mathcal{L}_{ts}$
  - $M_{\mathcal{L}_{ts},\psi_{\mathcal{L}_{tr}}(\mathcal{L}_{ts})}$: the confusion matrix yielded by true and predicted classes of $\mathcal{L}_{ts}$
  - $h\left(M_{\mathcal{L}_{ts},\psi_{\mathcal{L}_{tr}}(\mathcal{L}_{ts})}\right)$: a performance measure (accuracy, total error, AUC, etc) yielded from confusion matrix $M_{\mathcal{L}_{ts},\psi_{\mathcal{L}_{tr}}(\mathcal{L}_{ts})}$
- Given two distinct algorithms ψ and φ, the random variable of interest is the difference between the measured performances:

$$\delta = h\left(M_{\mathcal{L}_{ts},\psi_{\mathcal{L}_{tr}}(\mathcal{L}_{ts})}\right) - h\left(M_{\mathcal{L}_{ts},\varphi_{\mathcal{L}_{tr}}(\mathcal{L}_{ts})}\right)$$

$$\forall(\mathcal{L}_{ts}, \mathcal{L}_{tr}) \in \mathcal{P}(\mathcal{X} \times \{1 \ldots K\})^2.$$

Comparing Algorithms – Confidence Intervals

- We denote by $\mu_\delta = E_{\mathcal{P}(\mathcal{X}\times\{1\ldots K\})^2}(\delta)$
  - i.e. the mean of performance differences between ψ_A and ψ_B over all possible pairs of training and sample tests
- If algorithms ψ and φ perform equally, then $\mu_\delta = 0$.
- μ_δ is unknown $\Rightarrow$ we may obtain a confidence interval for it.
- One $(1-\alpha)$% confidence interval for μ_δ:
  - Interval $[a, b]$ yielded from a sample that should include the true value of μ_δ, with probability $1-\alpha$,
  - We say that θ belongs to interval $[a, b]$ with *confidence* $1-\alpha$.
- We shall see a method for obtaining the confidence interval via Cross Validation

Comparing Algorithms – Cross Validation

Input: $\mathcal{L}$: Original dataset V: Number of CV folds

```
Partition $\mathcal{L}$ in $V$ disjoinct subsets $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_V$ of the same size
for $v$ from 1 to $V$ do
    Take $\mathcal{L}_v$ as test set and $\mathcal{L}_v^c = \mathcal{L} - \mathcal{L}_v$ as training set
    Build classifiers $\psi_{\mathcal{L}_v^c}(\bullet)$ and $\varphi_{\mathcal{L}_v^c}(\bullet)$ using $\mathcal{L}_v^c$
    Apply both classifiers on test set $\mathcal{L}_v$, yieldind the confusion matrices
    $M_{\mathcal{L}_v,\psi_{\mathcal{L}_v^c}(\mathcal{L}_v)}$ and $M_{\mathcal{L}_v,\varphi_{\mathcal{L}_v^c}(\mathcal{L}_v)}$
    Compute the performance difference:
        $\delta_v = h\left(M_{\mathcal{L}_v,\psi_{\mathcal{L}_v^c}(\mathcal{L}_v)}\right) - h\left(M_{\mathcal{L}_v,\varphi_{\mathcal{L}_v^c}(\mathcal{L}_v)}\right)$
end
Return the mean of $\delta_1, \delta_2, \ldots, \delta_V$ and its corresponding standard error:
```

$$\overline{\delta} = \frac{1}{V}\sum_{v=1}^{V}\delta_v, s_{\overline{\delta}} = \sqrt{\frac{1}{V(V-1)}\sum_{i=1}^{V}(\delta_i - \overline{\delta})^2}$$

Comparing Algorithms – Confidence Intervals

- If V is large ($V > 100$): approximation by standard sormal distribution
- If V is small: approximation by Student t distribution
- Confidence interval for μ_δ using Student t distribution:

$$\mu_\delta \in [a, b] = \left[\overline{\delta} \pm z \, s_{\overline{\delta}}\right]$$

where z represents the quantile $1 - \alpha/2$ of Student t distribution with $V - 1$ degrees of freedom.

Comparing Algorithms – Confidence Intervals

- To test the hipothesys $H_0 : \mu_\delta = 0$ (i.e. algorithms ψ and φ perform equally well):
  - Build the confidence interval $[a, b]$
  - if $0 \in [a, b]$: we *don't reject* reject H_0
    $\Rightarrow$ Differences of performances are *not* significant on a $(1 - \alpha)$% confidence level
    $\Rightarrow$ Algorithms are considered as equivalent in performance
  - if $0 \notin [a, b]$: we *reject* H_0
    $\Rightarrow$ Differences of performances are significant on a $(1 - \alpha)$% confidence level
    $\Rightarrow$ The algorithm with larger CV average performance is considered better
- Usual values for the confidence level:
  - $0.90, 0.95$ e 0.99 ($\alpha = 0.1, 0.05$ e 0.01, respectively)
- The higher the confidence level:
  - the larger (less precise) the interval
  - the higher the chance of accepting H_0