UNIT V DESIGN AND ANALYSIS OF MACHINE LEARNING EXPERIMENTS 8

Guidelines for machine learning experiments, Cross Validation (CV) and resampling – K-fold CV, bootstrapping, measuring classifier performance, assessing a single classification algorithm and comparing two classification algorithms – t test, Mc Nemar's test, K-fold CV paired t test

5.1 Guidelines for Machine Learning Experiments

The steps in machine learning are the same as for any type of experimentation, that at this point, it is not important whether the task is classification or regression, or whether it is an unsupervised or a reinforcement learning application. The same overall discussion applies; the difference is only in the sampling distribu- tion of the response data that is collected.

A. Aim of the Study

Given two learning algorithms and a particular problem as defined bya dataset, we may want to determine which one has less generalization error. These can be two different algorithms, or one can be a proposed improvement of the other, for example, by using a better feature extrac-tor.In the general case, we may have more than two learning algorithms, and we may want to choose the one with the least error, or order them in terms of error, for a given dataset.In an even more general setting, instead of on a single dataset, we may want to compare two or more algorithms on two or more datasets.

B. Selection of the Response Variable

We need to decide on what we should use as the quality measure. Most frequently, error is used that is the misclassification error for classifica- tion and mean square error for regression. We may also use some variant;for example, generalizing from 0/1 to an arbitrary loss, we may use a riskmeasure. In information retrieval, we use measures such as precision andrecall. In a cost-sensitive setting, not only the output butalso system parameters, for example, its complexity, are taken into account.

C. Choice of Factors and Levels

What the factors are depend on the aim of the study. If we fix an al- gorithm and want to find the best hyperparameters, then those are the factors. If we are comparing algorithms, the learning algorithm is a fac-tor. If we have different datasets, they also become a factor.

The levels of a factor should be carefully chosen so as not to miss a

good configuration and avoid doing unnecessary experimentation. It is always good to try to normalize factor levels. For example, in optimizing k of k-nearest neighbor, one can try values such as 1, 3, 5, and so on, but in optimizing the spread h of Parzen windows, we should not try absolute values such as 1.0, 2.0, and so on, because that depends on the scale of the input; it is better to find some statistic that is an indicator of scale—for example, the average distance between an instance and its nearest neighbor—and try h as different multiples of that statistic.

D. Choice of Experimental Design

It is always better to do a factorial design unless we are sure that the factors do not interact, because mostly they do. Replication number de-pends on the dataset size; it can be kept small when the dataset is large;we will discuss this in the next section when we talk about resampling.However, too few replicates generate few data and this will make com-paring distributions difficult; in the particular case of parametric tests, the assumptions of Gaussianity may not be tenable.

Generally, given some dataset, we leave some part as the test set and use the rest for training and validation, probably many times by resam-pling. How this division is done is important. In practice, using small datasets leads to responses with high variance, and the differences will not be significant and results will not be conclusive.

It is also important to avoid as much as possible toy, synthetic data and use datasets that are collected from real-world under real-life cir-cumstances. Didactic one- or two-dimensional datasets may help provide intuition, but the behavior of the algorithms may be completely different in high-dimensional spaces.

E. Performing the Experiment

Before running a large factorial experiment with many factors and levels,it is best if one does a few trial runs for some random settings to check that all is as expected. In a large experiment, it is always a good idea to save intermediate results (or seeds of the random number generator), so that a part of the whole experiment can be rerun when desired. All the results should be reproducable. In running a large experiment with many factors and factor levels, one should be aware of the possible negative effects of software aging.

It is important that an experimenter be unbiased during experimen-tation. In comparing one's favorite algorithm with a competitor, both should be investigated equally diligently. In large-scale studies, it may even be envisaged that testers be different from developers.

One should avoid the temptation to write one's own "library" and in- stead, as much as possible, use code from reliable sources; such code would have been better tested and optimized.

As in any software development study, the advantages of good docu-mentation cannot be underestimated, especially when working in

groups.All the methods developed for high-quality software engineering should also be used in machine learning experiments.

F. Statistical Analysis of the Data

This corresponds to analyzing data in a way so that whatever conclusion we get is not subjective or due to chance. We cast the questions thatwe want to answer in a hypothesis testing framework and check whether the sample supports the hypothesis. For example, the question "Is *A* a more accurate algorithm than *B*?" becomes the hypothesis "Can we say that the average error of learners trained by *A* is significantly lower than the average error of learners trained by *B*?"

As always, visual analysis is helpful, and we can use histograms of errordistributions, whisker-and-box plots, range plots, and so on

G. Conclusions and Recommendations

Once all data is collected and analyzed, we can draw objective conclu- sions. One frequently encountered conclusion is the need for further experimentation. Most statistical, and hence machine learning or data mining, studies are iterative. It is for this reason that we never start with all the experimentation. It is suggested that no more than 25 percent of the available resources should be invested in the first experiment (Mont- gomery 2005). The first runs are for investigation only. That is also why it is a good idea not to start with high expectations, or promises to one's boss or thesis advisor.

We should always remember that statistical testing never tells us if the hypothesis is correct or false, but how much the sample seems to concur with the hypothesis. There is always a risk that we do not have aconclusive result or that our conclusions be wrong, especially if the data is small and noisy.

When our expectations are not met, it is most helpful to investigate whythey are not. For example, in checking why our favorite algorithm *A* has worked awfully bad on some cases, we can get a splendid idea for some improved version of *A*. All improvements are due to the deficiencies of the previous version; finding a deficiency is but a helpful hint that there is an improvement we can make!

But we should not go to the next step of testing the improved version before we are sure that we have completely analyzed the current data andlearned all we could learn from it. Ideas are cheap, and useless unless tested, which is costly.

5.2 Cross-Validation and Resampling Methods

- For replication purposes, our first need is to get a number of training and validation set pairs from a dataset X (after having left out some part as the test set).
- To get them, if the sample X is large enough, we can randomly divide it into K parts, then randomly divide each part into two and use one half for training and the other half for validation.

- K is typically 10 or 30. Unfortunately, datasets are never large enough to do this.
- So we should do our best with small datasets.
- This is done cross-validation by repeated use of the same data split differently; this is called crossvalidation.
- The catch is that this makes the error percentages dependent as these different sets share data.
- So, given a dataset X, we would like to generate K training/validation set pairs, {Ti, Vi}K i=1, from this dataset.
- We would like to keep the training and validation sets as large as possible so that the error estimates are robust, and at the same time, we would like to keep the overlap between different sets as small as possible.
- We also need to make sure that classes are represented in the right proportions when subsets of data are held out, not to disturb the class prior probabilities; this is called stratification.
- If a class has 20 percent examples in the whole dataset, in all samples drawn from the dataset, it should also have approximately 20 percent examples.

5.3K-Fold Cross-Validation

- K-fold In K-fold cross-validation, the dataset X is divided randomly into K equalcross-validation sized parts, Xi, i = 1,...,K.
- To generate each pair, we keep one of the K parts out as the validation set and combine the remaining K − 1 parts to form the training set.
- Doing this K times, each time leaving out another one of the K parts out, we get K pairs: V1 = X1 T1 = X2 U X3 U···UXK V2 = X2 T2 = X1 U X3 U···UXK . . . VK = XK TK = X1 U X2 U···UXK−1
- There are two problems with this. First, to keep the training set large, we allow validation sets that are small.
- Second, the training sets overlap considerably, namely, any two training sets share K − 2 parts. K is typically 10 or 30.
- As K increases, the percentage of training instances increases and we get more robust estimators, but the validation set becomes smaller.
- Furthermore, there is the cost of training the classifier K times, which increases as K is increased.
- As N increases, K can be smaller; if N is small, K should be large to allow large enough training leave-one-out sets.
- One extreme case of K-fold cross-validation is leave-one-out where given a dataset of N instances, only one instance is left out as the validation set (instance) and training uses the N − 1 instances.

- We then get N separate pairs by leaving out a different instance at each iteration.
- This is typically used in applications such as medical diagnosis, where labeled data is hard to find.
- Leave-one-out does not permit stratification. Recently, with computation getting cheaper, it has also become possible to have multiple runs of K-fold cross-validation, for example, 10×10- fold, and use average over averages to get more reliable error estimates

5×2 Cross-Validation

- Dietterich (1998) proposed the 5 × 2 cross-validation, which uses training and validation sets of equal size.
- We divide the dataset X randomly into two parts, $X^{(1)}_1$ and $X^{(2)}_1$, which gives our first pair of training and validation sets, $T1 = X^{(1)}_1$ and $V1 = X^{(2)}_1$.
- Then we swap the role of the two halves and get the second pair: $T2 = X^{(2)}_1$ and $V2 = X^{(1)}_1$.
- This is the first fold; $X^{(j)}_i$ denotes half j of fold i. To get the second fold, we shuffle X randomly and divide this new fold into two, $X^{(1)}_2$ and $X^{(2)}_2$.
- This can be implemented by drawing these from X randomly without replacement, namely, $X^{(1)}_1 \cup X^{(2)}_1 = X^{(1)}_2 \cup X^{(2)}_2 = X$.
- We then swap these two halves to get another pair. We do this for three more folds and because from each fold, we get two pairs, doing five folds, we get ten training and validation sets: $T1 = X^{(1)}_1 V1 = X^{(2)}_1 T2 = X^{(2)}_1 V2 = X^{(1)}_1$ $T3 = X^{(1)}_2 V3 = X^{(2)}_2 T4 = X^{(2)}_2 V4 = X^{(1)}_2 \ldots T9 = X^{(1)}_5 V9 = X^{(2)}_5 T10 = X^{(2)}_5 V10 = X^{(1)}_5$ Of course, we can do this for more than five folds and get more training/validation sets, but Dietterich (1998) points out that after five folds, the sets share many instances and overlap so much that the statistics calculated from these sets, namely, validation error rates, become too dependent and do not add new information.
- Even with five folds, the sets overlap and the statistics are dependent, but we can get away with this until five folds. On the other hand, if we do have fewer than five folds, we get less data (fewer than ten sets) and will not have a large enough sample to fit a distribution to and test our hypothesis on

Table 19.1 Confusion matrix for two classes.

	Predicted class		
True Class	Positive	Negative	Total
Positive	*tp* : true positive	*fn* : false negative	*p*
Negative	*fp* : false positive	*tn* : true negative	*n*
Total	p'	n'	*N*

Bootstrapping

- To generate multiple samples from a single sample, an alternative to bootstrap cross-validation is the bootstrap that generates new samples by drawing instances from the original sample with replacement.
- We saw the use of bootstrapping in section 17.6 to generate training sets for different learners in bagging.
- The bootstrap samples may overlap more than cross-validation samples and hence their estimates are more dependent; but is considered the best way to do resampling for very small datasets.
- In the bootstrap, we sample N instances from a dataset of size N with replacement.
- The original dataset is used as the validation set. The probability that we pick an instance is 1/N; the probability that we do not pick it is $1 - 1/N$.
- The probability that we do not pick it after N draws is

$$(1 - 1/N)^N \approx e^{-1} = 0.368$$

This means that the training data contains approximately 63.2 percent of the instances; that is, the system will not have been trained on 36.8 percent of the data, and the error estimate will be pessimistic. The solution is replication, that is, to repeat the process many times and look at the average behavior.

5.3 Measuring Classifier Performance

For classification, especially for two-class problems, a variety of measures has been proposed. There are four possible cases, as shown in table 19.1. For a positive example, if the prediction is also positive, this is a true positive; if our prediction is negative for a positive example, this is a false negative. For a negative example, if the prediction is also negative, we

Table 19.2 Performance measures used in two-class problems.

Name	Formula
error accuracy	*(f p + fn)/N* *(tp + tn)/N* = 1-error
tp-rate fp-rate	*tp/p* *fp/n*
precision recall	*tp/p'* *tp/p* = tp-rate
sensitivity specificity	*tp/p* = tp-rate *tn/n* = 1- fp-rate

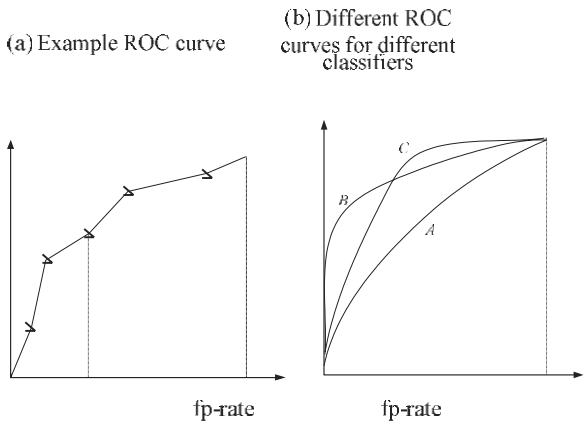
have a true negative, and we have a false positive if we predict a negative example as positive.

In some two-class problems, we make a distinction between the two classes and hence the two type of errors, false positives and false negatives. Different measures appropriate in different settings are given in table 19.2. Let us envisage an authentication application where, for example, users log on to their accounts by voice. A false positive is wrongly logging on an impostor and a false negative is refusing a valid user. It is clear that the two type of errors are not equally bad; the former is much worse. True positive rate, tp-rate, also known as hit rate, measures what proportion of valid users we authenticate and false positive rate, fp-rate, also known as false alarm rate, is the proportion of impostors we wrongly accept.

Let us say the system returns $P(C \hat{} 1|x)$, the probability of the positive class, and for the negative class, we have $P(C \hat{} 2|x) = 1 - P(C \hat{} 1|x)$, and we choose "positive" if $P(C \hat{} 1|x) > \theta$. If θ is close to 1, we hardly choose the positive class; that is, we will have no false positives but also few true positives. As we decrease θ to increase the number of true positives, we risk introducing false positives.

For different values of θ, we can get a number of pairs of (tp-rate, fp-rate) values and by connecting them we get the receiver operating characteristics characteristics (ROC) curve, as shown in figure 19.3a. Note that different values of θ correspond to different loss matrices for the two types of error and the ROC curve can also be seen as the behavior of a classifier

Figure 19.3 (a) Typical ROC curve. Each classifier has a threshold that allows us to move over this curve, and we decide on a point, based on the relative importance of hits versus false alarms, namely, true positives and false positives. The area below the ROC curve is called AUC. (b) A classifier is preferred if its ROC curve is closer to the upper-left corner (larger AUC). B and C are preferred over A; B and C are preferred under different loss matrices

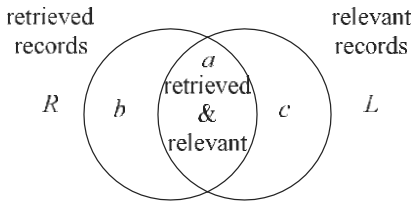


Ideally, a classifier has a tp-rate of 1 and a fp-rate of 0, and hence a classifier is better the more it gets closer to the upper-left corner. On the diagonal, we make as many true decisions as false ones, and this is the worst one can do (any classifier that is below the diagonal can be improved by flipping its decision). Given two classifiers, we can say one is better than the other one if it is above the other one; if two ROC curves intersect, we can say that the two classifiers are better under

different loss conditions, as seen in figure 19.3b.

ROC allows a visual analysis; if we want to reduce the curve to a single the number we can do this by calculating the area under the curve (AUC).A classifier ideally has an AUC of 1 and AUC values of different classifiers can be compared to give us a general performance averaged over different loss conditions.

In information retrieval, there is a database of records; we make a



Precision = a/a+b
Recall = a/a+c

(a) Precision and recall

Figure 19.4 (a) Definition of precision and recall using Venn diagrams. (b) Precision is 1; all the retrieved records are relevant but there may be relevant ones not retrieved. (c) Recall is 1; all the relevant records are retrieved but there may also be irrelevant records that are retrieved.

Precision=1 Recall=1

query, for example, by using some keywords, and a system (basically a two-class classifier) returns a number of records. In the database, there are relevant records and for a query, the system may retrieve some of them (true positives) but probably not all (false negatives); it may also wrongly retrieve records that are not relevant (false positives). The set of relevant and retrieved records can be visualized using a Venn diagram, as shown in figure 19.4a. Precision is the number of retrieved and relevant records divided by the total number of retrieved records; if precision is 1, all the retrieved records may be relevant but there may still be records that are relevant but not retrieved. Recall is the number of retrieved relevant records divided by the total number of relevant records; even if recall is 1, all the relevant records may be retrieved but there may also be irrelevant records that are retrieved, as shown in figure19.4c. As in the ROC curve, for different threshold values, one can draw a curve for precision vs. recall.

From another perspective but with the same aim, there are the two measures of sensitivity and specificity. Sensitivity is the same as tp-rate and recall. Specificity is how

well we detect the negatives, which is the number of true negatives divided by the total number of negatives; this is equal to 1 minus the false alarm rate. One can also draw a sensitivity vs. specificity curve using different thresholds.

In the case of $K > 2$ classes, if we are using 0/1 error, the class confumatrix sion matrix is a $K \times K$ matrix whose entry (i, j) contains the number of instances that belong to C_i but are assigned to C_j . Ideally, all off-diagonals should be 0, for no misclassification. The class confusion matrix allows us to pinpoint what types of misclassification occur, namely, if there are two classes that are frequently confused. Or, one can define K separate two-class problems, each one separating one class from the other $K - 1$.

5.4 Assessing a Classification Algorithm's Performance

We will discuss the case of classification error, but the same methodology applies for squared error in regression, log likelihoods in unsupervised learning, expected reward in reinforcement learning, and so on, as long as we can write the appropriate parametric form for the sampling distribution. We will also discuss nonparametric tests when no such parametric form can be found.

Binomial Test

Let us start with the case where we have a single training set T and a single validation set V . We train our classifier on T and test it on V . We denote by p the probability that the classifier makes a misclassification error. We do not know p; it is what we would like to estimate or test a hypothesis about. On the instance with index t from the validation set V , let us say x^t denotes the correctness of the classifier's decision: x^t is a 0/1 Bernoulli random variable that takes the value 1 when the classifier commits an error and 0 when the classifier is correct. The binomial random variable X denotes the total number of errors:

$$X = \sum_{t=1}^{N} x^t$$

We would like to test whether the error probability p is less than or equal to some value p_0 we specify:

$$H0 : p \leq p0 \text{ vs. } H1 : p > p0$$

If the probability of error is p, the probability that the classifier commits j errors out of N is

$$P\{X = j\} = \binom{N}{j} p^j (1 - p)^{N-j}$$

It is reasonable to reject $p \leq p0$ if in such a case, the probability that binomial test we see $X = e$ errors or more is very unlikely. That is, the binomial test rejects the hypothesis if

$$P\{X \geq e\} = \sum_{x=e}^{N} \binom{N}{x} p_0^x (1-p_0)^{N-x} < \alpha \quad (19.10)$$

where α is the significance, for example, 0.05.

Approximate Normal Test

If p is the probability of error, our point estimate is $\hat{p} = X/N$. Then, it is reasonable to reject the null hypothesis if $\hat{p}$ is much larger than p_0. How large is large enough is given by the sampling distribution of $\hat{p}$ and the significance α.

Because X is the sum of independent random variables from the same distribution, the central limit theorem states that for large N, X/N is approximately normal with mean p_0 and variance $p_0(1 - p_0)$. Then

$$\frac{X/N - p_0}{\sqrt{p0(1 - p0)}} \sim Z \quad (19.11)$$

where $\sim$ denotes "approximately distributed." Then, using equation 19.7, the

approximate normal test rejects the null hypothesis if this value for X = e is greater than $z\alpha$. $Z_{0.05}$ is 1.64. This approximation will work well as long as N is not too small and p is not very close to 0 or 1; as a rule of thumb, we require $Np \geq 5$ and $N(1 - p) \geq 5$.

T Test

The two tests we discussed earlier use a single validation set. If we run the algorithm K times, on K training/validation set pairs, we get K error percentages, pi, i = 1,...,K on the K validation sets. Let x^t_i be 1 if the classifier trained on T_i makes a misclassification error on instance t of Vi; x^t_i is 0 otherwise. Then

$p_i = \sum_{t=1}^{N} x^t_i / N$ Given that $m = \sum_{i=1}^{K} p^i / K$, $S2 = \sum_{i=1}^{K} (p_i - m)^2 / K - 1$

from equation 19.8, we know that we have

$\sqrt{K}(m - p_0)/ S \sim t_{K-1}$ (19.12)

and the t test rejects the null hypothesis that the classification algorithm has p_0 or less error percentage at significance level α if this value is greater than $t_{\alpha,K-1}$. Typically, K is taken as 10 or 30. $T_{0.05,9}$= 1.83 and $t_{0.05,29}$ = 1.70.

5.5 Comparing Two Classification Algorithms

Given two learning algorithms, we want to compare and test whether they construct classifiers that have the same expected error rate.

McNemar's Test

Given a training set and a validation set, we use two algorithms to train two classifiers on the training set and test them on the validation set and compute their errors. A contingency table, like the one shown here, is an array of natural numbers in matrix form representing counts, or frequencies:

e_{00}: Number of examples misclassified by both	e_{01}: Number of examples misclassified by 1 but not 2
e_{10}: Number of examples misclassified by 2 but not 1	e_{11}: Number of examples correctly classified by both

Under the null hypothesis that the classification algorithms have the same error rate, we expect $e_{01} = e_{10}$ and these to be equal to $(e_{01}+e_{10})/2$. We have the chi-square statistic with one degree of freedom

$((|e_{01} - e_{10}| - 1)^2 / e_{01} + e_{10}) \sim X^2_1$

and McNemar's test rejects the hypothesis that the two classification algorithms have the same error rate at significance level α if this value is greater than $X^2_{\alpha,1}$. For $\alpha = 0.05$, $X^2_{0.05,1} = 3.84$

K-Fold Cross-Validated Paired t Test

This set uses K-fold cross-validation to get K training/validation set pairs. We use the two classification algorithms to train on the training sets Ti, i = 1,...,K, and test on the validation sets Vi. The error percentages of the classifiers on the validation sets are recorded as p^1_i and p^2_i .

If the two classification algorithms have the same error rate, then we expect them to have the same mean, or equivalently, that the difference of their means is 0. The

difference in error rates on fold i is $p_i = p^1_i - p^2_i$. This is a paired test; that is, for each i, both algorithms see the same training and validation sets. When this is done K times, we have a distribution of p_i containing K points. Given that p^1_i and p^2_i are both (approximately) normal, their difference p_i is also normal. The null hypothesis is that this distribution has 0 mean:

$$H0 : \mu = 0 \text{ vs. } H1 : \mu \neq 0$$

We define

$$m = \sum_{i=1}^{K} p^i / K, \quad S2 = \sum_{i=1}^{K} (p_i - m)^2 / K - 1$$

Under the null hypothesis that $\mu = 0$, we have a statistic that is tdistributed with K − 1 degrees of freedom:

$$\sqrt{K}(m - 0)/ S = \sqrt{K} \cdot m/ S \sim t_{K-1} \quad (19.14)$$

Thus the K-fold cv paired t test rejects the hypothesis that two clastest sification algorithms have the same error rate at significance level α if this value is outside the interval $(-t_{\alpha/2,K-1}, t_{\alpha/2,K-1})$. $T_{0.025,9} = 2.26$ and $t_{0.025,29} = 2.05$.

If we want to test whether the first algorithm has less error than the second, we need a one-sided hypothesis and use a one-tailed test:

$$H0 : \mu \geq 0 \text{ vs. } H1 : \mu < 0$$

If the test rejects, our claim that the first one has significantly less error is supported.

5 × 2 cv Paired t Test

In the 5 × 2 cv t test, proposed by Dietterich (1998), we perform five replications of twofold cross-validation. In each replication, the dataset is divided into two equal-sized sets. $P^{(j)}_i$ is the difference between the error rates of the two classifiers on fold j = 1, 2 of replication i = 1,..., 5. The average on replication i is $p^-_i = (p^{(1)}_i + p^{(2)}_i)/2$, and the estimated variance is $s^2_i = (p^{(1)}_i - p^-_i)^2 + (p^{(2)}_i - p^-_i)^2$.

Under the null hypothesis that the two classification algorithms have the same error rate, $p^{(j)}_i$ is the difference of two identically distributed proportions, and ignoring the fact that these proportions are not independent, $p^{(j)}_i$ can be treated as approximately normal distributed with 0 mean and unknown variance σ^2. Then $p^{(j)}_i / \sigma$ is approximately unit normal. If we assume $p^{(1)}_i$ and $p^{(2)}_i$ are independent normals (which is not strictly true because their training and test sets are not drawn independently of each other), then $s^2_i / \sigma + 2$ has a chi-square distribution with one degree of freedom. If each of the
s^2_i are assumed to be independent (which is not true because they are all computed from the same set of available data), then their sum is chi-square with five degrees of freedom:

$$M = \sum_{i=1}^{5} s^2_i / \sigma^2 \sim X^2_5$$

and

$$t = p^{(1)}_1 / \sigma / \sqrt{M/5} = p^{(1)}_1 / \sqrt{\sum_{i=1}^{5} s^2_i / 5} \sim t_5 \quad (19.15)$$

giving us a t statistic with five degrees of freedom. The 5 × 2 cv paired t test rejects the hypothesis that the two classification algorithms have the same error rate at significance level α if this value is outside the interval $(-t_{\alpha/2,5}, t_{\alpha/2,5})$. $T_{0.025,5} = 2.57$.

5 × 2 cv Paired F Test

We note that the numerator in equation 19.15, p(1) 1 , is arbitrary; actually, ten different values can be placed in the numerator, namely, $p^{(j)}_i$, j = 1, 2, i = 1,..., 5, leading to ten possible statistics:

$$t^{(j)}_i = p^{(j)}_i / \sqrt{\sum_{i=1}^{5} s^2_i / 5} \quad (19.16)$$

Alpaydın (1999) proposed an extension to the 5 × 2 cv t test that combines the results of

the ten possible statistics. If $p^{(j)}_i / \sigma \sim Z$, then $(p^{(j)}_i)^2 / \sigma 2 \sim X^2_1$ and their sum is chi-square with ten degrees of freedom:

$$N = \sum^5_{i=1} \sum^2_{j=1} (p^{(j)}_i)^2 / \sigma^2 \sim X^2_{10}$$

Placing this in the numerator of equation 19.15, we get a statistic that is the ratio of two chi-square distributed random variables. Two such variables divided by their respective degrees of freedom is F-distributed with ten and five degrees of freedom (section A.3.8):

$$f = (N/10)/(M/5) = \sum^5_{i=1} \sum^2_{j=1} (p^{(j)}_i)^2 / 2 \sum^5_{i=1} s^2_i \sim F_{10,5}$$

5 × 2 cv paired F test rejects the hypothesis that the classification algotest rithms have the same error rate at significance level α if this value is greater than $F_{\alpha,10.5}$. $F_{0.05,10.5} = 4.74$.