ACCURACY AND ERROR MEASURES

Introduction

- After trained a classifier or predictor, there may be many questions going through our mind.

- For example, suppose you used data from previous sales to train a classifier to predict customer purchasing behaviour.

- We would like an estimate of how accurately the classifier can predict the purchasing behaviour of future customers, that is, future customer data on which the classifier has not been trained.

- We may even have tried different methods to build more than one classifier (or predictor) and now wish to compare their accuracy.

Cont…

- But what is accuracy?
- How can we estimate it?
- Are there strategies for increasing the accuracy of a learned model?

A) Classifier Accuracy Measures

- The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier.

- In the pattern recognition literature, this is also referred to as the overall recognition rate of the classifier, that is, it reflects how well the classifier recognizes tuples of the various classes.

Cont…

- We can also speak of the error rate or misclassification rate of a classifier,M, which is simply 1-*Acc(M)*, where *Acc(M)* is the accuracy of *M*.

- If we were to use the training set to estimate the error rate of a model, this quantity is known as the resubstitution error.

- This error estimate is optimistic of the true error rate (and similarly, the corresponding accuracy estimate is optimistic) because the model is not tested on any samples that it has not already seen.

Cont...

- The confusion matrix is a useful tool for analyzing how well your classifier can recognize tuples of different classes.
- Given m classes, a confusion matrix is a table of at least size *m* by *m*. An entry, *CMi, j* in the first *m* rows and *m* columns indicates the number of tuples of class i that were labeled by the classifier as class j.
- For a classifier to have good accuracy, ideally most of the tuples would be represented along the diagonal of the confusion matrix, from entry *CM1, 1* to entry *CMm, m*, with the rest of the entries being close to zero.
- The table may have additional rows or columns to provide totals or recognition rates per class.

Cont…

- Given two classes, we can talk in terms of positive tuples (tuples of the main class of interest, e.g., buys computer = yes) versus negative tuples (e.g., buys computer = no).

- True positives refer to the positive tuples that were correctly labeled by the classifier,

- while true negatives are the negative tuples that were correctly labeled by the classifier.

Cont…

- False positives are the negative tuples that were incorrectly labeled (e.g., tuples of class buys computer = no for which the classifier predicted buys computer = yes).

- False negatives are the positive tuples that were incorrectly labeled (e.g., tuples of class buys computer = yes for which the classifier predicted buys computer = no).

		Predicted class	
		C_1	C_2
Actual class	C_1	true positives	false negatives
	C_2	false positives	true negatives

Cont…

- "Are there alternatives to the accuracy measure?" Suppose that you have trained a classifier to classify medical data tuples as either "cancer" or "not cancer."

- An accuracy rate of, say, 90% may make the classifier seem quite accurate, but what if only, say, 3–4% of the training tuples are actually "cancer"?

- Clearly, an accuracy rate of 90% may not be acceptable—the classifier could be correctly labelling only the "not cancer" tuples,

- The sensitivity and specificity measures can be used, respectively, for this purpose.

- Sensitivity is also referred to as the true positive (recognition) rate (that is, the proportion of positive tuples that are correctly identified), while specificity is the true negative rate (that is, the proportion of negative tuples that are correctly identified).

- In addition, we may use precision to access the percentage of tuples labeled as "cancer" that actually are "cancer" tuples.

Cont...

These measures are defined as:-

$$sensitivity = \frac{t\_pos}{pos}$$

$$specificity = \frac{t\_neg}{neg}$$

$$precision = \frac{t\_pos}{(t\_pos + f\_pos)}$$

Cont...

- where *t pos* is the number of true positives ("cancer" tuples that were correctly classified as such), *pos* is the number of positive ("cancer") tuples, *t neg* is the number of true negatives ("not cancer" tuples that were correctly classified as such), *neg* is the number of negative ("not cancer") tuples, and *f pos* is the number of false positives ("not cancer" tuples that were incorrectly labeled as "cancer").

- It can be shown that accuracy is a function of sensitivity and specificity:

$$accuracy = sensitivity\frac{pos}{(pos+neg)} + specificity\frac{neg}{(pos+neg)}.$$

B) Evaluating the Accuracy of a Classifier or Predictor

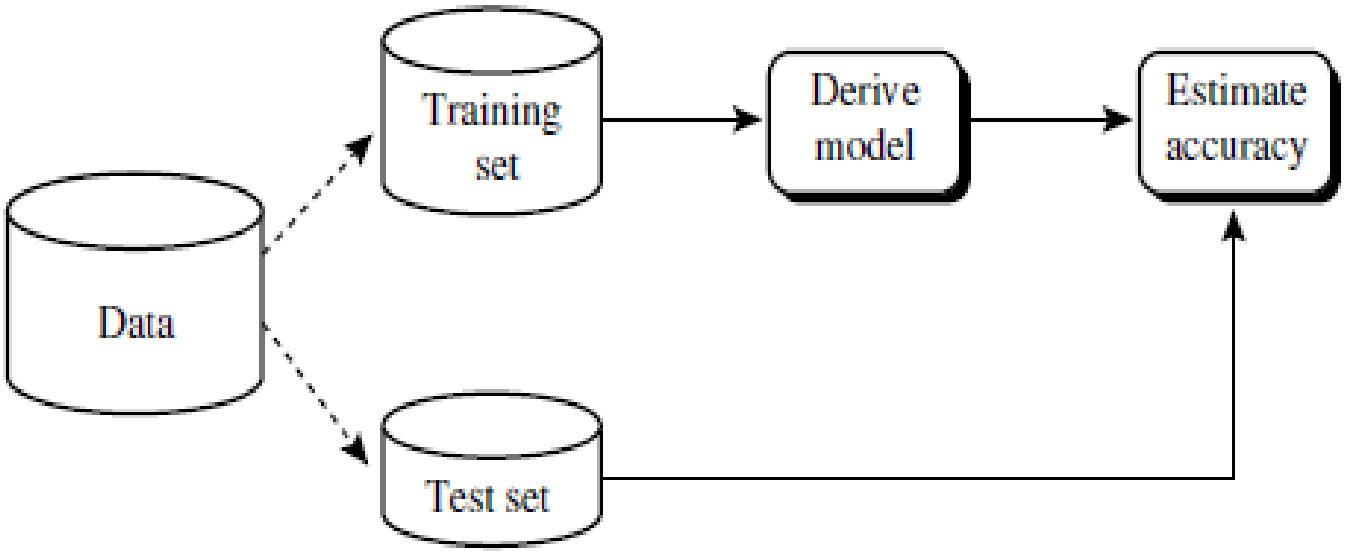


Fig.1: Estimating accuracy with the holdout method.

I)Holdout Method and Random Subsampling

- In this method, the given data are randomly partitioned into two independent sets, a training set and a test set.

- Typically, two-thirds of the data are allocated to the training set, and the remaining one-third is allocated to the test set.

- The training set is used to derive the model, whose accuracy is estimated with the test set.
- Random subsampling is a variation of the holdout method in which the holdout method is repeated k times.
- The overall accuracy estimate is taken as the average of the accuracies obtained from each iteration.

II) Cross-validation

- In *k-fold cross-validation*, the initial data are randomly partitioned into *k* mutually exclusive subsets or "folds," D1, D2, ..., Dk, each of approximately equal size.

- Training and testing is performed *k* times. In iteration *i*, partition *Di* is reserved as the test set, and the remaining partitions are collectively used to train the model.

- That is, in the first iteration, subsets D2,....., Dk collectively serve as the training set in order to obtain a first model, which is tested on D1; the second iteration is trained on subsets D1, D3, ..., Dk and tested on D2; and so on.

Cont...

- Unlike the holdout and random subsampling methods above, here, each sample is used the same number of times for training and once for testing.

- For classification, the accuracy estimate is the overall number of correct classifications from the k iterations, divided by the total number of tuples in the initial data.

- For prediction, the error estimate can be computed as the total loss from the k iterations, divided by the total number of initial tuples.

Cont...

- Leave-one-out is a special case of k-fold cross-validation where k is set to the number of initial tuples.
- That is, only one sample is "left out" at a time for the test set.
- In stratified cross-validation, the folds are stratified so that the class distribution of the tuples in each fold is approximately the same as that in the initial data.
- In general, stratified 10-fold cross-validation is recommended for estimating accuracy.

III) Bootstrap

- The bootstrap method samples the given training tuples uniformly with replacement.
- That is, each time a tuple is selected, it is equally likely to be selected again and readded to the training set.
- In sampling with replacement, the machine is allowed to select the same tuple more than once.
- There are several bootstrap methods. A commonly used one is the .632 bootstrap.

Cont...

- which works as follows. Suppose we are given a data set of d tuples. The data set is sampled *d* times, with replacement, resulting in a bootstrap sample or training set of *d* samples.

- It is very likely that some of the original data tuples will occur more than once in this sample.

- The data tuples that did not make it into the training set end up forming the test set. Suppose we were to try this out several times. As it turns out, on average, 63.2% of the original data tuples will end up in the bootstrap, and the remaining 36.8% will form the test set (hence, the name, .632 bootstrap.)

Cont...

- "Where does the figure, 63.2%, come from?" Each tuple has a probability of 1/d of being selected, so the probability of not being chosen is (1-1/d).
- We have to select d times, so the probability that a tuple will not be chosen during this whole time is $(1-1/d)^d$
- If d is large, the probability approaches $e^{-1} = 0.368$
- Thus, 36.8% of tuples will not be selected for training and thereby end up in the test set, and the remaining 63.2% will form the training set.

Cont...

- We can repeat the sampling procedure k times, where in each iteration, we use the current test set to obtain an accuracy estimate of the model obtained from the current bootstrap sample.

- The overall accuracy of the model is then estimated as:-

$$Acc(M) = \sum_{i=1}^{k} \left(0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set}\right),$$

Cont...

- where $Acc(Mi)_{\text{test set}}$ is the accuracy of the model obtained with bootstrap sample i when it is applied to test set i.

- $Acc(Mi)_{train\ set}$ is the accuracy of the model obtained with bootstrap sample i when it is applied to the original set of data tuples.

- The bootstrap method works well with small data sets.

C) Ensemble Methods—Increasing the Accuracy

- Are there general strategies for improving classifier and predictor accuracy?
- The answer is yes. Bagging and boosting are two such techniques .
- They are examples of ensemble methods, or methods that use a combination of models.
- Each combines a series of k learned models (classifiers or predictors), *M1, M2, ..., Mk*, with the aim of creating an improved composite model, M^*.
- Both bagging and boosting can be used for classification as well as prediction.

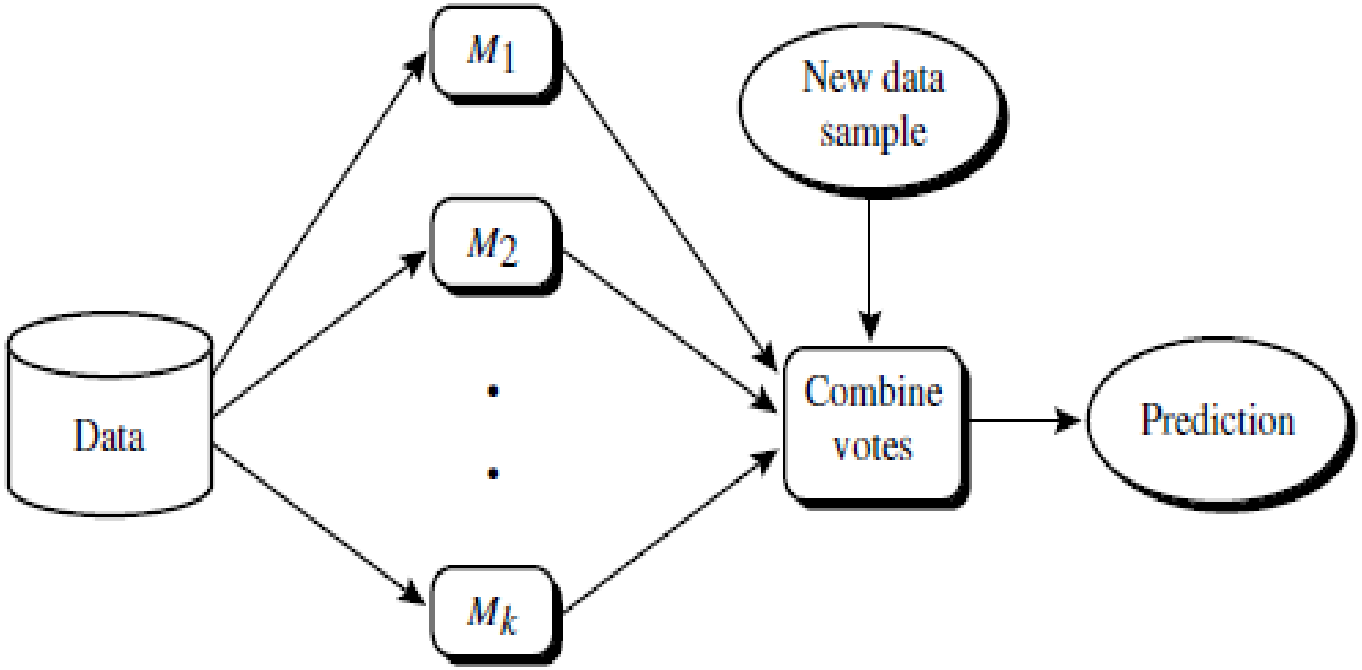
I) Bagging

- We first take an intuitive look at how bagging works as a method of increasing accuracy.
- For ease of explanation, we will assume at first that our model is a classifier.
- Suppose that you are a patient and would like to have a diagnosis made based on your symptoms.
- Instead of asking one doctor, you may choose to ask several. If a certain diagnosis occurs more than any of the others, you may choose this as the final or best diagnosis.

Cont...

- That is, the final diagnosis is made based on a majority vote, where each doctor gets an equal vote.
- Now replace each doctor by a classifier, and you have the basic idea behind bagging.
- Intuitively, a majority vote made by a large group of doctors may be more reliable than a majority vote made by a small group.

Cont…



Increasing model accuracy: Bagging and boosting each generate a set of classification or prediction models, *M1, M2, ..., Mk. Voting strategies are used to combine the prediction* for a given unknown tuple.

Cont...

- Given a set, *D*, of *d* tuples, bagging works as follows. For iteration i (i = 1, 2, ……., *k*), a training set,*Di*, of *d* tuples is sampled with replacement fromthe original set of tuples,*D*.

- sampling with replacement is used, some of the original tuples of *D* may not be included in *Di*, whereas others may occur more than once.

- A classifier model, Mi, is learned for each training set, Di.

- To classify an unknown tuple, X, each classifier, Mi, returns its class prediction, which counts as one vote.

Cont...

- The bagged classifier, M, counts the votes and assigns the class with the most votes to X.
- Bagging can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple.
- The bagged classifier often has significantly greater accuracy than a single classifier derived from D, the original training data.

Cont...

- It will not be considerably worse and is more robust to the effects of noisy data.
- The increased accuracy occurs because the composite model reduces the variance of the individual classifiers.
- For prediction, it was theoretically proven that a bagged predictor will always have improved accuracy over a single predictor derived from D.

Cont...

- Algorithm: Bagging. The bagging algorithm—create an ensemble of models (classifiers or predictors) for a learning scheme where each model gives an equally-weighted prediction.

Input:

- D, a set of *d* training tuples;
- *k*, the number of models in the ensemble;
- a learning scheme (e.g., decision tree algorithm, backpropagation, etc.)

Output: A composite model, M.

Cont...

- Method:

(1) for i = 1 to k do // create k models:

(2) create bootstrap sample, Di, by sampling D with replacement;

(3) use Di to derive a model, Mi;

(4) endfor

To use the composite model on a tuple, X:

(1) if classification then

(2) let each of the k models classify X and return the majority vote;

(3) if prediction then

(4) let each of the k models predict a value for X and return the average predicted value;

II) Boosting

- As in the previous section, suppose that as a patient, we have certain symptoms.
- Instead of consulting one doctor, we choose to consult several.
- Suppose we assign weights to the value or worth of each doctor's diagnosis, based on the accuracies of previous diagnoses they have made.
- The final diagnosis is then a combination of the weighted diagnoses.
- This is the essence behind boosting.

Cont...

- In boosting, weights are assigned to each training tuple.
- A series of k classifiers is iteratively learned.
- After a classifier Mi is learned, the weights are updated to allow the subsequent classifier,Mi+1, to "pay more attention" to the training tuples that were misclassified by Mi.
- The final boosted classifier, M, combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy. The boosting algorithm can be extended for the prediction of continuous values.

Cont...

- Adaboost is apopularboosting algorithm. Supposewewould like toboost the accuracy of some learning method.
- We are given D, a data set of d class-labeled tuples, (X1, y1), (X2, y2), ...,(Xd, yd), where yi is the class label of tupleXi.
- Initially, Adaboost assigns each training tuple an equal weight of 1/d.
- Generating *k* classifiers for the ensemble requires *k* rounds through the rest of the algorithm.
- In round i, the tuples from D are sampled to forma training set,*Di*, of size *d*.
- Sampling with replacement is used—the same tuple may be selected more than once.

Cont...

- Each tuple's chance of being selected is based on its weight.
- A classifier model, *Mi*, is derived from the training tuples of *Di*. Its error is then calculated using *Di* as a test set.
- The weights of the training tuples are then adjusted according to how they were classified.
- If a tuple was incorrectly classified, its weight is increased.
- If a tuple was correctly classified, its weight is decreased.

Cont...

- These weights will be used to generate the training samples for the classifier of the next round.
- The basic idea is that when we build a classifier, we want it to focus more on the misclassified tuples of the previous round.
- To compute the error rate of model *Mi*, we sum the weights of each of the tuples in *Di* that *Mi* misclassified.

$$error(M_i) = \sum_{j}^{d} w_j \times err(X_j),$$

Cont...

- where *err(Xj)* is the misclassification error of tuple *Xj*: If the tuple was misclassified, then err(Xj) is 1. Otherwise, it is 0.

- If the performance of classifier Mi is so poor that its error exceeds 0.5, then we abandon it. Instead, we try again by generating a new Di training set, from which we derive a new Mi.

Cont...

- The error rate of Mi affects howtheweights of the training tuples are updated.

- If a tuple in round i was correctly classified, its weight is multiplied by *error(Mi)/(1-error(Mi))*.

- Once the weights of all of the correctly classified tuples are updated, the weights for all tuples (including the misclassified ones) are normalized so that their sum remains the same as it was before.

Cont...

- To normalize a weight, we multiply it by the sum of the old weights, divided by the sum of the new weights.

- As a result, the weights of misclassified tuples are increased and the weights of correctly classified tuples are decreased, as described above.

- "Once boosting is complete, how is the ensemble of classifiers used to predict the class label of a tuple, X?" Unlike bagging, where each classifier was assigned an equal vote, boosting assigns a weight to each classifier's vote, based on how well the classifier performed.

Cont...

- The lower a classifier's error rate, the more accurate it is, and therefore, the higher its weight for voting should be.
- The weight of classifier Mi's vote is For each class, c, we sum the weights of each classifier that assigned class c to X.
- The class with the highest sum is the "winner" and is returned as the class prediction for tuple X.

$$log \frac{1 - error(M_i)}{error(M_i)}$$

Cont...

- Algorithm: Adaboost. A boosting algorithm—create an ensemble of classifiers. Each one gives a weighted vote.

Input:

- D, a set of *d* class-labeled training tuples;
- *k*, the number of rounds (one classifier is generated per round);
- a classification learning scheme.

Output: A composite model.

Cont...

Method:

(1) initialize the weight of each tuple in D to 1=d;

(2) for i = 1 to k do // for each round:

(3) sample D with replacement according to the tuple weights to obtain Di;

(4) use training set Di to derive a model, Mi;

(5) compute error(Mi), the error rate of Mi

(6) if error(Mi) > 0:5 then

Cont...

(7) reinitialize the weights to 1=d

(8) go back to step 3 and try again;

(9) endif

(10) for each tuple in Di that was correctly classified do

(11) multiply the weight of the tuple by error(Mi)=(1-error(Mi)); // update weights

(12) normalize the weight of each tuple;

(13) endfor

Cont...

To use the composite model to classify tuple, X:

(1) initialize weight of each class to 0;

(2) for i = 1 to k do // for each classifier:

(3) wi = log 1-error(Mi)/error(Mi) ; // weight of the classifier's vote

(4) c = Mi(X); // get class prediction for X from Mi

(5) add wi to weight for class c

(6) endfor

(7) return the class with the largest weight;

Cont...

- "How does boosting compare with bagging?" Because of the way boosting focuses on the misclassified tuples, it risks overfitting the resulting composite model to such data.

- Therefore, sometimes the resulting "boosted" model may be less accurate than a single model derived from the same data. Bagging is less susceptible to model overfitting.

- While both can significantly improve accuracy in comparison to a single model, boosting tends to achieve greater accuracy.

Thank You