Data Mining: Accuracy and Error Measures for Classification and Prediction

Chapter · January 2018

DOI: 10.1016/B978-0-12-809633-8.20474-3

2 authors:

Paola Galdi
The University of Edinburgh
13 PUBLICATIONS **14** CITATIONS
SEE PROFILE

Roberto Tagliaferri
Università degli Studi di Salerno
247 PUBLICATIONS **1,951** CITATIONS
SEE PROFILE

Data Mining: Accuracy and Error Measures for Classification and Prediction

Paola Galdi, Roberto Tagliaferri

NeuRoNe Lab, DISA-MIS, University of Salerno, via Giovanni Paolo II 132, 84084, Fisciano (SA), Italy

Abstract

A variety of measures exist to assess the accuracy of predictive models in data mining and several aspects should be considered when evaluating the performance of learning algorithms. In this chapter, the most common accuracy and error scores for classification and regression are reviewed and compared. Moreover, the standard approaches to model selection and assessment are presented, together with an introduction to ensemble methods for improving the accuracy of single classifiers.

Keywords: accuracy estimation, assessment, bagging, boosting, bootstrap, cross-validation, holdout method, model selection, sensitivity, specificity

1. Introduction

The goal of data mining is that of building learning models to automatically extract knowledge from big amounts of complex data. In supervised learning, prior information is used to train a model to learn latent relationships between data objects. Once the model is trained, it is used to perform predictions on previously unseen data.

According to the type of prediction, we can distinguish between classification, where the output is a categorical class label, and regression, where the model learns a continuous function. In the former task, data are divided into groups according to some discriminative features. In the latter, input features are put into a functional relationship with a variable of interest. In both cases, a method is needed to quantify the performance of a model, i.e. to determine how accurate are its predictions.

During the training phase, measuring accuracy plays a relevant role in model selection: parameters are selected in order to maximize prediction accuracy on training samples. At the end of the learning step, accuracy is measured to assess the model predictive ability on new data. Being learning algorithms trained on finite samples, the risk is to overfit training data: the model might memorize the training samples instead of learning a general rule, i.e. the data generating model. For this reason, a high accuracy on unseen data is an index of the model generalization ability.

The remainder of the chapter is organized as follows: in section 2 and 3 the main accuracy measures used in classification and regression tasks are presented, respectively. Section 4 explores how accuracy measures can be used in combination with validation techniques for model selection and model assessment. Section 5 introduces bagging and boosting, two techniques for improving the prediction accuracy of classification models.

2. Accuracy Measures in Classification

The accuracy of a classifier is the probability of correctly predicting the class of an unlabelled instance and it can be estimated in several ways (Baldi et al., 2000). Let us assume for simplicity to have a two-class problem: as an example, consider the case of a diagnostic test to discriminate between subjects affected by a disease (patients) and healthy subjects (controls). Accuracy measures for binary classification can be described in terms of four values:

- TP or true positives, the number of correctly classified patients.
- TN or true negatives, the number of correctly classified controls.
- FP or false positives, the number of controls classified as patients.
- FN or false negatives, the number of patients classified as controls.

The sum of TP, TN, FP and FN equals N, the number of instances to classify. These values can be arranged in a 2×2 matrix called contingency matrix, where we have the actual classes P and C on the rows, and the predicted classes $\tilde{P}$ and $\tilde{C}$ on the columns.

	$\tilde{P}$	$\tilde{C}$
P	TP	FN
C	FP	TN

The classifier sensitivity (also known as recall) is defined as the proportion of true positives on the total number of positive instances:

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{1}$$

In the above example, it is the number of correctly diagnosed patients on the total number of subjects affected by the disease.

The classifier specificity (also referred to as precision) is defined as the proportion of true positives on the total number of instances identified as positive[1]:

$$\text{specificity} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2}$$

In the example, it corresponds to the number of correctly diagnosed patients among all the positive diagnoses.

A low sensitivity corresponds to a high number of false negatives, while a low specificity indicates the presence of many false positives (see figure 1 for an example). According to the context, high rates of false negative or false positive predictions might have different implications: consider failing to diagnose a sick subject (false negative) versus assigning a treatment to a healthy subject (false positive).

The simplest way to estimate accuracy is to compute the percentage of correctly classified patients (positive instances):

$$\text{PA}_{pos} = 100 \cdot \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

which is complemented by the percentage of correctly classified controls (negative instances):

$$\text{PA}_{neg} = 100 \cdot \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{4}$$

[1]In some contexts, the term specificity is used to indicate the sensitivity of the negative class.

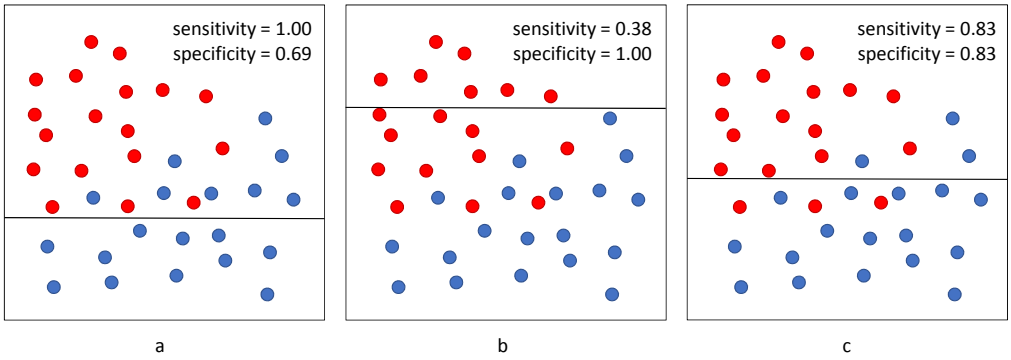

Figure 1: The trade-off between specificity and sensitivity. In the first panel (a) the sensitivity for the red class is highest since all red dots have been assigned to the same class, but the specificity is low since there are many false positives (blue dots in the red class). In the second panel (b) the specificity for the red class is highest since all points assigned to that class are red, but the sensitivity is low for the presence of many false negatives (red dots in the blue class). In the last panel (c) there is a balance between sensitivity and specificity.

Taking into account all the correctly classified instances, accuracy can be computed as:

$$\mathrm{PA} = 100 \cdot \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{5}$$

Another option is to average PA_{pos} and PA_{neg} to obtain the average percentage accuracy on both classes, as follows:

$$\mathrm{PA}_{avg} = \frac{\mathrm{PA}_{pos} + \mathrm{PA}_{neg}}{2}. \tag{6}$$

F-measure (or F-score or F_1-score) has been introduced to balance between sensitivity and specificity. It is defined as the harmonic mean of the two scores, multiplied by 2 to obtain a score of 1 when both sensitivity and specificity equal 1:

$$\mathrm{F} = 2 \cdot \frac{1}{\frac{1}{\text{sensitivity}} + \frac{1}{\text{specificity}}} = 2 \cdot \frac{\text{specificity} \cdot \text{sensitivity}}{\text{specificity} + \text{sensitivity}}. \tag{7}$$

Care must be taken when selecting an accuracy measure in presence of unbalanced classes. Following again the above example, consider the case of a dataset where 90% of the subjects are sick subjects and the remaining

10% consists of healthy controls. Taking into account only the percentage of correctly classified instances, a model which assigns every subject to the patients class would attain an accuracy of 90%. In the same case the F-measure would be equal to 0.95, while the average percentage accuracy on both classes PA_{avg} would yield a more informative score of 45%.

When working with more than two classes, say M, the resulting contingency matrix $X = \{x_{ij}\}$ is a $M \times M$ matrix where each entry x_{ij} is the number of instances belonging to class i that have been assigned to class j, for $i, j = 1, \ldots, M$. The sensitivity for class i can then be computed as:

$$\text{sensitivity}_i = 100 \cdot \frac{x_{ii}}{n_i} \tag{8}$$

where n_i is the number of instances in class i. Similarly, the specificity for class i is:

$$\text{specificity}_i = 100 \cdot \frac{x_{ii}}{p_i} \tag{9}$$

where p_i is the total number of instances predicted to be in class i. The percentage of all correct predictions corresponds to:

$$PA = 100 \cdot \frac{\sum_i x_{ii}}{N}. \tag{10}$$

3. Accuracy Measures in Regression

A regression model is accurate if it predicts for a given input pattern its target value with a low error. Given a vector $\hat{y}$ of N predictions and the vector y of N actual observed target values, the most commonly adopted error measure is the mean squared error, computed as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2. \tag{11}$$

Another variant is the root mean squared error:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}. \tag{12}$$

Other measures are based on the absolute difference between predicted and actual values, like the mean absolute error:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i|. \tag{13}$$

The above measures have the same scale as the input data. A scale-independent alternative is the relative absolute error, computed as:

$$\text{RAE} = \frac{\sum_{i=1}^{N} |\hat{y}_i - y_i|}{\sum_{i=1}^{N} |\bar{y} - y_i|} \tag{14}$$

where $\bar{y}$ is the mean of vector y, and therefore the quantity $\sum_{i=1}^{N} |\bar{y} - y_i|$ acts as a normalizing factor to get a value between 0 and 1. Multiplying this value by 100 yields the percentage absolute error.
The relative squared error is obtained in a similar fashion:

$$\text{RSE} = \frac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{N} (\bar{y} - y_i)^2}. \tag{15}$$

The choice between squared and absolute error depends on several considerations. For example, squared error emphasizes larger differences while absolute error is more robust to outliers causing occasionally large errors. Squared error is sometimes preferred because of its mathematical properties. Contrary to absolute error, it is continuously differentiable, making analytical optimization easier. In addition, when fitting a Gaussian distribution, the maximum likelihood fit is that which minimizes the squared error. On the other hand, absolute error uses the same unit as the data (instead of square units) and might be more straightforward to interpret.

4. Model Selection and Assessment

The standard approach when evaluating the accuracy of a model is the hold out method. It consists in splitting the available data in two disjoint sets: the training set, which is fed into the learning algorithm, and the test set, used only to evaluate the performance of the model. Commonly, two thirds of the data items are destined to the training and the remainder is reserved for the testing, but other proportions like 70:30, 80:20 and 90:10

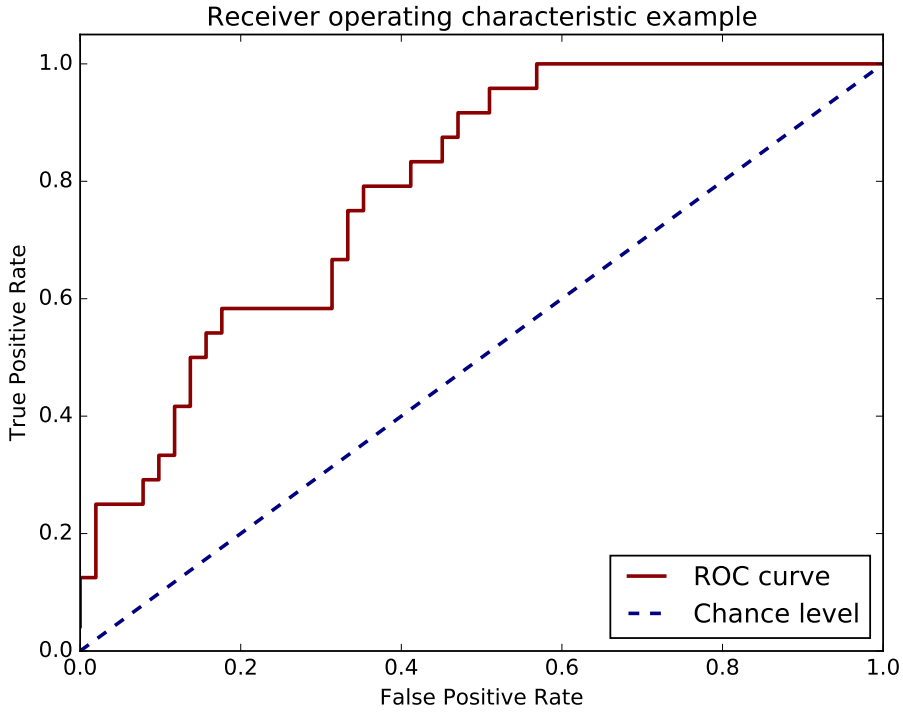

Figure 2: Receiver Operating Characteristic (ROC) curve.

are also adopted. There is a trade-off between the training set size and the test set size: using more data for the training allows to build more accurate models, but with fewer instances in the test set the estimate of prediction accuracy will have a greater variance.

When models depend on one or more parameters, a third set of data, the validation set, is required to perform model selection: several models are trained using only data coming from the training set, with different values for each parameter; from the pool of trained models, one winner is selected as that which achieves the best score according to a given accuracy measure on the validation set; finally, the generalization ability of the model is assessed on the test set. Two separated sets of data are necessary for model selection and assessment because using the validation set for the selection of the final model might introduce bias. Therefore, new unseen data are needed to avoid the underestimation of the true error.

Since the choice of the accuracy measure to optimize greatly affects the selection of the best model, then the proper score should be determined taking into account the goal of the analysis. When performing model selection in a binary classification problem, e.g. when selecting the best threshold for a classifier with a continuous output, a reasonable criterion is to find a compromise between the amount of false positives and the amount of false negatives. The receiver operating characteristic (ROC) curve is a graphical representation of the true positive rate (the sensitivity) as a function of the false positive rate (the so called false alarm rate, computed as $\mathrm{FP}/(\mathrm{FP}+\mathrm{TN})$), as shown in figure 2. A perfect classifier, with a 100% true positive rate and no false positives would be represented by a $(0, 1)$ coordinate in the ROC space, while chance level corresponds to the diagonal of the plot. Therefore, a good classifier would be represented by a point near the upper left corner of the graph and far from the diagonal. An indicator related to the ROC curve is the area under the curve (AUC), that is equal to 1 for a perfect classifier and to 0.5 for a random guess.

The main drawback of the hold out method is that it makes an inefficient use of data using only a part of them for the training, thus providing a pessimistic estimate of the model actual accuracy. For this reason, other approaches have been devised to take advantage of all available data, especially in the case of datasets with few samples. Another limitation of the hold out method is that the outcome depends on the specific choice of training and test set. To address this issue, random subsampling can be used to randomly generate multiple training/test splits and the accuracy is computed as the average accuracy across runs. However, if a class happens to be over-represented in the training set, it will be under-represented in the test set, and vice versa, thus affecting the prediction accuracy estimation. Therefore, stratification can be used to preserve the proportion of classes in both training and test set.

Cross-validation is a technique for accuracy estimation where the original dataset is randomly split into k disjoint sets of the same size. The model to be evaluated is trained k times, selecting in rotation $k-1$ sets for the training and the k-th for the validation. The final accuracy is the overall number of correctly classified instances in each fold divided by N, the total number of instances in the dataset. When $k = N$, the scheme is called leave-one-out, since in each fold the test set is composed by a single object. As for the hold out method, cross-validation can be repeated multiple times using different

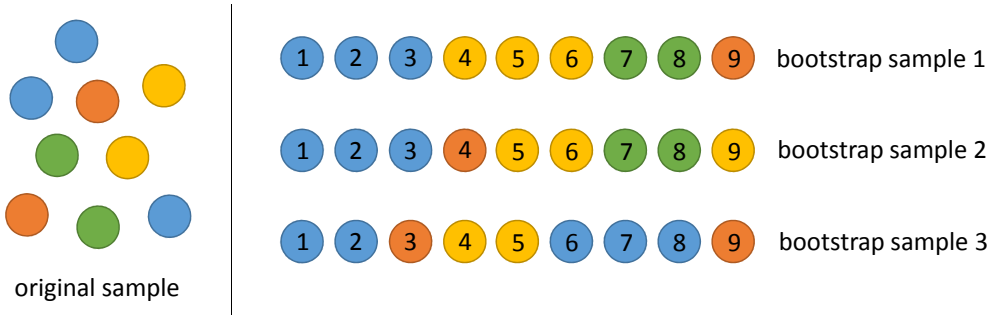

Figure 3: An example of bootstrap sampling. Since objects are subsampled with replacement, some classes might be over-represented (yellow marbles in bootstrap samples 1 and 2), others might be under-represented (red marbles in bootstrap samples 1 and 2) or even missing (green marbles in bootstrap sample 3).

random splits to obtain a better estimation of prediction accuracy[2], and splits can be stratified to preserve class proportions. The advantage of this approach is that it exploits all available data. In particular, the leave-one-out method tests the model on all the possible N splits, and it is suitable in case of few samples, but it is characterized by a high variance. For this reason, small values for k are usually preferred, like in 5- and 10-cross-validation schemes (Kohavi et al., 1995).

Bootstrap (Efron, 1983) is an alternative to cross-validation that relies on random subsampling with replacement (see figure 3 for an example). It is based on the assumption that samples are independently and identically distributed and it is especially recommended when the sample size is small. After n extractions, the probability that a given object has not been extracted yet is equal to $(1 - \frac{1}{n})^n \approx e^{-1} \approx 0.368$, therefore the expected number of distinct instances in the sample is $0.632n$. The .632 bootstrap estimate of accuracy, introduced in Efron and Tibshirani (1997), is defined as:

$$\text{acc}_{boot} = \frac{1}{b}\sum_{i=1}^{b}(0.632 \cdot \text{acc}_i + 0.368 \cdot \text{acc}_{train}) \tag{16}$$

where b is the number of generated bootstrap sample, acc_i is the accuracy obtained with a model trained on sample i and tested on the remaining

[2]In a complete cross-validation scheme, all the $\binom{N}{k}$ possible splits are tested, but this is usually prohibitive in terms of computational costs.

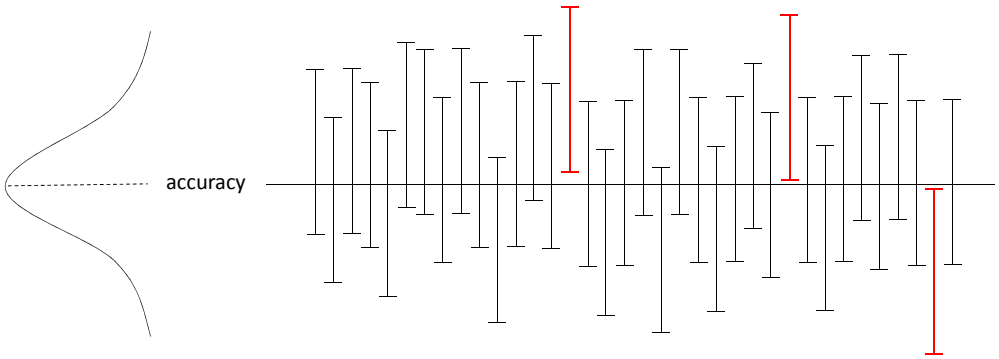

Figure 4: Confidence intervals estimated with 90% confidence level: in 3 out of 30 samples from the same population the confidence intervals do not contain the true value for accuracy.

instances, and acc_{train} is the accuracy of the model trained on the full training set.

Once the accuracy of a model has been estimated on a sample with one of the above methods, the amount of uncertainty associated with it can be described by a confidence interval, a range of values that is likely to contain the true value. Confidence intervals are computed for a given a confidence level expressed as a percentage (commonly adopted values are 90%, 95% and 99%), that is interpreted in the following way: if multiple samples are extracted from the same population and a confidence interval with a confidence level of, say, 90%, is computed for each sample, 90% of intervals will contain the true value (figure 4). The width of a confidence interval is affected by the size of the sample and its variability, i.e. a small sample characterized by a high variability would yield larger confidence intervals.
In binary classification problems, confidence intervals for the accuracy estimate can be calculated assuming a binomial distribution for the number of correct predictions on the test set; in other words, each prediction is modelled as a Bernoulli variable whose two possible values correspond to correct or incorrect prediction. In general, when a sufficient number of samples is available, approximated confidence intervals can be calculated assuming a Gaussian sampling distribution.

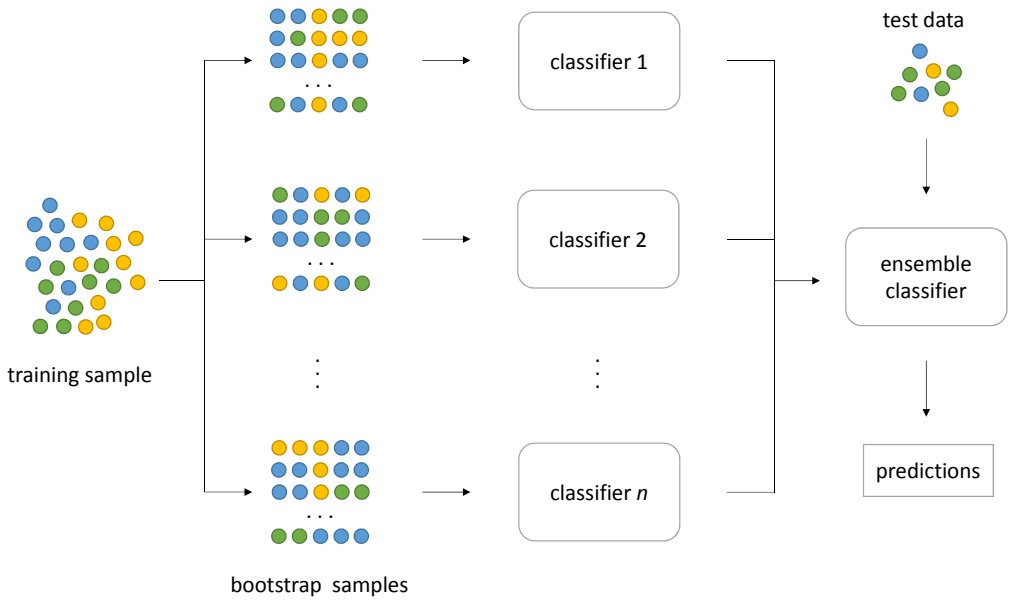

Figure 5: The bagging approach. Several classifier are trained on bootstrap samples of the training data. Predictions on test data are obtained combining the predictions of the trained classifiers with a majority voting scheme.

5. Improving Accuracy

Ensemble methods are a class of algorithms that build models by combining predictions of multiple base classifiers. Two prerequisites for the ensemble of classifiers are that they should be diverse, i.e. predict with different accuracies on new data, and accurate, meaning that they should perform better than a random guess (Dietterich, 2000a). There are several approaches to this problem, the most relevant ones being bagging and boosting. The main idea behind these techniques is to generate many perturbed versions of the original training set on which different classifiers are built. Then, the final classification is obtained with a majority voting scheme, where each classifier expresses its vote for class membership of test instances. Both methods have been shown to outperform prediction accuracies of single classifiers (Quinlan et al., 1996).

Bagging (as depicted in figure 5) works by creating several training sets by subsampling with replacement the input data, as in bootstrap sampling: the new sets have the same size of the original training set but with some

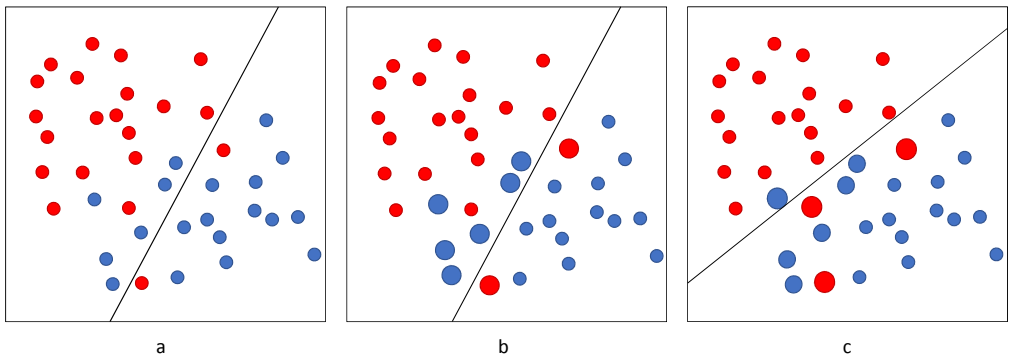

Figure 6: The boosting approach. A classifier is trained on the original data (a). The weights of misclassified instances (dot size in the figure) are increased (b). A new classifier is trained on the new data set and weights are updated accordingly (c).

instances missing and some present more than once. The performance of this method depends on how unstable the base models are: if small perturbations of the training set cause a significant variability in the trained classifiers, then bagging can improve the final accuracy (Breiman, 1996). This approach has been successfully applied in the Random Forest model (Breiman, 2001), where the outputs of an ensemble of Decision Tree classifiers are combined in a final solution.

While in bagging each sample is extracted independently and the different classifiers are trained in parallel, boosting follows an iterative scheme, where at each repetition each object is associated with a weight that reflects the performance of the current classifier. Specifically, weights of misclassified instances are increased while weights of correctly classified instances are decreased (see figure 6). Then, at the beginning of the next iteration, objects are sampled from the original training set with a probability proportional to their weights (Dietterich, 2000b). The goal is to guide the training of new models towards a solution that correctly predicts the instances that were misclassified in the original training set, thus affecting accuracy. AdaBoost is a prominent example of this class of methods (Freund and Schapire, 1995; Freund et al., 1996).

Further Reading

- Efron, B., Tibshirani, R. J., 1994. An introduction to the bootstrap. CRC press.
- Molinaro, A. M., Simon, R., Pfeiffer, R. M., 2005. Prediction error estimation: a comparison of resampling methods. Bioinformatics 21 (15), 33013307.

References

Baldi, P., Brunak, S., Chauvin, Y., Andersen, C. A., Nielsen, H., 2000. Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics 16 (5), 412–424.

Breiman, L., 1996. Bagging predictors. Machine learning 24 (2), 123–140.

Breiman, L., 2001. Random forests. Machine learning 45 (1), 5–32.

Dietterich, T. G., 2000a. Ensemble methods in machine learning. In: International workshop on multiple classifier systems. Springer, pp. 1–15.

Dietterich, T. G., 2000b. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine learning 40 (2), 139–157.

Efron, B., 1983. Estimating the error rate of a prediction rule: improvement on cross-validation. Journal of the American Statistical Association 78 (382), 316–331.

Efron, B., Tibshirani, R., 1997. Improvements on cross-validation: the 632+ bootstrap method. Journal of the American Statistical Association 92 (438), 548–560.

Freund, Y., Schapire, R. E., 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In: European conference on computational learning theory. Springer, pp. 23–37.

Freund, Y., Schapire, R. E., et al., 1996. Experiments with a new boosting algorithm. In: icml. Vol. 96. pp. 148–156.

Kohavi, R., et al., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Ijcai. Vol. 14. Stanford, CA, pp. 1137–1145.

Quinlan, J. R., et al., 1996. Bagging, boosting, and C4.5. In: AAAI/IAAI, Vol. 1. pp. 725–730.