A Posteriori Corrections to Classification Methods.

Włodzisław Duch and Łukasz Itert

Department of Informatics, Nicholas Copernicus University,
Grudziądzka 5, 87-100 Toruń, Poland; http://www.phys.uni.torun.pl/kmk

Abstract. *A posteriori* corrections are computational inexpensive and may improve accuracy, confidence, sensitivity or specificity of the model, or correct for the differences between a priori training and real (test) class distributions. Such corrections are applicable to neural and any other classification models.

1 Introduction

Training neural networks and other classification models may be time consuming, therefore one should try to derive as much information from the final model as possible. In pathological cases neural and other classifiers may lead to results that are below the base rates, i.e. below the values obtained by majority classifier. This may happen especially in case of strong imbalance between the number of samples from different classes. Another problem arises when the data available for training has different *a priori* probabilities than the test data. This problem arises frequently in biostatistics and medical statistics where controlled experiments do not reflect the true class distributions, in use of stratified cross-validation for training (see [2] for more examples). The overall accuracy may be not so important as the sensitivity, specificity or the confidence in the results that should be increased.

A posteriori procedures may frequently correct such problems. Correction increasing overall accuracy, accounting for differences in class distribution and increasing the desired aspects of the classification process are presented in subsequent sections.

2 Corrections increasing overall accuracy

Most classifiers, neural networks included, do not estimated rigorously probabilities. However, normalizing the classifier's outputs in such a way that they sum to unity some estimate of probabilities $p(C_i|\mathbf{X})$ that a vector $\mathbf{X}$ belongs to the class $C_i, i = 1..K$, is obtained. Suppose that the majority of samples belong to the last class C_K. Introducing $K-1$ linear scaling parameters:

$$P(C_i|\mathbf{X}) = \kappa_i p(C_i|\mathbf{X}), i = 1 \ldots K-1; \quad P(C_K|\mathbf{X}) = 1 - \sum_{i=1}^{K-1} P(C_i|\mathbf{X}) \tag{1}$$

it is easy to assure that even a very bad classifier will perform at least at the level of the majority classifier. This is achieved when all $\kappa_i = 0$. If all $\kappa_i = 1$ then nothing will change. If the coefficients $\kappa_i \in [0,1]$ normalization is preserved, but the probability of the majority class may only grow. In some situations it may be of advantage to decrease this probability. For that reason assume that $\kappa_i \in [0,+\infty], i = 1\dots K-1$, and $\kappa_K = 1$, and the final probabilities are obtained from the softmax transformation instead of the linear rescaling:

$$P(C_i|\mathbf{X};\kappa) = \exp\left(\kappa_i p(C_i|\mathbf{X}\right)\Big/\exp\left(\sum_{j=1}^{K}\kappa_j p(C_i|\mathbf{X})\right),\ i = 1\dots K \tag{2}$$

If all $\kappa_i = 1$ the softmax transformation “flattens” probabilities while linear transformation does not change anything. For example, if there are two classes and $p(C_1|\mathbf{X}) + p(C_2|\mathbf{X}) = 1, p(C_i|\mathbf{X}) \in [0,1]$ the softmax transformed probabilities belong to $P(C_1|\mathbf{X}) \in [(1+e^{-1})^{-1}, (1+e^{+1})^{-1}]$, or approximately $[0.27, 0.73]$.

Rescaled probabilities depend on the κ coefficients. For the linear transformation the quadratic cost function may be taken as:

$$E(\kappa) = \sum_{\mathbf{X}}\sum_{i=1}^{K}\left(P(C_i|\mathbf{X};\kappa) - P_i(\mathbf{X})\right)^2 \tag{3}$$

where $P_i(\mathbf{X})$ are true probabilities that vector $\mathbf{X}$ belongs to the class C_i. In most cases only the class label is given instead of $P_i(\mathbf{X})$ probabilities, therefore $P_i(\mathbf{X}) = 1$ if the label of the training vector $\mathbf{X}$ is C_i, and 0 otherwise. Most classifiers minimize this error so there will be little to correct, but there are systems (for example Kohonen networks, rough set classifiers or decision trees) that do not minimize cost function directly. Adding such corrections may be regarded as the simplest form of stacking, i.e. training one classifier on the outputs of another [1].

The error function is quadratic in κ parameters:

$$E(\kappa) = \sum_{\mathbf{X}}\sum_{i=1}^{K}\left(\kappa_i p(C_i|\mathbf{X}) - P_i(\mathbf{X})\right)^2 \tag{4}$$

If linear rescaling is used and $\kappa_i \in [0,1], i = 1\dots K-1$ then due to normalization (1) the cost function is:

$$E(\kappa) = \sum_{\mathbf{X}}\sum_{i=1}^{K-1}\left(\kappa_i p(C_i|\mathbf{X}) - P_i(\mathbf{X})\right)^2 + \sum_{\mathbf{X}}\left(1 - \sum_{i=1}^{K-1}\kappa_i p(C_i|\mathbf{X}) - P_K(\mathbf{X})\right)^2 \tag{5}$$

Minimum of this function is obtained from a solution of linear equation:

$$\mathbf{P}\cdot\kappa = \mathbf{Q};\ \ \mathbf{P}_{ij} = \sum_{\mathbf{X}}(1+\delta_{ij})p(C_i|\mathbf{X})p(C_j|\mathbf{X}), \quad i,j = 1\dots K-1; \tag{6}$$

$$\mathbf{Q}_i = \sum_{\mathbf{X}} p(C_i|\mathbf{X})(1 + P_i(\mathbf{X}) - P_K(\mathbf{X})) \tag{7}$$

Note that for the perfect classifier all $\kappa_i = 1$ since there is nothing to correct. For the majority classifier all $\kappa_i = 0$ because all probabilities except $p(C_K|\mathbf{X})$ are zero and no scaling of other classes will change it. However, if softmax transformation is used rescaling becomes possible. Starting from $p_K(C_i|\mathbf{X}) = 1$ and all other probabilities equal to zero, minimization of the cost function (4) with softmax transformed probabilities (2) leads to $P(C_K|\mathbf{X}) = N_K/N$ and all other probabilities $P(C_i|\mathbf{X}) = (N - N_K)/N(K-1)$. These probabilities may further be corrected by rescaling them and applying the softmax transformation iteratively. For 2-class problems there is only one scaling parameter κ, therefore it is easy to plot the number of errors as a function of κ. The number of errors is discontinuous and for more than two classes a minimum should be found using a global minimization method.

The primate splice-junction gene sequences (DNA) was used in the Statlog project [3]. It contains a set of 3190 sequences composed of 60 nucleotides. The task is to distinguish if there is an “intron => exon" or “exon => intron" boundary in the string, or neither. 2000 strings are used for training and 1190 for testing. The k nearest neighbor (kNN) classifer used for this task gave 85.4% on the test data. Optimizing k and the distance metric (using crossvalidation on the training data) only a slight improvement was obtained: with k=11 and Manhattan distance function accuracy reached 85.8% on training and 85.7% on the test data. After optimization of the κ parameters ($\kappa_1 = 1,0282,\ \kappa_1 = 0,8785$ the results improved on the training set to 86.4% and on the test set to 86.9%. Many probabilities of wrongly classified vectors became close to the threshold of predicting correct class, therefore comparison of an area under the ROC curves shows even more significant improvement. Similar improvements of the kNN results were noticed on several other datasets. More tests should be conducted with other data and classification methods but since calculation of such corrections does not cost anything they may be worthwhile to implement.

2.1 Changes in the a priori class distribution

Another type of corrections is useful if the a priori class distribution in the training data differs from the test data. If both the training and the test data represent the same process underlying data generation the densities $p(\mathbf{X}|C_i)$ in each class should remain constant, but the a posteriori probabilities $p(C_i|\mathbf{X})$ may change. According to the Bayes theorem:

$$p(\mathbf{X}|C_i) = p(C_i|\mathbf{X})p(\mathbf{X})/p(C_i) \tag{8}$$

An estimate of the a priori probabilities $p(C_i)$ should be made on the test data. This may be done naively by sampling the new data or in a more sophisticated way using iterative Expectation Maximization algorithm [2]. As usually the classification model is created on the training data, providing priors $p_t(C_i)$ and posteriors $p_t(C_i|\mathbf{X})$. Using Bayes theorem for the probabilities p_t estimated on the training set and using the assumption that the a posteriori probabilities do not change $p_t(\mathbf{X}|C_i) = p(\mathbf{X}|C_i)$ we obtain:

$$p(C_i|\mathbf{X})p(\mathbf{X})/p(C_i) = p(\mathbf{X}|C_i) = p_t(C_i|\mathbf{X})p_t(\mathbf{X})/p_t(C_i) \tag{9}$$

To eliminate $p(\mathbf{X})$ and $p_t(\mathbf{X})$ note that the $p(C_i|\mathbf{X})$ probabilities sum to 1 over all classes. Therefore

$$p(C_i|\mathbf{X}) = \frac{p_t(C_i|\mathbf{X})p(C_i)/p_t(C_i)}{\sum_j p_t(C_j|\mathbf{X})p(C_j)/p_t(C_j)} \tag{10}$$

i.e. the new posterior probabilities are simply corrected by the ratio of new priors $p(C_i)$ to the old priors $p_t(C_i)$ estimated on the training set, and renormalized to sum to one. A model estimated on the training data set may thus be applied to new data that has different a priori probabilities. For example, a model trained on data from controlled clinical experiments (with $p_t(disease)$ usually around 0.5) may be used in real world situations, when prior probabilities may be different (when $p_t(disease) \ll 0.5$).

Tests on several two-class medical data that were used in the Statlog project [3] showed that if the training data has $p(C_1)/p(C_2) = 1$ (typical for controlled experiments) and the class distribution is changed to $p(C_1)/p(C_2) = 5$ (even higher difference may be observed in real applications) this correction frequently increases accuracy by more than 10%.

2.2 Rejection rate, confidence, sensitivity and specificity

Frequently there is a need to improve confidence in the classification model. This may be done at a cost of rejecting some cases, i.e. assigning them to the “unknown” class. After training neural network or other classifiers outputs from these models are interpreted as probabilities. Suppose that the confusion matrix $\mathsf{P}(C_i,C_j|M)$ for the two-class problem is known:

$$\mathsf{P}(C_i,C_j|M) = \frac{1}{n}\left(\begin{array}{c|ccc|c} & C_+ & C_- & C_r & \\ C_+ & n_{++} & n_{+-} & n_{+r} & n_+ \\ C_- & n_{-+} & n_{--} & n_{-r} & n_- \end{array}\right) = \left(\begin{array}{c|ccc|c} & C_+ & C_- & C_r & \\ C_+ & p_{++} & p_{+-} & p_{+r} & p_+ \\ C_- & p_{-+} & p_{--} & p_{-r} & p_- \end{array}\right) \tag{11}$$

with rows corresponding to true classes, columns to predicted classes, and p_{ij} computed for a model M on $n = n_+ + n_-$ samples. The $p_{ij} = p(C_i,C_j|M)$ quantities are the training set estimations of joint probabilities of finding true class C_i and the predicted class C_j in the results; the model may also reject some cases as unpredictable, assigning them to the class C_r. The $p_\pm = p(C_\pm) = n_\pm/n$ are the a priori probabilities for the two classes, $p_\pm = p_{\pm+} + p_{\pm-} + p_{\pm r}$, and $p_+ + p_- = 1$.

To increase confidence in the decision of the model the errors on the test set may be decreased at the cost of rejection of some vectors. In neural networks this is done by defining minimum and maximum thresholds for the activity of output units. In models estimating probability similar thresholds may be introduced. For crisp logical rules confidence optimization is also possible if uncertainty of inputs are taken into account [4]. The following error function may be used for corrections after training of the model to set these thresholds:

$$E(M;\gamma) = \gamma \sum_{i \neq j} \mathsf{P}(C_i,C_j|M) - \mathrm{Tr}\,\mathsf{P}(C_i,C_j|M) \geq -1 \tag{12}$$

It should be minimized over model parameters M without constraints. Several quantities are used to evaluate classification models M created to distinguish C_+ class:

- Overall accuracy $A(M) = p_{++}(M) + p_{--}(M)$
- Overall error rate $L(M) = p_{-+}(M) + p_{+-}(M)$
- Overall rejection rate $R(M) = p_{+r}(M) + p_{-r}(M) = 1 - L(M) - A(M)$
- Sensitivity $S_+(M) = p_{+|+}(M) = p_{++}(M)/p_+$, or conditional probability of predicting class C_+ when the vector was indeed from this class.
- Specificity $S_-(M) = p_{-|-}(M) = p_{--}(M)/p_-$ (same for class C_-).

Note that the overall accuracy is equal to a combination of sensitivity and specificity weighted by the a priori probabilities:

$$A(M) = p_+S_+(M) + p_-S_-(M) \tag{13}$$

Thus sensitivity (specificity) plays the role of accuracy of the model for the C_+ (C_-) class only, with p_+ (p_-) being the fraction of samples from this class (other classes) in the training set. In the K-class problem one can always use a separate model to distinguish between a single class C_+ and all other classes C_-. The cost function for the model M_+ is (all $p_{ij} = p_{ij}(M_+)$):

$$E(M_+;\gamma) = \gamma L(M_+) - A(M_+) = \gamma\,(p_{+-} + p_{-+}) - (p_{++} + p_{--}) \tag{14}$$

should be minimized over parameters of the M_+ model created for the C_+ class. For large γ only the error is important and it may decrease at the expense of the rejection rate, for example by making the rule intervals more tight or the thresholds for neural activity closer to 1 and 0. In extreme case no errors will be made on the training set since the classifier will reject all such cases. For $\gamma = 0$ only accuracy is maximized, leading to less rejections. Using the error (loss) and the rejection rate the formula (14) becomes:

$$\min_M E(M;\gamma) \Leftrightarrow \min_M \{(1+\gamma)L(M) + R(M)\} \tag{15}$$

For $\gamma = 0$ a sum of the overall error and rejection rate is minimized, while for large γ the error term dominates, allowing the rejection rate to grow. In many applications it is important to achieve highest sensitivity or specificity. The error function (15) distinguishes only one of these quantities. Introduction of relative costs should be used instead. If the cost of assigning vectors from true class C_- to the predicted class C_+ is set to 1, and the cost of making an opposite error is α, the cost function is:

$$\min_M\ E(M;\alpha) = \min_M \{p_{+-}(M) + \alpha p_{-+}(M)\} \tag{16}$$

$$= \min_M\ \{p_+(1 - S_+(M)) - p_{+r}(M) + \alpha\,[p_-(1 - S_-(M)) - p_{-r}(M)]\} \tag{17}$$

For $\alpha = 0$ this is equivalent to maximization of $p_{++}(M) + p_{+r}(M)$ and for large α to maximization of $p_{--}(M) + p_{-r}(M)$.

Receiver Operator Characteristic (ROC) curves show the S_+ values as a function of $1 - S_-$, allowing for another way of adjusting the rejection thresholds. The approach described here allows for optimization with explicit costs; we have used it to optimize logical rules [4] where all predictions are binary and ROC approach is not so convenient.

3 Conclusions

Three types of corrections that can be applied *a posteriori*, i.e. after the model has already been trained, has been described in this paper. Some of these corrections may also be applied during the training process. They are aimed either at:
1) the overall increase of accuracy by scaling the probabilities on the training set and adding linear or nonlinear (softmax) scaling factors,
2) restoring the balance between the training and the test set if the distributions are quite different, or
3) improving confidence of classification, selectivity or specificity of results.

The first of these corrections is similar (although simpler) to the effect of adding an additional linear or non-linear perceptron trained on results obtained from other models, a procedure known as stacking [1]. Tests with the kNN method on DNA splice data set and other datasets show that this correction may improve results with no computational cost. The second correction is useful in real-world situations, when real distribution of classes in the data does not match the distribution in the data given for training. Finally the overall accuracy is not always the most important parameter. Relations between costs, selectivity, specificity and error functions have been investigated. We found them especially useful for optimization of logical rules.

More empirical tests are needed to evaluate usefulness of these corrections in practice, but our preliminary results are encouraging.

Acknowledgments: Support by the Polish Committee for Scientific Research, grant 8 T11C 006 19, is gratefully acknowledged.

References

1. Wolpert D.H. (1992) Stacked generalization. Neural Networks **5**, 241-259
2. Saerens M, Lattinne P, Decaestecker C. (2002) Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. Neural Computations (in press).
3. D. Michie, D.J. Spiegelhalter and C.C. Taylor, “Machine learning, neural and statistical classification". Elis Horwood, London 1994
4. Duch W, Adamczak R. and Grźbczewski K. (2001) Methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks* **12**: 277-306