Easily Interpretable, Non-parametric Sample Transformation for Classification

Cédric DUBOIS[1], Jean-Olivier IRISSON[2], Éric DEBREUVE[1]

[1]Université Côte d'Azur, CNRS, Inria, Equipe Morpheme, Laboratoire I3S, Sophia Antipolis, France.

[2]Sorbonne Université, Equipe COMPLEx, Laboratoire LOV, Institut IMEV, Villefranche-sur-mer, France.

cedric.dubois@univ-cotedazur.fr, jean-olivier.irisson@imev-mer.fr, eric.debreuve@univ-cotedazur.fr

Résumé – Les CNNs (Réseaux de Neurones Convolutionnels) sont largement utilisés pour la classification supervisée. Bien que les réseaux eux-mêmes soient généralement vus comme des classifieurs, ce sont en fait des régresseurs optimisés pour approximer la relation entre les données brutes et p vecteurs prédéfinis de $\mathbb{R}^p$ jouant le rôle de représentants de classe, où p est le nombre de classes. La véritable classification est faite par un classifieur par plus proche voisin appliqué aux sorties du réseau.

Malgré des performances de classification généralement très bonnes, les ANNs (Réseaux de Neurones Artificiels) ne sont pas toujours aussi simples à utiliser que les classifieurs classiques car ils nécessitent souvent de grandes quantités de données, une charge de calcul importante, et parfois une solide expérience pour être entraînés. Pourtant, le principe des ANNs (transformation des données d'entrée vers $\mathbb{R}^p$, puis classification par plus proche voisin) est intéressant. Dans ce travail, nous proposons une alternative simple, facilement interprétable et nécessitant une charge de calcul modérée, et qui suit ce même principe. Elle repose sur une combinaison pondérée de translations idéales des échantillons d'apprentissage vers des vecteurs cibles prédéfinis. En raison de sa simplicité, elle ne peut pas traiter directement les données brutes comme le font les ANNs. Elle s'applique plutôt à des caractéristiques extraites. Expérimentalement, nous avons obtenu des performances de classification comparables à celles de classifieurs classiques, y compris sur une base de données réelle d'images de plancton.

Abstract – CNNs (Convolutional Neural Networks) are widely used for supervised classification. Although the networks themselves are designated as classifiers, they are in fact regressors trained to approximate the relationship between raw data and p predefined vectors of $\mathbb{R}^p$ playing the role of class representatives, where p is the number of classes. The actual classification decisions are taken by a nearest-neighbor classifier applied to the network outputs.

Despite their usually impressive classification accuracies, ANNs (Artificial Neural Networks) are not always as straightforward to use as classical classifiers since they typically require large amounts of data, a high computational effort, and sometimes a solid experience to be trained. Yet, the principle of ANNs (input transformation into $\mathbb{R}^p$, then basic nearest-neighbor classification) is interesting. In this work, we propose a simple, easily interpretable, and low on computational requirements alternative following the same principle. It relies on a weighted combination of ideal translations from the learning samples to some predefined targets. Because of its simplicity, it cannot directly deal with raw data as the ANNs do. Instead, it works with extracted features. Our experimental results, including on a realworld database of Plankton images, show classification accuracies on par with some classical classifiers.

1 Introduction

ANNs, and in particular CNNs, are widely used for supervised classification [6, 7].[1] Despite designating the networks themselves as classifiers, they are in fact multi-valued regressors where the independent variables are defined by the data space, and the dependent variables are defined in $\mathbb{R}^p$, where p is the number of classes. During the learning phase, the "observed" value (i.e., the desired network output) for a given input is typically set to one of the vectors of the canonical basis of $\mathbb{R}^p$ which play the role of class representatives. Let these vectors be denoted as targets. Since the networks generally have a so-called softmax layer as their last layer, their outputs are restricted to the probability simplex in $\mathbb{R}^p$. When correctly trained, the networks transform inputs into vectors that concentrate close to the targets. During the prediction phase, the actual classification decision is then taken by a NN (Nearest Neighbor) classifier applied to the network output. Let us call t-NN this classifier. In practice, this decision is rather described as an $\arg\max$ operation, but the equivalence with NN classification is trivial to establish.

While ANNs often reach impressive classification accuracies, they are not always as straightforward to train and interpret as classical classifiers. Here are some issues:

- They typically require large amounts of training data.Their collection and annotation can be very time consuming.
- Their training demands a high computational effort, which, with the democratization of Deep Learning, questions the ecological impact of such approaches [4].

[1]The literature is so huge on this topic that it is impossible to select a reasonable number of even "essential" references. Instead, we decided to provide only two fairly recent references, one application and one book.

- Their training sometimes requires a solid experience: architecture selection/design, learning parameter tuning, what to change when the training fails?
- Even though some studies are conducted to understand the behavior of ANNs and interpret their outputs [10], the proposed tools still do not provide simple ways to know how to modify a network architecture or the training process when the accuracy is not good enough.

Yet, the principle of ANNs (input transformation, then basic NN classification) is interesting. It can also be found in classification using logistic regression [5] for example. The aim of this work is to propose a simple and easily interpretable solution with low computational requirements following the same principle. Because of the simplicity of the proposed solution, we cannot directly deal with raw data as the ANNs do. Instead, we take extracted features as input (either hand-crafted or automatically computed).

2 Proposed Method

As described in section 1, the classification approach of ANNs is: *(i)* during learning, a sample transformation is optimized so that samples are condensed into compact, per-class clusters, and *(ii)* during prediction, a simple decision is made to assign a class to a transformed sample. Our goal is to propose a classification method following the same approach, but with an analytical sample transformation definition (i.e., not being the result of an optimization).

2.1 Definitions

Let $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ be a dataset of m distinct n-dimensional samples $\{\mathbf{x}_1, \ldots \mathbf{x}_m\} = \mathcal{X}$ with their associated labels (or classes) $\{y_1, \ldots, y_m\} = \mathcal{Y}$ where the labels $y_{i\in[1..m]}$ are integers between 1 and p, the number of classes. Let $\mathcal{T} = \{\mathbf{t}_1, \ldots, \mathbf{t}_p\}$ be a set of distinct n-dimensional target vectors, or simply targets. They will play the role of the canonical basis vectors used by ANNs. Let us call $\mathbf{t}_{y_i} - \mathbf{x}_i$ the target translation of $\mathbf{x}_i$.

2.2 Sample Transformation

The purpose is to define a transformation mapping a given sample to a vector as close as possible to the target corresponding to its class. For the samples of the learning dataset $\mathcal{D}$, the ideal transformation is to apply the corresponding target translation. For a sample in general, the transformation could combine the target translations, using a typical weighted sum. It is therefore defined as follows

$$\varphi_\gamma(\mathbf{u}) = \mathbf{u} + \phi_\gamma(\mathbf{u}) = \mathbf{u} + \sum_{j=1}^{m} \alpha_\gamma(\mathbf{u} - \mathbf{x}_j)(\mathbf{t}_{y_j} - \mathbf{x}_j) \quad (1)$$

where the weight function α_γ is defined such that

$$\alpha_\gamma(\mathbf{u} - \mathbf{x}_i) = \frac{\kappa_0 K_\gamma(\mathbf{u} - \mathbf{x}_i)}{\sum_{j=1}^{m} \kappa_0 K_\gamma(\mathbf{u} - \mathbf{x}_j)} \quad (2)$$

where K_γ is a symmetric kernel monotonically decreasing with the $\mathcal{L}2$ norm of its argument and fulfilling the usual properties (real-valued, non-negative, and integrable with a unit integral). This form of function is called normalized RBF (Radial Basis Function). The parameter γ controls the decreasing rate of the kernel. For convenience, it is defined such that higher values correspond to higher decreasing rates. Thus, if the kernel is a Gaussian, γ is proportional to the inverse of the variance. Finally, the constant κ_0 is equal to $1/K_\gamma(0)$. Its purpose is to anchor the kernel to a fixed value (a value of 1 at the origin) to avoid technical difficulties when γ tends toward zero or infinity. Let us call γ the kernel width and let $\varphi_\gamma(\mathcal{X})$ denotes the set of the samples of $\mathcal{X}$ transformed by φ_γ.

This transformation definition does provide a sound solution to our problem. However, it can be noted that the "price to pay" for extending in this way the transformation from the learning dataset to any sample is that $\phi_\gamma(\mathbf{x}_i)$ will generally not be equal to the target translation of $\mathbf{x}_i$. The proposed classification procedure is then to transform a sample with eq. (1), and (similarly to ANNs) to take the classification decision using a t-NN classifier on the transformed sample with the targets $\mathcal{T}$ as class representatives (see section 2.3.1 for the effective choice of targets). In addition to the assigned class, a form of confidence vector can be obtained by applying a softmax-based function to the components of a vector of $\mathbb{R}^p$ composed of the Euclidean distances between the transformed sample and each of the targets

$$C(\mathbf{x}_i) = 1 - \text{softmax}\{||\varphi_\gamma(\mathbf{x}_i) - t_l||_2, l \in [1..p]\}, \quad (3)$$

which is exactly the kind of information provided by the output layer of classification ANNs.

Note that ϕ_γ is a multi-valued version of the Nadaraya–Watson kernel regression estimator [9] where each value is a component of the translation to apply to a sample $\mathbf{u}$.

As a reminder, the class assigned to a sample is the one of the target nearest to its transformed version.

2.3 Influence of the Kernel Width γ

2.3.1 Limit Cases and Choice of Targets

It can be verified that the weight function α_γ take the following expressions when γ tends toward zero or infinity[2]

$$\alpha_{\gamma\to 0}(\mathbf{u} - \mathbf{x}_i) = \frac{1}{m} \quad \forall i, \quad (4)$$

$$\alpha_{\gamma\to\infty}(\mathbf{u} - \mathbf{x}_i) = \begin{cases} 1 & \text{if } \mathbf{u} = \mathbf{x}_i \\ 0 & \text{if } \mathbf{u} = \mathbf{x}_j, j \neq i \end{cases}. \quad (5)$$

Consequently,

$$\varphi_{\gamma\to 0}(\mathbf{u}) = \mathbf{u} + \frac{1}{m}\sum_{j=1}^{m}(\mathbf{t}_{y_j} - \mathbf{x}_j), \quad (6)$$

$$\varphi_{\gamma\to\infty}(\mathbf{x}_i) = \mathbf{x}_i + (\mathbf{t}_{y_i} - \mathbf{x}_i) = \mathbf{t}_{y_i}, \quad (7)$$

[2]The study of $\alpha_{\gamma\to\infty}(\mathbf{u} - \mathbf{x}_i)$ for $\mathbf{u} \notin \mathcal{X}$ is left for further works.

which means that when γ tends toward zero, samples are translated by a constant vector, while when it goes to infinity, the samples of $\mathcal{X}$ are perfectly mapped to their corresponding targets. The choice of an optimal γ is discussed in section 2.3.2.

As a consequence of eq. (6), if the targets are chosen equal to the centroids of the samples in each class, then $\varphi_{\gamma\to 0}$ becomes the identity. This is our motivation for defining the targets in this manner

$$\mathbf{t}_l = \frac{1}{m_l} \sum_{i\,|\,y_i=l} \mathbf{x}_i \tag{8}$$

where m_l is the number of samples of class l, i.e. the cardinality of $\{i\,|\,y_i = l\}$.

2.3.2 Optimal Kernel Width γ

In machine learning, a typical way of optimizing a parameter is by cross-validation. Normally, this would be done based on the classification accuracy. However, since the proposed sample transformation can be viewed as a solution to a regression problem (see section 2.2), we can instead study the error between the transformed samples $\varphi_\gamma(\mathcal{X})$ and their respective targets. Thus, we can rely on the following result in leave-one-out cross-validation for regression. The total squared error on the qth component of a transformed sample, defined as

$$J^{(q)}(\gamma) = \sum_i^m \left(\mathbf{t}_{y_i}^{(q)} - \varphi_\gamma^{\{-i\}}(\mathbf{x}_i)^{(q)}\right)^2, \tag{9}$$

can be computed as

$$J^{(q)}(\gamma) = \sum_i^m \left(\frac{\mathbf{t}_{y_i}^{(q)} - \varphi_\gamma(\mathbf{x}_i)^{(q)}}{1 - \frac{1}{\kappa_0 \sum_j^m K_\gamma\left(\mathbf{x}_i - \mathbf{x}_j\right)}}\right)^2, \tag{10}$$

which relies on the original transformation whereas the definition (9) involves $\varphi_\gamma^{\{-i\}}$, corresponding to a leave-on-out version (the contribution of $\mathbf{x}_i$ is not accounted for) (see [8], Theorem 20.22, p. 320). This makes eq. (10) less computationally intensive to apply because the weights $\alpha_\gamma(\mathbf{x}_i - \mathbf{x}_j), j > i$ are computed only once, as opposed to an order of m times with eq. (9).

The value of γ minimizing $J^{(q)}$ depends on q. Instead, we want a unique optimizer for all the sample vector components in order to use eq. (1). This can be obtained by defining a total error for all the components as a function of the $J^{(q)}$'s, $q \in [1..n]$

$$J(\gamma) = \mathcal{F}(J^{(1)}(\gamma), J^{(2)}(\gamma), \ldots, J^{(n)}(\gamma)). \tag{11}$$

Two obvious choices for $\mathcal{F}$ are the mean or the maximum of its arguments. We chose to use the stricter maximum option. Even though the minimization of this regression error does not guaranty the minimization of the empirical classification risk, we observed that it led to optimal values of γ very satisfying for the classification task (see section 3).

3 Experimental Results

3.1 Numerical Implementation

For each sample $\mathbf{u}$, the weights $\alpha_\gamma(\mathbf{u} - \mathbf{x}_i), i \in [1..m]$ must be computed to apply the transformation (1). In practice, the weights for the samples of $\mathcal{X}$ far enough from $\mathbf{u}$ are negligible. Thus, to reduce the computation load (and time), we propose to implement the sample transformation as

$$\tilde{\varphi}_\gamma(\mathbf{u}) = \mathbf{u} + \sum_{\mathbf{x}_j \in \mathcal{N}_h(\mathbf{u})} \alpha_\gamma(\mathbf{u} - \mathbf{x}_j)(\mathbf{t}_{y_j} - \mathbf{x}_j), \tag{12}$$

where $\mathcal{N}_h(\mathbf{u})$ is the set of the h NNs of $\mathbf{u}$ among the samples of $\mathcal{X}$.To efficiently select the NNs, we put the learning samples into a k-D Tree (k-Dimensional Tree) structure [1]. Hence, the search complexity becomes logarithmic in the cardinality of $\mathcal{X}$, as opposed to linear for the brute-force search. The parameter h has to be tuned for each experiment, but we found that the method performances are not very sensitive to it in a fairly large interval. Nevertheless, h could be adjusted by cross-validation.

All the experiments were conducted with a Gaussian kernel so that $\gamma = 1/(2\sigma^2)$ where σ is the standard deviation. The optimal value of γ is related to the density of the data. It was obtained by a grid search optimization of the total error defined in section 2.3.2.

3.2 Synthetic Dataset

To be able to check the results visually, we created a 2-dimensional synthetic dataset. It is composed of 4000 (learning) samples evenly split into 4 classes, two of which being more spread (see fig. 1). The optimization of γ was done on a grid of 5 values distributed on a logarithmic-scale from 10^1 to 10^5. The optimal value was $\gamma = 10^3$. We took $h = 100$ NNs for the computation of the sample transformations. We then generated 4000 test samples in the same conditions to evaluate the classification accuracy of the proposed method and four other classifiers: the t-NN classifier applied to untransformed samples, a k-nn (k-Nearest Neighbors) classifier, a kernel-SVM (Support Vector Machine) with a Gaussian kernel, and a RF (Random Forest) [7]. For the latter three, we ran 5-fold cross-validation within a grid search to find optimal parameters. For the k-nn classifier, the best parameter was $k = 50$. For the kernel-SVM, the width of the kernel was 100 and the regularization parameter C was equal to 100. For the RF, the minimum samples per leaf was 10, with no limit on the depth of the trees, and 100 trees. The accuracy of the t-NN classifier was 75%. The accuracies of the other four classifiers were not significantly different, and around 90%. To get a feeling of the sensitivity of the proposed method to γ, we also applied it with $\gamma = 10^2$, 10^4, and 10^5. We obtained accuracies of 89%, 89%, and 86%, respectively. So, on this coarse grid, the optimal γ indeed corresponds to a maximum of a concave (with this discretization) function.

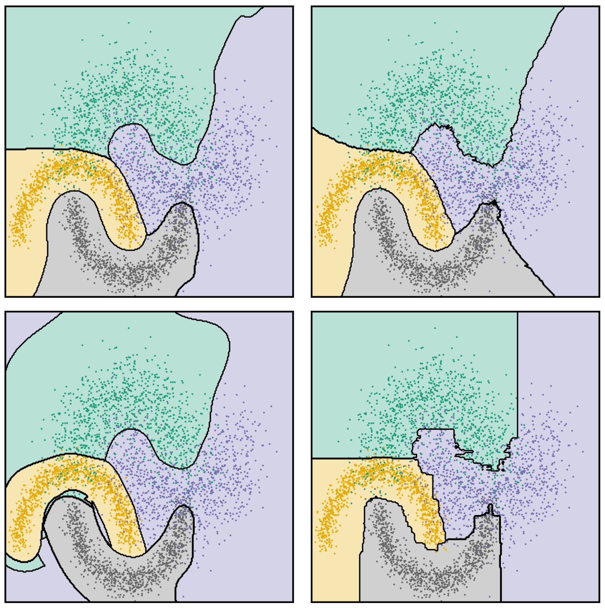

FIG. 1: Classification frontiers plotted in black from predictions on a fine, regular sample grid. The dots are the training samples. Top left: proposed method, top right: k-nn classifier, bottom left: kernel-SVM, bottom-right: RF.

3.3 Real world application: ZooScan dataset

The ZooScan dataset [3] is composed of $m = 1.45$ million grayscale images of plankton divided into $p = 136$ classes, which makes the learning of a classifier a challenging task. Additionally, there is a huge unbalance between the classes, with the smallest containing only 4 samples while the largest contains almost 170000. In order to apply the proposed method, we took the features output by the last layer before the fully connected part of a CNN classifier.[3] Then, we performed a Principal Component Analysis, and we retained the 10 principal components as the features for our tests ($n = 10$). We partitioned the dataset into a learning set (70% of the samples; used for parameter optimization and learning) and a test set. In order to mitigate the effect of the class unbalance, we adopted the usual strategy of weighting the sample-related terms proportionally to the inverse of the class frequencies. The optimal parameter γ was 10^2, and we set the parameter h to 30. We obtained an accuracy of 73% and a balanced accuracy of 74%. These scores are close to each other, meaning that the class unbalance management was appropriate. For comparison, we learned a RF with 100 trees and a minimum samples per leaf of 25, and obtained 77% and 72% of accuracy and balanced accuracy, respectively. We do not explain why this classifier did not benefit more from the class unbalance management. Whether balanced or not, the accuracies of both classifiers are comparable, which illustrates on a real dataset that the proposed method is competitive.

[3]See details at `github.com/ecotaxa/ecotaxa_ML_template`.

4 Conclusion and Discussion

Inspired by the ANNs principle, we proposed a classification method based on an easily interpretable, low on computational requirements, non-parametric sample transformation. If we stick with the Gaussian kernel for α_γ and ignore the parameter h (which is just an implementation trick), it has only one parameter γ which can be optimized using a cross-validation procedure based on an efficiently computable regression error. It achieved very good results on a synthetic and a real dataset, on par with some classical classifiers.

In future works, we will turn the global parameter γ into per-sample parameters γ_i in order to adapt to datasets with inhomogeneous spatial sample density, whether inter- or intra-class. We can also note that, besides the already mentioned interpretation of ϕ_γ in terms of Nadaraya–Watson kernel regression [9], it can also be viewed as a normalized RBF Network [2] with one hidden layer. We plan to exploit this analogy to optimize the proposed sample transformation.

Experimentally, we will study the behavior of the method as the number of features n increases.

References

[1] J. L. Bentley. “Multidimensional Binary Search Trees Used for Associative Searching”. In: *Commun. ACM* 18.9 (1975), pp. 509–517.

[2] D. Broomhead and D. Lowe. “Radial basis functions, multi-variable functional interpolation and adaptive networks”. In: *Royal Signals and Radar Establishment Malvern (United Kingdom)* RSRE-MEMO-4148 (1988).

[3] G. Gorsky et al. “Digital zooplankton image analysis using the ZooScan integrated system”. In: *Journal of Plankton Research* 32.3 (Mar. 2010), pp. 285–303.

[4] P. Henderson et al. “Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning”. In: *Journal of Machine Learning Research* 21.248 (2020), pp. 1–43.

[5] D.W. Hosmer, S. Lemeshow, and R.X. Sturdivant. *Applied Logistic Regression*. John Wiley & Sons, Ltd, 2000.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Commun. ACM* 60.6 (2017), pp. 84–90.

[7] M. Kubat. *An introduction to machine learning*. Vol. 2. Springer, 2017.

[8] L. Wasserman. *All of Statistics*. Ed. by Springer. New York, NY: Springer, 2004.

[9] G. S. Watson. “Smooth Regression Analysis”. In: *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)* 26.4 (1964), pp. 359–372.

[10] N. Xie et al. *Explainable Deep Learning: A Field Guide for the Uninitiated*. arXiv, 2020. URL: `https://arxiv.org/abs/2004.14545`.