Wine classification: feature selection with various machine learning techniques

FOUFA Mastafa

School of Computer Sciences
KTH Royal Institute of Technology
Brinellvägen 8, 114 28 Stockholm,
Sweden
foufa@kth.se

REY Romain

School of Computer Sciences
KTH Royal Institute of Technology
Brinellvägen 8, 114 28 Stockholm,
Sweden
rrey@kth.se

Abstract— We present and study different feature selection techniques, namely PCA (Principal Component Analysis), stacked autoencoders (SAE) and variational autoencoders (VAE). The goal is to compare them based on different metrics. Empirical comparison between those different feature selection methods allows to know which technique is the best for classifying our dataset.

A tradeoff of some introduced criteria helps us better estimate which technique should be preferred.

Keywords—Auto-encoder, PCA, Dimensionality Reduction, Visualization

I. INTRODUCTION

Inductive supervised learning infers a functional relation $y = f(x)$ from a set of training examples $T = \{(x_1, y_1), \dots, (x_n, y_n)\}$

In the following, we consider input data representing a wine data set. Each input is a vector of the form $[x_{i1}, x_{i2}, \dots, x_{ip}]\ in\ R^p$ and $y\ in\ \{-1, 1\}$. We refer to f as a classifier. Our goal is to produce a classifier that makes accurate predictions for future input vectors.

Applications where p is large need our attention as computational complexity of a lot of algorithms is quadratic or exponential in p. This means the larger the p the more computation time will be expected. Moreover, large p generally requires complexity control to avoid overfitting the training data. Other cases where p is relatively large compared to our dataset size also need particular attention. As a result, we can think about several data reduction techniques mitigating this problem. Ideally, we want a method that preserves properties of the original space like distances and angles. Principal Components Analysis (PCA) is very popular but other techniques based on deep learning are now also used for this kind of tasks. Using a Neural Network based on the encoder-decoder paradigm, data is projected into a manifold. Then we obtain feature vectors (encodings) which constitute a reduced representation of our original data. Consequently, it will be possible to work on those feature vectors instead of the original data.

Autoencoder, also known as auto-association, is a three-layered neural network and was studied by a number of researchers in 1990s. In 1991, Kramer [1] introduced how to conduct nonlinear principal component analysis using auto encoders with three hidden layers. But, learning is difficult in deep learning and, because of that, deep networks with stacked autoencoders was outside of the attention for a long time. But thanks to Geoffrey Hinton and the idea of layer-wise pretraining [2] to multi-layer auto-encoders using RBM, autoencoders gained attention. A lot of breakthroughs have been witnessed in the past years thanks to deep learning techniques. Recently, Le et al. [4] discovered that a 9-layered locally connected sparse auto-encoder was sensitive to high-level concepts such as cat faces.

In this report, we describe experiments that examine the efficacy of deep learning techniques for supervised learning and compare it with more simple ones like PCA. Deep learning techniques are very powerful but in general need a lot of computation time. Relevant criteria are hence introduced to better compare the different techniques. We will also introduce some metrics to measure properties that dimensionality algorithm can bring as tolerance to outliers and ability to learn on smaller samples of data. In our work, we first briefly discuss the theoretical backgrounds of PCA, SAE and VAE, then we introduce the metrics we use to compare those methods on our wine datasets. We then finish by discussing the results.

II. METHODS

A. Autoencoders for dimensionality reduction

1) Stacked autoencoder

Let $\boldsymbol{x}$ be the original input belonging to an n-dimensional space and $\boldsymbol{y}$ be its representation in the bottleneck layer of our autoencoder belonging to an m-dimensional space with $m < n$. Let h be a 3-layered autoencoder st $h(x) = g(f(x)) \approx x$. We set the output of the autoencoder equal to the input. Hence, it's an unsupervised learning algorithm which uses backpropagation for training. We can work for example with J the reconstruction error st $J(W, x, y) = \|h(x) - y\|^2$

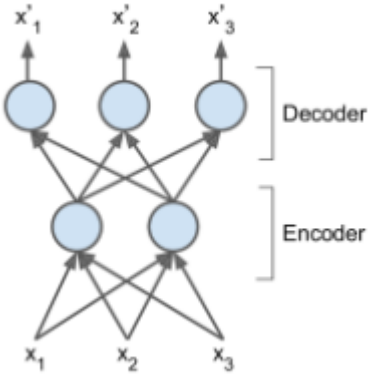

Fig. 1. Simple representation of an autoencoder

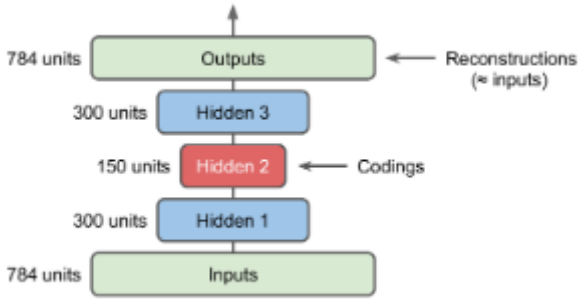

Fig. 2. Simple stacked autoencoder

When we use less units in our hidden layer than our input layer, we obtain a compressed representation of the input and focus on the most important patterns. This is the core of dimensionality reduction.
Stacked autoencoders are merely autoencoders with various hidden layers. Figs. 1 and 2 show the structures of simple autoencoders and how are added hidden layers for creating a stacked autoencoder.

2) . Variational autoencoder

VE is adding a stochastic approach to the prior neural network. It constrains the feature space to follow a simple distribution, generally a Gaussian distribution. We maintain the reconstruction term and this last measure is added to the loss objective that we aim to minimize during training.

More formally, we want to learn the data distribution $p_\theta(X) = \prod_{i=1}^{N} p_\theta(x^{(i)})$ where X is a set of points st $X = \{x^{(1)}, \ldots, x^{(N)}\}$

We decide not to divide into the statistical approach in this report. A simple representation of the process is given in Fig. 3.

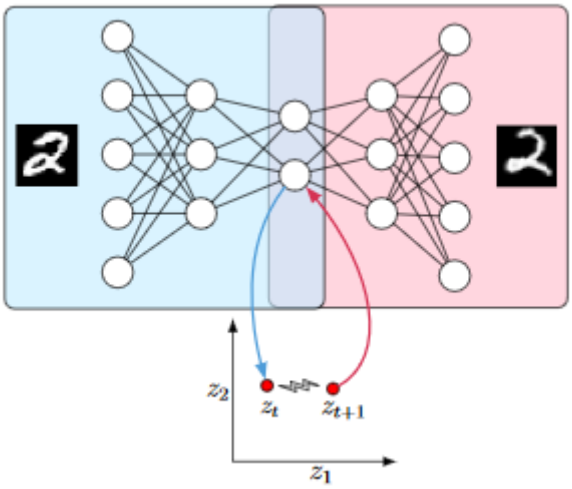

Fig. 3. From left to right: the encoder encodes the image into the latent space, providing a vector z_t. Then, performing some perturbation over this vector, we get the altered version z_t+1. Finally, we decode the last vector into a new image that would be a modified version of the original input

Generation method:

In this paper, we only focus on one method by adding gaussian noise in our bottleneck. To add noise to the latent vectors, we use the following formula [5]

$$\hat{z}^i = z^i + \alpha * X \, , X \sim N(0, \sigma^2)$$

Where z^i is the latent vector and $\hat{z}^i$ the perturbed version as in Fig. 3. Alpha parameter controls the influence of the noise.

B. PCA

Principal component analysis is a very popular linear technique for dimensionality reduction. Given a dataset with features in an p-dimensional space, PCA finds a linear subspace of dimension $q < p$ which tries to maintain most of the variability of the data. In our experiments we use the method with the correlation matrix. If X is centered with mean 0, then correlation matrix is $S = \frac{1}{n} X^T X$. With SVD decomposition of X, we have $S = V D' V^T$ where $D' = \frac{1}{n-1} D^2$ and D diagonal matrix with diagonal elements as singular values of input X.

In our work, we normalize the data. Therefore, we don't need to worry about mean value and relative scale of components. Given an observation x_i, we can project it into a lower dimensional space by rows of the matrix of eigen vector V_q st $x_i' = x_i V_q$.

III. Experiment

A. Description of data

The first part of our work focused on the statistical exploratory analysis of two datasets: red wine data and white wine data. The data has been processed through this exploratory analysis. In appendix, some graphs are available to get a grasp of the data.

The problem has been moved to binary classification with good and bad wines. The attributes are real-valued and normalized before projection and classification. Both datasets have a lot of features compared to the number of observations which justifies the use of dimensionality reduction.

B. . Experimental setup

We compare PCA with the two other deep learning techniques. We use random forest as our unique classifier. We use the default setting for this classifier, only modifying first the number of trees in our forest. That way, we will see the impact of the projection techniques at classification time.

We introduce different points of comparison. First, training time of the methods. Second, the AUC (Area Under ROC Curve) performance of the classification after projection with each method. AUC has been privileged over other metrics as we have unbalanced data. Also, we introduce two other metrics. The 'capacity learning n%' corresponding to the learning performance once we only train our projection+classifier method with n% of the training data. And the 'robustness n%' corresponding to the AUC performance after replacing n% of our training data with false data (labels inverted).

Projection is done into a 5-dimensional space based on the analysis of all possible values. We expect the autoencoders to perform well by representing better the data into this lower dimensional space. Those non-linear methods should indeed have a better performance than the linear PCA.

IV. RESULTS

A. Global results

This part focus on the results presented in Table I.

TABLE I. COMPARISON OF DIMENTIONALITY REDUCTION ALGORITHM FOR WHITE / RED WINE DATASETS

White / Red Wine	**AUC**	**Time**	**Robustness 20%**	**Relative robustness 20%**	**Capacity Learning 50%**	**Relative Capacity Learning 50%**
Default	**0.775 / 0.721**	**0.129 / 0.044**	**0.704 / 0.685**	0.908 / 0.951	0.684 / 0.645	0.882 / 0.895
PCA	0.760 / 0.686	0.138 / 0.052	0.651 / 0.617	0.857 / 0.90	0.684 / 0.620	0.900 / 0.904
Stacked Auto encoder	0.743 / 0.690	63 / 17	0.662 / 0.661	0.891 / 0.957	0.672 / **0.659**	0.905 / **0.954**
VAE	0.745 / 0.691	299 / 87	0.668 / 0.664	**0.896 / 0.961**	**0.692** / 0.658	**0.928** / 0.953

1) Performance (AUC)

The main metric here is the AUC which is measuring the performance of our system. An AUC of 1 would be for a perfect classifier and 0.5 for a random one. (performing no better than a chance – no information classifier)

The AUC does not differ much between the methods but we can see that PCA performs better than VAE, which performs better than stacked auto-encoder. Also, we have to note that all techniques give an AUC smaller than the default (non-preprocessing).

2) Time

The time measured here is the total time, meaning the sum of the time of preprocessing, training of the classifier, and classifying. The duration is 2500 times for the VAE and 500 times for the stacked autoencoder longer than for default. These two are significantly time consuming even for a small dataset as the one we had. On the other hand, PCA takes nearly as long as the default one, with only 1.2 times longer.

3) Robustness 20%

First of all, as expected all results are lower than the basic AUC results. Also, it is interesting to note that now PCA gives results which are worse than the two other dimensionality reduction techniques.

4) Relative robustness 20%

Here we are measuring the ratio between the AUC with robustness at 20% and the normal AUC. The meaning of this ratio could be the following: "what is the impact of introducing some false data in the training phase" in other words "how robust is the model to some data alteration?". Taking the ratio helps to remove the absolute performance and to focus on the effect of the data alteration.

We can see that according to what we observed in the previous column the PCA gives the worst performance. Also, the two autoencoder show the same relative robustness 20%, which is still lower than the one for default.

5) Capacity learning 50%

First, as expected all results are lower than the

basic AUC results. We can see that VAE / stacked autoencoder are respectively performing the best for white / red wine.

6) Relative capacity learning 50%

Here we are measuring the ratio between the AUC with capacity learning at 50% and the normal AUC. The meaning of this ratio could be the following: "what is the impact of training on less data?" in other words "how good is the model to extract the main feature from few data?". Taking the ratio helps to remove the absolute performance and to focus on the effect of the data diminution.
Here VAE methods is the best dimensionality reduction technique for these criteria. It is also interesting to see that dimensionality reduction methods give always better results than default for this metric.

B. Evolution of Relative Robustness

In this part we will focus on Fig. 5.
We chose a logarithmic scale for the ratio tested as the vector of tested ratios was [0.01, 0.02, 0.05, 0.1, 0.3, 0.5]. A ratio of 0.3 means that 30% of the training data has been changed (label inverted).

Globally we can see that all method (including default) are following the same trend: the bigger the ratio is the lower the AUC is. Overall stacked autoencoder looks to be a little bit better than the other methods, especially for small values.
It is also interesting to notice that after a limit (between 0.3 and 0.5) the order is reversed.

C. Evolution of Relative capacity learning

In this part we will focus on Fig. 6.
We tested the model with different percentages of available training data: [0.3, 0.4, 0.5, 0.6, 0.7, 0.8].
It is interesting to see that the VAE is performing way better than the other model when only few training data are used. The improvement of PCA compare to default is quasi linear (always a little bit better whatever the percentage) whereas for VAE the best improvement is for small values. Stacked auto encoder does not seem to perform well here.

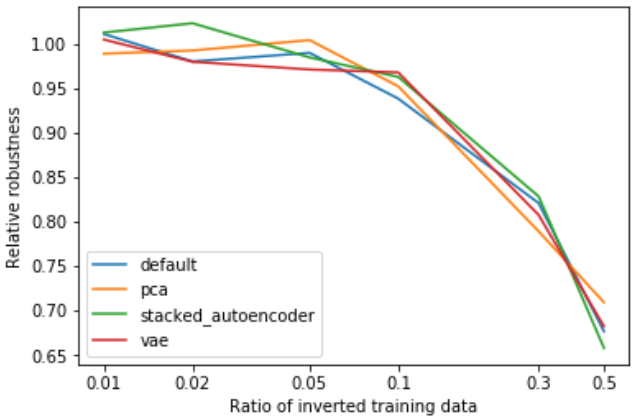

Fig. 5. Evolution of Relative robustness for different ratio of modified training data for each dimensionality reduction method.

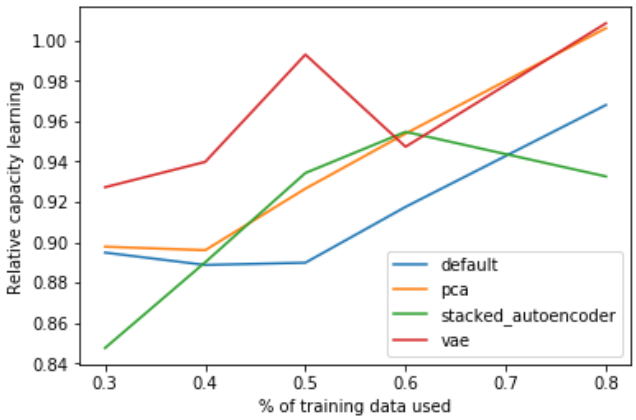

Fig. 6. Evolution of Relative capacity learning for different percentages of available training data for each dimensionality reduction method.

D. Choosing the number of features

In this part we will focus on Fig. 7.
In order to determine the number of features we will create with our dimensionality reduction algorithm we ran different simulation with every number of feature possible (from 1 to 11) using PCA as the preprocessing. According to the figure we can see that 5 features seem to be a good compromise. Indeed after 5 the AUC is not improving significantly anymore, and it starts to go up and down.
So, we chose to keep 5 features for the PCA model and also for the autoencoder models.
Also using 5 features means that we are using half of the number of initial features.

E. Interpretation of the results

1) AUC

As we saw, overall AUC is lower with our dimensionality reductions techniques than with the raw data. This can be explained by the fact that in the wine dataset there are not enough features for the dimensionality reduction to improve the results. It is common to achieve better result with dimensionality reduction techniques but for this, the dataset must have more features.

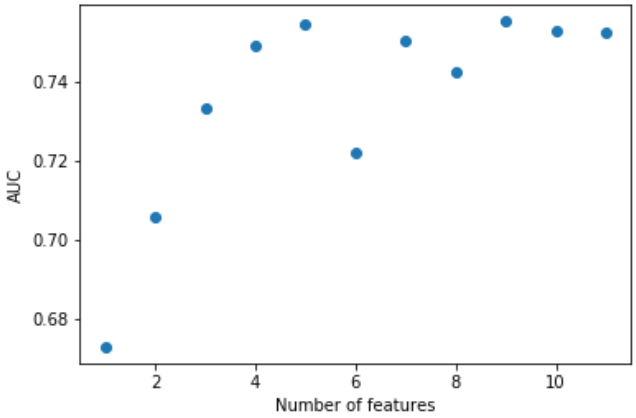

Fig.7. AUC for different number of features using PCA

2) Time

Unsurprisingly deep learning techniques are time consuming. Even if we could have reduced the duration of the preprocessing by optimizing the training phase with early stop for example, it would have still been longer than a technique like PCA. However, something to keep in mind is that the total time is the sum of preprocessing, training of the classifier, and classifying. And the more features you have, the bigger the time of training of the classifier and classifying is. So even if the preprocessing time is also dependent of the number of features, there is a tradeoff where the time lost in preprocessing to reduce the number of features is less than the time won in the classifier phase. That is probably what would have happened if we compared PCA and default with more data. Even if the time of preprocessing of the PCA would have been longer than for default (where there is no preprocessing), computing the classification on 5 features would be faster than on 11, so the final time would probably be smaller. And the more complex the classifier is, the bigger this effect is.

3) Robustness and relative robustness

We saw that auto encoder were performing better than PCA. This could be explained because auto encoders try to learn a more complex (nonlinear) relation in the data itself. Also, PCA try to maximize the variance, but introducing some outlier have a high impact on the variance and might be a source of problem for PCA in this case.

4) Capacity learning and Relative capacity learning

We think that if VAE overperformed the other methods it could be because in its inner process it adds noise to the data. And when we have only a really small amount of data samples this means that we are likely to have only a limited representation of the true distribution. The results show that adding noise looks like a good method to improve the generalization made from a limited representation. This idea is not new and is frequently used in machine learning model (especially in deep learning ones).
Also, the fact that all dimensionality reduction algorithm aim to extract the main characteristics of the data might explain why they generalize better when there are only few data: by approximating the data by their main features, we are less likely to overfit some very particular properties.

V. DISCUSSION

A. Putting it together

During this study we have seen different interesting findings that are worth summarizing:

- Using a dimensionality reduction preprocessing can be made for different purposes: impacting the time of processing, the resistance of outliers or the capacity to train on limited data
- Some deep learning techniques can be effectively used in order to preprocess the data, often leading to overperforming the classical algorithms used in the field like PCA. But this has a cost (time consuming).
- Dimensionality reduction algorithms are not always the best preprocessing to do. Especially in the case of dataset like the wine one where the number of features is already limited. Indeed, we saw that the basic AUC was better with the default method. This will be really dependent on the dataset and on the classifier used.
- But on the other hand, we highlighted that even on this kind of datasets, dimensionality reduction algorithms can be worth trying in order to improve some particular aspects like tolerance to outliers or the capacity to learn with limited data.

B. Limits and Further works

1) Datasets

As a next step, it would be interesting to run our algorithm on different datasets and to see which properties of the dataset affect the metrics we were studying for each preprocessing methods. That should be straightforward using the framework that we defined in our notebook. Indeed, we developed generic functions that are easy to use on different datasets.
Also including a really large and long dataset would have been interesting and would have illustrate the classical benefits we attribute to dimensionality reduction algorithms like time reduction, and better performance in the case of too many features.

2) Models

Of course, the selection of dimensionality algorithms we made is not exhaustive and a lot of other methods could be interesting to study like Kernel PCA with different kernels. But our idea was mainly to compare deep learning approaches with more classical ones.

In addition, we only used random forest as the classifier model, but a next step could be to use other classifiers and identify if there are some stacked models (preprocessing with dimensionality reduction then classifying) which are better performing than other.

Finally building an ensemble model with different models, each one trained with one dimensionality reduction would have been a probably good performing idea to take advantage of the benefits of all preprocessing algorithms.

ACKNOWLEDGMENT

We would like to thank professor Henrik Boström for allowing us to work on such a project within the Programming for Data Science course.

REFERENCES

[1] Mark A. Kramer, Nonlinear principal component analysis using auto-associative neural networks, AIChE J. 37 (2) (1991) 233–243

[2] Geoffrey Hinton, Simon Osindero, A fast learning algorithm for deep belief nets, Neural Comput. 18 (2006) (2006)
[3] Geoffrey Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (20 06) 504–507
[4] Q.V. Le, Building high-level features using large scale unsupervised learning, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 8595–8598.