Machine Learning Homework 3: Unsupervised Learning and Dimensionality Reduction

Nicolas Six

April 1, 2018

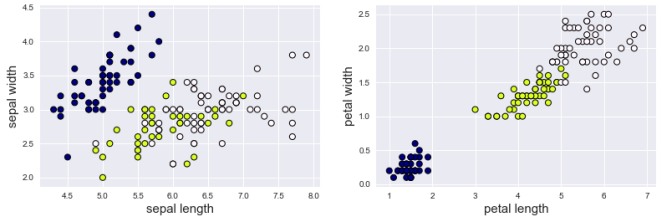
Contents

1 Introduction

In this work we will focus on dimension reduction and clustering. We will try here to show how clustering and dimension reduction behave in different situations. To do this we will first work on direct application of clustering in the particular case where we went to fund back known cluster. We will then do the same study but with an additional step of dimension reduction before clustering. After that we will explore how we can use dimension reduction algorithm as a first step in training a neural network as well as consider the option of adding a clustering layer to generate new features for the neural network.

1.1 Datasets

For this work we focused on two different dataset. The first is the well known Iris dataset, this dataset describe four different parameters of three different Iris variety and contain 151 different entries. We chose this dataset because of the low dimension space that it cover, allowing us to plot easily understandable 2D figures to show the results of clustering and dimension reduction algorithms. Figure 1 show the repartition of the classes on the dataset. You can see that the classes are already well defined and we can so suppose that clustering algorithm will perform well.



(a) Classes repartition according to sepal length and width

(b) Classes repartition according to petal length and width

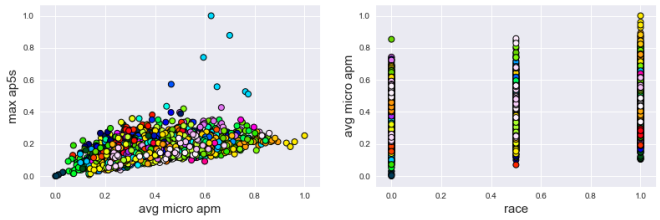
Figure 1. Repartition of the classes on Iris dataset according to different parameters

The second dataset is here again the player recognition dataset on Starcraft II. For a more in depth description please refer to Homework 1 and 2. This dataset is again very challenging here, as it as 200 classes visualization will be hard and it's 72 dimensions wont help to represent it on paper. In addition with only 3035 examples cluster are very small in average, with some having 4 elements while others have more than 50. As you can see on Figure 2 the plots are not readable, even when we select dimension which are known to easily differentiate players, such as the action per minutes (APM) or the race used. We also explored the possibility to add a third dimension, but it didn't improve the visualization here. In addition the different feature of this dataset are on very different intervals, going from 0 to 3 to 0 to 7000. As we know that such characteristic doesn't help clustering algorithm we choose to normalize all the different features.

2 Clustering

Applying clustering like K-means or Expectation Maximization (EM) to our two different dataset give good results.

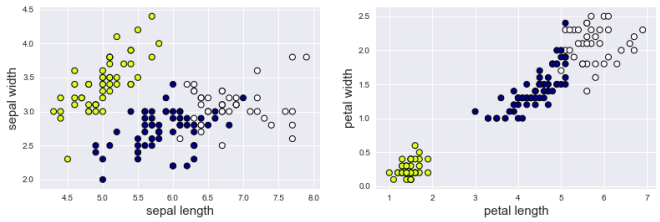
For the Iris dataset both clustering algorithms perform really well. As you can see on Figure 3, kmeans get here



(a) Classes repartition according to avg APM and max number of action per 5 seconds

(b) Classes repartition according to avg APM and race used by the player

Figure 2. Repartition of the classes on Starcraft dataset according to different parameters



(a) Classes repartition according to sepal length and width

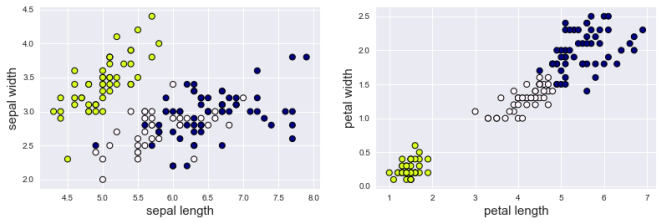
(b) Classes repartition according to petal length and width

Figure 3. Repartition of the K-Means clustering results on Iris dataset according to different parameters

a near perfect classification result. The typical potatoid shape of the kmeans clusters is easy to spot on Figure 3b, but this particular bias of kmeans is also the reason why it make a few mistakes that are easy to spot on Figure 3a. EM on it side perform even better than kmeans here. Few mistakes can be spotted on the edges between blue and white area of Figure 4b, which is not surprising as the two original cluster were very close on every dimensions. In addtion we can note that given it's nature EM let more the blue and white cluster to mix together on Figure 4a, this is due to the large space between point in this part of the cluster.

Such good results are not surprising for Iris dataset as it's really a toy dataset in terms of complexity, this was desired here to show how the algorithm behave.

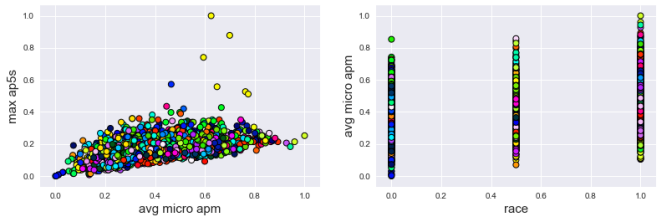
The Starcraft dataset is tricky to represent and analyses for the same reason as described in section 1.1. However, if in general both clustering did not get good results we can note that the outliers are generally classified together,



(a) Classes repartition according to sepal length and width

(b) Classes repartition according to petal length and width

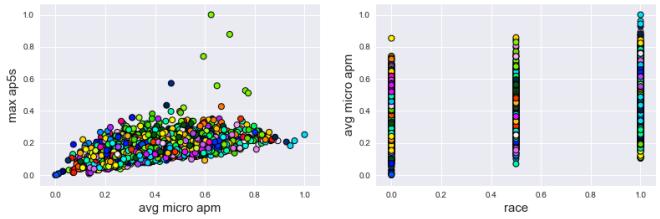
Figure 4. Repartition of the EM clustering results on Iris dataset according to different parameters



(a) Classes repartition according to avg APM and max number of action per 5 seconds

(b) Classes repartition according to avg APM and race used by the player

Figure 5. Repartition of the K-Means clustering results on Starcraft dataset according to different parameters



(a) Classes repartition according to avg APM and max number of action per 5 seconds

(b) Classes repartition according to avg APM and race used by the player

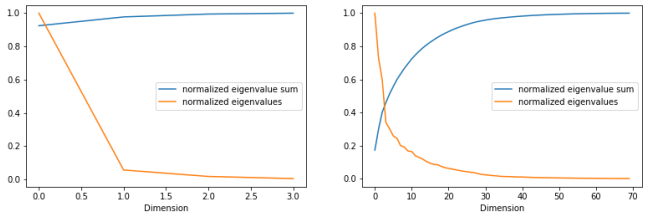
Figure 6. Repartition of the EM clustering results on Starcraft dataset according to different parameters

which is the right things to do according to Figure 2. This is easy to spot on Figures 5a and 6a. Such a comportment is not surprising as both clustering algorithms try to group similar features together, thus outliers are easily grouped while at the same time easy to spot when looking at the plots. The fact that those outliers are in the same class here allow us to suppose that they are close in others dimensions, as is they were not K-Means and EM will have chosen a different classes for each point or found link on other dimension to give result such as the one of Figure 4a.

To the light of those results we can say that clustering algorithm are really dependent on the statistical repartition of the data. Even if data are normalized, density is not constant and induce bias in the clustering algorithm. An other bias came from the redundancy of the data. For Iris, it's not really the case as it has very few dimensions which were selected by an expert and have thus their part of domain knowledge. However, for Starcraft we can assume that a lot of information is redundant as no particular filter were applied, and correlation between features such as the average APM and the max APM on 5 seconds are trivial. This redundancy is equivalent to give more weight to some features, which can be very harmful to clustering if this features are not relevant. In the next section we will thus try different algorithms which do such things, without adding any domain knowledge.

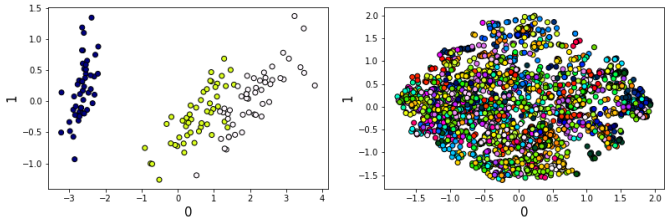
3 Dimension reduction

Reduce the number of dimension is often crucial in machine learning, particularly because the curse of dimen-



(a) Iris dataset (b) Starcraft dataset

Figure 7. Evolution of the score according to the number of dimension for both dataset



(a) Iris dataset (b) Starcraft dataset

Figure 8. Representation of the classes after projection in the best plan fund by PCA

sions. A lot of different algorithms exist to try to reduce the number of dimension to a set of meaningful one. But meaningful does not have any meaning on its own and it's probably why they are so many different ways to do it.

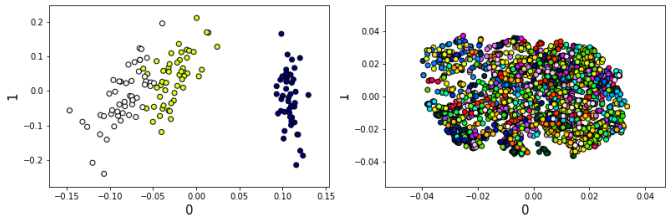
Here we will explore the results of the dimension reduction algorithms seen during the lessons: PCA, ICA, Random projections and LDA. The later is a bit particular as it make the projection knowing what you want to find, it's also why chose it, to show how giving such a crucial information change the results.

3.1 PCA

An interesting possibility with PCA is that you can easily score the dimensions chosen thanks to the associated eigenvalue. Plots representing those values for both datasets can be fund on Figure 7. On both case the eigenvalue follow an exponentially decreasing shape with the number of dimensions. Thus you can see that on Iris, according to PCA, one dimension already give a good description of the dataset while ten dimensions is still short to represent the Starcraft dataset.

As you can see on Figure 8, the projections does it job. In the Iris case, the classes really appear on different clusters. We can also understand here why axes 0 as a better score than axes 1, as here the data nearly appear like group of vertical line, which is great given that PCA didn't know the labels, but still managed to nearly split them on the first dimension.

For the Starcraft dataset the things are a bit different, if it's sure that the data are more widely spread in Figure 8b than it was before a PCA, the 200 classes still appear as noise. This is not surprising given that we try here to represent in two dimensions data that PCA say it can represent correctly in ten to twenty dimensions. It's hard to know if it's our brain that try to fit pattern or if they



(a) Iris dataset (b) Starcraft dataset

Figure 9. Representation of the classes after projection in the best plan fund by ICA

are really pattern here, but we can see that same colors are generally on similar area, which show that PCA find some consistent way to reduce the dimensionality, but that two dimensions wasn't enough.

In a clustering point of view, Iris is still an easy case. K-Means get here worst result than on the full dataset because it try to split the two cluster on the right in a horizontal way. This comportment can be countered by a different norm or data normalization, but doing so can be called supervised learning, which is not our goal here. EM get here near perfect results, probably taking more advantage of the hight density in the center of the clusters.

Starcraft suffer again from the lake of dimension, both clustering algorithms try to find cluster that does not exist. While this gave us beautiful plot of a large potatoid full of small colorful potatoid, but we did not get back the players we were looking for. Which is not really surprising when you know how the players are split (Figure 8b).

In conclusion, PCA is fast and efficiently reduce the number of dimensions while giving us nice values to help us to chose how much dimension we want.

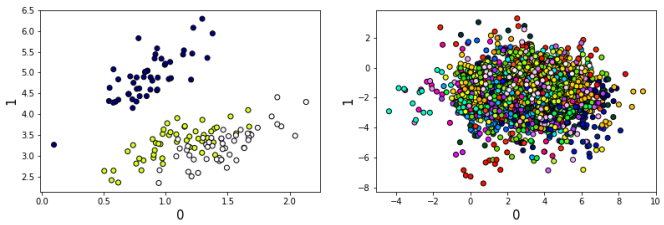
3.2 ICA

ICA look for independent underling variable and is so very used in signal processing when different signals are mixed together. In our case, it's probably possible to find underling variable to our data, the principal one being the label, but on our two cases it's hard to find other examples. This does not mean that ICA will not perform well here but that it's probably not the best algorithm we can use.

On the Iris dataset, as you can see on Figure 9a, the result are very similar than the ones of PCA (Figure 8a). If you perform a 180 degree rotation of the plot you will see that they are nearly identical. Thus, we can apply the same conclusion than for PCA here. But it's important to note that the scale are different, here the ratio of the range of axis 0 over the one of axis 1 is about 0.7, while it was 2.4 in PCA, explaining the clustering difference.

On Starcraft, The potatoid shape is quite different from PCA but it is not enough to split the different players apart, as you can see on Figure 9b. Still the result look better than for PCA, as color spot are easier to find.

On a clustering point of view, due to change of proportion both K-Means and EM wrongly split the left cluster



(a) Iris dataset (b) Starcraft dataset

Figure 10. Representation of the classes after projection in a plan by random projection

and preferred to split them horizontally. In addition K-means managed to get wrong one point of the right cluster. As described for PCA, axes 0 is here the one carrying the more information, reducing it's amplitude regarding to the one of axes 1, give far more importance to this last axis in both clustering experiments. Again one may say that this problem is easy to overcome, but doing it is not our goal here as it tend to become supervised learning with its own problem such as overfitting.

Starcraft is still difficult here for clustering algorithms. The result are very similar to the ones of PCA. As the classes are really mixed together, clustering only manage to group some sort of similar player but didn't manage to find the real player.

In conclusion, ICA gave us very similar results than PCA, probably because our data were too difficult, or too easy, to differentiate the two algorithms, but also because none of the datasets has trivial underling variables.

3.3 Randomized projections

Random projection is a very time efficient way to reduce dimension. The fact that it is random driven let us expect worst results than the other algorithms explored here.

On Iris most of the random projection we tried gave average results, not better than a plan in the original nor really worst. As you can see on Figure 10a, the global shape is familiar as very close to the one get for PCA and ICA, with one cluster far from the two others which are really close but with very few common part. This projection is far from optimal in a clustering point of view, but the cluster are still there.

The results on Starcraft are more interesting. Not that the player are well grouped together, but this projection managed to mix feature in a way that allow out-layers to stand out of the global mass. Thus we cannot expect clustering algorithm to get every classes right but they might fund so of them, which were not rely the case of both PCA and ICA.

Clustering on random projection mostly gave bad results on Iris. For example, EM chose here to group the two bottom cluster together and to split the top one. While K-Means cut the bottom group in two with a horizontal line, one of the worst chose if you want to find the original labels.

On Starcraft, the clustering algorithm unsurprisingly gave bad results. Some outliers were correctly classified but were also most of the time split in multiple clusters. The central mass was as previously too diffuse to get any thing right out of it. As previously we can say that 2 dimensions are not enough to represent those data. Particularly with random projection, which may need more dimensions to conserve the same amount of information than PCA and ICA.

Thus, random projection does not do a great job to apply clustering algorithm, but still keep the global shape of the data and can be useful for quick visualization purpose or as reference to evaluate the performance of other dimension reduction algorithms.

3.4 LDA

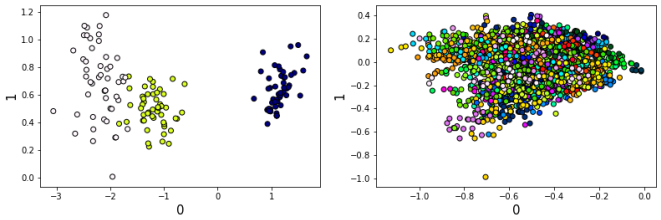
LDA is very different from the others dimension reduction algorithm explored in this report, as it is the only one to include the labels to make the best projection. This information make the whole clustering more supervised than other thing. Thus, LDA does not give the same information than other algorithms, as this one is biased toward what we are looking for. This bias is interesting here to see how this information change the results, but usually clustering is used when you do not know what you are looking for, case where LDA is useless.

The projection of Iris into two dimensions displayed on Figure 11a show particularly well split clusters. Two classes are still very close to each other, but the original data want that. We can, here, also look at the range ratio, but it will not be useful as axes 0 already give most of the information needed to correctly classify every thing, as long as this ration stay near 1, we can expect good clustering results later. So LDA project here the Iris dataset into nearly one dimension, showing that with only a linear combination of the variable we have enough information to find all the classes of this dataset.

The best projection fund by LDA for Starcraft is a bit different in term of shape from the one seen previously. Me must also note than players seems to be more close to each others than before, but players are still mixed together. So even when including what we are looking for, we are not able to linearly combine dimensions to project data to a plan where clustering algorithms will be able to manage them. Knowing that it is possible to get 90% validation accuracy on this problem we can deduce that, ether we need more dimension or linear combination is not sufficient here.

As already stated for Iris, the clustering after this projection is stretch forward. Both EM and K-Means does near perfect classification here in terms of original label. Both of them with some errors in the frontier of the two clusters, but in a smaller way than what we have seen before. They do not get confused by the second dimension probably because it range on a smaller interval than the first one.

For Starcraft, it's difficult to say if the result are better or worse than before. We can probably say that they are



(a) Iris dataset (b) Starcraft dataset

Figure 11. Representation of the classes after projection in the best plan fund by LDA

as worse as in all the previous cases. If you can see, on Figure 11b, that, globally, all the games of a player tend to be on the same region, they are still too mixed with other players to allow clustering algorithm to find them.

In conclusion LDA does a great job on Iris but doesn't do miracles on Starcraft. In addition for this little improvement with regard to other algorithm such as PCA and ICA, LDA come with a large cost, the one of the labels. It is so not used in the same cases than the previous ones. While the tree others algorithms are very useful when you look for something in the data but not knowing exactly what, LDA is useful when you know what you are looking for and just want to reduce the dimensionality before running classification algorithm for example.

3.5 Conclusion

In this section we have seen that applying dimension reduction before clustering can be very efficient to compress the data while keeping them as meaningful as possible. This was really useful here for visualization purpose, which is not negligible when the main question is describe what you see, but can also be very useful to limit the effect of the curse of dimensionality.

Run time were not really discussed before, mainly because most of them are not significant. All the dimension reduction algorithm does there work in less than a few seconds, it's also the same for K-means, but EM is far slower taking about a minute to complete all the 30 runs asked (all of them being initialized by running a K-Means). Thus if EM generally perform better than K-Means, their timing difference probably explain why K-Means is still very popular.

We chosen to keep Starcraft dataset here to have an example of a case were both clustering and dimension reduction were not capable of producing exploitable data. We thus wanted to explore what kind of results we can expect from such high dimension data.

As a comparison, we also tried to apply our network from homework 1 to Starcraft with a layer of only two neurones to force it to do a dimension reduction. On all the possibility tested, the best validation accuracy was lower than 7%. This result is an other indication that lot of information are lost by reducing the dimension to two.

To improve this report we also tried to display Starcraft on three dimensions plots, but those plots were even less readable than the two dimensions one. In addition if human eyes can distinguish millions of colors, we did not fund a good way to make all the players perfectly distinguishable on the figures, while this is not crucial for our conclusion, we are sorry for the inconvenience.

We also apologies for the lack of figures in this section, but due to space constraint we have to do some sacrifice. Thus the plots of the different cluster were not included here as easy to imagine from the projections.

4 Training a Neural Network after dimension reduction

Reducing dimension before trying to learn a classifier such as a neural network (NN) is a good solution to limit the effect of the curse of dimensions. One may expect that reducing the number of dimension will reduce the convergence time of a neural network, but this may not be true as compressing the information may make it harder for a NN to find useful way to use it.

To properly asses that question, among others, we will use here the same network structure from Homework 1. In this structure, we set most of the parameters explored in this previous work. For example, we used batch normalization in all our experiment, as well as the sigmoid activation function and the rmsprop optimizer from Keras. Thus, the reader can easily compare the results with the one presented previously.

We will also stick to the same graphical convention as previously: the hard line represent the median value while the hue area go from the minimum to the maximum.

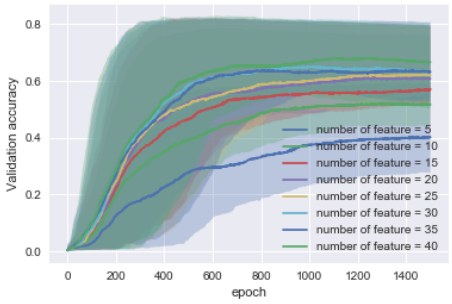
The same convention on the description of the NN structure will also hold. (n) describe a perceptron with one hidden layer of n neurones and (a, b) a perceptron with two hidden layers of a and b neurones respectively.

If the part of the structure of the NN stay the same than in Homework 1, we still explored here a different set of parameters. The most important among them are number of features, the dimension reduction algorithm and the shape of the hidden layers. This last parameters cover the same range as in Homework 1, to keep comparable results. The dimension reduction was first fit on the train dataset and then applied to both train and test data. Repartition between train and test reminded the same than in assignment one, please refer to it for more information.

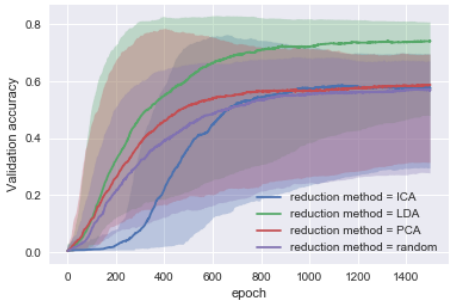
Before going more in depth, algorithm by algorithm, we will draw here some global trends.

As you can see on Figure 12b, the different reduction methods does not gave the same results. However, we must note that, apart from the particular case of LDA, the median result of all the algorithm converge toward the same accuracy. It also worth mentioning that ICA gave some hard time to the NN with first improvements appearing far later than others methods.

It was also easy to predict that the more feature you give the better the result. Such a comportment can be seen on Figure 12a, where you can see that there is a clear progression from 5 to 20 features but that the evolution is

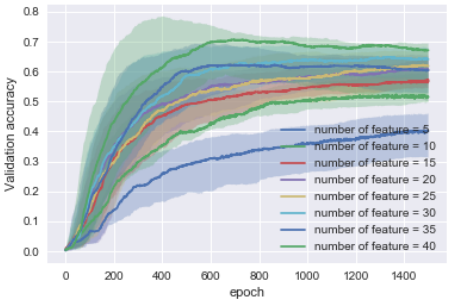


(a) Effect of the number of feature

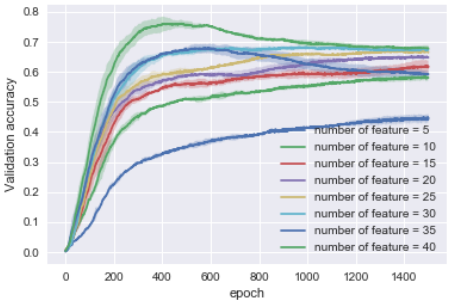


(b) Effect of the reduction method used

Figure 12. Validation accuracy according to the epoch for all the different combination



(a) Effect of the number of features for all tested layers



(b) Effect of the number of features for (35) and (25) layers

Figure 13. Validation accuracy according to the epoch with PCA

then far smaller. This is particularly truth for the maximum, which have nearly all the same values. Thus, as a first impression we can say that 20 features is enough for a NN to learn how to classify the different players. We must also note that adding new feature does not augment the number of epoch needed by the NN to converge. This is probably because with more features it's easier for gradient descant to find one to focus on.

4.1 PCA

PCA is the best performer here, after LDA. As you can see on Figure 13a, the global variation observed previously with the number features perfectly fit here. It is also hard not to notice the pick near epoch 400 for 40 features. By selecting only a subset of layers we can see that, as displayed on Figure 13b, this pick is created by layers (35) and (25), that are able to take advantage of the extra features to improve their accuracy. The interesting fact here is that this advantage is quickly transformed into overfitting. Overfitting also occur for 35 features, with accuracy in that case never exceeding the one get with 30 features going as low as the one with only 10 features.

Adding new input features rise the number of parameters in the network, which also rise it chance to overfit. This is also know as the curse of dimensions. The lack of different examples here probably forbid the NN to properly integrate the new features letting it overfit on the train set.

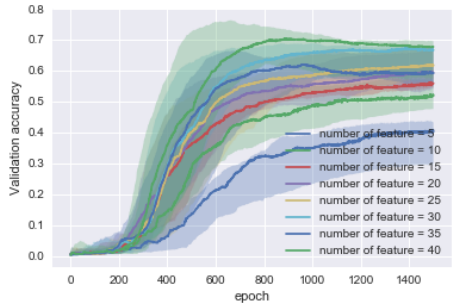
In terms of layers the results are here very similar to the one of the first homework. The different numbers of features given by PCA does not significantly improve the result of a given layer, and it is the same the other way around. It is important here to empathize that, if the number of features does not change the convergence speed, the size of the layers do. The validation accuracy of small layers is rising slower than the one of layers with more neurones. This is probably because, having more weights, the bigger layers are easier to fit into a local optimum. So if you want to approximate a complex function it's easier with lot of simple one than few complex one, even if the later is probably more reliable.

In conclusion PCA allow a perceptron to learn good results on fewer features. But, those results are not as good as the one obtained on the original dataset and training is slower in terms of epochs. But we are here in a case with enough examples to manage the curse of dimension, in a case where we are not, PCA look like a good choice to reduce its effects.

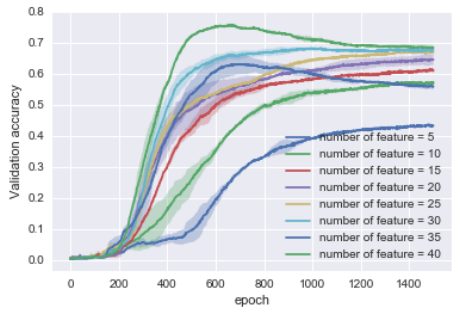
4.2 ICA

The first things you probably note on Figure 12b, is that ICA give features that make the perceptron slower to converge. This behavior is common to all the combination explored here, as you can see on Figure 14. Thus we suppose that the way ICA split features is not an easy one to understand for a neural network, even if the valuable information is still present as it finally get a similar median accuracy than both PCA and random projections.

We can also notice that as with PCA, having 35 or 40 features give overfitting, for every layers tried here. The explanation given on the PCA case is still relevant on this case.

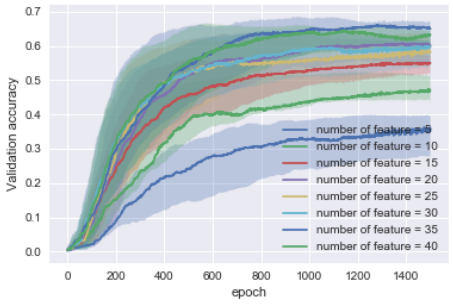


(a) Effect of the number of features for all tested layers

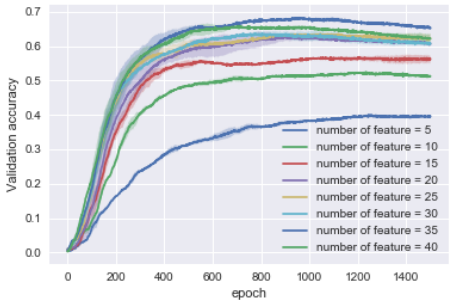


(b) Effect of the number of features for (35) and (25) layers

Figure 14. Validation accuracy according to the epoch with ICA



(a) Effect of the number of features for all tested layers



(b) Effect of the number of features for (35) and (25) layers

Figure 15. Validation accuracy according to the epoch with random projections

In terms of accuracy according to the number of feature, ICA give here very similar results than PCA once convergence is reached. Although the maximum validation accuracy reached by ICA is smaller than the one on PCA, while the two were run with the same set of parameters.

Thus, ICA give good median results after convergence, but those results are not as good as PCA and convergence need a lot more epochs than for this last algorithm. So, we consider that ICA is not really relevant in our case, probably because they are not really hidden variable here to discover.

4.3 Randomized projections

As we already said on this report, randomized projections gave surprisingly good results given that it is only random. This conclusion is also true here, as you can see on Figure 12b, this projection algorithm give as good results as PCA and ICA on a median basis. But as you can see on Figure 15a and 15b, the maximum results are not as good as for the other algorithms and adding more feature does not necessary end up giving better results. Thus, we get here better result with 20 features than with 30 as well as 35 and 40. This is the result of being random, but the general trend observed before is still true, the more you add features the better the accuracy is.

This random projection are also not as easy to learn from as PCA. The improvement of validation accuracy on the very first steps is small, but this is negligible in comparison to the one with ICA.

We must also highlight that their is no major overfitting occurring here, at the contrary of ICA and PCA. This show that overfitting is not only dependent on the shape of the data and of the network but is also highly dependent on the content of the data. So, we can assume here that random projection did not produce features that were particularly relevant on the train dataset but were completely uncorrelated to the class in the test dataset. But as the results show, this come with the cost of a lower accuracy.

In conclusion, random projection is really efficient when you consider that it is only random and thus, very fast. But it did not allow as good results as PCA and ICA, particularly when looking for the best ones. But we must emphasize that it is particularly powerful when you need to cut a lot of dimensions, in such case it even sometimes manage better results than PCA and ICA.

4.4 LDA

LDA is completely in an other world than the previous reduction dimension algorithms. First because of its supervised approach, in which you say what information you want to keep. As discussed before, this strategy is often more a problem than a solution, but here we are exactly in the case where their are big benefit to use it. Second because of its results, as you can see on Figure 12b, LDA is by far the best performer here and get result as good as the one get during homework 1, with the full dataset.

The first thing you probably notice on Figure 16a is that, apart from always failing 5 feature case, no matter

the number of feature you add, you only gain very little accuracy. The only variation occurring here is due to the different layers, but as in the previous case no layers take more advantage than another of this particular data. Thus we again fund (35) and (25) as the best performer. From this result we can things that this dataset is can probably be condensed in about ten dimensions and still contain enough information to find the original player, with the same accuracy than with the 72 original dimensions. But that does not say that we did not loss any information, we just kept the ones pertinent in that case. LDA is the only case were the perceptron was able to reach the 100% accuracy on training set with as few as 10 features. If one can consider that going there only result in overfit, this show that those dimensions has all the informations needed.

It is this information on labels that give LDA such a pertinence here. As every supervised algorithm, LDA can overfit and loss its generalization capabilities. Here we trained both LDA and the perceptron on a dataset and computed accuracy on another one, and so we are pleased to see that the transformation learn by LDA is still pertinent on our test dataset.

As we stated before, thanks to LDA, the perceptron quickly achieve an accuracy of 100% on training data. It is then not very surprising to see the NN overfit, but this overfitting is quite slower than the one seen in PCA or ICA. Probably because the dimensions chosen by LDA are far more generalizable to the test dataset.

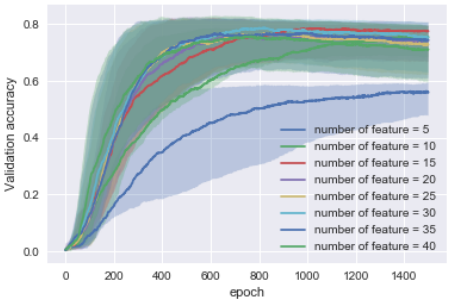
In addition, in terms of epoch, the training time of the perceptron is equivalent to the one get for PCA. This timing is also similar to the one get in homework one, and we can suppose that it will be hard to be faster than that while reducing the number of dimensions.

We must admit that we mostly chosen LDA in this report for this particular section and were pleased by such good results. LDA seams to perfectly fit to its task here, it allow to quickly reduce the number of dimensions to give to a learning method, which can help to improve its results. Depending on its structure a perceptron can be do feature reduction in an internal layer, but this dimension reduction is not as efficient as the one we have shown here. We never managed to get a perceptron to have anything like similar to the results we get here when it had a layer of ten neurones. Showing the importance of dimension reduction algorithm.

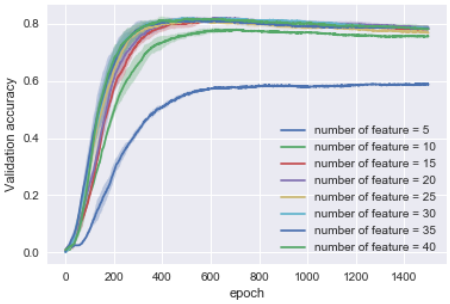
4.5 Conclusion

According to the results of this section, we can say that the learning capability of a NN depend more on the data than on the internal structure. If we have shown in homework 1 that some choice of architecture can completely destroy the accuracy, the size of the layers does not have as much importance as the information available in the data.

We can also add that the curse of dimensions is certainly linked with the number of input features to a NN, but as we saw with ICA, the hidden dimensions of the information also has its importance. Particularly we showed that



(a) Effect of the number of features for all tested layers



(b) Effect of the number of features for (35) and (25) layers

Figure 16. Validation accuracy according to the epoch with random projections

it is hard for a NN to understand that a feature is just noise and has no relation with a label.

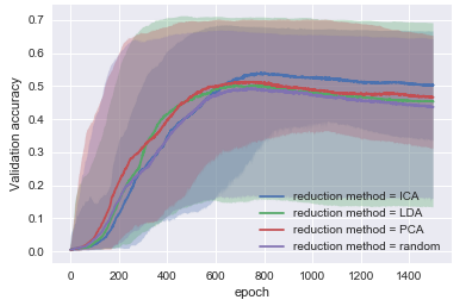
5 Adding cluster as labels

In this last section we will study the effect of adding the results of the clustering algorithm as new features to input into the perceptron. It is hard in our case to consider clustering as a dimension reduction algorithm as we have 200 classes for a maximum of 72 original feature, still adding this information can provide additional information to the perceptron, particularly if some cluster are highly correlated to a given class. In the same time adding 200 features will probably not help the NN to find the best way to do such links. In addition the results of section 3 make us doubt such an improvement, as clustering do not look pertinent on our experiment.

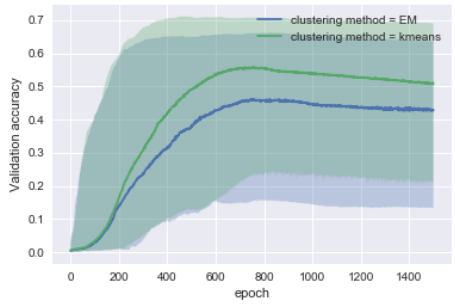
As a global trend, you can see on Figure 17, that the best results are significantly lower than one the previous section, with a maximum accuracy around 70% while it was above 80% in section 4, thanks to LDA. Another major change is that every reduction method give very similar results, on a median value perspective as well as on a maximal value.

We must also note on Figure 17b, that while we preferred the results of EM over the one of K-Means in section 3, here K-Means allow better results than EM no matter the metric you use. Which is probably a good news as K-Means run much faster than EM.

On Figure 17a, you can note that in a median basis,



(a) Effect of the reduction method on validation accuracy



(b) Effect of the clustering method on validation accuracy

Figure 17. Validation accuracy according to the epoch When cluster are added as new features

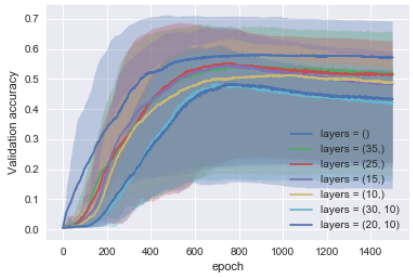


Figure 18. Effect of the clustering method on validation accuracy for different layers

all the reduction method have difficulties to start learning something useful, while in previous section this case only show up for ICA. At the same time the maximum show that at least a set of parameters do not follow this rule. The parameter responsible for such a change was easy to find, its effect are displayed of Figure 18. As you can see the layers are by far the most important parameters here, with the one connecting directly input to output neurones getting the best results. This is not surprising as all the others try to reduce the number of dimension in their intern layers. We must also stress that () is the layer which has the more parameters, with ten times more of them than training examples, explaining the overfitting. Even if, it appear to be stronger in the others layers, on which it probably occur because of their dimension reduction component, forcing the NN to chose some input to weight more than others.

5.1 PCA

PCA follow here the general trend, in such a way that we can qualify it as the median dimension reduction algorithm here. Thus it follow all general observation discussed above and give only little advantage to K-Means over EM compared to most of the other algorithms.

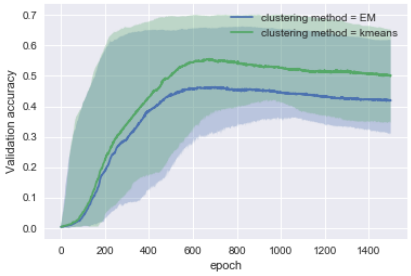


Figure 19. Effect of the clustering method on validation accuracy when using PCA

5.2 ICA

ICA is again very particular. As show on Figure 20, it is the only case where both K-Means and EM have very similar results, even if K-Means is still a little above EM. ICA is also the only case where layer () do not manage to converge on the first epochs. This probably come from irrelevant clusters as we observed on section 3.2 or simply because the features are difficult to understand as discussed on section 4.2.

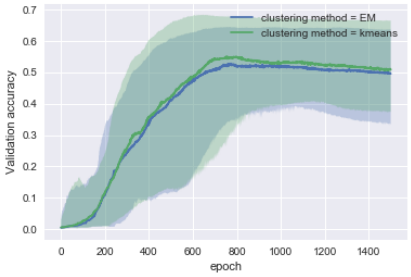


Figure 20. Effect of the clustering method on validation accuracy when using ICA

5.3 Randomized Projections

Random projections is the most discriminant algorithm we have here when it come to compare EM and K-Means. While with EM the perceptron still does better than chance, the median result show that it predict the wrong class 2 times over 3. Which must be compared to the results get on section 4.3, where the results were not the best, but still far better than this case, which only has more informations as input. On a K-Means point of view, the results are very similar to the one of the other algorithms, showing once again that random projection may look stupid but is still very efficient particularly with regard to its complexity.

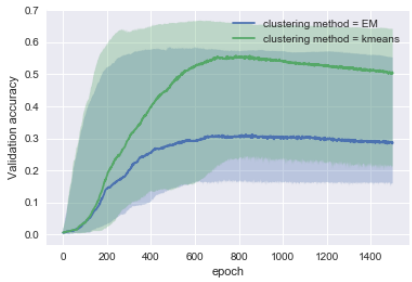


Figure 21. Effect of the clustering method on validation accuracy when using random projections

5.4 LDA

The surprising fact with LDA is the difference between K-means and EM, which is close to be as important as the one observed for random projection. This is again very surprising given that without this additional cluster information the perceptron manage to get great results. We suppose that this huge difference is because LDA try to split the data in a K-Means pertinent way that do not take into account the probabilistic vision of EM.

Another important fact is that again this new information confuse more than help the NN.

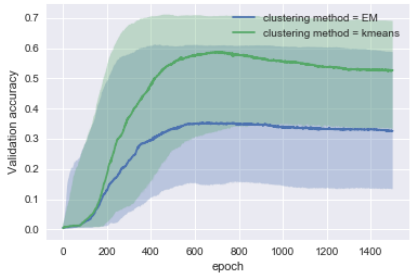


Figure 22. Effect of the clustering method on validation accuracy when using LDA

5.5 Conclusion

We can thus say that adding the result of a clustering algorithm as features for a neural network does not help it to get better results, at least in our specific case. An experience that would have been interesting here, is to replace the computed cluster by the wanted classes to see if the problem come from the structure of the perceptron or from the clusters.

One explanation of the results we get here is that the cluster generated on the train dataset give good insight to find the desired class, which are used by the perceptron, but that those cluster does not extend to the test data set and are in a way an overfit of the train data. This conclusion is strengthened by the fact that accuracy on the train data set converge far quicker toward perfect results on this section than on section 4. So, we assume that the perceptron take advantage here to learn more efficiently, but that it ether overfit on the train data or the cluster information is not coherent from a dataset to the other.

It would have been interesting to also explore other machine learning methods here as for examples random forest. Which are known to be more efficient with large number of features, in addition to show very good results on the original dataset.

6 Conclusion

This work allowed us to discover the advantages and drawback of dimension reduction and clustering algorithms, as well as get a first experience with them.

The first section showed us that it is very difficult to interpret clustering results, even when you know what you are looking for. This interpretation, mostly rely on graphical representation which is difficult to do in more than three dimensions. Interpret data with more than four dimensions can be very difficult, especially when you do not know what you are looking for, as in data mining.

We learned, then, that dimension reduction can be a very powerful tool when used in the right range. The difficulty being to find that range, with the same problem of visualization stated previously.

In section 4, we explored the possibility given by the dimension reduction to train a neural network. Study which allowed us to learn that even if in a theoretical point of view it is better to have fewer dimensions as input, loss of information in the input is often worse in term of validation accuracy. This exploration also show us, once more, that the representation used for the data given to a neural network is very important and can make it very difficult for a NN to learn something.

We finally discovered that clustering on our dataset was not stable enough to generate additional features to train our perceptron. But learned that such approach can be useful in some situations.

Dimensions reduction was something we have already heard of, but not properly use before this semester. It was very interesting for us to learn more about them, in particular to experiment with them, which will probably allow us to use them when needed.

This is far less true for clustering algorithm, as we already had some exposition to it previously. If this assignment can be considered a bit light on the cluster side, it completely fit with to the goal of the courses to our mind. As we had less to learn on that side, we may have gone over this subject a bit too quickly. If so, please accept our apologies.

Our goal here was not to find the best parameters possible to have good results. While we still tried to chose coherent parameters, we mainly wanted to study the effects of different parameters for our problems. In order to learn how the different algorithm react to them. In the goal to be able to do correct choice when we will have to use similar algorithm in the future. If this was not the goal of this homework, it was at least the one of our work.