Assessment of the applicability of dimensionality reduction techniques on seizure prediction

Yogatheesan Varatharajah
Department of Electrical and Computer Engineering,
University of Illinois, Urbana, Illinois 61801-29180
Email: varatha2@illinois.edu

Abstract

In this article, I present an assessment of dimensionality reduction techniques and their applications in the classification of Electroencephalography(EEG) signals collected during Canine epileptic seizures. The data tokens contain a large number of features and therefore making dimensionality reduction necessary. I have tried three dimensionality reduction techniques namely, Principal Component Analysis(PCA), Independent Component Analysis(ICA) and Restricted Boltzmann Machine(RBM) auto-encoder. To compare the performances of these techniques, I have trained a two-layer neural network on the features generated by these techniques and used simulated annealing to find the optimal weights of the neural network. The comparison of these techniques is made in terms of the running times and the confusion matrices produced by the classifier. A comparison proves that doing ICA on top of PCA outperforms other techniques by a significant margin.

Article organization

Section 1 gives a brief introduction to the problem, the dataset and the motivation towards the attempt to reduce dimensionality. Section 2 gives an overview to the dimensionality reduction techniques that were used in the project, namely PCA, ICA and RBM auto-encoders. Section 3 explains the setup used for evaluation, the methods of evaluation and the results obtained for each of the techniques. Finally, Section 4 gives some conclusive comments to this assessment.

1 Introduction

Detection of seizures is an important task in bio-medical engineering. A large amount of literature exist for this task and there has been significant milestones achieved. However, the task at my hand is slightly different from a conventional seizure detection challenge. Here, I am trying to do a prediction instead of a detection, i.e. I am looking to find signatures that exist in EEG signals prior to a seizure. The importance of predicting seizures does not need to be explained. It helps the medical personnel to take preventive measures towards avoiding the seizure instead of treating the seizure. This is a fairly new concept and is an active area of research.

1.1 Dataset

The data I have used for this project was obtained from an online machine learning competition[1]. The data contains of Intracranial EEG (iEEG) data clips of canine subjects. The data is organized

into ten minute EEG clips representing the EEG activity of either a very normal condition or a pre-seizure condition from 16 different channels each sampled at 400Hz. Figure 1, shows how the EEG signals change over a period of transition from a normal condition to a seizure. Clips are

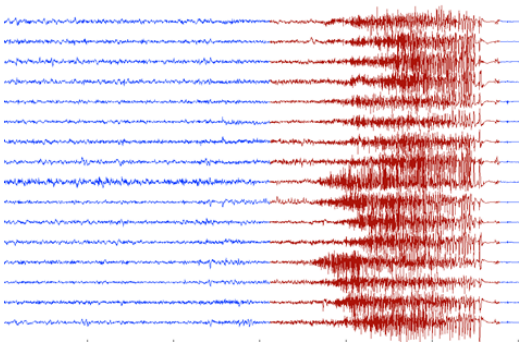
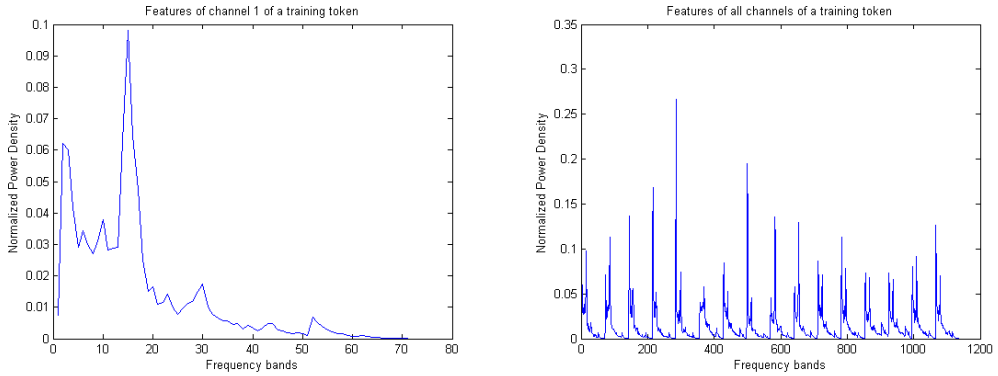


Figure 1: EEG signals corresponding to a normal condition transforming into a seizure

labeled "Pre-ictal" for pre-seizure data segments or "Inter-ictal" for non-seizure data segments. In total, there are 732 EEG clips each of length 10 minutes. The data comes pre-labled and this is purely a supervised learning task. For the sake of making this learning problem more feasible and convenient, I have broken each 10 minute clip down into ten 1 minute clips and labeled them all as that of the original 10 minute clip. This way, I got more training tokens and also reduced the number of features to deal with.

1.2 Feature Extraction - Motivation

For each channel, the frequency spectrum can be estimated up to 200Hz. I have used powers in frequency components as features. To make each of the frequency to have a power in the same range as others, I have calculated the powers in bands. For the first 50Hz, power is calculated at each frequency component. For the next 50Hz, power is calculated at 5Hz bands. And for the next 100Hz, power is calculated at 10Hz bands. So, all-together, a channel will end up having 71 features and a clip will end up having $16 \times 71 = 1136$ features. Figure 2 show the features generated for a data token, where Figure 2a shows the features of a channel and Figure 2b shows the features of all channels as a long vector.



(a) Features of Channel 1

(b) Features of all channels

Figure 2: Features obtained for a training token

Clearly, there is a lot of repetition in the second figure. So, the number 1136 can be reduced to a smaller number. It is arguable that the features can be chosen more intelligently so that the number of features is small. However, it requires a deep context based knowledge and that was infeasible to acquire during the timeline of this class project. Therefore, I have decided to use dimensionality reduction techniques to reduce the number of features per data token.

2 Dimensionality reduction

Advances in data collection and storage capabilities during the past decades have led to an information overload in most sciences. Researchers working in domains as diverse as engineering, astronomy, biology, remote sensing, economics, and consumer transactions, face larger and larger observations and simulations on a daily basis. Such datasets, in contrast with smaller, more traditional datasets that have been studied extensively in the past, present new challenges in data analysis. Traditional statistical methods break down partly because of the increase in the number of observations, but mostly because of the increase in the number of variables associated with each observation. The dimension of the data is the number of variables that are measured on each observation.

High-dimensional datasets present many mathematical challenges as well as some opportunities, and are bound to give rise to new theoretical developments. One of the problems with high-dimensional datasets is that, in many cases, not all the measured variables are "important" for understanding the underlying phenomena of interest. While certain computationally expensive novel methods can construct predictive models with high accuracy from high-dimensional data, it is still of interest in many applications to reduce the dimension of the original data prior to any modeling of the data.

In mathematical terms, the problem we investigate can be stated as follows: given the p-dimensional random variable $x = (x_1, \cdots, x_p)^T$, find a lower dimensional representation of it, $s = (s_1, \cdots, s_k)^T$ with $k \leq p$, that captures the content in the original data, according to some criterion. The components of s are sometimes called the hidden components.

Throughout this paper, we assume that we have n observations, each being a realization of the p-dimensional random variable $x = (x_1, ..., x_p)^T$ with mean $E(x) = \mu = (\mu_1, ..., \mu_p)^T$ and covariance matrix $E[(x - \mu)(x - \mu)^T] = \Sigma_{p \times p}$. We denote such an observation matrix by $X = \{x_{i,j} : 1 \leq i \leq p, 1 \leq j \leq n\}$. If μ_i and $\sigma_i = \sqrt{\Sigma_{i,i}}$ denote the mean and the standard deviation of the i_{th} random variable, respectively, then we will often standardize the observations $x_{i,j}$ by $(x_{i,j} - \mu_i)/\sigma_i$, where $\mu_i = \frac{1}{n}\sum_{j=1}^{n} x_{i,j}$, and $\sigma_i = \frac{1}{n}\sum_{j=1}^{n}(x_{i,j} - \mu_i)^2$.

2.1 Principal Component Analysis(PCA)

In essence, PCA seeks to reduce the dimension of the data by finding a few orthogonal linear combinations (the PCs) of the original variables with the largest variance. Given unit vector u and a point x, the length of the projection of x onto u is given by $x^T u$, i.e., if $x(i)$ is a point in our

dataset, then its projection onto u is distance $x^T u$ from the origin. Hence, to maximize the variance of the projections, we would like to choose a unit-length u so as to maximize:

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}(x^{(i)^T}u)^2 &= \frac{1}{n}\sum_{i=1}^{n}u^T x^{(i)}x^{(i)^T}u \\ &= u^T(\frac{1}{n}\sum_{i=1}^{n}x^{(i)}x^{(i)^T})u\end{aligned}$$

We easily recognize that the maximizing this subject to $\| u \|_2 = 1$ gives the principal eigenvector of $\Sigma = \frac{1}{n}\sum_{i=1}^{n}x^{(i)}x^{(i)^T}$, which is just the empirical covariance matrix of the data.
To summarize, we have found that if we wish to find a 1-dimensional subspace with with to approximate the data, we should choose u to be the principal eigenvector of Σ. More generally, if we wish to project our data into a q-dimensional subspace $(q < p)$, we should choose $u_1, \cdots, u_q$ to be the top q eigenvectors of Σ. The u_is now form a new, orthogonal basis for the data.
Then, to represent $x^{(i)}$ in this basis, we need only compute the corresponding vector

$$y(i) = \begin{pmatrix} u_1^T x^{(i)} \\ u_2^T x^{(i)} \\ \vdots \\ u_q^T x^{(i)} \end{pmatrix} \in \mathcal{R}^q \tag{1}$$

Thus, whereas $x^{(i)} \in \mathcal{R}^p$, the vector $y^{(i)}$ now gives a lower, q-dimensional, approximation for $x^{(i)}$.

2.2 Independent Component Analysis(ICA)

To rigorously define ICA, we can use a statistical "latent variables" model. Assume that we observe n linear mixtures $x_1, \cdots, x_n$ of n independent components

$$x_j = a_{j1}s_1 + a_{j2}s_2 + \cdots + a_{jn}s_n \tag{2}$$

In the ICA model we assume that each mixture x_j as well as each independent component s_k is a random variable instead of a proper time signal. The observed values x_j are then a sample of this random variable. Without loss of generality, we can assume that both the mixture variables and the independent components have zero mean. If this is not true, then the observable variables x_i can always be centered by subtracting the sample mean, which makes the model zero-mean.
It is convenient to use vector-matrix notation instead of the sums like in the previous equation. Let us denote by x the random vector whose elements are the mixtures $x_1, \cdots, x_n$ and likewise by s the random vector with elements $s_1, \cdots, s_n$. Let us denote by A the matrix with elements a_{ij}. Using this vector-matrix notation, the above mixing model is written as

$$x = As \tag{3}$$

Sometimes we need the columns of matrix A; denoting them by a_j the model can also be written as

$$x = \sum_{i=1}^{n} a_i s_i \tag{4}$$

The statistical model in Eq3 is called independent component analysis or ICA model. The ICA model is a generative model, which means that it describes how the observed data are generated by a process of mixing the components s_i. The independent components are latent variables meaning that they cannot be directly observed. Also the mixing matrix is assumed to be unknown. All we observe is the random vector x and we must estimate both A and s using it. This must be done under as general assumptions as possible. The starting point for ICA is the very simple assumption that the components s_i are statistically independent. Statistical independence is defined in several ways. Depending upon the definition of independence, different approaches exist to estimate the matrix A.

- Non-gaussianity approach
- Minimum mutual information approach
- Maximum likelihood approach

Then after estimating the matrix A, we can compute its inverse, say W and obtain the independent component simply by:

$$s = Wx \tag{5}$$

2.2.1 Maximum likelihood approach

The algorithm we describe is due to Bell and Sejnowski, and the interpretation we give will be of their algorithm as a method for maximum likelihood estimation. (This is different from their original interpretation, which involved a complicated idea called the infomax principal, that is no longer necessary in the derivation given the modern understanding of ICA.)
We suppose that the distribution of each source s_i is given by a density p_s, and that the joint distribution of the sources s is given by

$$p(s) = \prod_{i=1}^{n} p_s(s_i) \tag{6}$$

Note that by modeling the joint distribution as a product of the marginal, we capture the assumption that the sources are independent. Using our formulas from the previous section, this implies the following density on $x = As = W^{-1}s$:

$$p(x) = \prod_{i=1}^{n} p_s(w_i^T x) \cdot \mid W \mid \tag{7}$$

All that remains is to specify a density for the individual sources p_s.
Recall that, given a real-valued random variable z, its cumulative distribution function(cdf) F is defined by $F(z_0) = P(z \leq z_0) = \int_{-\infty}^{z_0} p_z(Z)dZ$ Also, the density of z can be found from the cdf by taking its derivative: $p_z(z) = F'(z)$.
Thus, to specify a density for the s_is, all we need to do is to specify some cdf for it. A cdf has to be a monotonic function that increases from zero to one. Following our previous discussion, we cannot choose the cdf to be the cdf of the Gaussian, as ICA doesnt work on Gaussian data. What well choose instead for the cdf, as a reasonable "default" function that slowly increases from 0 to 1, is the sigmoid function $g(s) = 1/(1 + e^{-s})$. Hence, $p_s(s) = g'(s)$.

The square matrix W is the parameter in our model. Given a training set $x^{(i)}; i = 1, \cdots, m$, the log likelihood is given by

$$l(W) = \sum_{i=1}^{m}(\sum_{j=1}^{n} \log g'(w_j^T x_(i)) + log|W|) \tag{8}$$

We would like to maximize this in terms W. By taking derivatives and using the fact(from the first set of notes) that $\nabla_W |W| = |W|(W^{-1})^T$, we easily derive a stochastic gradient ascent learning rule. For a training example $x^{(i)}$, the update rule is:

$$W := W + \alpha(\begin{pmatrix} 1 - 2g(w_1^T x_(i)) \\ 1 - 2g(w_2^T x_(i)) \\ \vdots \\ 1 - 2g(w_n^T x_(i)) \end{pmatrix} x^{(i)^T} + (W^T)^{-1} \tag{9}$$

where α is the learning rate.
After the algorithm converges, we then compute $s^{(i)} = Wx^{(i)}$ to recover the original sources.

2.3 Restricted Boltzmann Machine(RBM) auto-encoder

(This overview is based on the guide for RBM by Geoffrey Hinton [3].)

2.3.1 An overview of Restricted Boltzmann Machines

Consider a training set of binary vectors. The training set can be modeled using a two-layer network called a "Restricted Boltzmann Machine" [6, 2, 4] in which stochastic, binary inputs are connected to stochastic, binary feature detectors using symmetrically weighted connections. The inputs correspond to "visible" units of the RBM because their states are observed; the feature detectors correspond to "hidden" units. A joint configuration, (v, h) of the visible and hidden units(as depicted in Fig 3 has an energy[5] given by:

$$E(v, h) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{10}$$

where v_i, h_j are the binary states of visible unit i and hidden unit j, a_i, b_j are their biases and w_{ij} is the weight between them. The network assigns a probability to every possible pair of a visible

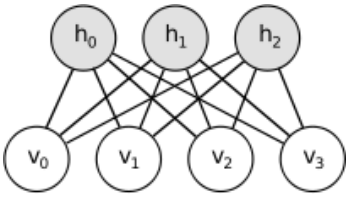


Figure 3: Restricted Boltzmann Network with configuration (v, h)

and a hidden vector via this energy function:

$$p(v, h) = \frac{1}{Z} e^{-E(v,h)} \tag{11}$$

where the "partition function", Z is given by by summing over all possible pairs of visible and hidden vectors:

$$p(v,h) = \sum_{v,h} e^{-E(v,h)} \tag{12}$$

The probability that the network assigns to a visible vector, v, is given by summing over all possible hidden vectors:

$$p(v,h) = \frac{1}{Z} \sum_{h} e^{-E(v,h)} \tag{13}$$

The probability that the network assigns to a training image can be raised by adjusting the weights and biases to lower the energy of that image and to raise the energy of other images, especially those that have low energies and therefore make a big contribution to the partition function. The derivative of the log probability of a training vector with respect to a weight is surprisingly simple.

$$\frac{\partial \log p(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \tag{14}$$

where the angle brackets are used to denote expectations under the distribution specified by the subscript that follows. This leads to a very simple learning rule for performing stochastic steepest ascent in the log probability of the training data:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}) \tag{15}$$

where ϵ is a learning rate.
Because there are no direct connections between hidden units in an RBM, it is very easy to get an unbiased sample of $\langle v_i h_j \rangle_{data}$. Given a randomly selected training image, v, the binary state, h_j , of each hidden unit, j, is set to 1 with probability

$$p(h_j = 1|v) = \sigma(b_j + \sum_i v_i w_{ij}) \tag{16}$$

$$p(v_i = 1|h) = \sigma(a_i + \sum_j h_j w_{ij}) \tag{17}$$

Getting an unbiased sample of $\langle v_i h_j \rangle_{model}$, however, is much more difficult. It can be done by starting at any random state of the visible units and performing alternating Gibbs sampling for a very long time. One iteration of alternating Gibbs sampling consists of updating all of the hidden units in parallel using equation 16 followed by updating all of the visible units in parallel using equation 17.

2.3.2 Auto-encoders

Once we have obtained the minimum energy Boltzmann configuration, we can build an auto-encoder as depicted in Fig 4. Both the encoder and the decoder parts in the figure will be replaced by the trained Boltzmann weights and a version of Gradient Descent will be used to further fine-tune the weights. When we have lesser number of nodes in the middle layers compared to the input and output layers, this auto-encoder gives us a dimensionality reduced representation of the input.

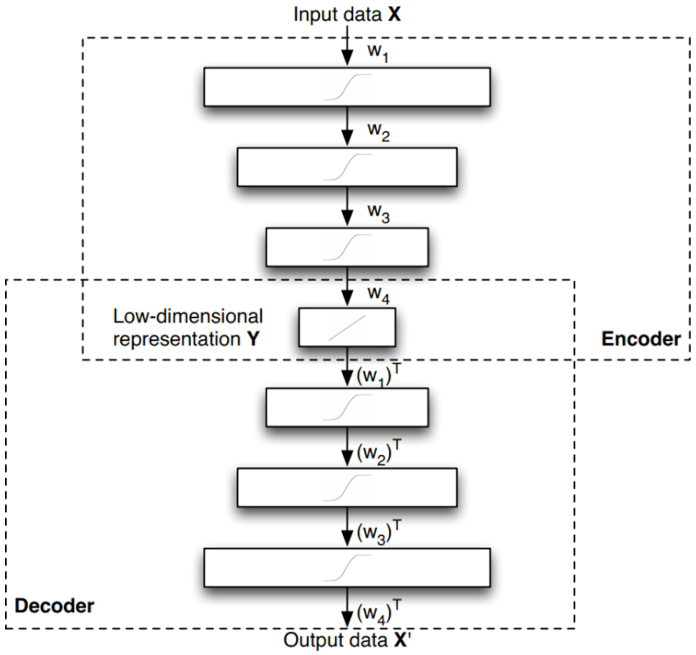


Figure 4: Restricted Boltzmann Machine auto-encoder

3 Evaluation

3.1 Evaluation

The evaluation setup is as explained in Fig 5. The initial training set contains data tokens of 1136 dimensions. Dimensionality is then reduced using the three techniques explained in previous sections to 50 dimensions. Obtaining the 50-dimensional data from PCA is ridiculously obvious. We take the best 50 Eigenvectors and project the high-dimensional data onto the space spanned by those 50 vectors. Since ICA alone itself can't reduce dimensionality, PCA was used to first reduce the dimensionality and to whiten the data and then Maximum Likelihood ICA was performed to make the components independent. For RBM auto-encoder, a linear visible layer and two binary hidden layers were used. The hidden layers were configured with 1000 and 50 nodes, to get a reduced representation of 50-dimensions. Contrastive-divergence algorithm was used to pre-train the Boltzmann weights and then conjugate gradient descent was used to fine-tune the weights. The classifier, which is a neural network, was configured to have two hidden layers with 32, 16 nodes respectively. The optimal weights for the neural network were obtained by performing simulated annealing.

The evaluation was made based on the running times of the algorithms and the true positive-true negative characteristics of the classifier.

3.2 Results

Table 1 displays the running times of the three methods evaluated on the dataset explained in Section 1. Even though the iterative versions of both PCA and RBM (contrastive divergence only) should take $\mathcal{O}(mn^2)$ running time, where m is the number of training tokens and n is the feature dimensionality, PCA takes lesser time since the I have used MATLAB's implementation of the

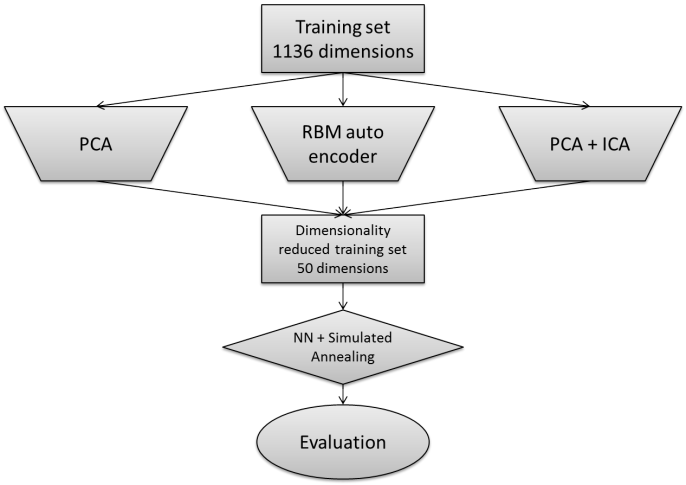


Figure 5: Evaluation setup

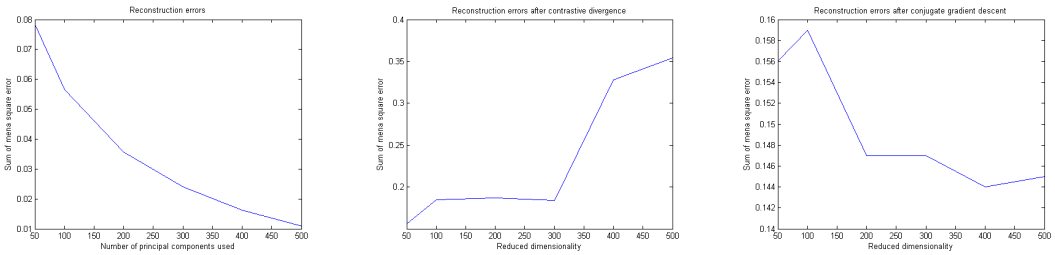
Eigenvalue decomposition. On the other hand, the maximum likelihood ICA also takes roughly around $\mathcal{O}(mn^2)$.

Table 1: Running times of the techniques

PCA	0.4203 ± 0.0219 s
ICA	4.4744 ± 0.3877 s
RBM AE	Pre-train - $33.0280s \pm 0.8947$ s
	Fine-tune - 87.8901 ± 3.1925 s

The next measure I used for evaluation is reconstruction errors. Since, ICA alone cannot perform dimensionality reduction and since it is impossible to recover the original features after performing ICA, I am comparing only PCA and RBM auto-encoder for this measure. Figure 6 shows the reconstruction errors generated by these techniques. Figure 6a shows the reconstruction errors generated by PCA for different number of reduced dimensions. Since PCA is a linear dimensionality reduction method and since the dimensionality is reduced based on the importance of Eigenvectors, the curve is easily explainable, a gradual reduction of errors. Figure 6b and Figure 6c show the reconstruction error curves for RBM auto-encoder after performing contrastive divergence and conjugate-gradient descent respectively. However, the reconstruction error for RBM auto-encoder is less intuitive. Contrastive divergence gives a reconstruction error that is increasing with the number of dimensions in the output. Meanwhile, when we fine-tune the weights using conjugate-gradient method, the reconstruction error gradually decreases. It will be interesting to study the logic behind the contrastive divergence algorithm which causes this phenomenon.

Table 2 displays the True positive-true negative characteristics of the classifier which was trained on the features generated by the dimensionality reduction techniques. These values were obtained by training and performing a 6-fold cross validation on the dataset. All three techniques perform reasonably well, while PCA + ICA combination outperforms the other two techniques.



(a) PCA (b) Contrastive divergence (c) Conjugate gradient descent

Figure 6: Reconstruction errors for PCA and RBM auto-encoder

Table 2: True positives-true negative characteristics

Technique	True positives		True negatives	
	Training	Test	Training	Test
PCA	0.79	0.81	0.86	0.88
PCA + ICA	0.84	0.85	0.87	0.89
RBM	0.77	0.80	0.93	0.78

4 Conclusion

I was able to do a fair assessment of the dimensionality reduction techniques and their applicability in seizure prediction. PCA, ICA and RBM auto-encoders were assessed as dimensionality reduction techniques. A neural network classifier was trained with simulated annealing to do the prediction which was then used to do the comparison of the performances of the dimensionality reduction techniques. A comparison of three dimensionality reduction techniques was made based on their ability to preserve the necessary information. This claim is validated by the performance of the classifier being significantly good for all three techniques. PCA + ICA combination outperforms the other two techniques. Nevertheless, all three techniques perform reasonably well.

References

[1] Kaggle machine learning competitions, 2014.

[2] Y. Freund and D. Haussler. *Unsupervised learning of distributions of binary vectors using two layer networks*. Computer Research Laboratory [University of California, Santa Cruz], 1994.

[3] G. Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

[4] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[5] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[6] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. 1986.