Auto-associative Memory: The First Step in Solving Cocktail Party Problem

Introduction:

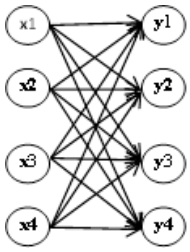
One of the most interesting and challenging problems in the area of Artificial Intelligence is solving the Cocktail Party problem. This is the task of attending to one speaker among several competing speakers and being able to switch the attention from one speaker to another at any given time. Human brain is remarkably efficient in solving this problem. There have been numerous attempts to emulating this ability in machines. Independent Component Analysis (ICA) and Blind Source Separation (BSS) are two of the most popular solutions to this problem but they both fail to generate similar results as human brain does. They are also very computationally expensive which makes them incapable of producing results in real-time. Moreover, they generally require at least two microphones to converge but as we know human brain can also work with one ear covered. This is evident from the fact that covering one ear does not make attending to one speaker in a cocktail party any harder.

I believe the reason that these methods fail to generate similar results is that they fail to capture the fundamental functionality of separation of sound sources in the brain. ICA and BBS both assume no prior knowledge about the sounds that they are receiving. This is not generally true about human brain. In the course of maturation of auditory system, brain is exposed and attuned to many sounds and learns when two or more tones happen together and when they are not. So when a new sound signal is presented to the brain it automatically groups the tones in the sound based on the likelihood that they are happening together.

I propose a new solution, one based on separation of sound sources through learning from past experience. In order to accomplish this, one needs to design a system that can regenerate sounds that it has become accustomed to when it is exposed to a noisy or distorted version of them. This task is best accomplished by an auto-associative memory.

Auto-associative Memory:

Auto-associative memories are content based memories which can recall a stored sequence when they are presented with a fragment or a noisy version of it. They are very effective in de-noising the input or removing interference from the input which makes them a promising first step in solving the cocktail party problem. The simplest version of auto-associative memory is linear associator which is a 2-layer feed-forward fully connected neural network where the output is constructed in a single feed-forward computation. The figure below illustrates its basic connectivity:



All inputs are connected to all outputs via the connection weight matrix W where W_{ij} denotes the strength of unidirectional connection from the i^{th} input to the j^{th} output. Since $x = y$ in auto-associative memories, we have $x = W^T x$. Therefore, all stored sequences must be eigenvectors of matrix W. Assuming all stored sequences are orthogonal to each other, we can represent the weight matrix as $W = XX^{-1} = XX^T$ where X is orthonormal

matrix of all stored sequences. Also, we are enforcing the weight matrix W to be symmetric so $W_{ij} = W_{ji}$. So, assuming we have p sequences to store in our memory we can rewrite weight matrix as:

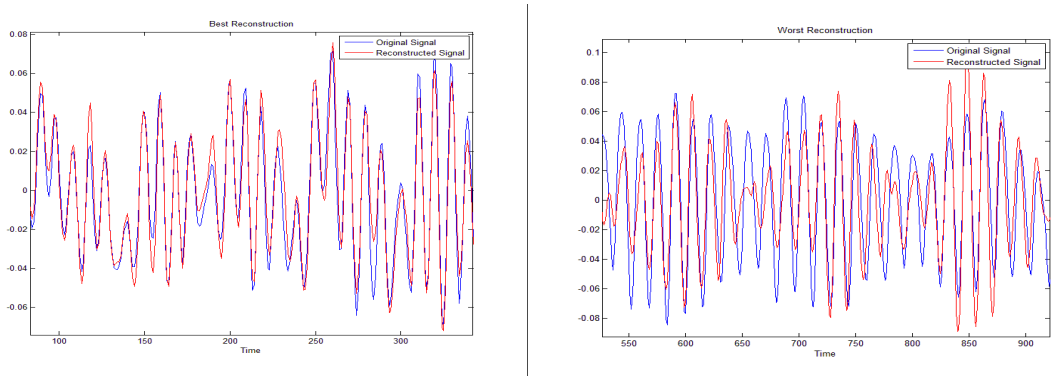
$$W = \sum_{i=1}^{p} x^{(i)} \times x^{(i)}$$

Initial Testing:

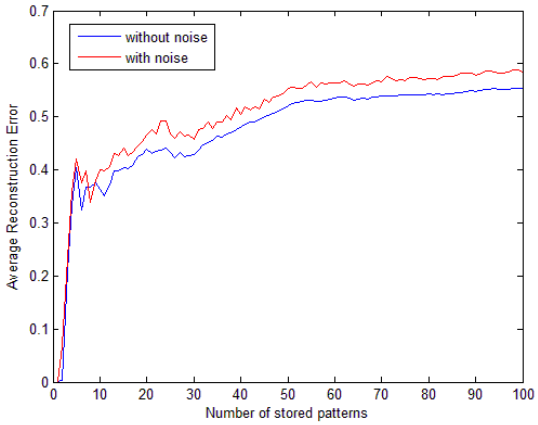
To verify functionality of the linear associator, I started by training the auto-associative network using segments of "Sleep Away" song which is in "Sample Music" folder of all PC's. Since the default sampling rate of 44100 samples/second is too much to process the data efficiently, I compressed the data to 10000 samples/second and partitioned it to segments of 1000 samples per segment, which corresponds to one tenth of a second of the original song. Noting that we can have perfect reconstruction if we store sequences that are orthogonal to each other, I used inner product as a measure of how orthogonal segments are with respect to each other. I calculated a score of orthogonality for each segment according to the sum of their inner products with all other segments:

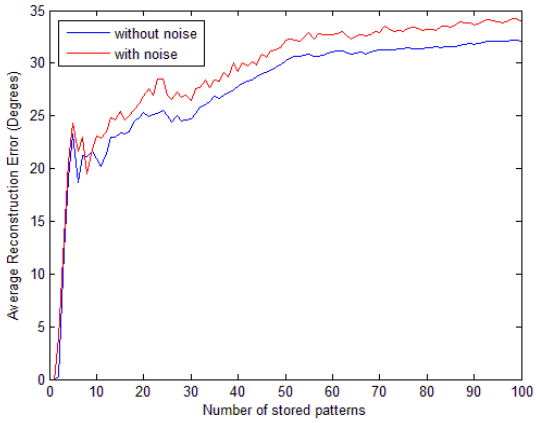
$$O\_Score_i = \sum_{j=1}^{1000, j \neq i} \frac{Segment_i}{||Segment_i||} \cdot \frac{Segment_j}{||Segment_j||}$$

The weight matrix is then calculated using 100 segments with the lowest score, according to the formula above. Then, I ran several tests in order to assess reconstruction of the stored patterns. In the first test, I tried to find out how the original signals compare with the reconstructed signals. The plots below show the reconstructed signal versus the original signal for two stored segments with the best and worst O_Score's.



In order to get a better sense of the performance of the linear associator, I ran another test where I consecutively increased the number of stored segments from 1 to 100 and measured the reconstruction error. In one setting, I applied the original signals without any noise and measured the average error made per reconstruction and in another setting, I applied noise with the same power as the original signals (i.e. 0dB SNR), to the original signals and again measured the average error made per reconstruction. Since the original signals and the reconstructed signals are forced to have the same length, the reconstruction error can also be thought of as the average angle by which the reconstructed signals deviate from the original signals. The plots below show the results for the average reconstruction error and average angle of deviation from the original signal as a function of number of stored patterns:





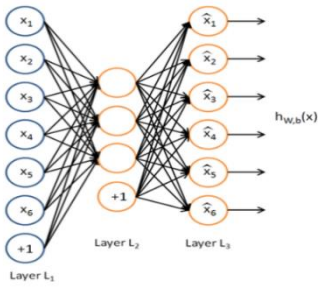
As it can be seen from the plots, the reconstruction error rises very rapidly as the number of stored patterns reaches 10 but pretty much flattens out afterwards. This is expected because there is a high correlation between segments of a song as there are same instruments being played during all segments. Therefore, there are very few segments that are orthogonal to others which makes the reconstruction imperfect. The flattening out of reconstruction error for high number of stored patterns can also be explained by high correlation between the segments because if a pattern to be stored can be expressed by previously stored patterns the weight matrix won't change after learning the new pattern thus average reconstruction error remains the same.

Autoencoder as Autoassociative Memory:

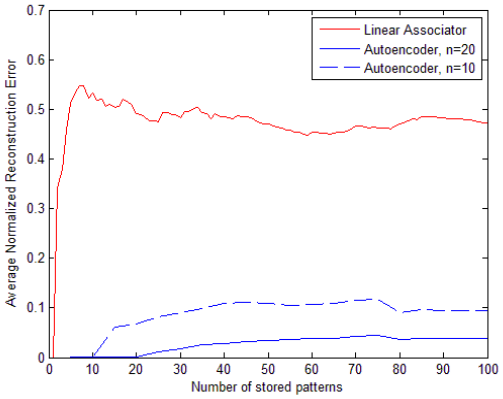
Initial testing of linear associator indicates that it does quite a poor job storing patterns that are linearly dependent. In order to better distinguish between linearly dependent patterns, nonlinearity of some sort has to be included in the system. Since the input domain is [-1 1], hyperbolic tangent was chosen as the activation function for each neuron to incorporate nonlinearity into the system.

Next step was to choose the topology of neural network. Feed-forward networks and networks with feedback like Hopfield networks were considered for implementation of autoassociative memory but feed-forward networks were chosen because of their relative simplicity and feasibility to train. After a successful implementation of a feed-forward autoassociative memory, impact of feedback on noise reduction was also studied.

Among feed-forward networks, autoencoder was finally chosen for several reasons. First, the simplicity of these networks makes them easy to work with. Also, the smaller number of hidden neurons makes training these networks much faster and at the same time enables them to store larger size patterns. Perhaps the most important reason for choosing autoencoder is based on the observation that by limiting the number of neurons in the hidden layer, each pattern will get compressed in the hidden layer to be reconstructed again at the output so any noise in the input should get averaged out and rejected at the hidden layer. To assess this theory, noise performances of two autoencoders with different number of hidden neurons were tested. The results are discussed in the following section. The downside of choosing autoencoder, however, is the lower number of patterns that can be stored because of lower number of neurons in the hidden layer.



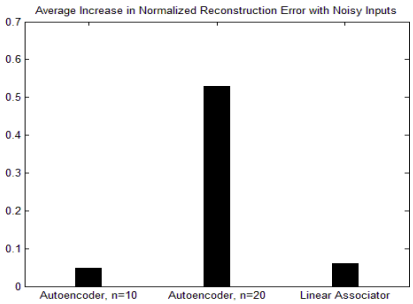
The procedure for training and testing the network was similar to the one for linear associator with an exception on the number of samples per segment. To make training time reasonable, number of samples per segment was reduced to 100. Then, the reconstruction error was measured as a function of number of stored patterns for two different autoencoders; one with 10 hidden neurons and another with 20. Results were compared with reconstruction error of a linear associator also trained with 100 samples per segment to make the comparison fair. The plot below shows the results:



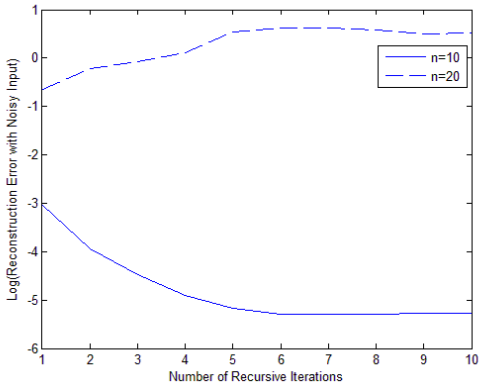
As it can be seen from the plot, incorporating nonlinearity into memory can significantly reduce reconstruction error. Moreover, the higher the number of neurons in the hidden layer the better the network can fit more patterns and therefore the lower the reconstruction error will be. As it is evident from the plots, the maximum number of stored patterns in an autoencoder that can be perfectly reconstructed is as high as the number of hidden neurons. As the number of stored patterns increases beyond that, the average reconstruction error rises gradually and then flattens out. This makes autoencoders also a pretty decent candidate for data compression for applications that can tolerate a small loss of information!

Noise Performance:

In order for an autoassociative memory to work well in a cocktail party problem application, it has to be able to reject most of the noise and interference. So, in final evaluation of autoencoder autoassociative memory, I tested noise performance of autoencoders with different number of hidden neurons. To get the best noise performance, I trained each network with four noisy versions of each patterns while forcing the output to be the ideal noiseless pattern. To reduce over fitting while maintaining a low reconstruction error, I enabled regularization with a small regularization coefficient (β=0.0005). Then, I applied noisy inputs with SNR=10dB to each network and measured average increase in reconstruction error. The plot below shows the results for a linear associator, an autoencoder with 10 hidden neurons and an autoencoder with 20 hidden neurons.



A surprising result from this experiment is how much better the noise performance of a linear associator gets when the size of each segment is reduced from 1000 to 100. However, since the reconstruction error remains high it is still not a suitable topology for implementing an autoassociative memory. Another interesting result is the superiority of autoencoder with 10 hidden neurons in noise performance compared with the one with 20 hidden neurons. This is in line with the hypothesis that noise gets averaged out at the hidden layer and thus, the lower the number of neurons in the hidden layer the higher the averaging and the higher the rejection of noise. Since noise power at the output of this network is less than the noise power at the input, we can get an even better noise performance if we place a feedback loop around the system and recursively apply the output to the input. The plot below shows log of reconstruction error of the two autoencoders with noisy inputs as a function of number of recursive iterations:



Conclusion:

This project illustrates that autoencoders can be used to store linearly dependent patterns with negligible reconstruction error. Results, however, suggest that there is a trade-off between minimizing the reconstruction error and maximizing noise rejection. While networks with higher number of hidden neurons can store more patterns and with less reconstruction error, they will have lower noise rejection and vice versa. Moreover, if an autoencoder with an optimum number of hidden neurons, is placed in a feedback loop, saved patterns will act as attractors. So, with each recursive iteration, more noise is rejected until a noiseless pattern is reconstructed at the output. This makes autoencoders very attractive candidates for implementation of an autoassociative memory in solving the cocktail party problem. Further research needs to be completed to find the optimum number of hidden neurons for a given input size, minimum noise rejection or maximum reconstruction error.

References:

http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial