Machine Learning in the Web

Evaluating ICA – by Timo Mechsner and David Leins

# The Task

The Task was to play around with Independent Component Analysis: Compare results of ICA and PCA, evaluate noise sensitivity, try ICA on self-recorded sources and so on. We divided our project into nine parts of independent but conceptually consecutive experiments implemented in Matlab, which will be described in the following paragraphs.

# Experiment 1 – Separation of sound sources from the internet

We took two sounds from <http://research.ics.aalto.fi/ica/cocktail/cocktail_en.cgi> like presented in the lecture and applied ICA to two random mixes of these sources. The result is a good separation of sources, ICA is capable of restoring the original sources with very little error.

# Experiment 2 - Live recorded sounds

We recorded two people talking with two microphones to create two different mixes of the same audio source. ICA was not capable of restoring the original sources. One reason could be that the voices of the two people are very similar. Another reason could be noise in the recordings. So we decided to examine the impact of noise to the signals and to the mix.

# Experiment 3 - Generated Signals

To experiment with noise we decided to examine generated signals. The FastICA Matlab package offers a method to generate and mix four simple signals (sawtooth, sine, white noise and another funny periodic function). Applying ICA to four random mixes of these four signals results in a very good reconstruction of the original sources.

# Experiment 4 - Noisy Sources

Now we add some noise to the signals before mixing them. We therefore implemented a function that adds noise limited to the mean of the absolute value of the signal. The quality of the result decreased significantly but still the shapes of the original sources are recognizable.

# Experiment 5 - Noisy Mix

Adding the same amount of noise to the mix resulted in an even worse result. This might be the reason why ICA did not work on our live recordings, as both microphone recordings (=mixes) are influenced by hall, noise of the microphones, etc.

We read about the possibility to reduce noise by feeding ICA more mixes and use information of the additional output signals, but we did not have enough time to test this.

# Experiment 6 & 7 - Evaluation of Noise-Error Relation

We were interested in the relation of noise and error (=difference between original signal and ICA output). Here we faced two problems: First, the order of the ICA outputs does not necessarily match the order of the input signals. Second, the signals might be inversed. We solved this by brute force: Our function matchAndEval compares each source to each output and each inverted output. We assume that the mean error, that is the mean of the absolute difference values, is minimal when the compared signals are corresponding input and output signals. The function then returns a list of all mean errors of matching input and output signals as well as the standard deviation of each error.

Now we could examine the relation of noise to error by applying ICA to ten signals with noise / mixes with noise limited to p times the absolute mean of the signal, with  
p in {0.0, 0.5, 1.0, 1.5, ... , 5.0}.

In the plot we could see that the ICA is much more sensitive to a noisy mix than to noisy source signals.

# Experiment 8 & 9 - Comparison with PCA

We ran a pca on our mixed generated sources (same scenario as experiment 3) and on the mixed Aalto webside signals (same scenario as experiment 1) to compare the output to the result of (fast)ica.

By plotting the data one can clearly see that the ica results were much better. This is due to the fact that pca decorrelates data by finding new basis vectors which have to be orthogonal. In our case where input and output dimensions are equal, PCA just rotates the data so that the resulting axes are not correlated. This seems not to be working for blind source separation.  
ICA on the other hand tries to find statistical dependencies in the data and is more flexible in placing new axes as they do not need to be orthogonal.

Fun fact: ICA presumes the signals to be independent, thus whitening a dataset guarantees a better result. Fastica uses a fixpoint-iteration scheme to accomplish that as it happens to be 10 to 100 times faster than conventional methods, but PCA can also be used to whiten datasets. In fact some ICA implementations make use of it’s predecessor PCA.

Code available on: https://github.com/tmechsner/mlw-ss2016.git