Blind Separation of Two Speeches Using ICA

Yigit Bilgen

Elliot Rosen

Independent Component Analysis

- "Cocktail Party Problem"
- Separate up to *n* sources from *n* signals
- Assume signals are independent
- Model signals as a linear combination of source signals
- (Blindly) Guess the weighting of the signals to recover the original source signals

FastICA

- Preprocess
  - Center the mixture
  - Whiten the mixture
- Iterate until weight vectors stabilize:
  - Increase (approximated) negentropy
  - Normalize weight vectors
  - Decorrelate weight vectors

Preprocessing

- Center the data
- Whitening: decorrelate the signals to make computation easier
  - Covariance matrix = I
- These are necessary for the negentropy approximation used later (zero mean, unit variance)

Preprocess.m

```
function [xwhitened, mu, whitener] = Preprocess(x)
% x: nxk vector, n is the number of mixtures, k is the number of samples
% Returns: xwhitened = preprocessed data;
% mu = data means
% whitener = matrix used for whitening. Used later for comparison.


% Center
mu = mean(x, 2);
xcenter = x - repmat(mu, 1, length(x));

% Covariance
sigma = cov(xcenter');
[V, D] = eig(sigma);

% Whiten
whitener = V * diag(diag(D).^(-0.5)) * V';
xwhitened =  whitener * xcenter;

end
```

Negentropy

- Definition $J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y})$
- Maximize this to maximize independence
- Minimize mutual information
- Approximation of J for faster calculation

$$J(y) \propto \left(E\{G(y)\} - E\{G(v)\}\right)^2$$

$$G(u) = \log(\cosh(u))$$
$$g(u) = G'(u) = \tanh(u)$$
$$g'(u) = 1 - \tanh^2(u)$$

Iteration

- Increase negentropy: $\mathbf{w}^{+} = E\left\{\mathbf{x}g\left(\mathbf{w}^{T}\mathbf{x}\right)\right\} - E\left\{g'\left(\mathbf{w}^{T}\mathbf{x}\right)\right\}\mathbf{w}$
- Normalize (because we still want variance of 1, keeps things scaled)
- Decorrelate W, to prevent weights from biasing to the same source.

$$\mathbf{W}_{next} = \left(\mathbf{W}\mathbf{W}^{T}\right)^{-1/2}\mathbf{W} = \mathbf{F}\mathbf{D}^{-1/2}\mathbf{F}^{T}\mathbf{W}$$

IterateICASingle.m

```
function [out] = IterateICASingle(w, x)
% Performs one iteration of ICA
% W: Weighting vector (nx1); x: data (nxk)

[n, k] = size(x);

p = w'*x;
gp = g(p);

% E{x * g(p)}
out = x * gp' / k;

% E{g'(p)} * w
g2 = mean(gdot(p), 2) * w;
out = out - g2;

% Normalize
out = out / norm(out);
end
```

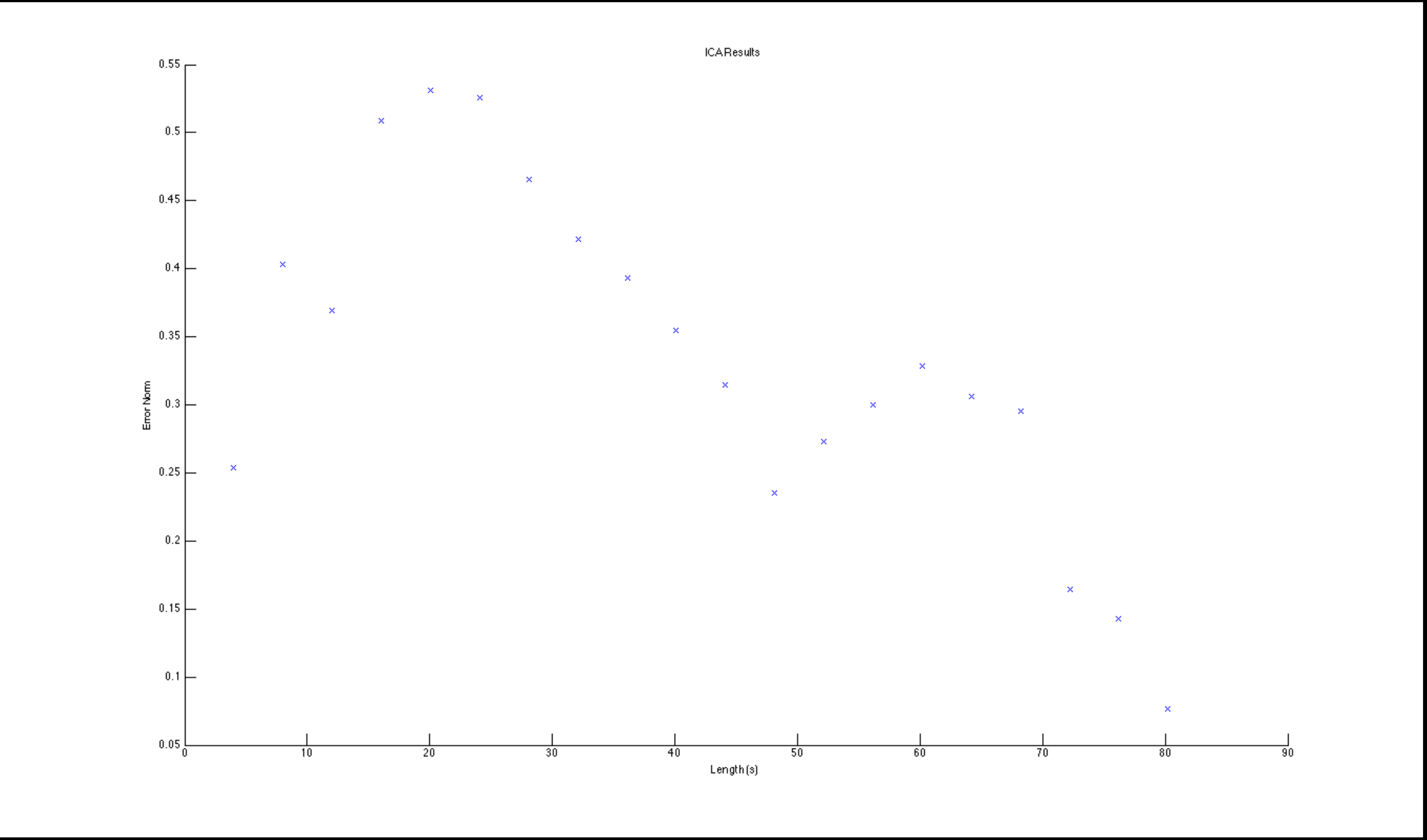
ICA.m

```
[x, mu, whitener] = Preprocess(x);

w1 = [2; -1];
w2 = [-1; 2];
w1 = w1 / norm(w1);
w2 = w2 / norm(w2);

W0 = zeros(2, 2); % old value of W
W = [w1, w2]';
while norm(eye(2, 2) - abs(W' * W0)) > 0.001
    w1 = IterateICASingle(w1, x);
    w2 = IterateICASingle(w2, x);
    W0 = W;
    W = [w1, w2]';
    [V, D] = eig(W*W');
    W = (V / sqrt(D)) * V' * W;
    Wt = W';
    w1 = Wt(:,1);
    w2 = Wt(:,2);
end

near_i = abs(W * whitener * A);
near_i = near_i / norm(near_i);

% distance from identity matrix. Flip rows if necessary.
score = min(norm(eye(2, 2) - near_i), norm(eye(2, 2) - flipud(near_i)));
```

Testing

- Confirm accuracy for different mixing matrices.
- Do a Monte Carlo simulation for a mixing matrix generated by a random parameter
- Normalize $\left|\mathbf{W} * \tilde{\mathbf{A}}\right|$ and compare to I
  - Should be close and consistent for a good result
- Use the same speech

Results (10 trials per clip length)



Conclusions

- FastICA converged *very* consistently, regardless of the mixing parameters
- Learning generally increased with time
- FastICA in this form is only applicable to artificially mixed sources

Other ideas

- Live mixtures
- More preprocessing specific to time-variant signals (like sound)
- Alternate approximation and decorrelation methods
- Establish when overlearning occurs

Sources

- Pulp Fiction (1994)
- Doctor Who: “Blink” (2007)
- Hyvärinen, Aapo, and Erkki Oja. "Independent component analysis: algorithms and applications." *Neural networks* 13.4 (2000): 411-430.