Derivation of FastICA algorithm

Zein
GitHub@zein0115

July 29, 2020

1 Requirements

Basic knowledge of matrix calculus and multivariable calculus.
Newton's method.

2 Notations

The notations are almost same as article by Aapo Hyvärinen.

2.1 Datasets

The original dataset $\mathbf{s}$ with dimension d_s

$$\mathbf{s} = \begin{bmatrix} | & | & \cdots & | \\ s^{(1)} & s^{(2)} & \cdots & s^{(n)} \\ | & | & \cdots & | \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{d_s} \end{bmatrix} \in \mathbb{R}^{d_s \times n}$$

The mixed dataset $\mathbf{x}$ with dimension d_x, which is a linear combination of $\mathbf{s}$

$$\mathbf{x} = \begin{bmatrix} | & | & \cdots & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(n)} \\ | & | & \cdots & | \end{bmatrix} \in \mathbb{R}^{d_x \times n}$$

2.2 Transforming matrices

Since $\mathbf{x}$ is a linear combination of $\mathbf{s}$, assume that

$$A\mathbf{s} = \mathbf{x}$$

Where

$$A \in \mathbb{R}^{d_x \times d_s}$$

Assume $W = A^{-1}$ exists, then

$$\mathbf{s} = W\mathbf{x}$$

If we let $w_i \in \mathbb{R}^{d_x}$ be a vector which satisfy

$$W = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_{d_s}^T \end{bmatrix}$$

Then

$$s_i = w_i^T \mathbf{x}$$

Alternatively, if we just use w^T to express one row of W, then

$$y = w^T \mathbf{x}$$

Implies $y = s_i$ for some i.

2.3 Preprocessing

Before doing FastICA, dataset $\mathbf{x}$ need to be centerning and whitening to simplify the calculation

$$\mathbf{x} = \mathbf{x} - E[\mathbf{x}]$$
$$\tilde{\mathbf{x}} = ED^{-1/2}E^T\mathbf{x}$$

Where E and D are generate by eigonvalue decomposition of the covariance matrix, $D^{-1/2} = \sqrt{D^{-1}}$

$$E[\mathbf{x}\mathbf{x}^T] = EDE^T$$

After centerning and whitening ($\mathbf{x}$ is used to denote $\tilde{\mathbf{x}}z$)

$$E[\mathbf{x}] = 0$$
$$E[\mathbf{x}\mathbf{x}^T] = I$$

2.4 Optimization

The goal of FastICA for one unit is to find a vector w that maximize the nongaussianity of $w^T x$, where Nongaussianity is a measurement of independence.
Nongaussianity is measured by the approximation of negentropy $J(w^T\mathbf{x})$. More negentropy, more independence.

$$J(w^T\mathbf{x}) \propto \{E[G(w^T\mathbf{x})] - E[G(\nu)]\}^2$$

Where ν is a Gaussian variable of zero mean and unit variance, while G is a nonquadratic function. For example

$$G_1(u) = \frac{1}{a_1}\log\cosh a_1 u, G_2(u) = -\exp(-u^2/2)$$

2.5 Special note

I am going to introduce a matrix product rule called "$*$". Which is same as ".*" in MATLAB and "*" in Numpy for matrix calculation, namely, element product. Following are some examples

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} * \begin{bmatrix} d \\ e \\ f \end{bmatrix} = \begin{bmatrix} ad \\ be \\ cf \end{bmatrix}, \begin{bmatrix} a \\ b \\ c \end{bmatrix} * \begin{bmatrix} d & g \\ e & h \\ f & i \end{bmatrix} = \begin{bmatrix} ad & ag \\ be & bh \\ cf & ci \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \end{bmatrix} * \begin{bmatrix} d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} ad & be & cf \\ ag & bh & ci \end{bmatrix}$$

Intuitively, $(A * B)^T = A^T * B^T$.

3 Derivation of FastICA for one unit

To maximize $J(w^T \mathbf{x})$, we need to find the optima of $E[G(w^T \mathbf{x})]$.
To simplify the calculation, let $\|w\| = 1$, i.e., $w^T w = 1$.

Then our goal is to find the w that maximize $E[G(w^T \mathbf{x})]$ constrained by $\|w\| = 1$, according to Kuhn-Tucker conditions, we need to find w satisfy

$$\frac{\partial E[G(w^T \mathbf{x})]}{\partial w} = \lambda \frac{\partial w^T w}{\partial w} \tag{1}$$

Let $g = G'$ and $\beta = 2\lambda$, where λ is Lagrange multiplier. The left hand side equals to

$$\begin{aligned}
\frac{\partial E[G(w^T \mathbf{x})]}{\partial w} &= \frac{\partial}{\partial w} \frac{1}{n} \sum_{i=1}^{n} G(w^T x^{(i)}) \\
&= \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial w} G(w^T x^{(i)}) \\
&= \frac{1}{n} \sum_{i=1}^{n} \frac{\partial w^T x^{(i)}}{\partial w} \frac{\partial}{\partial w^T x^{(i)}} G(w^T x^{(i)}) \\
&= \frac{1}{n} \sum_{i=1}^{n} x^{(i)T} g(w^T x^{(i)}) \\
&= \left[\frac{1}{n} \sum_{i=1}^{n} x^{(i)} g(w^T x^{(i)}) \right]^T \\
&= E\left[\mathbf{x} * g(w^T \mathbf{x})\right]^T
\end{aligned}$$

The right hand side is

$$\lambda \frac{\partial w^T w}{\partial w} = 2\lambda w^T = \beta w^T$$

Thus the euqation (1) could be rewrite to

$$E\left[\mathbf{x} * g(w^T \mathbf{x})\right]^T - \beta w^T = 0 \tag{2}$$

Equivalent to

$$E\left[\mathbf{x} * g(w^T \mathbf{x})\right] - \beta w = 0 \tag{3}$$

FastICA uses Newton's method to solve equation (3). Newton's method is a numerical method to find root of a nonlinear equation. For a nonlinear equation $F(x) = 0$, do iteration

$$x = x - \frac{F(x)}{F'(x)}$$

If F is a vector-valued multivariable function and x is a vector, do

$$x = x - J_F^{-1}(x) F(x)$$

Where $J_F(x) = F'(x)$ is Jacobian matrix while $J_F^{-1}(x)$ is the inverse.

Consider $F(w) = E\left[\mathbf{x} * g(w^T\mathbf{x})\right] - \beta w$, then we are going to solve

$$F(w) = 0 \tag{4}$$

First find the Jacobian matrix

$$\begin{aligned}
J_F(w) &= \frac{\partial F(w)}{\partial w} \\
&= \frac{\partial}{\partial w} E\left[\mathbf{x} * g(w^T\mathbf{x})\right] - \frac{\partial}{\partial w}\beta w \\
&= \frac{\partial}{\partial w}\frac{1}{n}\sum_{i=1}^{n} x^{(i)} g(w^T x^{(i)}) - \beta I \\
&= \frac{1}{n}\sum_{i=1}^{n} x^{(i)} x^{(i)T} g'(w^T x^{(i)}) - \beta I
\end{aligned}$$

To simplify the calculation, we assume that $x^{(i)}x^{(i)T}$ and $g'(w^T x^{(i)})$ are independent variables, then the approximation is given as

$$\begin{aligned}
J_F(w) &= \frac{1}{n}\sum_{i=1}^{n} x^{(i)} x^{(i)T} g'(w^T x^{(i)}) - \beta I \\
&\approx \frac{1}{n}\sum_{i=1}^{n} x^{(i)} x^{(i)T} \frac{1}{n}\sum_{i=1}^{n} g'(w^T x^{(i)}) - \beta I \\
&= \frac{1}{n}\mathbf{x}\mathbf{x}^T \frac{1}{n}\sum_{i=1}^{n} g'(w^T x^{(i)}) - \beta I \\
&= E[\mathbf{x}\mathbf{x}^T]E[g'(w^T x^{(i)})] - \beta I \\
&= I E[g'(w^T x^{(i)})] - \beta I \\
&= I\left\{E[g'(w^T x^{(i)})] - \beta\right\}
\end{aligned}$$

Then $J_F(w)$ is a diagonal matrix, the inverse could be simply find as $J_F^{-1} = I\left\{E[g'(w^T x^{(i)})] - \beta\right\}^{-1}$. Apply $J_F^{-1}(w)$ to the iteration euqation $w = w - J_F^{-1}(w)F(w)$

$$\begin{aligned}
w &= w - J_F^{-1}F(w) \\
&= w - I\frac{E\left[\mathbf{x} * g(w^T\mathbf{x})\right] - \beta w}{E[g'(w^T x^{(i)})] - \beta} \\
&= \frac{1}{E[g'(w^T x^{(i)})] - \beta}\left\{E[g'(w^T x^{(i)})]w - \beta w - E[x * g(w^T x^{(i)})] + \beta w\right\} \\
&= \frac{1}{E[g'(w^T x^{(i)})] - \beta}\left\{E[g'(w^T x^{(i)})]w - E[x * g(w^T x^{(i)})]\right\}
\end{aligned}$$

Since $E[g'(w^T x^{(i)})] - \beta$ is a scaler and w is constrained as $\|w\| = 1$, $E[g'(w^T x^{(i)})] - \beta$ could be ignored by applying

$$w = E[g'(w^T x^{(i)})]w - E[x * g(w^T x^{(i)})] \tag{5}$$

$$w = \frac{w}{\|w\|} \tag{6}$$

in each iteration, which is a basic form of FastICA.

I would not post the derivation of FastICA for several unit here, since the derivation in the original article is comprehensive enough.

Reference

[1] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.