INFERENCE AND REPRESENTATION: HOMEWORK 4

STUDENT: MARGARITA BOYARSKAYA

Problem 1. *Log-likelihood*

Let's examine the Kullbach-Leibler divergence

$$D(X, WH) = \sum_{ij}(x_{ij} log \frac{x_i j}{(WH)_{ij}} - (x_{ij} - (WH)_{ij})). \tag{1}$$

We are interested in finding $arg\min_{W,H} D(X, WH)$, where for $X_{ij} > 0$ the function attains it's minimum. Dropping the constants, this is equivalent to finding $\max L(W, H)$, where

$$L(W, H) = \sum(x_{ij} \log(WH_{ij} - WH_{ij}). \tag{2}$$

Indeed, this follows from the fact that, given i.i.d. variables X_i, by strong law of large numbers $\frac{1}{n}\sum_i - \log p(X - i) \to E(-\log p(X_i)$. Note that $E[X - WH] = E[\log \frac{X}{WH}] = D(X, WH)$. Divergence D is strictly positive and attains zero value iff X=WH (this follows from (1) and from inequality $x \log x \leq x - 1$).

By showing that $E[\log x_i j - \log WH_{ij}] \geq 0$ we demonstrate that minimizing $E[-\log X_{ij}]$ gets us closer to the true value, thus finding $arg \min D(W, H)$ is equivalent to finding $arg \max E[\log X]$, which, disregarding constants, is precisely equivalent to maximizing expression (2). □

a) (non-decreasing update sequence) The definition can be interpreted as stating that the surface $x \to g(x, x^t)$ lies below the surface $f(x)$ and is tangent to it at point $x = x^t$. If x^{t+1} is the value of the argument x at which the maximum of of $g(x, x^t)$ occurs, then the follwoing holds: $g(x^{t+1}, x^t) \geq g(x^t, x^t)$. From this, together with the definition of majorant function, it follows that:

$$f(x^t) = g(x^{t+1}, x^t) + f(x^{t+1}) - g(x^{t+1}, x^t)$$

$$\geq g(x^t, x^t) + f(x^t) - g(x^t, x^t) = f(x^t). \ \square$$

b) Let us prove first a more general statement: for non-negative $a_1, ..., a_n$ and $b_1, ..., b_n$

$$\sum_{i=1}^{n} a_i \log \frac{a_i}{b_i} \geq \{\sum_{i=1}^{n} a_i\} \log \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}. \tag{3}$$

The proof follows from Jensen's inequality, drawing on the convexity of $x \log x$:

$$f(\sum_i \alpha_i x_i) \leq \sum_i \alpha_i f(x_i)$$

for $\alpha : \alpha \geq 0, \sum_i \alpha = 1$ Now let's take $\alpha_i = b_i / \sum_j b_j$ and $x_i = a_i / b_i$. We get:

$$\sum_i \frac{b_i}{\sum_j b_j} \frac{a_i}{b_i} \log \frac{a_i}{b_i} = \frac{a_i}{\sum_j b_j} \log \frac{a_i}{b_i} \geq \sum_i \frac{a_i}{\sum_j b_j} \log \sum_j \frac{a_i}{\sum_j b_j}.$$

This proves equation (1). In order to obtain the required inequality, one needs only to transition to the notation used in the problem, denoting y_i as b_k, a_i as c_k, and using the fact that $\sum_k c_k = 1$. $\square$

c) The statement of part (c) is proved the same way as above, using the fact that for $c_k = \frac{w_{ik}^t h_{kj}^t}{\sum_{k' \leq r} w_{ik'}^t h_{k'j}^t}$ conditions $0 \leq c_k \leq 1, \sum_k c_k = 1$ hold.

d) It is immediate to verify that $G(W, H, W^t, H^t) = L(W, H)$, ignoring the constant that do not constraint optimization. Let us prove that $G(H, W, H^t, W^t) \leq L(H, W)$:

$$G(H, W, H^t, W^t) = \sum_{ijk} x_{ij} c_{kij} (\log w_{ik} + \log h_{kj}) - w_{ik} h_{kj} \leq$$

$$\leq [\text{ since } c_i k \leq 1] \leq \sum_{ij} x_{ij} \sum_k c_{kij} \log \frac{w_{ik} h_{kj}}{c_i k} - w_{ik} h_{kj} \leq$$

$$\leq [\text{ by } (c)] \leq \sum_{ij} x_{ij} \log \sum_k w_{ik} h_{kj} - w_{ik} h_{kj},$$

where the last expression is precisely $L(H, W)$. $\square$

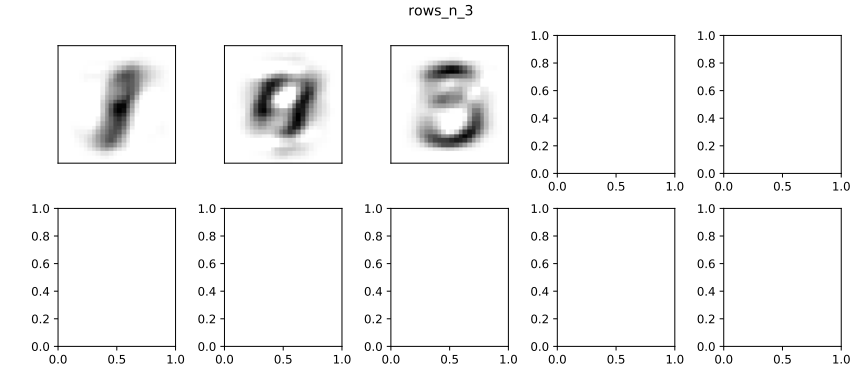
e) $\frac{\partial G}{\partial w} = \sum_{ijk} x_{ij} C_{ijk} \frac{1}{w_i k} - \sum_{ijk} h_{ik} = \sum_{ijk} x_{ij} \frac{w_{ik}^t h_{kj}^t}{\sum_{k'} w_{ik}^t h_{kj}^t} \frac{1}{w_{ik}} - \sum_{ijk} h_{ik} = 0 \leftrightarrow \leftrightarrow w_{ik} = w_{ik}^t \frac{\sum_j x_{ij} h_{kj}^t / W H_{ij}^t}{\sum_j h_{kj}}$. The update rule for h is derived is the same way. $\square$

Problem 2. *NMF vs. PCA on images.*

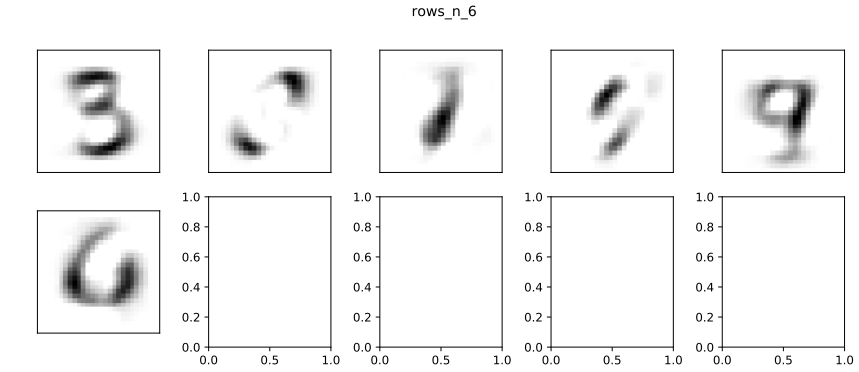
The code that implements the solutions for problems $2.a, 2.b$, and $2.c$ is included in the attached file *nmf.py*. Any new lines embedded in the original code provided by the instructors are marked with capitalized commentary lines preceded by *#!!!*. For ease of navigation, commentary lines mark and separate solutions to individual parts of the problem.

Below are the images resulting from the implementation of *nmf.py*:

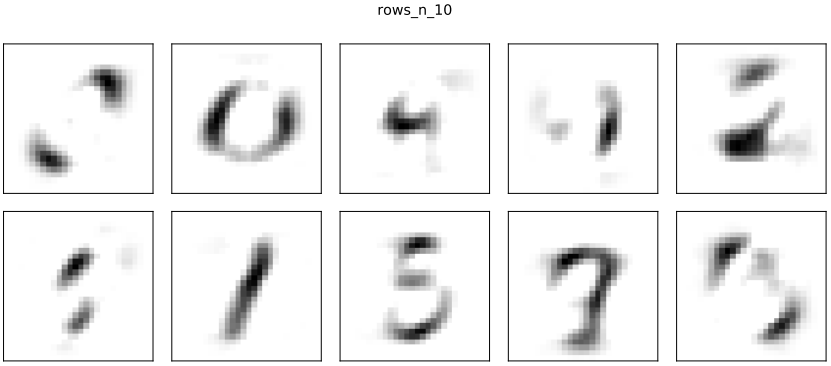
a) Rows of H, as obtained for $r \in \{3, 6, 10\}$:



$r = 3$



$r = 6$

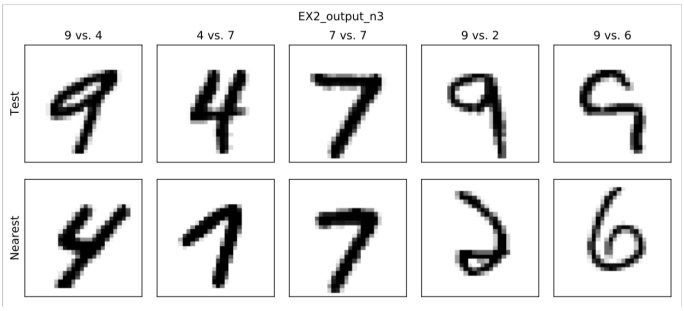


$r = 10$

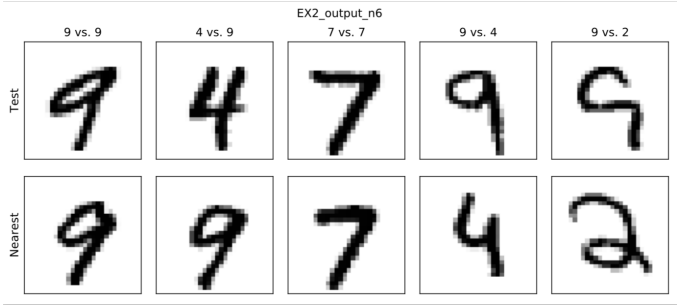
The original image files are included in the attachments and are titled *rows_n_X.pdf*.

As we see, for $r = 3$ the NMF method learns to distinguish digit $1, 9, 3$, and for $r = 6$ it recognizes key structural features of all 10 digits.

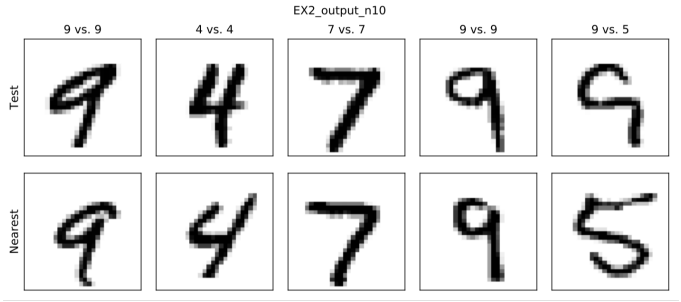
b) The results of running the nearest neighbours on the test images are below:



$r = 3$



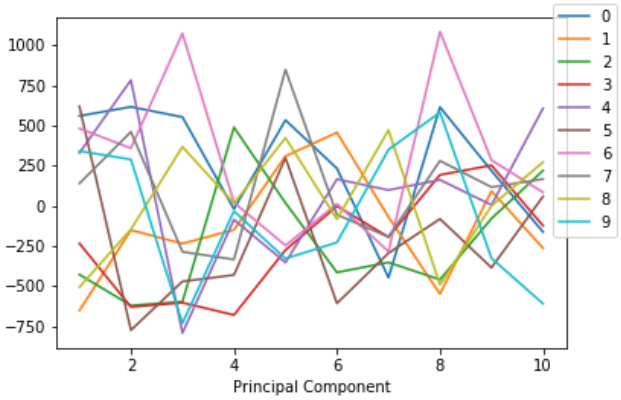
$r = 6$



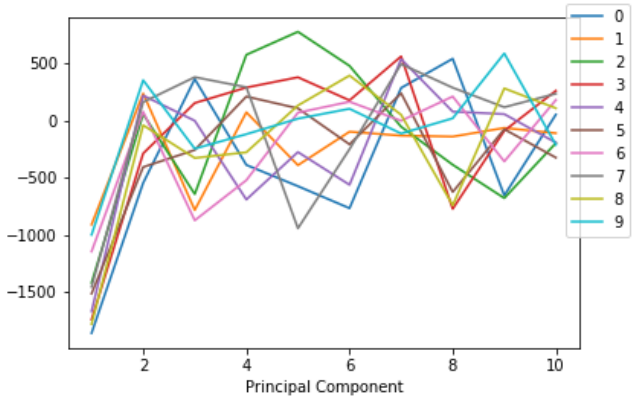
$r = 10$

The original image files are included in the attachments and are titled *EX2_output_nX.pdf*.

c) Let us examine two plots:



Random image coefficients (first 10 PCs).



Random image coefficients ($r = 10$ NMF).

It is evident that in contrast to the PCA, the NMF method shows coefficients exhibiting one or two 'peaks', suggesting that the method succeeds in establishing determinant 'patterns' for each digit. The figure above shows that the digits $3, 7$, and 4 share pattern 7. The NMF appears to induce better performance in the task of constructing low-dimensional representation of the digits, perhaps due to better 'base images'/patterns it creates.

Problem 3. *Factor Analysis on Covariance and Correlation.*

a) The given inequality for the Pearson coefficient is a formulation of the Cauchy-Bunyakovsky inequality with $\langle X, Y\rangle := \mathrm{E}(XY)$.

Let t be a real variable. Keeping in mind the bilinearity of covariance, it is easy to see that:

$$0 \leq Var[tX_i + X_j] = \sigma_{ii}t^2 + 2\sigma_{ij}t + \sigma_{jj}.$$

For quadratic polynomial function $Q(t) = At^2 + Bt + C$ the inequality above necessitates that the determinant is non-positive:

$$4\sigma_{ij}^2 - 4\sigma_{ii}\sigma_{jj} \leq 0,$$

which is precisely q.e.d. □

b) The factor analysis model is scale invariant: factor pattern resulting from performing FA on a correlation matrix is simply a rescaling of the factor pattern obtained by using the covariance matrix. The covariance matrix can be transformed into the corresponding correlation matrix as $P = D_\sigma^{-1} \sum D_\sigma^{-1}$, where $D_\sigma = diag(\sum)^{1/2}$. FA output can be transformed into the associated correlation structure via multiplying the model parameters by functions of the variable standard deviations: if λ_{ij} is a loading from the correlation matrix, then $\lambda_{ij}\sigma_{ii}$ is the corresponding loading from the covariance matrix.

The scale invariance property does not hold for PCA. The PCA models a subspace to capture maximum variability in the data, which can be seen as modeling the covariance structure of the data. Using the covariance matrix to perform PCA makes most sense when the variable scales are similar. Otherwise, if the variables are scaled differently, a correlation matrix would yield more informative results that reflect the relationship between variables, as correlation is insensitive to linear transformation.

c) The code for the implementation of part C is included in *CD.py*. The data is architected based on a matrix of random values R, with 3 random variables exhibiting the following structure:
$X_1 = R_1$
$X_2 = R_1 + 0.001 * R_2$
$X_3 = 100 * R_3$.

The PCA implementation native to the *sklearn.decomposition* library uses the covariance matrix, hence the package was used to demonstrate the results below:

```
cov(data):
[[  1.00036433e+00   1.00036446e+00  -8.68201216e-02]
 [  1.00036446e+00   1.00036460e+00  -8.68071886e-02]
 [ -8.68201216e-02  -8.68071886e-02   9.99492031e+03]]

PCA:
[[ -8.68816361e-06  -8.68686965e-06   1.00000000e+00]
 [  7.07106733e-01   7.07106829e-01   1.22860038e-05]]

Factor Analysis:
[[  1.00017714e+00   1.00017728e+00  -8.67995897e-02]
 [ -4.28921724e-08   8.65951880e-08   9.99690594e+01]]
```

It is clear that the third vector, due to its disproportional scale, dominates the result of performing covariance PCA on the data. Factor analysis, in contrast, captures the structure of the data.

d) The code for the implementation of part C is included in *CD.py*. For this part, the data was constructed based on a matrix of random values R, with 3 random variables exhibiting the following structure:
$X_1 = R_1$
$X_2 = 20 * R_2$
$X_3 = 200 * R_3$.

Factor Analysis factors results in components that exhibit strong correlations between the variables within the factors and weak or zero correlations between the variables across factors. The results are below:

```
cov(data2):
[[  1.22227268e+00  -3.51656764e+00   5.53689631e+01]
 [ -3.51656764e+00   2.39381206e+02   1.03010608e+03]
 [  5.53689631e+01   1.03010608e+03   4.83481402e+04]]

Factor Analysis:
[[   0.2428351     4.54631535  212.42372429]
 [   0.30879583  -14.20399527    0.30363951]]

PCA:
[[ 0.0011429   0.02139719  0.9997704 ]
 [ 0.02172852 -0.99953555  0.02136733]]
```

PCA appears to be more successful in capturing the correlation structure of the data.

e) We have

$$X_l = AY_l + \epsilon \quad \text{for } l \in \{1, 2, ...L\},$$

where
$A = [\alpha_1^{'}, \alpha_2^{'}, ..., \alpha_m^{'}]' - (m \times J)$ matrix;
$\epsilon_l = [\epsilon_{1,l}, \epsilon_{2,l}, ..., \epsilon_{m,l}]$;
Y_l is J-variate covariance-stationary process with $E[Y_l] = \mu_Y$,
$cov[Y_l] = E[(Y_l - \mu_Y)(Y_l - \mu_Y)] =: \Omega_Y$.

Instead of the β-notation, we will use Ψ, declaring ϵ_l as m-variate Gaussian with $E(\epsilon_l) = \bar{0}$,
$var(\epsilon_l) = E[\epsilon_l \epsilon_l^{'}] =: \Psi$, where Ψ is a $(m \times m)$ $diag(\sigma_1^2, \sigma_2^2, ..., \sigma_m^2)$ with $\sigma_i^2 = var(\epsilon_{i,l})$;
$cov[\epsilon_l, \epsilon_l^{'}] = E[\epsilon_l \epsilon_{l'}'] = 0 \ \forall l \neq l'$.
In the standard formulation of the factor analysis model, the factors are orthonormal: $\Omega_Y = I_K$, and zero-mean factors are assumed: $\mu_Y = \bar{0}$. Under these assumptions, the (unconditional) covariance matrix is:

$$cov(X_t) =: \Sigma_X := AA' + \Psi.$$

Let's formulate the joint likelihood of the model:

$$L(\Sigma_X) = p(X_1, ...X_L|\Sigma) = \prod_{l=1}^{L}[p(X_l|\Sigma)] =$$

$$= \prod_{l=1}^{L}[((2\pi)^{-m/2})|\Sigma|^{-1/2} exp(-\frac{1}{2} X_l^{'} \Sigma_X^{-1} X_l] =$$

$$= (2\pi)^{-} Lm/2 |\Sigma|^{-L/2} exp[-\frac{1}{2} \sum_{l=1}^{L} X_l^{'} \Sigma_X^{-1} X_l].$$

Then the log-likelihood of the Factor Model is:

$$l(\Sigma_X) = \log L(\Sigma_X) = -\frac{LJ}{2} log(2\pi) - \frac{J}{2} \log|\Sigma| - \frac{1}{2} \sum_{l=1}^{L} X_l^{'} \Sigma_X^{-1} X_l.$$

The maximum likelihood estimators of the parameters would thus be the values which maximize $l(\Sigma_X)$ where $\Sigma_X = AA' + \Psi$.

As a latent variable is present, one must employ the expectation-maximization method to proceed.

f) The parameters are not uniquely specified. Indeed, factor loadings A and the diagonal matrix of specific variances Ψ are chosen to approximate the covariance/correlation matrix Σ of the data:

$$\Sigma_X \approx AA' + \Psi.$$

One can prove that A is not uniquely defined by considering any rotation matrix R and observing that a matrix product AR would satisfy the condition equally well:

$$AR(AR)' = ARR'A' = AA'.$$

g) Setting all factor-specific variances to be equal to a single value as:

$$\Psi = \sigma_1^2 I$$

essentially transforms the model to be equivalent to a probabilistic PCA.
The first component is constructed to account for the greatest amount of variance in variables, giving the weighting of these variables to form the single component. The second component is then constructed to account for leftover variance not accounted for by the first component. The process continues until as many components as there are original variables are constructed.
MLEs can be obtained in closed form:

$$\sigma_M^2 L = \frac{1}{L-J} \sum_{i=J+1}^{L} \lambda_i,$$

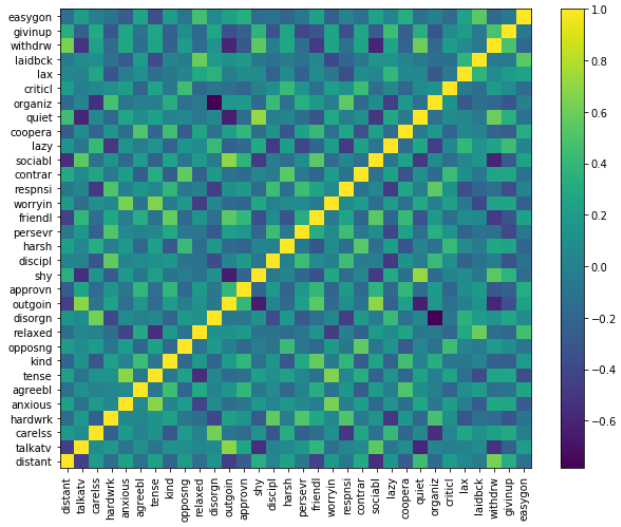
representing the variance lost in the projection averaged over the number of dimensions decreased.

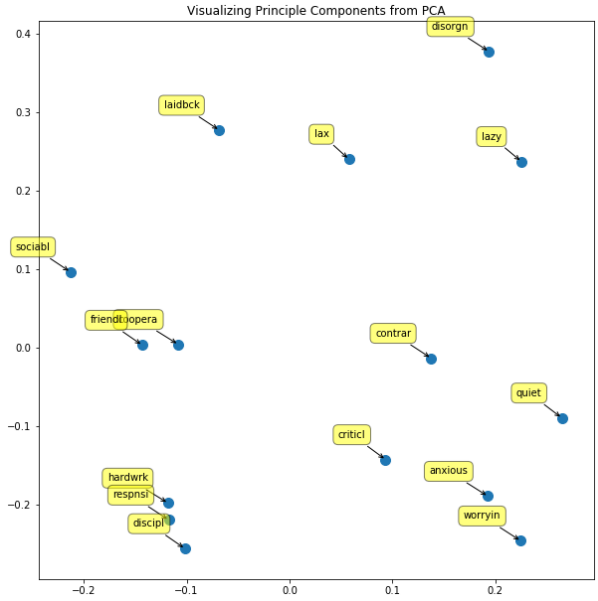
$$A_M L = U_J(\Lambda_J - \sigma^2 I)^{1/2} R$$

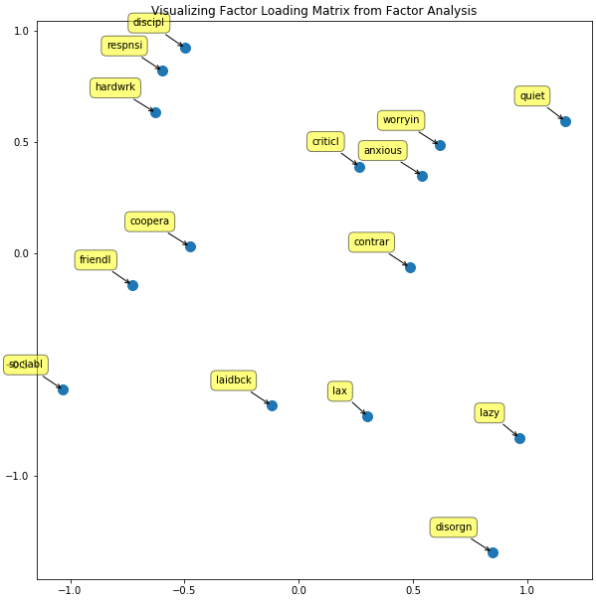
– the mapping of the latent space containing Y to that ?? the principal subspace containing X, where U_J is the matrix of principal eigenvectors of Σ_X, Λ_J is a diagonal matrix of the corresponding eigenvalues, and R is an arbitrary rotation matrix (as discussed before).

Problem 4. *Factor Analysis and PCA on personality data.*

The code that implements the solution os included in the file titled *fa.py*. Below are the images visualizing the output:







One can observe that, up to a reflection, Factor Analysis yields results that are very similar to those of the PCA. This is consistent with what one would expect theoretically. In general, he scale invariance property implies that, unlike the PCA, the FA does not "chase" large-noise features that are uncorrelated with other features.