Inference and Representation, Fall 2017

Problem Set 4: PCA and factor analysis.

Zhuoru Lin
zlin@nyu.edu

Disclaimer: I adhered to NYU honor code in this assignment.

1. Non-Negative matrix factorization

   (a) Likelihood Solution:

   Let $\lambda_{ij} = (WH)_{ij}$. For i, j we have that the probability: $p(X_{ij} = x_{ij}) = e^{\lambda_{ij}} \frac{\lambda_{ij}^{x_{ij}}}{x_{ij}!}$. Let's write $p(X_{ij} = x_{ij})$ as $p(x_{ij})$. The log-likelihood can be calculated as:

   $$\begin{aligned} L_{ij} &= \log(p(x_{ij})) \\ &= \lambda_{ij} + x_{ij} \log(\lambda_{ij}) - \log(x_{ij}!) \end{aligned} \tag{1}$$

   $log(x_{ij}!)$ in equation (1) is independent of W and H. Therefore, we can claim that the likelihood $L(W, H)$ to be:

   $$L(W, H) = \sum_{i,j} x_{ij} \log((WH)_{ij}) - (WH)_{ij} \tag{2}$$

   up to a constant.

(a) By the definition of minorize, we must have:

$$f(x^t) = g(x^t, x^t).$$

Since $x^{t+1} = \arg\max_x g(x, x^t)$:

$$g(x^t, x^t) \leq g(x^{t+1}, x^t).$$

Again by definition of minorize:

$$g(x^{t+1}, x^t) \leq f(x^{t+1})$$

Combine the inequalities above, we have $F(x^t) \leq f(x^{t+1})$.

(b) The concavity of logarithm directly prove the equation by:

$$\begin{aligned} \log(\sum_{k\leq r} y_k) &= \log(\sum_{k\leq r} c_k \frac{y_k}{c_k}) \\ &\geq \sum_{k\leq r} c_k \log(\frac{y_k}{c_k}) \end{aligned} \tag{3}$$

(c) This statement follows directly by setting $c_k = c_{kij}$ and $y_k = w_{ik}h_{kj}$ in equation (3) in section (1b).

$$log(\sum_{k \leq r} w_{ik}h_{kj}) \geq \sum_{k \leq r} c_{kij} \log(\frac{w_{ik}h_{kj}}{c_{kij}}) \tag{4}$$

(d) We first show that $g(W, H; W^t, H^t) \leq L(W, H)$:

$$g(W, H; W^t, H^t) = \sum_{i,j,k} [x_{ij} c_{kij} (\log(w_{ik}) + \log(h_{kj})) - w_{ik} h_{kj}] \tag{5}$$

$$= \sum_{i,j} [x_{ij} \sum_k c_{kij} \log(w_{ik} h_{kj}) - \sum_k w_{ik} h_{kj}] \tag{6}$$

Since $c_{kij} \leq 1$ we must have $\log(c_{kij}) \leq 0$. We also have $x_{ij} \geq 0$ since X is non-negative. Then we can transfer equation (6) into an inequality:

$$g(W, H; W^t, H^t) \leq \sum_{i,j} [x_{ij} \sum_k c_{kij} \log(w_{ik} h_{kj} / c_{kij}) - \sum_k w_{ik} h_{kj}] \tag{7}$$

$$\leq \sum_{i,j} [x_{ij} \sum_k \log(\sum_k w_{ik} h_{kj}) - \sum_k w_{ik} h_{kj}] \tag{8}$$

$$= L(W, H) \tag{9}$$

by equation (4) in section (1c). Next we would need to show $g(W, H; W^t, H^t) = L(W^t, H^t)$ when $WH = W^t H^t$ for some W and H. This is trivial by setting $w_{ik} h_{kj}$ to be non zero only for one k.

(e) By $\frac{\partial g}{\partial w_{ik}} = 0$, we get:

$$\frac{\sum_j x_{ij} c_{kij}}{w_{ik}} - \sum_j h_{kj} = 0 \tag{10}$$

$$w_{ik} = \frac{\sum_j x_{ij} c_{kij}}{\sum_j h_{kj}} \tag{11}$$

$$w_{ik} = w_{ik}^t \frac{\sum_j h_{kj} x_{ij} / (W^t H_{ij}^t)}{\sum_j h_{kj}}. \tag{12}$$

By $\frac{\partial g}{\partial h_{ik}} = 0$, we get:

$$\frac{\sum_j x_{ij} c_{kij}}{h_{kj}} - \sum_j w_{ik} = 0 \tag{13}$$

$$h_{kj} = \frac{\sum_i x_{ij} c_{kij}}{\sum_i w_{ik}} \tag{14}$$

$$h_{kj} = h_{kj}^t \frac{\sum_i w_{ik} x_{ij} / (W^t H_{ij}^t)}{\sum_i w_{ik}}. \tag{15}$$

2. NMF vs PCA on images

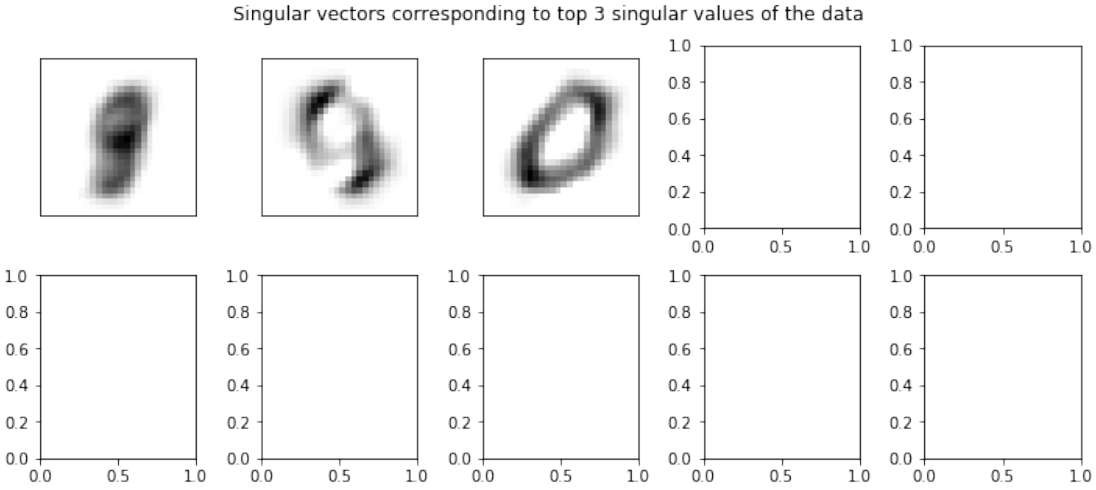

Figure 1: H rows when r=3

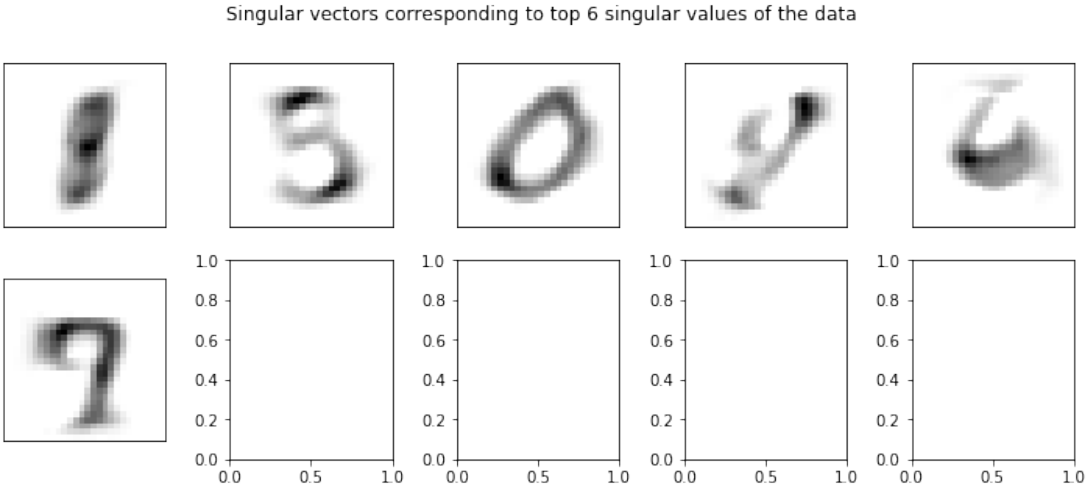

Figure 2: H rows when r=6

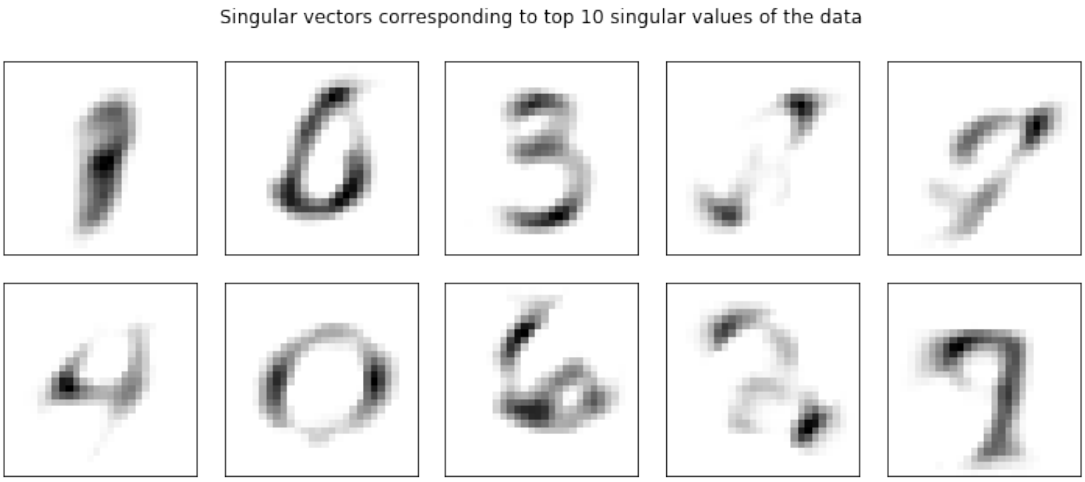

Figure 3: H rows when r=10

(a) I implemented NMF as shown below:

```
def NMF(input_data, r=3, n_iters=10):
    X = input_data.copy()
    N, p = input_data.shape
    # Initialize W and H
    W = np.ones((N, r), dtype=np.float64)
    H = np.ones((r, p), dtype=np.float64)
    for _ in range(n_iters):
        # Update rows of W
        for k in range(r):
            H_rsum = np.sum(H[k,:])
            WH = np.dot(W,H)
            WH[np.where(WH==0)] +=0.001
            if(H_rsum!=0):
                W[:, k] = W[:, k] * np.sum(H[k,:]*(X/WH),1)/ H_rsum
        for k in range(r):
            W_rsum = np.sum(W[:,k])
            WH = np.dot(W,H)
            WH[np.where(WH==0)] +=0.001
            if(W_rsum!=0.0):
                H[k, :] = H[k, :] * np.sum(W[:, k].reshape(N,1)*(X/WH),0)/W_rsum
    return H
```

The results are shown in Figure 1, 2 and 3.

(b) Nearest neighbor search I chose r=13 and the nearest neighbor correspond to the exact same digits as the original images. Results are shown in Figure 4.

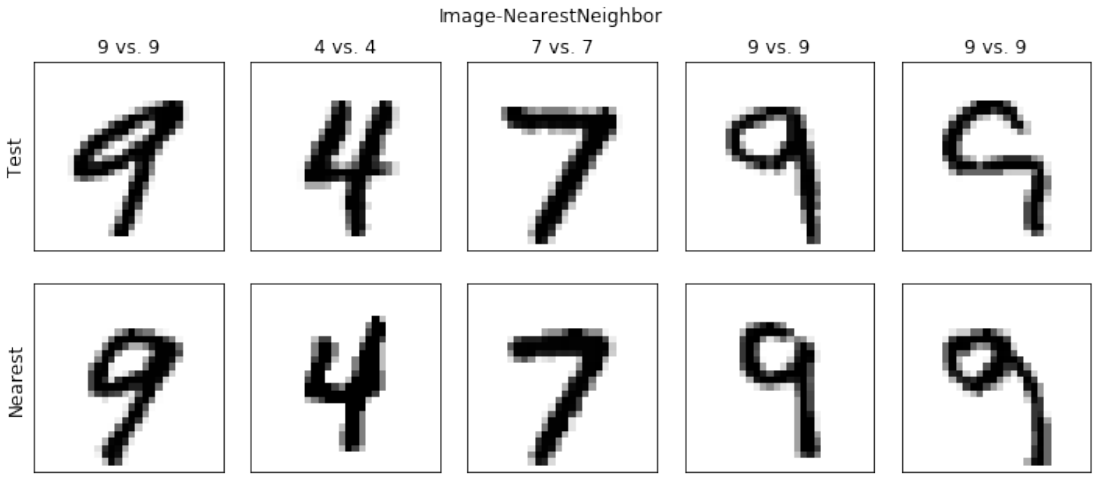

Figure 4: Nearest neighbors when r=13

(c) NMF vs PCA Let's look at the H computed using NMF and PCA and r=10, (Figure 5). The NMF extracts digit-like components while PCA's H are more like the compose of digits.

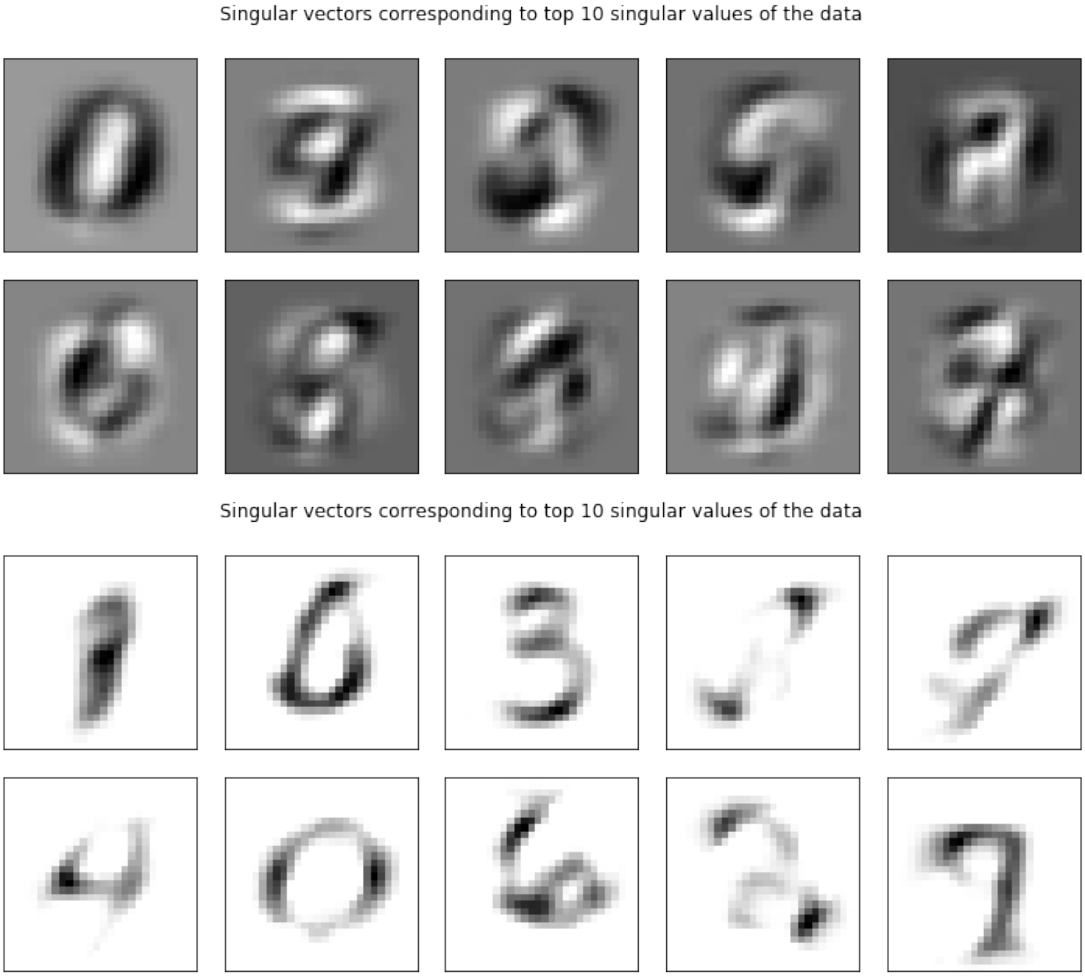

Figure 5: Upper: PCA, Lower: NMF

3. Factor Analysis, Covariance and Correlation

   (a) In order to show this results I need to use two lemmas:
   Lemma 1: $< u, v >= E(uv)$ defines a valid inner product.
   Lemma 2 (Cauchy-Schwartz inequality): $|< u, v >|^2 \leq |u|^2|v|^2$
   Based on these lemmas, the statement $-1 \leq \tilde{\sigma_{i,j}} \leq 1$ follows directly by setting $u = X_i - E[X_i]$ and $v = X_j - E[X_j]$.

(b) Suppose we have $X = AY + \beta$, where X is observed data, Y is latent variables and β is independent noise. By factor analysis we can decomposed the covariance matrix $\Sigma_x as$:

$$\Sigma_x = AA^T + B, \tag{16}$$

where B is the diagonal matrix formed by $B_{ii} = \beta_i$. Suppose we now apply normalization to the data X and denoted the normalized X as $\tilde{X}$. In this case:

$$\begin{aligned}
\Sigma_{\tilde{X}} &= \tilde{\Sigma_x} && (17)\\
&= diag(\Sigma_x)^{-\frac{1}{2}} \Sigma_x diag(\Sigma_x)^{-\frac{1}{2}} && (18)\\
&= diag(\Sigma_x)^{-\frac{1}{2}} (AA^T + B) diag(\Sigma_x)^{-\frac{1}{2}} && (19)\\
&= \tilde{A}\tilde{A}^T + \tilde{B}, && (20)
\end{aligned}$$

where $\tilde{A} = diag(\Sigma_x)^{-\frac{1}{2}} A$ and $B = diag(\Sigma_x)^{-\frac{1}{2}} B diag(\Sigma_x)^{-\frac{1}{2}}$. Therefore we can still retained the same latent variable Y by making some transformation on A. This shows that factor analysis is robust to the change of scale.
However on the other hand, if we use PCA, then:

$$\Sigma_x = AA^T, \tag{21}$$

At the same time we required A to be orthogonal. There fore changing the scale of X will result in the change of Y.

(c) Consider observed data X_1, X_2, X_3 generated by latent variable Z_1, Z_2, Z_3 by:

$$X_1 = Z_1$$
$$X_2 = X_1 + 0.001Z_2$$
$$X_3 = 1000Z_3$$

In this example, the factor analysis will generate leading latent variable that explain the correlation between X_1 and X_2 by constructing latent variable $Z = aX_1 + bX_2$, while the PCA leading latent variable will be aligned with X_3 which generates most variance.

(d) Consider observed data X_1, X_2, X_3 generated by latent variable Z_1, Z_2, Z_3 by:

$$X_1 = Z_1$$
$$X_2 = 0.0001X_1 + 1000Z_2$$
$$X_3 = 10Z_3$$

In this case PCA should be able to catch the variable X_2 that contribute a lot to the covariance of the data, while Factor analysis would fail to catch X_2 with the weak correlation between X_1 and X_2

(e) Without proving (since we already have in DS1003), we should use the fact that if we assume Y follows normal distribution $N(0, I)$ and ϵ follows normal distribution $N(0, \beta)$, then the posterior of $X = AY + \epsilon$ must follow normal distribution $N(0, AA^T + \Sigma_\beta)$, where Σ_β is the covariance matrix of ϵ and $\Sigma_\beta = I\beta$. The posterior is proportional to the Likelihood up to a constant. That is:

$$L(X, A, \beta) \propto N(0, AA^T + \Sigma_\beta)$$

To optimize A and β, we need to take the derivative of log likelihood with respect to A and β and set it to zero.

(f) The parameters found by MLE is not uniquely specified. Since we should have multiple combination of A correspond to the same likelihood of data. Consider a orthogonal matrix R, we must have:

$$ARR^T A^T = AA^T \tag{22}$$

Therefore we can always define $\tilde{A} = AR$ and get the exact same Likelihood:

$$L(X, \tilde{A}, \beta) = L(X, A, \beta) \tag{23}$$

(g) We need to derive the solution to the maximum log-likelihood problem we mentioned before. The derivation is very long, so I borrowed the results from equation (21.2.21), Barber, D. (2012). Bayesian reasoning and machine learning. Cambridge University Press. Let's suppose we are interested in finding a H-dimension approximation of x. The MLE of A is given by:

$$S = \frac{1}{N}(x - \bar{x})(x - \bar{x})^T \tag{24}$$

$$\Sigma_\beta^{-\frac{1}{2}} S \Sigma_\beta^{-\frac{1}{2}} = U \Lambda U \tag{25}$$

$$A = \Sigma_\beta^{\frac{1}{2}} U_H (\Lambda_H - I_H)^{\frac{1}{2}}. \tag{26}$$

The equation (24) is an expression of sample covariance. Equation (25) is a SVD of $\Sigma_\beta^{-\frac{1}{2}} S \Sigma_\beta^{-\frac{1}{2}}$. U is orthogonal eigenvectors and Λ is the corresponding eigenvalues. The MLE of A is given by equation (26), where Λ_H correspond to the H largest eigenvalues and U_H is a matrix formed by corresponding eigenvectors.
Therefore given a known distribution of ϵ and its variance β. We can find the MLE of A by going through the calculation listed above.

4. Factor Analysis vs PCA on personality data.
   The leading factors extracted by factor analysis and the principle components analysis are shown in Figure 6 and 7. This support our former conclusions that factor analysis assign leading factor to those combinations that explain most correlation. In the graph (Figure 6) , those with highest loading factor values are 'Discipline, hardwork, responsible.' which are intuitively very correlated. In PCA, these effect are not observed.

   The following codes are my implementation of factor analysis:

```
def squared_norm(x):
    return np.linalg.norm(x)**2

def svd_top_n(X,n_components):
    _, s, V = np.linalg.svd(X, full_matrices=False)
    return (s[:n_components], V[:n_components],
                        squared_norm(s[n_components:]))

def FactorAnalysis(data, n_components) :
    data = data-data.mean(0)
    n_samples, n_features = data.shape
    nsqrt = np.sqrt(n_samples)

    llconst = n_features * np.log(2. * np.pi) + n_components
    var = np.var(data, axis=0)
    psi = np.ones(n_features)
    loglike = []
    old_ll = -np.inf
    SMALL = 1e-12
    iteration = 10
    tol = 1e-5
    for i in range(iteration):
        sqrt_psi = np.sqrt(psi) + SMALL
        s, V, unexp_var = svd_top_n(data / (sqrt_psi * nsqrt), n_components)
        s **= 2

        W = np.sqrt(np.maximum(s - 1., 0.))[:, np.newaxis] * V
        del V
        W *= sqrt_psi

        # loglikelihood
        ll = llconst + np.sum(np.log(s))
        ll += unexp_var + np.sum(np.log(psi))
        ll *= -n_samples / 2.
        loglike.append(ll)
        if (ll - old_ll) < tol:
            break
        old_ll = ll

        psi = np.maximum(var - np.sum(W ** 2, axis=0), SMALL)
    return(psi,loglike[-1],W.T)
```

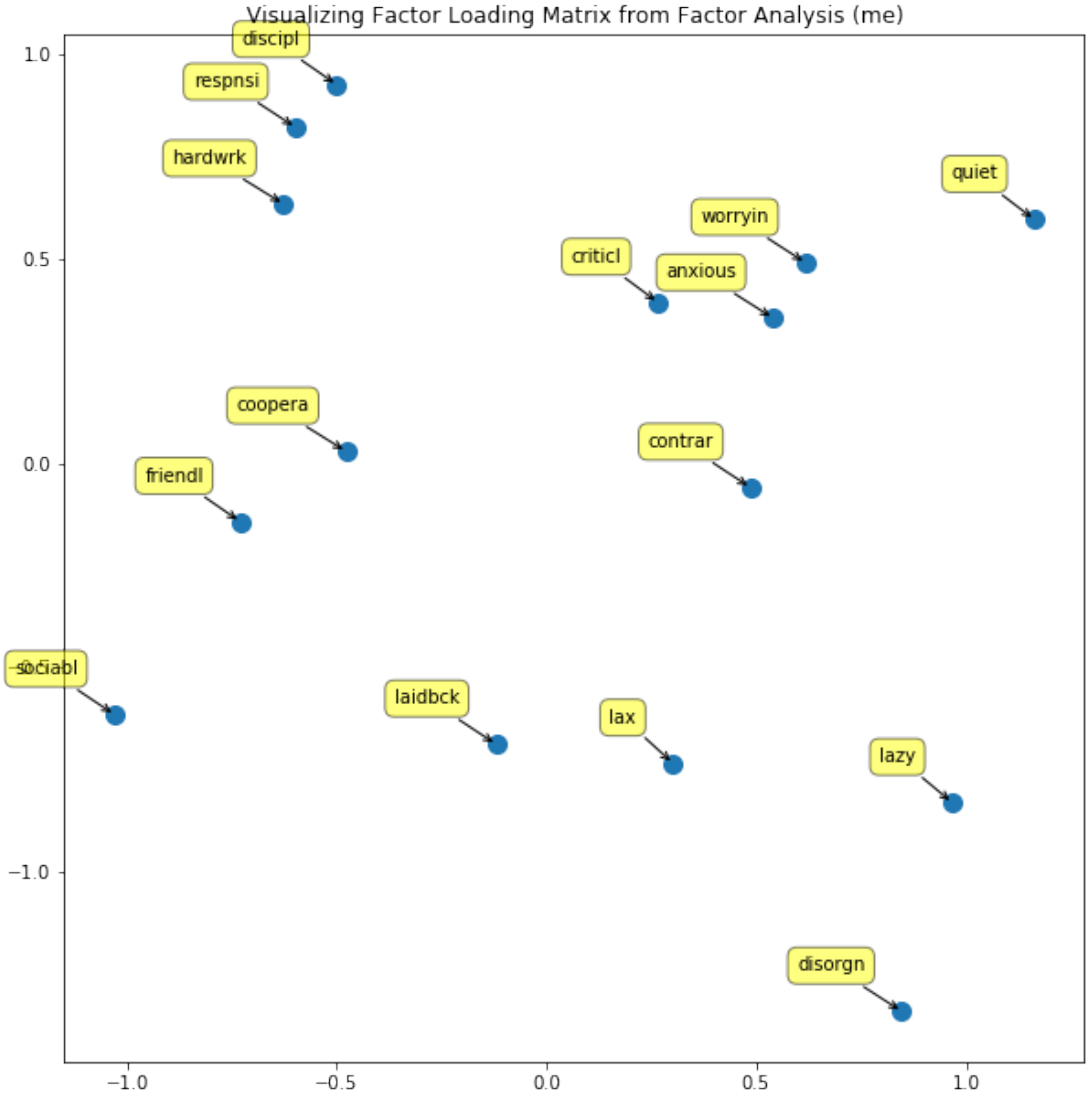

Figure 6: Factor analysis of personality data

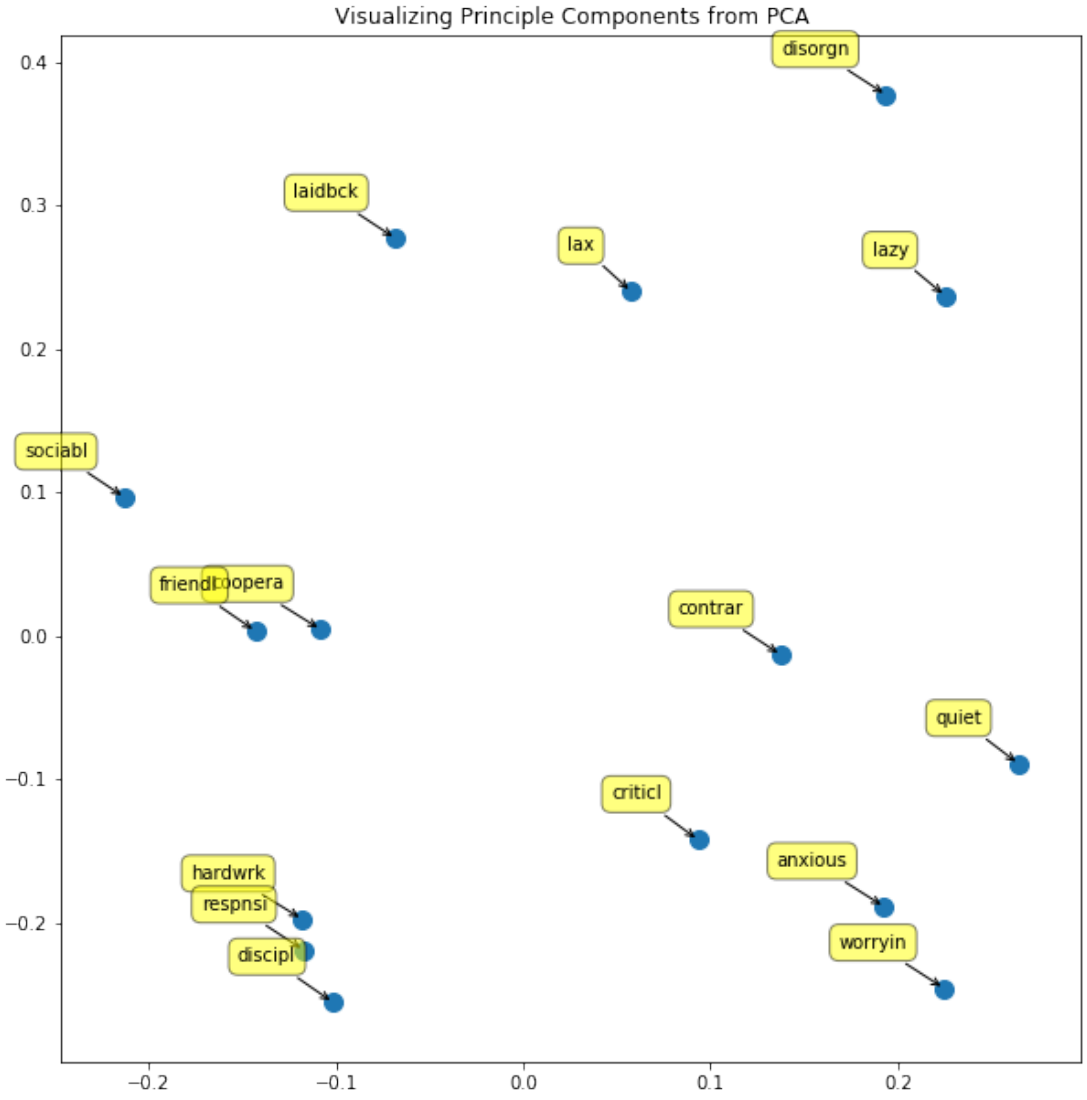

Figure 7: PCA of personality data