## 1 PCA

1. (Judge) In PCA, using the largest eigenvalues ensures that the reconstruction has zero error.
   No (we have zero error only if we use ALL eigenvalues or if the unused eigenvalues are zero)

2. (Judge) Performing SVD on a symmetric matrix has the same outcome as PCA.

   - eigendecomposition: write a matrix into its eigenvalue and eigenvector format. $A = V\Gamma V^{-1}$
   - PCA: principle component analysis. Using eigenvector is only its method, not the essence. The essence is to find the principle components given matrix $X$. Method is to doing eigenvalue decomposition $\frac{1}{N}(X-\overline{X})(X-\overline{X})^T = UDU^T = UDU^{-1}$
   - SVD: doing SVD decomposition of a matrix $A$, means finding $U, D, V$ s.t. $A = UDV^T$, where $D$ is a diagonal matrix, $U$ and $V$ are orthonormal.
     The SVD decomposition is **not unique** for a given matrix. Web link. If $A = U_1 D V_1^T$, then for any unitary diagonal matrix $P$, $A = (U_1 P) D (V_1 P)^T$
   - symmetric matrix: if a matrix $A$is symmetric, doing eigenvector decomposition on it we will have $A = U\Lambda U^{-1} = U\Lambda U^T$.
   - positive semidefinite: not all symmetric matrix is positive semi-definite. Anti-example: $A = \begin{bmatrix} 1 & 0.8 & -0.8 \\ 0.8 & 1 & 0.8 \\ -0.8 & 0.8 & 0.01 \end{bmatrix}$ If a matrix is symmetric, only when all of its eigenvalues are non-negative, it could be a positive semidefinite. For any real invertible matrix $A$, $AA^T$ or $A^TA$ is positive semidefinite.

## 2 SVD

1. (Judge)We first perform SVD on a data matrix $X$. We then rotate the data matrix $X$. The singular vectors of the rotated matrix will be the same.
   No. The singular vectors will also be rotated.
   The geometry interpretation of

   - matrix: transformation. If $X \in \mathbb{R}^{D\times M}$, it does transformation of a vector $v \in \mathbb{R}^M$ to space $\mathbb{R}^D$. The transformation could be: rotate, stretch, squeeze, reflection, etc.
   - eigenvector: unchanged direction under the matrix transformation. $Av = \alpha v$. $A$ is a matrix, $v$ is a vector, $\alpha$ is a number.
   - singular vector: a set of orthogonal basis in original space to another set of orthogonal basis in the new space. $Av_i = \sigma_i u_i$, $UU^T = I, VV^T = I$. Note that, different from eigenvector, which can be find one by one, the singular vectors must be a complete set, which should satisfy the orthogonal condition, not only the direction keeping.

2. Doing SVD for matrix $A$, we have $A = UDV^T \Rightarrow rank(A) = rank(D)$

## 3 GMM

1. (Judge) EM algorithm is faster than K-means algorithm
   No. EM is much slower.

2. (Judge)GMM clustering is equivalent to using soft assignments instead of K-means hard assignments.
   No. soft assignments is a more general notion than GMM, which includes more than this particular mixture model

### 3.1 Naive solution

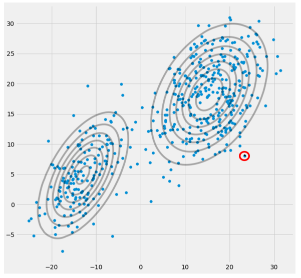
Try to solve the parameter $\lambda_j$ directly from the original log-likelihood. Then, we will find the solution of $\lambda_j$ depending on all the other $\lambda_k$, because they are inside of the log and cannot be divided by deriving. That's why we turn to EM to extract the sum of $\lambda_k$ from inside of log to outside, further to seperate them.

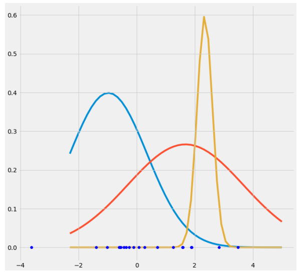
### 3.2 Compared with K-means

Refer to the Table. 1

### 3.3 Singularity Problem

Detailed explanation is given by 2Obe





Singularity means the covariance matrix of one Gaussian model is singular. Singular matrix means invertible, the determinant of which is 0. One common case of a singular covariance matrix in Gaussian model is the $\Sigma = 0 \cdot I$. This can happen when a Gaussian model is fitted on only one node.

**Why could it be a problem?** It will be a problem if we judge our clustering according to the objective function. As the picture shows, if there are actually two clusters, but we use three clusters to fit the points, one cluster could be singular, making the objective function of 3 clusters infinite, much larger than that of 2 clusters. Judging from the objective function's value, we will think 3-clustering is better. Since the infinity, it makes the maximization of the log likelihood function not a well posed problem.

**Don't impact significantly the EM fitting**. The singularity only impacts the likelihood. But in our EM training of parameters, the likelihood of a data point $P(x;\theta_i)$ only impacts $\{\pi_i\}$ and $\gamma_{nk}$.
$\gamma_{nk} = \frac{\pi_k p(x_n;\theta_k)}{\sum_l \pi_l p(x_n;\theta_l)} \to 1$, if $x_n$ is the "central"point of the singular $k$ Gaussian.
$\pi_i = \frac{\sum_n \gamma_{nk}}{N}$, one $\gamma_{nk} = 1$ not making big influence.

**Not appear in the single Gaussian model** See from the right-side picture. Supposing we only have one Gaussian model and it is singular, denoted by the yellow line, the likelihood of all the other nodes $P(x) = P(x;\theta)$ except the "centeröne will be zero. Hence, the likelihood of the whole dataset will be $0 = \infty \cdot 0 \cdots$.
In multiple Gaussian models, supposing we add the blue line and red line to represent another two Gaussians, the likelihood of point $P(x) = \sum_{i=1}^K \pi_i P(x;\theta_i)$ could be larger than 0 (Because these nodes are very likely to appear under other Gaussians.) Hence, the likelihood of the whole dataset will be $\infty = \infty \cdot a_1 \cdot a_2 \cdots, a_i > 0$

**Heuristic avoiding strategy**:
- resetting the mean of the collapsed Gaussian component as another value, e.g. a randomly chosen value;
- resetting the covariance of the collapsed Gaussian component to some large value.

## 4 K-means

1. Show intuitively that K-means always terminates, i.e. converges in a finite number of steps.
Hard-assignments $\Rightarrow$ a finite number of possible cluster assignments $\Rightarrow$ the algorithms must enter into a cycle at some point.
Non-increase of the objective function $\Rightarrow$ the cycle has length 1.

**Non-convexity**

The objective function $J = \sum_{i=1}^N \sum_{j=1}^K z_{ij} \|\mathbf{x}_i - \mathbf{u}_j\|_2^2$ is not convex because $\sum_{j=1}^K z_{ij}\|\mathbf{x}_i - \mathbf{u}_j\|_2^2 = \min_j \|\mathbf{x}_i - \mathbf{u}_j\|_2^2$ is not convex.

## 5 AIC and BIC

1. (Judge)BIC score cannot be smaller than the AIC score for large enough databases and fixed number of free parameters.
   Yes. The difference of AIC and BIC is:

   - AIC: $\kappa(\theta)$
   - BIC: $\frac{1}{2}\kappa(\theta)\ln(N)$

   Fixed number of free parameters means $\kappa(\theta)$ is fixed, while large enough database means $\ln(N)$ is large.

## 6 NMF

1. (Judge) Consider NMF for clustering a set of datapoints $X$. In the solution to the NMF problem, every datapoint is assigned to at most one cluster.
   No. This statement means NMF is a hard assignment. But it is a **soft** assignment, assigning the probability.

## 7 Neural Network

**MLP (multi-layer perceptron) VS Logistic Regression**
- Logistic: linear functions on inputs
- MLP: learn intermediate feature representations.

**Regularization**

1. add penalty for large parameters: $\mathcal{L}_\lambda(\theta;\mathcal{X}) = \frac{1}{N}\sum_{i=1}^N l(y_i, \hat{y}_i) + \frac{\lambda}{2}\|\theta\|_2^2$
2. dropout. Refer to this post

- To approximate training a large number of neural networks with different architectures in parallel.
- To make the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs.

**CNN**

**Receptive fields** to extract localized features. One field correspondents to one feature. Realized in convolution layer by defining the size and parameters of the padding.
**Translation invariance** Invariance means that you can recognize an object as an object, even when its appearance varies in some way.Realized by convolution and pooling.
**Weight sharing** Realized by convolution layer, defining the filter mask (receptive fields).

**Multi-layer convolution calculation** A 2D convolutional kernel can be represented as a 4D tensor ($C_{out}$, $C_{in}$, height, width), where $C_{out}$ is the number of output channels (i.e. the number of filters) and $C_{in}$ is the number of input channels (the dimensionality of the input).
**Example** $C_{out} = L, C_{in} = C$, height = width = $2k+1$, $I_c$ denotes the $c$th-input matrix, $K^l$ denotes the $l$th-layer filter mask.
Then the $l$-th output matrix is
$(I \star K^l)_{i',j'} = \sum_{1\leq c\leq C}^{max} \sum_{-k\leq i,j\leq k} (I_c)_{i'+i,j'+j}(K^l_c)_{i,j}$, where $(I_c)_{a,b} = 0$ for $(a,b)$ outside the range of the original $c$th-input matrix.

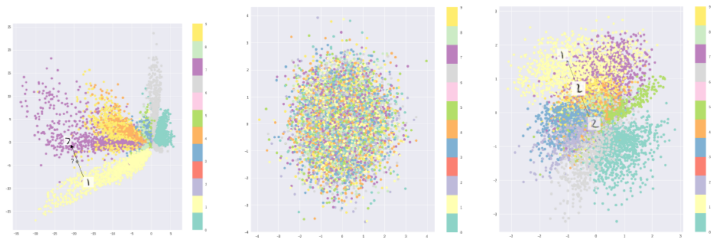
## 8 Generative Models

**Deep generative models**

- Deep means "deep neural network". We use DNN to learn a distribution which is as similar as possible to the true data distribution.
- The problem setting is: we want to learn any kind of data distribution using **unsupervised** learning.
VAE and GAN are typical examples of deep generative models.

**VAE (Variational autoencoder)**

**Motivation** enforce a structure on the latent space at the expense of reconstruction error for **generating new samples and interpolation**. **Latent variable model** representing data into low-dimensional latent variables (e.g. clustering is the discrete version), and reconstruct from it by minimizing the reconstruction error.
**Characteristic**:
- continuous: in clustering, the latent variables are discrete, e.g. the center of each cluster; here, latent variable $h$ is continuous. Instead of using $P(h=k)$, we will use $f(h)$.
- variational: don't expect the exactly same point as the input, but randomly sample from the latent space.
**Creative point** Usually, we model on the latent variable space. What VAE does is not making assumption on $q(h|x)$ (furtherly $f(h)$) but do the variable replacement. Using $f(x,\epsilon) \sim q(h|x)$, where $\epsilon$ follows a known simple distribution.
It will be helpful to learn VAE (a continuous version) by comparing with mixture clustering model (a discrete version). See the Table. 2
**ELBO** $\mathbb{E}_{q_\phi(h|x)} \log p(x|h) - D[q_\phi(h|x) \| p_\theta(h)]$
- $\mathbb{E}_{q_\phi(h|x)} \log p(x|h)$: try to maximize this item, the likelihood of generating $\mathbf{x}$, which is related to **reconstruction quality**.
- $D[q_\phi(h|x) \| p_\theta(h)]$: try to minimize this item, that is making posterior distribution to be close to the prior distribution, which acts as a **regularizer**.


Reconstruction loss only (normal autoencoder) | KL loss only (meaningless Gaussian samples) | Both (variational autoencoder)

**Maximization log-likelihood procedure**

1. solve $q_\phi$: solving ELBO to make the approximate posterior distribution close to the latent variable prior.
2. sample $\mathbf{z}$: $\mathbf{z} \sim q_\phi$, and pass it to the decoder network.
3. sample $\hat{x}$, by maximize the reconstruction quality with respect to the original input.

we can use SGD to train the VAE model.

**GAN(Generative adversarial network)**

**Objective** $\min_G \max_D \mathbb{E}_{x\sim P_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1 - D(G(z)))]$
- $D(x)$: the probability that $x$ came from the data rather than $p_z$.
- Discriminator: to **maximize** the probability to assign correct label to both training examples and samples from $G$.
- Generator: to **minimize** the probability of correct labeling.

**GAN VS VAE**

**VAE**

- quality: blurry images. 1. caused by pixel-wise factorization and local loss; 2. high-frequency details are poorly correlated and hard to predict.
- training: somewhat easier but not easy.
- applications: learn an explicit density, generate and encode.

**GAN**

- quality: sharper images. Because the discriminator learns a "perceptual"loss.
- training: very hard to train. Architecture and hyperparameters play key roles.
- applications: learn an implicit density, only for generating.

## 9 Orthogonal Matrices and Bases

**Relationships**

Considering the following conditions:

1. linearly independent
2. span the space
3. $\|\mathbf{u}\|_2 = 1$
4. $\langle \mathbf{u}, \mathbf{v} \rangle = 0, \forall \mathbf{u} \neq \mathbf{v}$

The following concepts satisfy different conditions:
**Basis**: 1,2.
**Orthogonal**: 4.
**Orthogonal basis**: 1,2,4
**Orthonormal basis**: 1,2,3,4

1. Goodness of orthogonal bases:
   - explicit formula for the projections under the basis $\mathbf{U}$: $U^T x$
   - Energy preservation: $\|x\|_2 = \|Ux\|_2$
   - (Quick inversion?): $U^{-1} = U^T$. computing inverse transform is efficient.
2. (Judge) Using overcomplete dictionary yields sparser solutions.
   False. Hopefully, it yields sparser solutions but not necessarily.

## 10 SGD

Gradient Descent:

1. Aim: to **Minimize** an objective, not to maximize, which is usual for (log-)likelihood and common strategy is EM.
2. Update: using minus not plus. $x^{(k+1)} = x^{(k)} - \gamma \nabla f(x^{(k)}) = x^{(k)} - \gamma \Sigma_i \nabla f_i(x^{(k)})$

$\nabla f(x)$ is a sum of several $f_i(x^{(k)})$. In SGD, we randomly choose $i \Rightarrow x^{(k+1)} = x^{(k)} - \gamma \nabla f_i(x^{(k)})$

## 11 Dictionary Learning

**Computation efficiency of different transform**

- via matrix multiplication : $\mathcal{O}(D^2)$
- Fourier basis : $\mathcal{O}(D \log D)$
- Wavelet basis: $\mathcal{O}(D)$ or $\mathcal{O}(D \log D)$

**Dictionary selection strategy**

- manually by inspecting signals.
- try several dictionaries, and choose the one giving the sparest coding.
- from multiple basis: using algorithms to pick several atoms from different dictionaries and compose as a new one.

## 12 Summary

**Models comparison**

See the Table. **??**, 4 and 5.
Some explanations: **Category**:

- Dimension reduction (DR): what we are interested in is the latent lower dimension space, e.g. the topics in word*document context, the features for collaborative filtering.
- Matrix reconstruction(MR): what are we interested in is the unknown items in the matrix. However, we may use dimension reduction's methods but that is not our purpose.

| | K-means | GMM |
|---|---|---|
| Objective function | squared Euclidean norm: $\sum_{i=1}^{N} \Vert x_i - \mu_{z_i} \Vert_2^2$ | log-likelihood $\sum_{i=1}^{N} \log(\pi_{z_i} \mathcal{N}(x_i; \mu_{z_i}, \Sigma_{z_i}))$ |
| Prior | uniform prior. $P_z(k) = \frac{1}{K}$ | $\{\pi_k\}$, $P_z(k) = \pi_k \Rightarrow P(z) = \prod_j \pi_j^{z_j}$ |
| Runtime | faster | slower |

Tabelle 1: GMM VS K-means

| | Mixture Clustering | VAE |
|---|---|---|
| Latent variable | Prior: $\pi_j \sim \text{Categorical}(\pi)$<br>Posterior: $Pr(z_j = 1 \mid x) = \frac{\pi_j p(x;\theta_j)}{\sum_{l=1}^{K} \pi_l p(x;\theta_l)}$ | Prior: $p(h)$<br>Posterior: $p(h \mid x)$ |
| Lower Bound of the log-likelihood (for one data point) | $\log p(x;\theta) = \log(\sum_{j=1}^{K} \pi_j p(x;\theta_j)) \geq \sum_{j=1}^{K} q_j[\log p(x;\theta_j) + \log \pi_j - \log q_j]$<br>- $q_j$ is to approximate $Pr(z_j = 1 \mid x)$<br>- $p(x;\theta_j) = p(x;\Theta \mid \pi_j)$ | $\log p(x;\theta) = \log \int p(x,h)dh \geq \int q(h \mid x)[\log p(x \mid h) + \log p(h) - \log q(h \mid x)]dh$<br>- $q(h \mid x)$ is to approximate $p(h \mid x)$<br>- $B(q,x)$ is defined by the right-side. Can be also rewritten as $\int q(h \mid x) \log p(x \mid h)dh - D[q(h \mid x) \Vert p(h)]$<br>- $D[q(h \mid x) \Vert p(h)]$ KL-divergence.[1] |
| Optimization objective | - $\hat{\pi}$ (achieved by $q_{ij}$),<br>- $\hat{\Theta}$ | - $q(h \mid x)$ denoted by parameter $\phi : q_\phi(h \mid x)$,<br>- $p(x \mid h)$ denoted by $\theta : p_\theta(x \mid h)$ |
| EM update: Expectation (Inference: find the probability of latent variable) | - $\boldsymbol{\pi}_j^* := \frac{1}{N} \sum_{i=1}^{N} q_{ij}^*$<br>- $q_{ij}^* = \frac{\pi_j p(x_i;\theta_j)}{\sum_{l=1}^{K} \pi_l p(x_i;\theta_l)}$ | $\phi : q_\phi(h \mid x)$ given $\theta$ ($p_\theta$). $\nabla_\phi \mathbb{E}_{q_\phi}[\mathcal{L}(\mathbf{x},\mathbf{h})]$<br>Re-parameterization trick: $q_\phi(h;x) = g_\phi(\zeta;x)$, $\zeta$ ~simple distribution, e.g. $\zeta \sim \mathcal{N}(0,I), h = \mu + U\zeta \Rightarrow h \sim \mathcal{N}(\mu, UU^\top)$<br>Reinforce trick: using $\nabla q = q \nabla \log q$ :<br>$\nabla_\phi \mathbb{E}_{q_\phi}[\mathcal{L}(\mathbf{x},\mathbf{h})] = \mathbb{E}_{q_\phi}[\mathcal{L}(\mathbf{x},\mathbf{h}) \nabla_\phi log q_\phi(\mathbf{h};\mathbf{x})]$ |
| EM update: Maximization (Generation: find the probability of $x$) | $\Theta : \{\theta_j\}$ (Gaussian version)<br>- $\boldsymbol{\mu}_j^* := \frac{\sum_{i=1}^{N} q_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} q_{ij}}$<br>- $\Sigma_j^* = \frac{\sum_{i=1}^{N} q_{ij}(\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^\top}{\sum_{i=1}^{N} q_{ij}}$ | $\theta : p_\theta(x \mid h)$ given $q_\phi(h \mid x)$<br>Stochastic approximation: $\nabla_\theta \mathbb{E}_{q_\phi}[\mathcal{L}(\mathbf{x},\mathbf{h})] \simeq \frac{1}{L} \sum_{r=1}^{L} \nabla_\theta \log p_\theta(\mathbf{x} \mid \mathbf{h}^{(r)})$, $\mathbf{h}^{(r)} \sim q_\phi(\cdot \mid \mathbf{x})$, i.i.d<br>In the log-likelihood context: $\mathbb{E}_{q_\phi}[\mathcal{L}(\mathbf{x},\mathbf{h})] = B(q_\phi, \mathbf{x})$ |

Tabelle 2: VAE VS Mixture clustering model

| Models | SVD | PCA | Matrix Reconstruction | pLSA | LDA |
|---|---|---|---|---|---|
| Category | DR, MR | DR, MR | MR | DR | DR |
| Objective | $\min \lVert A-X\rVert_F^2$ or $\min \lVert A-X\rVert_2$ | $\min \lVert A-X\rVert_F^2$ or $\min \lVert A-X\rVert_2$ by finding principle components $\mathbf{u}$<br>$\mathbf{u} \leftarrow \arg\max[\mathbf{u}^\top \Sigma \mathbf{u}] = \arg\max[\mathbf{u}^\top(\frac{1}{n}\sum_i \mathbf{x}_i\mathbf{x}_i^\top)\mathbf{u}]$ | $\min f(\mathbf{U},\mathbf{V}) = [\sum_{i,j\in\mathcal{I}}(a_{ij} - \langle \mathbf{u}_i, \mathbf{v}_j\rangle)^2] = \min \lVert A-B\rVert_{\mathbf{G}}^2$ | $\max \mathcal{L}(\mathbf{U},\mathbf{V}) = \sum_{i,j} x_{ij} \log \sum_{z=1}^K v_{zj}u_{zi} \geq \sum_{i,j} x_{ij}[\sum_{z=1}^K q_{zij}(\log u_{zj} + \log u_{zi} - \log q_{zij})]$ | $\max p(\mathbf{x},\mathbf{V},\alpha) = \int p(\mathbf{x}\mid\mathbf{V},\mathbf{u})p(\mathbf{u}\mid\alpha)d\mathbf{u}$ |
| Constraints | $\text{rank}(X) = k$ | $\lVert\mathbf{u}\rVert_2 = 1$ | $\text{rank}(\mathbf{B}) = k$ and $\mathcal{I} = \{(i,j) :$ observed$\}$ | - $v_{zj} \geq 0, \sum_z v_{zj} = 1$<br>- $u_{zi} \geq 0, \sum_z u_{zi} = 1$ | $p(\mathbf{u}_i\mid\alpha) \propto \prod_{z=1}^K \mathbf{u}_{zi}^{\alpha_z - 1}$ |
| Parameters | -known: $A \in \mathbb{R}^{M\times N}, k \in \mathbb{Z}$<br>- unknown $X \in \mathbb{R}^{M\times N}$ | $\Sigma = \frac{1}{N}XX^\top \in \mathbb{R}^{m\times m}$ | $\mathbf{G}$: binary, denoting whether the entry has been observed. | $v_{zj} = p(w_j\mid z), u_{zi} = p(z\mid d_i)$ | -$\mathbf{V}$: real parameters<br>- $\mathbf{U}$: can be re-constructed. $p(x\mid V,u) = \text{Multi}(x\mid\pi), \pi_j := \sum_z v_{zj}u_z, \pi_j = \sum_z v_{zj}u_z$ |
| Convexity | non-convex: space: rank$= k$ | non-convex: space: rank$= k$ | non-convex jointly on $\mathbf{U}$ and $\mathbf{V}$, but convex specifically | non-convex | |
| Algorithm | $A = UDV^\top \Rightarrow X = U_kD_kV_k^\top$ $U \in \mathbb{R}^{M\times D}, V \in \mathbb{R}^{D\times N}, D$ : Diagonal | - (one by one): Lagarange multiplier $\mathcal{L}(\mathbf{u},\lambda) = \mathbf{u}^\top\Sigma\mathbf{u} + \lambda\langle\mathbf{u},\mathbf{u}\rangle, \nabla_{\mathbf{u}}\mathcal{L} = 0 \Rightarrow \Sigma\approxeq = \lambda\approxeq$, then works on $\Sigma' = \Sigma - \lambda\mathbf{u}\mathbf{u}^\top$<br>- (one by one, power iteration) $\mathbf{v}_{t+1} = \frac{\mathbf{A}\mathbf{v}_t}{\lVert\mathbf{A}\mathbf{v}_t\rVert}, \lim_{t\to\infty}\mathbf{v}_t \to \mathbf{u}_1$<br>- $\Sigma = \mathbf{U}\Lambda\mathbf{U}^\top \Rightarrow \tilde{\mathbf{X}} = \mathbf{U}_d\mathbf{U}_d^\top\mathbf{X}$ | alternating minimization (ALS: alternating least square) | EM(Expectation Maximization)+latent variable | Variational EM, MCMC |
| Parameter update | once | Once<br>$\mathbf{U}$: eigenvectors of $(\frac{1}{N})\mathbf{A}\mathbf{A}^\top$<br>$\mathbf{V}$: eigenvectors of $(\frac{1}{N})\mathbf{A}^\top\mathbf{A}$<br>$\Lambda = \frac{1}{N}DD^\top$, $\Lambda' = \frac{1}{N}D^\top D$, (identical up to zero padding) | repeat until convergence<br>- $\mathbf{U} \leftarrow \min_{\mathbf{U}} f(\mathbf{U},\mathbf{V})$<br>- $\mathbf{V} \leftarrow \min_{\mathbf{V}} f(\mathbf{U},\mathbf{V})$ | repeat until convergence<br>- (Expectation step) $q_{zij} = \frac{u_{zi}v_{zj}}{\sum_{k=1}^K u_{ki}v_{kj}} = \frac{p(w_j\mid z)p(z\mid d_i)}{\sum_{k=1}^K p(w_j\mid k)p(k\mid d_i)}$<br>- (Maximization step) $u_{zi} = \frac{\sum_j x_{ij}q_{zij}}{\sum_{zj} x_{ij}q_{zij}} = \sum_j x_{ij}, v_{zj} = \frac{\sum_i x_{ij}q_{zij}}{\sum_{i,l} x_{il}q_{zil}}$ | |
| Interpretation | -$U$: (items, feature) matrix. $U_{ij}$: item $i$'s value on feature $j$.<br>-$V$: (users, feature) matrix. $V_{ij}$: user $i$'s value on feature $j$.<br>- $D_i$: the significant level of feature $i$ | $\mathbf{u}_i$: the $i$-th principle component<br>$\lambda_i$: the $i$-th biggest eigenvalue of $\frac{1}{N}\mathbf{X}\mathbf{X}^\top$ | | $q_{zij} := Pr(Q_{zij} = 1)$ | Dirichlet($\alpha$): the objective distribution of topics (latent)<br>$\mathbf{u}_i$: the topic distribution of the document $d_i$ |
| Practical consideration | find suitable $k$ by knee point | Need to center the data first | | | |
| Others | Reconstruction error:<br>- Euclidean norm $d_{k+1}$<br>- Frobenius norm $\sum_{i=k+1}^D d_i^2$ | Reconstruction error: $\frac{1}{N}\lVert\tilde{\mathbf{X}} - \mathbf{X}\rVert_F^2 = \frac{1}{N}\sum_{i=1}^N \lVert x_i - \tilde{x}_i\rVert_2^2 = \sum_{i=K+1}^D \lambda_i$ | Convex relaxation: nuclear norm | EM: guaranteed convergence, not guaranteed to global optimum | Generative model:<br>1. for $d_i$, sample $u_i \sim$ Dirichlet($\alpha$)<br>2. for word slot $t$: (1)sample topic $z^t \sim \text{Multi}(u_i)$; (2) sample word $w^t \sim \text{Multi}(v_{z^t})$ |

Tabelle 3: Models Comparison

| Models | GMM | K-Means | VAE | GAN |
|---|---|---|---|---|
| Category | DR, clustering | DR, Clustering | Generate, encode | Generate |
| Objective | $\max_{\pi,\Theta} \log p_\theta(\mathbf{X}) = \log \prod_{n=1}^N p_\theta(\mathbf{x}_n) = \sum_{n=1}^N \log \sum_{k=1}^K \pi_k p_{\theta_k}(\mathbf{x}_n)$ | $\min J(\mathbf{U},\mathbf{Z}) = \lVert\mathbf{X} - \mathbf{U}\mathbf{Z}^\top\rVert_F^2$ | $\max \mathrm{ELBO} = \max \mathbf{E}_{q_\phi}[\log p_\theta(x\|z)] - KL(q_\phi(z\|x)\Vert p_\theta(z))$ | $\min_G \max_D \mathbf{E}_{x\sim P_{data}(x)}[\log D(x)] + \mathbf{E}_{z\sim P_z(z)}[\log(1-D(G(z)))]$ |
| Constraints | $\pi_j \geq 0, \sum_{j=1}^K \pi_j = 1$ | $z_{ij} \in \{0,1\}, \sum_{j=1}^K z_{ij} = 1$ | | |
| Parameters | $\Pr(z_j = 1) = \pi_j$ or $p_\pi(\mathbf{z}) = \prod_{j=1}^K \pi_j^{z_j}$ | $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_N] \in \mathbb{R}^{D\times N}$, data matrix $\mathbf{U} = [\mathbf{u}_1 \cdots \mathbf{u}_K] \in \mathbb{R}^{D\times K}$, centroid matrix | ELBO: $\log p_\theta(x) = \mathbb{E}_{h\sim q_\phi}[\log p_\theta(x)] = \mathbb{E}_h[\log p_\theta(x\|h)] - D[q_\phi(h\|x)\Vert p_\theta(h)] + D[q_\phi(h\|x)\Vert p_\theta(h\|x)]$ | |
| Convexity | no closed-form solutions | non-convex | | |
| Algorithm | EM (Expectation Maximization)+latent variable | Alternating minimization (EM) | | SGD |
| Parameter update | Expectation step: $q_j^* = \frac{\pi_j p(x;\theta_j)}{\sum_{l=1}^K \pi_l p(x;\theta_l)}$ $\boldsymbol{\pi}_j^* := \frac{1}{N}\sum_{i=1}^N q_{ij}^*$ Maximization step: $\boldsymbol{\mu}_j^* := \frac{\sum_{i=1}^N q_{ij}\mathbf{x}_i}{\sum_{i=1}^N q_{ij}}$, $\Sigma_j^* = \frac{\sum_{i=1}^N q_{ij}(\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^\top}{\sum_{i=1}^N q_{ij}}$ | $z_{ij} = 1, j = \arg_k \min \lVert x_i - u_k\rVert^2$ (closed centroid) $\mathbf{u}_j = \frac{\sum_{i=1}^N z_{ij}\mathbf{x}_i}{\sum_{i=1}^N z_{ij}}$ | | $\theta^{t+1} = \theta^t - \eta\nabla_\theta l(\theta^t, \phi^t)$, minimize: minus $\phi^{t+1} = \phi^t + \eta\nabla_\phi l(\theta^{t+1}, \phi^t)$, maximize: plus |
| Interpretation | Using Bayes rule: $q_j^* = \Pr(z_j = 1\|\mathbf{x})$ E-step: compute probabilistic assignments of points to clusters (keeping their location and shape fixed) M-step: recompute optimal cluster locations and shapes, given probabilistic assignments | | | -$\theta$: Generator, $\tilde{p}_\theta(\mathbf{x}, y=1) = p(y=1)\cdot p(\mathbf{x}), \tilde{p}_\theta(\mathbf{x}, y=0) = p(y=0)\cdot p_\theta(\mathbf{x})$ - $\phi$: Discriminator. $q_\phi : \mathbf{x} \mapsto [0;1]$ |
| Practical consideration | takes many more iterations to reach convergence, each cycle requires significantly more computation. | Faster than GMM. | Stochastic approximation: $\nabla_\theta \mathbf{E}_{q_\phi}[\mathcal{L}(\mathbf{x},\mathbf{h})] \simeq \frac{1}{L}\sum_{r=1}^L \nabla_\theta \log p_\theta(\mathbf{x}\|\mathbf{h}^{(r)})$, $\mathbf{h}^{(r)} \sim q_\phi(\cdot\|\mathbf{x})$, i.i.d Re-parametrization trick $q_\phi(h;x) = g_\phi(\zeta;x)$, $\zeta$ ~simple distribution Reinforce trick: $\nabla_\phi \mathbf{E}_{q_\phi}[\mathcal{L}(x,z)] = \mathbf{E}_{q_\phi}[\mathcal{L}(\mathbf{x},\mathbf{z})\nabla_\phi \log q_\phi(\mathbf{z};\mathbf{x})]$ | |
| Others | K-means algorithm can be used to nd a good initialization | K-means++: initialization strategy; Coresets: subset strategy | tend to generate blurry images somewhat easier to train | generate sharper image very hard to train. |

Tabelle 4: Models Comparison

| Models | Latent Vector | GLoVe | Signal reconstrution | Dictionary Learning |
|---|---|---|---|---|
| Category | Word Embedding | Word Embedding | Encode | Encode |
| Objective | $\max \mathcal{L}(\theta; \mathbf{w}) = \sum_{t=1}^{T} \sum_{\triangle \in \mathcal{I}} \log p_\theta(w^{(t+\triangle)} \| w^{(t)})$ | $\min \mathcal{H}(\theta; \mathbf{N}) = \sum_{(i,j)} f(n_{ij})(\log n_{ij} - \log \tilde{p}_\theta(w_i \| w_j))^2$<br>with $f := 1, \Leftrightarrow \min \\|\mathbf{N} - \mathbf{X}^\top \mathbf{Y}\\|_F^2$ | $\min \\|\mathbf{z}\\|_0$ | $\min_{\mathbf{U},\mathbf{Z}} \\|\mathbf{X} - \mathbf{U} \cdot \mathbf{Z}\\|_F^2$ |
| Constraints | | | $\mathbf{x} = \mathbf{U}\mathbf{z}$ | - $\mathbf{z}_i$ sparse<br>- $\\|\mathbf{u}_l\\|_2 = 1$ |
| Parameters | $\log p(w' \| w) = \langle \mathbf{x}_w, \mathbf{z}_{w'} \rangle + b_{w'} + \text{const}$ | - $f(n_{ij})$weighting function $= \min\{1, (\frac{n_{ij}}{n_{\max}})^\alpha\}$<br>- $\tilde{p}_\theta(w_i \| w_j) = \exp[\langle \mathbf{x}_i, \mathbf{y}_j \rangle]$ | | |
| Convexity | | non joint convex | NP hard combinatorial problem | |
| Algorithm | Negative sampling $\mathcal{L}(\theta) = \sum_{(i,j)\in\Delta^+} \log \sigma(\langle \mathbf{x}_i, \mathbf{z}_j \rangle) + \sum_{(i,j)\in\Delta^-} \log \sigma(-\langle \mathbf{x}_i, \mathbf{z}_j \rangle)$ | SGD | (Greedy) matching pursuit | Iterative greedy minimization |
| Parameter update | | $\mathbf{x}_i^{new} \leftarrow \mathbf{x}_i + 2\eta f(n_{ij})(\log n_{ij} - \langle \mathbf{x}_i, \mathbf{y}_j \rangle)\mathbf{y}_j$;<br>$\mathbf{y}_j^{new} \leftarrow \mathbf{y}_j + 2\eta f(n_{ij})(\log n_{ij} - \langle \mathbf{x}_i, \mathbf{y}_j \rangle)\mathbf{x}_i$ | $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \langle \mathbf{u}_{j^\star}, \mathbf{r_i} \rangle \mathbf{u}_{j^\star}$<br>$\mathbf{r}_{i+1} \leftarrow \mathbf{r}_i - \langle \mathbf{u}_{j^\star}, \mathbf{r}_i \rangle \mathbf{u}_{j^\star}$ | Coding step: $\mathbf{Z}^{t+1} \in \arg\min_{\mathbf{Z}} \\|\mathbf{X} - \mathbf{U}^t \mathbf{Z}\\|_F^2$ subject to $\mathbf{Z}$ being sparse ($\mathbf{z}_n^{t+1} \in \arg\min_{\mathbf{z}} \\|\mathbf{z}\\|_0$ s.t.$\\|\mathbf{x}_n - \mathbf{U}^t \mathbf{z}\\|_2 \leq \sigma \\|\mathbf{x}_n\\|_2$)<br>Dictionary update step: $\mathbf{U}^{t+1} \in \arg\min_{\mathbf{U}} \\|\mathbf{X} - \mathbf{U}\mathbf{Z}^{t+1}\\|_F^2$.<br>$\forall l \in 1, 2 \cdots, L$:<br>set $\mathbf{U} = [\mathbf{u}_1^t \cdots \mathbf{u}_l \cdots \mathbf{u}_L^t]$<br>- $\min_{u_l} \\|\mathbf{X} - \mathbf{U}\mathbf{Z}^{t+1}\\|_F^2 = \\|(\mathbf{X} - \sum_{e \neq l} \mathbf{u}_e^t (\mathbf{z}_e^{t+1})^\top) - \mathbf{u}_l \mathbf{z}_l^{t+1}\\|_F^2 = \\|\mathbf{R}_l^t - \mathbf{u}_l (\mathbf{z}_l^{t+1})^\top\\|_F^2$<br>- Doing 1-SVD on $\mathbf{R}_l' = \sum_i \sigma_i \tilde{\mathbf{u}}_i \tilde{\mathbf{v}}_i^\top$<br>update $\mathbf{u}_l^{t+1} = \tilde{\mathbf{u}}_1, z_l^{t+1} = \sigma_1 \tilde{\mathbf{v}}_i$ |
| Interpretation | | quadratic loss with log-counts as targets | | |
| Practical consideration | Sampling distribution: ratio to the appearance frequency. Sampling number: given | | | |
| Others | | Convex relaxation: $\min \\|\mathbf{z}\\|_1$ | | |

Tabelle 5: Models Comparison