Feature Processing & Learning

Bor-shen Lin at NTUST

Dimension Reduction: Why

- Complexity depends on the number of input dimension
  - Images/speeches/documents are high-dimensional
- Simpler models are more robust on small data set
- Fewer features are easy for explanation and knowledge discovery
- Few dimensions are easy to plot and visualize.

OUTLINES

- Vector Quantization
- Principal Component Analysis
- Latent Semantic Indexing
- Probabilistic Latent Semantic Indexing
- Matrix Factorization
- Linear Discriminant Analysis
- Restricted Boltzman Machines / Auto Encoder
- Transformer / GAN
- t-SNE

Vector Quantization

- Similar to K-Means Clustering
  - Minimization of square error
- Build a *codebook* that contains *codewords*
  - Select representatives for the vector space
  - Might induce quantization errors
- Vectors with continuous variables can be converted into discrete symbols
  - CHMM ➔ DHMM
- Applied to data compression or statistical modeling

OUTLINES

- Vector Quantization
- Principal Component Analysis
- Latent Semantic Indexing
- Probabilistic Latent Semantic Indexing
- Matrix Factorization
- Linear Discriminant Analysis
- Restricted Boltzman Machines / Auto Encoder
- t-SNE

Principal Components Analysis

National Taiwan University of Science and Technology

- Also known as *Karhunen-Loeve transformation (KLT)*
- Projection of $\boldsymbol{x}$ on the direction $\boldsymbol{w}$ is

  z=$\boldsymbol{w}^t\boldsymbol{x}$, $\mu_z = \boldsymbol{w}^t\boldsymbol{\mu}_x$

  $var(z) = E\left((\boldsymbol{z}-\boldsymbol{\mu}_z)^2\right) = E\left(\left(\boldsymbol{w}^t\boldsymbol{x}-\boldsymbol{\mu}_x\right)^2\right) = \boldsymbol{w}^t\Sigma\boldsymbol{w}.$
- To maximize $var(z)$ for $|\boldsymbol{w}| = 1$

  Lagrangian function $\mathrm{J}(\boldsymbol{w}) = \boldsymbol{w}^t\Sigma\boldsymbol{w} + \alpha(\boldsymbol{w}^t\boldsymbol{w}-1)$

  $\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \frac{\partial J}{\partial \boldsymbol{w}} = 2\Sigma\boldsymbol{w} - 2\alpha\boldsymbol{w} = 0,\ \Sigma\boldsymbol{w} = \alpha\boldsymbol{w}$
- $\boldsymbol{w}^*$ is the *eigenvector* for Σ with corresponding *eigenvalue* α
- var(z)=$\boldsymbol{w}^t\Sigma\boldsymbol{w}$=$\boldsymbol{w}^t\alpha\boldsymbol{w} = \alpha|\boldsymbol{w}|^2 = \alpha$

  → Eigenvalue α equals to *the variance of z*.
- Principal components: with largest eigenvalues (variances)

  → reserve discriminant capability for reduced dimensions

Realization of PCA

- Given the samples $X = \{\boldsymbol{x}_i\}$

  $$\boldsymbol{\mu} \equiv E[\boldsymbol{x}] \cong \frac{1}{n}\sum_i \boldsymbol{x}_i$$

  $$\boldsymbol{\Sigma} \equiv E\left[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^t\right] \cong \frac{1}{n}\sum_i (\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^t$$

  $$\mathrm{D} = \begin{bmatrix} {\sigma_1}^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & {\sigma_n}^2 \end{bmatrix}$$

- Covariance matrix $\boldsymbol{\Sigma}$ can be diagonalized through Eigen-decomposition

  $$\Sigma = PDP^t \ \ where\ \ PP^t = P^tP = I$$

  $$D = P^t\Sigma P = P^tE\left[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^t\right]P$$

  $$= E\left[P^t(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^tP\right]$$

- Let $\boldsymbol{x}'=P^t\boldsymbol{x}$ then

  $$\boldsymbol{\mu}' = P^t\boldsymbol{\mu}$$

  $$\Sigma' = E\left[(\boldsymbol{x}'-\boldsymbol{\mu}')(\boldsymbol{x}'-\boldsymbol{\mu}')^t\right] = D$$

  $$\boldsymbol{x} \xrightarrow{P^t} \boldsymbol{x}'$$

  $$\boldsymbol{\mu}, \Sigma \qquad\qquad \boldsymbol{\mu}', D$$

- Transform linearly the vectors to a vector space whose dimensions are mutually uncorrelated.

Realization of PCA

$$D=\begin{bmatrix}\sigma_1{}^2 & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & \sigma_n{}^2\end{bmatrix}$$

- Let $\boldsymbol{x}''=D^{-\frac{1}{2}}P^t(\boldsymbol{x}-\boldsymbol{\mu})$ then

  $\boldsymbol{\mu}''=0$

  $$\Sigma''=E\left[D^{-\frac{1}{2}}P^t(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^tPD^{-\frac{1}{2}}\right]$$

  $$=D^{-\frac{1}{2}}P^tE\left[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^t\right]\boldsymbol{P}\boldsymbol{D}^{-\frac{1}{2}}$$

  $$=D^{-\frac{1}{2}}P^t\Sigma PD^{-\frac{1}{2}}=D^{-\frac{1}{2}}DD^{-\frac{1}{2}}=I$$

$$\begin{array}{ccc}\boldsymbol{x} & \xrightarrow{D^{-\frac{1}{2}}P^t} & \boldsymbol{x}''\\ \boldsymbol{\mu},\ \Sigma & & \boldsymbol{0},\ \boldsymbol{I}\end{array}$$

- Transformed linearly to a vector space of which each dimension has zero mean and unit variance.
- *Pre-whitening* for images/speeches

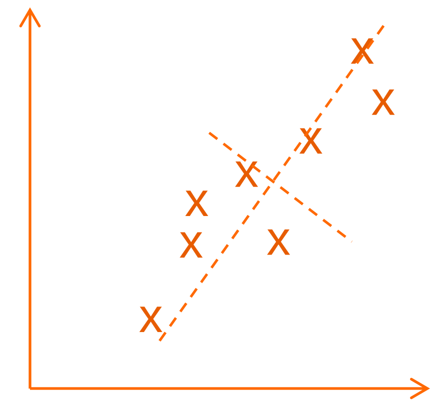
DISCUSSIONS ON PCA

- PCA: perform diagonalization & normalization on the covariance.
  - All dimensions become uncorrelated after being transformed
- $P^t\boldsymbol{x}$: representation of $\boldsymbol{x}$ on eigenspace (orthogonal)
  - P is the rotary matrix that contains the eigenvectors and forms the basis of eignespace
- $D^{-\frac{1}{2}}$: normalization by the standard deviations
  - containing the eigenvectors of Σ
  - D contains the eigenvalues of Σ

  $\lambda_1 > \lambda_2 > \ldots > 0$ (usually sorted in descending order)

DISCUSSIONS ON PCA

- Reduction of dimension
  - Eigenvalues are sorted and k largest eigenvalues with the corresponding eigenvectors are reserved (P_k)
  - Keep those dimensions with the largest variances (most relevant) that preserve a large portion total variance.



- All vectors can be projected onto the K-dimensional eigenspace

  → $\boldsymbol{x}'=P_k^t\boldsymbol{x}$ or $\boldsymbol{x}'=D_k^{-\frac{1}{2}}P_k^t\boldsymbol{x}$

- In eigenspace, distance or cosine similarity between any two vectors can be computed.

Discussions on PCA

- When $\boldsymbol{x}' = D^{-\frac{1}{2}}P^t\boldsymbol{x}$, all dimensions in eigenspace are uncorrelated.
- $\langle \boldsymbol{x}', \boldsymbol{y}' \rangle = \boldsymbol{x}'^t\boldsymbol{y}' = (D^{-\frac{1}{2}}P^t\boldsymbol{x})^t D^{-\frac{1}{2}}P^t\boldsymbol{y}$

  $= \boldsymbol{x}^t P D^{-\frac{1}{2}} D^{-\frac{1}{2}} P^t \boldsymbol{y} = \boldsymbol{x}^t P D^{-1} P^t \boldsymbol{y}$

  $= \boldsymbol{x}^t \Sigma^{-1} \boldsymbol{y} \neq \boldsymbol{x}^t \boldsymbol{y}$

  $\mathrm{d}(\boldsymbol{x}', \boldsymbol{y}') \equiv \langle x' - y', x' - y' \rangle = (\boldsymbol{x} - \boldsymbol{y})^t \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{y})$

  $\mathrm{d}(\boldsymbol{x}, \boldsymbol{y}) \equiv \langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle = (\boldsymbol{x} - \boldsymbol{y})^t (\boldsymbol{x} - \boldsymbol{y})$

  $\mathrm{d}(\boldsymbol{x}', \boldsymbol{y}') \neq \mathrm{d}(\boldsymbol{x}, \boldsymbol{y})$
- Definition of inner product in original space should consider convariance $\boldsymbol{\Sigma}$ with correlated dimensions.
  - Gaussian: $\mathrm{p}(\boldsymbol{x}) = \frac{1}{(2\pi|\Sigma|)^{\frac{D}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}$

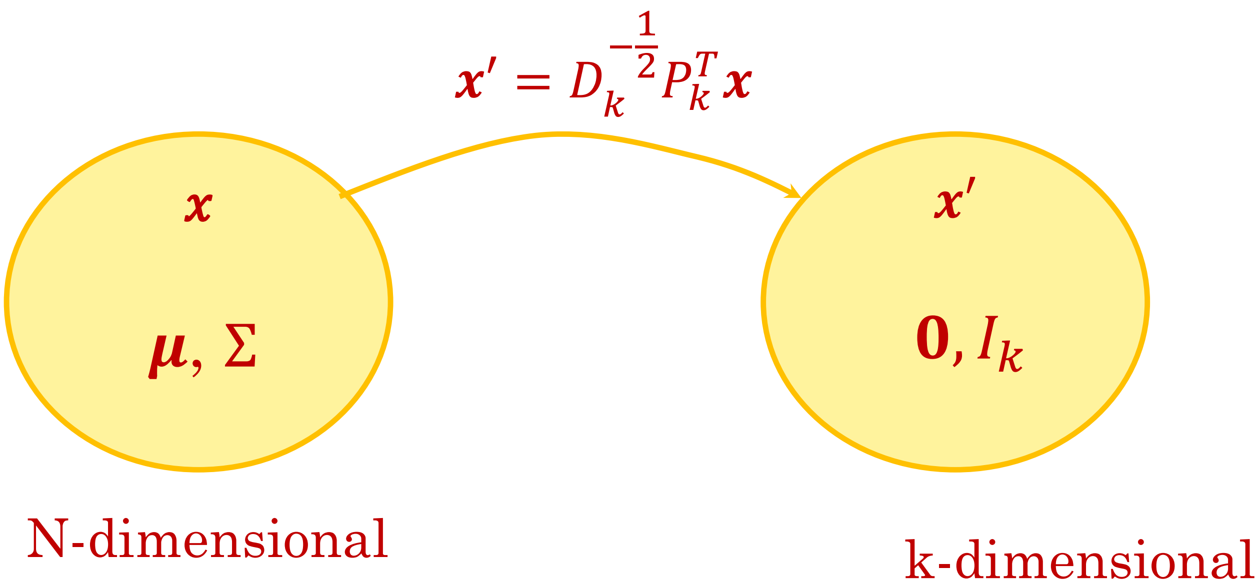
Principal Component Analysis (PCA)

- $\boldsymbol{z} = D^{-\frac{1}{2}} P^t (\boldsymbol{x} - \boldsymbol{\mu}_x)$, then
  - $\boldsymbol{\mu}_z = 0,\ \Sigma_z = I$
  - All dimensions of variable z are independent statistically with zero mean and unit variance
- $\boldsymbol{x}' = D_k^{-\frac{1}{2}} P_k^t \boldsymbol{x}$ is a k-dimensional sub-space of the eigenspace (k eigenvectors with k eigenvalues)
  - The k dimensions with the highest variances (eigen-values) are reserved while the others are discarded.
  - A high-dimensional vector can be reduced to a vector with dimension k. (e.g. image feature)
  - All dimensions become uncorrelated statistically after the conversion.

Principal Component Analysis

- $x' = D_k^{-\frac{1}{2}} P_k^t x$: transformed to a k-dimensional orthonormal space (covariance $\Sigma_k'^{-1} = I_k$)
  - $\langle x', y' \rangle = x'^t y' = x^t \mathrm{P}_k D_k^{-1} P_k^t y = x^t \Sigma_k^{-1} y.$
- Similarity or distance measures in transformed space
  - Norm-2: $|x'|^2 = \langle x', x' \rangle = x^t \Sigma_k^{-1} x$
  - Distance: $d(x', y') = |x' - y'|^2 = (x - y)^t \Sigma_k^{-1} (x - y)$
  - $cos(x', y') = \frac{\langle x', y' \rangle}{|x'||y'|} = \frac{x^t \Sigma_k^{-1} y}{\left((x^t \Sigma_k^{-1} x)(y^t \Sigma_k^{-1} y)\right)^{\frac{1}{2}}}$
- Note $\langle x, y \rangle \equiv x^t y$, then $\langle x', y' \rangle \neq \langle x, y \rangle$ even if k=n
  - $\langle x, y \rangle \equiv x^t y = \sum x_i y_i$ appropriate for orthonormal space
- If $\langle x, y \rangle$ is redefined as $x^t \Sigma^{-1} y$ (instead of $x^t y$),
  - $\langle x, y \rangle = x^t \Sigma^{-1} y \neq x^t y$ for non-orthonormal space (normalized by Σ^{-1})
  - $\langle x', y' \rangle = x'^t \Sigma_k'^{-1} y' = x'^t I_k y' = x'^t y'$ for orthonormal space

PCA for Dimension Reduction

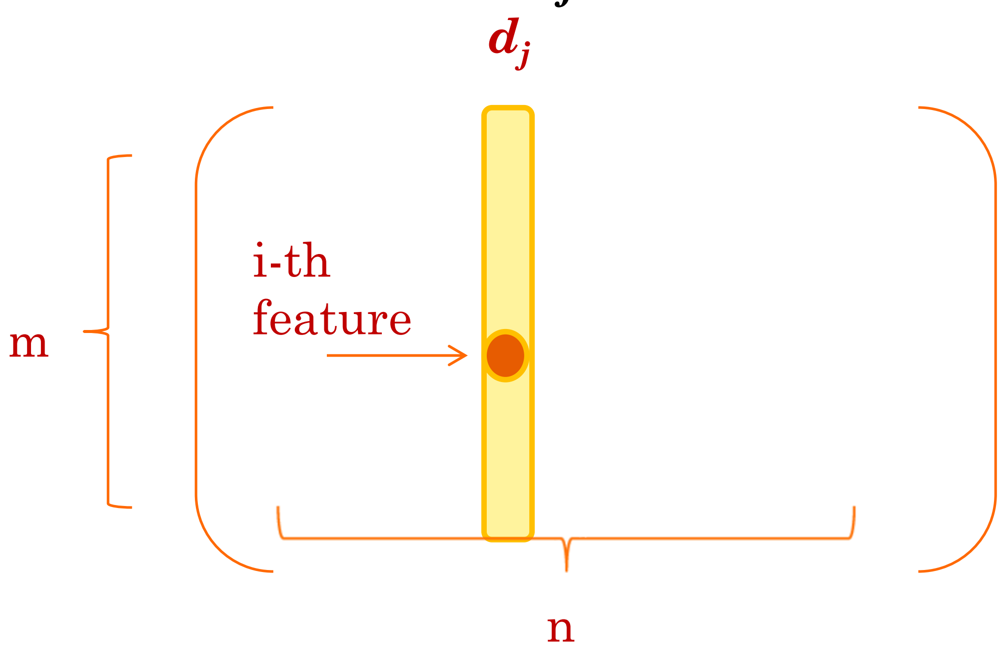


OUTLINES

- Vector Quantization
- Principal Component Analysis
- Latent Semantic Indexing
- Probabilistic Latent Semantic Indexing
- Matrix Factorization
- Linear Discriminant Analysis
- Restricted Boltzman Machines / Auto Encoder
- t-SNE

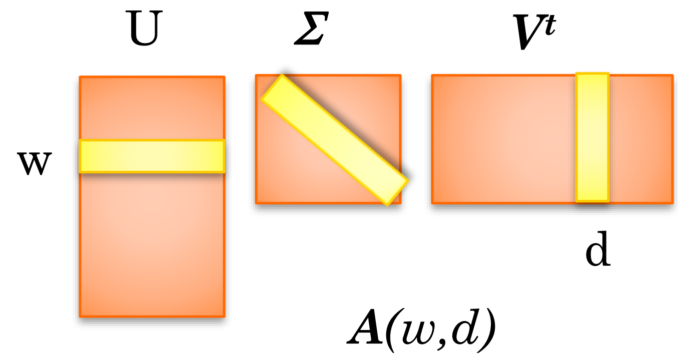
Latent Semantic Indexing (LSI)

- Assume there are n vectors (e.g. documents) each containing m features (terms).
- The n vectors, $\{ \boldsymbol{d}_j \}$, can be represented as an $m \times n$ matrix, **A**, which is called the *term-document matrix*.
- j-th column vector $\boldsymbol{d}_j$ for matrix **A** ~ j-th document.



$$\boldsymbol{A}_{m \times n} = \boldsymbol{U}_{m \times r}\, \boldsymbol{\Sigma}_{r \times r}\, \boldsymbol{V}^{t}_{r \times n}$$

Singular Value Decomposition (SVD)



- $\boldsymbol{A}_{m\times n} = \boldsymbol{U}_{m\times r}\,\boldsymbol{\Sigma}_{r\times r}\,\boldsymbol{V^t}_{r\times n}$
- $\boldsymbol{U}$: right singular vectors
- $\boldsymbol{V}$: left singular vectors
- $\boldsymbol{\Sigma} = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$ where $\sigma_i \geqq 0$

  σ_i : singular values (square root of eigen-values)
- Select the largest k among r singular values (corresponding to the highest variances)
- $\boldsymbol{A'} = \boldsymbol{U_k}\,\boldsymbol{\Sigma_k}\,\boldsymbol{V_k}^t$ (reduce to k-dimensional vector)
- Document vectors: in $\boldsymbol{V_k}^t$
- Term vectors are in $\boldsymbol{U_k}$
- Query vector: $\boldsymbol{q_k}^t = \boldsymbol{q}^t\,\boldsymbol{U^t}_k\boldsymbol{\Sigma_k}^{-1}$

Discussions on LSI

- Convert document/term into eigenspace in which all dimensions are statistically independent.
  - In LSI, dimensions with largest variances (σ_i)will be kept while the rest dimensions are eliminated so as to reduce the computational complexity.
  - In eigenspace, terms and documents are represented in the same form. Therefore, the similarity between documents or that between terms can be computed.
  - In original document space, the cosine similarity between, say, *tour* and *travel* is 0. In eigenspace, their similarity might not be 0.
  - Search , clustering, classification, and GMM training can be performed in eigenspace.

Applicability of LSI

- It is possible to defined similarity between terms (or abstract concepts)
  - similarity(tour ~ travel) ? similarity(tour, visa)?
  - Computed not by *word senses*, but by the *usage contexts* (the documents) of the words
- Generalization
  - Matrix A is NOT limited to *term-document* relationships. It can be generalized for *arbitrary M-to-M mappings*

    i.e. any tabular data in attribute-value form
  - SVD may be performed, and so do search, clustering, classification, statistical models (e.g. GMM) etc.
  - *topic/author* → similarity between authors/topics

    user/items → clustering products/users, recommendation

    album/style → distance between albums/styles

Outlines

- Vector Quantization
- Principal Component Analysis
- Latent Semantic Indexing
- Probabilistic Latent Semantic Indexing
- Matrix Factorization
- Linear Discriminant Analysis
- Restricted Boltzman Machines / Auto Encoder
- t-SNE

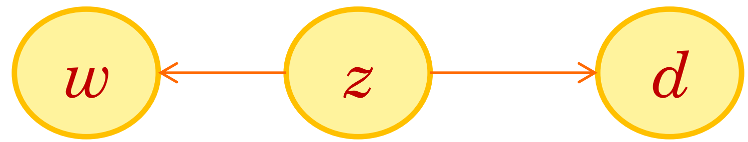
Probabilistic Latent Semantic Indexing (PLSI)

- d document, w word, z latent variable

  $d \in D = \{d_1, d_2, \dots, d_N\}$

  $w \in W = \{w_1, w_2, \dots, w_M\}$

  $z \in Z = \{z_1, z_2, \dots, z_K\}$



- To maximize $L = \sum_{d,w} p(d,w) \log(p(d,w))$

  based on Expectation Maximization

  E-step: $p(z|d,w) = \frac{p(z,d,w)}{p(d,w)} = \frac{p(z)p(w|z)p(d|z)}{\sum_{z'} p(z')p(w|z')p(d|z')}$

  $$p(d,w) = \sum_{z'} p(z')p(w|z')p(d|z') \rightarrow L$$

  M-step: $p'(w|z) = \frac{p(z,w)}{p(z)} = \frac{\sum_d n(d,w)p(z|d,w)}{\sum_{d,w'} n(d,w')p(z|d,w')}$

  $$p'(d|z) = \frac{p(z,d)}{p(z)} = \frac{\sum_w n(d,w)p(z|d,w)}{\sum_{d',w} n(d',w)p(z|d',w)}$$

  $$p'(z) = \frac{\sum_{d,w} n(d,w)p(z|d,w)}{\sum_{d,w} n(d,w)}$$

Training Procedure

- Parameters for estimation

  $p(z) \sim \Sigma_{k,k} = p(z_k)$ K x K diagonal matrix

  $p(w|z) \sim U_{k,j} = p(w_j|z_k)$ M x K matrix

  $p(d|z) \sim V_{j,k} = p(d_j|z_k)$ N x K matrix

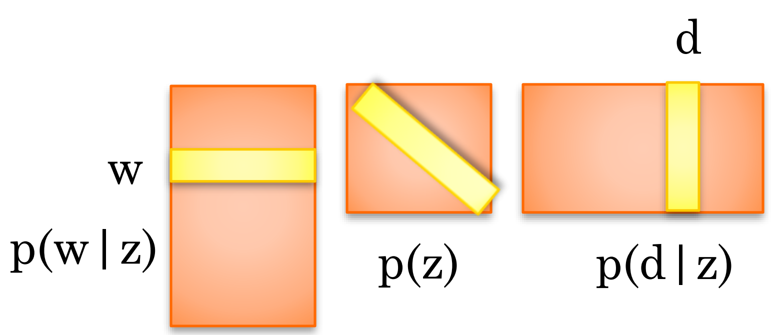
- Training
  - Random initial values
  - Reestimation of parameters based on EM algorithm
- Estimation of $p(d|z)$ for new document d
  - Compute $p(d, w, z)$ for d with $p(z)$, $p(w|z)$ fixed
  - $p(z, d) = \Sigma_w p(z, w, d)$, $p(d|z)=p(z,d)/p(z)$ → iteration
- After converging, compute $p(z|d)=p(z,d)/\Sigma_{z'}p(z',d)$
  - $p(z_k|d)$'s form a K-dimensional vector (summed to 1)

Discussions on PLSI

- Comparison with LSI

  $p(d,w) = \Sigma_z\, p(z)p(w|z)p(d|z)$

  $U\Sigma V^t$ is a M x N matrix ~ term-doc matrix
- Interpretation
  - U (for $p(w|z)$) and V(for $p(d|z)$) correspond to the eigenvectors by SVD in LSI
  - Σ (for $p(z)$) corresponds to the singular values in LSI



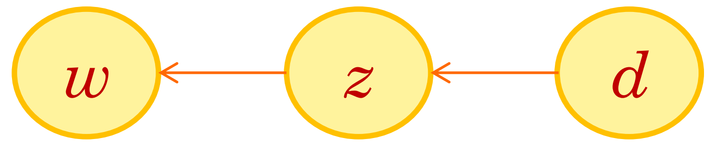
PLSI – Different Formulation

- d document, w word, z latent variable

  $d \in D = \{d_1, d_2, \dots, d_N\}$

  $w \in W = \{w_1, w_2, \dots, w_M\}$

  $z \in Z = \{z_1, z_2, \dots, z_K\}$



- To maximize $L = \sum_{d,w,z} p(d,w,z) \log(p(d,w,z))$ based on Expectation Maximization

  E-step: $p(z|d,w) = \frac{p(z,w|d)}{p(w|d)} = \frac{p(z|d)p(w|z)}{\sum_{z'} p(z'|d)p(w|z')}$

  $$p(d,w) = p(d) \sum_{z'} p(z'|d)p(w|z')$$

  M-step: $p'(w|z) = \frac{p(z,w)}{p(z)} = \frac{\sum_d n(d,w)p(z|d,w)}{\sum_{d,w'} n(d,w')p(z|d,w')}$

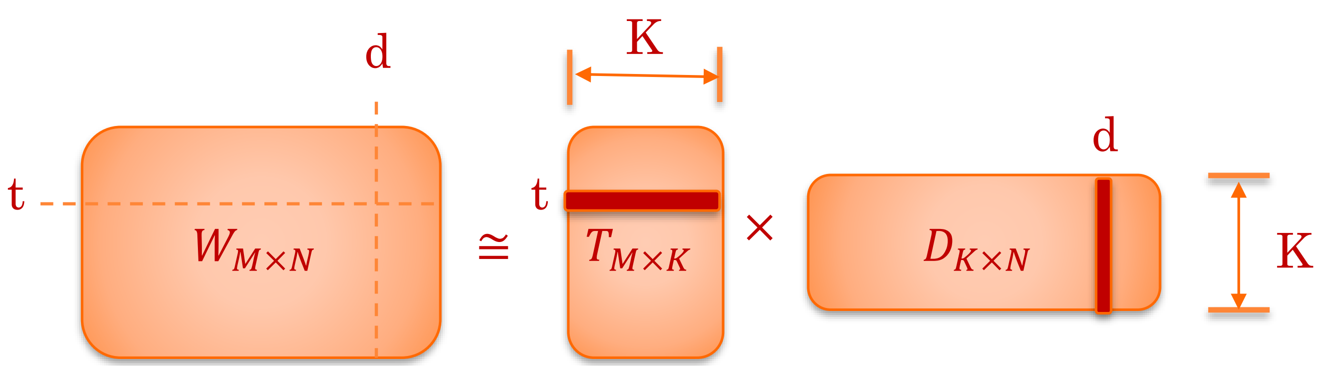
  $$p'(z|d) = \frac{p(z,d)}{p(d)} = \frac{\sum_w n(d,w)p(z|d,w)}{\sum_w n(d,w)}$$

OUTLINES

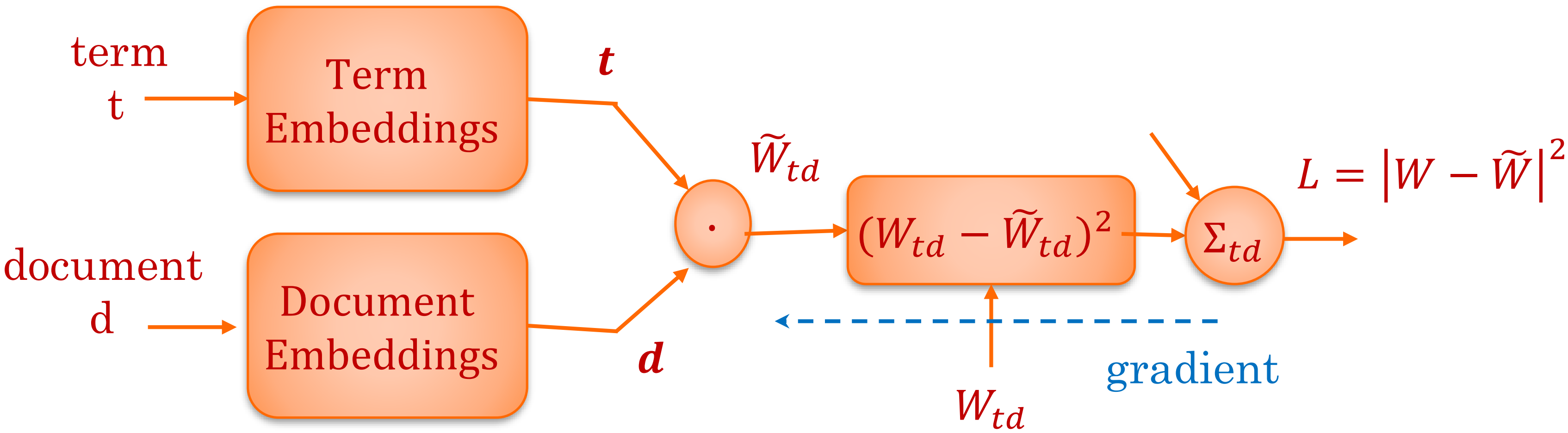
- Vector Quantization
- Principal Component Analysis
- Latent Semantic Indexing
- Probabilistic Latent Semantic Indexing
- Matrix Factorization
- Linear Discriminant Analysis
- Restricted Boltzman Machines / Auto Encoder
- t-SNE

Matrix Factorization (MF)

- A is a term-document (or user-item) matrix may be represented with a generative model of latent space.
  - Factorized as the multiplication of two matrix
  - Term matrix T and document matrix D are learnable
  - $\widetilde{W}_{td} = \sum_k T_{tk} D_{kd}$ is the inner product of T_t and D_d
  - All vectors are of dimension k.



MF by Gradient Descent



- Embeddings: a set of learnable k-dimensional vectors
- $\boldsymbol{t}$ and $\boldsymbol{d}$ are K-dimensional and $\boldsymbol{t} \cdot \boldsymbol{d} = \sum_k \boldsymbol{t}(k)\boldsymbol{d}(k)$
- $\frac{\partial L}{\partial \widetilde{W}_{td}} = 2\left(W_{td} - \widetilde{W}_{td}\right)(-1) = -2\left(W_{td} - \widetilde{W}_{td}\right)$
- $\nabla_{\boldsymbol{d}} L = \frac{\partial L}{\partial \widetilde{W}_{dt}} \frac{\partial \widetilde{W}_{dt}}{\partial \boldsymbol{d}} = 2\left(W_{td} - \widetilde{W}_{td}\right)\boldsymbol{t} \rightarrow \Delta \boldsymbol{d} = -\varepsilon \nabla_{\boldsymbol{d}} L$
- $\nabla_{\boldsymbol{t}} L = \frac{\partial L}{\partial \widetilde{W}_{td}} \frac{\partial \widetilde{W}_{dt}}{\partial \boldsymbol{t}} = 2\left(W_{td} - \widetilde{W}_{td}\right)\boldsymbol{d} \rightarrow \Delta \boldsymbol{t} = -\varepsilon \nabla_{\boldsymbol{t}} L$

Discussions

- Notice MF does not guarantee the latent vectors to be positive or summed to one (like PLSI).
  - Every k-D vector belongs to R^K
- Similar computation graph could also be used to learn latent vectors with the constraint of PLSI based on gradient descent (without using singular value decomposition)
  - May use softmax to make an array of probabilities (non-negative) summed to 1.
  - e.g. add softmax to make $|\boldsymbol{d}| = 1$, $|\boldsymbol{t}| = 1$, $\sum_t \widetilde{W}_{td} = 1$

OUTLINES

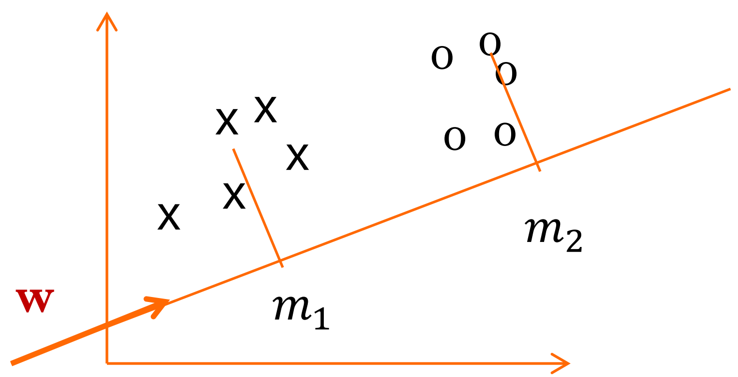
- Vector Quantization
- Principal Component Analysis
- Latent Semantic Indexing
- Probabilistic Latent Semantic Indexing
- Matrix Factorization
- Linear Discriminant Analysis
- Restricted Boltzman Machines / Auto Encoder
- t-SNE

Linear Discriminant Analysis (LDA)

National Taiwan University of Science and Technology

- Reduction of dimension for training high-dimensional vectors such that the low-dimensional feature is discriminant for classification
- Binary classification (C_1, C_2) & K = 2
- Conversion of feature $z = \mathbf{w}^t\mathbf{x}$

  find optimal direction w
- $\boldsymbol{m}_i = \frac{1}{|C_i|}\sum_{x_t \in C_i} \boldsymbol{x}_t$
- $m_i = \boldsymbol{w}^t \boldsymbol{m}_i$
- ${s_i}^2 = \sum_{x_t \in C_i} (\boldsymbol{w}^t \boldsymbol{x}_t - m_i)^2$
- $J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{{s_1}^2 + {s_2}^2}$



Linear Discriminant Analysis (LDA)

- $(m_1 - m_2)^2 = (\boldsymbol{w}^t\boldsymbol{m}_1 - \boldsymbol{w}^t\boldsymbol{m}_2)^2$
  $= \boldsymbol{w}^t(\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^t\boldsymbol{w}$
  $= \boldsymbol{w}^t\boldsymbol{S}_B\boldsymbol{w}$
- Between-class scatter matrix
  $\boldsymbol{S}_B = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^t$
- ${s_i}^2 = \sum_{x_t \in C_i}(\boldsymbol{w}^t\boldsymbol{x}_t - m_i)^2 = \sum_{x_t \in C_i}(\boldsymbol{w}^t(\boldsymbol{x}_t - \boldsymbol{m}_i))^2$
  $= \sum_{x_t \in C_i}\boldsymbol{w}^t(\boldsymbol{x}_t - \boldsymbol{m}_i)(\boldsymbol{x}_t - \boldsymbol{m}_i)^2\boldsymbol{w} = \boldsymbol{w}^t\boldsymbol{S}_i\boldsymbol{w}$
  $\boldsymbol{S}_i \equiv \sum_{x_t \in C_i}(\boldsymbol{x}_t - \boldsymbol{m}_i)(\boldsymbol{x}_t - \boldsymbol{m}_i)^2$
  ${s_1}^2 + {s_2}^2 = \boldsymbol{w}^t(\boldsymbol{S}_1 + \boldsymbol{S}_2)\boldsymbol{w} = \boldsymbol{w}^t\boldsymbol{S}_w\boldsymbol{w}$
- Within-class scatter matrix
  $\boldsymbol{S}_w = \boldsymbol{S}_1 + \boldsymbol{S}_2$

Linear Discriminant Analysis (LDA)

- Maximization of $J(\boldsymbol{w}) = \frac{\boldsymbol{w}^t \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^t \boldsymbol{S}_W \boldsymbol{w}}$

  $\boldsymbol{w}^* = c\boldsymbol{S}_W{}^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$ (for K = 1)

  is called Fisher's linear discriminant
- A special case

  When $p(\mathbf{x} \mid C_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$, $\boldsymbol{w}^* = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$.

LDA Generalized for K > 2

- $\boldsymbol{z} = \boldsymbol{w}^t \boldsymbol{x}$ (reduced dimension > 1)
- $S_w = \sum_{i=1}^{K} S_i$
- $S_B = \sum_{i=1}^{K} |C_i|(\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^t$

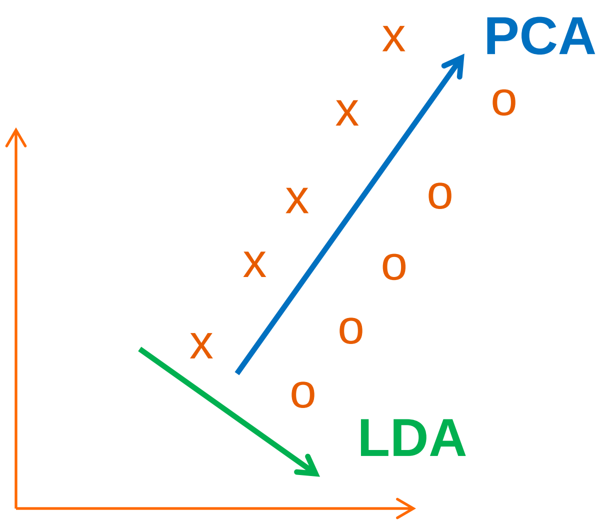
  $\boldsymbol{m} = \sum_{i=1}^{K} \boldsymbol{m}_i$
- Maximization of $J(\boldsymbol{w}) = \frac{|\boldsymbol{w}^t S_B \boldsymbol{w}|}{|\boldsymbol{w}^t S_w \boldsymbol{w}|}$

  $\boldsymbol{w}^*$: eigenvectors *of* ${S_w}^{-1} S_B$ with largest eigenvalues.

LDA VS. PCA

- LDA: for classification (with class labels)
  - To find the dimensions for discriminating the classes
- PCA: dimensions with maximum variance

  (class labels are not required or provided)



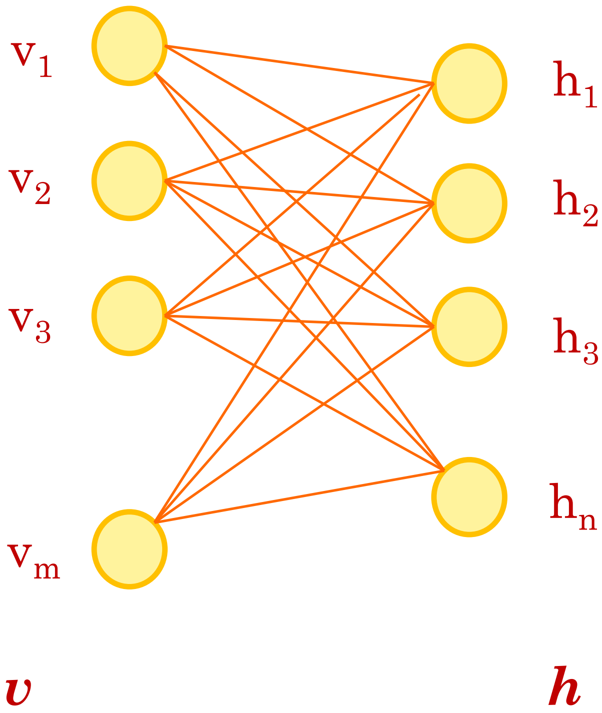
Outlines

- Vector Quantization
- Principal Component Analysis
- Latent Semantic Indexing
- Probabilistic Latent Semantic Indexing
- Matrix Factorization
- Linear Discriminant Analysis
- Restricted Boltzman Machines / Auto Encoder
- t-SNE

Feature Learning Using ANN

- Conventionally, features for classification need to be determined in advance based on domain knowledge
  - Gaussian pyramid for image recognition
  - MFCC for speech recognition
- In classification task, the input features could be pre-trained automatically using ANN (pre-training)
  - Restricted Boltzman Machine (RBM)
  - Auto Encoder (AE)
- The input could also be the raw data (images/speeches), and the features can be trained and optimized jointly with the classifier in a deep neural network.
  - Convolutional neural network (CNN) for image classification
  - DNN-HMM for speech recognition

Restricted Boltzman Machine



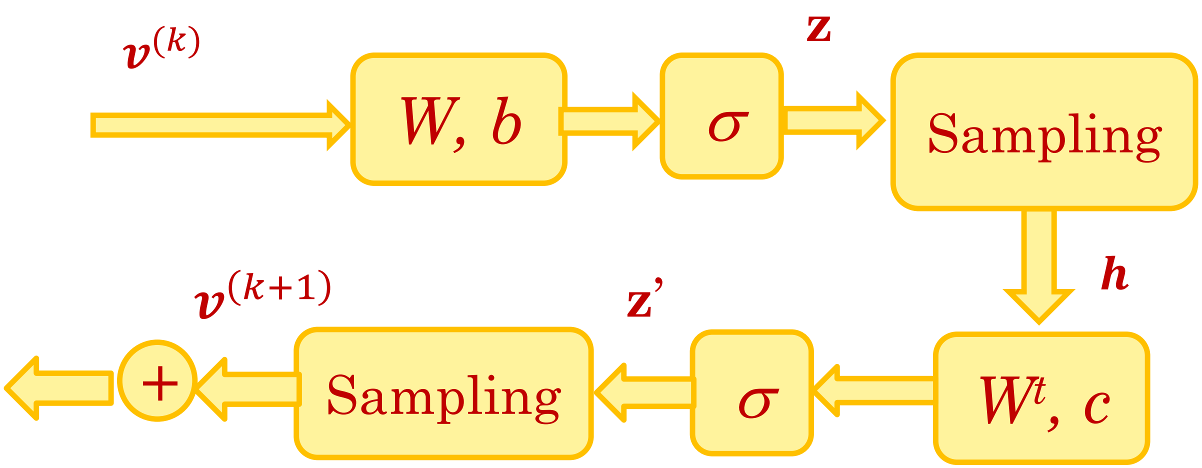
- Model: $p(\boldsymbol{v}, \boldsymbol{h}) \propto e^{-E(\boldsymbol{v},\boldsymbol{h})}$
- $E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{h}^t W \boldsymbol{v} - \boldsymbol{b}^t \boldsymbol{v} - \boldsymbol{c}^t \boldsymbol{h}$

  $= -\sum_{i,j} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_j$
- $p(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) = \frac{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}}{Z}$
- $Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$
  - normalizing the probability weights such that $\sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) = 1$
- Objective: $\sum_{\boldsymbol{v}} log\ p(\boldsymbol{v})$
  - $p(\boldsymbol{v})$ fits for all the $\boldsymbol{v}$'s

Unsupervised Learning of RBM

- Stochastic process
  - $z_i = P(H_i = 1|\boldsymbol{v}) = \sigma\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right)$
  - $z_j{}' = P\left(V_j = 1\middle|\boldsymbol{h}\right) = \sigma(\sum_{i=1}^{n} w_{ij} h_i + b_j)$
- k-step contrastive divergence
  - $dw_{ij} = p\left(H_i = 1\middle|\boldsymbol{v}^{(0)}\right) v_j{}^{(0)} - p\left(H_i = 1\middle|\boldsymbol{v}^{(k)}\right) v_j{}^{(k)}$
  - $db_j = v_j{}^{(0)} - v_j{}^{(k)}$
  - $dc_i = h_i{}^{(0)} - v_i{}^{(k)}$
- High-level features $\boldsymbol{h}$'s can be learned from original features $\boldsymbol{v}$'s.
- Multiple features can be learned and stacked layer by layer.

Computational Graph of RBM



- **v** and **h** are binary vectors (0/1)
- **z** and **z**' : $z_i, z_i' \in [0,1]$
- W, b, and c are learnable parameters
- Uncertainty is included through random sampling

K-STEP CONTRASTIVE DIVERGENCE

Algorithm 1. *k*-step contrastive divergence

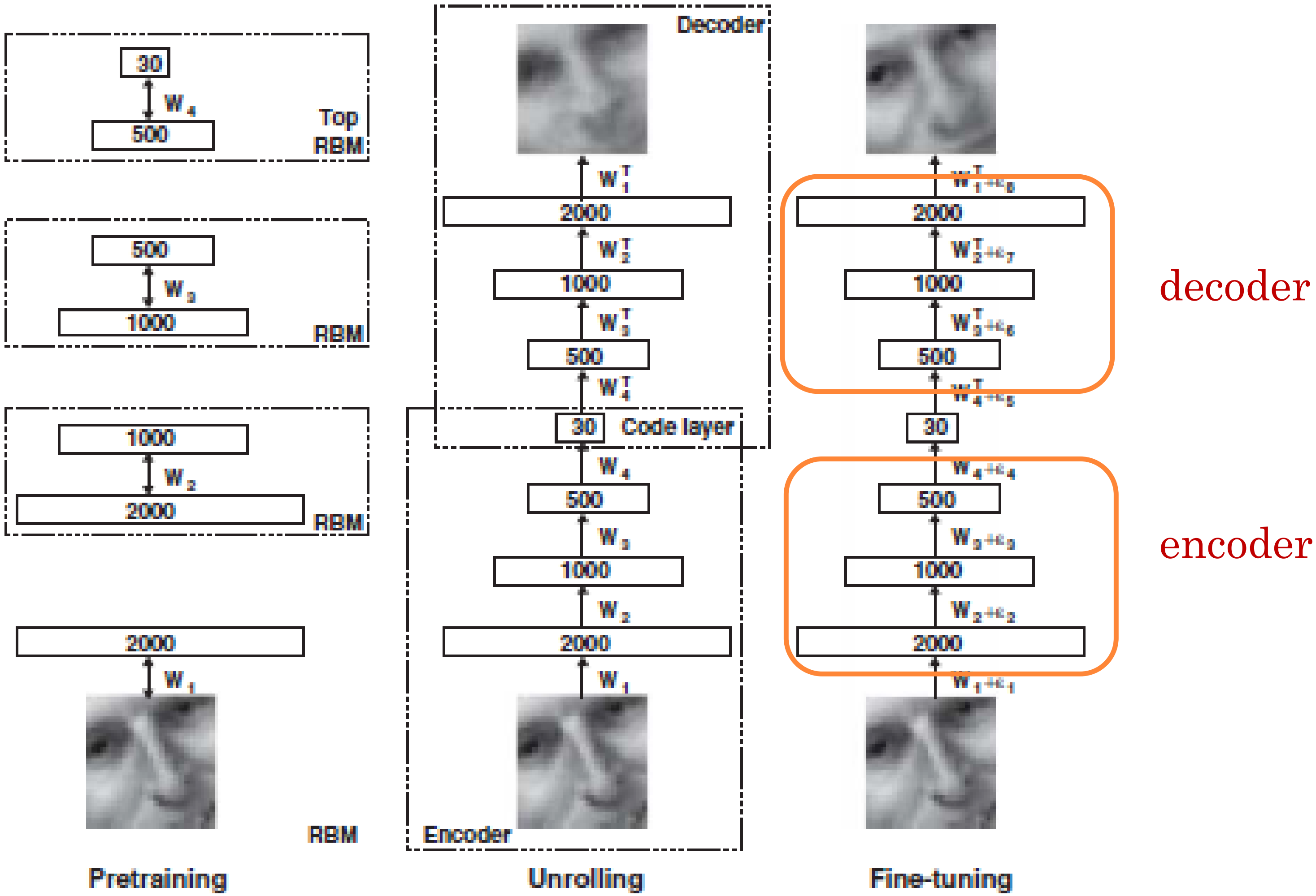
Input: RBM $(V_1, \dots, V_m, H_1, \dots, H_n)$, training batch S

Output: gradient approximation Δw_{ij}, Δb_j and Δc_i for $i = 1, \dots, n$, $j = 1, \dots, m$

```
init Δw_ij = Δb_j = Δc_i = 0 for i = 1, ..., n, j = 1, ..., m
forall the v ∈ S do
    v^(0) ← v
    for t = 0, ..., k − 1 do
        for i = 1, ..., n do  sample h_i^(t) ~ p(h_i | v^(t))
        for j = 1, ..., m do  sample v_j^(t+1) ~ p(v_j | h^(t))
    for i = 1, ..., n, j = 1, ..., m do
        Δw_ij ← Δw_ij + p(H_i = 1 | v^(0))·v_j^(0) − p(H_i = 1 | v^(k))·v_j^(k)
        Δb_j ← Δb_j + v_j^(0) − v_j^(k)
        Δc_i ← Δc_i + p(H_i = 1 | v^(0)) − p(H_i = 1 | v^(k))
```

- Cited from A. Fischer etc.

STACKED RBMS (AUTO-ENCODER)

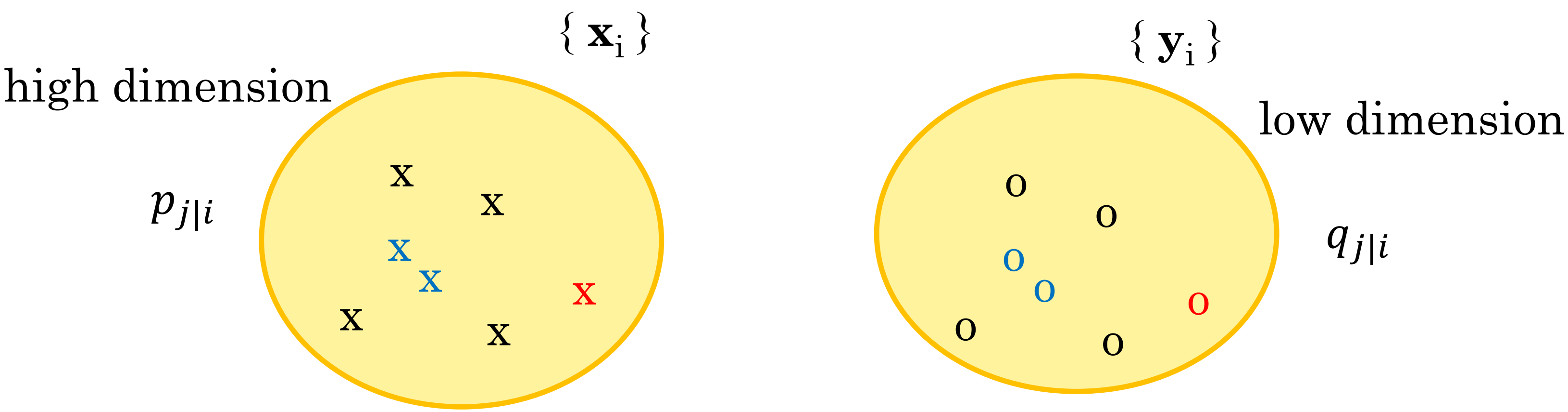


- Unsupervised learning (cited from Hinton etc.)

Outlines

- Vector Quantization
- Principal Component Analysis
- Latent Semantic Indexing
- Probabilistic Latent Semantic Indexing
- Matrix Factorization
- Linear Discriminant Analysis
- Restricted Boltzman Machines / Auto Encoder
- t-SNE

T-SNE



- $p_{j|i} = \dfrac{\exp\left(-\frac{|x_i - x_j|^2}{2{\sigma_i}^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{|x_i - x_k|^2}{2{\sigma_i}^2}\right)}$, $q_{j|i} = \dfrac{\exp\left(-|\boldsymbol{y}_i - \boldsymbol{y}_j|^2\right)}{\sum_{k \neq i} \exp(-|\boldsymbol{y}_i - \boldsymbol{y}_k|^2)}$
- Thinking: observe points *j*’s from point *i*.
- KL-Divergence as cost $C = \sum_{i \neq j} p_{j|i} \, log\left(\frac{p_{j|i}}{q_{j|i}}\right)$
- $\dfrac{\partial C}{\partial \boldsymbol{y}_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(\boldsymbol{y}_i - \boldsymbol{y}_j)$
- Asymmetric: $p_{i|j} \neq p_{j|i}$, $q_{i|j} \neq q_{j|i}$

Symmetric SNE

- $p_{ij} = \frac{\exp\left(-\frac{|\boldsymbol{x}_i - \boldsymbol{x}_j|^2}{2\sigma^2}\right)}{\sum_{k\neq l}\exp\left(-\frac{|\boldsymbol{x}_k - \boldsymbol{x}_l|^2}{2\sigma^2}\right)}$, $q_{ij} = \frac{\exp\left(-|\boldsymbol{y}_i - \boldsymbol{y}_j|^2\right)}{\sum_{k\neq l}\exp(-|\boldsymbol{y}_l - \boldsymbol{y}_k|^2)}$
- $\mathrm{C} = \sum_{i\neq j} p_{ij}\, log\left(\frac{p_{ij}}{q_{ij}}\right)$
- $\frac{\partial C}{\partial \boldsymbol{y}_i} \equiv \nabla_{\boldsymbol{y}_i} C = 4\sum_j (p_{ij} - q_{ij})(\boldsymbol{y}_i - \boldsymbol{y}_j)$
- Using student t-distribution (Cauchy distribution) to alleviate crowding problem, $q_{ij} = \frac{\left(1+|\boldsymbol{y}_i - \boldsymbol{y}_j|^2\right)^{-1}}{\sum_{k\neq l}(1+|\boldsymbol{y}_k - \boldsymbol{y}_l|^2)^{-1}}$
- Thinking: try to make
  - q_{ij} approaches p_{ij} after dimension reduction
  - Invariable for the relative distances among points
  - Gaussian: minus log probability ~ distance

Expectation *Minimization*

- Build { $\mathbf{y}_i$ } randomly
- Iteration
  1. Compute the pairwise probabilities p_{ij} and q_{ij}
  2. E-step: compute the objective KLD. If the KLD converges, then terminate.
  3. M-step: update $\mathbf{y}_i$ by gradient descent, that is, $d\boldsymbol{y}_i = -\epsilon \nabla_{\boldsymbol{y}_i} C$, or, $\boldsymbol{y}_i{}' = \boldsymbol{y}_i - \epsilon \nabla_{\boldsymbol{y}_i} C$
  4. Repeat 1-3 till convergence.
- After the training, for a high-dimensional new point $\mathbf{x}$, the same procedure can be run with the $\mathbf{y}_i$'s fixed to obtain the corresponding.
  - Transformation is NOT in a closed form (e.g. PCA)

Feature Learning

	Name	Concept	Algorithm
PCA	Principal component analysis	Reserve those dimensions with highest variances	Eigen Decomposition
LSI/LSA	Latent semantic indexing	Reserve those dimensions with highest variances	Singular Value Decomposition
PLSI/PLSA	Probabilistic LSI	Topic model	EM/Gradient descent
MF	Matrix Factorization	Latent vectors	Gradient descent
t-SNE	t-Distributed Stochastic Neighbor Embedding	Din low-D space similar to that in high-D space	Gradient descent
RBM / Auto-encoder	Restricted Boltzman Machine	Unsupervised learning (fine tuning)	Gradient descent

Some References

- Asja Fischer and Christian Igel, *An Introduction to Restricted Boltzman Machines*, LNCS 7441, pp.14-36, 2012. Springer-Verlag.
- G.E. Hinton and R. R. Salak Hutdinov, *Reducing the Dimensionality of Data with Neural Network*, Vol. 313, Science, 2006.
- Laurens van der Maaten, Geoffrey Hinton and Yoshua Bengio, *Visualizing Data using t-SNE*, Journal of Machine Learning Research 9 (2008) 2579-2605, 2008.