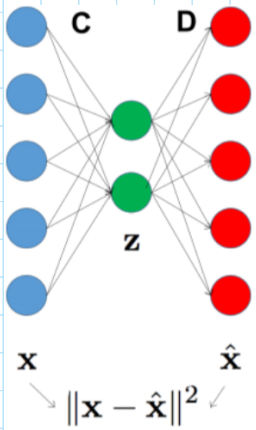
Computational Intelligence Lab Zusammenfassung



- Linear Autoencoder

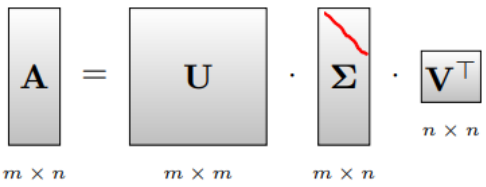
- C ≙ Encoder
- D ≙ Decoder
- Loss: $L(x, \theta) = \frac{1}{2}\|x - \hat{x}(\theta)\|^2$
- $\hat{x}(\theta) = D \cdot C \cdot x$
- Parameters: D and C
- z is bottleneck layer / latent representation
- z is bottleneck => reduces rank to k => loss of information
- $\text{rank}(C \cdot D) \leq \min\{\text{rank}(C), \text{rank}(D)\} \leq k$

- Eckart-Young Theorem

- $\arg\min \|X - \hat{X}\|_F^2 = U \cdot \Sigma_k \cdot V^T$
- You get the best rank k approximation of a matrix when you take the SVD and the first k dominant singular values

- Singular Value Decomposition (Recommender System)

- U/V: - Left/right singular vectors
  - orthogonal
  - $U \cdot U^T = I$
  - $U^T = U^{-1}$
- Σ: - Singular values
  - $\text{diag}(\sigma_1, \ldots, \sigma_s)$ $(s \leq \min(m, n))$



- Optimal Autoencoder

- $C^* = U_k^T$
- $D^* = U_k$

$$\hat{x} = D \cdot C \cdot x = D \cdot C \cdot (U \cdot \Sigma \cdot V^T)$$
$$= U_k U_k^T \cdot U \cdot \Sigma \cdot V^T$$
$$\approx U \cdot I_k \cdot \Sigma \cdot V^T$$
$$= U \cdot \Sigma_k \cdot V^T$$

- Principle Eigenvector Decomposition (Image compression)
  - Principle eigenvector has the highest trend direction
  - A must be squared matrix
  - u is eigenvector of A iff $A \cdot u = \lambda \cdot u$
  - $\hookrightarrow A \cdot u = \lambda \cdot u$
  - λ is called eigenvalue, stored in Λ
  - We preserve largest data variance
  - $\Sigma = U \cdot \Lambda \cdot U^T$
  - Σ is symmetric: $\Sigma = \Sigma^T$
  - $\hookrightarrow$ Variance covariance matrix: $\Sigma = \frac{1}{n} \cdot X \cdot X^T = \frac{1}{n} \sum_i x_i \cdot x_i^T$
    - diag: variance between same feature
    - off-diag: variance between two features
  - Σ is p.s.d $\Rightarrow$ all eigenvalues are positive
  1. Remove mean $\hat{X} = X - M$ $(m \times n)$
  2. Get $\Sigma = \frac{1}{n} \cdot \hat{X} \cdot \hat{X}^T$ $(m \times m)$
  3. Pick d eigenvectors U_d from Σ: $\Sigma = U \cdot \Lambda \cdot U^T$
  4. Reduced matrix $Z = \underbrace{U_d^T}_{d \times m} \cdot \underbrace{\hat{X}}_{m \times n}$
  4. $\tilde{X} = U_d \cdot Z$

- Power Method
  - $v_{t+1} = \frac{A \cdot v_t}{\|A \cdot v_t\|}$
  - $\lim_{t \to \infty} v_t = u_1$

- Iterative view
  - Find eigenvector of $(\Sigma - \lambda \cdot u u^T)$ (= 2nd EV of Σ)
  - Iterate to find lower EV

- Detect knee in eigenspectrum to choose d
- PCA vs. Autoencoder: - PCA is unique and interpretable
- From SVD to PCA: $A \cdot A^T = (U \cdot \Sigma \cdot V^T) \cdot (U \cdot \Sigma \cdot V^T)^T$

$$= U \cdot \Sigma \cdot V^T \cdot V \cdot \Sigma^T \cdot U^T$$

$$= U \cdot \underbrace{\Sigma \Sigma^T}_{\Lambda} \cdot U^T = U \cdot \Sigma^2 \cdot U^T$$

$\hookrightarrow \Sigma^2 = \Lambda$

- Lagrange Multiplier
  - can be used for PCA
  - $\mathcal{L}(u_1, \lambda_1) = u_1^T \Sigma \cdot u_1 - \lambda \underbrace{(u_1^T u_1 - 1)}_{u \text{ needs to be unit length}}$

  - $\nabla_{u_1} \mathcal{L}(u_1, \lambda_1) = 2(\Sigma \cdot u_1 - \lambda_1 \cdot u_1) = 0$

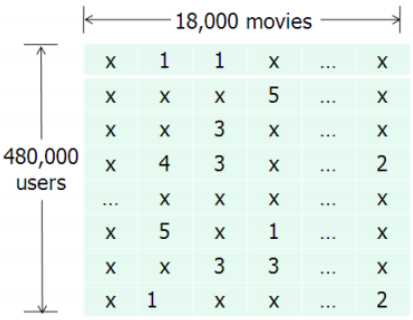
    $\Sigma \cdot u_1 - \lambda_1 \cdot u_1 = 0$

    $\Sigma \cdot u_1 = \lambda_1 \cdot u_1$ (=Definition of Eigenvector)

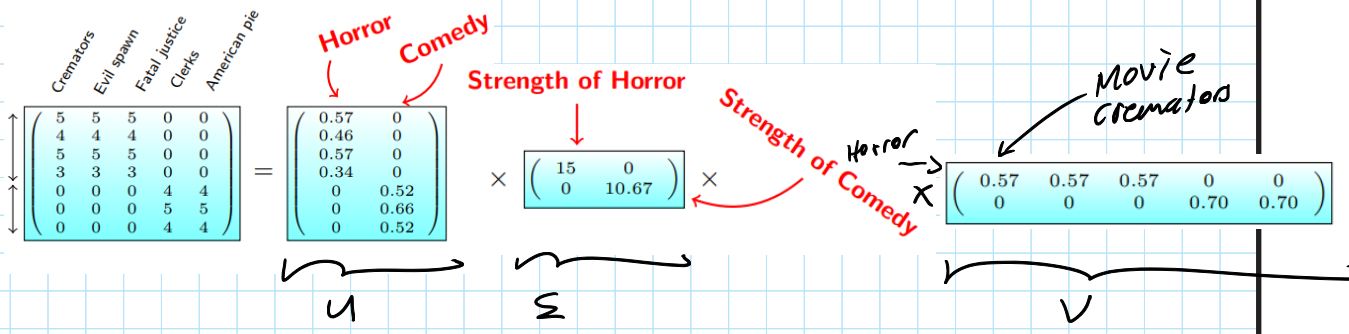
  - If we want to get second eigenvector we need to add that u_1 and u_2 must be orthogonal: $u_2^T \cdot u_1 = 0$

- Recommender System

- Matrix completion
- Find best low-rank approximation
- Frobenius norm: $\|A\|_F = \sqrt{\sum\sum a_{ij}^2}$
  - Sum of all entries



- k-dimensional: number of latent factors
- SVD of rating matrix:
  - U: User-to-factor association
  - Σ: Dominancy of each factor
  - V: Items-to factor association



- For 0 elements we take the mean
- SVD not convex optimization problem
  - ↳ $f(u,v) = \frac{1}{I} \sum (a_{ij} - \langle u_i, v_j \rangle)^2$ => Not convex
- $u \leftarrow \text{argmin}_u f(u,v)$ => convex
- $v \leftarrow \text{argmin}_v f(u,v)$ => convex

} fix other variable and alternate

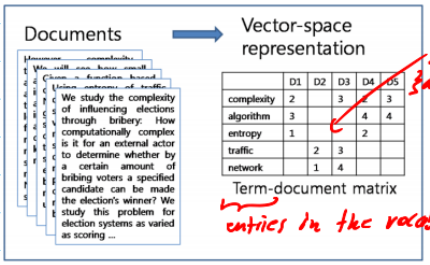
- $A \cdot A^T$: Similarity between users
- $A^T \cdot A$: Similarity between movies

} Based on ratings

- Frobenius norm only depends on singular values Σ
  - ↳ $\|F\|_F^2 = \sum \sigma_i^2$
- Euclidian 2-norm only depends on largest sing. value
  - ↳ $\|A\|_2 = \sigma_1$

Approx error: only Eigenvalues that are not included

- Document Representation
  - Vocabulary: · All meaningful words in a language
  - Term filtering: · Remove misspellings
    · Exclude stop words (the, is, at, ...)
  - Term normalization: · Reduce word to stem
    · argue (argued, arguing, ...)
  - Bag of words: · Ignore structure of words
    ↳ Only co-occurence counts



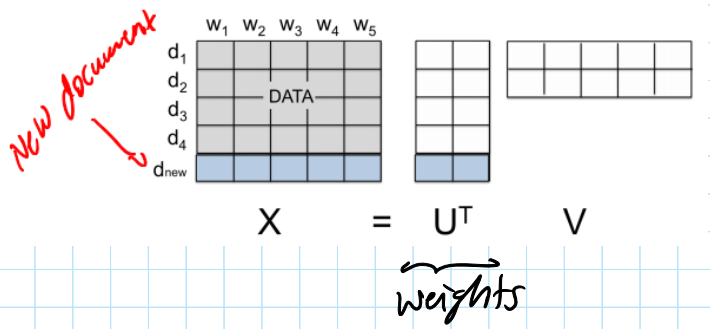
- Probabilistic LSA (=pLSA)
  - Get main topics of text
  - Document has distribution of topics
  - Topic: · distribution over terms
    · Soccer topic: high "teams" word probability

1. For each document get the topic
2. For each topic get the words

- $p(w|d) = \sum_z p(w,z|d) = \sum p(w|d,z) \cdot p(z|d) \overset{\#}{=} \sum_{z=1}^{K} p(w|z) \cdot p(z|d)$
  - $p(w|d)$: word w in doc d
  - $\#$ conditional independence assump.
  - K: # topics predefined
  - $p(w|z)$: ②
  - $p(z|d)$: ①
- $X = x_{i,j}$ (= # of occurences of w_j in document d_i)
- $\ell(U,V) = \sum_{i,j} x_{i,j} \cdot \log p(w_j|d_i) = \sum_{[i,j] \in X} \log \sum_{z=1}^{K} p(w_j|z) \cdot p(z|d_i)$
  - $p(w_j|z) = v_{z,j}$
  - $p(z|d_i) = u_{z,i}$
  - Probabilities, therefore non-negative matrices
- $Q_{z,i,j} \in [0,1]$, probability that word w_j in doc d_i is generated from topic z

- Expectation step (solve for optimal q)
  - ↳ $q_{z,i,j} = \frac{u_{zi} \cdot v_{zj}}{\sum_{k=1}^{K} u_{ki} \cdot v_{kj}}$ } word comes from topic z / } word comes from other topics
- Maximization step (solve for optimal parameters)
  - ↳ v: reestimate probability of word w for topic z
  - u: reestimate probability of doc i covering topic z

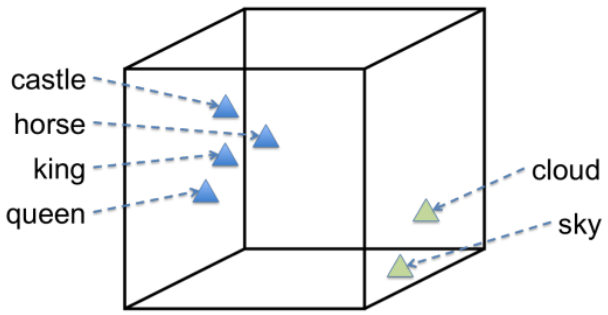
- Latent Dirichlet Allocation
  - Generate a new document



- Need probability of u: doc drew covering topic z
- Use dirchlet distribution for generating topic weights u
- u_i contains probability of each topic being in document i
- LDA tries to use an appropriate prior for the language rather than just the one fits all simple 2nd order correlation that pLSA uses.
- Dirchlet prior acts as regularizer
- Need to have good hyperparameter α

- Word Embeddings
  - Predict context word given recent word (=skip-gram)
  - $p_\theta(w|w')$ = probability that w occurs in context window w'
  - Objective: $\mathcal{L}(\theta; w) = \sum_{t=1}^{T} \sum_{p \in \mathcal{I}} \log p_\theta(w^{t+\Delta} | w^{(t)})$
  - Each word is represented in vector space (=Embedding)
  - $p_\theta(w|w') = \langle x_w, x_{w'} \rangle + b_w$

$$w \mapsto (\mathbf{x}_w, b_w) \in \mathbb{R}^{a+1}, \quad \text{(vector + bias)}$$



  - Context vector:
    - Distinguish output vocabulary V and input vocabulary C
    - Output embedding x_w
    - Input embedding y_w
    - $p_\theta(w|w') = \langle x_w, y_{w'} \rangle + b_w$
  - Negative sampling
    - Reduce to binary classification
    - Negative samples don't belong together
    - $\mathcal{L}(\theta) = \sum_{(i,j) \in \Delta^+} \log \sigma(\langle x_i, y_j \rangle) + \sum_{(i,j) \in \Delta^-} \log \sigma(-\langle x_i, y_j \rangle)$
  - GloVe (=Try to match co-occurence count of two words)
    - Summarize data in co-occurance matrix
    - n_{ij} = #occurences of $\underbrace{w_j}_{\text{output}}$ in context of $\underbrace{w_j}_{\text{input}}$
    - $n_{\text{castle,king}}$ = #occurences of castle in context of word king
    - N can be computed in one pass
    - Sparse matrix

- Weighted least squares
  - $H(\theta, N) = \sum f(n_{ij}) \cdot (\underbrace{\log n_{ij}}_{\text{Target}} - \underbrace{\log p_\theta(w_i | w_j)}_{\text{Model to train}})^2$
  - $f(n)$: downweights word-pairs with small counts
    - Cut of at certain threshold $\Rightarrow$ limit influence

- Data Clustering

- Assign each data point a cluster
- Clusters represented by centroids u
- Mapping via nearest centroid rule
  - ↳ $\pi(x) = \arg\min_{j=1\dots k} \|u_j - x\|$
- Optimization problem
  - Find clusters u and assignments π
  - Indicator variable $z_{ij} = \begin{cases} 1 & \text{if } \pi(x) = j \\ 0 & \text{otherwise} \end{cases}$
  - $\sum_{j=1}^{K} z_{ij} = 1$
  - $J(u, z) = \sum_{i=1}^{N} \sum_{j=1}^{K} z_{ij} \cdot \|x_i - u_j\|^2$
    - ↳ Alternating minimization
- Alternating optimization (=K-means)
  - Compute z, given u (=cluster assignment/E-step)
    - ↳ $z_{ij}^* = \begin{cases} 1 & \text{if } j = \arg\min_k \|x_i - u_k\|^2 \\ 0 & \text{otherwise} \end{cases}$
  - Compute u, given z (=centroid update/M-step)
    - ↳ $u_k^+ = \frac{\sum_{n=1}^{N} z_{kn}^+ \cdot x_n}{\sum_{n=1}^{N} z_{kn}^{(+)}}$ } Mean of all x values; } Normalization e.g. $1/n$

initialize $\mathbf{U}$ on K distinct random data points as initial centroids
initialize $\mathbf{Z} \leftarrow \mathbf{Z}^*(\mathbf{U})$
repeat
  $\mathbf{U} \leftarrow \mathbf{U}^*(\mathbf{Z})$ (see above) ← Assignments Update
  $\mathbf{Z}^{\text{new}} \leftarrow \mathbf{Z}^*(\mathbf{U})$ (see above) ← centroid Update
  same $= (\mathbf{Z}^{\text{new}} == \mathbf{Z})$
  $\mathbf{Z} \leftarrow \mathbf{Z}^{\text{new}}$
until (same) ← until nothing changes

↳ $O(n \cdot k \cdot d)$

} converges to local minimum, but global minimum is NP-hard

- K-means ++
  - No random initialization
  - Pay more attention to datapoints that are far away

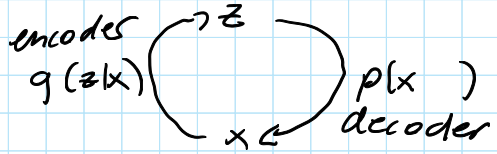
- Probabilistic clustering
  - Instead of hard assignments, use probabilities
  - $p(x, \theta) = \sum_{j=1}^{K} \hat{\pi}_j \cdot p(x, \mu_j, \Sigma_j)$
    - $\sum_{j=1}^{K} \hat{\pi}_j = 1$ (Mixing coefficient)
      - Like prior
  - $p(x, \theta_j)$ = location & shape of cluster
  - $\hat{\theta} = \arg\max_\theta \sum_{i=1}^{N} \log \left[\sum_{j=1}^{K} \hat{\pi}_j \cdot p(x_i, \theta_j) \right]$
    - Datapoints, Clusters
    - Expectation: Points to clusters
    - Maximization: cluster location and shape

- Generate data point
  1. Sample cluster index
  2. Sample data point from sampled cluster

Comparison with K-means

- Assignments
  - K-means algorithm: hard assignment points to clusters
  - EM algorithm: soft assignment based on posteriors
- Shapes
  - K-means: spherical cluster shapes, uniform spread
  - EM algorithm: can learn covariance matrix
- K-means as a special case
  - Gaussian mixture model with (fixed) covariances $\mathbf{\Sigma}_j = \sigma^2 \mathbf{I}$
  - in the limit of $\sigma \to 0$, recover K-means (hard assignments)
  - can be more formally derived (EM objective $\to$ K-means objective)

- Variational Autoencoder



- Generative model: Get x from z
- Inference model: Get z from x

1. $\log p(x)$
2. $p(x) = \int p(x|z) \cdot p(z) \cdot dz$
3. $p(x) = \int p(x|z) \cdot \frac{p(z)}{q(z|x)} \cdot q(z|x)$ ← we get this from encoder
4. $\log p(x) \geq ELBO(\phi, \theta) = E_{q_\phi}[\log p_\theta(x|z) - KL(q_\theta(z|x) \| p(z))]$

- Maximize θ - generative model
- Maximize ϕ : inference model

$\log p_\theta(x|z)$ ↑ Reconstruction quality

$KL(q_\theta(z|x) \| p(z))$ ↑ Regularizer

- Inference model suggests z for generative model

- Generative Adversarial Network

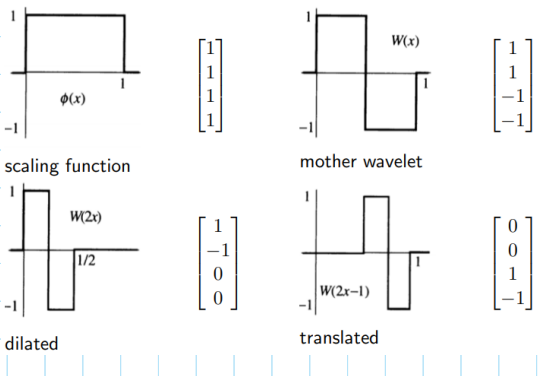
$\theta^* := \text{argmin}\{\sup L(\theta, \phi)\}$

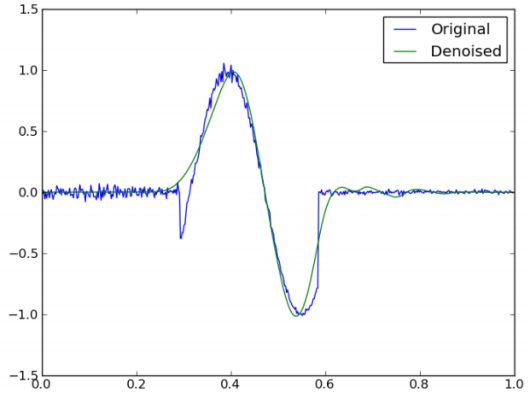
↑ Generator ↑ Classifier

- Autoregressive models

- Justified by chain rule: $p(x) = \prod_{t=1}^{n} p(x_t | x_{1:t-1})$

- Haar Wavelets
  - Original signal x and orthogonal transform matrix U
  - $z = U^T \cdot x = \langle U, x \rangle$ / $x = U \cdot z = x = U \cdot U^T \cdot x$
  - Energy & distance & angles is preserved: $\|U^T x\|^2 = \|x\|^2$
  - Signal compression: - Truncate small values of z
    - $\hat{x} = U \hat{z}_{truncated}$
    - Can denoise signals
  - No information is lost
  - Haar wavelets good for localized signals





  - Fourier basis: - Good for sine like signals
    - Global support
    - Not good for localized signals
  - Fixed transform basis U
    - sender/receiver agrees on basis beforehand
    - Transmit only non-zero elements of $\hat{z}$
  - PCA transform basis U
    - called Hoteling transform
    - Use K largest eigenvectors
    - Optimal for given Σ
  - Basis: 1. Linearly independent vectors
    2. spans the space

- Overcomplete Dictionaries
  - More general dictionaries
  - $U \in \mathbb{R}^{D \times L}$, where $L > D$ (more atoms than dim.)
  - More atoms (dictionary elements) than dimensions
  - U is overcomplete, more unknowns than equations
  - ↳ find sparsest $z \in \mathbb{R}^L$ such that $x = Uz$
  - ↳ $z^* \in \arg\min \|z\|_0$ s.t. $x = Uz$
  - Column ≙ Atom
  - All atoms ≙ dictionary
- Matching Pursuit
  - Chooses one atom at a time to greedily reduce approximation error
  - Want to minimize residual $R_{n+1} = r_n - \hat{x}_n$
  - Initialize $r_0 = x$ and $\hat{x}_0 = 0$
  1. Find $j^* = \arg\max_j |\langle r_i, u_j \rangle|$
  2. Compute better approximation $\hat{x}_{i+1} \leftarrow \hat{x}_i + \langle r_i, u_j \rangle \cdot u_j$
  3. Update residual $r_{i+1} \leftarrow r_i - \underbrace{\langle r_i, u_j \rangle \cdot u_j}_{\text{Save these in } z}$
- Compressive Sensing
  - Compressing while collecting the data
  - $y = W \cdot x = W \cdot U \cdot z$ (W: Gaussian random matrix)
  - $z^* \in \arg\min_z \|z\|_0$ such that $y = \Theta z$ where $\Theta = W \cdot U$

- Dictionary Learning (= Matrix Factorization)
  - Mixture of different characteristics in U
  - we adapt dictionary to signal characteristics
  - $(U, Z) = \underset{U,Z}{\operatorname{argmin}} \|X - U \cdot Z\|_F^2$
    ↳ Not jointly convex
  - Shared dictionary among many samples
  - Coding step: $z_n \in \arg\min \|z\|_0$ for $n = 1, \dots, N$
    s.t. $\|x_n - U z\|_2 \leq \sigma \cdot \|x_n\|_2$
    ↑ approx error
  - Dictionary step: 1. Fix all atoms except u_i
    2. zero out atom u_i
    3. Look at all points that used u_i
    4. Update atom as first singular vector from residual