Computational Intelligence Lab

Doruk Çetin

September 13, 2019

Contents

1 Linear Autoencoder

Motivations for **dimensionality reduction**:

- Visualization — e.g. 2D or 3D
- Data compression — fewer coefficients
- Generative models — latent variables
- Signal recovery — discard irrelevant information (noise)
- Discover modes of variation — intrinsic properties of data
- Feature discovery — learn better representations

Linear dimension reduction is in the form of

$$\mathbf{z} = \mathbf{C}\mathbf{x} \iff z_r = \sum_{s=1}^{m} c_{rs} x_s,\ \mathbf{C} = (c_{rs})_{1\le r\le k, 1\le s\le m}\ ,$$

where $\mathbf{C}$ is a linear map and each feature z_r is a linear combination of input variables (i.e. linear unit in neural network terminology). Linear autoencoder performs **low-rank approximation** with parameters $\theta = (\mathbf{C}, \mathbf{D})$ as coder/decoder ($\mathbf{z}$ acts as a bottleneck layer).

$$\text{Sample reconstruction error: } J(\theta) = \frac{1}{2n}\sum_{i-1}^{n} ||\mathbf{x}_i - \hat{\mathbf{x}}_i(\theta)||^2 = \frac{1}{2n}||\mathbf{X} - \hat{\mathbf{X}}(\theta)||_F^2$$

Frobenius norm doesn't pay attention to how the coefficients are organized in the matrix, because every entry is treated independently.

Rank of a linear map $\mathbf{A} : \mathbb{R}^k \to \mathbb{R}^l$ is $rank(\mathbf{A}) \le min\{k, l\}$. For a matrix product (composition of linear maps) $rank(\mathbf{AB}) \le min\{rank(\mathbf{A}), rank(\mathbf{B})\}$

Eckart-Young theorem: Optimal rank-k approximation can be obtained via Singular Value Decomposition (SVD):

$$\underset{\hat{X}:rank(\hat{X}=k)}{\arg\min}\ ||\mathbf{X} - \hat{\mathbf{X}}||_F^2 = \mathbf{U}\mathbf{\Sigma}_k\mathbf{V}^T$$

$$\text{Minimal reconstruction loss: } \min_{\theta} J(\theta) = \sum_{l=k+1}^{\min\{n,m\}} \sigma_l^2$$

Define $\mathbf{U}_k$ as the first k columns of $\mathbf{U}$. $\mathbf{C}_k^* = \mathbf{U}_k^T$ and $\mathbf{D}_k^* = \mathbf{U}_k$ yields minimal reconstruction error for a linear autoencoder with k hidden units. For $\mathbf{D} \in \mathbb{R}^{m\times k}$ and $\mathbf{C} \in \mathbb{R}^{k\times m}$, proof is as follows (result is optimal by Eckart-Young theorem above):

$$\hat{\mathbf{X}} = \mathbf{D}^*\mathbf{C}^*\mathbf{X} = \mathbf{U}_k\mathbf{U}_k^T\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right) = \mathbf{U}_k\begin{bmatrix}\mathbf{I}_k & \mathbf{0}\end{bmatrix}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{U}\begin{bmatrix}\mathbf{I}_k & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{U}\mathbf{\Sigma}_k\mathbf{V}^T$$

Corollary: weight sharing $\mathbf{D} = \mathbf{C}^T$ without reducing model power. Mapping $\mathbf{x} \mapsto \mathbf{z}$ uniquely determined up to rotations (permutations, reflections) and degrees of freedom reduced from $2km$ to km.

2 Principal Component Analysis

- Converting inner products to outer products.
- Sigma is very easy to compute in a streaming manner.
- Correlations between input dimensions mean redundancy that we can compress out.
- We want to retain directions where the datapoints vary a lot. Direction of smallest reconstruction error is direction of largest data variance.

PCA: Final Answer

Optimal reduction to d dimensions: diagonalize Σ and pick the d principal eigenvectors $\widetilde{\mathbf{U}} = (\mathbf{u}_1 \dots \mathbf{u}_d)$, $d \leq m$. Dimension reduction for $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $\mathbf{D} \in \mathbb{R}^{m \times d}$: $\mathbf{Z} = \widetilde{\mathbf{U}}^T \mathbf{X} \in \mathbb{R}^{d \times n}$. For optimal reconstruction in d dimensions use eigenbasis: $\widetilde{\mathbf{X}} = \widetilde{\mathbf{U}}\mathbf{Z} = \widetilde{\mathbf{U}}\widetilde{\mathbf{U}}^T \mathbf{X}$.

PCA transforms a dataset X into a dataset $Z = A^T X$ by defining a new basis using the eigenvectors of the covariance matrix Σ_X of the dataset X. With this particular choice of a new basis, the covariance matrix Σ_Z of the transformed dataset Z is diagonalized. We would like to decouple the dimensions/measurements in the transformed dataset, i.e. we would like to have uncorrelated dimensions. We see that the covariance matrix of Z becomes the diagonal eigenvalue matrix Λ: Choosing the eigenvectors associated with the highest eigenvalues results in capturing high variances in the transformed dataset.

Comparison w/ Linear Autoencoder Network

- PCA clarifies that one should (ideally) center the data
- PCA representation is unique (if no eigenvalue multiplicities) and as such (in principle) interpretable
- Linear autoencoder w/o weight sharing is highly non-interpretable (lack of identifiability)
- Linear autoencoder w/ weight sharing: $\mathbf{A} = \mathbf{B}^T$ only identifies a subspace, but axis are non-identifiable (can an autoencoder be modified to identify the principle axes?)
- General lesson: caution with naively interpreting learned (neural) representations

If we do weight sharing, linear autoencoder will identify the subspace of the top k principal eigenvectors of the matrix (same as PCA). It will project the data to the same subspace but it might choose a different basis. It might not be the nice eigenvector basis, it might not be orthogonal. In short, representations that we get may be as good as PCA representations but in terms of interpretability it is still weaker. There are hacks to enforce ordering in autoencoder to get PCA.

2.1 Power Method

A simple algorithm for finding the dominant eigenvector of A.

$$\mathbf{v}_{t+1} = \frac{\mathbf{A}\mathbf{v}_t}{||\mathbf{A}\mathbf{v}_t||} \implies \lim_{t \to \infty} \mathbf{v}_t = \mathbf{u}_1, \text{ assuming } \langle \mathbf{u}_1, \mathbf{v}_0 \rangle \neq 0 \text{ and } |\lambda_1| > |\lambda_j| \forall j \geq 2$$

$$\text{We can recover } \lambda_1 \text{ from Rayleigh quotient: } \lambda_1 = \lim_{t\to\infty} \frac{||\mathbf{A}\mathbf{v}_t||}{||\mathbf{v}_t||}$$

2.2 PCA Step-by-step

1. Organize the dataset as a matrix:
$$\mathbf{X} \in \mathbb{R}^{D\times N}$$
2. Calculate the empirical mean:
$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$$
3. Center the data by subtracting mean from each sample:
$$\bar{\mathbf{X}} = \mathbf{X} - \mathbf{M}, \text{ where } \mathbf{M} = [\bar{\mathbf{x}}, \dots, \bar{\mathbf{x}}]$$
4. Compute the covariance matrix:
$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T = \frac{1}{N}\bar{\mathbf{X}}\bar{\mathbf{X}}^T$$
5. Eigenvalue decomposition:
$$\boldsymbol{\Sigma} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T, \text{ where } \mathbf{U} = [u_1, \dots, u_D]$$
6. Model selection (to capture maximal variance of the data):
$$\text{Pick a } K \leq D$$
7. Transform the data onto the new basis of K dimensions:
$$\bar{\mathbf{Z}} = \mathbf{U}_K^T\bar{\mathbf{X}}, \text{ where } \mathbf{U}_K = [u_1, \dots, u_K]$$
8. Reconstruction (going back to the original basis):
$$\widetilde{\mathbf{X}} = \widetilde{\bar{\mathbf{X}}} + \mathbf{M}, \text{ where } \widetilde{\bar{\mathbf{X}}} = \mathbf{U}_K\bar{\mathbf{Z}}$$

2.3 PCA Reconstruction Error

$$\begin{aligned}
\text{err} &= \frac{1}{N}||\tilde{\bar{\mathbf{X}}} - \bar{\mathbf{X}}||_F^2 \\
&= \frac{1}{N}||(\mathbf{U}_K\mathbf{U}_K^T - \mathbf{I}_d)\bar{\mathbf{X}}||_F^2 \\
&= \frac{1}{N}\text{trace}((\mathbf{U}_K\mathbf{U}_K^T - \mathbf{I}_d) \cdot \bar{\mathbf{X}}\bar{\mathbf{X}}^T \cdot (\mathbf{U}_K\mathbf{U}_K^T - \mathbf{I}_d)^T) \\
&= \frac{1}{N}\text{trace}((\mathbf{U}_K\mathbf{U}_K^T - \mathbf{I}_d) \cdot \mathbf{\Sigma} \cdot (\mathbf{U}_K\mathbf{U}_K^T - \mathbf{I}_d)^T) \\
&= \frac{1}{N}\text{trace}((\mathbf{U}_K\mathbf{U}_K^T - \mathbf{I}_d) \cdot \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \cdot (\mathbf{U}_K\mathbf{U}_K^T - \mathbf{I}_d)^T) \\
&= \frac{1}{N}\text{trace}((\mathbf{U}_K\mathbf{U}_K^T\mathbf{U} - \mathbf{U}) \cdot \mathbf{\Lambda} \cdot (\mathbf{U}^T\mathbf{U}_K\mathbf{U}_K^T - \mathbf{U}^T)^T) \\
&= \frac{1}{N}\text{trace}((\begin{bmatrix}\mathbf{U}_K & \mathbf{0}\end{bmatrix} - \mathbf{U}) \cdot \mathbf{\Lambda} \cdot (\begin{bmatrix}\mathbf{U}_K & \mathbf{0}\end{bmatrix} - \mathbf{U}^T)^T) \\
&= \text{trace}(\sum_{i=K+1}^{D} \lambda_i u_i u_i^T) \\
&= \sum_{i=K+1}^{D} \lambda_i \cdot \text{trace}(u_i u_i^T) \\
&= \sum_{i=K+1}^{D} \lambda_i
\end{aligned}$$

3 Matrix Approximation and Reconstruction

3.1 Collaborative Filtering

An alternative to content filtering relies only on past user behavior —for example, previous transactions or product ratings— without requiring the creation of explicit profiles. This approach is known as **collaborative filtering**. Idea is to exploit collective data from many users to generalize across users and, possibly, across items.

Content filtering:

+ Got a new item to add? No problem, just be sure to include the side information with it
– Assumes access to side information about items (e.g. properties of a song)

Collaborative filtering:

+ Does not assume access to side information about items (e.g. does not need to know about movie genres)
– Does not work on new items that have no ratings

Two types of collaborative filtering:

- Neighborhood Methods: Find neighbors based on similarity of movie preferences and recommend movies that those neighbors watched

- Latent Factor Methods: Assume that both movies and users live in some low dimensional space describing their properties and recommend a movie based on its proximity to the user in the latent space

The most convenient data is high-quality explicit feedback, which includes explicit input by users regarding their interest in products. Usually, explicit feedback comprises a sparse matrix, since any single user is likely to have rated only a small percentage of possible items.

Earlier systems relied on imputation to fill in missing ratings and make the rating matrix dense. However, imputation can be very expensive as it significantly increases the amount of data. In addition, inaccurate imputation might distort the data considerably. Hence, more recent works suggested modeling directly the observed ratings only, while avoiding overfitting through a regularized model.

$\mathbf{A}_k$ is an optimal approximation in the sense of the spectral norm

$$\min_{rank(\mathbf{B})=k} ||\mathbf{A}-\mathbf{B}||_2 = ||\mathbf{A}-\mathbf{A_k}||_2 = \sigma_{k+1}$$

Define weighted Frobenius norm with regard to matrix $\mathbf{G} \geq \mathbf{0}$:

$$||\mathbf{X}||_{\mathbf{G}} = \sqrt{\sum_{i,j} g_{ij} x_{ij}^2}$$

special case: $g_{ij} \in \{0,1\}$ (Boolean, partly observed matrix)

Two formalizations of the collaborative filtering problem:

- Low-rank matrix factorization (through ALS)

$$\min_{\mathbf{X}:rank(\mathbf{X})\leq k} \|\mathbf{A}-\mathbf{X}\|_{\mathcal{I}}^2$$

- Exact matrix recovery (through convex relaxation)

$$\min_{\mathbf{X}} rank(\mathbf{X}), \text{ such that } \|\mathbf{A}-\mathbf{X}\|_{\mathcal{I}} = 0$$

where $\mathcal{I}$ is the subset of indices containing the observed entries. Note that we cannot simply just do SVD as it needs a complete matrix.

3.2 Alternating Least Squares

Low-rank approximations are (in general) intrinsically hard

$$\mathbf{B}^* \xrightarrow{\min} \ell(\mathbf{B}) = \|\mathbf{A}-\mathbf{B}\|_{\mathbf{G}}^2, \text{ s.t. } rank(\mathbf{G}) \leq k$$

is NP-hard, even for $k = 1$.

Instead of optimizing $\mathbf{B}$ as a rank-k matrix, we can optimize its factorization that ensures the rank constraint (inner dimension is k). Convexity depends on the parametrization of the objective. Reparametrize $\mathbf{B} = \mathbf{UV}$, $\mathbf{U} \in \mathbb{R}^{m\times k}$, $\mathbf{V} \in \mathbb{R}^{k\times n}$ then $rank(\mathbf{B}) \leq k$ by definition.

Non-convex objective is as follows:

$$f(\mathbf{U},\mathbf{V}) = \frac{1}{|\mathcal{I}|}\sum_{(i,j)\in\mathcal{I}}(a_{ij} - \langle \mathbf{u}_i, \mathbf{v}_j\rangle)^2$$

f is convex in $\mathbf{V}$ for fixed $\mathbf{U}$ and convex in $\mathbf{U}$ for fixed $\mathbf{V}$, which does not mean f is jointly convex in $\mathbf{U}$ and $\mathbf{V}$. Idea is to use **alternating minimization**, where f is never increased and lower bounded by 0 (ALS is coordinate descent, it has mathematical guarantees):

- $\mathbf{U} \leftarrow \arg\min_{\mathbf{U}} f(\mathbf{U},\mathbf{V})$
- $\mathbf{V} \leftarrow \arg\min_{\mathbf{V}} f(\mathbf{U},\mathbf{V})$
- Repeat until convergence

We can decompose f into subproblems for columns of $\mathbf{V}$ (or $\mathbf{U}$). That is, in ALS, the system computes each $\mathbf{u}_i$ independently of the other item factors and computes each $\mathbf{v}_j$ independently of the other user factors. This gives rise to potentially massive parallelization of the algorithm.

Typically we minimize $f(\mathbf{U},\mathbf{V}) + \mu\Omega(\mathbf{U},\mathbf{V})$, where $\Omega(\mathbf{U},\mathbf{V}) = ||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2$ is a norm regularizer (preventing entries becoming too large). This does not change separability structure of problem. Complete objective function becomes:

$$\begin{aligned} L(\mathbf{U},\mathbf{V}) &:= \left\|\mathbf{A} - \mathbf{U}^T\mathbf{V}\right\|_{\mathcal{I}}^2 + \lambda\left\|\mathbf{U}\right\|_F^2 + \lambda\left\|\mathbf{V}\right\|_F^2 \\ &= \sum_{(i,j)\in\mathcal{I}}(a_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2 + \lambda\sum_{i=1}^{m}\left\|\mathbf{u}_i\right\|^2 + \lambda\sum_{j=1}^{n}\left\|\mathbf{v}_j\right\|^2 \end{aligned}$$

3.3 Convex Relaxation

$$\min_{\mathbf{B}\in\mathcal{P}_k} ||\mathbf{A}-\mathbf{B}||_{\mathbf{G}}^2,\ \mathcal{P}_k := \{\mathbf{B} : ||\mathbf{B}||_* \le k\}, \text{ where } \mathcal{P}_k \supseteq \mathcal{Q}_k = \{\mathbf{B} : rank(\mathbf{B}) \le k\}$$

as nuclear norm is the tightest convex lower bound for rank:

$$rank(\mathbf{B}) \ge ||\mathbf{B}||_*, \text{ for } ||\mathbf{B}||_2 \le 1$$

Stackoverflow: A nuclear norm of a matrix is equivalent to the L1-norm of the vector of its eigenvalues. Thus, you are injecting sparsity to the vector of eigenvalues. Essentially, this sparsity means you are reducing the rank of the original matrix. iIt has been shown that the ℓ_1 norm is the convex envelope of the $|| \bullet ||_0$ pseudo-norm while the nuclear norm is the convex envelope of the rank. The nuclear norm can be thought of as a convex relaxation of the number of non-zero eigenvalues (i.e. the rank). [link]

4 Non-Negative Matrix Factorization

Given a corpus of text documents (e.g. web pages) goal is finding a low-dimensional document representation in semantic space of topics or concepts (i.e. aboutness of documents, topic models).

We represent each document as a bag-of-words (after preprocessing). Data is then reduced to co-occurrence counts $\mathbf{X} = x_{ij}$, where x_{ij} denotes occurrences of w_j in document d_i.

4.1 Probabilistic LSA

We represent documents as a mixture of topics and try to discover the topics in an unsupervised fashion. Two-stage (hierarchical) sampling: sample topic for each document and sample token given sampled topic.

Conditional independence (on topics) assumption:

$$p(w \mid d) = \sum_z p(w, z \mid d) = \sum_z p(w \mid d, z) p(z \mid d) = \sum_z p(w \mid z) p(z \mid d)$$

Log-likelihood:

$$\sum_{i,j} \log p(w_j \mid d_i)^{x_{ij}} = \sum_{i,j} x_{ij} \log \sum_z p(w_j \mid z) p(z \mid d_i) = \sum_{i,j} x_{ij} \log \sum_z v_{zj} u_{zi}$$

Missing data $q_{zij} \in \{0, 1\}$: w_j in d_i generated via z, $\sum_z q_{zij} = 1$. This leads to a lower bound from Jensen's inequality:

$$\log \sum_z q_{zij} \frac{u_{zi}, v_{zj}}{q_{zij}} \geq \sum_z q_{zij} [\log u_{zi} + \log v_{zj} - \log q_{zij}]$$

EM for pLSA is guaranteed to converge, but not guaranteed to find the local minimum:

- Expectation step:

$$q_{zij} = \frac{u_{zi} v_{zj}}{\sum_{k=1}^K u_{ki} v_{kj}} = \frac{p(w_j \mid z) p(z \mid d_i)}{\sum_{k=1}^K p(w_j \mid k) p(k \mid d_i)}$$

- Maximization step:

$$u_{zi} = \frac{\sum_j x_{ij} q_{zij}}{\sum_j x_{ij}}, v_{zj} = \frac{\sum_i x_{ij} q_{zij}}{\sum_{i,l} x_{il} q_{zil}}$$

Multiplicative update rule for NMF $V = WH$, in matrix form:

- $$\mathbf{H}_{[i,j]}^{n+1} \leftarrow \mathbf{H}_{[i,j]}^{n} \frac{((\mathbf{W}^n)^T \mathbf{V})_{[i,j]}}{((\mathbf{W}^n)^T \mathbf{W}^n \mathbf{H}^n)_{[i,j]}}$$

- $$\mathbf{W}_{[i,j]}^{n+1} \leftarrow \mathbf{W}_{[i,j]}^{n} \frac{(\mathbf{V}(\mathbf{H}^{n+1})^T)_{[i,j]}}{(\mathbf{W}^n \mathbf{H}^{n+1} (\mathbf{H}^{n+1})^T)_{[i,j]}}$$

5 Word Embeddings

Also see related section on both NLU and ML4H notes. Notes on word embeddings are currently collected in a separate file.

Regarding polysemy, disambiguating the whole corpus would be cumbersome. Even more problematically, it is unclear what the meanings are anyway.

5.1 Skip-gram and negative sampling

- **Skip-gram model:** predict context word, given active word
- Distributional semantics model = distribution of co-occurring words determines lexical semantics.
- An alternative to using windows would be to use words within same sentence.
- log-bilinear model where $w \mapsto (\mathbf{x}_w, b_w) \in \mathbb{R}^{d+1}$:

$$\log p_\theta(w \mid w') = \langle \mathbf{x}_w, \mathbf{x}_{w'} \rangle + b_w + const$$

- log-likelihood of the basic model becomes

$$\mathcal{L}(\theta; \mathbf{w}) = \sum_{t=1}^{T} \sum_{\Delta \in \mathcal{I}} \left[\langle \mathbf{x}_{w^{(t+\Delta)}}, \mathbf{x}_{w^{(t)}} \rangle + b_{w^{(t+\Delta)}} - \log \sum_{v \in \mathcal{V}} \exp\left[\langle \mathbf{x}_v, \mathbf{x}_{w^{(t)}} \rangle + b_v \right] \right]$$

- **Context Vectors:** using the same vectors for context and active words would be imposing and implicit constraint (bi-linearity), so we can use different vectors for inputs and output embeddings. Model dimensionality grows larger but this adds more flexibility to the model.
- **Negative sampling:** simplified version of noise contrastive estimation where the aim is to reduce estimation to binary classification. Negative sampling is done through sampling "random" ($P(w_j)^\alpha$, e.g. $\alpha = 3/4$) context words. Exponent (smaller than one) of the unigram probability dampens frequent words. We move to logistic function from softmax, that needs the computation of partition function with large cardinality.
- log-likelihood of the model with context vectors and negative sampling:

$$\mathcal{L}(\theta) = \sum_{(i,j) \in \Delta^+} \log \sigma\left(\langle \mathbf{x}_i, \mathbf{y}_j \rangle\right) + \sum_{(i,j) \in \Delta^-} \log \sigma\left(- \langle \mathbf{x}_i, \mathbf{y}_j \rangle\right)$$

- In a way, what skip-gram model with negative sampling does is essentially performing a low-rank decomposition of the pointwise mutual information matrix. Negative sampling idea which was a bit heuristic actually seems to be doing something quite reasonable.

5.2 GloVe

Using the same notation for output vocabulary $\mathcal{V}$ and input vocabulary $\mathcal{C}$, co-occurence matrix $\mathcal{N} = (n_{ij}) \in \mathbb{N}^{|\mathcal{V}|\cdot|\mathcal{C}|}$ where $n_{ij} = \#$ occurences of $w_i \in \mathcal{V}$ in context of $w_j \in \mathcal{C}$. It is a sparse matrix which can be computed in one pass over the corpus.

GloVe objective is the weighted least squares fit of log-counts:

$$\mathcal{H}(\theta; \mathbf{N}) = \sum_{i,j} f(n_{ij}) \left(\log n_{ij} - \log \widetilde{p}_\theta(w_i \mid w_j)\right)^2$$

with unnormalized distribution

$$\widetilde{p}_\theta(w_i \mid w_j) = \exp\left[\langle \mathbf{x}_i, \mathbf{y}_j \rangle + b_i + c_j\right]$$

and weighting function $f(n) = \min\left\{1, \left(\frac{n}{n_{\max}}\right)^\alpha\right\}$ e.g. $\alpha = \frac{3}{4}$

Weighting function is continuous. $f(x) \to 0$ when $x \to 0$. It is non-decreasing so that rare co-occurrences are not overweighted. It should be also relatively small for large values of x, so that frequent co-occurrences are not overweighted.

GloVe with $f \coloneqq 1$ solves a matrix factorization problem of the log-count matrix.

Advantage of the GloVe is that we model unnormalized probabilities which does not need the computation of the partition function.

Optimization through SGD: full gradient is often too expensive to compute for $\mathcal{H}(\theta; \mathbf{N})$, instead use stochastic optimization. We sample (i, j) such that $n_{ij} > 0$ uniformly at random. While the inner product is not big enough (i.e. w.r.t. log-counts) we move the two vectors closer to each other.

$$\mathbf{x}_i^{\text{new}} \leftarrow \mathbf{x}_i + 2\eta f(n_{ij})(\log n_{ij} - \langle \mathbf{x}_i, \mathbf{y_j} \rangle)\mathbf{y}_j$$
$$\mathbf{y}_j^{\text{new}} \leftarrow \mathbf{y}_j + 2\eta f(n_{ij})(\log n_{ij} - \langle \mathbf{x}_i, \mathbf{y_j} \rangle)\mathbf{x}_i$$

6 Data Clustering and Mixture Models

Goal is to find a meaningful partition of the data.

6.1 K-means

K-means objective function:

$$J(\mathbf{U}, \mathbf{Z}) = \sum_{i=1}^{N} \sum_{j=1}^{K} z_{ij} ||\mathbf{x}_i - \mathbf{u}_j||^2 = ||\mathbf{X} - \mathbf{U}\mathbf{Z}^T||_F^2$$

where $\mathbf{Z}$ is the assignment matrix. Computational strategy is alternating minimization.

The distance metric is the squared Euclidean distance. Note that it is not the Euclidean distance, which results in another algorithm (K-medoids).

There are different initialization strategies, one is random points. Empty clusters can be handled in different ways, one way is random re-initialization. Computational cost of each iteration is $O(knd)$. Convergence is guaranteed but K-means optimizes a non-convex objective, so global optimum is not guaranteed.

K-means++: more sophisticated seeding through incremental D^2 sampling:

$$\mathcal{U}_1 = \{\mathbf{x}_I\}, \text{ where } I \sim \text{Uniform}[1:N]$$

$$D_i := \min_{\mathbf{u}\in\mathcal{U}_k} ||\mathbf{x}_i - \mathbf{u}||, \mathcal{U}_{k+1} := \mathcal{U}_k \cup \{\mathbf{x}_I\}, \text{ where}$$

$$I \sim \text{Categorical}(\mathbf{p}),\, p_i := \frac{D_i^2}{\sum_{j=1}^N D_j^2}$$

It is more expensive, but consistently better experimental results. Theoretical guarantee is $O(\log k)$-competitiveness in expectation. Probability of choosing a datapoint which has small distance to closest centroid would be small, we can think of D_i as a measure of negligence.

Core Sets for K-means: average of uniform distribution and distance square sampling. A core set of size m gives each sample a relative weight $\frac{1}{mp_i}$.

In core set techniques you subsample the original dataset into a smaller weighted dataset (i.e. core set), run the algorithm on the smaller dataset (e.g. weighted K-means) and by construction of the core set you get guarantees for what holds for the large dataset.

$$I \sim \text{Categ}(\mathbf{p}),\, p_i := \frac{1}{2N} + \frac{D_i^2}{2\sum_{j=1}^N D_j^2},\, D_i^2 = ||\mathbf{x}_i - \mu||^2$$

ϵ-approximation guarantees (with probability δ) for

$$m \propto \frac{dk\log k + \log 1/\delta}{\epsilon^2}$$

6.2 Mixture Models

Gaussian Mixture Model (GMM):

$$p(\mathbf{x};\theta) = \sum_{j=1}^K \pi_j p(\mathbf{x};\mu_j,\mathbf{\Sigma}_j)$$

We assume latent assignment variables $\mathbf{z}$ and model their probabilities $\pi_j = P(z_j = 1)$.

Lower-bounding the log-likelihood (using Jensen's inequality):

$$\log p(\mathbf{x};\theta) = \log\left[\sum_{j=1}^K \pi_j p(\mathbf{x};\theta_j)\right] = \log\left[\sum_{j=1}^K q_j \frac{\pi_j p(\mathbf{x};\theta_j)}{q_j}\right]$$
$$\geq \sum_{j=1}^K q_j[\log p(\mathbf{x};\theta_j) + \log\pi_j - \log q_j]$$

Optimize bound with regard to the distribution q yields:

$$q_j^* = \frac{\pi_j p(\mathbf{x};\theta_j)}{\sum_{l=1}^{K} \pi_l p(\mathbf{x};\theta_l)}$$

meaning the optimal q distribution equals posterior (given the parameters).

K-means is a special case of EM algorithm with fixed covariance matrices and hard assignments. K-means algorithm can be used to find a good initialization for the EM algorithm.

7 Neural Networks

Also see related section on MP notes

We can observe that SGD converges to a better solution in fewer epochs, and that the decision boundary is sharper. SGD converges much faster than batch GD because it exploits the redundancy of the data. Moreover, the gradient noise allows the optimizer to escape sharp local minima and converge to wider minima (which has an implicit regularization effect).

A drawback of SGD is that it reduces parallelism (which can be a problem for GPUs), but this can be tackled by using sufficiently large minibatches.

8 Generative Models

Also see related section on MP notes

For $\mathbf{x} = F_\theta(\mathbf{z})$, law of the unconscious statistician (LOTUS) gives $\mathbb{E}_{\mathbf{x}}[f(\mathbf{x})] = \mathbb{E}_{\mathbf{z}}[f(F_\theta(\mathbf{z}))]$:

$$\textbf{LOTUS: } \mathbb{E}(y) = \int_{-\infty}^{\infty} y \cdot f_y(y)dy = \int_{-\infty}^{\infty} g(x) \cdot f_x(x)dx, \text{ for } y = g(x)$$

Inception score

- Quality: classify images with pretrained Inception network and compute entropy of classes (must be low)
- Diversity: look at entropy of generated images (must be high)

Fréchet Inception Distance (FID)

1. Use pretrained Inception network to extract features from generated images
2. Compare their distributions with those of a real dataset

GANs learn a loss (adapts to the task) [instead of using L2 as reconstruction loss] that tries to classify if the output image is real or fake, while simultaneously training a conditional generative model to minimize this loss.

8.1 Autoencoders, VAEs

Non-linear autoencoders:

- Powerful data-driven compression
- Can also be used for denoising (denoising autoencoder)
- However: no clear interpretation/structure of latent space
- Unclear how to sample or interpolate
- Visualization of the latent space is tricky (many dimensions are used in practice: 128+)

Variational autoencoder (motivation): We want to enforce a structure on the latent space, at the expense of the reconstruction quality. One possible choice is to force a prior on the latent space (e.g. Gaussian distribution). We can then generate by decoding a sample from the distribution. The compactness of the latent space enables smooth interpolation.

Variational autoencoder (idea): Model latent codes as soft regions instead of points. Sampling with reparameterization trick. KL divergence to enforce Gaussian prior —without it, the model would learn $\sigma \to 0$, reverting to a normal autoencoder. The latent space of a VAE approximates a Gaussian distribution, which makes sampling easy.

Using a diagonal covariance means for D dimensions, $O(D)$ parameters instead of $O(D^2)$ for full covariance matrix. Enforce $\sigma > 0$ by predicting $\log(\sigma^2)$ with the accordingly updated formulas.

Posterior collapse: Model gets stuck in a bad local minimum, no learning occurs. Can be easily detected (KL term goes to 0). Workaround: decrease strength of KL term (β-VAE).

(V)AEs tend to generate blurry images caused by pixel-wise factorization and local loss. High-frequency details are poorly correlated and hard to predict.

An **isotropic Gaussian** is one where the covariance matrix is represented by the simplified matrix $\Sigma = \sigma^2 I$. When the dimensions are independent, i.e. the distribution is isotropic, it means that the distribution is aligned with the axis.

9 Sparse Coding

Motivation is that natural signals often allow for sparse representations. We need to find suitable dictionary of atoms $\mathcal{U} = \{\mathbf{u}_1, \dots, \mathbf{u}_L\}$, such that accurate signal representation in $span(\mathcal{U})$.

Given original signal $\mathbf{x} \in \mathbb{R}^D$, key idea is to compute a linear transformation $\mathbf{z} = \mathbf{U}^T\mathbf{x}$ for orthogonal matrix $\mathbf{U}$. Matrix $\mathbf{U}$ preserves length and also energy ($||\mathbf{U}^T\mathbf{x}||^2 = ||\mathbf{x}||^2$), as a direct consequence of orthogonality.

We truncate "small" values of $\mathbf{z}$ (e.g. through thresholding) and estimate $\hat{\mathbf{z}}$. We reconstruct through inverse transform efficiently as $\mathbf{U}$ is orthogonal: $\hat{\mathbf{x}} = \mathbf{U}\hat{\mathbf{z}}$.

We evaluate representations by reconstruction error ($||\mathbf{x} - \hat{\mathbf{x}}||$) and the sparsity of the coding vector $\hat{\mathbf{z}}$.

9.1 Short review

- Inner product for $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$:

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T \mathbf{v} = \sum_{i=1}^{d} \mathbf{u}_i \mathbf{v}_i$$

- Two vectors are orthogonal if and only if their inner product is zero.
- A **basis** of a vector space is a set of vectors with the following two properties: it is linearly independent and it spans the space.
- A basis $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is called **orthonormal** if

$$\mathbf{v}_i^T \mathbf{v}_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

- A square matrix $\mathbf{A}$ with orthonormal columns is called an **orthogonal matrix**.

9.2 Coding via orthogonal transforms

A priori, there does not exist a choice of a transform that is better than all other choices. It depends on the signal type.

Basis transform via matrix multiplication is $O(D^2)$, but in practice we exploit fast transforms.

Fourier basis	Wavelet basis
• Global support	• Local support
• Good for "sine like" signals	• Good for localized signal
• Poor for localized signal	• Poor for non-vanishing signals
• $O(D \log D)$	• $O(D)$ or $O(D \log D)$

Haar wavelets span the space of square integrable functions on the unit interval. Expanding wavelets to higher dimensions: just keep doing dilation and translation.

Fourier transform makes periodic assumptions and would be horrible in very localized signals (e.g. earthquakes). We also have a lot of localized information in images (e.g. corners, local changes of color/contrast).

In PCA, we do not know the basis beforehand, instead we learn it from the data samples. It is data driven rather than based on domain knowledge. PCA is also known as Karhunen-Loeve transform or Hoteling transform. For PCA, $\mathbf{U}_K$ is data-dependent, optimal for given Σ.

We can consider images as signals in 2D and take their Fourier transform by first taking the FT of columns and then the FT of rows (interchangeable). Large changes in the pixel values correspond to high frequencies.

9.3 Overcomplete dictionaries

Overcomplete dictionaries: more atoms than dimension, as no single basis is optimally sparse for all signal classes. Trade-off is largely a computational one, decoding is involved as no closed form exists for reconstruction formula.

There is a need for overcompleteness as the signal might be a superposition of several characteristics, hence a single orthonormal basis cannot sparsely code both. Overcomplete representations can be more powerful than component analysis techniques.

Overcompleteness factor is defined as $\frac{L}{D}$. As it increases, the linear dependency between atoms is increases, potentially as well as the sparsity of the coding.

Coherence is a linear dependency measure for dictionaries:

$$m(\mathbf{U}) = \max_{i,j:i\neq j} |\mathbf{u}_i^T \mathbf{u}_j|$$

$m(\mathbf{B}) = 0$ for orthogonal basis $\mathbf{B}$ and $m([\mathbf{B}\ \mathbf{u}]) \geq \frac{1}{\sqrt{D}}$ if atom $\mathbf{u}$ is added to orthogonal $\mathbf{B}$.

Signal reconstructing is an ill-posed problem for overcomplete dictionaries.

$$\mathbf{z}^* \in \arg\min_{\mathbf{z}} ||\mathbf{z}||_0 \text{ s.t. } \mathbf{x} = \mathbf{U}\mathbf{z}$$

Without any constraint overcomplete coding is an ill-posed problem as ther more unknowns than equations: encoding not unique. Sparsity constraint makes it a NP-hard problem.

Selecting the right set of atoms in an overcomplete dictionary is a combinatorial problem, so a greedy approximation would be **Matching Pursuit**, though it is not theoretically satisfactory. The algorithm is based on greedily choosing the atoms that minimize the residual. It has a high computational complexity as the entire dictionary has to be searched at every iteration.

1. $\mathbf{z} \leftarrow \mathbf{0}, \mathbf{r} \leftarrow \mathbf{x}$
2. Select atom with maximum absolute correlation to residual
$$d^* \leftarrow \arg\max_d |\langle \mathbf{u}_d, \mathbf{r}\rangle|$$
3. Update coefficient vector and residual
$$z_{d^*} \leftarrow z_{d^*} + \langle \mathbf{u}_{d^*}, \mathbf{r}\rangle$$
$$\mathbf{r} \leftarrow \mathbf{r} - \langle \mathbf{u}_{d^*}, \mathbf{r}\rangle \mathbf{u}_{d^*}$$
4. Repeat until approximation is satisfactory

ℓ_1-norm solution: If the coherence of the dictionary is not too large, meaning that things are almost orthogonal to each other (the angles between atoms are not too small), solving this [ℓ_1-norm] problem can give you in many cases actually the exact answer or a very good approximation. Using ℓ_1-norm is also known as basis pursuit.

10 Dictionary Learning

10.1 Compressive Sensing

Given knowledge about a signal's sparsity, the signal may be reconstructed, through optimization, with even fewer samples than the Nyquist–Shannon sampling theorem requires. Idea is to compress data while gathering. It decreases acquisition time, power consumption and required storage space.

Original signal $\mathbf{x} \in \mathbb{R}^D$ is K-sparse in orthonormal basis $\mathbf{U}$:

$$\mathbf{x} = \mathbf{U}\mathbf{z} \text{ such that } ||\mathbf{z}||_0 = K$$

Main idea is to acquire a set $\mathbf{y}$ of M linear combinations of signal (so-called compressive measurements) and then reconstruct the signal from those measurements:

$$\mathbf{y} = \langle \mathbf{w}_k, \mathbf{x} \rangle,\ k = 1, \dots, M$$

$$\mathbf{y} = \mathbf{W}\mathbf{x} = \mathbf{W}\mathbf{U}\mathbf{z} = \mathbf{\Theta}\mathbf{z}, \text{ with } \mathbf{\Theta} = \mathbf{W}\mathbf{U} \in \mathbb{R}^{M \times D}$$

If $M \ll D$, measured signal $\mathbf{y}$ is much shorter than $\mathbf{x}$. Given any orthonormal basis $\mathbf{U}$, we can obtain a stable reconstruction for any K-sparse, compressible signal.

Sufficient conditions:

- $\mathbf{W}$ is a Gaussian random projection, i.e. $w_{ij} \sim \mathcal{N}(0, \frac{1}{D})$
- $M \geq cK \log(\frac{D}{K})$ for some constant c

Given $\mathbf{z}$, we can easily reconstruct $\mathbf{x}$ via $\mathbf{x} = \mathbf{U}\mathbf{z}$, but finding $\mathbf{z}$ is ill-posed as it requires solving an underdetermined matrix equation. However we can impose the constraint that $\mathbf{z}$ is constraint and optimize it through convex optimization or matching pursuit (i.e. find the sparsest such that equality holds):

$$\mathbf{z}^* \in \arg\min_{\mathbf{z}} ||\mathbf{z}||_0, \text{ s.t. } \mathbf{y} = \mathbf{\Theta}\mathbf{z}$$

10.2 Dictionary Learning

- Fixed orthonormal basis
  - + Efficient coding by matrix multiplication $\mathbf{z} = \mathbf{U}^{\mathbf{T}}\mathbf{x}$
  - − Only sparse for specific classes of signals (strong a priori assumptions)
- Fixed overcomplete basis
  - + Sparse coding for several signal classes
  - − Finding sparsest code may require approximation algorithm (e.g. matching pursuit) and problematic if dictionary size L and coherence $m(\mathbf{U})$ are large
- Learning the dictionary
  - + We adapt a dictionary to signal characteristics, meaning same approximation error achievable with smaller L

– Challenging to solve the matrix factorization problem $\mathbf{X} \approx \mathbf{UZ}$ subject to sparsity constraint on $\mathbf{Z}$ and column/atom norm constraint on $\mathbf{U}$.

$$(\mathbf{U}^*, \mathbf{Z}^*) \in \arg\min_{\mathbf{U},\mathbf{Z}} ||\mathbf{X} - \mathbf{U} \cdot \mathbf{Z}||_F^2$$

Objective not jointly convex in $\mathbf{U}$ and $\mathbf{Z}$, convex in either $\mathbf{U}$ or $\mathbf{Z}$ (with unique minimum).

Iterative greedy minimization:

- Coding step: $\mathbf{Z}^{t+1} \in \arg\min_{\mathbf{Z}} ||\mathbf{X} - \mathbf{U^t} \cdot \mathbf{Z}||_F^2$,
  subject to $\mathbf{Z}$ being sparse (non-convex) and $\mathbf{U}$ being fixed
- Dictionary update step: $\mathbf{U}^{t+1} \in \arg\min_{\mathbf{U}} ||\mathbf{X} - \mathbf{U} \cdot \mathbf{Z^{t+1}}||_F^2$,
  subject to $||\mathbf{u_l}||_2 = 1$ for all $l = 1, \ldots, L$ and $\mathbf{Z}$ being fixed

Coding Step

In coding step, residual is column separable as $||\mathbf{R}||_F^2 = \sum_{i,j} r_{i,j}^2 = \sum_j ||\mathbf{r}_j||_2^2$, meaning N independent sparse coding steps: for all $n = 1, \ldots, N$

$$\mathbf{z}_n^{t+1} \in \arg\min_{\mathbf{z}} ||\mathbf{z}||_0, \text{ s.t. } ||\mathbf{x_n} - \mathbf{U^t}\mathbf{z}||_2 \leq \sigma ||\mathbf{x}_n||_2$$

Dictionary Update

Residual not separable in atoms (columns of $\mathbf{U}$), choice of one atom obviously influences what other atoms should capture. Approximation by updating one atom at a time (t is the time index):

- Set $\mathbf{U} = [\mathbf{u}_1^t \ldots \mathbf{u}^l \ldots \mathbf{u}_L^t]$, i.e fix all atoms except $\mathbf{u}_l$
- Isolate $\mathbf{R}_l^t$, the residual that is due to atom $\mathbf{u}_l$

$$\begin{aligned} &\left\|\mathbf{X} - [\mathbf{u}_1^t \ldots \mathbf{u}^l \ldots \mathbf{u}_L^t] \cdot \mathbf{Z}^{t+1}\right\|_F^2 \\ =& \left\|\mathbf{X} - \left(\sum_{e \neq l} \mathbf{u}_e^t (\mathbf{z}_e^{t+1})^T + \mathbf{u}_l (\mathbf{z}_l^{t+1})^T\right)\right\|_F^2 \\ =& \left\|\mathbf{R}_l^t - \mathbf{u}_l (\mathbf{z}_l^{t+1})^T\right\|_F^2 \end{aligned} \tag{1}$$

- Find $\mathbf{u}_l^*$ that minimizes $\mathbf{R}_l^t$, subject to $||\mathbf{u}_l^*||_2 = 1$:

  $\mathbf{u}_l (\mathbf{z}_l^{t+1})^T$ is an outer product, i.e. a matrix. We can approximate the residual by a rank 1 matrix (cf. Eckart-Young theorem), via SVD of of $\mathbf{R}_l^t$

$$\mathbf{R}_l^t = \widetilde{\mathbf{U}} \mathbf{\Sigma} \widetilde{\mathbf{V}}^T = \sum_i \sigma_i \tilde{\mathbf{u}}_i \tilde{\mathbf{v}}_i^T$$

  $\mathbf{u}_l^* = \tilde{\mathbf{u}}_1$ is first left-singular vector. $\|\mathbf{u}_l^*\|_2 = 1$ is naturally satisfied. We also update the l-th row of $\mathbf{Z}$ here.

Initialization:

- Random atoms: sample $\mathbf{u}_l^0$ on a unit sphere
- Samples from X: sample uniformly from the data

Algorithm 1 K-SVD

```
1: Input: $\mathbf{X} = \mathbb{R}^{D\times N}$; $\mathbf{U} = \mathbb{R}^{D\times L}$; $\mathbf{Z} = \mathbb{R}^{L\times N}$;
2: Output: Updated dictionary $\mathbf{U}$
3: for $l \leftarrow 1$ to $L$ do
4:     $\mathbf{u}_{(:,l)} \leftarrow \mathbf{0}$
5:     $\mathcal{N} \leftarrow \{n \mid Z_{ln} \neq 0, 1 \leq n \leq N\}$ % Active data points
6:     $\mathbf{R} \leftarrow \mathbf{X}_{(:,\mathcal{N})} - \mathbf{U}\mathbf{Z}_{(:,\mathcal{N})}$ % Residual
7:     $\mathbf{g} \leftarrow \mathbf{z}^T_{(l,\mathcal{N})}$
8:     $\mathbf{h} \leftarrow \mathbf{Rg}/||\mathbf{Rg}||$ % Power iteration
9:     $\mathbf{g} \leftarrow \mathbf{R}^T\mathbf{h}$
10:    $\mathbf{u}_{(:,l)} \leftarrow \mathbf{h}$ % Update
11:    $\mathbf{z}_{(l,\mathcal{N})} \leftarrow \mathbf{g}^T$
```

- Fixed overcomplete dictionary: use overcomplete DCT etc.

- Coding step: can be done for each image separately
- Instead of an ϵ [in the coding step error minimization] you many want something that is scale-invariant over different inputs
- In dictionary update step we only modify the data points that use that atom. We can zero them out but never make a zero data point non-zero. That is, as you compute the atoms, you change the coefficients but not the sparsity pattern, as there's this weird coupling. The data points do not use new atoms.
- In the simplest case you just one power iteration.
- Here [at the dictionary update step] you recompute the encoding just to avoid being stuck in a local optimum, just for the purpose of finding the atoms. Sparsity pattern might change when you recompute the **Z** matrix [in the coding step].
- No guarantees for the algorithm (it is greedy), but it's a nice workhorse in the area. It also depends on the chosen ordering as it is a greedy process.

11 Appendix

- $$p(x) := \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$
- $$p(\mathbf{x}) := \frac{1}{(2\pi)^{\frac{d}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right)$$

11.1 Convexity

- f is called convex if

$$\forall x_1, x_2 \in X, \forall t \in [0,1] : f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

- f is called strictly convex if

$$\forall x_1 \neq x_2 \in X, \forall t \in (0,1) : f(tx_1 + (1-t)x_2) < tf(x_1) + (1-t)f(x_2)$$

- A function f is said to be (strictly) concave if $-f$ is (strictly) convex.
- A differentiable function of one variable is convex on an interval if and only if the function lies above all of its tangents:

$$f(x) \geq f(y) + f'(y)(x-y)$$

- A twice differentiable function of one variable is convex on an interval if and only if its second derivative is non-negative there; this gives a practical test for convexity.
- A convex set is closed under convex combinations:

$$(1-t)x + ty \in C, \forall x, y \in C \text{ and } t \in [0,1]$$

  - The empty set and the whole space are convex.
  - The intersection of any collection of convex sets is convex.
  - The union of a sequence of convex sets is convex, if they form a non-decreasing chain for inclusion. For this property, the restriction to chains is important, as the union of two convex sets need not be convex.

11.2 Singular Value and Eigenvalue Decompositions

Singular Value Decomposition:

The singular value decomposition (SVD) factorizes a linear operator $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ such that $A = USV^T$ with U and $m \times r$ orthonormal matrix spanning A's column space $im(A)$, S an $r \times r$ diagonal matrix of singular values, and V an $n \times r$ orthonormal matrix spanning A's row space $im(A^T)$.

Eigenvalue Decomposition:

The eigenvalue decomposition applies to mappings from $\mathbb{R}^n$ to itself, i.e., a linear operator $A : \mathbb{R}^n \rightarrow \mathbb{R}^n$ described by a square matrix. An eigenvector e of A is a vector that is mapped to a scaled version of itself, i.e., $Ae = \lambda e$, where λ is the corresponding eigenvalue. Furthermore, if A is full rank $r = n$ then A can be factorized as $A = E\Lambda E^{-1}$. In fact, if and only if A is symmetric and positive definite (abbreviated SPD), we have that the SVD and the eigen-decomposition coincide $A = USU^T = E\Lambda E^{-1}$ with $U = E$ and $S = \Lambda$. Given a non-square matrix $A = USV^T$, two matrices and their factorization are of special interest:

$$A^T A = VS^2V^T$$

$$AA^T = US^2U^T$$

Differences:

- The singular value decomposition is very general in the sense that it can be applied to any $m \times n$ matrix whereas eigenvalue decomposition can only be applied to diagonalizable matrices (i.e. non-defective square matrices).
- The basis of the eigendecomposition is not necessarily orthogonal, the eigenbasis of the SVD is orthonormal.
- In the SVD, the nondiagonal matrices U and V are not necessarily the inverse of one another. They are usually not related to each other at all. In the eigendecomposition the nondiagonal matrices E and E^{-1} are inverses of each other.

11.3 Matrix norms

Induced p-norm shows how much an input vector is maximally stretched. Induced 1-norm is the absolute column sum of a matrix and induced ∞-norm is the absolute row sum of the matrix.

$$||A||_p := \sup\{||Ax||_p : ||x||_p = 1\}, \text{ where } ||x||_p := \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$

Schatten p-norms are computed as the p-norm of the vector of singular values of a matrix (e.g. rank of a matrix is its "Schatten 0-norm"):

$$\|\mathbf{A}\|_p := \|\sigma(\mathbf{A})\|_p$$

- **Frobenius norm:**

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{\text{trace}(A^TA)} = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(A)}$$

  Frobenius norm of a matrix is sum of its squared singular values, so it only depends on the singular values of that matrix. It is the Schatten p-norm for $p = 2$. It is also called Hilbert-Schmidt norm and $L_{2,2}$ norm.

- **Nuclear norm:**

$$||A||_* = \text{trace}(\sqrt{A^*A}) = \sum_{i=1}^{\min\{m,n\}} \sigma_i(A)$$

  Nuclear norm of a matrix is the ℓ_1 norm of the vector of its eigenvalues. It is the Schatten p-norm for $p = 1$. It is also called trace norm.

- **Spectral norm:**

$$||A||_2 = \sqrt{\lambda_{\max}(A^*A)} = \sigma_{\max}(A)$$

  The spectral norm of a matrix A is the largest singular value of A i.e. the square root of the largest eigenvalue of the matrix A^*A. It is induced p-norm for $p = 2$ and Schatten p-norm for $p = \infty$.