CIL Cheat Sheet 2015

Jan Wilken Dörrie

Linear Algebra Primer

Equivalent Conditions

For $\mathbf{A} \in \mathbb{R}^{M \times M}$ the following conditions are equivalent: $\mathbf{A}$ has an inverse $\mathbf{A}^{-1}$; $\mathrm{rank}(\mathbf{A}) = M$; $\mathrm{range}(\mathbf{A}) = \mathbb{R}^M$; $\mathrm{null}(\mathbf{A}) = \{\mathbf{0}\}$; 0 is not an eigenvalue of $\mathbf{A}$; 0 is not a singular value of $\mathbf{A}$.

Norms

Vector norms

A *norm* is a function $\|\cdot\| : V \to \mathbb{R}$ quantifying the size of a vector. It must satisfy

1) Positive scalability: $\|a \cdot \mathbf{x}\| = |a| \cdot \|\mathbf{x}\|$ for $a \in \mathbb{R}$
2) Triangle inequality: $\|\mathbf{x} + \mathbf{y}\| \le \|\mathbf{x}\| + \|\mathbf{y}\|$, $\mathbf{x}, \mathbf{y} \in V$.
3) Seperability: $\|\mathbf{x}\| = 0$ implies $\mathbf{x} = 0$.

- Most common are *p-norms*: $\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$
- Special case is *Euclidean norm*: $\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^n x_i^2}$
- The "0-*norm*" is $\|\mathbf{x}\|_0 := |\{x_i \mid x_i \neq 0\}|$

Matrix norms

We can also define norms on matrices, satisfying the properties described above. $\mathbf{A} \in \mathbb{R}^{M \times N}$:

- *Frobenius*: $\|\mathbf{A}\|_F := \sqrt{\sum_{ij} a_{ij}^2} = \sqrt{\sum_{i=1}^{\min(M,N)} \sigma_i^2}$
- *p-norms for matrices*: $\|\mathbf{A}\|_p := \sup\{\|\mathbf{A}\mathbf{x}\|_p / \|\mathbf{x}\|_p\}$
- *Euclidean*: $\|\mathbf{A}\|_2 := \sup\{\|\mathbf{A}\mathbf{x}\|_2 / \|\mathbf{x}\|_2\} = \sigma_{\max}$
- *Nuclear norm*: $\|\mathbf{A}\|_* := \sum_{i=1}^{\min(M,N)} \sigma_i$

Statistics

Kullback-Leibler Divergence

- Divergence between discrete probability distributions P and Q: $D_{\mathrm{KL}}(P\|Q) = \sum_{\omega \in \Omega} P(\omega) \log\left(\frac{P(\omega)}{Q(\omega)}\right)$.
- Properties of the Kullback-Leibler Divergence
  - $D_{\mathrm{KL}}(P\|Q) \ge 0$.
  - $D_{\mathrm{KL}}(P\|Q) = 0$ if and only if P and Q are identical.
  - $D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$!
  - caution: the KL-Divergence is not symmetric, therefore it is not a metric/distance!

Dimension Reduction

Principal Component Analysis (PCA)

Orthogonal linear projection of high dimensional data onto low dimensional subspace. Objectives:

1) Minimize error $\|\mathbf{x} - \tilde{\mathbf{x}}\|_2$ of point $\mathbf{x}$ and its approximation $\tilde{\mathbf{x}}$.
2) Preserve information: maximize variance.

Both objectives are shown to be formally equivalent.

Statistics of Projected Data:

- Mean of the data: sample mean $\bar{\mathbf{x}}$
- Covariance of the data:

$$\mathbf{\Sigma} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top$$

Solution: Eigenvalue Decomposition: The eigenvalue decomposition of the covariance matrix $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ contains all relevant information.

- For $K \le D$ dimensional projection space: Choose K eigenvectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_K\}$ with largest associated eigenvalues $\{\lambda_1, \ldots, \lambda_K\}$.

Singular Value Decomposition

Theorem (Eckart-Young). *Let $\mathbf{A}$ be a matrix of rank R, if we wish to approximate $\mathbf{A}$ using a matrix of a lower rank K then, $\tilde{\mathbf{A}} = \sum_{k=1}^K d_k \mathbf{u}_k \mathbf{v}_k^\top$ is the closest matrix in the Frobenius norm. (Assumes ordering of singular values $d_k \ge d_{k+1}$)*

Clustering

K-Means

Motivation:

- Given: set of data points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^D$
- Goal: find *meaningful partition* of the data
  - i.e. a labeling of each data point with a unique label

$$\pi : \{1, \ldots, N\} \to \{1, \ldots, K\} \text{ or } \pi : \mathbb{R}^D \to \{1, \ldots, K\}$$

  - note: numbering of clusters is arbitrary
  - k-th cluster recovered by $\pi^{-1}(k) \subseteq \{1, \ldots, N\}$ or $\subseteq \mathbb{R}^D$

Vector Quantization:

- Partition of the space $\mathbb{R}^D$
- Clusters represented by *centroids* $\mathbf{u}_k \in \mathbb{R}^D$
- Mapping induced via nearest centroid rule

$$\pi(\mathbf{x}) = \underset{k=1,\ldots,K}{\operatorname{argmin}} \|\mathbf{u}_k - \mathbf{x}\|_2$$

Objective Function for K-Means:

- Useful notation: represent π via indicator matrix $\mathbf{Z}$:

$$z_{kn} := [\pi(\mathbf{x}_n) = k]$$

- K-means Objective function

$$J(\mathbf{U}, \mathbf{Z}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{kn} \|\mathbf{x}_n - \mathbf{u}_k\|_2^2 = \|\mathbf{X} - \mathbf{U}\mathbf{Z}\|_F^2,$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$ and $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K] \in \mathbb{R}^{D \times K}$

K-means Algorithm: Optimal Assignment:

- Compute optimal assignment $\mathbf{Z}$, given centroids $\mathbf{U}$
  - minimize each column of $\mathbf{Z}$ separately

$$\mathbf{z}_{\bullet n}^* = \underset{z_{1n}, \ldots, z_{Kn}}{\operatorname{argmin}} \sum_{k=1}^{K} z_{kn} \|\mathbf{x}_n - \mathbf{u}_k\|_2^2$$

  - optimum is attained by mapping to the closest centroid

$$z_{kn}^*(\mathbf{U}) = \left[k = \underset{l}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{u}_l\|_2\right]$$

K-means Algorithm: Optimal Assignment:

- Compute optimal choice of $\mathbf{U}$, given assignments $\mathbf{Z}$
  - continuous variables: compute gradient and set to zero (necessary optimality condition)
  - look at (partial) gradient for every centroid $\mathbf{u}_k$

$$\nabla_{\mathbf{u}_k} J(\mathbf{U}, \mathbf{Z}) = \sum_{n=1}^{N} z_{kn} \nabla_{\mathbf{u}_k} \|\mathbf{x}_n - \mathbf{u}_k\|_2^2 = -2 \sum_{n=1}^{N} z_{kn} (\mathbf{x}_n - \mathbf{u}_k)$$

  - setting gradient to zero

$$\nabla_{\mathbf{U}} J(\mathbf{U}, \mathbf{Z}) \stackrel{!}{=} 0 \implies \mathbf{u}_k^*(\mathbf{Z}) = \frac{\sum_{n=1}^{N} z_{kn} \mathbf{x}_n}{\sum_{n=1}^{N} z_{kn}}$$

K-means Algorithm: Analysis:

- Computational cost of each iteration is $O(KND)$
- K-means convergence is guaranteed
- K-means optimizes a non-convex objective. Hence we are not guaranteed to find the global optimum.
- Finds a local optimum $(\mathbf{U}, \mathbf{Z})$ in the following sense
  - for each $\mathbf{Z}'$ with $\frac{1}{2} \|\mathbf{Z} - \mathbf{Z}'\|_0 = 1$ (differs in one assignment)
  - $J(\mathbf{U}^*(\mathbf{Z}'), \mathbf{Z}') \geq J(\mathbf{U}, \mathbf{Z})$
  - may gain by changing assignments of ≥ 2 points
- K-means algorithm can be used to compress data
  - with information loss, if $K < N$
  - store only the centroids and the assignments

Mixture Models

Gaussian Mixture Model (GMM):

$$p_\theta(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

where $\pi_k \geq 0$, $\sum_{k=1}^{K} \pi_k = 1$.

Complete Data Distribution:

- Explicitly introduce latent variables in the generative model
- Assignment variable (for a generic data point) $z_k \in \{0, 1\}$, $\sum_{k=1}^{K} z_k = 1$
- We have that $\Pr(z_k = 1) = \pi_k$ or $p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$
- Joint distribution over $(\mathbf{x}, \mathbf{z})$ (*complete data* distribution) $p(\mathbf{x}, \mathbf{z}) = \prod_{k=1}^{K} [\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_k}$

Posterior Assignments: Posterior probabilities for assignments

$$\Pr(z_k = 1 \mid \mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^{K} \pi_l \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}$$

Lower Bounding the Log-Likelihood:

- Expectation Maximization
  - maximize a lower bound on the log-likelihood
  - systematic way of deriving a family of bounds
  - based on complete data distribution
- Specifically:

$$\begin{aligned}
\ln p_\theta(\mathbf{x}) = \ln \sum_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) &= \ln \sum_{k=1}^{K} p(\mathbf{x}, \theta_k) \pi_k \\
&= \ln \sum_{k=1}^{K} q_k \frac{p(\mathbf{x}; \theta_k) \pi_k}{q_k} \\
&\geq \sum_{k=1}^{K} q_k \left[\ln p(\mathbf{x}, \theta_k) + \ln \pi_k - \ln q_k\right]
\end{aligned}$$

  - follows from Jensen's inequality (concavity of logarithm)
  - can be done for the contribution of each data point (additive)

Mixture Model: Expectation Step:

$$q_k = \frac{\pi_k \, p(\mathbf{x}; \theta_k)}{\sum_{l=1}^{K} \pi_l \, p(\mathbf{x}, \theta_l)} = \Pr(z_k = 1 \mid \mathbf{x})$$

Mixture Model: Maximization Step:

$$\pi_k^* = \frac{1}{N} \sum_{n=1}^{N} q_{kn}$$

$$\boldsymbol{\mu}_k^* = \frac{\sum_{n=1}^{N} q_{kn} \mathbf{x}_n}{\sum_{n=1}^{N} q_{kn}}$$

$$\boldsymbol{\Sigma}_k^* = \frac{\sum_{n=1}^{N} q_{kn} (\mathbf{x}_n - \boldsymbol{\mu}_k^*)(\mathbf{x}_n - \boldsymbol{\mu}_k^*)^\top}{\sum_{n=1}^{N} q_{kn}}$$

AIC and BIC:

- Trade-off: achieve balance between data fit — measured by likelihood $p(\mathbf{X} \mid \theta)$ — and complexity. Complexity can be measured by the number of free parameters $\kappa(\cdot)$.
- Different Heuristics for choosing K
  - *Akaike Information Criterion* (AIC)

$$\mathrm{AIC}(\theta \mid \mathbf{X}) = -\ln p_\theta(\mathbf{X}) + \kappa(\theta)$$

  - *Bayesian Information Criterion* (BIC)

$$\mathrm{BIC}(\theta \mid \mathbf{X}) = -\ln p_\theta(\mathbf{X}) + \frac{1}{2} \kappa(\theta) \ln N$$

- Generally speaking, the BIC criterion penalizes complexity more than the AIC criterion.

Non-Negative Matrix Factorization

Non-Negative Matrix Factorization:

- *Document-term matrix* $\mathbf{X} \in \mathbb{R}_{\geq 0}^{D \times N}$ storing the word counts for each document:

$$\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$$

x_{dn}: Frequency of the d-th word in the n-th document.
- *Non-negative matrix factorization* (NMF) of $\mathbf{X}$:

$$\mathbf{X} \approx \mathbf{UZ}$$

  - with $\mathbf{U} \in \mathbb{R}_{\geq 0}^{D \times K}$ and $\mathbf{Z} \in \mathbb{R}_{\geq 0}^{K \times N}$
    - N: number of documents
    - D: vocabulary size
    - K: number of dimensions (design choice)
    - data reduction: $(D + N)K \ll DN$

pLSI — Generative Model:

- For a given *document* sample $\mathrm{len}(\mathrm{document})$ words by a two-stage procedure:
  - sample a topic according to $P(\mathrm{topic} \mid \mathrm{document})$
  - sample a word according to $P(\mathrm{word} \mid \mathrm{topic})$
- Key assumption: *conditional independence* of word and document given topic
- Conditional distribution of a word, given a document:

$$P(\mathrm{word} \mid \mathrm{document}) = \sum_{k=1}^{K} P(\mathrm{word} \mid \mathrm{topic}_k) P(\mathrm{topic}_k \mid \mathrm{document})$$

- Side note: how to sample a "new" document? Can use fully generative model of LDA.

pLSI — Matrix Factorization View:

- *Normalize* the elements of $\mathbf{X}$ so that they correspond to relative frequencies:

$$T := \sum_{d=1}^{D} \sum_{n=1}^{N} x_{dn}, \qquad x_{dn} \leftarrow \frac{x_{dn}}{T}$$

- *Matrix Factorization*
  - pLSI can be understood as a matrix factorization of the form $\mathbf{X} \approx \mathbf{UZ}$, with $\mathbf{U} \in \mathbb{R}_{\geq 0}^{D \times K}$, and $\mathbf{Z} \in \mathbb{R}_{\geq 0}^{K \times N}$
  - where additionally we have the constraints:
    * $\sum_{d=1}^{D} u_{dk} = 1 (\forall k)$, identify $u_{dk} \equiv P(\mathrm{word}_d \mid \mathrm{topic}_k)$
    * $\sum_{k,n} z_{kn} = 1$, identify $z_{kn} \equiv P(\mathrm{topic}_k \mid \mathrm{document}_n) P(\mathrm{document}_n)$

pLSI — Parameter Estimation:

- Goal: maximize the likelihood of the data under the model
- Data: the relative frequencies $\mathbf{X}$
- Probabilistic model: $P(\mathrm{word}_d, \mathrm{document}_n) = \sum_{k=1}^{K} P(\mathrm{word}_d \mid \mathrm{topic}_k) P(\mathrm{topic}_k \mid \mathrm{document}_n) = (\mathbf{UZ})_{dn}$
- *Log likelihood*: $\log \mathcal{L}(\mathbf{U}, \mathbf{Z}; \mathbf{X}) = \log P(\mathbf{X}; \mathbf{U}, \mathbf{Z}) = \sum_{d=1}^{D} \sum_{n=1}^{N} x_{dn} \log \sum_{k=1}^{K} u_{dk} z_{kn}$

EM for pLSI — Variational Likelihood:

- Follow similar recipe as for Gaussian Mixture Model
- Reindex the observations in a per token manner with $t = 1, \ldots, T$
  - pairs of word/documents indexes (d_t, n_t)
  - note that $\sum_{t=1}^{T} f(d_t, n_t) = \sum_{d=1}^{D} \sum_{n=1}^{N} x_{dn} f(d, n)$ for arbitrary functions f
- *Variational Likelihood*

$$\begin{aligned} &\log P(\mathbf{X}; \mathbf{U}, \mathbf{Z}) \\ &= \sum_{t=1}^{T} \log (\mathbf{UZ})_{d_t n_t} = \sum_{t=1}^{T} \log \left[\sum_{k=1}^{K} u_{d_t k} z_{k n_t} \right] \\ &\geq \sum_{t=1}^{T} \max_{q \in \mathcal{S}_K} \sum_{k=1}^{K} q_k [\log u_{d_t k} + \log z_{k n_t} - \log q_k] \end{aligned}$$

  - $\mathcal{S}_K := \left\{ x \in \mathbb{R}^K \mid x \geq 0, \sum_{k=1}^{K} x_k = 1 \right\}$ (probability simplex)

EM for pLSI — Derivation of E-step:

- Compute the argmin in the variational bound

$$q_t^* = \operatorname*{argmax}_{q \in \mathcal{S}_K} \sum_{k=1}^{K} q_k [\log u_{d_t k} + \log z_{k n_t} + \log q_k]$$

- Form Lagrangian and differentiate

$$\frac{\partial}{\partial q_k} \{ q_k [\log u_{d_t k} + \log z_{k n_t} - \log q_k - \lambda_t^*] \} \stackrel{!}{=} 0$$

$$\implies q_{tk}^* \propto u_{d_t k} z_{k n_t}, \text{ i.e. } q_{tk}^* = \frac{u_{d_t k} z_{k n_t}}{\sum_{l=1}^{K} u_{d_t l} z_{l n_t}}$$

- q_{tk}^* = posterior probability that t-th token (i.e. word with index d_t in document with index n_t) has been generated from topic k

EM for pLSI — Derivation of M-step:

- Differentiate lower bound with plugged in optimal choices for q_t^* $(t = 1, \ldots, T)$
- *M*-step solution for $\mathbf{U}$ and $\mathbf{Z}$

$$u_{dk}^* = \frac{\sum_{t:d_t = d} q_{tk}^*}{\sum_{t=1}^{T} q_{tk}^*} \qquad z_{kn}^* = \frac{\sum_{t:n_t = n} q_{tk}^*}{T}$$

Sparse Coding

Optimization

Coordinate Descent: Idea: Update one coordinate at a time, while keeping others fixed.

- Algorithm:
  - initialize $\mathbf{x}^{(0)} \in \mathbb{R}^D$
  - for $t = 0, \ldots, \mathrm{maxIter}$
    * $d \leftarrow \mathcal{U}\{1, D\}$
    * $u^* \leftarrow \mathrm{argmin}_{u \in \mathbb{R}} f \left(x_1^{(t)}, \ldots, x_{d-1}^{(t)}, u, x_{d+1}^{(t)}, \ldots, x_D^{(t)} \right.$
    * $x_d^{(t+1)} \leftarrow u^*, \quad x_{d'}^{(t+1)} \leftarrow x_{d'}^{(t)}$ for $d' \neq d$

Gradient Descent Method:

- Algorithm:
  - initialize $\mathbf{x}^{(0)} \in \mathbb{R}^D$
  - for $t = 0, \ldots, \mathrm{maxIter}$
    * $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \gamma \nabla f \left(\mathbf{x}^{(t)} \right)$
- simple to implement
- good scalability and robustness
- *stepsize* γ usually decreasing with $\gamma \approx \frac{1}{t}$

Stochastic Gradient Descent:

- Optimization Problem Structure: minimize $f(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} f_n(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^D$
- Algorithm:
  - initialize $\mathbf{x}^{(0)} \in \mathbb{R}^D$
  - for $t = 0, \ldots, \mathrm{maxIter}$
    * $n \leftarrow \mathcal{U}\{1, N\}$
    * $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \gamma \nabla f_n \left(\mathbf{x}^{(t)} \right)$

Duality for Constrained Optimization:

- Constrained Problem Formulation (Standard Form): minimize $f(\mathbf{x})$ subject to $g_i(\mathbf{x}) \leq 0$, $i = 1, \ldots, m$, $h_i(\mathbf{x}) = 0$, $i = 1, \ldots, p$
- Unconstrained Problem: minimze $f(\mathbf{x}) + \sum_{i=1}^{m} I_{-}(g_i(\mathbf{x})) + \sum_{i=1}^{p} I_0(h_i(\mathbf{x}))$. I_{-} and I_0 are "brickwall" indicator functions.

Dual Problem:

- Lagrangian: $L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) := f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x})$
- Lagrange dual function: $d(\boldsymbol{\lambda}, \boldsymbol{\nu}) := \inf_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$
- Lagrange *dual problem*: maximize $d(\boldsymbol{\lambda}, \boldsymbol{\nu})$ subject to $\boldsymbol{\lambda} \geq \mathbf{0}$.
  - It is always a lower bound on the primal value $f(\mathbf{x})$ of any feasible $\mathbf{x}$ and thus a lower bound on the unknown solution value $f(\mathbf{x}^*)$ of the primal problem.
  - Strong Duality: If the primal optimization problem is convex and under some additional conditions, the solution value of the dual problem is *equal* to the solution value $f(\mathbf{x}^*)$ of the primal problem.

Convexity:

- Convex Set: A set Q is convex if for any $\mathbf{x}, \mathbf{y} \in Q$ and any $\theta \in [0, 1]$, we have $\theta\mathbf{x} + (1-\theta)\mathbf{y} \in Q$.
- Convex Function: A function $f\colon \mathbb{R}^D \to \mathbb{R}$ is convex if $\operatorname{dom} f$ is a convex set and if for all $\mathbf{x}, \mathbf{y} \in \operatorname{dom} f$, and $\theta \in [0, 1]$ we have $f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y})$.
- Convex Optimization: Convex Optimization Problems are of the form $\min f(\mathbf{x})$ s.t. $\mathbf{x} \in Q$ where both f is a convex function and Q is a convex set (note: $\mathbb{R}^D$ is convex). In Convex Optimization Problems every local minimum is a *global minimum*.

ROBUST PCA