Extended CIL Summary
FS 2013

Pascal Spörri
pascal@spoerri.io

August 10, 2013

This summary is based on the course slides of the Computational Intelligence Lab slides[1] from spring semester 2013.

[1] http://cil.inf.ethz.ch

Contents

Part I.
Dimensionality Reduction

Select the *most interesting* dimensions.

1. Intrinsic Dimensionality

Pairwise Distances

Assume components of data $x = (x_1, \ldots, x_D)^T \in \mathbb{R}^D$ are i.i.d. Gaussian distributed:

$$x_d \sim \mathcal{N}(0,1) \implies x_d - y_d \sim \mathcal{N}(0,2).$$

Using χ^2-distribution:

$$\frac{1}{2}(x_d - y_d)^2 \sim \chi^2(1),$$

and extending to D dimensions:

$$\frac{1}{2}\sum_{d=1}^{D}(x_d - y_d)^2 \sim \chi^2(D) = \Gamma\left(\frac{D}{2}, 2\right)$$

$$\text{Recall: } \forall z, k, \theta > 0,\ \Gamma(z;k,\theta) = \frac{\theta^k}{\Gamma(k)} y^{k-1} e^{-\theta z}$$

Hence, the dimension-normalised squared distance is:

$$\frac{1}{D}\sum_{d=1}^{D}(x_d - y_d)^2 \sim \Gamma\left(\frac{D}{2}, \frac{4}{D}\right)$$

is Gamma distributed with mean 2 and variance $\frac{8}{D}$.
$\Gamma\left(\frac{D}{2}, \frac{4}{D}\right)$ tends towards normality with shrinking width for large D. Therefore, most points have *constant* pairwise distances in this limit.

2. Principal Component Analysis

Objectives of PCA:

1. Minimise error $\|x_n - \tilde{x}_n\|$ of point x_n and its approximation $\tilde{x}_n$.
2. Reveal "interesting" information: maximise *variance*.

Both objectives are show to be formally equivalent.
Consider a set of observations $\{x_n\},\ n = 1, \ldots, N$ and $x_n \in \mathbb{R}^D$.

Goal Project data onto $K < D$ dimensional space while maximising variance of the projected data.

For K = 1 Define direction of projection as u_1. Set $\|u_1\|_2 = 1$ (only the direction of the projection is important.

2.1. Statistics of Projected Data

Original Data

Mean is given by the sample mean $\bar{x}$.

Covariance of the Data:

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T$$

Projected Data

Mean is given by: $u_1^T\bar{x}$.

Variance is given by:

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\left\{u_1^T x_n - u_1^T\bar{x}\right\}^2 &= \frac{1}{N}\sum_{n=1}^{N}\left\{u_1^T(x_n - \bar{x})\right\}^2 \\ &= \frac{1}{N}\sum_{n=1}^{N}u_1^T(x_n - \bar{x})(x_n - \bar{x})^T u_1 \\ &= u_1^T\Sigma u_1.\end{aligned}$$

2.2. Maximisation Problem

These statistics now can be fed into a maximisation problem:

$$\max_{u_1}\ u_1^T\Sigma u_1$$

such that $\|u_1\|_2 = 1$.
Writing the Lagrangian results in in:

$$\mathcal{L} := u_1^T\Sigma u_1 + \lambda_1(1 - u_1^T u_1).$$

Setting $\frac{\delta}{\delta u_1}\mathcal{L} \stackrel{!}{=} 0$ results in:

$$\Sigma u_1 = \lambda_1 u_1$$

We observe that u_1 is an *eigenvector* of Σ and λ_1 it's associated *eigenvalue*. Furthermore λ_1 is also the variance of the projected data:

$$\lambda_1 = u_1^T\Sigma u_1$$

2.2.1. Second principal direction

The second principal direction can be obtained by maximising the variance $u_2^T \Sigma u_2$, subject to $\|u_2\|_2 = 1$ and $u_2^T u_1 = 0$:

$$\mathcal{L} = u_2^T \Sigma u_2 + \lambda_2 \left(1 - u_2^T u_2\right) + \nu \left(u_2^T u_1\right).$$

The maximum i found by setting $\frac{\delta \mathcal{L}}{\delta u_2} \stackrel{!}{=} 0$:

$$2\Sigma u_2 - 2\lambda_2 u_2 + \nu u_1 = 0.$$

Because of the orthogonality between u_2 and u_1 we observe that u_2 contains no component of u_1 and hence $\nu = 0$. We get:

$$\Sigma u_2 = \lambda_2 u_2.$$

We observe that u_2 is an eigenvector of Σ with the second largest eigenvalue of λ_2.

2.3. Solution: Eigenvalue Decomposition

Hence we see that the eigenvalue decomposition of the covariance matrix

$$\Sigma = U \Lambda U^T$$

contains all relevant information.
For a projection space of size $K \leq D$ we choose the K eigenvectors $\{u_1, \ldots, u_k\}$ with the largest associated eigenvalues $\{\lambda_1, \ldots, \lambda_2\}$.

2.4. Error Formulation

We define an *orthonormal* basis $\{u_d\}$, $d = 1, \ldots, D$ of $\mathbb{R}^D$. The scalar projection of x_n onto u_d (magnitude) is given by:

$$z_{n,d} = x_n^T u_d.$$

The associated projection onto u_d amounts to $z_{n,d} u_d$. Therefore, each data point can be represented in the basis by:

$$x_n = \sum_{d=1}^{D} z_{n,d} u_d = \sum_{d=1}^{D} \left(x_n^T u_d\right) u_d.$$

Restricted representation using $K < D$ basis vectors can be written as:

$$\tilde{x}_n = \sum_{d=1}^{K} a_{n,d} u_d + \sum_{d=K+1}^{D} b_d u_d,$$

where b_d does not depend on the data point x_n.

The approximation error can be represented by:

$$J(\{a_{n,d}\},\{b_d\}) = \frac{1}{N}\sum_{n=1}^{N}\|x_n - \tilde{x}_n\|_2^2$$

$$\text{Minimisation of } J \text{ w.r.t.} a_{n,d} = x_n^T$$

$$\text{Minimisation of } J \text{ w.r.t.} b_d = \bar{x}^T u_d$$

The displacement can be obtained by resubsittuing $a_{n,d}$ and b_d:

$$x_n - \tilde{x}_n = \sum_{d=K+1}^{D}\left\{(x_n - \bar{x})^T u_d\right\} u_d.$$

We observe that the displacement vector is orthogonal to the principal space! Resubstituting the displacement into the error criterion leads to:

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{d=K+1}^{D}\left(x_n^T u_d - \bar{x}^T u_d\right)^2 = \sum_{d=K+1}^{D} u_d^T \Sigma u_d$$

2.5. Matrix viewpoint

The data can be represented as matrix:

$$X = [x_1, \ldots, x_n, \ldots, x_N]$$

The corresponding zero-centered data is:

$$\bar{X} = X - M,$$

where $M = \underbrace{[\bar{x}, \ldots, \bar{x}]}_{N \text{ times}}$.
Compute the projection of $\bar{X}$ on $U_k = [u_1, \ldots, u_K]$ with:

$$\underbrace{\bar{Z}_K}_{K\times N} = \underbrace{U_K^T}_{K\times D} \cdot \underbrace{\bar{X}}_{D\times N}.$$

To approximate $\bar{X}$, we return to the original basis:

$$\tilde{\bar{X}} = U_K \cdot \bar{Z}_K.$$

For $K = D$ we obtain a perfect reconstruction.

2.6. Computation

First compute the *empirical mean*:

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

Then *center the data* by subtracting the mean from each sample:

$$\bar{X} = X - M,$$

where $M = \underbrace{[\bar{x}, \ldots, \bar{x}]}_{N \text{ times}}$. Now compute the *Covariance matrix*:

$$\Sigma = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T = \frac{1}{N} \underbrace{\bar{X}\bar{X}^T}_{\text{Scatter Matrix } \mathbf{S}}.$$

Σ is *symmetric*.
Now the *Eigenvalue decomposition* can be computed:

$$\Sigma = U \Lambda U^T,$$

where $\Lambda = diag[\lambda_1, \ldots, \lambda_D]$, such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D$ with orthonormal eigenvectors.

Transformation the data can be transformed on to the new basis of K dimensions:

$$\tilde{\bar{Z}} = U_K^T \bar{X},$$

$\bar{Z} \in \mathbb{R}^{K \times N}$: We obtain a dimension reduction of the data.

Reconstruction Go back to the original basis by computing

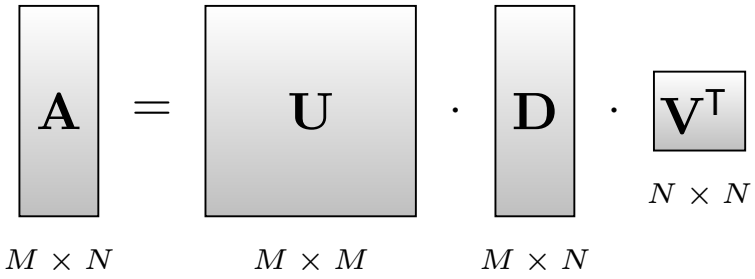
$$\tilde{\bar{X}} = U_K \bar{Z}$$
$$\tilde{X} = \tilde{\bar{X}} + M$$

3. Singular Value Decomposition

3.1. Introduction

The *Singular Value Decomposition* (SVD) is a widely used technique to decompose a matrix into several component matrices exposing many of the useful and interesting properties of the original matrix like rank, null-space, orthogonal basis of column and row space.
Every rectangular, real or complex matrix S has an SVD decomposition into a set of three matrix factors.

Let A be any real M by N matrix, $A \in \mathbb{R}^{M \times N}$, then A can be decomposed as $A = UDV^T$:



- U is an $M \times M$ orthogonal matrix, $U^T U = I$
- D is an $M \times N$ diagonal matrix
- V^T is an $N \times N$ orthogonal matrix, $V^T V = I$

3.2. Singular values

The elements of D are only non-zero on the diagonal and are called the *singular values*. By convention, the order of the singular vectors is determined by the *high-to-low* sorting of singular values, with the highest singular value in the upper left index of the D matrix. The first r columns of U are called *left singular vectors*, they form an orthogonal basis for the space spanned by the columns of the original matrix A.
Similarly the first r rows of V^T are the *right singular vectors*, they form an orthonormal basis for the row space of A.
SVD provides an explicit representation of the range and null-space of a matrix A.

- The right side singular vectors corresponding to vanishing singular values of A, span the null space of A:

$$d_i = 0 \quad \Longrightarrow \quad Av_i = 0 \quad \Longrightarrow \quad v_i \in Null(A).$$

- The left singular vectors corresponding to the non-zero singular values of A span the range of A.

As a consequence, the rank of A equals the number of non-zero singular values (= the number of non-zero elements in D).

$$Rank(A) = \#d_i > 0.$$

3.3. Closest Rank-k Matrix

Let the SVD of $A \in \mathbb{R}^{M \times N}$ be given by $A = UDV^T$. If $k < r = Rank(A)$ and

$$A_k = \sum_{i=1}^{k} d_i u_i v_i^T.$$

Then

$$\min_{Rank(B)=k} \|A - B\|_2 = \|A - A_k\|_2.$$

This means that A_k is the closest $Rank(k)$ approximation to A in the Eculidean matrix norm sense hence:

$$\|A - A_k\|_2 = d_{k+1}.$$

3.4. Properties

The columns of U are the eigenvectors of AA^T. This claim can be verified using the SVD decomposition:

$$AA^T = UDV^T VDU^T = UD^2U^T.$$

Similarly the rows of V^T (or columns of V) are the eigenvectors of A^TA as:

$$A^TA = VDU^T UDV^T = VD^2V^T.$$

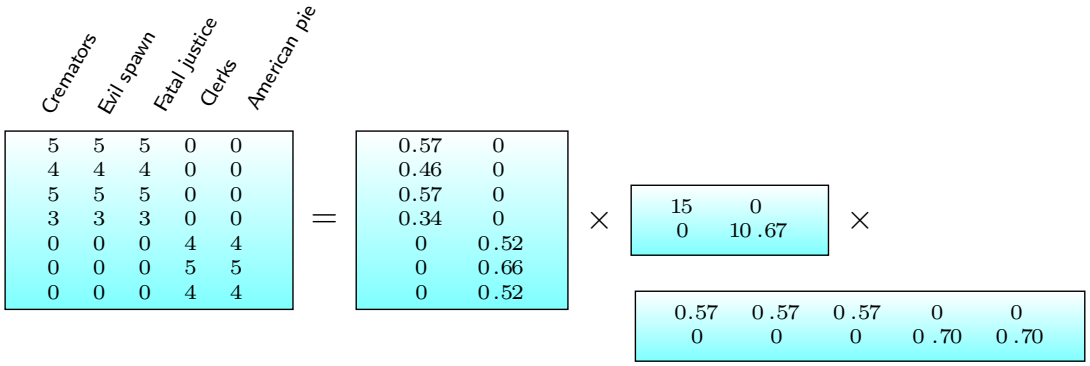
3.5. Movie Example

Let A be a list of users with their respective movie preferences. Then the SVD decomposition

$$A = UDV^T,$$

can be interpreted in the following way:

- **U**: Users-to-concept affinity matrix.
- **D**: Expression level of the different concepts in the data.
- **V**: Movies-to-concept similarity matrix.


Cremators
Evil spawn
Fatal justice
Clerks
American pie
5 5 5 0 0
4 4 4 0 0
5 5 5 0 0
3 3 3 0 0
0 0 0 4 4
0 0 0 5 5
0 0 0 4 4
=
0.57 0
0.46 0
0.57 0
0.34 0
0 0.52
0 0.66
0 0.52
×
15 0
0 10.67
×
0.57 0.57 0.57 0 0
0 0 0 0.70 0.70

Part II.
Clustering

4. Introduction

A set of datapoints in a d-dimensional Euclidean space is given.

Aim The aim is to find a *meaningful partition* of the data; i.e. label each data point with a unique value $\{1, \ldots, k\}$.

Objective The partition should group together similar data points, while the different groups/clusters should be as dissimilar as possible from each other.

This way we can uncover similarities between data points and give rise to data compression schemes.

4.1. Problem

Consider N data points in a D-dimensional space. Each data vector is denoted by x_n, $n = 1, \ldots, N$. Our goal is to partition the data set into K clusters: Find vectors $u_1, \ldots, u_K$ that represent the centroid of each cluster.
A datapoint x_n belongs to cluster k if the Euclidean distance between x_n and u_k is smaller than the distance to any any other centroid.
Mathematically, the clustering problem defines a mixed discrete continuous optimisation problem.

4.1.1. The Cost Function of Vector Quantisation

Objective Minimise the cost function

$$J(U, Z) = \|X - UZ\|_F^2 = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{k,n} \|x_n - u_k\|_2^2$$

where

$$\begin{aligned} X &= [x_1, \ldots, x_N] \in \mathbb{R}^{D \times N} \\ U &= [u_1, \ldots, u_K] \in \mathbb{R}^{D \times K}, && \text{centroids} \\ Z &\in \{0, 1\}^{K \times N}, && \text{assignments} \end{aligned}$$

with $\sum_k z_{k,n} = 1 \forall n$ i.e., one element per columns set to 1.

Assignment notation:

Assignment Notation : Vector $\hat{z} \in \{1, \ldots, K\}^N$ indicating for each data point to which cluster index it is assigned:

$$\hat{z} = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

Matrix Notation : The matrix $Z \in \{0, 1\}^{K \times N}$ with only one non-zero entry per column, assigns data points to clusters:

$$Z = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

5. K-Means

5.1. Overview

The algorithm alternates between two steps:

- *Assigning data points* to clusters Let $k^*(x_n)$ denote the cluster index with the minimal distance between a cluster centroid and the data point x_n:

$$k^*(x_n) = \arg\min_k \left\{ \|x_n - u_1\|_2^2, \ldots, \|x_n - u_k\|_2^2, \ldots, \|x_n - u_K\|_2^2 \right\}$$

- *Updating the cluster centroids* based on all the data points assigned to it. Compute the mean/centroid of a cluster that can be written as:

$$u_k = \frac{\sum_{n=1}^N z_{k,n} x_n}{\sum_{n=1}^N z_{k,n}} \quad \forall k, \quad k \in \{1, \ldots, K\}$$

5.2. Algorithm

1. Initiate with a random choice of $u_1^{(0)}, \ldots, u_K^{(0)}$ (or let $u_1^{(0)}, \ldots, u_K^{(0)}$ equal data points from the set). Set $t = 1$.

2. **Cluster assignment.** Solve $\forall n$:

$$k^*(x_n) = \arg\min_k \left\{ \left\| x_n - u_1^{(t)} \right\|_2^2, \ldots, \left\| x_n - u_K^{(t)} \right\|_2^2 \right\}.$$

   Then, $z_{k*(x_n),n}^{(t)} = 1$ and $z_{j,n}^{(t)} = 0 \; \forall j \neq k, \quad j = 1, \ldots, K.$

3. **Centroid update**

$$u_k^{(t)} = \frac{\sum_{n=1}^{N} z_{k,n}^{(t)} x_n}{\sum_{i=1}^{N} z_{k,n}^{(t)}} \; \forall k, \; k \in \{1, \ldots, K\}$$

4. Increment t. Repeat step 2 until $\left\| u_k^{(t)} - u_k^{(t-1)} \right\|_2^2 < \varepsilon \; \forall K \; (0 < \varepsilon << 1)$ or until $t = t_{\text{finish}}$.

Aspects:

- The computational cost of each iteration is $\mathcal{O}(KN)$.
- Convergence is guaranteed
- Optimises a *non-convex* objective. Hence only a local minimum can be guaranteed.
- Can be used to compress data: store only the centroids and the assignments of data point to clusters.

Problems:

- Non-convex objective, local minima and sensitive to initialisations.
- Not appropriate for non-Euclidean data $\mapsto$ need to use other distances.
- The optimal number of clusters K is unknown: One has to find a balance between total compression ($K = 1$) and no loss of information ($K = N$).

5.3. Stability

5.3.1. High-Level Stability test

The following is a high-level stability test for a given set of data points and a given number of clusters:

1. Generate perturbed versions of the set for example by adding noise or drawing sub-samples.
2. Apply the clustering algorithm on all versions.
3. Compute pair-wise distances between all clusterings (using some distance measure).
4. Compute the *instability* as the mean distance between all clusterings.

Repeat this for different numbers of clusters and choose the one that minimises the instability.

5.3.2. Distance between Clusterings

For two clusterings C and C' that are defined on the same data points we compute the distance between clusterings d in the following procedure:

1. Compute the distances between the two clusterings by counting points on which the two clusterings agree or disagree.
2. Repeat over all permutations of the cluster labels (since the same cluster might be sometimes labeled 1 and sometimes 2 etc...).
3. Choose the permutation with minimal distance and the corresponding distances is d.

In other words,

$$d = \min_{\pi} \left\| Z - \pi(Z') \right\|_0$$

where $\pi(Z')$ is one of the possible row permutations of Z' and $\|Z\|_0$ denotes the cardinality of Z. If two clusterings are defined on different data sets but many points overlap, we use only these for comparison, otherwise, a mapping from one domain to the other is required.

5.3.3. Calculation of Stability

The rate of inconsistent data items r is computed as follows"

1. Cluster data sets X, X' to infer assignments Z, Z'.
2. Train a classifier φ on (X, Z) to transfer the clustering results Z on X to X'.
3. Apply φ on X' and compare the optimally permuted output with Z':

$$r := \frac{1}{N} \min_{\pi \in \mathbb{S}_K} \left\{ \sum_{i=1}^{N} \mathbb{I}_{\{\pi(\varphi(x'_i)) \neq \hat{z}'_i\}} \right\}.$$

   The indicator function $\mathbb{I}_{\{p\}}$ is 1 if predicate p is true, and 0 otherwise.

   Minimisation of $\pi \in \mathbb{S}_K$ compensates for the permutation of the cluster numbers.

The higher the number of clusters, the more difficult it is to have a small rate r of inconsistent cluster assignments. Given K clusters of equal size, a random assignment yields

$$r_{rand} = \frac{K-1}{K}.$$

To be able to compare hypotheses with different K, relate r to r_{rand}. The *stability* is this defined as:

$$stab := 1 - \frac{r}{r_{rand}}.$$

- $stab = 1$: No inconsistent assignments
- $stab = 0$: Not better than a random assignment

6. Clustering as Matrix Factorisation

SVD is a class matrix factorisation technique according to which every matrix matrix can be decomposed into $X = UDV^T$. With $U \in \mathbb{R}^{D\times D}$, $D \in \mathbb{R}^{D\times N}$ and $V \in \mathbb{R}^{N\times N}$. By setting UD in the decomposition as U and renaming V^T to Z, we can write

$$X = UZ.$$

Approximating X using the K largest singular values we get a factorisation involving matrices of the same dimensionality as K-Means.

7. Mixture Models

Soft Clustering The term Soft Clustering is ambiguous since it can refer to the algorithm or to the model:

Algorithmic: Soft K-Means: Instead of assigning a point to exactly one cluster. Consider assigning a probability that a data point belongs to a certain cluster.

Model relaxation: The model can be relaxed by replacing the "hard" constraint given by

$$z_{k,n} \in \{0,1\}, \quad \sum_{k=1}^{K} z_{k,n} = 1,$$

and replace it by the *soft constraint*:

$$z_{k,n} \in [0,1], \quad \sum_{k=1}^{K} z_{k,n} = 1.$$

This relaxed model **cannot** be written in Matrix factorisation form.

7.1. Introduction

The mixture of K probability densities is defined as

$$p(x) = \sum_{k=1}^{K} \pi_K p(x|\theta_k).$$

Each probability distribution $p(x|\theta_k)$ is a *component* of the mixture and has its own parameters θ_k. Almost any continuous density can be approximated by using a sufficient number of component distributions. For a Gaussian component distribution the parameters θ_k are given by the mean μ_k and the covariance Σ_k.
Mixture models are constructed from:

- Component distributions of the form $p(x|\theta_k)$.

- Mixing coefficients π_k that give the probability of each component.

In order for $p(x)$ to be a proper distribution, we have to ensure that

$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{and} \quad \pi_k \geq 0,\ 1 \leq k \leq K.$$

Therefore, the parameters π_k, $1 \leq\leq K$ define a *categorical* distribution which represents the probability of each component distribution.

7.2. Gaussian Mixture Model

The Gaussian Mixture Model (GMM) uses Gaussians as the component distributions. We thus assume the following distribution for our data:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k).$$

Given data points $\{x_1, \ldots, x_N\}$, we then aim at learning parameters μ_k, Σ_k, such that we approximate the data as closely as possible. This is equivalent to finding the parameters that *maximise the likelihood* of the data.

7.2.1. Generative Viewpoint

Given the model parameters Σ, μ, π, sample the data x_n as follows:

1. Sample a cluster index k according to the probabilities π_k.
2. Sample a data point x_n from the distribution $p(x_n|\mu_k, \Sigma_k)$.

7.2.2. Parameter Estimation

We assume that the data points x_n are independent and identically distributed (i.i.d.). The probability or likelihood of the observed data X, given the parameters can thus be written as:

$$p(X|\pi, \mu, \Sigma) = \prod_{n=1}^{N} p(x_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k).$$

We aim at finding the parameters that maximise the likelihood of the data:

$$(\hat{\pi}, \hat{\mu}, \hat{\Sigma}) \in \arg\max_{\pi,\mu,\Sigma} p(X|\pi, \mu, \Sigma).$$

To simplify the expression one takes the logarithm, such that the product becomes a sum:

$$(\hat{\pi}, \hat{\mu}, \hat{\Sigma}) \in \arg\max_{\pi,\mu,\Sigma} \sum_{n=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\}.$$

Due to the presence of the summation over k inside the logarithm, the maximum likelihood solution for the parameters no longer has a closed-form analytic solution. In order to solve the equation we make use of an algorithmic approach: *Expectation Maximisation*.

7.2.3. Probability of assigning a data point to a cluster

We use the Bayes' rule to get:

$$\begin{aligned}\gamma(z_k) := p(z_k = 1|x) &= \frac{p(z_k = 1)p(x|z_k = 1)}{\sum_{j=1}^{k} p(z_j = 1)p(x|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)}.\end{aligned}$$

7.2.4. Expectation-Maximisation for Gaussian Mixture Models

A powerfule method for finding maximum likelihood solutions for models with latent variables. Setting the derivatives of $\log p(X|\pi, \mu, \Sigma)$ with respect to the means μ_k to zero, we obtain:

$$0 = \sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\Sigma_j \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}}_{\gamma(z_{k,n})} \Sigma_k^{-1}(x_n - \mu_k).$$

Assume that Σ_k is not singular. Multiplying by Σ_k we obtain:

$$\begin{aligned}\sigma_k &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{k,n}) x_n, \\ N_k &= \sum_{n=1}^{N} \gamma(z_{k,n}).\end{aligned}$$

The mean μ_k is obtained by taking a weighted mean of all the points in the data set. Maximising $\log p(X|\pi, \mu, \Sigma)$ with respect to the mixing coefficients π_k and taking account of the constraint which requires the mixing coefficients to sum to one, can be achieved using the following Lagrangian:

$$\log p(X|\pi, \mu, \Sigma) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right),$$

which gives

$$\begin{aligned}0 = \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\Sigma_j \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} + \lambda, \\ \implies \pi_k = \frac{N_k}{N}.\end{aligned}$$

Therefore, π_k is given by the average responsibility of the k-th component.

7.2.5. Algorithm

Given a Gaussian mixture model, the goal is to maximise the likelihood function with respect to the parameters.

1. Initialise the means μ_k, and mixing coefficients π_k. Set the Σ_k to the given covariances.

2. **E-step**. Evaluate the responsibilities using the current parameter values:

$$\gamma(z_{k,n}) = \frac{\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)}.$$

3. **M-step**. Re-estimate the parameters using the current responsibilities

$$\begin{aligned}
\mu_k^{new} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{k,n}) x_n, \\
\Sigma_k^{new} &= \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{k,n}) \cdot (x_i - \mu_k^{new})(x_i - \mu_k^{new})^T \qquad 1 \leq k \leq K, \\
\pi_k^{new} &= \frac{N_k}{N} \qquad \text{where } N_k = \sum_{n=1}^{N} \gamma(z_{k,n}).
\end{aligned}$$

4. Evaluate the log likelihood and check for convergence of either the parameters or the log likelihood.

7.2.6. Relation to K-means

The EM algorithm makes a soft assignment based on posterior probabilities[2]. Whereas the K-means algorithm performs a hard assignment of data points to clusters.
The K-means algorithm does not estimate the covariances of the clusters but only the cluster means. The EM algorithm can be reduced to K-means.
The EM algorithm takes more iterations to converge than K-means.
The K-means algorithm can be used to find a suitable initialisation for a Gaussian mixture model.
There will generally be multiple local maxima of the log likelihood function, and EM is not guaranteed to find the largest of these maxima.

7.2.7. Log Likelihood

$$\mathbb{E}_Z[\log p(X, Z|\mu, \Sigma, \pi)] \to -\frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{k,n} \|x_n - \mu_k\|_2^2 + \text{ const.}$$

[2]The posterior probability is the probability of the parameters θ given the evidence X : $p(\theta|X)$ (Wikipedia)

7.2.8. Matrix Factorisation

Assumption: $z_{k,n} \in [0,1]$ and $\sum_{k=1}^{K} z_{k,n} = 1 \; \forall n$.

$$\begin{aligned}
\|X - UZ\|_F^2 &= \sum_{n=1}^{N} \left\| x_n \sum_{k=1}^{K} z_{k,n} u_k \right\|_2^2 \\
&= \sum_{n=1}^{N} \left(\|x_n\|^2 - 2 \sum_{k=1}^{K} x_n u_k z_{k,n} + \left(\sum_{k=1}^{K} z_{k,n} u_k \right) \right) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{k,n} \|x_n - u_k\|_2^2 - \sum_{n=1}^{N} \sum_{k=1}^{K} z_{k,n} \left(u_k - \sum_{k'=1}^{K} z_{k',n} u_{k'} \right)^2
\end{aligned}$$

For identity covariance matrices GMM gives an upper bound!

8. Model Order Selection and Information Criteria

The selection of the number of clusters is a trade-off between two conflicting goals:

Data Fit: We want to predict the data well, e.g., maximises the likelihood. The likelihood usually increases by increasing the clusters.

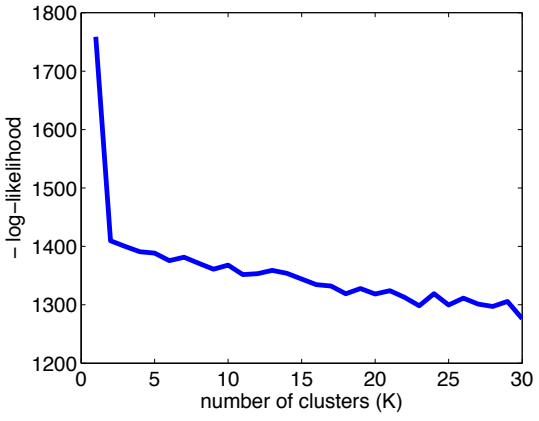
Complexity: Choose a model that abstracts a lot of data. This is often measured by the number of free parameters.

8.0.9. Negative Log-Likelihood

The smaller the negative log-likelihood, the better the fit.

$$-\log p(X|\pi, \mu, \Sigma) = -\sum_{n=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\}$$

In practice increasing K does not always decrease the negative log-likelihood due to local minima.



8.1. AIC and BIC

Achieve balance between data fit (measured by likelihood $p(X|.)$) and complexity. Complexity can be measured by the number of free parameters $\kappa(\cdot)$.

8.1.1. AIC - Akaike Information Criterion

$$AIC(U, Z|x_1, \ldots, x_N) = -\log(p(X|.) + \kappa(U, Z)$$

8.1.2. BIC - Bayesian Information Criterion

$$BIC(U, Z|x_1, \ldots, x_N) = -\log(p(X|.) + \frac{1}{2}\kappa(U, Z)\log N$$

Generally speaking, the BIC criterion penalises complexity more than the AIC criterion. A single AIC (BIC) result is meaningless. One has to repeat the analysis for different Ks and compare the differences: the most suitable number of clusters corresponds to the smallest AIC (BIC) value.

Example Mixture of Gaussians with fixed covariance. NUmber of free parameters:

$$\kappa(U, Z) = K \cdot D + (K - 1).$$

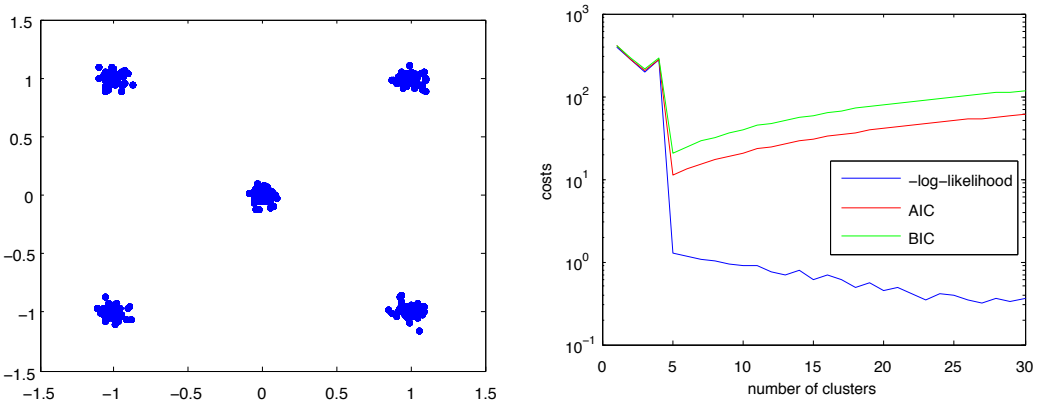


Figure 1: Comparison of AIC, BIC and negative Log-Likelihood an a synthetic dataset with 5 clusters.

9. Multi-Assignment Clustering

9.1. Role-Based Access Control (RBAC)

Given a *user-permission* matrix $X \in \mathbb{B}^{D \times N}$, find

Roles $U \in \mathbb{B}^{D \times K}$ and

Assignments $Z \in \mathbb{B}^{K \times N}$

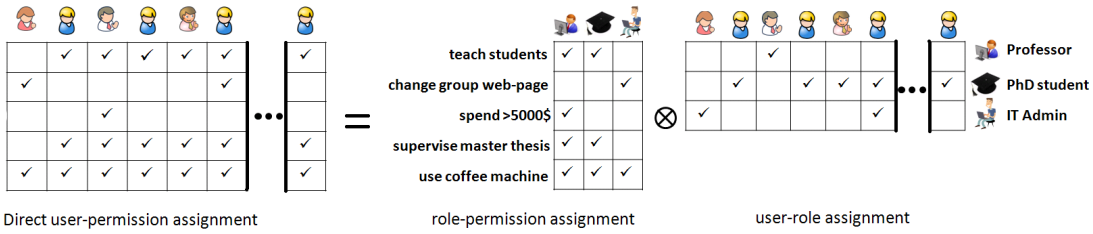
with $\mathbb{B} = \{0, 1\}$ such that

$$X = U \otimes Z \quad \Leftrightarrow \quad x_{dn} = \bigvee_k [u_{dk} \wedge z_{kn}].$$

- Each role defines a set of permissions
- Users are assigned to a set of roles and get all permissions of these roles.

9.1.1. Notation

- $x_{dn} \in \{0, 1\}$: Assignment of user n to permission d.
- $z_{kn} \in \{0, 1\}$: Assignment of user n to role k.
- $u_{dk} \in \{0, 1\}$: Assignment of permission d to role k.
- $\beta_{dk} \in [0, 1]$: Probability of $u_{dk} = 0$.



9.1.2. Evaluation Criteria

For each role mining problem definition, there is a (set of) evaluation criteria:

- Matrix Reconstruction
- Number of Sources
- Inference Quality
- Generalisation
- Stability

9.2. Binary Matrix Factorisation

Min-Noise Approximation: Given K, find the matrices $\hat{U}$, $\hat{Z}$ such that

$$(\hat{U}, \hat{Z}) = \arg\min_{U,Z} \|X - U \otimes Z\|_1$$

with $U \in \mathbb{B}^{D \times K}$ and $Z \in \mathbb{B}^{K \times N}$. This problem is also called *approximate Boolean Matrix decomposition* and has the following properties:

- **All** matrices are Boolean,
- It is a *combinatorial optimisation problem*,
- It is proven to be *NP-hard* (reducible to set basis problem).

In contrast to recommender systems a the outputs $\hat{U}$ and $\hat{Z}$ are always binary.

9.2.1. Rounded SVD

1. Compute the singular value decomposition

$$X = U \cdot S \cdot V^T$$

2. Discard columns $K+1, \ldots, D$ of U to get $U_{(K)}$.
   Discard rows $K+1, \ldots, N$ of V to get $V_{(K)}$.

3. Round the left and right singular vectors to get Boolean matrices $\hat{U}$ (roles) and $\hat{Z}$ (role assignments):

$$\hat{U} = (U_{(K)} > t_U)$$
$$\hat{Z} = (V_{(K)} > t_V)$$

   where t_U and t_V are thresholds.

The rounded continuous decomposition is a poor Boolean decomposition!

9.2.2. K-means

K-means partitions objects into disjoint groups (clusters) s.t. the average *distance* between data and corresponding cluster prototypes is minimal. The standard objective function for K-means uses the *Euclidean distance* measure:

$$J(U, Z) = \|X - UZ\|_2^2 = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{kn} \|x_n - u_k\|_2^2$$

and centroids are updated via mean operation.

In order to use K-means for a Boolean decomposition the distance is adapted to use *Hamming distance* (0-norm):

$$J(U, Z) = \|X - UZ\|_0 = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{kn} \|x_n - u_k\|_0$$

The centroids u_k are restricted to Boolean values and the centroid update step needs to be adapted:

$$u_{dk} = \text{median}(\{x_{dn} | z_{kn} = 1\}) \; \forall k \in \{1, \ldots, K\}, \; \forall d \in \{1, \ldots, D\}$$

K can be found with cross-validation. K-means only yields *disjoint clustering*, i.e. Z matrix has only a single 1 in each column. In RBAC however a user can have multiple role.

9.2.3. RoleMiner

RoleMiner is a heuristic for role mining. Idea: A set of common permissions could potentially be a role. Roles are created by finding common sets of permissions between user.
Comes in two variants: *CompleteMiner* and *FastMiner*.
RoleMiner is very sensitive to noise:

- If only a few individual bits are noisy, then the number of candidate roles gets much larger.
- The result is unstable: If the noise changes slightly, then the solution is completely different.

9.2.4. DBPsolver

DBPsolver approximately solves the *Discrete Basis Problem*.

Discrete Basis Problem: For a given Boolean matrix $X \in \mathcal{B}^{D \times N}$ and a number K of basis vectors, find a Boolean matrix $U \in \mathcal{B}^{D \times K}$ and a Boolean matrix $Z \in \mathcal{B}^{K \times N}$ minimising

$$\|X - U \otimes Z\|_F^2.$$

The discrete basis problem and min-noise Role Mining problem are identical.

9.2.5. Probabilistic Clustering

Difficulties so far

- Searching in the full Boolean spaces has a *too high complexity*.

- Restricting the Boolean search spaces *ignores solutions*
- Searching solutions in continuous space and rounding *produces poor results*.
- Search heuristics are prone to *overfitting*

Modeling of RBAC Computing a likelihood requires to design a probabilistic model $p(X|U, Z)$ for the generation process of the data X. This is the probability X given the model, the cluster assignments Z, and the cluster centroids U.

Derivation I: One Object For the simple case we consider one cluster per object:

$$\sum_k z_{kn} = 1, \qquad \forall n \in \{1, \ldots, N\}.$$

k_n is the cluster of object n:

$$k_n = \{k \in \{1, \ldots, L\} | z_{kn} = 1\}.$$

Consider on entry x_{dn}: WTF

$$p(x_{dn} = 0|\beta_{d\cdot}, z_{\cdot n}) = \beta_{dk_n}.$$

Therefore:

$$\begin{aligned} p(x_{dn} = 1|\beta_{d\cdot}, z_{\cdot n}) &= 1 - p(x_{dn} = 0|\beta_{d\cdot}, z_{\cdot n}) \\ &= 1 - \beta_{dk_n} \end{aligned}$$

Derivation II: All objects Objects are now independent given the parameters Z and U. Thus:

$$\begin{aligned} p(X|\beta, Z) &= \prod_{n=1}^{N} \prod_{d=1}^{D} p(x_{dn} = 1|\beta_{d\cdot}, z_{\cdot n})^{x_{dn}} \cdot p(x_{dn} = 0|\beta_{d\cdot}, z_{\cdot n})^{1-x_{dn}} \\ &= \prod_{n,d} (1 - \beta_{dk_n})^{x_{dn}} \beta_{dk_n}^{1-x_{dn}} \end{aligned}$$

9.2.6. Multi-Assignment Clustering

Compared to probabilistic clustering we don't restrict an object to one cluster.
One $x_{\cdot n}$ is generated by a *set of clusters* $\mathcal{L}_n := \{k|z_{kn} = 1\}$. The resulting Boolean features are generated by *disjunction* of the corresponding Boolean cluster centroids u_k, (logical OR).

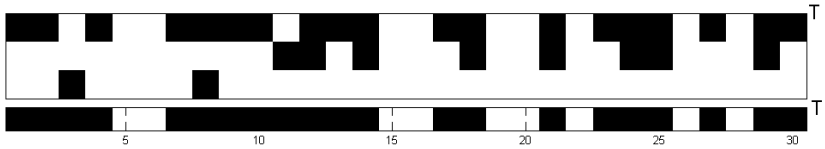


Figure 2: An object as the disjunction of the three clusters it belongs to.

Probabilistic view: An object being generated by two clusters k_1, k_2 has probability $\beta_{dk_1}\beta_{dk_2}$ to have a 0 at this dimension.
Generally, an object n belonging to the set of clusters $\mathcal{L}_n := \{k|z_{kn} = 1\}$ has a probability $\beta_{\mathcal{L}_n} := \prod_{k\in\mathcal{L}_n} \beta_{dk}$ for a 0 at dimension d.
In turn, it holds that $p(x_{dn} = 1|z, \beta) = 1 - \beta_{\mathcal{L}_n}$. Thus we get the following model:

$$p(X|\beta, Z) = \prod_{n,d} \underbrace{\left(1 - \prod_k \beta_{dk}^{z_{kn}}\right)^{x_{dn}}}_{\text{Noise component}} \underbrace{\left(\prod_k \beta_{dk}^{z_{kn}}\right)^{1-x_{dn}}}_{\text{Signal component}}$$

Not sure about the component naming

9.2.7. Noise model for RBAC

The deterministic RBAC generation of X:

$$X = U \otimes Z \quad \Leftrightarrow \quad x_{dn} = \bigvee_k [u_{dk} \wedge z_{kn}].$$

Since this generation rule is not able to explain erroneous assignments in X as noise we introduce a *mixture noise model*:

$$x_{dn} = (1 - \xi_{dn})(U \otimes Z)_{dn} + \xi_{dn}\eta_{dn}$$

where ξ_{dn} is a binary noise indicator and η_{dn} is a binary random variable.

Mixture Noise Model Noise generation:

- ξ_{dn} indicates whether a bit is generated by $p_N(\xi_{dn} = 1)$ or by $(U \otimes Z)_{dn}(\xi_{dn} = 1)$. ξ_{dn} is Bernoulli distributed:

  $$p(\xi_{dn}|\varepsilon) = \varepsilon^{\xi_{dn}}(1 - \varepsilon)^{1-\xi_{dn}},$$

  where ε is the probability to choose a random bit.

- If the bits is noisy ($\xi_{dn} = 1$), draw $\eta_{dn} = x_{dn}$ from

  $$p_N(x_{dn}|r) = r^{x_{dn}}(1 - r)^{1-x_{dn}},$$

  with r being the probability that a noisy bit is 1 ie. a user exceptionally gets a permission.

This can be modelled to a structure model:

$$p_S(X|\beta, Z) = \prod_{n,d} \left(1 - \prod_k (\beta_{dk})^{z_{kn}}\right)^{x_{dn}} \left(\prod_k (\beta_{dk})^{z_{kn}}\right)^{1-x_{dn}},$$

with $\beta_{dk} = p(u_{dk} = 0)$.
This then can be combined with the noise model:

$$p(X|Z,\beta,\xi,r) = \prod_{n,d} p_N(x_{dn}|r)^{\xi_{dn}} p_S(x_{dn}|\beta_{d\cdot}, z_{\cdot n})^{1-\xi_{dn}}$$

$$= \prod_{n,d} \underbrace{\left(r^{x_{dn}}(1-r)^{1-x_{dn}}\right)^{\xi_{dn}}}_{x_{dn} \text{ generated by noise}} \underbrace{\left(\left[1 - \prod_k (\beta_{dk})^{z_{kn}}\right]^{x_{dn}} \left[\prod_k (\beta_{dk})^{z_{kn}}\right]^{1-x_{dn}}\right)^{1-\xi_{dn}}}_{x_{dn} \text{ generated by roles}}.$$

The ξ_{dn} are unobservable (hidden) variables:

- They are unknown.
- They are too many to be estimated.

We thus integrate the ξ out of $p(X|Z,\beta,\varepsilon,\xi,r)$:

$$p(X|Z,\beta,\varepsilon,r) = \sum_{\{\xi\}} p(X,\xi|Z,\beta,\varepsilon,r),$$

where

$$\begin{aligned} p(X,\xi|Z,\beta,\varepsilon,r) &= p(X|Z,\beta,\xi,r)p(\xi|\varepsilon) \\ &= p(X|Z,\beta,\xi,r) \prod_{n,d} \varepsilon^{\xi_{dn}}(1-\varepsilon)^{1-\xi_{dn}} \\ &= \prod_{n,d} \left(\varepsilon r^{x_{dn}}(1-r)^{1-x_{dn}}\right)^{\xi_{dn}} \left((1-\varepsilon)(1-\beta_{d,\mathcal{L}_n})^{x_{dn}}(\beta_{d,\mathcal{L}_n})^{1-x_{dn}}\right)^{1-\xi_{dn}}. \end{aligned}$$

Mixture Model: The likelihood function can thus be written as:

$$\begin{aligned} p(X|Z,\beta,\varepsilon,r) &= \prod_{n,d} \left(\varepsilon r^{x_{dn}}(1-r)^{1-x_{dn}} + (1-\varepsilon)(1-\beta_{d,\mathcal{L}_n})^{x_{dn}}(\beta_{d,\mathcal{L}_n})^{1-x_{dn}}\right) \\ &= \prod_{n,d} \left(\varepsilon r + (1-\varepsilon)(1-\beta_{d,\mathcal{L}_n})\right)^{x_{dn}} \left(\varepsilon(1-r) + (1-\varepsilon)\beta_{d,\mathcal{L}_n}\right)^{1-x_{dn}} \end{aligned}$$

Parameters to estimate:

- Z: user-role assignments.
- β: probabilities of role-permission assignments U to be 0.
- ε: noise probability.
- r: probability of noisy bits to be 1.

This function requires a non-convex objective function to maximise the log-likelihood.

10. Non-Negative Matrix Factorisation

Problem:

- Given: Corpus of text documents such as web pages
- Goal: Find a low-dimensional representation of these documents.

Document Representation

Vocabulary Every semantically "useful"word in a language. This excludes all the stop words (the, is, at, which, etc...). Stemming reduces the text to it's root form: Cats, catlike, catty, etc. all map to the same word "cat".

D denotes the size of the vocabulary.

Document *Bag of words*-model: Ordering of the words in a document is ignored. The document is represented by a vector of length D with frequencies/counts of different words. This vector is usually very sparse.

10.1. Matrix view

Here N denotes the number of documents and K denotes the number of "clusters" with D being the vocabulary size.

- $X \in \mathbb{R}_+^{D \times N}$ denotes the document-term matrix which stores the word counts for each document:

  $$X = [x_1, \ldots, x_N]$$

  x_{dn}: Frequency of the d-th word in the n-th document.

- We study a non-negative matrix factorisation (NMF) of the document matrix X:

  $$X \approx UZ$$

  with $U \in \mathbb{R}_+^{D \times K}$ and $Z \in \mathbb{R}_+^{K \times N}$, with $\mathbb{R}_+ := [0, \ldots, \infty)$.

10.2. Methods

10.2.1. Full Singular Value Decomposition

A singular value decomposition can be used to decompose the document word matrix:

$$X = U\Sigma V^T.$$

However the procured U and V are not guaranteed to be non-negative.

10.2.2. Classic Latent Semantic Indexing (LSI)

The LSI method uses a partial SVD

$$\tilde{X} = U\tilde{\Sigma}V^T \approx X.$$

With $\tilde{\Sigma}$ having all but the largest K singular values set to zero.

Mapping function A query $x \mapsto (U\tilde{\Sigma})^T x$ can now be mapped by the query can now be compared/queried with an inner product:

$$\langle \overline{x}_1, \overline{x}_2 \rangle = \left(\left(U\tilde{\Sigma} \right)^T x_1 \right)^T \left(U\tilde{\Sigma} \right)^T x_2 = x_1 U\tilde{\Sigma}^2 U^T x_2.$$

10.2.3. Probabilistic Latent Semantic Indexing (pLSI)

Fixes some shortcomings of LSI:

- Probabilistic
- Takes non-negativity into account.

Generative model In order to generate a tuple $(document,\ word)$:

- Sample a document according to $P(document)$.
- Sample a word according to $P(word|document)$.
- Assume a factorisation:

$$P(word|document) = \sum_{topic} P(word|topic)P(topic|document)$$

Which assumes a conditional independence of word and document given the topic.

The joint distribution of a document and a word is therefore:

$$P(word,\ document) = P(word|document)P(document).$$

Matrix Factorisation View In a first step normalise the elements of X so that they can be interpreted as probabilities:

$$P(d-th\ word,\ n-th\ document) = \frac{x_{dn}}{\sum_{d',\ n'} x_{d'n'}}.$$

pLSI can be understood as a matrix factorisation of the form

$$X \approx UZ,$$

with $U \in R_+^{D \times K}$ and $Z \in R_+^{K \times N}$. Additionally we have the constraints:

$$\sum_{d=1}^{D} u_{dk} = 1\ \forall k \qquad \text{and} \qquad \sum_{k,n} z_{kn} = 1.$$

Parameter Estimation We want to maximise the likelihood of the data under the mode. Data: The normalised occurrence X with the model:

$$P(word,\ document) = \sum_{topic} P(word|topic)P(topic,\ document).$$

Due to the constraints on U and Z the pLSI model can be equivalently written as:

$$P(d-th\ word,\ n-th\ document) = x_{dn} = (UZ)_{dn}.$$

Kullback-Leibler Divergence between two discrete probability distributions $P(x)$ and $Q(x)$:

$$D_{KL}(P||Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right)$$

Properties:

$$\begin{aligned} &D_{KL}(P||Q) \geq 0. \\ &D_{KL}(P||Q) = 0 && \text{iff } P \text{ and } Q \text{ are identical.} \\ &D_{KL}(P||Q) \neq D_{KL}(Q||P)! \end{aligned}$$

The KL-divergence is not symmetric and therefore is not a metric/distance.

Kullback-Leibler Divergence applied for pLSI maximising the likelihood of the data under the model is the same as minimising the KL-divergence of X and UZ:

$$\begin{aligned} &\min_{U,Z} \sum_{d=1}^{D}\sum_{n=1}^{N} x_{dn}\log\left(\frac{x_d n}{(UZ)_{dn}}\right) \\ &\text{s.t. } \sum_{d=1}^{D} u_{dk} = 1 \forall k, \\ &\sum_{k,n} z_{kn} = 1 \\ &u_{dk} \geq 0,\ z_{kn} \geq 0. \end{aligned}$$

U, Z can be minimised by an EM algorithm similar to the one for the Gaussian Mixture Model.

10.2.4. Quadratic NMF

So far: pLSI as a specific NMF for document indexing. NMF is a more general concept though.

Consider a non-negative X and a quadratic cost function as in K-means:

$$\begin{aligned}\min_{U,Z} J(U,Z) &= \frac{1}{2}\|X - UZ\|_F^2 \\ \text{s.t.} u_{dk} &\in [0,\infty) \ \forall d,k \\ z_{kn} &\in [0,\infty) \ \forall k,n.\end{aligned}$$

Algorithm The NMF algorithm is similar to the K-means algorithm in that it alternates between two update steps:

$U \leftarrow rand(D,K)$
$Z \leftarrow rand(K,N)$
for $i = 1 : maxiter$ **do**
  Update Factors U: $u_{dk} \leftarrow u_{dk}\frac{(XZ^T)_{dk}}{(UZZ^T)_{dk}}$.
  Update coefficients Z: $z_{kn} \leftarrow z_{kn}\frac{(U^TX)_{kn}}{(U^TUZ)_{kn}}$.
end for

This leads to $X \approx UZ$ when $K < N$ or possibly $X = UZ$ for $N \leq K$.

10.2.5. Semi-NMF (for Quadratic Cost)

When we relax the non-negativity assumption on U this leads to the so called *Semi Non-Negative Matrix Factorisation*. The semi-NMF algorithm is similar to the K-means algorithm and to the NMF algorithm. It also alternates between two update steps:

Update U: $U = XZ^T(ZZ^T)^{-1}$.
Update Z: $z_{kn} \leftarrow z_{kn}\sqrt{\frac{(U^TX)^+_{kn} + [(U^TU)^-Z]_{kn}}{(U^TX)^-_{kn} + [(U^TU)^+Z]_{kn}}}$.

Here: $a^+_{ij} := \max(0, a_{ij})$ and $a^-_{ij} := \min(0, a_{ij})$.$u_{dk} \leftarrow u_dk\frac{(XZ^T)_{dk}}{(UZZ^T)_{dk}}$.

10.2.6. K-means Clustering Theorem

Z-orthoganl ($ZZ^T = I$) semi-NMF is equivalent to K-means clustering:

$$\begin{aligned}\min_{U,Z} \ J(U,Z) &= \frac{1}{2}\|X - UZ\|_F^2, \\ \text{s.t.} \ Z &\in \{0,1\}^{K\times N}, \\ \sum_k z_{kn} &= 1.\end{aligned}$$

is equivalent to:

$$\begin{aligned}\min_{U,Z} \ J(U,Z) &= \frac{1}{2}\|X - UZ\|_F^2, \\ \text{s.t.} \ ZZ^T &= I, Z \geq 0.\end{aligned}$$

If Z is not restricted to be orthogonal, semi-NMF is equivalent to a *soft* K-means clustering (not GMM!).

10.2.7. Convex-NMF

For interpretability reasons we can impose the constraint that U lies within the column space of X:

$$u_k = w_{1k}x_1 + \ldots + w_{nk}x_n, \qquad U = XW,$$

with $W \in \mathbb{R}_+^{N \times K}$.
So X is factorized in the following way:

$$X \approx UZ = XWZ.$$

U is a *convex combination* of input data. The convexity of the data combination *does not lead* to a convex optimisation problem. The algorithm for convex-NMF follows an analogous iterative update procedure.
Convex-NMF factors U are naturally *sparse*

Part III.
Sparse Coding

A signal and its representation are not the same thing. There are an infinite number of possible representation, each capturing different characteristics of the signal.
Natural signals have a *sparse representation* in a suitable dictionary due to regularity. Sparsity means that many coefficients vanish, e.g. have zero (or close to zero) magnitude.

11. Signal Compression

Given original signal x and given an orthogonal matrix A. Compute the linear transformation (change of basis) $z = Ax$. Truncate "small" values of z which yields an estimate $\hat{z}$. Compute the inverse transform $\hat{x} = A^T\hat{z}$.
The key idea being the *orthogonality* of A.

Compressions Algorithm

Given an original signal x and an *orthogonal matrix A*.

Compute the linear transformation (change of basis) $z = Ax$.

Since A is orthogonal, the transformed vector z has the same "energy" as the original signal x.

Truncate "small" values, giving $\hat{z}$.

We are interested in a *sparse signal*, that is we want z with a small number K of non-zeros.

We have compressed the signal since we only need to store K values.

The compression is lossy since we have discarded elements of z (by setting them to zero).

Compute the inverse transform $\hat{x} = A^T\hat{z}$.

Since $A^{-1} = A^T$ computing the inverse transform is efficient.

Decomposition and Reconstruction

Given a signal f and an orthonormal basis $\{u_1, \ldots, u_L\}$.

The coefficients representing signal f in the basis are given by

$$z_l = \langle f, u_l \rangle,$$

that is f can be reconstructed by

$$f = \sum_{l=1}^{L} z_l u_l = \sum_{l=1}^{L} \langle f, u_l \rangle u_l.$$

Setting certain coefficients z_l to zero is equivalent to not using certain basis functions u_l that is for a subset σ of size K,

$$\hat{f} = \sum_{k \in \sigma} z_k u_k.$$

The reconstruction error is given by

$$\left\| f - \hat{f} \right\|^2 = \sum_{k \notin \sigma} |\langle f, u_k \rangle|^2,$$

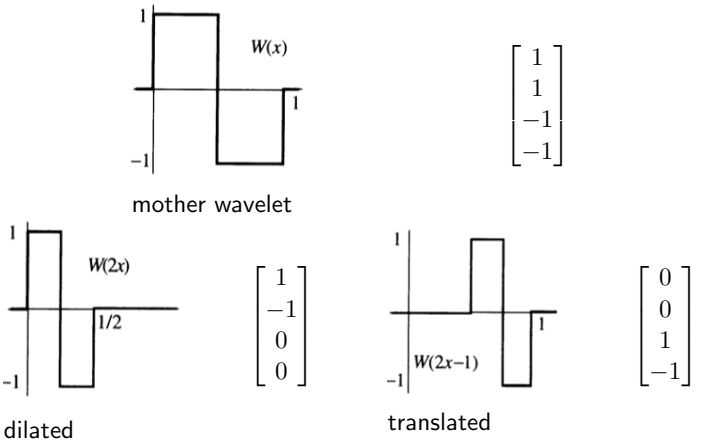
since $\langle u_k, u_l \rangle = 0$ for $k \neq l$.

11.1. Fourier Basis

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi} dx \qquad \forall x \in \mathbb{R}$$

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(x) e^{2\pi i \xi x} d\xi \qquad \forall x \in \mathbb{R}$$

11.2. Haar Wavelets



$$\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}$$

11.3. Fourier Basis vs Wavelet Basis

Fourier Basis

- Global support
- Good for "sine like" signals
- Poor for localised signals

Wavelet Basis

- Local support
- Good for localised signals
- Poor for non-vanishing signals.

12. Principal Component Analysis (PCA)

Given $X = [x_1, \ldots, x_N]$ a set of vectors we want to compress. Compute the (centered) covariance matrix:

$$\Sigma = \frac{1}{N}(X - M)(X - M)^T.$$

Compute the eigenvector decomposition:

$$[U\Lambda] = eig(\Sigma).$$

Note that since Σ is symmetric U is an orthonormal matrix. Choose K eigenvectors corresponding to the largest eigenvalues U_K.
For a given signal x the compressed coefficients are

$$\hat{z} = U_K x.$$

Also known as the Karhunen-Loeve transform or Hoteling transform.

12.0.1. Communication Cost

PCA Basis

The basis U_k is dependent on the data and is optimal for the given covariance.

Need to transmit both the basis U and the elements of $\hat{z}$.

Fixed Basis

Both parties (compressor and decompressor) agree upon a particular basis, e.g. Haar Wavelets.

Hence we only need to transmit the non-zero elements of the transformed signal $\hat{z}$.

12.1. Compressive Sensing

Assume that the original signal $x \in \mathbb{R}^D$ is sparse in some orthonormal basis U with K large coefficients in its sparse representation z:

$$x = Uz, \qquad \text{s.t. } |I_{z>>0}| = K.$$

The main idea of this method is to acquire the set y of M linear combinations of the initial signal instead of the signal itself and then reconstruct the initial signal from the measured one.

$$\begin{aligned} y_k &= \langle w_k, x \rangle, \ k = 1, \dots, M \\ y &= Wx = WUz =: \Theta z \qquad\qquad \text{with } \Theta = WU \in \mathbb{R}^{M \times D}. \end{aligned}$$

Surprisingly given any orthonormal basis U we can obtain a stable reconstruction for any K-sparse and compressible signal.
This is true under two conditions:

1. All elements $w_{i,j}$ of matrix W are i.i.d. random variables with a Gaussian distribution with zero mean and variance $\frac{1}{D}$.
2. $M : M \geq cK \log\left(\frac{D}{K}\right)$, where c is some constant.

To recover the initial signal $x \in \mathbb{R}^D$ from the measured signal $y \in \mathbb{R}^M$ we need to find a sparse representation z:

$$y = Wx = WUz = \Theta z, \qquad \Theta \in \mathbb{R}^{M \times D}.$$

Given z we can easily construct x:

$$x = Uz.$$

The problem of finding z appears to be *ill-posed* as $M << D$: Since there are many more unknowns than equations.
Generally this problem is NP hard. Using an approximation scheme like Matching Pursuit is recommended.

12.2. Sparse Coding

Coding via orthogonal transforms Given original signal x and orthogonal matrix A

- Compute linear transformation

$$z = Ax$$

- Truncate "small" values, giving $\hat{z}$.
- Compute inverse transform (recall $A^{-1} = A^T$)

$$\hat{x} = A^T \hat{z}$$

Measuring Performance Sparsity and low reconstruction error.

- Measure the reconstruction error

$$\|x - \hat{x}\|.$$

- Measure the sparsity of the coding vector z that is $nnz(z)$.

Dictionary Choice General considereations

- Fourier dictionary is good for "sine like" signals.
- Wavelet dictionary is good for localised signals.

Compressive Sensing

- Gather and store already compressed signal.
- Use l_0-norm minimisation to recover initial signal.

13. Overcomplete Dictionaries

13.1. Introduction

Overcompleteness ($L > D$): More atoms (dictionary elements) than dimensions.

13.2. Morphology of Signals

Different dictionaries are appropriate for different kinds of signals.

Dictionary selection Strategy Either choose the appropriate dictionary manually or try several and choose the one which affords sparsest coding.

13.2.1. Unions of Bases

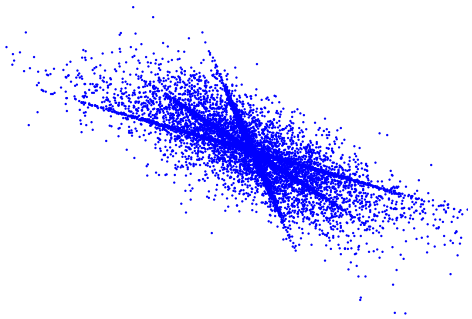
Union of orthonormal bases $[U_1 \dots U_B] \in \mathbb{R}^{D \times (B \cdot D)}$

$$\mathbf{x} = \underbrace{\mathbf{U}_1 \dots \mathbf{U}_B}_{D \times (B \cdot D)} \cdot \mathbf{z}$$

Each basis U_b is responsible for one characteristic of the signal, and the total number of atoms is $K = B \cdot D$.

13.3. General Overcomplete Dictionaries

Consider the data set $\{x_1, \ldots, x_{10000}\} \in \mathbb{R}^3$:



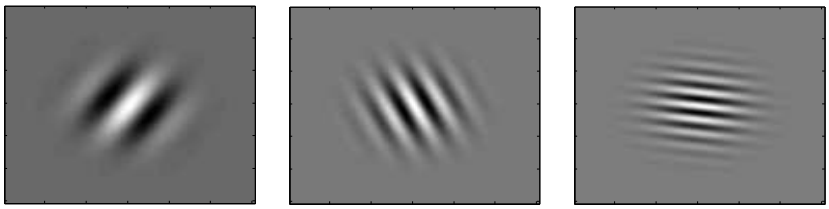
- Full coding ($K = 3$) in spanning basis $U \in \mathbb{R}^{3\times 3}$.
- $K = 2$ coding possible using a four atom dictionary:

  $$\tilde{U} = [u_1 u_2 u_3 u_4] \in \mathbb{R}^{3\times 4}$$

  aligned with densely populated subspaces.
- $L > D$ atoms are no longer linearly dependent.

Example: Directional Gabor Wavelets Directional oscillation, amplitude modulated by Gaussian window:

$$g(n_1, n_2; \mu_1, \mu_2, f\theta) \propto \exp\left[-(n_1 - \mu_1)^2\right] \exp\left[-(n_2 - \mu_2)^2\right] \times \Re\left\{\exp\left[i \cdot f \cdot (n_1 \cos\theta + n_2 \sin\theta)\right]\right\}$$



$(0, 0, 5, 1)$ $(0, 0, 10, 2)$ $(0, 0, 15, 3)$

13.4. Coherence

Increasing the *over-completeness factor* $\frac{L}{D}$:

- Potentially increases the sparsity of the coding
- Increases the linear dependency betwee atoms.

Linear dependency measure for dictionaries: *coherence*

$$m(U) = \max_{i,j:\ i \neq j} \left| u_i^T u_j \right|$$

$m(B) = 0$ for an orthogonal basis B.

$m([Bu]) \geq \frac{1}{\sqrt{D}}$ if atom u is added to B.

13.5. Signal Coding

U is *orthonormal*:

Matrix multiplication $z = U^T x$

U is a *spanning basis* (D linearly dependent atoms):

- $z = U^{-1}$
- inverting U can be ill-conditioned.

$U \in \mathbb{R}^{D \times L}$ is *overcomplete*:

- *Ill-posed* problem: more unknowns than equations
- Have to add constraint: Find sparsest z such that the equation holds:

$$z^* = \arg\min_z \|z\|_0$$
$$\text{s.t.} x = Uz,$$

where $\|z\|_0$ counts the number of non-zero elements in z.

$\|\cdot\|_0$ is piece-wise constant.

Finding the sparsest solution z^* is NP hard (combinatorial problem).

Brute-force approach: Try all possible atom subsets with $|\Delta| \leq D$.

Needs an approximation scheme for realistic problem instances.

13.6. Noise Observations

13.6.1. Additive Noise

$$x = Uz + n$$

Assumes each dimension is independently corrupted by zero-mean Gaussian nois:

$$n_d \sim \mathcal{N}(0, \sigma^2)$$

with variance σ^2.

Solve by maximising the sparsity of z, while the *residual* (approximation error) remains below σ:

$$z^* = \arg\min_z \|z\|_0$$
$$\text{s.t.} \qquad \|x - Uz\|_2 < \sigma$$

or alternatively:
Minimise the residual while selecting less than K atoms from the dictionary:

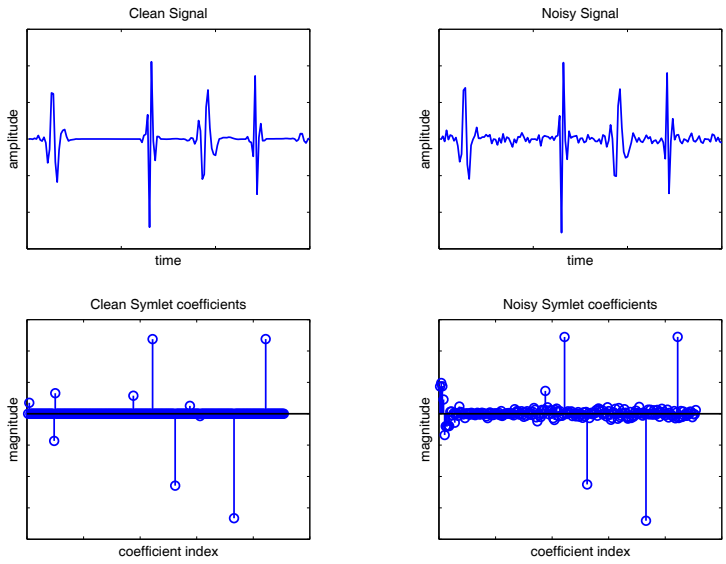
$$z^* = \arg\min_z \|x - Uz\|_2$$
$$\text{s.t.} \|z\|_0 \leq K$$

13.6.2. Approximate Sparse Coding

Explain the signal accurately with few atoms:

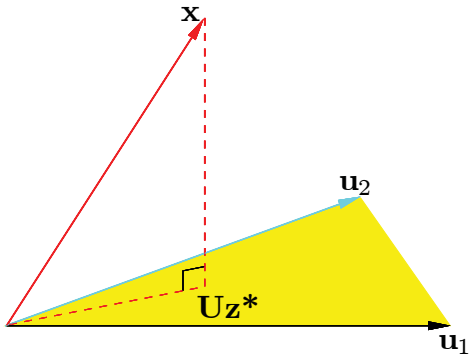
$$x = Uz + n = Uz + Uy = U(z + y),$$

where y is the coding of n in U:



We observe that *noise cannot be sparsely coded* in any dictionary, therefore y has many small coefficients. Large coefficients are still due to z.

Geometry of the sparse coding solution z^*:



The orthogonal projection of x onto the subspace spanned by the selected atoms $\{u_d | z_d^* \neq 0\}$ minimises $\|x - Uz\|_2$.

14. Matching Pursuit

Matching pursuit is a *Greedy* algorithm: Thus approximates a NP hard problem iteratively each time taking the action which is short-term optimal.

Application to sparse coding:

1. Start with the zero vector $z = 0$ and residual $r^0 = x$.
2. At each iteration t, take a step in the direction of the atom $u_{d^*(t)}$ that maximally reduces the residual $\|x - Uz\|_2$.

14.1. Minimising the Residual

Atom selection at iteration t:

$$d^*(t) = \arg\max_d |\langle r^t, u_d \rangle|$$

Proof for the first iteration:

1. Project $r^0 - x$ on atom u_d, to get
$$x = \langle x, u_d \rangle u_d + r^1$$
2. Since r^1 is orthogonal to u_d, and $u_d^T u_d = 1$,
$$\|x\|_2^2 = |\langle x, u_d \rangle|^2 + \|r^1\|_2^2.$$
3. Therefore, $\|r^1\|_2^2$ is minimised by maximising $|\langle r^0, u_d \rangle|^2$.

14.2. Matching Pursuit Algorithm

Objective

$$z^* = \arg\min_z \|x - Uz\|_2,$$
$$\text{s.t} \quad \|z\|_0 \leq K.$$

Algorithm

$z \leftarrow 0$
$r \leftarrow x$
while $\|z\|_0 < K$ **do**
Select atom with maximum absolute correlation to residual:

$$d^* \leftarrow \arg\max_d \left| u_d^T r \right|$$

Update coefficient vector and residual:

$$z_{d^*} \leftarrow z_{d^*} + u_{d^*}^T r$$
$$r \leftarrow r - \left(u_{d^*}^R r\right) u_{d^*}.$$

end while

14.3. Discussion

Assume $x = Uz$ has a K sparse coding

$$x = \sum_{d \in \Delta} z_d u_d.$$

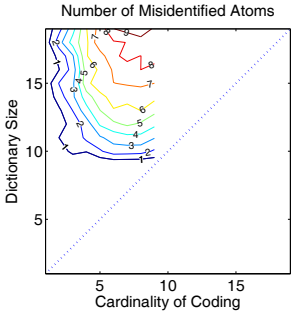
Under which condition is MP successful in recovering the true *support*

$$\Delta_{MP} = \Delta,$$

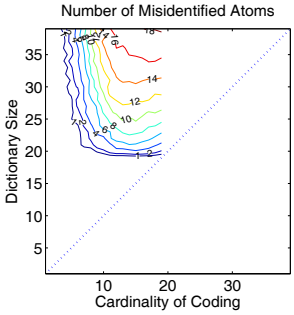
where $\Delta_{MP} = \{d | z_{d^*} \neq 0\}$ is the set of coding atoms for the MP solution z^*? Ie. when is the greedy approximation exact?

1. Generate dictionary with atoms uniformly sampled on hypersphere.
2. Generate observations with K sparse coding.

Compare Δ_{MO} with Δ:



$D = 10$



$D = 20$

We observe that the recovery is possible even for $L > D$ if the true coding is *sparse enough*.

Exact recovery condition

$$K < \frac{1}{2}\left(1 + \frac{1}{m(U)}\right)$$

guarantees support recovery by matching pursuit.

Intuition: If the coherence is small, explaining a generating atom with other atoms is not sparse. Therefore sparse coding recovers support.

Trade-Off: Increasing L

- Leads to sparser coding (smaller possible K),
- But it increases coherence $m(U)$.

14.4. Inpainting

Idea

1. Sparse coding of known parts of the image,
2. Predicting missing parts by reconstruction from sparse code.

Algorithm

1. Define a diagonal masking matrix M, $m_{d,d} = 1$ if pixel d is known, $m_{d,d} =$ if pixel d is missing.
2. Sparse coding of known parts in over-complete dictionary U:
$$z^* = \min_z \|z\|_0$$
$$\text{s.t.} \quad \|M(x - Uz)\|_2 < \sigma.$$
3. Image reconstruction using mask:
$$\hat{x} = Mx + (I - M)Uz^*.$$

Iterative EM Algorithm

1. Assume initial sparse coding z^0 of image.
2. Reconstruct image using mask

$$x^1 = Mx + (I - M)Uz^0.$$

3. Sparse coding of reconstructed image x^1:

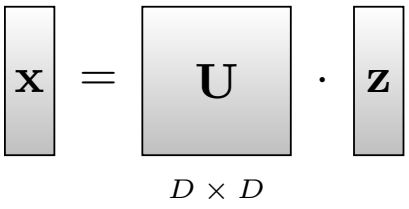
$$\begin{aligned} z^1 &= \min_z \|z\|_0 \\ \text{s.t.} \quad &\left\|x^1 - Uz\right\|_2 < \sigma. \end{aligned}$$

4. Iterate steps (2.) and (3.) t times until convergence.
5. Image reconstruction using mask:

$$\hat{x} = Mx + (I - M)x^t.$$

15. Dictionary Learning

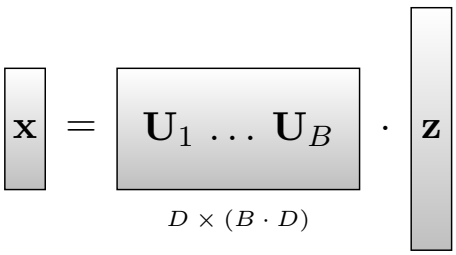
Fixed Orthonormal Basis



Advantage: Efficient coding by matrix multiplication $z = U^T x$.

Disadvantage: Only sparse for a single class of signals.

Fixed Overcomplete Basis



Advantage: Sparse coding for several signal classes.

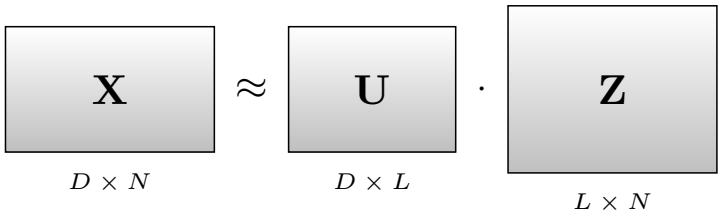
Disadvantage: Finding the sparsest code

- Requires an approximation algorithm,
- Is problematic if the dictionary size is L and the coherence $m(U)$ is large.

Learning the Dictionary

Advantage: We adapt the dictionary to signal characteristics and thus get the same approximation error achievable with a smaller L.

Disadvantage: We have to solve a matrix factorisation problem.



subject to sparsity constraint on Z and atom norm constraint on U.

15.0.1. Matrix Factorisation

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

We observe that the objective function is *not convex* in both U and Z (local minima). But convex in either U or Z (unique minimum).

Iterative Greedy Minimisation

1. Initialisation

   The minimisation is sensitive to the choice of U^0: The initial candidate solution is optimised locally and greedily until no progress is possible.

   Random atoms:

   Sample $\{u_l^0\}$ on the unit sphere:

   a) Sample D dimensional standard normal distribution:

   $$u_l^0 \sim \mathcal{N}(0, ID).$$

   b) Scale to unit length:

   $$u_l^0 \leftarrow \frac{u_l^0}{\|u_l^0\|_2}.$$

Samples from X:

a) $u_l^0 \leftarrow x_n$, where $n \sim \mathcal{U}(1, N)$ is sampled uniformly.

b) Scale to unit length:

$$u_l^0 \leftarrow \frac{u_l^0}{\left\|u_l^0\right\|_2}.$$

Use *fixed overcomplete dictionary*, e.g. overcomplte DCT.

2. Coding step:

$$Z^{t+1} \in \arg\min_Z \left\|X - U^t \cdot Z\right\|_F^2,$$

subject to Z being sparse and fixed U.

Since each column is separable we get:

$$\|R\|_F^2 = \sum_{i,j} r_{i,j}^2 = \sum_j \|r_j\|_2^2.$$

Thus we obtain N *independent* sparse coding steps:

$$z_n^{t+1} \in \arg\min_z \|z\|_0, \text{s.t.} \qquad \|x_n\|$$

3. Dictionary update step:

$$U^{t+1} \in \arg\min_U \|X - U \cdot Z\|_F^2,$$

subject to $\|u_l\|_2 - 1 \; \forall l \in [1, L]$ and fixed Z.

In contrast to the coding step the residual is *not separable* in atoms (columns of U).

Approximation: Update on atom at a time. $\forall l \in [1, L]$:

a) Set $U = [u_1 \dots u_l \dots u_L^t]$ fix for all atoms except u_l.

b) Isolate R_l^t that minimises R_l^t subject to $\|u_l^*\|_2 = 1$.

$$\begin{aligned}
&\left\|X - [u_1^t \dots u_l \dots u_L^t] \cdot Z^{t+1}\right\|_F^2 \\
=&\left\|X - \left(\sum_{e \neq l} u_e^t (z_e^{t+1})^T + u_l (z_l^{t+1})^T\right)\right\|_F^2 \\
=&\left\|R_l^t - u_l (z_l^{t+1})^T\right\|_F^2,
\end{aligned}$$

where z_l^T is the l-th row of matrix Z.

c) Find u_l^* that minimises R_l^t, subject to $\|u_l^*\|_2 = 1$.

We observe that $u_l(z_l^{t+1})^T$ is an outer product, i.e. a matrix. Hence we need to minimise the residual

$$\left\|R_l^t - u_l(z_l^{t+1})^T\right\|_F^2$$

by approximating R_l^t with a rank-1 $u_l(z_l^{t+1})^T$.

This is achieved by SVD of R_l^t:

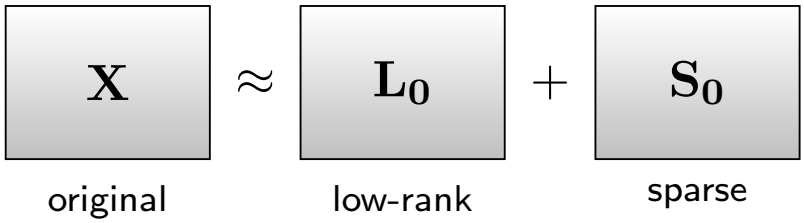
$$R_l^t = \tilde{U}S\tilde{V}^T = \sum_i s_i \tilde{u}_i \tilde{v}_i^T,$$

$u_l^* = \tilde{u}_1$ being the first left-singular vector.

We also observe that $\|u_l^*\| = 1$ is naturally satisfied.

16. Robust Principal Component Analysis

Goal: Find a low rank representation of a matrix X, which is corrupted by a sparse perturbation or sparse structured noise.



In contrast to the topic we have discussed so far we have compute a *additive decomposition*.

16.1. Additive Decomposition Problem

$$\begin{aligned} &\text{minimise } rank(L) + \lambda \cdot card(S) \\ &\text{subject to } L + S = X, \end{aligned}$$

where $card(\cdot)$ counts the number of non-zero entries.
This optimisation problem formalises the notion of separating a matrix into a low-rank and a sparse one. However, this decomposition is difficult to compute in general. For instance, it is not a convex optimisation problem.

16.2. Principal Component Pursuit (PCP)

$$\text{minimise } \|L\|_* + \lambda \|S\|_1, \text{subject to} \qquad L + S = X,$$

where $\|\cdot\|$ denotes the nuclear norm:

$$\sum_{i=1}^{\min\{m,n\}} \sigma_i,$$

where σ_i denotes the i'th singular value of a $n \times m$ matrix and $\|\cdot\|_1$ denotes the sum of absolute values of all matrix entries.
Note that this is not the same problem as minimising $rank(L) + \lambda \cdot card(S)$. The $L1$-Norm is only a convex relaxation of cardinality and the nuclear norm a convex relaxation of the rank.
But this relaxation will allow us to efficiently solve the problem, and if we are fortunate (i.e. under some conditions on the sparsity and low-rankness of our matrices) we recover the original matrices. We will see that this coincidence of solutions happens under surprisingly broad conditions.
In contrast to the additive decomposition problem, *PCP* is *convex*.

16.3. Lagrange Mutlipliers

16.3.1. Primal Optimisation Problem

$$\begin{aligned} &\text{minimise } f(x) && \\ &\text{subject to } g_i(x) \leq 0 && i \in [1, m] \\ &\qquad\quad h_i(x) = 0 && i \in [1, p] \end{aligned}$$

16.3.2. Unconstrained Problem

$$\text{minimise } f(x) + \sum_{i=1}^{m} I_-(g_i(x)) + \sum_{i=1}^{p} I_0(h_i(x))$$

$$I_-(u) = \begin{cases} 0 & u \leq 0 \\ \infty & u > 0 \end{cases}$$

$$I_0(u) = \begin{cases} 0 & u = 0 \\ \infty & u \neq 0 \end{cases}$$

I_0 and I_- penalise perturbations with violating constraints by "brick wall" penalty functions.
We can approximate $I_-(u)$ linearly with $\lambda_i u, \lambda_i \geq 0$ and $I_0(u)$ with $\nu_i u$:

$$\begin{aligned} I_-(u) &\approx \lambda_i u && \lambda_i \geq 0 \\ I_0(u) &\approx \nu_i u, && \end{aligned}$$

where λ_i and ν_i are called Lagrange multipliers.

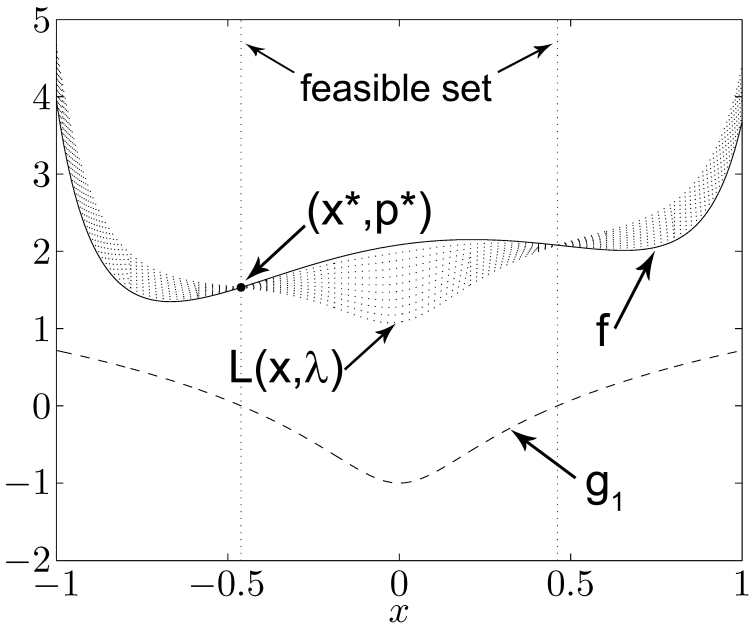
Lagrangian

$$L(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} \nu_i h_i(x).$$

Lagrange dual function

$$d(\lambda, \nu) = \inf_x L(x, \lambda, \nu).$$

Since $\lambda_i u \leq I_-(u)$ and $\nu_i u \leq I_0(u)$ for all u: The dual function is a lower bound on the optimal value of the primal problem p^*.



16.4. Dual Problem

Lagrangian

$$L(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} \nu_i h_i(x).$$

Lagrangian Dual Function

$$d(\lambda, \nu) = \inf_x L(x, \lambda, \nu).$$

Now find the best lower bound on p^*:

Lagrange dual problem

$$\begin{aligned} &\text{maximise } d(\lambda, \nu) \\ &\text{subject to } \lambda \geq 0 \end{aligned}$$

Strong Duality If the primal optimisation problem is convex, under some conditions, the solution to the dual problem is equivalent to the solution of the primal problem. It is always a lower bound on the solution of the primal problem.
In order to compute the additive decomposition for RPCA we will introduce a method called *Alternating Direction Method of Multipliers*. This method builds upon two other methods, whose positive features it combines: Dual decomposition and method of multipliers.

16.5. Convex Optimisation with Equality Constraints

$$\begin{aligned} &\text{minimise } f(x) \\ &\text{subject to } Ax = b. \end{aligned}$$

Lagrangian: $L(x, \nu) = f(x) + \nu^T(Ax - b)$

Dual function: $d(\nu) = \inf_x L(x, \nu)$

Dual problem: maximise $d(\nu)$

Recover optimal x : $x^* \in \arg\min_x L(x, \nu^*)$

Gradient Method for Dual Problem

$$\begin{aligned} \nu^{k+1} &= \nu^k + \alpha^k \nabla d(\nu^k) \\ \nabla d(\nu^k) &= A\tilde{x} - b \\ \tilde{x} &= \arg\min_x L(x, \nu^k), \end{aligned}$$

- ∇d: Gradient of the dual function.
- α^k: Step length at step k.

Dual Decomposition Goal: Parallelise individual optimisation steps on a cluster.

Suppose $f(x)$ is separable

$$f(x) = f_1(x_1) + \ldots + f_N(x_N) \qquad\qquad x = (x_1, \ldots, x_N)$$

Now L is separable

$$\begin{aligned} L(x, \nu) &= L_1(x_1, \nu) + \ldots + L_N(x_n, \nu) - \nu^T b \\ L_i(x_i, \nu) &= f_i(x_i) + \nu^T A_i x_i. \end{aligned}$$

Split x-minimisation step and we get the dual decomposition:

$$\begin{aligned} x_i^{k+1} &:= \arg\min_{x_i} L_i(x_i, \nu^k) \qquad\qquad i \in [1, N] \\ \nu^{k+1} &:= \nu^k + \alpha^k \left(A_i x_i^{k+1} - b \right) \end{aligned}$$

Parallelisable Algorithm:

init ν^1 and x^1
for $k = 2, \ldots, K$ **do**
  for $i = 1, \ldots, N$ **do**

// this can be parallelised
$x_i^{k+1} := \arg\min_{x_i} L_i(x_i, \nu^k)\ i = 1, \dots N$
$\Theta^{k+1} = A_i x_i$
end for
$\nu^{k+1} := \nu^k + \alpha^k \left(\sum_{i=i}^{N} \Theta^{k+1} - b \right)$
end for
return x^{k+1}

i missing in Θ?

16.6. Method of Multipliers

To overcome some problems of Dual ascent, namely convergence only under strict convexity and finiteness of f we introduce:

Augmented Lagrangian

$$L_\rho(x, \nu) = f(x) + \nu^T (Ax - b) + \frac{\rho}{2} \|Ax - b\|_2^2,$$

for any feasible x : $L_\rho(x, \nu) = L(x, \nu)$.

Method of Multipliers

$$x^{k+1} := \arg\min_x L_\rho(x, \nu^k)$$
$$\nu^{k+1} = \nu^k + \rho(Ax^{k+1} - b)$$

This adds more penalty to for violating the constraints and converges under far more general conditions than dual ascent.

16.6.1. Dual Update Step ρ

Why choose ρ as step size?

Optimality Conditions Primal and dual feasibility

$$Ax^* - b = 0$$
$$\nabla f(x^*) + A^T \nu^* = 0.$$

By definition x^{k+1} minimises $L_\rho(x, \nu^k)$, so

$$\begin{aligned} 0 &= \nabla_x L_\rho(x^{k+1}, \nu^k) \\ &= \nabla_x f(x^{k+1}) + A^T(\nu^k + \rho(Ax^{k+1} - b)) \\ &= \nabla_x f(x^{k+1}) + A^T \nu^{k+1} \end{aligned}$$

Therefore using ρ as step size the iterate (x^{k+1}, ν^{k+1}) is dual feasible. The primal residual $Ax^{k+1} - b$ converges to zero as the method of multipliers proceeds.

16.7. Alternating Directions Method of Multipliers (ADMM)

The main disadvantage of the method of multipliers is that the augmented Laplacian is not separable anymore. Hence we can not parallelise the x-minimisation step.
To get the superior convergence properties of the method of multipliers and the decomposability of dual ascent, we finally introduce the *Alternating Direction Method of Multipliers* (ADMM). The method splits the variable x into two parts, with the objective function separable over this splitting. The algorithm then separately minimises the two primal variables x and z.

$$\begin{aligned} &\text{minimise } f(x) + p(x) && f,\ p \text{ convex} \\ &\text{subject to } Ax + Bz = c. \end{aligned}$$

Augmented Lagrangian

$$L_\rho(x, z, \nu) = f(x) + p(z) + \nu^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2.$$

ADMM

$$\begin{aligned} x^{k+1} &:= \arg\min_x L_\rho(x, z^k, \nu^k) \\ z^{k+1} &:= \arg\min_z L_\rho(x^{k+1}, z, \nu^k) \\ \nu^{k+1} &:= \nu^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

16.7.1. Optimality Conditions

Primal Feasibility

$$Ax^* + Bz^* - c = 0.$$

Dual Feasibility

$$\begin{aligned} \nabla f(x^*) + A^T\nu^* &= 0 \\ \nabla p(z^*) + B^T\nu^* &= 0 \end{aligned}$$

These primal and dual feasibility conditions are the *necessary* and *sufficient* optimality conditions for the ADMM problem. It can be shown that the residual of the other two feasibility conditions converge to zero as ADMM proceeds.

We will show that z^{k+1}, one of the two primal variables, and ν^{k+1}, the dual variable always satisfy the second dual feasibility condition.

$$\nabla_z L_\rho(x, z, \nu) = \nabla_z p(z) + B^T\nu + \rho B^T(Ax + Bz - c)$$

Since z^{k+1} minimises $L_\rho(x^{k+1}, z, \nu^k)$ by definition we have that

$$\begin{aligned} 0 &= \nabla_z p(z^{k+1}) + B^T \nu^k + \rho B^T (Ax^{k+1} + Bz^{k+1} - c) \\ &= \nabla_z p(z^{k+1}) + B^T (\nu^k + \rho(Ax^{k+1} + Bz^{k+1} - c)) \\ &= \nabla_z p(z^{k+1}) + B^T \nu^{k+1} \end{aligned}$$

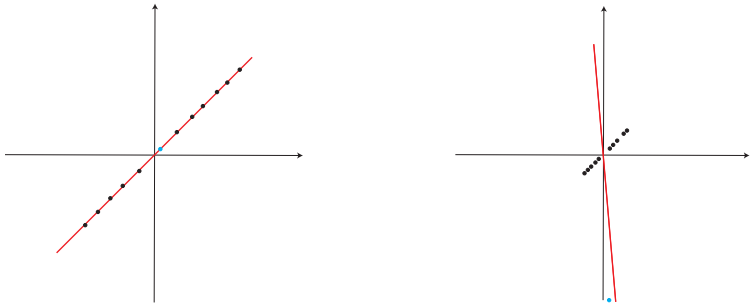
We observe that z^{k+1} and ν^{k+1} always satisfy:

$$\nabla p(z^*) + B^T \nu^* = 0$$

16.8. Robustness

16.8.1. Classical PCA

Classical PCA is very sensitive to outliers. *One single* grossly corrupted point completely changes the principal components.



The main reason is that the sum of squares is minimised:

$$\frac{1}{N} \sum_{n=1}^{N} \|x_n - \tilde{x}_n\|_2^2.$$

16.9. Robust PCA

RPCA explicitly models the errors as sparse noise due to separation into low-rank and sparse matrices:

$$\begin{aligned} &\text{minimise } rank(L) + \lambda \cdot card(S) \\ &\text{subject to } L + S = X \end{aligned}$$

which can be modelled by the principal component pursuit.

$$\begin{aligned} &\text{minimise } \|L\|_* + \lambda \|S\|_1 \\ &\text{subject to } L + S = X, \end{aligned}$$

where $\|\cdot\|_*$ denotes the nuclear norm and $\|\cdot\|$ denotes the L_1-norm of S seen as a vector.

Advantages RPCA is robust to grossly corrupted observations of X. It does not need to know the sparsity pattern of S_0. Can be easily extended to matrix completion.

Applications where large errors frequently occur are

Image processing: Salt & Pepper noise

Web data analysis: Adversarial information

Bioinformatics: Spurious errors in measurements and sensor failures.

Computer Vision: Occlusions

16.9.1. Recovery of L_0 and S_0

Find the decomposition of X into low rank and sparse matrix $X = L_0 + S_0$:
Principal Component Pursuit (PCP):

$$\begin{aligned} &\text{minimise } \|L\|_* + \lambda \|S\|_1 \\ &\text{subject to } L + S = X, \end{aligned}$$

Under broad conditions the recovery is perfect, i.e. the solution (L^*, S^*) obeys:

$$L^* = L_0, S^* = S_0.$$

- X can not be low-rank and sparse.
- L_0 can not be sparse:

$$L_0 \in \mathbb{R}^{n \times n} = U \Sigma M^T = \sum_{1 \leq i \leq r} \sigma_i u_i v_i^T \qquad r = rank(L_0)$$
$$e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$$

- Coherence condition:

$$\begin{aligned} \left\| U^T e_i \right\|^2 &\leq \frac{\mu r}{n} \\ \left\| M^T e_i \right\|^2 &\leq \frac{\mu r}{n} \\ |UM^T|_{ij}^2 &\leq \frac{\mu r}{n^2} \end{aligned}$$

  This means, that Principal Components can not be sparse (spiky).

16.9.2. Theorem

For

L_0: $n \times n$, of $rank(L_0) \leq \rho_r n \mu^{-1} (\log n)^{-2}$

S_0: $n \times n$, random sparsity pattern of cardinality $n \leq \rho_s n^2$.

ρ_s, ρ_r are positive constraints.

With probability $1 - \mathcal{O}\left(n^{-10}\right)$, PCP with $\lambda = \frac{1}{\sqrt{n}}$ is exact. The same holds for rectangular matrices with $\lambda = \frac{1}{\sqrt{\max(n,m)}}$.

16.10. Solving Robust PCA

We want to solve

$$\begin{aligned} &\text{minimise } \|L\|_* + \lambda \|S\|_1 \\ &\text{subject to } L + S = X. \end{aligned}$$

This can now be easily solved using ADMM:
We have chosen $f(x) = \|L\|_*$ and $p(z) = \|S\|_1$:

$$L_\mu(L, S, N) = \|L\|_* + \lambda \|S\|_1 + \langle N, (X - L - S) \rangle + \frac{\mu}{2} \|X - L - S\|_F^2$$

with $\langle X, Y \rangle$ being the Frobenius inner product: $\sum_i \sum_j X_{ij} Y_{ij}$.

ADMM for RPCA:

$$\begin{aligned} L^{k+1} &:= \arg\min_L L_\mu(S, L^k, N^k) \\ S^{k+1} &:= \arg\min_S L_\mu(S^{k+1}, L, N^k) \\ N^{k+1} &:= N^k + \mu(X - L^{k+1} - S^{k+1}) \end{aligned}$$

We observe that we can solve the minimisation over L and S explicitly:

$$\begin{aligned} \arg\min_S L_\mu(S, L, N) &= \mathcal{S}_{\lambda\mu^{-1}}(X - L + \mu^{-1} N) \\ \arg\min_L L_\mu(S, L, N) &= \mathcal{D}_{\mu^{-1}}(X - S + \mu^{-1} N) \end{aligned}$$

with

$$\begin{aligned} \mathcal{S}_\tau(x) &= \text{sgn}(x) \max(|x| - \tau, 0) \\ \mathcal{S}_\tau(X) &: \text{Apply } \mathcal{S}_\tau \text{ to each element} \\ \mathcal{D}_\tau(X) &= U \mathcal{S}_\tau(\Sigma) M^T \\ textwhereSVDX &= U \Sigma M^T \end{aligned}$$

Combining all of the above we get an algorithm to solve robust PCA:

$S_0 := 0$
$\nu_0 : 0$
$\mu > 0$
while not converged **do**
$L^{k+1} := \mathcal{D}_{\mu^{-1}}(X - S^k + \mu^{-1}N^k)$
$S^{k+1} := \mathcal{S}_{\lambda\mu^{-1}}(X - L^{k+1} + \mu^{-1}N^k)$
$N^{k+1} := N^k + \mu(X - L^{k+1} - S^{k+1})$
end while

Appendix

A. Matrix Definitions and Theorems

A.1. Norms

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

- *Positive scalability:*

$$\|a \cdot x\| = |a| \cdot \|x\|.$$

- *Triangle inequality*

$$\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in V.$$

- *Separability*:

$$\|x\| = 0 \implies x = 0.$$

A.1.1. Vector norms

p-norms The most commonly used matrix norms are p-norms.

$$\|x\|_p := \left(\sum_{i-1}^{n} |x_i|^p \right)^{1/p}$$

for $p \in [1, \infty]$, where $|x_i|$ denotes the absolute value of coordinate x_i.

A special case of the p norm is the *Eclidean norm*:

$$\|x\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2}.$$

0-**norm** technically not really a norm is defined by:

$$\|x\|_0 := \text{number of nonzero coordinates in } x.$$

A.1.2. Matrix norms

p-**norm** for matrices:

$$\|X\|_p := \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}.$$

A special case is the Euclidean or *spectral norm*:

$$\|X\|_2 = \sigma_{\max}(X),$$

the largest singular value of X.

Frobenius norm is defined as:

$$\|X\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}^2} = \sum_{i=1}^{\min(m,n)} \sigma_i^2,$$

where σ_i are the singular values of X.

A.2. Orthogonality

Orthogonal vectors Two vectors in an inner product are orthogonal if their inner product is zero.

Orthonormal vectors Orthogonal vectors that have unit length 1

Orthogonal matrix An orthogonal matrix is a square matrix with real entries whose columns and rows are orthogonal unit vectors (i.e. orthonormal vectors). For orthogonal matrices it also holds that

$$A^T A = I \quad \Longrightarrow \quad A^T = A^{-1} \text{ since,}$$
$$(A^T A)_{i,j} = a_i^T a_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

B. Occam's Razor

> Entities must not be multiplied beyond necessity.

It states that among competing hypotheses, the hypothesis with the fewest assumptions should be selected. It is often understood as 'the simplest explanation is usually the correct one', although this is potentially misleading. The application of the principle often shifts the burden of proof in a discussion.[a] The razor states that one should proceed to simpler theories until simplicity can be traded for greater explanatory power. The simplest available theory need not be most accurate. Philosophers also point out that the exact meaning of simplest may be nuanced.[3]

C. Probability

We denote Ω the sample space and by A an event, which is a subset of Ω.

Probability distribution A function p that assigns a real number $p(A)$ to each event A is a *probability distribution* if it satisfies the following three axioms:

- $p(A) \geq 0$ for every A.
- $p(\Omega) = 1$.

[3] http://en.wikipedia.org/wiki/Occam's_razor

- If A_1, A_2, ... are disjoint then

$$p\left(\bigcup_{i=1}^{\infty}\right) = \sum_{i=1}^{\infty} p(A_i)$$

Random Variable A random variable is a mapping:

$$\begin{aligned} X \ : \ \Omega \ &\to \ \mathcal{K} \\ \omega \ &\mapsto \ X(\omega) \end{aligned}$$

that assigns an element $X(\omega) \in \mathcal{K}$ to each outcome ω.

Notation and types of random variables (sample spaces):

Notation :

X	Random variable
x	a value taken by the r.v. X.

Discrete random variables :

- Finite: e.g. $X \in \mathcal{B} \equiv \{0, 1\}$ or $X \in \mathcal{S}_n$ (set of permutations)
- Contably Infinite: e.g $X \in \mathbb{N}, \mathbb{Z}$ etc...

Continous random variables : e.g $X \in \mathbb{R}, [a, b]$ etc...

C.1. Distributions

C.1.1. Categorical distribution

Multinomial distribution.

Example The sample space of throwing two dice is $\Omega = [1, \ldots, 6] \times [1, \ldots, 6]$. We consider the two random variables $X_1 + X_2$ given by summing up the numbers x_1, x_2 of the two dice. The random variable can thus take values in $[2, \ldots, 12]$. This leads to the following probabilites:

Random variable	2	3	4	5	6	7	8	9	10	11	12
Probability	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

The random variable $X_1 + X_2$ is an example of the *categorical distribution*: A discrete distribution over K events. Each event has the probability π_k of occurring.

Definition The categorical distribution is a discrete probability distribution whose sample space is the set of $1-\text{of}-K$ encoded random vectors x of dimensions K having the property:

$$\sum_{k=1}^{K} z_k = 1, \quad z_k \in \{0, 1\}.$$

The probability mass function is defined as

$$p(z|\pi) = \prod_{k=1}^{K} \pi_k^{z_k},$$

where π_k represents the probability of seeing element k ($\pi_k \geq 0$, $\sum_{k=1}^{K} \pi_k = 1$).

C.1.2. Gaussian Distribution

The probability density function of the Gaussian distribution is given by:

$$p(x|\mu, \sigma) = \mathcal{N}(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$
$$\mu = \text{ mean of distribution}$$
$$\sigma^2 = \text{ variance of distribution.}$$

Probability for $X \in [a, b]$ is given by an integral:

$$P(a < X < b) = \int_a^b p(x)dx = \frac{1}{\sqrt{2\pi}\sigma} \int_a^b exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} dx.$$

C.1.3. Multivariate Gaussian Distribution

Generalises the univaraite Gaussian distribution to higher dimensions. The distribution samples the space $\mathcal{X} \subseteq \mathbb{R}^D$. A random vector $X = (X_1, \ldots, X_D)^T$ has a multivariate normal distribution if every linear combination of its components (i.e., $Y = a_1X_1 + \ldots + a_DX_D$) has a univariate normal distribution.

Definition :

$$p(x|\mu\Sigma) = \mathcal{N}(x|\mu, \Sigma) := \frac{1}{(\sqrt{2\pi})^D|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

C.1.4. Multidimensional Moment Statistics

Expectation Vector of component expectations:

$$\mathbb{E}[X] = \begin{pmatrix} \mathbb{E}[X_1] \\ \vdots \\ \mathbb{E}[X_D] \end{pmatrix}$$

Variance Generalised covariance:

$$\begin{aligned} Cov[X,Y] :&= \int_{\mathcal{X}}\int_{\mathcal{Y}} p(x,y)(x-\mu_X)(y-\mu_Y)dxdy \\ &= \mathbb{E}_{X,Y}[(x-\mu_X)(y-\mu_Y)] \end{aligned}$$

Covariance Matrix For random variables $X_1, \ldots, X_D$ we record covariances in the covariance matrix Σ:

$$\Sigma_{i,j} := Cov[X_i, Y_i] \quad i,j \in \{1,\ldots,D\}.$$

Σ generalises the notion of variance to sets of random variables for multiple dimensions.

C.2. Latent Variables

Define a K-dimensional binary random variable z having a 1-of-K representation in which a particular element z_k is equal to 1 and all other elements are equal to 0, i.e.:

$$z_k \in \{0,1\}, \qquad \sum_k z_k = 1.$$

The marginal distribution over z is specified in terms of the mixing coefficients π_k, i.e.

$$p(z_k = 1) = \pi_k.$$

Because z uses a 1-of-K representation, we can write this distribution in the form:

$$p(z) = \prod_{k=1}^{K} \pi_k^{z_k}.$$

Similarly, the conditional distribution of x given a particular value for z is a gaussian:

$$p(x|z_k = 1) = \mathcal{N}(x|\mu_k, \Sigma_k)^{z_k}.$$

The conditional distribution can also be written in the form

$$p(x|z) = \prod_{k=1}^{K} \mathcal{N}(x|\mu_k, \Sigma_k)^{z_k}.$$

The marginal distribution of x is obtained by summing the joint distribution over all possible states of z to yield:

$$p(x) = \sum_z p(z)p(x|z) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k).$$

For the full data log-likelihood we get:

$$\log p(X|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k) \right\}.$$

C.3. Bayes' rule

The conditional probability of A given B (posterior) is given by:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}.$$

We call $p(A)$ prior, $p(B|A)$ likelihood and $p(B)$ evidence.

D. Lagrange Multipliers

This part was taken from Wikipedia[4].
In mathematical optimisation the method of Lagrange multipliers is a strategy for finding the local maxima and minima of a function subject to *equality constraints*.
Consider the two-dimensional problem:

$$\begin{aligned} &\text{maximise } f(x, y) \\ &\text{subject to } g(x, y) = c, \end{aligned}$$

with both f and g to having *continuous first partial derivatives*.
We can visualise contours of f given by

$$f(x, y) = d,$$

for various values of d, and the contour of g given by

$$g(x, y) = c.$$

Suppose we walk along the contour line $g = c$. In general the contour lines of f and g may be distinct, so following the contour line for $g = c$ one could intersect with or cross the contour lines of f. Only when the contour line for $g = c$ meets contour lines of f *tangentially*, do we not increase or decrease the value of f (in terms of maximising $f(x, y)$ subject to $g(x, y) = c$).
The contour lines of f and g touch when the *tangent vectors* of the contour lines are parallel. Thus we want points (x, y) where $g(x, y) = c$ and

$$\nabla_{x,y} f = -\lambda \nabla_{x,y} g.$$

To solve this system we introduce an auxiliary function:

$$\Lambda(x, y, \lambda) = f(x, y) + \lambda \cdot (g(x, y) - c),$$

and solve

$$\nabla_{x,y,\lambda} \Lambda(x, y, \lambda) = 0.$$

[4] http://en.wikipedia.org/wiki/Lagrangian_multiplier

E. Convex Set

A set C is convex if the line segment between any two points in C lies in C, i.e., if for any $x1, x2 \in C$ and any θ with $0 \leq \theta \leq 1$, we have

$$\theta x_1 + (1 - \theta)x_2 \in C.$$

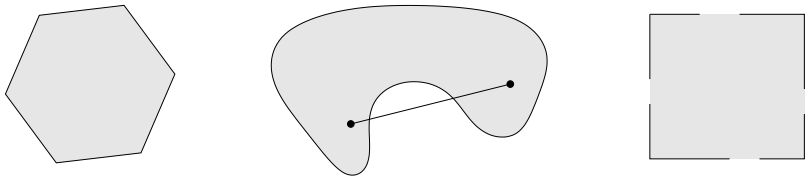


*Figure 2.2 from S. Boyd, L. Vandenberghe

Left Convex.

Middle Not convex, since the line segment is not in the set.

Right Not convex, since some, but not all boundary points are contained in the set.

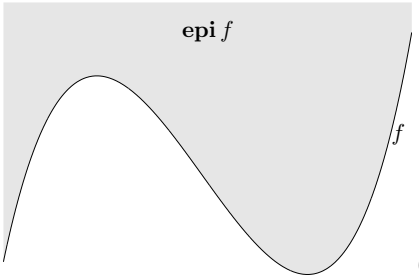
F. Convex Function

The graph of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\{(x, f(x)) \mid x \in \text{dom } f\}.$$

The *epigraph* of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\{(x, f(x)) \mid x \in \text{dom } f,\ f(x) \leq t\}.$$



Examples of convex functions

- Linear functions: $f(x) = c^T x$
- Affine functions: $f(x) = Ax + b$

- Exponential: $f(x) = e^{\alpha x}$
- Norms Every norm on $\mathbb{R}^n$ is convex.

Convexity of a norm $f(x)$ By the triangle inequality

$$f(x + y) \leq f(x) + f(y)$$

and homogeneity of a norm

$$f(\alpha x) = |\alpha| f(x) \qquad \alpha \text{ a scalar}$$

we can prove that:

$$f(\theta x + (1 - \theta)y) \leq f(\theta x) + f((1 - \theta)y) = \theta f(x) + (1 - \theta) f(y).$$