Computational Statistics
Unsupervised Learning

Luca Bortolussi

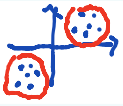
Department of Mathematics and Geosciences
University of Trieste

Office 238, third floor, H2bis
luca@dmi.units.it

Trieste, Winter Semester 2015/2016

Outline

CLUSTERING: AN OVERVIEW



Given input data $\mathbf{x_1}, \ldots, \mathbf{x_N}$, group data into K separate groups, such that points in each group are as similar as possible and points in different groups are as different as possible.

- We need a notion of dissimilarity between input points. Different measures can produce different clusters.
- Clustering can be defined as a (hard) combinatorial optimisation problem. Clustering algorithms implement different approximate search strategies.
- Some methods require to fix a priori the number of clusters (k-means, k-medoids).
- Other methods produce a tree of possible clusters (hierarchical clustering).
- Soft clustering returns a probabilistic assignment of each point to each cluster.

DISSIMILARITY MEASURES

- There are many different ways of constructing a dissimilarity between input points, depending on the nature of the data (e.g. categorical, ordinal, numerical). The choice is usually data and application oriented.
- Typically, each input point can be seen as a vector of attributes $\mathbf{x_i} = x_{i1}, \ldots, x_{in}$.
- On numerical data ($\mathbf{x} \in \mathbb{R}^n$) one usually uses a p-norm, like the (squared) Euclidean norm, or the 1-norm.
- On categorical data, one can start from a dissimilarity between single attributes and then combine it by adding the dissimilarities of single attributes in a vector of attributes, possible weighted:

$$d(\mathbf{x}, \mathbf{y}) = \sum_k w_k d(x_i, y_i)$$

- On ordinal data, one can take the distance of the (normalised) rank.

DISSIMILARITY MEASURES

- A K-clustering can be seen as a map $C : \{1, \ldots, N\} \to \{1, \ldots, K\}$, assigning each input point to a cluster.
- There are two important quantities associated with a a clustering. The within cluster distance is

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(j)=k} d(\mathbf{x_i}, \mathbf{x_j})$$

  while the between-cluster distance is

$$B(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(j)\neq k} d(\mathbf{x_i}, \mathbf{x_j})$$

- It holds that $W(C) + B(C) = T = \frac{1}{2} \sum_i \sum_{j \neq i} d(\mathbf{x_i}, \mathbf{x_j})$ where T is the total distance.
- Clustering algorithms try to solve (approximatively) the NP-hard combinatorial optimisation problem:

$$\operatorname{argmin}_C B(C) = \operatorname{argmax}_C W(C)$$

HIERARCHICAL CLUSTERING

- Hierarchial clustering combines (or divide) the dataset pairwise, producing a tree of successive groupings, called dendrogram.
- The dissimilarity measure can be used to assign a length to the edges of the dendrogram.

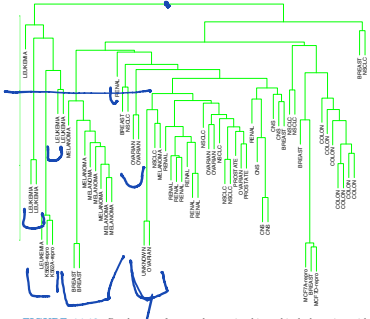
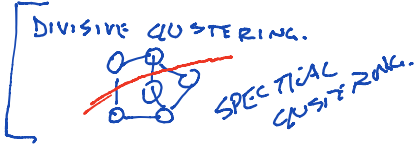


FIGURE 14.12. *Dendrogram from agglomerative hierarchical clustering with average linkage to the human tumor microarray data.*

- Agglomerative HC combines pairwise clusters (initially single data points), until they are all merged. The sequence of combinations produces the dendrogram.
- Divisive HC starts from a single cluster and splits it in two iteratively.

AGGLOMERATIVE HIERARCHICAL CLUSTERING

- Agglomerative HC keeps a list with the current clusters, and at each step combines the two clusters G, H that are closer to each other. Different ways of measure the cluster dissimilarity give rise to different dendrograms.



- Single Linkage:

$$d_{SL}(G, H) = \min_{i \in G, j \in H} d(\mathbf{x}_i, \mathbf{x}_j)$$

- Complete Linkage:

$$d_{CL}(G, H) = \max_{i \in G, j \in H} d(\mathbf{x}_i, \mathbf{x}_j)$$

- Group Average:

$$d_{GA}(G, H) = \frac{1}{N_G N_H} \sum_{i \in G, j \in H} d(\mathbf{x}_i, \mathbf{x}_j)$$

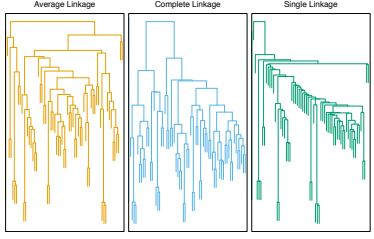
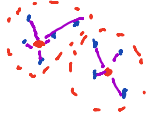


FIGURE 14.13. *Dendrograms from agglomerative hierarchical clustering of human tumor microarray data.*

k-MEANS

- The number of clusters k is fixed. The algorithm assumes numerical vectors and works with the euclidean distance.
- Each cluster is represented by its centroid $\mathbf{y}_j$. The assignment of input points $\mathbf{x}_n$ to clusters is obtained by a 1-of-k scheme, with boolean variables r_{nj} equal to one iff point $\mathbf{x}_n$ is assigned to cluster j.
- The algorithm tries to minimise the following distortion measure, related to the inter-cluster distance:

$$J = \sum_{j=1}^{k} \sum_{n=1}^{N} r_{nj} \|\mathbf{x}_n - \mathbf{y}_j\|^2$$



k-MEANS

- Minimisation of J follows a greedy strategy, and alternates between two steps:
- Minimise J w.r.t. r_{nj} holding $\mathbf{y}_j$ fixed. This is achieved by assigning each point $\mathbf{x}_n$ to the closest centroid (ties broken arbitrarily).
- Minimise J w.r.t. $\mathbf{y}_j$. The derivative in this case is

$$2 \sum_n r_{nj}(\mathbf{x}_n - \mathbf{y}_j) \stackrel{!}{=} 0$$

leading to the solution:

$$\mathbf{y}_j = \frac{\sum_n r_{nj} \mathbf{x}_n}{\sum_n r_{nj}}$$

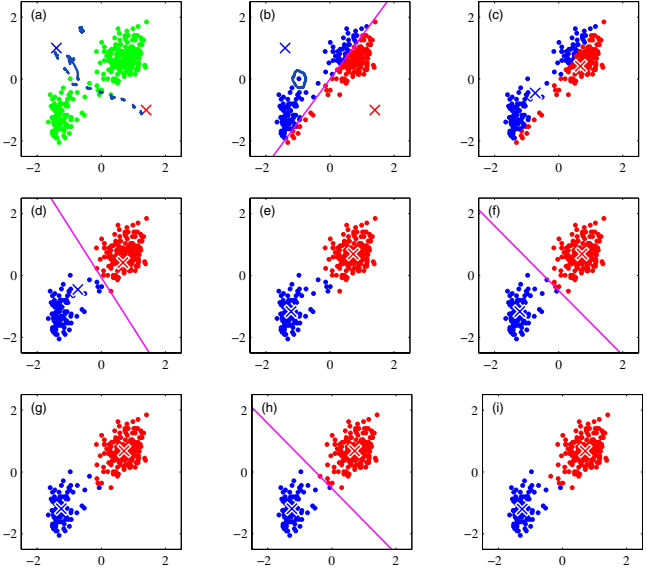
BARICENTRUM

N_j : points in j

i.e. each $\mathbf{y}_j$ is reassigned to the current cluster center.

- The algorithm iterates until convergence. Initially, centroids can be initialised randomly or to random data points (preferrable).

k-MEANS



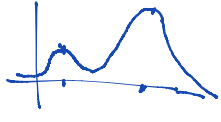
k-MEDOIDS

- Works similarly to k-means, with two major differences:
- The distance between two points $\mathbf{x}$ and $\mathbf{x}'$ is given by a generic function $D(\mathbf{x}, \mathbf{x}')$.
- Centroids are restricted to be selected among data points.
- restricting centroids to datapoints makes the algorithm more robust to outliers.

MIXTURES OF GAUSSIANS

- This is a soft clustering technique: each point will have a certain probability of being assigned to any of the classes.
- It is a generative approach, assuming data is generated by a mixture of Gaussians of the form

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



- We can then learn from the input data the parameters of the mixtures, and compute the probability of assigning each point $\mathbf{x}$ to a class k.
- This learning problem is best solved by introducing latent variables $\mathbf{z}$ for the class of each point $\mathbf{x}$, and the using the Expectation-Maximisation algorithm maximise the likelihood.

MIXTURES OF GAUSSIANS

- Let us introduce latent variables $\mathbf{z} = (z_1, \ldots, z_K)$, such that z_k is one iff a point belongs to the k-th Gaussian in the mixture.
- Latent variables are not observed, but we can assume the full input would consist of pairs $(\mathbf{x}_n, \mathbf{z}_n)$.
- Then $p(\mathbf{x})$ is the marginal distribution

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) = \sum_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z}),$$

where

$$p(\mathbf{z}) = \prod_k \pi_k^{z_k}$$

and

$$p(\mathbf{x}|\mathbf{z}) = \prod_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)^{z_k}$$

- An important quantity is the responsibility $\gamma(z_k)$ (i.e. the probability of assigning $\mathbf{x}$ to class k):

$$\gamma(z_k) = p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_k \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)} = \frac{p(x,z)}{p(x)}$$

Outline

LATENT VARIABLES

- Expectation-Maximisation (EM) is a general algorithm to maximise likelihood for models with observed variables $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N$ and latent (non-observed) variables $\mathbf{Z} = \mathbf{z}_1, \ldots, \mathbf{z}_N$.
- We assume family of models parameterised by θ. The log-likelihood we have to optimise is

$$\log p(\mathbf{X}|\theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta)$$

- With some work, one can prove that the following decomposition holds (where $q(\mathbf{Z})$ is a generic distribution on $\mathbf{Z}$):

$$\log p(\mathbf{X}|\theta) = \mathcal{L}(q, \theta) + KL(q||p)$$

$$\mathcal{L}(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{X}, \mathbf{Z}|\theta)}{q(\mathbf{Z})}$$

$$KL(q||p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{Z}|\mathbf{X}, \theta)}{q(\mathbf{Z})}$$

LIKELIHOOD DECOMPOSITION

- Let's prove: $\log p(\mathbf{X}|\theta) = \mathcal{L}(q,\theta) + KL(q||p)$, with

$$\mathcal{L}(q,\theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{X},\mathbf{Z}|\theta)}{q(\mathbf{Z})} \qquad KL(q||p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{Z}|\mathbf{X},\theta)}{q(\mathbf{Z})}$$

$$\log p(X|\theta) = \sum_{Z} q(Z) \log p(X|\theta) =$$

$$= \sum_{Z} q(Z) \cdot \left[\log p(X|\theta) + \log p(Z|X,\theta) - \log p(Z|X,\theta) + \log q(Z) - \log q(Z) \right]$$

$$= \underbrace{\sum_{Z} q(Z) \log \frac{p(X,Z|\theta)}{q(Z)}}_{\mathcal{L}(q,\theta)} - \underbrace{\sum_{Z} q(Z) \log \frac{p(Z|X,\theta)}{q(Z)}}_{KL(q||p)}$$

LIKELIHOOD DECOMPOSITION

- Let's prove: $\log p(\mathbf{X}|\theta) = \mathcal{L}(q,\theta) + KL(q||p)$, with

$$\mathcal{L}(q,\theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{X},\mathbf{Z}|\theta)}{q(\mathbf{Z})} \quad KL(q||p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{Z}|\mathbf{X},\theta)}{q(\mathbf{Z})}$$

- (Use $\log p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log p(\mathbf{X}|\theta)$, add and subtract to the log factor $\log p(\mathbf{Z}|\mathbf{X},\theta)$, then use $\log p(\mathbf{X}|\theta) + \log p(\mathbf{Z}|\mathbf{X},\theta) = \log p(\mathbf{X},\mathbf{Z}|\theta)$, finally add and remove $\sum_{\mathbf{Z}} q(\mathbf{Z}) \log q(\mathbf{Z})$.)

LIKELIHOOD DECOMPOSITION AND EM

- $\mathcal{L}(q, \theta)$ is a functional of q (it is a distribution on the latent variables Z) and a function of the parameters θ.
- As $KL(q||p) \geq 0$ with equality iff $q = p(\mathbf{Z}|\mathbf{X}, \theta)$, it follows that

$$\mathcal{L}(q, \theta) \leq \log p(\mathbf{X}|\theta)$$

  i.e. $\mathcal{L}(q, \theta)$ is a lower bound on the log likelihood of interest.
- Expectation-Maximisation is an optimisation algorithm which optimises the lower bound $\mathcal{L}(q, \theta)$ alternating two phases: one in which $\mathcal{L}$ is optimised w.r.t. q (E step) and one in which it is optimised w.r.t. θ (M step).
- It is guaranteed to converge to a local optimum of $\log p(\mathbf{X}|\theta)$.

EXPECTATION STEP

- In the E step, $\mathcal{L}(q, \theta)$ is optimised w.r.t. $q(\mathbf{Z})$, holding the current value θ_{old} of θ fixed.
- To find the solution, consider the decomposition $\log p(\mathbf{X}|\theta_{old}) = \mathcal{L}(q, \theta_{old}) + KL(q||p)$, and note that $\log p(\mathbf{X}|\theta_{old})$ does not depend on q, hence the value of $\mathcal{L}(q, \theta_{old})$ can never exceed $\log p(\mathbf{X}|\theta_{old})$.
- Furthermore, it attains this value when $KL(q||p) = 0$, i.e. for

$$q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta_{old})$$

- When observations $\mathbf{X}_n$ are i.i.d., with corresponding latent variables $\mathbf{z}_n$, then $p(\mathbf{Z}|\mathbf{X}, \theta)$ factorises w.r.t. observations:

$$p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) = \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{\sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})} = \frac{\prod_{n=1}^{N} p(\mathbf{x}_n, \mathbf{z}_n|\boldsymbol{\theta})}{\sum_{\mathbf{Z}} \prod_{n=1}^{N} p(\mathbf{x}_n, \mathbf{z}_n|\boldsymbol{\theta})} = \prod_{n=1}^{N} p(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta}) \quad (9.75)$$

MAXIMISATION STEP

- In the M step, the distribution $q(\mathbf{Z})$ is held fixed, and the lower bound $\mathcal{L}(q, \theta)$ is optimised w.r.t. θ, obtaining a novel point θ_{new}.
- For the new value θ_{new}, $\mathcal{L}(q, \theta_{new})$ does not necessarily coincide with $\log p(\mathbf{X}|\theta_{new})$, i.e. the KL-divergence is generally non-zero.
- In particular, as we are optimising, this implies that both (a) the value of $\mathcal{L}(q, \theta)$ and (b) the value of $\log p(\mathbf{X}|\theta)$ are increased in the M step.
- By plugging $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta_{old})$ into $\mathcal{L}(q, \theta)$, we see that we are optimising

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$$

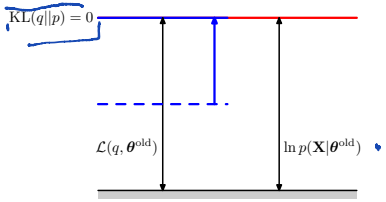
which can be rewritten as

$$\mathcal{L}(q, \theta) = \mathbb{E}_{\mathbf{Z}|\mathbf{X}, \theta_{old}}[\log p(\mathbf{X}, \mathbf{Z}|\theta)] + H(\mathbf{Z}|\mathbf{X}, \theta_{old})$$

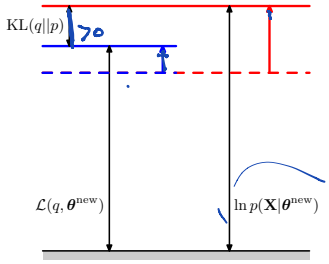
ENERGY TERM

ENTROPY TERM

EM VISUALLY

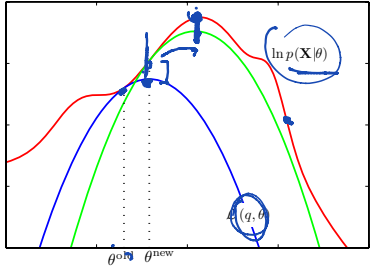


E step



M step

Figure 9.14 The EM algorithm involves alternately computing a lower bound on the log likelihood for the current parameter values and then maximizing this bound to obtain the new parameter values. See the text for a full discussion.



EM: MISCELLANEA

- In the EM algorithm, both the E and the M steps increase the lower bound, and a complete cycle increases the full log-likelihood. Hence, the algorithm will eventually converge to a (local) maximum of the full log-likelihood.
- A similar approach can be used to maximise the log-posterior distribution $\log p(\theta|\mathbf{X}) = \log p(\theta, \mathbf{X}) - \log p(\mathbf{X})$:

$$\begin{aligned} \ln p(\boldsymbol{\theta}|\mathbf{X}) &= \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q\|p) + \ln p(\boldsymbol{\theta}) - \ln p(\mathbf{X}) \\ &\geqslant \mathcal{L}(q, \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) - \ln p(\mathbf{X}). \end{aligned} \tag{9.77}$$

  Here the E step is the same ($\log p(\theta)$) does not depend on q, while the M step is required to maximise $\mathcal{L}(q, \theta) + \log p(\theta)$.
- There are several Generalised EM (GEM) algorithms that try to overcome a hard E or M step. E.g. the M step can be replaced by some steps increasing $\mathcal{L}(q, \theta)$ without reaching an optimum.

EM FOR THE MIXTURE OF GAUSSIANS

- Remember that for a mixture of K gaussians, we have

$$p(\mathbf{Z}) = \prod_n \prod_k \pi_k^{z_{nk}}$$

and

$$p(\mathbf{X}|\mathbf{Z}) = \prod_n \prod_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)^{z_{nk}}$$

hence the log-likelihood of the joint distribution is

$$\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \{\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\} . \tag{9.36}$$

EM FOR THE MIXTURE OF GAUSSIANS

- In the E step, we need to compute $p(\mathbf{Z}|\mathbf{X}, \mu\Sigma, \pi)$, which is given by

$$p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \propto \prod_{n=1}^{N} \prod_{k=1}^{K} [\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_{nk}} . \tag{9.38}$$

where the expectations of the z_{nk} are

$$\begin{aligned} \mathbb{E}[z_{nk}] &= \frac{\sum_{z_{nk}} z_{nk} \left[\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]^{z_{nk}}}{\sum_{z_{nj}} \left[\pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\right]^{z_{nj}}} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} = \gamma(z_{nk}) \end{aligned} \tag{9.39}$$

EM FOR THE MIXTURE OF GAUSSIANS

- In the M step, we first compute the expectation w.r.t. $p(\mathbf{Z}|\mathbf{X}, \mu\Sigma, \pi)$, of the complete data log-likelihood

$$\mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})] = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \{\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}. \quad (9.40)$$

- Then we maximise this expression w.r.t. the parameters, obtaining

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \quad (9.24)$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^{\mathrm{T}} \quad (9.25)$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \quad (9.26)$$

EM FOR THE MIXTURE OF GAUSSIANS

- The algorithm is initialised by choosing μ_k, Σ_k, π_k. Typically, one runs a k-means clustering, and initialised the parameters as the result of the clustering:
  - μ_k, Σ_k: sample mean and variances in cluster k;
  - π_k: fraction of data points in cluster k.
- Each loop the EM algorithm thus compute the responsibilities and the new mean, variance and mixture probabilities.
- Computation is iterated until convergence is met, i.e. the change in parameters, or in the log-likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{9.28}$$

becomes smaller than a prescribed error.

MIXTURE OF GAUSSIANS AND k-MEANS

- k-means and mixtures of Gaussians are related: the latter is a soft version of k-means: each data point is assigned to each cluster with a given probability.
- Suppose we run EM on a gaussian mixture, by fixing the covariance to be equal to ϵI, where ϵ is held fixed. The responsibilities are now estimated as

$$\gamma(z_{nk}) = \frac{\pi_k \exp\{-\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2/2\epsilon\}}{\sum_j \pi_j \exp\{-\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2/2\epsilon\}}. \tag{9.42}$$

- In the limit $\epsilon \to 0$, this converges to 1 for the component minimising $\|\mathbf{x}_n - \mu_k\|$ (as in k-means). Means also converge to the same expression for k-means. Furthermore, the data log-likelihood in this limit is

$$\mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})] \to -\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 + \text{const.} \tag{9.43}$$

i.e. EM and k-means minimise the same score function.

Outline

PRINCIPAL COMPONENT ANALYSIS

- PCA is a widely used method for dimensionality reduction, feature extraction, lossy compression, and visualisation.
- The starting point is a dataset $\mathbf{X}$ of d-dimensional input data $\mathbf{x}_1, \ldots, \mathbf{x}_N$.
- It is a linear projection technique. The idea is to project a d-dimensional dataset into an m-dimensional one, $m < d$, such that either (a) the total sum of square error is minimised or (b) the variance of the projected data is maximised.
- Both methods lead to the same result.
- The so obtained linear subspace is known as principal subspace, and its axes as principal components.
- There exist a probabilistic formulation of PCA, which assumes a linear Gaussian generative model for the data and learns its parameters by maximum likelihood, possibly exploiting an EM algorithm.

PCA: MAXIMUM VARIANCE FORMULATION

- Consider a dataset $\mathbf{X}$, and assume $\mathbf{u}_1, \ldots, \mathbf{u}_m$ is an orthonormal basis of the m-dimensional space we are looking for. Arrange them column-wise in a matrix U.
- The projection of a point $\mathbf{x}_n$ in the subspace spanned by U is given by $U^T \mathbf{x}_j$.
- The mean of the projected data is thus $U^T \bar{\mathbf{x}}$, where $\bar{\mathbf{x}} = \frac{1}{N} \sum_n \mathbf{x}_n$.
- The variance of the projected data instead is

$$\frac{1}{N} \sum_n [\mathbf{U}^T \mathbf{x}_n - U^T \bar{\mathbf{x}}]^2 = U^T S U$$

where

$$S = \frac{1}{N} \sum_n (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$$

is the data-covariance matrix

PCA: MAXIMUM VARIANCE FORMULATION

- Consider the case $m = 1$, for simplicity. We maximise the variance, subject to the constraint that $\mathbf{u}_1$ is normalised (otherwise the optimal solution is to take it to infinity). For this we introduce a Langrange multiplier λ_1, and maximise the Lagrangian:

$$\mathbf{u}_1^T S \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^T \mathbf{u}_1)$$

- Deriving w.r.t. $\mathbf{u}_1$ and setting to zero we get

$$S\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

hence λ_1 is an eigenvalue of S and $\mathbf{u}_1$ an eigenvector.

- Multiplying both sides for $\mathbf{u}_1^T$ and using $\mathbf{u}_1^T \mathbf{u}_1 = 1$, we get

$$\mathbf{u}_1^T S \mathbf{u}_1 = \lambda_1$$

which shows that the variance is maximised by taking the eigenvector of the largest eigenvalue of S.

PCA: MAXIMUM VARIANCE FORMULATION

- In the general case of $m > 1$, one can inductively show that the optimal choice is to the the eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_m$ associated to the largest m eigenvalues $\lambda_1, \ldots, \lambda_m$.
- The cost of finding all eigenvalues/ eigenvectors of S is $O(d^3)$ (plus the cost of computing S, which is $O(Nd^2)$. If we are only interested in m eigenvectors, we can use specialised algorithms that cost $O(md^2)$.

PCA: MINIMUM-ERROR FORMULATION

- Here we take a complementary approach to variance maximisation. We fix an orthonormal basis $\mathbf{u}_j$, and express the data points in this new basis, as

$$\mathbf{x}_n = \sum_j (\mathbf{x}_n^T \mathbf{u}_j)\mathbf{u}_j$$

- The goal is to best approximate these points using only m dimensions, i.e. with points of the form

$$\tilde{\mathbf{x}}_n = \sum_{j=1}^{m} z_{nj}\mathbf{u}_j + \sum_{j=m+1}^{d} b_j \mathbf{u}_j$$

where $z_{nj} = \mathbf{x}_n^T \mathbf{u}_j$ and $b_j = \bar{\mathbf{x}}^T \mathbf{u}_j$.

PCA: MINIMUM-ERROR FORMULATION

- By taking the mean sum of square error,

$$J = \frac{1}{N} \sum_n \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2$$

and inserting the expression for $\tilde{\mathbf{x}}_n$, we get

$$J = \sum_{j=m+1}^{d} \mathbf{u}_j^T S \mathbf{u}_j$$

- From this expression, using lagrange multipliers like for the max variance case, we see immediately that the minimum is obtained by taking the m principal components as the eigenvectors of the m largest eigenvalues, so that J is the sum of the $d - m$ smallest eigenvalues.

PCA APPLICATIONS

- Dimensionality reduction: run PCA for the m largest eigenvalues explaining $\alpha\%$ of the data variance.
- Data compression: reduce coordinates of points by PCA and reconstruct them by using $\tilde{\mathbf{x}}_n = \sum_{j=1}^{m} z_{nj}\mathbf{u}_j + \sum_{j=m+1}^{d} b_j\mathbf{u}_j$.
  Example: handwritten digits

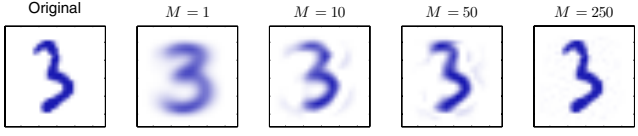


Figure 12.5 An original example from the off-line digits data set together with its PCA reconstructions obtained by retaining M principal components for various values of M. As M increases the reconstruction becomes more accurate and would become perfect when $M = D = 28 \times 28 = 784$.

PCA APPLICATIONS

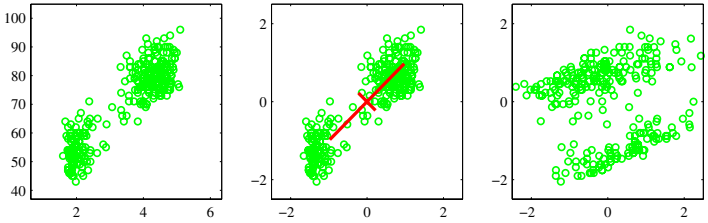
- A third common application is data renormalisation, a technique usually known as withening or sphering.
- The idea is to do a PCA with $m = d$, in order to make the data have zero mean and unit covariance.
- Consider the full eigenvalue equation $SU = UL$, where L is the diagonal matrix with eigenvalues. After solving it, we renormalise data as

$$\mathbf{y}_n = L^{-1/2} U^T (\mathbf{x}_n - \bar{\mathbf{x}})$$

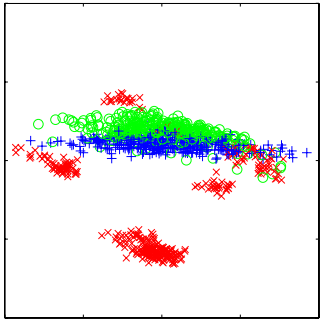
- These new points have unit covariance:

$$\begin{aligned} \frac{1}{N}\sum_{n=1}^{N} \mathbf{y}_n \mathbf{y}_n^{\mathrm{T}} &= \frac{1}{N}\sum_{n=1}^{N} \mathbf{L}^{-1/2}\mathbf{U}^{\mathrm{T}}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}\mathbf{U}\mathbf{L}^{-1/2} \\ &= \mathbf{L}^{-1/2}\mathbf{U}^{\mathrm{T}}\mathbf{S}\mathbf{U}\mathbf{L}^{-1/2} = \mathbf{L}^{-1/2}\mathbf{L}\mathbf{L}^{-1/2} = \mathbf{I} \end{aligned} \tag{12.25}$$

PCA applications



- An example of withering above.
- Finally, PCA can be used for data visualisation, by projecting data on a 2D space.



PCA IN HIGH DIMENSIONS

- By defining the matrix X with rows $(\mathbf{x}_n - \bar{\mathbf{x}})^T$, we have that $S = N^{-1}X^T X$, hence the eigenvector equation is $N^{-1}X^T X\mathbf{u}_i = \lambda_i \mathbf{u}_i$.
- By multiplying both sides by X, and calling $X\mathbf{u}_i = \mathbf{v}_i$, we have that the equation $N^{-1}XX^T\mathbf{v}_i = \lambda_i \mathbf{v}_i$ holds for the same eigenvalues.
- We can solve it for $\mathbf{v}_i$ and obtain $\mathbf{u}_i$ back by setting

$$\mathbf{u}_i = \frac{1}{(N\lambda_i)^{1/2}} X^T \mathbf{v}_i$$

- The matrix XX^T is $N \times N$, while $X^T X$ is $d \times d$, hence if $N << d$, this second formulation is more convenient (notice: $X^T X$ will have at most $N - 1$ non null eigenvalues).

PROBABILISTIC PCA

- Probabilistic PCA rephrases PCA in a probabilistic framework by defining a generative model for the data. Assume $\mathbf{z}$ is a vector in $\mathbb{R}^m$, with distribution $\mathcal{N}(\mathbf{z}|\mathbf{0}, I)$. The generative model for $\mathbf{x}$ is

$$\mathbf{x} = W\mathbf{z} + \mu + \epsilon$$

  with $\epsilon = \mathcal{N}(0, \sigma^2 I)$.
- Hence Probabilstic PCA learns a map (i.e. W, μ, σ^2) from the low dimensional space to the high dimensional one, by maximum likelihood.
- Solution is $\mu = \bar{\mathbf{x}}$, $W = U(L - \sigma^2 I)^{1/2} R$, $\sigma^2 = 1/(d-m) \sum_{j=m+1}^{d} \lambda_j$; where U is the matrix with columns given by the m largest eigenvectors of S, L is the diagonal matrix with the m largest eigenvalues, and R is an arbitrary rotation matrix.
- The projection of a point $\mathbf{x}$ is given by $\mathbb{E}[\mathbf{z}|\mathbf{x}] = MW^T(\mathbf{x} - \bar{\mathbf{x}})$, with $M = W^T W + \sigma^2 I$ ($\mathbf{z}$ is pushed closer to $\mathbf{0}$ than with PCA).
- For $\sigma^2 \to 0$, we obtain back the classic PCA solution.

Other dimensionality reduction methods

- There are many dimensionality reduction techniques, that try to circumvent the limitations of PCA, mainly the linearity of the manifold we project into.
- We list a few here: kernel PCA, using a dual formulation in terms of kernels, principal curves and surfaces, working with non-linear manifolds, autoassociative neural networks, for which the projection is expressed as a NN, Indipendent component analysis, which uses a probabilistic formulation with a non-gaussian, but factorised distribution over the reduced variables $\mathbf{z}$.