T-61.3050 Machine Learning: Basic Principles

Clustering

Kai Puolamäki

Laboratory of Computer and Information Science (CIS)
Department of Computer Science and Engineering
Helsinki University of Technology (TKK)

Autumn 2007

Remaining Lectures

- 6 Nov: Dimensionality Reduction & Clustering (Aplaydin Ch 6&7)
- 13 Nov: Clustering & Algorithms in Data Analysis (PDF chapter)
- 20 Nov: Assessing Algorithms & Decision Trees (Alpaydin Ch 14&9)
- 27 Nov: Machine Learning @ Google /TBA (additionally, Google recruitment talk in afternoon in T1 at 16 o'clock, see `http://www.cis.hut.fi/googletalk07/`)
- 4 Dec: Decision Trees & Linear Discrimination (Alpaydin Ch 10)
- (7 Dec: last problem session.)
- 11 Dec: Recap
- The plan is preliminary (may still change)

About the Text Book

- This course has Alpaydin (2004) as a text book.
- The lecture slides (neither mine nor the ones on the Alpaydin's site) are not meant to be a replacement for the text book.
- It is important also to read the book chapters.
- Library has some reading room copies (they are planning to order some more). If nothing else, you should probably at least copy some key chapters.

Outline

Principal Component Analysis (PCA)

- PCA finds low-dimensional linear subspace such that when $\mathbf{x}$ is projected there information loss (here defined as variance) is minimized.
- Finds directions of maximal variance.
- Projection pursuit: find direction $\mathbf{w}$ such that some measure (here variance $\mathrm{Var}(\mathbf{w}^T\mathbf{x})$) is maximized.
- Equivalent to finding eigenvalues and -vectors of covariance or correlation matrix.

Principal Component Analysis (PCA)

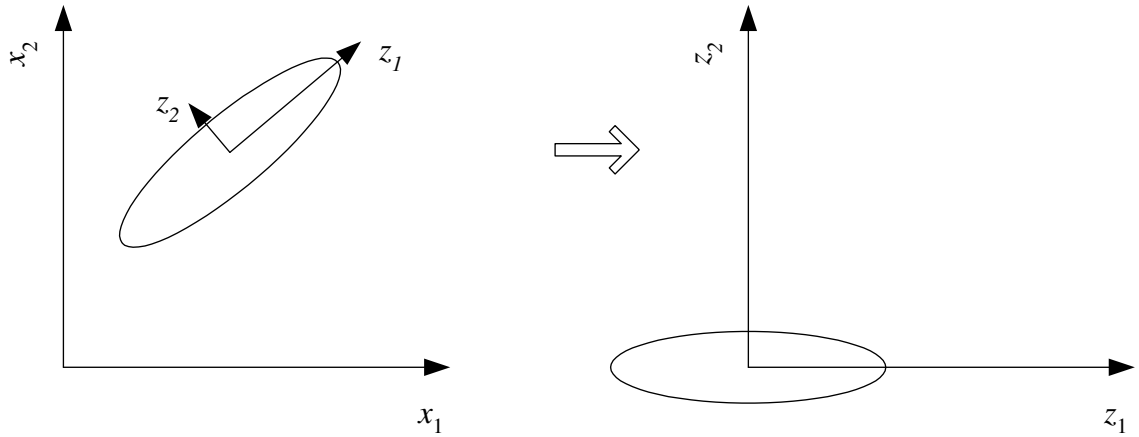


Figure 6.1: Principal components analysis centers the sample and then rotates the axes to line up with the directions of highest variance. If the variance on z_2 is too small, it can be ignored and we have dimensionality reduction from two to one. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Principal Component Analysis (PCA)

- More formally: data $\mathcal{X} = \{\mathbf{x}^t\}_{t=1}^N$, $\mathbf{x}^t \in \mathbb{R}^d$.
- Center data: $\mathbf{y}^t = \mathbf{x}^t - \mathbf{m}$, where $\mathbf{m} = \sum_t \mathbf{x}^t / N$.
- Two options:
  - Use covariance matrix $S = \sum_t \mathbf{y}\mathbf{y}^T / N$.
  - Use correlation matrix R, where $R_{ij} = S_{ij} / \sqrt{S_{ii} S_{jj}}$.
- Diagonalize S (or R) using Singular Value Decomposition (SVD): $C^T S C = D$, where C is an orthogonal (rotation) matrix satisfying $CC^T = C^T C = \mathbf{1}$ and D is a diagonal matrix whose diagonal elements are the eigenvalues $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$.
- ith column of C is the ith eigenvector.
- Project data vectors $\mathbf{y}^t$ to principal components $\mathbf{z}^t = C^T \mathbf{y}^t$ (equivalently $\mathbf{y}^t = C\mathbf{z}^t$).

Principal Component Analysis (PCA)

- Observation: covariance matrix of $\{\mathbf{z}^t\}_{t=1}^N$ is a diagonal matrix D whose diagonal elements are the variances.

$$\begin{aligned} S_{\mathbf{z}} &= \sum_t \mathbf{z}\mathbf{z}^T/N = \sum_t C^T\mathbf{y}\mathbf{y}^T C/N \\ &= C^T\left(\sum_t \mathbf{y}\mathbf{y}^T/N\right) C = C^T S C = D, \end{aligned}$$

  where the diagonal elements of D are the variances $D_{ii} = \sigma^2_{\mathbf{z}i}$.

- Eigenvalues $\lambda_i \Leftrightarrow$ variances σ_i^2.

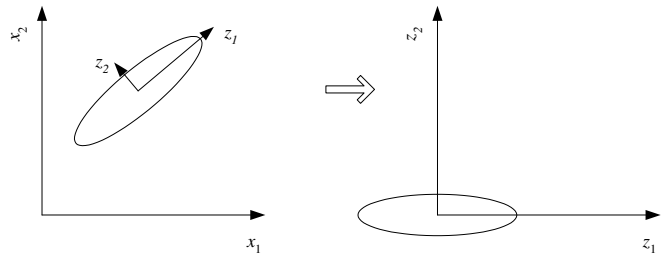


Figure 6.1: Principal components analysis centers the sample and then rotates the axes to line up with the directions of highest variance. If the variance on z_2 is too small, it can be ignored and we have dimensionality reduction from two to one. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Principal Component Analysis (PCA)

- Idea: in the PC space ($\mathbf{z}$ space), k first principal components explain the data well enough, where $k < d$.
- "Well enough" means here that the reconstruction error is small enough. More formally:
- Project the data vectors $\mathbf{y}^t$ into $\mathbb{R}^k$ using $\hat{\mathbf{z}}^t = W^T \mathbf{y}^t$, where $W \in \mathbb{R}^{d \times k}$ is a matrix containing the first k columns of C. ("W <- C[,1:k]"). $\hat{\mathbf{z}}^t$ is a representation of $\mathbf{y}^t$ in k dimensions.
- Project $\hat{\mathbf{z}}^t$ back to $\mathbf{y}^t$ space:

$$\hat{\mathbf{y}}^t = W\hat{\mathbf{z}}^t = WW^T\mathbf{y}^t$$

What is the average reconstruction error $\mathcal{E} = \sum_t (\hat{\mathbf{y}}^t - \mathbf{y}^t)^T (\hat{\mathbf{y}}^t - \mathbf{y}^t)/N$?

Principal Component Analysis (PCA)

- What is the average reconstruction error $\mathcal{E} = \sum_t \left(\hat{\mathbf{y}}^t - \mathbf{y}^t\right)^T \left(\hat{\mathbf{y}}^t - \mathbf{y}^t\right)/N$?

$$
\begin{aligned}
\mathcal{E} &= \mathrm{Tr}(E\left[(\hat{\mathbf{y}} - \mathbf{y})\left(\hat{\mathbf{y}} - \mathbf{y}\right)\right]) \\
&= \mathrm{Tr}\left(\left(WW^T - \mathbf{1}\right) E\left[\mathbf{y}\mathbf{y}^T\right]\left(WW^T - \mathbf{1}\right)\right) \\
&= \mathrm{Tr}\left(WW^T CDC^T WW^T\right) + \mathrm{Tr}\left(CDC^T\right) - 2\mathrm{Tr}\left(W^T CDC^T W\right) \\
&= \sum_{i=k+1}^{d} \lambda_i,
\end{aligned}
$$

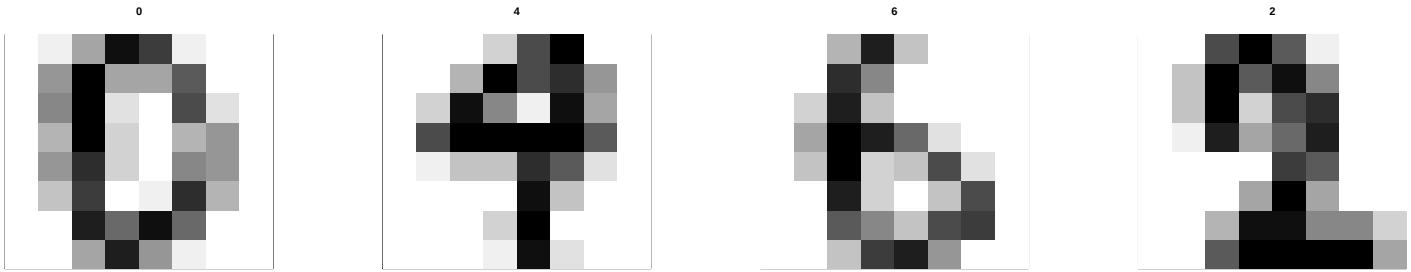
where we have used the fact that $S = CDC^T = E\left[\mathbf{y}\mathbf{y}^T\right]$ and the cyclic property of the trace, $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$.

Principal Component Analysis (PCA)

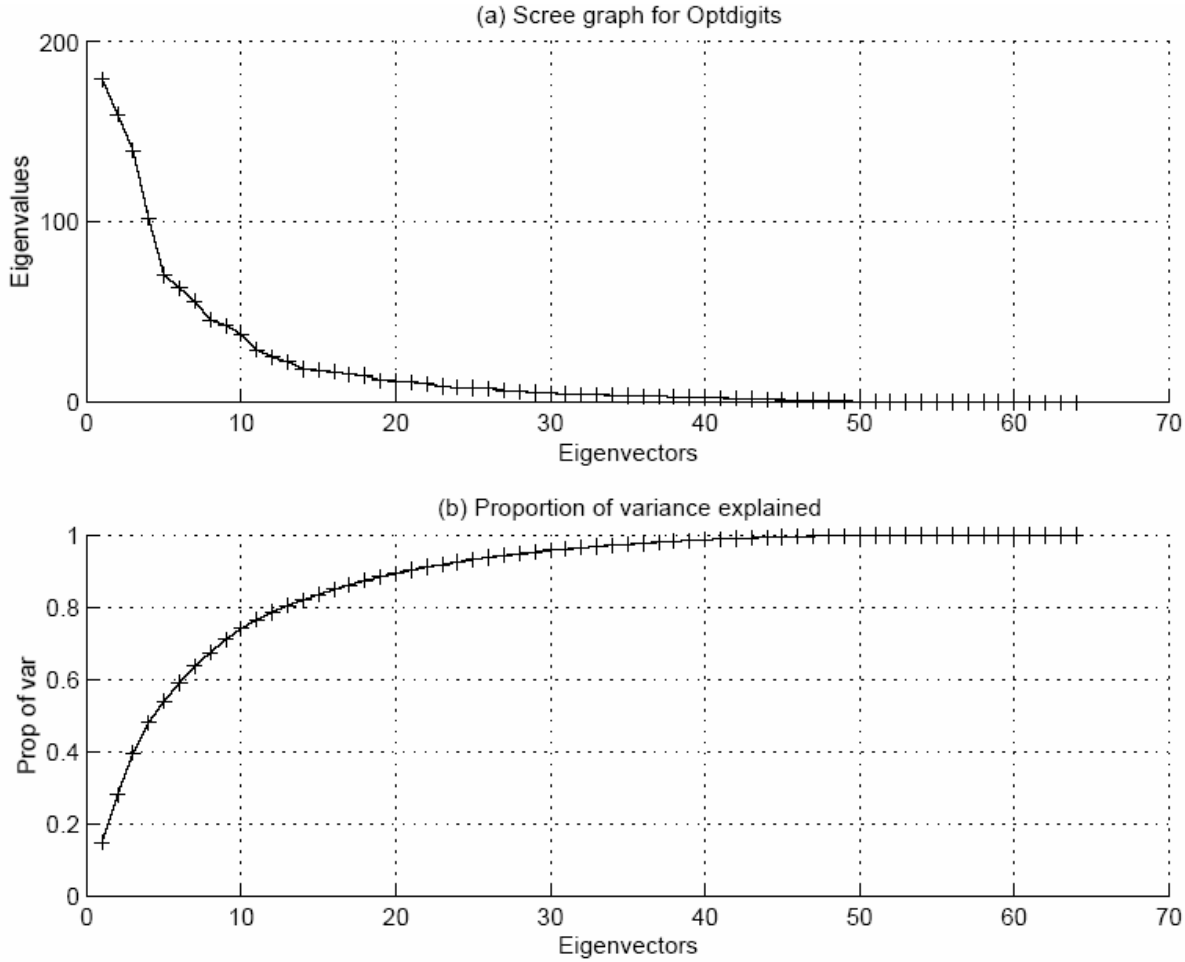
- Result: PCA is a linear projection of data from $\mathbb{R}^d$ into $\mathbb{R}^k$ such that the average reconstruction error $\mathcal{E} = E\left[(\hat{\mathbf{y}} - \mathbf{y})^T (\hat{\mathbf{y}} - \mathbf{y})\right]$ is minimized.
- Proportion of Variance (PoV) Explained: $\text{PoV} = \sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{d} \lambda_i$. Some rules of thumb to find a good k: $\text{PoV} \approx 0.9$, or PoV curve has an elbow.
- Dimension reduction: it may be sufficient to use $\hat{\mathbf{z}}^t$ instead of $\hat{\mathbf{x}}^t$ to train a classifier etc.
- Visualization: plotting the data to $\hat{\mathbf{z}}^t$ using $k = 2$ (first thing to do with new data).
- Data compression: instead of storing the full data vectors $\mathbf{y}^t$ it may be sufficient to store only $\hat{\mathbf{z}}^t$ and then reconstruct the original data using $\hat{\mathbf{y}}^t = W\hat{\mathbf{z}}^t$, if necessary.

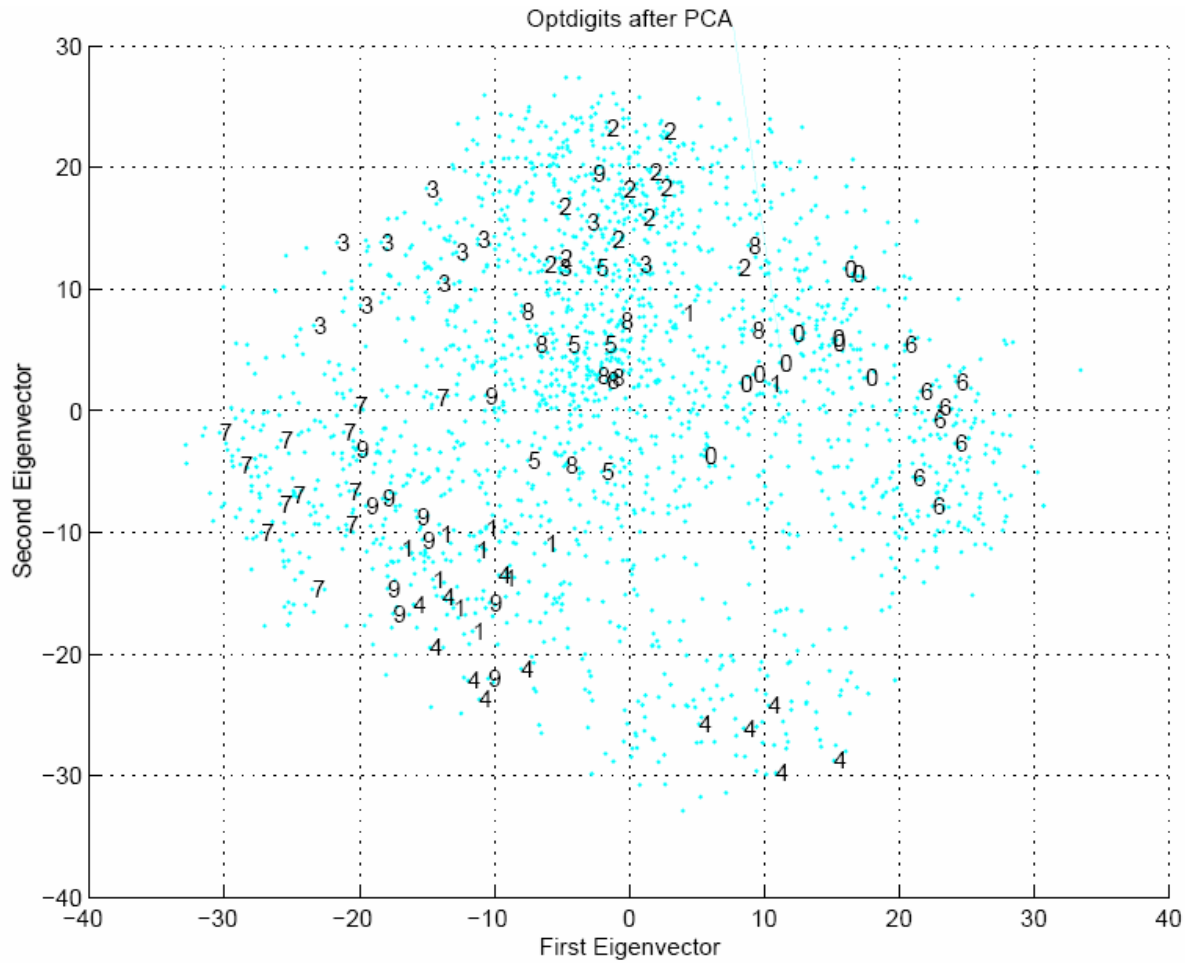
Example: Optdigits

- OPTDIGITS data set contains 5620 instances of digitized handwritten digits in range 0–9.
- Each digit is a $\mathbb{R}^{64}$ vector: $8 \times 8 = 64$ pixels, 16 grayscales.



Example: Optdigits

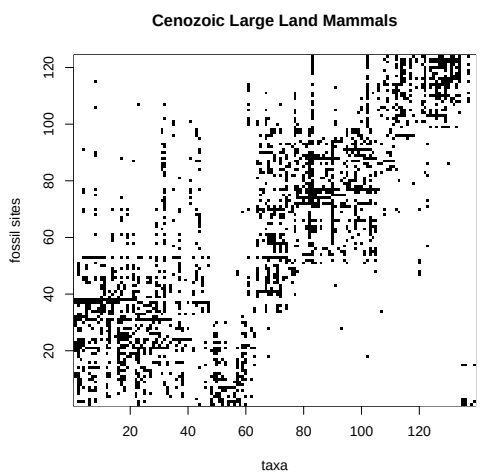


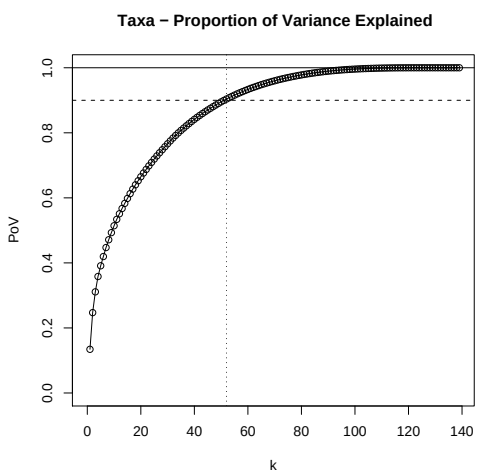


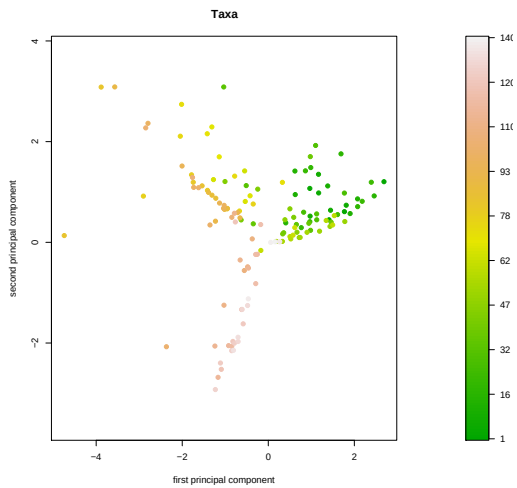
Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press (V1.1)

Example: Fossils

- Large European land mammals: 124 fossil find sites (dated 23–2 million years old), 139 taxa
- Reconstruction of site vectors given PCA taxon representation for different k: $\hat{\mathbf{y}} = W\hat{\mathbf{z}} = WW^T\mathbf{y}$, or $\hat{\mathbf{x}} = WW^T(\mathbf{x} - \mathbf{m}) + \mathbf{m}$.

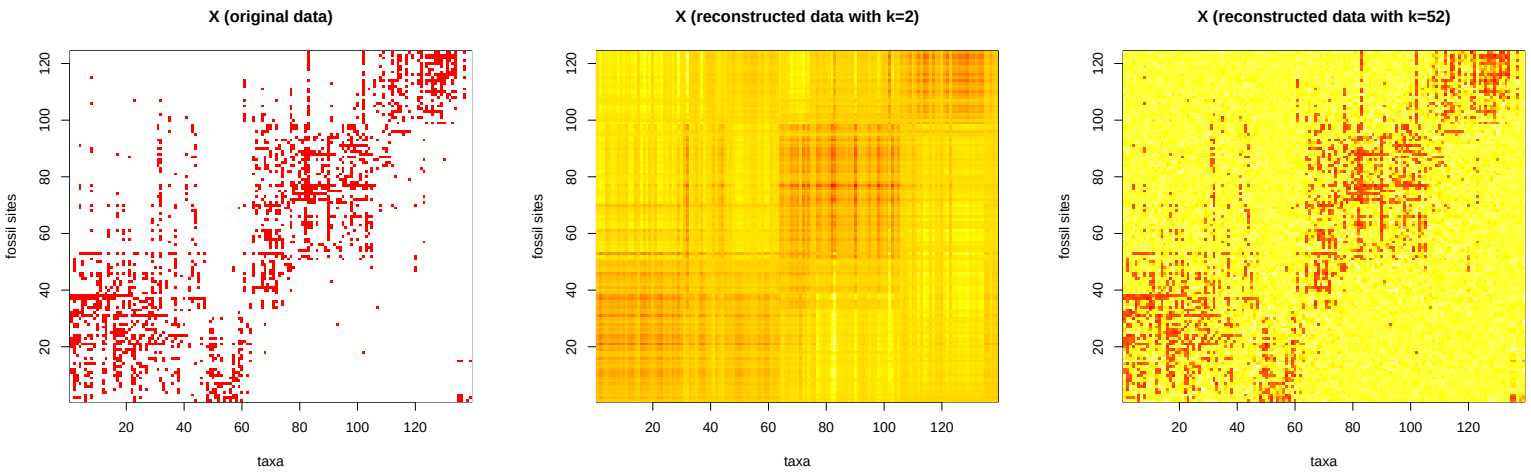






Example: Fossils

- Large European land mammals: 124 fossil find sites (dated 23–2 million years old), 139 taxa
- Reconstruction of site vectors given PCA taxon representation for different k: $\hat{\mathbf{y}} = W\hat{\mathbf{z}} = WW^T\mathbf{y}$, or $\hat{\mathbf{x}} = WW^T(\mathbf{x} - \mathbf{m}) + \mathbf{m}$.



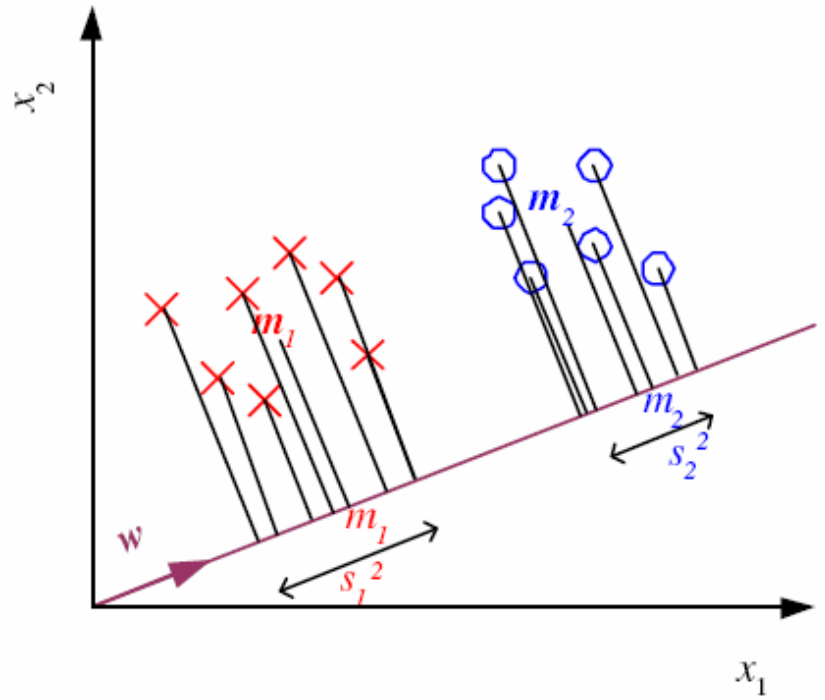
Outline

Linear Discriminant Analysis (LDA)

- PCA is unsupervised method (class information is not usually used).
- Linear Discriminant Analysis (LDA) is supervised method for dimensionality reduction in classification problems.
- As PCA, LDA can be accomplished with standard matrix algebra (eigenvalue decompositions etc.). This makes it relatively simple and useful.
- PCA is a good general purpose dimensionality reduction method, LDA is a good alternative if we want to optimize the separability of classes in a specific classification task, and are happy with dimensionality of less than the number of classes ($k < K$).

Linear Discriminant Analysis (LDA)

- Find a low-dimensional space such that when $\boldsymbol{x}$ is projected, classes are well-separated.
- Find $\boldsymbol{w}$ that maximizes



$$J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

$$m_1 = \frac{\sum_t \boldsymbol{w}^T \boldsymbol{x}^t r^t}{\sum_t r^t} \qquad s_1^2 = \sum_t \left(\boldsymbol{w}^T \boldsymbol{x}^t - m_1\right)^2 r^t$$

Linear Discriminant Analysis (LDA)

- More formally: data $\mathcal{X} = \{(\mathbf{r}^t, \mathbf{x}^t)\}_{t=1}^N$, where r_i^t is one if $\mathbf{x}^t$ is in class i, zero otherwise, and $\mathbf{x}^t \in \mathbb{R}^d$.
- Within-class scatter: $S_W = \sum_{i=1}^K S_i$, where $S_i = \sum_t r_i^t (\mathbf{x}^t - \mathbf{m}_i)(\mathbf{x}^t - \mathbf{m}_i)^T$.
- Between-class scatter: $S_B = \sum_{i=1}^K N_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$, where $N_i = \sum_t r_i^t$. $(\mathrm{rank}(S_B) < K)$
- $k = 1$: find $\mathbf{w} \in \mathbb{R}^d$ that maximizes Fisher's discriminant

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}.$$

- $K > k > 1$: find $W \in \mathbb{R}^{d \times k}$ that maximizes Fisher's discriminant

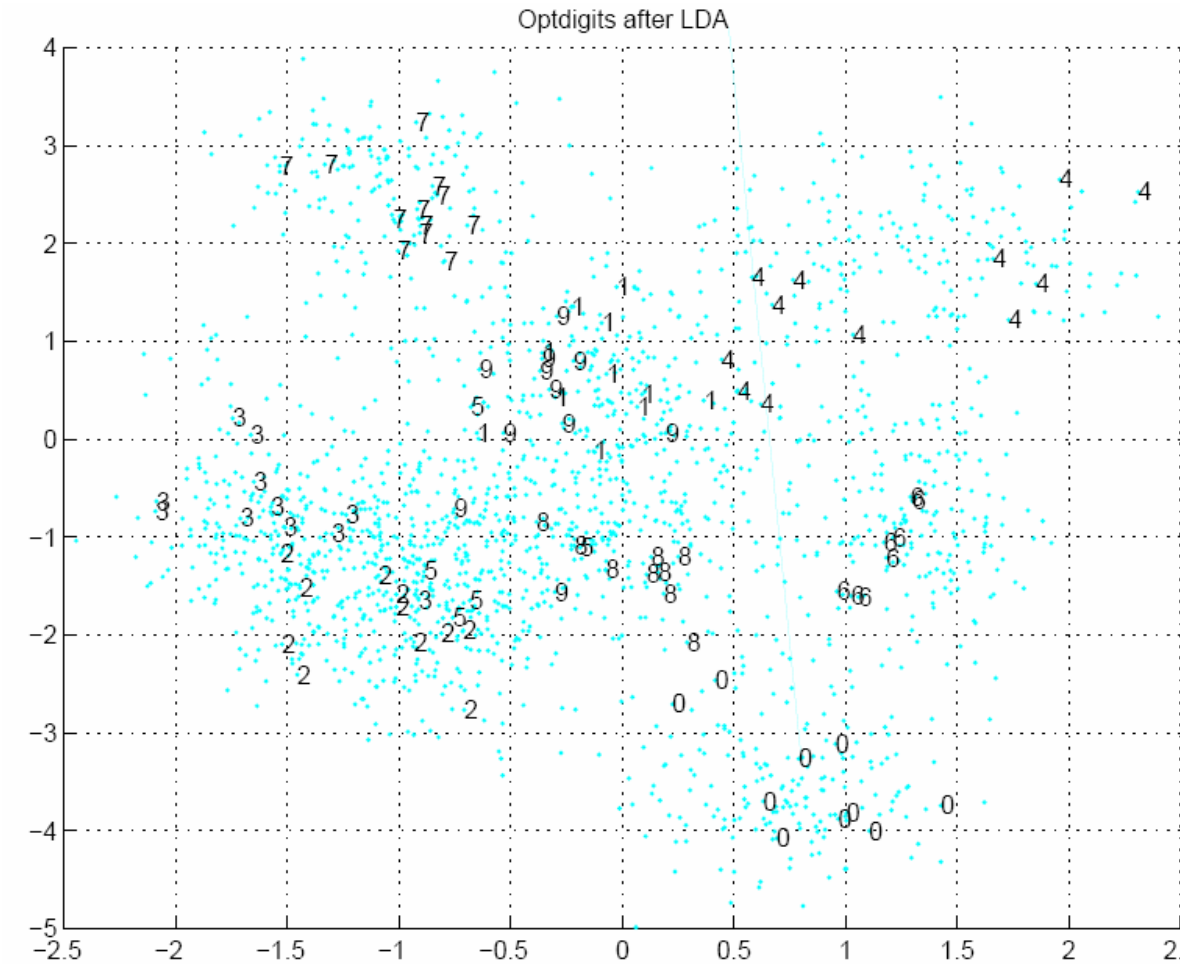
$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}.$$

- The projection from $\mathbb{R}^d$ to $\mathbb{R}^k$ is given by $\hat{\mathbf{z}} = W^T(\mathbf{x} - \mathbf{m})$.

- Find $W \in \mathbb{R}^{d \times k}$ that maximizes Fisher's discriminant

$$J(W) = \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}.$$

- Write $V = S_W^{1/2} W \in \mathbb{R}^{d \times k}$, where $S_W^{1/2}$ is a matrix such that $S_W^{1/2} S_W^{1/2} = S_W$: $J(V) = \left|V^T S_W^{-1/2} S_B S_W^{-1/2} V\right| / \left|V^T V\right|$.
- Determinant is a product of eigenvalues. To maximize $J(V)$ V must contain the k largest eigenvectors of $S_W^{-1/2} S_B S_W^{-1/2}$ (like in PCA!): $V^T S_W^{-1/2} S_B S_W^{-1/2} V = D \Leftrightarrow$ $W S_W^{-1/2} S_W^{-1/2} S_B S_W^{-1/2} S_W^{1/2} W = D \Leftrightarrow W^T S_W^{-1} S_B W = D$.
- $\Rightarrow$ LDA is the k largest eigenvector decomposition of $S_W^{-1} S_B$ (like PCA is of covariance matrix).
- At most $K - 1$ non-zero eigenvalues, that is, one should choose $k < K$.

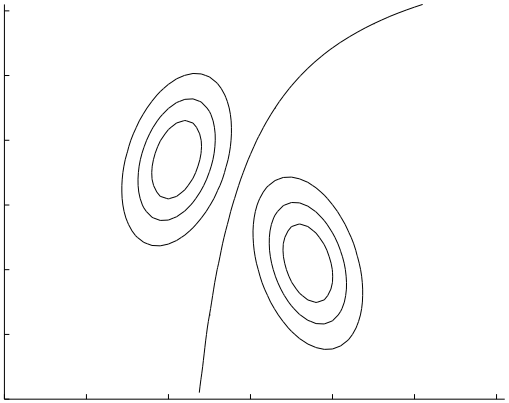


Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press (V1.1)

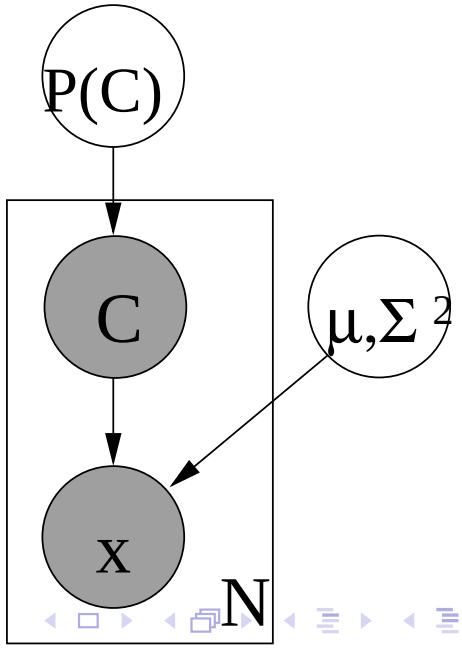
Outline

Mixture densities

- $p(\mathbf{x}) = \sum_{i=1}^{k} p(\mathbf{x} \mid C_i) p(C_i)$
- Classification: labels $\mathbf{r}^t$ are known in training data. Task: predict $\mathbf{r}$ for new data vectors $\mathbf{x}$
- Clustering: data is unlabeled, that is, $\mathbf{r}^t$ are unknown. Task: assign a cluster label $\mathbf{r}$ for new data vectors $\mathbf{x}$.
- Gaussian mixture model:



From Figure 5.3 of Alpaydin (2004).



Classes vs. Clusters

- Supervised: $\mathcal{X} = \{ \boldsymbol{x}^t, \boldsymbol{r}^t \}_t$
- Classes C_i i=1,...,K

$$p(\boldsymbol{x}) = \sum_{i=1}^{K} p(\boldsymbol{x} \mid C_i) P(C_i)$$

where $p(\boldsymbol{x} \mid C_i) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$

- $\Phi = \{P(C_i), \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N} \quad \boldsymbol{m}_i = \frac{\sum_t r_i^t \boldsymbol{x}^t}{\sum_t r_i^t}$$

$$\mathbf{S}_i = \frac{\sum_t r_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T}{\sum_t r_i^t}$$

- Unsupervised : $\mathcal{X} = \{ \boldsymbol{x}^t \}_t$
- Clusters $\mathcal{G}_i$ i=1,...,k

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} p(\boldsymbol{x} \mid \mathcal{G}_i) P(\mathcal{G}_i)$$

where $p(\boldsymbol{x} \mid \mathcal{G}_i) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$

- $\Phi = \{P(\mathcal{G}_i), \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{k}$

Labels, $\boldsymbol{r}^t_i$?

Outline

k-means Clustering

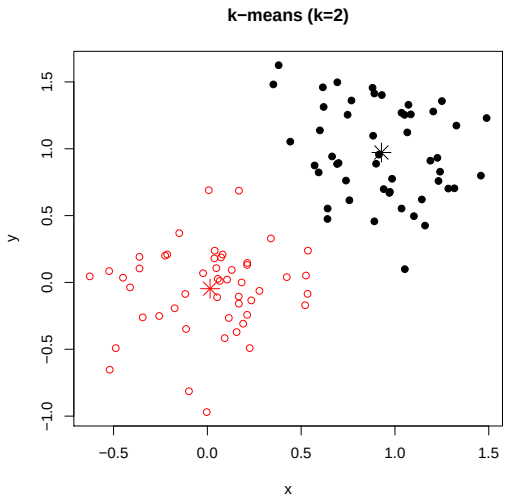
- The simplest Bayesian classifier was nearest mean classifier: classify a data vector to class which has a nearest mean.
- k-means clustering: find k prototype vectors $\mathbf{m}_i$ ("means") which best represent data.
- Error function:

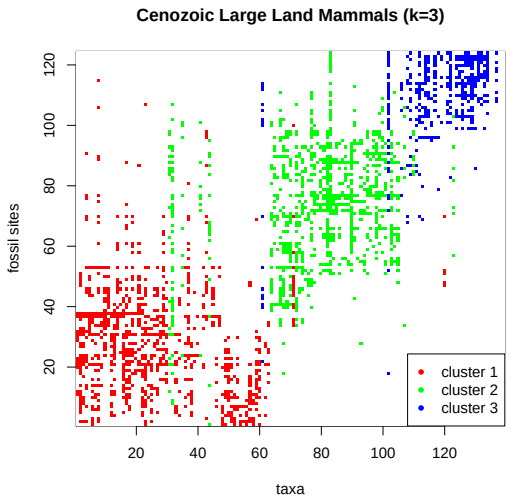
$$\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X}) = \sum_{t=1}^{N} \min_i \left|\left|\mathbf{x}^t - \mathbf{m}_i\right|\right|^2.$$

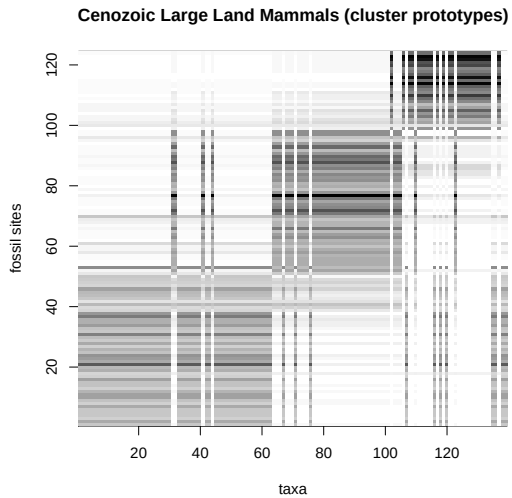
- Task: find prototype vectors $\mathbf{m}_i$ such that error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$ is minimized.
- No direct probabilistic interpretation. Can be viewed as *approximation* of the Bayesian nearest mean classifier where data vector belongs to a class/cluster with probability 0 or 1 only.

k-means Clustering

- The vectors are assigned to the nearest means.
- In R: cl <- kmeans(t(X),centers=3)







k-means Clustering

- Compression: a real vector (image etc.) can be represented with a number in $\{1, \ldots, k\}$.
- Dimensionality reduction: one can use cluster indexes instead of the real vectors to train a classifier etc.
- Interpretation of the data: clusters have often a meaning. Taxa from various time periods, customer segments, etc.
- Labeling of data: cluster indexes may be used as class labels.

k-means Clustering

Example: image compression

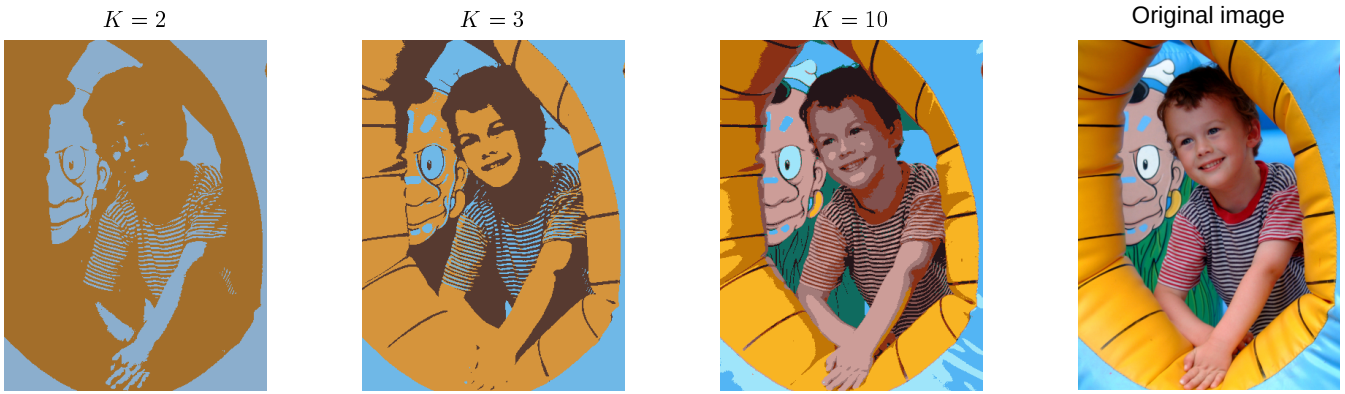


Figure 9.3 of Bishop (2006).

- Data set is the set of pixels.
- Each pixel is a vector in three-dimensional RGB space.
- K-means is applied to the data set of pixels of an image.
- The compressed representation is then the prototype vectors, and cluster index for each pixel.

k-means Clustering
Lloyd's algorithm

- Lloyd's algorithm: the most famous algorithm to minimize the k-means cost function. Easy to understand and implement.
- Sensitive to initialization: should be run on several random initializations and choose the result with the smallest cost.
- In practice one should consider some more advanced method (type help(kmeans) in R for some

Initialize $\boldsymbol{m}_i, i = 1, \ldots, k$, for example, to k random $\boldsymbol{x}^t$
Repeat
  For all $\boldsymbol{x}^t \in \mathcal{X}$

$$b_i^t \leftarrow \begin{cases} 1 & \text{if } \|\boldsymbol{x}^t - \boldsymbol{m}_i\| = \min_j \|\boldsymbol{x}^t - \boldsymbol{m}_j\| \\ 0 & \text{otherwise} \end{cases}$$

  For all $\boldsymbol{m}_i, i = 1, \ldots, k$

$$\boldsymbol{m}_i \leftarrow \sum_t b_i^t \boldsymbol{x}^t / \sum_t b_i^t$$

Until $\boldsymbol{m}_i$ converge

Figure 7.3: k-means algorithm. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

k-means Clustering

Lloyd's algorithm

Initialize $\mathbf{m}_i$, $i = 1, \ldots, k$, randomly.

repeat

for all $t \in \{1, \ldots, N\}$ **do** {E step}

$$b_i^t \leftarrow \begin{cases} 1 & , \quad i = \arg\min_i ||\mathbf{x}^t - \mathbf{m}_i|| \\ 0 & , \quad \text{otherwise} \end{cases}$$

end for

for all $i \in \{1, \ldots, k\}$ **do** {M step}

$$\mathbf{m}_i \leftarrow \frac{\sum_t b_i^t \mathbf{x}^t}{\sum_t b_i^t}$$

end for

until the error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$ does not change

k-means Clustering
Lloyd's algorithm

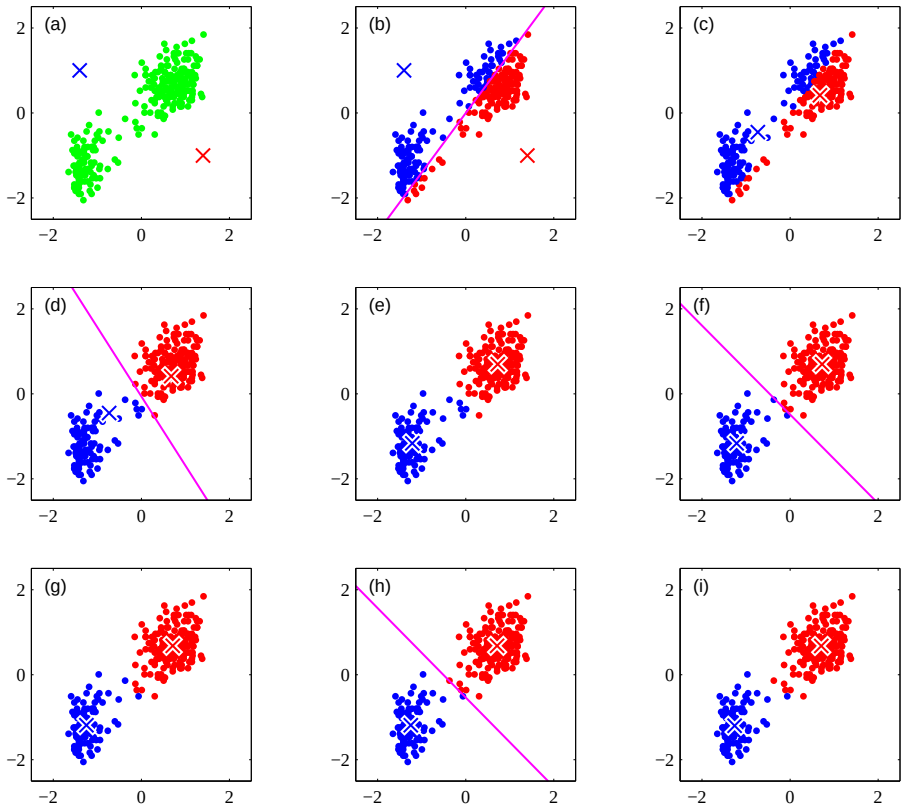


Figure 9.1 of Bishop (2006)

k-means Clustering
Lloyd's algorithm

Observations:

- Iteration cannot increase the error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$.
- There are finite number, k^N, of possible clusterings.
- It follows that the algorithm always stops after a finite time. (It can take no more than k^N steps.)
- Usually k-means is however relatively fast. "In practice the number of iterations is generally much less than the number of points." (Duda & Hart & Stork, 2000)
- Worst-case running time with really bad data and really bad initialization is however $2^{\Omega(\sqrt{N})}$ — luckily this usually does not happen in real life (David A, Vassilivitskii S (2006) How slow is the k-means method? In Proc 22nd SCG.)

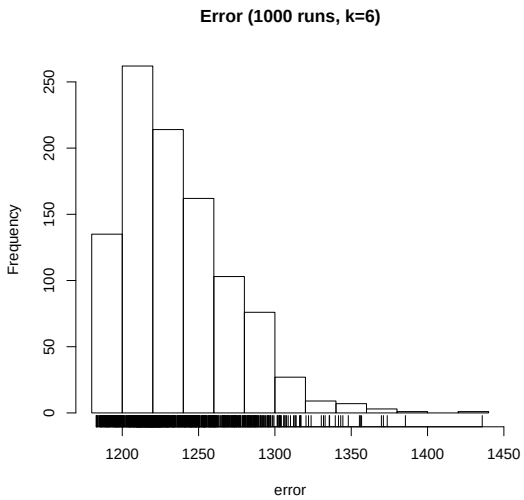
k-means Clustering
Lloyd's algorithm

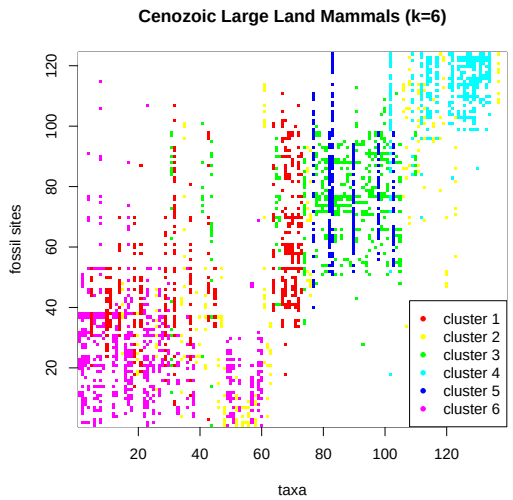
Observations:

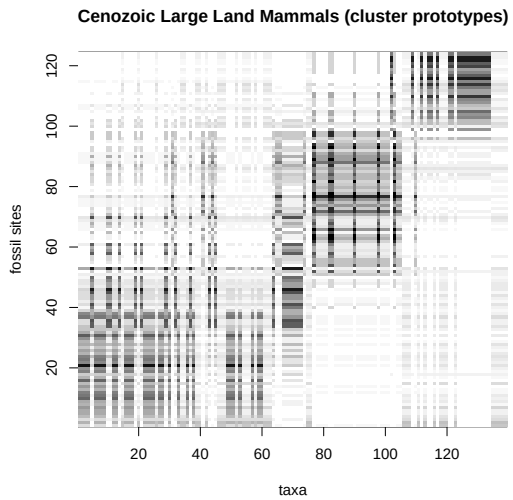
- The result can in the worst case be really bad.
- Example:
  - Four data vectors ($N = 4$) from $\mathbb{R}^d$ in $\mathcal{X}$: $\mathbf{x}^1 = (0, 0, \ldots, 0)^T$, $\mathbf{x}^2 = (1, 0, \ldots, 0)^T$, $\mathbf{x}^3 = (0, 1, \ldots, 1)^T$ and $\mathbf{x}^4 = (1, 1, \ldots, 1)^T$.
  - Optimal clustering into two ($k = 2$) is given by the prototype vectors $\mathbf{m}_1 = (0.5, 0, \ldots, 0)^T$ and $\mathbf{m}_2 = (0.5, 1, \ldots, 1)^T$, error being $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X}) = 1$.
  - Lloyd's algorithm can however converge also to $\mathbf{m}_1 = (0, 0.5, \ldots, 0.5)^T$ and $\mathbf{m}_2 = (1, 0.5, \ldots, 0.5)^T$, error being $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X}) = d - 1$. (Check that iteration stops here!)

k-means Clustering
Lloyd's algorithm

- Example: cluster taxa into $k = 6$ clusters 1000 times with Lloyd's algorithm.
- The error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$ is different for different runs!
- You should try several random initializations, and choose the solution with smallest error.
- For a cool initialization see Arthur D, Vassilivitskii S (2006) k-means++: The Advantages of Careful Seeding.







Outline

EM Algorithm

- Expectation-Maximization algorithm (EM): soft cluster assignments
- Probabilistic interpretation

EM Algorithm

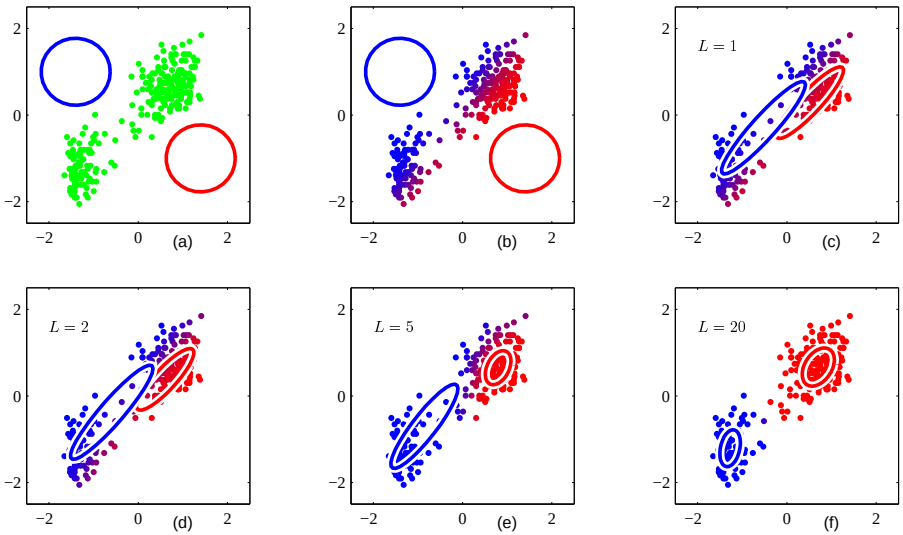
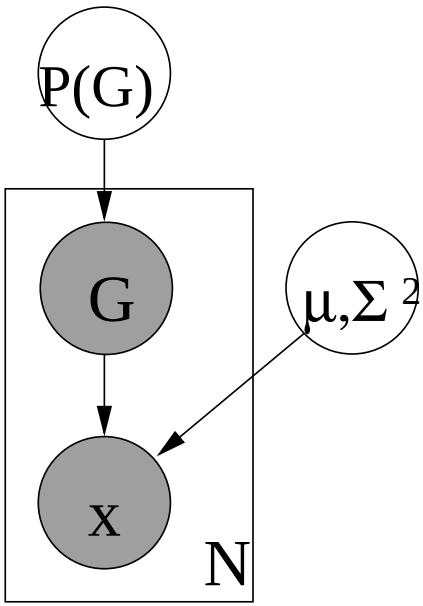


Figure 9.8 of Bishop (2006)

- EM algorithm is like k-means, except cluster assignments are "soft": each data point is a member of a given cluster with certain probability.
- $b_i^t \in \{0, 1\} \longrightarrow h_i^t \in [0, 1]$.

EM Algorithm

- Find maximum likelihood solution of the mixture model $\mathcal{L} = \log \prod_{t=1}^{N} p(\mathbf{x}^t \mid \theta)$, where the parameters θ are μ_i, Σ_i and $\pi_i = P(G_i)$.
- Maximum likelihood solution is found by the EM algorithm (which is essentially generalization of the Lloyd's algorithm to soft cluster memberships)
- Idea: iteratively find the membership weights of each data vector in clusters, and the parameter values. Continue until convergence.
- End result is intuitive.



EM Algorithm

Example: soft Gaussian mixture, fixed shared diagonal covariance matrix $\Sigma_i = s^2\mathbf{1}$

Initialize $\mathbf{m}_i$ and π_i, $i = 1, \ldots, k$, randomly.
repeat
  for all $t \in \{1, \ldots, N\}$ **do** {E step}

$$h_i^t \leftarrow \frac{\pi_i \exp\left[-\frac{1}{2s^2}\left|\left|\mathbf{x}^t - \mathbf{m}_i\right|\right|^2\right]}{\sum_j \pi_j \exp\left[-\frac{1}{2s^2}\left|\left|\mathbf{x}^t - \mathbf{m}_j\right|\right|^2\right]}$$

  end for
  for all $i \in \{1, \ldots, k\}$ **do** {M step}

$$\mathbf{m}_i \leftarrow \frac{\sum_t h_i^t \mathbf{x}^t}{\sum_t h_i^t}$$

$$\pi_i \leftarrow \frac{\sum_t h_i^t}{N}$$

  end for
until convergence

EM Algorithm

- For derivation, see Alpaydin (2004), section 7.4 (pages 139–144); for an alternative derivation, see Bishop (2006), section 9.4 (pages 450–455). A sketch of follows.
- Task: find an ML solution of a likelihood function given by $p(\mathbf{X} \mid \theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \mid \theta)$.

$$\begin{aligned}\sum_t \log p(\mathbf{x}^t \mid \theta) &\geq \sum_t \log p(\mathbf{x}^t \mid \theta) - \sum_t KL(h_i^t \mid\mid p(\mathbf{z}^t \mid \mathbf{x}^t, \theta)) \\ &= \sum_t \sum_i h_i^t \log p(\mathbf{x}^t, \mathbf{z}^t \mid \theta) + \sum_t H(h_i^t),\end{aligned}$$

where we have used the Kullback-Leibler (KL) divergence $KL(q(i) \mid\mid p(i)) = \sum_i q(i) \log (q(i)/p(i))$. KL divergence is always non-negative and it vanishes only when the distributions q and p are equal. The entropy is given by $H(q(i)) = -\sum_i q(i) \log q(i)$.

EM Algorithm

- Expectation step (E Step): find h_i^t by minimizing the KL divergence.
- Maximization step (M Step): find θ by maximizing the expectation.

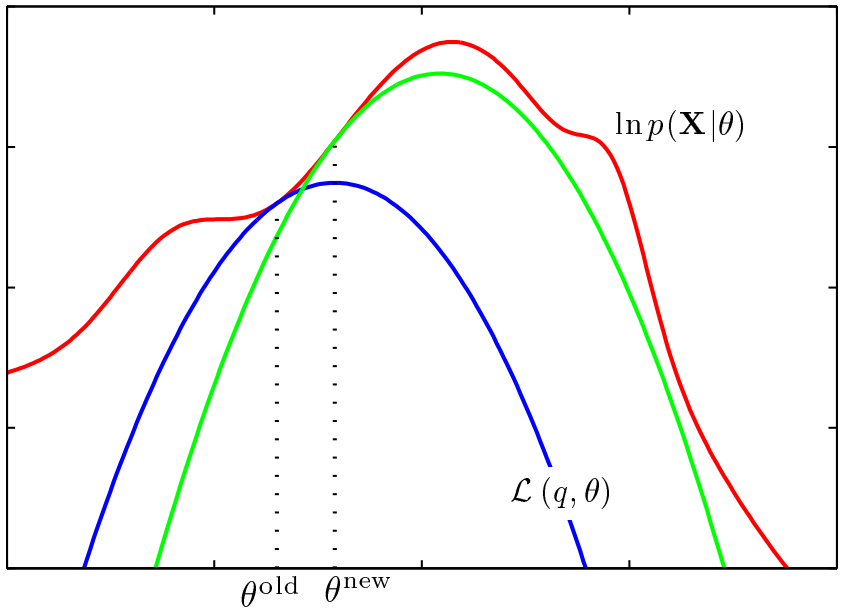


Figure 9.14 of Bishop (2006)