Introduction to Statistical Learning and Machine Learning

Chap 10 – Unsupervised Learning and Dimension Reduction

Yanwei Fu

School of Data Science, Fudan University

Unsupervised Learning

- Supervised learning algorithms have a clear goal: produce desired outputs for given inputs
- Goal of unsupervised learning algorithms (no explicit feedback whether outputs of system are correct) less clear:
  - Reduce dimensionality
  - Find clusters
  - Model data density
  - Find hidden causes
- Key utility
  - Compress data
  - Detect outliers
  - Facilitate other learning

Unsupervised vs Supervised Learning:

- Most of this course focuses on *supervised learning* methods such as regression and classification.
- In that setting we observe both a set of features $X_1, X_2, \ldots, X_p$ for each object, as well as a response or outcome variable Y. The goal is then to predict Y using $X_1, X_2, \ldots, X_p$.
- Here we instead focus on *unsupervised learning*, we where observe only the features $X_1, X_2, \ldots, X_p$. We are not interested in prediction, because we do not have an associated response variable Y.

The Goals of Unsupervised Learning

- Primary problems, approaches in unsupervised learning fall into three classes:

  1. Dimensionality reduction: represent each input case using a small number of variables (e.g., principal components analysis, factor analysis, independent components analysis)
  2. Clustering: represent each input case using a prototype example (e.g., k-means, mixture models)
  3. Density estimation: estimating the probability distribution over the data space

The Challenge of Unsupervised Learning

- Unsupervised learning is more subjective than supervised learning, as there is no simple goal for the analysis, such as prediction of a response.
- But techniques for unsupervised learning are of growing importance in a number of fields:
  - subgroups of breast cancer patients grouped by their gene expression measurements,
  - groups of shoppers characterized by their browsing and purchase histories,
  - movies grouped by the ratings assigned by movie viewers.

Another advantage

- It is often easier to obtain *unlabeled data* — from a lab instrument or a computer — than *labeled data*, which can require human intervention.
- For example it is difficult to automatically assess the overall sentiment of a movie review: is it favorable or not?

- PCA produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated.
- Apart from producing derived variables for use in supervised learning problems, PCA also serves as a tool for data visualization.

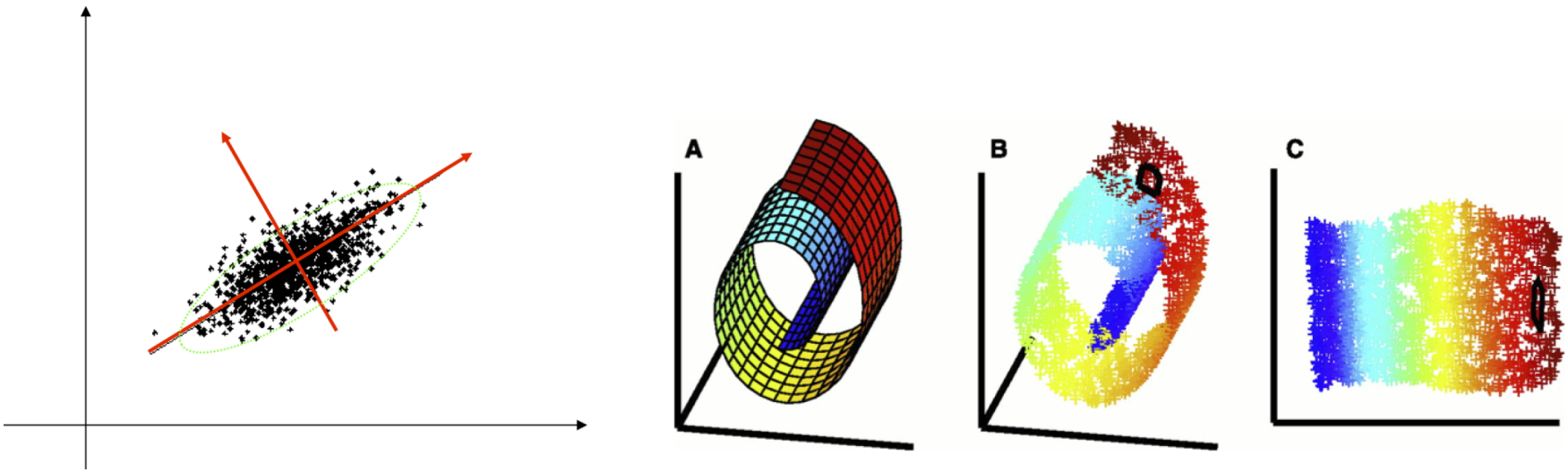


Figure: Find an affine space to approximate data variation in PCA/MDS. (b) Swiss Roll data distributed on a nonlinear 2-D submanifold in Euclidean space R3. Our purpose is to capture an intrinsic coordinate system describing the submanifold.

Principal Components Analysis: details

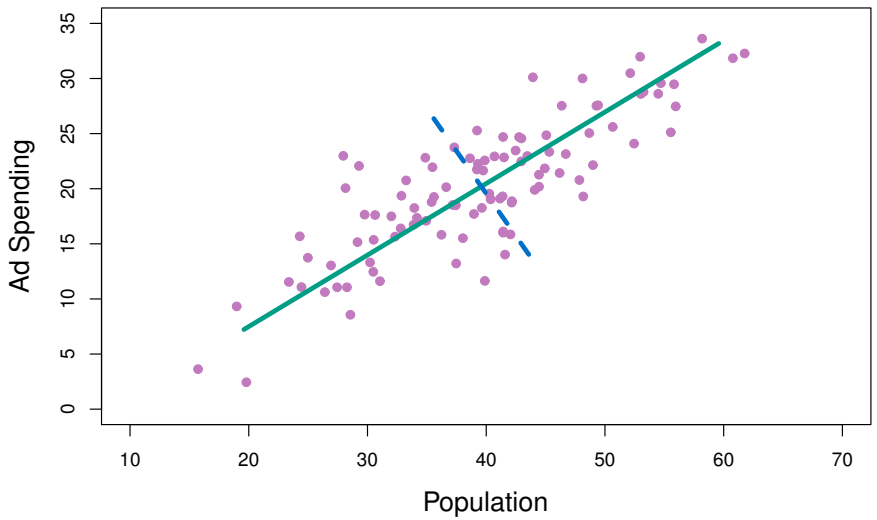
- The *first principal component* of a set of features $X_1, X_2, \ldots, X_p$ is the normalized linear combination of the features

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \ldots + \phi_{p1} X_p$$

that has the largest variance. By *normalized*, we mean that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.

- We refer to the elements $\phi_{11}, \ldots, \phi_{p1}$ as the loadings of the first principal component; together, the loadings make up the principal component loading vector, $\phi_1 = (\phi_{11}\ \phi_{21}\ \ldots\ \phi_{p1})^T$.
- We constrain the loadings so that their sum of squares is equal to one, since otherwise setting these elements to be arbitrarily large in absolute value could result in an arbitrarily large variance.

Principal Components Analysis: details



The population size (pop) and ad spending (ad) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component direction, and the blue dashed line indicates the second principal component direction.

Principal Components Analysis: details

- Suppose we have a $n \times p$ data set $\mathbf{X}$. Since we are only interested in variance, we assume that each of the variables in $\mathbf{X}$ has been centered to have mean zero (that is, the column means of $\mathbf{X}$ are zero).
- We then look for the linear combination of the sample feature values of the form

$$z_{i1} = \phi_{11} x_{i1} + \phi_{21} x_{i2} + \ldots + \phi_{p1} x_{ip} \tag{1}$$

  for $i = 1, \ldots, n$ that has largest sample variance, subject to the constraint that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.
- Since each of the x_{ij} has mean zero, then so does z_{i1} (for any values of ϕ_{j1}). Hence the sample variance of the z_{i1} can be written as $\frac{1}{n} \sum_{i=1}^{n} z_{i1}^2$.

Principal Components Analysis: details

- Plugging in (1) the first principal component loading vector solves the optimization problem

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2 \text{ subject to } \sum_{j=1}^{p}\phi_{j1}^2 = 1.$$

- This problem can be solved via a singular-value decomposition of the matrix $\mathbf{X}$, a standard technique in linear algebra.
- We refer to Z_1 as the first principal component, with realized values $z_{11}, \ldots, z_{n1}$

PCA: example

- The loading vector ϕ_1 with elements $\phi_{11}, \phi_{21}, \ldots, \phi_{p1}$ defines a direction in feature space along which the data vary the most.
- If we project the n data points $x_1, \ldots, x_n$ onto this direction, the projected values are the principal component scores $z_{11}, \ldots, z_{n1}$ themselves.

- The loading vector ϕ_1 with elements $\phi_{11}, \phi_{21}, \ldots, \phi_{p1}$ defines a direction in feature space along which the data vary the most.
- If we project the n data points $x_1, \ldots, x_n$ onto this direction, the projected values are the principal component scores $z_{11}, \ldots, z_{n1}$ themselves.

- The second principal component is the linear combination of $X_1, \ldots, X_p$ that has maximal variance among all linear combinations that are *uncorrelated* with Z_1.
- The second principal component scores $z_{12}, z_{22}, \ldots, z_{n2}$ take the form

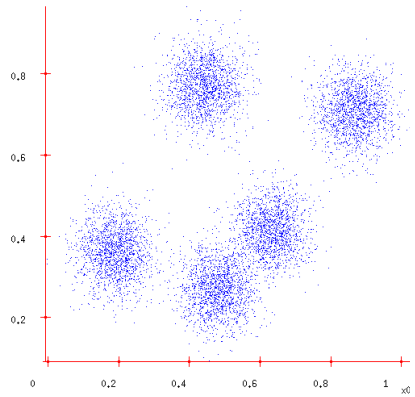
$$z_{i2} = \phi_{12} x_{i1} + \phi_{22} x_{i2} + \ldots + \phi_{p2} x_{ip},$$

where ϕ_2 is the second principal component loading vector, with elements $\phi_{12}, \phi_{22}, \ldots, \phi_{p2}$.

Further principal components

- It turns out that constraining Z_2 to be uncorrelated with Z_1 is equivalent to constraining the direction ϕ_2 to be orthogonal (perpendicular) to the direction ϕ_1. And so on.
- The principal component directions ϕ_1, ϕ_2, $\phi_3, \ldots$ are the ordered sequence of right singular vectors of the matrix $\mathbf{X}$, and the variances of the components are $\frac{1}{n}$ times the squares of the singular values. There are at most $\min(n-1, p)$ principal components.

- Grouping N examples into K clusters one of canonical problems in unsupervised learning



- Motivations: prediction; lossy compression; outlier detection
- We assume that the data was generated from a number of different classes. The aim is to cluster data from the same class together.
  - How many classes?
  - Why not put each datapoint into a separate class?
- What is the objective function that is optimized by sensible clusterings?

- *Clustering* refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set.
- We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other,
- It make this concrete, we must define what it means for two or more observations to be *similar* or *different*.
- Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied.

- PCA looks for a low-dimensional representation of the observations that explains a good fraction of the variance.
- Clustering looks for homogeneous subgroups among the observations.

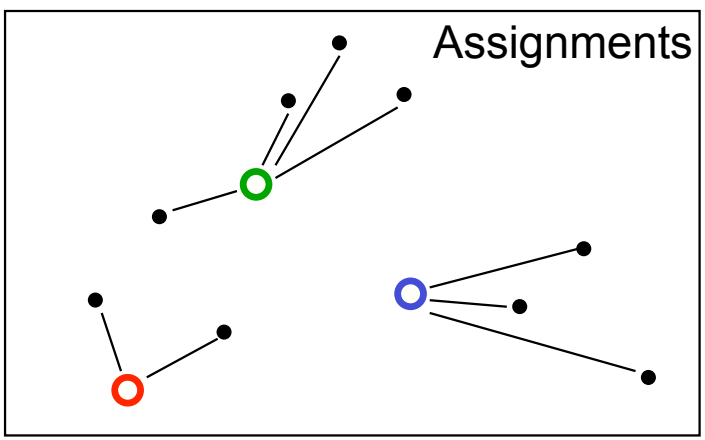
PCA vs Clustering

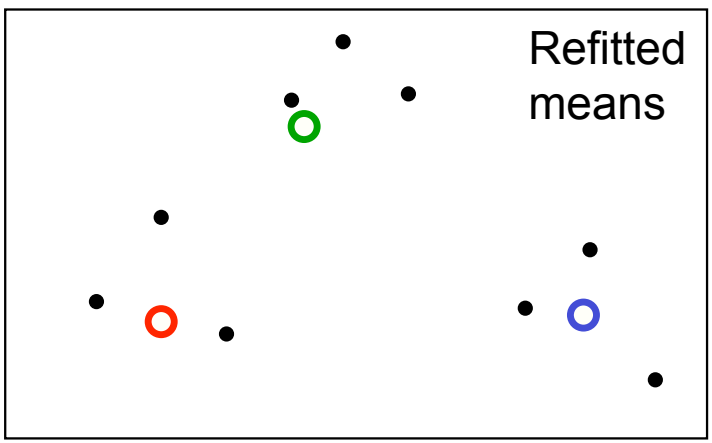
K-means clustering method: we seek to partition the observations into a pre-specied number of clusters. We introduce **K-means** and **soft-Kmeans** here.

hierarchical clustering method: we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a dendrogram, that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to n.

The K-means clustering algorithm

- Assume the data lives in a Euclidean space.
- Assume we want k classes/patterns
- Initialization: randomly located cluster centers
- The algorithm alternates between two steps:
  - Assignment step: Assign each datapoint to the closest cluster.
  - Refitting step: Move each cluster center to the center of gravity of the data assigned to it.





- Objective: minimize sum squared distance of datapoints to their assigned cluster centers

$$\min_{\{\mathbf{m}\},\{\mathbf{r}\}} E(\{\mathbf{m}\},\{\mathbf{r}\}) = \sum_n \sum_k r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

$$\text{s.t.} \sum_k r_k^{(n)} = 1, \forall n, \quad r_k^{(n)} \in \{0,1\}, \forall k, n$$

- Optimization method is a form of coordinate descent ("block coordinate descent")
  - Fix centers, optimize assignments (choose cluster whose mean is closest)
  - Fix assignments, optimize means (average of assigned datapoints)

- Initialization: Set K means $\{\mathbf{m}_k\}$ to random values
- Assignment: Each datapoint n assigned to nearest mean

$$\hat{k}^n = arg\min_k d(\mathbf{m}_k, \mathbf{x}^{(n)})$$

and Responsibilities (1 of k encoding)

$$r_k^{(n)} = 1 \longleftrightarrow \hat{k}^{(n)} = k$$

- Update: Model parameters, means, are adjusted to match sample means of datapoints they are responsible for:

$$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}$$

- Repeat assignment and update steps until assignments do not change

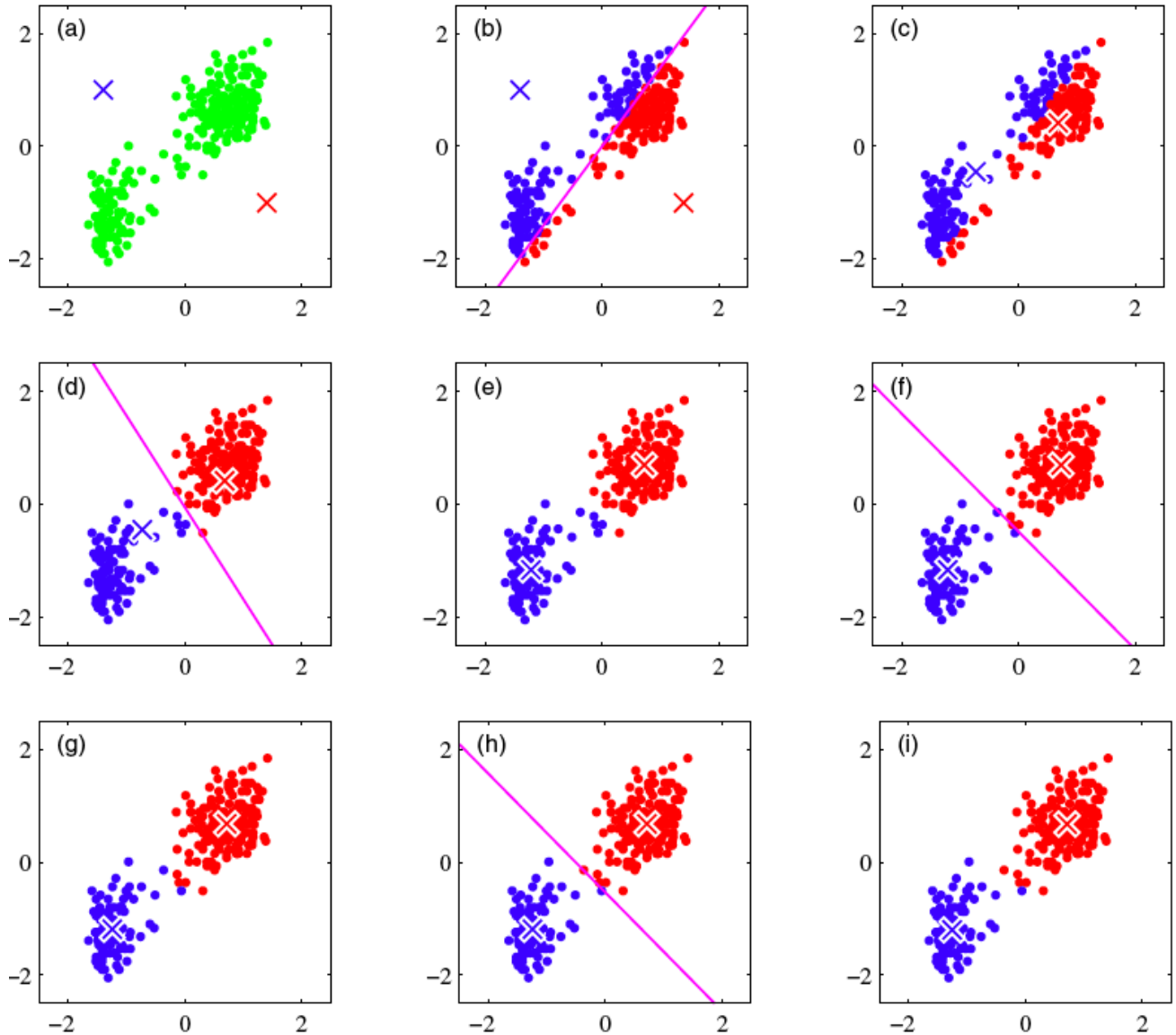


Figure from Bishop.

K-means for Image Segmentation and Vector Quantization:

Figure from Bishop.

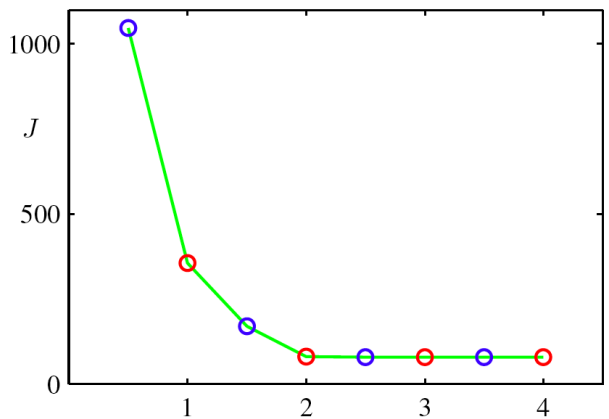
Questions about K-means

- Why does update set $\mathbf{m}_k$ to mean of assigned points?
- Where does distance d come from?
- What if we used a different distance measure?
- How can we choose best distance?
- How to choose K?
- How can we choose between alternative clusterings?
- Will it converge?

Hard cases – unequal spreads, non-circular spreads, inbetween points

Why K-means converges

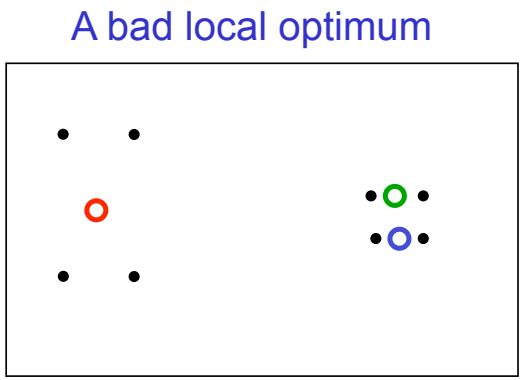
- Whenever an assignment is changed, the sum squared distances of datapoints from their assigned cluster centers is reduced.
- Whenever a cluster center is moved the sum squared distances of the datapoints from their currently assigned cluster centers is reduced.
- Test for convergence: If the assignments do not change in the assignment step, we have converged (to at least a local minimum).



- K-means cost function after each E step (blue) and M step (red). The algorithm has converged after the third M step

Local Minima

- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points
- We could try non-local split-and-merge moves:
  - Simultaneously merge two nearby clusters
  - and split a big cluster into two



Soft K-means Algorithm

- Instead of making hard assignments of datapoints to clusters, we can make soft assignments. One cluster may have a responsibility of .7 for a datapoint and another may have a responsibility of .3.
  - Allows a cluster to use more information about the data in the refitting step.
  - What happens to our convergence guarantee?
  - How do we decide on the soft assignments?

Soft K-means Algorithm

- Initialization: Set K means $\{\mathbf{m}_k\}$ to random values
- Assignment: Each datapoint n given soft "degree of assignment" to each cluster mean k, based on responsibilities

$$r_k^{(n)} = \frac{\exp[-\beta d(\mathbf{m}_k, \mathbf{x}^{(n)})]}{\sum_j \exp[-\beta d(\mathbf{m}_j, \mathbf{x}^{(n)})]}$$

- Update: Model parameters, means, are adjusted to match sample means of datapoints they are responsible for:

$$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}$$

- Repeat assignment and update steps until assignments do not change

- K-means clustering requires us to pre-specify the number of clusters K. This can be a disadvantage (later we discuss strategies for choosing K)
- *Hierarchical clustering* is an alternative approach which does not require that we commit to a particular choice of K.
- In this section, we describe *bottom-up* or *agglomerative* clustering. This is the most common type of hierarchical clustering, and refers to the fact that a dendrogram is built starting from the leaves and combining clusters up to the trunk.

Visual Saliency Attention in a bottom-up way

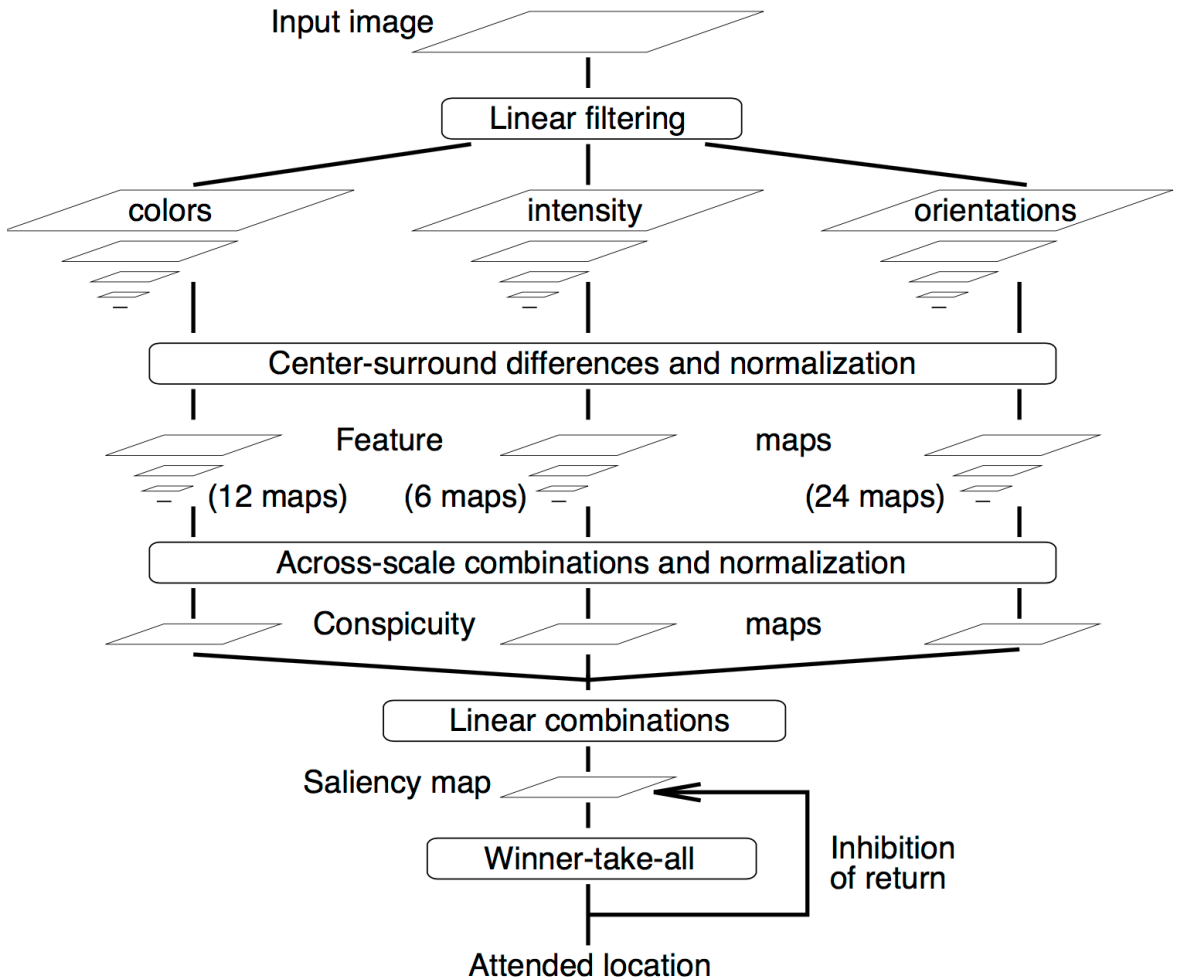
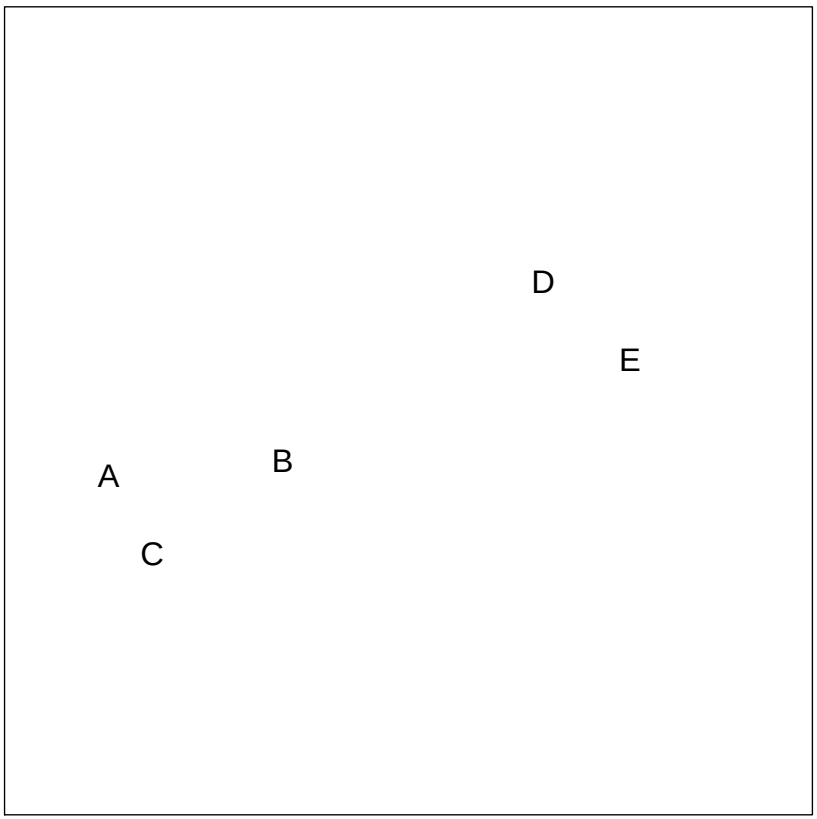


Figure: Visual Saliency in a bottom-up way (itti's " a model of saliency-based visual attention for rapid scene analysis" IEEE TPAMI 1998)

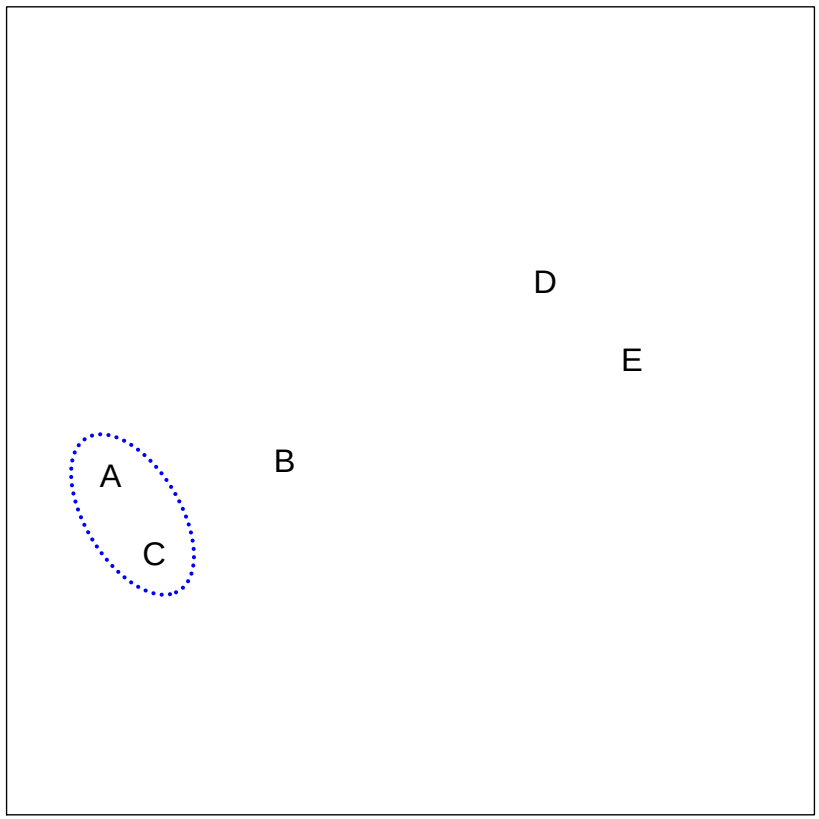
Hierarchical Clustering: the idea

Builds a hierarchy in a "bottom-up" fashion...



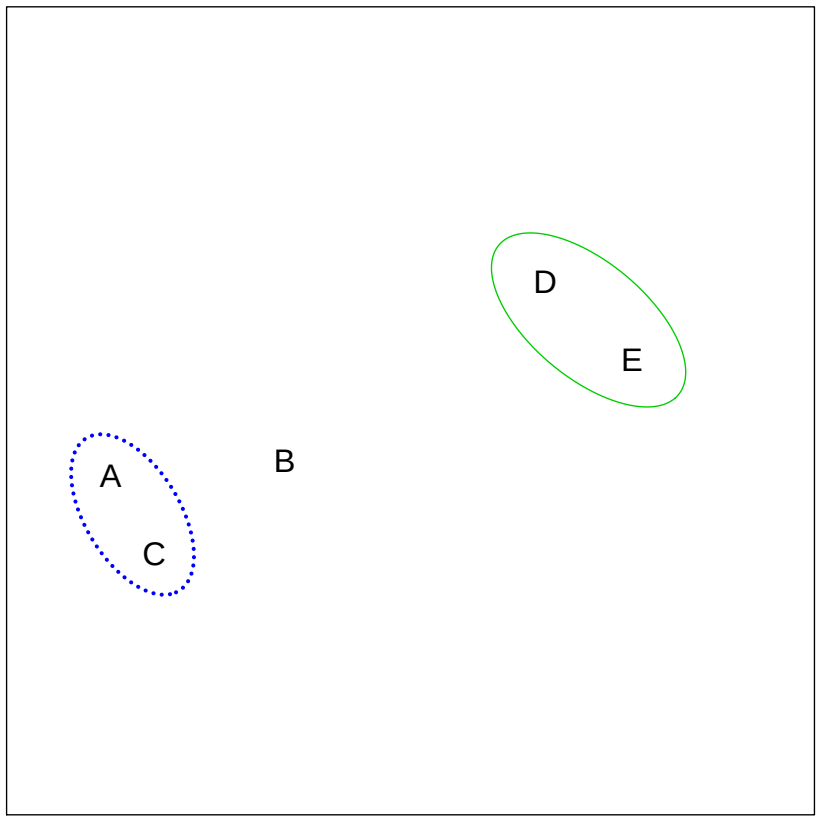
Hierarchical Clustering: the idea

Builds a hierarchy in a "bottom-up" fashion...



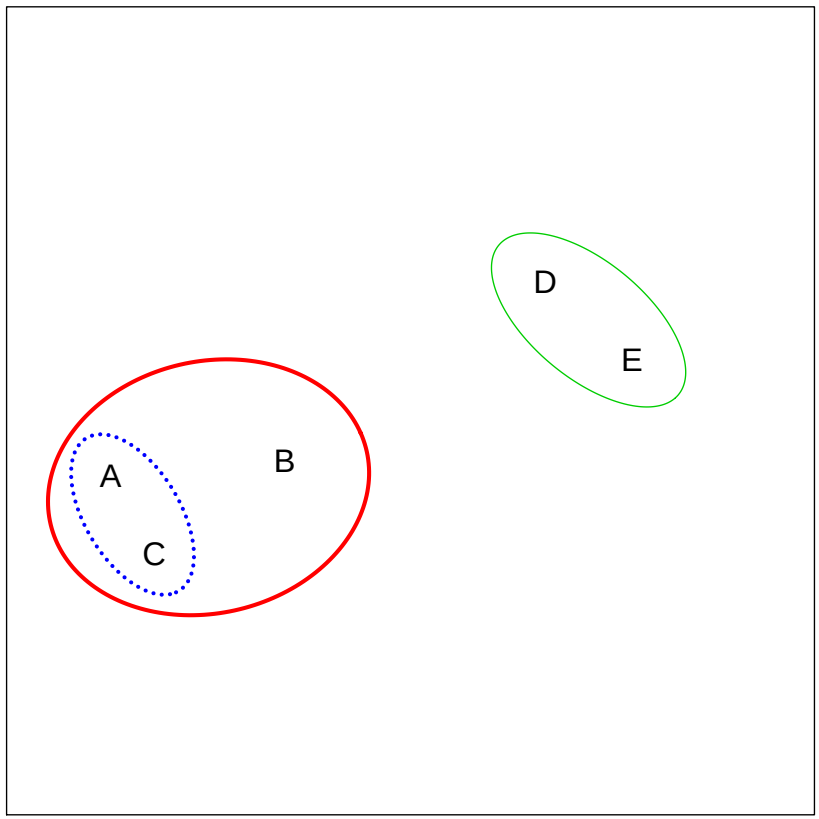
Hierarchical Clustering: the idea

Builds a hierarchy in a "bottom-up" fashion...



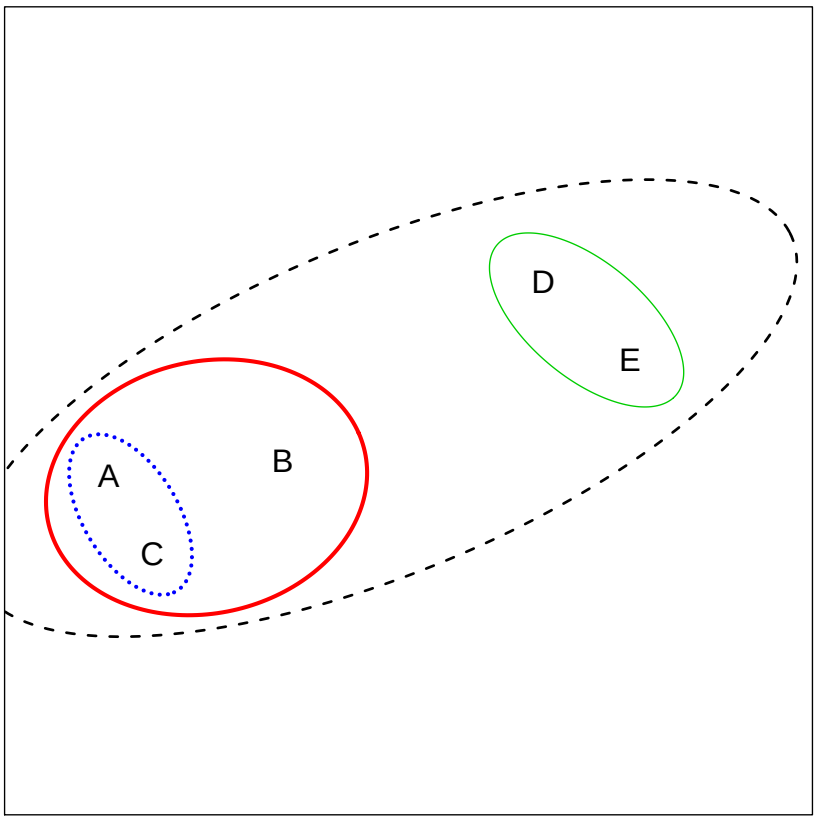
Hierarchical Clustering: the idea

Builds a hierarchy in a "bottom-up" fashion...



Hierarchical Clustering: the idea

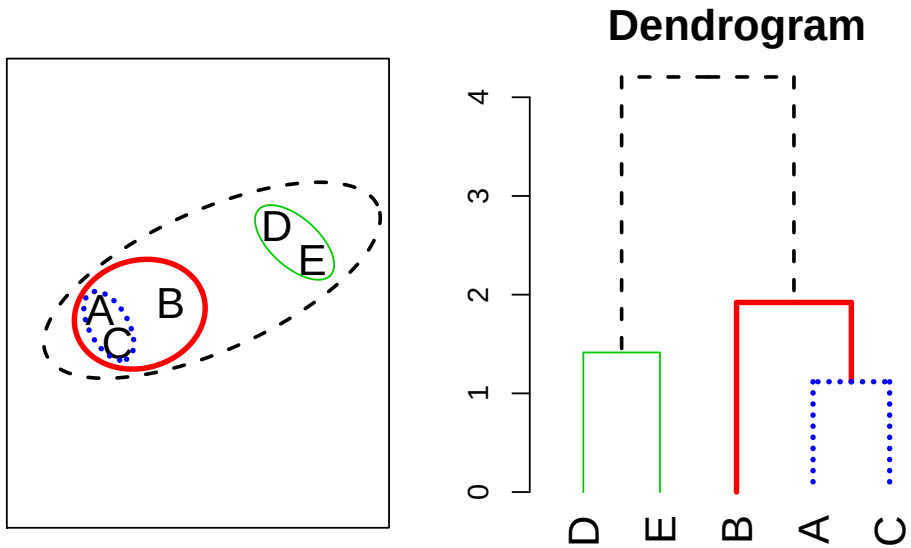
Builds a hierarchy in a "bottom-up" fashion...



Hierarchical Clustering Algorithm

The approach in words:

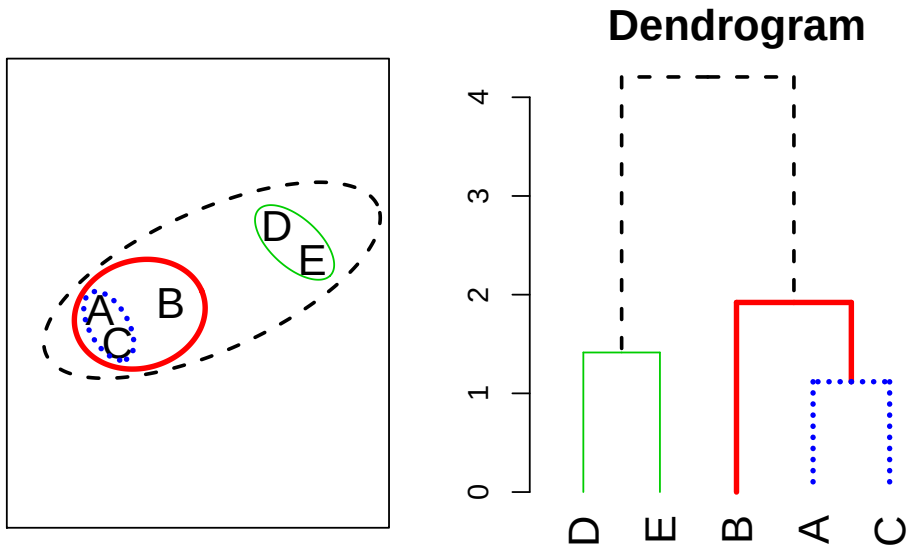
- Start with each point in its own cluster.
- Identify the closest two clusters and merge them.
- Repeat.
- Ends when all points are in a single cluster.



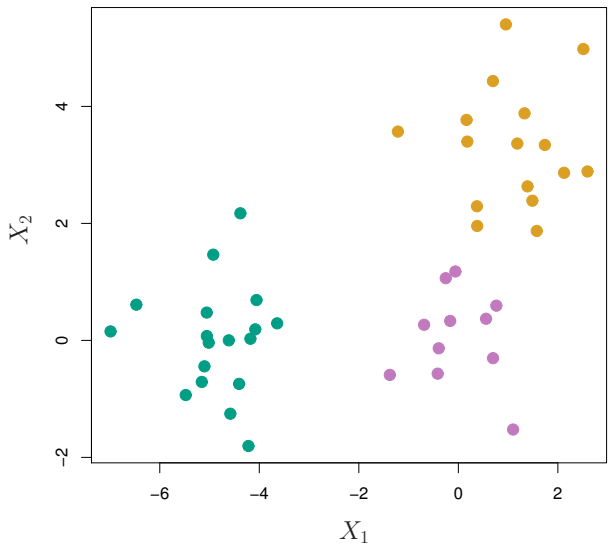
Hierarchical Clustering Algorithm

The approach in words:

- Start with each point in its own cluster.
- Identify the closest two clusters and merge them.
- Repeat.
- Ends when all points are in a single cluster.

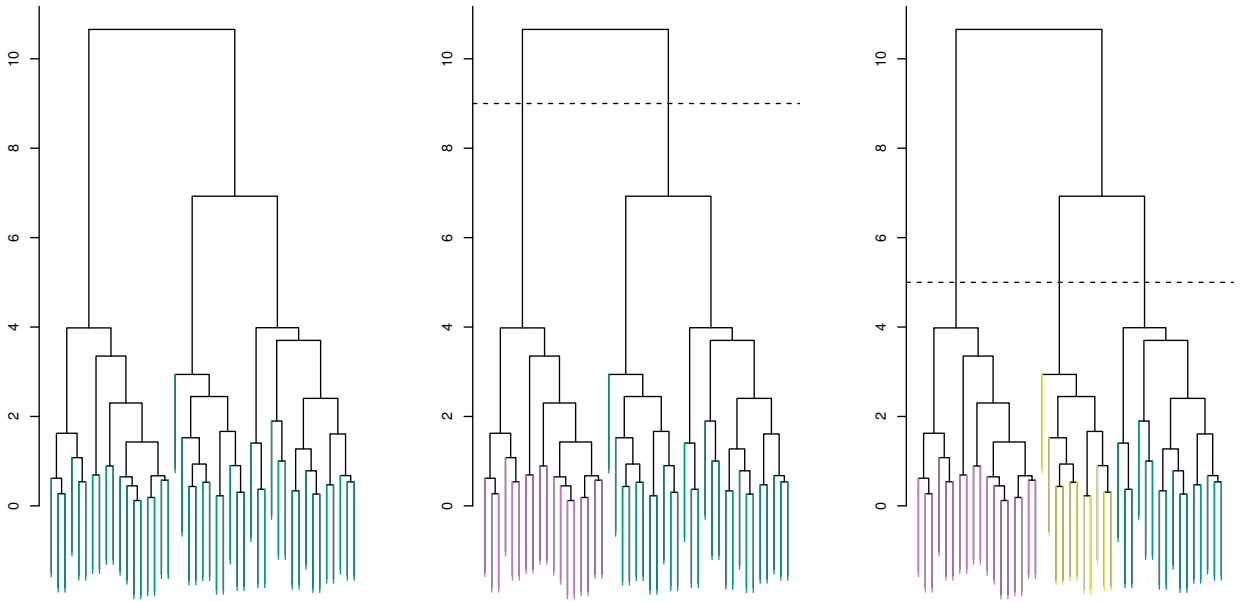


An Example of Hierarchical Clustering



45 observations generated in 2-dimensional space. In reality there are three distinct classes, shown in separate colors. However, we will treat these class labels as unknown and will seek to cluster the observations in order to discover the classes from the data.

An Example of Hierarchical Clustering

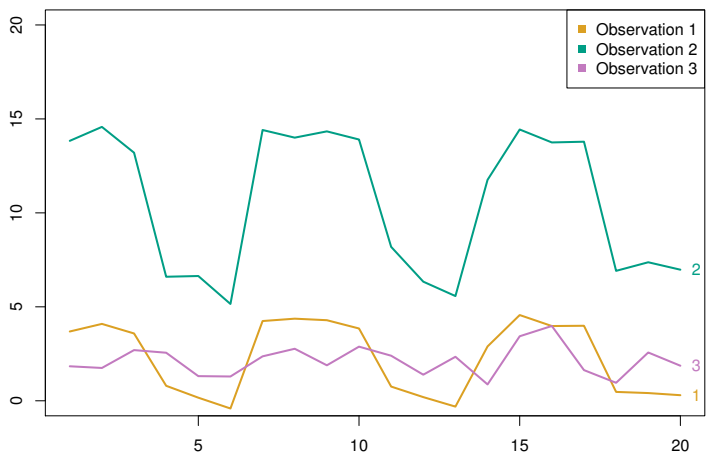


An Example of Hierarchical Clustering

- *Left:* Dendrogram obtained from hierarchically clustering the data from previous slide, with complete linkage and Euclidean distance.
- *Center:* The dendrogram from the left-hand panel, cut at a height of 9 (indicated by the dashed line). This cut results in two distinct clusters, shown in different colors.
- *Right:* The dendrogram from the left-hand panel, now cut at a height of 5. This cut results in three distinct clusters, shown in different colors. Note that the colors were not used in clustering, but are simply used for display purposes in this figure

Choice of Dissimilarity Measure

- So far have used Euclidean distance.
- An alternative is *correlation-based distance* which considers two observations to be similar if their features are highly correlated.
- This is an unusual use of correlation, which is normally computed between variables; here it is computed between the observation profiles for each pair of observations.



Conclusion

- *Unsupervised learning* is important for understanding the variation and grouping structure of a set of unlabeled data, and can be a useful pre-processor for supervised learning
- It is intrinsically more difficult than *supervised learning* because there is no gold standard (like an outcome variable) and no single objective (like test set accuracy)
- It is an active field of research, with many recently developed tools such as *self-organizing maps*, *independent components analysis* and *spectral clustering*.
  See *The Elements of Statistical Learning*, chapter 14.

Gaussian Mixture Model (GMM)

A generative view of clustering

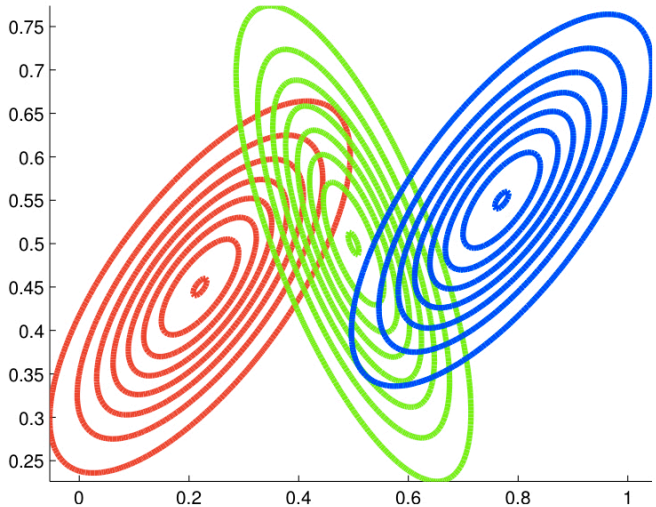
- Last time: hard and soft k-means algorithm
- Today: statistical formulation of clustering $\rightarrow$ principled, justification for updates
- We need a sensible measure of what it means to cluster the data well.
  - This makes it possible to judge different methods.
  - It may help us decide on the number of clusters.
- An obvious approach is to imagine that the data was produced by a generative model.
  - Then we adjust the model parameters to maximize the probability that it would produce exactly the data we observed.

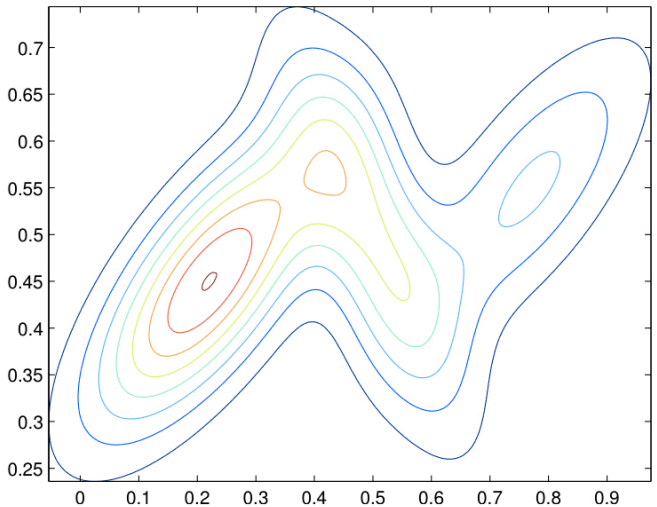
- A Gaussian mixture distribution can be written as

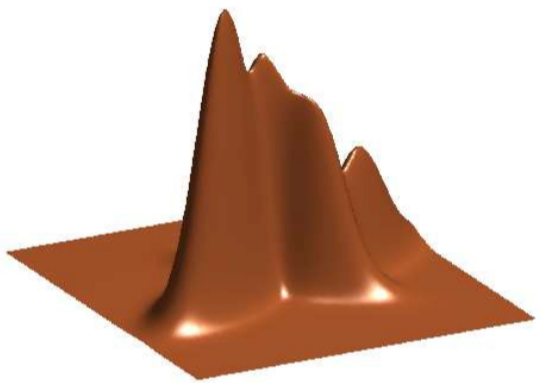
$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

with π_k the mixing coefficients

Gaussian mixture model







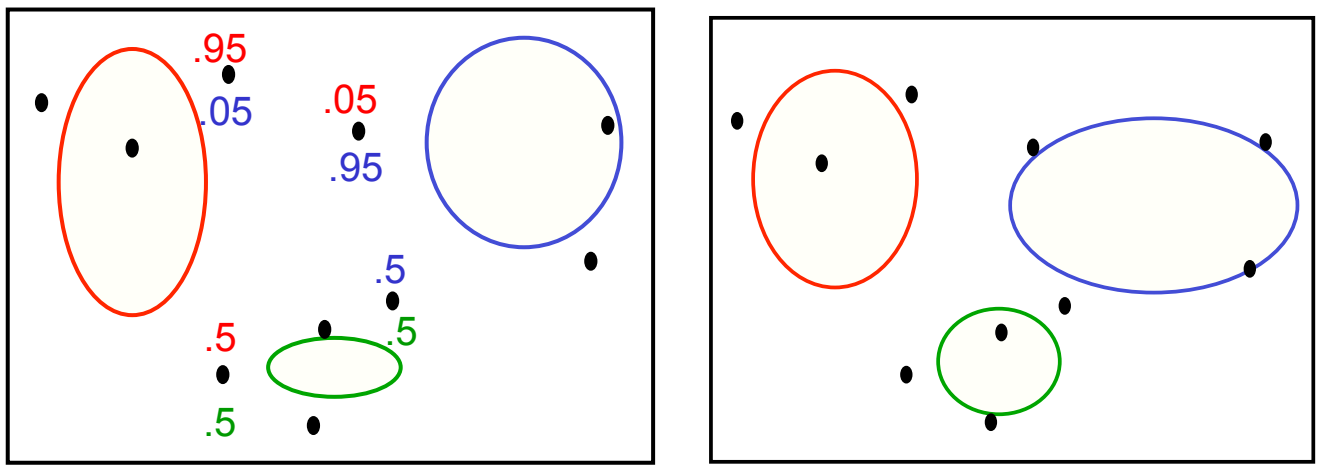
- A Gaussian mixture distribution can be written as

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

  with π_k the mixing coefficients
- Its a density estimator
- Where have we already use a density estimator?

Fitting a mixture of Gaussians

- Optimization uses the Expectation Maximization algorithm, which alternates between two steps:
  1. E-step: Compute the posterior probability that each Gaussian generates each datapoint (as this is unknown to us)
  2. M-step: Assuming that the data really was generated this way, change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.



Latent Variable Models

- Some model variables may be unobserved, either at training or at test time, or both
- If occasionally unobserved they are missing, e.g., undefined inputs, missing class labels, erroneous targets
- Variables which are always unobserved are called latent variables, or sometimes hidden variables

- We may want to intentionally introduce latent variables to model complex dependencies between variables – this can actually simplify the model
- Form of divide-and-conquer: use simple parts to build complex models

- In a mixture model, the identity of the component that generated a given datapoint is a latent variable

- A Gaussian mixture distribution can be written as

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

- Let z_k be a K-dimensional binary random variable z having a 1-of-K encoding

$$z_k \in \{0, 1\}, \quad \sum_k z_k = 1$$

- Joint distribution

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

- The marginal distribution over $\mathbf{z}$ is specified in terms of the mixing coefficients

$$p(z_k = 1) = \pi_k, \quad \text{with} \quad 0 \leq \pi_k \leq 1, \quad \sum_{k=1}^{K} \pi_k = 1$$

- Because $\mathbf{z}$ uses a 1-of-K representation, we can also write

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

- The conditional distribution of $\mathbf{x}$ given a particular value for $\mathbf{z}$ is a Gaussian

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

- The marginal can then be computed as

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

- Every data point has its own latent variable $\mathbf{z}^{(n)}$

- Conditional probability (using Bayes rule) of $\mathbf{z}$ given $\mathbf{x}$

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

- $\gamma(z_k)$ can be viewed as the responsibility

- Maximum likelihood maximizes

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

  w.r.t $\Theta = \{\pi_k, \mu_k, \Sigma_k\}$

- Problems:
  - Singularities: Arbitrarily large likelihood when a Gaussian explains a single point
  - Identifiability: Solution is up to permutations
- How would you optimize this?
- Can we have a closed form update?
- Don't forget to satisfy the constraints on π_k

Objective: Expected Complete Data Likelihood

- Maximum likelihood maximizes

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

- Hard to maximize (log-)likelihood of data directly
- General problem: sum inside the log

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

- Elegant and powerful method for finding maximum likelihood solutions for models with latent variables
  1. E-step:
     - In order to adjust the parameters, we must first solve the inference problem: Which Gaussian generated each datapoint?
     - We cannot be sure, so it's a distribution over all possibilities.

$$\gamma(z_k^{(n)}) = p(z_k = 1|\mathbf{x})$$

  2. M-step:
     - Each Gaussian gets a certain amount of posterior probability for each datapoint.
     - At the optimum we shall satisfy

$$\frac{\partial \ln p(\mathbf{X}|\pi, \mu, \Sigma)}{\partial \Theta} = 0$$

     - We can derive closed form updates for all parameters

$$\frac{\partial \ln p(\mathbf{X}|\pi,\mu,\Sigma)}{\partial \mu_k} = 0 = \sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j \mathcal{N}(\mathbf{x}|\mu_j,\Sigma_j)}}_{\gamma(z_k^{(n)})} \Sigma_k^{-1}(\mathbf{x}^{(n)} - \mu_k)$$

- This gives

$$\mu_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_k^{(n)})\mathbf{x}^{(n)}$$

  with N_k the effective number of points in cluster k

$$N_k = \sum_{n=1}^{N}\gamma(z_k^{(n)})$$

- We just take the center-of gravity of the data that the Gaussian is responsible for
- Just like in K-means, except the data is weighted by the posterior probability of the Gaussian.
- Guaranteed to lie in the convex hull of the data (Could be big initial jump)

- We can get similarly expression for the variance

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k^{(n)})(\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T$$

- We can also minimize w.r.t the mixing coefficients

$$\pi_k = \frac{N_k}{N}, \quad \text{with} \quad N_k = \sum_{n=1}^{N} \gamma(z_k^{(n)})$$

- The optimal mixing proportion to use (given these posterior probabilities) is just the fraction of the data that the Gaussian gets responsibility for.
- Note that this is not a closed form solution of the parameters, as they depend on the responsibilities $\gamma(z_k^{(n)})$, which are complex functions of the parameters
- But we have a simple iterative scheme to optimize

- Initialize the means μ_k, covariances Σ_k and mixing coefficients π_k
- E-step: Evaluate the responsibilities

$$\gamma(z_k) = p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)}$$

- M-step: Re-estimate the parameters

$$\begin{aligned}
\mu_k &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k^{(n)}) \mathbf{x}^{(n)} \\
\Sigma_k &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k^{(n)}) (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T \\
\pi_k &= \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^{N} \gamma(z_k^{(n)})
\end{aligned}$$

- Evaluate log likelihood and check for convergence

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

An Alternative View of EM

- Hard to maximize (log-)likelihood of data directly
- General problem: sum inside the log

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

- Complete data $\{\mathbf{x}, \mathbf{z}\}$, and $\mathbf{x}$ is the incomplete data
- If we knew z, then easy to maximize (replace sum over k with just the k where $z_k = 1$)
- Unfortunately we are not given the complete data, but only the incomplete.
- Our knowledge about the latent variables is $p(\mathbf{Z}|\mathbf{X}, \Theta^{old})$
- In the E-step we compute $p(\mathbf{Z}|\mathbf{X}, \Theta^{old})$
- In the M-step we maximize w.r.t Θ

$$Q(\Theta, \Theta^{old}) = \sum_{z} p(\mathbf{Z}|\mathbf{X}, \Theta^{old}) \ln p(\mathbf{X}, \mathbf{Z}|\Theta)$$

1. Initialize Θ^{old}
2. E-step: Evaluate $p(\mathbf{Z}|\mathbf{X}, \Theta^{old})$
3. M-step:
$$\Theta^{new} = arg\max_{\Theta} Q(\Theta, \Theta^{old})$$
where
$$Q(\Theta, \Theta^{old}) = \sum_{z} p(\mathbf{Z}|\mathbf{X}, \Theta^{old}) \ln p(\mathbf{X}, \mathbf{Z}|\Theta)$$
4. Evaluate log likelihood and check for convergence (or the parameters). If not converged, $\Theta^{old} = \Theta$, Go to step 2

How do we know that the updates improve things?

- Updating each Gaussian definitely improves the probability of generating the data if we generate it from the same Gaussians after the parameter updates.
  - But we know that the posterior will change after updating the parameters.
- A good way to show that this is OK is to show that there is a single function that is improved by both the E-step and the M-step.
  - The function we need is called Free Energy.

Why EM converges

- Free energy F is a cost function that is reduced by both the E-step and the M-step.

$$F = \text{expected energy} - \text{entropy}$$

- The expected energy term measures how difficult it is to generate each datapoint from the Gaussians it is assigned to. It would be happiest assigning each datapoint to the Gaussian that generates it most easily (as in K-means).
- The entropy term encourages "soft" assignments. It would be happiest spreading the assignment probabilities for each datapoint equally between all the Gaussians.

- Our goal is to maximize

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

- Typically optimizing $p(\mathbf{X}|\Theta)$ is difficult, but $p(\mathbf{X}, \mathbf{Z}|\Theta)$ is easy
- Let $q(\mathbf{Z})$ be a distribution over the latent variables. For any distribution $q(\mathbf{Z})$ we have

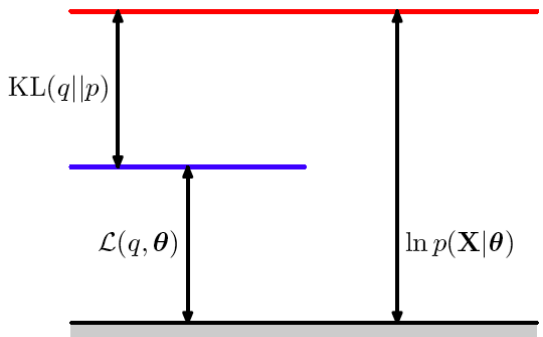
$$\ln p(\mathbf{X}|\Theta) = \mathcal{L}(q, \Theta) + KL(q||p(\mathbf{Z}|\mathbf{X}, \Theta))$$

where

$$\begin{aligned} \mathcal{L}(q, \Theta) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\Theta)}{q(\mathbf{Z})} \right\} \\ KL(q||p) &= -\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \Theta)}{q(\mathbf{Z})} \right\} \end{aligned}$$

- Since the KL-divergence is always positive and have value 0 only if $q(Z) = p(\mathbf{Z}|\mathbf{X}, \Theta)$
- Thus $\mathcal{L}(q, \Theta)$ is a lower bound on the likelihood

$$\mathcal{L}(q, \Theta) \leq \ln p(\mathbf{X}|\Theta)$$



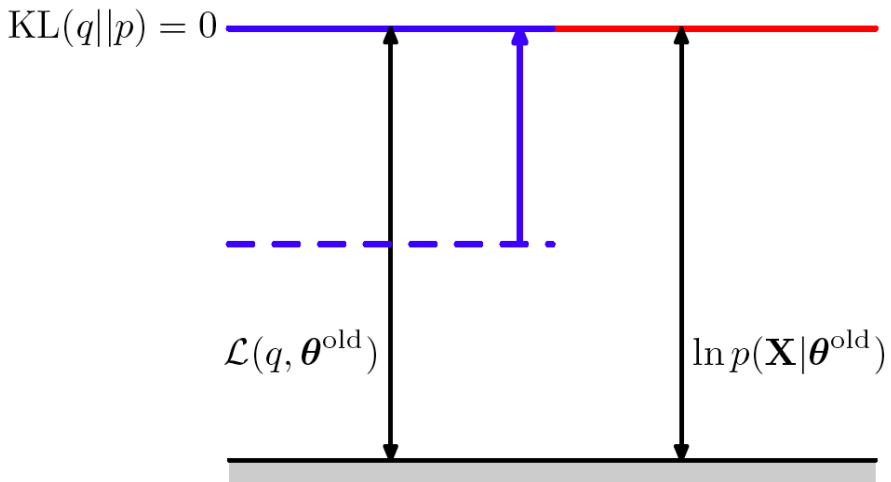
$$\ln p(\mathbf{X}|\Theta) = \mathcal{L}(q, \Theta) + KL(q||p(\mathbf{Z}|\mathbf{X}, \Theta))$$

- In the E-step we maximize w.r.t $q(\mathbf{Z})$ the lower bound $\mathcal{L}(q, \Theta)$
- Since $\ln p(\mathbf{X}|\theta)$ does not depend on $q(\mathbf{Z})$, the maximum $\mathcal{L}$ is obtained when the KL is 0
- This is achieved when $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \Theta)$
- The lower bound $\mathcal{L}$ is then

$$\begin{aligned}\mathcal{L}(q, \Theta) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \Theta^{old}) \ln p(\mathbf{X}, \mathbf{Z}|\Theta) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \Theta^{old}) \ln p(\mathbf{Z}|\mathbf{X}, \Theta^{old}) \\ &= Q(\Theta, \Theta^{old}) + \text{const}\end{aligned}$$

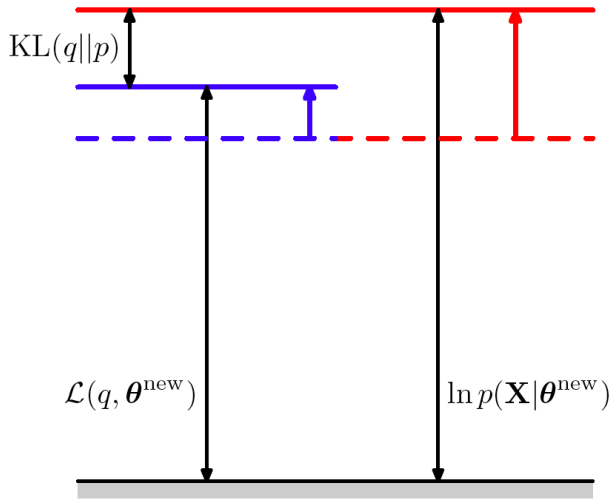
  with the content the entropy of the q distribution, which is independent of Θ
- In the M-step the quantity to be maximized is the expectation of the complete data log-likelihood
- Note that Θ is only inside the logarithm and optimizing the complete data likelihood is easier

Visualization of E-step



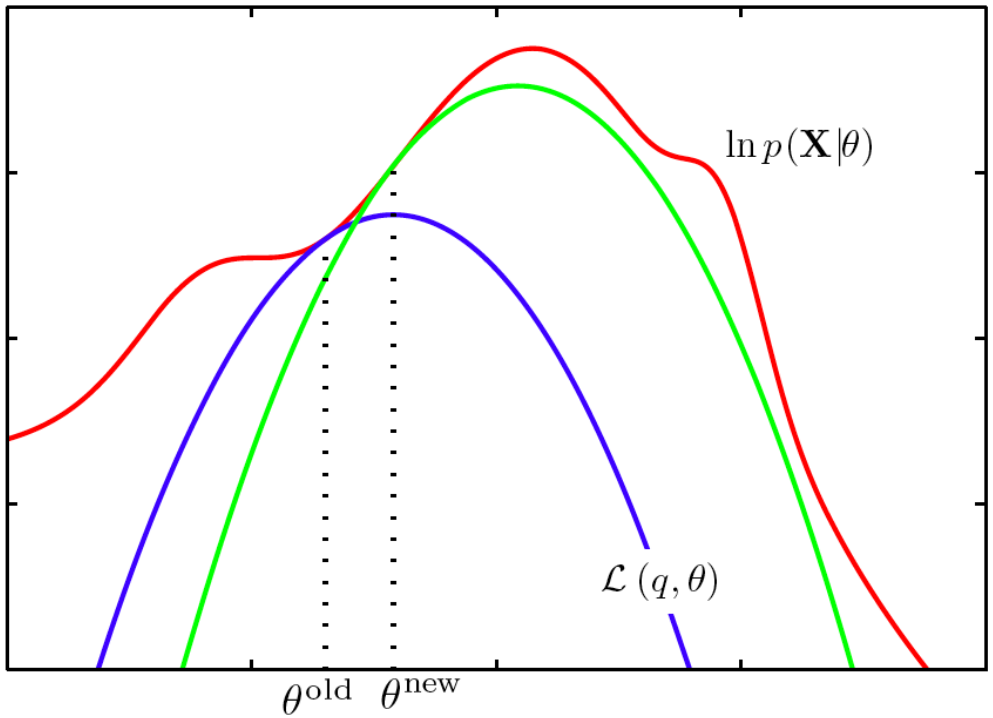
- The q distribution equal to the posterior distribution for the current parameter values Θ^{old}, causing the lower bound to move up to the same value as the log likelihood function, with the KL divergence vanishing.

Visualization of M-step



- The distribution $q(\mathbf{Z})$ is held fixed and the lower bound $\mathcal{L}(q, \Theta)$ is maximized with respect to the parameter vector Θ to give a revised value Θ^{new}. Because the KL divergence is nonnegative, this causes the log likelihood $\ln p(\mathbf{X}|\Theta)$ to increase by at least as much as the lower bound does.

Visualization of the EM Algorithm



- 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.

Summary: EM is coordinate descent in Free Energy Term–Optimization

$$\begin{aligned}\mathcal{L}(q,\Theta) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\Theta^{old}) \ln p(\mathbf{X},\mathbf{Z}|\Theta) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\Theta^{old}) \ln p(\mathbf{Z}|\mathbf{X},\Theta^{old}) \\ &= Q(\Theta,\Theta^{old}) + \text{const} \\ &= \text{expected energy} - \text{entropy}\end{aligned}$$

- The E-step maximizes F by finding the best distribution over hidden configurations for each data point.
- The M-step holds the distribution fixed and maximizes F by changing the parameters that determine the energy of a configuration.

Mixture of Gaussians vs. K-means

- EM for mixtures of Gaussians is just like a soft version of K-means, with fixed priors and covariance
- Instead of hard assignments in the E-step, we do soft assignments based on the softmax of the squared Mahalanobis distance from each point to each cluster.
- Each center moved by weighted means of the data, with weights given by soft assignments
- In K-means, weights are 0 or 1

Acknowledgement

Some slides are in courtesy of
(1)Chap 10 of James *et. al.* "An Introduction to Statistical Learning with applications in R", 2011;
(2) Lecture 12,13, Raquel Urtasun & Rich Zemel, University of Toronto.