CO902
Probabilistic and statistical inference

Lecture 7

Tom Nichols
Department of Statistics &
Warwick Manufacturing Group

t.e.nichols@warwick.ac.uk

Critical Reading Assignment

- 10 minute presentation
- Based on journal paper about or using Machine Learning methods
  Please sign-up on sheet!
- Aim for ~5 slides, not much more
- Try to think of intuitive descriptions for any algorithms/procedures used
- Only 10 minutes! Not a lecture or a tutorial
  Can't possibly explain everything…
  But should be able to express general ideas in play
- Don't read your slides!
  Slides shouldn't have full sentences,
  Just key words/phrase
  to anchor audience's attention, and
  to help guide/remind you of the flow
- Make good use of pretty, meaningful pictures when possible
- Look at the audience as much as possible!
- Practice!

Outline of course

A. Basics: Probability, random variables (RVs), common distributions, introduction to statistical inference

B. Supervised learning: Regression, classification, including high-dimensional issues and Bayesian approaches

C. Unsupervised learning: Dimensionality reduction, clustering and mixture models

D. Networks: Probabilistic graphical models, learning in graphical models, inferring network structure

Unsupervised learning

- **Unsupervised learning**: finding interesting patterns or regularities in data without labelled training data

- **Dimensionality reduction**: finding informative low-dimensional data representations

- **Clustering**: finding interesting groups in data

Dimension Reduction Redux

- For d-dimensional data $\mathbf{x}_i$, PCA finds best k-dimensional representation $\mathbf{y}_i$
  - "Best" in maximizing variance of Y
  - "Best" in minimizing error in an approx. of X

$$\mathbf{y}_i = \mathbf{U}^\top \mathbf{x}_i$$
$$\mathbf{U} \ : \ d \times k$$

- Transformation $\mathbf{U}$ is k first eigenvectors of

$$\mathbf{S} = \frac{1}{n}(\mathbf{X} - \bar{\mathbf{X}})(\mathbf{X} - \bar{\mathbf{X}})^\top$$

… the $d \times d$ covariance of $\mathbf{y}$

$\mathbf{X}$: $d \times n$ matrix of all data

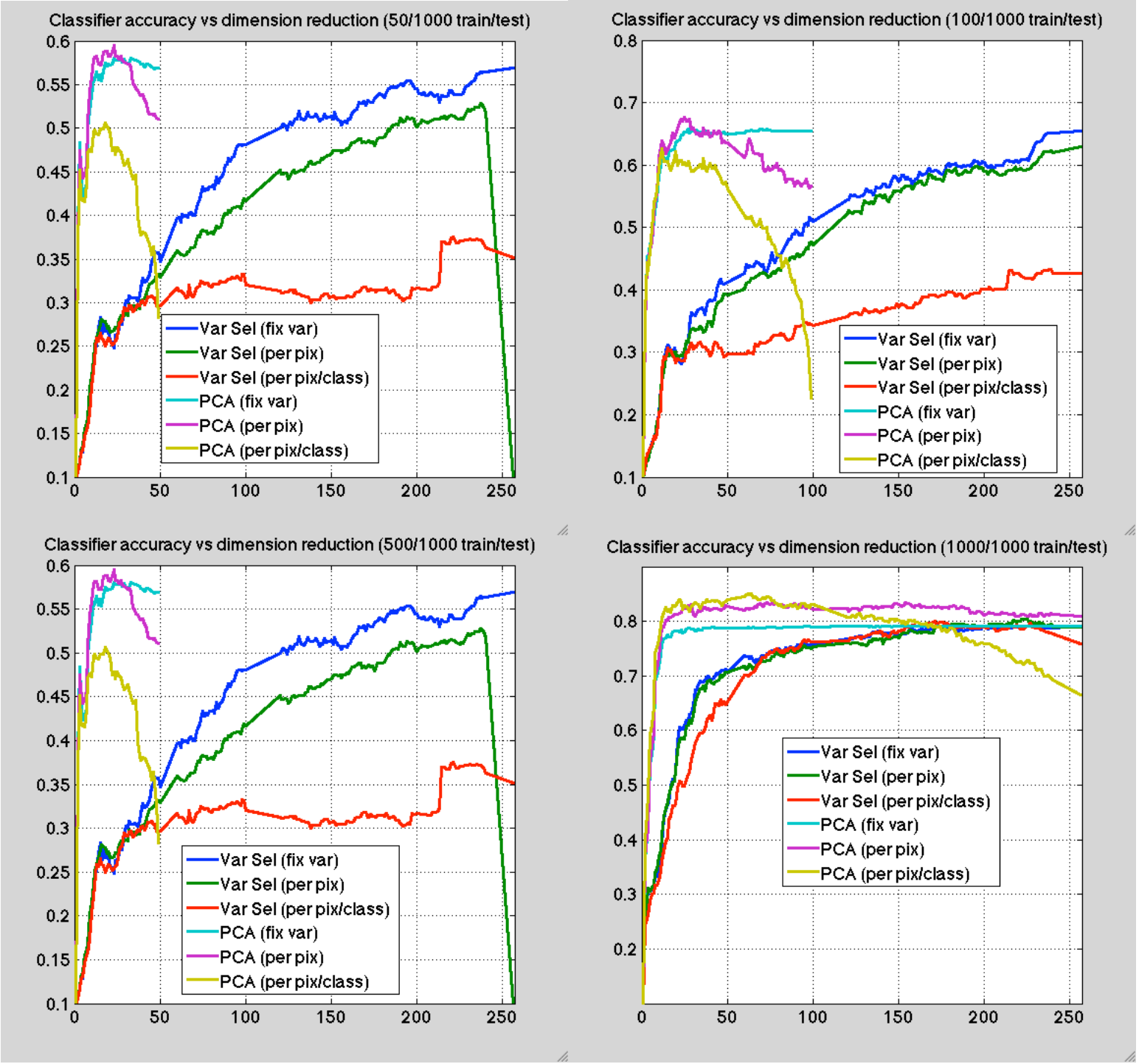
$\bar{\mathbf{X}}$: $d \times n$ matrix of means; each column is identical and equal to $\frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$

- PCA approximation

$$\hat{\mathbf{X}} = \bar{\mathbf{X}} + \mathbf{U}\mathbf{U}^\top(\mathbf{X} - \bar{\mathbf{X}})$$

Lab 6

- PCA clear winner per dimension
- Often absolute winner
- Best assumption on the variance depends on how much data you have!



SVD for PCA

- Singular Value Decomposition
  - Factorisation for arbitrary (non-square) matrices
  - For $d \times n$ matrix $\mathbf{X}$

    $$\mathbf{X} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V'}$$

    where

    $\mathbf{U}$ – Eigenvectors of $\mathbf{XX'}$

    $\mathbf{V}$ – Eigenvectors of $\mathbf{X'X}$

    $\mathbf{\Sigma}$ – $(\mathbf{\Sigma})_{jj} = \sqrt{\lambda_j}$ where λ_j are common eigenvalues of $\mathbf{XX'}$ and $\mathbf{X'X}$

- Can find eigenvectors of $\mathbf{XX'}$ directly with $\mathbf{X}$
  - No need to make huge covariance matrix
  - Very handy if $d \gg n$
    - e.g. thousands of genes, 20 subjects

SVD for PCA

- Carefully… *(on the board… details also in PCA PDF on web)*

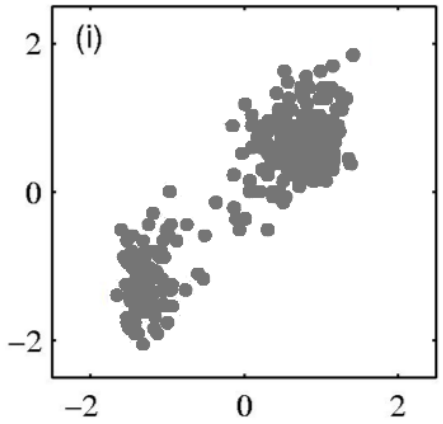
- So… can either compute
  - [U, D] = eig(S);
    Col's of U eigenvectors of S
    Diagonal of D eigenvalues of S (sorted **ascending**)
  - [U, Dc] = svd(Xcenter)
    Col's of U eigenvectors of S
    diag(Dc).^2/n eigenvalues of S (sorted **descending**)

Today

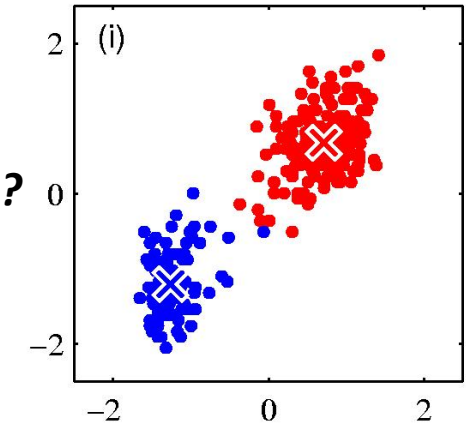
- Next couple of lectures on clustering:
  - Intro to clustering
  - K-means clustering
  - Mixture models
  - EM algorithm

Clustering

- **Clustering**: task of finding interesting groups in data
- Idea is to uncover groups of data points that are similar to each other



Two "clusters" ?



- Again, something we animals do unconsciously all the time
- Arguably one of the main ways we learn autonomously from the environment

(Figures from Bishop, Chapter 9)

Some examples

- Microarray data: finding groups of tissues which share molecular characteristics across many dimensions
- For example:
  - Biopsy large number of tumours
  - Measure vector **X** of gene expression levels in each tumour
  - Want to discover tumour subgroups with broadly similar expression patterns
- Basically what scientists routinely do manually
- Need automated, formalized approaches
  - To speed up discovery
  - To go beyond limitations of what the human eye can visualize
- *Another example…*

Some examples

- Images: finding groups of pixels which have similar RGB vector ("vector quantization")

Original image

$K = 10$

$K = 3$

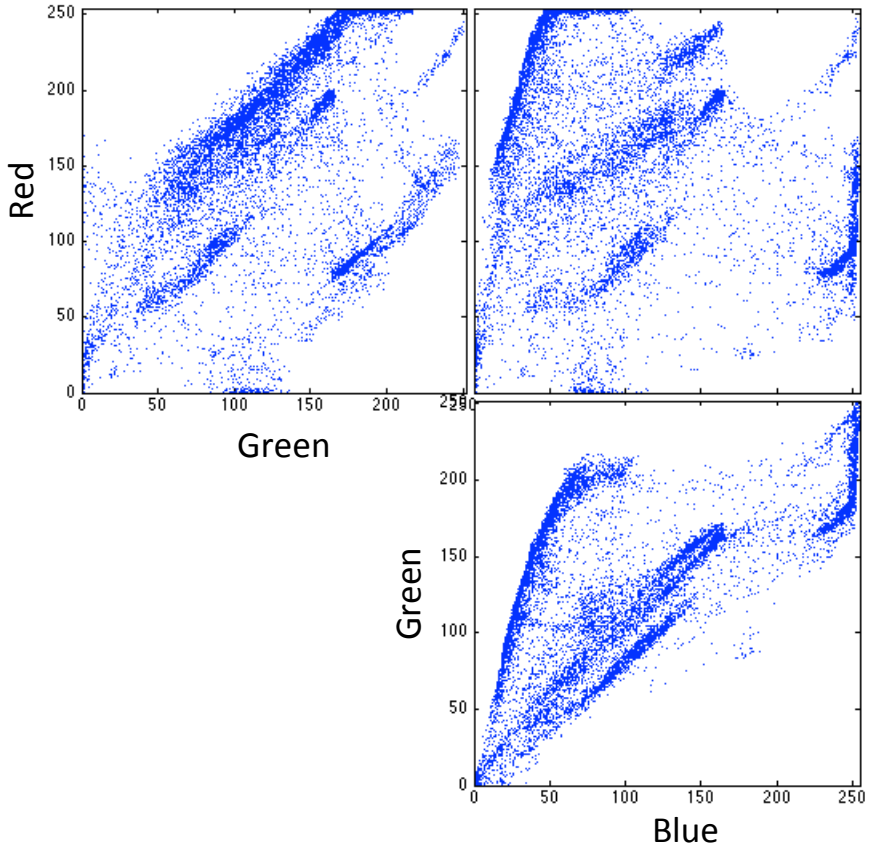
- **Q: What are the "data vectors" here, i.e. the objects being clustered?**

Original image

$K = 10$

$K = 3$

Original RGB Distribution

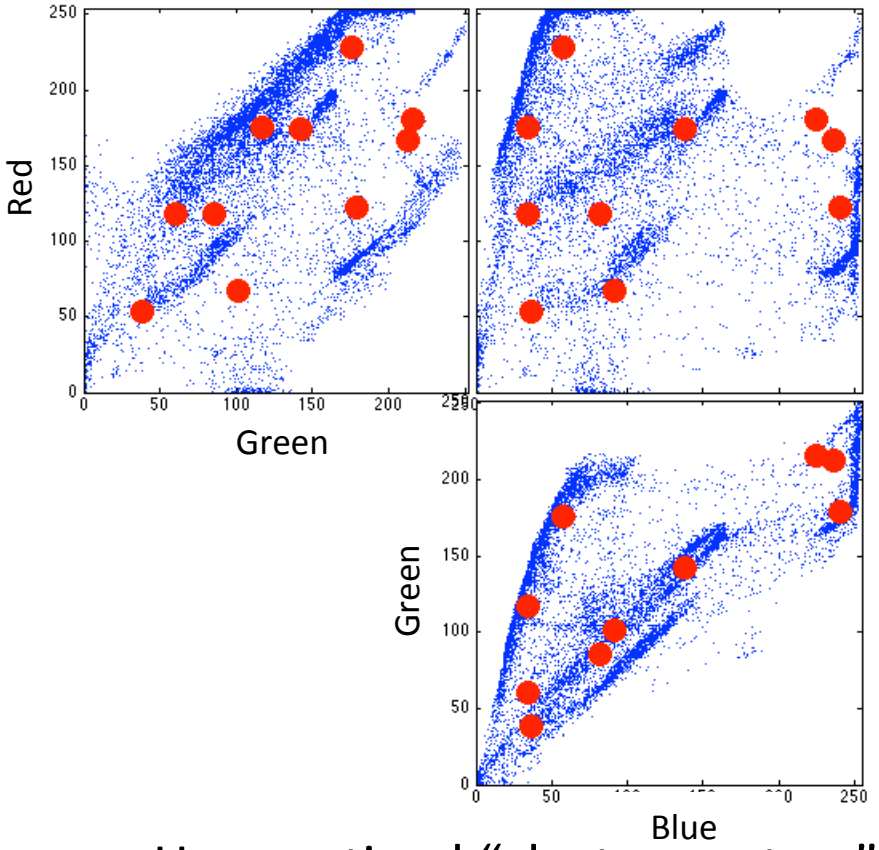


Original image

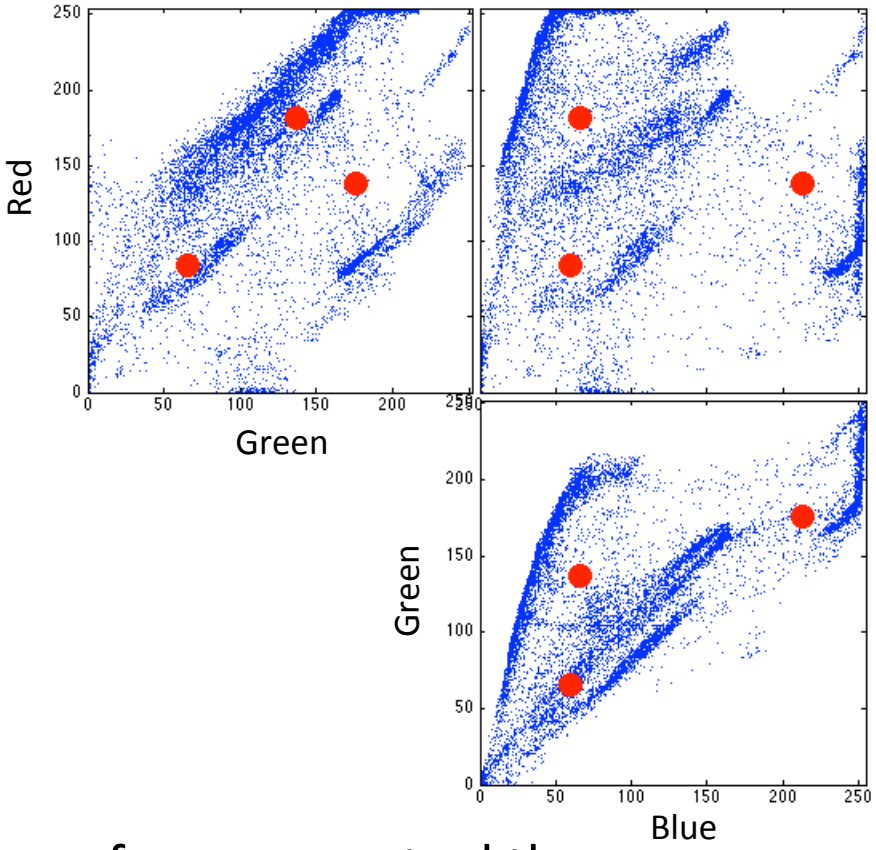
$K = 10$

$K = 3$

K = 10 “Clusters”

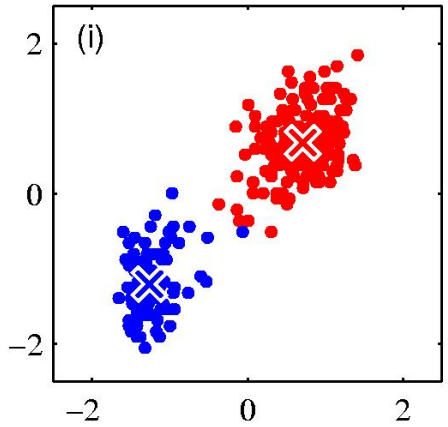


K = 3 “Clusters”



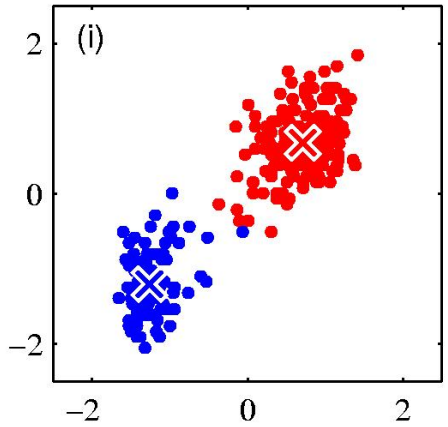
- Here optimal “cluster centres” come from perceptual theory
  - Not, e.g., minimization of intra-cluster variance

K-means clustering



- Clusters are groups of data vectors that are similar to each other
- **K-means clustering** is a simple approach to clustering in which we seek to place the data points into *K* groups so as to maximize the "tightness" of those groups

Some notation

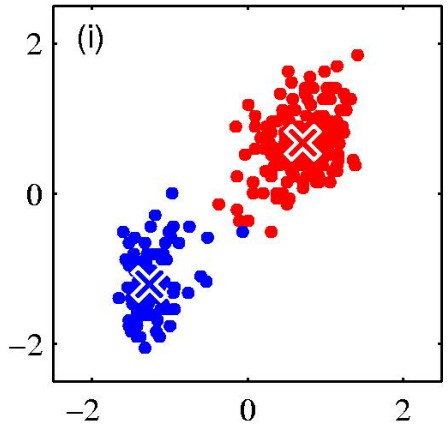


Data: $\mathbf{X}_1, \mathbf{X}_2 \ldots \mathbf{X}_n, \mathbf{X}_i \in \mathbb{R}^d$

Clusters: $\{1 \ldots K\}$

Cluster assignment fn: $i \in \{1 \ldots n\}, C(i) \in \{1 \ldots K\}$

Objective function to be minimized



- Choose cluster assignments and means to minimize:

$$J(\{\boldsymbol{\mu}_k\}, \{C(i)\}) = \sum_{k=1}^{K} \sum_{i:C(i)=k} \|\mathbf{X}_i - \boldsymbol{\mu}_k\|^2$$

- That is, find *K* groups which have lowest overall within-group deviation from cluster-specific means

Objective function

- Choose cluster assignments and means to minimize:

$$J(\{\boldsymbol{\mu}_k\}, \{C(i)\}) = \sum_{k=1}^{K} \sum_{i:C(i)=k} \|\mathbf{X}_i - \boldsymbol{\mu}_k\|^2$$

- We need to infer
  - *K* means (hence the name)
  - Cluster assignments
- Write objective function J explicitly as sum over clusters:

$$J(\{\boldsymbol{\mu}_k\}, \{C(i)\}) = \sum_{i:C(i)=1} \|\mathbf{X}_i - \boldsymbol{\mu}_1\|^2 + \sum_{i:C(i)=2} \|\mathbf{X}_i - \boldsymbol{\mu}_2\|^2 \ldots + \sum_{i:C(i)=K} \|\mathbf{X}_i - \boldsymbol{\mu}_K\|^2$$

Algorithm

- Write objective function J explicitly as sum over clusters:

$$J(\{\boldsymbol{\mu}_k\}, \{C(i)\}) \quad = \quad \sum_{i:C(i)=1} \|\mathbf{X}_i - \boldsymbol{\mu}_1\|^2 + \sum_{i:C(i)=2} \|\mathbf{X}_i - \boldsymbol{\mu}_2\|^2 \ldots + \sum_{i:C(i)=K} \|\mathbf{X}_i - \boldsymbol{\mu}_K\|^2$$

- (A) First, initial random guess for means (*K* is fixed)
- (B) Then, two-steps:

  (1) keeping means fixed, optimize assignments *C(i)*

  (2) Then, keeping newly updated *C(i)* fixed, optimize means

- **Q: what are optimal means and assignments for (1) and (2)?**

Updates

- Write objective function J explicitly as sum over clusters:

$$J(\{\boldsymbol{\mu}_k\}, \{C(i)\}) = \sum_{i:C(i)=1} \|\mathbf{X}_i - \boldsymbol{\mu}_1\|^2 + \sum_{i:C(i)=2} \|\mathbf{X}_i - \boldsymbol{\mu}_2\|^2 \ldots + \sum_{i:C(i)=K} \|\mathbf{X}_i - \boldsymbol{\mu}_K\|^2$$

Keeping assignments *C(i)* fixed, update means

We know the mean minimizes sum-of-squares, so fixing *C(i)*:

$$\begin{aligned} \hat{\boldsymbol{\mu}}_k &= \frac{1}{n_k} \sum_{i:C(i)=k} \mathbf{X}_i \\ n_k &= |\{i : C(i) = k\}| \end{aligned}$$

Keeping means fixed, update cluster assignments *C(i)*

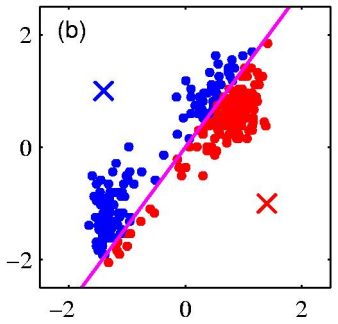
Note that each data vector appears in exactly one term above

So all we have to do is assign each data point to the cluster with the closest mean:

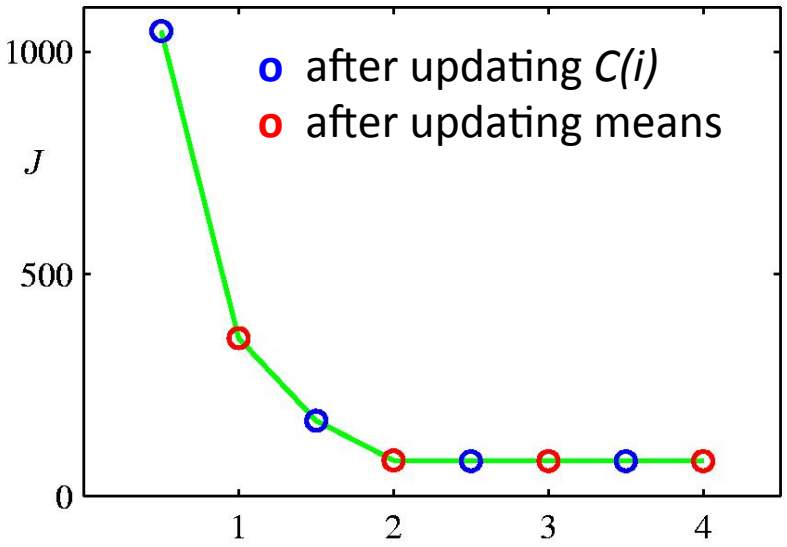
$$\hat{C}(i) = \underset{k}{\operatorname{argmin}} \|\mathbf{X}_i - \boldsymbol{\mu}_k\|^2$$

Stopping criterion

- Each step (1) and (2) decreases (or doesn't increase) objective *J*
- So the algorithm improves the score monotonically
- We stop when either
  - (i) Little or no improvement
  - (ii) Some max_iters reached



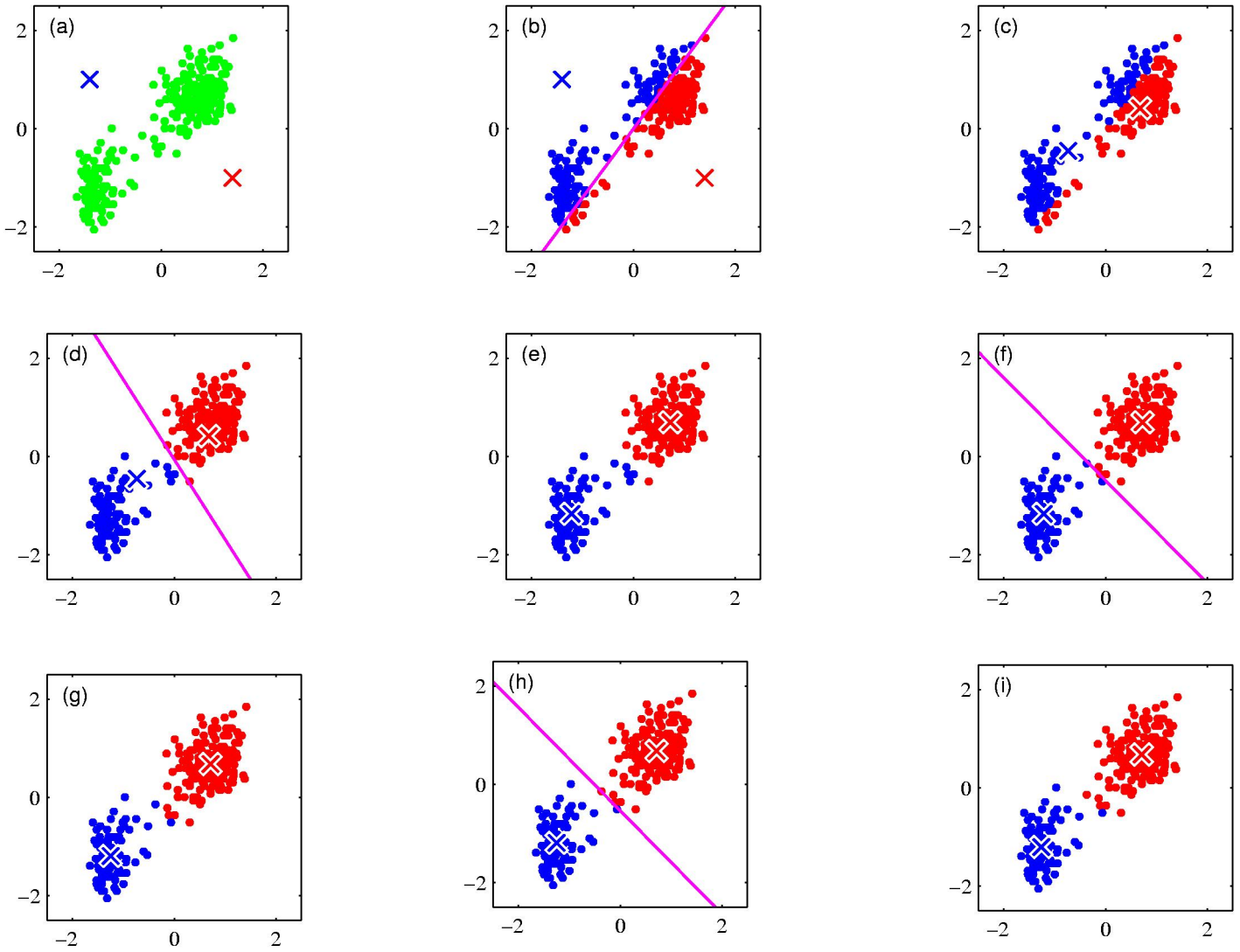
Initialization



J vs. iterations

(Figures from Bishop, Chapter 9)

Example



(Figures from Bishop, Chapter 9)

Digits Example

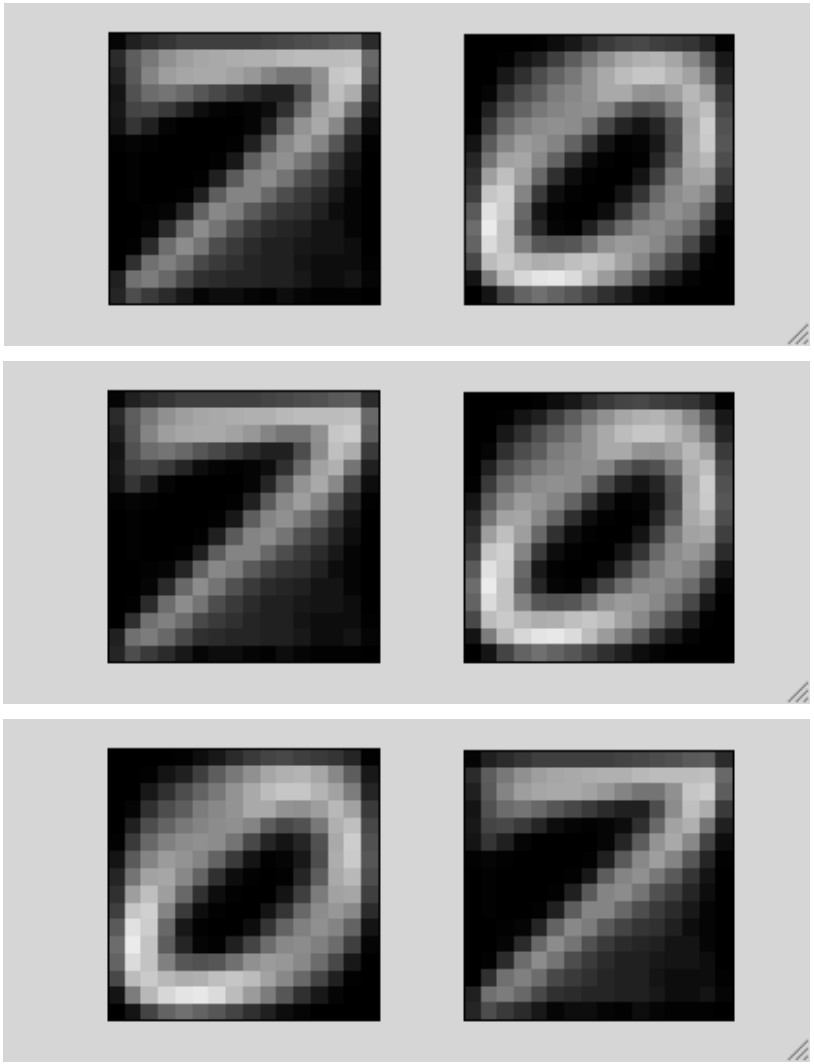
- Digits 7 & 10, 100 samples each
  - Not a lot of data, given d = 256!
- 3 initializations…

	Class 1	Class 2
True 7	0.9900	0.0100
True 0	0.0200	0.9800

Supervised validation of an unsupervised method

	Class 1	Class 2
True 7	0.9900	0.0100
True 0	0.0200	0.9800

	Class 1	Class 2
True 7	0.0200	0.9800
True 0	0.9900	0.0100



Digits Example

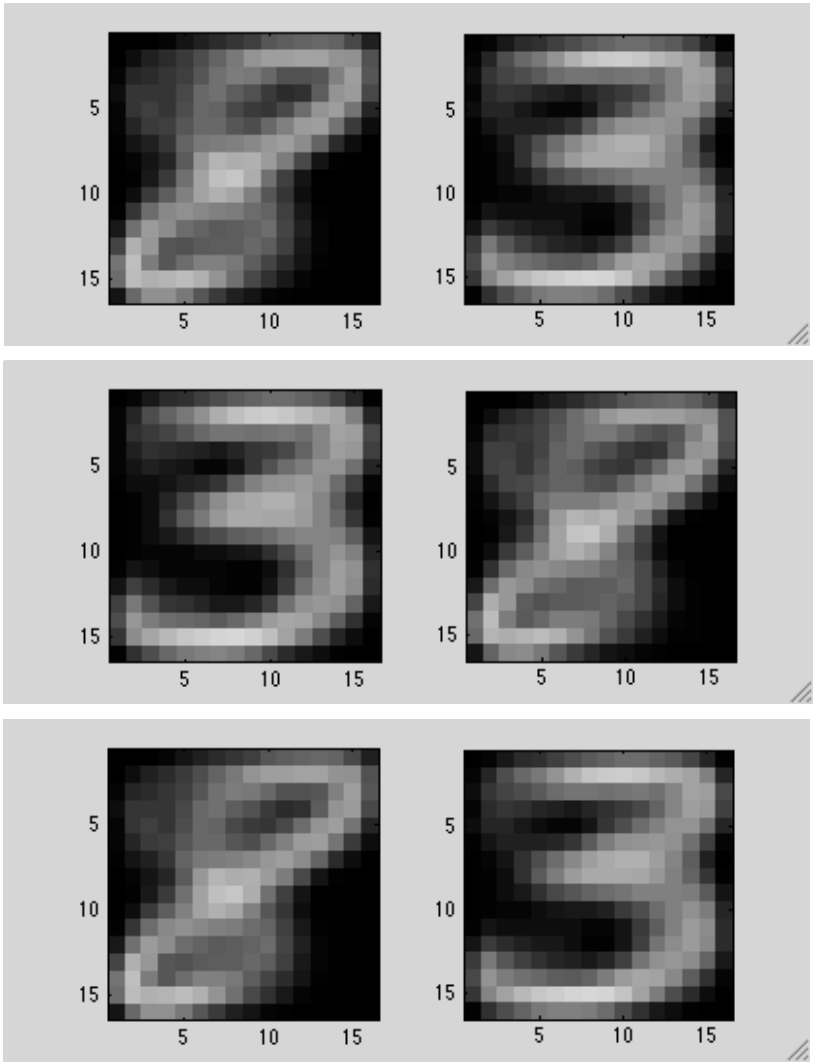
- Digits 3 & 8, 100 samples each
  - Not a lot of data, given d = 256!
- 3 initializations…

	Class 1	Class 2
True 3	0.0800	0.9200
True 8	0.8700	0.1300

Supervised validation of an unsupervised method

	Class 1	Class 2
True 3	0.9200	0.0800
True 8	0.0900	0.9100

	Class 1	Class 2
True 3	0.0800	0.9200
True 8	0.8700	0.1300



Practical issues

- Algorithm gives at best a local minimum of *J*
- Dependent on initialization
- In practice, you would start it with multiple, random initial means, and return the single best result (i.e. lowest *J*)

- How to set K?
- **Q: What's the *optimal* K, to minimize J?**

Setting K

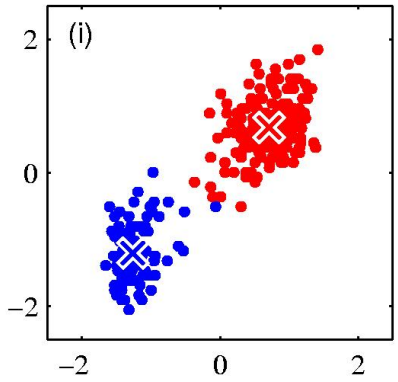
- How to set K?
- **Q: What's the optimal K, to minimize J?**

- Another example of overfitting!

- One way is to simply increase K, monitor J, and look for a "kink"
- Not very satisfactory, and doesn't always work
- This is one key implementation issue (limitation) with K-means

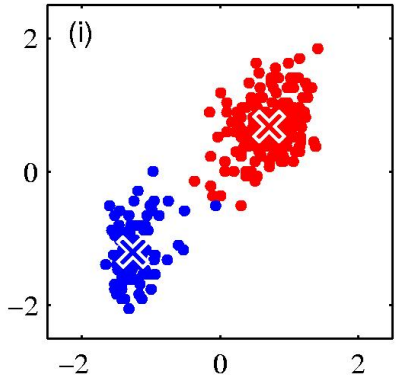
- Another issue is the choice of distance metric
- **Q: can you draw a couple of good, simple clusters which K means will have trouble finding?**

Quantifying certainty in cluster membership



- Not all points which end up assigned to a cluster are equally obviously members of it
- Can be useful to quantify this (not so easy to visualize for higher dimensional data)
- E.g. you've found two kinds of breast cancer, want to do follow up work on the tissues, but to focus on the most clear cluster exemplars

Probabilistic clustering



- K-means useful approach, but no probability model
- Want to quantify how certain we are of cluster membership
- We can do clustering under an explicit **probability model**, then can…

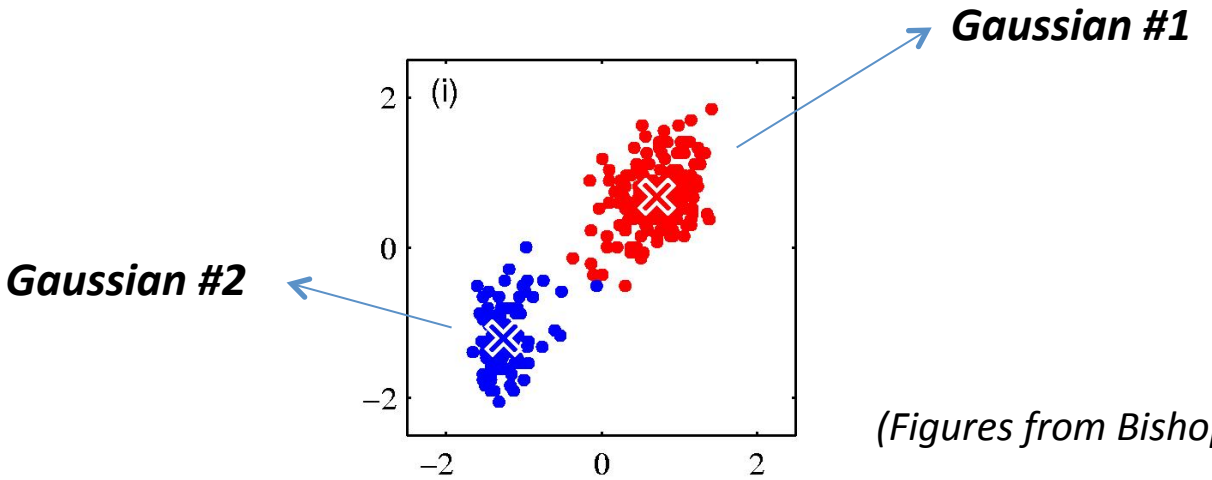
  (i) quantifying membership,

  (ii) getting distance metric appropriate for underlying cluster densities,

  (iii) learning K
- Also offers a window into a very general maximum-likelihood method for fitting models where *some data are unobserved*
- How? Think back to classification…

Clustering re-visited

- Consider analogy with classification using a **class-conditional Gaussian** model:



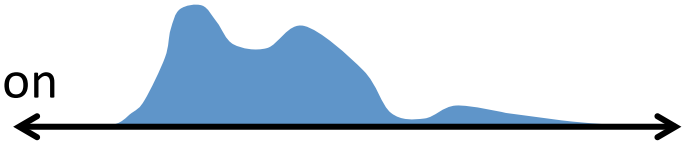
(Figures from Bishop, Chapter 9)

- Why not simply describe clusters with probability densities – one per cluster – in the way we did for classes?
- First, we need a way to combine multiple pdfs…

Mixture Models in General

- Data & "latent" variable (a missing label)

X …data with some continuous distribution



$Z : P(Z = j) = \pi_j, \quad j = 1, \ldots, k$ … discrete label

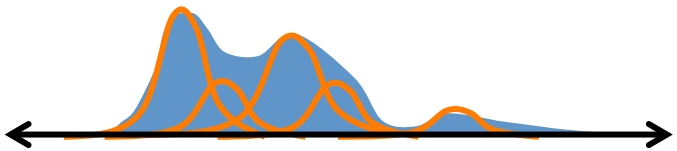
- Now use sum and product rules…

$$p(x) = \sum_{j=1}^{k} p(x, Z = j)$$

(some abuse of measure theory… p( ) here is a pdf and pmf at the same time)

$$= \sum_{j=1}^{k} p(x|Z = j)P(Z = j)$$

$$= \sum_{j=1}^{k} p_j(x)\ \pi_j$$



In other words…
A pdf of a mixture of random variables is a convex combination pdf's

Remember! Z_i's are latent… they don't necessarily *have to* have real/ actual interpretations…
… but for "clustering" we *will* make such interpretation…

Mixture models

- A mixture model is a pdf formed by a linear combination of "component" pdfs
- A **Gaussian Mixture Model** (GMM) with *k components*:

"Mixture components"

$$p(x) = \sum_{j=1}^{k} \pi_j \mathcal{N}(x \mid \mu_j, \sigma_j^2)$$

Mixing weights (or coefficients)

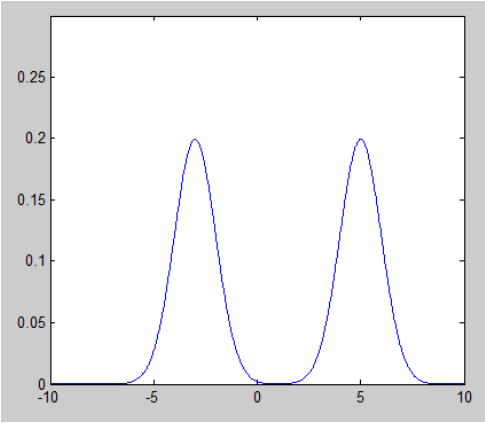
with $\sum_{j=1}^{k} \pi_j = 1, \;\; 0 \leq \pi_j \leq 1$

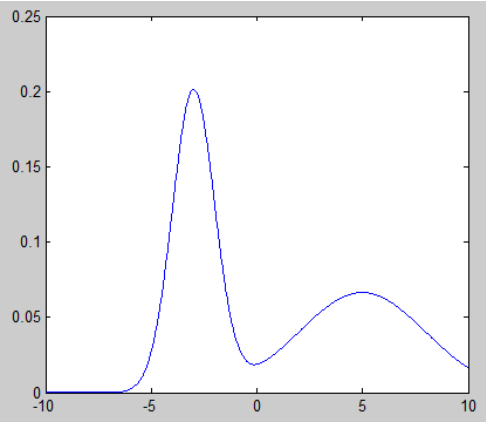
- Sanity check! Does this integrate to 1?

- Some one-dimensional examples…

Mixture models: 1-d example

- Density function: $$p(x) = \sum_{j=1}^{k} \pi_j \mathcal{N}(x \mid \mu_j, \sigma_j^2)$$





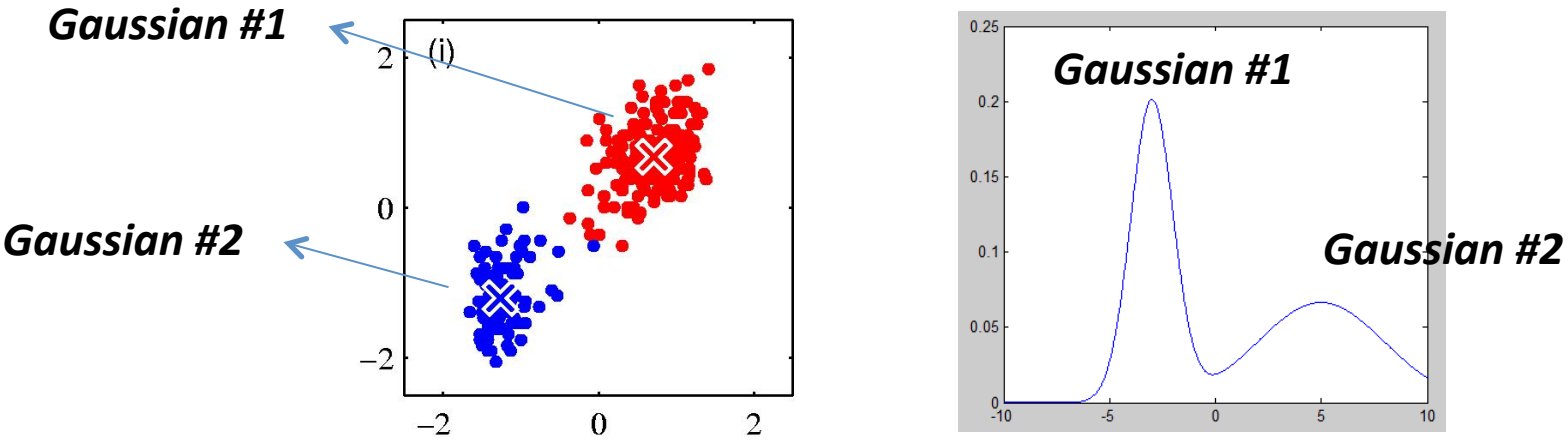

```
>> xs=linspace(-10,10,1000);
>> plot(xs,0.5*normpdf(xs,-3,1)+0.5*normpdf(xs,5,1))
```

```
>> plot(xs,0.5*normpdf(xs,-3,1)+0.5*normpdf(xs,5,3))
>>
```

- **Q: Can you relate the pictures and code to the pdf and its parameters? How would you *generate data* from a mixture model?**

Mixture models and clustering

- Density function: $p(x) = \sum_{j=1}^{k} \pi_j \mathcal{N}(x \mid \mu_j, \sigma_j^2)$



- If we could *fit* a mixture model to data, we'd be characterizing cluster-specific densities
- We will use an algorithm called EM to fit mixture models…

Fitting a mixture model

- Density function: $p(x) = \sum_{j=1}^{k} \pi_j \mathcal{N}(x \mid \mu_j, \sigma_j^2)$
- Parameters: $\{\mu_j, \sigma_j^2, \pi_j\} \quad j = 1..k$
- Fitting the model would be easy if we knew which clusters each data point came from: then it would be like learning class-conditional distributions
- But we don't know this piece of information
- Will use algorithm called **Expectation-Maximization** or **EM**

EM algorithm

- Algorithm called **Expectation-Maximization** or **EM**
- Iterative:

  "E-step": work out "soft" cluster assignments

  "M-step": estimate parameters, using soft assignments

- First, initialize all parameters (randomly, or based on K-means)
- Then, iterate between steps, monitoring likelihood
- Here's what it looks like for a 1-d Gaussian mixture model

EM for 1-d GMM (1)

- **Initialize parameters:** $\{\mu_j, \sigma_j^2, \pi_j\} \quad j = 1..k$

- **E-step:**

$$\gamma_{ij} = \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \sigma_j^2)}{\sum_{l=1}^{k} \pi_l \mathcal{N}(x_i \mid \mu_l, \sigma_l^2)}$$

"soft" cluster assignments

Are these positive?
For each case i, do they sum to unity over classes $j=1, \ldots, k$?

- **M-step (for means):**

$$\hat{\mu}_j = \frac{\sum_{i=1}^{n} \gamma_{ij} x_i}{\sum_{i=1}^{n} \gamma_{ij}}$$

EM for 1-d GMM (2)

- **E-step:**

$$\gamma_{ij} = \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \sigma_j^2)}{\sum_{l=1}^{k} \pi_l \mathcal{N}(x_i \mid \mu_l, \sigma_l^2)}$$

"soft" cluster assignments

- **M-step (for variances):**

$$\hat{\sigma}_j^2 = \frac{\sum_{i=1}^{n} \gamma_{ij}(x_i - \hat{\mu}_j)^2}{\sum_{i=1}^{n} \gamma_{ij}}$$

- **M-step (for mixing coefficients):**

$$\hat{\pi}_j = \frac{1}{n} \sum_{i=1}^{n} \gamma_{ij}$$

Analogous to "proportion" of points assigned to cluster j

EM for 1-d GMM (3)

- **Store current (mixture-model) likelihood:**

$$p(x_1 \ldots x_n | \{\hat{\mu}_j, \hat{\sigma}_j^2, \hat{\pi}_j\}) = \prod_{i=1}^{n} \left[\sum_{j=1}^{k} \hat{\pi}_j \mathcal{N}(x_i \mid \hat{\mu}_j, \hat{\sigma}_j^2) \right]$$

- Repeat E- and M-steps until either (i) convergence of likelihood, or (ii) maximum iterations

EM for general GMM (1)

- Applies in natural way to higher dimensional Gaussians
- Model: $p(\mathbf{x}) = \sum_{j=1}^{k} \pi_j \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$
- Parameters: $\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \pi_j\} \quad j = 1..k \qquad \sum_{j=1}^{k} \pi_j = 1$
- E-step: $\gamma_{ij} = \frac{\pi_j \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{l=1}^{k} \pi_l \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}$

EM for general GMM (2)

- Model: $$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_j \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

- M-step:

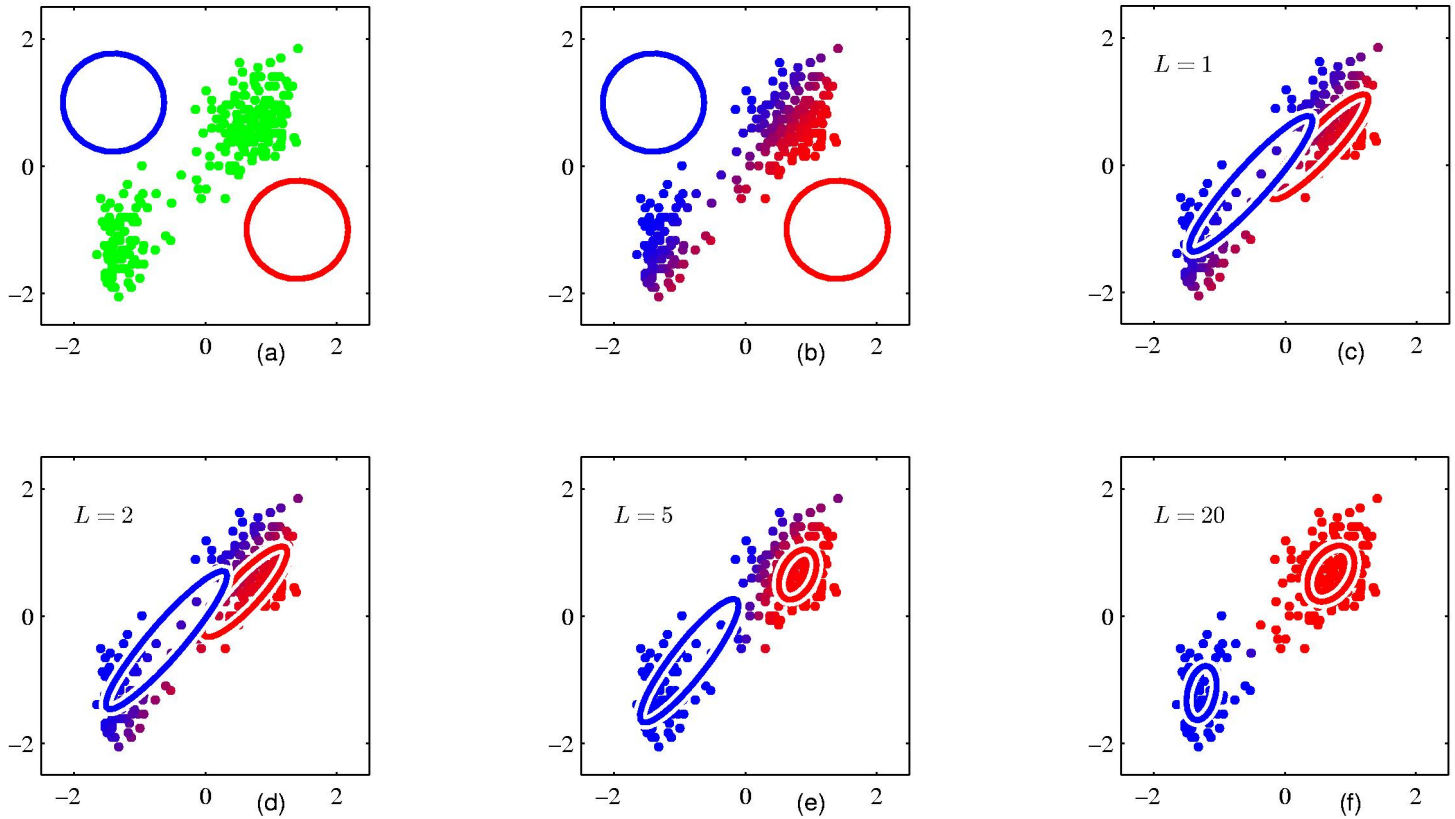
$$\hat{\boldsymbol{\mu}}_j = \frac{\sum_{i=1}^{n} \gamma_{ij} \mathbf{x}_i}{\sum_{i=1}^{n} \gamma_{ij}} \qquad \hat{\boldsymbol{\Sigma}}_j = \frac{\sum_{i=1}^{n} \gamma_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T}{\sum_{i=1}^{n} \gamma_{ij}}$$

$$\hat{\pi}_j = \frac{1}{n} \sum_{i=1}^{n} \gamma_{ij}$$

- Likelihood:

$$p(\mathbf{x}_1 \dots \mathbf{x}_n | \{\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j, \hat{\pi}_j\}) = \prod_{i=1}^{n} \sum_{j=1}^{k} \hat{\pi}_j \mathcal{N}(\mathbf{x}_i \mid \hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j)$$

EM: example



(From Bishop, Chap 9)

EM - advantages

- Probabilistic:
  - K-means used Euclidean distance. This approach induces a distance metric appropriate to the probability densities involved
  - Can quantify certainty of cluster membership
  - Doesn't treat a point which is almost equally close to two clusters the same as one which is very close to one
  - Can do model selection (i.e. setting *k*) using usual statistical machinery

EM - advantages

- Convergence properties:
  - Guaranteed to not decrease likelihood at each step (monotonicity)
  - But only a local maximum of the likelihood function

- Theory:
  - Uses the "latent variable model", where cluster assignments Z_i are the "missing data"
  - Algorithm based on very general idea of maximizing the expectation of the "full" likelihood (i.e. as if you knew the right cluster assignments) under probability of cluster membership given data (hence "E" and "M" steps)
  - EM applies also to many other problems, e.g. missing data, models with unobserved parts

Conclusions

- Clustering: very useful approach for grouping data and *discovering* “types” of objects (as opposed to learning to categorize into known classes)
- As with all unsupervised learning, must be careful not to over interpret: it’s not so easy to assess predictive ability