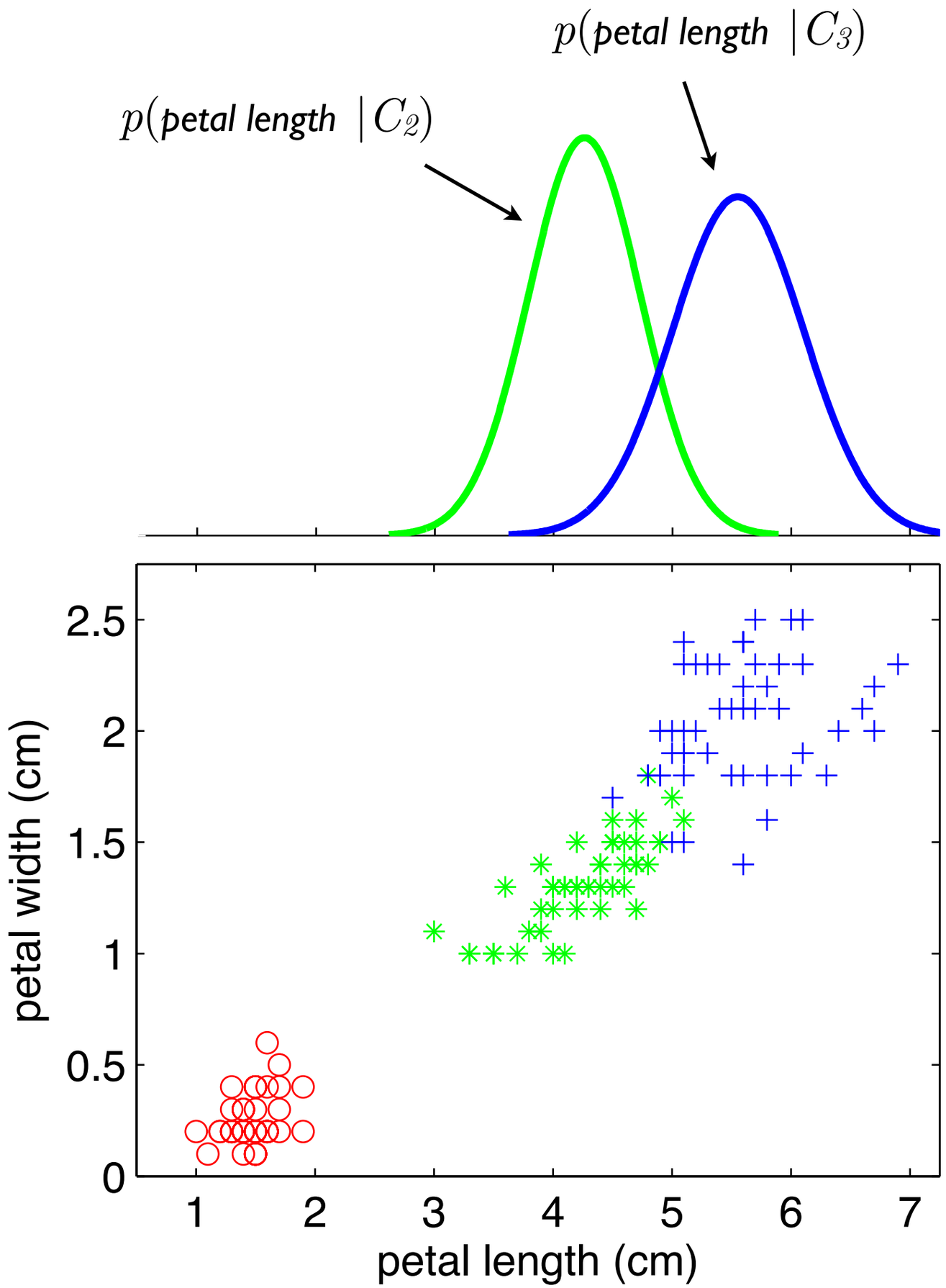
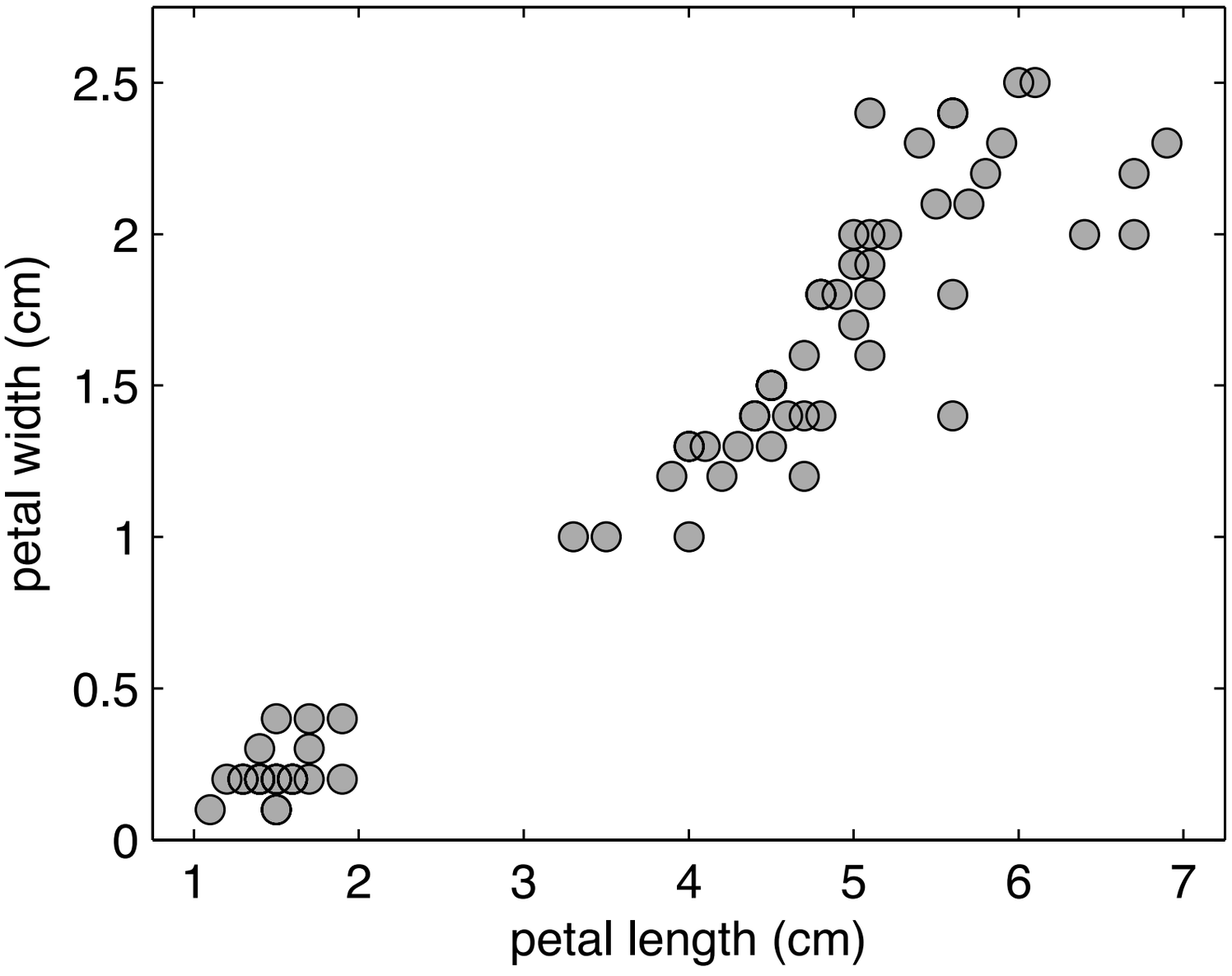
Artificial Intelligence
EECS 491

Gaussians, Mixture Models, and Clustering


$p(\text{petal length} \mid C_3)$
$p(\text{petal length} \mid C_2)$
petal width (cm)
petal length (cm)
0
0.5
1
1.5
2
2.5
1
2
3
4
5
6
7

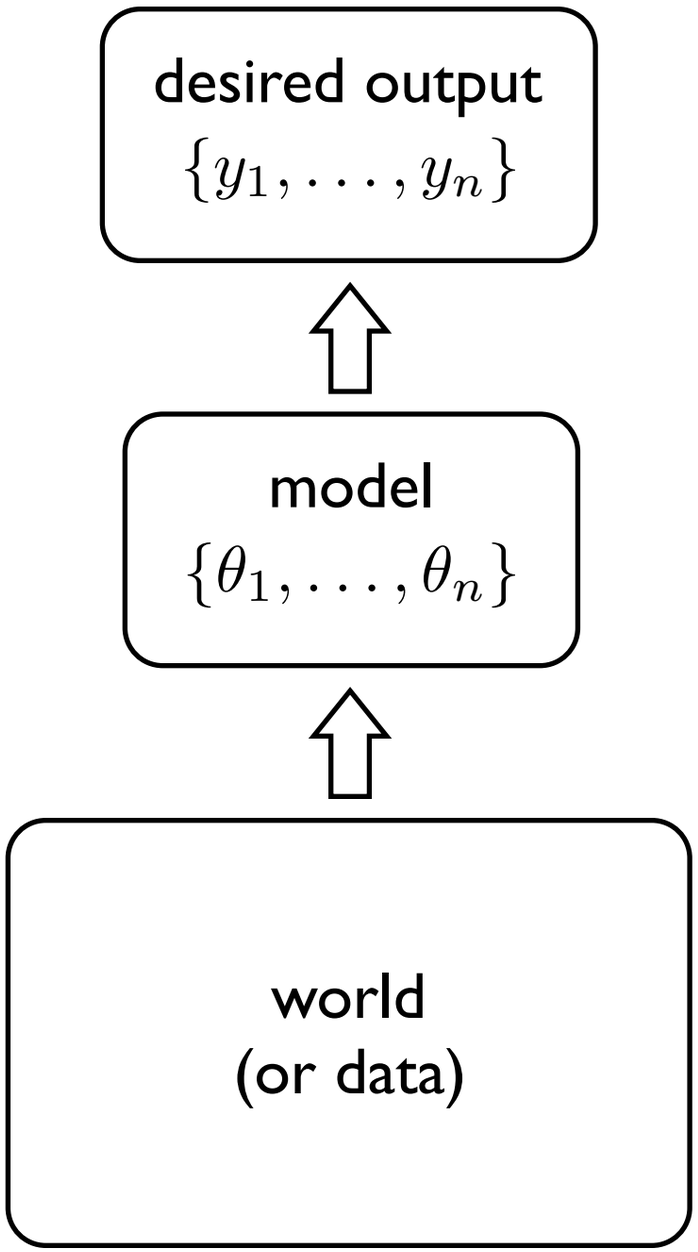
Clustering: Classification without labels

- In many situations we don't have labeled training data, only unlabeled data.
- Eg, in the iris data set, what if we were just starting and didn't know any classes?

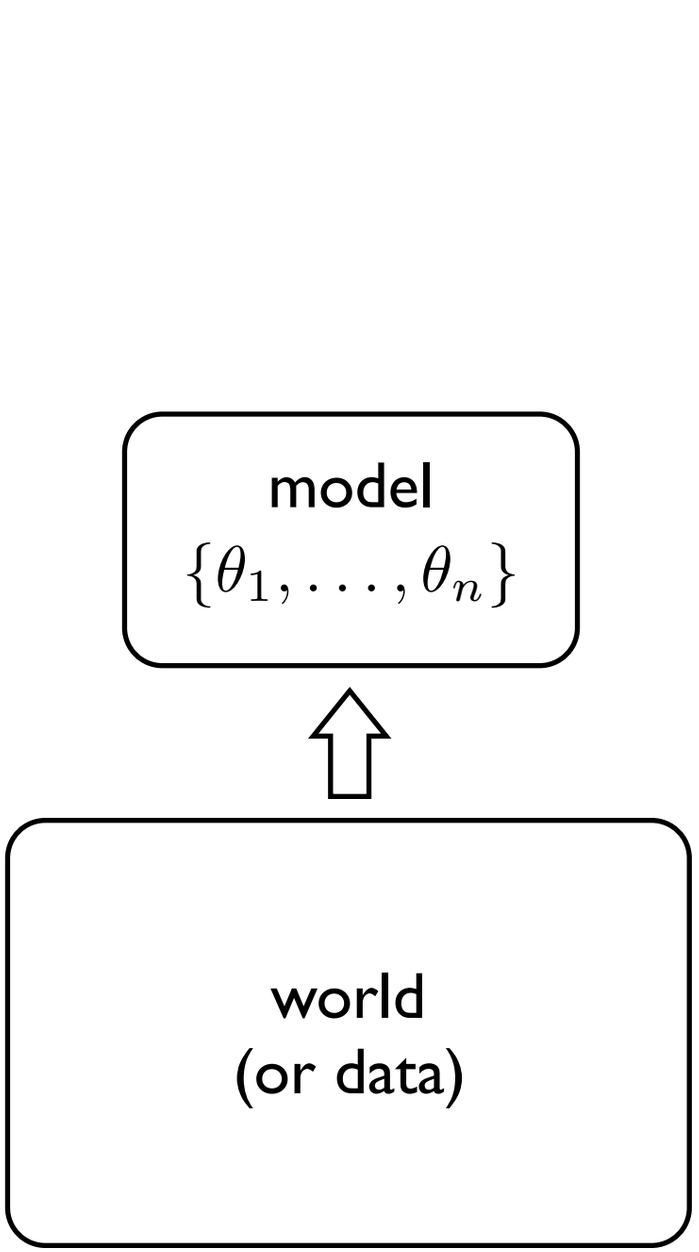


Types of learning

supervised



unsupervised



reinforcement

A different approach to classification

- Nearby points are likely to be members of the same class.
- What if we used the points themselves to classify?

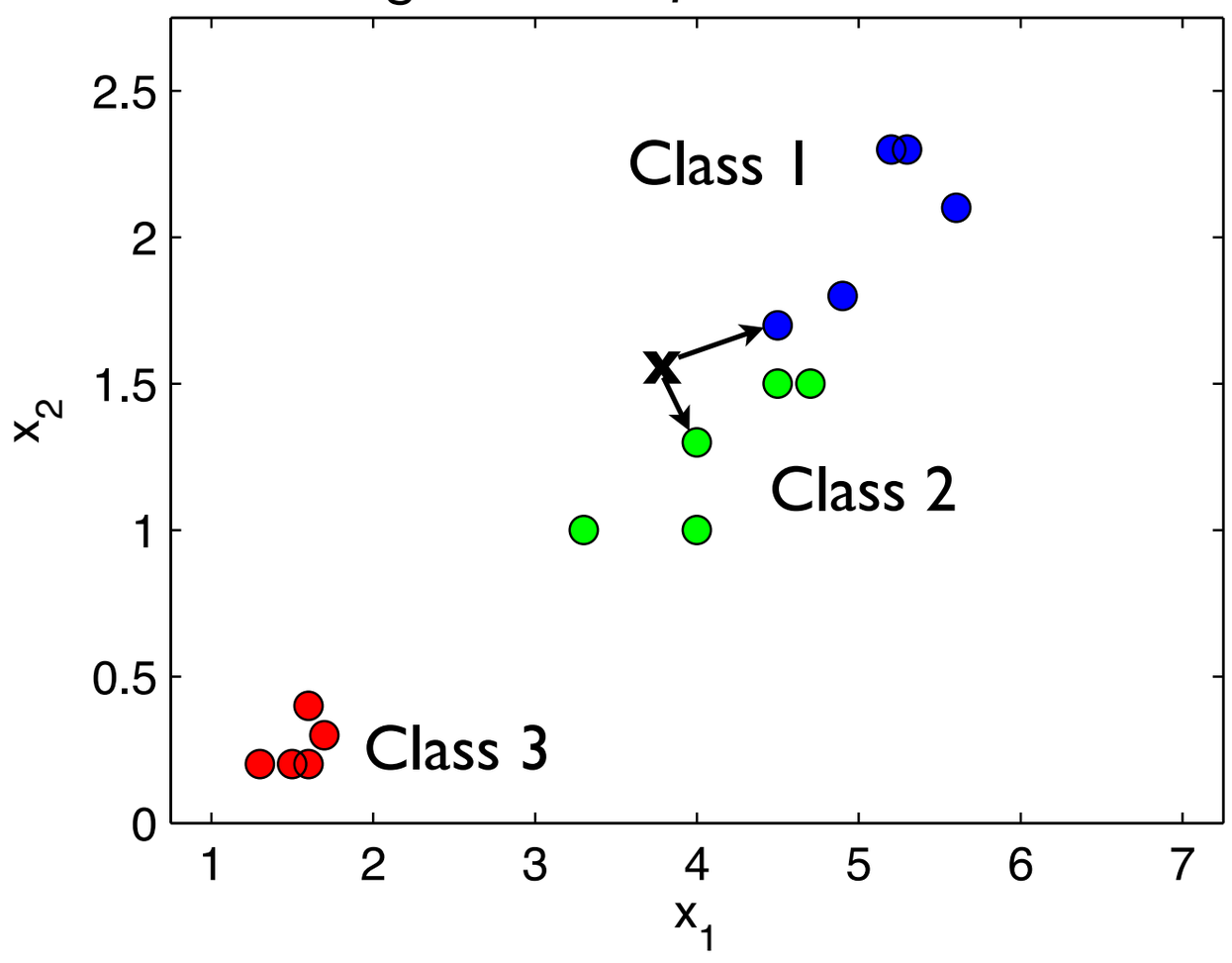
  classify $\mathbf{x}$ *in* C_k *if* $\mathbf{x}$ *is "similar" to a point we already know is in* C_k.

- Eg: unclassified point $\mathbf{x}$ is more similar Class 2 than Class 1.
- Issue: How to define "similar" ? Simplest is Euclidean distance:

$$d(\mathbf{x}, \mathbf{y}) = \sum_i (x_i - y_i)^2$$

- Could define other metrics depending on application, e.g. text documents, images, etc.

Nearest neighbor classification on the iris dataset

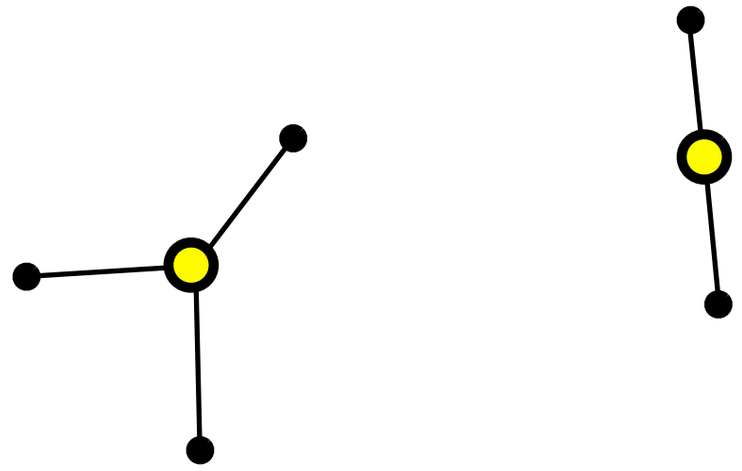


Potential advantages:

- don't need an explicit model
- the more examples the better
- might handle more complex classes
- easy to implement
- "no brain on part of the designer"

k-means clustering

- Idea: try to estimate k cluster centers by minimizing "distortion"
- Define distortion as:



$$D = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk} \parallel \mathbf{x}_n - \boldsymbol{\mu}_k \parallel^2$$

$$r_{nk} = 1 \text{ if } \mathbf{x}_n \in \text{cluster } k, \text{ 0 otherwise.}$$

- r_{nk} is 1 for the closest cluster mean to $\mathbf{x}_n$.
- Each point $\mathbf{x}_n$ is the minimum distance from its closet center.
- How do we learn the cluster means?
- Need to derive a ***learning rule***.

Deriving a learning rule for the cluster means

- Our objective function is:

$$D = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \parallel \mathbf{x}_n - \boldsymbol{\mu}_k \parallel^2$$

- Differentiate w.r.t. to the mean (the parameter we want to estimate):

$$\frac{\partial D}{\partial \boldsymbol{\mu}_k} = 2 \sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

- We know the optimum is when

$$\frac{\partial D}{\partial \boldsymbol{\mu}_k} = 2 \sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k) = 0$$

- Here, we can solve for the mean:

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}$$

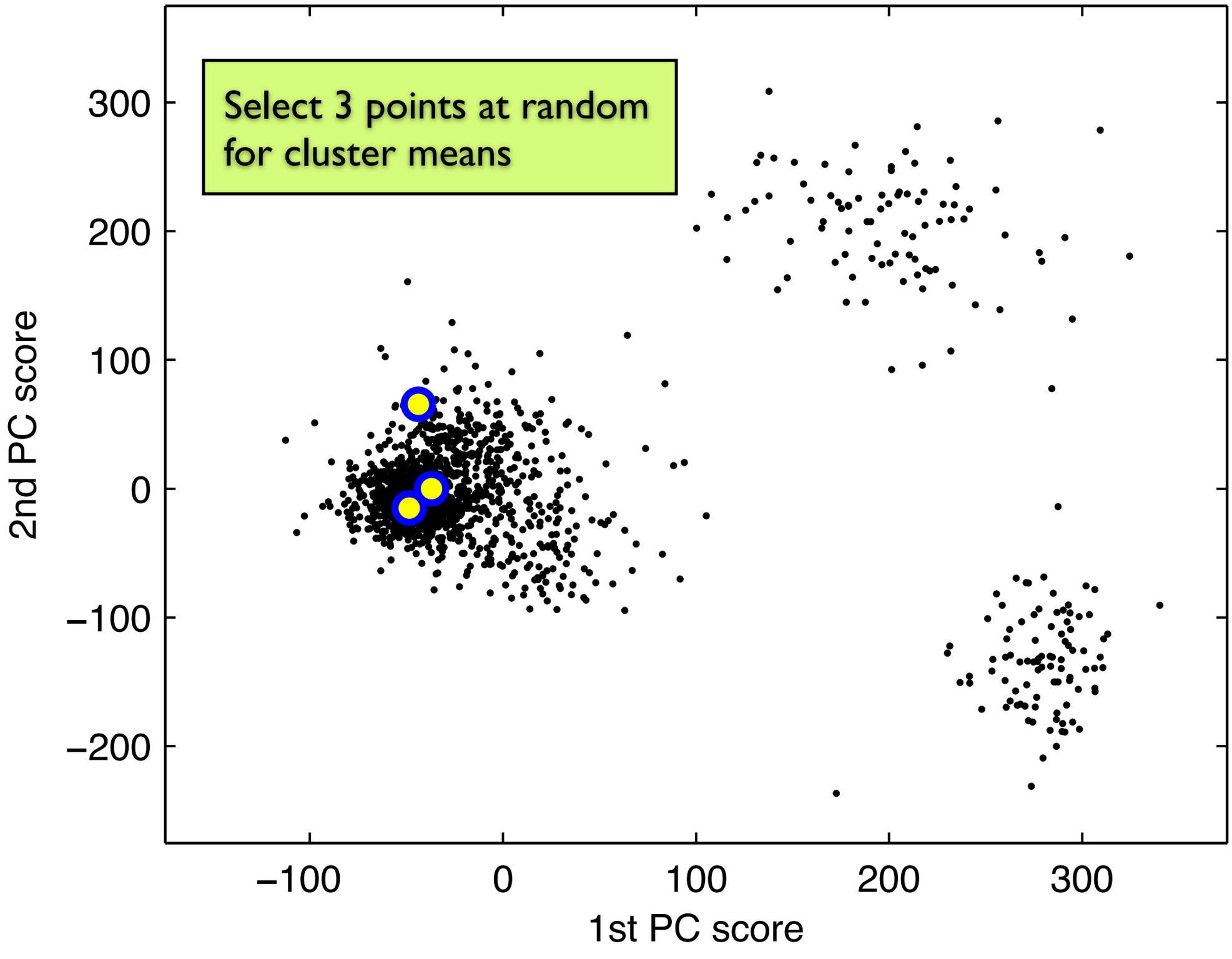
- This is simply a weighted mean for each cluster.

- Thus we have a simple estimation algorithm (*k-means clustering*)

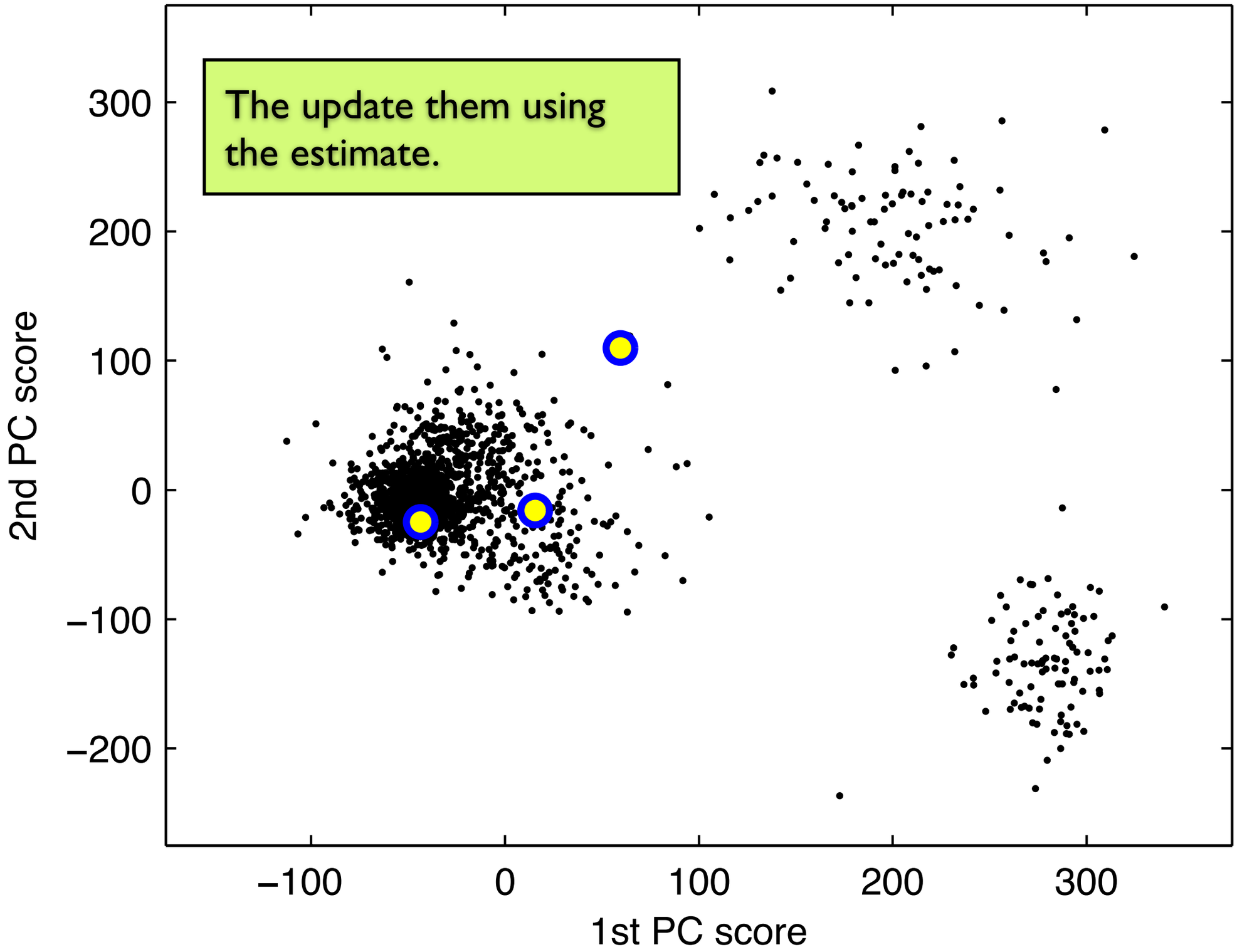
  1. select k points at random
  2. estimate (update) means
  3. repeat until converged

- convergence (to a local minimum) is guaranteed

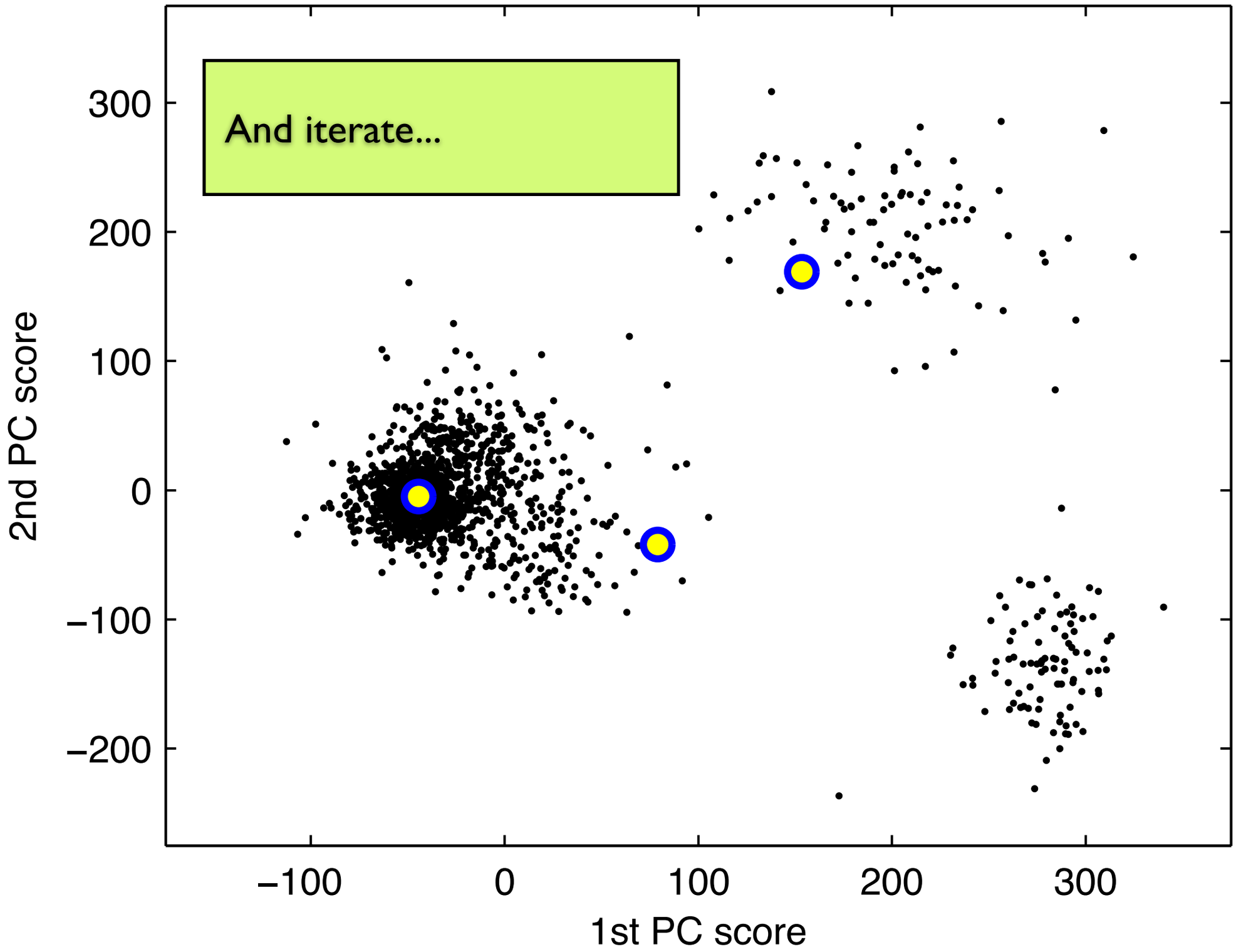
k-means clustering example



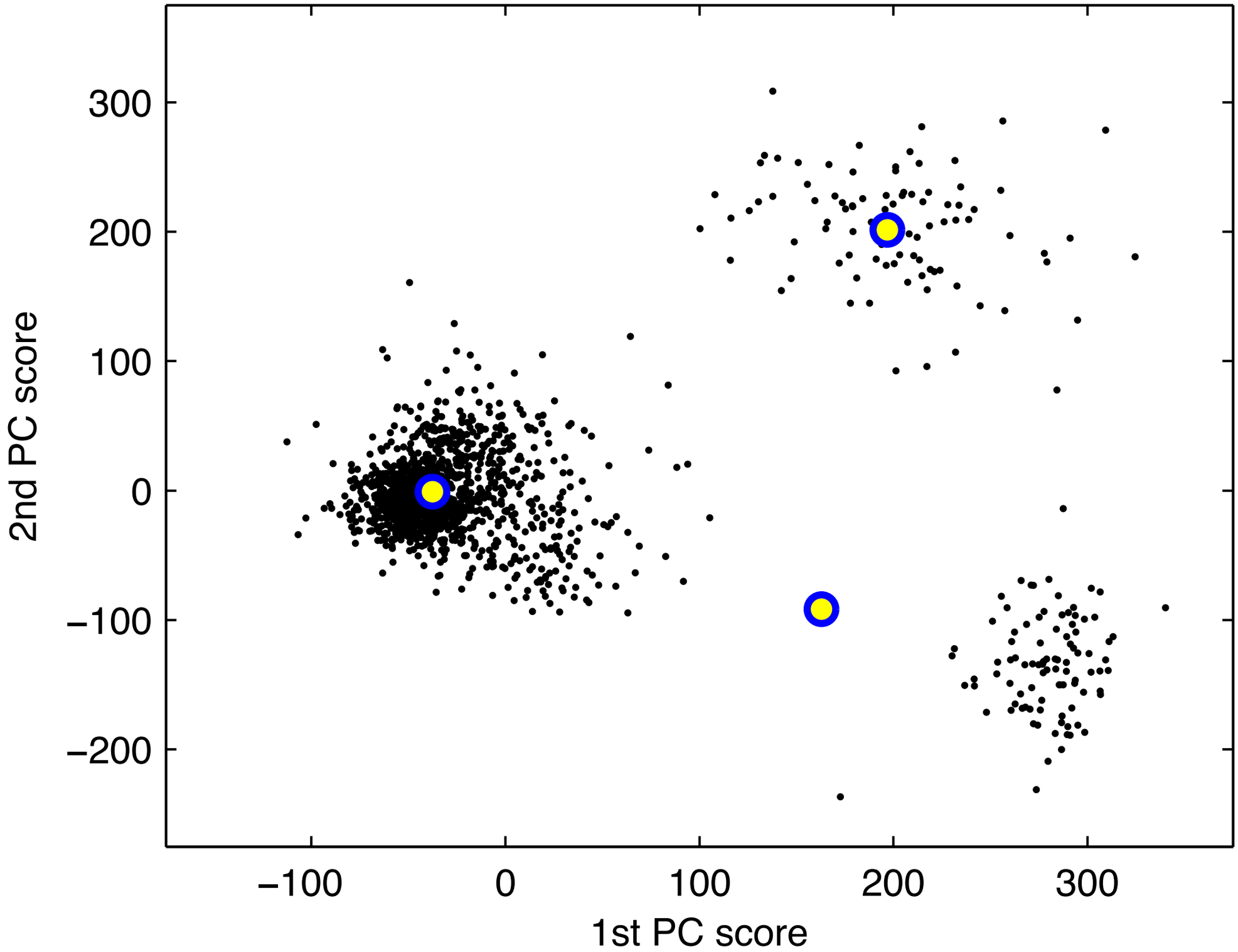
k-means clustering example



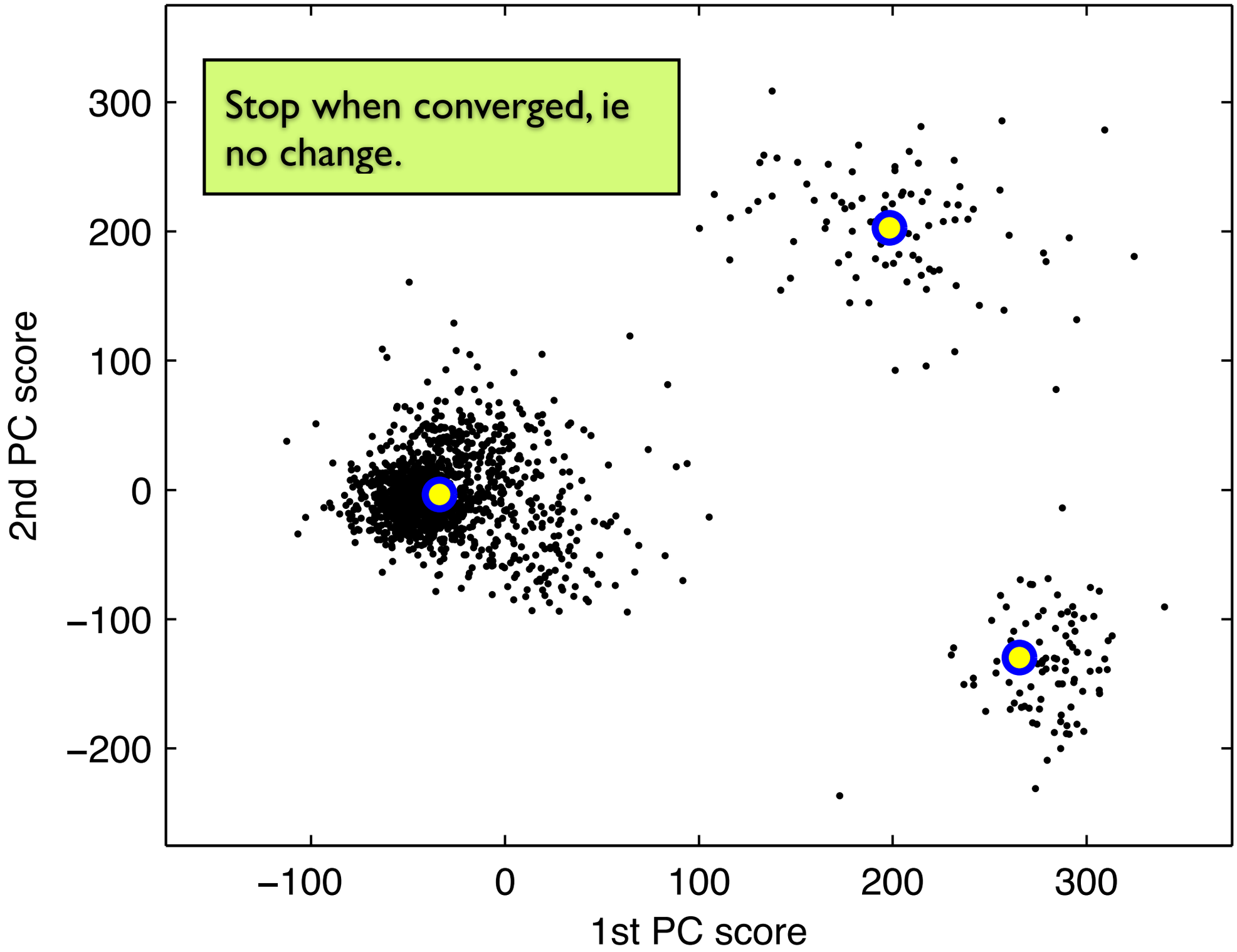
k-means clustering example



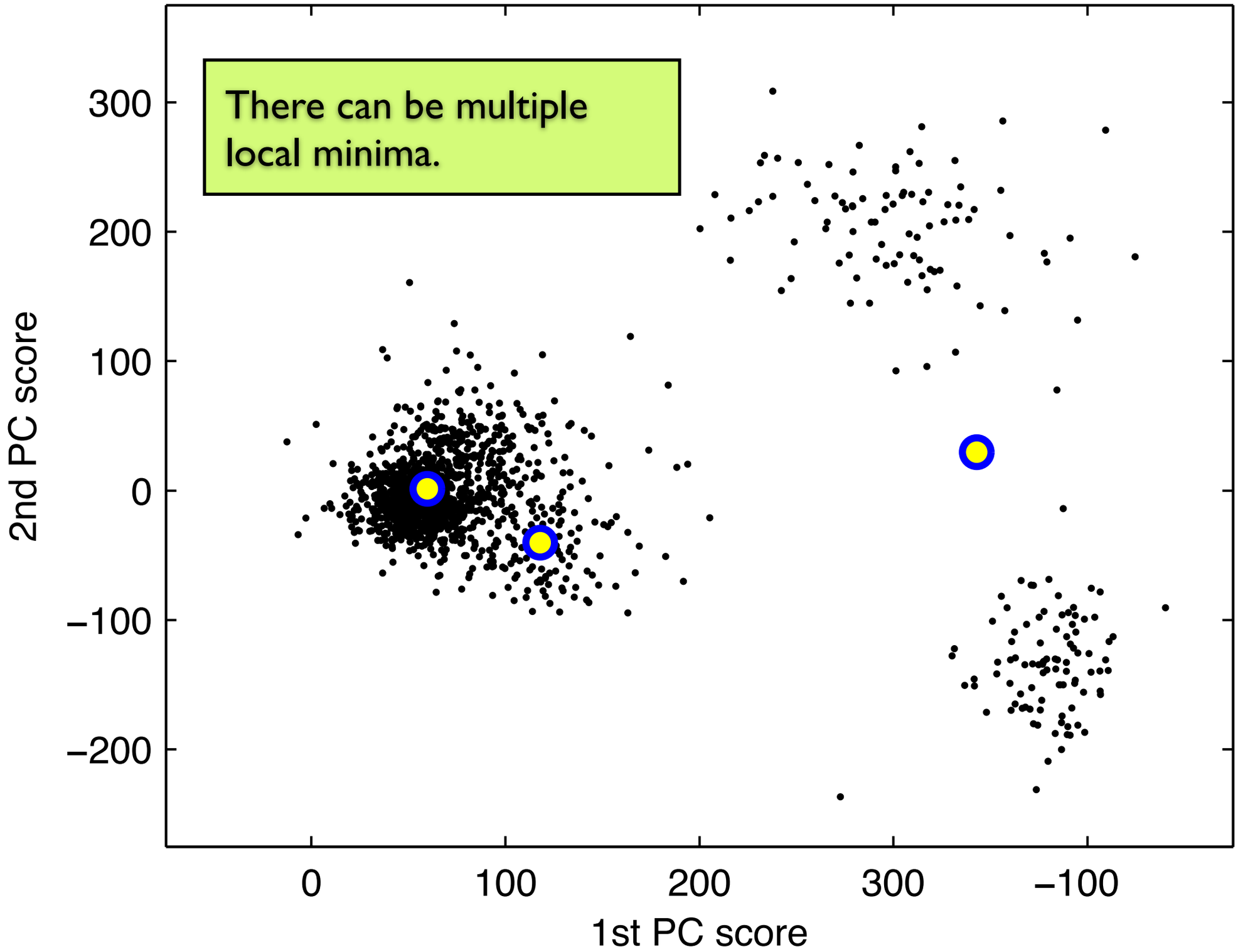
k-means clustering example



k-means clustering example

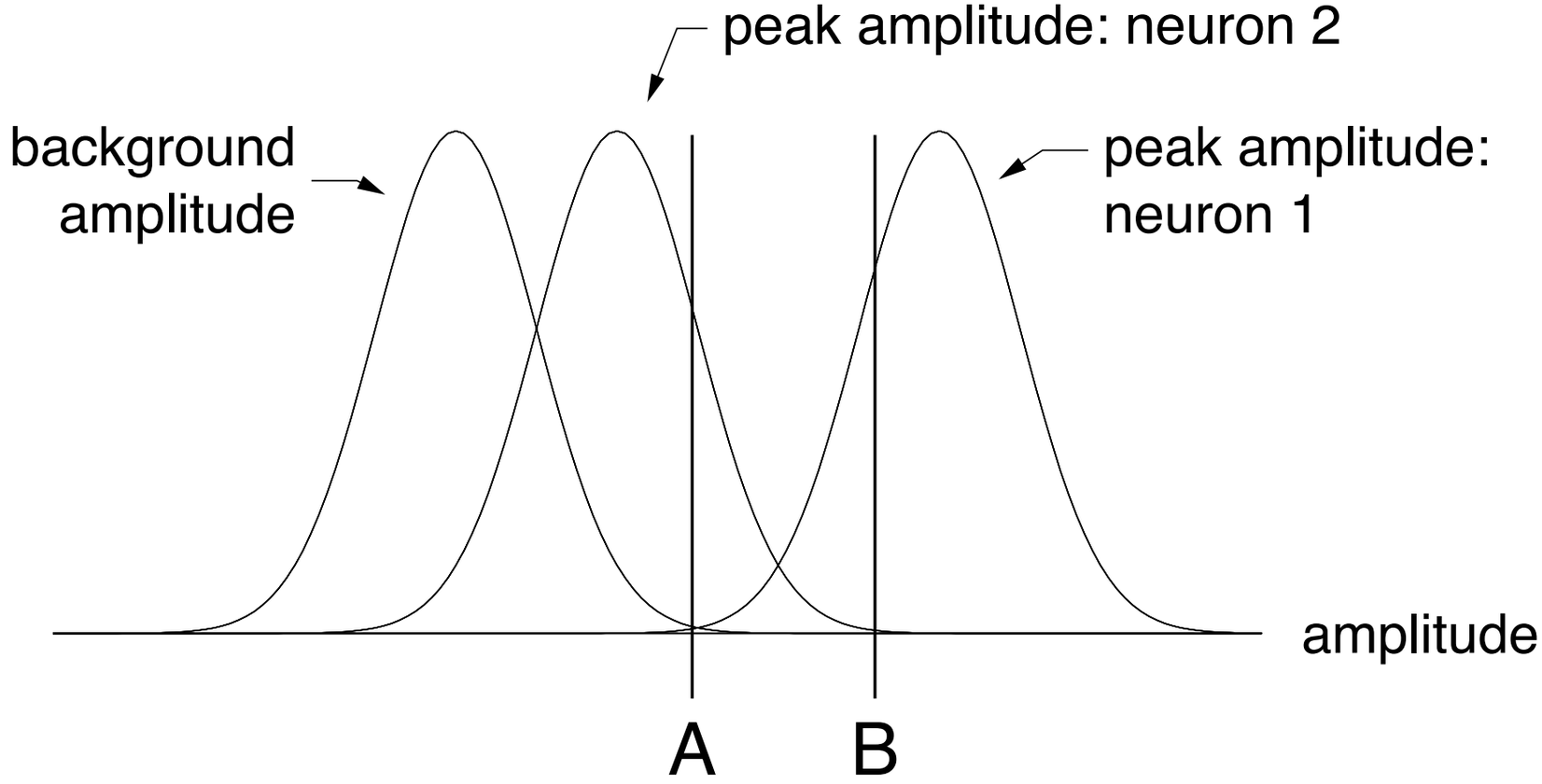


An example of a local minimum



A probabilistic interpretation: Gaussian mixture models

- We've already seen a one-dimensional version
- This example has three classes: neuron 1, neuron 2, and background noise.
- Each can be modeled as a Gaussian
- Any given data point comes from just *one* Gaussian
- The whole set of data is modeled by a *mixture* of three Gaussians
- How do we model this?



The Gaussian mixture model density

- The likelihood of the data given a particular class c_k is given by

$$p(x|c_k, \mu_k, \Sigma_k)$$

- x is the spike waveform, μ_k and Σ_k are the mean and covariance for class c_k.

- The marginal likelihood is computed by summing over the likelihood of the K classes

$$p(x|\theta_{1:\mathrm{K}}) = \sum_{k=1}^{K} p(x|c_k, \theta_k)p(c_k)$$

- $\theta_{1:\mathrm{K}}$ defines the parameters for all of the classes, $\theta_{1:\mathrm{K}} = \{\mu_1, \Sigma_1, \ldots, \mu_K, \Sigma_K\}$.

- $p(c_k)$ is the probability of the kth class, with $\sum_k p(c_k) = 1$.

- What does this mean in this example?

Multivariate Gaussians

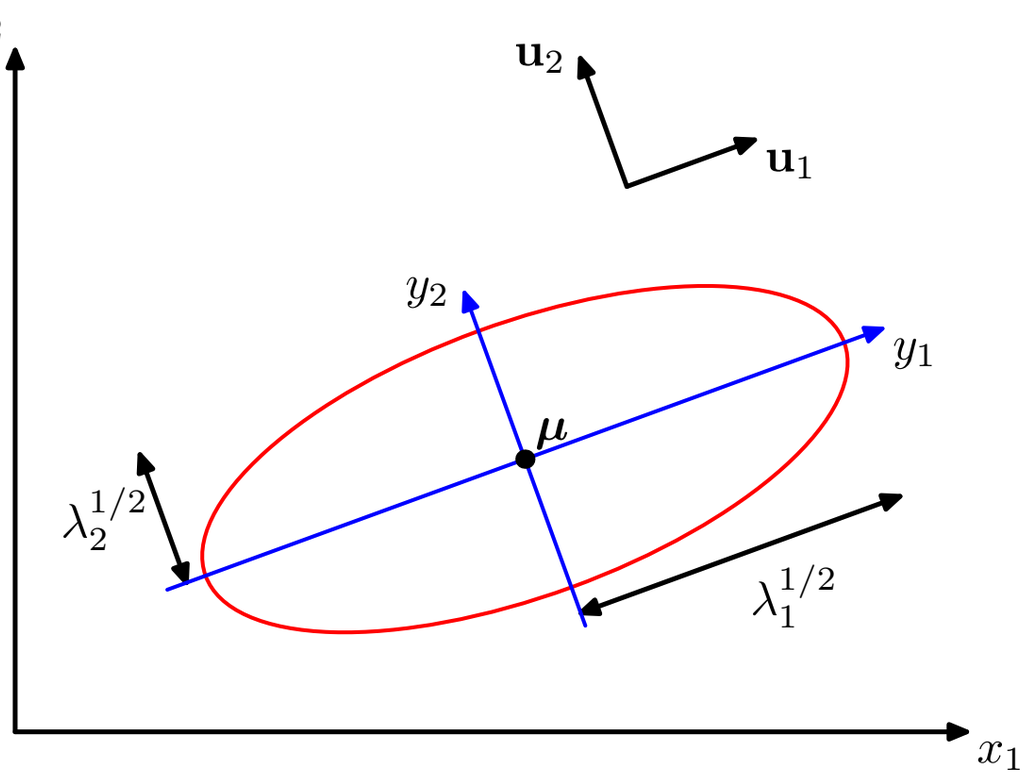
- Recall the univariate Gaussian

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

- The multivariate Gaussian is defined as follows:

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

- This can be interpreted as transforming into a new coordinate system defined by the eigenvectors (derived on board).



Some nice properties of Gaussians

- Partition Gaussian variable as follows:

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix} \quad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{bmatrix}$$

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{bmatrix}$$

- Note: $\boldsymbol{\Lambda}_{aa} \neq \boldsymbol{\Sigma}_{aa}^{-1}$

- Conditional probability is also Gaussian, and a linear function of $\mathbf{x}_b$

$$p(\mathbf{x}_a|\mathbf{x}_b) = \mathcal{N}(\boldsymbol{\mu}_{a|b}, \boldsymbol{\Sigma}_{a|b})$$

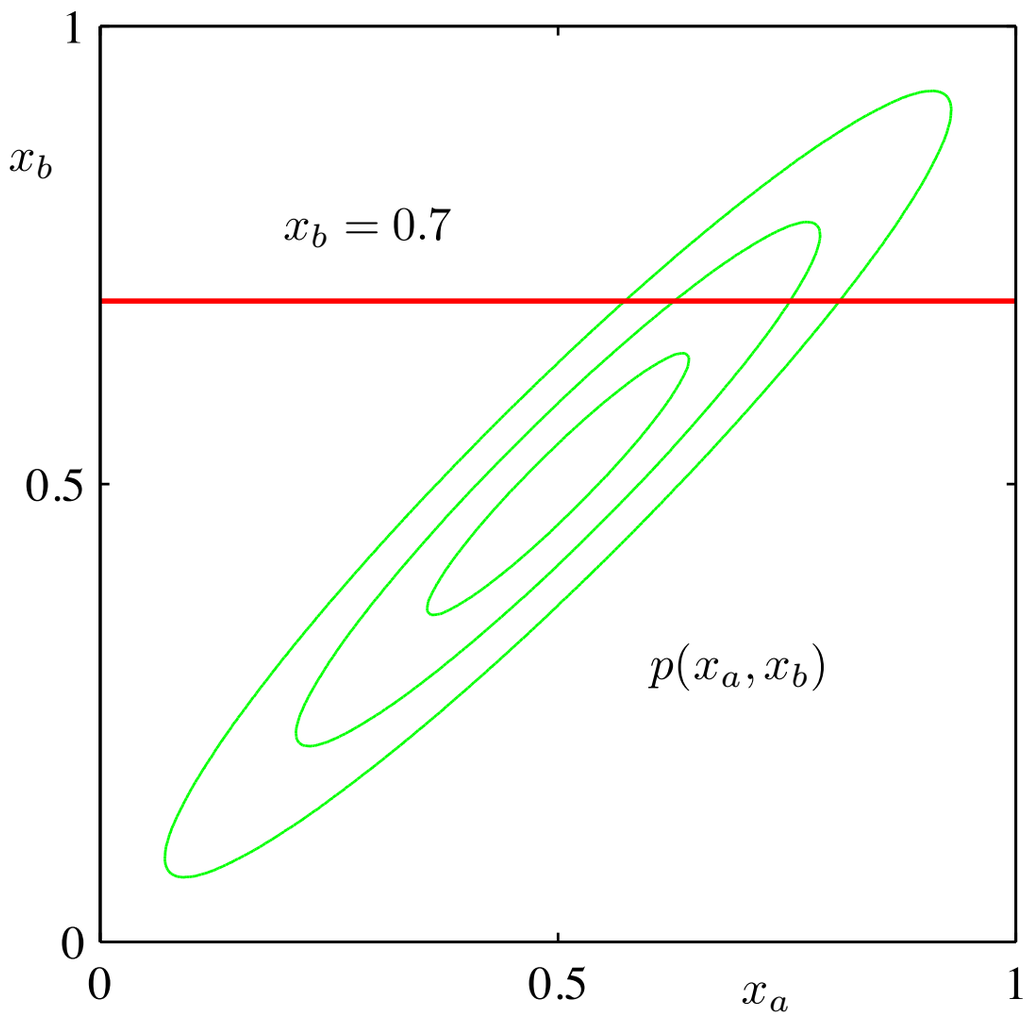
$$\begin{aligned} \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b) \\ &= \boldsymbol{\mu}_a + \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_a - \boldsymbol{\mu}_b) \end{aligned}$$

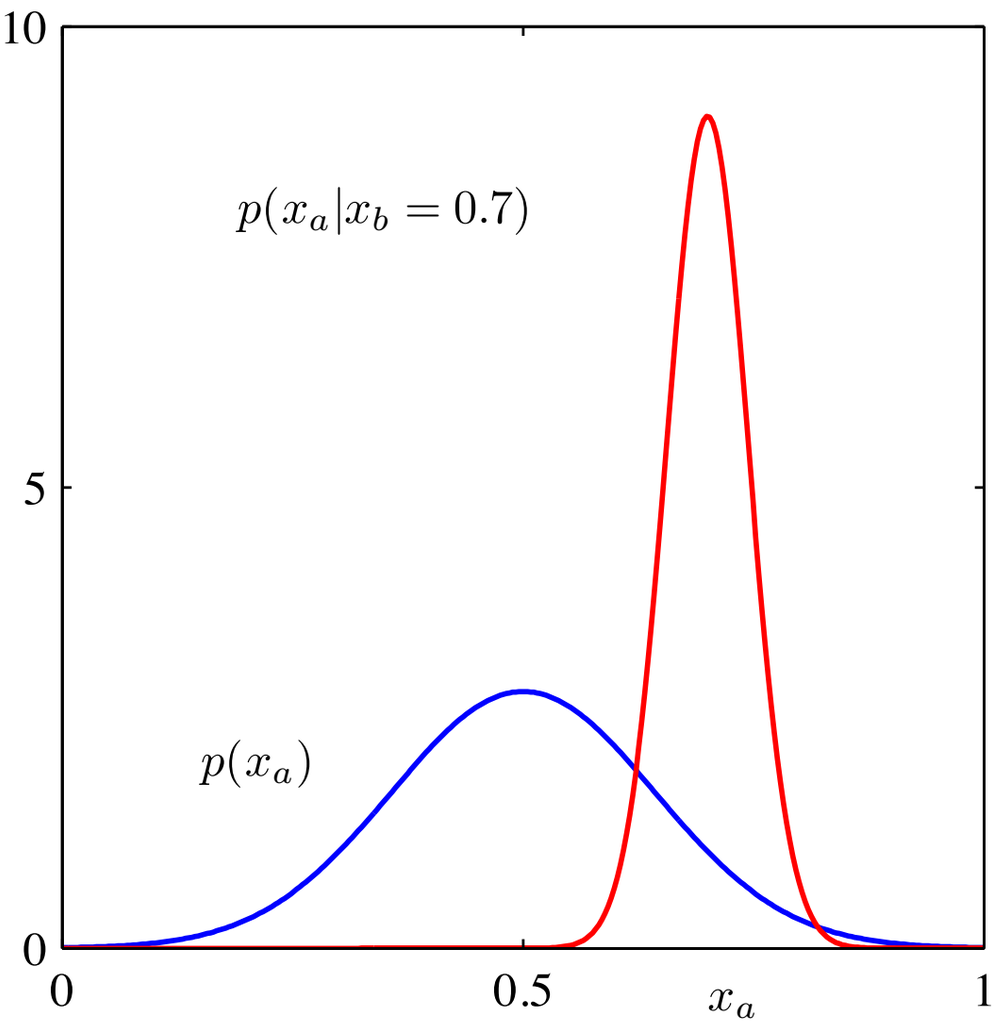
$$\begin{aligned} \boldsymbol{\Sigma}_{a|b} &= \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba} \\ &= \boldsymbol{\Lambda}_{aa}^{-1} \end{aligned}$$

- As is the marginal probability:

$$\begin{aligned} p(\mathbf{x}_a) &= \int p(\mathbf{x}_a, \mathbf{x}_b) d\mathbf{x}_b \\ &= \mathcal{N}(\mathbf{x}_a|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa}) \end{aligned}$$

Marginal and conditional Gaussian distributions





Bayes theorem for Gaussian variables

- Can we derive

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{x})}$$

where $\mathbf{y}$ is a linear function of $\mathbf{x}$:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$$
$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1})$$

- This is an example of a *linear Gaussian model.*

- Since the prior and likelihood are Gaussian, the posterior distribution is also Gaussian:

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}})$$
$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma})$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}$$

Hierarchical linear Gaussian graphical models

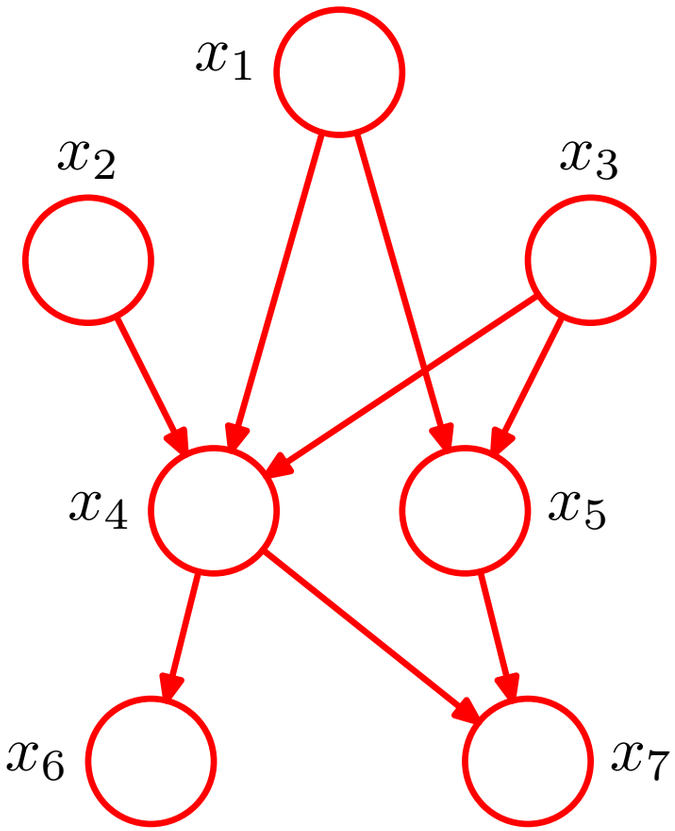
- First consider the 1-D case:

$$p(x_i|\mathrm{pa}(x_i) = j_p) = \mathcal{N}\left(x_i \middle| \sum_{j_p} w_{ij} x_j + b_i, v_i\right)$$

- x_i is a linear function of it's parents, which are also Gaussian random variables.
- More generally, we can let the node variables be vectors:

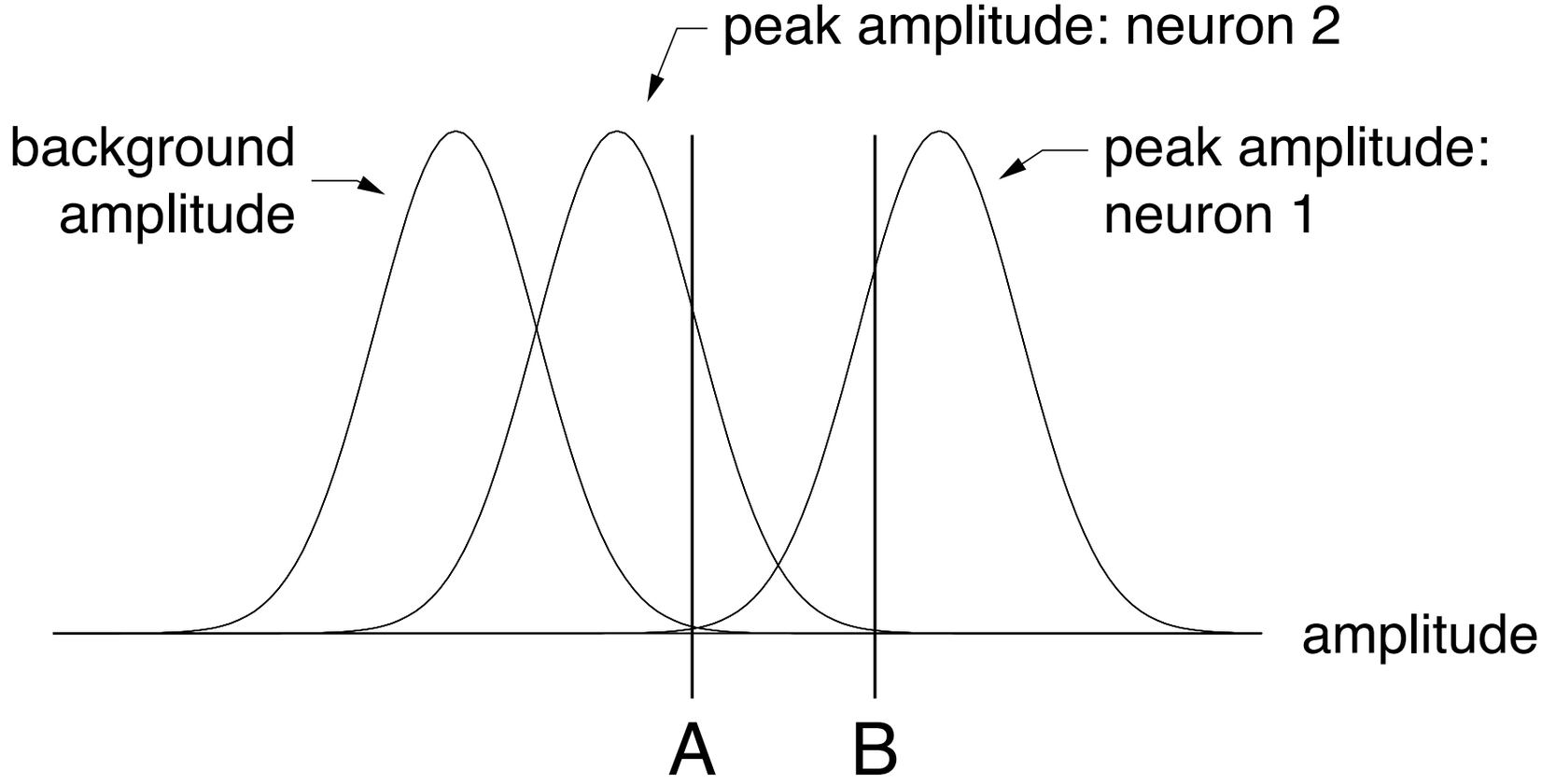
$$p(\mathbf{x}_i|\mathrm{pa}(\mathbf{x}_i) = j_p) = \mathcal{N}\left(\mathbf{x}_i \middle| \sum_{j_p} \mathbf{w}_{ij} \mathbf{x}_j + \mathbf{b}_i, \mathbf{\Sigma}_i\right)$$

- Because the variables are Gaussian, the relevant quantities for inference can be computed analytically.



A probabilistic interpretation: Gaussian mixture models

- We've already seen a one-dimensional version
- This example has three classes: neuron 1, neuron 2, and background noise.
- Each can be modeled as a Gaussian
- Any given data point comes from just *one* Gaussian
- The whole set of data is modeled by a *mixture* of three Gaussians
- How do we model this?



The Gaussian mixture model density

- The likelihood of the data given a particular class c_k is given by

$$p(x|c_k, \mu_k, \Sigma_k)$$

- x is the spike waveform, μ_k and Σ_k are the mean and covariance for class c_k.

- The marginal likelihood is computed by summing over the likelihood of the K classes

$$p(x|\theta_{1:\mathrm{K}}) = \sum_{k=1}^{K} p(x|c_k, \theta_k)p(c_k)$$

- $\theta_{1:\mathrm{K}}$ defines the parameters for all of the classes, $\theta_{1:\mathrm{K}} = \{\mu_1, \Sigma_1, \ldots, \mu_K, \Sigma_K\}$.

- $p(c_k)$ is the probability of the kth class, with $\sum_k p(c_k) = 1$.

- What does this mean in this example?

Bayesian classification

- How do we determine the class c_k from the data x ?

- Again use Bayes' rule

$$p(c_k|x^{(n)}, \theta_{1:\mathrm{K}}) = p_{k,n} = \frac{p(x^{(n)}|c_k, \theta_k)p(c_k)}{\sum_k p(x^{(n)}|c_k, \theta_k)p(c_k)}$$

- This tells is the probability that waveform $x^{(n)}$ came from class c_k.

- Let's review the process:
  1. *define* model of problem ✓
  2. *derive* posterior distributions and estimators ✓
  3. *estimate* parameters from data ??
  4. *evaluate* model accuracy

How do we do this?

Estimating the parameters: fitting the model density to the data

- The objective of density estimation is to maximize the likelihood of the data

- If we assume the samples are independent, the data likelihood is just the product of the *marginal* likelihoods

$$p(x_{1:\mathrm{N}}|\theta_{1:\mathrm{K}}) = \prod_{n=1}^{N} p(x_n|\theta_{1:\mathrm{K}})$$

- The class parameters are determined by optimization.

- Is far more practical to optimize the log-likelihood.

- One elegant approach to this is the EM algorithm.

The EM algorithm

- EM stands for Expectation-Maximization, and involves two steps that are iterated. For the case of a Gaussian mixture model:

  1. E-step: Compute $p_{n,k} = p(c_k|x^{(n)}, \theta_{1:\mathrm{K}})$. Let $p_k = \sum_n p_{n,k}$
  2. M-step: Compute new mean, covariance, and class prior for each class:

Something can go bad here...

$$\mu_k \leftarrow \sum_n p_{n,k} x^{(n)} / p_k$$

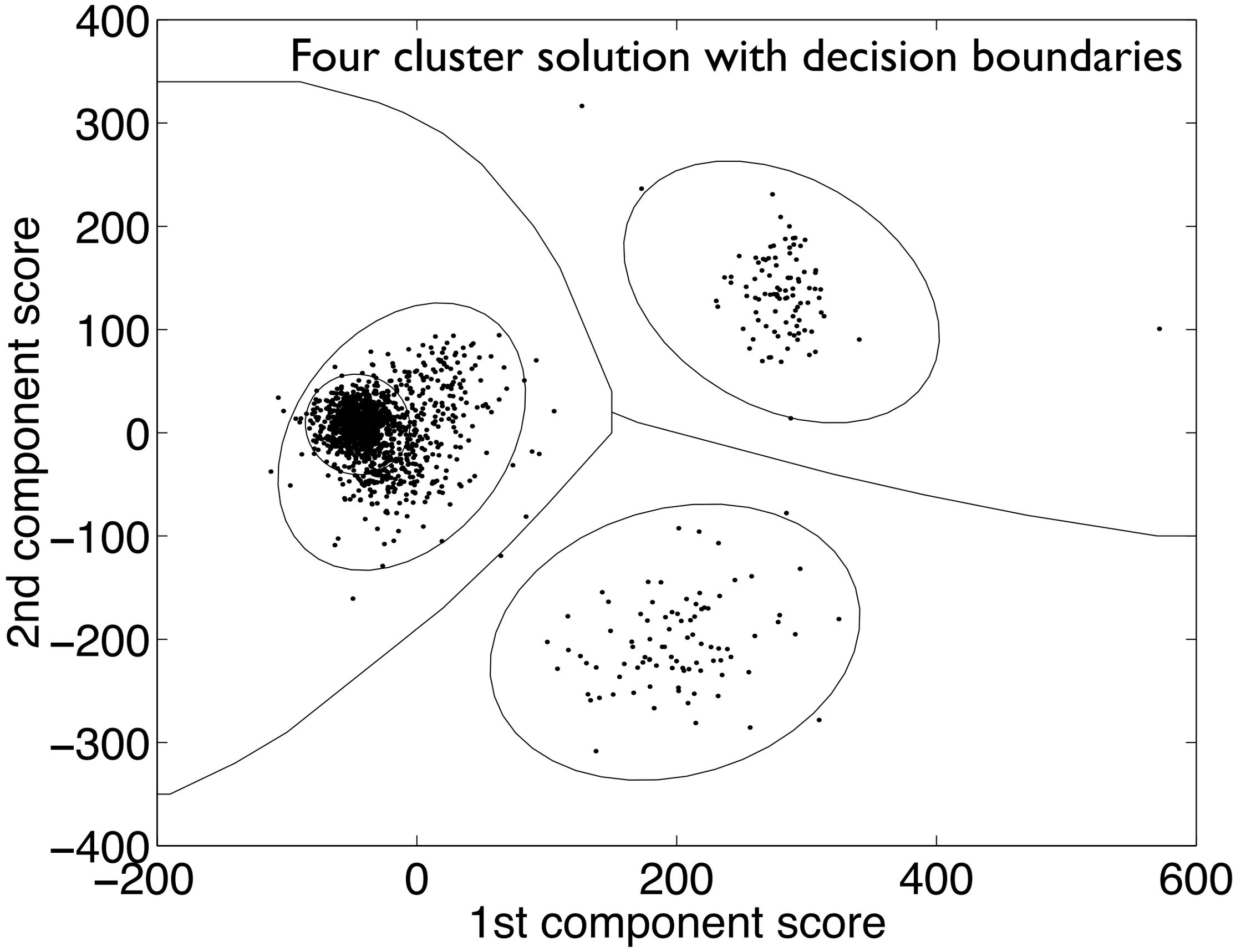
What if these are zero?

$$\Sigma_k \leftarrow \sum_n p_{n,k} (x^{(n)} - \mu_k)(x^{(n)} - \mu_k)^T / p_k$$

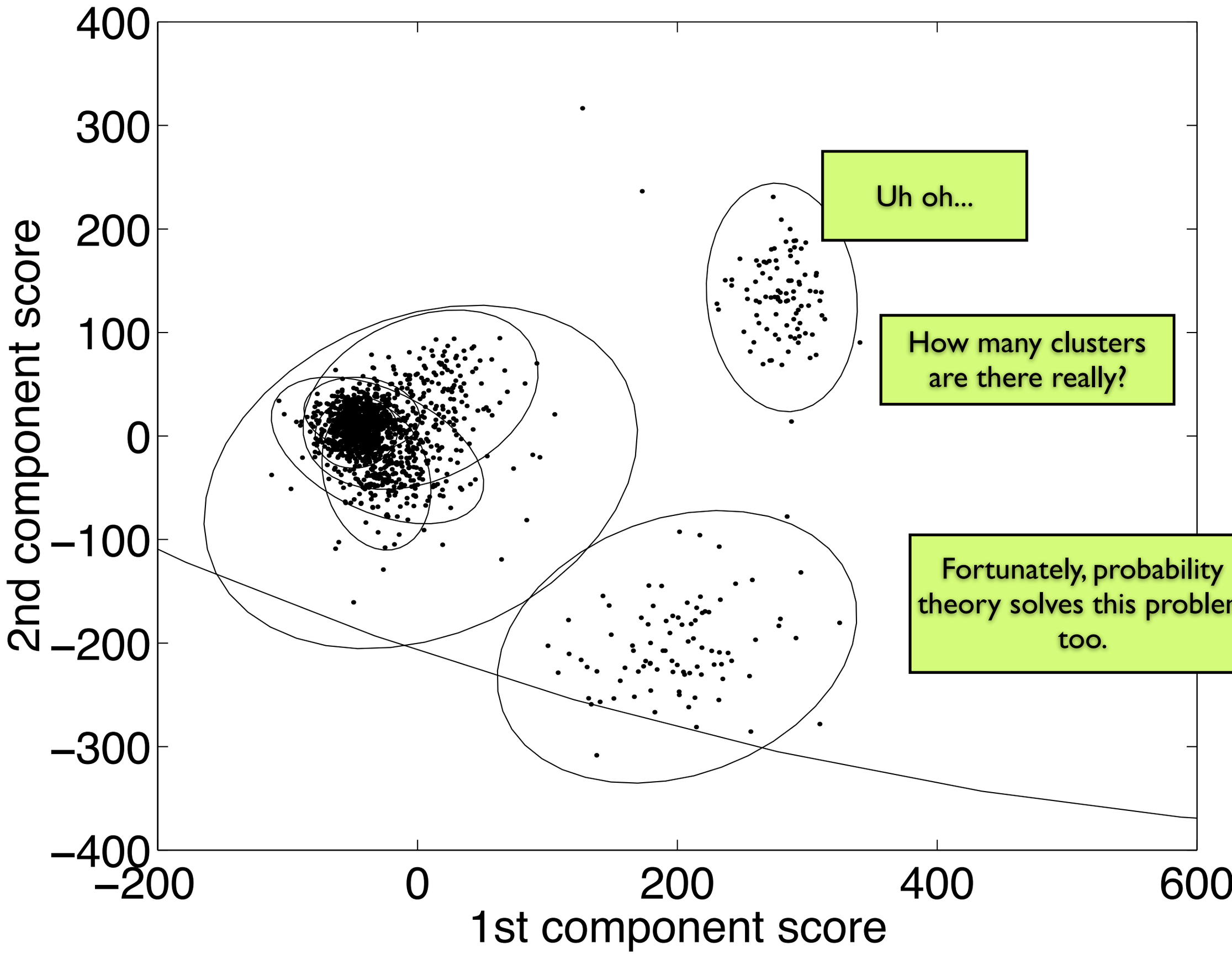
$$p(c_k) \leftarrow p_k$$

- This is just the sample mean and covariance, weighted by the class conditional probabilities $p_{n,k}$.

- Derived by solving setting log-likelihood gradient to zero (i.e. the maximum).


Four cluster solution with decision boundaries
400
300
200
100
0
−100
−200
−300
−400
2nd component score
−200
0
200
400
600
1st component score

B


400
300
200
100
0
−100
−200
−300
−400
2nd component score
−200
0
200
400
600
1st component score
Uh oh...
How many clusters are there really?
Fortunately, probability theory solves this problem too.

Bayesian model comparison

- Let M_K represents a model with K classes. (Here we will assume that we can choose the best among all such models, but this assumption is not necessary). How do we evaluate the probability of model M_K? Bayes rule again.

- We start with our existing model, but add a term to represent the model itself. Also, we marginalize out the dependency on the parameters, because we want the result independent of any specific value. Letting $\mathcal{X} = x^{(1:\mathrm{N})}$, we have

$$p(M_K|\mathcal{X}) = \frac{p(\mathcal{X}|M_K)p(M_K)}{p(\mathcal{X})}$$

- The denominator is constant across models, and if we assume all models are equally probable a priori, the only data-dependent term is $\mathcal{X} = x^{(1:\mathrm{N})}$.

- How do we compute $p(\mathcal{X}|M_K)$? We've encountered this before.

Evaluating the model evidence

- $p(\mathcal{X}|M_K)$ is just the normalizing constant for the posterior for parameters

$$p(\theta_K|\mathcal{X}, M_K) = \frac{p(\mathcal{X}|\theta_K, M_K)p(M_K)}{p(\mathcal{X}|M_K)}$$

- (slight change of notation: θ_K represents all parameters for model K.)

- The normalizing constant here is evaluated just like before by marginalization

$$p(\mathcal{X}|M_K) = \int p(\mathcal{X}|\theta_K, M_K)p(M_K)d\theta_K$$

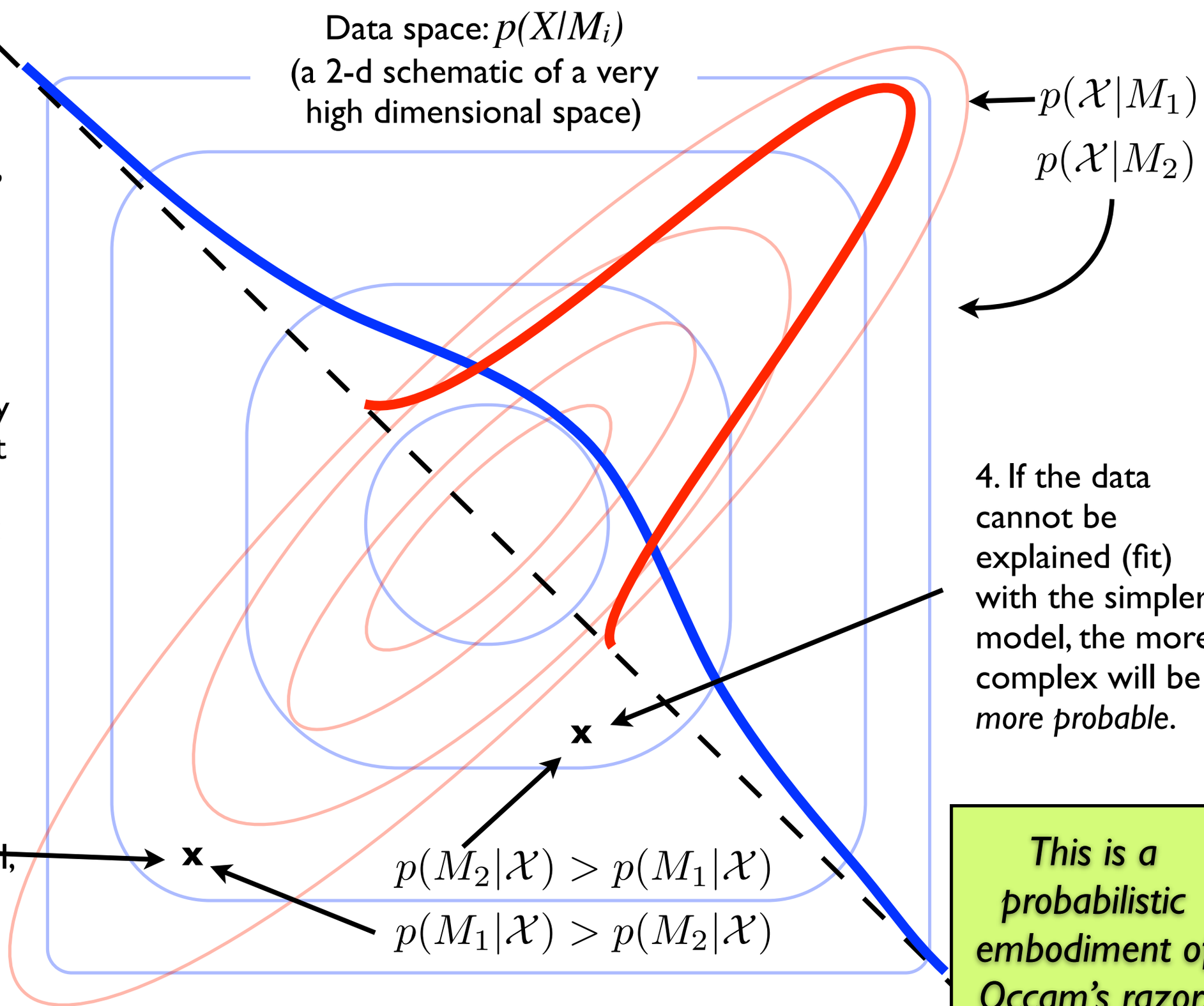
- Evaluating this term is practically a whole subfield of probability theory: Laplace's method, monte carlo integration, variational approximation, etc.

Schematic of Bayesian model comparison

1. M2 is has more degrees of freedom than M1, and therefore a larger set of data as higher probability

2. Both probability distributions must integrate to one, so more complex models are necessarily more spread out

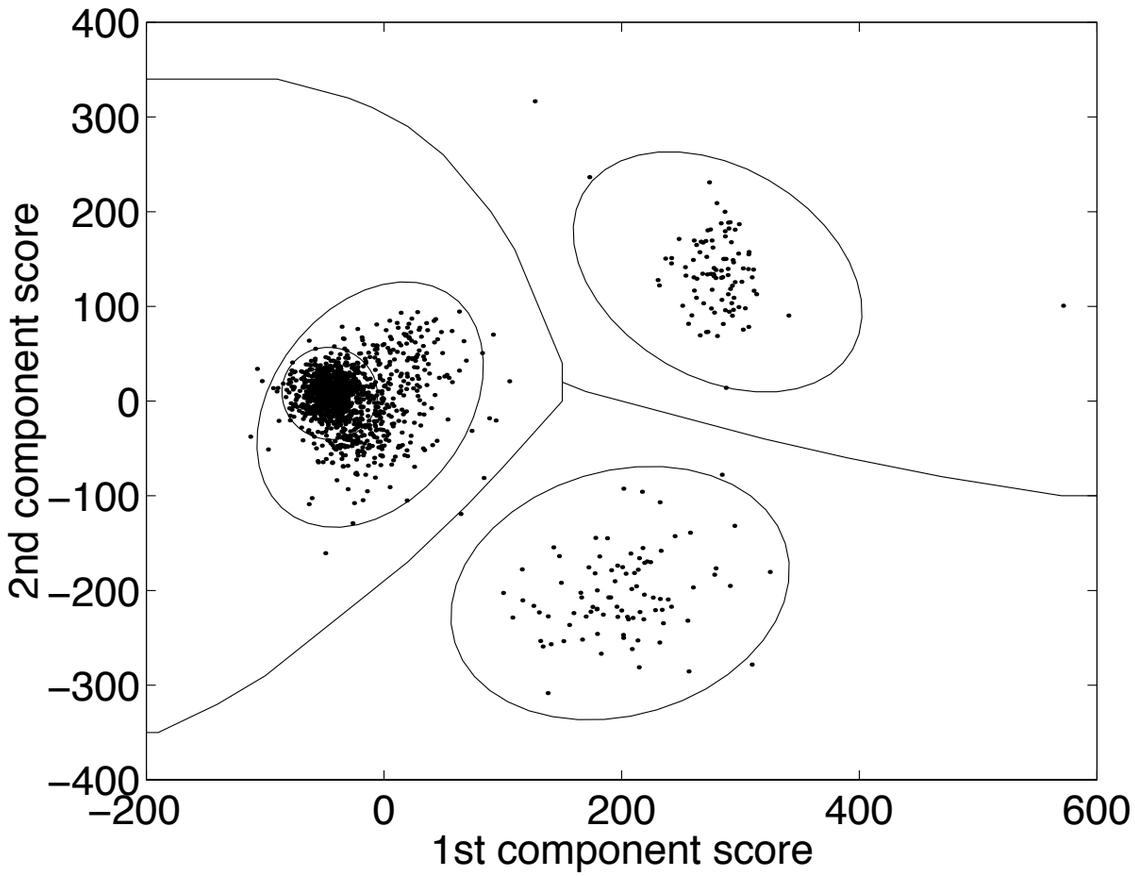
3. If the data falls within the plausible range of the simpler model, it will be *more probable*.

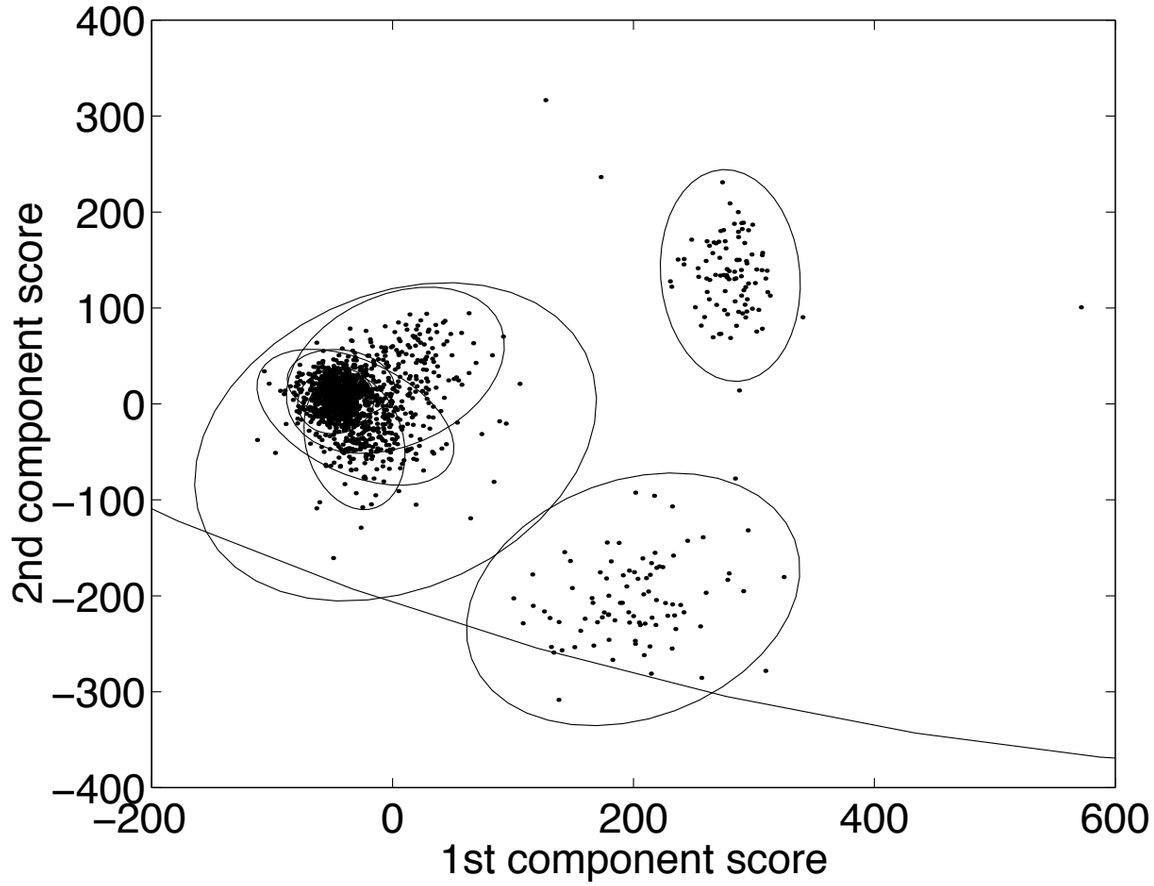


4. If the data cannot be explained (fit) with the simpler model, the more complex will be *more probable*.

This is a probabilistic embodiment of Occam's razor.

Back to the clusters





- Which model is more probable?
  - P(M9 | X) is exp(160) times greater than P(M4 | X).
  - Why might this not agree with our "intuitions"?
  - The conclusions are always only as valid as the model.
  - But P(M9 | X) is exp(16) times greater than P(M11 | X).
- False assumptions can lead to false conclusions.