Mixture Models

Prof. Dr. H. H. Takada

Quantitative Research – Itaú Asset Management
Institute of Mathematics and Statistics – University of São Paulo

Mixture Models

A mixture model is one in which a set of component models is combined to produce a richer model:

$$p(v) = \sum_{h=1}^{H} p(v|h)p(h)$$

The variable v is 'visible' or 'observable' and $h = 1, \ldots, H$ indexes each component model $p(v|h)$, along with its weight $p(h)$.

Clustering

Mixture models have a natural interpretation in terms of clustering with each state of h corresponding to a cluster model $p(v|h)$.

Graphical Model

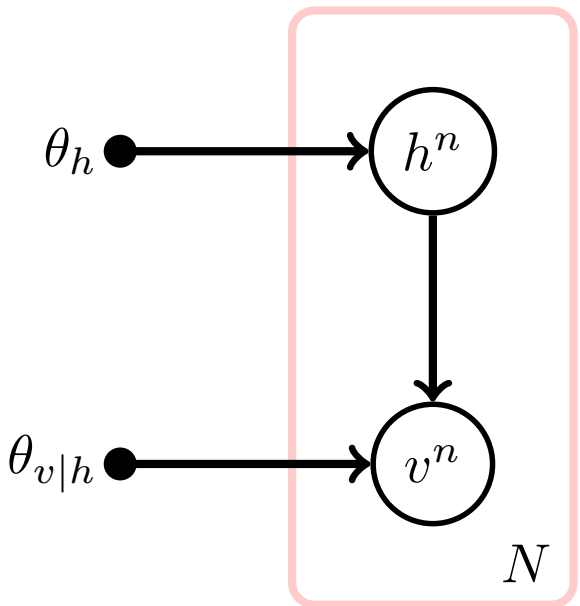


Figure: A mixture model has a trivial graphical representation as a DAG with a single hidden node, which can be in and one of H states, $i = 1 \ldots, H$. The parameters are assumed common across all datapoints.

Mixture of Independent Bernoulli

A simple model that can be used to cluster a set of binary vectors, $\mathbf{v}^n = (v_1^n, \ldots, v_D^n)^\mathsf{T}$, $v_i \in \{0, 1\}, n = 1, \ldots, N$ is

$$p(\mathbf{v}) = \sum_{h=1}^{H} p(h) \prod_{i=1}^{D} p(v_i|h)$$

where each term $p(v_i|h)$ is a Bernoulli distribution.

Parameters

We need to learn $p(v_i = 1|h = \mathsf{h})$ and $p(h = \mathsf{h})$.

EM training

For i.i.d. data, the energy is:

$$\sum_n \langle \log p(\mathbf{v}^n, h) \rangle_{p^{old}(h|\mathbf{v}^n)}$$

$$= \sum_n \sum_i \langle \log p(v_i^n | h) \rangle_{p^{old}(h|\mathbf{v}^n)} + \sum_n \langle \log p(h) \rangle_{p^{old}(h|\mathbf{v}^n)}$$

Finding the updates

Since the energy terms are simply related to KL divergences, we can immediately write down the EM updates.

Updates

M-step

$$p^{new}(v_i = 1|h = j) = \frac{\sum_n \mathbb{I}\left[v_i^n = 1\right] p^{old}(h = j|\mathbf{v}^n)}{\sum_n \mathbb{I}\left[v_i^n = 1\right] p^{old}(h = j|\mathbf{v}^n) + \sum_n \mathbb{I}\left[v_i^n = 0\right] p^{old}(h = j|\mathbf{v}^n)}$$

$$p^{new}(h = j) = \frac{\sum_n p^{old}(h = j|\mathbf{v}^n)}{\sum_{h'} \sum_n p^{old}(h'|\mathbf{v}^n)}$$

E-step

$$p^{old}(h = j|\mathbf{v}^n) \propto p(h = j) \prod_{i=1}^{D} p(v_i^n|h = j)$$

Clustering binary digits

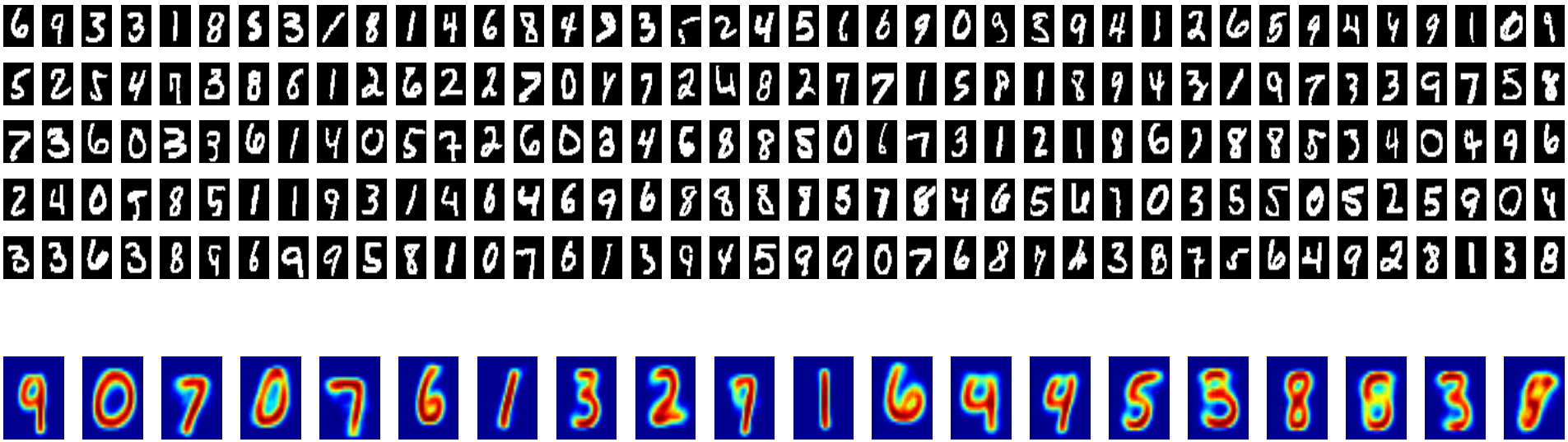


Figure: Top: a selection of 200 of the 5000 handwritten digits in the training set. Bottom: the trained cluster outputs $p(v_i = 1|h)$ for $h = 1, \ldots, 20$ mixtures. See `demoMixBernoulliDigits.m`.

Clustering Questionnaires

If an attribute i is missing for this model one simply removes the corresponding factor $p(v_i^n|h)$ from the algorithm.

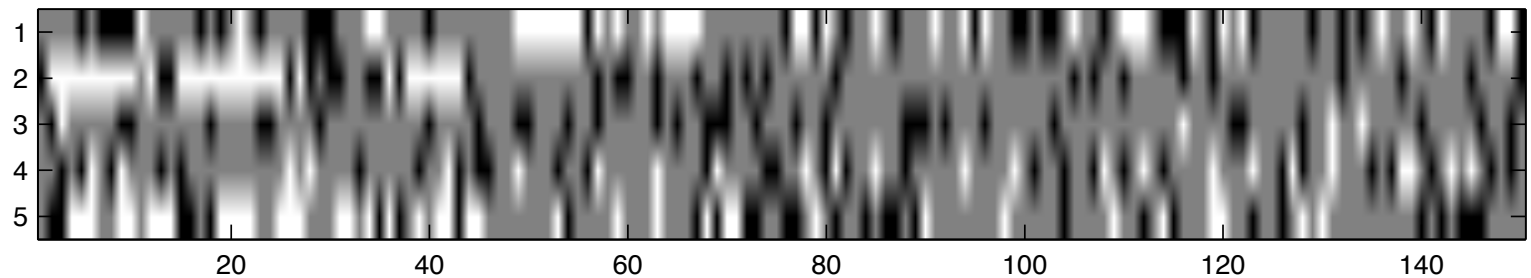


Figure: Data from questionnaire responses. 150 people were each asked 5 questions, with 'yes' (white) and 'no' (gray) answers. Black denotes that the absence of a response (missing data). This training data was generated by two component Binomial mixture. Missing data was simulated by randomly removing values from the dataset.

Gaussian Mixture Model

A D dimensional Gaussian distribution for a continuous variable $\mathbf{x}$ is

$$p(\mathbf{x}|\mathbf{m},\mathbf{S}) = \frac{1}{\sqrt{\det(2\pi\mathbf{S})}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^{\mathsf{T}}\mathbf{S}^{-1}(\mathbf{x}-\mathbf{m})\right\}$$

where $\mathbf{m}$ is the mean and $\mathbf{S}$ is the covariance matrix. A mixture of Gaussians is then

$$p(\mathbf{x}) = \sum_{i=1}^{H} p(\mathbf{x}|\mathbf{m}_i,\mathbf{S}_i)p(i)$$

where $p(i)$ is the mixture weight for component i.

Clustering

Mixture models have natural application in clustering data, where h indexes the cluster. This interpretation can be gained from considering how to generate a sample datapoint v from the model. First we sample a cluster h from $p(h)$, and then draw a visible state v from $p(v|h)$.

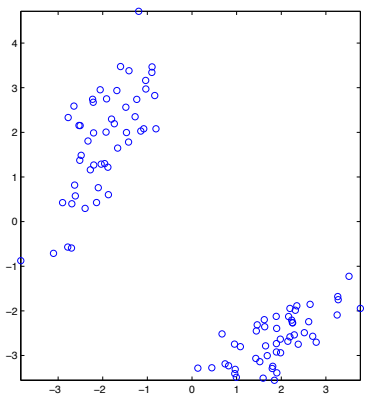


Figure: Two dimensional data which displays clusters. In this case a Gaussian mixture model $1/2\mathcal{N}(\mathbf{x}|\mathbf{m}_1, \mathbf{C}_1) + 1/2\mathcal{N}(\mathbf{x}|\mathbf{m}_2, \mathbf{C}_2)$ would fit the data well for suitable means $\mathbf{m}_1, \mathbf{m}_2$ and covariances $\mathbf{C}_1, \mathbf{C}_2$.

Maximum Likelihood

For a set of data $\mathcal{X} = \left\{\mathbf{x}^1, \ldots, \mathbf{x}^N\right\}$ and under the usual i.i.d. assumption, the log likelihood is

$$\sum_{n=1}^{N} \log \sum_{i=1}^{H} p(i) \frac{1}{\sqrt{\det\left(2\pi \mathbf{S}_i\right)}} \exp\left\{-\frac{1}{2}\left(\mathbf{x}^n - \mathbf{m}_i\right)^{\mathsf{T}} \mathbf{S}_i^{-1}\left(\mathbf{x}^n - \mathbf{m}_i\right)\right\}$$

Parameter Constraints

The $\mathbf{S}_i$ must be symmetric positive definite matrices, in addition to $0 \leq p(i) \leq 1$, $\sum_i p(i) = 1$. Gradient based optimisation approaches are feasible under a parameterisation of the $\mathbf{S}_i$ (*e.g.* Cholesky decomposition) and $p(i)$ (*e.g.* softmax) that enforce the constraints. An alternative is the EM approach which in this case is particularly convenient since it automatically provides parameter updates that ensure these constraints.

Expectation Maximisation

The energy term is

$$\sum_{n=1}^{N} \langle \log p(\mathbf{x}^n, i) \rangle_{p^{old}(i|\mathbf{x}^n)} = \sum_{n=1}^{N} \langle \log \left[p(\mathbf{x}^n|i) p(i) \right] \rangle_{p^{old}(i|\mathbf{x}^n)}$$

Plugging in the definition of the Gaussian components, we have

$$\sum_{n=1}^{N} \sum_{i=1}^{H} p^{old}(i|\mathbf{x}^n) \left\{ -\frac{1}{2} \left(\mathbf{x}^n - \mathbf{m}_i \right)^{\mathsf{T}} \mathbf{S}_i^{-1} \left(\mathbf{x}^n - \mathbf{m}_i \right) \right.$$
$$\left. -\frac{1}{2} \log \det \left(2\pi \mathbf{S}_i \right) + \log p(i) \right\}$$

M-step : optimal mixture coefficients

If no constraint is placed on the weights, the update is simply given by

$$p(i) = \frac{1}{N} \sum_{n=1}^{N} p^{old}(i|\mathbf{x}^n)$$

M-step : optimal $\mathbf{m}_i$

Maximising the energy with respect to $\mathbf{m}_i$ is equivalent to minimising

$$\sum_{n=1}^{N}\sum_{i=1}^{H} p^{old}(i|\mathbf{x}^n)\,(\mathbf{x}^n-\mathbf{m}_i)^{\mathsf{T}}\,\mathbf{S}_i^{-1}\,(\mathbf{x}^n-\mathbf{m}_i)$$

Differentiating with respect to $\mathbf{m}_i$ and equating to zero, optimally

$$\mathbf{m}_i=\frac{\sum_{n=1}^{N} p^{old}(i|\mathbf{x}^n)\mathbf{x}^n}{\sum_{n=1}^{N} p^{old}(i|\mathbf{x}^n)}$$

By defining the membership distribution

$$p^{old}(n|i)\equiv\frac{p^{old}(i|\mathbf{x}^n)}{\sum_{n=1}^{N} p^{old}(i|\mathbf{x}^n)}$$

which quantifies the membership of datapoints to cluster i,

$$\mathbf{m}_i=\sum_{n=1}^{N} p^{old}(n|i)\mathbf{x}^n$$

M-step : optimal $\mathbf{S}_i$

Optimising the energy with respect to $\mathbf{S}_i$ is equivalent to minimising

$$\sum_{n=1}^{N} \left\langle (\Delta_i^n)^\mathsf{T} \mathbf{S}_i^{-1} \Delta_i^n - \log\det\left(\mathbf{S}_i^{-1}\right) \right\rangle_{p^{old}(i|\mathbf{x}^n)}$$

where $\Delta_i^n \equiv \mathbf{x}^n - \mathbf{m}_i$. We isolate the dependency on $\mathbf{S}_i$ to give

$$\mathrm{trace}\left(\mathbf{S}_i^{-1} \sum_{n=1}^{N} p^{old}(i|\mathbf{x}^n) \Delta_i^n \left(\Delta_i^n\right)^\mathsf{T} \right) - \log\det\left(\mathbf{S}_i^{-1}\right) \sum_{n=1}^{N} p^{old}(i|\mathbf{x}^n)$$

Differentiating with respect to $\mathbf{S}_i^{-1}$ and equating to zero, we obtain

$$\sum_{n=1}^{N} p^{old}(i|\mathbf{x}^n) \Delta_i^n \left(\Delta_i^n\right)^\mathsf{T} - \mathbf{S}_i \sum_{n=1}^{N} p^{old}(i|\mathbf{x}^n) = \mathbf{0}$$

Using the membership $p^{old}(n|i)$, the optimal $\mathbf{S}_i$ is given by

$$\mathbf{S}_i = \sum_{n=1}^{N} p^{old}(n|i) \left(\mathbf{x}^n - \mathbf{m}_i\right) \left(\mathbf{x}^n - \mathbf{m}_i\right)^\mathsf{T}$$

This ensures that $\mathbf{S}_i$ is symmetric positive semi-definite.

M-step diagonal covariance

- For high dimensional data, learning the covariance matrix is problematic, and one often restricts the covariance to be diagonal.
- It is straightforward to show that in the diagonal case, the M-step corresponds to taking the diagonal entries of the general covariance update.

E-step

$$p(i|\mathbf{x}^n) \propto p(\mathbf{x}^n|i)p(i)$$

Explicitly, this is given by the responsibility

$$p(i|\mathbf{x}^n) = \frac{p(i)\exp\left\{-\frac{1}{2}(\mathbf{x}^n - \mathbf{m}_i)^\mathsf{T}\mathbf{S}_i^{-1}(\mathbf{x}^n - \mathbf{m}_i)\right\}}{\sum_{i'} p(i')\exp\left\{-\frac{1}{2}(\mathbf{x}^n - \mathbf{m}_{i'})^\mathsf{T}\mathbf{S}_{i'}^{-1}(\mathbf{x}^n - \mathbf{m}_{i'})\right\}}$$

Infinite troubles

Consider placing a component $p(\mathbf{x}|\mathbf{m}_i, \mathbf{S}_i)$ with mean $\mathbf{m}_i$ set to one of the datapoints $\mathbf{m}_i = \mathbf{x}^n$. The contribution from that Gaussian will be

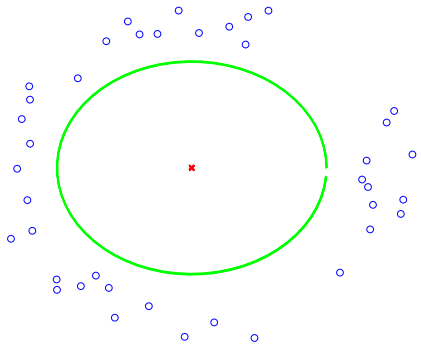
$$p(\mathbf{x}^n|\mathbf{m}_i, \mathbf{S}_i) = \frac{1}{\sqrt{\det(2\pi\mathbf{S}_i)}} e^{-\frac{1}{2}(\mathbf{x}^n - \mathbf{x}^n)^\mathsf{T}\mathbf{S}_i^{-1}(\mathbf{x}^n - \mathbf{x}^n)} = \frac{1}{\sqrt{\det(2\pi\mathbf{S}_i)}}$$

As the covariance goes to zero, this probability density becomes infinite. This means that one can obtain a Maximum Likelihood solution by placing zero-width Gaussians on a selection of the datapoints, resulting in an infinite likelihood.

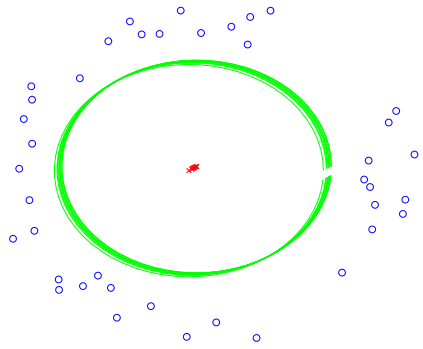
Remedy

Include an additional constraint on the width of the Gaussians, ensuring that they cannot become too small.

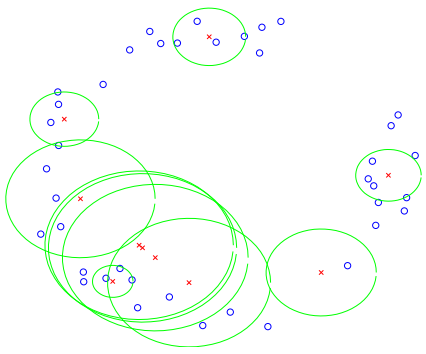
Symmetry Breaking



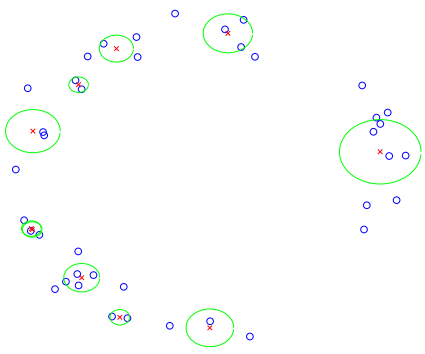
(a) 1 iteration



(b) 50 iterations



(c) 125 iterations



(d) 150 iterations

Figure: Training a mixture of 10 isotropic Gaussians **(a)**: If we start with large variances for the Gaussians, even after one iteration, the Gaussians are centred close to the mean of the data. **(b)**: The Gaussians begin to separate **(c)**: One by one, the Gaussians move towards appropriate parts of the data **(d)**: The final converged solution. The Gaussians are constrained to have variances greater than a set amount.

The Parzen estimator

Place a 'bump of mass', $\rho(\mathbf{x}|\mathbf{x}^n)$, on each datapoint:

$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\rho(\mathbf{x}|\mathbf{x}^n)$$

A popular choice is (for a D dimensional $\mathbf{x}$)

$$\rho(\mathbf{x}|\mathbf{x}^n) = \mathcal{N}\left(\mathbf{x}|\mathbf{x}^n, \sigma^2\mathbf{I}_D\right)$$

Training?

There is no training required for a Parzen estimator – only the positions of the N datapoints need storing.

Density Estimation versus compression

Whilst the Parzen technique is a reasonable and cheap way to form a density estimator, it does not enable us to form a simpler description of the data.

K-Means

Consider a mixture of K isotropic Gaussians in which each covariance is constrained to be equal to $\sigma^2\mathbf{I}$,

$$p(\mathbf{x}) = \sum_{i=1}^{K} p_i \mathcal{N}\left(\mathbf{x}|\mathbf{m}_i, \sigma^2\mathbf{I}\right)$$

As $\sigma^2 \to 0$, the membership distribution becomes deterministic

$$p^{old}(n|i) \propto \begin{cases} 1 & \text{if } \mathbf{m}_i \text{ is closest to } \mathbf{x}^n \\ 0 & \text{otherwise} \end{cases}$$

In this limit the EM update for the mean $\mathbf{m}_i$ is given by taking the average of the points closest to $\mathbf{m}_i$.

Vector Quantisation

K-means is often used as a simple form of data compression. Rather than sending the datapoint $\mathbf{x}^n$, one sends instead the index of the centre to which it is associated. This is called vector quantisation.

K-means

1: Initialise the centres $\mathbf{m}_i$, $i = 1, \ldots, K$.
2: **while** not converged **do**
3: For each centre i, find all the $\mathbf{x}^n$ for which i is the nearest (in Euclidean sense) centre.
4: Call this set of points $\mathcal{N}_i$. Let N_i be the number of datapoints in set $\mathcal{N}_i$.
5: Update the means

$$\mathbf{m}_i^{new} = \frac{1}{N_i} \sum_{n \in \mathcal{N}_i} \mathbf{x}^n$$

6: **end while**

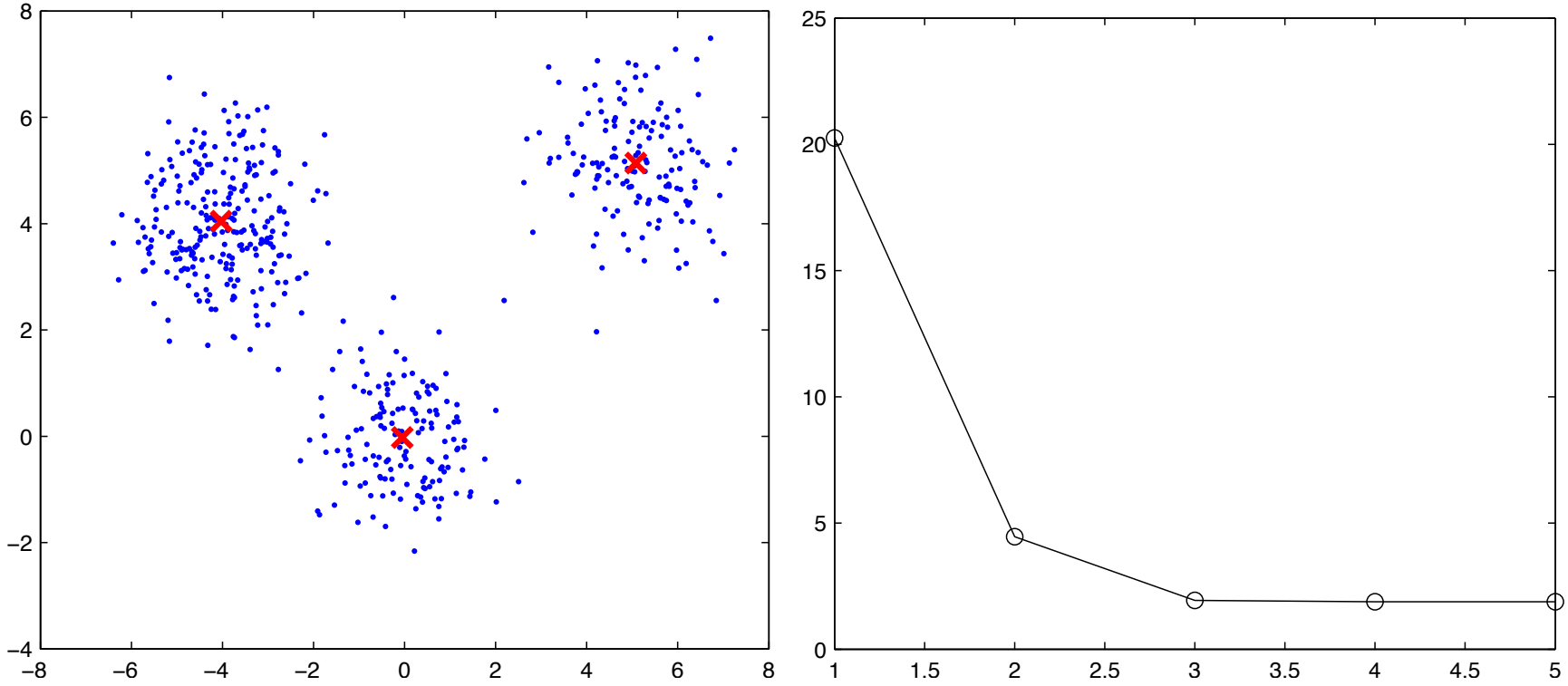


Figure: **(a)**: 550 datapoints clustered using K-means with 3 components. The means are given by the red crosses. **(b)**: Evolution of the mean square distance to nearest centre with iterations of the algorithm. The means were initialised to close to the overall mean of the data.