Expectation-Maximization Algorithm.

Petr Pošík
petr.posik@fel.cvut.cz

Czech Technical University in Prague
Faculty of Electrical Engineering
Dept. of Cybernetics

Likelihood maximization

Let's have a random variable X with probability distribution $p_X(x|\theta)$.

- The notation emphasizes that the distribution is parameterized by $\theta \in \Theta$, i.e. the distribution comes from certain parametric family. Θ is the space of possible parameter values.

Learning task: assume the parameters θ are unknown, but we have an i.i.d. training dataset $T = \{x_1, \ldots, x_n\}$ which can be used to estimate the unknown parameters.

- The probability of observing dataset T given some parameter values θ is

$$P(T|\theta) = \prod_{j=1}^{n} p_X(x_j|\theta) \overset{\text{def}}{=} L(\theta; T).$$

- This probability can be interpretted as a degree with which the model parameters θ conform to the data T. It is thus called the **likelihood of parameters θ** w.r.t. data T.
- The optimal θ^* is obtained by maximizing the likelihood

$$\theta^* = \arg\max_{\theta\in\Theta} L(\theta; T) = \arg\max_{\theta\in\Theta} \prod_{j=1}^{n} p_X(x_j|\theta)$$

- Since $\arg\max_x f(x) = \arg\max_x \log f(x)$, we often maximize the **log-likelihood** $l(\theta; T) = \log L(\theta; T)$

$$\theta^* = \arg\max_{\theta\in\Theta} l(\theta; T) = \arg\max_{\theta\in\Theta} \log \prod_{j=1}^{n} p_X(x_j|\theta) = \arg\max_{\theta\in\Theta} \sum_{j=1}^{n} \log p_X(x_j|\theta),$$

  which is often easier than maximization of L.

Question

Although we defined the likelihood for discrete distributions only, an analogical definition can be made for continuous distributions.

For normal distribution $N(x|\theta)$, the parameters are $\theta = (\mu, \sigma^2)$.
What are the maximum likelihood estimates $\theta^* = (\mu^*, v^*)$ of the parameters of normal distribution, given the training data $T = (x_1, \ldots, x_n)$?

A $\quad \mu^* = \frac{1}{n-1}\sum_{i=1}^{n} x_i \quad$ and $\quad v^* = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu^*)^2$

B $\quad \mu^* = \frac{1}{n-1}\sum_{i=1}^{n} x_i \quad$ and $\quad v^* = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu^*)^2$

C $\quad \mu^* = \frac{1}{n}\sum_{i=1}^{n} x_i \quad$ and $\quad v^* = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu^*)^2$

D $\quad \mu^* = \frac{1}{n}\sum_{i=1}^{n} x_i \quad$ and $\quad v^* = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu^*)^2$

Incomplete data

Assume we cannot observe the objects completely:

- r.v. X describes the observable part, r.v. K describes the unobservable, hidden part.
- We assume there is an underlying distribution $p_{XK}(x,k|\theta)$ of objects (x,k).

Learning task: we want to estimate the model parameters θ, but the training set contains i.i.d. samples for the observable part only, i.e. $T_X = \{x_1,\ldots,x_n\}$. (Still, there also exists a hidden, unobservable dataset $T_K = \{k_1,\ldots,k_n\}$.)

- If we had complete data (T_X, T_K), we could directly optimize $l(\theta; T_X, T_K) = \log P(T_X, T_K|\theta)$. But we do not have access to T_K.
- If we would like to maximize

$$l(\theta; T_X) = \log P(T_X|\theta) = \log \sum_{T_K} P(T_X, T_K|\theta),$$

  the summation inside $\log()$ results in complicated expressions, or we would have to use numerical methods.
- Our state of knowledge about T_K is given by $P(T_K|T_X,\theta)$.
- The complete-data likelihood $L(\theta; T_X, T_K) = P(T_X, T_K|\theta)$ is a random variable since T_K is unknown, random, but governed by the underlying distribution.
- Instead of optimizing it directly, consider its expected value under the posterior distribution $P(T_K|T_X,\theta)$ over latent variables (E-step), and then maximize this expectation (M-step).

Expectation-Maximization algorithm

EM algorithm:

- A general method of finding MLE of prob. dist. parameters from a given dataset when data is incomplete (hidden variables, or missing values).
- Hidden variables: mixture models, Hidden Markov models, ...
- It is a family of algorithms, or a recipe to derive a ML estimation algorithm for various kinds of probabilistic models.

1. Pretend that you know θ. (Use some initial guess $\theta^{(0)}$.) Set iteration counter $i = 1$.
2. **E-step:** Use the current parameter values $\theta^{(i-1)}$ to find the posterior distribution of the latent variables $P(T_K|T_X,\theta^{(i-1)})$. Use this posterior distribution to find the expectation of the complete-data log-likelihood evaluated for some general parameter values θ:

$$Q(\theta,\theta^{(i-1)}) = \sum_{T_K} P(T_K|T_X,\theta^{(i-1)}) \log P(T_X, T_K|\theta).$$

3. **M-step:** maximize the expectation, i.e. compute an updated estimate of θ as

$$\theta^{(i)} = \arg\max_{\theta\in\Theta} Q(\theta,\theta^{(i-1)}).$$

4. Check for convergence: finish, or advance the iteration counter $i \Longleftarrow i+1$, and repeat from 2.

EM algorithm features

Pros:

- Among the possible optimization methods, EM exploits the structure of the model.
- For $p_{X|K}$ from exponential family:
  - M-step can be done analytically and there is a unique optimizer.
  - The expected value in the E-step can be expressed as a function of θ without solving it explicitly for each θ.
- $P(T_X|\theta^{(i)}) \geq P(T_X|\theta^{(i-1)})$, i.e. the process finds a local optimum.
- Works well in practice.

Cons:

- Not guaranteed to get globally optimal estimate.
- MLE can overfit; use MAP instead (EM can be used as well).
- Convergence may be slow.

K-means

K-means algorithm

Clustering is one of the tasks of *unsupervised learning*.

K-means algorithm for clustering [Mac67]:

- K is the apriori given number of clusters.
- Algorithm:
  1. Choose K centroids μ_k (in almost any way, but every cluster should have at least one example.)
  2. For all x, assign x to its closest μ_k.
  3. Compute the new position of centroids μ_k based on all examples $x_i, i \in I_k$, in cluster k.
  4. If the positions of centroids changed, repeat from 2.

Algorithm features:

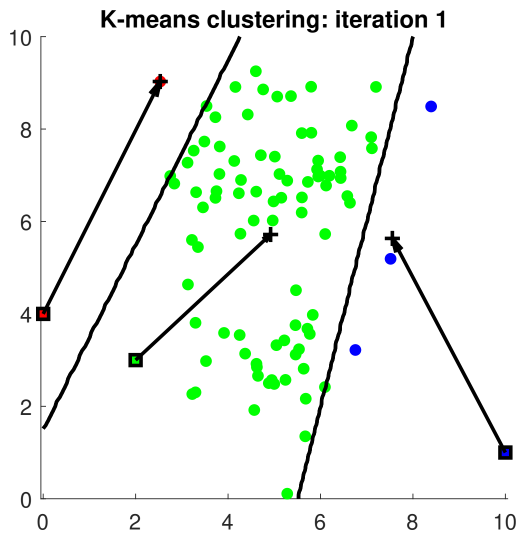
- Algorithm minimizes the function (intracluster variance):

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n_j} |x_{i,j} - c_j|^2 \tag{1}$$

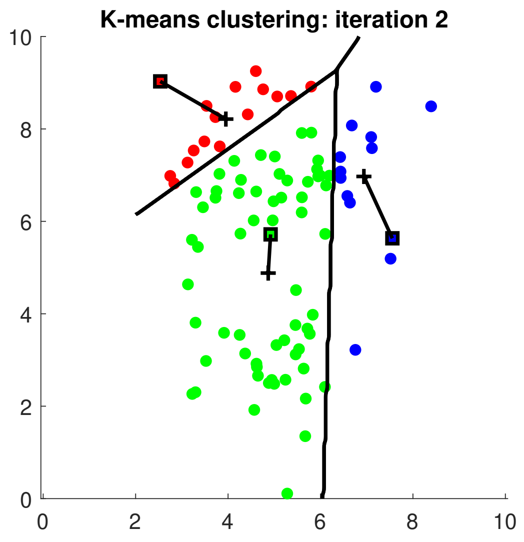
- Algorithm is fast, but each time it can converge to a different local optimum of J.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

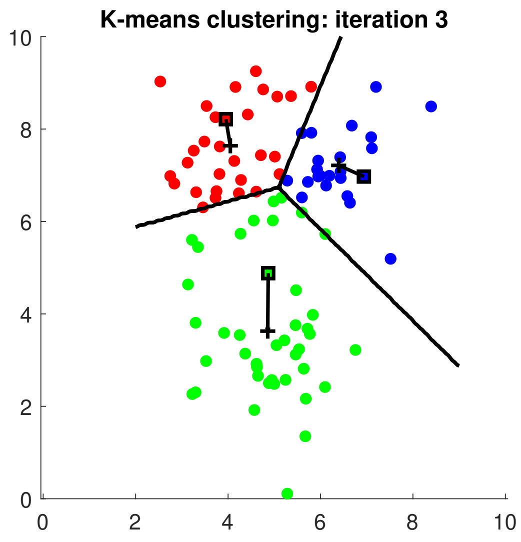
Illustration



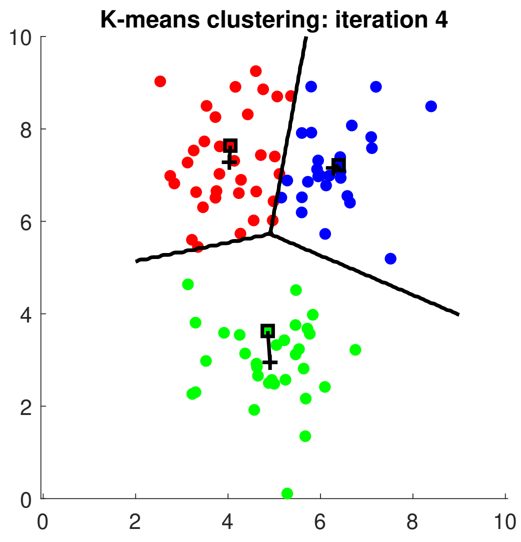
Illustration



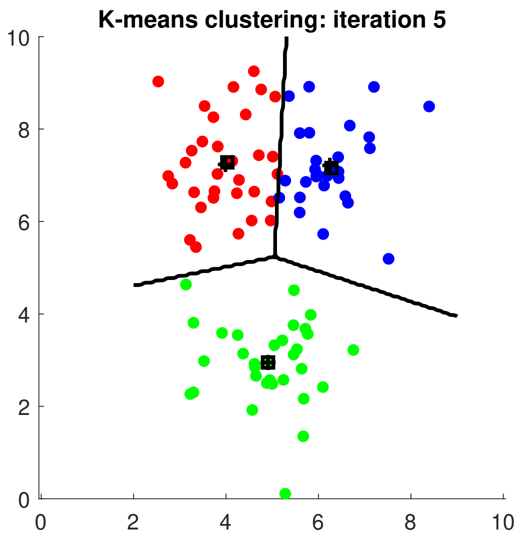
Illustration



Illustration



Illustration



K-means: EM view

Assume:

- An object can be in one of the $|K|$ states with equal probabilities.
- All $p_{X|K}(x|k)$ are isotropic Gaussians: $p_{X|K}(x|k) = \mathcal{N}(x|\mu_k, \sigma^2\mathbf{I})$.

Recognition (Part of E-step):

- The task is to decide the state k for each $\boldsymbol{x}$, assuming all μ_k are known.
- The Bayesian strategy (minimizes the probability of error) chooses the cluster with the center closest to observation $\boldsymbol{x}$:

$$q^*(\boldsymbol{x}) = \arg\min_{k\in K}(\boldsymbol{x} - \mu_k)^2$$

- If $\mu_k, k \in K$, are not known, it is a parametrized strategy $q_\theta(\boldsymbol{x})$, where $\theta = (\mu_k)_{k=1}^K$.
- Deciding state k for each $\boldsymbol{x}$ assuming known μ_k is actually the computation of a degenerate probability distribution $P(T_K|T_X, \theta^{(i-1)})$, i.e. the first part of E-step.

Learning (The rest of E-step and M-step):

- Find the maximum-likelihood estimates of μ_k based on known $(x_1, k_1), \dots, (x_l, k_l)$:

$$\mu_k^* = \frac{1}{|I_k|}\sum_{i\in I_k} \boldsymbol{x}_i,$$

  where I_k is a set of indices of training examples (currently) belonging to state k.
- This completes the E-step and implements the M-step.

General mixture distributions

Assume the data are samples from a distribution factorized as

$$p_{XK}(x,k) = p_K(k)p_{X|K}(x|k), \text{ i.e.}$$
$$p_X(x) = \sum_{k \in K} p_K(k)p_{X|K}(x|k)$$

and that the distribution is known (except the distribution parameters).

Recognition (Part of E-step):

- Let's define the result of recognition not as a single decision for some state k (as done in K-means), but rather as
- a set of posterior probabilities (sometimes called *responsibilities*) for each k given x_i

$$\gamma_k(x_i) = p_{K|X}(k|x_i, \theta^{(t)}) = \frac{p_{X|K}(x_i|k)p_K(k)}{\sum_{k \in K} p_{X|K}(x_i|k)p_K(k)}$$

  that an object was in state k when observation x_i was made.
- The $\gamma_k(x)$ functions can be viewed as discriminant functions.

General mixture distributions (cont.)

Learning (The rest of E-step and M-step):

- Given the training multiset $T = (x_i, k_i)_{i=1}^n$ (or the respective $\gamma_k(x_i)$ instead of k_i),
- assume $\gamma_k(x)$ is known, $p_K(k)$ are not known, and $p_{X|K}(x|k)$ are known except the parameter values θ_k, i.e. we shall write $p_{X|K}(x|k, \theta_k)$.
- Let the object *model* m be a "set" of all unknown parameters $m = (p_K(k), \theta_k)_{k \in K}$.
- The log-likelihood of model m if we assume k_i is known:

$$\log L(m) = \log \prod_{i=1}^n p_{XK}(x_i, k_i) = \sum_{i=1}^n \log p_K(k_i) + \sum_{i=1}^n \log p_{X|K}(x_i|k_i, \theta_{k_i})$$

- The expectation of log-likelihood of model m if we assume a distribution γ_k is known for all observations x_i:

$$\log L(m) = \sum_{i=1}^n \sum_{k \in K} \gamma_k(x_i) \log p_K(k) + \sum_{i=1}^n \sum_{k \in K} \gamma_k(x_i) \log p_{X|K}(x_i|k, \theta_k)$$

- We search for the optimal model using maximum likelihood:

$$m^* = (p_K^*(k), \theta_k^*) = \arg\max_m \log L(m)$$

- i.e. we compute

$$p_K^*(k) = \frac{1}{n}\sum_{i=1}^n \gamma_k(x_i) \text{ and solve } k \text{ independent tasks}$$

$$\theta_k^* = \arg\max_{\theta_k \in \Theta_k} \sum_{i=1}^n \gamma_k(x_i) \log p_{X|K}(x_i|k, \theta_k).$$

EM for mixture distribution

Unsupervised learning algorithm [?] for general mixture distributions:

1. Initialize the model parameters $m = ((p_K(k), \theta_k) \forall k)$.
2. Perform the **recognition** task, i.e. assuming m is known, compute

$$\gamma_k(x_i) = \hat{p}_{K|X}(k|x_i) = \frac{p_K(k) p_{X|K}(x_i|k, \theta_k)}{\sum_{j \in K} p_K(j) p_{X|K}(x_i|j, \theta_j)}.$$

3. Perform the **learning** task, i.e. assuming $\gamma_k(x_i)$ are known, update the ML estimates of the model parameters $p_K(k)$ and θ_k for all k:

$$p_K(k) = \frac{1}{n} \sum_{i=1}^{n} \gamma_k(x_i)$$

$$\theta_k = \arg\max_{\theta_k \in \Theta_k} \sum_{i=1}^{n} \gamma_k(x_i) \log p_{X|K}(x_i|k, \theta_k)$$

4. Iterate 2 and 3 until the model stabilizes.

Features:

- The algorithm does not specify how to update θ_k in step 3, it depends on the chosen form of $p_{X|K}$.
- The model created in iteration t is always at least as good as the model from iteration $t-1$, i.e. $L(m) = P(T|m)$ increases.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

Special Case: Gaussian Mixture Model

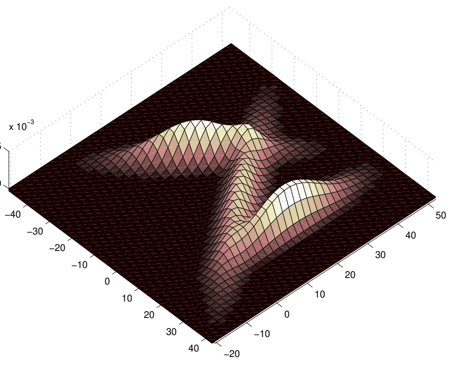
Each kth component is a Gaussian distribution:

$$\mathcal{N}(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_k|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)\}$$

Gaussian Mixture Model (GMM):

$$p(x) = \sum_{k=1}^{K} p_K(k) p_{X|K}(x|k, \theta_k) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

$$\text{assuming } \sum_{k=1}^{K} \alpha_k = 1 \text{ and } 0 \leq \alpha_k \leq 1$$



EM for GMM

1. Initialize the model parameters $m = ((p_K(k), \mu_k, \Sigma_k)\forall k)$.
2. Perform the **recognition** task as in the general case, i.e. assuming m is known, compute

$$\gamma_k(x_i) = \hat{p}_{K|X}(k|x_i) = \frac{p_K(k)p_{X|K}(x_i|k,\theta_k)}{\sum_{j\in K} p_K(j)p_{X|K}(x_i|j,\theta_j)} = \frac{\alpha_k \mathcal{N}(x_i|\mu_k,\Sigma_k)}{\sum_{j\in K}\alpha_j\mathcal{N}(x_i|\mu_j,\Sigma_j)}.$$

3. Perform the **learning** task, i.e. assuming $\gamma_k(x_i)$ are known, update the ML estimates of the model parameters α_k, μ_k and Σ_k for all k:

$$\alpha_k = p_K(k) = \frac{1}{n}\sum_{i=1}^{n}\gamma_k(x_i)$$

$$\mu_k = \frac{\sum_{i=1}^{n}\gamma_k(x_i)x_i}{\sum_{i=1}^{n}\gamma_k(x_i)}$$

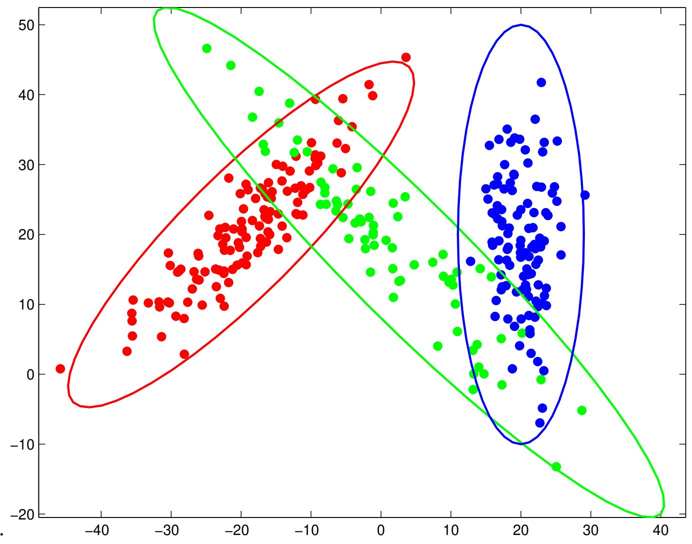
$$\Sigma_k = \frac{\sum_{i=1}^{n}\gamma_k(x_i)(x_i-\mu_k)(x_i-\mu_k)^T}{\sum_{i=1}^{n}\gamma_k(x_i)}$$

4. Iterate 2 and 3 until the model stabilizes.

Remarks:

- Each data point belongs to all components to a certain degree $\gamma_k(x_i)$.
- The eq. for μ_k is just a weighted average of x_is.
- The eq. for Σ_k is just a weighted covariance matrix.

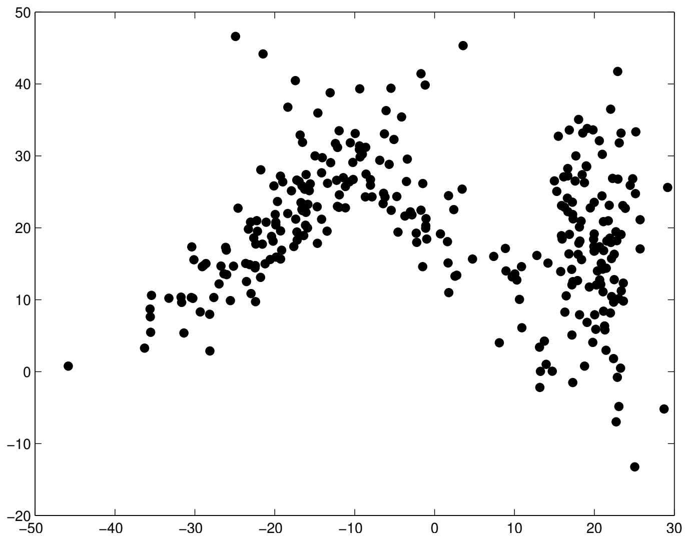
Example: Source data



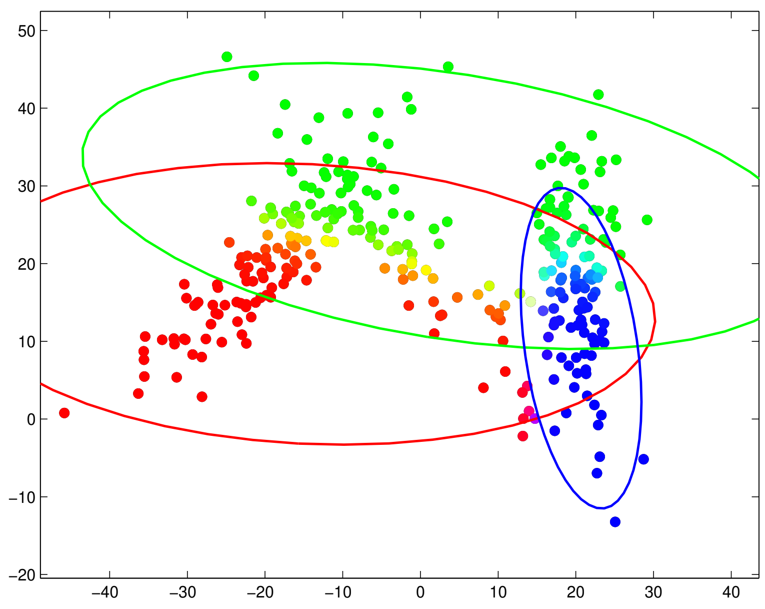
Source data generated from 3 Gaussians.

Example: Input to EM algorithm

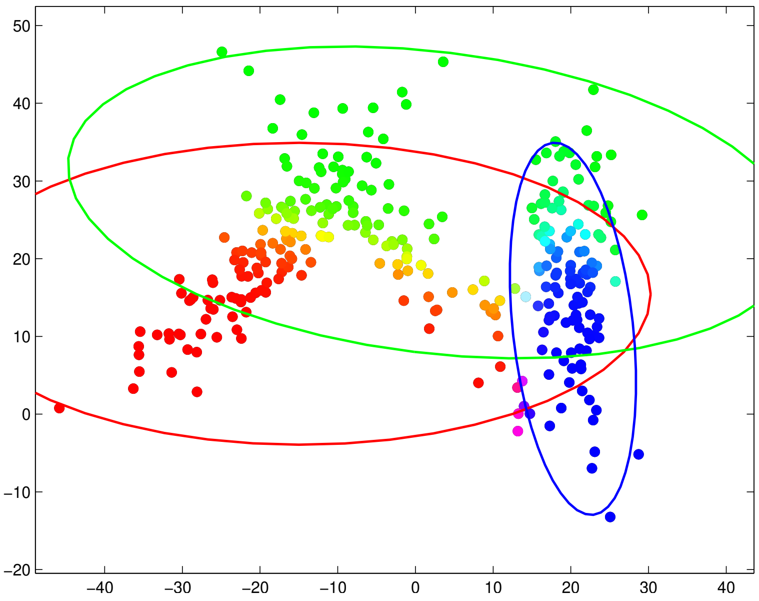
The data were given to the EM algorithm as an unlabeled dataset.



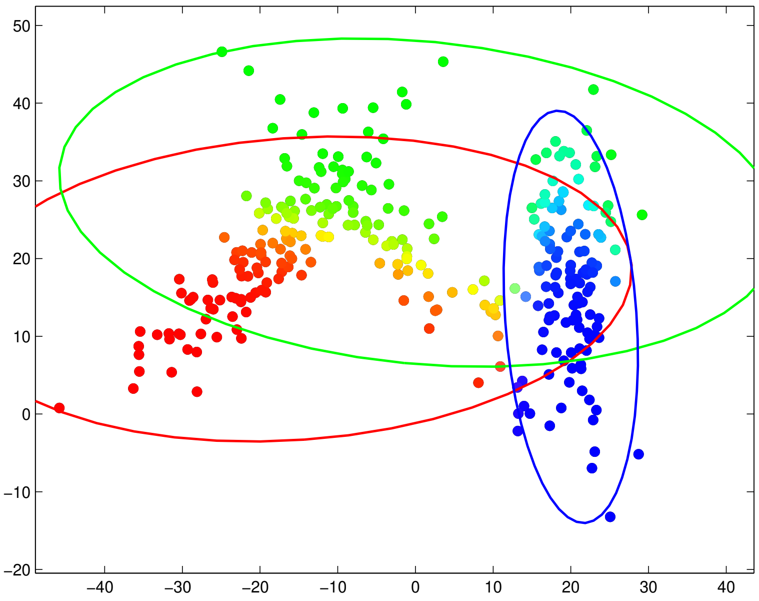
Example: EM Iterations



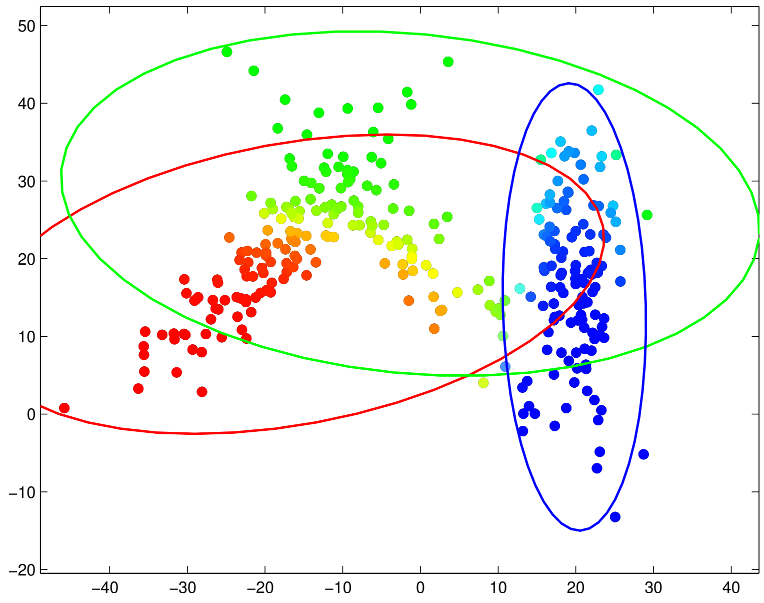
Example: EM Iterations



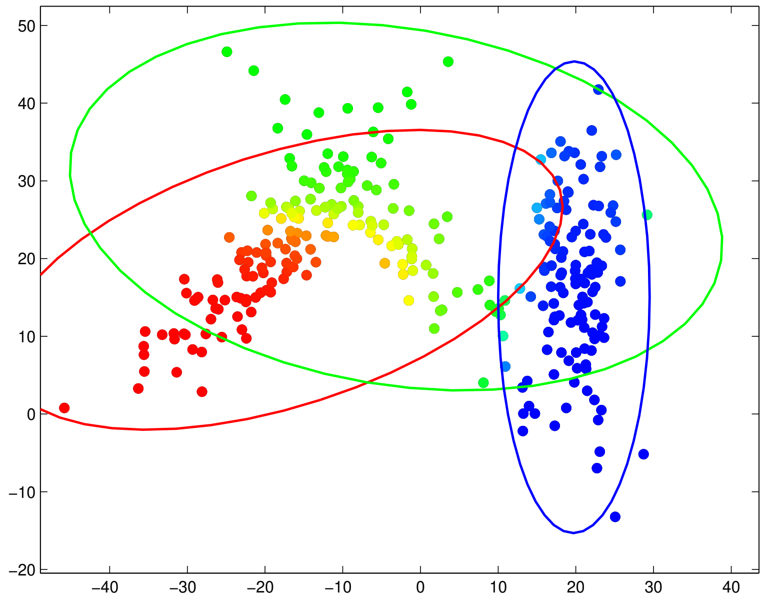
Example: EM Iterations



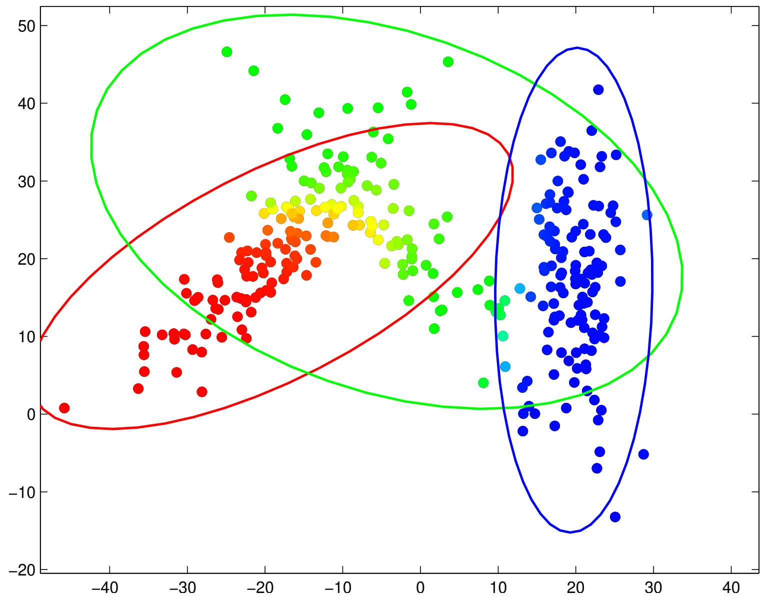
Example: EM Iterations



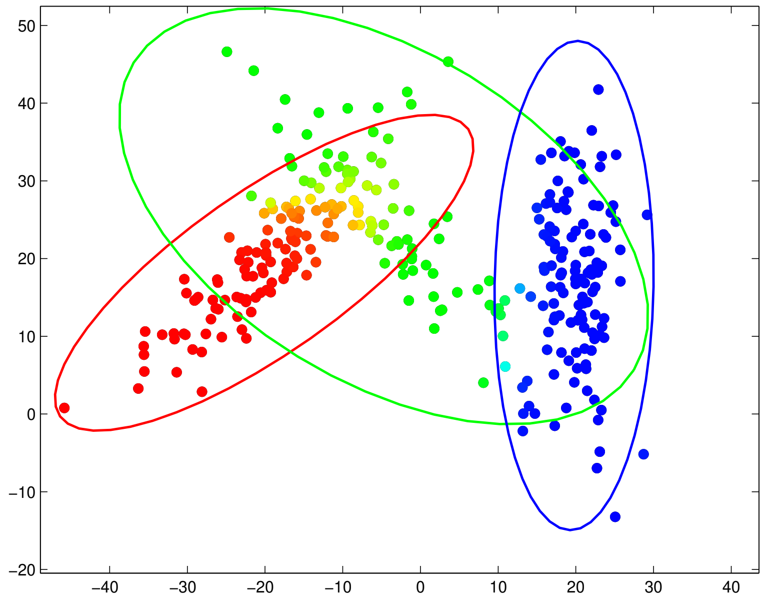
Example: EM Iterations



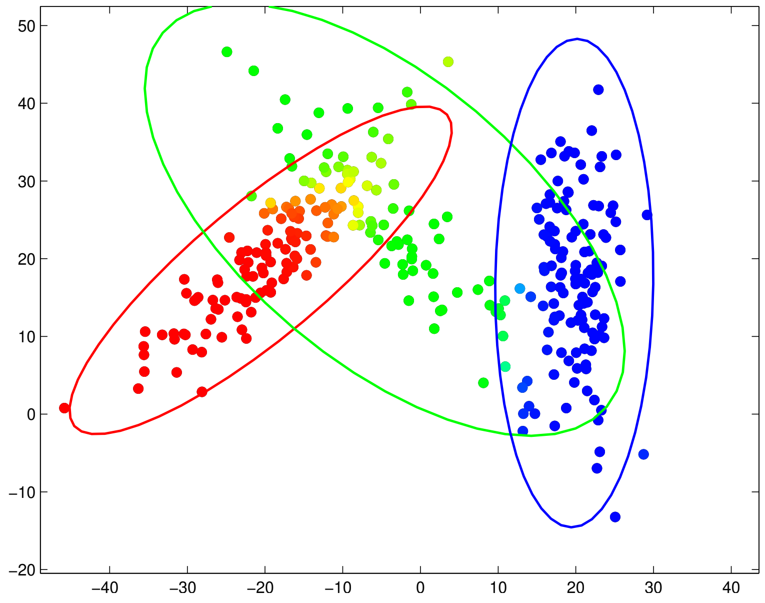
Example: EM Iterations



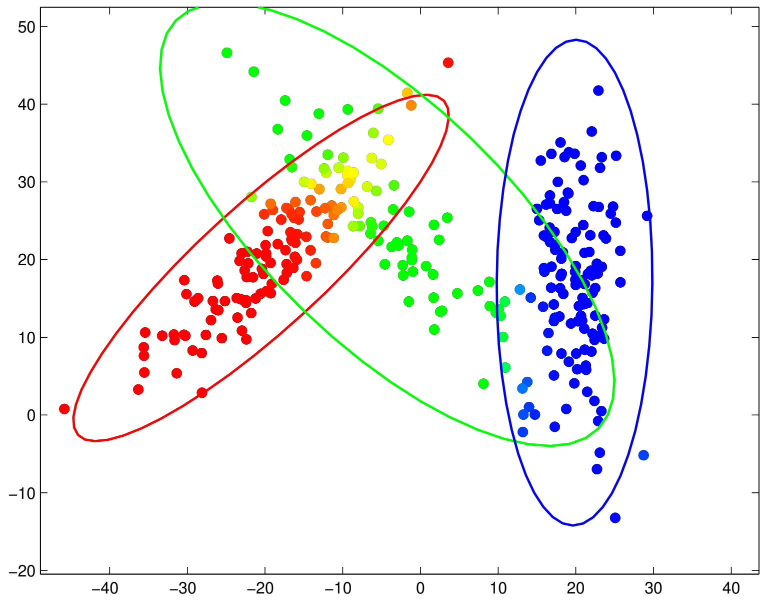
Example: EM Iterations



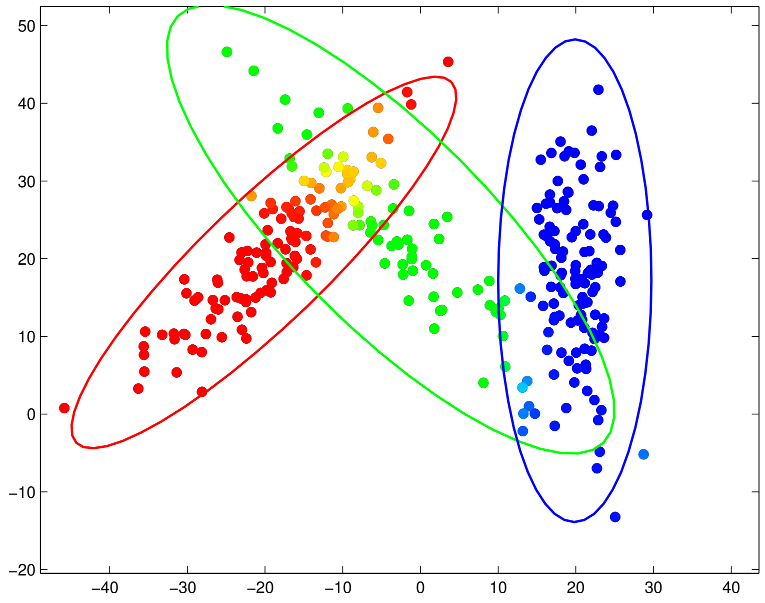
Example: EM Iterations



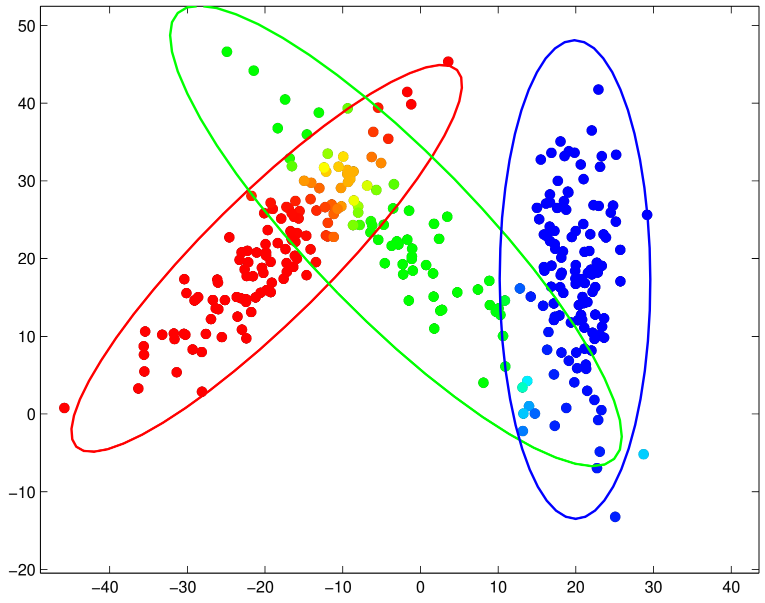
Example: EM Iterations



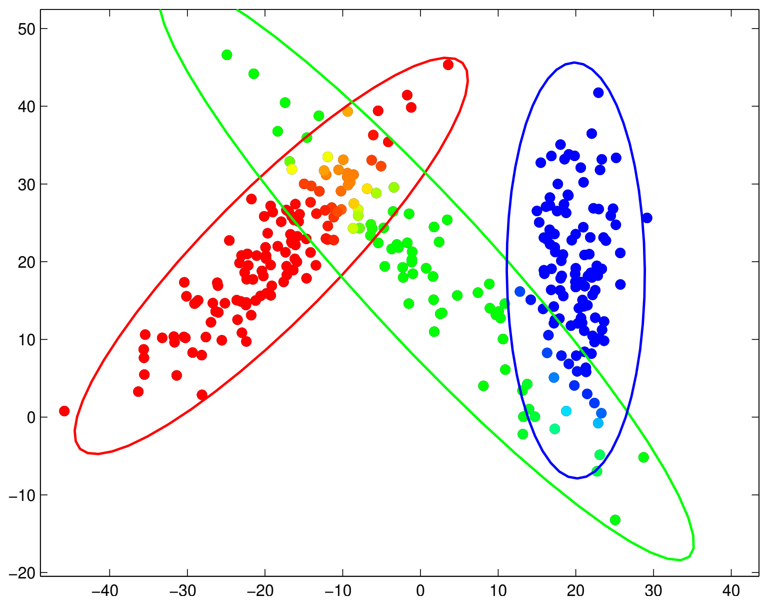
Example: EM Iterations



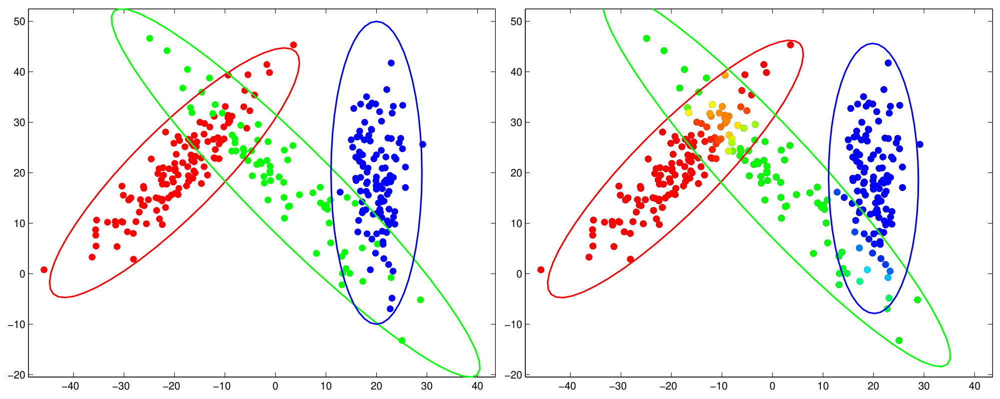
Example: EM Iterations



Example: EM Iterations



Example: Ground Truth and EM Estimate



The ground truth (left) and the EM estimate (right) are very close because

- we have enough data,
- we know the right number of components, and
- we were lucky that EM converged to the right local optimum of the likelihood function.

Baum-Welch Algorithm:

EM for HMM

Hidden Markov Model

1st order HMM is a generative probabilistic model formed by

- a sequence of hidden variables $X_0, \ldots, X_t$,
  the domain of all of them is the set of states $\{s_1, \ldots, s_N\}$.
- a sequence of observed variables $E_1, \ldots, E_t$,
  the domain of all of them is the set of observations $\{v_1, \ldots, v_M\}$.
- an initial distribution over hidden states $P(X_0)$,
- a transition model $P(X_t|X_{t-1})$, and
- an emission model $P(E_t|X_t)$.

Simulating HMM:

1. Generate an initial state x_0 according to $P(X_0)$. Set $t \leftarrow 1$.
2. Generate a new current state x_t according to $P(X_t|x_{t-1})$.
3. Generate an observation e_t according to $P(E_t|x_t)$.
4. Advance time $t \leftarrow t + 1$.
5. Finish, or repeat from step 2.

With HMM:

- efficient algorithms exist for solving inference tasks;
- but we have no idea (so far) how to learn HMM parameters from the observation sequence, because we do not have access to the hidden states.

Learning HMM from data

Is it possible to learn HMM from data?

- No known way to analytically solve for the model which maximizes the probability of observations.
- No optimal way of estimating the model parameters from the observation sequences.
- We can find model parameters such that the probability of observations is maximized $\longrightarrow$ **Baum-Welch algorithm** (a special case of EM).

Let's use a slightly different notation to emphasize the model parameters:

- $\pi = [\pi_i] = [P(X_1 = s_i)]$ ...vector of the initial probabilities of states
- $A = [a_{i,j}] = [P(X_t = s_j|X_{t-1} = s_i)]$ ...the matrix of transition probabilities to next state given the current state
- $B = [b_{i,k}] = [P(E_t = v_k|X_t = s_i)]$ ...the matrix of observation probabilities given the current state
- The whole set of **HMM parameters** is then $\theta = (\pi, A, B)$

The algorithm (presented on the next slides)

- computes the expected numbers of being in a state or taking a transition given the observations and *the current model parameters* $\theta = (\pi, A, B)$, and then
- computes the new estimate of model parameters $\theta' = (\pi', A', B')$,
- such that $P(e_1^t|\theta') \geq P(e_1^t|\theta)$.

Sufficient statistics

Let's define

- the probability of transition from state s_i at time t to state s_j at time $t+1$, given the model and the observation sequence e_1^t:

$$\xi_t(i,j) = P(X_t = s_i, X_{t+1} = s_j | e_1^t, \theta) = \frac{\alpha_t(s_i) a_{ij} b_{jk} \beta_{t+1}(s_j)}{P(e_1^t|\theta)} =$$
$$= \frac{\alpha_t(s_i) a_{ij} b_{jk} \beta_{t+1}(s_j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(s_i) a_{ij} b_{jk} \beta_{t+1}(s_j)},$$

  where α_t and β_t are the forward and backward messages computed by the forward-backward algorithm, and
- the probability of being in state s_i at time t, given the model and the observation sequence:

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i,j).$$

Then we can interpret

- $\sum_{k=1}^{T-1} \gamma_k(i)$ as *the expected number of transitions from state s_i*, and
- $\sum_{k=1}^{T-1} \xi_k(i,j)$ as *the expected number of transitions from s_i to s_j.*

Baum-Welch algorithm

The re-estimation formulas are

$$\pi_i' = \text{expected frequency of being in state } s_i \text{ at time } (t=1) =$$
$$= \gamma_1(i)$$

$$a_{ij}' = \frac{\text{expected number of transitions from } s_i \text{ to } s_j}{\text{expected number of transitions from } s_i} =$$
$$= \frac{\sum_{k=1}^{T-1} \xi_k(i,j)}{\sum_{k=1}^{T-1} \gamma_k(i)}$$

$$b_{jk}' = \frac{\text{expected number of times being in state } s_j \text{ and observing } v_k}{\text{expected number of times being in state } s_j} =$$
$$= \frac{\sum_{t=1}^T I(e_t = v_k) \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

As with other EM variants, with the old model parameters $\theta = (\pi, A, B)$ and new, re-estimated parameters $\theta' = (\pi', A', B')$, the new model is at least as likely as the old one:

$$P(e_1^t|\theta') \geq P(e_1^t|\theta)$$

The above equations are used iteratively with θ' taking place of θ.

Summary

Competencies

After this lecture, a student shall be able to . . .

1. define and explain the task of maximum likelihood estimation;
2. explain why we can maximize log-likelihood instead of likelihood, describe the advantages;
3. describe the issues we face when trying to maximize the likelihood in case of incomplete data;
4. explain the general high-level principle of Expectation-Maximization algorithm;
5. describe the pros and cons of the EM algorithm, especially what happens with the likelihood in one EM iteration;
6. describe the EM algorithm for mixture distributions, including the notion of responsibilities;
7. explain the Baum-Welch algorithm, i.e. the application of EM to HMM; what parameters are learned and how (conceptually).