Gaussian Mixture Models

Machine Learning 1 — Lecture 13

19th June 2023

Robert Peharz
Institute of Theoretical Computer Science
Graz University of Technology

K-Means Problem

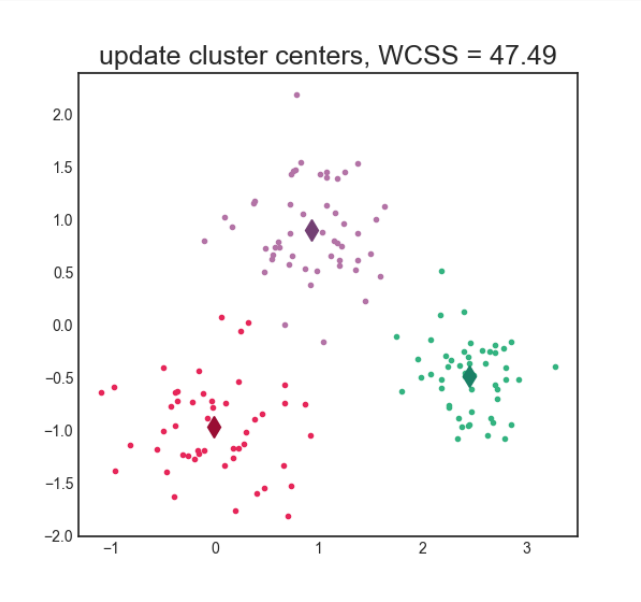
$$\min_{\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^K,\boldsymbol{Z}} \quad \sum_{i=1}^{N}\sum_{k=1}^{K} z_{i,k} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(k)} \right\|^2$$

$$\text{s.t.} \quad z_{i,k} \in \{0,1\}$$

$$\sum_{k} z_{i,k} = 1 \quad i = 1,\ldots,N$$

K-Means Algorithm

- assign data points to cluster centers
- set cluster centers as means of assigned data points
- repeat



Today, we discuss **Gaussian mixture models (GMMs)** which can be seen as a probabilistic version of k-means. The k-means algorithm corresponds to the **expectation-maximization (EM)** algorithm for learning GMMs.

Main purpose of GMMs is **density estimation**, i.e., approximating the true data generating distribution p_{true}.

Assume N samples $\mathcal{D} = \left\{ \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)} \right\}$ drawn i.i.d. (**independently and identically distributed**) from a true distribution with density $p_{true}(\boldsymbol{x})$. In **density estimation**, we want to learn/estimate a **model distribution** $p_{model}(\boldsymbol{x})$ such that

$$p_{model} \approx p_{true}$$

In the following, we will denote p_{model} simply as p.

Applications of Density Estimation

Say we have found a model with $p_{model} \approx p_{true}$. Generally, knowing p_{true} allows us—in principle—to answer any question we might have about the data.

The model can be used for

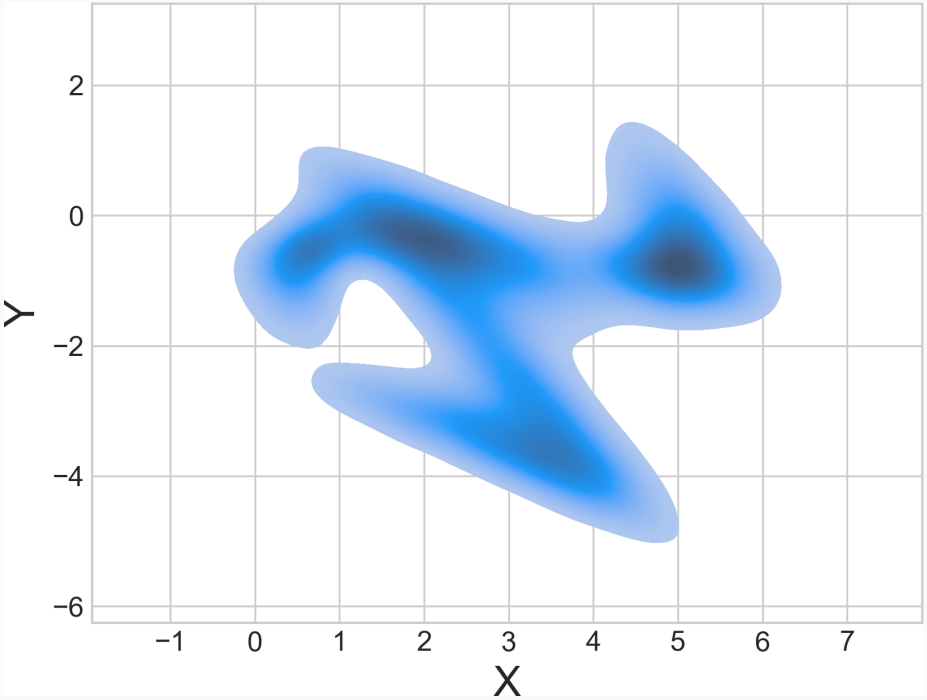
- **outlier detection** (monitor whether $p_{model}(\boldsymbol{x})$ is low)
- **generative modeling**: sample "new data" ($\boldsymbol{x} \sim p_{model}$)
- prediction via **conditional distributions**

$$p_{model}(\boldsymbol{x}_q \mid \boldsymbol{x}_e) = \frac{p_{model}(\boldsymbol{x}_q, \boldsymbol{x}_e)}{p_{model}(\boldsymbol{x}_e)}$$

where $\boldsymbol{x}_e$ is evidence for some variables $\boldsymbol{X}_e$ and $\boldsymbol{X}_q$ are query variables.

Regression

Say you want to predict Y from X, and you would know the true distribution $p_{true}(y, x)$ producing i.i.d. samples of X, Y.

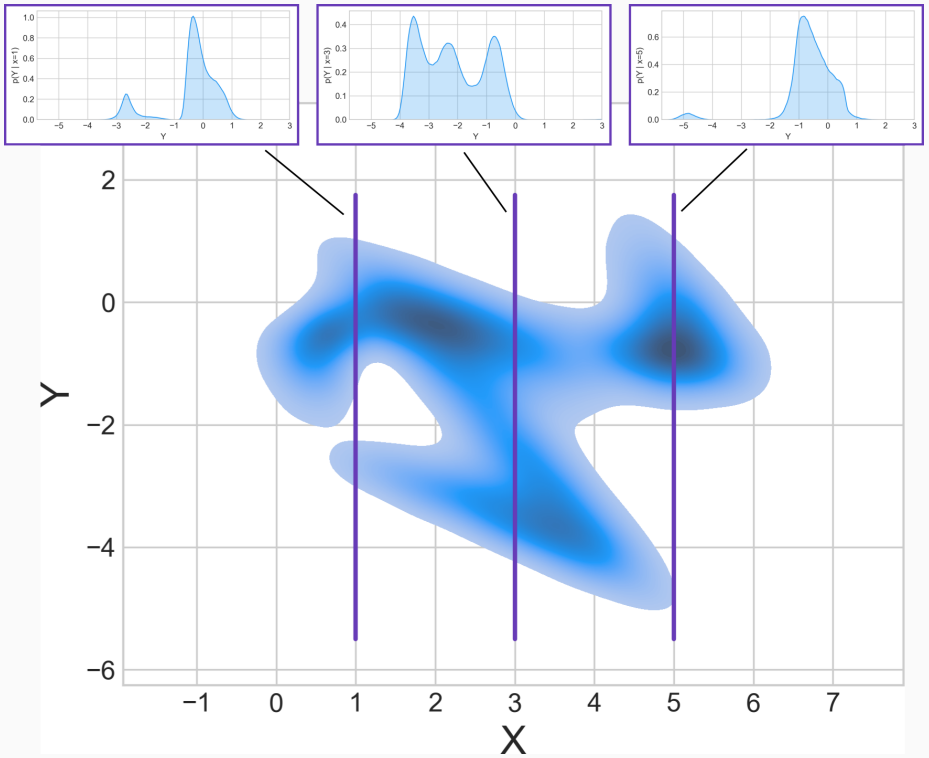


In practice, we almost never know p_{true}, so this is a very strong assumption.

You might compute the **conditional distribution**

$$p_{true}(y \mid x) = \frac{p_{true}(y, x)}{\underbrace{p_{true}(x)}_{=\int p_{true}(x,y)\,\mathrm{d}y}}$$

for all values of x.

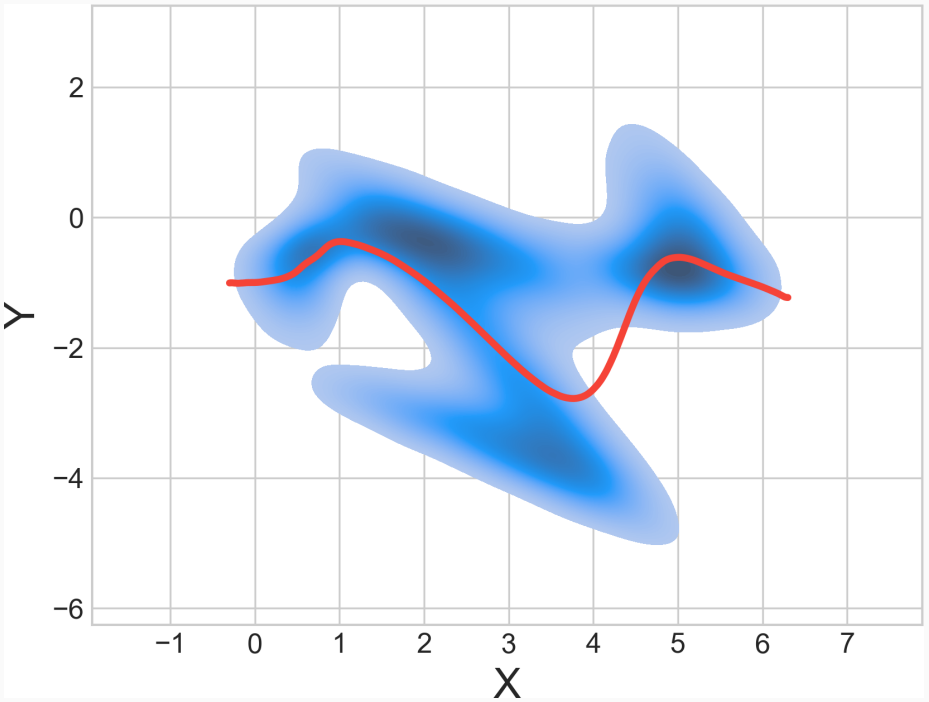


Then you can construct the **true regression function**

$$f(x) := \mathbb{E}_{p_{true}(y \mid x)}[Y \mid x]$$

which is just the **expectation** of Y given x

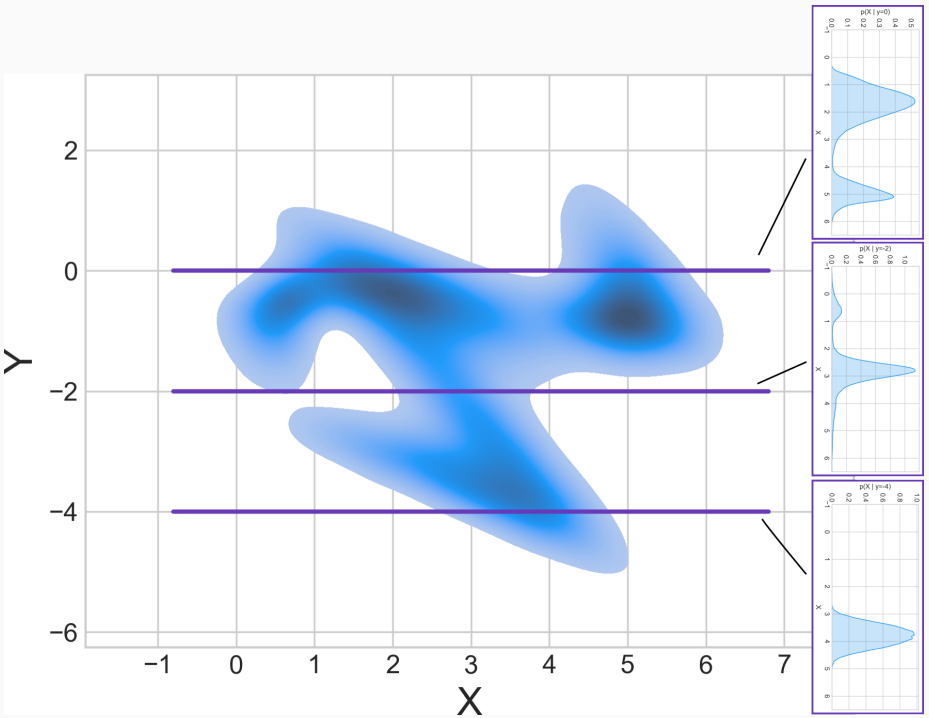
No other function has lower **true squared loss**.



Of course, when we now replace p_{true} with p_{model}, this is not true. In practice, directly learning $f(x)$ will deliver better results.

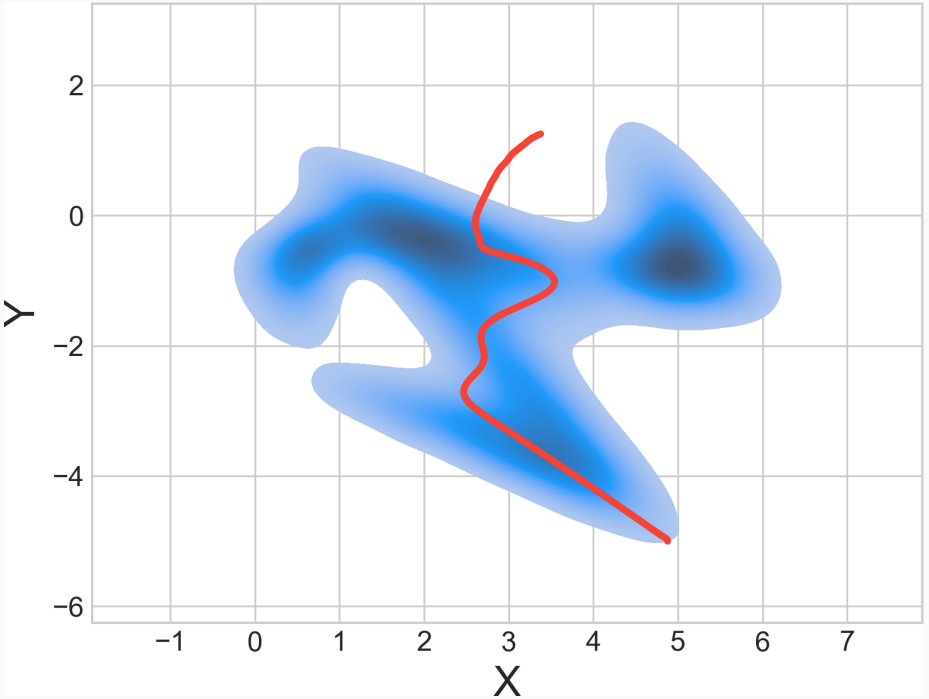
Of course, that Y is output and X is input was arbitrary, so the whole "trick" works also in the other direction:

$$p_{true}(x \mid y) = \frac{p_{true}(y, x)}{\underbrace{p_{true}(y)}_{= \int p_{true}(x,y)\,\mathrm{d}x}}$$



Of course, that Y is output and X is input was arbitrary, so the whole "trick" works also in the other direction:

$$f(y) := \mathbb{E}_{p_{true}(x \mid y)}[X \mid y]$$



Maximum Likelihood Principle

Setting

Say we have a dataset $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\}$ drawn i.i.d. from some unknown distribution $p_{true}(\boldsymbol{x})$.

Further, assume we have a **parametric model** $p(\boldsymbol{x}; \boldsymbol{\theta})$, i.e., a **family of distributions** or a **model class** described by a parameter vector $\boldsymbol{\theta}$, e.g., Gaussian, Poisson, Bernoulli, etc.

How do we fit $p(\boldsymbol{x}; \boldsymbol{\theta})$ to $p_{true}(\boldsymbol{x})$?

Since the samples are drawn i.i.d., the probability density of the whole dataset $\mathcal{D}$ is

$$p(\mathcal{D}; \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

Note: Generally, multiple random variables $X_1, X_2, \ldots, X_D$ are **statistically independent** if their distribution factorizes:

$$p(X_1, X_2, \ldots, X_D) = p(X_1) \times p(X_2) \times \cdots \times p(X_D)$$

The probability density $p(\mathcal{D}; \boldsymbol{\theta})$ factorizes due to the i.i.d. assumption.

The quantity

$$p(\mathcal{D};\boldsymbol{\theta}) = \prod_{i=1}^{N} p(\boldsymbol{x}^{(i)};\boldsymbol{\theta})$$

depends on the dataset $\mathcal{D}$ and the parameters $\boldsymbol{\theta}$.

- When interpreted as function of $\mathcal{D}$: probability density of $\mathcal{D}$
- When interpreted as function of $\boldsymbol{\theta}$: **likelihood** of $\boldsymbol{\theta}$

Maximum Likelihood Estimator: Parameters $\boldsymbol{\theta}^*$ which maximize the likelihood

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \; p(\mathcal{D};\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \; \prod_{i=1}^{N} p(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),$$

i.e., the parameters $\boldsymbol{\theta}^*$ under which the data is most likely.

Instead of likelihood, we can maximize the **log-likelihood**:

$$\mathcal{L}(\boldsymbol{\theta}) := \log \prod_{i=1}^{N} p(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

The log-likelihood is measured in **bits** (logarithm of base 2) or **nats** (logarithm of base e). Since the log is a **strictly increasing** function, the log-likelihood has exactly the same maxima as the likelihood, i.e. maximizing log-likelihood is equivalent to maximizing likelihood.

Why the Log?

Optimizing the log-likelihood (sum of terms) is usually easier than optimizing the likelihood (product of factors). Moreover, when using numerical optimizers, the log-likelihood is **numerically stable**, while the likelihood, a product of many small factors, becomes quickly numerical zero.

Why is maximizing the (log-)likelihood meaningful?

In density estimation, we want to find parameters $\boldsymbol{\theta}$ such that

$$p(\boldsymbol{x};\boldsymbol{\theta}) \approx p_{true}(\boldsymbol{x})$$

To quantify the approximation quality ("$\approx$") we require a distance or **divergence measure** for comparing $p(\boldsymbol{x};\boldsymbol{\theta})$ and $p_{true}(\boldsymbol{x})$. One of the most widely divergences is the **Kullback-Leibler divergence**:

$$KL(p_{true}||p) = \mathbb{E}_{p_{true}}\left[\log \frac{p_{true}(\boldsymbol{x})}{p(\boldsymbol{x};\boldsymbol{\theta})}\right] = \int p_{true}(\boldsymbol{x}) \log \frac{p_{true}(\boldsymbol{x})}{p(\boldsymbol{x};\boldsymbol{\theta})} \mathrm{d}\boldsymbol{x}$$

It holds that

- $KL(p_{true}||p) \geq 0$ and
- $KL(p_{true}||p) = 0$ iff $p_{true} \equiv p$ almost everywhere.

Kullback-Leibler Divergence and Log-likelihood

Note that the KL divergence can be written as

$$
\begin{aligned}
KL(p_{true}||p) &= \int p_{true}(\boldsymbol{x}) \log \frac{p_{true}(\boldsymbol{x})}{p(\boldsymbol{x};\boldsymbol{\theta})} \mathrm{d}\boldsymbol{x} \\
&= \int p_{true}(\boldsymbol{x}) \left[\log p_{true}(\boldsymbol{x}) - \log p(\boldsymbol{x};\boldsymbol{\theta})\right] \mathrm{d}\boldsymbol{x} \\
&= \mathbb{E}_{p_{true}} \left[\log p_{true}(\boldsymbol{x}) - \log p(\boldsymbol{x};\boldsymbol{\theta})\right] \\
&= \underbrace{\mathbb{E}_{p_{true}} \left[\log p_{true}(\boldsymbol{x})\right]}_{\text{neg. entropy } -H[p_{true}], const.} - \mathbb{E}_{p_{true}} \left[\log p(\boldsymbol{x};\boldsymbol{\theta})\right]
\end{aligned}
$$

The second term, $-\mathbb{E}_{p_{true}} \left[\log p(\boldsymbol{x};\boldsymbol{\theta})\right]$, can be estimated with Monte Carlo, which is proportional to the log-likelihood:

$$
-\mathbb{E}_{p_{true}} \left[\log p(\boldsymbol{x};\boldsymbol{\theta})\right] \approx \overbrace{-\frac{1}{N} \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)};\boldsymbol{\theta})}^{\text{log-likelihood } \mathcal{L}(\boldsymbol{\theta})}
$$

Kullback-Leibler Divergence and Log-likelihood cont'd

Thus, maximum likelihood (ML) is well-justified for density approximation:

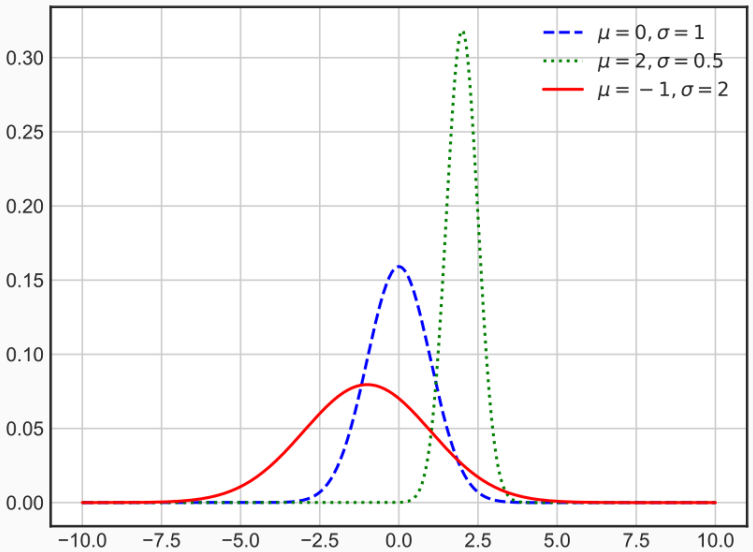
- maximizing log-likelihood is equivalent to minimizing a Monte Carlo estimate of the KL divergence $KL(p_{true}||p)$
- if p_{true} is contained in the model class $\{p(\boldsymbol{x};\boldsymbol{\theta})\}_{\boldsymbol{\theta}}$, then the maximum likelihood distribution $p(\boldsymbol{x};\boldsymbol{\theta}^*)$ will converge to p_{true} when $N \to \infty$
- if p_{true} is not contained in the model class $\{p(\boldsymbol{x};\boldsymbol{\theta})\}_{\boldsymbol{\theta}}$, $p(\boldsymbol{x};\boldsymbol{\theta}^*)$ will converge to the best approximation of p_{true} in KL sense

Gaussian Distribution, Normal Distribution

Let p be a probability density defined as

$$p(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

with parameters **mean** μ and **standard deviation** $\sigma > 0$. A distribution with density p is called **Gaussian distribution** or **normal distribution**.



Carl Friedrich Gauss 1777–1855

Image: wikipedia

Maximum Likelihood for Gaussians

- Assume i.i.d. dataset $\mathcal{D} = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$
- The Gaussian pdf is $p(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$
- Thus, the log-likelihood is

$$\mathcal{L}(\mu, \sigma) = \log p(\mathcal{D}; \mu, \sigma) = \sum_{i=1}^{N} -\log(\sqrt{2\pi}\,\sigma) - \frac{1}{2}\left(\frac{x^{(i)} - \mu}{\sigma}\right)^2$$

- We want to maximize $\mathcal{L}(\mu, \sigma)$ w.r.t. μ and $\sigma > 0$
- It turns out, that we can solve the problem step-wise: first find a maximizer for μ, then for σ

Maximum Likelihood for Gaussians: μ^*

The log-likelihood is a negative quadratic function in μ:

$$\mathcal{L}(\mu, \sigma) = \sum_{i=1}^{N} -\log(\sqrt{2\pi}\,\sigma) - \frac{1}{2}\left(\frac{x^{(i)} - \mu}{\sigma}\right)^2$$

Derive after μ and set to 0:

$$\frac{\partial \mathcal{L}(\mu, \sigma)}{\partial \mu} = \sum_{i=1}^{N} -\frac{1}{2}\,2\left(\frac{x^{(i)} - \mu}{\sigma}\right)\left(-\frac{1}{\sigma}\right) \stackrel{!}{=} 0$$

Multiplying by σ^2 and bringing μ on the other side delivers the **average** as maximum likelihood (ML) solution:

$$\sum_{i=1}^{N} x^{(i)} = \sum_{i=1}^{N} \mu^* = N\mu^* \quad \Rightarrow \quad \mu^* = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$$

Note that this is the ML solution for μ for **any** σ. Thus we can already assume μ^* when computing σ^*.

Maximum Likelihood for Gaussians: σ^*

$$\mathcal{L}(\mu^*, \sigma) = \sum_{i=1}^{N} -\log(\sqrt{2\pi}\,\sigma) - \frac{1}{2}\left(\frac{x^{(i)} - \mu^*}{\sigma}\right)^2$$

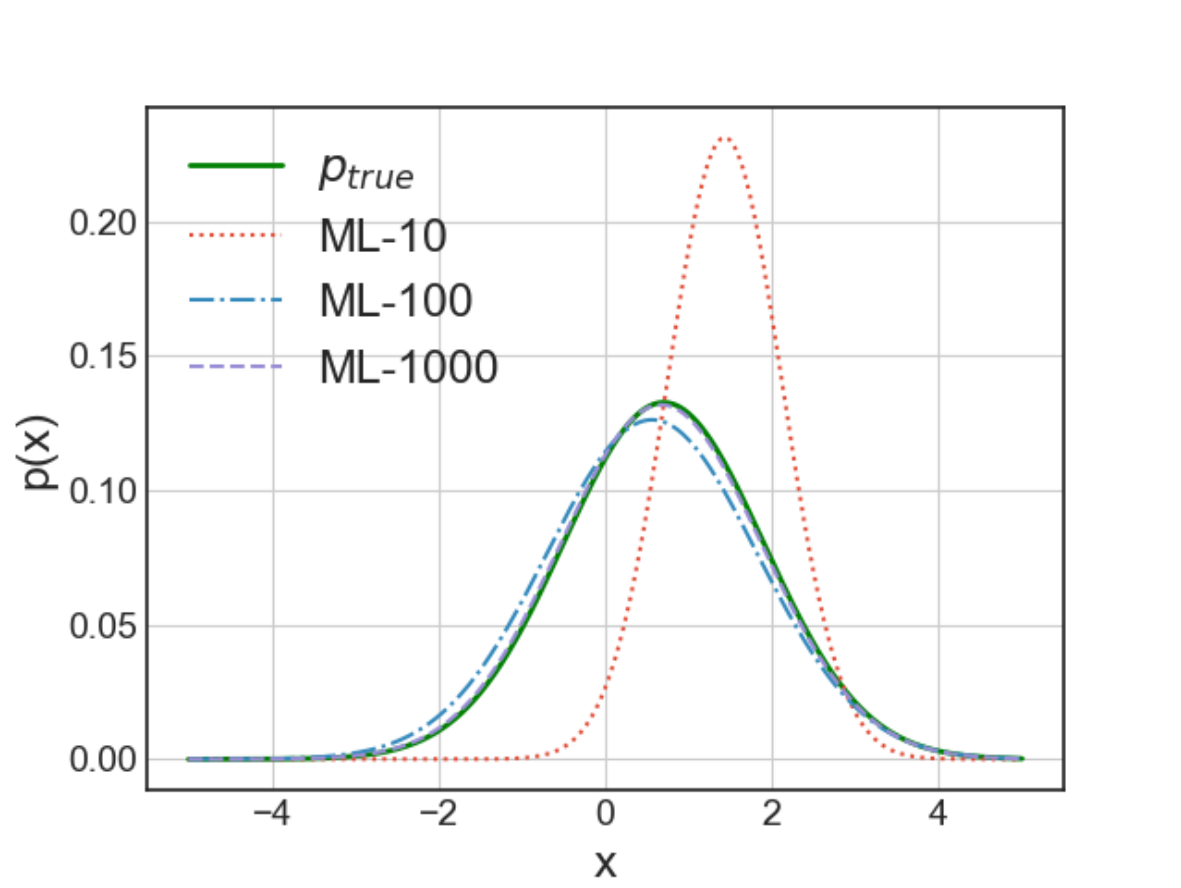
Deriving after σ:

$$\begin{aligned}\frac{\partial \mathcal{L}(\mu, \sigma)}{\partial \sigma} &= \sum_{i=1}^{N} -\frac{\sqrt{2\pi}}{\sqrt{2\pi}\,\sigma} - \frac{1}{2}\,2\left(\frac{x^{(i)} - \mu^*}{\sigma}\right)(-1)\left(\frac{x^{(i)} - \mu^*}{\sigma^2}\right) \\ &= \sum_{i=1}^{N} -\frac{1}{\sigma} + \left(\frac{(x^{(i)} - \mu^*)^2}{\sigma^3}\right) \stackrel{!}{=} 0\end{aligned}$$

Multiplying by σ^3 and resolving after σ delivers the **empirical standard deviation** as ML solution:

$$\sigma^* = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x^{(i)} - \mu^*)^2}$$

Drawing three datasets with $N = 10$, $N = 100$ and $N = 1000$ from a ground-truth Gaussian with $\mu_{true} = 0.7$ and $\sigma_{true} = 1.2$. With increased sample size, the ML fit approaches the ground-truth:

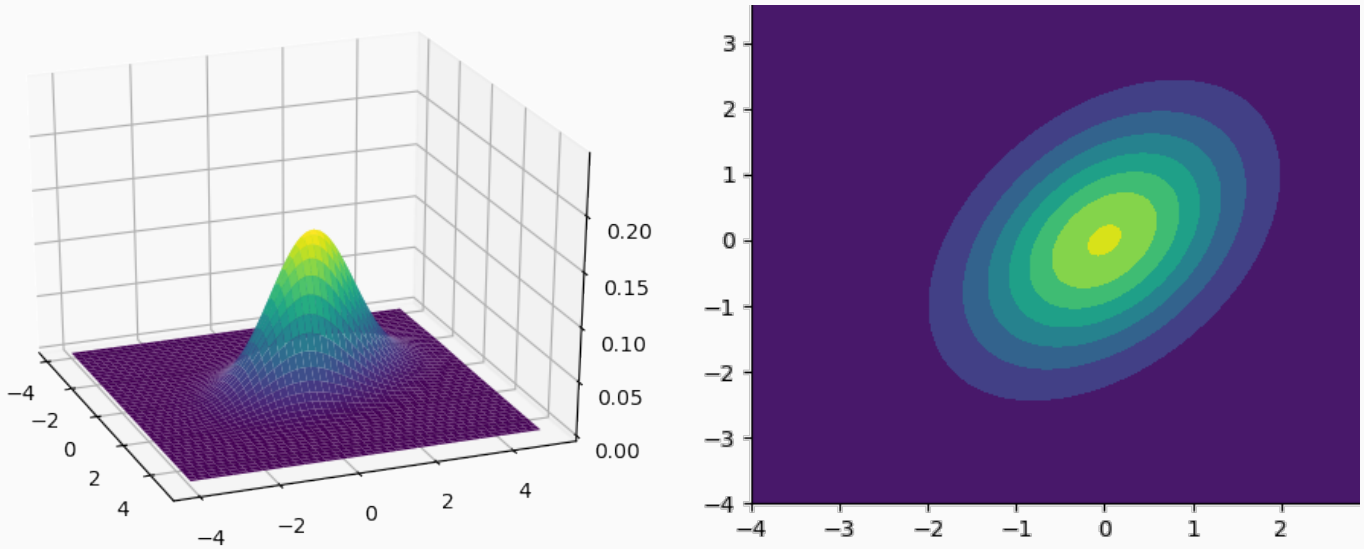


Multivariate Gaussian

Let p be defined as

$$p(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}$$

with D-dimensional **mean vector** $\boldsymbol{\mu}$ and $D \times D$ positive definite **covariance matrix** Σ. A distribution with density p is called **multivariate Gaussian distribution**. It generalizes Gaussians to higher dimensions.



Maximum Likelihood for Multivariate Gaussians

- The ML solution for 1d Gaussians...

$$\mu^* = \frac{1}{N}\sum_{i=1}^{N} x^{(i)} \qquad \sigma^{2*} = \frac{1}{N}\sum_{i=1}^{N} (x^{(i)} - \mu^*)^2$$

- ...carries over to multivariate Gaussians:

$$\boldsymbol{\mu}^* = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}^{(i)} \qquad \Sigma^* = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^*)(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^*)^T$$

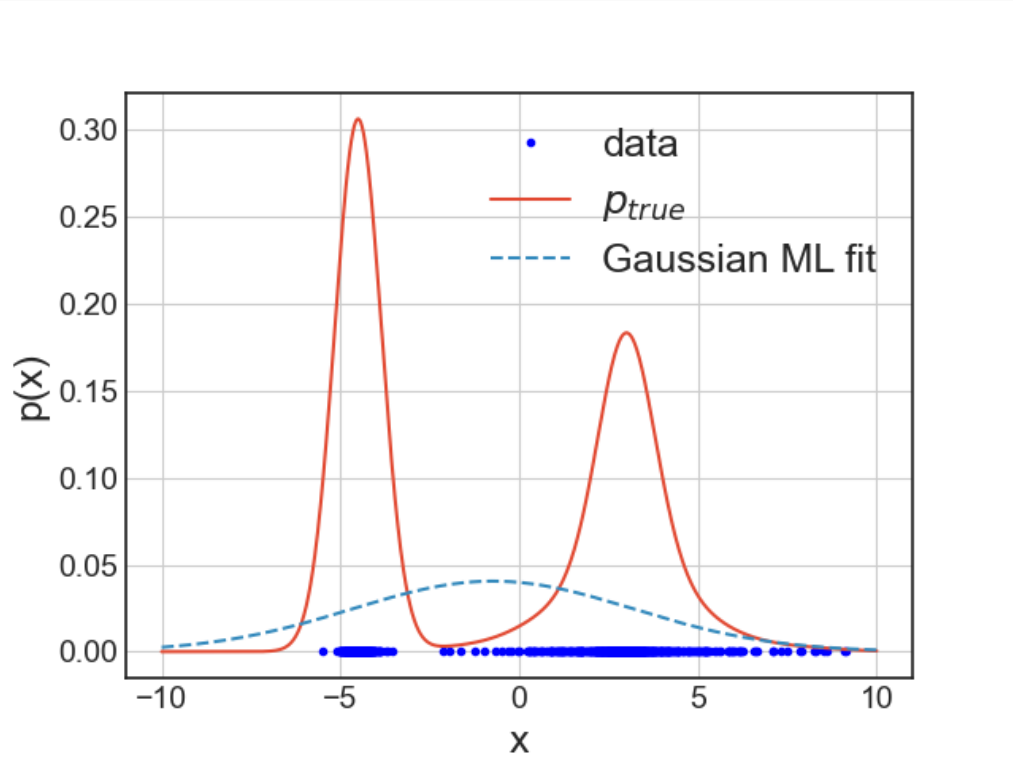
Gaussian Mixture Models

Are Gaussian models always a good choice?

A Gaussian (Mis)fit

Example

Assume a true distribution p_{true} as shown below. We collect $N = 1000$ samples and fit a Gaussian with maximum likelihood:



The fit is awful, since p_{true} is non-Gaussian. In particular, it is a **bimodal** distribution (**mode**: local maximum of density).

Let $K \geq 1$ and

- $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$ be K mean parameters
- $\Sigma_1, \Sigma_2, \ldots, \Sigma_K$ be K covariance matrices
- $w_1, w_2, \ldots, w_K$ be K **mixture weights** with $w_k \geq 0$ and $\sum_k w_k = 1$

The **Gaussian mixture model (GMM)** parametrized by $\boldsymbol{\theta} = \{w_k, \boldsymbol{\mu}_k, \Sigma_k\}_{k=1}^K$ is given as

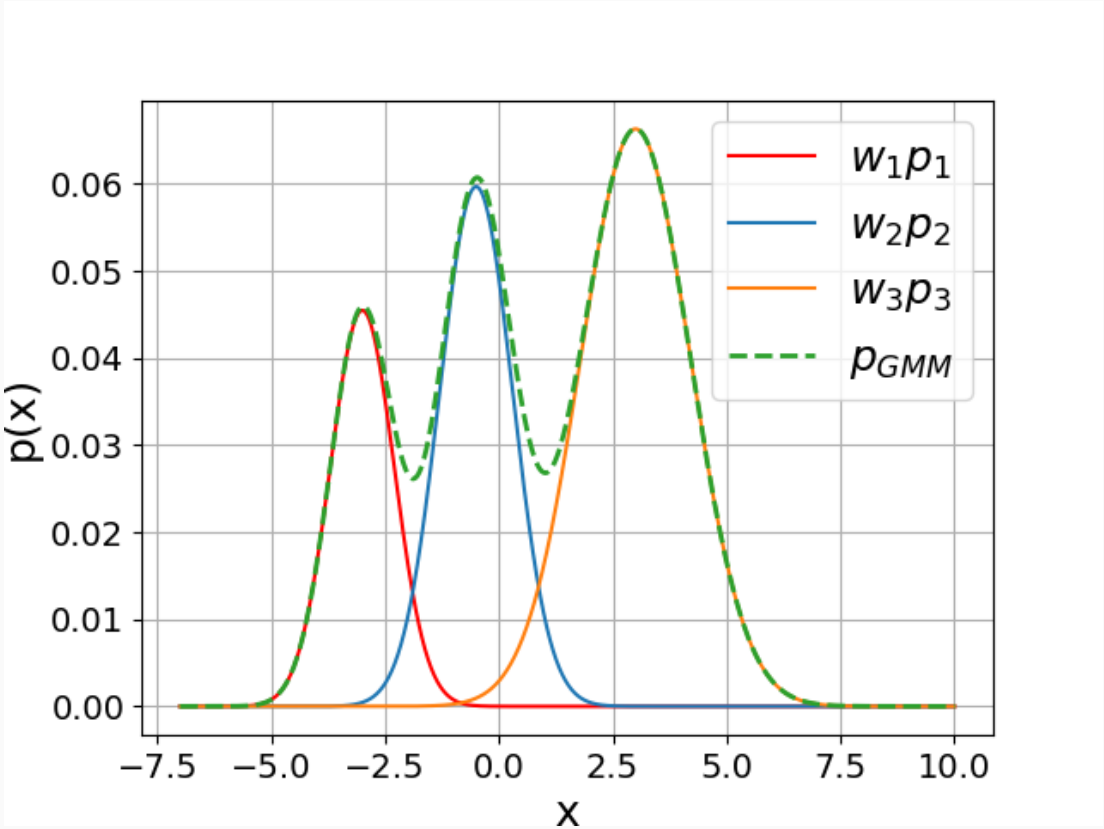
$$p_{GMM}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{k=1}^{K} w_k \, p(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k)$$

where $p(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k)$ is the Gaussian density with parameters $\boldsymbol{\mu}_k, \Sigma_k$.

That is, a GMM is a **convex combination** of K **Gaussian components**.

Gaussian Mixture Model, 1 Dimension

Example



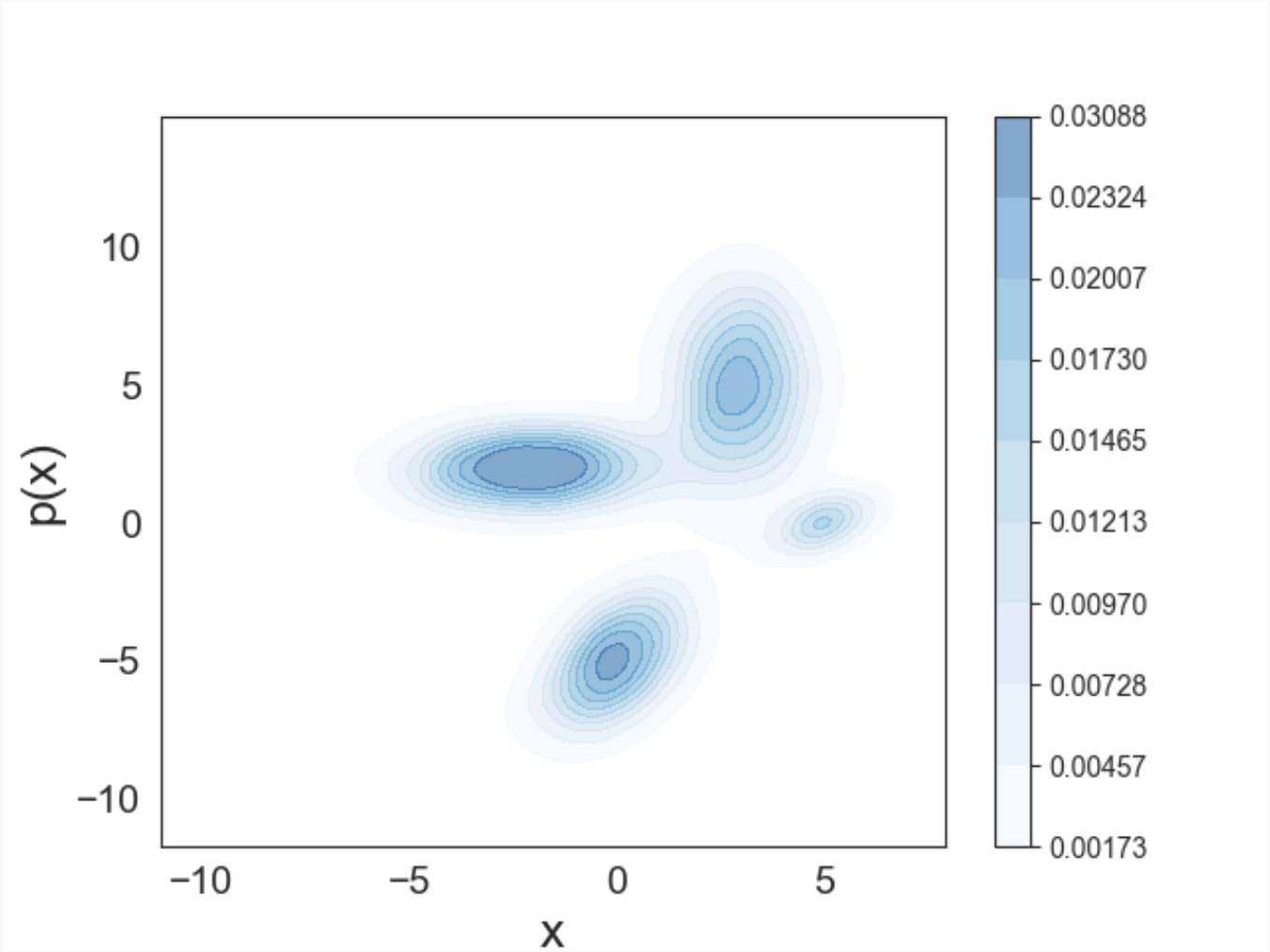
3 components:

$w_1 = 0.2, w_2 = 0.3, w_3 = 0.5,$

$\mu_1 = -3, \mu_2 = -0.5, \mu_3 = 3,$

$\sigma_1 = 0.7, \sigma_2 = 0.8, \sigma_3 = 1.2$

A GMM with 4 components in 2 dimensions:



Properties of GMMs

Any GMM specifies a proper probability density, since

- $p_{GMM}(\boldsymbol{x})$ is non-negative:
  - $w_k \geq 0$ and
  - $p(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k) \geq 0$
  - Hence, $p_{GMM}(\boldsymbol{x}) = \sum_{k=1}^{K} w_k \, p(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k) \geq 0$
- p_{GMM} is normalized:

$$\begin{aligned}\int p_{GMM}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} &= \int \sum_{k=1}^{K} w_k \, p(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k) \, \mathrm{d}\boldsymbol{x} \\ &= \sum_{k=1}^{K} w_k \int p(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k) \, \mathrm{d}\boldsymbol{x} \\ &= \sum_{k=1}^{K} w_k \, 1 = 1\end{aligned}$$

Properties of GMMs cont'd

GMMs are **universal approximators** of densities, that is, any density can be approximated arbitrarily well by a GMM. **This is true even when restricted Gaussian with only diagonal covariances.**

However, we don't know a big K needs to be and how to set the parameters.

Expectation-Maximization

GMM as Latent Variable Model

Note that the mixture weights w_k are non-negative and sum to one. Thus, we can interpret them as the distribution of a **categorical latent variable** Z taking values in $\{1, \ldots, K\}$:

$$p_{GMM}(\boldsymbol{x}) = \sum_{k=1}^{K} w_k \, p(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k)$$

$$= \sum_{k=1}^{K} p(z = k) \, p(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k)$$

Thus, a GMM actually defines a joint distribution over $\boldsymbol{X}$ and Z:

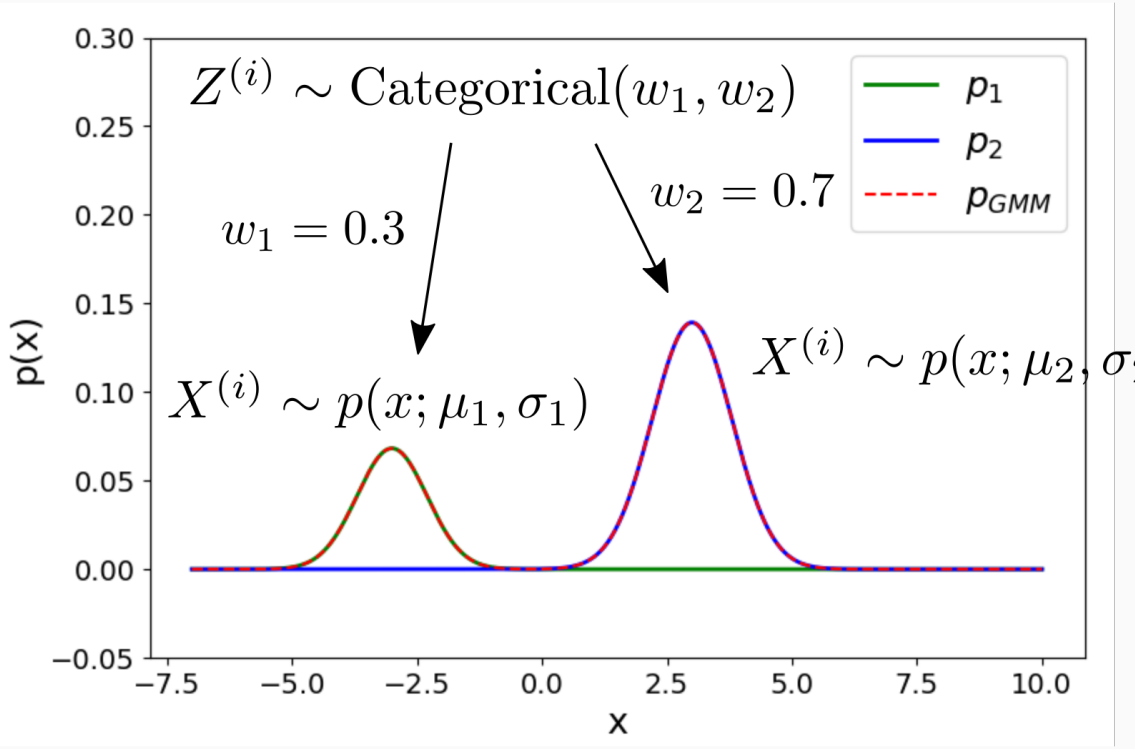
$$p_{GMM}(\boldsymbol{x}, z = k) = \overbrace{p(z = k)}^{w_k} \overbrace{p(\boldsymbol{x} \mid z = k)}^{p(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k)}$$

and $p_{GMM}(\boldsymbol{x})$ is actually just the **marginal distribution**

$$p_{GMM}(\boldsymbol{x}) = \sum_{k} p_{GMM}(\boldsymbol{x}, z = k)$$

Thus, a GMM can be seen as a **hierarchical generative model**:
For each $i = 1 \dots N$, generate $Z^{(i)}, \boldsymbol{X}^{(i)}$ as follows:

$$Z^{(i)} \sim \text{Categorical}(w_1, \dots, w_K)$$
$$\boldsymbol{X}^{(i)} \sim p(\boldsymbol{x} \mid z^{(i)}) = p(\boldsymbol{x}; \boldsymbol{\mu}_{z^{(i)}}, \Sigma_{z^{(i)}})$$



Connection to K-Means

Recall, in k-means, we had $z_{i,k} = 1$ iff $\boldsymbol{x}^{(i)}$ belongs to k^{th} cluster.

The latent variables $Z^{(i)}$ in GMMs have exactly the same meaning: they encode which Gaussian component has generated $\boldsymbol{x}^{(i)}$. Thus, GMMs can be seen as probabilistic clustering, or "soft clustering."

Hen and Egg Problem: In GMMs, neither the parameters $\boldsymbol{\theta} = \{w_k, \boldsymbol{\mu}_k, \Sigma_k\}_{k=1}^{K}$ nor the latent variables $\boldsymbol{Z} = \{Z^{(1)}, Z^{(2)}, \ldots, Z^{(N)}\}$ are known. The problem would be easy, if either were known.

The Hen and Egg Problem

If Z was known, ML estimation for θ would be easy:

- $\boldsymbol{Z}$ specify clusters: $\mathcal{D}_k = \{\boldsymbol{x}^{(i)} \mid z^{(i)} = k\}$
- mean vectors $\boldsymbol{\mu}_k^*$ are estimated as the means of the k^{th} cluster

$$\boldsymbol{\mu}_k^* = \frac{1}{|\mathcal{D}_k|} \sum_{\boldsymbol{x} \in \mathcal{D}_k} \boldsymbol{x}$$

- covariance matrices Σ_k^* are estimated as empirical covariances of $\mathcal{D}_k$

$$\Sigma_k^* = \frac{1}{|\mathcal{D}_k|} \sum_{\boldsymbol{x} \in \mathcal{D}_k} (\boldsymbol{x} - \boldsymbol{\mu}_k^*)(\boldsymbol{x} - \boldsymbol{\mu}_k^*)^T$$

- the ML mixture weights are the empirical frequencies of $Z^{(i)}$:

$$w_k^* = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(z^{(i)} = k)$$

The Hen and Egg Problem cont'd

If θ was known, we could infer Z via Bayes rule: *

$$p(Z^{(i)} = k \mid \boldsymbol{x}^{(i)}) = \frac{p(\boldsymbol{x}^{(i)} \mid Z^{(i)} = k)\, p(Z^{(i)} = k)}{p(\boldsymbol{x}^{(i)})}$$

$$= \frac{p(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}_k, \Sigma_k)\, w_k}{\sum_{k'} p(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}_{k'}, \Sigma_{k'})\, w_{k'}} =: \gamma_{i,k}$$

Note that $\gamma_{i,k} \geq 0$ and $\sum_k \gamma_{i,k} = 1$.

$\gamma_{i,k}$: posterior over $Z^{(i)}$ given $\boldsymbol{x}^{(i)}$, also called "**cluster responsibilities**." $\gamma_{i,k}$ can be interpreted as a fractional weight, i.e., how much the i^{th} sample belongs to the k^{th} component. Furthermore, $\gamma_{i,k}$ can be seen as a "soft" version of the "hard" cluster assignments $z_{i,k} \in \{0, 1\}$ in k-means.

*Bayes rule in general: $p(Y \mid X) = \frac{p(X \mid Y)\, p(Y)}{p(X)}$

Idea of the **Expectation-Maximization (EM)** algorithm:

Repeat:

- **E-step:** Compute cluster responsibilities $\gamma_{i,k}$ using the current parameters $\boldsymbol{\theta}$
- **M-step:** Update $\boldsymbol{\theta}$ using maximum likelihood with samples weighted by $\gamma_{i,k}$

Expectation Maximization Algorithm for GMMs

input: Data $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\}$

output: Learned GMM parameters $\boldsymbol{\theta} = \{w_k, \boldsymbol{\mu}_k, \Sigma_k\}_{k=1}^{K}$

Initialize $\boldsymbol{\theta} = \{w_k, \boldsymbol{\mu}_k, \Sigma_k\}_{k=1}^{K}$ randomly

for *counter* $= 1 \ldots$ *max number of iterations* **do**

  E-Step

  for $i = 1 \ldots N$, $k = 1 \ldots K$ **do**

    let $\gamma_{i,k} \leftarrow \frac{p(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}_k, \Sigma_k)\, w_k}{\sum_{k'} p(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}_{k'}, \Sigma_{k'})\, w_{k'}}$ ▷ cluster responsibilities

  end

  M-Step

  for $k = 1 \ldots K$ **do**

    $\boldsymbol{\mu}_k \leftarrow \frac{\sum_{i=1}^{N} \gamma_{i,k}\, \boldsymbol{x}^{(i)}}{\sum_{i=1}^{N} \gamma_{i,k}}$ ▷ weighted mean

    $\Sigma_k \leftarrow \frac{\sum_{i=1}^{N} \gamma_{i,k} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k)(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^{N} \gamma_{i,k}}$ ▷ weighted covariance

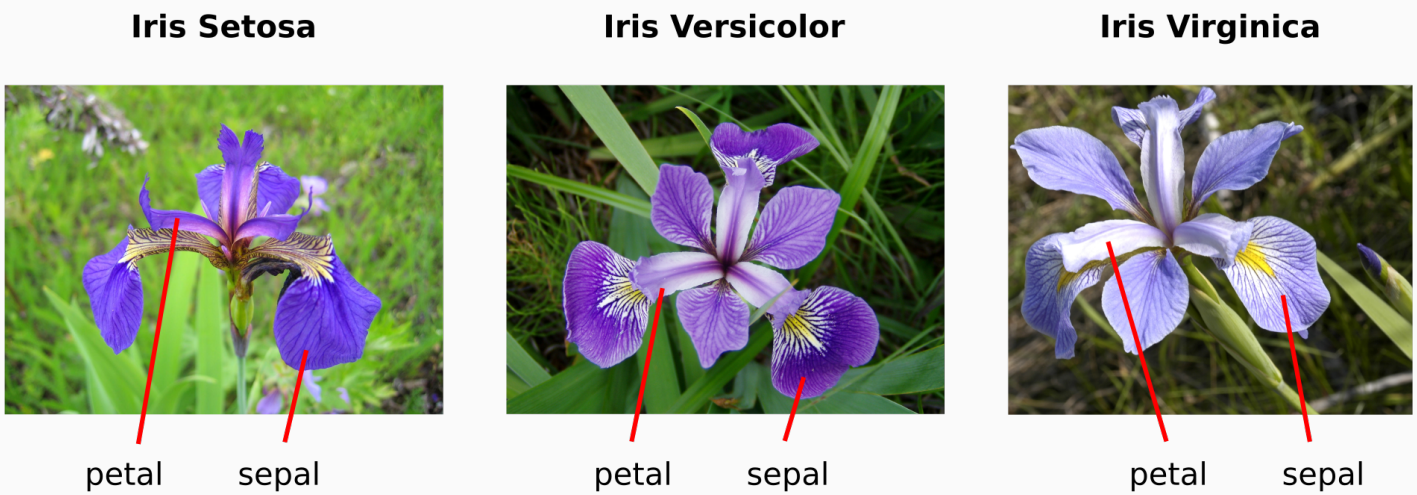
    $w_k \leftarrow \frac{\sum_{i=1}^{N} \gamma_{i,k}}{N}$ ▷ weighted empirical frequencies

  end

end

Learning GMM with EM on Iris Data

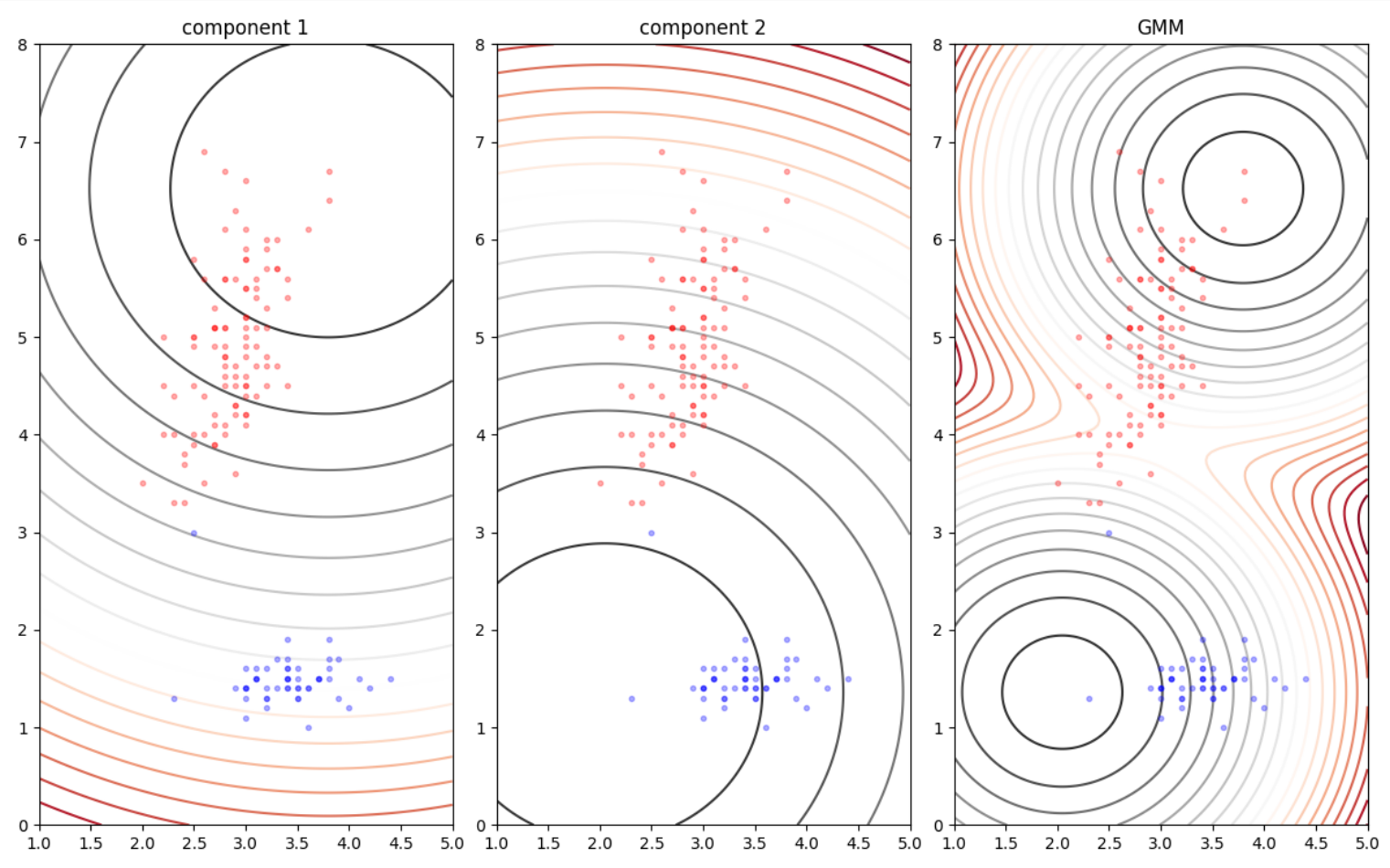
Example



- $D = 2$ dimension
- $K = 2$ components
- Initialization $w_1 = w_2 = 0.5$
- μ_1, μ_2: uniformly at random from data range
- $\Sigma_1 = \Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Learning GMM with EM on Iris Data

Example

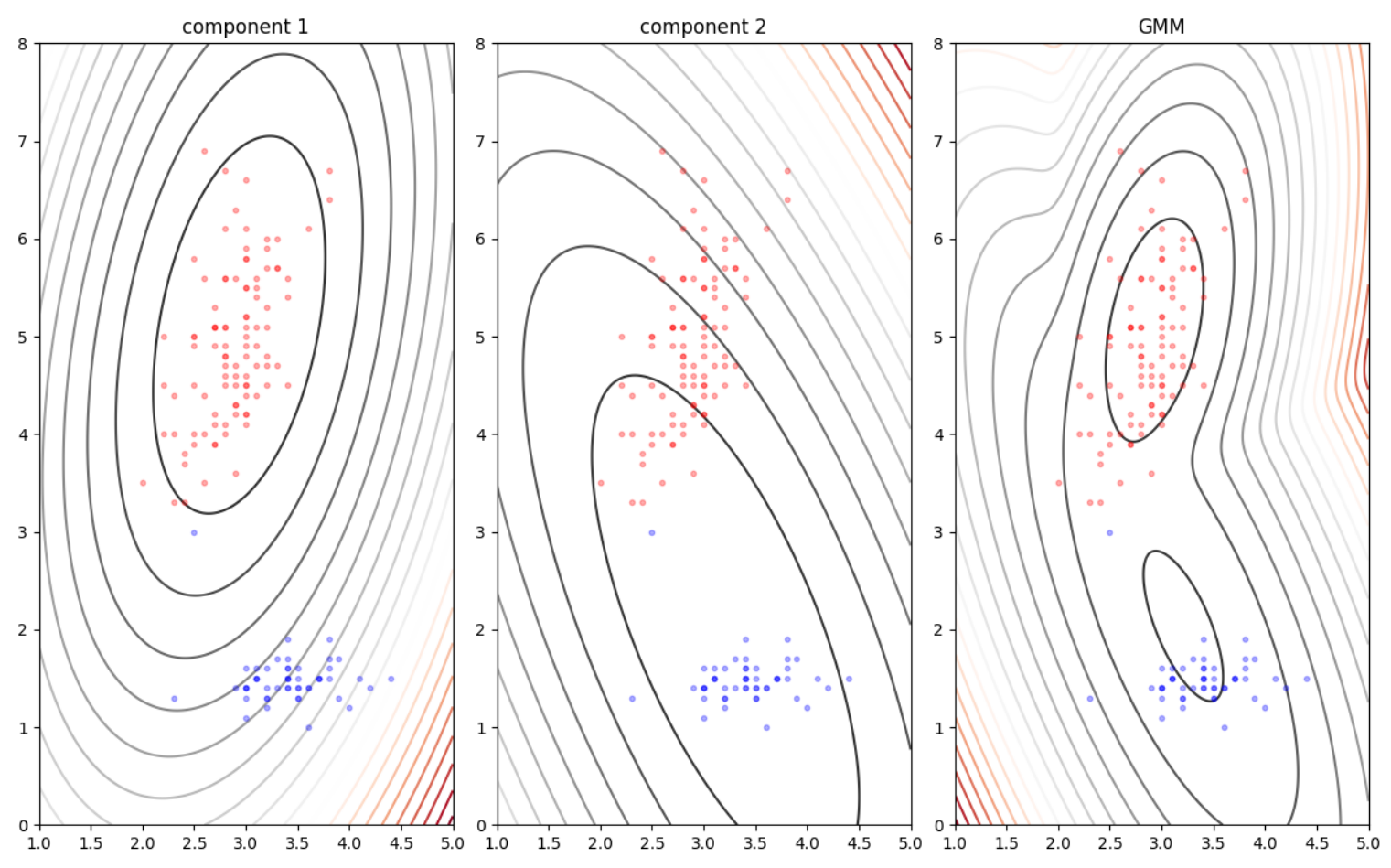


Init: log-likelihood $= -604.86\,\mathrm{nats}$, $w_1 = 0.5$, $w_2 = 0.5$

The blue and red labels is the clustering by K-Means; this **not** used by EM

Learning GMM with EM on Iris Data

Example

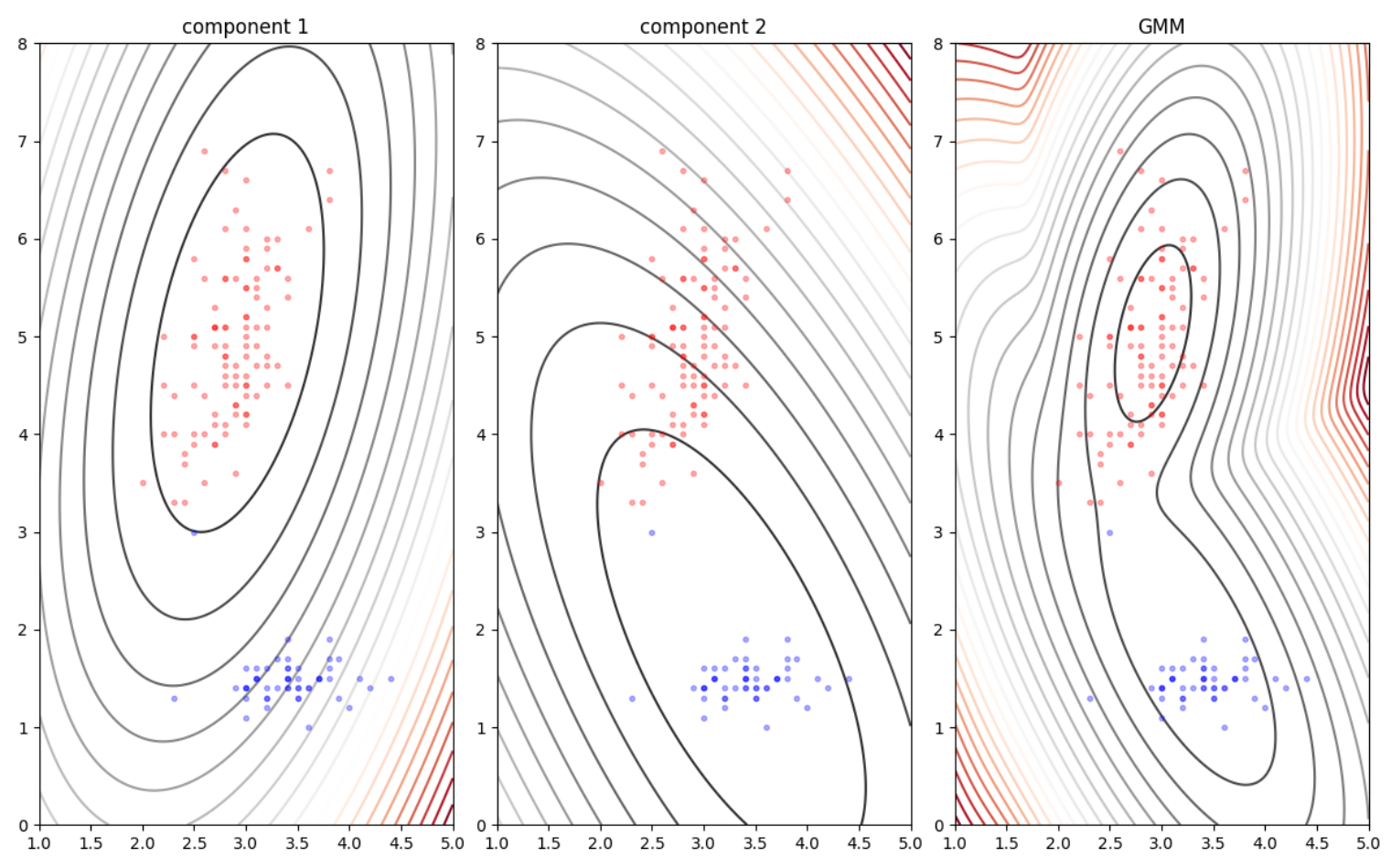


Iteration 1: log-likelihood $= -309.27\,\mathrm{nats}$, $w_1 = 0.56$, $w_2 = 0.44$

The blue and red labels is the clustering by K-Means; this **not** used by EM

Learning GMM with EM on Iris Data

Example

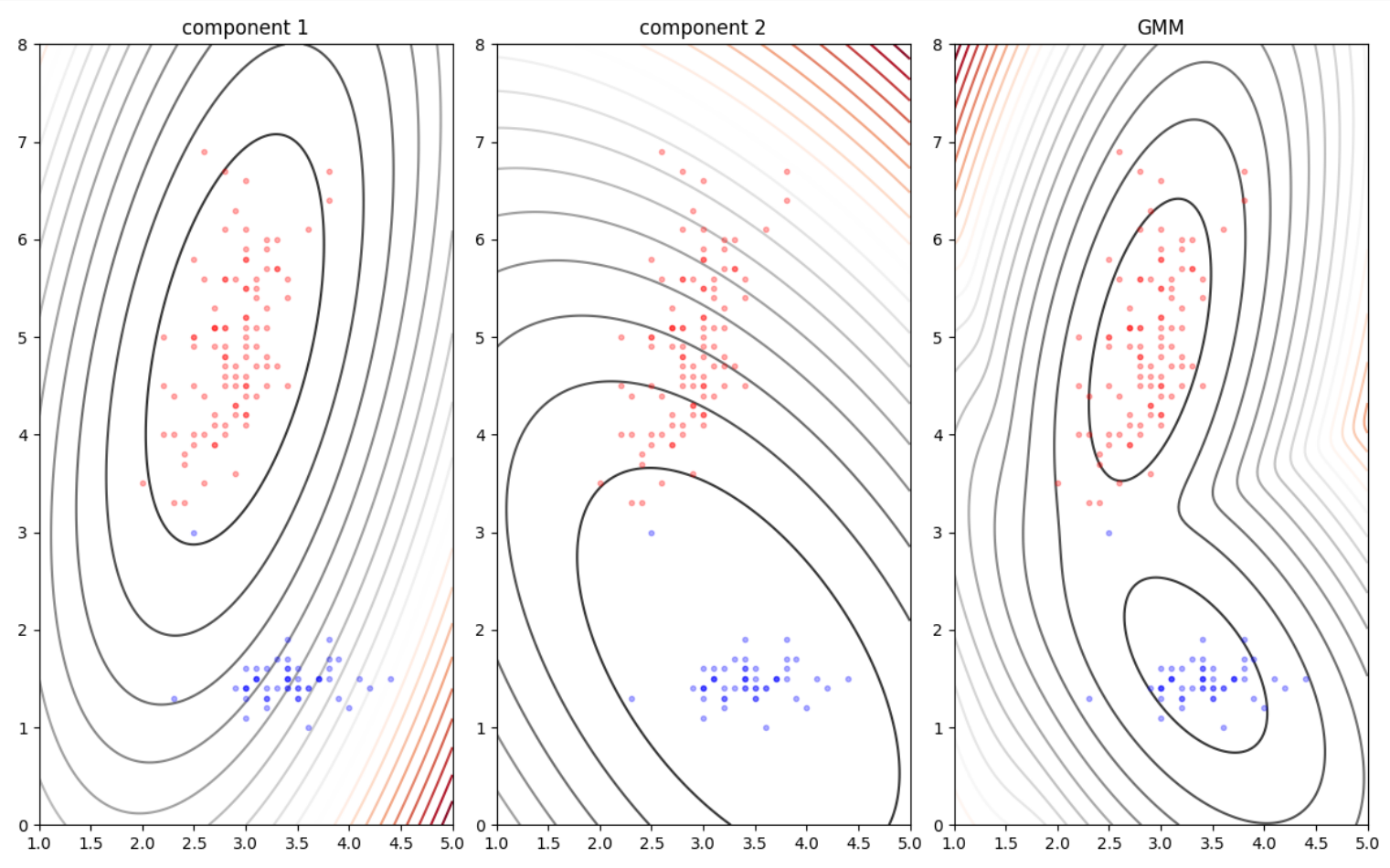


Iteration 5: log-likelihood $= -299.75\,\mathrm{nats}$, $w_1 = 0.6$, $w_2 = 0.4$

The blue and red labels is the clustering by K-Means; this **not** used by EM

Learning GMM with EM on Iris Data

Example

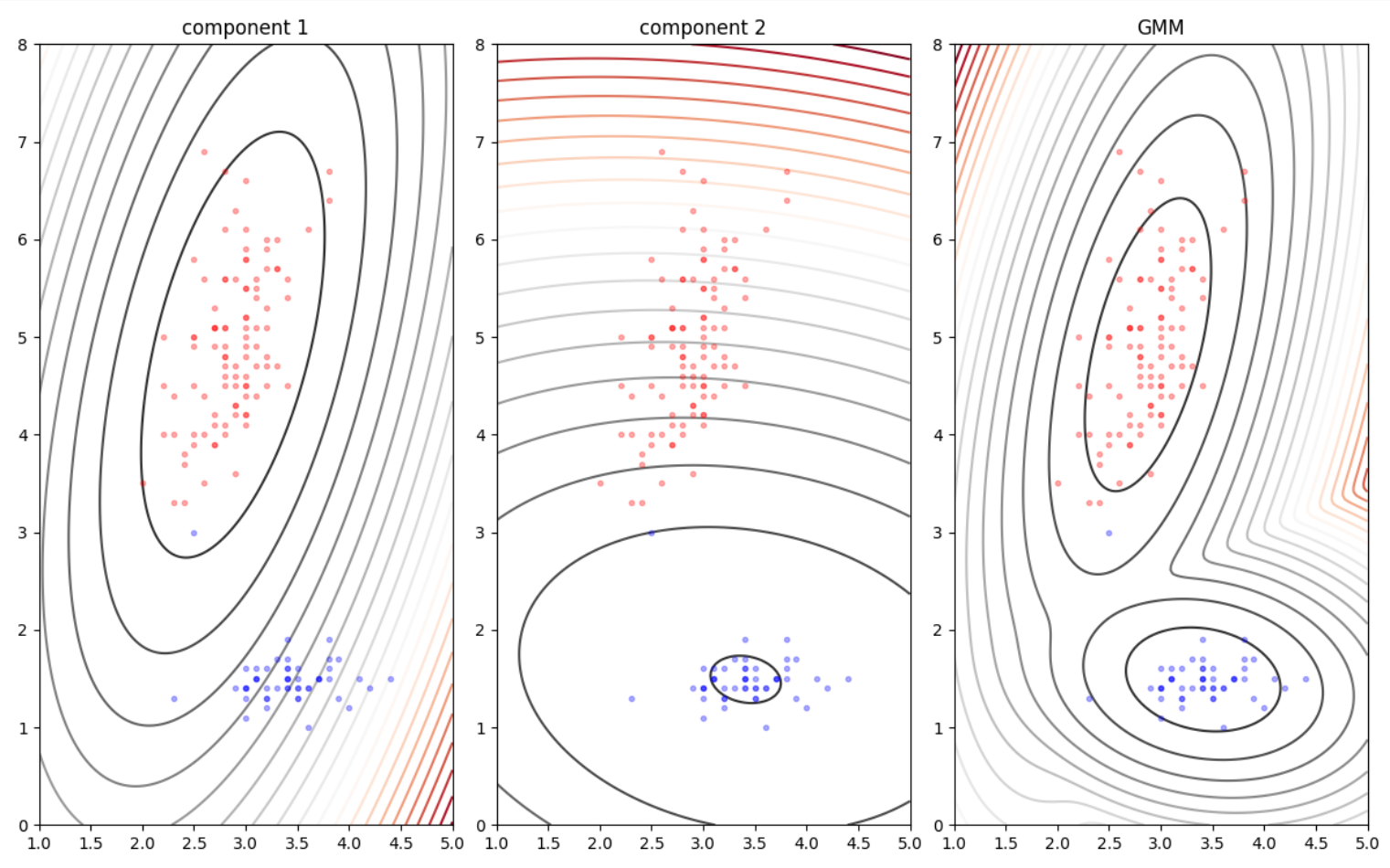


Iteration 10: log-likelihood $= -282.92\,\mathrm{nats}$, $w_1 = 0.64$, $w_2 = 0.36$

The blue and red labels is the clustering by K-Means; this **not** used by EM

Learning GMM with EM on Iris Data

Example

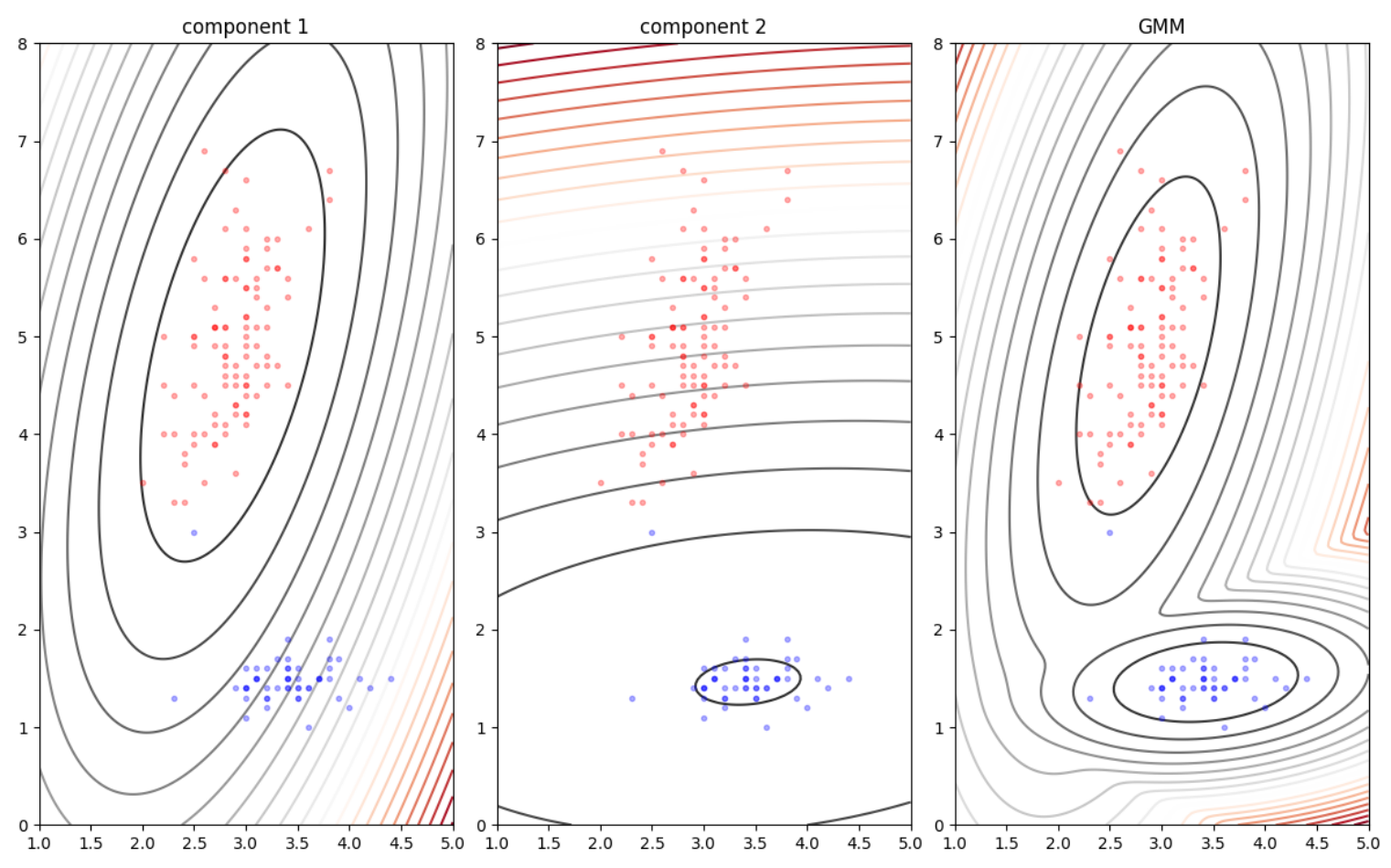


Iteration 15: log-likelihood $= -248.68\,\mathrm{nats}$, $w_1 = 0.66$, $w_2 = 0.34$

The blue and red labels is the clustering by K-Means; this **not** used by EM

Learning GMM with EM on Iris Data

Example

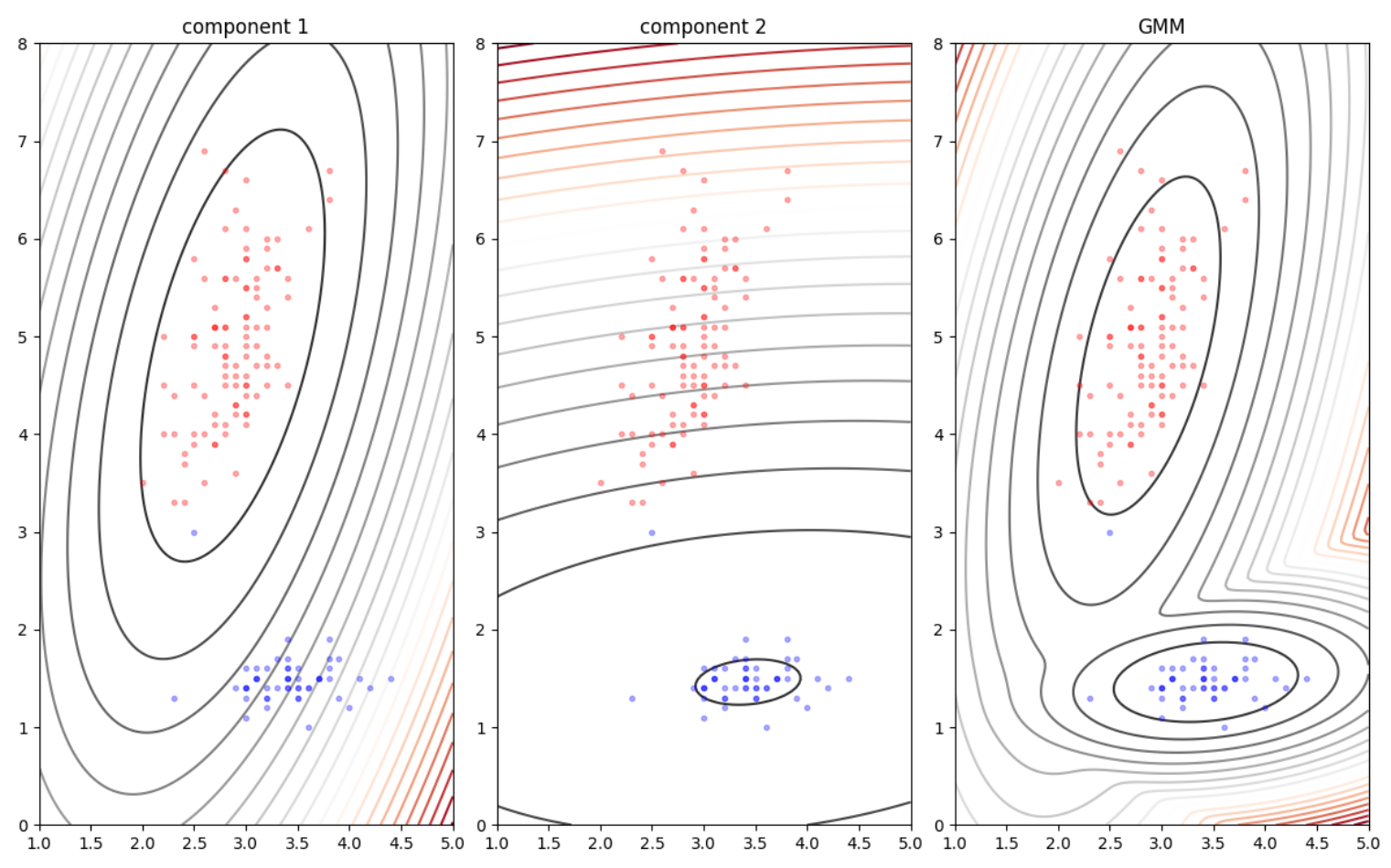


Iteration 20: log-likelihood $= -237.35\,\mathrm{nats}$, $w_1 = 0.67$, $w_2 = 0.33$

The blue and red labels is the clustering by K-Means; this **not** used by EM

Learning GMM with EM on Iris Data

Example



Iteration 25: log-likelihood $= -237.35\,\mathrm{nats}$, $w_1 = 0.67$, $w_2 = 0.33$

The blue and red labels is the clustering by K-Means; this **not** used by EM

Properties of the EM Algorithm

- **Monotonous in log-likelihood:** In each iteration, the GMM's log-likelihood is **guaranteed** to increase or stay the same (when converged)
- Thus, we can terminate the algorithm when improvement is small (stopping criterion)
- However, for GMMs, the log-likelihood is non-concave in the parameters, thus only a **local maximum** can be achieved in general
- EM tends to **increase log-likelihood rapidly in the beginning**; close to an optimum it has a rather slow theoretical convergence

Summary

- Gaussian distributions might be a bad fit
- Gaussian mixture models: universal approximators of densities
- Related to K-means: hard vs. soft clustering
- EM algorithm analogous to K-means algorithm
  - assign data points to clusters
  - update parameters
- EM algorithm monotonous in (log-)likelihood, but converges to local optimum only

Please Evaluate

ML1 (VO) is nominated for an Award for Excellence in Teaching