Chapter 12

Bayesian Classifier

Chapter List

- Bayes Decision Theory
- Maximum Likelihood Estimation
- Naïve Bayes Classifier
- EM Algorithm

Chapter List

- **Bayes Decision Theory**
- Maximum Likelihood Estimation
- Naïve Bayes Classifier
- EM Algorithm

Bayes Decision Theory

- Bayesian decision theory is a fundamental decision-making approach under the probability framework.
  - When all relevant probabilities were known, Bayesian decision theory makes optimal classification decisions based on the probabilities and costs of misclassifications.

Bayes Decision Theory

- Bayesian decision theory is a fundamental decision-making approach under the probability framework.
  - When all relevant probabilities were known, Bayesian decision theory makes optimal classification decisions based on the probabilities and costs of misclassifications.
- Let us assume that there are N distinct class labels, that is, $y = \{c_1, c_2, \ldots, c_N\}$. Let λ_{ij} denote the cost of misclassifying a sample of class c_j as class c_i . Then, with the posterior probability $P(c_i \mid \boldsymbol{x})$ we can calculate the expected loss of classifying a sample $\mathbf{x}$ as class c_i, that is, the conditional risk of the sample $\mathbf{x}$:

$$R\left(c_i \mid \mathbf{x}\right) = \sum_{j=1}^{N} \lambda_{ij} P\left(c_j \mid \mathbf{x}\right) \qquad (7.1)$$

- Our task is to find a decision rule $h : X \mapsto Y$ that minimizes the overall risk:

$$R(h) = \mathbf{E}_x \left[R(h(\mathbf{x}) \mid \mathbf{x})\right] \qquad (7.2)$$

Bayes Decision Theory

- The overall risk $R(h)$ is minimized when the conditional risk $R(h(\mathbf{x}) \mid \mathbf{x})$ of each sample $\mathbf{x}$ is minimized.

Bayes Decision Theory

- The overall risk $R(h)$ is minimized when the conditional risk $R(h(\mathbf{x}) \mid \mathbf{x})$ of each sample $\mathbf{x}$ is minimized.

- This leads to the Bayes decision rule: to minimize the overall risk, classify each sample as the class that minimizes the conditional risk $R(c \mid \mathbf{x})$

$$h^*(x) = \underset{c \in y}{\operatorname{argmin}} R(c \mid x)$$

  - where h^* is called the Bayes optimal classifier, and its associated overall risk $R(h^*)$ is called the Bayes risk.
  - $1 - R(h^*)$ is the best performance that can be achieved by any classifiers, that is, the theoretically achievable upper bound of accuracy for any machine learning models.

Bayes Decision Theory

- To be specific, if the objective is to minimize the misclassification rate, then the misclassification loss λ_{ij} can be written as

$$\lambda_{i,j} = \begin{cases} 0, & \text{if } i = j; \\ 1, & \text{otherwise,} \end{cases}$$

Bayes Decision Theory

- To be specific, if the objective is to minimize the misclassification rate, then the misclassification loss λ_{ij} can be written as

$$\lambda_{i,j} = \begin{cases} 0, & \text{if } i = j; \\ 1, & \text{otherwise,} \end{cases}$$

- and the conditional risk is

$$R(c \mid \mathbf{x}) = 1 - P(c \mid \mathbf{x})$$

Bayes Decision Theory

- To be specific, if the objective is to minimize the misclassification rate, then the misclassification loss λ_{ij} can be written as

$$\lambda_{i,j} = \begin{cases} 0, & \text{if } i = j; \\ 1, & \text{otherwise}, \end{cases}$$

- and the conditional risk is

$$R(c \mid \mathbf{x}) = 1 - P(c \mid \mathbf{x})$$

- Then, the Bayes optimal classifier that minimizes the misclassification rate is

$$h^*(x) = \underset{c \in y}{\operatorname{argmax}} P(c \mid x)$$

  - which classifies each sample $\mathbf{x}$ as the class that maximizes its posterior probability $P(c \mid \mathbf{x})$.

Bayes Decision Theory

- We can see that the Bayes decision rule relies on the posterior probability $P(c \mid \mathbf{x})$.

- However, it's often difficult to obtain in practice. The task of machine learning is then to accurately estimate the posterior probability $P(c \mid \mathbf{x})$ from the finite training samples.
- Generally speaking, there are two strategies :
  - discriminative models
    - Given $\mathbf{x}$, predict C by estimating $P(c \mid \mathbf{x})$ directly.
    - Decision trees, BP neural networks and support vector machines.
  - generative models
    - estimate the joint probability $P(\mathbf{x}, c)$ first and then estimate $P(c \mid \mathbf{x})$
    - For generative models, we must evaluate:

$$P(c \mid \mathbf{x}) = \frac{P(\mathbf{x}, c)}{P(\mathbf{x})}$$

Bayes Decision Theory

- For generative models, we must evaluate:

$$P(c \mid \mathbf{x}) = \frac{P(\mathbf{x}, c)}{P(\mathbf{x})}$$

Bayes Decision Theory

- For generative models, we must evaluate:

$$P(c \mid \mathbf{x}) = \frac{P(\mathbf{x}, c)}{P(\mathbf{x})}$$

- According to Bayes' theorem,

  $P(c \mid \mathbf{x})$ can be written as:

$$P(c \mid \mathbf{x}) = \frac{P(c)P(\mathbf{x} \mid c)}{P(\mathbf{x})}$$

Bayes Decision Theory

- For generative models, we must evaluate:

$$P(c \mid \mathbf{x}) = \frac{P(\mathbf{x}, c)}{P(\mathbf{x})}$$

- According to Bayes' theorem,

  $P(c \mid \mathbf{x})$ can be written as:

$$P(c \mid \mathbf{x}) = \frac{P(c)P(\mathbf{x} \mid c)}{P(\mathbf{x})}$$

the prior probability represents the proportion of each class in the sample, which can be estimated by the frequency of each class in the training set

Bayes Decision Theory

- For generative models, we must evaluate:

$$P(c \mid \mathbf{x}) = \frac{P(\mathbf{x}, c)}{P(\mathbf{x})}$$

- According to Bayes' theorem,

  $P(c \mid \mathbf{x})$ can be written as:

$$P(c \mid \mathbf{x}) = \frac{P(c)P(\mathbf{x} \mid c)}{P(\mathbf{x})}$$

the prior probability represents the proportion of each class in the sample, which can be estimated by the frequency of each class in the training set

the evidence factor, which is independent of the class

Bayes Decision Theory

- For generative models, we must evaluate:

$$P(c \mid \mathbf{x}) = \frac{P(\mathbf{x}, c)}{P(\mathbf{x})}$$

- According to Bayes' theorem, $P(c \mid \mathbf{x})$ can be written as:

$$P(c \mid \mathbf{x}) = \frac{P(c)P(\mathbf{x} \mid c)}{P(\mathbf{x})}$$

the class-conditional probability, also known as the likelihood, of the sample $\mathbf{x}$ with respect to class c

the prior probability represents the proportion of each class in the sample, which can be estimated by the frequency of each class in the training set

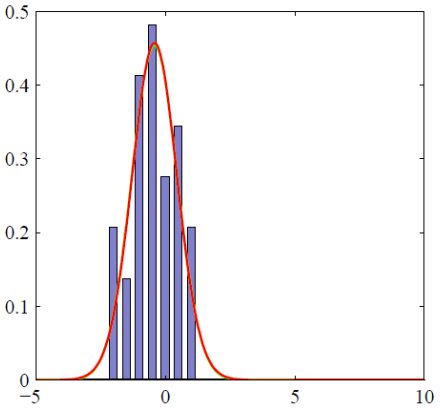
the evidence factor, which is independent of the class

Chapter List

- Bayes Decision Theory
- **Maximum Likelihood Estimation**
- Naïve Bayes Classifier
- EM Algorithm

Maximum Likelihood Estimation

- A general strategy of estimating the class-conditional probability is to hypothesize a fixed form of probability distribution, and then estimate the distribution parameters using the training samples.



- let $P(\mathbf{x} \mid c)$ denote class-condition probability of class C ,
  - suppose $P(\mathbf{x} \mid c)$ has a fixed form determined by a parameter vector $\boldsymbol{\theta}_c$. Then, the task is to estimate $\boldsymbol{\theta}_c$ from a training set D.

Maximum Likelihood Estimation

- The training process of probabilistic models is the process of parameter estimation. There are two different ways of thinking about parameters:
  - **(The Frequentist school)** Parameters have unknown but fixed values, and hence they can be determined by some approaches such as optimizing the likelihood function.
  - **(The Bayesian school)** Parameters are unobserved random variables following some distribution, and hence we can assume prior distributions for the parameters and estimate posterior distribution from observed data.

Maximum Likelihood Estimation

- Let D_c denote the set of class c samples in the training set D, and further suppose the samples are *i.i.d.* samples. Then, the likelihood of D_c for a given parameter $\boldsymbol{\theta}_c$ is:

$$P(D_c \mid \boldsymbol{\theta_c}) = \prod_{\mathbf{x} \in D_c} P(\mathbf{x} \mid \boldsymbol{\theta_c})$$

  - Applying the MLE to $\boldsymbol{\theta}_c$ is about finding a parameter value $\hat{\theta}_c$ that maximizes the likelihood $P(D_c \mid \theta_c)$. Intuitively, the MLE aims to find a value of θ_c that maximizes the "*likelihood*" that the data will present.

Maximum Likelihood Estimation

- Let D_c denote the set of class c samples in the training set D, and further suppose the samples are *i.i.d.* samples. Then, the likelihood of D_c for a given parameter $\boldsymbol{\theta}_c$ is:

$$P(D_c \mid \boldsymbol{\theta_c}) = \prod_{\mathbf{x} \in D_c} P(\mathbf{x} \mid \boldsymbol{\theta_c})$$

  - Applying the MLE to $\boldsymbol{\theta}_c$ is about finding a parameter value $\hat{\theta}_c$ that maximizes the likelihood $P(D_c \mid \theta_c)$. Intuitively, the MLE aims to find a value of θ_c that maximizes the "likelihood" that the data will present.

- Since the product of a sequence can lead to underflow, we often use the log-likelihood instead:

$$\begin{aligned} LL(\boldsymbol{\theta_c}) &= \log P(D_c \mid \boldsymbol{\theta_c}) \\ &= \sum_{\mathbf{x} \in D_c} \log P(\mathbf{x} \mid \boldsymbol{\theta}_c) \end{aligned}$$

- and the MLE of $\boldsymbol{\theta}_c$ is $\hat{\boldsymbol{\theta}}_c$:

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{c}} = \underset{\boldsymbol{\theta}_c}{\operatorname{argmax}}\, LL(\boldsymbol{\theta}_c)$$

Maximum Likelihood Estimation

- For example, suppose the features are continuous and the probability density function follows the Gaussian distribution $p(\mathbf{x} \mid c) \sim N(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2)$, then the MLE of the parameters $\boldsymbol{\mu}_c$ and $\boldsymbol{\sigma}_c^2$ are

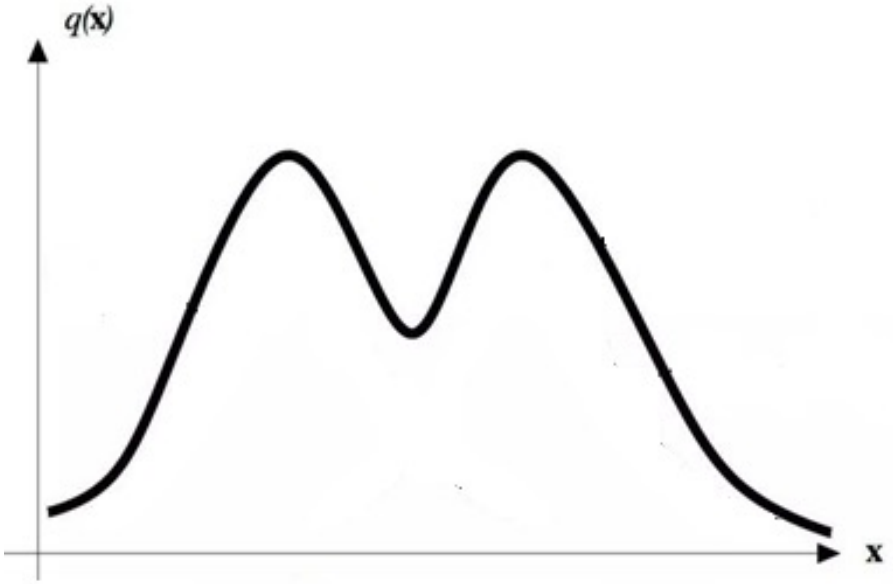
$$\hat{\boldsymbol{\mu}}_c = \frac{1}{|D_c|} \sum_{\mathbf{x} \in D_c} \mathbf{x}$$

$$\hat{\boldsymbol{\sigma}}_c^2 = \frac{1}{|D_c|} \sum_{\mathbf{x} \in D_c} (\mathbf{x} - \hat{\boldsymbol{\mu}}_c)(\mathbf{x} - \hat{\boldsymbol{\mu}}_c)^{\mathrm{T}}$$

- In other words, the estimated mean of Gaussian distribution obtained by the MLE is the sample mean, and the estimated variance is the mean of $(\mathbf{x} - \hat{\boldsymbol{\mu}}_c)(\mathbf{x} - \hat{\boldsymbol{\mu}}_c)^{\mathrm{T}}$; this conforms to our intuition.

Maximum Likelihood Estimation

- Such kind of parametric methods simplify the estimation of posterior probabilities, but the accuracy of estimation heavily relies on whether the hypothetical probability distribution matches the unknown ground-truth data distribution. In practice, a "guessed" probability distribution could incur misleading results.



Chapter List

- Bayes Decision Theory
- Maximum Likelihood Estimation
- **Naïve Bayes Classifier**
- EM Algorithm

Naïve Bayes Classifier

- The difficulty of estimating the posterior probability $P(c \mid \mathbf{x})$: it is not easy to calculate the class-conditional probability $P(\mathbf{x} \mid c)$ from the finite training samples since $P(\mathbf{x} \mid c)$ is the joint probability on all attributes.
  - For example, d binary properties $\rightarrow 2^d$ possible values, 2^d >> the number of samples
- To avoid this, the naïve Bayes classifier makes the "attribute conditional independence assumption": given any known class, assume all attributes are independent of each other.
- With the independence assumption, we have:

$$P(c \mid \mathbf{x}) = \frac{P(c)P(\mathbf{x} \mid c)}{P(\mathbf{x})} = \frac{P(c)}{P(\mathbf{x})} \prod_{i=1}^{d} P(x_i \mid c)$$

  - where d is the number of attributes.

Naïve Bayes Classifier

$$P(c \mid \mathbf{x}) = \frac{P(c)P(\mathbf{x} \mid c)}{P(\mathbf{x})} = \frac{P(c)}{P(\mathbf{x})} \prod_{i=1}^{d} P(x_i \mid c)$$

Naïve Bayes Classifier

$$P(c \mid \mathbf{x}) = \frac{P(c)P(\mathbf{x} \mid c)}{P(\mathbf{x})} = \frac{P(c)}{P(\mathbf{x})} \prod_{i=1}^{d} P(x_i \mid c)$$

Since $P(x)$ is the same for all classes, from the Bayes decision rule, we have

$$h_{nb}(\mathbf{x}) = \underset{c \in y}{\operatorname{argmax}} P(c) \prod_{i=1}^{d} P(x_i \mid c)$$

- which is the formulation of the naïve Bayes classifier.

Naïve Bayes Classifier

- To train a naïve Bayes classifier, we compute the prior probability $P(c)$ from the training set D and then compute the conditional probability $P(x_i \mid c)$ for each attribute.
  - Let D_c denote a subset of D containing all samples of class c. Then, given sufficient i.i.d. samples, the prior probability can be easily estimated by

$$P(c) = \frac{|D_c|}{|D|}$$

  - For discrete attributes, let D_{c,x_i} denote a subset of D_c containing all samples taking the value x_i on the i-th attribute. Then, the conditional probability $P(x_i \mid c)$ can be estimated by

$$P(x_i \mid c) = \frac{|D_{c,x_i}|}{|D_c|}$$

  - For continuous features, suppose $p(x_i \mid c) \sim N(\mu_{c,i}, \sigma^2_{c,i})$, where $\mu_{c,i}$ and $\sigma^2_{c,i}$ are, respectively, the mean and variance of the i-th feature of class c samples. Then, we have

$$P(x_i \mid c) = \frac{1}{\sqrt{2\pi}\sigma_{c,i}} \exp\left(-\frac{(x_i - \mu_{c,i})^2}{2{\sigma_{c,i}}^2}\right)$$

Laplace (add-1) Smoothing

- If a feature value has never appeared together with a particular class，it becomes problematic to use the probability. For instance, given a testing sample with *sound* = *crisp*, the naïve Bayes classifier trained on the watermelon data set 3.0 will predict 0. The classification result will always be *ripe* = *false* regardless of the values of other features. Such a behavior is unreasonable.

$$P_{crisp|true} = \frac{0}{8} = 0$$

ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
10	green	straight	crisp	clear	flat	soft	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

Laplace (add-1) Smoothing

- If a feature value has never appeared together with a particular class , it becomes problematic to use the probability. For instance, given a testing sample with *sound* = *crisp*, the naïve Bayes classifier trained on the watermelon data set 3.0 will predict 0. the classification result will always be *ripe* = *false* regardless of the values of other features. Such a behavior is unreasonable.
- To avoid "removing" the information carried by other features, a common choice is the Laplace smoothing.
  - let N denote the number of distinct classes in the training set D, N_i denote the number of distinct values the i-th feature can take. Then, (7.16) and (7.17) can be, respectively, smoothed as

$$\hat{P}(c) = \frac{|D_c| + 1}{|D| + N}, \qquad \hat{P}(x_i \mid c) = \frac{|D_{c,x_i}| + 1}{|D_c| + N_i}$$

Naïve Bayes Classifier

- Now, let us train a naïve Bayes classifier using the watermelon data set 3.0 classify the following watermelon T1:

ID	color	root	sound	texture	umbilicus	surface	density	sugar	ripe
T1	green	curly	muffled	clear	hollow	hard	0.697	0.460	?

Text Classification

Which class assigns the higher probability to s?

Model pos	
0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

Model neg	
0.2	I
0.001	love
0.01	this
0.005	fun
0.1	film

I	love	this	fun	film
0.1	0.1	0.01	0.05	0.1
0.2	0.001	0.01	0.005	0.1

$$P(s|pos) > P(s|neg)$$

Text Classification

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Priors:

$P(c) = ?$

$P(j) = ?$

Conditional Probabilities:

$P(Chinese|c) = ?$

$P(Tokyo|c) = ?$

$P(Japan|c) = ?$

$P(Chinese|j) = ?$

$P(Tokyo|j) = ?$

$P(Japan|j) = ?$

Choosing a class:

$P(c|d5) = ?$

$P(j|d5) = ?$

$$\hat{P}(w|c) = \frac{count(w,c)+1}{count(c)+|V|}$$

Text Classification

Priors:

$$P(c) = \frac{3+1}{4+2} = \frac{2}{3}$$
$$P(j) = \frac{1+1}{4+2} = \frac{1}{3}$$

Text Classification

Conditional Probabilities:

$P(\text{Chinese}|c) = (5+1) / (8+6) = 6/14 = 3/7$

$P(\text{Tokyo}|c) = (0+1) / (8+6) = 1/14$

$P(\text{Japan}|c) = (0+1) / (8+6) = 1/14$

$P(\text{Chinese}|j) = (1+1) / (3+6) = 2/9$

$P(\text{Tokyo}|j) = (1+1) / (3+6) = 2/9$

$P(\text{Japan}|j) = (1+1) / (3+6) = 2/9$

Text Classification

Choosing a class:

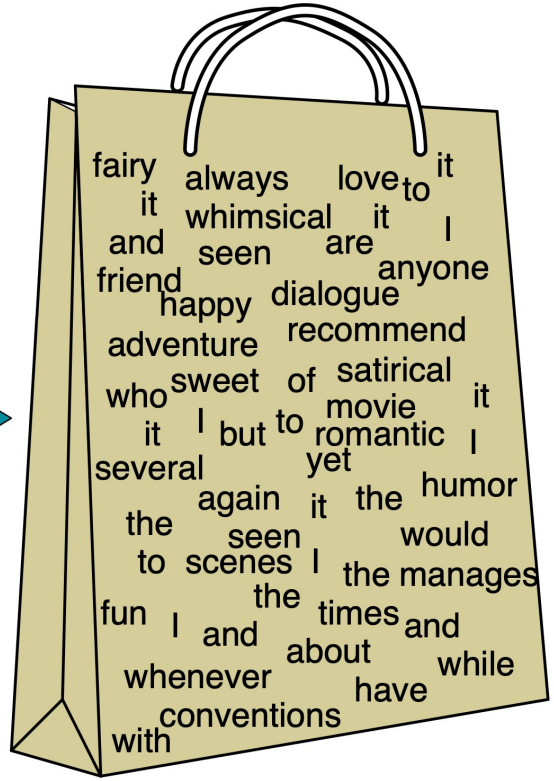
$$P(c|d5) \propto \frac{2}{3} * \left(\frac{3}{7}\right)^3 * \frac{1}{14} * \frac{1}{14} \approx 0.00027$$

$$P(j|d5) \propto \frac{1}{3} * \left(\frac{2}{9}\right)^3 * \frac{2}{9} * \frac{2}{9} \approx 0.00018$$

Text Classification

The Bag of Words Representation

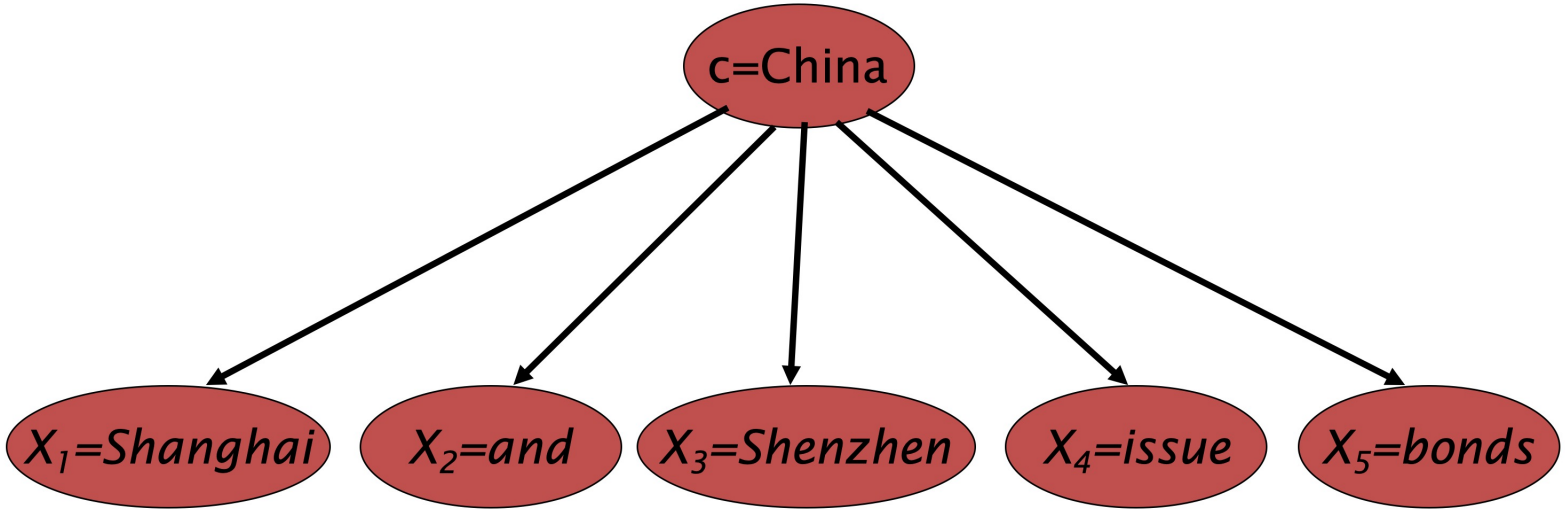
I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

Generative Model for NB

$$P(x_i \mid c)$$



Summary: Naive Bayes is Not Naive

- Very Fast, low storage requirements
- Robust to Irrelevant Features
  - Irrelevant Features cancel each other without affecting results
- Very good in domains with many equally important features
  - Decision Trees suffer from *fragmentation* in such cases – especially if little data
- Optimal if the independence assumptions hold:
  - If assumed independence is correct, then it is the Bayes Optimal Classifier
- A good dependable baseline for text classification

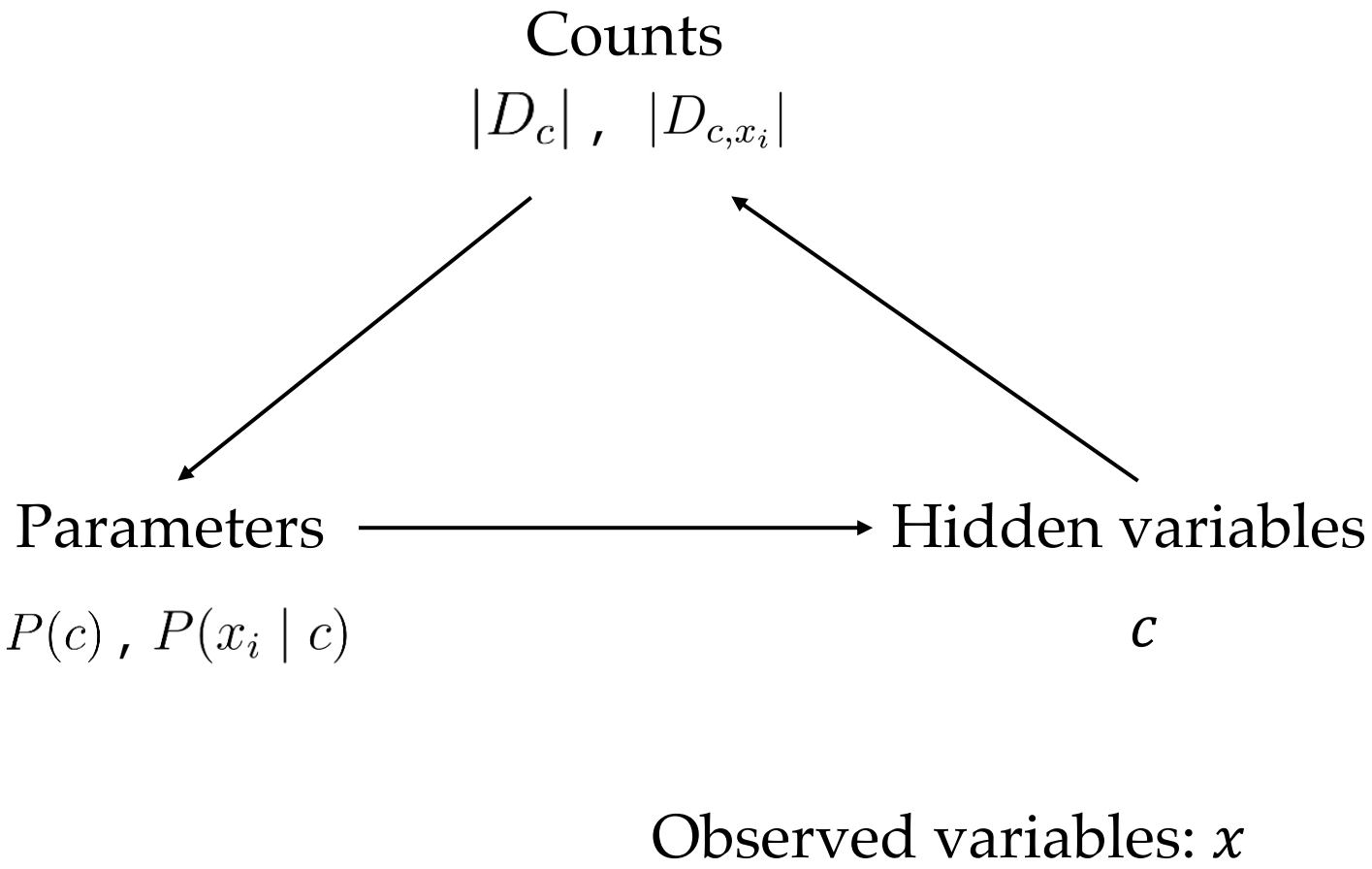
Chapter List

- Bayes Decision Theory
- Maximum Likelihood Estimation
- Naïve Bayes Classifier
- EM Algorithm

Chapter List

- Bayes Decision Theory
- Maximum Likelihood Estimation
- Naïve Bayes Classifier
- **EM Algorithm**

Dealing with Hidden Variables



EM Algorithm

- We often have *incomplete* training samples: the root of a watermelon may have been removed, that is, the feature root is unknown. Then, in the presence of unobserved features, how can we estimate the model parameters?

EM Algorithm

- We often have *incomplete* training samples: the root of a watermelon may have been removed, that is, the feature root is unknown. Then, in the presence of unobserved features, how can we estimate the model parameters?
- Let $\mathbf{X}$ denote the set of observed variables, $\mathbf{Z}$ denote the set of latent variables, and Θ denote the model parameters. Then, the maximum likelihood estimation of Θ maximizes the log-likelihood

$$LL(\boldsymbol{\Theta} \mid \mathbf{X}, \mathbf{Z}) = \ln P(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\Theta}) \qquad (7.34)$$

EM Algorithm

- We often have *incomplete* training samples: the root of a watermelon may have been removed, that is, the feature root is unknown. Then, in the presence of unobserved features, how can we estimate the model parameters?
- Let $\mathbf{X}$ denote the set of observed variables, $\mathbf{Z}$ denote the set of latent variables, and Θ denote the model parameters. Then, the maximum likelihood estimation of Θ maximizes the log-likelihood

$$LL(\boldsymbol{\Theta} \mid \mathbf{X}, \mathbf{Z}) = \ln P(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\Theta}) \qquad (7.34)$$

- We cannot solve (7.34) directly because $\mathbf{Z}$ are latent variables. However, we can use the expectation of $\mathbf{Z}$ to maximize the log marginal likelihood of the observed data:

$$LL(\boldsymbol{\Theta} \mid \mathbf{X}) = \ln P(\mathbf{X} \mid \boldsymbol{\Theta}) = \ln \sum_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\Theta}) \qquad (7.35)$$

EM Algorithm

The Expectation-Maximization (EM) algorithm [Dempster et al., 1977] is a powerful iterative method for estimating latent variables.

- Given the value of Θ, from the training data, we can infer the expected value for each latent variable in $\mathbf{Z}$ (the E-step))
- Given the values of the latent variables in $\mathbf{Z}$, we can estimate Θ with the maximum likelihood estimation (the M-step).

EM Algorithm

The Expectation-Maximization (EM) algorithm [Dempster et al., 1977] is a powerful iterative method for estimating latent variables.

- Given the value of Θ, from the training data, we can infer the expected value for each latent variable in $\mathbf{Z}$ (the E-step))
- Given the values of the latent variables in $\mathbf{Z}$, we can estimate Θ with the maximum likelihood estimation (the M-step).

We initialize (7.35) with Θ^0 and iteratively apply the following two steps until convergence: :

- Infer the expectation of $\mathbf{Z}$ with Θ^t, denoted by $\mathbf{Z}^t$;
- Estimate Θ with the maximum likelihood estimation based on $\mathbf{Z}^t$ and the observed variables $\mathbf{X}$, denoted by Θ^{t+1} ;
- The above iterative process is the prototype of the EM algorithm.

(Hard) EM Algorithm

repeat

  Expectation step:

  $\mathbf{Z}^t \leftarrow \arg\max_{\mathbf{Z}} \log P(\mathbf{X}, \mathbf{Z}|\Theta^t)$;

  Maximisation step:

  $\Theta^{t+1} \leftarrow \arg\max_{\Theta} \log P(\mathbf{X}, \mathbf{Z}^t|\Theta)$;

  $t \leftarrow t + 1$;

until CONVERGE$(\mathbf{Z}, \Theta)$;

(Soft) EM Algorithm

Instead of using the expectation of $\mathbf{Z}$, we can compute the probability distribution $P(\mathbf{Z} \mid \mathbf{X}, \Theta^t)$ based on Θ^t, and the above two steps become:

- E-step (Expectation): infer the latent variable distribution $P(\mathbf{Z} \mid \mathbf{X}, \Theta^t)$ based on Θ^t, and then compute the expectation of the log-likelihood $LL(\Theta \mid \mathbf{X}, \mathbf{Z})$ with respect to $\mathbf{Z}$ as:

$$Q(\boldsymbol{\Theta} \mid \boldsymbol{\Theta}^t) = \mathbb{E}_{\mathbf{Z}|\mathbf{X},\boldsymbol{\Theta}^t} LL(\boldsymbol{\Theta} \mid \mathbf{X}, \mathbf{Z})$$

- M-step (Maximization): find the parameters that maximize the expected log-likelihood, that is,

$$\boldsymbol{\Theta}^{\boldsymbol{t+1}} = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}}\, Q(\boldsymbol{\Theta} \mid \boldsymbol{\Theta}^t)$$

- EM algorithm finds the parameters that maximize the expected log-likelihood in the E-step. Iterating the above two steps until converging to a local optimum.

Relationships between MLE and Q-function

- When the outputs are hidden variables, and if z is known, we can turn EM algorithm to MLE in supervised settings.
  - supposed that each $\boldsymbol{x}_i$ has a supervised label y_i
  - defining

$$P\left(z \mid \boldsymbol{x}_i, \Theta^t\right) = \begin{cases} 1 \text{ if } z = y_i \\ 0 \text{ otherwise} \end{cases}$$

$$\begin{aligned} Q\left(\Theta \mid \Theta^t\right) &= \sum_{i=1}^{N} \sum_{\boldsymbol{z} \in Z} P\left(\boldsymbol{z} \mid \boldsymbol{x}_i, \Theta^t\right) \log P(\boldsymbol{x}_i, \boldsymbol{z} \mid \Theta) \\ &= \sum_{i=1}^{N} \log P(\boldsymbol{x}_i, y_i \mid \Theta) \end{aligned}$$

which is exactly the maximum log-likelihood training objective.

Graphical interpretation

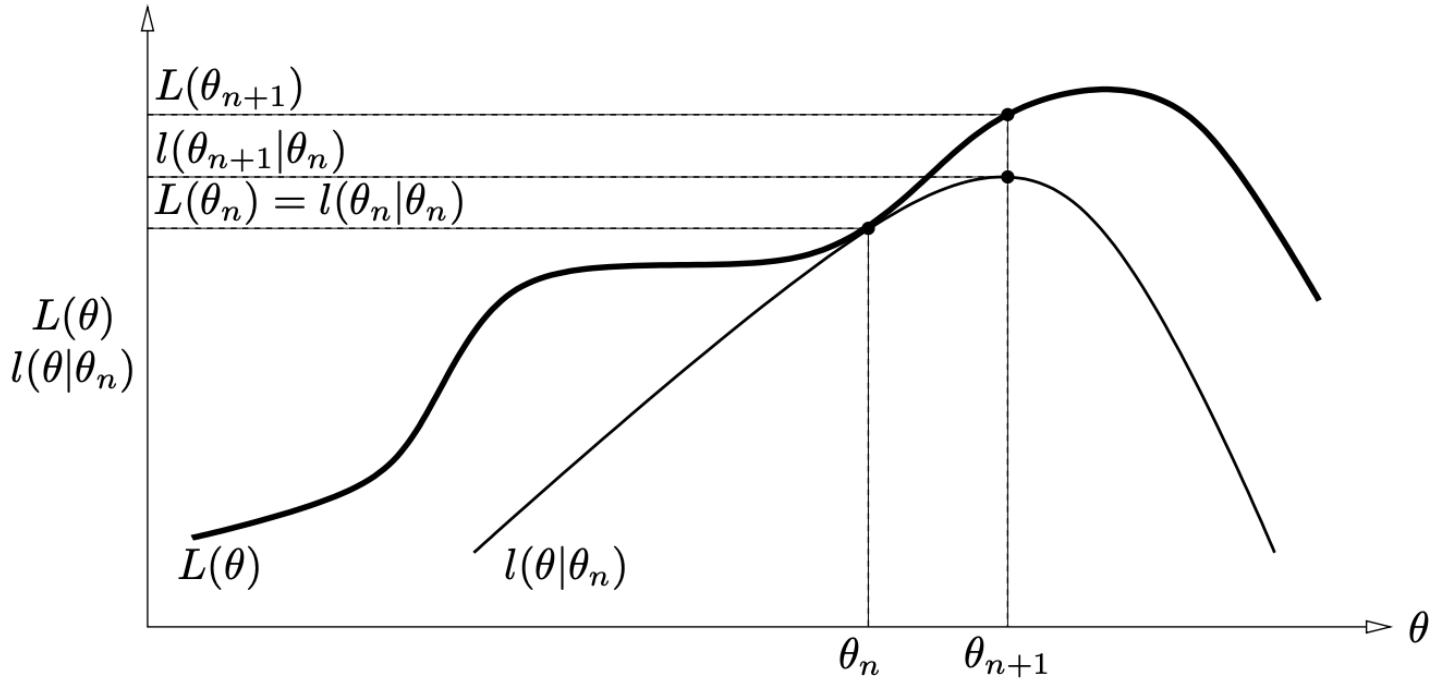


Figure 2: Graphical interpretation of a single iteration of the EM algorithm: The function $l(\theta|\theta_n)$ is upper-bounded by the likelihood function $L(\theta)$. The functions are equal at $\theta = \theta_n$. The EM algorithm chooses θ_{n+1} as the value of θ for which $l(\theta|\theta_n)$ is a maximum. Since $L(\theta) \geq l(\theta|\theta_n)$ increasing $l(\theta|\theta_n)$ ensures that the value of the likelihood function $L(\theta)$ is increased at each step.

Convergence

EM is guaranteed to converge to a point with **zero gradient**.

$$
\begin{aligned}
L(\theta) - L(\theta_n) &= \ln \sum_{\mathbf{z}} \mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta) - \ln \mathcal{P}(\mathbf{X}|\theta_n) \\
&= \ln \sum_{\mathbf{z}} \mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta) \cdot \frac{\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)}{\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)} - \ln \mathcal{P}(\mathbf{X}|\theta_n) \\
&= \ln \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \left(\frac{\mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta)}{\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)} \right) - \ln \mathcal{P}(\mathbf{X}|\theta_n) \\
&\geq \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln \left(\frac{\mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta)}{\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)} \right) - \ln \mathcal{P}(\mathbf{X}|\theta_n) \\
&= \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln \left(\frac{\mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta)}{\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)\mathcal{P}(\mathbf{X}|\theta_n)} \right) \\
&\triangleq \Delta(\theta|\theta_n)
\end{aligned}
$$

WHY? 

Convergence

Theorem 2 (Jensen's inequality) *Let f be a convex function defined on an interval I. If $x_1, x_2, \ldots, x_n \in I$ and $\lambda_1, \lambda_2, \ldots, \lambda_n \geq 0$ with $\sum_{i=1}^{n} \lambda_i = 1$,*

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) \leq \sum_{i=1}^{n} \lambda_i f(x_i)$$

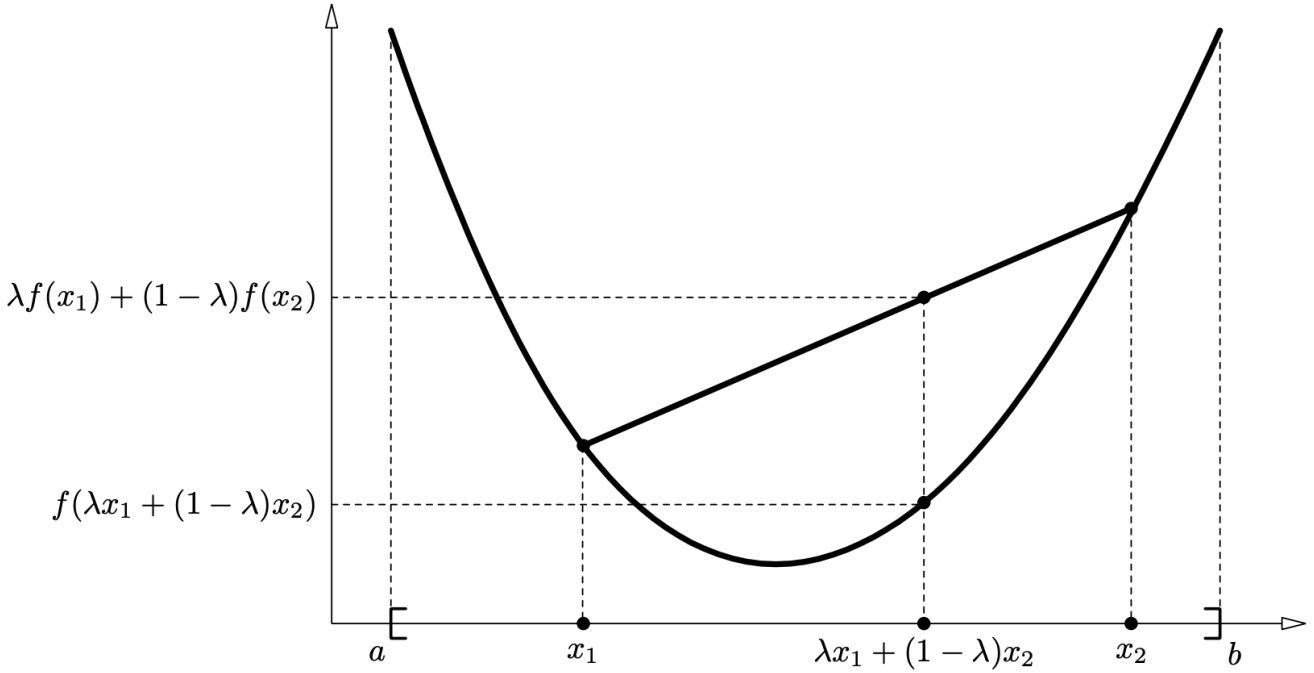


Figure 1: f is *convex* on $[a, b]$ if $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$ $\forall x_1, x_2 \in [a, b], \quad \lambda \in [0, 1]$.

Convergence

Equivalently we may write,

$$L(\theta) \geq L(\theta_n) + \Delta(\theta|\theta_n)$$

and for convenience define,

$$l(\theta|\theta_n) \triangleq L(\theta_n) + \Delta(\theta|\theta_n)$$

so that

$$L(\theta) \geq l(\theta|\theta_n).$$

Additionally, observe that,

$$\begin{aligned}
l(\theta_n|\theta_n) &= L(\theta_n) + \Delta(\theta_n|\theta_n) \\
&= L(\theta_n) + \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X}, \theta_n) \ln \frac{\mathcal{P}(\mathbf{X}|\mathbf{z}, \theta_n)\mathcal{P}(\mathbf{z}|\theta_n)}{\mathcal{P}(\mathbf{z}|\mathbf{X}, \theta_n)\mathcal{P}(\mathbf{X}|\theta_n)} \\
&= L(\theta_n) + \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X}, \theta_n) \ln \frac{\mathcal{P}(\mathbf{X}, \mathbf{z}|\theta_n)}{\mathcal{P}(\mathbf{X}, \mathbf{z}|\theta_n)} \\
&= L(\theta_n) + \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X}, \theta_n) \ln 1 \\
&= L(\theta_n),
\end{aligned}$$

Convergence

$$
\begin{aligned}
\theta_{n+1} &= \arg\max_{\theta} \{l(\theta|\theta_n)\} \\
&= \arg\max_{\theta} \left\{ L(\theta_n) + \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln \frac{\mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta)}{\mathcal{P}(\mathbf{X}|\theta_n)\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)} \right\} \\
&\quad \text{Now drop terms which are constant w.r.t. } \theta \\
&= \arg\max_{\theta} \left\{ \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln \mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta) \right\} \\
&= \arg\max_{\theta} \left\{ \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln \mathcal{P}(\mathbf{X},\mathbf{z}|\theta) \right\} \\
&= \arg\max_{\theta} \left\{ \mathrm{E}_{\mathbf{Z}|\mathbf{X},\theta_n} \left\{ \ln \mathcal{P}(\mathbf{X},\mathbf{z}|\theta) \right\} \right\}
\end{aligned}
$$

$$
Q(\boldsymbol{\Theta} \mid \boldsymbol{\Theta}^t) = \mathbb{E}_{\mathbf{Z}|\mathbf{X},\boldsymbol{\Theta}^t} LL(\boldsymbol{\Theta} \mid \mathbf{X}, \mathbf{Z})
$$

Graphical interpretation

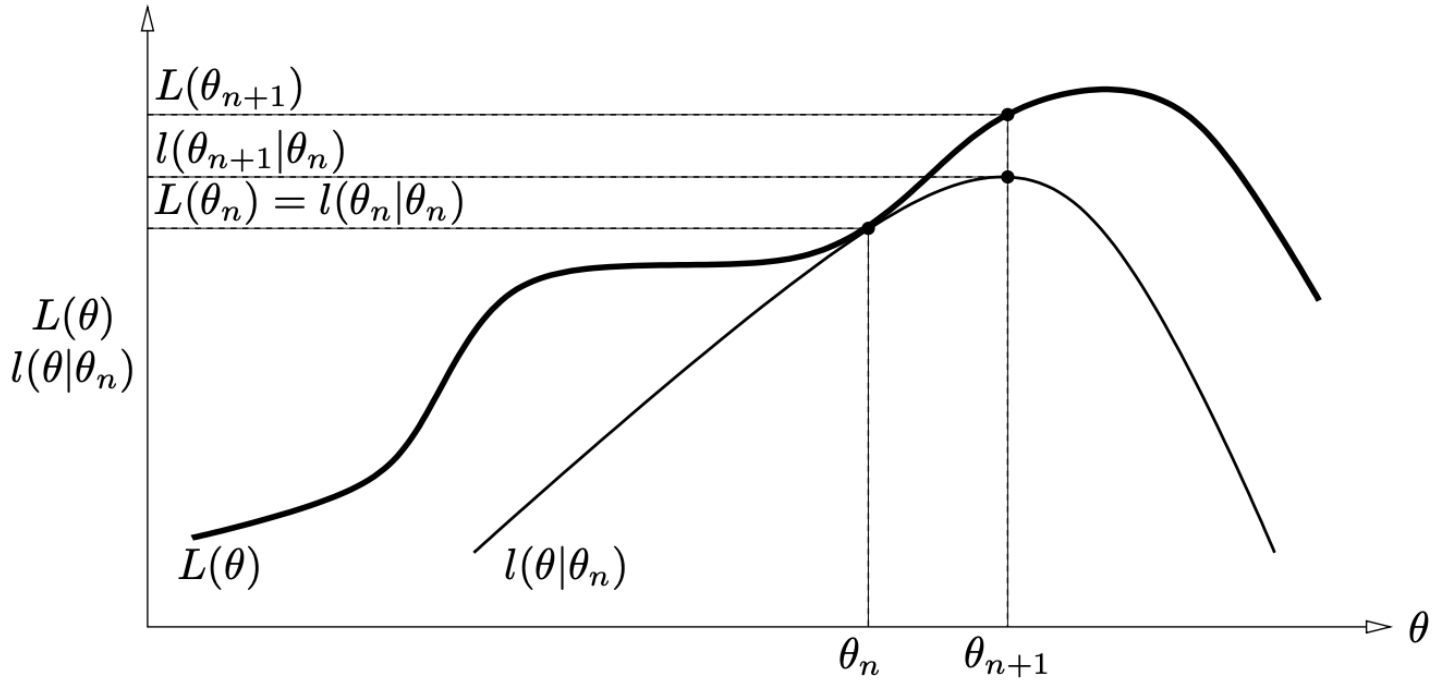
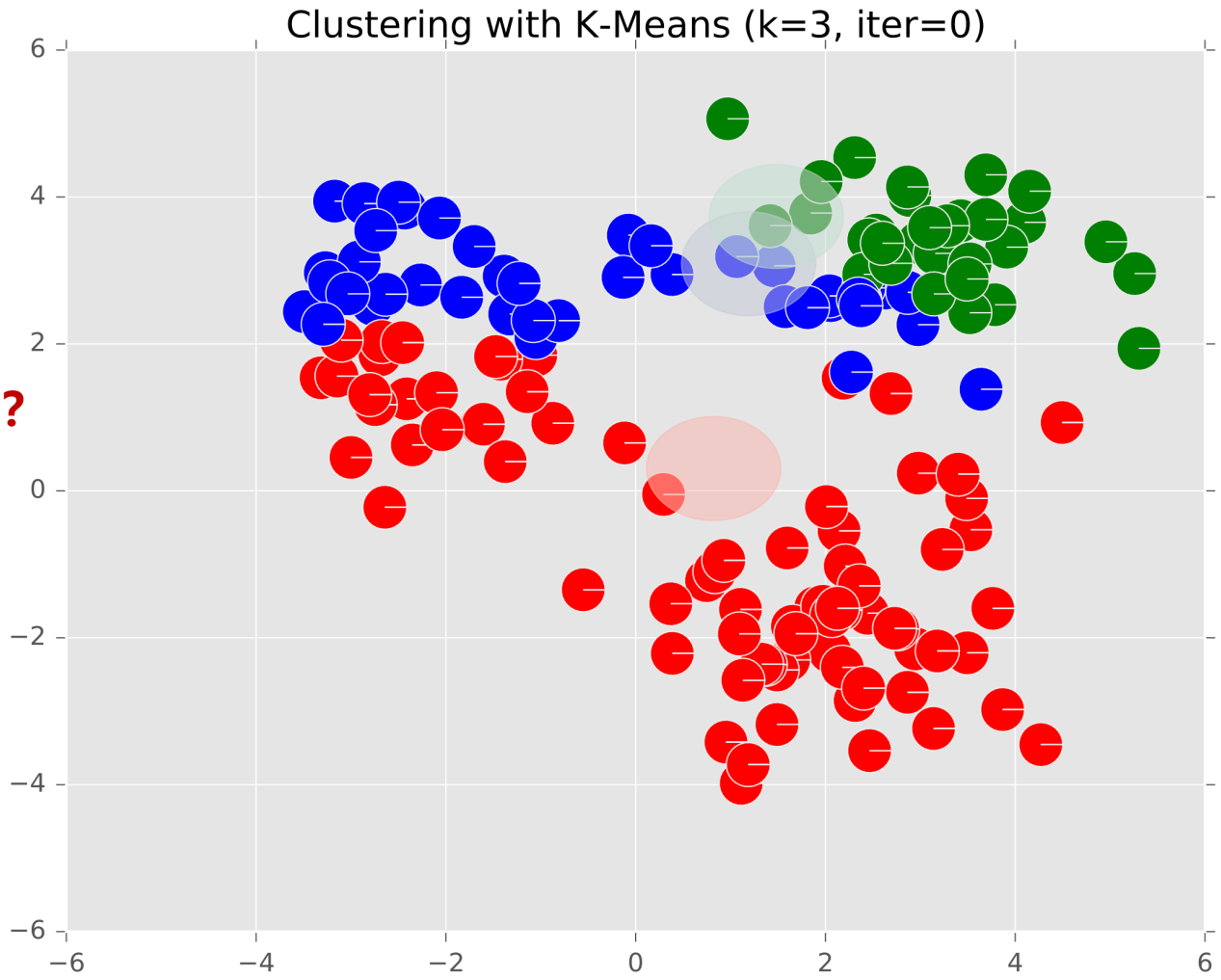


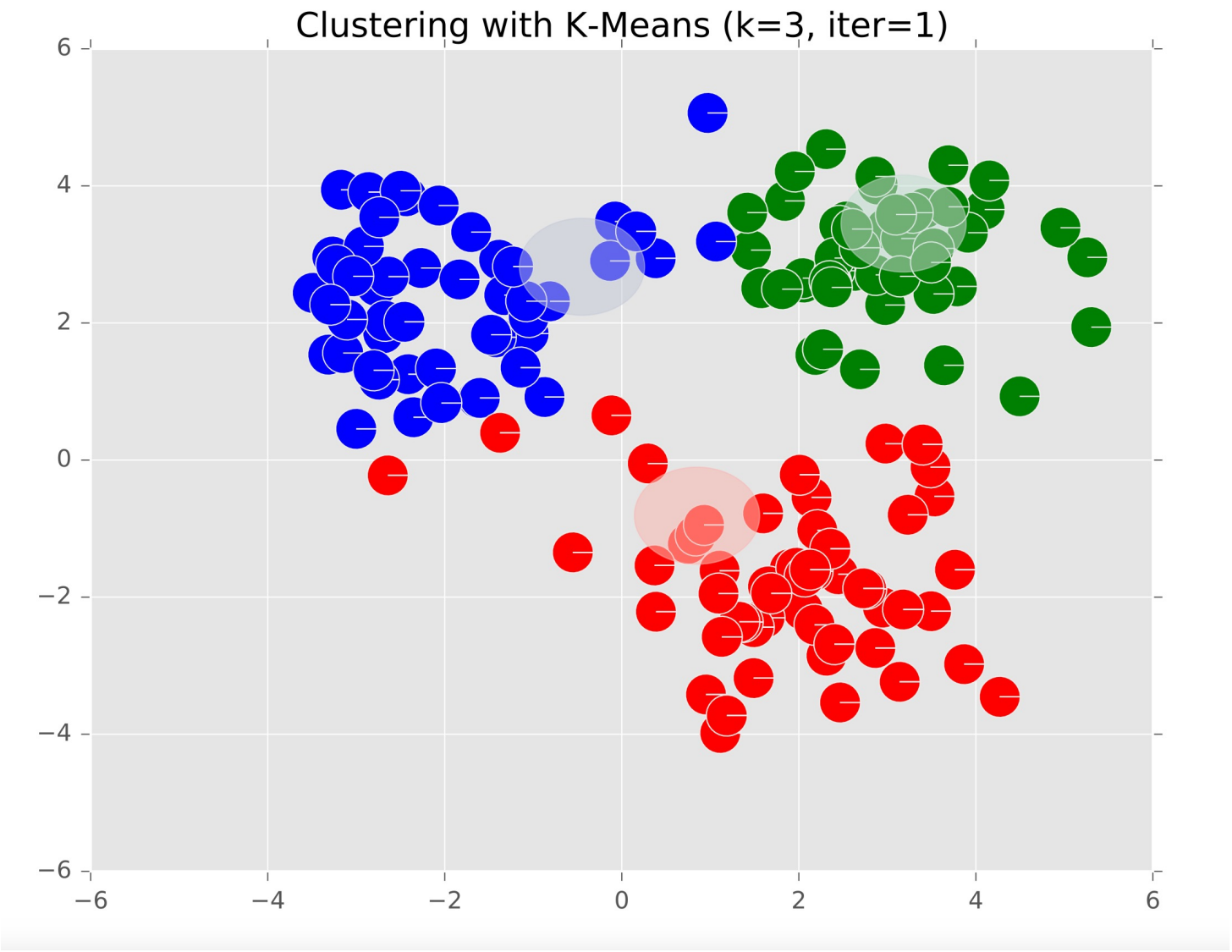
Figure 2: Graphical interpretation of a single iteration of the EM algorithm: The function $l(\theta|\theta_n)$ is upper-bounded by the likelihood function $L(\theta)$. The functions are equal at $\theta = \theta_n$. The EM algorithm chooses θ_{n+1} as the value of θ for which $l(\theta|\theta_n)$ is a maximum. Since $L(\theta) \geq l(\theta|\theta_n)$ increasing $l(\theta|\theta_n)$ ensures that the value of the likelihood function $L(\theta)$ is increased at each step.

Example: K-Means

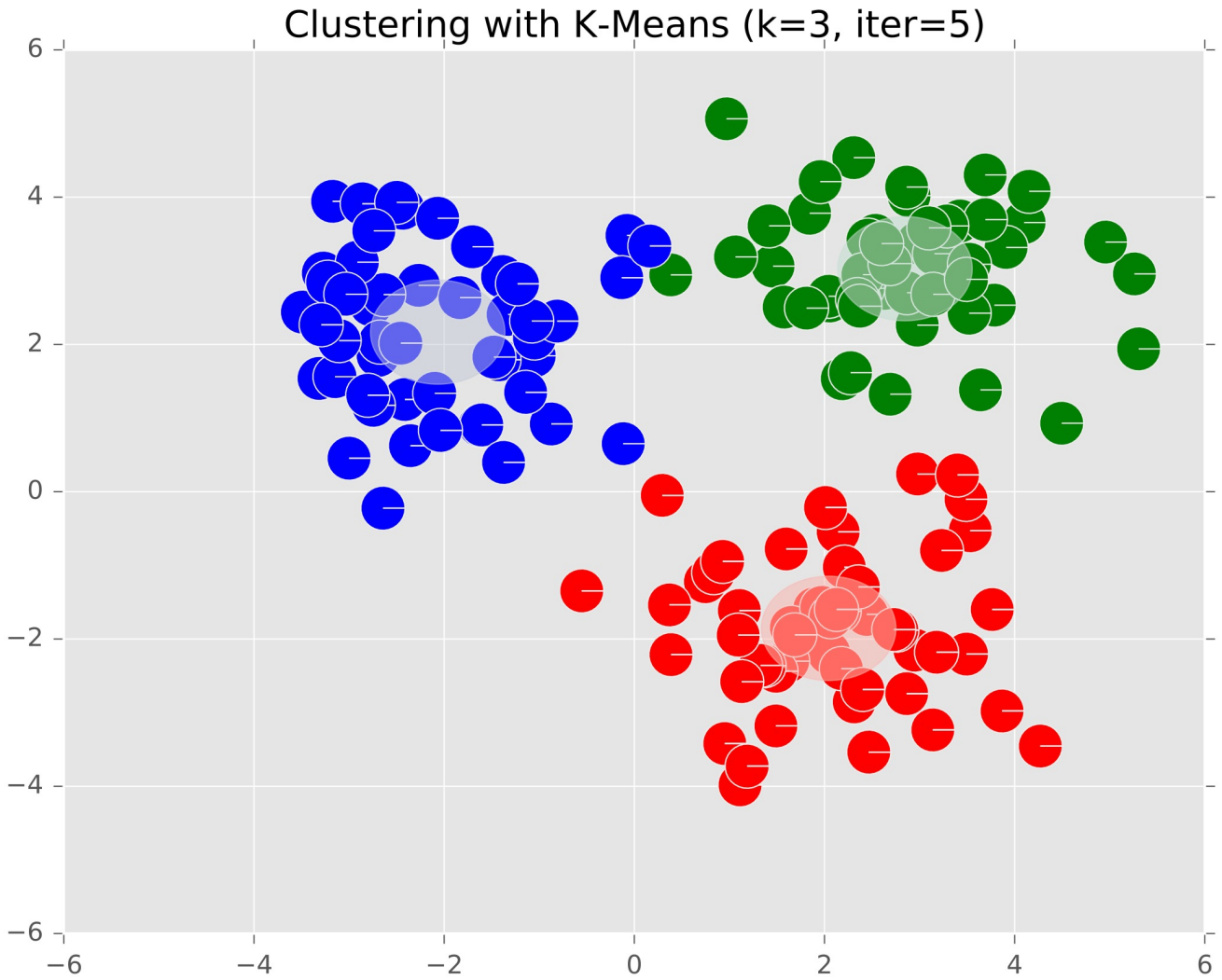
Hidden Variable ?



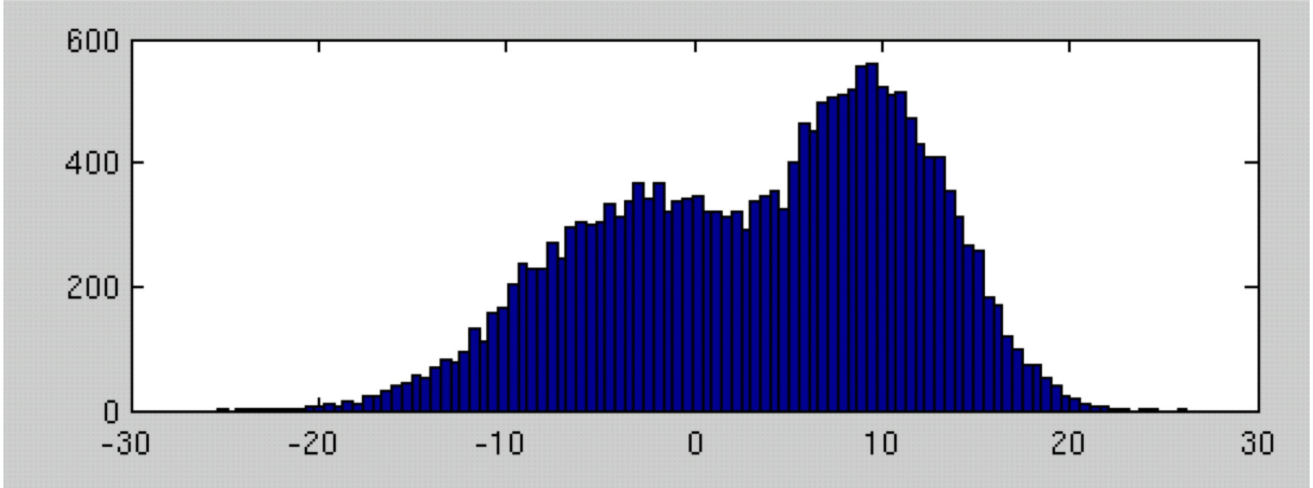
Example: K-Means

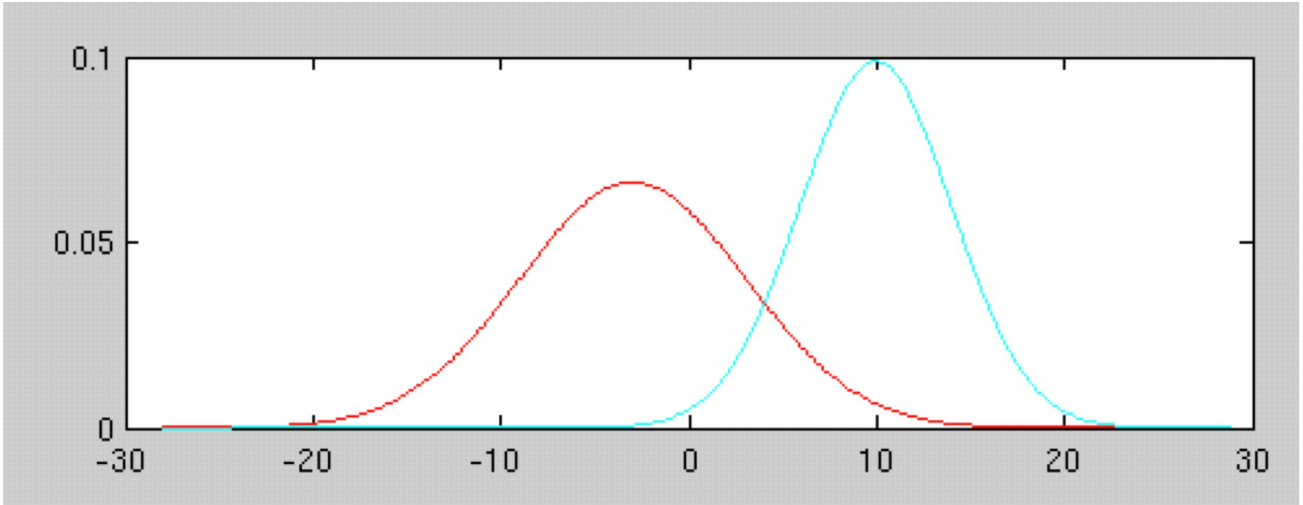


Example: K-Means



Gaussian Mixture-Model

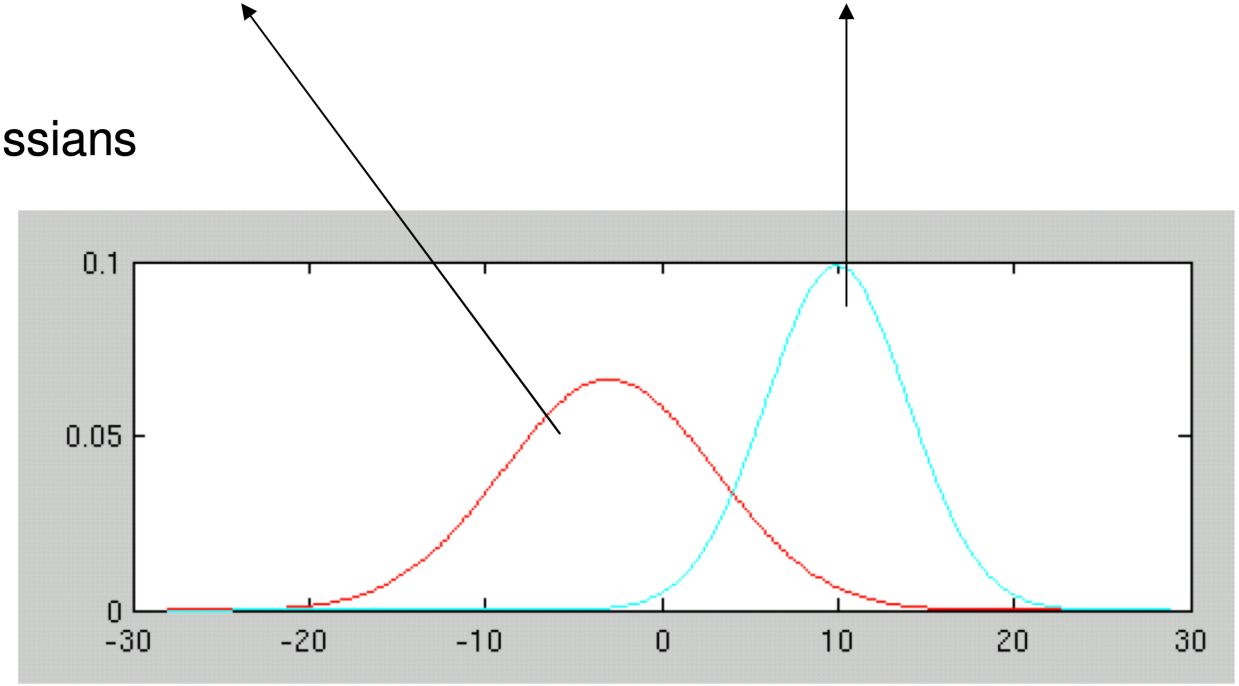




Gaussian Mixture-Model

$$\mathrm{P(x)} = \pi\mathcal{N}(x|\mu_1, \sum_1) + (1-\pi)\mathcal{N}(x|\mu_2, \sum_2)$$

GMM with 2 gaussians



Hidden Variable ?

PDF of Guassian: $\frac{1}{(2\pi)^{d/2}\,|\Sigma_j|^{1/2}} e^{-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)}$

Gaussian Mixture-Model

Data: $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$

Generative Story:

$$z \sim \text{Categorical}(\boldsymbol{\phi})$$

$$\mathbf{x} \sim \text{Gaussian}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$$

Model:

Joint: $p(\mathbf{x}, z; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\mathbf{x}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \boldsymbol{\phi})$

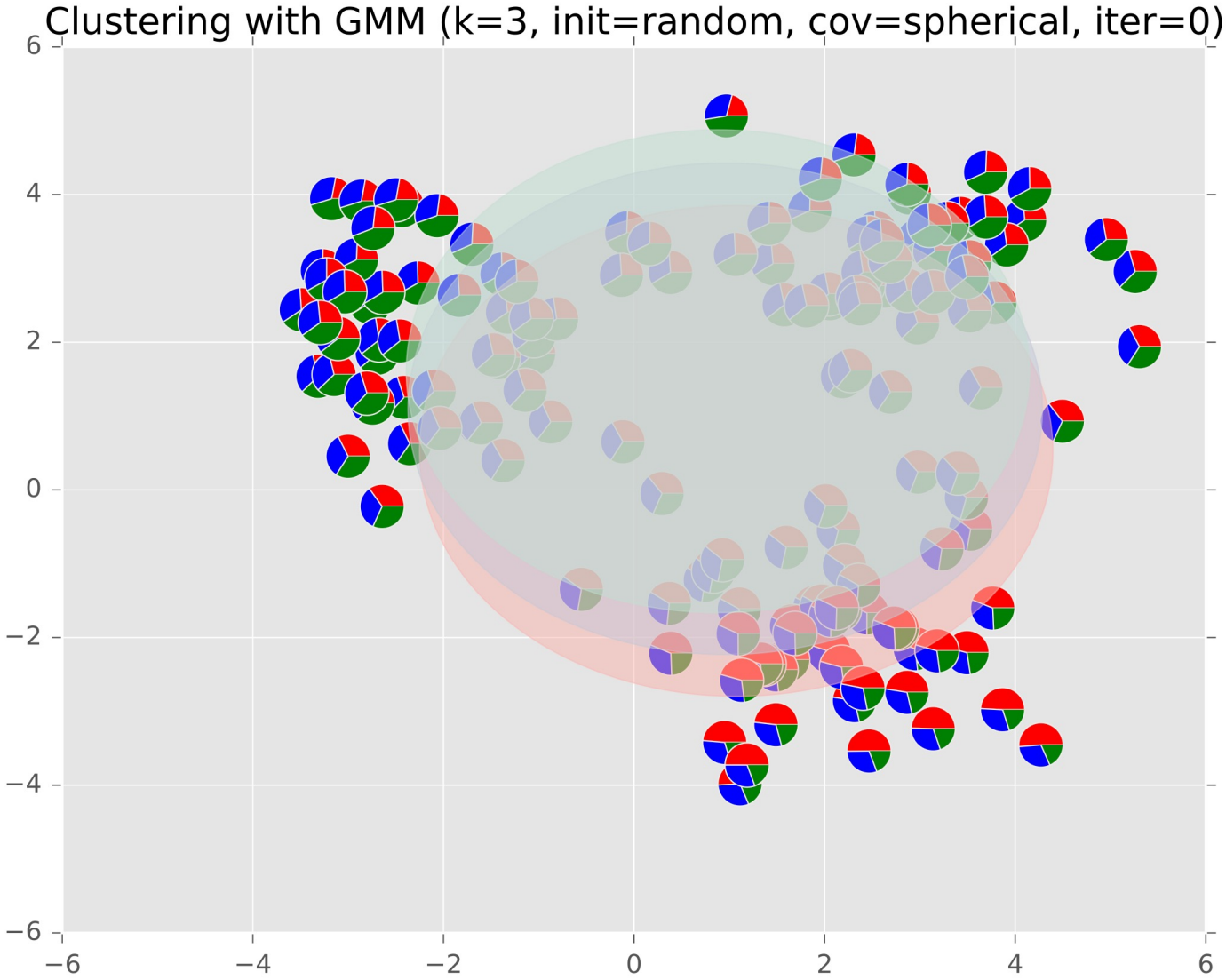
Marginal: $p(\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{z=1}^{K} p(\mathbf{x}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \boldsymbol{\phi})$

(Marginal) Log-likelihood:

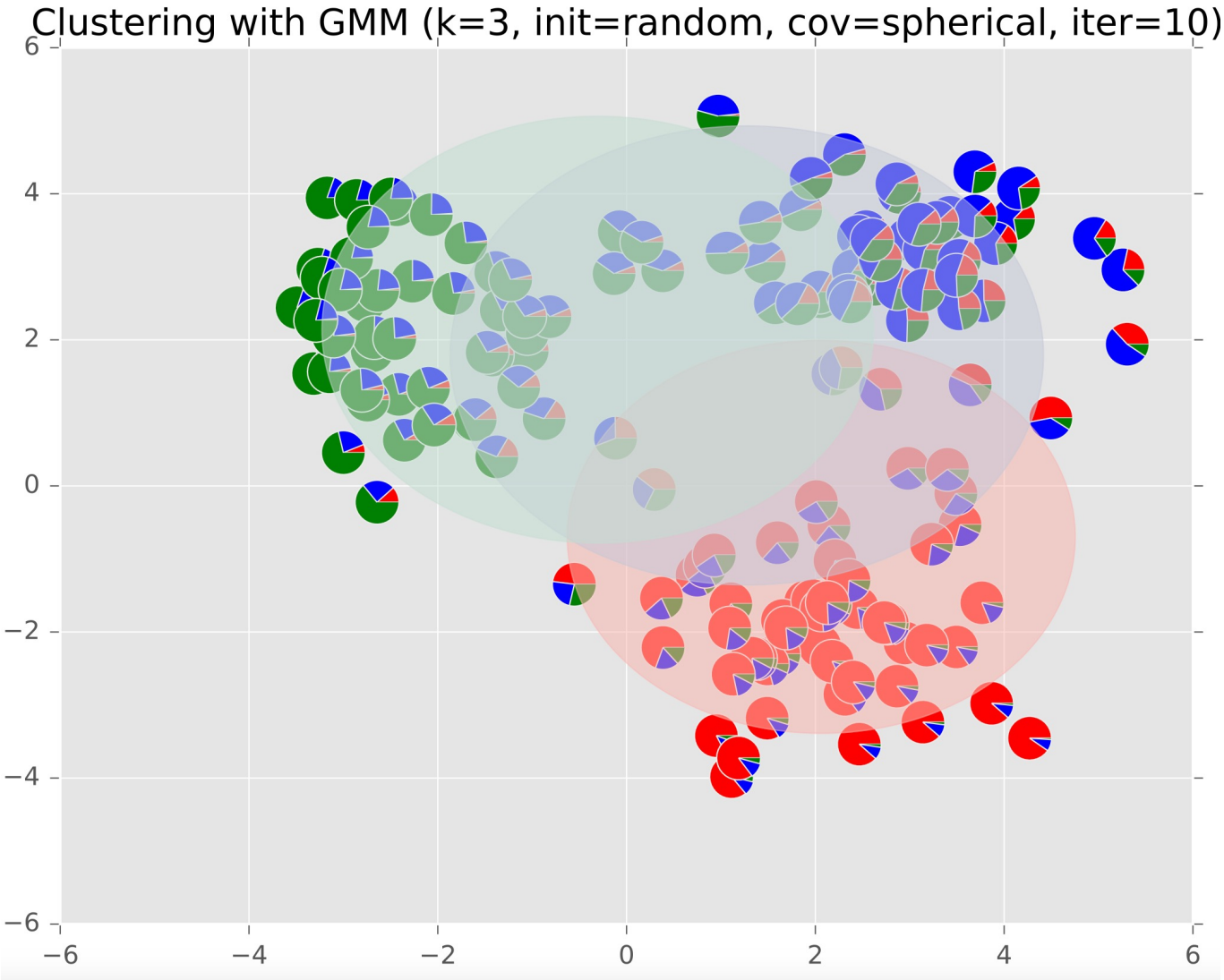
$$\ell(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log \prod_{i=1}^{N} p(\mathbf{x}^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$= \sum_{i=1}^{N} \log \sum_{z=1}^{K} p(\mathbf{x}^{(i)}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \boldsymbol{\phi})$$

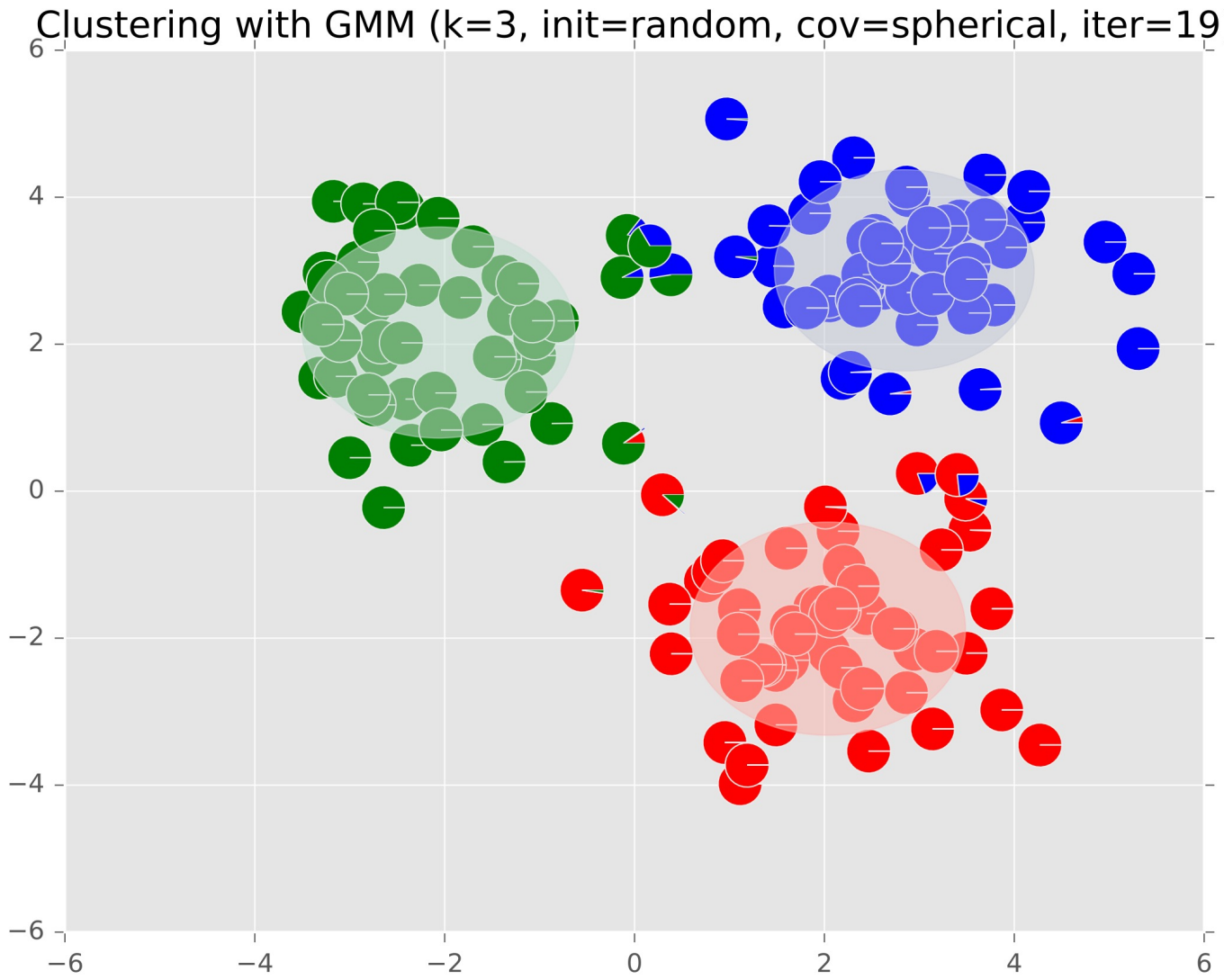
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=0)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=10)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=19)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6