GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

E016350 - Artificial Intelligence

Lecture 11

Reasoning under Uncertainty & Bayesian ML

Learning probabilistic models

Aleksandra Pizurica

Ghent University
Fall 2024

Overview

- Bayesian machine learning
- MAP learning
- Maximum-likelihood parameter learning
- Naive Bayes classifier
- Bayesian parameter learning
- Clustering by Learning mixtures of Gaussians

[R&N], Chapter 20

This presentation is partly based on: S. Russel and P. Norvig: *Artificial Intelligence: A Modern Approach*, Fourth Ed.), denoted as [R&N] and the resource page http://aima.cs.berkeley.edu/

Bayesian view on machine learning

- View learning as a form of uncertain reasoning from observations
  - Learning task as probabilistic inference
- Devise models to represent uncertain world
- Bayesian view of learning is very powerful
  - General solutions to noise, overfitting and optimal prediction
  - Don't necessarily choose one single hypothesis but take each with its probability
- Meets real-life challenges: AI agents are not omniscient
  - Not certain about which model is correct, yet must decide/act

Statistical learning

- The same key concepts from the theory of learning: data and hypotheses
  but we deal with random variables (r.v.s)
- Now data are **evidence**
  instantiations of some (or all) domain r.v.s
- Hypotheses are now **probabilistic theories**
  of how the domain "works"

Surprise Candy case

Our favorite surprise candy comes in two flavors: cherry and lime. It's wrapped in the same opaque wrapper, regardless of flavor, and sold in very large bags, of which there are known to be five kinds – again, indistinguishable from the outside:

- h_1: 100% cherry
- h_2: 75% cherry + 25% lime
- h_3: 50% cherry + 50% lime
- h_4: 25% cherry + 75% lime
- h_5: 100% lime

- Random variables:
  - H: type of the bag; possible values $h \in \{h_1, \dots h_5\}$
  - $D^{(i)}$: data revealed when i-th candy opened; $d^{(i)} \in \{cherry, lime\}$
- Task faced by the AI agent: predict the flavor of the next piece of candy

Bayesian learning - basic concepts

- Calculate the probability of each hypothesis, given the data
- Make predictions using all the hypotheses weighted by their probabilities
- Learning becomes probabilistic inference!

Let $\mathbf{D}$ be all the data, with observed value $\mathbf{d}$

$$\underbrace{P(h_j \mid \mathbf{d})}_{\substack{\text{posterior prob.} \\ \text{of hypotheis}}} = \alpha \underbrace{P(\mathbf{d} \mid h_j)}_{\text{likelihood}} \underbrace{P(h_j)}_{\substack{\text{hypothesis} \\ \text{prior}}}$$

Prediction about unknown X:

$$\mathbf{P}(X|\mathbf{d}) = \sum_j \mathbf{P}(X|h_j) P(h_j|\mathbf{d})$$

Bayesian learning on the *Surprise Candy* case

We want to predict the flavor of the $(N+1)$-th candy given the N opened ones:

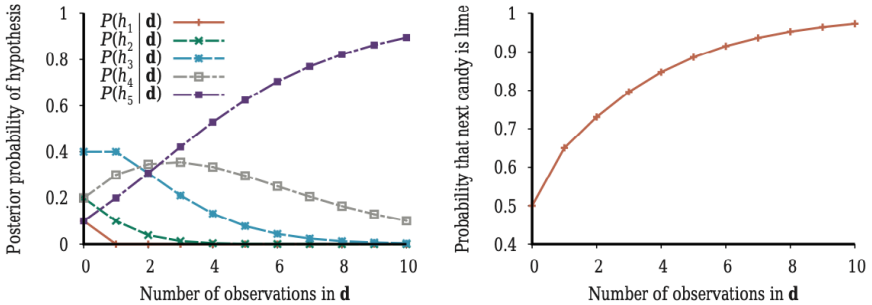
$$\mathbf{P}(D^{(N+1)}|\mathbf{d}) = \sum_j \mathbf{P}(D^{(N+1)}|h_j)P(h_j|\mathbf{d})$$

Assume the data $\mathbf{d} = \{d^{(1)}, \ldots, d^{(N)}\}$ are **i.i.d.** Then it holds

$$P(\mathbf{d}|h_j) = \prod_i P(d^{(i)}|h_j)$$

We still need the prior probabilities of h_j's. Suppose the prior distribution is given (e.g., as advertised by manufacturer): $\mathbf{P}(h_1, \ldots, h_5) = \langle 0.1, 0.2, 0.4, 0.2, 0.1 \rangle$

Bayesian learning on the *Surprise Candy* case



Left: Posterior probabilities $P(h_j|d^{(1)}, \ldots d^{(N)})$.
Right: Bayesian prediction $P(D^{(N+1)} = lime|d^{(1)}, \ldots, d^{(N)})$.

Calculated for the case where N ranges from 1 to 10, and each observation is $lime$.

Maximum a Posteriori and Maximum Likelihood Estimates

- The hypothesis space is usually very large
  $\rightarrow$ Bayesian learning may be intractable.
- Solution: resort to approximate or simplified methods
  - Make prediction based on a single most probable hypothesis
    - ⋆ **Maximum a Posteriori** (**MAP**) hypothesis:

$$h_{MAP} = \arg\max_{h \in \mathcal{H}} P(h|\mathbf{d})$$

Predictions made by MAP approximate Bayesian ML to the extent that

$$P(X|\mathbf{d}) \approx P(X|h_{MAP})$$

  - If we assume uniform prior over the hypothesis space
    - ⋆ **Maximum-Likelihood** hypothesis

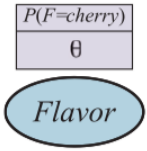
$$h_{ML} = \arg\max_{h \in \mathcal{H}} P(\mathbf{d}|h)$$

Learning with complete data

- **Density estimation** – learning a probability model, given data
  that are assumed to be generated by that model
- It is a form of unsupervised learning
- We focus on parameter learning
  – finding the parameter values of a probability model whose structure is fixed
- We start from the simplest case: learning with complete data
  (each data point contains values for every variable in the model being learned)

Maximum-likelihood learning: Discrete data (Example 1)

- Let the fraction of cherry be a parameter $\theta \in [0, 1]$
- Hypotheses are now h_θ
- Assume all proportions are equally likely *a priori*
- Model the situation with a Bayesian network



- We need only one r.v.: $Flavor$ with $P(Flavor = cherry) = \theta$

Suppose we unwrap N candies of which c are cherry and $l = N - c$ are lime

$$P(\mathbf{d}|h_\theta) = \prod_{i=1}^{N} P(d^{(i)}|h_\theta) = \theta^c(1-\theta)^l$$

To get h_{ML}, let $\ell(\theta) = \log P(\mathbf{d}|h_\theta) = \sum_{i=1}^{N} \log P(d^{(i)}|h_\theta) = c \log \theta + l \log(1-\theta)$
From $\frac{d\ell(\theta)}{d\theta} = 0 \implies \theta = \frac{c}{c+l} = \frac{c}{N}$. Hence, $h_{ML} = h_{c/N}$, i.e., $\hat{\theta}_{ML} = \frac{c}{N}$

ML asserts that the actual proportion of cherry is the same as the observed proportion

Maximum-likelihood learning: Discrete data

In general, maximum-likelihood learning with $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_M\}$,

1. Write down an expression for $\ell(\boldsymbol{\theta}) = \log P(\mathbf{d}|h_{\boldsymbol{\theta}})$
2. Write down the derivatives $\frac{\partial \ell(\boldsymbol{\theta})}{\partial \theta_k},\ k = 1, \ldots, M$
3. Find θ_k such that $\frac{\partial \ell(\boldsymbol{\theta})}{\partial \theta_k} = 0,\ k = 1, \ldots, M$

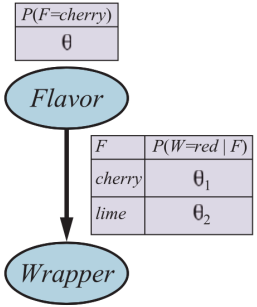
The last step often requires iterative optimization

Key problem: when the data set is small such that some events have not yet been observed (e.g., no cherry candy's yet), h_{ML} assigns zero probabilities to those events

Tricks to avoid this include different initializations

Maximum-likelihood learning: Discrete data (Example 2)

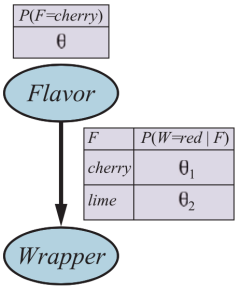
- Let the fraction of cherry be a parameter $\theta \in [0, 1]$
- New: two wrapper colors $red, green$
- Wrapper for each candy selected according to some unknown distribution $\mathbf{P}(Wrapper|Flavor)$
- Model the situation with a Bayesian network
- We have 2 r.v.s and 3 parameters: θ, θ_1 and θ_2



E.g., $P(F = cherry, W = green|h_{\theta,\theta_1,\theta_2})$
$= P(F = cherry|h_{\theta,\theta_1,\theta_2})P(W = green|F = cherry, h_{\theta,\theta_1,\theta_2}) = \theta(1 - \theta_1)$

Maximum-likelihood learning: Discrete data (Example 2, contd.)

- We unwrap N candy's, c are cherry and l are lime
- r_c of the cherry candy's have red wrappers and g_c green
- r_l of the lime candy's have red wrappers and g_l green



The likelihood of the data is

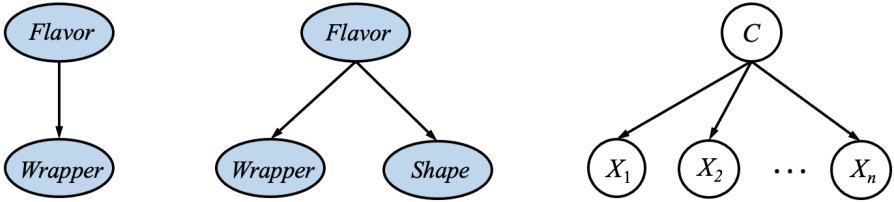
$$P(\mathbf{d}|h_{\theta,\theta_1,\theta_2}) = \theta^c(1-\theta)^l \cdot \theta_1^{r_c}(1-\theta_1)^{g_c} \cdot \theta_2^{r_l}(1-\theta_2)^{g_l}$$

Setting the partial derivatives of the log-likelihood $\ell(\theta,\theta_1,\theta_2) = \log P(\mathbf{d}|h_{\theta,\theta_1,\theta_2})$ to zero yields

$$\theta = \frac{c}{c+l}, \quad \theta_1 = \frac{r_c}{r_c+g_c}, \quad \theta_2 = \frac{r_l}{r_l+g_l}$$

Note: With complete data, the maximum-likelihood parameter learning problem for a Bayesian network decomposes into separate learning problems, one for each parameter

Naive Bayes models



The class variable C is the root (to be predicted), and X_i are the attributes (features)

$$\mathbf{P}(C|x_1,\ldots,x_n) = \alpha\mathbf{P}(C)\prod_j \mathbf{P}(x_j|C)$$

In the case where all r.v.s are Boolean:

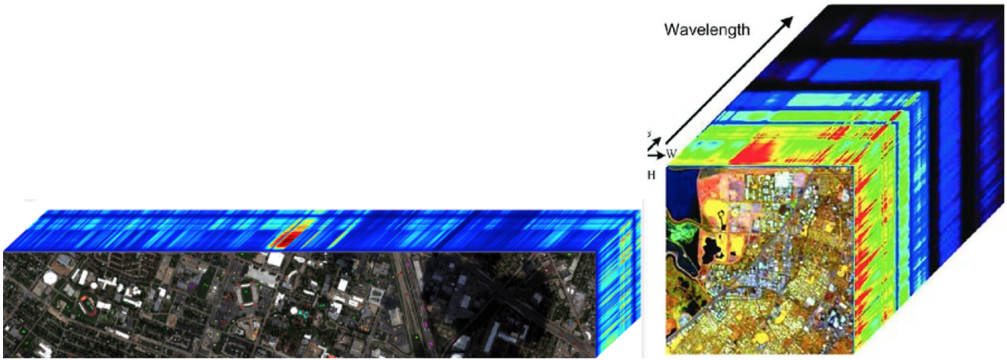
$$\theta = P(C=1), \quad \theta_{j1} = P(X_j=1|C=1), \quad \theta_{j2} = P(X_j=1|C=0)$$

Let $(\mathbf{x}^{(i)}, c^{(i)})$ be ith data point. $\theta_{jk} = \frac{\sum_i \mathbb{1}[x_j^{(i)}=1 \wedge c^{(i)}=k]}{\sum_i \mathbb{1}[c^{(i)}=k]}$ e.g., $\frac{\#[W=red \wedge F=cherry]}{\#[F=cherry]}$

Naive Bayes models

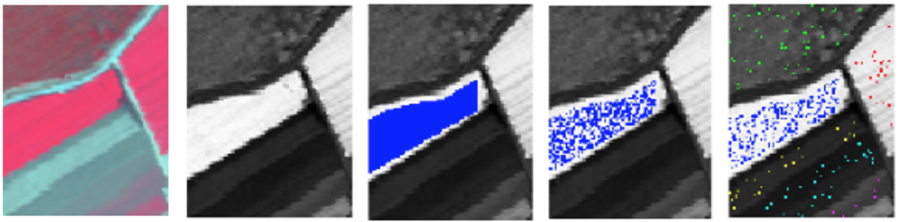
- Naive Bayes is a commonly used model in machine learning
- A deterministic prediction can be obtained by choosing the most likely class
- Performs well in a wide range of applications
- The boosted version is one the most effective general-purpose learning algorithms
- Naive Bayes learning scales well to very large problems
  - with n Boolean attributes there are only $2n + 1$ parameters
- Deals well with noisy or missing data
- Can give probabilistic predictions when approapriate
- Drawback: the conditional independence assumption is seldom accurate
  - can lead to overconfident probabilities that are often close to 0 or 1 especially with large numbers of attributes

Application in Hyperspectral Image (HSI) classification



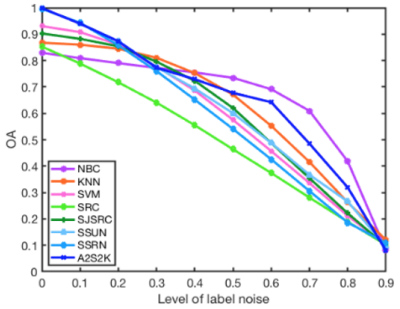
Left: *Houston University* HSI (144 bands); Left: *Pavia University* HSI (115 bands)

Naive Bayes robustness: Example from HSI analysis with noisy labels

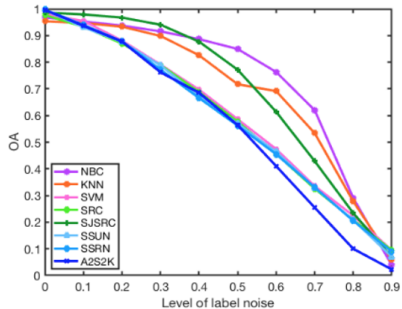


An illustration of label noise; (a) part of an original HSI; (b) one of the image bands; (c) ground truth for class 1 overlayed; (d) an example of training data for class 1 when all the labels are correct; (e) erroneous labels (noisy labels) are present. Some samples from other classes (denoted by dots in colors other than blue) are wrongly declared as examples for class 1.

Naive Bayes robustness: Example from HSI analysis with noisy labels



Houston University HSI dataset



Pavia University dataset

Comparison of classification accuracies at different levels of label noise.

NBC: Naive Bayes Classifier; SRC, SJSRC: Sparse Representation Classification models

SSUN, SSRN, A2S2K: deep learning models.

Observe how the overall accuracy (OA) of NBC gracefully decreases with label noise

M. Li, S. Huang, J. De Bock, G. De Cooman, and A. Pizurica. A robust dynamic classifier selection approach for hyperspectral images with imprecise label information. SENSORS, 20(18), 2020. https://doi.org/10.3390/s20185262

Naive Bayes robustness: Example from HSI analysis with noisy labels

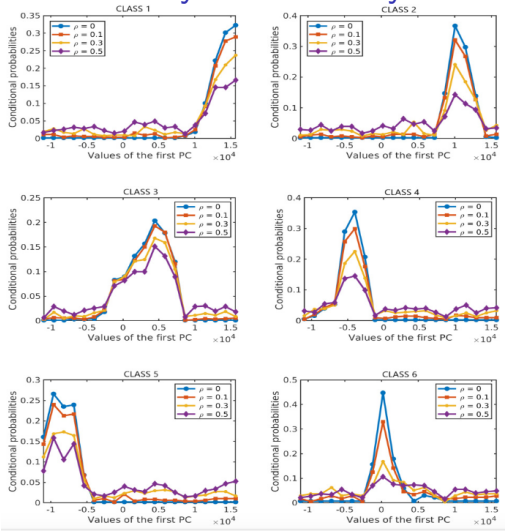
Why is NBC so robust to label noise?

- Features are the first N PCs
  - $f_i = PC_i$
  - $\hat{c} = \arg\max_c P(c) \prod_{i=1}^{N} P(f_i|c)$
- Conditional densities shown

$$P(f_i|c), \quad c \in \{1, \ldots, 6\}$$

for $i = 1,\ \rho \in \{0, 0.1, 0.4, 0.5\}$

- Similar behaviour for all i
- $P(f_i|c)$ for different c retain $\approx$ their relative proportions until they get flattened at very large ρ



ρ is the fraction of erroneous labels

M. Li, S. Huang, J. De Bock, G. De Cooman, and A. Pizurica. SENSORS, 20(18), 2020. https://doi.org/10.3390/s20185262

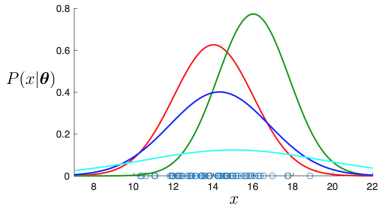
Generative and discriminative models

Two kinds of ML models are used for classifiers

- **Generative** – models the probability distribution of each class
  - E.g. the naive Bayes classifier
- **Discriminative** – learns the decision boundary between classes
  - E.g. logistic regression, decision trees, support vector machines
  - Gives the output class, but cannot generate new representatives from that class

Discriminative models tend to perform better in the classification tasks on very large data sets but on very small data sets generative models often do better

Maximum-likelihood learning: Continuous models



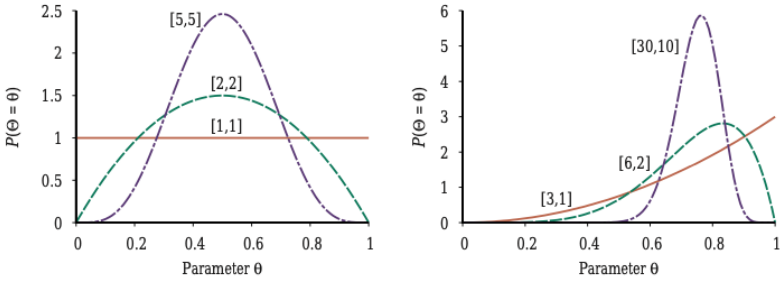
Assume $P(x|\boldsymbol{\theta}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. Our task is: learn $\boldsymbol{\theta} = \{\mu, \sigma\}$

Log-likelihood: $\ell(\mu, \sigma) = -\frac{N}{2}\log(2\pi) - \frac{N}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i^{(i)} - \mu)^2$

From $\frac{\partial \ell}{\partial \mu} = 0$ and $\frac{\partial \ell}{\partial \sigma} = 0$ we get:

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)} \quad \text{and} \quad \hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(x^{(i)} - \hat{\mu})^2$$

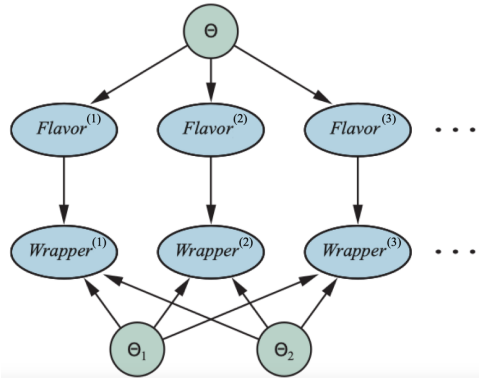
Bayesian parameter learning: Beta distribution



Examples of beta distributions for different values of (a,b)

$$Beta(\theta; a, b) = \alpha\ \theta^{(a-1)}(1-\theta)^{(b-1)}$$

Bayesian learning process

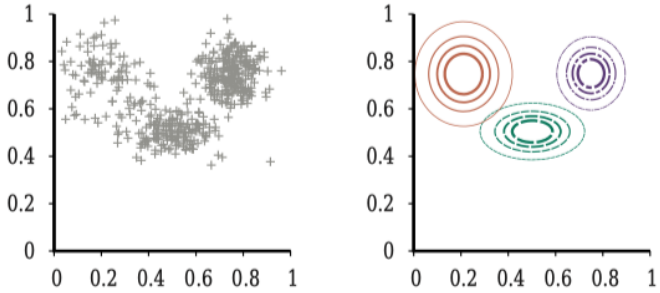


E.g., $$\begin{aligned} P(\theta \mid D^{(1)} = cherry) &= \alpha P(D^{(1)} = cherry \mid \theta) P(\theta) \\ &= \alpha' \theta \cdot Beta(\theta; a, b) = \alpha' \theta \cdot \theta^{(a-1)} (1-\theta)^{(b-1)} \\ &= \alpha' \theta^{a} (1-\theta)^{(b-1)} = \alpha' Beta(\theta; a+1, b) \end{aligned}$$

Hidden Variables and Missing Data

- Missing data: In practice data entries are often missing resulting in incomplete information to specify a likelihood
- Observational variables may be split into
  - Visible – those for which we actually know the state and
  - Missing – those whose states would nominally be known but are missing for a particular datapoint
- Latent Variables: Another scenario in which not all variables in the model are observed, but there are so-called hidden or latent variables. Latent variables are essential for the model description but never observed.
  - E.g., the underlying physics of a model may contain latent processes which are essential to describe the model, but cannot be directly measured

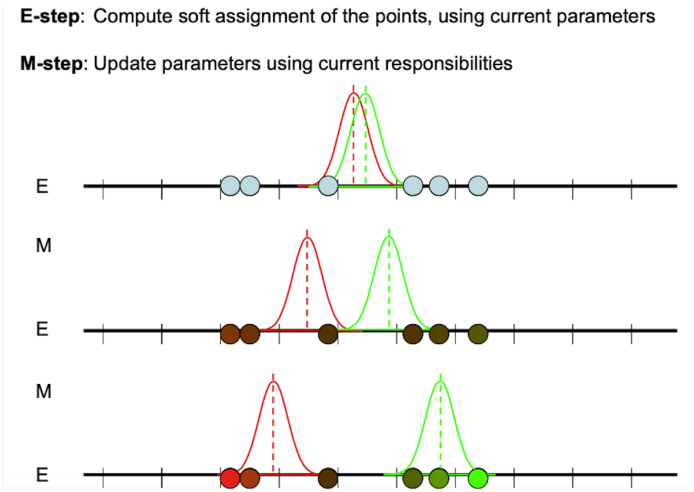
Gaussian Mixture Model (GMM)



A mixture distribution with k components: $P(\mathbf{x}) = \sum_{i=1}^{k} \underbrace{P(C=i)}_{\pi_i} P(\mathbf{x}|C=i)$

A Gaussian mixture model: $P(\mathbf{x}) = \sum_{i=1}^{k} \pi_i \, \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i)$

Expectation Maximization (EM) algorithm



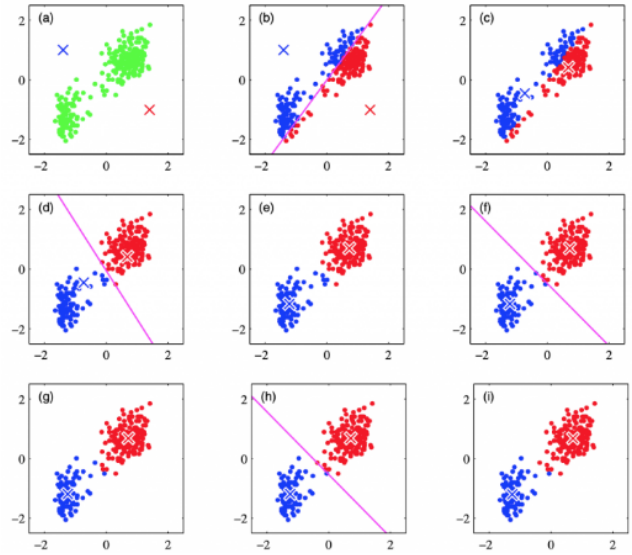
Credit: A. Zisserman: Clustering & Mixture Models

EM algorithm for the mixtures of Gaussians

Input: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$. Initialize the parameters arbitrarily and iterate the 2 steps:

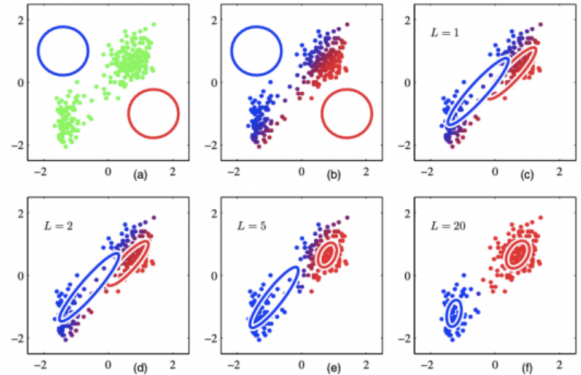
- **E-step**: Compute probabilities $p_{ij} = P(C = i|\mathbf{x}^{(j)})$
  - By Bayes' rule:
    $$p_{ij} = \alpha P(\mathbf{x}^{(j)}|C = i)P(C = i)$$
  - Define n_i as the effective number of data points assigned to component i:
    $$n_i = \sum_j p_{ij}$$
- **M-step**: Compute the new mean, covariance and weights
  - Means:
    $$\mu_i \leftarrow \sum_j p_{ij}\mathbf{x}^{(j)}/n_i$$
  - Covariance matrices:
    $$\Sigma_i \leftarrow \sum_j p_{ij}(\mathbf{x}^{(j)} - \mu_i)(\mathbf{x}^{(j)} - \mu_i)^\top/n_i$$
  - Weights:
    $$\pi_i \leftarrow n_i/N$$

Digression: K-means clustering



C. Bishop: *Pattern Recognition and Machine Learning*

Clustering by learning mixtures of Gaussians



C. Bishop: *Pattern Recognition and Machine Learning*

Hard clustering versus probabilistic clustering

K-Means

Initialize $\boldsymbol{\mu} = \mu_1, \dots \mu_k$ randomly
Iterate:

- **Step 1**: Set assignments $\mathbf{c}$ given $\boldsymbol{\mu}$ ($\mathbf{x}^{(j)} \rightarrow$ nearest centroid):

$$\forall j, \quad c^{(j)} = \arg \min_{i=1,\dots,k} \|\mathbf{x}^{(j)} - \mu_i\|^2$$

- **Step 2**: Set centroids $\boldsymbol{\mu}$ given $\mathbf{c}$ $i = 1, \dots, k$:

$$\mu_i \leftarrow \frac{1}{|\{j : c^{(j)} = i\}|} \sum_{j:c^{(j)}=i} \mathbf{x}^{(j)}$$

GMM

Initialize $\mu_i, \Sigma_i, i = 1, \dots k$
Iterate:

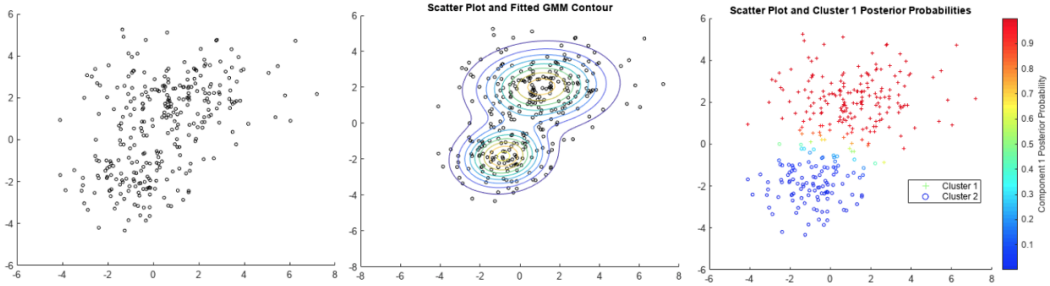
- **E-step**: Find probabilities that data points were generated by different components

$$\forall i, j \quad p_{ij} = P(C = i|\mathbf{x}^{(j)})$$

- **M-step**: Find new parameters that maximize the log-likelihood:

Using p_{ij}, update μ_i, Σ_i, π_i

GMM clustering example



Left: Simulated data from a mixture of Gaussian distributions; **Middle**: fitted 2-component GMM; **Right**: estimated class memberships and posterior probabilities

`https://fr.mathworks.com/help/stats/cluster-data-from-mixture-of-gaussian-distributions.html`

Summary

- Bayesian learning = learning as a form of probabilistic inference
  - Prior distribution over h updated based on observations
- MAP learning selects a single most likely hypothesis given the data
  - Still usess a prior on h; More tractable than full Bayesian learning
- Maximum-likelihood learning: simply maximize the data likelihood
  - Equivalent to MAP learning with a uniform prior
  - Naive Bayes learning is a particularly effective technique that scales well
- When some variables are hidden, parameters can be learned by the EM algorithm
  - Applications include clustering using Gaussian Mixture Models (GMM)
  - GMM-based clustering can be seen as a probabilistic version of K-means