USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

Hidden Markov Models

and related topics

Part 1

Machine Learning 2019

Michael Wand, Jürgen Schmidhuber, Cesare Alippi

TAs: Robert Csordas, Krsto Prorokovic, Xingdong Zou, Francesco Faccio, Louis Kirsch

Introduction

- Today we return to probabilistic modeling!
  - Important class of machine learning models
  - HMMs were for many years the standard way to deal with sequential data
    - concepts are useful also in the time of neural networks
  - Numerous further applications (you will meet Markov models again in *Reinforcement Learning,* for example)

- Outline for the next two lectures:
  - the *Gaussian Mixture* classifier: a model with hidden variables
  - Markov models
  - HMMs
  - Applications

Gaussian Mixtures

Recap: Gaussian distribution

- Remember the Gaussian distribution $N(\mu, \sigma^2)$ in one dimension
- Probability *density* is given by the formula

$$N(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

- Parameters: mean μ, variance σ^2
  - we know how to estimate them from a sample
- Prototypical example of a distribution which models a value with some uncertainty
  - if a value is generated by many independent random factors, its value follows a Gaussian distribution (under some preconditions, ->CLT)

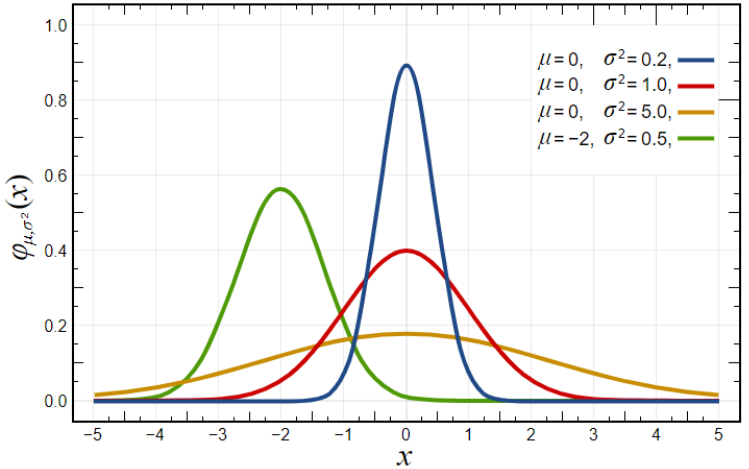


Image source: Wikipedia

Multivariate Gaussian distribution

USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

- A random *vector* $x \in R^N$ can also be Gaussian
  - probability density is given by the formula

$$N(x|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^N \det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

  - where μ is the N -dimensional mean vector and Σ is the $N \times N$ *covariance matrix.*
- Example: two-dimensional Gaussian $(X,Y) \sim N(\mu,\Sigma)$

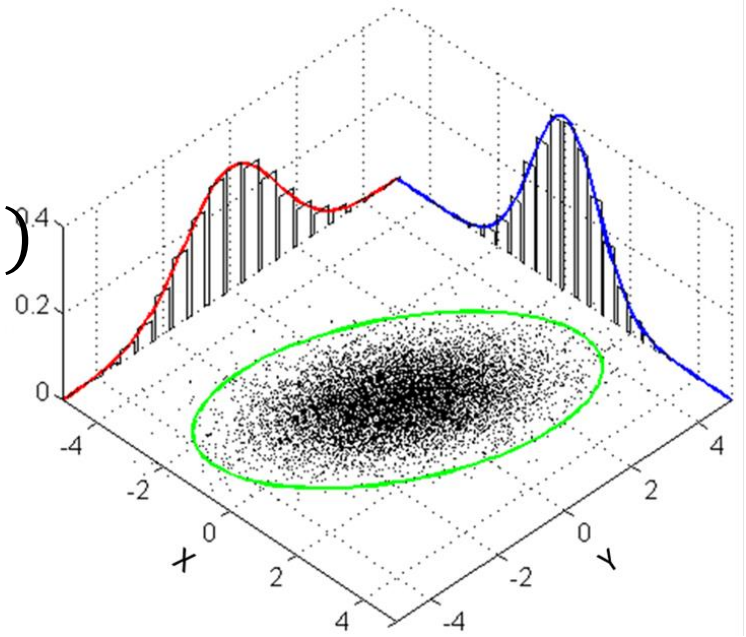


Image source: Wikipedia

Multivariate Gaussian distribution

- The *covariance* of two random variables *X* and *Y* is defined as follows:

$$Cov(X,Y) = E[\,(X - E(X)) \cdot (Y - E(Y))]$$

- Note that this is a generalization of the variance since

$$Cov(X,Y) = Var(X)$$

- When two random variables are independent, their covariance is zero (but *not* the other way round!)
  - However, if two random variables are (known to be) Gaussian, equivalence holds: Two Gaussians are independent if and only if their covariance is zero

Multivariate Gaussian distribution

- The covariance can be estimated from joint samples of X and Y, similar to the variance estimation:

$$Cov_{estimated}(X,Y) = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_x)(y_n - \mu_y)$$

with *N* joint samples *((x_n,y_n)).*

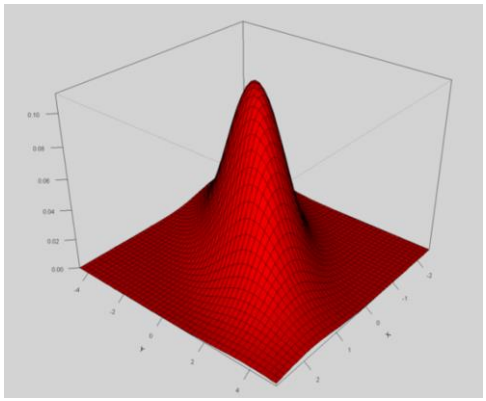
- Finally, the *covariance matrix* generalizes the variance to multiple dimensions. It holds the covariance for pairs of components, for example for a three-dimensional random vector *(X_1, X_2, X_3):*

$$\Sigma = \begin{pmatrix} Cov(X_1,X_1) & Cov(X_1,X_2) & Cov(X_1,X_3) \\ Cov(X_2,X_1) & Cov(X_2,X_2) & Cov(X_2,X_3) \\ Cov(X_3,X_1) & Cov(X_3,X_2) & Cov(X_3,X_3) \end{pmatrix}$$

It is symmetric; the diagonal holds the variances of the single variables.

Multivariate Gaussian distribution

- We typically display two-dimensional Gaussians with height lines, as the green one in the image
- more dimensions are hard to draw…



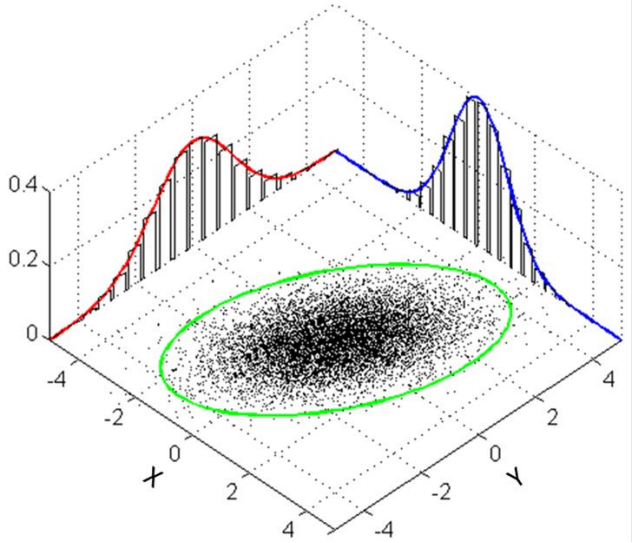


Image source: Wikipedia

Multivariate Gaussian distribution

- three possible cases of the bivariate Gaussian:
  - a) general case: X and Y are *not* independent, the covariance matrix is a full 2*2 matrix
  - b) X and Y are *independent*, the covariance matrix is a diagonal matrix
  - c) X and Y are independent and have the same variance

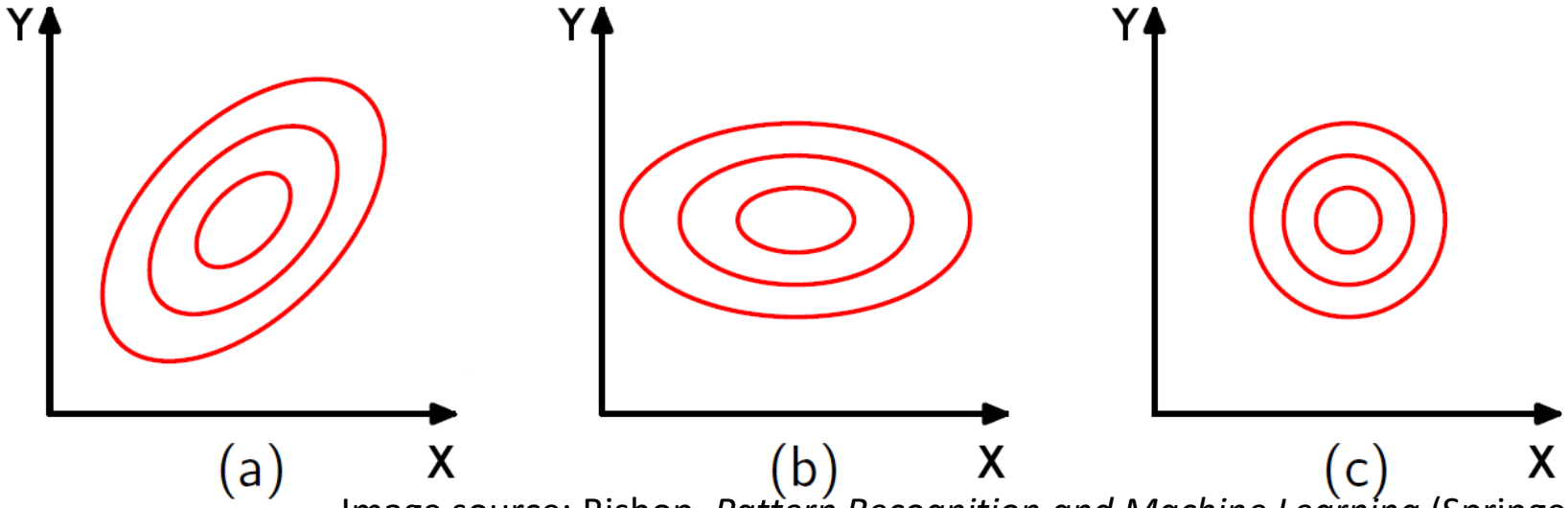


Image source: Bishop, *Pattern Recognition and Machine Learning* (Springer 2006), modified

Recap: Gaussian Classifier

- We have learned that one can use the Gaussian distribution to define a probabilistic classifier.
  - for each class, estimate the (usually multivariate) Gaussian by ML method
    - this means estimating the mean vector and the covariance matrix
    - resulting distributions $\mathcal{N}_c = \mathcal{N}(\mu_c, \sigma_c)$ for each class c
  - apply the classifier to a new sample by computing the value of the probability density function for each class, then take the argmax
    - $\hat{c} = \text{argmax}_c \mathcal{N}_c(x)$ is the estimated class

Gaussian Mixtures

- **Problem:** a single Gaussian per class is not good enough for modeling!
- Consider the example to the right: the best single Gaussian does not well describe the underlying data
  - with two Gaussians, it's quite OK!

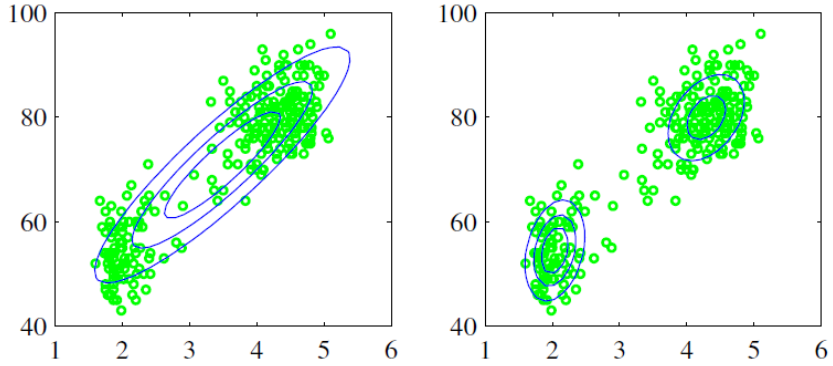


Image source: Bishop, fig. 2.21

- Solution: use a *Gaussian Mixture* distribution

$$p(x) = \sum_k \pi_k \mathcal{N}(\mu_k, \sigma_k)$$

with positive weights π_k whose sum is 1, and *k* single Gaussians $\mathcal{N}(\mu_k, \sigma_k)$

- Just for clarification: this is to model a *single* class, not all possible classes!

Gaussian Mixtures

- A further example: Component Gaussians, mixture, 3D plot of the density function

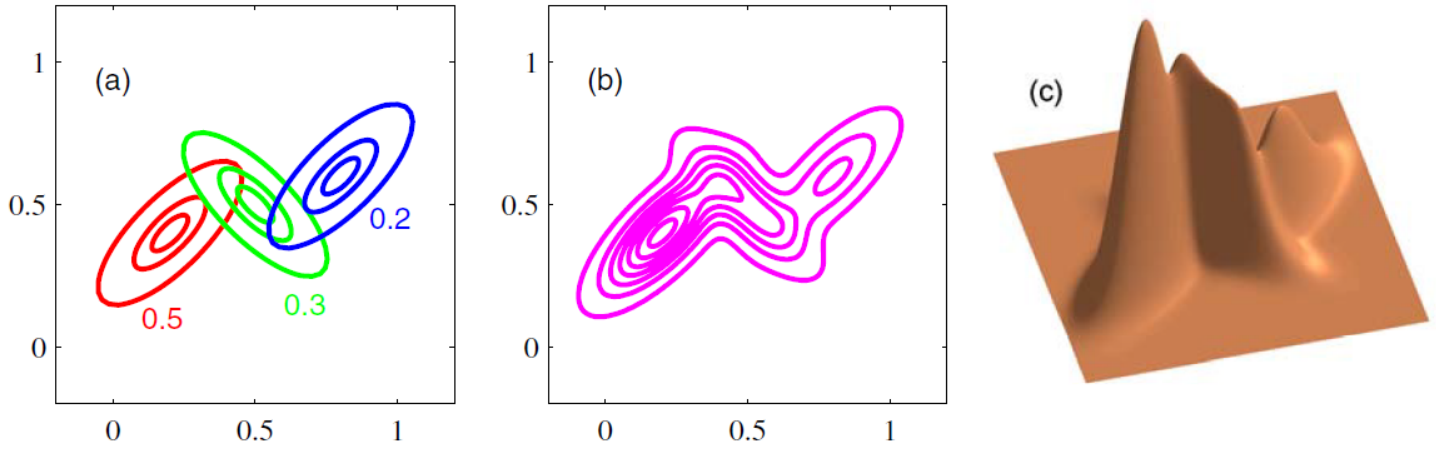


Figure 2.23 Illustration of a mixture of 3 Gaussians in a two-dimensional space. (a) Contours of constant density for each of the mixture components, in which the 3 components are denoted red, blue and green, and the values of the mixing coefficients are shown below each component. (b) Contours of the marginal probability density $p(\mathbf{x})$ of the mixture distribution. (c) A surface plot of the distribution $p(\mathbf{x})$.

Image source: Bishop, fig. 2.23

Gaussian Mixtures

- Problem: how do we do Maximum Likelihood estimation for Gaussian mixtures?
  - would be easy if we knew which data points "belong" to which Gaussian
    - which may not be known
    - for real data this question might not even make sense
    - remember that we *assume* that we can model our data reasonably well with a mixture of Gaussians
  - we need to *jointly* estimate class *parameters* and class *assignments*
  - ***no closed-form solution is known***

Gaussian Mixtures

- The problem is linked to Bayesian reasoning:
  - Define z_n as the assignment of data point x_n to a Gaussian
  - z_n is a vector with *K* components, where *K* is the number of classes, e.g. $z_n = (0,0,0,1,0,...)$ if data point x_n belongs to the fourth Gaussian
  - We have $p(x_n|z_{nk} = 1) = N(x|\mu_k, \Sigma_k)$
  - $p(z_{nk} = 1) = \pi_k$ can be seen as the *prior probability* that a data point belongs to class *k*
    - prior: *before* having examined the actual data point
  - z_n is called a *hidden* or *latent* variable.

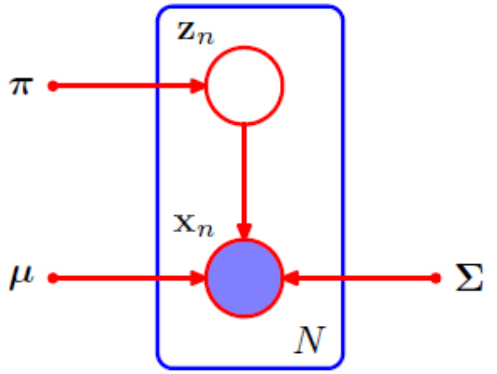


Image source: Bishop, fig. 9.6

Gaussian Mixtures

USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

- z_n: Assignment vector of data point x_n to one of K underlying Gaussians
- We have $p(x_n|z_{nk} = 1) = N(x|\mu_k, \Sigma_k)$
- $p(z_{nk} = 1) = \pi_k$ can be seen as the *prior probability* that a data point belongs to class k
- Consequently,

$$p(x_n) = \sum_k p(x_n \wedge z_{nk} = 1) = \sum_k p(z_{nk} = 1) \cdot p(x_n|z_{nk} = 1) = \sum_k \pi_k \mathcal{N}(\mu_k, \sigma_k)$$

- Finally, we define the posterior probabilities $\gamma_{nk} := p(z_{nk} = 1|x_n)$ and compute (Bayes!)

$$\gamma_{nk} = p(z_{nk} = 1|x_n) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \sigma_k)}{\sum_j \pi_j \mathcal{N}(x_n|\mu_j, \sigma_j)}$$

Image source: Bishop, fig. 9.6

The Expectation Maximization Algorithm

- We cannot optimize class assignments and class parameters at once
- But we can optimize them *iteratively!*
- Alternatingly recompute class assignments ("responsibilities" of each data point for each Gaussian), and class parameters
- This important algorithm is called *Expectation Maximization (EM)*
  - we will not go into the mathematical details on why this name makes sense

The Expectation Maximization Algorithm

- **EM (Expectation Maximization) Algorithm**
  1. Initialize (randomly) the means μ_k, covariances Σ_k, and mixing coefficients π_k
  2. E-step - evaluate "responsibilities" of each data point for each class, using current parameters:

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}(\mu_k, \sigma_k)}{\sum_j \pi_j \mathcal{N}(\mu_j, \sigma_j)}$$

  3. M-step – Reestimate class parameters using new responsibilities

$$\mu_k^{new} = \frac{1}{N_k} \sum_n \gamma_{nk} x_n \qquad \Sigma_k^{new} = \frac{1}{N_k} \sum_n \gamma_{nk} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N}$$

     with $N_k = \sum_n \gamma(z_{nk})$. Note that $\sum_k N_k = N$.
  4. Repeat steps 2 and 3 until the dataset likelihood $p(X|\mu, \Sigma, \pi)$ converges.

The Expectation Maximization Algorithm

- It can be shown that the likelihood (or, for practical computation, its logarithm) increases in each step
- The EM algorithm might converge to a local maximum (instead of true maximum)
- Degenerate cases (singularities) are possible
  - and get increasingly problematic at high dimensionalities
  - heuristics might be required to avoid them
  - *dimensionality reduction* (PCA, LDA) often required
- Overfitting is possible
  - frequent solution: constrain covariance matrices to be diagonal
- Convergence may be slow
  - start with further constraints
- *Hidden variables* is the most important concept to remember today.

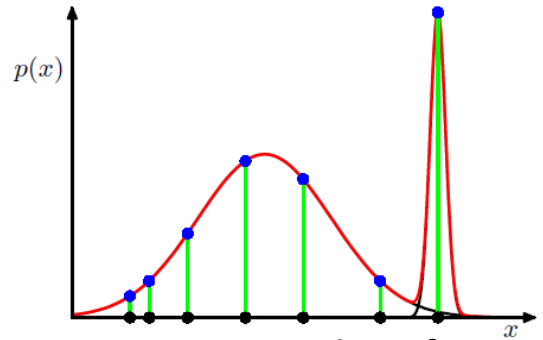


Image source: Bishop, fig. 9.7

EM Example

- This is what it looks like when the algorithm converges decently
- After *L=20* steps, convergence is reached
- Notice the totally random initialization

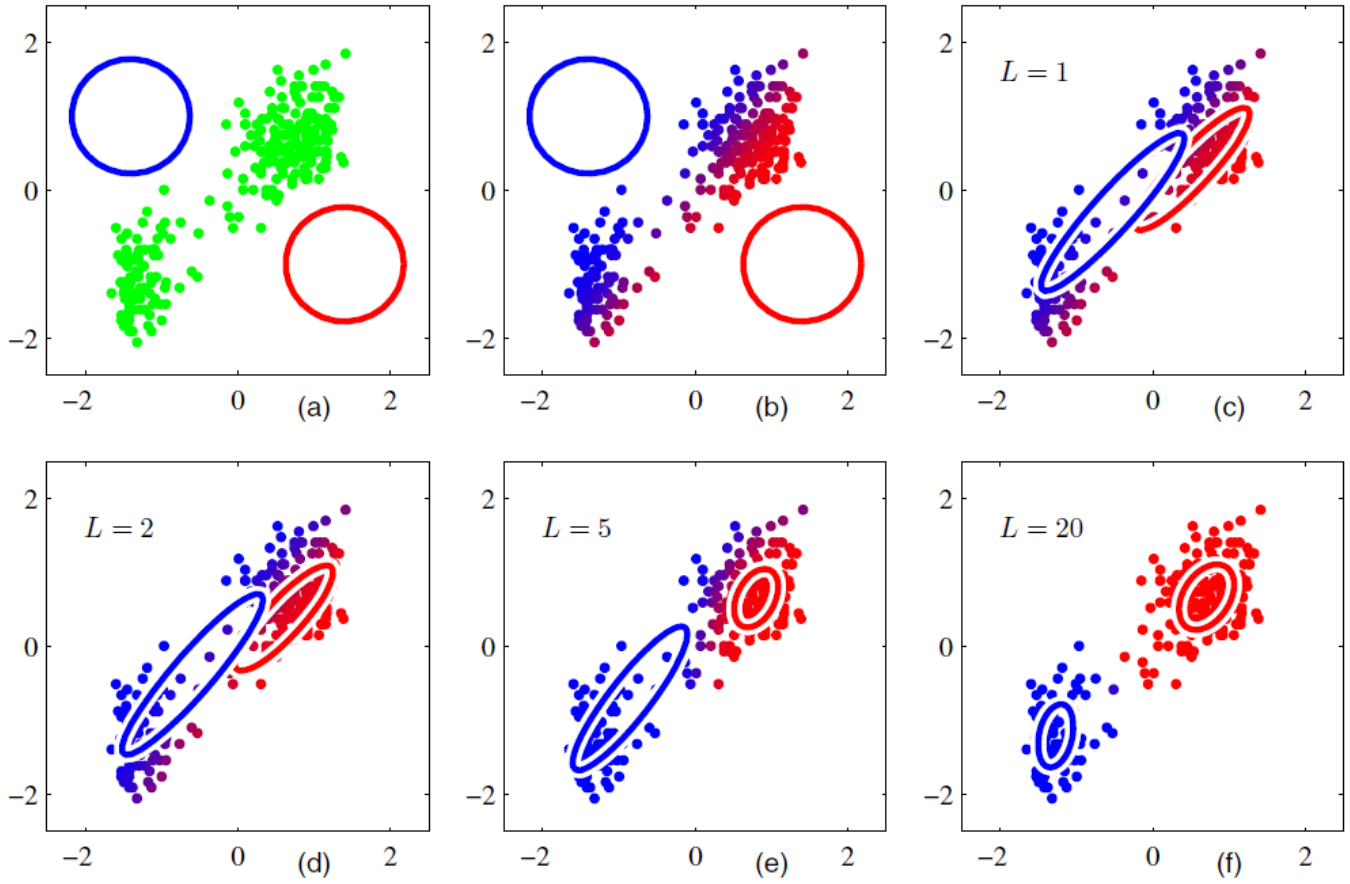


Image source: Bishop, fig. 9.8

The Gaussian Mixture Classifier

USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

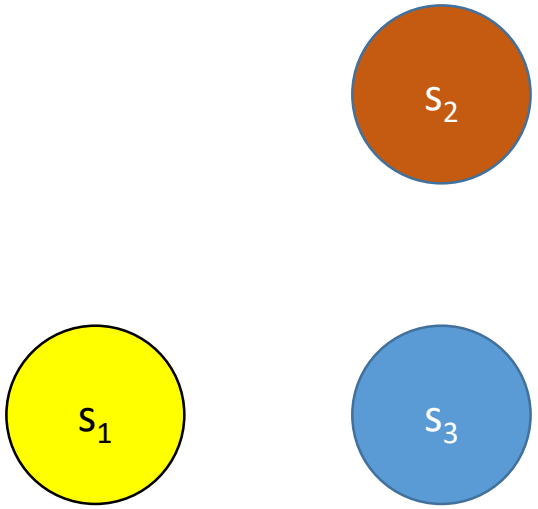
- We now have collected all parts which we need for constructing a practically useful classifier: The Gaussian Mixture classifier
  - Step 1: for *each* class, collect all samples and perform EM in order to obtain a good probabilistic model for the class: A Gaussian Mixture model (GMM)
  - Step 2: to apply the classifier to a new sample x, compute the mixture probabilities at position x for each class, and compute the argmax

Markov Models

(this part is based on a lecture By Prof. Andrew Moore, Carnegie Mellon University)

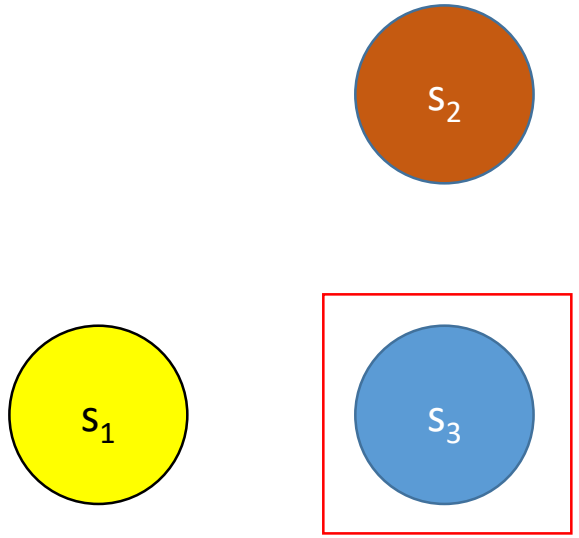
A Markov system

- Has *N* states, called s_1, s_2, … s_N.
- There are discrete timesteps *t=0, t=1,* …



A Markov system

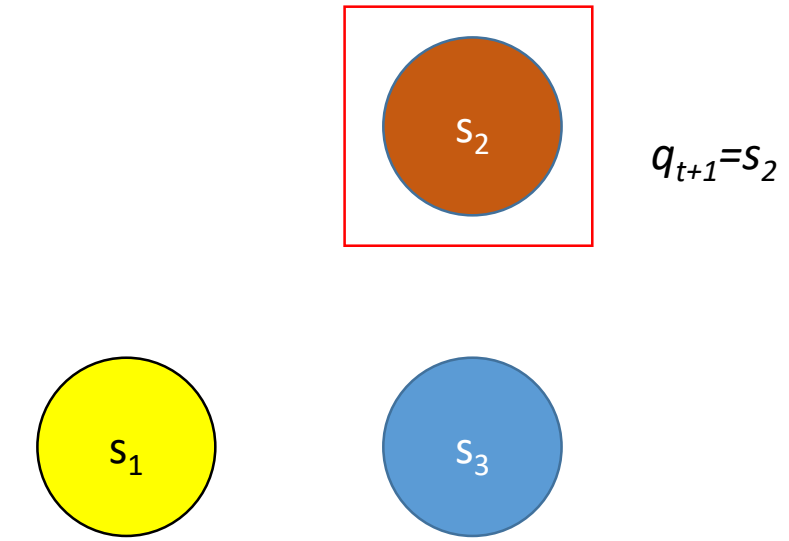
- Has *n* states, called s_1, s_2, … s_N.
- There are discrete timesteps *t=0, t=1,* …
- At each timestep, the system is in exactly one of the available states. Call it q_t
- $q_t \in \{s_1, \ldots, s_N\}$



current state: q_t=s_3

A Markov system

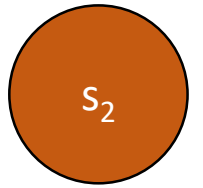
- Has *n* states, called s_1, s_2, … s_N.
- There are discrete timesteps *t=0, t=1,* …
- At each timestep, the system is in exactly one of the available states. Call it q_t
- $q_t \in \{s_1, \ldots, s_N\}$
- Between each timestep, the next state is chosen randomly

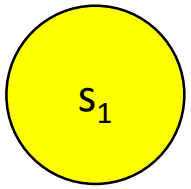


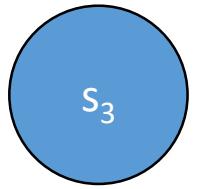
A Markov system

- Has *n* states, called s_1, s_2, … s_N.
- There are discrete timesteps *t=0, t=1,* …
- At each timestep, the system is in exactly one of the available states. Call it q_t
- $q_t \in \{s_1, \ldots, s_N\}$
- Between each timestep, the next state is chosen randomly
- The current state (and *only* the current state) determines the distribution for the next state

$$P(q_{t+1} = s_1 | q_t = s_2) = \frac{1}{2}$$
$$P(q_{t+1} = s_2 | q_t = s_2) = \frac{1}{2}$$
$$P(q_{t+1} = s_3 | q_t = s_2) = 0$$







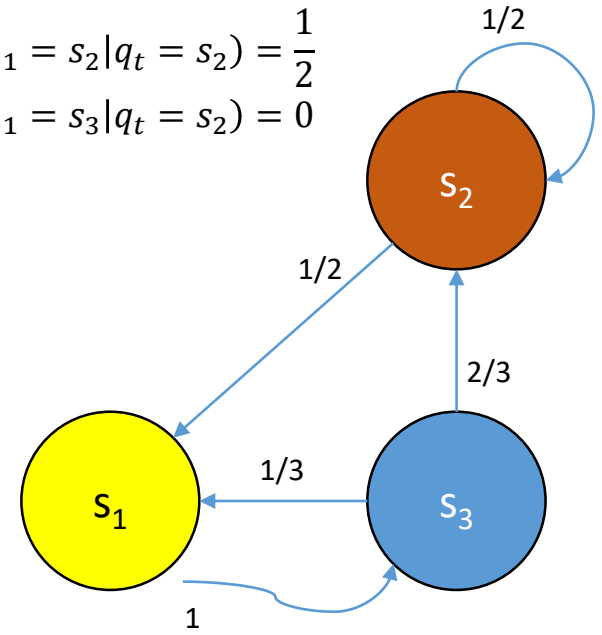
$$P(q_{t+1} = s_1 | q_t = s_1) = 0$$
$$P(q_{t+1} = s_2 | q_t = s_1) = 0$$
$$P(q_{t+1} = s_3 | q_t = s_1) = 1$$

$$P(q_{t+1} = s_1 | q_t = s_3) = \frac{1}{3}$$
$$P(q_{t+1} = s_2 | q_t = s_3) = \frac{2}{3}$$
$$P(q_{t+1} = s_3 | q_t = s_3) = 0$$

A Markov system

- Has *n* states, called s_1, s_2, … s_N.
- There are discrete timesteps *t=0, t=1,* …
- At each timestep, the system is in exactly one of the available states. Call it q_t
- $q_t \in \{s_1, \ldots, s_N\}$
- Between each timestep, the next state is chosen randomly
- The current state (and *only* the current state) determines the distribution for the next state

$$P(q_{t+1} = s_1 | q_t = s_2) = \frac{1}{2}$$
$$P(q_{t+1} = s_2 | q_t = s_2) = \frac{1}{2}$$
$$P(q_{t+1} = s_3 | q_t = s_2) = 0$$

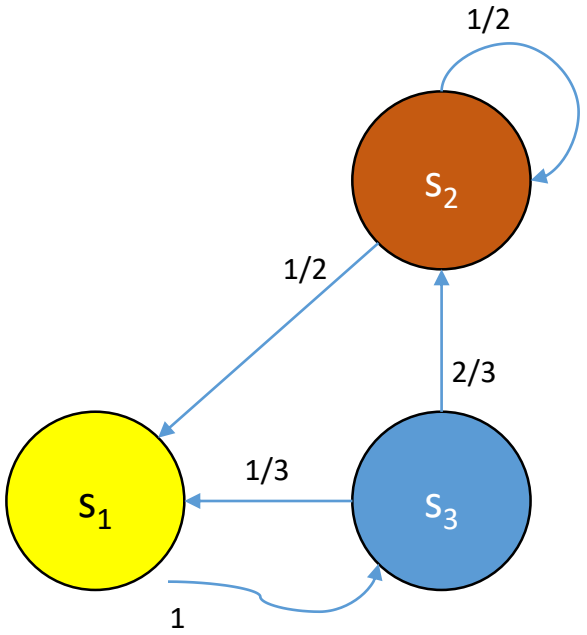


$$P(q_{t+1} = s_1 | q_t = s_1) = 0$$
$$P(q_{t+1} = s_2 | q_t = s_1) = 0$$
$$P(q_{t+1} = s_3 | q_t = s_1) = 1$$

$$P(q_{t+1} = s_1 | q_t = s_3) = \frac{1}{3}$$
$$P(q_{t+1} = s_2 | q_t = s_3) = \frac{2}{3}$$
$$P(q_{t+1} = s_3 | q_t = s_3) = 0$$

A Markov system

- Has *n* states, called s_1, s_2, … s_N.
- There are discrete timesteps *t=0, t=1,* …
- At each timestep, the system is in exactly one of the available states. Call it q_t
- $q_t \in \{s_1, \ldots, s_N\}$
- Between each timestep, the next state is chosen randomly
- The current state (and *only* the current state) determines the distribution for the next state
- The initial state can be random or deterministic (e.g. $q_1=s_1$)



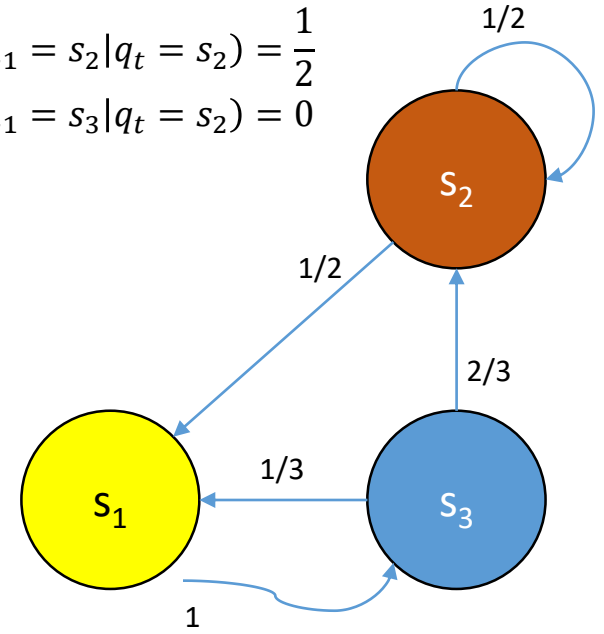
The *Markov Property*

USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

- The *Markov Property*:
  - q_{t+1} is *conditionally independent* of q_{t-1}, q_{t-2}, … given q_t.
- That is,

$$P(q_{t+1} = s_j | q_t = s_i) = P(q_{t+1} = s_j | q_t = s_i, \text{any earlier history})$$

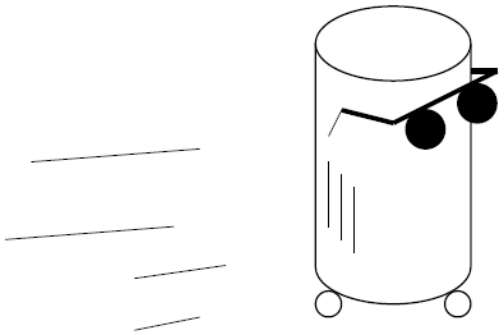
$$P(q_{t+1} = s_1 | q_t = s_2) = \frac{1}{2}$$
$$P(q_{t+1} = s_2 | q_t = s_2) = \frac{1}{2}$$
$$P(q_{t+1} = s_3 | q_t = s_2) = 0$$



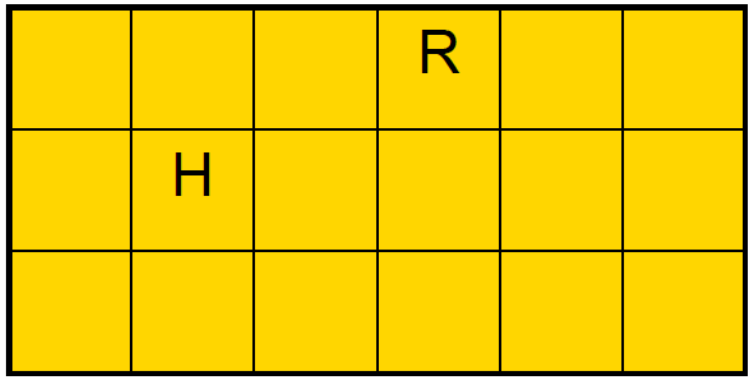
$$P(q_{t+1} = s_1 | q_t = s_1) = 0$$
$$P(q_{t+1} = s_2 | q_t = s_1) = 0$$
$$P(q_{t+1} = s_3 | q_t = s_1) = 1$$

$$P(q_{t+1} = s_1 | q_t = s_3) = \frac{1}{3}$$
$$P(q_{t+1} = s_2 | q_t = s_3) = \frac{2}{3}$$
$$P(q_{t+1} = s_3 | q_t = s_3) = 0$$

Markov Example: A Blind Robot



A human and a robot wander around randomly on a grid…



STATE q = ⟨ Location of Robot, Location of Human ⟩

Note: N (num. states) = 18 * 18 = 324

Markov Example: A Blind Robot

- System dynamics:
  - in each timestep, the human moves randomly to an adjacent cell
  - in each timestep, the robot also moves randomly to an adjacent cell

Markov Example: A Blind Robot

- System dynamics:
  - in each timestep, the human moves randomly to an adjacent cell
  - in each timestep, the robot also moves randomly to an adjacent cell
- Typical questions:
  - "What is the expected time until the human is crushed by the robot?"
  - "What's the probability that the robot will hit the left wall before it hits the human?"
  - "What's the probability that the Robot crushes human on next time step?"

Markov Example: A Blind Robot

USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

- System dynamics:
  - in each timestep, the human moves randomly to an adjacent cell
  - in each timestep, the robot also moves randomly to an adjacent cell
- Typical questions:
  - "What is the expected time until the human is crushed by the robot?"
  - "What's the probability that the robot will hit the left wall before it hits the human?"
  - "What's the probability that the Robot crushes human on next time step?"

Blind robot: Will compute this now

Omnipotent (and evil) robot: too easy and not probabilistic -> boring

Markov Example: A Blind Robot

USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

- System dynamics:
  - in each timestep, the human moves randomly to an adjacent cell
  - in each timestep, the robot also moves randomly to an adjacent cell
- Typical questions:
  - "What is the expected time until the human is crushed by the robot?"
  - "What's the probability that the robot will hit the left wall before it hits the human?"
  - "What's the probability that the Robot crushes human on next time step?"

Blind robot: Will compute this now

Omnipotent (and evil) robot: too easy and not probabilistic -> boring

Robot with some sensors but incomplete information: Requires *Hidden* Markov Model (next time)

Markov Model Computations

USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

- Question: What is $P(q_t=s)$?
- Step 1: For any path $Q = (q_1, q_2, \ldots, q_t)$, compute its probability:

$$P(q_1, q_2, \ldots, q_t) = P(q_t|q_1, \ldots, q_{t-1}) \cdot P(q_1, \ldots, q_{t-1}) = P(q_t|q_{t-1}) \cdot P(q_1, \ldots, q_{t-1})$$

because of the Markov property (!).
Now we can recurse and get

$$P(q_1, q_2, \ldots, q_t) = P(q_t|q_{t-1}) \cdot P(q_{t-1}|q_{t-2}) \cdot \ldots \cdot P(q_2|q_1)$$

Finally, we have

$$P(q_t = s) = \sum_{Q \in \text{paths of length t which end in state s}} P(Q)$$

Markov Model Computations

USI/SUPSI
Istituto
Dalle Molle
di studi
sull'intelligenza
artificiale
IDSIA

- Unfortunately, this computation is extremely expensive (count the number of possible paths…)
- Can we do better?

Markov Model Computations

- Unfortunately, this computation is extremely expensive (count the number of possible paths…)
- Can we do better? Yes, by recursion.
- For each state s_i, define $p_t(i) = P(q_t = s_i)$
- Easily done inductively:
  - $p_0(i) = \begin{cases} 1 \text{ if } s_i \text{ is the start state} \\ 0 \text{ otherwise} \end{cases}$

  $p_{t+1}(j) = \sum_{i=1}^{N} P(q_{t+1} = s_j \wedge q_t = s_i)$

  $= \sum_{i=1}^{N} P(q_{t+1} = s_j | q_t = s_i) \cdot P(q_t = s_i) = \sum_{i=1}^{N} a_{ij} p_t(i)$

  with $a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$.

Markov Model Computations

- Unfortunately, this computation is extremely expensive (count the number of possible paths…)
- Can we do better? Yes, by recursion.
- For each state s_i, define $p_t(i) = P(q_t = s_i)$
- Easily done inductively:
  - $p_0(i) = \begin{cases} 1 \text{ if } s_i \text{ is the start state} \\ 0 \text{ otherwise} \end{cases}$

$p_{t+1}(j) = \sum_{i=1}^{N} P(q_{t+1} = s_j \wedge q_t = s_i)$

$= \sum_{i=1}^{N} P(q_{t+1} = s_j | q_t = s_i) \cdot P(q_t = s_i) = \sum_{i=1}^{N} a_{ij} p_t(i)$

with $a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$.

- Computation is simple.
- Just fill in this table in this order:

t	$p_t(1)$	$p_t(2)$	…	$p_t(N)$
0	0	1		0
1				
:				
t_{final}				

Markov Model Computations

USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

- With the inductive method, $p_t(i)$ can now be computed in $O(tN^2)$ steps
  - compare *dynamic programming*
- With the naïve method, $O(N^t)$ steps are required

- When we do *Hidden* Markov models, we will encounter further examples of this type of computation!

Preview of Hidden Markov Models

USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

- The Markov models we got to know
  - a) don't seem to have anything to do with GMMs
  - b) don't seem to have to do a lot with classification
- Enter the *Hidden* Markov Model (HMM)
  - we have *observations*, which are tied to Markov states
  - but the *states themselves are hidden* – we cannot directly observe them

Preview of Hidden Markov Models

USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

- The Markov models we got to know
  - a) don't seem to have anything to do with GMMs
  - b) don't seem to have to do a lot with classification
- Enter the *Hidden* Markov Model (HMM)
  - we have *observations*, which are tied to Markov states
  - but the *states themselves are hidden* – we cannot directly observe them
  - we will have a variety of HMM tasks which are computed in a similar way as the Markov Model example before
  - the observations are probabilistic themselves and often modeled as Gaussian Mixtures
  - *very powerful framework* whose concepts still remain important (even though the classical HMM sequence model is being replaced by RNNs)

Conclusion / Outlook

USI/SUPSI
Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

- Today, we first got to know the Gaussian Mixture model
  - from which the Gaussian Mixture classifier can be derived
  - important concept: *hidden variables,* which in this case were a mathematical / statistical tool to formulate our problem
- and then the Markov models
  - as description of processes "without a memory" (future depends on current state, but not on the past)
  - with efficient recursive calculations
- Next time, we'll merge the Markov models and hidden variables -> HMM