Data Mining
Classification: Classifiers

Lecture Notes for Chapter 2

R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed. New York: Wiley, 2001.

Lecture Notes for Chapter 5

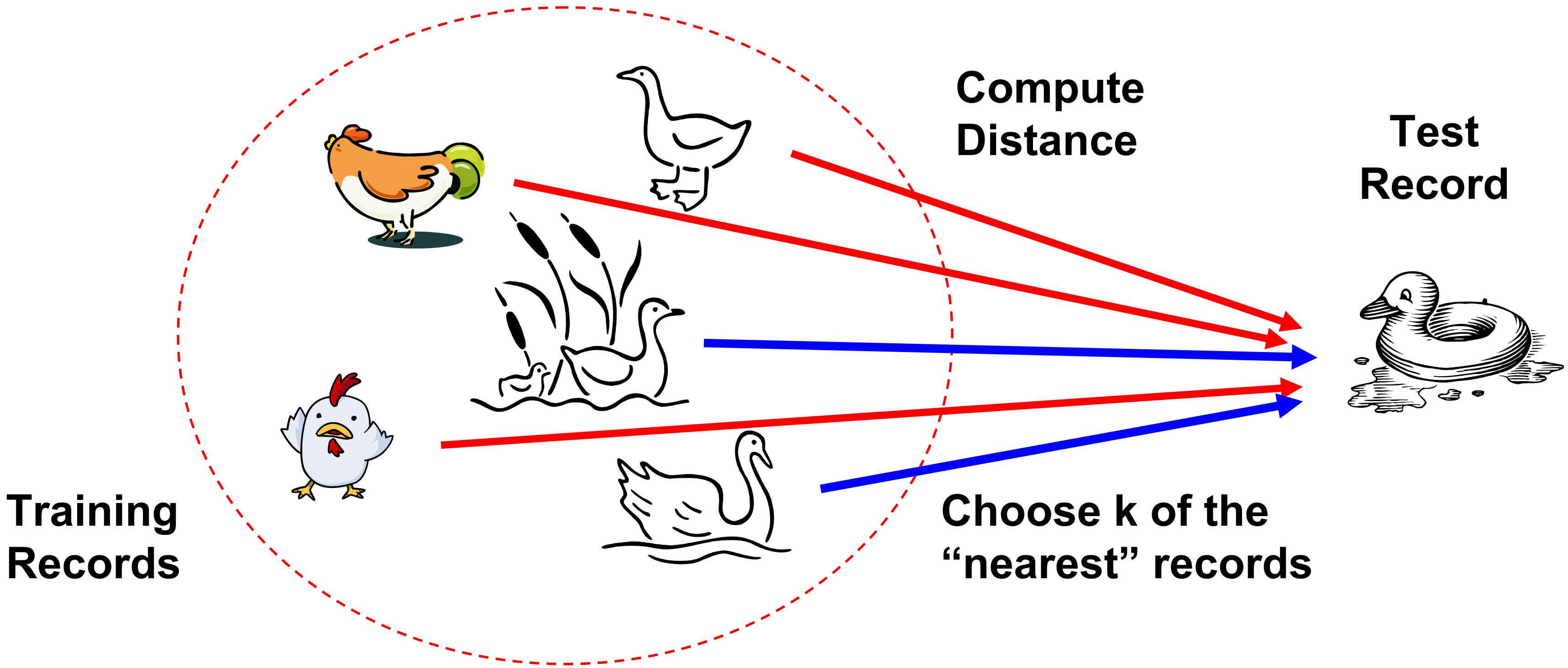
Introduction to Data Mining

by

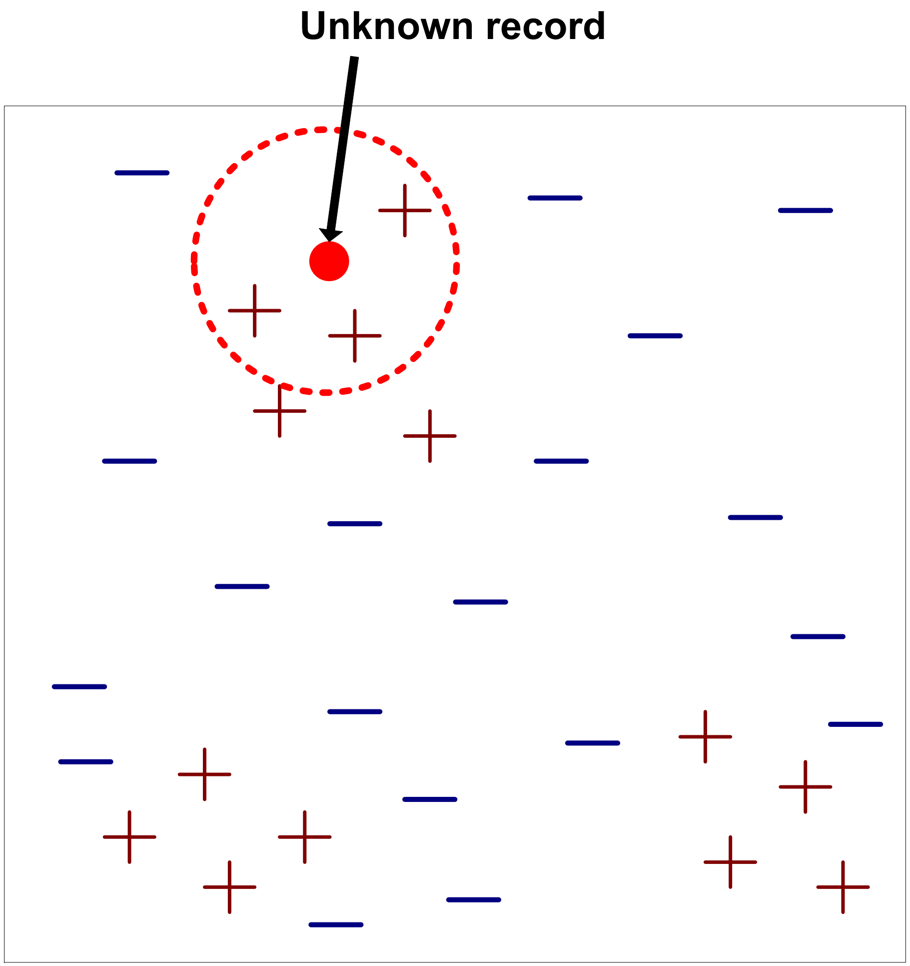
Tan, Steinbach, Kumar

Nearest Neighbor Classifiers (KNN)

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



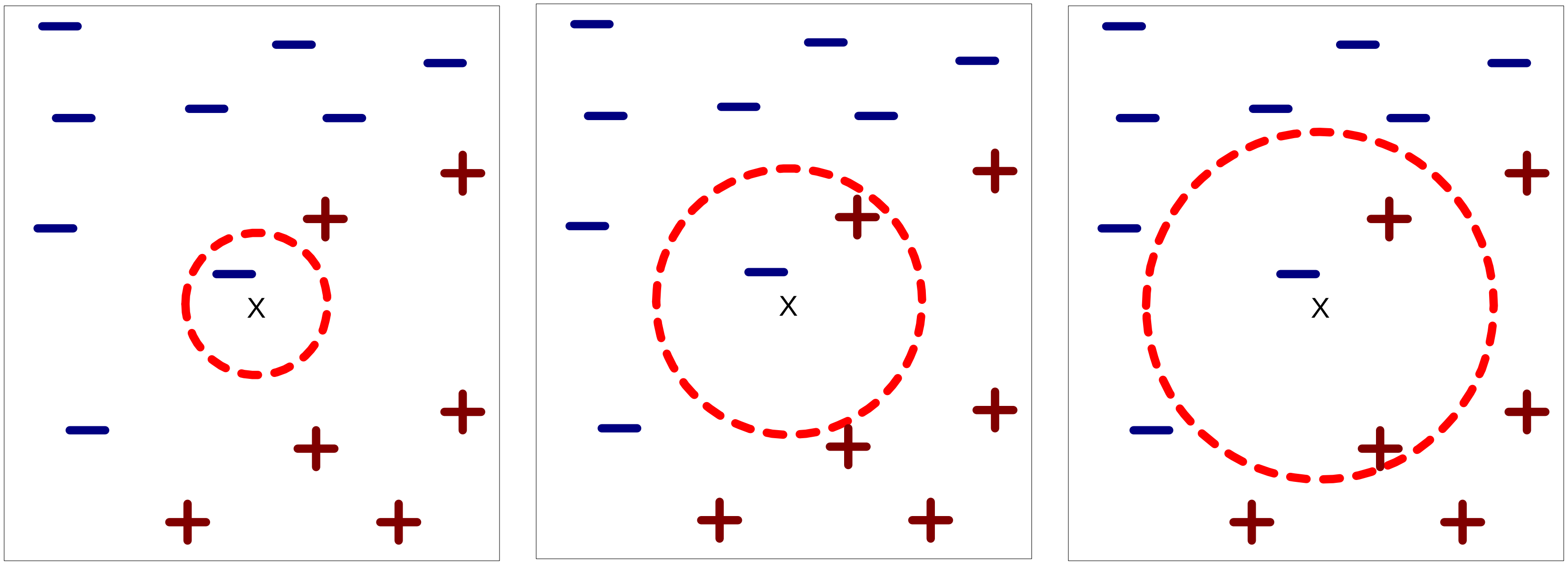
Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor

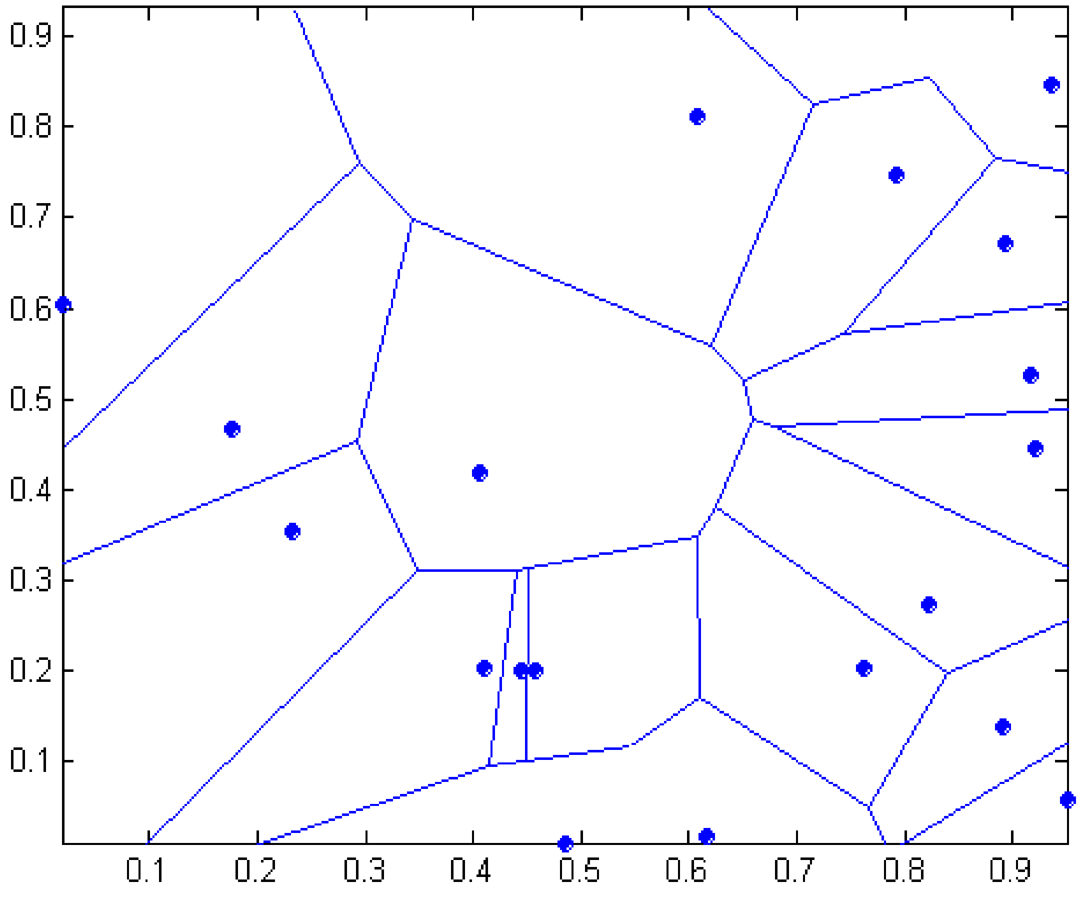


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

1 nearest-neighbor

Voronoi Diagram: http://en.wikipedia.org/wiki/Voronoi_diagram



Nearest Neighbor Classification

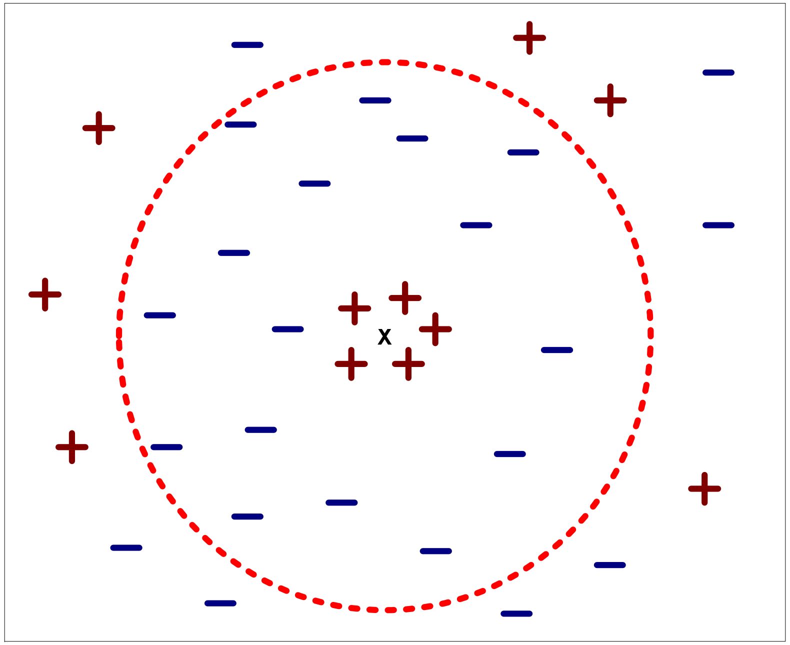
- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Nearest Neighbor Classification…

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M

Nearest Neighbor Classification…

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

1 1 1 1 1 1 1 1 1 1 1 0	vs	1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1		0 0 0 0 0 0 0 0 0 0 0 1
d = 1.4142		**d = 1.4142**

- Solution: Normalize the vectors to unit length

Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Classifying unknown records are relatively expensive (weakness)

Demo: **http://www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html**

Bayes Classifier (Bayesian Decision Theory)

- A probabilistic framework for solving classification problems
- Bayesian decision theory:

$$posterior = \frac{likelihood \times prior}{evidence}.$$

- Bayes theorem:

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)},$$

Naïve Bayes Classifier

- Assume independence among attributes x_i when class is given:
  - $P(x_1, x_2, \ldots, x_n \mid w_j) = P(x_1 \mid C_j)\, P(x_2 \mid C_j) \ldots P(x_n \mid w_j)$
  - Can estimate $P(x_i \mid w_j)$ for all x_i and w_j.
  - New point is classified to w_j if $P(w_j)\, \Pi\, P(A_i \mid C_j)$ is maximal.

How to Estimate Probabilities from Data?

- All model parameters (*i.e.*, class priors and feature probability distributions) can be approximated with relative frequencies from the training set.
- These are maximum likelihood estimates of the probabilities.
- Non-discrete features need to be discretized first. Discretization can be unsupervised (ad-hoc selection of bins) or supervised (binning guided by information in training data).

How to Estimate Probabilities from Data?

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

$$\text{Original: } P(x_i \mid \omega_j) = \frac{N_{ij}}{N_j}$$

$$\text{Laplace:} P(x_i \mid \omega_j) = \frac{N_{ij} + 1}{N_j + c}$$

$$\text{m-estimate:} P(x_i \mid \omega_j) = \frac{N_{ij} + mp}{N_j + m}$$

c: number of classes

p: prior probability

m: parameter

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(C) = N_c/N$
  - e.g., P(No) = 7/10, P(Yes) = 3/10
- For discrete attributes:

$$P(A_i \mid C_k) = |A_{ik}| / N_{c_k}$$

  - where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
  - Examples:

P(Status=Married|No) = 4/7
P(Refund=Yes|Yes)=0

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i,c_i) pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(x|\omega_1) = \frac{6}{7}\times\frac{6}{7}\times\frac{2}{7}\times\frac{2}{7} = 0.06$$

$$P(x|\omega_2) = \frac{1}{13}\times\frac{10}{13}\times\frac{3}{13}\times\frac{4}{13} = 0.0042$$

$$P(x|\omega_1)P(\omega_1) = 0.06\times\frac{7}{20} = 0.021$$

$$P(x|\omega_2)P(\omega_2) = 0.004\times\frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

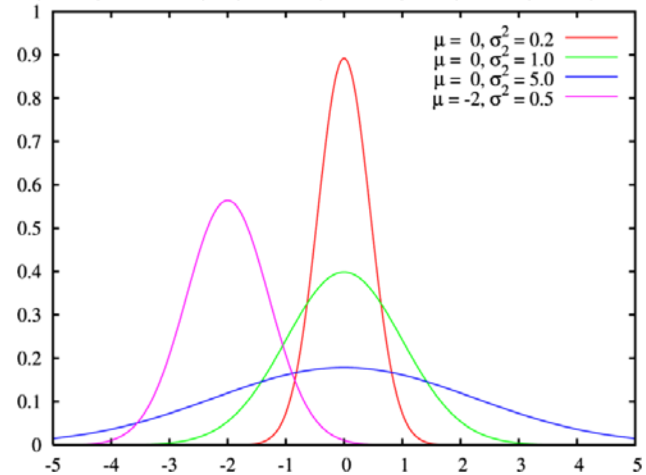
P(x|w1)P(w1) > P(x|w2)P(w2)

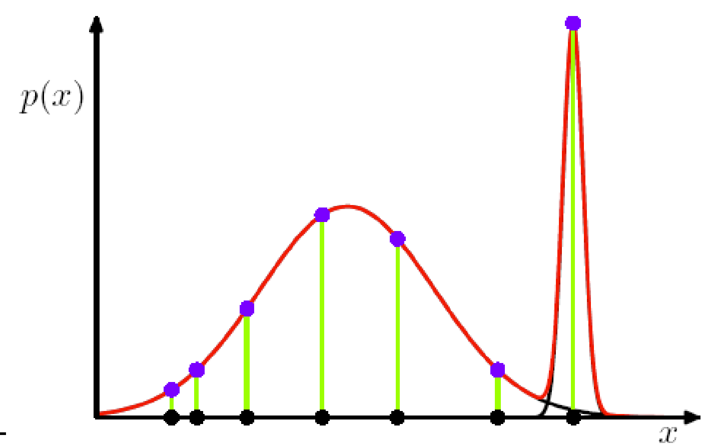
=> Mammals

Gaussian Mixture Models (GMMs)

- Motivation: traditional Guassian models are unimodal, but real data distributions are often multimodal.

- Solution: using mixture models to approach the multimodal distribution in real-world data.





Gaussian Mixture Models (GMMs)

- Multivariate Gaussian Distribution:

$$N(\mathrm{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathrm{x}-\boldsymbol{\mu})^t\Sigma^{-1}(\mathrm{x}-\boldsymbol{\mu})\right],$$

- Gaussian mixture models (GMMs):

$$p(\mathbf{x}|\omega) = \sum_{m=1}^{M} w_m N(\mathbf{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m),$$

where the mixing weight, $\sum_{m=1}^{M} w_m = 1, w_m > 0.$

Generative vs Discriminative Models

- Classifiers broadly fall into two categories: discriminative models and generative models.
- In statistics, a **generative model** is a model for randomly generating observable data, typically given some hidden parameters. It specifies a joint probability distribution over observation and label sequences.
- **Discriminative models** differ from generative models in that they do not allow one to generate observations from joint probability distributions.
- More details:
  – http://en.wikipedia.org/wiki/Generative_model
  – http://en.wikipedia.org/wiki/Discriminative_model

Examples

- Generative models include:
  - Gaussian distribution
  - Gaussian mixture model
  - Multinomial distribution
  - Hidden Markov model
  - Naive Bayes
  - Latent Dirichlet allocation
- Discriminative models include:
  - Linear discriminant analysis
  - Support vector machines
  - Boosting
  - Conditional random fields
  - Logistic regression
  - Neural Networks

GMMs: Parameter Estimation

- Maximum-likelihood estimation (MLE)
  - Suppose that training data D contains n samples, x_1, x_2,…, x_n

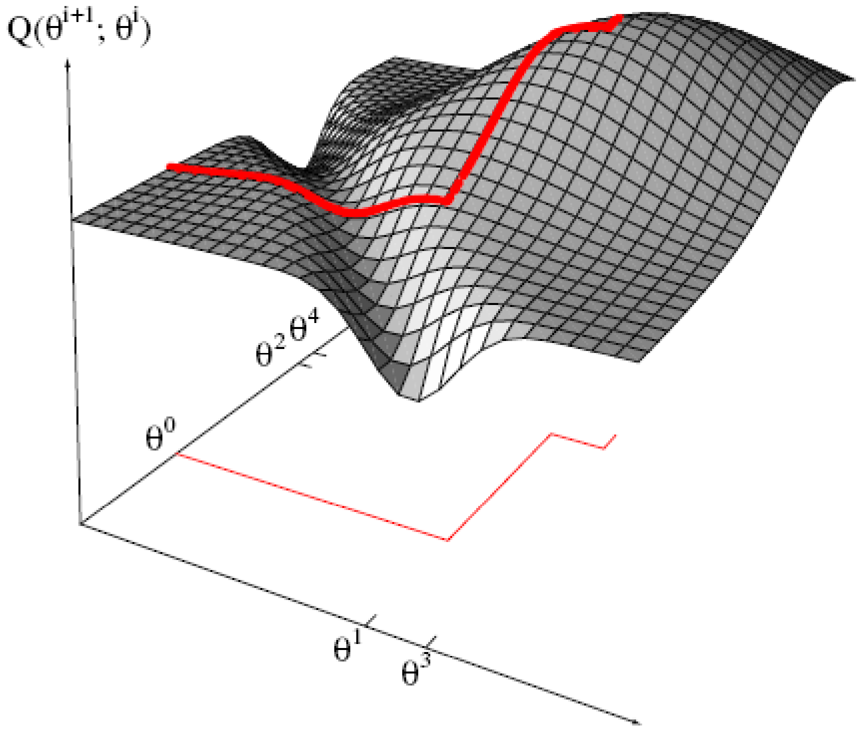
$$P(D \mid \theta) = \prod_{k=1}^{k=n} P(x_k \mid \theta) = F(\theta)$$

$P(D \mid \theta)$ is called the likelihood of θ w.r.t. the set of samples)

  - MLE of parameter θ is, by definition the value that maximizes $P(D \mid \theta)$
  - “It is the value of θ that best agrees with the actually observed training sample”.

Expectation-maximization (EM) Algorithm

- EM is an optimization algorithm that can find the parameter θ that maximizes P(D | θ) according to the MLE.



Two Steps

- Initialize the means ***μ***, covariances **Σ** and mixing coefficients *w*, and evaluate the initial value of the log likelihood.
- E-step: evaluate the responsibilities using the current parameter values.

$$L_{nm} = \frac{w_m N(\mathbf{x}_n; \boldsymbol{\mu}_m, \Sigma_m)}{\sum_{m=1}^{M} w_m N(\mathbf{x}_n; \boldsymbol{\mu}_m, \Sigma_m)}$$

- M-step: Re-estimate the parameters using the current responsibilities.

$$N_m = \sum_{n=1}^{N} L_{nm} \qquad \hat{\boldsymbol{\mu}}_m = \frac{1}{N_m}\sum_{n=1}^{N} L_{nm}\mathbf{x}_n$$

$$w_m = \frac{N_m}{N} \qquad \hat{\Sigma}_m = \frac{1}{N_m}\sum_{n=1}^{N} L_{nm}(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_m)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_m)^{\mathrm{T}}$$

Hidden Markov Models (HMMs)

- Top ten cited papers in IEEE:
  - Lawrence R. Rabiner (February 1989). "A tutorial on Hidden Markov Models and selected applications in speech recognition". *Proceedings of the IEEE* **77** (2): 257-286.
- Markov Chains
- Goal: make a sequence of decisions
  - Processes that unfold in time, states at time t are influenced by a state at time t-1
  - Applications: speech recognition, gesture recognition, parts of speech tagging and DNA sequencing,
  - Any temporal process without memory

    $\omega^T = \{\omega(1), \omega(2), \omega(3), \ldots, \omega(T)\}$ sequence of states

    We might have $\omega^6 = \{\omega 1, \omega 4, \omega 2, \omega 2, \omega 1, \omega 4\}$
  - The system can revisit a state at different steps and not every state need to be visited

First-order Markov models

- Our productions of any sequence is described by the transition probabilities

  $P(\omega_j(t + 1) \mid \omega_i(t)) = a_{ij}$

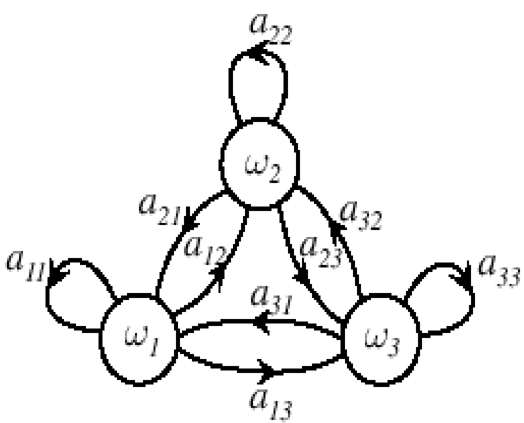


FIGURE 3.8. The discrete states, ω_i, in a basic Markov model are represented by nodes, and the transition probabilities, a_{ij}, are represented by links. In a first-order discrete-time Markov model, at any step t the full system is in a particular state $\omega(t)$. The state at step $t+1$ is a random function that depends solely on the state at step t and the transition probabilities. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Example

$\theta = (a_{ij}, \omega^T)$

$P(\omega^T \mid \theta) = a_{14} \,.\, a_{42} \,.\, a_{22} \,.\, a_{21} \,.\, a_{14} \,.\, P(\omega(1) = \omega_i)$

Example: speech recognition

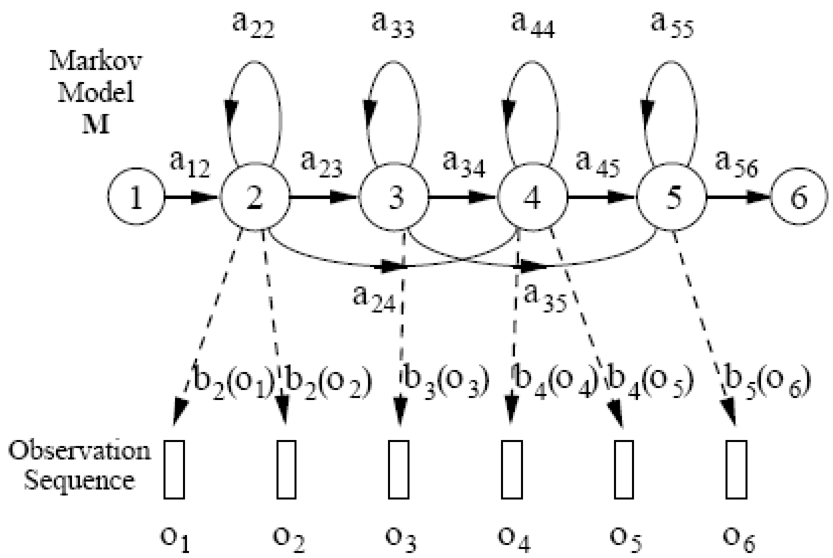
"production of spoken words"

Production of the word: "pattern" represented by phonemes

/p/ /a/ /tt/ /er/ /n/ // (// = silent state)

Transitions from /p/ to /a/, /a/ to /tt/, /tt/ to er/, /er/ to /n/ and /n/ to a silent state

HMMs



- Three problems are associated with this model
  - The evaluation problem
  - The decoding problem
  - The learning problem

The evaluation problem

It is the probability that the model produces a sequence V^T of visible states. It is:

$$P(V^T) = \sum_{r=1}^{r_{\max}} P(V^T \mid \omega_r^T) P(\omega_r^T)$$

where each r indexes a particular sequence $\omega_r^T = \{\omega(1), \omega(2), ..., \omega(T)\}$ of T hidden states.

$$(1) \qquad P(V^T \mid \omega_r^T) = \prod_{t=1}^{t=T} P(v(t) \mid \omega(t))$$

$$(2) \qquad \mathrm{P}(\omega_{\mathrm{r}}^{\mathrm{T}}) = \prod_{t=1}^{t=T} P(\omega(t) \mid \omega(t-1))$$

The evaluation problem…

Using equations (1) and (2), we can write:

$$P(V^T) = \sum_{r=1}^{r_{\max}} \prod_{t=1}^{t=T} P(v(t) \mid \omega(t))\ P(\omega(t) \mid \omega(t-1))$$

Interpretation: The probability that we observe the particular sequence of T visible states V^T is equal to the sum over all r_{max} possible sequences of hidden states of the conditional probability that the system has made a particular transition multiplied by the probability that it then emitted the visible symbol in our target sequence.

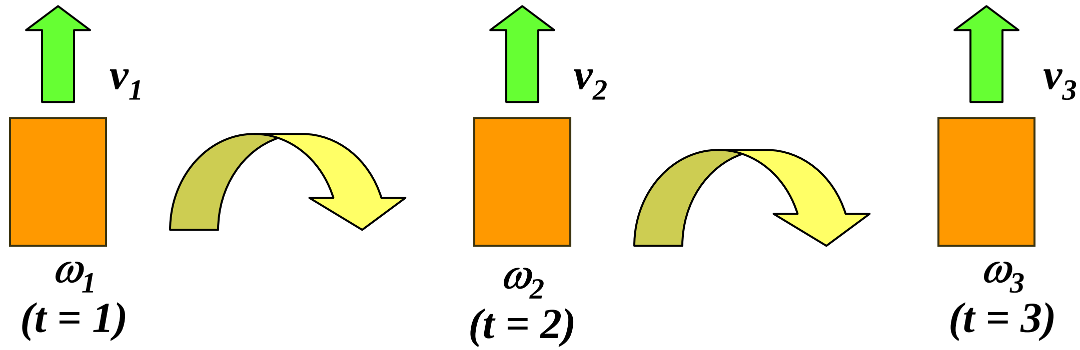
Example: Let $\omega_1, \omega_2, \omega_3$ be the hidden states; v_1, v_2, v_3 be the visible states and $V^3 = \{v_1, v_2, v_3\}$ is the sequence of visible states

$P(\{v_1, v_2, v_3\}) = P(\omega_1).P(v_1 \mid \omega_1).P(\omega_2 \mid \omega_1).P(v_2 \mid \omega_2).P(\omega_3 \mid \omega_2).P(v_3 \mid \omega_3)$

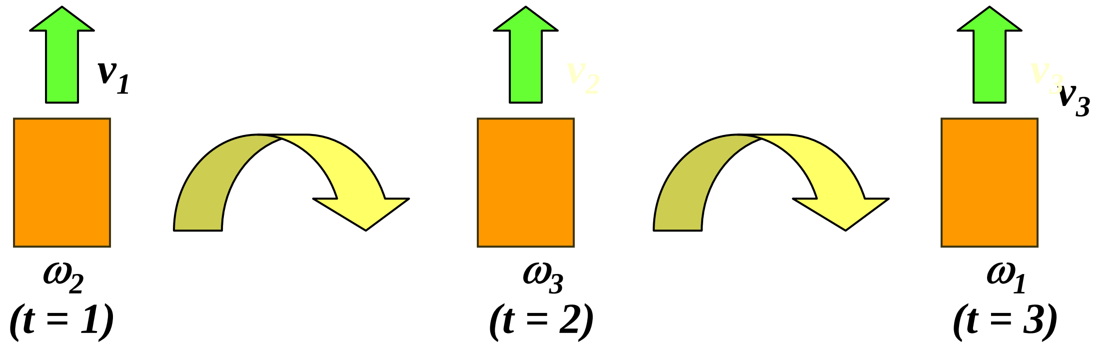
+…+ (possible terms in the sum= all possible (3^3= 27) cases !)

The evaluation problem…

First possibility:



Second Possibility:



$P(\{v_1, v_2, v_3\}) = P(\omega_2).P(v_1 \mid \omega_2).P(\omega_3 \mid \omega_2).P(v_2 \mid \omega_3).P(\omega_1 \mid \omega_3).P(v_3 \mid \omega_1) + \ldots +$

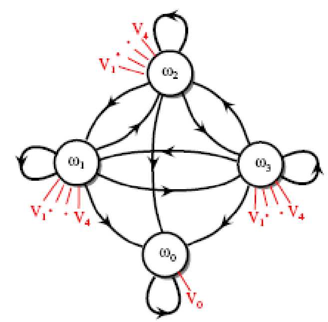
Therefore:

$$P(\{v_1, v_2, v_3\}) = \sum_{\substack{\text{possible sequence} \\ \text{of hidden states}}} \prod_{t=1}^{t=3} P(v(t) \mid \omega(t)).P(\omega(t) \mid \omega(t-1))$$

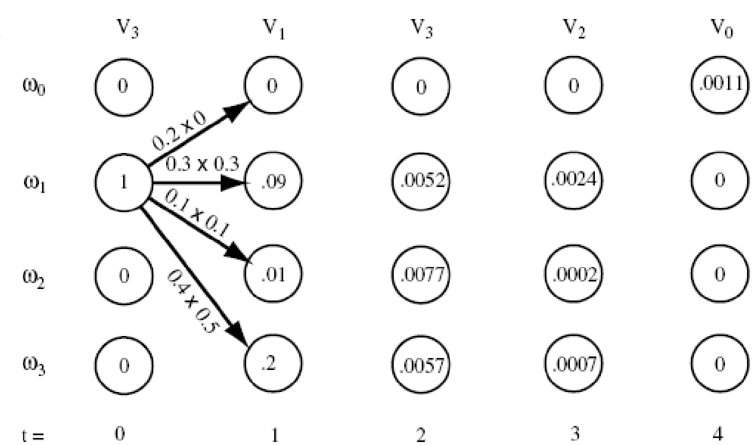
The Forward-backward Algorithm

- This is a belief propagation (BP) algorithm, that passes messages to neighboring states.

initialize $\omega(1), t = 0, a_{ij}, b_{jk}$, visible sequence $\mathbf{V}^T, \alpha(0) = 1$
    for $t \leftarrow t + 1$
$$\alpha_j(t) \leftarrow \sum_{i=1}^{c} \alpha_i(t-1) a_{ij} b_{jk}$$
    until $t = T$
return $P(\mathbf{V}^T) \leftarrow \alpha_0(T)$
end



initialize $\omega(T), t = T, a_{ij}, b_{jk}$, visible sequence V^T
    for $t \leftarrow t - 1$;
$$\beta_j(t) \leftarrow \sum_{i=1}^{c} \beta_i(t+1) a_{ij} b_{jk} v(t+1)$$
    until $t = 1$
return $P(V^T) \leftarrow \beta_i(0)$ for the known initial state
end



Left-right HMMs

- Simple structure with wide applications

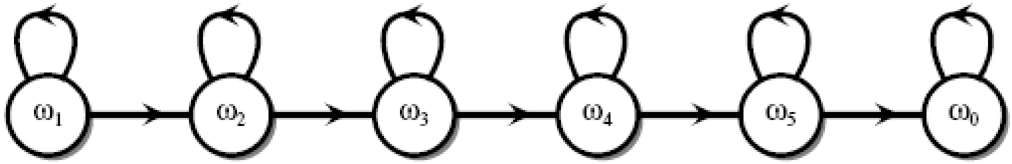


Figure 3.12: A left-to-right HMM commonly used in speech recognition. For instance, such a model could describe the utterance "viterbi," where ω_1 represents the phoneme /v/, ω_2 represents /i/, ..., and ω_0 a final silent state. Such a left-to-right model is more restrictive than the general HMM in Fig. 3.10, and precludes transitions "back" in time.

The decoding problem

- Given a sequence of visible states V^T, the decoding problem is to find the most probable sequence of hidden states.
- This problem can be expressed mathematically as: *find the single "best" state sequence (hidden states).*

$$\hat{\omega}(1),\hat{\omega}(2),\ldots,\hat{\omega}(T) \;\; such \;\; that:$$

$$\hat{\omega}(1),\hat{\omega}(2),\ldots,\hat{\omega}(T) = \underset{\omega(1),\omega(2),\ldots,\omega(T)}{\arg\max} \; P\left[\omega(1),\omega(2),\ldots,\omega(T),v(1),v(2),\ldots,V(T) \,|\, \lambda\right]$$

- Note that the summation disappeared, since we want to find Only one unique best case !

The decoding problem

Where: $\lambda = [\pi, A, B]$

$\pi = P(\omega(1) = \omega)$ *(initial state probability)*

$A = a_{ij} = P(\omega(t+1) = j \mid \omega(t) = i)$

$B = b_{jk} = P(v(t) = k \mid \omega(t) = j)$

In the preceding example, this computation corresponds to the selection of the best path amongst:

$$\{\omega_1(t=1), \omega_2(t=2), \omega_3(t=3)\}, \{\omega_2(t=1), \omega_3(t=2), \omega_1(t=3)\}$$
$$\{\omega_3(t=1), \omega_1(t=2), \omega_2(t=3)\}, \{\omega_3(t=1), \omega_2(t=2), \omega_1(t=3)\}$$
$$\{\omega_2(t=1), \omega_1(t=2), \omega_3(t=3)\}$$

The Viterbi Algorithm

- This is a dynamic programming algorithm for finding the most likely sequence of hidden states.

begin initialize Path $= \{\}, t = 0$

  for $t \leftarrow t + 1$

    $k = 0, \alpha_0 = 0$

    for $k \leftarrow k + 1$

      $\alpha_k(t) \leftarrow b_{jk} v(t) \sum_{i=1}^{c} \alpha_i(t-1) a_{ij}$

    until $k = c$

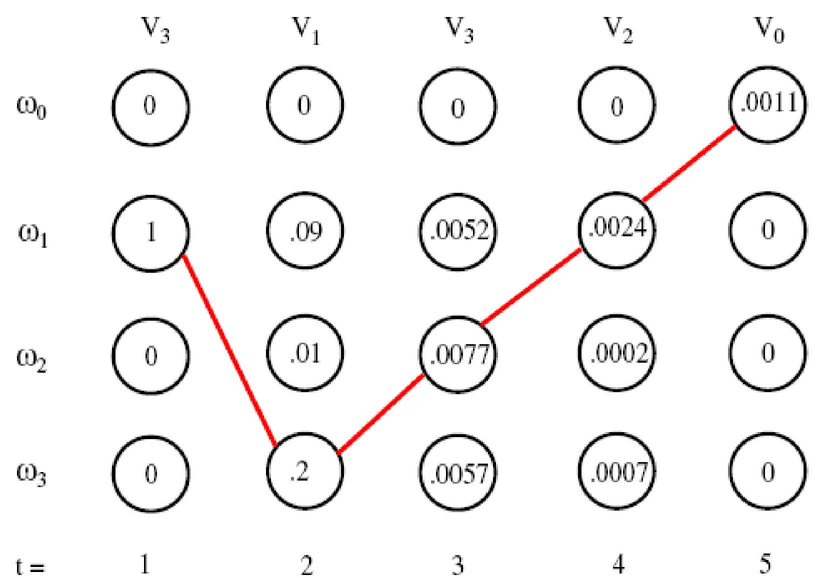
    $j' \leftarrow \arg\max_j \alpha_j(t)$

    AppendTo Path $\omega_{j'}$

  until $t = T$

 return Path

end



Pass the best messages to neighbors!

The learning problem

- This third problem consists of determining a method to adjust the model parameters $\lambda = [\pi, A, B]$ to satisfy a certain optimization criterion. We need to find the best model

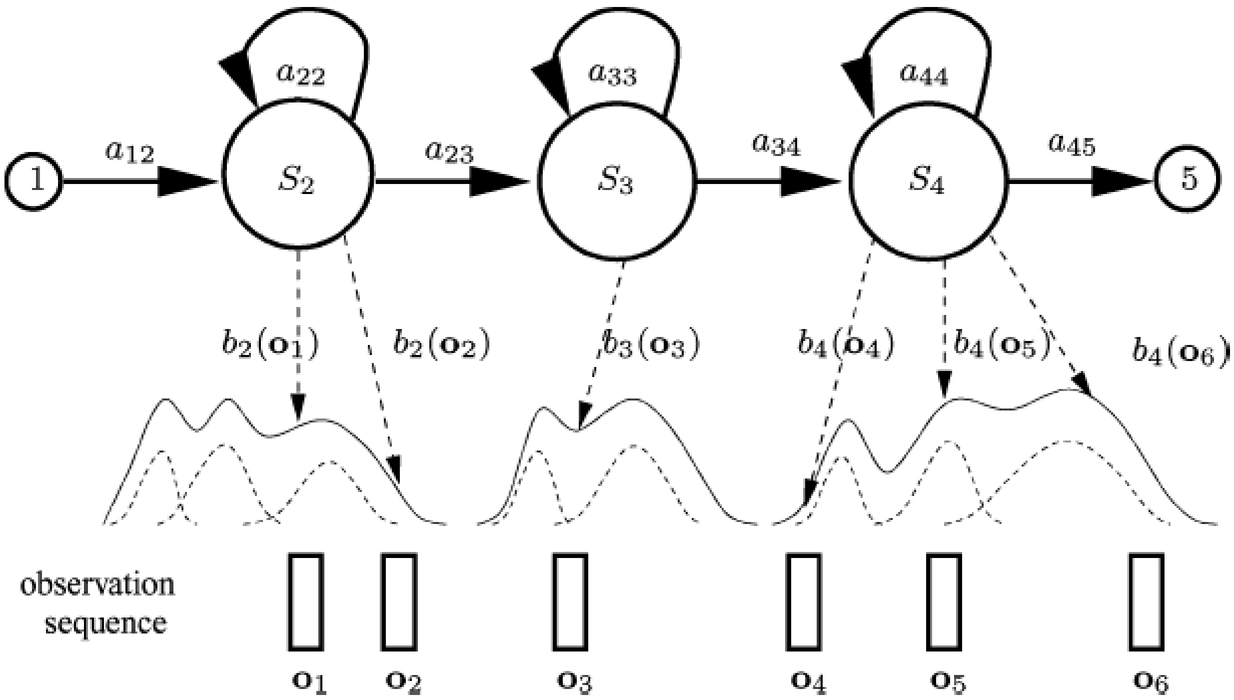
$$\hat{\lambda} = [\hat{\pi}, \hat{A}, \hat{B}]$$

Such that to maximize the probability of the observation sequence:

$$\underset{\lambda}{Max}\, P(V^T \mid \lambda)$$

We use an iterative procedure such as Baum-Welch (EM algorithm) or Gradient to find this local optimum

GMMs as the hidden states!



Learning: Parameter Estimation

Transition:

$$\hat{a}_{ij} = \frac{\sum_{r=1}^{R} \frac{1}{P_r} \sum_{t=1}^{T_r-1} \alpha_i^r(t) a_{ij} b_j(\boldsymbol{o}_{t+1}^r) \beta_j^r(t+1)}{\sum_{r=1}^{R} \frac{1}{P_r} \sum_{t=1}^{T_r} \alpha_i^r(t) \beta_i^r(t)} \qquad \hat{a}_{1j} = \frac{1}{R} \sum_{r=1}^{R} \frac{1}{P_r} \alpha_j^r(1) \beta_j^r(1) \qquad \hat{a}_{iN} = \frac{\sum_{r=1}^{R} \frac{1}{P_r} \alpha_i^r(T) \beta_i^r(T)}{\sum_{r=1}^{R} \frac{1}{P_r} \sum_{t=1}^{T_r} \alpha_i^r(t) \beta_i^r(t)}$$

Updating weights:

$$L_j(t) = \frac{1}{P} \alpha_j(t) \beta_j(t),$$

Update mean vector, covariance matrix and mixir weights:

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T_r} L_{jm}^r(t) \mathbf{o}_t^r}{\sum_{r=1}^{R} \sum_{t=1}^{T_r} L_{jm}^r(t)},$$

$$\hat{\boldsymbol{\Sigma}}_{jm} = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T_r} L_{jm}^r(t) (\mathbf{o}_t^r - \hat{\mu}_{jm})(\mathbf{o}_t^r - \hat{\mu}_{jm})'}{\sum_{r=1}^{R} \sum_{t=1}^{T_r} L_{jm}^r(t)},$$

$$\hat{w}_{jm} = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T_r} L_{jm}^r(t)}{\sum_{r=1}^{R} \sum_{t=1}^{T_r} L_j^r(t)}.$$

Summary

- K-nearest-neighbor (non-parametric method)
- Naïve Bayes classifier (density modeling)
- Gaussian mixture models (density modeling)
- Hidden Markov models (sequential data)