Tutorial on Statistical Machine Learning

with Applications to Multimodal Processing

Samy Bengio

IDIAP Research Institute
Martigny, Switzerland
bengio@idiap.ch
http://www.idiap.ch/~bengio

October 7th, 2005 - ICMI, Trento

Outline of the Tutorial

Outline

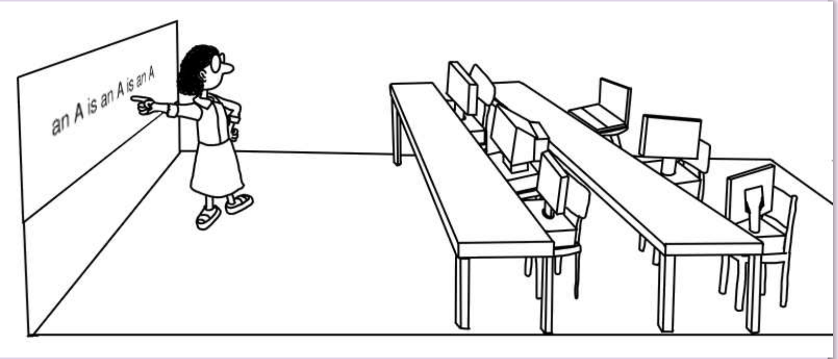
Updated slides

`http://www.idiap.ch/~bengio/icmi2005.pdf`

Part I

Introduction

What is Machine Learning? (Graphical View)

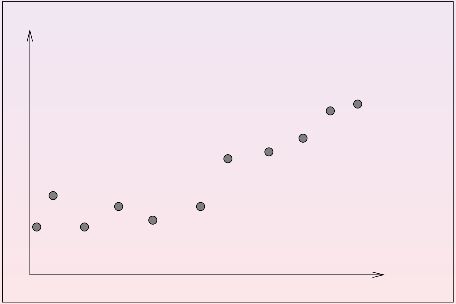


What is Machine Learning?

- Learning is an essential human property
- Learning means changing in order to be better (according to a given criterion) when a similar situation arrives
- Learning IS NOT learning by heart
- Any computer can learn by heart, the difficulty is to generalize a behavior to a novel situation

Why Learning is Difficult?

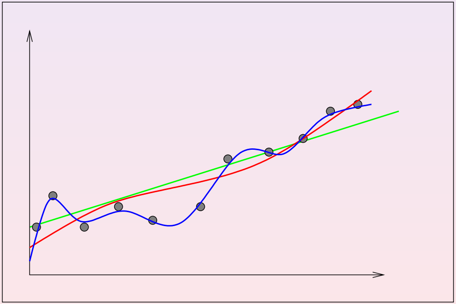
- Given a finite amount of training data, you have to derive a relation for an infinite domain
- In fact, there is an infinite number of such relations



- How should we draw the relation?

Why Learning is Difficult? (2)

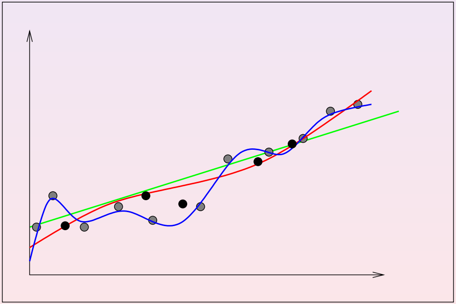
- Given a finite amount of training data, you have to derive a relation for an infinite domain
- In fact, there is an infinite number of such relations



- Which relation is the most appropriate?

Why Learning is Difficult? (3)

- Given a finite amount of training data, you have to derive a relation for an infinite domain
- In fact, there is an infinite number of such relations



- ... the hidden test points...

Occam's Razor's Principle

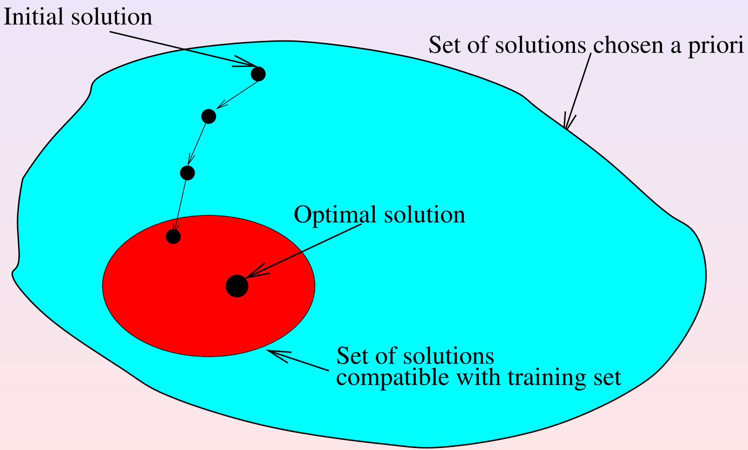
- William of Occam: Monk living in the 14th century
- Principle of Parcimony:

  One should not increase, beyond what is necessary, the number of entities required to explain anything

- When many solutions are available for a given problem, we should select the simplest one
- But what do we mean by simple?
- We will use prior knowledge of the problem to solve to define what is a simple solution

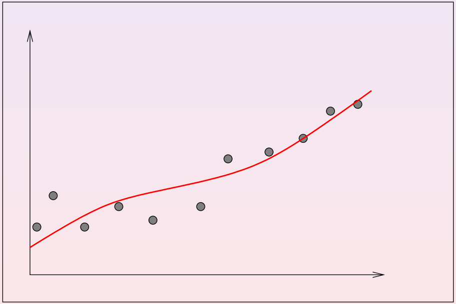
  Example of a prior: smoothness

Learning as a Search Problem



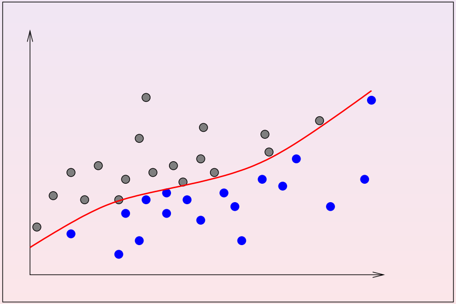
Types of Problems

- There are 3 kinds of problems:
  - regression



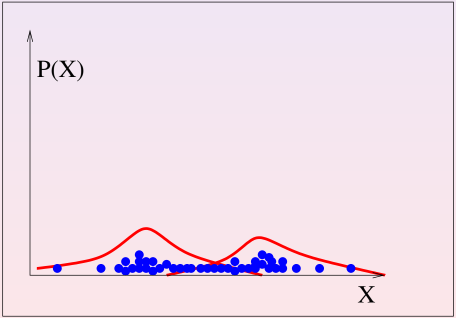
Types of Problems

- There are 3 kinds of problems:
  - regression, classification



Types of Problems

- There are 3 kinds of problems:
    - regression, classification, density estimation



Applications

- Vision Processing
  - Face detection/verification
  - Handwritten recognition
- Speech Processing
  - Phoneme/Word/Sentence/Person recognition
- Others
  - Finance: asset prediction, portfolio and risk management
  - Telecom: traffic prediction
  - Data mining: make use of huge datasets kept by large corporations
  - Games: Backgammon, go
  - Control: robots
- ... and plenty of others of course!

Part II

Statistical Learning Theory

The Data

Available training data

- Let $Z_1, Z_2, \cdots, Z_n$ be an n-tuple random sample of an unknown distribution of density $p(z)$.
- All Z_i are independently and identically distributed (iid).
- Let D_n be a particular instance $= \{z_1, z_2, \cdots, z_n\}$.

Various forms of the data

- Classification: $Z = (X, Y) \in \mathbb{R}^d \times \{-1, 1\}$
  objective: given a new x, estimate $P(Y|X = x)$
- Regression: $Z = (X, Y) \in \mathbb{R}^d \times \mathbb{R}$
  objective: given a new x, estimate $E[Y|X = x]$
- Density estimation: $Z \in \mathbb{R}^d$
  objective: given a new z, estimate $p(z)$

The Function Space

Learning: search for a good function in a function space $\mathcal{F}$

Examples of functions $f(\cdot;\theta) \in \mathcal{F}$:

- Regression:

$$\hat{y} = f(x; a, b, c) = a \cdot x^2 + b \cdot x + c$$

- Classification:

$$\hat{y} = f(x; a, b, c) = \mathrm{sign}(a \cdot x^2 + b \cdot x + c)$$

- Density estimation

$$\hat{p}(z) = f(z; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{|z|}{2}} \sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(z-\mu)^T \Sigma^{-1} (z-\mu)\right)$$

The Loss Function

Learning: search for a good function in a function space $\mathcal{F}$

Examples of loss functions $L : \mathcal{Z} \times \mathcal{F}$

- Regression:

$$L(z, f) = L((x, y), f) = (f(x) - y)^2$$

- Classification:

$$L(z, f) = L((x, y), f) = \begin{cases} 0 & \text{if } f(x) = y \\ 1 & \text{otherwise} \end{cases}$$

- Density estimation:

$$L(z, f) = -\log p(z)$$

The Risk and the Empirical Risk

Learning: search for a good function in a function space $\mathcal{F}$

- Minimize the Expected Risk on $\mathcal{F}$, defined for a given f as

$$R(f) = E_Z[L(z, f)] = \int_Z L(z, f)p(z)dz$$

- Induction Principle:
  - select $f^* = \arg\min_{f \in \mathcal{F}} R(f)$
  - problem: $p(z)$ is unknown!!!
- Empirical Risk:

$$\hat{R}(f, D_n) = \frac{1}{n}\sum_{i=1}^{n} L(z_i, f)$$

The Risk and the Empirical Risk

- The empirical risk is an unbiased estimate of the risk:

$$E[\hat{R}(f, D)] = R(f)$$

- The principle of empirical risk minimization:

$$f^*(D_n) = \arg\min_{f \in \mathcal{F}} \hat{R}(f, D_n)$$

- Training error:

$$\hat{R}(f^*(D_n), D_n) = \min_{f \in \mathcal{F}} \hat{R}(f, D_n)$$

Is the training error a biased estimate of the risk?

The Training Error

Is the training error a biased estimate of the risk? yes.

$$E[R(f^*(D_n)) - \hat{R}(f^*(D_n), D_n)] \geq 0$$

- The solution $f^*(D_n)$ found by minimizing the training error is better on D_n than on any other set $D_n^{'}$ drawn from $p(Z)$.
- Can we bound the difference between the training error and the generalization error?

$$|R(f^*(D_n)) - \hat{R}(f^*(D_n), D_n)| \leq ?$$

- Answer: under certain conditions on $\mathcal{F}$, yes.
- These conditions depend on the notion of capacity h of $\mathcal{F}$.

The Capacity

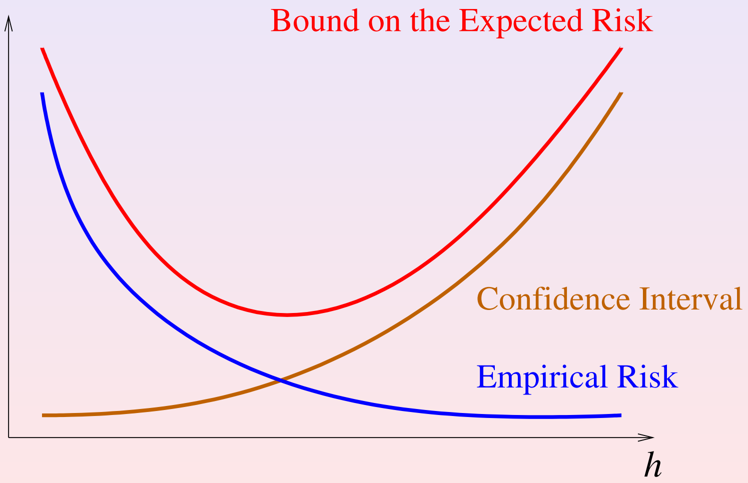
- The capacity $h(\mathcal{F})$ is a measure of its size, or complexity.
- Classification:

  The capacity $h(\mathcal{F})$ is the largest n such that there exist a set of examples D_n such that one can always find an $f \in \mathcal{F}$ which gives the correct answer for all examples in D_n, for any possible labeling.

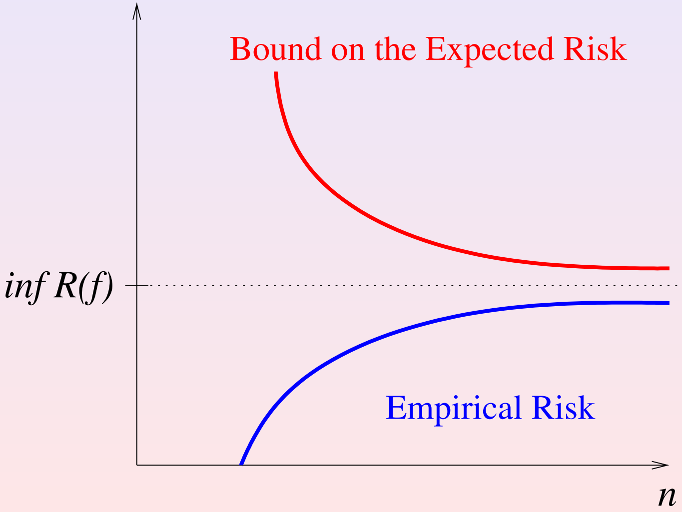
- Regression and density estimation: capacity exists also, but more complex to derive (for instance, we can always reduce a regression problem to a classification problem).
- Bound on the expected risk: let $\tau = \sup L - \inf L$. $\forall \eta$ we have

$$P\left(\sup_{f \in \mathcal{F}} |R(f) - \hat{R}(f, D_n)| \leq 2\tau \sqrt{\frac{h\left(\ln \frac{2n}{h} + 1\right) - \ln \frac{\eta}{9}}{n}}\right) \geq 1 - \eta$$

Theoretical Curves



Theoretical Curves



Methodology

- First: identify the goal! It could be
    1. to give the best model you can obtain given a training set?
    2. to give the expected performance of a model obtained by empirical risk minimization given a training set?
    3. to give the best model and its expected performance that you can obtain given a training set?
- If the goal is (1): use need to do model selection
- If the goal is (2), you need to estimate the risk
- If the goal is (3): use need to do both!
- There are various methods that can be used for either risk estimation or model selection:
    - simple validation
    - cross validation (k-fold, leave-one-out)

Model Selection - Validation

- Select a family of functions with hyper-parameter θ
- Divide your training set D_n into two parts
  - $D^{tr} = \{z_1, z_2, \cdots, z_{tr}\}$
  - $D^{va} = \{z_{tr+1}, z_{tr+2}, \cdots, z_{tr+va}\}$
  - $tr + va = n$
- For each value θ_m of the hyper-parameter θ
  - select $f^*_{\theta_m}(D^{tr}) = \arg\min_{f \in \mathcal{F}_{\theta_m}} \hat{R}(f, D^{tr})$
  - estimate $R(f^*_{\theta_m})$ with $\hat{R}(f^*_{\theta_m}, D^{va}) = \dfrac{1}{va} \sum_{z_i \in D^{va}} L(z_i, f^*_{\theta_m}(D^{tr}))$
- select $\theta^*_m = \arg\min_{\theta_m} R(f^*_{\theta_m})$
- return $f^*(D_n) = \arg\min_{f \in \mathcal{F}_{\theta^*_m}} \hat{R}(f, D_n)$

Model Selection - Cross-validation

- Select a family of functions with hyper-parameter θ
- Divide your training set D_n into K distinct and equal parts $D^1, \cdots, D^k, \cdots, D^K$
- For each value θ_m of the hyper-parameter θ
  - For each part D^k (and its counterpart $\bar{D}^k$)
    - select $f^*_{\theta_m}(\bar{D}^k) = \arg \min_{f \in \mathcal{F}_{\theta_m}} \hat{R}(f, \bar{D}^k)$
    - estimate $R(f^*_{\theta_m}(\bar{D}^k))$ with $\hat{R}(f^*_{\theta_m}(\bar{D}^k), D^k) = \frac{1}{|D^k|} \sum_{z_i \in D^k} L(z_i, f^*_{\theta_m}(\bar{D}^k))$
  - estimate $R(f^*_{\theta_m}(D_n))$ with $\frac{1}{K} \sum_k R(f^*_{\theta_m}(\bar{D}^k))$
- select $\theta^*_m = \arg \min_{\theta_m} R(f^*_{\theta_m}(D))$
- return $f^*(D_n) = \arg \min_{f \in \mathcal{F}_{\theta^*_m}} \hat{R}(f, D_n)$

Estimation of the Risk - Validation

- Divide your training set D_n into two parts
  - $D^{tr} = \{z_1, z_2, \cdots, z_{tr}\}$
  - $D^{te} = \{z_{tr+1}, z_{tr+2}, \cdots, z_{tr+te}\}$
  - $tr + te = n$
- select $f^*(D^{tr}) = \arg\min_{f \in \mathcal{F}} \hat{R}(f, D^{tr})$

  (this optimization process could include model selection)

- estimate $R(f^*(D^{tr}))$ with

  $$\hat{R}(f^*(D^{tr}), D^{te}) = \frac{1}{te} \sum_{z_i \in D^{te}} L(z_i, f^*(D^{tr}))$$

Estimation of the Risk - Cross-validation

- Divide your training set D_n into K distinct and equal parts $D^1, \cdots, D^k, \cdots, D^K$
- For each part D^k
  - let $\bar{D}^k$ be the set of examples that are in D_n but not in D^k
  - select $f^*(\bar{D}^k) = \arg\min_{f \in \mathcal{F}} \hat{R}(f, \bar{D}^k)$

    (this process could include model selection)
  - estimate $R(f^*(\bar{D}^k))$ with
    $$\hat{R}(f^*(\bar{D}^k), D^k) = \frac{1}{|D^k|} \sum_{z_i \in D^k} L(z_i, f^*(\bar{D}^k))$$
- estimate $R(f^*(D_n))$ with $\displaystyle\frac{1}{K} \sum_k R(f^*(\bar{D}^k))$
- When $k = n$: leave-one-out cross-validation

Estimation of the Risk and Model Selection

- When you want both the best model and its expected risk.
- You then need to merge the methods already presented.
  For instance:
  - train-validation-test: 3 separate data sets are necessary
  - cross-validation + test: cross-validate on train set, then test on separate set
  - double-cross-validation: for each subset, need to do a second cross-validation with the $K-1$ other subsets
- Other important methodological aspects:
  - compare your results with other methods!!!!
  - use statistical tests to verify significance
  - verify your model on more than one datasets

Train - Validation - Test

- Select a family of functions with hyper-parameter θ
- Divide your training set D_n into three parts D^{tr}, D^{va}, and D^{te}
- For each value θ_m of the hyper-parameter θ
  - select $f^*_{\theta_m}(D^{tr}) = \arg \min_{f \in \mathcal{F}_{\theta_m}} \hat{R}(f, D^{tr})$
  - let $\hat{R}(f^*_{\theta_m}(D^{tr}), D^{va}) = \frac{1}{va} \sum_{z_i \in D^{va}} L(z_i, f^*_{\theta_m}(D^{tr}))$
- select $\theta^*_m = \arg \min_{\theta_m} \hat{R}(f^*_{\theta_m}(D^{tr}), D^{va})$
- select $f^*(D^{tr} \cup D^{va}) = \arg \min_{f \in \mathcal{F}_{\theta^*_m}} \hat{R}(f, D^{tr} \cup D^{va})$
- estimate $R(f^*(D^{tr} \cup D^{va}))$ with $\frac{1}{te} \sum_{z_i \in D^{te}} L(z_i, f^*(D^{tr} \cup D^{va}))$

Cross-validation + Test

- Select a family of functions with hyper-parameter θ
- Divide you dataset D_n into two parts:

  a training set D^{tr} and a test set D^{te}

- For each value θ_m of the hyper-parameter θ

  estimate $R(f^*_{\theta_m}(D^{tr}))$ with D^{tr} using cross-validation

- select $\theta^*_m = \arg\min_{\theta_m} R(f^*_{\theta_m}(D^{tr}))$
- retrain $f^*(D^{tr}) = \arg\min_{f \in \mathcal{F}_{\theta^*_m}} \hat{R}(f, D^{tr})$
- estimate $R(f^*(D^{tr}))$ with $\displaystyle\frac{1}{te} \sum_{z_i \in D^{te}} L(z_i, f^*(D^{tr}))$

Examples of Known Models

- Multi-Layer Perceptrons (regression, classification)
- Radial Basis Functions (regression, classification)
- Support Vector Machines (classification, regression)
- Gaussian Mixture Models (density estimation, classification)
- Hidden Markov Models (density estimation, classification)
- Graphical Models (density estimation, classification)
- AdaBoost and Bagging (classification, regression, density estimation)
- Decision Trees (classification, regression)

Part III

EM and Gaussian Mixture Models

Reminder: Basics on Probabilities

A few basic equalities that are often used:

1. (conditional probabilities)

$$P(A, B) = P(A|B) \cdot P(B)$$

2. (Bayes rule)

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

3. If $(\bigcup B_i = \Omega)$ and $\forall i, j \neq i \ (B_i \bigcap B_j = \emptyset)$ then

$$P(A) = \sum_i P(A, B_i)$$

What is a Gaussian Mixture Model

- A Gaussian Mixture Model (GMM) is a distribution
- The likelihood given a Gaussian distribution is

$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{|x|}{2}}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

  where μ is the mean and Σ is the covariance matrix of the Gaussian. Σ is often diagonal.
- The likelihood given a GMM is

$$p(x) = \sum_{i=1}^{N} w_i \cdot \mathcal{N}(x;\mu,\Sigma)$$

  where N is the number of Gaussians and w_i is the weight of Gaussian i, with

$$\sum w_i = 1 \text{ and } \forall i : w_i \geq 0$$

Characteristics of a GMM

- While Multi-Layer Perceptrons are universal approximators of functions,
- GMMs are universal approximators of densities.

  (as long as there are enough Gaussians of course)
- Even diagonal GMMs are universal approximators.
- Full rank GMMs are not easy to handle: number of parameters is the square of the number of dimensions.
- GMMs can be trained by maximum likelihood using an efficient algorithm: Expectation-Maximization.

Basics of Expectation-Maximization

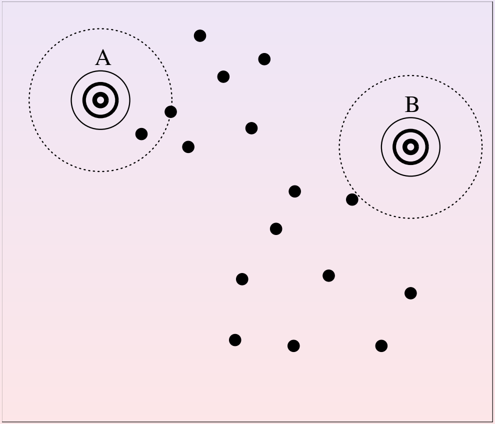
- Objective: maximize the likelihood $p(X;\theta)$ of the data X drawn from an unknown distribution, given the model parameterized by θ:

$$\theta^* = \arg\max_\theta p(X|\theta) = \arg\max_\theta \prod_{p=1}^{n} p(x_p|\theta)$$

- Basic ideas of EM:
  - Introduce a hidden variable such that *its knowledge would simplify the maximization of $p(X;\theta)$*
  - At each iteration of the algorithm:
    - E-Step: estimate the distribution of the hidden variable given the data and the current value of the parameters
    - M-Step: modify the parameters in order to maximize the joint distribution of the data and the hidden variable

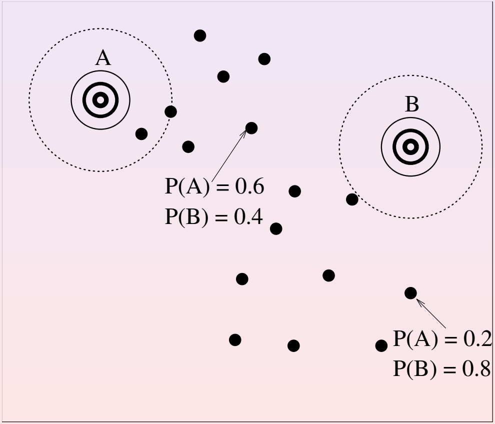
EM for GMM (Graphical View, 1)

Hidden variable: for each point, which Gaussian generated it?



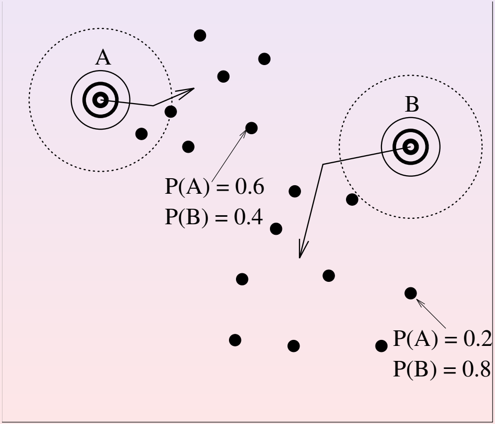
EM for GMM (Graphical View, 2)

E-Step: for each point, estimate the probability the each Gaussian generated it



EM for GMM (Graphical View, 3)

M-Step: modify the parameters according to the hidden variable to maximize the likelihood of the data (and the hidden variable)



EM: More Formally

- Let us call the hidden variable Q.
- Let us consider the following auxiliary function:

$$A(\theta, \theta^s) = E_Q[\log p(X, Q|\theta)|X, \theta^s]$$

- It can be shown that maximizing A

$$\theta^{s+1} = \arg\max_{\theta} A(\theta, \theta^s)$$

always increases the likelihood of the data $p(X|\theta^{s+1})$, and a maximum of A corresponds to a maximum of the likelihood.

EM for GMM: Hidden Variable

- For GMM, the hidden variable Q will describe which Gaussian generated each example.
- If Q was observed, then it would be simple to maximize the likelihood of the data: simply estimate the parameters Gaussian by Gaussian
- Moreover, we will see that we can easily estimate Q
- Let us first write the mixture of Gaussian model for one x_i:

$$p(x_i|\theta) = \sum_{j=1}^{N} P(j|\theta)p(x_i|j,\theta)$$

- Let us now introduce the following indicator variable:

$$q_{i,j} = \begin{cases} 1 & \text{if Gaussian } j \text{ emitted } x_i \\ 0 & \text{otherwise} \end{cases}$$

EM for GMM: Auxiliary Function

- We can now write the joint likelihood of all the X and Q:

$$p(X, Q|\theta) = \prod_{i=1}^{n} \prod_{j=1}^{N} P(j|\theta)^{q_{i,j}} p(x_i|j, \theta)^{q_{i,j}}$$

- which in log gives

$$\log p(X, Q|\theta) = \sum_{i=1}^{n} \sum_{j=1}^{N} q_{i,j} \log P(j|\theta) + q_{i,j} \log p(x_i|j, \theta)$$

EM for GMM: Auxiliary Function

Let us now write the corresponding auxiliary function:

$$
\begin{aligned}
A(\theta, \theta^s) &= E_Q[\log p(X, Q|\theta)|X, \theta^s] \\
&= E_Q\left[\sum_{i=1}^{n}\sum_{j=1}^{N} q_{i,j} \log P(j|\theta) + q_{i,j} \log p(x_i|j, \theta)|X, \theta^s\right] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{N} E_Q[q_{i,j}|X, \theta^s] \log P(j|\theta) + E_Q[q_{i,j}|X, \theta^s] \log p(x_i|j, \theta)
\end{aligned}
$$

EM for GMM: Update Rules

Means $$\hat{\mu}_j = \frac{\displaystyle\sum_{i=1}^{n} x_i \cdot P(j|x_i, \theta^s)}{\displaystyle\sum_{i=1}^{n} P(j|x_i, \theta^s)}$$

Variances $$(\hat{\sigma}_j)^2 = \frac{\displaystyle\sum_{i=1}^{n} (x_i - \mu_j)^2 \cdot P(j|x_i, \theta^s)}{\displaystyle\sum_{i=1}^{n} P(j|x_i, \theta^s)}$$

Weights: $$\hat{w}_j = \frac{1}{n} \sum_{i=1}^{n} P(j|x_i, \theta^s)$$

Initialization

- EM is an iterative procedure that is very sensitive to initial conditions!
- Start from trash $\rightarrow$ end up with trash.
- Hence, we need a good and fast initialization procedure.
- Often used: K-Means.
- Other options: hierarchical K-Means, Gaussian splitting.

Part IV

Hidden Markov Models

Markov Models

- Stochastic process of a temporal sequence: the probability distribution of the variable q at time t depends on the variable q at times $t-1$ to 1.

$$P(q_1, q_2, \ldots, q_T) = P(q_1^T) = P(q_1) \prod_{t=2}^{T} P(q_t | q_1^{t-1})$$

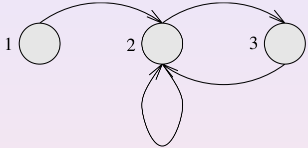
- First Order Markov Process:

$$P(q_t | q_1^{t-1}) = P(q_t | q_{t-1})$$

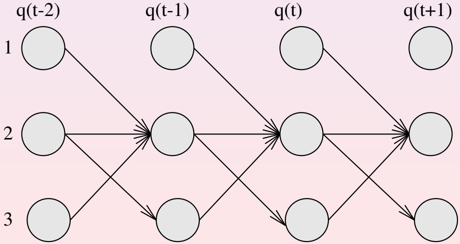
- Markov Model: model of a Markovian process with discrete states.
- Hidden Markov Model: Markov Model whose state is not observed, but of which one can observe a manifestation (a variable x_t which depends only on q_t).

Markov Models (Graphical View)

- A Markov model:

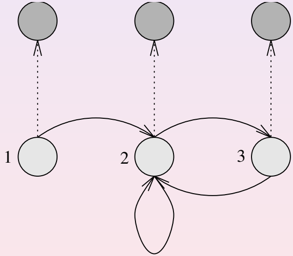


- A Markov model unfolded in time:

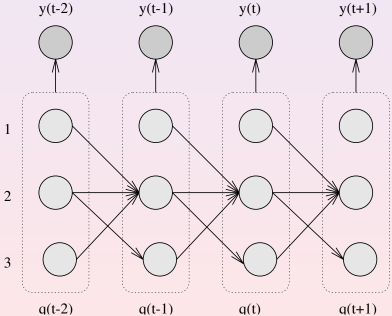


Hidden Markov Models

- A hidden Markov model:



- A hidden Markov model unfolded in time:



Elements of an HMM

- A finite number of states N.
- Transition probabilities between states, which depend only on previous state: $P(q_t{=}i|q_{t-1}{=}j, \theta)$.
- Emission probabilities, which depend only on the current state: $p(x_t|q_t{=}i, \theta)$ (where x_t is observed).
- Initial state probabilities: $P(q_0 = i|\theta)$.
- Each of these 3 sets of probabilities have parameters θ to estimate.

The 3 Problems of HMMs

- The HMM model gives rise to 3 different problems:
  - Given an HMM parameterized by θ, can we compute the likelihood of a sequence $X = x_1^T = \{x_1, x_2, \ldots, x_T\}$:

$$p(x_1^T|\theta)$$

  - Given an HMM parameterized by θ and a set of sequences D_n, can we select the parameters θ^* such that:

$$\theta^* = \arg\max_{\theta} \prod_{p=1}^{n} p(X(p)|\theta)$$

  - Given an HMM parameterized by θ, can we compute the optimal path Q through the state space given a sequence X:

$$Q^* = \arg\max_{Q} p(X, Q|\theta)$$

HMMs as Generative Processes

HMMs can be use to generate sequences:

- Let us define a set of starting states with initial probabilities $P(q_0 = i)$.
- Let us also define a set of final states.
- Then for each sequence to generate:
  1. Select an initial state j according to $P(q_0)$.
  2. Select the next state i according to $P(q_t = i | q_{t-1} = j)$.
  3. Emit an output according to the emission distribution $P(x_t | q_t = i)$.
  4. If i is a final state, then stop, otherwise loop to step 2.

Markovian Assumptions

- Emissions: the probability to emit x_t at time t in state $q_t = i$ does not depend on anything else:

$$p(x_t | q_t = i, q_1^{t-1}, x_1^{t-1}) = p(x_t | q_t = i)$$

- Transitions: the probability to go from state j to state i at time t does not depend on anything else:

$$P(q_t = i | q_{t-1} = j, q_1^{t-2}, x_1^{t-1}) = P(q_t = i | q_{t-1} = j)$$

- Moreover, this probability does not depend on time t:

$$P(q_t = i | q_{t-1} = j) \text{ is the same for all } t$$

we say that such Markov models are homogeneous.

Derivation of the Forward Variable α

the probability of having generated the sequence x_1^t and being in state i at time t:

$$
\begin{aligned}
\alpha(i,t) &\stackrel{\text{def}}{=} p(x_1^t, q_t = i) \\
&= p(x_t | x_1^{t-1}, q_t = i) p(x_1^{t-1}, q_t = i) \\
&= p(x_t | q_t = i) \sum_j p(x_1^{t-1}, q_t = i, q_{t-1} = j) \\
&= p(x_t | q_t = i) \sum_j P(q_t = i | x_1^{t-1}, q_{t-1} = j) p(x_1^{t-1}, q_{t-1} = j) \\
&= p(x_t | q_t = i) \sum_j P(q_t = i | q_{t-1} = j) p(x_1^{t-1}, q_{t-1} = j) \\
&= p(x_t | q_t = i) \sum_j P(q_t = i | q_{t-1} = j) \alpha(j, t-1)
\end{aligned}
$$

From α to the Likelihood

- Reminder: $\alpha(i,t) \stackrel{\text{def}}{=} p(x_1^t, q_t = i)$
- Initial condition:

$$\alpha(i,0) = P(q_0 = i) \rightarrow \text{ prior probabilities of each state } i$$

- Then let us compute $\alpha(i,t)$ for each state i and each time t of a given sequence x_1^T
- Afterward, we can compute the likelihood as follows:

$$\begin{aligned} p(x_1^T) &= \sum_i p(x_1^T, q_T = i) \\ &= \sum_i \alpha(i, T) \end{aligned}$$

- Hence, to compute the likelihood $p(x_1^T)$, we need $\mathcal{O}(N^2 \cdot T)$ operations, where N is the number of states

EM Training for HMM

- For HMM, the hidden variable Q will describe in which state the HMM was for each observation x_t of a sequence X.
- The joint likelihood of all sequences $X(l)$ and the hidden variable Q is then:

$$p(X, Q|\theta) = \prod_{l=1}^{n} p(X(l), Q|\theta)$$

- Let us introduce the following indicator variable:

$$q_{i,t} = \begin{cases} 1 & \text{if } q_t = i \\ 0 & \text{otherwise} \end{cases}$$

Joint Likelihood

$$
\begin{aligned}
p(X, Q|\theta) &= \prod_{l=1}^{n} p(X(l), Q|\theta) \\
&= \prod_{l=1}^{n} \left(\prod_{i=1}^{N} P(q_0 = i)^{q_{i,0}} \right) \cdot \\
&\quad \prod_{t=1}^{T_l} \prod_{i=1}^{N} p(x_t(l)|q_t = i)^{q_{i,t}} \prod_{j=1}^{N} P(q_t = i|q_{t-1} = j)^{q_{i,t} \cdot q_{j,t-1}}
\end{aligned}
$$

Joint Log Likelihood

$$
\begin{aligned}
\log p(X, Q|\theta) \quad = \quad & \sum_{l=1}^{n} \sum_{i=1}^{N} q_{i,0} \log P(q_0 = i) + \\
& \sum_{l=1}^{n} \sum_{t=1}^{T_l} \sum_{i=1}^{N} q_{i,t} \log p(x_t(l)|q_t = i) + \\
& \sum_{l=1}^{n} \sum_{t=1}^{T_l} \sum_{i=1}^{N} \sum_{j=1}^{N} q_{i,t} \cdot q_{j,t-1} \log P(q_t = i|q_{t-1} = j)
\end{aligned}
$$

Auxiliary Function

Let us now write the corresponding auxiliary function:

$$
\begin{aligned}
A(\theta, \theta^s) &= E_Q[\log p(X, Q|\theta)|X, \theta^s] \\
&= \sum_{l=1}^{n} \sum_{i=1}^{N} E_Q[q_{i,0}|X, \theta^s] \log P(q_0 = i) + \\
&\quad \sum_{l=1}^{n} \sum_{t=1}^{T_l} \sum_{i=1}^{N} E_Q[q_{i,t}|X, \theta^s] \log p(x_t(l)|q_t = i) + \\
&\quad \sum_{l=1}^{n} \sum_{t=1}^{T_l} \sum_{i=1}^{N} \sum_{j=1}^{N} E_Q[q_{i,t} \cdot q_{j,t-1}|X, \theta^s] \log P(q_t = i|q_{t-1} = j)
\end{aligned}
$$

From now on, let us forget about index l for simplification.

Derivation of the Backward Variable β

the probability to generate the rest of the sequence x_{t+1}^T given that we are in state i at time t

$$\begin{aligned}
\beta(i,t) &\stackrel{\text{def}}{=} p(x_{t+1}^T | q_t{=}i) \\
&= \sum_j p(x_{t+1}^T, q_{t+1}{=}j | q_t{=}i) \\
&= \sum_j p(x_{t+1} | x_{t+2}^T, q_{t+1}{=}j, q_t{=}i) p(x_{t+2}^T, q_{t+1}{=}j | q_t{=}i) \\
&= \sum_j p(x_{t+1} | q_{t+1}{=}j) p(x_{t+2}^T | q_{t+1}{=}j, q_t{=}i) P(q_{t+1}{=}j | q_t{=}i) \\
&= \sum_j p(x_{t+1} | q_{t+1}{=}j) p(x_{t+2}^T | q_{t+1}{=}j) P(q_{t+1}{=}j | q_t{=}i) \\
&= \sum_j p(x_{t+1} | q_{t+1}{=}j) \beta(j, t+1) P(q_{t+1}{=}j | q_t{=}i)
\end{aligned}$$

Final Details About β

- Reminder: $\beta(i,t) = p(x_{t+1}^T | q_t{=}i)$
- Final condition:

$$\beta(i,T) = \begin{cases} 1 & \text{if } i \text{ is a final state} \\ 0 & \text{otherwise} \end{cases}$$

- Hence, to compute all the β variables, we need $\mathcal{O}(N^2 \cdot T)$ operations, where N is the number of states

E-Step for HMMs

- Posterior on emission distributions:

$$
\begin{aligned}
E_Q[q_{i,t}|X,\theta^s] &= P(q_t = i|x_1^T,\theta^s) = P(q_t = i|x_1^T) \\
&= \frac{p(x_1^T, q_t = i)}{p(x_1^T)} \\
&= \frac{p(x_{t+1}^T|q_t = i, x_1^t)p(x_1^t, q_t = i)}{p(x_1^T)} \\
&= \frac{p(x_{t+1}^T|q_t = i)p(x_1^t, q_t = i)}{p(x_1^T)} \\
&= \frac{\beta(i,t)\cdot\alpha(i,t)}{\sum_j \alpha(j,T)}
\end{aligned}
$$

E-Step for HMMs

- Posterior on transition distributions:

$$E_Q[q_{i,t} \cdot q_{j,t-1}|X, \theta^s] = P(q_t = i, q_{t-1} = j|x_1^T, \theta^s)$$

$$= \frac{p(x_1^T, q_t = i, q_{t-1} = j)}{p(x_1^T)}$$

$$= \frac{p(x_{t+1}^T|q_t{=}i)P(q_t{=}i|q_{t-1}{=}j)p(x_t|q_t{=}i)p(x_1^{t-1}, q_{t-1}{=}j)}{p(x_1^T)}$$

$$= \frac{\beta(i,t)P(q_t = i|q_{t-1} = j)p(x_t|q_t = i)\alpha(j, t-1)}{\sum_j \alpha(j, T)}$$

E-Step for HMMs

- Posterior on initial state distribution:

$$
\begin{aligned}
E_Q[q_{i,0}|X,\theta^p] &= P(q_0 = i|x_1^T,\theta^s) = P(q_0 = i|x_1^T) \\
&= \frac{p(x_1^T, q_0 = i)}{p(x_1^T)} \\
&= \frac{p(x_1^T|q_0 = i)P(q_0 = i)}{p(x_1^T)} \\
&= \frac{\beta(i,0)\cdot P(q_0 = i)}{\sum_j \alpha(j,T)}
\end{aligned}
$$

M-Step for HMMs

- Find the parameters θ that maximizes A, hence search for

$$\frac{\partial A}{\partial \theta} = 0$$

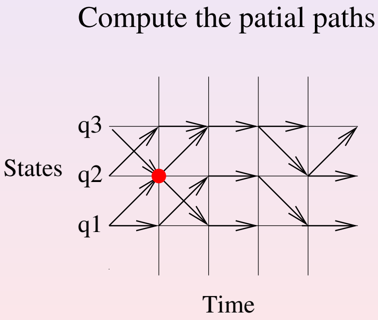
- When transition distributions are represented as tables, using a Lagrange multiplier, we obtain:

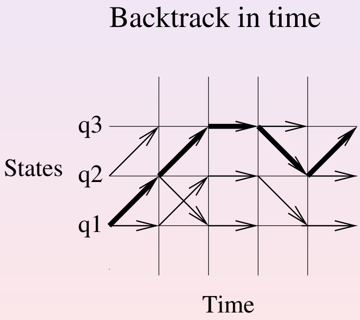
$$P(q_t = i | q_{t-1} = j) = \frac{\displaystyle\sum_{t=1}^{T} P(q_t = i, q_{t-1} = j | x_1^T, \theta^s)}{\displaystyle\sum_{t=1}^{T} P(q_t = i | x_1^T, \theta^s)}$$

- When emission distributions are implemented as GMMs, use already given equations, weighted by the posterior on emissions $P(q_t = i | x_1^T, \theta^s)$.

The Most Likely Path (Graphical View)

- The Viterbi algorithm finds the best state sequence.





The Viterbi Algorithm for HMMs

The Viterbi algorithm finds the best state sequence.

$$
\begin{aligned}
V(i,t) &\stackrel{\text{def}}{=} \max_{q_1^{t-1}} p(x_1^t, q_1^{t-1}, q_t{=}i) \\
&= \max_{q_1^{t-1}} p(x_t | x_1^{t-1}, q_1^{t-1}, q_t{=}i) p(x_1^{t-1}, q_1^{t-1}, q_t{=}i) \\
&= p(x_t | q_t{=}i) \max_{q_1^{t-2}} \max_j p(x_1^{t-1}, q_1^{t-2}, q_t{=}i, q_{t-1}{=}j) \\
&= p(x_t | q_t{=}i) \max_{q_1^{t-2}} \max_j p(q_t{=}i | q_{t-1}{=}j) p(x_1^{t-1}, q_1^{t-2}, q_{t-1}{=}j) \\
&= p(x_t | q_t{=}i) \max_j p(q_t{=}i | q_{t-1}{=}j) \max_{q_1^{t-2}} p(x_1^{t-1}, q_1^{t-2}, q_{t-1}{=}j) \\
&= p(x_t | q_t{=}i) \max_j p(q_t{=}i | q_{t-1}{=}j) V(j, t-1)
\end{aligned}
$$

From Viterbi to the State Sequence

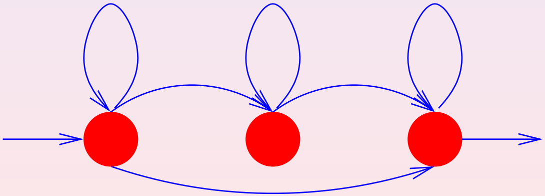
- Reminder: $V(i,t) = \max_{q_1^{t-1}} p(x_1^t, q_1^{t-1}, q_t{=}i)$
- Let us compute $V(i,t)$ for each state i and each time t of a given sequence x_1^T
- Moreover, let us also keep for each $V(i,t)$ the associated argmax previous state j
- Then, starting from the state $i = \arg\max_j V(j,T)$ backtrack to decode the most probable state sequence.
- Hence, to compute all the $V(i,t)$ variables, we need $\mathcal{O}(N^2 \cdot T)$ operations, where N is the number of states

HMMs for Speech Recognition

- Application: continuous speech recognition:

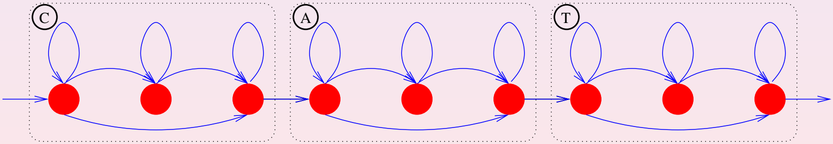
  Find a sequence of phonemes (or words) given an acoustic sequence

- Idea: use a phoneme model



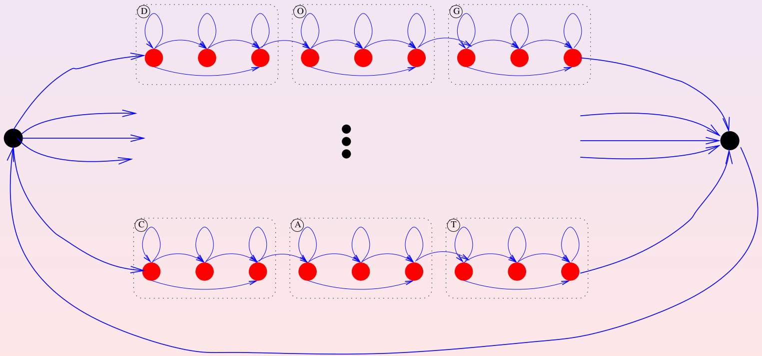
Embeded Training of HMMs

- For each acoustic sequence in the training set, create a new HMM as the concatenation of the HMMs representing the underlying sequence of phonemes.
- Maximize the likelihood of the training sentences.



HMMs: Decoding a Sentence

- Decide what is the accepted vocabulary.
- Optionally add a language model: $P(\text{word sequence})$
- Efficient algorithm to find the optimal path in the decoding HMM:



Measuring Error

- How do we measure the quality of a speech recognizer?
- Problem: the target solution is a sentence, the obtained solution is also a sentence, but they might have different size!
- Proposed solution: the Edit Distance:
    - assume you have access to the operators insert, delete, and substitute,
    - what is the smallest number of such operators we need to go from the obtained to the desired sentence?
    - An efficient algorithm exists to compute this.
- At the end, we measure the error as follows:

$$\text{WER} = \frac{\#\text{ins} + \#\text{del} + \#\text{subst}}{\#\text{words}}$$

- Note that the word error rate (WER) can be greater than 1...

Part V

Advanced Models for Multimodal Processing

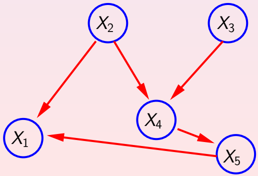
Graphical Models

- A tool to efficiently model complex joint distributions:

$$p(X_1, X_2, \cdots, X_n) = \prod_{i=1}^{n} p(X_i | \text{parents}(X_i))$$

- Introduces and makes use of known conditional independences.

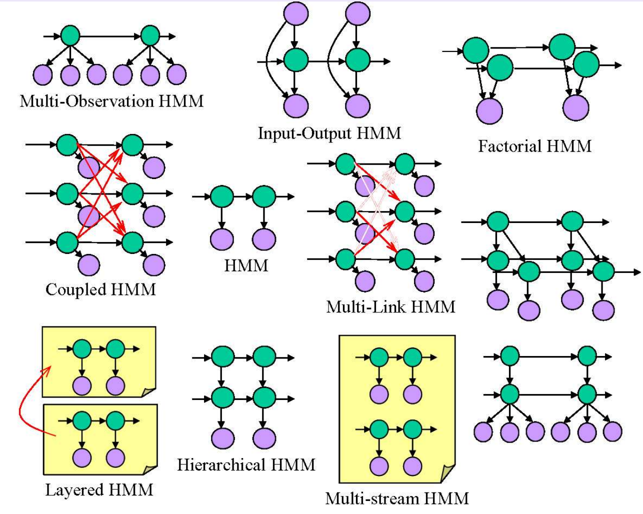
$$p(X_1, X_2, \cdots, X_n) = \prod \begin{cases} p(X_1|X_2, X_5) \\ p(X_2) \\ p(X_3) \\ p(X_4|X_2, X_3) \\ p(X_5|X_4) \end{cases}$$



Graphical Models

- Can handle an arbitrary number of random variables
- Junction Tree Algorithm (JTA): used to estimate joint probability (inference)
- Expectation-Maximization (EM): used to train
- Depending on the graph, EM and JTA are tractable or not
- Dynamic Bayes Nets: temporal extension of graphical models

A Zoo of Graphical Models for Multi-Channel Processing



Notation for Multimodal Data

Notation for One Stream

- Let us have a training set of L observations sequences.
- The observation sequence l: $\mathbf{O}^l = \left(\mathbf{o}_1^l, \mathbf{o}_2^l, \ldots, \mathbf{o}_{T_l}^l\right)$ with $\mathbf{o}_t^l$ the vector of multimodal features at time t for sequence l, of length T_l.

Notation for Several Streams

- Let us consider N streams for each observation sequence.
- The stream n of observation sequence l: $\mathbf{O}^{l:n} = \left(\mathbf{o}_1^{l:n}, \mathbf{o}_2^{l:n}, \ldots, \mathbf{o}_{T_l}^{l:n}\right)$ with $\mathbf{o}_t^{l:n}$ the vector of features of stream n at time t for sequence l, of length T_l.

Goals for Multimodal Data

Inference

$$\text{Likelihood: } \prod_{l=1}^{L} p\left(\left\{\mathbf{O}^{l:n}\right\}_{n=1}^{N};\theta\right)$$

Training

$$\theta^* = \arg\max_{\theta} \prod_{l=1}^{L} p\left(\left\{\mathbf{O}^{l:n}\right\}_{n=1}^{N};\theta\right)$$

Early-Integration HMMs

- Sample all streams at the same frame rate
- Emission distributions

$$p\left(\left\{\mathbf{o}_t^{l:n}\right\}_{n=1}^{N} | q_t = i\right)$$

- Transition distributions

$$p(q_t = i | q_{t-1} = j)$$

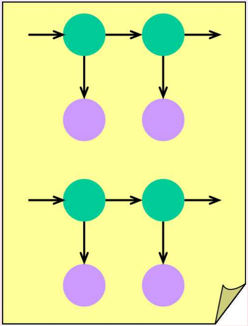
Multi-Stream HMMs

Train a separate model for each stream n

$$\theta_n^* = \arg\max_\theta \prod_{l=1}^{L} p\left(\mathbf{O}^{l:n}; \theta_n\right)$$

Inference, for each state $q_t = i$

$$p\left(\left\{\mathbf{o}_t^{l:n}\right\}_{n=1}^{N} | q_t = i\right) = \prod_{n=1}^{N} p\left(\mathbf{o}_t^{l:n} | q_t = i; \theta_n\right)$$



Multi-Stream HMMs

Training and Inference Complexity

- Per iteration, per observation sequence l:

$$\mathcal{O}(N \cdot S^2 \cdot T_l)$$

with S the number of states of each HMM stream, N the number of streams, T_l the length of the observation sequence

Additional Notes

- All stream HMMs need to be of the same topology.
- One can add stream weights ω_n as follows:

$$p\left(\left\{\mathbf{o}_t^{l:n}\right\}_{n=1}^{N} | q_t = i\right) = \prod_{n=1}^{N} p\left(\mathbf{o}_t^{l:n} | q_t = i; \theta_n\right)^{\omega_n}$$

Multi-Stream HMMs

Pros

- Efficient training (linear in the number of streams)
- Robust to stream-dependent noise
- Easy to implement

Cons

- Does not maximize the joint probability of the data
- Each stream needs to use the same HMM topology
- Assumes complete stream independence during training, and stream independence given the state during decoding.

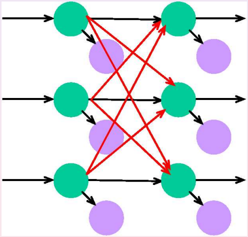
Coupled HMMs

- Let $q_t^n = i$ be the state at time t for stream n
- Stream emission distributions:

$$p\left(\mathbf{o}_t^{l:n} | q_t^n = i; \theta_n\right)$$

- Coupled transition distributions:

$$p\left(q_t^n | \left\{q_{t-1}^m\right\}_{m=1}^N; \theta_n\right)$$



Coupled HMMs

Complexity

- Per iteration, per observation sequence l:

$$\mathcal{O}(S^{2 \cdot N} \cdot T_l)$$

  with S the number of states of each HMM stream, N the number of streams, T_l the length of the observation sequence
- N-Heads approximation:

$$\mathcal{O}(N^2 \cdot S^2 \cdot T_l)$$

Coupled HMMs

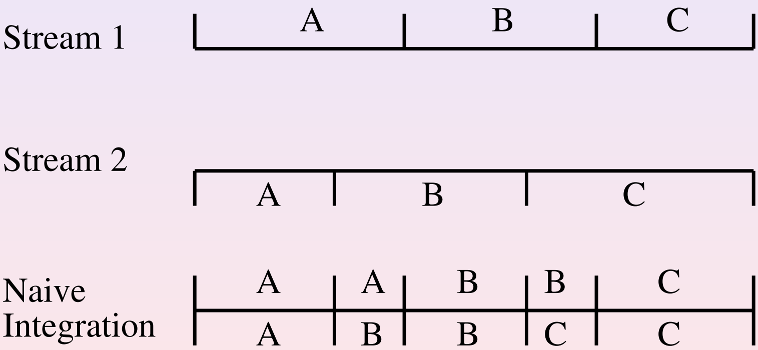
Pros

- Considers correlations between streams during training and decoding.

Cons

- The algorithm is intractable (unless using the approximation).
- Assumes synchrony between the streams.

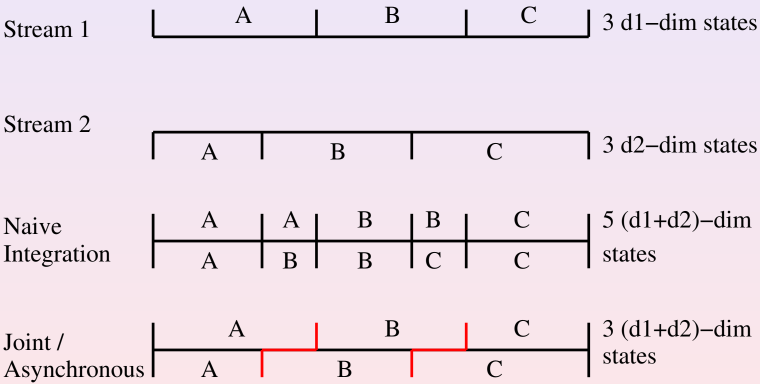
Challenges in Multi Channel Integration: Asynchrony



Some Evidence of Stream Asynchrony

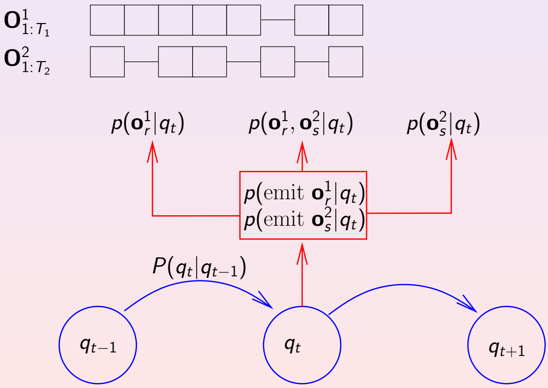
- Audio-visual speech recognition with lip movements: lips do not move at the same time as we hear the corresponding sound.
- Speaking and pointing: pointing to a map and saying "I want to go there".
- Gesticulating, looking at, and talking to someone during a conversation.
- In a news video, the delay between the moment when the newscaster says "Bush" and the moment when Bush's picture appears.
- ...

Asynchrony Revisited



Asynchronous HMMs

- Enables re-synchronization of streams.
- One HMM: maximizes the likelihood of all streams jointly.



Training and Complexity

Training

EM algorithm maximizes the joint probability $p(\mathbf{x}_1^{T_1}, \mathbf{y}_1^{T_2}, \cdots)$.

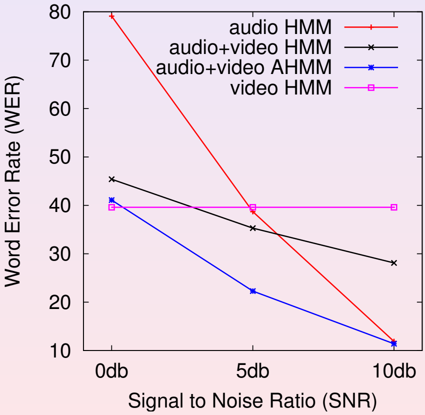
Complexity grows with number of streams, can be controlled

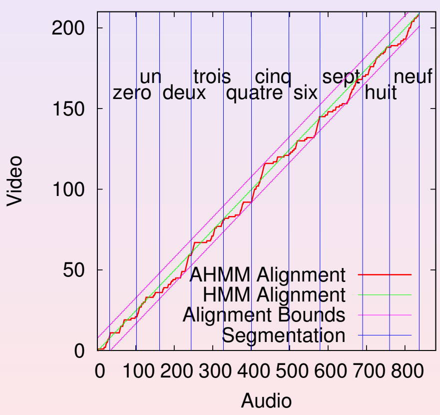
- Exact complexity: $\mathcal{O}(S^2 \cdot \prod_{i=1}^{N} T_i)$
- Introduction of temporal constraints: $\mathcal{O}(S^2 \cdot d \cdot T)$ with d the maximum delay allowed between streams.

Applications

- Easily adaptable to complex tasks such as speech recognition.
- Significant performance improvements in
  - audio-visual speech and speaker recognition
  - meeting analysis.

Alignments with Asynchronous HMMs

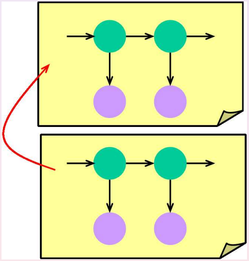




Layered HMMs

Philosophy

- Divide and Conquer for complex tasks
- Idea: transform the data from a raw representation into a higher level abstraction
- Then, transform again the result into yet another and higher level of abstraction, and continue as needed
- For temporal sequences: go from a fine time granularity to a coarser one.



General Algorithm

Layered Algorithm

For $i = 0$ to M layers, do:

1. Let $\mathbf{O}^i_{1:T}$ be a sequence of observations of HMM model m_i.
2. Train m_i using EM in order to maximise $p(\mathbf{O}^i_{1:T}|m_i)$.
3. Compute, for each state s^i_j for m_i the data posterior: $p(s^i_j|\mathbf{O}^i_{1:T}, m_i)$
4. Define $\mathbf{O}^{i+1}_{1:T}$ as the sequence of vectors of $p(s^i_j|\mathbf{O}^i_{1:T}, m_i)$

Complexity and Additional Notes

Complexity

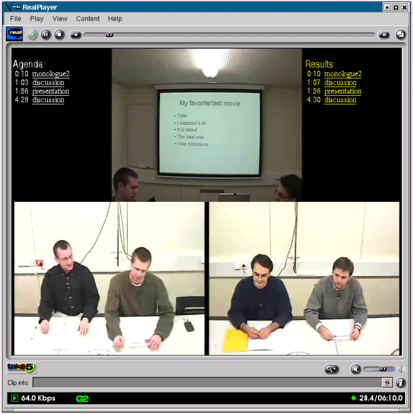
Training complexity: $\mathcal{O}(N \cdot S^2 \cdot M)$ for M layers.

A Speech Example

- A phoneme layer (with phoneme constraints)
- A word layer (with word constraints)
- A sentence layer (with language constraints)

Complexity of the Meeting Scenario

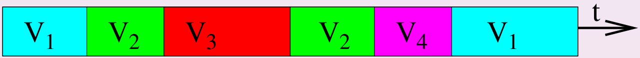
- Modeling multimodal group-level human interactions in meetings.
- Multimodal nature: data collected from multiple sensors (cameras, microphones, projectors, white-board, etc).
- Group nature: involves multiple interacting persons at the same time.



Meeting Analysis

- Structure a meeting as a sequence of group actions taken from an exhaustive set V of N possible actions:

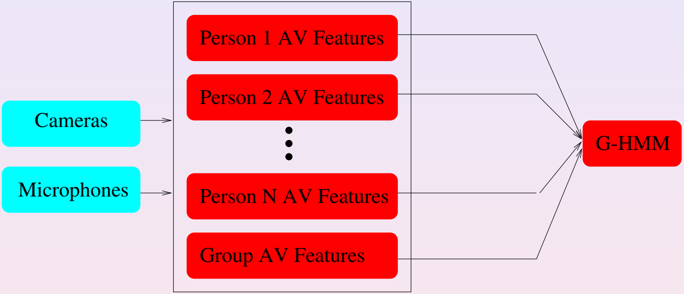
$$V = \{v_1, v_2, v_3, \cdots, v_N\}$$



- Recorded and annotated 30 training a 30 test meetings.
- Extract high level audio and visual features.
- Try to recover the target action sequence of unseen meetings.

I. McCowan, D. Gatica-Perez, S. Bengio, G. Lathoud, M. Barnard, and D. Zhang. Automatic Analysis of Multimodal Group Actions in Meetings. *IEEE Transactions on PAMI*, 27(3), 2005.

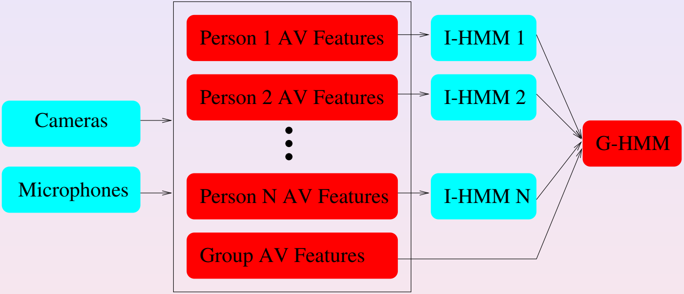
A One-Layer Approach



Classical Approach

- A large vector of audio-visual features from each participant and group-level features are concatenated to define the observation space.
- A general HMM is trained using a set of labeled meetings.

A Two-Layer Approach



Advantages

- Smaller observation space.
- I-HMMs share parameters, person independent, trained on simple task (write, talk, passive).
- Last layer less sensitive to variations of low-level data.

Experiments

Group Actions: Turn-Taking

- Discussion
- Monologue
- Monologue/Note-Taking
- Presentation
- Presentation/Note-Taking
- White-board
- White-board/Note-Taking

Individual Actions

Speaking - Writing - Passive

Results

Method	Features	AER
One-Layer	Visual Only	48.20
	Audio Only	36.70
	Audio Visual	23.74
Two-Layer	Visual Only	42.45
	Audio Only	32.37
	Audio Visual	16.55
	Async HMM	15.11

D. Zhang, D. Gatica-Perez, S. Bengio, I. McCowan, and G. Lathoud. Modeling Individual and Group Actions in Meetings: a Two-Layer HMM Framework. In *CVPR*, 2004.

Part VI

References

Statistical Learning Theory

1. V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
   NOTE: The theory is explained here with all the equations.
2. V. N. Vapnik. *The nature of statistical learning theory*. Springer, second edition, 1995.
   NOTE: A good introduction to the theory, not much equations.

Multi-Layer Perceptrons

1. Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.
   NOTE: Very good paper proposing a series of tricks to make neural networks really working.
2. B. D. Ripley. *Pattern recognition and Neural networks*. Cambridge University Press, Cambridge, UK, 1996.
   NOTE: A good general book on machine learning and neural networks. Orientation: statistics.
3. C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, London, UK, 1995.
   NOTE: A good general book on machine learning and neural networks. Orientation: physics.
4. S. Haykin. *Neural Networks. A Comprehensive Foundation, 2nd edition*. Macmillan College Publishing, New York, 1994.
   NOTE: A good general book on machine learning and neural networks. Orientation: signal processing.

Gaussian Mixture Models and Hidden Markov Models

1. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39:1–38, 1977.
   NOTE: A theoretical paper introducing the EM algorithm.
2. L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
   NOTE: A good introduction to HMMs and speech recognition.
3. L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 1986.
   NOTE: A very good introduction to HMMs.

Advanced Models for Multimodal Processing

1. M. Jordan. *Learning in Graphical Models*. MIT Press (1999)
2. H. Bourlard and S. Dupont. Subband-based speech recognition. In Proc. IEEE ICASSP (1997)
3. M. Brand. Coupled hidden markov models for modeling interacting processes. Technical Report 405, MIT Media Lab Vision and Modeling (1996)
4. S. Bengio. Multimodal speech processing using asynchronous hidden markov models. *Information Fusion* **5** (2004) 81–89
5. N. Oliver, E. Horvitz, and A. Garg. Layered representations for learning and inferring office activity from multiple sensory channels. In Proc. of the Int. Conf. on Multimodal Interfaces. (2002)