Sequential Data Analysis

Modèles

- Modèles statistiques de langage
- Modèles de Markov discrets et continus (HMM)
- ~~Champs Aléatoire Conditionnels (CRF~~)
- Modèles Neuro-Markoviens
- Réseaux de neurones récurrents (RNN – BLSTM)

Applications

- Parole et écriture
- Mise en œuvre

Extensions

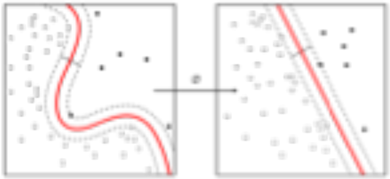
- Modèles neuronaux à attention

Introduction

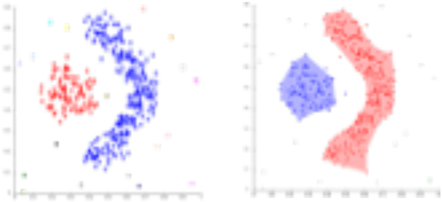
Static Representation / Sequence Representation of Data

A static representation is commonly an observation or a description of the data in a fixed size, n-dimensional, real valued, feature space.

Supervised classification : we seek to find an optimal discriminator between the categories, or classes, that are assigned to the data (**labeled data**), by exploiting this n-dimensional real valued representation, vector space.



Unsupervised classification: we seek to determine some structures in the data by examining this n-dimensional real valued representation (observation) only.
Unlabeled data: no information about classes!



Possible solutions are:
Supervised classification : SVM, KNN, NN, DeepNN, Decision Trees, Random Forest, etc…
Unsupervised classification: GMM, K-means, fuzzy k-means, GAN etc…

Introduction

Static Representation / Sequence Representation of Data

A sequence representation is an ordered representation of observations of variable length T.
The position in the sequence matters, it is denoted by the variable t

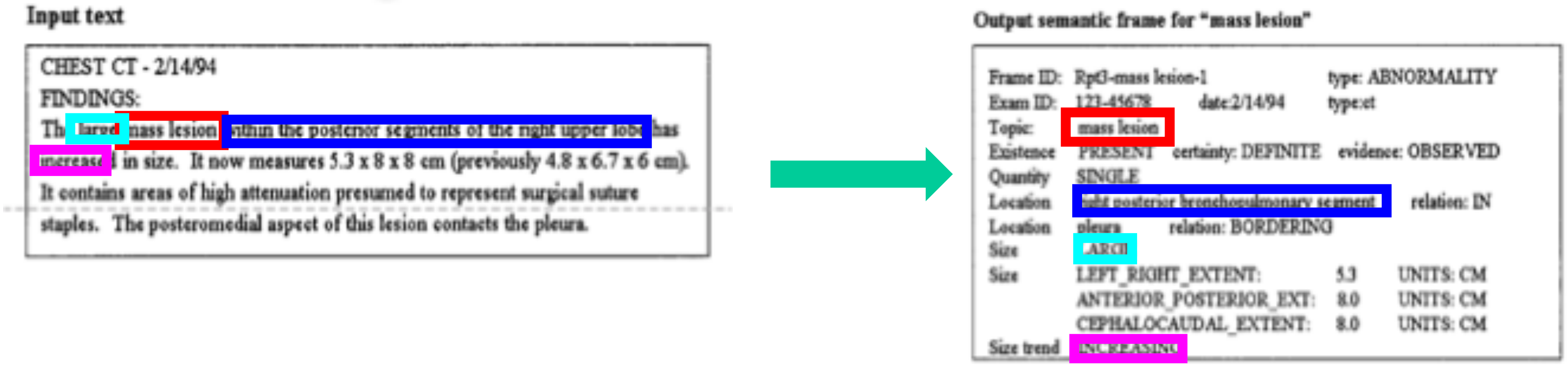
$$O = (o_1 o_2 \cdots o_t \cdots o_T\,)$$

Discrete observation sequences

observations belong to a finite set of symbols $o_t \in V = \{v_1, v_2\,, \cdots, v_M\,\}$

Character strings encoding a language (M = 100)
Word sequences (M = 10K…50K…)

Ex: Language translation, Natural Language Processing (NLP), Textual Information Extraction (IE),



Input text

CHEST CT - 2/14/94
FINDINGS:
The larged mass lesion within the posterior segments of the right upper lobe has increased in size. It now measures 5.3 x 8 x 8 cm (previously 4.8 x 6.7 x 6 cm). It contains areas of high attenuation presumed to represent surgical suture staples. The posteromedial aspect of this lesion contacts the pleura.

Output semantic frame for "mass lesion"

Frame ID:	Rpt3-mass lesion-1		type: ABNORMALITY
Exam ID:	123-45678	date:2/14/94	type:ct
Topic:	mass lesion		
Existence	PRESENT	certainty: DEFINITE	evidence: OBSERVED
Quantity	SINGLE		
Location	right posterior bronchopulmonary segment		relation: IN
Location	pleura	relation: BORDERING	
Size	LARGE		
Size	LEFT_RIGHT_EXTENT:	5.3	UNITS: CM
	ANTERIOR_POSTERIOR_EXT:	8.0	UNITS: CM
	CEPHALOCAUDAL_EXTENT:	8.0	UNITS: CM
Size trend	INCREASING		

Introduction

Static Representation / Sequence Representation of Data

Continuous observation sequences

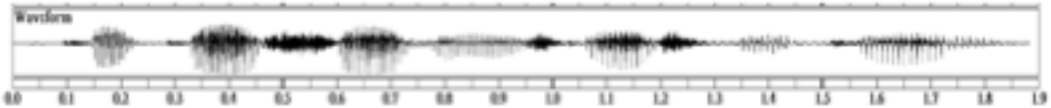
sequence of variable length

observations are made of N measurements

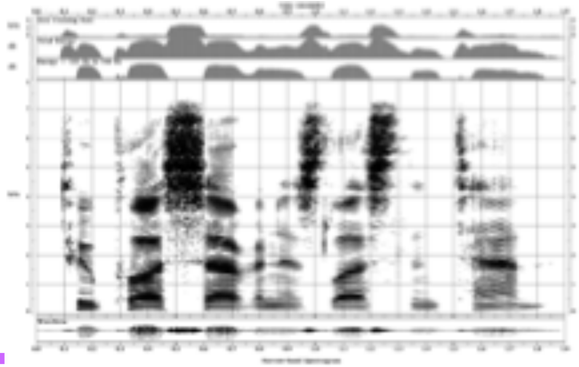
each measurement is a real valued sample

$$O = (o_1 o_2 \cdots o_t \cdots o_T\,) \qquad o_t \in \mathbb{R}^N \qquad O = \begin{pmatrix} o_{11} & o_{21} & & o_{t1} & & o_{T1} \\ \vdots & \vdots & \cdots & \vdots & \dots & \vdots \\ o_{1N} & o_{2N} & & o_{tN} & & o_{TN} \end{pmatrix}$$

Example:
Raw speech signal, n=1 measurement every 1/8000 sec (Fe=8 KHz)



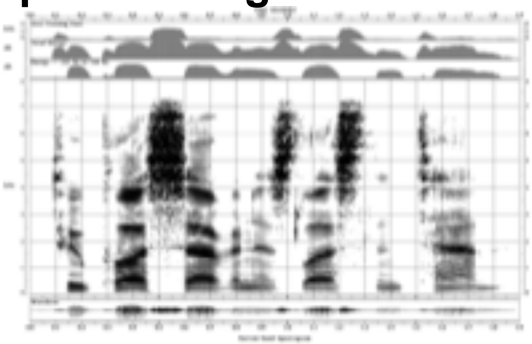
Short term spectrogram, cepstral coefficients : n=39 cepstral coefficients over a sliding window of 256 samples (32 ms)



Introduction

Applications

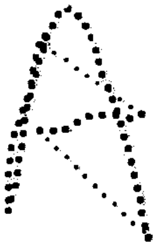
Speech recognition



Pen Based Computing

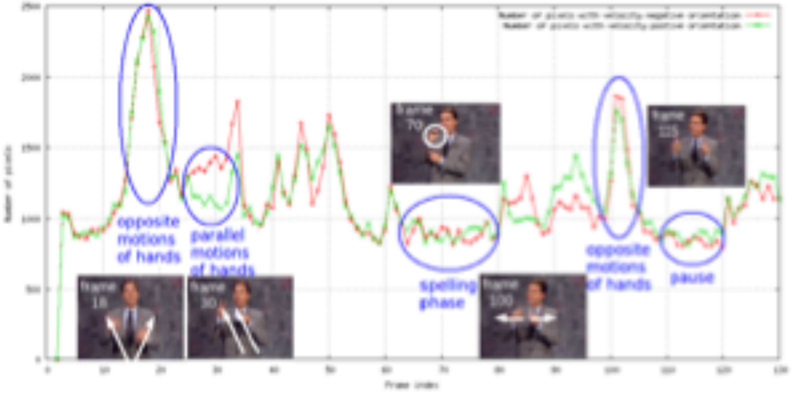
Mobile Interfaces

PDA



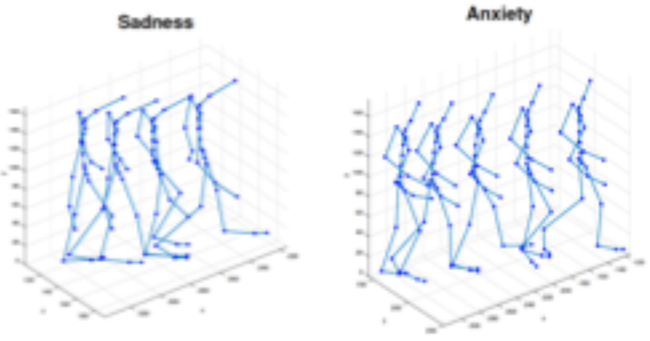
Gesture recognition in videos

sign language



3D gestures recognition

Kinect, games and animation



Introduction

Static Representation / Sequence Representation of Data

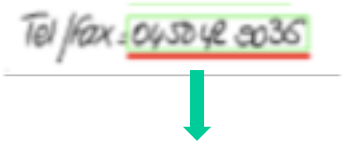
We are interested to process the sequence representations for two main purposes

1. Labeling the sequence as a whole entity : assign a label (**many to one**):

classification
with a variable size
representation

2. Labeling the components of the sequence and obtain a sequence of labels (more general problem, **many to many**):

Handwriting recognition from images



0450430036

Information extraction from raw textual data

```
…and responsability. John Smith, chief financial officer of prime coorp since....
  I        I           Per  Per     Pos    Pos        Pos      I  I     I    I
```

detection
and **classification**
with a variable size
representation

multiple decisions
along the sequence

Introduction

Static Representation / Sequence Representation of Data

General formulation of the sequence labeling problem (problem2)

Look for the best label sequence $W^* = (w_1^* w_2^* \cdots w_j^* \cdots w_J^*)$ $w_j \in \{l_1, l_2, \cdots, l_d\}$=D

that can be associated to the observation sequence $O = (o_1 o_2 \cdots o_t \cdots o_T)$ $o_t \in \mathbb{R}^N$

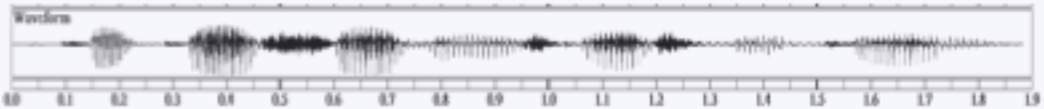
$$W^* = \underset{W}{\operatorname{argmax}}\, P(W|O)$$

The sequence of labels ***W*** may not have the same length as ***O***
D *is the set of possible output labels,* ***lexicon*** *of possible words*
d *= 20K,40K, 60K…. in case of natural language*

Example of speech signal

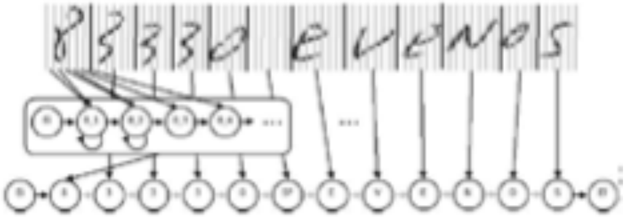
input observation ***O***

output label sequence ***W****



the chief financial officer of prime coorp

Example of handwriting image



Introduction

Static Representation / Sequence Representation of Data

General formulation of the labelling problem

$$W^* = \underset{W}{\operatorname{argmax}}\, P(W|O) = \underset{W}{\operatorname{argmax}} \frac{P(O|W)P(W)}{P(O)} \propto \underset{W}{\operatorname{argmax}}\, P(O|W)P(W)$$

$P(O|W)$ is the **data attachment model** (acoustic model, image model ….)

$P(W)$ is the **language model**

We need to design two kinds of models

Language models : n-grams, RNN
Data attachement models: HMM, NN-HMM, RNN

Towards End to End models : RNN + RNN

Statistical language models (LM)

We focus our attention on how to model sequences of words of a language

so as to design efficient models capable of estimating the probability of a certain sequence of words to occur

i.e. compute $P(W)$ where W= $(w_1 w_2 \cdots w_i \cdots w_I)$

Language modelling is the art of determining the probability of a sequence of words (Joshua, 2001).
The Shannon game of guessing the next letter or next word in a text (Shannon 1951)

The model is sufficiently general to apply to any type of sequence, not only words

=> sequence of characters, gene sequence, etc…

The **principal assumption** for building a language model is to consider that words have some specific collocations. They do not occur independently one from the other

Examples

strong tea

weapons of mass destruction

death penalty

white house

peine de mort

carte de séjour

conseil des ministres

Statistical language models (LM)

W= $(w_1 w_2 \cdots w_i \cdots w_I)$

We assume that collocations of words occur within a certain **context** of the words, or **history** of the words denoted by h_i=$(w_1 w_2 \cdots w_{i-1})$ for word w_i

Then, the probobability of the sentence is the product of the word probabilities in their respective contexts

$$P(W) = \prod_{i=1}^{I} P(w_i | h_i)$$

Problem: a very large number of histories, and subsequently a very large number of probabilities to estimate

Solution: regroup histories by equivalence classes colled n-gram denoted by $\phi(h_i)$ such that the following approximation holds : $P(w_i|h_i) \approx P(w_i|\phi(h_i))$
n-gram are sequences of n consecutive words $\phi(h_i) = (w_{i-n+1} \cdots w_{i-2} w_{i-1} w_i)$

The probabilities $P(w_i|h_i) \approx P(w_i|w_{i-n+1} \cdots w_{i-2}\, w_{i-1})$ are the parameters of the LM

Assume *d=#D=20K words and n=2 (bigrams) there is 20 000 X 19 999 = 400 10^6 parameters*
and n=3 (trigrams) there is 20 000^2 X 19 999 = 8 10^{12} parameters
and n=4 (four-grams) there is 20 000^3 X 19 999 = 1.6 10^{17} parameters

n-gram models require a huge amount of parameters to be estimated, even for small dictionary size, and small history. Large datasets of text exemples are required to have correct estimates of these quantities

Statistical language models (LM)

Statistical Estimators of n-gram LM

Notations

N *amount of training words*

w_{i-n+1}^{i} *the n-gram* $w_{i-n+1} \cdots w_{i-1} \, w_i$

$C(w_{i-n+1}^{i})$ *the number of occurrences of n-gram*

We want to estimate the following quantities $P(w_i | w_{i-n+1}^{i-1})$ *from the observation of a training corpus of text.*

The Maximum Likelihood estimator (MLE)

$$P_{MLE}(w_i | w_{i-n+1}^{i-1}) = \frac{C(w_{i-n+1}^{i})}{C(w_{i-n+1}^{i-1})}$$

Statistical language models (LM)

Assume the following training corpus

<s> JOHN READ MOBY DICK </s>

<s> OPEND READ BOOK LIBRARY </s>

<s> MARY READ A DIFFERENT BOOK </s>

<s> SHE READ A BOOK BY CHER </s>

The probability of the sentence "JOHN READ A BOOK" is the following, with n = 2.

$$\text{P}(<s> JOHN\ READ\ A\ BOOK <\backslash s>) = P(JOHN| <s>)P(READ|JOHN)P(A|READ)\ P(BOOK|A)\ P(<\backslash s> |BOOK)$$

$$= \frac{C(<s> JOHN)}{C(<s>)} \times \frac{C(JOHN\ READ)}{C(JOHN)} \times \frac{C(READ\ A)}{C(READ)} \times \frac{C(A\ BOOK)}{C(A)} \times \frac{C(BOOK <\backslash s>)}{C(BOOK)}$$

$$= \frac{1}{4} \times \frac{1}{1} \times \frac{2}{4} \times \frac{1}{2} \times \frac{1}{3} = \frac{1}{3X16} \approx 0{,}0208$$

Limitation : MLE estimates assign 0 probability to unseen events

We look for better estimators able to affect low probability to unseen events

=> Introduce smoothing in the estimation

Statistical language models (LM)

A smooth estimator has two components : a **discounting model** and a **redistribution model**

The discounting model is used to deduct a small amount of the probabilities allocated to the word sequences that are present in the training set.
Probability re-distribution is the process of affecting some values to unseen n-gram

Linear discounting
Linear discounting consists in taking a fraction of the probability of a n-gram, according to its number of occurrences in the training set.

$$P_{LD}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \dfrac{C(w_{i-n+1}^{i})\times r}{C(w_{i-n+1}^{i-1})} & if \quad C(w_{i-n+1}^{i}) > 0 \\ 0 & otherwise \end{cases}$$

where r is the **discounting factor** (slightly less than one) $0 < r < 1$

The total probability mass kept aside of the seen events on the training corpus is

$$\sum_{C(w_{i-n+1}^{i})} P_{MLE}(w_i|w_{i-n+1}^{i-1}) - \sum_{C(w_{i-n+1}^{i})} P_{LD}(w_i|w_{i-n+1}^{i-1}) = 1\text{-}r$$

Statistical language models (LM)

Absolute discounting

Absolute discounting consists in discounting a fixed value to the number of occurrences of n-grams in the training set.

$$P_{AD}\left(w_i|w_{i-n+1}^{i-1}\right) = \begin{cases} \dfrac{C\left(w_{i-n+1}^{i}\right) - D}{C\left(w_{i-n+1}^{i-1}\right)} & if \quad C\left(w_{i-n+1}^{i}\right) > 0 \\ 0 & otherwise \end{cases}$$

D is the **fixed discounting factor**

The total probability mass kept aside of the seen events on the training corpus is

$$\sum_{C\left(w_{i-n+1}^{i}\right)} P_{MLE}\left(w_i|w_{i-n+1}^{i-1}\right) - \sum_{C\left(w_{i-n+1}^{i}\right)} P_{AD}\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{\left|\left\{w_{i-n+1}^{i-1} : C\left(w_{i-n+1}^{i-1}\right) > 0\right\}\right| \times D}{C\left(w_{i-n+1}^{i-1}\right)}$$

$\left|\left\{w_{i-n+1}^{i-1}: C\left(w_{i-n+1}^{i-1}\right) > 0\right\}\right|$ is the number of distinct word histories preceding word w_i

Statistical language models (LM)

Probability redistribution

Goal : Affect some non null probabilities to unseen events by redistributing the previously discounted probability mass to the events seen in the training corpus

The back-off model of probability redistribution

Estimate the probability of an unseen n-gram by exploiting the probability of a lower order n-gram. i.e. going back to (n-1) gram if seen, otherwise (n-2) gram if seen, otherwise (n-3) gram if seen etc…

The model can use any type of discounting estimators (Linear, absolute,….)

$$P_{Back-off}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \lambda(w_{i-n+1}^{i-1}) P_{Back-off}(w_i|w_{i-n+2}^{i-1}) & if \;\; C(w_{i-n+1}^{i}) = 0 \\ P_{Disc}(w_i|w_{i-n+1}^{i-1}) & if \;\; C(w_{i-n+1}^{i}) > 0 \end{cases}$$

with $P_{Disc}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} P_{LD}(w_i|w_{i-n+1}^{i-1}) \\ or \\ P_{AD}(w_i|w_{i-n+1}^{i-1}) \end{cases}$

$\lambda(w_{i-n+1}^{i-1})$ the back-off weights, computed so that probabilities sum to 1

Statistical language models (LM)

Language model evaluation

Assume the test dataset T of N sentences, M words

LM estimates the probability of each sentence W $P(W)$

We can derive the probability of T $P(T)=\prod_{k=1}^{N} P(W^k)$

The best LM is the one that assigns the highest probability to the test dataset

The cross entropy evaluates how different the language model is with the equiprobable distribution.

$$H_{LM}(T)=-\frac{1}{M}Log_2P(T)$$

This is the average number of bits required to encode the M words of the dataset

The perplexity is the average number of words that can follow the current word. It tells how much the LM hesitates in the prediction of the next word.

$$PP_{LM}(T)=2^{H_{LM}(T)}$$

$PP_{LM}(T)=k$ means there are on average, k equally likely words after a word

The lower the perplexity the better the LM

Statistical language models (LM)

Available tools for statistical LM training and evaluation

SRI LM : https://www.sri.com/engage/products-solutions/sri-language-modeling-toolkit
MIT LM : http://projects.csail.mit.edu/cgi-bin/wiki/view/SLS/MITLMTutorial

You can train a LM with one of these toolkit and export the LM using a standard DARPA format

Most of the platform for speech processing and training are supporting these formats

Data Attachment models

General solution to the labeling problem

$$W^* = \underset{W}{\operatorname{argmax}} P(W|O) = \underset{W}{\operatorname{argmax}} \frac{P(O|W)P(W)}{P(O)} \propto \underset{W}{\operatorname{argmax}} P(O|W)P(W)$$

$P(O|W)$ is the **data attachment model** (acoustic model in case of speech, graphic model in case of handwriting)

$P(W)$ is the **language model**

We now concentrate on the possible data attachement models

- ✓ **Hidden Markov models :** the historical first model applied to speech in the 80's
- ✓ Hybrid Neuro-Markov models 90's
- ✓ Pure Neuronal models thanks to the introduction of Recurrent Neural Networks (RNN)

Andrei Markov
1856-1922

The Markov hypothesis

Idea: the probability of a character in a text in highly conditioned by the preceding characters

$$P(Q) = P(q_1 q_2 q_3 \ldots q_T)$$

The law of conditional probabilities writes

$$P(q_1 q_2 q_3 \ldots q_T) = P(q_T \mid q_1 q_2 \ldots q_{T-1}) P(q_1 q_2 q_3 \ldots q_{T-1})$$

The first order Markov assumption writes

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

Thus the probability of the whole sequence can be factorized as follows

$$P(q_1 q_2 q_3 \ldots q_T) = P(q_1) \prod_{t=2}^{T} P(q_t \mid q_{t-1})$$

Notice : the first order Markov hypothesis highly reduces the computational complexity but it is often too simple to model real, Language models use n-gram models of higher order at least 2 or 3, but 10 or more are possible models.

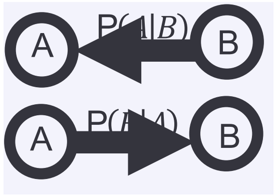
Order 1 => bi-grams order 2 => tri-grams order n-1 => n-grams

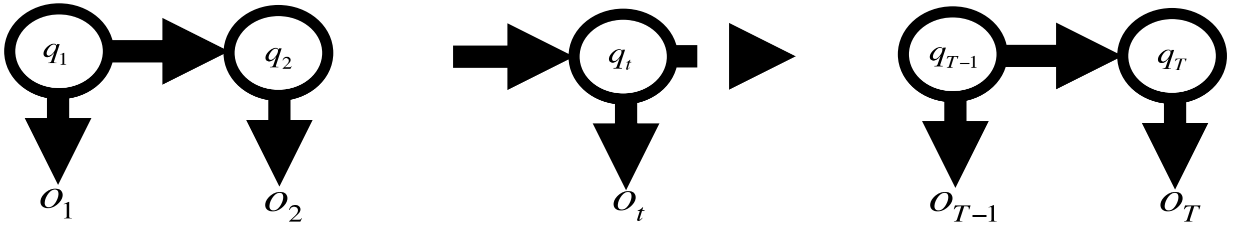
The Hidden Markov Model (HMM)

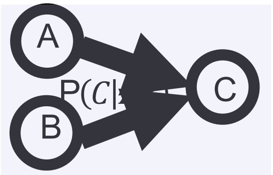
It is a **generative model** that describes how the **label sequence *Q*** can generate the **observation sequence *O***. The two sequences have a same length T

It is represented by an oriented temporal graph (local label are represented by circles, while local observations are not).

The arrows represent conditional probabilities







Model hypotheses

1- **1st ordre label dependencies** over time

$$P(Q) = P(q_1)\prod_{t=2}^{T} P(q_t | q_{t-1})$$

2- **Conditional independence** of observations

$$P(O|Q,M) = p(o_1|q_1)\,p(o_2|q_2)\ldots p(o_T|q_T)$$

The model factorizes as

$$P(O,Q|M) = \pi_{q_1}\prod_{t} p(o_t|q_t)\,p(q_t|q_{t-1})$$

HMM Parameters

A HMM is **stationary**: conditional probabilities do not change over time

Hidden States or labels. $q_t \in S \qquad S = \left\{ s_1, s_2 \ldots s_K \right\}$

Transition probabilities

$$A = \begin{pmatrix} P(s_1 | s_1) & \ldots & P(s_K | s_1) \\ . & P(s_i | s_j) & . \\ P(s_1 | s_K) & \ldots & P(s_K | s_K) \end{pmatrix} = \begin{pmatrix} a_{11} & \ldots & a_{1K} \\ . & a_{ij} & . \\ a_{K1} & \ldots & s_{KK} \end{pmatrix}_{(K,K)} \qquad \sum_{1 \le k \le K} a_{ik} = 1$$

Initial probabilities

$$\Pi = \left(\pi_k \right) = \left(P(q_1 = s_k) \right)_K \qquad \sum_{1 \le k \le K} \pi_k = 1$$

For discrete **observations** $o_t \in V \qquad V = \left\{ v_1, \ v_2 \ \ldots \ v_M \right\}$

Emission probabilities

$$B = \left(P(o_t = v_m | s_k) \right) = \begin{pmatrix} P(v_1 | s_1) & P(v_2 | s_1) & \ldots & P(v_M | s_1) \\ . & . & \ldots & . \\ P(v_1 | s_N) & P(v_2 | s_N) & \cdots & P(v_M | s_N) \end{pmatrix}_{(K,M)}$$

HMM Parameters

For **continuous observations** $o_t \in \mathbb{R}^N$

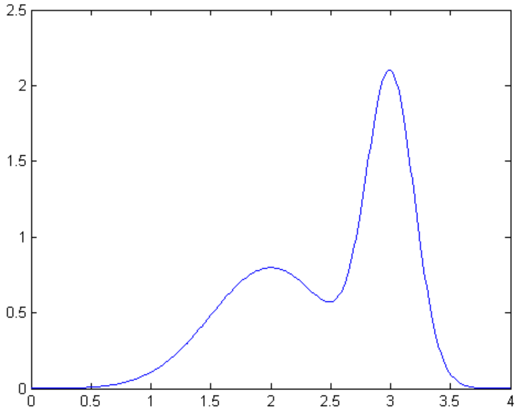
Emission probabilities : one Gaussian Mixture Models (GMM) per state

$$p(o_t|q_t = s_k) = \sum_{1 \leq m \leq M} P(C_{mk})\,\mathcal{N}(o_t, \mu_m, \Sigma_m)$$

K X M N-dimensional Gaussians…

K = 26 character x 10
M = 10…20 Gaussians per state
N =50

=> 2600 Gaussians (clusters)



HMM as a dynamic model

We can represent the model by a an Oriented Graphical Model showing the stationary conditional dependencies over time (from t to t+1)

example of a 3 states and 2 discrete observations model

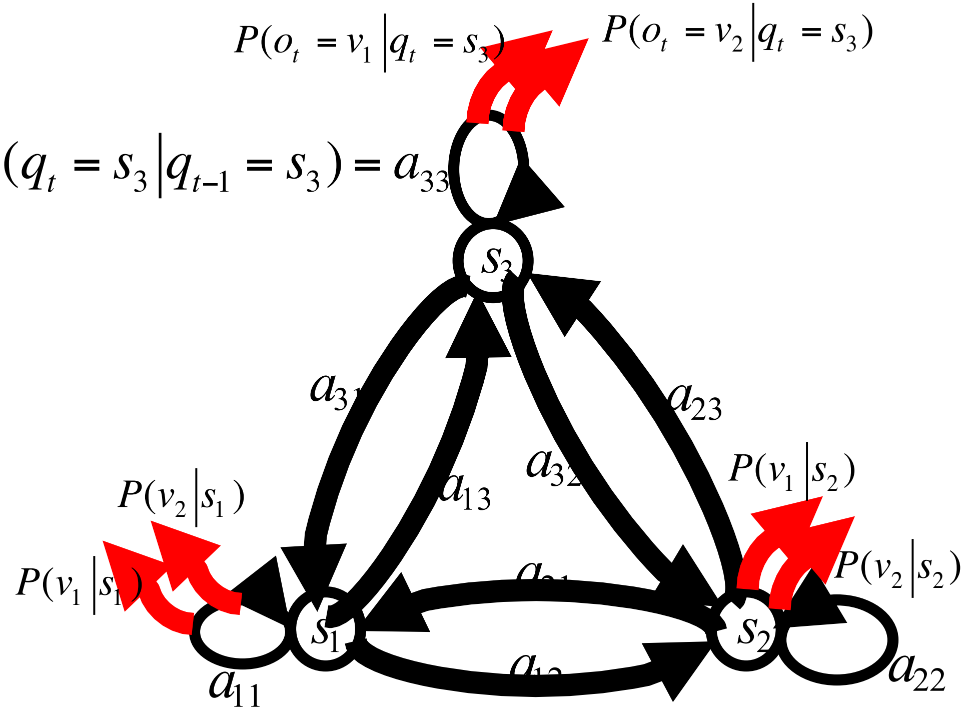
$$S = \{s_1, s_2, s_3\}$$

$$V = \{v_1, v_2\}$$

$$o_t \in V$$

$$q_t \in S$$

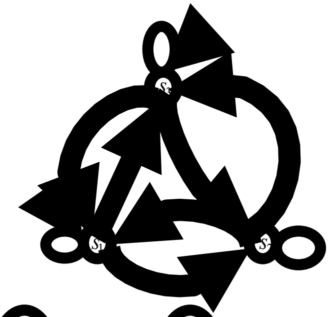
$$P(q_t = s_3 | q_{t-1} = s_3) = a_{33}$$



$$\pi_k = P(Y = y_k) \qquad k = 1, \ldots, K \qquad avec \qquad \sum_k \pi_k = 1$$

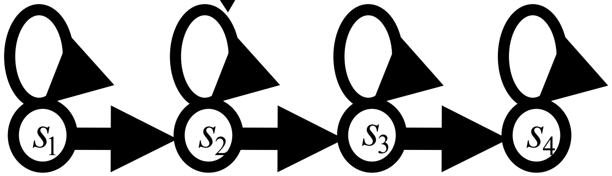
Various HMM Topologies

Ergodic



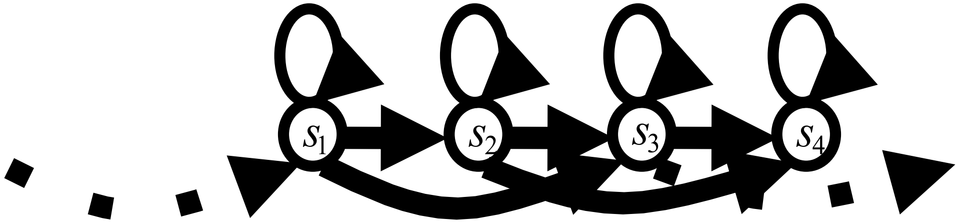
$$\Pi = \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{pmatrix} \quad A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

Left Right



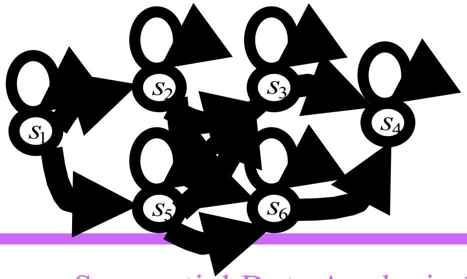
$$\Pi = \begin{pmatrix} \pi_1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad A = \begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{pmatrix}$$

Bakis

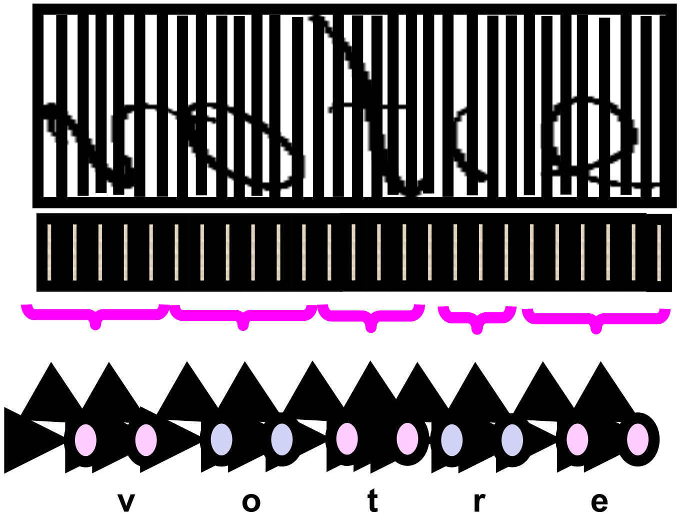


$$\Pi = \begin{pmatrix} \pi_1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{23} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{pmatrix}$$

Parallel

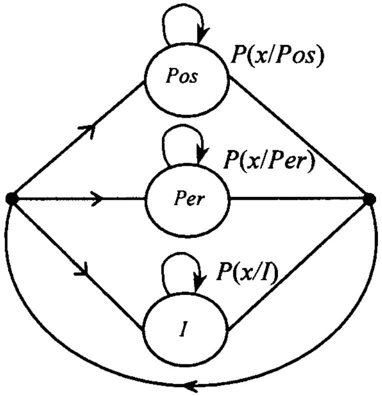


Tasks related topologies



Sequential models for seq to seq problems

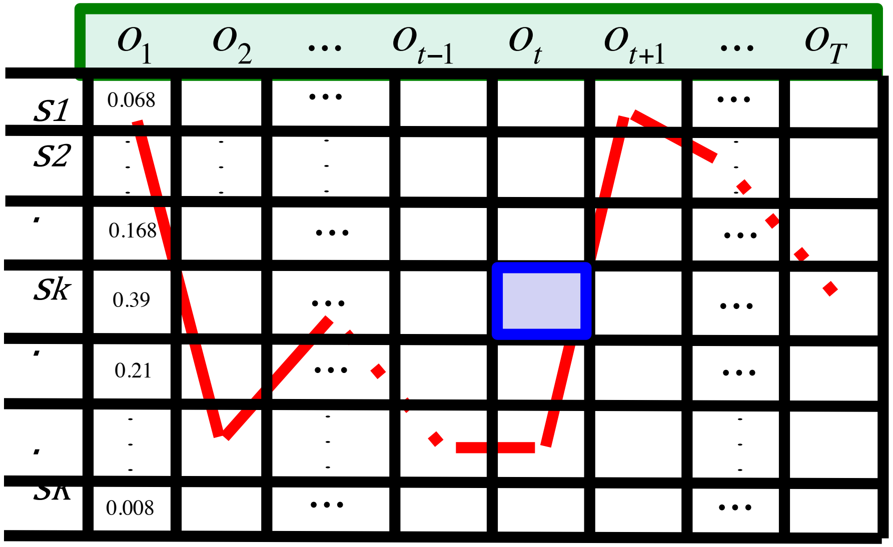
…and responsability. John Smith, chief financial officer of prime coorp since…..
I I Per Per Pos Pos Pos I I I I



Ergodic model for text labeling (IE)

Inference Algorithms with HMM

Inference : Look for a solution to a certain problem through a lattice of hypothesis



Viterbi : Which is the best sequence of hidden states ?

$$\boxed{Q^* = \underset{Q}{\operatorname{argmax}}\, P(O,Q|M)}$$

Forward : What is the likelihood of the observation sequence ?

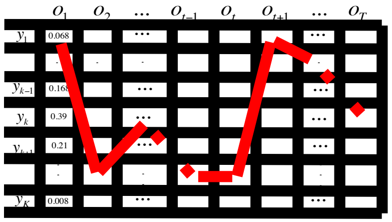
$$\boxed{P(O|M)}$$

Forward-Backward : What is the local state posterior at time t ?

$$\boxed{P(q_t = s_k|O,M)}$$

Andrew James Viterbi
1935-...

Viterbi inference (1967)



We look for the **optimal** sequence Q that best « explains » the observation.

Among the K^T possible state sequences, we look for the **most probable one**

$$Q^* = \underset{i=1,..,K^T}{\operatorname{argmax}} \left(P(O, Q_i \mid M) \right)$$

Direct computation is intractable but can take benefit of the time lattice structure so as to factorize sub-paths when computing the cost of a path i.e. state sequence

$$P(O, Q \mid M) = \pi_{q_1} \prod_t p(o_t \mid q_t) p(q_t \mid q_{t-1})$$

This product can be decomposed into elementary probabilities (always >0).
This quantity is monotonically decreasing.

The **Dynamic Programming (DP)** principle can apply (Richard Bellman 1950)

« The global optimum is composed of local optima »

Define $$\delta_t(i) = \max_{q_1 q_2 .. q_{t-1}} \left(P(o_1 o_2 o_3 \ldots o_t, q_1 q_2 q_3 \ldots q_t = i \mid M) \right)$$ the partial solution till t

The one time step optimum is $$\delta_t(i) = \max_j \left(\delta_{t-1}(j) a_{ji} \right) b_i(o_t)$$

Complexity is reduced from K^T to K^2 X T

The Viterbi algorithm (1967)

Initialisation

$$\delta_1(k) = \pi_k b_k(o_1) \quad 1 \le k \le K$$

$$\psi_1(k) = 0$$

Recursion

$$\delta_t(k) = \max_j \left(\delta_{t-1}(j) a_{jk}\right) b_k(o_t) \quad 1 \le i \le K \qquad 2 \le t \le T$$

$$\psi_t(k) = \operatorname{argmax}_j \left(\delta_{t-1}(j) a_{jk}\right) \qquad 1 \le k \le K \qquad 2 \le t \le T$$

Termination

$$P^* = \max_k \left(\delta_T(k)\right)$$

$$q_T^* = \operatorname{argmax}_k \left(\delta_T(k)\right)$$

Best path decoding (backtracking)

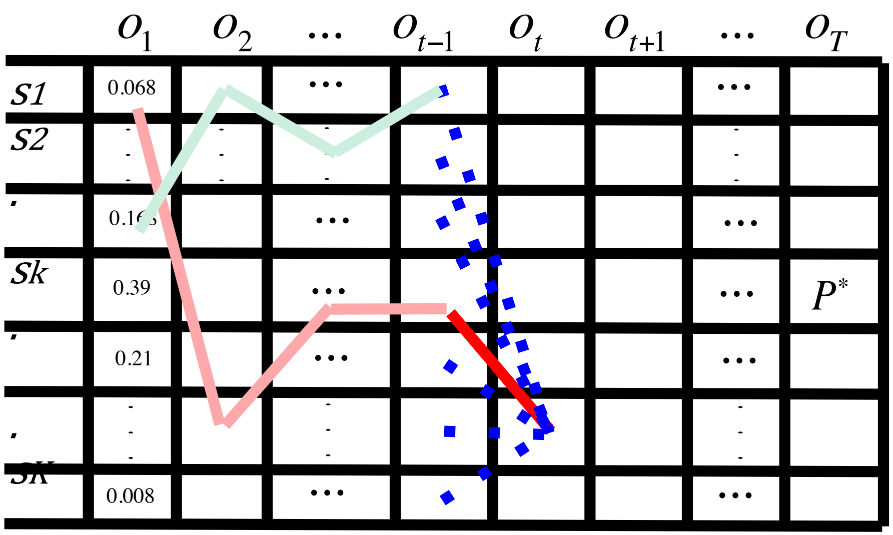
$$q_t^* = \psi_{t+1}\left(q_{t+1}^*\right) \quad t = T-1, T-2, \ldots, 1$$

Best implementation when using log probabilities instead of probabilities, so as to avoid multiplications.

The Viterbi Algorithm (1967)

Illustration

$$\delta_t(i) = \max_j \left(\delta_{t-1}(j) a_{ji}\right) b_i(o_t)$$



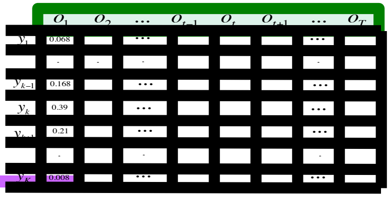
The optimal path is unknown till the last column is reached $P^* = \max_j \left(\delta_T(j)\right)$

Then we can compute the last state of the best path $q_T^* = \arg\max_j \left(\delta_T(j)\right)$

Then we can retrieve the previous states of the optimal path, if the optimal local paths have been stored in the variable $\psi \quad t(i)$

$$q_{t-1}^* = \psi_t(i) = \arg\max_j \left(\delta_{t-1}(j) a_{ji}\right)$$

The Forward Inference



What is the likelyhood of the observation sequence? $\boxed{P(O|M)}$
… regardless the hidden states

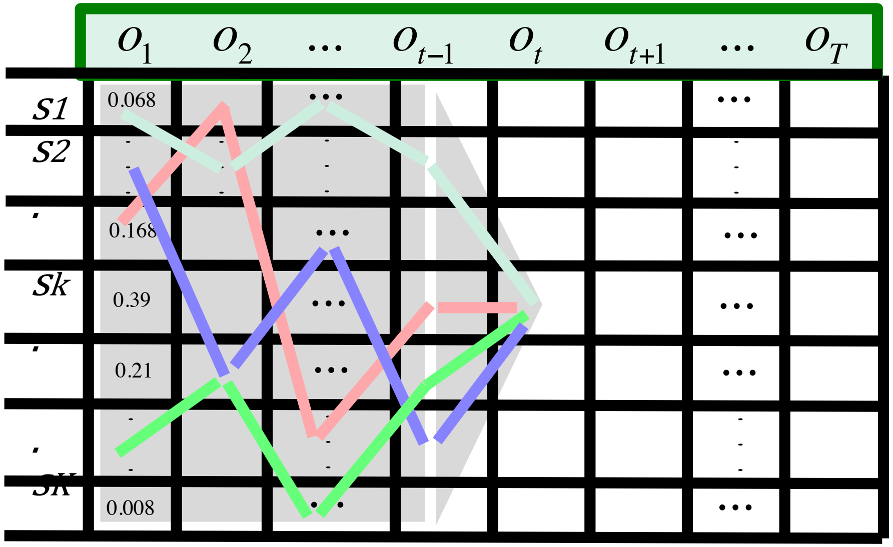
⇒ sum over every possible paths $P(O|M)=\sum_{1\le i,\le K^T} P(O,Q_i|M)$: K^T possible paths!
⇒ factorize the comon sub-paths

$P(o_1 o_2 o_3 \ldots o_t, q_t = k|M)$ is the probabily of observing the beginning of the sequence until time t and reaching state k (cell (t,k) of the lattice)

It is the sum over every possible sub-paths reaching state k at time t

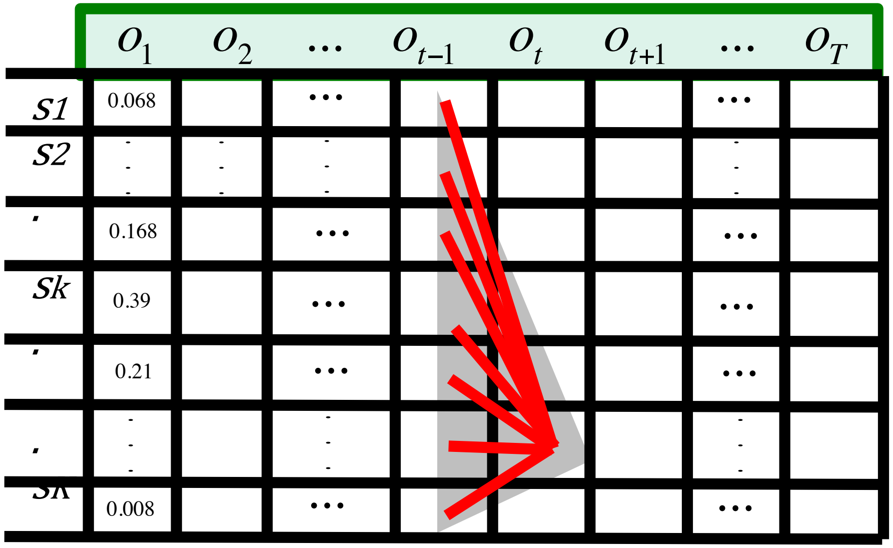
This is a « forward » probability
(forward till k,t)

$$\alpha_t(i) = P(o_1 o_2 o_3 \ldots o_t, q_t = i|M)$$



Course 3 : Forward Inference

at one time step, there are only N possible paths coming from t-1



	o_1	o_2	…	o_{t-1}	o_t	o_{t+1}	…	o_T
S1	0.068		…				…	
S2	⋮	⋮	⋮				⋮	
.	0.168		…				…	
Sk	0.39		…				…	
.	0.21		…				…	
.	⋮		⋮				⋮	
SK	0.008		…				…	

and we can write $P(o_1 o_2 \ldots o_t, q_t = i | M) = \sum_{1 \le k \le K} P(o_1 o_2 \ldots o_{t-1}, q_{t-1} = k | M) a_{ki} b_i(o_t)$

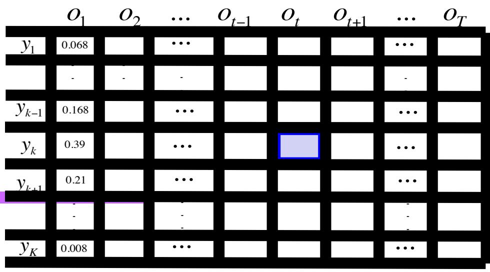
Thus the following recursion $\alpha_t(i) = \sum_{1 \le k \le K} \alpha_{t-1}(k) a_{ki} b_i(o_t)$

and finaly $P(O|M) = \sum_{1 \le k \le K} \alpha_T(k)$ complexity T x K^2 instead of K^T !!

The Forward Algorithm

```
Forward Algorithm
        Initialisation
            $\alpha_1(k) = \pi_k \times b_k(o_1) \qquad k = 1,\ldots,K$
        Recursion for t=2:T
            $\alpha_t(k) = \sum_{j=1}^{N} \alpha_{t-1}(j) a_{jk} b_k(o_t) \qquad k = 1,\ldots,K$
        Termination
            $P(O|M) = \sum_{i=1}^{N} \alpha_T(i)$
```

The Forward-Backward Inference



What is the *a posteriori* probability *of state* s_k at time t ? $P(q_t = s_k|O,M)$

The local posterior knowing the global observation

we write $\gamma_t(k) = P(q_t = s_k|O,M) = \dfrac{P(q_t = s_k, O|M)}{P(O|M)}$

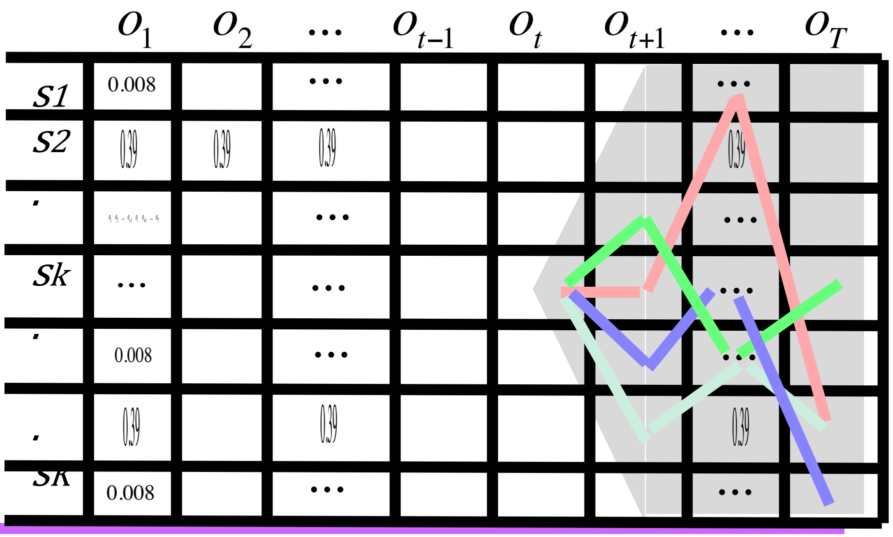
denominator: $P(O|M) = \sum_{1 \le k \le K} \alpha_T(k)$

The numerator writes : $P(q_t = s_k, O|M) = P(o_1 o_2 \cdots o_t, q_t = s_k|M) \times P(o_{t+1} o_{t+2} \cdots o_T|q_t = s_k, M)$

then $P(q_t = s_k, O|M) = \alpha_t(k) \times \beta_t(k)$

with $\beta_t(k) = P(o_{t+1} o_{t+2} \cdots o_T|q_t = s_k, M)$

The « backward » probability
of observing the end of the sequence till time t+1
while beeing in state K at time t.



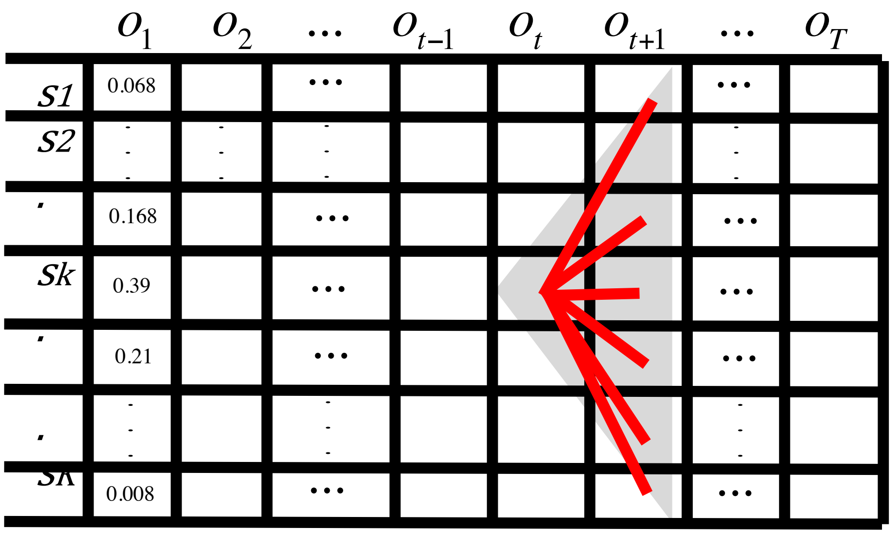
Course 3 : The Forward-Backward Inference

One time step backward, there are N possible paths and we write

$$P(o_{t+1}o_{t+2}...o_T|q_t = k, M) = \sum_{1 \le i \le K} a_{ki} b_i(o_{t+1}) P(o_{t+2}o_{t+3}...o_T|q_{t+1} = i, M)$$

Thus we can write a backward recursion :

$$\beta_t(k) = \sum_{1 \le i \le K} a_{ki} b_i(o_{t+1}) \beta_{t+1}(i)$$



and the following formulas hold

$$P(O|M) = \sum_{1 \le k \le K} \beta_1(k)$$

$$P(O|M) = \sum_{1 \le k \le K} \alpha_t(k) \beta_t(k)$$

$$P(O|M) = \sum_{1 \le k \le K} \alpha_T(k)$$

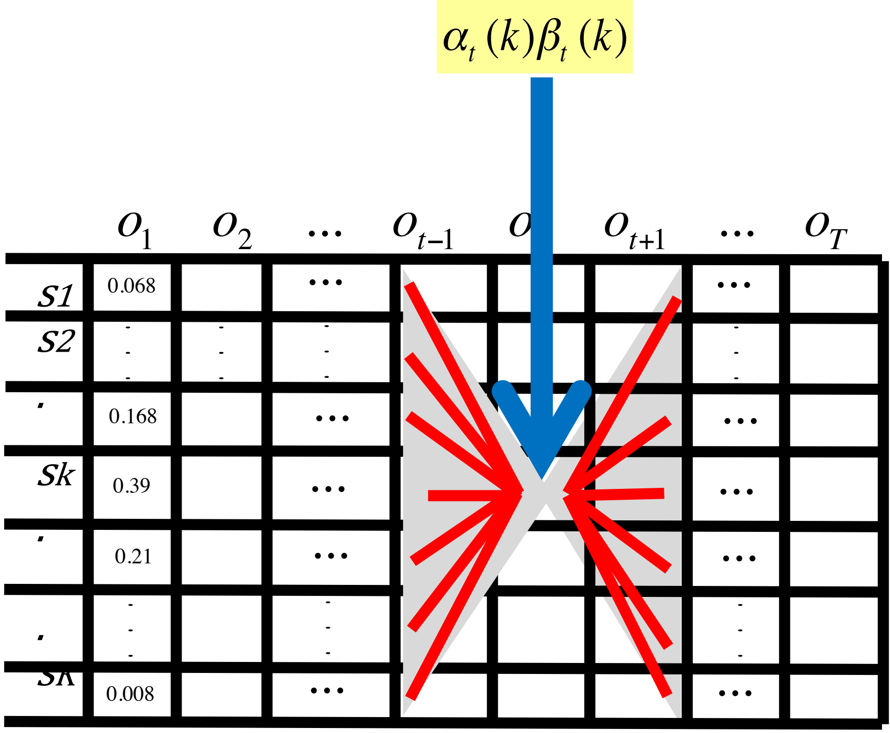
Course 3 : The Forward-Backward Inference

Finaly the *a posteriori* probability writes :

$$\gamma_t(k) = P(q_t = s_k | O, M) = \frac{\alpha_t(k)\beta_t(k)}{\sum_{1\le i\le K} \alpha_T(i)}$$

but also :

$$\gamma_t(k) = \frac{\alpha_t(k)\beta_t(k)}{\sum_{1\le i\le K} \alpha_t(i)\beta_t(i)}$$



	O_1	O_2	…	O_{t-1}	O	O_{t+1}	…	O_T
s1	0.068		…				…	
s2	- - -	- - -	- - -				- - -	
.	0.168		…				…	
Sk	0.39		…				…	
.	0.21		…				…	
.	- - -		- - -				- - -	
SK	0.008		…				…	

Training a HMM

We have a set of N training sequences $O = \left\{ O^{1 \le l \le N} \right\} = \left\{ \begin{pmatrix} o_1^l & o_2^l & \dots & o_{T_l}^l \end{pmatrix}^{1 \le l \le N} \right\}$

We want to model it by a HMM $M = \{A, B, \Pi\}$

The best model gets the highest likelihood (Maximum Likelihood (ML) criterion)

$$M^* = \underset{M}{\operatorname{argmax}} \left(\prod_{l=1}^{L} P(O^l | M) \right) \quad \text{with} \quad P(O^l | M) = \sum_{i \le k \le K} \alpha_T(k)$$

Problem: Computing the criterion requires the model to be known, and thus the statistics about the hidden states, or knowing the hidden states that are responsible for having generated each example.

This knowledge is missing, we don't know the sequence of labels (states) associated to each observation sequence (**similar to unsupervised classification**, but considering temporal depencies).

Solution: Extend the unsupervised classification techniques to sequences (Clustering, EM algorithm)

Introduce the unknown label sequence of each observation sequence Q^l

$O = \left\{ O^{1 \le l \le N} \right\}$ is the set of observation sequences : the incomplete data

$Q = \left\{ Q^{1 \le l \le N} \right\}$ is the set of label sequences : the unknown data

$Z = (O, Q)$ is the set of the complete data

Training a HMM

The EM Algorithm

Initialisation: - define one initial model M^0 (random guess, or k-means)
- set i = 0

E Step (Expectation): Estimate the missing data (the missing labels) using the current model M^i

$$P\left(q_t^l = s_k \middle| O^l, M^i\right) = \frac{\alpha_t(k)\beta_t(k)}{\sum_{1\le j\le K} \alpha_{T_l}(j)}$$

M Step (Maximisation): compute a better model M^{i+1} by improving the likelihood criterion

$$\mathrm{LV}\left(Z, M \middle| M^i\right) = \sum_{1\le l\le L} \sum_Q Log\left(P\left(O^l, Q^l = Q \middle| M\right)\right) P\left(O^l, Q^l = Q \middle| M^i\right)$$

```
    If   LV^i > LV^{i-1}      then
                   $i=i+1$ , goto step E

    else  $M^* = M^i$

End
```

Note : If the missing labels were known the criterion would simplify to

$$\mathrm{LV}(Z, M) = log\left(P(O, Q|M)\right) = log\left(\prod_{1\le l\le L} P\left(O^l, Q^l = Q \middle| M\right)\right) = \sum_{1\le l\le L} Log\left(P\left(O^l, Q^l = Q \middle| M^i\right)\right)$$

Training a discrete HMM

Finaly, with N training observation sequences

The general re-estimation formulas of discrete HMM are

$$\pi_i^* = \frac{1}{N} \sum_{l=1}^{N} \gamma_1^l(i)$$

$$b_j^*(o_t = v_k) = \frac{\sum_{l=1}^{N} \sum_{t=1, o_t = v_k}^{T_l} \gamma_t^l(j)}{\sum_{l=1}^{N} \sum_{t=1}^{T_l} \gamma_t^l(j)}$$

$$a_{ij}^* = \frac{\sum_{l=1}^{N} \sum_{t=1}^{T_k - 1} \xi_t^l(i,j)}{\sum_{l=1}^{N} \sum_{t=1}^{T_l - 1} \gamma_t^l(i)} = \frac{\sum_{l=1}^{N} \sum_{t=1}^{T_k - 1} \alpha_t^l(i) a_{ij} b_j(o_{t+1}^l) \beta_{t+1}^l(j)}{\sum_{l=1}^{N} \sum_{t=1}^{T_k - 1} \alpha_t^l(i) \beta_t^l(i)}$$

Training a continuous HMM

Use EM algorithm with the following reestimation formulas during M step

$$c_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{l=1}^{M} \gamma_t(j,l)}$$

$$\mu_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k).o_t}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

$$\Sigma_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k).(o_t - \mu_{jk})(o_t - \mu_{jk})^t}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

with

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \right] \left[\frac{c_{jk} N(o_t, \mu_{jk}, \Sigma_{jk})}{\sum_{m=1}^{M} c_{jm} N(o_t, \mu_{jm}, \Sigma_{jm})} \right]$$

The transition matrix A is estimated as in the discrete case

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)}$$

$$\pi_i^* = \frac{1}{N} \sum_{l=1}^{N} \gamma_1^l(i)$$

Training a HMM

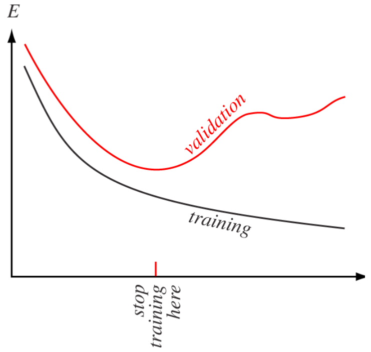
The ML criterion

Over-training:
The model overfit the training dataset with the risk to degrade on the test dataset (unseen data)

Ovoid over-training using a validation dataset during training
From the whole training W dataset build two subsets T & V: W = T + V = Training + Validation

Training on T with Likelihood criterion = L_T

Overtraining is detected when Error E on V starts increasing
Early Stopping criterion : $E_V^i > E_V^{i+1}$ *or* $E_V^i > E_V^{i+1}$ *for i=* i_0, … ,i_0+*N*



Extension : Cross-validation
Repeat the experiment on multiple partitions of W : A_i + V_i then average the performances

Training a HMM

Initialization :

Performance of the model depend on the initial model chosen to start EM

- A and π do not influence much
- *Initial value for B* is prominent

  - initialize B through a clustering technique (GMM) with no care for temporal information
  - eventually use Viterbi to train a first model

Model complexity :

Care about the number of free parameters to be estimated

- minimiser la complexité du modèle pour une meilleure estimation

solution 1: few states, small alphabet, few Gaussians in the mixture

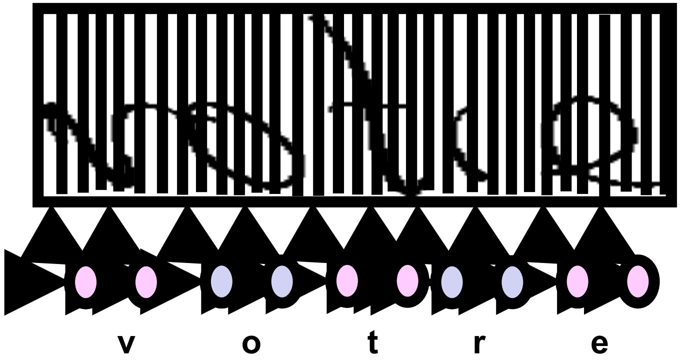
solution 2: introduce structural zero probabilities
(left-right, Bakis….)

solution 3: introduce tied states that share the same mixtures

Embedded Training

Train sub-units models (characters, phones…) in a continuous, unsegmented stream of data

- Only give the ground truth at the word, line, or paragraph
- Gather the missing statistics (E-step) with the whole HMM (word, line, paragraph)
- Update the parameters of the involved sub-units only M-step



```
i=0
Intitialize $M^i$
While L increases do
begin
    for each training example do
    begin
        build the HMM (concatenation of sub-units)
        compute forward et backward variables
        update L
        update the statistics of the sub-units involved
            $\sum_{t=1}^{T_k-1} \xi_t^k(i,j)$   et   $\sum_{t=1}^{T_k-1} \gamma_t^k(i)$
    end
    estimate new model parameters $M^{i+1}$of every sub-units
end
```

There is no proven result that convergence will reach optimal sub-units
Only convergence towards a better global model (EM convergence)
But works rather well for unsegmented data !!

HMM toolkits ressources

HTK : Hidden Markov Model Toolkit for continuous speech recognition
http://htk.eng.cam.ac.uk/

The historical reference toolkit, initially developped by microsoft, transfered to Cambridge University, not maintained anymore.

Include many functionalities for speech decoding, not only training HMM
Include HMM decoding using large lexicons and n-gram language models, and grammars

Have a look at the tutorial !!

KALDI : a more recent toolkit designed at the Johns Hopkins, Baltimore (USA), by Daniel Povey and his team http://kaldi-asr.org/doc/

Similar to HTK but still supported and having new developments

Very large lexicons are supported during decoding, in addition to high order language models, thanks to using weighted finite states transducers (WFST) allowing encoding lexicons and language model by WFST of characters.

We will give more insights about continuous speech and handwriting recognition algorithms later in this course.

Hybrid Neuro-HMM

They have been introduced in the continuous HMM framework so as to replace the GMM.

GMM are generative models trained to model each class, with unknown labels

they estimate the data likelihood $\mathrm{P}(X|s_k) = \sum_{1 \le j \le M} P(c_j) N(X, \mu_j, \Sigma_j)$

NN are discriminative models, requires known labels

they estimate class posteriors $\mathrm{P}(C_k|X)$ (softmax activation on the last layer)

Hidden states ***sk*** are the classes ***Ck*** to be discriminated by the NN

Introduce normalized likelihood scores at the output of the NN

$$\mathrm{P}(X|s_k) = \frac{\mathrm{P}(s_k|X) \times P(X)}{P(s_k)} \propto \frac{\mathrm{P}(s_k|X)}{P(s_k)}$$

We expect HMM working with high dimensionnal input features

Have better performance due to discriminative training of the data model

How to train the NN with unknown labels ?

Training a Neuro-HMM

Similar to HMM using EM Algorithm

Initialisation: - define one initial model $M^0 \Leftarrow (A^0, \Pi^0, NN^0)$ random guess, or k-means)
- set i = 0

E Step (Expectation): Estimate the missing data (the missing labels) using the current model M^i

$$P(q_t^l = s_k | O^l, M^i) = \frac{\alpha_t(k)\beta_t(k)}{\sum_{1\le j\le K}\alpha_{T_l}(j)}$$

M Step (Maximisation): compute a better model M^{i+1} by improving the likelihood criterion $LV(Z, M|M^i)$

update A^i, Π^i similarly to a HMM

train NN^{i+1} using the local posteriors of the HMM as the desired outputs $\left(t_1 \ldots \frac{\alpha_t(k)\beta_t(k)}{\sum_{1\le j\le K}\alpha_{T_l}(j)} \ldots t_K\right)$

need to modify the last layer update training formula of the NN (by considering every labels)

one should prefer in this case using the cross-entropy criteria O$= -\sum_{(X,T)\in Train}\sum_k t_k \ln y_k$

i=i+1

If LV^i > LV^{i-1} **then** goto step E

else $M^*, NN^* = M^i,\ NN^i$ **Stop**

End

HMM bibliography

J. A. Bilmes, A gentle Tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and Hidden Markov Models, Berkeley, 1998.

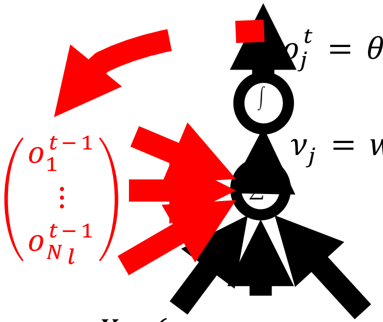
L. R. Rabiner, A tutorial on Hidden Markov Model and selected applications in speech recognition, IEEE proceedings, Vol. 77, pp. 257-286, 1989.

L.R. Rabiner & B.H. Juang, Fundamentals of Speech Recognition, Prentice Hall 1993.

Recurrent Neural Networks (RNN)

Recurrent Neural Networks have the hability to recognize patterns in sequential context by the introduction of recurrent units, and layers of recurrent units. They perform similar tasks as HMM or Neuro-HMM.

a single cell unit j of a RNN



$$\begin{pmatrix} o_1^{t-1} \\ \vdots \\ o_{N_l}^{t-1} \end{pmatrix}$$

$$o_j^t = \theta(v_j)$$

$$v_j = w_{0j} + \sum_{i=1}^{N_{l-1}} w_{ij} x_i + \sum_{i=1}^{N_l} w_{(i+N_{l-1})j} o_i^{t-1}$$

$$X_{t=} (x_{1t}\ x_{2t}\ \ldots x_{it} \ldots x_{N_{l-1}t})$$

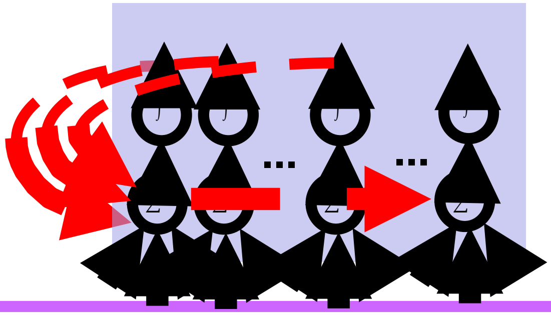
o_j^t is the output of unit j at time t
θ is the activation function of unit j at time t

w_{ij} the weights of unit j + the recurrent weights
w_{0j} its bias

the input vector of unit j

Computs a non-linear transformation of its inputs and its past outputs: need to memorise the past output of the cell to compute the forward pass

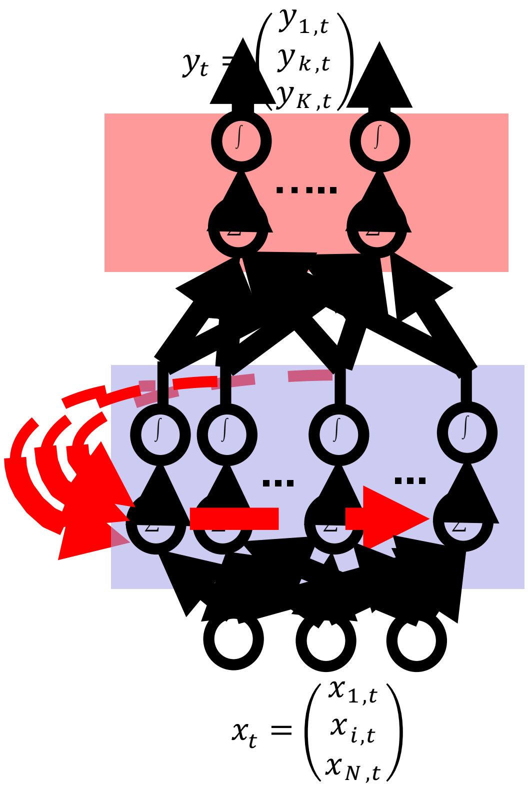
A reccurrent layer
every cell is recurrent with itself and any other cell of the layer



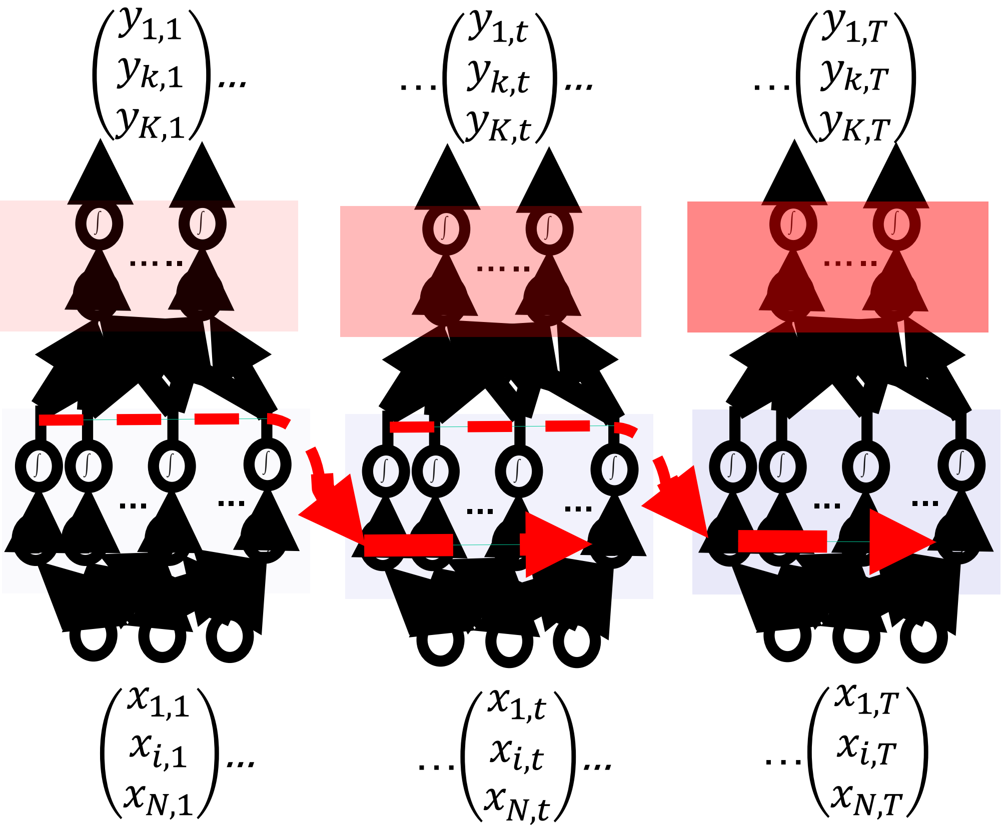
Recurrent Neural Networks (RNN)

A RNN architecture is typically composed of

- The input layer, fed by the observation vector x_t at time t,
- The recurrent layer, fed by x_t *and by its previous output* o_{t-1}
- The output layer (softmax),which provides the class *a posteriori* probabilities
- The network, that moves forward through time


$y_t = \begin{pmatrix} y_{1,t} \\ y_{k,t} \\ y_{K,t} \end{pmatrix}$
……
…
…
$x_t = \begin{pmatrix} x_{1,t} \\ x_{i,t} \\ x_{N,t} \end{pmatrix}$

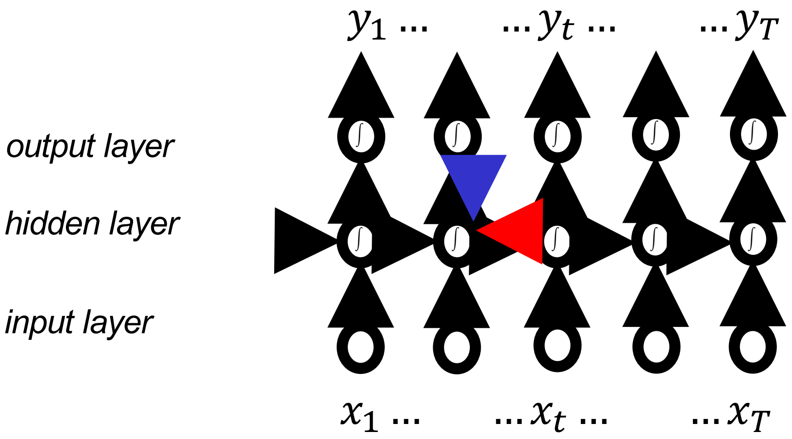
The unfolded RNN through time


$\begin{pmatrix} y_{1,1} \\ y_{k,1} \\ y_{K,1} \end{pmatrix}\dots$
$\dots\begin{pmatrix} y_{1,t} \\ y_{k,t} \\ y_{K,t} \end{pmatrix}\dots$
$\dots\begin{pmatrix} y_{1,T} \\ y_{k,T} \\ y_{K,T} \end{pmatrix}$
……
…
$\begin{pmatrix} x_{1,1} \\ x_{i,1} \\ x_{N,1} \end{pmatrix}\dots$
$\dots\begin{pmatrix} x_{1,t} \\ x_{i,t} \\ x_{N,t} \end{pmatrix}\dots$
$\dots\begin{pmatrix} x_{1,T} \\ x_{i,T} \\ x_{N,T} \end{pmatrix}$

Course 4 : Recurrent Neural Networks (RNN) for sequence analysis

Training a RNN with Back propagation through time (BPTT)

We can look at the unfolded network through time



∫ is a layer, not a cell

the gradients of the error at the hidden layer at time t depends on the gradients of the error at the output layer but also on the gradients of the error at the next time step (t+1) thus the name error back propagation throught time BPTT.

$$\delta_j^{l,t} = \left(\sum_{k=1,\dots,N_{l+1}} \delta_k^{l+1,t}\, w_{ik}^{l+1} + \sum_{k=1,\dots,N_l} \delta_k^{l,t+1}\, w_{ik}^{l}\right)\theta'\left(v_j^{l,t}\right)$$

Since weights are independent of time, we must sum the gradients over time, to get the weights update formula

$$w_{ij}^{l} = w_{ij}^{l} - \lambda \sum_{t=1,\dots T} \delta_j^{l,t} x_i^{l-1,t}$$

The output layer weights update formula is unchanged

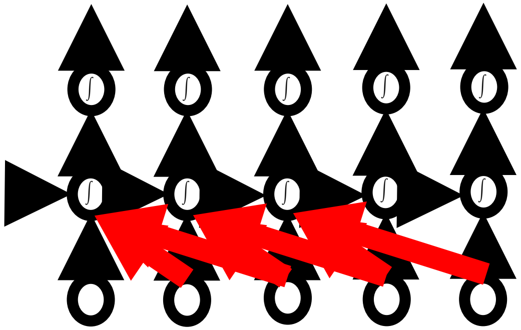
$$\delta_j^{L+1,t} = -\theta'\left(v_j^{L+1,t}\right)\left(t_{j,t} - y_{j,t}\right)$$

Bidirectional RNN

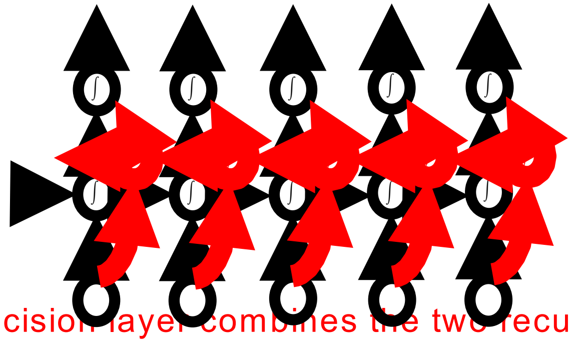
Standard RNN make a prediction at the output layer, by taking benefit from past observations, but there are many applications for which the future observations are known. For example every applications for which real time decisions are not necessary.

We would like to take benefit from the future observations to make a better decision

1- introduce the future context by looking multiple time steps ahead, but introduce more weights, but a fixed context



2- introduce a second Recurrent Layer in the opposite direction



The decision layer combines the two recurrent layers

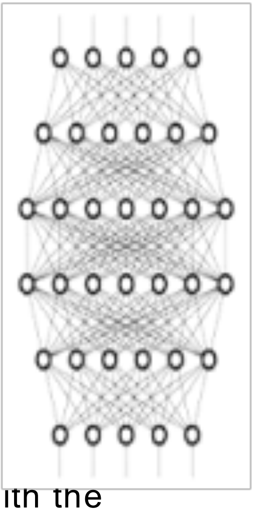
Training a Bidirectional RNN

1. Backward pass of the output layer
2. Backward pass of the forward layer
3. Backward pass of the backward layer

Long Short-Term Memory

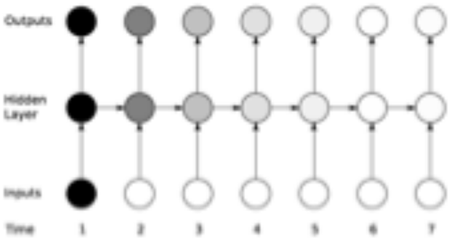
The gradient vanishing problem

Feed Forward Neural Networks achieve classification tasks with at most 2 Hidden Layers.



However, attempts to train Neural Network architectures with more layers have been confronted to gradient vanishing during the backward pass.
The gradient is small and its value decreases through the layers,
thus training the first layers is difficult. Deep Learning has been introduced (2007), to overcome this limitation (un-supervised pre-training one layer after the other,
Fine tuning the whole Network at last, introducing ReLu units).
Support training with very large datasets…

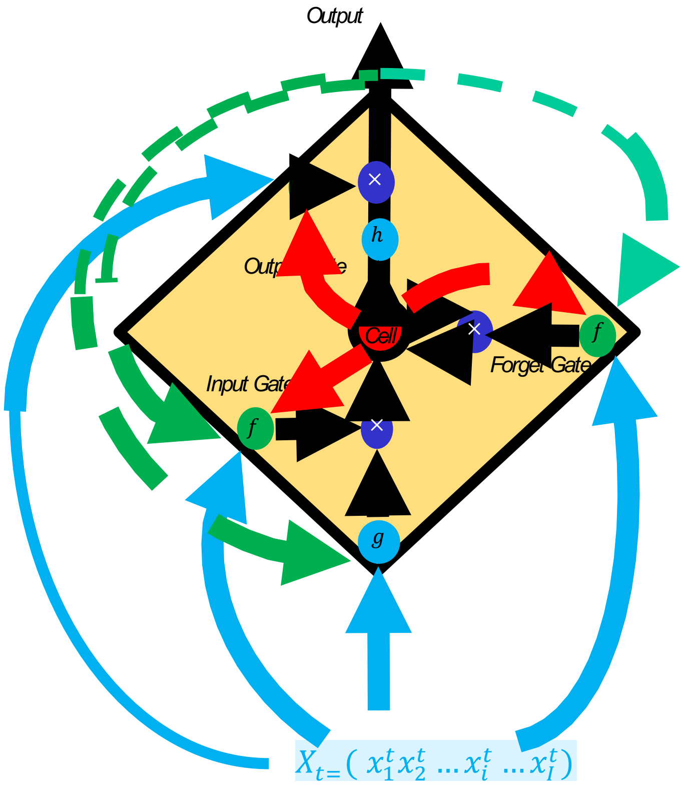
Recurrent Neural Networks, although introduced many years ago, have been confronted with the gradient vanishing problem from their conception, as they need to learn sequential data. The depth of the unfolded network is equal to the longer sequence to be analyzed. Thus the gradient vanishes through time, and the network cannot learn long time dependencies, which was the expected strength of such networks compared to HMM.



Long Short-Term Memory

Long Short-Term Memory cells (mémoire longue à court terme) have been introduced by Hochreiter & Schmidhuber (1997) so as to encompass the limitations of standard RNN

Ideas: introduce a memory mechanism (to prevent vanishing values) with update policy

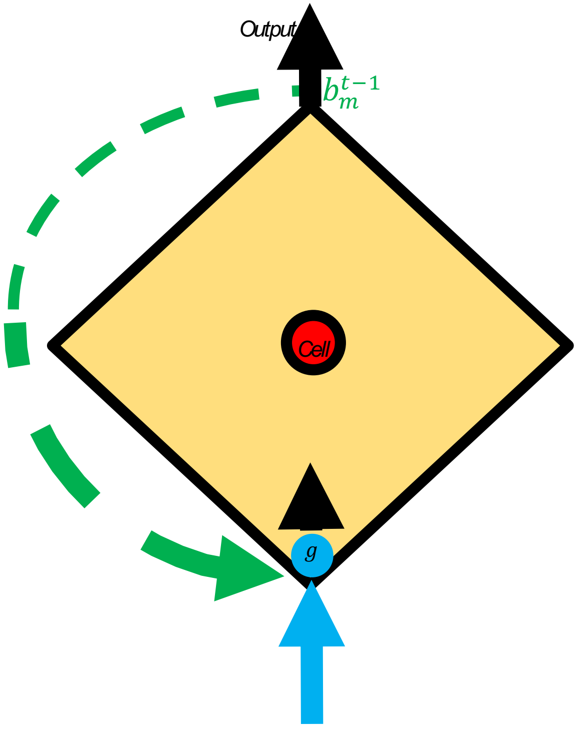


- *f* there are 3 recurrent control gates with sigmoid activation functions
- *g* 1 standard recurrent unit with sigmoid or *tanh*
- ● C memory cells (C=1)
- ◆ there are M memory blocs
- *h* *h* is preferably a *tanh* activation

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation , 9(8):1735-1780, 1997.

Long Short-Term Memory

Long Short-Term Memory cells (mémoire longue à court terme)
have been introduced by Hochreiter & Schmidhuber (1997) so as to encompass the limitations of standard RNN
Ideas: introduce a memory mechanism (to prevent vanishing values) with update policy



$X_{t=}(x_1^t x_2^t \dots x_i^t \dots x_I^t)$

1

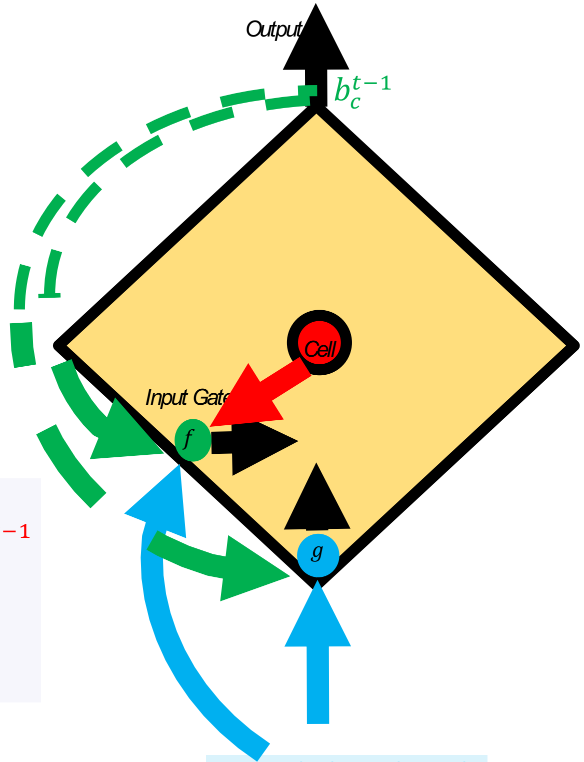
$$a_m^t = \sum_{i=1}^{I} w_{ic}\, x_i^t + \sum_{m=1}^{M} w_{mc} b_m^{t-1}$$

$$b_m^t = g(a_m^t)$$

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation , 9(8):1735-1780, 1997.

Long Short-Term Memory

Long Short-Term Memory cells (mémoire longue à court terme)
have been introduced by Hochreiter & Schmidhuber (1997) so as to encompass the limitations of standard RNN
Ideas: introduce a memory mechanism (to prevent vanishing values) with update policy



2

$$a_I^t = \sum_{i=1}^{I} w_{iI}\, x_i^t + \sum_{m=1}^{M} w_{uI} b_m^{t-1} + \sum_{c=1}^{C} w_{cI} s_c^{t-1}$$

$$b_I^t = f(a_I^t)$$

1

$$a_c^t = \sum_{i=1}^{I} w_{ic}\, x_i^t + \sum_{m=1}^{M} w_{mc} b_m^{t-1}$$

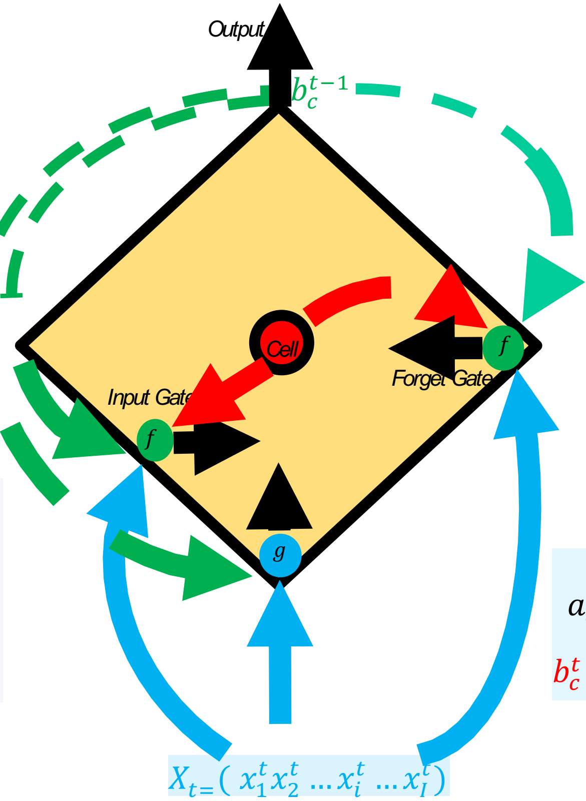
$$b_c^t = g(a_c^t)$$

$X_{t=}(x_1^t x_2^t \dots x_i^t \dots x_I^t)$

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation , 9(8):1735-1780, 1997.

Long Short-Term Memory

Long Short-Term Memory cells (mémoire longue à court terme)
have been introduced by Hochreiter & Schmidhuber (1997) so as to encompass the limitations of standard RNN
Ideas: introduce a memory mechanism (to prevent vanishing values) with update policy



3

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi}\, x_i^t + \sum_{m=1}^{M} w_{m\phi} b_m^{t-1} + \sum_{c=1}^{C} w_{c\phi} s_c^{t-1}$$

$$b_\phi^t = f(a_\phi^t)$$

2

$$a_I^t = \sum_{i=1}^{I} w_{iI}\, x_i^t + \sum_{m=1}^{M} w_{uI} b_m^{t-1} + \sum_{c=1}^{C} w_{cI} s_c^{t-1}$$

$$b_I^t = f(a_I^t)$$

1

$$a_c^t = \sum_{i=1}^{I} w_{ic}\, x_i^t + \sum_{m=1}^{M} w_{mc} b_m^{t-1}$$

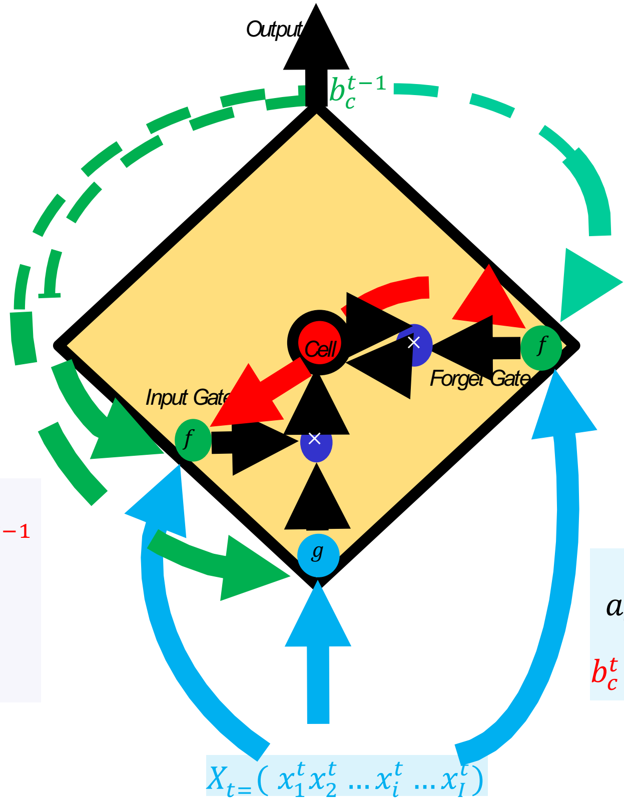
$$b_c^t = g(a_c^t)$$

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation , 9(8):1735-1780, 1997.

Long Short-Term Memory

Long Short-Term Memory cells (mémoire longue à court terme)
have been introduced by Hochreiter & Schmidhuber (1997) so as to encompass the limitations of standard RNN

Ideas: introduce a memory mechanism (to prevent vanishing values) with update policy



1

$$a_c^t = \sum_{i=1}^{I} w_{ic}\, x_i^t + \sum_{m=1}^{M} w_{mc} b_m^{t-1}$$

$$b_c^t = g(a_c^t)$$

2

$$a_I^t = \sum_{i=1}^{I} w_{iI}\, x_i^t + \sum_{m=1}^{M} w_{uI} b_m^{t-1} + \sum_{c=1}^{C} w_{cI} s_c^{t-1}$$

$$b_I^t = f(a_I^t)$$

3

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi}\, x_i^t + \sum_{m=1}^{M} w_{m\phi} b_m^{t-1} + \sum_{c=1}^{C} w_{c\phi} s_c^{t-1}$$

$$b_\phi^t = f(a_\phi^t)$$

4

$$s_c^t = b_\phi^t s_c^{t-1} + b_I^t b_c^t$$

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation , 9(8):1735-1780, 1997.

Long Short-Term Memory

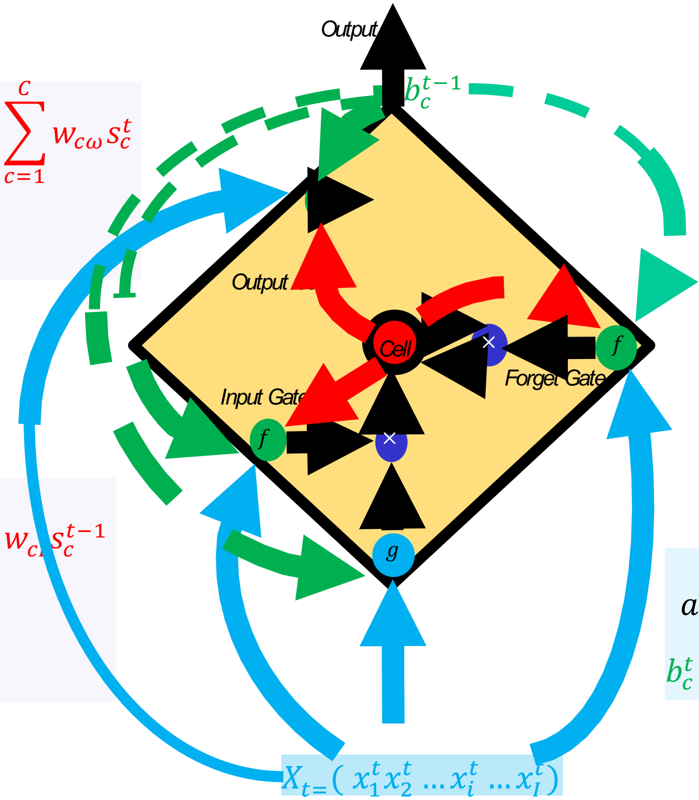
Long Short-Term Memory cells (mémoire longue à court terme)
have been introduced by Hochreiter & Schmidhuber (1997) so as to encompass the limitations of standard RNN

Ideas: introduce a memory mechanism (to prevent vanishing values) with update policy

5

$$a_\omega^t = \sum_{i=1}^{I} w_{i\omega}\, x_i^t + \sum_{m=1}^{M} w_{m\omega} b_m^{t-1} + \sum_{c=1}^{C} w_{c\omega} s_c^t$$

$$b_w^t = f(a_w^t)$$



4

$$s_m^t = b_\phi^t s_c^{t-1} + b_I^t b_c^t$$

3

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi}\, x_i^t + \sum_{m=1}^{M} w_{m\phi} b_m^{t-1} + \sum_{c=1}^{C} w_{c\phi} s_c^{t-1}$$

$$b_\phi^t = f(a_\phi^t)$$

2

$$a_I^t = \sum_{i=1}^{I} w_{iI}\, x_i^t + \sum_{m=1}^{M} w_{uI} b_m^{t-1} + \sum_{c=1}^{C} w_{cI} s_c^{t-1}$$

$$b_I^t = f(a_I^t)$$

1

$$a_c^t = \sum_{i=1}^{I} w_{ic}\, x_i^t + \sum_{m=1}^{M} w_{mc} b_m^{t-1}$$

$$b_c^t = g(a_m^t)$$

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation , 9(8):1735-1780, 1997.

Long Short-Term Memory

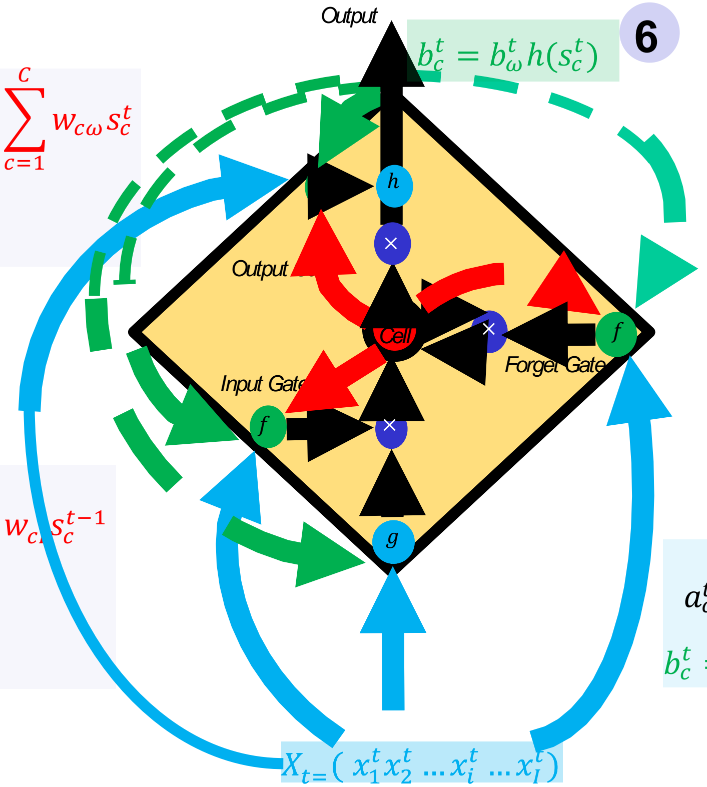
Long Short-Term Memory cells (mémoire longue à court terme)
have been introduced by Hochreiter & Schmidhuber (1997) so as to encompass the limitations of standard RNN

Ideas: introduce a memory mechanism (to prevent from vanishing values) with update policy

5

$$a_\omega^t = \sum_{i=1}^{I} w_{i\omega}\, x_i^t + \sum_{m=1}^{M} w_{m\omega}\, b_m^{t-1} + \sum_{c=1}^{C} w_{c\omega}\, s_c^t$$

$$b_w^t = f(a_w^t)$$



6

$$b_c^t = b_\omega^t\, h(s_c^t)$$

4

$$s_c^t = b_\phi^t\, s_c^{t-1} + b_I^t\, b_c^t$$

3

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi}\, x_i^t + \sum_{m=1}^{M} w_{m\phi}\, b_m^{t-1} + \sum_{c=1}^{C} w_{c\phi}\, s_c^{t-1}$$

$$b_\phi^t = f(a_\phi^t)$$

2

$$a_I^t = \sum_{i=1}^{I} w_{iI}\, x_i^t + \sum_{m=1}^{M} w_{uI}\, b_m^{t-1} + \sum_{c=1}^{C} w_{cI}\, s_c^{t-1}$$

$$b_I^t = f(a_I^t)$$

1

$$a_c^t = \sum_{i=1}^{I} w_{ic}\, x_i^t + \sum_{m=1}^{M} w_{mc}\, b_m^{t-1}$$

$$b_c^t = g(a_c^t)$$

$$X_{t=}(\, x_1^t x_2^t \dots x_i^t \dots x_I^t)$$

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation , 9(8):1735-1780, 1997.

Training a LSTM with CTC

Training LSTM with Connectionist Temporal Classification

Training any RNN, like training any NN, requires **having the ground truth** at each time step to be known. This cannot be considered as it is too tedious a task to label sequetial data at such a low level.

Generally one provides the ground truth at word, or sentence level (speech or handwriting)

Similarly to HMM or Neuro-HMM we could use the **Forward-Backward EM Embedded Training** algorithm
E. Senior, T. Robinson, Forward Backward retraining of recurrent neural networks, Neural Information Processing Systems, 1996.

Alex Graves introduced a variant of the Forward Backward Embedded Training algorithm, called the **Connectionist Temporal Classification (CTC)**.

- The HMM layer is reduced to an **automaton where transitions between states have 0/1 weight** (no transition probabilities anymore).
- Introduce an additional specific **blank state** acting like a non character state

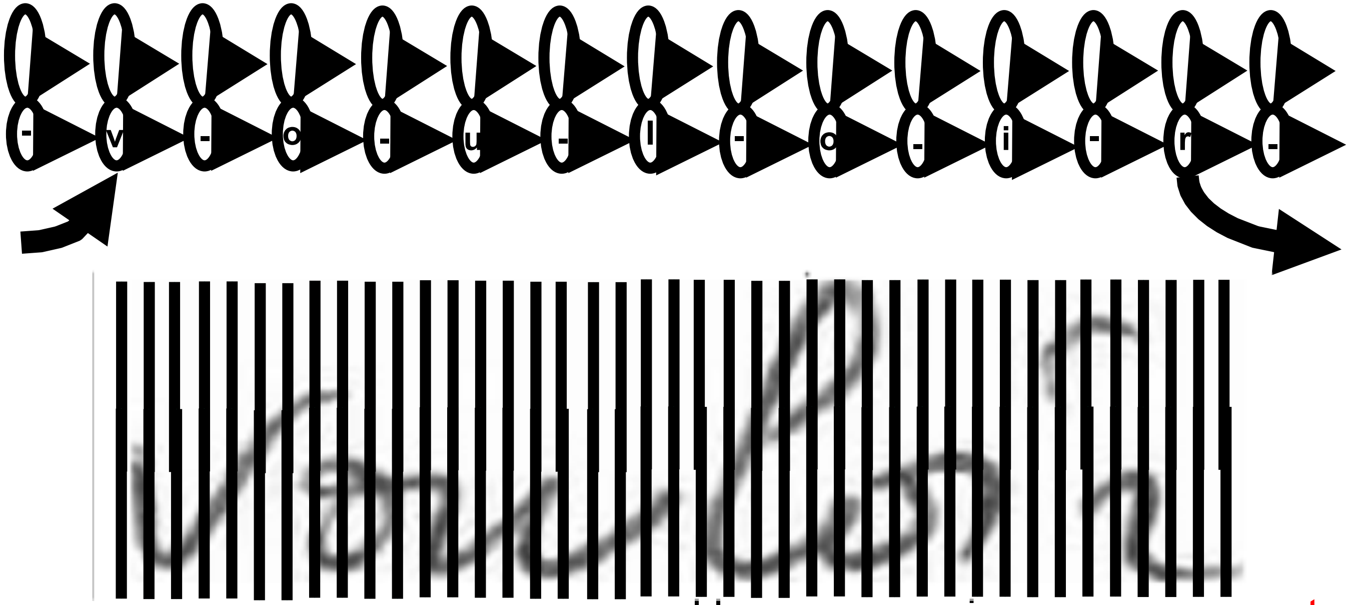
A ground truth sequence of characters is modified to include a **blank label** between characters.

vouloir -v-o-u-l-o-i-r-

Training a LSTM with CTC

CTC automaton alignment
During training we force the RNN to align to the paths accepted by the automaton only

The network may recognize a character spanning multiple frames



- - v v- - - -o - - - u - - - - - - - - - l l - - - - o - - - i - - - - r - - -

- v vv v -oooo - - u u u u u - - l l l l l - o o o- - i - r r r r r r -

two accepted paths

Notice that by the introduction of the blank label we are **not able to locate the characters** anymore

Training a LSTM with CTC

Training RNN with CTC

Assume the local class conditional *posteriors* are independant, thus the probability of a path is the product of the local class *posterior* probabilities

--vv-- --o---u------ --ll ----o---i-- --r--- one accepted path π

$$P(\pi|X) = \prod_{t=1}^{T} y_{\pi_t,t}$$

$y_{\pi_t,t}$ the probability of the class on π at position t

we denote by l the labelling sequence : l = -v-o-u-l-o-i-r-

Then the probability of the network to produce the labelling whatever the path π is the sum over every possible paths accepted by the CTC automaton

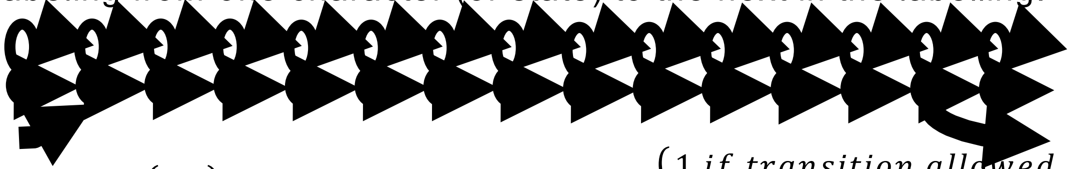
$$P(l|X) = \sum_{\pi \epsilon \mathcal{F}^{-1}(l)} P(\pi|X)$$

with $\mathcal{F}^{-1}(l)$ the set of accepted paths matching the sequence l

Training a LSTM with CTC

Training RNN with CTC

We can represent the CTC automaton with a binary transition matrix A encoding the possible paths along the labeling from one character (or state) to the next in the labelling.



A= $\left(a_{ij}\right)_{i,j=1,\ldots,2U+1}$ with $a_{ij} = \begin{cases} 1 \; if \; transition \; allowed \\ 0 \; otherwise \end{cases}$

Then, the **forward probability** of a partial labelling $l(1{:}j)$ ending at position j with state $l(j)$ knowing the observation X until position t writes

$$\alpha(t,j) = P(l(1{:}j)|x_1 \ldots x_t)$$

And we can write the following recursion

$$\alpha(t,j) = y_{l(j),t} \sum_{i=1}^{2U+1} \alpha(t-1,i)a_{ij}$$

With the initial conditions

$$\alpha(1,l(1)) = y_{l(1),1}$$

$$\alpha(1,l(2)) = y_{l(2),1}$$

$$\alpha(1,l(i)) = 0 \quad \forall\, i > 2$$

Training a LSTM with CTC

Training RNN with CTC

Similarly we can write the backward probability

$$\beta(t,i) = P(l(i\!:2U+1)|x_{t+1} \dots x_T)$$

And we can write the following recursion

$$\beta(t,i) = \sum_{j=1}^{2U+1} \beta(t+1,j) a_{ij} y_{l(j),t+1}$$

Initial conditions

$$\beta(T,2U+1) = 1$$
$$\beta(T,2U) = 1$$
$$\beta(1,l(i)) = 0 \quad \forall\, i < 2U$$

Finally we can deduce the local class *posterior* probabilities with the forward bckward product

$$\alpha(t,j)\beta(t,j) = P(\pi_t = l(j)|X)$$

Summing over every possible classes at position t, we get the probability of the labelling

$$P(l|X) = \sum_{j=1,\dots,2U+1} \alpha(t,j)\beta(t,j)$$

Training a LSTM with CTC

Training RNN with CTC, similar to NN-HMM using EM like Algorithm

Initialisation: - define one initial model RNN^0
- set i = 0

E Step (Expectation): Estimate the missing data (the missing labels) using the current model RNN^i

$$P(\pi_t = l(j)|X) = \alpha(t,j)\beta(t,j)$$

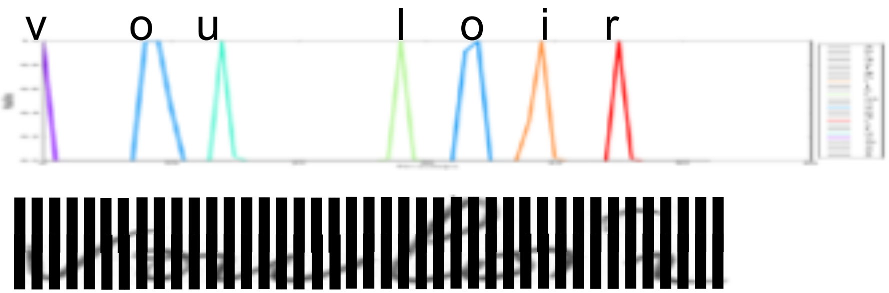
M Step (Maximisation):

train RNN^{i+1} using the local *posteriors* as the desired outputs and using the cros-entropy criterion

while criterion improves **goto** i=i+1 step E

else $RNN^* = RNN^i$ **Stop**

Typical outputs of LSTM trained with CTC and blank label



Adding the blank label allows the network have no classification decision at some frames

Whereas it provides high probabilities of the characters at very few positions (1 or 2 consecutive frames)

Don’t know where the characters are exactly !!

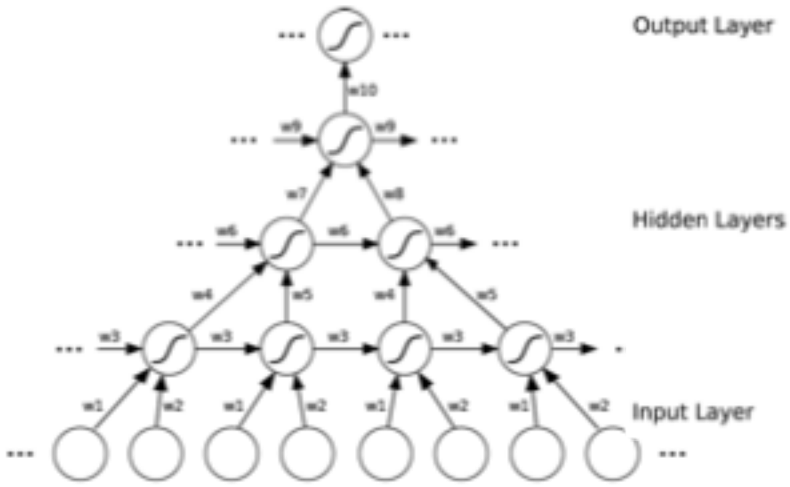
BLSTM Hierarchical architecture

Stacked BLSTM Layers feed the decision layer

Interest: similar to NN or Deep NN, make the RNN **learn representations (features)** and progressively get more discriminative features in the upper layers.

Hierarchical architectures introduce a kind of subsampling operator but more elaborated than the traditional subsampling operator which simply forgets a proportion of the information.

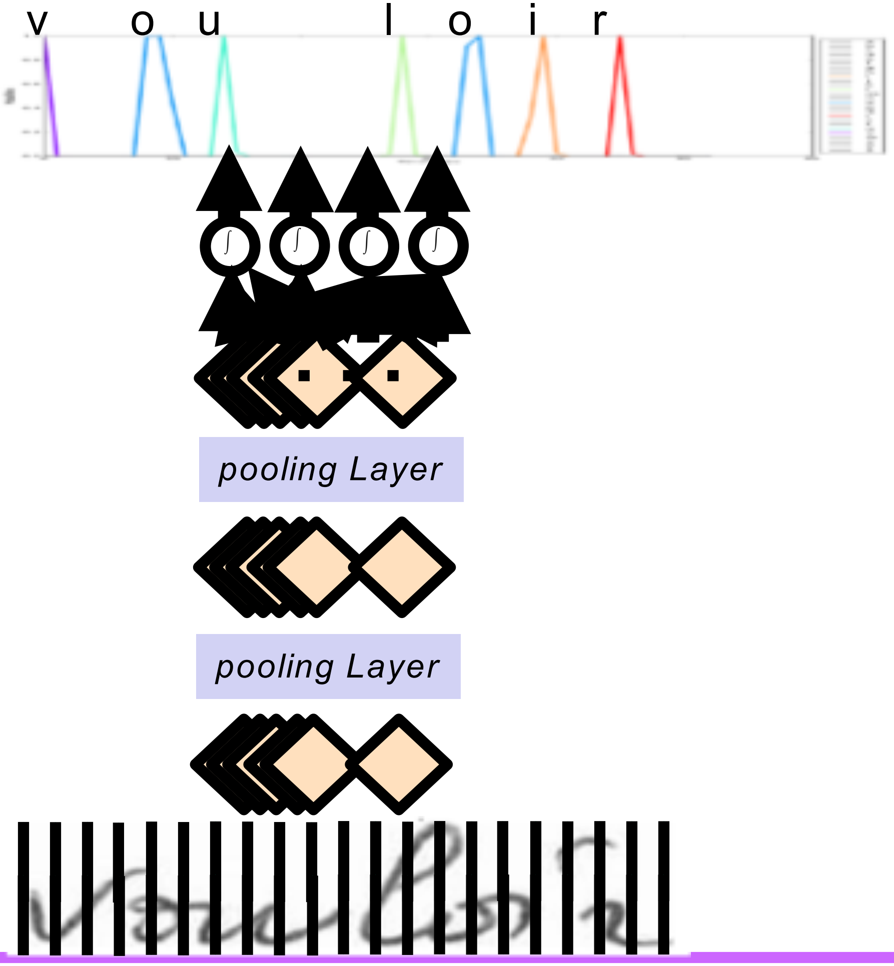
Allows having **long range dependencies** without having to cope with too large windows (and weights)



- *Example with 3 hidden layers and a sub sampling factor of S=2*
- *Each unit of one hidden layer has an observation window of size S=2 frames.*
- *The observation window of each layer moves with a stride of S=2 frames along its input (no overlap between windows)*
- *The size of the gates is impacted by doubling the size of the input vector at each time frame compared to non hierarchical architectures*

BLSTM Hierarchical architecture

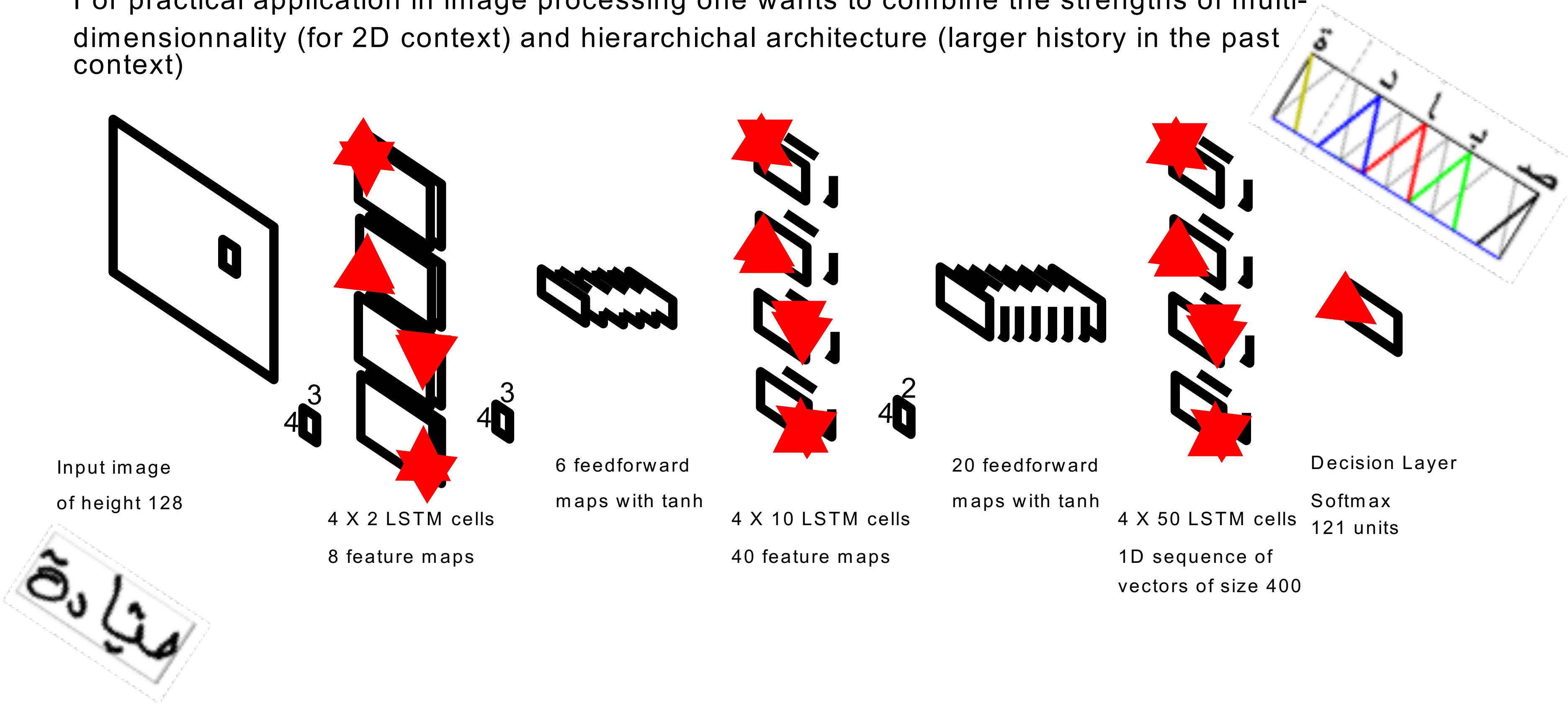
LSTM Neural Networks architecture
Stacked BLSTM Layers feed the decision layer



LSTM Hierarchical multidimentional architectures

BLSTM hierarchical multidimentional architecture (Graves 2008)

For practical application in image processing one wants to combine the strengths of multi-dimensionnality (for 2D context) and hierarchichal architecture (larger history in the past context)



Input image
of height 128

4 X 2 LSTM cells
8 feature maps

6 feedforward
maps with tanh

4 X 10 LSTM cells
40 feature maps

20 feedforward
maps with tanh

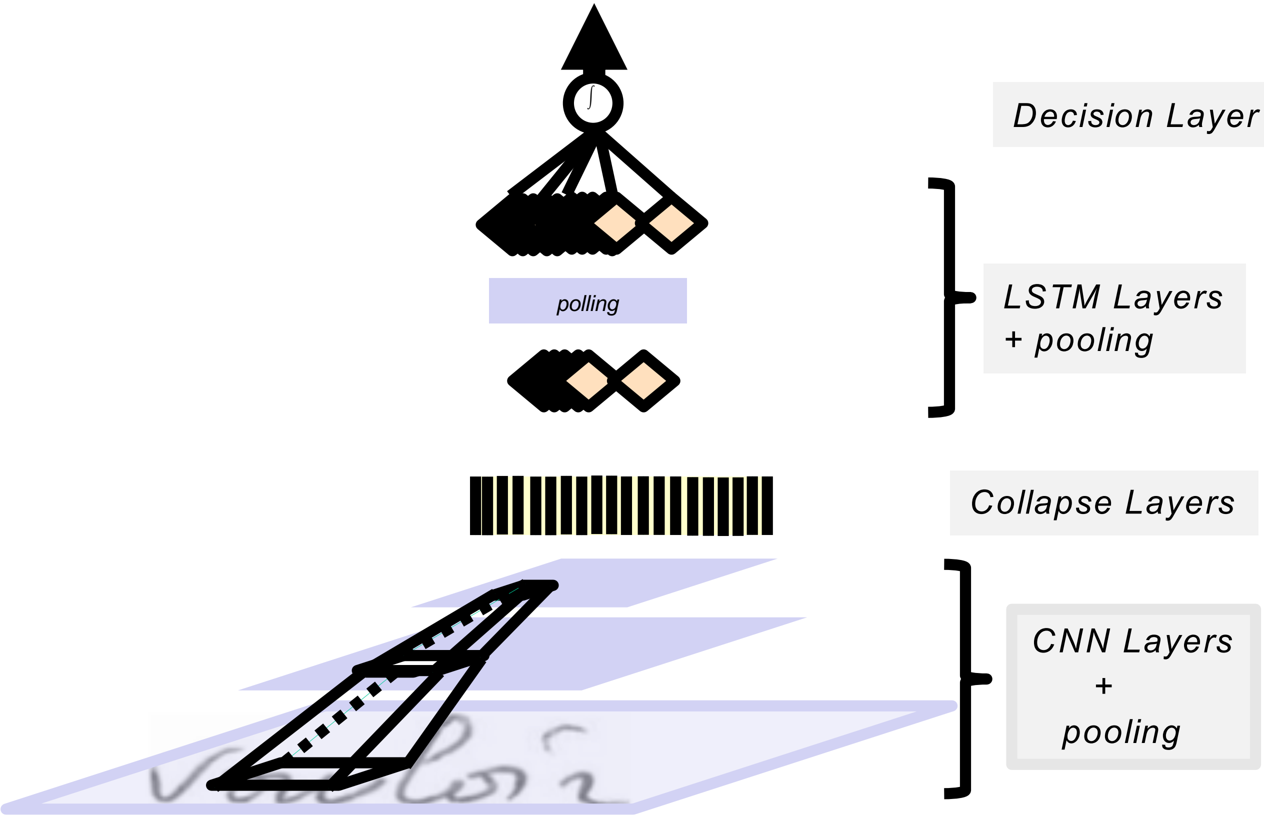
4 X 50 LSTM cells
1D sequence of
vectors of size 400

Decision Layer
Softmax
121 units

CNN-RNN architectures

-- The learnt representations may capture long term depencies that may not generalize well
-- training a RNN does not parallelize well

++ CNN can account for local 2D features in a natural way
++ RNN are powerful models to learn sequential representations



Decision Layer

polling

LSTM Layers
+ pooling

Collapse Layers

CNN Layers
+
pooling

Neural Networks & Recurrent Neural Networks Bibliography

C. M. Bishop. Neural Networks for Pattern Recognition . Oxford University Press, 1995.

Alex Graves, Supervised Sequence Labelling with Recurrent Neural Networks, PhD dissertation, Technischen Universitat Munchen, 2008.

Théodure Bluche, Deep Neural Networks for Large Vocabulary Handwritten Text Recognition, thèse de doctorat de l'université Paris Sud, 2015.

Applications

Modèles

- Modèles statistique de langage
- Modèles de Markov discrets et continus (HMM)
- ~~Champs Aléatoire Conditionnels (CRF~~)
- Modèles Neuro-Markoviens
- Réseaux de neurones récurrents (RNN – BLSTM)

Applications

- Parole et écriture
- Mise en œuvre

Extensions

- Modèles neuronaux à attention

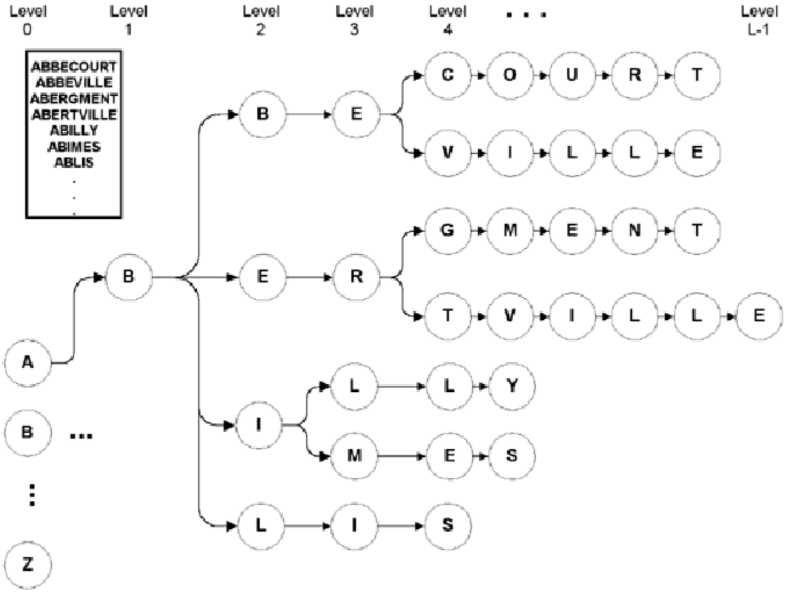
Applications to Speech and Handwriting Recognition

Isolated word recognition

Whatever the type of model (HMM or RNN) word recognition consists in searching the best hypothesis belonging to a lexicon : **Lexicon Driven Recognition**

$$M^* = \underset{i}{\operatorname{argmax}}\big(P(O|M_i)\big)$$

- ✓ Decoding with a « flat lexicon » is too expensive, use a prefix tree intead



Applications to Speech and Handwriting Recognition

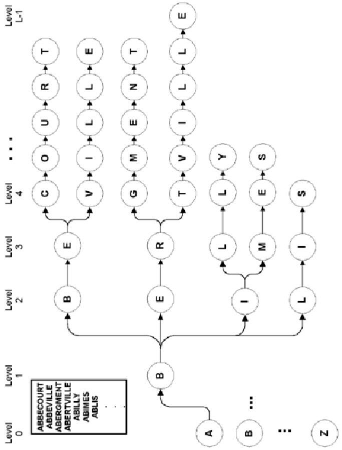
Isolated word recognition

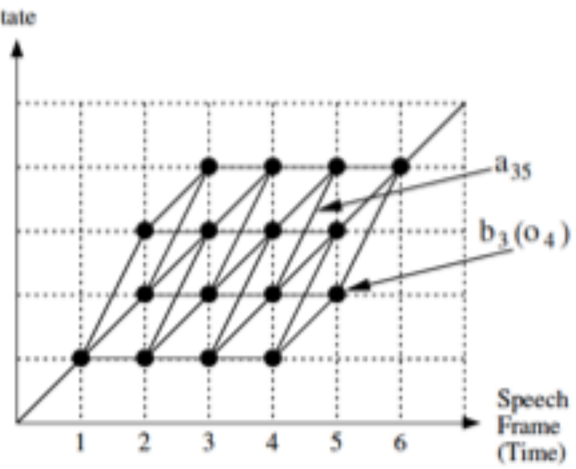
Viterbi decoding constrained by the prefix tree $P(O|M_i) \cong P(O,Q^*|M_i) = \max\limits_{1\le k\le K}\left(\delta_T(k)\right)$

Allows implementing different optimization technics for

- dealing with large lexicons (10K, 20K,….60K,…) in nearly real time
- integrate language models during the decoding phase : **continuous speech recognition**

$$M^*, Q^* = \operatorname*{argmax}_{i,j}\left(P\left(O, Q_j | M_i\right)\right)$$





Continuous Speech and Handwriting Recognition

The recognition process consists in searching the best word sequence $W^* = (w_1 \quad w_2 \quad \dots \quad w_m)$ matching the observation $O = (o_1 , \quad o_2 , \dots \quad o_t , \quad \dots \quad o_T)$

$$W^* = \underset{W}{\operatorname{argmax}} P(W|O) = \underset{W}{\operatorname{argmax}} P(O|W) P(W)$$

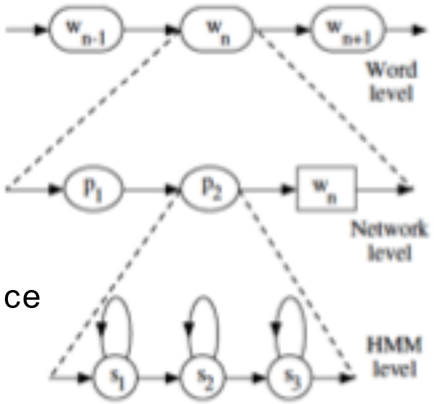
$P(O|W)$ is the phonetic or optical model likelihood

$P(W)$ is the language model likelihood.

Viterbi decoding using the only admissible character transitions encoded by the prefix tree and the admissible transition by **the language model**.
Use of a state graph to encode the whole admissible sequence of characters with their respective probabilities

- as many states as there are states in words in sentences…
- transitions between words are the LM probabilities

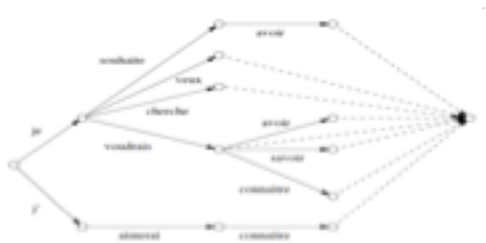
Decoding is finding the best path in this graph, matching the observation sequence



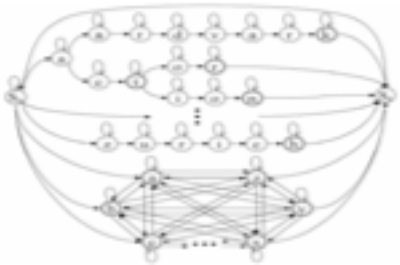
Continuous Speech and Handwriting Recognition

Finite-State Transducers in Language and Speech Processing, M. Mohri, 1997

Continuous speech recognition

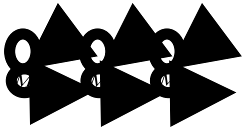


WFST G : Word sequences follow the LM rules of the statistical language model (n-gram)



WFST L : Characters sequences follow some other syntactical rules (lexicon) Finite State Transducer (FST)

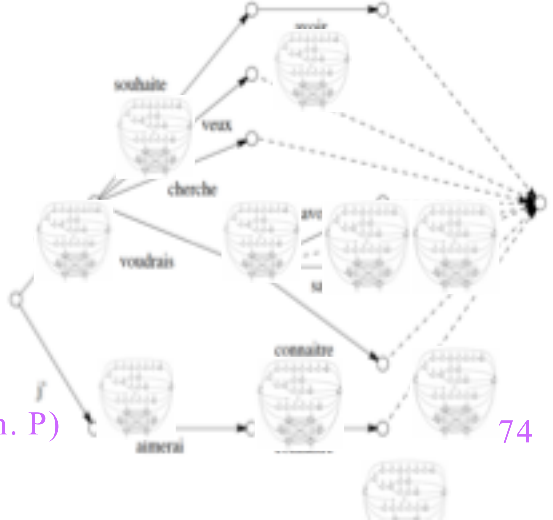
Encoding regular expressions as well (alphanumeric expressions)



WFST T Left-Right Character HMM or CTC Automaton are the lowest models giving the final state transition rules;

They can be encoded with a

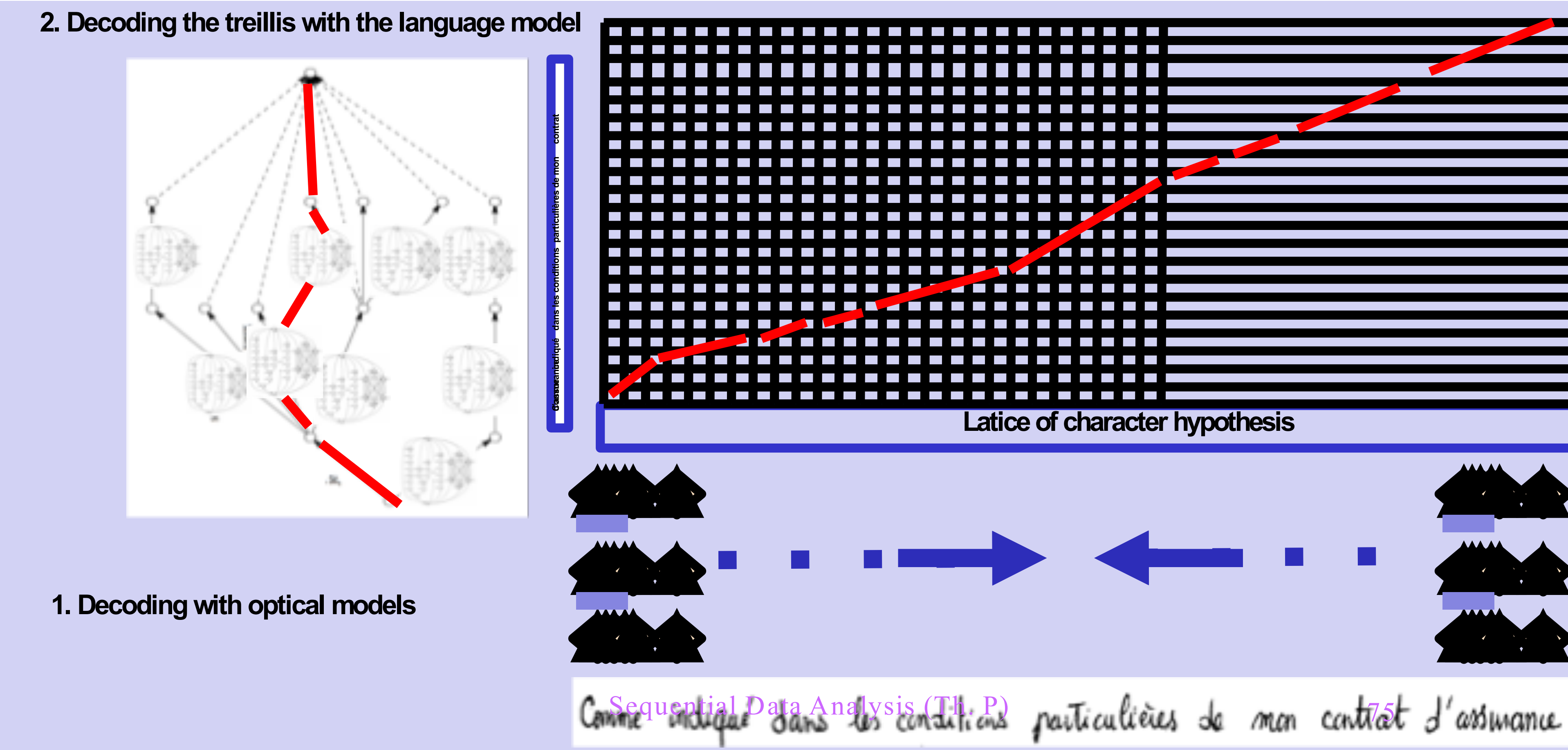
T o min(det(L o G)) *:* provides the Transducer of the model



Continuous Speech and Handwriting Recognition

Continuous speech recognition

Find the most probable path, matching the optical models accepted by the Language & Lexicon Automaton
=> optimisation problem


2. Decoding the treillis with the language model
Comme indiqué dans les conditions particulières de mon contrat d'assurance
Latice of character hypothesis
1. Decoding with optical models
Comme indiqué dans les conditions particulières de mon contrat d'assurance

Modèles à attention

Modèles

- Modèles statistique de langage
- Modèles de Markov discrets et continus (HMM)
- ~~Champs Aléatoire Conditionnels (CRF~~)
- Modèles Neuro-Markoviens
- Réseaux de neurones récurrents (RNN – BLSTM)

Applications

- Parole et écriture
- Mise en œuvre

Extensions

- Modèles neuronaux à attention

Towards End to End NN architectures

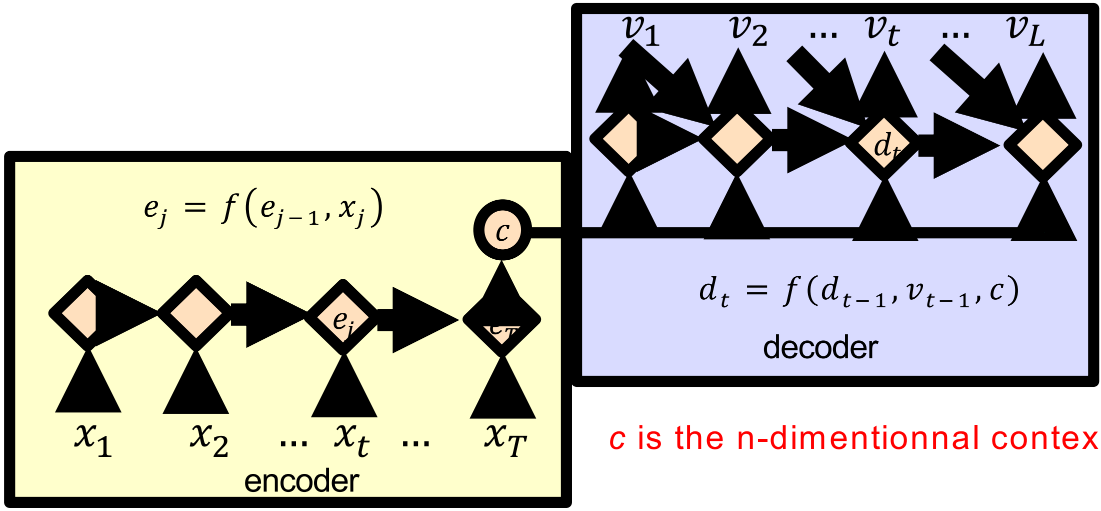
CNN-RNN are powerful models to learn sequential representations in signal and images

RNN are also competitive language models combined with n-gram

- *Recurrent neural* network *based language model,T. Mikolov, et al. 2010.*

RNN are also powerful language generators, and translators

- *Learning Phrase Representations using* ***RNN Encoder-Decoder*** *for Statistical Machine Translation, Cho et al., EMNLP 2014*
- *Sequence to Sequence Learning with Neural Networks, Sutskever, et al. , NIPS 2014.*



v_1 v_2 … v_t … v_L
d_t
$e_j = f(e_{j-1}, x_j)$
c
e_j
e_T
x_1 x_2 … x_t … x_T
encoder
$d_t = f(d_{t-1}, v_{t-1}, c)$
decoder

- Compute the best translation
- Probability of a translation

c is the n-dimentionnal context encoding the input sequence

Training loss : $\frac{1}{N}\sum_{1\le n\le N} log\left(p(V_n|X_n)\right)$ with $P(v_t|v_{t-1}v_{t-2}\dots v_2v_1,c) = g(d_t, v_{t-1}, c)$

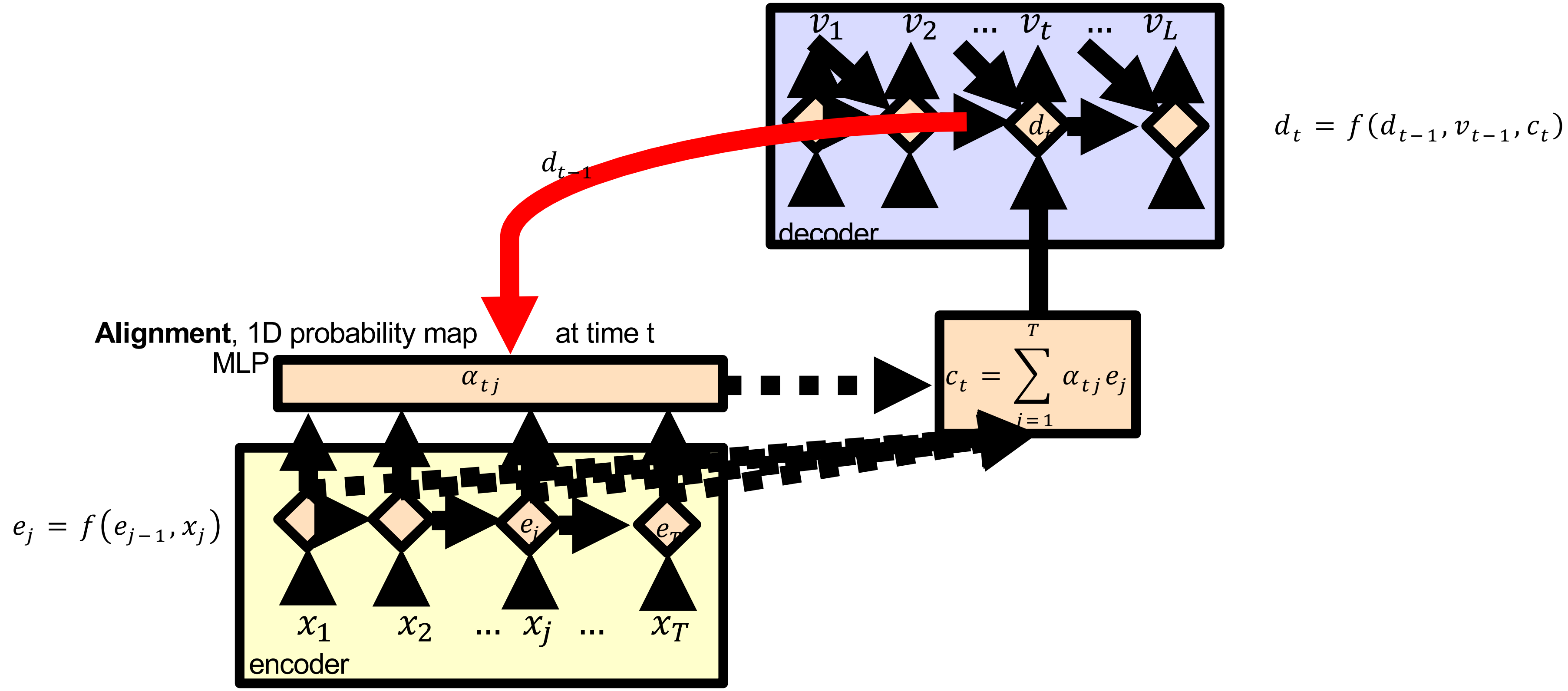
Towards End to End NN architectures

Sequence to sequence with attention mechanism

Introduce a time dependant context c_t during decoding

Select the most appropriate part of the input sentence to produce the curent output

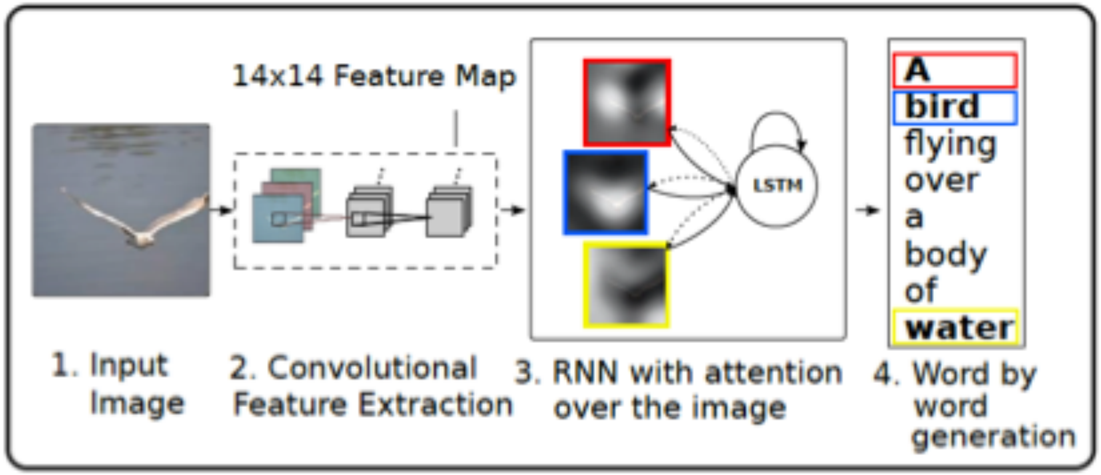
Neural Machine Translation by jointly Learning to Align and Translate, D. Bahdanau et al., ICLR 2015.

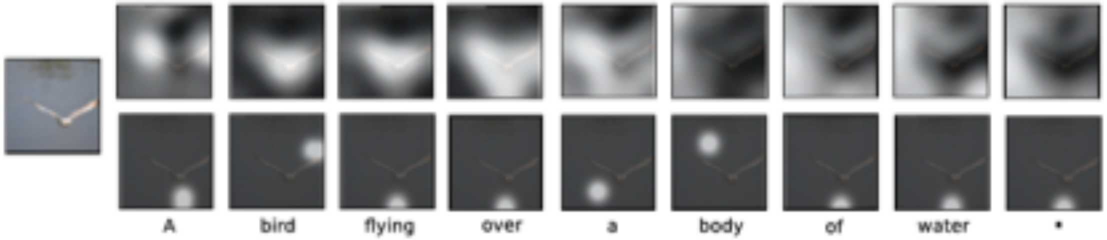


v_1 v_2 ... v_t ... v_L
d_t
decoder
$d_t = f(d_{t-1}, v_{t-1}, c_t)$
d_{t-1}
Alignment, 1D probability map at time t
MLP
α_{tj}
$c_t = \sum_{i=1}^{T} \alpha_{tj} e_j$
$e_j = f(e_{j-1}, x_j)$
e_j
e_T
x_1 x_2 ... x_j ... x_T
encoder

Towards End to End NN architectures

Image to sequence with attention mechanism
For many computer vision tasks the system must recognize subparts of the image by putting attention at some specific locations, **recognize** the object at this location **and predict the next position** of attention


14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

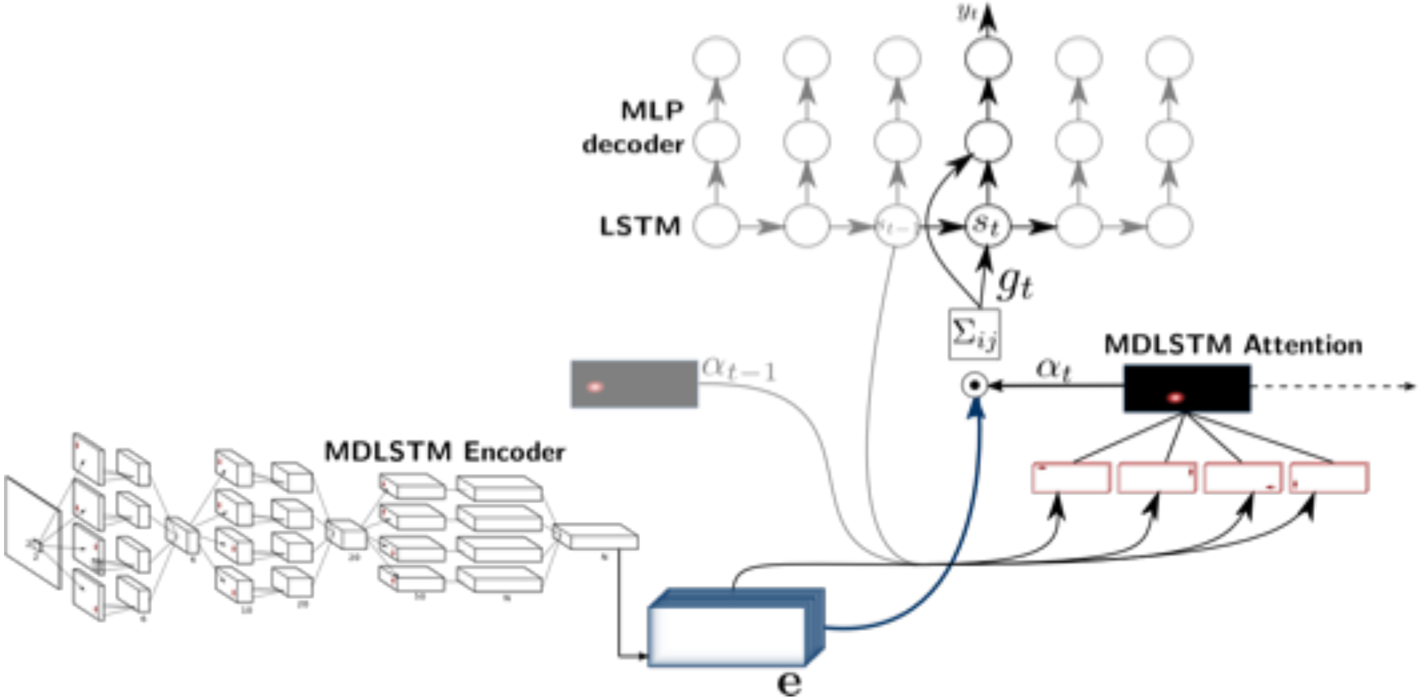


Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, K. Xu et al. , ICML 2015.

Towards End to End NN architectures

Reading text with visual attention

An End to End reading system: extract features with a MDLSTM encoder, update the 2D attention map while decoding characters with a LSTM that embeds a language model



Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attentionT Bluche, J Louradour, R Messina, ICDAR, 2017.

Joint line segmentation and transcription for end-to-end handwritten paragraph recognition T Bluch, NIPS, 2016