

Quelques notes sur l'apprentissage machine

felipe@iro.umontreal.ca

RALI
Dept. Informatique et Recherche Opérationnelle
Université de Montréal



V0.001

Last compiled: 14 octobre 2019



Plan

Portraits de quelques approches

Modèles de Markov cachés

Visible/Caché

Problème 1 : calcul de la probabilité d'une observation

Problème 2 : recherche de la séquence d'états

Problème 3 : apprendre λ

Varia

HMMs en reconnaissance de la parole

Perceptron

Une plateforme amusante

Plan

Portraits de quelques approches

Modèles de Markov cachés

Visible/Caché

Problème 1 : calcul de la probabilité d'une observation

Problème 2 : recherche de la séquence d'états

Problème 3 : apprendre λ

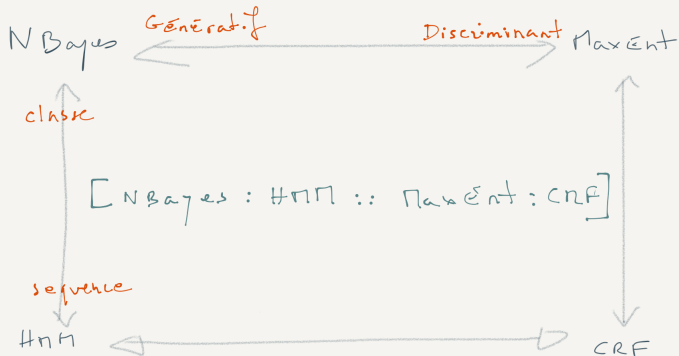
Varia

HMMs en reconnaissance de la parole

Perceptron

Une plateforme amusante

Analogie quand tu nous tiens



Naive Bayes

in : $D = \{(x_1^n, y)\}$ où $x_i \in \mathcal{X}$ et $y \in \mathcal{Y}$



$$\begin{aligned} \hat{y} &= \operatorname{argmax}_{y \in \mathcal{Y}} p(y|x_1^n) \\ &\propto \operatorname{argmax}_{y \in \mathcal{Y}} p(x_1^n, y) \\ &= \operatorname{argmax}_{y \in \mathcal{Y}} p(y) \times p(x_1^n|y) \\ &= \operatorname{argmax}_{y \in \mathcal{Y}} p(y) \times \prod_i p(x_i|y) \end{aligned}$$

- ▶ si l'input est un document d , on peut considérer que x_i est booléen (présence ou non du i e mot du vocabulaire dans d) ou un scalaire (compte i e mot dans d); la jointe peut alors être représentée par une binomiale ou une multinomiale respectivement
- ▶ on peut sélectionner un sous-ensemble de mots, les lemmatiser ou non, etc. un bon baseline
- ▶ Lire [\[McCallum and Nigam, 1998\]](#) pour une application à la classification de documents



MaxEnt

in : $D = \{(x_1^n, y)\}$ où $x_i \in \mathcal{X}$, $y \in \mathcal{Y}$, et m fonctions booléennes $f_j(x, y)$

► soit $n(x, y) = \exp\left(\sum_{j=1}^m \lambda_j f_j(x, y)\right)$,

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_{y \in \mathcal{Y}} p_\lambda(y|x) \\ &= \operatorname{argmax}_{y \in \mathcal{Y}} \frac{n(x, y)}{\sum_{y \in \mathcal{Y}} n(x, y)}\end{aligned}$$

- entraînement (λ_j) par une variante de EM ou descente de gradient (une couche softmax sans couche cachée). Intuitivement, c'est le modèle parmi ceux *compatibles avec le corpus d'entraînement* qui a la plus grande entropie croisée : $H(y|x) = \sum_{(x,y)} p(y, x) \log p(y|x)$
- très souple, les fonctions peuvent se recouvrir en terme d'information. Lire [\[Berger et al., 1996\]](#).
- **note** : on peut simuler un naive bayes avec des fonctions $f_{w,c}(x, y)$ retournant vrai si w apparait au moins c fois dans x

HMM

in : $D = \{(x_1^n, y_1^n)\}; x_i \in \mathcal{X}, y_i \in \mathcal{Y}$

- ▶ **décodage** à l'aide de Viterbi en temps $\mathcal{O}(E^2 \times n)$; E est le nombre d'états et n la longueur d'une séquence

$$\begin{aligned} \hat{y}_1^n &= \operatorname{argmax}_{y_1^n} p(y_1^n | x_1^n) &= \operatorname{argmax}_{y_1^n} p(y_1^n, x_1^n) \\ &= \operatorname{argmax}_{y_1^n} \prod_{i=1}^n p(y_i | y_{i-1}) \times p(x_i | y_i) \end{aligned}$$

- ▶ estimée des paramètres par fréquence relative (supervisé) ou via EM (transition = bigramme, émission = unigramme)
- ▶ cons : ne capture pas les dépendances longues
- ▶ Lire *[Rabiner, 1989]*

CRF (linéaire)

in : $D = \{(x_1^n, y_1^n)\}$; $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, K fonctions valuées $f_k \in \mathbb{R}$

- Inférence via Viterbi :

$$\begin{aligned} \hat{y}_1^n &= \operatorname{argmax}_{y \in \mathcal{Y}^n} p(y|x) \\ &= \operatorname{argmax}_{y \in \mathcal{Y}^n} \frac{1}{Z(x)} \underbrace{\exp \left(\sum_i \sum_{k=1}^K \lambda_k f_k(y_i, y_{i-1}, x, i) \right)}_{\phi(x,y)} \end{aligned}$$

où $Z(x) = \sum_{y'} \phi(x, y')$

- De très bon packages (ex : [Wapiti](#)), approche forte.
- Lire [\[Lafferty, 2001\]](#).

Plan

Portraits de quelques approches

Modèles de Markov cachés

Visible/Caché

Problème 1 : calcul de la probabilité d'une observation

Problème 2 : recherche de la séquence d'états

Problème 3 : apprendre λ

Varia

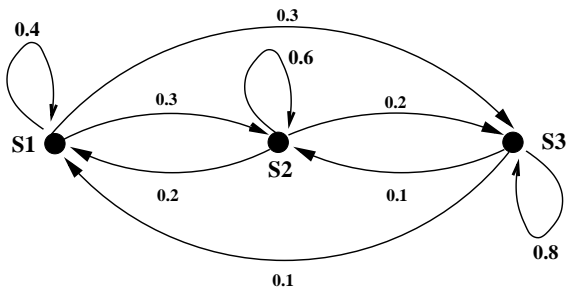
HMMs en reconnaissance de la parole

Perceptron

Une plateforme amusante

Modèle de Markov visible

$S_1 = \text{pluie}; S_2 = \text{nuage}; S_3 = \text{soleil}$



$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \text{ matrice de transitions}$$

Note : $\forall (i, j) \in [1, 3], a_{ij} \geq 0$, et $\forall i \in [1, 3], \sum_{j=1}^3 a_{ij} = 1$

Quel temps fera-t-il les huit prochains jours ?

$$p(S_3 S_3 S_3 S_1 S_1 S_3 S_2 S_3) \stackrel{\text{def}}{=}$$

$$p(S_3) p(S_3 | S_3) p(S_3 | S_3 S_3) p(S_1 | S_3 S_3 S_3) p(S_1 | S_1 S_3 S_3 S_3) \\ p(S_3 | S_1 S_1 S_3 S_3 S_3) p(S_2 | S_3 S_1 S_1 S_3 S_3 S_3) \\ p(S_3 | S_2 S_3 S_1 S_1 S_3 S_3 S_3) \approx$$

$$p(S_3) p(S_3 | S_3) p(S_3 | S_3) p(S_1 | S_3) p(S_1 | S_1) \\ p(S_3 | S_1) p(S_2 | S_3) p(S_3 | S_2) = \\ \pi_3 \times a_{33} \times a_{33} \times a_{31} \times a_{11} \times a_{13} \times a_{32} \times a_{23} = 1.536 \times 10^{-4}$$

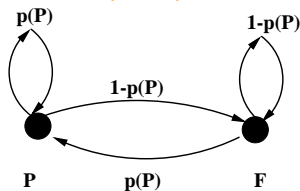
- ▶ hypothèse markovienne d'ordre n :

$$p(S_t | S_{t-1} \dots S_1) = p(S_t | S_{t-1} \dots S_{t-n})$$

- ▶ indépendance au temps : $p(q_t = S_j | q_{t-1} = S_i) = p(S_j | S_i)$

Modèles markoviens et jets de pièces

- ▶ On vous annonce oralement le résultat de tirages (pile ou face) sans vous montrer comment on procède aux tirages.
- ▶ **Première modélisation** : Il existe une pièce (possiblement biaisée). Un état pour pile, un état pour face. Avec ce modèle chaque observation spécifie la séquence d'états
 - ▶ c'est un modèle visible (VMM)

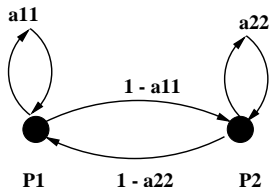


O	=	P	P	F	F	P	F	F	P	...
S	=	1	1	2	2	1	2	2	1	...

paramètre : $p(\text{Pile})$

Modèles markoviens et jets de pièces

- ▶ **Deuxième modélisation** : Il existe deux pièces (possiblement biaisées de manière différente) que l'opérateur change à son gré lors de chaque tirage.
- ▶ Cette fois-ci, l'observation des tirages ne nous spécifie pas dans quel état (pièce) de notre modèle on se trouve.
 - ▶ la séquence d'états S est cachée.



O	=	P	P	F	F	P	F	F	P	...
S	=	1	2	1	2	1	2	2	1	...

4 paramètres : p_1 (Pile), p_2 (Pile), a_{11} , a_{22}

Des urnes et des balles

- ▶ N urnes qui contiennent chacune des balles de couleur. Il y a M couleurs différentes de balles.
- ▶ **Processus génératif** : Un génie tire une balle d'une urne initiale. Il annonce la couleur de la balle et repose la balle dans son urne. Selon un processus aléatoire dépendant de la dernière urne concernée, le génie choisit une nouvelle urne (qui peut-être la même que la précédente) et effectue le tirage d'une balle dans cette urne, etc.
- ▶ Modélisation possible : un HMM à N états (un par urne) ; chaque état **émet** une balle de couleur selon une loi de probabilité spécifique à l'état. Le choix d'une urne par le génie est modélisé par les probabilités de transition d'un état vers un autre.

Des urnes et des balles

- ▶ 3 couleurs : V J R (symboles), 3 urnes (états : s_1, s_2, s_3).
- ▶ Le modèle est spécifié par $\lambda = \{\pi, A, B\}$ où $\pi = \{0.3, 0.3, 0.4\}$, et

$$A = \left[\begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline s_1 & 0.5 & 0.3 & 0.2 \\ s_2 & 0.4 & 0.0 & 0.6 \\ s_3 & 0.0 & 0.3 & 0.7 \end{array} \right] \quad B = \left[\begin{array}{c|ccc} & V & J & R \\ \hline s_1 & 0.0 & 1.0 & 0.0 \\ s_2 & 0.5 & 0.1 & 0.4 \\ s_3 & 0.2 & 0.0 & 0.8 \end{array} \right]$$

- ▶ A et B sont respectivement les matrices de **transition** et **d'émission**.
- ▶ Soit une **séquence d'observations** : $O = \{VJRV\}$
 $p(O|\lambda)$?

Caractérisation d'un HMM

les états ($\{S_1, \dots, S_N\}$) : la séquence d'états est cachée, mais un état correspond souvent à un phénomène précis (ex. urne).

les observations ($\{v_1, \dots, v_M\}$) : l'alphabet avec lequel on décrit les observations (ex : la couleur des balles).

les probabilités de transition ($A = \{a_{i,j}\}$) pour tout $(i, j) \in [1, N]$ avec :
 $a_{ij} = p(q_t = S_j | q_{t-1} = S_i), \quad \forall (i, j) \in [1, N]$

Note : $a_{i,j} \geq 0$ et $\sum_{j=1}^N a_{ij} = 1 \quad \forall (i, j) \in [1, N]$

les probabilités d'émission ($B = \{b_j(k)\}$) pour tout $j \in [1, N]$ et $k \in [1, M]$ avec : $b_j(k) = p(v_k \text{ à l'instant } t | q_t = S_j)$

Note : $b_j(k) \geq 0$ et $\sum_{o=1}^M b_j(o) = 1$

les probabilités initiales ($\pi = \{\pi_i\}$) avec $\pi_i = p(q_1 = S_i), \quad \forall i$

Note : $\pi_i(k) \geq 0$ et $\sum_{i=1}^N \pi_i = 1$

Les trois problèmes fondamentaux des HMMs

Évaluation : Sachant $O = \{O_1 O_2 \dots O_T\}$ et $\lambda = (A, B, \pi)$, comment calculer $p(O|\lambda)$?

Évaluer une observation selon un modèle

Retirer le H de Hidden : Sachant $O = \{O_1 O_2 \dots O_T\}$ et $\lambda = (A, B, \pi)$, comment trouver la séquence (cachée) **optimale** d'états?

Ce qui en pratique nous intéresse le plus souvent

Apprentissage : Sachant un corpus d'entraînement O , comment ajuster les paramètres λ du modèle?

Le problème le plus difficile



Des urnes et des balles : Évaluation

- Il existe 5 chemins (séquences) qui génèrent O :

$$c_1 = \{s_2 s_1 s_3 s_3\} \quad c_3 = \{s_2 s_1 s_2 s_3\} \quad c_5 = \{s_3 s_2 s_3 s_3\}$$

$$c_2 = \{s_2 s_1 s_3 s_2\} \quad c_4 = \{s_3 s_2 s_3 s_2\}$$

$$p(O, c_1 | \lambda) = \pi_2 \cdot b_2(V) \cdot a_{21} \cdot b_1(J) \cdot a_{13} \cdot b_3(R) \cdot a_{33} \cdot b_3(V) = 0.00134$$

$$p(O, c_2 | \lambda) = \pi_2 \cdot b_2(V) \cdot a_{21} \cdot b_1(J) \cdot a_{13} \cdot b_3(R) \cdot a_{32} \cdot b_2(V) = 0.00144$$

$$p(O, c_3 | \lambda) = \pi_2 \cdot b_2(V) \cdot a_{21} \cdot b_1(J) \cdot a_{12} \cdot b_2(R) \cdot a_{23} \cdot b_3(V) = 0.000664$$

$$p(O, c_4 | \lambda) = \pi_3 \cdot b_3(V) \cdot a_{32} \cdot b_2(J) \cdot a_{23} \cdot b_3(R) \cdot a_{32} \cdot b_2(V) = 0.0001728$$

$$p(O, c_5 | \lambda) = \pi_3 \cdot b_3(V) \cdot a_{32} \cdot b_2(J) \cdot a_{23} \cdot b_3(R) \cdot a_{33} \cdot b_3(V) = 0.00016128$$

$$p(O | \lambda) = \sum_{i=1}^5 p(O, c_i | \lambda) \times p(c_i | \lambda) = 0.00377808$$

- La séquence qui explique le mieux l'observation est c_2
- Les calculs peuvent être factorisés



Problème 1 : Évaluation

- ▶ Soit $Q = q_1 q_2 \dots q_T$ une séquence d'états pouvant "expliquer" O .

$$p(O|\lambda) = \sum_{\text{all } Q} p(O, Q|\lambda) = \sum_{\text{all } Q} p(O|Q, \lambda)p(Q|\lambda)$$

- ▶ avec :

$$p(O|Q, \lambda) = \prod_{t=1}^T p(o_t|q_t, \lambda) = b_{q_1}(o_1) \times b_{q_2}(o_2) \dots b_{q_T}(o_T)$$

$$p(Q|\lambda) = \pi_{q_1} \times a_{q_1 q_2} \times \dots \times a_{q_{T-1} q_T}$$

- ▶ donc :

$$p(O|\lambda) = \sum_{q_1 \dots q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \dots a_{q_{T-1} q_T} b_{q_T}(o_T)$$

Complexité : $(2T - 1) \times N^T$ multiplications, $N^T - 1$ additions

Ex : $N = 5$ (états), $T = 100$ (observations)

▷ de l'ordre de $2 \times 100 \times 5^{100} \approx 10^{72}$ opérations !



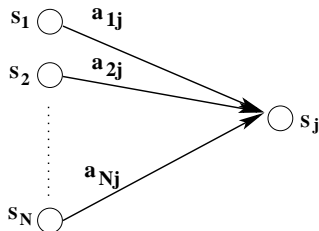
Calcul forward (en avant) :

$$\alpha_t(i) = p(o_1 \dots o_t, q_t = s_i | \lambda)$$

Init : $\alpha_1(i) = \pi_i b_i(o_1), \forall i \in [1, N]$

Induction : $\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(o_{t+1})$, pour tout $t \in [1, T-1]$ et pour tout $j \in [1, N]$

Terminaison : $p(O|\lambda) = \sum_{i=1}^N \alpha_T(i)$



Complexité : $o(N^2 \times T)$ opérations au lieu de $o(T \times N^T)$

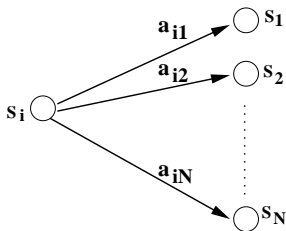
Ex : $N = 5, T = 100 \triangleright$ environ 3000 opérations (vs 10^{72} !!!)

Calcul backward (en arrière) :

$$\beta_t(i) = p(o_{t+1} \dots o_T | q_t = s_i, \lambda)$$

Init : $\beta_T(i) = 1, \forall i \in [1, M]$

Induction : $\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$ pour tout $t \in [1, T-1]$ et pour tout $i \in [1, M]$



Note : même complexité que le calcul forward.

Problème 2 (trouver la séquence)

- ▶ soit $\gamma_t(i) = p(q_t = s_i | O, \lambda)$
- ▶ **Un critère possible** : choisir l'état le plus probable individuellement pour chaque t

$$\hat{q}_t = \operatorname{argmax}_{1 \leq i \leq N} [\gamma_t(i)], \quad \forall t \in [1, T]$$

$$\gamma_t(i) = \frac{p(q_t = s_i, O | \lambda)}{p(O | \lambda)} = \frac{\alpha_t(i) \times \beta_t(i)}{\sum_{i=1}^N \alpha_t(i) \times \beta_t(i)}$$

(On peut calculer les γ une fois les α et β calculés)

- ▶ **Problème** : rien ne garantit que les transitions entre chaque état de \hat{Q} sont valides
- ▶ Critère local.



Algorithme de Viterbi

- ▶ $\hat{Q} = \operatorname{argmax}_Q p(Q|O, \lambda) = \operatorname{argmax}_Q p(O, Q|\lambda)$
car $p(O, Q|\lambda) = p(Q|O, \lambda) \times p(O|\lambda)$
- ▶ Posons :

$$\delta_t(i) = \max_{q_1 \dots q_{t-1}} p(q_1 q_2 \dots q_t = s_i, o_1 \dots o_t | \lambda)$$

- ▶ Par induction :

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \times b_j(o_{t+1})$$

- ▶ On conserve pour chaque t et chaque i l'état ayant amené au maximum $\delta_t(j)$, soit $\phi_t(j)$

Viterbi

init : $\delta_1(i) = \pi_i b_i(o_1)$ et $\phi_1(i) = 0$

réursion :

$$\begin{aligned}\delta_t(j) &= \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(o_t) & 2 \leq t \leq T \\ \phi_t(j) &= \operatorname{argmax}_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] & 1 \leq j \leq N\end{aligned}$$

fin :

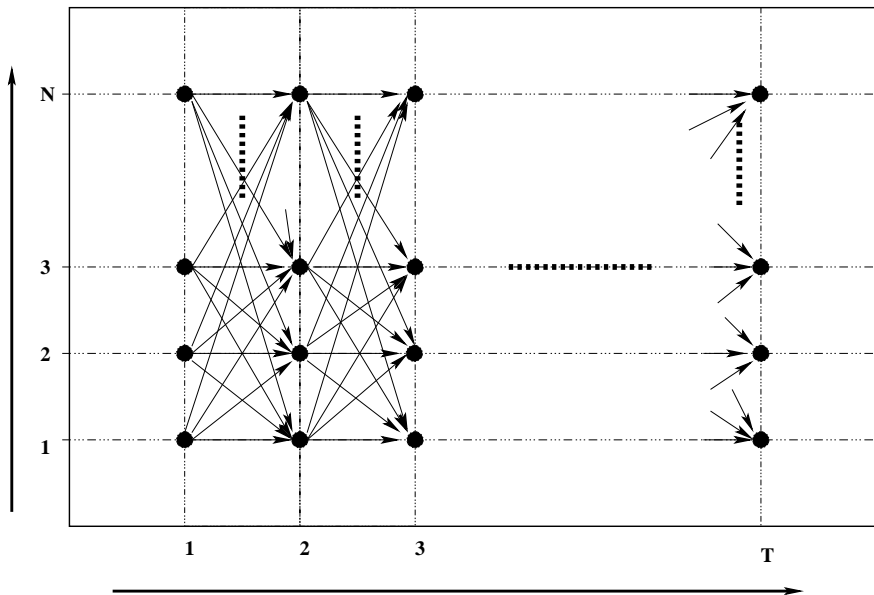
$$\hat{p} = \max_{1 \leq i \leq N} \delta_T(i)$$

$$\hat{q}_T = \operatorname{argmax}_{1 \leq i \leq N} \delta_T(i)$$

meilleure séquence : $\hat{q}_t = \phi_{t+1}(\hat{q}_{t+1})$, $t = T - 1, T - 2, \dots, 1$



Structure en treillis



Algorithme de Viterbi

- ▶ Soit S une table $N \times T$, telle que :
$$S[t, i] = (\delta_t(i), \phi_t(i)) = (S[t, i].p, S[t, i].b)$$
- ▶ Soit A ($N \times N$) la matrice de transition et B ($N \times M$) la matrice d'émission. (matrice de transition souvent creuse)
- ▶ Pour éviter de faire un cas particulier pour les probabilités initiales, on peut étendre A d'une ligne et d'une colonne : $A[0, i] = \pi_i$ et $A[i, 0] = 0 \forall i \in [1, N]$

Algorithme de Viterbi

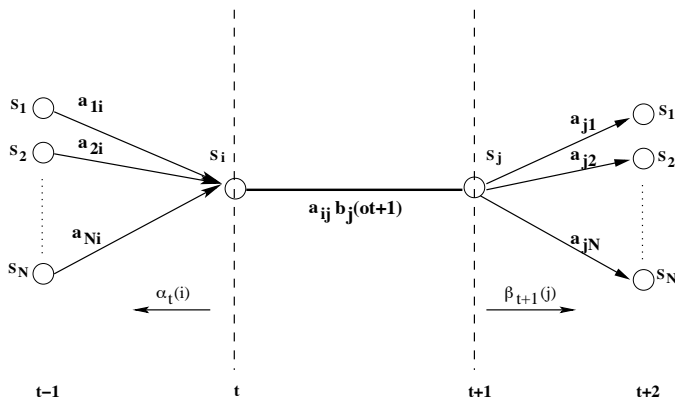
```

// Décodage de  $o_1 o_2 \dots o_T$  (à faire dans le domaine des log!)
 $S[0, 0].p \leftarrow 1$ 
for  $t \leftarrow 1$  à  $T$  do
  for  $i \leftarrow 1$  à  $N$  do
     $S[t, i].p \leftarrow 0$ 
     $e \leftarrow B[i, o_t]$ 
    for  $j \leftarrow 1$  à  $N$  do
      if  $(A[j, i] > 0)$  et  $((m = (S[t - 1, j].p \times A[j, i])) > S[t, i].p)$ 
      then
         $S[t, i] \leftarrow (m, j)$ 
     $S[t, i].p \leftarrow S[t, i].p \times e$ 
// Retour
 $\hat{s} \leftarrow \operatorname{argmax}_{i \in [1, M]: S[T, i].p > 0} S[T, i].p$ 
if  $S[T, \hat{s}].p > 0$  then
  Retourner le chemin à partir de  $S[T, \hat{s}]$ 
else
  Échec de reconnaissance

```

Problème 3 (estimer λ) : Algorithme Baum-Welch

- ▶ à maximum de vraisemblance, par une procédure itérative (EM) qui maximise $p(O|\lambda)$
- ▶ Soit $\xi_t(i, j) = p(q_t = s_i, q_{t+1} = s_j | O, \lambda)$ la probabilité de transiter de i vers j au temps t sachant l'observation O et le modèle.



Algorithme Baum-Welch

$$\begin{aligned}\xi_t(i, j) &= \frac{p(q_t = s_i, q_{t+1} = s_j, O | \lambda)}{p(O | \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}\end{aligned}$$

Note : $\gamma_t(i) = p(q_t = s_i | O, \lambda)$, d'où : $\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j)$

- ▶ $\sum_{t=1}^{T-1} \gamma_t(i)$ = le nombre espéré de transitions depuis s_i , sachant l'observation O et le modèle.
- ▶ $\sum_{t=1}^{T-1} \xi_t(i, j)$ = le nombre espéré de transitions depuis s_i vers s_j , sachant O et le modèle.

Algorithme Baum-Welch

$\overline{\pi}_i$ = nombre espéré de fois où au temps 1 on est en s_i
 (= $\gamma_1(i)$)

$$\overline{a}_{ij} = \frac{\text{nb. espéré de transitions de } s_i \text{ vers } s_j}{\text{nb. espéré de transitions depuis } s_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\begin{aligned} \overline{b}_j(k) &= \frac{\text{nb. espéré de fois où on est en } s_j \text{ et on observe } v_k}{\text{nb. espéré de fois où on est dans } s_j} \\ &= \frac{\sum_{t=1: o_t=v_k}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \end{aligned}$$

- [Baum, 1972] démontre la convergence de cet algorithme

Algorithme Baum-Welch : Une instance de EM

- ▶ Variable cachée : la séquence d'états q
- ▶ La fonction auxiliaire A est :

$$A(\lambda, \lambda') = \sum_{q \in \mathcal{Q}} p(O, q | \lambda') \log p(O, q | \lambda)$$

ou (ce qui revient au même en terme de maximisation) :

$$A'(\lambda, \lambda') = \sum_{q \in \mathcal{Q}} p(q | O, \lambda') \log p(O, q | \lambda)$$

Algorithme Baum-Welch : Une instance de EM

- ▶ Posons $q = q_1 \dots q_T$ et $O = o_1 \dots o_T$.
- ▶ $p(O, q|\lambda) = \pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^T a_{q_{t-1}q_t} b_{q_t}(o_t)$
- ▶ On peut décomposer A en 3 termes indépendants (au regard de la maximisation) :

$$\begin{aligned}
 A(\lambda, \lambda') = & \\
 & \sum_{q \in \mathcal{Q}} (\log \pi_{q_1} \times p(O, q|\lambda') + \log b_{q_1}(o_1) \times p(O, q|\lambda')) + \\
 & \sum_{q \in \mathcal{Q}} \left(\sum_{t=2}^T \log a_{q_{t-1}q_t} \right) p(O, q|\lambda') + \\
 & \sum_{q \in \mathcal{Q}} \left(\sum_{t=2}^T \log b_{q_t}(o_t) \right) p(O, q|\lambda')
 \end{aligned}$$

Algorithme Baum-Welch : Une instance de EM : cas des π_i

- ▶ Condition nécessaire de la maximisation : $\frac{\delta}{\delta \pi_i} A = 0$
- ▶ Maximiser A (selon π_i) est équivalent à :
 - ▶ maximiser seulement $\sum_{q \in \mathcal{Q}} \log \pi_{q_1} p(O, q | \lambda')$
 - ▶ qui revient à maximiser $\sum_{i=1}^N \log \pi_i p(O, q_1 = i | \lambda')$
- ▶ Coefficient de Lagrange (μ) pour la contrainte $\sum_{j=1}^N \pi_j = 1$:

$$\frac{\delta}{\delta \pi_i} \left(\sum_{i=1}^N \log \pi_i p(O, q_1 = i | \lambda') - \mu \left(\sum_{j=1}^N \pi_j - 1 \right) \right) = 0$$

Algorithme Baum-Welch : Une instance de EM : cas des π_i

$$\frac{p(O, q_1 = i | \lambda')}{\pi_i} - \mu = 0 \quad \forall i \in [1, N]$$

Soit :

$$\pi_i = \frac{p(O, q_1 = i | \lambda')}{\mu} \quad \forall i \in [1, N]$$

Or :

$$\sum_{i=1}^N \pi_i = 1 = \sum_{i=1}^N \frac{p(O, q_1 = i | \lambda')}{\mu} \implies \mu = \sum_{i=1}^N p(O, q_1 = i | \lambda')$$

D'où :

$$\pi_i = \frac{p(O, q_1 = i | \lambda')}{\sum_j p(O, q_1 = j | \lambda')} = \gamma_1(i)$$

Limitations de l'approche à maximum de vraisemblance

- ▶ Soit V modèles (λ_v) , $v \in [1, V]$, et une tâche de reconnaissance :

$$\hat{v} = \operatorname{argmax}_v p(O|\lambda_v)$$

- ▶ Lors de l'entraînement ML des modèles, on a :

$$p_v^* = \max_{\lambda_v} p(O^v|\lambda_v)$$

où O^v représente l'ensemble des données étiquetées v dans le corpus d'entraînement (O).

- ▶ un jeu d'observations séparé pour l'entraînement de chaque modèle

Autre critère

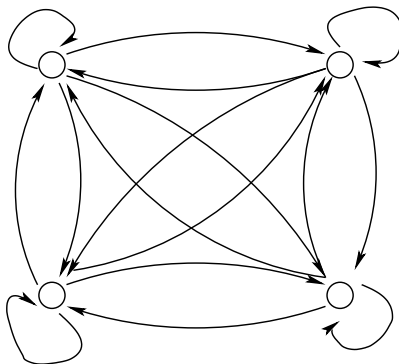
$$I = \max_{\lambda} \left\{ \sum_{v=1}^V \left[\log p(O^v | \lambda_v) - \log \sum_{k \neq v} p(O^v | \lambda_k) \right] \right\}$$

avec $\lambda = \{\lambda_1, \dots, \lambda_V\}$

- Apprentissage plus coûteux

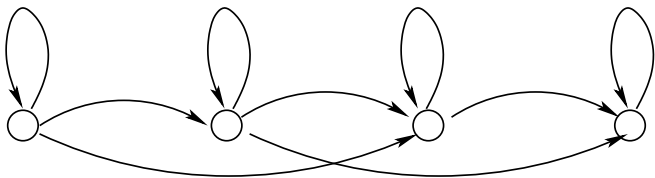
Topologie d'un modèle

- ▶ Un modèle est **ergodique** si tout état est atteignable depuis tout autre état en un nombre fini de transitions.
- ▶ Exemple pour $N = 4$:



Topologie d'un modèle

- ▶ modèle de Bakis :



- ▶ Dans un modèle gauche-droite : $a_{ij} = 0$ si $j < i$

$$\pi_i = \begin{cases} 1 & \text{si } i = 1 \\ 0 & \text{sinon} \end{cases}$$

- ▶ On a souvent des contraintes supplémentaires comme : $a_{ij} = 0$ si $j > i + \Delta$ ($\Delta = 2$ dans les modèles de Bakis)
- ▶ Les modèles gauche-droite permettent de modéliser des signaux qui évoluent avec le temps (ex : parole)

Détail : Scaling des $\alpha_t(i), \beta_t(i)$

$\alpha_t(i) = p(o_1 o_t, q_t = s_i | \lambda)$ nécessite le calcul de :

$$\left(\prod_{s=1}^{t-1} a_{q_s q_{s+1}} \prod_{s=1}^t b_{q_s}(o_s) \right)$$

- ▶ Plus t est grand et plus le résultat de ce calcul tend vers 0. La précision des réels en C (ou équivalent) est insuffisante pour coder ces valeurs, plus t augmente (de l'ordre de 100).

⇒ Il faut normaliser $\alpha_t(i)$.

- ▶ **Idée** : on multiplie $\alpha_t(i)$ par une valeur qui ne dépend que de t et qui assure une bonne dynamique de $\alpha_t(i)$. On applique également le même coefficient à $\beta_t(i)$ (même problème de précision). À la fin, ces coefficients s'annulent.

Observations multiples

- ▶ Souvent, le corpus d'entraînement est présenté comme un ensemble de séquences d'observations (ex : phrases).

- ▶ Soit K le nombre de séquences :

$$O = [O^1, O^2, \dots, O^K] \text{ avec } O^i = [O_1^i O_2^i \dots O_{T_i}^i] \quad \forall i \in [1, K]$$

- ▶ en supposant l'indépendance de chaque phrase :

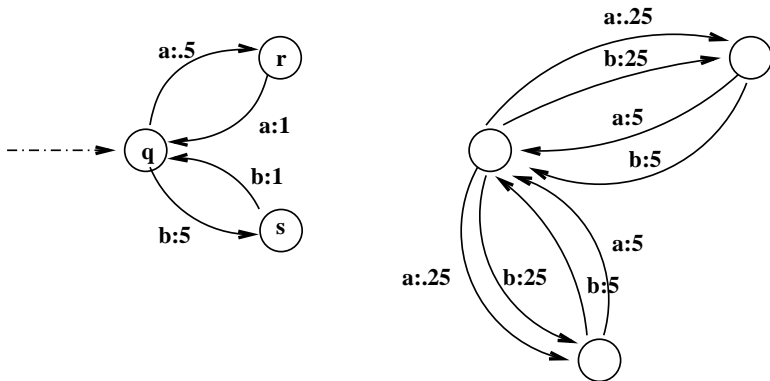
$$p(O|\lambda') = \prod_{k=1}^K p(O^k|\lambda') = \prod_{k=1}^K P_k$$

- ▶ Alors les formules de réestimations sont :

$$\overline{a_{ij}} = \frac{\sum_{k=1}^K \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_j(o_{t+1}^k) \beta_{t+1}^k(j)}{\sum_{k=1}^K \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(j)}$$

$$\overline{b_j(l)} = \frac{\sum_{k=1}^K \frac{1}{P_k} \sum_{t=1: o_t=l}^{T_k-1} \alpha_t^k(i) \beta_t^k(j)}{\sum_{k=1}^K \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(j)}$$

Les points critiques



- ▶ Apprentissage sur le corpus $\{aabb\}$:
 - ▶ avec le second modèle, les estimées ne changeront pas
- ▶ ajouter du bruit lors de l'initialisation avant apprentissage

Problèmes d'EM

- ▶ Si une transition (par exemple) n'est pas utile pour la génération de O , alors cette transition va voir sa probabilité décroître à chaque itération.

▷ sur-entraînement

- ▶ EM converge vers un maximum local de la vraisemblance.

▷ importance des choix initiaux



HMMs en reconnaissance de la parole (RAP)

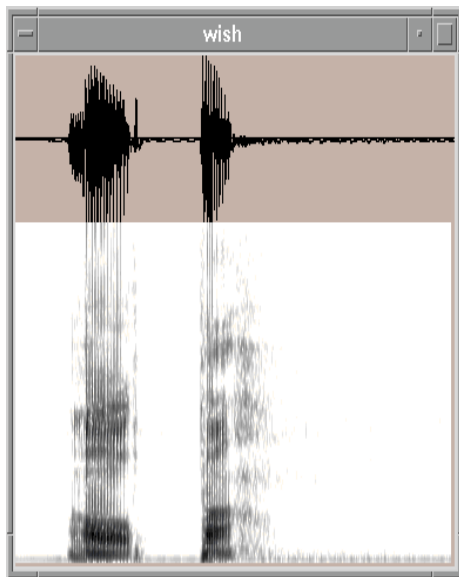
À l'entrée du canal : une séquence de mots w

À la sortie du canal : un signal de parole O

$$\begin{aligned}
 \hat{w} &= \operatorname{argmax}_{w \in \mathcal{F}} p(w|O) \\
 &= \operatorname{argmax}_{w \in \mathcal{F}} \frac{p(O|w) \times p(w)}{p(O)} \\
 &= \operatorname{argmax}_{w \in \mathcal{F}} \underbrace{p(O|w)}_{\text{acoustico-phonétique}} \times \underbrace{p(w)}_{\text{modèle de langue}}
 \end{aligned}$$

- Pour un bon panorama des techniques markoviennes utilisées dans les systèmes de RAP, lire [\[Huang et al., 1990\]](#)

Représentation du signal

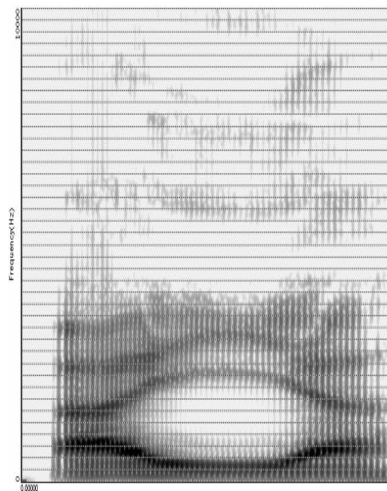


Typiquement :

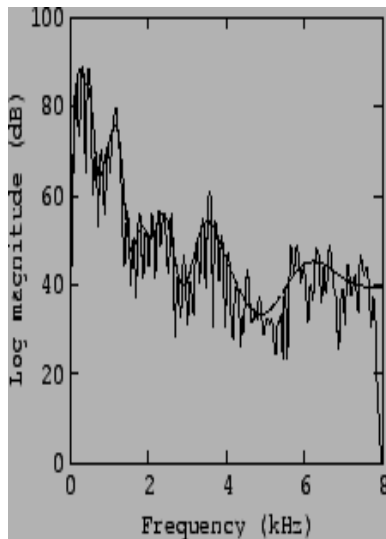
Signal 16 000 échantillons
par seconde
(1 échantillon =
16 bits)

Spectro 39 coefficients par
trame
(1 trame = 10 ms)
⇒ 3 900 valeurs
par seconde.

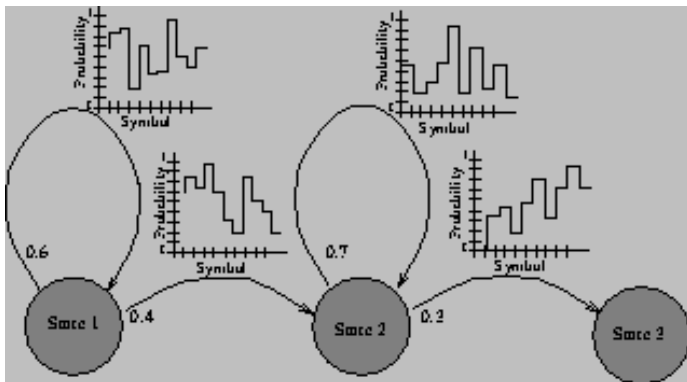
Représentation du signal



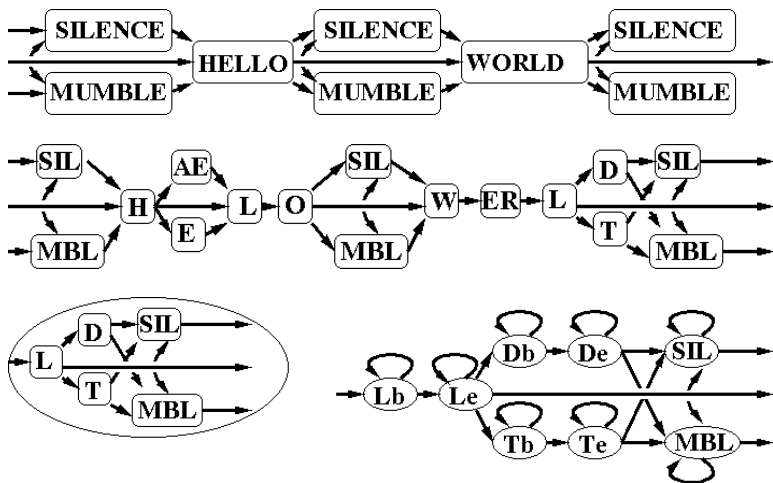
[a j a]



Exemple de modèle acoustique



Exemple de graphe de reconnaissance



Plan

Portraits de quelques approches

Modèles de Markov cachés

Visible/Caché

Problème 1 : calcul de la probabilité d'une observation

Problème 2 : recherche de la séquence d'états

Problème 3 : apprendre λ

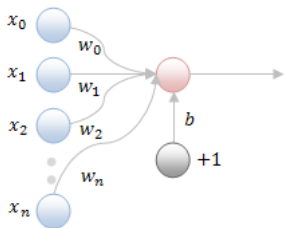
Varia

HMMs en reconnaissance de la parole

Perceptron

Une plateforme amusante

Perceptron (Rosenblatt, 1957)



- ▶ x_j input (valeur réelle)
- ▶ w_j poids synaptique (valeur réelle)
- ▶ b biais (peut être représenté par un poids supplémentaire et une entrée fixe)
- ▶ $h = \sum_j x_j \times w_j + b$ est le signal arrivant au neurone
- ▶ l'activation du neurone est déterminée par une fonction d'activation $\hat{y} \equiv f(h)$ (ex : $f = \text{sign}$)

Perceptron (Rosenblatt, 1957)

- ▶ soit $(x_i, y_i)_{i \in [1, L]}$ un corpus d'entraînement où y_i (ici à valeur dans $\{-1, +1\}$) est la réponse (supervision) et $x_i \in \mathbb{R}^{n+1}$
- ▶ apprendre les w_j peut se faire **en ligne** en ajustant itérativement les poids une fois chaque observation x_i rencontrée :
$$\forall j \in [0, n], w_j \leftarrow w_j + \eta(y_i - \hat{y}_i) \cdot x_i$$

où η est le **learning rate**

résout des problèmes linéairement séparables



Voted-Perceptron [*Freund and Schapire, 1999*]

- ▶ Entraînement en ligne :

Require: $\{(\mathbf{x}_i, y_i)\}_{i \in [1, L]}$ où $y_i \in \{+1, -1\}$ et $x_i \in \mathbb{R}^n$

Ensure: a pool of K perceptrons $\{(\mathbf{v}_k, c_k)\}_{k \in [1, K]}$, $\mathbf{v}_k \in \mathbb{R}^n$, $c_k \in \mathbb{N}$

$k, c_1 \leftarrow 0$

$\mathbf{v}_1 \leftarrow [0 \dots 0]^T$

for all epoch **do**

for all $i \in [1, L]$ **do**

$\hat{y} \leftarrow \text{sign}(\mathbf{v}_k \cdot \mathbf{x}_i)$

if $\hat{y} \neq y_i$ **then**

$\mathbf{v}_{k+1} \leftarrow \mathbf{v}_k + y_i \mathbf{x}_i$

$c_{k+1} \leftarrow 1$

$k \leftarrow k + 1$

else

$c_k \leftarrow c_k + 1$

Garanties de convergence (même dans les cas non linéairement séparables)

Voted-perceptron [*Freund and Schapire, 1999*]

- ▶ Test :

$$\hat{y} = \text{sign} \left(\sum_k c_k \cdot \text{sign}(\mathbf{v}_k \cdot \mathbf{x}) \right)$$

- ▶ requiert la sauvegarde de K perceptrons
- ▶ calcul en $O(K \times n)$. On peut également ne retenir qu'un sous-ensemble des perceptrons (à l'extrême un seul, par exemple celui de plus fort c_k).

Perceptron structuré [Collins, 2002]

```

w0, wa ← 0
repeat
  for all example (x, y) ∈ D do
     $\hat{y} = \operatorname{argmax}_y \mathbf{w}_0^T \Phi(x, y)$ 
    if  $\hat{y} \neq y$  then
       $\mathbf{w}_0 \leftarrow \mathbf{w}_0 + \Phi(x, y) - \Phi(x, \hat{y})$ 
       $\mathbf{w}_a \leftarrow \mathbf{w}_a + c\Phi(x, y) - c\Phi(x, \hat{y})$ 
       $c \leftarrow c + 1$ 
until converged
return  $\mathbf{w}_0 - \mathbf{w}_a/c$ 

```

Plan

Portraits de quelques approches

Modèles de Markov cachés

Visible/Caché

Problème 1 : calcul de la probabilité d'une observation

Problème 2 : recherche de la séquence d'états

Problème 3 : apprendre λ

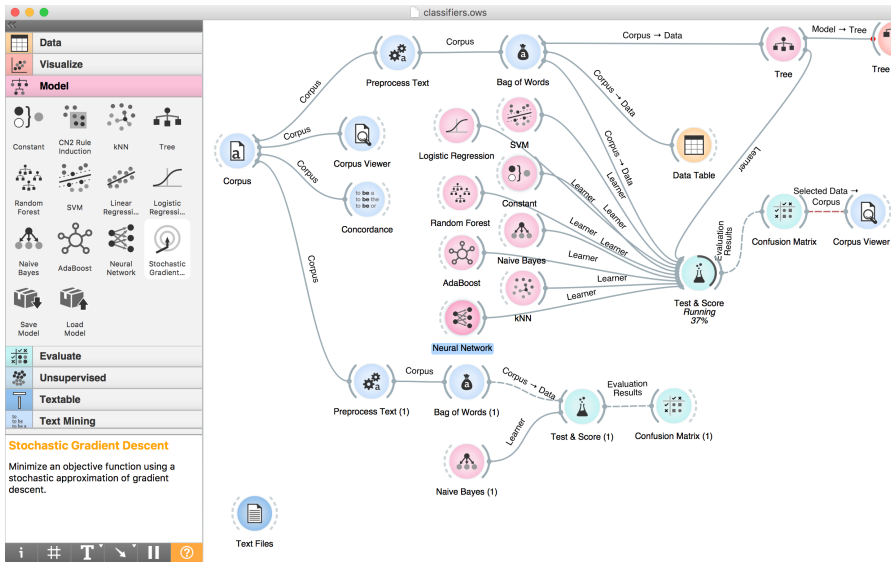
Varia

HMMs en reconnaissance de la parole

Perceptron

Une plateforme amusante

Orange : une expérience



Orange : visualisation

The screenshot shows the Orange3 software interface. The main window is titled "Classifiers.ows" and displays a "Corpus Viewer" window. The Corpus Viewer window is divided into several sections:

- Info:** Documents: 140; Preprocessed: False; Tokens: n/a; Types: n/a; POS tagged: False; N-grams range: 1-1; Matching: 140/140.
- Search features:** Category, Text.
- Display features:** Category, Text.
- Actions:** Show Tokens & Tags (unchecked), Auto send is on (checked).
- RegEx Filter:** (empty)
- Document List:** A table with 17 rows, numbered 84 to 100, each containing a document ID and the text "Document 84".
- Text Content:** A large text area displaying the content of a document, starting with "Category: adult" and "Text: was beginning to increase that vague feeling of uneasiness which I always have when the Count is near But at the instant I saw that the cut had bled a little and the blood was trickling over my chin I laid down the razor turning as I did so half round to look for some sticking plaster When the Count saw my face his eyes blazed with a sort of demoniac fury and he suddenly made a grab at my throat I drew away and his hand touched the string of beads which held the crucifix It made an instant change in him for the fury passed so quickly that I could hardly believe that it was ever there Take care he said take care how you cut yourself It is more dangerous than you think in this country Then seizing the shaving glass he went on And this is the wretched thing that has done the mischief It is a foul bauble of man's vanity Away with it! And opening the window with one wrench of his terrible hand he flung out the glass which was shattered into a thousand pieces on the stones of the courtyard far below Then he withdrew without a word It is very annoying for I do not see how I am to shave unless in my watch-case or the bottom of the shaving pot which is fortunately of metal When I went into the dining room breakfast was prepared but I could not find the Count anywhere So I breakfasted alone It is strange that as yet I have not seen the Count eat or drink He must be a very peculiar man! After breakfast I did a little exploring in the castle I went out on the stairs and found a room looking towards the South The view was magnificent and from where I stood there was every opportunity of seeing it The castle is on the very edge of a terrific precipice A stone falling from the window would fall a thousand feet without touching anything! As far as the eye can reach is a sea of green tree tops with occasionally a deep rift where there is a chasm Here and there are silver threads where the rivers wind in deep gorges through the forests But I am not in heart to describe beauty for when I had seen the view I explored further Doors doors everywhere and all locked and bolted In

The interface also shows a sidebar with various machine learning models like Constant, CN2 Rule Induction, kNN, Random Forest, SVM, Linear Regression, Naive Bayes, AdaBoost, and Neural Network. There are also buttons for "Save Model" and "Load Model".

Stochastic Gradient Descent

Minimize an objective function using a stochastic approximation of gradient descent.



Text Files

Orange : concordancier

Info

Tokens: 133484
Types: 11744
Matching: 43/140

Number of words: 10

Query: dear

1	... eyes were closed and his face a horrible colour	Dear	deary me cried my mother what a
2	...that one of the newcomers carried a lantern My	dear	said my mother suddenly take the
3	news : Old Anchor Inn Bristol March 1 17 --	Dear	Livesey -- As I do not know wheth
4	where I had lived since I was born and the	dear	old Admiral Benbow -- since he w
5	...ow -- since he was repainted no longer quite so	dear	One of my last thoughts was of the
6	...g out shod in silver shoes with pointed toes Oh	dear	! Oh dear ! cried Dorothy clasping
7	in silver shoes with pointed toes Oh dear ! Oh	dear	! cried Dorothy clasping her hands
8	surrounds this Land of Oz ! ' m afraid my	dear	you will have to live with us Doroth
9	the words on it asked is your name Dorothy my	dear	? Yes answered the child looking u
10	and ask him to help you Good - bye my	dear	The three Munchkins bowed low to
11	no idea of looking at anything and still slept on	Dear	me ! exclaimed Polly in consternat
12	...d Polly pointing tragically to the little heap Well	dear	me ! said Jasper Why Polly -- as h
13	... looking up into his face Indeed I am Grandpapa	dear	very hungry Oh to think of it ! Yes
14	the bedclothes I wish you ' d take me Grandpapa	dear	she said holding up her arms So I
15	...r he walked up and down the room There there	dear	! Oh why doesn ' t that Sarah hurry
16	t mean to be didn ' t mean to Oh	dear	me ! exclaimed old Mr King in dism

Stochastic Gradient Descent

Minimize an objective function using a stochastic approximation of gradient descent.

Text Files

felipe@iro.umontreal.ca

Quelques notes sur l'apprentissage machine

Orange : neural network

The screenshot shows the Orange3 software interface with a workflow for text classification. The workflow includes the following components and connections:

- Data Source:** A 'Text Files' widget (document icon) provides 'Corpus' data to 'Preprocess Text' and 'Bag of Words' widgets.
- Preprocessing:** 'Preprocess Text' and 'Bag of Words' both output 'Corpus' data to a 'Data Table' widget.
- Learning:** The 'Data Table' is used by multiple 'Learner' widgets: 'Naive Bayes', 'Neural Network', 'SVM', 'Linear Regression', 'Logistic Regression', 'Tree', and 'Random Forest'.
- Evaluation:** The 'Learner' outputs are fed into 'Test & Score Running 51%' and 'Confusion Matrix' widgets.

The 'Neural Network' dialog box is open, showing the following configuration:

- Name: Neural Network
- Network: (empty)
- Neurons per hidden layer: 100
- Activation: ReLu
- Solver: Adam
- Alpha: 0,00010
- Max iterations: 200
- Apply Automatically
- Buttons: Cancel

On the left sidebar, the 'Model' section is expanded to show various machine learning models. The 'Evaluate' section is also visible, with 'Unsupervised', 'Textable', and 'Text Mining' sub-sections.

Stochastic Gradient Descent
 Minimize an objective function using a stochastic approximation of gradient descent.

Orange : résultats

The screenshot displays the Orange3 interface. On the left, a workflow is visible with the following components: Preprocess Text (1), Corpus, Bag of Words (1), Naive Bayes (1), Corpus, Data, Test & Score (1), Evaluation Results, and Confusion Matrix (1). The 'Test & Score' widget is open, showing the following settings:

- Sampling:**
 - Cross validation
 - Number of folds: 10
 - Stratified
 - Cross validation by feature
 - Random sampling
 - Repeat train/test: 10
 - Training set size: 66%
 - Stratified
 - Leave one out
 - Test on train data
 - Test on test data
- Target Class:** (Average over classes)

The **Evaluation Results** table is as follows:

Method	AUC	CA	F1	Precision	Recall
kNN	0.955	0.921	0.921	0.922	0.921
Tree	0.779	0.779	0.778	0.779	0.779
SVM	0.996	0.957	0.957	0.957	0.957
Random Forest	0.902	0.843	0.843	0.843	0.843
Neural Network	0.998	0.979	0.979	0.979	0.979
Naive Bayes	0.879	0.843	0.839	0.880	0.843
Logistic Regression	0.994	0.950	0.950	0.950	0.950
Constant	0.500	0.500	0.495	0.500	0.500
AdaBoost	0.793	0.793	0.793	0.794	0.793

Below the table, a 'Text Files' icon is visible.

Orange : matrice de confusion

The screenshot shows the Orange3 software interface. On the left is a widget palette with categories: Data, Visualize, Model, Evaluate, Unsupervised, Textable, and Text Mining. The main workspace contains a workflow with the following widgets: Preprocess Text (1), Corpus, Bag of Words (1), Naive Bayes (1), Neural Network, kNN, Corpus → Data, Test & Score (1), Evaluation Results, and Confusion Matrix (1). A 'Confusion Matrix' window is open in the foreground, displaying the following data:

Classifiers: SVM, Constant, Naive Bayes, kNN, Logistic Regression, Random Forest, AdaBoost, Neural Network, Tree.

Output: Predictions, Probabilities, Send Automatically.

Show: Number of instances

		Predicted		Σ
		adult	children	
Actual	adult	59	11	70
	children	11	59	70
Σ		70	70	140

Buttons: Select Correct, Select Misclassified, Clear Selection.

Below the workflow, there is a 'Text Files' widget icon and a description for the 'Confusion Matrix' widget:

Confusion Matrix

Display a confusion matrix constructed from the results of classifier evaluations.

[more...](#)



Baum, L. (1972).

An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process.

Inequalities, 3 :1–8.



Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. (1996).

A maximum entropy approach to natural language processing.

Comput. Linguist., 22(1) :39–71.



Collins, M. (2002).

Discriminative training methods for hidden markov models : Theory and experiments with perceptron algorithms.

In Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10, EMNLP '02, pages 1–8.



Freund, Y. and Schapire, R. E. (1999).

Large margin classification using the perceptron algorithm.

Mach. Learn., 37(3) :277–296.



Huang, X., Ariki, Y., and Jack, M. (1990).

Hidden Markov Models for Speech Recognition.

Edinburgh University Press.

ISBN-0-7486-0162-7.



Lafferty, J. (2001).

Conditional random fields : Probabilistic models for segmenting and labeling sequence data.

In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann.



McCallum, A. and Nigam, K. (1998).

A comparison of event models for naive Bayes text classification.

In *Learning for Text Categorization : Papers from the 1998 AAAI Workshop*, pages 41–48.



Rabiner, L. R. (1989).

A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, chapter 6.

IEEE.