7 Apprentissage

Il y a trois grands avantages de modèles probabilistes :

- ⋆ **[apprentissage]** on peut ajuster les paramètres du modèle selon des principes mathématiques bien fondés (p.e., maximisation de vraisemblance)
- ⋆ **[confidence]** l'inférence est accompagnée de mesures statistiques de confidence (p.e., probabilité postérieure de colonnes d'un alignement)
- ⋆ **[choix]** on peut choisir l'architecture du modèle selon des mesures de *fit* statistique (p.e., pénalisation par déscription minimale)

7.1 Vraisemblance

Soit $X = (x_1, x_2, \ldots, x_n)$ un jeu de données observées, ou un *échantillon*. Un modèle probabiliste M se décrit par son architecture (obsérvation + variables aléatoires inobservées avec dépendances) et par la vecteur de ses paramètres θ. Le modèle définit la distribution d'obsérvations. Typiquement, chaque x_i est tirée indépendamment de la même distribution $p(x|\theta)$. Ainsi, on définit la vraisemblance

$$L(\theta) = \mathbb{P}\Big\{\text{on obsèrve } X \;\Big|\; \text{modèle } M \text{ avec paramètres } \theta\Big\} = \prod_{i=1}^{n} p(x_i|\theta). \tag{7.1}$$

7.2 Vraisemblance et modèles graphiques

Le modèle probabiliste comprend un ensemble de variables aléatoires inobservées $Y_1, \ldots, Y_m$ et l'obsérvation Y_0 même :

$$p(x|\theta) = \sum_{y_1, y_2, \ldots, y_m} \mathbb{P}\Big\{Y_0 = x \;\Big|\; Y_1 = y_1, \ldots, Y_m = y_m; \theta\Big\} \cdot \mathbb{P}\Big\{Y_1 = y_1, \ldots, Y_m = y_m \;\Big|\; \theta\Big\}.$$

La somme suggère une temps exponentiel en m, mais on peut souvent exploiter les indépendances conditionnelles entre les Y_i pour accélerer le calcul. Considérons le graphe G de dépendances entres les variables aléatoires. Les nœuds de G correspondent aux variables $Y_i \colon i = 0, 1, \ldots, m$. Les arcs de G sont orientés. L'ensemble des arcs entrants au nœud j, dénoté par $\mathsf{pred}(j) = \big\{i \colon ij \in G\big\}$, définit l'indépendance conditionnelle :

$$\mathbb{P}\Big\{Y_j \;\Big|\; Y_i \colon i \neq j\Big\} = \mathbb{P}\Big\{Y_j \;\Big|\; Y_i \colon \in \mathsf{pred}(j)\Big\}.$$

Un *modèle graphique* est un modèle probabiliste dont le graphe de dépendances ne contient aucun cycle. Exemple : évolution d'un caractère dans un arbre phylogénétique. On associe une valeur inobservée avec chaque nœud ancestral. On obsèrve le caractère aux feuilles. L'arbre montre les dépendances : $\mathsf{pred}(j)$ contient le parent du nœud j. La vraisemblance se calcule rapidement dans l'ordre inverse du tri topologique.

[algorithme de Felsenstein pour calculer vraisemblance sur un arbre phylogénétique]

7.3 Maximisation de la vraisemblance

Une méthode standarde d'ajuster les paramètres du modèle est de maximiser la vraisemblance. La valeur des paramètres θ_{ML} qui maximise $L(\theta)$ est un estimateur *consistent* sous quelques conditions génériques[1] sur

[1] Wald A. "Note on the consistency of the maximum likelihood estimate", *Annals of Mathematical Statistics*, **20** :595–601, 1949

la topologie de l'espace de paramètres : sur un échantillon généré par le même modèle, quand $n \to \infty$, θ_{ML} converge vers les vraies valeurs. Une des conditions très importantes est l'*identifiabilité* du modèle : $p(x|\theta)$ doit être une distribution différente pour toute valeur de θ.

Optimisation numérique. Le plus simple est de considérer la vraisemblance de l'équation (7.1) comme une fonction de la vecteur de paramètres. On maximise alors sa valeur par une méthode numérique standarde comme la méthode de la bisection ou la méthode du gradient conjugué. Dans l'implantation du calcul, il faut payer attention aux issues de débordement numérique. Typiquement, on travaille avec la log-vraisemblance $\log L(\theta) = \sum_{i=1}^{n} \log p(x_i|\theta)$.

7.4 Espérance-maximisation

W(en)

L'algorithme d'espérance-maximisation (EM) sert à optimiser la vraisemblance d'un modèle probabiliste. L'algorithme calcule une séquence d'estimateurs $\theta_0, \theta_1, \theta_2, \ldots$ en une itération jusqu'à convergence. En chaque itération, on exécute un étape E («espérance») et un étape M («maximisation»). Pour simplifier la discussion, soit X l'observation aléatoire, et Y la vecteur de variables aléatoires inobservées. On obsèrve $X = x$. En itération $i = 1, 2, \ldots$, on fait :

⋆ **Étape E.** On définit la fonction

$$Q_i(\theta) = \mathbb{E}_{Y|X,\theta_{i-1}} \log \mathbb{P}\Big\{Y; X = x \;\Big|\; \theta\Big\} = \sum_y \mathbb{P}\Big\{Y = y \;\Big|\; X = x; \theta_{i-1}\Big\} \log \mathbb{P}\Big\{Y; X = x \;\Big|\; \theta\Big\}. \quad (7.2)$$

⋆ **Étape M.** On maximise Q :

$$\theta_i \leftarrow \arg\max_\theta Q_i(\theta). \quad (7.3)$$

L'algorithme EM converge vers le maximum de la vraisemblance.

L'algorithme de Baum-Welch pour les HMM suit cette technique générique. Les variables inobservées sont les états. Supposons qu'on obsèrve la séquence $\mathbf{x} = x_1 x_2 \cdots x_n$, génerée par la séquence d'états inconnue $Q_1, \ldots, Q_n$. Soit $\gamma_i(q) = \mathbb{P}\Big\{Q_i = q \;\Big|\; x\Big\}$, la probabilité postérieure de l'état Q_i, ce qu'on calcule par l'algorithme forward-backward. Soit $e(x|q) = \mathbb{P}\Big\{X_i = x \;\Big|\; Q_i = q\Big\}$ la probabilité d'émission du caractère x en état q. En équation (7.2), $e(x|q)$ paraît dans les termes :

$$Q(e) = \sum_{i=1}^{n} \gamma_i(q) \log e(x_i|q).$$

Pour un alphabet à $r < \infty$ lettres, on a $Q(e_1, \ldots e_r) = \sum_{a=1}^{r} n_a \log e_a$, où $n_a = \sum_{i:\, x_i = a} \gamma_i(q)$ est le nombre d'émissions a en état q, et $e_a = e(a|q)$ et la probabilité d'émission du caractère a. Or, cette expression est maximisée avec le choix $e_a = n_a / \sum_{j=1}^{r} n_j$ dans l'étape M, et c'est exactement la formule dans l'aprentissage Baum-Welch.

7.5 Algorithme des k-moyennes

W(en)

L'algorithme des k-moyennes sert à partitionner un jeu de donnés $x_1, x_2, \ldots, x_n$ de d dimensions ($x_i \in \mathbb{R}^d$). Le modèle probabiliste sous-jacente comprend k *centres* $\mu_1, \mu_2, \ldots, \mu_k$ inobservés. Pour générer un

seul point X observé, on choisit un des centres au hasard ($Y = 1, 2, \ldots, k$), et on tire une valeur selon la distribution normale centrée à μ_Y. En étape E, on doit calculer

$$\rho_i(j) = \mathbb{P}\Big\{Y_i = j \;\Big|\; X_i = x_i; \mu_1, \ldots, \mu_k\Big\} \propto \exp\Big(-(x_i - \mu_j)^2/2\Big) :$$

c'est l'affectation «souple» de point i au centre j. En étape M, on trouve le choix de centres maximisant la fonction Q en (7.3) par $\mu_j \leftarrow \frac{\sum_i \rho_i(j) x_i}{\sum_i \rho_i(j)}$: c'est la vote des points ponderée selon l'affectation souple.

En une démarche altérnative de groupage «dur» (évoquant la stratégie d'apprentissage Viterbi), on utilise $\rho_i(j) = 1$ pour le centre le plus proche ($(x_i - \mu_j)^2$ minimale) et $\rho_i(j) = 0$ pour tout autre centre. L'affectation de l'étape M est simplement $\mu_j \leftarrow \frac{\sum_i \{\rho_i(j)=1\} \cdot x_i}{\sum_i \{\rho_i(j)\}}$.

7.6 Bootstrapping

La maximisation de la vraisemblance nous fournit un seul choix optimal pour les paramètres. Parfois il est désirable d'étudier l'incertitude de valeurs de paramètres. Une approche générale consiste de simulations : on examine comment l'optimisation fonctionne sur un échantillon aléatoire. Le **bootstrapping** est applicable quand l'observation est iid. Étant donné la vraie observation $x_1, x_2, \ldots, x_n$, on produit un échantillon simulé $y_1, \ldots, y_n$ de même taille. Pour chaque y_i on choisit un des x_j au hasard (avec répétitions permises). Ensuite, on performe la procédure d'optimisation de paramètres sur l'échantillon simulé. On répète la procédure plusieurs fois ($N = 1000, 10000, 100000, \ldots$) et on collecte des statistiques sur la distribution de valeurs optmisées dans les échantillons. On construit une intervalle de confiance en excluant les valeurs les plus grandes et les plus petites (p.e., pour 95%, on exclut 25-25 parmi 1000).

W(fr)

En «boostrap paramétrique», l'échantillon simulé est construit à partir du modèle directement. Notamment, on génère n obsérvations iid, en utilisant la distribution impliée par le modèle. L'échantillon ainsi produit est soumis à la procédure normale d'apprentissage. On collecte des statistiques en répétant la procédure.

7.7 Markov Chain Monte Carlo

Idéalement, on voudrait avoir de l'information complète sur la distribution postérieure des paramètres : on veut calculer $\mathbb{P}\Big\{\theta \;\Big|\; \mathbf{x}\Big\}$. Une telle formulation de la question est possible si on introduit la distribution *a priori* des paramètres $p(\theta)$. Par la loi de Bayes,

$$\mathbb{P}\Big\{\theta \;\Big|\; \mathbf{x}\Big\} = \frac{\mathbb{P}\Big\{\mathbf{x} \;\Big|\; \theta\Big\} \cdot p(\theta)}{\mathbb{P}\{\mathbf{x}\}} \propto L(\theta) \cdot p(\theta). \tag{7.4}$$

Notez qu'en général, on ne peut pas directement calculer le dénominateur $\mathbb{P}\{\mathbf{x}\}$ car il requiert l'intégrale de la vraisemblance pour tout choix de θ. L'approche de méthodes *Markov chain Monte Carlo* (MCMC) est de définir une chaîne de Markov dont les états sont les valeurs possibles de θ. L'espace des états est l'espace de paramètres. On définit les transitions entre états de telle façon que la chaîne converge vers la distribution postérieure de θ. Dans d'autres mots, on produit un *échantillonnage* de l'espace de paramètres. Il s'agit d'une collection $\theta_0, \theta_1, \theta_2, \ldots, \theta_N$ (états simulés de la chaîne) telle que

$$\lim_{N\to\infty} \mathbb{P}\{\theta_i = \theta\} = \mathbb{P}\Big\{\theta \;\Big|\; \mathbf{x}\Big\}.$$

Pour achever la convergence vers la distribution souhaitée, on choisit des probabilités de transition pour lesquelles la distribution stationnaire est exactement $\mathbb{P}\Big\{\theta \;\Big|\; \mathbf{x}\Big\}$. Rappel : la distribution stationnaire π (vue

comme vecteur ligne) d'une chaîne de Markov avec matrice de transition $\mathbf{M}$ est sa vecteur propre avec $\pi \cdot \mathbf{M} = \pi$. Si la chaîne est apériodique et irréductible (on peut accéder à tout état à partir de n'importe quel état), alors elle converge vers la distribution stationnaire. Étant une vecteur propre, on veut alors qu'avec $\pi(\theta) = \mathbb{P}\Big\{\theta \;\Big|\; \mathbf{x}\Big\}$,

$$\sum_a \pi(a)\mathbf{M}[a,b] = \pi(b)$$

en choisissant les cases de la matrice $\mathbf{M}$.

MCMC // *(algorithme de MCMC générique)*
M1 initialiser θ_0 // *(p.e., au hasard selon distribution $\pi(\theta)$)*
M2 **for** $i \leftarrow 1, 2, \ldots$ **do**
M3 générer θ aléatoire selon la distribution conditionnelle de transitions $\mathbf{M}[\theta_i, \cdot]$
M4 $\theta_i \leftarrow \theta$ // *(écrire valeur échantillonnée dans un fichier)*

Algorithme Metropolis-Hastings. Dans l'algorithme de Metropolis-Hastings, on *propose* une nouvelle vecteur de paramètres θ' selon une règle quelconque de propositions $Q(\theta \to \theta')$. On calcule le *Hastings ratio*

$$\alpha = \frac{\mathbb{P}\Big\{\theta' \;\Big|\; \mathbf{x}\Big\} \cdot Q(\theta' \to \theta)}{\mathbb{P}\Big\{\theta \;\Big|\; \mathbf{x}\Big\} \cdot Q(\theta \to \theta')}.$$

Si $\alpha \geq 1$, on *accepte* ($\theta_i \leftarrow \theta'$), sinon on rejette ($\theta_i \leftarrow \theta_{i-1}$) la proposition.

MH // *(algorithme Metropolis-Hastings)*
H1 initialiser θ_0 // *(p.e., au hasard selon distribution $\pi(\theta)$)*
H2 **for** $i \leftarrow 1, 2, \ldots$ **do**
H3 générer θ' aléatoire selon la distribution de propositions $Q(\theta_{i-1} \to \theta)$
H4 $\alpha \leftarrow \frac{L(\theta') \cdot \pi(\theta') \cdot Q(\theta' \to \theta_{i-1})}{L(\theta_{i-1}) \cdot \pi(\theta_{i-1}) \cdot Q(\theta_{i-1} \to \theta')}$
H5 **if** $\alpha \geq 1$ ou $\mathsf{rand}() < \alpha$ **then** $\theta_i \leftarrow \theta'$ **else** $\theta_i \leftarrow \theta_{i-1}$

Notez qu'en Ligne H4, il suffit de calculer les vraisemblances parce que le dénominateur normalisant en (7.4) est le même pour θ_{i-1} et θ'.