Introduction à l'algorithme EM **2009/2010**

Cours 2 — 7 octobre

Enseignant: Francis Bach *Scribe: Arnaud Fouchet, Pierre-Alain Reigneron*

2.1 Quelques *brefs* rappels d'optimisation

(Les rappels qui vont suivre représentent la fin du cours du 30 septembre)

2.1.1 Cadre

Dans la suite de cette partie, on va s'intéresser à la minimisation sans contrainte d'une fonction convexe $x \mapsto f(x)$, régulière (typiquement C^2). On rappelle qu'on a le résultat:

$$x^* \text{est le minimum global pour } f \text{ si et seulement si } \nabla_x f(x^*) = 0.$$

Ces méthodes permettent, entre autres, de résoudre des problèmes où on ne dispose pas de solution analytique

2.1.2 Cas sans solution analytique : la régression logistique

les données du problème sont :

- la variable de sortie y $\in \{0, 1\}$
- la variable d'entrée x $\in \mathbb{R}^p$
- on suppose la probabilité $p_\theta(y = 1|x) = \sigma(\theta^T x)$
  avec $\sigma(r) = \frac{1}{1+e^{-r}}$ avec r$\in \mathbb{R}$
- on dispose de données $(x_i, y_i)_{i\in\{1,\dots,n\}}$ i.i.d.

On calcule la log vraisemblance de θ

$$l(\theta) = \sum_{i=1}^{n} y_i log(\sigma(\theta^T x_i)) + (1 - y_i) log(1 - \sigma(\theta^T x_i))$$

Nous ne connaissons pas de solutions analytiques à ce problème, nous allons donc utiliser des méthodes itératives d'optimisation.

2.1.3 Méthode du première ordre, descente de gradient

Une descente produit une séquence $x^{(k)}$ telle que $x^{(k+1)} = x^{(k)} + \varepsilon^{(k)} d^{(k)}$ où $d^{(k)}$ est une direction de descente, $\varepsilon^{(k)} > 0$ est le pas, de sorte à avoir $f(x^{(k+1)}) < f(x^{(k)})$ (sauf pour le $x^{(k)}$ optimal).
Dans la cadre de la descente de gradient, on choisit $d^{(k)} = -\nabla_x f(x^{(k)})$.
Il existe plusieurs stratégies pour définir le pas $\varepsilon^{(k)} > 0$:

1. Le pas constant: $\varepsilon^{(k)} = \varepsilon$. Dans ce cas l'algorithme n'est pas toujours convergent.
2. Un pas décroissant $\varepsilon^{(k)} \propto \frac{1}{k}$ (avec $\sum_k \varepsilon^{(k)} = \infty$ et $\sum_k {\varepsilon^{(k)}}^2 < \infty$). Toujours convergent.
3. La "*Line Search*" qui cherche à trouver $\min_\varepsilon f(x^{(k)} + \varepsilon d^{(k)})$:
   - soit de manière exacte (en pratique, c'est une opération coûteuse et souvent inutile). Toujours convergent.
   - soit de manière approchée (voir le chapitre 3 de [BGLS06]). Toujours convergent.

2.1.4 Méthode du second ordre, méthode de Newton

L'idée sous-jacente à la méthode de Newton est de minimiser l'approximation quadratique de f en $x^{(k)}$, $x \mapsto \tilde{f}(x) = f(x^{(k)}) + \nabla_x f(x^{(k)})^T (x - x^{(k)}) + (x - x^{(k)})^T \nabla_x^2 f(x^{(k)})(x - x^{(k)})$.
On reprend le même schéma de descente décrit précédemment avec désormais, la direction de descente égale à $d^{(k)} = -(\nabla_x^2 f(x^{(k)}))^{-1} \nabla_x f(x^{(k)})$ (on suppose que la Hessienne est bien conditionnée en $x^{(k)}$).
Pour rendre la méthode de Newton globalement convergente, il est nécessaire d'avoir recours à une *Line Search* (voir le chapitre 4 de [BGLS06] ou encore la partie 9.5 de [BV04]).
La complexité algorithmique de cette méthode est de l'ordre de $O(p^3 + p^2 n)$ (où p et n sont respectivement la dimension et le nombre de points), correspondant à la formation ainsi qu'à l'inversion de la Hessienne. Il existe de nombreuses extensions de méthodes Newtoniennes - telles que les méthodes dites Quasi-Newtoniennes - qui essaient de réduire la complexité de la procédure en approximant le calcul de la Hessienne.

Remarque: en pratique, lorsque l'on utilisera la méthode de Newton dans la cas de la régression logistique, on pourra constater une convergence en environ 20 itérations, sans line search. De plus, dans le cas de la régression logistique, si le nombre d'observations n est grand devant la dimension de la variable d'entrée p, alors la méthode de Newton est globalement convergente.

2.2 Méthode génératives pour la classification supervisée

Jusqu'ici, on regardait seulement $p(y = 1 \mid x)$. Ici, nous allons modéliser la variable d'entrée x. On remarque, avec les formules de Bayes que

$$p(y = \varepsilon|x) = \frac{p(x|y = \varepsilon)p(y = \varepsilon)}{\underbrace{p(x)}_{independant\,de\,\varepsilon}}$$

Puisque le dénominateur ne dépend pas de ε, on peut écrire $p(y = \varepsilon|x) \propto p(x|y = \varepsilon)p(y = \varepsilon)$. En notant $p(y = \varepsilon) = \pi_\varepsilon$, nous avons seulement besoin d'une probabilité $p(x|y = \varepsilon)$.

2.2.1 Modèle Gaussien

Nous choisissons un modèle gaussien $p(x|y = \varepsilon) = \mathcal{N}(\mu_\varepsilon, \Sigma_\varepsilon)$. Utilisons ceci notre relation sur $p(y = \varepsilon|x)$

$$p(y = \varepsilon|x) \propto \frac{\Pi_\varepsilon}{(det\Sigma_\varepsilon)^{\frac{1}{2}}} \exp(\frac{-1}{2}(x - \mu_\varepsilon)^T\Sigma_\varepsilon^{-1}(x - \mu_\varepsilon))$$

2.2.2 Hypothèse "Linear Discriminant Analysis"

Pour continuer nos calculs, nous allons faire l'hypothèse que les matrices de variances sont les mêmes pour $\varepsilon = 1$ ou $= 0$ (hypothèse "Linear Discriminant Analysis"). Alors, on obtient

$$p(y = \varepsilon|x) \propto \underbrace{\frac{\Pi_\varepsilon}{(det\Sigma_0)^{\frac{1}{2}}}}_{indpendant\,de\,\mu_\varepsilon} \exp(\frac{-1}{2}(\underbrace{x^T\Sigma_0^{-1}x}_{indpendant\,de\,\mu_\varepsilon} - 2x^T\Sigma_0^{-1}\mu_\varepsilon + \mu_\varepsilon^T\Sigma_0^{-1}\mu_\varepsilon))$$

En regroupant tous les facteur qui ne dépendent pas de μ_ε, on peut synthétiser la formule en :

$$p(y = \varepsilon|x) = C_\varepsilon \exp(-D_\varepsilon^T x)$$

D'où :

$$p(y = 1 \mid x) = \frac{p(y = 1 \mid x)}{p(y = 1 \mid x) + p(y = 0 \mid x)}$$

$$p(y = 1 \mid x) = \frac{C_1 \exp(-D_1^T x)}{C_1 \exp(-D_1^T x) + C_0 \exp(-D_0^T x)}$$

$$p(y = 1 \mid x) = \frac{1}{1 + \exp(-(D_0 - D_1)^T x + log(\frac{C_1}{C_0}))}$$

$$p(y = 1 \mid x) = \sigma((D_0 - D_1)^T x + log(\frac{C_1}{C_0}))$$

On remarque que pour les méthodes d'optimisation, on calculait un maximum de vraisemblance sur une probabilité conditionnelle, alors que pour la méthode générative on calcule ce maximum de vraisemblance pour une probabilité jointe. De même pour la LDA on fait une hypothèse gaussienne sur X alors qu'on en fait pas dans le cas de l'optimisation.

2.3 Expectation-Maximization (EM)

Jusqu'à présent, on a abordé des situations où on cherchait à trouver le paramètre θ qui maximisait la vraisemblance de modèles $p_\theta(x)$ lorsque les données x sont effectivement *observables* (autrement dit, on a accès à des réalisations x de la variable aléatoire X). On va désormais étudier des modèles $p_\theta(x, z)$ pour lesquels, seuls les x sont observables. Cela nous amène à introduire l'algorithme EM (Expectation-Maximization).

2.3.1 Cadre théorique

Notation X représente les variables aléatoires observées, Z les variables aléatoires cachées et θ les paramètres du modèle.

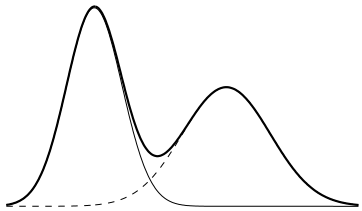

Figure 2.1. Exemple de distribution pour laquelle il est naturel d'introduire une variable cachée.

Situations pratiques d'utilisation :

1. Il y a des données manquantes (situation fréquente dans l'industrie).

2. Le modèle est plus simple si on introduit une variable cachée (cf figure 2.1).
   Ici, on a $Z \in \{1, 2\}$ et $X|Z = i \sim \mathcal{N}(\mu_i, \Theta_i)$.
   $p(x) = \sum_z p(x, z) = \sum_z p(z)p(x|z) = \sum_i p(z = i)\mathcal{N}(\mu_i, \Theta_i)$.
   La densité $p(x)$ est une combinaison convexe de densités normales.

On parle de modèle de mélanges ("mixtures").
Sans introduire de variable cachées, la représentation de $p(x)$ aurait posé problème. On peut citer l'exemple de la distribution des tailles parmi une population, distribution qui se modélise mieux par une approche multimodale (le sexe des individus en l'occurence).

Pour parler d'estimation de paramêtres "cachés", les Français et les Anglais utilisent des appelations qui peuvent porter à confusion. Dans un cadre supervisé, les Anglais parleront de *classification*, alors que les Français utiliseront *discrimination*. Dans un contexte non-supervisé, les Anglais parleront cette fois de *clustering*, alors que les Français utiliseront *classification*.

Cadre classique On a des données i.i.d., $x_1, \ldots, x_n$.

$$p(Donnees|\theta) = \prod_i p(x_i|\theta) = \prod_i \sum_{z_i} p(x_i, z_i|\theta)$$

$$\log p(Donnees|\theta) = \sum_i \log \sum_{z_i} p(x_i, z_i|\theta)$$

. Déux solutions s'offrent alors à nous :

1. Maximiser directement s'il est possible d'utiliser la structure intrinsèque au problème étudié.
2. Utiliser l'algorithme Expectation Maximization (EM).

Un résultat utile pour l'étude de l'algorithme EM: l'inégalité de Jensen

1. Soit f une fonction convexe et x une variable aléatoire. Si $\mathbb{E}f(x) < \infty$ et $f(\mathbb{E}x) < \infty$, alors $\mathbb{E}f(x) \geq f(\mathbb{E}x)$.
2. Soit f une fonction strictement convexe et x une variable aléatoire. Si $\mathbb{E}f(x) < \infty$ et $f(\mathbb{E}x) < \infty$, alors $\mathbb{E}f(x) \geq f(\mathbb{E}x)$ avec égalité si et seulement si X est constante presque sûrement.

L'algorithme EM Nous introduisons la fonction $q(z|x)$ ayant les propriétés d'une probabilité, c'est à dire $q(z|x) \geq 0$ et $\forall x \sum_z q(z|x) = 1$.

Nous avons alors:

$$\begin{aligned}
\log p(x|\theta) &= \log \sum_z p(x,z|\theta) \\
&= \log \sum_z \left(\frac{p(x,z|\theta)}{q(z|x)} \right) q(z|x) \\
&\geq \sum_z q(z|x) \log \frac{p(x,z|\theta)}{q(z|x)}, \text{ par l'inégalité de Jensen} \\
&= \sum_z q(z|x) \log p(x,z|\theta) - \sum_z q(z|x) \log q(z|x) \\
&= \mathcal{L}(q,\theta)
\end{aligned}$$

L'application de l'inégalité de Jensen montre que $\log p(x|\theta) = \mathcal{L}(q,\theta)$ si et seulement si pour tout z, $\frac{p(x,z|\theta)}{q(z|x)}$ est constant. Cela est équivalent à dire que $q(z|x) = p(z|x,\theta)$ grâce aux conditions de normalisation de $q(z|x)$ et $p(z|x,\theta)$.

Proposition 2.1 *Pour tout q, on a $\log p(x|\theta) \geq \mathcal{L}(q,\theta)$, avec égalité si et seulement si $q(z|x) = p(z|x,\theta)$.*

L'algorithme Expectation Maximization consiste à maximiser $\mathcal{L}(q,\theta)$ de manière alternée. On commence par initialiser θ_0 puis, pour tout $t \geq 0$ on calcule

- E-step : $q_{t+1} \in \text{argmax}_q \mathcal{L}(q,\theta_t)$
  On fait $q_{t+1}(z|x) = p(z|x,\theta_t)$. On trouve la meilleure borne inf.
- M-step : $\theta_{t+1} \in \text{argmax}_\theta \mathcal{L}(q_{t+1},\theta)$
  C'est l'étape de maximisation, il faut trouver le maximum de la borne inf.

Les propriétés de l'algorithme EM

1. $\forall t, p(x|\theta_{t+1}) \geq p(x|\theta_t)$
2. L'algorithme convergence vers un point stationnaire de $\theta \mapsto p(x|\theta)$.
3. Nous sommes dans un cas non convexe (notre problème n'est pas convexe de manière jointe en (q,θ)). L'algorithme EM souffre par conséquent de plusieurs défauts:
   - L'optimum global n'est pas garanti
   - La limite dépend *fortement* de l'initialisation
   - L'optimum global a souvent une vraisemblance ∞

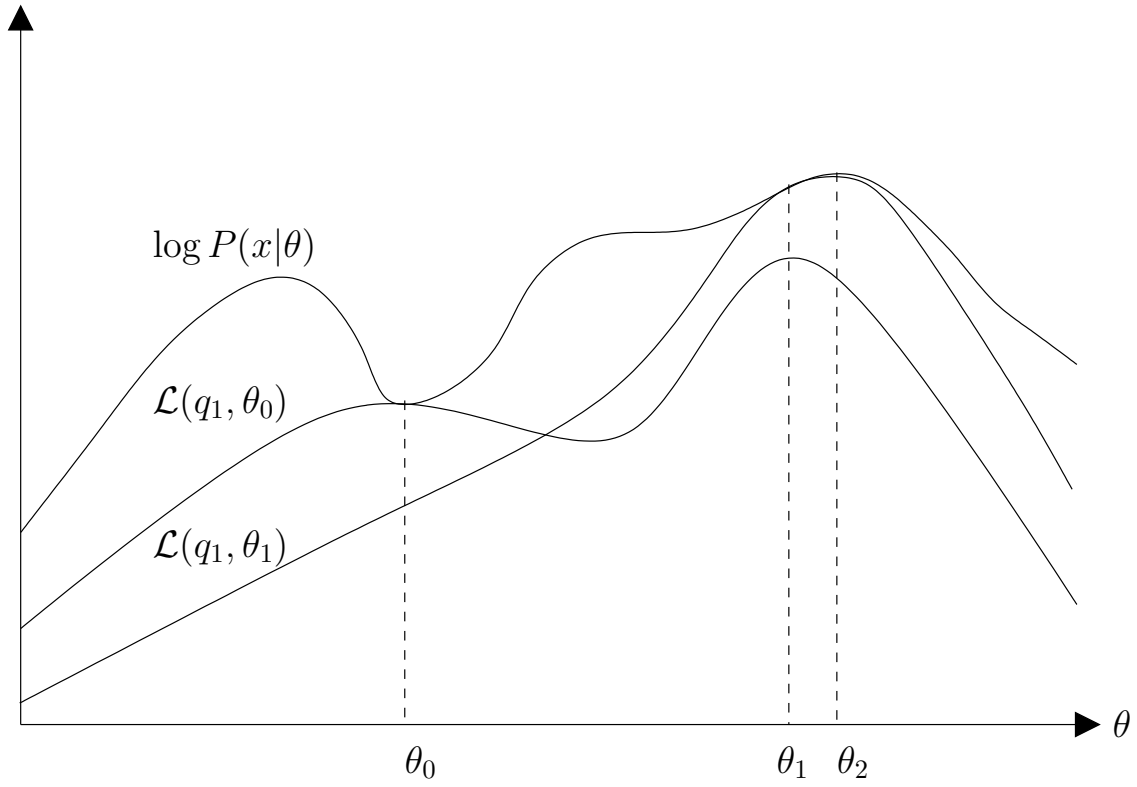

Figure 2.2. Illustration de l'algorithme EM: minimisations itératives de bornes inférieures de la log-vraisemblance.

Recette Les développements précédents permettent d'utiliser la *recette* suivante:

1. Écrire la vraisemblance complète $l_c = \log p(x, z|\theta)$.
2. E-step : espérance de l_c sous $p(z|x, \theta)$. On obtient une fonction de θ. On veut $q(z|x) = p(z|x, \theta)$
3. M-step : maximiser en fonction de θ.

2.3.2 Sur un exemple : mixtures Gaussiennes

Notation Z prends q valeurs et suit une loi multinomiale Π, $X \in \mathbb{R}^d$ est tel que $X|Z = j \sim \mathcal{N}(\mu_j, \Sigma_j)$.

Nous avons des données $x_i \in \mathbb{R}^d$ i.i.d. et nous voulons estimer $p(z|x)$ et les paramètres $\theta = (\mu, \Sigma, \Pi)$.

Calcul de $p(z|x)$

$$\begin{aligned} p(z=j|x) &= \frac{p(x|z=j)p(z=j)}{\sum_k p(x|z=k)p(z=k)} \\ &= \frac{\Pi_j \mathcal{N}(x|\mu_j,\Sigma_j)}{\sum_k \Pi_k \mathcal{N}(x|\mu_k,\Sigma_k)} \\ &\propto \exp\left(\log \Pi_j - \frac{1}{2}(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j) - \log\det(\Sigma_j)^{1/2}\right) \end{aligned}$$

Remarque : $p(z=j|x)$ est le softmax des valeurs

$$\log \Pi_j - \frac{1}{2}(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j) - \log\det(\Sigma_j)^{1/2}.$$

Estimation de $\theta = (\Pi, \mu_j, \Sigma_j)$ On suite la "recette EM":

Vraisemblance complète

$$\begin{aligned} l_c = \log p(x,z|\theta) &= \sum_{i=1}^n \log p(x_i,z_i|\theta) \\ &= \sum_{i=1}^n \sum_{k=1}^q \delta(z_i=k)\log p(x_i,k|\theta) \\ &= \sum_{i=1}^n \sum_{k=1}^q z_i^k(\log p(z=k|\theta) + \log p(x_i|z_i=k,\theta)), \text{ avec } z_i^k = \delta(z_i=k) \\ &= \sum_{i=1}^n \sum_{k=1}^q z_i^k(\log \pi_k + \log \mathcal{N}(x_i|\mu_k,\Sigma_k)) \\ < l_c >_{z|x,\theta} &= \sum_{i=1}^n \sum_{k=1}^q < z_i^k > (\log \pi_k + \log \mathcal{N}(x_i|\mu_k,\Sigma_k)) \end{aligned}$$

E-step

$$\begin{aligned} \tau_i^k(t) &= < z_i^k > \\ &= \mathbb{E}(\delta(z_i=k)) \\ &= p(z_i=k|x_i,\theta(t)) \\ &= \frac{\pi_k(t)\mathcal{N}(x_i|\mu_k(t),\Sigma_k(t))}{\sum_s \pi_s(t)\mathcal{N}(x_i|\mu_s(t),\Sigma_s(t))} \end{aligned}$$

M-step maximisation par rapport à π: $\sum_k(\sum_i \tau_i^k(t)) \log \pi_k$ Donc $\pi_k(t+1) = 1/n \sum_i \tau_i^k(t)$
par rapport à μ_k, Σ_k
$\sum_i \tau_i^k(t) \log \mathcal{N}(x_i|\mu_k(t).\Sigma_k(t)) = \sum_i \tau_i^k(t) - \frac{1}{2}(x_i - \mu_k)^T \Sigma_k^{-1}(x_i - \mu_k) - \log \det(\Sigma_k)^{1/2}$

$$\mu_k(t+1) = \frac{\sum_i \tau_i^k(t) x_i}{\sum_i \tau_i^k(t)}$$

$$\Sigma_k(t+1) = \frac{\sum_i \tau_i^k(t)(x_i - \mu_k(t+1))(x_i - \mu_k(t+1))^T}{\sum_i \tau_i^k(t)}$$

2.3.3 Un problème pratique de l'algorithme EM : l'initialisation

Comme l'algorithme EM trouve un minimum local, son initialisation est très importante et on essaie donc de trouver une configuration initiale proche du maximum de vraisemblance global.

Pour cela, on utilise l'algorithme K-means qui, étant donné un échantillon de points x_i (que l'on suppose séparable en K clusters de centres μ_k) va chercher à déterminer ces μ_k. Il est important de remarquer ici que le paramêtre K est considéré comme connu.

À chaque x_i, on assigne une les variables indicatrices z_i^k telles que $z_i^k = 1$ ssi x_i appartient au cluster k. L'algorithme K-means va alors chercher à minimiser la fonctionnelle suivante, appelée "mesure de distortion" :

$$J(z, \mu) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k \|x_i - \mu_k\|^2$$

l'algorithme K-means L'algorithme effectue une minimisation alternée :

- minimisation par rapport à z: $z_i^k = 1$ pour $k \in \arg\min \|x_i - \mu_k\|^2$
- minimisation par rapport à μ: $\mu_k = \frac{\sum_i z_i^k x_i}{\sum_i z_i^k}$

Remarque: on peut voir ces deux étapes comme une version *hard* de l'algorithme EM, où les assignements des points aux moyennes ont des probabilites égale à 0 ou 1.

Les propriétés de l'algorithme K-means K-means converge vers un minimum local, donc on l'utilise de la manière suivante :

- On lance K-means un grand nombre de fois, avec des initialisations aléatoires.
- On retient les μ pour lesquels on obtient la plus faible distortion.
- On les utilise pour initialiser l'algorithme EM (avec des variances larges)

Bibliography

[BGLS06] J. Frédéric Bonnans, Jean Charles Gilbert, Claude Lemaréchal, and Claudia A. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects (Universitext)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. CUP, 2004.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.