Estimation de mélange de Gaussiennes sur données compressées

Anthony BOURRIER[12], Rémi GRIBONVAL[2], Patrick PÉREZ[1]

[1]Technicolor
975 Avenue des Champs Blancs, 35576 Cesson-Sévigné, France

[2]INRIA Rennes-Bretagne Atlantique
Campus de Beaulieu, 35042 Rennes, France

anthony.bourrier@technicolor.com, remi.gribonval@inria.fr,
patrick.perez@technicolor.com

Résumé – Estimer les paramètres d'un mélange de densités à partir d'un ensemble de vecteurs requiert typiquement une utilisation de mémoire d'autant plus importante que les données sont volumineuses. Nous proposons ici un cadre dans lequel les données sont collectivement compressées en une représentation de taille fixe, appelée *sketch*, représentant des moments empiriques calculés sur les données. Par analogie avec un problème de *compressive sensing*, nous proposons un algorithme d'estimation des paramètres à partir du *sketch*. Nous montrons expérimentalement que l'algorithme permet une estimation précise des paramètres tout en consommant moins de mémoire qu'un algorithme EM dans le cas de données nombreuses. Il permet également une estimation sans accès aux données dans le cas où celles-ci sont confidentielles.

Abstract – Estimating a probability mixture model from a set of vectors typically requires a large amount of memory if the data is voluminous. We propose a framework where the data is jointly compressed to a fixed-size representation called sketch, composed of empirical moments calculated from the data. By analogy with compressive sensing, we derive a parameter estimation algorithm from the sketch. We experimentally show that our algorithm allows precise estimation while consuming less memory than an EM algorithm for voluminous data. The algorithm also provides a privacy-preserving estimation tool since the sketch does not disclose information about individual datum it is based on.

1 Introduction

L'exploitation d'une collection de données pour des tâches d'apprentissage fait souvent intervenir une étape d'approximation du nuage de points considéré par un modèle probabiliste de mélange de densités pouvant servir plusieurs fins différentes.

Pour estimer les paramètres du mélange, les méthodes classiques consistent en une optimisation de fonction de coût et requièrent souvent de nombreux accès mémoire aux données de départ. Dans le cas où les données sont nombreuses, l'ensemble de départ est ainsi généralement sous-échantillonné afin de se conformer aux limitations des ressources informatiques disponibles pour exécuter l'algorithme.

Nous proposons ici de calculer un *sketch* des données initiales, que nous définissons comme une concaténation de moments empiriques aléatoires calculés sur les données en une passe. Ce genre de représentation est classique dans le domaine du traitement de bases de données [7]. Nous cherchons à estimer, à partir de cette forme compressée, les paramètres d'un mélange sans retourner aux données de départ. Cela permet ainsi une économie de mémoire dans le cas où celles-ci sont nombreuses. La motivation de l'approche peut être rapprochée de celle du *compressed learning* [11]. L'idée générale est illustrée en Figure 1. Une liste de travaux liés exploitant des représentations similaires est revue en Section 2.

Pour estimer les paramètres, la démarche adoptée est la recherche de mélanges ayant un *sketch* similaire à celui calculé sur les données. Ce problème, défini en Section 3 dans le cadre de l'estimation d'un mélange de Gaussiennes, est analogue à celui d'un problème inverse et le calcul des moments empiriques aléatoires s'apparente à une projection aléatoire de la densité sous-jacente, inspirée du *compressed sensing* [8]. Nous proposons en Section 4 un algorithme inspiré de l'*Iterative Hard Thresholding* (IHT) [2] pour le résoudre. En Section 5,

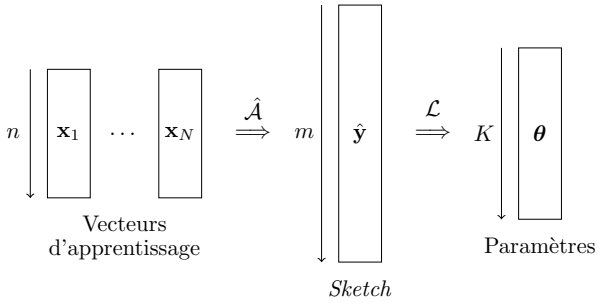

FIG. 1: *Illustration de la méthode proposée. L'ensemble d'apprentissage est compressé en un* sketch $\hat{\mathbf{y}}$ *par un opérateur* $\hat{\mathcal{A}}$ *et les paramètres* $\boldsymbol{\theta}$ *du mélange sont estimés par un algorithme d'apprentissage* $\mathcal{L}$ *exploitant uniquement le* sketch*. Intuitivement, on doit avoir* $K \leq m$ *pour que l'estimation soit possible et* $m \ll Nn$ *pour bénéficier d'une réduction effective du volume de données à traiter.*

nous testons expérimentalement cet algorithme pour estimer des mélanges en dimension 20 et comparons notre approche avec un algorithme EM classique en terme de qualité de l'approximation et de coût mémoire.

2 Travaux liés

Des algorithmes utilisant des *sketchs* apparaissent dans la littérature des bases de données [9, 7]. Dans ce cas, un *sketch* peut être mis à jour à chaque ajout ou suppression d'élément de la base de données sans reconsidérer toute la base. Une application usuelle est la recherche de motifs récurrents d'un flux de données [6].

L'estimation d'histogramme à partir d'un *sketch* a été explorée [13] dans le cas des vecteurs de petite dimension (2 et 3). Le *sketch* est une projection linéaire aléatoire accumulée des vecteurs. Cette méthode ne s'applique pas aux grandes dimensions, la construction de l'histogramme à partir du *sketch* ayant une complexité exponentielle en la dimension.

Des problèmes inverses sur des mélanges de densités ont également été considérés [5, 1]. Les deux travaux proposent, à partir de données tirées selon le mélange de densités, de formuler le problème de reconstruction comme l'optimisation d'une fonctionnelle favorisant la parcimonie des poids du mélange. Les deux méthodes requièrent que l'ensemble des densités de probabilité soit fini et que cet ensemble soit incohérent. Ces hypothèses ne sont pas vérifiées dans notre cas : les centres des Gaussiennes peuvent varier de façon continue et peuvent être arbitrairement proches les uns des autres.

Dans [12], une représentation compressée de lois de probabilités sur des vecteurs binaires est étudiée. La compression est effectuée en mesurant la distribution de probabilité sous-jacente via des perceptrons aléatoires. L'auteur prouve que la représentation compressée conserve assez d'information pour pouvoir reconstruire la distribution initiale via une minimisation ℓ_1. Nous nous plaçons dans un cadre différent dans lequel les données peuvent prendre des valeurs continues.

3 Définition du problème

Soit $F \subset L^1(\mathbb{R}^n)$ un ensemble de densités de probabilité, *i.e.* $\forall f \in F, f$ est positive et $||f||_1 = \int_{\mathbb{R}^n} f(\mathbf{x})d\mathbf{x} = 1$.

Soit $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^n$ un ensemble de vecteurs indépendants tirés selon le mélange $p = \sum_{s=1}^{k} \alpha_s f_s$, avec $\forall s, f_s \in F$, $\alpha_s \geq 0$ et $\sum_{s=1}^{k} \alpha_s = 1$. On cherche à inférer α_s et f_s à partir de $\mathcal{X}$.

On considère le cas où F est une famille de Gaussiennes isotropes dont la matrice de covariance est de la forme $\sigma^2\mathbf{Id}$, où $\sigma > 0$:

$$F = \left\{ f_{\boldsymbol{\mu}} : \mathbf{x} \mapsto \frac{1}{(2\pi)^{\frac{n}{2}}\sigma^n} \exp\left(\frac{\|\mathbf{x}-\boldsymbol{\mu}\|_2^2}{2\sigma^2}\right), \boldsymbol{\mu} \in \mathbb{R}^n \right\}. \tag{1}$$

Etant données m fréquences $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m \in \mathbb{R}^n$, l'opérateur de *sketch* $\mathcal{A}$ est défini comme :

$$\mathcal{A} : L^1(\mathbb{R}^n) \to \mathbb{C}^m, \qquad q \mapsto (h_1(q), \ldots, h_m(q)), \tag{2}$$

où $h_j(q) = \int_{\mathbb{R}^n} q(\mathbf{x}) e^{-i\langle \mathbf{x}, \boldsymbol{\omega}_j \rangle} d\mathbf{x}$ et $\langle ., . \rangle$ est le produit scalaire canonique de $\mathbb{R}^n$. Le *sketch* d'un mélange q de densités est ainsi obtenu en échantillonnant la fonction caractéristique de q en m points. Les données empiriques $\mathcal{X}$ permettent de calculer une approximation $\hat{\mathbf{y}}$ de la quantité $\mathcal{A}p$ par la formule $\hat{\mathbf{y}} = \left(\hat{h}_1(\mathcal{X}), \ldots, \hat{h}_m(\mathcal{X})\right)$, où :

$$\hat{h}_j(\mathcal{X}) = \frac{1}{N} \sum_{i=1}^{N} e^{-i\langle \mathbf{x}_i, \boldsymbol{\omega}_j \rangle}. \tag{3}$$

La reconstruction de p comme un mélange de k fonctions de F peut être exprimée comme la minimisation du coût suivant :

$$\hat{p} = \underset{q \in \Sigma_k}{\operatorname{argmin}} \|\hat{\mathbf{y}} - \mathcal{A}q\|_2^2, \tag{4}$$

où $\Sigma_k := \{f = \sum_{s=1}^{k} \alpha_s f_s | \forall s, f_s \in F \wedge \alpha_s \geq 0\}$. Il est à noter que nous ne contraignons pas les α_s à sommer à 1, ce qui n'est pas préjudiciable à la qualité de la reconstruction pour les cas expérimentaux étudiés ci-après.

Ce problème de reconstruction rappelle un problème de *compressed sensing*, où l'on cherche à reconstruire un signal discret $\mathbf{x}$ parcimonieux dans une certaine base à partir d'un nombre restreint de ses images par des formes linéaires souvent aléatoires. Ici, la dimension ambiante est inifinie et l'on suppose que le vecteur est parcimonieux dans un ensemble infini et très cohérent, *i.e.* composé de vecteurs qui peuvent être très corrélés (ce qui arrive pour deux Gaussiennes lorsque leurs centres sont proches).

4 Algorithme

Pour résoudre le problème 4, nous proposons un algorithme similaire à l'*Iterative Hard Thresholding* (IHT) [2], qui vise à résoudre le même problème dans le cas où le support est fini. Son principe est résumé ci-dessous.

4.1 Cas discret

L'IHT vise à reconstruire un signal $\mathbf{x}$ k-parcimonieux de dimension n à partir de m mesures définies par $\hat{\mathbf{y}} = \mathbf{A}\mathbf{x} + \mathbf{e}$, où $\mathbf{A}$ est une matrice de taille $m \times n$ (avec $m < n$) et $\mathbf{e}$ est un bruit de mesure. À chaque itération, l'estimation $\hat{\mathbf{x}}$ de $\mathbf{x}$ est mise à jour en diminuant la fonction objectif suivante :

$$\Phi : \mathbf{z} \mapsto \frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{A}\mathbf{z}\|_2^2, \tag{5}$$

tout en s'assurant de la k-parcimonie de $\hat{\mathbf{x}}$. La procédure est accomplie en deux étapes :

1. Une descente de gradient est effectuée sur Φ et $\hat{\mathbf{x}}$ est mis à jour selon $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} - \lambda\mathbf{g}$, où λ est un pas de descente et $\mathbf{g}$ est le gradient courant de Φ.

2. $\hat{\mathbf{x}}$ est projeté sur l'ensemble des vecteurs k-parcimonieux : ceci est effectué en appliquant sur $\hat{\mathbf{x}}$ un seuillage H_k qui ne conserve que les k plus grands coefficients (en magnitude) du vecteur et fixe les autres à 0.

4.2 Cas continu

Ici, le "signal" à reconstruire est le mélange de densités p et l'estimation $\hat{p} = \sum_{s=1}^{k} \hat{\alpha}_s f_{\hat{\boldsymbol{\mu}}_s}$ est paramétrée par un vecteur $\hat{\boldsymbol{\alpha}}$ de poids positifs et par le support $\hat{\Gamma} = \{\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k\}$ correspondant aux moyennes des Gaussiennes. Le résidu courant est défini par $\hat{\mathbf{r}} = \hat{\mathbf{y}} - \mathcal{A}\hat{p}$, où $\hat{\mathbf{y}}$ joue le rôle de la mesure bruitée, analogue au cas précédent.

Les différences de cadre d'application avec l'IHT requièrent une modification de la procédure. La principale différence dans notre cas est que le gradient n'est pas de dimension finie mais indexé continûment sur $\mathbb{R}^n$. Il est défini par la fonction suivante :

$$g_{\hat{p}}(\boldsymbol{\mu}) = \left(\frac{\partial}{\partial t} \frac{1}{2} \|\hat{\mathbf{y}} - \mathcal{A}(\hat{p} + t f_{\boldsymbol{\mu}})\|_2^2 \right)_{t=0} = -\langle \mathcal{A} f_{\boldsymbol{\mu}}, \hat{\mathbf{r}} \rangle. \quad (6)$$

Cette fonction représente la variation infinitésimale de la fonction de coût (4) lorsque $f_{\boldsymbol{\mu}}$ est ajoutée au support courant. Contrairement à l'IHT, on ne peut pas ajouter temporairement tous les vecteurs du support à l'estimation courante, on doit donc en choisir un nombre fini qui constituent de "bons candidats". Il s'agit de minima locaux de la quantité (6). L'étape de seuillage consiste ensuite en une projection du *sketch* $\hat{\mathbf{y}}$ sur les $\mathcal{A} f_{\boldsymbol{\mu}}$, où $\boldsymbol{\mu}$ est un paramètre du support courant. Enfin, une étape de descente de gradient est effectuée sur la fonction objectif avec une initialisation sur les paramètres courants. Une itération de notre algorithme comprend ainsi trois phases :

1. M minima locaux de la fonction $\boldsymbol{\mu} \mapsto g_{\hat{p}}(\boldsymbol{\mu})$ sont recherchés à l'aide d'une descente de gradient initialisée aléatoirement. Ces minima $\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_M$ sont ajoutés au support courant : $\hat{\Gamma}' \leftarrow \hat{\Gamma} \cup \{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_M\}$.
2. Le *sketch* $\hat{\mathbf{y}}$ est projeté sur ce support avec une contrainte de positivité sur les coefficients. Si $\hat{\Gamma}' = \{\mathbf{u}_1, \ldots, \mathbf{u}_K\}$, on calcule ainsi $\hat{\boldsymbol{\alpha}}'$ solution du problème :

$$\underset{\boldsymbol{\beta} \in \mathbb{R}_+^K}{\operatorname{argmin}} \; ||\hat{\mathbf{y}} - \mathbf{U}\boldsymbol{\beta}||_2, \quad (7)$$

   avec $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]$.
   Seuls les k plus grands coefficients et les vecteurs correspondants du support sont gardés et respectivement consignés dans le vecteur $\hat{\boldsymbol{\alpha}}$ et le support $\hat{\Gamma} = \{\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k\}$.
3. Un algorithme de descente de gradient est appliqué sur la fonction objectif afin de mettre à jour les poids et les vecteurs du support tout en réduisant le fonction de coût. On cherche donc un minimum local de la fonction :

$$\begin{array}{lcl} \mathbb{R}^k \times (\mathbb{R}^n)^k & \to & \mathbb{R}_+ \\ (\boldsymbol{\beta}, \boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_k) & \mapsto & \|\hat{\mathbf{z}} - [\mathcal{A} f_{\boldsymbol{\nu}_1}, \ldots, \mathcal{A} f_{\boldsymbol{\nu}_k}]\boldsymbol{\beta}\|_2, \end{array}$$

   avec une initialisation à $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k)$.

Les paramètres de l'algorithme sont donc le nombre M de minima locaux à chercher, le nombre k de composantes du mélange, et le nombre n_{iter} d'itérations à effectuer. Une description plus détaillée de l'algorithme est disponible dans [4].

4.3 Estimation de la mémoire utilisée

Considérons que n, k et m sont largement supérieurs à 1. Si l'on suppose que les algorithmes d'optimisation n'utilisent que des dérivées d'ordre 1, leurs coûts mémoire sont en $\mathcal{O}(kn)$. Le calcul de la fonction de coût de l'étape 3 de l'algorithme requiert $\mathcal{O}(km)$. Enfin, le stockage de l'opérateur $\mathcal{A}$ (via les fréquences $\boldsymbol{\omega}_j$) requiert $\mathcal{O}(mn)$.

Ainsi, la mémoire totale requise est en $\mathcal{O}((k+n)m + kn)$ et ne dépend pas du nombre N de vecteurs. En comparaison, la mémoire requise pour un algorithme EM est en $\mathcal{O}((k+n)N)$ pour stocker à la fois les vecteurs d'apprentissage et leurs probabilités d'appartenir à chaque composante du mélange. L'algorithme compressé permet des gains en mémoire dès que $m + \frac{kn}{k+n} \lesssim N$. Puisque $kn \lesssim m$, cette condition est approximativement équivalente à $m \lesssim N$.

5 Expériences

5.1 Cadre expérimental

Pour évaluer l'algorithme, des expériences ont été effectuées sur des vecteurs tirés selon un mélange de k Gaussiennes isotropes de covariance $\mathbf{Id}$. Dans chaque cas, les poids des Gaussiennes ont été tirés uniformément sur le simplexe [1], et les moyennes ont été obtenues en tirant des vecteurs Gaussiens dont tous les coefficients étaient tirés selon une loi $\mathcal{N}(0,1)$.

Les expériences ont été effectuées de la façon suivante : après le choix du mélange p, N vecteurs ont été tirés selon cette loi et le *sketch* empirique de la distribution a été calculé en un passage sur les données. L'algorithme de reconstruction a ensuite été appliqué au *sketch* pour obtenir un mélange approché $\hat{p}$. Pour l'algorithme de reconstruction, nous avons fixé $M = 2k$.

5.2 Mesures de qualité

Deux mesures de divergence ont été utilisées pour mesurer la proximité entre le vrai mélange et le mélange estimé : une version symétrisée de la divergence de Kullback-Leibler (KL) et la distance de Hellinger. Ces deux mesures sont estimées empiriquement en tirant $N' = 10^5$ vecteurs $(\mathbf{y}_i)_{i=1}^{N'}$ selon p. Les estimations sont alors respectivement données par les quantités $\frac{1}{N'} \sum_{i=1}^{N'} \left[\ln\left(\frac{p(\mathbf{y}_i)}{\hat{p}(\mathbf{y}_i)}\right) + \frac{\hat{p}(\mathbf{y}_i)}{p(\mathbf{y}_i)} \ln\left(\frac{\hat{p}(\mathbf{y}_i)}{p(\mathbf{y}_i)}\right) \right]$ pour la divergence KL et par $1 - \frac{1}{N'} \sum_{i=1}^{N'} \sqrt{\frac{\hat{p}(\mathbf{y}_i)}{p(\mathbf{y}_i)}}$ pour la distance de Hellinger. La divergence KL peut prendre des valeurs dans $\mathbb{R}_+$, la distance de Hellinger dans $[0,1]$. Dans les deux cas, la valeur est d'autant plus faible que les distributions sont proches.

1. Nous avons également fait des expériences avec des poids tous égaux à $\frac{1}{k}$ en observant des résultats similaires.

N	Compressé		
	Div. KL	Hell.	**Mém.**
10^3	0.68 ± 0.28	0.06 ± 0.01	**0.6**
10^4	0.24 ± 0.31	0.02 ± 0.02	**0.6**
10^5	0.13 ± 0.15	0.01 ± 0.02	**0.6**
N	EM		
	Div. KL	Hell.	**Mém.**
10^3	0.68 ± 0.44	0.07 ± 0.03	**0.24**
10^4	0.19 ± 0.21	0.01 ± 0.02	**2.4**
10^5	0.13 ± 0.21	0.01 ± 0.02	**24**

TAB. 1: *Comparaison entre l'algorithme compressé proposé et un algorithme EM en termes de précision de l'estimation et d'utilisation de la mémoire (en Mégaoctets). Les expériences ont été effectuées avec* $n = 20$*,* $k = 10$*,* $m = 1000$*. Dans chaque cellule, la valeur affichée est la médiane sur* 10 *expériences ainsi que l'écart-type pour la mesure de précision considérée.*

5.3 Résultats

La Table 1 compare notre algorithme avec un algorithme EM standard pour $n = 20$, $k = 10$, $m = 1000$ et un nombre N de vecteurs d'apprentissage allant de 10^3 à 10^5. Dans les deux cas, la précision de l'estimation augmente avec le nombre d'échantillons. Dans le cas de l'algorithme compressé, cela peut s'expliquer par le fait que les composantes du *sketch* sont mieux estimées avec plus de vecteurs. Nous pouvons remarquer que la mémoire requise par l'algorithme EM est proportionnelle à la taille de l'ensemble d'apprentissage, alors que la mémoire requise par l'algorithme compressé ne dépend pas de ce paramètre, ce qui aboutit à un gain substantiel de mémoire utilisée pour $N \geq 10^4$. Même avec ce coût mémoire réduit, l'algorithme compressé est capable de reconstruire le mélange avec une précision comparable à celle de l'algorithme EM.

6 Conclusion

Nous avons proposé une méthode d'estimation de mélange de densités exploitant un *sketch* des données au lieu des données elles-mêmes. Ce *sketch* est calculable en un passage sur les données, qui peuvent être supprimées directement, permettant un gain de mémoire utilisée et une confidentialité des données préservée. Nous avons instantié cette méthode sur un problème d'estimation de mélange de Gaussiennes isotropes en dérivant un alorithme par analogie avec l'IHT. Expérimentalement, nous avons montré que cet algorithme permet une reconstruction de précision comparable à celle de l'EM standard tout en utilisant sensiblement moins de mémoire lors de l'estimation dans le cas de données nombreuses.

Ces résultats laissent à penser que le *compressed sensing* peut être étendu à des modèles plus généraux que les modèles discrets usuels. Des travaux futurs chercheront à préciser le caractère bien posé de problèmes tel que celui étudié ici.

Du côté expérimental, il serait intéressant de comparer notre méthode avec des algorithmes comme l'*online EM*[10] ou le gradient stochastique[3].

Remerciements

Travaux partiellement financés par le Conseil Européen de la Recherche, projet PLEASE (ERC-StG-2011-277906).

Références

[1] K. Bertin, E. Le Pennec, and V. Rivoirard. Adaptive Dantzig density estimation. *Annales de l'Institut Henri Poincaré (B) Probabilités et Statistiques*, 47(1):43–74, 2011.

[2] T. Blumensath and M. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

[3] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Neural Information Processing Systems Conference*, 2008.

[4] Anthony Bourrier, Rémi Gribonval, and Patrick Pérez. Compressive gaussian mixture estimation. In *ICASSP*, 2013.

[5] F. Bunea, A. Tsybakov, M. Wegkamp, and A. Barbu. Spades and mixture models. *Annals of Statistics*, 38(4):2525–2558, 2010.

[6] G. Cormode and M. Hadjieleftheriou. Methods for finding frequent items in data streams. *VLDB Journal*, 19(1):3–20, 2010.

[7] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[8] D.L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52:1289–1306, 2006.

[9] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *Conference on Very Large Databases*, 2002.

[10] P. Liang and D. Klein. Online EM for unsupervised models. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2009.

[11] O.-A. Maillard and R. Munos. Compressed least-squares regression. In *Neural Information Processing Systems Conference*, 2009.

[12] X. Pitkow. Compressive neural representation of sparse, high-dimensional probabilities. In *Neural Information Processing Systems Conference*, 2012.

[13] N. Thaper, S. Guha, P. Indyk, and N. Koudas. Dynamic multidimensional histograms. In *ACM SIGMOD International conference on Management of data*, 2002.