Provable Tensor Methods for Learning Mixtures of Generalized Linear Models

Hanie Sedghi* Anima Anandkumar†

October 13, 2015

Abstract

We consider the problem of learning mixtures of generalized linear models (GLM) which arise in classification and regression problems. Typical learning approaches such as expectation maximization (EM) or variational Bayes can get stuck in spurious local optima. In contrast, we present a tensor decomposition method which is guaranteed to correctly recover the parameters. The key insight is to employ certain feature transformations of the input, which depend on the input generative model. Specifically, we employ score function tensors of the input and compute their cross-correlation with the response variable. We establish that the decomposition of this tensor consistently recovers the parameters, under mild non-degeneracy conditions. We demonstrate that the computational and sample complexity of our method is a low order polynomial of the input and the latent dimensions.

Keywords: Mixture of generalized linear models, score function, spectral/tensor decomposition.

1 Introduction

A generalized linear model (GLM) is a flexible extension of linear regression which allows the response or the output to be a non-linear function of the input via an *activation* function. In other words, in a GLM, the linear regression of the input is passed through an activation function to generate the response. GLMs unify popular frameworks such as logistic regression and Poisson regression with linear regression. At the same time, they can be learnt with guarantees using simple iterative methods (Kakade et al., 2011).

In many scenarios, however, GLMs may be too simplistic, and mixtures of GLMs can be much more effective since they combine the expressive power of latent variables with the predictive capabilities of the GLM. Mixtures of GLMs have widespread applicability including object recognition (Quattoni et al., 2004), human action recognition (Wang and Mori, 2009), syntactic parsing (Petrov and Klein, 2007), and machine translation (Liang et al., 2006).

Traditionally, mixture models are learnt through heuristics such as expectation maximization (EM) (Jordan and Jacobs, 1994; Xu et al., 1995) or variational Bayes (Bishop and Svensen, 2003). However, these methods can converge to spurious local optima and have slow convergence rates for high dimensional models. In contrast, we employ a method-of-moments approach for guaranteed learning of mixtures of GLMs.

*University of Southern California. Email: hsedghi@usc.edu

†University of California, Irvine. Email: a.anandkumar@uci.edu

The method of moments paradigm dates back to Pearson (Pearson, 1894), and involves fitting the observed moments to parametric distributions. Recently, it has been highly successful in unsupervised learning of a wide range of latent variable models such as Gaussian mixtures, topic models, hidden Markov models (Anandkumar et al., 2014a), network community models (Anandkumar et al., 2013), mixture of ranking models (Awasthi et al., 2014; Oh and Shah, 2014), and so on. The basic idea is to find an efficient spectral decomposition of low order observed moment tensors. Under natural non-degeneracy assumptions, the tensor method is guaranteed to correctly recover the underlying model parameters with low computational and sample complexities. Moreover, in practice, these methods are embarrassingly parallel and scalable to large-scale datasets (Huang et al., 2014).

Earlier works on tensor methods (Anandkumar et al., 2014a) consider unsupervised learning and a key assumption is that the observables are linear functions of the latent variables (in expectation). However, here, we consider mixtures of GLMs, which are non-linear, and this rules out a direct application of tensor methods.

We address the above challenges with the following insight: we have additional flexibility in the regression setting since we have both the response and the input. We can therefore form different moments involving transformations of the input and the response. What are the appropriate transforms for forming the moments which are amenable to tensor decomposition methods? As detailed below, the key ingredient is using a specific feature transformation of the input, based on its probability distribution.

1.1 Summary of Result

The main contribution of this work is to provide a guaranteed method for learning mixtures of GLMs using score function transformations. The m^{th} order score function $\mathcal{S}_m(x)$ is related to the normalized m^{th} order derivative of the pdf of input x, see (9). We assume knowledge of these score functions, and this can be estimated via various unsupervised learning methods using only unlabeled samples (e.g. spectral methods).

We then construct the cross-moment tensor between the response variable and the input score function. We establish that the decomposition of this tensor consistently recovers the components of the GLM mixture under some simple non-degeneracy assumptions. Let the response or the output y be generated from a mixture of GLMs: $\mathbb{E}[y|h,x] = g(\langle Uh, x\rangle) + \langle \tilde{b}, h\rangle)$, where $g(\cdot)$ is the activation function, x is the input and h is the hidden choice variable. Let r be the number of mixture components, d be the input dimension and $s_{\min}(U)$ be the r^{th} largest singular value of U. Assume the weight matrix $U = [u_1|\ldots|u_r]$ is full column rank. Then, we have the following result.

Theorem 1 (Informal Result)**.** *We recover the weight vectors $\{u_i\}$ (up to scaling) by performing tensor decomposition on the cross-moment tensor $\mathbb{E}[y\cdot\mathcal{S}_3(x)]$. If we have $n = \tilde{O}\left(\frac{d^3 r^4}{\epsilon^2 s^2_{\min}(U)}\right)$ samples, the error in recovering each weight vector u_i is bounded by ϵ.*

The above result requires third order score function $\mathcal{S}_3(x)$ to consistently estimate the weight vectors $\{u_i\}$ of the GLM components in the mixture. Note that the second order score function $\mathcal{S}_2(x)$ is only a matrix (assuming a vector input x) and can only identify the weights $\{u_i\}$ up to the subspace. Thus, we require at least the third order score function to consistently estimate the GLM mixture model. When the number of components exceeds the input dimension, the full column rank assumption on U is violated, and in this case, we can resort to higher order score functions to consistently estimate the parameters.

We employ the tensor decomposition methods from (Anandkumar et al., 2014a,b) to learn the weight vectors u_i (up to scaling). The tensor method is efficient to implement and does not suffer from spurious local optima. Thus, we guarantee consistent estimation of the weight vectors of GLM mixtures through decomposition of the cross-moment tensor involving the response variable and the input score functions. Our method is shown in Algorithm 1.

1.2 Overview of Techniques

Representation learning is the key: A crucial ingredient in this work is to first learn the probabilistic model of the input, and employ transformations based on the model for learning the GLM mixture. Thus, we characterize how unsupervised learning on the input can be carried over for learning conditional models of the output via tensor methods.

The feature transformations we employ are the (higher order) score functions,[1] which capture local variation of the probability density function of the input. This follows our recent key result that the cross-moments between the response variable and the input score functions yield (expected) derivatives of the response, as a function of the input (Janzamin et al., 2014).

Incorporating score functions into tensor decomposition framework: In this paper, we exploit the above result to form the expected derivatives of the output as a function of the input. We then show that the expected derivatives have a nice relationship with the unknown parameters of the GLM mixture, and the form reduces to a tensor CP decomposition form. We require only a mild assumption on the activation function that it has non-vanishing third derivative (in expectation). For linear regression, this condition is violated, but we can easily overcome this by considering higher powers of the output in the moment estimation framework.

2 Problem Formulation

Notations: Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Let $e_i \in \mathbb{R}^d$ denote the standard basis vectors in $\mathbb{R}^d$. Let $I_d \in \mathbb{R}^{d \times d}$ denote the identity matrix. Throughout this paper, $\nabla_x^{(m)}$ denotes the m-th order derivative w.r.t. variable x and notation $\otimes$ represents tensor (outer) product.

A real *p-th order tensor* $T \in \bigotimes_{i=1}^{p} \mathbb{R}^{d_i}$ is a member of the tensor product of Euclidean spaces $\mathbb{R}^{d_i}$, $i \in [p]$. As is the case for vectors (where $p = 1$) and matrices (where $p = 2$), we may identify a p-th order tensor with the p-way array of real numbers $[T_{i_1, i_2, \ldots, i_p} : i_1, i_2, \ldots, i_p \in [d]]$, where $T_{i_1, i_2, \ldots, i_p}$ is the $(i_1, i_2, \ldots, i_p)$-th coordinate of T with respect to a canonical basis.

CP decomposition and tensor rank: A 3rd order tensor $T \in \mathbb{R}^{d \times d \times d}$ is said to be rank-1 if it can be written in the form $T = a \otimes b \otimes c \Leftrightarrow T(e_i, e_j, e_l) = a(i) \cdot b(j) \cdot c(l)$, where notation $\otimes$ represents the *tensor product*. A tensor T is said to have a CP rank $k \geq 1$ if it can be written as the sum of k rank-1 tensors $T = \sum_{i \in [k]} a_i \otimes b_i \otimes c_i$.

[1] In this paper, we refer to the derivative of the log of the density function with respect to the variable as the score function. In other works, typically, the derivative is taken with respect to some model parameter (Jaakkola et al., 1999). Note that if the model parameter is a location parameter, the two quantities only differ in the sign. Higher order score functions involve higher order derivatives of the density function. For the exact form, refer to (Janzamin et al., 2014).

2.1 Learning Problem

Let y denote the output and $x \in \mathbb{R}^d$ be the input. We consider both the regression setting, where y can be continuous or discrete, or the classification setting, where y is discrete. For simplicity, we assume y to be a scalar: in the classification setting, this corresponds to binary classification ($y \in \{-1, 1\}$).

We consider the *realizable* setting, where we assume that the output y is drawn from an associative model $p(y|x)$, given input x. In addition, we assume that the input x is drawn from some continuous probability distribution with density function $p(x)$. We will incorporate this generative model in our algorithm for learning the associative model.

We first consider mixtures of generalized linear models (GLM) (Agarwal et al., 2014; Kakade et al., 2011) and then extend to mixture of kernel GLMs. The class of GLMs is given by

$$\mathbb{E}[y|x] = g(\langle u, x\rangle + b), \tag{1}$$

where g is the *activation* function, u is the weight vector, and b is the bias. $g(\cdot)$ is usually chosen to be the logistic function, although we do not impose this limitation. In the binary classification setting, (1) corresponds to a single classifier. Note that a linear regression can be modeled using a linear activation function.

A mixture of r GLM models is then given by employing a hidden choice variable $h \in \{e_1, e_2, \ldots, e_r\}$, where e_i is the basis vector in $\mathbb{R}^r$ to select of the r GLM models, i.e.

$$\mathbb{E}[y|x, h] = g(\langle Uh, x\rangle + \langle \tilde{b}, h\rangle), \tag{2}$$

where $U = [u_1|u_2 \ldots u_r] \in \mathbb{R}^{d_x \times r}$ has the r weight vectors of component GLMs as columns and $\tilde{b} \in \mathbb{R}^r$ is the vector of biases for the component GLMs. Let $w := \mathbb{E}[h]$ be the probability vector for selecting the different GLMs.

We then extend our results to learning mixture of kernel GLMs where

$$\mathbb{E}[y|x, h] = g(\langle Uh, \phi(x)\rangle, \langle \tilde{b}, h\rangle), \tag{3}$$

for some known function $\phi(\cdot)$.

Given training samples $\{x_i, y_i\}$, our goal is to learn the parameters of the associative mixture described above. We consider a moment-based approach, which involves cross-moments of y and a function of x. We first assume that the exact moments are available, and we later carry out sample analysis, when empirical moments are used.

Throughout this paper we make the following assumptions unless otherwise stated. Derivative and expectation are interchangeable. The activation function g is differentiable up to the third order. The choice variable is independent of the input x, i.e., h does not depend on x. The score function $\nabla_x \log p(x)$ exists and all the entries of $g(x) \cdot p(x)$ go to zero on the boundaries of support of $p(x)$.

3 Learning under Gaussian Input

We now present the method for learning the mixture models in (2) and (3). We first start with the simple case, where the input x is Gaussian, and we have a single GLM model, instead of a mixture, and then extend to more general cases.

Algorithm 1 Learning mixture of associative models $\mathbb{E}[y|x,h] = g(\langle Uh, x\rangle + \langle \tilde{b}, h\rangle)$

input Labeled samples $(x_i, y_i), i \in [n]$.

input Score function of the input $\mathcal{S}_3(x)$ as in Equation (9).

1: Compute $\widehat{M}_3 = \frac{1}{n}\sum_i y_i \cdot \mathcal{S}_3(x_i)$, Empirical estimate of M_3.

2: $\{\hat{u}_j\}_{j\in[r]}$ = tensor power decomposition($\widehat{M}_3$). (Algorithm 4 in the Appendix)

3: Recover scale and biases using EM (as in Appendix C).

3.1 Toy Example: single GLM

We first assume a white Gaussian input $x \sim \mathcal{N}(0, I_d)$ to demonstrate our ideas. Assuming that y is generated from a GLM

$$\mathbb{E}[y|x] = g(\langle u, x\rangle + b),$$

we have the following result on the cross-moment $\mathbb{E}[y \cdot x]$.

Lemma 2 (Moment form for Gaussian input and single GLM)**.** *We have*

$$M_1 = \mathbb{E}[y \cdot x] = \mathbb{E}[\nabla_{x'} g(x')] \cdot u,$$

where the expectation is over $x' := \langle u, x\rangle + b$*, and* $x \sim \mathcal{N}(0, I_d)$*.*

Proof follows from Stein's identity (Stein, 1972) as discussed below. Thus, by forming the first-order cross-moment M_1, we can recover the weight vector u up to scaling. Note that the scaling and the bias b are just scalar parameters which can be estimated separately.

The main message behind Lemma 2 is that the cross-moments between the output y and the input x contain valuable information about the associative model. In the special case of Gaussian input and single GLM, the first order moment is sufficient to learn almost all the parameters of the GLM. But how general is this framework? Can we use a moment-based framework when there are mixture of GLMs? We exhibit that higher order moments can be used to learn the GLM mixture under Gaussian input in the next section. What about the case when the input is not Gaussian, but has some general distribution? We consider this setting in Section 4 and show that surprisingly we can form the appropriate cross-moments for learning under any general (continuous) input distribution.

Stein's Identity: The proof of Lemma 2 follows from the Stein's identity for Gaussian distribution. It states that for all functions $G(x)$ satisfying mild regularity conditions, we have (Stein, 1972)

$$\mathbb{E}[g(x) \cdot x] = \mathbb{E}[\nabla_x g(x)]. \tag{4}$$

Thus, Lemma 2 is a direct application of the Stein's identity by substituting $G(x)$ with $g(\langle u, x\rangle + b)$.

3.2 Learning GLM mixtures

We now consider learning mixture of GLMs

$$\mathbb{E}[y|x,h] = g(\langle Uh, x\rangle + \langle \tilde{b}, h\rangle),$$

where $U = [u_1|u_2 \dots u_r]$ has the r weight vectors of component GLMs as columns and $\tilde{b}$ is the vector of biases for the component GLMs. Recall that $w := \mathbb{E}[h]$ is the probability vector for selecting different GLMs.

For the mixture of GLMs, the first order moment $M_1 := \mathbb{E}[y \cdot x]$ is now a combination of (scaled) weight vectors u_i's, i.e.

$$M_1 := \mathbb{E}[y \cdot x] = \sum_{j \in [r]} w_j \mathbb{E}[\nabla_{x'_j} g(x'_j)] u_j,$$

where the expectation is over $x'_j = \langle u_j, x \rangle + \tilde{b}_j$, and $x \sim \mathcal{N}(0, I_d)$. Thus, the first order moment does not suffice for learning mixture of GLMs.

Now, let us look at the second order moment,

$$M_2 := \mathbb{E}[y \cdot (x \otimes x - I)] = \sum_{j \in [r]} \mathbb{E}[\nabla^{(2)}_{x'_j} g(x'_j)] w_j \cdot u_j \otimes u_j,$$

where, as before, the expectation is over $x'_j = \langle u_j, x \rangle + \tilde{b}_j$. If the expectations (and w_j's) are non-zero, then we can recover the subspace spanned by the weight vectors u_j's. However, we cannot recover the individual weight vectors u_j's. Moreover, if the biases $\tilde{b} = 0$ and g is a symmetric function, then the expectations are zero, and the second order moment M_2 vanishes. A mirror trick is introduced in (Sun et al., 2013b) to alleviate this problem, but this still only recovers the subspace spanned by the u_j's.

We now consider the third order moment M_3 in the hope of recovering the weight vectors u_j's for mixture of GLMs. We show that by adjusting the moment $\mathbb{E}[y \cdot x \otimes x \otimes x]$ appropriately, we obtain a CP tensor form in terms of the weight vectors u_j's. Specifically, consider

$$\begin{aligned} M_3 := \mathbb{E}[y \cdot x \otimes x \otimes x] - \sum_{j \in [d]} \mathbb{E}[y \cdot e_j \otimes x \otimes e_j] \\ - \sum_{j \in [d]} \mathbb{E}[y \cdot e_j \otimes e_j \otimes x] - \sum_{j \in [d]} \mathbb{E}[y \cdot x \otimes e_j \otimes e_j]. \end{aligned} \tag{5}$$

Note that M_3 can be considered as a special case of the form $\mathbb{E}[y \cdot \mathcal{S}_3(x)]$ for white Gaussian input $x \sim \mathcal{N}(0, I_d)$, where $\mathcal{S}_3(x)$ is the third order score function of the input as defined in Section 4.1.

Lemma 3 (Adjusted third order moments)**.** *We have*

$$M_3 = \sum_{j \in [r]} \rho_j w_j \cdot u_j \otimes u_j \otimes u_j, \tag{6}$$

where $\rho_j := \mathbb{E}[\nabla^{(3)}_{x'_j} g(x'_j)]$ *and the expectation is over* $x'_j = \langle u_j, x \rangle + \tilde{b}_j$.

The proof follows from Stein's Identity. See Appendix A.1 for details. Having the CP-form allows us to recover the component weight vectors through the tensor decomposition method. We present the result below.

Theorem 4 (Recovery of mixture of GLMs)**.** *Assuming that the weight matrix* $U \in \mathbb{R}^{d \times r}$ *is full column rank,* $\rho_j, w_j \neq 0 \ \ \forall\, j \in [r]$*, given* M_3*, we can recover the component weight vectors* $u_j, j \in [r]$*, up to scaling, using tensor method given in Algorithm 4 (in the Appendix).*

The proof follows from Lemma 3. The computational complexity of tensor decomposition in this factor form is $O(nrdL)$, where n is the number of samples and L is the number of initialization. Having recovered the normalized weight vectors, we can then estimate the scaling and the biases through expectation maximization or other methods. These are just $2r$ additional parameters, and thus, the majority of the parameters are estimated by the tensor method.

Theorem 5 (Sample Complexity). *Assume the conditions for Theorem 4 are met. Suppose the sample complexity*

$$n = \tilde{O}\left(\frac{d^3 r^4}{\epsilon^2 s_{\min}^2(U)}\right),$$

then for each weight vector u_j, the estimate $\hat{u}_j$ from line 2 Algorithm 1 satisfies w.h.p

$$\|u_j - \hat{u}_j\| \leq \tilde{O}(\epsilon), \quad j \in [r].$$

Proof outline: From Lemma 3, we know that the exact cross-moment $\mathbb{E}[y \cdot \mathcal{S}_3(x)]$ has rank-one components as columns of matrix U; see Equation (6) for the tensor decomposition form. Thus given the exact moment, the theorem is proved by applying the tensor decomposition guarantees in Anandkumar et al. (2014c). In the noisy case where the moment is empirically formed by observed samples, we use the analysis and results of tensor power iteration in Anandkumar et al. (2014d). They show that when the perturbation tensor is small, the tensor power iteration initialized by the SVD-based Procedure 3 in the Appendix recovers the rank-1 components up to some small error. The sample complexity is also proved by applying standard matrix concentration inequalities. In particular, we matricize the error tensor between exact moment and the empirical moment, and bound its norm with matrix Bernstein's inequality.

Remark : We can also handle the case when the full column rank assumption on $U \in \mathbb{R}^{d \times r}$ is violated under some additional constraints. In the overcomplete regime, we have the latent dimensionality exceeding the input dimensionality, i.e. $r > d$. The tensor method can still recover the weight vectors u_j, if we assume they are incoherent. A detailed analysis of overcomplete tensor decomposition is given in (Anandkumar et al., 2014d).

Remark : If we assume the u_j are normalized, the above approach suffices to completely learn the parameters w_j. This is because we obtain $w_j \rho_j$ and we have the knowledge of ρ_j, where the activation function and the input distributions are known. Otherwise, we need to perform EM to fully learn the weights. Note that initializing with our method results in performing EM in a low dimension instead of input dimension. The reason is that the only unknown parameters are the scale and biases of the components. We initialize with the output of our method (Algorithm 1) and proceed with EM algorithm as proposed by Xu et al. (1995). For details see Appendix C.

Remark: If $\rho_j = 0$, which is the case for mixture of linear regression, we cannot recover the weight vectors from the tensor given in (5). In this case, we form a slightly different tensor to recover the weight vectors. We elaborate on this in the next section.

Remark: Our results can be easily extended to multi-label and multi-class settings (one-versus-all strategy) as well as vector-valued regression problems.

3.3 Learning Mixtures of Linear Regression

We now consider mixtures of linear regressions:

$$\mathbb{E}[y|x, h = e_j] = \sum_{j \in [r]} w_i \langle u_j, x \rangle + b_j,$$

where $e_j \in \mathbb{R}^r$ denotes the j-th basis vector.

In this case higher order derivatives ($m \geq 2$) of the activation function vanish. Therefore, the cross-moment matrix and tensor defined in Section 3.2 can not yield the parameters. For this setting, we form

$$\begin{aligned} M_2 &:= \mathbb{E}[y^2 \cdot (x \otimes x - I)] \\ M_3 &:= \mathbb{E}[y^3 \cdot x \otimes x \otimes x] - \sum_{j \in [d]} \mathbb{E}[y^3 \cdot e_j \otimes x \otimes e_j] \\ &\quad - \sum_{j \in [d]} \mathbb{E}[y^3 \cdot e_j \otimes e_j \otimes x] - \sum_{j \in [d]} \mathbb{E}[y^3 \cdot x \otimes e_j \otimes e_j]. \end{aligned} \tag{7}$$

Lemma 6 (Adjusted third order moments)**.** *We have*

$$M_3 = \sum_{j \in [r]} \rho_j w_j \cdot u_j \otimes u_j \otimes u_j. \tag{8}$$

The proof follows from Stein's Identity and it is provided in Appendix A.2. Having the CP-form allows us to recover the component weight vectors through the tensor decomposition method. We present the result below.

Theorem 7 (Recovery of linear regression mixtures)**.** *Assuming that the weight matrix $U \in \mathbb{R}^{d \times r}$ is full column rank, $\rho_j, w_j \neq 0 \ \ \forall\, j \in [r]$, given M_3 as in* (7)*, we can recover the component weight vectors $u_j, j \in [r]$, up to scaling, using tensor method given in Algorithm 4 (in the Appendix).*

The proof is similar to Lemma 6.

Theorem 8 (Sample Complexity)**.** *Assume the conditions for Theorem 7 are met. Suppose the sample complexity*

$$n = \tilde{O}\left(\frac{d^3 r^4}{\epsilon^2 s_{\min}^2(U)}\right),$$

then for each weight vector u_j, the estimate $\hat{u}_j$ in line 2 Algorithm 1 satisfies w.h.p

$$\|u_j - \hat{u}_j\| \leq \tilde{O}(\epsilon), \quad j \in [r].$$

The proof follows the same approach as the one described for Theorem 5.

4 Learning GLM Mixtures under General Input Distribution

In the previous section, we established consistent estimation of the parameters of mixture of GLMs under Gaussian input. However, this assumption is limiting, since the input is usually far from Gaussian in any real scenario. We now extend the results in the previous section to any general (continuous) input.

4.1 Extensions of Stein's identity

The key ingredient that enabled learning in the previous section is the ability to compute the expected derivatives of the output as a function of the input. Stein's identity shows that these derivatives can be obtained using the cross-moments between the output and the score function of input. Is there a general unified framework where we can compute the expected derivatives under any general input distribution?

We provide an affirmative answer in a recent work (Janzamin et al., 2014). We show that by computing the cross-moment between the output and the (higher order) score functions of the input, we compute expected derivatives of any order. This key result allows us to extend the results in the previous section to any general input distribution.

Definition: Score function The score of $x \in \mathbb{R}^d$ with pdf $p(x)$, denoted by $\mathcal{S}_1(x)$, is the random vector $\nabla_x \log p(x)$. In (Janzamin et al., 2014), we define the m^{th} order score function as

$$\mathcal{S}_m(x) := (-1)^m \frac{\nabla^{(m)} p(x)}{p(x)}. \tag{9}$$

They have also shown that score function can be equivalently derived using the recursive form

$$\mathcal{S}_m(x) = -\mathcal{S}_{m-1}(x) \otimes \nabla_x \log p(x) - \nabla_x \mathcal{S}_{m-1}(x). \tag{10}$$

Theorem 9 (Higher order derivatives (Janzamin et al., 2014))**.** *For random vector $x \in \mathbb{R}^d$, let $p(x)$ and $\mathcal{S}_m(x)$ respectively denote the pdf and the corresponding m-th order score function. Consider any continuously differentiable output-function $\mathbb{E}[y|x] = g(x) : \mathbb{R}^d \to \mathbb{R}$ satisfying some mild regularity conditions. Then we have*

$$\mathbb{E}\left[y \cdot \mathcal{S}_m(x)\right] = \mathbb{E}\left[g(x) \cdot \mathcal{S}_m(x)\right] = \mathbb{E}\left[\nabla_x^{(m)} g(x)\right].$$

For details, see (Janzamin et al., 2014). In order to learn mixture of GLMs for general input distributions, we utilize score function $\mathcal{S}_m(\cdot)$ of order $m = 3$.

4.2 Moment forms

We now consider the cross-moment $M_3 := \mathbb{E}[y \cdot \mathcal{S}_3(x)]$, which is a third order tensor. By alluding to Theorem 9, we show that the moment M_3 has a CP decomposition where the components are the weight vectors u_j's.

Theorem 10 (Recovery of mixture of GLMs under general input)**.** *Given score function $\mathcal{S}_3(x)$ as in (9), we have*

$$M_3 := \mathbb{E}[y \cdot \mathcal{S}_3(x)] = \sum_{j \in [r]} \rho_j \cdot w_j \cdot u_j^{\otimes 3},$$

where $\rho_j := \mathbb{E}[\nabla^{(3)}_{x'_j} g(x'_j)]$ and the expectation is with respect to $x'_j := \langle u_j, x \rangle + \tilde{b}_j$.

Assuming that matrix U is full column rank and $\rho_j, w_j > 0$, $\forall j$, we can recover the weight vectors $u_j, j \in [r]$, up to scaling, using tensor decomposition on M_3 given in Algorithm 4 (in the Appendix).

The proof follows from Theorem 9. Thus, we have a guaranteed recovery of the weight vectors of mixture of GLMs under any general input distribution.

Remark: Sample complexity: For general input sample complexity can be found in a similar approach to Gaussian case. The general form is $n \geq \tilde{O}\left(\mathbb{E}\left[\|H_3(x)H_3^\top(x)\|\right]\frac{d^{1.5}r^4}{\epsilon^2 s_{\min}^2(U)}\right)$. Here $H_3(x) \in \mathbb{R}^{d\times d^2}$ is the matricization of $\mathcal{S}_3(x)$. Theorem 5 follows from the fact that for Gaussian input $\mathbb{E}\left[\|H_3(x)H_3^\top(x)\|\right] = O(d^{1.5})$.

Remark: Score function estimation: There are various efficient methods for estimating the score function. The framework of score matching is popular for parameter estimation in probabilistic models (Hyvärinen, 2005; Swersky et al., 2011), where the criterion is to fit parameters based on matching the data score function. For instance, Swersky et al. (2011) analyzes fitting the data to RBM (Restricted Boltzmann Machine) model. Therefore, one option is to use this method for estimating $\mathcal{S}_1(x)$ and use the recursive form in (10) to estimate higher order score functions for the active layer.

Remark: Computational Complexity: If we fit the input data into an RBM model, the computational complexity of our method, when performed in parallel, is $O(\log(\min(d, d_h)))$ with $O(rnLd^2d_h/\log(\min(d, d_h)))$ processors. Here d_h is the number of neurons of the first layer of the RBM used for approximating the score function.

4.3 Learning Mixtures of Linear Regression

As discussed earlier, our framework can easily handle the case of mixtures of linear regression. Here, we describe it under general input distribution. Let,

$$\mathbb{E}[y|x, h = e_j] = \sum_{j\in[r]} w_j\langle u_j, x\rangle + b_j,$$

where x, y respectively denote the input and output, and h is the hidden variable that chooses the regression parameter u_j from the set $\{u_j\}_{j\in[r]}$, $w_j = p(h = e_j)$ and b_j is the bias.

Theorem 11 (Recovery of linear regression mixtures under general input)**.** *Given score function $\mathcal{S}_3(x)$ as in Equation* (9)*, we have*

$$M_3 = \mathbb{E}[y^3 \cdot \mathcal{S}_3(x)] = \sum_{j\in[r]} w_j \cdot u_j^{\otimes 3}.$$

Assuming that matrix U is full column rank, $w_j \neq 0, \quad \forall j$, we can recover the weight vectors $u_j, j \in [r]$, up to scaling, using tensor decomposition on M_3 given in Algorithm 4 (in the Appendix).

For proof, see Appendix A.3.

Remark: Sample complexity: For general input sample complexity can be found in a similar approach to Gaussian case. The general form is $n \geq \tilde{O}\left(\mathbb{E}\left[\|H_3(x)H_3^\top(x)\|\right]\frac{d^{1.5}r^4}{\epsilon^2 s_{\min}^2(U)}\right)$. Here $H_3(x) \in \mathbb{R}^{d\times d^2}$ is the matricization of $\mathcal{S}_3(x)$. Theorem 8 follows from the fact that for Gaussian input $\mathbb{E}\left[\|H_3(x)H_3^\top(x)\|\right] = O(d^{1.5})$.

Remark: Chaganty and Liang (2013) consider learning a mixture of linear regression models, using tensor decomposition approach on the higher order moments of of the output y. They model the problem as an optimization on a third order tensor and prove that the optimal tensor would have the weight vectors as its rank-1 components. Minimizing an objective function over a tensor

variable is expensive (in fact, quadratic for each variable (Liu and Vandenberghe, 2009), and their computational complexity scales as $O(nd^{12})$. Hence their proposed method is not practical in large scale. Whereas, as discussed earlier our computational complexity is $O(nd^2)$. While we require the additional knowledge of the input distribution, in many scenarios, this is not a major limitation since there are large amounts of unlabeled samples which can be used for model estimation. Moreover, we can handle non-linear mixtures, while Chaganty and Liang (2013) limit to linear ones.

4.4 Extension to Mixtures of Kernel GLMs

We have so far provided guarantees for learning mixture of kernel GLMs. We now extend the results to cover non-linear models. We consider the class of mixture of kernel GLMs (Yu and Joachims, 2009) under the realizable setting as

$$\mathbb{E}[y|x,h] = g\left(\langle Uh, \phi(x)\rangle + \tilde{b}\right), \tag{11}$$

where $\phi(x)$ represents the nonlinear mapping of x. Assuming that $\phi(\cdot)$ is known, we propose simple ideas to extend our previous results to the setting in (11).

The key idea is to compute the score function $\mathcal{S}_m(\phi(x))$ corresponding to $\phi(x)$ rather than the input x. There is a simple relationship between the scores. The connection can be made from the probability density of the transformed variable as follows. Let $t = \phi(x)$, $D_t(i,j) := \left[\frac{\partial x_i}{\partial t_j}\right]$. We have

$$p_{\phi(x)}(t_1, \cdots, \top_p) = p_x(\phi_1^{-1}(t), \cdots, \phi_p^{-1}(t))|\det(D_t)|, \tag{12}$$
$$\mathcal{S}_m(t) = (-1)^m \frac{\nabla_t^{(m)} p_{\phi(x)}(t)}{p_{\phi(x)}(t)}.$$

Theorem 12 (Recovery of mixture of kernel GLMs under general input)**.** *Given score function $\mathcal{S}_3(\phi(x))$ as in Equation* (12)*, we have*

$$M_3 := \mathbb{E}[y \cdot \mathcal{S}_3(\phi(x))] = \sum_{j\in[r]} \rho_j \cdot w_j \cdot u_j^{\otimes 3},$$

where $\rho_j := \mathbb{E}[\nabla_{z_j}^{(3)} g(z_j)]$ and the expectation is with respect to $z_j := \langle u_j, \phi(x)\rangle + \tilde{b}_j$.

Assuming that matrix U is full column rank, $w_j, \rho_j \neq 0, \ \ \forall j$, we can recover the weight vectors $u_j, j \in [r]$, up to scaling, using tensor decomposition on M_3 given in Algorithm 4 (in the Appendix).

We therefore have a guaranteed recovery of the parameters of mixture of kernel GLMs under the realizable setting, given score function $\mathcal{S}_3(\phi(x))$.

5 Related Works

Mixture of Experts/ Regression Mixtures: The mixture of experts model was introduced as an efficient probabilistic "divide" and "conquer" paradigm in (Jordan and Jacobs, 1994). Since then, it has been considered in a number of works, e.g. (Xu et al., 1995; Bishop and Svensen, 2003). Learning is carried out usually through EM (Jordan and Jacobs, 1994; Xu et al., 1995) or variational approaches (Bishop and Svensen, 2003), but the methods have no guarantees. Works with

guaranteed learning of associative mixture models are fewer. Chaganty and Liang (2013) consider learning a mixture of linear regression models, using tensor decomposition approach on the higher order moments of of the label y. Yi et al. (2013) also consider mixed linear regression problem with two components and provide consistency guarantees in the noiseless setting for an alternating minimization method. Chen et al. (2014) provide an alternative convex method for the same setting under noise and established near optimal sample complexity. However, all these guaranteed methods are restricted to mixture of linear regressions and do not extend to non-linear models.

Learning mixture of GLMs: For the mixture of generalized linear models (GLM), Li (1992) and Sun et al. (2013a) present methods for learning the subspace of the weight vectors of the component GLMs, assuming that the input is white Gaussian distribution. Li (1992) propose the so-called principal Hessian directions (PHd), where the eigenvectors of the second-order moment matrix $\mathbb{E}[y \cdot x \otimes x]$ are used to learn the desired subspace (the notation $\otimes$ represents tensor (outer) product). However, the PHd method fails when the output y is a symmetric function of the input x, since the moment matrix vanishes in this case. Sun et al. (2013a) overcome this drawback through their clever "mirroring" trick which transforms the output y to $r(y)$ such that the resulting second order moment $\mathbb{E}[r(y) \cdot x \otimes x]$ matrix does not vanish.

Our work has some key differences: the works in (Li, 1992; Sun et al., 2013a) assume Gaussian input x, while we allow for any probabilistic model (with continuous density function). Another important difference between (Li, 1992; Sun et al., 2013a) and our work, is that we use tensor-based learning techniques, while (Li, 1992; Sun et al., 2013a) only operate on matrices. Operating on tensors allows us to learn the individual weight vectors (up to scaling) of the mixture components, while (Li, 1992; Sun et al., 2013a) only learn the subspace of the weight vectors.

Spectral/Moment based methods for discriminative learning: Karampatziakis and Mineiro (2014) obtain discriminative features via generalized eigenvectors. They consider the tensor $\mathbb{E}[y \otimes x \otimes x]$ and then treat $\mathbb{E}[x \otimes x|y = i]$ as the signal for class i and $\mathbb{E}[x \otimes x|y = j]$ as the noise due to class j. They contrast their method against classical discriminative procedures such as Fisher LDA and show good performance on many real datasets. However, their method has some drawbacks: they cannot handle continuous y, and also when y has a large number of classes m and $x \in \mathbb{R}^d$ has high dimensionality, the method is not scalable since it requires m^2 eigen-decompositions of $d \times d$ matrices. Another line of moment based methods are the so-called *sliced inverse regression* (SIR) (Li, 1991), where input x is regressed against output y. These methods project the input to a lower dimension subspace that preserves the required information. Li (1991) consider top eigen components of the moment $\mathbb{E}[\mathbb{E}[x|y]\mathbb{E}[x|y]^\top]$ for dimensionality reduction.

6 Conclusion

In this paper, we propose a tensor method for efficient learning of associative mixtures. In addition to employing the learnt weight vectors in the mixture of GLMs model for prediction, we can employ them in a number of alternative ways in practice. For instance, we can utilize the output of the tensor-based methods as initializers for likelihood based techniques such as expectation maximization. Since these objective functions are non-convex, in general, they can get stuck in bad local optima. Initializing with the tensor methods can lead to convergence to better local optima. Moreover, we can employ the learnt weight vectors to construct discriminative features and train a different classifier using them. Thus, our method yields discriminative information which is useful in myriad ways.

There are many future directions to consider. We assume that the choice variable for selecting the mixture components is independent of the input. This is also the assumption in a number of other works for learning regression/classifier mixtures (Sun et al., 2013a; Chaganty and Liang, 2013). In the general mixture of experts framework, the choice variable is known as the gating variable, and it selects the classifier based on the input. Considering this scenario is of interest. Moreover, we have considered continuous input distributions, extending this framework to discrete input is of interest.

Acknowledgment

The authors thank Majid Janzamin for detailed discussions on sample complexity and constructive comments on the draft.

A. Anandkumar is supported in part by Microsoft Faculty Fellowship, NSF Career award CCF-1254106, NSF Award CCF-1219234, ARO YIP Award W911NF-13-1-0084 and ONR Award N00014-14-1-0665. H. Sedghi is supported by ONR Award N00014-14-1-0665.

A Proofs

A.1 Proof of Lemma 3

Notation: Tensor as multilinear forms: We view a tensor $T \in \mathbb{R}^{d\times d\times d}$ as a multilinear form. Consider matrices $M_r \in \mathbb{R}^{d\times d_r}, r \in \{1,2,3\}$. Then tensor $T(M_1, M_2, M_3) \in \mathbb{R}^{d_1} \otimes \mathbb{R}^{d_2} \otimes \mathbb{R}^{d_3}$ is defined as

$$T(M_1, M_2, M_3)_{i_1,i_2,i_3} := \sum_{j_1,j_2,j_3\in[d]} T_{j_1,j_2,j_3} \cdot M_1(j_1, i_1) \cdot M_2(j_2, i_2) \cdot M_3(j_3, i_3). \tag{13}$$

In particular, for vectors $u, v, w \in \mathbb{R}^d$, we have [2]

$$T(I, v, w) = \sum_{j,l\in[d]} v_j w_l T(:, j, l) \ \in \mathbb{R}^d, \tag{14}$$

which is a multilinear combination of the tensor mode-1 fibers. Similarly $T(u, v, w) \in \mathbb{R}$ is a multilinear combination of the tensor entries, and $T(I, I, w) \in \mathbb{R}^{d\times d}$ is a linear combination of the tensor slices.

Now, let us proceed with the proof.

Proof: Let $x' := \langle u, x\rangle + b$. Define $l(x) := y \cdot x \otimes x$. We have

$$\mathbb{E}[y \cdot x^{\otimes 3}] = \mathbb{E}[l(x) \otimes x] = \mathbb{E}[\nabla_x l(x)],$$

by applying Stein's lemma. We now simplify the gradient of $l(x)$.

$$\begin{aligned} \nabla_x l(x) &= \nabla_x (g(\langle u, x\rangle) \cdot x \otimes x), \\ \mathbb{E}\{\nabla_x l(x)\} &= \mathbb{E}[y \cdot \nabla_x (x \otimes x)] + \mathbb{E}[(\nabla_{x'} g(x'))(x \otimes x \otimes u)]. \end{aligned} \tag{15}$$

[2] Compare with the matrix case where for $M \in \mathbb{R}^{d\times d}$, we have $M(I, u) = Mu := \sum_{j\in[d]} u_j M(:, j) \in \mathbb{R}^d$.

We now analyze the first term. We have

$$\nabla_x(x\otimes x)_{i_1,i_2,j} = \frac{\partial x_{i_1}x_{i_2}}{\partial x_j} = \begin{cases} x_{i_2}, & i_1 = j, \\ x_{i_1}, & i_2 = j, \\ 2x_j, & i_1 = i_2 = j, \\ 0, & \text{o.w.} \end{cases} \tag{16}$$

This can be written succinctly as

$$\nabla_x(x\otimes x) = \sum_i e_i\otimes x\otimes e_i + \sum_i x\otimes e_i\otimes e_i + \sum_i 2x_i(e_i\otimes e_i\otimes e_i)$$

and therefore, the expectation for the first term in (15) is given by

$$\mathbb{E}[y\cdot\nabla_x(x\otimes x)] = \sum_i \left(\mathbb{E}[y\cdot e_i\otimes x\otimes e_i] + \mathbb{E}[y\cdot x\otimes e_i\otimes e_i] + 2\mathbb{E}[y\cdot x_i\cdot e_i\otimes e_i\otimes e_i]\right).$$

Now for the second term in (15), let $f(x) := \nabla_{x'}g(x')\cdot x\otimes u$. The transposition of the second term in (15) is given by

$$\begin{aligned}\mathbb{E}[\left(\nabla_{x'}g(x')\cdot x\otimes u\right)\otimes x] &= \mathbb{E}[f(x)\otimes x] \\ &= \mathbb{E}[\nabla_x f(x)],\end{aligned}$$

where we have swapped modes 2 and 3 in $\mathbb{E}[(\nabla_{x'}g(x'))(x\otimes x\otimes u)]$ to obtain the above. We will compute $\nabla_x f(x)$ and then switch the tensor modes again to obtain the final result. We have

$$\begin{aligned}\nabla_x f(x) &= \nabla_x\left(\nabla_{x'}g(x')x\otimes u\right) \\ &= (\nabla^{(2)}_{x'}g(x'))\cdot x\otimes u\otimes u + (\nabla_{x'}g(x'))\cdot\nabla_x(x\otimes u),\end{aligned} \tag{17}$$

and thus, $\nabla_x(x\otimes u) = \sum_i(e_i\otimes u\otimes e_i)$. The first term is given by

$$\mathbb{E}\left[(\nabla^{(2)}_{x'}g(x'))\cdot x\otimes u\otimes u\right] = \mathbb{E}\left[(\nabla^{(3)}_{x'}g(x'))\cdot u\otimes u\otimes u\right]$$

So the second term in (17) is given by

$$\sum_i(\nabla_{x'}g(x'))\cdot(e_i\otimes u\otimes e_i).$$

Note that

$$\mathbb{E}\left[(\nabla_{x'}g(x'))\cdot(e_i\otimes u\otimes e_i)\right] = \mathbb{E}\left[e_i\otimes\nabla_x g(\langle x,u\rangle)\otimes e_i\right] = \mathbb{E}\left[g(x')\cdot(e_i\otimes x\otimes e_i)\right],$$

since if we apply Stein's left to right-hand side, we obtain the left hand side of the equation. Swapping the modes 2 and 3 above, we obtain the result by substituting in (15). □

A.2 Proof of Lemma 6

By replacing y by y^3 in Proof of Lemma 3 (Appendix A.1), we have that

$$\begin{aligned}M_3 &= \mathbb{E}_x\left[\nabla^3_x(y^3)\right] = \mathbb{E}_x\left[\nabla^3_x\mathbb{E}_h\left[y^3|h=e_j\right]\right] \\ &= \mathbb{E}_x\left[\nabla^3_x\left(\sum_{j\in[r]} w_j\langle u_j,x\rangle^3\right)\right] = \sum_{j\in[r]}\rho_j w_j\cdot u_j\otimes u_j\otimes u_j.\end{aligned}$$

Note that the third equation results from the fact that for each sample only one of the $u_j, j\in[r]$ is chosen by h and no other terms are present. Therefore,the expression has no cross terms.

A.3 Proof of Theorem 11

Proof.

$$\begin{aligned} M_3 &= \mathbb{E}_x[y^3 \cdot \mathcal{S}_3(x)] = \mathbb{E}_h\left[\mathbb{E}_x[y^3 \cdot \mathcal{S}_3(x)|h = e_j]\right] \\ &= \mathbb{E}_x\left[\mathbb{E}_h[y^3 \cdot \mathcal{S}_3(x)|h = e_j]\right] = \mathbb{E}_x\left[\sum_{j\in[r]} w_j\langle u_j, x\rangle^3\right] \\ &= \mathbb{E}_x\left[\nabla_x^3\left[\sum_{j\in[r]} w_i\langle u_j, x\rangle^3\right]\right] = \sum_{j\in[r]} w_j \cdot u_j^{\otimes 3}. \end{aligned}$$

Note that the fourth equation results from the fact that for each sample only one of the $u_j, j \in [r]$ is chosen by h and no other terms are present. Therefore,the expression has no cross terms. ■

B Tensor Decomposition Method

We now recap the tensor decomposition method Anandkumar et al. (2014d) to obtain the rank-1 components of a given tensor. This is given in Algorithm 4. Let $\widehat{M}_3$ denote the empirical moment tensor input to the algorithm.

Since in our case modes are the same, the asymmetric power updates in (Anandkumar et al., 2014d) are simplified to one update. These can be considered as rank-1 form of the standard alternating least squares (ALS) method. If we assume the weight matrix U (i.e. the tensor components) has incoherent columns, then we can directly perform tensor power method on the input tensor $\widehat{M}_3$ to find the components. Otherwise, we need to whiten the tensor first. We take a random slice of the empirical estimate of $\widehat{M}_3$ and use it to find the whitening matrix[3]. Let $\widehat{V}$ be the average of the random slices. The whitening matrix $\widehat{W}$ can be found by using a rank-r SVD on $\widehat{V}$ as shown in Procedure 2.

Since the tensor decomposition problem is non-convex, it requires good initialization. We use the initialization algorithm from (Anandkumar et al., 2014d) as shown in Procedure 3. The initialization for different runs of tensor power iteration is performed by the SVD-based technique proposed in Procedure 3. This helps to initialize non-convex power iteration with good initialization vectors when we have large enough number of initializations. Then, the clustering algorithm is applied where its purpose is to identify which initializations are successful in recovering the true rank-1 components of the tensor.

C Expectation Maximization for Learning Un-normalized Weights

If we assume the weight vectors are normalized, our proposed algorithm suffices to completely learn the parameters w_i. Otherwise, we need to perform EM to fully learn the weights. Note that initializing with our method results in performing EM in a lower dimension than the input dimension. In addition, we can also remove the independence of selection parameter from input features when doing EM. We initialize with the output of our method (Algorithm 1) and proceed

[3]If $\mathbb{E}[y|x]$ is a symmetric function of x, then the second moment M_2 is zero. Therefore, we cannot use it for whitening. Instead, we use random slices of the third moment M_3 for whitening.

Procedure 2 Whitening

input Tensor $T \in \mathbb{R}^{d\times d\times d}$.

1: Draw a random standard Gaussian vector $\theta \sim \mathcal{N}(0, I_d)$.
2: Compute $\widehat{V} = T(I, I, \theta) \in \mathbb{R}^{d\times d}$.
3: Compute the rank-r SVD $\widehat{V} = \tilde{U}\,\mathrm{Diag}(\tilde{\lambda})\tilde{U}^\top$.
4: Compute the whitening matrix $\widehat{W} = \tilde{U}\,\mathrm{Diag}(\tilde{\lambda}^{-1/2})$.
5: **return** $T\left(\widehat{W}, \widehat{W}, \widehat{W}\right)$.

Procedure 3 SVD-based initialization when $r = O(d)$ (Anandkumar et al., 2014d)

input Tensor $T \in \mathbb{R}^{r\times r\times r}$.

1: Draw a random standard Gaussian vector $\theta \sim \mathcal{N}(0, I_r)$.
2: Compute u_1 as the top left and right singular vector of $T(I, I, \theta) \in \mathbb{R}^{r\times r}$.
3: $\hat{a}_0 \leftarrow u_1$.
4: **return** $\hat{a}_0$.

with EM algorithm as proposed by Xu et al. (1995), Section 3. Below we repeat the procedure in our notation for completeness.

Consider the gating network

$$g_j(x,\nu) = \frac{w_j p(x|\nu_j)}{\sum_i w_i p(x|\nu_i)}, \quad \sum_i w_i = 1, \quad w_i \geq 0,$$
$$p(x|\nu_j) = a_j(\nu_j)^{-1} b_j(x) \exp\{c_j(\nu_j)^\top t_j(x)\},$$

where $\nu = \{w_j, \nu_j, j = 1, \cdots, r\}$, and the $p(x|\nu_j)$'s are density functions from the exponential family.

In the above equation, $g_j(x,\nu)$ is actually the posterior probability $p(j|x)$ that x is assigned to the partition corresponding to the $j-$th expert net. From Bayes' rule:

$$g_j(x,\nu) = p(j|x) = \frac{w_j p(x|\nu_j)}{p(x,\nu)}, \quad p(x,\nu) = \sum_i w_i p(x|\nu_i).$$

Hence,

$$p(y|x,\Theta) = \sum_j \frac{w_j p(x|\nu_j)}{p(x,\nu)} p(y|x,u_j),$$

where Θ includes $u_j, j = 1, \cdots, r$ and ν. Let

$$Q^g(\nu) = \sum_t \sum_j f_j^{(k)}(y^{(t)}|x^{(t)}) \ln g_j^{(k)}(x^{(t)}, \nu^{(t)}),$$
$$Q_j^g(\nu_j) = \sum_t f_j^{(k)}(y^{(t)}|x^{(t)}) \ln p(x^{(t)}|\nu_j), \quad j \in [r]$$
$$Q_j^e(\theta_j) = \sum_t f_j^{(k)}(y^{(t)}|x^{(t)}) \ln p(y^{(t)}|x^{(t)}, \theta_j), \quad j \in [r]$$
$$Q^w = \sum_t \sum_j f_j^{(k)}(y^{(t)}|x^{(t)}) \ln w_j, \quad \text{with} \;\; w = \{w_1, \ldots, w_r\}$$

Algorithm 4 Robust tensor power method (Anandkumar et al., 2014d)

input symmetric tensor $T \in \mathbb{R}^{d \times d \times d}$, number of iterations N, number of initializations L, parameter ν.

output the estimated eigenvector/eigenvalue pair.

Whiten T using the whitening method n Procedure 2.
for $\tau = 1$ to L **do**
  Initialize $\hat{a}_0^{(\tau)}$ with SVD-based method in Procedure 3.
  for $t = 1$ to N **do**
    Compute power iteration update

$$\hat{a}_t^{(\tau)} := \frac{T(I, \hat{a}_{t-1}^{(\tau)}, \hat{a}_{t-1}^{(\tau)})}{\|T(I, \hat{a}_{t-1}^{(\tau)}, \hat{a}_{t-1}^{(\tau)})\|} \tag{18}$$

  end for
end for
$S := \left\{ a_\tau^{(N+1)} : \tau \in [L] \right\}$
while S is not empty **do**
  Choose $a \in S$ which maximizes $|T(a, a, a)|$.
  Do N more iterations of (18) starting from a.
  Output the result of iterations denoted by $\hat{a}$.
  Remove all the $a \in S$ with $|\langle a, \hat{a} \rangle| > \nu/2$.
end while

The EM algorithm is as follows:

1. E-step. Compute

$$f_j^{(k)}(y^{(t)}|x^{(t)}) = \frac{w_j^{(k)} p(x^{(t)}|\nu_j^{(k)}) p(y^{(t)}|x^{(t)}, u_j^{(k)})}{\sum_i w_i^{(k)} p(x^{(t)}|\nu_i^{(k)}) p(y^{(t)}|x^{(t)}, u_i^{(k)})}.$$

2. M-Step Find a new estimate for $j = 1, \cdots, r$

$$u_j^{(k+1)} = \arg\max_{u_j} Q_j^e(u_j), \quad \nu_j^{(k+1)} = \arg\max_{\nu_j} Q_j^g(\nu_j),$$
$$w^{(k+1)} = \arg\max_{w} Q^w, \quad \text{s.t.} \quad \sum_i w_i = 1.$$

References

Alekh Agarwal, Sham M Kakade, Nikos Karampatziakis, Le Song, and Gregory Valiant. Least squares revisited: Scalable approaches for multi-class prediction. In *Proc. of ICML*, 2014.

A. Anandkumar, R. Ge, D. Hsu, and S. M. Kakade. A Tensor Spectral Approach to Learning Mixed Membership Community Models. In *Conference on Learning Theory (COLT)*, June 2013.

A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *J. of Machine Learning Research*, 15:2773–2832, 2014a.

Anima Anandkumar, Rong Ge, and Majid Janzamin. Sample Complexity Analysis for Learning Overcomplete Latent Variable Models through Tensor Methods. *arXiv preprint arXiv:1408.0553*, Aug. 2014b.

Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15: 2773–2832, 2014c.

Animashree Anandkumar, Rong Ge, and Majid Janzamin. Guaranteed non-orthogonal tensor decomposition via alternating rank-1 updates. *arXiv preprint arXiv:1402.5180*, 2014d.

Pranjal Awasthi, Avrim Blum, Or Sheffet, and Aravindan Vijayaraghavan. Learning mixtures of ranking models. In *Proc. of NIPS*, 2014.

Christopher M Bishop and Markus Svensen. Bayesian hierarchical mixtures of experts. In *Proc. of Uncertainty in Artificial Intelligence*, 2003.

Arun Chaganty and Percy Liang. Spectral experts for estimating mixtures of linear regressions. In *Proc. of The 30th International Conference on Machine Learning*, 2013.

Yudong Chen, Xinyang Yi, and Constantine Caramanis. A convex formulation for mixed regression with two components: Minimax optimal rates. In *Conf. on Learning Theory*, 2014.

F. Huang, U.N. Niranjan, M. Hakeem, and A. Anandkumar. Online Tensor Methods for Learning Latent Variable Models. *Accepted to JMLR*, 2014.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. In *Journal of Machine Learning Research*, pages 695–709, 2005.

Tommi Jaakkola, David Haussler, et al. Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems*, pages 487–493, 1999.

Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Score Function Features for Discriminative Learning: Matrix and Tensor Frameworks. *arXiv preprint arXiv:1412.2863*, Dec. 2014.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Sham M Kakade, Adam Kalai, Varun Kanade, and Ohad Shamir. Efficient learning of generalized linear and single index models with isotonic regression. In *NIPS*, pages 927–935, 2011.

Nikos Karampatziakis and Paul Mineiro. Discriminative features via generalized eigenvectors. In *Proceedings of The 31st International Conference on Machine Learning*, pages 494–502, 2014.

Ker-Chau Li. Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86(414):316–327, 1991.

Ker-Chau Li. On principal hessian directions for data visualization and dimension reduction: another application of stein's lemma. *Journal of the American Statistical Association*, 87(420): 1025–1039, 1992.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 761–768. Association for Computational Linguistics, 2006.

Zhang Liu and Lieven Vandenberghe. Interior-point method for nuclear norm approximation with application to system identification. *SIAM Journal on Matrix Analysis and Applications*, 31(3): 1235–1256, 2009.

Seewong Oh and Devavrat Shah. Learning mixed multinomial logit model from ordinal data. In *Proc. of NIPS*, 2014.

K. Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society, London, A.*, page 71, 1894.

Slav Petrov and Dan Klein. Discriminative log-linear grammars with latent variables. In *Advances in Neural Information Processing Systems*, pages 1153–1160, 2007.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In *Advances in neural information processing systems*, pages 1097–1104, 2004.

Charles Stein. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*, pages 583–602, Berkeley, Calif., 1972. University of California Press.

Yuekai Sun, Stratis Ioannidis, and Andrea Montanari. Learning mixtures of linear classifiers. *Arxiv 1311.2547*, 2013a.

Yuekai Sun, Stratis Ioannidis, and Andrea Montanari. Learning mixtures of linear classifiers. *arXiv preprint arXiv:1311.2547*, 2013b.

Kevin Swersky, David Buchman, Nando D Freitas, Benjamin M Marlin, et al. On autoencoders and score matching for energy based models. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1201–1208, 2011.

Yang Wang and Greg Mori. Max-margin hidden conditional random fields for human action recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 872–879. IEEE, 2009.

Lei Xu, Michael I Jordan, and Geoffrey E Hinton. An alternative model for mixtures of experts. *Advances in Neural Information Processing Systems*, pages 633–640, 1995.

Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. Alternating minimization for mixed linear regression. *arXiv preprint arXiv:1310.3745*, 2013.

Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1169–1176. ACM, 2009.