Provable Tensor Methods for Learning Mixtures of Classifiers

Hanie Sedghi∗ Anima Anandkumar†

March 3, 2015

Abstract

We consider the problem of learning associative mixtures for classification and regression problems, where the output is modeled as a mixture of conditional distributions, conditioned on the input. In contrast to approaches such as expectation maximization (EM) or variational Bayes, which can get stuck in bad local optima, we present a tensor decomposition method which is guaranteed to correctly recover the parameters. The key insight is to learn score function features of the input, and employ them in a moment-based approach for learning associative mixtures. Specifically, we construct the cross-moment tensor between the label and higher order score functions of the input. We establish that the decomposition of this tensor consistently recovers the components of the associative mixture under some simple non-degeneracy assumptions. Thus, we establish that feature learning is the critical ingredient for consistent estimation of associative mixtures using tensor decomposition approaches.

Keywords: Associative mixtures, mixture of classifiers, feature learning, score function, spectral/tensor decomposition.

1 Introduction

An associative mixture model consists of a mixture of conditional probability distributions, where the choice variable is latent or hidden. They are employed in classification and regression settings, where the output is modeled as a mixture of conditional distributions, conditioned on the input. Associative mixtures combine the expressive power of latent variables with the predictive capabilities of regression/classification frameworks. They have been widely employed in a number of tasks, 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, associative mixtures 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 associative mixtures.

∗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., 2013).

In this paper, we demonstrate how the tensor methods can be effectively employed for learning associative mixtures, which are discriminative latent variable models. Several important challenges arise here: in the earlier works on tensor methods (Anandkumar et al., 2014a), a key assumption is that the observables are linear functions of the latent variables (in expectation). However, for classification problems, non-linear models are better suited, and this rules out a direct application of tensor methods. Moreover, in the earlier works (Anandkumar et al., 2014a), we employ higher order moments of the observables for learning. In the classification setting, higher order moments of the label do not contain any additional information, and is not useful for learning. Thus, the problem of learning associative mixtures is significantly different from learning linear latent variable models, considered in the earlier works.

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

**Representation learning is the key:** A crucial ingredient is to first learn the probabilistic model of the input, and employ features based on the model for learning the associative mixture model for the label. Employing representation or feature learning for accurate classification underlies many popular frameworks such as deep learning (Bengio et al., 2013) and Fisher kernels (Jaakkola et al., 1999). Such approaches have shown superior performance compared to purely discriminative techniques such as support vector machines (SVM) in many settings. In this paper, we establish that feature learning can also be incorporated into tensor decomposition methods for consistent estimation of discriminative models.

The features we employ are the (higher order) score functions\(^1\), which capture local variation of the probability density function of the input. We recently establish a key result that the cross-moments between the label and the score functions yield (expected) derivatives of the label, as a function of the input (Janzamin et al., 2014).

**Incorporating score functions into tensor decomposition framework:** In this paper, we exploit the above important result to form the expected derivatives of the label as a function

---

\(^1\)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).
of the input. We then show that the expected derivatives have a nice relationship with the unknown parameters of the associative mixture model. When the associative mixture is a mixture of generalized linear models (GLM), i.e. \( \mathbb{E}[y|h,x] = g(Uh + \langle \tilde{b}, h \rangle) \), where \( g(\cdot) \) is some activation function, \( h \) is the choice variable, and \( U = [u_1 | u_2 | \ldots | u_r] \) is the matrix of the weights for GLM components, we establish that the expected derivative tensor (of order at least three) has a rank-1 component along each component weight vector \( u_i \). 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. Moreover, this framework can be easily extended to learning mixture of non-linear classifiers, where \( \mathbb{E}[y|h,x] = g(Uh + \phi(x) + \langle \tilde{b}, h \rangle) \), for any function \( \phi(\cdot) \), by estimating the score function of \( \phi(x) \), instead of \( x \). Thus, we guarantee consistent estimation of the weight vectors of associative mixtures through decomposition of the cross-moment tensor involving the label and the input score functions. Note that this is the realizable setting for the latent kernel SVM (Yu and Joachims, 2009; Chang et al., 2010).

**Sample complexity bounds:** We also establish that the method has efficient sample complexity, assuming that the score functions (up to third order) are bounded and have bounded variance. We further provide explicit results in the special case of Gaussian input \( x \sim \mathcal{N}(0, \sigma^2) \) and incoherent weight matrix \( U \). The results depend on the variance of the input. To contrast the sample bounds, we characterize them in low and high variance regimes, where the variance refers to the variance of the Gaussian input. Low variance regime refers to the case where \( \sigma^2 = O(1) \) and high variance regime represents the case where \( \sigma^2 = O(d_x) \), \( d_x \) is the dimension of the input, and \( r \) is the number of classifiers in the mixture. We show that for correct recovery, we need the number of samples to be \( \Omega(r \sqrt{d_x}) \) and \( \tilde{\Omega}(r^2 d_x^2) \) in low-variance regime and in high-variance regime respectively. Therefore, our method has a very efficient sample complexity. In the scenario, where the weight matrix is not incoherent \( U \), the sample complexity degrades since we need to orthogonalize the tensor, before running the spectral decomposition method. We compute a whitening matrix using a random slice of the tensor. In this case, under Gaussian input, assuming that the \( r^{th} \) singular value of the weight matrix \( U \) is bounded, for correct recovery, we need the number of samples to be \( \Omega(r d_x^{3.5}) \) and \( \Omega(r^2 d_x^2) \) in low-noise regime and in high-noise regime respectively. Thus, we present a guaranteed method for learning associative mixtures with low computational and sample complexities.

**Employing the results from the tensor method:** In addition to employing the learnt weight vectors in the mixture of classifiers 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 or discriminative approaches such as latent SVMs (Yu and Joachims, 2009; Chang et al., 2010). 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 use them in alternative classification frameworks such as kernel SVMs. For instance, for each learnt weight vector \( u_i \), we can construct features \( \beta(\langle u_i, x \rangle) \), for any non-linear function \( \beta \), and such approaches have been used previously to construct discriminative features, e.g. (Karampatziakis and Mineiro, 2014).

In a nutshell, the main contribution of this work is to provide a guaranteed method for learning
associative mixture of classifiers and regression problems. The key insight is to learn score function features of the input, and employ them in a moment-based approach for learning associative mixtures. Specifically, we construct the cross-moment tensor between the label and higher order score functions of the input. We establish that the decomposition of this tensor consistently recovers the components of the associative mixture under some simple non-degeneracy assumptions.

1.1 Related Work

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. T. Chaganty and Liang (2013) consider learning a mixture of linear regression models, using tensor decomposition approach on the higher order moments 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 \( E[y \cdot x \otimes x] \) are used to learn the desired subspace (the notation \( \otimes \) represents tensor (outer) product). However, the PHd method fails when the label \( y \) 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 label \( y \) to \( r(y) \) such that the resulting second order moment \( E[r(y) \cdot x \otimes x] \) matrix does not vanish.

The techniques in (Li, 1992; Sun et al., 2013a) provided the inspiration to start our line of investigation. However, there are 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). The key insight is the extension of Stein’s identity to higher order differential operators for general distributions, carried out in our recent work (Janzamin et al., 2014). This not only allows us to handle any general input distribution, but also brings forth the observation that the key features to be used are the score functions. 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: We employ a moment based approach which uses a tensor involving the label \( y \) and score functions of the input, \( S_m(x) \),
for $m \leq 3$. For the special case when the input distribution is Gaussian, the moment tensor is $E[y \cdot x \otimes x \otimes x]$, due to the specific form of the score function for the Gaussian distribution. Many earlier works have attempted to use such moments for discriminative learning. Karampatziakis and Mineiro (2014) obtain discriminative features via generalized eigenvectors. They consider the tensor $E[y \otimes x \otimes x]$ and then treat $E[x \otimes x | x = i]$ as the signal for class $i$ and $E[x \otimes x | y = j]$ 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 label $y$. These methods project the input to a lower dimension subspace that preserves the required information. Li (1991) consider the moment $E[E[x | y]E[x | y]^\top]$ and compute its top eigen components to do dimensionality reduction (assuming whitened elliptical $x$). The method is referred as sliced inverse regression, since $y$ is binned into slices and conditional covariance is computed over the slices.

Generalized Linear Models (without latent variables): Kakade et al. (2011) and Agarwal et al. (2014) consider learning GLMs using a convex surrogate loss function. They establish that when the activation function is Lipschitz and strongly monotone, they can provably learn the weight vector with convergence rate guarantees. In this paper, we consider a mixture of GLMs instead of a single GLM. Moreover, we do not impose monotonicity assumptions on the activation function. Instead, we require non-vanishing derivatives of the activation function up to third order. The techniques are quite different since we utilize non-convex tensor based methods in contrast to convex techniques (when the activation function is known) in (Kakade et al., 2011; Agarwal et al., 2014).

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 $T = a \otimes b \otimes c \iff T(e_i, e_j, e_l) = a(i) \cdot b(j) \cdot c(l)$, 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 $T = \sum_{i \in [k]} a_i \otimes b_i \otimes c_i$. 

5
2.1 Learning Problem

Let \( y \) denote the label 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 label \( y \) 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 non-linear mixtures. The class of GLMs is given by

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

(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 linear classifier.

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.

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

(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 non-linear models where

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

(3)

for some known function \( \phi(\cdot) \). Note that the above corresponds to latent SVM (Yu and Joachims, 2009) under the realizable setting.

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 representations 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 link function \( g \) is differentiable up to the third order and has non-vanishing derivative 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 Results

We now present methods 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.
3.1 Toy Example: Gaussian input and single GLM

We first assume a white Gaussian input $x \sim \mathcal{N}(0, I_{d_x})$ 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 1** (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 $x \sim \mathcal{N}(0, I_{d_x})$.

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 1 is that the cross-moments between the label $y$ 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 is 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 1 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)

$$\mathbb{E}[g(x) \cdot x] = \mathbb{E}[\nabla_xg(x)].$$

Thus, Lemma 1 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 under Gaussian input

We now consider learning mixture of GLMs

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

where $U = [u_1 | u_2 \ldots u_r]$ 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 the 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.

$$M_1 := \mathbb{E}[y \cdot x] = \sum_{i \in [r]} w_i \mathbb{E}[\nabla_{x'}g(x'_i)]u_i,$$

where the expectation is over $x'_i := \langle u_i, x \rangle + \tilde{b}_i$, and $x \sim \mathcal{N}(0, I_{d_x})$. Thus, the first order moment does not suffice for learning mixture of GLMs.
Now, let us look at the second order moment,

\[ M_2 := \mathbb{E}[y \cdot x \otimes x] = \sum_{i \in [r]} \mathbb{E}[\nabla^{(2)}_{x'_i} g(x'_i)] w_i \cdot u_i \otimes u_i, \]

where, as before, the expectation is over \( x'_i = \langle u_i, x \rangle + \tilde{b}_i \). If the expectations (and \( w_i \)'s) are non-zero, then we can recover the subspace spanned by the weight vectors \( u_i \)'s. However, we cannot recover the individual weight vectors \( u_i \)'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 clever mirror trick is introduced in (Sun et al., 2013b) to alleviate this problem, but this still only recovers the subspace spanned by the \( u_i \)'s.

We now consider the third order moment \( M_3 \) in the hope of recovering the weight vectors \( u_i \)'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 \( u_i \)'s. Specifically, consider

\[ M_3 := \mathbb{E}[y \cdot x \otimes x \otimes x] - \sum_{j \in [d_x]} \mathbb{E}[y \cdot e_j \otimes x \otimes e_j] + \sum_{j \in [d_x]} \mathbb{E}[y \cdot e_j \otimes e_j \otimes x] + \sum_{j \in [d_x]} \mathbb{E}[y \cdot x \otimes e_j \otimes e_j]. \tag{5} \]

**Lemma 2 (Adjusted third order moments).** We have

\[ M_3 = \sum_{i \in [r]} \rho_i w_i \cdot u_i \otimes u_i \otimes u_i, \tag{6} \]

where \( \rho_i := \mathbb{E}[\nabla^{(3)}_{x'_i} g(x'_i)] \) and the expectation is over \( x'_i = \langle u_i, x \rangle + \tilde{b}_i \).

The proof follows from Stein’s Identity. See Appendix A 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 3 (Recovery of mixture of GLMs).** Assuming that the weight matrix \( U \in \mathbb{R}^{d_x \times r} \) is full rank, \( \rho_i, w_i > 0 \ \forall \ i \), given \( M_3 \), we can recover the component weight vectors \( u_i, i \in [r] \), up to scaling, using tensor method given in Algorithm 4 (in the Appendix).

The proof follows from Lemma 2 and Theorem 7 (for incoherent \( U \)) or Theorem 8 (for general \( U \)). 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.

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

In practice, we do not have the exact moments \( M_3 \), but only empirical ones estimated from samples. In Section 5 we present a detailed analysis of sample complexity bounds.
Algorithm 1 Learning mixture of associative models

```
input Labeled samples \((x_i, y_i), i \in [n]\).
input Score function of the input.
1: Compute \(\hat{M}_3 = \frac{1}{n} \sum_i y_i S_3(x_i)\), Empirical estimate of \(M_3\).
2: if Whitening then
3: Calculate \(\hat{M}_3 = \text{Whiten}(\hat{M}_3)\) (Procedure 2 in the Appendix)
4: end if
5: \(\{u_j, \lambda_j\}_{j \in [r]} = \text{tensor power decomposition}(\hat{M}_3)\). (Algorithm 4 in the Appendix)
6: return \(\{u_j, \lambda_j\}_{j \in [r]}\).
```

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 to Stein’s identity

The key ingredient that enabled learning in the previous section is the ability to compute the expected derivatives of the label as a function of the input. Stein’s identity shows that these derivatives can be obtained using the cross-moments between the label and the 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 label 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 \(x\) with probability density \(p(x)\), denoted by \(S_1(x)\), is the random vector \(\nabla_x \log p(x)\). We (Janzamin et al., 2014), define the \(m\)th order score function as
\[
S_m(x) := (-1)^m \frac{\nabla^{(m)} p(x)}{p(x)}.
\] (7)

We have also shown that score function can be equivalently derived using the recursive form
\[
S_m(x) = -S_{m-1}(x) \otimes \nabla_x \log p(x) - \nabla_x S_{m-1}(x).
\] (8)

**Theorem 4** (Higher order derivatives (Janzamin et al., 2014)). For random vector \(x \in \mathbb{R}^{d_x}\), let \(p(x)\) and \(S_m(x)\) respectively denote the joint density function and the corresponding \(m\)-th order differential operator. Consider any continuously differentiable label-function \(\mathbb{E}[y|x] = g(x) : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}\) satisfying some mild regularity conditions. Then we have
\[
\mathbb{E} [y \cdot S_m(x)] = \mathbb{E} [g(x) \cdot S_m(x)] = \mathbb{E} \left[ \nabla^{(m)}_x g(x) \right].
\]

For details, see (Janzamin et al., 2014). In order to learn mixture of GLMs for general input distributions, we utilize score function \(S_m\) of order \(m = 3\).
4.2 Moment forms under general input distribution

We now consider the cross-moment $M_3 := \mathbb{E}[y \cdot S_3(x)]$, which is a third order tensor. By alluding to Theorem 4, we show that the moment $M_3$ has a CP decomposition where the components are the weight vectors $u_i$'s.

**Theorem 5 (Recovery of mixture of GLMs under general input).** Given score function $S_3(x)$ as in (7), we have

$$M_3 = \mathbb{E}[y \cdot S_3(x)] = \mathbb{E} \left[ \nabla^3_y \left( \sum_{i \in [r]} w_i g(\langle x, u_i \rangle + \tilde{b}_i) \right) \right] = \sum_{i \in [r]} \rho_i \cdot w_i \cdot u_i \otimes^3,$$

where $\rho_i := \mathbb{E}[\nabla^{(3)}_x g(x_i')]$ and the expectation is with respect to $x_i' := \langle u_i, x \rangle + \tilde{b}_i$.

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

The proof follows from Theorem 4 and Theorem 9. Thus, we have a guaranteed recovery of the weight vectors of mixture of GLMs under any general input distribution. We provide sample complexity bounds in Section 5.2 using Bernstein inequalities.

**Remark**: If we assume the $u_i$ are normalized, the above approach suffices to completely learn the parameters $w_i$. This is because we obtain $w_i \rho_i$ and we have the knowledge of $\rho_i$, 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 D.

4.3 Extension to Mixture of Non-Linear Classifiers

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

$$\mathbb{E}[y|x,h] = g((Uh, \phi(x))), \quad (9)$$

Similar to SVM framework (Bishop et al., 2006), $\phi(x)$ represents the nonlinear mapping on $x$. Assuming that $\phi(\cdot)$ is known, we propose simple ideas to extend our previous results to the setting in (9).

The key idea is to compute the score function $S_m(\phi(x))$ 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

$$p_{\phi(x)}(t_1, \cdots, t_p) = p_x(\phi^{-1}_1(t), \cdots, \phi^{-1}_p(t)) | \det(D_t)|, \quad (10)$$

$$S_m(t) = (-1)^m \frac{\nabla^m_{t_i} p_{\phi(x)}(t)}{p_{\phi(x)}(t)}.$$
Theorem 6 (Recovery of mixture of GLMs under general input). Given score function $S_3(\phi(x))$ as in Equation (10), we have

$$M_3 = \mathbb{E}[y \cdot S_3(\phi(x))] = \mathbb{E} \left[ \nabla^3_{\phi(x)} \left( \sum_{i \in [r]} w_i g(\langle \phi(x), u_i \rangle + \tilde{b}_i) \right) \right] = \sum_{i \in [r]} \rho_i \cdot w_i \cdot u_i \otimes^3 i,$$

where $\rho_i := \mathbb{E}[\nabla^3_{z_i} g(z_i)]$ and the expectation is with respect to $z_i := \langle u_i, \phi(x) \rangle + \tilde{b}_i$.

Assuming that matrix $U$ is full rank, $w_i, \rho_i > 0 \ \forall \ i$, we can recover the weight vectors $u_i, i \in [r]$, 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 a latent SVM under the realizable setting, given score function $S_3(\phi(x))$. Thus, the key ingredient in learning associative mixtures lies in estimation of the corresponding input score functions.

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. In addition, in deep learning, [Alain and Bengio, 2012] argue that denoising auto-encoders approximately learn the first order score function of the input, as the noise variance goes to zero. Therefore, we can use any of these methods for estimating $S_1(x)$ and use the recursive form in (8) to estimate higher order score functions for the active layer.

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

5 Sample Complexity Analysis

In this section, we provide sample complexity analysis for both Gaussian and general input. As discussed earlier, $M_3 = \mathbb{E}[y \cdot S_3]$. In practice we have access to empirical estimate of $M_3$ and we denote it by $\hat{M}_3$. Since we run the tensor decomposition algorithm on $\hat{M}_3$, in order to provide guarantees on recovery of the weight vectors we need the difference $\hat{M}_3 - M_3$ to be small enough. This enforces the number of required samples.

Let $\lambda_i = w_i \rho_i$, where $\rho_i$ is given by $\rho_i := \mathbb{E}[\nabla^3_{z_i} g(z_i)]$ and the expectation is with respect to $z_i := \langle u_i, x \rangle + \tilde{b}_i$ and $\Psi := \hat{M}_3 - M_3$. We assume that the weight vectors $u_i$ are normalized. We provide guarantees for two cases. First, assuming matrix $U$ is incoherent, we directly use tensor power method. Then following lines of [Anandkumar et al., 2014b], assuming conditions stated below are satisfied, we have the global convergence guarantees as in Theorem 7 for Gaussian input and in Theorem 9 for general input. Second, we remove incoherence assumption. In this case, we need to whiten the tensor before running the power method. The goal of whitening is to make the tensor components orthogonal to each other. In this case, in addition to the perturbations we have in the first case, we have additional whitening error. We have the global convergence guarantees as in Theorem 8 for Gaussian input. We provide the results in two extreme regimes of variance to contrast the corresponding sample bounds. The analysis can be easily extended to intermediate regimes of variance as well.
Settings for Theorem 7, 8, 9 The weight vectors \( u_i \) are generated uniformly at random from unit sphere \( S^{d_x-1} \) and \( \alpha_0 \geq 1 \) is a constant. Number of iterations of Algorithm 4 (in the Appendix), \( N = \Theta \left( \log \left( \frac{1}{\delta} \right) \right) \), where \( \gamma := \frac{\lambda_{\min}}{\lambda_{\min}} \), \( \lambda_i = \rho_i w_i, \rho_i := \mathbb{E}[\nabla_{x_i}^3 g(x_i)] \) and \( \hat{\epsilon}_R \) is the desired error bound on recovering \( U \) as discussed in the Theorems below. In initialization in each run of Algorithm 3 (in the Appendix) is performed by SVD-based technique proposed in Procedure 3 (in the Appendix) with the number of initializations as \( L \geq \Theta(\gamma^4(1+r/d_x)^2) \). Weight vector \( u_i \) is normalized and rank condition \( r = O(d_x) \).

5.1 Perturbation Analysis for Tensor Methods: Gaussian input

Following (Anandkumar et al., 2014b), we assume input vectors are independent of the hidden variable and \( x \sim \mathcal{N}(0, \sigma^2) \), \( \sigma^2 \) is a scalar denoting the variance of each entry. We provide perturbation bounds on estimation of matrix \( U = [u_1 \ldots u_r] \) and vector \( \Lambda = [\lambda_1 \ldots \lambda_r]^\top \).

**Theorem 7** (Perturbation bounds for Algorithm 4 for incoherent \( U \)). Assuming the above conditions hold, weight vectors are incoherent and number of samples, \( n \), satisfies

\[
n \geq \left\{ \begin{array}{ll}
\tilde{\Omega} \left( r^2 d_x^2 \right), & \sigma^2 = \Theta(1), \\
\tilde{\Omega} \left( r \sqrt{d_x} \right), & \sigma^2 = \Theta \left( \frac{1}{d_x} \right).
\end{array} \right.
\]  

(11)

Then, Algorithm 4 (in the Appendix) outputs \( \hat{U} \) and \( \hat{\Lambda} \) satisfying w.h.p.

\[
\|\hat{U} - U\|_F \leq \tilde{O} \left( \sqrt{\frac{\sigma^2 d_x^5}{\lambda_{\min}} \left( \frac{1}{n} + \frac{1}{d_x \sqrt{n}} \right)} \right), \quad \|\hat{\Lambda} - \Lambda\| \leq \tilde{O} \left( \sqrt{\frac{\sigma^2 d_x^5}{\lambda_{\min}} \left( \frac{1}{n} + \frac{1}{d_x \sqrt{n}} \right)} \right).
\]

See Appendix C.1 for proof.

Since we know the form of \( g(.) \) function, after learning \( \lambda_i \) and \( u_i \), we can learn the probabilities \( w_i, \ i \in [r] \), assuming that the \( u_i \) are normalized.

**Theorem 8** (Perturbation bounds under whitening). Assuming the above conditions hold and the number of samples, \( n \), satisfies

\[
n \geq \left\{ \begin{array}{ll}
\tilde{\Omega} \left( r^2 d_x^3 / \sigma_{\min}^6(U) \right), & \sigma^2 = \Theta(1), \\
\tilde{\Omega} \left( rd_x^3 / \sigma_{\min}^3(U) \right), & \sigma^2 = \Theta \left( \frac{1}{d_x} \right).
\end{array} \right.
\]  

(12)

Then, Algorithm 4 (in the Appendix) outputs \( \hat{U} \) and \( \hat{\Lambda} \) satisfying w.h.p.

\[
\|\hat{U} - U\|_F \leq \tilde{O} \left( \sqrt{\frac{\sigma^2 d_x^5}{\min \sigma_{\min}^4(U)} \left( \frac{1}{n} + \frac{1}{d_x \sqrt{n}} \right)} \right), \quad \|\hat{\Lambda} - \Lambda\| \leq \tilde{O} \left( \sqrt{\frac{\sigma^2 d_x^5}{\sigma_{\min}^4(U)} \left( \frac{1}{n} + \frac{1}{d_x \sqrt{n}} \right)} \right).
\]

See Appendix C.2 for proof.

5.2 Perturbation Analysis for Tensor Methods: General input

Let \( Z_i \) be the matricized version of the tensor \( y_i \cdot S_3(x_i) \) into a matrix \( d_1 \times d_f \) for sample \( i \). Assume \( \mathbb{E}[Z_i] = 0, \quad \|Z_i\| \leq B \) almost surely. Then the variance term is dominant in Bernstein inequality. Let \( C_Z := \max \left\{ \sum_{i \in [n]} Z_i Z_i^\top, \sum_{i \in [n]} Z_i^\top Z_i \right\} \). With probability \( 1 - \delta \), we have \( \|\hat{M}_3 - M_3\| \leq O \left( \sqrt{\frac{\ln(1/\delta) C_Z}{n}} \right) \).
Theorem 9 (Perturbation bounds for Algorithm 4 for incoherent $U$, general input). Let $r = O(d_x)$ and assume that conditions and settings mentioned above hold. Assuming the above conditions hold and the number of samples, $n$, satisfies

$$n \geq \Omega \left( \frac{C_z \ln 1/\delta}{\lambda_{\min}^2 \log r} \right).$$

Then, Algorithm 4 (in the Appendix) outputs $\hat{U}$ and $\hat{\Lambda}$ satisfying w.h.p.

$$\|\hat{U} - U\|_F \leq \tilde{O} \left( \frac{\sqrt{rC_z \ln 1/\delta}}{\lambda_{\min} \sqrt{n}} \right), \quad \|\hat{\Lambda} - \Lambda\| \leq \tilde{O} \left( \frac{\sqrt{rC_z \ln 1/\delta}}{n} \right).$$

For proof, see the Appendix C.3.

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 classifiers 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 or discriminative approaches such as latent SVMs (Yu and Joachims, 2009; Chang et al., 2010). 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 that can be exploited 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; T. 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

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 Proof of Lemma 2

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 $T(M_1, M_2, M_3) \in \mathbb{R}^{d_1 \otimes d_2 \otimes d_3}$ is defined as

$$T(M_1, M_2, M_3)_{i_1, j_2, k_3} := \sum_{j_1, j_2, j_3 \in [d]} T_{j_1, j_2, j_3} \cdot M_1(j_1, i_1) \cdot M_2(j_2, i_2) \cdot M_3(j_3, i_3). \quad (13)$$
In particular, for vectors $u, v, w \in \mathbb{R}^d$, we have
\begin{equation}
T(I, v, w) = \sum_{j,l \in [d]} v_j w_l T(:,j,l) \in \mathbb{R}^d,
\end{equation}
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 $l(x)$.
\[
\nabla_x l(x) = \nabla_x (g(\langle u, x \rangle) \cdot x \otimes x),
\]
\[
\mathbb{E}\{\nabla_x l(x)\} = \mathbb{E}[y \cdot \nabla_x (x \otimes x)] + \mathbb{E}[\nabla_x g(x')(x \otimes x \otimes u)].
\]

We now analyze the first term. We have
\[
\nabla_x (x \otimes x)_{i_1,i_2,j} = \frac{\partial x_{i_1}x_{i_2}}{\partial x_j} = \begin{cases} 
    x_{i_2}, & i_1 = j, \\
    x_{i_1}, & i_2 = j, \\
    2x_j, & i_1 = i_2 = j, \\
    0, & \text{o.w.}
\end{cases}
\]
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
\[
\mathbb{E}[\nabla_x' g(x') \cdot x \otimes u] = \mathbb{E}[f(x) \otimes x] = \mathbb{E}[\nabla_x f(x)],
\]
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
\[
\nabla_x f(x) = \nabla_x \left( \nabla_x' g(x')x \otimes u \right) = (\nabla_x^{(2)} g(x')) \cdot x \otimes u \otimes u + (\nabla_x' g(x')) \cdot \nabla_x (x \otimes u),
\]
\[\text{(17)}\]

\footnote{Compare with the matrix case where for $M \in \mathbb{R}^{d \times d}$, we have $M(I, u) = Mu := \sum_{j \in [d]} u_j M(:,j) \in \mathbb{R}^d$.}
and thus, $\nabla_x(x \otimes u) = \sum_i (e_i \otimes u \otimes e_i)$. The first term is given by

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

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

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

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). □

B Tensor Decomposition Method

We now recap the tensor decomposition method (Anandkumar et al., 2014c) to obtain the rank-1 components of a given tensor. This is given in Algorithm 4. Let $\hat{M}_3$ 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., 2014c) 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 $\hat{M}_3$ to find the components. Otherwise, we need to whiten the tensor first. We take a random slice of the empirical estimate of $\hat{M}_3$ and use it to find the whitening matrix. Let $V$ be the average of the random slices. The whitening matrix $\hat{W}$ can be found by using a rank-$r$ SVD on $\hat{V}$ 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., 2014c) 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.

Anandkumar et al. (2014c) argue that the tensor power method recovers the rank-1 components of the input tensor up to some fixed error when the rank-1 components are not necessarily orthogonal to each other. This is because the non-orthogonal true components are not the fixed points of power iteration. In order to remove this additional residual error, they propose a residual error removal method described in Algorithm 5. This method runs a coordinate descent iteration combined with a fixing procedure to remove the residual error remaining after running power iteration.

---

3If $E[y|x]$ 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 $\hat{M}_3 \in \mathbb{R}^{d_x \times d_x \times d_x}$.

1. Draw a random standard Gaussian vector $\theta \sim N(0, I_{d_x})$.
2. Compute $\hat{V} = \hat{M}_3(I, I, \theta) \in \mathbb{R}^{d_x \times d}$. 
3. Compute the rank-k SVD $\hat{V} = \tilde{U} \text{Diag}(\tilde{\lambda}) \tilde{U}^\top$.
4. Compute the whitening matrix $\hat{W} = \tilde{U} \text{Diag}(\tilde{\lambda}^{-1/2})$.
5. return $\hat{M}_3(\hat{W}, \hat{W}, \hat{W})$.

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

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

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

C Proof of Guarantees for Learning Mixture of Classifier

In this section, we provide the proof for sample complexity analysis for Gaussian and general input. For perturbation analysis for tensor power method, we follow Anandkumar et al. (2014d).

The goal is to bound the difference between $M_3$ and its empirical estimate $\hat{M}_3$. We define

$$\tilde{M}_3 := E[y \cdot S_3(x)]h_i, \ i \in [n] = \frac{1}{n} \sum_{i=1}^{n} \lambda_i u_i^{\otimes 3}, \quad (20)$$

where the expectation is conditioned on the choice of hidden states for $n$ samples, and taken over the randomness of input. Here, $h_i$ denotes the choice vector for sample $i$. Notice that tensor $\tilde{M}_3$ has the same form as true tensor $M_3$ in where $\tilde{M}_3 = \sum_{j \in [r]} \tilde{\lambda}_j u_j^{\otimes 3}$.

Here $\tilde{\lambda}_j, j \in [r]$ are the empirical frequencies of different choice vectors $h \in \{e_j, j \in [r]\}$. Anandkumar et al. (2014d) argue that when $n > \Omega(\frac{\log r}{\lambda_{\min}})$, for perturbation analysis of $M_3 - \hat{M}_3$, it suffices to bound $\tilde{M}_3 - \tilde{M}_3$. Therefore, we bound $\Psi_2 = \tilde{M}_3 - \tilde{M}_3$ which translates into bounding $\Psi = M_3 - \tilde{M}_3$.

C.1 Proof of Theorem 7

As we can see in Equation (5), for Gaussian input, $M_3$ consists of two parts, i.e., $E[y \cdot x^{\otimes 3}]$ and some other terms. For simplicity, we provide a two-step analysis for perturbation analysis of $M_3$. we first analyze $A_3 := E[y \cdot x^{\otimes 3}]$ and provide sample complexity bounds for this term. Next, we discuss that the additional terms are of lower order and hence do not have an impact on the sample complexity.

For sample complexity analysis in case of Gaussian mixtures, we use the idea presented in Anandkumar et al. (2014d), i.e., symmetrization and bucketing large and small terms. In our case, we assume that the
Algorithm 4 Robust tensor power method (Anandkumar et al., 2014c)

input symmetric tensor $\bar{T} \in \mathbb{R}^{d_x \times d_y \times d_z}$, number of iterations $N$, number of initializations $L$, parameter $\nu$.

output the estimated eigenvector/eigenvalue pair.

1:  for $\tau = 1$ to $L$ do
2:   Initialize $\hat{a}_0^{(\tau)}$ with SVD-based method in Procedure 3.
3:  for $t = 1$ to $N$ do
4:    Compute power iteration update
5:      $\hat{a}_t^{(\tau)} := \frac{\bar{T}(I, \hat{a}_{t-1}^{(\tau)}, \hat{a}_{t-1}^{(\tau)})}{\|\bar{T}(I, \hat{a}_{t-1}^{(\tau)}, \hat{a}_{t-1}^{(\tau)})\|}$ \hspace{1cm} (18)
6:  end for
7: $S := \{a_t^{(N+1)} : \tau \in [L]\}$
8: while $S$ is not empty do
9:  Choose $a \in S$ which maximizes $|T(a, a, a)|$.
10: Do $N$ more iterations of (18) starting from $a$.
11: Output the result of iterations denoted by $\hat{a}$.
12: Remove all the $a \in S$ with $|\langle a, \hat{a} \rangle| > \nu/2$.
13: end while

labels are bounded by some positive value $B$, i.e., $|y_i| \leq B \forall i$. we have

$$\tilde{A}_3 - \hat{A}_3 = \frac{1}{n} \sigma^3 d_x^{1.5} \sum_{i \in [n]} y_i a_i \otimes b_i \otimes c_i.$$ Using Claim 1 in Anandkumar et al. (2014b), we have with high probability

$$\left\| \frac{1}{n} \sum_{i \in [n]} y_i a_i \otimes b_i \otimes c_i \right\| \leq B \tilde{O}\left( \frac{1}{n} + \frac{1}{d_x \sqrt{n}} \right).$$

Hence

$$\| \tilde{A}_3 - \hat{A}_3 \| \leq \tilde{O}\left( \sigma^3 d_x^{1.5} \left( \frac{1}{n} + \frac{1}{d_x \sqrt{n}} \right) \right).$$ \hspace{1cm} (21)

In the high variance regime $\sigma^2 = \Theta(1)$ and in the low variance regime $\sigma^2 = \Theta(1/d_x)$. Therefore, in the former case, the term $\sigma^3 \sqrt{d_x / n}$ is the dominant term and in the latter case, the term $\sigma^3 d_x^{1.5} / n$ is the dominant term.

Following Anandkumar et al. (2014b), we require the above perturbation tensor $\Psi_2$ to be bounded as

$$\|\Psi_2\| \leq \frac{\lambda_{\min} \sqrt{\log r}}{a_0 \sqrt{d_x}}.$$
Algorithm 5 Coordinate descent algorithm for removing the residual error (Anandkumar et al., 2014d)

input Tensor $T \in \mathbb{R}^{d_x \times d_x \times d_x}$, initialization set $\{\hat{A}^{(0)}, \hat{B}^{(0)}, \hat{C}^{(0)}, \hat{w}^{(0)}\}$, number of iterations $N$.

1: for $t = 0$ to $N - 1$ do
2: for $i = 1$ to $r$ do
3:  \[
            \tilde{w}_i^{(t+1)} = \left\| T \left( \hat{A}_i^{(t)}, \hat{B}_i^{(t)}, I \right) - \sum_{j \neq i} \hat{w}_j^{(t)} \langle \hat{A}_i^{(t)}, \hat{A}_j^{(t)} \rangle \langle \hat{B}_i^{(t)}, \hat{B}_j^{(t)} \rangle \cdot \hat{C}_j^{(t)} \right\|, \\
            \tilde{C}_i^{(t+1)} = \frac{1}{\tilde{w}_i^{(t+1)}} \left( T \left( \hat{A}_i^{(t)}, \hat{B}_i^{(t)}, I \right) - \sum_{j \neq i} \hat{w}_j^{(t)} \langle \hat{A}_i^{(t)}, \hat{A}_j^{(t)} \rangle \langle \hat{B}_i^{(t)}, \hat{B}_j^{(t)} \rangle \cdot \hat{C}_j^{(t)} \right). \\
        \]
4: end for
5: Update $\hat{C}^{(t+1)}$ by applying Procedure 6 with inputs $\tilde{C}^{(t+1)}$ and $\hat{C}^{(t)}$.
6: Repeat the above steps (with appropriate changes) to update $\hat{A}^{(t+1)}$ and $\hat{B}^{(t+1)}$.
7: Update $\hat{w}^{(t+1)}$: for any $i \in [k],$
8: \[
            \hat{w}_i^{(t+1)} = \begin{cases} 
            \tilde{w}_i^{(t+1)}, \\
            \tilde{w}_i^{(t)} + \text{sgn}(\tilde{w}_i^{(t+1)} - \tilde{w}_i^{(t)}) \cdot \eta_0 \frac{\sqrt{\sigma}}{d}, \quad \text{o.w.}
        \end{cases}
\]
9: return $\{\hat{A}^{(N)}, \hat{B}^{(N)}, \hat{C}^{(N)}, \hat{w}^{(N)}\}$.

Assuming $\lambda_{\text{min}} = \mathcal{O}(1/r)$, the above bound translates into

\[
        n \geq \begin{cases} 
            \Omega \left( \frac{r^2 d_x^2}{\sigma^2} \right), & \sigma^2 = \Theta(1), \\
            \Omega \left( \frac{r \sqrt{d_x}}{\sigma^2} \right), & \sigma^2 = \Theta \left( \frac{1}{u_w} \right).
        \end{cases}
\]

Next, we use this concentration bound along with Theorem 11 in (Anandkumar et al., 2014b) to provide sample complexity guarantees for $A_3$.

Next, we consider the other terms in $M_3$. We have $M_3 = \mathbb{E}[y \cdot x^{\otimes 3}] - R$, where

\[
        R = \sum_{j \in [d]} \mathbb{E}[y \cdot e_j \otimes x \otimes e_j] + \sum_{j \in [d]} \mathbb{E}[y \cdot e_j \otimes e_j \otimes x] + \sum_{j \in [d]} \mathbb{E}[y \cdot x \otimes e_j \otimes e_j].
\]

This additional term is of the form (29c) in (Anandkumar et al., 2014b). Thus, using Claim 3 (Anandkumar et al., 2014b) and since Gaussian input satisfies RIP condition, we have with high probability

\[
        ||\hat{R} - \bar{R}|| \leq \tilde{O} \left( \sqrt{d_x} \left( \frac{1}{n} + \frac{w_{\text{max}}}{n} \right) \right). \\
\]

Hence, for perturbation analysis of $M_3$, we just need to add this new term to the error term in Equation (21), and proceed as before. This does not change the dominant terms for high variance and low variance regime compared to Section C.1 and thus, sample complexity remains the same.
### Procedure 6 Fixing procedure ([Anandkumar et al., 2014c])

**input** Matrices $\tilde{C}^{(t+1)}$, $\hat{C}^{(t)}$.

1. Compute the SVD of $\tilde{C}^{(t+1)} = UDV^\top$.
2. Let $\hat{D}$ be the truncated version of $D$ as
   $$\hat{D}_{i,i} := \min \left\{ D_{i,i}, \eta_1 \sqrt{\frac{k}{d}} \right\}.$$ 
3. Let $Q := UDV^\top$.
4. Update $\hat{C}^{(t+1)}$: for any $i \in [k]$,
   $$\hat{C}_i^{(t+1)} = \left\{ \begin{array}{ll}
   Q_i, & \|Q_i - \hat{C}_i^{(t)}\| \leq \eta_0 \sqrt{\frac{k}{d}}, \\
   \hat{C}_i^{(t)} + \eta_0 \sqrt{\frac{k}{d}} \frac{1}{\|Q_i - \hat{C}_i^{(t)}\|} \left( Q_i - \hat{C}_i^{(t)} \right), & \text{o.w.}
   \end{array} \right.$$ 
5. return $\hat{C}^{(t+1)}$.

### C.2 Proof of Theorem 8

In this section we prove the results for cases when weight vectors are not incoherent and we need to perform whitening prior to tensor decomposition. Since we do not have access to an empirical estimate of $M_2$, we use random slices of $M_3$ as shown in Algorithm 2. Therefore, in addition to the idea discussed in the previous section, for analysis of the perturbation, we need to consider the perturbation that results from whitening and how it changes the perturbation of the $M_3$, i.e., the tensor perturbation bound in this setting has an additional term as a result of whitening.

Following lines of [Song et al., 2013], we analyze the whitening error. It should be noted that our setting differs from that of Song et al. (2013) in the sense that we are analyzing the perturbation on slices rather than the second moment of input. Therefore, if we define $\epsilon_{pairs}$ to be the error we encounter with this empirical estimation of the slices, this error is bounded by the error we make in empirical estimate of $M_3$. It is easy to show that in the tensor perturbation bound this new term is not the dominant term and hence, following Song et al. (2013) and compensating for the difference in the setting we have that

$$\|\tilde{M}_3 - \hat{M}_3\| \leq \frac{2\sqrt{2}\epsilon_{triples}}{(d_x^2 \sigma_{\min}(U))^2},$$

where $\epsilon_{triples}$ refers to the error from the last section, i.e., the tensor perturbation without the whitening. The term $d_x^2$ follows from the fact that different slices have different weights depending on the $u_i$. For details on this term see Claim 2.9 ([Arora et al., 2012]).

Then, following [Anandkumar et al., 2014b], we require the above perturbation tensor $\Psi$ to be bounded as

$$\|\Psi\| \leq \frac{\lambda_{\min} \sqrt{\log r}}{\alpha_0 \sqrt{d_x}}.$$
Assuming $\lambda_{\min} = \mathcal{O}(1/r)$, the above bound translates into

\[
n \geq \begin{cases} 
\tilde{\Omega}\left(\frac{r^2d_x^5}{\sigma_{\min}(t)^5}\right), & \sigma^2 = \Theta(1), \\
\tilde{\Omega}\left(\frac{r^2d_x^5}{\sigma_{\min}(t)^5}\right), & \sigma^2 = \Theta\left(\frac{1}{d_x}\right), 
\end{cases}
\]

Next, we use this concentration bound along with Theorem 11 in \cite{Anandkumar:2014} to provide sample complexity guarantees for learning mixture of classifiers model.

### C.3 Proof of Theorem 9

For the general distribution case we proceed with similar approach as in the Gaussian case, except for bounding the error $\tilde{M}_3 - \hat{M}_3$. The reason is that, the earlier analysis requires RIP condition which may not hold for general input. Here, since we do not make any assumption on the probability distribution, we use Bernstein bound on the matricized version of the tensors which is more loose than the bound we derived for Gaussian case. Let $Z_i$ be the matricized version of the tensor $y_i \cdot S_3(x_i)$ into a $d_1 \times d_2$ matrix for sample $i$. Assume $E[Z_i] = 0$, $\|Z_i\| \leq B$ almost surely.

We define $\sigma^2 := \frac{1}{n} \max \left\{ \sum_{i \in [n]} Z_i Z_i^\top, \sum_{i \in [n]} Z_i^\top Z_i \right\}$. Then by matrix Bernstein \cite{Tropp:2012} we have that for all $t > 0$

\[
P\{\| \sum_{i \in [n]} Z_i \| \geq t \} \leq \{d_1 + d_2\} \exp\left\{ \frac{-t^2/2}{\sigma^2 + Bt} \right\}.
\]

Assuming bounded score function, the variance term dominates. Let

\[
C_Z := \max \left\{ \sum_{i \in [n]} Z_i Z_i^\top, \sum_{i \in [n]} Z_i^\top Z_i \right\}
\]

Therefore, we conclude that with probability $1 - \delta$,

\[
\| \tilde{M}_3 - \hat{M}_3 \| \leq \mathcal{O}\left( \sqrt{\frac{\ln(1/\delta)C_Z}{n}} \right).
\]

Following \cite{Anandkumar:2014}, for global convergence, we require the above perturbation tensor $\Psi$ to be bounded as

\[
\|\Psi\| \leq \frac{\lambda_{\min} \sqrt{\log r}}{\alpha_0 \sqrt{d_x}},
\]

for some constant $\alpha_0$. This is equivalent to

\[
n \geq \Omega\left(\frac{C_z \ln 1/\delta}{\lambda_{\min}^2 \log r}\right).
\]

The results follow from Theorem 11 in \cite{Anandkumar:2014}.
D 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 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 \( \nu = \{w_j, \nu_j, j = 1, \ldots, r\} \), 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,

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

where \( \Theta \) includes \( u_j, j = 1, \ldots, r \) and \( \nu \). Let

\[
Q^\theta(\nu) = \sum_t \sum_j f_j^{(k)}(y^{(t)}|x^{(t)}) \ln g_j^{(k)}(x^{(t)}, \nu^{(t)}),
\]

\[
Q_j^\nu(\nu_j) = \sum_t f_j^{(k)}(y^{(t)}|x^{(t)}) \ln p(x^{(t)}|\nu_j), \quad j \in [r]
\]

\[
Q_j^\theta(\theta_j) = \sum_t f_j^{(k)}(y^{(t)}|x^{(t)}) \ln p(y^{(t)}|x^{(t)}, \theta_j), \quad j \in [r]
\]

\[
Q^w = \sum_t \sum_j f_j^{(k)}(y^{(t)}|x^{(t)}) \ln w_j, \quad \text{with } w = \{w_1, \ldots, w_r\}
\]

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 $j = 1, \cdots, r$

$$u^{(k+1)}_j = \arg \max_{u_j} Q^e_j(u_j), \quad \nu^{(k+1)}_j = \arg \max_{\nu_j} Q^g_j(\nu_j),$$

$$w^{(k+1)} = \arg \max_w Q^w, \text{ s.t. } \sum_i w_i = 1.$$ 

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.

Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data generating distribution. arXiv preprint arXiv:1211.4246, 2012.

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, and Majid Janzamin. Guaranteed non-orthogonal tensor decomposition via alternating rank-1 updates. arXiv preprint arXiv:1402.5180, 2014c.

Sanjeev Arora, Rong Ge, Ankur Moitra, and Sushant Sachdeva. Provable ica with unknown gaussian noise, with implications for gaussian mixtures and autoencoders. In Advances in Neural Information Processing Systems, pages 2375–2383, 2012.

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

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 35(8):1798–1828, 2013.

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

Christopher M Bishop et al. Pattern recognition and machine learning, volume 1. springer New York, 2006.

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. Discriminative learning over constrained latent representations. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 429–437. Association for Computational Linguistics, 2010.
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. Fast Detection of Overlapping Communities via Online Tensor Methods. ArXiv 1309.0787, Sept. 2013.

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.

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.

Le Song, Animashree Anandkumar, Bo Dai, and Bo Xie. Nonparametric estimation of multi-view latent variable models. arXiv preprint arXiv:1311.3287, 2013.
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.

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

Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

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.