Supplementary Material for the AISTATS 2016 Paper: Provable Tensor Methods for Learning Mixtures of Generalized Linear Models

Hanie Sedghi* Majid Janzamin† Anima Anandkumar‡

1 Proofs

1.1 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

$$T(M_1, M_2, M_3)_{i_1,i_2,i_3} := \sum_{j_1,j_2,j_3\in[d]} T_{j_1,j_2,j_3} \cdot M_1(j_1, i_1) \cdot M_2(j_2, i_2) \cdot M_3(j_3, i_3). \tag{1}$$

In particular, for vectors $u, v, w \in \mathbb{R}^d$, we have [1]

$$T(I, v, w) = \sum_{j,l\in[d]} v_j w_l T(:, j, l) \ \in \mathbb{R}^d, \tag{2}$$

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

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

*Allen Institute for Artificial Intelligence. Email: hanies@allenai.org. This work was done while the author was a visiting researcher at University of California, Irvine.

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

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

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

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

$$\mathbb{E}\left[\nabla_x l(x)\right] = \mathbb{E}[y \cdot \nabla_x (x \otimes x)] + \mathbb{E}[(\nabla_{x'} g(x'))(x \otimes x \otimes u)]. \tag{3}$$

We now analyze the first term. We have

$$\nabla_x (x \otimes x)_{i_1,i_2,j} = \frac{\partial x_{i_1} x_{i_2}}{\partial x_j} = \begin{cases} x_{i_2}, & i_1 = j, \\ x_{i_1}, & i_2 = j, \\ 2x_j, & i_1 = i_2 = j, \\ 0, & \text{o.w.} \end{cases} \tag{4}$$

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 (3) 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 (3), let $f(x) := \nabla_{x'} g(x') \cdot x \otimes u$. The transposition of the second term in (3) is given by

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

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

$$\begin{aligned} \nabla_x f(x) &= \nabla_x \left(\nabla_{x'} g(x') x \otimes u\right) \\ &= (\nabla_{x'}^{(2)} g(x')) \cdot x \otimes u \otimes u + (\nabla_{x'} g(x')) \cdot \nabla_x (x \otimes u), \end{aligned} \tag{5}$$

The first term is given by

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

So the second term in (5) 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 (3).

We need to mention that, Lemma 3 can be directly proved by Theorem 9 as specific form of score function for Gaussian input. Here, we have provided step by step first principles proof of the lemma for easy understanding. □

1.2 Proof of Lemma 6

By replacing y by y^3 in Proof of Lemma 3 (Appendix 1.1), we have that

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

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.

1.3 Proof of Theorem 11

Proof.

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

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. ■

2 Tensor Decomposition Method

We now recap the tensor decomposition method Anandkumar et al. [2014] 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., 2014] 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[2]. 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.

[2]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 $\widehat{V} = \tilde{U} \operatorname{Diag}(\tilde{\lambda}) \tilde{U}^\top$.
4: Compute the whitening matrix $\widehat{W} = \tilde{U} \operatorname{Diag}(\tilde{\lambda}^{-1/2})$.
5: **return** $T\left(\widehat{W}, \widehat{W}, \widehat{W}\right)$.

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

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** $\hat{a}_0$.

Since the tensor decomposition problem is non-convex, it requires good initialization. We use the initialization algorithm from [Anandkumar et al., 2014] 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.

3 Expectation Maximization for Learning Unnormalized 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

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

Algorithm 4 Robust tensor power method [Anandkumar et al., 2014]

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

$$\hat{a}_t^{(\tau)} \quad := \quad \frac{T(I, \hat{a}_{t-1}^{(\tau)}, \hat{a}_{t-1}^{(\tau)})}{\|T(I, \hat{a}_{t-1}^{(\tau)}, \hat{a}_{t-1}^{(\tau)})\|} \tag{6}$$

  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 (6) 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

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

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

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 $j = 1, \cdots, r$

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

References

Animashree Anandkumar, Rong Ge, and Majid Janzamin. Guaranteed Non-Orthogonal Tensor Decomposition via Alternating Rank-1 Updates. *arXiv preprint arXiv:1402.5180*, Feb. 2014.

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.