Learning Latent Variable Models by Improving Spectral Solutions with Exterior Point Methods

Amirreza Shaban **Mehrdad Farajtabar** **Bo Xie** **Le Song** **Byron Boots**
College of Computing,
Georgia Institute of Technology
{amirreza, mehrdad, bo.xie}@gatech.edu {lsong, bboots}@cc.gatech.edu

Abstract

Probabilistic latent-variable models are a fundamental tool in statistics and machine learning. Despite their widespread use, identifying the parameters of basic latent variable models continues to be an extremely challenging problem. Traditional maximum likelihood-based learning algorithms find valid parameters, but suffer from high computational cost, slow convergence, and local optima. In contrast, recently developed spectral algorithms are computationally efficient and provide strong statistical guarantees, but are not guaranteed to find valid parameters. In this work, we introduce a two-stage learning algorithm for latent variable models. We first use a spectral method of moments algorithm to find a solution that is *close* to the optimal solution but not necessarily in the valid set of model parameters. We then incrementally refine the solution via an exterior point method until a local optima that is arbitrarily near the valid set of parameters is found. We perform several experiments on synthetic and real-world data and show that our approach is more accurate than previous work, especially when training data is limited.

1 INTRODUCTION & RELATED WORK

Probabilistic latent variable models are a fundamental tool in statistics and machine learning that have successfully been deployed in a wide range of applied domains including robotics, bioinformatics, speech recognition, document analysis, social network modeling, and economics. Despite their widespread use, identifying the parameters of basic latent variable models like multi-view models and hidden Markov models (HMMs) continues to be an extremely challenging problem. Researchers often resort to local search heuristics such as expectation maximization (EM) (Dempster et al., 1977) that attempt to find parameters that maximize the likelihood of the observed data. Unfortunately, EM has a number of well-documented drawbacks, including high computational cost, slow convergence, and local optima.

In the past 5 years, several techniques based on *method of moments* (Pearson, 1894) have been proposed as an alternative to maximum likelihood for learning latent variable models (Hsu et al., 2009; Siddiqi et al., 2010; Song et al., 2010; Parikh et al., 2011, 2012; Hsu and Kakade, 2012; Anandkumar et al., 2012a,c,b; Balle et al., 2012; Cohen et al., 2013; Song et al., 2014). These algorithms first estimate low-order moments of observations, such as means and covariances, and then apply a sequence of linear algebra to recover the model parameters. Moment estimation is linear in the number of training data samples, and parameter estimation, which relies on techniques like the singular value decomposition (SVD) is typically fast and numerically robust.

For example, moment-based algorithms have been proposed for learning *observable* representations of HMMs, which explicitly avoid recovering HMM transition and observation matrices (Hsu et al., 2009; Siddiqi et al., 2010; Song et al., 2010). These *spectral* algorithms first perform a SVD of second-order moments of adjacent observations, and then use this result, along with additional low-order moments, to recover parameters for filtering, predicting, and simulating from the system. Unlike previous maximum likelihood-based approaches, spectral algorithms are fast, statistically consistent, and do not resort to local search.

Spectral algorithms were recently extended to the more difficult problem of estimating the parameters of latent variable models including the stochastic transition and observation matrices of HMMs (Anandkumar et al., 2012c).[1] Again, the estimators are based on SVD and a sequence of linear operations, applied to low-order moments of observations and come with learning guarantees under mild rank

[1] In contrast to the *observable* representation identified by the previous spectral learning algorithms (Hsu et al., 2009; Siddiqi et al., 2010; Song et al., 2010).

conditions. This work has been further extended to learning parameters of parametric and nonparametric multi-view latent variable models (Anandkumar et al., 2012b; Song et al., 2014) by introducing a symmetric tensor decomposition algorithm that unifies several previous method of moments-based approaches.

One of the benefits of method of moments over EM and other local search heuristics is that moment-based algorithms come with theoretical guarantees such as statistical consistency and finite sample bounds. In other words, under mild assumptions, method of moments can guarantee that as the amount of training data increases, the learned parameters are converging to the true parameters of the model that generated the data (Hsu et al., 2009; Anandkumar et al., 2012b). This is especially promising because the resulting parameters can be used to *initialize* EM in a two-stage learning algorithm (Zhang et al., 2014; Balle et al., 2014), resulting in the best of both worlds: parameters found by method of moments provide a good initialization for a maximum likelihood approach.

Unfortunately, spectral method of moments algorithms and two-stage learning algorithms have worked less well in practice. With finite samples, method of moments estimators are *not* guaranteed to find a valid set of parameters. Although error in the estimated parameters are bounded, the parameters themselves may lie outside the class of valid models. For example, the learned transition matrix of a HMM may have small negative entries. A consequence is that the learned model cannot be used or even serve as an initialization for EM.

To fix these problems, the method of moments solution is typically projected onto the space of valid model parameters: e.g. by flipping the sign of negative parameters and renormalizing the model (Cohen et al., 2013), or projecting the parameters onto the ℓ_1-ball (Duchi et al., 2008). While these heuristics produce a useable model, it invalidates any theoretical guarantees: the resulting model may no longer be close to the true parameters. As demonstrated in Balle et al., models that are learned by method of moments and then "corrected" in this way, do not necessarily serve as a good initialization to EM (Balle et al., 2014).

1.1 EXTERIOR POINT METHODS

Consider the problem of minimizing objective function $r(\boldsymbol{v}) : \mathbb{R}^n \rightarrow \mathbb{R}^+$, subject to the constraint $\boldsymbol{v} \in \mathscr{A}$. Generally, a series of unconstrained optimization problem are solved to achieve a local optima in the limit. The optimization problem in the k^{th} step can be written as:

$$\text{minimize } r(\boldsymbol{v}) + l_k(\boldsymbol{v})$$

with the local optima $\boldsymbol{v}^{(k)}$. By defining the function $l_k(\boldsymbol{v})$ appropriately, one can then show that $\boldsymbol{v}^* = \lim_{k\rightarrow\infty} \boldsymbol{v}^{(k)}$ is a local optima of the original constrained optimization problem (Bloom, 2014). In *Interior point* methods every intermediate solution $\boldsymbol{v}^{(k)}$ is in the feasible set, however, in *exterior point* methods, only the convergence point of the sequence needs to be feasible (Byrne, 2008). Examples of interior and exterior point methods are Barrier function methods (Boyd and Vandenberghe, 2004), and exact penalty function methods (Fletcher, 2013) respectively. In exterior point methods, the function $l_k(\boldsymbol{v})$ usually has a positive value for solutions outside the feasible set to discourage these solutions and a value of zero for feasible inputs. In interior point methods, the function $l_k(\boldsymbol{v}) \rightarrow \infty$ when $\boldsymbol{v}$ approaches to the boundary of the constraint set. Polyak (2008), and Yamashita and Tanabe (2010) propose primal-dual exterior point methods for convex and non-convex optimization problems respectively. In Section 3, we show that by defining $l_k(\boldsymbol{v})$ appropriately, the algorithm converges to a local optima arbitrarily close to the feasible set by doing simple *forward-backward splitting* steps (Combettes and Pesquet, 2011).

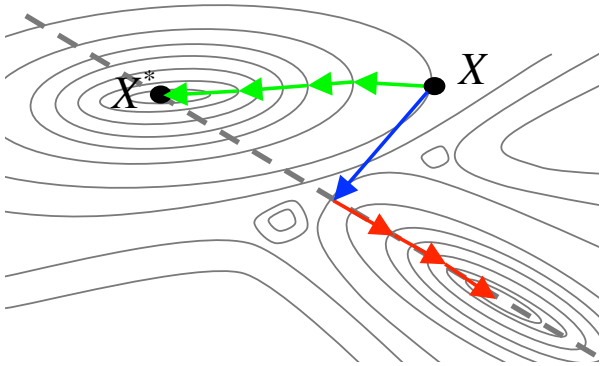


Figure 1: Exterior point methods versus projection followed by interior point methods. The dashed line shows the feasible set and the optimal solution is labeled $\boldsymbol{X}^*$. Starting from the method of moments solution $\boldsymbol{X}$, the exterior point method (green arrows) converges to a point arbitrarily close to the feasible set. Current optimization methods for learning latent variable models first project the method of moments solution into the feasible set (blue arrow) and then use an interior point method (red arrows show interior point method trajectory). Assuming that the initial solution is near to the optimal solution, as in this example, the projection step may change the convergence point to a point far from the optimal solution $\boldsymbol{X}^*$.

An important advantage of exterior point methods is that they are likely to achieve a better local minimum than interior point methods when the feasible set is narrow by allowing solutions to exist outside the feasible set during the intermediate steps of the optimization (Yamashita and Tanabe, 2010).

1.2 THE PROPOSED METHOD

One of the primary drawbacks of using method of moments for learning latent variable models, is that the estimated pa-

rameters can lie outside the class of valid models (Balle et al., 2014). To combat this problem, we propose a two-stage algorithm for learning the parameters of latent variable models.

In the first stage, the parameters are estimated via a spectral method of moments algorithm (similar to Anandkumar et al. (2012c)). Like previous method of moments-based learning algorithms, if the estimated moments are inaccurate, then the estimated parameters of this model may lie outside of the model class.

In the second stage, the estimate is refined by an iterative optimization scheme. Unlike previous work that projects method of moments onto the feasible space of model parameters and then uses the projected parameters to initialize EM (Zhang et al., 2014; Balle et al., 2014), we use exterior point methods for non-convex optimization directly initialized with the result of method of moments *without modification*. The exterior point method iteratively refines the solution until a local optima that is arbitrarily close to the valid set of model parameters is found. A comparison between the two approaches is illustrated in Figure 1.

1.3 BASICS AND NOTATION

We use following notation to distinguish scalars, vectors, matrices, and third-order tensors: scalars are denoted by either lowercase or uppercase letters, vectors are written as boldface lowercase letters, matrices correspond to boldface uppercase letters, and third-order tensors are represented by calligraphic letters. In this paper, $(\boldsymbol{A})_{ij}$ means the entry in the i^{th} row and j^{th} column of the matrix $\boldsymbol{A}$, we use similar notation to index entries of vectors and third-order tensors. Furthermore, the i^{th} *column* of the matrix $\boldsymbol{A}$ is denoted as $(\boldsymbol{A})_i$, i.e., $\boldsymbol{A} = [(\boldsymbol{A})_1, (\boldsymbol{A})_2, \ldots, (\boldsymbol{A})_n]$. We show a $n \times m$ matrix with entries one by $\mathbf{1}^{n\times m}$, $n \times n$ identity matrices by $\boldsymbol{I}^n$, and $n \times n \times n$ identity tensors by $\mathcal{I}^n$. We use $\mathbb{R}_+^{n\times m}$ to show the set of n by m matrices with non-negative entries, and Δ^n is the set of all $n+1$-dimensional vector on the n-dimensional simplex.

We also define the following functions for ease of notation: $\text{sum}(\boldsymbol{X}) = \boldsymbol{X}^\top \mathbf{1}$ computes column sum of the matrix $\boldsymbol{X}$, and the function $\text{diag}(\boldsymbol{v})$, which returns a diagonal matrix where its diagonal elements are a vector $\boldsymbol{v}$. For matrices or 3-way tensors, $\text{diag}(\cdot)$ returns diagonal elements of the given input in vector form.

1.3.1 n-mode product (Lathauwer et al., 2000)

The n-mode product of a tensor $\mathcal{A} \in \mathbb{R}^{I_1\times I_2\times I_3}$ by a matrix $\boldsymbol{B} \in \mathbb{R}^{J_n\times I_n}$ for $1 \le n \le 3$, shown as $\mathcal{A} \times_n \boldsymbol{B}$, is an $K_1 \times K_2 \times K_3$ tensor, for which $K_i = I_i$ for all dimensions except the n-th one which is $K_n = J_n$. The entries are given by:

$$(\mathcal{A} \times_n \boldsymbol{B})_{i_1\ldots j_n\ldots i_3} = \sum_{i_n} (\mathcal{A})_{i_1\ldots i_n\ldots i_3} (\boldsymbol{B})_{j_n i_n}. \quad (1)$$

We benefit from following properties of n-mode product in future sections:

- Given a tensor $\mathcal{A} \in \mathbb{R}^{I_1\times I_2\times I_3}$ and a matrix $\boldsymbol{C} \in \mathbb{R}^{J_n\times I_n}$ of the same size as $\boldsymbol{B}$, one can show that:
$$(\mathcal{A} \times_n \boldsymbol{B}) \times_n \boldsymbol{C} = \mathcal{A} \times_n (\boldsymbol{C}\boldsymbol{B}). \quad (2)$$
- For a matrix $\boldsymbol{D} \in \mathbb{R}^{J_m\times I_m} (n \ne m)$:
$$(\mathcal{A} \times_n \boldsymbol{B}) \times_m \boldsymbol{D} = (\mathcal{A} \times_m \boldsymbol{D}) \times_n \boldsymbol{B}.$$
- For matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ with appropriate sizes:
$$\boldsymbol{A} \times_1 \boldsymbol{B} \times_2 \boldsymbol{C} = \boldsymbol{B}\boldsymbol{A}\boldsymbol{C}^\top.$$

2 PARAMETER ESTIMATION VIA METHOD OF MOMENTS

In this section we derive two method of moments algorithms for estimating the parameters of latent variable models, one for multi-view models and one for HMMs. If the estimated parameters lie outside the feasible set of solutions, they are used to initiate an exterior point method (Section 3).

2.1 MULTI-VIEW MODELS

In a multi-view model, observation variables $o_1, o_2, \ldots, o_l$ are conditionally independent given a latent variable h (Figure 2a). Assume each observation variable can take one of n_o different values. The observation vector $\boldsymbol{x}_t \in \mathbb{R}^{n_o}$ is defined as follows:

$$\boldsymbol{x}_t = \boldsymbol{e}_j \text{ iff } o_t = j \text{ for } 0 < j \le n_o, \quad (3)$$

where $\boldsymbol{e}_j$ is the j^{th} canonical basis. In this paper, we consider the case where $l = 3$, however, the techniques can be easily extended to cases where $l > 3$. Let $h \in \{1, \ldots, n_s\}$ be a discrete random variable and $\Pr\{h = j\} = (\boldsymbol{w})_j$, where $\boldsymbol{w} \in \Delta^{n_s-1}$, then the conditional expectation of observation vector $\boldsymbol{x}_t$ for $t \in \{1, 2, 3\}$ is:

$$\mathbb{E}[\boldsymbol{x}_t \mid h = i] = \boldsymbol{u}_i^t, \quad (4)$$

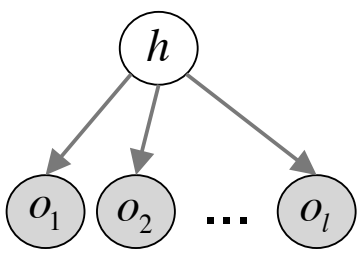
where $\boldsymbol{u}_i^t \in \Delta^{n_o-1}$. We define the observation matrix as $\boldsymbol{U}^t = [\boldsymbol{u}_1^t, \ldots, \boldsymbol{u}_{n_s}^t]$ for $t \in \{1, 2, 3\}$, and the diagonal $n_s \times n_s \times n_s$ tensor $\mathcal{H}$, where $\text{diag}(\mathcal{H}) = \boldsymbol{w}$ for ease of notation. The following proposition relates $\boldsymbol{U}^t$s and $\boldsymbol{w}$ to the moments of $\boldsymbol{x}_t$s.

Proposition 1. (Anandkumar et al., 2012b) *Assume that columns of $\boldsymbol{U}^t$ are linearly independent for each $t = \{1, 2, 3\}$. Define*

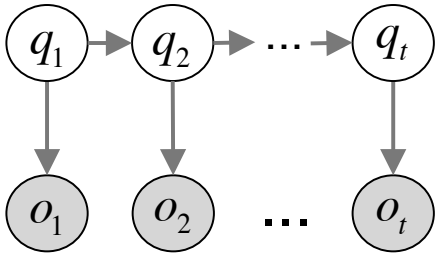
$$\mathcal{M} = \mathbb{E}(\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \boldsymbol{x}_3)$$

Then

$$\mathcal{M} = \mathcal{H} \times_1 \boldsymbol{U}^1 \times_2 \boldsymbol{U}^2 \times_3 \boldsymbol{U}^3 \quad (5)$$



(a) a multi-view model.



(b) A hidden Markov model.

Figure 2: The two latent variable models discussed in the text.

In the next proposition, the moments of $\boldsymbol{x}_t$s are related to a specific $\boldsymbol{U}^t$:

Proposition 2. (Anandkumar et al., 2012b) *Assume that columns of $\boldsymbol{U}_t$ are linearly independent for each $t = \{1,2,3\}$. Let (a,b,c) be a permutation of $\{1,2,3\}$. Define*

$$\begin{aligned}\boldsymbol{x}'_a &= \mathbb{E}(\boldsymbol{x}_c \otimes \boldsymbol{x}_b)\mathbb{E}(\boldsymbol{x}_a \otimes \boldsymbol{x}_b)^{-1}\boldsymbol{x}_a \\ \boldsymbol{x}'_b &= \mathbb{E}(\boldsymbol{x}_c \otimes \boldsymbol{x}_a)\mathbb{E}(\boldsymbol{x}_b \otimes \boldsymbol{x}_a)^{-1}\boldsymbol{x}_b \\ \boldsymbol{M}_c &= \mathbb{E}(\boldsymbol{x}'_a \otimes \boldsymbol{x}'_b) \\ \mathcal{M}_c &= \mathbb{E}(\boldsymbol{x}'_a \otimes \boldsymbol{x}'_b \otimes \boldsymbol{x}_c)\end{aligned}$$

Then

$$\begin{aligned}\boldsymbol{M}_c &= \boldsymbol{U}^c \operatorname{diag}(\mathbf{w}) \boldsymbol{U}^{c\top} \\ \mathcal{M}_c &= \mathcal{H} \times_1 \boldsymbol{U}^c \times_2 \boldsymbol{U}^c \times_3 \boldsymbol{U}^c\end{aligned} \tag{6}$$

Also, define $\boldsymbol{m}^c = \mathbb{E}(\boldsymbol{x}_c) = \boldsymbol{U}^c \boldsymbol{w}$. Anandkumar et al. (2012b) transform $\mathcal{M}_c$ to a orthogonally decomposable tensor and recover the matrices $\boldsymbol{U}^t$s and $\boldsymbol{w}$ from it. Our approach here is slightly different and is more similar to Anandkumar et al. (2012c): we reduce the problem into the orthogonal decomposition of a matrix derived from $\mathcal{M}_c$.

First, let $\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1/2}$, where columns of $\boldsymbol{V}$ are orthonormal eigenvectors of $\boldsymbol{M}_c$ and $\boldsymbol{\Sigma}$ is a diagonal matrix whose elements are corresponding eigenvalues of $\boldsymbol{V}$. The columns of $\tilde{\boldsymbol{U}}^c = \boldsymbol{S}^\top \boldsymbol{U}^c \operatorname{diag}(\boldsymbol{w})^{1/2}$ are orthonormal vectors (Anandkumar et al., 2012b). Using this and Equation (6) we have:

$$\begin{aligned}\boldsymbol{M}_{\boldsymbol{\eta}} =& \mathcal{M}_c \times_1 \boldsymbol{S}^\top \times_2 \boldsymbol{S}^\top \times_3 \boldsymbol{\eta}^\top \\ =& \mathcal{H} \times_1 (\boldsymbol{S}^\top \boldsymbol{U}^c) \times_2 (\boldsymbol{S}^\top \boldsymbol{U}^c) \times_3 (\boldsymbol{\eta}^\top \boldsymbol{U}^c) \\ =& \mathcal{I}_{n_s} \times_1 \tilde{\boldsymbol{U}}^c \times_2 \tilde{\boldsymbol{U}}^c \times_3 (\boldsymbol{\eta}^\top \boldsymbol{U}^c) \\ =& \tilde{\boldsymbol{U}}^c \operatorname{diag}(\boldsymbol{\eta}^\top \boldsymbol{U}^c)(\tilde{\boldsymbol{U}}^c)^\top\end{aligned} \tag{7}$$

where $\boldsymbol{\eta}$ is a random vector sampled from the n_o dimensional normal distribution. For the first equality we used property in Equation (2), in the second equality we used the fact that $\mathcal{H} = \mathcal{I}_{n_s} \times_1 \operatorname{diag}(\boldsymbol{w})^{1/2} \times_2 \operatorname{diag}(\boldsymbol{w})^{1/2}$, and finally in the last equality we used equality $\mathcal{I}_{n_s} \times_3 (\boldsymbol{\eta}^\top \boldsymbol{U}^c) = \operatorname{diag}(\boldsymbol{\eta}^\top \boldsymbol{U}^c)$. In Anandkumar et al. (2012c) it is shown that $\boldsymbol{\eta}^\top \boldsymbol{U}^c$ has distinct values with a high probability. Thus, $\tilde{\boldsymbol{U}}^c$ can be recovered by a SVD decomposition of $\boldsymbol{M}_{\boldsymbol{\eta}}$. Then $\boldsymbol{w} = ((\tilde{\boldsymbol{U}}^c)^+ \boldsymbol{S}^\top \boldsymbol{m}^c)^{\wedge} 2$, where $^{\wedge}$ is element-wise power operator. Having computed $\boldsymbol{w}$, one can recover $\boldsymbol{U}^c$ via the equation $\boldsymbol{U}^c = \boldsymbol{S}^{+\top} \tilde{\boldsymbol{U}}^c \operatorname{diag}(\boldsymbol{w})^{-1/2}$. Finally, we take the average of 3 copies of $\boldsymbol{w}$ which are computed for different values of c. The overall moment-based approach is shown in Algorithm 1.

Algorithm 1 *Moment-based parameter estimation*

Input: Estimated third order moment $\hat{\mathcal{M}}_c$ for $c = \{1,2,3\}$
Output: Estimated parameters $\hat{\boldsymbol{U}}^1, \hat{\boldsymbol{U}}^2, \hat{\boldsymbol{U}}^3, \hat{\boldsymbol{w}}$

for each $t \in \{1,2,3\}$ **do**
  $\hat{\boldsymbol{M}}_t \leftarrow \hat{\mathcal{M}}_t \times_3 \mathbf{1}$ (compute second-order moment)
  $\hat{\boldsymbol{m}}_t \leftarrow \hat{\mathcal{M}}_t \times_2 \mathbf{1} \times_3 \mathbf{1}$ (compute first-order moment)
  $\boldsymbol{S} \leftarrow \boldsymbol{V}\boldsymbol{\Sigma}^{-1/2}$ ($\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is $\hat{\boldsymbol{M}}_t$'s eigenvalue decomposition)
  $\boldsymbol{\eta} \leftarrow$ drawn randomly from Normal distribution
  $\boldsymbol{M}_{\boldsymbol{\eta}} \leftarrow \hat{\mathcal{M}}_t \times_1 \boldsymbol{S}^\top \times_2 \boldsymbol{S}^\top \times_3 \boldsymbol{\eta}^\top$ (Eq. 7)
  $\tilde{\boldsymbol{U}}^t \leftarrow \boldsymbol{K}$ (columns of $\boldsymbol{K}$ are $\boldsymbol{M}_{\boldsymbol{\eta}}$'s eigenvectors)
  $\hat{\boldsymbol{w}}^t \leftarrow ((\tilde{\boldsymbol{U}}^t)^+ \boldsymbol{S}^\top \hat{\boldsymbol{m}}^t)^{\wedge} 2$
  $\hat{\boldsymbol{U}}^t \leftarrow \boldsymbol{S}^{+\top} \tilde{\boldsymbol{U}}^t \operatorname{diag}(\hat{\boldsymbol{w}}^t)^{-1/2}$
end for
$\hat{\boldsymbol{w}} \leftarrow \frac{\hat{\boldsymbol{w}}^1 + \hat{\boldsymbol{w}}^2 + \hat{\boldsymbol{w}}^3}{3}$

2.2 HIDDEN MARKOV MODELS

Hidden Markov models generate sequences of observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \in \mathbb{R}^{n_o}$. Each $\boldsymbol{x}_t$ is independent of all other observations given the corresponding hidden state $q_t \in \{1,2,\ldots,n_s\}$ (Figure 2b). Similar to multi-view models, n_s and n_o are the number of hidden states and number of observations respectively. Note that observations are represented as *indicator vectors* $\boldsymbol{x}_t$, which are all zero except for exactly one element which is set to 1. The conditional probability distribution of $\boldsymbol{x}_t$ given q_t is defined via an observation matrix $\boldsymbol{O} \in \mathbb{R}^{n_o \times n_s}$ according to $\Pr\{\boldsymbol{x}_t = \boldsymbol{e}_i | q_t = j\} = (\boldsymbol{O})_{ij}$. The stochastic transition matrix $\boldsymbol{T} \in \mathbb{R}^{n_s \times n_s}$ is defined as $\Pr\{q_{t+1} = i | q_t = j\} = (\boldsymbol{T})_{ij}$ for all $t > 1$ and the initial state probability distribu-

tion is $\boldsymbol{\pi} \in \Delta^{n_s-1}$. If $\mathfrak{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ is a sequence of observations, then we define forward and backward variables as

$$\begin{aligned} \Pr\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, q_t = j\} &= \alpha_t(j) \\ \Pr\{\boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_T, q_t = j\} &= \beta_t(j) \end{aligned} \tag{8}$$

These will help computing the probability of observations. For example,

$$f(\mathfrak{X}; [\boldsymbol{O}, \boldsymbol{T}, \boldsymbol{\pi}]) = \sum_{i=1}^{n_s} \sum_{j=1}^{n_s} \alpha_t(i) (\boldsymbol{T})_{ij} (\boldsymbol{O})_j^\top \boldsymbol{x}_t \beta_{t+1}(j) \tag{9}$$

for all $1 \leq t \leq T$ (Levinson et al., 1983). Note that the values of function $\alpha_t(.)$ and $\beta_t(.)$ can be computed efficiently using dynamic programming.

Under mild conditions, HMM parameter estimation reduces to estimating multi-view model parameters, using considering triple of observation $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$.

Proposition 3. (Anandkumar et al., 2012b) *let $h = q_2$ then:*

- *$\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ are conditionally independent given h.*
- *The distribution of h is $\boldsymbol{w} = \boldsymbol{T}\boldsymbol{\pi}$.*
- *For all $j \in \{1, 2, \ldots, n_s\}$*

$$\begin{aligned} \mathbb{E}[\boldsymbol{x}_1 | h = j] &= \boldsymbol{O}\, \mathrm{diag}\,(\boldsymbol{\pi}) \boldsymbol{T}^\top\, \mathrm{diag}\,(\boldsymbol{w})^{-1/2} \boldsymbol{e}_j \\ \mathbb{E}[\boldsymbol{x}_2 | h = j] &= \boldsymbol{O}\boldsymbol{e}_j \\ \mathbb{E}[\boldsymbol{x}_3 | h = j] &= \boldsymbol{O}\boldsymbol{T}\boldsymbol{e}_j \end{aligned}$$

under mild conditions.

Thus, provided that $\boldsymbol{O}$ and $\boldsymbol{T}$ both have full column rank, the parameters of HMM can be recovered as $\boldsymbol{O} = \boldsymbol{U}^2$, $\boldsymbol{T} = \boldsymbol{O}^+ \boldsymbol{U}^3$, and $\boldsymbol{\pi} = \boldsymbol{T}^{-1}\boldsymbol{w}$.

It is also important to note that, using the above proposition and Equation (9) we can alternatively write each entries of tensor $\mathcal{M} = \mathbb{E}(\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \boldsymbol{x}_3)$ as:

$$(\mathcal{M})_{ijk} = f((\boldsymbol{e}_i, \boldsymbol{e}_j, \boldsymbol{e}_k); [\boldsymbol{O}, \boldsymbol{T}, \boldsymbol{\pi}]). \tag{10}$$

3 EXTERIOR POINT METHODS

While exact parameters can be recovered from the population moments, in practice we work with empirical moments $\hat{\boldsymbol{M}}$, and $\hat{\mathcal{M}}$ which are computed using a finite set of training data. Thus, the estimated parameters are not necessarily exact and do not necessarily minimize the estimation error $||\hat{\mathcal{M}} - \mathcal{H} \times_1 \boldsymbol{U}^1 \times_2 \boldsymbol{U}^2 \times_3 \boldsymbol{U}^3||_F$.

In this section, we show that estimated parameters from Section 2 can directly initialize an iterative exterior point method that minimizes the above error while obeying constraints on model parameters. Although this initial seed may violate these constraints, we show that under mild conditions the parameters satisfy the model constraints once the algorithm converges. First, we prove the convergence of the algorithm for multi-view models and then show how the algorithm can be applied to HMMs.

3.1 MULTI-VIEW MODELS

Let $\boldsymbol{v} \in \mathbb{R}^{n_s(3n_o+1)}$ be a vector comprised of the parameters of the multi-view model $\{\boldsymbol{U}^1, \boldsymbol{U}^2, \boldsymbol{U}^3, \mathrm{diag}(\mathcal{H})\}$, and $\mathcal{R}(\boldsymbol{v}) = (\hat{\mathcal{M}} - \mathcal{H} \times_1 \boldsymbol{U}^1 \times_2 \boldsymbol{U}^2 \times_3 \boldsymbol{U}^3)$ be the residual estimation tensor. For ease of notation, we also define function $s(\cdot) : \mathbb{R}^{n_s(3n_o+1)} \rightarrow \mathbb{R}^{3n_s+1}$ that computes column sum of $\boldsymbol{U}^1$, $\boldsymbol{U}^2$, $\boldsymbol{U}^3$, and $\mathrm{diag}(\mathcal{H})$. As discussed above, the estimated parameters in the previous section do not necessarily minimize the estimation error $||\mathcal{R}(\boldsymbol{v})||_F$ and also may violate the constraints for the model parameters. With these limitations in mind, we rewrite the factorization in Equation (6) in the form of an optimization problem:

$$\begin{aligned} &\text{minimize } \frac{1}{2}||\mathcal{R}(\boldsymbol{v})||_F^2. \\ &\text{s.t. } \boldsymbol{v} \in \mathbb{R}_+^{n_s(3n_o+1)}, \mathrm{s}(\boldsymbol{v}) = \boldsymbol{1} \end{aligned} \tag{11}$$

Defining optimization problem in this form has two advantages over maximum likelihood optimization schemes. First, since $\hat{\mathcal{M}}$ is computed in the previous stage, the optimization cost is asymptotically independent of the number of training samples which makes the proposed optimization algorithm faster than EM for large training sets. Second, the value of this objective function $||\mathcal{R}(\boldsymbol{v})||_F$ is also defined *outside* of the feasible set. We use this property to extend the optimization problem for a simple exterior point method in Section 3.1.1, below.

3.1.1 The Optimization Algorithm

Instead of solving constrained optimization problem in Equation (11), we solve the following unconstrained optimization problem:

$$\begin{aligned} \text{minimize } \frac{1}{2}||\mathcal{R}(\boldsymbol{v})||_F^2 + \frac{\lambda_1}{2}||\mathrm{s}(\boldsymbol{v}) - \boldsymbol{1}||_p^2 \\ + \lambda_2 |\boldsymbol{v}|_-, \end{aligned} \tag{12}$$

where $|\boldsymbol{v}|_-$ is the absolute sum of all negative elements in the vector $\boldsymbol{v}$, i.e., $|\boldsymbol{v}|_- = \sum_i |(\boldsymbol{v})_i|_-$. We set $p = 2$ in our method. For $p = 1$, there exists a λ_1 and a λ_2 such that the solution to this unconstrained optimization is *also* the solution to the objective in Equation (11). A thorough survey on solving non-differentiable exact penalty functions can be found in (Fletcher, 2013). Our approach, however, is different in the sense that for $p = 2$ the solution of our optimization algorithm is not guaranteed to satisfy the constraints in Equation (11), however, we show in Theorem 7 that the solution will be arbitrarily close to the simplex. In return for this relaxation, the above optimization problem can be easily solved by a standard *forward-backward splitting* algorithm (Combettes and Pesquet, 2011). In this

Algorithm 2 *The exterior point algorithm*

Input: Estimated third order moment $\hat{\mathcal{M}}$, initial point (obtained from Algorithm 1) $\boldsymbol{v}^{(0)}$ is comprised of $\{\hat{\boldsymbol{U}}^1, \hat{\boldsymbol{U}}^2, \hat{\boldsymbol{U}}^3, \text{diag}(\hat{\mathcal{H}})\}$, parameters λ_1, and λ_2, sequence $\{\beta_k\}$, and constant $c > 0$
Output: Convergence point $\boldsymbol{v}^*$

$k \leftarrow 0$
while not converged **do**
  $k \leftarrow k + 1$
  if $|\boldsymbol{v}^{(k-1)}|_- > 0$ **then**
    $\alpha_k \leftarrow max\{c, \beta_k\}$
  else
    $\alpha_k \leftarrow \beta_k$
  end if
  $\tilde{\boldsymbol{v}}^{(k)} \leftarrow \boldsymbol{v}^{(k-1)} - \alpha_k \nabla g(\boldsymbol{v}^{(k-1)})$
  $\boldsymbol{v}^{(k)} \leftarrow \text{prox}(\tilde{\boldsymbol{v}}^{(k)})$ (see Equation (15))
end while

method, the function is split into a smooth part:

$$g(\boldsymbol{v}) = \frac{1}{2}||\mathcal{R}(\boldsymbol{v})||_F^2 + \frac{\lambda_1}{2}||\text{s}(\boldsymbol{v}) - \mathbf{1}||_2^2 \tag{13}$$

and a non-smooth part $\lambda_2|\boldsymbol{v}|_-$. We then minimize the objective function by alternating between a gradient step on the smooth part $\nabla g(\boldsymbol{v})$ (forward) and proximal step of the non-smooth part (backward). The overall algorithm is shown in Algorithm 2 where $\text{prox}(\tilde{\boldsymbol{v}}^{(k)})$ function is defined as

$$\boldsymbol{v}^{(k)} = \underset{\boldsymbol{y}}{\text{argmin}}(\alpha_k \lambda_2 |\boldsymbol{y}|_- + \frac{1}{2}||\tilde{\boldsymbol{v}}^{(k)} - \boldsymbol{y}||_F^2). \tag{14}$$

and optimized by following transformation (Shalev-Shwartz and Zhang, 2013):

$$(\boldsymbol{v}^{(k)})_i = \begin{cases} (\tilde{\boldsymbol{v}}^{(k)})_i + \alpha_k \lambda_2 & (\tilde{\boldsymbol{v}}^{(k)})_i < -\alpha_k \lambda_2 \\ 0 & -\alpha_k \lambda_2 \leq (\tilde{\boldsymbol{v}}^{(k)})_i < 0 \\ (\tilde{\boldsymbol{v}}^{(k)})_i & 0 \leq (\tilde{\boldsymbol{v}}^{(k)})_i \end{cases} \tag{15}$$

We only need to find the gradient of function $g(\boldsymbol{v})$ for the given model parameter $\boldsymbol{v}$. Following lemma shows the gradient of $g(\boldsymbol{v})$ can be computed efficiently.

Lemma 4. *Let $r(\boldsymbol{v}) = \frac{1}{2}||\mathcal{R}(\boldsymbol{v})||_F^2$. The following are true for all $0 < i \leq n_s$:*

a. $\frac{\partial r(\boldsymbol{v})}{\partial \mathcal{H}_{iii}} = -\mathcal{R}(\boldsymbol{v}) \times_1 \boldsymbol{u}_i^1 \times_2 \boldsymbol{u}_i^2 \times_3 \boldsymbol{u}_i^3$

b. $\nabla_{\boldsymbol{u}_i^1} r(\boldsymbol{v}) = -\mathcal{H}_{iii} \times \mathcal{R}(\boldsymbol{v}) \times_2 \boldsymbol{u}_i^2 \times_3 \boldsymbol{u}_i^3$

c. $\nabla_{\boldsymbol{u}_i^2} r(\boldsymbol{v}) = -\mathcal{H}_{iii} \times \mathcal{R}(\boldsymbol{v}) \times_1 \boldsymbol{u}_i^1 \times_3 \boldsymbol{u}_i^3$

d. $\nabla_{\boldsymbol{u}_i^3} r(\boldsymbol{v}) = -\mathcal{H}_{iii} \times \mathcal{R}(\boldsymbol{v}) \times_1 \boldsymbol{u}_i^1 \times_2 \boldsymbol{u}_i^2$

Next we show that by using the forward-backward splitting steps in Algorithm 2 there are lower bounds for λ_1 and λ_2 in which the iterative algorithm converges to a local optimum arbitrarily close to the simplex. For this purpose, we assume that estimated parameters remain in a compact set during the optimization, then we use the following corollary to bound the gradient of function $r(\boldsymbol{v})$.

Corollary 5. *For every $\boldsymbol{v}$ which is comprised of the multi-view parameters and is in the compact set $\mathscr{F} = \{\boldsymbol{v} \mid \forall i, 0 < i \leq n_s, 1 < t \leq 3 : ||(\boldsymbol{U}^t)_i||_2 \leq L, ||diag(\mathcal{H})||_2 \leq L\}$, every element of the gradient of the function $r(\boldsymbol{v})$ is bounded as:*

$$|\frac{\partial r(\boldsymbol{v})}{\partial (\boldsymbol{v})_i}| \leq L^3 ||\mathcal{R}(\boldsymbol{v})||_F \tag{16}$$

Although, the norm of the residual error can also be bounded by L in Equation (16), since we initiate the algorithm with method of moments estimation, its value remains considerably smaller than its upper bound during the iterative procedure in Algorithm 2. Let $\Upsilon > \sup_k ||\mathcal{R}(\boldsymbol{v}^{(k)})||_F$; the next lemma shows that for a large enough λ_2 and after a fixed number of iterations, all of the elements in $\boldsymbol{v}^{(k)}$ become non-negative.

Lemma 6. *Assuming that sequence $\{\boldsymbol{v}^{(k)}\}$ produced by Algorithm 2 is in the set $\mathscr{F}$, and λ_2 is selected such that:*

$$\lambda_2 > L^3 \Upsilon + \lambda_1(\sqrt{n_o} L + 1), \tag{17}$$

there is a constant K such that for $k > K$ we have $|\boldsymbol{v}^{(k)}|_- = 0$. Also, for $k > K$ the proximal operator in Algorithm 2 reduces to the orthogonal projection operator into the convex set $\mathscr{C} = \mathbb{R}_+^{n_s(3n_o+1)}$:

$$\forall k > K : prox(\tilde{\boldsymbol{v}}^{(k)}) = proj_{\mathscr{C}}(\tilde{\boldsymbol{v}}^{(k)}) \tag{18}$$

The proof is provided in the Appendix. According to the above lemma, after K iterations of forward-backward splitting steps, all entries of $\boldsymbol{v}^{(k)}$ have non-negative values and the optimization algorithm reduces to gradient projection steps (Bertsekas, 1999) into the set $\mathbb{R}^{n_s(3n_o+1)}$ for optimizing non-convex function $g(\boldsymbol{v})$. There is a lot of research on the convergence guarantees of gradient projection methods for non-convex optimization with different line search algorithms (Bertsekas, 1999), which also can be used in Algorithm 2. The only requirement of our method is that stepsize α_k should be strictly bounded away from zero for $k \leq K$ which is guaranteed in Algorithm 2 by taking the $max\{c, \beta_k\}$ for $k \leq K$ for an arbitrary constant $c > 0$. Finally, the following theorem shows that by choosing large enough λ_1 and λ_2, the algorithm ends up with a solution arbitrarily close to the simplex.

Theorem 7. *For every $\epsilon_1 > 0$, set $\lambda_1 > \frac{L^3 \Upsilon}{\epsilon_1}$ and $\lambda_2 > L^3 \Upsilon + \lambda_1(\sqrt{n_o} L + 1)$, in Equation (12). For the convergence point of the sequence $\{\boldsymbol{v}^{(k)}\} \subset \mathscr{F}$ which is generated by Algorithm 2 we have:*

$$|\boldsymbol{v}^*|_- = 0, \; ||s(\boldsymbol{v}^*) - \mathbf{1}||_2 \leq \epsilon_1$$

The proof of Theorem 7 is provided in the Appendix.

To summarize, we initiate our exterior point method with the result of the method of moments estimator in Sec-

tion 2.1. By choosing large enough λ_1, and λ_2, the convergence point of the exterior point method will be at a local optimum of $g(\boldsymbol{v})$ with the constraint $\boldsymbol{v} \in \mathbb{R}_+^{n_s(3n_o+1)}$ in which the column sum of parameters set is arbitrarily close to 1. It is important to note that the proven lower bounds for λ_1, and λ_2 are sufficient condition for the convergence. In practice, cross-validation can find the best parameter to balance the speed of mapping to the simplex with optimizing the residual function.

3.2 HIDDEN MARKOV MODELS

In Section 2.2 we showed that method of moments parameter estimation for HMMs essentially reduces to parameter estimation of of multi-view models. After finding parameters from the method of moments algorithm, Algorithm 2 can be used to further refine the solution. In order to use this algorithm, we just need to define the residual estimation term for HMMs, define the function $g(\cdot)$, and compute its gradient. Assuming the parameters of the HMM $\boldsymbol{T}$, $\boldsymbol{O}$, and $\boldsymbol{\pi}$ are as defined in Section 2.2, let vector $\boldsymbol{z}$ be comprised of these parameters. Similar to the multi-view case when the estimated moments are not exact, the equality in Equation (10) does not hold, and we define the residual prediction error of the model for the triples $(\boldsymbol{e}_i, \boldsymbol{e}_j, \boldsymbol{e}_k)$ as $(\mathcal{R}(\boldsymbol{z}))_{ijk} = (\hat{\mathcal{M}})_{ijk} - f((\boldsymbol{e}_i, \boldsymbol{e}_j, \boldsymbol{e}_k); [\boldsymbol{O}, \boldsymbol{T}, \boldsymbol{\pi}])$. Thus, the optimization problem is:

$$\text{minimize } g(\boldsymbol{z}) + \lambda_2 |\boldsymbol{z}|_-, \tag{19}$$

where $g(\boldsymbol{z})$ is defined as:

$$g(\boldsymbol{z}) = \frac{1}{2}||\mathcal{R}(\boldsymbol{z})||_F^2 + \frac{\lambda_1}{2}||\mathrm{s}(\boldsymbol{z}) - \mathbf{1}||_2^2. \tag{20}$$

The following lemma shows that the gradient of the first term in above equation can be represented by forward and backward variables in Equation (8) efficiently.

Lemma 8. *Let $r(\boldsymbol{z}) = \frac{1}{2}||\mathcal{R}(\boldsymbol{z})||_F^2$, for all $0 < a, b \leq n_s$ and $0 < c \leq n_o$ following holds:*

a. $\frac{\partial r(\boldsymbol{z})}{\partial (\boldsymbol{\pi})_a} = \sum (\mathcal{R}(\boldsymbol{z}))_{ijk} (\boldsymbol{O})_a^\top \boldsymbol{x}_1 \beta_1(a)$

b. $\frac{\partial r(\boldsymbol{z})}{\partial (\boldsymbol{T})_{ab}} = \sum (\mathcal{R}(\boldsymbol{z}))_{ijk} \sum_{t=1}^{2} \alpha_t(a) (\boldsymbol{O})_b^\top \boldsymbol{x}_{t+1} \beta_{t+1}(b)$

c. $\frac{\partial r(\boldsymbol{z})}{\partial (\boldsymbol{O})_{cb}} = \frac{1}{(\boldsymbol{O})_{cb}} \sum (\mathcal{R}(\boldsymbol{z}))_{ijk} \sum_{t:\boldsymbol{x}_t = \boldsymbol{e}_c} \alpha_t(b) \beta_t(b)$

where the outer sums are over $0 < i, j, k < n_o$ and $\boldsymbol{x}_1 = \boldsymbol{e}_i$, $\boldsymbol{x}_2 = \boldsymbol{e}_j$, and $\boldsymbol{x}_3 = \boldsymbol{e}_k$ ($\boldsymbol{e}_i$ is the i^{th} canonical basis).

To summarize: to estimate the parameters of a HMM, we initialize Algorithm 2 with the method of moments estimate of the parameters. Then, using lemma 8, we compute $\nabla g(\boldsymbol{z})$ at each iteration to solve the optimization (Equation (19)).

4 EXPERIMENTAL RESULTS

We evaluate the performance of our proposed method (EX&SVD) on both synthetic and real world datasets. We compare our approach to several state-of-the-art alternatives including EM initialized with 10 random seeds (EM), EM initialized with the method of moments result described in Section 2 after projecting the estimated parameters into simplex (EM&SVD), and the recently published symmetric tensor decomposition method (STD) (Anandkumar et al., 2012b). To evaluate the performance gain due to exterior point algorithm, we also included results from method of moments without the additional optimization (SVD) Section 2.

To ensure a fair time comparison, all of the methods were implemented in Matlab. In all methods, the iteration was stopped whenever the change in $\frac{obj^{(t-1)} - obj^{(t)}}{|avg(obj^{(t)}, obj^{(t-1)})|}$ was less than δ. We set parameter δ in EM-based approaches, and exterior point algorithm to 10^{-4} (Murphy; Parikh et al., 2012), and 10^{-3} respectively.

Parameters λ_1, and λ_2 controls the speed of mapping parameters into the simplex while estimation error term is optimized simultaneously. We find the best parameters using cross-validation. In our experiments, we sample N training and M testing points from each model. For the evaluation we use $M = 2000$ test samples and calculate normalized ℓ_1 *error* $= \frac{1}{M} \sum_{i=1}^{M} \frac{|\mathbb{P}(X_i) - \hat{\mathbb{P}}(X_i)|}{\mathbb{P}(X_i)}$.

We found that, empirically, spectral methods and exterior point algorithm outperform EM for small sample sizes. In these situations, we believe that EM is overfitting, resulting in poor performance on the test dataset. Similar results are also reported in (Parikh et al., 2012). As the number of training data points increases, EM begins to outperform the spectral methods. However, our experiments show that EX&SVD constantly outperforms other methods in terms of estimation error while remaining an order of magnitude faster than EM.

It is important to note that the gap between estimation error of EX&SVD and EM&SVD is considerably larger in the situations where the number of training data points is relatively small compared to the number of model parameters. In these situations, estimation error in the SVD method is not accurate and the error of projection into the simplex is relatively high in EM&SVD method. However, when the SVD parameter estimates are used to initialize our exterior point algorithm we get considerably better parameter estimates. When the SVD estimate of the parameters is accurate (due to the large training set and small number of parameters) EM&SVD and EX&SVD estimations are close to each other. We believe that this observation strongly supports our approach to use exterior point method to find a set of parameters in the valid set of models (rather than a naive projection).

4.1 MULTI-VIEW MODEL EXPERIMENTS

To test the performance of the proposed method on learning multi-view models in different settings, we generate observations from randomly sampled models. The first set of models has 5 hidden states and 10 discrete observations per view, and the second set of models has 10 hidden states and 20 observations per view.

Figure 3 shows the average error of the implemented algorithms run on 10 different datasets generated by *i.i.d* sampled models. Each dataset consisted of up to 100,000 triples of observations sampled from each model. We used log-log scale for better demonstration of results. In these experiments, EM is initialized with 10 different random seeds and the best model is reported. Both EM&SVD and EX&SVD are initialized with 1 sample from our SVD decomposition method. As discussed earlier, EM outperforms SVD and the tensor decomposition method with respect to estimation error as the number of training samples increases. However, both EX&SVD and EM&SVD outperform EM, which shows the effectiveness of using the method of moments result as an initial seed for optimization. The performance of the EX&SVD method is significantly better especially in the small sample size region where the method of moments result is far from the simplex and projection step in EM&SVD method change the value of initial seed a lot. As the number of training samples increases, the error induced by the projection decreases and the results of the two methods converge.

Comparing the results from the two model classes, we see that the difference between the performance of EX&SVD and other methods is more pronounced as the number of parameters increases. This is due to the fact that the error of SVD increases as the number of parameters increases; which, in turn, is due to poor population estimates of the moments. The error of projecting the SVD result into the simplex is also high in the EM&SVD method. As illustrated in Figure 3, the result of SVD is comparable to STD while SVD is orders of magnitude faster. Considering the large number of parameters in this experiment, it is not strange that both method of moments algorithms (STD and SVD) do not show a good performance, however, both EM&SVD and EX&SVD outperform EM which shows the method of moments estimation are a better initialization point than a random selection.

To study the performance of different methods under different parameter set sizes in more detail, we investigate the performance of the different algorithms in estimating parameters of models with different numbers of hidden states. To this end, we sample $N = 4000$ training points from models with different numbers of hidden states and evaluate the performance of different method in estimating these models parameters. The average error of 10 independent runs is reported in figure 4 for different values of n_s. In

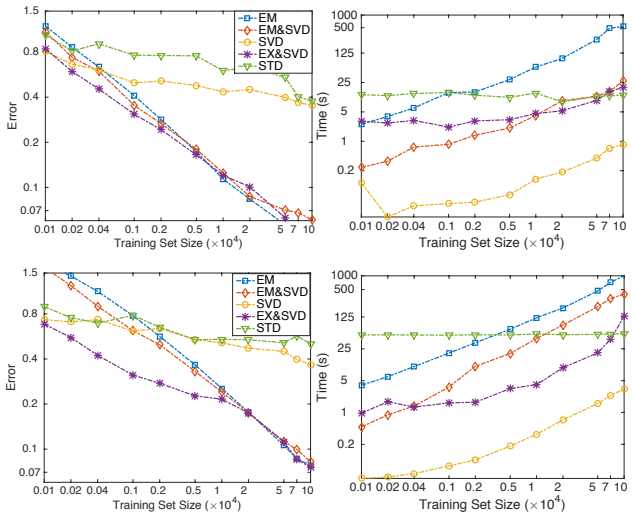


Figure 3: Error vs. #training (first column), and Time vs. #training (second column) for multi-view models. $n_s = 5$, $n_o = 10$ in the first row, and $n_s = 10$, $n_o = 20$ in the second row.

each case we set n_o to twice the value of n_s. As n_s increases, the number of model parameters also increases while the number of training points remains fixed. This results in the estimation error increasing as the models get larger for all of the methods. However, the difference between the performance of EX&SVD and other methods becomes more pronounced with n_s, which shows the effectiveness of our method in learning models with large state spaces and relatively smaller datasets.

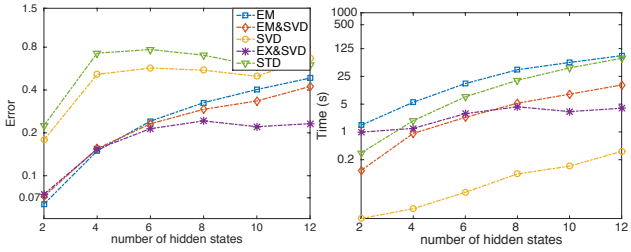


Figure 4: Error vs. n_s (left), time vs. n_s (right) for multi-view model for #training = 4000.

4.2 HIDDEN MARKOV MODEL EXPERIMENTS

We also evaluate the performance of our algorithm by estimating the parameters of HMMs on synthetic and real-world datasets. Similar to the multi-view case, we randomly sample parameters from two different classes of models to generate synthetic datasets. The first set of models again has 5 hidden states and 10 discrete observations, and the second set of models has 10 hidden states and 20 observations. Figure 5 shows the average error of the implemented algorithms run on 10 different datasets generated by *i.i.d* sampled models. Each dataset consisted of

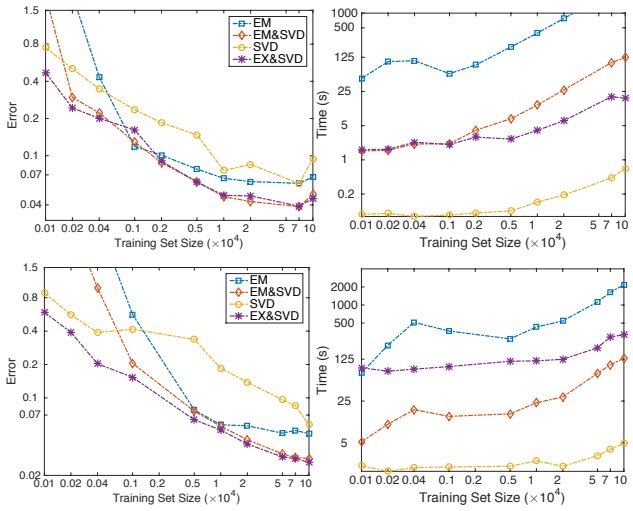


Figure 5: Error vs. #training (first column), and time vs. #training (second column) for HMM models. $n_s = 5$, $n_o = 10$ (first row), and $n_s = 10$, $n_o = 20$ (second row).

up to 100,000 triples of observations sampled from each model. Although, estimated parameters in the SVD method do not have good performance in terms of normalized $l1$ error, using them to initiate an iterative optimization procedure improves performance as demonstrated by both the EM&SVD and the EX&SVD methods. On the other hand, our algorithm can also outperform EM&SVD, especially in the low and medium sample size regions when the error of projection step is relatively high. For medium and large training set sizes, EM&SVD and EX&SVD are almost the same speed, and both are considerably faster than EM alone.

4.2.1 Splice Dataset

In this experiment we consider the task of recognizing splice sites on a DNA sequence (Bache and Lichman, 2013). The dataset consist of 3190 examples. Each example is a sequence of 60 fields in which every field is filled by either A,T,C, or G. The label of each example could be Intron/Exon site, Exon/Intron site, or neither. For training, we train a HMM with $n_s = 4$ for each class using different methods and use the rest of examples for testing. For each test example we compute the probability of the sequence for each model, and choose the label corresponding to the model with the highest test probability. For each test example in our method, we compute the histogram of triples in the test sequence, and choose the label corresponding to the model whose probability distribution over different triples has the lowest ℓ_1 distance to the computed histogram. This method of classification is a natural fit for our method, since our optimization method finds model parameters such that its probability distribution over different triples has minimum distance to the empirically estimated distribution of tripes $\hat{\mathcal{M}}$. Figure 6 shows the average classification error of each method for 10 randomly

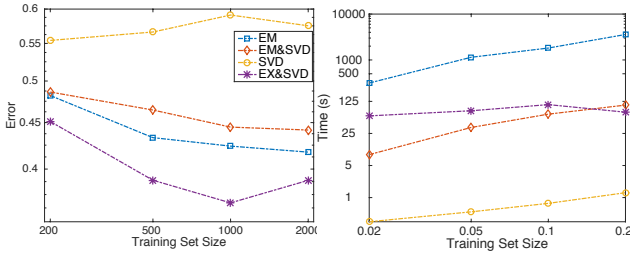


Figure 6: Error vs. #training (left), time vs. #training (right) for splice dataset.

chosen training set with several different sized training sets. The results are consistent with the experiments on the synthetic datasets in terms of speed and accuracy, despite EM outperforms EM&SVD in real world dataset. However, our method performs considerably better than EM.

5 CONCLUSION

We present a new approach to learning latent variable models such as multi-view models and HMMs. Recent work on learning such models by method of moments has produced exciting theoretical results that explicitly bound the error in the learned model parameters. Unfortunately, these results have failed to translate into accurate and numerically robust algorithms in practice. In particular, the parameters learned by method of moments may lie outside of the feasible set of models. This is especially likely to happen when the population moments are estimated inaccurately from small quantities of training data. To overcome this problem, we propose a two-stage algorithm for learning the parameters of latent variable models. In the first stage, we learn an initial estimate of the parameters by method of moments. In the second stage, we use an exterior point algorithm that incrementally refines the solution until the parameters are at a local optima and arbitrarily close to valid model parameters. We prove convergence of the method and perform several experiments to compare our method to previous work. An empirical evaluation on both synthetic and real-world datasets demonstrates that our algorithm learns models that are generally more accurate than method of moments or EM alone. By elegantly contending with parameters that may be outside of the model class, we are able to learn models that are much more accurate than EM initialized with method of moments when only a limited amount of training data is available.

Acknowledgement

The research was supported in part by NSF IIS-1116886, NSF/NIH BIGDATA 1R01GM108341, and NSF CAREER IIS-1350983.

References

A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y. Liu. Two svds suffice: Spectral decompositions for probabilistic topic modeling and latent dirichlet allocation. *CoRR*, abs/1204.6703, 2012a.

A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *arXiv preprint arXiv:1210.7559*, 2012b.

A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden markov models. In *Proc. Annual Conf. Computational Learning Theory*, pages 33.1–33.34, 2012c.

K. Bache and M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

B. Balle, A. Quattoni, and X. Carreras. Local loss optimization in operator models: A new insight into spectral learning. In *Proceedings of the International Conference on Machine Learning*, 2012.

B. Balle, W. Hamilton, and J. Pineau. Methods of moments for learning stochastic languages: Unified presentation and empirical comparison. In *Proceedings of the International Conference on Machine Learning*, pages 1386–1394, 2014.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, second edition, 1999.

V. J. Bloom. *Exterior-Point Algorithms for Solving Large-Scale Nonlinear Optimization Problems*. PhD thesis, George Mason University, 2014.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, England, 2004.

C. Byrne. Sequential unconstrained minimization algorithms for constrained optimization. *Inverse Problems*, 24(1), 2008.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*, 2013.

P. L. Combettes and J. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–22, 1977.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandrae. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the International Conference on Machine Learning*, 2008.

R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

D. Hsu and S.M. Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions, 2012. URL arXiv:1206.5766.

D. Hsu, S. Kakade, and T. Zhang. A spectral algorithm for learning hidden markov models. In *Proc. Annual Conf. Computational Learning Theory*, 2009.

L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *Bell System Technical Journal*, 62(4):1035–1074, April 1983.

K. Murphy. Hidden markov model (hmm) toolbox for matlab. URL https://github.com/probml/pmtk3.

A. Parikh, L. Song, and E. P. Xing. A spectral algorithm for latent tree graphical models. In *Proceedings of the International Conference on Machine Learning*, 2011.

A. P. Parikh, L. Song, M. Ishteva, G. Teodoru, and E.P. Xing. A spectral algorithm for latent junction trees. In *Conference on Uncertainty in Artificial Intelligence*, 2012.

K. Pearson. Contributions to the mathematical theory of evolution. *Transactions of the Royal Society of London*, 185:71–110, 1894.

R. A Polyak. Primal–dual exterior point method for convex optimization. *Optimisation Methods and Software*, 23 (1):141–160, 2008.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, pages 1–41, 2013.

S. Siddiqi, B. Boots, and G. J. Gordon. Reduced-rank hidden Markov models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-2010)*, 2010.

L. Song, B. Boots, S. Siddiqi, G. Gordon, and A. J. Smola. Hilbert space embeddings of hidden markov models. In *International Conference on Machine Learning*, 2010.

L. Song, A. Anamdakumar, B. Dai, and B. Xie. Nonparametric estimation of multi-view latent variable models. In *International Conference on Machine Learning (ICML)*, 2014.

H. Yamashita and T. Tanabe. A primal-dual exterior point method for nonlinear optimization. *SIAM Journal on Optimization*, 20(6):3335–3363, 2010.

Y. Zhang, X. Chen, D. Zhou, and M. I Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. In *Advances in Neural Information Processing Systems 14*, pages 1260–1268, 2014.

5.1 Appendix

For ease of notation we define another operator on the vector $\boldsymbol{v}$. Vector $\boldsymbol{v}$ is comprised of $3n_o + 1$ different probability distributions of the multi-view model (each of $\boldsymbol{U}^t$ columns and $diag(\mathcal{H})$). $\boldsymbol{v}_{[l]}$ shows the l^{th} probability vector of the model. Using this definition for function $h(\boldsymbol{v}) = \frac{1}{2}||\text{s}(\boldsymbol{v}) - \mathbf{1}||_2^2$, one can show:

$$(\nabla h(\boldsymbol{v}))_i = (\text{sum}(\boldsymbol{v}_{[l]}) - 1), \tag{21}$$

where $\boldsymbol{v}_{[l]}$ is the probability distribution vector that $\boldsymbol{v}_i$ belongs to it. We use these definitions in the proof of following theorems.

Lemma 6. *Assuming that sequence $\{\boldsymbol{v}^{(k)}\}$ produced by Algorithm 2 is in the set $\mathscr{F}$, and λ_2 is selected such that:*

$$\lambda_2 > L^3\Upsilon + \lambda_1(\sqrt{n_o}L + 1), \tag{22}$$

there is a constant K such that for $k > K$ we have $|\boldsymbol{v}^{(k)}|_- = 0$. Also, for $k > K$ the proximal operator in Algorithm 2 reduces to the orthogonal projection operator into the convex set $\mathscr{C} = \mathbb{R}_+^{n_s \times (3n_o+1)}$:

$$\forall k > K : prox(\tilde{\boldsymbol{v}}^{(k)}) = proj_{\mathscr{C}}(\tilde{\boldsymbol{v}}^{(k)}) \tag{23}$$

Proof. For each entry of $\boldsymbol{v}^{(k-1)}$ one of these situations will happen:

$$\text{if } (\boldsymbol{v}^{(k-1)})_i \geq 0 \text{ then } |(\boldsymbol{v}^{(k)})_i|_- = 0, \text{ else} \tag{24}$$

$$\text{either } (\boldsymbol{v}^{(k-1)})_i < 0 \text{ and } |(\boldsymbol{v}^{(k)})_i|_- = 0, \text{or} \tag{25}$$

$$(\boldsymbol{v}^{(k-1)})_i < 0 \text{ and } |(\boldsymbol{v}^{(k)})_i|_- < |(\boldsymbol{v}^{(k-1)})_i|_- - c\epsilon \tag{26}$$

To show the above predicates we first need to bound entries of $\nabla g(\boldsymbol{v})$ by λ_2. Using Corollary 5 :

$$\begin{aligned}
&\forall k > 0 : |\frac{\partial g(\boldsymbol{v}^{(k)})}{\partial (\boldsymbol{v}^{(k)})_i}| \\
&\leq |\frac{\partial r(\boldsymbol{v}^{(k)})}{\partial (\boldsymbol{v}^{(k)})_i}| + \frac{\lambda_1}{2}|\frac{\partial (||sum(\boldsymbol{v}^{(k)}) - \mathbf{1}||_2^2)}{\partial (\boldsymbol{v}^{(k)})_i}| \\
&\leq L^3||\mathcal{R}(\boldsymbol{v}^{(k)})||_F + \lambda_1(\sqrt{n_o}L + 1),
\end{aligned}$$

where in the last step we used Equation (16) to bound the first term, and following to bound the second term:

$$\begin{aligned}
|\frac{\partial (||\text{s}(\boldsymbol{v}^{(k)}) - \mathbf{1}||_2^2)}{\partial (\boldsymbol{v}^{(k)})_i}| &= 2|\text{sum}(\boldsymbol{v}_{[l]}) - 1| \\
&\leq 2(|\text{sum}(\boldsymbol{v}_{[l]})| + 1) \leq 2(\sqrt{n_o}L + 1),
\end{aligned}$$

Thus, having the lower bound for λ_2 in Equation (22) we have:

$$\lambda_2 - \sup_k |(\nabla g(\boldsymbol{v}^{(k)}))_i| > \epsilon \tag{27}$$

Now, we will show statement (24) is correct. First, assuming $(\boldsymbol{v}^{(k-1)})_i \geq 0$ we show $(\boldsymbol{v}^{(k)})_i$ is also non-negative: According to Equation (27) we have $\alpha_k\lambda_2 - \alpha_k \frac{\partial g(\boldsymbol{v}^{(k-1)})}{\partial (\boldsymbol{v}^{(k-1)})_i} \geq 0$ for any $\alpha_k \geq 0$. Thus, since $(\boldsymbol{v}^{(k-1)})_i \geq 0$:

$$(\tilde{\boldsymbol{v}}^{(k)})_i = (\boldsymbol{v}^{(k-1)})_i - \alpha_k \frac{\partial g(\boldsymbol{v}^{(k-1)})}{\partial (\boldsymbol{v}^{(k-1)})_i} \geq -\alpha_k\lambda_2 \tag{28}$$

Thus, only the second and the third rules of proximal updates in Equation (15) can occur for $(\tilde{\boldsymbol{v}}^{(k)})_i$. Either way, $(\boldsymbol{v}^{(k)})_i \geq 0$ and thus $|(\boldsymbol{v}^{(k)})_i|_- = 0$.

Next, we proceed to prove predicates (25) and (26) which means that for the negative entry $(\boldsymbol{v}^{(k-1)})_i$ doing one step of forward-backward splitting algorithm will either makes it non-negative (25) or its value shrinks at least by $c\epsilon$ (26). If $(\boldsymbol{v}^{(k)})_i \geq 0$, then we are done with (25) since this means $|(\boldsymbol{v}^{(k)})_i|_- = 0$. Otherwise, we will prove correctness of

Equation (26). Since $(\boldsymbol{v}^{(k)})_i < 0$ we have

$$
\begin{aligned}
&|(\boldsymbol{v}^{(k-1)})_i|_- - |(\boldsymbol{v}^{(k)})_i|_- = -(\boldsymbol{v}^{(k-1)})_i + (\boldsymbol{v}^{(k)})_i \\
&= -(\boldsymbol{v}^{(k-1)})_i + (\boldsymbol{v}^{(k-1)})_i - \alpha_k \frac{\partial g(\boldsymbol{v}^{(k-1)})}{\partial (\boldsymbol{v}^{(k-1)})_i} + \alpha_k \lambda_2 \\
&= \alpha_k \left(\lambda_2 - \frac{\partial g(\boldsymbol{v}^{(k-1)})}{\partial (\boldsymbol{v}^{(k-1)})_i} \right) > c\epsilon
\end{aligned}
$$

where ϵ and α_k are defined in Equation (27) and Algorithm 2 respectively. Thus, Equation (26) has been proved.

Now we interpret the results in Equations (24), (25), and (26). Equation (24) tells us that if $(\boldsymbol{v}^{(k)})_i$ becomes positive it remains positive forever. If it is negative, then at the next step either it becomes positive or becomes closer to 0 by $c\epsilon$. Thus, since

$$
L \geq \max \left\{ |(\boldsymbol{v}^{(0)})_i| : (\boldsymbol{v}^{(0)})_i < 0 \right\} \tag{29}
$$

then after at most $K = \frac{L}{c\epsilon}$ steps we have $(\boldsymbol{v}^{(k)})_i \geq 0$ for all i.

Furthermore, Equation (24) tells us after $k > K$ all the proximal updates maintain the non-negativity condition, which means that the first rule in (15) will not happen for these ks. Therefore, proximal updates reduce to the orthogonal projection into the set $\mathscr{C}$. □

Theorem 7. *For every $\epsilon_1 > 0$, set $\lambda_1 > \frac{L^3 \Upsilon}{\epsilon_1}$ and $\lambda_2 > L^3 \Upsilon + \lambda_1(\sqrt{n_o}L + 1)$, in Equation* (12)*. For the convergence point of the sequence $\{\boldsymbol{v}^{(k)}\} \subset \mathscr{F}$ which is generated by Algorithm 2 we have:*

$$
\begin{aligned}
|\boldsymbol{v}^*|_- &= 0 \\
||s(\boldsymbol{v}^*) - \mathbf{1}||_2 &\leq \epsilon_1
\end{aligned} \tag{30}
$$

Proof. As we discussed earlier, according to Lemma 6, if we set $\lambda_2 > L^3 \Upsilon + \lambda_1(\sqrt{n_o}L + 1)$, then we will eventually arrive to the region where $|\boldsymbol{v}^*|_- = 0$, and after that the algorithm will behave similarly to gradient projection method which is guaranteed to converge with an appropriate sequence of step sizes $\{\beta_k\}$. Next, by setting $\epsilon_1 \geq 0$ we will prove that the condition $||\mathrm{s}(\boldsymbol{v}^*) - \mathbf{1}||_2 \leq \epsilon_1$ holds when the algorithm converges. For a local optimum of function $g(\boldsymbol{v})$ with the constraint $\boldsymbol{v} \in \mathbb{R}^{n_s(3n_o+1)}$ we have:

$$
\forall\, \boldsymbol{y} \in \mathbb{R}^{n_s(3n_o+1)},\ \gamma > 0 : \left(\boldsymbol{v}^* + \gamma \boldsymbol{y} \in \mathbb{R}_+^{n_s(3n_o+1)} \Longrightarrow \langle \boldsymbol{y}, \nabla g(\boldsymbol{v}^*) \rangle \geq 0 \right) \tag{31}
$$

We proceed by a proof by contradiction. For the sake of contradiction, assume that when the algorithm converges $||\mathrm{s}(\boldsymbol{v}^*) - \mathbf{1}||_2 > \epsilon_1$. If we show for this $\boldsymbol{v}^*$ there are $\boldsymbol{y}$ and γ such that $\boldsymbol{v}^* + \gamma \boldsymbol{y} \in \mathbb{R}_+^{n_s(3n_o+1)}$ but $\langle \boldsymbol{y}, \nabla g(\boldsymbol{v}^*) \rangle < 0$ will result in a contradiction.

We can show function $g(.)$ as the summation of defined residual function $r(\cdot)$ and function $h(\cdot)$:

$$
g(\boldsymbol{v}) = r(\boldsymbol{v}) + \lambda_1 h(\boldsymbol{v}). \tag{32}
$$

$\boldsymbol{y}$ is initially set to $-\nabla h(\boldsymbol{v}^*)$. Equation (21) shows that for every i that belongs to the l^{th} part of the vector $\boldsymbol{v}$, vector $\boldsymbol{y}$ has equal entries. For parts that the entries of $\boldsymbol{y}$ are positive, we randomly keep one element unchanged and set all others element in that part to 0. For parts in which all the elements are negative, there must be a positive element in the corresponding part of $\boldsymbol{v}^*$ since the sum of that part in $\boldsymbol{v}^*$ have to become positive to a get positive gradient in Equation (21). We keep the corresponding element of that positive element unchanged in $\boldsymbol{y}$ and set all other elements of that part to 0. With the above definition of $\boldsymbol{y}$ if we set $\gamma = \min \{|\boldsymbol{y}_i| : \boldsymbol{y}_i \neq 0\}$, we have $\boldsymbol{v}^* + \gamma \boldsymbol{y} \in \mathbb{R}_+^{n_s(3n_o+1)}$.

Before proceed let us briefly state two useful facts about $\boldsymbol{y}$. Since in each part of $\boldsymbol{y}$ we only leave one element unchanged and absolute value of the these entries are equal to absolute value of the corresponding entries of $\nabla h(\boldsymbol{v}^*)$ we have:

$$
||\boldsymbol{y}||_2 = ||\mathrm{s}(\boldsymbol{v}^*) - \mathbf{1}||_2 \tag{33}
$$

Also, we will further use the property:

$$
\langle \boldsymbol{y}, \nabla h(\boldsymbol{y}^*) \rangle = -||\boldsymbol{y}||_2^2 \tag{34}
$$

Now, for chosen $\boldsymbol{y}$ and γ we evaluate $\langle \boldsymbol{y}, \nabla g(\boldsymbol{v}^*) \rangle$:

$$
0 \leq \langle \boldsymbol{y}, \nabla g(\boldsymbol{v}^*) \rangle = \langle \boldsymbol{y}, \nabla r(\boldsymbol{v}^*) \rangle + \lambda_1 \langle \boldsymbol{y}, \nabla h(\boldsymbol{v}^*) \rangle
$$

Using Cauchy-Schwarz inequality and Equation (34):

$$\begin{aligned} 0 &\leq ||\nabla r(\boldsymbol{v}^*)||_2||\boldsymbol{y}||_2 - \lambda_1||\boldsymbol{y}||_2^2 \\ &= ||\boldsymbol{y}||_2 \left(||\nabla r(\boldsymbol{v}^*)||_2 - \lambda_1||\boldsymbol{y}||_2\right) \end{aligned}$$

By using Equation (16) and Equation (33) and the fact that $||\boldsymbol{y}||_2 > 0$ we get:

$$L^3\Upsilon - \lambda_1||\mathrm{s}(\boldsymbol{v}^*) - \mathbf{1}||_2 \geq 0, \tag{35}$$

which by using the assumption that $||\mathrm{s}(\boldsymbol{v}^*) - \mathbf{1}||_2 > \epsilon_1$ it reduces to $\lambda_1 \leq \frac{L^3\Upsilon}{\epsilon_1}$ which is contradicting with the way we chose λ_1. □