

# **Estimating Mixture Models via Mixtures of Polynomials**

Sida I. Wang Arun Tejasvi Chaganty Percy Liang Department of Computer Science Stanford University

4. Recover parameter moments (v)



#### **Motivation:** estimate mixture models

- ► Mixture models are important in ML with many applications
- ▶ We have no general and tractable methods for parameter estimation in mixture models ▶ EM is generally applicable, but prone to local optima
- ▶ method of moments can sometimes provide global guarantees, but difficult to use
- ▶ We want to generalize the class of mixture models solvable by the method of moments.

Method	Model	Result			
maximum-likelihood	any	intractable for most mixture models			
expectation-maximization	latent variables	convergence to local min.			
solving moment equations	case by case	moment matching			
tensor factorization	tensor structured models	moment matching			
polymom (this work)	polynomial structure	moment matching			

## Contributions: more general. unifying, and turnkey

- ► More general: we can estimate any mixture model with polynomial moments
- ▶ More general than tensor structured models
- ▶ Includes mixtures of Gaussians, Poissons, and linear regressions. Also multiview mixtures.
- ▶ Unifying: the same algorithm can be used in both Pearson's mixture of 2 Gaussians (need high order moments) in 1D and high dimensional mixtures (low order moments).
- ► Naturally supports parameter sharing, like EM
- ► **Turnkey**: the user just needs to provide coefficients of some polynomials
- ► Connecting estimating mixture models to polynomial optimization and computer algebra

#### Problem setup: the model class we want to mix (user specified)

- $\triangleright p(\mathbf{x}; \boldsymbol{\theta})$  is a distribution parameterized by  $\boldsymbol{\theta}$ .
- ▶ Moments  $\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}; \boldsymbol{\theta})}[\phi(\mathbf{x})]$  of  $p(\mathbf{x}; \boldsymbol{\theta})$  can be expressed in terms of parameters  $\boldsymbol{\theta}$ .  $\triangleright$  For example  $\phi(\mathbf{x}) = x_1$ , or  $x_1^2 x_3$ ,  $\log(x_1)$ ,  $\sin(x_1)$
- ► The scope of this work is when these moments are polynomials

$$f( heta) riangleq \mathbb{E}_{\mathsf{x} \sim p(\mathsf{x}; heta)}[\phi(\mathsf{x})] = \sum_{lpha} a_{lpha} heta^{lpha}, ext{ where } heta^{lpha} = \prod_{p=1}^P heta_p^{lpha_p}.$$

 $\blacktriangleright$  The choice of  $\phi(\mathbf{x})$  and the coefficients of  $f(\theta)$  are both model dependent and user **specified**. Choosing a suitable set of  $\phi(x)$  and findings the corresponding coefficients is what Polymom needs as inputs.

# Example

For the Gaussian distribution in 1d with mean  $\xi$  and varaince  $\sigma^2$ . The observation functions  $\phi(x) = [x^1, \dots, x^6, x^7]$  corresponds to polynomials  $f(\theta) = [\xi, \xi^2 + \sigma^2, \xi^3 + 3\xi\sigma^2, \dots]$ Note that this is for a single component to be mixed.

#### Problem setup: the mixture model

Given the component model  $p(x; \theta)$ , we can defined the corresponding mixture model, where each data point  $\mathbf{x} \in \mathbb{R}^D$  is associated with a latent component  $z \in [K]$ 

$$z \sim \text{Multinomial}(\pi), \quad \mathbf{x} \mid z \sim p(\mathbf{x}; \theta_z),$$
 (2)

where  $\pi = (\pi_1, \dots, \pi_K)$  are the mixing coefficients,  $\theta_{k}^* \in \mathbb{R}^P$  are the true model parameters for the  $k^{\text{th}}$  mixture component, and  $\mathbf{x} \in \mathbb{R}^D$  is the data.

## From a component to its mixture

For each **component**, we have polynomials expressions for observation functions  $\phi(\mathbf{x})$ .

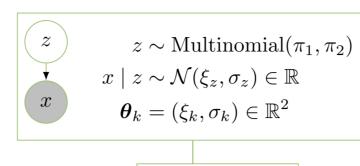
$$f(\theta) \triangleq \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}; \theta)}[\phi(\mathbf{x})] = \sum_{\alpha} a_{\alpha} \theta^{\alpha}$$
 (3)

The moments for the entire **mixture** is

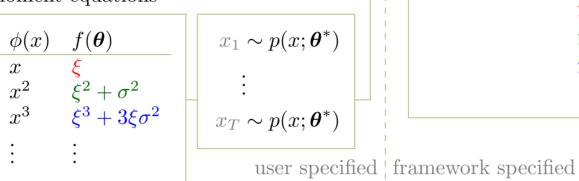
$$\mathbb{E}[\phi(\mathbf{x})] = \sum_{k=1}^K \pi_k \mathbb{E}[\phi(\mathbf{x})|z=k] = \sum_{k=1}^K \pi_k f(\theta_k). \tag{4}$$

- ► Solving this system of polynomial equations is the task of parameter estimation using method of moments.
- ▶ Unfortunately, pairtially due to its symmetries (K! solutions), this is a hard polynomial system to solve even when it is easy to solve the single component case.

#### 1. Write down a mixture model



2. Derive single mixtur. Add data moment equations

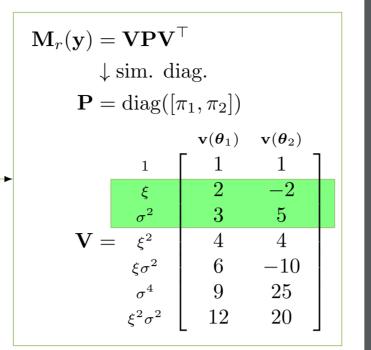


 $\begin{bmatrix} y_{0,1} & y_{1,1} & y_{2,1} & y_{0,2} \end{bmatrix}$ minimize  $\operatorname{tr}(\mathbf{M}_r(\mathbf{y}))$ 

 $\mathbf{M}_r(\mathbf{y}) \succeq 0, \ y_{0,0} = 1$  $\mathbf{y}_{1,0} = \frac{1}{T} \sum_{t}^{T} x_t$  $y_{2,0} + y_{0,1} = \frac{1}{T} \sum_{t=0}^{T} x_t^2$  $y_{3,0} + 3y_{1,1} = \frac{1}{T} \sum_{t=0}^{T} x_t^3$ 

 $y_{0,0}$   $y_{1,0}$   $y_{2,0}$   $y_{0,1}$ 

5. Solve for parameters



## Conceptual idea: get rid of symmetries by working with moments of parameters

The key idea of Polymom is to exploit the mixture structure of the moment equations (4). Specifically, let  $\mu^*$  be a particular "mixture" over the component parameters  $\theta_1^*,\ldots,\theta_k^*$  (i.e.  $\mu^*$  is a probability measure). Then we can express the moment conditions (4) in terms of  $\mu^*$ 

$$\mathbb{E}[\phi_n(\mathbf{x})] = \int f_n(\theta) \ \mu^*(d\theta), \text{ where } \mu^*(\theta) = \sum_{k=1}^K \pi_k \delta(\theta - \theta_k^*). \tag{5}$$

The following feasibility problem is equivalent to the moment conditions in (4)

find 
$$\mu \in \mathcal{M}_{+}(\mathbb{R}^{P})$$
, the set of probability measures over  $\mathbb{R}^{P}$  s.t.  $\int f_{n}(\theta) \ \mu(d\theta) = \mathbb{E}[\phi_{n}(\mathbf{x})], \quad n = 1, \dots, N$  (6)  $\mu$  is  $K$ -atomic (i.e. sum of  $K$  deltas),

where we deliberately "forget" the permutation of the components by using  $\mu$  to represent the problem instead of  $[\theta_1, \ldots, \theta_K]$ .

## Moment Completion: making things tractable

The Generalized Moment Problem framework allows us to work with measures by working with their moments using semidefinite programming. Let  $\mathscr{L}_{\mathbf{y}}(\theta^{\alpha}) \triangleq y_{\alpha}$ , and we will work with the moment sequence  $\mathbf{y} = (y_{\alpha})_{\alpha \in \mathbb{N}^{P}}$ . Note that these moments **y** are moments in the **parameter space** (like  $y_{\alpha} = \mathbb{E}_{\mu}[\theta^{\alpha}]$ ), and not of the data as before (like  $\mathbb{E}_{p(\mathbf{x};\theta^{*})}[\phi_{n}(\mathbf{x})]$ ).

find 
$$\mathbf{y} \in \mathbb{R}^{\mathbb{N}}$$
 (or equivalently, find  $\mathbf{M}(\mathbf{y})$ )  
s.t.  $\sum_{\alpha} a_{n\alpha} y_{\alpha} = \mathbb{E}[\phi_{n}(\mathbf{x})], \quad n = 1, \dots, N$   
 $\mathbf{M}_{r}(\mathbf{y}) \succeq \mathbf{0}, \quad y_{0} = \mathbf{1}$   
 $\operatorname{rank}(\mathbf{M}_{r}(\mathbf{y})) = K \text{ and } \operatorname{rank}(\mathbf{M}_{r-1}(\mathbf{y})) = K.$  (7)

Unfortunately, we still cannot deal with rank constraints, but the following relaxation (a semidefinite program) is tractable:

minimize 
$$\operatorname{tr}(CM_r(y))$$
  
s.t.  $\sum_{\alpha} a_{n\alpha} y_{\alpha} = \mathbb{E}[\phi_n(x)], \quad n = 1, \dots, N$   
 $M_r(y) \succeq 0, \quad y_0 = 1$  (8)

- For some models like multiview mixture and mixture of linear regressions, the linear constraints might fully determines y and we do not need to solve an SDP. In such cases, Polymom provides an unifying view and some guarantees.
- $\triangleright$  After obtaining y, there are several generic ways to extract  $\theta$  based on solving some kind of eigenvalue problem.

## **Experimental results**

	Methd.	EM	TF	Poly	EM	TF	Poly	EM	TF	Poly
Gaussians	K, D	$T = 10^3$			$T = 10^4$			$T = 10^5$		
spherical	2, 2	0.37	2.05	0.58	0.24	0.73	0.29	0.19	0.36	0.14
diagonal	2, 2	0.44	2.15	0.48	0.48	4.03	0.40	0.38	2.46	0.35
constrained	2, 2	0.49	7.52	0.38	0.47	2.56	0.30	0.34	3.02	0.29
Others	K,D	$T=10^4$		$T = 10^5$		$T = 10^6$				
3-view	3, 3	0.38	0.51	0.57	0.31	0.33	0.26	0.36	0.16	0.12
lin. reg.	2, 2	_	-	3.51	_	-	2.60	-	-	2.52

Table: T is the number of samples. **Methods:** EM: sklearn initialized with k-means using 5 random restarts; TF: tensor power method implemented in Python; Poly: Polymom by solving Problem 8. **Models:** for mixture of Gaussians, we have  $\sigma \approx 2||\mu_1 - \mu_2||_2$ ; 'spherical' and 'diagonal' describes the type of covariance matrix. The mean parameters of constrained Gaussians satisfies  $\mu_1 + \mu_2 = 1$ . The best result is **bolded**. TF only handles spherical variance, but it was of interest to see what TF does if the data is drawn from mixture of Gaussians with diagonal covariance, these results are in strikeout.

## **Examples:** what the user needs to provide Polymom

#### Table: Applications of the Polymom framework.

## Mixture of linear regressions

#### Model

 $\mathbf{x} = [x, v]$  is observed where  $x \in \mathbb{R}^D$  is drawn from an unspecified distribution and  $v \sim \mathcal{N}(\mathbf{w} \cdot \mathbf{x}, \sigma^2 I)$ , and  $\sigma^2$ known. The parameters are  $\theta_k^* = (\mathbf{w}_k) \in \mathbb{R}^D$ .

#### **Observation functions**

 $\phi_{\alpha,b}(\mathbf{x}) = \mathbf{x}^{\alpha} \mathbf{v}^{b}$  for  $\mathbf{0} \le |\alpha| \le 3, b \in [2]$ .

Moment polynomials

 $f_{\alpha,1}(\theta) = \sum_{p=1}^{P} \mathbb{E}[x^{\alpha+\gamma_p}] w_p$  $f_{\alpha,2}(\theta) = \mathbb{E}[x^{\alpha}]\sigma^2 + \sum_{p,q=1}^{P} \mathbb{E}[x^{\alpha}x_px_q]w_pw_q$ , where the  $\gamma_p \in \mathbb{N}^P$  is **1** in position p and **0** elsewhere.

#### **Mixture of Gaussians**

#### Model

 $\mathbf{x} \in \mathbb{R}^D$  is observed where  $\mathbf{x}$  is drawn from a Gaussian with diagonal covariance:  $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\xi}, \mathbf{diag(c)})$ . The parameters are  $\theta_k^* = (\xi_k, \mathbf{c}_k) \in \mathbb{R}^{D+D}$ 

**Observation functions** 

 $\phi_{\alpha}(\mathbf{x}) = \mathbf{x}^{\alpha} \text{ for } 0 \leq |\alpha| \leq 4.$ 

Moment polynomials

 $f_{\alpha}(\theta) = \prod_{d=1}^{D} h_{\alpha_d}(\xi_d, c_d).$ 

## Multiview mixtures

#### Model

With 3 views,  $\mathbf{x} = [x^{(1)}, x^{(2)}, x^{(3)}]$  is observed where  $x^{(1)}, x^{(2)}, x^{(3)} \in \mathbb{R}^D$  and  $x^{(\ell)}$  is drawn from an unspecified **Moment polynomials** distribution with mean  $\xi^{(\ell)}$  for  $\ell \in [3]$ . The parameters are  $\theta_k^* = (\xi_k^{(1)}, \xi_k^{(2)}, \xi_k^{(3)}) \in \mathbb{R}^{D+D+D}$ .

#### **Observation functions**

 $\phi_{ijk}(\mathbf{x}) = x_i^{(1)} x_i^{(2)} x_k^{(3)}$  where  $1 \le i, j, k \le D$ .

 $f_{ijk}(\theta) = \xi_i^{(1)} \xi_i^{(2)} \xi_k^{(3)}$ 

# Discussions, future work, and limitations

- ► Compared to previous literature on this topic, we do not need a 1-to-1 correspondence between the data dimensions and the parameter dimensions. The structure of the model class is encoded in these polynomials that can be derived systematically.
- ightharpoonup Constraints on parameters like  $\xi_1=\xi_2$ , or  $\xi_1^2\geq 3\sigma^2$  translates to tractable linear or semidefinite constraints.
- ► More statistically efficient formulations? For example, the generalized method of moments allows us to model variances in data moments using a weighting matrix W:

minimize 
$$g^{\mathsf{T}}\mathsf{W}g$$
  
s.t.  $g_n = \sum_{\alpha} a_{n\alpha} y_{\alpha} - \mathbb{E}[\phi_n(\mathsf{x})], \quad n = 1, \dots, N$  (9)  
 $\mathsf{M}(\mathsf{y}) \succeq \mathbf{0}$ 

#### Limitations and wishlist

- ▶ Price for generality: Polymom only gives us moment matching solutions, we do not have formal guarantees on consistent parameter estimations except for specific models
- ► Solving big SDPs is still an issue: fragile and scales badly
- ▶ Does this optimization viewpoint give us insight on identifiability?
- ► More than just the mixture structure?

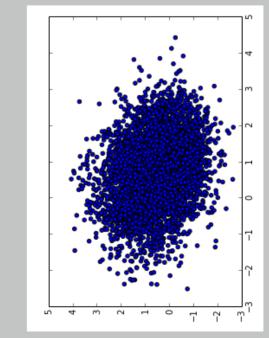
We hope that the Polymom point of view can be a step towards making method of moments more systematic, and more statistically efficient.

# Software

https://github.com/sidaw/polymom

https://github.com/sidaw/mompy: a Generalized Moment Problem package

## Mixture of 2 Gaussians and constraints



 $\xi_1 - 0.7567\xi_2 - 0.5174$  $c_1 + \xi_1^2 - 1.4955$  $3c_2\xi_2+\xi_2^3-2.0393$  $3c_1^2 + 6c_1\xi_1^2 + \xi_1^4 - 5.7752$  $3c_1\xi_1\xi_2 + \xi_1^3\xi_2 - 0.3837$  $c_1c_2 + c_1\xi_2^2 + c_2\xi_1^2 + \xi_1^2\xi_2^2 - 1.6215$  $3c_2\xi_1\xi_2 + \xi_1\xi_2^3 - 0.5527$  $3c_2^2 + 6c_2\xi_2^2 + \xi_2^4 - 5.8174$ 

NLP at Stanford University http://bit.ly/codalab-polymom {sidaw,chaganty,pliang}@cs.stanford.edu