

Simplifying Mixture Models through Function Approximation

Kai Zhang, James T. Kwok

Department of Computer Science and Engineering
The HongKong University of Science and Technology
Kowloon, Hong Kong



香港科技大學

THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

Introduction

Mixture models:

$$f(\mathbf{x}) = \sum_{j=1}^n \alpha_j \phi_j(\mathbf{x})$$

- $\sum_{j=1}^k \alpha_j = 1$
- ϕ : parametric
 - Gaussian \rightarrow Gaussian mixtures
 - Bernoulli, Poisson, etc
- flexible and powerful

Widely used in, e.g.,

- clustering and density modeling (model estimation using the EM algorithm)
- particle filtering, belief propagation

Related to

- Parzen window estimator
- SVM decision function

Large Mixture Models

Problem:

- Large number of components estimation/update is inefficient

Existing methods:

- Spatial data structures
 - kd-tree: [Moore 1998; Kanungo et al 2002]
 - Binning/pre-clustering the data [Hardle and Scott 1992; Babich and Camps 1996]
- Collapsing of mixture models [Goldberger and Roweis 2005]
 - minimize a KL-based distance measure between the two mixtures
 - avoids explicit resampling of data points

Goal

$$f(\mathbf{x}) = \sum_{j=1}^n \alpha_j \phi_j(\mathbf{x})$$

- $\phi_j(\mathbf{x}) = |\mathbf{H}_j|^{-1/2} K_{\mathbf{H}_j}(\mathbf{x} - \mathbf{x}_j)$
- K is a kernel with bandwidth matrix \mathbf{H}_j .
- notation: $K_{\mathbf{H}}(\mathbf{x}) = K(\mathbf{H}^{-1/2}\mathbf{x}) = k(\mathbf{x}'\mathbf{H}^{-1}\mathbf{x})$
- k is called *profile* of (radially symmetric) kernel K , such that $K(\mathbf{x}) = k(\|\mathbf{x}\|^2)$.

Question

How to approximate $f = \sum_{j=1}^n \alpha_j \phi_j$ by $g = \sum_{i=1}^m w_i g_i$, with $m \ll n$?

Basic Algorithm

1. (Component clustering)

Partition the components (ϕ_j 's) into m clusters S_i 's ($m \ll n$)

- The mixture model f is decomposed accordingly into m local models $f = \sum_{i=1}^m f_i$, with

$$f_i(\mathbf{x}) = \sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x}).$$

2. (Local approximation)

Approximate each local model f_i by **one** component $w_i g_i$

$$w_i g_i(\mathbf{x}) = w_i |\tilde{\mathbf{H}}_i|^{-1/2} K_{\tilde{\mathbf{H}}_i}(\mathbf{x} - \mathbf{t}_i)$$

- $w_i \in \mathbb{R}$: weight;
- $\tilde{\mathbf{H}}_i$: bandwidth;
- \mathbf{t}_i : center of g_i

3. Simplified model: $g(\mathbf{x}) = \sum_{i=1}^m w_i g_i(\mathbf{x})$

Component Clustering

Idea: Use **vector quantization** (VQ) to group mixture components ϕ_j 's into clusters

- based on a distance $D(\cdot, \cdot)$ on functions

Problems with VQ:

- sensitive to the initial choice of centers
- can be expensive on large data sets

Sequential Sampling

1. Randomly choose one component $\phi_j \in \mathcal{S}$ and add to the “representative” set \mathcal{R}
2. For each remaining ϕ_j 's in \mathcal{S} , go through \mathcal{R}_i 's in \mathcal{R}
 - Once if $D(\mathcal{R}_i, \phi_j) \leq r$ (predefined threshold r), assign ϕ_j to \mathcal{R}_i , and then process next ϕ_j
 - If no such \mathcal{R}_i exists, ϕ_j is added to \mathcal{R} as a new representative
4. Repeat step 2 until all ϕ_j 's are processed

Component Clustering

In practice, we use sequential sampling to partition the components ϕ_j 's into m clusters, and then perform


Vector Quantization

1. For the i th cluster ($i = 1, 2, \dots, m$), find the prototype $\mathcal{R}_i(\cdot)$ that minimizes the local distortion error $\sum_{j \in \mathcal{S}_i} \alpha_j D(\phi_j, \mathcal{R})$ ¹.
2. Re-assign each ϕ_j to the closest prototype $\mathcal{R}_{\pi(j)}$, where

$$\pi(j) = \arg \min_{i=1,2,\dots,m} D(\phi_j, \mathcal{R}_i).$$

3. Repeat steps 1 and 2 until the quantization error converges,

$$Q = \sum_{i=1}^m \sum_{j \in \mathcal{S}_i} \alpha_j D(\phi_j, \mathcal{R}_{\pi(j)}). \quad (1)$$

¹Such \mathcal{R}_i will satisfy $\mathcal{R}_i = w_i g_i / (\sum_{j \in \mathcal{S}_i} \alpha_j)$ (detailed skipped here). 

Local Approximation

L_2 error: $D(c, c') = \int (c(x) - c'(x))^2 dx$

Distance between original model $f = \sum_{j=1}^n \alpha_j \phi_j$ and approximator $g = \sum_{i=1}^m w_i g_i$:

$$\mathcal{E} = \int \left(\sum_{i=1}^m w_i g_i(x) - \sum_{j=1}^n \alpha_j \phi_j(x) \right)^2 dx \leq m \sum_{i=1}^m \int \underbrace{\left(w_i g_i(x) - \sum_{j \in S_i} \alpha_j \phi_j(x) \right)^2}_{\bar{\mathcal{E}}_i}$$

$$\bar{\mathcal{E}}_i = w_i^2 \frac{c_{Kd}}{|2\tilde{\mathbf{H}}_i|^{1/2}} - 2w_i \sum_{j \in S_i} \frac{c_{Kd} \alpha_j k(r_{ij})}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{1/2}} + c_i,$$

- $c_{Kd} = \int k(\mathbf{x}'\mathbf{x}) d\mathbf{x}$ is a kernel-dependent constant
- c_i is a data-dependent constant
- $r_{ij} = (\mathbf{t}_i - \mathbf{x}_j)'(\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1}(\mathbf{t}_i - \mathbf{x}_j)$
 - Mahalanobis distance between \mathbf{x}_j and \mathbf{t}_i

Task: Minimize $\bar{\mathcal{E}}_i$ (upper bound of \mathcal{E}_i) w.r.t. w_i , $\tilde{\mathbf{H}}_i$, and \mathbf{t}_i

- direct minimization is difficult

Minimizing the Local Error Bound

- $\bar{\mathcal{E}}_i$ is a quadratic function of w_i . By setting

$$w_i = |\mathbf{2}\tilde{\mathbf{H}}_i|^{\frac{1}{2}} \sum_{j \in \mathcal{S}_i} \alpha_j k(r_{ij}) (|\mathbf{H}_j + \tilde{\mathbf{H}}_i|)^{-1/2},$$

- we have $\bar{\mathcal{E}}_i^{\min}$. Let its derivatives w.r.t. $\tilde{\mathbf{H}}_i$ and \mathbf{t}_i be zero, we have

$$\mathbf{t}_i = \mathbf{M}_i^{-1} \sum_{j \in \mathcal{S}_i} \frac{\alpha_j k'_{\mathbf{H}_j + \tilde{\mathbf{H}}_i}(\mathbf{x}_j - \mathbf{t}_i) (\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1} \mathbf{x}_j}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{1/2}}$$

$$\bullet \mathbf{M}_i = \sum_{j \in \mathcal{S}_i} \frac{\alpha_j k'_{\mathbf{H}_j + \tilde{\mathbf{H}}_i}(\mathbf{x}_j - \mathbf{t}_i) (\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1}}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{1/2}}$$

$$\tilde{\mathbf{H}}_i = \mathbf{P}_i^{-1} \sum_{j \in \mathcal{S}_i} \frac{\alpha_j (\tilde{\mathbf{H}}_i + \mathbf{H}_j)^{-1}}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{1/2}} \left(k(r_{ij}) \mathbf{H}_j + 4(-k'(r_{ij})) \mathbf{C}_i^{\mathbf{x}} (\tilde{\mathbf{H}}_i + \mathbf{H}_j)^{-1} \tilde{\mathbf{H}}_i \right)$$

- $\mathbf{C}_i^{\mathbf{x}} = (\mathbf{x}_j - \mathbf{t}_i)(\mathbf{x}_j - \mathbf{t}_i)'$, $\mathbf{P}_i = \sum_{j \in \mathcal{S}_i} \frac{(\tilde{\mathbf{H}}_i + \mathbf{H}_j)^{-1}}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{1/2}} \alpha_j k(r_{ij})$
- Iterate these two steps until convergence

Robust Covariance Estimation

- Consider a simple case where all \mathbf{H}_i 's are the same as \mathbf{H} , and all α_j 's are the same(to gain more insight).
- The the local bandwidth matrix can be computed by

$$\tilde{\mathbf{H}}_i = \mathbf{H} + 4\tilde{\mathbf{H}}_i(\tilde{\mathbf{H}}_i + \mathbf{H})^{-1}\mathbf{V}_i,$$

where

$$\mathbf{V}_i = \frac{\sum_{j \in \mathcal{S}_i} (-k'(r_{ij}))(\mathbf{x}_j - \mathbf{t}_i)(\mathbf{x}_j - \mathbf{t}_i)'}{\sum_{j \in \mathcal{S}_i} k(r_{ij})}$$

- \mathbf{V}_i is a robust estimate of the local cluster covariance S_i , which has been used by (Vincent and Bengio) to handle high dimensional data more reliably.

Relationship with Mean Shift

Mean shift

- popular density-based clustering algorithm in vision
- each peak of the density function is a mode
- iteratively shift the data points along the density gradient until they reach local maxima

Computation of the center \mathbf{t}_i is indeed a mean-shift procedure

$$\mathbf{t}_i = \frac{\sum_{\mathbf{x}_j \in \mathcal{S}_i} k'_{\mathbf{H} + \tilde{\mathbf{H}}_i}(\mathbf{x}_j - \mathbf{t}_i) \mathbf{x}_j}{\sum_{\mathbf{x}_j \in \mathcal{S}_i} k'_{\mathbf{H} + \tilde{\mathbf{H}}_i}(\mathbf{x}_j - \mathbf{t}_i)}$$

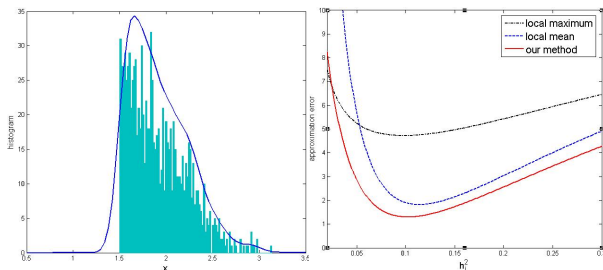
Can locate the peaks of the density

$$\rho_i(\mathbf{x}) = |\mathbf{H} + \tilde{\mathbf{H}}_i|^{-1/2} \sum_{\mathbf{x}_j \in \mathcal{S}_i} K_{\mathbf{H} + \tilde{\mathbf{H}}_i}(\mathbf{x} - \mathbf{x}_j)$$

Comparison with

- maximum of the local model $f_i(\mathbf{x}) = \sum_{j \in \mathcal{S}_i} |\mathbf{H}|^{-1/2} K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_j)$
- mean of \mathcal{S}_i

Illustration



Distribution is asymmetric

- the center t_i should be biased towards the heavier side of the data distribution \rightarrow maximum of f_i fails

Our method gives the best performance

- p_i has a larger bandwidth than the original f_i \rightarrow its maximum will move towards the heavier side of the distribution

Computational Complexity

Partitioning step

- by using a hierarchical scheme [Feder and Greene, 1988]
- $O(n \log(m)d)$
 - n : data size; m : number of clusters; d : dimensionality

Local approximation step

- $\sum_{i=1}^m l_i n_i d^3 = Lnd^3$, where L is the maximum number of iterations
- use diagonal $\tilde{\mathbf{H}}_i$: \rightarrow complexity linear in d

Overall complexity

- $O(nd(\log(m) + L))$
- linear in both data size and dimensionality

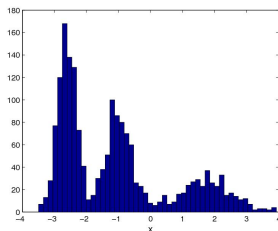
Experiment: Simplifying Parzen Window Estimator

Goal:

- Reduce the number of kernels in Parzen window estimator

Data:

- 1-dimensional data with 1,800 samples
- 3-Gaussian mixture

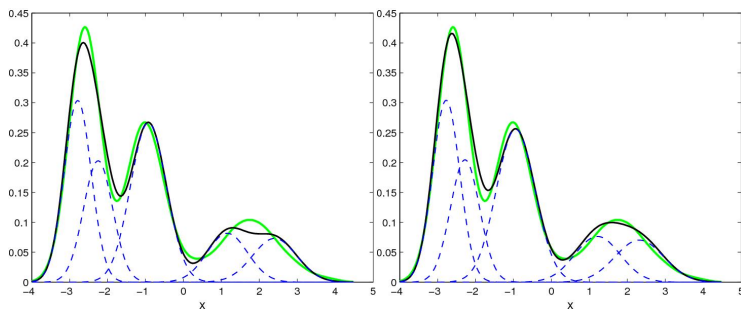


Procedure:

- Use $m = 5$ kernels to approximate the density
- Gaussian kernel with fixed bandwidth $h = 0.3$
- Randomly generate the data 100 times

Results

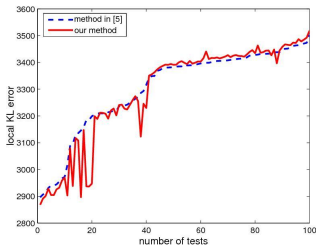
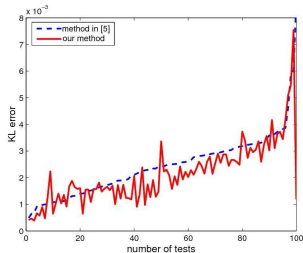
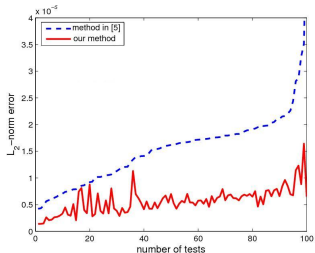
(left: [Goldberger and Roweis 2005]; right: ours)



(green: Parzen window estimator; black: Simplified model;
blue-dashed: Components of the model)

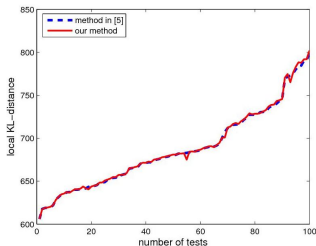
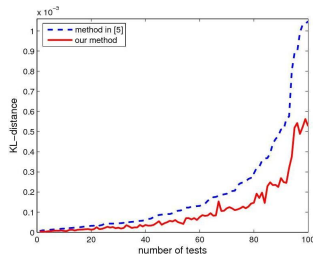
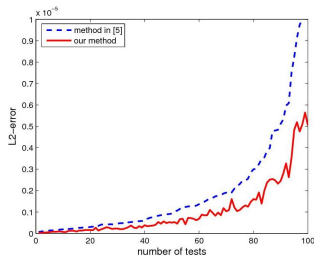
- note the break in the third component on the left figure

Quantitative Evaluation



- Repeated 100 times
- more accurate than [Goldberger and Roweis 2005]

Quantitative Evaluation

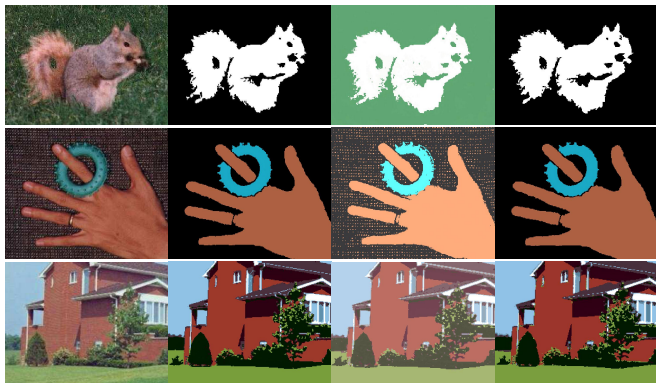


Approximate $f = \sum_{i=1}^{500} \alpha_i \mathcal{N}(x_i, \Sigma_i)$

- x_i and α_i randomly chosen from $[0, 10]$ and $[0, 1]$
- reduced to $m = 10$ kernels
- repeated 100 times

Experiment: Image Segmentation

- Mean shift clustering using RGB + XY features:
- Gaussian kernel
(Original image; Mean shift; [Goldberger and Roweis 2005]; Ours)



- comparable segmentation results

Time and Number of Components Obtained

Comparison with

- standard mean shift
- fast mean shift using kd-trees (using the ANN library)

Codes in C++ and run on a 2.26GHz Pentium-III machine

image	data size	time(s)			num of components (ours)
		mean shift standard	kd-tree	ours time (s)	
<i>squirrel</i> (209×288)	60,192	1215.8	11.94	0.18	81
<i>hand</i> (243×302)	73,386	1679.7	12.92	0.35	120
<i>house</i> (192×255)	48,960	1284.5	5.16	0.22	159
<i>lake</i> (512×512)	262,144	3343.0	85.65	3.67	440

- number of components much smaller
- much faster than standard and fast mean shift

More Segmentation Results

Images from the Berkeley segmentation benchmark



Segmentation results



Conclusion

- An **efficient** approach for simplifying mixture models
- Complexity is **linear** in sample size and dimensionality
- Empirical performance shows that the use of L_2 distance is better than KL-based distance

Future work:

- More experiments on speeding up SVM testing
- Design of better partitioning methods and tighter error bound