New techniques for distribution learning

Abbas Mehrabian

McGill University

31 January 2018

My research

2009–11 Graph theory (University of Waterloo)

2012–15 Random graphs with applications to network science (Waterloo)

2015–16 Randomized algorithms with applications to network science/distributed computing (e.g. gossip and load balancing protocols) (University of British Columbia)

2017–18 Theoretical machine learning (UC Berkeley, McGill)

Generally interested in mathematical foundations of CS.

What is distribution learning?

- ✓ The goal for unsupervised learning is to find structure in the data in order to learn more about the data, e.g.

  1. Clustering
  2. Anomaly detection
  3. Principal component analysis

What is distribution learning?

- ✓ The goal for unsupervised learning is to find structure in the data in order to learn more about the data, e.g.

  1. Clustering
  2. Anomaly detection
  3. Principal component analysis

- ✓ Distribution learning (density estimation) means explicitly estimating the distribution underlying the data

  1. can be explored to find structure in the data
  2. can be used to generate new data

Applications of distribution learning

detecting breast cancer

- ✓ Training data consists of non-cancerous X-ray images.
- ✓ A probability density function $\mathbf{P} : \mathbb{R}^d \rightarrow \mathbb{R}$ is learned from the data.
- ✓ When a new input x is presented, a high value for $\mathbf{P}(x)$ indicates a non-cancerous image, while a low value indicates a novel input, which might be characteristic of cancer.

Distribution learning task

Given an i.i.d. sample generated from an unknown target distribution $\mathbf{P}$, output a distribution $\widehat{\mathbf{P}}$ that is close to the target $\mathbf{P}$.

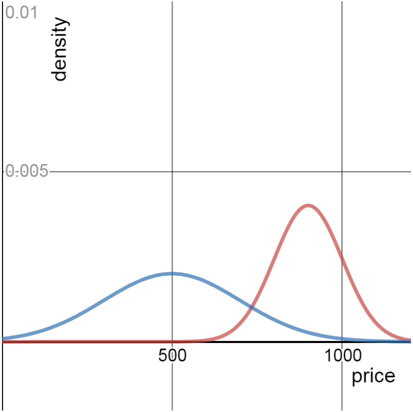
Possible assumptions on the distribution $\mathbf{P}$ underlying the data

- ✓ Gaussians
- ✓ Mixtures of simpler distributions
- ✓ Graphical models
- ✓ Neural networks
- ✓ No assumption!

Each assumption defines a class of distributions $\mathcal{F}$.
Today, $\mathcal{F}$ will be mixtures of Gaussians.

A model for house prices

Mixtures of Gaussians

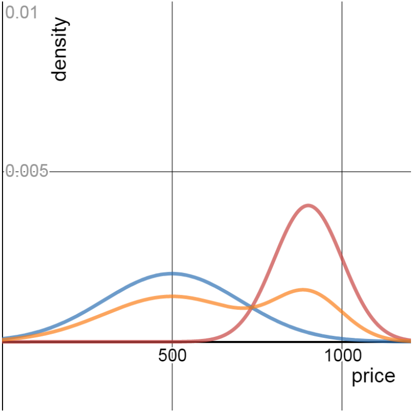


distribution of prices of a 2-bedroom

distribution of prices of a 1-bedroom

A model for house prices

Mixtures of Gaussians



distribution of prices of a 2-bedroom

distribution of prices of a 1-bedroom

distribution of prices of a 1 or 2-bedroom

Mixture of Gaussians

A mixture of k Gaussians in d dimensions has density $\sum_{i=1}^{k} w_i \mathcal{N}(\mu_i, \Sigma_i)(x)$, where $w_i \geq 0$ and $\sum w_i = 1$.

$\mathcal{N}(\mu, \Sigma)(x) = \exp(-(x-\mu)^T \Sigma^{-1} (x-\mu)/2)(2\pi)^{-d/2} \det(\Sigma)^{-1/2}$

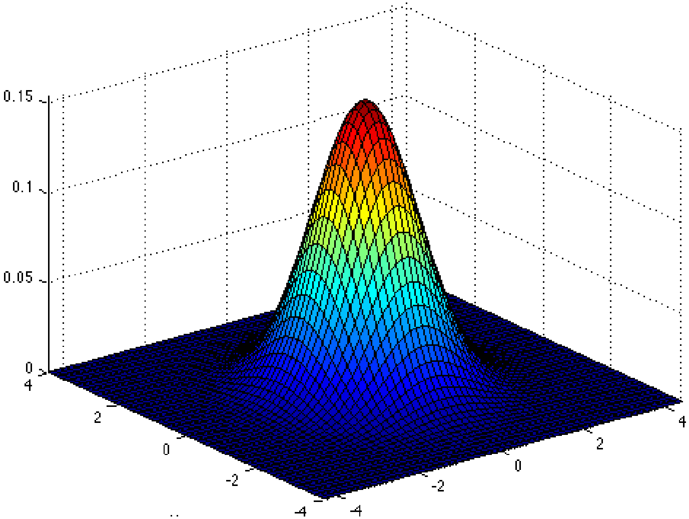
Mixture of Gaussians

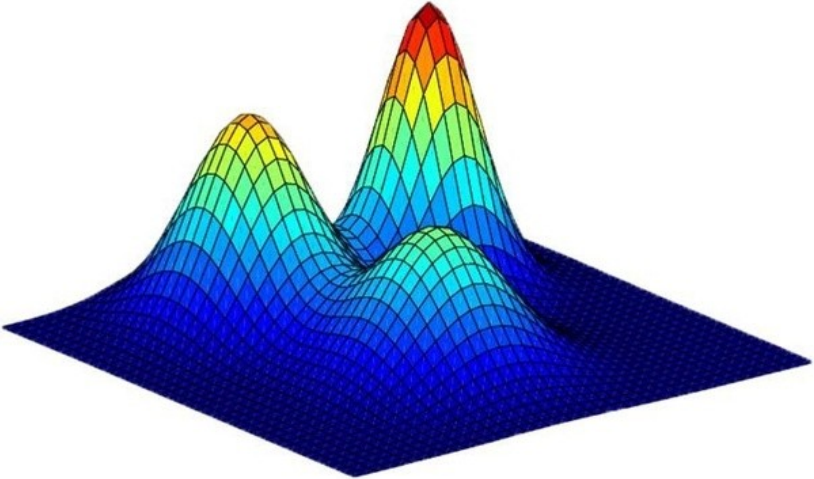
A mixture of k Gaussians in d dimensions has density $\sum_{i=1}^{k} w_i \mathcal{N}(\mu_i, \Sigma_i)(x)$, where $w_i \geq 0$ and $\sum w_i = 1$.

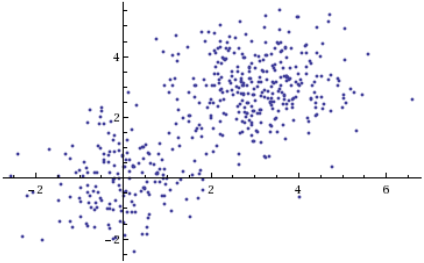
$\mathcal{N}(\mu, \Sigma)(x) = \exp(-(x-\mu)^T \Sigma^{-1} (x-\mu)/2)(2\pi)^{-d/2} \det(\Sigma)^{-1/2}$

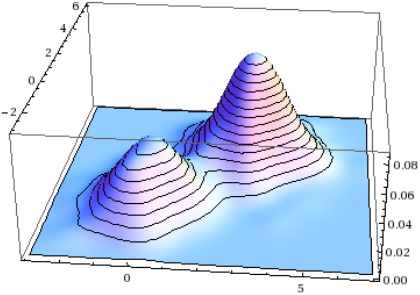
Lots of applications, e.g.,

- ✓ Financial returns often behave differently in normal situations and during crisis times.
- ✓ House prices in different areas.
- ✓ Whenever each data point belongs to one of some number of different sources or categories, each of them being almost a Gaussian.


0.15
0.1
0.05
0
4
2
0
-2
-4
-4
-2
0
2
4




-2
2
4
6
4
2
-2

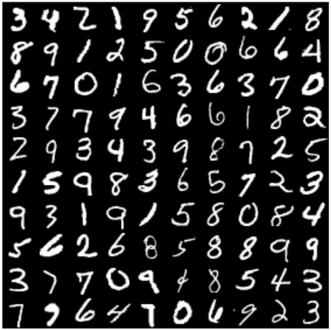

6
4
2
0
-2
0.08
0.06
0.04
0.02
0.00
0
5

Application of distribution learning

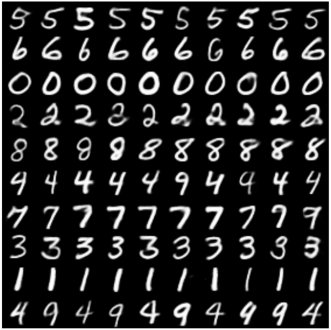
Generating new data

1. Let's say we want to learn to generate random handwritten digits.
2. The training data is the MNIST database: 60,000 handwritten digits.
3. We feed this data into a distribution learning algorithm, which learns a mixture of Gaussians.
4. Then we generate new data from the learned distribution.

Distribution learning in action



training data



generated data

Pictures from Dilokthanakul, Mediano, Garnelo, Lee, Salimbeni, Arulkumaran, Shanahan (Imperial College London), January 2017

Efficiency of the algorithm

We would like to design an algorithm that

- ✓ requires as few input data as possible (sample complexity)
- ✓ runs as fast as possible (computational complexity)

Today we focus on sample complexity (information theoretic/statistical aspects of the problem).
The first step for designing an efficient algorithm.

Guarantee of the algorithm

Our algorithm should guarantee that with high probability, output distribution is "close" to the underlying distribution.

There are various common distance measures between distributions. One important one is the total variation distance: $\mathrm{TV}(\mathbf{P}, \widehat{\mathbf{P}}) \coloneqq \sup_E |\mathbf{P}(E) - \widehat{\mathbf{P}}(E)|$.

Properties of total variation distance

1. Symmetric, lies in [0,1], scale-invariant.
2. $\mathrm{TV}(\mathbf{P}, \widehat{\mathbf{P}}) = \frac{1}{2} \int |\mathbf{P}(x) - \widehat{\mathbf{P}}(x)| dx = \frac{1}{2} \|\mathbf{P} - \widehat{\mathbf{P}}\|_1$
3. If $\mathrm{TV}(\mathbf{P}, \widehat{\mathbf{P}}) \le \varepsilon$, for *all* events E, $\widehat{\mathbf{P}}(E)$ approximates $\mathbf{P}(E)$ within ε. So we get a uniform guarantee.

Distribution learning task

Our setup

Given an i.i.d. sample generated from an unknown target distribution $\mathbf{P}$ belonging to a known class $\mathcal{F}$, output a distribution $\widehat{\mathbf{P}}$ that is close to the target $\mathbf{P}$.
What is the smallest number of samples needed to guarantee $\mathrm{TV}(\widehat{\mathbf{P}}, \mathbf{P}) \leq \varepsilon$ with probability 99%? $m_{\mathcal{F}}(\varepsilon)$.

Distribution learning task

Our setup

Given an i.i.d. sample generated from an unknown target distribution $\mathbf{P}$ belonging to a known class $\mathcal{F}$, output a distribution $\widehat{\mathbf{P}}$ that is close to the target $\mathbf{P}$.
What is the smallest number of samples needed to guarantee $\mathrm{TV}(\widehat{\mathbf{P}}, \mathbf{P}) \leq \varepsilon$ with probability 99%? $m_{\mathcal{F}}(\varepsilon)$.

Main problem

provide bounds for $m_{\mathcal{F}}(\varepsilon)$ for various classes $\mathcal{F}$.

Main results

mixtures of Gaussians

Theorem (Ashtiani, Ben-David, M'17)

The sample complexity for learning mixtures of k Gaussians in d dimensions within distance ε is upper bounded by $\widetilde{O}(kd^2/\varepsilon^4)$.

✓ Improvement over previous upper bounds

- $\widetilde{O}(k^4d^4/\epsilon^2)$ (Karpinski and Macintyre'97)
- $\widetilde{O}(k^3d^2/\epsilon^4)$ (Diakonikolas, Kane, Stewart'17)

Theorem (Harvey, Liaw, M, Plan'18)

This sample complexity is lower bounded by $\widetilde{\Omega}(kd^2/\varepsilon^2)$.

Main results

mixtures of axis-aligned Gaussians

Theorem (Ashtiani, Ben-David, M'18)

Sample complexity for learning mixtures of k axis-aligned Gaussians in d dimensions within distance ε is bounded by $\widetilde{O}(kd/\varepsilon^2)$.

axis-aligned Gaussian: $\sum_{i=1}^{k} w_i \mathcal{N}(\mu_i, \Sigma_i)$, $w_i \geq 0$, $\sum w_i = 1$,

each Σ_i is a diagonal matrix.

Main results

mixtures of axis-aligned Gaussians

Theorem (Ashtiani, Ben-David, M'18)

Sample complexity for learning mixtures of k axis-aligned Gaussians in d dimensions within distance ε is bounded by $\widetilde{O}(kd/\varepsilon^2)$.

axis-aligned Gaussian: $\sum_{i=1}^{k} w_i \mathcal{N}(\mu_i, \Sigma_i)$, $w_i \geq 0$, $\sum w_i = 1$,

each Σ_i is a diagonal matrix.

- ✓ Tight bound (up to log factors): a matching lower bound was proved in Suresh, Orlitsky, Acharya, Jafarpour'14.
- ✓ Improvement over known upper bounds
  - $\widetilde{O}((k^4 d^2 + k^3 d^3)/\epsilon^2)$ (Karpinski and Macintyre'97)
  - $\widetilde{O}(k^9 d/\epsilon^4)$ (Suresh, Orlitsky, Acharya, Jafarpour'14)

Main results

mixtures of Gaussians

Theorem (Ashtiani, Ben-David, M'17)

The sample complexity for learning mixtures of k Gaussians in d dimensions within distance ε is upper bounded by $\widetilde{O}(kd^2/\varepsilon^4)$.

Now we discuss the main ideas ...

A generic bound for mixtures

Theorem (Ashtiani, Ben-David, M'17)

Assume that $\mathcal{F}$ has sample complexity $m_{\mathcal{F}}(\epsilon)$. Then k-mix($\mathcal{F}$) has sample complexity

$$m_{k\text{-mix}(\mathcal{F})}(\epsilon) = O\left(\frac{k\log(1+k)\cdot m_{\mathcal{F}}(\epsilon)}{\epsilon^2}\right).$$

k-mix($\mathcal{F}$) is the class of k mixtures of members of $\mathcal{F}$.

High level overview of the generic mixture bound

two rounds of sampling

1. Build a finite set of “candidate” distributions based on the 1st sample, such that one of them is ε-close to target.
2. Choose the best candidate based on the 2nd sample.

High level overview of the generic mixture bound

two rounds of sampling

1. Build a finite set of “candidate” distributions based on the 1st sample, such that one of them is ε-close to target.
2. Choose the best candidate based on the 2nd sample.

We know how to learn a single component using $m_{\mathcal{F}}(\varepsilon)$ samples. In the case of mixtures, we don’t know which sample point came from which component of mixture; but we can try “all” possible cases (exhaustive search) to generate the candidates.

High level overview of the generic mixture bound

two rounds of sampling

1. Build a finite set of "candidate" distributions based on the 1st sample, such that one of them is ε-close to target.
2. Choose the best candidate based on the 2nd sample.

We know how to learn a single component using $m_{\mathcal{F}}(\varepsilon)$ samples. In the case of mixtures, we don't know which sample point came from which component of mixture; but we can try "all" possible cases (exhaustive search) to generate the candidates. For choosing the best candidate:

Theorem (Devroye and Lugosi'01)

Given a finite set $\mathcal{C}$ of candidates, given $O(\log(|\mathcal{C}|)/\epsilon^2)$ additional samples from the target distribution, we can find the candidate that is closest to the target.

Algorithm

Target is $\sum_{i=1}^{k} w_i G_i$, with $\sum w_i = 1$, each $G_i \in \mathcal{F}$
In words: try all possible ways of partitioning data into components, and all possible mixture weights

```
Input: $k, \epsilon$ and an iid sample $S$ of size $k m_{\mathcal{F}}(\varepsilon)$
0. Let $\widehat{W}$ be an $(\epsilon/k)$-cover for $\Delta_k$ in $\ell_\infty$ distance.
1. $\mathcal{C} = \emptyset$. (set of candidate distributions)
2. For each $(\widehat{w}_1, \ldots, \widehat{w}_k) \in \widehat{W}$ do:
    3. For each possible partition of $S$ into $A_1, A_2$, ..., $A_k$:
        4. Provide $A_i$ to the $\mathcal{F}$-learner, let $\widehat{G}_i$ be its output.
        5. Add the candidate distribution $\sum_{i \in [k]} \widehat{w}_i \widehat{G}_i$ to $\mathcal{C}$.
6. Apply the algorithm for finite classes to $\mathcal{C}$.
```

Algorithm

Target is $\sum_{i=1}^{k} w_i G_i$, with $\sum w_i = 1$, each $G_i \in \mathcal{F}$
In words: try all possible ways of partitioning data into components, and all possible mixture weights

```
Input: $k, \epsilon$ and an iid sample $S$ of size $km_{\mathcal{F}}(\varepsilon)$
0. Let $\widehat{W}$ be an $(\epsilon/k)$-cover for $\Delta_k$ in $\ell_\infty$ distance.
1. $\mathcal{C} = \emptyset$. (set of candidate distributions)
2. For each $(\widehat{w}_1, \ldots, \widehat{w}_k) \in \widehat{W}$ do:
    3. For each possible partition of $S$ into $A_1, A_2, ..., A_k$:
        4. Provide $A_i$ to the $\mathcal{F}$-learner, let $\widehat{G}_i$ be its output.
        5. Add the candidate distribution $\sum_{i \in [k]} \widehat{w}_i \widehat{G}_i$ to $\mathcal{C}$.
6. Apply the algorithm for finite classes to $\mathcal{C}$.
```

✓ $|C| \leq k^{km_{\mathcal{F}}(\varepsilon)} \times (1/\varepsilon)^k$ so $\log |C| \leq (km_{\mathcal{F}}(\varepsilon)) \log k + k \log(1/\varepsilon)$.

✓ Remains to prove that there is an ϵ-close candidate in $\mathcal{C}$.

From generic mixture bound to mixtures of Gaussians

Theorem (Ashtiani, Ben-David, M'17)

$m_{k\text{-mix}(\mathcal{F})}(\epsilon) = O\left(\frac{k \log k \cdot m_{\mathcal{F}}(\epsilon)}{\epsilon^2}\right).$

Sample complexity of single Gaussians

The class of d-dimensional Gaussians has sample complexity $O(d^2/\epsilon^2)$.

Theorem (Ashtiani, Ben-David, M'17)

By the generic mixture bound, the class of mixtures of k Gaussians in $\mathbb{R}^d$ has sample complexity $\widetilde{O}(kd^2/\epsilon^4)$.

Very recently, we showed a lower bound of $\widetilde{\Omega}(kd^2/\epsilon^2)$.

Mixtures of axis-aligned Gaussians

Theorem (Ashtiani, Ben-David, M'17)

$m_{k\text{-mix}(\mathcal{F})}(\epsilon) = O\left(\frac{k \log k \cdot m_{\mathcal{F}}(\epsilon)}{\epsilon^2}\right).$

Sample complexity of single axis-aligned Gaussians

The class of d-dimensional axis-aligned Gaussians has sample complexity $O(d/\epsilon^2)$.

Theorem (Ashtiani, Ben-David, M'17)

By the generic mixture bound, the class of mixtures of k axis-aligned Gaussians in $\mathbb{R}^d$ has sample complexity $\widetilde{O}(kd/\epsilon^4)$.

We improve this to $\widetilde{O}(kd/\epsilon^2)$ in the next part!

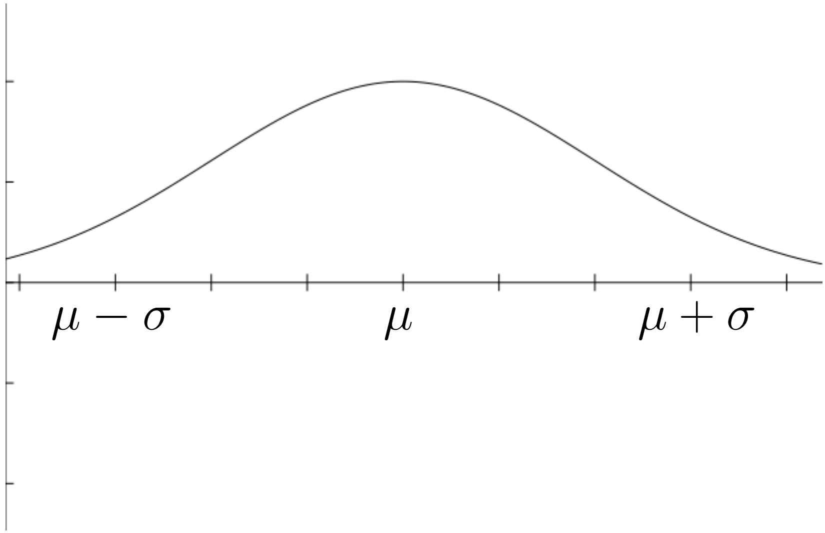
Distribution learning via compression

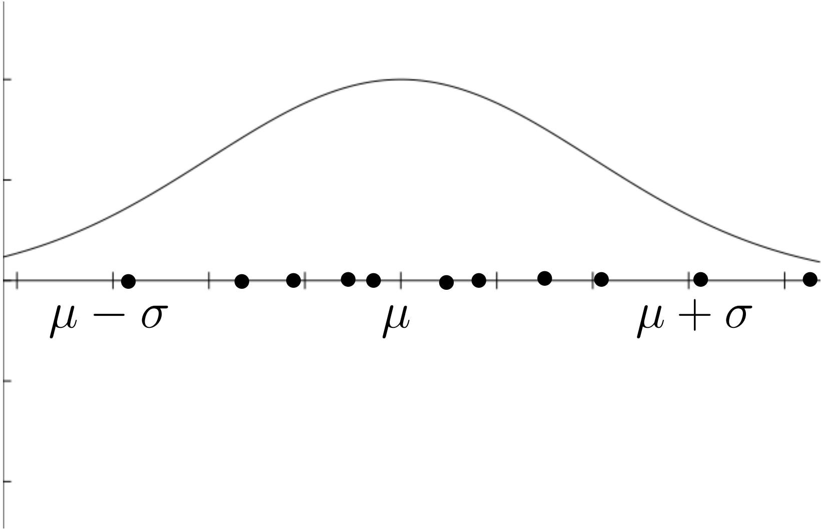
Distribution decoder

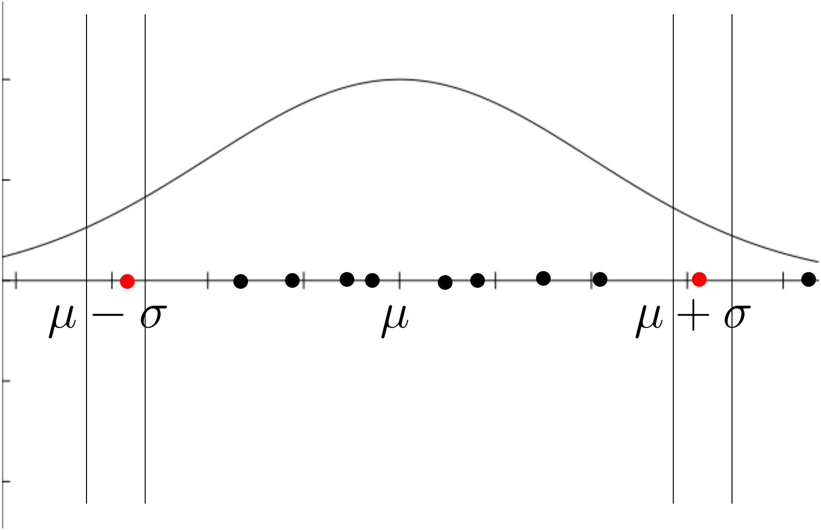
A *distribution decoder* for $\mathcal{F}$ is a function $\mathcal{J}$ that takes a finite sequence of data points, and outputs a member of $\mathcal{F}$.

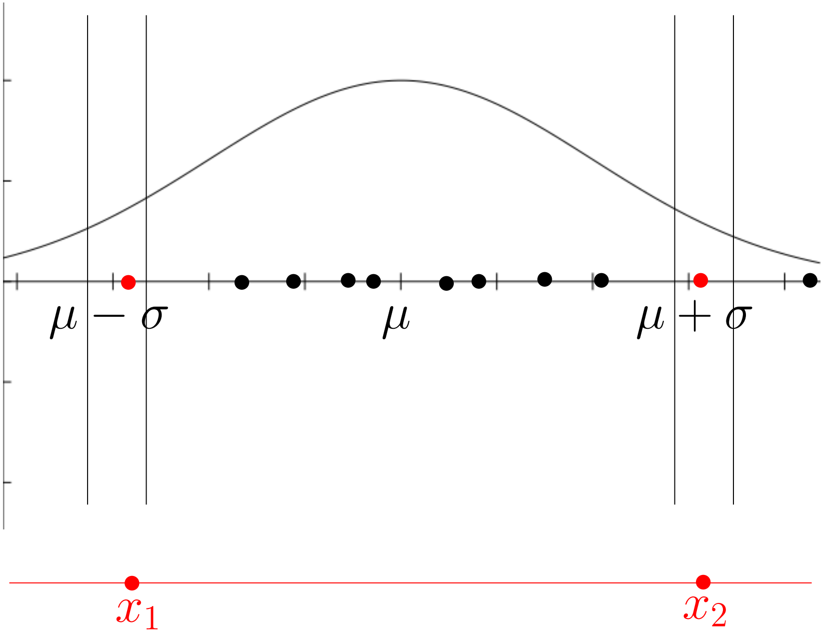
Distribution compression schemes

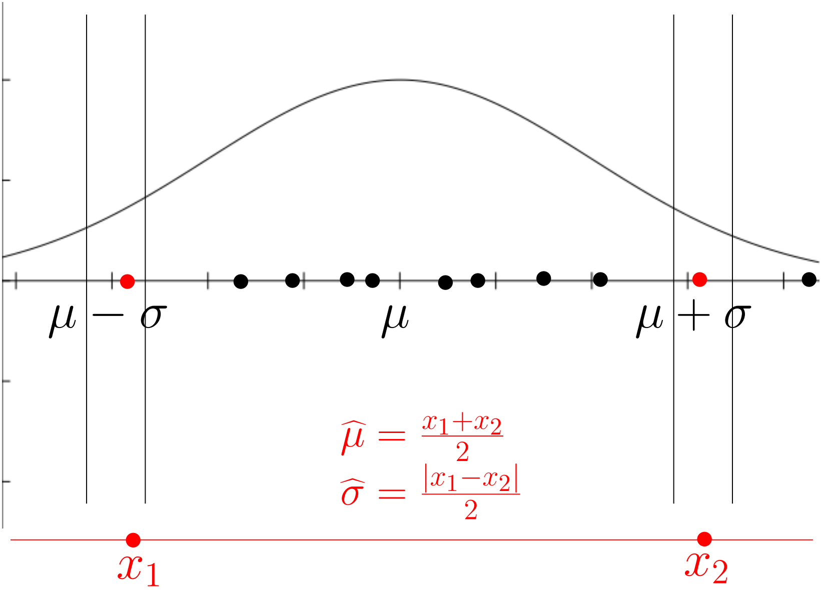
Class $\mathcal{F}$ admits τ-compression if there exists a decoder $\mathcal{J}$ for $\mathcal{F}$ such that for any $\mathbf{P} \in \mathcal{F}$ and any $\varepsilon \in (0, 1)$, if S is a large enough sample generated from $\mathbf{P}$, then with probability 99% there exists a sequence L of at most τ points of S such that $\mathrm{TV}(\mathcal{J}(L), \mathbf{P}) \leq \epsilon$.


$\mu - \sigma$
μ
$\mu + \sigma$


$\mu - \sigma$
μ
$\mu + \sigma$

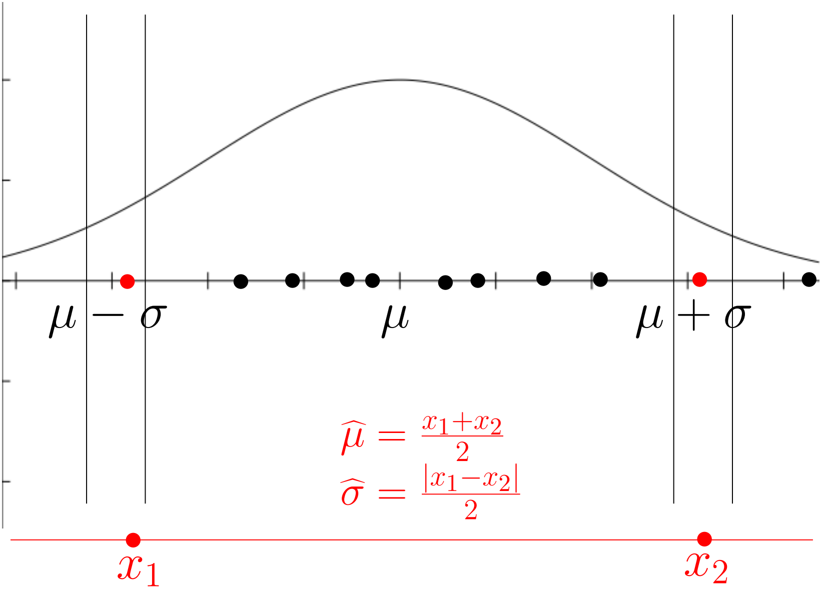

$\mu - \sigma$
μ
$\mu + \sigma$


$\mu - \sigma$
μ
$\mu + \sigma$
x_1
x_2



$$\widehat{\mu} = \frac{x_1 + x_2}{2}$$

$$\widehat{\sigma} = \frac{|x_1 - x_2|}{2}$$


$\mu - \sigma$
μ
$\mu + \sigma$
$\widehat{\mu} = \frac{x_1 + x_2}{2}$
$\widehat{\sigma} = \frac{|x_1 - x_2|}{2}$
x_1
x_2

$\mathrm{TV}(\mathcal{N}(\widehat{\mu}, \widehat{\sigma}), \mathcal{N}(\mu, \sigma)) \leq \varepsilon$, so a single 1 dimensional Gaussian is 2-compressible.

Distribution learning via compression

Compression implies learnability

If $\mathcal{F}$ admits τ-compression, then it can be learned using $\widetilde{O}(\tau/\varepsilon^2)$ samples.

Compressing product distributions

If $\mathcal{F}$ admits τ-compression, then $\mathcal{F}^d$ admits $(d\tau)$-compression.

Compressing mixtures

If $\mathcal{F}$ admits τ-compression, then k-mix$(\mathcal{F})$ admits $(k\tau + k\log k)$-compression.

Learning Mixtures of Axis-Aligned Gaussians

Compressing 1-dimensional Gaussians

1-dimensional Gaussians admit 2-compression.

Compressing axis-aligned Gaussians

Axis-aligned Gaussians over $\mathbb{R}^d$ admit $2d$-compression.

Compressing mixtures of axis-aligned Gaussians

Mixtures of k axis-aligned Gaussians over $\mathbb{R}^d$ admit $(2kd + k \log k)$-compression.

Learning mixtures of axis-aligned Gaussians

Mixtures of k axis-aligned Gaussians over $\mathbb{R}^d$ can be learned using $\widetilde{O}(kd/\epsilon^2)$ samples.

Review: Our main results

k is the number of mixture components

d is the dimension, ε is the error tolerance

Theorem (Ashtiani, Ben-David, M'17)

Sample complexity for learning mixtures of general Gaussians is upper bounded by $\widetilde{O}(kd^2/\varepsilon^4)$.

Theorem (Harvey, Liaw, M, Plan'18)

Sample complexity for learning mixtures of general Gaussians is lower bounded by $\widetilde{\Omega}(kd^2/\varepsilon^2)$.

Theorem (Ashtiani, Ben-David, M'18)

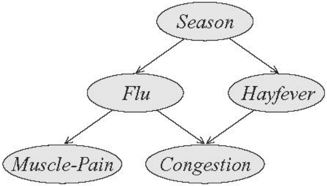
Sample complexity for learning mixtures of axis-aligned Gaussians is upper bounded by $\widetilde{O}(kd/\varepsilon^2)$. (matching lower bound is known).

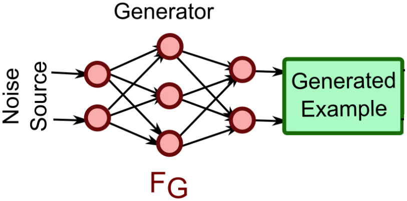
Future work

- ✓ For mixtures of Gaussians, $kd^2/\varepsilon^2 \leq$ sample complexity $\leq kd^2/\varepsilon^4$. What is the correct bound? Since a mixture of k Gaussians in d dimensions has kd^2 parameters, we conjecture that kd^2/ε^2 is the correct answer.
- ✓ Our algorithms are exponential time. Can we design polynomial time algorithms with similar guarantees?

Future work

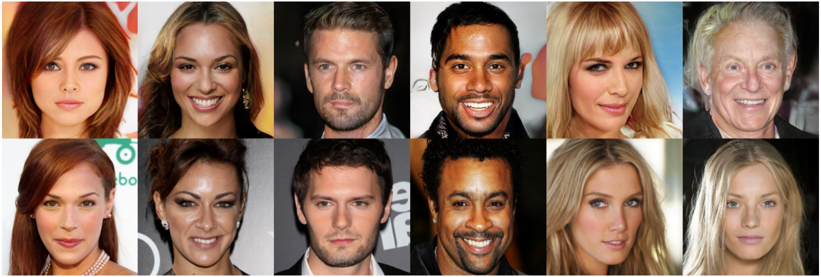
✓ Sample complexity of learn more general distribution families, e.g. graphical models, neural networks.





Future work

Neural networks can generate real-looking images



Top: generated images using a distribution learning method based on deep learning (generative adversarial networks)

Bottom: training data

Picture from Karras, Aila, Laine, and Lehtinen (NVIDIA and Aalto University), October 2017

Thanks to my co-authors

Hassan Ashtiani
(Waterloo)

Shai Ben-David
(Waterloo)

Nick Harvey
(UBC)

Chris Liaw
(UBC)

Yaniv Plan
(UBC)