Information Geometry with Applications to Classification

Nitin Kumar, Ryan Marren, Jason Yim
Johns Hopkins University
AS.110.446 Final Project

Abstract

Information geometry provides methods to represent distributions in terms of the amount of information they contain. We explore the most common method, entropy, used in conjunction with KL divergence to represent the distance between distributions and families of distributions. We show maximizing entropy on given constraints allow us to minimize the distance between a predicted distribution and actual distribution. Then, by making these constraints moments of the distribution, we can reduce our problem to the moment matching problem and solve the problem through representation as an exponential family. We present an algorithm that solves this problem using gradient descent on the canonical parameters of the exponential family and test it on classification problems with image and text data.

1 Setting, Motivation, and Layout

Suppose we have a d-dimensional random vector $X \in \Omega$ where $X \sim p^*$ where p^* is a probability distribution over Ω. We then take a set $\{x_i\}_{i=1}^n$ of n *iid* observations of this random vector X.

We then consider a set of functions $\Phi = \{\phi_j\}_{j=1}^k$. Each ϕ_j is a mapping $\Omega \to \mathbb{R}$ which is called a *potential function*, *feature mapping*, or a *sufficient statistic* depending on whether you consult statistical physics, machine learning, or mathematical statistics literature. We define $\Phi(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_k(x)]^T \in \mathbb{R}^k$ as a vector of all our feature mappings called a *feature vector*.

Given some Φ, we can compute the empirical expectations of our feature mappings by taking $\hat{\mu_j} := \frac{1}{n}\sum_{i=1}^n \phi_j(x_i)$. Similarly, we can define the empirical expectation of our feature vector $\hat{\mu} := [\mu_1, \mu_2, \ldots, \mu_k]^T \in \mathbb{R}^k$.

Suppose we wanted to find, or estimate, the true distribution p^* of X over Ω. If we had any distribution p over Ω, we could consider the expectation of $\Phi(X)$ given by $\mathbb{E}_p[\Phi(X)] = [\mathbb{E}_p[\phi_1(x)], \mathbb{E}_p[\phi_2(x)], \ldots, \mathbb{E}_p[\phi_k(x)]]^T$. If we wanted to search for $p \approx p^*$, it would be wise to ensure that $\mathbb{E}_p[\Phi(X)] = \mathbb{E}_{p^*}[\Phi(X)]$. We do not know $\mathbb{E}_{p^*}[\Phi(X)]$, so we replace this quantity with the empirical estimation μ to get $\mathbb{E}_p[\Phi(X)] = \hat{\mu}$. This is known as the *moment matching problem*, and is a fundamental problem in statistical inference, to which problems such as finding maximum entropy, minimum relative entropy, or maximum likelihood distributions can be reduced.

To solve this problem, we will first need to answer three core questions

1. What is the set of probability distributions $\mathcal{P}$ over which to search for p^*?
2. If there are multiple $p \in \mathcal{P}$ which satisfy the moment constraints, how do we define an objective function to choose amongst them in a principled way?
3. Given answers to the previous two questions, how would we algorithmically find the optimal $p^* \in \mathcal{P}$ maximizing the objective function, subject to p^* satisfying the moment constraints?

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

In section 2 we answer question 2 by introducing the Principle of Maximum Entropy and deriving a principled objective function from it. Then, in section 3, we show that any distribution maximizing the objective function defined in section 2 must live in a set of distributions called an *exponential family*. Next, in section 4, we introduce an iterative optimization algorithm which attempts to solve the optimization problem posed by minimizing the relative entropy between the optimal p^* and distributions in the exponential family. Finally, we apply this algorithm to several corpus and image classification tasks, and report our results.

2 The Principle of Maximum Entropy

Suppose some set of information $\mathcal{I}$ about a probability distribution q. $\mathcal{I}$ can be any information about the shape or structure of q, and we write $q \to \mathcal{I}$ to denote that, by some process, $\mathcal{I}$ can be extracted from distribution q. Given only $\mathcal{I}$ but not q, how would you find a distribution $\hat{q}$ to predict q? Intuitively, you would want to choose $\hat{q}$ such that all of the information $\mathcal{I}$ can be extracted from $\hat{q}$ ($\hat{q} \to \mathcal{I}$), but no other information $\mathcal{I}'$ can be extracted from $\hat{q}$ ($\hat{q} \not\to \mathcal{I}'$ for all $\mathcal{I}' \not\subseteq \mathcal{I}$). This gives us a principled way to choose $\hat{q}$, in that $\hat{q}$ should inform us of $\mathcal{I}$, but should otherwise be as uninformative as possible, so that we are not injecting into $\hat{q}$ information which we do not know to be true about q.

Measuring the amount of information in a probability distribution is a fundamental problem in the field of *information theory*, and therefore there exists techniques to do so. The most common way to do so is to measure the entropy in a distrib ution.

Definition 1. *The **entropy** of a probability distribution p is:*

$$H(p) = \int_{x \in \Omega} p(x) \log \frac{1}{p(x)} dx = -\mathbb{E}_p[\log p(X)]$$

Where Ω is the support of the random variable X distributed p.

Entropy can be seen as a measure of randomness. The randomness of a probability distribution can be thought of as an inverse to the amount of information extractable from the distribution. Indeed, the entropy has several properties which we would like such a measure of un-informativeness to have. For example:

1. Expansibility: If X has distribution $(p_1, p_2, \ldots, p_n)$ and Y has distribution $(p_1, p_2, \ldots, p_n, 0)$, then $H(X) = H(Y)$. Note that this works because $0 \log_2(\frac{1}{0}) = \lim_{n \to 0} n \log_2(\frac{1}{n}) = 0$.
2. Symmetry: If $X = (p, 1-p)$ and $Y = (1-p, p)$ then $H(X) = H(Y)$.
3. Additivity: If X and Y are independent then $H(X, Y) = H(X) + H(Y)$. n
4. Subadditivity: If X and Y are random variables, then $H(X, Y) \leq H(X) + H(Y)$.
5. Zero for deterministic variables: If $X \sim Bernoilli(1)$, then $H(x) = 0$.

In fact, entropy is the only measure which will satisfy all of these properties.

Another useful tool from information theory similar to entropy is the relative entropy. This can be thought of as the difference in entropy of two distributions.

Definition 2. *Given two distributions p and q for a random variable X, the **relative entropy** (also called the **Information Divergence** or **KL Divergence**) is defined as*

$$D(p\|q) = \mathbb{E}_p\left[\log_2 \frac{1}{q}\right] - \mathbb{E}_p\left[\log \frac{1}{p}\right] = \mathbb{E}_p\left[\log \frac{p}{q}\right] = \int_{x \in \Omega} p(x) \log \frac{p(x)}{q(x)} dx$$

This measurement can be thought of intuitively as the cost to encode X under probability distribution q when in reality X is distributed according to p.

Two important properties of the relative entropy are:

1. $D(p\|q) \geq 0$ and equals 0 iff $p = q$.
2. $D(p\|q) \neq D(q\|p)$ in generality.

Using the Principle of Maximum Entropy, when given a set of feasible probability distributions $\mathcal{P}$ subject to some constraints, we would choose the $p^* \in \mathcal{P}$ which maximizes $H(p)$ subject to p satisfying the constraints.

In fact, if we maximize this objective function with respect to the moment matching constraints discussed in section 1, we can conclude that $p^* \in \mathcal{F}$ where $\mathcal{F}$ is a distribution with very favorable properties.

3 Entropy Maximization, Exponential Families, and an Alternative Computation to solve the Moment Matching Problem

Recall from our motivating problem that given some observations $\{x_i\}_{i=1}^n$, a mapping to feature vectors $\Phi : \Omega \to \mathbb{R}^k$, and a calculation of the empirical expectations of these feature vectors given by μ, we would like to find a distribution p such that $\mathbb{E}_p[\Phi(X)] = \mu$.

Given our discussion in section 2, we have now determined that if there were multiple distributions which satisfied these constraints, we would choose the one with maximum entropy. Thus, we would like to solve the following optimization problem:

$$\begin{aligned} &\arg\max_p H(p) \\ &\text{subject to } \mathbb{E}_p[\Phi(X)] = \mu \\ &\int_{x\in\Omega} p(x) = 1 \\ &p(x) > 0 \ \forall \ x \in \Omega \end{aligned} \tag{1}$$

Note that we add two new constraints to ensure that the result is a probability distribution.

We solve optimization (1) by taking the Lagrangian. (**Note**: *Here we assume that the supporting set Ω is finite. A similar result can be shown for the general case.*):

$$L(\lambda, p) = H(p) + \sum_{i=1}^{k} \lambda_k(\mu_i - \sum_{x\in\Omega} p(x)\phi_k(x)) + \lambda_0(1 - \sum_{x\in\Omega} p(x)) - \sum_{x\in\Omega} \lambda(x)p(x).$$

Now, treating $[p(x) : x \in \Omega]$ as a vector, we can take, for each $x \in \Omega$,

$$\frac{\partial L(\lambda, p)}{\partial p(x)} = 1 + log(p(x)) - \sum_{i=1}^{k} \lambda_k\phi_k(x) + \lambda_0 - \lambda(x) = 1 + log(p(x)) - \langle \lambda, \Phi(x)\rangle + \lambda_0 - \lambda(x).$$

Setting the derivative to 0, we get:

$$p(x) = e^{\langle \lambda, \Phi(x)\rangle} - 1 - \lambda_0 - \lambda(x)$$

Now in this form, we have that $p(x) \geq 0$ for all $x \in \Omega$, and by complementary slackness conditions we have that $\lambda(x) = 0$ for all $x \in \Omega$. We are left with λ_0 which is the constraint to ensure the distribution is normalized. By taking $\lambda_0 = 1 - \Phi(\lambda) = \log \sum_{x\in\Omega}^{n} e^{\langle \lambda, \Phi(x)\rangle}$.

This tells us that the optimal solution p is of the form $p(x) = e^{\langle \lambda, \Phi(x)\rangle - \Psi(\lambda)}$.

Distributions of this form are called *exponential family* distributions.

Definition 3. *An **exponential family** is a set of probability distributions $\mathcal{F} = \{p_\theta\}_{\theta\in\Theta}$ of the form:*

$$p_\theta(x) = e^{\langle \theta, \Phi(x)\rangle - \Psi(\theta)}$$

where:

- *$\theta \in \Theta$ is called the* canonical parameter *and lives within a space $\Theta = \{\theta : \Psi(\theta)$ is finite$\}$ of* admissible parameters.
- *$\Phi : \Omega \to \mathbb{R}^k$ is the feature mapping defined in section 1. When the features are linearly independent (e.g. there does not exists a non-zero vector $a \in \mathbb{R}^k$ where $\langle a, \Phi(x)\rangle = 0$, we say that the exponential family is **minimal**.*

- $\Psi(\theta) := \log \int_{x \in \Omega} e^{\langle \theta, \Psi(x) \rangle}$ *is called the* log partition function.

We now state and prove some important properties of the family $\mathcal{F}$.

Lemma 4. *If Ω is a supporting set of a random variable X, $p_\theta \in \mathcal{F}$ is a distribution of X over Ω, and Φ is a feature mapping from Ω to $\mathbb{R}^k$, then $\nabla_\theta \Psi(\theta) = \mathbb{E}_{p_\theta}[\Phi(X)]$.*

Proof.

$$\nabla_\theta \Psi(\theta) = \nabla_\theta \log \int_{x \in \Omega} e^{\langle \theta, \Phi(X) \rangle} = \frac{\nabla_\theta \int_{x \in \Omega} e^{\langle \theta, \Phi(X) \rangle}}{\int_{x \in \Omega} e^{\langle \theta, \Phi(X) \rangle} dx} = \frac{\int_{x \in \Omega} \nabla_\theta e^{\langle \theta, \Phi(X) \rangle}}{e^{\Psi(\theta)}} = \int_{x \in \Omega} \Phi(x) e^{\langle \theta, \Phi(X) \rangle - \Psi(\theta)}$$

$$= \mathbb{E}_{p_\theta}[\Phi(X)]$$

□

Lemma 5. *Suppose Ω is a supporting set of a random variable X, $p_\theta \in \mathcal{F}$ is a distribution of X over Ω, and Φ is a feature mapping from Ω to $\mathbb{R}^k$ with linearly independent features (e.g. $\mathcal{F}$ is* minimal*). Let $\mathcal{M} = \{\mu : \exists p \text{ such that } \mathbb{E}_p[\Phi(X)] = \mu\}$. Then the map $\nabla_\theta \Psi : \Theta \to \mathcal{M}, \nabla_\theta \Psi(\theta) \mapsto \mathbb{E}_{p_\theta}$ is a bijective.*

Proof.

- We can see $\nabla_\theta \Psi$ is *surjective* from the solution to optimization problem (1). Since we show that the maximum entropy distribution with feature expectations μ is an exponential family distribution parameterized by the Lagrange multipliers λ, we have $\nabla_\theta \Psi(\lambda) = \mathbb{E}_\lambda[\Phi(X)]$.
- $\nabla_\theta \Psi$ is *injective* since $\mathcal{F}$ is minimal, so parameters $\theta \in \Theta$ uniquely parameterize distributions with expectations $\mu \in \mathcal{M}$.

□

Suppose Ω is a supporting set of a random variable X, $p_{\theta^*} \in \mathcal{F}$ is a distribution of X over Ω, and Φ is a feature mapping from Ω to $\mathbb{R}^k$ with linearly independent features (e.g. $\mathcal{F}$ is *minimal*). Suppose p_{θ^*} is the maximum entropy distribution which exhibits expectations μ. Such a distribution must be an exponential family distribution, and therefore finding it can be reduced to finding the parameterization θ^*. Further, since our feature mapping is minimal, the mapping $\nabla_\theta \Psi$ is bijective, and thus if $p_\theta = p_{\theta^*} \Rightarrow \mathbb{E}_{p_\theta}[\Phi(X)] = \mathbb{E}_{p_{\theta^*}}[\Phi(X)] \Rightarrow \theta = \theta^*$.

Since $p = q$ *iff* $D(p \| q) = 0$, we can further reduce this problem to finding p_{θ^*} and p_θ such that $D(p_{\theta^*} \| p_\theta) = 0$. We note that $D(p, q) \geq 0$ for all p, q and further that since $\theta^* \in \Theta$,there trivially exists $\theta \in \Theta$ such that $D(p_{\theta^*} \| p_\theta) = 0$.

Thus, the solution to the following optimization problem exists and is unique:

$$\begin{aligned} &\arg\min_\theta \; D(p_{\theta^*} \| p_\theta) \\ &\text{subject to } \theta \in \Theta \end{aligned} \tag{2}$$

We can calculate

$$D(p_{\theta^*} \| p_\theta) = \mathbb{E}_{p_{\theta^*}}[\frac{\log p_{\theta^*}}{p_\theta}] = \mathbb{E}_{p_{\theta^*}}[\log e^{\langle \theta^*, \Phi(X) \rangle - \Psi(\theta^*)} - \log e^{\langle \theta, \Phi(X) \rangle - \Psi(\theta)}]$$

$$= \mathbb{E}_{p_{\theta^*}}[\langle \theta^*, \Phi(X) \rangle - \Psi(\theta^*) - \langle \theta, \Phi(X) \rangle + \Psi(\theta)] = \langle \theta^*, \mu \rangle - \Psi(\theta^*) - \langle \theta, \mu \rangle + \Psi(\theta)$$

and note that $\langle \theta^*, \mu \rangle$ and $\Psi(\theta^*)$ do not depend on θ. Thus in optimization problem (2) we could minimize the objective $g(\theta) = \Psi(\theta) - \langle \theta, \mu \rangle$. Finally, we note that $\nabla_\theta g(\theta) = \nabla_\theta \Psi(\theta) - \nabla_\theta \langle \theta, \mu \rangle = \mathbb{E}_{p_\theta}[\Phi(X)] - \mu$.

This result gives us a more feasible way to find the $\theta \in \Theta$ such that $\mathbb{E}_{p_\theta}[\Phi(X)]$ than naively attempting to find $(\nabla_\theta \Psi(\theta))^{-1}$, which is an intractable problem. We will explore a simple iterative algorithm for solving this minimization problem in the next section.

4 Gradient Descent in the Canonical Parameter Space

Suppose we are given data $\{x_i\}_{i=1}^n$ where $x_i \in \Omega$ and a feature mapping $\Phi : \Omega \to \mathbb{R}^k$. We compute the empirical expectations of the feature vectors $\hat{\mu} = \frac{1}{n}\sum_{i=1}^n \Phi(x_i)$.
Given this situation, and our discussion of the Principal of Maximum Entropy in the 2nd section, we would like to find a probability distribution p^* which is the solution to the following optimization problem:

$$\begin{aligned} &\arg\max_p \quad H(p) \\ &\text{subject to } \mathbb{E}_p[\mathcal{T}] = \eta \end{aligned}$$

In the last section, we showed that this problem is equivalent to solving the optimization problem

$$\begin{aligned} &\arg\min_\theta \; g(\theta) = \Psi(\theta) - \langle \theta, \mu \rangle \\ &\text{subject to } \theta \in \Theta \end{aligned} \tag{3}$$

Note that in lemma 4 of section 3, we show that $\nabla_\theta \Psi(\theta) = \mathbb{E}p_\theta[\Phi(X)]$. Following effectively the same calculation twice, we could show that $\nabla^2_\theta \Psi(\theta) = \mathbb{E}_{p_\theta}[(\Phi(X))^2]$. Thus, $\nabla^2_\theta \Psi(\theta)$ is strictly positive, meaning $\nabla^2_\theta g(\theta)$ is strictly positive and therefore is a convex function. Thus, we can rely on strictly gradient based optimization techniques and expect them to converge well.

Here use gradient descent to attempt to solve this optimization problem. **Algorithm 1** describes the implementation used.

4.1 Simulations

We test our algorithm on some standard statistical distributions to test its effectiveness. For all of the following, we will use $\Phi_k(x) = [x, x^2, \ldots, x^k]^T$ for the one dimensional case, and $\Phi_k(x) = [x_1, x_2, x_1^2, x_2^2, \ldots, x_1^k, x_2^k]^T$ for the two dimensional case.

4.1.1 Standard Normal Distribution

We draw n samples $\{x_i\}_{i=1}^n$ *iid* from $\mathcal{N}(0, 1)$, and form a data matrix $X = (x_1|\ldots|x_n)$. Here, we show the results of $GD(X, \Phi_2, 250, 1, 2000)$.

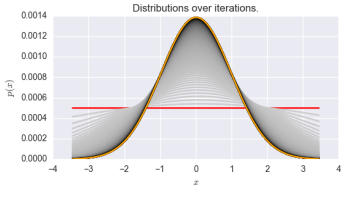

Figure 1: Distributions varying with t. Darker = later iterations, red = first iteration, orange = last iteration.

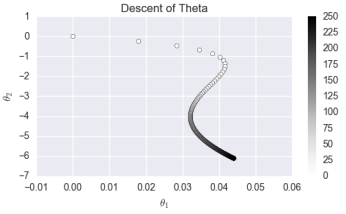

Figure 2: Parameter θ_t varying smoothly on the manifold Θ as t increases.

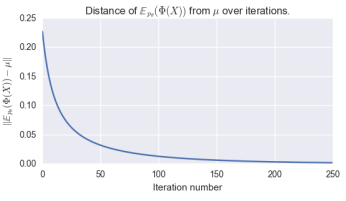

Figure 3: $\mathbb{E}_{p_{\theta_t}}[\Phi(X)]$ gets closer to $\hat{\mu}$ as t increases.

4.1.2 Bernoulli 0.9 Distribution

We draw n samples $\{x_i\}_{i=1}^n$ *iid* from $\mathcal{B}(0.9)$, and form a data matrix $X = (x_1|\ldots|x_n)$. Here, we show the results of $GD(X, \Phi_6, 1000, 1, 2000)$.

Algorithm 1 Gradient Descent for Maximum Entropy Modeling (GDME)

Require:

- $X = \frac{1}{n}\{x_i\}_{i=1}^n, x_i \in \mathbb{R}^d$ the sample points.
- $\Phi : \Omega \to \mathbb{R}^k$ the feature mapping.
- $T \in \mathbb{Z}_{\geq 0}$: the number of iterations of gradient descent to take.
- d the dimension of our support set Ω.
- N a sample domain size.

Ensure:

$\hat{\theta^*} \in \mathbb{R}^k$, the predicted optimal canonical parameter for the probability distribution of X.

procedure GD(X, Φ, T, d, N)
  $\mu = \sum_{i=1}^n x_i$ ▷ $\mu \in \mathbb{R}^d$ is the mean of the raw data.
  $\overline{X} = \{x_i - \mu\}_{i=1}^n$ ▷ Mean center.
  $M = \max_{i \in [n]} \|x_i - \mu\|$
  $\tilde{X} = \{\frac{x_i - \mu}{M}\}_{i=1}^n$ ▷ Normalize to unit ball.
  $\hat{\mu} = \frac{1}{n}\sum_{i=1}^n \Phi(\frac{x_i - \mu}{M})$ ▷ $\hat{\mu} \in \mathbb{R}^k$ is the mean of the feature vectors.
  Set $\theta_0 = \vec{0}$ ▷ Initialize to uniform distribution.
  Sample $\Omega = \{\omega_i\}_{i=1}^N$ ▷ ω_i are sampled uniformally from $\mathcal{B}^d$.
  Compute $\Psi_0 = \sum_{x \in \Omega} e^{\langle \theta_0, \Phi(x) \rangle}$
  Set $p_0(x) = e^{\langle \theta_0, \Phi(x) \rangle - \Psi_0}$
  Compute $\mu_0 = \sum_{x \in \Omega} p_0 \Phi(x)$.
  Compute $\nabla_0 = \mu_0 - \hat{\mu}$
  for $t := 0, \ldots, T-1$ **do**
    Set $\theta_{t+1} = \theta_t - \nabla_t$
    Compute $\Psi_{t+1} = \sum_{x \in \Omega} e^{\langle \theta_{t+1}, \Phi(x) \rangle}$
    Compute $p_{t+1}(x) = e^{\langle \theta_{t+1}, \Phi(x) \rangle - \Psi_{t+1}}$
    Compute $\mu_{t+1} = \sum_{x \in \Omega} p_{t+1} \Phi(x)$.
    $\nabla_{t+1} = \mu_{t+1} - \hat{\mu}$
  end for
  return θ_T.
end procedure

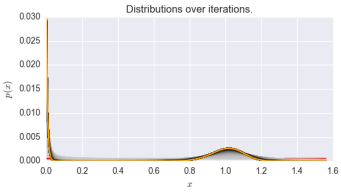

Figure 4: Distributions varying with t. Darker = later iterations, red = first iteration, orange = last iteration. Note here that there is more area at 1, though the spike at 0 is high.

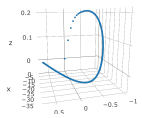

Figure 5: Parameter θ_t varying smoothly on the manifold Θ as t increases. Projected onto first 3 singular directions of $\Theta_T = (\theta_1 | \ldots | \theta_T)$

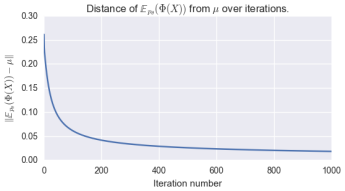

Figure 6: $\mathbb{E}_{p_{\theta_t}}[\Phi(X)]$ gets closer to $\hat{\mu}$ as t increases.

4.1.3 Uniform (-0.5, 0.5)Distribution

We draw n samples $\{x_i\}_{i=1}^n$ *iid* from $\mathcal{U}(-0.5, 0.5)$, and form a data matrix $X = (x_1 | \ldots | x_n)$. Here, we show the results of $GD(X, \Phi_{200}, 100, 1, 2000)$.

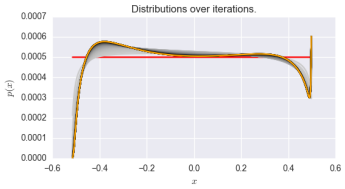

Figure 7: Distributions varying with t. Darker = later iterations, red = first iteration, orange = last iteration.

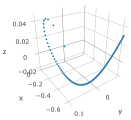

Figure 8: Parameter θ_t varying smoothly on the manifold Θ as t increases. Projected onto first 3 singular directions of $\Theta_T = (\theta_1 | \ldots | \theta_T)$

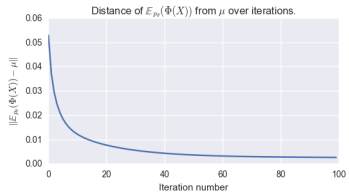

Figure 9: $\mathbb{E}_{p_{\theta_t}}[\Phi(X)]$ gets closer to $\hat{\mu}$ as t increases.

5 Results

As part of our data exploration, we test our algorithm on various classification tasks. In the first task, we try to classify word to the documents they belong to. In the second task, we test our algorithm on the canonical MNIST digit recognition task. All programs were written in python.

5.1 Text classification

For our data, we use a collection of abstracts from the 2011 Neural Information Processing Systems (NIPS) and Association for Computational Linguistics (ACL) conferences. We first parsed every sentence and made word embeddings for every word using a Word2Vec[1] model trained on the vocabulary of both collections. We then further pruned the data to throw away common words in both collections and those that appeared more than 20 times (we also report results on keeping all words or just throwing away words with high frequency). Using the GDME algorithm with 1000 iterations and 2 constraints[2], we fit the data to its maximum entropy distribution in the exponential family. After training we have a set of natural parameters $\theta_{ACL}, \theta_{NIPS}$ for each collection.

To classify words, we compute the probability of each word, x, being in a collection and classify it according to the maximum likelihood.

$$\hat{y}(x) = \arg \min_{\theta \in \{\theta_{ACL}, \theta_{NIPS}\}} -\log P_\theta(x|\mathcal{D})$$

We use 2-fold cross validation to split the data into test-train set and report the average classification accuracy across all splits. Accuracy was measured as (words classified correctly)/(total number of words). We report results in three different settings: (1) using all words in both collections, (2) using words that appear less than 20 times across both collections, (3) only using words that don't appear in both collections (in other words only using the unique words).

Word Classification	All-words	Pruned-words	Unique-words
NIPS	32.8%	85.3%	79.6%
ACL	66.5%	16.4%	63.7%
Both	50%	52%	69%

Figure 10: Text Classification task accuracies. We show results when we classify all words in both collections (All-words), only words that appear less than 20 times (Pruned-words), and only words that are unique to either collection (Unique-words).

Our results are shown in Figure 4 and the confusion matrix in Figure 10. As it shows, our algorithm performs poorly on this task. The main reason we believe our algorithm does poorly is due to the non-separability of the data as shown in Figure 5 and 6. These figures show the data is much more separable and spread out in the unique words case because the number of words is much smaller and we've kept only words unique to a collection. Intuitively we hoped this data would be separable since words unique in one collection would capture semantic or contextual meaning for that collection.

[1] Word2vec is a 2-layer neural network trained on a corpus. It uses the context surrounding words to assign each unique word a unique vector. Words that are similar like 'man' and 'king' will have shorter euclidean distance.

[2] We tried different parameter settings and did not find a difference in performance.

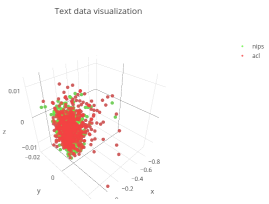

Figure 11: Visualization of all unique words after projecting onto first 3 eigenvectors.

Figure 12: Visualization of all words after projecting onto first 3 eigenvectors.

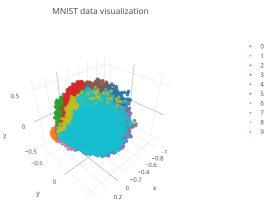

Figure 13: Visualization of MNIST digits after projecting onto first 3 eigenvectors

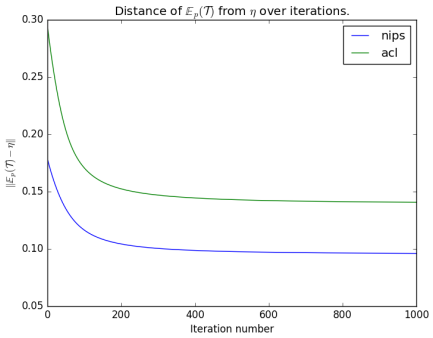

Figure 14: Text classification: Distance from the true moments to the estimated moments for each class.

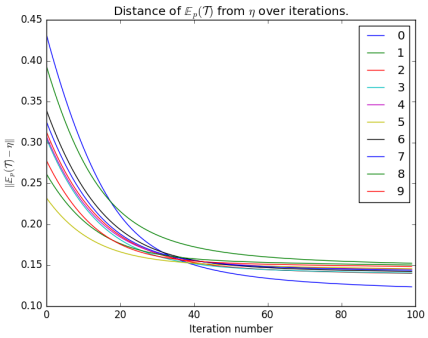

Figure 15: Digit classification: Distance from the true moments to the estimated moments for each class.

Unfortunately this discriminability was not as evident as we hoped. But we see that in the setting for unique words, the data is much more discriminable and achieves better results.

Lastly in Figure 4 it's interesting to note how the accuracies change across settings for each individual collection. When using all words the algorithm performs better on ACL but when using pruned words it performs better on NIPS. We believe it's most likely with how the data cluster changes in each setting. In Figure 6, we see ACL words overshadowing most of the NIPS points whereas in Figure 5 the ACl and NIPS data points are more evenly distributed.

Another evaluation of how well our algorithm learned the distributions underlying the data is by seeing the distance from the learned moments from the true moments of the data. Figure 8 shows how well the moments of each collection is learned over 1000 iterations in the unique words setting. As we see the algorithm learns the moments better for NIPS than ACL. This matches our results in Figure 4 where the algorithm performs better on NIPS.

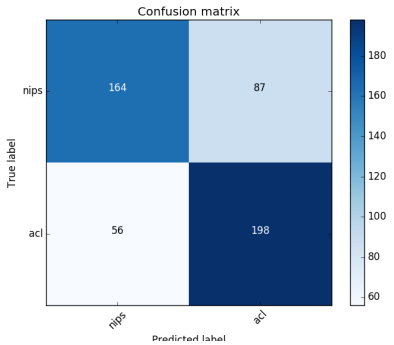

Figure 16: Text classification: Confusion matrix using only unique words

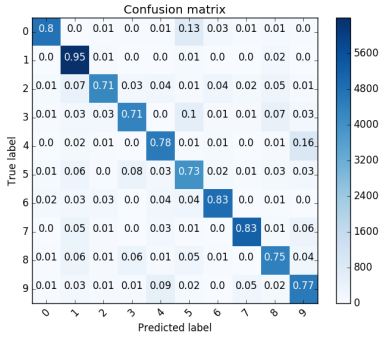

Figure 17: Digit classification: Confusion matrix

Classes	0	1	2	3	4	5	6	7	8	9	all
Accuracy	78.1%	95.5%	72.4%	72.7%	79.7%	66.0%	83.0%	81.8%	73.0%	78.1%	78.37%

Figure 18: Accuracy on classifying MNIST digits. Each column is the accuracy for each digit class with the last column being the overall accuracy for all classes.

5.2 Digit classification

In this task, we classify digits from the MNIST data set[3]. We first reduced the dimensionality of each image by running SVD on all the data and projecting each image onto the first 20 eigenvectors. A visualization of the projection onto the first 3 eigenvectors is given in Figure 5. Then we separated each image into its respective classes $C_i = \{\text{all images with label i}\}$. For each class we run GDCP and learn the natural parameters for that class. This gives us $\theta_{C_0}, ..., \theta_{C_9}$, the natural parameters for each class.

To classify images we do calculate the log likelihood of each image for each distribution and pick the maximum likelihood:

$$\hat{y}(x) = \arg \min_{\theta \in \{\theta_{C_0}, ..., \theta_{C_9}\}} -\log P_\theta(x|\mathcal{D})$$

MNIST provides us with a train and test split to use so we report our accuracies on the given test set. Accuracy is defined as (# digits classified correctly)/(total number of digits). We used 2 constraints and 100 iterations. Our results are in figure 12 and the confusion matrix in 11. As we see the algorithm performs decently on the digit recognition task but nowhere near state of the art. We see that the algorithm does very well on some digits compared to others. The classes the algorithm does poorly on are the more ambiguous digits like 3 and 5 which have lots of similar parts and often got classified as each other as we see in the confusion matrix. Running the algorithm with more constraints and more iterations slightly increased accuracy but not noticeably.

Figure 9 shows the distance between the learned moments and the true moments. Digits we do better on have lower error (distance) and match our results in Figure 12.

5.3 Discussion

As discussed in the previous section, our classification algorithm performs well on stimulated data when the samples are from known exponential family distributions. We explored how well our algorithm performs on arbitrary distributions formed by the data of our text or digit classification tasks. We found our algorithm to be underperforming compared to state of the art methods but shows that it is able to learn something about the data and perform some sort of classification. It performs better when the data has structure that resembles some sort of exponential family distribution as evident by it performing better on digits than text. Figure 4 shows the clusters of each digit form some sort of cluster the algorithm can learn.

An interesting possibility for future work is to come up with an unsupervised learning scheme for our algorithm. In our work, we use labeled data to learn distributions for each class but incorporating our algorithm into something like Mixture Models would be interesting to explore. We also did not have time to explore any generative properties of our model. After fitting the data to different exponential family distributions it would be interesting to sample from it and explore what the data would look like.

6 Contributions

Nitin wrote the theory related to information geometry with some contributions from Jason. Ryan wrote the theory related to the algorithm and optimization problem. Ryan implemented the gradient descent for maximum entropy algorithm and ran the simulations (section 4.1). Jason did the data exploration with some contributions from Nitin. He wrote the programs to use Ryan's code and run experiments on MNISt and the text corpus.

[3] http://yann.lecun.com/exdb/mnist/

7 References

Our main references was the Information Processing course at Carnegie Mellon University. The lectures pertinent to the theory aspect of our project were:

http://www.cs.cmu.edu/~aarti/Class/10704/lec1-entropy.pdf

http://www.cs.cmu.edu/~aarti/Class/10704/lec2-dataprocess.pdf

http://www.cs.cmu.edu/~aarti/Class/10704/lec3-maxent.pdf

http://www.cs.cmu.edu/~aarti/Class/10704/lec4-maxent2.pdf

http://www.cs.cmu.edu/~aarti/Class/10704/lec5-burg.pdf

And slides from a separate course:

http://www.cs.huji.ac.il/~shashua/papers/class3-ML-MaxEnt.pdf

The algorithm drew inspiration from these links:

https://www.renyi.hu/~csiszar/infgeom.pdf

http://web.mit.edu/6.291/www/handouts/csiszar_Idiv.pdf

http://curtis.ml.cmu.edu/w/courses/index.php/Alternating_Minimization

Lastly the MNIST dataset was obtained from Yann Lecunn's site and the ACL/NIPS abstracts were used from an assignment Jason worked on for a separate course.