Centro de Investigación en Matemáticas, A.C.

DISTRIBUTION REGRESSION AND ITS APPLICATION TO ECOLOGICAL INFERENCE

T E S I S

Que para obtener el grado de

Maestra en Ciencias

con Orientación en

Ciencias de la Computación y Matemáticas Industriales

Presenta

Lizette Lemus González

Director de Tesis:

Dr. Johan Van Horebeek

Autorización de la versión final

Guanajuato, Gto., 11 de febrero de 2019

Abstract

Distribution Regression is about learning a relation between probability distributions and real-valued response variables. Frequently, each distribution is only observed through a sample. There are a wide variety of applications in which objects are represented as a collection of its components. For example, an image can be represented as a set of local descriptors, a 3D object as a set of coordinates and a text as a set of words.

In this thesis we will focus on a novel application of Distribution Regression to predict voting behavior for demographic subgroups when we only have access to group level data proposed by Flaxman et al.(*Who supported Obama? Ecological Inference through Distribution Regression*, KDD 2015). This problem is known as Ecological Inference. The Ecological Inference problem has been subject to controversy since it is known about it and there aren't many sources of clear information about the methods to solve it. We provide a historical review of the solutions that have been proposed, starting with the classic solutions and including the most recent advances.

To present a solution to the Distribution Regression problem, we use a framework based on kernel mean embeddings of distributions of the Gaussian kernel and Ridge regression introduced by Szabo et al.(*Two-stage sampled learning theory on distributions*, AISTATS 2015). The objectives of this thesis are to understand the Gaussian kernel based similarity, to propose alternative similarity measures in order to improve the prediction results and to perform some experiments to compare these methods.

We propose three similarity functions: the pyramid match kernel, the marginal kernel and the Wasserstein kernel. We also present an alternative to the kernel methods using neural networks. We generate two synthetic datasets to compare these methods in quality of prediction, computational time and parameter selection.

Finally, we selected two methods to perform Ecological Inference for the US 2016 Presidential Elections and we show that Distribution Regression is a suitable approach to solve the Ecological Inference problem.

Acknowledgements

First of all, I would like to thank my advisor Dr. Johan Van Horebeek for allowing me to work with him and guiding me through the process of writing this thesis. Dr. Van Horebeek deeply cared about my academic development and I thank him for his helpful advice, patience, dedication and support.

Besides my advisor, I would like to thank my reviewers, Dr. Oscar Dalmau and Dr. Leticia Ramirez, for taking the time to read and evaluate my thesis.

I would also like to express my gratitude to my family. To my parents and brother, for their unconditional love and support. To my husband, for listening to my ideas and encouraging me to work on them, but above all for the joy that he brings to my life.

Thanks to the friends and classmates that I met at CIMAT, for making this stage of my life more enjoyable. Specially, I would like to thank Fernanda Lemus, for the honest conversations and the times we spent together studying and celebrating. Thanks to the Improv group, for all the fun moments that we shared.

Lastly, thanks to CONACYT for the economic support and to the professors of CIMAT who gave me the opportunity to be part of the program.

Contents

Chapter 1

Introduction

Distribution Regression is about learning a relation between a probability distribution P and a response variable Y. Frequently, we don't have access to P, instead we observe the distribution through a sample.

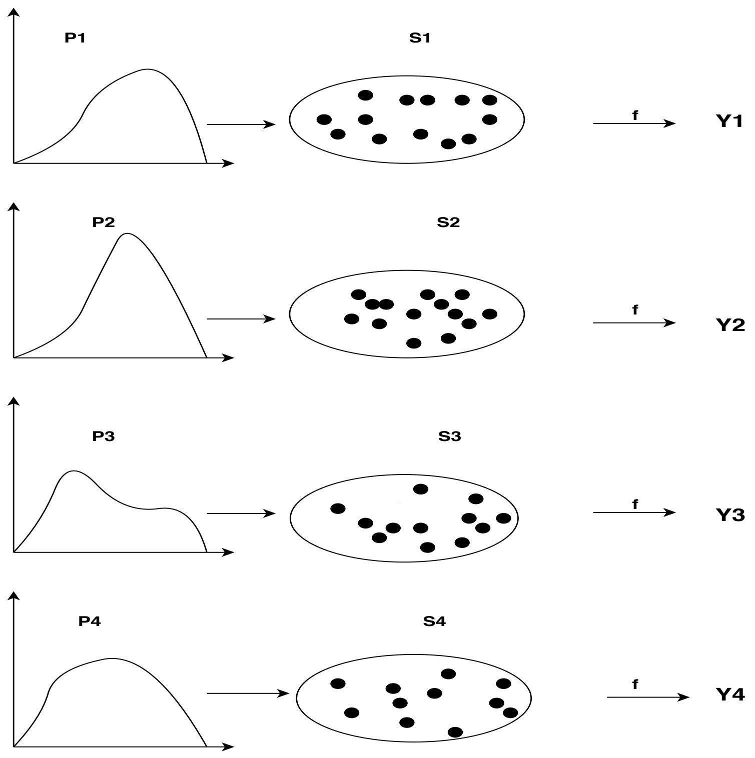

Figure 1.1: Distribution Regression.

The problem of Distribution Regression is not as well known as its counterpart, classic regression, in which we find a relation between a response variable and a predictor variable. However, there are a wide variety of applications in which we would like to obtain information about a data object that is represented with a collection of its components. For example:

- Classifying images represented by local descriptors of interest points [1].
- Learning mass measurements of galaxy clusters using their velocity dispersion[2].
- Classifying documents, where each document is represented as a set of words [3].

- Classifying 3D geometric data, where each object is represented as a point cloud [4].
- Predicting voting behavior for demographic subgroups of geographical regions, where each region is represented as a set of individuals [5].

Motivated by [5], in this thesis we will focus on the last application mentioned, the problem of predicting voting behavior for demographic subgroups of geographical regions, also known as Ecological Inference (EI).

The Ecological Inference problem can be explained with an example: suppose that we have n regions, and for each region we have a contingency table with the total of votes per category and per demographic subgroup, the quantities in the intersection are missing and the problem is to infer them. To do this, it is assumed that regions have something in common and we can learn a general model from all of them.

	Democrats	Republicans	
Men	?	?	1500
Women	?	?	2000
	2200	1300	

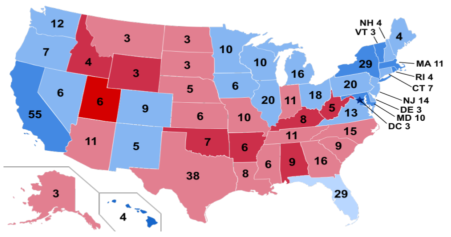

Figure 1.2: Learn a general model from all the regions.

1.1 Scope of the thesis

In this thesis we will use the following notation to describe our data:

$$\{(\{x_1\}, y_1), ..., (\{x_n\}, y_n)\}$$

where $\{x_k\}$ is a sample with N_k elements from a distribution X_k and y_k is its corresponding response variable.

To walk through the problem and its possible solutions, we consider first the simplest case. Let $X_k \sim Bernoulli(\theta_k)$. Assume that we have n samples drawn from this group of distributions $\{\{x_1\}, \{x_2\}, ...\{x_n\}\}$ and their response variables $\{y_1, y_2, ...y_n\}$.

In order to model the relation between $\{x_k\}$ and the response y_k we would like to find $\psi(\{x_k\})$, a value that condenses all the information about the distribution and fit a regression model afterwards.

In this particular example, we can intuitively propose $\psi(\{x_k\}) = \hat{\mu_k} = \frac{\sum_{j=1}^{N_k} x_k^j}{N_k}$, assuming that N_k is sufficiently large and therefore its empirical mean will converge to its real mean θ_k.

Extending the above to e.g. the continuous case is far from obvious, candidates for $\psi(\{x_k\})$ could be based on $\widehat{E(X_k)}, \widehat{E(X_k^2)}, \widehat{E(X_k^3)}$ or $\widehat{Var(X_k)}$. In general, for any distribution, $\psi(\{x_k\})$ should be a vector. If we are able to find a function ψ that transforms every distribution into a characteristic vector, then this problem would turn into a classic regression problem:

$$y_k = \alpha + \beta^T \psi(\{x_k\}) + \epsilon$$

Solving it in a linear way would require to specify ψ. On the contrary, if we use the kernel formulation of regression it is sufficient to specify the kernel matrix $[K_{ab}]$. As we will explain later, K_{ab} is a similarity measure between samples of distributions X_a and X_b. We use the work presented in [5] as our starting point, in which K_{ab} is defined through the kernel mean embedding of the distributions.

We dedicate Chapter 2 to explain the solution to the Distribution Regression in the framework of kernel mean embeddings and kernel Ridge regression. As we will see later, in [5] they use a Gaussian kernel to calculate the mean embeddings. We develop an explicit expression for the similarity between Gaussian distributions using Gaussian kernel mean embeddings; this expression will allow us to understand the similarity in terms of the mean and variance of the distributions.

In chapter 3 we explain the EI problem and give some historical context about it. Then, we describe some of the classic solutions and discuss their limitations. Finally we explain the solution using Distribution Regression and compare it with a more recent solution based on Learning with Label Proportions.

In chapter 4 we explore alternatives to calculate similarity measures between samples of distributions. We present an additive approach to calculate kernels for multidimensional distribution and show that for kernels based on mean embeddings, adding the kernels per dimension is equivalent to concatenating the explicit featurization. Then we describe the procedure to build the Pyramid Match Kernel, a similarity measure based on multi resolution histograms. We talk about Distance Substitution kernels and define a kernel based on the Wasserstein distance. We analyze the behavior of these similarity measures using Gaussian distributions.
In the second part of this chapter, we explore another framework to solve the Distribution Regression problem using a neural network formulation.

In chapter 5 we compare the performance of the methods presented in Chapter 4. We construct two synthetic datasets. The first one allows us to evaluate the methods in the general context of Distribution Regression while the second one is useful for evaluating the methods in the context of Ecological Inference. Finally, we use a neural networks formulation to perform Ecological Inferences on the voting results for the 2016 US Elections.

Chapter 2

Distribution Regression through kernel based mean embeddings

In this chapter we explain the solution to the Distribution Regression problem in the framework of mean embeddings and Ridge regression. We talk about general concepts of kernel functions and we also explain how to find an explicit featurization to approximate a kernel. Finally, we develop an expression for the similarity between Gaussian distributions in terms of their mean and variance.

2.1 Solution using kernel mean embeddings

Suppose we have the dataset:

$$\{(\{x_1\}, y_1), ..., (\{x_n\}, y_n)\}$$

where the *bags* $\{x_k\}$ are samples with N_k elements from a distribution X_k and y_k is its corresponding response variable.

We use the work presented in [5] as our starting point, in which they present a general framework to solve the Distribution Regression problem. The solution can be explained in two stages. First, build a similarity measure between samples of two distributions, denoted by K_{ab}. Second, fit a regression model using this similarity measure.

The first stage of the solution is to define K_{ab}:

$$K_{ab} =< \psi(X_a), \psi(X_b) >$$

where $\psi(X_a)$ is of the form:

$$\psi(X_a) = E_{x \sim X_a}[\phi(x)]$$

$\psi(X_a)$ is called the *kernel mean embedding* of distribution X_a and frequently denoted as μ_a.
ϕ is known as a *kernel map.*

As we explained in the introduction, we would like that $\psi(X_a)$ condenses all the information about the distribution. The concept of *characteristic kernel* formally defines this property. A kernel is called *characteristic* if the map $\mu : X \to \mu_X$ is injective, i.e. any two distributions with a difference will be mapped to different points.

In practice, the real distribution is unknown and it is only observed through a sample, therefore, an empirical estimator for kernel mean embedding of the distribution is used.

$$\psi(\{x_a\}) = \frac{1}{N_a}\sum_j \phi(x_a^j)$$

If we expand the equation for K_{ab} using the empirical estimator for the kernel mean embedding we obtain:

$$\begin{aligned}K_{ab} &=< \psi(\{x_a\}), \psi(\{x_b\}) > \\ &=< \frac{1}{N_a}\sum_j \phi(x_a^j), \frac{1}{N_b}\sum_l \phi(x_b^l) > \\ &= \frac{1}{N_a N_b}\sum_j\sum_l < \phi(x_a^j), \phi(x_b^l) > \\ &= \frac{1}{N_a N_b}\sum_j\sum_l k(x_a^j, x_b^l)\end{aligned}$$

k is a kernel function between individual observations.

In the next section we explain some general properties of kernel functions that will be used in this thesis.

2.1.1 Properties of kernels

For x, y in $\mathbb{R}^d$, there are some cases in which $\phi(x)$ can be explicitly defined. For example: If we use a linear kernel map $\phi(x) = x$ then

$$k(x, y) =< x, y >$$

and

$$< \psi(\{x_a\}), \psi(\{x_b\}) >= \frac{1}{N_a N_b}\sum_{ij} < x_a^i, x_b^j >$$

If we use the second moment $\phi(x) = x^2$, where $x^2 = (x_1^2, x_2^2, ..., x_d^2)$, then

$$k(x, y) =< x^2, y^2 >$$

and

$$< \psi(\{x_a\}), \psi(\{x_b\}) >= \frac{1}{N_a N_b}\sum_{ij} < (x_a^i)^2, (x_b^j)^2 >$$

However, it is possible to work with k without defining ϕ, this is also known as the *kernel trick*. As explained in [6], if k is *symmetric* and *positive definite* there always exists ϕ for which $k(x, y) =< \phi(x), \phi(y) >$.

A kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is *symmetric* if

$$k(x, y) = k(y, x)$$

for any pairs $x, y \in \mathbb{R}^d$

A symmetric kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is *positive definite* if

$$\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$$

for $n \in \mathbb{N}$, any choice of $x_1, ..., x_n \in \mathbb{R}^d$ and $c_1, ..., c_n \in \mathbb{R}$.

An important remark is that the kernel trick does not only apply for Euclidean data, it can be used in any domain on which it is possible to define a positive definite kernel.

If $k(x, y) = k(x - y)$ the kernel is called *translation invariant.* In the next section we will see that this property is useful to find an explicit featurization that approximates the kernel mean embedding.

An example of a kernel that is characteristic, positive definite and translation invariant is the Gaussian kernel also known as radial basis function kernel (RBF), defined as:

$$k(x, y) = exp(-\frac{||x - y||^2}{2\sigma^2})$$

Using the Gaussian kernel, the similarity measure between samples of distributions will be:

$$K_{ab} = \frac{1}{N_a N_b} \sum_j \sum_l exp(-\frac{||x_a^j - x_b^l||^2}{2\sigma^2})$$

we choose σ using the *median heuristic.* This heuristic consists in defining σ as the median of the pairwise distance between elements of the sets.

Finally, the matrix formed with all the pairwise similarity measures K_{ab} will be denoted by $\mathbb{K}$ and is known as the *Gram matrix.*

2.1.2 Kernel Ridge regression

Once K_{ab} is defined, the second stage of the solution is to fit a regression model. In [5] they propose to use Gaussian Process Regression to incorporate spacial data to the regression. For simplicity, in this thesis we will use Kernel Ridge Regression as presented and analyzed in [7].

Given our training data:

$$\{(\{x_1\}, y_1), ..., (\{x_n\}, y_n)\}$$

Define:

$$\mathbf{X} = \begin{bmatrix} \psi(\{x_1\}) \\ \psi(\{x_2\}) \\ \vdots \\ \psi(\{x_n\}) \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

The problem of Kernel Ridge Regression is to find $\hat{\beta}$ such that:

$$\hat{\beta} = \arg\min(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda||\beta||^2$$

where λ is a regularization term that can be selected using a validation set.

According to [8], the solution is given by:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda I_d)^{-1}\mathbf{X}^T\mathbf{y}$$

the elements of $\hat{\beta}$ are known as the *primal variables.*

Using the primal variables, the prediction for a new sample $\{x_*\}$ is:

$$\hat{\mathbf{y}}^* = \psi(\{x_*\})\hat{\beta}$$

In order to calculate the previous solution, it is necessary to know the explicit featurization ψ. However, the solution can be rewritten so that it is only expressed in terms of the similarity functions as follows:

$$\hat{\beta} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda I_n)^{-1}\mathbf{y}$$

where $\mathbf{X}\mathbf{X}^T = \mathbb{K}$ is the Gram Matrix.

If we define:

$$\begin{gathered}\mathbf{k}^* = \psi(\{x_*\})\mathbf{X}^T \\ = [< \psi(\{x_*\}), \psi(\{x_1\}) >, < \psi(\{x_*\})\psi(\{x_2\}) >, ..., < \psi(\{x_*\}), \psi(\{x_n\}) >] \\ = [K_{*1}, ..., K_{*n}]\end{gathered}$$

then the prediction for a new sample $\{x_*\}$ will be:

$$\begin{aligned}\hat{mathbfy}^* &= \mathbf{k}^*(\mathbb{K} + \lambda I_n)^{-1}\mathbf{y} \\ &= \mathbf{k}^*\alpha \\ &= [K_{*1}, ...K_{*n}][\alpha_1, ..., \alpha_n]^T\end{aligned}$$

the elements of α are known as the *dual variables.*

In chapter 4 we will explore different alternatives to define K_{ab} and use this property to fit our regression models.

2.1.3 Approximating K

In order to find the solution in terms of similarity functions we need to calculate $\mathbb{K}$. As we explained before, we will use the Gaussian Kernel to calculate the entries of this matrix as follows:

$$K_{ab} = \frac{1}{N_a N_b} \sum_j \sum_l e^{-\frac{||x_a^j - x_b^l||^2}{2\sigma^2}}$$

Assuming that all the bags have the same number of elements $N_i = N$, computing each of the n^2 entries of $\mathbb{K}$ requires N^2 operations. In our real data example, $n \approx 900$ and $N \approx 10,000$, this means that we require approximately 81×10^{12} operations to calculate $\mathbb{K}$.

As described in [5], finding an explicit feature representation $z : \mathbb{R}^d \to \mathbb{R}^D$ such that $z(x)^T z(y) \approx k(x,y)$ reduces the number of operations even if the representation is high dimensional.

In [9] they present a technique to approximate a shift invariant positive definite kernel k using an explicit low dimensional map by projecting the data into a randomly selected line and then applying a cosine function and scaling.

This result is based on Bochner's Theorem [10], which guarantees that if a shift invariant kernel $k(\delta)$ is positive definite and properly scaled then its Fourier transform $p(\omega)$ is a proper probability distribution. Defining: $\zeta(x) = e^{i\omega^T x}$

$$k(x-y) = \int_{\mathbb{R}^d} p(\omega) e^{i\omega^T(x-y)} d\omega = E[\zeta(x)\overline{\zeta(y)}]$$

Since $p(\omega)$ and $k(\delta)$ are real, the integral gives the same result when the exponentials are replaced with cosines. It can be proved that setting $z_\omega(x) = \sqrt{2}cos(\omega^T x + b)$

$$k(x,y) = E_\omega[z_\omega(x) z_\omega(y)]$$
$$\omega \sim p(\omega) \text{ y } b \sim U(0, 2\pi)$$

An approximation for the kernel is found by taking an empirical estimator of $E_\omega[z_\omega(x)z_\omega(y)]$, which in this case is the average over D samples.

$$\begin{aligned} k(x,y) &\approx \frac{1}{D} \sum_{j=1}^{D} z_{\omega j}(x) z_{\omega j}(y) \\ &= z(x)^T z(y) \end{aligned}$$

where $z(x) = [z_{\omega 1}(x), ..., z_{\omega D}(x)]$

Putting all together, we can calculate the explicit featurization z as follows:
1. Choose a positive definite shift invariant kernel k, in our case the Gaussian kernel:

$$k(\Delta) = e^{-\frac{||\Delta||^2}{2\sigma^2}}$$

2. Compute the Fourier transform of k. For the Gaussian kernel:

$$p(\omega) = (\frac{\sigma}{\sqrt{2\pi}})^D e^{-\frac{\sigma^2||\omega||^2}{2}}$$

3. Draw D iid samples $w_1, ..., w_D$ from p.
4. Draw D iid samples $b_1, ..., b_D$ from a uniform distribution in $[0, 2\pi]$.
5. Calculate $z(x) = [\sqrt{2}cos(\omega_1^T x + b_1), ..., \sqrt{2}cos(\omega_D^T x + b_D)]$.

We can use this feature representation to calculate the mean embeddings and afterwards the similarity function as:

$$\begin{aligned} K_{ab} &= \frac{1}{N_a N_b} \sum_j \sum_l k(x_a^j, x_b^l) \\ &\approx \frac{1}{N_a N_b} \sum_j \sum_l z(x_a^j)^T z(x_b^l) \\ &= [\sum_j \frac{1}{N_a} z(x_a^j)]^T \sum_l \frac{1}{N_b} z(x_b^l) \end{aligned}$$

The complexity of calculating each $z(x)$ is $O(dD)$. For each x the feature representation $z(x)$ only has to be calculated once and for each region we calculate the mean embedding $\sum_j \frac{1}{N} z(x^j)$ and store it in a vector. The complexity of calculating all the mean embeddings is $O(nNDd)$. Once we have the mean embeddings for each region, we can compute the Gram matrix using n^2 dot products between D dimensional vectors. So the total complexity of calculating $\mathbb{K}$ is $O(n^2 D + nNDd)$.

For our real data example the number of real variables d is equal to 19 and we will set D to be equal to 190. Then the approximate number of operations to compute $\mathbb{K}$ is reduced to 33×10^9.

Another advantage of having these explicit representations is that once we learn $\hat{\beta}$ we can find the prediction for a new bag using:

$$\hat{\mathbf{y}}^* = [\sum_j \frac{1}{N_*} z(x_*^j)]^T \hat{\beta}$$

Predicting using this equation requires approximately NDd operations to compute the mean embedding for the new bag and D operations to compute the dot product. As opposed to using the dual formulation in which it is necessary to calculate the similarity between each point in the training set and the new point using approximately $N^2 n$ operations and then n operations to perform the dot product.

$$\begin{aligned} \hat{\mathbf{y}}^* &= \mathbf{k}^* (\mathbb{K} + \lambda I_n)^{-1} \mathbf{y} \\ &= \mathbf{k}^* \alpha \\ &= [K_{*1}, ...K_{*n}][\alpha_1, ..., \alpha_n]^T \end{aligned}$$

2.2 Gaussian kernel mean embedding

In order to understand the similarity measure defined through the mean embeddings of a Gaussian kernel we calculate an explicit expression of $K(X,Y)$ for the case of normal random variables.
Let $X \sim N(\mu_X, \sigma_X)$ and $Y \sim N(\mu_Y, \sigma_Y)$, be independent random variables.

$$\begin{aligned}
K(X,Y) &= E[e^{-\frac{(X-Y)^2}{2\sigma^2}}] \\
&= E[e^{-\frac{(X-Y)^2}{2\sigma^2}}] \\
&= \int\int e^{-\frac{(x-y)^2}{2\sigma^2}} f_X(x) f_Y(y) dx dy \\
&= \int\int e^{-\frac{(y-x)^2}{2\sigma^2}} f_X(x) dx f_Y(y) dy \\
&= \int \sigma\sqrt{2\pi} \int \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-x)^2}{2\sigma^2}} f_X(x) dx f_Y(y) dy
\end{aligned}$$

Define Z: $Z \sim N(0, \sigma^2)$

$$\begin{aligned}
&= \int \sigma\sqrt{2\pi} \big[\int f_Z(y-x) f_X(x) dx \big] f_Y(y) dy \\
&= \sigma\sqrt{2\pi} \int f_{Z+X}(y) f_Y(y) dy \\
&Z + X \sim N(\mu_X, \sigma^2 + \sigma_X^2) \\
&= \sigma\sqrt{2\pi} \int \frac{1}{\sqrt{2\pi(\sigma^2+\sigma_X^2)}} e^{-\frac{(y-\mu_X)^2}{2(\sigma^2+\sigma_X^2)}} f_Y(y) dy \\
&= \sigma\sqrt{2\pi} \int \frac{1}{\sqrt{2\pi(\sigma^2+\sigma_X^2)}} e^{-\frac{(\mu_X-y)^2}{2(\sigma^2+\sigma_X^2)}} f_Y(y) dy
\end{aligned}$$

Define W: $W \sim N(0, \sigma^2 + \sigma_X^2)$

$$\begin{aligned}
&= \sigma\sqrt{2\pi} \int f_W(\mu_X - y) f_Y(y) dy \\
&= \sigma\sqrt{2\pi} f_{W+Y}(\mu_X) \\
&W + Y \sim N(\mu_Y, \sigma^2 + \sigma_X^2 + \sigma_Y^2) \\
&= \sigma\sqrt{2\pi} \big[\frac{1}{\sqrt{2\pi(\sigma^2+\sigma_X^2+\sigma_Y^2)}} e^{-\frac{(\mu_X-\mu_Y)^2}{2(\sigma^2+\sigma_X^2+\sigma_Y^2)}} \big] \\
&= \frac{\sigma}{\sigma^2+\sigma_X^2+\sigma_Y^2} e^{-\frac{(\mu_X-\mu_Y)^2}{2(\sigma^2+\sigma_X^2+\sigma_Y^2)}}
\end{aligned}$$

This expression allows us to understand the similarity in terms of the variances and the means.

In the next plot we show $K(X,Y)$ for Gaussian distributions X and Y as a function of μ_Y and σ_Y.

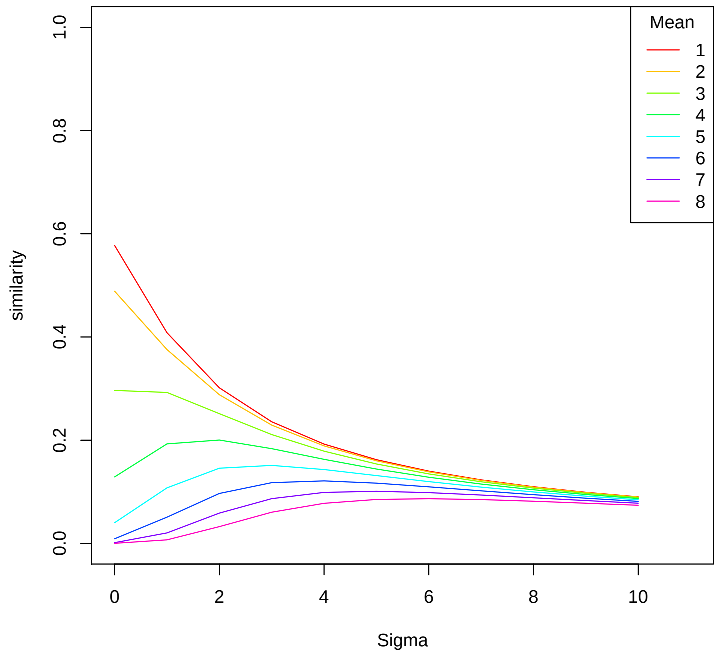

Figure 2.1: Similarity between Gaussian distributions.

We notice that as the mean distance increases the similarity decreases, as expected. However the same doesn't happen for the difference in sigma.

We made a similar exercise for the case of a Gaussian mixture:
Let $X_1 \sim N(\mu_{X1}, \sigma_{X1})$, $X_2 \sim N(\mu_{X2}, \sigma_{X2})$ and $Y \sim N(\mu_Y, \sigma_Y)$ be independent random variables. Let $X = \alpha X_1 + (1-\alpha)X_2$.

$$E[e^{-\frac{(X-Y)^2}{2\sigma^2}}]$$
$$= \int\int e^{-\frac{(x-y)^2}{2\sigma^2}} f_X(x) f_Y(y) dxdy$$
$$= \int\int e^{-\frac{(x-y)^2}{2\sigma^2}} (\alpha f_{X1}(x) + (1-\alpha) f_{X2}(x)) f_Y(y) dxdy$$
$$= \alpha \int\int e^{-\frac{(x-y)^2}{2\sigma^2}} f_{X1}(x) f_Y(y) dxdy$$
$$+ (1-\alpha) \int\int e^{-\frac{(x-y)^2}{2\sigma^2}} f_{X2}(x) f_Y(y) dxdy$$

Similar to the previous case, this equals:

$$= \alpha \frac{\sigma}{\sqrt{\sigma^2 + \sigma_{X1}^2 + \sigma_Y^2}} e^{-\frac{(\mu_{X1}-\mu_Y)^2}{2(\sigma^2+\sigma_{X1}^2+\sigma_Y^2)}} + (1-\alpha) \frac{\sigma}{\sqrt{\sigma^2 + \sigma_{X2}^2 + \sigma_Y^2}} e^{-\frac{(\mu_{X2}-\mu_Y)^2}{2(\sigma^2+\sigma_{X2}^2+\sigma_Y^2)}}$$

In this plot we illustrate the behavior of the similarity of X and Y, where X is a Gaussian Mixture and Y are Gaussians with variable means.

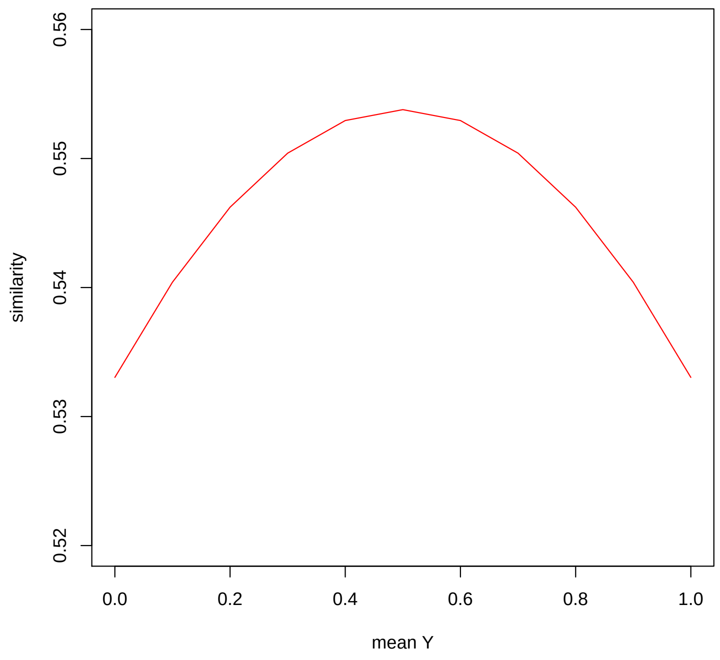

Figure 2.2: $X1 \sim N(0,1), X2 \sim N(1,1), Y \sim N(\mu_Y, 1), \alpha = 0.5$.

As expected, we observe that the point in which X and Y are more similar is when $\mu_Y = 0.5$.

Chapter 3

Ecological Inference

As we explained in the introduction, in this thesis we will use Distribution Regression to perform Ecological Inferences on the voting behavior for the 2016 Presidential Elections. The EI problem has been present in the political science literature for many years and despite the multiple attempts to solve it, it wasn't until 2015 when it was formalized as a machine learning problem in [5]. In this chapter we explain the EI problem and provide the historical context about it. Then we explain some of the solutions that have been proposed. First, we review some of the classic solutions, the method of bounds, Goodman's and King's methods. Second, we explain the solutions presented in the machine learning community, using Distribution Regression and using Learning with Label Proportions.

3.1 Problem statement

Ecological inference is the process of inferring individual behavior from grouped data. We illustrate the problem with a particular example:

Given n regions, we have the total percentages of men X_i and women $1 - X_i$ per region, as well as the total percentages of people that voted for candidate A, T_i, and people that voted for candidate B, $1 - T_i$. We assume that we don't know the joint distribution, therefore we want to infer the quantities β_i^M and β_i^W for each region.

	Candidate A	Candidate B	Total
Men	β_i^M	$1 - \beta_i^M$	X_i
Women	β_i^W	$1 - \beta_i^W$	$1 - X_i$
Total	T_i	1 - T_i	

3.2 Historical context

The EI problem emerged in political science and has been subject to controversy ever since. According to King [11], one of the first appearances was in the 1919, in a study of women voting behavior, when William Ogbutn and Inez Goltra wondered about the possibility of counting the women's votes using an indirect method. King mentions that there were other early works in which ecological inferences were made, even without recognizing the complexity of the problem. However, in 1950 Robinson warned not to use aggregate correlations to deduce individual correlations and in 1958 the concept of *Ecological fallacy* was introduced. The ecological fallacy occurs when using aggregate data to draw conclusions about individual behavior.

Two of the first most widely used methods were Goodman's method and the method of bounds. As we will see in the next section, each of them has certain limitations. In 1997 King proposed a Bayesian approach to solve the EI problem. However, this method also presents limitations and there were multiple criticisms about it which will be commented in the next section too.

Finally, in 2005 [5] the problem was formalized and an assumption of access to individual level data was added. This presented the opportunity to solve the problem using machine learning techniques. In [5] it is solved as an application of Distribution Regression in which each region is represented as a bag of individuals and the objective is to fit a model to predict the aggregated vote. While in [12] it is presented as a Learning with Label Proportions problem, in which each region is represented as a bag with a global label formed by some aggregation function on individual labels that we want to infer.

3.3 Classic methods

3.3.1 Method of bounds

The problem formulation generates the following restrictions:

$$\beta_i^M \in [max(0, \frac{T_i - (1 - X_i)}{X_i}), min(\frac{T_i}{X_i}, 1)]$$
$$\beta_i^W \in [max(0, \frac{T_i - X_i}{1 - X_i}), min(\frac{T_i}{1 - X_i}, 1)]$$

Reporting these intervals is known as the method of bounds and it was proposed by Duncan and Davis in 1953. Sometimes the intervals are narrow enough to provide meaningful information, however, there is no guarantee that this will be the case.

3.3.2 Goodman's method

Goodman's method is based on the identity:

$$T_i = X_i \beta_i^M + (1 - X_i)\beta_i^W$$

The method consists in running a regression of T_i in X_i and $1 - X_i$ to estimate the coefficients β, assuming that they are constant for all the regions. As a consequence, the main disadvantage of this method is that it only produces two general estimators β^M and β^W. Another problem is that there is no guarantee that the estimators will fall within the bounds. Lastly, the method is based on the assumption that the parameters and the X_i are not correlated, but this assumption is hardly verifiable.

3.3.3 King's method

King's method is based on a Bayesian approach presented in multiple stages. There are three observations and three assumptions that lead to the model. The first observation is that the points (β_i^M, β_i^W) are in $[0,1] \times [0,1]$. The second observation is that Goodman's equation can be arranged as the equation of a line:

$$\beta_i^W = \frac{T_i}{1 - X_i} - \frac{X_i}{1 - X_i}\beta_i^M$$

This means that when we know T_i and X_i the support of (β_i^M, β_i^W) becomes a line in the unit square. King represents these lines in what he calls a tomography plot. In this plot each line corresponds to all the possible values of the parameters for a region.

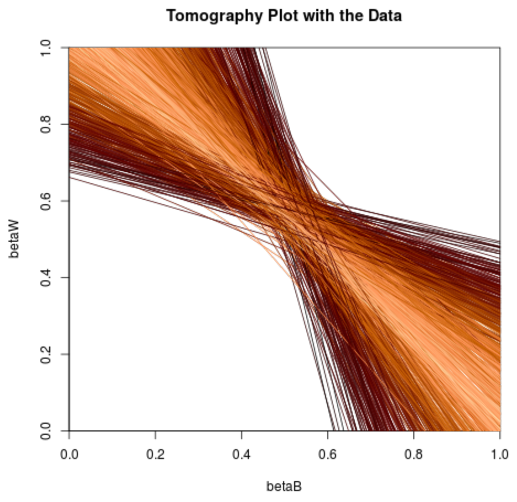

Figure 3.1: Tomography plot. Image generated with [13].

The third observation is what is known as the method of bounds:

$$\beta_i^M \in [max(0, \frac{T_i - (1 - X_i)}{X_i}), min(\frac{T_i}{X_i}, 1)]$$
$$\beta_i^W \in [max(0, \frac{T_i - X_i}{1 - X_i}), min(\frac{T_i}{1 - X_i}, 1)]$$

King makes the assumption that the parameters (β_i^M, β_i^W) are drawn from a bivariate normal distribution, conditional on X_i. Due to the first observation, this distribution is truncated in the unit square.

$$(\beta_i^M, \beta_i^W) \sim TN(B, \Sigma)$$

where $B = \begin{bmatrix} B^M \\ B^W \end{bmatrix}$, $\Sigma = \begin{bmatrix} \sigma_M^2 & \sigma_{MW} \\ \sigma_{WM} & \sigma_W^2 \end{bmatrix}$.

The truncated bivariate normal distribution can be represented with contours plots in the same square as the data.

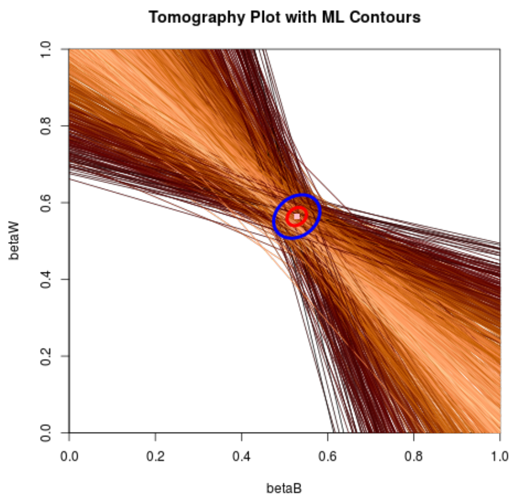

Figure 3.2: Tomography plot with contours. Image generated with [13].

The second assumption is that β_i^M and β_i^W are mean independent of X_i, i.e.

$$E(\beta_i^M|X_i) = E(\beta_i^M)$$
$$E(\beta_i^W|X_i) = E(\beta_i^W)$$

Finally, King assumes that the values of T_i are independent given X_i.

Using these observations and assumptions, the goal is to find a distribution of the parameters β_i^M and β_i^W conditioned to the T_is.

The first step is to estimate the parameters of the truncated bivariate normal distribution $\hat{\psi} = \{\hat{B}^M, \hat{B^W}, \hat{\sigma_M}, \hat{\sigma_W}, \sigma_{\hat{M}W}\}$ by the method of maximum likelihood using the function:

$$L(\hat{\psi}|T) \propto \prod_{X_i \in (0,1)} N(T_i|\mu_i, \sigma_i^2)\frac{S(B,\Sigma)}{R(B,\Sigma)}$$

where μ_i, σ_i, S and R are functions of B, Σ, X_i and T_i

Then King uses the multivariate normal as an approximation for the posterior distribution of the parameters:

$$P(\psi|T) \approx N(\hat{\psi}, V(\hat{\psi}))$$

where $V(\hat{\psi}))$ is the inverse of the matrix of second derivatives of the likelihood function.

An analytic expression is developed for the probability of β_i^M given T_i and $\hat{\psi}$:

$$P(\beta_i^M|T, \hat{\psi}) = N(\beta_i^M|B^M + \frac{w_i}{\sigma_i^2}\epsilon_i, \sigma_M^2 - \frac{w_i}{\sigma_i^2})\frac{1(\beta_i^M)}{S(B,\Sigma)}$$

where

$$w_i = \sigma_M^2 X_i + \sigma_{MW}(1 - X_i)$$
$$\epsilon_i = Ti - B^M X_i - B^W(1 - X_i)$$
$$S(B,\Sigma) = \int_{max(0,\frac{T_i-(1-X_i)}{X_i})}^{min(1,\frac{T_i}{X_i})} N(\beta_i^M | B^M + \frac{w_i}{\sigma_i^2}\epsilon_i, \sigma_M^2 - \frac{w_i}{\sigma_i^2}) d\beta^M$$

To find the estimates, King uses a simulation procedure that consists in drawing K values of $\hat{\psi}$ and plugging each one into the equation of $P(\beta_i^M|T,\hat{\psi})$ to draw K values of β_i^M. The average of this sample will be used as an estimate of the parameter β_i^M. We can find an estimate of the value β_i^W using the equation:

$$\beta_i^W = \frac{T_i}{1 - X_i} - \frac{X_i}{1 - X_i}\beta_i^M$$

The resulting estimates can be represented in the tomography plot.

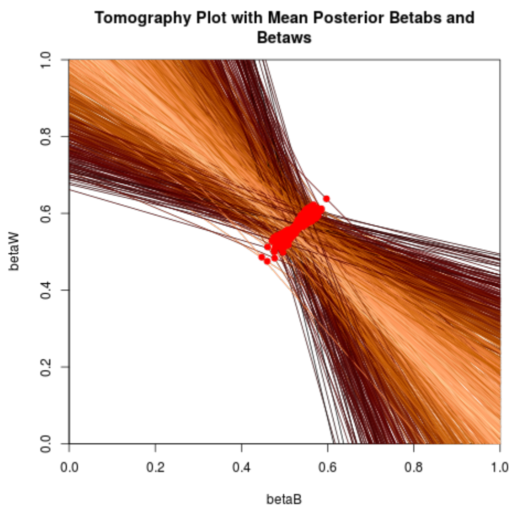

Figure 3.3: Tomography plot with estimated β_is. Image generated with [13].

To calculate a general estimate of β^M we calculate the weighted average of the β_i^M, considering the weights proportional to the size of the population per region.

Kings method can be generalized for RxC tables.

	Output 1	Output 2	$\cdots$	Output C	Total
Group 1	β_i^{11}	β_i^{12}	$\cdots$	β_i^{1C}	X_i^1
Group 2	β_i^{21}	β_i^{22}	$\cdots$	β_i^{2C}	X_i^2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Group R	β_i^{R1}	β_i^{R2}	$\cdots$	β_i^{RC}	X_i^R
Total	T_i^1	T_i^2	$\cdots$	T_i^C	

There are two strategies proposed in King's book. The first strategy consists in decomposing the table into a set of 2x2 tables and solving the problem for each of them with the procedure described before. This approach might be inconsistent, given that each parameter can be calculated in two different ways and the results might not always be similar. In the case of noticing the inconsistency another model should be applied.

The second strategy proposed in the book consists of following procedure analogue to the one presented for 2x2 tables. Assume that the data are drawn from a truncated multivariate normal distribution and find the generalized bounds for the parameters.

$$\beta_i^{rc} \in [max(0, \frac{T_i^c - (1 - X_i^r)}{X_i^r}), min(\frac{T_i^c}{X_i^r}, 1)]$$

Then calculate the parameters ψ of the distribution using maximum likelihood and approximate the conditional posterior distribution of the parameters. Develop an explicit expression for $P(\beta_i^{rc}|T, \psi)$ and use it to draw K samples of β_i^{rc}. The average over this sample will be the estimate of β_i^{rc}.

Comments on King's method

King's method is one of the most known methods in the ecological inference literature, however, since its release there have been multiple criticisms about it. The main issue is whether or not it is a solution to the ecological inference problem.

In 1998, Freedman, Klein, Ostland and Roberts made a review [14] that presented evidence of datasets in which the method gave incorrect results after validating the assumptions. It is important to mention that the methods proposed by King to evaluate the validity of the assumptions are empirical and this can lead to inaccurate results. For example, to verify the assumption of normality King proposes to observe what he calls tomography plots. None of the datasets used by King or Freedman are available for replication.

In May 2001, another article was published. It criticizes the lack of criteria used in the proposed method and points out some details in its development. They mention that the book is very difficult to read and that the method doesn't follow the formalism of a statistical model. Also, they argue that the distribution of $\beta_i|T_i$ is miscalculated, King assumes that β_i^M, β_i^W is bivariate normal then conditions and at the end truncates, without considering that β_i^M, β_i^W follows a truncated bivariate normal distribution from the beginning and inverting the truncation step won't give the same result. For the global estimator, there is no justification on why King uses the weighted average of the estimates per region instead of the estimated B^M of the truncated bivariate normal distribution.

For these reasons we won't consider King's method as a solution to the EI problem.

3.4 Ecological inference through Distribution Regression

In this section we will explain the solution to the Inference problem using Distribution Regression.

Given n regions, the available information can be expressed as:

- Samples of demographic data per region $\{x_1\}, ..., \{x_n\}$
- Results per region $y_1, y_2, ..., y_n$.

The objective is to find a relation between the demographic data and the results per region. Each demographic subgroup can be seen as a new region for which we want to infer the quantities.

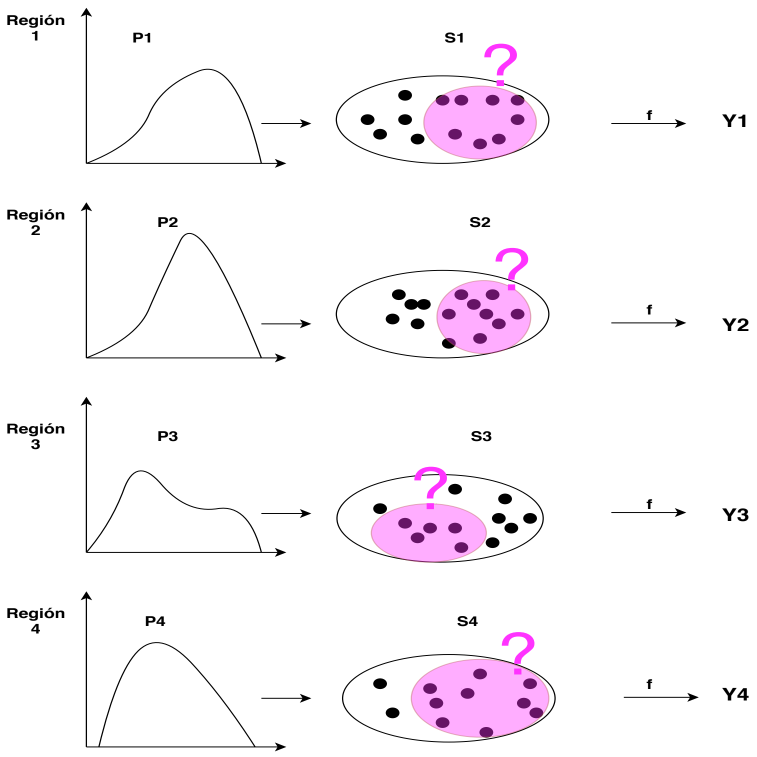

Figure 3.4: Ecological inference through Distribution Regression.

This method was presented in [5] using the framework based on kernel mean embeddings and Ridge regression. However, it is important to notice that any Distribution Regression method can be adapted to solve the Ecological Inference problem. The key idea is to fit a model using the individual information and results for all the regions, then use this model to generate predictions for the demographic subgroups of interest.

We will present the method using our base example in which we want to infer the percentage of women that voted for candidate A and the percentage of women that voted for candidate B.

First, we use the approximation method presented in section 2.1.3 to calculate explicit feature maps z for the individuals of each region. Then, calculate the mean embedding per region.

$$\psi(x) = \frac{\sum_j w_j z(x^j)}{\sum_j w_j}$$

where w_j are the weights reported in the census. These weights are reported for each individual to correct in case of problems in the sample such as lack of representativity or non-response.

Once we calculate $\psi(x)$ for x in $\{\{x_1\}, ..., \{x_n\}\}$, we use a kernel method to find a function f such that:

$$y = f(\psi(x))$$

In this work we used kernel Ridge regression to find f. We will see later that it is possible to define similarity measures between two distributions without explicitly defining the feature map z. In this case we will use the kernel formulation presented in section 2.1.2 to fit the model.

We use the parameters learned to make predictions for the demographic subgroup of interest. In this example, the demographic group of interest are women per region.

$$\hat{\mu_i}^W = \frac{\sum_{j=1}^{N_i^W} w_j z(x_i^j)}{\sum_{j=1}^{N_i^W} w_j}$$

We calculate $f(\hat{\mu_i}^W)$ using the parameters of the fitted model.

3.5 Learning with Label Proportions

One problem that is closely related to Ecological Inference is the problem of Learning with Label proportions (LLP). As described in [16], the LLP problem arises when training instances are provided as sets or bags and for each bag only the proportions of the labels are available. The objective is to infer the individual labels for each set.

To solve the LLP problem, there are different methods available in the literature, such as Optimizing Cluster Model Selection [15], Alternating Mean Map [17] and $\propto$SVM [16].

In [12] they propose a method based on a probabilistic graphical model and the Expectation Maximization (EM) algorithm. This model will be explained below.

We will use the following notation to describe the data:

$$\{(\{x_1\}, z_1), ..., (\{x_n\}, z_n)\}$$

where $z_i = \sum_{j=0}^{N_i} y_j$ is the number of positive labels in the bag i and y_j is the label of the jth observation.

For simplicity, we will explain the case in which each y_j is one dimensional binary variable. However,the model can be extended for multidimensional binary variables. The task is to infer the missing labels y_j for each bag.

The EM algorithm is an iterative algorithm whose objective is to maximize a likelihood function when there are missing values. In order to do so, two steps are performed alternately. The E step in which the objective is to estimate the missing values and then the M step in which the objective is to optimize the parameters to maximize the likelihood [8].

They formulate the problem in terms of finding θ that maximizes the likelihood:

$$p(\mathbf{y}, \mathbf{z}|\mathbf{x}; \theta) = \prod_{i=1}^{n} \prod_{j=1}^{N_i} p(y_j|x_j; \theta) p(z_i|\mathbf{y_i})$$

where $\mathbf{y_i} = \{y_1, ..., y_{N_i}\}$ will denote the set of labels in bag i and $\mathbf{x_i} = \{x_1, ..., x_{N_i}\}$ will denote the set of feature vectors in bag i. They denote $\mathbf{y}$, $\mathbf{z}$ and $\mathbf{x}$ to the concatenation of $\mathbf{y_i}$, $\mathbf{z_i}$, and $\mathbf{x_i}$ respectively. They set $p(z_i|\mathbf{y_i}) = \mathbb{1}[z_i = \sum_{j=0}^{N_i} y_j]$.

For $p(y_i = 1|x_i, \theta)$ they use a logistic regression model:

$$p(y_i = 1|x_i, \theta) = \sigma(x_i^T \theta)$$

where $\sigma(u) = \frac{1}{1+e^{-u}}$

In the tth iteration of the EM algorithm the following function will be maximized:

$$Q_t(\theta) = E_{\mathbf{y}|\mathbf{z},\mathbf{x}}[log(p(\mathbf{y}, \mathbf{z}|\mathbf{x}; \theta))]$$
$$= \sum_{i=0}^{n} E_{\mathbf{y_i}|\mathbf{z_i},\mathbf{x_i}} \big[log(p(z_i|\mathbf{y_i}) + \sum_{j=1}^{N_i} log(p(y_j|x_j; \theta))\big]$$

The term $\log(p(z_i|\mathbf{y_i})\big]$ is constant with respect to θ and can be ignored during the optimization:

$$Q_t(\theta) = \sum_{i=0}^{n} \sum_{j=1}^{N_i} E_{y_j|\mathbf{z_i},\mathbf{x_i}}[log(p(y_j|x_j; \theta))] + const$$

Dropping the constant term and substituting the logistic regression model, the function that will be maximized is:

$$Q_t(\theta) = \sum_{i=0}^{n} \sum_{j=1}^{N_i} q_j log(\sigma(x_j^T \theta)) + (1 - q_j) log(1 - \sigma(x_j^T \theta))$$

where $q_j = p(y_j = 1|z_i, \mathbf{x_i}; \theta)$

The maximization (M step) is straightforward and can be computed with standard solvers.

Calculating the expectation (E step) is challenging. It requires calculating the q_js for all the bags. This is done performing *exact inference* in a graphical model using *cardinality potentials*. Exact inference consists in enumerating all the possible cases and calculating the exact value for the marginal probabilities q_j, as opposed to using an approximation method.

For a single bag, they propose the following model:

$$q(y_1, ..., y_n) \propto \prod_j \psi_j(y_j)\phi(\sum_j y_j)$$

where $\psi_j(y_j) = \sigma(x_j^T\theta)$ are called *unary potentials* and $\phi(\sum_j y_j) = \mathbb{1}(z_{obs} = \sum_j y_j)$ is known as a *counting potential.* This counting potential is introduced because the $\{y_i\}$ are not independent, they are restricted to add up to a given value.

They propose to construct a factor graph (see Appendix A) by adding auxiliary variables z that represent the sum of nested subsets and arranging them in a tree shape such that the observed total becomes the root $z_{obs} = \sum_j y_j$. Between the variables there will be factors $\psi = 1[z_p = z_l + z_r]$ that represent relation between the variables. For each y_j there will be factor with the unary potential $\psi_j(y_j)$. There will be a factor attached to the root node with the counting potential $\phi(\sum_j y_j)$.

In the next figure we illustrate the tree construction for four variables.

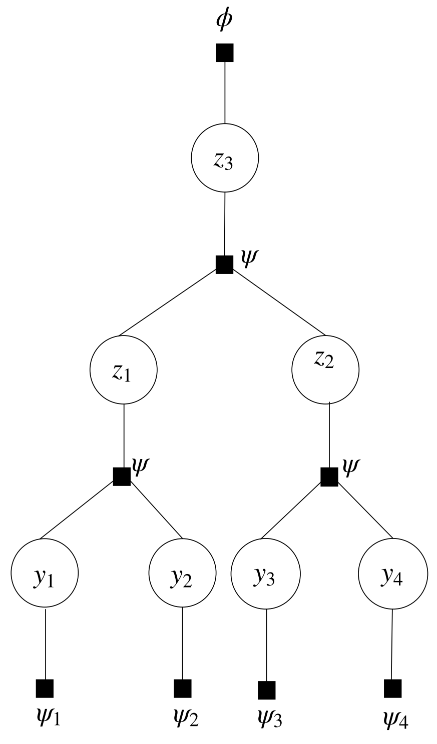

Figure 3.5: Tree construction for inference. Image adapted from [12].

In order to calculate the marginals they propose a recursive approach. First run a leaf to root computation using the following equations:

$$\begin{aligned}\alpha_p(z_p) &= \sum_{z_l \in \{0,\ldots,2^i\}} \sum_{z_r \in \{0,\ldots,2^i\}} a_l(z_l)\alpha_r(z_r)\mathbb{1}[z_p = z_l + z_r] \\ &= \sum_{z_l \in \{0,\ldots,2^i\}} a_l(z_l)\alpha_r(z_p - z_r)\end{aligned}$$

where $\alpha_0(y_i) = \psi_i(y_i)$
Then run a root to leaf computation using the following equations:

$$\begin{aligned}\beta_l(z_l) &= \sum_{z_p \in \{0,\ldots,2*2^i\}} \sum_{z_r \in \{0,\ldots,2^i\}} \beta_p(z_p)\alpha_r(z_r)\mathbb{1}[z_p = z_l + z_r] \\ &= \sum_{z_p \in \{0,\ldots,2*2^i\}} \beta_p(z_p)\alpha_r(z_p - z_r)\end{aligned}$$

$$\begin{aligned}\beta_r(z_r) &= \sum_{z_p \in \{0,\ldots,2*2^i\}} \sum_{z_l \in \{0,\ldots,2^i\}} \beta_p(z_p)\alpha_l(z_l)\mathbb{1}[z_p = z_l + z_r] \\ &= \sum_{z_p \in \{0,\ldots,2*2^i\}} \beta_p(z_p)\alpha_l(z_p - z_l)\end{aligned}$$

where $\beta_0(z) = \phi(z)$
In the next figure we illustrate the message passing scheme.

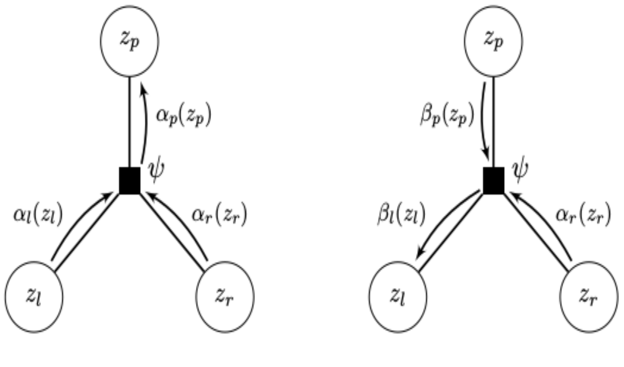

Figure 3.6: Message passing scheme. Image taken from [12].

3.5.1 Comparison with Distribution Regression

They use this method to perform Ecological Inferences for the 2016 Presidential Election. They mention that LLP is a more suitable approach to solve the EI problem than Distribution Regression. They argue that Distribution regression ignores known information about the problem such as the aggregation mechanism: each individual has a vote and these votes and the total proportion of votes per region is obtained through an addition. Also they claim that Distribution Regression unnecessarily aggregates the individual information by constructing a mean embedding, however this aggregation is done with the objective of finding a characteristic vector for each sample.

In [5] they mention LLP and discard it. First because the interest is not really estimating the vote for each individual but rather understand them as part of demographic subgroups and second because there are so many individual records that calculating each label is expensive and unnecessary. Instead of working with all the individuals, in [12] they take a representative sample of size 1000 for each region using the population weights.

In the next figure we illustrate the two methods.

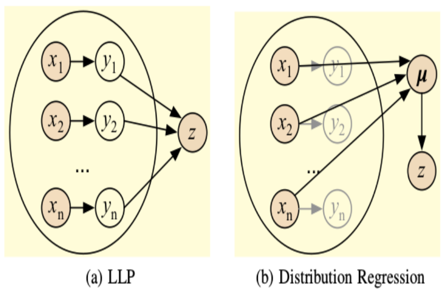

Figure 3.7: Comparison between LLP and Distribution Regression for EI. Image taken from [12].

In [12] they provide exit poll estimates for different demographic subgroups and evaluate the performance of LLP and Distribution Regression. Assuming that the exit poll estimates are accurate, none of the methods appears significantly better than the other. We will present these results in Chapter 5.

Chapter 4

Alternative similarity functions

In this chapter we propose alternatives to calculate similarity measures between samples of distributions for Distribution Regression. We present an additive approach to calculate kernels for multidimensional distributions. We show that for kernels based on mean embeddings, adding the kernels per dimension is equivalent to concatenating the explicit featurization. We explain how to build the Pyramid Match Kernel, a similarity measure based on multi resolution histograms. We define a kernel based on the Wasserstein distance, which we call the Wasserstein kernel. We analyze the behavior of these similarity measures using our base example on Gaussian distributions. Once we have K_{ab} we can solve the problem using kernel ridge regression as explained in section (2.2).
Finally, we explore another framework to solve the Distribution Regression problem using a neural network formulation.

4.1 Kernel based methods

4.1.1 Marginal kernel

Calculating similarity functions between multidimensional distributions is not straightforward. Some papers suggest defining similarity functions based on "signatures" of the form $\{(x_1, p_1), ..., (x_m, p_m)\}$ where x_i is the center of cluster i and p_i the number of points in the cluster. In the one dimensional case, these signatures correspond to the counts in histograms.

In this thesis we used a different method. For each dimension we calculated the similarity between the two samples, then added them up. If each kernel is positive definite then the sum will also be positive definite. We propose this method as an alternative to the method presented in [5], in which each observation is treated a long vector without selecting a specific kernel for the type of variable. A similar method is presented in [20].

Using this additive method, the similarity measure is defined as:

$$K_{ab} = \sum_{i=1}^{d} K_{ab}^{(i)}$$

where d is the number of dimensions of the distributions.

Defining the similarity measures using this method allowed us to use some libraries that are designed only for one dimensional distributions.

In the case of kernels based on mean embeddings it is possible to show that adding the kernels per dimension is equivalent to concatenating the explicit featurizations per dimension.

$$K_{ab} = \sum_{i=1}^{d} \frac{1}{N_a N_b} \sum_j \sum_l k(x_a^{j(i)}, x_b^{l(i)})$$

$$= \frac{1}{N_a N_b} \sum_j \sum_l \sum_{i=1}^{d} k(x_a^{j(i)}, x_b^{l(i)})$$

$$= \frac{1}{N_a N_b} \sum_j \sum_l \sum_{i=1}^{d} < \phi_i(x_a^{j(i)}), \phi(x_b^{l(i)}) >$$

$$= \frac{1}{N_a N_b} \sum_j \sum_l < [\phi_1(x_a^{j(1)}), \phi_2(x_a^{j(2)}), ..., \phi_d(x_a^{j(d)})], [\phi_1(x_b^{j(1)}), \phi_2(x_b^{j(2)}), ..., \phi_d(x_b^{j(d)})] >$$

$$=< \frac{1}{N_a} \sum_j [\phi_1(x_a^{j(1)}), \phi_2(x_a^{j(2)}), ..., \phi_d(x_a^{j(d)})], \frac{1}{N_b} \sum_l [\phi_1(x_b^{j(1)}), \phi_2(x_b^{j(2)}), ..., \phi_d(x_b^{j(d)})] >$$

This featurization is useful because it allows us to select the right kernel for each type of variable. We propose to use a Gaussian kernel for continuous variables and a linear kernel for categorical variables.

4.1.2 Pyramid match kernel

The Pyramid Match Kernel, introduced in [1] mainly for computer vision applications, presents a similarity function based on histograms at different resolutions.
In order to build this similarity function, it is assumed that the data has a maximum range D and that the minimum distance between two points is 1. This can be enforced by scaling the data and truncating the values to integers.
The information for each sample $\{x_k\}$ is condensed in a pyramid of histograms of the form:

$$\psi(\{x_k\}) = [H_0(\{x_k\}), ..., H_{L-1}(\{x_k\})]$$

where $L = \lceil log_2 D \rceil + 1$ and $H_i(\{x_k\})$ is a histogram with bins of side length 2^i.

The pyramid is constructed in order to guaranty that the bins at the finest level H_0 are small enough that every observation falls into a single bin, the bins increase in size until all the points fall into the same bin. The first time two points share a bin they are considered *matched*, the size of that bin is an indicator of the similarity between the points, as it reflects the maximum distance between the points.

In order to find the total number of matched pairs at resolution i, an intersection function is defined:

$$I(H_i(\{x_a\}), H_i(\{x_b\})) = \sum_{j=1}^{nBins} min(H_i(\{x_a\})^{(j)}, H_i(\{x_b\})^{(j)})$$

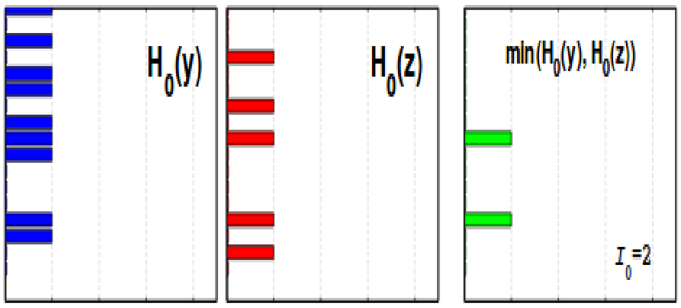

Figure 4.1: Intersection between histograms at the finest resolution. Image taken from [1].

Next, a measure of total matched points at level i is defined as:

$$N_i = \begin{cases} I(H_i(\{x_a\}), H_i(\{x_b\})) - I(H_{i-1}(\{x_a\}), H_{i-1}(\{x_b\})) & \text{if } i > 0 \\ I(H_0(\{x_a\}), H_0(\{x_b\})) & \text{if } i = 0 \end{cases}$$

Finally, the similarity between two samples $\{x_a\}$ and $\{x_b\}$ is defined as:

$$K_{ab} = \sum_{i=0}^{L-1} w_i N_i$$

where $w_i = \frac{1}{2^i}$ is a weight that reflects the similarity of points at each level. It intuitively means that two points matched at finer resolutions are more similar than two points matched at a higher resolution.

In the next plot we illustrate the Pyramid Match similarity between samples of size $10,000$ of Gaussian distributions X and Y as a function of μ_Y and σ_Y.

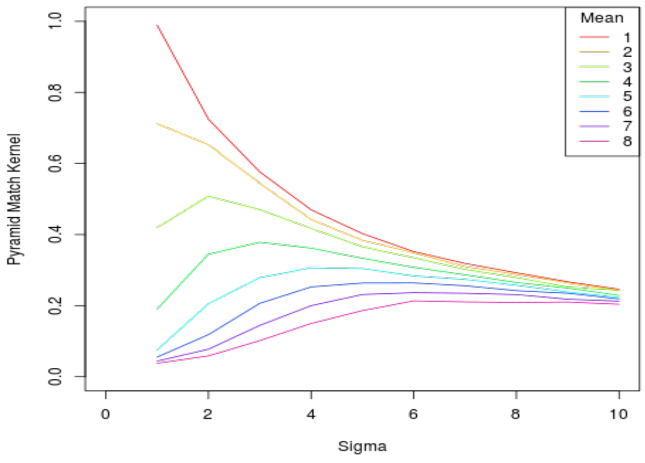

Figure 4.2: Pyramid match similarity on Gaussian distributions.

We observe that the Pyramid Match similarity has the same drawbacks as the similarity using the Gaussian kernel mean embeddings, while it correctly reflects the similarity in the mean it does not do the same for the variance.

4.1.3 Wasserstein kernel

Following the work presented in [21], the Earth Mover's Distance (EMD) is defined for "signatures" of the form $\{(x_1, p_1), ...(x_m, p_m)\}$ where x_i is the center of cluster i and p_i the number of points in the cluster. The signatures are not normalized and their total masses are allowed to be different.

As described in [22], the EMD is defined as the solution to a *transportation problem*. In the transportation problem several suppliers are given a certain amount of goods and they are required to transport a limited amount of units to several consumers with limited capacity. Associating this problem with signatures is done by defining one signature as the supplier and the other signature as the consumer. There is a given cost to transport each unit for each supplier to each consumer. In the case of signatures, this cost corresponds to a measure of dissimilarity between elements in the first signature and elements in the second signature. Intuitively, the solution is the minimum cost to transform one signature into another.

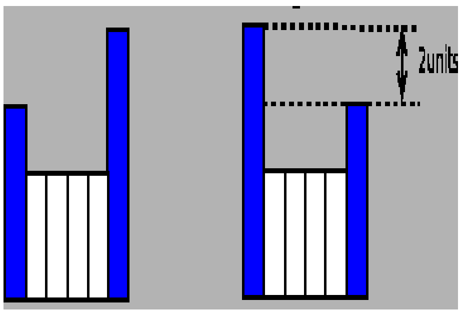

Figure 4.3: Earth Mover's Distance: the minimum cost to transform one signature into another. Image taken from [23].

Formally, given two signatures $P = \{(x_1, p_1), ...(x_m, p_m)\}$ and $Q = \{y_1, q_1), ...(y_n, q_n)\}$, and a measure of dissimilarity, $d_{ij} = d(x_i, y_j)$, between elements of P and Q. The objective is to find the optimal flow f^*_{ij} that minimizes the cost:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} d_{ij}$$

Subject to the following constraints:

$$f_{ij} \geq 0, \qquad 1 \leq i \leq m, 1 \leq j \leq n$$

$$\sum_{j=1}^{n} f_{ij} \leq p_i, \qquad 1 \leq i \leq m$$

$$\sum_{i=1}^{m} f_{ij} \leq q_j, \qquad 1 \leq j \leq n$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} = min(\sum_{i=1}^{m} p_i, \sum_{j=1}^{n} q_j)$$

The first constraint allows to interpret the f_{ij} as the number of elements transported from P to Q. The second constraint limits the amount of elements that can be moved from cluster i of P to the total number of elements p_i. The third constraint limits the amount of elements that can be received by cluster j in the transformed signature to the total elements in cluster j of Q. The last constraint forces to move the maximum number of elements possible. When the total masses of P and Q are different, the number of elements that should be moved is the total of elements in the signature with minimum mass. This is known as *partial matching.*

The EMD between P and Q is defined as:

$$EMD(P,Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}^{*} d_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}^{*}}$$

When working with probability distributions, the Earth Mover's Distance belongs to a family of distances known as p-Wasserstein distance or Mallows distance. It is a particular case when p = 1.

As described in [21], given two random variables X and Y with probability distributions P and Q respectively, the p-Wasserstein distance is defined as the minimum of the expected difference between X and Y, taken over all joint distributions F for (X, Y) such that the marginal distribution of X is P and the marginal of Y is Q:

$$W_p(P,Q) = \min\{(E_F||X-Y||^p)^{1/p} : (X,Y) \sim F, X \sim P, Y \sim Q\}$$

In the case of two discrete distributions $P = \{(x_1, p_1), ...(x_m, p_m)\}$ and $Q = \{y_1, q_1), ...(y_n, q_n)\}$, the problem is to find $F = (f_{ij})$ that minimizes:

$$E_F||X-Y||^p = \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}||x_i - y_j||^p = \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}d_{ij}$$

Subject to the following constraints:

$$f_{ij} \geq 0, \qquad 1 \leq i \leq m, 1 \leq j \leq n$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} = 1$$

$$\sum_{j=1}^{n} f_{ij} = p_i, \qquad 1 \leq i \leq m$$

$$\sum_{i=1}^{m} f_{ij} = q_j, \qquad 1 \leq j \leq n$$

The first two restrictions guarantee that F is a probability distribution. While the last two restrictions guarantee that the marginal distribution of X is P and the marginal of Y is Q.

The Wasserstein distance can calculated between signatures. In order to do this, signatures should be converted to probability distributions by normalizing the weights to add up to one.

As described in [22], for two signatures with the same number of elements, normalizing doesn't affect the result and the EMD is exactly the same as the 1-Wasserstein distance. However, when the total masses are different the EMD does something different from the Wasserstein distance.

We will illustrate the difference between the EMD and the Wasserstein distance with an example. Let $P = \{(1,1),(2,2),(3,1)\}$ and $Q = \{(1,1),(2,1),(3,1)\}$ be two signatures. P has a total mass of 4 while Q has a total mass of 3. Because Q is a subset of P, the EMD between them is 0.

If we convert these signatures to discrete probability distributions we obtain:
$X = \{(1,1/4),(2,1/2),(3,1/4)\}$ and $Y = \{(1,1/3),(2,1/3),(3,1/3)\}$. The optimal flow is given by:

$$F(x,y) = \begin{cases} 1/12 \text{ if x = 2, y = 1} \\ 1/12 \text{ if x = 2, y = 3} \\ 0 \qquad \text{otherwise} \end{cases}$$

The Wasserstein distance between them is 1/6.

In the following figure we illustrate our example:

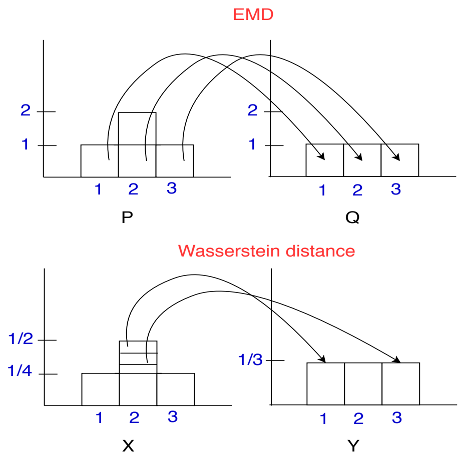

Figure 4.4: Comparison between EMD and 1-Wasserstein distance.

With this example we observe the EMD property of partial matching. This property can be useful for some computer vision applications such as image retrieval, however in the context of comparing distributions it could be a shortcoming. As in the example, two distributions could have EMD equal to 0 just because they share all the possible observations even if their shapes are completely different. For these reasons we will normalize the signatures and calculate the Wasserstein distance.

In the next plot we illustrate the 1-Wasserstein distance between samples of size $10,000$ of Gaussian distributions X and Y as a function of μ_Y and σ_Y.

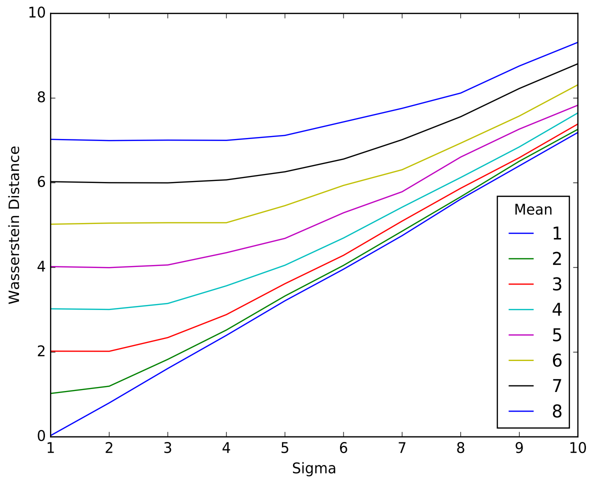

Figure 4.5: 1-Wasserstein distance between Gaussians

we observe that the 1-Wasserstein distance behaves as expected, it increases as the mean and variance of the distribution Y increase.

In the case of Gaussian Distributions there is an explicit expression of the 2-Wasserstein distance [24]. Let $X \sim N(\mu_X, \Sigma_X)$ and $Y \sim N(\mu_Y, \Sigma_Y)$. The 2-Wasserstein distance between these distributions is given by:

$$W_2(X,Y) = ||\mu_X - \mu_Y||^2 + trace(\Sigma_X + \Sigma_Y - 2(\Sigma_Y^{1/2}\Sigma_X\Sigma_Y^{1/2})^{1/2})$$

In the next plot we illustrate the 2-Wasserstein distance between Gaussian distributions X and Y as a function of μ_Y and σ_Y.

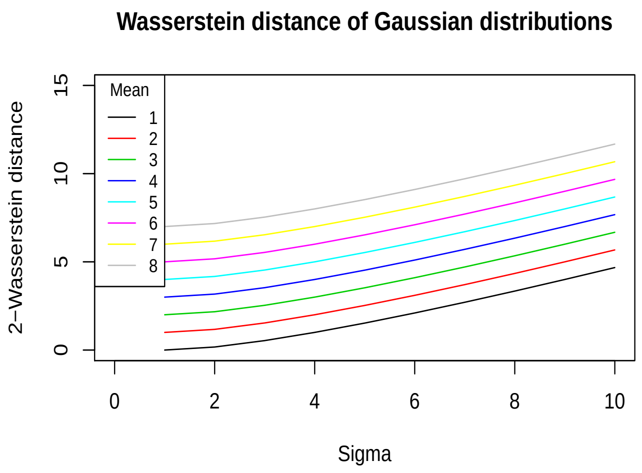

Figure 4.6: 2-Wasserstein distance between Gaussians

The 2-Wasserstein distance behaves as expected for Gaussian distributions, it increases as the mean and variance of the distribution Y increase.

We propose to define a similarity measure using the Wasserstein distance. In [25] they formalize the problem of defining a kernel K through a distance function $d(x, y)$. The call them Distance Substitution (DS) kernels and prove positive definiteness for some DS kernels. In particular, we will use the rbf kernel, defined as:

$$K(x,y) = e^{-\gamma d^2(x,y)}$$

As we explained in section (3.1), when working with multivariate distributions we will construct our kernel using an additive approach:

$$K_{ab} = \sum_{i=1}^{d} e^{-\gamma W_1(\{x_a^{(i)}\},\{x_b^{(i)}\})}$$

where $W_1(\{x_a^{(i)}\}, \{x_b^{(i)}\}$ is the Earth Mover's distance between two samples in dimension i.

To calculate the Wasserstein distance we used the function *wasserstein_distance* provided in the package *scipy.stats* of Python. The selection of the parameter γ plays a very important role in the performance of this kernel. In our experiments, the median heuristic didn't perform well. Instead, we used a grid search to find λ and γ. This grid search makes the training process slow given that we have to run the code as many times as parameters we have to check.

4.2 Deep Sets

In [26] they propose another framework to solve the Distribution Regression problem. Without making any assumptions about the data origin, they simply treat the $\{x_1\}, ..., \{x_n\}$ as sets. The objective then is transformed into finding a function $f : \{x_k\} \to y_k$ that is well defined on sets.

They propose a way to characterize a function to be valid on sets: the output must be indifferent to the ordering of the elements. Formally, this property is called permutation invariance and it means that for every π

$$f(\{x_k^1, ...x_k^{N_k}\}) = f(\{x_k^{\pi(1)}, ...x_k^{\pi(N_k)}\})$$

They provide a way to characterize the structure of a permutation invariant function through the next theorem.

Theorem [26]: A function $f(X)$ operating on a set X having elements from a countable universe, is a valid set function, i.e invariant to the permutation of instances of X, if and only if it can it can be decomposed in the form $f(X) = \rho(\sum_{x \in X} \phi(x))$ for suitable transformations ρ and ϕ.

They use neural networks to define ϕ and ρ leading to the following structure:
1. Each element of the set $\{x_k\}^j$ is transformed into some representation $\phi(\{x_k\}^j)$.
2. The representations $\phi(\{x_k\}^j)$ are added up and the output is processed using ρ.

This framework to solve the problem is called Deep Sets. We illustrate it in the next figure.

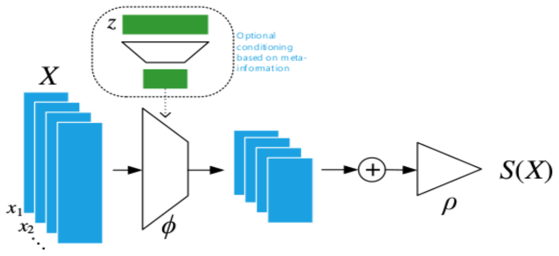

Figure 4.7: Deep Sets. Image taken from [26].

We observe that this framework is not completely different from the model presented in Chapter 2. In fact, the formulation using kernel mean embeddings lies under the structure of the theorem.

Drawing an analogy between these two methods, ρ can be seen as an extension to linear regression while ϕ represents an optimized featurization for the specific problem.

One of the main differences between the two frameworks is that when using kernel methods, a two step approach is used, while Deep Sets provides a one stage method. Having a one stage method is convenient because once the model is trained, it is really fast to generate predictions for a new set. Another advantage of the neural network formulation is that it allows for online learning, i.e learning when the data becomes available in sequential order [27], as opposed to the kernel methods in which it is necessary to have all the training set at a time to fit the model.

Another difference is that for the kernel methods there are usually one or two parameters to be estimated while for the neural networks the training is more complex. The Deep Sets model provides a family of functions that work to solve the general problem. However, for a specific dataset, constructing the network requires to make a series of experiments to select the parameters, such as number and size of layers, activation function, weight initialization, learning rate, number of iterations, etc. In the following section we explain the implementation details of the architecture that we used for our experiments.

4.2.1 Implementation details

We used three fully connected layers of size 250 for ϕ and three fully connected layers of size 75 for ρ. For the hidden layers we used the RELU activation function, defined as:

$$RELU(x) = \max(0, x)$$

We trained the model using the L2 loss function:

$$L2_{loss} = \sum_{i=1}^{n} (y_{true} - y_{predicted})^2$$

The neural networks were optimized using mini batch gradient descent with Adam Optimizer [28]. Each iteration of the mini batch gradient descent algorithm, also known as epoch, consists in splitting the full training set into random mini batches and optimizing the parameters on these subsets. We used mini batches of size 10 (sets). It has been empirically shown that the Adam optimizer makes the training faster. In the tth iteration, the Adam update is [29]:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v_t}} + \epsilon}\hat{\mu_t}$$

where $\hat{v_t}$ and $\hat{\mu_t}$ are unbiased estimates of the first and second momentum of the past gradients and η is the learning rate.

We used a learning rate of 0.0001 and we trained the model for 100 epochs. To set these parameters we performed multiple experiments on the validation set.

To initialize the weights we used the following heuristic, presented in [30].

$$W_{ij} \sim U\Big[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\Big]$$

where $U[-a, a]$ is the uniform distribution in the interval $(-a, a)$ and n is the size of the previous layer (the number of columns of W).

We observed that the network performed better using the mean instead of the addition, so we modified the function as:

$$f(X) = \rho\Big(\frac{\sum_{x \in X} \phi(x)}{N_X}\Big)$$

where N_X is the total of elements in X. This modification doesn't change the permutation invariance property of the function.

The network was implemented in TensorFlow, the code is available in Appendix B.

Chapter 5

Experiments

In this chapter we use two synthetic experiments to compare the performance of the methods presented before. We select the methods that show the best balance between accuracy and execution time. With these methods we infer voting behavior of demographic subgroups for the 2016 US Presidential Election.

5.1 Synthetic experiments

In this section we compare our methods using two different synthetic datasets. The first dataset is generated with a hierarchical model using the Gamma distribution. This model was presented in [31] and replicating it allows us to evaluate the methods in the general context of Distribution Regression, as well as comparing our methods with the Bayesian methods presented in [31]. The second dataset is constructed using a model based on a multivariate Gaussian distribution and a linear probability model. We created this model in order to simulate the Ecological Inference Problem. For both models we created training, testing and validation sets. The procedure to generate these models will be explained below.

5.1.1 Gamma model

This model is generated as follows:

First, generate $y_i \sim Uniform(4,8)$, the label for the *ith* bag. Next, draw $x_j^i \in \mathbb{R}^5$ using the following distribution:

$$x_j^{i\,(d)} \sim \frac{1}{y_i}\left[\Gamma(\frac{y_i}{2}, \frac{1}{2})\right] + \epsilon$$

where $j \in \{1, ..., N_i\}$, $d \in \{1, ..., 5\}$ and ϵ is an added noise term that will vary depending on the experiment.

We used the code provided in *https://github.com/hcllaw/bdr/blob/master/bdr/data/toy.py* to generate the data.

Following the experiments created in [31] we simulated two datasets. In the first experiment we evaluate the methods when there are bags of fixed size and added noise $\epsilon \sim N(0,1)$. In the second experiment we evaluate the performance when there are bags of variable size and no added noise. For both experiments we generate 1000 bags for training, 500 bags for validation and 1000 bags for testing.

Fixed bag size

For this experiment, all the bags contain a fixed number of elements and normal noise is added to each observation. We replicate the experiments in the paper and use $N_i = 1000$ for all i and $\epsilon \sim N(0, 1)$.

In the next table we compare the methods based on the test mean squared error and execution time. All the experiments were executed on a single threaded Intel® Core$^{\text{TM}}$ i5-7200U CPU @ 2.50GHz 4.

Method	Test MSE	Execution time (seconds)
Gaussian kernel	0.7497	40
Marginal Gaussian kernel	0.5268	147
Pyramid match kernel	0.3698	300
Wasserstein kernel	0.2448	20623
Deep Sets	0.2433	9154

The most accurate results are given by Deep Sets and the Wasserstein kernel. The mean squared error for both methods is very similar, however the execution time of the Wasserstein kernel is considerably longer than the one using Deep Sets.

Even though the marginal kernel improves the results of the Gaussian kernel, the mean squared error for both methods is really high. We also notice that for these methods the execution time is very low compared to the rest of methods. This is a consequence of two factors. First, the only parameter that needs to be selected is σ and the selection of this parameter is given by the median heuristic. Second, the dimension of the data is 5. As we explained in section 2.1.3, we used an explicit featurization with the property that the complexity of the computation depends on the dimension of the input data.

The Pyramid Match kernel error is not that far from the error obtained with the best methods and its execution time is substantially shorter. The method indicates to rescale the data in order to have one observation for each bin at the finest resolution. However, for our experiments we noticed that rescaling the data didn't improve the accuracy and applied the method without without rescaling.

Our results using the Wasserstein kernel and the Deep Sets methods are competitive with the results of the Bayesian methods presented in [31]. In contrast to the Bayesian methods, the parameter selection for the Wasserstein kernel is straightforward. We only need to select two parameters: γ and λ. This would be an advantage if we had an efficient method to automatically select the parameter γ, such as the median heuristic. This heuristic didn't give the expected results, instead we used a grid search to select the parameters and this increased the time to fit the model.

Because the Bayesian methods are developed in a neural networks formulation as well as the Deep Sets model they share some advantages and disadvantages. The first advantage is the possibility of optimizing the parameters using backpropagation. The second advantage is that both methods use a one stage approach and once the model is trained it is very easy and fast to generate predictions. A disadvantage of both methods is that selecting the parameters of the neural network, such as the learning rate, number of iterations and weight initialization require to perform multiple experiments.

We consider the simplicity of the Deep Sets formulation as an advantage over the Bayesian methods. While the Bayesian methods fit certain parameters to quantify different types of uncertainty, the Deep Sets method provides a general approach that can be adjusted to fit any kind of dataset.

As we explained in section 3.5, any Distribution Regression method can be applied to solve the Ecological Inference problem. For this reason, adapting the Bayesian methods to solve this problem would take as much work as adapting any of our methods.

Variable bag size

This experiment is designed to evaluate the behavior of our methods when there are variable bag sizes in the dataset and no added noise. We replicate the experiment in the paper and use the following procedure to generate the data. We fix four sizes $N_i \in \{5, 20, 100, 1000\}$. For each generated dataset, 25% of the bags have $N_i = 20$ and 25% of the bags have $N_i = 100$. For the other 50% of the data, we define s_5 as the overall percentage of bags with $N_i = 5$, the rest will be bags with $N_i = 1000$. We create datasets in which we vary s_5 from 0% to 50%. We use $\epsilon = 0$.

In the next plot, we illustrate our results.

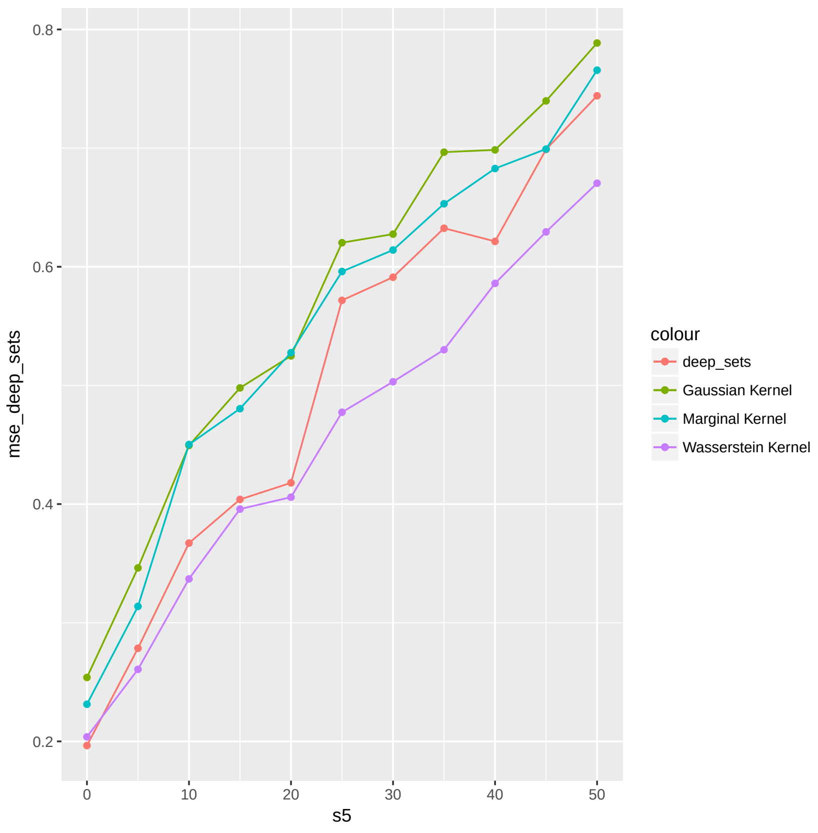

Figure 5.1: Test mean squared error on datasets with variable bag size.

For this experiments the most accurate methods are the Wasserstein kernel and Deep Sets. Both methods provide similar errors when the percentage of bags with five elements is less than 25%. However, when this percentage increases the Wasserstein kernel is the only method that produces competitive results.

The marginal kernel slightly improves the results of the Gaussian kernel. We observe that this improvement is consistent and doesn't get affected by the amount of bags with few elements.

On a different plot we observe the Pyramid Match Kernel, which underperforms compared with the other methods. We attribute this behavior to the presence of bags with a small number of observations. When there are few individuals per bag, the pyramid of histograms doesn't provide so much more information than a single histogram.

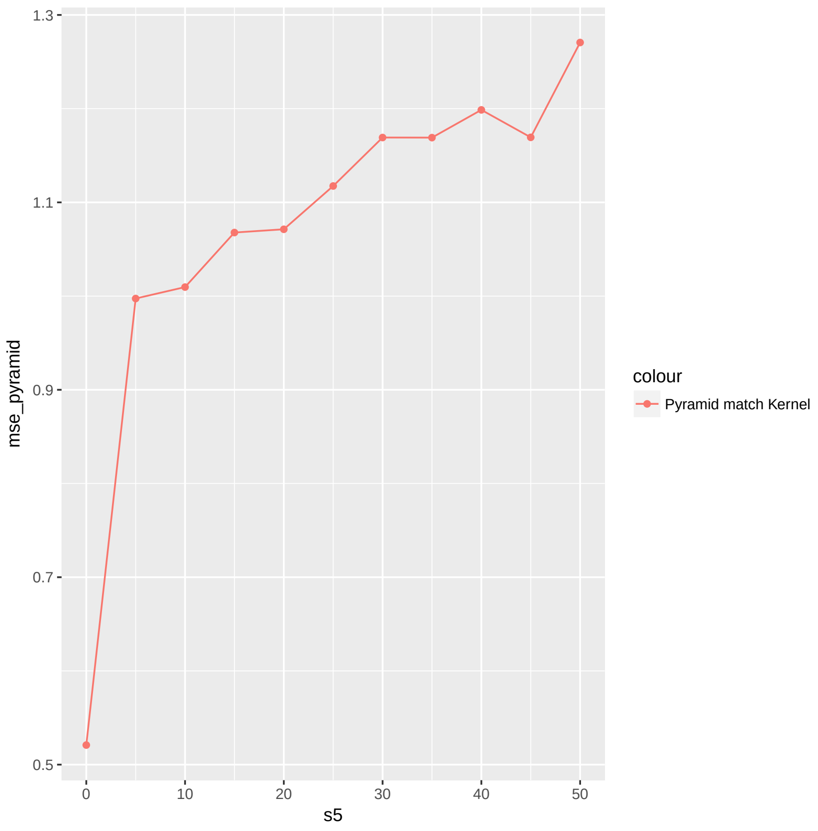

Figure 5.2: Test mean squared error on datasets with variable bag size.

Even if the Wasserstein kernel and Deep sets don't explicitly take into account the variability of the bag sizes, their accuracy is comparable with the Bayesian methods. As we mentioned before, we consider this as an advantage of the Deep Sets and the Wasserstein kernel methods over the Bayesian methods.

5.1.2 Gaussian model

In order to compare the performance of the methods in the context of Ecological Inference, we created the following model:

First, we generate the population for the *ith* bag. This population will be formed by two subgroups that we will call Category A and Category B. In a real life example these categories correspond to demographic subgroups divided by certain characteristics, such as sex, region type, income level, etc.

To generate the population for category A we generate a sample of the following distribution:

$$[x_i^j]_A \sim N(0, \Sigma_i^A), j \in [N_i^A]$$

where $\Sigma_i^A = \sigma_i^A * Identity(10)$.
We generate the population for category B using an analogous procedure.

To distinguish the categories in our population we added a binary variable that will represent elements from category A and category B with zeros and ones respectively.

In the following table we show an example of the population for a region.

$x_i^{(1)}, x_i^{(2)} \ldots, x_i^{(10)}$	$x_i^{(11)}$ = Category
$N(0, \Sigma_i^A)$	0 $\vdots$ 0
$N(0, \Sigma_i^B)$	1 $\vdots$ 1

To generate the labels we calculate a vector of probabilities π_i using the following expression:

$$\pi_i^j = \frac{e^{([x_i^j]^T \beta)}}{1 + e^{([x_i^j]^T \beta)}}$$

where β is a random vector of dimension 10 drawn from a $U(0, 1)$. This vector will be generated once and we will use it to generate the labels for all the bags.

Next, we draw y_i^j from $Bernoulli(\pi_i^j)$. In a real life example, y_i^j would be the vote of person j

The label for region i will be the percentage of elements with vote 1.

$$\bar{y_i} = \frac{\sum_{j=1}^{N^i} y_i^j}{N_i}$$

We generate different experiments varying the number of elements per bag N_i and relation between populations over bags and categories using σ_i^A and σ_i^B.

For all of our experiments we generated a training set and a validation set with 1000 and 500 bags respectively. The test set was obtained from the training set, subsetting all the elements corresponding to category A per bag. We evaluated our experiments using the mean squared error on the test set.

We created four experiments to evaluate our methods in different types of populations.

Experiment 1. All regions from the same distribution and size

In this experiment all the bags contain the same number of elements $N_i = 1000$, with half of the elements of each category $N_i^A = N_i^B = 500$. The covariance matrix for both categories is the identity, i. e. $\Sigma_i^A = \Sigma_i^B = 1$.

Experiment 2. All regions from the same distribution and random size

In this experiment we would like to evaluate the performance of the methods when the size of the bags is generated randomly. We draw N_i^A and N_i^B random from $U(250, 500)$. As in experiment 1, the covariance matrix for both categories is the identity, i. e. $\Sigma_i^A = \Sigma_i^B = 1$.

Experiment 3. Distribution varies between categories and random size

In these experiment we evaluate the methods when the distribution between categories is different. To do this, we use $\Sigma_i^A = 1$ and $\Sigma_i^B = 10$. We draw N_i^A and N_i^B random from $U(250, 500)$.

Experiment 4. Distribution varies between bags

Finally, in this experiment we want to evaluate the models when the distribution in every region is different and the distribution between categories is the same. We vary the covariance matrix per region. We use $\Sigma_i^A = \Sigma_i^B = i$. We draw N_i^A and N_i^B random from $U(250, 500)$.

In the next table we show the test mean squared error for our experiments.

Method	1	2	3	4
Gaussian Kernel	0.0037	0.0012	0.0053	0.0006
Gaussian Kernel Marginal	0.0038	0.0007	0.0006	0.0004
Pyramid match kernel	0.0468	0.0209	0.0268	0.0209
Wasserstein Kernel	0.0015	0.0004	0.0261	0.0078
Deep Sets	0.0023	0.0004	0.0005	0.0002

Our experiments were designed for two purposes. First, they allow us to perform Ecological Inference and to compare the results of our methods in a controlled environment, i.e we know the labels for each individual, in consequence we also know the proportions for every subset. Second, they allow us to understand the influence of the type of population on the performance of the methods.

Experiments 1 and 2 are similar except for the size of the bags. As opposed to what we would expect, the error for all of the methods is lower for experiment 2 than for experiment 1. If we compare this result with the Gamma experiment, we can conclude that having bags of variable bag size doesn't affect the quality of the predictions as long as these bags have a large amount of elements.

We observe that the highest errors overall the experiments are obtained in experiment 3, in which the population for each region is divided in two categories.

In comparison with the Gamma datasets, for these experiments most of the methods are very accurate. We observe that in all of our experiments, the method that consistently performs well is Deep Sets. It also stands out that the Pyramid Match kernel underperforms in all the cases.

The Gaussian kernel and the marginal kernel are accurate. For most of the experiments the marginal kernel improves the results of the Gaussian kernel.

The Wasserstein Kernel is the second best method, except for experiment 3 in which its error is comparable with the Pyramid Match kernel error.

For our real data application we will discard the Wasserstein kernel for its computational cost and the Pyramid Match kernel for its high prediction error. Therefore, we will calculate our results using Deep Sets and the Marginal kernel.

5.2 Ecological Inference for the US 2016 Elections

In this section we use the Deep Sets model to infer voting behavior for different demographic subgroups. We also present our results using the marginal kernel and Ridge regression. To build the demographic subgroups we divide the regions by sex, race and educational attainment.

5.2.1 Data

Following the work in [5], the individual level census data was obtained from the American Community Survey's Public Use Microdata Sample files (ACS PUMS). ACS PUMS are files containing records for a sample of housing units, with information about the characteristics of the housing and the people that live in it. We are using a 5 % sample of housing units in the United States. This sample is organized on Public Use Microdata Areas (PUMAS), which are based on counties and may be single counties or group of counties.

The elections results by county were taken from *https://github.com/flaxter/us2016.* In [?] they report that this results were scraped from nbc.com on 9 November 2016.

Some of the PUMAS don't coincide with the regions in which the elections results are based. We used the Pummeler package to match the regions and merge the PUMS data from 2012 to 2015. The Pummerler package was presented in [5] and can be downloaded from *https://github.com/dougalsutherland/pummeler.*

We obtained a total of 979 regions after preprocessing. Each region has 19 real variables and 101 discrete variables.[1] Some examples of the real variables are age, travel time to work in minutes and salary income in the past 12 months. While some examples of the discrete variables are citizen status, sex, recorded detailed race and educational attainment.

We normalize the real variables to have mean 0 and standard deviation 1. We transform the discrete variables into dummy variables. As a result we obtain a total of 3877 variables.

The average number of individual records per region is 942, the maximum is 256,352 and the minimum is 2214. Each individual record includes a survey weight to correct for oversampling and non response. These weights are taken in consideration as explained in section 3.4.

[1]All the information about the variables is available in *https://www2.census.gov/programs surveys/acs/tech _ docs/pums/data_ dict/PUMSDataDict15.txt*

5.2.2 Results

We estimate the proportion of votes for Clinton using different demographic subgroups formed by splitting by sex, race and educational attainment.

We consider only two categories for voting: Clinton and Trump. The proportion of voting for Clinton is calculated as:

$$proportion\_Clinton = \frac{votes\_Clinton}{votes\_Trump + votes\_Clinton}$$

To train the Deep Sets model we use 60 % of our regions for training, 20 % for validation and 20 % for testing. We trained our model using the architecture presented in 4.2.1. Due to memory limitations we restricted the amount of individuals per region to 10,000, taking a representative sample when the number of individuals exceeded this limit. For the same reason we used batch size one, i.e we trained the model on a single region per iteration. We used 10 epochs.

To train our marginal kernel we use 75 % of our regions for training and 25 % for testing. The parameter λ is automatically selected using cross validation on the training set. To calculate the mean embeddings we used all the individuals per region.

In the next plots we show the models predictions versus real proportions for the test regions.

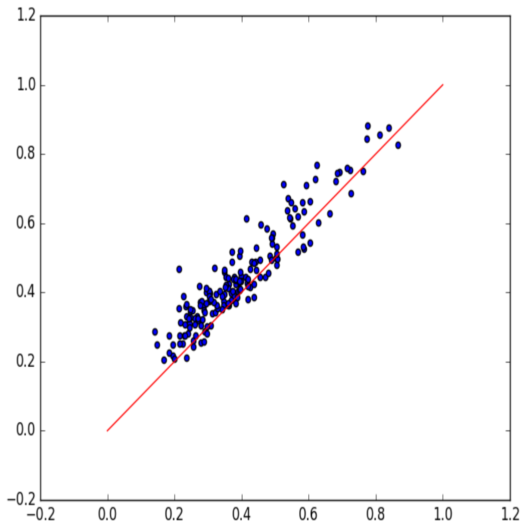

Figure 5.3: Fit using deep sets

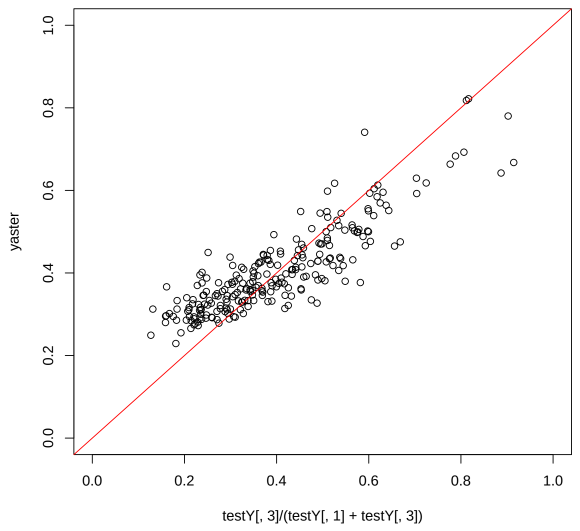

Figure 5.4: Fit using marginal kernel + Kernel Ridge regression

To compare our results we use the explained variance score:

$$\text{explained\_variance}(y, \hat{y}) = 1 - \frac{Var(y - \hat{y})}{Var(y)}$$

Using the Deep Sets method we obtained an explained variance score 0.8772 and a test mean squared error of 0.005. While using the Marginal kernel and Ridge regression we obtained an explained variance score 0.7251 and a test mean squared error of 0.006.

In the next table we show our predicted results for Clinton proportion for different demographic subgroups. As a point of comparison we also present the results for the LLP model using cardinality potentials and the exit poll results. These results were taken from [12]. For the exit poll results they used collected by Edison Research and scraped from the CNN website.

Variable	Subgroup	Marginal kernel	Deep Sets	LLP	Exit Poll
SEX	Men	0.34	0.43	0.41	0.44
	Women	0.47	0.48	0.45	0.57
RACP1P	White	0.37	0.37	0.31	0.39
	Black	0.57	1.02	0.99	0.92
	Asian	0.82	0.79	1	0.71
SCHL	High school or less	0.34	0.38	0.18	0.47
	Some college	0.49	0.44	0.26	0.46
	College graduate	0.52	0.57	0.71	0.53
	Postgraduate	0.55	0.64	0.68	0.61

We notice that for most of the categories, the results using Deep Sets are closer to the exit poll than the results obtained with the marginal kernel and the LLP model.

Something that stands out is the 1.02 inference of Deep Sets for the category black on race. Because we are using regression, there is no guarantee that the results will be in the interval $[0, 1]$. In our experiments we observed that adding a sigmoid function to restrict the interval really worsened the fit, for this reason we decided not to use it. Even with this problem, the result is closer to the exit poll result than with the marginal kernel.

A special case is the category of women, in which the inferred percentages for all the methods are very far from the exit poll results. The reason why the methods fail could be that the population for each region was very different for men than for women. As we saw in Experiment 3 of the Gaussian dataset, it is harder to predict when the population for each region is divided.

We conclude that when the goal is to predict for demographic subgroups as opposed to every individual label, Distribution Regression is a suitable approach to solve the Ecological inference problem.

Chapter 6

Conclusions

This thesis was motivated by [5], in which they present a method for Distribution Regression based on the Gaussian kernel and Ridge regression. In this paper they also present a method to apply Distribution Regression to solve the Ecological Inference problem.

The objectives of this thesis were to understand the similarity measure defined with the Gaussian kernel, to propose different similarity measures to improve the quality of the predictions and to perform some experiments to compare these methods.

When searching for information about the Ecological Inference problem, we realized of the lack of clear sources about the topic. We made a review of the problem and the solutions that have been proposed until now. We expect this review to be a contribution for filling the gap in the Ecological Inference literature.

We developed an explicit expression for the Gaussian based similarity on Gaussian distributions and we observed that the difference in variance wasn't reflected in the similarity. With the objective of finding a similarity function that reflected the expected behavior we proposed three different ways of calculating similarity: the pyramid match kernel, the Wasserstein kernel and the marginal kernel.

The marginal kernel was proposed as a method to work with multi dimensional distributions. As opposed to the approach presented in [5] we proposed to select an individual kernel for each dimension to take into account the type of variable and the scale. In most of our synthetic experiments, we observed a slight improvement over the Gaussian kernel.

We implemented the pyramid match kernel as presented in [1]. In our experiments, the results using this kernel were inferior to the Gaussian kernel in most of the cases. We used this similarity for our example on Gaussian distributions and we noticed that it presented the same problem as the Gaussian kernel.

We proposed to use the Wasserstein distance and defined a kernel using this distance. The use of this similarity function considerably reduced the prediction error. However, the parameter selection was done using a grid search and this really slowed down the training. As a possible future work we propose to find an automated way to find the parameter γ in order to reduce the computation time.

We compared our kernel methods with a neural network formulation called Deep Sets. In our experiments we observed a significant improvement in the prediction results when using this method.

Finally, we used the marginal kernel and Deep Sets to perform EI on the US 2016 results. We compared our results with the exit poll results and observed very accurate results for certain demographic subgroups. Nevertheless, there were a few demographic subgroups for which the predictions were far from the exit poll results. Further research about the influence of the population on the prediction results could be another option for future work.

Bibliography

[1] Kristen Grauman, Trevor Darrell. *The Pyramid Match Kernel: Efficient Learning with sets of features.* Journal of Machine Learning Research 8 (2007) 725-760.

[2] Michelle Ntampaka, Hy Trac, Dougal J. Sutherland, Nicholas Battaglia, Barnabás Póczos, Jeff Schneide. *A Machine Learning Approach for Dynamical Mass Measurements of Galaxy Clusters.* The Astrophysical Journal 803(2) · October 2014.

[3] Yuya Yoshikawa, Tomoharu Iwata, Hiroshi Sawada. *Latent support measure machines for bag-of-words data classification.* NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2.

[4] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, Leonidas J. Guibas. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.* Conference on Computer Vision and Pattern Recognition (CVPR) 2017.

[5] Seth Flaxman, Yu-Xiang Wang, and Alexander J Smola. *Who Supported Obama in 2012? Ecological Inference through Distribution Regression.* KDD '15 Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[6] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Bernhard Schölkopf. *Kernel Mean Embedding of Distributions: A Review and Beyond.* Foundations and Trends in Machine Learning: Vol. 10: No. 1-2, pp 1-141 (2017).

[7] Zoltan Szabo, Arthur Gretton, Barnabas Poczos, Bharath Sriperumbudur. *Two-stage sampled learning theory on distributions.* International Conference on Artificial Intelligence and Statistics (AISTATS) 2015.

[8] Kevin Murphy. *Machine Learning. A probabilistic perspective.* The MIT Press 2012.

[9] Ali Rahimi, Benjamin Recht. *Random Features for Large-Scale Kernel Machines.* NIPS'07 Proceedings of the 20th International Conference on Neural Information Processing Systems.

[10] W. Rudin. *Fourier Analysis on Groups.* Wiley Classics Library. Wiley-Interscience, New York, reprint edition, 1994.

[11] Gary King. *A solution to the ecological inference problem.* Princenton Univertiyy Press, 1997.

[12] Tao Sun, Dan Sheldon, Brendan O'Connor. *A Probabilistic Approach for Learning with Label Proportions Applied to the US Presidential Election.* 2017 IEEE International Conference on Data Mining (ICDM).

[13] Gary King, Molly Roberts. *ei: Ecological Inference.* R package version 1.3-3.

[14] D. A. Freedman, S. P. Klein, U. C. Berkeley, CA 94720, M. Roberts. *On "Solutions" to the Ecological Inference Problem.* Technical Report No. 515, Statistics Department, 1998.

[15] Marco Stolpe, Katharina Morik. *Learning from Label Proportions by Optimizing Cluster Model Selection.* ECML PKDD'11 Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part III.

[16] Felix X. Yu, Dong Liu, Sanjiv Kumar, Tony Jebara, Shih-Fu Chang. *$\propto$SVM for Learning with Label Proportions.* ICML 2013.

[17] Giorgio Patrini, Richard Nock, Paul Rivera, Tiberio Caetano. *(Almost) No Label No Cry.* Advances in Neural Information Processing Systems 27 (NIPS 2014).

[18] Daniel Tarlow, Kevin Swersky, Richard S. Zemel, Ryan Prescott Adams. *Fast Exact Inference for Recursive Cardinality Models.* UAI, 2012.

[19] http://mlg.eng.cam.ac.uk/teaching/4f13/1819/factor%20graphs.pdf

[20] Seth Flaxman, Dougal Sutherland, Yu-Xiang Wang, Yee Whye Teh. *Understanding the 2016 US Presidential Election using ecological inference and distribution regression with census microdata.* 2016.arXiv:1611.03787.

[21] Elizaveta Levina, Peter Bickel. *The Earth Mover's Distance is the Mallows Distance: Some Insights from Statistics* Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, Volume: 2

[22] Yossi Rubner,Carlo Tomasi, Leonidas J. Guibas. *The Earth Mover's distance as a metric for image retrieval.* International Journal of Computer Vision. Volume 40 Issue 2, Nov. 2000.

[23] http://vellum.cz/ mikc/oss-projects/CarRecognition/doc/dp/node29.html

[24] https://en.wikipedia.org/wiki/Wasserstein_metric

[25] Bernard HaasdonkClaus Bahlmann *Learning with distance substitution kernels.* Rasmussen C.E., Bülthoff H.H., Schölkopf B., Giese M.A. (eds) Pattern Recognition. DAGM 2004. Lecture Notes in Computer Science, vol 3175. Springer, Berlin, Heidelberg

[26] Manzil Zaheer, Satwik Kottur, Siamak Ravanbhakhsh, Barnabás Póczos, Ruslan Salakhutdinov, Alexander J Smola. *Deep Sets.* 31st Conference on Neural Information Processing Systems (NIPS 2017).

[27] https://en.wikipedia.org/wiki/Online_machine_learning

[28] Diederik P. Kingma, Jimmy Ba. *Adam: A Method for Stochastic Optimization.* International Conference on Learning Representations, pages 1–13, 2015.

[29] Sebastian Ruder. *An overview of gradient descent optimization algorithms.* arXiv:1609.04747v2

[30] Xavier Glorot, Yoshua Bengio. *Understanding the difficulty of training deep feed-forward neural networks.* Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR 9:249-256, 2010.

[31] Ho Chung Leon Law, Dougal J. Sutherland, Dino Sejdinovic, Seth Flaxman. *Bayesian Approaches to Distribution Regression* AISTATS 2018.

Appendix A

Marginal inference in factor graphs with cardinality potentials

Given a function of the form

$$q(y_1, ..., y_n) \propto \prod_j \psi_j(y_j)\phi(\sum_j y_j)$$

where $y_1, ...y_n$ are binary random variables.

In [18] they show that it is possible to construct a tree structure factor graph and then perform exact marginal inference using a *message passing* algorithm. Factor graphs are a type of probabilistic graphical model that allow us to represent the product structure of a function [19]. The message passing algorithm will be useful to efficiently calculate the marginals of $q(y_1, ...y_n)$, which are the quantities of interest.

Appendix B

Deep Sets implementation

```
import numpy as np
import tensorflow as tf
#skl_groups was taken from https://pypi.org/project/skl-groups/
from skl_groups.features import Features

def init_weights(shape):
    """ Weight initialization """
    weights = tf.random_uniform(shape, minval= -1.0/np.sqrt(shape[0]), \
    maxval = 1.0/np.sqrt(shape[0]))
    return tf.Variable(weights)

def init_all_weights(dim):
    h_size = 250
    rho_size = 75
    W1 = init_weights((dim, h_size))
    b1 = init_weights((1, h_size))
    W2 = init_weights((h_size, h_size))
    b2 = init_weights((1, h_size))
    W3 = init_weights((h_size, h_size))
    b3 = init_weights((1, h_size))
    W4 = init_weights((h_size, rho_size))
    b4 = init_weights((1, rho_size))
    W5 = init_weights((rho_size, rho_size))
    b5 = init_weights((1, rho_size))
    W6 = init_weights((rho_size, 1))
    b6 = init_weights((1, 1))
    weights = {"W1": W1,
               "b1": b1,
               "W2": W2,
               "b2": b2,
               "W3": W3,
               "b3": b3,
               "W4": W4,
               "b4": b4,
               "W5": W5,
                "b5": b5,
                "W6": W6,
                "b6": b6}
    return weights

def forwardprop(X, weights, segment_ids):
    W1 = weights["W1"]
    b1 = weights["b1"]
    W2 = weights["W2"]
    b2 = weights["b2"]
    W3 = weights["W3"]
    b3 = weights["b3"]
    W4 = weights["W4"]
    b4 = weights["b4"]
    W5 = weights["W5"]
    b5 = weights["b5"]
    W6 = weights["W6"]
    b6 = weights["b6"]

    Z1 = tf.add(tf.matmul(X, W1), b1)
    A1 = tf.nn.relu(Z1)
    Z2 = tf.add(tf.matmul(A1, W2), b2)
    A2 = tf.nn.relu(Z2)
    Z3 = tf.add(tf.matmul(A2, W3), b3)
    mean_emb = tf.segment_mean(Z3, segment_ids)
    Z4 = tf.add(tf.matmul(mean_emb, W4), b4)
    A4 = tf.nn.relu(Z4)
    Z5 = tf.add(tf.matmul(A4, W5), b5)
    A5 = tf.nn.relu(Z5)
    Z6 = tf.add(tf.matmul(A5, W6), b6)
    return tf.reshape(Z6, [-1])

def create_placeholders(dim):
    X = tf.placeholder(tf.float32, shape=[None, dim])
    y = tf.placeholder(tf.float32, shape=[None])
    segment_ids = tf.placeholder(tf.int32, shape = [None])
    return X, y, segment_ids

def shuffle(feats):
    indexShuffle = np.random.permutation(len(feats))
    feats_shuffled = [feats[i] for i in indexShuffle]
    labels_shuffled = [feats.labels[i] for i in indexShuffle]
    return Features(feats_shuffled, labels = labels_shuffled)

def model(train, validation, test, learning_rate = 0.0001,
    num_epochs = 100, minibatch_size = 10, seed = 0):
    tf.set_random_seed(seed)
    #Read dimensions
```

```
dim =  train[0].shape[1]
num_bags_train = len(train)
num_bags_validation = len(validation)
num_bags_test = len(test)
#Create placeholders
X, y, segment_ids = create_placeholders(dim)
#Weight initializations
weights = init_all_weights(dim)
# Forward propagation
yhat = forwardprop(X, weights, segment_ids)
loss = tf.reduce_sum(tf.square(tf.subtract(y, yhat)))
mse =  tf.reduce_mean(tf.square(tf.subtract(y, yhat)))
updates = tf.train.AdamOptimizer(learning_rate).minimize(loss)
variables = tf.global_variables()

#Tensorboard
saver = tf.train.Saver(variables, max_to_keep = 1)

# Run SGD
sess =tf.Session()

#Restore model
try:
    saver.restore(sess, "./model.ckpt")
    print("Model restored")
except:
    init = tf.global_variables_initializer()
    sess.run(init)
    print("init variables")

#Training
loss_vec = []
validation_vec = []
for epoch in range(num_epochs):
    train = shuffle(train)
    num_batches = int(num_bags_train/minibatch_size)
    for i in range(num_batches):
        ini = minibatch_size*i
        end = minibatch_size*(i+1)
        if i==(num_batches-1):
            end = num_bags_train
        batch_X = train[ini:end]
        batch_y = train.labels[ini:end]
        segment_id_train = np.hstack([x*np.ones(batch_X[x].shape[0]) \
        for x in range(len(batch_X))])
        batch_X_stacked = np.vstack(batch_X)
        sess.run(updates, feed_dict={X: batch_X_stacked, y: batch_y, \
        segment_ids: segment_id_train})
```

```
            if i==0:
                loss_value = sess.run(loss, feed_dict = {X: batch_X_stacked, \
                y: batch_y, segment_ids: segment_id_train})
                print("Epoch %d ,loss: %0.4f" % (epoch + 1, loss_value))
                loss_vec.append(loss_value)
        #Validation
        segment_id_val = np.hstack([x*np.ones(validation[x].shape[0]) \
        for x in range(num_bags_validation)])
        mean_squared_error = sess.run(mse, feed_dict = {X:np.vstack(validation),\
        y: validation.labels, segment_ids:segment_id_val})
        validation_vec.append(mean_squared_error)
        print("val MSE = %0.4f" % mean_squared_error)

    #Test
    segment_id_test = np.hstack([x*np.ones(test[x].shape[0]) \
    for x in range(num_bags_test)])
    mean_squared_error = sess.run(mse, feed_dict = {X:np.vstack(test), \
    y: test.labels, segment_ids:segment_id_test})
    print("Test MSE = %0.4f" % mean_squared_error)
    weights = sess.run(weights)
    return(weights, mean_squared_error)
```