Report Project MAL 2 2022

Functional Analysis and Transfer Learning

Supervisors: J. Park and M. Mougeot

Authors: Meccheri Nicola, Antona Antonio

Contents

1 Exercise 1: Functional data analysis **4**

1.1 Exploratory data analysis 4

1.2 Smoothing the dataset 5

1.3 Functional PCA 6

1.4 Clustering analysis 6

2 Exercise 2: CT-Scan Dataset **8**

2.1 Question 1 - Transfer learning problem 8

2.2 Question 2 - Ridge regression model 8

2.3 Question 3 - Covariate shift 9

2.4 Question 4 - KDE 10

2.5 Question 5 - Nearest neighbor weighting (NNW) 10

2.6 Question 6 - Tuning the algorithm 11

2.7 Question 7 - KLIEP 11

2.8 Question 8 - Cross-validation 11

2.9 Question 9 - Testing models 12

2.10 Question 10 - Testing best possible models 13

List of Figures

1 Functions Plot 4

2 Mean(in black) and Variance (in blue) 4

3 First derivative 4

4 Second derivative 4

5 Outliergram 5

6 Nadaraya Watson smoothing with parameter = 30 5

7 KNeighbors smoothing with parameter = 39 5

8 Local Linear Regression smoothing with parameter = 20 6

9 Basis smoothing with parameter = 5 6

10 first and second PCA component 6

11 Cumulative explained variance ratio 6

12 first against second PCA component. The colors represent the different clusters 7

13 First cluster's functions 7

14 Second cluster's functions 7

15 Third cluster's functions 7

16 PCA source target 8

17 Output distribution source target 8

18 target variable vs predictions 9

19 target variable vs first PCA component 9

20 # of samples vs time of execution for KDE 10

21 # of samples vs time of execution for NNW 10

22 weighted PCA components 10

23 estimated weighted distribution on the source (blue) and target (orange) 11

24 # of neighbors vs KL divergence 11

25 New weighted distribution using KMM 12

List of Tables

1 Models performance on source dataset 12
2 Instance based methods performance 12
3 Hyperparameters space 13
4 Performance of the best combination of HP 13

1 Exercise 1: Functional data analysis

1.1 Exploratory data analysis

The dataset is composed by 18 functions each of them evaluated on 542 points, not equispaced. The functions are shown in Fig. 1. Fig. 2 shows the mean and the variance across all functions. We can see that the mean is always very close to zero. The variance is not constant as we can see by the graph, the functions have periods of higher and lower variance.

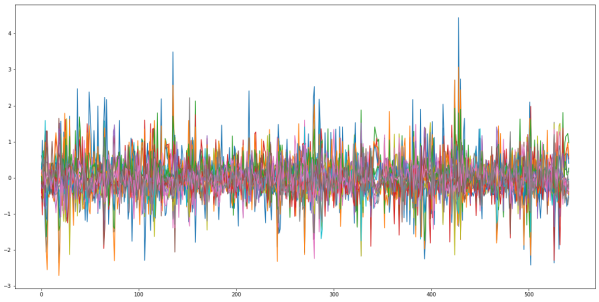

Figure 1: Functions Plot

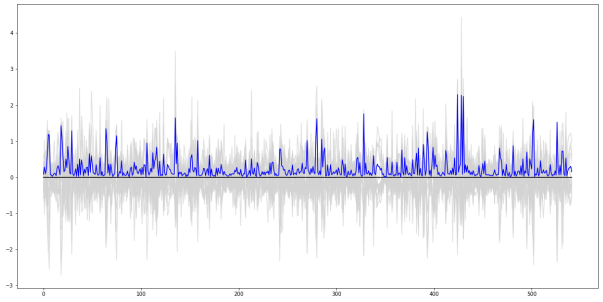

Figure 2: Mean(in black) and Variance (in blue)

Fig.2 and Fig.3 show the first and second derivative respectively. As we can see the first derivative is very noisy and changes sign constantly since the functions go from increasing to decreasing very rapidly and often. The second derivative is again very noisy for the same reason.

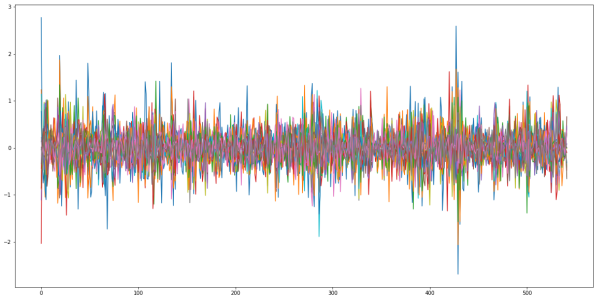

Figure 3: First derivative

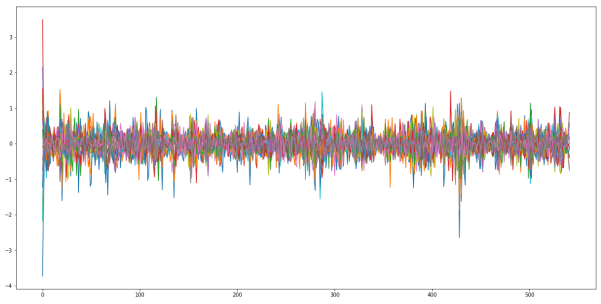

Figure 4: Second derivative

We used an outliergram to find possible outliers in the data. It plots the Modified Band Depth (MBD) on the Y axis and the Modified Epigraph Index (MEI) on the X axis for each functions. The bands represents the boundary between outlying and non-outlying observations. If a point is out of them it might be an outlier. This is the case of the blue point, so we decided to remove the function associated with it to improve the efficacy of the smoothing techniques in the next section.

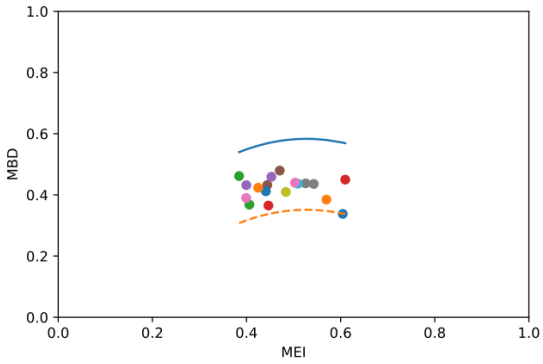

Figure 5: Outliergram

1.2 Smoothing the dataset

In order to find a suitable smoothing for the dataset we test two different techinques, namely kernel smoothing and basis smoothing. For kernel smoothing three different way to compute the Hat matrix are used: Nadaraya Watson, KNeighbors, Local Linear Regression. For basis smoothing a Kspline basis is used with 10 basis . The optimal smoothing parameter is selected minimizing the Generalized cross validation (GCV) score. GCV is a way of estimating the prediction error of a model by using the observed data to estimate the prediction error at different values of the smoothness parameter. The GCV estimate is obtained by minimizing the sum of the squared residuals of the model, subject to a penalty on the complexity of the model. The penalty is determined by the smoothness parameter and the number of data points. The range for the search is selected looking at the best score for different ranges. The smoothing with the best score are compared to select the best one. The figures bellow show the smoothed function with the best smoothing parameter against the original one, for one function chosen randomly. How we can see both the Nadaraya Watson and Basis smoothing oversmoothed the data while KNeighbors did not, although the latter one seems to have still a bit of noise. In conclusion the Local Linear Regression appears to be the best one since it didn't neither oversmoothed or undersmoothed the data, for this reason it will be the method used latter on.

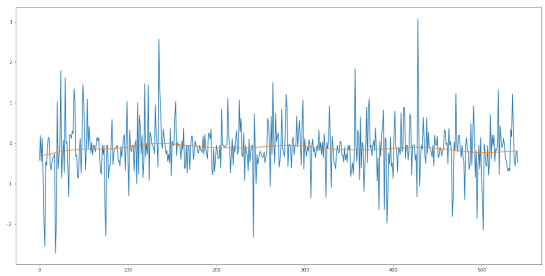

Figure 6: Nadaraya Watson smoothing with parameter = 30

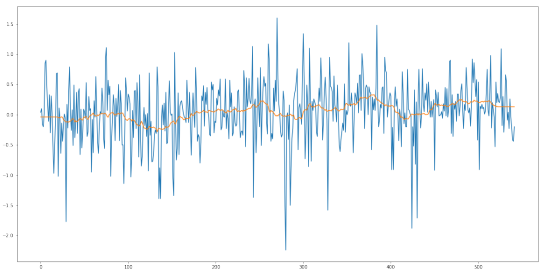

Figure 7: KNeighbors smoothing with parameter = 39

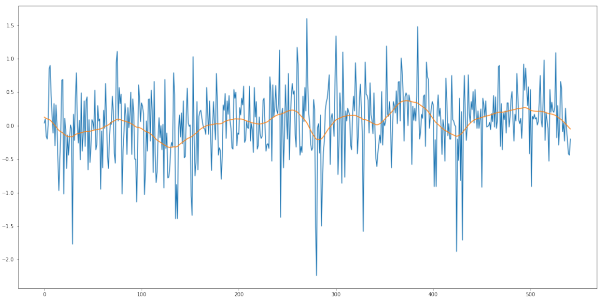

Figure 8: Local Linear Regression smoothing with parameter = 20

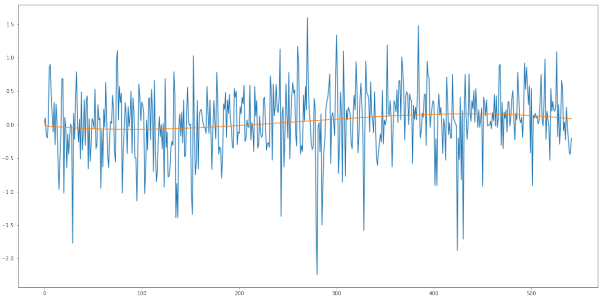

Figure 9: Basis smoothing with parameter = 5

1.3 Functional PCA

We use the smoothed data to perform functional PCA. The goal is to find a set of basis functions that can be used to represent the original functional data as a linear combination. These basis functions are called "principal components", and they are chosen such that they capture as much of the variation in the data as possible. Before applying PCA we standardize the data since PCA is sensitive to different scales in the data. In order to find a suitable number of components we plotted the cumulative explained variance ratio for different values. How we can see from Fig. 11 5 components explain more than 80 percent of the variance, thus this numbers is sufficient for representing the data .

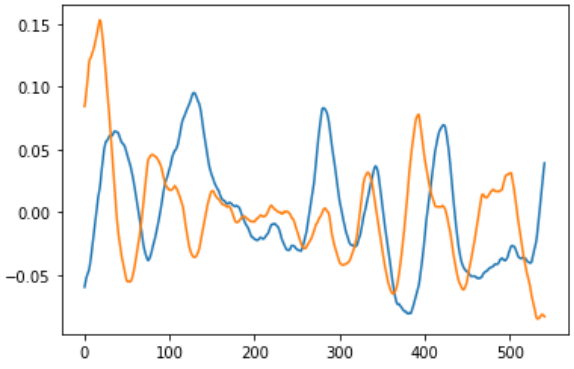

Figure 10: first and second PCA component

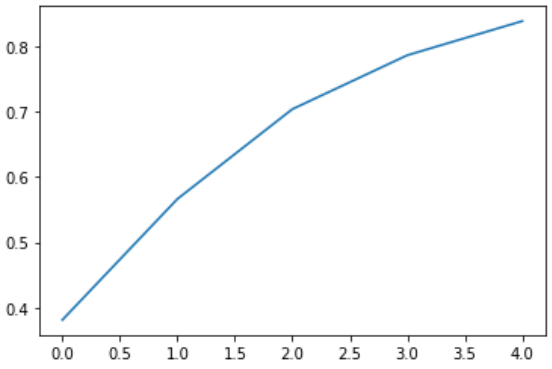

Figure 11: Cumulative explained variance ratio

1.4 Clustering analysis

We performed a clustering analysis on the scores returned by the PCA. For identifying the clusters we used a K-means algorithm setting the number of cluster to 3, this value has been chosen looking at the scatter plot of the first two PCA scores where we can clearly see three distinct group of points. Fig. 12 shows the 3 clusters the K-means algorithm found. The light blue cluster are points with negative first component scores and second component scores near zero, the yellow cluster's points have positive second

component score and first component close to zero and the black ones have positive first component score and negative second component's one.

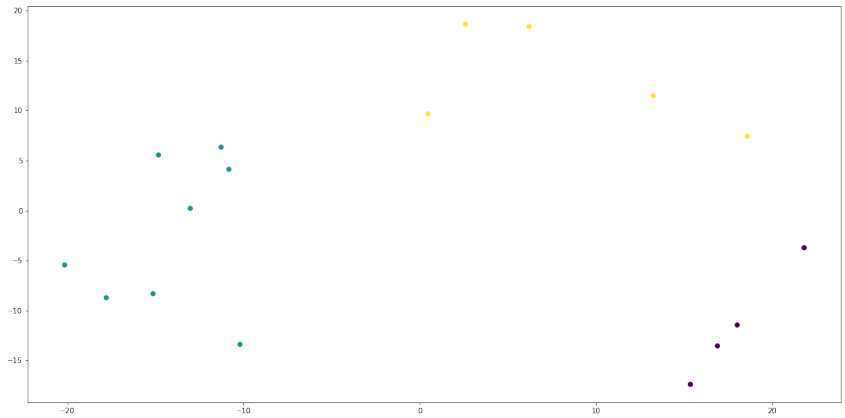

Figure 12: first against second PCA component. The colors represent the different clusters

The figures below show the functions belonging to each cluster. As we can see the functions in the same cluster have similar behaviors.

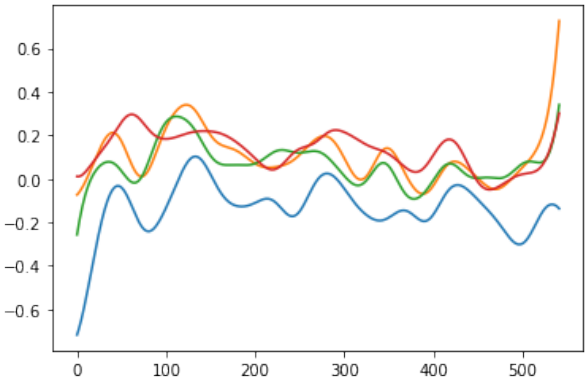

Figure 13: First cluster's functions

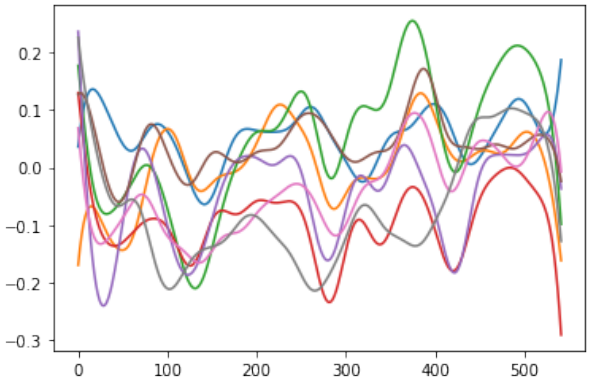

Figure 14: Second cluster's functions

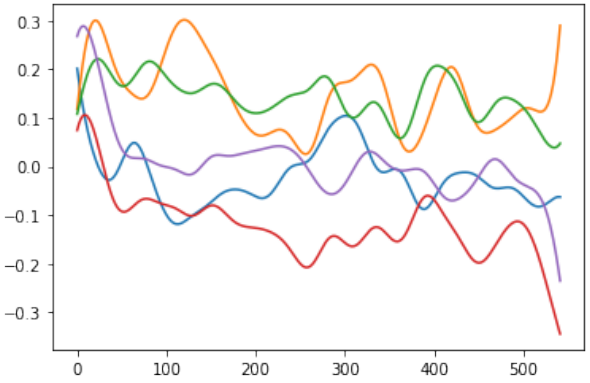

Figure 15: Third cluster's functions

2 Exercise 2: CT-Scan Dataset

The purpose is to use features extracted from more than 50000 CT-Scan slices to predict the relative position of the slice within the patient. We are in a realistic situation where a radiology expert has labelled data for many patients (namely the source dataset), but the model will be applied to a new patient for which no labels are available(namely the target dataset). To simulate this scenario, we will remove the patient with the most data and treat it as the target patient.

2.1 Question 1 - Transfer learning problem

We have divided the source dataset into a train and a test dataset. A PCA is applied to the 384 input features and the first 2 components are plotted. In the picture below we can see the features' components of the source dataset compared with the one of the targets as long as the target variable distribution for both the datasets. We can see that the feature space is pretty similar but the distribution of the target variable differs a lot. In the source dataset, we have a kind of bell-shaped one against a uniform distribution. We can conclude that we have a problem well suited for transfer learning, more precisely we have a task shift problem since the distribution of the target variable changes between the considered datasets but the feature space remains more or less the same.

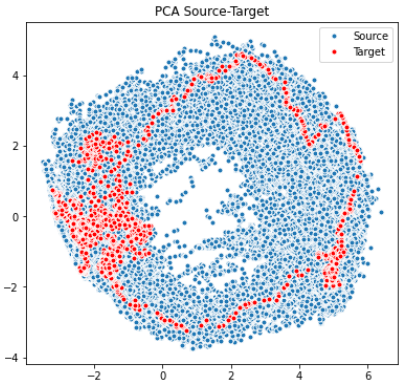

Figure 16: PCA source target

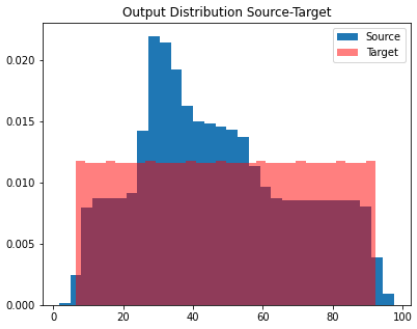

Figure 17: Output distribution source target

2.2 Question 2 - Ridge regression model

After scaling all the datasets using Min Max scaling (to avoid the model to give more importance to some variables respect to the others) we fitted a Ridge regression model on the training data set and we computed the source risk on the test set as long as the target risk, even tho we shouldn't be supposed to have access to the labels of the latter.

The figure shows the a scatter plot of the target variable vs the prediction on both the datasets. The diagonal straight line represents perfect predictions. We can see how the predictions on the target dataset are generally more distant from it, and this is reflected

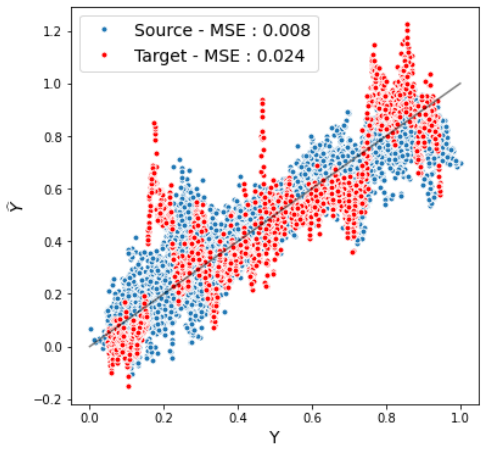

Figure 18: target variable vs predictions

on a mean squared error of 0.024, 3 times higher than the one on the source dataset. This is caused by the differences between the two dataset, consequently transfer learning should be applied to compensate such differences.

2.3 Question 3 - Covariate shift

The source risk is not a good estimation of the target risk for what said before. In the figure below we plotted first PCA component against the target variable for both dataset in order to understand is the covariate shift might be realistic or not. Ideally we want, for same values on the x-axis, values on the y-axis that are concentrated in the same area. We can see that we have more or less that.However the y values have different variances in the two datasets. This means that we don't have exactly the same conditional probability, but in practice we can assume the covariate-shift assumption to be realistic.

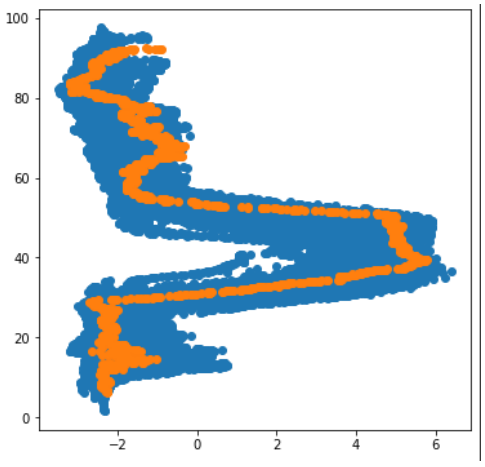

Figure 19: target variable vs first PCA component

2.4 Question 4 - KDE

We apply KDE to the data. KDE is a non-parametric method used to estimate the probability density function(PDF) of a random variable.The PDF is estimated by placing a kernel function at each data point and summing the contributions of all the kernel functions. We sample a random number of data points on which we apply KDE. Fig 5 shows the number of data instances sampled against the time of execution for the KDE. The plot can be used to estimate the time complexity of the algorithm, in our case we have exponential since the computational time increases exponentially.

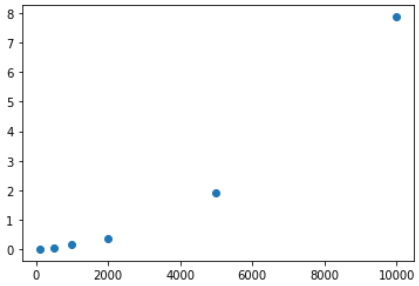

Figure 20: # of samples vs time of execution for KDE

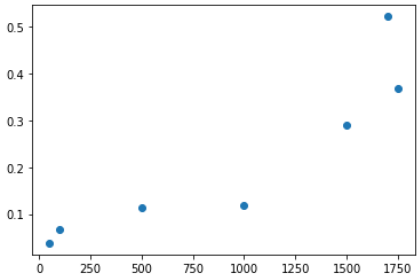

Figure 21: # of samples vs time of execution for NNW

2.5 Question 5 - Nearest neighbor weighting (NNW)

We apply nearest neighbor weighting (NNW) to the data. NNW s a technique used to assign weights to the observations in a dataset based on the distances between observations. Each observation is assigned a weight that is inversely proportional to its distance from the observations in the target dataset. In this case we have a linear complexity as shown by Fig. 6. We could prefer NNW over KDE since is faster to execute for an high number of samples. We then fitted a Ridge regression using the weights from NNW.We can see from Fig. 7 how the algorithm is assigning bigger weights to the data points that are closer to the target feature space.

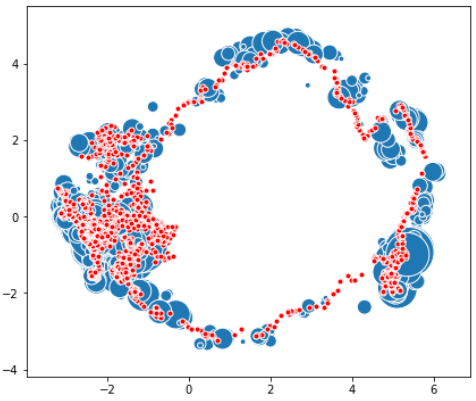

Figure 22: weighted PCA components

2.6 Question 6 - Tuning the algorithm

Since we are not supposed to have labels for the target dataset we have to use an unsupervised way in order to tune the number of neighbors used by the NNW. For this purpose we first multiplied each data istance by the weight computed by the NNW,then we used a KDE to estimate the new distribution of the first PCA component and compare it with the target's one. In order to quantify the difference between the two distributions KL divergence is used. The number of neighbors that minimize the Kl divergence it will be selected. We evaluated the algorithm for 5,10,25,50 neighbors, 50 gave the best result with a divergence of 17.15.

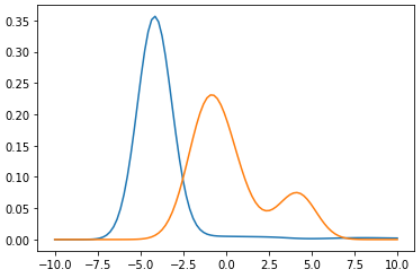

Figure 23: estimated weighted distribution on the source (blue) and target (orange)

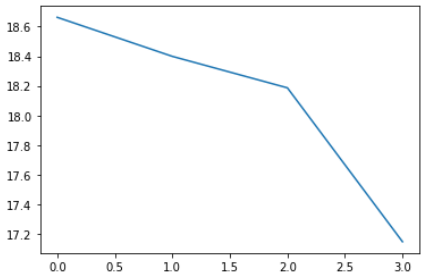

Figure 24: # of neighbors vs KL divergence

2.7 Question 7 - KLIEP

KLIEP works by estimating the relative importance of each sample in the estimation of the target distribution. This is done by comparing the target distribution to the reference distribution, which is the distribution from which the samples were drawn. The relative importance of each sample is then used to weight the sample in the final estimate of the target distribution.

We tried fitting the KLIEP using different gamma values, respectively 0.01, 0.1, 0.5, 1, 2 and 5. For each value we compute the KL divergence score as before. We noticed that increasing the value of gamma the KL divergence increases as well. The lower one (0.65) is given by a gamma value of 0.01. Thanks to this we can conclude that this method works significantly better than NNW.

2.8 Question 8 - Cross-validation

By passing a list of gammas, the algorithm performs a cross-validation to find the best value. The parameter that maximizes the J score is chosen. This score is defined as the averaged sum of the log of the source instances weights. In our case the value of 1 is selected with a J score of 10.9. Nevertheless, the KL divergence computed is very high (18.17) if compared with other gamma values. So we conclude that it is not the best value for gamma, but we rather use the one we found manually in question 7.

Model	Source Test Error
Random Forrest	0.055
AdaBoost	0.046
MLP	0.007

Table 1: Models performance on source dataset

Method	KL Divergence
KMM	0.18
IWC	0.27
IWN	1.0

Table 2: Instance based methods performance

2.9 Question 9 - Testing models

We evaluated other regression models other than the Ridge regression. Since we have used only a linear model until now we evaluate the effectiveness of some non-linear model to see if they are better suited for the task. We evaluate a random forest,an Adaboost model and a multi-layer perceptron (MLP). All of the models are paired with NNW. Default values for the hyperparameters are used. The MLP has 4 layers with 64,32.16 and 8 neurons each and one batch normalization layer to help the convergence. A relu activation function is used. The MLP has the best performance on the source test set after weighting the data points.

We also evaluated other instance based methods, namely kernel mean matching (KMM), importance weighting classifier (IWC) and importance weighting network (IWN). We used default values for the hyperparameters.For the IWN a similar MLP as before is used The methods are evaluated using the same methodology used in question 6, i.e. computing the KL divergence between the new weighted source distribution and the target's. All the methods performed way better than NNW. KMM had the best performance with a KL divergence of 0.18 and we can see from Fig 10 how the weighted source distribution resemble the target one.

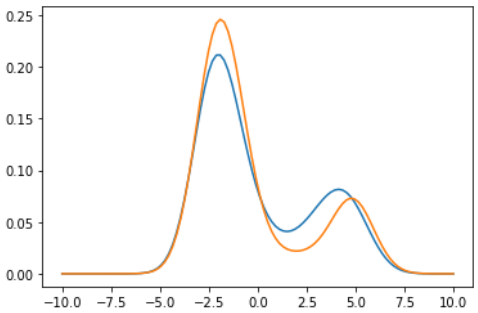

Figure 25: New weighted distribution using KMM

Model	HP	Values
Ridge	Alpha	[0.1,10]
AdaBoost	Depth	1,2,4,6
	Learning Rate	[1e-4,1e-1]
MLP	Neurons	8,16,32,64,128
	Activations	relu,tanh
	Epoch	10,25,50,100
	Learning rate	[1e-4,1e-1]

Table 3: Hyperparameters space

Model	Target MSE
Ridge	0.0135
AdaBoost	0.011
MLP	0.0013

Table 4: Performance of the best combination of HP

2.10 Question 10 - Testing best possible models

Finally we took 10 random target data with their labels and we try to build the best possible model. The models are evaluated using the MSE computed on the labeled target data. As instance based method we used KMM since it has been proved to be the best one until now. We first tuned the gamma parameter of the KMM monitoring the KL divergence. A gamma of 1.7 gave the best result with a divergence of 0.17. We then proceeded to tune and evaluate 3 different models: Ridge regression, AdaBoost and MLP. The MLP has the same structure used before. Table 3 shows the search space of hyperparameters, the best combination is selected looking at the lowest MSE on the labeled target data. Table 4 shows the performance of the best combination of hyperparameters for each model, the MLP was the one that performed the best with an error 10 times lower than the other models. Moreover the difference between the error on the source data and target data is extremely low (around 2e-4), and this led us to conclude that it represents the best model we have evaluated on the task.