Domain Adaptation in Acoustic Rainfall Sensors

Author:

Arman Naseri Jahfari

September 5, 2019

Domain Adaptation in Acoustic Rainfall Sensors

by

Arman Naseri Jahfari

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Thursday September 12, 2019 at 9:30 AM.

Student number:	4036735	
Thesis committee:	dr. D.M.J. Tax,	TU Delft, supervisor
	dr. ir. R. Heusdens,	TU Delft, chairman
	dr. M. Loog,	TU Delft

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

Dedicated to the memories of Moreno and Ashley. They have shown me and 'the gang' some invaluable traits needed in this journey of life.

Your positive attitude, curiosity, perception, guts and most importantly, a great sense of humor will be remembered and passed on.

CONTENTS

1

BACKGROUND INFORMATION

1.1 PROBLEM DESCRIPTION

The problem of rainfall analytics is non-trivial and yet, very important. In The Netherlands, damage due to rainfall, like flooding and traffic jams, does not frequently occur. However, when it happens, the financial and social costs are great[1][2]. Research on prediction of future rainfall also show that these situations will increase in intensity and frequency, due to climate change[3]. Therefore, measures must be taken to get more insight in the behaviour of extreme rainfall. Although satisfactory weather prediction models exist, they are not able to provide useful information on four criteria of rainfall estimation: accuracy, high temporal and high spatial resolution and range. The high temporal and spatial resolution is needed, because rainfall is non-stationary over time and space. Ideally, a street-level resolution prediction is needed. This high-resolution data can then be passed to urban hydrology models, which require an accurate estimate of the rainfall, so that it can be analyzed where flooding might occur, based on current infrastructure, e.g. sewage systems.

1.2 METHODS

Current measurement methods trade-off these criteria. State-of-the-art methods discussed are weather satellites, radars and rain gauges. These are compared to a novel acoustic rainfall estimation method. It will be discussed, that the latter can overcome limitations from other methods.

The following table quantitatively scores each method for every of the four criteria, followed by an elaboration.

	Temporal resolution	Spatial Resolution	Error performance	Range
Weather Satellite[4]	−	−	0	+
Weather radar[5][6]	−	−	0	+
(Acoustic) Rain gauge	+	+	+	−

Table 1.: *Comparison of different rainfall estimation systems. Four criteria are scored qualitatively. Plus, minus and zero indicate strong, weak and neutral satisfaction, respectively.*

Weather satellites infer rainfall by remotely sensing electromagnetic radiation from clouds. Corrections need to be applied for instrumental and atmospheric factors, resulting in low error performance. The satellite is positioned at a large distance to the Earth. Therefore, the range is large, but the spatial resolution low. Satellites orbiting around the Earth, have extremely low temporal resolution. Geostationary satellites mitigate the latter, by remaining stationary w.r.t. the Earth's rotation. Also, they are also very costly.

Weather radars transmit electromagnetic waves into the atmosphere. Part of it is reflected back by raindrops. The rainfall intensity is then infered from this reflected signal. Similar to satellites, the inference has a poor error performance. The electromagnetic beam tends to become wider for larger distances, which degrades spatial resolution. Blockages due to hills and buildings also cause inaccuracies. Weather radars measure while rotating 360 degrees and rainfall might have changed significantly in this time, which limits temporal resolution.

Spatio-temporal resolution and error performance can be greatly improved with rain gauges. Laser-based rain gauges, collect rain in a funnel, which releases water droplets with known volume. A laser beam at the bottom is interrupted, every time a droplet passes. Then, the amount of interruptions per unit time gives an estimation of the rainfall. In contrast to the previous methods, rain is measured directly, resulting in high error performance. The temporal resolution is only limited by practical factors like hardware. The spatial resolution is also high, because it is defined by the size of the funnel, which

is relatively small. Unfortunately, the small size of the rain gauge results in a small range; rain can only be physically measured locally. One solution is to distribute these sensor densely over a wide area. The Koninklijk Nederlands Meteorologisch Instituut (KNMI)/ Royal Netherlands Meteorological Institute employs rain gauges over the Netherlands, but due to high costs, they are sparsely distributed as can be seen in the following figure.

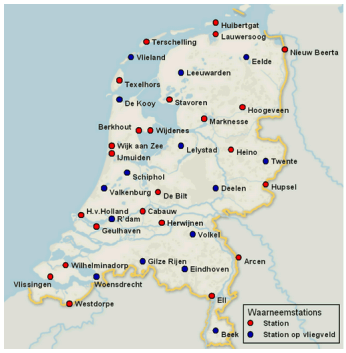

Figure 1.: *Weather station locations of the KNMI[7]*

A cost-effective acoustic 'rain gauge' is proposed[8], that can be densely distributed. Rainfall is infered from physical contact of rain drops onto a membrane. This results in decent error performance, while enabling a high spatio-temporal resolution. However, every acoustic sensor must be individually calibrated, which is infeasible if the number of sensors is large.

1.3 RESEARCH QUESTION

To tackle the calibration problem mentioned earlier, it is researched how to *transfer* a pattern recognition model trained in a certain point in time/space with a certain acoustic sensor, to another point in time/space **and or** sensor.

This is achieved through transfer learning. The acoustic sensor data, used to train a model, is defined as the source domain. Then, the data of another acoustic sensor, where the model is tested on, is defined to be the target domain. This transfer learning problem is scoped down to a subfield called domain adaptation, by the following observations. The source and target domain share the same task to estimate rainfall. Furthermore, the number of features are equal in both domains.

From the problem description, our project is scoped down to researching and engineering the following objective:

"How can non-stationary inference be made of rainfall, from acoustic data, using domain adaptation."

For this performance transfer, measurements of highly accurate ground truth sensors and simple, cost-effective acoustic sensors must be correlated. Then, the resulting model must be adapted to other acoustic sensors

2

DATA ACQUISITION

In this chapter, it is explained how data acquisition is performed to collect data for a pattern recognition model. An overview of the process is given, where after the hardware and software of the self-developed acoustic sensor and the ground truth weather station is explained. Finally, the data acquisition experiment is described. The framework to process this in a pattern recognition pipeline is described further in chapter 3.

2.1 TOP-LEVEL DESIGN

An overview of the flow of the data acquisition system is given in figure 2. First, the acoustic and ground truth sensor synchronously perform timestamped measurements. Then, the data is communicated over WiFi to a server. After preprocessing this data (removing corrupted data, aligning the data based on timestamps and formatting the data to a more suitable form for a machine learning library), it is input in the pattern recognition pipeline.

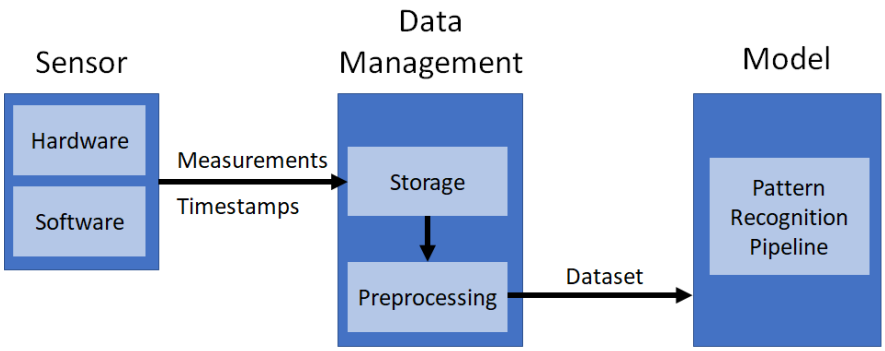

Figure 2.: *Data acquisition overview: The acoustic and ground truth sensors collect acoustic and rainfall data, respectively. This is then communicated to a server for storage. After preprocessing this data, it is input in a pattern recognition pipeline to construct a model that infers rainfall from the given audio data.*

2.2 HARDWARE

This section describes the hardware design of the acoustic sensor. Furthermore, an overview is given of the hardware of the accurate ground truth sensor that is used.

The self-developed acoustic sensor consists of a 20 Hz to 20kHz range electret microphone with an integrated MAX4466 amplifier. The resulting waveform is passed through a MCP3304 13-bit Analog-to-Digital Converter (ADC), which samples at 43 kSamples/second. This digital signal is then communicated to a Wemos processing board, through the Serial Peripheral Interface (SPI). This board includes an ESP32 microcontroller, which includes a WiFi chip for Internet of Things (IoT) capabilities, needed to transmit measurements to a server. Furthermore, it contains a user-friendly USB connection to a computer from where programs can be written using simple Arduino C code. An illustration of the (components of the) hardware used is given below.

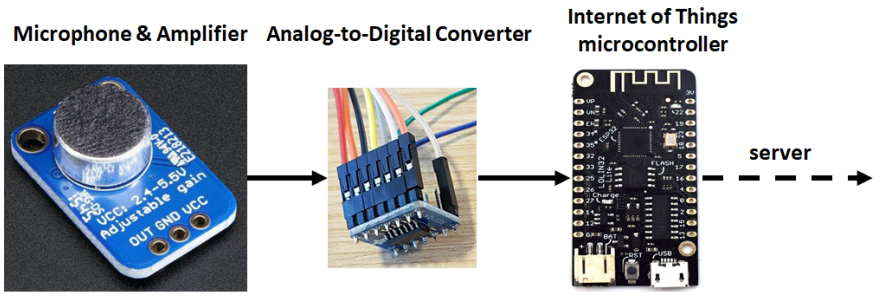

Figure 3.: *Subcomponents of the acoustic sensor. Left: Microphone and amplifier. Middle: ADC. Right: Wemos Internet of Things board.*

The sensors are deployed outside and the inside must therefore be robust to environmental factors, like dust and rainfall. Therefore, they are packaged in PVC pipes. The microphones are taped to a speciecap. This is a soft and flexible PVC cover, which allows for vibrations to propagate through the cap into the packaging. This minimizes attenuation from the cover, enabling measuring vibrations due to waterdrops hitting the speciecap directly, as well as acoustic pressure waves from the environment, reaching the inside of the PVC pipe. An example of an assembled sensor can be found below:

The annotations of the acoustic data are provided by the rain gauge of a weather station from the Trans African Hydro Meteorological Observatory (TAHMO). This rain gauge communicates its measurements digitally through the Serial Digital Interface at 1200 baud (SDI-12) protocol. Similar to the acoustic sensor,the measurements are then transmitted to a Wemos IoT board, which in turns transmits the mea-

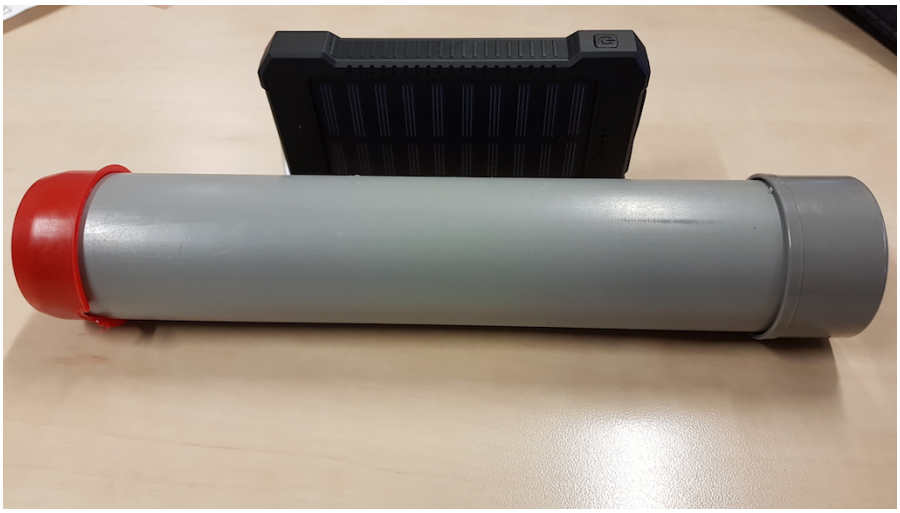

Figure 4.: *Acoustic sensor. From left to right: Red speciecap with microphone glued on the inside, PVC pipe where electronics are suspended, PVC bottom cap. In the back is a solar kit(solar panel, battery, USB ports) to provide electronics with power*

.

surements to a server over WiFi. A picture of a TAHMO station is shown below:

Figure 5.: *TAHMO station with a rain gauge sensor module on the top right, used to provide rainfall annotations for the acoustic sensor.*

2.3 SOFTWARE

This section describes the firmware of the microcontroller used in the acoustic and ground truth sensor. Because the firmware is similar for both acoustic and ground truth, it is described in one section, with differences elaborated when necessary.

The software is structured as a Finite State Machine (FSM), which runs different code depending on the current state and can change states based on a trigger. The FSM is presented below with an explanation of the states:

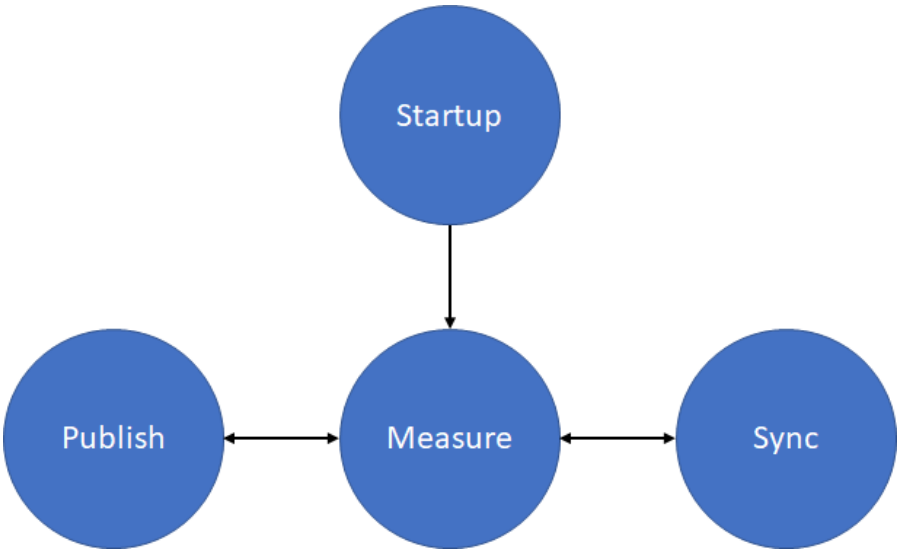

Figure 6.: *An overview of the FSM implemented in the Wemos electronic board*

.

The startup state starts up basic functionality of the board. It does so according to the following procedure:

- Initiate analog input pin connected to the microphone for the acoustic sensor.
- Connect to WiFi.
- Initiate SPI communication for the acoustic sensor and SDI-12 communication for the ground truth sensor.
- Synchronize with an external Network Time Protocol (NTP) network. NTP is an internet protocol that can accurately synchronize the internal clock of a device, to a universal time reference. This is essential for synchronizing the measurements of the acoustic and ground truth sensors. Furthermore, when the acoustic sensors are deployed, time drifts can result in inaccurate rainfall maps.

- When the startup is finished, the device proceeds to the Measure state

The measure state is the core of the program's functionality and samples signals measured by the microphone in the acoustic sensors and the rain gauge in the ground truth sensor.

- All devices are idle until the synchronized internal time is a multiple of a measurement interval of five seconds. In other words, all devices start a measurement at the fifth, tenth, up to the sixtieth second of a minute.
- When it is time to measure, the acoustic sensor acquires 4000 samples at a sampling rate of 43 kS/s, resulting in roughly 90 milliseconds of audio data every five seconds. The ground truth sensor acquires only one rainfall sample in this measurement interval.
- After measuring is complete, the device changes to the publish state.

Finally, in the publish state, the device communicates it's measurements to an external server through WiFi.

- In the publish state, the device starts by setting up an MQTT client. MQTT is a lightweight, small overhead messaging protocol.
- The data is sent to the MQTT server, where it is then stored in a database.
- The device returns to the measuring state, where it sleeps to preserve energy, until the next measurement must be performed. A small margin is included to account for time needed to wake up.

2.4 RAINFALL EMULATION EXPERIMENT

Now that the sensors are explained, this section describes the data acquisition through a rainfall emulation experiment. The experimental setup is schematically shown in the figure below, with a description of all components.

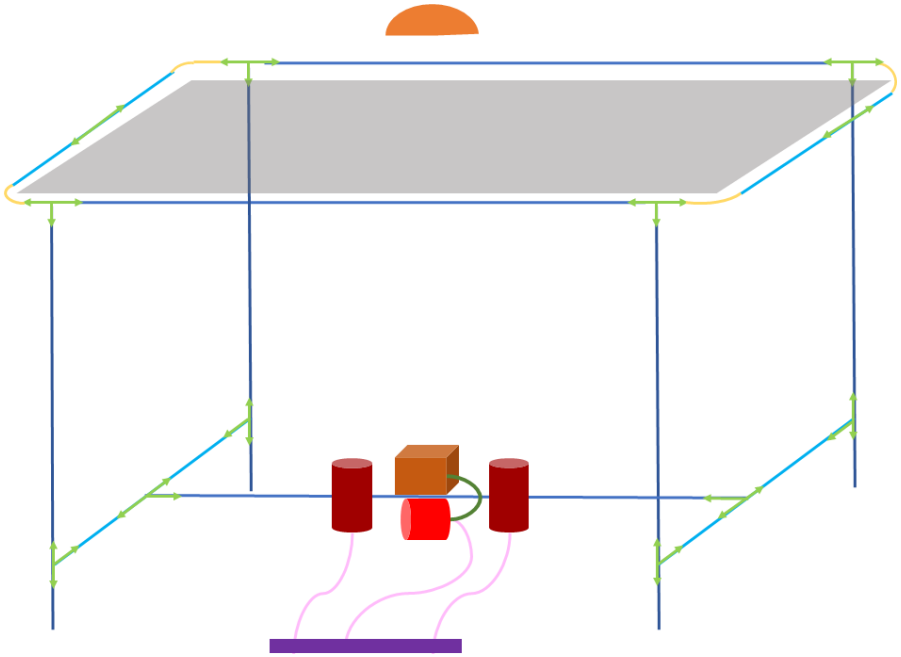

Figure 7.: *Schematic of rain emulation experiment for data acquisition*

.

- PVC T-shaped joints: Joints to connect PVC pipes of framework.
- PVC pipes(long): Long pipes for framework to hold all components.
- PVC pipes(short): Short pipes for framework so that wiring of power supply to sensors can also be kept short.
- Curved PVC joint: Due to the geometry of the T-joints, a curved pipe is needed to create a rectangle framework.
- Sprinkler: GARDENA Zoommax oscillating water sprinkler. The sprinkler providing water drops, is suspended to a pole with tie-wraps and this pole is in turn suspended to a ladder. Furthermore, water is provided by a waterhose. These are not depicted in the schematic.
- Mesh grid: The sprinkler ejects large beams of water. To emulate rain, a mesh grid is placed under the sprinker, that scatters these beams to small water droplets, improving uniformity. The

mesh grid hole size is five square millimeters. Furthermore, it is suspended to the framework with tie-wraps.

- TAHMO ground truth sensor: Positioned exactly below the sprinkler.
- Wemos electronic board: Datalogger for the measurements of the ground truth sensor. Packaged in PVC.
- Audio cable: Used as communication wire from TAHMO ground truth sensor to Wemos datalogger.
- Acoustic sensors: Packaged as in figure 4 and positioned 25 centimeters from TAHMO ground truth sensor.
- Power socket: Power supply to sensors
- Power cables: USB power cables providing power to sensors through power socket.

The data acquisition experiment has been performed twice over two days. The first time, two acoustic sensors were used as in figure 7. In the second experiment four acoustic sensors were used, where the two additional sensors were placed 25 centimeters from the original two sensors. Thus, all sensors, including the ground truth sensors, are placed at equal distance, with the ground truth sensors centered directly below the sprinkler. The first and second experiment, performed on August 9th 2018 and September 12th 2018, will be denoted as E1 and E2, respectively. The left and right acoustic sensors in figure 7 will be denoted as m1 and m2, respectively. In E2, m1 and m2 are positioned the same as E1. The additional sensor placed left of m1 will be denoted as m3 and the sensor right of m2 will be denoted m4. In other words, the order of acoustic sensors in E2 is from left right: m3, m1, m2, m4.

3

BASELINE

3.1 DATA PREPROCESSING

The acquired data contains corrupted data that need to be filtered out. When the rain gauge is saturated with too many raindrops, it reports a measured value of zero. In that case however, the internal temperature sensor also reports a value of zero. This is an indication of a corrupted measurement and thus, these are filtered out. Furthermore, for some very high rainfall measurements, the model was not able to predict them. The intensity of rainfall of these measurements do not occur in nature and therefore, were also chosen to be filtered out . Finally, due to communication errors, the acoustic sensors were not operational 100% of the time. This results in datasets, with different sample sizes per microphone. The resulting measurement size per microphone is given below:

	Measurement size
E1, m1	1941
E1, m2	1939
E2, m1	4479
E2, m2	4551
E2, m3	4871
E2, m4	4436

Table 2.: *Measurement size of acquired data for every microphone in each of the experiments. Every measurement consists of 4000 samples.*

A feature vector in the time-domain is defined as:

$$x_t = \begin{bmatrix} x_t(0) & x_t(1) & \dots & x_t(S-1) \end{bmatrix}^T \tag{1}$$

where $x_t \in \mathbb{R}^{1\times(S-1)}$ and every time feature vector consists of $S = 4000$ microphone samples.

The time feature matrix consists of these time feature vectors stacked together:

$$X_t = \begin{bmatrix} x_{t,0}(s) & x_{t,1}(s) & \dots & x_{t,N-1}(s) \end{bmatrix}^T \tag{2}$$

where $X_t \in \mathbb{R}^{(N-1)\times(S-1)}$. It consists of $N-1$ measurements, with each measurement consisting of $S = 4000$ microphone samples as mentioned earlier.

3.2 MFCC FEATURE GENERATION

The acoustic sensors measure 4000 samples in each measurement interval, annotated by a single rainfall measurement from the TAHMO ground truth sensor. As was mentioned in section 2.3, the sensors measure at a sampling frequency of 43 kiloSamples/s, resulting in a measurement time of 90 milliseconds. This results in feature vectors with 4000 features.

Suppose that a waterdrop is measured at the beginning of a sample, while a similar one is measured at the end of another sample. This would cause the first features in the first sample to have high values, while in the second sample, the last features would have high values. The pattern recognition model would distinguish these two situations to be different, while they are similar water drops, describing the same amount of rainfall. This representation is not time-invariant within the measurement interval. This is solved by using a spectral representation, because the magnitude spectrum is insensitive to timeshifts of the signal (only the phase spectrum changes). Furthermore, this representation is disadvantageous for a pattern recognition model due to the high dimensionality. Therefore, dimensionality is desirable. These problems can be solved by using the Mel frequency cepstral coefficients (MFCCs)[9]. The procedure to compute MFCCs for the acquired data, will be explained.

- Data framing: The audio signal of the entire microphone recording is non-stationary and therefore the spectral content is time-dependent. As a solution, the audio signal is split in small fragments, called frames, where after a Fourier transform is applied on each frame. The dataset acquired is implicitly already split up in frames of S microphone samples. Thus, every frame corresponds to a time feature vector in equation 2. Furthermore, because the time interval of five seconds is much larger, than the measurement duration (approximately 90 milliseconds), the frames are assumed to be independent. Thus, the framed audio

data is already given earlier as the stacked time feature matrix in equation 2.

- Apply Hanning window: After dividing the audio data in frames, a Hanning windowing function is applied. The frames are non-periodic fragments of the recording. Therefore, the endpoints of the waveform in a frame are discontinuous. These artificial discontinuities result in high-frequency components in the frequency domain, not present in the original time-domain frame. The Hanning window dampens the amplitude of the waveform at these endpoints to zero, to reduce the discontinuity. This is applied to every time feature vector, resulting in a windowed time feature matrix as defined in equation 3.

$$\begin{aligned} X^{t,w} &= \begin{bmatrix} x_0^t(s)w(s) & x_1^t(s)w(s) & \dots & x_{N-1}^t(s)w(s) \end{bmatrix}^T \\ &= \begin{bmatrix} x_0^{t,w}(s) & x_1^{t,w}(s) & \dots & x_{N-1}^{t,w}(s) \end{bmatrix}^T \end{aligned} \tag{3}$$

- Estimate periodogram: Now, for every windowed time feature vector, the Discrete Fourier Transform is computed through the Fast Fourier Transform. Because the transform is applied on short frames, instead of the entire audio data, it is also known as the Short Time Fourier Transform (STFT). The periodogram of the STFT for a single frame/measurement is given in equation 4.

$$\begin{aligned} x_n^F(k) &= |\mathcal{F}\{x_n^{t,w}(s)\}|^2 \\ &= \left| \sum_{s=0}^{S-1} x_n^{t,w}(s) e^{\frac{-2j\pi ks}{S}} \right|^2 \end{aligned} \tag{4}$$

where $k = 0, 1, ... \frac{S-1}{2}$ is the frequency bin index of the single-sided spectrum. Then, applying this to all measurements, the feature vector matrix is obtained:

$$\begin{aligned} X^F &= |\mathcal{F}\{X^{t,w}\}|^2 \\ &= \begin{bmatrix} x_0^F(k) & x_1^F(k) & \dots & x_{N-1}^F(k) \end{bmatrix}^T \end{aligned} \tag{5}$$

- Apply the Mel filterbank on theperiodograms: So far, the procedure for computing MFCCs is the same as a vanilla Fourier Transform. Now, a triangular filterbank is applied for dimensionality reduction, which can be regarded as a moving weighted average filter, where the filter width increases, when moving over all considered frequencies. This width increases due to

the use of the Mel scale. This scale aims to mimic the non-linear human auditory sensitivity to frequency changes, by being more discriminative at lower frequencies and less discriminative at higher frequencies, due to its exponential relationship with frequency. In other words, changes in the lower frequency area is generally perceived as a large change in pitch/Mels. At higher frequencies, the human auditory system is less sensitive to changes in frequency and thus, it is perceived as a small change in pitch/Mels. The use of the Mel scale originates from automatic speech recognition models, that mimic human auditory perception. This has however proven to also work for non-speech application. For this research, the number of filters used, is 40. An example of a triangular filterbank is given in figure 21.

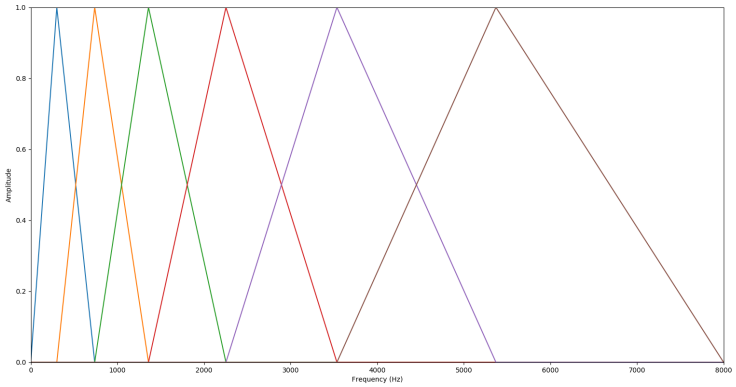

Figure 8.: *Example filterbank where the sampling rate is 16000 Hz and six filters are generated.*

- Apply logarithm function: The logarithm with base ten is applied on all filterbank energies. One motivation for this is that it emulates the human auditory system, which also perceives the magnitude of a sound wave logarithmically. More specifically to domain adaptation, the log power allows for cepstral mean normalization. This is a technique that removes the effect of the transfer function of a microphone. In doing so, different acoustic sensors will have more similar characteristics, which is beneficial, if prediction is performed on them, using one model. Because it is a technique beneficial for domain adaptation, it is further discussed in Chapter 4.

- Discrete Cosine Transform (DCT) Transform: A spectral transform is applied again. Instead of the Fourier transform, the DCT suffices, because the magnitude square operation in the periodogram results in real Fourier coefficients. The filters in the filterbank are overlapping and thus, the resulting coefficients are correlated. The DCT transform decorrelates them, i.e. the covariance matrix of the data is diagonal.

3.3 PERFORMANCE EVALUATION

To assess how the performance of a prediction model will generalize, 10-fold cross validation is used. It is motivated why the R^2 score is chosen to evaluate the performance of a model. Furthermore, it is discussed that every fold contains a random permutation of a microphone's data. Lastly, the use of an average R^2 score for domain adaptation is explained .

The ground truth sensor, used to annotate the measurements of the acoustic sensors, has a relative measurement error (in percent) w.r.t. the ground truth value. Consequentially, the **absolute** measurement error and thus the variance over multiple measurements, is proportional to the magnitude of the ground truth. This is indeed true for the ground truth time series of microphone E1, m1 below.

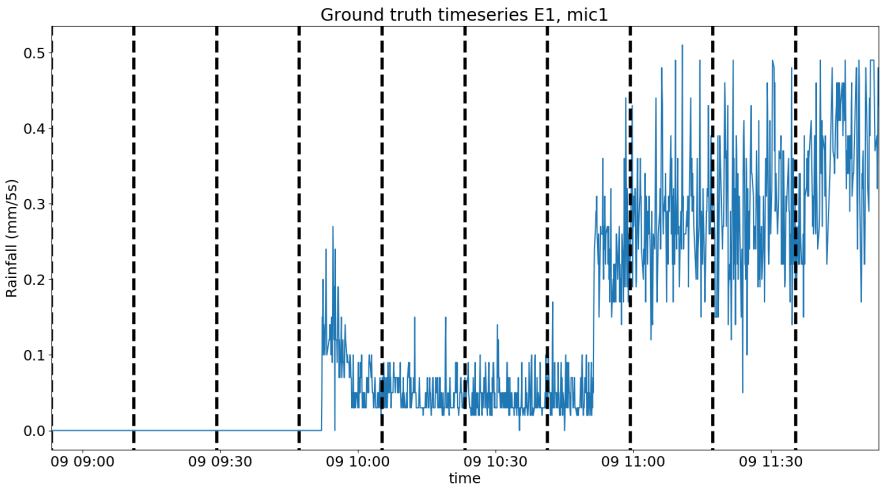

Figure 9.: *Ground truth time series corresponding to E1, m1. Black lines divide the values in ten cross-validation folds.*

The measurements have been evenly split into ten folds. It can be seen that the variance of the last three folds are much larger than the other folds. Similarly, the absolute prediction error and error variance of a model when using the mean square error (MSE), typically correlates with the magnitude of ground truth measurements as well. In figure 9, the MSE would report a larger absolute error and error variance for the last three folds, than the other ones. This holds even if the *relative* error is constant over all folds. It is chosen to therefore normalize the MSE of each fold by the variance of the ground truth values in that fold. This leads to the R^2 metric:

$$R^2 = 1 - \frac{\sum_{n=0}^{N-1}(y_n - \hat{y}_n)^2}{\sum_{n=0}^{N-1}(y_n - \bar{y})^2} \tag{6}$$

From another perspective, the R^2 score describes the squared error of an estimator given in the numerator, w.r.t. the squared error of using the sample mean as estimator. Thus, if $R^2 = 0$, the estimator performs equally well as the sample mean estimator. When R^2 is close to one or very negative, it performs much better or worse than the sample mean, respectively.

Figure 9 shows that the first few folds contain only ground truth values of zero. This will cause the R^2 to be undefined due to division by zero. Therefore, the samples are first uniformly random permuted. This is a necessity, but the disadvantage is that the model is not tested on ground truth values that are significantly different in magnitude than trained on. Consequentially, the score is a poor indication of the out-of-sample performance.

Every domain adaptation technique in section 4, is evaluated for all microphone pairs. There are 6 microphones. Thus, the evaluation of every technique results in a 6 by 6 score matrix, where every possible pairwise combination of microphones as source and target, is evaluated. This makes it difficult to compare multiple techniques or parameters. Therefore, the average R^2 score over all folds of all microphone pairs that are suitable for that method, is taken. For unsupervised and supervised domain adaptation, microphones originating from the same and different experiments, respectively, are considered for the average R^2. A paired t-test is then used to compare all the folds resulting from one method, with another. This is visually elaborated in the following tables.

Source \Target	E1, m1	E1, m2	E2, m1	E2, m2	E2, m3	E2, m4
E1, m1						
E1, m2						
E2, m1						
E2, m2						
E2, m3						
E2, m4						

Table 3.: *Microphone pairs that are considered (in green) to compute the average R^2 performance of unsupervised domain adaptation methods. Only pairs from the same experiment are considered. Also note that the self-performance, pairs where the source and target are the same microphone, are excluded. In total 14 microphone pairs, each consisting of 10 folds are considered. Therefore, the average R^2 score is computed over 140 fold scores. Furthermore, the t-test is performed between 140 folds of one method and 140 folds of the other.*

For the semi-supervised domain adaptation methods, only microphone pairs from different experiments (with different posterior distributions) are used to evaluate the average R^2 performance. The following table depicts, which microphone pairs are considered in the 6 by 6 performance matrix:

Source \Target	E1, m1	E1, m2	E2, m1	E2, m2	E2, m3	E2, m4
E1, m1						
E1, m2						
E2, m1						
E2, m2						
E2, m3						
E2, m4						

Table 4.: *Microphone pairs that are considered (in green) to compute the average R^2 performance of semi-supervised domain adaptation methods. Only pairs from different experiments are considered. In total 16 microphone pairs, each consisting of 10 folds are considered. Therefore, the average R^2 score is computed over 160 fold scores. Furthermore, the t-test is performed between 160 folds of one method and 160 folds of the other.*

3.4 CASCADED REGRESSION

Initially, ordinary least squares is used to predict the ground truth from the MFCC data. The resulting predictions are given below:

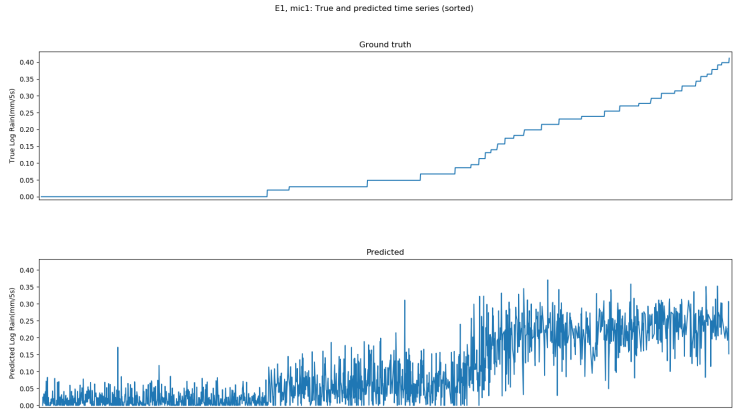

Figure 10.: *Ground truth (top) and predicted values (bottom) of E1, m1 using linear least squares. Ground truth time series is sorted ascending with predicted values sorted accordingly.*

The figures above are sorted ascending by the ground truth. In doing so, it can be seen that when the ground truth is zero (and thus no rain is detected) in the upper subfigures, a least squares model pre-

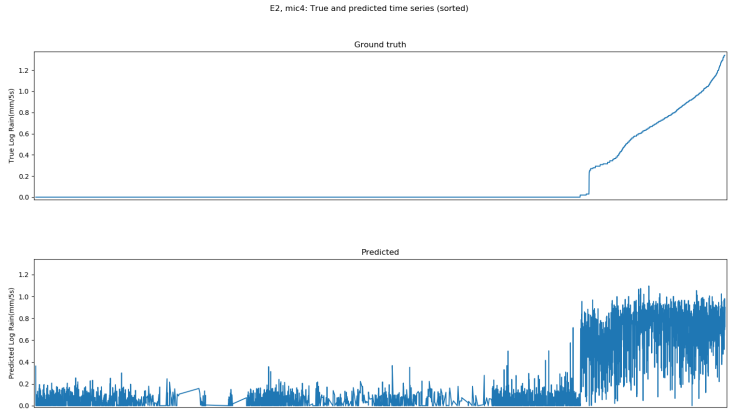

Figure 11.: *Ground truth (top) with predicted values (bottom) of E2, m4 using linear least squares. Ground truth time series is sorted ascending with predicted values sorted accordingly.*

dicts non-zero, as seen in the lower subfigures. To prevent these false positive predictions, a method similar to stacked generalization[10] is used. There, multiple base learners predict the ground truth individually, whereafter their predicted output is used as input of a meta learners. In contrast to this 'stacking', the learners are cascaded in a two-stage fashion. First, a logistic regressor classifies the data into no rain and rain. The use of a classifier in the first stage is motivated by the fact that it can predict a binary value, preventing the many false positive predictions as was seen in figure 10 and 11. Feature vectors that are classified as rain, are then passed on to an ordinary least squares regressor, to predict the actual rainfall. Finally, the no rain predictions of the logistic regressor are concatenated with the rain predictions of the least squares regressor. A schematic of this process is shown below:

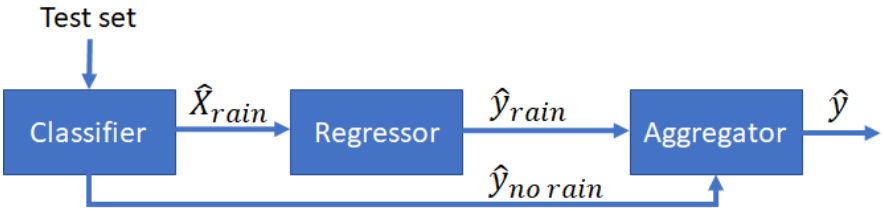

Figure 12.: *Cascaded regression. After the classifier and regressor have been trained by a train set, a test set is fed into the classifier. Feature vectors that have been predicted to be corresponding to rain are passed on to the regressor, which predicts the actual rainfall. The predictions for rain and no rain are then aggregated, resulting in the complete prediction.*

Now, the prediction on E1, m1 and E2, m4 are performed again, this time using cascaded regression.

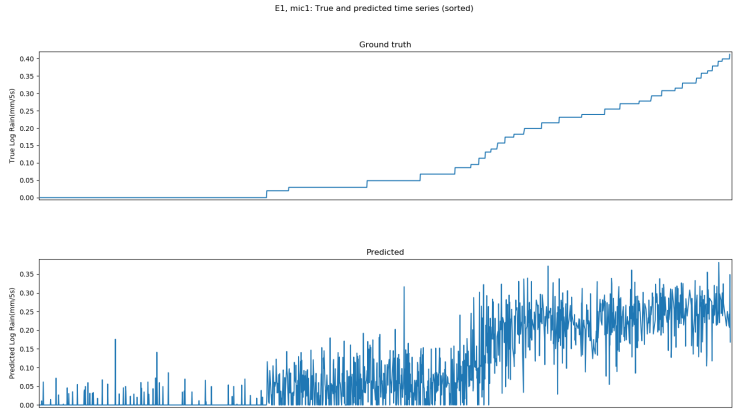

Figure 13.: *Ground truth (top) and predicted values (bottom) of E1, m1 using cascaded regression. Ground truth time series is sorted ascending with predicted values sorted accordingly.*

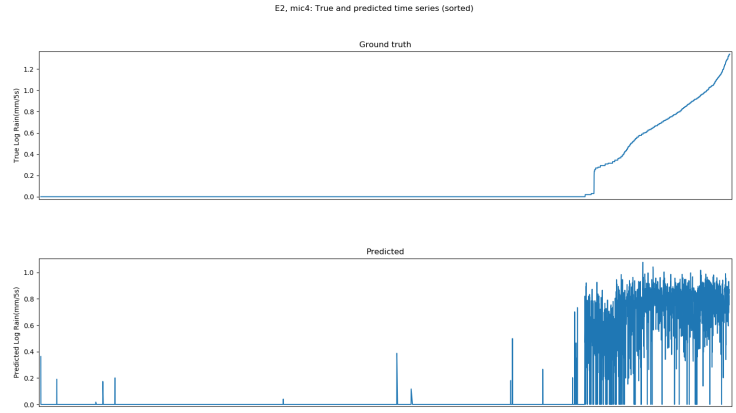

Figure 14.: *Ground truth (top) with predicted values (bottom) of E2, m4 using cascaded regression. Ground truth time series is sorted ascending with predicted values sorted accordingly.*

Comparing least squares and cascaded regression for E1, m1 in figure 10 and 13, it can be seen that the amount of false positives for rainfall activity are reduced, although not much. In contrast, the comparison for E2, m4 in figure 11 and 13, shows that the amount of false positives are significantly reduced. The performance of the classifier stage and the final predictions ($\hat{y}$ in figure 12 with cascaded regression is given in the table below:

Dataset	Classifier Accuracy	Stacked regression R^2 score
E1, m1	0.91 ± 0.02	0.76 ± 0.03
E1, m2	0.87 ± 0.02	0.65 ± 0.05
E2, m1	0.99 ± 0.01	0.68 ± 0.03
E2, m2	0.99 ± 0.01	0.65 ± 0.03
E2, m3	0.97 ± 0.01	0.60 ± 0.03
E2, m4	0.98 ± 0.01	0.67 ± 0.03

Table 5.: *Cascaded regression performance: Accuracy of classifier stage and R^2 score for the final predictions.*

Thus, in a supervised learning setting, the average R^2 score over all microphones is approximately 0.67. This can be used as reference score for the domain adaptation performance.

4

DOMAIN ADAPTATION

4.1 INTRODUCTION

In this chapter, it will be described how the cascaded regressor is used to perform domain adaptation between the six acoustic sensors. Two situations are considered and elaborated in the rest of this introduction.

In the situation of a semi-supervised domain adaptation, the posterior distributions of the source and target domain are considered equal and only the data distributions are matched[11]. Methods in this situation are described in section 4.2. Therefore, only labels in the source domain will be utilized, but no target labels. The difference lies in the joint density function in the following equation:

$$p^S(X, Y) \neq p^t(X, Y) \tag{7}$$

Where $p(\cdot)$ represents the joint density function, X the feature vector and Y the rainfall annotation, for the source or target.

Now, the assumption is made that the posterior distributions $P(Y|X)$ of the source and target domain are equal. In other words, fundamentally, the same relationship between acoustics and rainfall exists. Only the data distributions are different due to the characteristics of the acoustic sensor and the point in time/space the measurements were performed. Therefore, the joint distributions can be split up into the posterior and data distribution. This is described in the following equation:

$$\begin{gathered} p^S(X,Y) = p^S(Y|X)p^S(X) \neq p^t(Y|X)p^t(X) = p^t(X,Y) \\ p^S(Y|X) = p^t(Y|X) \\ p^S(X) \neq p^t(X) \end{gathered} \tag{8}$$

Thus, if the posterior distributions of the source and target are assumed to be equal, the joint density functions can be made more similar, by matching the data distributions.

This situation is illustrated by an example in the figure below:

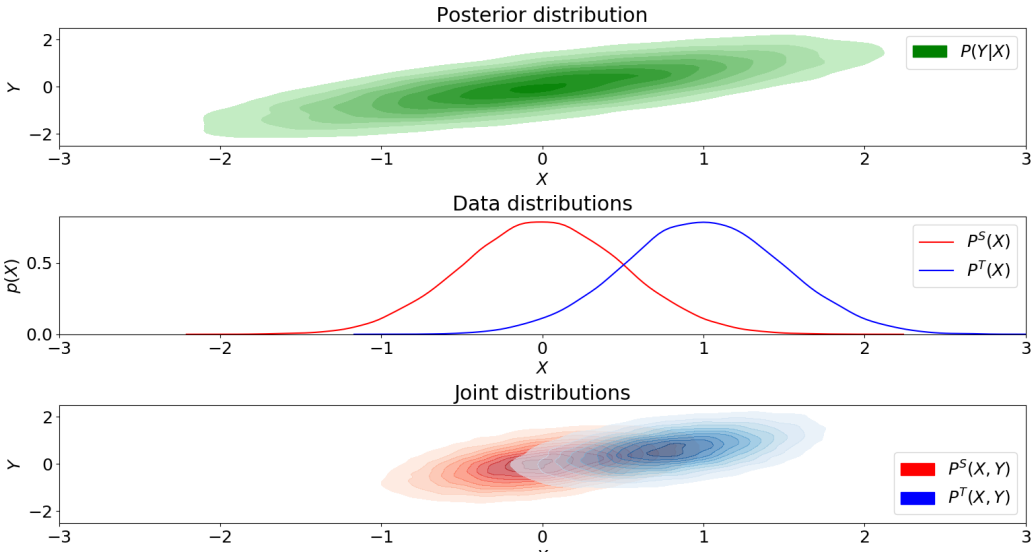

Figure 15.: *Upper: the source and target domain have the same posterior distribution. Middle: their data distributions is different. Lower: As a result, they have different joint distributions.*

- First, in the upper subfigure, an arbitrary posterior distribution is chosen, which is equal for the source and target: $P(Y|X) = p^S(Y|X) = p^t(Y|X) \sim \mathcal{N}(\mu, \Sigma)$, where $\mu = 0$ and $\Sigma = \begin{bmatrix} 1 & \frac{3}{4} \\ \frac{3}{4} & 1 \end{bmatrix}$.
- Then, the middle subfigure shows two different, arbitrary data distributions, $p^S(X) \sim \mathcal{N}(0, \frac{1}{2})$ and $p^t(X) \sim \mathcal{N}(1, \frac{1}{2})$.
- Finally, in the lower subfigure, the source and target joint distributions are computed through the identity given below:

$$P(X, Y) = P(Y|X) \cdot P(X) \tag{9}$$

In the above figure, the difference in data distributions, results in two different joint distributions for the source and target, despite having the same posterior distribution

It is validated whether the assumption that the posterior distribution is equal for all microphones holds. The posterior is obtained from the joint density and data distribution:

$$p(Y|X) = \frac{p(X,Y)}{p(X)} \tag{10}$$

Furthermore, the posterior is estimated by the histogram of the above joint density and data distribution. This can be achieved as follows:

$$h_{Y|X} = \begin{bmatrix} \frac{h_{X,Y}(0,0)}{h_X(0)} & \cdots & \frac{h_{X,Y}(B-1,0)}{h_X(B-1)} \\ \vdots & \ddots & \vdots \\ \frac{h_{X,Y}(0,B-1)}{h_X(0)} & \cdots & \frac{h_{X,Y}(B-1,B-1)}{h_X(B-1)} \end{bmatrix} \tag{11}$$

Where $h(\cdot)$ denotes the histogram value for the B-th bin.

The energies of the (MFCC) spectograms are primarily contained in the first few coefficients. Therefore, the posterior histograms are inspected using the first, second and third coefficients. The latter two can be found in Appendix A and the former is given below:

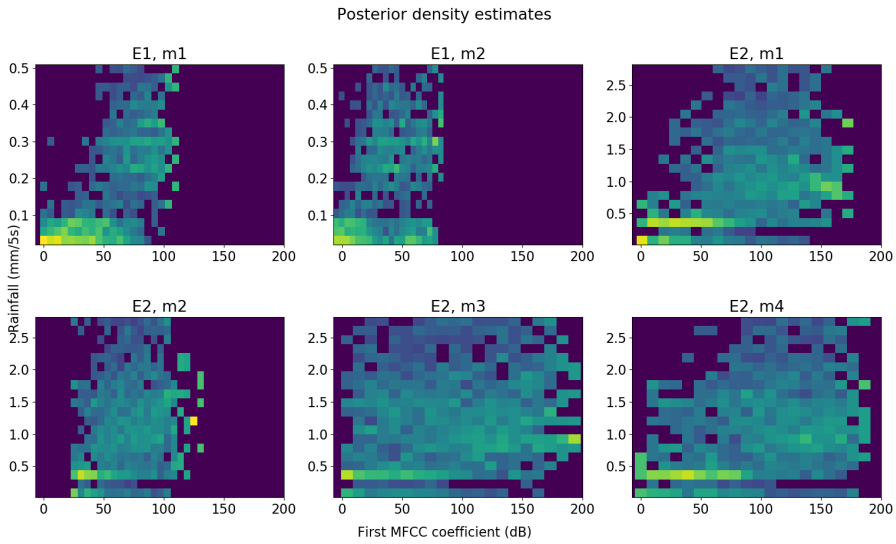

Figure 16.: *Posterior density estimates of the first MFCC feature with the labels.*

Although all posteriors partially overlap, in general the posteriors from E1 are significantly different from E2. This can be caused by a significantly different environment. Acoustic data in E1 and E2 that share the same labels, can then originate from different posteriors when the enviromental noise drastically influences the acoustic measurements. Furthermore, microphones from the same experiment

have reasonably similar posteriors. Therefore, microphones originating from the same experiment are dealt with semi-supervised domain adaptation techniques that assume equality in the posterior and only match the data distributions.

Microphones from different experiments require supervised domain adaptation techniques. In this setting, the microphones have different posterior distributions **and** data distributions. The consequence is that there is no choice, but to utilize both annotations from the source domain as well as some annotations from the target domain. Corresponding methods are described in section 4.3.

4.2 SEMI-SUPERVISED DOMAIN ADAPTATION

The following methods assume that the posterior distributions of all microphones are equal and only the data distributions must be matched.

4.2.1 *Standardization*

Conventionally, when the features vary largely in scale, it can be beneficial to re-scale the data to zero mean and unit variance as in the equation below:

$$\begin{aligned} X'_{n,d} &= \frac{X_{n,d} - \mu_d}{\sigma_d} \\ &= \frac{X_{n,d} - \frac{1}{N}\sum_n X_{n,d}}{\frac{1}{N}\sum_n \left(X_{n,d} - \mu_d\right)^2} \quad \forall d \in \{0, 1, \ldots, D-1\} \end{aligned} \tag{12}$$

where $X_{n,d}, x'_{n,d} \in \mathbb{R}^{N \times D}$ are the original and standardized d-th feature, respectively. μ_d and σ_d are the mean and variance of the d-th feature. D is the number of features. Standardization can be specifically beneficial for domain adaptation, because the source and target distributions will become more similar, due to the mean and variance being matched. From the perspective of MFCCs, the benefit of subtracting the mean from the data is that the different channel effects(transfer function of the acoustic sensor and environment) of every microphone is neutralized. This is known as cepstral mean subtraction and can be proved by considering the data in the frequency domain, where it was noted in section 3.2 that a log transform was applied:

$$\begin{aligned} log\left[X(f)\right] &= log\left[S(f) \cdot H(f)\right] \\ &= X^{log}(f) = S^{log}(f) + H^{log}(f) \end{aligned} \tag{13}$$

It can be seen that due to the log transform, the channel effects become an additive term. Then, the average over all measurements is taken:

$$\frac{1}{N}\sum_n X^{log}(f) = \frac{1}{N}\sum_n S^{log}(f) + H^{log}(f) \tag{14}$$

Finally, subtracting equation 14 from 13, it can be seen that subtracting the average from the data, similar to equation 12, eliminates channel effects:

$$X^{log}(f) - \frac{1}{N}\sum_{n} X^{log}(f) = S^{log}(f) - \frac{1}{N}\sum_{n} S^{log}(f) \tag{15}$$

The results of applying standardization compared to the baseline performance is given in Appendix B. It can be seen that indeed, standardization significantly improves the prediction performance, for most microphone pairs. It is noticable that the performance is low when the model is trained on a source microphone originating from a different experiment than the target microphone it is tested on. Therefore, standardization is only beneficial if the microphone datasets used, originate from the same experiment.

4.2.2 *Principal Component Analysis*

Principal Component Analysis (PCA)[11], can be beneficial to find a (sub)space where the soure and target data are more similar. PCA exploits the statistical information that lies in the data covariance matrix, where its eigenvectors and eigenvalues, describe the directions and amplitudes of the data variance in the feature space. The first benefit w.r.t. domain adaptation, is that the transformed data is centered with mean zero, which matches the mean of the source and target. Furthermore, PCA is invariant to rotation. Suppose the target data is simply a rotated version of the source in the feature space. Then, the direction of the eigenvectors rotate, adaptively. As a result, the transformed domain is not affected by rotations in the original domain.

To illustrate the above mentioned properties, consider a binary classification problem. First, both classes are drawn from an arbitrary normal distribution $\mathcal{N}(\mu, \Sigma)$. Then, a rotation matrix is applied, given in the equation below:

$$R = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \tag{16}$$

The source and target data is then formed by choosing $\theta = -20$ and $\theta = 30$ degrees, respectively. This results in two distributions, that are rotated versions of each other, but with a different mean. A PCA is applied on both source and target, individually. Finally, a logistic regression is trained on the source data and applied to the target data. This is done in the original domain and in the PCA domain:

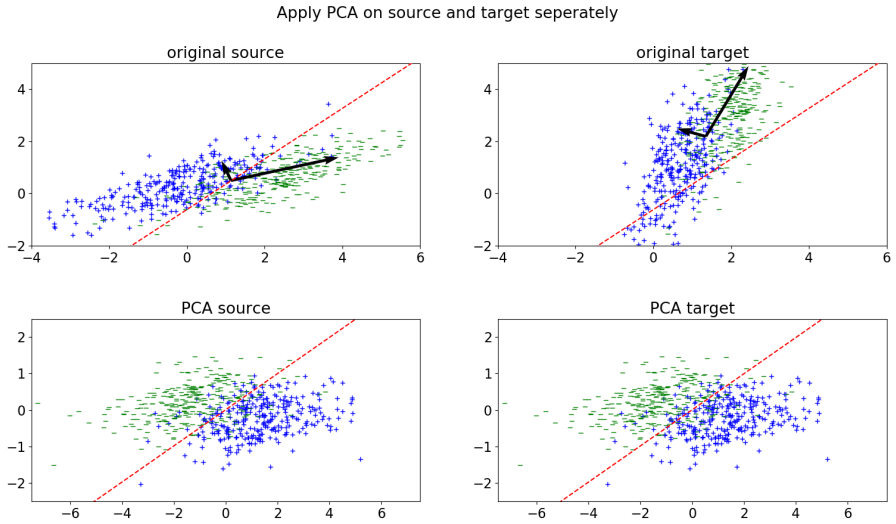

Figure 17.: *Illustration of insensitivity of PCA to mean and orientation of a distribution. Upper left and right: An arbitrary two-class Gaussian distribution is rotated $\theta = -20$ and $\theta = 30$, to construct the source and target data. A logistic regression is trained on the source data and applied on the target. The decision boundary is given in dashed red. The eigenvectors used to project the data, are given in black. Lower left and right: A PCA is applied on the source and target data, individually. Again, a logistic regression is trained on PCA transformed source data and applied on the PCA target data.*

In the figure above, it can be seen that the logistic regression trained on the original source data (upper left), does not perform well on the original target (upper right), because the latter is rotated and its mean is different. After applying PCA on both the source (lower left) and target (lower right), it can be seen that the logistic regression trained on the PCA transformed source, transfers very well on the PCA transformed target, because the distributions are more similar/ The means are zero in both cases and the effect of different orientations is removed. However, it should be noted that, in general, the covariance matrix of the source and target should be similar. Otherwise, their PCA transform will be different.

Now that it is shown that PCA can be beneficial for domain adaptation, it is applied on the rainfall datasets. The comeplete performance matrix is given in appendix B. As with standardization, microphone pairs originating from different experiments have low performance. Pairs originating from the same experiment do not consistently perform better with PCA, than the baseline. In section 5, the average R^2 will be used to gain more insight in the performance.

4.2.3 *Transfer Component Analysis*

Transfer Component Analysis (TCA)[12][11] is a domain adaptation technique that minimizes the Maximimum Mean Discrepancy (MMD) cost function. A complete derivation and discussion is given in Appendix C. Furthermore, an implementation can be found in Appendix F In this chapter, they keypoints for TCA are given and the technique is applied to the acoustic rainfall dataset.

The MMD describes the Euclidean distance between statistical descriptions of the source and target distribution. One such description is the sample mean, which is the sum of all feature vectors normalized by the sample size. In matrix form, this can be described with the kernel matrix and normalization. The kernel matrix K holds all information on the feature vectors from the source and target. The normalization matrix L contains information on the sample size of the source and target. It should be noted that a transformation $\phi(\cdot)$ can be applied before the sample mean operation. This transforms the feature vectors into a higher dimensional space, where not only the difference between the mean is considered, but also other statistical relations.

A weight matrix is added to the MMD, so that it can control the distance between the source and target mean. It can be shown, that the solution for the weight matrix consists of the eigenvectors corresponding to the m leading eigenvalues of equation 17, where m is an arbitrary desired dimensionality of the transformed TCA space.

$$W = eig\left(\frac{KHK}{I + \mu KLK}\right) \tag{17}$$

Finally, after finding the optimal weight matrix, it can be used to project the original source and target data in the TCA space:

$$X_{TCA} = K\hat{W} \tag{18}$$

Different kernel functions exist to compute the kernel matrix K. The linear kernel and Radial Basis Function (RBF) kernel are considered. It is known that a RBF kernel transforms the data, using the mapping $\phi(\cdot)$, such that not only the mean, but also higher moments are matched:

$$K(x_i, x_j) = e^{-\gamma||x_i - x_j||^2} \tag{19}$$

The power series expansion then shows that the kernel and thus the MMD incorporates exponentiations (all higher moments) of the dot product between two feature vectors in the original domain:

$$K(x_i, x_j) = e^{-\gamma||x_i||^2 - \gamma||x_j||^2} \sum_{n=0}^{\infty} \frac{(2\gamma)^n (x_i \cdot x_j)^n}{n!} \tag{20}$$

It can be seen that the RBF kernel is a Gaussian similarity function, where γ is inversely related to the standard deviation and thus the bandwidth of the Gaussian function. A value of 1 corresponds to x_i and x_j to be very close/similar in the feature space and a value of 0 to them being very far apart/dissimilar. A large γ, results the function to appoint a high similarity score only when x_i and x_j are very close. Conversely, a small γ results the function to appoint a high similarity, even when x_i and x_j are far apart.

A fitting γ parameter is found by inspecting the N^{th} nearest neighbour (NN) of all samples for a specific microphone. Then, the average of all these Euclidean distances is computed and is used as γ value as described in the equation below:

$$\bar{D}_m = \frac{1}{N} \sum_i D_{m,i} \tag{21}$$

Where m denotes the index of a microphone, and i is the sample index. $D_{m,i}$ therefore denotes the Euclidean distance to the Nth nearest neighbour for sample i of microphone m.

Then, it is computed what the average distance is over all microphone as defined below:

$$\bar{D} = \frac{1}{N} \sum_m \bar{D}_m \tag{22}$$

The table below shows the average distance as defined in the above equation, for the 5^{th} and 10^{th} nearest neighbour:

	$5^{th}NN$	$10^{th}NN$
$\bar{D}$	14.60 (5.17)	15.39 (5.23)

Table 6.: *Average Euclidean distance of the* 5^{th} *and* 10*th NN, over all samples and all microphones. Standard deviation is denoted by* $(\cdot)$.

It can be seen that the mean of the average distance is similar for the 5^{th} and 10^{th} NN. Therefore, it has been chosen to use a value that is approximately halfway through: $\gamma = 15$.

Before assessing the performance of TCA with the cascaded regressor, it is investigated how well the linear and RBF kernel perform in matching different moments. To do so, a score metric is devised to quantify the distance between the k^{th} moment of the source and target data. This moment distance score is given below:

$$S_k = \frac{||m^{Sk} - m^{tk}||_2}{\sigma_S^2 + \sigma_T^2} \tag{23}$$

and the k^{th} moment is defined below:

$$m^k = \begin{cases} \frac{1}{N}\sum_{i=1}^{N}(X_{i,d} - \bar{X}_d)^k, & k \geq 2 \\ \frac{1}{N}\sum_{i=1}^{N} X_{i,d}\ , & k = 0 \end{cases} \quad \forall d \tag{24}$$

Therefore, the second moment is equivalent to the variance and the 0^{th} moment is equivalent to the sample mean. Note that m^1 should be zero. This is then computed for every feature, thus $m^k \in \mathbb{R}^{1\times d}$. The variances in the denominator of equation 23, consider the entire feature matrix, i.e. all values of all features.

This score is applied to the original domain as well as in the TCA domain. Then, the ratio between the two is considered to quantify the domain matching between the source and target. The ratio is given below:

$$S^k_{ratio} = \frac{S_{TCA}}{S_{original}} \tag{25}$$

where both for the moment distance score in equation 23, as the ratio score in equation 25, a lower value is better.

The domain adaption performance of TCA, is assessed for a linear and RBF kernel for different moments, with the found γ parameter in table 6. This is done for two cases. In case one E1, m1 is the source data with E1, m2 as target data. In case two E1, m1 is again the source data, but E2, m1 the target data. The original data, TCA with a linear kernel, RBF with $\gamma = 15$ and RBF with $\gamma = 1$ are evaluated. The latter is included, to verify that the method to find a suitable γ parameter discussed earlier, performs better, than a 'default' value. The data matching is visualized, by inspecting the first two MFCC components in the original domain and the first two TCA components in the TCA domain. The figures are given below. Furthermore, the tables where different moments are evaluated using the ratio score are also given below.

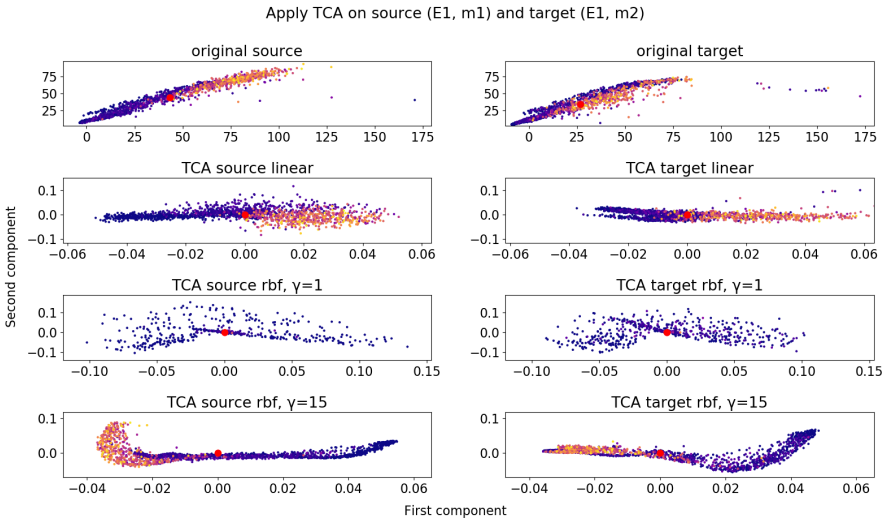

Figure 18.: *Visualization of the first two components of E1, m1 as source and E1, m2 as target. This is done for the first two MFCC components in the original domain and first two TCA components in the TCA domain using a linear and two RBF kernels. Red dot shows location of the sample mean. Color indicates rainfall intensity(darker is less rain)*

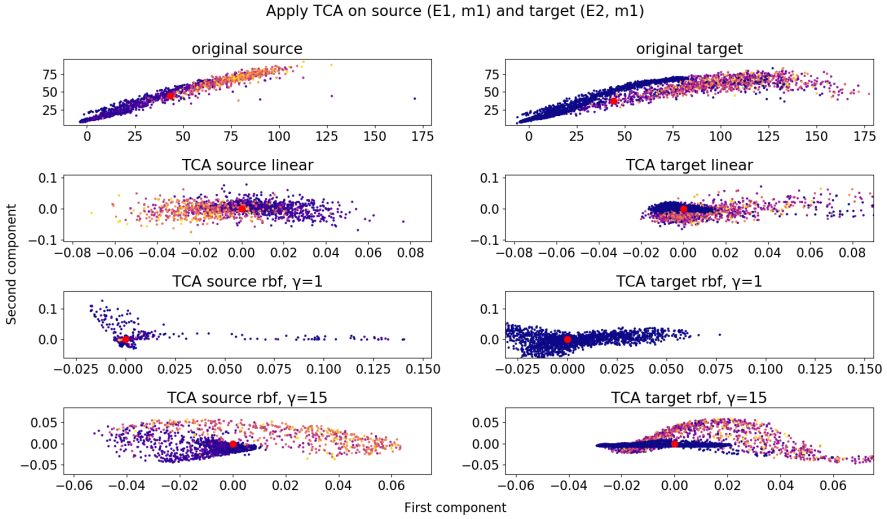

Figure 19.: *Visualization of the first two components of E1, m1 as source and E2, m2 as target. This is done for the first two MFCC components in the original domain and first two TCA components in the TCA domain using a linear and two RBF kernels. Red dot shows location of the sample mean. Color indicates rainfall intensity (darker is less rain)*

The corresponding tables are given below:

E1, m1(source)	E1, m2(target)	k=0	k=2	k=3	k=4
Linear kernel, $\mu = 1$		$2 \cdot 10^{-4}$	$5 \cdot 10^{-3}$	$5 \cdot 10^{-6}$	$1 \cdot 10^{-8}$
RBF kernel, $\mu = 1, \gamma = 1$		318	2	$2 \cdot 10^{-3}$	$7 \cdot 10^{-6}$
RBF kernel, $\mu = 1, \gamma = 15$		0.3	$2 \cdot 10^{-4}$	$3 \cdot 10^{-7}$	$9 \cdot 10^{-10}$

Table 7.: *Evaluation of the data distribution matching of E1, m1 as source and E1, m2 as target. This is done for a linear and RBF kernel. Different moments are evaluated using the score functions in equation 23 and 24.*

E1, m1(source)	E2, m1(target)	k=0	k=2	k=3	k=4
Linear kernel, $\mu = 1$		$4 \cdot 10^{-4}$	$3 \cdot 10^{-3}$	$2 \cdot 10^{-6}$	$1 \cdot 10^{-9}$
RBF kernel, $\mu = 1, \gamma = 1$		612	0.5	$3 \cdot 10^{-3}$	$3 \cdot 10^{-7}$
RBF kernel, $\mu = 1, \gamma = 15$		0.3	$7 \cdot 10^{-5}$	$2 \cdot 10^{-8}$	$2 \cdot 10^{-11}$

Table 8.: *Evaluation of the data distribution matching of E1, m1 as source and E2, m2 as target. This is done for a linear and RBF kernel. Different moments are evaluated using the score functions in equation 23 and 24.*

The above tables show that all kernels, except for an RBF with $\gamma = 1$, improve the matching of the analyzed moments. The RBF with $\gamma = 15$ underperforms in matching the mean compared to a linear kernel, but is better at matching higher moments. Even though the tables imply that the matching works well for both cases inn figure 18, TCA seems to match the shape of the source and target distribution less well, than in figure 18. It is concluded that a linear kernel and RBF kernel with $\gamma = 15$ are candidates to be used with the cascaded regressor. It is further investigated, which of the two performs better.

It was mentioned that in TCA, the number of components m determine the dimensionality of the transformed data space. It is investigated how many leading components are optimal. A linear kernel is used with trade-off parameter $\mu = 1$. To compare the performance matrices, the average R^2 performance over all folds of all relevant microphone pairs is taken. This is then compared for a range of m. The results are found in the figure below:

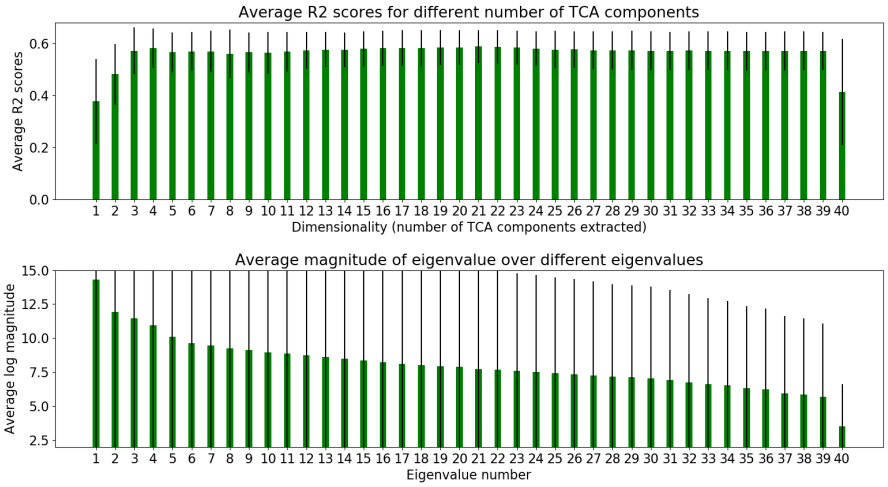

Figure 20.: *Upper: Average R^2 score for different amounts of components extracted. Lower: Corresponding log magnitude of the eigenvalues retrieved from TCA with linear kernel. $m = 21$ components is optimal according to paired t-test*

In the upper figure, it can be seen that the performance of TCA increases drastically with every extra component added. Applying a paired t-test between the fold scores of the extracted components, it was found that $m = 21$ components perform significantly better, than all other values of m. Furthermore, in the lower figure it can be seen that the average magnitude of the eigenvalues decreases exponentially. Comparing this with the fact that the performance drastically increases, with only a few components added, it is concluded that the magnitude of these eigenvalues, are a good indicator for the amount of information a component contains.

Similarly the same experiment is performed for TCA with an RBF kernel:

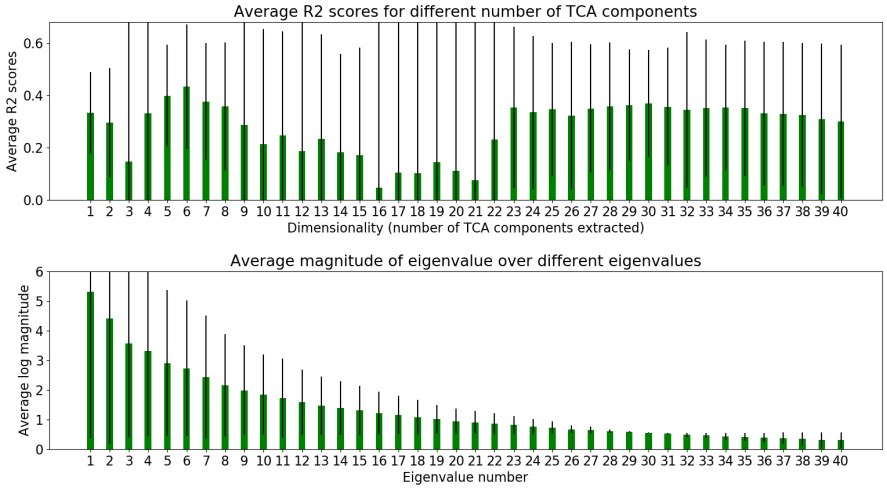

Figure 21.: *Upper: Average R^2 score for different amounts of components extracted. Lower: Corresponding log magnitude of the eigenvalues retrieved from TCA with RBF kernel. $m = 6$ components is optimal according to paired t-test*

From the above two figures, it is concluded that TCA with a linear kernel performs best. The linear kernel is better at matching the mean and the RBF kernel better at matching higher moments. This can imply that matching the mean is more important, than the higher moments. The complete performance matrices can be found in Appendix B.

4.3 SUPERVISED DOMAIN ADAPTATION

In the previous section, the methods assumed similar posterior distributions. Therefore, the joint distributions were implicitly matched by matching the data distributions. Therefore, no information on the labels of the target samples were used. It was found that the methods were only effective for datasets of microphone pairs that originated from the same experiment. For domain adaptation of microphone pairs from different experiments, both the posteriors and data distributions are assumed to be different. Therefore, the joint distributions of the source, $p^S(X^S, y^S)$ and target, $p^t(X^t, y^t)$ must be matched explicitly. To achieve this, supervised domain adaptation is needed where *some* target labels are acquired to match the joint distributions.

An approximation of the joint distribution is explained in terms of the source and target data distributions, $p^S(X^S)$ and $p^t(X^t)$ and the source and target prior distributions, $p^S(y^S)$ and $p^t(y^t)$. Then, it will be shown that TCA cannot be used to perform the prior matching. Furthermore, a mean and variance minimization method are shown that reweight the source labels, such that the mean and variance of the source and target are more similar. For each method, a situation is considered where the available target labels are used in batch and in an online fashion.

The high-dimensional space and relatively small sample size result in sparse estimations of the joint distributions. Therefore, the matching is approximated by matching the transformed data distributions and prior distributions separately. In terms of loss functions, this can be expressed as:

$$\begin{aligned}\mathcal{L}\left(p^S(X^S, y^S), p^t(X^t, y^t)\right) \approx & \mathcal{L}_X\left(p^S\left(\psi(X^S)\right), p^t\left(\psi(X^t)\right)\right) \\ & + \mathcal{L}_y\left(p^S\left(\Omega(y^S)\right), p^t\left(\Omega(y^t)\right)\right)\end{aligned} \tag{26}$$

where $\psi(\cdot)$ and $\Omega(\cdot)$ transform the feature vectors and labels, respectively, such that the data and prior distributions are matched. It was shown that TCA was used to find the transformation ψ, which used the MMD as loss function $\mathcal{L}_X$. Optimality is achieved when the losses are zero. In turn, these loss functions themselves are also approximations to describe the similarity between two distributions and optimality in these loss functions does not guarantee a perfect matching

of the distributions[13]. However, the assumption will be made that this does hold:

$$\begin{aligned}\mathcal{L}_X\left(p^S\left(\psi(X^S)\right), p^t\left(\psi(X^t)\right)\right) = 0 \rightarrow p^S(\psi(X^S)) = p^t(\psi(X^t))\\ \mathcal{L}_y\left(p^S\left(\Omega(y^S)\right), p^t\left(\Omega(y^t)\right)\right) = 0 \rightarrow p^S(\Omega(y^S)) = p^t(y^t)\end{aligned} \tag{27}$$

The posteriors of the transformed distributions are assumed equal:

$$p^S(\Omega(y^S)|\phi(X^S) = p^t(\Omega(y^t)|\phi(X^t) \tag{28}$$

however, this is not guaranteed if the transformed distributions are not perfectly matched. Suppose that the data distributions are more similar after applying a transformation, but not equal. Furthermore, the prior distributions are perfectly matched and thus its corresponding loss is zero. The posteriors are not matched in this case, because two different data distributions correspond to the same prior distribution. It would then be more beneficial to transform the priors such that their loss is unequal to zero, in order to match the posteriors and thus, have more similar joint distributions. This is contradicting with the approximate loss function in equation 26 where the transformations ψ and Ω *independently* minimize their corresponding losses.

Note that in the previous equations, methods are considered where the target prior distribution is not transformed. The reason will be elaborated by considering TCA for the prior matching. Only a small fraction of target labels is available and therefore, this matching has to be generalized to the unknown target labels. Consider the TCA transform of all N source labels, y^S_{TCA} and a subset of P target labels, y^p_{TCA}, resulting in $L = N + P$ labeled samples. A TCA transform is performed on L labels followed by a prediction of all M target labels, $\hat{y}^T_{TCA}$. It is not clear how an inverse TCA transform on all $N + M$ labels can be performed to retrieve the unknown target labels in the *original* domain. First, the available source and target labels are TCA transformed:

$$y_{TCA} = \begin{bmatrix} y^S_{TCA} \\ y^p_{TCA} \end{bmatrix} = K_l W \tag{29}$$

where $y_{TCA}, W \in \mathbb{R}^{L\times 1}$, $y^S_{TCA} \in \mathbb{R}^{N\times 1}$, $y^p_{TCA} \in \mathbb{R}^{P\times 1}$ and $K_l \in \mathbb{R}^{L\times L}$.

Then, the cascaded regressor is used to predict the target labels:

$$\hat{y}^t_{TCA} = f(X^S, X^t, y^S_{TCA}) \tag{30}$$

where $\hat{y}^t_{TCA} \in \mathbb{R}^{M\times 1}$, with M the total number of target labels. The predicted target labels are represented in the TCA domain and it is

not clear how to project these back to the original domain. The P target labels could be replaced by the M predicted ones. The projection matrix W was constructed based on the original labels and replacing them by predicted values might not be valid for this matrix. Nonetheless, an attempt to do so is described as:

$$\hat{y}_{TCA} = \begin{bmatrix} y^S_{TCA} \\ \hat{y}^t_{TCA} \end{bmatrix} = KW \tag{31}$$

where $\hat{y}_{TCA} \in \mathbb{R}^{(N+M)\times 1}$ and $K \in \mathbb{R}^{(N+M)\times(N+M)}$ The above equation is however invalid, because the shape of the projection matrix W is incompatible with K and $\hat{y}_{TCA}$. Therefore TCA is not considered for the prior matching.

Instead, a mean matching and variance minimization method is considered, that match the mean and variance of the source to that of the target, by weighting the source labels. The means and variances of the source and target labels are represented as:

$$m(y^S) = m^S = \frac{1}{N_s}\sum_i y^S(i) \tag{32}$$

$$m(y^t) = m^t = \frac{1}{N_t}\sum_i y^t(i) \tag{33}$$

$$s(y^S) = s^S = \frac{1}{N_s}\sum_i (y^S(i) - m^S)^2 \tag{34}$$

$$s(y^t) = s^t = \frac{1}{N_t}\sum_i (y^t(i) - m^t)^2 \tag{35}$$

Applying a weight to transform the source mean and variance can then be represented as:

$$m(wy^S) = wm^S \tag{36}$$

$$s(wy^S) = w^2 s^S \tag{37}$$

For the mean matching the weight w must satisfy:

$$\begin{aligned} m(wy^S) &= m(y^t) \\ wm^S &= m^t \end{aligned} \tag{38}$$

Thus, the optimal w is:

$$w = \frac{m^t}{m^S} \tag{39}$$

Furthermore, a variance minimization method is devised that minimizes the variance difference of the weighted source labels to that of the target labels, while constraining the mean difference. The minimization problem is given below.

$$\begin{aligned} &min_w(w^2 s^S - s^t)^2 \\ &s.t. \left(\frac{w m^S - m^t}{m^t}\right)^2 \leq (\alpha)^2 \end{aligned} \tag{40}$$

where α sets an upper bound on the maximum allowed difference in the means. Therefore, this method trades off the mean and variance minimization, based on a chosen α. The minimization results in five solutions of which two are dependent on α:

$$w_1 = \frac{m^t}{m^S}(1 + \alpha) \tag{41}$$

$$w_2 = \frac{m^t}{m^S}(1 - \alpha) \tag{42}$$

where the first multiplicative term scales the source labels such that its mean matches that of the target labels. The second term involving α then adds (w_1) or removes (w_2) a margin to trade off the mean matching for a better variance matching. Depending on the influence of the first term on the variance, the second term can be used to either increase

The derivation and solution can be found in Appendix E.

The matching performance of the two methods is inspected on the rainfall labels, by computing the relative difference between the source and target variance w.r.t. the variance of the target labels. This is done before the reweighting is applied in equation 43 and after in equation 44:

$$\Delta s_{original} = \frac{s^S - s^t}{s^t} \tag{43}$$

$$\Delta s_{new} = \frac{w^2 s^S - s^t}{s^t} \tag{44}$$

Furthermore, the mean matching is evaluated as:

$$\Delta m_{original} = m^S - m^t \tag{45}$$

$$\Delta m_{new} = w m^S - m^t \tag{46}$$

The prior distribution of microphones from the same experiment are almost identical. Therefore, instead of reporting the pairwise matching performance for all microphones, only the performance between experiments is reported. The performance of the mean matching method is given below:

source/target	E1	E2
E1		0.98, 0.47
E2	42, 0.89	

Table 9.: *Variance difference performance for the mean matching method. Score between experiments is formatted as:*$\Delta s_{original}, \Delta s_{new}$.

source/target	E1	E2
E1		-0.48, 0
E2	0.48, 0	

Table 10.: *Mean difference for the mean matching method. Score between experiments is formatted as:*$\Delta m_{original}, \Delta m_{new}$.

It can be seen that the mean matching indeed focuses on strictly matching the means, but this implicitly also matches the variances. The performance of the variance minimization method is given below:

source/target	E1	E2
E1		0.98, 0.24
E2	42, 0.25	

Table 11.: *Variance difference for the mean matching method. Score between experiments is formatted as:*$\Delta s_{original}, \Delta s_{new}$.

source/target	E1	E2
E1		-0.48, 0.12
E2	0.48,-0.2	

Table 12.: *Mean difference for the variance minimization method. Score between experiments is formatted as:*$\Delta m_{original}, \Delta m_{new}$.

It can be seen in table 12 that the variance minimization allows for some difference in the means, in favour of matching the variances, which it does better than the mean matching approach.

Both prior matching methods can be used in a batched an online fashion. In a batched approach, many target labels are gathered, before the target mean and variance are re-estimated and a matching method is applied. In contrast, an online approach, immediately updates the target mean and variance, every time a new target label is available. The recursive estimation of the target mean and variance can be implemented as smoothing filters:

$$m^t_{new} = \alpha m^t_{old} + (1 - \alpha) y^t(i) \tag{47}$$

$$s^t_{new} = \alpha s^t old + (1 - \alpha)(y^t(i) - m^t_{new})^2 \tag{48}$$

Note that for the solution of the mean matching, only the target mean is used and for the variance minimization, both mean and variance must be known. α is a trade-off parameter between the old estimate and a newly available target label $y^t(i)$. This has arbitrarily chosen to be 0.9. At the beginning of the recursion, the mean and variance are initialized by the mean and variance of the source labels. Its purpose is to reduce the variance of subsequent estimations (i.e. to smooth). Furthermore it prevents the estimation to converge for many samples. This would be problematic if the target mean and variance signifi-

cantly change, since the online estimators cannot adaptively follow the change, accordingly.

The average R^2 performance of the prior matching methods using a batched and online estimation is inspected for different percentages of target labels used. The chosen target labels are uniformly selected. As mentioned in section 3, only the performance of microphone pairs that originate from different experiments are considered. Two approaches are considered. The best found data distribution matching, TCA with linear kernel is used with the best prior matching, the variance minimization. Also, linear TCA is used with the mean matching method. The result are shown below:

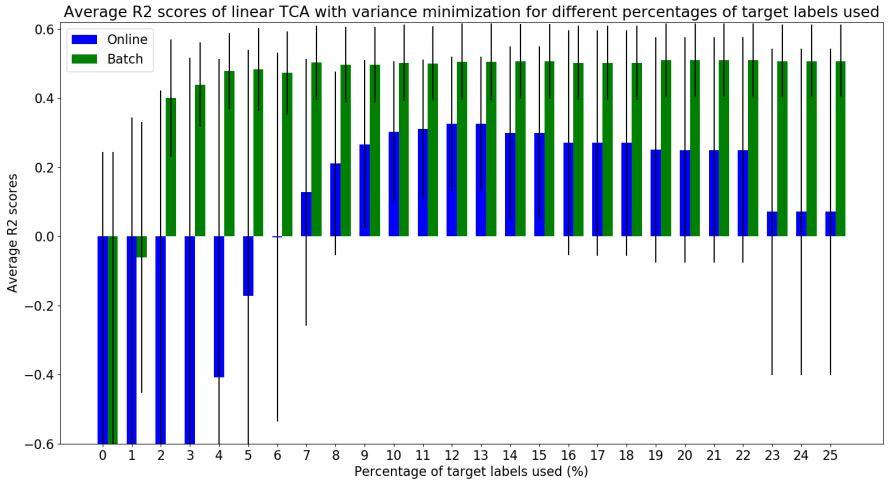

Figure 22.: *Average R2 scores of an online and batched approach, using TCA with linear kernel for data distribution matching and the variance minimization method for prior matching, using* $\alpha = 0.2$. *o% denotes no target labels utilized. The corresponding mean sits at approximately -1, but is omitted from the figure to keep the other scores more readable. Black lines indicate standard deviation over all folds of all used microphone pairs.*

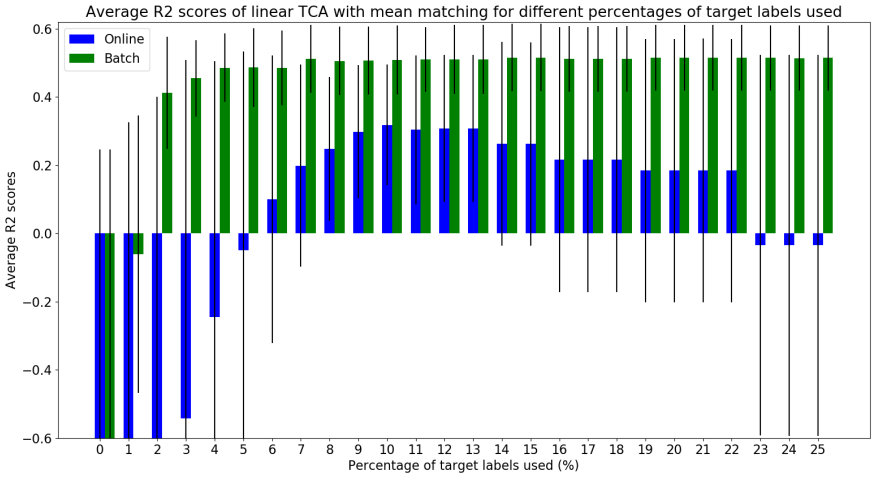

Figure 23.: *Average R2 scores of an online and batched approach, using TCA with linear kernel for data distribution matching and the mean matching method for prior matching. 0% denotes no target labels utilized. The corresponding mean sits at -1, but is omitted from the figure to keep the other scores more readable. Black lines indicate standard deviation over all folds of all used microphone pairs.*

Although the variance minimization method was shown to match the variance of the prior distributions better than mean matching, figure , the performance is similar. This can be seen from figure 22 and 23. A paired t-test shows that the variance minimization does not perform significantly better than mean matching. Therefore, linear TCA for data matching and the mean matching method for prior matching is concluded to perform better. A paired t-test is used to compare the fold scores between different percentages of target labels used. This is done for the batched and online approach. The results are given below:

	Average R^2	Percentage target labels used (%)
Online	0.32 (0.241)	10
Batch	0.52 (0.01)	14

Table 13.: *Best average R^2 of figure 23, according to a paired t-test.*

5

RESULTS

The average R^2 performance for the semi-supervised domain adaptation methods is given in the following figure:

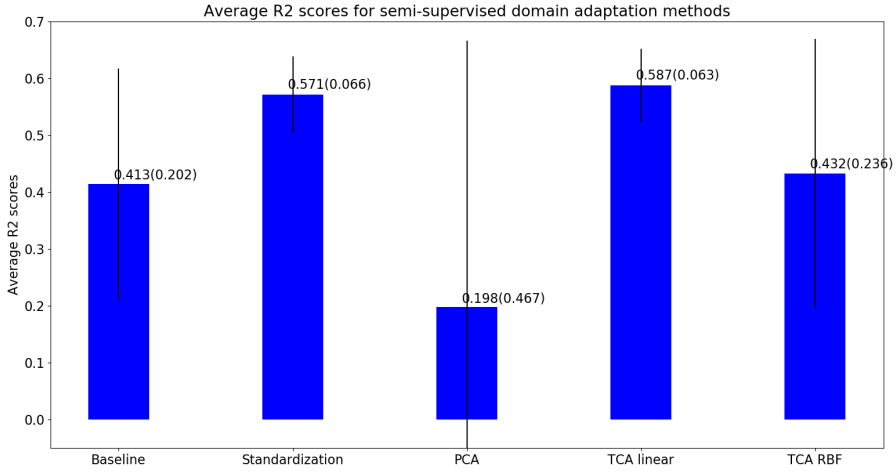

Figure 24.: *Average R2 scores for the supervised domain adaptation methods. Only performances of microphone pairs from different experiments are considered.Black line denotes standard deviation.*

A paired t-test shows that TCA with linear kernel performs better than other methods.

The average R^2 performance for the supervised domain adaptation methods is given in the following figure:

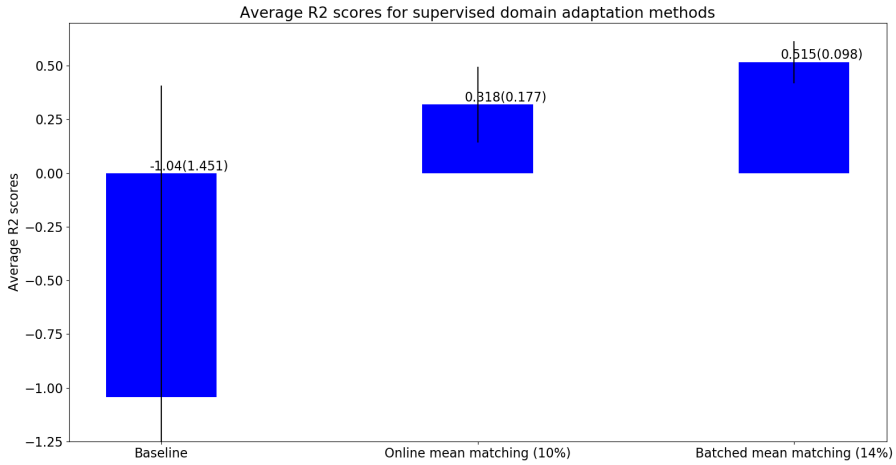

Figure 25.: *Average R2 scores for the supervised domain adaptation methods. Only performances of microphone pairs from different experiments are considered.Black line denotes standard deviation*

It was shown in the previous chapter and summarized in the figure above, that a batched mean estimation approach with 14% of the target labels used, was the best performer. An online approach with 10% target labels used can provide real-time predictions, but the trade off is a significantly worse performance.

6

CONCLUSION & DISCUSSION

The criteria for rainfall estimation consists of temporal/spatial resolution, range, accuracy and cost. An acoustic approach has potential to achieve a better trade-off of these criteria, compared to the state of the art. A machine learning model to estimate rainfall from acoustic sensor data has been proposed. Furthermore, it has been investigated how to transfer the model trained on one acoustic sensor, called the source domain, to a characteristically different one, called the target domain.

A two-stage regression model has been devised to estimate rainfall from acoustics. In the first stage, data is classified as rain presence or absence. This reduced false positive predictions of the model; the prediction of rain presence, when there is no rain. Data corresponding to rain presence is then processed by the second stage, where an estimate of the rainfall intensity is predicted.

Despite the sparsity of the acquired acoustic data (approximately 90 millisecond recordings every 5 seconds), when training and testing was performed on the same sensor, in a supervised learning manner, this resulted in an average R^2 scores of 0.65.

Using domain adaptation techniques, t was explained that transferring the model gained from a source domain to a target domain boiled down to matching the joint distributions of the two. The joint distribution can be decomposed in terms of the posterior and data distribution.

Data acquisition was done through two rainfall emulation experiment over two days. Two microphones were used in the first experiment and four microphones in the second. It turned out that sensors originating from the same experiment had similar posteriors and thus semi-supervised domain adaptation techniques that only match the data distributions without knowledge on the target labels were used for this case. The methods that were investigated were standardization, PCA and TCA. The latter had the best performance.

TCA transforms the data onto a subspace, where the source and target data distributions are more similar. A kernel function is required. It was shown that theoretically, an RBF kernel is more advantageous than a linear kernel. This is because, a linear kernel only matches the mean of the soure and target, while an RBF kernel also matches higher moments of the data. Comparing the two, it was shown that TCA with an RBF kernel matched the higher moments of the acoustic data better, but the mean much worse. The RBF kernel also performed worse with the two-stage regressor. It was therefore hypothesized that, for the acoustic data, matching the mean is more important than the higher moments and that this was the reason the linear kernel performed better. Thus, for sensors from the same experiment, linear TCA performed best, with an R^2 of 0.58.

For sensors from different experiments, the posteriors were assumed to be different. This required supervised domain adaptation techniques. The consequence was that the joint distributions had to be be explicitly matched and therefore, (a small selection of) target labels were needed. This was approximated by matching the data and prior distributions, seperately. It was found that this method is indeed beneficial in matching the joint distributions, but is prone to overfitting.

The data distribution matching was the same as for the semi-supervised approach and the priors were matched with two methods. The mean matching method transforms the source prior distribution to have the same mean as the distribution of the acquired target labels. Similarly, the variance minimization method minimizes the difference between the variances, while constraining the difference in the mean. This did not perform better than mean matching.

For both methods, two situations were considered. One where a batch of target labels were available before performing the matching and prediction. In the other, the target labels were exposed one by one, requiring an online learning approach. Every time a new target label was available, the matching and prediction model was updated and applied to unlabeled data, until the next target label was available. Mean matching was chosen as the preferred method and the batched and online approach resulted in an average R^2 of 0.45 and 0.30, respectively.

An approximation that matches the data and prior distributions independently was used. A method could be devised that explicitly matches the source and target joint distributions.

The prediction model combined benefits of a classifier and regressor. In the same way, multiple domain adaptation methods with different characteristic benefits can be combined. For example, this could be

done in the form of a composition $f\left(g(X^s, X^t)\right)$, where $f(\cdot)$ and $g(\cdot)$ denote transformations from different methods.

It was shown in chapter 4, that TCA constrains the covariance matrix of the transformed data to be identity. This is an arbitrary choice and results in the same projection for both the source and target. It would be of interest to incorporate the covariance matrices of the original domain in the constraint, such that a seperate projection matrix can be constructed for the source and target. This would add an extra degree of freedom to match the two. However, the constraint is a covariance matrix in the TCA domain. A method must then be found that transforms the original covariance matrices onto this TCA domain.

Finally, a more formal online target mean and variance estimation method could be used, like Recursive Least Squares.

BIBLIOGRAPHY

[1] Nu.nl, "Schade door noodweer randstad geschat op 20 miljoen euro." https://www.nu.nl/binnenland/4282024/schade-noodweer-randstad-geschat-20-miljoen-euro.html. Accessed: 15-07-2019.

[2] Metro Nieuws, "Drukste avondspits van het jaar." https://www.metronieuws.nl/nieuws/binnenland/2017/12/drukste-avondspits-van-het-jaar. Accessed: 15-07-2019.

[3] KNMI, "Intensiteit van extreme neerslag in een veranderend klimaat." https://www.knmi.nl/kennis-en-datacentrum/achtergrond/intensiteit-van-extreme-neerslag-in-een-veranderend-klimaat.

[4] C. Zhang *et al.*, "Evaluation and intercomparison of high-resolution satellite precipitation estimates—gpm, trmm, and cmorph in the tianshan mountain area," *Remote Sensing*, vol. 10, no. 10, p. 1543, 2018.

[5] P. Hazenburg, *Rainfall estimation for hydrology using volumetric weather radar*. PhD thesis, Wageningen University, 2013.

[6] S. Thorndahl *et al.*, "Weather radar rainfall data in urban hydrology," *Hydrology and Earth System Sciences*, vol. 21, no. 3, p. 1359, 1380.

[7] KNMI, "Weather station locations of the knmi." https://www.weerwoord.be/uploads/2210201273640.gif. Accessed: December 14, 2017.

[8] R. Hut, *New Observational Tools and Datasources for Hydrology: Hydrological data Unlocked by Tinkering*. PhD thesis, Delft University of Technology, 2013.

[9] J. Lyons, "Mel frequency cepstral coefficient tutorial." http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/. Accessed: 15-05-2019.

[10] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5,

no. 2, pp. 241–259, 1991.

[11] D. Tuia, C. Persello, and L. Bruzzone, "Domain adaptation for the classification of remote sensing data: An overview of recent advances," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 41–57, 2016.

[12] S. Pan *et al.*, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–200, 2011.

[13] W. Kouw and M. Loog, "An introduction to domain adaptation and transfer learning." `https://arxiv.org/abs/1812.11806v2`. Accessed: 31-08-2019.

[14] Weather Underground, "Wunderground website." `https://www.wunderground.com/`. Accessed: 20-12-2017.

Appendices

A

POSTERIOR HISTOGRAMS

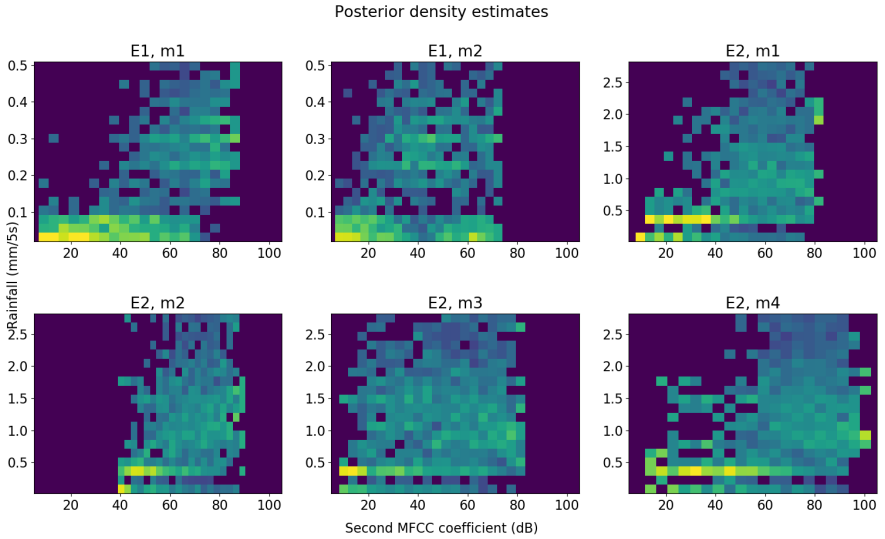

Figure 26.: *Posterior density estimates of the second MFCC feature with the labels.*

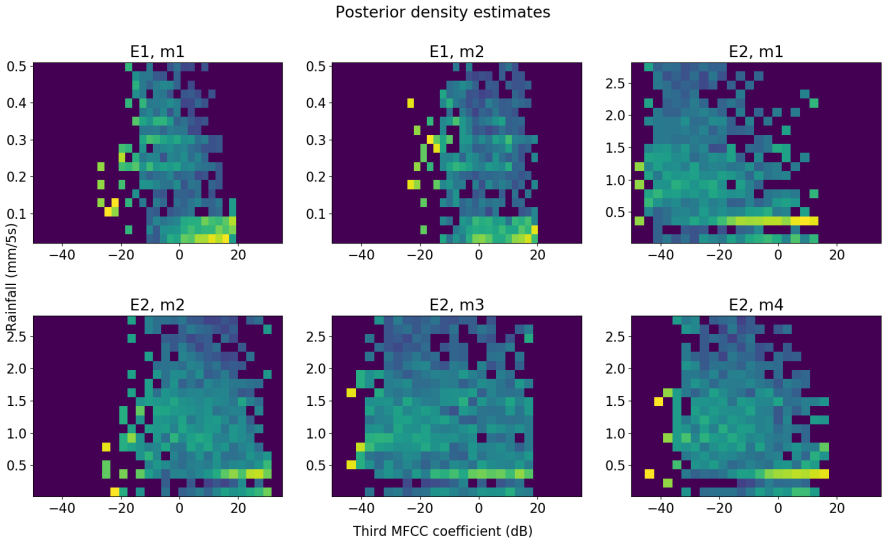

Figure 27.: *Posterior density estimates of the third MFCC feature with the labels.*

B

RESULTS

Source \Target		E1, m1	E1, m2	E2, m1	E2, m2	E2, m3	E2, m4
E1, m1	B	0.758 (0.031)	0.304 (0.094)	0.048 (0.031)	-0.140 (0.031)	-0.109 (0.035)	-0.100 (0.036)
	S	0.758 (0.031)	0.503 (0.065)	0.078 (0.023)	0.006 (0.021)	-0.035 (0.022)	-0.001 (0.022)
E1, m2	B	0.649 (0.052)	0.645 (0.050)	0.039 (0.025)	-0.188 (0.020)	-0.096 (0.021)	0.041 (0.018)
	S	0.659 (0.053)	0.645 (0.050)	0.011 (0.023)	-0.020 (0.018)	-0.020 (0.025)	0.068 (0.017)
E2, m1	B	-3.331 (0.528)	0.004 (0.221)	0.676 (0.032)	0.364 (0.038)	0.453 (0.041)	0.457 (0.037)
	S	-9.129 (1.019)	-4.643 (1.166)	0.676 (0.032)	0.570 (0.035)	0.494 (0.041)	0.593 (0.037)
E2, m2	B	-1.266 (0.389)	-1.166 (0.477)	-0.112 (0.070)	0.650 (0.027)	0.150 (0.077)	0.354 (0.077)
	S	-7.284 (1.436)	-9.179 (1.933)	0.618 (0.039)	0.650 (0.027)	0.484 (0.032)	0.536 (0.047)
E2, m3	B	-3.615 (1.221)	-3.618 (1.184)	0.553 (0.036)	0.557 (0.037)	0.599 (0.034)	0.539 (0.047)
	S	-9.008 (1.769)	-9.987 (2.127)	0.614 (0.036)	0.595 (0.039)	0.599 (0.034)	0.603 (0.030)
E2, m4	B	-2.537 (0.470)	-0.672 (0.331)	0.627 (0.046)	0.392 (0.030)	0.506 (0.044)	0.669 (0.032)
	S	-8.286 (0.883)	-5.781 (1.357)	0.630 (0.040)	0.568 (0.032)	0.529 (0.040)	0.669 (0.032)

Table 14.: *Pairwise R^2 comparison between the baseline (B) and standardization (S). Rows and columns indicate source and target microphone, respectively. Green indicates method that performs significantly better, based on a t-test on the ten folds.*

Source \Target		E1, m1	E1, m2	E2, m1	E2, m2	E2, m3	E2, m4
E1, m1	B	0.758 (0.031)	0.304 (0.094)	0.048 (0.031)	-0.140 (0.031)	-0.109 (0.035)	-0.100 (0.036)
	PCA	0.758 (0.031)	0.406 (0.069)	0.181 (0.021)	-0.131 (0.019)	0.084 (0.027)	0.126 (0.020)
E1, m2	B	0.649 (0.052)	0.645 (0.050)	0.039 (0.025)	-0.188 (0.020)	-0.096 (0.021)	0.041 (0.018)
	PCA	0.454 (0.038)	0.645 (0.050)	0.097 (0.018)	-0.167 (0.014)	0.053 (0.018)	0.062 (0.020)
E2, m1	B	-3.331 (0.528)	0.004 (0.221)	0.676 (0.032)	0.364 (0.038)	0.453 (0.041)	0.457 (0.037)
	PCA	-6.806 (0.772)	-7.295 (1.459)	0.676 (0.032)	0.024 (0.039)	0.438 (0.054)	0.575 (0.044)
E2, m2	B	-1.266 (0.389)	-1.166 (0.477)	-0.112 (0.070)	0.650 (0.027)	0.150 (0.077)	0.354 (0.077)
	PCA	-21.722 (3.763)	-22.557 (2.628)	-0.971 (0.229)	0.650 (0.027)	-0.600 (0.102)	-0.062 (0.085)
E2, m3	B	-3.615 (1.221)	-3.618 (1.184)	0.553 (0.036)	0.557 (0.037)	0.599 (0.034)	0.539 (0.047)
	PCA	-8.359 (1.222)	-8.145 (1.675)	0.554 (0.043)	0.096 (0.048)	0.599 (0.034)	0.614 (0.043)
E2, m4	B	-2.537 (0.470)	-0.672 (0.331)	0.627 (0.046)	0.392 (0.030)	0.506 (0.044)	0.669 (0.032)
	PCA	-7.147 (1.014)	-7.440 (1.354)	0.559 (0.047)	0.148 (0.028)	0.539 (0.035)	0.669 (0.032)

Table 15.: *Pairwise R^2 comparison between the baseline (B) and PCA. Rows and columns indicate source and target microphone, respectively. Green indicates method that performs significantly better, based on a t-test on the ten folds.*

Source \ Target		E1, m1	E1, m2	E2, m1	E2, m2	E2, m3	E2, m4
E1, m1	B	0.758 (0.031)	0.304 (0.094)	0.048 (0.031)	-0.140 (0.031)	-0.109 (0.035)	-0.100 (0.036)
	LTCA	0.757 (0.028)	0.543 (0.069)	0.155 (0.023)	0.085 (0.013)	0.076 (0.025)	0.159 (0.013)
E1, m2	B	0.649 (0.052)	0.645 (0.050)	0.039 (0.025)	-0.188 (0.020)	-0.096 (0.021)	0.041 (0.018)
	LTCA	0.670 (0.037)	0.634 (0.054)	0.067 (0.024)	0.076 (0.016)	0.006 (0.022)	0.069 (0.012)
E2, m1	B	-3.331 (0.528)	0.004 (0.221)	0.676 (0.032)	0.364 (0.038)	0.453 (0.041)	0.457 (0.037)
	LTCA	-6.627 (0.770)	-6.168 (1.127)	0.676 (0.034)	0.532 (0.040)	0.508 (0.036)	0.625 (0.042)
E2, m2	B	-1.266 (0.389)	-1.166 (0.477)	-0.112 (0.070)	0.650 (0.027)	0.150 (0.077)	0.354 (0.077)
	LTCA	-5.140 (0.891)	-6.774 (0.900)	0.606 (0.032)	0.638 (0.030)	0.539 (0.031)	0.606 (0.047)
E2, m3	B	-3.615 (1.221)	-3.618 (1.184)	0.553 (0.036)	0.557 (0.037)	0.599 (0.034)	0.539 (0.047)
	LTCA	-6.470 (1.321)	-8.125 (1.532)	0.638 (0.031)	0.559 (0.034)	0.594 (0.031)	0.631 (0.033)
E2, m4	B	-2.537 (0.470)	-0.672 (0.331)	0.627 (0.046)	0.392 (0.030)	0.506 (0.044)	0.669 (0.032)
	LTCA	-6.937 (0.903)	-4.785 (0.812)	0.651 (0.040)	0.582 (0.029)	0.534 (0.040)	0.667 (0.037)

Table 16.: *Pairwise R^2 comparison between the baseline (B) and TCA with linear kernel (LTCA) where 21 and 6 components are extracted, respectively. Trade-off parameter $\mu = 1$. Rows and columns indicate source and target microphone, respectively. Green indicates method that performs significantly better, based on a paired t-test on the ten folds.*

Source \ Target		E1, m1	E1, m2	E2, m1	E2, m2	E2, m3	E2, m4
E1, m1	LTCA	0.757 (0.028)	0.543 (0.069)	0.155 (0.023)	0.085 (0.013)	0.076 (0.025)	0.159 (0.013)
	RTCA	0.762 (0.035)	0.179 (0.104)	-0.018 (0.039)	-0.249 (0.020)	-0.057 (0.026)	0.057 (0.039)
E1, m2	LTCA	0.670 (0.037)	0.634 (0.054)	0.067 (0.024)	0.076 (0.016)	0.006 (0.022)	0.069 (0.012)
	RTCA	0.374 (0.090)	0.661 (0.059)	-0.098 (0.042)	-0.261 (0.021)	-0.225 (0.027)	-0.188 (0.037)
E2, m1	LTCA	-6.627 (0.770)	-6.168 (1.127)	0.676 (0.034)	0.532 (0.040)	0.508 (0.036)	0.625 (0.042)
	RTCA	-2.876 (0.514)	-1.897 (0.520)	0.682 (0.030)	0.461 (0.071)	0.531 (0.041)	0.516 (0.026)
E2, m2	LTCA	-5.140 (0.891)	-6.774 (0.900)	0.606 (0.032)	0.638 (0.030)	0.539 (0.031)	0.606 (0.047)
	RTCA	-3.800 (0.848)	-0.600 (0.094)	-1.473 (0.752)	0.652 (0.032)	-0.477 (0.092)	-2.187 (0.430)
E2, m3	LTCA	-6.470 (1.321)	-8.125 (1.532)	0.638 (0.031)	0.559 (0.034)	0.594 (0.031)	0.631 (0.033)
	RTCA	-10.373 (2.362)	-3.990 (1.297)	0.627 (0.027)	0.124 (0.183)	0.605 (0.034)	0.650 (0.048)
E2, m4	LTCA	-6.937 (0.903)	-4.785 (0.812)	0.651 (0.040)	0.582 (0.029)	0.534 (0.040)	0.667 (0.037)
	RTCA	-3.134 (0.550)	-2.472 (0.568)	0.608 (0.036)	0.582 (0.031)	0.567 (0.037)	0.672 (0.033)

Table 17.: *Pairwise R^2 comparison between TCA with linear (LTCA) and RBF (RTCA) kernel where 21 components are extracted. Trade-off parameter $\mu = 1$ and bandwidth $\gamma = 15$. Rows and columns indicate source and target microphone, respectively. Green indicates method that performs significantly better, based on a t-test on the ten folds.*

C

TCA DERIVATION

In this chapter, the full derivation of the TCA domain adaptation method is given. First, the use of the Maxmimum Mean Discrepancy (MMD) cost function is motivated and is rewritten in matrix form in section C.1. Then, the kernel trick and it's benefits w.r.t. the MMD is explained in C.2. Finally, the optimization problem constructed from the MMD, which is the core of TCA, is derived in C.3.

C.1 MAXIMUM MEAN DISCREPANCY

The starting point is the Maximum Mean Discrepancy cost function. This is a distance measure between the source and target.

The MMD is defined as follows:

$$Dist(X^S, X^T) = \left\| \frac{1}{N_S} \sum_{i=1}^{N_S} x_i^S - \frac{1}{N_t} \sum_{i=1}^{N_t} x_i^t \right\| \tag{49}$$

where $X^S \in \mathbb{R}^{N_S \times D}$, is the data matrix in the source domain, consisting of N_S number of D-dimensional samples. Similarly, $X^T \in \mathbb{R}^{N_t \times D}$, is the data matrix in the target domain. The MMD can be rewritten more succently in matrix form. To realize this, equation 49 can be written out, resulting in the following equation:

$$\begin{aligned} Dist(X^S, X^T) = & \frac{1}{N_S^2} \sum_{i=1}^{N_S} \sum_{j=1}^{N_S} x_i^S x_j^S \\ & - \frac{2}{N_S N_t} \sum_{i=1}^{N_S} \sum_{j=1}^{N_t} x_i^S x_j^t \\ & + \frac{1}{N_t^2} \sum_{i=1}^{N_t} \sum_{j=1}^{N_t} x_i^T x_j^t \end{aligned} \tag{50}$$

Comparing the MMD in equation 49 and 50, a familiar pattern can be seen, that one would expect from the square of a sum/subtraction. The first term of equation 50 describes the quadratic influence of the source data of equation 49. The double sum operators show that the MMD describes all possible pairwise products of the source and target samples, normalized according to the sample size of the corresponding domain. This can be rewritten in matrix form as follows:

$$\frac{1}{N_S^2}\sum_{i=1}^{N_S}\sum_{j=1}^{N_S} x_i^S x_j^S = tr(K_{SS}L_{SS}) \tag{51}$$

where K_{SS} contains all pairwise dot products of the source samples and L_{SS} contains all corresponding normalizations. These are described in the following two equations:

$$K_{SS} = \begin{bmatrix} x_1^S x_1^S & \dots & x_1^S x_{N_S}^S \\ \vdots & \ddots & \vdots \\ x_{N_S}^S x_1^S & \dots & x_{N_S}^S x_{N_S}^S \end{bmatrix} \tag{52}$$

$$L_{SS} = \begin{bmatrix} \frac{1}{N_S^2} & \cdots & \frac{1}{N_S^2} \\ \vdots & \ddots & \vdots \\ \frac{1}{N_S^2} & \cdots & \frac{1}{N_S^2} \end{bmatrix} \tag{53}$$

Similarly, the second term of equation 50 describes the cross influence of the source and target and the last term describes the quadratic influence of the target data. These terms can also be rewritten in matrix form, as was explicitly shown for the first term in equation 52 and 53. This results in the matrix form of the MMD:

$$\begin{aligned} Dist(X_S, X_T) &= tr(K_{SS}L_{SS}) + tr(K_{St}L_{St}) + tr(K_{tS}L_{tS}) + tr(K_{tt}L_{tt}) \\ &= tr(KL) \end{aligned} \tag{54}$$

where the normalization matrix L is:

$$L = \begin{bmatrix} L_{SS} & L_{St} \\ L_{tS} & L_{tt} \end{bmatrix} = \begin{bmatrix} \frac{1}{N_S^2}11^T & \frac{-1}{N_S N_t}11^T \\ \frac{-1}{N_S N_t}11^T & \frac{-1}{N_t^2}11^T \end{bmatrix} \tag{55}$$

where $L_{SS} \in \mathbb{R}^{N_S \times N_S}, L_{St} \in \mathbb{R}^{N_S \times N_t}, L_{tS} \in \mathbb{R}^{N_t \times N_S}, L_{tt} \in \mathbb{R}^{N_t \times N_t}$ and $L \in \mathbb{R}^{(N_S+N_t)\times(N_S+N_t)}$. Similarly, the pairwise products can be grouped in a kernel matrix:

$$\begin{aligned} K &= \begin{bmatrix} K_{SS} & K_{St} \\ K_{tS} & K_{tt} \end{bmatrix} \\ &= \begin{bmatrix} x_1^S x_1^S & \dots & x_1^S x_{N_S}^S & x_1^S x_1^t & \dots & x_1^S x_{N_t}^t \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{N_S}^S x_1^S & \dots & x_{N_S}^S x_{N_S}^S & x_{N_S}^S x_1^t & \dots & x_{N_S}^S x_{N_t}^t \\ x_1^t x_1^S & \dots & x_1^t x_{N_S}^S & x_1^t x_1^t & \dots & x_1^t x_{N_t}^t \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{N_t}^t x_1^S & \dots & x_{N_t}^t x_{N_S}^S & x_{N_t}^t x_1^t & \dots & x_{N_t}^t x_{N_t}^t \end{bmatrix} \end{aligned} \tag{56}$$

where the shape of K and it's submatrices are equivalent to that of L. Note that from equation 56, it can be seen that the kernel matrix is the outer product of all feature vectors. Therefore, if all source and target data are stacked into a super matrix as in equation 57, the kernel matrix can be written as equation 58.

$$X = \begin{bmatrix} X^S \\ X^t \end{bmatrix} \tag{57}$$

$$K = XX^T \tag{58}$$

Finally, all terms of the MMD in equation 50, can be rewritten similar to 51:

$$\begin{aligned} Dist(X_S, X_T) &= tr(K_{SS}LSS) + tr(K_{St}L_{St}) + tr(K_{tS}L_{tS}) + tr(K_{tt}L_{tt}) \\ &= tr(KL) \end{aligned} \tag{59}$$

C.2 THE KERNEL TRICK

It is seen in equation 49, a mapping $\phi(\cdot)$ is applied to the source and target data, before the distance between the means is determined. Such a mapping transforms the feature vectors into a higher-dimensional space, extracting (non-linear) information from the original data. Although the MMD originally described the distance between the mean and source data, by applying this mapping, it allows to also include other non-linear measures that relate to the variance or higher moments. Consider one dimensional feature vectors where the mapping is given by equation 60

$$\phi(x) = (x, x^2) \tag{60}$$

It can be seen that from the original one dimensional data, including the square adds non-linear information and gives a new two dimensional feature vector. Using this mapping, the MMD in 49, results in equation 61

$$Dist(X^S, X^T) = (\frac{1}{N_S}\sum_{i=1}^{N_S} x_i^S - \frac{1}{N_t}\sum_{i=1}^{N_t} x_i^t)^2 + (\frac{1}{N_S}\sum_{i=1}^{N_S}(x_i^S)^2 - \frac{1}{N_t}\sum_{i=1}^{N_t(} x_i^t)^2)^2 \tag{61}$$

Indeed, now, the MMD not only describes a square distance between the mean of the source and target data, but also something that relates to the variance. Thus, choosing an appropriate mapping helps to augment the MMD to describe additional properties between the source and data distribution, aside from the sample mean.

The quest for explicitly finding a mapping is non-systematic. Furthermore, it can be computationally heavy or infeasible. Luckily, there exist functions, referred to as kernels, that find this mapping implicitly, by computing the dot product of mapped feature vectors. Re-visiting the equation for the kernel matrix, which is a part of the MMD, in equation 56, it can be seen that, indeed, this information turns out to be sufficient to utilize the mapping. Thus, instead of computing the mapping directly, finding the pairwise dot products of all mapped feature vectors suffices. To show that such functions exist, consider a two-dimensional feature vector and a mapping applied on it, given by equation 62 and equation 63, respectively.

$$x = (x_1, x_2) \tag{62}$$

$$\phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \tag{63}$$

Now, consider another feature vector, y, where the same mapping has been applied. Then, the dot product of the two mapped feature vectors, is given in equation 64.

$$\begin{aligned}\phi(x)^T\phi(y) &= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1y_1x_2y_2 \\ &= ((x_1, x_2)(y_1, y_2)^T)^2 \\ &= (x^ty)^2\end{aligned} \tag{64}$$

The dot product of mapped feature vectors using the mapping in equation 63, is equivalent to the polynomial kernel in equation 64, which computes the dot product of the original feature vectors, where after the result is squared. The latter is computationally more efficient and systematic. Thus, the kernel matrix of equation 56 can be more efficiently computed by choosing an appropriate kernel function. The kernel chosen defines which statistical measures the MMD

can describe as well as the characterstics of the higher-dimensional space, the original feature vectors are transformed onto. The MMD in equation 49 can be modified to utilize the mapping discussed in this section. This is shown in the following equation:

$$Dist(X^S, X^T) = \left\| \frac{1}{N_S} \sum_{i=1}^{N_S} \phi(x_i^S) - \frac{1}{N_t} \sum_{i=1}^{N_t} \phi(x_i^t) \right\| \tag{65}$$

The kernel matrix from equation 58, is also modified accordingly as shown in the equation below:

$$K = \phi(X)\phi(X)^T \tag{66}$$

Thus, every feature vector x, is transformed to $\phi(x)$, so that the MMD not only matches the means of the source and target distribution, rather other (higher moment) statistical properties as well.

C.3 OPTIMIZATION PROBLEM

Up until now, it has only been described how to *measure* the distance between the source and target data. However, in order to *control* their distance, i.e. make their respective distributions more similar, every feature vector must be reweighted. This can be done by applying a weight matrix to the complete data matrix of equation 57 as in equation 67.

$$\hat{\phi}(X) = \phi(X)W \tag{67}$$

where $W \in \mathbb{R}^{(N_S+N_t)\times m}$. This weight matrix thus adds a weight to all features and transforms them to a m-dimensional latent space, lower than the original space, i.e. $m < D$. Then, a modified kernel matrix can be defined, by substituting equation 67 in equation 58, resulting in equation 68.

$$\begin{aligned} \tilde{K} &= \hat{\phi}(X)\hat{\phi}(X)^T \\ &= \phi(X)WW^T\phi(X)^T \end{aligned} \tag{68}$$

The modified kernel matrix can be expressed in terms of the original kernel matrix by expressing the kernel matrix as equation 69.

$$K = KK^{-\frac{1}{2}}K^{-\frac{1}{2}}K \tag{69}$$

Comparing equation 58 with 69, it can be seen that the mapped data in the TCA domain can be written in terms of the kernel. This is given in equation 70.

$$\phi(X) = KK^{-\frac{1}{2}} \tag{70}$$

Then, this can be substituted in equation 68, to describe the modified kernel in terms of the original kernel and weight matrix. Finally, the MMD in matrix form in equation 59, with a weight matrix incorporated, can then be expressed as equation 71

$$\begin{aligned} Dist(X_S, X_T) &= tr(\tilde{K}L) \\ &= tr((KK^{-\frac{1}{2}}WW^TK^{-\frac{1}{2}}K)L) \\ &= tr((K\tilde{W}\tilde{W}^TK)L) \\ &= tr(\tilde{W}^TKLK\tilde{W}) \end{aligned} \tag{71}$$

where $K^{-\frac{1}{2}}$ is applied to the original weight matrix W, resulting in a modified weight matrix, given in equation 72.

$$\tilde{W} = K^{-\frac{1}{2}}W \tag{72}$$

Now, the core of TCA has been reached. The goal is to minimize the weighted MMD from equation 71. Additionally, a regularization term is included in order to control the complexity of $\tilde{W}$. Thus, the optimization problem of equation 73, must be solved.

$$min_{\tilde{W}} \;\; \mu tr(\tilde{W}^T\tilde{W}) + tr(\tilde{W}^TKLK\tilde{W}) \tag{73}$$

Before continuing with deriving the objective function of equation 73, it can be seen that there are trivial solutions for this minimization. Indeed, $\tilde{W} = 0$, minimizes the objective, but collapses all samples from the source and target onto the origin. Also, the possibility exists, that there exist a $\tilde{W} \neq 0$, such that all samples collapse onto some other point in the feature space, which is not the origin. Lastly, if $\tilde{W}$ is in the null space of K, similar to $\tilde{W} = 0$, the MMD part, excluding the regularization term is minimized and all samples collapse onto the origin. This can be more clearly seen, when regarding the transformation from the original domain to the TCA domain, once the optimal $\tilde{W}$ is found, given in equation 74.

$$\begin{aligned} X_{TCA} &= \phi(X)W \\ &= K\tilde{W} \end{aligned} \tag{74}$$

This is done by applying the weight matrix to the mapped feature vectors . However, this is equivalent to projecting the modified weight

matrix onto the kernel matrix, which can be seen by substitution using equation 70 and 72.

The collapsing of the transformed feature vectors, due to the reasons described, can be prevented by constraining the variance of the feature vectors in the TCA domain to be fixed. To succinctly apply a centering on the feature vectors to achieve a mean of zero, as part of the computation of the covariance matrix, a centering matrix is applied. Multiplication by this centering matrix is equivalent to subtracting the mean from the feature vectors. It is defined as:

$$H = I_{N_S+N_t} - \frac{1}{N_S+N_t}11^T \tag{75}$$

where $H \in \mathbb{R}^{(N_S+N_t)\times(N_S+N_t)}$, which is also the case for the identity matrix, I and the all ones matrix 11^T, in equation 75. Two properties of the centering matrix that will be used in defining the constraint are given below:

- idempotent: $HH = H$
- symmetric: $H^T = H$

For simplicity, the scattering matrix is constrained instead of the covariance matrix. It does not require normalization by the sample size as with the covariance matrix, which is merely a scaling operation affecting all features equally and thus, does not affect the performance of TCA to prevent transformed samples to collapse onto one point in the TCA feature space. The constraint for the scattering matrix is then:

$$\begin{aligned} S &= (HX_{TCA})(HX_{TCA})^T \\ &= (HK\tilde{W})(HK\tilde{W})^T \\ &= \tilde{W}^T KHK\tilde{W} \end{aligned} \tag{76}$$

where substitution with equation 74 is used. Furthermore, in the last equality, the idempotency property of the centering matrix is used and the symmetry property of both the centering and kernel matrix.

Then, combining equation 73 and 76, the complete constrained optimization problem becomes:

$$\begin{aligned} &min_{\tilde{W}} \;\; \mu tr(\tilde{W}^T\tilde{W}) + tr(\tilde{W}^T KLK\tilde{W}) \\ &s.t. \tilde{W}^T KHK\tilde{W} = I \end{aligned} \tag{77}$$

The Lagrangian is:

$$\mathcal{L}(\tilde{W}, Z) = tr[\tilde{W}^T(I_{N_S+N_t} + \mu KLK)\tilde{W}] + (\tilde{W}^T KHKW - I)Z^1 \quad (78)$$

where Z is a symmetric matrix, consisting of the Lagrangian multipliers. It should be noted that, when the constraint is a vector, as would be the case in the form $Ax = b$, the Lagrangian multipliers also form a vector. In this case, the constraint is a matrix and thus, every Lagrangian multiplier is a vector, forming a matrix.

Setting the partial derivative of equation 78 w.r.t. $\tilde{W}$ to zero, results in equation 79

$$\begin{aligned} \frac{\partial \mathcal{L}(\tilde{W}, Z)}{\partial \tilde{W}} &= 0 \\ (I + \mu KLK)\tilde{W} &= KHK\tilde{W}Z \end{aligned} \quad (79)$$

where the matrix identities for deriving the first and second term of 78 are given in equation 80 and 81, respectively.

$$\frac{\partial tr(X^T AX)}{\partial X} = X^T(A + A^T) \quad (80)$$

Where $A = (I_{N_S+N_t} + \mu KLK)$.

$$\frac{\partial tr(X^T BXC)}{\partial X} = BXC + B^T XC^T \quad (81)$$

Where $B = KHK$ and $C = Z$.

Then, after left-multiplying equation 79 by $\tilde{W}^T$, it can be substituted back in 78, resulting in the following equation:

$$\begin{aligned} \mathcal{L}(\tilde{W}, Z) &= tr(Z) \\ &= tr[(\tilde{W}^T KHK\tilde{W})^{-1}\tilde{W}^T(I_{N_S+N_t} + \mu KLK)\tilde{W}] \end{aligned} \quad (82)$$

where the last equality is gained by solving for Z in equation 79. Now, that the penalty term consisting of the Lagrangian multipliers, Z, has been substituted in the Lagrangian, equation 82 must be minimized. However, it is noted that the minimization of this division

1 In the original paper, a trace operation is applied to the constraint, before being added to the Lagrangian. This is because the constraint is a matrix, whereas the trace operation summarizes it's properties as a scalar. However, conceptually this means that any constraint in this matrix can be heavily violated in favor of reducing the overall penalty from other constraints. However, it is preferable that all constraints are satisfied jointly.

is equivalent to the maximization of it's inverse. Using this fact, the maximization problem to be solved from equation 82 is:.

$$max_{\tilde{W}} \ \ tr\left[\frac{\tilde{W}^T KHK\tilde{W}}{\tilde{W}^T(I_{N_S+N_t}+\mu KLK)\tilde{W}}\right] = max_{\tilde{W}} \ \ tr\left[f(\tilde{W})\right] \tag{83}$$

Using the chain rule, equation 82 can be simplified as follows:

$$\begin{aligned}\frac{\partial tr\left[f(\tilde{W})\right]}{\partial \tilde{W}} &= \frac{\partial tr\left[f(\tilde{W})\right]}{\partial f(\tilde{W})} \cdot \frac{\partial f(\tilde{W})}{\partial \tilde{W}} \\ &= \frac{\partial \sum_{i=0}^{N-1} f(\tilde{W})_{ii}}{\partial f(\tilde{W})_{ij}} \cdot \frac{\partial f(\tilde{W})}{\partial \tilde{W}} \\ &= I \cdot \frac{\partial f(\tilde{W})}{\partial \tilde{W}} = \frac{\partial f(\tilde{W})}{\partial \tilde{W}}\end{aligned} \tag{84}$$

where the elements of the trace derivative in the second equality are 0, if $i \neq j$ and 1, if $i = j$, hence it is the identity matrix.

Thus, the maximization of equation 83 simplifies to deriving the division within the trace operator, according to equation 84 and setting it to zero. Using the quotient rule this results in:

$$\frac{\partial f(\tilde{W})}{\partial \tilde{W}} = \tilde{W}^T KHK\tilde{W}(I_{N_S+N_t}+\mu KLK)\tilde{W} - \left[\tilde{W}^T(I_{N_S+N_t}+\mu KLK)\tilde{W}\right] KHK\tilde{W} = 0 \tag{85}$$

Rewriting the above equation results in the final solution:

$$KHK\tilde{W} = \frac{\tilde{W}^T KHK\tilde{W}}{\tilde{W}^T(I_{N_S+N_t}+\mu KLK)\tilde{W}}(I_{N_S+N_t}+\mu KLK)\tilde{W}^2 \tag{86}$$

It can be seen that this is in the form of $A\tilde{W} = \lambda B\tilde{W}$. Indeed, TCA boils down to a generalized eigenvalue decomposition, where the eigenvectors corresponding to the m leading/largest eigenvalues of $\frac{KHK}{I_{N_S+N_t}+\mu KLK}$ form the columns of the weight matrix. Finally, the found weight matrix can be used to project the original data onto the TCA domain using equation 74.

2 In the original paper, the generalized eigenvalue problem consists of the identity matrix I, which is defined to be of size $m \times m$. This should be $I_{n_1+n_2}$ according to the notation of the paper and thus $I_{N_S+N_t}$ according to that of this thesis.

D

TCA EXAMPLE

A step-by-step example of TCA is given below. Consider a data feature matrix, where the first and last two rows represent feature vectors of the source and target domain, respectively:

$$X = \begin{bmatrix} 0 & 2 \\ -1 & 1 \\ \frac{1}{2} & 2 \\ \frac{1}{2} & 4 \end{bmatrix} \tag{87}$$

Using a linear kernel, the kernel matrix is then computed.

$$K = XX^T + \epsilon I = \begin{bmatrix} 4.0001 & 2 & 4 & 8 \\ 2 & 2.0001 & 1.5 & 3.5 \\ 4 & 1.5 & 4.2501 & 8.25 \\ 8 & 3.5 & 8.25 & 16.2501 \end{bmatrix} \tag{88}$$

Where the second term ensures that the kernel matrix is positive definite and $\epsilon = 10^{-4}$. The normalization matrix is computed as below.

$$\begin{aligned} L &= \begin{bmatrix} \frac{1}{N_S^2} & \frac{1}{N_S^2} & -\frac{1}{N_S N_T} & -\frac{1}{N_S N_T} \\ \frac{1}{N_S^2} & \frac{1}{N_S^2} & -\frac{1}{N_S N_T} & -\frac{1}{N_S N_T} \\ -\frac{1}{N_S N_T} & -\frac{1}{N_S N_T} & \frac{1}{N_S^2} & \frac{1}{N_S^2} \\ -\frac{1}{N_S N_T} & -\frac{1}{N_S N_T} & \frac{1}{N_S^2} & \frac{1}{N_S^2} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix} \end{aligned} \tag{89}$$

Where N_S and N_T denote the number of samples in the source and target domain, which is two for both.

Furthermore, the centering matrix is given below:

$$\begin{aligned} H &= I - 11^T \frac{1}{N_S + N_T} \\ &= \begin{bmatrix} \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} \end{bmatrix} \end{aligned} \tag{90}$$

Now, the eigenvalues and eigenvectors of the following cost function can be computed:

$$\frac{KHK}{I + \mu KLK} \tag{91}$$

Where $\mu = 10^{-4}$. As was discussed in Appendix A, any multiple k, of the columns of the eigenvectors is a correct solution. Typically, software libraries give the eigenvectors such that their length is 1. Because the constraint in TCA requires the transformed data to have a variance of 1, all eigenvectors returned by the software package, are divided by the square root of their corresponding eigenvalues. More specifically, every column of the returned eigenvector matrix, which is also the weight matrix, is divided by it's corresponding eigenvalue. Furthermore, because the feature vectors are two-dimensional, only the two largest eigenvectors with corresponding eigenvectors are considered. The resulting eigenvalues and normalized eigenvector matrix is given below:

$$\Lambda = \begin{bmatrix} 120.52 \\ 0.8167 \end{bmatrix} \tag{92}$$

with corresponding normalized eigenvector/weight matrix:

$$W = \begin{bmatrix} -0.03510 & 0.19503 \\ -0.01017 & 1.04894 \\ -0.03878 & -0.2807 \\ -0.07387 & -0.0855 \end{bmatrix} \tag{93}$$

It can be checked wether the constraint indeed holds, using this normalized weight matrix. The constraint is therefore computed below:

$$W^T KHKW = \begin{bmatrix} 1.0636 & -4.441 \cdot 10^{-16} \\ -4.545 \cdot 10^{-16} & 1.0001 \end{bmatrix} \tag{94}$$

Indeed, the diagonal elements are close to one and the off-diagonals are many orders of magnitude smaller. Thus, the normalization of

the eigenvectors by the corresponding eigenvalues, has successfully resulted in satisfying the constraint in TCA.

Now, the weight matrix is used to project the original feature matrix onto the TCA domain:

$$X_{TCA} = KW = \begin{bmatrix} -.9069 & 1.0709 \\ -.4073 & 1.7676 \\ -.9300 & 0.4548 \\ -1.837 & 1.5258 \end{bmatrix} \tag{95}$$

Where the first and last two rows represent the source and target feature vectors in the TCA domain, respectively.

To inspect whether TCA has brought the source and target distributions closer together, the moment score metric from Chapter 4, equation 24 is used with $k = 0$. Thus, the sample means of the source and target are compared and the score in this form is repeated below:

$$S = \frac{||\mu_S - \mu_T||_2}{\sigma_S + \sigma_T} \tag{96}$$

where the lower the score, the better the means of the source and target are matched, considering their variances. Using the score metric for the original feature matrix in equation 87 and the TCA transformed feature matrix in equation 95, results in the scores $S_{original} = 0.41$ and $S_{TCA} = 0.22$. Indeed, the score for the TCA transformed data is almost twice as small as the original data and thus, it is concluded that TCA has successfully accomplished the goal of making the source and target distributions more similar, by bringing their respective means closer together.

E

VARIANCE MINIMIZATION

A sample weighting method is considered that transforms the source labels, such that its transformed variance, $s(wy^S) = \frac{1}{N_s}\sum_i (wy^S(i) - m^S)^2 = w^2 s(y^S)$, is closer to that of the target, $s(y^t) = \frac{1}{N_t}\sum_i (y^t(i) - m^t)^2$. This is done under the constraint that the sample mean of the transformed source labels, $m(wy^S) = \frac{1}{N_s}\sum_i wy^S(i)$, is within a margin α of the target sample mean, $m(y^t) = \frac{1}{N_t}\sum_i y^t(i)$. The source and target variances and means will be represented as s^S, s^t, m^S and m^t.

E.1 DERIVATION

The optimization problem is defined as:

$$\begin{aligned} &min_w (w^2 s^S - s^t)^2 \\ &s.t. \left(\frac{wm^S - m^t}{m^t}\right)^2 \leq (\alpha)^2 \end{aligned} \tag{97}$$

where α is the upper bound by which the transformed source mean, wm^S, is allowed to deviate from the target mean. The upper bound is defined relative to the target mean. For example, for $\alpha = 0.1$ the transformed source mean is allowed to deviate less than or equal to 10% from the target mean.

The Lagrangian is:

$$L(w, \lambda) = (w^2 s^S - s^t)^2 + \lambda \left(\left(\frac{wm^S - m^t}{m^t}\right)^2 - \alpha^2 \right) \tag{98}$$

Derive and set to zero:

$$\frac{\partial L(w,\lambda)}{\partial w} = 4w^3(s^S)^2 - 4ws^Ss^t + \lambda\left(\frac{2w(m^s)^2 - 2m^Sm^t}{(m^t)^2}\right) = 0 \quad (99)$$

Solve for λ:

$$\lambda = \frac{-2w^3(s^S)^2(m^t)^2 + 2ws^Ss^t(m^t)^2}{w(m^S)^2 - m^tm^S} \quad (100)$$

Substituting this λ in equation 98:

$$L(w) = (w^2s^S - s^t)^2 + \left(\frac{-2w^3(s^S)^2(m^t)^2 + 2ws^Ss^t(m^t)^2}{w(m^S)^2 - m^tm^S}\right)\left(\left(\frac{wm^S - m^t}{m^t}\right)^2 - \alpha^2\right) \quad (101)$$

Deriving again, simplifying and setting to zero:

$$\frac{\partial L(w)}{\partial w} = \frac{p_5w^5 + p_4w^4 + p_3w^3 + p_2w^2 + p_1w + p_0}{m^S(wm^S - m^t)^2} = 0 \quad (102)$$

$$p_5 = -4(m^S)^3(s^S)^2 \quad (103)$$

$$p_4 = 14(m^S)^2m^t(s^S)^2 \quad (104)$$

$$p_3 = 4\alpha^2m^S(m^t)^2(s^S)^2 - 16m^S(m^t)^2(s^S)^2 \quad (105)$$

$$p_2 = -6\alpha^2(m^t)^3(s^S)^2 - 2(m^S)^2m^ts^Ss^t + 6(m^t)^3(s^S)^2 \quad (106)$$

$$p_1 = 4m^S(m^t)^2s^Ss^t \quad (107)$$

$$p_0 = 2\alpha^2(m^t)^3s^Ss^t - 2(m^t)^3s^Ss^t \quad (108)$$

Solving equation 102 for w, results in five solutions:

$$w_1 = \frac{m^t(1+\alpha)}{m^S} \quad (109)$$

$$w_2 = \frac{m^t(1-\alpha)}{m^S} \tag{110}$$

$$w_3 = \frac{(m^t)^2(s^S)^2 + m^t s^S f^{\frac{1}{3}} + f^{\frac{2}{3}}}{2m^S s^S f} \tag{111}$$

$$w_4, w_5 = \frac{-f^{\frac{1}{3}}}{4m^S s^S} - \frac{(m^t)^2 s^S}{4m^S f^{\frac{1}{3}}} + \frac{m^t}{2m^S} \pm \frac{1}{2} j\sqrt{3}\left(\frac{f}{2m^S s^S} - \frac{(m^t)^2 s^S}{2m^S f}\right) \tag{112}$$

where f is:

$$f = m^t(s^S)^2\left(-2s^t(m^S)^2 + (m^t)^2 s^S + 2m^S\sqrt{s^t(s^t(m^S)^2 - (m^t)^2 s^S)}\right) \tag{113}$$

It is noticeable that only w_1 and w_2 depend on α. The influence of α on the variance matching is described for different situations, based on these two weights.

- $s^S < s^t, m^S < m^t$: when the source mean is smaller than the target mean, the former is scaled up to match the latter by the mean ratio $\frac{m^t}{m^S}$, because it will be greater than 1. The source variance is also smaller than the target variance, which also requires an upscaling. This is also achieved by the same scaling with the mean ratio. If after this upscaling the source variance is still smaller, w_1 is preferable which adds a margin $(1+\alpha)$. If the upscaling is so large that it upscales the source variance to be much larger than the target variance, the margin $(1-\alpha)$ in w_2 can counteract the scaling in favor of the variance matching. This however, results in the transformed source labels to deviate from the target labels: $m(wy^S) = \frac{m^t}{m^S}(1+\alpha)m^S = m^t(1+\alpha)$. Thus, this results in a deviation of αm^t or α percent w.r.t. the target mean. Therefore, the matching in the mean is trade off in favour of a better variance matching.

- $s^S > s^t, m^S > m^t$: similarly, in this case the source mean and variance must be scaled down. The mean ratio achieves this, because it is smaller than 1 in this case. If after this downscaling the source variance is still smaller, w_2 is preferable which removes a margin $(1-\alpha)$. If the downscaling is so large that it downscales the source variance to be much smaller than the target variance, the margin $(1+\alpha)$ in w_2 can counteract the scaling in favor of the variance matching.

- $s^S < s^t, m^S > m^t$: the source mean must be downscaled, which counteracts the source variance, which must be upscaled. The mean ratio $\frac{m^t}{m^S}$ therefore contributes to match the means, while the factor $(1 + \alpha$ in w_1 is needed to counteract this in favour of the variance matching.

- $s^S > s^t, m^S < m^t$: Similarly, the source mean must be upscaled, which counteracts the source variance, which must be downscaled. The mean ratio $\frac{m^t}{m^S}$ therefore contributes to match the means, while a factor $(1 - \alpha$ in w_2 is needed to counteract this in favour of the variance matching.

- $m^S = m^t$: when the source and target mean are equal, the mean ratio $\frac{m^t}{m^S} = 1$ and has no effect when applied to the source labels. However, the factor $(1 + \alpha)$ or $(1 - \alpha)$ can still scale the variance up or down, which results the means to become unmatched.

- $s^S = s^t$: Similar to the previous case, the weight has no effect on the source variance, but the factor $(1 + \alpha)$ or $(1 - \alpha)$ can scale the mean up or down, which results the variances to become unmatched.

It can be seen that in the above cases, the choice of w_1 or w_2 depends on the inequality between means and variances; whether the source is larger or smaller than the target. Furthermore, it also depends on the absolute distance between the means and variances.

Instead of considering every specific case before deciding on a suitable weight, the weight is chosen based on the following:

$$\begin{aligned} w* = & argmin(w^2 s^S - s^t)^2 \\ & w \in W \end{aligned} \tag{114}$$

where W is the set containing the five solutions.

Thus, the weight that fulfills the objective best is chosen.

E.2 EXPERIMENT

The variance method is applied on the rainfall dataset where E1, m1 is chosen as source and E2, m1 as target. In this setting, $s^S < s^t, m^S < m^t$.

The method is evaluated for $0 \leq \alpha \leq 1$ and the scores for the variance and mean matching are:

$$\Delta_s = \frac{|w^2 s^S - s^t|}{s^t} \tag{115}$$

$$\Delta_m = \frac{|w m^S - m^t|}{m^t} \tag{116}$$

The results are given below:

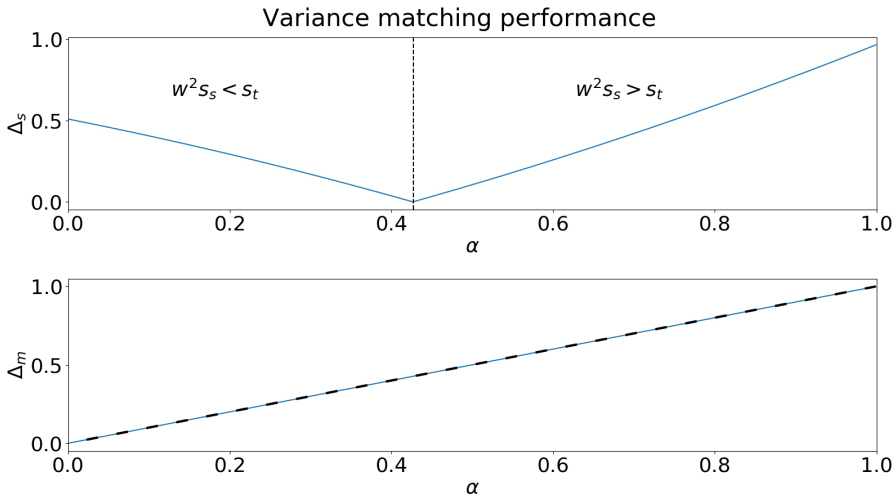

Figure 28.: *Upper: Variance matching performance for different α. Vertical line splits graph in two parts. In the left part the variance of the transformed source labels is smaller than that of the target labels. In the right part, this is the opposite. Lower: Mean matching performance for different α. Black line represents the upper bound of the constraint in equation 97, which states that the relative difference between the source and target mean, may not exceed α.*

It can be seen in the lower plot that the mean matching satisfies the constraint with equality. Furthermore, it can be seen that adding a margin in the mean matching, the variance matching improves, but degrades after $\alpha = 0.42$. This is because initially $s^S < s^t$, which requires an upscaling of the source variance. Choosing a too large α, scales the source variance such that the inequality flips and $w^2 s^S > s^t$, causing the variances to move away from each other. Therefore, a gridsearch could be performed to find the α that results in the best variance matching. Any α higher than this is not suitable.

F

TCA CODE

```
# Author: Arman Naseri Jahfari
# Simplified and slightly optimized from:
# Original author: W.M. Kouw
# https://github.com/wmkouw/libTLDA/blob/master/libtlda/tca.py

import numpy as np
from scipy.sparse.linalg import eigs
from scipy.linalg import cholesky, LinAlgError, inv
from scipy.spatial.distance import cdist

class TCA(object):
    """
    Transfer Component Analysis.
    """

    def __init__(self,
                 mu=1.0,
                 num_components=40, # Number of MFCC features
                 kernel_type='rbf',
                 bandwidth=1.0,
                 order=2.0,
                 kernel_reg_step = 1e-10,
                 kernel_reg_start = 0):
        """
        Select parameters for transfer component analysis.
        Parameters
        ----------
        mu : float
            trade-off parameter (def: 1.0)
        num_components : int
            number of transfer components to maintain (def: 1)
        kernel_type : str
            type of kernel to use, options: 'linear', 'polynomial', 'rbf',
            'sigmoid' (def: 'linear')
```

```
        bandwidth : float
            kernel bandwidth for RBF kernel (def: 1.0)
        order : float
            order of polynomial kernel (def: 2.0)
        kernel_reg_step : float
            step size to increase kernel regularization (def: 1e-10)
        kernel_reg_start : float
            Initial value of kernel regularization (def: 0)
        Returns
        -------
        None
        Attributes
        ----------
        """
        self.mu = mu
        self.num_components = num_components

        self.kernel_type = kernel_type
        self.bandwidth = bandwidth
        self.order = order
        self.kernel_reg_step = kernel_reg_step
        self.kernel_reg_start = kernel_reg_start

    def kernel(self, X, Z, type='rbf', order=2, bandwidth=1.0):
        """
        Compute kernel for given data set.
        Parameters
        ----------
        X : array
            data set (N samples by D features)
        Z : array
            data set (M samples by D features)
        type : str
            type of kernel, options: 'linear', 'polynomial', 'rbf',
            'sigmoid' (def: 'linear')
        order : float
            degree for the polynomial kernel (def: 2.0)
        bandwidth : float
            kernel bandwidth (def: 1.0)
        Returns
        -------
        array
            kernel matrix (N+M by N+M)
        """
        # Data shapes
        N, DX = X.shape
```

```
        M, DZ = Z.shape

        # Assert equivalent dimensionalities
        if not DX == DZ:
            raise ValueError('Dimensionalities of X and Z should be equal.')

            # Select type of kernel to compute
        if type == 'linear':
            # Linear kernel is data outer product
            return np.dot(X, Z.T)
        elif type == 'polynomial':
            # Polynomial kernel is an exponentiated data outer product
            return (np.dot(X, Z.T) + 1) ** p
        elif type == 'rbf':
            # Radial basis function kernel
            return np.exp(-cdist(X, Z) / (2. * bandwidth ** 2))
        elif type == 'sigmoid':
            # Sigmoidal kernel
            return 1. / (1 + np.exp(np.dot(X, Z.T)))
        else:
            raise NotImplementedError('Loss not implemented yet.')

    def transfer_component_analysis(self, X, Z, **flags):
        """
        Transfer Component Analysis.
        Parameters
        ----------
        X : array
            source data set (N samples by D features)
        Z : array
            target data set (M samples by D features)
        Returns
        -------
        X : array
            domain adapted source data set (N samples by m features)
        Z : array
            domain adapted target data set (M samples by m features)
        """
        # Data shapes
        N, DX = X.shape
        M, DZ = Z.shape
        # print(X.shape)
        # print(Z.shape)

        # Assert equivalent dimensionalities
        if not DX == DZ:
```

```
        raise ValueError('Dimensionalities of X and Z should be equal.')

    # Assert correct number of components for given dataset
    if not self.num_components <= N + M - 1:
        raise ValueError('''Number of components must be
                            smaller than or equal to the
                            source sample size plus target sample
                            size plus 1.''')

    ### Compute kernel matrix ###
    XZ = np.concatenate((X, Z), axis=0)
    K = self.kernel(XZ, XZ, type=self.kernel_type,
                    bandwidth=self.bandwidth)

    ### Footnote 1, page 3 ###
    ### add regularization to diagonals to make K positive definite ###

    posdef_warning = False

    regct = self.kernel_reg_start
    K += np.eye(N + M) * regct

    while not is_pos_def(K):
        # Give warning only once if not PD
        if not posdef_warning:
            print('Warning: covariate matrices not PD.')
            posdef_warning = True

        regct += self.kernel_reg_step
        print('Adding regularization: ' + str(regct))

        # Add regularization
        K += np.eye(N + M)*self.kernel_reg_step

    # Normalization matrix
    LSS = np.ones([N, N])/N**2
    LST = -1*np.ones([N, M])/(N*M) # equal to LTS
    LTT = np.ones([M, M])/M**2])
    L = np.vstack([np.hstack([LSS , LST ]),
                   np.hstack([LTS,  LTT ])

    I = np.eye(N + M)

    # Centering matrix
    H = I - np.ones((N + M, N + M)) / float(N + M)
```

```
        KLK = np.dot(np.dot(K, L), K)
        KHK = np.dot(np.dot(K, H), K)

        ### page 3, last paragraph ###
        ### Solution of optimization problem ###
        ### is Generalized eigenvector ###
        # a, b = KHK , I + self.mu * KLK
        # lambdas, W = geig(a, b)

        # Rewriting as 'regular' eigenvalue decomposition  is faster
        # (I + mu*K*L*K)^{-1}*K*H*K
        J = np.dot(inv(I + self.mu * KLK), KHK)
        lambdas, W = eigs(J, k=self.num_components, which='LM')

        # Normalize to unit variance to make sure constraint is satisfied.
        # This is done by dividing every eigenvector by the square root
        # of the corresponding eigenvalue
        W = 1/np.sqrt(lambdas)*W

        # Discard imaginary numbers (possible computation issue)
        W = np.real(W)

        # Project source and target data onto transfer components
        X = np.dot(K[:N, :], W)
        Z = np.dot(K[N:, :], W)

        return X, Z

def is_pos_def(K):
    """ Kernel is positive-definite if Cholesky decomposition exists """
    try:
        cholesky(K)
        return True
    except LinAlgError:
        return False
```