Covariate Shift Adaptation via Sparse Filtering for High-Dimensional Periodic Data

Fabio Massimo Zennaro
School of Computer Science
The University of Manchester
Manchester, M13 9PL, UK
zennarof@cs.manchester.ac.uk

Ke Chen
School of Computer Science
The University of Manchester
Manchester, M13 9PL, UK
chen@cs.manchester.ac.uk

1 Introduction

A common assumption in machine learning is the *independence and identical distribution* (i.i.d.) of the data samples, which states that training and test samples are independent and drawn from the same probability distribution. Unfortunately, real-world data rarely satisfies this assumption. The *covariate shift* assumption [10] states that the training data $\mathbf{X}^{tr}$ and test data $\mathbf{X}^{tst}$ are sampled from two multivariate random variables X^{tr} and X^{tst} with different distributions $p\left(X^{tr}\right)$ and $p\left(X^{tst}\right)$, while the conditional distribution of labels given the data $p\left(Y|X\right)$ is assumed to be the same over training and test data, i.e., $p\left(Y|X^{tr}\right) = p\left(Y|X^{tst}\right)$. A common approach to perform *covariate shift adaptation* (CSA) is through *representation learning*: learn a mapping f that projects the training and test data onto new representations $\mathbf{Z}^{tr}$ and $\mathbf{Z}^{tst}$ having similar distributions $p\left(Z^{tr}\right)$ and $p\left(Z^{tst}\right)$, then learn $p\left(Y|Z\right)$ using standard machine learning algorithms [2].

Feature distribution learning (FDL) is an approach to representation learning that disregards the problem of modeling the distribution of the original data and focuses instead on shaping a useful distribution for the learned representations. *Sparse filtering* (SF) is a FDL algorithm that learns a maximally sparse representation of the data [8]. SF has exhibited good performance in real applications [8] and in previous work we showed that SF works by preserving the structure of the data under the cosine metric, by generating hyper-conical filters in the original space of the data [13].

So far, FDL has never been used for CSA. We consider the possibility of exploiting the insensitivity of FDL to the distribution of the original data in order to efficiently carry out CSA, especially for high-dimensional data. We aim at combining the efficiency and the computational lightness typical of FDL algorithms with the simplicity of the representation learning approach to for CSA.

In this paper we focus on the problem of CSA for high-dimensional data that exhibit a periodic structure using a FDL approach. We propose a novel semi-supervised sparse filtering-based algorithm, named *periodic sparse filtering*. We devised this algorithm to work with high-dimensional data that are affected by covariate shift and that exhibit local similarities. This model is particularly relevant for dealing with high-dimensional user-dependent data: when training and test data are collected from different users, they may be described by different pdfs whose local behavior (areas high or low density and conditional distributions) may be similar. In the following we offer a theoretical analysis and an empirical evaluation of PSF.

2 Theoretical Analysis

Let us consider a *semi-supervised setting* in which we are given the training data $\mathbf{X}^{tr}$, the corresponding labels $\mathbf{Y}^{tr}$ defining C classes, and the test data $\mathbf{X}^{tst}$, with the assumptions of covariate shift and periodic data structure. To devise a SF-based algorithm that performs CSA in this setting, we need to enrich the original SF algorithm with two additional properties: (i) the ability to generate

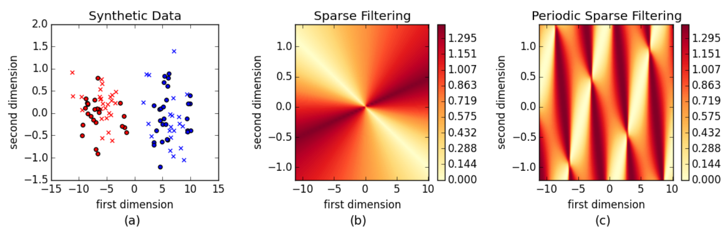

Figure 1: (a) Synthetic data: training samples in blue, test samples in red; positive-labeled samples as crosses, negative-labeled samples as dots. (b) Illustration of a filter instantiated by SF. (c) Illustration of a filter instantiated by PSF.

periodic filters through a sinusoidal non-linearity; and, (ii) the ability to capture the periodic structure underlying the data through the label information.

We thus propose a novel algorithm, called *periodic sparse filtering* (PSF) that defines a new transformation of the data $f : \mathbb{R}^O \to \mathbb{R}^L$ as $\mathbf{Z} = \ell_{2,col}\left(\ell_{2,row}\left(g\left(\mathbf{WX}\right)\right)\right)$, where $\mathbf{W}$ is a weight matrix, $\ell_{2,row}$ is the ℓ_2-normalization along the rows, $\ell_{2,col}$ is the ℓ_2-normalization along the columns, and $g(x)$ is a positive element-wise sinusoidal function, such as $1 + \sin(x)$. We can then prove the following theorem that guarantees that PSF does indeed generate periodic filters:

Theorem 1 *Let $\mathbf{X}^{(1)} \in \mathbb{R}^O$ be a point in the original space $\mathbb{R}^O$ and let $\mathbf{Z}^{(1)} \in \mathbb{R}^L$ be the representation learned by PSF. Then, there are infinite points $\mathbf{X}^{(i)} \in \mathbb{R}^O$ built from $\mathbf{X}^{(1)}$ with period $\mathbf{W}^{-1}\mathbf{k}\pi$, where $\mathbf{W}$ is the weight matrix of PSF and $\mathbf{k}$ is a vector of integer constants in $\mathbb{Z}$, that maps to the same representation $\mathbf{Z}^{(1)}$.*

Proof. See Section S.1 in the Supplementary Material for the proof.

To exploit the information carried by the labels we redefine the loss function as:

$$\mathcal{L}\left(\mathbf{W};\mathbf{X},\mathbf{Y}^{tr}\right) = \sum_{i=1}^{N}\sum_{j=1}^{L}\mathbf{Z}_j^{(i)} - \sum_{k=1}^{C}\lambda_k\mathbf{A}_{kk}, \tag{1}$$

where λ_k is a scaling factor and $\mathbf{A}_{ij}$ is a block sub-matrix of $\mathbf{Z}$ containing the i^{th} group of learned features from the samples belonging to the j^{th} class. The first term of the loss function pushes for learning sparse representations, while the second term pushes the mass of the sparse representations on the sub-matrices $\mathbf{A}_{kk}$, for $1 \leq k \leq C$. Minimizing the loss in (1) leads to our PSF algorithm (see Section S.2 in the Supplementary Material for the pseudo-code of the algorithm).

Overall, PSF can generate a periodic tiling of the original data space with a different frequency in each dimension and with a displacement driven by the labeled data.

3 Empirical validation

We empirically validate our proposed PSF algorithm with synthetic and real data sets.

Synthetic data. We generate a synthetic data set with the following properties: (i) easily visualizable; (ii) covariate shift affecting one dimension of the input; (iii) the labeling function being a periodic function defined over the dimension exhibiting covariate shift (see Figure 1(a)). More details on data generation can be found in Section S.3 in the Supplementary Material.

In our experiment, we train a SF and a PSF module, with learned dimensionality $L = 2$ (for visualization purposes) and with the scaling factors $\lambda_1 = \lambda_2 = 1$ (to keep all the loss terms in the same order of magnitude). We run off-the-shelf classification algorithms (linear SVM and RBF kernel SVM) on the original data as a baseline, and on the SF and the PSF representations for comparison.

We evaluate the results using following methods: (i) we visualize example filters learned by SF and PSF; (ii) we employ a Kolmogorov-Smirnov (KS) test to analyze the distribution of each feature [6], testing the hypotheses (p-value 0.05) that the features of training and test data come from different

Table 1: Synthetic data set: Kolmogorov-Smirnov statistical tests; classification accuracy on the training and test data set, and percentage drop using a linear SVM classifier and a RBF kernel SVM classifier.

	Kolmogorov-Smirnov		Linear SVM			RBF SVM		
Data	(train,test)	(positive,negative)	Training	Test	P-drop	Training	Test	P-drop
Raw	0.5	0.5	0.71 ± 0.03	0.51 ± 0.02	28.32 ± 1.39	0.98 ± 0.01	0.51 ± 0.02	48.02 ± 1.74
SF	0.0	0.0	0.60 ± 0.01	0.52 ± 0.01	13.21 ± 2.66	0.59 ± 0.01	0.51 ± 0.02	12.75 ± 3.10
PSF	0.3 ± 0.15	0.7 ± 0.15	0.75 ± 0.04	$\mathbf{0.72 \pm 0.05}$	4.10 ± 3.75	0.75 ± 0.04	$\mathbf{0.70 \pm 0.05}$	6.56 ± 4.13

distributions, and that the features for positive-labeled and negative-labeled data come from different distributions; (iii) we report the classification accuracy on the training and the test data, and we estimate cross-domain generalization with the metrics of percentage drop suggested in [11]. We report the mean and the standard error computed over ten simulations with randomly re-sampled data.

Figures 1(b) and 1(c) show prototypical filters learned by SF and PSF, respectively. For each point $\mathbf{X}^{(i)}$ in the original data space, the plot shows the distance of the learned representation $\mathbf{Z}^{(i)}$ from a perfect 1-sparse representation. As expected, SF instantiates filters with a conical shape centered in the origin of the original data space, while PSF generates periodic filters with arbitrary shape.

Table 1 reports the proportion of features exhibiting a different pdf according to the KS test. In general, the KS test evaluates only the pdf of individual features $p(Z_j)$, and it can not assess the pdf of the learned representation $p(Z)$. Due to the independence copula, however, we can actually assess the joint pdf of the learned representations $p(Z)$ from the marginal distribution of each feature $p(Z_j)$ in our experiment. For the raw data, the KS test easily detects covariate shift on one dimension ($\mathbf{X}_1$) and a difference in the distributions of positive-labeled and negative-labeled data via a single dimension ($\mathbf{X}_1$). For the SF representations, the KS test reveals that covariate shift adaptation takes place as the feature distributions of the training and the test data appear to be identical. Unfortunately, differences in the distribution of positive-labeled and negative-labeled data are lost. For the PSF representations, the results on the KS test suggest that our proposed PSF algorithm leads to sparse features that carry out covariate shift adaptation and retain the discriminative information defined in the labels simultaneously.

Table 1 also reports the performance in classification. For the raw data, the linear SVM returns a low performance as the data are not linearly separable. The kernel SVM reaches almost perfect discrimination on the training data, but it clearly overfits, and its performance on the test data is reduced to chance level due to covariate shift. For SF representations, classification accuracy is always set to chance level, since any difference between data of different labels vanished. For PSF representations, the classification accuracy is significantly higher than chance level, and the difference between the performance on the training data set and the test data set is remarkably small, as is evident from the percentage drop.

The experimental results suggest that PSF is able to provide representations that allow for learning the conditional distribution of the labels from the training data and for generalizing it to the test data.

Real data sets. For experiments with real data, we considered emotional speech data sets, which are highly regarded as a class of data affected by covariate shift [9]. We chose two emotional speech data sets widely used in the affective computing community: Berlin Emotional (EMODB) [3] and Vera am Mittag (VAM) [5]. EMODB is constituted of recordings of 10 German speakers, while VAM contains emotional utterances of 47 speakers participating in a German talk show. All the recordings are pre-processed into standard representations made up of 72-dimensional Mel-frequency cepstrum coefficient (MFCC) feature vectors [4]. Samples from EMODB are associated with a binary label denoting the presence or the absence of emotional content (1065 emotional samples, 146 non-emotional samples), while samples from VAM are provided with a binary label denoting a state of high or low arousal (1091 high-arousal samples, 1404 low-arousal samples). Underlying our experiments is the assumption that acoustic samples from each user are described by different distributions, but each distribution shows regularities in relation to emotional labeling.

We follow a protocol similar to the one described for synthetic data experiments. We train an SF and a PSF module, and we run the off-the-shelf linear SVM classifier on the raw data and on the learned representations. We set the learned dimensionality $L = 80$ (following the conservative decision of preserving the approximate dimensionality of the original samples); we also set the

Table 2: Real data set: UAR on the training and test data set, and percentage drop using a linear SVM classifier for the EMODB and the VAM data sets.

	Linear SVM			
	Data	Train	Test	P-drop
EMODB	Raw	0.60 ± 0.03	0.52 ± 0.01	11.86 ± 3.60
	SF	0.65 ± 0.01	0.52 ± 0.01	20.42 ± 2.03
	PSF	0.96 ± 0.002	0.52 ± 0.04	44.94 ± 3.93
	PSF + KS crit.	0.85 ± 0.01	$\mathbf{0.53 \pm 0.05}$	37.09 ± 5.22
VAM	Raw	0.55 ± 0.02	0.52 ± 0.02	5.96 ± 1.11
	SF	0.66 ± 0.003	0.57 ± 0.01	14.45 ± 1.96
	PSF	0.94 ± 0.001	0.53 ± 0.03	43.50 ± 1.83
	PSF + KS crit.	0.76 ± 0.005	$\mathbf{0.58 \pm 0.02}$	23.88 ± 4.88

dimensions of the block matrices $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ to 35 (to balance the learned features among the two classes). We set the other hyper-parameters (λ_1 and λ_2) to the optimal value found via cross-validation. For each data set, we keep all the samples of one speaker for validation and all the samples of another speaker for testing. Ten trials on each cross-validation configuration are run to increase the reliability of our results. Because of the high variance exhibited by PSF in cross-validation, we additionally devise a simple unsupervised criterion to retain half of the trials in which the distance between the distribution of training and test data is minimal: the trial t' is retained if the average distance between learned features computed using the KS test is smaller than the median of the average distance between learned features computed over all the trials t, i.e., $\left\{\frac{1}{L}\sum_{j=1}^{L} KS\left[\mathbf{Z}_j^{tr};\mathbf{Z}_j^{tst}\right]\right\}_{t'} < Median_t \left\{\frac{1}{L}\sum_{j=1}^{L} KS\left[\mathbf{Z}_j^{tr};\mathbf{Z}_j^{tst}\right]\right\}_t$.

Because of the high class imbalance, instead of the accuracy metric we use the unweighted average recall (UAR) [1]: $\frac{1}{K}\sum_{k=1}^{K} recall(k)$, where $recall(k)$ denotes the recall for class k and K is the total number of classes. We report the mean and the standard error computed over all the trials.

Table 2 contains the results of classification using a linear SVM. For the raw data, the covariate shift reduces the performance on the test data of the linear SVM close to chance level on both data sets. For the SF representations, the performance on the test data is close to chance level for EMODB, but higher for VAM. This may suggest the presence of different structures underlying the two data sets. For the PSF representations, classification performance is close or even lower than the performance of the standard SF algorithm. However, PSF representations have higher variance, as shown by the standard error. This suggests that PSF may find good solution, as well as unsatisfactory solutions. Adopting the criterion based on the KS test, we improve the final performance. The standard error is constant, but, since it is computed on half trials, it reveals a decrease in the variance. For EMODB, the overall increase in performance is limited, probably due to the difficulty of learning a periodic structure from a highly unbalanced data set. For VAM data set, PSF provides a clear improvement over raw representations and a moderate one over SF.

In summary, the above experimental results empirically verify our formal analysis and demonstrates that our PSF algorithm may be effective in learning sparse features for covariate shift adaptation.

4 Discussion

Our research suggests that PSF may be a suitable algorithm to perform CSA under a proper assumption about the structure of the data. Our theoretical analysis provides a firm ground to understand the behavior of PSF and it offers an insight on how to develop alternative FDL algorithms for CSA. Our experimental results verify our statements, despite more work is being carried out to address the problem of high variance when working with real-world data.

CSA has been studied widely in the literature [7] and the PSF dynamics show resemblances with *sinusoid filters* and *Walsh filters*, which are often used as bases for coding neurons [12]. However, to the best of our knowledge, no one studied the use of algorithms for sparse FDL to perform CSA.

In conclusion, we believe that FDL algorithms may be a promising avenue of research for CSA. Their simplicity, effectiveness and implicit insensitivity to the pdfs of the original data under covariate shift make them a strong candidate for CSA with high-dimensional data.

References

[1] A. Batliner, S. Steidl, D. Seppi, and B. Schuller. Segmenting into adequate units for automatic recognition of emotion-related episodes: a speech-based approach. *Advances in Human-Computer Interaction*, 2010(1):3–18, 2010.

[2] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 137–144, 2007.

[3] F. Burkhardt, A. Paeschke, M. Rolfes, W. F. Sendlmeier, and B. Weiss. A database of German emotional speech. In *Proceedings of the Interspeech Conference*, pages 1517–1520, 2005.

[4] F. Eyben, M. Wollmer, and B. Schuller. opensmile - the munich versatile and fast open-source audio feature extractor. In *Proceedings of the ACM International Conference on Multimedia*, 2010.

[5] M. Grimm, K. Kroschel, and S. Narayanan. The vera am mittag german audio-visual emotional speech database. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2008.

[6] A. Hassan, R. Damper, and M. Niranjan. On acoustic emotion recognition: compensating for covariate shift. *IEEE Trans. Audio, Speech, Language Process.*, 21(7):1458–1468, 2013.

[7] J. Jiang. A literature survey on domain adaptation of statistical classifiers. Technical report, University of Illinois at Urbana-Champaign, 2008.

[8] J. Ngiam, Z. Chen, S. A. Bhaskar, P. W. Koh, and A. Y. Ng. Sparse filtering. In *Advances in Neural Information Processing Systems*, pages 1125–1133, 2011.

[9] B. Schuller, B. Vlasenko, F. Eyben, M. Wollmer, A. Stuhlsatz, A. Wendemuth, and G. Rigoll. Cross-corpus acoustic emotion recognition: variances and strategies. *IEEE Trans. Affect. Comput.*, 1:119–131, 2010.

[10] M. Sugiyama and M. Kawanabe. *Machine learning in non-stationary environments: introduction to covariate shift adaptation*. MIT Press, 2012.

[11] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1521–1528, 2011.

[12] B. Willmore and D. J. Tolhurst. Characterizing the sparseness of neural codes. *Network: Computation in Neural Systems*, 12(3):255–270, 2001.

[13] F. M. Zennaro and K. Chen. Towards understanding sparse filtering: A theoretical perspective. *arXiv preprint arXiv:1603.08831*, 2016.

Supplementary Materials: Covariate Shift Adaptation via Sparse Filtering for High-Dimensional Periodic Data

S.1 Proof of Theorem 1

Theorem 2 *Let $\mathbf{X}^{(1)} \in \mathbb{R}^O$ be a point in the original space $\mathbb{R}^O$ and let $\mathbf{Z}^{(1)} \in \mathbb{R}^L$ be the representation learned by PSF. Then, there are infinite points $\mathbf{X}^{(i)} \in \mathbb{R}^O$ built from $\mathbf{X}^{(1)}$ with period $\mathbf{W}^{-1}\mathbf{k}\pi$, where $\mathbf{W}$ is the weight matrix of PSF and $\mathbf{k}$ is a vector of integer constants in $\mathbb{Z}$, that maps to the same representation $\mathbf{Z}^{(1)}$.*

Proof. The proof of this proposition is based on the following logical steps: (a) rigorous definition of the PSF computation; (b-e) back-computation through all the steps of PSF up to the input (ℓ_2-normalization along the columns, ℓ_2-normalization along the rows, non-linearity, linear projection).

(a) Let $\mathbf{X}^{(1)} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_O^{(1)} \end{bmatrix}^T$ and $\mathbf{X}^{(2)} = \begin{bmatrix} x_1^{(2)} & x_2^{(2)} & \cdots & x_O^{(2)} \end{bmatrix}^T$ be two points in the original space $\mathbb{R}^O$ and let $\mathbf{Z}^{(1)} = PSF\left(\mathbf{X}^{(1)}\right) = \begin{bmatrix} z_1^{(1)} & z_2^{(1)} & \cdots & z_L^{(1)} \end{bmatrix}^T$ and $\mathbf{Z}^{(2)} = PSF\left(\mathbf{X}^{(2)}\right) = \begin{bmatrix} z_1^{(2)} & z_2^{(2)} & \cdots & z_L^{(2)} \end{bmatrix}^T$ be the two representations in $\mathbb{R}^L$ learned by PSF. Recall that PSF is defined as $PSF(\mathbf{X}) = \ell_{2,col}\left(\ell_{2,row}\left(g\left(\mathbf{WX}\right)\right)\right)$, where we take the non-linearity to be an element-wise positive sinusoidal function, that is $g(x) = 1 + \epsilon + \sin(x)$.

Now, let us suppose that the two learned representations are identical, that is $\mathbf{Z}^{(1)} = \mathbf{Z}^{(2)}$.

(b) By definition of PSF, $\mathbf{Z}^{(1)} = \mathbf{Z}^{(2)}$ implies:

$$\begin{aligned} \ell_{2,col}\left(\tilde{\mathbf{F}}^{(1)}\right) &= \ell_{2,col}\left(\tilde{\mathbf{F}}^{(2)}\right) \\ \frac{\tilde{\mathbf{F}}_j^{(1)}}{\sqrt{\sum_{j=1}^{L}\left(\tilde{\mathbf{F}}_j^{(1)}\right)^2}} &= \frac{\tilde{\mathbf{F}}_j^{(2)}}{\sqrt{\sum_{j=1}^{L}\left(\tilde{\mathbf{F}}_j^{(2)}\right)^2}}, \end{aligned}$$

where $\tilde{\mathbf{F}}^{(i)} = \begin{bmatrix} \tilde{f}_1^{(i)} & \tilde{f}_2^{(i)} & \cdots & \tilde{f}_L^{(i)} \end{bmatrix}^T$ is the intermediate output of PSF defined as $\tilde{\mathbf{F}} = \ell_{2,row}\left(g\left(\mathbf{WX}\right)\right)$. Now, for the ℓ_2-normalization along the columns to be equal, it must hold that:

$$\begin{bmatrix} \frac{\tilde{f}_1^{(1)}}{d^{(1)}} & \frac{\tilde{f}_2^{(1)}}{d^{(1)}} & \cdots & \frac{\tilde{f}_L^{(1)}}{d^{(1)}} \end{bmatrix}^T = \begin{bmatrix} \frac{\tilde{f}_1^{(2)}}{d^{(2)}} & \frac{\tilde{f}_2^{(2)}}{d^{(2)}} & \cdots & \frac{\tilde{f}_L^{(2)}}{d^{(2)}} \end{bmatrix}^T,$$

where $d^{(i)} = \sqrt{\sum_{j=1}^{L}\left(\tilde{\mathbf{F}}_j^{(i)}\right)^2}$ is a sample-dependent scaling factor. Therefore, it follows that $\mathbf{Z}^{(1)} = \mathbf{Z}^{(2)}$ if and only if $\tilde{\mathbf{F}}^{(1)} = \lambda\tilde{\mathbf{F}}^{(2)}$ for any $\lambda \in \mathbb{R}$.

(c) By definition of PSF, $\tilde{\mathbf{F}}^{(1)} = \lambda\tilde{\mathbf{F}}^{(2)}$ implies:

$$\ell_{2,row}\left(\mathbf{F}^{(1)}\right) = \lambda\ell_{2,row}\left(\mathbf{F}^{(2)}\right)$$

$$\begin{aligned} \frac{\mathbf{F}_j^{(1)}}{\sqrt{\sum_{i=1}^{N}\left(\mathbf{F}_j^{(i)}\right)^2}} &= \lambda \frac{\mathbf{F}_j^{(2)}}{\sqrt{\sum_{i=1}^{N}\left(\mathbf{F}_j^{(i)}\right)^2}}, \end{aligned}$$

where $\mathbf{F}^{(i)} = \left[\begin{array}{cccc} f_1^{(i)} & f_2^{(i)} & \cdots & f_L^{(i)} \end{array}\right]^T$ is the intermediate output of PSF defined as $\mathbf{F} = g\left(\mathbf{WX}\right)$. Now, for the ℓ_2-normalization along the rows to be equal, it must hold that:

$$\left[\begin{array}{cccc} \frac{f_1^{(1)}}{d_1} & \frac{f_2^{(1)}}{d_2} & \cdots & \frac{f_L^{(1)}}{d_L} \end{array}\right]^T = \lambda \left[\begin{array}{cccc} \frac{f_1^{(2)}}{d_1} & \frac{f_2^{(2)}}{d_2} & \cdots & \frac{f_L^{(2)}}{d_L} \end{array}\right]^T,$$

where $d_j = \sqrt{\sum_{i=1}^{N}\left(\mathbf{F}_j^{(i)}\right)^2}$ is a feature-dependent scaling factor. Therefore, it follows that $\tilde{\mathbf{F}}^{(1)} = \lambda\tilde{\mathbf{F}}^{(2)}$ if and only if $\mathbf{F}^{(1)} = \mathbf{F}^{(2)}$.

(d) By definition of PSF, $\mathbf{F}^{(1)} = \lambda\mathbf{F}^{(2)}$ implies:

$$\begin{aligned} g\left(\mathbf{H}^{(1)}\right) &= g\left(\mathbf{H}^{(2)}\right) \\ 1 + \epsilon + \sin\left(\mathbf{H}^{(1)}\right) &= 1 + \epsilon + \sin\left(\mathbf{H}^{(2)}\right) \\ \sin\left(\mathbf{H}^{(1)}\right) &= \sin\left(\mathbf{H}^{(2)}\right), \end{aligned}$$

where $\mathbf{H}^{(i)} = \left[\begin{array}{cccc} h_1^{(i)} & h_2^{(i)} & \cdots & h_L^{(i)} \end{array}\right]^T$ is the intermediate output of PSF defined as $\mathbf{H} = \mathbf{WX}$. Now, for the applications of the sinusoidal function to be equal, it must hold that:

$$\begin{aligned} \sin\left(\mathbf{H}^{(1)}\right) &= \sin\left(\mathbf{H}^{(2)}\right) \\ \mathbf{H}^{(1)} &= \arcsin\left(\sin\left(\mathbf{H}^{(2)}\right)\right) \\ \mathbf{H}^{(1)} &= \mathbf{H}^{(2)} + \mathbf{k}\pi, \end{aligned}$$

where $\mathbf{k}_j$ is a vector of feature-dependent periodic factors in $\mathbb{Z}$.

(e) By definition of PSF, $\mathbf{H}^{(1)} = \mathbf{H}^{(2)} + \mathbf{k}\pi$ implies:

$$\begin{aligned} \mathbf{WX}^{(1)} &= \mathbf{WX}^{(2)} + \mathbf{k}\pi \\ \mathbf{X}^{(1)} &= \mathbf{X}^{(2)} + \mathbf{W}^{-1}\mathbf{k}\pi. \end{aligned}$$

Thus, there are infinite points $\mathbf{X}^{(i)} \in \mathbb{R}^O$ built from $\mathbf{X}^{(1)}$ with period $\mathbf{W}^{-1}\mathbf{k}\pi$ that maps to the same representation $\mathbf{Z}^{(1)}$. ■

S.2 Pseudo-code of the PSF algorithm

Below we provide the pseudo-code for the PSF algorithm [1].

Algorithm 1 Periodic Sparse Filtering (PSF)

```
 1: Input: training data $\mathbf{X}^{tr}$, test data $\mathbf{X}^{tst}$, training label $\mathbf{Y}^{tr}$
 2: Hyper-params: learned dimensionality $L$, partitioning vector $\mathbf{V}$, lambda vector $\lambda$

 3: $\mathbf{X} \leftarrow \mathbf{X}^{tr} \cup \mathbf{X}^{tst}$
 4: $\mathbf{W} \leftarrow \mathcal{N}(0,1)$
 5: $C \leftarrow$ #classes in $\mathbf{Y}^{tr}$

 6: repeat
 7:     $\mathbf{H} \leftarrow \mathbf{W} \cdot \mathbf{X}$
 8:     $\mathbf{F} \leftarrow 1 + \epsilon + \sin H$
 9:     $\tilde{\mathbf{F}} \leftarrow \frac{\mathbf{F}_j^{(i)}}{\sqrt{\sum_{i=1}^{N} \left(\mathbf{F}_j^{(i)}\right)^2}}$
10:     $\mathbf{Z} \leftarrow \frac{\tilde{\mathbf{F}}_j^{(i)}}{\sqrt{\sum_{j=1}^{L} \left(\tilde{\mathbf{F}}_j^{(i)}\right)^2}}$
11:     $\mathcal{L} \leftarrow \sum_{i=1}^{N} \sum_{j=1}^{L} \mathbf{Z}_j^{(i)} + \sum_{k=1}^{C} \sum_{i:\mathbf{X}^{(i)} \in \mathbf{X}^{tr} \wedge \mathbf{Y}^{(i)}=k} \sum_{j:j \in \mathbf{V}^k} \lambda_k \cdot \mathbf{Z}_j^{(i)}$

12:     $\mathbf{W} \leftarrow \mathbf{W} - \eta \nabla \mathcal{L}$
13: until termination condition for gradient descent is met
    return $\mathbf{Z}$
```

Algorithm 2 Derivative for PSF

```
1: Input: PSF output $\mathbf{Z}$

2: $\frac{\partial \mathbf{Z}}{\partial \tilde{\mathbf{F}}}{}_j^{(i)} \leftarrow \frac{\sqrt{\sum_{j=1}^{L} \left(\tilde{\mathbf{F}}_j^{(i)}\right)^2} - \mathbf{Z}_j^{(i)} \cdot \sum_{j=1}^{L} \tilde{\mathbf{F}}_j^{(i)}}{\sum_{j=1}^{L} \left(\tilde{\mathbf{F}}_j^{(i)}\right)^2}$
3: $\frac{\partial \mathbf{Z}}{\partial \mathbf{F}}{}_j^{(i)} \leftarrow \frac{\frac{\partial \mathbf{Z}}{\partial \tilde{\mathbf{F}}}{}_j^{(i)} \sqrt{\sum_{i=1}^{N} \left(\mathbf{F}_j^{(i)}\right)^2} - \tilde{\mathbf{F}}_j^{(i)} \cdot \sum_{i=1}^{N} \left(\frac{\partial \mathbf{Z}}{\partial \tilde{\mathbf{F}}}{}_j^{(i)} \cdot \mathbf{F}_j^{(i)}\right)}{\sum_{i=1}^{N} \left(\mathbf{F}_j^{(i)}\right)^2}$
4: $\frac{\partial \mathbf{Z}}{\partial \mathbf{H}} \leftarrow \frac{\partial \mathbf{Z}}{\partial \mathbf{F}} \cdot \cos \mathbf{H}$
5: $\frac{\partial \mathbf{Z}}{\partial \mathbf{W}} \leftarrow \lambda \frac{\partial \mathbf{Z}}{\partial \mathbf{H}} \cdot \mathbf{X}$
   return $\frac{\partial \mathbf{Z}}{\partial \mathbf{W}}$
```

[1] The source code is available at: `https://github.com/EldarFeatel/PSF`

S.3 Detailed presentation of the experiments on the synthetic data set

In our *synthetic data set* simulation we generated a data set that would capture in a simplified way the case of user-dependent data: different users have distributions located far apart in the feature space, but their data exhibit local regularities. Training data and test data are then sampled from two bivariate Gaussian pdfs and they are labeled in a binary way by a deterministic function. For the training data set we generated 50 samples from $\mathbf{X}^{tr} \sim \mathcal{N}\left(\begin{bmatrix} 2\pi \\ 2 \end{bmatrix}, \begin{bmatrix} 2 & 0 \\ 0 & .5 \end{bmatrix}\right)$ and for the test data set we generated 50 samples from $\mathbf{X}^{tst} \sim \mathcal{N}\left(\begin{bmatrix} -2\pi \\ 2 \end{bmatrix}, \begin{bmatrix} 2 & 0 \\ 0 & .5 \end{bmatrix}\right)$. Binary labels over the training data and the test data are defined by a deterministic square wave function with period 1 on the domain of the first feature. Figure 2 shows the synthetic data in two dimensions. Figure 3 shows the projection of the synthetic data along the first dimension, that is, the dimension affected by covariate shift. Each figure shows separately positive-labeled training data (blue crosses), negative-labeled training data (blue dots), positive-labeled test data (red crosses), and negative-labeled test data (red dots); moreover, it also shows the empirical and the real (where possible) distributions of the training, test, positive-labeled and negative-labeled data.

Figure 4 shows actual filters instantiated by SF and PSF in relation to the data in the synthetic data set simulation. The figure confirms that PSF is able to learn filters that better explain the data.

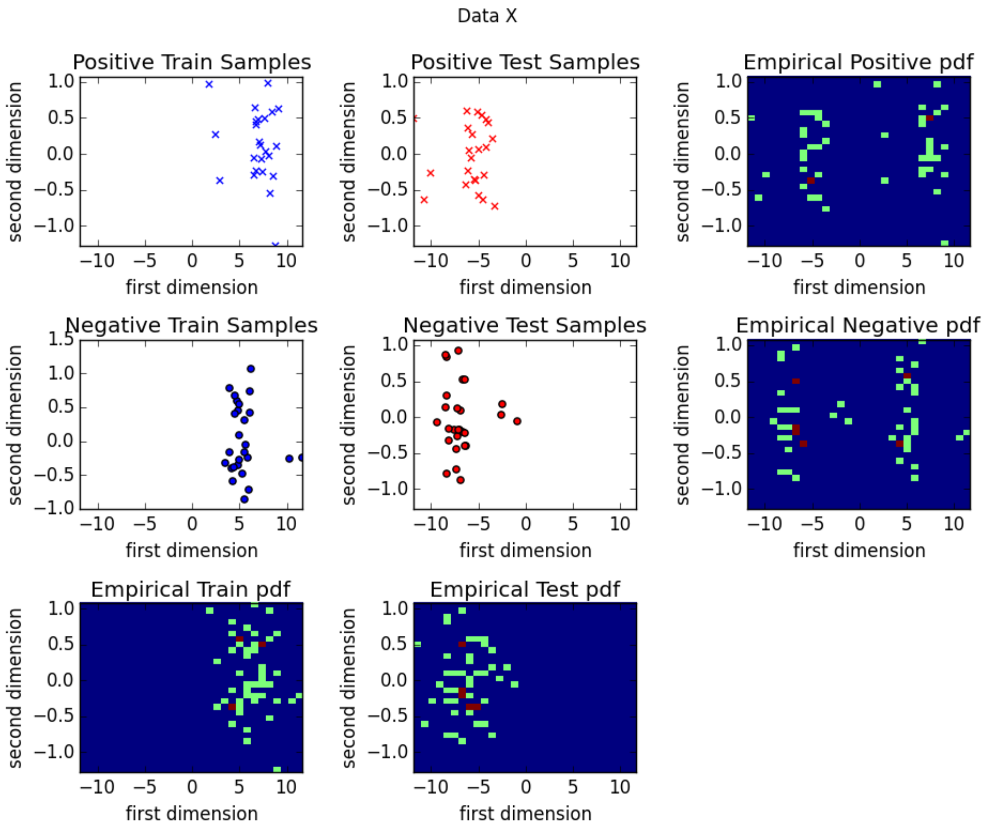

Figure 2: Synthetic data $\mathbf{X}$ used in the simulation.

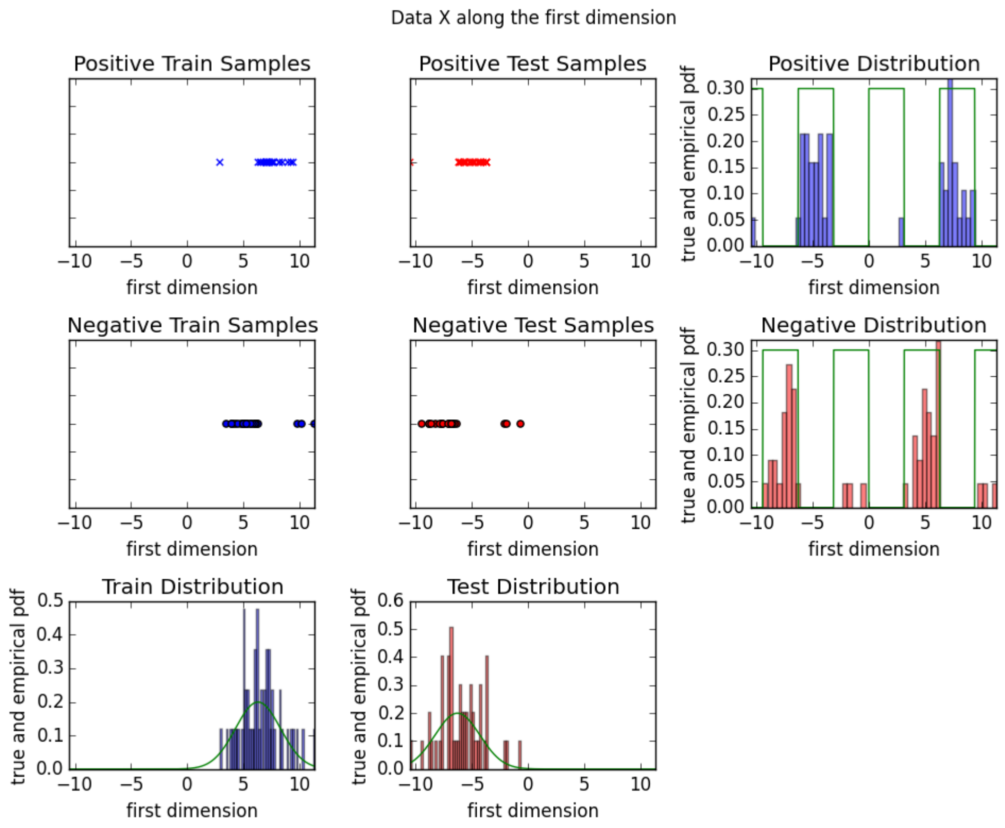

Figure 3: Projection of the synthetic data $\mathbf{X}$ used in the simulation along the first dimension.

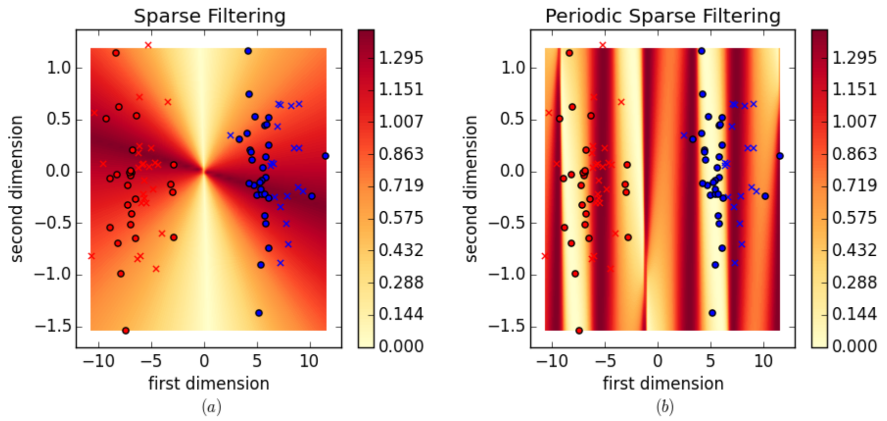

Figure 4: (a) Illustration of a filter instantiated by SF. (b) Illustration of a filter instantiated by PSF.