Random Walk Kernel Applications to Classification using Support Vector Machines

Vasileios Gavriilidis, Anastasios Tefas
Department of Informatics, Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
Email: {vgavril,tefas}@aiia.csd.auth.gr

***Abstract*—Kernel Methods are algorithms that are widely used, mainly because they can implicitly perform a non–linear mapping of the input data to a high dimensional feature space. In this paper, novel Kernel Matrices, that reflect the general structure of data, are proposed for classification. The proposed Matrices exploit properties of the graph theory, which are generated using power iterations of already known Kernel Matrices and three approaches are presented. Experiments on various datasets are conducted and statistical tests are performed, comparing our proposed approach against current Kernel Matrices used on support vector machines. Also, experiments on real datasets for folk dance and activity recognition that highlight the superiority of our proposed method, are provided.**

I. Introduction

In machine learning, similarities or dissimilarities are used by many algorithms. In computer vision, tangent distance [1], earth mover's distance [2], shape matching distance [3] pyramid match kernel [4] are all algorithms that use similarities between images, which, in turn, are used for image retrieval and object recognition. Likewise, in bioinformatics there are popular algorithms used to compute similarities for protein classification [5], [6]. For more details on similarity based classification refer to [7].

Similarity matrices can be created by applying any known metric to data. Most common metrics used are inner product and cosine similarity. Moreover, a similarity matrix $\mathbf{S}$ can be used as a Kernel Matrix $\mathbf{K}$ by simply replacing $\mathbf{K}$ with $\mathbf{S}$, ignoring the fact that $\mathbf{S}$ might be indefinite. This is not always correct, therefore, there are many sophisticated methods to modify $\mathbf{S}$ for it to be positive definite such as denoise, flip, diffuse and shift. A detailed discussion about transforming indefinite matrices into Kernels can be found in [8].

Kernel methods [9], [10] have received increased attention, particularly due to the popularity of the Support Vector Machines. Kernel functions can be used in many applications as they provide a transformation from linearity to non-linearity and can be expressed in terms of dot products, using the kernel trick. The most common kernels are the linear kernel and the Gaussian kernel, which is a non-linear kernel.

Another representation, based on pairwise similarities between samples, is a graph. Graphs provide a general sense of similarity between objects that reflect the whole structure of data. For example in dimensionality reduction, often, a k–nearest neighbour graph is created [11], [12]. However, this operation on graphs is not applicable to classification algorithms that use all pairwise distances, such as SVM. Ideally, we would like to have a fully connected graph, so that each pair of samples is connected and has a similarity value.

In addition, graphs have many properties and are used widely [13]. One of the properties of a graph, that we will use in this paper, is that if W represents the adjacency matrix between nodes, where $W(i,j) = 1$ if nodes i and j are connected and $W(i,j) = 0$ otherwise, then p–th power of adjacency matrix, $W^p(i,j)$, gives the number of paths of length p between nodes i and j. This notion can be applied to either directed or undirected graphs and can also be extended to weighted graphs, $W(i,j) \in [0, \inf]$.

We propose applying the aforementioned property of a graph to both linear and nonlinear Kernel matrices for classification, using Support Vector Machines as classifier.

The structure of the paper is organised as follows: In section II, we state the problem we solve. We, then, introduce our method for classification in section III, providing some theoretical background, and then, we describe our three proposed approaches. In section IV, we explain the way we conducted our experiments and present classification results to various datasets. We also provide classification results to activity recognition based datasets. Finally, we give concluding remarks and discussion of future work in section V.

II. Problem Statement

Assume a supervised binary–class, classification problem with data matrix, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$, where $\mathbf{x}_i \in \Re^D$ and $y_i \in \{-1, 1\}$ indicates the class to which the sample $\mathbf{x}_i$ belongs, where $i = 1, \ldots, N$. N is the number of samples and D is the dimension of data. Suppose we are given pairwise similarities of the samples in kernel matrix $\mathbf{K}$.

The SVM is a binary classifier based on structural risk minimization [14]. From [15] the definition of the quadratic programming problem for support vector learning can be expressed as:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i}^{N} \sum_{j}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (1)$$

$$\text{s.t.} \quad 0 \leqslant \alpha_i \leqslant C \quad \forall i$$

$$\sum_{i=1}^{N} y_i \alpha_i = 0,$$

where C is a penalty parameter. Notice that samples do not participate in maximizing equation (1); only pairwise similarities estimated by the kernel between samples are necessary. By exploiting the kernel trick [9] we can use a non–linear function $\Phi(\mathbf{x}_i)$ to represent the samples in a higher dimensional space, where they can be linearly separable.

III. Proposed Method

There are various kernel functions that can be used - most common are linear, polynomial, RBF and sigmoid. Moreover, there are other kind of kernels, such as random walk kernel, which was first proposed in [16] and was later better defined as kernel matrix for semi–supervised learning using cluster kernels in [17].

Random walk kernel based on [17] is computed in two steps. First, RBF kernel matrix is computed and then, each value is normalised by the sum of its row. The resulted matrix is a transition matrix of a random walk on a graph, that defines the probabilities of starting from one point and arriving at another. Using this approach, a diagonal matrix which is defined as $\mathbf{D}_{ij} = \sum_j \mathbf{K}_{ij}$ can be used to construct the transition matrix as $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$, thus the matrix $\mathbf{P}^p = (\mathbf{D}^{-1}\mathbf{K})^p$ suggests transition probability after p steps. Unfortunately, matrix $\mathbf{P}^p$ is not symmetric, hence it can not be used as a kernel for a SVM classifier, but a trick is provided in [17] to transform it to a positive definite matrix.

In addition, a family of kernels on graphs, based on the notion of regularization operators, are defined in [18], where another example of a random walk kernel is introduced. Let an unweighted graph G consist of a set of vertices numbered from 1 to N that represent the data matrix and its adjacency matrix $\mathbf{W}$, with $W_{ij} = 1$ if samples i and j are neighbours. Let $\mathbf{D}$ be the same diagonal matrix as before. The Laplacian of G is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$ and the Normalised Laplacian is $\tilde{\mathbf{L}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$. A kernel matrix, the p–step random walk kernel, is introduced and is computed as $\mathbf{K} = (a\mathbf{I} - \tilde{\mathbf{L}})^p$, with $a \geqslant 2$ in [18]. Keep in mind that in general, $\mathbf{W}$ is not restricted and can be extended to weighted graphs allowing $W_{ij} \in [0, \inf]$. Also, notice that parameter a ensures positive definiteness of $\mathbf{K}$.

As discussed, random walk kernels are already known, however, no classification performance results have ever been published. A possible reason may be the difficulty in determining the selection of p and positive definiteness, which requires tricks in order to be used as a kernel to a SVM. We now present how a random walk based kernel can be used to classification, using a SVM classifier.

Our approach requires a known kernel to be computed is in classification using precomputed kernels with a SVM. In this paper, we will use a linear kernel, more specifically the inner product, and a non linear kernel, in our case the RBF kernel, however, any known kernel could be used.

A. Linear Kernel

A similarity matrix $\mathbf{W}$, expressing the similarity between i-th and j-th sample, can be defined as the inner product:

$$W(i,j) = \mathbf{x}_i^T \mathbf{x}_j. \tag{2}$$

Let i–th and j–th samples be represented as nodes in an unweighted graph with $W(i,j) = 0$ meaning samples are not similar and $W(i,j) = 1$ meaning samples are similar.

We may now propose the similarity matrix:

$$\mathbf{W}^p = \underbrace{\mathbf{W}\mathbf{W}\dots\mathbf{W}}_{\text{p times}}. \tag{3}$$

We can say that $W^p(i,j)$ expresses the similarity between i–th and j–th samples after visiting all possible paths passing from $p-1$ in–between similar samples.

Extending the discrete values of similar and not similar (0 and 1) to continuous values, we define a relaxed definition of a weighted graph which can take values in $[0 - \inf]$, where 0 is the least similar and $\inf$ is the most similar. This way, when two samples' similarity is computed, more paths are available, since the only paths that are not viable are those that pass from an intermediate sample that has zero similarity value. In reality, every single path is involved because even though the similarity of two samples can be small, it is rarely zero. For example, the similarity matrix passing from one intermediate sample can be computed as $\mathbf{W}^2 = \mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X}$.

Without loss of generality, let's assume that data matrix has zero mean, hence $\mathbf{X}\mathbf{X}^T = \mathbf{\Sigma}$, where $\mathbf{\Sigma}$ is the covariance matrix, thus $\mathbf{W}^2 = \mathbf{X}^T\mathbf{\Sigma}\mathbf{X}$ also holds. Moreover, it is straightforward to show that $\mathbf{W}^p = \mathbf{X}^T\mathbf{\Sigma}^{p-1}\mathbf{X}$, with $p \geqslant 1$.

So, $\mathbf{W}^p$ is a similarity matrix and $W^p(i,j)$ expresses the similarity of two samples beginning from the i–th sample and ending at the j–th sample after passing through $p-1$ intermediate samples. The aim is that for similar nodes to be connected by several paths. Even if $\mathbf{W}^p$ is a similarity matrix, this does not necessarily mean that it can be used as a kernel matrix. We now prove that apart from $\mathbf{W}$, which is by definition a kernel matrix, $\mathbf{W}^p$ is also a kernel matrix.

It is safe to replace $\mathbf{W}$ by $\mathbf{K}$ since $\mathbf{K}$ is positive definite. $\mathbf{K}$ has an eigenvalue decomposition $\mathbf{K} = \mathbf{U}^T\mathbf{\Lambda}\mathbf{U}$, where $\mathbf{U}$ is an orthogonal matrix and $\mathbf{\Lambda}$ is a diagonal matrix of real and positive eigenvalues, that is, $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, \dots, \lambda_D)$. So, now, $\mathbf{W}^p$ can be written as:

$$\begin{aligned}
\mathbf{W}^p &= \underbrace{\mathbf{K}\mathbf{K}\dots\mathbf{K}}_{\text{p times}} \\
&= \mathbf{U}^T\mathbf{\Lambda}\underbrace{\mathbf{U}\mathbf{U}^T}_{I}\mathbf{\Lambda}\underbrace{\mathbf{U}\mathbf{U}^T}_{I}\dots\underbrace{\mathbf{U}\mathbf{U}^T}_{I}\mathbf{\Lambda}\mathbf{U} \\
&= \mathbf{U}^T\underbrace{\mathbf{\Lambda}\dots\mathbf{\Lambda}}_{\text{p times}}\mathbf{U} \\
&= \mathbf{U}^T\mathbf{\Lambda}^p\mathbf{U}.
\end{aligned} \tag{4}$$

Since eigenvalues $\lambda_i > 0, \quad \forall i = 1, \dots, N$ then $\lambda_i^p > 0$, $\forall i = 1, \dots, N$, which leads to $\mathbf{x}\mathbf{W}^p\mathbf{x}^T \geq 0, \forall \mathbf{x}$, which is the definition of a positive definite matrix. Notice that no

assumptions were made for the original kernel matrix. Thus, in general, every kernel matrix elevated to any power is also a kernel matrix.

Now, $\mathbf{W}^p$ can safely be used as a precomputed kernel matrix in SVM. Moreover, when inner product is used as the initial kernel matrix, assuming data have zero mean, we arrive at an interesting property. The covariance matrix, $\mathbf{\Sigma}$, is symmetric and has real values, so it has an eigenvalue decomposition that can be written as:

$$\mathbf{\Sigma} = \mathbf{U}\mathbf{D}\mathbf{U}^T. \tag{5}$$

Hence:

$$\begin{aligned}\mathbf{W}^p =& \mathbf{X}^T\mathbf{\Sigma}^{p-1}\mathbf{X}\\ =& \mathbf{X}^T\mathbf{U}\mathbf{D}^{p-1}\mathbf{U}^T\mathbf{X}\\ =& (\mathbf{D}^{\frac{p-1}{2}}\mathbf{U}^T\mathbf{X})^T(\mathbf{D}^{\frac{p-1}{2}}\mathbf{U}^T\mathbf{X}).\end{aligned} \tag{6}$$

So, kernel $\mathbf{W}^p$ can be applied differently by transforming the data. Multiplying data matrix, $\mathbf{X}$, by $\mathbf{D}^{\frac{p-1}{2}}\mathbf{U}^T$ and using inner product with transformed data, yields the same results as when using original data and $\mathbf{W}^p$. Now, let's examine another initial kernel matrix.

B. Non Linear Kernel

Another popular kernel is RBF kernel which is defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{|\mathbf{x}_i - \mathbf{x}_j|^2}{2\sigma^2}}, \tag{7}$$

and $K^p(\mathbf{x}_i, \mathbf{x}_j), p \geqslant 1$ can be expressed as:

$$K^p(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l_1}^{N} \cdots \sum_{l_{p-1}}^{N} e^{-\frac{\left|\mathbf{x}_i - \mathbf{x}_{l_1}\right|^2 + \cdots + \left|\mathbf{x}_{l_{p-1}} - \mathbf{x}_j\right|^2}{2\sigma^2}}. \tag{8}$$

Examining equation (8) we see that the distance between two nodes is relative to the whole structure of the graph, since, in order to compute the distance of two nodes, all the nodes of the graph are taken into account, which resembles a graph based distance, and then transforming it to similarity accordingly.

C. Available approaches

Kernel based classifiers, like SVM, use the similarities of each pair of samples to train and then to test, rather than using the original space in which the samples belong. In order to use the precomputed kernel in SVM, we must take into account that training and test samples are treated differently. Assume kernel matrix $\mathbf{K}$ consists of similarities of all data. Hence its size is $(N_T + N_t) \times (N_T + N_t)$ where N_T is the number of samples for training purposes, and N_t is the number of test samples of which their class needs to be predicted. Also, assume that samples of pairwise similarities inside $\mathbf{K}$ are in the following order: training samples and then test samples, so now, using submatrices, $\mathbf{K}$ can be defined as:

$$\mathbf{K} = \begin{bmatrix}\mathbf{A} & \mathbf{C}^T\\ \mathbf{C} & \mathbf{D}\end{bmatrix}. \tag{9}$$

The matrix $\mathbf{A}$ $(N_T \times N_T)$ is used for building a model that satisfies equation (1) and the similarity matrix $\mathbf{C}$ $(N_t \times N_T)$, which shows the similarities between test samples and training samples, is used for predicting the class of test samples. That being said, one could wonder how to treat samples when a kernel is elevated to some power. We may now propose three ways to treat test samples, each one based on the fact that we can control the available similarity paths, and each way treats samples differently.

1) Pass through All (PA): In this approach, we assume that all data are available, with some of them being treated as training samples, while others as test samples. Every time a new test sample's class needs to be predicted, we train a new model based on all samples, including the test samples, by computing $\mathbf{K}^p$. Then, we choose the submatrix of $\mathbf{K}^p$ that consists of the first N_T rows and first N_T columns to be used, in order to build the SVM model. Finally, the matrix used to predict the test samples' class, consists of the rows $[N_T + 1, \ldots, N_T + N_t]$ and first N_T columns.

2) Pass through Train (PT): This is similar to (PA), only that this approach is more flexible, since the kernel matrix used for training does not require test samples in order to be computed. First, we define another matrix, the restriction kernel matrix $\mathbf{K}_r$, which is computed as $\mathbf{K}$ is computed, but replacing the rows of $\mathbf{K}$ corresponding to test samples with zeros. This way, we allow similarity paths that can pass exclusively through training samples. Then $\mathbf{K}_r^{p-1}$ is computed and finally $\mathbf{K}\mathbf{K}_r^{p-1}$ is used. In the matrix that consists of first N_T rows and first N_T columns of $\mathbf{K}\mathbf{K}_r^{p-1}$, which is used for the training procedure, test samples are not involved, this way computing the power of the kernel is ready to be used for out-of-samples classification and thus is more flexible than (PA).

3) Pass through Support Vectors (PSV): This is a more sophisticated approach and requires one additional step. Using a kernel matrix which consists of only the similarities between training samples, a SVM model is created. The next step is to treat the training samples that were support vectors differently from those samples that were not. The difference is that only similarity paths passing through support vectors are computed. Similarly to (PT), the restriction matrix is computed by replacing rows of the test samples as well as the rows of training samples that were not support vectors with zeros. Notice that, even though non support vectors do not contribute to similarity paths, they are taken into account when the labels of test samples need to be predicted.

IV. Experiments

In this section we show the results from various datasets, taken from UCI [19]. The last three datasets were artificially created, as well as Balance1 and Balance3 whose method of creation will be explained later on. The method used to perform our experiments is as follows: we split each dataset into train and test samples by doing 5×2 cross validation. The final test classification result is computed by averaging

TABLE I: Inner product – Various Datasets

Dataset	Samples	Dimensions	Classes	Inner Product - Grid Search				Inner Product - Best power			
				simple	PA	PT	PSV	simple	PA	PT	PSV
australian	690	14	2	85.42	**85.45**	85.42	85.39	85.42	85.51	85.51	**85.62**
diabetes	768	8	2	**76.88**	**76.88**	**76.88**	**76.88**	**76.88**	76.80	76.80	76.80
liver-dis	345	6	2	68.00	**68.17**	67.94	66.78	68.00	**68.69**	**68.69**	**68.69**
mamographic	961	5	2	**81.50**	**81.50**	**81.50**	**81.50**	81.50	**82.02**	**82.02**	**82.02**
PlanRelax	182	12	2	**71.43**	**71.43**	**71.43**	**71.43**	**71.43**	**71.43**	**71.43**	**71.43**
habberman	306	3	2	**73.46**	**73.46**	**73.46**	**73.46**	73.46	**73.53**	**73.53**	**73.53**
Iris	150	4	3	96.80	**97.20**	**97.20**	96.40	96.80	**98.00**	**98.00**	97.73
Balance1	400	2	3	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Balance	625	4	3	**91.62**	90.85	91.23	**91.62**	**91.62**	91.36	91.36	91.36
Balance3	729	6	3	**93.42**	**93.42**	**93.42**	**93.42**	**93.42**	**93.42**	**93.42**	**93.42**
Ecoli	336	7	8	86.14	86.08	86.14	**86.14**	**86.14**	**86.50**	**86.50**	**86.50**
halfmoon	800	2	2	**83.83**	83.58	83.58	83.47	**83.83**	83.75	83.75	83.75
cocentral	685	2	3	50.80	66.10(*)	65.96(*)	**66.51**(*)	50.80	66.10(*)	66.10(*)	**66.57**(*)
swissroll	1600	3	4	75.66	**77.01**	**77.01**	76.85	75.66	**77.45**	**77.45**	77.12
Wins/Ties/Losses				-	5/6/3	3/8/3	3/7/4	-	8/3/3(*)	8/3/3(*)	8/3/3(*)

TABLE II: Heat Kernel – Various Datasets

Dataset	Samples	Dimensions	Classes	Heat Kernel - Grid Search				Heat Kernel - Best power			
				simple	PA	PT	PSV	simple	PA	PT	PSV
australian	690	14	2	85.04	85.36	**85.51**	85.39	85.04	85.83	**86.00**	85.56
diabetes	768	8	2	**75.81**	75.70	75.68	75.47	75.81	**76.69**	76.56	76.56
liver-dis	345	6	2	69.74	**69.85**	68.46	69.74	69.74	70.55	**70.66**	70.03
mamographic	961	5	2	80.94	**81.44**	81.37	81.08	80.94	81.50	**81.73**	81.02
PlanRelax	182	12	2	70.22	69.56	70.44	**71.65**	70.22	**71.54**	70.88	71.43
habberman	306	3	2	74.38	74.51	74.57	**74.70**	**74.38**	**74.38**	74.05	73.73
Iris	150	4	3	95.47	95.20	95.07	**96.00**	95.47	95.33	95.47	**96.27**
Balance1	400	2	3	94.55	**99.75**(*)	**99.75**(*)	94.55	94.55	**100.00(*)**	**100.00**(*)	94.55
Balance	625	4	3	92.13	**96.64**(*)	96.58(*)	92.13	92.13	**97.41**(*)	97.28(*)	92.07
Balance3	729	6	3	93.03	98.79(*)	**99.04**(*)	93.03	93.03	98.85(*)	**99.09**(*)	93.03
Ecoli	336	7	8	**86.37**	86.08	85.96	86.13	**86.37**	86.08	86.08	86.08
halfmoon	800	2	2	**99.97**	**99.97**	**99.97**	**99.97**	99.97	**100.00**	**100.00**	**100.00**
cocentral	685	2	3	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
swissroll	1600	3	4	**98.29**	98.16	98.26	98.20	98.29	**98.31**	**98.31**	**98.31**
Wins/Ties/Losses				-	7/2/5	7/2/5	5/5/4	-	11/1/2(**)	10/2/2(**)	8/3/3

all ten performances. All random splits are kept the same for all algorithms to ensure fairness.

The effectiveness of a SVM model depends on the parameters used for training. The most commonly used parameter selection process is grid search. More specifically, we trained with exponentially growing sequences of $C \in \{2^{-5}, \ldots, 2^{15}\}$ and $\gamma \in \{2^{-15}, \ldots, 2^{3}\}$, where $\gamma = \frac{1}{\sigma^2}$. Obviously, inner product uses only parameter C and the RBF matrix uses both parameters C and γ. As it is typically performed, each combination of parameter choices is checked using cross validation (in our case 5 fold cross validation). Finally, the parameters with best cross-validation accuracy are selected. Again, all random splits are kept the same for all algorithms to ensure fairness. The final model, which is used for testing and for classifying new data, is then trained on the whole training set, using the selected parameter as described in [20].

Moreover, in our proposed method, parameter p needs to be selected. We use the same process for parameter selection as before, but add the sequence of $p \in \{1, 2, \ldots, 7\}$. Notice that when $p = 1$ the kernel matrices used do not change and the process is the same as using the simple SVM. Thus, it could be said that simple SVM is a specific parameter selection of our method.

Another issue we have to address is that most optimization libraries are sensitive to the range of values in the kernel matrix. The method we used for normalizing the kernel matrix is described in [21] and is computed as:

$$\tilde{K}^p(\mathbf{x}_i, \mathbf{x}_j) = \frac{K^p(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{K^p(\mathbf{x}_i, \mathbf{x}_i) K^p(\mathbf{x}_j, \mathbf{x}_j)}}. \tag{10}$$

Be aware that the kernel normalization process is performed on $\mathbf{K}\mathbf{K}_r^{p-1}$ which in all three methods, are square, symmetric and their diagonal values are non-zero.

In addition, some datasets have more than 2 classes, which transforms the problem from a binary classification to multi-class classification problem. Multiclass problems are treated as a one versus one problem by building $k(k-1)/2$ binary-class SMV models, where k is the number of classes.

Balance, as described in [19], is generated as 4 features, taking values from 1 to 5 resulting in all possible combinations of $5^4 = 625$. The class for each sample is computed by comparing the product of the first two features to the product of the last two features. The result of this comparison has three outcomes, either the product of first two features is bigger, equal, or smaller than the product of the last two features, resulting in assigning the sample to a class, depending on the outcome. Extending this notion to 2 features and 6 features we created Balance1 and Balance3 respectively. In Balance1, two features take values from 1 to 20 for all combinations of

$20^2 = 400$, and in Balance3 six features take values from 1 to 3 for all combination of $3^6 = 729$. The classes of samples are again computed by comparing the product of their features. In Balance1 the two features are compared to each other, while in Balance3 the product of the first three features is compared to the product of the last three features.

Observing the overall performances in Table I and II, we can say that the grid search does indeed perform slightly better. Moreover, using the overall performance of each algorithm we compare each proposed method against simple SVM as described in [22] by using both the Wilcoxon test and the sign test. If the best power p of the kernel is used, then all of our approaches have better performance than simple SVM in both kernels. Using the aforementioned statistical tests, with 95% confidence, the difference of PA and PT with heat kernel from simple SVM is statistically significant. In inner product, using the Wilcoxon test, there are statistical differences for all proposed approaches against simple SVM.

Moreover, we performed the 5×2cv classification statistical test as described in [23] for all datasets and found that the there are statistically significant differences in performance in dataset cocentral and in all three datasets of Balance. In all of our proposed methods, the statistical test value was over 9, which is larger than 4.74 that Dietterich suggests in order for the difference to be statistically significant.

Now, let's compare the computation complexity of our method against simple SVM. Generally, the computation complexity of SVM is $O(n^3)$ [24]. In addition, the computation of matrix multiplication theoretically is $O(n^{2.373})$ [25], even though we could perform $p-1$ matrix multiplications with cost $O((p-1)n^{2.373})$, in practice, we find the power of a matrix differently. Due to the fact that, in our approach, we generally elevate the matrix to many different powers, in practice it is faster to break kernel matrix $\mathbf{K}$ to its eigenvalue decomposition $\mathbf{K} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ with complexity cost $O(n^3)$ and then apply the property $\mathbf{K}^p = \mathbf{U}\mathbf{D}^p\mathbf{U}^T$ which can easily be computed.

So the main difference of simple SVM against our proposed method is the added cost of multiplying the matrix. We can say that the complexity cost is $O(n^3 + n^3)$ for approaches PA and PT while for PSV is $O(n^3 + n^3 + n^3)$ since, in the latter approach, the knowledge of which samples are support vectors is required. So we can safely state that our approaches have the same order of computation complexity with the simple SVM.

A. Application to Folk Dance and Activity Learning

Apart from classic datasets, we used some real world classification problems. We include some activity based classification datasets, in each of which people perform various activities, which result in a dataset that is multilabeled. However, the problem we selected to solve is that of identifying the activity a person performs. Every dataset can be represented by a number of different dimensions, which is concatenated to its name, and more information about these datasets can be found in [26] and [27].

In order to split the dataset into training and testing sets, we created P folds, where P is the number of people in each dataset. In each fold we use the activities from one person for the testing set and the remaining activities, from the remaining $P-1$ persons, were used for training. Performing the aforementioned splitting method p times results in an overall performance by averaging the performances of each testing set.

More so in the linear kernel and less so in heat kernel used for comparing, we can see in Tables III and IV that our method is superior to simple SVM. The form of data can be complex, and the simple distance of the data does not reflect the whole structure of the data. Our method has more effectively captured the structure of data, and attains generally superior classification results.

In addition, we have used a folk dances dataset. The goal of dance recognition is to identify which dance was performed. The dataset description can be found in [28]. Again, we can see in Tables V and VI that our proposed kernel improves the performance.

V. Conclusion and future work

In this paper we aim to reintroduce random walk kernels, and prove that they can be used in classification. We have provided some mathematical foundation for how this can be simply used in similarity based classifiers and, finally, we have discussed some interesting properties. We also have explained how train and test samples in SVM classifier can be treated in three different ways. This is contingent on the fact that we can control the available similarity paths. Experiments have been conducted on different kind of datasets including activity recognition based datasets, and results look promising.

Future work will be focused on finding new ways of handling training and testing samples by controlling available similarity paths.

Acknowledgment

This research has been co–financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operation Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: THALIS–UOA–ERASITECHNIS MIS 375435.

References

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[2] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vision*, vol. 40, no. 2, pp. 99–121, Nov. 2000.

[3] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.

[4] K. Grauman and T. Darrell, "The pyramid match kernel: Efficient learning with sets of features," *J. Mach. Learn. Res.*, vol. 8, pp. 725–760, May 2007.

[5] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," 1981.

[6] D. Lipman and W. Pearson, "Rapid and sensitive protein similarity searches," *Science*, vol. 227, no. 4693, pp. 1435–1441, 1985.

[7] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti, "Similarity-based classification: Concepts and algorithms," *J. Mach. Learn. Res.*, vol. 10, pp. 747–776, Jun. 2009.

TABLE III: Inner product – Activity Learning Datasets

Dataset	Samples	actions	actors	Inner Product - Grid Search				Inner Product - Best power			
				simple	PA	PT	PSV	simple	PA	PT	PSV
mobiserv5	1288	3	12	73.58	73.59	**73.81**	72.69	73.58	74.27	74.19	**74.85**
mobiserv25				88.11	**89.35**	88.76	88.03	88.11	**89.50**	89.35	89.15
mobiserv75				88.22	88.66	**88.83**	88.22	**88.22**	**88.22**	**88.22**	**88.22**
mobiserv100				86.87	86.53	86.20	**86.94**	86.87	**87.99**	86.87	86.87
ixmas10	2160	12	12	29.63	**30.51**	30.32	30.42	29.63	30.09	**30.19**	**30.19**
ixmas100				36.71	**37.59**	37.13	37.55	36.71	38.19	37.82	**38.52**
ixmas150				35.83	**38.15**	37.87	38.10	35.83	**38.15**	37.87	38.10
ixmas200				36.34	36.30	35.60	**36.39**	36.34	36.57	36.34	**36.67**
ixmas300				36.57	**40.51**	40.28	39.49	36.57	**40.51**	40.28	39.49

TABLE IV: Heat Kernel – Activity Learning Datasets

Dataset	Samples	actions	actors	Heat Kernel - Grid Search				Heat Kernel - Best power			
				simple	PA	PT	PSV	simple	PA	PT	PSV
mobiserv5	1288	3	12	72.85	73.22	**73.35**	73.20	72.85	74.62	74.54	**75.54**
mobiserv25				**87.36**	**87.36**	87.22	87.22	**87.36**	**87.36**	**87.36**	**87.36**
mobiserv75				89.10	89.05	**89.27**	89.10	89.10	89.36	**89.36**	89.34
mobiserv100				**88.63**	88.00	88.47	87.89	**88.63**	**88.63**	**88.63**	**88.63**
ixmas10	2160	12	12	**33.84**	33.15	32.96	33.24	**33.84**	**33.84**	**33.84**	**33.84**
ixmas100				39.54	38.84	38.89	**39.58**	**39.54**	**39.54**	**39.54**	**39.54**
ixmas150				**38.80**	37.87	38.47	38.56	**38.80**	**38.80**	**38.80**	**38.80**
ixmas200				38.98	**39.35**	38.66	38.47	38.98	**39.03**	38.98	38.98
ixmas300				38.98	38.75	**39.26**	38.84	**38.98**	**38.98**	**38.98**	**38.98**

TABLE V: Inner product – Folk Dance Dataset

Dataset	Train Samples	Test Samples	Number Of Dances	Inner product - Grid Search				Inner product - Best power			
				simple	PA	PT	PSV	simple	PA	PT	PSV
isa.100	497	516	5	40.12	43.02	46.71	**52.33**	40.12	**53.88**	**53.88**	53.10
traj.1000				43.60	**47.67**	35.85	32.56	43.60	**51.16**	43.60	38.57

TABLE VI: Heat Kernel – Folk Dance Dataset

Dataset	Train Samples	Test Samples	Number Of Dances	Heat Kernel - Grid Search				Heat Kernel - Best power			
				simple	PA	PT	PSV	simple	PA	PT	PSV
isa.100	497	516	5	**50.97**	37.02	**50.97**	37.02	50.97	**52.52**	50.97	51.36
traj.1000				**39.73**	**39.73**	**39.73**	29.07	39.73	**47.29**	39.73	37.40

[8] G. Wu, E. Y. Chang, and Z. Zhang, "An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[9] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[10] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[11] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *SCIENCE*, vol. 290, pp. 2323–2326, 2000.

[12] S. Yan, D. Xu, B. Zhang, H. jiang Zhang, Q. Yang, S. Member, and S. Lin, "Graph embedding and extension: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 29, pp. 40–51, 2007.

[13] F. R. K. Chung, *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society, Dec. 1996.

[14] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[15] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods – Support Vector Learning*, B. Schlkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1998, pp. 42–65.

[16] M. Szummer and T. Jaakkola, "Partially labeled classification with markov random walks," in *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press, 2001.

[17] O. Chapelle, J. Weston, and B. Scholkopf, "Cluster kernels for semi-supervised learning," in *Advances in Neural Information Processing Systems*, 2002, p. 15.

[18] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," 2003.

[19] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[20] C. wei Hsu, C. chung Chang, and C. jen Lin, "A practical guide to support vector classification," 2010.

[21] A. B. A. Graf and S. Borer, "Normalization in support vector machines," in *Proc. DAGM 2001 Pattern Recognition*. SpringerVerlag, 2001, pp. 277–282.

[22] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," 2006.

[23] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," 1998.

[24] O. Chapelle, "Training a support vector machine in the primal," *Neural Computation*, vol. 19, pp. 1155–1178, 2007.

[25] V. V. Williams, "Breaking the coppersmith-winograd barrier. unpublished manuscript," 2011.

[26] A. Iosifidis, A. Tefas, and I. Pitas, "View-invariant action recognition based on artificial neural networks." *IEEE Trans. Neural Netw. Learning Syst.*, vol. 23, no. 3, pp. 412–424, 2012.

[27] ——, "Activity-based person identification using fuzzy representation and discriminant learning." *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 530–542, 2012.

[28] I. Kapsouras, S. Karanikolos, N. Nikolaidis, and A. Tefas, "Feature comparison and feature fusion for traditional dances recognition," in *Engineering Applications of Neural Networks*. Springer Berlin Heidelberg, 2013, vol. 383, pp. 172–181.