The Generalized Shortest Path Kernel for Classifying Cluster Graphs

Linus Hermansson
Department of Mathematical and Computing Sciences,
Tokyo Institute of Technology, Meguro-ku Ookayama, Tokyo 152-8552, Japan

***Abstract*—We consider using an SVM together with graph kernels in order to classify graphs into classes. Although several graph kernels exist and have been shown experimentally to work for certain graph classification problems, it is often difficult to theoretically analyze for which graph classification problems a particular graph kernel will work well. We provide a semi-theoretical analysis of the feature vectors of the newly published generalized shortest path kernel, which results in a conjecture about the accuracy of an SVM which uses the generalized shortest path kernel. We back up our conjecture with experimental results, for a classification task where the goal is to classify if a graph contains k or $k+1$ number of clusters. Keywords: SVM, Graph Kernel, Machine Learning.**

1 Introduction

A graph kernel can be seen as a similarity measure between graphs, this similarity measure can be used together with a *support vector machine* (SVM) in order to be able to classify graphs into different classes [3], [8], [9]. Graph classification has many useful applications and by solving this problem it is possible to for example classify if human tissue contains cancer or not [1].

One particular type of graphs which comes up in many applications are cluster graphs. By a cluster in a graph we mean a subgraph that contains a higher density of edges inside the clusters than between clusters. Cluster graphs can represent for instance social networks, where each node represents a person and each edge represents a friendship relation. For such graphs the clusters correspond to groups of friends.

The *shortest path* (SP) kernel is a very popular graph kernel to use in combination with an SVM for graph classification and has been shown to be able to classify a wide variety of graphs, such as for instance classifying protein graphs by their enzyme class [3] or classifying if a graph represents a low-density-parity-check code [9]. The newly introduced *generalized shortest path* (GSP) kernel [7], when used by an SVM classifier, has been shown to outperform the SP kernel at classifying certain types of graphs, even though calculating the feature vectors for the GSP kernel takes almost the same amount of time as the SP kernel when using standard algorithms. In this paper we discuss in more detail why and when the SVM classifier, which uses the GSP kernel, outperforms the SVM classifier which uses the SP kernel. For this purpose, we consider harder cluster graph classification problems than the one discussed in [7], that is, for each k, the problem of distinguishing whether a given graph consist of k-clusters or $k+1$-clusters. The hardness of this problem is determined by four parameters: p, the probability of having an edge between two nodes that are inside the same cluster; q_1 (or q_2), the probability of having an edge between two nodes that are in different clusters, and k, the number of clusters. We investigate how the accuracy of the SVM classifier which uses the GSP kernel relates to these four parameters. The details of this classification problem can be found in Sect. 3.

This work is supported in part by KAKENHI No. 24106008.

Our experimental results show that for certain parameter values, the SP kernel does not work at all, even though the GSP kernel completely solves the problem. This happens when p is large and k is small. The reason that the SP kernel does not work well is most likely due to the fact that for these parameters, all node pairs are at very close distances, making it difficult to distinguish feature vectors from the different classes of graphs for the SP kernel.

For understanding the GSP based SVM classifier, we think that it is important to give some formula estimating its accuracy. Unfortunately, though, there are several parameters even for our relatively simple classification task, and the mechanism of the GSP based SVM classifier is not so simple, it seems difficult to give a rigorous analysis of its performance. Thus, we took a heuristic approach in this paper. Based on observations from our experimental results, we propose some statistical value as a "hidden parameter" that determines the accuracy of our classifier. We then derive, as our main technical contribution, some evidence supporting this conjecture using our experimental results.

The rest of the paper is organized as follows. Section 2 contains definitions and terms used throughout the paper. In Sect. 3 we define the random models used to generate graphs and formally define our graph classification problem. Section 4 defines the graph kernels which we investigate. Section 5 contains our semi-theoretical analysis. Section 6 contains our experimental results and Sect. 7 contains our conclusions.

2 Preliminaries

In this section we define our notations that will be used throughout the paper. For a fixed graph, G,V and E denote the graph, the set of vertices and set of edges respectively. We denote the number of nodes and edges in a particular graph by the symbols n and m.

Since our approach is based on graph kernels, (see Sect. 4 for details) which counts the number of node pairs in

graphs that are at particular distances and have a particular number of shortest paths, we define necessary notations for the feature vectors of these graph kernels so that we can appropriately discuss the relevant graph properties. We denote the number of node pairs, that have a shortest path of length $d \geq 1$, in a graph G, by n_d. For $d, x \geq 1$, by $n_{d,x}$, we denote the number of node pairs that are at distance d and have x number of shortest paths. For any given graph G, We call a vector $\boldsymbol{v}_{\rm sp} = [n_1, n_2, \ldots]$ a *SPI feature vector*. For any given graph G, We call a vector $\boldsymbol{v}_{\rm gsp} = [n_{1,1}, n_{1,2}, \ldots, n_{2,1} \ldots]$ a *GSPI feature vector*. Note that $n_d = \sum_x n_{d,x}$. We sometimes in our analysis use feature vectors where we consider shortest paths from a fixed node in a graph, instead of all node pairs. Whenever we use such a version of the feature vectors we point it out in the text.

In this paper we consider different random models for generating graphs. For any specific such model, $\mathrm{E}[\,\boldsymbol{v}_{\rm sp}\,] = [\mathrm{E}[n_1], \mathrm{E}[n_2], \ldots]$, denotes the expected SPI feature vector. We denote the expected GSPI feature vector by $\mathrm{E}[\,\boldsymbol{v}_{\rm gsp}\,] = [\mathrm{E}[n_{1,1}], \mathrm{E}[n_{1,2}] \ldots, \mathrm{E}[n_{2,1}] \ldots]$. In order to separate k and $k+1$-cluster graphs, we use a super script $^{(k)}$ or $^{(k+1)}$. So that for instance $n_{2,1}^{(k)}$ is the number of node pairs that have a shortest path of length 2 and exactly 1 shortest path, *in a k-cluster graph*. We also consider nodes from particular clusters, we define that $n_d^{(y,k)}$ is the number of nodes at distance d, in a k cluster graph, *that are in cluster y only*.

We compare the performances for an SVM that uses either the SP kernel or the GSP kernel. Sometimes for simplicity we write "the accuracy of the SP/GSP kernel", when we mean the accuracy of an SVM which uses the SP/GSP kernel. When showing figures of average experimental feature vectors, the average is always over 200 i.i.d. feature vectors.

3 Random graph model and our graph classification task

The problem that we analyze in this paper is the problem of classifying random graphs as having either k or $k+1$ number of clusters, where k is an integer greater than or equal to 2. A cluster is a subgraph of a graph where the edge density is higher than between the clusters. The graphs which we use in our experiments are generated according to the *planted partition model* [10], which is an extension of the well known *Erdős-Rényi* model [2]. In the planted partition model, for a k-cluster graph, all nodes belong to one of k clusters. Each cluster contains n/k number of nodes. The presence of an edge between any node pair is determined randomly. Two nodes that are from the same cluster are connected with the probability p, while nodes that are from different clusters are connected with probability q_1. Where we define q_1 as

$$q_1 = p(1-\beta).$$

with parameter β which is one of the key parameters in our experiments. We denote the model for generating such graphs as $G(n, k, p, q_1)$.

Each dataset consists of k-cluster graphs and $k+1$-cluster graphs, For the $k+1$-cluster graphs, two nodes that are from the same cluster are connected with probability p, which is the same probability as for the k-cluster graphs. Two nodes that are from different clusters are connected with probability q_2. In order that it is not possible to simply distinguish the different graph types based on the number of edges, we fix q_2 so that the expected number of edges in the k-cluster graphs and the $k+1$-cluster graphs are the same. It is easy to verify that this implies that we have to choose q_2 as

$$q_2 = q_1 + \frac{1}{k^2}(p - q_1) = q_1 + p\frac{\beta}{k^2}.$$

We call the model for generating such graphs $G(n, k+1, p, q_2)$.

The problem which we consider in this paper is to train an SVM using graphs from both models (for a fixed set of parameters p, q_1, q_2, k). We then want to be able to use this SVM to predict, for random graphs which are generated from one of the two models, which of the two models was used to generate each particular graph. I.e. we want our SVM to correctly be able to classify, after training, if a given random graph is a k-cluster graph or a $k+1$-cluster graph.

4 Shortest path kernel based SVM and generalized shortest path kernel based SVM

Here we define the relevant graph kernels which we use together with an SVM in order to classify graphs from our classification problem defined in Sect. 3. A graph kernel is a function $k(G_1, G_2)$ on pairs of graphs, which can be represented as an inner product $k(G_1, G_2) = \langle \phi(G_1), \phi(G_2) \rangle_{\mathcal{H}}$ for some mapping $\phi(G)$ to a Hilbert space $\mathcal{H}$, of possibly infinite dimension. It is convenient to think of graph kernels as similarity functions on graphs. Graph kernels have been used as tools for SVM classifiers for several graph classification problems [3], [4], [8].

The *shortest path* (SP) kernel, compares graphs based on the shortest path length of all pairs of nodes [3]. Let $D(G)$ denote the multi set of shortest distances between all node pairs in the graph G. For two given graphs G_1 and G_2, the SP kernel is then defined as:

$$K_{\rm SP}(G_1, G_2) = \sum_{d_1 \in D(G_1)} \sum_{d_2 \in D(G_2)} k(d_1, d_2),$$

where k is a positive definite kernel [3]. One of the most common kernels for k is the indicator function, as used in [3]. This kernel compares shortest distances for equality. Using this choice of k we obtain the following definition of the SP kernel:

$$K_{\rm SPI}(G_1, G_2) = \sum_{d_1 \in D(G_1)} \sum_{d_2 \in D(G_2)} \mathbb{1}\left[d_1 = d_2\right].$$

We call this the *shortest path index* (SPI) kernel. It is easy to check that $K_{\rm SPI}(G_1, G_2)$ is simply the inner product of the SPI feature vectors of G_1 and G_2.

The *generalized shortest path* (GSP) kernel, is defined using the shortest path length and number of shortest paths between all node pairs. For a given graph G, by $ND(G)$ we denote the multi set of numbers of shortest paths between all node pairs of G. The GSP kernel is

then defined as:

$$K_{\mathrm{GSP}}(G_1, G_2) = \sum_{d_1 \in D(G_1)} \sum_{d_2 \in D(G_2)} \sum_{t_1 \in ND(G_1)} \sum_{t_2 \in ND(G_2)} k(d_1, d_2, t_1, t_2),$$

where k is a positive definite kernel. In this paper we investigate the GSP kernel which considers node pairs equal if they have the same shortest distance *and* the same number of shortest paths. Which gives us the following definition, called the *generalized shortest path index* (GSPI) kernel [7].

$$K_{\mathrm{GSPI}}(G_1, G_2) = \sum_{d_1 \in D(G_1)} \sum_{d_2 \in D(G_2)} \sum_{t_1 \in ND(G_1)} \sum_{t_2 \in ND(G_2)} \mathbb{1}\left[d_1 = d_2\right] \mathbb{1}\left[t_1 = t_2\right].$$

It is easy to see that this is equivalent to the inner product of the GSPI feature vectors of G_1 and G_2.

Computing the SPI and GSPI feature vectors can be done efficiently. One possible method is to use Dijkstra's algorithm [6] for each node in a graph. Doing so takes $\mathcal{O}(nm + n^2 \log n)$ time for one graph and gives us the shortest path length of all node pairs in a graph. This information is exactly what is needed to construct a SPI feature vector. When running Dijkstra's algorithm, the algorithm actually computes all shortest paths between each node pair, although most applications do not store this information, by saving this information we can count the number of shortest paths between each node pair in the graph *without any extra cost in running time*. This means that calculating a GSPI feature vector takes basically the same time as calculating a SPI feature vector.

Note that all the graph kernels used in this paper are inner products between finite dimensional feature vectors. We choose to still call them graph kernels in order to preserve the connection to other researchers' work on graph kernels, see for instance [3], [4], [9].

5 Analysis

In this section we analyze in which situations, and why, the GSP kernel outperforms the SP kernel. Further evidence for our analysis can be found in Sect. 6, which contains our experimental results.

In this section, instead of considering the number of node *pairs* which are at distance d from each other, we consider the number of nodes at a particular distance from a fixed source node s. By $n_{d,x}^{(k)}$ we mean the number of nodes that are at distance d from s and have exactly x number of shortest paths to s, in a k-cluster graph.

5.1 Preliminary approximations

Here we give some preliminary approximations that we use in order to derive our main result. We estimate the expected number of nodes at distances 1 and 2, from a fixed node s, in a k-cluster graph. Our analysis closely parallels [7]. Consider the probability that s is connected to some other fixed node t. Let $F_{t,1}$ be the probability that s and t are connected. $F_{t,1}$ is obviously p if t is in the same cluster as s and q_1 otherwise. Let $F_{t,2}$ be the probability that there exists at least one path of length 2 between s and t. Let $A_{u,v}$ be the event that u and v are connected. $F_{t,2}$ is then equal to

$$F_{t,2} = \Pr\left[\bigvee_{v \in V \setminus \{s,t\}} A_{s,v} \wedge A_{v,t}\right]. \quad (1)$$

$F_{t,2}$ can be calculated using the inclusion-exclusion principle and will give a different result depending on if t is in the same cluster as s or not. Let $f_{t,2}$ denote the probability that s and t are at distance 2. This happens if s and t have a path of length 2 and no path of length 1. I.e. $f_{t,2} = F_{t,2} - F_{t,1}$. Note that $f_{t,2}, F_{t,2}$ and $F_{t,1}$ vary depending on where t is. Let f_2 denote the probabillity that s is at distance 2 to a *random* node t. f_2 is simply the weighted average sum over the cases when t is in the same cluster as s and when t is not in the same cluster as s. The probability of t being in the same cluster as s is simply $1/k$ and the probability that t is not in the same cluster as s is $(k-1)/k$. With this f_2, we are able to estimate that the expected number of nodes at distance 2 is $(n-1)f_2$.

In order to simplify (1), we use an approximation of the inclusion-exclusion principle from [7]. I.e. we approximate

$$\Pr\left[\bigcup_{i=1}^{l} E_i\right] \approx 1 - \exp\left(-\sum_{i=1}^{l} \Pr[E_i]\right).$$

Where E_i, in the case of (1), is $A_{s,v} \wedge A_{v,t}$.

5.2 Analysis of GSPI feature vectors

Here we analyze the expected GSPI feature vectors, $\mathrm{E}[\,\boldsymbol{v}_{\mathrm{gsp}}^{(k)}\,]$ and $\mathrm{E}[\,\boldsymbol{v}_{\mathrm{gsp}}^{(k+1)}\,]$, where the graphs are generated as specified in Sect. 3. Here we consider the number of shortest paths between a fixed source node s and a random target node t. We focus on the part of the GSPI feature vectors that corresponds to nodes at distance 2 from s, namely the subvectors $[\mathrm{E}[n_{2,x}^{(k)}]]_{x \geq 1}$ and $[\mathrm{E}[n_{2,x}^{(k+1)}]]_{x \geq 1}$. Since we in this section consider feature vectors for a fixed s, note that for instance, $\mathrm{E}[n_{2,4}^{(k)}]$ is the expected number of nodes, that are at distance 2 from s and have 4 number of shortest paths to s in a k-cluster graph.

We first consider a k-cluster graph but we use q instead of q_1 so that we are later able to replace q by either q_1 or q_2. Assume, without loss of generality, that s is in cluster 1. Note that s has an expected $(n/k)p$ number of nodes at distance 1, in cluster 1. s also has an expected $(n/k)q$ number of nodes at distance 1 in each of the other $k-1$ clusters. For our analysis, we assume that the number of nodes at distance 1, in each cluster, is actually equal to the expected number of such nodes. We also assume that each node at distance 1 is connected to any node at distance 2 *independently* at random. We give evidence later in this section that these are actually reasonable assumptions.

We now analyze the number of shortest paths between s and t, when s and t are at distance 2. We first consider the case when t is in cluster 1. In this case we know that s has an expected $(n/k)p$ nodes at distance 1 in cluster 1. Due to our assumption of independence of each of those $(n/k)p$ nodes having an edge to t, we say that each such

node is connected to t with probability p independently at random. Which means that if we consider the number of shortest paths between s and t, that goes through cluster 1, the number of shortest paths is distributed according to the binomial distribution $\mathrm{Bin}((n/k)p, p)$. There could also be shortest paths that go from s to cluster 2 then to t, from s to cluster 3 then to t and so on. For any of these situations, the number of shortest paths is distributed according to the distribution $\mathrm{Bin}((n/k)q, q)$. There are obviously $k-1$ such cases. The final distribution of the number of shortest paths when s and t are both in cluster 1 is then

$$\mathrm{Bin}(\frac{np}{k}, p) + (k-1)\mathrm{Bin}(\frac{nq}{k}, q). \quad (2)$$

Similarly, we can calculate that, when t is not in cluster 1, the number of shortest paths between s and t is distributed according to the distribution

$$\mathrm{Bin}(\frac{np}{k}, q) + \mathrm{Bin}(\frac{nq}{k}, p) + (k-2)\mathrm{Bin}(\frac{nq}{k}, q). \quad (3)$$

Note here that we have two different cases, the first case where t is in cluster 1 and the second case where t is in any other cluster. Since the position of t is randomly determined, the final distribution of the number of shortest paths between s and t is a *mixture distribution* [11] of (2) and (3). Where the weight of the first distribution is the number of nodes at distance 2 in cluster 1 over the total number of nodes at distance 2. Which we write as

$$\frac{n_2^{(1,k)}}{n_2^{(k)}} = w_1^{(k)}.$$

We call this weight $w_1^{(k)}$. The weight of the second distribution is the sum of the number of nodes at distance 2 in all clusters except cluster 1, over the total number of nodes at distance 2. Written as

$$\sum_{i=2}^{k} \frac{n_2^{(i,k)}}{n_2^{(k)}} = w_2^{(k)}.$$

We call this weight $w_2^{(k)}$. Note that $w_1^{(k)} + w_2^{(k)} = 1$. The final mixture distribution, for the number of shortest paths between s and a random node t, that is at distance 2 from s, is then

$$w_1^{(k)}(\mathrm{Bin}(\frac{np}{k}, p) + (k-1)\mathrm{Bin}(\frac{nq}{k}, q)) + \quad (4)$$
$$w_2^{(k)}(\mathrm{Bin}(\frac{np}{k}, q) + \mathrm{Bin}(\frac{nq}{k}, p) + (k-2)\mathrm{Bin}(\frac{nq}{k}, q)).$$

This mixture distribution must then have two peaks, the first one being

$$x_{\mathrm{peak}}^{(1,k)} = \frac{n}{k}(p^2 + (k-1)q^2)$$

and the second one being

$$x_{\mathrm{peak}}^{(2,k)} = \frac{n}{k}(2pq + (k-2)q^2).$$

When using the above formulas for a k-cluster graph, we simply replace q by q_1. To obtain the relevant formulas for a $k+1$-cluster graph, we need to replace q by q_2 and k by $k+1$.

We denote the difference between the peaks for a k-cluster graph as

$$\Delta x_{\mathrm{peak}}^{(k)} = x_{\mathrm{peak}}^{(1,k)} - x_{\mathrm{peak}}^{(2,k)} = \frac{np^2\beta^2}{k}. \quad (5)$$

Note that the two weights in (4) will be different for the k-cluster graphs and the $k+1$-cluster graphs. In particular, for fixed values of p, β and k, we always have $w_2^{(k)} < w_2^{(k+1)}$, since the weight of the second distribution increases when the number of clusters increases (since then there is a lower chance of t being in cluster 1). Also note that $w_1^{(k)}$ decreases when k grows. If $w_1^{(k)}$ becomes negligible, we will not have a two peak shape even if the two peaks are separated by a wide margin. This is however never the case for the parameters we consider in our experiments, i.e. $k \in [2, 9]$.

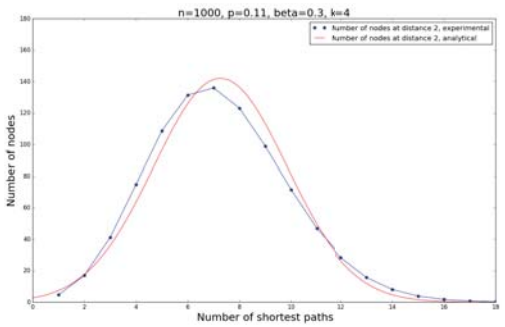

Fig. 1. Comparison between the average experimental and analytical feature subvectors $[\mathrm{E}[n_{2,x}^{(k)}]]_{x\geq 1}$. For n=1000, p=0.11, beta=0.3, k=4.

In the above semi-theoretical analysis, we made two noteworthy assumptions. First that the number of nodes at distance 1 from a fixed node s, is equal to its expected value. We also assumed that all of the distance 1 nodes are connected to distances 2 nodes independently at random, which finally gave us the distribution (4) for the number of shortest paths between s and a random node t at distance 2 from s. In order to provide some evidence that this model is actually reasonably close to what really happens, we provide two figures where we compare the subvector $[\mathrm{E}[n_{2,x}^{(k)}]]_{x\geq 1}$ from the experiments and the subvector which we obtain by using (4) multiplied by the number of nodes at distance 2. We use the approximation from Sect. 5.1 to approximate the number of nodes at distance 2 and substitute the binomial distributions in (4) for normal distributions in order to simplify summing the distributions. The results can be seen in Figures 1 and 2. As can be seen, the distributions obtained from our analysis are very close to the experimental values, no matter if the distribution appears to have only one peak (the two peaks are very close), or if the two peaks are far away.

It is important to note that if the difference between the peaks is large enough, the distribution will look very different compared to when the difference between the peaks is close to 0. If the difference between the peaks is large, the distribution will look skewed compared to a standard normal distribution. This can be seen in Fig. 3, which shows the average subvectors $[\mathrm{E}[n_{2,x}^{(k)}]]_{x\geq 1}$, when $n = 1000$, $p = 0.15$, $k = 2$ for $\beta = 0.2$ and

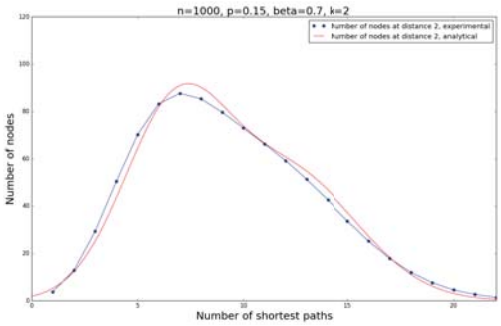

Fig. 2. Comparison between the average experimental and analytical feature subvectors $[\mathrm{E}[n_{2,x}^{(k)}]]_{x\geq 1}$. For n=1000, p=0.15, beta=0.7, k=2.

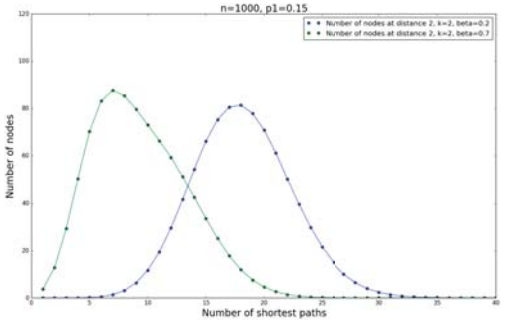

Fig. 3. Average subvectors $[\mathrm{E}[n_{2,x}^{(k)}]]_{x\geq 1}$, when $n = 1000$, $p = 0.15$, $k = 2$ for $\beta = 0.2$ and $\beta = 0.7$. The difference between the peaks when $\beta = 0.2$ is 0.45 and the difference between the peaks when $\beta = 0.7$ is 5.51.

$\beta = 0.7$. Note that the difference between the peaks is a lot greater when $\beta = 0.7$ than when $\beta = 0.2$. In fact the difference between the two peaks is equal to 0.45 when $\beta = 0.2$ and 5.51 when $\beta = 0.7$. The distributions of the k and $k+1$-cluster graphs also appears to look more different when the difference between the peaks is large. Figure 4, contains the average subvectors $[\mathrm{E}[n_{2,x}^{(k)}]]_{x\geq 1}$ and $[\mathrm{E}[n_{2,x}^{(k+1)}]]_{x\geq 1}$, when $n = 1000$, $p = 0.15$, $k = 2$, $\beta = 0.7$. The difference between the peaks for the k-cluster graphs is 5.51 and the difference between the peaks for the $k+1$-cluster graphs is 2.07. If the distance between the peaks is low, the feature vectors tend to look the same for k-cluster graphs and $k+1$-cluster graphs. An example of this can be seen in Fig. 5. Obviously, for such datasets, it is not possible to distinguish the feature subvectors.

Now we propose a "hidden parameter", a major factor in determining the accuracy of the GSPI based SVM classifier. It is well known that the difficulty of distinguishing between a simple one peak distribution and a mixture distribution with two peaks, is related to the ratio of the distance of the two peaks and the standard deviation of the distributions [11]. Here we propose to use this ratio for the hidden parameter. Note that the standard deviation of both (2) and (3) can be approximated as $\sqrt{nq_1^2(1-q_1)}$. Thus, the ratio can be approximated by

$$R_{n,p,k,\beta} = \frac{np^2\beta^2}{k}/\sqrt{nq_1^2(1-q_1)} \approx \sqrt{n}\ p\ \beta^2/(k(1-\beta)). \quad (6)$$

Which we conjecture to be the major factor determining the accuracy of the GSPI based SVM, for the graph classification problem of deciding if a graph contains k or k+1 clusters. In Sect. 6.3 we derive some evidence supporting this conjecture based on experimental results.

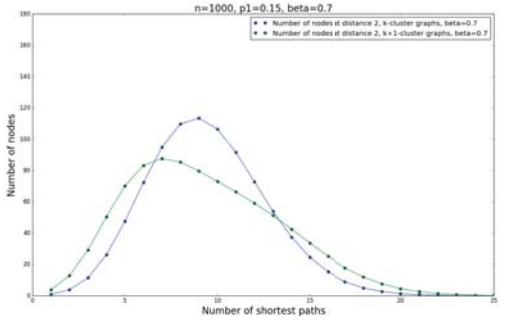

Fig. 4. Average subvectors $[\mathrm{E}[n_{2,x}^{(k)}]]_{x\geq 1}$ and $[\mathrm{E}[n_{2,x}^{(k+1)}]]_{x\geq 1}$, when $n = 1000$, $p = 0.15$, $\beta = 0.7$, $k = 2$. The difference between the peaks for the k-cluster graphs is 5.51 and the difference between the peaks for the $k+1$-cluster graphs is 2.07.

6 Experiments

In this section we show and explain experimental results.

6.1 Dataset specifications and experiment parameters

All datasets are generated according to the models $G(n,k,p,q_1)$ and $G(n,k+1,p,q_2)$. Where each dataset consists of 200 graphs from each model (400 in total). We performed the experiments for the parameters $n = 1000$, $p \in \{0.07, 0.09, 0.11, 0.13, 0.15\}$, $k \in \{2, 3, \ldots, 9\}$ and $\beta \in \{0.2, 0.22, 0.24, \ldots, 0.7\}$. For the SVM, we used *libsvm* [5], with 10-fold cross validation. For the cross validation we tried $C = 2^i$, for $i = [0, 49]$ and the result from the best C was used as the accuracy. Each experiment was repeated 10 times, with different random seeds, and the final accuracy is the average of the 10 runs. If any of the 10 runs resulted in less than 56% accuracy, the final accuracy for that dataset and kernel was set to 56% in order to save running time. Preliminary

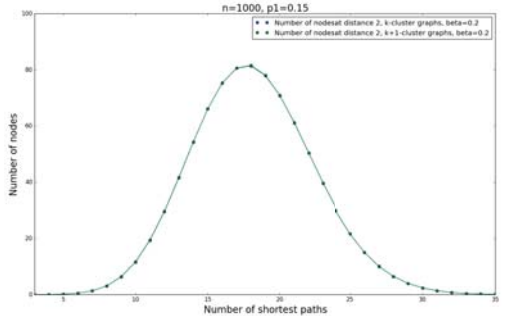

Fig. 5. Average subvectors $[\mathrm{E}[n_{2,x}^{(k)}]]_{x\geq 1}$ and $[\mathrm{E}[n_{2,x}^{(k+1)}]]_{x\geq 1}$, when $n = 1000$, $p = 0.15$, $\beta = 0.2$, $k = 2$. The difference between the peaks for the k-cluster graphs is 0.45 and the difference between the peaks for the $k+1$-cluster graphs is 0.17. The distributions in this picture are indistinguishable.

TABLE 1
COMPARISON OF ACCURACY FOR THE SPI KERNEL AND THE GSPI KERNEL. $p = 0.15$, $k = 2$.

β	SPI accuracy	GSPI accuracy
0.2	56%	63.6%
0.22	56%	70.2%
0.24	56%	78.2%
0.26	56%	85.3%
0.28	56%	92.0%
0.3	56%	95.2%
0.32	56%	98.4%

TABLE 2
COMPARISON OF ACCURACY FOR THE SPI KERNEL AND THE GSPI KERNEL. $p = 0.09$, $k = 2$.

β	SPI accuracy	GSPI accuracy
0.26	60.3%	71.3%
0.28	61.7%	74.7%
0.3	60.3%	82.3%
0.32	62.2%	84.1%
0.34	68.1%	91.9%
0.36	74.5%	92.8%
0.38	78.9%	98.5%

TABLE 3
COMPARISON OF ACCURACY FOR THE SPI KERNEL AND THE GSPI KERNEL. $p = 0.15$, $k = 4$.

β	SPI accuracy	GSPI accuracy
0.26	56%	64.4%
0.28	56%	68.0%
0.30	56%	73.0%
0.32	56%	87.2%
0.34	56%	92.1%
0.36	56%	94.4%
0.38	56%	97.8%

TABLE 4
COMPARISON OF ACCURACY FOR THE SPI KERNEL AND THE GSPI KERNEL. $p = 0.09$, $k = 4$.

β	SPI accuracy	GSPI accuracy
0.34	57.9%	64.2%
0.36	60.8%	74.7%
0.38	64.2%	80.0%
0.4	70.0%	86.2%
0.42	73.6%	90.3%
0.44	78.9%	92.5%
0.46	84.9%	97.1%

experiments showed that $i < 1$ never gave any competitive accuracy. We used the SPI and GSPI kernels, with their feature vectors as defined in Sect. 4, with the modification that each feature vector is normalized by its Euclidean norm, this means that all inner products are in $[0, 1]$.

6.2 Results

Both kernels increase in accuracy when β increases. The SPI does not perform good when p is very large, this is because of the fact that for large values of p, all node pairs are at very close distances. For none of the datasets did the SPI kernel significantly outperform the GSPI kernel, although for very low values of p, the kernels perform similarly in terms of accuracy. For larger values of p, the GSPI significantly outperforms the SPI kernel. A comparison between the accuracies of the kernels can be seen in Tables 1, 2, 3 and 4.

The SPI kernel is only able to detect changes in the number of nodes at certain distances, i.e. $n_1, n_2 \ldots$. Also note that the expected number of edges is the same for the k-cluster graphs and the $k+1$-cluster graphs, which means $\mathrm{E}[n_1^{(k)}] = \mathrm{E}[n_1^{(k+1)}]$. Because of this, for datasets where all nodes are at distances 1 and 2, the SPI kernel will not be able to gain any advantage over a random classifier. The number of nodes at distances 3 and greater, is larger for smaller values of p and larger values of β.

6.3 The accuracy of the GSPI kernel

We plot the accuracy of the GSPI kernel for $p = 0.11$ and $p = 0.15$. These results can be seen in Figures 6 and 7. Our analysis is focused on the subvectors at distance 2 for analyzing the accuracy of the GSPI kernel. The reason for this is because of the fact that for most of the datasets, the number of node pairs that are at distances 1 and 2 is over 50%, meaning that the number of such node pairs is never negligible. In fact, for certain datasets, nearly all the node pairs are at distances 1 and 2. Such a range of parameters is for instance, $p = 0.15, \beta \in [0.2, 0.4], k = 2, 3$, where the number of node pairs at distance more than 2 is always less than 0.1% of the total number of node pairs. This fact, that the number of node pairs at distances greater than 2 is very small, is one reason that the SPI kernel does not perform well for those datasets.

Now let us derive some evidence from our experimental results supporting our conjecture, that is, the ratio $R_{n,p,k,\beta}$ is the major factor determing the accuracy of the GSPI based SVM classifier. Unfortunately, it is computationally hard to get data for various graph sizes n. Thus, in this paper, we consider one reasonably large graph size, namely, $n = 1000$, fixed, and obtain experimental results by changing the other parameters, p, k, and β. Thus, in the following discussion, we regard n in the ratio $R_{n,p,k,\beta}$ as a constant.

First we note the relation between β and the accuracy of the GSPI kernel; see, Figures 6 and 7. From the figures

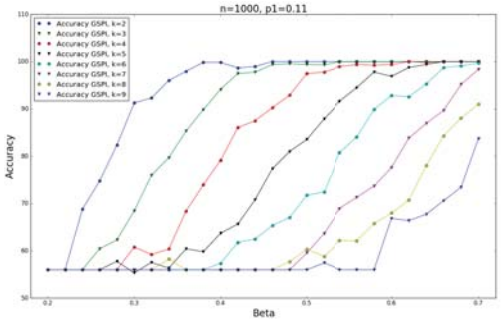

Fig. 6. Accuracy of the GSPI kernel when $p = 0.11$.

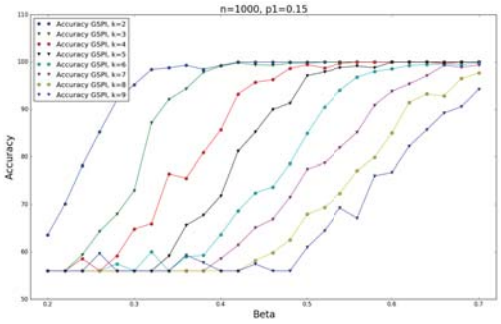

Fig. 7. Accuracy of the GSPI kernel when $p = 0.15$.

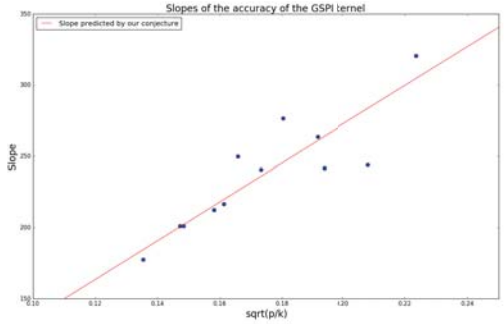

Fig. 8. The slopes of the accuracy of the GSPI kernel for $p \in \{0.11, 0.13, 0.15\}$, $k \in \{3, 4, 5, 6\}$ and the slope as predicted by our conjecture that the slope is proportional to $\sqrt{p/k}$.

it seems that the accuracy increases linearly in β for the interval of β where the accuracy has started to increase above 60% until it reaches close to 100%. That is, for such an interval of β, the accuracy could be written as $a\beta+b$ at least approximately. Note on the other hand that

$$\sqrt{R_{n,p,k,\beta}} \propto \left(\sqrt{p/k}\right) \beta/\sqrt{1-\beta},$$

and that $\beta/\sqrt{1-\beta}$ is close to linear for that interval of β. Thus, if $R_{n,p,k,\beta}$ were the major factor of the accuracy, the accuracy can be expressed as

$$\sqrt{R_{n,p,k,\beta}} + b \approx c\left(\sqrt{p/k}\right) \beta + b,$$

and hence, the slopes of graphs, e.g., in Figures 6 and 7 (i.e., how quickly the accuracy increases when β increases) should be proportional to $\sqrt{p/k}$. We can check this point from our experimental results.

More specifically, we check whether the slopes are proportional to $\sqrt{p/k}$ as follows. First calculate the slope, from the experiments, of the accuracy of the GSPI kernel for several combinations of the parameters p and k. The slopes were obtained by picking the intervals of β values for which the GSPI accuracy seems to increase linearly, where we considered $p \in \{0.11, 0.13, 0.15\}$, $k \in \{3, 4, 5, 6\}$. We then fitted a line $a\beta+b$ using the least squares method in order to obtain the slope a. We then plotted the obtained slopes (values of a) against $\sqrt{p/k}$ and fitted it by a function $g(x) = cx$, again using the least squares method. The result of this can be seen in Fig. 8, where c was determined to be 1362.8. In the figure, each dot represents an experimental slope value obtained from one combination of p and k and the line is the slope that we predict based on

$$\text{Slope} \propto \sqrt{p/k}.$$

As can be seen in the figure, the slope obtained from our conjecture is reasonably close to the real values.

7 Conclusions

We have provided a semi-theoretical analysis of the feature vectors of the GSPI kernel for a random graph classification problem where we classify graphs as having k or $k+1$ number of clusters. Our experiments showed that the analysis closely parallels what really happens. We conjectured that the important parameter for determining the accuracy of the GSPI kernel is related to the distance between the peaks of the mixture distribution of number of shortest paths of distance 2, divided by the standard deviation of the two distributions forming the mixture distribution. We came up with a conjecture for determining what the slope of the accuracy of the GSPI kernel will be, and verified for certain datasets that our conjectured slope value is close to the real value.

Future work should focus on extending the analysis to other values of n. Throughout the paper we only considered $n = 1000$, it would be interesting to analyze what behavior the GSPI kernel has as n increases.

References

[1] C. Bilgin, C. Demir, C. Nagi, and B. Yener. Cell-graph mining for breast tissue modeling and classification. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 5311–5314. IEEE, 2007.

[2] B. Bollobás. *Random graphs*. Springer, 1998.

[3] K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Prof. of ICDM*, 2005.

[4] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl 1):i47–i56, 2005.

[5] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[6] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[7] L. Hermansson, F. D. Johansson, and O. Watanabe. Generalized shortest path kernel on graphs. In *Discovery Science*, pages 78–85. Springer, 2015.

[8] L. Hermansson, T. Kerola, F. Johansson, V. Jethava, and D. Dubhashi. Entity disambiguation in anonymized graphs using graph kernels. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1037–1046. ACM, 2013.

[9] F. Johansson, V. Jethava, D. Dubhashi, and C. Bhattacharyya. Global graph kernels using geometric embeddings. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 694–702, 2014.

[10] S. D. A. Kolla and K. Koiliaris. Spectra of random graphs with planted partitions.

[11] M. F. Schilling, A. E. Watkins, and W. Watkins. Is human height bimodal? *The American Statistician*, 56(3):223–229, 2002.