Flip-Flop Sublinear Models for Graphs

Brijnesh Jain

Technische Universität Berlin, Germany
brijnesh.jain@gmail.com

Abstract. Extending linear classifiers from feature vectors to attributed graphs results in sublinear classifiers. In contrast to linear models, the classification performance of sublinear models depends on our choice as to which class we label as positive and which as negative. We prove that the expected classification accuracy of sublinear models may differ for different class labelings. Experiments confirm this finding for empirical classification accuracies on small samples. These results give rise to flip-flop sublinear classifiers that consider both class labelings during training and select the model for prediction that better fits the training data.

Keywords: graph matching, classification, perceptron learning.

1 Introduction

Linear models are one of the most simple prediction methods that make strong assumptions about the structure of the underlying data and yields stable, but possibly inaccurate predictions [4]. In addition, linear methods form a basis for understanding and devising nonlinear ones.

Application of linear methods, however, is confined to real-valued feature vectors. In [5,12], linear models have been generalized to sublinear models for graphs. Similarly as for linear models, an understanding of sublinear methods is essential for understanding extensions of non-sublinear models on graphs [6,7].

Here, we are interested in understanding the relationship between the performance of sublinear models and the different ways with which we can label the classes. In two-class problems, it is common practice to label one class as positive and the other as negative. For linear models, the classification performance is independent of how we label both classes. The reason is that each vector has an additive inverse. The existence of an inverse allows us to interpret the class regions separated by a hyperplane $\mathcal{H}$ in two ways: the normal of $\mathcal{H}$ points to the positive class. The additive inverse of a normal of $\mathcal{H}$ is also a normal pointing towards the opposite direction. Thus, normal and its additive inverse define the same class regions but with different class labels. As a consequence, there is a dual to each linear function that defines the same class regions but with flipped labels. Since a well-defined addition on graphs is unknown within the framework of sublinear models, the question arises whether there is also a dual for each sublinear function on graphs.

This contribution proves that in almost all cases there is no dual of a sublinear function. Empirical evaluation on relatively small samples confirm that the classification performance of sublinear models depend on whether we label a given class as positive or negative. These findings suggest to devise flip-flop sublinear models that choose the class labeling resulting in better classification accuracy. In experiments we show that flip-flop sublinear models perform better than standard sublinear models.

2 Sublinear Models on Attributed Graphs

This section introduces sublinear models for graphs as proposed by [12].

2.1 The Space of Attributed Graphs

Let $\mathbb{A}$ be a set of node and edges attributes. For the sake of convenience, we assume that $\mathbb{A}$ is the Euclidean space $\mathbb{R}^d$, though the theory presented in this paper can be adapted to the case where node and edges attributes come from arbitrary and possibly disjoint sets. We consider graphs of the form $X = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where $\mathcal{V}$ represents a set of vertices, $\mathcal{E}$ a set of edges, and $\mathcal{A} \subseteq \mathbb{A}$ a set of attributes of the nodes and edges. Node attributes take the form $\boldsymbol{x}_{ii} \in \mathcal{A}$ for each node $i \in \mathcal{V}$ and edges attributes are given by $\boldsymbol{x}_{ij} \in \mathcal{A}$ for each edge $(i, j) \in \mathcal{E}$. By $\mathcal{X}_\mathcal{G}$ we denote the space of all graphs with attributes from $\mathbb{A}$.

Without loss of generality, we may assume that edges must have non-zero attributes. Then each graph can be regarded as a complete graph, where non-edges are treated as edges with zero-attribute. Note that vertices may have zero as well as non-zero attributes. Including non-edges as edges with zero attribute allows us to express graphs X by a matrix $\boldsymbol{X} = (\boldsymbol{x}_{ij})$ with elements $\boldsymbol{x}_{ij} \in \mathcal{A}$.

2.2 Sublinear Dot Product

We equip the space $\mathcal{X}_\mathcal{G}$ with a graph similarity, called sublinear dot product.

Suppose that X is a graph with matrix representation $\boldsymbol{X}$. The particular form of the matrix depends on how the nodes of X are ordered. Since there is no canonical ordering of the nodes, each re-ordering may result in a different matrix representation of X. Let $[\boldsymbol{X}]$ denote the equivalence class of all matrices obtained by permuting the nodes of graph X in all possible ways. We write $\boldsymbol{X}' \in X$ to denote that $\boldsymbol{X}'$ is a representtive of the equivalence class $[\boldsymbol{X}]$.

To formulate the sublinear dot product of two graphs X and Y, we assume that both graphs have the same number of nodes. If this is not the case, we can safely add isolated nodes with zero attribute to the smaller graph until both graphs have the same number of nodes.

Let $\boldsymbol{X} = (\boldsymbol{x}_{ij})$ and $\boldsymbol{Y} = (\boldsymbol{y}_{ij})$ be matrix representations of X and Y, resp., of the same size. Then we define the dot product of $\boldsymbol{X}$ and $\boldsymbol{Y}$ by

$$\boldsymbol{X}^T\boldsymbol{Y} = \sum_{i,j} \boldsymbol{x}_{ij}^T \boldsymbol{y}_{ij},$$

where ${\boldsymbol{x_{ij}}}^T\boldsymbol{y}_{ij}$ denotes the dot product between attribute vectors $\boldsymbol{x}_{ij}$ and $\boldsymbol{y}_{ij}$. Observe that the dot products ${\boldsymbol{x_{ij}}}^T\boldsymbol{y}_{ij}$ correspond to node and edge similarities.

The sublinear dot product maximizes the dot product over all possible matrix representations and is of the form

$$X \cdot Y = \max\left\{\boldsymbol{X}^T\boldsymbol{Y} \,:\, \boldsymbol{X} \in X, \boldsymbol{Y} \in Y\right\}.$$

As shown in [8], we can equivalently express $X \cdot Y$ by

$$X \cdot Y = \max\left\{\boldsymbol{X}^T\boldsymbol{Y} \,:\, \boldsymbol{X} \in X\right\} = \max\left\{\boldsymbol{X}^T\boldsymbol{Y} \,:\, \boldsymbol{Y} \in Y\right\},$$

where $\boldsymbol{Y} \in Y$ is an arbitrarily chosen matrix representation for the first equation and $\boldsymbol{X} \in X$ for the second equation. We call $\boldsymbol{X}$ and $\boldsymbol{Y}$ optimally aligned, if

$$X \cdot Y = \boldsymbol{X}^T\boldsymbol{Y}.$$

The sublinear dot product extends the dot product from vectors to graphs. It is straightforward to verify that the function $f_Y(X) = X \cdot Y$ as a pointwise maximizer of dot products is sublinear, that is convex and positively homogeneous. For this reason, we call $X \cdot Y$ sublinear. Though the sublinear dot product is not linear, it shares similar geometrical properties and generalizes the concept of maximum common subgraph [8]. It can be reduced to a special case of the graph-edit distance and is widely used in different guises as a common choice of proximity measure for graphs [2,3,17].

2.3 Sublinear Models for Graphs

Sublinear Functions. Sublinear functions on graphs are functions of the form

$$f(X) = W \cdot X + b, \tag{1}$$

where W is the weight graph and $b \in \mathbb{R}$ is the bias. We assign a graph X to class $y = +1$ if $f(X) \geq 0$ and to class $y = -1$ if $f(X) < 0$. Then the equation $f(X) = 0$ defines a decision surface

$$\mathcal{H}_f = \{X \in \mathcal{X}_\mathcal{G} \,:\, f(X) = 0\} \subseteq \mathcal{X}_\mathcal{G}$$

that separates both class regions.

The Learning Problem. The goal of learning consists in finding a weight graph W and bias b such that the s-linear discriminant $f(X) = W \cdot X + b$ minimizes the expected risk

$$E(f) = \int_{\mathcal{X}_\mathcal{G}} L(f(X), y)\, dP(X, y), \tag{2}$$

where $P(X, y)$ is the joint probability distribution on $\mathcal{X}_\mathcal{G} \times \mathcal{Y}$ and $L(\hat{y}, y)$ is a differentiable loss function that measures the cost of predicting class $\hat{y}$ when the actual class is y.

Since the distribution $P(X, y)$ is usually unknown, the expected risk $E(f)$ can not be computed directly. Instead, we approximate the expected risk by minimizing the empirical risk

$$E_N(f) = \frac{1}{N}\sum_{i=1}^{N} L(f(X_i), y_i)$$

on the basis of N training examples $(X_i, y_i) \in \mathcal{X}_{\mathcal{G}} \times \mathcal{Y}$.

Subgradient Learning Rules. To minimize the empirical risk $E_N(f)$, we present the margin perceptron algorithm for graphs. For this let $f(X) = W \cdot X + b$ be a sublinear function. We define a tube $\mathcal{T}_{f,\lambda}$ around the decision boundary $\mathcal{H}_f$ consisting of all graphs X with $|f(X)| \leq \lambda$, where $\lambda \geq 0$ is the margin parameter. Suppose that (X, y) is a new training example. The margin perceptron updates the weight graph and bias if one of the two cases occurs: (1) f misclassifies X, or (2) f correctly classifies X, but X lies in the tube $\mathcal{T}_{f,\lambda}$. Both conditions are met when $y \cdot f(X) \leq 0$. In this case the update rule is of the form

$$\begin{aligned} \boldsymbol{W} &\leftarrow \boldsymbol{W} + \eta \cdot y \cdot \boldsymbol{X} \\ b &\leftarrow b + \eta \cdot y, \end{aligned}$$

where η is the learning rate and $\boldsymbol{W} \in W$ and $\boldsymbol{X} \in X$ are optimally aligned matrix representations of W and X, that is $W \cdot X = \boldsymbol{W}^T \boldsymbol{X}$.

As shown in [12], the update rule of the graph perceptron minimizes the empirical risk $E_N(f)$, where the underlying loss function is of the form

$$L(f(X), y) = \max\{0, \lambda - y \cdot f(X)\}.$$

For $\lambda = 0$, we obtain the graph perceptron algorithm as a special case. Convergence is discussed in [11,12].

3 Flip-Flop Sublinear Models

The main result of this contribution is Theorem 1 stating that the performance of sublinear models depends on which of both classes is labeled as positive class. As an implication of Theorem 1, we introduce flip-flop sublinear models.

Each sublinear function $f(X) = W \cdot X + b$ separates the graph space $\mathcal{X}_{\mathcal{G}}$ into two class regions of the form

$$\mathcal{R}_+(f) = \{X \in \mathcal{X}_{\mathcal{G}} \,:\, f(X) \geq 0\} \quad \text{and} \quad \mathcal{R}_-(f) = \{X \in \mathcal{X}_{\mathcal{G}} \,:\, f(X) < 0\},$$

where $\mathcal{R}_+(f)$ is the region for the positive and $\mathcal{R}_-(f)$ the region of the negative class. We define the class-dual of f as a sublinear function $f'(X) = W' \cdot X + b'$ separating $\mathcal{X}_{\mathcal{G}}$ into class regions of the form

$$\mathcal{R}_+(f') = \mathcal{R}_-(f) \quad \text{and} \quad \mathcal{R}_-(f') = \mathcal{R}_+(f).$$

By definition, the class-dual f' is a sublinear function that implements the same class regions as f but with flipped labels. In vector spaces, each linear function $h(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ with non-zero weight $\boldsymbol{w}$ has a unique class-dual, which is of the form $h'(\boldsymbol{x}) = -\boldsymbol{w}^T\boldsymbol{x} - b$. This statement is invalid in graph spaces as shown by the next result (a proof is presented in [13]).

Theorem 1. *There is no class-dual of a sublinear function with probability one.*

Suppose that the graph space is partitioned in two class regions $\mathcal{R}_+$ and $\mathcal{R}_-$. Consider the expected classification accuracy of the sublinear function f

$$\mathbb{C}_{ref}[f] = \int [f(X), y]\, dP(X, y),$$

where $[f(X), y] = 1$ if f correctly classifies X as y, and 0 otherwise. The subscript of $\mathbb{C}_{ref}$ refers to the current labeling of the classes as the reference labels. If there is a sublinear model f^* with $\mathbb{C}_{ref}[f^*] = 1$, then f^* perfectly separates both classes. When flipping the labels, we assign graphs from class region $\mathcal{R}_+$ negative and graphs from class region $\mathcal{R}_-$ positive labels. Under the assumption that $\mathbb{C}_{ref}[f^*] = 1$, Theorem 1 yields

$$\max_f \mathbb{C}_{flip}[f] < \mathbb{C}_{ref}[f^*] = 1,$$

where $\mathbb{C}_{flip}$ is the expected classification accuracy when the original class labels have been flipped. In a more general setting, we arrive at the following result:

Corollary 1. *For two-class problem, we generally have*

$$\max_f \mathbb{C}_{ref}[f] \neq \max_f \mathbb{C}_{flip}[f],$$

where the maximum is taken over all sublinear functions on graphs.

Corollary 1 gives rise to flip-flop sublinear models that selects the labeling resulting in a better separation of the data:

Algorithm 1. (Flip Flop Classifier)

```
Input:
    Sample $\mathcal{S} \subseteq \mathcal{X}_\mathcal{G} \times \mathcal{Y}$.
Procedure:
    Learn sublinear classifier $f(X)$ on the basis of $\mathcal{S}$ with accuracy $\alpha_\mathcal{S}(f)$.
    Construct dual sample $\mathcal{S}'$ according to $(X, y) \in \mathcal{S} \Leftrightarrow (X, -y) \in \mathcal{S}'$.
    Learn sublinear classifier $f'$ on the dual sample $\mathcal{S}'$ with accuracy $\alpha_{\mathcal{S}'}(f')$.
Return:
    Classifier and labeling that yields a higher classification accuracy
        $f^* = \arg\max_{f, f'} \{\alpha_\mathcal{S}(f), \alpha_{\mathcal{S}'}(f')\}$
```

4 Experiments

Experiments on two-class problems aims at investigating the behavior of sublinear models under different class-labelings. Experiments on multi-class problems aim at assessing the performance of flip-flop sublinear models in a practical setting when class-labeling is random-like.

4.1 Data

We selected subsets of the following training data sets from the IAM graph database repository [16]: letter (low, medium, high), fingerprint, grec, and coil. The *letter* data sets compile distorted letter drawings from the Roman alphabet that consist of straight lines. Lines of a letter are represented by edges and endpoints of lines by vertices. The distortion levels are low, medium, and high. Fingerprint images of the *fingerprints* data set are converted into graphs, where vertices represent endpoints and bifurcation points of skeletonized versions of relevant regions. Edge represent ridges in the skeleton. The *grec* data set consists of graphs representing symbols from noisy versions of architectural and electronic drawings. Vertices represent endpoints, corners, intersections, or circles. Edges represent lines or arcs. The $coil_{13,16}$ and $coil_{42,44}$ data sets are subsets of the coil-100 data set consisting of objects corresponding to the subscripted pairs of indices $3, 16$ and $42, 44$ (starting at index 0). The first pair of indices refers to images representing two different types of rubber cats and the second pair to images representing two different types of cups. After preprocessing, the images are represented by graphs, where vertices represent endpoints of lines and edges represent lines. All datasets consist of a fixed training-, validation-, and test-set.

4.2 Experiments – Label Dependency

Data. We considered the following two-class problems: (1) letters A and H of the letter-high data set, (2) letters E and F of the letter-high data set, (3) classes 0 and 1 of the fingerprint data set, (4) $\text{coil}_{13,16}$, and (5) $\text{coil}_{42,44}$.

Experimental Protocol. For each data set, we randomly sampled 50% off all data for training in a stratified manner. The remaining examples formed the test set. Then we applied the graph-perceptron algorithm ($\lambda = 0$) using the graduated assignment algorithm [3] for computing the sublinear dot product. We recorded the classification accuracy on the training- and test-examples. The learning-rates of the perceptron algorithm were taken from [12]. We repeated this experiments 50 times. Next, for each data set, we flipped the labels of both classes and repeated the same experiment again 50 times.

Results and Discussion. Table 1 summarizes the results. The Shapiro-Wilk test at significance level $\alpha = 0.05$ rejected in about half of the cases the hypothesis that the classification accuracies are normally distributed. For this reason,

Table 1. Results of the graph-perceptron algorithm for two-class classification problems using the original and the flipped class labeling. Shown are the average classification accuracies over 50 trials, the standard deviation and the p-values obtained from the Mann-Whitney U-Test. Rows marked with $'+'$ refer to a class labeling with better results than rows marked with $'-'$. The quantity d/N is the ratio of the dimension d of the largest matrix representation of a training graph and the number N of training examples.

			training			test		
	d/N		avg	std	p-val	avg	std	p-val
Letter-High$_{A,H}$	0.3	+	100.0	0.0	0.000	96.2	2.0	0.000
		−	95.4	1.4		89.2	2.7	
Letter-High$_{E,F}$	0.3	+	98.4	0.9	0.007	90.3	2.8	$\times$0.259
		−	97.6	1.6		89.8	2.2	
Fingerprint$_{0,1}$	0.8	+	99.8	0.1	0.000	96.3	0.3	0.000
		−	73.2	3.3		72.0	3.2	
Coil$_{13,16}$(cats)	59.1	+	100.0	0.0	$\times$0.230	89.2	7.0	0.000
		−	99.6	1.3		75.2	9.4	
Coil$_{42,44}$ (cups)	41.0	+	100.0	0.0	0.000	87.3	5.5	0.000
		−	97.4	4.2		70.2	6.85	

we applied the Mann-Withney U-Test for testing the null hypothesis whether the classification accuracies of the original labeling and the flipped labeling come from the same distribution. In all but two cases (marked as $\times$), the resulting p-values were less than the significance levels $\alpha = 0.01$ and $\alpha = 0.05$. In these 8 out of 10 cases, we rejected the null hypothesis and accepted the alternative hypothesis that the accuracies come from different distributions.

The results show that the average accuracies are different in 8 out of 10 cases and that the differences are significant. From this we conclude that the average accuracy of a graph-perceptron depends on our choice as to which class we label as positive and negative. Recall that Theorem 1 and its implications consider the expected classification accuracy. Empirical classification accuracies based on finite samples of relatively small size according to the ratio d/N confirm that the theoretical findings are of practical relevance.

In all cases, a class labeling with better average accuracy results in a better average generalization performance. This also holds when the difference on the training set is not statistically significant as in the Coil$_{13,16}$ data set. Conversely, a statistically significant difference on the training set does not guarantee a statistically significant difference on the test set as shown for Letter-High$_{E,F}$.

As expected generalization performance was lower than the performance on the training examples. Notable is the strong decrease of generalization performance for both Coil data sets indicating overfitting. Inspecting the ratio d/N of the dimension of the largest matrix representation of a graph in the training set and the number N of training examples shows that the dimension is roughly 40 and 60 times higher than the number of training examples. According to Covers

Function Counting Theorem, we can always find a separating decision surface for the training set provided the classes are labeled favorably and the training examples are in general position. In addition, we can also expect good results on the training data, when the class labeling is unfavorable. Due to the high dimension and the low number of training examples, it is likely that the learned models do not generalize well.

4.3 Experiments – Flip-Flop Sublinear Models

Data. We considered the following multi-class problems: letter (low, medium, high), fingerprint, and grec.

Experimental Protocol. To cope with multiple classes, we applied the flip-flop perceptron and the flip-flop margin perceptron algorithm using a one-versus-all approach. For computing the sublinear dot product, we again applied the graduated assignment algorithm [3]. The learning-rates and margin parameters were taken from [12]. For each data set, we trained both flip-flop sublinear models on the union of the training and validation data. We assessed the generalization performance on the test data. We repeated this experiment 10 times. We used the given splits instead of random splits of the training and test data in order to make the results comparable to other methods.

Results and Discussion. Table 2 summarizes the training and test results of the flip flop perceptron and flip flop margin perceptron algorithm for multi-class problems.

We first compare the training results of the four different graph perceptron algorithms. We observe that on average both flip-flop perceptrons better separate the training data than their standard counterparts. We also observe that for flip-flop classifiers, the margin perceptron does not yield the best training results on three out of five data sets (Letter M, F'print, GREC). This is in contrast to the standard versions of both perceptron algorithms. Finally, we see that the differences are small compared to those obtained in our first experiments on two-class problems. One reason for this is that our experiments on two-class problems consider the extreme case of an unfavorable vs. a favorable labeling, whereas these experiments consider the natural labeling vs. the favorable labeling. The natural labeling in a one-against-all classification approach labels the corresponding single class as positive and all other classes as negative. For most classes, this labeling turns out to be the favorable one, such that flipping the labeling is necessary only in few cases. Thus, for most one-against-all dichotomies in these experiments, the natural labeling coincides with the favorable labeling.

Next, we compare the test results of the four different graph perceptron algorithms. The first observation to be made is that both flip-flop perceptrons perform better on average than their standard counterparts on all but the GREC data set.

Table 2. Training and test classification accuracies for multi-class problems. The number of classes is shown in parentheses next to the identifier of the respective data set. Shown are the average accuracy, standard deviation, and maximum accuracy over 10 runs of perceptron (perc), margin perceptron (mperc), flip-flop perceptron (ffperc), and flip-flop margin perceptron (ffmperc) for training and test data. Generalization performance is compared against graph Bayes (bayes$_2$), generalized learning graph quantization (glgq), similarity-kernel SVM (sk-svm), support vector machine applied on dissimilarity embeddings after dimension reduction using PCA (pca+svm), and optimized dissimilarity space embedding (odse.v2). Results for entries with a dash – are not available. Results marked with an asterique * are not comparable, because the grec data set as used in [1] differs from the on publicly available at [16].

train	Letter L (15)		Letter M (15)		Letter H (15)		F'print (4)		GREC (22)	
	avg	max	avg	max	avg	max	avg	max	avg	max
ffperc	$97.0^{\pm 0.5}$	97.6	$93.3^{\pm 0.4}$	94.0	$86.7^{\pm 0.8}$	87.7	$87.1^{\pm 0.5}$	88.0	$99.5^{\pm 0.3}$	100.0
ffmperc	$98.9^{\pm 0.2}$	99.2	$93.1^{\pm 0.2}$	93.5	$88.9^{\pm 0.5}$	89.7	$86.5^{\pm 0.7}$	87.3	$99.4^{\pm 0.2}$	99.8
perc	$96.9^{\pm 0.3}$	97.5	$92.0^{\pm 0.6}$	93.1	$84.3^{\pm 0.7}$	85.3	$80.0^{\pm 1.7}$	81.8	$98.2^{\pm 0.3}$	98.6
mperc	$96.9^{\pm 0.3}$	97.2	$92.6^{\pm 0.5}$	93.7	$86.4^{\pm 0.5}$	87.1	$80.7^{\pm 2.9}$	84.3	$99.0^{\pm 0.2}$	99.1

test	Letter L (15)		Letter M (15)		Letter H (15)		F'print (4)		GREC (22)	
	avg	max	avg	max	avg	max	avg	max	avg	max
ffperc	$95.6^{\pm 0.5}$	96.1	$89.4^{\pm 0.6}$	90.3	$83.5^{\pm 0.9}$	84.3	$82.3^{\pm 0.7}$	83.4	$96.1^{\pm 0.4}$	96.6
ffmperc	$97.0^{\pm 0.4}$	97.5	$90.4^{\pm 0.8}$	91.7	$85.6^{\pm 0.7}$	86.5	$83.1^{\pm 0.5}$	84.0	$96.9^{\pm 0.3}$	97.4
perc [12]	$94.5^{\pm 0.7}$	96.0	$86.1^{\pm 1.1}$	87.5	$80.7^{\pm 1.1}$	82.5	$76.8^{\pm 1.6}$	79.1	$96.3^{\pm 0.5}$	97.0
mperc [12]	$95.5^{\pm 0.3}$	95.7	$88.7^{\pm 0.6}$	89.5	$84.1^{\pm 0.5}$	84.8	$79.5^{\pm 2.6}$	82.4	$97.5^{\pm 0.6}$	98.1
bayes$_2$ [9]	–		–		80.4		79.2		89.9	
glgq [10]	–		–		88.4		84.8		97.5	
sk-svm [1]	99.2		94.7		92.8		81.7		92.2*	
pca+svm [1]	99.2		94.9		92.1		82.2		92.0*	
odse.v2 [14]	99.0		96.8		96.2		–		97.9	

Let us compare the results of the flip-flop classifiers against bayes$_2$ and glgq. Both are also classifiers based on the graph orbifold framework. Bayes$_2$ is based on bell-shaped distributions around center graphs and glgq is an extension of generalized learning vector quantization to the graph domain. The results show that both flip-flop classifiers are superior than bayes$_2$ but perform worse compared with glgq.

Finally, we compare both flip-flop classifiers against other state-of-the-art algorithms. From the results we see that sk-svm, pca+svm, and odse.v2 are clearly superior on all letter data set and therefore more robust against noise. The picture changes when it comes to the F'print and GREC data set. The graph perceptrons algorithms are slightly better on F'print and comparable to odse.v2 on GREC. These findings are similar to linear in vector spaces: graph perceptrons are simple methods that yield possibly inaccurate results. Further improvements are possible in two ways: first, by more extensively exploring the hyperparameters, and second, by controlling the VC dimension via the number of nodes and edges of the weight graph (for details see [12]).

5 Conclusion

Theorem 1 states that there is no dual of a sublinear model with probability one. An immediate consequence of this result is that the expected classification performance of sublinear models depends on our choice as to which class we label as positive and which as negative. Experiments on finite samples of relatively small size compared to the dimensionality of the data confirm the theoretical findings for empirical classification performance. This justifies flip-flop classifiers that consider both labelings during training and select the model with the better classification performance of the training data.

References

1. Bunke, H., Riesen, K.: Improving vector space embedding of graphs through feature selection algorithms. Pattern Recognition 44(9), 1928–1940 (2011)
2. Cour, T., Srinivasan, P., Shi, J.: Balanced graph matching. In: NIPS (2006)
3. Gold, S., Rangarajan, A.: A graduated assignment algorithm for graph matching. IEEE Trans. on Pattern Analysis and Machine Intelligence 18(4), 377–388 (1996)
4. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning. Springer, New York (2001)
5. Jain, B., Wysotzki, F.: Multi-Layer Perceptron Learning in the Domain of Graphs. IJCNN 3, 1993–1998 (2003)
6. Jain, B., Wysotzki, F.: Structural perceptrons for attributed graphs. In: Fred, A., Caelli, T.M., Duin, R.P.W., Campilho, A.C., de Ridder, D. (eds.) SSPR&SPR 2004. LNCS, vol. 3138, pp. 85–94. Springer, Heidelberg (2004)
7. Jain, B.: Structural Neural Learning Machines. PhD thesis, TU Berlin (2005)
8. Jain, B., Obermayer, K.: Structure Spaces. The Journal of Machine Learning Research 10, 2667–2714 (2009)
9. Jain, B., Obermayer, K.: Maximum likelihood for gaussians on graphs. In: Jiang, X., Ferrer, M., Torsello, A. (eds.) GbRPR 2011. LNCS, vol. 6658, pp. 62–71. Springer, Heidelberg (2011)
10. Jain, B., Obermayer, K.: Generalized learning graph quantization. In: Jiang, X., Ferrer, M., Torsello, A. (eds.) GbRPR 2011. LNCS, vol. 6658, pp. 122–131. Springer, Heidelberg (2011)
11. Jain, B., Obermayer, K.: Learning in Riemannian Orbifolds. arXiv preprint arXiv:1204.4294 (2012)
12. Jain, B.: Sublinear Models for Graphs. arXiv preprint arXiv:1204.4294 (2014)
13. Jain, B.: Flip-Flop Sublinear Models for Graphs: Proof of Theorem 1. arXiv preprint arXiv:cs.LG (2014)
14. Livi, L., Rizzi, A., Sadeghian, A.: Optimized Dissimilarity Space Embedding for Labeled Graphs. Information Sciences (2014)
15. Riesen, K., Neuhaus, M., Bunke, H.: Graph embedding in vector spaces by means of prototype selection. In: Escolano, F., Vento, M. (eds.) GbRPR. LNCS, vol. 4538, pp. 383–393. Springer, Heidelberg (2007)
16. Riesen, K., Bunke, H.: IAM graph database repository for graph based pattern recognition and machine learning. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSPR&SPR 2008. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)
17. Umeyama, S.: An eigendecomposition approach to weighted graph matching problems. IEEE Transactions on PAMI 10(5), 695–703 (1988)