ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

Exploiting Vertices States in GraphESN by Weighted Nearest Neighbor

Claudio Gallicchio and Alessio Micheli

Department of Computer Science - University of Pisa
Largo B. Pontecorvo, 3 - 56127 Pisa, Italy.

Abstract. Graph Echo State Networks (GraphESN) extend the Reservoir Computing approach to directly process graph structures. The reservoir is applied to every vertex of an input graph, realizing a contractive encoding process and resulting in a structured state isomorphic to the input. Whenever an unstructured output is required, a state mapping function maps the structured state into a fixed-size feature representation that feeds the linear readout. In this paper we propose an alternative approach, based on distance-weighted nearest neighbor, to realize a more flexible readout exploiting the state information computed for every vertex according to its individual relevance.

1 Introduction

Echo State Networks (ESN) and Reservoir Computing (RC) [1] represent efficient approaches to Recurrent Neural Networks (RNNs) modeling. An ESN is composed of an untrained recurrent non-linear *reservoir* and of a feed-forward trained linear *readout*. Recursive Neural Networks (RecNNs) [2] are a generalization of RNNs for processing in structured domains (SDs), e.g. domains of trees or graphs. Recently, the TreeESN [3, 4] and GraphESN [5] models have been proposed as efficient approaches to RecNNs modeling, allowing to extend the RC framework for mapping input structures into vectorial outputs. The reservoir of a GraphESN is applied to every vertex of an input graph, computing a state that represents both its label and local topology, implementing a contractive encoding process that converges to stable state values. The condition of contractivity, inherited from ESN models, allows GraphESNs to deal with directed/undirected, cyclic/acyclic labeled graphs. Contractivity thereby assumes a further relevant meaning in terms of the extension of the class of data structures supported by RecNN models, typically limited to directed acyclic graphs.

Whenever a single vector is required in output, there is the problem of extracting/weighting the vertices states to get a fixed-size feature representation to feed the readout. A general approach in this concern should be able to deal with graphs of different sizes without forcing alignments limiting the structure topologies. A supersource *state mapping function* (SMF), selecting the state of the supersource (when defined), and a mean SMF, averaging the state over the vertices, have been proposed [3, 5]. The choice of the SMF has revealed a critical role on the model in relation to the properties of the task, showing the relevant effect of the extraction of information from the reservoir space [3]. Although such fixed metrics turned out to be effective in applications [3, 5], more flexible

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

solutions still need to be explored. In this paper we propose an alternative approach to implement the readout of a GraphESN, based on the distance-weighted K-nearest neighbor (K-NN) algorithm. In particular, we attempt to model the influence of every vertex state on the final output, proportionally to its ability to retain useful information about the distribution of the vertices states with respect to the target. Such approach would fit more flexibly the properties of the target task, e.g. in the context of toxicity prediction, where the influence of different atoms on the toxicity of a molecule could significantly vary.

2 GraphESNs for SD processing

A graph $\mathbf{g}$ is a couple $(V(\mathbf{g}), E(\mathbf{g}))$, where $V(\mathbf{g})$ is the set of vertices and $E(\mathbf{g}) = \{(u, v) : u, v \in V(\mathbf{g})\}$ denotes the set of edges. The number of vertices of $\mathbf{g}$ is denoted by $|V(\mathbf{g})|$. We use V and E in place of $V(\mathbf{g})$ and $E(\mathbf{g})$, respectively, if the reference to the graph is implicit. The neighborhood of $v \in V$ is the set of adjacent vertices of v in the undirected version of $\mathbf{g}$, i.e. for an undirected graph $\mathcal{N}(v) = \{u \in V : (u, v) \in E\}$. The degree of v is the dimension of its neighborhood, i.e. $|\mathcal{N}(v)|$. The maximum degree over the set of graphs considered is indicated as k. Each vertex v has a vectorial numerical label associated, $\mathbf{u}(v) \in \mathbb{R}^{N_U}$, where $\mathbb{R}^{N_U}$ is a vectorial label input space. The set of graphs with labels in $\mathbb{R}^{N_U}$ and maximum degree k is indicated by $(\mathbb{R}^{N_U})^{\#k}$.

We are interested in computing *structural transductions* (STs), in particular functions mapping an input graph $\mathbf{g}$ into a fixed-size vectorial output $\mathbf{y}(\mathbf{g}) \in \mathbb{R}^{N_Y}$, i.e. $\mathcal{T} : (\mathbb{R}^{N_U})^{\#k} \to \mathbb{R}^{N_Y}$. Such STs are also called *structure-to-element* STs [5]. STs can be conveniently computed as $\mathcal{T} = \mathcal{T}_{out} \circ \mathcal{X} \circ \mathcal{T}_{enc}$, where $\mathcal{T}_{enc}$ is an *encoding* transduction, $\mathcal{X}$ is the SMF and $\mathcal{T}_{out}$ is an *output* function. $\mathcal{T}_{enc} : (\mathbb{R}^{N_U})^{\#k} \to (\mathbb{R}^{N_R})^{\#k}$ maps an input structure $\mathbf{g}$ into an isomorphic structured state $\mathbf{x}(\mathbf{g})$, where $\mathbb{R}^{N_R}$ is a feature (state) space. The SMF $\mathcal{X} : (\mathbb{R}^{N_R})^{\#k} \to \mathbb{R}^{N_R}$ maps a structured state $\mathbf{x}(\mathbf{g})$ into a fixed-size feature representative for the whole graph. Finally, $\mathcal{T}_{out} : \mathbb{R}^{N_R} \to \mathbb{R}^{N_Y}$ is applied to compute the output, i.e. $\mathbf{y}(\mathbf{g})$.

A GraphESN computes partially adaptive STs. It consists in an input layer of N_U units, a *reservoir* layer of N_R non-linear (sparsely connected) recursive units, and an *readout* layer of N_Y linear feed-forward units. The reservoir computes a fixed encoding $\mathcal{T}_{enc}$ by implementing a *local encoding function* τ, which has the role of recursive state transition function and is applied to every vertex of the input graph, i.e. $\mathbf{x}(v) = \tau(\mathbf{u}(v), \mathbf{x}(\mathcal{N}(v))) = f(\mathbf{W}_{in}\mathbf{u}(v) + \sum_{v' \in \mathcal{N}(v)} \hat{\mathbf{W}}\mathbf{x}(v'))$, where $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U}$ is the input-to-reservoir weight matrix, $\hat{\mathbf{W}} \in \mathbb{R}^{N_R \times N_R}$ is the (sparse) recurrent weight matrix for the states of the neighbors and f is the component-wise applied activation function of reservoir units (we use *tanh*). Function τ expresses the dependencies of the state of v from the states of its neighbors. Differently from standard ESN processing, the computation of the state for v does not depend only on the state of a predecessor, but more generally from the states of vertices in the neighborhood of v, topologically extended over $\mathbf{g}$. The reservoir is initialized to implement a *contractive* τ, which ensures the existence and uniqueness of a solution of $\mathbf{x}(v) = \tau(\mathbf{u}(v), \mathbf{x}(\mathcal{N}(v)))\, \forall v \in V$, also

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

in case of mutual dependencies among states of different vertices (i.e. allowing to manage undirected and cyclic graphs), according to the Banach Contraction Principle. Under such condition, a stable encoding can be implemented using an iterated version of τ: $\mathbf{x}_t(v) = f(\mathbf{W}_{in}\mathbf{u}(v) + \sum_{v' \in \mathcal{N}(v)} \hat{\mathbf{W}}\mathbf{x}_{t-1}(v'))$. At each pass t of the encoding process on graph $\mathbf{g}$, the previous equation is applied to every $v \in V$, until the state of each vertex converges to a stable solution. In practice, the encoding is stopped when $\|\mathbf{x}_t(v) - \mathbf{x}_{t-1}(v)\|_2 \leq \epsilon \, \forall v \in V$, where ϵ is a small threshold. A null initial state is set for every vertex, i.e. $\mathbf{x}_0(v) = \mathbf{0} \in \mathbb{R}^{N_R}$. Overall, the encoding requires a single presentation of input graphs, with cost $O(N_R k|V|)$[5]. Contractivity of τ also bounds reservoir dynamics within a region characterized by interesting *Markovian* properties [4, 6]. Accordingly, the state computed for vertex v depends on its label and local topology, with more distant vertices having a progressively decreasing influence. Function τ is contractive[1] with respect to the state whenever $\exists C \in [0, 1)$ such that $\forall \mathbf{u} \in \mathbb{R}^{N_U}, \forall \mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{x}'_1, \ldots, \mathbf{x}'_k \in \mathbb{R}^{N_R}$: $\|\tau(\mathbf{u}, \mathbf{x}_1, \ldots, \mathbf{x}_k) - \tau(\mathbf{u}, \mathbf{x}'_1, \ldots, \mathbf{x}'_k)\| \leq C \, max_{1 \leq i \leq k} \|\mathbf{x}_i - \mathbf{x}'_i\|$. Assuming $tanh$ as activation function, a condition for the contractivity of τ in the Euclidean norm is given: $\forall \mathbf{u} \in \mathbb{R}^{N_U}$, $\forall \mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{x}'_1, \ldots, \mathbf{x}'_k \in \mathbb{R}^{N_R}$ we have that $\|\tau(\mathbf{u}, \mathbf{x}_1, \ldots, \mathbf{x}_k) - \tau(\mathbf{u}, \mathbf{x}'_1, \ldots, \mathbf{x}'_k)\|_2 = \|tanh(\mathbf{W}_{in}\mathbf{u} + \sum_{i=1}^{k} \hat{\mathbf{W}}\mathbf{x}_i) - tanh(\mathbf{W}_{in}\mathbf{u} + \sum_{i=1}^{k} \hat{\mathbf{W}}\mathbf{x}'_i)\|_2 \leq \|\sum_{i=1}^{k} \hat{\mathbf{W}}(\mathbf{x}_i - \mathbf{x}'_i)\|_2 \leq \|\hat{\mathbf{W}}\|_2 \|\sum_{i=1}^{k} (\mathbf{x}_i - \mathbf{x}'_i)\|_2 \leq \|\hat{\mathbf{W}}\|_2 \sum_{i=1}^{k} \|\mathbf{x}_i - \mathbf{x}'_i\|_2 \leq \|\hat{\mathbf{W}}\|_2 k \max_{i=1,\ldots,k} \|\mathbf{x}_i - \mathbf{x}'_i\|_2$. Contractivity is ensured when $\sigma = \|\hat{\mathbf{W}}\|_2 \, k < 1$, where σ is the *contraction coefficient*, controlling the degree of contractivity of reservoir dynamics. A simple ESN-like initialization for a GraphESN consists in a random setting of $\hat{\mathbf{W}}$, satisfying $\sigma < 1$ [2]. $\mathbf{W}_{in}$ weights are chosen from a uniform distribution in $[-w_{in}, w_{in}]$, where w_{in} is an *input scaling* parameter.

The encoding terminates yielding a stable structured state $\mathbf{x}(\mathbf{g})$. The SMF is then applied to extract a fixed-size $\mathcal{X}(\mathbf{x}(\mathbf{g}))$ from the set of reservoir states for the vertices of $\mathbf{g}$. In standard RecNNs this operation consists in selecting the state of the supersource of $\mathbf{g}$ (supersource SMF). Such choice, however, is limited to classes of graphs for which a supersource is defined, e.g. the root for trees. More in general, a mean SMF can be used, i.e. $\mathcal{X}(\mathbf{x}(\mathbf{g})) = |V|^{-1} \sum_{v \in V} \mathbf{x}(v)$, which averages the states of the vertices in $\mathbf{g}$. The linear readout is then applied to $\mathcal{X}(\mathbf{x}(\mathbf{g}))$, implementing an adaptive $\mathcal{T}_{out}$, i.e. $\mathbf{y}(\mathbf{g}) = \mathcal{T}_{out}(\mathcal{X}(\mathbf{x}(\mathbf{g})))$.

GraphESN with mean SMF has shown loosely comparable performance to those of (more costly) state-of-the-art models for SDs [5], including Graph Neural Networks (GNNs) [7], featured by *adaptive* contractive encodings. However, using fixed SMFs potentially discards useful information about the distribution of the vertices states relatively to the characteristics of the task [3]. In particular, the mean SMF implies an equal weighting of the states of vertices in the input graph, so that every vertex has the same influence on the output. We propose a different approach to weight the influence of every vertex on the output, proportionally to the extent to which it is helpful in characterizing the target.

[1] Note that the definition of contractivity given here is different from the one given in [5].

[2] Contractivity of τ may hold in any norm. Thus, larger values of σ can be used, yielding to convergent encodings even if violating the condition in the Euclidean norm.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

3 Exploiting Vertices Information Using weighted K-NN

Suppose we have a training set $\mathscr{T} = \{(\mathbf{g}, \mathbf{y}_{tg}(\mathbf{g})) : \mathbf{g} \in \mathcal{G}, \mathbf{y}_{tg}(\mathbf{g}) \in \mathbb{R}^{N_Y}\}$ [3], where $\mathcal{G}$ is a finite subset of $(\mathbb{R}^{N_U})^{\#k}$. Given the set of reservoir states computed by the GraphESN for the training graphs, we associate to every training vertex state in the reservoir space the target of the corresponding graph, i.e. $\mathbf{y}_{tg}(v) \equiv \mathbf{y}_{tg}(\mathbf{g}), \forall v \in V(\mathbf{g}), \forall \mathbf{g} \in \mathcal{G}$. Thereby, given an unseen input graph $\mathbf{g}$ and the reservoir states $\mathbf{x}(v)$ $\forall v \in V(\mathbf{g})$, an output value $\mathbf{y}(v)$ is computed for every vertex of $\mathbf{g}$. This is realized by resorting to a distance-weighted K-NN algorithm: $\mathbf{y}(v) = (\sum_{i=1}^{K} w_i^{(v)} \mathbf{y}_{tg}(v_i^N)) / \sum_{i=1}^{K} w_i^{(v)}$, where $v_1^N, \ldots, v_K^N$ are the training vertices corresponding to the K closest training reservoir states to $\mathbf{x}(v)$, and $w_i^{(v)} = (\|\mathbf{x}(v) - \mathbf{x}(v_i^N)\|_2^2)^{-1}$ is the inverse square of the Euclidean distance between $\mathbf{x}(v)$ and $\mathbf{x}(v_i^N)$. The final output of the model is computed by a weighted sum $\mathbf{y}(\mathbf{g}) = (\sum_{v \in V(\mathbf{g})} \alpha_r(v) \mathbf{y}(v)) / \sum_{v \in V(\mathbf{g})} \alpha_r(v)$, where $\alpha_r(v)$ represents the relative *relevance* of vertex v on the model output. Such relevance is assumed to be stronger for vertices whose states are in a region of the reservoir space corresponding to rather uniform training target values. This insight can be particularly appreciated e.g. in the context of predictive toxicology, where atoms of specific elements within a specific topology (e.g. halogens) could consistently have a stronger influence than others (e.g. hydrogens) on the toxicity of a molecule. Rather than treating in the same way the information from different atoms, modeling their relevance would result in a more suitable approach. Accordingly, $\alpha_r(v) = (\sum_{i=1}^{K} w_i^{(v)}) / \sum_{i=1}^{K} w_i^{(v)} (\mathbf{y}(v) - \mathbf{y}_{tg}(v_i^N))^2$, i.e. the inverse distance-weighted variance of the target associated to the vertices of the K neighbors of $\mathbf{x}(v)$. The computation of the output of a GraphESN in which the readout is implemented using K-NN with the weighting scheme described here (denoted as GraphESN-wnn), is shown in Fig. 1. Such readout implementation would combine in a flexible approach the Markovian organization of the reservoir space and the properties of the target task.

4 Experimental Results

We applied GraphESNs to a set of real-world classification tasks from a Chemical domain. The Predictive Toxicology Challenge (PTC) dataset [8] reports the carcinogenicity of 417 molecules relatively to four types of rodents: male mice (MM), female mice (FM), male rats (MR) and female rats (FR). Molecules are represented as undirected graphs, where vertices correspond to atoms and edges to bonds. Each vertex label contains a 1-of-m encoding of the atom element and its partial charge information, with label dimension of 24 and maximum degree $k = 4$. A classification task is defined for each type of rodents as in [9], with target $+1$ for active molecules and -1 for inactive molecules. The number of molecules with defined target is 336 (MM), 349 (FM), 344 (MR) and 351 (FR).

We considered reservoirs with 40% of connectivity, $\sigma \in \{0.5, 1, 2, 3\}$ and $w_{in} \in \{1, 0.1, 0.01\}$. For standard GraphESNs we used $N_R = 100$, mean SMF,

[3] E.g. $\mathbf{y}_{tg}(\mathbf{g}) \in \{-1, +1\}$ for classification tasks.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

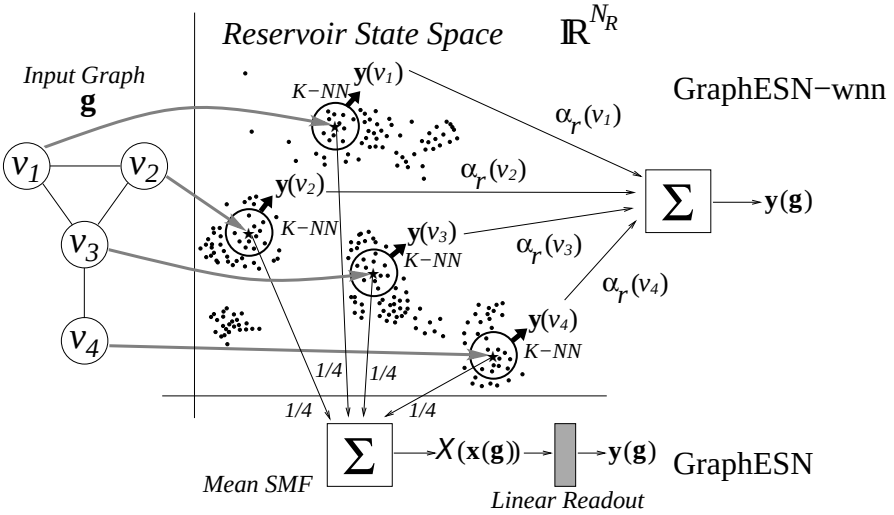

Fig. 1: Reservoir space of states computed for vertices of training graphs, and output processing for a test graph in GraphESN and GraphESN-wnn.

and readout trained using pseudo-inversion and ridge regression with regularization parameter $\lambda \in \{10^{-i} | 1 \leq i \leq 7\}$. For GraphESNs-wnn we used $N_R = 10$, with $K \in \{1, 5, 15, 30, 50\}$. For computational efficiency, the K-NN search was implemented using kd-trees (reducing the cost up to $O(log \sum_{\mathbf{g} \in \mathcal{G}} |V(\mathbf{g})|)$ for each vertex) and approximating the reservoir (search) space of the training patterns with the space of its first three principal components (PCs). Such approximation is particularly meaningful in light of the Markovian characterization of reservoir spaces due to contractive dynamics, with the first PCs collecting almost all the signal [6]. The performance accuracy was evaluated by 5-fold stratified cross-validation as in [9], with 5 independent (random guessed) reservoirs for every reservoir hyper-parametrization. For model selection, in each fold the training samples were split into a training and a (20%) validation set. Reservoir hyper-parametrizations yielding non convergent encodings were discarded. Table 1 compares the mean test performance of GraphESNs and GraphESNs-wnn after model selection on the validation set of the reservoir hyper-parameters and readout regularization. GraphESN-wnn outperform GraphESN on every PTC task with the exception of MR. The distance between the performances is particularly noteworthy for FM and FR. A comparison with state-of-the-art kernels for SDs

Model	*MM*	*FM*	*MR*	*FR*
GraphESN	62.87(±1.2)	60.40(±1.7)	59.43(±1.9)	64.44(±0.9)
GraphESN-wnn	63.04(±2.7)	63.32(±2.6)	58.02(±2.1)	67.37(±2.5)

Table 1: Mean test accuracies (%) on PTC for GraphESNs and GraphESNs-wnn, after model selection on reservoir and readout.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

is provided in Table 2. We considered the performance of Marginalized (MG), Optimal Assignment (OA) and Expected Match (EM) kernels on the same tasks [9]. By adopting a model selection criterion similar to [9], Table 2 reports the 'best classification' results (for reservoir guesses) after model selection of the readout regularization only. GraphESN-wnn compares well with all the kernels, with significantly better results in particular for FM and MR.

Model	*MM*	*FM*	*MR*	*FR*
GraphESN	68.45(±2.4)	64.77(±3.5)	65.99(±2.6)	68.95(±2.2)
GraphESN-wnn	69.65(±2.7)	67.91(±4.8)	67.43(±4.5)	69.25(±3.1)
MG-Kernel	69.05(±1.5)	64.76(±1.2)	62.50(±1.2)	70.09(±0.6)
OA-Kernel	67.87(±1.7)	65.33(±0.9)	63.39(±2.1)	70.37(±1.1)
EM-Kernel	66.97(±1.1)	64.47(±1.2)	60.84(±1.7)	68.95(±0.7)

Table 2: Mean best test accuracies (%) on PTC for GraphESN, GraphESN-wnn and kernels for SDs after model selection on the readout.

5 Conclusions

We have presented a method for weighting the contribution of vertices states in an instance-based readout of GraphESN. Preliminary experimental results on real-world tasks suggest the potentiality of the approach. Our investigations pave the way for further studies on adaptive modeling of the influence of each vertex state on the output. Preserving the extreme efficiency of fixed contractive RC encodings, more flexible methods for dealing with reservoir information can result in effective models for approaching relational and SD learning tasks.

References

[1] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127 – 149, 2009.

[2] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

[3] C. Gallicchio and A. Micheli. TreeESN: a preliminary experimental analysis. In *Proceedings of the ESANN 2010*, pages 333–338. d-side, 2010.

[4] B. Hammer, P. Tiňo, and A. Micheli. A mathematical characterization of the architectural bias of recursive models. Technical Report 252, Universitat Osnabruck, Germany, 2004.

[5] C. Gallicchio and A. Micheli. Graph echo state networks. In *Proceedings of the IJCNN 2010*, pages 2159–2166. IEEE, 2010.

[6] C. Gallicchio and A. Micheli. A markovian characterization of redundancy in echo state networks by PCA. In *Proceedings of the ESANN 2010*, pages 321–326. d-side, 2010.

[7] F. Scarselli, M. Gori, A.C.Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[8] C. Helma, R. King, and S. Kramer. The predictive toxicology challenge 2000-2001. In *Bioinformatics*, number 17, pages 107–108, 2001.

[9] H. Fröhlich, J.K. Wegner, and A. Zell. Assignment kernels for chemical compounds. In *Proceedings of the IJCNN 2005*, pages 913–918, 2005.