An Experimental Study on Learning with Good Edit Similarity Functions

Aurélien Bellet, Marc Sebban
Laboratoire Hubert Curien UMR CNRS 5516
University of Jean Monnet
42000 Saint-Etienne Cedex 2, France
Email: {aurelien.bellet, marc.sebban}@univ-st-etienne.fr

Amaury Habrard
Laboratoire d'Informatique Fondamentale UMR CNRS 6166
University of Aix-Marseille
13453 Marseille Cedex 13, France
Email: amaury.habrard@lif.univ-mrs.fr

***Abstract*—Similarity functions are essential to many learning algorithms. To allow their use in support vector machines (SVM), i.e., for the convergence of the learning algorithm to be guaranteed, they must be valid *kernels*. In the case of structured data, the similarities based on the popular *edit distance* often do not satisfy this requirement, which explains why they are typically used with k-nearest neighbor (k-NN). A common approach to use such *edit similarities* in SVM is to transform them into potentially (but not provably) valid kernels. Recently, a different theory of learning with (ϵ, γ, τ)-*good* similarity functions was proposed, allowing the use of non-kernel similarity functions. Moreover, the resulting models are supposedly *sparse*, as opposed to standard SVM models that can be unnecessarily dense. In this paper, we study the relevance and applicability of this theory in the context of string edit similarities. We show that they are naturally *good* for a given string classification task and provide experimental evidence that the obtained models not only clearly outperform the k-NN approach, but are also competitive with standard SVM models learned with state-of-the-art edit kernels, while being much sparser.**

***Keywords*-similarity function; linear program; sparsity; edit distance; structured data**

I. Introduction

A good similarity function[1] (distance, kernel, etc.) between objects is key to the effectiveness of many supervised and unsupervised learning methods, among which the popular k-nearest neighbor (k-NN), k-means and support vector machines (SVM). For this reason, a lot of research has been put into learning good similarity functions between numerical vectors, often in the form of a Mahalanobis distance (e.g., [1], [2], [3]), which is usually referred to as *metric learning*. When dealing with structured data (strings, trees, graphs), which is the case in this paper, a popular choice of similarity functions are those based on the *edit distance*, such as the string edit distance [4], the tree edit distance [5], the graph edit distance [6] or the local gapped alignment [7]. Roughly speaking, the edit distance between two objects is the minimum number of operations (insertion, deletion, substitution) required to transform an object into another. Because they involve more complex procedures, less work has been devoted to learning such similarities [8], [9], [10], [11].

For a similarity function to be a *kernel*, which is essential to the convergence of the SVM learning algorithm, it must be positive semi-definite (PSD) and symmetric. However, it can be proved that the above *edit similarities* are not PSD [12]. This explains why their use is often limited to k-NN classifiers. There exist two general approaches to go around this problem. The first one consists in building a function that may be a valid kernel under some circumstances. A popular choice is to plug the edit similarity e_s into a Gaussian-like kernel of the form $K(x, x') = e^{-t \cdot e_s(x,x')}$, where $t > 0$ is a parameter [12], [13], [14]. In practice, t is tuned by cross-validation and the value offering the best classification accuracy is chosen. Note that if this strategy is likely to ensure the resulting K to be valid, Cortes et al. [12] proved that there exist values of t for which K is not a kernel. Moreover, a slight modification of the value of t can lead to dramatic changes in performance. The edit functions proposed by Neuhaus & Bunke [15] also fall in the same category: they may work well in practice but their positive semi-definiteness has not been established. The second strategy consists in directly building a valid kernel from the edit similarity, usually by expressing it as a dot product (which ensures positive semi-definiteness). For instance, Bellet et al. [16] proposed the kernel $K(x, x') = \sum_{y \in \Sigma^*} e_s(x, y) \cdot e_s(x', y)$, where y is any string on the alphabet Σ. Despite the presence of an infinite sum, this kernel can be computed exactly (using a composition of stochastic edit transducers), but becomes intractable when the input strings are long.

On top of the difficulty of building valid kernels from edit similarities, a general drawback of SVM learning is that it produces rather *dense* models, i.e., the ratio between the number of support vectors (SV) and the total number of training examples tends to be large. The number of SV is actually known to grow linearly with the size of the training set [17]. In many situations, especially in the presence of redundant or irrelevant noisy features [18], a sparser model would lead to equal or higher classification accuracy, while saving many kernel evaluations at classification time. This explains why a decent amount of research has gone into

Preprint submitted to ICTAI 2011.

[1]Throughout this paper, we will use the term "similarity function" to refer to a function giving a measure of either closeness or distance.

developing *sparse* versions of SVM. A classic approach, often referred to as *1-norm SVM* [18], is to use L_1-norm regularization to obtain sparsity. However, a significant part of the handy SVM theory (such as some learning guarantees in the projection space given by a nonlinear kernel and the notion of reproducing kernel Hilbert space) does not hold for 1-norm SVM. For this reason, other methods aim at developing sparse versions of standard (L_2) SVM [19], [20], [21], [22], [23]. Unfortunately, most of these methods are expensive to compute and applying them on structured data, where kernel evaluations are often costly, would slow down the computation even more.

Recently, Balcan et al. [24], [25] introduced a theory of learning with so-called (ϵ, γ, τ)*-good similarity functions* that can be seen as a less restrictive and sparse alternative to SVM. It generalizes the notion of good kernel without requiring the similarity function to be PSD nor symmetric. Unlike in SVM theory, the learning guarantees are in terms of natural properties of the similarity function. Furthermore, learning is achieved through a linear program that can be solved efficiently and tends to induce sparse models.

In this paper, we study the applicability and relevance of the theory of Balcan et al. in the context of string edit similarities. We first show that standard edit similarities are naturally (ϵ, γ, τ)-good for a given classification task. Then, we provide experimental evidence that learning with Balcan et al.'s linear program induces a separator that (i) clearly outperforms the k-NN approach, and (ii) is competitive with the standard SVM approach, but is much sparser and provides sparsity control. Lastly, we provide results suggesting that the use of the exponential together with edit similarities may further improve accuracy.

The rest of the paper is organized as follows: in Section II we give a few notations and review the theory of Balcan et al. [25]. In Section III, we introduce two string edit similarities, explain how one can estimate their (ϵ, γ, τ)-goodness and, given a classification task, show that they are indeed good for that task. Section IV presents an experimental study on two datasets: a handwritten digit dataset (where images have been coded into strings) and a dataset of English and French words. We conclude this work by outlining a promising line of research on similarity-based learning.

II. Notations and Related Work

A. Notations

We assume we are given some labeled examples (x, ℓ) drawn from an unknown distribution P over $X \times \{-1, 1\}$, where X is the instance space. We want to learn a classifier $h : X \to \{-1, 1\}$ whose error rate is as low as possible, only using pairwise similarities between examples given by a similarity function $K : X \times X \to [-1, 1]$. We say that K is symmetric if $K(x, x') = K(x', x)$ for all $x, x' \in X$. K is a valid kernel if it is symmetric and PSD.

B. Learning with (ϵ, γ, τ)-Good Similarity Functions

In recent work, Balcan et al. [24], [25] highlight two major drawbacks of the kernel theory. First, the PSD requirement often rules out natural similarity functions for the problem at hand. As we have mentioned before, this is especially true for structured data (edit similarities make a striking example). Second, the notion of good kernel (which is related to the underlying margin in an implicit, possibly unknown projection space) is not often intuitive and does not provide insights into how to design a good kernel for a given problem. To overcome these limitations, Balcan et al. introduced a new theory of learning with any similarity function $K : X \times X \to [-1, 1]$. In particular, they propose the following, rather intuitive definition of a good similarity function:

Definition 1 (Balcan et al. [25])**.** *A similarity function K is an (ϵ, γ, τ)-good similarity function for a learning problem P if there exists a (random) indicator function $R(x)$ defining a (probabilistic) set of "reasonable points" such that the following conditions hold:*

1) *A $1 - \epsilon$ probability mass of examples (x, ℓ) satisfy*

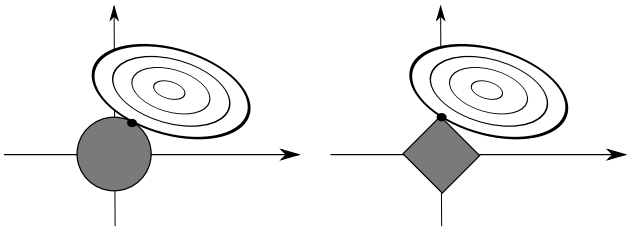
$$\mathbf{E}_{(x',\ell')\sim P}[\ell\ell' K(x, x')|R(x')] \geq \gamma$$

2) $\Pr_{x'}[R(x')] \geq \tau$.

Assuming that the proportion of positive and negative examples in R is equal, we can interpret the first condition as *most examples x are on average 2γ more similar to random reasonable examples of the same class than to random reasonable examples of the opposite class* and the second condition as *at least a τ proportion of the examples should be reasonable*.

Of course, this is just one way of defining what makes a similarity function good for a given problem. Other definitions are possible, for example those proposed by Wang et al. for unbounded similarity functions [26]. Yet Definition 1 is very interesting in two respects. First, it includes all good kernels as well as some non-PSD similarity functions. In that sense, this is a strict generalization of the notion of good kernel [25]. Second, these conditions are sufficient to be able to learn well. This is formalized by the following PAC-like learning guarantee.

Theorem 1 (Balcan et al. [25])**.** *[25] Let K be an (ϵ, γ, τ)-good similarity function for a learning problem P. Let $S = \{x'_1, x'_2, \ldots, x'_d\}$ be a (potentially unlabeled) sample of $d = \frac{2}{\tau}\left(\log(2/\delta) + 8\frac{\log(2/\delta)}{\gamma^2}\right)$ landmarks drawn from P. Consider the mapping $\phi^S : X \to \mathbb{R}^d$ defined as follows: $\phi^S_i(x) = K(x, x'_i)$, $i \in \{1, \ldots, d\}$. Then, with probability at least $1 - \delta$ over the random sample S, the induced distribution $\phi^S(P)$ in $\mathbb{R}^d$ has a linear separator of error at most $\epsilon + \delta$ relative to L_1 margin at least $\gamma/2$.*



(a) L_2 regularization (b) L_1 regularization

Figure 1. Geometric interpretation of L_2 and L_1 constraints (adapted from [27]). The L_1-norm tends to zero out coordinates, thus reducing dimensionality. This intuition also holds in the case of L_1 regularization.

Therefore, if we are given an (ϵ, γ, τ)-good similarity function for a learning problem P and enough (unlabeled) landmark examples, then with high probability there exists a low-error linear separator in the explicit "ϕ-space", which is essentially the space of the similarities to the d landmarks. As Balcan et al. mention, using d_u unlabeled examples and d_l labeled examples, we can efficiently find this separator $\boldsymbol{\alpha} \in \mathbb{R}^{d_u}$ by solving the following problem:[2]

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{d_l} \left[1 - \sum_{j=1}^{d_u} \alpha_j \ell_i K(x_i, x'_j) \right]_+ + \lambda \|\boldsymbol{\alpha}\|_1, \quad (1)$$

where $[1-z]_+ = \max(0, 1-z)$ is the hinge-loss and $\lambda \geq 0$ a parameter.

Problem (1) is a linear program (LP) and can be solved efficiently. It is essentially a 1-norm SVM with an empirical similarity map [25]. While one cannot enjoy some of the SVM learning guarantees when using 1-norm SVM, the theory of Balcan et al. offers a suitable framework for learning well using (1).

Note that the use of L_1 regularization on $\boldsymbol{\alpha}$ tends to induce sparse models, as opposed to L_2 regularization (see Figure 1). Indeed, a significant proportion of $\boldsymbol{\alpha}$'s coordinates will be set to zero during learning, resulting in the corresponding landmarks being ignored at classification time. In that sense, by solving (1), one performs a selection of "relevant prototypes", which one will compare the test examples to in order to classify them, ignoring the other landmarks. Therefore, one does not need to know in advance the set of reasonable points: R is automatically worked out during learning (it corresponds to the set of landmarks that have non-zero coordinates in the resulting separator $\boldsymbol{\alpha}$). Moreover, the parameter λ offers a direct way to the sparsity of the solution (the larger λ, the sparser $\boldsymbol{\alpha}$ tends to be).

III. Are Edit Similarities (ϵ, γ, τ)-Good?

In this section, we check if edit similarities are (ϵ, γ, τ)-good. We propose to undertake this study using the standard string edit distance, known as the *Levenshtein distance* [28], and a stochastic learned version [10]. The Levenshtein distance e_L is defined as follows.

Definition 2. *The Levenshtein distance $e_L(x, x')$ between two strings x and x' (of length m and n respectively) is the minimum number of edit operations to transform x into x'. The allowable operations are insertion, deletion and substitution of a single symbol.*

e_L can be computed in $O(m \cdot n)$ time using dynamic programming. Instead of only counting the minimum number of required operations, we can use a matrix M that provides a cost (or probability) for each edit operation, and define the *generalized* Levenshtein distance as being the sequence of edit operations of minimum cost (or maximum probability). There exists a few methods to learn M for a given string classification task (e.g., [8], [9], [10]) which often take the form of probabilistic models that allow us to compute the conditional edit probability $p_e(x'|x)$ that x is transformed into x' using edit operations.

Whichever edit similarity function is used, it raises a crucial question: is it (ϵ, γ, τ)-good? Looking at Definition 1, we can easily estimate ϵ, γ and τ using a randomly selected set of examples. Let us illustrate this on the well-known NIST Special Database 3 of the National Institute of Standards and Technology, describing a set of handwritten characters in the form of 128x128 bitmap images. Each instance can be represented as a string of Freeman codes [29] following the contour (starting from the top left pixel of the digit), as shown in Figure 2.

We consider the Levenshtein distance $e_L(x, x')$ and the edit probability function $p_e(x'|x)$ learned using the method presented in [10],[3] known to perform well on this task. Note that we actually used $-e_L$ so that both similarities express a measure of closeness, making the comparison easier. We also normalized them.[4] In the following, they are referred to as $\tilde{e_L}$ and $\tilde{p_e}$.

Since we do not know the set of reasonable points before learning, we fix $\tau = 1$ (i.e., all points are considered reasonable) and estimate ϵ as a function of γ. In order to analyze these parameter estimates in different contexts, we randomly selected 500 instances of each class and estimated the goodness of the similarities for each binary problem. For the sake of clarity, we will only discuss the goodness curves for "0 vs. 1" and "0 vs. 8", shown in Figure 3. The interpretation (given by Definition 1) is that a margin

[2]The original formulation proposed in [25] was actually L_1-constrained. We transformed it into an equivalent L_1-regularized one.

[3]We do not go into details of this learning method, but note that it is based on a likelihood maximization process and not on a classification rate optimization. More information and online software are available at http://labh-curien.univ-st-etienne.fr/SEDiL/

[4]This is because Definition 1 requires the similarity function to be in $[-1, 1]$. We thus normalized our edit similarities to have zero-mean and unit-variance, followed by the thresholding to 1 and -1 of the values > 1 and < -1 respectively. Note that there may be better normalizations but this is not the purpose of this work.

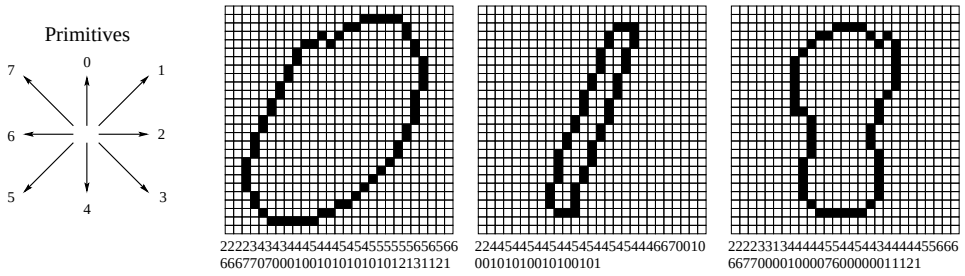


Figure 2. Three digits (0, 1, 8) and their respective string representation.

of γ leads to an ϵ proportion of examples violating the margin. For the "0 vs. 1" problem, shown in Figure 3(a), both similarities achieve good margin while sustaining few violations. We see that the learned similarity $\tilde{p_e}$ behaves slightly better. The "0 vs. 8" problem is a harder task: as shown in the example from Figure 2, an eight may look similar to a zero since we only encode the contour of the digits. Figure 3(b) reflects the difficulty of the task, since margin violations are always higher for a given γ that in the "0 vs. 1" case. For the "0 vs. 8" task, the learned similarity provides an important improvement over the standard edit distance: it achieves very few violations for small values of the margin.

To sum up, we see that decent values for γ and ϵ are achieved even without selecting an appropriate subset R of reasonable points. Note that we observed the same behavior for all binary problems in the dataset. Therefore, the edit similarities fit Definition 1, and the learning guarantees of Theorem 1 hold. Furthermore, we get better values with $\tilde{p_e}$ than with $\tilde{e_L}$, which suggests that $\tilde{p_e}$ should lead to better generalization performance (we will see that it is indeed the case in the experiments).

IV. Experimental Results

In this section, we provide experimental evidence that learning with good edit similarities outperforms a k-NN approach and is competitive with a standard SVM approach, while inducing much sparser models. We compare the following approaches: (i) learning with (1) using $K(x, x') = \tilde{e_L}(x, x')$, (ii) learning with (1) using $K(x, x') = \tilde{p_e}(x'|x)$, (iii) SVM learning using $K(x, x') = e^{-t \cdot e_L(x, x')}$, the baseline SVM approach based on e_L, (iv) SVM learning using $K(x, x') = e^{\frac{1}{2}t(\log p_e(x'|x) + \log p_e(x|x'))}$, the equivalent of (iii) when using edit probabilities,[5] (v) 1-NN using $e_L(x, x')$, and (vi) 1-NN using $-p_e(x'|x)$. We choose LIBSVM [30] as our SVM implementation, which takes a one-versus-one approach for multi-class classification. We thus take the same principle for multi-class classification using (1). Note that in this series of experiments, we simply take the training examples to be the landmarks. Therefore, the different algorithms use strictly the same input information (that is, similarity measurements between training examples), allowing a very fair comparison.

In the following, we present results on the multi-class handwritten digit classification task (already used in Section III) and on a dataset of English and French words.

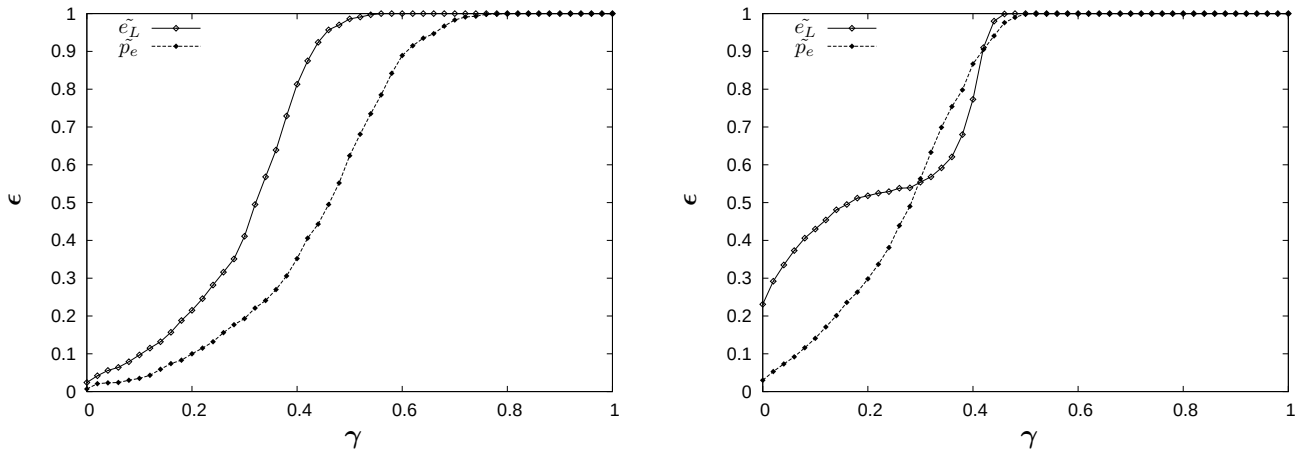
A. Handwritten digit classification

Using the handwritten digit classification dataset introduced in Section III, we first aim at evaluating the performance of the models obtained with the different methods. We use 40 to 6,000 training examples, reporting the results under 5-fold cross-validation. The parameters of the models, such as λ for approaches (i-ii) or C and t for approaches (iii-iv), are tuned by cross-validation on an independent set of examples, always selecting the value that offers the best classification accuracy.

1) Accuracy and sparsity: Classification accuracy is reported in Figure 4(a). All methods perform essentially the same, except for 1-NN that is somewhat weaker. Note that the methods based on the learned edit probabilities are, as expected, more accurate than those based on the standard edit distance. Figure 4(b) shows the average size of a binary model for approaches (i-iv), i.e., the number of training examples (reasonable points or support vectors) involved in the classification of new examples. Approaches (i-ii) are 5 to 6 times sparser than (iii-iv), which confirms that learning with (1) leads to much sparser models than SVM learning.

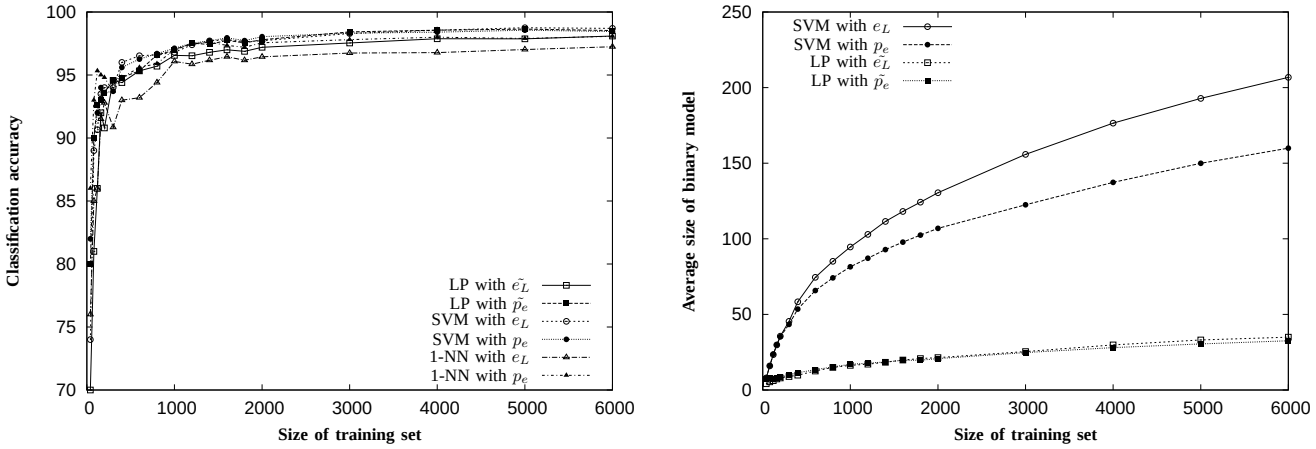
2) Influence of the parameters: We now study the influence of parameters on the accuracy and sparsity of the models. Results are obtained on 4,000 training examples. The influence of λ on the models learned with (1) is shown in Figure 5. The results confirm that λ can be conveniently used to control the sparsity of the models thanks to L_1 regularization. It is worth nothing that while the best accuracy is obtained with relatively small values ($\lambda \in [1; 10]$), one can get even sparser but still very accurate models with larger values ($\lambda \in [10; 200]$). This is especially true when using $\tilde{p_e}$. Therefore, one can learn a model using (1) that is just slightly less accurate than the corresponding SVM model while being 10 to 18 times sparser. This can certainly

[5]Since p_e is not symmetric, the kernel is made symmetric by adding $p_e(x'|x)$ and $p_e(x|x')$.



(a) 0 vs. 1 (b) 0 vs. 8

Figure 3. Estimation of ϵ as a function of γ for $\tilde{e_L}$ and $\tilde{p_e}$ on two handwritten digits binary classification tasks.



(a) Accuracy (b) Sparsity

Figure 4. Digit dataset: classification accuracy and sparsity results for methods (i-vi) over a range of training set sizes.

be a useful feature in applications where data storage is limited and/or high classification speed is required. One might wonder whether the SVM parameter C can also be used to improve the sparsity of the models in the same way as λ. In order to assess this, we try a wide range of C values and record the average sparsity of the models. SVM could not match the sparsity of the models learned with (1).[6] The best average size for a binary model was greater than 100, i.e., more than 2 times bigger than the worst model size obtained with (1). This results from the tendency of L_2 regularization to select models that put small weights on many coordinates. Lastly, we investigate the influence of parameter t on the performance of the SVM models. Results are shown in Figure 6 (a log-scale is used to allow a better appreciation of the variations). Both the accuracy and the sparsity of the SVM models are heavily dependent on t: only a narrow range of t values (probably those achieving positive semi-definiteness) allows for accurate and acceptably sized models. Furthermore, this range appears to be specific to the edit similarity used. Therefore, t must be tuned very carefully, which represents a waste of time and data.

[6]We do not report the details of the results due to the limitation of space.

3) Reasonable points: Remember that we classify a new example by computing similarity scores to the set of reasonable points only, using the vector $\boldsymbol{\alpha}$ to weight each score. Unlike support vectors (that lie on the margin), reasonable points rather correspond to discriminative prototypes reflecting the diversity of both classes and may provide valuable *a posteriori* information on the underlying classification problem. In the context of digits, Figure 7 gives an example of a set of 8 reasonable points for a model learned on

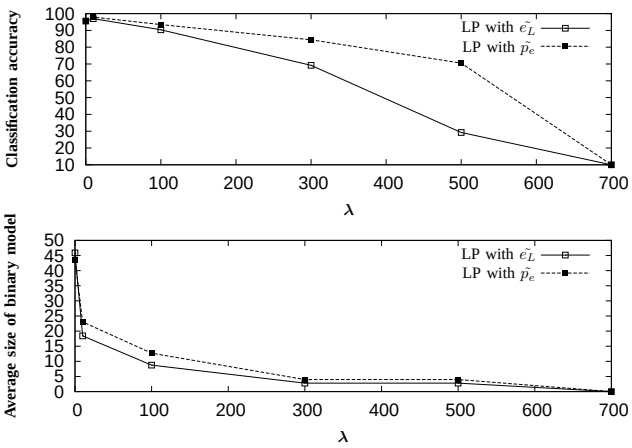


Figure 5. Digit dataset: classification accuracy and sparsity results with respect to the value of λ.

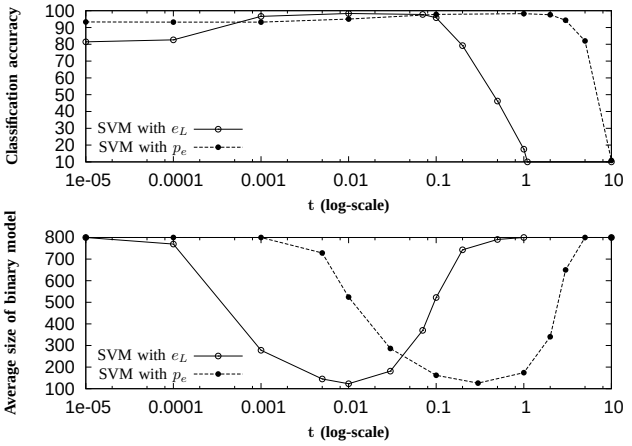


Figure 6. Digit dataset: classification accuracy and sparsity results with respect to the value of t (log-scale).

zeros and eights with $\tilde{p_e}$.[7] They are rather discriminative (the selected members of each class do not look like any member of the other class) and reflect the within-class variability (they are of various shapes, sizes and orientations).

B. English and French words classification

In this second series of experiments, we choose a fairly different and harder task: classifying words as either English or French. We use the 2,000 top words lists from Wiktionary.[8] We only consider unique words (i.e., not appearing in both lists) of length at least 4, and we also get rid of accent and punctuation marks. We end up with about 1,300 words of each language. We keep 600 words aside for cross-validation of parameters, 400 words to test the models and use the remaining words to learn the models.

[7]We used a large λ in order to get a small set, making the analysis easier.

[8]http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists

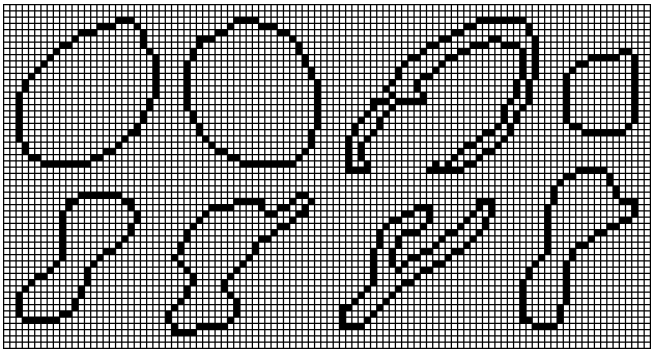


Figure 7. Example of a set of reasonable points for the task of classifying zeros and eights.

1) Accuracy and sparsity: Classification accuracy is reported in Figure 8(a). Note that this binary task is significantly harder than the one presented in the previous section, and that once again, models based on p_e perform better than those based on e_L. Models learned with (1) clearly outperform k-NN, while SVM models are the most accurate. Sparsity results are shown in Figure 8(b). The gap in sparsity between models learned with (1) and SVM models is even more marked on this task: the number of support vectors grows linearly with the number of training examples.

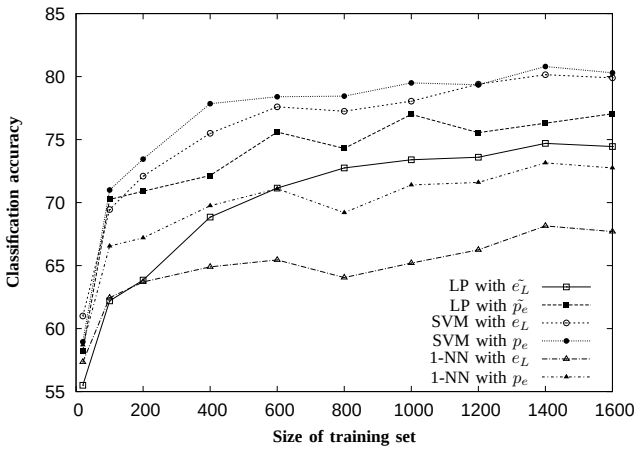
2) Reasonable points: Table I provides an example of a set of 17 reasonable points in the context of word classification, obtained with $\tilde{p_e}$.[9] Working on real words makes the analysis easier. The small set shown in Table I actually carries a lot of discriminative patterns (shown in Table II along with their number of occurrences in each class over the entire dataset). For example, words ending with *ly* correspond to English words, while those ending with *que* characterize French words. Note that Table I also reflects the fact that English words are shorter on average (6.99) than French words (8.26) in the dataset. The selected English (resp. French) reasonable points are significantly shorter (resp. longer) than the average (mean of 5.56 and 9.75 resp.), which allows better discrimination.

C. Role of the exponential

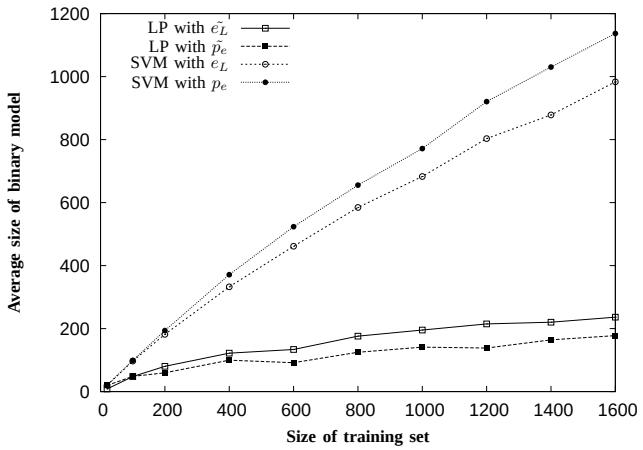
Finally, we assess the role of the exponential. Indeed, in addition to allow the similarity to be PSD for some values of t, taking the exponential of the edit similarity can also be seen as introducing some nonlinearity to further separate distant examples while moving closer neighboring examples. In order to see whether using the exponential when learning with (1) improves the models, we test two approaches: (vii) learning with (1) using $K(x, x') = \exp(-e_L(x, x'))$, (viii) learning with (1) using $K(x, x') = \exp(p_e(x'|x))$. Figure 9 gives the accuracy results for the word dataset.[10] We see that

[9]As in the digit classification setup, we used a large λ value.

[10]Results on the digit dataset are not shown because they yielded very similar results to those presented in Figure 4.



(a) Accuracy



(b) Sparsity

Figure 8. Word dataset: classification accuracy and sparsity results for methods (i-vi) over a range of training set sizes.

English					French			
previous	hoped	liked	waited	ways	parcours	positions	economiques	critique
works	hardly	whole	showed		continue	informatique	partenaire	exportations

Table I
EXAMPLE OF A SET OF 17 REASONABLE POINTS.

	w	y	k	q	tion	gh	ai	ed$	ly$	es?$	ques?$	^h
English	146	144	83	14	12	34	39	151	51	265	0	62
French	7	19	5	72	65	0	114	51	0	630	43	14

Table II
SOME DISCRIMINATIVE PATTERNS EXTRACTED FROM THE REASONABLE POINTS OF TABLE I (^: START OF WORD, $: END OF WORD, ?: 0 OR 1 OCCURRENCE OF PRECEDING LETTER).

using the exponential provides improvement over the results of Figure 8(a) without having to tune a parameter t. The sparsity of the models is essentially the same as that shown on Figure 8(b) (without exponential). However, tuning t anyway may further improve the similarity and allow models learned with (1) to match the accuracy of SVM.

V. CONCLUSION AND FUTURE WORK

In this work, we applied Balcan et al.'s framework to string edit similarities. We showed through a simple procedure that edit similarities are (ϵ, γ, τ)-good for a given classification task. We then provided experimental evidence on two radically different datasets (a task where strings are derived from images and a natural string classification task) that this approach performs better than k-NN and is competitive with SVM, while inducing sparser models. This feature can be crucial in many applications due to the generally high computational cost of edit similarities. Lastly, we highlighted the potential positive role of the exponential in the context of edit similarity learning. Our approach could easily be extended to tree classification using tree edit similarities [5], [11]. A natural future work could include learning the edit costs so that the resulting edit similarity is

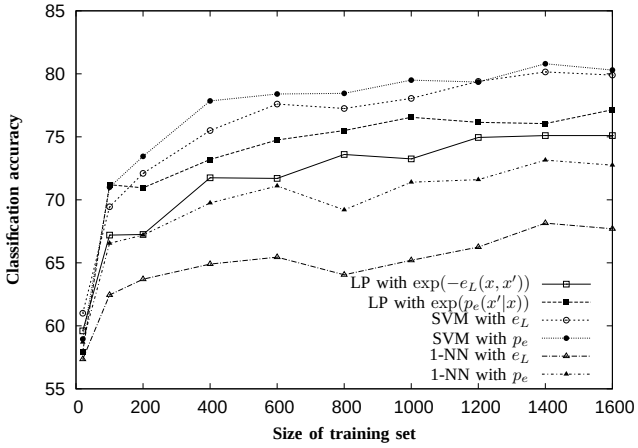


Figure 9. Word dataset: classification accuracy results for methods (iii-viii) over a range of training set sizes.

optimized to be "as (ϵ, γ, τ)-good as possible" for a given classification problem.

ACKNOWLEDGMENT

We would like to acknowledge support from the ANR LAMPADA 09-EMER-007-02 project and the PASCAL 2 Network of Excellence.

REFERENCES

[1] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood Components Analysis," in *Adv. in Neural Inf. Proc. Sys. (NIPS)*, vol. 17, 2004, pp. 513–520.

[2] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2007, pp. 209–216.

[3] K. Q. Weinberger and L. K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," *J. of Mach. Learn. Res. (JMLR)*, vol. 10, pp. 207–244, 2009.

[4] R. A. Wagner and M. J. Fischer, "The String-to-String Correction Problem," *Journal of the ACM*, vol. 21, no. 1, pp. 168–173, 1974.

[5] P. Bille, "A survey on tree edit distance and related problems," *Theoretical Computer Science*, vol. 337, no. 1-3, pp. 217–239, 2005.

[6] X. Gao, B. Xiao, D. Tao, and X. Li, "A survey of graph edit distance," *Pattern Analysis & Applications*, vol. 13, no. 1, pp. 113–129, 2010.

[7] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, 1981.

[8] E. S. Ristad and P. N. Yianilos, "Learning String-Edit Distance," in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 5, 1998, pp. 522–532.

[9] M. Bilenko and R. J. Mooney, "Adaptive Duplicate Detection Using Learnable String Similarity Measures," in *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 2003, pp. 39–48.

[10] J. Oncina and M. Sebban, "Learning Stochastic Edit Distance: application in handwritten character recognition," *Pattern Recognition*, vol. 39, no. 9, pp. 1575–1587, 2006.

[11] M. Bernard, L. Boyer, A. Habrard, and M. Sebban, "Learning probabilistic models of tree edit distance," *Pattern Recognition*, vol. 41, no. 8, pp. 2611–2629, 2008.

[12] C. Cortes, P. Haffner, and M. Mohri, "Rational Kernels: Theory and Algorithms," *J. of Mach. Learn. Res. (JMLR)*, vol. 5, pp. 1035–1062, 2004.

[13] H. Li and T. Jiang, "A class of edit kernels for SVMs to predict translation initiation sites in eukaryotic mRNAs," in *Proc. of the Annual Int. Conf. on Research in Comp. Molecular Biology*, 2004, pp. 262–271.

[14] H. Saigo, J.-P. Vert, N. Ueda, and T. Akutsu, "Protein homology detection using string alignment kernels," *Bioinformatics*, vol. 20, no. 11, pp. 1682–1689, 2004.

[15] M. Neuhaus and H. Bunke, "Edit distance-based kernel functions for structural pattern classification," *Pattern Recognition*, vol. 39, pp. 1852–1863, 2006.

[16] A. Bellet, M. Bernard, T. Murgue, and M. Sebban, "Learning state machine-based string edit kernels," *Pattern Recognition*, vol. 43, no. 6, pp. 2330–2339, 2010.

[17] I. Steinwart, "Sparseness of Support Vector Machines," *J. of Mach. Learn. Res. (JMLR)*, vol. 4, pp. 1071–1105, 2003.

[18] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, "1-norm Support Vector Machines," in *Adv. in Neural Inf. Proc. Sys. (NIPS)*, vol. 16, 2003.

[19] C. J. Burges, "Simplified Support Vector Decision Rules," in *Proc. of the Int. Conf. on Machine Learning (ICML)*, 1996, pp. 71–77.

[20] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proc. of the SIAM Int. Conf. on Data Mining (ICDM)*, 2001.

[21] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *J. of Mach. Learn. Res. (JMLR)*, vol. 1, pp. 211–244, 2001.

[22] M. Wu, B. Scholkopf, and G. Bakir, "A Direct Method for Building Sparse Kernel Learning Algorithms," *J. of Mach. Learn. Res. (JMLR)*, vol. 7, pp. 603–624, 2006.

[23] T. Joachims and C.-N. J. Yu, "Sparse kernel SVMs via cutting-plane training," *Machine Learning*, vol. 76, no. 2-3, pp. 179–193, 2009.

[24] M.-F. Balcan and A. Blum, "On a Theory of Learning with Similarity Functions," in *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2006, pp. 73–80.

[25] M.-F. Balcan, A. Blum, and N. Srebro, "Improved Guarantees for Learning via Similarity Functions," in *Proc. of the Annual Conf. on Learning Theory (COLT)*, 2008, pp. 287–298.

[26] L. Wang, C. Yang, and J. Feng, "On Learning with Dissimilarity Functions," in *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2007, pp. 991–998.

[27] F. Bach and G. Obozinski, "Sparse Methods for Machine Learning: Theory and Algorithms," ECML/PKDD Tutorial, 2010, http://www.di.ens.fr/~fbac/ecml2010tutorial.

[28] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics-Doklandy*, vol. 6, pp. 707–710, 1966.

[29] H. Freeman, "Computer Processing of Line-Drawing Images," *ACM Computing Surveys*, vol. 6, pp. 57–97, 1974.

[30] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.