Learning Good Edit Similarities with Generalization Guarantees

Aurélien Bellet[1] Amaury Habrard[2] Marc Sebban[1]

[1]Laboratoire Hubert Curien, UMR CNRS 5516, Université de Saint-Etienne
{aurelien.bellet,marc.sebban}@univ-st-etienne.fr

[2]Laboratoire d'Informatique Fondamentale, UMR CNRS 6166, Aix-Marseille Université
amaury.habrard@lif.univ-mrs.fr

ECML PKDD '11, Athens

Introduction: Similarity Learning

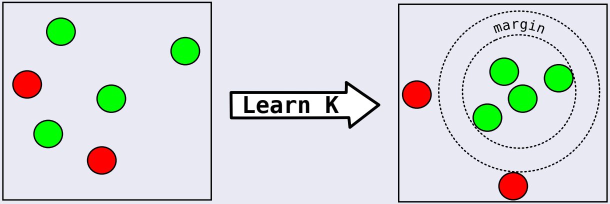
Similarity functions in classification

- Common approach in supervised classification: learn to classify objects using a **pairwise similarity (or distance) function**.
- Successful examples: k-Nearest Neighbor (k-NN), Support Vector Machines (SVM).
- Best way to get a "good" similarity function for a specific task: **learn it from data!**

Similarity learning

Similarity learning overview

Learning a similarity function $K(x, x')$ implying a new instance space where the performance of a given algorithm is improved.



Very popular approach for numerical data

Learn the transformation matrix A of a Mahalanobis distance:

$$d(x, y) = (x - y)A^T A(x - y)$$

Goals of our work

Goals of our work

1. Learn a similarity function for **string** classification;
2. which is **guaranteed to generalize well** to new examples;
3. and **provably induce low-error classifiers** for the task at hand.

Building block

Make use of the **theory of learning with (ϵ, γ, τ)-good similarity functions** (Balcan et al.).

(ϵ, γ, τ)-good similarity functions

Definition

Balcan et al. (2006, 2008) wanted a definition of **good similarity function** that

1. talks in terms of natural, direct properties;
2. includes the usual notion of good kernel, without PSD requirement;
3. provides guarantees for learning.

Definition (Balcan et al., 2008)

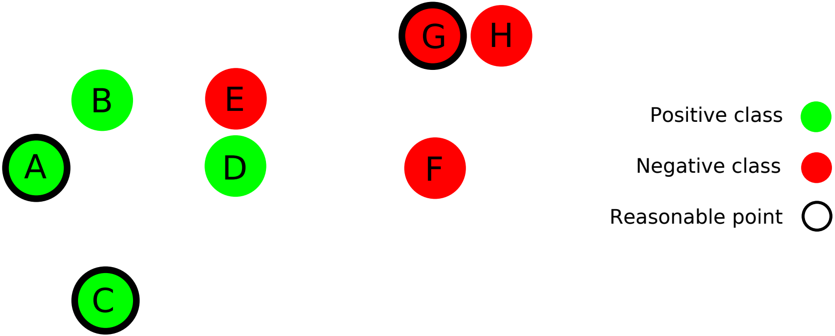
A similarity function $K \in [-1, 1]$ is an (ϵ, γ, τ)-**good similarity function** for a learning problem P if there exists an indicator function $R(x)$ defining a set of "reasonable points" such that the following conditions hold:

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

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

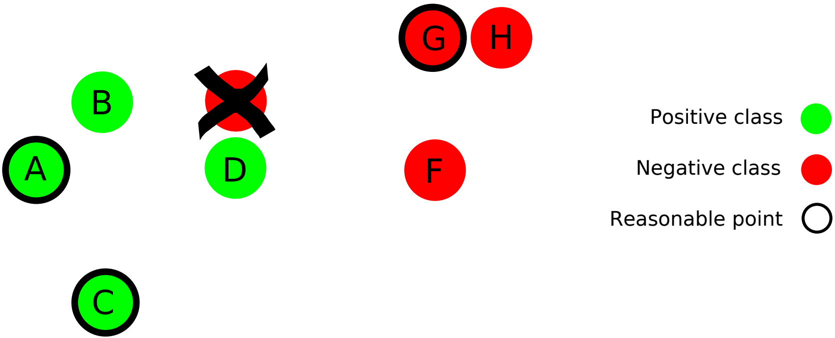
2. $\Pr_{x'}[R(x')] \geq \tau$. $\quad \epsilon, \gamma, \tau \in [0, 1]$

Intuition behind the definition



$K(x, x') = -\|x - x'\|_2$ is good with $\boldsymbol{\epsilon = 0}$, $\boldsymbol{\gamma = 0.03}$, $\boldsymbol{\tau = 3/8}$

Intuition behind the definition

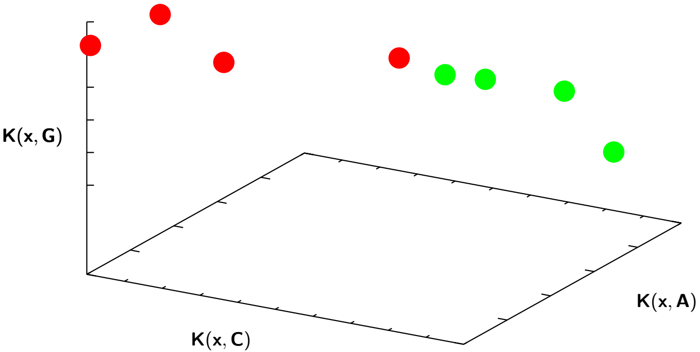


$K(x, x') = -\|x - x'\|_2$ is good with $\boldsymbol{\epsilon = 1/8}$, $\boldsymbol{\gamma = 0.12}$, $\boldsymbol{\tau = 3/8}$

Implications for learning

Strategy

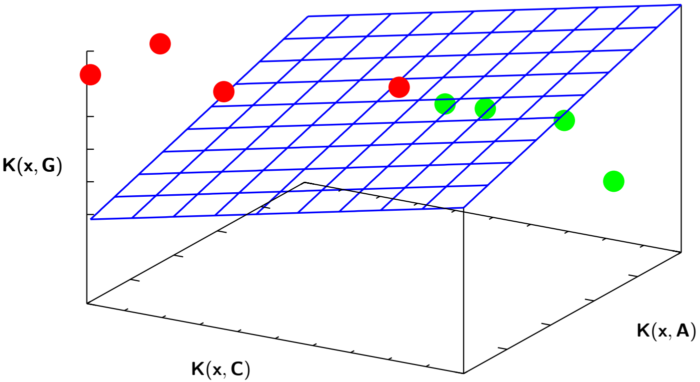
Each example is mapped to the space of "the similarity scores with the reasonable points".



Implications for learning

Theorem (Balcan et al., 2008)

Given K is (ϵ, γ, τ)-good, there exists a linear separator $\boldsymbol{\alpha}$ in the above-defined projection space that has error close to ϵ at margin γ.



Learning rule

Learning the separator α with a **linear program**

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

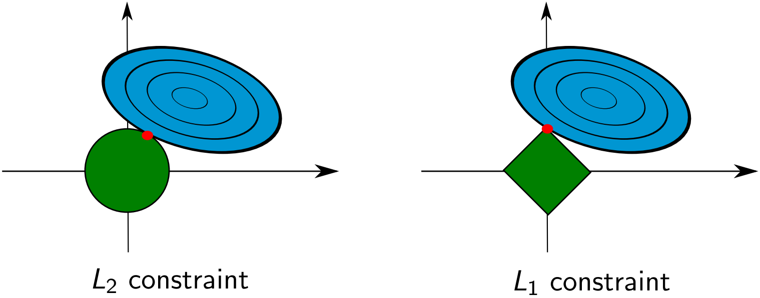
where $[1 - c]_+ = max(1 - c, 0)$ is the hinge-loss.

Automatic selection of reasonable points

The best set of reasonable points is automatically chosen among the examples thanks to the $\mathbf{L_1}$**-regularization** on α.

L_1-norm and Sparsity

- Why does L_1-norm constraint/regularization induce sparsity? Geometric interpretation:



- Examples corresponding to non-zero coordinates in α are the **reasonable points**.

Learning good edit similarities

Motivations for our work

Two main motivations for our work:

Motivation 1

The definition of (ϵ, γ, τ)-good similarity function gives us **a natural objective to optimize**:

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

If we satisfy this, then we can find a low-error classifier for the task.

Motivation 2

Similarity functions for **structured data** (strings, trees...) are often **not PSD**. Not so easy to use in SVM.

The string edit distance

Standard (Levenshtein) edit distance $\mathbf{e_L}$ between two strings x and y: **minimum number of operations** to transform x into y. Allowable operations are **insertion**, **deletion** and **substitution** of symbols.

Example 1

$$e_L(abb, aa) = C(b, a) + C(b, \$) = 1 + 1 = 2$$

Generalized version $\mathbf{e_C}$: use a **cost** for each operation.

Example 2

C	\$	a	b
\$	-	1	0
a	1	-	3
b	0	3	-

$$\Longrightarrow e_C(abb, aa) = C(b, \$) + C(b, \$) + C(\$, a) = 1$$

\$: empty symbol

A feel of the state-of-the-art in edit cost learning

There exists a decent amount of literature on **learning edit costs** (or probabilities) **from data**. See Ristad & Yianilos (1998), Bilenko & Mooney (2003), Oncina & Sebban (2006), Takasu (2009)...

Drawbacks of the state-of-the-art

- most of them use an **iterative procedure**, which can be costly.
- they often make use of **positive pairs only** (i.e., moving examples of the same class "closer" together). What about negative pairs?
- above all, they are **not learned to be (ϵ, γ, τ)-good**.
  $\hookrightarrow$ they are not guaranteed to perform well for the task at hand.

Our edit similarity function

An iterative approach is usually needed because **the optimal edit script** (= best sequence of operations) **depends on the edit costs**.
$\hookrightarrow$ Solution: **define a different type of edit function**!

Definition of e_G

$$e_G(x, x') = \sum_{0 \leq i,j \leq A} C_{i,j} \times \#_{i,j}(x, x')$$

where A is the size of the alphabet, C the edit cost matrix and $\#_{i,j}(x, x')$ the number of times the operation (i, j) appears in the Levenshtein script.
We will optimize:

Definition of K_G

$$K_G(x, x') = 2e^{-e_G(x,x')} - 1 \in [-1, 1]$$

Optimize the goodness

Optimizing the (ϵ, γ, τ)-goodness of K_G is difficult for two reasons:

1. Optimizing the definition directly would result in nonconvexity (summing/subtracting up exponential terms).
2. We do not know the set of reasonable points R at this point.

Solution to the first issue

Optimize a criterion that bounds goodness:

$$\mathbf{E}_{(x,l)}\left[\mathbf{E}_{(x',\ell')}\left[\left[1-\ell\ell' K_G(x,x')/\gamma\right]_+ |R(x')\right]\right] \leq \epsilon'.$$

Interpretation: goodness is required with respect to each reasonable point (instead of considering the average similarity to these points).

Optimize the goodness ctd

What about the second issue?

- Taking all points as reasonable is **not** a good idea (defines an overconstrained problem).
- Reasonable points can be seen as good representatives of a subset of the class examples.

Solution to second issue

Use an indicator matching function $f_{land} : T \times S_L \rightarrow \{0, 1\}$ that associates each training example in T with N_L examples in S_L.

In our experiments, we matched each example with its P nearest-neighbors of same class and its P farthest-neighbor of opposite class in T using the Levenshtein distance.

Convex formulation of the problem

Recall the underlying idea

Moving closer pairs of the same class and further those of opposite class.

Our convex formulation

$$\min_{C,B_1,B_2} \quad \frac{1}{N_T N_L} \sum_{\substack{1 \leq i \leq N_L, \\ 1 \leq j \leq N_T, \\ f_{land}(x_i, x'_j)=1}} V(C, z_i, z'_j) \quad + \quad \beta \|C\|^2$$

$$s.t. \quad V(C, z_i, z'_j) = \begin{cases} [B1 - e_G(x_i, x'_j)]_+ \text{ if } \ell_i \neq \ell'_j \\ [e_G(x_i, x'_j) - B2]_+ \text{ if } \ell_i = \ell'_j \end{cases}$$

$$B_1 \geq -\log(\tfrac{1}{2}), \quad 0 \leq B_2 \leq -\log(\tfrac{1}{2}), \quad B_1 - B_2 = \eta_\gamma$$

$$C_{i,j} \geq 0, \quad 0 \leq i, j \leq A$$

Parameters:

- β: regularization parameter on the edit costs.
- η_γ: the "desired margin".

Learning guarantees

Bounding the true error of an edit model C

$$L(C) = \mathbf{E}_{z_k, z'_j}[V(C, z_k, z'_j)]$$

Uniform stability [Bousquet et al. 02, Jin et al. 09]

Idea: **study the impact of a small change in the training sample**.

$$\forall (T, z), |T| = N_T, \forall i, \sup_{z_1, z_2} |V(C_T, z_1, z_2) - V(C_{T^{i,z}}, z_1, z_2)| \leq \frac{\kappa}{N_T}$$

$T^{i,z}$ set obtained by replacing $z_i \in T$ by z

$\hookrightarrow$ Generalization bound

Convergence and learning guarantees

Theorem: Algorithm has a uniform stability in κ/N_T

$$\kappa = \frac{2(2+\alpha)W^2}{\beta\alpha}$$

W is a bound on the string sizes; $0 \leq \alpha \leq 1$ such that $N_L = \alpha N_T$.

Theorem: Generalization bound - Convergence in $O(\sqrt{1/N_T})$

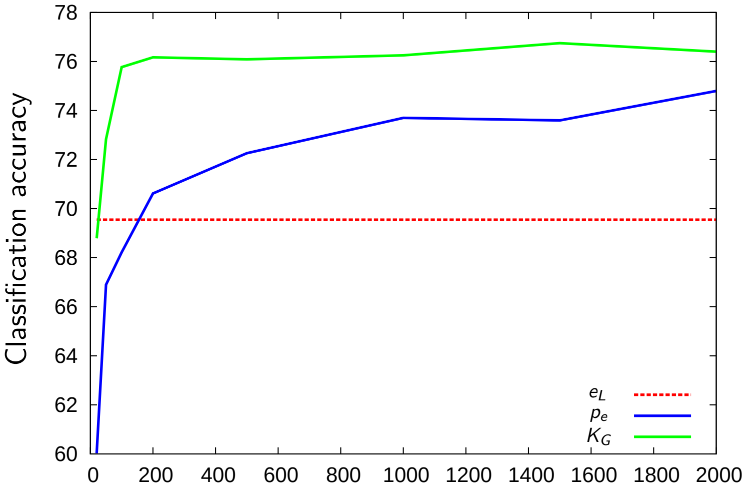
$$L(C) < \hat{L}(C) + 2\frac{\kappa}{N_T} + (2\kappa + B)\sqrt{\frac{\ln(2/\delta)}{2N_T}}$$

$\hat{L}(C)$: empirical error on learning sample.

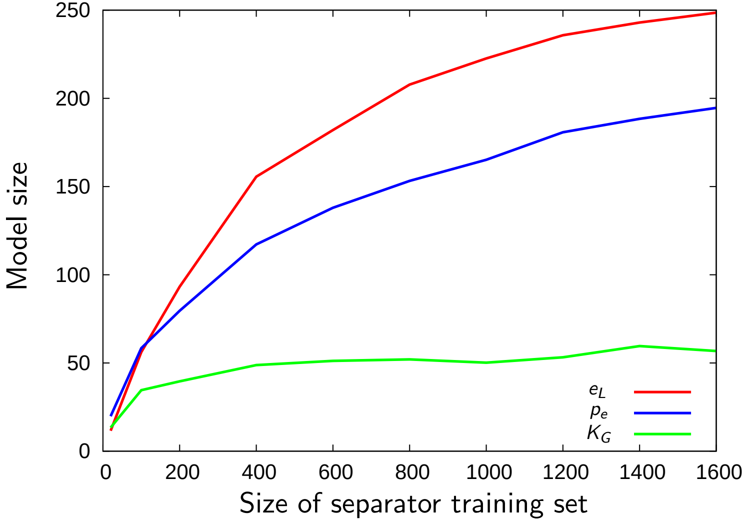
$\hookrightarrow$ **Independence from the size of the alphabet**

Convergence rate: accuracy

Task: **classify words** as either French or English (top words lists from Wiktionary).



Classification performance: sparsity



Conclusions

Recap

- We made use of the framework of Balcan et al. to create a **novel, efficient way to learn string similarities**.
- The resulting similarities provably **generalize well** to new examples and **induce low-error classifiers** for the task at hand.

Future work

- Adapt our method to **tree** edit cost learning (straightforward).
- Learn **other types of similarities** (e.g. numerical distances such as Mahalanobis distance).