Sentiment analysis with genetically evolved Gaussian kernels

I. Roman[1], A. Mendiburu[1], R. Santana[1], and J. A. Lozano[1,2]

[1]University of the Basque Country UPV/EHU
[2]Basque Center for Applied Mathematics BCAM

Abstract

Sentiment analysis consists of evaluating opinions or statements based on text analysis. Among the methods used to estimate the degree to which a text expresses a certain sentiment are those based on Gaussian Processes. However, traditional Gaussian Processes methods use a predefined kernels with hyperparameters that can be tuned but whose structure can not be adapted. In this paper, we propose the application of Genetic Programming for the evolution of Gaussian Process kernels that are more precise for sentiment analysis. We use use a very flexible representation of kernels combined with a multi-objective approach that considers simultaneously two quality metrics and the computational time required to evaluate those kernels. Our results show that the algorithm can outperform Gaussian Processes with traditional kernels for some of the sentiment analysis tasks considered.

1 Introduction

Among Natural Language Processing (NLP) problems, Sentiment Analysis (SA) has attracted much attention from the literature [12, 45, 1]. SA consists of understanding in an automated manner the opinion from written or spoken language. A common choice to solve these tasks is to compute a vector-based representation of the words, or embeddings, for all words in a sentence. Embeddings are then used by supervised learning algorithms to predict the presence or absence of an emotion in the text.

Kernel-based methods, due to their flexibility to capture different notions of similarity, have shown promising results inferring the latent sentiments from the embeddings [3, 2, 4]. Gaussian Process (GaussProc) models [41] are one example of kernel-based methods which have been successfully applied to emotion or sentiment classification [1]. GaussProcs rely on strong Bayesian inference foundations that allow them to update the model when new evidence is revealed. In comparison to other regression methods, they provide not only a prediction

of a given function, but also an estimate uncertainty of the predictions. Apart from SA, GaussProcs also have been used in other NLP tasks, such as, text classification [37], modeling periodic distributions of words over time [39], and quality estimation [9, 44].

In all previous applications of GaussProc to NLP, the choice of the most appropriate kernel was made a-priori, among some well-known kernel functions. While there is a repertoire of kernels available in the literature [41, 15, 22], the selection of the most suitable one for a given problem is not straightforward. For example, for SA, the most commonly applied kernel is the Squared Exponential (SE), also referred as the Radial Basis Function (RBF) kernel. However, Matern kernels have shown better results in this particular task [1].

Moreover, kernels usually have some parameters that need to be adjusted, which hardens the kernel selection problem. These parameters, often called hyperparameters, are usually tuned by maximizing a given metric (e.g., the marginal likelihood) [7].

Choosing the best metric to measure the quality of the kernels is not clear neither. While in the SA literature the Pearson's Correlation Coefficient (PCC) [35] has been used, this metric does not provide insights about the uncertainty that the GaussProc is able to model. As the GaussProcs offers a probabilistic prediction about the emotions, metrics such as the Log Marginal Likelihood (LML) [41], or the Negative Log Predictive Densities (NLPD) [40], seem better suited to choose between GaussProc models.

In this paper we propose a method that does not rely on an a-priori specified kernel. This means that finding the kernel expression is part of the *model selection* process. The learning algorithm that we use, which is based on Genetic Programming (GP) [26], is able to learn the kernel expression, together with an assignment of the corresponding hyperparameters. The evolved kernels keep the convenient property of being able to estimate the uncertainty of the predictions.

Furthermore, by using a multi-objective approach the GP algorithm evaluates kernels using three different criteria. PCC and NLPD measure kernel accuracy, and the third criterion minimizes the computational time needed by learning the kernel, therefore indirectly penalizing complexity.

The remainder of the paper is organized as follows: The next section presents the addressed problem in the context of SA. Section 3 introduces the main concepts related to GaussProc regression. The multi-objective GP approach to evolve kernel functions is presented in Section 4. In Section 5, a review on related work is provided. We describe the experimental framework used to validate our algorithm, along with the numerical results in Section 6. The conclusions of the paper and discussion of future work are presented in Section 7.

2 Sentiment Analysis

Sentiment Analysis (SA) is an automated process that infers the opinion or feeling from a piece of text. It can be considered as particular type of semantic annotation of the text. SA is a very complex problem due to several factors

including the ambiguity of human language, the large variability in the use of terms across individual, and the complexity of grammatical rules. However, the emergence of large text corpora and the usefulness of mining these corpora, e.g., for opinion mining related to products, services or politics [34], have contributed to develop more advanced machine learning algorithms for this task.

One of the directions of extending SA methods is to go beyond text categorization in positive or negative classes, to a more fine-grained emotion annotation [47]. This could be done by extending the number of classes in which a text is classified, but also by allowing a continuous value of the strength in which the sentiment is manifested in the text. A subject is requested to evaluate, in a range $[0, 100]$ a given sentiment (e.g., joy, fear, etc.) in a text. In the sentence "Alonso would be happy to retire with three titles", for example, joy and sadness feelings are mixed. The problem we address in this paper is to automatically estimate this continuous value from the analysis of the text. This problem is posed as a supervised regression problem in which a number of annotated examples are available with the sentiment value.

2.1 Sentence embeddings

Machine learning methods require a representation of text in order to do semantic analysis. There are a number of NLP approaches to represent and extract relevant information from text. In this domain, feature engineering to obtain informative features can be very labor-intensive. One increasingly used representation are word-vectors, or embeddings.

In a word embedding representation [31], each word is assigned a vector of continuous values, which are commonly learned using neural network models. Some of the first embedding methods, and still among the most used ones, are the Continuous Bags of words (CBOW) and the Skip-gram models. The details on how these algorithms work is beyond the scope of this paper.

Here, we focus on sentence embeddings, a common text representation for NLP tasks. For each sentence, we compute the embedding representation of all the words, while the words missing in the dictionary are assigned a zero-vector representation. The average of the embedding representations of all the words is computed to generate the sentence embedding. We will estimate the sentiment value from this representation.

3 Gaussian Process Regression

A Gaussian Process (GaussProc) is a stochastic process, defined by a collection of random variables, any finite number of which have a joint Gaussian distribution [41]. A GaussProc can be interpreted as a distribution over functions, and each sample of a GaussProc as a function.

GaussProcs can be completely defined by a mean function $m(\mathbf{x})$ and a covariance function, which depends on a kernel $k(\mathbf{x}, \mathbf{x}')$. Given that, a GaussProc

can be expressed as follows:

$$f(\mathbf{x}) \sim GaussProc(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$ (1)

where we assume that $\mathbf{x} \in \mathbb{R}^d$. We also consider an a-priori equal-to-zero mean function $(m(\mathbf{x}) = 0)$, to focus on the kernel search problem.

A GaussProc can be used for regression by getting its posterior distribution given some (training) data. Thus, the GaussProc can provide a probabilistic model to infer the sentiment of an sentence embedding. The joint distribution between the training outputs $\mathbf{f} = (f_1, f_2, ..., f_n)$ (where $f_i \in \mathbb{R}$, $i \in \{1, ..., n\}$ and $n \in \mathbb{N}$) and the test outputs $\mathbf{f}_* = (f_{n+1}, f_{n+2}, ..., f_{n+n_*})$ is given by:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right)$$ (2)

where $N(\mu, \Sigma)$ is a multivariate Gaussian distribution. Moreover, $X = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ $(\mathbf{x}_i \in \mathbb{R}^d$, $i \in \{1, ..., n\}$ and $n \in \mathbb{N})$ corresponds to the training inputs and $X_* = (\mathbf{x}_{n+1}, ..., \mathbf{x}_{n+n_*})$ to the test inputs. $K(X, X_*)$ denotes the $n \times n_*$ matrix of the covariances evaluated for all the (X, X_*) pairs.

The predictive Gaussian distribution can be found by obtaining the conditional distribution given the training data and the test inputs:

$$\begin{aligned} \mathbf{f}_* | X_*, X, \mathbf{f} &\sim \mathcal{N}(\hat{M}(X_*), \hat{K}(X_*, X_*)) \\ \hat{M}(X_*) &= K(X_*, X)K(X, X)^{-1}\mathbf{f} \\ \hat{K}(X_*, X_*) &= K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*) \end{aligned}$$ (3)

3.1 Covariance functions

GaussProc models use a kernel to define the covariance between any two function values [15]:

$$cov\left(f(\mathbf{x}), f(\mathbf{x}')\right) = k(\mathbf{x}, \mathbf{x}')$$ (4)

The kernel functions used in GaussProcs are positive semi-definite (PSD) kernels. According to the Mercer's Theorem [30], any PSD kernel can be represented as an inner product in some Hilbert Space.

The best known kernels in GaussProc literature are translation invariant, often referred to as stationary kernels. Among them, we focus on isotropic kernels where the covariance function depends on the norm:

$$k(\mathbf{x}, \mathbf{x}') = \hat{k}(r) \text{ where } r = \frac{1}{\theta_l} \|\mathbf{x} - \mathbf{x}'\|$$ (5)

where θ_l is the lengthscale hyperparameter and $\hat{k}$ a function that guarantees that the kernel is PSD.

On the other hand, non-stationary kernels are the ones that may vary with translation. Within this family, the most common kernels are those that depend

on the dot product of the input vectors. These kernels are usually referred to as dot-product kernels:

$$k(\mathbf{x}, \mathbf{x}') = \hat{k}(s) \text{ where } s = \frac{1}{\theta_l} (\mathbf{x} - \theta_s \mathbf{1}) (\mathbf{x}' - \theta_s \mathbf{1})^T \qquad (6)$$

where θ_l is again the lengthscale hyperparameter, θ_s is the shift hyperparameter and $\mathbf{1}$ is a vector of ones.

Some of these kernels have already been used also in NLP applications [2, 1, 46]. The Squared Exponential (SE) kernel ($\hat{k}_{SE}(r) = \theta_0^2 \, exp\left(-\frac{1}{2}r^2\right)$) is known to capture the smoothness property of the objective functions. This kernel, was compared to Matern class kernels (Matern 32 $\hat{k}_{M32}(r) = \theta_0^2 \left(1 + \sqrt{3}r\right) exp\left(-\sqrt{3}r\right)$ and Matern 52 $\hat{k}_{M52}(r) = \theta_0^2 \left(1 + \sqrt{5}r + \frac{5}{3}r^2\right) exp\left(-\sqrt{5}r\right)$) and the linear kernel ($\hat{k}_{LIN}(s) = s$) in [1] for SA tasks. There is little knowledge of the behaviour of other kernels when using GaussProcs in SA.

3.2 Model selection

The choice of the kernel and its hyperparameters has a critical influence on the behavior of the model, and it is crucial to achieve good results in NLP applications of GaussProcs [1]. This selection has been usually made by choosing one kernel a-priori, and then adjusting the hyperparameters of the kernel function so to optimize a given metric for the data. The most common approach is to find the hyperparameter set that maximizes the log marginal likelihood (LML):

$$log\, p\left(\mathbf{f}|X, \boldsymbol{\theta}, K\right) = -\frac{1}{2}\mathbf{f}^T K(X, X)^{-1}\mathbf{f} - \frac{1}{2}log\,|K(X, X)| - \frac{n}{2}\,log\,2\pi \qquad (7)$$

where $\boldsymbol{\theta}$ is the set of hyperparameters of the kernel and n is the length of X.

4 Evolving kernel functions

In this work we automatically search for new kernel functions in order to better predict sentiment expressed in a text. To guide this search, we propose to use a multi-objective GP approach.

We encode the kernel functions by means of mathematical expression trees. Each expressions are defined in a strongly-typed grammar [33] that specifies the possible combinations in which these kernels can be composed. We rely on the basic operations present in the well-known kernels shown in Section 3.1 (multiplication, square root, ...) to define the grammar.

These expression trees are evolved according to the GP approach shown in Algorithm 1. First, an initial population of N kernels is generated. In order to do so, each individual is created at random. Each generation, the offspring is evaluated. Next, the relative improvement of each objective O is measured.

If the relative improvement in the current population is greater than a threshold β for any of the objectives, a new population is generated through selec-

tion and variation. To generate an offspring population of N new individuals, the algorithm randomly chooses the variation method between a mutation or a crossover operator (with probability p_m and p_{cx} respectively, where $p_{cx} = 1 - p_m$), after selecting the μ best individuals. The next population is made up of the selected individuals and the offspring population.

When the relative improvement is lower or equal to the threshold β, the current population is replaced by a randomly generated one.

This procedure is repeated for G generations.

Algorithm 1 MOECov algorithm

1: **procedure** MOECov(N, G, O, μ, p_m, p_{cx}, β)
2:    $offspring = \text{GenRandPop}(N)$
3:    $bestfit_{-2} = (\infty)_{j=0}^{O}$
4:    $bestfit_{-1} = (\infty)_{j=0}^{O}$
5:    $pop = offspring$
6:    $all = offspring$
7:    $i = 0$
8:    **while** $i < G - 1$ **do**
9:       $\text{Evaluate}(offspring)$
10:      $best = \text{Select}(pop, 1)$
11:      $bestfit_i = \text{GetFitness}(best)$
12:      $relimprov = \left(\frac{\frac{1}{2}(bestfit_{i-2,j} + bestfit_{i-1,j}) - bestfit_{i,j}}{|bestfit_{i,j}|} \right)_{j=0}^{O}$
13:      **if** $\beta < MAX(relimprov)$ **then**
14:         $sel = \text{Select}(pop, \mu)$
15:         $offspring = \text{Variate}(sel, N, p_m, p_{cx})$
16:      **else** $\triangleright$ Restart procedure
17:         $sel = \emptyset$
18:         $offspring = \text{GenRandPop}(N)$
19:         $bestfit_i = (\infty)_{j=0}^{O}$
20:      **end if**
21:      $pop = sel \cup offspring$
22:      $all = all \cup offspring$
23:      $i = i + 1$
24:    **end while**
25:    $\text{Evaluate}(offspring)$
26:    $best = \text{SelectBest}(all)$
27:    **return** $best$
28: **end procedure**

4.1 Random kernel creation

To randomly generate kernel expression trees that conform the initial population, we propose a grow method based on the work done in [26]. The GEN-

RANDPOP function in Algorithm 1, generates random trees by a recursive process where, at each step, a random terminal or operator is added in a type-safe manner.

4.2 Variation operators

Our kernel search method is based on perturbation or variation methods that modify previous solutions to obtain new ones. We use two variation operators, which are randomly selected every VARIATE function call in Algorithm 1: A crossover operator, which combines two kernel functions to generate a new one that keeps some of the features of its parents, and a mutation operator, which introduces slight modifications to the original kernel to obtain a new individual.

We propose a crossover operator that randomly selects a subtree from each kernel and combines them with the sum or the product operator. This method merges the properties of each kernel while avoiding an excessive growth of the trees, known as bloating [26].

The mutation operator works by randomly selecting one of the following methods in a type-safe manner:

- **Insert**: Inserts an elementary mathematical expression at a random position in the tree, as long as their output types agree. The subtree at the chosen position is used as the input of the created expression. If more inputs are required by the new primitive, new terminals are chosen at random.

- **Shrink**: This operator shrinks the expression tree by randomly choosing a branch and replacing it with one of the branch inputs (also randomly chosen) of the same type. This helps to control the depth of the generated trees.

- **Uniform**: Randomly selects a point in the expression tree and replaces a subtree at that point by the subtree generated using our random generation method (Described in detail in Section 4.1).

- **Node Replacement**: Replaces a randomly chosen operator from the kernel expression by an operator with the same number of inputs and types, also randomly chosen.

4.3 Fitness evaluation

To estimate the quality of each kernel several metrics can be used. In SA tasks, Pearson correlation coefficient (PCC) is one of the most used metrics. It divides the covariance of two variables by the product of their standard deviations.

$$PCC(X, \mathbf{f}) = \frac{\sum_{i=1}^{n}(f_i - \hat{f})(\mu_i - \hat{\mu})}{\sqrt{\sum_{i=1}^{n}(f_i - \hat{f})^2 \sum_{i=1}^{n}(\mu_i - \hat{\mu})^2}} \tag{8}$$

7

where $\hat{f}$ is the mean of the vector $\mathbf{f}$ and $\hat{\mu}$ is the mean of the posterior mean μ.

However, PCC does not take into account the probabilistic information given by the GaussProc.

On the other hand, the Negative Log Predictive Density (NLPD) [40], sums the likelihood of each prediction, and is more informative about the GaussProc's performance.

$$NLPD(X, \mathbf{f}) = \frac{1}{n} \sum_{i=1}^{n} -\frac{(f_i - \mu_i)^2}{2\sigma_i^2} - \frac{1}{2} log \, \sigma_i^2 - \frac{1}{2} \, log \, 2\pi \qquad (9)$$

where μ_i and σ_i are the posterior mean and variance for $\mathbf{x}_i$.

Although both metrics are related, they evaluate the quality of the kernel from different perspectives. Ideally, it is desirable to guarantee a high correlation to the variable to be predicted along with a informative model of the uncertainty. Thus, the EVALUATE function in Algorithm 1 measures both, PCC and NLPD metrics, being the objectives of our search.

In addition, we also take the evaluation time as an third objective. We have observed that the complexity of the kernels, in term of number of primitives or depth, grows during the search, increasing the evaluation time. In order to obtain kernels that are efficient to evaluate we will measure the evaluation time, indirectly discouraging the development of large expressions.

4.4 Multi-objective kernel selection

We use a multi-objective selection operator based on the NSGA2 algorithm [10] to achieve good results in both metrics. As it can be seen in Algorithm 1, the SELECT function chooses μ individuals each generation. The population is divided into non-dominant groups, iteratively selecting the non-dominant group and repeating the operation with the rest of the individuals. The individuals in each group are sorted by crowding distance, and finally, the best μ individuals are chosen. To select the final best individual in the SELECTBEST function, the LML metric is used, in order balance the results of the objectives.

4.5 Hyperparameter Optimization

In contrast to other GP applications, the solutions in our approach do not encode all the necessary information to be evaluated. The optimal values of the hyperparameters, according to the LML, have to be determined. Thus, the performance of the solutions depends on the results of the hyperparameter optimization.

In this paper, the hyperparameters are optimized by means of *Powell*'s local search algorithm [38]. As this algorithm is not bounded, the search space has to be constrained by penalizing non-feasible hyperparameter sets. On the other hand, as the function to optimize might be multi-modal, a multi-start approach was used, performing a random restart every time the stopping criteria of the *Powell*'s algorithm are met, and getting the best overall result. During this

8

hyperparameter search, a maximum number of 150 evaluations of the LML were allowed.

Once the hyperparameters were optimized, in order to obtain the fitness of each kernel, PCC and the NLPD were evaluated by dividing the training set into 3 cross-validation folds. The results in each fold were averaged to obtain the actual fitness of each kernel. Finally, the time spent measuring both metrics in each fold was summed as a third objective.

Note that, as a result of the inclusion of the randomized restarts, the hyperparameters found for a certain kernel in two independent evaluations may not be the same. In fact, this implies that the fitness function optimized by the GP algorithm is stochastic.

5 Related work

Word embeddings [32] are extensively applied to NLP tasks [28]. The usual approach when combining embeddings from words in a sentence is to compute the average. This is the procedure used in [1], where 100-dimensional Glove embeddings [36] are the representation of choice for mapping texts to emotion scores using GaussProc regression.

GaussProcs are particularly suited to model uncertainty in the predictions and allow to accurately predict under noisy conditions. As such, there are diverse scenarios in which GaussProc can be applied to NLP tasks [8]. In all these problems the kernel function was selected beforehand, although its choice varies depending on the problem. The most frequently used kernel is the RBF kernel. However, this kernel is not suitable for all the problems. In [39], text periodicities of tweets are modeled using GaussProc with kernels specifically suited to capture the periodicities of the tweets. In the same paper, the periodic kernel (PER) and the periodic spike (PS) are shown to outperform non-periodic kernels and capture different periodic patterns. In [1], where GaussProcs are applied to SA, four different kernels are compared: SE, Linear, two Matern kernels. The Matern kernels are reported to produce better results than SE. In addition to numerical kernels, structural-kernels (e.g., tree-kernels) have been also combined with GaussProc.

Research on the evolution of kernel functions using evolutionary algorithms have shown that it is also possible to explore the space of kernel functions beyond the hyperparameter optimization. In the GaussProc literature this has been done by combining known kernels [27, 16, 29]. Kernels have also been evolved for Support Vector Machines (SVMs) [23, 20, 13, 48, 14, 25] and Relevance Vector Machines (RVMs) [5]. Some of the SVM approaches are also based in combining the well-known kernels [48, 14], although in some other works the kernels are learned from simple mathematical expressions [23, 20, 13, 25].

The particular characteristics determined by the GaussProcs make the evolution of kernels in this domain different to those algorithms used for SVMs and RVMs. Our method considers different ways to evaluate the kernels based on a multi-objective approach. It also brings to the GaussProc field the cre-

ation of kernels from scratch, without seeding components of human-designed kernels. This characteristic allows us to derive kernels that are not constrained by the prior knowledge while at the same time being optimized for the desired objectives.

6 Experiments

The goal of our experiments is to evaluate the performance of the proposed algorithm for the task of sentiment prediction from text. First we introduce the problem benchmark and word embeddings to evaluate the algorithm. Then, we describe the parameters used by the algorithms and explain the characteristics of the experimental framework, including the metrics used to compare the algorithms. Finally, we present the numerical results obtained from the experiments and discuss these results.

6.1 Problem benchmark and word-embeddings

We use the SemEval2007 Affective Text shared task dataset [47][1], following the work done in [1]. In this dataset, news headlines were manually annotated by experts, assigning to each text a degree of presence for each six Eckman [17] emotions: anger, disgust, fear, joy, sadness and surprise. In the original work, where this dataset was introduced, texts were divided into "dev" and "test" datasets. For the experiments presented in [1], the two sets were combined and further divided into 10 folds used for cross-validation. We use this 10-fold partition to evaluate our algorithms.

In order to compute a representation for each text, punctuations in each headline were removed, tokenized [6], and case ignored. From the resulted text, the word-vector representation of each word were obtained using the 100-dimensional GloVe embeddings [36][2]. After deleting the words that were not found in the embedding, the representation of each headline was computed as the average of the words.

6.2 Experimental setup

Our experiments consists of learning a kernel for a GP regressor that predicts a particular emotion based on sentence embeddings. We evaluate the quality of the final kernels in terms of the PCC and NLPD metrics.

We compare MOECov algorithm with different variants of a-priori defined kernel methods, which were shown in [1] to produce good prediction results for the six emotions previously described. The kernels are presented in Section 3.1.

The parameters used by the evolutionary algorithm were chosen after some preliminary experimentation:

- Population size: $N = 38$.

---

[1]Available at `https://web.eecs.umich.edu/\~mihalcea/downloads.html\#affective`
[2]Available from `https://nlp.stanford.edu/projects/glove/`

- Number of generations: $G = 65$.

- Number of objectives: $O = 3$.

- Mutation ($p_m = 0.4$) and crossover ($p_{cx} = 0.6$) probabilities.

- Selection size: $\mu = 9$.

- Restart threshold: $\beta = 1e-5$.

All algorithms were coded in Python. The implementation of MOECov is based on the EA software DEAP[3] [18].

Due to the stochastic nature of the algorithm, for all algorithms and in every dataset, the kernel search process was repeated 30 times along 10 random cross-validation folds.

6.3 Results of the comparison between the algorithms

Table 1 shows the average PCC metric of the best solution obtained by the algorithms in the 30 experiments. We remark that, these values have been computed on the test data, which we have not used for learning the GP programs. The results, for the NLPD metrics are shown in Table 2. In the tables, the best average value obtained for each of the sentiments are highlighted.

The analysis of the tables reveals that, in terms of the average fitness, MOE-Cov improves all other kernels for both metrics in most of the cases. For the PCC metric, only in the disgust dataset, MOECov was not able to outperform the M52 kernel. For the NLPD metric, MOECov outperforms all the algorithms for all the sentiments.

Among the well-known kernels, M52 seems a better choice than the others, as it gets the second best result in average in the rest of the problems according to the PCC metric. M52 is also the second best choice in NLPD, only surpassed by SE kernel in the fear dataset, while the results of M32 and SE kernels are similar. As it can be appreciated in the tables, the LIN kernel is the worst performing kernel according to the average results.

	LIN	M32	M52	SE	MOECov
anger	0.58592	0.63556	0.6401	0.62629	**0.64690**
disgust	0.44828	0.52492	**0.52782**	0.50111	0.52456
fear	0.68056	0.728096	0.73059	0.72737	**0.73555**
joy	0.53832	0.55775	0.57459	0.56341	**0.59158**
sadness	0.63625	0.67148	0.68205	0.67876	**0.69710**
surprise	0.40311	0.45416	0.45647	0.43758	**0.46751**

Table 1: Mean results for PCC metric. Best results are shown in bold. More is better.

---

[3]https://deap.readthedocs.io

	LIN	M32	M52	SE	MOECov
anger	3.94141	3.92037	3.91162	3.93041	**3.91084**
disgust	3.81476	3.78148	3.77491	3.80068	**3.77419**
fear	4.16636	4.10615	4.1006	4.0986	**4.07623**
joy	4.34633	4.32588	4.30362	4.32737	**4.29549**
sadness	4.31082	4.28845	4.27618	4.28176	**4.24454**
surprise	4.06292	4.04524	4.04519	4.0511	**4.02712**

Table 2: Mean results for NLPD metric. Best results are shown in bold. Less is better.

We conducted a statistical test to assess the existence of significant differences among the algorithms. For each metric and emotion, we applied the Friedman's test [19] and we found significant differences in every comparison (p-values can be seen in Figure 1 and Figure 2). Then, for each configuration, we applied a post-hoc test based on Friedman's test as in [11], and adjusted its results with the Shaffer's correction [43].

The results are shown in Figure 1 and Figure 2. The results confirm a coherent pattern where MOECov is the best performing algorithm for most of the datasets. However, it is also appreciated that according to this test, for most of the datasets, the differences between MOECov and M52 are not significant.

In evaluating these results it is important to take into account that the kernels produced by MOECov have been generated completely from scratch, with no prior knowledge of the existing kernels. The algorithm is able evolve a well performing kernel starting from elementary mathematical components.

On the other hand, the computational overhead introduced by MOECov should be considered. Most of the effort is spent evaluating kernels. The rest of the calculations required by the algorithm are negligible. Therefore, in this experiment, MOECov has required approximately 2500 times more computational time than fixed kernels. On the other hand, in evaluation time, once the kernel is learned, the effort required by the kernels generated by MOECov is similar to well known kernels.

6.4   Analysis of the MOECov evolution

One characteristic feature of our approach is that we simultaneously optimize different characteristics of the kernels. In order to determine whether the multi-objective approach effectively leads to the creation of more efficient kernels, both in terms of the accuracy for the prediction task, and in terms of efficiency, we analyze the fitness distribution of the solutions in the first and last population of MOECov for the `anger` dataset. These results are shown in Figure 3 where, in order to ease the visualization, results are shown for only one execution of the algorithm.

In Figure 3 we represent the scatter plots for each possible pair of objectives. As it can be appreciated in the figures, from the first to the last population there

(a) Anger   (b) Disgust
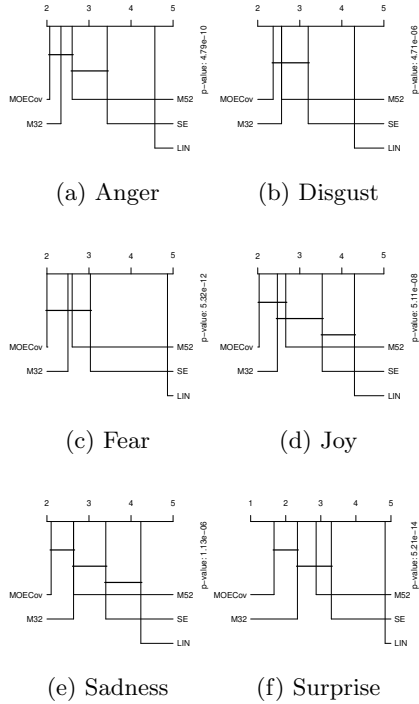
(c) Fear   (d) Joy

(e) Sadness   (f) Surprise

Figure 1: Critical difference diagrams for the PCC metric. The kernels are ordered following the results in their ranking. The metrics with no significant differences between them are matched with a straight line.

is an improvement in the values of the objective values for the PCC and NLPD metrics. However, the computational time actually increases from the first to the last population. This result is not surprising since it is expected that the increase in accuracy of the trees is achieved by also augmenting their complexity. In this scenario we expect that using the time as a third objective can serve to counteract useless complexity gain of the programs, but the average time for learning the kernels will necessarily increase.

6.5 Transferability of the evolved kernels

An important question to analyze is whether the kernels evolved by MOECov are only valid for the sentiment datasets in which they have been learned or they can also be used to predict sentiment in the other datasets. This question can be frame on the general research that investigate the transferability of solutions found by evolutionary algorithms [24, 21, 42]. In order to answer this question, we have used the best programs learned for the anger dataset to
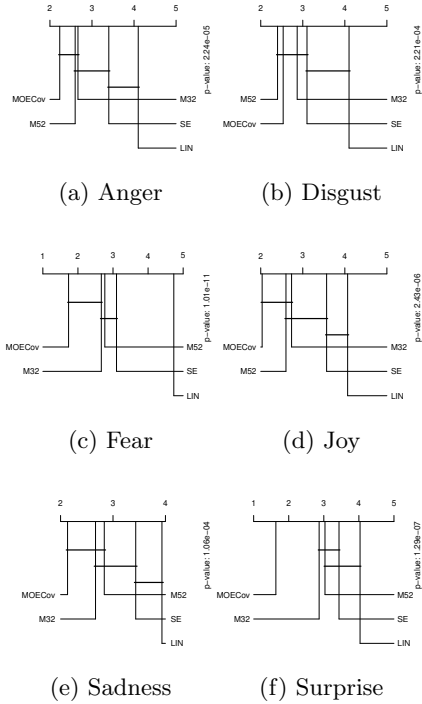
(a) Anger (b) Disgust

(c) Fear (d) Joy

(e) Sadness (f) Surprise

Figure 2: Critical difference diagrams for the NLPD metric. The kernel are ordered following the results in their ranking. The metrics with no significant differences between them are matched with a straight line.

make predictions in the other datasets. This can be considered as a transfer learning scenario in which the **anger** dataset is the source domain and all the other datasets serve as target domains. Notice, that in this particular example we do not recompute the hyperparameter values for the kernels. We simply apply the kernels *as they are* to the target datasets.

Figure 4 and Figure 5 respectively show the distributions of the objectives values obtained of this experiment for the PCC and NLPD metrics. In the figures, MOECov_anger indicates the kernels learned using the **anger** dataset. Notice that all the other algorithms have been learned using (training) data for each target dataset. The analysis of the figures indicate that the transferability of the kernels depends on the type of metric used. Results for PCC are at least as good as those obtained with the other kernels. However, for the NLPD metric results are slightly worse.

The main conclusion from this experiments is that the kernels evolved for predicting some sentiment can be also useful to predict other sentiments. This means that the embeddings contain the relevant information for the prediction
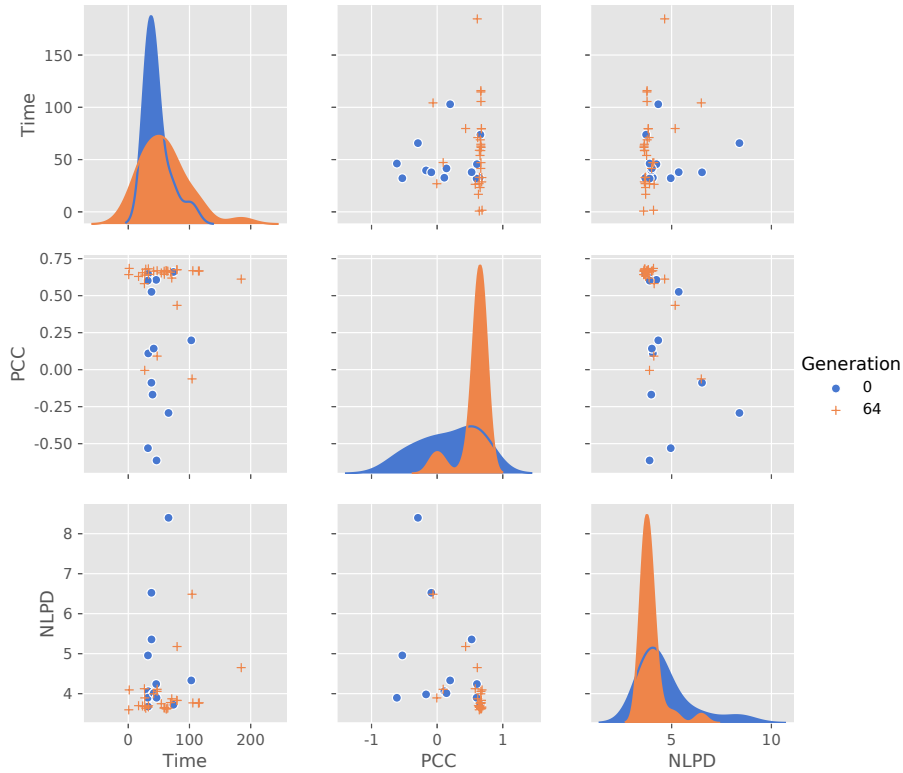
Figure 3: Distribution of the objective values in the first and last population of MOECov for one execution of the algorithm in the `anger` dataset.

and that the type of transformations that make a kernel a good predictor are similar across sentiment domains.

7   Conclusions

Sentiment analysis is a relevant problem in NLP. Particularly, when considering predicting the sentiment with a finer level of detail, beyond binary classification. Although, in previous work it has been shown that GaussProc regression is an efficient approach to solve this regression problem, our hypothesis was that the fixed structure used by classical kernels lacks the flexibility to capture more subtle differences in datasets. Therefore, in this paper we have proposed MOECov as way to evolve the structure of the kernels. By addressing the creation of kernels as a multi-objective problem we have been able to generate kernels that simultaneously optimize two of the accuracy metrics proposed, along with the
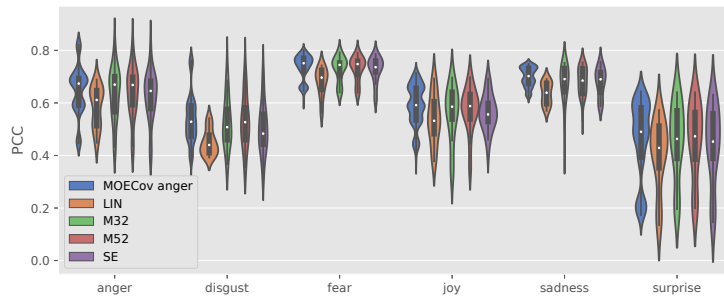
Figure 4: Results of the transferability experiments for the PCC metric. Each coloured shape shows a kernel density estimation of the distribution. A boxplot of the results is shown inside. More is better.
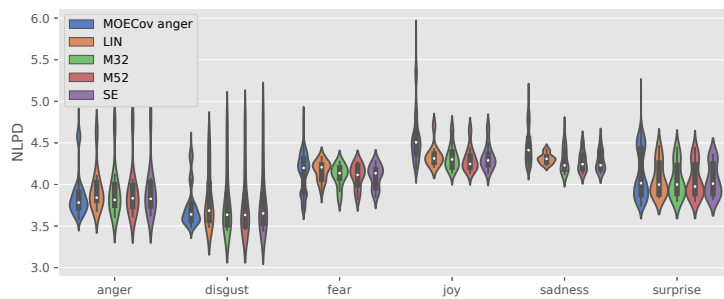

Figure 5: Results of the transferability experiments for the NLPD metric. Similar to the previous figure, a boxplot is shown inside each coloured shape that shows the estimation of the distribution. Less is better.

computational complexity.

As far as the authors are concerned this is the first work that uses evolved GaussProcs for multi-objective problems. We had not found either previous studies that tries to optimize the hyper-parameters of fixed kernels simultaneously considering two or more metrics.

There are a number of ways in which our studied could be extended. Other sentiment datasets, possibly in other languages, could be considered. Furthermore, other semantic analysis tasks, such as, the post-editing effort prediction could be addressed using our approach. We also noticed that at the time of stopping the MOECov algorithm the quality of the solutions was still improving. Therefore, more fitness evaluations are likely to produce better results of the algorithm.

ACKNOWLEDGEMENTS

References

[1] BECK, D. Modelling Representation Noise in Emotion Analysis using Gaussian Processes. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (Taipei, Taiwan, Nov. 2017), Asian Federation of Natural Language Processing, pp. 140–145.

[2] BECK, D., AND COHN, T. Learning Kernels over Strings using Gaussian Processes. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (Taipei, Taiwan, Nov. 2017), Asian Federation of Natural Language Processing, pp. 67–73.

[3] BECK, D., COHN, T., HARDMEIER, C., AND SPECIA, L. Learning Structural Kernels for Natural Language Processing. *Transactions of the Association for Computational Linguistics 3* (Dec. 2015), 461–473.

[4] BECK, D. E. *Gaussian Processes for Text Regression.* phd, University of Sheffield, June 2017.

[5] BING, W., WEN-QIONG, Z., LING, C., AND JIA-HONG, L. A GP-based kernel construction and optimization method for RVM. In *2010 The 2nd*

International Conference on Computer and Automation Engineering (IC-CAE) (Feb. 2010), vol. 4, pp. 419–423.

[6] Bird, S., Klein, E., and Loper, E. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[7] Blum, M., and Riedmiller, M. Optimization of Gaussian Process Hyperparameters using Rprop. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (2013).

[8] Cohn, T., Preotiuc-Pietro, D., and Lawrence, N. Gaussian processes for natural language processing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Tutorials* (2014), pp. 1–3.

[9] Cohn, T., and Specia, L. Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2013), vol. 1, pp. 32–42.

[10] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. In *Parallel Problem Solving from Nature PPSN VI* (2000), M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds., Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 849–858.

[11] Demšar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res. 7* (2006), 1–30.

[12] Deriu, J., Lucchi, A., De Luca, V., Severyn, A., Müller, S., Cieliebak, M., Hofmann, T., and Jaggi, M. Leveraging Large Amounts of Weakly Supervised Data for Multi-Language Sentiment Classification. In *Proceedings of the 26th International Conference on World Wide Web* (Republic and Canton of Geneva, Switzerland, 2017), WWW '17, International World Wide Web Conferences Steering Committee, pp. 1045–1052. event-place: Perth, Australia.

[13] Diosan, L., Rogozan, A., and Pecuchet, J. P. Evolving kernel functions for SVMs by genetic programming. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)* (2007), pp. 19–24.

[14] Dioşan, L., Rogozan, A., and Pecuchet, J.-P. Improving classification performance of Support Vector Machine by genetically optimising kernel shape and hyper-parameters. *Applied Intelligence 36*, 2 (Mar. 2012), 280–294.

[15] Duvenaud, D. *Automatic model construction with Gaussian processes.* Thesis, University of Cambridge, 2014.

[16] DUVENAUD, D., LLOYD, J., GROSSE, R., TENENBAUM, J., AND ZOUBIN, G. Structure Discovery in Nonparametric Regression through Compositional Kernel Search. In *Proceedings of The 30th International Conference on Machine Learning* (2013), pp. 1166–1174.

[17] EKMAN, P. Facial expression and emotion. *American psychologist 48*, 4 (1993), 384.

[18] FORTIN, F.-A., RAINVILLE, F.-M. D., GARDNER, M.-A., PARIZEAU, M., AND GAGNÉ, C. DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research 13*, Jul (2012), 2171–2175.

[19] FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association 32*, 200 (1937), 675–701.

[20] GAGNÉ, C., SCHOENAUER, M., SEBAG, M., AND TOMASSINI, M. Genetic Programming for Kernel-Based Learning with Co-evolving Subsets Selection. In *Parallel Problem Solving from Nature - PPSN IX*, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2006, pp. 1008–1017.

[21] GARCIARENA, U., SANTANA, R., AND MENDIBURU, A. Evolved GANs for Generating Pareto Set Approximations. In *Proceedings of the Genetic and Evolutionary Computation Conference* (New York, NY, USA, 2018), GECCO '18, ACM, pp. 434–441. event-place: Kyoto, Japan.

[22] GENTON, M. G. Classes of Kernels for Machine Learning: A Statistics Perspective. *J. Mach. Learn. Res. 2* (Mar. 2002), 299–312.

[23] HOWLEY, T., AND MADDEN, M. G. An Evolutionary Approach to Automatic Kernel Construction. In *Artificial Neural Networks – ICANN 2006* (Sept. 2006), Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 417–426.

[24] IQBAL, M., ZHANG, M., AND XUE, B. Improving classification on images by extracting and transferring knowledge in genetic programming. In *2016 IEEE Congress on Evolutionary Computation (CEC)* (July 2016), pp. 3582–3589.

[25] KOCH, P., BISCHL, B., FLASCH, O., BARTZ-BEIELSTEIN, T., WEIHS, C., AND KONEN, W. Tuning and evolution of support vector kernels. *Evolutionary Intelligence 5*, 3 (Sept. 2012), 153–170.

[26] KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[27] KRONBERGER, G., AND KOMMENDA, M. Evolution of Covariance Functions for Gaussian Process Regression Using Genetic Programming. In *Computer Aided Systems Theory - EUROCAST 2013* (Feb. 2013), Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 308–315.

[28] LAMPOS, V., ZOU, B., AND COX, I. J. Enhancing Feature Selection Using Word Embeddings: The Case of Flu Surveillance. In *Proceedings of the 26th International Conference on World Wide Web* (Republic and Canton of Geneva, Switzerland, 2017), WWW '17, International World Wide Web Conferences Steering Committee, pp. 695–704. event-place: Perth, Australia.

[29] LLOYD, J. R., DUVENAUD, D., GROSSE, R., TENENBAUM, J., AND GHAHRAMANI, Z. Automatic Construction and Natural-Language Description of Nonparametric Regression Models. In *Twenty-Eighth AAAI Conference on Artificial Intelligence* (June 2014).

[30] MERCER, J., AND A, B. XVI. Functions of positive and negative type, and their connection the theory of integral equations. *Phil. Trans. R. Soc. Lond. A 209*, 441-458 (Jan. 1909), 415–446.

[31] MIKOLOV, T., CHEN, K., CORRADO, G., AND DEAN, J. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]* (Jan. 2013). arXiv: 1301.3781.

[32] MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. S., AND DEAN, J. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[33] MONTANA, D. J. Strongly Typed Genetic Programming. *Evolutionary Computation 3*, 2 (June 1995), 199–230.

[34] PANG, B., AND LEE, L. Opinion Mining and Sentiment Analysis. *Foundations and Trends® in Information Retrieval 2*, 1–2 (July 2008), 1–135.

[35] PEARSON, K. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London 58* (1895), 240–242.

[36] PENNINGTON, J., SOCHER, R., AND MANNING, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543.

[37] POLAJNAR, T., ROGERS, S., AND GIROLAMI, M. Protein interaction detection in sentences via Gaussian Processes: a preliminary evaluation. *International journal of data mining and bioinformatics 5*, 1 (2011), 52–72.

[38] POWELL, M. J. D. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal 7*, 2 (Jan. 1964), 155–162.

[39] PREOŢIUC-PIETRO, D., AND COHN, T. A temporal model of text periodicities using Gaussian Processes. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (2013), pp. 977–988.

[40] QUIÑONERO-CANDELA, J., RASMUSSEN, C. E., SINZ, F., BOUSQUET, O., AND SCHÖLKOPF, B. Evaluating Predictive Uncertainty Challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment* (2006), J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. d'Alché Buc, Eds., Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 1–27.

[41] RASMUSSEN, C. E., AND WILLIAMS, C. K. *Gaussian processes for machine learning.* MIT Press, 2006.

[42] SANTANA, R., MENDIBURU, A., AND LOZANO, J. A. Structural transfer using EDAs: An application to multi-marker tagging SNP selection. In *2012 IEEE Congress on Evolutionary Computation* (June 2012), pp. 1–8.

[43] SHAFFER, J. P. Modified Sequentially Rejective Multiple Test Procedures. *Journal of the American Statistical Association* (Mar. 2012).

[44] SHAH, K., COHN, T., AND SPECIA, L. An investigation on the effectiveness of features for translation quality estimation. In *Proceedings of the Machine Translation Summit* (2013), vol. 14, Citeseer, pp. 167–174.

[45] SOCHER, R., PENNINGTON, J., HUANG, E. H., NG, A. Y., AND MANNING, C. D. Semi-supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Stroudsburg, PA, USA, 2011), EMNLP '11, Association for Computational Linguistics, pp. 151–161. event-place: Edinburgh, United Kingdom.

[46] SPECIA, L. Exploiting objective annotations for measuring translation post-editing effort. In *Proceedings of the 15th Conference of the European Association for Machine Translation* (2011), pp. 73–80.

[47] STRAPPARAVA, C., AND MIHALCEA, R. SemEval-2007 Task 14: Affective Text. In *Proceedings of the 4th International Workshop on Semantic Evaluations* (Stroudsburg, PA, USA, 2007), SemEval '07, Association for Computational Linguistics, pp. 70–74.

[48] SULLIVAN, K. M., AND LUKE, S. Evolving Kernels for Support Vector Machine Classification. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2007), GECCO '07, ACM, pp. 1702–1707.