MATHEMATICAL MODELING AND COMPUTING, Vol. 10, No. 2, pp. 511–517 (2023)

Towards a polynomial approximation of support vector machine accuracy applied to Arabic tweet sentiment analysis

Banou Z., Elfilali S., Benlahmar H.

Faculty of Sciences Ben M'Sik – Hassan II University,
Bd Commandant Driss Al Harti, 7955, Casablanca, Morocco

(Received 4 March 2023; Accepted 4 May 2023)

Machine learning algorithms have become very frequently used in natural language processing, notably sentiment analysis, which helps determine the general feeling carried within a text. Among these algorithms, Support Vector Machines have proven powerful classifiers especially in such a task, when their performance is assessed through accuracy score and f1-score. However, they remain slow in terms of training, thus making exhaustive grid-search experimentations very time-consuming. In this paper, we present an observed pattern in SVM's accuracy, and f1-score approximated with a Lagrange polynomial.

Keywords: *Lagrange polynomial; SVM; machine learning; sentiment analysis; hyper-parameter optimization.*

2010 MSC: 68T50, 68T01 **DOI:** 10.23939/mmc2023.02.511

1. Introduction

SVM models are parametric algorithms used for machine classification and regression. Their functioning is based on a matrix kernel that may be linear, polynomial, gaussian or sigmoid based, which makes this type of algorithms enormously powerful estimators regardless of data linearity. Nevertheless, the size of kernel increases proportionally with the number of samples in the training dataset. This means that larger datasets lead to larger kernel matrices, and consequently taking much longer to fit. This issue becomes more imminent in an experimental environment where the goal is to explore various combinations of hyper-parameters to improve performance metrics. Moreover, in the case of text classification tasks in general, frequency-based vectorization algorithms such as TF-IDF tend to generate multiple columns, each representing a word, making the calculation of kernel values more time and memory consuming. Henceforth, the need of a predictable performance pattern is considered a promising solution, as it would help obtain the parameters to an optimal performance score without experimental tests.

2. Background

Several hyper-parameter optimization algorithms can be classified as model-free, gradient-based, and Bayesian optimization. This section will discuss the advantages and disadvantages of some well-known hyper-parameter optimization approaches.

2.1. Manual approach

As the name suggests, the manual approach consists of manually changing parameters until a satisfying result is obtained. This method requires not only a tedious human intervention but also human experience in terms of hyper-parameters that are objects of optimization [1]. A more automated approach is often preferred when different algorithms are considered in an experimental study.

2.2. Random search

One of the most used hyper-parameter optimization algorithms in machine learning is random search [2], which power relies on choosing random combinations of hyper-parameters with no reg-

ular pattern or distribution. Despite the gain in training time, the probability of such algorithms landing on an optimum is equal to skipping it, as parameters randomly fluctuate within their respective domains of definition. Ref. [3] has proven that a random search is an ineffective approach when applied to Deep Belief Networks (DBNs). This implies the necessity of a more rigorous approach.

2.3. Grid search

In contrast to random search algorithms, grid search tends to exhaustively test every possible combination of hyper-parameters to find the best values for performance metrics. Ref. [4] used grid-search hyper-parameter optimization (GSHPO) on an EDHS dataset for HIV/AIDS prediction, resulting in an accuracy of 87.6 on the Gradient Boosting algorithm. While [5] have reached an accuracy score of 0.95 by using GSHPO on a sentiment analysis dataset of hotel reviews collected from Booking.com. However, this approach remains time and resource-consuming, especially in a low-space configuration.

2.4. Cuckoo search

Cuckoo search (CSO) belongs to the family of nature-inspired algorithms of optimization, as it simulates cuckoos' nesting process which consists of finding the best host nest to lay their eggs. Analogically, the CSO attempts to find the combination of hyper-parameters which gives the highest performance. Given a set of combinations, one is arbitrarily chosen to be compared to a newly generated combination using the formula indicated by the equation (1):

$$x_{i+1} = x_i + m \cdot a, \tag{1}$$

where x_i are solutions in the hyper-parameter space, a is the scaling factor, and m is the local random walk, which can also be calculated via the Lévy flight formula (2) as the authors in [6] proceeded in their study.

$$m = L(\beta, \gamma, \delta) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \cdot \frac{e^{-\frac{\gamma}{2(\beta-\delta)}}}{(\beta-\delta)^{\frac{3}{2}}}, & 0 < \beta < \delta < \infty, \\ 0 & \text{otherwise}, \end{cases} \tag{2}$$

where β is the step length, γ is a scaling parameter for Lévy's formula, and δ is the minimum step length. Figure 1 explains the workflow of the optimization process.

2.5. Particle swarm

Another algorithm inspired from nature is Particle Swarm Optimization (PSO); one of the most widely used metaheuristics which was introduced by [7]. Its concept relies on the ensemble retrieval of the global optimum, by generating a set of particles denoted P such that P is a subset of the hyper-parameter space. Each particle p_i is defined by its vectors $X^i(t)$ and $V^i(t)$, which represent respectively a hyper-parameter combination within the defined space and the velocity of the particle p_i at the iteration t. The vector coordinate of each particle changes according to its velocity, as shown in the equation 3:

$$X^i(t+1) = X^i(t) + V^i(t+1). \tag{3}$$

Particle velocity is computed at each iteration via the formula (4):

$$V^i(t+1) = w \cdot V^i(t) + c \cdot r_1\big(\text{best}(p_i) - X^i(t)\big) + s \cdot r_2\big(B_t - X^i(t)\big), \tag{4}$$

where w is the constant weight inertia, r_1 and r_2 are random values between 0 and 1, c is the cognitive coefficient, s is the social coefficient, $\text{best}(p_i)$ is the value of $X^i(t)$ that yielded the best result so far and B_t is the global optimum at the iteration t. The coefficients w, c and s are the parameters of the PSO algorithm, they help control the balance between exploration and exploitation. The power of the PSO resides in the use of several particles scattered around the hyper-parameter space, which increases the probability of finding a better optimum in less iterations, however, this could still be cost-ineffective regarding execution time when we consider that the objective function to optimize is an SVM classifier. This caveat can be overcome by executing the optimization in parallel for each particle, nevertheless, it would require a large computational power for a large set of particles.

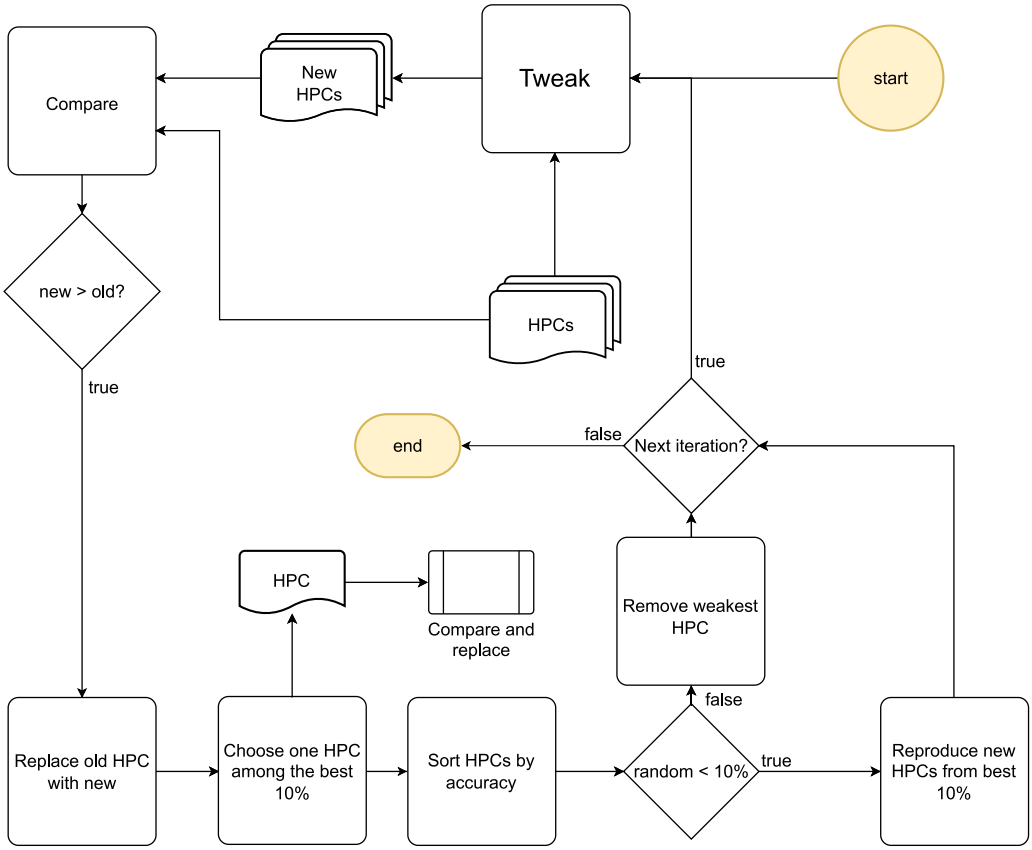

Fig. 1. Cuckoo search optimization flowchart.

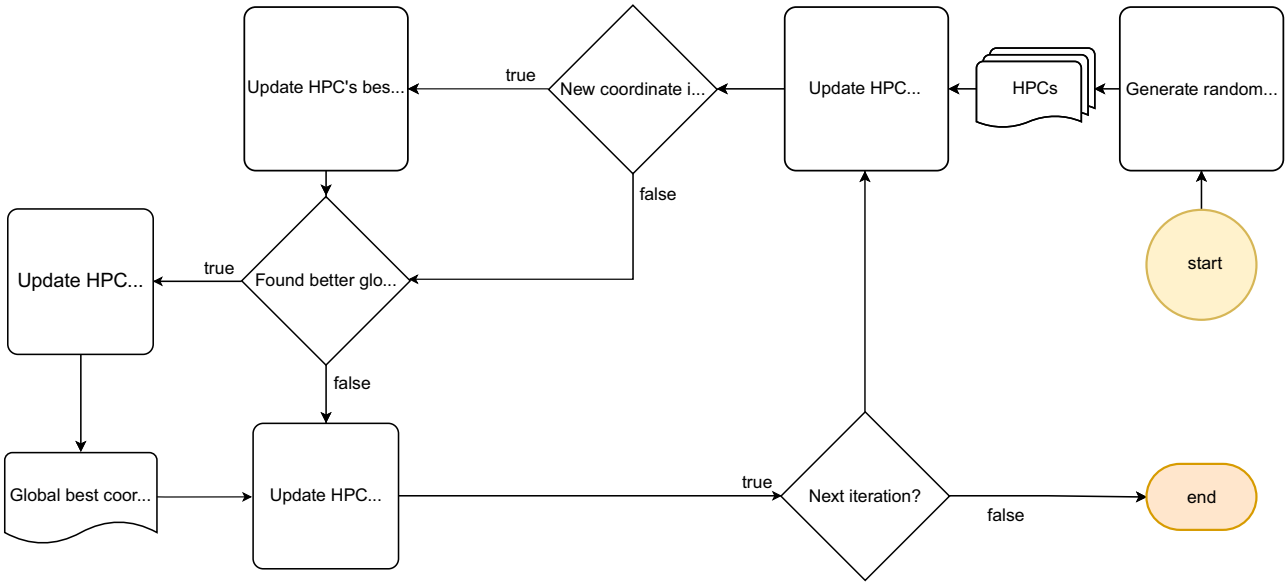

Fig. 2. Particle swarm optimization flowchart.

2.6. Polar bear optimization

Similarly, Polar Bear Optimization (PBO) [8] is an algorithm imitating the natural life cycle of polar bears. It simulates the birth of a new bear by combining features from the best models in the optimization space, and death is simulated by the generation of a new model with a random combination of hyper-parameters from the optimization space.

The algorithm resembles CSO in its iterative nature, however, the performance is ensured to be improved since at each iteration the weakest model is replaced by a potentially more robust classifier, which leaves at the end of the optimization process a set of more powerful estimators.

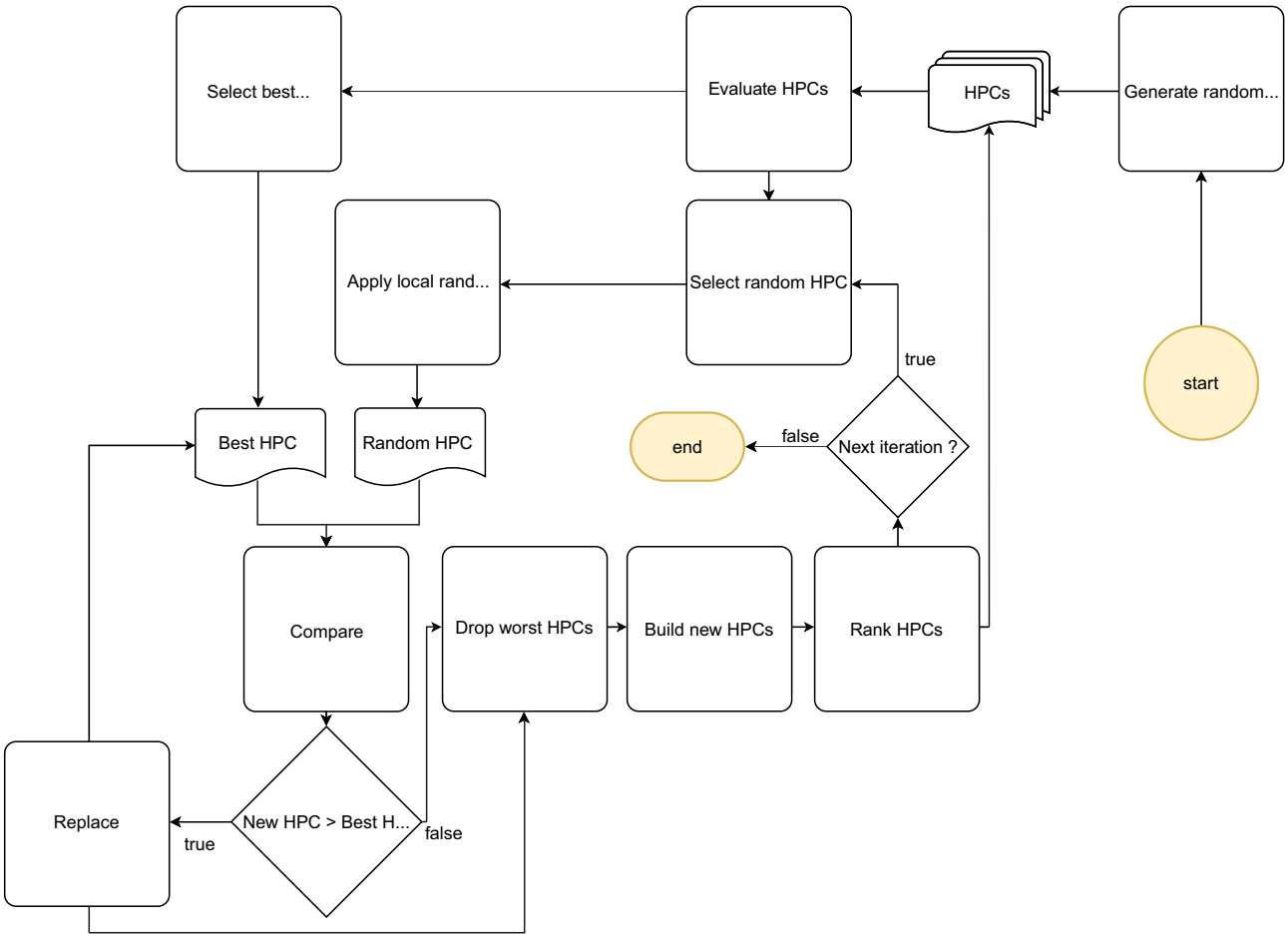

Fig. 3. Polar bear optimization flowchart.

Table 1. Comparison between hyper-parameter optimization approaches.

Algorithm	**Criteria**		
	Efficiency	***Speed***	***Human intervention***
Manual optimization	Depends on human experience	Slow	Yes
Random search	Random	Fast	No
Grid search	Depends on defined domain	Slow	No
Cuckoo search	Efficient	Medium	No
Particle swarm	Efficient	Medium	Initial parameters
Polar bear optimization	Efficient (Eliminates weaker models)	Medium	No

Table 1 sums up the comparison between the different approaches we have discussed in this section.

3. Experimental study

Our experiment consists in training SVM models on four different datasets: Arabic Sentiment Twitter Dataset (ASTD) [9], Moroccan Sentiment Twitter Dataset (MSTD) [10], Arabic Speech Act and Sentiment (ArSAS) [11], and Multi-Domain Arabic Resources for Sentiment Analysis (MARSA) [12]. Since each dataset was designed for a different purpose, we have only conserved Positive and Negative tweets in each dataset to assess SVM' s performance on 2-way sentiment classification applied to various datasets with several features. We have varied the regularization hyper-parameter C during our

experimental phase while fixing the kernel type as Radial Basis Kernel, in order to obtain an optimal performance based on this parameter.

To analyze the performance of SVMs, we plotted accuracy scores against the parameter C such that it varies between 0 and 1, during the rest of the study, we will denote by Y_M and $\hat{Y}_M$ respectively the real values and the interpolated values.

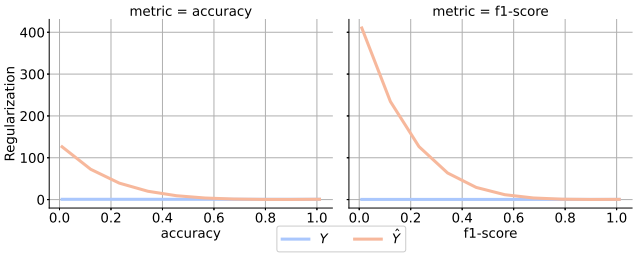

Fig. 4. Comparison between true performance scores and interpolated scores on ASTD dataset.

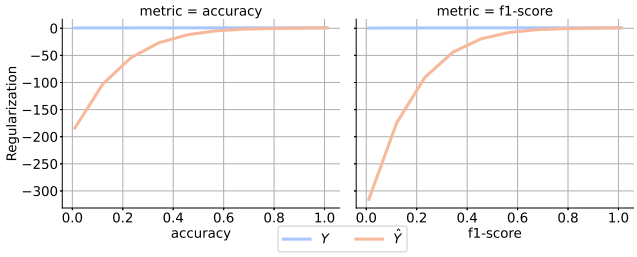

Fig. 5. Comparison between true performance scores and interpolated scores on ArSAS dataset.

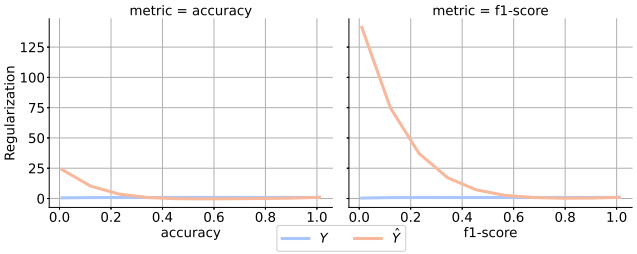

Fig. 6. Comparison between true performance scores and interpolated scores on MARSA dataset.

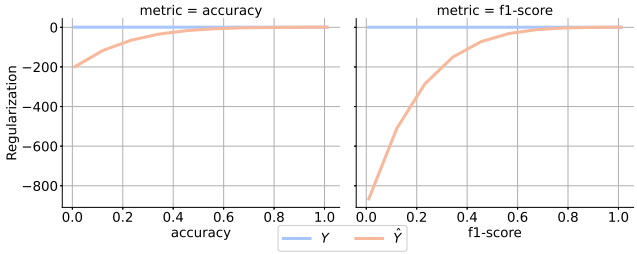

Fig. 7. Comparison between true performance scores and interpolated scores on MSTD dataset.

Upon looking through these observations, we calculated the mean absolute error and mean squared error between real and interpolated scores.

Table 2. Error scores between real evaluation metrics and their interpolated values.

Dataset	***Mean absolute error***		***Mean squared error***	
	Accuracy	***F1-score***	***Accuracy***	***F1-score***
ASTD	26.81	87.73	2285.05	24357.73
ArSAS	39.35	66.11	4874.12	14119.45
MARSA	**4.03**	**27.72**	**64.59**	**2699.11**
MSTD	44.83	193.49	5955.81	112251.75

Experimentally, we observed that one dataset has significantly lower error values than others. Therefore, we investigated four datasets we used so we can find potential correlations between error values and any dataset related parameter, notably balance parameters in terms of speech act and sentiment polarity. In this context, we calculated and normalized the number of tweets per class for each dataset, as shown in Table 3.

Table 3. Percentage of tweet sentiment class and distribution coefficients per dataset.

Dataset	**Positive tweets (%)**	**Negative tweets (%)**	Δ_C	Δ_S
ASTD	32.12	67.88	0.2529	0.1068
ArSAS	37.07	62.93	0.1829	0.1059
MARSA	45.37	54.63	0.0655	0.1025
MSTD	23.84	76.16	0.3700	0.1228

By $\Delta_C(D)$ and $\Delta_S(D)$, we respectively denote the standard deviation of tweet percentage per sentiment class and speech act within the dataset D. By reading the results in Tables 2 and 3, we have eye-witnessed a correlation between Δ_C and Δ_S on one hand, and error scores on the other hand. Table 4 shows correlation values, which proves empirically that the accuracy of the interpolated F1-score is linearly correlated with the balance of the dataset, either by classes or speech acts. Accuracy scores do still correlate despite their lower coefficients of correlation. These results can be interpreted as it is easier to anticipate the performance of the model on a balanced dataset than on an imbalanced dataset.

Table 4. Correlation coefficients between interpolation errors and dataset balance parameters for each evaluation metric.

Dataset balance parameter	Mean absolute error		Mean squared error	
	Accuracy	F1-score	Accuracy	F1-score
Δ_C	0.7582	0.9921	0.7521	0.9589
Δ_S	0.7184	0.9872	0.7655	0.9980

4. Conclusion and Future avenues

This study of hyper-parameter optimization applied to support vector regularization will potentially reduce time and effort deployed to obtain a powerful SVM sentiment classifier, therefore, studies can be further expanded to cover more hyper-parameters of different classification algorithms. Moreover, this study has only been conducted on 2-way classification tasks, however, many benchmark datasets, including those we used in our experimental setup, are designed for multi-way classification tasks. Therefore, a potential research axis is to explore the validity of our theory on the datasets in their full extent, as well as to explore the correlations with resulting coefficients of the Lagrange polynomial to find new rules to estimate the accuracy score in advance.

[1] Yang L., Shami A. On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing. **415**, 295–316 (2020).

[2] Bergstra J., Bengio Y. Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research. **13**, 281–305 (2012).

[3] Bergstra J., Bardenet R., Bengio Y., Kégl B. Algorithms for Hyper-Parameter Optimization. Advances In Neural Information Processing Systems. **24** (2011).

[4] Belete D. M., Huchaiah M. D. Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results. International Journal of Computers and Applications. **44** (9), 875–886 (2021).

[5] Elgeldawi E., Sayed A., Galal A. R., Zaki A. M. Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis. Informatics. **8** (4), 79 (2021).

[6] Woźniak M., Połap D., Napoli C., Tramontana E. Graphic object feature extraction system based on Cuckoo Search Algorithm. Expert Systems with Applications. **66**, 20–31 (2016).

[7] Kennedy J., Eberhart R. Particle swarm optimization. Proceedings Of ICNN'95 – International Conference On Neural Networks. **4**, 1942–1948 (1995).

[8] Połap D., Woźniak M. Polar Bear Optimization Algorithm: Meta-Heuristic with Fast Population Movement and Dynamic Birth and Death Mechanism. Symmetry. **9** (10), 203 (2017).

[9] Nabil M., Aly M., Atiya A. ASTD: Arabic Sentiment Tweets Dataset. Proceedings of The 2015 Conference on Empirical Methods in Natural Language Processing. 2515–2519 (2015).

[10] Mihi S., Ait B., El I., Arezki S., Laachfoubi N. MSTD: Moroccan Sentiment Twitter Dataset. International Journal of Advanced Computer Science and Applications. **11** (10), (2020).

[11] Elmadany A., Mubarak H., Magdy W. ArSAS: An Arabic Speech-Act and Sentiment Corpus of Tweets (2018).

[12] Alowisheq A., Al-Twairesh N., Altuwaijri M., Almoammar A., Alsuwailem A., Albuhairi T., Alahaideb W., Alhumoud S. MARSA: Multi-Domain Arabic Resources for Sentiment Analysis. IEEE Access. **9**, 142718–142728 (2021).

До поліноміальної апроксимації точності методу опорних векторів, застосованого до аналізу тональності твітів арабською мовою

Бану З., Ельфілалі С., Бенлахмар Х.

Факультет наук Бен М'Сік – Університет Хасана II, бульвар Коменданта Дріс Аль Харті, 7955, Касабланка, Марокко

Алгоритми машинного навчання стали дуже часто використовуватися в обробці природної мови, зокрема в аналізі тональності, який допомагає визначити загальне відчуття, яке міститься в тексті. Серед цих алгоритмів метод опорних векторів (SVM) є потужними класифікаторами, особливо в такому завданні, коли їхня продуктивність оцінюється через показник точності та показник f1. Однак вони залишаються повільними з точки зору навчання, що робить вичерпні експерименти з пошуку по сітці дуже трудомісткими. У цій статті представлено спостережувану закономірність точності SVM і показник f1, апроксимований поліномом Лагранжа.

Ключові слова: *поліном Лагранжа; метод опорних векторів; машинне навчання; аналіз тональності текстів; гіперпараметрична оптимізація.*