Optimizing the Gabriel graph construction algorithm

Jose G. Fernandes, Vitor M. Hanriot and Antonio P. Braga
Universidade Federal de Minas Gerais
Belo Horizonte, MG 31270-901
Contact Author: `josegeraldof@ufmg.br`

Abstract

Gabriel graph based models approach large margin classifiers by obtaining support vectors from geometric properties. However, they have been limited to small and medium-sized applications, mainly due to the cost of computing the graph. This work presents an algorithm to compute the graph optimizing computational and memory costs. For the former, we exploit a distance matrix computed in advance and for the latter we use a bootstrap approach to construct the graph from batches of the dataset, with upper bound convergence analysis. Our approach aims to enable applications with large datasets for these models.

1 Introduction

Large margin classifiers have brought attention in the machine learning community to the idea of maximizing the margin of separation between classes [24], rather than simply minimizing an empirical cost in the training set. This has been approached in various ways such as Boosting [17, 9], Gaussian mixture [18] and Support vector machines [3]. The latter achieves this by quadratic programming in the norm of the weights of the support vectors (SV) following the principle of structural risk minimization, or alternatively by solving a system of linear equations [19, 5].

Another way to consider SVs is from their geometric properties in a classification problem [2, 14, 25]. This relationship has already been used to find large margin classifiers in affine hulls [6, 15]. Chipclass [20, 1, 22] is an example of this approach using the Gabriel Graph (GG) [10], a planar graph that encodes the local spatial properties of the dataset from distance relations. The main advantage of this method, in particular, is its potential for autonomous learning, as it requires minimal to no adjustment of hyperparameters, making it well suited for applications such as the Internet of Things (IoT) [12, 16] and edge computing [4, 7].

However, since the computational cost of computing the GG is very high, these studies have mostly conducted experiments on small and medium-sized datasets. Therefore, this work proposes an algorithm for constructing GG optimized in time and memory, allowing the use of these models for large datasets. All code and experiments are available at our repository[1].

2 Gabriel graph

Given a dataset of N samples represented as $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, the Gabriel graph of $\mathbf{X}$ is represented by the $N \times N$ adjacency matrix $\mathbf{A} = \{a_{ij}\}$ obtained by the application of the following graph formation rule to all elements of the Euclidean distance matrix $\mathbf{D} = \{d_{ij}\}$. Element $a_{ij} = 1$ if $d_{ij}^2 \leq [d_{ik}^2 + d_{jk}^2] \ \ \forall \ \mathbf{x}_k \ \epsilon \ \mathbf{X}$, or, in other words, edge e_{ij} exists if and only if there is no other sample $\mathbf{x}_k$ inside the hypersphere that has samples $\mathbf{x}_i$ and $\mathbf{x}_j$ diametrically opposed, as represented schematically in Figure 1 for a set with 5 samples.

[1]https://github.com/cljosegfer/gg-neurips24

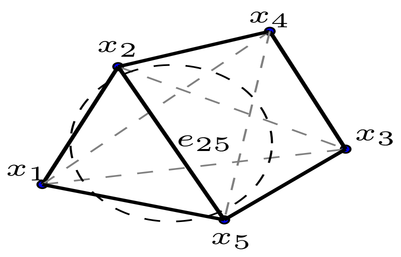


Figure 1: Schematic representation of a Gabriel graph construction. The solid lines represent the edges that are included in the graph. The dashed circle is represented between samples $\mathbf{x}_2$ and $\mathbf{x}_5$ in order to show that there is no other sample inside the circle, so edge e_{25} exists.

An important property of the Gabriel graph is that it is planar, edges do not intercept each other and, therefore, the structure of the graph preserves local spatial properties of the data, since only the smallest edges will pass the triangle inequality rule. Also, intrinsic properties of the graph, like centrality and dominance may unfold important properties of the data. Such graph representation has enabled the construction of different classifiers, which are solely based on information extracted from the graph [23, 21, 11].

3 Optimizing computational cost

In the construction of the Gabriel graph, the operation that compares d_{ij}^2 with the sum of all pairs of distances d_{ik}^2 and d_{jk}^2 in the dataset is computationally expensive and a bottleneck for graph implementation. During graph construction, all elements a_{ij} are potential candidates to become edges, so they are all tested for the triangle inequality until a sample violates the edge. The computational cost of graph construction can be as high as $O(N^3)$, limiting the use of GG-based models in large-scale applications.

However, since the comparisons are independent, they can be computed in parallel if we allocate the distance matrix $\mathbf{D}$ in a GPU. This will improve the computational cost at the expense of the memory. The bottleneck then becomes the computation of the distance matrix which should be be given in advance, and since we are using PyTorch's `cdist` implementation [13] the cost is $O(N^2)$. The method for computing the GG in parallel is shown in Algorithm 1.

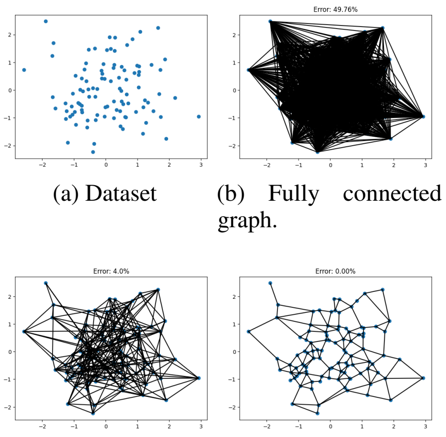
Algorithm 1 Pseudocode for the parallel implementation of Gabriel graph.

```
Inputs: $\mathbf{X}$ {dataset}
Outputs: $\mathbf{A}$ {adjacency matrix}
N ← length($\mathbf{X}$)
$\mathbf{D}$ ← cdist($\mathbf{X}, \mathbf{X}$)$^2$
$\mathbf{A}$ ← Boolean adjacency matrix for vertices, all false
for $i = 0$ to $N - 1$ do
    $\mathbf{S} \leftarrow \mathbf{F}[i, :] + \mathbf{F}[i+1, :]$          ▷ sum of all pairs of distances
    $\mathbf{m}$ ← argmin($\mathbf{S}$, axis = 1)          ▷ we only need to verify for the closest point
    $\mathbf{s} \leftarrow \mathbf{A}$[arange($N$), $m$]          ▷ distance of the closest point
    $\mathbf{A}[i, i+1:] \leftarrow s - \mathbf{F}[i, i+1:] > 0$          ▷ the edge exists if the closest doesn't violate the rule
$\mathbf{A} \leftarrow \mathbf{A} + \mathbf{A}^T$          ▷ we compute only the triangular half as the matrix is symmetric
```

4 Optimizing memory cost

Optimizing the computational cost was possible at the expense of memory. However, when the computations are performed on a GPU with limited RAM, the algorithm is infeasible. We address this problem by employing a bootstrap approach to incrementally construct the graph. It is possible to detect edge violations in the main graph by subsampling the dataset and verifying the triplets ($\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{x}_k$) in the corresponding sub-graphs. Once an edge is rejected in a sub-graph, it can also be safely rejected in the main graph, since distance relations are preserved. This approach not only alleviates the constraints on local memory constraints in GPU RAM during distance computations but also facilitates the detection of edge violations using smaller datasets. Initially, all edges are assumed to exist, and if a point $\mathbf{x}_k$ is found within the subset of $\mathbf{x}_i$ and $\mathbf{x}_j$, contained within the corresponding hypersphere, that specific edge is removed. Only a few points $\mathbf{x}_k$ violate the graph rule, if they indeed exist, so the graph can be incrementally constructed until the *fortunate* event occurs that some of them fall within the same subset as the samples $\mathbf{x}_i$ and $\mathbf{x}_j$ takes place.

As an example of the bootstrap construction, Figure 2a shows a dataset sampled from a bivariate normal distribution, which will be used to demonstrate the method. Initially all samples are connected, since it is assumed that all edges exist, as shown in Figure 2b. Each iteration, therefore, serves to possibly falsify the violating edges by monotonically converging, but slowing down, since with each iteration it becomes rarer to find a point that falsifies a remaining violating edge. Figure 2c shows the graph at iteration 5 and Figure 2d shows the final graph obtained after 19 iterations, which in this example is 100% equivalent to the exact graph. It may happen that the final graph is not exact, since its performance depends on the subset size and on the number of iterations. The algorithm is shown in Algorithm 2. Ideally, the batch size should be as large as possible when computing the partial distance matrix. The remaining challenge lies in establishing the number of epochs needed to achieve confidence in the graph approximation.



(a) Dataset (b) Fully connected graph.

(c) Graph at iteration 5. (d) Final graph

Figure 2: Example of graph formation as edges are falsified.

Algorithm 2 Pseudocode for bootstrap implementation of Gabriel graph.

```
Inputs: $\mathbf{X}$ {dataset}; b {relative batch size}; T {number of epochs}
Outputs: $\mathbf{A}$ {adjacency matrix}
$\mathbf{A} \leftarrow$ Boolean adjacency matrix for vertices, all true      ▷ now we initialize all edges to true
for $i = 0$ to $T$ do
    shuffle dataset to get random subsets
    while loop for batches do                                  ▷ [batch] means we are indexing the subset
        $\mathbf{A}_{batch} \leftarrow$ gg($\mathbf{X}[batch]$)                        ▷ GG of the subset ($b$ x $b$)
        $A[batch] \leftarrow A[batch] * A_{batch}$                          ▷ false is preserved
```

Upper bound of iterations. Assuming that the samples are selected according to a uniform distribution in order to generate the batches, the edge construction can be represented as a Bernoulli trial. In the worst case, only one point $\mathbf{x}_k$ contravene the edge ($\mathbf{x}_i$, $\mathbf{x}_j$). By characterizing the convergence of this scenario, it becomes possible to establish an upper bound on the probability that this edge is incorrect, which depends on the number of iterations. In the worst case scenario, the probability that point $\mathbf{x}_k$ is in the same subset as points $\mathbf{x}_i$ and $\mathbf{x}_j$, or the likelihood of correctly evaluating the edge ($\mathbf{x}_i$, $\mathbf{x}_j$) in this iteration, is simply the ratio b which is the relative batch size. Note that we make no prior assumption about the data distribution, we only calculate the probability for the worst case to provide an upper bound for any general distribution from a real dataset. Also note that when

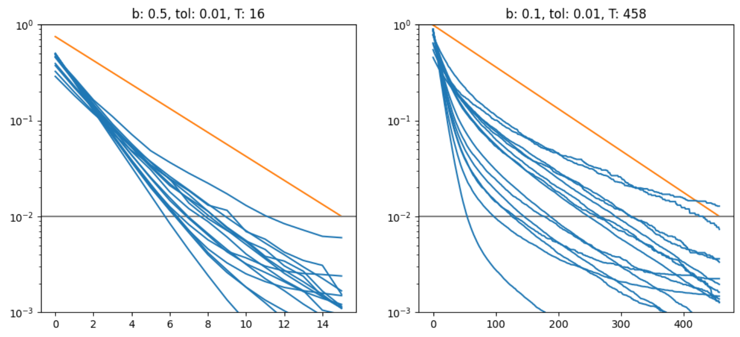


Figure 3: Convergence curves with relative batch sizes of 50% and 10%. The blue curves represent the fraction of incorrect edges relative to the number of iterations for each dataset. The orange curve represents the upper bound derived from Equation 1 and the horizontal line indicates the target tolerance of 1%.

partitioning the points into subsets, only this specific fraction of edges, b, is evaluated during each iteration. In worst case there is only one sample $\mathbf{x}_k$ that falsifies the edge, the probability that this specific sample is in the present subset is b^2. Consequently, the average proportion of incorrect edges over the course of multiple epochs t can be described as the expectation in Equation 1, where ϵ is the fraction of wrong edges at iteration t. We can also define a safe number of iterations T based on a tolerance δ for the expected fraction of incorrect edges in the graph approximation.

$$\mathbb{E}[\epsilon] = (1 - b^2)^t; \; T(\delta, b) = \frac{\log \delta}{\log 1 - b^2} \tag{1}$$

Although it is not the main motivation, we can also use the Equation 1 to evaluate the computational cost of the bootstrap approach compared to the parallel method. Since the cost of computing the graph in parallel is $O(N^2)$, and the bootstrap is just a call of this computation on smaller sets, the final cost can be computed as $T \times O(bN)^2$, where $b < 1$. However, the upper bound on convergence is very tight, in fact it is the worst case expectation, and the more we challenge this bound, the more competitive the bootstrap becomes. In practice, one would sacrifice time for memory cost by restrictively respecting this upper bound, but if the applicator has access to the specific convergence of the dataset itself, which depends on the density of the data points, the number of iterations can be significantly reduced. We tested this on 15 general-purpose datasets from the UCI repository [8] varying the relative batch sizes (50% and 10%) as shown in Figure 3. Note that the approximation exceeds the target tolerance in only one dataset, although it is very close, still less than 2%. This phenomenon highlights another important characteristic: the deviation from the probability of the upper bound. This characteristic is shown by the spread of the blue curves as the batch size varies, the smaller the batch, the larger the deviation, as expected from the binomial distribution variance. Therefore, it is not surprising that the exceptional experiment occurred with the smaller batch size.

5 Conclusion

The use of GG-based models has been limited in small and medium-sized datasets, mainly due to the cost of computing the graph. To enable this application in large datasets, we have introduced an algorithm to compute the graph while optimizing computational and memory costs. For time complexity, we use a distance matrix that is computed in advance and loaded into the GPU to perform all comparisons in parallel. However, with limited GPU RAM, when it's infeasible to load the entire distance matrix, we developed a bootstrap approach to incrementally construct the graph with subsets and also compute the upper bound probability of iterations to safely trust the graph's approximation.

For future work, we plan to apply GG-based models to large and unstructured datasets using the learned representation of self-supervised learning techniques, since their embedding has semantic information encoded in distance relations. We also plan to investigate smart indexing strategies for the subsets to further speed up the computation of GG, especially in the final epochs when it is rarer to find a point that falsifies a remaining violating edge. Promising ideas include sampling the batches from the complete set of all triplets (edges and possible violating points), evaluating these triplets in advance, and adding stopping criteria based on relationships between batches from different epochs and the resulting edge update.

References

[1] Janier Arias-Garcia, Alan Cândido de Souza, Liliane Gade, Jones Yudi, Frederico Coelho, Cristiano L Castro, Luiz CB Torres, and Antonio P Braga. Improved design for hardware implementation of graph-based large margin classifiers for embedded edge computing. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):1320–1329, 2022.

[2] Kristin P Bennett and Erin J Bredensteiner. Duality and geometry in svm classifiers. In *ICML*, volume 2000, pages 57–64. Citeseer, 2000.

[3] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[4] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. An overview on edge computing research. *IEEE access*, 8:85714–85728, 2020.

[5] BPR Carvalho and AP Braga. Ip-lssvm: A two-step sparse classifier. *Pattern Recognition Letters*, 30(16):1507–1515, 2009.

[6] Hakan Cevikalp, Bill Triggs, Hasan Serhan Yavuz, Yalçın Küçük, Mahide Küçük, and Atalay Barkana. Large margin classifiers based on affine hulls. *Neurocomputing*, 73(16-18):3160–3168, 2010.

[7] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[8] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[9] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[10] K Ruben Gabriel and Robert R Sokal. A new statistical approach to geographic variation analysis. *Systematic zoology*, 18(3):259–278, 1969.

[11] Vítor M Hanriot, Luiz CB Torres, and Antônio P Braga. Multiclass graph-based large margin classifiers: Unified approach for support vectors and neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[12] Shancang Li, Li Da Xu, and Shanshan Zhao. The internet of things: a survey. *Information systems frontiers*, 17:243–259, 2015.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[14] Xinjun Peng and Yifei Wang. Geometric algorithms to large margin classifier based on affine hulls. *IEEE Transactions on Neural Networks and Learning Systems*, 23(2):236–246, 2011.

[15] Xinjun Peng and Dong Xu. Geometric algorithms for parametric-margin ν-support vector machine. *Neurocomputing*, 99:197–205, 2013.

[16] Karen Rose, Scott Eldridge, and Lyman Chapin. The internet of things: An overview. *The internet society (ISOC)*, 80(15):1–53, 2015.

[17] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5:197–227, 1990.

[18] Fei Sha and Lawrence K Saul. Large margin gaussian mixture modeling for phonetic classification and recognition. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE, 2006.

[19] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9:293–300, 1999.

[20] LCB Torres, CL Castro, F Coelho, F Sill Torres, and AP Braga. Distance-based large margin classifier suitable for integrated circuit implementation. *Electronics Letters*, 51(24):1967–1969, 2015.

[21] Luiz CB Torres, Cristiano L Castro, Frederico Coelho, and Antônio P Braga. Large margin gaussian mixture classifier with a gabriel graph geometric representation of data set structure. *IEEE Transactions on Neural Networks and Learning Systems*, 32(3):1400–1406, 2020.

[22] Luiz CB Torres, Cristiano L Castro, Honovan P Rocha, Gustavo M Almeida, and Antonio P Braga. Multi-objective neural network model selection with a graph-based large margin approach. *Information Sciences*, 599:192–207, 2022.

[23] Luiz CB Torres, Cristiano Leite Castro, and Antônio P Braga. Gabriel graph for dataset structure and large margin classification: A bayesian approach. In *ESANN*, 2015.

[24] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[25] Wan Zhang and Irwin King. A study of the relationship between support vector machine and gabriel graph. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 1, pages 239–244. IEEE, 2002.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction mention the algorithm for computational cost (Section 3) and memory cost (Section 4).

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 4 discusses the limitations of the algorithm in Section 3 but also poses new challenges, namely the convergence and variance of the upper bound probability.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [No]

   Justification: The upper bound probability has only an informal proof, but it's simple enough and short-papers are not allowed to have any supplementary material.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Now in camera-ready stage, we provide the source code for all experiments. The pseudocodes fairly describe the algorithms that are the main contribution. Although we haven't explicitly specified the datasets used in the convergence experiment, we believe that any reasonable set of UCI datasets will reproduce the same results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Now in camera-ready stage, we provide the source code for all experiments. The pseudocodes fairly describe the algorithms that are the main contribution. Although we haven't explicitly specified the datasets used in the convergence experiment, we believe that any reasonable set of UCI datasets will reproduce the same results.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The hyperparameters for the convergence experiment are described. Since there was no training, there were no data splits and optimizers.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The variance in the convergence experiment was weakly addressed with a simple experiment. We considered it to be a minor problem since the upper bound limit is already tight.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: All experiments were cheap enough for us to run on the Google Colab T4 GPU, so we considered the computational resources not significant enough to be mention.

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research does not deal with human subjects or sensitive data, nor does it have any copyright issues.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We can't think in any direct societal impact.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We only use public data from UCI that has been referenced.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Only the pseudocode is described.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]