DISPERSION EFFECT ON GENERALISATION ERROR IN CLASSIFICATION
Experimental Proof and Practical Algorithm

Benoît Gandar [1,2], Gaëlle Loosli[1]
[1]*Clermont Université, Université Blaise Pascal, 63000 Clermont-Ferrand, France and CNRS, UMR 6158, LIMOS, 63 173 Aubière, France*

Guillaume Deffuant[2]
[2]*Cemagref de Clermont-Ferrand, Laboratoire LISC, 24 avenue des Landais, 63 172 Aubière Cedex 1, France*

Keywords: Machine Learning, Classification, Space Filling Design, Dispersion.

Abstract: Recent theoretical work proposes criteria of dispersion to generate learning points. The aim of this paper is to convince the reader, with experimental proofs, that dispersion is a good criterion in practice for generating learning points for classification problems. Problem of generating learning points consists then in generating points with the lowest dispersion. As a consequence, we present low dispersion algorithms existing in the literature, analyze them and propose a new algorithm.

1 INTRODUCTION

In the context of classification tasks, we address the question of optimal position of points in the feature space in the case one as to define a given number of training points anywhere in the space. Those training points will be named *sequence* from now on. To that purpose, we want to use the dispersion of the sequence, which is an estimator of the spread. The dispersion is defined in the next section as well as references to a paper stating that this criterion is likely to be more efficient for classification tasks than other well known criteria such as discrepancy. Our point in this position paper is to convince the reader that a) the dispersion is in practice a good criterion and b) we can provide an efficient algorithm to use it, even though evaluating this dispersion is costly.

2 DEFINITION OF DISPERSION

Dispersion is an estimator of the spread of a sequence used in numerical optimization. It is usually used in iterative algorithms in order to approximate the extremum of a non derivable function in a compact set. The error approximation can also be theoretically expressed by a function of dispersion (Niederreiter, 1992).

Dispersion of a Sequence: Let I^s be the unit cube in dimension s with the euclidian distance d. The dispersion of a sequence $x = \{x_1, \ldots, x_n\}$ is defined by:

$$\delta(x) = \sup_{y \in I^s} \min_{i=1,\ldots,n} d(y, x_i)$$

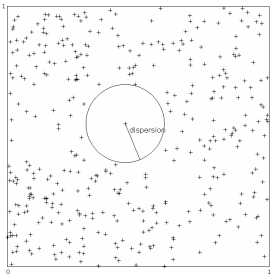

Figure 1: Estimation of dispersion. The dispersion is the radius of the largest ball containing no point.

The dispersion of a sequence is the radius of the largest empty ball of I^s (see Figure 1).

Bounds of Dispersion of a Sequence: (Niederreiter, 1992) shows that dispersion of a sequence $x = \{x_1, \ldots, x_n\}$ in I^s is lower bounded by:

$$\delta(x) \geq \frac{1}{2\lfloor \sqrt[s]{n} \rfloor}. \quad (1)$$

Moreover, he proves that for each dimension s, it ex-

Gandar B., Loosli G. and Deffuant G..
DISPERSION EFFECT ON GENERALISATION ERROR IN CLASSIFICATION - Experimental Proof and Practical Algorithm.
DOI: 10.5220/0003293007030706
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 703-706
ISBN: 978-989-8425-40-9

ists a sequence x in I^s such:

$$\lim_{n\to+\infty} \sqrt[s]{n}\delta(x) = \frac{1}{log(4)}.$$

As a consequence, for each dimension s, it exists a sequence x in I^s such $\delta(x) = O\left(\frac{1}{\sqrt[s]{n}}\right)$. Considering previous inequality, this order is the lowest.

3 EXPERIMENTAL EVIDENCE

In order to illustrate theoretical results and to observe the effects of decreasing dispersion of training points, we present experiments involving the following three steps:

1. Learning with k-NN from random samples of fixed size for different classification functions, and estimating the generalization error.
2. Then we decrease the sequence dispersion by running a few iterations of the algorithm described in section 4. We apply the classification algorithm using these new samples and estimate the generalization error. This step is repeated until the dispersion algorithm stops.
3. Finally, we represent the generalization error rate depending on the dispersion rate (decreasing) with boxes and whiskers. Note that the dispersion is observed on sequences and is not a parameter. Each box has lines at the lower quartile, median, and upper quartile values. Whiskers extend from each end of the box to the adjacent values in the data. The most extreme values are within 1.5 times the interquartile range from both end of the box. Adjacent values represent 86.6% of population for a gaussian distribution.

We have generated two types of classification rules in spaces of dimension 2 to 5:

- The first type of rules is relative to simple classification problems. The classification boundaries have small variations and are smooth.
- The second type of rules is relative to difficult classification problems. Classification boundaries have more important variations and are less smooth. Moreover classifications surfaces have more connected components.

Experimental Protocol: We have made experiments with 2000 learning points and 5000 test points on 500 learning problems, in space of dimension 2 to 5. The results are similar in all tested dimensions, and we report here for dimension 5 in Figures 2 and 3.

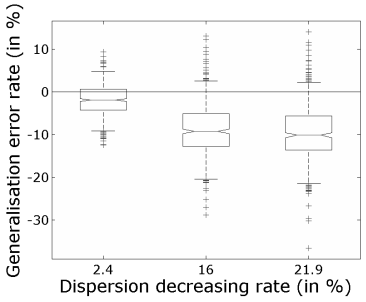

Figure 2: Relative decrease of generalization error varying with the relative decrease of dispersion for 500 simple learning problems in dimension 5 with 2000 learning points. (if δ_i is the initial dispersion and δ_t the dispersion after several iterations of the algorithm, the relative decrease is $\frac{\delta_i-\delta_t}{\delta_i}$). The boxes represent distribution of error rates.

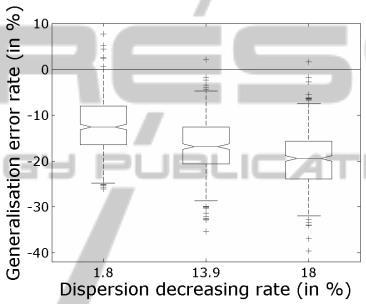

Figure 3: Relative decrease of generalization error varying with the relative decrease of dispersion for 500 hard learning problems in dimension 5 with 2000 learning points.

Conclusion and Discussion: We can see that lines of median value (middle lines of boxes) are below zero in the majority of problems. It shows that decreasing dispersion reduces error generalization in more than 50% of cases. We can also remark that the higher the dispersion decrease, the lower the generalization error rate. Upper liners of boxes (they represent 75% of population) are also below zero once dispersion minimizing algorithm has converged. Moreover it seems that more complex learning problems are, more k-NN algorithm is sensitive to dispersion of learning points.

4 LOW DISPERSION ALGORITHMS

In this part, we look for generating a sequence with the lowest dispersion as possible for a fixed size. Pioneering works of (Johnson et al., 1990) about generation of low dispersion sequences proposed two different criteria: a criterion to minimize called *minimax*

and a criterion to maximize called *maximin*. In a first part, we present in first criterion of *minimax* and algorithms based on it. In a second part, we present criterion of *maximin* and also algorithms based on it. In each part, we explain why different algorithms presented are not optimal.

4.1 Minimax

Minimax criterion based algorithms try to minimize the maximal distance between points of the sequence and candidate points.

Algorithms based on Swaping Points Process: (Johnson et al., 1990) have proposed an algorithm which makes the reduction of *minimax* criterion by a simple swap of points, starting from random. The convergence of the algorithm is guaranteed, but no guarantee is given to have generated the best possible sequence.

Algorithms based on Adding Points Process: (Lindemann and LaValle, 2004) have developed an incremental algorithm reducing dispersion of sequences by adding points. They reduce also dispersion and get around the initial global problem of minimization. However, this method reduces incrementally the dispersion by adding points and is not optimal when the user has a fixed maximal number of points.

4.2 Maximin

An alternative to minimizing the dispersion consists in maximizing the distance between the points of the sequence. It is easier to optimize numerically and corresponds to the *maximin* criterion defined by $\delta_2(x) = \inf_{(x_1,x_2)\in I^{s2}} d(x_1,x_2)$. We can once more distinguish two types of algorithms: **Algorithms based on Deleting Points Process:** (Sergent et al., 1997) have proposed to generate a sequence with a lot of points and to delete superfluous points until we obtain a sequence with a fixed minimal distance between points. The major disadvantage of this algorithm is the quasi-impossibility to predict the size of the final sequence. **Algorithms based on Adding Points Process:** It is also possible to generate a low dispersion sequence by adding points to an initial sequence according to *maximin* criterion where the dispersion is high . However the algorithms based on maximisation of $\delta_2(x) = \inf_{(x_1,x_2)\in x^2} d(x_1,x_2)$ push generally points towards the frontier of cube. This effect can be avoided by considering the criterion $\delta_3(x) = \inf_{(x_1,x_2)\in x^2} d(x_1,\{x_2\}\cup I^{s'})$ where $I^{s'} = \{x \in \mathcal{R}^s | x \notin I^s\}$. A set which maximizes $\delta_3(x)$ is also said a *maximin set*. Being inspired by works of (Lindemann and LaValle, 2004), (Teytaud et al., 2007) have developed an algorithm based on criterion $\delta_3(x)$ and random trees to explore the space. At first step, they generate a point in the middle of unit cube. Then, at each step, they add a point with maximization of criterion $\delta_3(x)$. As a consequence, adding this point is optimum toward the previous space configuration. However the obtained configuration is not necessarily the best one with this total number of points.

To Summarize. All theses methods are not optimal and we have to considers the total number of points to generate low dispersion sequence: it is the purpose of the next part.

5 A NEW LOW DISPERSION ALGORITHM

We propose an algorithm using the properties of dispersion established by (Niederreiter, 1992), and based on *maximin* criterion together with spring variables. Those spring variables are function of the distance between the points of the sequence and between the points and the borders of unit cube. This algorithm has good properties in practice: for an appropriate number of points, it converges to the Shukarrev grid which minimizes dispersion and it converges generally quickly. Its complexity is about $O\left(n^2k\right)$ for a sequence of size n and k iterations.

Sketch of the Algorithm: The basic idea of the algorithm is to achieve two tasks : spread the points as much as possible and remain inside the unit cube and not too close to the boundaries (similarly to Shukkarev grids). Hence we have defined two steps, each dealing with one of the tasks. The algorithm iterates over those steps until convergence. In practice, we also have to deal with some local minima (leading to oscillations) and the stopping criteria.

Description of Each Step:

Initialization. We generate randomly a sequence S with n points preferably in the middle of the unit cube, $S = \{x_i\}_{i=1,\ldots,n}$ in dimension s. At each step, each point pushes away its neighbors that are closer that $d_m = \frac{1}{\lfloor \sqrt[s]{n} \rfloor}$. Indeed, we know from inequality (1), that $\delta(S) \geq d_m$.

Spreading the Points. For each point x of S, we only consider as neighbors the points x_i of S with distance inferior to d_m. We compute a spring variable between x and each neighbor x_i defined by

$\left(\frac{2*d_m - d(x,x_i)}{2*d_m}\right)^p$. Parameter p is a positive integer and we have observed experimentally that $p = 4$ is satisfactory. With $p = 4$, the value of the spring variable varies from $1/16$ (further points) to 1 (nearest points). Then, we move point x proportionally to each spring variable into the direction of vector $x - x_i$: the closer the points are, the more the algorithm spaces them. Moreover proportionality used is decreasing in time until a threshold value, from which it becomes constant.

This process is similar to the *minimax* approach and pushes the points outside of the unit cube. Problem of generating a low dispersion sequence consists then in a minimization criteria with box constraints. We apply also these contraints: cube's borders are repulsive in the direction of cube's center, depending on the distance between these points and borders.

Applying box Constraints. In order to keep points inside the hypercube, we apply a repulsive force on points near borders. These points are detected with one of their coordinates which is inferior to $\frac{d_m}{2} + \varepsilon_m$ or superior to $1 - \frac{d_m}{2} - \varepsilon_m$: these values are the coordinates of extremum points of a Sukharev grid with a tolerance about $\varepsilon_m = \frac{d_m}{4}$. Intensity of this force has the same proprieties as forces used in step called *Spreading the points*. Globally, we perform a local dispersion minimization which becomes global after iterating this process.

Avoiding Configuration with Local Minimum. Applying iteratively these two previous steps can lead to local minimum and oscillations : a high number of points can be aligned on the edge. Repulsive forces push points on the same direction with the same intensity and the trend to push points outside hypercube at the step called *Spreading the points* nullifies the action of repulsive forces. There are then oscillations of these points. In order to avoid development of theses local minimum configurations, after a few number of iterations, we select randomly a point on each borders in each dimension and change their coordinate along these dimensions to a random value near the middle of cube.

Stopping Criteria: Different stopping criteria can be used to end the iterations: a maximal number of iterations, the stabilization of the dispersion or a minimum threshold on the average changes of the points during one iteration. This point still requires some further exploration.

6 CONCLUSIONS

In this paper, we illustrate experimentally the theoretical result established by (Gandar et al., 2009), showing that dispersion is probably a pertinent criterion for generating samples for classification and we deal with the question of generating the best low dispersion samples. We provide a quite simple algorithm able to minimize the dispersion for a fixed size sequence.In experimental design, grids are usually used to select sets of parameters for experiments. However, using grids imposes hard limits to the number of parameters that can be explored (often less than 6). We believe that being able to efficiently generate low dispersion sequences can help in this context, since the number of points can be fixed to any value and in any dimension (obviously, the number of points has be realistic depending on the dimension). In active learning, the learning algorithm has to select which training point will be used (implying that its label is asked for, which has a cost). Most of times, the training points pre-exist but it happens that one can ask for any point in the space. In that particular case, given a limited budget for labels (which provides the maximum number of training points), the proposed algorithm could be directly applied. Concerning the selection task, our algorithm can be adapted to be able to select an existing point nearby the ideal position. This is our future work.

REFERENCES

Gandar, B., Loosli, G., and Deffuant, G. (2009). How to optimize sample in active learning : Dispersion, an optimum criterion for classification ? In *European conference ENBIS European Network for Business and Industrial Statistics*.

Johnson, M., Moore, L., and Ylvisaker, D. (1990). Mimimax and maximin distance designs. *Journal of Statistical Planning Inference*, 26(2):131–148.

Lindemann, S. and LaValle, S. (2004). Incrementally Reducing Dispersion by Increasing Voronoi Bias in RRTs. In *IEEE International Conference on Robotics and Automation*.

Niederreiter, H. (1992). *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics.

Sergent, M., Phan Tan Luu, R., and Elguero, J. (1997). Statistical Analysis of Solvent Scales. *Anales de Quimica*, 93(Part. 1):3–6.

Teytaud, O., Gelly, S., and Mary, J. (2007). Active learning in regression, with application to stochastic dynamic programming. In *Proceedings of International Conference on Informatics to Control, Automation and Robotics*.