ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 283-288

Input pruning for neural gas architectures

Barbara Hammer[1] and Thomas Villmann[2]

(1) University of Osnabrück, Department of Mathematics/Computer Science, Albrechtstraße 28, 49069 Osnabrück, Germany, e-mail: hammer@informatik.uni-osnabrueck.de
(2) University of Leipzig, Clinic for Psychotherapy and Psychosomatic Medicine, Karl-Tauchnitz-Straße 25, 04107 Leipzig, Germany, e-mail: villmann@informatik.uni-leipzig.de

Abstract. The neural gas algorithm provides a method to cluster a data space via an adaptive lattice of neurons which captures the topology of the data space. We propose different methods to determine the relevance of the single data dimensions for the overall neural architecture. This enables us to perform input pruning for the unsupervised neural gas architecture. The methods are tested on various datasets.

1. Introduction

The neural gas algorithm (NG) or topology representing networks (TRN) are introduced in [10, 11] based on ideas of Kohonen's self organizing map [7]. They provide a clustering of the training data together with a neighborhood function on the centers of the clusters, the codebooks. The topology which is induced by this neighborhood function is detected by the algorithm itself and hence fitted to the respective data in contrast to the self organizing map where additional consideration may be required in order to find an appropriate topology [2, 5, 16, 17]. There exist various possibilities of further processing the outputs of the NG: attaching labels or local linear maps to the codebooks gives rise to general functions [7, 13], hence unsupervised methods can be involved in prediction tasks in machine learning. Substituting the input vectors by the distances to the codebooks yields a reduced representation of the original data [6], hence unsupervised methods can be used for preprocessing. Finally, the Delaunay graph computed by TRN allows structured data mining [8].

Often, data are high dimensional. Since time and memory required for training increases with increasing data dimension, this should be kept as small as possible, which can be interpreted as data pruning. Thereby data pruning is taken as weighting of the several input dimensions represented by the respective data coefficients. The possibility of pruning may give us further insight to the problem since it tells us which dimensions are relevant. Formally speaking, pruning means choosing input weights $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ which indicate whether input dimension $i \in \{1, \ldots, n\}$ is important for the overall function or not.

There exists a couple of pruning algorithms for *supervised* neural networks: weight decay, skeletonization, optimum brain damage, ... [9, 14]. Several ideas can be transferred to unsupervised methods *used for function approximation*

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 283-288

directly after additional changes such as substituting the winner-takes-all function by a differentiable version if necessary. One adaptation of these methods is presented in [12]. Other possibilities for dimensionality reduction are *statistical methods* such as principal component analysis [15] or semantic nets [8]. However, these methods are entirely *independent of the learning methods*. We would like to obtain pruning criteria which are *fitted to* the NG and, at the same time, do *not* assume that the NG is used for *supervised tasks*.

Now the question occurs of which are typical characteristics of unsupervised architectures? The architecture should somehow yield a compact representation of the data – the precise way in which this task is approached as well as a precise mathematical formulation of this task is often not obvious. The architecture and topology which we consider is maintained by NG or TRN; for convenience, we refer to NG in the following. We propose different objectives the sensitivities of which provide a ranking of the input weights Γ: We consider how pruning affects the *dispersion*, the *clustering*, the *energy function*, or the *topology* of the architecture. These methods are tested on artificial data with an obvious topology as well as data from the UCI repository [3].

2. The neural gas algorithm

Assume a finite set of training data $X = \{x^1, \ldots, x^m\} \subset \mathbb{R}^n$ is given. The single components of a vector $x \subset \mathbb{R}^n$ are denoted by $(x_1, \ldots, x_n)$ in the following. The goal of NG is to spread codebook vectors $w^1, \ldots, w^K$ among X, K being a fixed number of codebooks which is chosen a priori, such that they mirror the dataset as accurately as possible. For this purpose, the codebooks are initialized with random vectors and iteratively updated as follows:

repeat: *choose* $x^i \in X$
for all codebooks w^j
$w^j := w^j + \epsilon \cdot h_\lambda(k^j(x^i, w)) \cdot (x^i - w^j)$

where $h_\lambda(t) = \mathrm{e}^{-t/\lambda}$ and $k^j(x^i, w)$ is the number of codebook vectors w^l, $l \neq j$, such that $|w^l - x^i| < |w^j - x^i|$. ϵ and λ are positive numbers which are often decreased during training in order to ensure convergence. The update corresponds to a stochastic gradient descent on the cost function

$$E(w, \lambda) = \frac{1}{2 \cdot C(\lambda)} \sum_{i=1}^{K} \sum_{j=1}^{m} h_\lambda(k^i(x^j, w)) |x^j - w^i|^2$$

where $C(\lambda) = \sum_{i=0}^{K-1} h_\lambda(i)$ [10]. Under idealized assumptions the dynamic approaches the superposition of minimization of a potential induced by the data distribution and a force towards directions with a low density of codebooks.

After training the codebooks maintain a clustering of X. The cluster represented by w^i is the receptive field $R_i = \{x \in X \mid \forall w^j \, (j \neq i \rightarrow |x - w^i| \leq |x - w^j|)\}$. Additionally, a neighborhood on the codebooks is provided as follows: those codebooks w^i and w^j are neighbored such that $R_i \cap R_j \neq \emptyset$. Reference [11] provides a possibility to compute the neighborhood iteratively during the training process. After training, two codebooks are neighbors iff they constitute the closest codebooks for some training point.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 283-288

3. Pruning methods

We would like to determine the smallest set $I = \{i_1, \ldots, i_p\} \subset \{1, \ldots, n\}$ such that the behavior of the NG architecture reduced to dimensions I remains (approximately) the same. Two questions arise in this context: What are adequate characterizations of the behavior of the architecture? How can we avoid the necessity to consider all subsets of $\{1, \ldots, n\}$? We propose a greedy heuristic for the latter question, i.e. we iteratively compute and rank the input weights Γ according to some significance measure and prune the least important input dimension. To some extend we *have to* use heuristics:

Theorem 1 *Assume $C \geq 1$ is fixed. For any $Y \subset \mathbb{R}^n$ and indices $I = \{i_1, \ldots, i_j\} \subset \{1, \ldots, n\}$ denote the restriction of Y by $Y_I = \{(p_{i_1}, \ldots, p_{i_j}) \,|\, p \in Y\}$. Unless* P=NP*, no polynomial time algorithm can solve the following task: Given $n > 0$ and a finite set $Y \subset \mathbb{R}^n$ as input. Find indices $I \subset \{1, \ldots, n\}$ such that $|Y_I| = |Y|$ and $|I| \leq C \cdot \min\{|J| \,|\, |Y_J| = |Y|\}$.*

Proof: The hitting set problem is the following problem: Given a finite set Z and subsets $S_1, \ldots, S_p \subset Z$, find $Z' \subset Z$ such that $S_i \cap Z' \neq \emptyset$ for all i and the cardinality of Z' is at most C times the cardinality of the smallest set such that this condition holds. This problem is NP-complete [1]. We reduce the above problem to the problem as stated in the theorem.

Given an instance $Z = \{m^1, \ldots, m^n\}$, $S_1, \ldots, S_p$ of the hitting set problem, define $n = |Z|$ and $Y = \{(0, \ldots, 0)\} \cup \{i \cdot e^{S_i} \mid i = 1, \ldots, p\}$ where $e^{S_i} \in \mathbb{R}^n$ is the vector with

$$e_j^{S_i} = \begin{cases} 1 & \text{if } m^j \in S_i\,, \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, $|Y_I| = |Y|$ holds for $I \subset \{1, \ldots, n\}$ iff $\{m^i \mid i \in I\}$ is a hitting set. Assumed we could find indices I such that $|Y_I| = |Y|$ and $|I|$ is at most C times the minimum achievable size, $\{m^i \mid i \in I\}$ would constitute a hitting set the size of which is at most C times the minimum achievable size. □

Hence we cannot efficiently find indices such that the codebooks remain mutually disjoint and the number of indices is close to the optimum achievable number unless P=NP – we *have to rely on heuristics for input pruning!* Which are possible functions $F : \Gamma \to \mathbb{R}$ determining the input weights and hence measuring the importance of the input dimensions? As above, denote the codebooks by $w^1, \ldots, w^K$ and the training data by $X = \{x^1, \ldots, x^m\}$. We propose the following measures:

Dispersion: For clustered data the patterns belonging to a cluster are closer to the center than other points. Define

$$F_1(\gamma_I) = 2 - \frac{1}{K} \cdot \sum_{j=1}^{K} \frac{\sum_{x^l \in R_j} |x_I^l - w_I^j| / |R_j|}{\sum_{x^l \in X} |x_I^l - w_I^j| / |X|}$$

where R_j denotes the receptive field of w^j. F_1 measures to which extend the dispersion of the Ith components of points *in the receptive field* is reduced in comparison to the dispersion of the Ith components of *all points*. Those γ_I where the dispersion is not reduced, i.e. $F_1(\gamma_I)$ is small, are less important.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 283-288

Weight function: One can assign the function $f : \mathbb{R}^n \to \mathbb{R}^n$ to the NG architecture, which maps an input to the closest codebook vector. Those input weights are important which most affect this mapping. Note that this mapping is piecewise constant. Hence it will not change at all if we prune only one dimension in a high dimensional data space. Therefore we approximate f by

$$\tilde{f}(x) = \sum_{j=1}^{K} w^j \cdot \frac{\mathrm{e}^{-|x-w^j|^2/\beta}}{\sum_{o=1}^{k} \mathrm{e}^{-|x-w^o|^2/\beta}}$$

with some value $\beta > 0$, $\tilde{f}$ being differentiable. For the sake of efficiency we approximate the difference of the above term and the term with Ith component fixed to 0 by the derivative in the direction I and obtain the significance measure $F_2(\gamma_I) = 2/(Knm\beta)\cdot$

$$\sum_{l=1}^{m} \left| \frac{\sum_{j=1}^{K}(x_I^l - w_I^j)^2 \mathrm{e}^{lj} \cdot \sum_{j=1}^{K} w^j \mathrm{e}^{lj} - \sum_{j=1}^{K} \mathrm{e}^{lj} \cdot \sum_{j=1}^{K} w^j (x_I^l - w_I^j)^2 \mathrm{e}^{lj}}{\left(\sum_{j=1}^{K} \mathrm{e}^{lj}\right)^2} \right|$$

where $\mathrm{e}^{lj} = \mathrm{e}^{-|x^l - w^j|^2/\beta}$.

Cost function: NG minimizes the function $E(w, \lambda)$ via a stochastic gradient descent. Those input weights are to be ranked important for the architecture which affect this minimum most. Again, we approximate the piecewise constant k^i occurring in E by a differentiable function: $\tilde{k}^i(x, w) = \sum_{o=1}^{K} \mathrm{sgd}((|x-w^i|^2 - |x - w^o|^2)/\beta)$ where $\mathrm{sgd}(t) = (1 + \mathrm{e}^{-t})^{-1}$ and $\beta > 0$. We approximate the induced significance measure by the derivative in the direction of I:

$$F_3(\gamma_I) = \frac{1}{\beta C(\lambda) Kmn} \left| \sum_{i=1}^{K} \sum_{j=1}^{m} h'_\lambda(\tilde{k}^i(x^j, w)) |x - w^i|^2 \right.$$
$$\left. \sum_{o=1}^{K} \mathrm{sgd}'((|x^j - w^i|^2 - |x^j - w^o|^2)/\beta)((x_I^j - w_I^i)^2 - (x_I^j - w_I^o)^2) \right| .$$

Topology preservation: The NG architecture provides a lattice which mirrors the topology of the data. Those weights γ_I are less important which do not affect the topology. We approximate the neighborhood via the definition: two codebooks w^i and w^j are neighbored $\iff$ the point in between, $(w^i + w^j)/2$, is closest to w^i and w^j. This depends on the codebooks *only* and approximates the neighborhood structure unless the data manifold is not reasonably convex. Hence those input weights are less important where all neighbors remain neighbored codebooks after pruning, i.e. $\sum_{w^i \leftrightarrow w^j} \sum_{k \neq i,j} \mathrm{H}(|w^k|^2 - w^k(w^i + w^j) + w^i w^j) = 0$ after pruning, H denoting the step function and $w^i \leftrightarrow w^j$ being a shorthand notation for neighbored codebooks. Again we approximate via a differentiable function and the derivative in direction I: $F_4(\gamma_I) = 2/(K^2\beta)\cdot$

$$\left| \sum_{w^i \leftrightarrow w^j, k \neq i,j} \mathrm{sgd}'((|w^k|^2 - w^k(w^i + w^j) + w^i w^j)/\beta)((w_I^k)^2 - w_I^k(w_I^i + w_I^j) + w_I^i w_I^j) \right|$$

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 283-288

4. Experiments

Artificial data: We consider two dimensional data which are either uniformly distributed or form uniformly distributed or random clusters in $[-1,1] \times [-1,1]$. We add four dimensions with uniform noise in $[-0.5, 0.5]$, $[-0.35, 0.35]$, $[-0.2, 0.2]$, and $[-0.05, 0.05]$, respectively, and four copies of the first component with uniform noise from the above intervals added. Hence dimensions 1 and 2 provide structure information, dimensions $3-6$ contain pure noise – which may contribute to the overall structure for unsupervised processing, either; dimensions $7-10$ partially substitute dimension 1, with additional structure due to the noise. We train a NG architecture with various numbers of codebooks. F_1 separates dimensions without structure (F_1 yields values of approximately 1) and dimensions with structure (F_1 yields values of approximately 2). The remaining measures yield comparable results and rank dimension 2 and at least one of dimensions 1, 7, and 8 high. Depending on whether more codebooks than clusters are available, dimensions 3 or 4 which contain additional structure due to random noise with large variance are ranked high either.

Iris data: The task is to predict three classes from 4 real valued attributes in 150 instances. Denote by (x, y) the inputs and unary encoded labels. We train on (x, y). We map inputs x to y_i, assumed (x_i, y_i) is the closest codebook to $(x, 0)$. An architecture with 6 codebooks yields 3 misclassifications. Pruning dimensions $1, 2, 3$, or 4 leads to 4, 4, 6, and 5 misclassifications and suggests a ranking $3, 3, 1, 2$ of the dimensions. F_1 ranks all dimensions as important, i.e. yields values approximately 2, the other measures provide the ranking $1, 4, 2, 3$ for F_2, $4, 1, 2, 3$ for F_3, or $4, 3, 1, 2$ for F_4 as can be seen in Table 1. F_4 precisely matches the significance, F_2 and F_3 differ in only one dimension.

Mushroom data: The task is to predict two classes from 22 symbolic attributes. Unary encoding of the attributes yields 112 dimensions and 8124 instances, hence a difficult task for unsupervised methods. Accordingly, we obtain an error of about 10% with 8 codebooks. F_1 ranks about half of the dimensions as important (values approximately 2), the remaining dimensions as less important (values approximately 1). The other measures provide the five highest values for the input weights γ_I, I being $(26, 54, 36, 37, 108)$ for F_2, $(100, 56, 36, 37, 26)$ for F_3, or $(50, 51, 91, 26, 36)$ for F_4. This comprises weights corresponding to dimensions encoding attributes with only few possible values (e.g. 36) and hence large separation ability. Additionally, all measures rank 26 (which corresponds to the attribute 'no odor') high. According to [4] where logical rules are extracted from supervised classifiers, this is one of the most significant attributes for this task which allows – together with one additional attribute – an accuracy of more than 95%.

	1	2	3	4	ranking
errors	4	4	6	5	3 3 1 2
F_2	0.73	0.32	0.64	0.33	1 4 2 3
F_3	0.73	1.74	1.64	0.81	4 1 2 3
F_4	0.75	0.78	1.67	0.97	4 3 1 2

Table 1: Significance measures for the iris-database

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 283-288

5. Conclusions

We proposed a method for determining the relevance of input dimensions for the unsupervised NG architecture. The significance measures are based on values which capture certain aspects of the NG architecture: The reduction of the dispersion due to the clustering, the effect on the classification, the effect on the function which is minimized during training, and the effect on the topology. The four different measures are tested on artificial data sets where the topology is known as well as benchmarks from the UCI repository. It turns out that the first measure provides only rough estimations, whereas the other 3 measures provide promising results and enable us to perform efficient input pruning. Obviously, the optimum choice of the significance measure depends on which aspects of the NG architecture are most important for the respective behavior in the concrete learning task. Other measures such as referring to the entropy or expert knowledge constitute reasonable alternatives to the proposed significance measures. Further work will lie in applications where no labeling is provided and we are mainly interested in the topology of the unsupervised neural architecture such as the organization and visualization of documents or large databases and automated keyword or feature extraction.

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, Complexity and Approximation, Springer, 1999.

[2] H.-U. Bauer and K. Pawelzik, Quantifying the neighbourhood preservation of self-organizing feature maps, IEEE Transactions on Neural Networks 3(4), pp.570-579, 1992.

[3] C.L. Blake and C. J. Merz, UCI Repository of machine learning databases , Irvine, CA: University of California, Department of Information and Computer Science.

[4] W. Duch, R. Adamczak, and K. Grąbcweski, Extraction of crisp logical rules using constrained backpropagation networks, in M. Verleysen (ed.), Proceedings of ESANN'97, D-facto Publications, 1997.

[5] B. Fritzke, Growing self-organizing networks – why?, in M. Verleysen (ed.), Proceedings of ESANN'96, D-facto Publications, pp.61-72, 1996.

[6] S. Kaski, Dimensionality reduction by random mapping: fast similarity computation for clustering, in Proceedings of IJCNN'98, pp.413-418, 1998.

[7] T. Kohonen, Self-Organizing Maps, Springer, 1997.

[8] K. Lagus, T. Honkela, S. Kaski, and T. Kohonen, WEBSOM for textual data mining, Artificial Intelligence Reviews 13 (5/6), pp.345-264, 1999.

[9] Y. LeCun, J. Denker, and S. Solla, Optimal brain damage, in D. Touretzky (ed.), Advances in NIPS 2, Morgan Kaufmann, pp. 598-605, 1990.

[10] T.M. Martinetz, G. Berkovich, and K.L. Schulten, 'Neural-Gas' network for vector quantization and its application to time-series prediction, IEEE Transactions on Neural Networks 4(4), pp.558-569, 1993.

[11] T.M. Martinetz and K.L. Schulten, Topology representing networks, Neural Networks 7(3), pp. 507-522, 1993.

[12] U. Matecki, Automatische Merkmalsauswahl für Neuronale Netze mit Anwendung in der pixelbezogenen Klassifikation von Bildern, Shaker, 1999.

[13] A. Meyering and H. Ritter, Learning 3D-Shape-Perception with Local Linear Maps, Proceedings of the IJCNN'92, pp.432-436, 1992.

[14] M. Mozer and P. Smolensky, Skeletonization: a technique for trimming the fat from a network via relevant assessment, in D. Touretzky (ed.), Advances in NIPS 1, Morgan Kaufmann, pp.107-115, 1989.

[15] E. Oja, Principal component analysis, in M. Arbib (ed.), The Handbook of Brain Theory and Neural Networks, MIT Press, pp.753-756, 1995.

[16] H. Ritter, Self-organizing maps in non-euclidean spaces, in E. Oja, S. Kaski (eds.), Kohonen Maps, pp.97-108, 1999.

[17] T. Villmann, R. Der, M. Herrmann, and T. M. Martinetz, Toplogy Preservation in Self-Organizing Feature Maps: Exact Definition and Precise Measurement, IEEE Transactions on Neural Networks 8 (2), pp. 256 - 266, 1997.