2010 International Conference on Pattern Recognition

Exploring Pattern Selection Strategies for Fast Neural Network Training

Szilárd Vajda, Gernot A. Fink
Department of Computer Science
TU Dortmund
Otto-Hahn Str. 16, 44221, Dortmund, Germany
{*szilard.vajda, gernot.fink*}*@udo.edu*

***Abstract*—Nowadays, the usage of neural network strategies in pattern recognition is a widely considered solution. In this paper we propose three different strategies to select more efficiently the patterns for a fast learning in such a neural framework by reducing the number of available training patterns. All the strategies rely on the idea of dealing just with samples close to the decision boundaries of the classifiers. The effectiveness (accuracy, speed) of these methods is confirmed through different experiments on the MNIST handwritten digit data [1], Bangla handwritten numerals [2] and the Shuttle data from the UCI machine learning repository [3].**

***Keywords*-fast pattern selection; neural networks; machine learning;**

I. Introduction

In the last few decades, the popularity of different neural network (NN) strategies increases in the domain of pattern recognition, signal processing, image processing, etc. This is due to the fact, that NNs can easily be manipulated, the network topology can be adapted to different problems [4], [1] and the training mechanisms are defined with a strict mathematical rigor [4].

However, the main asset is that the NN can define its own classification rules automatically during the training process, without considering any a-priori knowledge. A network generalizes if and only if the internal rule can classify both training and test set, respectively [5]. If there is no a-priori knowledge about the data, to achieve generalization during the training, a huge amount of data should be considered for learning purpose. Hence, the training becomes quite a challenging task. Moreover, the computational complexity of such a training is high and we have to deal with the *course of dimensionality* [4]. Therefore, to achieve a fast and good generalization over the training process, an accurate pattern selection strategy is an important task in the classification chain.

The training of such a NN consists in updating the different weights in between the connected neurons belonging to different layers. This set of updated weights defines the hyper-planes which separate the input space into the corresponding class boundaries. Different authors [5], [6] have shown that the pertinent information for a better generalization lies close to the class boundaries. Our selection strategies are guided by these conclusions.

In this paper, we explore three new strategies to select the relevant patterns for a faster training. The first method selects the hard patterns using the same network for selection and training incrementally, while the last two methods select first the hard patterns close to the class boundaries and afterwards a NN is trained. We validated and compared our selection methods considering several benchmark data sets.

II. Related work

Many researchers proposed different data selection strategies to reduce the amount of data and to achieve faster training and better generalization. One trend is the so-called *incremental learning* defined by Engelbrechts [7]. In this framework the classifiers start with an initial subset selected somehow from a candidate set. During the learning process, at specified selection intervals, some new samples are selected based on some error measures [8] from the candidate set and these patterns are than added to the training set. While these *wrapper* methods use the target classifiers, some other techniques based on *filtering* do not use the classifier itself. They do some pre-selection on the data first and afterwards a classifier is trained with that data.

The main idea for this second trend, especially considered to train multilayer neural networks [5], lies in the fact that the decision surfaces (hyper-planes) defined during the training procedure mainly depend on the training samples adjacent to the decision boundaries.

In [5] the authors propose a method which selects randomly "pair data" based on the Euclidean distance, selecting in a binary scheme the closest patterns belonging to different classes. The method can be extended for multiclass problems and promising results have been reported for the Iris data [3].

The dataset condensing method described by Choi et al. [6] is based on a k-nearest neighbor method to estimate the posterior probabilities. By definition, the data which lies on the decision surface has a posterior probability of 0.5, while the rest is far from the decision surface and, therefore, of minor importance to proper training. Although, the results reported are impressive, they have applied this method just to binary classification tasks. A similar strategy was proposed in [9] to reduce the input data for a SVM based training. In order not to affect the SVM accuracy the authors proposed a heuristic as well for the estimation of the k value based on neighbors' entropy.

All these strategies describe similar solutions to detect such items from a dense data set which can really contribute with relevance to the learning process. Our algorithms also exploit similar ideas of looking for those special (relevant) items close to the decision boundaries.

III. Pattern selection strategies

The objective of the pattern selection strategies is to minimize the data requirements of learning. The selection process can be considered a kind of data compression, by selecting a relevant subset from the existing data.

A. Hard patterns selection (HPS)

The goal of HPS is to build at run-time a learning corpus based on least mean square by selecting the "hard patterns" during the training from the available training set. Unlike Engelbrecht [7], we propose a selection criterion based on the error committed by the classifier, selecting those candidates that have been misclassified previously.

Let $D_n = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be a training set where $x_i \in R^q$ and $y_i \in \{C_1, C_2, \ldots, C_M\}$ where n denotes the number of samples in the corpus, q is the input space dimension, M stands for the classes, while C_i is the corresponding class label. The objective is to achieve good generalization by adjusting the weights by learning. This is a nonlinear optimization problem, where we minimize the additive error function:

$$E(D_n|W) = \sum_{i=1}^{n} E(y_i|x_i, W). \qquad (1)$$

E stands for the error, while W is the NN's weights. The training starts with a reduced subset $D_0 \subset D_n$ and increases the training set incrementally. Rather than attempting to minimize the error function (Eq.1), we minimize the error components:

$$E(D_n|W) = E(D_{n_0}|W) + \ldots + E(D_{n_k}|W). \qquad (2)$$

n_i is the i^{th} size of the training set satisfying the relation: $D_{n_0} \subset D_{n_1} \subset \ldots \subset D_{n_k} \subset D_n$ where $n_0 < n_1 < \ldots < n_k << n$.

Let us denote by *Global Learning Corpus* (GLC) the whole available data, by *Global Validation Corpus* (GVC) the patterns which guide the training, by *Global Test Corpus* (GTC) the test patterns and by *Dynamic Learning Corpus* (DLC) the minimal set of patterns contributing to the training. Let us also denote by NN the neural network and by N the iterator which provides the number of new pattern candidates to be added at each stage. M denotes the number of classes. The detailed algorithm description can be found in Fig. 1.

```
Initialization:
  $DLC = \{x_i \in GLC \mid i = \overline{1, M}\}$
  $GLC = GLC \setminus DLC$
Database Building:
do
  do
    NN← TrainNet(DLC)
  while (NetErr(NN,DLC)> $\lambda_1$)
  if(NetErr(NN,GVC)< $\lambda_2$) then STOP end if
  else
    if(NetErr(NN,GLC)< $\lambda_1$) then STOP end if
    else
      $DLC = DLC \cup \{x_j \in GLC \mid j = \overline{1, N}\}$
      $GLC = GLC \setminus DLC$
    end else
  end else
while ($|GLC| > 0$)
Results:
NN contains the modified weight set
DLC contains the selected patterns subset
```

Figure 1: Hard pattern selection algorithm

The algorithm initializes the DLC set by selecting one sample (x_i) for each class. Then it performs the training with these samples until the error is less than a threshold (λ_1). The NN is tested with GVC. If the error criterion is satisfied, the algorithm stops. Otherwise, we pick-up the N worst recognized elements from GLC and move them into DLC and restart the training. The algorithm stops when the error is reduced or there are no more available patterns in GLC.

B. Support vectors selection (SVS)

Support Vector Machines (SVM) are a statistical learning architecture that perform structural risk minimization by defining margins between the data to be separated. The patterns that belong to the margins (Fig. 2) are the critical ones –so called *Support Vectors* (SVs)–, hence directly affecting training [9].

While Shin and Choi [10] proposed some initial pattern selection to avoid the computational burden in the quadratic programming (QP) for the SVM training, we select those SVs to train a NN. The method performs first an optimization which yields the largest margin between classes. Secondly, all the samples $(x_i, y_i) \in D_n$ (previous notation considered) which satisfy the equation of the margins defined by the QP optimization are selected and considered to train the NN.

C. Density based selection (DBS)

Looking at the points in the Fig.3, we can easily distinguish the different clusters. This is due to the fact that within each cluster we have a typical density of points which is considerably higher than outside the cluster. This idea has been exploited by the DBSCAN algorithm [11]. Our strategy is also based on this presumption, namely, the samples that

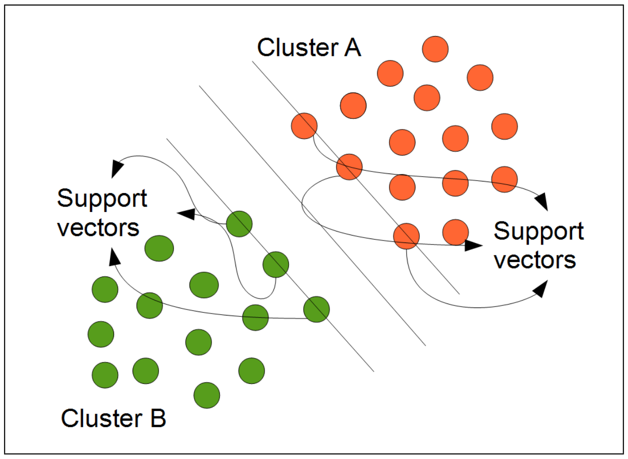

Figure 2: Support vectors defining the margins

lie at the margins of the clusters have less similar neighbors, so probably they are closer to the hyperplanes that separate the different class samples.
Let us denote by β-neighborhood of a point $(x_i, y_i) \in D_n$ the $N_\beta(x_i, y_i) = \{(x_j, y_j) \in D_n | dist(x_i, x_j) < \beta, i \neq j\}$, where $dist()$ is the Euclidean distance.

Considering the β-neighborhood of a point, we define the notion of (x_i, y_i) is density reachable from (x_j, y_j) if $(x_i, y_i) \in N_\beta(x_j, y_j))$, $|N_\beta(x_j, y_j)| \geq P_{min}$, where $y_i = y_j$ and P_{min} is the minimal number of identically labeled points which should be around point x_i.

The proposed data selection process is based on the β-neighborhood of a point. We select into the reduced dataset all the points which are not density reachable considering a given number of k (number of neighbors). The k parameter controls the degree of the data reduction. These points will be at the boundaries of the different classes. If the training can deal with this "boundary data" [5] there is no more need to train the samples which are inside the clusters.

IV. Results

A. Data description

The experiments were performed using several datasets. The MNIST corpus [1] contains $60,000$ and $10,000$ normalized handwritten digit data for training and testing, respectively. The ISI-Bengali handwritten digit data [2] contains 19,391 and 4,000 for training and testing. The digits in these datasets are gray-valued 28×28 pixel images. Finally, we also considered the Shuttle dataset [3] from the UCI repository, containing 43,500 and 14,500 samples, respectively. Each Shuttle data point is described by 9 features representing 7 different classes.

B. Network setup

For the experiments, a fully connected multi-layer perceptron with sigmoid transfer function has been used. Conventional error back-propagation minimizing a squared error metric was considered to train the network. The topology of the network was set based on the dataset and trial runs. The number of units in the input layer corresponds to the dimension of the input space, while the number of output units is the same as the number of classes to be separated. For the digit data we considered a 784-500-10 topology [8], while for the Shuttle data we used a 9-15-7 network setup.

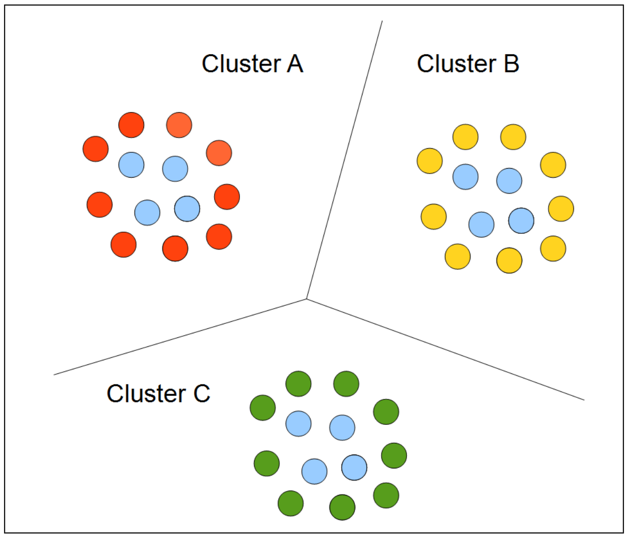

Figure 3: Data clusters with the extreme points

C. Results

Selecting only $1,960$ from $60,000$ samples of the MNIST data by the HPS, we obtained 98.36% recognition accuracy. While loosing 0.22% of accuracy using just 3.26% of the data we realize a 13x speed-up. Such a speed gain is enormous considering the tremendous time consumption to select the ideal topology and learning parameter for a neural network [6]. For the ISI data a 8x speed-up has been achived selecting only 1,210 samples from the possible $19,391$, but a 6.33% accuracy loss has been observed.

The Shuttle data is not suitable for the HPS method as a heavy unbalanced aspect can be detected between classes.

The SVS and DBS results are reported in Tab.I and Tab.II, respectively. The speed results for the SVS were calculated based on the SVM training, SVs selection and the NN training. Similarly, for the DBS the time consumed for the density reachability calculus and the NN training has been considered. These results are less impressive in speed terms but still promising. The moderate speed gain is due to the fact that in this case the classifier has not been used during the selection and a totally independent method has been considered. The achieved accuracy for SVS (98.74%) is even higher than using the total data from MNIST, but the speed is just 3x faster than the original solution. For the Shuttle data the SVS method results in a 10x speed-up and a higher accuracy (99.82%).

Data	Orig[%]/samp.	Red.[%]/samp.	Speed
MNIST	98.59/60,000	98.74/12,223	3x
ISI	94.35/19,391	94.40/9,415	1.5x
Shuttle	99.80/43.500	99.82/1,209	10x

Table I: Results concerning the SVS method

Data	Orig[%]/samp.	Red.[%]/samp.	Speed
MNIST	98.59/60,000	98.44/12,000	2.5x
ISI	94.35/19,391	92.38/5,817	2.6x
Shuttle	98.80/43,500	76.47/4,351	1.1x

Table II: Results concerning the DBS method

A kind of superiority can be distinguished for the SVS over the DBS method which can be explained by the fact that, the support vectors detected by the SVM training are optimized for the margins lying in between the different clusters. Meanwhile, the density based solution is more sensitive to noise and can introduce into the training set isolated elements, i.e. outliers which can distort the results.

V. Conclusion

In this paper we presented one wrapper method (HPS) and two filter methods (SVS, DBS) to select the "most relevant" patterns that help a neural network to achieve high accuracy and realize a considerable speed-up in the training procedure which can be helpful to test different network topologies and parameter settings.

While the filter method (HPS) produces an impressive 13x speed gain as the selection and training process fuse, the SVS method guided by SVs produces more accurate scores. The reported results are comparable with the state-of-the-art results (99.7%–MNIST[1], 98.2%–ISI[2], 99.8%–Shuttle[3]) but our primary goal is not to achieve the highest accuracy but rather to achieve a fast model training. This time gain will allow researchers to test several network topologies using different parameters to refine their recognition systems. The presented methods are general and our tests performed on different type of datasets show the success of these selection strategies.

References

[1] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Int. Conf. on Document Analysis and Recognition*, 2003, pp. 958–963.

[2] U. Bhattacharya and B. Chaudhuri, "Handwritten numeral databases of indian scripts and multistage recognition of mixed numerals," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, pp. 444–457, 2009.

[3] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*, 2nd ed. Springer, October 2007.

[5] K. Hara and K. Nakayama, "A training method with small computation for classification," *Int. Joint Conference on Neural Networks*, vol. 3, pp. 543–548, 2000.

[6] S.-H. Choi and P. Rockett, "Reducing the training times of neural classifiers with dataset condensing," in *Joint Int. Workshops on Advances in Pattern Recognition*, 2000, pp. 650–657.

[7] A. P. Engelbrecht, "Selective learning for multilayer feedforward neural networks," in *Int. Work-Conference on Artificial and Natural Neural Networks*, 2001, pp. 386–393.

[8] S. Vajda, H. Cecotti, Y. Rangoni, and A. Belaid, "A learning strategy using pattern selection for feedforward neural networks," in *Int. Workshop on Frontiers in Handwriting Recognition*, 2006, pp. 145–150.

[9] H. Shin and S. Cho, "Neighborhood property–based pattern selection for support vector machines," *Neural Comput.*, vol. 19, no. 3, pp. 816–855, 2007.

[10] ——, "Fast pattern selection for support vector classifiers," in *Advances in Knowledge Discovery and Data Mining*, 2003, pp. 376–387.

[11] M. Ester, H.-p. Kriegel, S. Jörg, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Int. Conf. on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.