Tangent Models: Comparing the TD-Neuron Constructive Algorithm versus SVD Based Algorithms

Diego Sona Alessandro Sperduti Antonina Starita
Dipartimento di Informatica, Università di Pisa
Corso Italia, 40, 56125, Pisa, Italy

Abstract

In this paper we present comparative results of our constructive algorithm versus the HSS. Specifically, we tested the HSS algorithm using both the original version based on the two-sided tangent distance, and a new version based on the one-sided tangent distance. Empirical results over the NIST-3 database show that the TD-Neuron is superior to SVD based algorithms since it reaches a better trade-off between error and rejection.

1 Introduction

In several pattern recognition systems the principal and most desired feature is the robustness against transformations of patterns. Different approaches to the problem aiming at the reduction of the classification time and space requirements, using tangent distance theory [SLD93], were studied by some authors.

Specifically, Hastie et al. [HSS95] developed rich models for representing large subsets of the prototypes through a Singular Value Decomposition (SVD) based algorithm, while Schwenk & Milgram [SM95b, SM95a] proposed a modular classification system (*Diabolo*) based on several auto-associative multi-layer perceptrons, which use tangent distance as the error reconstruction measure. Finally, Sona et al. [SSS97] devised a constructive algorithm based on the definition of a neuron (TD-Neuron) where the input net is computed by using the one-sided tangent distance instead of the standard dot product.

In this paper we compare the performance of Sona et al. [SSS97] constructive algorithm versus the HSS algorithm. In particular, we tested the HSS algorithm using both the original version, and a new version based on the one-sided tangent distance. The one-sided version of the HSS algorithm was derived in order to have a fair comparison against the TD-Neuron, which exploits the one-sided tangent distance. Empirical results over the NIST-3 database of handwritten digits show that the TD-Neuron is superior to both SVD based algorithms since it reaches a better trade-off between error and rejection.

2 Tangent Distance Overview

2.1 HSS models

Given a set of patterns $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$, all of the same class, Hastie et al. [HSS95] proposed the *tangent subspace model*, $\boldsymbol{W}(\boldsymbol{\theta}) = \boldsymbol{W} + \sum_{i=1}^{n} \boldsymbol{T}_i \theta_i$, where $\boldsymbol{W}$ is the *centroid* and the $\boldsymbol{T}_i$ constitute the associated invariant subspace. The model is computed minimizing over $\boldsymbol{W}$ and $\boldsymbol{T}_i$ the error function

$$\sum_{i=1}^{N} \min_{\boldsymbol{\theta}_i \ \boldsymbol{\alpha}_i} \|\boldsymbol{W}(\boldsymbol{\theta}_i) - \boldsymbol{X}_i(\boldsymbol{\alpha}_i)\|^2. \tag{1}$$

The above problem can be solved for a fixed subspace dimension by an iterative algorithm based on Singular Value Decomposition, proposed by Hastie et al. [HSS95]. The problem can also be formulated using the one-sided tangent distance, in this way the equation (1) becomes

$$\sum_{i=1}^{N} \min_{\boldsymbol{\theta}_i} \|\boldsymbol{W}(\boldsymbol{\theta}_i) - \boldsymbol{X}_i\|^2, \tag{2}$$

which is a way of representing the principal component analysis of the $\boldsymbol{X}_i$, also called *Karhunen-Loéve Expansion*.

2.2 TD-Neuron

The TD-Neuron is so called since it can be considered as a neural computational unit which receives an input vector $\boldsymbol{X}_k$, and computes (as input net) the square of the one-sided tangent distance of the input vector with respect to a set of internal parameters (weights). This set of parameters are organized in such a way to form a tangent model. Formally, we have

$$net_k = \min_{\boldsymbol{\theta}} \|\boldsymbol{W}(\boldsymbol{\theta}) - \boldsymbol{X}_k\|^2 + \beta, \tag{3}$$

where β is the offset. Under the condition that the tangent vectors constitute an orthonormal basis, equation (3) can exactly and easily be computed by using the projections of the input vector over the model subspace (see Figure 1)

$$net_k = \|\underbrace{\boldsymbol{X}_k - \boldsymbol{W}}_{\boldsymbol{d}_k}\|^2 - \sum_{i=1}^{n} [(\boldsymbol{X}_k - \boldsymbol{W})^t \mathbf{T}_i]^2 + \beta = \boldsymbol{d}_k^t \boldsymbol{d}_k - \sum_{i=1}^{n} [\underbrace{\boldsymbol{d}_k^t \boldsymbol{T}_i}_{\gamma_{ik}}]^2 + \beta. \tag{4}$$

The output of the TD-Neuron is then computed by transforming the *net* through a nonlinear monotone function f. In our experiments, we have used the symmetric sigmoidal function $o_k = \frac{2}{1+e^{net_k}} - 1$.

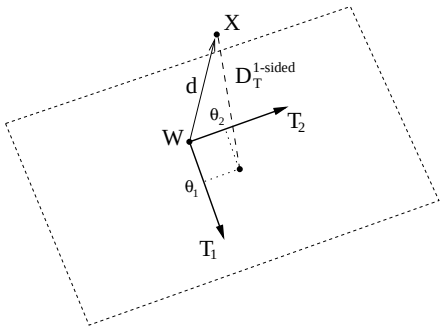

Figure 1: Geometric interpretation of equation (4). Note that $net = (D_T^{1-sided})^2$.

2.2.1 Training the TD-Neuron

Given a training set $\{(\boldsymbol{X}_1, t_1), \ldots, (\boldsymbol{X}_N, t_N)\}$, where $t_i \in \{-1, 1\}$ is the i-th desired output, and N is the total number of patterns in the training set, an error function can be defined as $\mathcal{E} = \frac{1}{2}\sum_{k=1}^{N}(t_k - o_k)^2$, where o_k is the output of the TD-Neuron for the k-th input pattern.

Using equation (4) with the sigmoidal function, it is trivial to compute the changes for the centroid, the tangent vectors, and the offset, by using a gradient descent approach. Before training the TD-Neuron by gradient descent, however, the tangent subspace dimension must be decided. To solve this problem we have developed a constructive algorithm which adds tangent vectors one by one, according to the computational needs. This idea is also justified by the observation that a typical run of gradient descent, over equation (4), leads to the sequential convergence of the tangent vectors according to their relative importance. This behavior suggests starting the training using only the centroid and then adding tangent vectors as needed. Under this learning scheme (see Figure 2), the changes to the parameters can be computed as

$$\Delta \boldsymbol{W} = -\eta \left(\frac{\partial \mathcal{E}}{\partial \boldsymbol{W}} \right) = -2\,\eta \sum_{k=1}^{N} [(t_k - o_k)\, f' \boldsymbol{d}_k] \tag{5}$$

$$\Delta \boldsymbol{T}_i = -\eta \left(\frac{\partial \mathcal{E}}{\partial \boldsymbol{T}_i} \right) = -2\,\eta \sum_{k=1}^{N} [(t_k - o_k)\, f' \,\gamma_{ik}\, \boldsymbol{d}_k] \tag{6}$$

$$\Delta \beta = -\eta_\beta \left(\frac{\partial \mathcal{E}}{\partial \beta} \right) = \eta_\beta \sum_{k=1}^{N} [(t_k - o_k)\, f' \,] \tag{7}$$

where η and η_β are learning parameters.

On the basis of empirical evidence we have concluded that the learning phase of the centroid can considerably be reduced by initializing the centroid with the mean value of the patterns belonging to the positive class. We have also devised a "good" initialization algorithm for tangent vectors (see Figure 3) which tries

CONSTRUCTIVE ALGORITHM

Initialize the centroid $\boldsymbol{W}$
Update β and $\boldsymbol{W}$ with equations 7 and 5 till they converge
Freeze $\boldsymbol{W}$
REPEAT
    Initialize a new tangent vector $\boldsymbol{T}_i$
    Update $\boldsymbol{T}_i$ and β with equations 6 and 7, and orthonormalize $\boldsymbol{T}_i$ with respect to $\{\boldsymbol{T}_1, \ldots, \boldsymbol{T}_{i-1}\}$ till it converges
    Freeze $\boldsymbol{T}_i$
UNTIL new $\boldsymbol{T}_i$ gives little accuracy changes

Figure 2: The constructive algorithm for the TD-Neuron.

to minimize the drop in the input net for all the patterns due to the increase in the tangent subspace dimension. This is obtained by introducing a new tangent vector which mainly spans the residual subspace between the patterns in the positive class and the current model. In this way, patterns which are in the negative class will only be mildly affected by the new introduced tangent vector.

3 Results

We have tested our constructive algorithm versus the HSS algorithm (with the one-sided and the two-sided tangent distance) using 10704 binary digits taken from the NIST-3 dataset. The binary 128x128 digits were transformed into 64-grey level 16x16 format by a simple local counting procedure. The only one preprocessing transformation performed was the elimination of empty borders.

The training set consisted of 5000 randomly chosen digits, while the remaining digits were used in the test set. For each algorithm, a single tangent model for each class of digit was computed (for the two-sided version of the HSS algorithm we used 6 transformations for each pattern). The classification of the test digits was performed using the label of the closest model for HSS

TANGENT VECTOR INITIALIZATION

for each class c of patterns compute the mean value of differences between patterns and model: $\boldsymbol{d}_c = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{X}_i - \boldsymbol{W}(\boldsymbol{\theta}))$;
orthonormalize the mean value of differences $\boldsymbol{d}_c$ of all negative classes ($c \in \{x | x \text{ is a negative class}\}$);
orthonormalize the mean value of the positive class with respect to the other classes.

Figure 3: Initialization procedure for the tangent vectors.

# Tang.	**HSS 1-sided** % Cor	**HSS 2-sided** % Cor	**TD-Neuron** % Cor
0	83.19	85.33	92.41
1	88.50	89.59	94.01
2	89.45	91.36	94.65
3	92.13	93.92	95.04
4	93.16	95.25	95.18
5	94.06	95.81	95.46
6	94.36	96.07	95.72
7	94.86	96.20	95.97
8	95.50	96.48	95.93
9	95.85	**96.60**	96.13
10	95.88	96.41	96.23
11	95.71	96.20	96.30
12	95.78	96.39	96.32
13	95.86	96.42	96.44
14	96.13	96.28	96.46
15	**96.42**	96.18	**96.51**
1-NN with Euclidean distance			**96.84**

Table 1: The results obtained with models generated by both versions of HSS, the TD-Neuron, and Euclidean 1-NN algorithms.

or the highest output of the TD-Neurons. We have performed also a classification using the Nearest Neighbor rule (1-NN) with the Euclidean distance as the classification metric. In Table 1 we have reported the results obtained for different numbers of tangent vectors. From the results it can be noted that while the two-sided HSS algorithm does overfit the data after the 9th tangent, this is not true for the remaining two algorithms. Nevertheless all the models reach a similar performance with the same amount of parameters (the two-sided version of HSS with 9 tangents uses 6 additional tangents for input patterns), which is slightly below the performance attained by the 1-NN classifier using Euclidean distance. However the tangent models have the advantage of being less demanding both in space and response time.

Although from Table 1 it seems that the generated tangent models are equivalent, when introducing a rejection criterion (threshold over the difference between the first and the second best outputs) the model generated by the TD-Neuron outperforms the other two models, due to its discriminant capability (see Figure 4). Furthermore, introducing the rejection criterion leads also to the surprising result that the one-sided version of HSS performs better than the two-sided version.

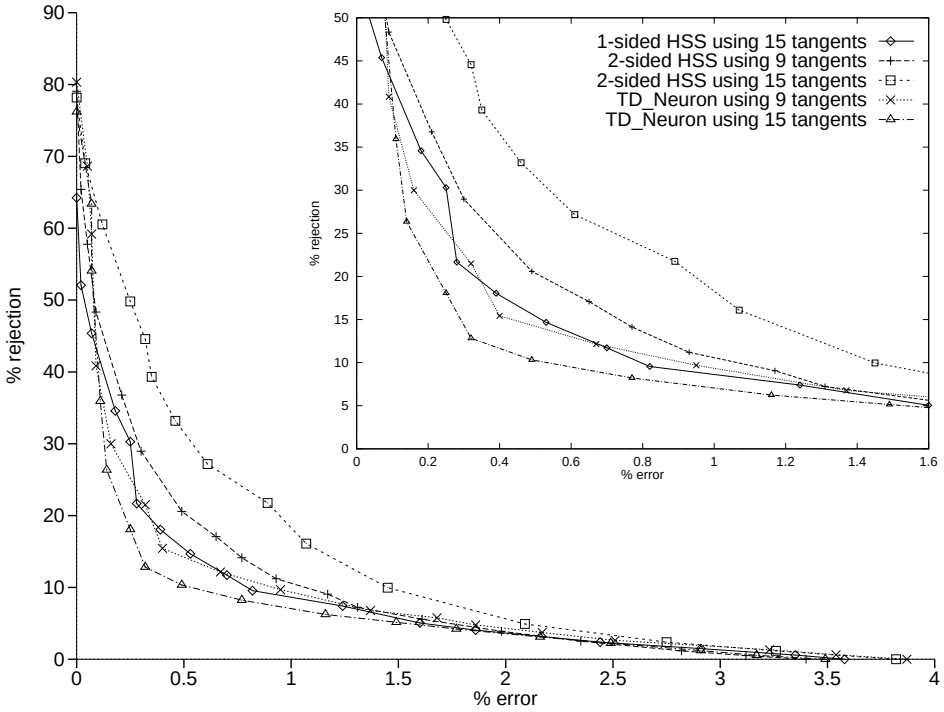

Figure 4: Error-rejection curves for the tangent models.

4 Discussion and Conclusion

In this paper we compared the TD-Neuron constructive algorithm versus two different versions of the HSS algorithm. The obtained results over the NIST-3 database of handwritten digits show that the TD-Neuron is superior to the HSS algorithms based on Singular Value Decomposition since it reaches a better trade-off between error and rejection.

It must be pointed out that, during the model generation, for a fixed number of tangent vectors, the HSS algorithm is faster than our, because it needs only a fraction of the training examples (only one class). However, our algorithm is remarkably more efficient when a family of tangent models, with an increasing number of tangent vectors, must be generated.

An additional advantage of TD-Neuron model is that, because the training algorithm is based on a gradient descent technique, several TD-Neurons can be arranged in a feed-forward network, which can be trained by a trivial extension of back-propagation. This may leads to a remarkable increase in the transformation invariant features of the system.

References

[HSS95] T. Hastie, P.Y. Simard, and E. Säckinger. Learning prototype models for tangent distance. In G. Teasauro, D.S. Touretzky, and T.K. Leen, editors, *NIPS*, volume 7, pages 999–1006, Cambridge MA, 1995. MIT Press.

[SLD93] P.Y. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S.J. Hanson, J.D. Cowan, and C.L. Giles, editors, *NIPS*, volume 5, pages 50–58, San Mateo CA, 1993. Morgan Kaufmann.

[SM95a] H. Schwenk and M. Milgram. Learning discriminant tangent models for handwritten character recognition. In *International Conference on Artificial Neural Networks*, pages 985–988. Springer-Verlag, 1995.

[SM95b] H. Schwenk and M. Milgram. Transformation invariant autoassociation with application to handwritten character recognition. In G. Teasauro, D.S. Touretzky, and T.K. Leen, editors, *NIPS*, volume 7, pages 991–998, Cambridge MA, 1995. MIT Press.

[SSS97] D. Sona, A. Sperduti, and A. Starita. A constructive learning algorithm for discriminant tangent models. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, *NIPS*, volume 9, pages 786–792, Cambridge MA, 1997. MIT Press.