

Generalized Similarity Metric Learning for a Variable Kernel Classifier.

J.J. Naudé^{1,2}, M.A. van Wyk¹

¹Rand Afrikaans University
Auckland Park, South Africa

²Kentron Dynamics
Centurion, South Africa

mavw@ing.rau.ac.za

jjnaude@ieee.org

Abstract

Proximity based classifiers such as RBF-networks and nearest-neighbour classifiers are notoriously sensitive to the metric used to determine distance between samples. In this paper a method for learning such a metric from training data is presented. This algorithm is a generalization of the so called Variable-Kernel Similarity Metric (VSM) Learning, originally proposed by Lowe [1] and is therefore known as Generalized Variable-Kernel Similarity Metric (GVSM) learning. Experimental results show GVSM to be superior to VSM for extremely noisy or cross-correlated data.

1. Introduction

In a classification problem, we are given T training observations belonging to I distinct classes. Each training observation consists of a D -dimensional feature vector $\bar{x}_t = (x_{t1}, \dots, x_{tD}) \in \mathbb{R}^d$ and the known class label $L_t, t = 1, \dots, T$. The goal is to predict the class label of a previously unseen query \bar{x}_0 . The K Nearest neighbour classification method is a simple and appealing approach to this problem: it finds the K nearest neighbours of x_0 in the training set and then predicts the class label of x_0 as the one that occurs most frequently in the K neighbours. Nearest neighbour classifiers allow rapid incremental learning from new instances and controlled removal of outdated or invalid training samples. Also, they are much easier to interpret than neural networks. Unfortunately their generalization performance is often inferior to competing methods.

As Lowe[1] has pointed out, the performance of these methods is highly dependent on the similarity metric that is used to select the neighbours. One example of a situation where an appropriate metric is important is the case where one or more of the features are irrelevant. Such a case is depicted in fig.1. Based on this Lowe[1] proceeded to develop a technique known as Variable Similarity Metric (VSM) learning to automatically scale the features appropriately and demonstrated that the generalization performance of this technique is state of the art. VSM learning can be run as a black box with no problem-specific parameters to be set by the user.

It has come to the authors' attention that while VSM learning performs well for cases where the input features are uncorrelated, simple feature scaling is not powerful enough once the noise added to different features becomes

cross-correlated (as depicted in fig. 2). One can reasonably expect that this will be the case in many pattern recognition applications, since all the features are derived from the same original input.

The solution proposed here is a generalized form of VSM learning (GVSM). GVSM optimizes more parameters in order to obtain a metric, the principal axes of which are not necessarily aligned with the coordinate axes. Unfortunately, as always, more degrees of freedom implies more vulnerability to overfitting.

In Section 2 we introduce the notation used for the remainder of the paper, while in Section 3 the equations governing K -nearest neighbour classification is discussed in more detail. In order to optimize the metric used, we need a measure of the performance of a metric and the partial derivatives of this measure with respect to the variables to be optimized, these are introduced in Section 4. Nearest neighbour methods are often perceived as more computationally expensive than competing methods, and for this reason Section 5 discusses some of the speed-up techniques that can fruitfully be applied. Section 6 reports some results obtained when GVSM was compared to VSM on synthetic data. We discuss the alternative interpretation of GVSM as a transform design method in Section 7 and our conclusions and suggestions for future research appear in the last section.

2. Preliminaries

For the rest of the paper s_{ti} will denote the known probability (ie. 1 or 0) that sample number $t \in 1, 2, \dots, T$ falls in class $i \in 1, 2, \dots, I$ and p_{ti} will denote the estimated probability that sample number t falls in class i based on the training set excluding sample t . Similarly s_{tki} will denote the known probability that the k -th nearest neighbour of sample number t from the training set falls in class i . \bar{x}_t and \bar{c}_{tk} will denote the feature vectors of the t -th sample and it's k -th nearest neighbour respectively.

3. K -nearest neighbour classification

The K -nearest neighbour technique uses the following expression to determine the probability that a sample belongs to class

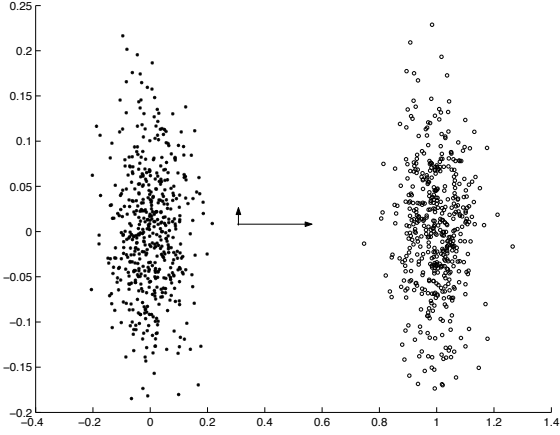


Figure 1: VSM learns a metric that accentuates differences along the x axis and suppresses those along the y axis. The arrows represent the kind of metric that VSM will typically learn for this case. (Classes are widely separated for illustrative purposes.)

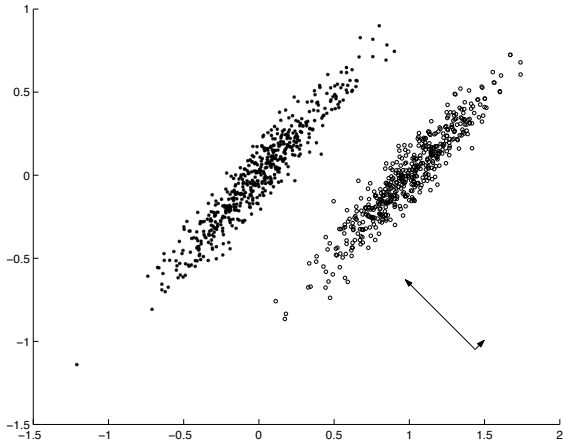


Figure 2: VSM is unable to learn an appropriate distance measure for cases where the noise affecting different features is correlated. The arrows represent the kind of metric that GVSM will typically learn for this case.

i

$$p_i = \frac{\sum_{k=1}^K n_k s_{ki}}{\sum_{k=1}^K n_k}$$

In the most basic form of the method all n_k coefficients are set to 1 and it becomes a simple vote. A slightly more sophisticated method attaches more importance to closer neighbours by determining the weight n_{tk} assigned to each neighbour using a kernel centered at x_t . In this case we will use a Gaussian kernel:

$$n_k = e^{-\frac{d_k^2}{2\sigma^2}}$$

The width of this kernel is determined by σ . If σ is too small, the truly nearest neighbour will dominate the decision and generalization will be poor. If it is too large, the method will fail to capture significant changes in the output. In general σ may be chosen smaller, the more densely the data is sampled. Since the density of data varies over the input space, fixed values of σ will not normally perform well. In order to make the width of the window vary with the density of available training samples, σ is set to some multiple of the average distance to the M nearest neighbours. It is better if only a fraction (e.g. $M = \frac{K}{2}$) of the nearest neighbours are used so the kernel becomes small even when only a few neighbours are close to the input.

$$\sigma = \frac{r}{M} \sum_{m=1}^M d_k$$

The difference between VSM and GVSM lies in a single equation. While VSM uses the expression

$$d_k^2 = \sum_{d=1}^D w_d^2 (x_d - c_{kd})^2$$

to define the distance between a sample x and its k -th nearest neighbour c_k , GVSM uses the more general matrix norm

$$d_k^2 = (\bar{x} - \bar{c}_k)^T A (\bar{x} - \bar{c}_k) \quad (1)$$

where A is a positive definite symmetric matrix. For the case where A is a diagonal matrix, this is exactly equivalent to VSM.

4. GVSM optimization

The first complication that arises when attempting to optimize a matrix norm is the fact that the matrix must be constrained to be symmetric positive definite.

A necessary condition for a matrix A to be a symmetric positive definite matrix, is that it can be expressed as $A = L^T L$ where L is upper triangular with positive diagonal elements. A sufficient condition for A to be symmetric positive definite, is that it can be written as $A = L^T L$ where L can be any non-singular matrix. Therefore if L is a non-singular upper triangular matrix

$$A = L^T L, \quad (2)$$

is a necessary and sufficient condition for A to be positive definite.

Expressing L as

$$L = \begin{bmatrix} L_{11} & L_{12} & L_{13} & \dots & L_{1d} \\ 0 & L_{22} & L_{23} & \dots & L_{2d} \\ 0 & 0 & L_{33} & \dots & L_{3d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & L_{dd} \end{bmatrix}$$

allows us to optimize the various elements L_{uv} assured in the knowledge that A will be a symmetric positive definite matrix and that all symmetric positive definite matrices can be obtained in this way. Although VSM learning uses conjugate-gradient descent to minimize the cross validation error over the training set we have only implemented a gradient descent method with a primitive form of line search for use with GVSM. However convergence is still attained within a reasonable time for most problems.

The cross validation error is defined as

$$E = \sum_t \sum_i (s_{ti} - p_{ti})^2.$$

The derivative of this error can be computed with respect to each of the parameters to be optimized to obtain

$$\frac{\partial E}{\partial L_{uv}} = -2 \sum_t \sum_i (s_{ti} - p_{ti}) \frac{\partial p_{ti}}{\partial L_{uv}}$$

where

$$\frac{\partial p_{ti}}{\partial L_{uv}} = \frac{\sum_k (s_{tki} - p_{ti}) \partial n_{tk} / \partial L_{uv}}{\sum_k n_{tk}}$$

and

$$\frac{\partial n_{tk}}{\partial L_{uv}} = \frac{n_{tk}}{2\sigma^2} \left(d_{tk}^2 \frac{2r}{M\sigma} \sum_{m=1}^M \frac{\partial d_{tkm}^2}{\partial L_{uv}} - \frac{\partial d_{tk}^2}{\partial L_{uv}} \right) \frac{\partial A_{op}}{\partial L_{uv}}, \quad (3)$$

$$\frac{\partial d_{tk}^2}{\partial L_{uv}} = (x_{to} - c_{tko})(x_{tp} - c_{tkp}) \frac{\partial A_{op}}{\partial L_{uv}},$$

$$\frac{\partial A_{op}}{\partial L_{uv}} = \begin{cases} 2L_{uv} & \text{if } o = p = v \\ L_{uo} & \text{if } o \neq p = v \\ L_{uv} & \text{if } p \neq o = v \\ 0 & \text{if } p \neq o \neq v \end{cases}.$$

In order to optimize the parameter r we simply use the derivative of n_{tk} with respect to r

$$\frac{\partial n_{tk}}{\partial r} = \frac{n_{tk} d_{tk}^2}{r\sigma^2}$$

in place of equation 3. For a d -dimensional input space GVSM optimizes $\frac{d(d+1)}{2} + 1$ parameters as opposed to the $d+1$ parameters optimized by VSM, which implies more power to select an appropriate metric, but also more potential for overfitting the data set. Thus VSM would be more appropriate if the size of the data set is small relative to the dimensionality of the input. On the other hand GVSM optimizes very few parameters compared to an equivalent neural net, which seems to indicate that the overtraining problem should not be excessive.

5. Improving run-time performance

Nearest neighbour methods are sometimes criticized for slow run-time performance. However with the correct optimizations nearest neighbour methods can actually outperform other algorithms. For example we can use the distributive law to expand our expression for distance

$$d_k^2 = (\bar{x} - \bar{c}_k)^T D (\bar{x} - \bar{c}_k)$$

to

$$d_k^2 = \bar{x}^T D \bar{x} - 2\bar{x}^T D \bar{c}_k + \bar{c}_k^T D \bar{c}_k.$$

The last term in this expansion is a constant that can be calculated at design-time for each exemplar in the database. The

first term is a constant that will be the same for all candidate neighbours and can thus be calculated once off and added to all candidates. However we can save computation by only calculating the pseudo-distance

$$\tilde{d}_k^2 = -2\bar{x}^T D \bar{c}_k + \bar{c}_k^T D \bar{c}_k,$$

then selecting the K nearest neighbours based on pseudo-distance and adding the $\bar{x}^T D \bar{x}$ term to each of these to obtain the true distances to the nearest neighbours. Since K is typically much smaller than the number of exemplars this saves a lot of addition. If we calculate the training vectors in the database as follows

$$\tilde{c}_k = \begin{bmatrix} -2\bar{c}_k^T D & \bar{c}_k^T D \bar{c}_k \end{bmatrix},$$

and concatenate them all into a database matrix

$$C = \begin{bmatrix} \tilde{c}_1 \\ \tilde{c}_1 \\ \vdots \\ \tilde{c}_T \end{bmatrix},$$

we may calculate the pseudo distance between the query vector and each of the samples in the database by the simple operation of augmenting the query vector

$$\tilde{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_D & 1 \end{bmatrix}^T,$$

and performing the Matrix multiplication

$$\tilde{d}_j = C \tilde{x}.$$

For problems with very large databases this matrix multiplication may take too long. In such cases the k - d tree algorithm [4,5] may be used to obtain further speedup. However the performance gain of this algorithm diminishes with increasing dimensionality of the input. Large databases with high dimensionality may require the use of approximate methods such as Best Bin First [6]. Using one or more of these speedup techniques very often results in better run-time performance than competing methods.

6. Test results

The GVSM algorithm was compared to VSM on the synthetic data set originally used by Lowe in [1] to demonstrate the working of VSM learning. The task is to solve a noisy XOR problem in which the first two real-valued inputs are randomly assigned values of 0 or 1 and the binary output class is determined by the XOR-function of these inputs. Noise was added to these 2 inputs drawn from a normal distribution with standard deviation of 0.3. The cross-correlation between the noise signals added to these two inputs α is the parameter against which we plot our results and varies from 0 to 0.95. The next two inputs were assigned the same initial 0 or 1 values as the first two, but had noise with standard deviation of 0.5 also correlated according to α . Finally another 4 inputs were added that had random zero mean values with a standard deviation of 2.

The training set consisted of 100 samples and the test set of 1000 samples. The presence of irrelevant features as well as less-important features makes this a very difficult task for classical nearest-neighbour classifiers to solve. As can be seen from fig. 3 both VSM and GVSM fare very well with GVSM

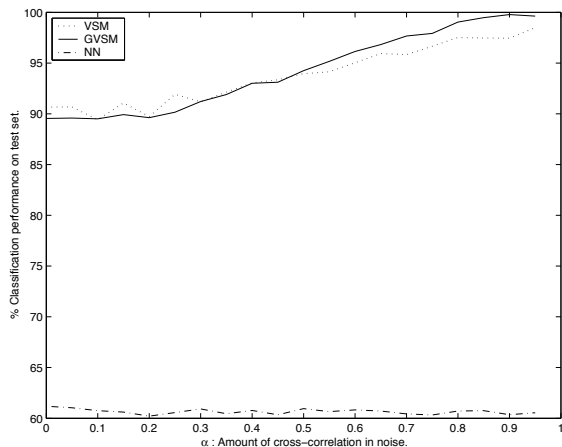


Figure 3: VSM, GVSM and untrained nearest-neighbour generalization performance on the synthetic data set plotted against the amount of correlation in the noise. Standard deviations of 0.3 and 0.5.

gaining a slight edge as the correlation increases.

An interesting result that was discovered only recently is that as the noise levels increase, GVSM becomes superior to VSM even for data in which the noise is not correlated. This result is shown in fig. 4 and was obtained by adding noise with a standard deviation of 0.55 and 0.7 respectively to the first two pairs of inputs. However on the real data-sets on which GVSM was evaluated VSM has consistently outperformed it. The reasons for this are not understood and is the topic of ongoing research.

7. Applications in feature generation

Suppose that we apply GVSM training to a training set which consists of raw data (e.g. a sampled speech signal or the pixel intensities of an image) rather than extracted features. If we substitute equation (2) into equation (1) we obtain

$$\begin{aligned}
 d_{tk}^2 &= (x_t - c_{tk})^T L^T L (x_t - c_{tk}) \\
 &= (L(x_t - c_{tk}))^T L(x_t - c_{tk}) \\
 &= C^T C
 \end{aligned}$$

where

$$C = L(x_t - c_{tk}).$$

This implies that the distance as measured by our metric is simply equivalent to Euclidian distance in the transformed space induced by L . In this way we may learn a linear transform that maximizes the class separation as measured by the cross-validation error of a K -nearest neighbour classifier.

Conversely, if we know that a specific general purpose linear transform, such as the Hadamard or Fourier transform, has good discrimination capabilities for a given problem we may use it as a starting point for GVSM learning and hopefully learn an even better transform specialized for the application. In addition GVSM automatically scales the various generated features, so feature selection reduces to selecting the features

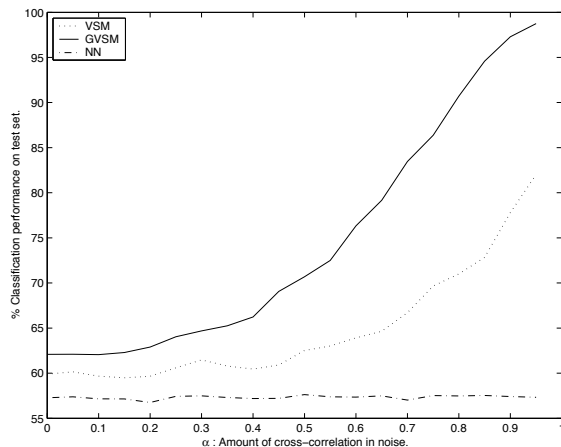


Figure 4: VSM, GVSM and untrained nearest-neighbour generalization performance on the synthetic data set plotted against the amount of correlation in the noise. Standard deviations of 0.5 and 0.7.

with the largest magnitudes.

8. Conclusion and future directions

A generalized method for learning a metric suitable for use with proximity based classifiers was presented. The superior performance of GVSM-learning on the synthetic XOR data set for increasing correlation seems to suggest that GVSM-learning can be a valuable tool to improve the generalization performance of proximity based classifiers. However, the absence of superior results on real data sets suggest that the conditions that GVSM was designed to exploit may not occur at significant levels in real datasets.

While GVSM consistently obtains better training performance than VSM the drop from training to test performance is also much bigger. This indicates that the poor test performance is due to overtraining, but why the overtraining penalty on GVSM should be so much larger than that experienced by a neural net with the same amount of parameters is not clear. Another promising approach that is currently being investigated is the use of VSM or GVSM to learn different norms for the different classes in a problem.

9. Acknowledgement

Kentron Dynamics is gratefully acknowledged for their financial contribution towards this work.

10. References

- [1] Lowe, D. G., "Similarity Metric Learning for a Variable-Kernel Classifier", *Neural Computation*, 7, 1 (January 1995), pp.72-85.
- [2] Cover, T.M., and Hart, P.E., Nearest neighbour pattern classification., *IEEE Transactions on Information Theory*, IT13,1,21-27, 1967.
- [3] Duda, R.O., and Hart, P.E., *Pattern Classification and Scene Analysis.*, New York, Wiley, 1973.

- [4] Friedman, J.H., Bentley, J.L., and Finkel R.A., An algorithm for finding best matches in logarithmic expected time., *ACM Trans. Math. Software*, 3, 209-226, 1973.
- [5] Sproull, R.F., Refinements to nearest-neighbour searching in k-d trees., *Algorithmica*, 6, 579-589, 1973.
- [6] Beis, J.S., Lowe, D.G., Shape indexing using approximate nearest neighbour in high-dimensional spaces., *Conference on Computer Vision and Pattern Recognition*, Puerto Rico, pp. 1000-1006, 1997.