Laboratory of Data Analysis
University of Jyväskylä

EUREDIT - WP4.5, WP5.5 Internal reports

READ D5.5.1 BEFORE THIS

D4.5.1 - Description of The Error Localication Methodology based on The Tree-Structured Self-Organizing Map

---

Pasi P. Koikkalainen - University of Jyväskylä
(Draft version 4. March 2002)

Contents

Summary

This report describes how the tree-structured self-organizing map (TS-SOM) can be used for error localication.

A more information about the methodology is given in the related document D5.5.1 (Imputation methodology) and its appendixes.
The methods are fully implemented under our NDA (Neural Data Analysis) software as described in the joint document of internal reports (D4.5.2 & D5.5.2).

1 Introduction

Error localization is always based on external knowledge. Techniques can be either strong or weak. Strong knowledge assumes that errors can be modelled, while weak knowledge expects that we are able to discriminate between acceptable and erroneous observations.

To use strong error detection one should be able to indentify the logical or the probabilistic nature of errors. Often this includes an understanding about the the mechanism that causes them. Yet some error types cannot be identified despite the underlying mechanism is known, and some can even if the actual mechanism is hidden. There are many statistical methodologies that are inherently based on strong knowledge, for example Bayesian approach requires that all causes behind data are modelled.

Weak knowlege is often easer to use than strong knowledge, especially when erroneous observations separate well from rest of data. In practice it is impossible to indentify all kinds of situations that can go wrong during data collection, and then the error localication mechanism can only relay on weak knowledge between "good" and "bad" data. This is typically done by modelling normal situations (good data) and measuring how probable it is that a given sample belongs to this category. If our measure indicates some type of novelty, then that observation is suspicious, likely to be an error.

The problem of weak knowledge is that it does not tell what to do with erroneous observations, that would require an understanding about the cause. We may, however, mark observed samples or variables as erroneous and impute them using suitable methodology.

Neural networks can be used for both strong and weak type of error localization. For known error classes, neural networks can be trained as classifers between good and bad data. Since this type of knowledge is rare, the use of weak knowledge is more common in neural systems. The objective is then to build a model that explains well all clean observations, but which gives low matching probabilities for erroneous ones. This can be done in two ways:

i) Clean data is used for model building. As most models are based on mean values, also a measure of accepted spread around the model is needed.

ii) When no clean training data is available, robut methods must be used for training. Then, according some criteria, samples that are suspicious are given less weight, or totally ignored, from the model. As well as in case i) a measure of accepted spread must be computed before actual error detection can be done.

1.1 About the measure of spread and robustness

At first we do not deny a possibility of using strong models with self-organizing maps.

Recall the idea of principal curves and surfaces. Our model model for $\mathbf{X}$ consists of a regression surface $\mathbf{x}(\mathbf{v})$ and randomness ϵ around it:

$$\mathbf{X} = \mathbf{x}(\mathbf{v}) + \epsilon.$$

The question is: *under which assumptions a sample* $\mathbf{X}(j)$ *is or is not explained by the model ?*

Assuming that we knwo what the true model is, a probablistic answer can be given through the cumulative distribution of the noise term

$$F_{\mathbf{Y}}(\mathbf{y}(\mathbf{x})) = \int f_Y(\epsilon|\mathbf{v})d\mathbf{y},$$

where $\mathbf{y}(\mathbf{x}') = \mathbf{x}' - \mathbf{x}(\mathbf{v})$ and $\mathbf{v}$ is the projection from observation space $\mathbf{x} \in R^M$ onto principal surface $\mathbf{v} \in R^S$

$$\mathbf{v} = \mathbf{v}'(\mathbf{x}') = \arg\min_{\mathbf{v}''} \|\mathbf{x}(\mathbf{v}'') - \mathbf{x}'\|.$$

As usual, for any single observation j the matching probability, better known as the goodness of fit, is

$$P_0 = \min\{\ F_{\mathbf{Y}}(\ \mathbf{y}(\ \mathbf{X}(j)\ ))\ ,\ 1 - F_{\mathbf{Y}}(\ \mathbf{y}(\ \mathbf{X}(j)\ ))\ \}.$$

Then, using classical decision theory we may rewrite the problem as hypothesis testing

$$\begin{array}{lll} \texttt{Acceptable} & \text{if} & P_0 > \theta \\ \texttt{Outlier} & \text{if} & P_0 < \theta, \end{array}$$

where θ is the rejection value of our hypothesis: $\mathbf{X}(j)$ is `Acceptable`.

But we do not know what the true model is, rather we must estimate it from data, which makes the problem more difficult. This is because the rejection boundary is also an estimate, that makes the P-value a random variable. Although a probabilistic solution is still possible in theory, assuming that our model is sufficient, it might be difficult to obtain in practice. It is also unlikely that we can find sufficient model for erroneuos observations.

Although there is nothing new in this argumentation, we are now motivated to believe that the use of strong knowlegde for probabilistic error detection with principal surfaces (and self-organizing maps) is practicly impossible.

1.2 Rejection of errors while training

When we accept that error mechanisms cannot be identified exactly, the next best thing is secure the modelling of the non erroneous part. This leads to weak methods for error detection. Since the use of weak knowledge is always more subjective than strong modelling, we can relax our methodology a little. For practical algorithms we divide our methodology into parts:

Step 1. Robust model building. This means the cleanning of errors during the learning, where we emphasize that erroneous observations should not influence the estimated model. Since is not the final error localication, we may ignore some portion of clean data as well.

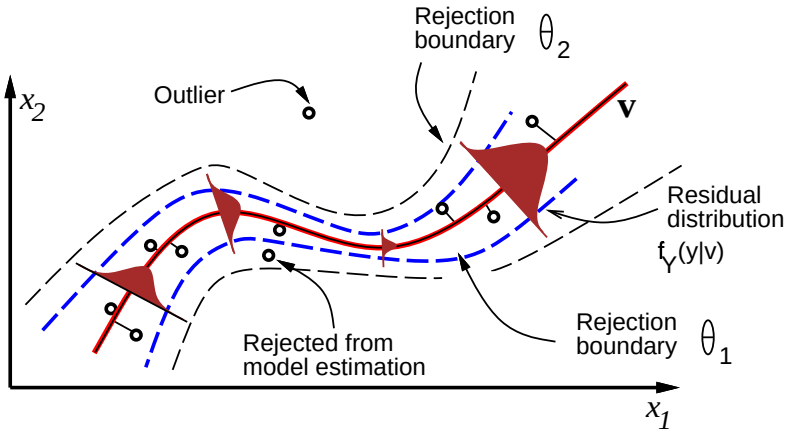
Step 2. Error localization. Here we are using the model of clean data, obtaind in step 1. Now the rejection probability must be selected more carefully, depending on the criteria of our application.

If needed, different mechanisms can be used for the two steps. This might be useful since there are more methods for robust model building than what there are for error detection. For example, some methods change the influence curve of model estimates but do not nessessary categorize any of the samples as errors. Also nonparametric methods that are independent of scale might be better founded for modelling than error detection, where scale is more closely related to the intepretation of the model, unless we magically know the percentange of errors in data.

Useful methodology is often based on relative simple methods. Despite this is not generally acceptable, we assume that $f_{\mathbf{Y}}(\epsilon|\mathbf{v})$ is gaussian, which allows us to use the same methdology for both robust model building, training with incomplete data, imputation and error detection.

This methodology has been implemented in the TS-SOM training algorithm such that two rejection values θ_1 and θ_2 can be given by the users, as depicted in Fig. 1.

Chart 1: Illustration of rejection bounds for principal curves.



2 Robut algorithm for the tree-structured self-organizing map.

Practical implementation of our robust modelling strategy is relatively easy to implement using the tree-structured self-organizing map. Since the principal curve is approximated with a set of nodes $\mathbf{v}_i$, the measure of spread can easily be done locally. For every node we estimate distribution of normal noise $f_Y(\epsilon_i|\mathbf{v}_i)$.

To avoid computationally expensive solutions, full multivariate measures are not used. Instead each variable is computed independently. and one global sperad over all directions is taken to minimize corner effects. Typical choices for scalar spreads for variable r are, for example:

a) Variance $(\sigma^2_{i,r})$, $\sigma_{i,r} = \frac{1}{N_i - 1} \sum_{j \in \Omega_i} \sqrt{(x_r(j) - w_{i,r})}^2$.

b) median spread $SMED_{i,r} = \arg\min^{j \in \Omega_i}_{x_r(j)} |(x_r(j) - w_{i,r})|$.

c) fractiles $x^f_r = \arg_{x'} \int^{x'}_{w_{i,r}} f_{X_r}(x_r|i) dx_r$

Since the variance is easiest to compute and it supports our intepretation of SOM are a Gaussian mixture model,

$$f_Y(\epsilon_i|\mathbf{v}_i) = f_{\mathbf{X}}(\mathbf{x}|\mathbf{v_i}, \mathbf{A}_i) \sim \exp\left\{ -\frac{1}{2}(\mathbf{x} - \overline{\mathbf{w}}_i)^T \mathbf{A}_i^{-1} (\mathbf{x} - \overline{\mathbf{w}}_i) \right\},$$

it is most commonly used. Similarl global measures of spread are, for example

a) Variance of distance (σ^2_i), $\sigma_i = \frac{1}{M(N_i - 1)} \sum_{j \in \Omega_i} \|\mathbf{x}(j) - \mathbf{w}_i)\|$.

b) Variance of L^1 distance $l_i = \frac{1}{M(N_i - 1)} \sum_{j \in \Omega_i} |\mathbf{x}(j) - \mathbf{w}_i)|$.

c) Median distance $MMED_i = \arg\min^{j \in \Omega_i}_{\mathbf{x}(j)} |(\mathbf{x}(j) - \mathbf{w}_i)|$.

The remaining question is what should be done for those samples that are out of our rejection boundaries. One can, of course, just ignore them during the training, but as easily one can mark them as incompete observations that are then included into training by the incomplete TS-SOM training algorithm.

The algorithm, using $|w_{i,r} - x_r(j)| > \sigma_1$ as rejection bounds can then be summarized as given in Algorithm 4. Potential errors are now marked with new indicator `EI`, otherwise the algoritm is same as the incomplete data training, Algorithm 3.

Algorithm 4: TS-SOM training for data that includes both errors and missingness. This is closely related to Algorithm 3.

0. Initially : $l = 0$ (layer is root);

Steps 0.1 *and* 0.2 (as before in Algoritm 3.)

1. Initialize new layer :

Steps 1.1, 1.2 *and* 1.3 (as before in Algoritm 3.)

2. Train layer : l

Step 2.1 *Use lookup search for every node* i (as before in Algoritm 3.)

Step 2.2 *Mark if rejected or missing* ?
For every node i sample $j \in \Omega_i$ and variable r :

$$EI_r^{\text{mis}}(j) = \begin{cases} \texttt{true} & \text{if } |w_{i,r} - x_r(j)| > \theta_1 \\ \texttt{true} & \text{if } I_r^{\text{mis}}(j) = \texttt{true} \\ \texttt{false} & \text{otherwise} \end{cases}$$

Step 2.3 *For every node* i

Step 2.3.1 (*Compute node priors for every variable* r)

$$N_{i,r}^{sw,\text{obs}} = \textstyle\sum_{j \in \texttt{Robs}} s\hat{w}(j), \quad \text{where } \texttt{Robs} = \{j \mid EI_r^{\text{mis}}(j) = \texttt{false}, j \in \Omega_i\}$$

$$N_{i,r}^{sw,\text{mis}} = \textstyle\sum_{j \in \texttt{Rmis}} s\hat{w}(j), \quad \text{where } \texttt{Rmis} = \{j \mid EI_r^{\text{mis}}(j) = \texttt{true}, j \in \Omega_i\}$$

Step 2.3.2 (*Compute node means for every variable* r)

$$\overline{x}_{k,r}^{\text{obs}} = \frac{1}{N_{k,r}^{s\hat{w},\text{obs}}} \textstyle\sum_{j \in \Omega_k, EI_r^{\text{mis}}(j) = \texttt{false}} s\hat{w}(j) X_r^{\text{obs}}(j)$$

$$\overline{x}_{k,r}^{\text{mis}} = w_{i,r}^t,$$

Step 2.4 *Compute node positions for every variable* r

$$w_{i,r}^{t+1} = \frac{1}{\sum_k h_{i,k}(N_{k,r}^{s\hat{w},\text{obs}} + N_{k,r}^{s\hat{w},\text{mis}})} \textstyle\sum_k h_{i,k}(N_{k,r}^{s\hat{w},\text{obs}} \overline{x}_{k,r}^{\text{obs}} + N_{k,r}^{s\hat{w},\text{mis}} \overline{x}_{k,r}^{\text{mis}}),$$

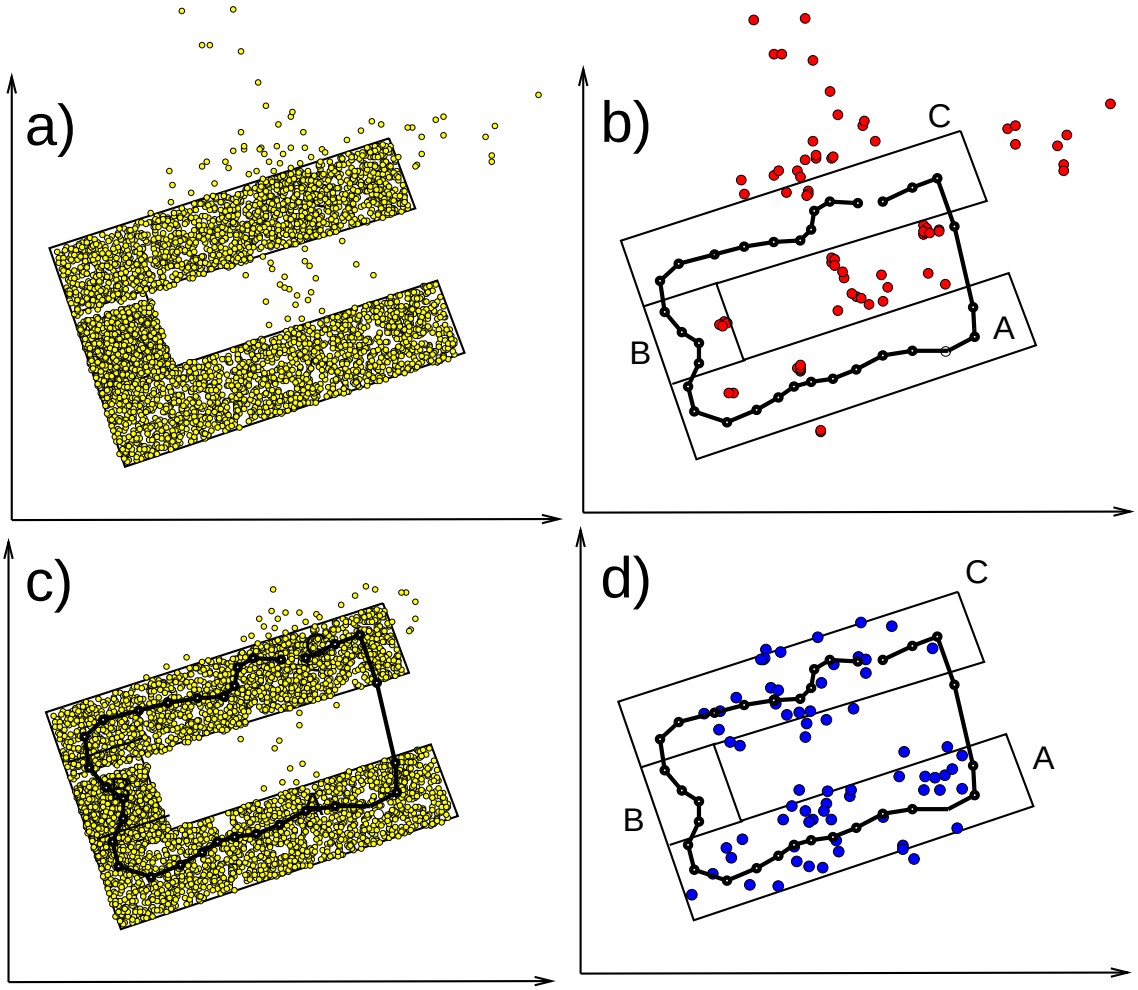
4. Repeat layer training until converged :
If $\|\mathbf{W}^{new} - \mathbf{W}^{old}\| > \delta$ GOTO 2;

5. Next layer. If more layers GOTO 1;

3 Toy example about error localization with TS-SOM

We follow the previous toy example, where some potential errors have been added from two wide spread gaussians. In the areas A,B and C there are 1600, 800 and 1600 samples, respectively. Then 2×100 additional samples are created two gaussians. This data is shown in Fig. 2 a).

Chart 2: A toy example of error detction and imputation with TS-SOM.



For model building a rejection boundary $\theta_1 = 3 \times \sigma$ was used. After training all samples out of $\theta_2 = 6 \times \sigma$ were marked as error, which detected the samples of Fig. 2 b) as errors. The cleaned data is then shown in Fig. 2 c) and finally the imputation of erronous samples is given in Fig. 2 d).

Note that the method classifies some good samples as errors, which is due the local estimation of variances. Since this seems unreliable, some smoothed or global variance estimates will be examined in future.

The model building seems to be quite robust in all our experiments, when rejection method is used. If it is not, then the SOM will easily go through the outliers that would make error detection is almost impossible.