Training Data, Prior Knowledge and Regularisation

Some Training data issues

- For the identification of a nonlinear dynamic process, collecting training data is usually a very difficult task;
- The only way to identify a very accurate ANN model is to collect training data that covers, the whole input subspace, $X \subset \boldsymbol{R}^n$ where the prediction is desired.
- For this purpose, the first step is to choose an appropriate input excitation signal $\boldsymbol{u}(t)$ for the plant or for the simulation that represents the behaviour of the system;
- In the case of dynamic models, the vector of input, $\boldsymbol{x}$, to the ANN, is related with the past time information of the input $\boldsymbol{u}(t)$ and the output $y(t)$ of the system, for example:

$$\boldsymbol{x} = \left[\begin{array}{cccccc} y(t-\tau_1^y) & y(t-\tau_2^y) & \cdots & \boldsymbol{u}(t) & \boldsymbol{u}(t-\tau_1^u) & \cdots \end{array}\right]^T,$$

where τ_1^y,τ_2^y and τ_1^u are appropriate delays. The output is usually the one-step ahead prediction $y(t)$. In many neural-fuzzy models or other neural networks models, a first-order model is used:

$$\boldsymbol{x} = \left[\begin{array}{cc} y(t-1) & \boldsymbol{u}(t-1) \end{array}\right]^T.$$

- To cover the entire sub-space $X \subset \boldsymbol{R}^n$, it is necessary to drive the output $y(t)$ in a range of the input space using just the input $\boldsymbol{u}(t)$.
- For estimating parameters of difference equations that represent linear time-invariant systems, there are some well known theoretical results about how to design input signals
- However, only a few results are known for the more difficult problem of selecting input signals for the identification of non-linear systems.
- Identification is usually carried out in open loop. However, situations where some practical restrictions must be placed on the identification (e.g. bounds on the outputs of the system or the system can be unstable in open loop), the identification experiment must be carried out in closed loop conditions.
- Closed-loop operation allows control of the output $y(t)$ of the system but there is no direct control of the input $\boldsymbol{u}(t)$.

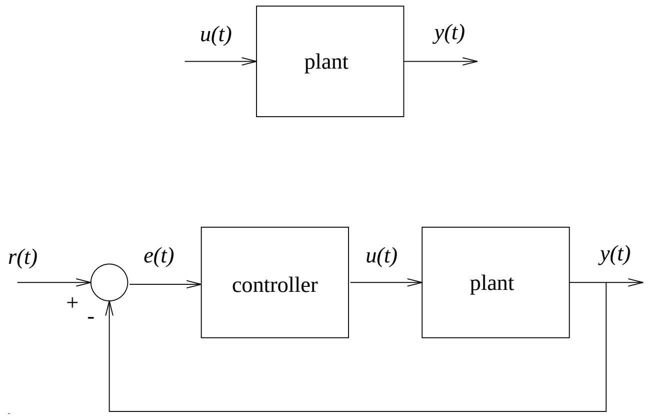

Figura 1: Block diagrams of open loop and closed loop configurations.

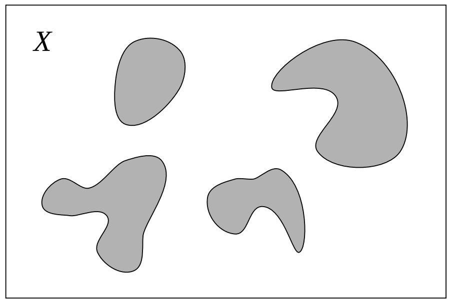

Figura 2: Incompleteness of training data. Training data are usually not enough to cover the sub-space X of the inputs of the model. Data are represented by grey areas.

Alguns Exemplos

Suppose that we are trying to approximate the following unidimensional function:

$$g(x) = 4.26(\exp(-x) - 4\exp(-2x) + 3\exp(-3x)) + 2.0. \tag{1}$$

The training data is created by sampling the following function:

$$y = g(x) + \eta, \tag{2}$$

where η is an additive Gaussian noise on the output with standard deviation of $\sigma_r = 0.1$.

Sixty uniformly distributed training data points (x_i, y_i) are generated in the interval $X = [0.0, 5.0]$.

The chosen neuro-fuzzy model $h(x)$ has 7 uniformly distributed triangular basis functions.

A batch least squares algorithm is used to estimate the parameters.

Exemplo 1

Initially, in this example, the whole training data set is considered and the following parameters are estimated:

$$\hat{\boldsymbol{W}}_{d_1} = \begin{bmatrix} 1.25 & 1.58 & \boxed{2.37} & 2.21 & 2.16 & 2.03 & 2.01 \end{bmatrix}.$$

Now the same experiment is conducted without some of the training data points. In this case, the parameters estimated are:

$$\hat{\boldsymbol{W}}_{d_2} = \begin{bmatrix} 1.35 & 1.35 & \boxed{5.17} & 2.27 & 2.19 & 2.04 & 2.00 \end{bmatrix}.$$

Example 2 : redundant information missing

If complete training data is used the following optimal parameters are estimated:

$$\hat{\boldsymbol{W}}_{d_3} = \left[\ 1.24 \ \ 1.55 \ \ 2.40 \ \ \boxed{2.22} \ \ \boxed{2.13} \ \ 2.05 \ \ 2.02\ \right].$$

If incomplete training data is used the following parameters are estimated:

$$\hat{\boldsymbol{W}}_{d_4} = \left[\ 1.24 \ \ 1.55 \ \ 2.40 \ \ \boxed{2.21} \ \ \boxed{2.17} \ \ 2.05 \ \ 2.02\ \right].$$

TD, Prior knowledge, and Regularisation

Example 3: the influence of the noise

For the complete training data the following optimal parameters are estimated:

$$\hat{\boldsymbol{W}}_{d_5} = \left[\ 1.22 \ \ 1.54 \ \ 2.47 \ \ 2.17 \ \ \boxed{2.24} \ \ \boxed{2.04} \ \ 2.03\ \right].$$

If incomplete training data is used, the following parameters are estimated:

$$\hat{\boldsymbol{W}}_{d_6} = \left[\ 1.22 \ \ 1.54 \ \ 2.48 \ \ 2.13 \ \ \boxed{2.55} \ \ \boxed{1.83} \ \ 2.09\ \right].$$

Regularisation

- Regularisation is a popular method used to constrain weight optimisation such that sensible models are produced;
- This is performed by penalising the performance criteria (Empirical Risk Function) by adding some prior distribution on the weights;
- The empirical risk function:

$$\mathcal{I}_{\mathrm{emp}}[h] = \frac{1}{N}\sum_{i=1}^{N}(y_i - h(\boldsymbol{x}_i))^2. \tag{3}$$

  is penalised by E^P, a penalising error, giving the modified risk function:

$$\mathcal{I}_{\mathrm{emp}}[h] = MSE + \lambda E^P. \tag{4}$$

- This penalising error E^P may take many different forms but in general is represented by a quadratic weight function, i.e.:

$$E^P = \boldsymbol{w}^T\boldsymbol{K}\boldsymbol{w}. \tag{5}$$

Some types of regularisation

Zero order

- This the commonest regularisation, known as Ridge Regression, Weight Decay, and also as the Levenberg-Marquardt optimisation method;
- The penalty term E^P represents the expected size of the output:

$$E^P = \int_{\mathcal{D}} |\hat{h}(\boldsymbol{x})|p(\boldsymbol{x})d\boldsymbol{x} \tag{6}$$

  where $p(\boldsymbol{x})$ is the probability density function for the input $\boldsymbol{x}$;
- This can be approximated by a quadratic penalty function:

$$E^P = \boldsymbol{w}^T\boldsymbol{w}. \tag{7}$$

Second order

- This form of regularisation makes the assumption that the function is smooth, and hences the expected curvature of the output is used to penalise the cost function:

$$E^P = \int_{\mathcal{D}} |\frac{d^2\hat{h}(\boldsymbol{x})}{\boldsymbol{dx^2}}| p(\boldsymbol{x}) d\boldsymbol{x} \tag{8}$$

- The expected curvature of the output can be approximated by a quadratic function of the weights, i.e.:

$$E^P = \boldsymbol{w}^T \boldsymbol{W} \boldsymbol{w} \tag{9}$$

The weight decay

In this case, the risk function becomes:

$$\mathcal{I}[h] = \sum_{i=1}^{N} (y_i - h(\boldsymbol{W}, \boldsymbol{x}_i))^2 + \lambda \boldsymbol{W}^T \boldsymbol{W}. \tag{10}$$

The parameters can be estimated by the following batch algorithm

$$\hat{\boldsymbol{W}} = (\overline{\boldsymbol{\Phi}}^T \overline{\boldsymbol{\Phi}} + \lambda I)^{-1} \overline{\boldsymbol{\Phi}}^T Y. \tag{11}$$

The regularisation parameter λ enforces a uniform smoothness over the input space.

For neural networks based on local basis functions like radial basis functions networks and neuro-fuzzy models, it is possible to explore a more flexible approach developed by Orr. The smoothness can be specified locally by associating a local regularisation parameter λ_q with each basis function $\phi_q(\boldsymbol{x})$. In this case, the risk function becomes:

$$\mathcal{I}[h] = \sum_{i=1}^{N}(y_i - h(\boldsymbol{W}, \boldsymbol{x}_i))^2 + \boldsymbol{W}^T \boldsymbol{\Lambda} \boldsymbol{W}. \tag{12}$$

And parameters can be estimated by:

$$\hat{\boldsymbol{W}} = (\overline{\boldsymbol{\Phi}}^T \overline{\boldsymbol{\Phi}} + \boldsymbol{\Lambda})^{-1} \overline{\boldsymbol{\Phi}}^T Y, \tag{13}$$

where $\boldsymbol{\Lambda}$ is a $q \times q$ diagonal matrix with the main diagonal represented by the following vector:

$$\boldsymbol{\lambda} = \left[\begin{array}{ccccc} \lambda_1 & \ldots & \lambda_q & \ldots & \lambda_Q \end{array} \right].$$

Some examples

Global regularisation 30 equally spaced basis functions with $\sigma_w = 0.07$ are used with global regularisation parameter $\lambda = 1.0$. Figure illustrates the results. The global regularisation approach adopted here, plus an inadequate choice of basis functions, are not able to make the solution smooth.

Local regularisation In this example, illustrated by the Figure, the width of the Gaussian basis functions is changed to $\sigma_w = 0.2$. The vector of local regularisation parameters is set to:

$$\lambda = \begin{array}{lllllllllll} [\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & \ldots \\ \ \ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & \ldots \\ \ \ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 &]. \end{array}$$

It may be observed from Figure that the solution is smooth. This smoothness is achieved by a suitable basis function selection and appropriate local regularisation parameters.

A localized leakage algorithm

The term *leakage* in adaptive control and adaptive signal processing refers to a *pull term* towards a given vector of parameters:

$$\boldsymbol{W}^{\#} = \left[\begin{array}{ccccc} w_1^{\#} & \dots & w_q^{\#} & \dots & w_Q^{\#} \end{array} \right]. \tag{14}$$

The idea is to penalize vectors of parameters that are distant from $\boldsymbol{W}^{\#}$. This can be achieved by finding a vector of parameters, $\hat{\boldsymbol{W}}$, that minimizes the following risk function:

$$\mathcal{I}[h] = \sum_{i=1}^{N} (y_i - h(\boldsymbol{W}, \boldsymbol{x}_i))^2 + \lambda(\hat{\boldsymbol{W}} - \boldsymbol{W}^{\#})^T(\hat{\boldsymbol{W}} - \boldsymbol{W}^{\#}), \tag{15}$$

where λ is the leakage parameter. A large λ forces the solution $\hat{\boldsymbol{W}}$ to be close to $\boldsymbol{W}^{\#}$.

The risk function is similar to the risk function adopted for the weight decay. In fact, the leakage algorithm provides regularisation of the estimation around $\boldsymbol{W}^{\#}$. Usually, regularisation is used to reduce the variance error which can arise due to noise in overparameterized models, however, in this section it is proposed as a way of avoiding problems with incomplete training data.

TD, Prior knowledge, and Regularisation

The parameters can be estimated by the following batch algorithm:

$$\hat{\boldsymbol{W}} = (\overline{\boldsymbol{\Phi}}^T\overline{\boldsymbol{\Phi}} + \lambda I)^{-1}(\overline{\boldsymbol{\Phi}}^T Y + \lambda \boldsymbol{W}^{\#}). \tag{16}$$

The use of a single parameter λ is not always flexible enough to allow the use of prior knowledge encoded in $\boldsymbol{W}^{\#}$ as it pulls the solution towards $\boldsymbol{W}^{\#}$ uniformly. The general case would be to have a different leakage parameter λ_q for each parameter $w_q^{\#}$. If there is more confidence in the correctness of a parameter, $w_q^{\#}$, a larger λ_q can be chosen to protect it from both noisy data and a lack of training data. Following the idea of the local regularisation approach, a similar modification in the global leakage algorithm is introduced to transform it into a local leakage algorithm:

$$\mathcal{I}[h] = \sum_{i=1}^{N} (y_i - h(\boldsymbol{W}, \boldsymbol{x}_i))^2 + (\hat{\boldsymbol{W}} - \boldsymbol{W}^{\#})^T \boldsymbol{\Lambda} (\hat{\boldsymbol{W}} - \boldsymbol{W}^{\#}). \tag{17}$$

The parameters can then be estimated by the following batch algorithm:

$$\hat{\boldsymbol{W}} = (\overline{\boldsymbol{\Phi}}^T\overline{\boldsymbol{\Phi}} + \boldsymbol{\Lambda})^{-1}(\overline{\boldsymbol{\Phi}}^T Y + \boldsymbol{\Lambda} \boldsymbol{W}^{\#}), \tag{18}$$

where $\boldsymbol{\Lambda}$ is a $q \times q$ diagonal matrix with the main diagonal represented by the following vector:

$$\boldsymbol{\lambda} = \left[\begin{array}{ccccc} \lambda_1 & \dots & \lambda_q & \dots & \lambda_Q \end{array} \right].$$