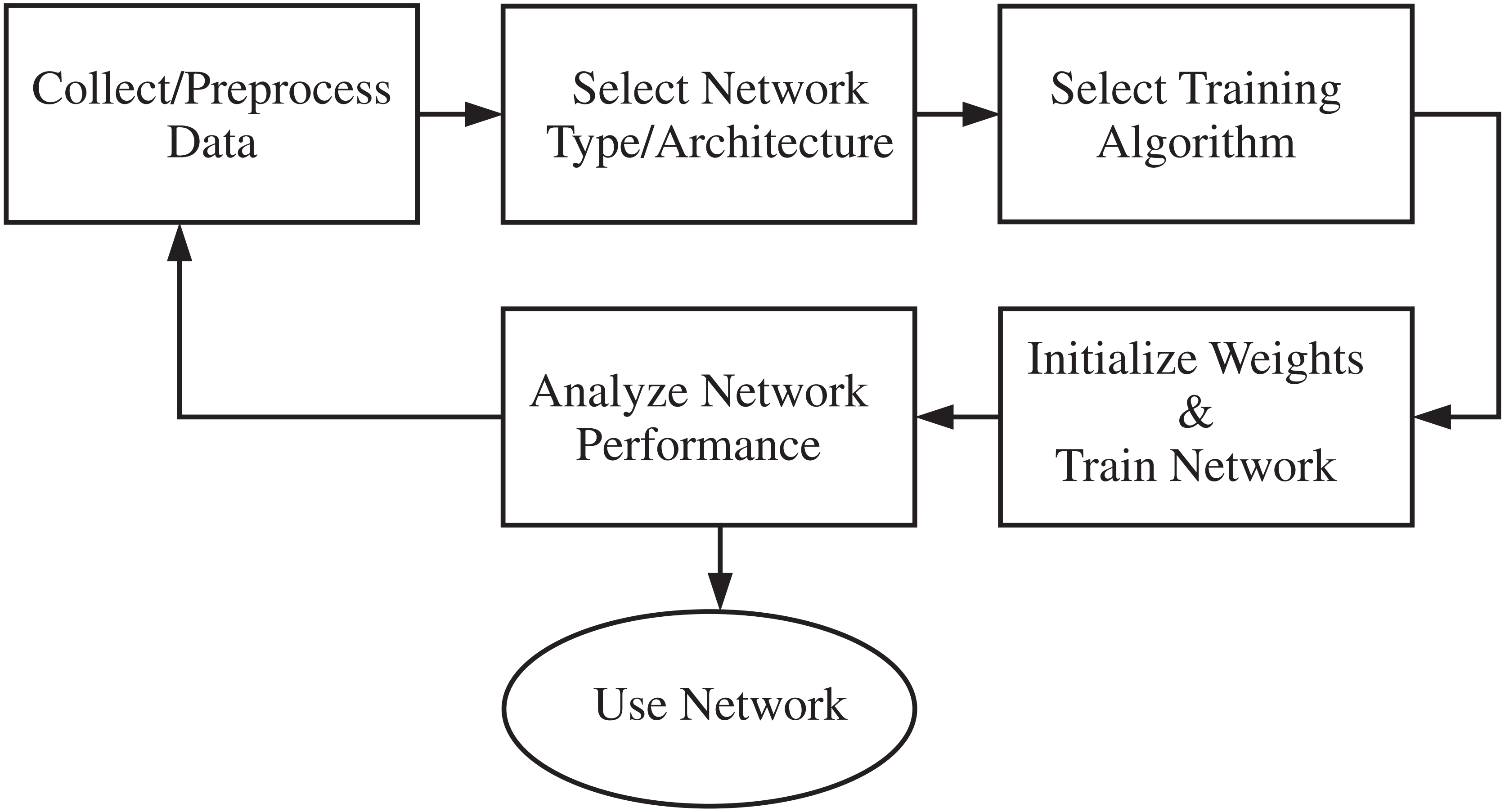
22

Practical Training Issues

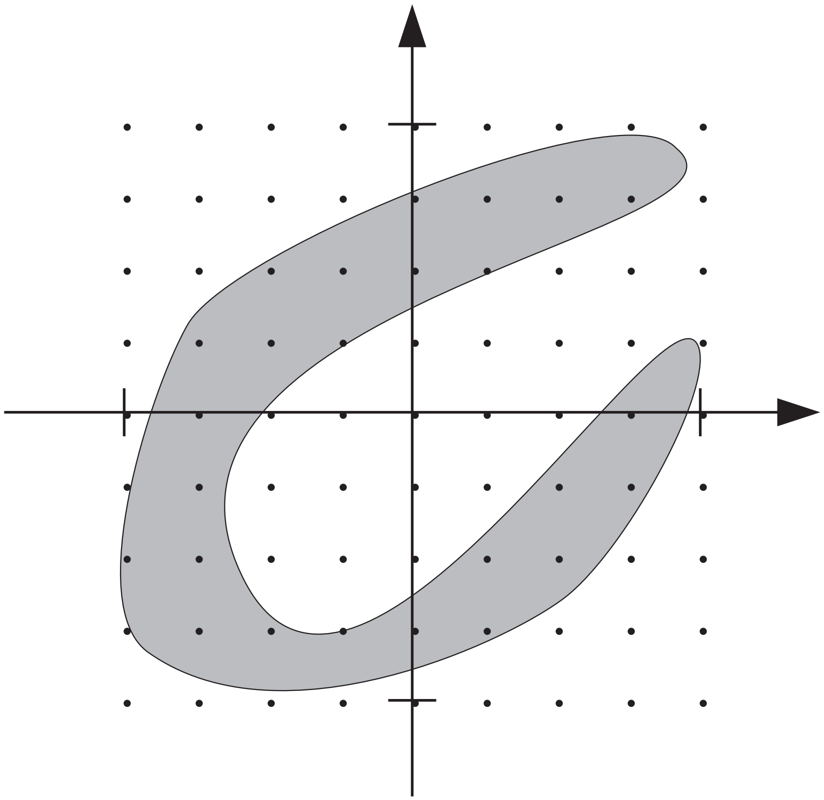
Network Training Steps


Collect/Preprocess Data
Select Network Type/Architecture
Select Training Algorithm
Initialize Weights & Train Network
Analyze Network Performance
Use Network

Selection of Data

- Data must adequately cover the relevant regions of the input space (to avoid extrapolation).
- Divide the data into training, validation and testing subsets (70%, 15%, 15%).
- Each of the subsets must cover the same parts of the input space.
- The amount of data required depends on the complexity of the function being approximated (or the complexity of the decision boundary).
- Post-training analysis may be needed to determine the adequacy of the data.

Determination of Input Range



Data Preprocessing

- Normalize inputs/targets to the range [-1,1].

$$\mathbf{p}^{n} = 2(\mathbf{p} - \mathbf{p}^{min})./(\mathbf{p}^{max} - \mathbf{p}^{min}) - 1$$

- Normalize inputs/targets to zero mean and unity variance.

$$\mathbf{p}^{n} = (\mathbf{p} - \mathbf{p}^{mean})./\mathbf{p}^{std}$$

- Nonlinear transformations.

$$\mathbf{p}^{t} = 1/\mathbf{p} \qquad\qquad \mathbf{p}^{t} = \exp(\mathbf{p})$$

- Feature extraction (dimensionality reduction).
  - Principal components

Coding Targets (Pattern Classification)

- There are three common ways to code targets. Assume that we have *N* classes.
- 1) You can have a scalar target that takes on *N* possible values (e.g., 1, 2, …, *N*)
- 2) You can code the target in binary code. This requires *P* output neurons, where 2^P is greater than or equal to *N*.
- 3) You can have N neurons in the output layer. The targets will be vectors whose elements are all equal to zero, except for the neuron that corresponds to the correct class.
- Method 3) generally produces the best results.

Importance of Transfer Function

- When coding the targets, we need to consider the output layer transfer function.
- For pattern recognition problems, we would typically use log-sigmoid or tangent-sigmoid.
- If we use the tangent-sigmoid in the last layer, which is more common, then we might consider assigning target values to -1 or 1.
- This tends to cause training difficulties (saturation of the sigmoid function).
- It is better to assign target values at the point where the second derivative of the sigmoid function is maximum. This occurs when the net input is -1 and 1, which corresponds to output values of -0.76 and +0.76.

22

Softmax Transfer Function

- If the network outputs should correspond to probabilities of belonging to a certain class, the softmax transfer function can be used.

$$a_i = f(n_i) = \frac{\exp(n_i)}{\sum_{j=1}^{S} \exp(n_j)}$$

Missing Data

- Replace the missing values in the input vector with the average value for that element of the input. Add an additional variable to the input vector as a flag to indicate missing data.
- For missing elements of the target vectors, do not include them in the calculation of squared error.

Problem Types

- Fitting (nonlinear regression). Map between a set of inputs and a corresponding set of targets. (e.g., estimate home prices from tax rate, pupil/teacher ratio, etc.; estimate emission levels from fuel consumption and speed; predict body fat level from body measurements.)
- Pattern recognition (classification). Classify inputs into a set of target categories. (e.g., recognize the vineyard from a chemical analysis of the wine; classify a tumor as benign or malignant, from uniformity of cell size, clump thickness and mitosis.)
- Clustering (segmentation) Group data by similarity. (e.g., group customers according to buying patterns, group genes with related expression patterns.)
- Prediction (time series analysis, system identification, filtering or dynamic modeling). Predict the future value of some time series. (e.g., predict the future value of some stock; predict the future value of the concentration of some chemical; predict outages on the electric grid.)

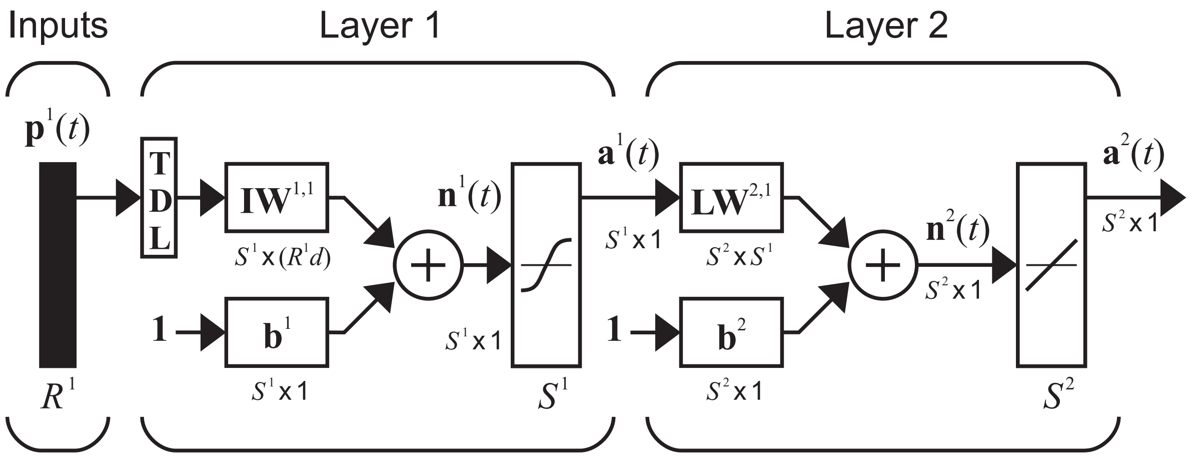
Choice of Network Architecture

- Fitting
  - Multilayer networks with sigmoid hidden layers and linear output layers.
  - Radial basis networks
- Pattern Recognition
  - Multilayer networks with sigmoid hidden layers and sigmoid output layers.
  - Radial basis networks.
- Clustering
  - Self-organizing feature map
- Prediction
  - Focused time-delay neural network
  - NARX network

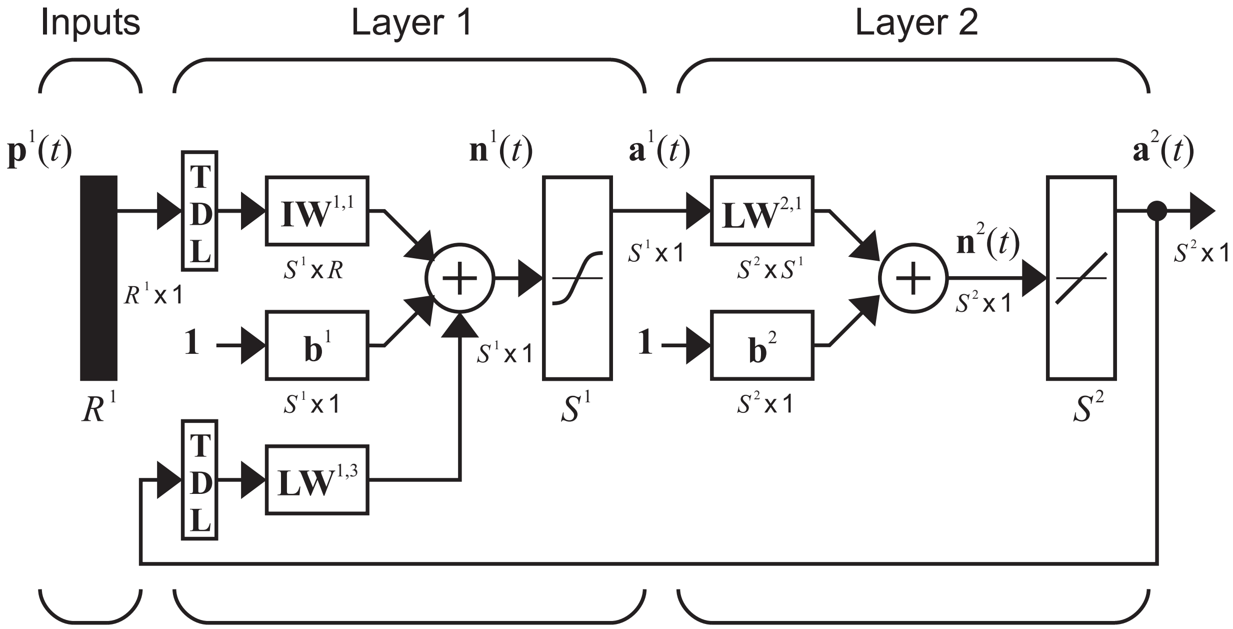
Prediction Networks

Focused

Time Delay



NARX



Architecture Specifics

- Number of layers/neurons
  - For multilayer network, start with two layers. Increase number of layers if result is not satisfactory.
  - Use a reasonably large number of neurons in the hidden layer (20). Use early stopping or Bayesian regularization to prevent overfitting.
  - Number of neurons in output layer = number of targets. You can use multiple networks instead of multiple outputs.
- Input selection
  - Sensitivity analysis (see later slide)
  - Bayesian regularization with separate α for each column of the input weight matrix.

Weight Initialization

For Multilayer Networks

- Random weights. Uniformly distributed between -0.5 and 0.5, if the inputs are normalized to fall between -1 and 1.
- Random direction for weights, with magnitude set to

$$\left\| {}_{i}\mathbf{w} \right\| = 0.7\left(S^{1}\right)^{1/R}$$

and biases randomly distributed between

$$-\left\| {}_{i}\mathbf{w} \right\| \quad \text{and} \quad \left\| {}_{i}\mathbf{w} \right\|.$$

Weight Initialization

- For Competitive Networks
- Small random numbers
- Randomly selected input vectors
- Principal components of the input vectors

Choice of Training Algorithm

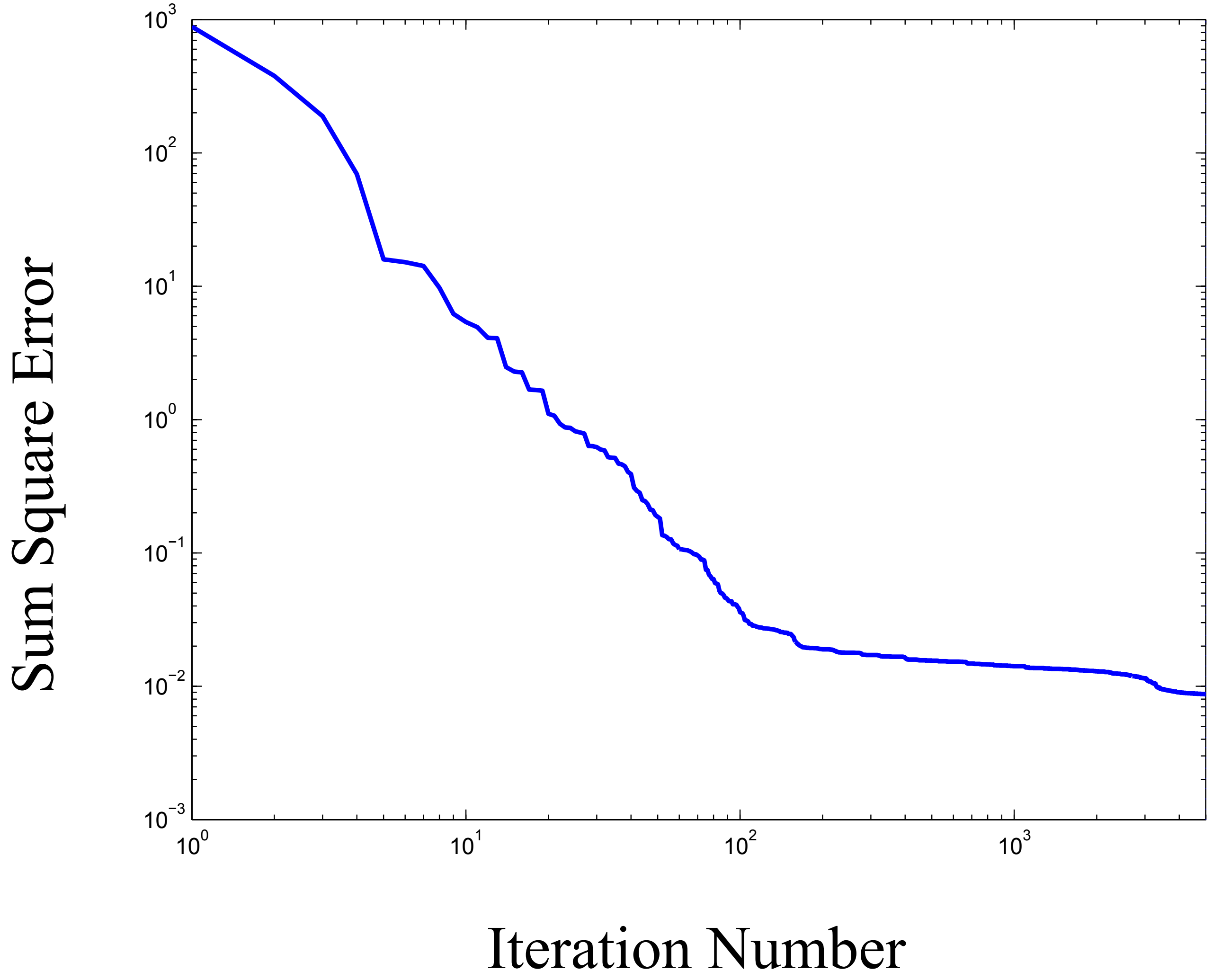
- For medium sized networks (several hundred weights) used for fitting or prediction problems, use the Levenberg-Marquardt algorithm (trainlm).
- For large networks (thousands of weights) used for fitting or prediction problems, or networks used for pattern recognition problems, conjugate gradient algorithms, such as the scaled conjugate gradient algorithm (trainscg) are generally faster.
- Of the sequential algorithms, the extended Kalman filter algorithm are generally fastest.

22

Stopping Criteria

- Norm of the gradient (of the mean squared error) less than a pre-specified amount (for example, 10^{-6}).
- Early stopping because the validation error increases.
- Maximum number of iterations reached.
- Mean square error drops below a specified threshold (not generally a useful method).
- Mean square error curve (on a log-log scale) becomes flat for some time (user stop).

Typical Training Curve



Competitive Network Stopping Criteria

- Stop when a specified number of iterations has been reached.
- Learning rate and neighborhood size (SOM) are decreased during training, so that they reach their smallest values when the maximum number of iterations have been reached.
- Post-training analysis is used to determine if retraining is required.

Choice of Performance Function

Mean Square Error

$$F(\mathbf{x}) = \frac{1}{QS^M}\sum_{q=1}^{Q}(\mathbf{t}_q - \mathbf{a}_q)^T(\mathbf{t}_q - \mathbf{a}_q)$$

$$F(\mathbf{x}) = \frac{1}{QS^M}\sum_{q=1}^{Q}\sum_{i=1}^{S^M}(t_{i,q} - a_{i,q})^2$$

Minkowski error

$$F(\mathbf{x}) = \frac{1}{QS^M}\sum_{q=1}^{Q}\sum_{i=1}^{S^M}\left|t_{i,q} - a_{i,q}\right|^K$$

Cross-Entropy

$$F(\mathbf{x}) = -\sum_{q=1}^{Q}\sum_{i=1}^{S^M} t_{i,q}\ln\frac{a_{i,q}}{t_{i,q}}$$

Committees of Networks

- Restart training at 5 to 10 different initial conditions to be sure to reach a global minimum.
- You can also train several different networks with different initial conditions and different divisions of the data into training and validation sets. This produces a committee of networks.
- Take the average of the committee outputs to produce a more accurate fit than any of the individual networks.
- For pattern recognition problems, you can take a vote of the committee of networks to produce a more accurate classification.

Post-Training Analysis

- Fitting
- Pattern Recognition
- Clustering
- Prediction

Fitting

Regression Analysis (Outputs vs Targets)

$$a_q = m t_q + c + \varepsilon_q$$

$$\hat{m} = \frac{\sum_{q=1}^{Q} (t_q - \bar{t})(a_q - \bar{a})}{\sum_{q=1}^{Q} (t_q - \bar{t})^2}$$

$$\bar{t} = \sum_{q=1}^{Q} t_q$$

$$\hat{c} = \bar{a}\hat{m} - \bar{t}$$

$$\bar{a} = \sum_{q=1}^{Q} a_q$$

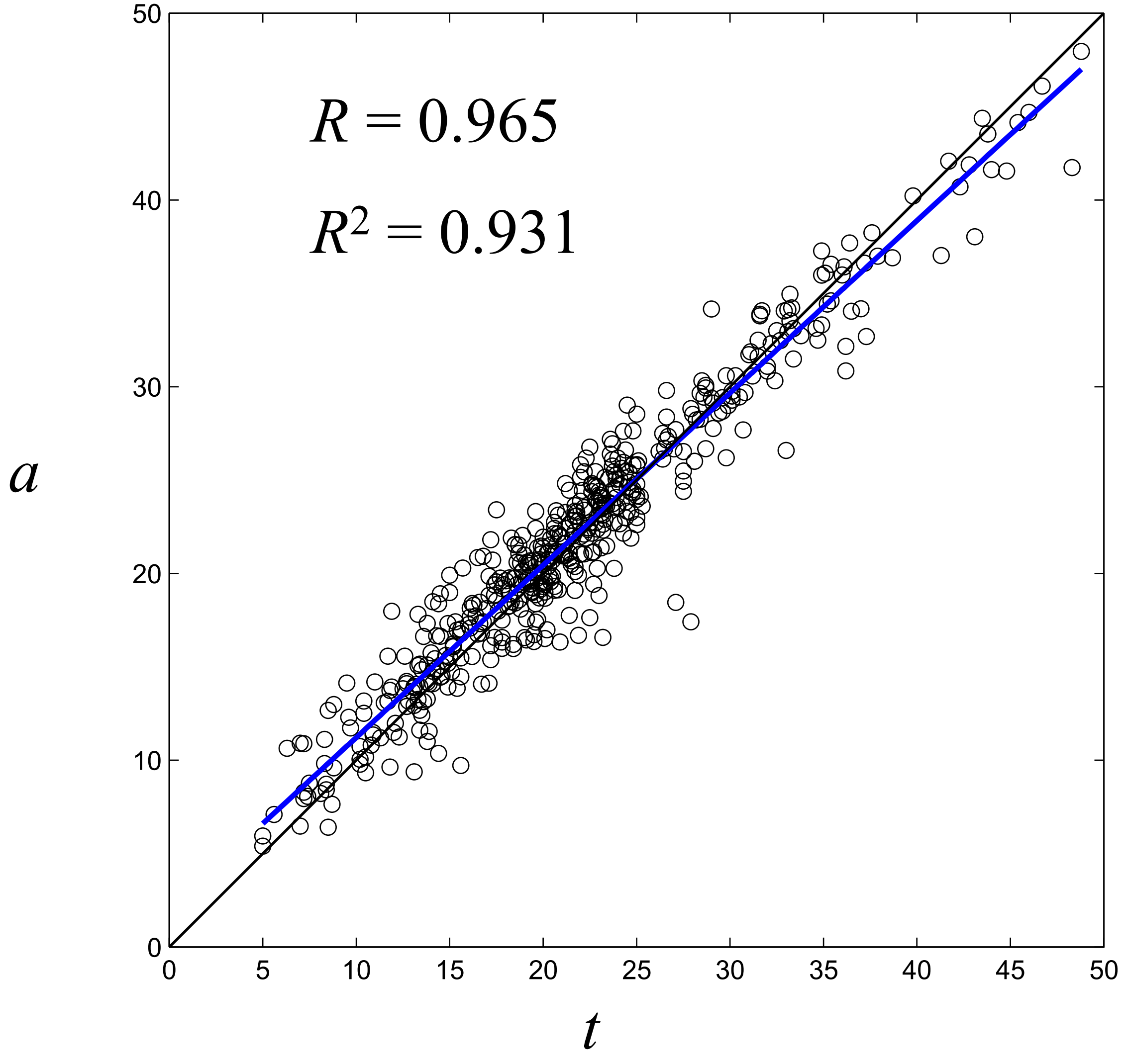
R Value (-1<*R*<1)

$$R = \frac{\sum_{q=1}^{Q} (t_q - \bar{t})(a_q - \bar{a})}{(Q-1) s_t s_a}$$

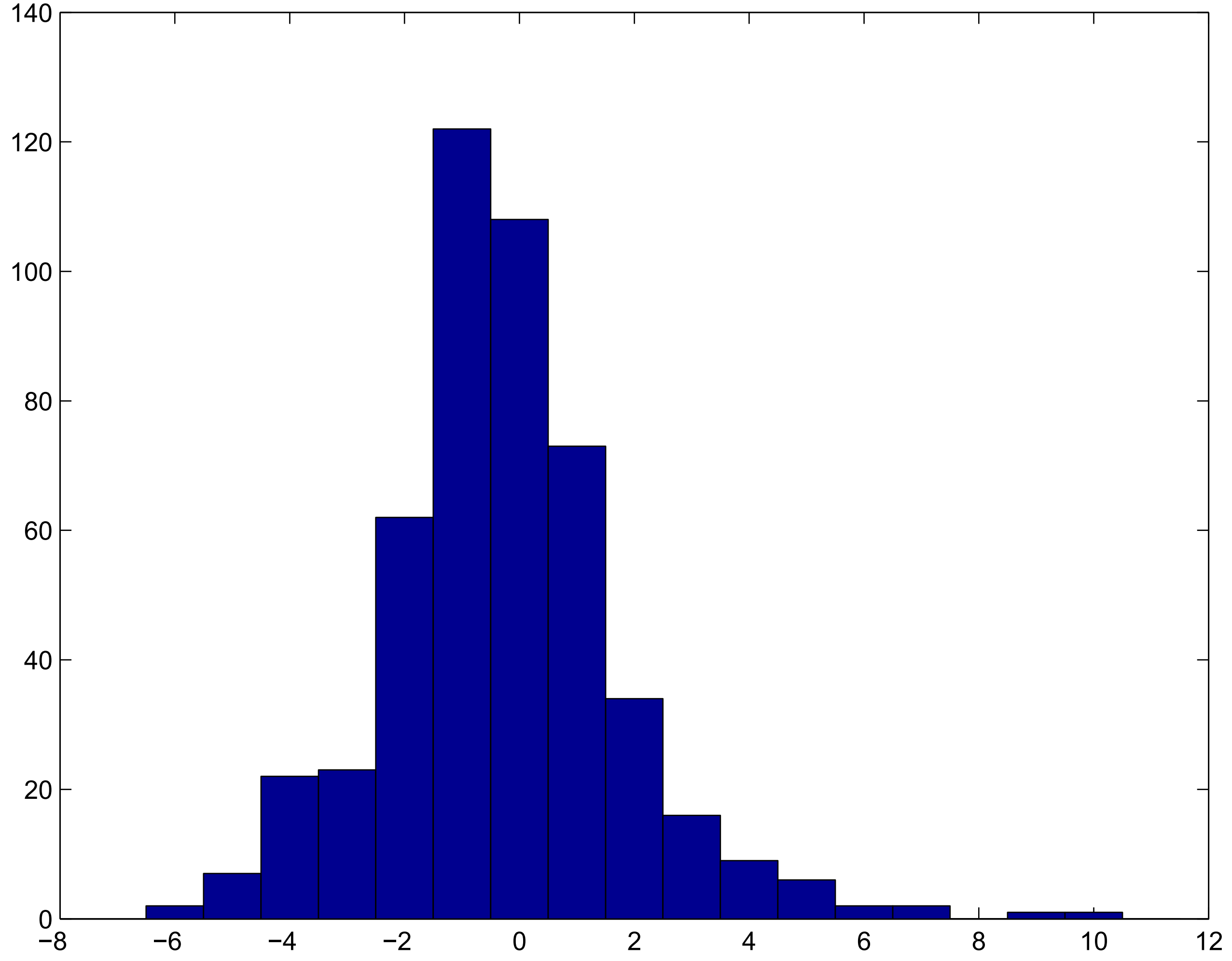
$$s_t = \sqrt{\frac{1}{Q-1} \sum_{q=1}^{Q} (t_q - \bar{t})}$$

$$s_a = \sqrt{\frac{1}{Q-1} \sum_{q=1}^{Q} (a_q - \bar{a})}$$

Sample Regression Plot



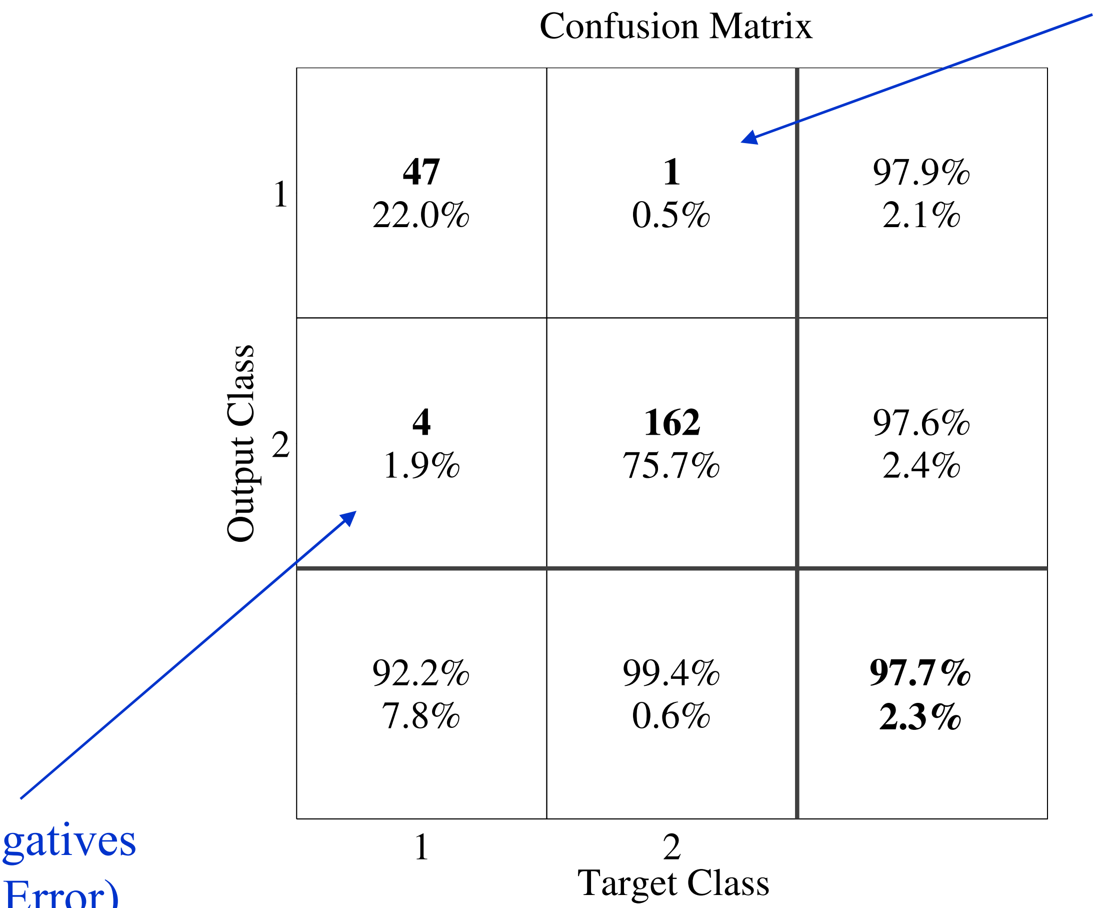
Error Histogram



$e = t - a$

22

Pattern Recognition



Confusion Matrix

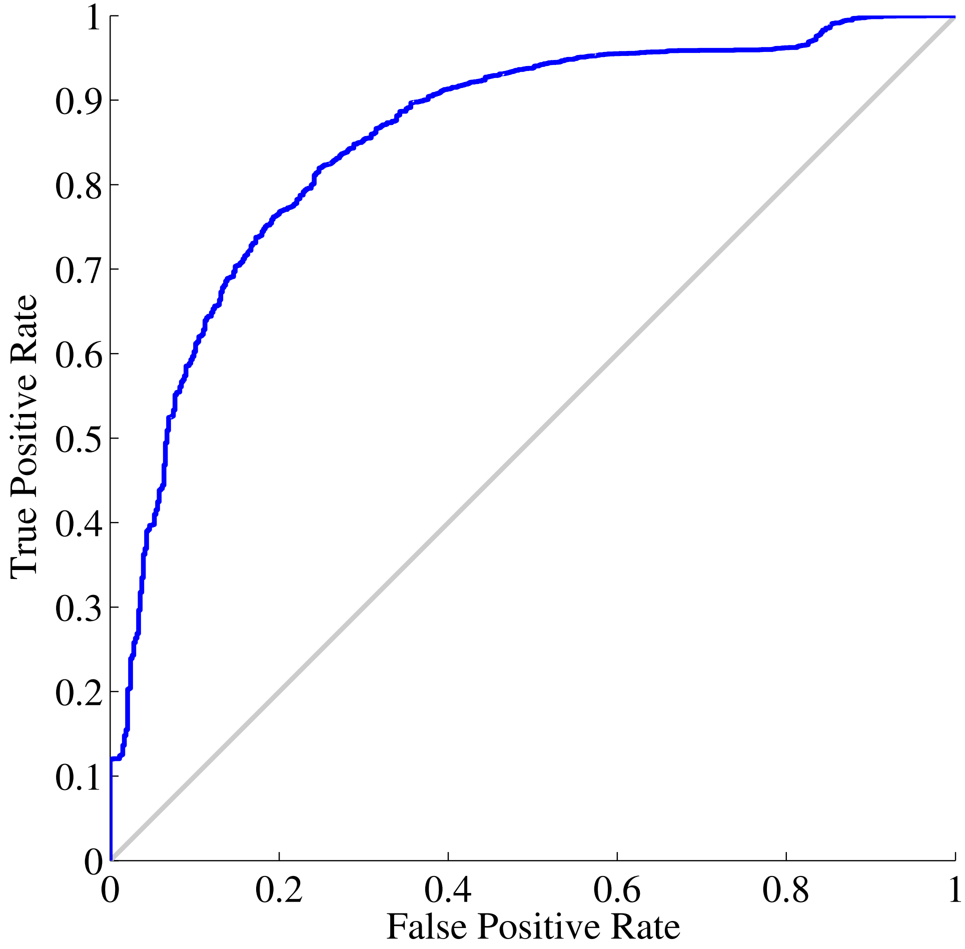
Output Class \ Target Class	1	2	
1	**47** 22.0%	**1** 0.5%	97.9% 2.1%
2	**4** 1.9%	**162** 75.7%	97.6% 2.4%
	92.2% 7.8%	99.4% 0.6%	**97.7%** **2.3%**

False Positives
(Type I Error)

False Negatives
(Type II Error)

Pattern Recognition

Receiver Operating Characteristic (ROC) Curve



Clustering (SOM)

- Quantization Error. The average distance between each input vector and the closest prototype vector.
- Topographic Error. The proportion of all input vectors for which the closest prototype vector and the next closest prototype vector are not neighbors in the feature map topology.
- Distortion

$$E_d = \sum_{q=1}^{Q} \sum_{i=1}^{S} h_{ic_q} \left\| {}_i\mathbf{w} - \mathbf{p}_q \right\|^2$$

Prototype closest to the input vector.

$$c_q = \arg\min_j \{ \left\| {}_j\mathbf{w} - \mathbf{p}_q \right\| \}$$

$h_{i,j}$ = neighborhood function

$$h_{ij} = \exp\left(\frac{-\left\| {}_i\mathbf{w} - {}_j\mathbf{w} \right\|^2}{2d^2} \right)$$

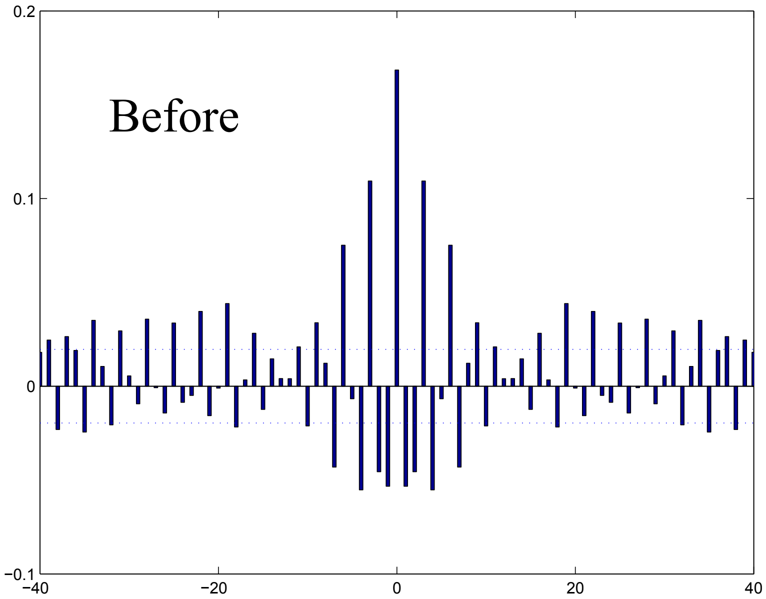
Prediction

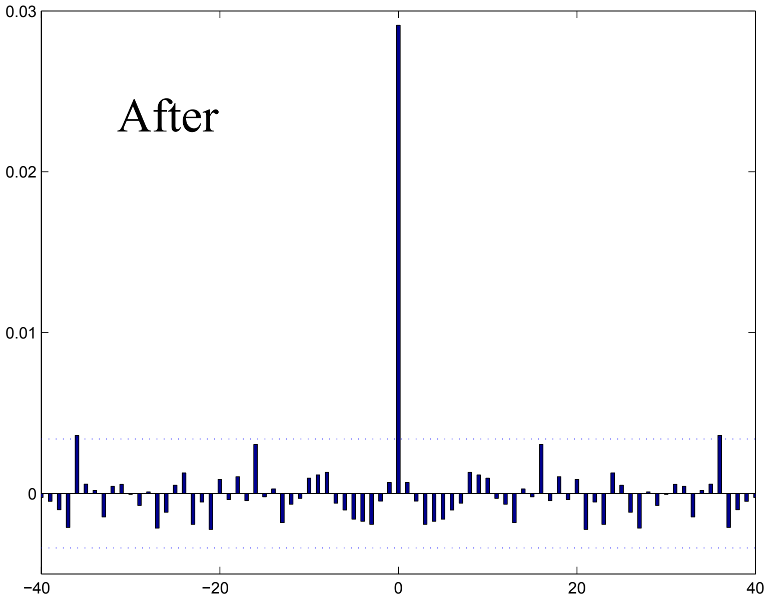
Autocorrelation Function of Prediction Errors.

$$R_e(\tau) = \frac{1}{Q-\tau}\sum_{t=1}^{Q-\tau} e(t)e(t+\tau)$$

Confidence Intervals.

$$-\frac{2R_e(0)}{\sqrt{Q}} < R_e(\tau) < \frac{2R_e(0)}{\sqrt{Q}}$$





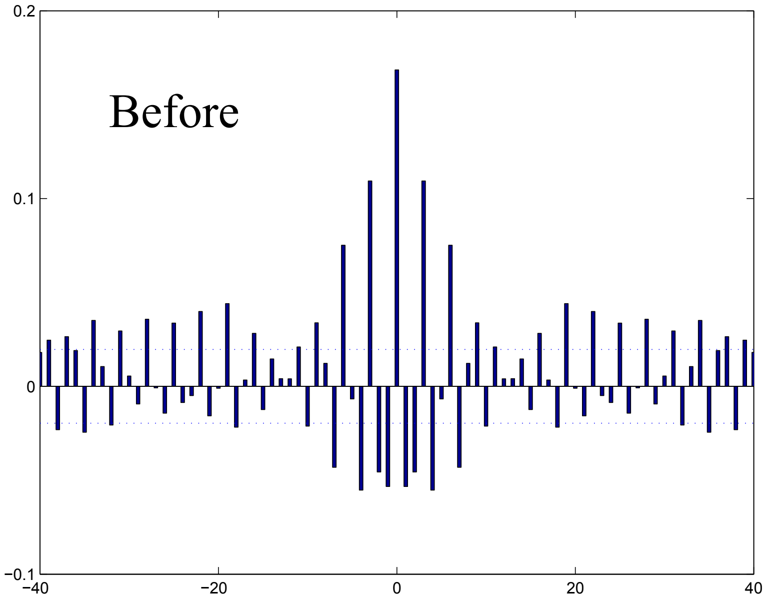
Prediction

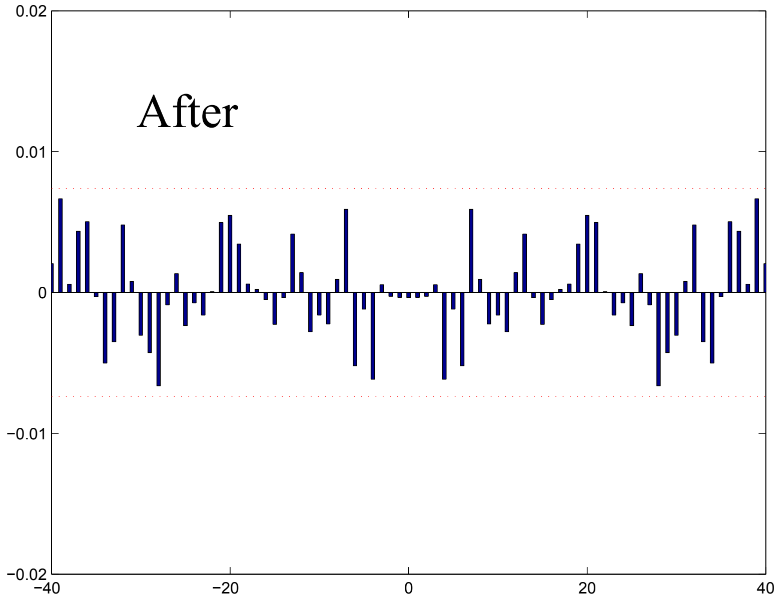
Cross-correlation Between Prediction Errors and Input.

$$R_{pe}(\tau) = \frac{1}{Q-\tau}\sum_{t=1}^{Q-\tau} p(t)e(t+\tau)$$

Confidence Intervals.

$$-\frac{2\sqrt{R_e(0)}\sqrt{R_p(0)}}{\sqrt{Q}} < R_{pe}(\tau) < \frac{2\sqrt{R_e(0)}\sqrt{R_p(0)}}{\sqrt{Q}}$$





Overfitting and Extrapolation

If, after a network has been trained, the test set performance is not adequate, then there are usually four possible causes:

- the network has reached a local minimum,
- the network does not have enough neurons to fit the data,
- the network is overfitting, or
- the network is extrapolating.

Diagnosing Problems

- The local minimum problem can almost always be overcome by retraining the network with five to ten random sets of initial weights.
- If the validation error is much larger than the training error, then overfitting has probably occurred.
- If the validation, training and test errors are all similar in size, but the errors are too large, then the network is not powerful enough to fit the data. Add neurons.
- If the validation and training errors are similar in size, but the test errors are significantly larger, then the network may be extrapolating.
- If training, validation and test errors are similar, and the errors are small enough, then we can put the multilayer network to use.

Novelty Detection

- To detect extrapolation, train a companion competitive network to cluster the input vectors of the training set.
- When an input is applied to the multilayer network, the same input is applied to the companion competitive network.
- When the distance of the input vector to the nearest prototype vector of the competitive network is larger than the distance from the prototype to the most distant member of its cluster of inputs in the training set, we can suspect extrapolation.

Sensitivity Analysis

Check for important inputs.

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m}$$

$$\frac{\partial \hat{F}}{\partial p_j} = \sum_{i=1}^{S^1} \frac{\partial \hat{F}}{\partial n_i^1} \times \frac{\partial n_i^1}{\partial p_j} = \sum_{i=1}^{S^1} s_i^1 \times \frac{\partial n_i^1}{\partial p_j}$$

$$n_i^1 = \sum_{j=1}^{R} w_{i,j}^1 p_j + b_i^1$$

$$\frac{\partial \hat{F}}{\partial p_j} = \sum_{i=1}^{S^1} \frac{\partial \hat{F}}{\partial n_i^1} \times \frac{\partial n_i^1}{\partial p_j} = \sum_{i=1}^{S^1} s_i^1 \times w_{i,j}^1$$

$$\boxed{\frac{\partial \hat{F}}{\partial \mathbf{p}} = (\mathbf{W}^1)^T \mathbf{s}^1}$$