Data Mining Modeling Techniques

Dr. Zhu Fangming
Institute of Systems Science,
National University of Singapore.
E-mail: isszfm@nus.edu.sg

Objectives

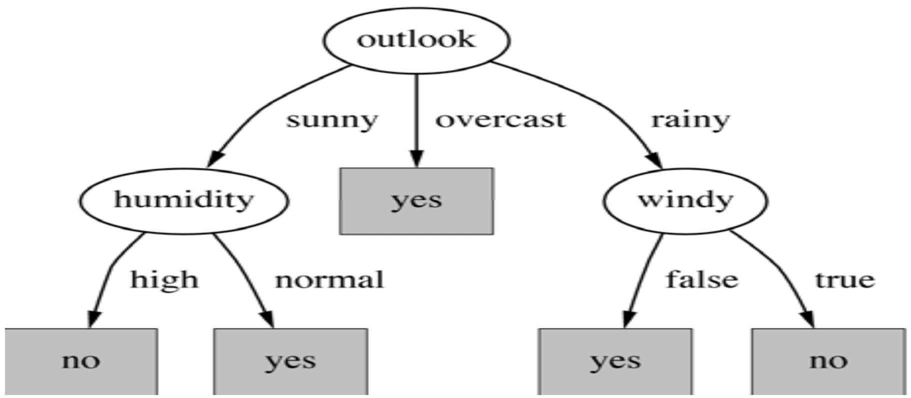
- To introduce some data mining modelling techniques
- To discuss the major modelling issues of these techniques

Agenda

- Introduction to Decision trees
- Decision tree modelling with R
- Introduction to Neural networks
- Introduction to Support vector machines
- Neural network and SVM modelling with R

Decision Tree

- A decision tree is a flow-chart-like tree structure.
  - An internal node performs a test on an attribute
  - A branch represents a result of the test
  - A leaf node represents a class label
  - At each node, one feature is chosen to split training examples into distinct classes
  - A new sample is classified by following a matching path to a leaf node



Building Decision Tree

- Top-down tree construction
  - At start, all training data are at the root.
  - Partition the examples recursively by choosing one feature each time.
- Bottom-up tree pruning
  - Remove subtrees or branches, in a bottom-up manner, to improve the estimated accuracy on new cases.

Choosing the Splitting Attribute

- At each node, available attributes are evaluated on the basis of separating the classes of the training examples. A goodness function is used for this purpose.
- Typical goodness measures:
  - Information gain (ID3/C4.5)
  - Information gain ratio (C4.5)
  - Gini index (CART)

Heuristic Search

- **Search bias:** Search the space of decision trees from simplest to increasingly complex (greedy search, no backtracking, prefer small trees)
- **Search heuristics:** At a node, select the attribute that is most useful for classifying examples, split the node accordingly
- **Stopping criteria:** A node becomes a leaf
  - if all examples belong to same class C_j, label the leaf with C_j
  - if all attributes were used, label the leaf with the most common value C_k of examples in the node

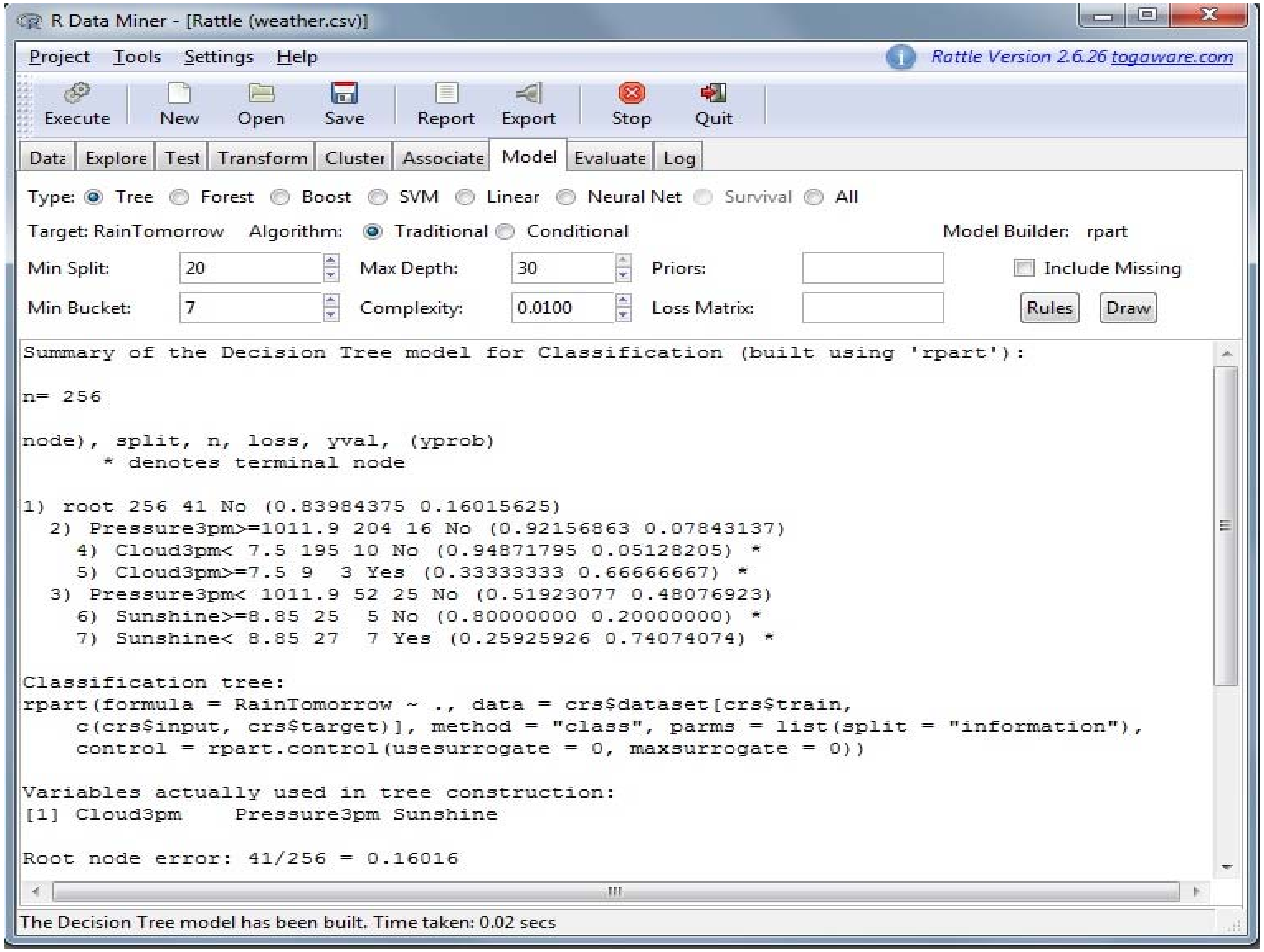
Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Pre-pruning (forward pruning): stop growing the tree e.g.
    - When data split not statistically significant
    - Too few examples are in a split
  - Postpruning: *Remove branches* from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"
- Forward pruning considered inferior

Decision Tree in R

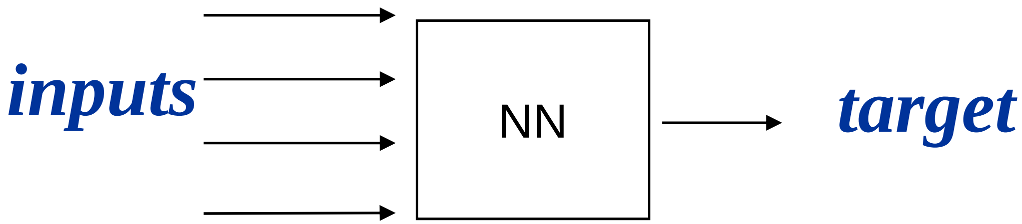
- **rpart**()
- Tuning parameters for rpart()
  - Modeling method (Method=)
  - Splitting Function (split=)
  - Minimum Split (minsplit=)
  - Minimum Bucket Size (minbucket=)
  - Complexity Parameter (cp=)
  - priors=, loss=, ...

Decision Tree in Rattle



Neural Networks

- Neural Networks (NN) are biologically inspired and attempt to build computational models that operate like a human brain.
- These networks can "learn" from the data and recognize patterns.



Architectures vs. Applications

- *Classification/diagnosis:*

  MLFF with BP

  RBF (Radial Basis Function)

- *Forecasting /Prediction / Function approximation / General mapping:*

  MLFF with BP

  RBF (Radial Basis Function)

  GRNN (Generalised Regression Neural Network)

- *Clustering / Grouping:*

  SOM (Kohonen's Self Organising Map)

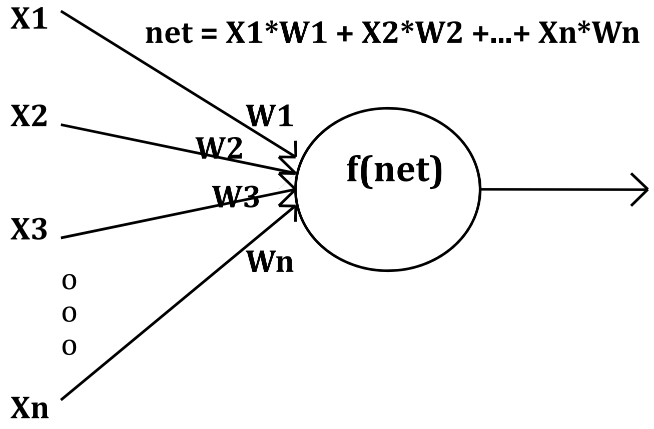
- *Data compression:*

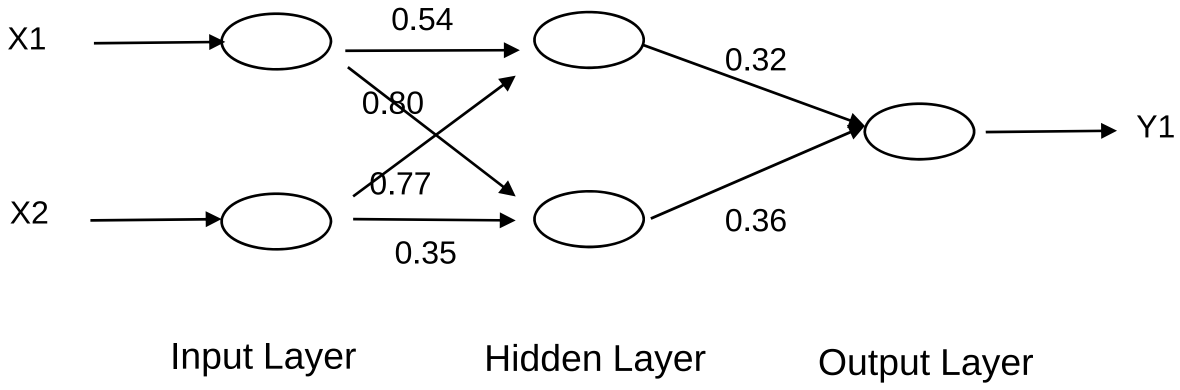
  SOM (Kohonen's Self Organising Map)

  MLFF with BP

General Architecture of Neural Networks

- Framework (in general, but not for all NNs)
  - Input layer + Hidden Layer + Output Layer
  - Weights
  - Activation functions





General Architecture of Neural Networks (cont.)

- Weights
  - Normally initial weights are randomised to small real numbers
- Learning rule
  - determine how to adapt connection weights in order to optimise the network performance $\mathbf{W_i(t+1)=W_i(t)+\Delta W_i(t)}$
- Activation calculation & Weight adjustment
  - Compute the activation levels across the network
  - Weight adjustment based on the errors /distance

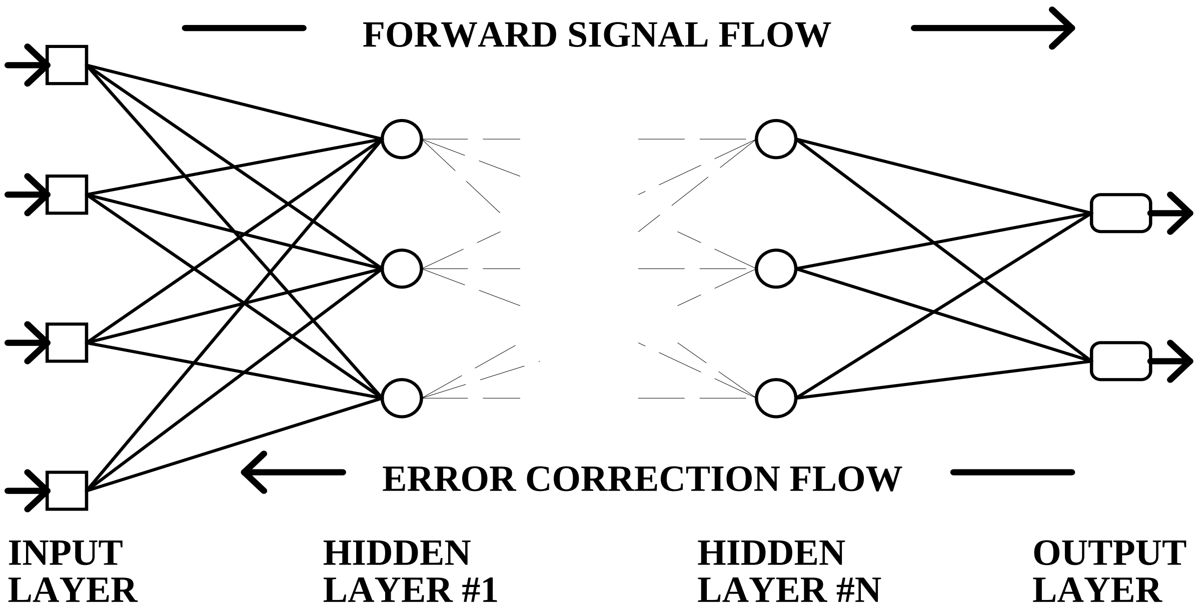
NN Architectures

MultiLayer FeedForward Networks (MLFF)

- Also referred to as Multilayer Perceptron (MLP)
- Features
  - Fully connected units
  - 1 input layer, 1 output layer and >=1 hidden layers
  - Non-linear, differentiable activation function *(typically the logistic or tanh function)*
  - Local minimum and overtraining problems
- Possible applications
  - Pattern classification, function approximation, time series prediction, forecasting …..
  - The most widely applied NN architecture

Multilayer Feedforward NN and Backpropagation Learning

- Propagate signals forward and then errors backward
- Backpropogation (BP) ~ gradient descent learning
- Weights in hidden layers are adjusted to reduce aggregate errors in the output layer



Steps of Backpropagation Algorithm

1. Initialize the weights to small random numbers

2. Randomly select a training pattern pair (x^p, t^p) and present the input pattern x^p to the network. Compute the corresponding network output pattern z^p

3. Compute the error E^p for pattern (x^p, t^p)

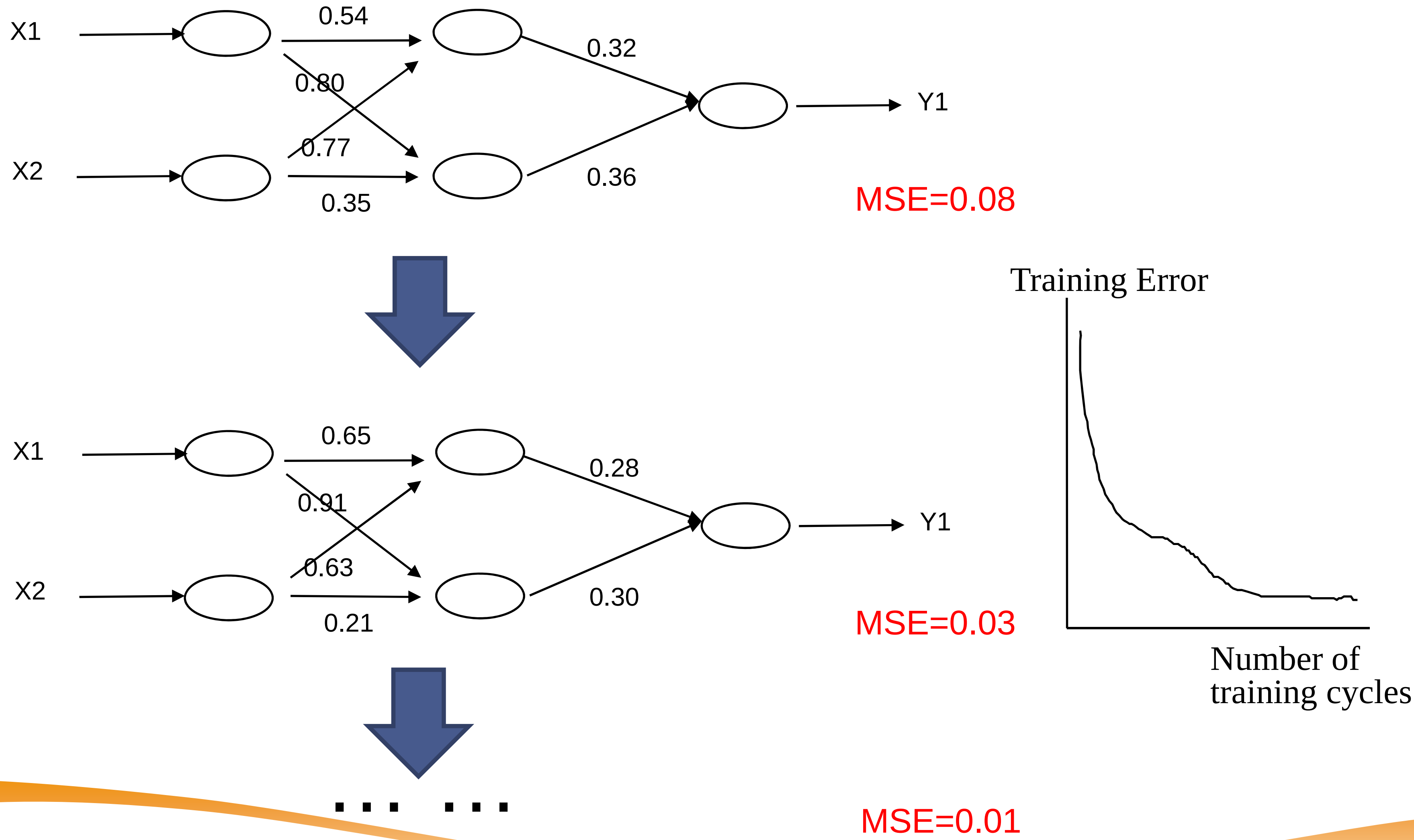
4. Backpropagate the errors according to the BP weight adjustment formulas

5. Test the mean square error (MSE) over *P* training patterns:
   If the MSE is below the required threshold, stop. Otherwise, repeat steps 2-5.

$$E = \frac{1}{P} \sum_{p=1}^{P} E^p$$

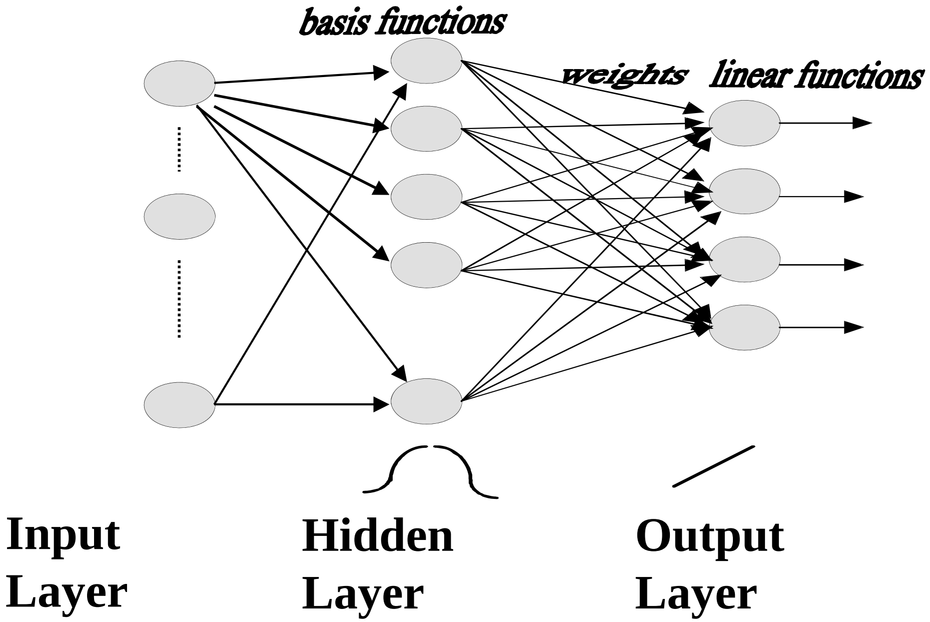
6. Test for generalization performance if appropriate

MultiLayer FeedForward Networks



Radial Basis Function Networks

- Radial basis function networks (RBF networks, or RBFN) use a *hybrid unsupervised and supervised learning*
- Architecture
  - Input layer is *fully* connected to the hidden layer
  - The hidden layer is *fully* connected to the output layer



Radial Basis Function Networks

- Training is performed in two stages:
  - finding centre and smoothing parameter values for the hidden (kernel) nodes (*unsupervised learning*)
    - Usually Hard C-Means clustering is employed
  - finding weights for the output nodes (*supervised learning*)

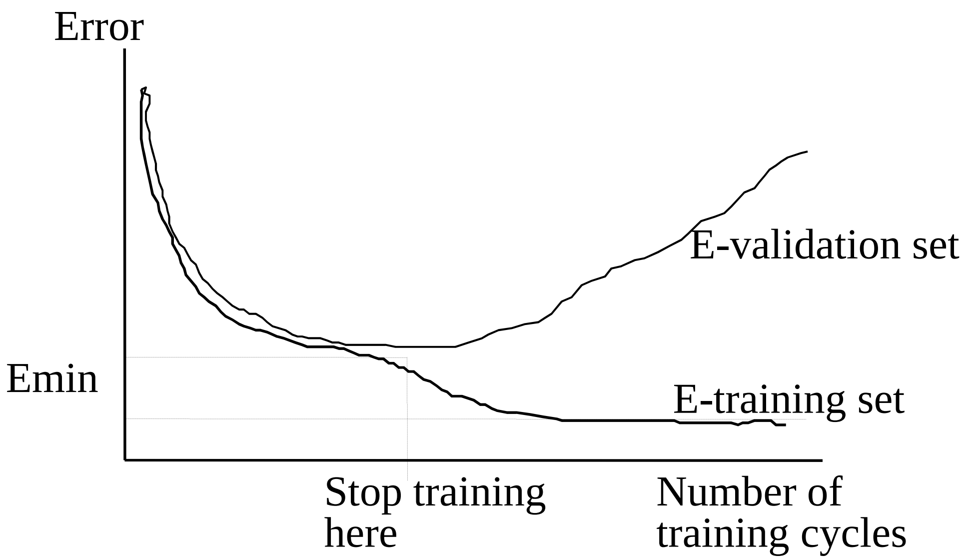
- Applications of RBF include function approximation, kernel regression, forecasting, etc.

Developing a Neural Network

- Specify NN architecture
  - Number of layers
  - Number of nodes in each layer
- Specify the type of learning
  - Learning algorithm (e.g. BP)
  - Learning rate [0, 1]
- Control the amount of learning
  - Error rate to stop at
  - Number of training epochs
  - Amount of time for learning
- WARNING: beware of over-training

Issues of Training: Generalization & Overtraining /Overfiting

- *Generalization* is the ability of a network to correctly classify a pattern it has not seen (not been trained on). NNs generalize when they recognize patterns not previously trained on or when they predict new outcomes from past behaviors.
- Networks can be *overtrained*. It means that they memorize the training set and are unable to generalize well.



Developing a Neural Network

- Training/test data set
  - Perform statistical analyses to support data set choices
  - Select representative training set
  - Divide training set & testing set appropriately
- Pre-processing the data
  - Data Smoothing
  - Data Transformation
    - Log $y=\log(x)$
    - Delta $\Delta x_i = x_i - x_{i-1}$
    - Normalization $$\mathbf{y} = \frac{\mathbf{x} - \mathbf{min}(\mathbf{x})}{\mathbf{max}(\mathbf{x}) - \mathbf{min}(\mathbf{x})}$$
    - Normalized Z score $$z = \frac{x - \mu}{\sigma}$$

Testing / Evaluation

- Mean Squared Error (MSE) measure is relevant and acceptable in many NNs
- Testing the Generalization ability of a trained NN
  - Look for good performance on a validation set and test set
  - Changing the training algorithm
- The performance varies with training/ solution procedures
- Periodic performance testing is essential to verify model's accuracy

NN Modeling with R and Rattle

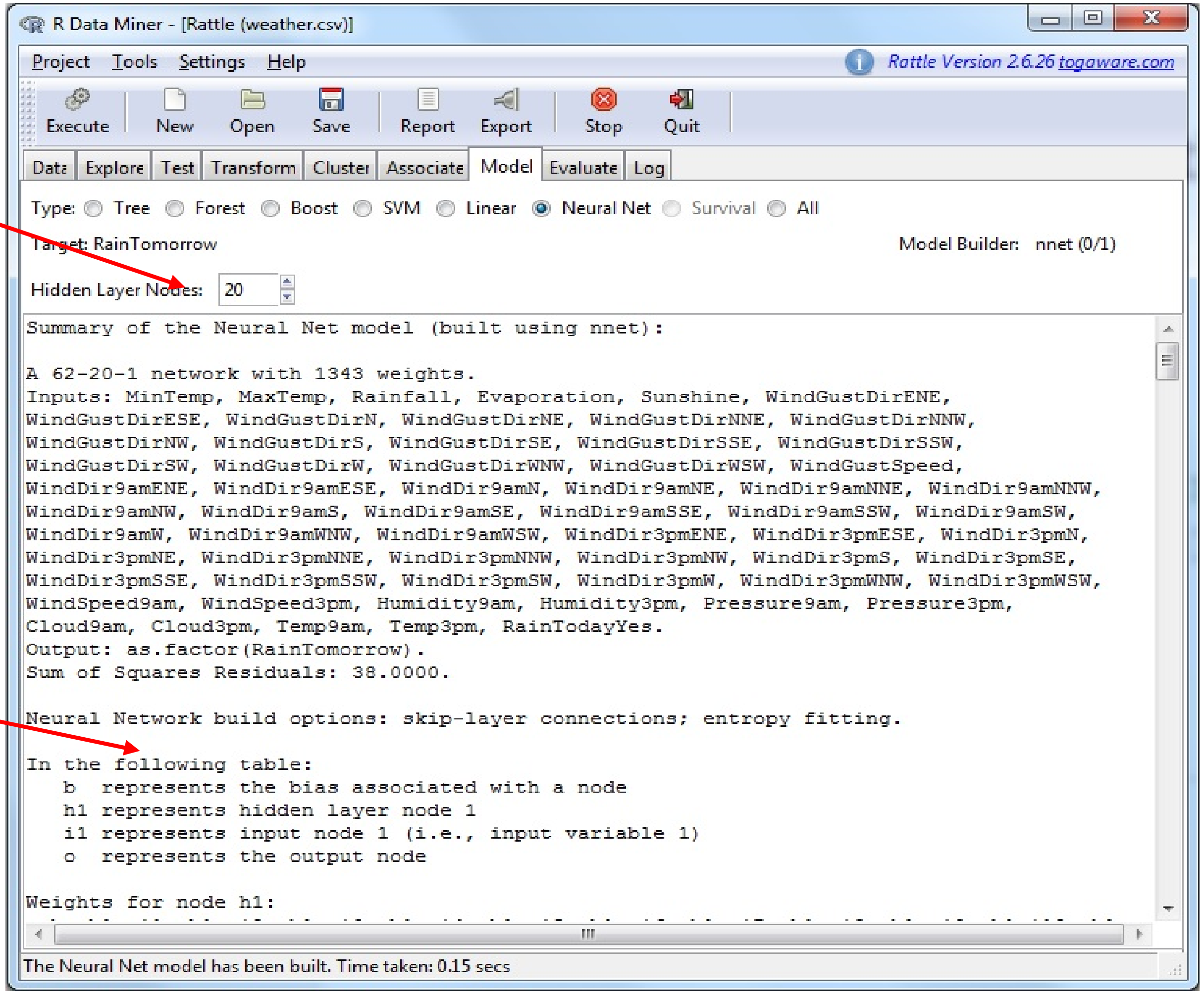
- nnet()

```
crs$nnet <- nnet(as.factor(RainTomorrow) ~ .,
        data=crs$dataset[crs$sample,c(crs$input, crs$target)],
        size=20, skip=TRUE, MaxNWts=10000, trace=FALSE,
        maxit=100)
```

NN Modeling with R and Rattle

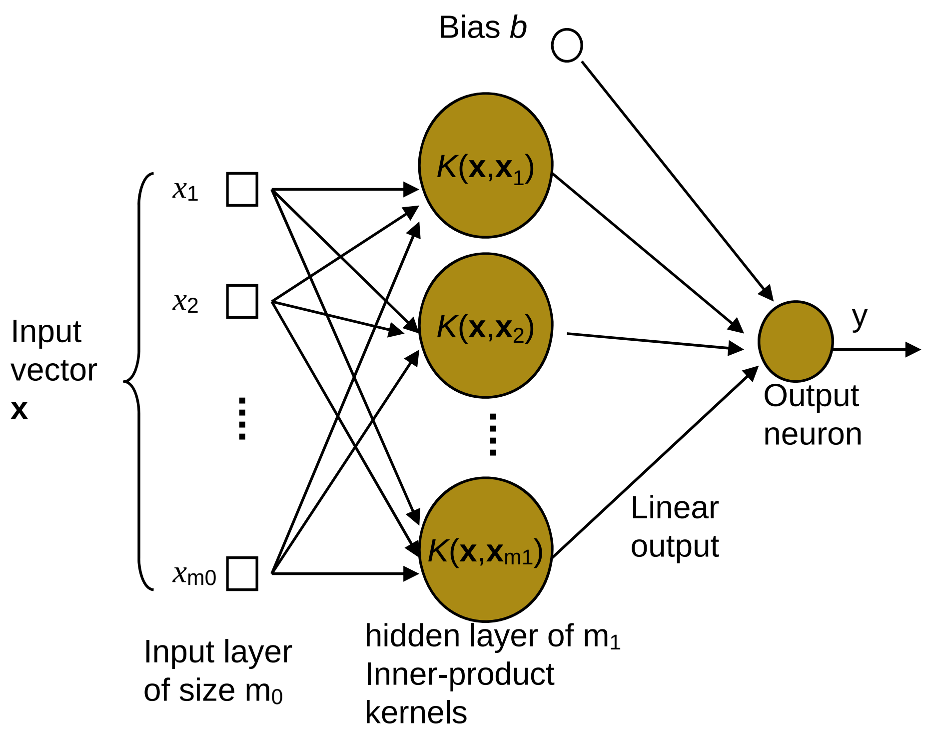
nnet () allows only one hidden layer, can specify number of hidden nodes

Neural network architecture and final weight set



Support Vector Machines — A Brief Introduction

- Another category of feed forward networks [Vapnik, 1992, 1995, 1998]
- Similar to MLFF and RBF, SVM can be used for pattern classification and non-linear regression – but uses statistical learning theory
- General architecture of a support vector machine
  - **Input layer**
  - **Hidden layer of Inner-product kernels (fully connected with the input layer)**
  - **Output neuron**



Support Vector Machines

- A relatively new classification method for both <u>linear and nonlinear</u> data
- It uses a <u>nonlinear mapping</u> to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating **hyperplane**
- SVM finds this hyperplane using **support vectors** ("essential" training tuples) and **margins** (defined by the support vectors)
- Training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- <u>Applications</u>:
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

Support Vector Machines: Optimal Hyperplane & Support Vector

- Important concepts from the theoretical background
  - *Optimal hyperplane* for separable or non-separable patterns
  - *Support vector*
- A training pattern can be represented as a *vector* from the problem space
- Consider linearly separable patterns
  - Training samples: $\{(x_i, d_i)\}$ $\quad i = 1, 2, \ldots, N$

    x_i: the input pattern for the *i*-th example

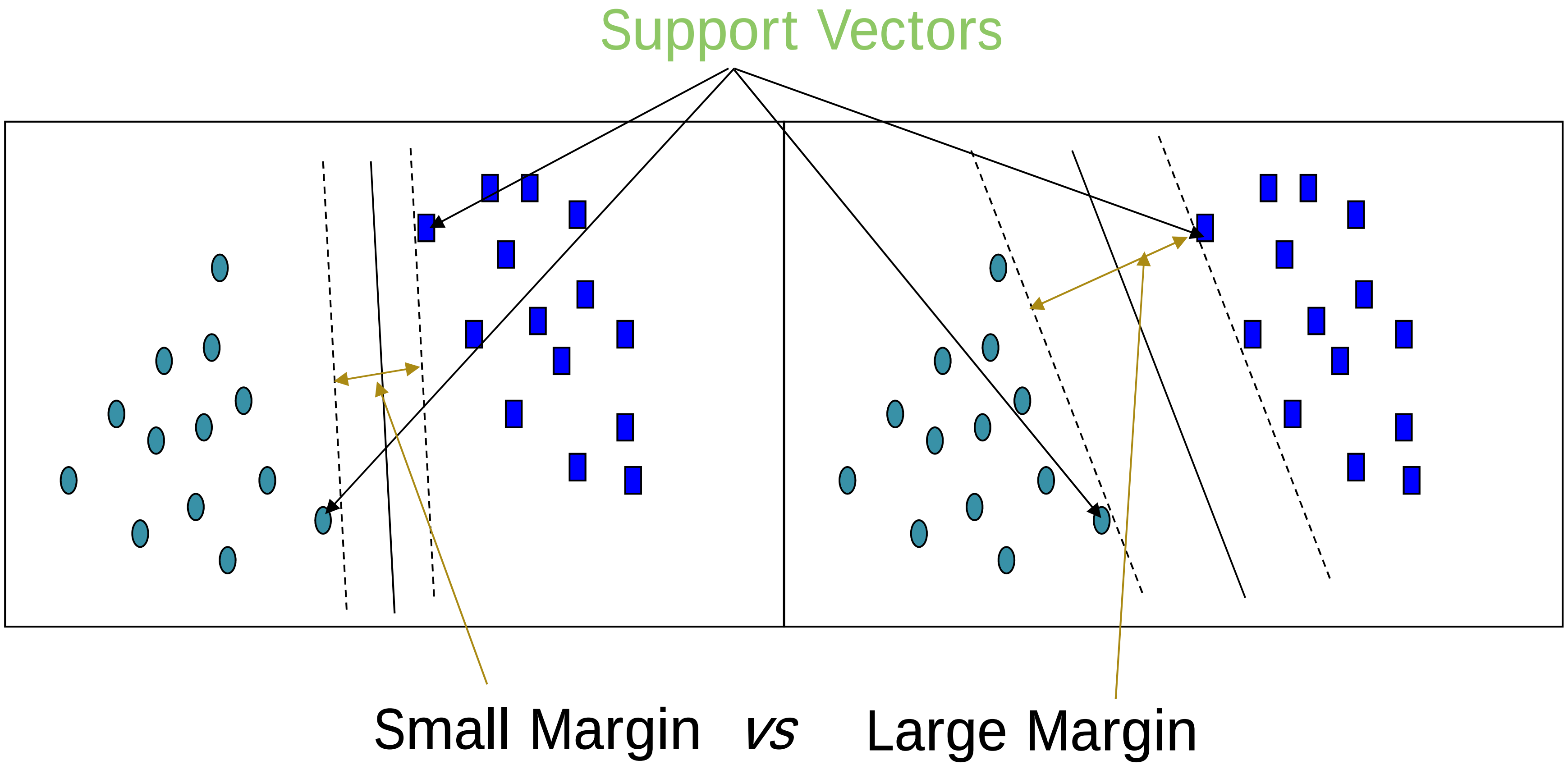
    $d_i \in \{0, 1\}$ (or $\{-1,1\}$): the corresponding desired output
  - The decision surface for the separation is a hyperplane

    $w^T x + b = 0$ (e.g. $w_1x_1 + w_2x_2 + \ldots + w_Nx_N + b = 0$)

    i.e. $w^T x + b \geq 0$ for $d_i = 1$

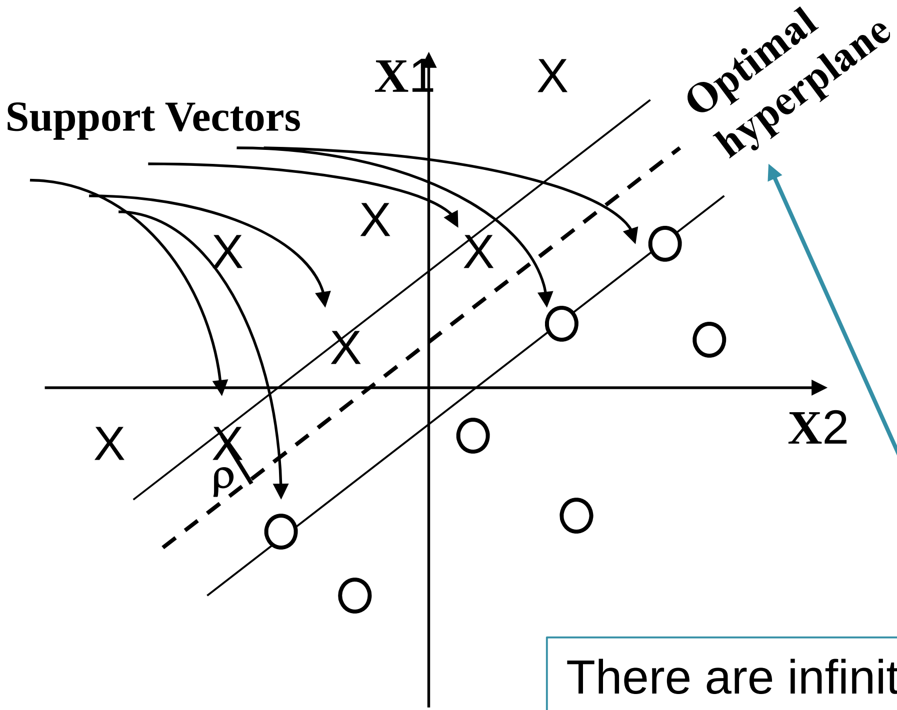
    $w^T x + b < 0$ for $d_i = 0$ (or -1)
- *Margin of separation* (denoted as ρ)
  - The separation between the decision surface hyperplane and the closest data point;

Support Vector Machines: Separation Margin & Support Vector



Support Vector Machines: Optimal Hyperplane & Support Vector

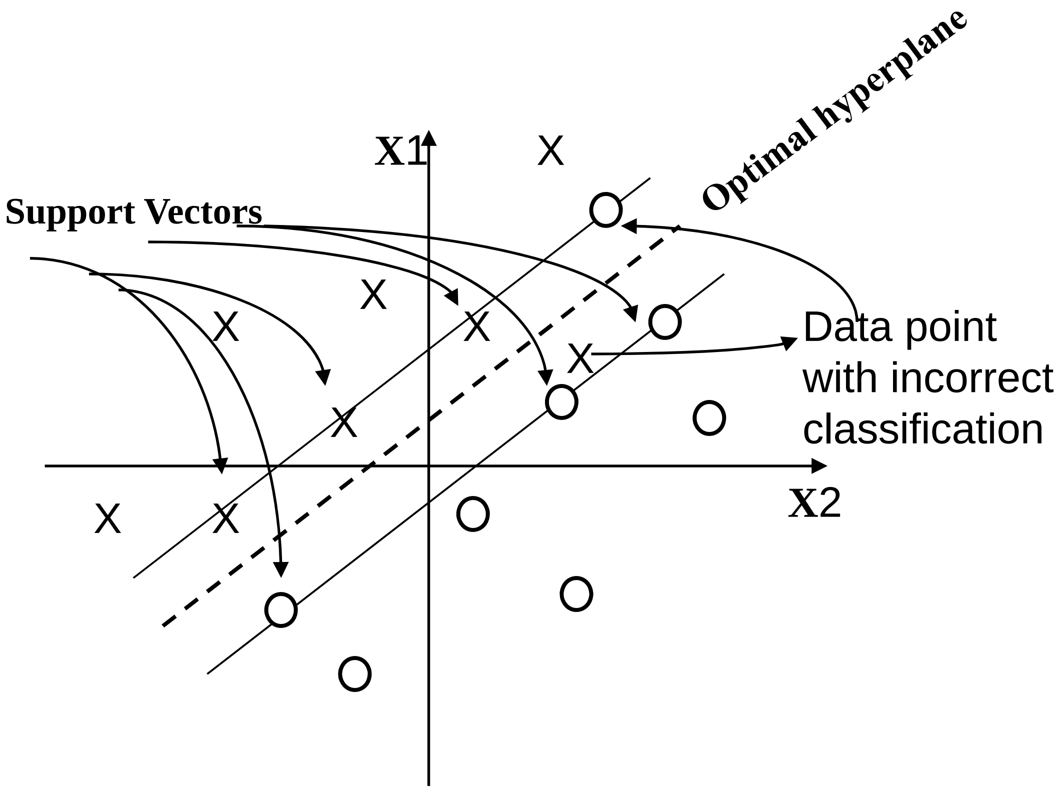
- The goal of a support vector machine for *linearly separable patterns* is to find the particular hyper-plane for which the margin of separation ρ is maximized.
- *Support vectors*: those data points that lie closest to the decision surface and are therefore the most difficult to classify



There are infinite hyperplanes, but SVM searches for the optimal hyperplane.

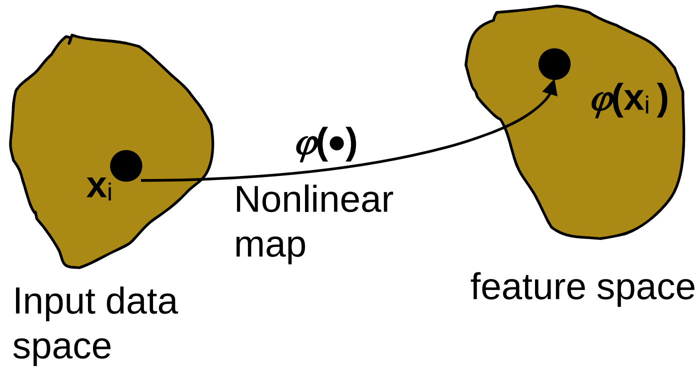
Support Vector Machines: Optimal Hyperplane & Support Vector

- Given a set of non-separable training patterns, it is not possible to construct a separating hyperplane without encountering classification error.
- The goal of a support vector machine for *nonseparable patterns* is to find an optimal hyperplane that minimizes the misclassification error, averaged over the training set.



Support Vector Machines

- To construct a SVM for classification with an input space made up of non-linearly separable patterns
- Form Inner-product kernels
  - The multidimensional input space is transformed to a new feature space where the patterns are linearly separable with high probability, provided



(a) The transformation is nonlinear

(b) The dimensionality of the feature is high enough

  - A subset of training samples $\{x_1, x_2, \ldots x_{m1}\}$ will be used as support vectors
- Define the separating hyperplane as a linear function of vector drawn from the feature space rather than the original input space

SVM : Typical Kernel functions for Nonlinear Classification

- Apply a kernel function $K(\mathbf{X_i}, \mathbf{X_j})$ to the original data, i.e.

$$K(\mathbf{X_i}, \mathbf{X_j}) = \Phi(\mathbf{X_i})\ \Phi(\mathbf{X_j})$$

- Typical Kernel Functions

Polynomial kernel of degree h: $K(\boldsymbol{X_i}, \boldsymbol{X_j}) = (\boldsymbol{X_i} \cdot \boldsymbol{X_j} + 1)^h$

Gaussian radial basis function kernel: $K(\boldsymbol{X_i}, \boldsymbol{X_j}) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$

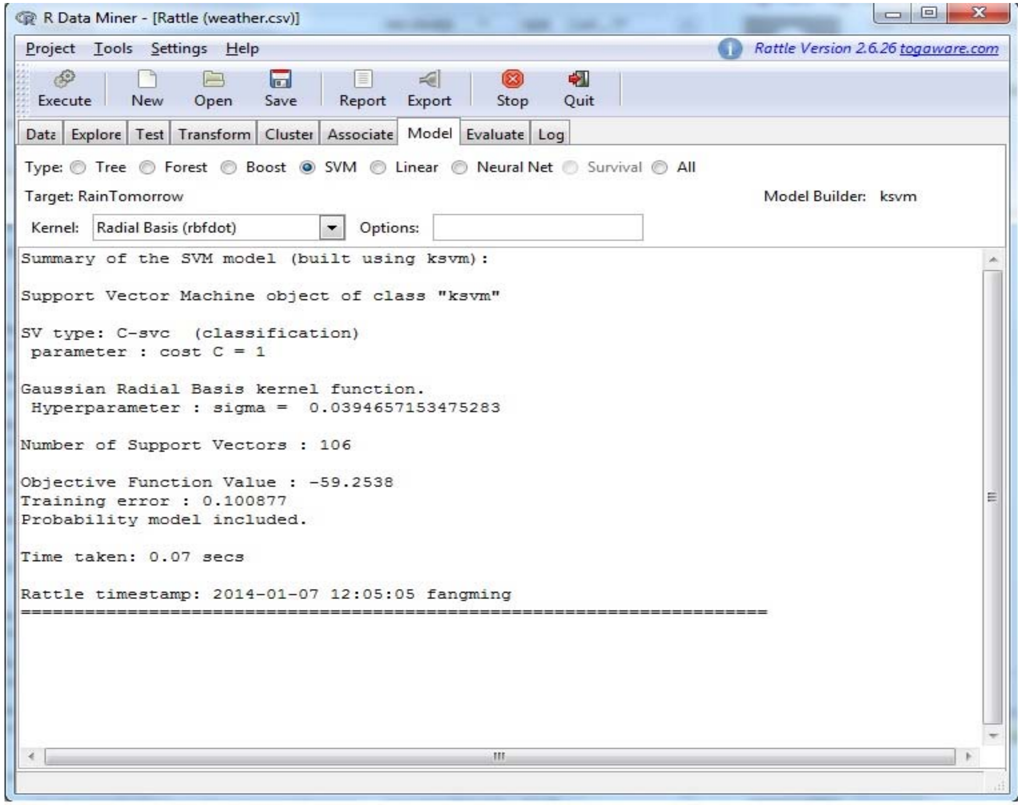
Hyperbolic Tangent kernel: $K(\boldsymbol{X_i}, \boldsymbol{X_j}) = \tanh(\kappa \boldsymbol{X_i} \cdot \boldsymbol{X_j} - \delta)$

Support Vector Machines

- The SVM is an elegant and highly principled learning method for the design of a feedforward network with a single hidden layer of nonlinear units
- Design hinges on the extraction of a subset of the training data that serves as support vectors and therefore represents a stable characteristic of the data
- Learning in SVM
  - Learning algorithm operates only in a batch mode
  - The near-to-perfect classification performance is achieved at the cost of a significant demand on computational complexity
- The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- An SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

SVM with R and Rattle

```
crs$ksvm <- ksvm(as.factor(RainTomorrow) ~ .,
            data=crs$dataset[crs$train,c(crs$input, crs$target)],
            kernel="rbfdot",
            prob.model=TRUE)
```



Summary

- Decision Tree, Neural network and Support Vector Machine are important data mining modeling techniques.
- Available in almost all data mining tools and packages, such as R, Rapidminer, SAS, SPSS modeller