Non-Bayesian Classifiers
Part III: Neural Networks and Decision Trees

Selim Aksoy

Department of Computer Engineering
Bilkent University
saksoy@cs.bilkent.edu.tr

CS 551, Spring 2010

Neural Networks

- Linear discriminants try to find hyperplane decision boundaries.
- Kernel mapping can be used to obtain arbitrary decision regions.
- *Neural networks* try to learn the parameters of the nonlinear mapping at the same time as the linear discriminant.
- A neural network consists of an *input layer*, an *output layer*, and usually one or more *hidden layers* that are interconnected by modifiable weights represented by links between layers.

Neural Networks

- The intermediate layers are called hidden because their activation are not directly seen from the input or output.
- The function of units is loosely based on properties of biological neurons, and hence they are sometimes called *neurons*.
- In pattern recognition applications, the input units represent the components of a feature vector and the output units give the values of the discriminant functions used for classification.

Neural Networks

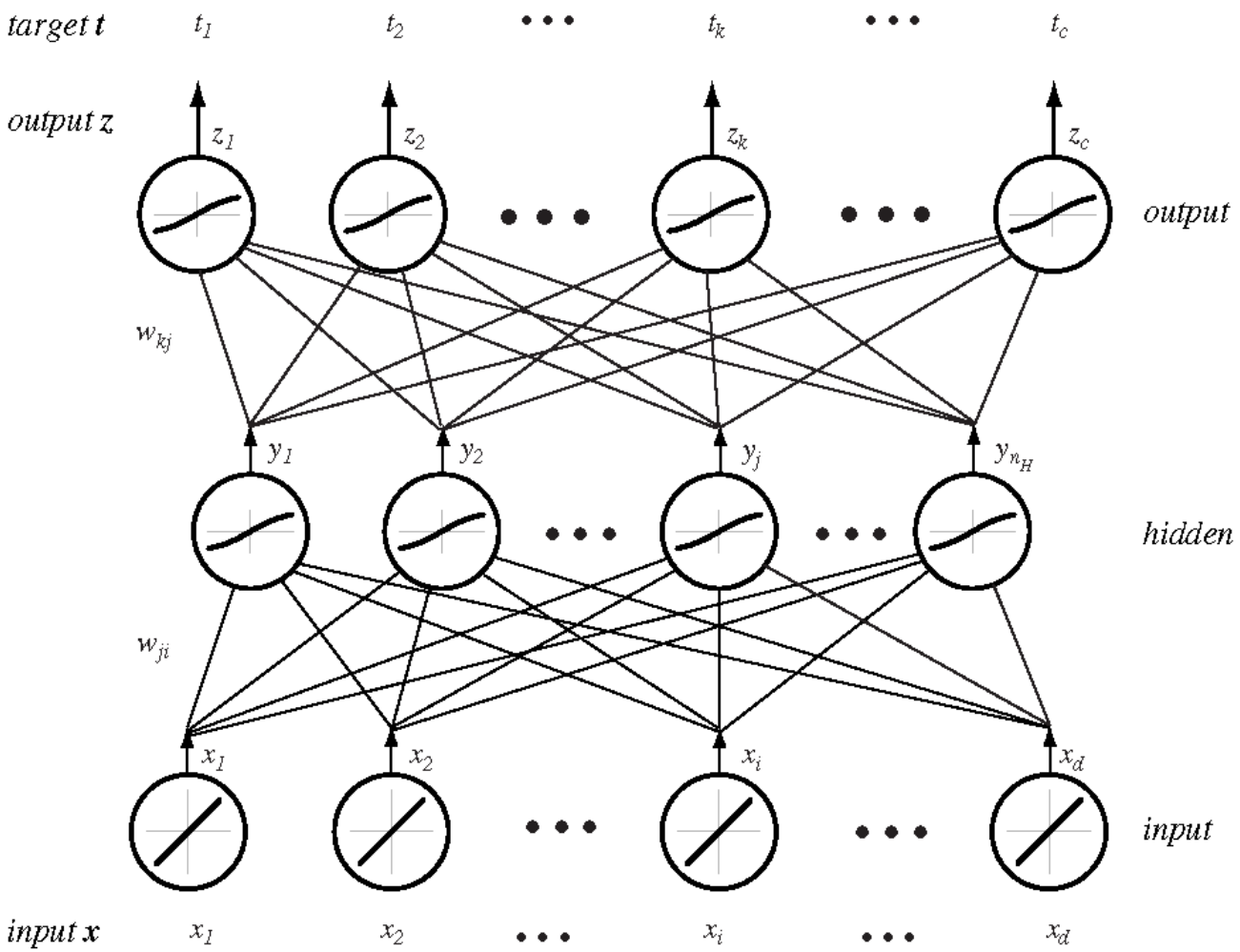

Figure 1: A $d - n_H - c$ fully connected three-layer network.

Feedforward Operation

- Each hidden unit computes the weighted sum of its inputs to form a net activation value, $net = \mathbf{w}^T\mathbf{x}$.
- Then, it emits an output that is a nonlinear function of its activation, $f(net)$.
- Each output unit similarly computes its net activation based on the hidden unit signals in the previous layer, and emits a value using a nonlinear function based on its activation.
- The output corresponds to a nonlinear function of the input feature vector.

Neural Networks

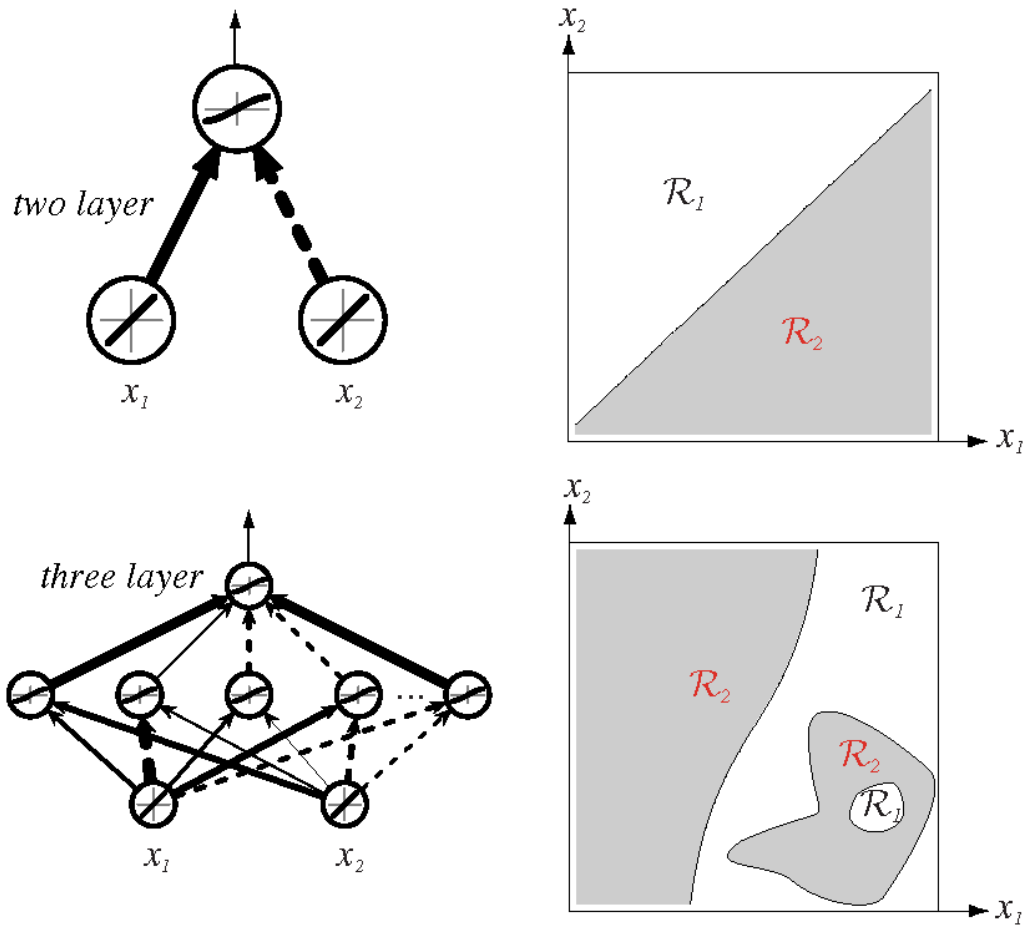

Figure 2: Whereas a two-layer network classifier can only implement a linear decision boundary, given an adequate number of hidden units, three or more layer networks can implement arbitrary decision boundaries.

Neural Networks

- We can think of the network as computing discriminant functions, and can classify the input according to which discriminant function is the largest.
- Theoretically, every decision (and every continuous function) can be implemented by a three-layer network given sufficient number of hidden units, proper nonlinearities, and weights.
- However, this theorem does not tell us how we can learn the network structure, the nonlinear functions, and the weights in practice.

Backpropagation Algorithm

- *Backpropagation* is one of the simplest and most general methods for supervised training of multilayer neural networks.
- It is based on the least-mean-square algorithm where the error is proportional to the square of the difference between the actual output and the desired output.
- The dependence of the error on the hidden-to-output layer weights is straightforward.
- The backpropagation algorithm allows us to calculate an effective error for each hidden unit and derive a learning rule for the input-to-hidden weights.

Backpropagation Algorithm

- Using the notation of Figure 1, define the training error on a pattern to be the sum over output units of the squared difference between the desired output t_k and the actual output z_k as

$$J(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2.$$

- The backpropagation learning rule changes the weights in a direction that reduces the error (gradient descent) as

$$\Delta w_{pq} = -\eta \frac{\partial J}{\partial w_{pq}}$$

where η is the learning rate.

Backpropagation Algorithm

- For the hidden-to-output weights, the weight update rule becomes

$$\Delta w_{kj} = -\eta \frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} = \eta (t_k - z_k) f'(net_k) y_j$$

  where net_k is the net activation value.

- For the input-to-hidden weights, the weight update rule becomes

$$\begin{aligned} \Delta w_{ji} &= -\eta \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} \\ &= \eta \left(\sum_{k=1}^{c} w_{kj} (t_k - z_k) f'(net_k) \right) f'(net_j) x_i. \end{aligned}$$

Backpropagation Algorithm

- The algorithm is called “backpropagation” because it propagates the error from the output layer back to the hidden layers.
- As with all gradient descent procedures, the exact behavior of the backpropagation algorithm depends on the starting point.
- The initial weights are randomly chosen from a uniform distribution symmetric around zero.

Backpropagation Algorithm

- There are three main training protocols for neural networks:
  - In *stochastic training*, patterns are chosen randomly from the training set and the weights are updated for each pattern presentation.
  - In *batch training*, all patterns are presented to the network before learning takes place.
  - In *online training*, each pattern is presented only once.

Decision Trees

- Most pattern recognition methods address problems where there is a natural measure of distance between feature vectors.
- What happens when the classification problem involves *nominal* (*categorical*) data, e.g., descriptions that are discrete and without any natural notion of similarity or even ordering?
- A common representation for this kind of data is a *list of attributes* (instead of a vector of real numbers).

Decision Trees

- It is natural and intuitive to classify a pattern through a sequence of questions, in which the next question asked depends on the answer to the current question.
- Such a sequence of questions can be represented as a *decision tree*.
- In a decision tree, the top node is called the *root node*, and is connected by directional links (*branches*) to other nodes.
- These nodes are similarly connected until terminal (*leaf*) nodes, which have no further links, are reached.

Decision Trees

- The classification of a particular pattern begins at the root node, which checks for the value of a particular attribute.
- Different branches from the root node correspond to different values of this attribute that are mutually distinct and exhaustive.
- Based on the particular value of that attribute for that pattern, the appropriate branch is followed to a subsequent node.
- Similar decisions are made in subsequent nodes until a leaf node is reached.
- Each leaf node bears a class label and the pattern is assigned the label of the leaf node reached.

Decision Trees

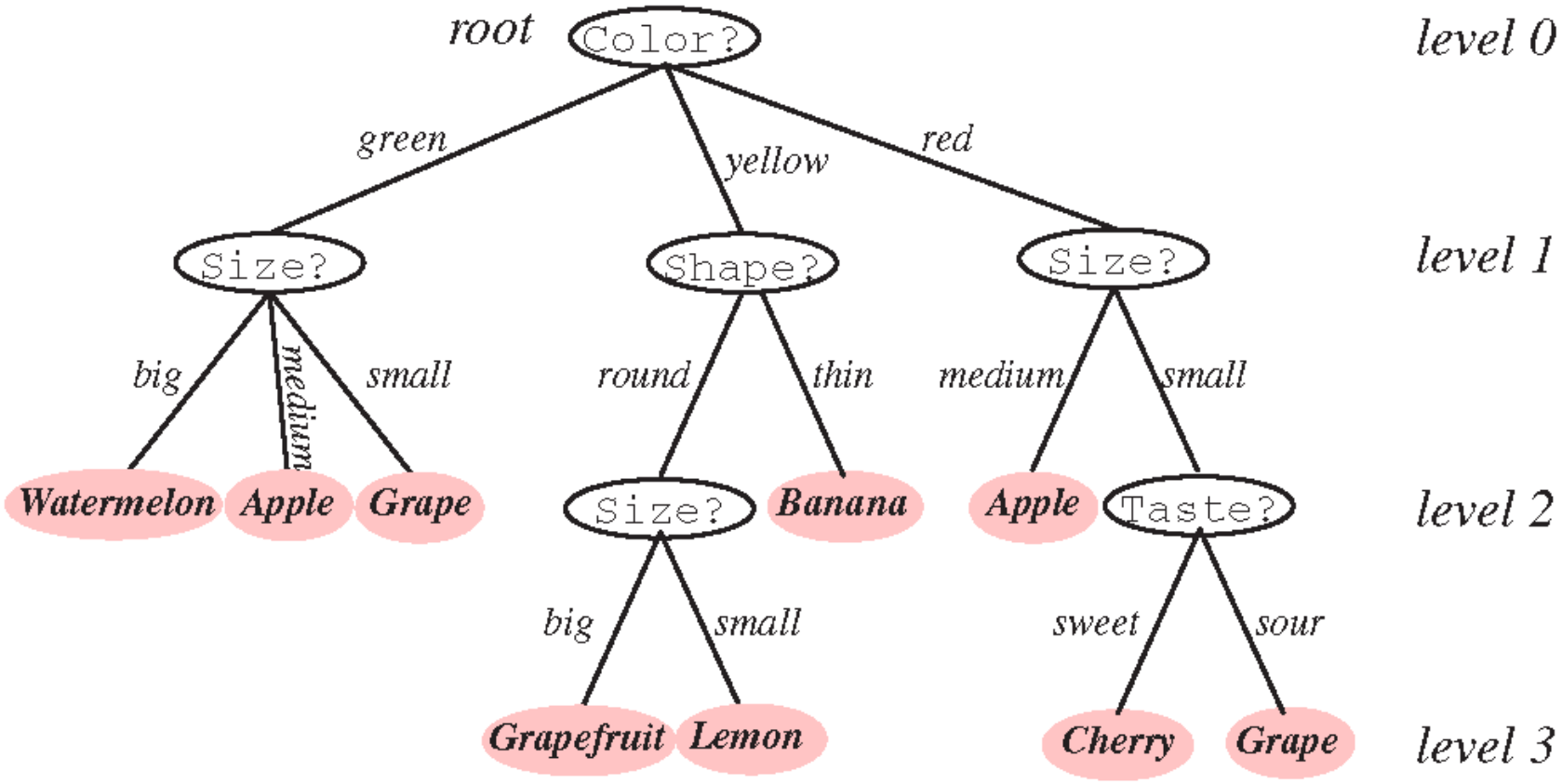

Figure 3: An example decision tree that uses the attributes {color, shape, taste, size}. Note that the same question can appear in different places in the tree and that different questions can have different numbers of branches. Moreover, different leaf nodes can be labeled by the same class.

Decision Trees

- Learning a decision tree is based on partitioning the set of training examples into smaller and smaller subsets where each subset is as *pure* as possible.
- Purity for a particular subset is measured according to the number of training samples in that subset having the same class label.
- Different criteria can grow trees differently.

Decision Trees

- Decision tree learning is a recursive process where the following questions arise:
  - Are the attributes numerical or nominal?
  - Are the nominal attributes binary-valued or multi-valued?
  - How many branches will there be at a node?
  - Which attribute should be tested at a node?
  - When should a node be declared as a leaf?
  - If a tree becomes too large, how can it be simplified?
  - How can a class label be assigned to a leaf node?
  - How should missing data be handled?

Decision Trees

- Each decision outcome at a node is called a *split*.
- Each tree can be represented using just binary splits (called a *binary tree*).
- For numerical attributes, splitting questions have the form "is $x \leq x_0$"?
- For nonnumerical (nominal) attributes, splitting questions have the form "is $x \in A$" where A is a subset of the possible values of x.

Decision Trees

- For a given node, the particular splitting attribute and the corresponding question can be chosen using a search according to the purity (or impurity) measures that are based on entropy, variance, Gini or misclassification criteria.
- Commonly used criteria about deciding when to stop splitting include thresholds on impurity or number of examples remaining at a node, or statistical tests on the significance of reduction in impurity.

Decision Trees

- Alternatively, a tree can be grown fully, and then can be *pruned* by considering the leaf nodes or even subtrees for elimination or merging.
- Once the leaf nodes are finalized, they can be labeled by the class that has the most patterns represented in the corresponding nodes.

Decision Trees

- An interesting problem that can arise during training, classification or both is *missing data*.
- There are several possible reasons for a value to be missing, such as:
  - it was not measured,
  - there was an instrument malfunction,
  - the attribute does not apply,
  - or the attribute's value cannot be known.

Decision Trees

- Decision trees can handle missing data by using the primary decision attribute at a node whenever possible, and use alternative attributes when a pattern is missing the primary attribute.
- These alternative attributes are called *surrogate splits* and are found by maximizing the probability of making the same decision as the primary split.

Decision Trees

- In summary, tree-based tools offer the following advantages:
  - They can operate on both numerical and nominal measurements.
  - They do not require any assumptions about neither the distributions nor the independence of attribute values.
  - They are often easy to interpret by creating subgroups of data which the user may graphically analyze.

Decision Trees

- Advantages of tree-based tools (cont.):
  - They can be easily converted to decision rules by tracing the tree from the root node to each leaf node and forming logical expressions.
  - They automatically perform feature selection by using only the attributes that can partition the measurement space the most effectively.
  - They are capable of dealing with missing data.