Machine Learning

Dr. Corina Sas

Computing Department
Lancaster University

Overview

- Supervised learning
  - Artificial Neural Networks (some)
  - Decision Trees
  - Learning Vector Quantisation
- Unsupervised learning
  - Self Organising Maps

Machine Learning Algorithms

- Supervised algorithms - require a set of classified examples.
  - Examples:
    - Artificial Neural Networks (some)
    - Decision tress
    - Learning Vector Quantisation (LVQ)
- Unsupervised algorithms - do not require classified examples.
  - Examples:
    - K-means clustering
    - Hierarchical clustering
    - Principal component analysis
    - Self Organising Maps (SOM)

Supervised learning

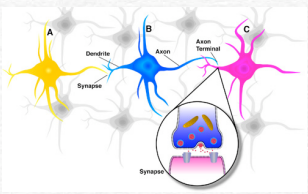
- Creates a function from the training data (inputs and their outputs).
- Task: predict the values of this function for any valid input (not necessarily part of the training set).
- Generalisation from training set to unseen inputs.

Artificial Neural Network

- Neuron
- Multilayer perceptron
- Backpropagation
- Recurrent NNs

Neurons

- Neurons – basic information processing structures
  - Cell body (soma)
  - Dendrites
  - Axon
  - Axon terminals



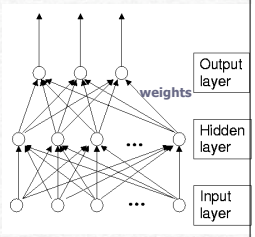
http://www.mind.ilstu.edu/curriculum2/neuro/neuron_1.html

Information Processing

- Neuron functions:
  - Receive input information
  - Process it
  - Send information as output.
- Neural network
  - Synapses enable information to flow
  - Input information determine neuron to generate electric signal (action potential)
    - When the electric property of the membrane reaches a point (threshold)
  - Action potential is transmitted along the axon and its output is sent to other neurons

Multilayer Perceptron

- Input layer generate impulses
- Transmitted to the units in hidden layer
  - Connections: excitatory (+1), inhibitory (-1), or (0).
- A hidden unit becomes active if the sum of its inputs exceed a threshold value.
- Hidden unit produces an output which is sent to the output layer.
- A output unit becomes active if the sum of its inputs exceed a threshold value



Multilayer Perceptron

Training:

- Repeatedly presented with sample inputs and desired targets.
  - During training a pattern is applied to the input layer, and the stimulus is propagated through the layers until an output layer unit is activated.
- Output and targets compared and error measured.
- Adjusts weights until correct output for every (most of) input.
  - If the correct output layer unit is activated, the output of the corresponding hidden layer units is increased
  - If the incorrect output layer unit is activated the output of the corresponding hidden layer units is decreased.

Training phases

Error back-propagation – minimises the error between the network output and the training samples.

Forward pass – an activity pattern is applied and its effect is propagated through the net.

- Activations of hidden layer units calculated from net input (sum input layer units they are connected to *connection weights) then passing through transfer function.
  - Input to hidden unit k: $net_k = \sum w_{ki} * x_i$
  - Output of hidden unit k: $output_k = f(net_k)$
- Activations of output layer units calculated from the activation of hidden layer units (net input*connection weights) then passing through transfer function.
  - Input to output unit i: $net_i = \sum w_{ik} * output_k$
  - Output of output unit i: $output_i = f(net_i)$

Training phases

Backward pass

- The network output is subtracted from the target output to produce an error, which is propagated backwards through the network.
- Compute the difference between actual activation of each output (y_k) and desired target (d_k): error = $d_k - y_k$
- Learning rate parameter η used to control amount weights which are updated during each cycle.
- The weights are not fixed but adjusted, according to:
  - $w_{ki}(t+1) = w_{ki}(t) + \eta[d_k(t) - y_k(t)]\, x_i(t)$,

The two phases are repeated many times for different input patterns and their targets, until error between actual outputs and targets output is small for all training patterns.

Backpropagation

- Backpropagation is an example of supervised learning
- Training inputs and their corresponding outputs are supplied to the network
- The network calculates error signals, and uses these to adjust the weights
- After many passes, the network settles to a low error on the training data
- It is then tested on test data that it has not seen before, to measure its generalisation ability

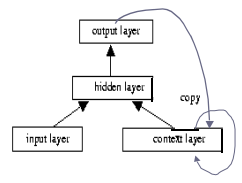
Recurrent Networks

- Time in cognition
  - Cognitive processes and behaviours unfold in real time
- Embodying time in NNs
  - Time should be represented by its effect on processing rather than an extra dimension of the input
- Jordan and Elman NNs

Jordan Recurrent NNs

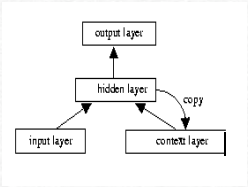
Jordan (1986) added recurrent connections by coping the output units in the context units

- Input neurons receive information directly from the input patterns
- Output neurons provide the response learned by the network
- Hidden neurons whose major role consists of discovering the significant features within the input patterns
- Context neurons which store short-term memory of the network, in terms of the output node activation patterns from previous time step.



Elman Recurrent NNs

- Elman (1990) modified Jordan's net
- Context neurons which store short-term memory of the network, in terms of the hidden node activation patterns from previous time step.
- Hidden units develop internal representations which are encodings of the temporal properties of the sequential input



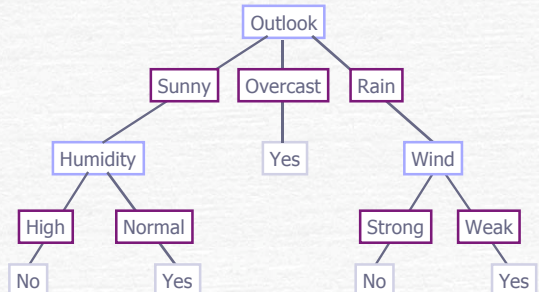
Decision Trees

- Decision trees are powerful tools for classification and prediction.
- Unlike neural networks, decision trees represent *rules*, easily understood by humans.
- Typical inductive approach to learn knowledge on classification.

Decision Trees

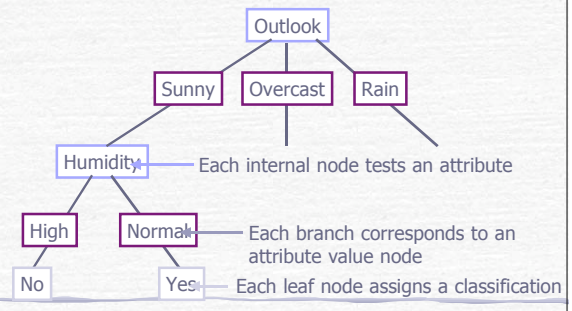
- Training set consists of a set of objects, each described through a set of values on a fixed collection of attributes.
- Pre-defined set of classes to each object from the training set is pre-assigned and each object of the testing set will be assigned by the decision tree.
- Each object belongs to only one class
- There should plenty of training cases (hundreds)
- Decision tree – a representation of a decision procedure for determining the class of an object by testing its value on the set of attributes.

Example



http://www.uonbi.ac.ke/acad_depts/ics/course_material/FoundationsofAI/

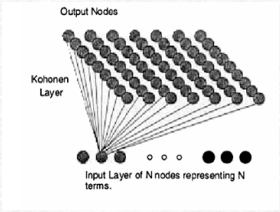
Example



http://www.uonbi.ac.ke/acad_depts/ics/course_material/FoundationsofAI/

SOM

- Unsupervised learning which maps multi-dimensional data into a two-dimensional representational space
- Visualisation of multidimensional data
  - similar patterns are close to each other and dissimilar patterns are far apart



SOM

- Input layer
  - A multidimensional vector with each unit coding the value from one dimension
- Output layer
  - A two-dimensional vector, where each node is associated with a reference vector (m_i): a set of weights from each input node to the specified output node.
- Matrix of connections between each output unit and all the input units.
  - Initially each neuron has own set of (random) weights

(Kohonen, 1996)

SOM

Each input vector is compared to all the reference vectors (weights) and the location of the best match is defined as winner.

Best match described in terms of a metric, i.e. Euclidean distances

- Difference between input vectors and weights of SOM units
- $D=(\sum (net_i - w_{ik})^2)^{1/2}$ (square root of the sum of the squared differences)

Update all weights in neighbourhood around winning unit.

As learning proceeds size of neighbourhood diminished

A winner node has its reference vector adjusted to better match the input vector

Positive excitatory feedback between SOM unit and nearest neighbours

- Causes all the units in 'neighbourhood' of winner unit to learn.
- As distance from winning unit increases degree of excitation falls until it becomes inhibition.
- Adjustment occurs for a neighbourhood around the winner node:
  - $m_i(t+1) = m_i(t) + \eta [x(t) - m_i(t)]$.

SOM

- Training
  - Ordering phase – neurons in different areas of the network learn to correspond to coarse cluster in the date
  - Fine-tuning phase – neurons adjust to reflect fine distinctions, much lengthy
- Winner-takes-all strategy
  - Nodes in the output map compete to each other to represent the input vectors.
  - Output layer = competitive layer
  - Competitive learning – adaptive process through which neurons from the output layer become slowly sensitive to the input data, learning to represent better different types of in inputs.

SOM

- Preserving continuity – maps similar inputs to neighbouring map locations.
- The greater variance between the input vectors features, the better their representation on the output map.
- These features should correspond to the most important dimensions of the inputs.

LVQ

- Learning Vector Quantisation (Kohonen, 1995) – supervised
- Input layer of multidimensional vectors described by their features
- Output layer whose neurons correspond to predefined classes
- Matrix of connections between each output unit and all the input units – weights vectors.
- Since each weight vector corresponds to a class, the vectors are considered labelled.
- Classifying an input vector – computing the distances (Euclidean dist.) between it and all the weight vectors and assigned it to the class associated with the closest weight vector (for which the distance is minimum).

LVQ

- Closest weight vector – the winning node.
- When both the input vector and the weight vector belong to the same class (correct classification), then the weight vector is modified to become a better approximation of the input vector.
- When the input vector is incorrectly classified, then the weight vector is adjusted such that to increase its distance of the input vector (since they belong to different classes).

Example

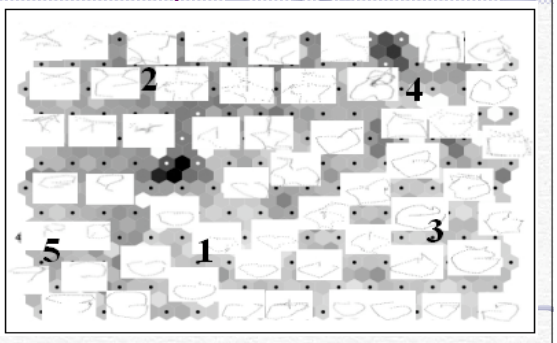
Trajectory classification

- Data pre-processing
  - degree of occupancy of a predefined set of spatial locations and an extra input node representing the degree of rotating in VE (29 nodes)
  - NV = $\log_{10}(9 \times LOC + 1)$

7	14	21	28
6	13	20	27
5	12	19	26
4	11	18	25
3	10	17	24
2	9	16	23
1	8	15	22

7	14	21	28
6	13	20	27
5	12	19	26
4	11	18	25
3	10	17	24
2	9	16	23
1	8	15	22

Map visualisation



LVQ

- Once the SOM was trained, the codebook vectors were used for initialising the weights for LVQ algorithm.
- Overall: classification accuracy increased from 72% obtained using random initialisation to 87%.