Artificial Neural Networks (ANNs) Training

Sabah Sayed

Department of Computer Science

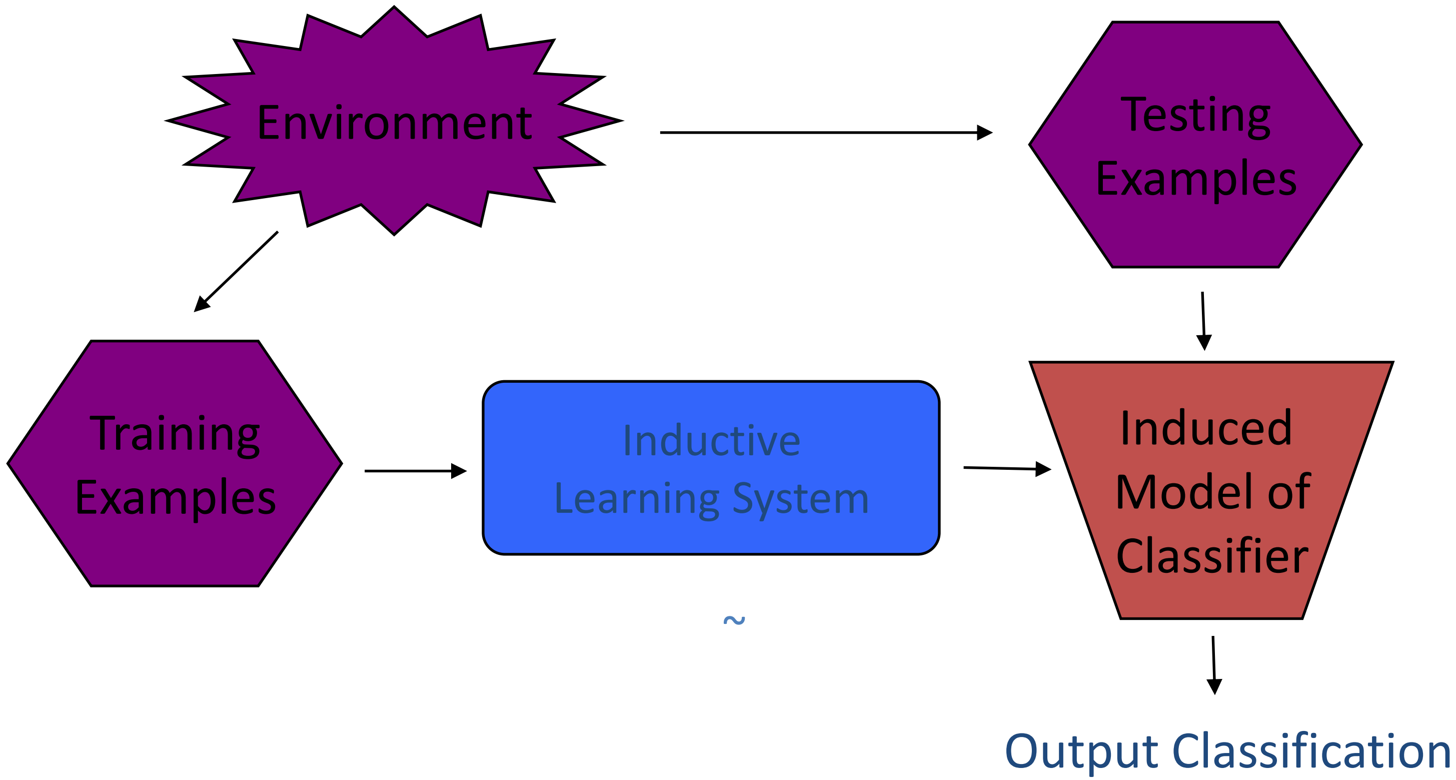
Faculty of Computers and Artificial Intelligence

Cairo University

Egypt

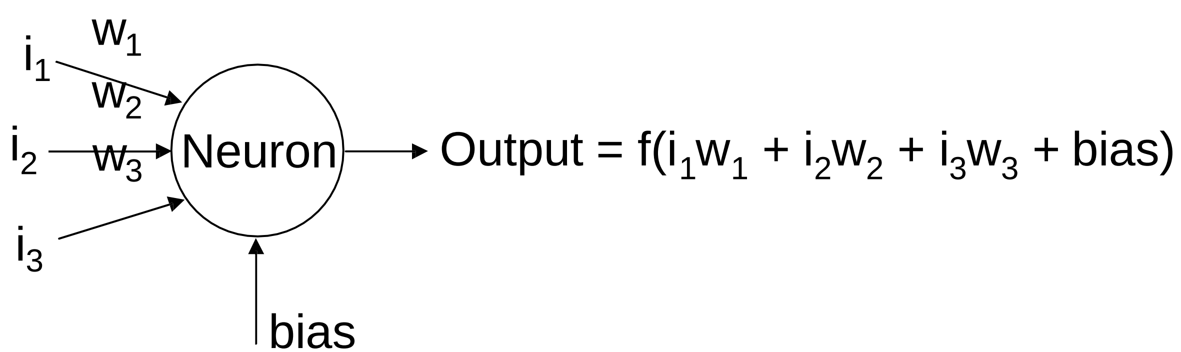
Classification Systems and Inductive Learning

Basic Framework for Inductive Learning



Node biases

- A node’s output is a weighted function of its inputs
- What is a bias?
- **Bias** allows you to shift the activation function by adding a constant (i.e. the given **bias**) to the input
  - output = sum (weights * inputs) + bias
- **Bias** in **Neural Networks** has the same role of a constant in a linear function
  - y =mx + c
- How can we learn the bias value?
  - treat them like just another weight
  - Just add one more input unit to the network topology



Network training

- Two main types of training
  - Supervised Training
    - Supplies the neural network with inputs and the desired outputs
    - Response of the network to the inputs is measured
      - The weights are modified to reduce the difference between the actual and desired outputs
  - Unsupervised Training
    - Only supplies inputs
    - The neural network adjusts its own weights so that similar inputs cause similar outputs
      - The network identifies the patterns and differences in the inputs without any external assistance

Epoch

- One iteration through the process of providing the network with an input and updating the network's weights
- Many epochs are required to train the neural network

Learning rate

- The learning rate is important
  - Range 0.01 - 0.99

  - Too small
    - Convergence extremely slow

  - Too large
    - Converges to a local minimum

Hidden Layers and Neurons

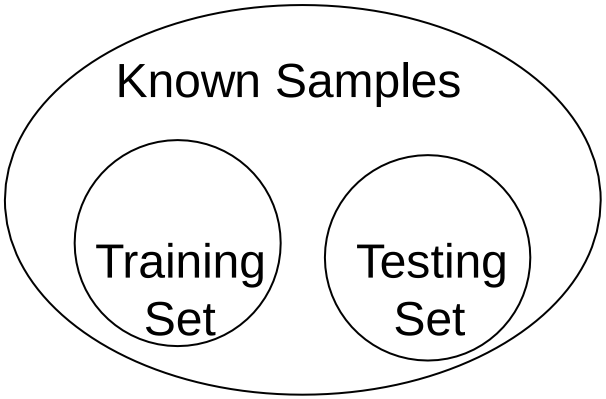
- For most problems, one layer is sufficient
- Two layers are required when the function is discontinuous
- The number of neurons is very important:
  - Too few
    - Underfit the data – NN can't learn the details
  - Too many
    - Overfit the data – NN learns the insignificant details (poor generalization)
  - Start small and increase the number until satisfactory results are obtained

Verification

- Provides an unbiased test of the quality of the network
- Common error is to test the neural network using the same samples that were used to train the neural network
  - The network was optimized on these samples, and will obviously perform well on them
  - Doesn't give any indication as to how well the network will be able to classify inputs that weren't in the training set

Training / Testing set

- The set of all known samples is broken into two independent sets:
  - Training set
    - A group of samples used to train the neural network
  - Testing set
    - A group of samples used to test the performance of the neural network
    - Used to estimate the error rate

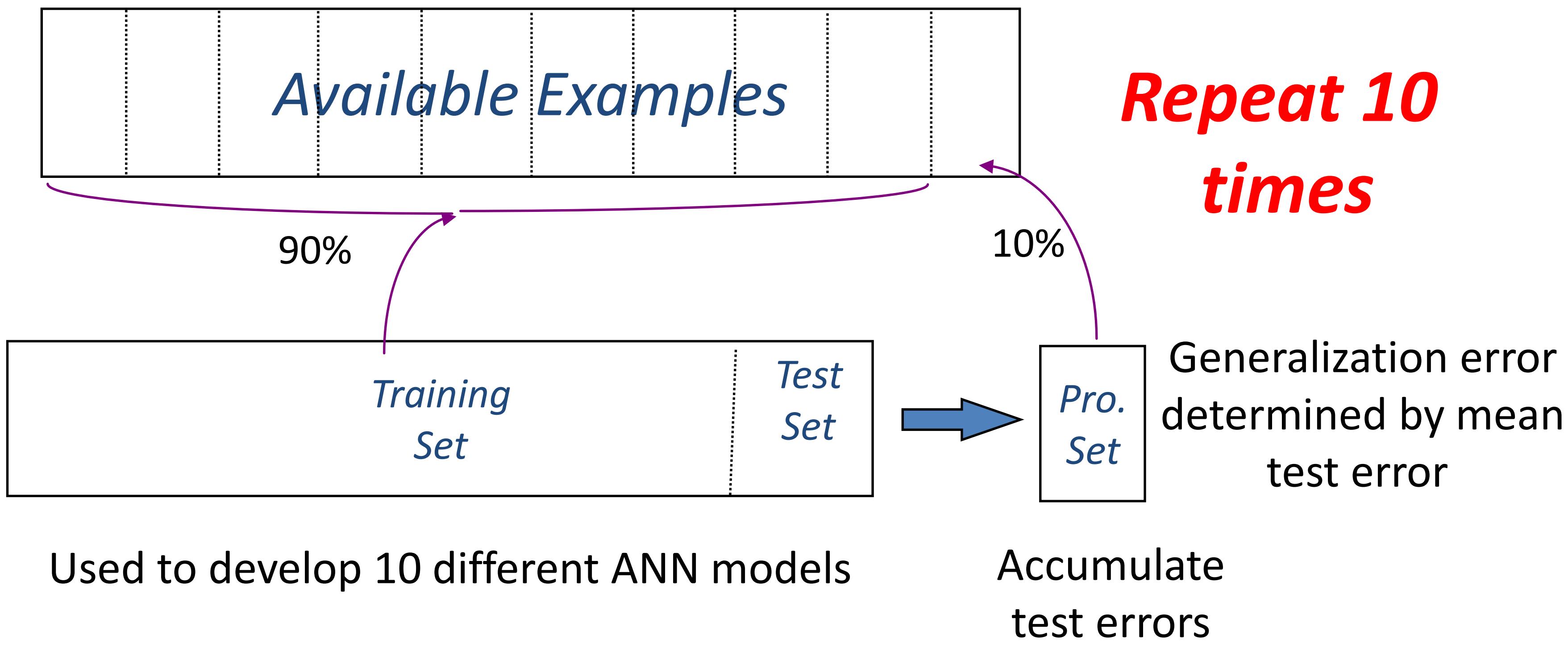


Training Set

- Good training set can avoid Overfitting
- What is the good training set?
  - Samples must represent the general population
  - Samples must contain members of each class
  - Samples in each class must contain a wide range of variations or noise effect
- The size of the training set is related to the number of hidden neurons
  - Eg. 10 inputs, 5 hidden neurons, 2 outputs:
  - 11(5) + 6(2) = 67 weights (variables)
  - If only 10 training samples are used to determine these weights, the network will end up being overfit
    - Any solution found will be specific to the 10 training samples
    - Having 10 equations, 67 unknowns, you can come up with a specific solution, but you can't find the general solution with the given information

Cross-validation

When the amount of available data is small ...



Data Preparation

- The quality of results relates directly to quality of the data
- Eliminate or estimate missing values
- Remove outliers (obvious exceptions)
- Reduce attribute dimensionality
  - remove redundant and/or correlating attributes
  - combine attributes (sum, multiply, difference)
- **Normalization**: De-correlate example attributes via normalization of values:
  - Euclidean: *n = x/sqrt(sum of all x^2)*
  - Percentage: *n = x/(sum of all x)*
  - Variance based: *n = (x - (mean of all x))/variance*

Weights initialization

- Random initial values +/- some range
- Smaller weight values for nodes with many incoming connections
- Rule of thumb: initial weight range should be approximately

$$\pm \frac{1}{\# weights}$$

coming into a node

- Genetic Algorithm

Training FFNN using Back propagation

1. Randomly initialize weights.
2. Apply feed forward pass with one input record.
3. Calculate the total error on output layer.
   – If total error <= accepted value then *stop*
   – Else *continue*
4. Apply backward pass.
5. Update weights.
6. Go to step 2.

➢ When all records of training are attempted, then one epoch is done.

➢ Usually thousands of epochs are executed to get the final weights for a trained network