Different Approaches - More Detail

Pattern Classification

Statistical Approach
- Supervised
  - Basic concepts: Bayesian decision rule (MPP, LL, DCM)
  - Parametric learning (ML, BL)
  - Non-Parametric learning (KNN)
  - NN (Perceptron, BP)

Syntactic Approach
- Unsupervised
  - Basic concepts: Distance Agglomerative method
  - k-means
  - Winter-take-all
  - Kohonen maps

Dimensionality Reduction
- Fisher’s linear discriminant
- KL transform (PCA)

Performance Evaluation
- ROC curve
- TP, TN, FP

Stochastic Methods
- Local optimization (GD)
- Global optimization (SA, GA)

Perceptron

\[ z = \begin{cases} 1 & \text{if } w^T x + w_0 > 0 \\ 0 & \text{otherwise} \end{cases} \]
Perceptron

- A program that learns “concepts” based on examples and correct answers
- It can only respond with “true” or “false”
- Single layer neural network
- By training, the weight and bias of the network will be changed to be able to classify the training set with 100% accuracy

Perceptron Criterion Function

\[ J_p(a) = \sum_{y \in \text{Y}} (-a^T y) \]
\[ \nabla J_p = \sum_{y \in \text{Y}} (-y) \]
\[ a(k+1) = a(k) + \eta(k) \sum_{y \in \text{Y}} y \]
Gradient descent learning

Perceptron Learning Rule

\[ w(k+1) = w(k) + \eta(T - z) \]
\[ -w_0(k+1) = w_0(k) + \eta(T - z) \]
T is the expected output
z is the real output
Training

- Step 1: Samples are presented to the network
- Step 2: If the output is correct, no change is made; Otherwise, the weight and biases will be updated based on perceptron learning rule
- Step 3: An entire pass through all the training set is called an “epoch”. If no change has been made for the epoch, stop. Otherwise, go back Step 1

Exercise (AND Logic)

\begin{align*}
&x_1 & x_2 & T & w_1 & w_2 & -w_0 \\
&0 & 0 & 0 & & & \\
&1 & 0 & 0 & & & \\
&0 & 1 & 0 & & & \\
&1 & 1 & 1 & & & \\
\end{align*}

% demonstration on perceptron
% AND gate
% Hairong Qi
input = 2;
tr = 4;
w = rand(1,input+1); % generate the initial weight vector
x = [0 0; 1 0; 0 1; 1 1]; % the training set (or the inputs)
T = [0; 0; 0; 1]; % the expected output
% learning process (training process)
finish = 0;
while ~finish
    disp(w)
    for i=1:tr
        z(i) = w(1:input) * x(i,:)’ > w(input+1);
        w(1:input) = w(1:input) + (T(i)-z(i))* x(i,:);
        w(input+1) = w(input+1) - (T(i)-z(i));
    end
    if sum(abs(z-T')) == 0
        finish = 1;
    end
    disp(z)
pause
end
disp(w)
Limitations

- The output only has two values (1 or 0)
- Can only classify samples which are linearly separable (straight line or straight plane)
- Can’t train network functions like XOR

Examples