Pattern Recognition

Non-Bayesian Classifiers
Motivation

• In probabilistic classifiers the distribution parameters should be estimated (ex. by using training data)

• Can we develop techniques that use training data to learn the classifiers directly without estimating any probabilistic structure?
Some of the Available Methods

• K-Nearest Neighbor
• Linear discriminant functions
• Neural networks
• Decision trees
Nearest Neighbor (NN)

• Given the training data $D = \{x_1, \ldots, x_n\}$ as a set of $n$ labeled examples, the nearest neighbor classifier assigns a test point $x$ the label associated with its closest neighbor in $D$.

• The nearest neighbor classifier partitions the feature space into cells consisting of all points closer to a given training point than to any other training points.
Nearest Neighbor (NN)

Nearest neighbor classifier forms a Voronoi tessellation of the feature space.
K Nearest Neighbor (k-NN)

- The k-nearest neighbor classifier classifies $x$ by assigning it the label most frequently represented among the k nearest samples.
K Nearest Neighbor (k-NN)

• The nearest neighbor classifier relies on a metric or a distance function between points.

• For all points \( x, y \) and \( z \), a metric \( D(\cdot, \cdot) \) must satisfy the following properties:
  
  – Nonnegativity: \( D(x, y) \geq 0 \).
  
  – Reflexivity: \( D(x, y) = 0 \) if and only if \( x = y \).
  
  – Symmetry: \( D(x, y) = D(y, x) \).
  
  – Triangle inequality: \( D(x, y) + D(y, z) \geq D(x, z) \).
Distance Metrics

• Minkowski metric also referred to as the $L_p$ norm

\[ L_p(x, y) = \left( \sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p} \]
Distance Metrics

The Euclidean distance

$$L_2(x, y) = \left( \sum_{i=1}^{d} |x_i - y_i|^2 \right)^{1/2}.$$
Distance Metrics

The Manhattan or city block distance

\[ L_1(x, y) = \sum_{i=1}^{d} |x_i - y_i|. \]
Distance Metrics

The $L_\infty$ norm is the maximum of the distances along individual coordinate axes

$$L_\infty(x, y) = \max_{i=1}^{d} |x_i - y_i|.$$
Feature Normalization

- Scaling of the coordinate axes is important when we use these metrics.
- Feature normalization can be used to approximately equalize ranges of the features and make them have approximately the same effect in the distance computation.
Feature Normalization

• Linear scaling to unit range:
  Given a lower bound $l$ and an upper bound $u$ for a feature $x$

\[
\tilde{x} = \frac{x - l}{u - l}
\]
Feature Normalization

• Linear scaling to unit variance:
  A feature $x \in \mathbb{R}$ can be transformed to a random variable with zero mean and unit variance as

$$\tilde{x} = \frac{x - \mu}{\sigma}$$
Linear Discriminant Functions

A discriminant function that is a linear combination of the components of $x$ is called a linear discriminant function and can be written as

$$g(x) = w^T x + w_0$$

where $w$ is the weight vector and $w_0$ is the bias (or threshold weight).
The Two-Category Case

Decide \( \begin{cases} w_1 & \text{if } g(x) > 0 \\ w_2 & \text{otherwise} \end{cases} \)

- The equation \( g(x) = 0 \) defines the decision boundary that separates points assigned to \( w_1 \) from points assigned to \( w_2 \).
- When \( g(x) \) is linear, the decision surface is a hyperplane whose orientation is determined by the normal vector \( w \) and location is determined by the bias \( w_0 \).
The Multi-Category Case

• $c(c-1)/2$ linear discriminants, one for every pair of classes.

• $c$ linear discriminants, one for each class. Assign $x$ to $w_i$ if $g_i(x) > g_j(x)$ for all $j <> i$. 
The Multi-Category Case
The Multi-Category Case
Generalized Linear Discriminant Functions

• The linear discriminant function $g(x)$ can be written as

$$g(x) = w_0 + \sum_{i=1}^{d} w_i x_i$$

• The quadratic discriminant function is obtained by adding second-order terms as

$$g(x) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_i x_j$$
Generalized Linear Discriminant Functions

![Diagram of Generalized Linear Discriminant Functions]
Neural Networks

• A neural network consists of an input layer, an output layer, and usually one or more hidden layers.

• The layers 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 based on the properties of biological neurons, and hence they are called neurons
Neural Networks
Feed forward Operation

• Each hidden unit computes the weighted sum of its inputs to form a net activation value, \( \text{net} = w^T x \).

• The output is a nonlinear function of the activation function, \( f(\text{net}) \).

• Each output unit similarly computes its net activation based on the hidden unit signals in the previous layer, and provides a value using a nonlinear function based on its activation.
Activation functions

\[ H[n] = \begin{cases} 
0, & n < 0 \\
1, & n \geq 0 
\end{cases} \]

\[ H(x) \approx \frac{1}{2} + \frac{1}{2} \tanh(kx) = \frac{1}{1 + e^{-2kx}} \]
Training the Network

• By training a Neural Network, we mean adjusting the weights of the links connecting the output of a layer as input to the next layer neurons.

• For updating the weights of the hidden layers backpropagation algorithm is used.
Back propagation Algorithm

• The training error on a pattern is defined as the sum of the output units of the squared difference between the desired output $t_k$ and the actual output $z_k$ as ($c$ is the number of outputs)

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

• 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 \( \eta \) is the learning rate.
Assignment

• Design and train a neural network with 3 layers (input, output and hidden layer) to recognize 4 letters A, C, D and Z
• Define each letter as an 8x8 binary matrix
• Try and verify the performance of your network with noisy data