# Pattern Recognition

K-Nearest Neighbor

Linear Discriminant Functions

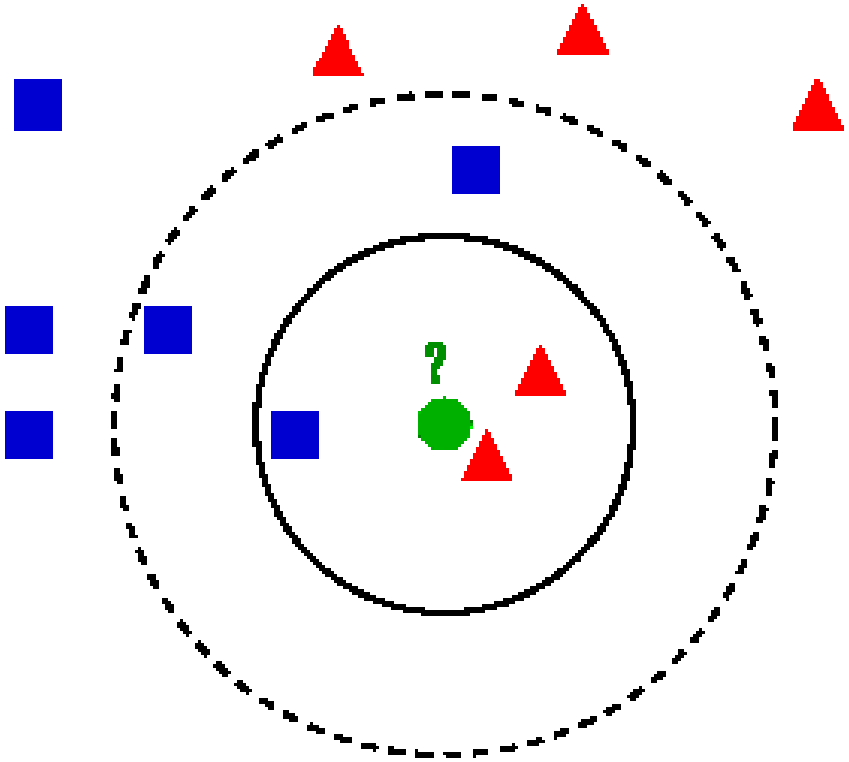
# k-Nearest Neighbors Classifier

- The $k_n$ nearest neighbor (KNN) density estimate is given by:

$$p_n(\mathbf{x}) = \frac{k_n}{nV_n},$$

- $V_n$ is the volume of the smallest possible x centered cell that contains $k_n$ training samples, and n is the total number of training samples

# K-Nearest Neighbor Example



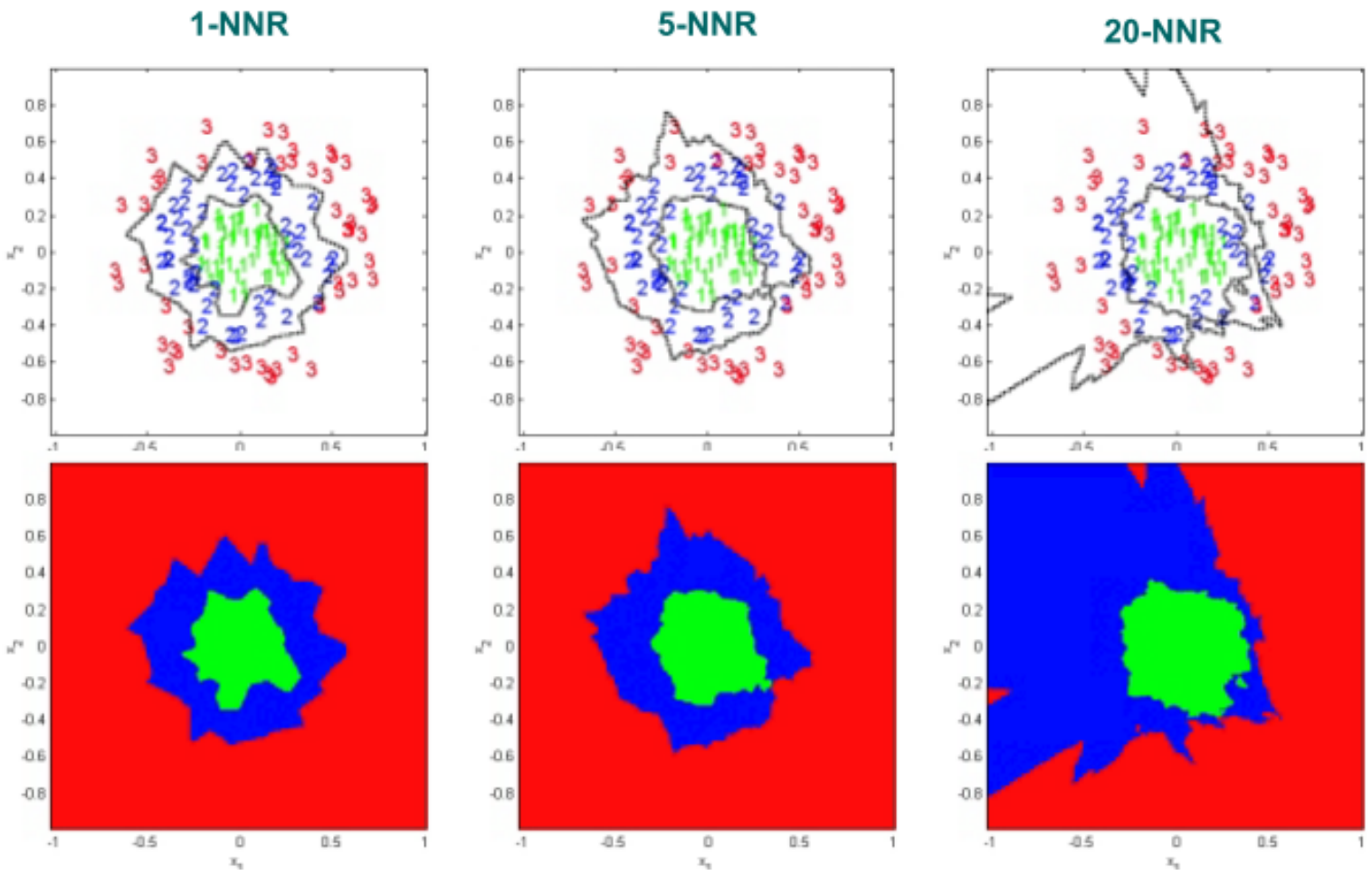
# Disadvantages

- The distance to all sample points should be computed at each classification. This computation can be very time consuming
- The accuracy of the *k*-NN algorithm can be severely degraded by the presence of noisy or irrelevant features.

# Optimizing K Parameter

- **1-NNR versus k-NNR**
  - The use of large values of k has two main advantages
    - Yields smoother decision regions
    - Provides probabilistic information; The ratio of examples for each class gives information about the ambiguity of the decision
  - However, too large values of k are harmful; They destroy the locality of the estimation since farther examples are taken into account
  - In addition, it increases the computational burden

# Example



# Distance Metrics

Requirements for a distance metric:

1. $L(a, b) \geq 0$
2. $L(a, b) = 0$ if and only if $a = b$.
3. $L(a, b) = L(b, a)$
4. $L(a, b) + L(b, c) \geq L(a, c)$.

# Sample Distance Metrics

- Euclidean metric

$$L(\mathbf{a}, \mathbf{b}) = ||\mathbf{a} - \mathbf{b}|| = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}.$$

- Minkowski metrics

$$L_m(\mathbf{a}, \mathbf{b}) = (\sum_{i=1}^{d} (|a_i - b_i|^m)^{1/m}.$$

- L-infinity metric

$$L_\infty(\mathbf{a}, \mathbf{b}) = \max_i |a_i - b_i|.$$

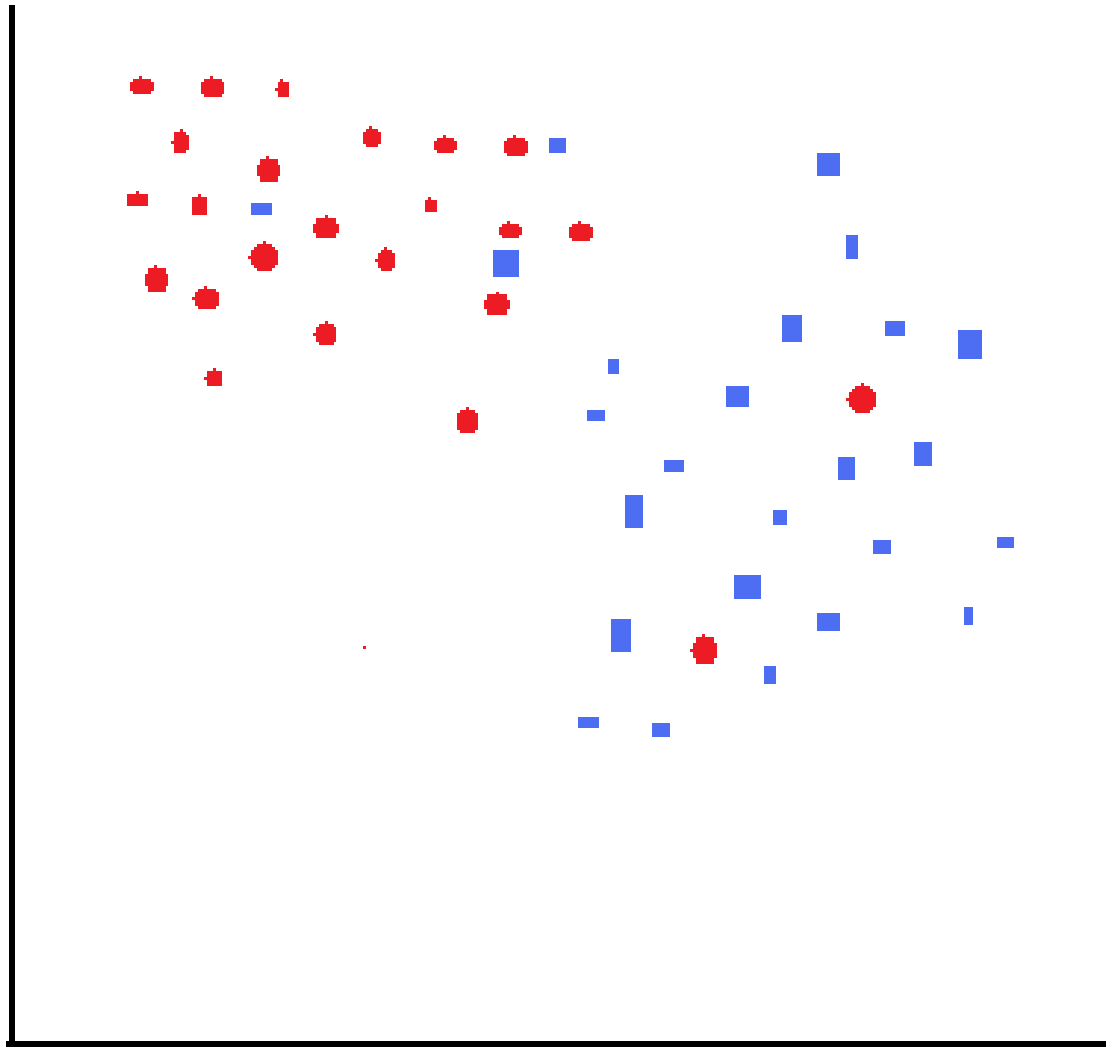
- Mahalanobis-distance

$$L_{Mahalanobis,C}(\mathbf{a}, \mathbf{b}) = \sqrt{(\mathbf{a} - \mathbf{b})^T C^{-1} (\mathbf{a} - \mathbf{b})}$$

# Improving KNN Classifier

- Classify all the examples in the training set and remove those examples that are misclassified, in an attempt to separate classification regions by removing ambiguous points
- The opposite alternative is to remove training examples that are classified correctly, in an attempt to define the boundaries between classes by eliminating points in the interior of the regions
- A different alternative is to reduce the training examples to a set of prototypes that are representative of the underlying data.
- [The issue of selecting prototypes will be the subject of the lectures on clustering]

# Example



# Improving KNN Search Algorithm

## Bucketing

- In the Bucketing algorithm, the space is divided into identical cells and for each cell the data points inside it are stored in a list.
- The cells are examined in order of increasing distance from the query point and for each cell the distance is computed between its internal data points and the query point
- The search terminates when the distance from the query point to the cell exceeds the distance to the closest point already visited

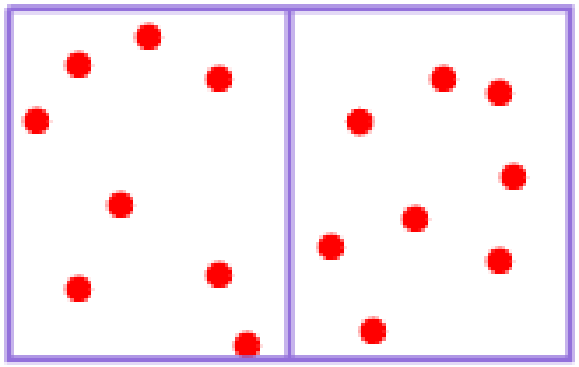
# Improving KNN Search Algorithm

## k-d trees

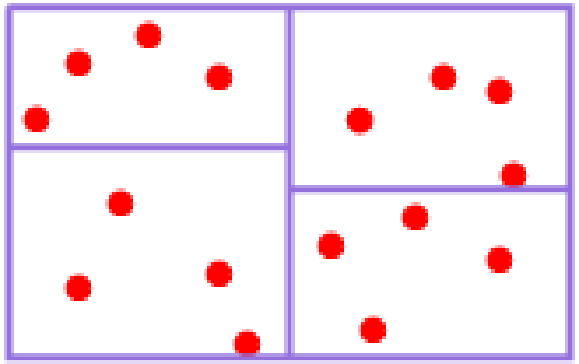
- A k-d tree is a generalization of a binary search tree in high dimensions. Each internal node in a k-d tree is associated with a hyper-rectangle and a hyper-plane orthogonal to one of the coordinate axis
- The hyper-plane splits the hyper-rectangle into two parts, which are associated with the child nodes
- The partitioning process goes on until the number of data points in the hyper-rectangle falls below some given threshold
- The effect of a k-d tree is to partition the (multi-dimensional) sample space according to the underlying distribution of the data, the partitioning being finer in regions where the density of pints is higher
- For a given query point, the algorithm works by first descending the tree to find the data points lying in the cell that contains the query point
- Then it examines surrounding cells if they overlap the ball centered at the query point and the closest data point so far

# K-d Tree Example

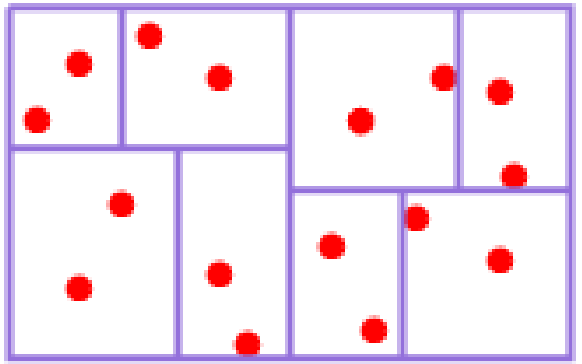
1.



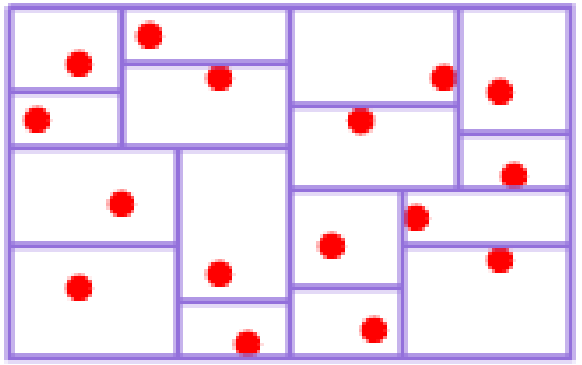
2.



3.



4.



# Discriminant Function

- For each class, there exists a discriminant function $g_i$ , i = 1, . . . , c whose input is a feature vector x. The sample identified by the feature vector x is assigned to $\omega_i$ if

$$g_i(\mathbf{x}) > g_j(\mathbf{x})$$

for all j = 1, . . . , c and i ≠ j

# Linear Classifiers

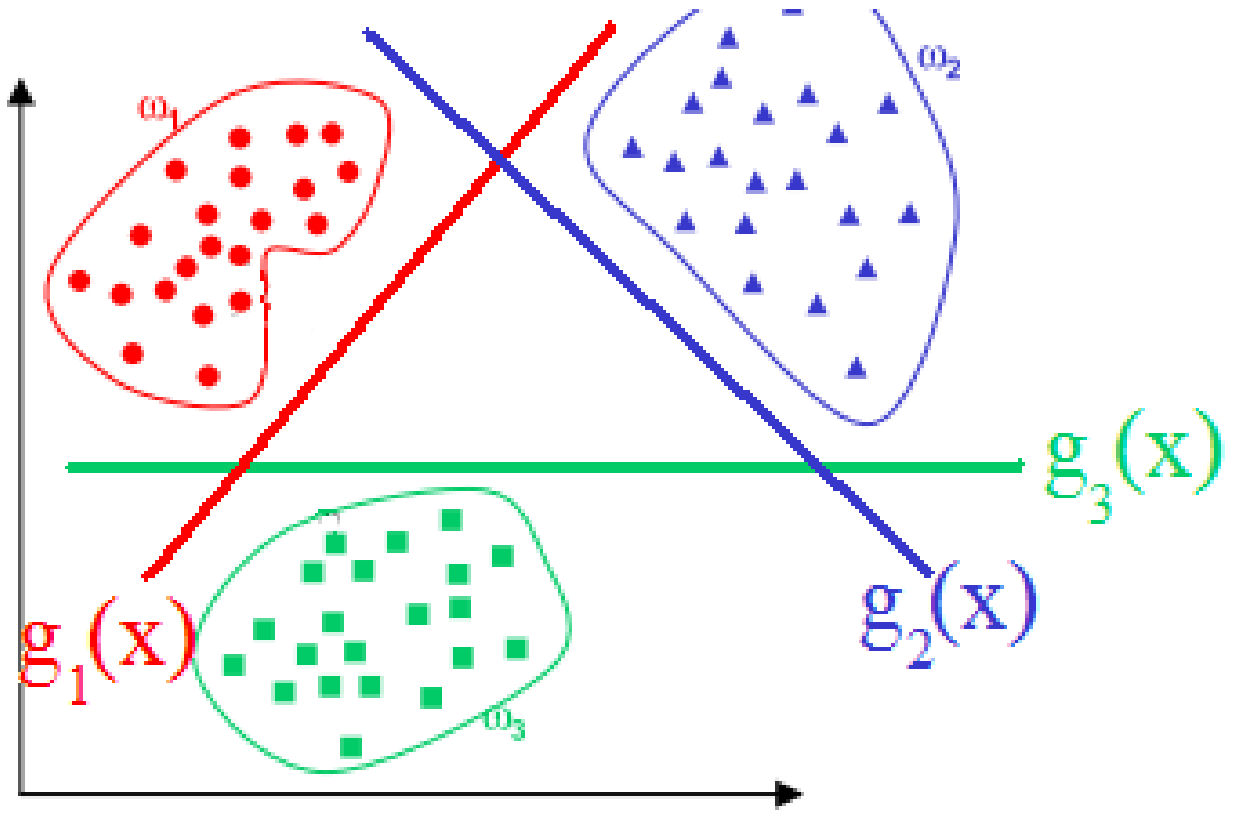
- A discriminant function is said to be linear if it can written as:

$$g_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + w_{i0} = \sum_{j=1}^{d} w_{ij} x_j + w_{i0},$$

where $w_i = [w_{i1}, \ldots, w_{id}]^T$ is the weight vector and the scalar $w_{i0}$ is threshold weight

The classifier relying only on the linear discriminant functions is called linear classifier

# Example: Linear Classifier



# Linearly Separable Training Samples

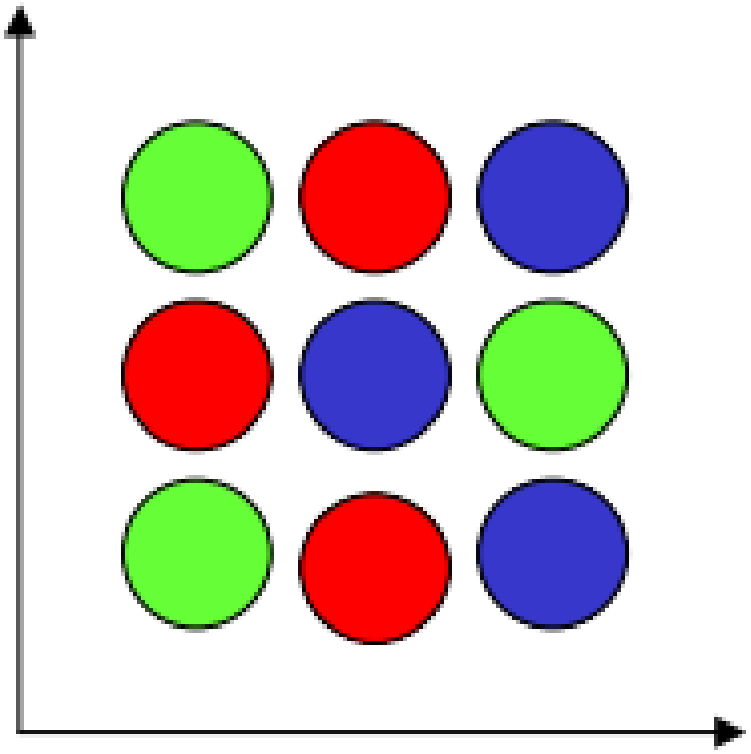
If there exists a linear classifier that classifies all the training samples correctly, i.e.

$g(x_{1j}) > 0$, for all $j = 1, \ldots, n_1$ and

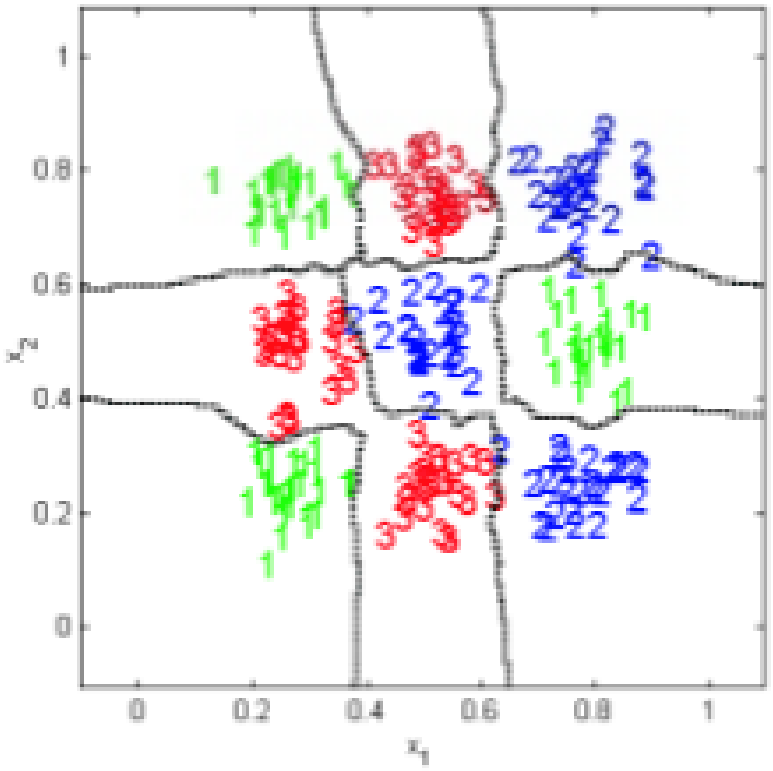
$g(x_{2j}) < 0$ for all $j = 1, \ldots, n_2$

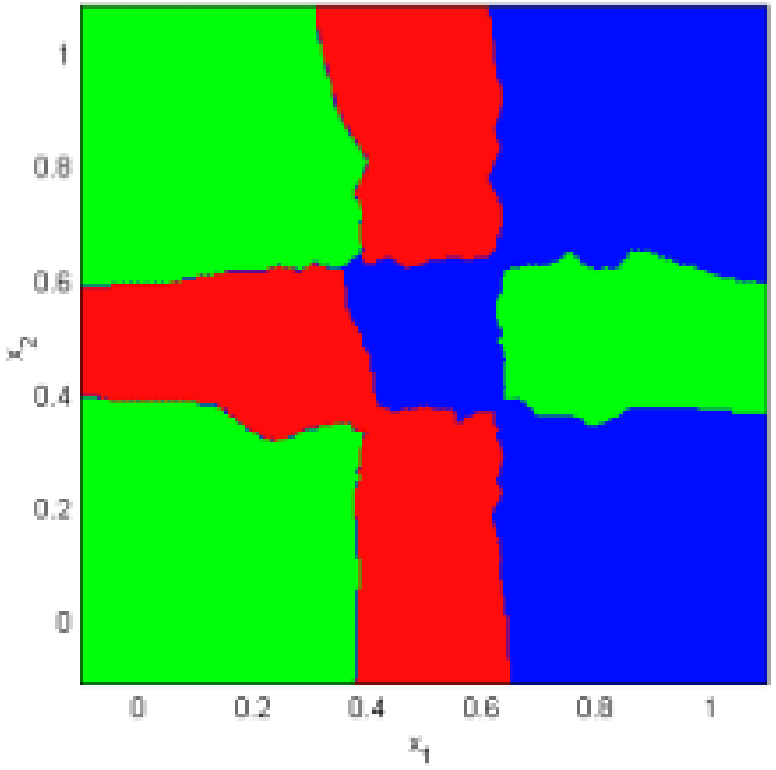
Then we say that the training sets/samples $D_1$ and $D_2$ are linearly separable.

# Linearly Inseparable Sets (Example)

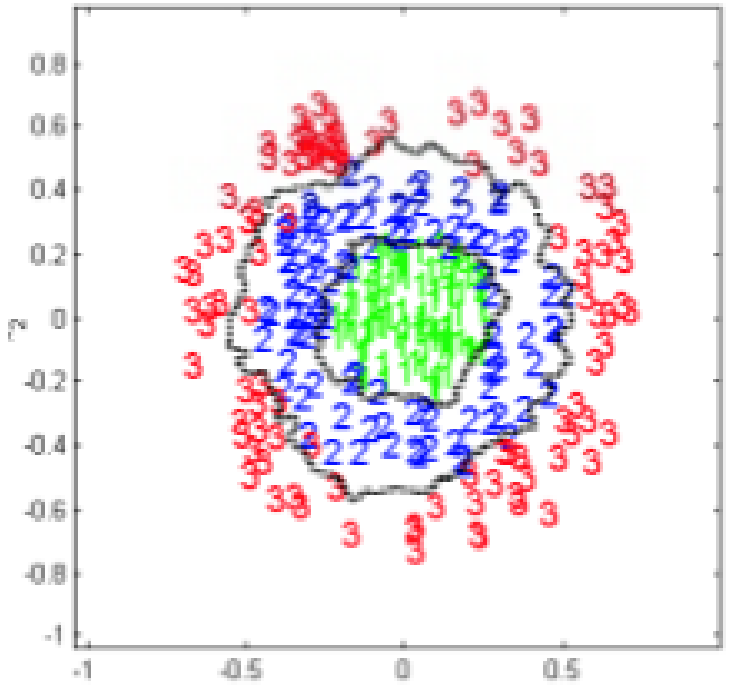


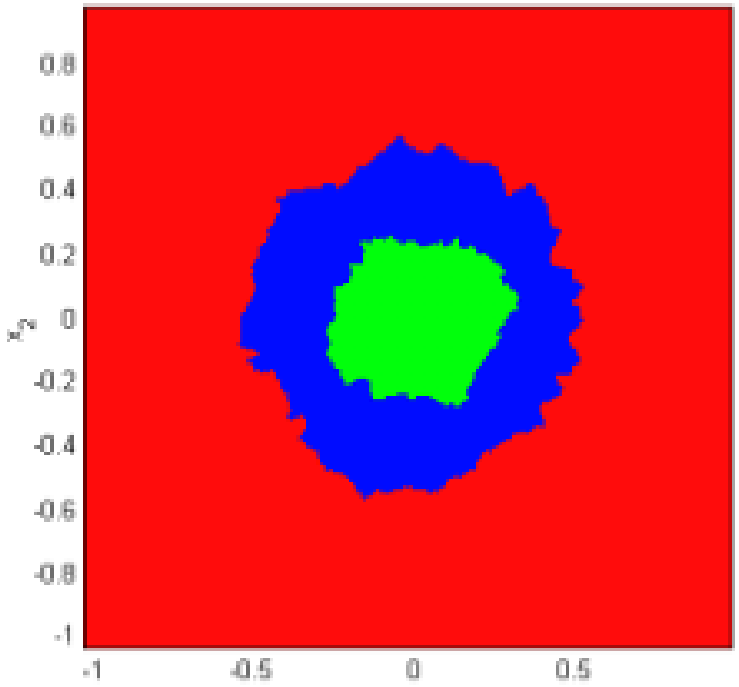
# Non-Linear Classification (Example 2)



# Non-Linear Classification (Example 1)

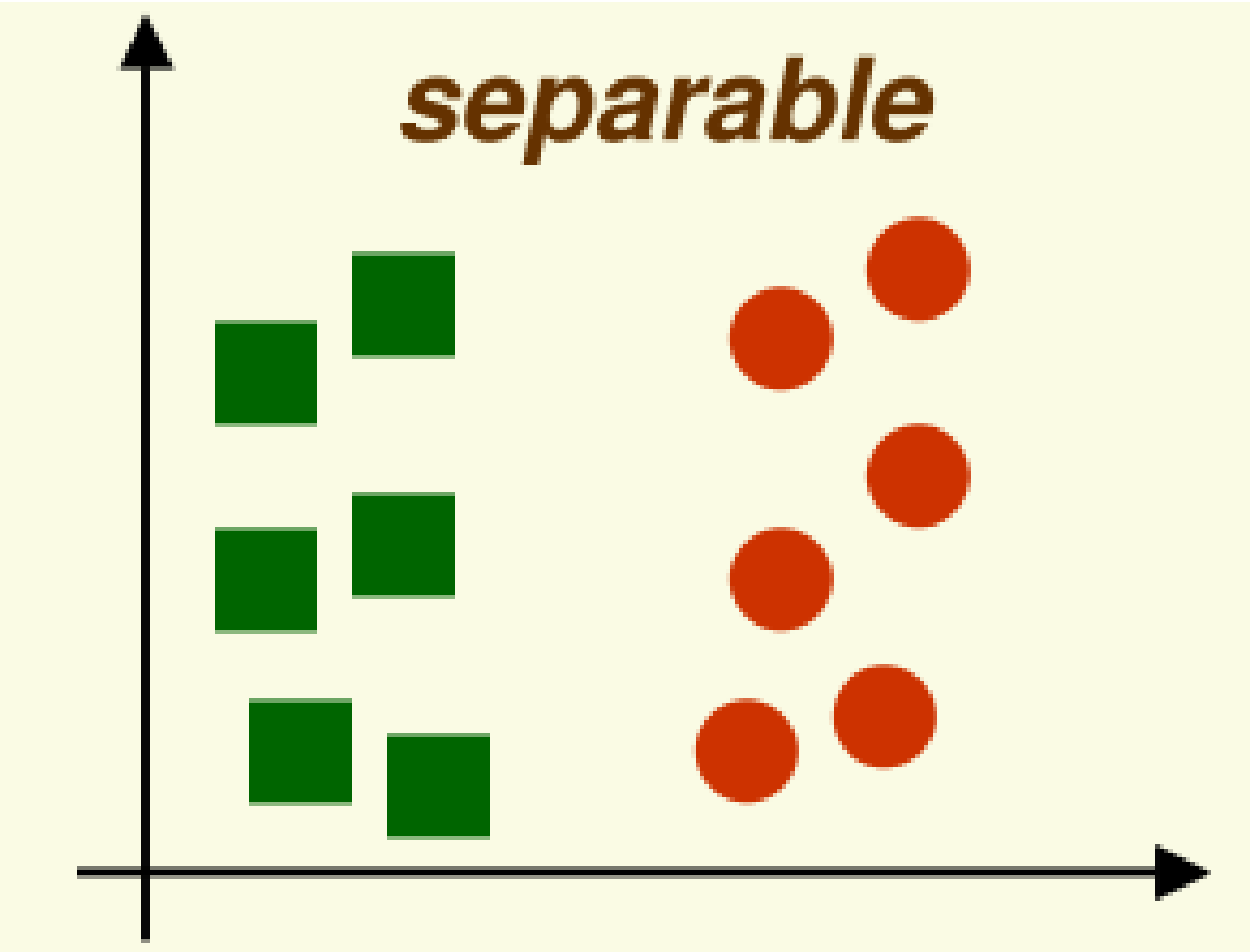


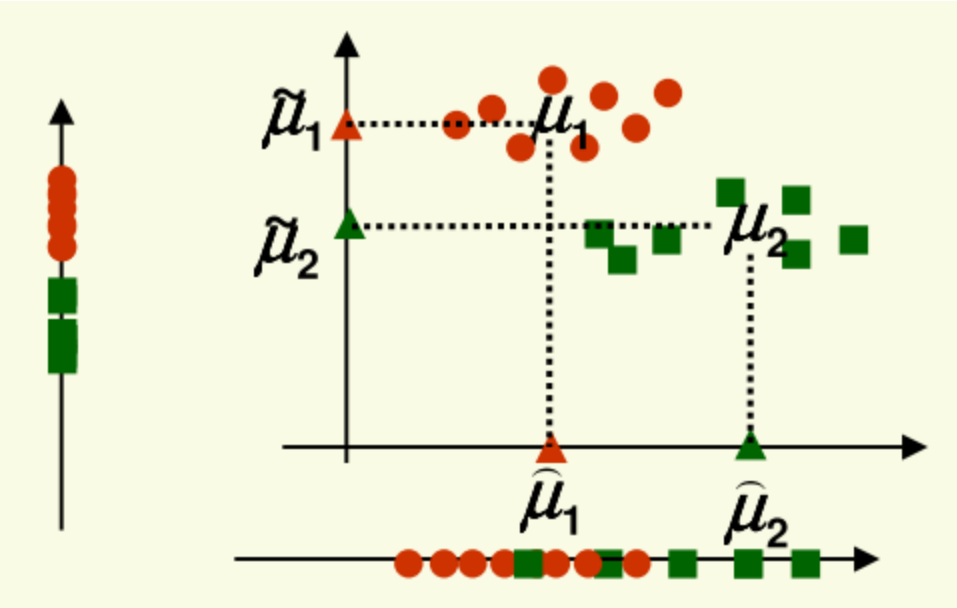
# Fisher Linear Classifier

- Problem definition: Classification based on thresholding in feature space cannot separate classes in many cases.
- Instead on mapping feature values on the feature vector axes, a different line can be used (Linear classification is assumed)

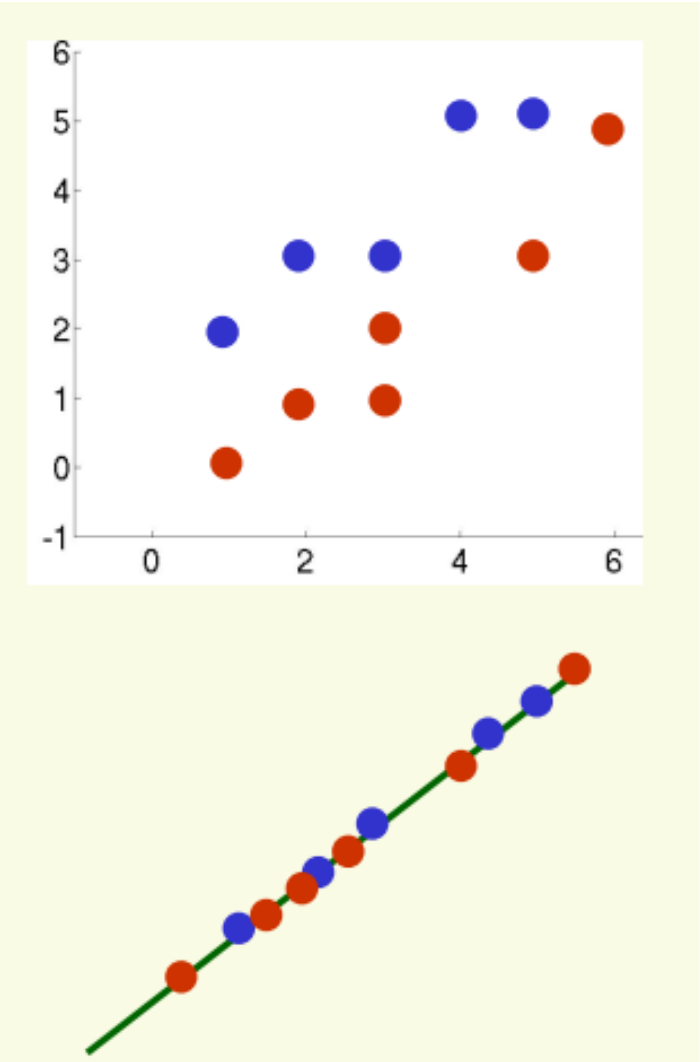
# Fisher Linear Classifier



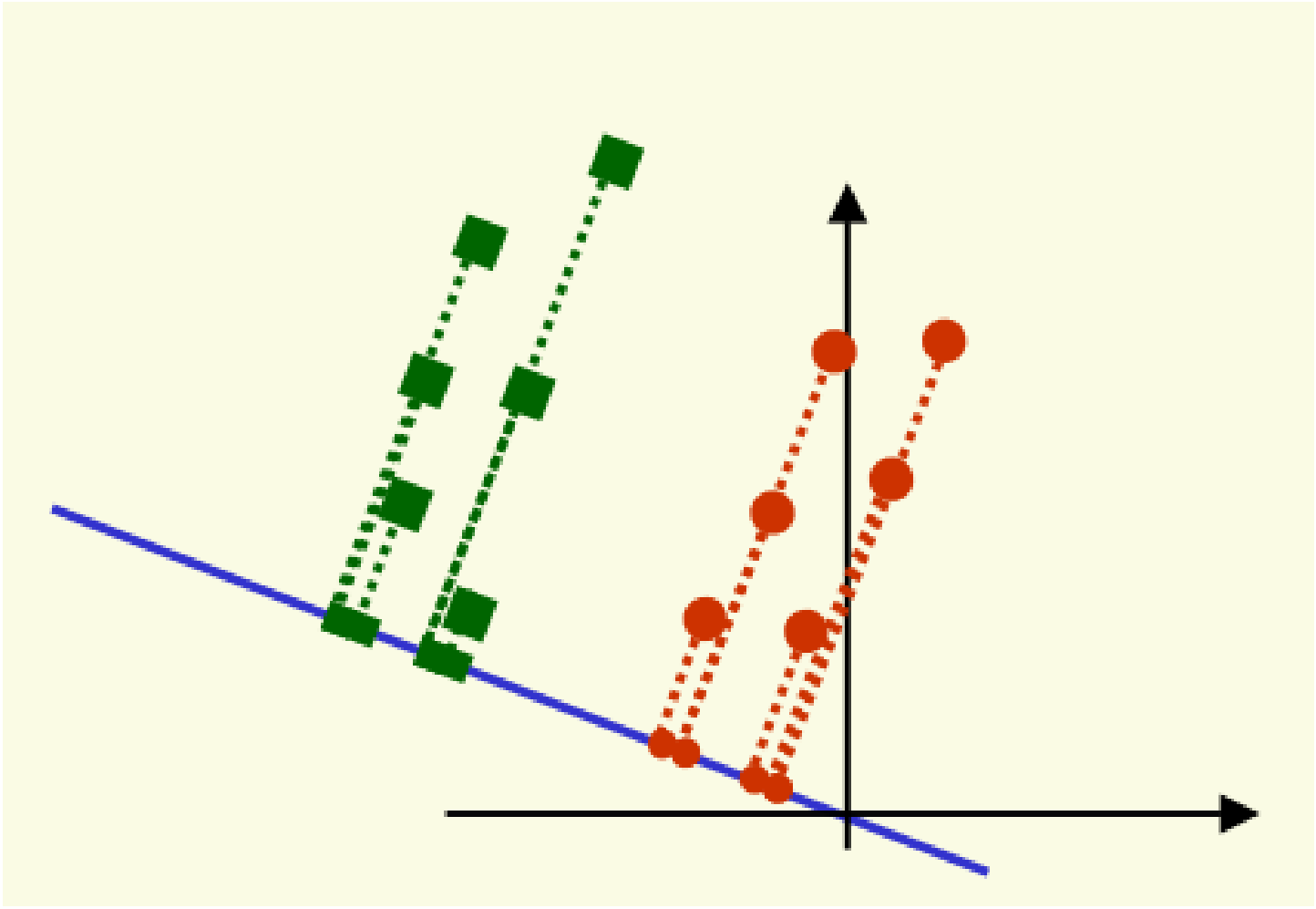
# Fisher Linear Classifier



# Fisher Linear Classifier



# Fisher Linear Classifier



# Fisher Linear Classifier

- Fisher Linear Classifier tries to maximize between class scatter, and minimize within class scatter.

- where

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

$$S_B = \sum_c (\boldsymbol{\mu}_c - \bar{\mathbf{x}})(\boldsymbol{\mu}_c - \bar{\mathbf{x}})^T$$

$$S_W = \sum_c \sum_{i \in c} (\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^T$$

# Fisher Linear Classifier

- W is obtained from

$$w^* = \arg\max \left[ \frac{w^T S_B w}{w^T S_W w} \right] = S_W^{-1}(\mu_1 - \mu_2)$$

# Example

$$X1 = \{(4,1), (2,4), (2,3), (3,6), (4,4)\}$$
$$X2 = \{(9,10), (6,8), (9,5), (8,7), (10,8)\}$$

$$S_1 = \begin{bmatrix} .8 & -.4 \\ & 2.64 \end{bmatrix} \quad S_2 = \begin{bmatrix} 1.84 & -.04 \\ & 2.64 \end{bmatrix}$$

$$\mu_1 = [3.0\ 3.6]^T; \quad \mu_2 = [8.4\ 7.6]^T$$

$$S_B = \begin{bmatrix} 29.16 & 21.6 \\ & 16.0 \end{bmatrix} \quad S_W = \begin{bmatrix} 2.64 & -.44 \\ & 5.28 \end{bmatrix}$$

$$w^* = S_W^{-1}(\mu_1 - \mu_2) = [-.91\ -.39]^T$$

# Reducing Feature Space Dimension

- Feature selection methods find a subset of the original features or attributes.
- In some cases, data analysis can be done in the reduced space more accurately than in the original space.
- Curse of dimensionality reduces the accuracy of the classifier
- Dimension reduction can be done once.
- Adaptive dimension reduction combines unsupervised learning with dimension reduction adaptively.

Questions?