ELEC 425

Introduction to
Machine Learning and Deep Learning

Nonparametric and Parametric Models
Linear Classifier
Perceptron

Xiaodan Zhu

ECE, Queen's University

Fall, 2025

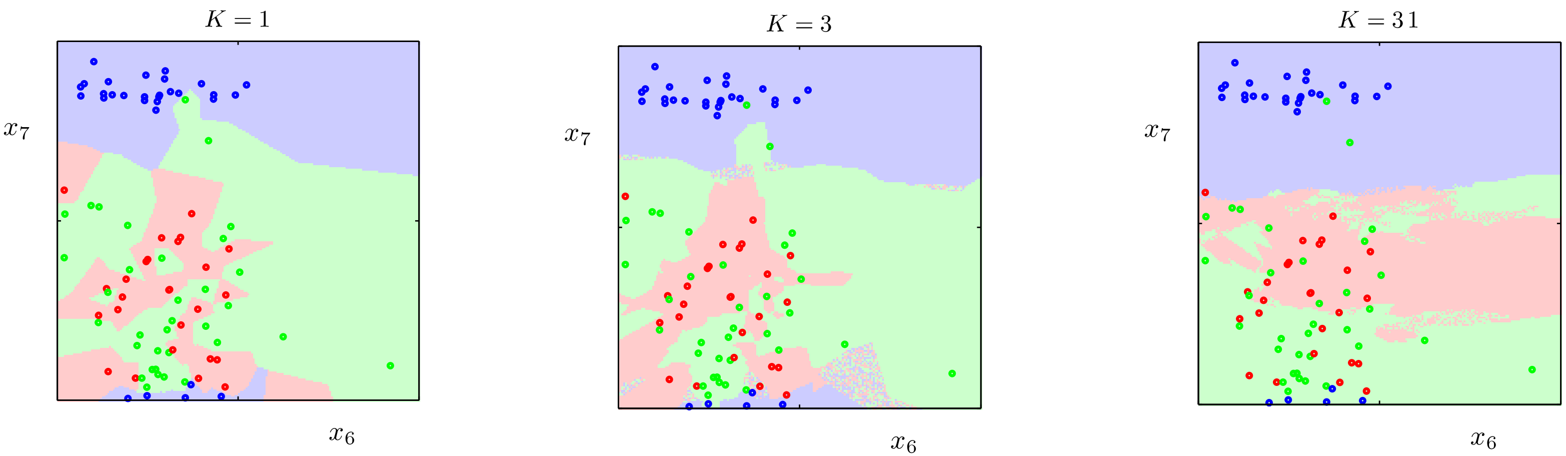
Misc.

- On your OnQ page, please subscribe to receive instant notifications via email or SMS.
- Install and start to play with Matlab on your laptop if you have not yet!!
- The first lab will be 6:30—8:30pm Sep. 10th
  - Room: Mitchell Hall 215/225/235
- Please send course-related emails to me using:

  elec425.instructor@queensu.ca

Last Lecture

- A specific type of classifier:
  K-Nearest Neighbor (KNN)



- Several basic machine learning concepts:
  - Datapoints, features, feature normalization
  - Training (e.g., finding decision boundaries/surfaces), validation, prediction/testing
  - Overfitting, underfitting
  - Cross validation

Revisit Classification: Types of Classification Models

- We have introduced several basic classification concepts, and a specific classification model: KNN.
- In general, there are different categories of ways to perform classification and view classification models.
  - Nonparametric vs. parametric models
  - Generative vs. discriminative models

Nonparametric Methods

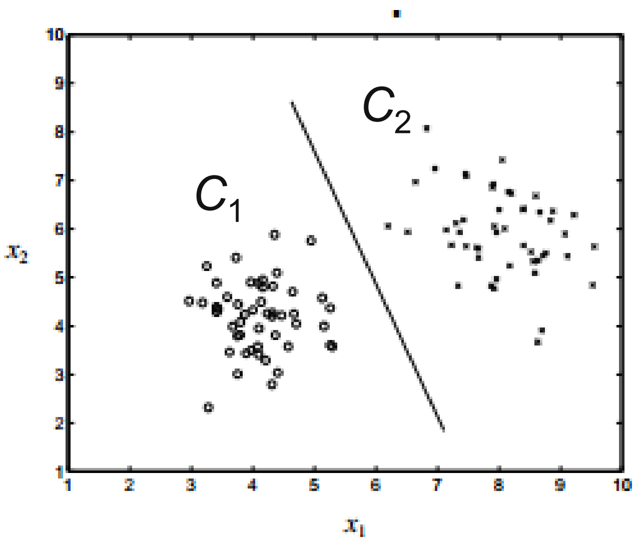
- Another name for instance-based or memory-based learning is **nonparametric methods**.
  - **Nonparametric methods** do **have** parameters, but the number of parameters is not fixed.
  - Parameters often grow with number of examples.
  - In KNN, the parameters are the entire training set, and we need the entire training set at test time.

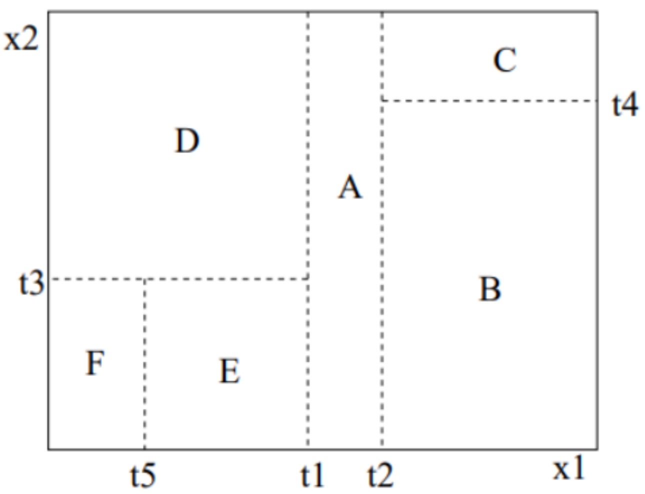
Parametric Methods

- On the other hand: **parametric methods** *or* **model-based methods** summarize data with <u>a set of parameters of fixed size</u> (independent of the number of training examples).
- No matter how much training data you feed into a parametric model, it won't change how many parameters it has.

Parametric Methods

- Two basic examples for parametric models
  - Example 1: Single linear boundary, of arbitrary orientation (**controlled by a set of parameters**)
    - Can be extended to form very powerful models.
  - Example 2: Many boundaries, but axis-parallel & tree structured

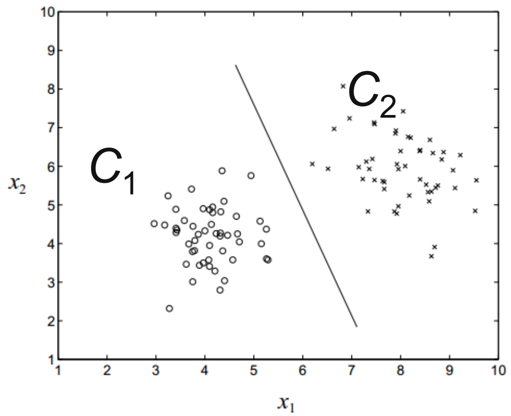




Note that for both types of classifiers, the number of parameters are fixed.

Linear Classifiers

- Objective: find the line (or hyperplane) which can "best" (under some criterion/objective) separate two classes:



$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

- The vector $\mathbf{w}$ controls the decision boundary (it is a vector perpendicular to the decision boundary); ω_0 is called *bias*.
- This model has fixed d + 1 parameters! (d is the dimension of a data point, i.e., number of features)
- There are many different criteria/objects to place the hyperplane.

Detailed Geometry of a Linear Decision Boundary

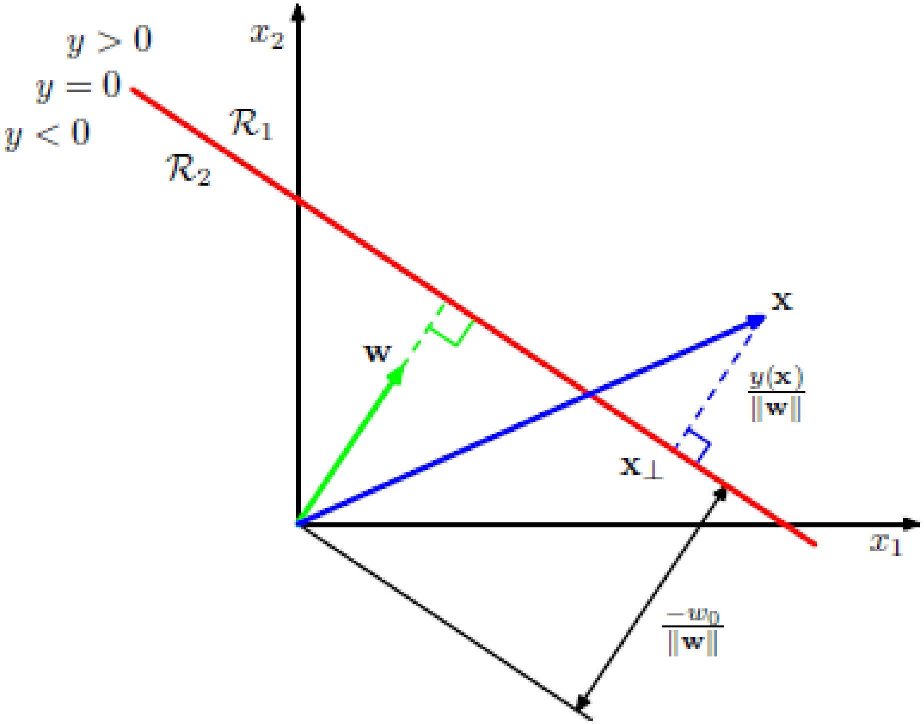


Figure 4.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$.

(Figure 4.1 of the Bishop textbook.)

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

- $\mathbf{w}$ is a vector perpendicular to decision boundary.

  (Consider two points x_A and x_B lying on the decision surface and the fact $w^T(x_A - x_B) = 0$)

- The distance from **x** to the decision surface is given by $|y(\mathbf{x})|/\|\mathbf{w}\|$

- The distance from the origin to the decision surface is given by $|w_0|/\|\mathbf{w}\|$.

* **Read section 4.1.1 of the Bishop textbook for details.**

Detailed Geometry of a Linear Decision Boundary

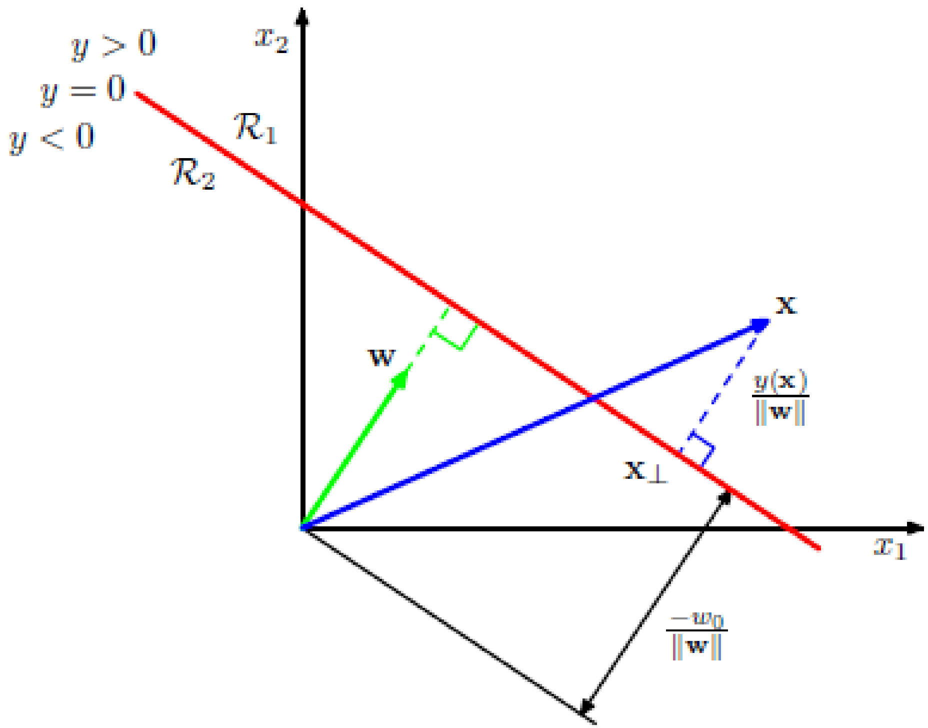


Figure 4.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$.

(Figure 4.1 of the Bishop textbook.)

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

We can assign an input vector **x** to class C_1 if $y(\mathbf{x}) > 0$ and to class C_2 otherwise.

* **Read section 4.1.1 of the Bishop textbook for details.**

Bias

- Note that we can augment $\mathbf{x}$ and $\mathbf{w}$ (both are column vectors) to absorb bias, so we don't have to treat bias separately as follows:

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\,\mathbf{x} + w_{\mathbf{0}}$$

- Instead, the above equation can be simply written as

$$y(\mathbf{x}) = \widetilde{\boldsymbol{w}}^{\mathrm{T}}\tilde{\mathbf{x}}$$

where $\widetilde{\boldsymbol{w}}^{\mathrm{T}} = (\mathbf{w}^{\mathrm{T}}, w_{\mathbf{0}})$ and $\tilde{\mathbf{x}} = (\mathbf{x}^{\mathrm{T}}, 1)^{\mathrm{T}}$,

or just written as $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\,\mathbf{x}$, when there is no confusion.

The Perceptron Classifier

- The perceptron uses the following algorithm to find the decision boundary.

```
PerceptronTrain(X, t){ // X: feature matrix; t: true labels (take a value of 1 or -1)
    w = small random values;
    do {
        errors = 0;
        for n =  0 : N-1 {   //loop through all training data points
            x = X (n) // get the feature vector of the n^th data point
            y(x) = f(w^T φ(x)) //φ(x) can simply just be x (with bias absorbed) or
                               a fixed non-linear transformation of x; f(.) is
                               a nonlinear activation function
```

$$f(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0 \end{cases}$$

```
            if (y(x) != t(n)){        //η is called learning rate
                w = w + η φ(x) t(n);
                errors ++;  }
        }
    } until (errors == 0)
    return w;
}
```