Week5 - Linear model for classification

1. Basic concept

Regression v.s Classification

- Regression: Modelling the relationship between input variables and one or more target variables.
- Classification: Modelling the relationship between input variables and one of variables.

Number of class:

- Two-class problem: binary representation
- For $K > 2$ class: 1-of-K coding scheme

Classifier

- An algorithm that implements classification mapping input data to a class (category)

Approaches for Classification problem

- 3 Approaches:
  - **Discriminant function** 判别方程:
    - Non-probabilistic models
    - takes an input vetor $\boldsymbol{x}$ and directly assign it to one of K classes
    - **Perceptron Algorithm**
  - **Probabilistic Discriminative Model** 概率判别模型:
    - Directly model the class posterior probability $p(C_k|\boldsymbol{x})$ for a given input $\boldsymbol{x}$
    - Model learns $\boldsymbol{w}$ from the training data
    - **Logistic Regression Model**
  - **Probabilistic Generative Model** 概率生成模型:
    - Learn the class-prior distribution $p(C_k)$ and the class-conditional distribution $p(x|C_k)$
    - Based on Bayes rule to get the class posterior $p(C_k|\boldsymbol{x})$.
- **Generative Model v.s Discriminative Model**:
  - Generative model can **generate synthetic data** whilst discriminative model can not
  - Generative model have **more parameters** than discriminative model (Generative model need to model the density of the input data)
  - **More parameter lead to more complexity**, prone to various problem (e.g. MLE leads to over-fitting)

Generalized linear models

For classification problem, the model is to predict discrete class labels. Hence, we need to transform the linear function of $\boldsymbol{w}$ using a nonlinear function $f(\cdot)$ which take the form:

$y(\boldsymbol{x}) = f(\boldsymbol{w}^T\boldsymbol{x} + w_0)$

where the function $f(\cdot)$ is known as activation function.

> The decision surfaces correspond to $y(\boldsymbol{x}) = constant$, so that $\boldsymbol{w}^T\boldsymbol{x} + w_0 = constant$ and hence the decision surfaces are linear function of $\boldsymbol{x}$, even if the function $f(\cdot)$ is nonlinear.

Activate functions

- Step function:
  - $f(u) = \begin{cases} 1 & u \geq 0 \\ 0 & u < 0 \end{cases}$
  - Using step function get the discetet value back as we need (e.g. perceptron)
- Logistic function:
  - $\sigma(a) := \dfrac{1}{1 + \exp(-a)}$
  - Mapping real values to the range $[0, 1]$, which allow us to interpret the output as a probability of belonging to a class (e.g. Logistic Regression)

Linearly separable Problems

- **Decision Boundary**: Based on the class labels, the input space is divided into decision regions where each region corresponds to a class label. There dicision regions are separated by the decision boundaries.
- **Hyperplane**: If the input space is D-dimensional, the decision boundary will be a $(D-1)$-dimensional hyperplane
- **Linear Separability**: the input space whose classes can be **separated by linear decision sufaces** are said to be **linearly-separable problems**.

---

1. Discriminant function

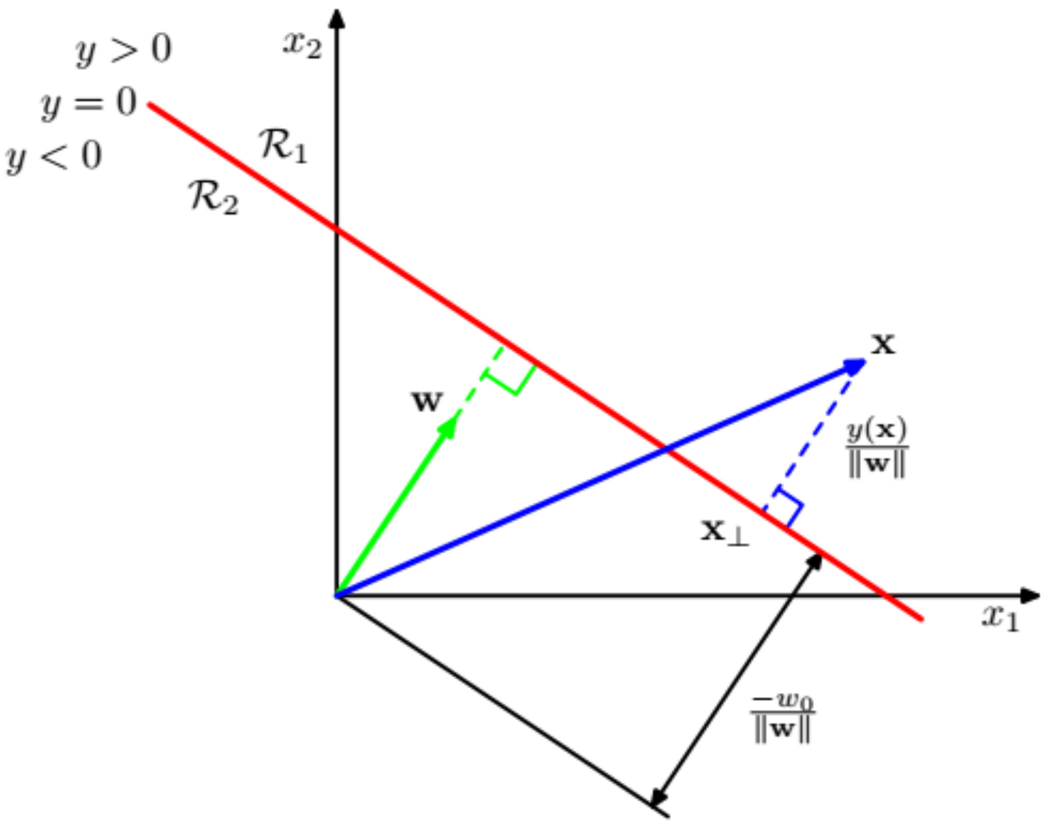
> Only focus on linear discriminant which the decision surface are hyperplanes
>
> 判断一个模型是否线性，只需要查看决策边界是否为直线，若为直线则为线性模型，否则为非线性模型

Two classes problem

The simplest representation of a linear discriminant function is obtained by taking a linear function of the input vector: $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$. $\boldsymbol{w}$ is called weight vector and w_0 is a bias (not the bias in the statistical sense) or threshold. Hence,

- If $y(\boldsymbol{x}) \geq 0$, the input vector $\boldsymbol{x}$ is assigned to class C_1
- If $y(\boldsymbol{x}) < 0$, the input vector $\boldsymbol{x}$ is assigned to class C_2

The corresponding decision boundary is defined by the relation $y(\boldsymbol{x}) = 0$.



Consider two distinct points $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ both lie on the decision surface, having $y(\boldsymbol{x}_A) = y(\boldsymbol{x}_B) = 0$ which can give $\boldsymbol{w}^T(\boldsymbol{x}_A - \boldsymbol{x}_B) = 0$.

> Vector concept: if the product of two vector equals to zero, both vectors are mutually orthogonal to each other.
>
> Clearly, we can find that **the decision surface is perpendicular to $\boldsymbol{w}$.**
>
> Since D-dimension vector $\boldsymbol{x}$ consists of numerous data points, the corresponding decision surface is a $(D-1)$-dimensional hyperspace . Hence, the vector $\boldsymbol{w}$ is orthogonal to every vector lying within the decision surface.

Also, since $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$, we can derive: $\frac{\boldsymbol{w}^T\boldsymbol{x}}{\|\boldsymbol{w}\|} = -\frac{w_0}{\|\boldsymbol{w}\|}$. We can find that **the bias parameter w_0 determines the location of the decision surface**.

Multiple classes

- Difficulties in building a K-class discriminant
  - $K-1$ **classifiers** (one-versus-rest):
    - Transfer multiclass classification problem into binary classification problem
    - Use k-1 discriminant classifiers, each solves a two-class problem of separating points in a particular class C_k from the rest.
  - **One-versus-one classifier**:
    - Build decision surfaces for each pair of classes
    - Introduce $K(K-1)/2$ binary discriminant functions, one for every possible pair of classes.
    - Each point is classified according to a majority vote among the discriminant

function
  - **Issue**: Both leads to regions of input space that are ambiguously classified.
- **Solution**:
  - Use the magnitude (value) of $y_k(\boldsymbol{x})$ to classify
  - A single K-class discriminant comprising K linear functions, each linear function takes the form:

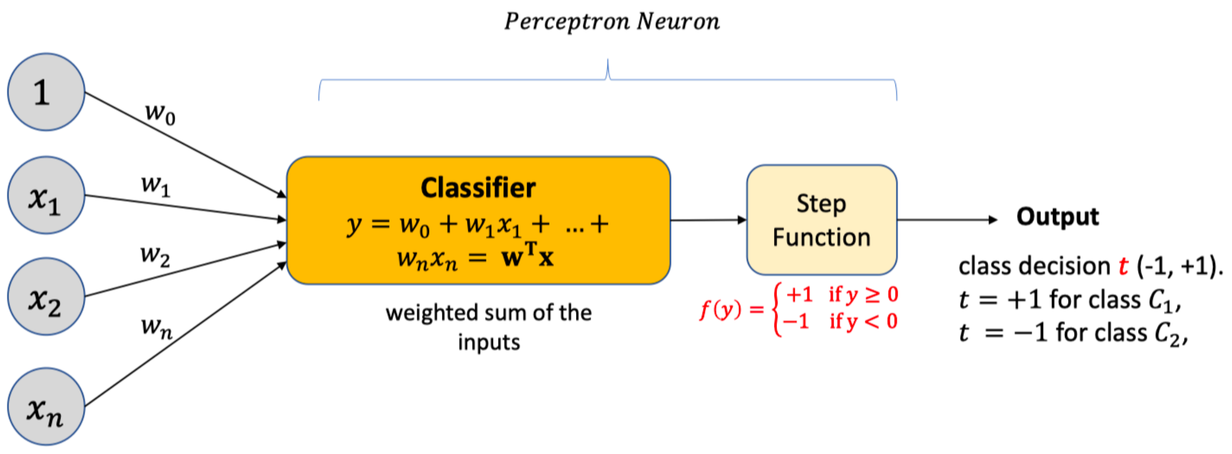
    $$y_k(\boldsymbol{x}) = \boldsymbol{w_k}^T\boldsymbol{x} + w_{k0}$$
  - $y = \arg\max y_k(\boldsymbol{x})$

    It assigns a point $\boldsymbol{x}$ to class C_k if $y_k(\boldsymbol{x}) > y_j(\boldsymbol{x})$ for all $j \neq k$.
  - The decision boundary between class C_k and class C_j is given by $y_k(\boldsymbol{x}) = y_j(\boldsymbol{x})$ and hence corresponds to a $(D-1)$-dimensional hyperplane defined by:

    $$(\boldsymbol{w_k} - \boldsymbol{w_j})^T\boldsymbol{x} + (w_{k0} - w_{j0}) = 0$$

    > Consider two points $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ both of which lie inside decision region R_k , then any point lies on the line connecting $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ can be expressed: $\hat{\boldsymbol{x}} = \lambda\boldsymbol{x}_A + (1-\lambda)\boldsymbol{x}_B$, where $\lambda \in [0, 1]$
    >
    > It indicate that If two points both lie inside the same decision region R_k , any point lie on the lie connecting these two points must lie in R_k . Hence:
    >
    > - The decision region must be singly connected and convex
    > - Without any ambiguous region

2. Perceptron Model



Perceptron Neuron
1
x_1
x_2
x_n
w_0
w_1
w_2
w_n
Classifier
$y = w_0 + w_1x_1 + \dots + w_nx_n = \mathbf{w}^T\mathbf{x}$
weighted sum of the inputs
Step Function
$f(y) = \begin{cases} +1 & \text{if } y \geq 0 \\ -1 & \text{if } y < 0 \end{cases}$
Output
class decision t (-1, +1).
$t = +1$ for class C_1,
$t = -1$ for class C_2,

- Key features:
  - Activation function: step function $f(u) = \begin{cases} +1 & u \geq 0 \\ -1 & u < 0 \end{cases}$
  - The neuron receive multiple inputs and process them to produce an output
  - The perceptron calculate 2 quantities:
    - Weighted sum of the input features
    - Threshoded by the activation function

- Objective:
    - Finding the decision surface (hyperplanes) that **minimizes the number of misclassified training data points**.
    - Determine **the weights (parameters) $\boldsymbol{w}$ to minimise an error function**
- **Error function**: Minimise the number of misclassified training data points
    - Perceptron criterion: $[\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_n) \cdot t_n < 0] \vee [t_n = -1 \wedge \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_n) = 0]$
        - if the sign of $\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_n)$ are different from t_n, then $\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_n) \cdot t_n$ must be negative
        - if $\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_n) = 0$, that means the data point is on the decision surface.
    - Given the perceptron criterion, we can construct the error function:

      $E(\boldsymbol{w}) = -\sum_{n \in M} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_n) \cdot t_n$

      where M is the set of misclasssified example, t_n is the true label of x_n.
    - Gradient of the error function: $\nabla E(\boldsymbol{w}) = -\boldsymbol{\phi}(\boldsymbol{x}_n) t_n$
    - Update weight: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla E(\boldsymbol{w}) = \boldsymbol{w} + \eta \boldsymbol{\phi}(\boldsymbol{x}_n) t_n$
    - Proof that the update will reduce the misclassified pattern:

      $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \eta \boldsymbol{\phi}(\boldsymbol{x}_n) t_n$

      $E(\boldsymbol{w}^{t+1}) = -(\boldsymbol{w}^t + \eta \boldsymbol{\phi}(\boldsymbol{x}_n) t_n) \cdot \boldsymbol{\phi}(\boldsymbol{x}_n) \cdot t_n$

      $\Rightarrow -\boldsymbol{w}^t \cdot \boldsymbol{\phi}(\boldsymbol{x}_n) \cdot t_n + \eta \boldsymbol{\phi}(\boldsymbol{x}_n) t_n \cdot \boldsymbol{\phi}(\boldsymbol{x}_n) \cdot t_n < -\boldsymbol{w}^t \cdot \boldsymbol{\phi}(\boldsymbol{x}_n) \cdot t_n \blacksquare$

      > Note that the change in weight vector may have caused some previously correctly classified patterns to become misclassified. Hence, the perceptron learning rule is **not guaranteed to reduce the total error function at each stage**.
- Pseudocode

```
Initialise w at random
While the error < epsilon
do{
    for each training point x_i
        compute the activation function
        if y(x_i) = t_i then break
        else update weight
}
```

- The perceptron convergence theorem

  If there exist an exact solution, which indicates the training data set is linearly separable, then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps.

Problems about convergence

- The number of steps require to achieve convergence could still be substantial.

  - In practice, until convergence is achieved, It is not able to distinguish between a nonseparable problem and one that is simply slow to converge.
  - Even when the data set is linearly separable, there may be many solutions. Finding one will depend on
    - The initialization of the parameters
    - the order of presentation of the data points
- **Limitation**

  Perceptron algorithm is not for $K > 2$ classification problem.

  The reason is based on the linear combination of fixed basis function.