Week 7 - Machine Learning: Classification, Perceptron

Gradient: The gradient refers to the derivative of the loss function with respect to the model parameters. In simpler terms, it represents the direction and rate of change of the loss function concerning each parameter.

Loss: The loss function, also known as the cost function or objective function, measures the difference between the predicted output of the model and the actual target output for a given input.

▼ Recap of clustering and introduction to ML

- **Clustering (Unsupervised learning)**
    - **Partitional Clustering**
        - K-means
            - Update the parameter

$$w_{n+1} = w_n - \alpha F_w(w_n)$$

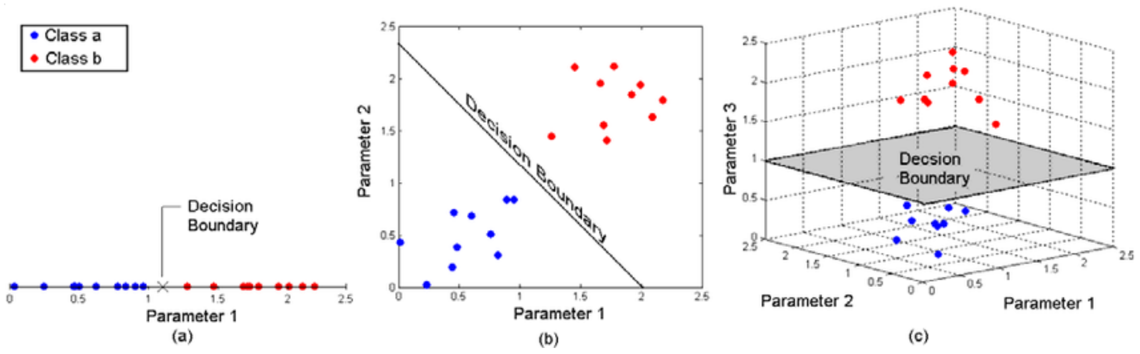
            - Convergence condition

$$\triangle F = |F(w_{n+1}) - F(w_n)|$$

            - If we write it like $w^T x_i$ (w transpose) it can be easier to do vector multiplication, updating weights and bias. Bias is consider weight. For example:
                - $f(x) = w_1x_1 + w_2x_2 + b$, treat it as $f(x) = w_1x_1 + w_2x_2 + w_3x_3$
                - W = [1, 2, 1], X = [1, 3, 1]
    - **Hierarchical Clustering**

- (for derivative to be defined, the right and left limits must agree ($lim_{h\to 0}, lim_{h\to 0+}, lim_{h\to 0-}$, these have to agree)

- Classification
  - Decision boundary
    - Minimize false positives and false negatives
    - If x has 1 attribute, the decision boundary is a point
    - If x has 2 attribute, the decision boundary is a line
    - If x has 3 attribute, the decision boundary is a plane (2D)
    - Decision boundary occurs when $w^T x = 0$



  - Classification model:

$$f(w,x) = sign(w_1 x^{[1]} + w_2 x^{[2]} + \cdots + w_d x^{[D]}) = sign(w^T x)$$
$$f : \mathbb{R}^D \to \{-1, 0, 1\}$$

ℝ → real number

  - Misclassification error ← **the goal is to minimize**

$$F(w) = \sum_{i=1}^{N} \mathbb{I}[f(w, x_i) \neq y_i]$$

assigns the same penalty to all incorrect decisions, **regardless of how 'bad' they are**

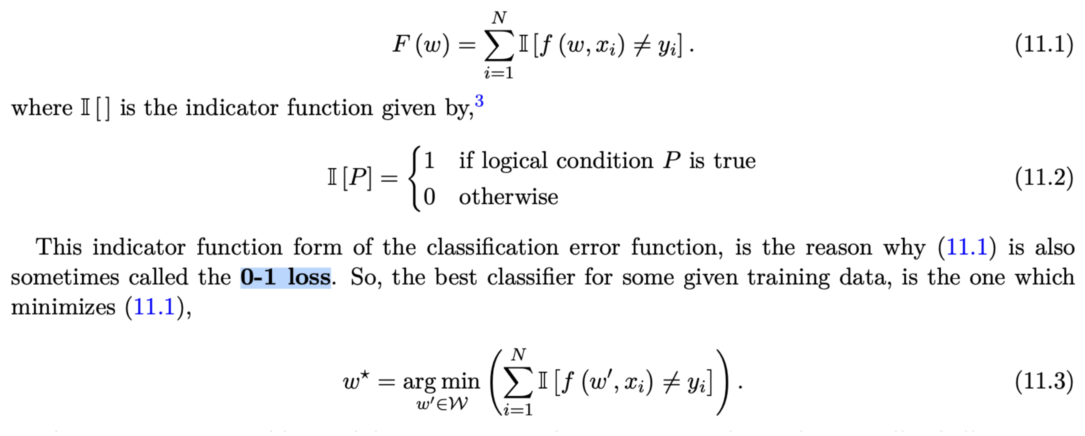
Find the good w parameters

$$w^* = \arg\min_{w' \in W} F(w')$$

$$w^* = \arg\min_{w' \in W} (\sum_{i=1}^{N} \mathbb{I}[f(w, x_i) \neq y_i])$$

- Indicator function $\mathbb{I}[P] = 1$

```
def indicator(P):
    if P:
        return 1
    else:
        return 0
```



$$F(w) = \sum_{i=1}^{N} \mathbb{I}[f(w, x_i) \neq y_i]. \quad (11.1)$$

where $\mathbb{I}[\,]$ is the indicator function given by,[3]

$$\mathbb{I}[P] = \begin{cases} 1 & \text{if logical condition } P \text{ is true} \\ 0 & \text{otherwise} \end{cases} \quad (11.2)$$

This indicator function form of the classification error function, is the reason why (11.1) is also sometimes called the **0-1 loss**. So, the best classifier for some given training data, is the one which minimizes (11.1),

$$w^\star = \arg\min_{w' \in \mathcal{W}} \left(\sum_{i=1}^{N} \mathbb{I}[f(w', x_i) \neq y_i] \right). \quad (11.3)$$

- For instance, we might be tempted to use SGD (Algorithm (9.1)) but the the gradient of $\mathbb{I}[\;]$is either *zero* (since the function is flat for nearly every input), or it is *not defined* (in fact, infinite, so we cannot even calculate its value!)
- In practice then, **most classification algorithms DO NOT use the misclassification error function**, they use a **proxy or surrogate error (loss)** which is much easier to optimize.
- **Proxy or surrogate error (loss) that is easier to optimize: (surrogate 替代) Mathematically convenient but in general not guaranteed to find the classifier which globally minimizes the misclassification error.**

- **Perceptron loss: 'Penalizes' incorrect decisions by the distance from the decision boundary wTx in the direction w (perpendicular distance).**

$$F(w) = \sum_{i=1}^{N} max[0, -y_i f(w, x_i)]$$

  - **Gradient with respect to *w*: Intuitively, gradient is just sum of $-y_i x_i$ over incorrectly classified points**

$$F_w(w) = -\sum_{i=1}^{N} y_i x_i \mathbb{I}[-y_i w^T x_i \geq 0]$$

$$w_{n+1} = w_n - \alpha F_w(w_n) = w_n - (-\sum_{i=1}^{N} y_i x_i \mathbb{I}[-y_i w^T x_i \geq 0])$$
$$= w_n + \sum_{i=1}^{N} y_i x_i \mathbb{I}[-y_i w^T x_i \geq 0]$$

  - **Perceptron training: algorithm**
    1. **Step 1. Initialization: Select a starting candidate classification model w_0, set iteration number *n* = 0, choose maximum number of iterations *R* and learning rate *α* > 0**
    2. **Step 2. Gradient descent step: Compute new model parameters: taking each *i* = 1, 2, . . . , *N* in turn, if $sign(w_n^T x_i \neq y_i)$,then $w_{n+1} = w_n + \alpha x_i y_i$**
    3. **Step 3. Iteration: If *n* < *R*, update *n* = *n* + 1, go to step 2, otherwise exit with solution $w^* = w_n$.**

```
# step 1
initial guess w_0
set iterations R
set learning rate alpha
```

```
# step 2
for n in range(R):
        for i in range(N):
                if f(x[i])!= y[i]:
                        w_{n+1) = w_n + alpha * x[i

return w^* = w_n
```

Perceptron algorithm: analysis

- If the data is linearly separable, perceptron algorithm always converges on a decision boundary with zero error but no guarantee on the number of iterations required to reach fixed point
- If data is not linearly separable, no convergence guarantee – can cycle between local optima of the perceptron error function, so we need to stop after some number of iterations *R*

Understanding the perceptron critical to understanding most of the main principles of modern ML classification

- Logistic loss:

$$F(w) = \sum_{i=1}^{N} log[1 + e^{-y_i f(w,x_i)}]$$

- Hinge loss:

$$F(w) = \sum_{i=1}^{N} max[0, 1 - y_i f(w, x_i)]$$

ML Classification: Comparisons

Model $[f(w,x)]$

Regression	Classification
$w^T x$	$\text{sign}\ (w^T x)$
> 0 (positive)	$+1$
< 0 (negative)	-1

Objective Function $[F(w)]$

True Label	Misclassification error	Perceptron loss	
y	$\mathbb{I}[f(w, x_i) \neq y_i]$	$-yw^T x$	$max\ (0, -yw^T x)$
$+1$	0	$-w^T x$	0
-1	1	$w^T x$	$w^T x$
-1	0	$w^T x$	$w^T x$
$+1$	1	$-w^T x$	0