Machine Learning (CE 40717)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

September 19, 2024

Definition

- Given: Training Set
  - A dataset D with N labeled instances $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$
  - $y^{(i)} \in \{1, \ldots, K\}$
- Goal: Given an input x, assign it to one of K classes
- Real-World Examples:
  - Email Spam Detection
  - Credit Scoring
  - Churn Prediction
  - ...

Classification vs. Regression

Aspect	Linear Regression	Linear Classification
Purpose	Predicts a continuous output (e.g., price, temperature)	Predicts a discrete class label (e.g., spam/not spam)
Output Type	Continuous values (real numbers).	Binary or Multi-class labels (e.g., 0/1, A/B/C)
Use Cases	Predicting house prices, stock market trends.	Email spam detection, Credit Scoring, Churn Prediction

Table 1: Linear Regression vs. Linear Classification

Discriminant Functions in Machine Learning

- **Definition:**
  - A function that assigns a score to an input vector x, to classify it into different classes.
  - It maps the input x to a real number $g(x)$, which represents the degree of confidence in assigning x to a particular class.

Discriminant Functions in Machine Learning

- **How it works:**
  - **Binary Classification**: Two functions $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$ for classes C_1 and C_2, respectively. The class is predicted by comparing these two functions:

$$\hat{y} = \begin{cases} 1 & \text{if } g_1(\mathbf{x}) > g_2(\mathbf{x}) \\ 2 & \text{otherwise} \end{cases}$$

  - **General Case**: For k-class problems, we compute $g_i(\mathbf{x})$ for every class i, and assign x to class with highest score:

$$\hat{y} = \arg\max_i g_i(\mathbf{x})$$

Decision Boundary

- **Definition**: A dividing hyperplane that separates different classes in a feature space, determining how data points are classified. Also known as "Decision Surface".

- **How to find**: Decision boundaries can be found using discriminant functions.
  - Boundary H between two classes i and j, separating samples between them:

$$\forall \mathbf{x} \in H, \; g_i(\mathbf{x}) = g_j(\mathbf{x})$$

Discriminant Functions: Two-Category

- **Function**: For two-category problem, we can only find a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$
  - $f_1(x) = f(x)$,
  - $f_2(x) = -f(x)$
- **Decision Boundary**: $f(x) = 0$
- At first, we start by explaining two-category classification for simplicity, and then extend the concept to multi-category classification for more complex problems.

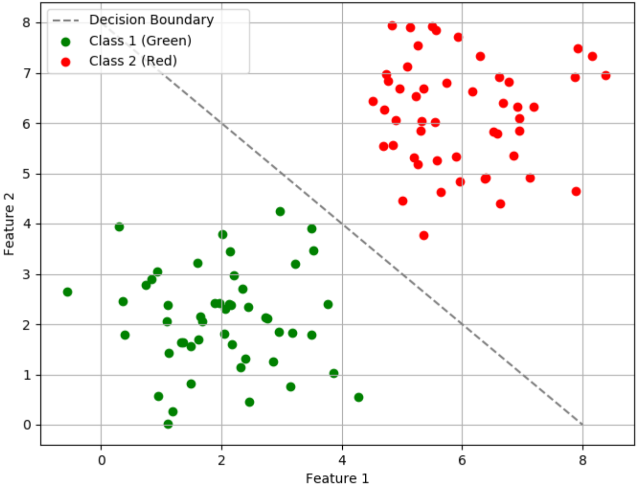
Linear Classifiers

- **Definition**: In case of linear classifiers, decision boundaries are linear in d ($\mathbf{x} \in \mathbb{R}^d$), or linear in some given set of functions of x.
- **Linearly separable data**: Data points that can be exactly separated by a linear decision boundary.
- **Why are they popular?**
  - Linear classifiers are popular due to their simplicity, efficiency, and effectiveness in solving many practical classification problems

Two Category Classification

- $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = f(\mathbf{x}) = w_d \cdot x_d + \ldots + w_1 \cdot x_1 + w_0$
  - $\mathbf{x} = [x_1 \ldots x_d]$
  - $\mathbf{w} = [w_1 \ldots w_d]$
  - w_0: bias
- $\begin{cases} C_1 & \text{if } \mathbf{w}^T\mathbf{x} + w_0 \geq 0 \\ C_2 & \text{otherwise} \end{cases}$
- **Decision Surface**: $\mathbf{w}^T\mathbf{x} + w_0 \implies$ $\mathbf{w}$ is orthogonal to every vector lying within the decision surface.

Example

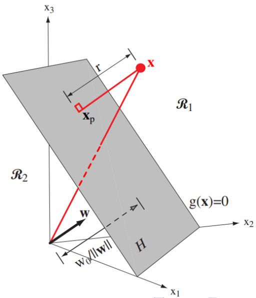

Decision Boundary
Class 1 (Green)
Class 2 (Red)
Feature 1
Feature 2
0
2
4
6
8
0
1
2
3
4
5
6
7
8

Two Category Classification Cont.

- Decision Boundary is a $(d-1)$-dimensional hyperplane H in the d-dimensional feature space. Some properties of H are:
  - Orientation of H is determined by the normal vector $[w_1 \dots w_d]$ $(\frac{w}{\|w\|})$.
  - w_0 determines the location of the surface.
    - $\frac{w_0}{\|w\|}$ Normal distance from origin to decision surface.

$$\mathbf{x} = \mathbf{x_p} + r \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

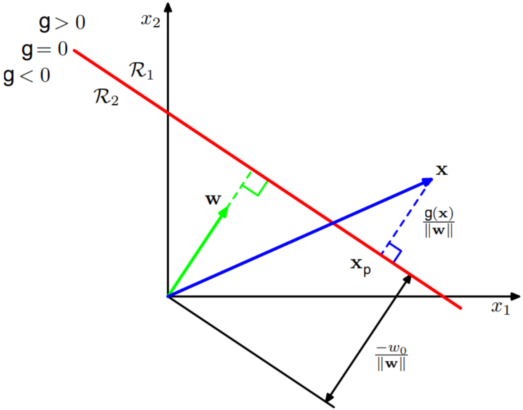
$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = r \cdot \|\mathbf{w}\| \Rightarrow r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|}$$



Linear Boundary: Geometry

- Geometry of a linear discriminant in 2D. The decision boundary (red line), orthogonal to **w**, shifted by the bias w_0.

  The orthogonal distance of point **x** from the boundary is determined by $\frac{g(\mathbf{x})}{\|w\|}$.



Multi-Category Classification

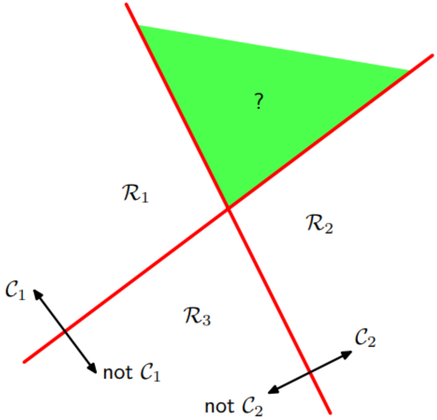
- **Solutions to multi-category classification problem**:
  - Extend the learning algorithm to support multi-class.
    - First, a function g_i for every class C_i is found.
    - Second, $\mathbf{x}$ is assigned to C_i if $g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \forall i \neq j$

$$\hat{y} = \underset{i=1,\dots,c}{\operatorname{argmax}}\, g_i(\mathbf{x})$$

  - Convert to a set of two-categorical problems.
    - Methods like **One-vs-Rest** or **One-vs-One**, where each classifier distinguishes between either **one class and the rest**, or **between pairs of classes**.

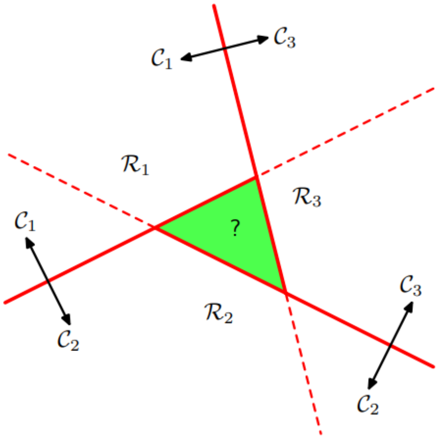
Multi-Category Classification

- **One-vs-Rest** (One-vs-All):



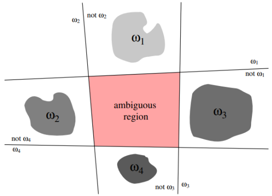
Multi-Category Classification

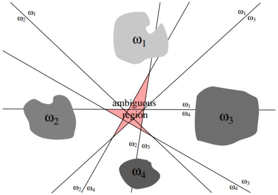
- **One-vs-One**:

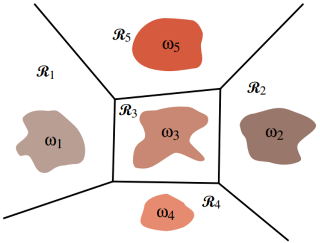


Multi-Category Classification: Ambiguity

- One-vs-One and One-vs-Rest conversion can lead to regions in which the classification is **undefined**.







Multi-Category Classification: Linear Machines

- **Linear Machines**: Alternative to One-vs-Rest and One-vs-One methods; Each class is represented by its own discriminant function.

- **Decision Rule:**

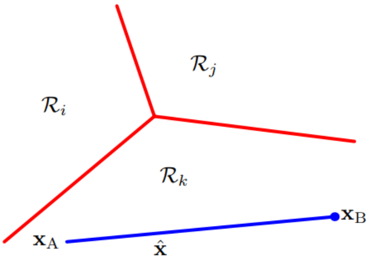
$$\hat{y} = \underset{i=1,\ldots,c}{\operatorname{argmax}}\, g_i(\mathbf{x})$$

The predicted class is the one with the highest discriminant function value.

- **Decision Boundary**: $g_i(\mathbf{x}) = g_j(\mathbf{x})$

$$(\mathbf{w}_i - \mathbf{w}_j)^T \mathbf{x} + (w_{0i} - w_{0j}) = 0$$

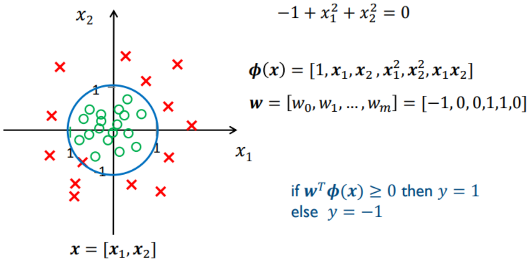
Linear Machines Cont.



- The decision regions of this discriminant are **convex** and **singly connected**. Any point on the line between two points within the same region can be expressed as $\mathbf{x} = \lambda \mathbf{x}_A + (1-\lambda)\mathbf{x}_B$ where $\mathbf{x}_A, \mathbf{x}_B \in R_k$.

Non-linear decision boundary

- **Non-linear Decision Boundaries:**
  - **Feature Transformation**: Non-linearity is introduced by transforming features into a higher-dimensional space.
  - **Linear in Transformed Space**: The decision boundary becomes linear in the new space, but non-linear in the original space.



Cost Functions

- **Cost Functions in Linear Classifiers:**
  - Purpose of cost functions is to measure the **difference** between **predicted** and **actual** class labels.
  - Finding discriminant functions is framed as minimizing a cost function.
    - Based on training set $D = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, a cost function $J(\mathbf{w})$ is defined.
    - Problem converts to finding optimal $\hat{g}(\mathbf{x}) = g(\mathbf{x}; \hat{\mathbf{w}})$ where

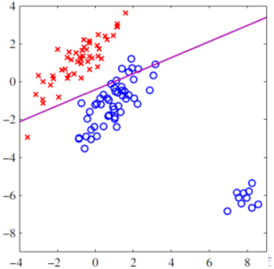
$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w})$$

Sum of Squared Error Cost Function

- **Sum of Squared Error (SSE) Cost Function:**
  - **Definition**: SSE measures the sum of the squared differences between predicted ($\hat{y}^{(i)}$) and actual ($y^{(i)}$) labels.
  - **Formula**:

$$J(\mathbf{w}) = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2, \quad \hat{y}^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)} + w_0$$

  - **Limitations**:
    - **Low Robustness to Noise**: Prone to over-fitting noisy data, as small variations can cause significant changes in the cost.

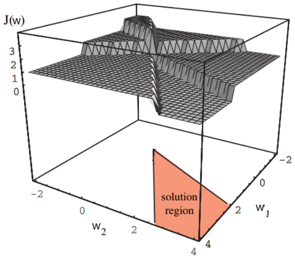


An Alternative for SSE Cost Function

- **Number of Misclassifications:**
  - **Definition**: Measures how many samples are misclassified by the model.
  - **Formula**:

$$J(\mathbf{w}) = \sum_{i=1}^{n} (y^{(i)} - \text{sign}(\hat{y}^{(i)}))^2, \quad \hat{y}^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)} + w_0$$

  - **Limitations**:
    - **Piecewise Constant**: The cost function is non-differentiable, so optimization techniques (like gradient descent) cannot be directly applied.



Perceptron

- **The Perceptron Algorithm:**
  - **Purpose**: A simple algorithm for binary classification, separating two classes with a linear boundary.
  - **Decision Rule**: $y = \text{sign}(\mathbf{w}^T \mathbf{x})$
  - Simplifying Assumption: For simplicity, the bias term is built into the vectors $\mathbf{x}$ and $\mathbf{w}$.
  - **Learning Process**: Iteratively updates $\mathbf{w}$ based on misclassified examples.

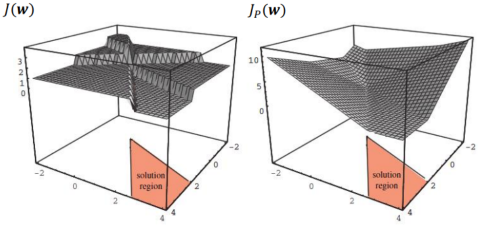
Perceptron Criterion

- **Cost Function**: The perceptron criterion focuses on misclassified points:

$$J_p(\mathbf{w}) = -\sum_{i \in M} y^{(i)}\, \mathbf{w}^T \mathbf{x}^{(i)}$$

  where M is the set of misclassified points.

- **Goal**: Minimize the loss by correctly classifying all points.



Batch Perceptron

- **Batch Perceptron**: Updates the weight vector using all misclassified points in each iteration.

- **Gradient Descent**: Solves the criterion by adjusting weights in the direction that reduces the loss:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} J_p(\mathbf{w})$$

$$\nabla_{\mathbf{w}} J_p(\mathbf{w}) = -\sum_{i \in M} y_i \mathbf{x}_i$$

where η is the learning rate.

  - Batch Perceptron converges in finite number of steps for linearly separable data.

Single-sample Perceptron

- **Single Sample Perceptron**: Updates the weight vector after each individual misclassified point.

- **Stochastic Gradient Descent (SGD) Update Rule:**
  - Update is performed using a single misclassified sample:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i$$

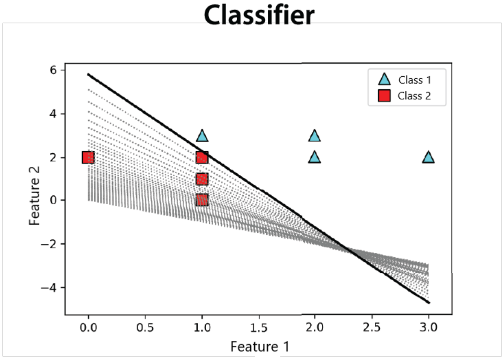
    where y_i is the label of the misclassified sample and η is the learning rate.

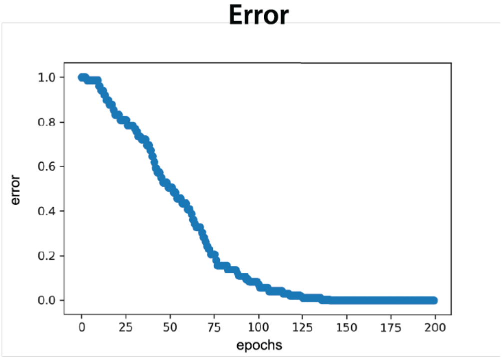
  - Lower computational cost per iteration, faster convergence.

  - If training data are linearly separable, the single-sample perceptron is also guaranteed to find a solution in a finite number of steps.

Perceptron: Example

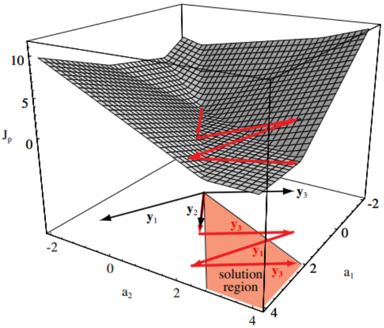
- Perceptron changes **w** in a direction that corrects error.





Convergence of Perceptron

- **Non-Linearly Separable Data**: When no linear decision boundary can perfectly separate the classes, the Perceptron fails to converge.
  - The Perceptron updates its weights based on misclassified points.
  - If data is not linearly separable, there will always be some points that the model cannot classify correctly.
  - As a result, the algorithm keeps adjusting the weights to fix the misclassified points, causing it to never converge.



Pocket Algorithm

- For the data that are not linearly separable due to noise, **Pocket Algorithm** keeps in its pocket the best $\mathbf{w}$ encountered up to now.

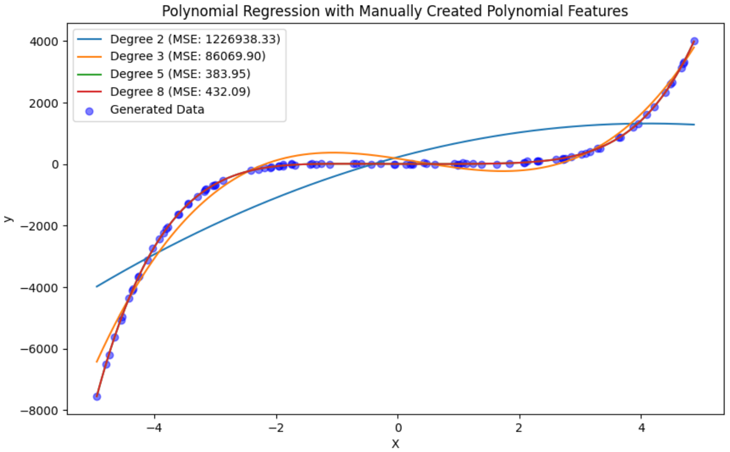
Algorithm 1 Pocket Algorithm

Initialize $\mathbf{w}$
for $t = 1$ to T **do**
    $i \leftarrow t \bmod N$
    if $\mathbf{x}^{(i)}$ is misclassified **then**
        $\mathbf{w}^{new} = \mathbf{w} + \mathbf{x}^{(i)} y^{(i)}$
        if $E_{train}(\mathbf{w}^{new}) < E_{train}(\mathbf{w})$ **then**
            $\mathbf{w} = \mathbf{w}^{new}$
        end if
    end if
end for

Model Selection via Cross Validation

- **Cross-Validation:**
  - **Purpose**: Technique for evaluating how well a model generalizes to unseen data.
  - **How It Works**: Split data into k folds; train on $k-1$ folds and validate on the remaining fold.
  - **Repeat Process**: Repeat k times, rotating the test fold each time. Average of all scores is the final score of the model.
  - Cross-validation reduces overfitting and provides a more reliable estimation of model performance.

Visualization of Cross-Validation



Leave-One-Out Cross-Validation (LOOCV)

- **Leave-One-Out Cross-Validation (LOOCV):**
  - **How It Works**: Uses a single data point as the validation set and the rest as the training set. Repeat for all data points.
  - **Properties:**
    - **No Data Wastage**: Every data point is used for both training and validation.
    - **High Variance, Low Bias**.
    - **Computationally Expensive**: Requires training the model N times for N data points, making it slow for large datasets.
    - **Best for small datasets**.

[1] Q. Lu, "A uow beamer theme," in *How to write beautiful LaTeX*, 2024.

Any Questions?