

Machine Learning (CE 40717)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

September 30, 2024



1 Introduction to Classification

2 Discriminant Functions

3 Linear Classifiers

4 Perceptron

5 Cost Functions

6 Cross Validation

7 Multi-Category Classification

8 References

1 Introduction to Classification

2 Discriminant Functions

3 Linear Classifiers

4 Perceptron

5 Cost Functions

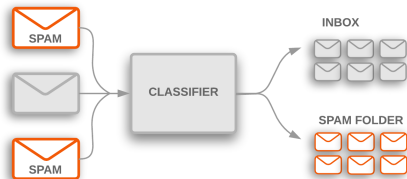
6 Cross Validation

7 Multi-Category Classification

8 References

Definition

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



Real-World Example of Classification

- Pima Indians Diabetes Dataset:**

- Problem:** Predict whether a patient has diabetes based on medical diagnostics.
- Context:** Early detection of diabetes is critical for treatment and management.

	Number of times pregnant	Glucose	Blood Pressure	Skin Thickness	Insulin	Diabetes pedigree function	Age	BMI	Label
Patient 1	6	148	72	35	0	0.627	50	33.6	Positive
Patient 2	1	85	66	29	0	0.351	31	26.6	Negative
Patient 3	0	137	40	35	168	2.288	33	43.1	Positive
Patient 4	1	89	66	23	94	0.167	21	28.1	Negative
.
.
.

Classification vs. Regression

Aspect	Linear Regression	Linear Classification
Output Type	Continuous values (real numbers).	Binary or Multi-class labels (e.g., -1/+1, A/B/C)
Use Cases	Predicting house prices, stock market trends.	Email spam detection, Credit Scoring, Churn Prediction

1 Introduction to Classification

2 Discriminant Functions

3 Linear Classifiers

4 Perceptron

5 Cost Functions

6 Cross Validation

7 Multi-Category Classification

8 References

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 \mathbf{x} to a real number $g(\mathbf{x})$, which represents the degree of confidence in assigning \mathbf{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:

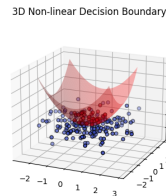
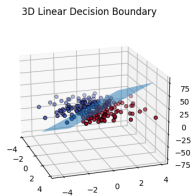
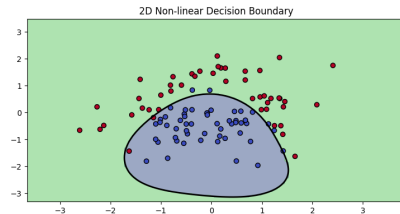
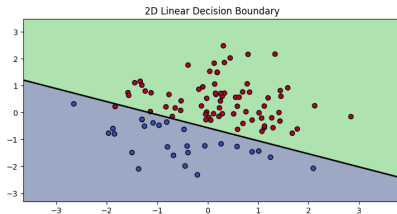
$$\hat{y} = \begin{cases} C_1 & \text{if } g_1(\mathbf{x}) > g_2(\mathbf{x}) \\ C_2 & \text{otherwise} \end{cases}$$

- **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, also known as "Decision Surface".



Discriminant Functions: Two-Category

- **Function:** For two-category problem, we can only find a function $g : \mathbb{R}^d \rightarrow \mathbb{R}$
 - $g_1(\mathbf{x}) = g(\mathbf{x})$,
 - $g_2(\mathbf{x}) = -g(\mathbf{x})$
- **Decision Boundary:** $g(\mathbf{x}) = 0$
- At first, we start by explaining two-category classification for simplicity, and then extend the concept to multi-category classification for more complex problems.

1 Introduction to Classification

2 Discriminant Functions

3 Linear Classifiers

4 Perceptron

5 Cost Functions

6 Cross Validation

7 Multi-Category Classification

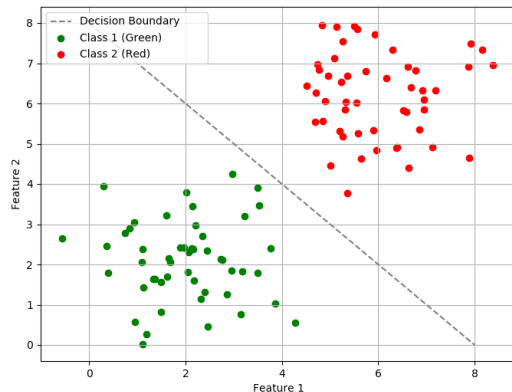
8 References

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?**
 - Simplicity, Efficiency, Effectiveness.

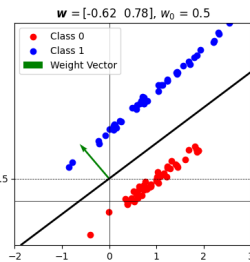
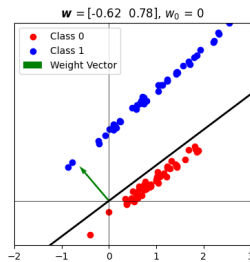
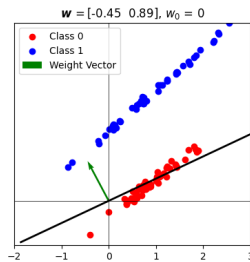
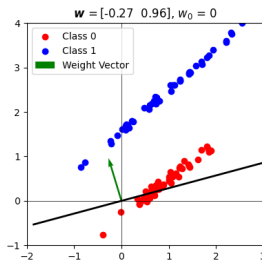
Two Category Classification

- $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 = w_d \cdot x_d + \dots + w_1 \cdot x_1 + w_0$
 - $\mathbf{x} = [x_1 \dots x_d]$
 - $\mathbf{w} = [w_1 \dots w_d]$
 - 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:** $\mathbf{w}^T \mathbf{x} + w_0$



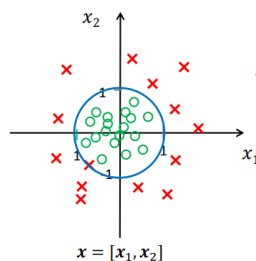
Two Category Classification Cont.

- 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 $[\frac{w_1}{\|w\|}, \dots, \frac{w_d}{\|w\|}]$.
 - w_0 determines the location of the surface.



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.



$$-1 + x_1^2 + x_2^2 = 0$$

$$\phi(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1x_2]$$

$$w = [w_0, w_1, \dots, w_m] = [-1, 0, 0, 1, 1, 0]$$

if $w^T \phi(x) \geq 0$ then $y = 1$
else $y = -1$

1 Introduction to Classification

2 Discriminant Functions

3 Linear Classifiers

4 Perceptron

5 Cost Functions

6 Cross Validation

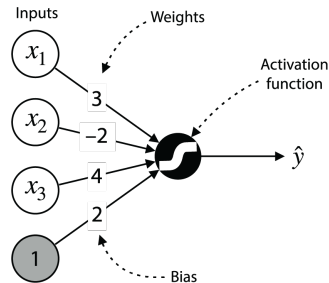
7 Multi-Category Classification

8 References

What Is Perceptron?

- **Perceptron Unit:**

- **Basic Building Block:** A perceptron is the simplest type of artificial neuron used in machine learning.
- **Linear Classifier:** It maps input features to an output by applying a linear combination and a threshold.
- **Binary Decision:** Outputs 1 if the weighted sum of inputs exceeds the threshold, otherwise 0.
- **Components:** Inputs, weights, bias, and an activation function (often a step or a sigmoid function).



Inspired by Biology

- **Biological Motivation Behind Perceptron:**

- **Inspired by Neurons:** Perceptron mimics the basic function of biological neurons in the brain.
 - Input and Output, Activation Function.

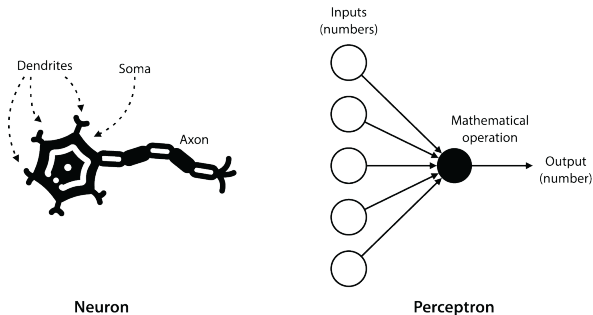


Figure adapted from Grokking Machine Learning, L. G. Serrano.

Single Neuron

- **Single Neuron as a Linear Decision Boundary**

- **Mathematical Form:** The output of a single neuron is computed as:

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

where:

- \mathbf{x} is the input vector.
 - \mathbf{w} is the weight vector.
 - w_0 is the bias term.
 - f is an activation function (e.g., step function).
- **Linear Separation:** A neuron defines a linear decision boundary:
 $\mathbf{w}^T \mathbf{x} + w_0 = \text{threshold}$ (0 for step, 0.5 for sigmoid)
 - **Decision Rule:** C_1 if $\mathbf{w}^T \mathbf{x} + w_0 \geq \text{threshold}$, otherwise C_2 .

$$\text{Class} = f(\mathbf{w}^T \mathbf{x} + w_0)$$

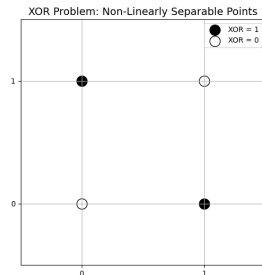


Figure adapted from Grokking Machine Learning, L. G. Serrano.

Limitations of a Single Perceptron

- **What a Single Perceptron Can and Can't Do:**

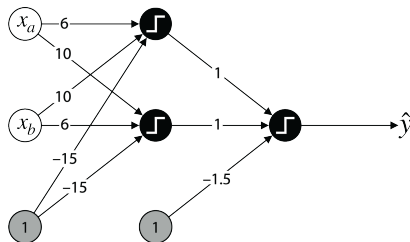
- **Performs Linear Separations:** A perceptron can handle linearly separable problems such as:
 - AND operation
 - OR operation
- **Fails on Non-Linear Problems:** A single perceptron fails to solve non-linear problems like XOR, as the data points cannot be separated by a straight line.



Towards Complex Decision Boundaries

- **Multi-Layer Perceptron (MLP):**

- **Adding Layers for More Complexity:** An MLP consists of multiple layers of neurons that allow us to model more complex functions than a single neuron.
 - Each layer introduces new decision boundaries, making it possible to separate non-linear data.
- **Two-Layer Example:**
 - Input Layer → Hidden Layer → Output Layer
 - Hidden layer introduces non-linear transformations that enable complex decision regions.



Figures adapted from Grokking Machine Learning, L. G. Serrano.

Refining the Decision Boundary

- **New Neurons for Better Separation:** By adding more neurons to a layer, we can further refine the decision boundary to better separate complex data.
- Each additional neuron introduces new features that help the model make more accurate decisions.

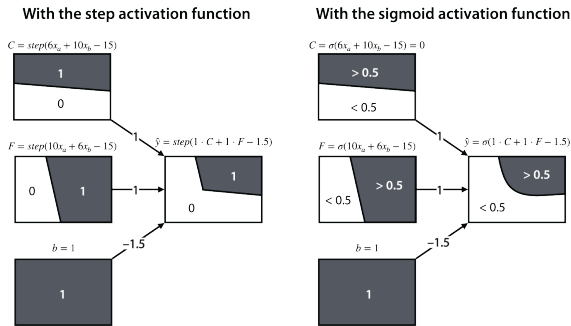


Figure adapted from Grokking Machine Learning, L. G. Serrano.

1 Introduction to Classification

2 Discriminant Functions

3 Linear Classifiers

4 Perceptron

5 Cost Functions

6 Cross Validation

7 Multi-Category Classification

8 References

Cost Functions


- **Understanding the Goal**

- In the perceptron, we use $\mathbf{w}^T \mathbf{x}$ to make predictions.
- Goal is to find the optimal \mathbf{w} so that the predicted labels match the true labels as much as possible.
- To achieve this, we define a cost function, which measures the **difference** between **predicted** and **actual** labels.
- Finding discriminant functions (\mathbf{w}^T, w_0) 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

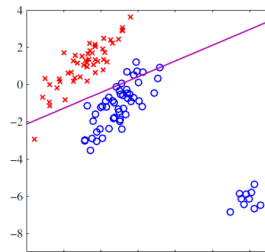
$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} J(\mathbf{w})$$

Sum of Squared Error Cost Function

- **Sum of Squared Error (SSE) Cost Function**

- **Formula:** $J(\mathbf{w}) = \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$, $\hat{y}^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)} + w_0$ 
- SSE minimizes the magnitude of the error, which is ideal for regression but **irrelevant** for classification.
- If the model predicts close to the true class but not exactly 0 or 1, SSE still shows positive error, even for correct predictions.

- SSE is also prone to overfitting 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:**

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

- **Limitations:**

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

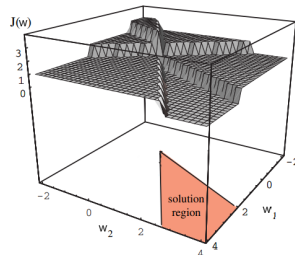


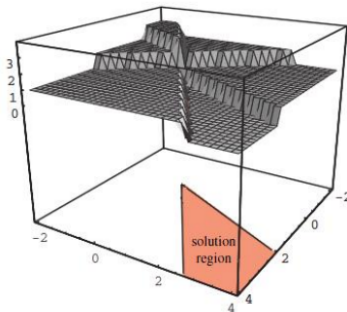
Figure adapted from Machine Learning and Pattern Recognition, Bishop

Perceptron Algorithm

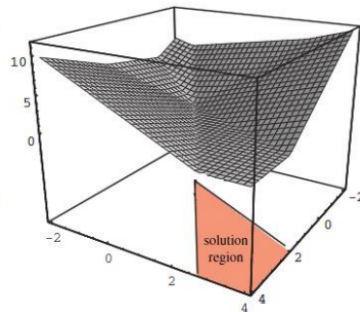
- **The Perceptron Algorithm**

- **Purpose:** A simple algorithm for binary classification, separating two classes with a linear boundary.

$J(\mathbf{w})$



$J_P(\mathbf{w})$



Perceptron Criterion

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

$$J_p(\mathbf{w}) = - \sum_{i \in M} y^{(i)} \mathbf{w}^T \mathbf{x}^{(i)}, \quad y^{(i)} \in \{-1, +1\}$$

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:** Adjusting weights in the direction that reduces the loss:

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

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

- 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 point.
- **Stochastic Gradient Descent (SGD) Update Rule:**
 - Using only one misclassified sample at a time:

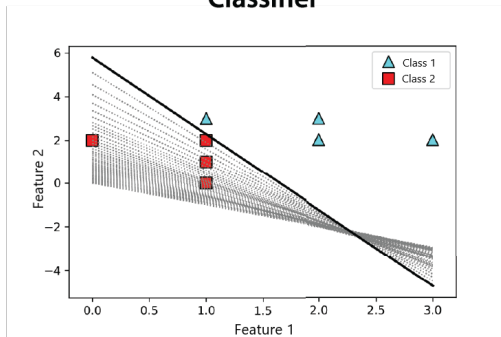
$$\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i$$

- 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.

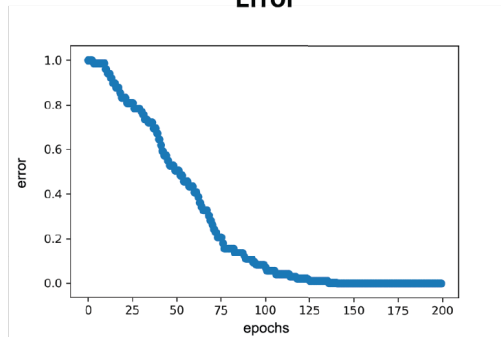
Example

- Perceptron changes \mathbf{w} in a direction that corrects error.

Classifier



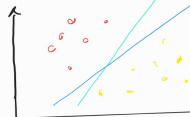
Error



Figures adapted from Grokking Machine Learning, L. G. Serrano.

Convergence of the Perceptron

prove perceptron converges in finite
of steps



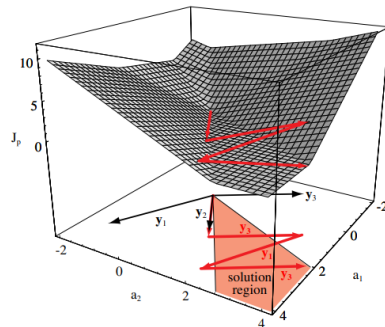
$$\begin{aligned} w^0 &= [0, c_1, \dots]^T \\ w^{t+1} &\leftarrow w^t + y^{(i)} x^{(i)} \quad y^{(i)} \in \{-1, 1\} \\ w^* \cdot w^{t+1} &= w^* (w^t + y^{(i)} x^{(i)}) \\ &= w^* w^t + \underbrace{y^{(i)} w^* x^{(i)}}_{\geq \gamma > 0} \end{aligned}$$

$w^* \cdot w^{t+1}$ grows with every iteration, however it cannot grow indefinitely since $\max(w^* \cdot w^t) = \cos(w^*, w^t)$ so, it has to stop at # of steps (aka converge)

Convergence of the Perceptron Cont.

Convergence of Perceptron Cont.

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



Pocket Algorithm

Algorithm 1 Pocket Algorithm

```
1: Initialize  $\mathbf{w}$ 
2: for  $t = 1$  to  $T$  do
3:    $i \leftarrow t \bmod N$ 
4:   if  $\mathbf{x}^{(i)}$  is misclassified then
5:      $\mathbf{w}^{new} = \mathbf{w} + \eta \mathbf{x}^{(i)} y^{(i)}$ 
6:     if  $E_{train}(\mathbf{w}^{new}) < E_{train}(\mathbf{w})$  then
7:        $\mathbf{w} = \mathbf{w}^{new}$ 
8:     end if
9:   end if
10: end for
```

$$\triangleright E_{train}(\mathbf{w}) = J_p(\mathbf{w})$$

1 Introduction to Classification

2 Discriminant Functions

3 Linear Classifiers

4 Perceptron

5 Cost Functions

6 Cross Validation

7 Multi-Category Classification

8 References

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.

K-Fold Cross Validation

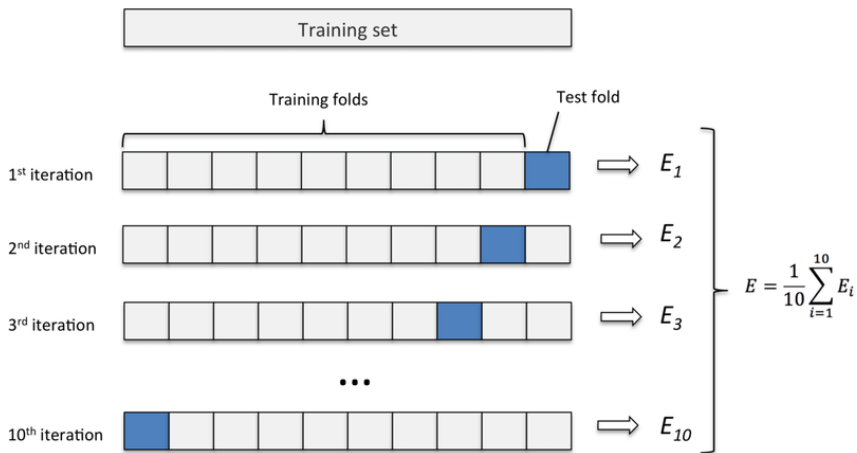
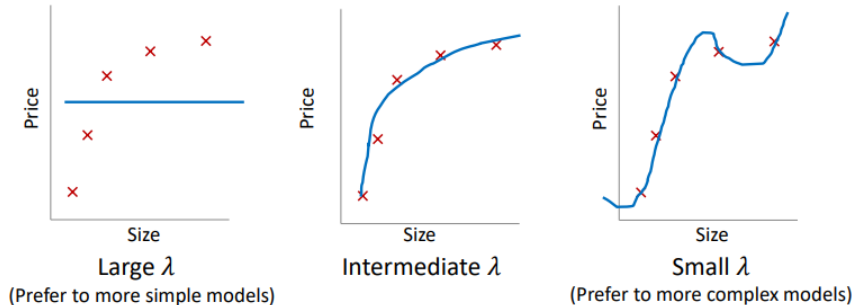


Figure adapted from Introduction to Support Vector Machines and Kernel Methods, J.M. Ashfaq.

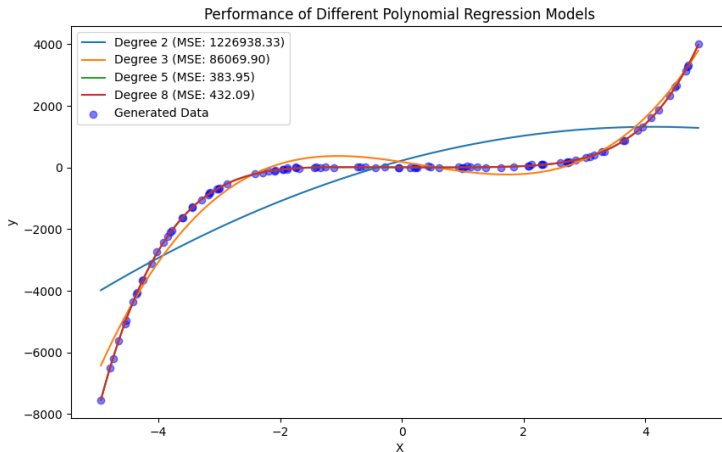
Leave-One-Out Cross-Validation (LOOCV)

- **Leave-One-Out Cross-Validation (LOOCV)**
 - **How It Works:** Uses a single data point as the validation set ($k = 1$) 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.**

Cross-Validation for Choosing Regularization Term



Cross-Validation for Choosing Model Complexity



1 Introduction to Classification

2 Discriminant Functions

3 Linear Classifiers

4 Perceptron

5 Cost Functions

6 Cross Validation

7 Multi-Category Classification

8 References

Multi-Category Classification

- **Solutions to multi-category classification problem:**

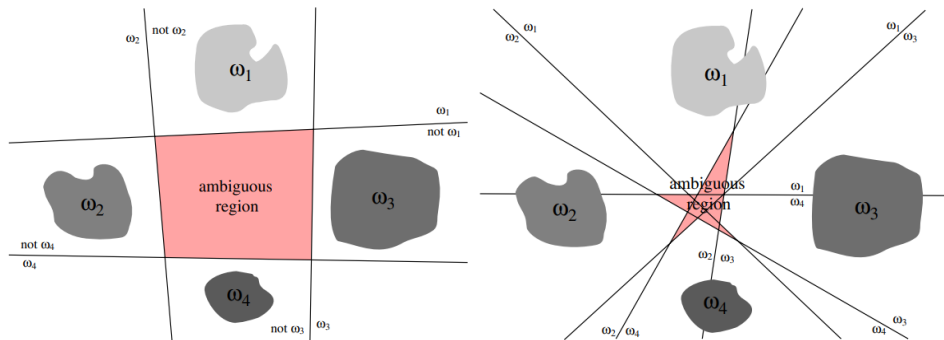
- Extend the learning algorithm to support multi-class.
 - First, a function g_i for every class C_i is found.
 - Second, \mathbf{x} is assigned to C_i if $g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \forall i \neq j$

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

- 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: 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:**

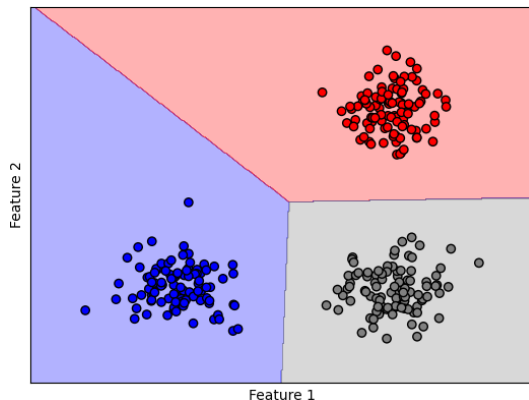
$$\hat{y} = \underset{i=1,\dots,c}{\operatorname{argmax}} g_i(\mathbf{x})$$

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 $\mathbf{x}_A, \mathbf{x}_B \in C_k$.

Multi-Class Perceptron Algorithm

- **Weight Vectors:**

- Maintain a weight matrix $W \in \mathbb{R}^{m \times K}$, where m is the number of features and K is the number of classes.
- Each column w_k of the matrix corresponds to the weight vector for class k .

$$\hat{y} = \underset{i=1, \dots, c}{\operatorname{argmax}} \mathbf{w}_i^T \mathbf{x}$$

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

where M is the set of misclassified points.

Multi-Class Perceptron Algorithm

Algorithm 2 Multi-class perceptron

```
1: Initialize  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_c], k \leftarrow 0$ 
2: while A pattern is misclassified do
3:    $k \leftarrow k + 1 \bmod N$ 
4:   if  $\mathbf{x}^{(i)}$  is misclassified then
5:      $\mathbf{w}_{\hat{y}^{(i)}} = \mathbf{w}_{\hat{y}^{(i)}} - \eta \mathbf{x}^{(i)}$ 
6:      $\mathbf{w}_{y^{(i)}} = \mathbf{w}_{y^{(i)}} + \eta \mathbf{x}^{(i)}$ 
7:   end if
8: end while
```

1 Introduction to Classification

2 Discriminant Functions

3 Linear Classifiers

4 Perceptron

5 Cost Functions

6 Cross Validation

7 Multi-Category Classification

8 References

Contributions

- **This slide has been prepared thanks to:**
 - Erfan Jafari

- [1] C. M. Bishop, *Pattern Recognition and Machine Learning*.
- [2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. 2001.
- [3] M. Soleymani, “Machine learning.” Sharif University of Technology.
- [4] S. F. S. Salehi, “Machine learning.” Sharif University of Technology.
- [5] Y. S. Abu-Mostafa, “Machine learning.” California Institute of Technology, 2012.
- [6] L. G. Serrano, *Grokking Machine Learning*. Manning Publications, 2020.
- [7] J. M. Ashfaq, “Introduction to support vector machines and kernel methods.” April 2019.