**CS 412 — Introduction to Machine Learning (UIC)** March 1, 2025

# Lecture 13

*Instructor: Aadirupa Saha* *Scribe(s): Aravind Madishetty*

## Overview

In the last lecture, we covered the following main topics:

1. KKT conditions (contd)
2. Dual Optimization
3. SVM with strong duality

This lecture focuses on

1. Kernels & SVM
2. Different Examples of Kernels
3. Properties of a Kernel Matrix

## 1 Linear and Non-Linear Data Separation

In Support Vector Machines (SVM), data that is **not linearly separable** requires feature transformation to make it separable in a higher-dimensional space. This is done to enable the classification of data points using a linear boundary in that higher-dimensional space.

- For **linear separation** in $\mathbb{R}^2$, the data can be separated by the following linear equation:

$$x_2 = mx_1 + b$$

  where $m = \tan\theta$ and $b$ is the bias term.

- For **non-linearly separable data**, a **parabolic separation** is used:

$$x_2 = \beta x_1^2 + \alpha x_1 + \gamma$$

  This equation represents a non-linear boundary that can separate data in the higher-dimensional space.

The dataset is denoted by:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

where $x_n \in \mathbb{R}^d$ are the data instances and $y_n \in \{-1, 1\}$ are the data labels.

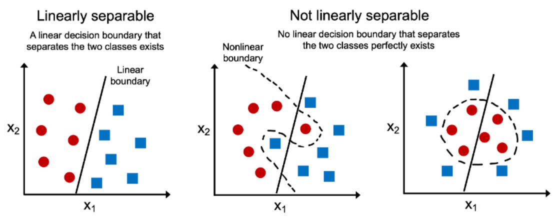
Linearly separable
A linear decision boundary that separates the two classes exists
Linear boundary
Not linearly separable
No linear decision boundary that separates the two classes perfectly exists
Nonlinear boundary
x2
x1

Figure 1: Linear and Non-Linear Data Separation

## 1.1 Why is Data Non-Linearly Separable?

Non-linearly separable data occurs when no straight line or hyperplane can divide the data points of two classes. The data points are so interwoven or placed in such a way that a simple linear boundary cannot effectively separate them.
In simpler terms:

- The data from one class might be in a circular or parabolic shape, while the other class surrounds it or exists outside it.
- Data points of different classes might have complex arrangements that no straight line can separate.

Example: Consider a situation where:
- One class forms a circular shape in a 2D plane.
- The other class surrounds this circular shape.

No straight line can separate these two classes. Therefore, non-linear methods are required to create more complex boundaries like curves or other shapes that can correctly divide the classes.

## 1.2 Mathematical Representation of Non-Linearly Separable Data

Given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where: - $x_n \in \mathbb{R}^d$ are the data instances (features).
- $y_n \in \{-1, 1\}$ are the labels of the instances.
For linearly separable data, the condition is:

$$y_n(w^T x_n + b) \geq 1, \quad \forall n$$

Where $w$ is the weight vector and $b$ is the bias term.
However, for non-linearly separable data, this condition cannot be satisfied by a single linear boundary. Therefore, non-linear boundaries are used to separate the data points, often achieved by mapping the data to a higher-dimensional space.

## 1.3 Handling Non-Linearly Separable Data

To address non-linear separability, we use techniques like:

- **Feature Transformation**: This maps the data to a higher-dimensional space where linear separation may become possible.
- **Kernel Trick**: Instead of explicitly transforming the data to higher dimensions, kernels are used to compute the inner products in the higher-dimensional space.

These techniques enable models like Support Vector Machines (SVM) to find a non-linear decision boundary by transforming the problem into a higher-dimensional feature space.

## 2 Solving the SVM Problem

### 2.1 Step 1: Problem Setup

We are given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where:

- $x_n \in \mathbb{R}^d$ are the data points.
- $y_n \in \{-1, 1\}$ are the corresponding class labels.

The goal is to find a hyperplane that separates the data points into two classes. The equation for this hyperplane is:

$$w^T x + b = 0$$

where $w$ is the weight vector, and $b$ is the bias term.

### 2.2 Step 2: Primal Formulation (Optimization Problem)

The **margin** is the distance between the hyperplane and the closest data points, called the support vectors. We aim to maximize the margin while ensuring correct classification. This leads to the following primal optimization problem:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

Subject to the constraints:

$$y_n(w^T x_n + b) \geq 1, \quad \forall n$$

Where $\|w\|^2$ is the squared Euclidean norm of $w$, and minimizing it maximizes the margin.

### 2.3 Step 3: Dual Formulation

To solve this problem, we introduce **Lagrange multipliers** $\alpha_n \geq 0$ for each constraint:

$$L(w, b, \alpha_n) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} \alpha_n \left[y_n(w^T x_n + b) - 1\right]$$

The dual form is obtained by maximizing the Lagrangian with respect to $\alpha_n$ and minimizing with respect to $w$ and $b$. The dual optimization problem becomes:

$$\max_{\alpha_n} \left( \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m K(x_n, x_m) \right)$$

Subject to the constraints:

$$\alpha_n \geq 0, \quad \sum_{n=1}^{N} \alpha_n y_n = 0$$

Where $K(x_n, x_m)$ is the **kernel function**.

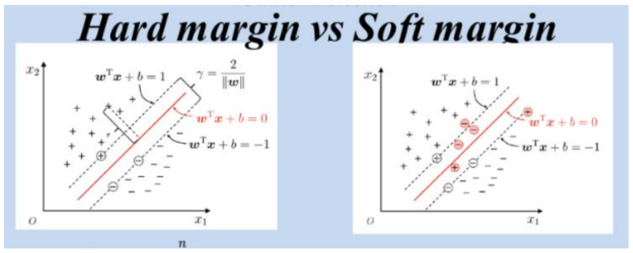

Figure 2: Hard-Margin SVM and Soft-Margin SVM

## 2.4 Step 4: Solving the Dual Problem

The dual problem is solved using Quadratic Programming (QP) solvers. The optimal values of $\alpha_n$ are determined, and the support vectors are the data points for which $\alpha_n > 0$.

## 2.5 Step 5: Compute the Weight Vector $w$

Once the dual problem is solved, the weight vector $w$ is computed as:

$$w = \sum_{n=1}^{N} \alpha_n y_n \varphi(x_n)$$

Where $\varphi(x_n)$ is the feature mapping function transforms the data to a higher-dimensional space.

## 2.6 Step 6: Compute the Bias Term $b^*$

There are two Cases for computing $b^*$:

### 2.6.1 Hard-Margin SVM (Perfect Separation)

If the data is perfectly separable, $b^*$ can be computed as:

$$b^* = y_n - w^T \varphi(x_n)$$

for any support vector $x_n$ that satisfies the equation $y_n(w^T \varphi(x_n) + b) = 1$.

### 2.6.2 Soft-Margin SVM (Allowing Misclassification)

In the case of misclassified data, we compute $b^*$ as the average over all support vectors:

$$b^* = \frac{1}{|SV|} \sum_{n \in SV} (y_n - w^T x_n)$$

where $|SV|$ is the total number of support vectors, and $SV = \{n \mid \alpha_n > 0\}$.

## 2.7 Step 7: Final Decision Rule

After computing $w$ and $b^*$, the final decision rule for classifying new data points is:

$$f(x) = \text{sign}(w^T \varphi(x) + b^*)$$

### 2.8 Step 8: Kernel Trick (Non-Linear SVM)

In real-world scenarios, the data is often non-linearly separable. The **kernel trick** is used to map the data into a higher-dimensional space where a linear separator may exist. Common kernels include:

- **Linear Kernel**: $K(x_i, x_j) = x_i^T x_j$
- **Polynomial Kernel**: $K(x_i, x_j) = (1 + x_i^T x_j)^q$
- **RBF Kernel**: $K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$

These kernels help to solve SVM problems with non-linear boundaries.

## 3 Dual Problem in Non-Linear SVM

The main challenge and difference (compared to Linear SVM) lies in computing the term $\varphi(x_n) \cdot \varphi(x_m)$ in the Dual Problem.

> **Q:** Is it easy to compute $\varphi(x_n)^T \varphi(x_m)$ in $\mathbb{R}^D$? *(where $D$ could be very large!)*

we will see the difficulty in computing the dot product $\varphi(x_n)^T \varphi(x_m)$ when the data points are mapped to a higher-dimensional space $\mathbb{R}^D$ through the feature mapping function $\varphi$. As the number of dimensions $D$ increases, the computation of this dot product becomes increasingly expensive and complex.

### 3.1 Feature Transformation

To map the data into a higher-dimensional space, we use feature transformations. For example, a feature transformation from $\mathbb{R}^2$ to $\mathbb{R}^6$ can be defined as:

$$\varphi(x) = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$$

This transformation allows the data to become linearly separable in the transformed space.

### 3.2 Dot Product Calculation in Higher Dimensions

Next, consider the calculation of the dot product between the transformed feature vectors $\varphi(x_n)$ and $\varphi(x_m)$, the calculation can be quite complex:

$$\varphi(x_n)^T \varphi(x_m) = \langle \varphi(x_n), \varphi(x_m) \rangle = 1 + (x_n^m)^2 + (x_m^m)^2 + 2\alpha_n\alpha_m + 2x_nx_m$$

This is difficult to compute directly in higher dimensions, especially as the number of dimensions increases.

### 3.3 Kernel Trick

We use the kernel trick instead of explicitly calculating the dot product in the higher-dimensional space. The kernel function allows us to directly compute the dot product in the higher-dimensional space without explicitly performing the transformation. The kernel for the transformation described above is:

$$K(x_n, x_m) = (1 + x_n^T x_m)^2$$

This kernel simplifies the calculation by giving us the same result as the inner product in the higher-dimensional space, but without the complexity of performing the actual transformation.

## 3.4 Examples of Kernels

There are several common types of kernels used in SVMs:

- **Linear Kernel**:
$$K_{\text{linear}}(x_n, x_m) = x_n^T x_m$$
This kernel is used when the data is already linearly separable.

- **Quadratic Kernel**:
$$K_{\text{quad}}(x_n, x_m) = (1 + x_n^T x_m)^2$$
This kernel corresponds to a quadratic feature mapping.

- **Polynomial Kernel** of degree $q$:
$$K_{\text{poly}}(x_n, x_m) = (1 + x_n^T x_m)^q$$
This kernel allows for higher-degree separability.

- **Radial Basis Function (RBF) Kernel**:
$$K_{\text{RBF}}(x_n, x_m) = \exp\left(-\gamma \|x_n - x_m\|^2\right)$$
The RBF kernel is a very useful kernel that can handle non-linearly separable data, especially when the data lies in higher-dimensional spaces.

## 3.5 RBF Kernel and Infinite Dimensions

The **RBF kernel** can map data to an infinite-dimensional space. The RBF kernel is defined as:

$$K_{\text{RBF}}(x, x') = \exp\left(-\gamma (x - x')^2\right)$$

Using the **exponential identity**:

$$e^a = 1 + a + \frac{a^2}{2!} + \frac{a^3}{3!} + \dots$$

we expand the RBF kernel to show that it maps data to an infinite-dimensional space. The expansion results in:

$$\exp(-\gamma x^2) \cdot \exp(-\gamma x'^2) = \sum_{k=0}^{\infty} (2\gamma)^k \, \frac{x^k (x')^k}{\sqrt{k!}}$$

Thus, the kernel can handle high-dimensional data by mapping it into a feature space where it becomes linearly separable.
This expansion demonstrates the infinite dimensionality of the feature space that the kernel maps the data to.

## 3.6 Underlying Mapping of the RBF Kernel

The underlying mapping for the RBF kernel can be shown as:

$$\varphi_{\text{RBF}}(x) = \exp(-\gamma x^2)$$

The result of this transformation is a high-dimensional vector, which can be written as:

$$\varphi_{\text{RBF}}(x) = \begin{bmatrix} 1/\sqrt{\gamma} \cdot x_1 \\ 1/\sqrt{\gamma} \cdot x_2 \\ 1/\sqrt{\gamma} \cdot x_3 \\ \vdots \end{bmatrix} \in \mathbb{R}^\infty$$

"This shows how $x$ is mapped to an infinite-dimensional space, enabling classification and regression in a richer feature space. The transformation makes it possible to separate the data linearly in this higher-dimensional space. The resulting feature space consists of an infinite series of dimensions that capture non-linear relationships between data points.
The kernel trick allows efficient computation of dot products in these higher-dimensional spaces, enabling SVMs to classify non-linearly separable data. The RBF kernel is especially useful, as it handles infinite-dimensional spaces, making it ideal for a wide range of machine learning problems."

## 4 Properties of the Kernel Matrix

The kernel matrix $K$ is an $N \times N$ matrix, where each element is computed using the kernel function $K(x_i, x_j)$. The matrix has the following properties:

$$K = \begin{pmatrix} \varphi(x_1)^T\varphi(x_1) & K(x_1, x_2) & \cdots & K(x_1, x_N) \\ \varphi(x_2)^T\varphi(x_1) & \varphi(x_2)^T\varphi(x_2) & \cdots & K(x_2, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi(x_N)^T\varphi(x_1) & \varphi(x_N)^T\varphi(x_2) & \cdots & \varphi(x_N)^T\varphi(x_N) \end{pmatrix}$$

This matrix contains all the evaluations of the kernel function between the data points, including their feature space mappings. For any two data points $x_i$ and $x_j$, we compute their dot product in the mapped feature space via $\varphi(x_i)$ and $\varphi(x_j)$.

### 4.1 Properties

The kernel matrix has two main properties:

1. **Symmetry**:

$$K(x_i, x_j) = K(x_j, x_i)$$

   This implies the matrix is symmetric, as the kernel function satisfies this property for any pair of points.

2. **Positive Semi-Definiteness**:

$$\lambda_1(K), \lambda_2(K), \ldots, \lambda_N(K) \geq 0, \quad \forall i \in \{1, 2, \ldots, N\}$$

   The eigenvalues of the kernel matrix are non-negative. This ensures that the kernel is a valid similarity measure between the data points in the feature space. It also means that for any vector $v$, $v^T K v \geq 0$.

This matrix and its properties are fundamental when using kernels in support vector machines (SVMs) to perform non-linear classification and regression tasks. The kernel function, particularly the RBF kernel, helps map data points into higher-dimensional spaces where they become linearly separable, which is vital for non-linear SVMs.

## Next Lecture

In the next lecture, we will explore:
1. Soft-Margin SVM
2. SVM Regression
3. Perceptron

## References

1. Support Vector Machines: Dealing with Non-Separable Data - Link

2. Lecture note From Cornell University-Link

3. SVM kernels and its type -Link