Support Vector Machine (SVM)

SVM

- Supervised learning methods for classification and regression. However, it is mostly used in classification problems.

- SVM finds the best line that separates the classes with maximum margin. i.e. The goal is to **find the best boundary (hyperplane)** that separates data points of different classes with the **maximum margin**.

- SVM searches for optimal hyperplane (i.e., decision boundary) separating the tuples of one class from another.
  - In general, for n features, a hyperplane is an (n−1)-dimensional surface.

SVM

- The **margin** is the **distance** between the hyperplane and the **closest data points** from either class.
  - SVM tries to **maximize** this margin - that's why it's often called a maximum margin classifier.

- Support vectors are the **data points that lie closest to the hyperplane**
  - These points are **critical** to defining the hyperplane.
  - The name SVM comes from these support vectors.

- SVM works well with higher dimensional data and thus avoids dimensionality problem.
  - Works well when number of features > number of samples
  - Memory efficient – uses only support vectors
  - Can handle non-linear data using kernels

SVM

- Although the SVM based classification (i.e., **training** time) is extremely slow, the result, is however highly accurate. Further, testing an unknown data is very fast.

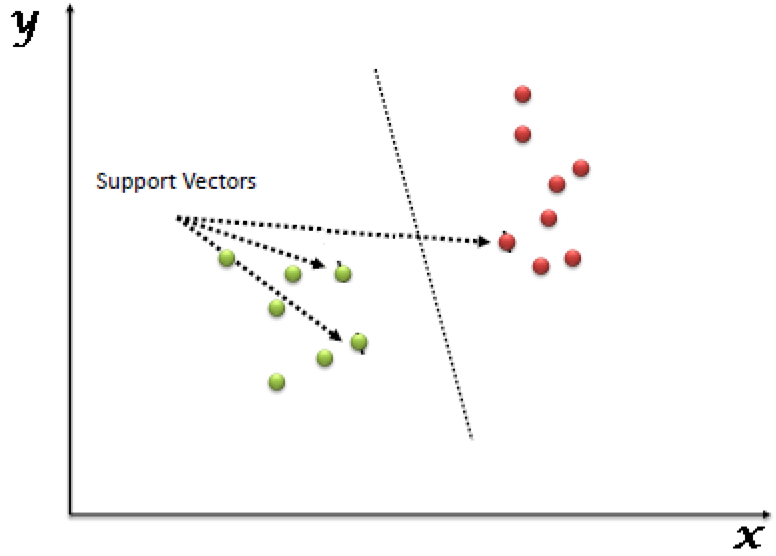
- SVM is less prone to **over fitting** than other methods. It also facilitates compact model for classification.

- **Cons**
  - Not ideal for very large datasets.
  - Doesn’t perform well with overlapping classes.
  - Choosing the right kernel and tuning hyperparameters (C, gamma) can be tricky.

SVM

- We plot each data item as a point in n-dimensional space (n is the number of features) with the value of each feature being the value of a particular coordinate.

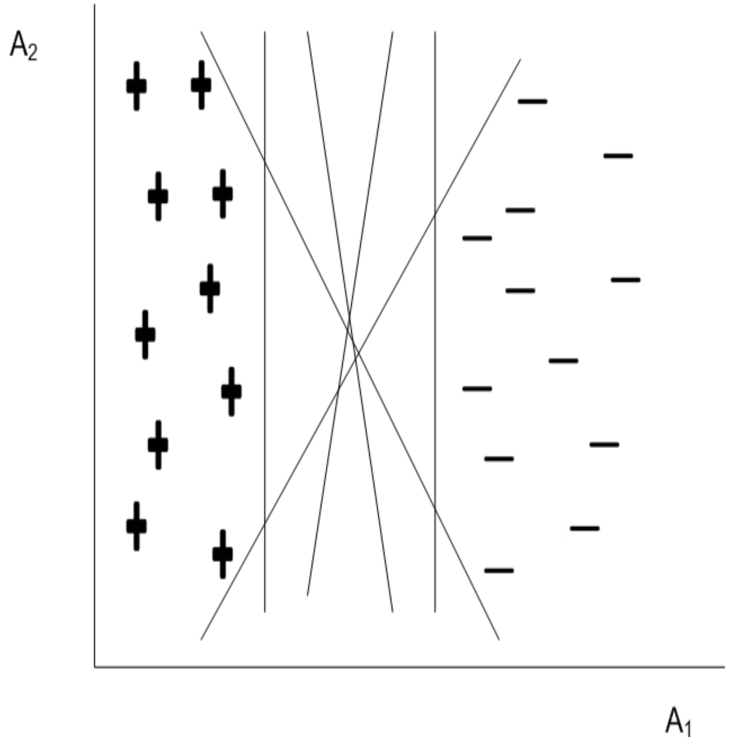
- We perform classification by finding the hyper-plane that differentiates the two classes very well.



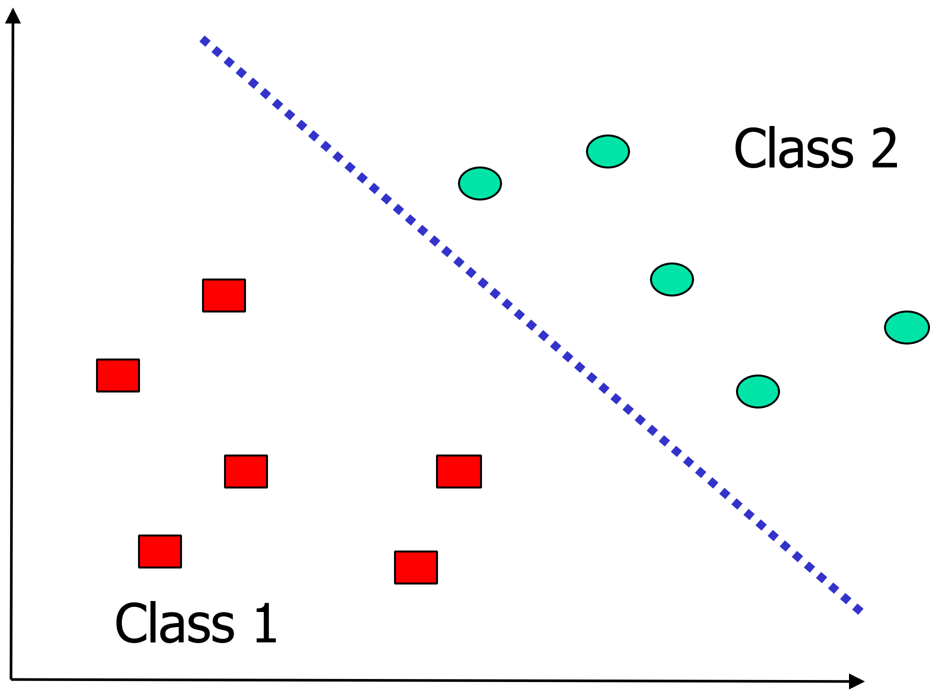
SVM

- A training data D $=\{t_1, t_2, \ldots t_n\}$ with a set of n tuples, which belong to two classes either + or - and each tuple is described by two attributes say A_1, A_2.

- The data is linearly separable, we can find a hyperplane (in this case, it is a straight line) such that all +’s reside on one side whereas all -’s reside on other side of the hyperplane.



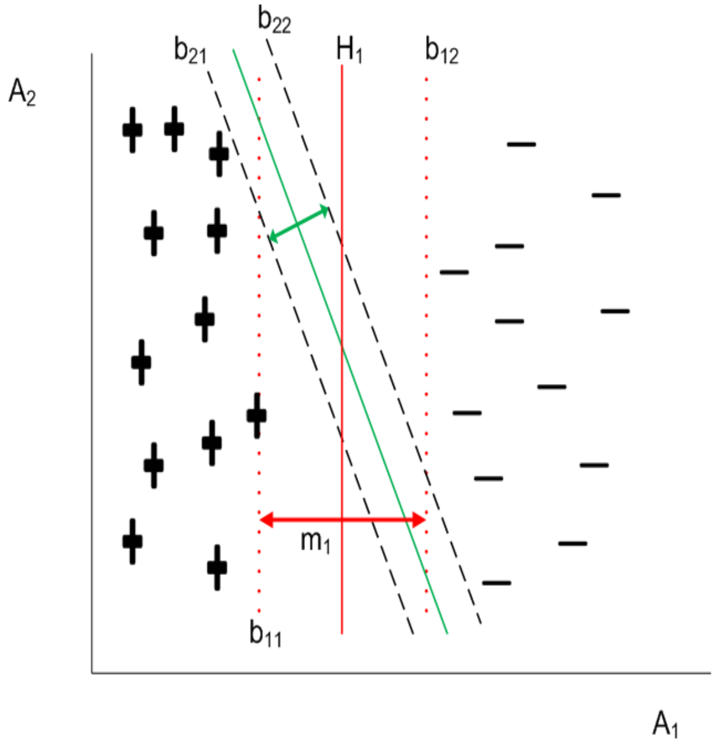
Two Class Problem: Linear Separable Case



- Many decision boundaries can separate these two classes
- Which one should we choose?

SVM

- Two hyperplanes H_1 and H_2 have their own boundaries called decision boundaries (denoted as b_{11} and b_{12} for H_1 and b_{21} and b_{22} for H_2).
- The distance between the two decision boundaries of a hyperplane is called the **margin**. So, if data is classified using Hyperplane H_1, then it is with larger margin than using Hyperplane H_2.
- The margin of hyperplane implies the error in classifier. In other words, the larger the margin, lower is the classification error.



Margin

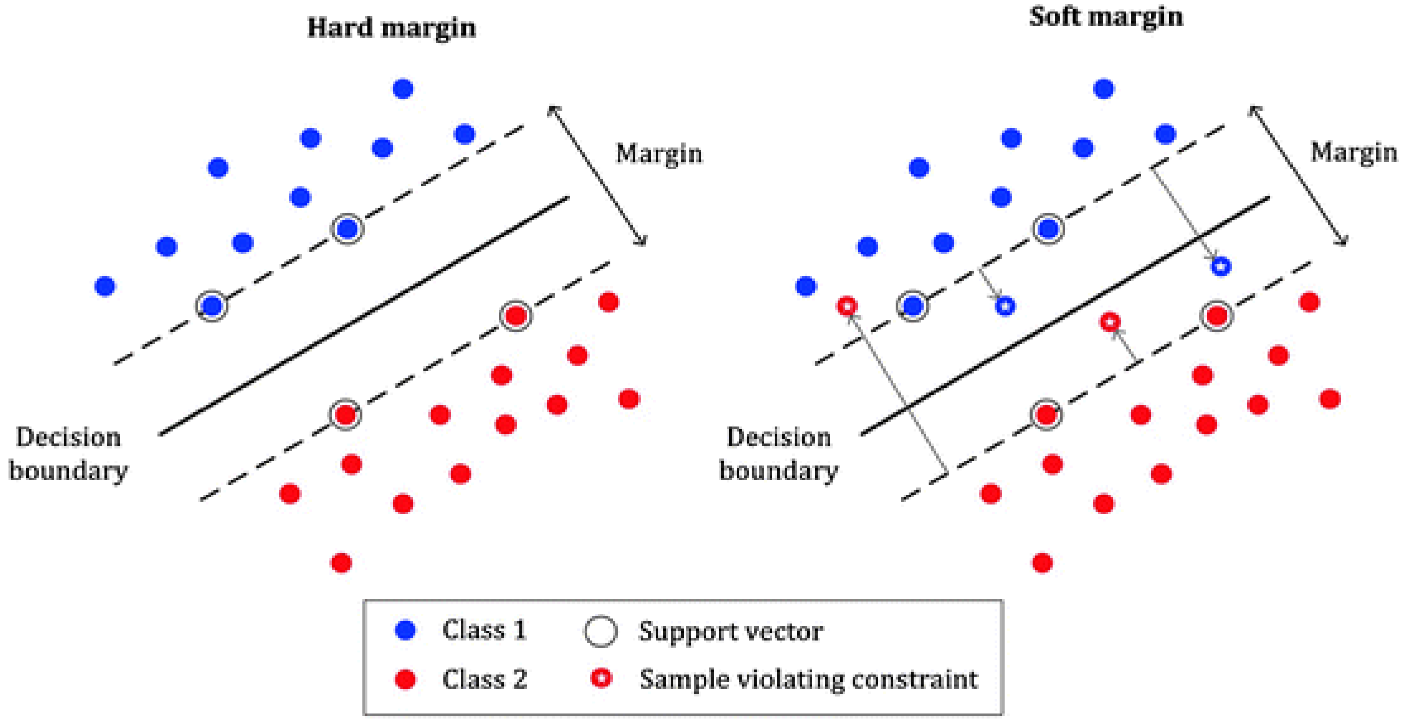
- The distance of the vectors from the hyperplane is called the **margin** which is a separation of a line to the closest class points. We choose a hyperplane that maximizes the margin between classes.

- Margin can be sub-divided into two types
  - **Hard Margin** – If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.

  - **Soft Margin** – As most of the real-world data are not fully linearly separable, we will allow some margin violation to occur which is called soft margin classification.

Margin

- It is better to have a large margin, even though some constraints are violated.
  - Margin violation means choosing a hyperplane, which can allow some data points to stay on the incorrect side of the hyperplane.



SVM Analogy

- Consider a classroom having students from two branches sitting together
  - CSE (Computer Science & Engineering)
  - BI (Bioinformatics)

- **Goal**: To separate two groups as clearly as possible.

Soft Margin SVM

- Some students already sit in each other’s area - maybe due to being friends or arriving late.

- No perfect separation anymore

- Allow a few students to be on the "wrong" side, but with penalties.

- **Soft margin SVM** — a few misclassifications are allowed, but try to minimize them.

Hard Margin SVM

- To draw a perfect divider (hyperplane) that completely separates the two groups without any misclassification.

- It ensures that all students are on the correct side of the line.

- It maximizes the space (margin) between the divider and the nearest student from each side.

- Used when data is
  - 100% clean and completely separable.
  - Noise is minimal or non-existent.
  - Misclassification is not acceptable at all (strict settings).

Applying SVM Concepts

- **Hyperplane** (Divider Line): An imaginary line or physical rope you stretch across the classroom to divide the two groups.

- **Goal:** To find a position for the divider that best separates the two groups without bias toward one side.

Applying SVM Concepts

- **Support Vectors** (Closest Students): These are the students sitting nearest to the divider from each branch.
  - These positions directly influence where you place the divider.
- Moving the divider slightly closer to one group, the nearest students would start crossing sides — so these positions ‘**support**’ the boundary.

Applying SVM Concepts

- **Margin** (Buffer Zone): Place divider so that distance to the nearest student from either branch is as wide as possible.

- **Goal:** This gives both groups enough breathing space and reduces the chance of future confusion (i.e., avoids misclassification).

Linear SVM

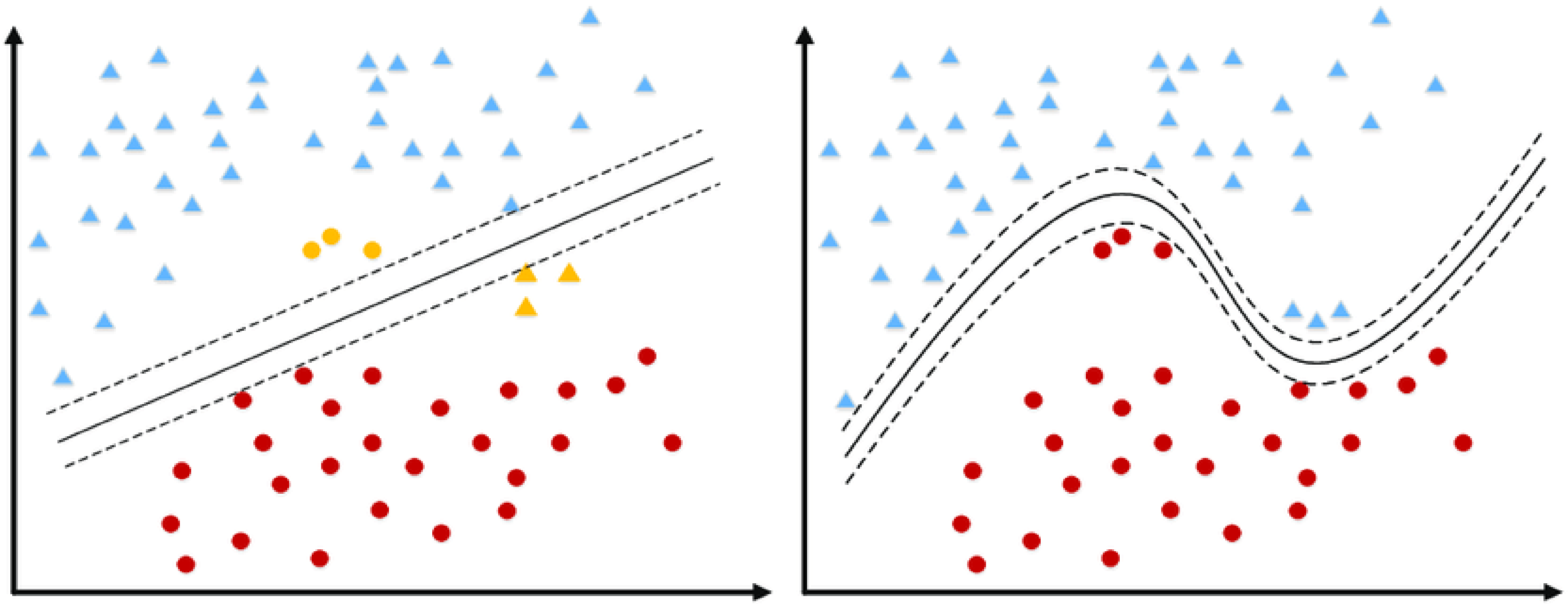
- SVM which is used to classify data which are linearly separable is called linear SVM.

- In other words, a linear SVM searches for a hyperplane with the maximum margin.

- This is why a linear SVM is often termed as a **maximal margin classifier** (MMC).

SVC vs SVM

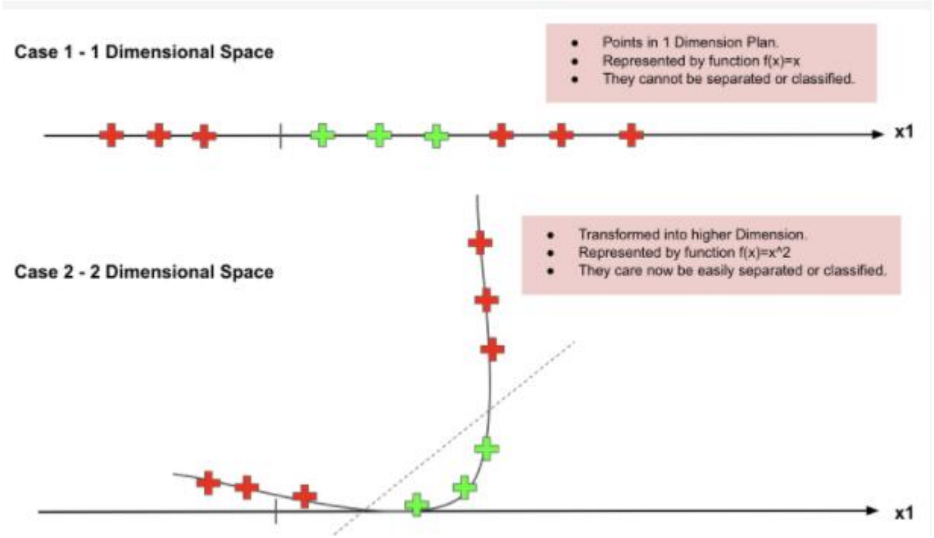
SVM is a general machine learning algorithm used for both linear and non-linear classification. SVC (Support Vector Classifier) is a specific implementation of SVM for classification tasks, which can use both linear and non-linear kernels.



A) Linear Separation B) Non-linear Separation

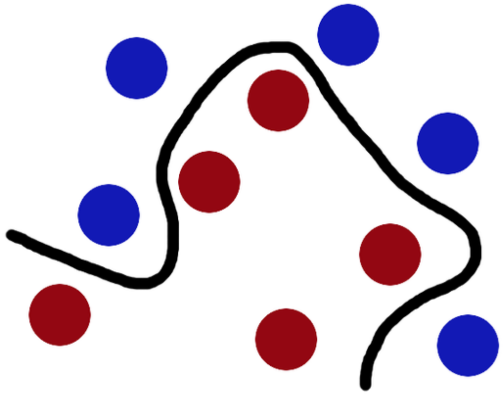
Kernel Trick

SVM uses a technique called the **kernel trick**, which involves kernel functions that map the input data from a lower-dimensional space into a higher-dimensional feature space. This transformation helps convert non-linearly separable problems into linearly separable ones in the new space, allowing the SVM to find a hyperplane that effectively separates the data.



Why Kernels

- The red and blue balls cannot be separated by a straight line as they are randomly distributed



- A **kernel** (kernel trick) is
  - A method of using a linear classifier to solve a non-linear problem. It entails transforming linearly inseparable data to linearly separable one.
  - The kernel function is applied on each data instance to map the original non-linear observations into a higher-dimensional space in which they become separable.

What is Kernel Trick?

- The **kernel trick** is a technique used in SVM.

- It allows the algorithm to handle non-linear data by transforming into a higher-dimensional space.

- **Why it's a trick?**

  Because SVM can do this transformation **without explicitly computing** the new higher-dimensional coordinates - it uses a mathematical shortcut (the kernel function).

Why is Kernel Trick Needed?

- In the **original (low-dimensional) space**, some datasets cannot be separated by a straight line (or hyperplane).

- By mapping the data into a **higher-dimensional space**, the data may become linearly separable.

  E.g.: A circular dataset in 2D cannot be separated by a line, but in 3D, it may be separated by a plane.

What does a kernel function do?

- A **kernel function** computes the dot product between two data points in the higher-dimensional space, without ever having to transform the data explicitly.

- This saves time and computation - especially when the feature space is very high-dimensional or even infinite.

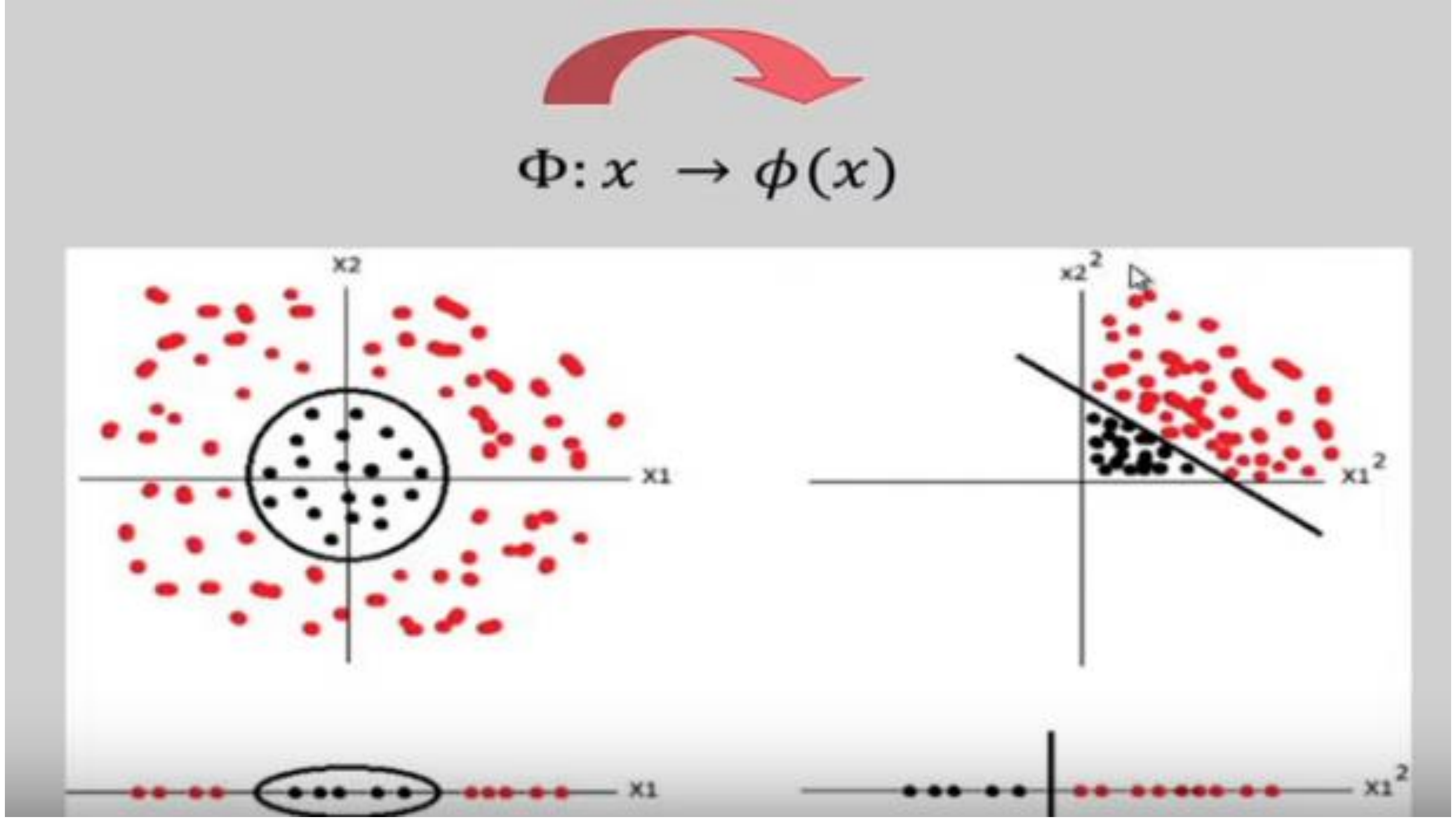
- **Benefits**
  - In high-dimensional space, complex patterns become simpler and easier to separate.
  - Helps SVM in creating powerful, flexible models that perform well on non-linear classification problems.

Various Kernel Functions

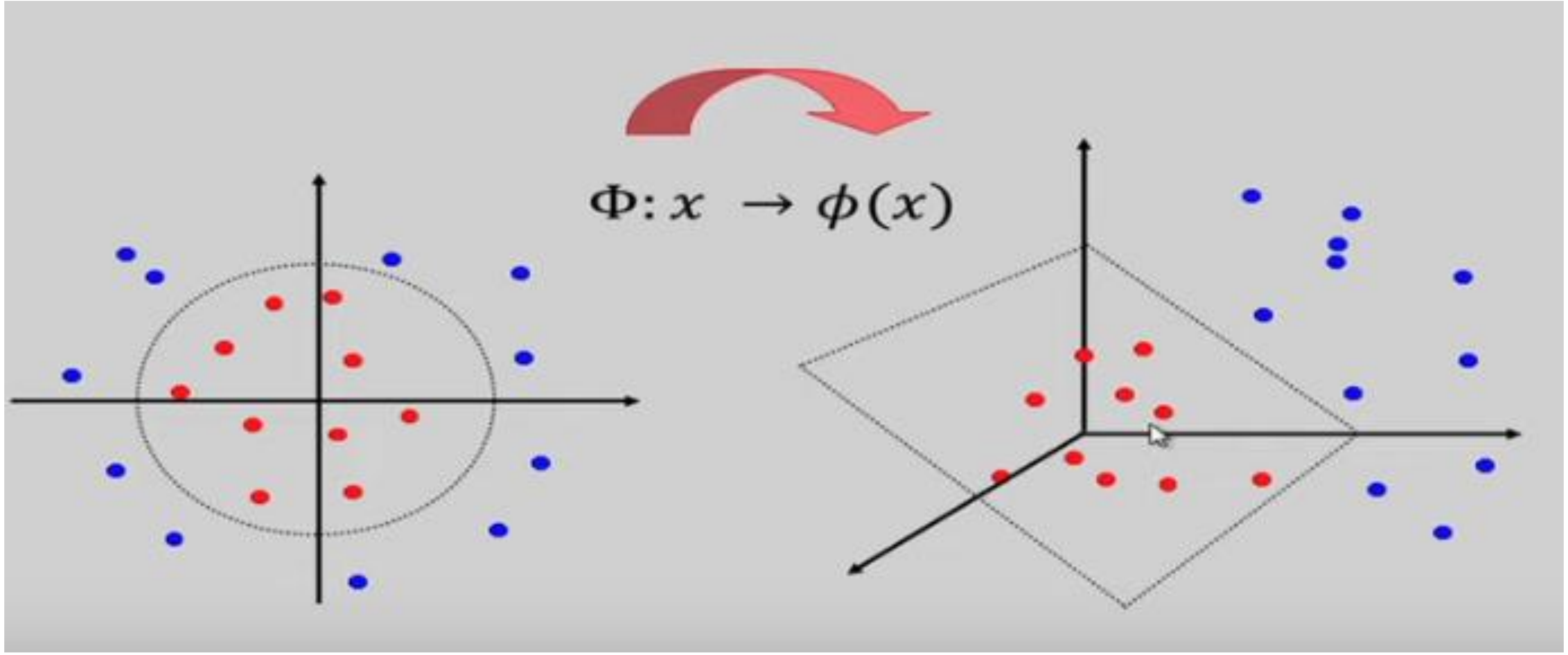
- **LK** is the fastest and works well when features are already meaningful.
- **PK** adds flexibility, depending on the degree you choose.
- **RBF** is the most commonly used because of its ability to handle complex patterns.
- **SK** is rarely used and sensitive to parameters.
- PCK provides faster training times and is used in non-linear SVMs where kernel computation is very expensive (e.g. RBF or PK).

Kernel Type	Formula	Usage
Linear Kernel	$K(x, x') = x^T x'$	Used for linearly separable data, with a straight-line decision boundary.
Polynomial Kernel	$K(x, x') = (x^T x' + c)^d$	Used for non-linearly separable data, when interactions between features exist.
RBF (Gaussian) Kernel	$K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$	Used for complex, non-linear decision boundaries. Most common kernel.
Sigmoid Kernel	$K(x, x') = \tanh(\alpha x^T x' + c)$	Used in neural networks, less common in SVMs. Can model non-linear boundaries.
Precomputed Kernel	$K(x, x') = \text{Precomputed matrix of similarity}$	Used when kernel values are precomputed or custom kernels are required.

Why Kernels



Why Kernels



Pros and cons of SVM

Pros:

- It is really effective in the higher dimension.
- Effective when the number of features are more than training examples.
- Best algorithm when classes are separable.

Cons:

- For a larger dataset, it requires a large amount of time to process.
- Does not perform well in case of overlapped classes.
- Selecting the appropriate kernel function can be tricky.

SVM

Pros & Cons

Pros of SVM	Cons of SVM
Effective in High-Dimensional Spaces: SVM performs well with high-dimensional data, making it suitable for tasks like text classification and gene data.	**Computationally Expensive**: Training SVM can be time-consuming, especially with large datasets.
Versatile with Kernels: The kernel trick allows SVM to handle non-linear data by mapping it to a higher-dimensional space without explicitly computing the transformation.	**Sensitive to Kernel Choice**: The performance depends on selecting the appropriate kernel and tuning its parameters.
Robust to Overfitting: By maximizing the margin between classes, SVM reduces the risk of overfitting, especially in high-dimensional spaces.	**Not Ideal for Large Datasets**: SVM can struggle with very large datasets due to its quadratic or cubic time complexity.
Works Well with Clear Margin of Separation: SVM performs excellently when there's a distinct margin separating the classes.	**Difficult to Interpret**: SVM models, especially with non-linear kernels, are not easy to interpret compared to simpler models like decision trees.
Global Optimality: SVM guarantees the global optimal solution (as long as the problem is convex), ensuring the best possible model.	**Requires Tuning**: SVM requires careful tuning of parameters like the regularization parameter and kernel-specific values.
Memory Efficient: SVM uses only the support vectors to make decisions, which reduces memory consumption.	**Poor Performance on Noisy Data**: SVM can perform poorly when classes overlap or contain noise.

A few random notes

- In machine learning, there's always a risk of finding a local minimum instead of the global minimum. A local minimum could be a good solution but not the best overall. However, SVM's optimization problem is structured in a way that guarantees finding the global optimal solution when using the right kernel.
  - A global optimum (minimum or maximum) is the **best possible solution** over the entire domain.
  - A local optimum is a solution that is better than all nearby solutions (neighbourhood) — but **not necessarily the best overall**.

- SVM were introduced by Vladimir **Vapnik** and Alexey Chervonenkis in the 1960s as part of their work on statistical learning theory. The modern SVM, using the kernel trick for non-linear classification, was developed in 1992 by **Vapnik** and Corinna Cortes.

Thanks.