# SUPPORT VECTOR MACHINE

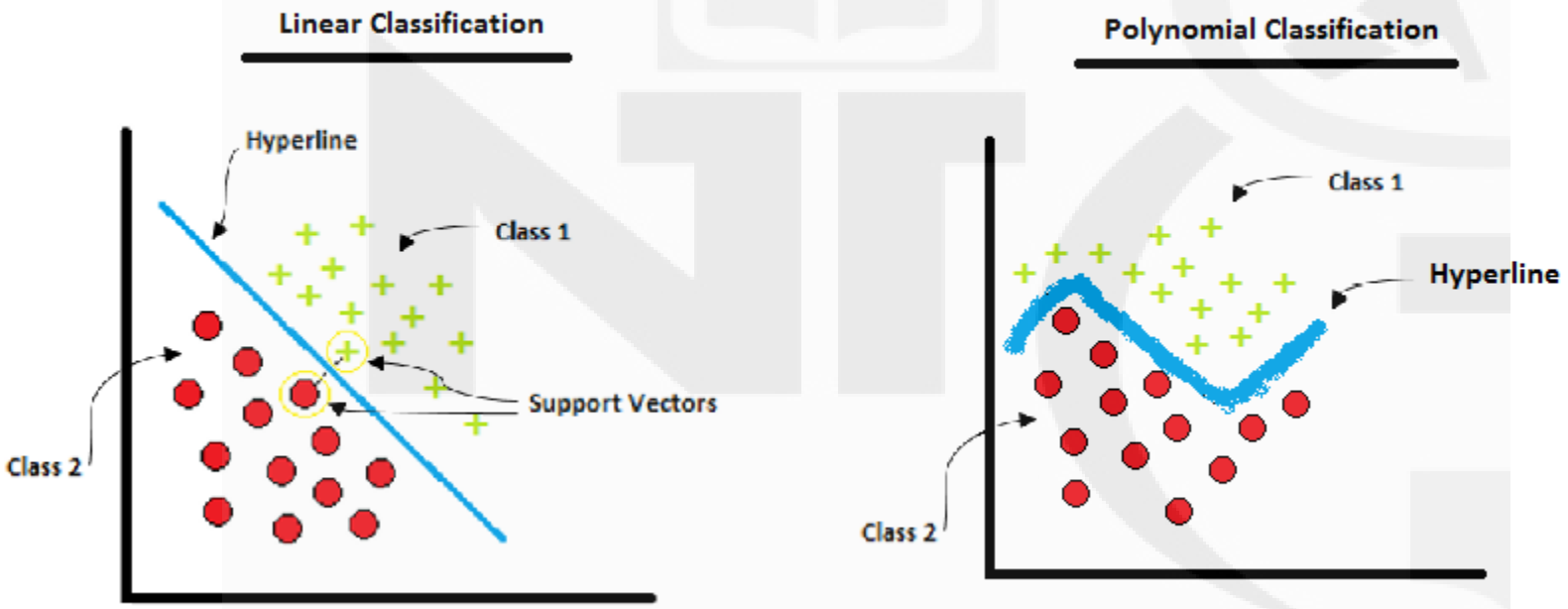
-MUKESH KUMAR

# Basic Idea

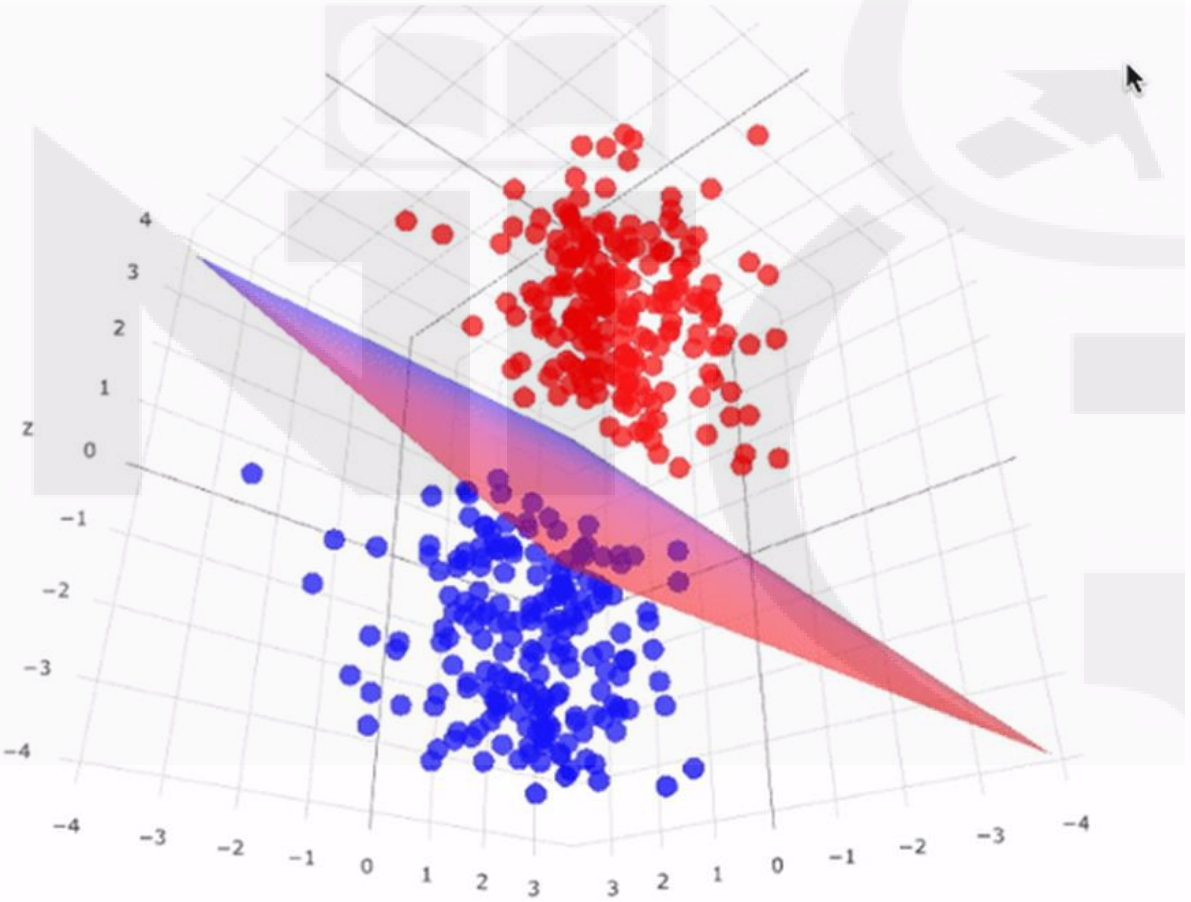
- **Classification:** The goal of an SVM is to find the best decision boundary (hyperplane) that separates different classes in the feature space. For example, in a 2D space, this boundary would be a line, while in a 3D space, it would be a plane.

- **Hyperplane:** A hyperplane is a decision boundary that separates the data into different classes. SVM tries to find the hyperplane that maximizes the margin between the classes.

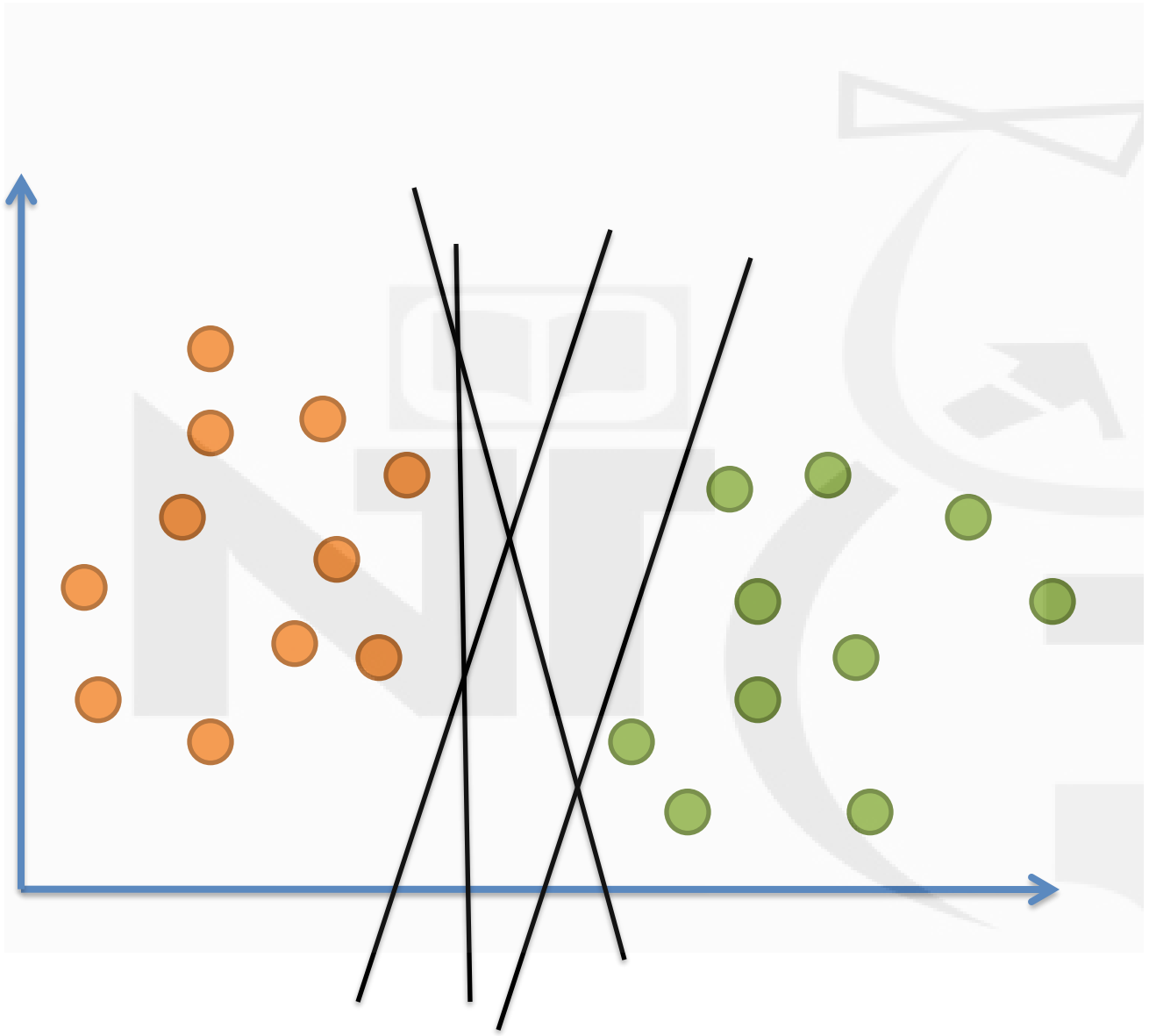
# SVM finds the decision boundary (Hyperplane)

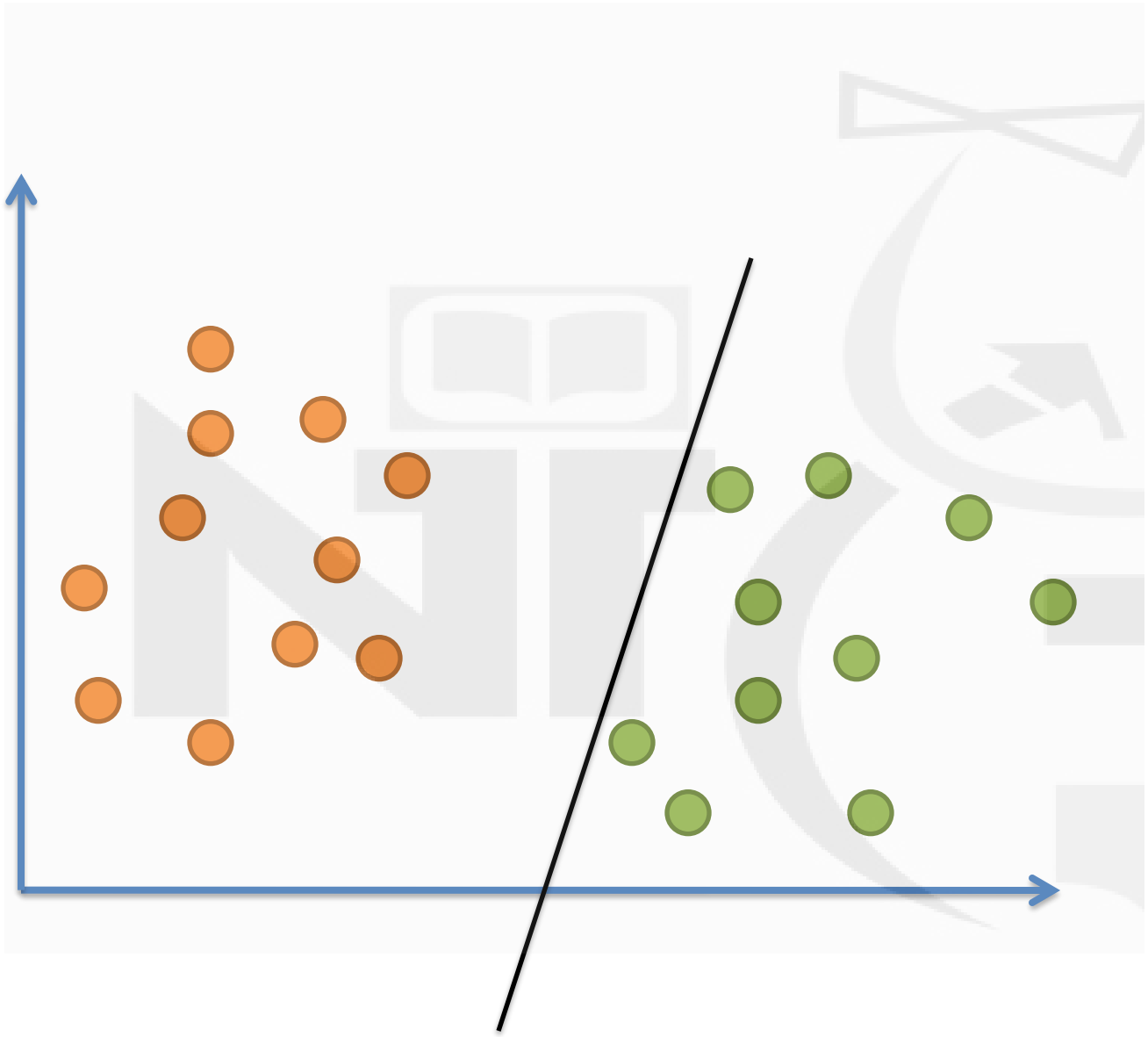


# SVM finds the decision boundary (Hyperplane)

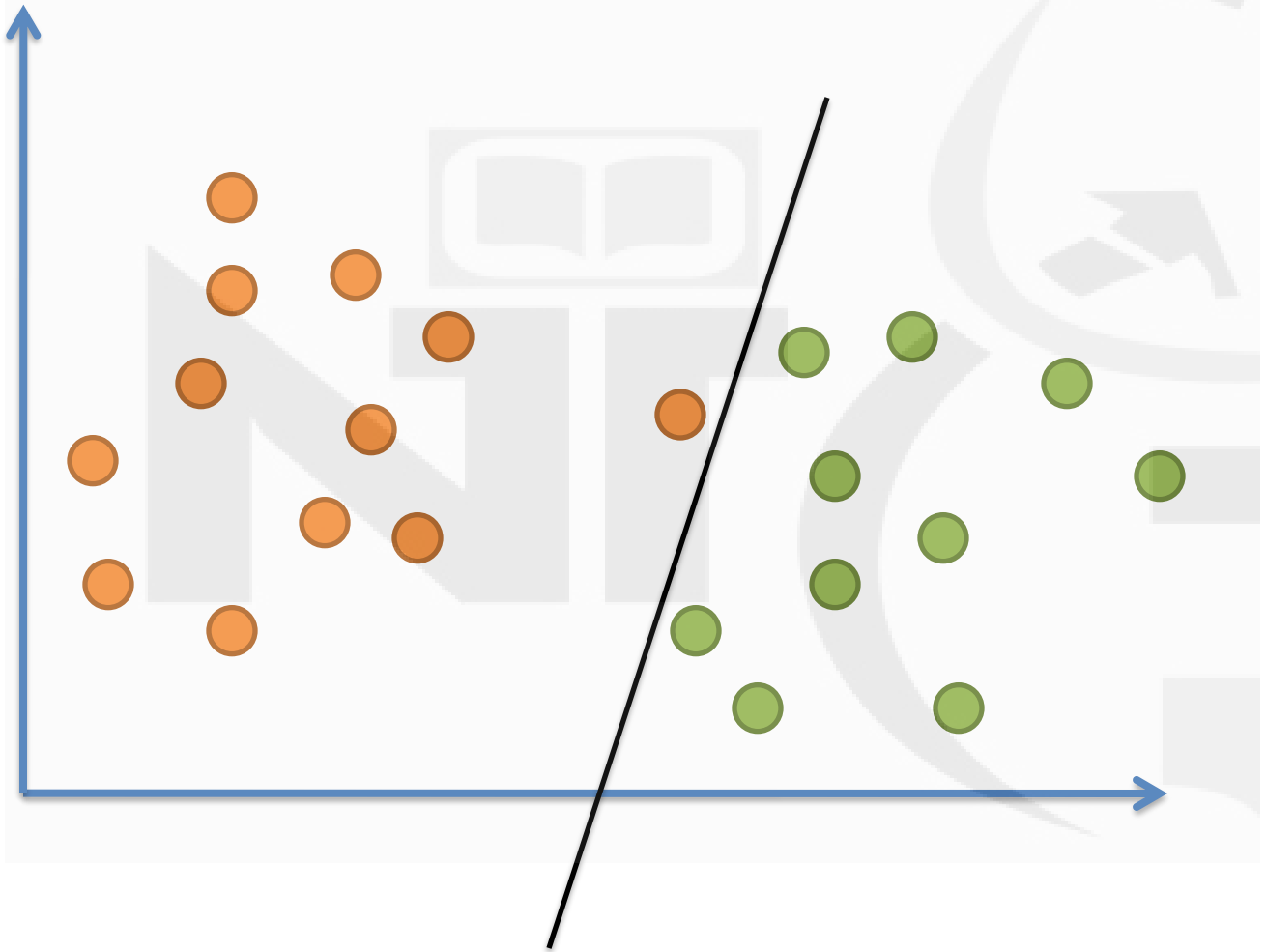


# UNDERSTANDING HOW SVM WORKS?



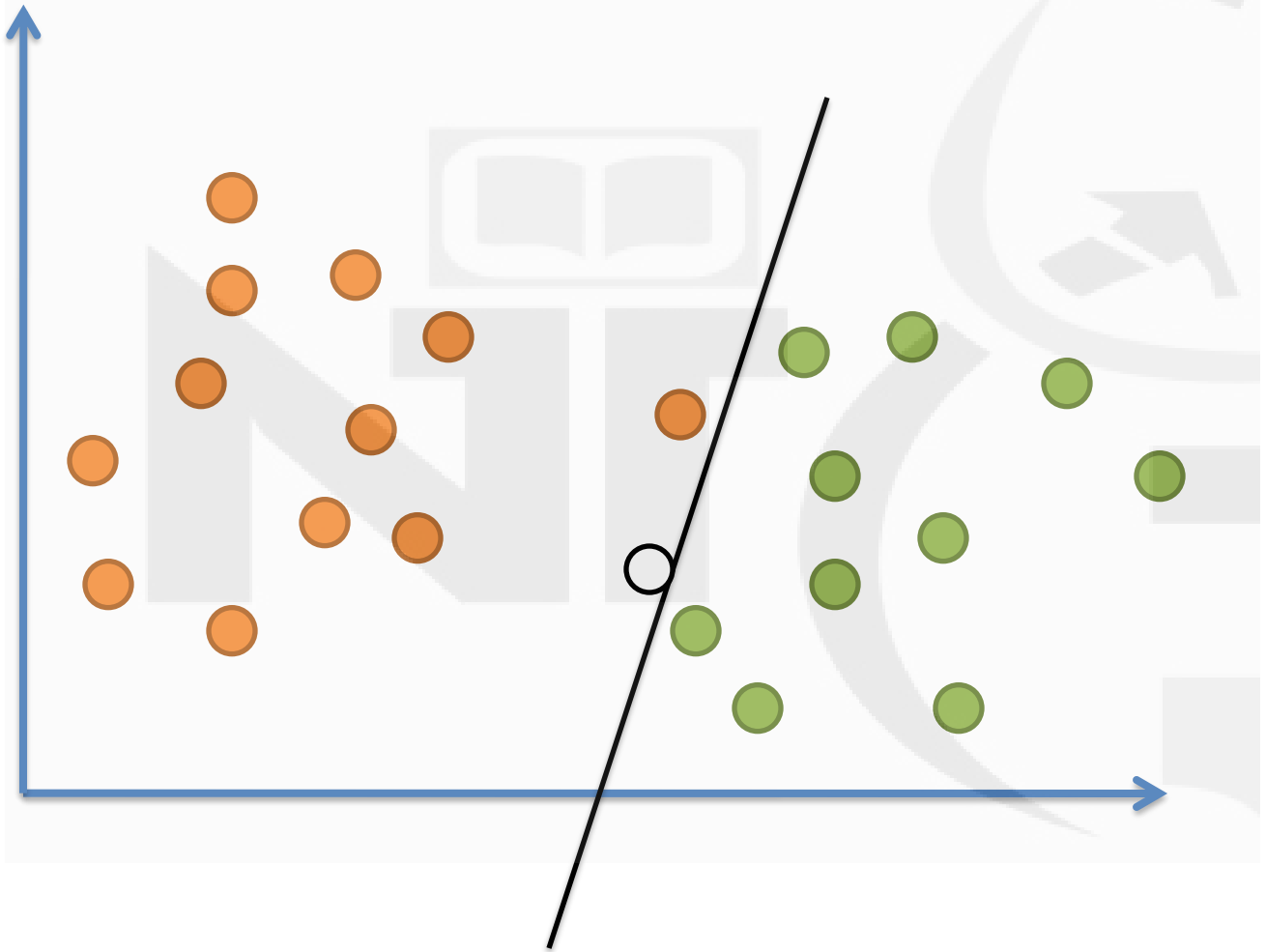


# Lets say we have an outlier in orange class and we move the threshold to the right



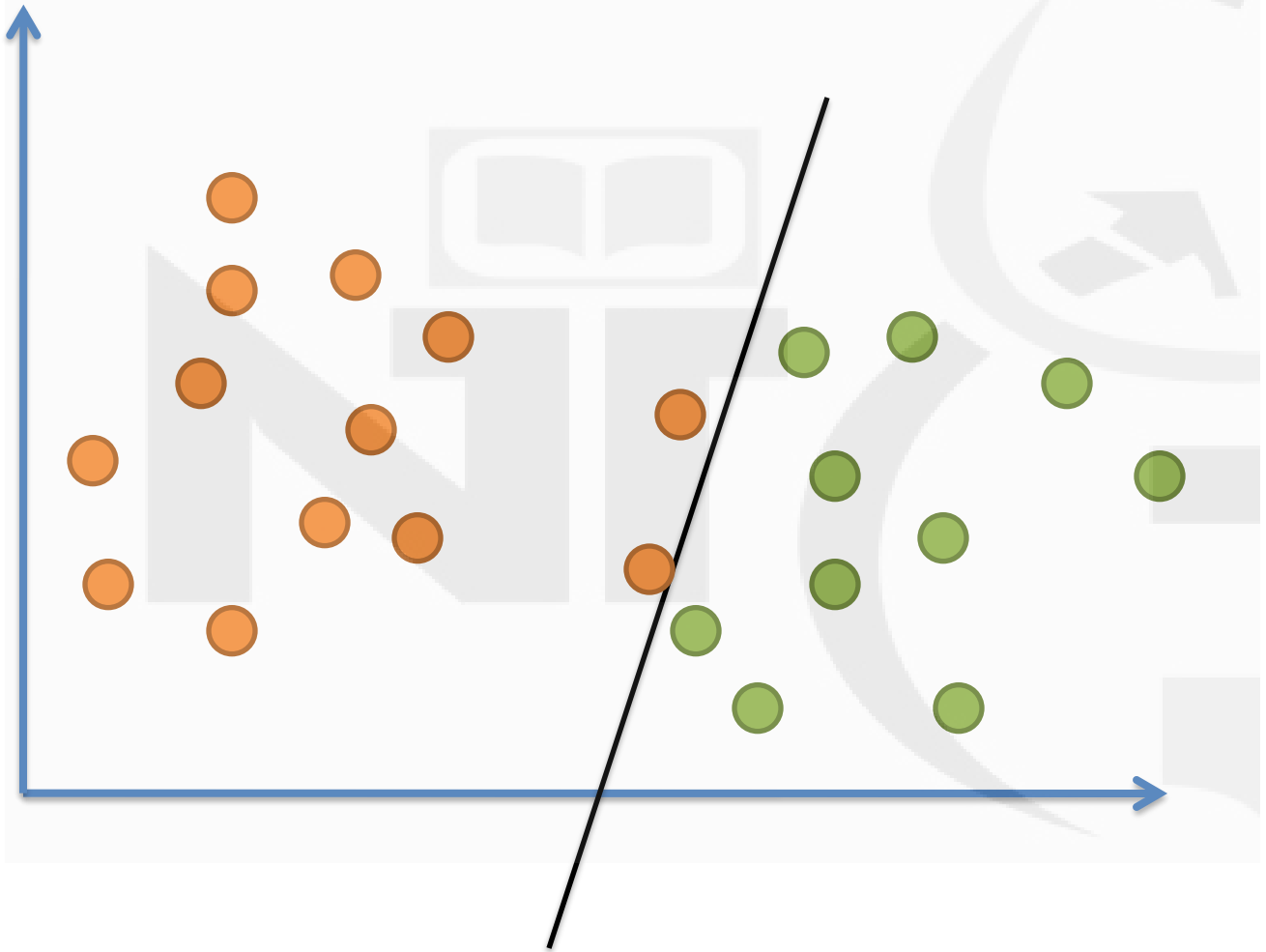
This kind of model will be very sensitive

# Lets say we got a new green data point



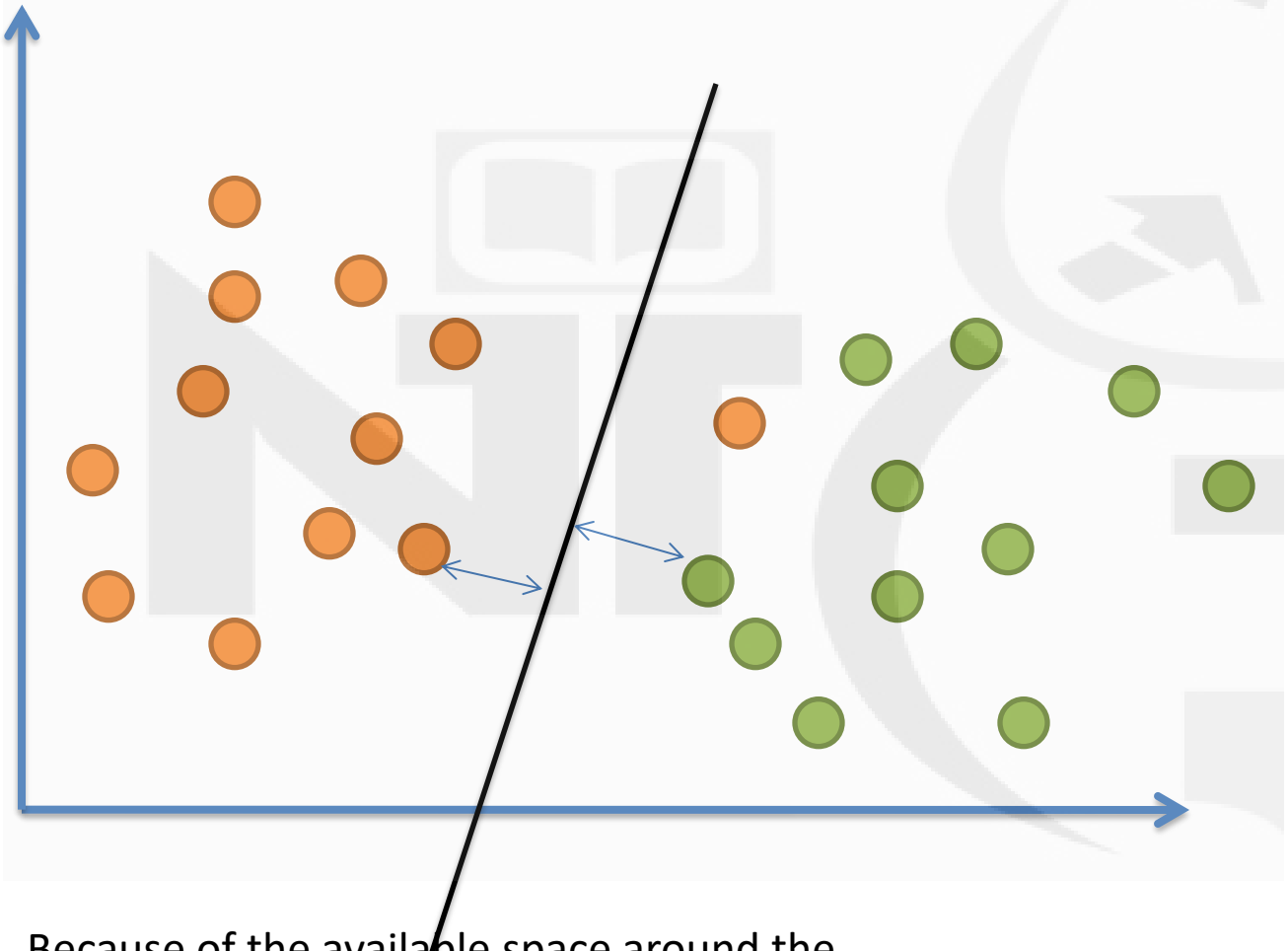
Though this  new data point is nearer to green its classified as orange

# Lets say we got a new green data point



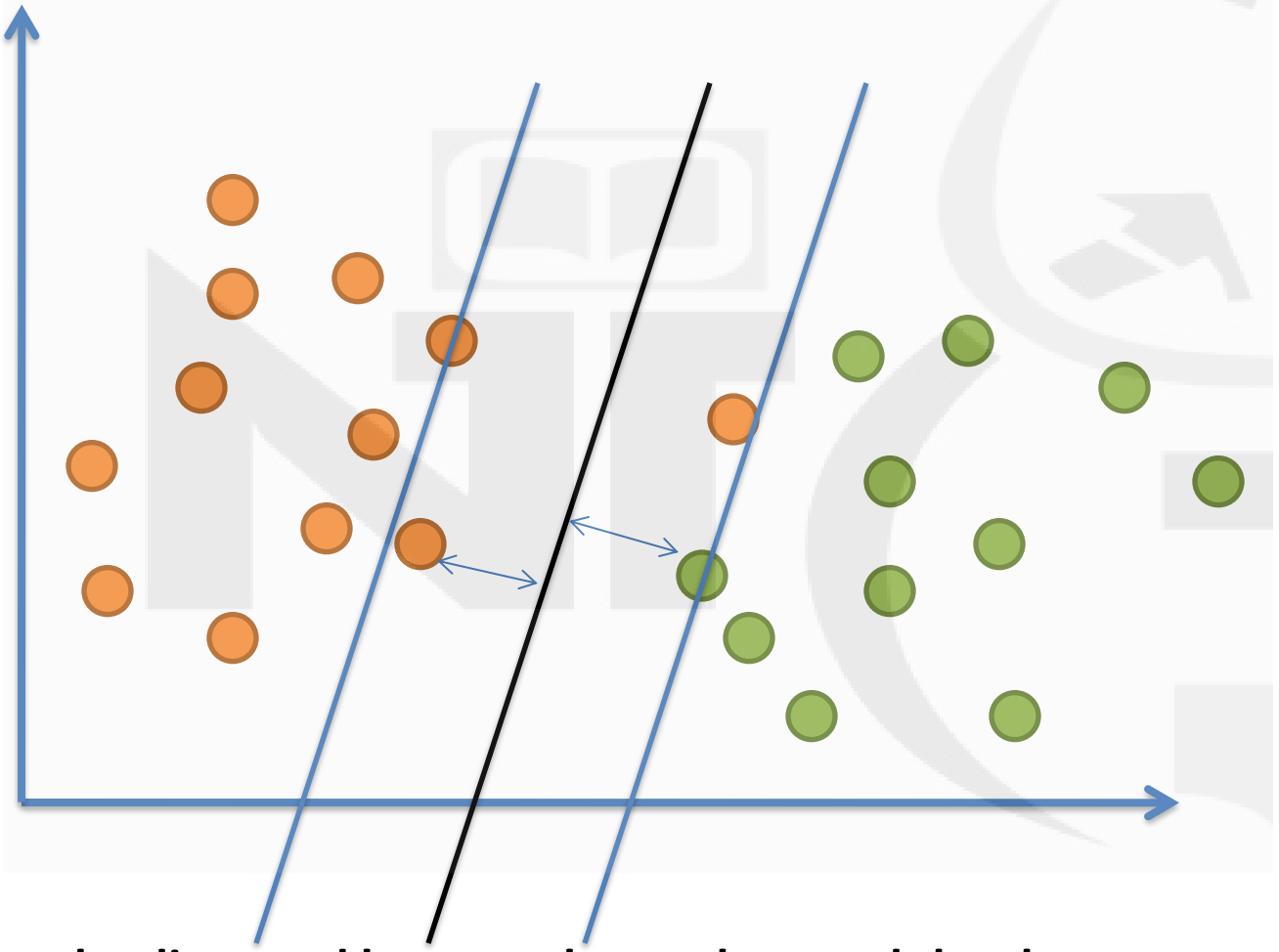
Though this  new data point is nearer to green its classified as orange

makes the model flexible, allowing it to better adapt to real-world data and unseen variations



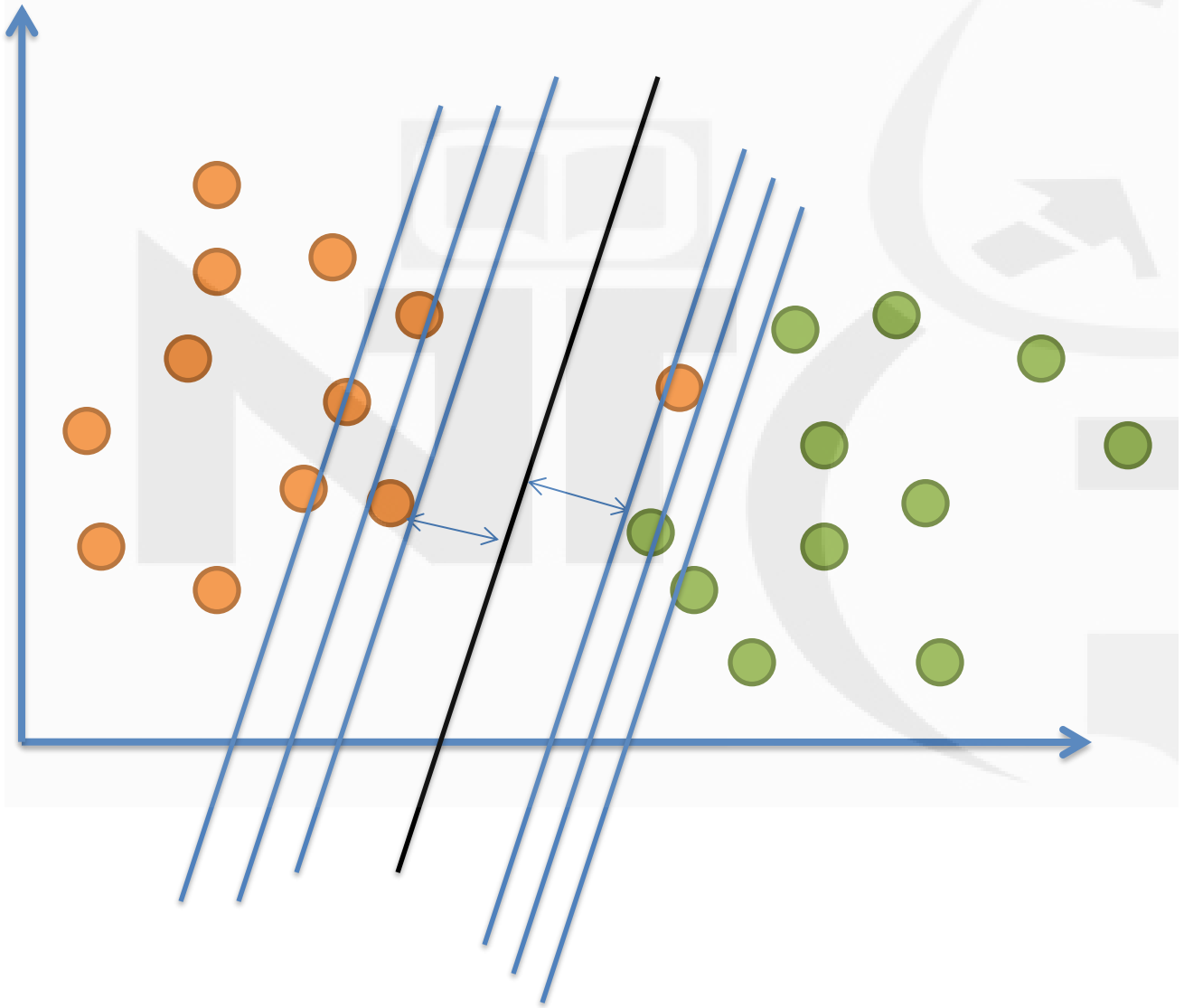
Because of the available space around the hyperplane model now correctly classified the earlier orange point to green now

# Soft margins



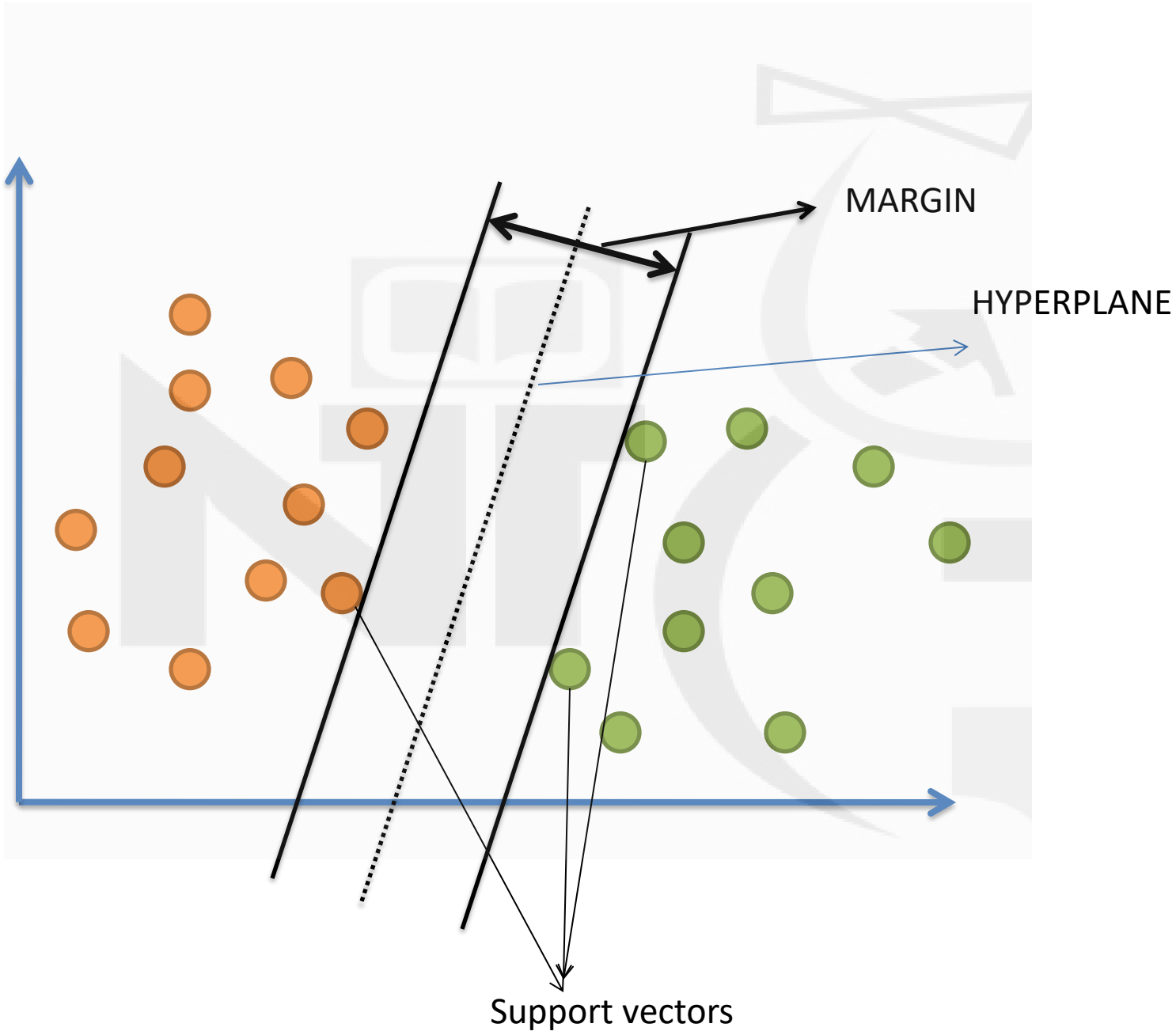
the distanced between hyperplane and the closest point from each class is call soft margin

# Many margins are posible



# We use cross-validation to find the best margin

- We choose many margins and for each margin we check how many misclassification and how many correct classifications are done within soft margin
- The one with least misclassification is selected


MARGIN
HYPERPLANE
Support vectors

# Key Concepts

- Hyperplane
- Margin
- Support Vectors
- Linear vs. Non-Linear

# Hyperplane

- A hyperplane is a decision boundary that separates data points of different classes.

- In two dimensions, it is a line; in three dimensions, it is a plane; and in higher dimensions, it is referred to as a hyperplane.

# Support Vectors

- Support vectors are the data points that are closest to the hyperplane.
- These points are critical as they determine the position and orientation of the hyperplane.
- The SVM algorithm focuses on these points to create the optimal decision boundary.

# Margin

- The margin is the distance between the hyperplane and the nearest data points from either class (the support vectors).
- A larger margin indicates better generalization to unseen data.

# How SVM Works

**Training the Model**:

- The SVM algorithm identifies the optimal hyperplane by maximizing the margin between the classes. This involves solving a quadratic optimization problem.

**Kernel Trick:**

- SVM can handle non-linear data by using kernel functions. Kernels transform the input space into a higher-dimensional space, making it easier to find a linear separation.

- **Common kernels include:**
  - Linear Kernel: For linearly separable data.
  - Polynomial Kernel: For polynomial decision boundaries.
  - Radial Basis Function (RBF) Kernel: For non-linear data, which maps input space into infinite dimensions.

- **Prediction**:
  - Once trained, the SVM model can classify new data points by determining which side of the hyperplane they fall on.

# Advantages

- Effective in high-dimensional spaces.
- Works well with clear margin of separation.
- Versatile due to the use of different kernel functions.

# Disadvantages

- Not suitable for very large datasets due to high training time.
- Performance can degrade with noisy data.
- Requires careful tuning of parameters (e.g., kernel type, regularization).

# Applications of SVM

- SVMs are widely used in various applications, including:Text Classification: Spam detection, sentiment analysis.
- Image Recognition: Face detection, handwriting recognition.
- Bioinformatics: Gene classification, protein sorting.

# HYPERPARAMETERS REVISITED

# Hard Margin

- In Support Vector Machines (SVM), **hard margin** refers to a specific type of SVM that enforces a **strict separation** between classes without allowing for any misclassification of data points. It is used when the data is **linearly separable**, meaning that there exists a clear boundary that can perfectly separate the two classes without any overlap or error

# Hard Margin

**Perfect Separation**:

- In a hard margin SVM, the goal is to find a hyperplane (in 2D, this is a line) that separates the data points of the two classes perfectly.
- No data points are allowed to fall on the wrong side of the hyperplane, meaning **no misclassification** is tolerated.

# Limitations of Hard Margin SVM:

- **Sensitive to Outliers**:
  - Since hard margin SVM does not allow any misclassification, it is very sensitive to **outliers** or noisy data.
  - If even one data point is on the wrong side of the decision boundary (due to noise or an outlier), the model will fail to find a solution.
- **Inapplicable to Non-Separable Data**:
  - Hard margin SVM cannot be used if the data is not linearly separable. In real-world applications, most datasets are not perfectly separable.

# Soft Margin

**Soft Margin SVM: A More Flexible Approach**

- To overcome the limitations of hard margin SVM, **soft margin SVM** was introduced. Instead of requiring a perfect separation, soft margin SVM allows for **some misclassifications** by introducing a **slack variable** that penalizes misclassified points.

# Soft Margin

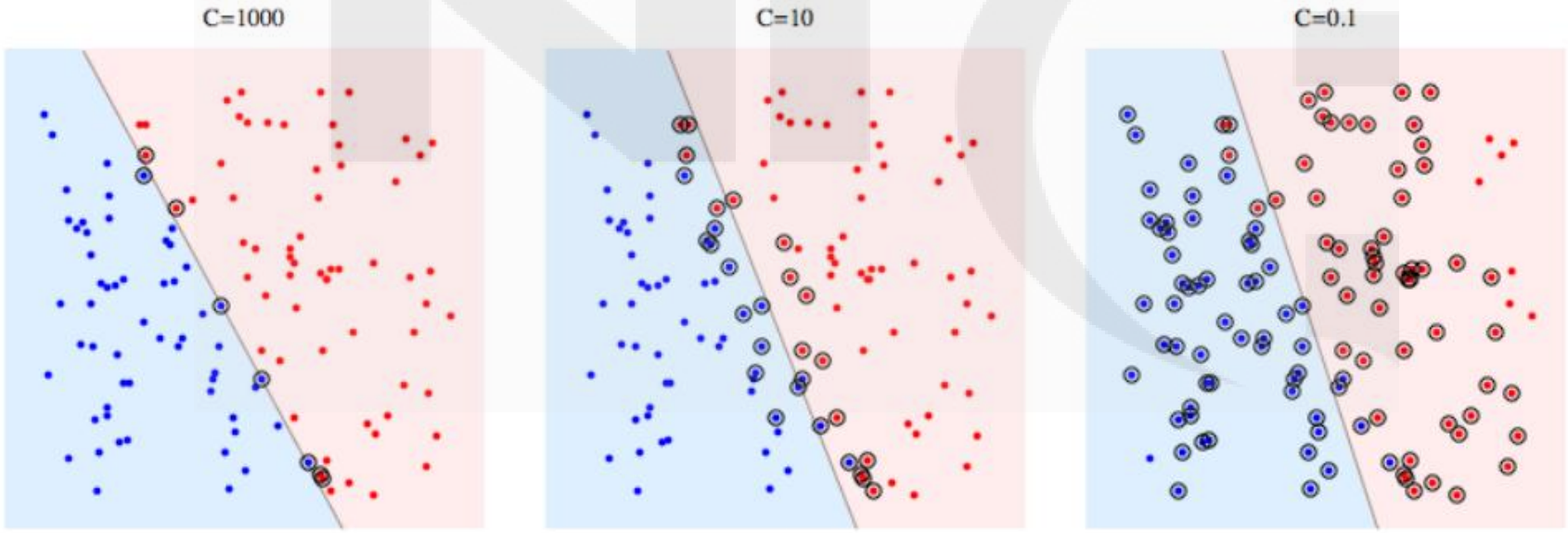
In soft margin SVM:

- A penalty is imposed for points that are either misclassified or lie within the margin.
- The trade-off between maximizing the margin and allowing misclassifications is controlled by the **C parameter**:
  - **High C**: Enforces a stricter separation, similar to hard margin.
  - **Low C**: Allows more flexibility, leading to a softer margin with potential misclassifications, but better generalization.

# Soft Margin

- In most practical applications, **soft margin SVM** is preferred since real-world data is rarely perfectly separable

- Circled points show support vectors. You can see that decreasing C causes classifier to sacrifice linear separability in order to gain stability, in a sense that influence of any single datapoint is now bounded by C.



- For hard margin SVM, support vectors are the points which are "on the margin". In the picture above, C=1000 is pretty close to hard-margin SVM, and you can see the circled points are the ones that will touch the margin (margin is almost 0 in that picture, so it's essentially the same as the separating hyperplane)

# NON-LINEAR SVM

# Non-Linear SVM

- Non-linear Support Vector Classification (SVC) is a powerful technique in machine learning used to classify data that isn't linearly separable.

- The primary difference between linear and non-linear SVC lies in the type of decision boundaries they create.

- While linear SVC tries to separate classes using a straight line (or a hyperplane in higher dimensions), non-linear SVC can use more complex curves and surfaces.

# Non-Linear SVC  Key Concepts

- The Kernel Trick
- Common Kernel Functions

# Kernel Trick

- Non-linear SVC leverages a technique called the **kernel trick**.
- The idea is to map the original input features into a higher-dimensional space where a linear separator (hyperplane) can be found, even if the data isn't linearly separable in the original space.
- The mapping itself doesn't require an explicit transformation; instead, it uses kernel functions that implicitly compute the similarity in this higher-dimensional space.
- Notebook: 1.DataTransformation_HigherDimensions, 4.Kernel Trick SVM

# Common Kernels

- *Notebook : Common_Kernel_Functions*

1. **Linear Kernel**:
   - Used when the data is linearly separable (can be divided by a straight line).
   - Equation: $K(x, y) = x \cdot y$
   - Suitable for high-dimensional datasets.
2. **Polynomial Kernel**:
   - Useful for non-linear data.
   - Equation: $K(x, y) = (\gamma(x \cdot y) + r)^d$
   - $d$ is the degree of the polynomial, $r$ is a constant, and $\gamma$ controls the influence of individual training examples.

# Common Kernels

3. **Radial Basis Function (RBF) Kernel / Gaussian Kernel**:
   - Popular for non-linear problems.
   - Equation: $K(x, y) = \exp(-\gamma||x - y||^2)$
   - $\gamma$ defines how far the influence of a single training example reaches.
4. **Sigmoid Kernel**:
   - Similar to the neural network's activation function.
   - Equation: $K(x, y) = \tanh(\gamma(x \cdot y) + r)$
   - Useful for certain non-linear problems, but less commonly used compared to RBF or polynomial kernels.

# How to choose Kernel

- If a straight line can effectively separate our classes, a **linear kernel** is a suitable choice.
- If the separation requires a curved boundary, a **polynomial kernel** is likely the best option.
- If the data exhibits circular patterns, a **Radial Basis Function** (**RBF**) **kernel** would be more appropriate.
- If the data is divided by a threshold with values on either side, a **sigmoid kernel** might provide better separation.

# SVM kernals

- https://scikit-learn.org/stable/modules/svm.html
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

# How Non-Linear SVC Works

- **Step 1: Data Mapping**:
  - Non-linear SVC uses the kernel function to map the input data into a higher-dimensional space.
- **Step 2: Finding the Optimal Hyperplane**:
  - In this transformed space, it finds the optimal hyperplane (a linear decision boundary) that maximizes the margin between different classes.
- **Step 3: Decision Boundary**:
  - The decision boundary, when mapped back to the original space, appears as a non-linear boundary that can separate complex data patterns.
- **Notebook**: https://colab.research.google.com/drive/1lJViQQ4zmq1xT2AkekO_pWMaSxkhfOXa#scrollTo=epGHsCzJhFG2

# Practical Considerations

- **Choosing the Kernel**: The choice of kernel depends on the specific problem and the nature of the data. The RBF kernel is a good default choice for many problems.
- **Hyperparameters**: Non-linear SVC requires tuning several hyperparameters, such as C (regularization), gamma (in RBF), and others depending on the kernel. Cross-validation is often used to find the best settings.
- **Computational Complexity**: Non-linear SVC can be computationally expensive, especially with large datasets or high-dimensional spaces. It can also be memory-intensive because it requires storing the entire dataset to compute the kernel matrix.

# SUMMARY

- **Flexibility and Power**: Non-linear SVC is a classification method that can handle complex data patterns.
- **Kernel Functions**: It uses kernel functions to implicitly transform the input space into a higher-dimensional space.
- **Non-linear Decision Boundaries**: This transformation allows SVC to find non-linear decision boundaries, enabling accurate separation of classes that are not linearly separable.
- **Hyperparameter Tuning**: Careful selection of kernels and tuning of hyperparameters are crucial for achieving good performance with non-linear SVC.

# GAMMA HYPERPARAMETER

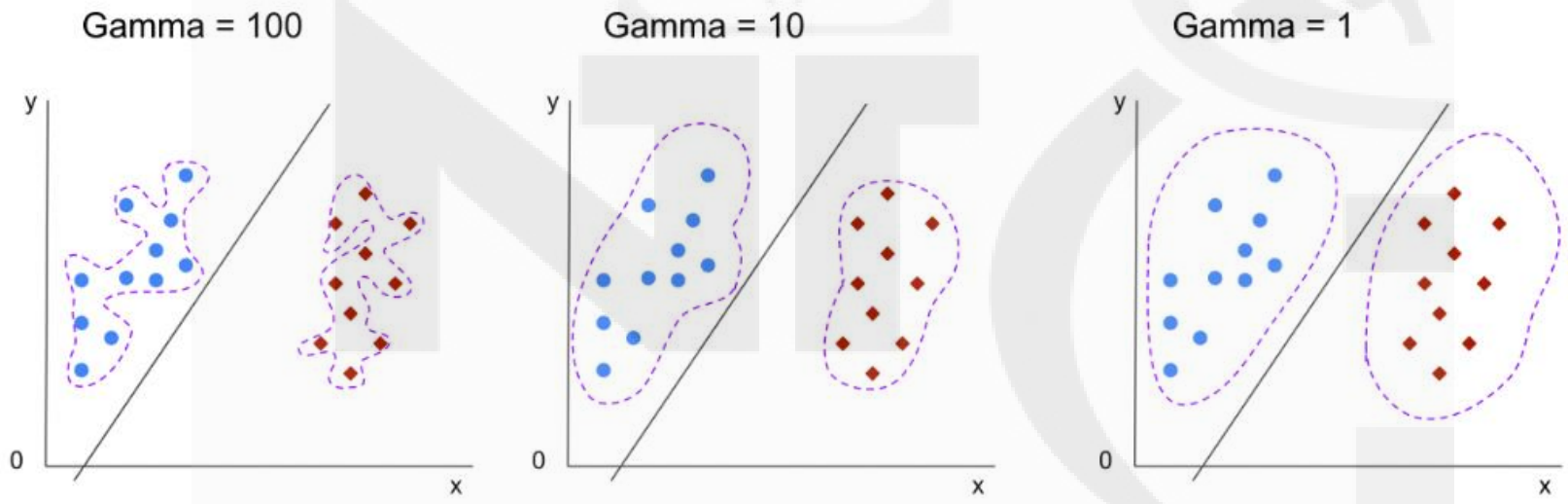
# Gamma

- The gamma hyperparameter in Support Vector Classification (SVC), particularly in the context of the Radial Basis Function (RBF) kernel, plays a crucial role in controlling the decision boundary's shape and complexity.

# Understanding Gamma

- **High Gamma (Large Value):**
  - A high gamma value makes each training example have a very localized influence, meaning the decision boundary is more complex and can closely fit the training data.
  - This can lead to a model that captures fine details, but it also increases the risk of **overfitting**, where the model performs well on the training data but poorly on unseen data.
- **Low Gamma (Small Value):**
  - A low gamma value causes the influence of each training example to be spread out over a larger area. This results in a smoother and simpler decision boundary.
  - This reduces the risk of overfitting but increases the risk of **underfitting**, where the model might not capture important details in the data and thus performs poorly both on the training data and unseen data.

- **High Gamma**: Decision boundaries tightly wrap around data points, creating intricate boundaries that can fit the data points almost exactly.
- **Low Gamma**: Decision boundaries are smoother and more generalized, potentially missing out on some details.


Gamma = 100
y
0
x
Gamma = 10
y
0
x
Gamma = 1
y
0
x

# Choosing Gamma

- **Cross-Validation**: To choose an optimal gamma, cross-validation is typically used, where the performance of different gamma values is evaluated on validation datasets.

- **Trade-off**: The key is finding a balance between too much complexity (high gamma) and too much simplicity (low gamma).

# Summary

- **High Gamma** → More complex, high variance, risk of overfitting.
- **Low Gamma** → Simpler, high bias, risk of underfitting.