Support Vector Machine (SVM)

1. Objective
2. Hyperplane
3. Support Vectors
4. Margin
5. Hard Margin
6. Soft Margin
7. Linear SVM
8. Non-Linear SVM
9. Kernel Trick:
    - Linear Kernel
    - Polynomial Kernel
    - Radial Basis Function (RBF) Kernel
    - Sigmoid Kernel

Support Vector Machine

- Support Vector Machines (SVMs) are powerful supervised learning algorithms
- It is used for classification, regression, and outlier detection tasks.
- It works well in high-dimensional spaces and handle non-linear decision boundaries efficiently using kernel tricks.
- Support vector machines were invented by V. Vapnik and his co-workers in 1970s in Russia and became known to the West in 1992.
- SVMs are linear classifiers that find a hyperplane to separate two classes of data, positive and negative.
- It is perhaps the best classifier for text classification.

Support Vector Machine

1. **Objective**:
    - SVM aims to find the best decision boundary (hyperplane) that separates classes in the feature space. For binary classification, this hyperplane maximizes the margin between the closest data points of the two classes.
2. **Hyperplane**:
    - A hyperplane in n-dimensional space is defined by:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

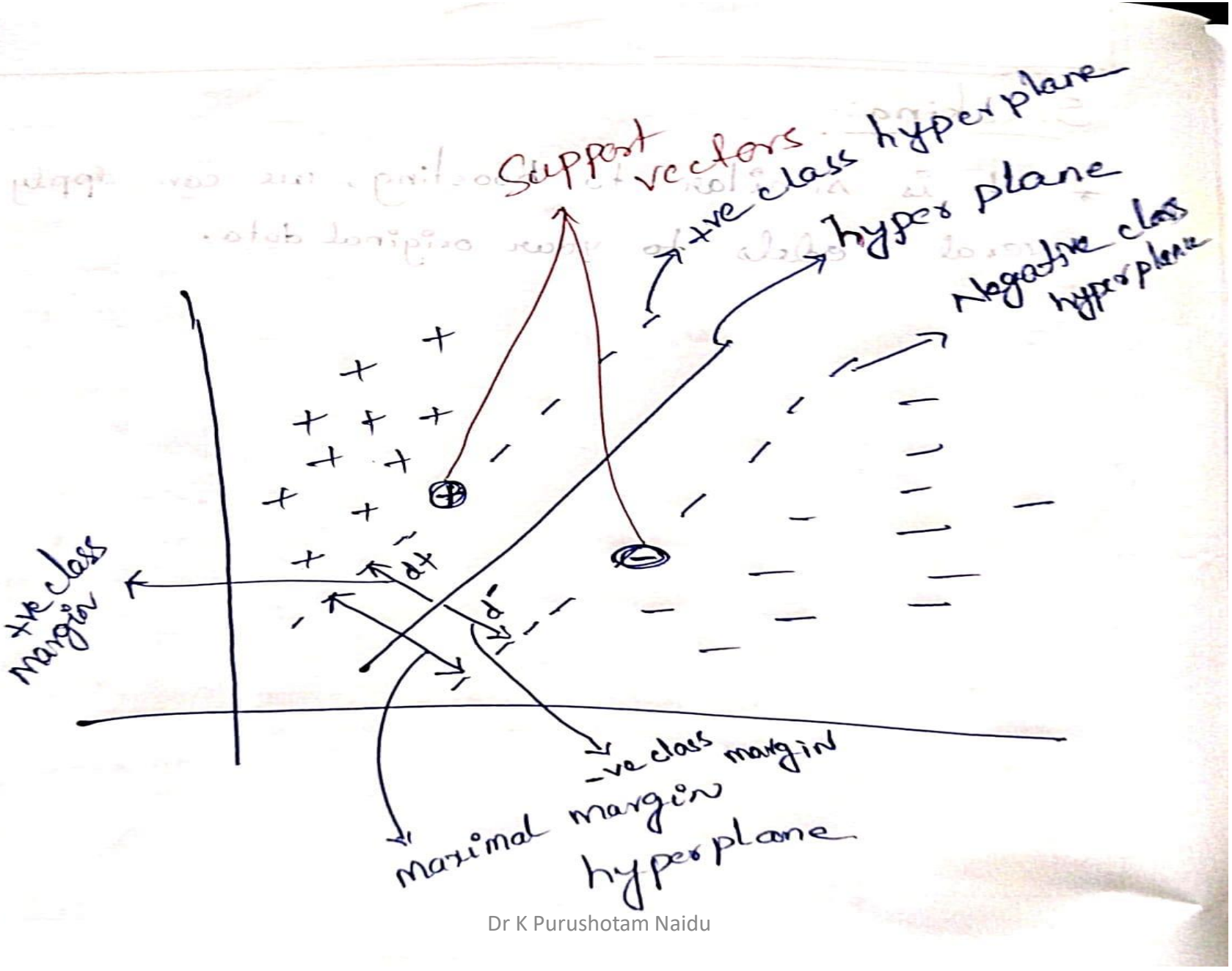
where $\mathbf{w}$ is the normal vector to the hyperplane, $\mathbf{x}$ is a point in the feature space, and b is the bias.

It is a decision plane or line which is divided between a set of instances having different classes

Support Vector Machine

3. **Support Vectors**:
    - These are the data points closest to the hyperplane. The margin is determined by these points. Removing other points doesn't change the decision boundary.
4. **Margin**:
    - The distance between the hyperplane and the nearest data points of any class. SVM maximizes this margin.

Support Vector Machine

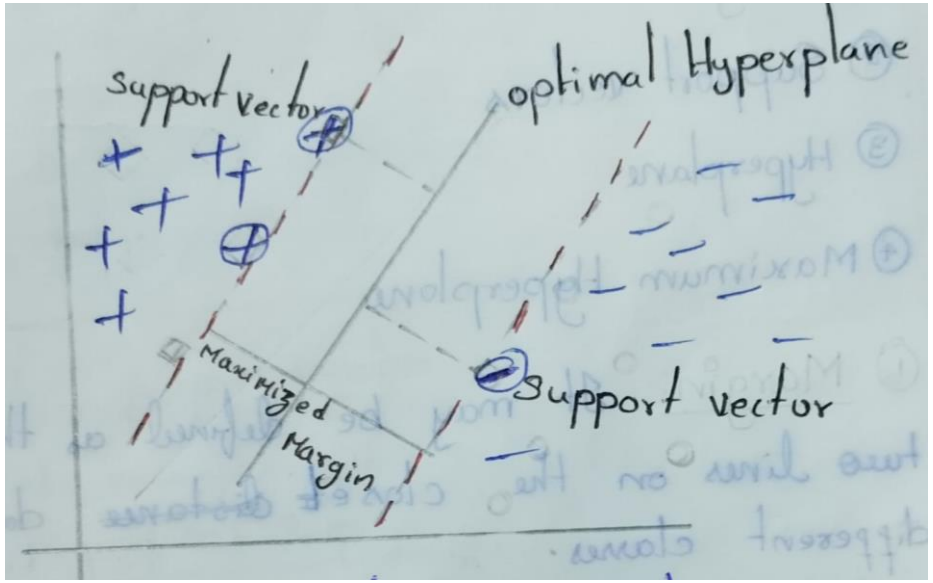


Support Vector Machine

Types of SVM

1.Linear SVM:

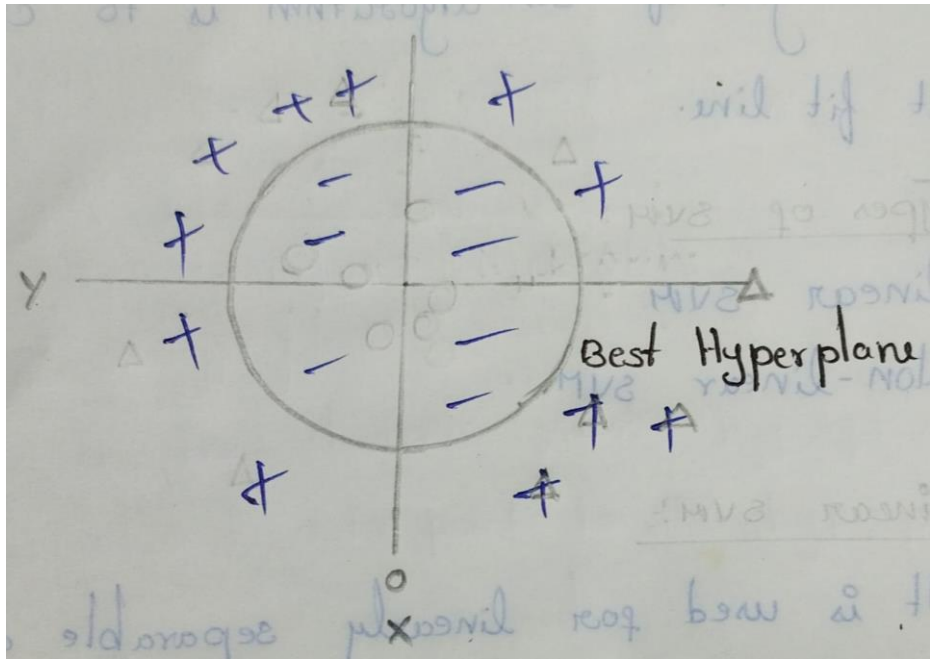
Used when the data is linearly separable, meaning a straight line (or hyperplane) can separate classes without errors.



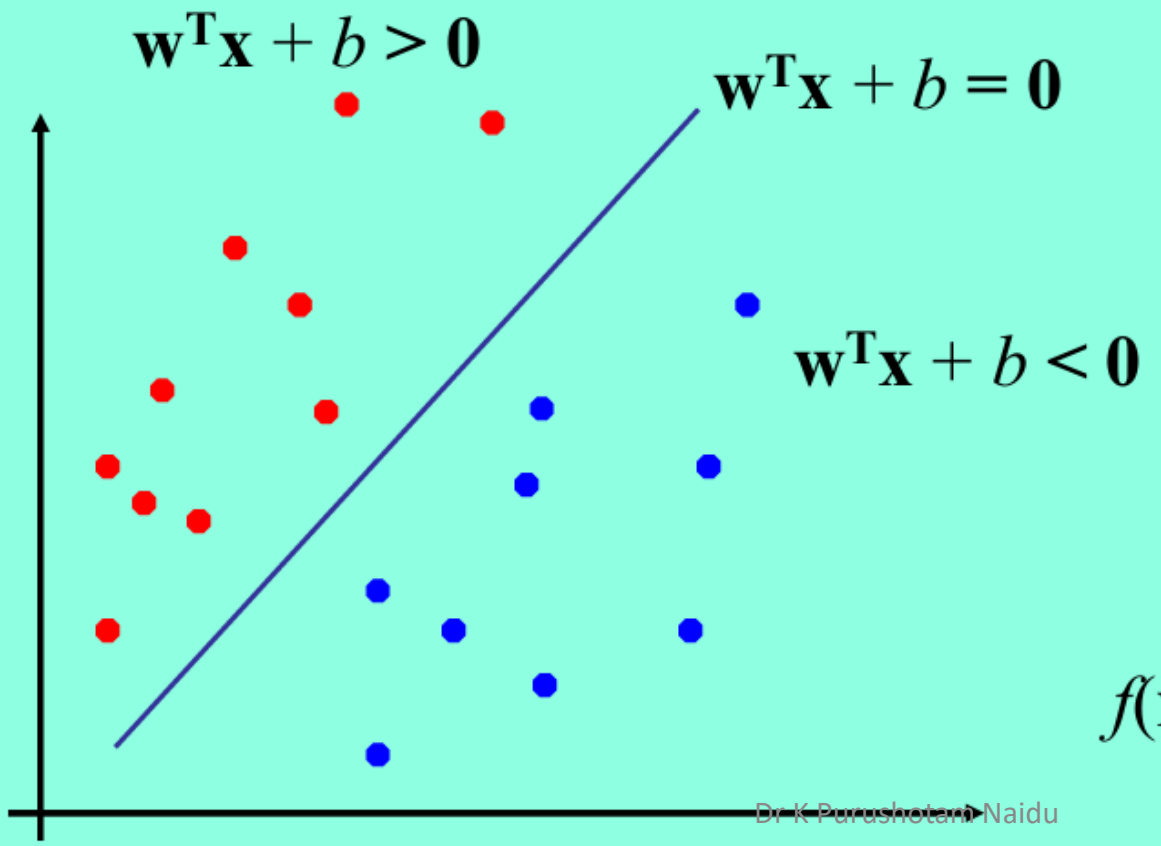
2. Non-linear SVM:

Used when the data is not linearly separable.

SVM employs kernel functions to map data into a higher-dimensional space where a linear hyperplane can separate the classes.



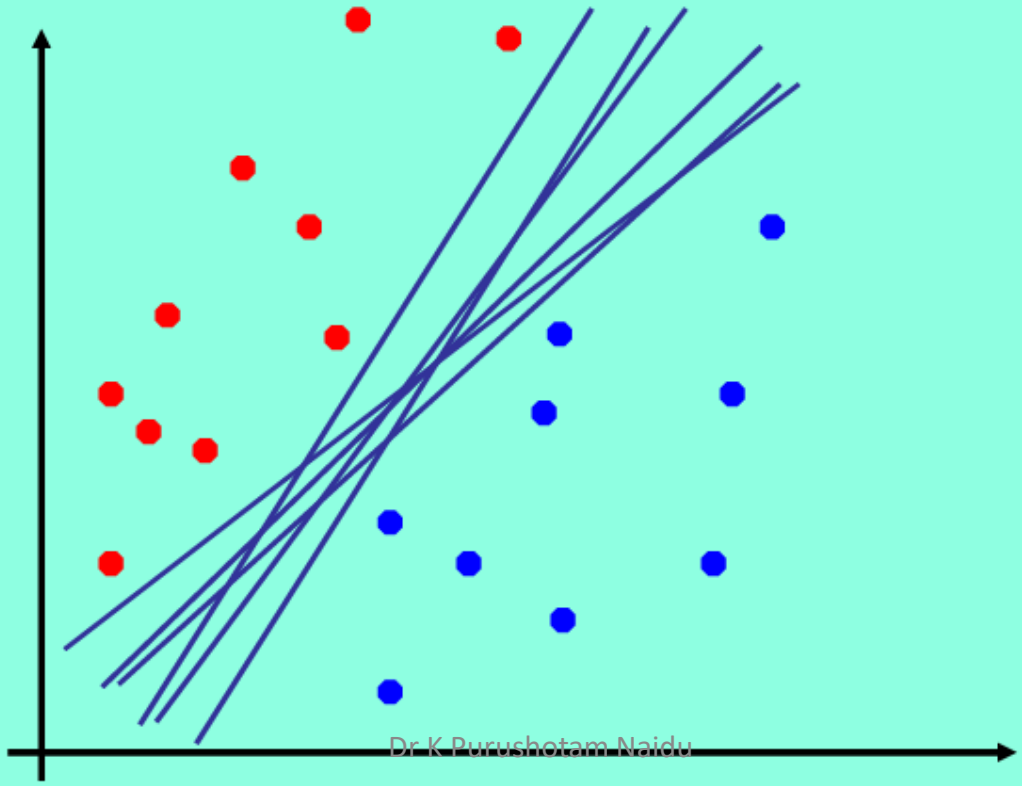
Linear Discriminant Functions



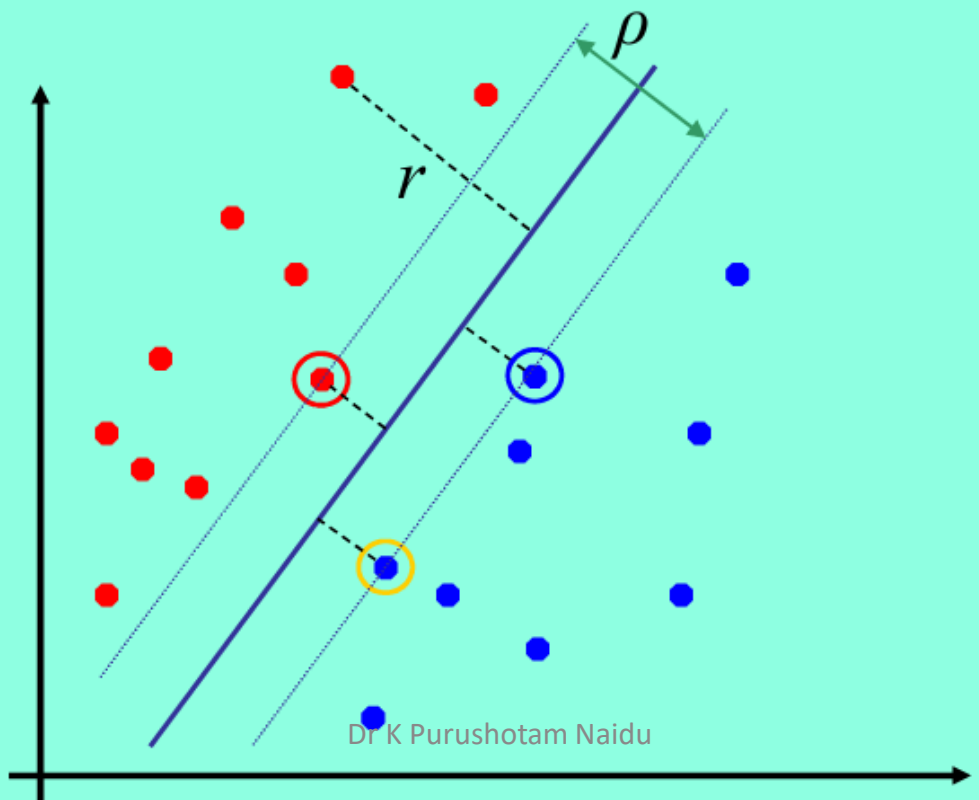
$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

Linear Discriminant Functions

- Which of the linear separators is optimal?

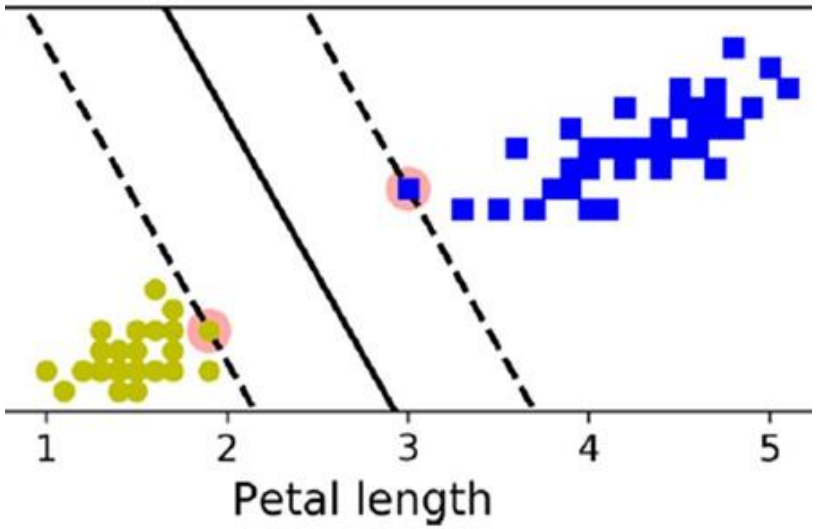


Maximum Margin Classification



Hard Margin

- Assumes the data is perfectly separable.
- The hyperplane is chosen such that all data points are correctly classified with no tolerance for misclassification.
- Maximizes the margin (distance between the hyperplane and nearest data points).



Limitations:

- Cannot handle noisy data or overlapping classes.
- Only works when data is strictly linearly separable.

it is sensitive to outliers.

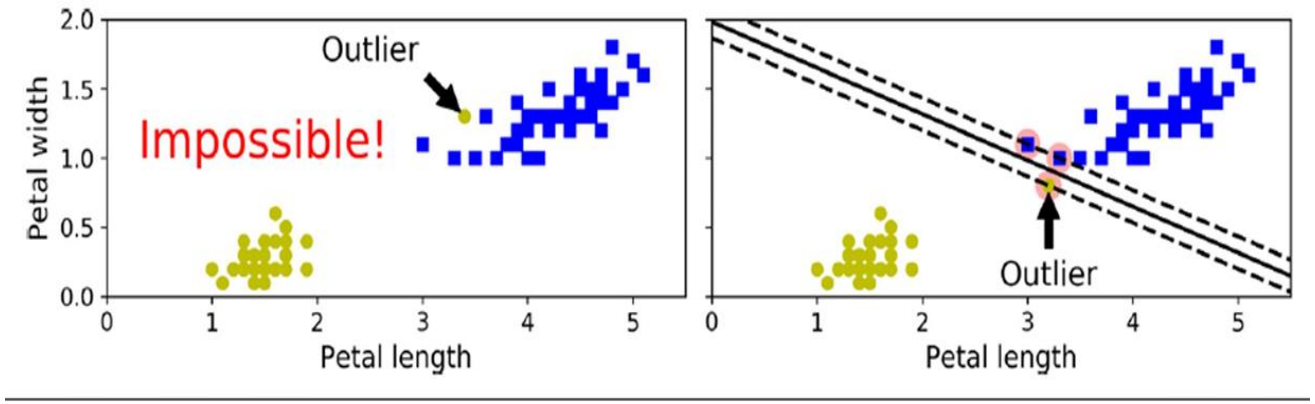


Figure 5-3. Hard margin sensitivity to outliers

Soft Margin

- Allows some misclassification or margin violations to handle noisy or overlapping data.
- Introduces a regularization parameter C to balance the trade-off between margin size and misclassification.
- Minimizes a combined objective:
    1. Maximizing the margin.
    2. Reducing the misclassification penalty.

Advantages:

- Works well with non-linearly separable and noisy data.
- More flexible in real-world applications.

Soft Margin Classification

- To avoid these issues, use a more flexible model.
- The objective is to find a good balance between
  - keeping the street as large as possible and
  - limiting the margin violations (i.e., instances that end up on the street or on the wrong side).
  - Regulated by using ‘C’ parameter.
    - Higher C = Narrower st, fewer margin violations
    - Smaller C = Wider st, more margin violations
- This is called **soft margin classification**.

Soft Margin Classification

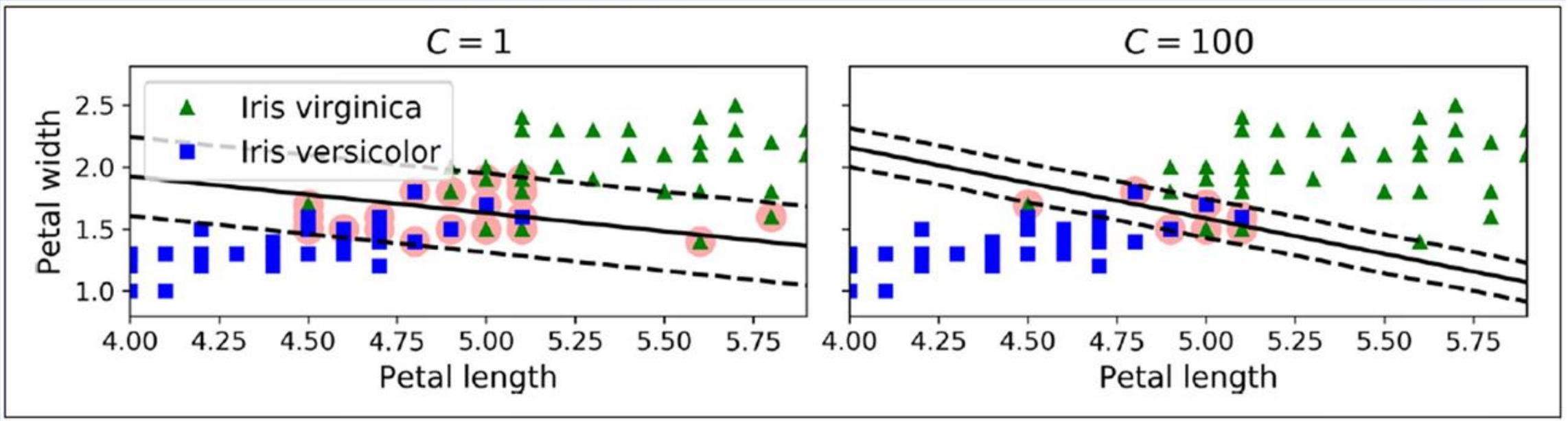


Figure 5-4. Large margin (left) versus fewer margin violations (right)

Comparison

Feature	Hard Margin	Soft Margin
Data Assumption	Perfectly separable data	Overlapping/noisy data allowed
Tolerance	No misclassification allowed	Allows some misclassification
Regularization	No regularization (strict)	Uses C for regularization
Practicality	Limited use in real-world cases	Widely used in practice

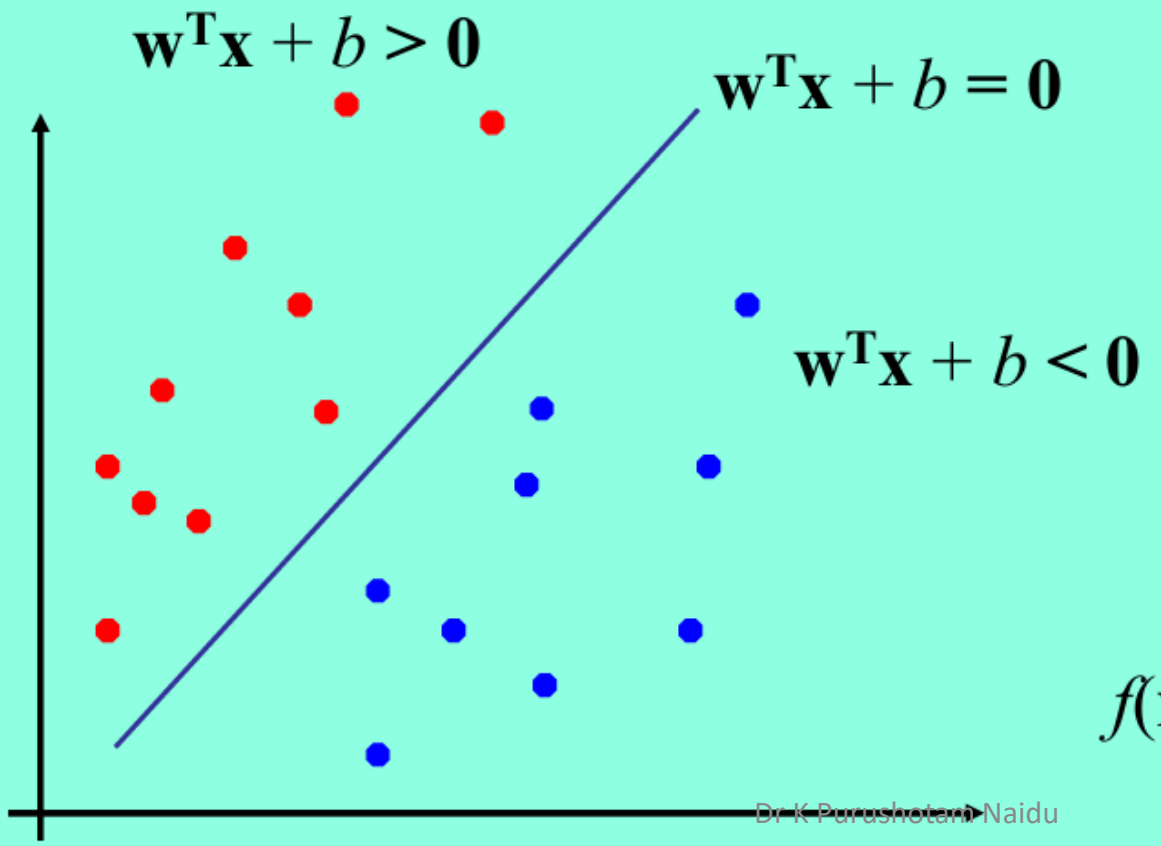
Linear SVM Classification

```
import numpy as np
from sklearn import datasets
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
iris = datasets.load_iris()
X = iris["data"][:, (2, 3)] # petal length, petal width
y = (iris["target"] == 2).astype(np.float64) # Iris virginica
svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge")),
    ])
svm_clf.fit(X, y)
```

Linear SVM Classification

- Instead of using the LinearSVC class, we could use the SVC class with a linear kernel, with SVC(kernel="linear", C=1).
- Or we could use the SGDClassifier class, with SGDClassifier(loss="hinge", alpha=1/(m*C)).
- It does not converge as fast as the LinearSVC class, but it can be useful to handle online classification tasks or huge datasets that do not fit in memory (out-of-core training).

Linear Discriminant Functions



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

Nonlinear Support Vector Machine: Kernals

- Support Vector Machines (SVMs) use **kernel functions** to handle non-linear data by implicitly mapping input data into a higher-dimensional space.
- This allows SVMs to create decision boundaries (hyperplanes) that would not be possible in the original feature space.

Mathematically, a kernel is defined as:

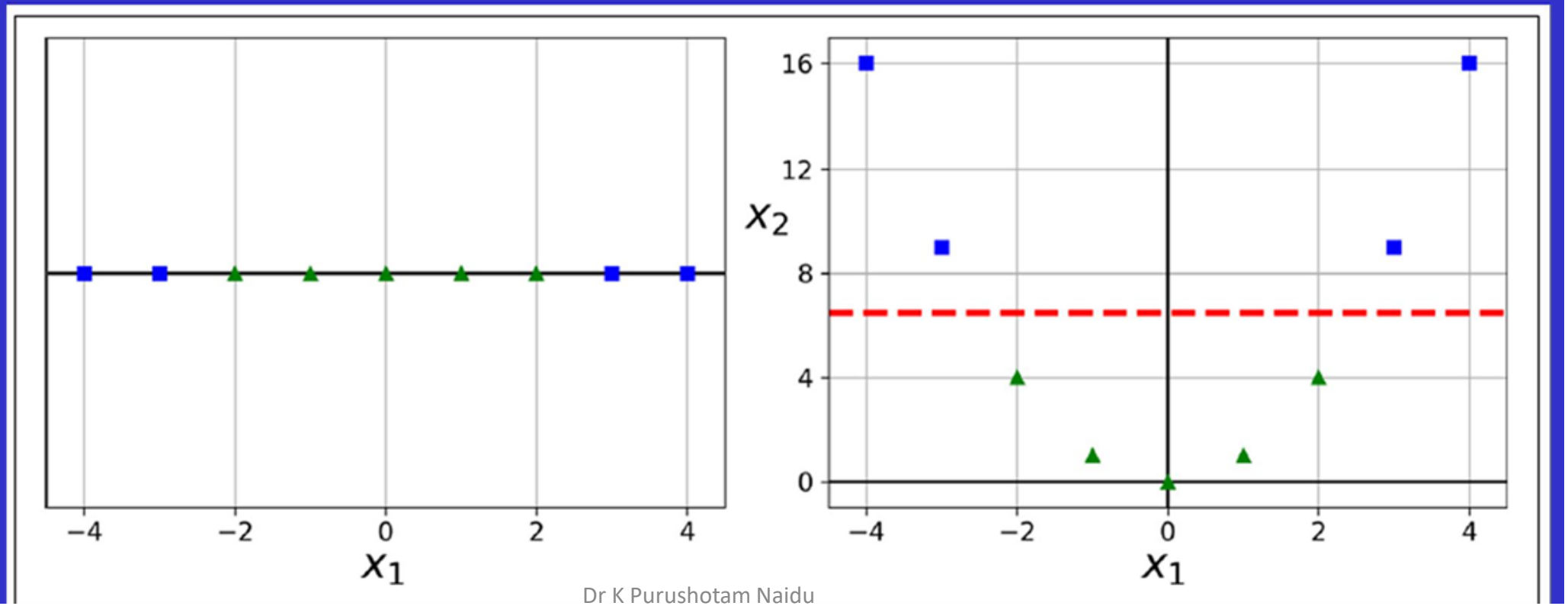
$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$$

where:

- $\phi(\cdot)$: Mapping function to higher dimensions.
- $\cdot$: Dot product in the transformed space.

Nonlinear SVM Classification

- Adding a second feature $x_2 = (x_1)^2$ results in a 2D dataset that is perfectly linearly separable.



Nonlinear SVM Classification

- The following code generates a toy moons dataset for binary classification in which the data points are shape as two interleaving half circles, applies PolynomialFeature and StandardScaler transformers before applying LinearS to fit a SVM model.

```
from sklearn.datasets import make_moons
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures

X, y = make_moons(n_samples=100, noise=0.15)
polynomial_svm_clf = Pipeline([
        ("poly_features", PolynomialFeatures(degree=3)),
        ("scaler", StandardScaler()),
        ("svm_clf", LinearSVC(C=10, loss="hinge"))
    ])

polynomial_svm_clf.fit(X, y)
```

Nonlinear SVM Classification

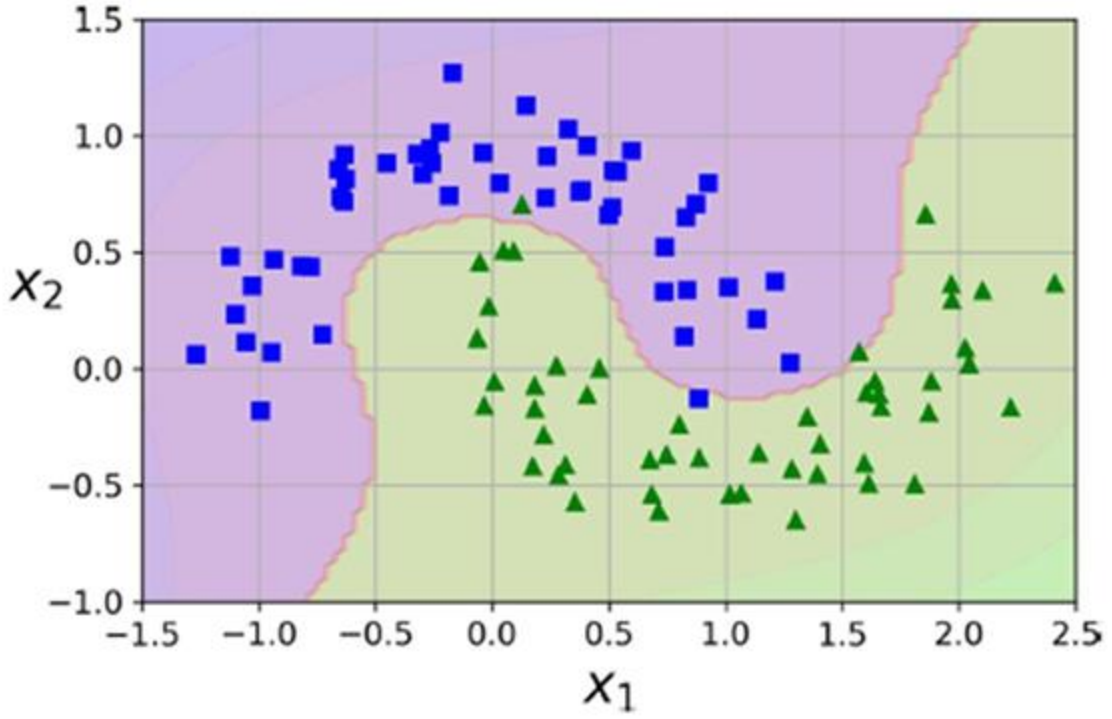
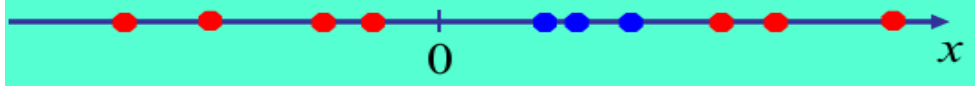
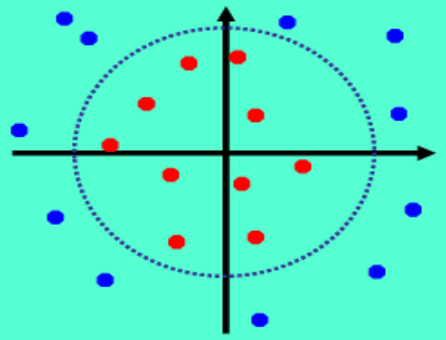


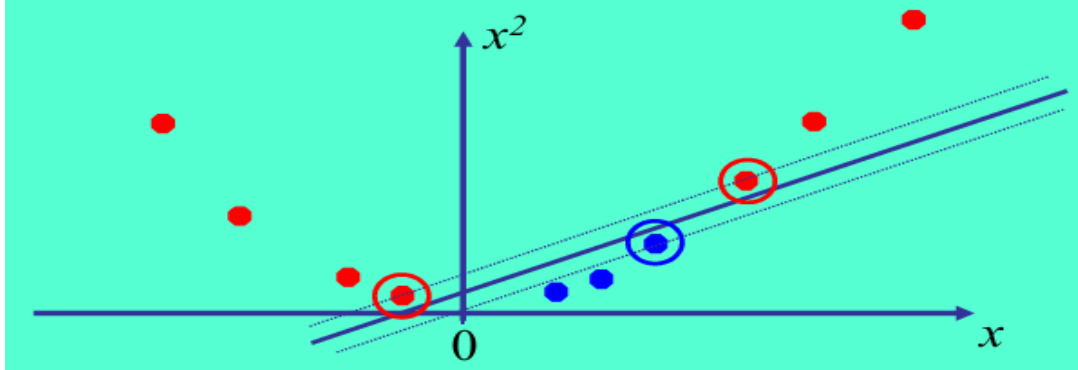
Figure 5-6. Linear SVM classifier using polynomial features

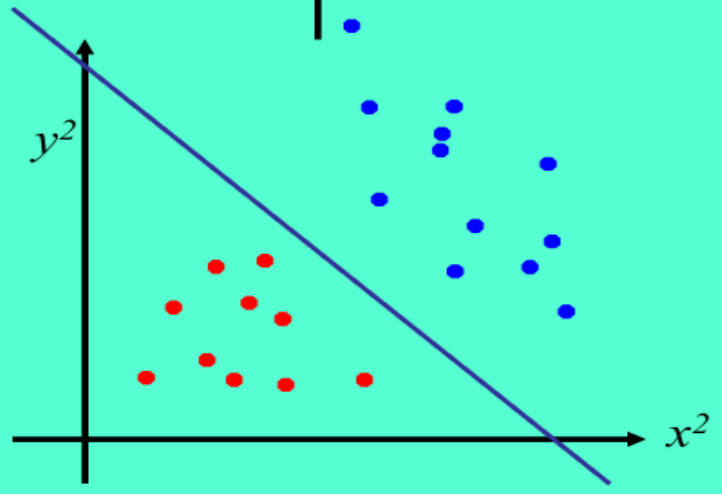
Kernel-Induced Feature Spaces

- What if the data is linearly inseparable?
- Can we apply some mathematical trick?



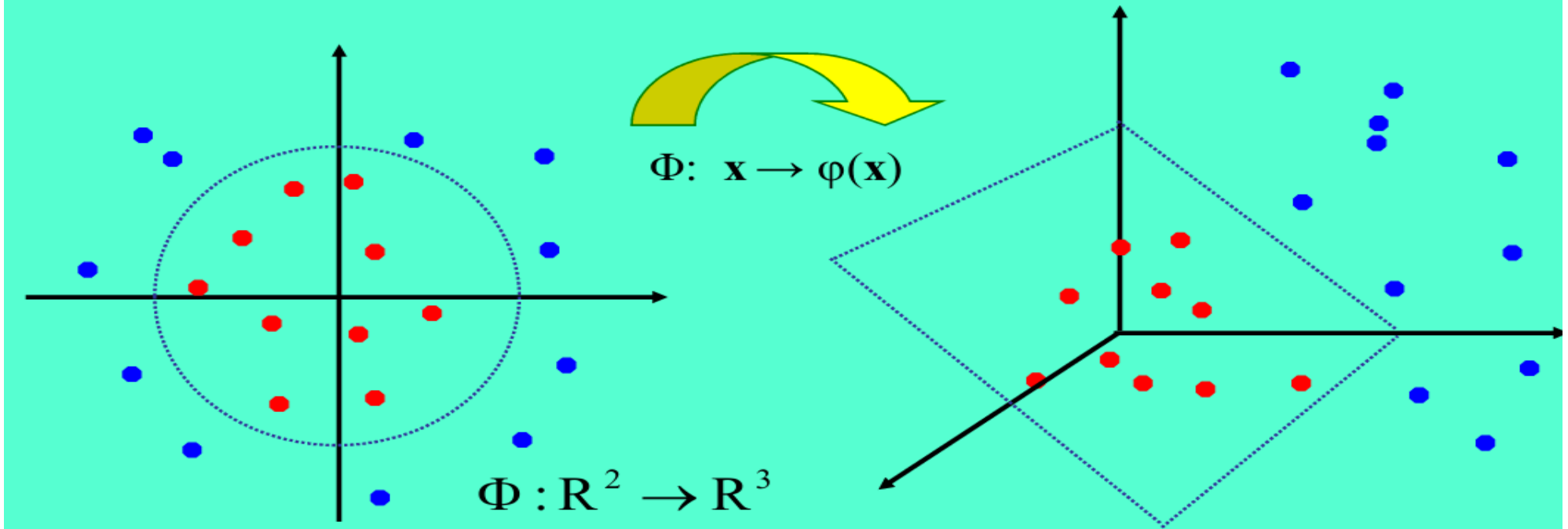






Kernel-Induced Feature Spaces

- General idea: the original input space can alwa be mapped to some higher-dimensional feature space where the training set is separable:



$$\Phi : \mathrm{R}^2 \rightarrow \mathrm{R}^3$$

$$\Phi : [x_1, x_2] \rightarrow [x_1^2, \sqrt{2}x_1x_2, x_2^2)]$$

Nonlinear Support Vector Machine: Kernals

1. Linear Kernel
2. Polynomial Kernel
3. Radial Basis Function (RBF) Kernel
4. Sigmoid Kernel
5. Custom Kernels

Nonlinear Support Vector Machine: Kernals

1. **Linear Kernel**:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$

- **Use Case**: When data is linearly separable.
- **Advantages**: Fast and simple; avoids the complexity of mapping.
- **Example**: Hyperplane in the original feature space.

Nonlinear Support Vector Machine: Kernals

2. **Polynomial Kernel**:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + c)^d$$

- **Parameters**:
  - c: Coefficient (controls flexibility).
  - d: Degree of the polynomial (controls complexity).
- **Use Case**: When data has polynomial relationships.
- **Advantages**: Allows flexible decision boundaries.
- **Disadvantages**: Computationally expensive for high-degree polynomials.

Polynomial Kernel

- Adding polynomial features is simple to implement and can work great with all sorts of Machine Learning algorithms (not just SVMs).

That said,

- at a low polynomial degree, this method cannot deal with very complex datasets, and
- with a high polynomial degree it creates a huge number of features, making the model too slow.

Polynomial Kernel

- Fortunately, when using SVMs you can apply an almost miraculous mathematical technique called the **kernel trick**.
- The kernel trick makes it possible
  - to get the same result as if you had added many polynomial features,
  - without actually having to add the features which makes the model slow.
- This trick is implemented by the SVC class.
  - No need to use PolynomialFeatures as in LinearSVC.

Polynomial Kernel

- The kernel trick is implemented by the SVC class.
  - No need to use PolynomialFeatures as in LinearSVC.

```
from sklearn.svm import SVC
poly_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
    ])
poly_kernel_svm_clf.fit(X, y)
```

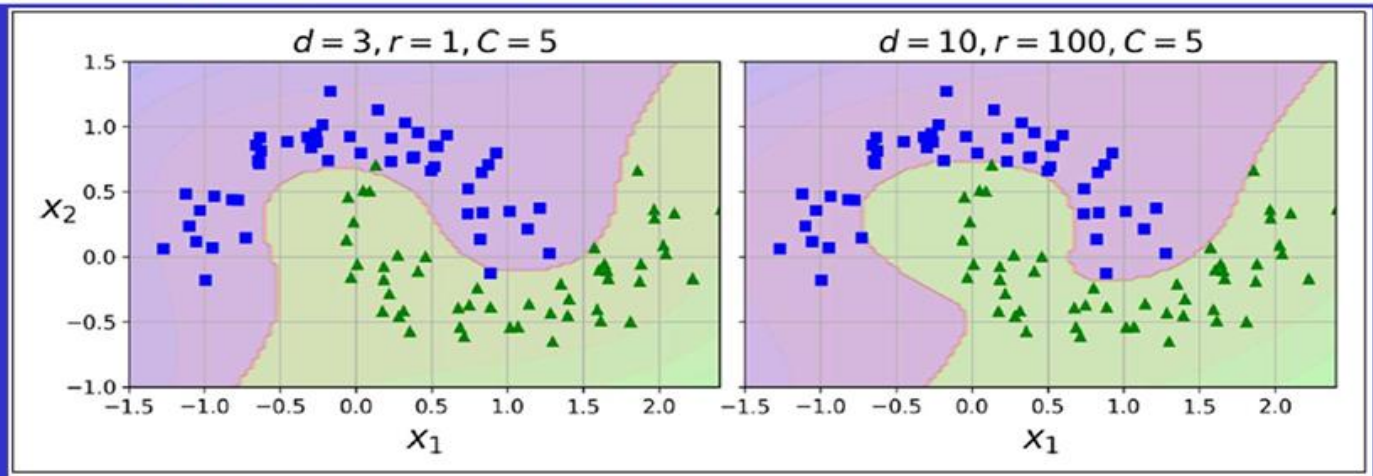


Figure 5-7. SVM classifiers with a polynomial kernel

Nonlinear Support Vector Machine: Kernals

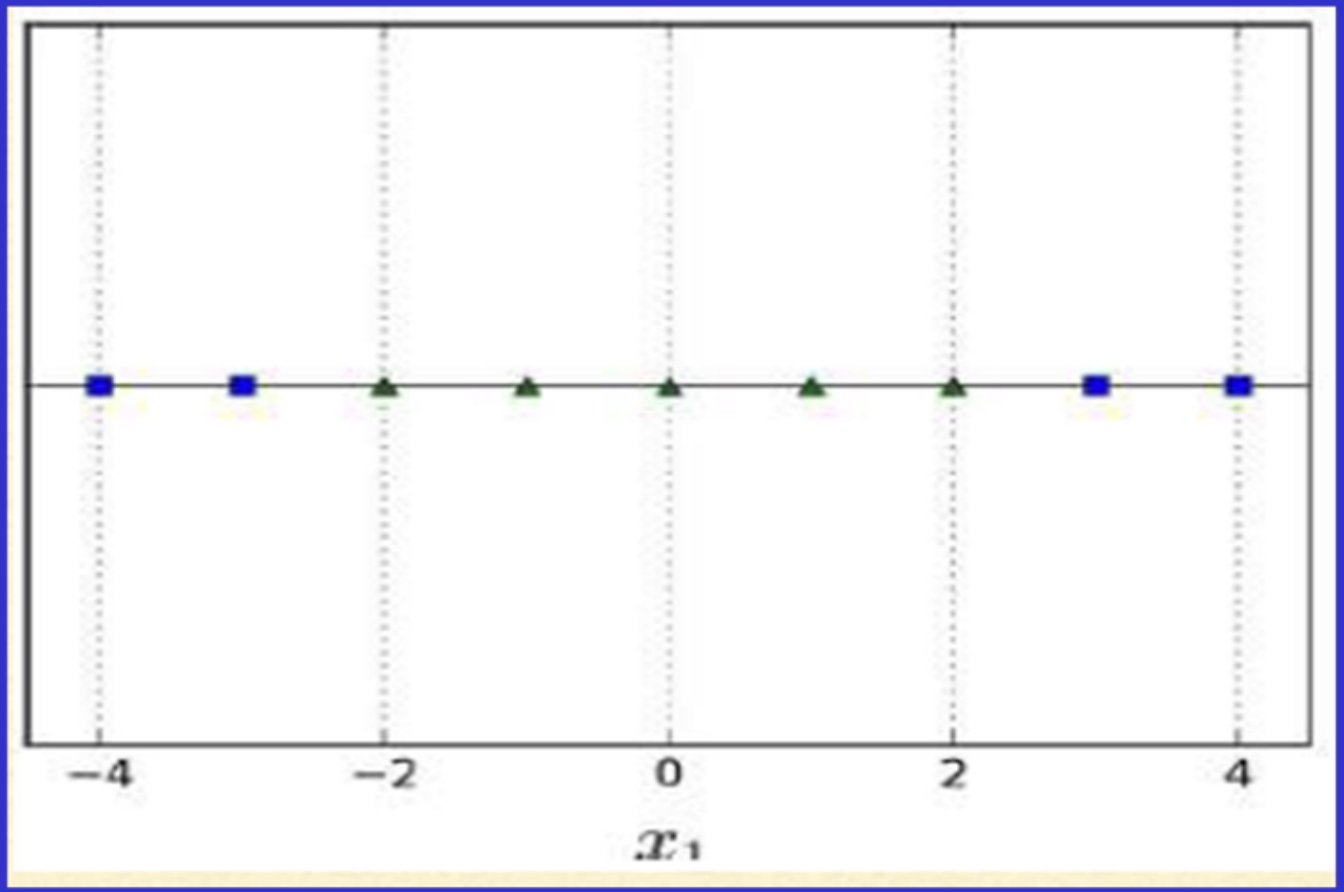
3. **Radial Basis Function (RBF) Kernel (Gaussian Kernel):**

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$$

- **Parameters**:
  - γ: Controls the influence of a single data point.
    - Small γ: Data points have broad influence.
    - Large γ: Data points have narrow influence.
- **Use Case**: Non-linear datasets.
- **Advantages**: Handles complex relationships; most commonly used.
- **Disadvantages**: May overfit if γ is too large.

Nonlinear SVM - SVC RBF

- Is this linearly separable?



Nonlinear SVM - SVC RBF

- Introduce two landmarks, and measure similarity of each point to these two landmarks.
- Nice bell curves (values ranging from 0 to 1).

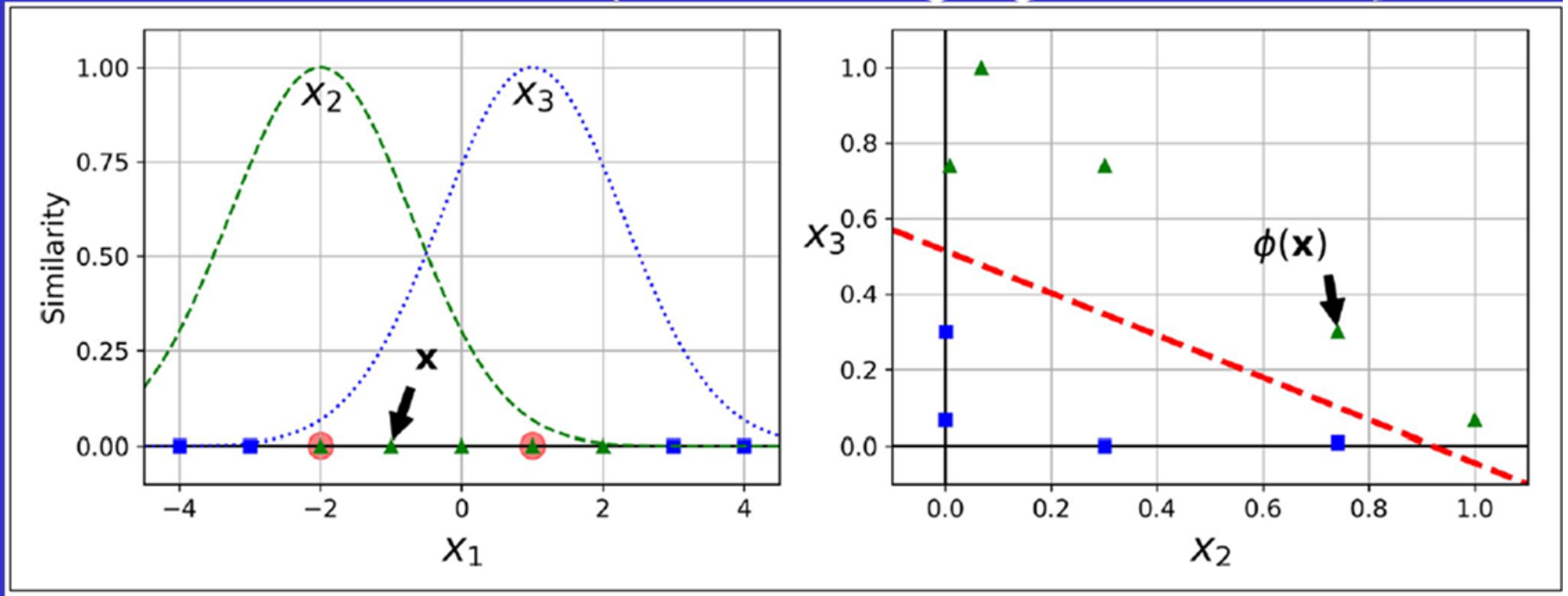


Figure 5-8. Similarity features using the Gaussian RBF

Nonlinear SVM - SVC RBF

How to select the landmarks?

- The simplest approach is to create a landmark at the location of each and every instance in the dataset.
- Doing that creates many dimensions and thus increases the chances of linear separability.
- The downside is that, if training set is huge, number of new features added will be huge.

Nonlinear SVM - SVC RBF

Gaussian RBF Kernel

- Polynomial Feature addition becomes slow with higher degrees
  - Kernel trick solves it
- Similarity function becomes slow with higher number of training dataset
  - SVM kernel trick again solves the problem
- It lets us to get similar results as if
  - We had added many similarity features
  - Without actually having to add them.

Nonlinear SVM - SVC RBF

Gaussian RBF Kernel in ScikitLean

```
rbf_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="rbf", gamma=5, C=0.001))
    ])
rbf_kernel_svm_clf.fit(X, y)
```

- Here, gamma acts like a regularization hyperparameter:
  - if your model is overfitting, you should reduce it;
  - if it is underfitting, you should increase it.

Nonlinear SVM - SVC RBF

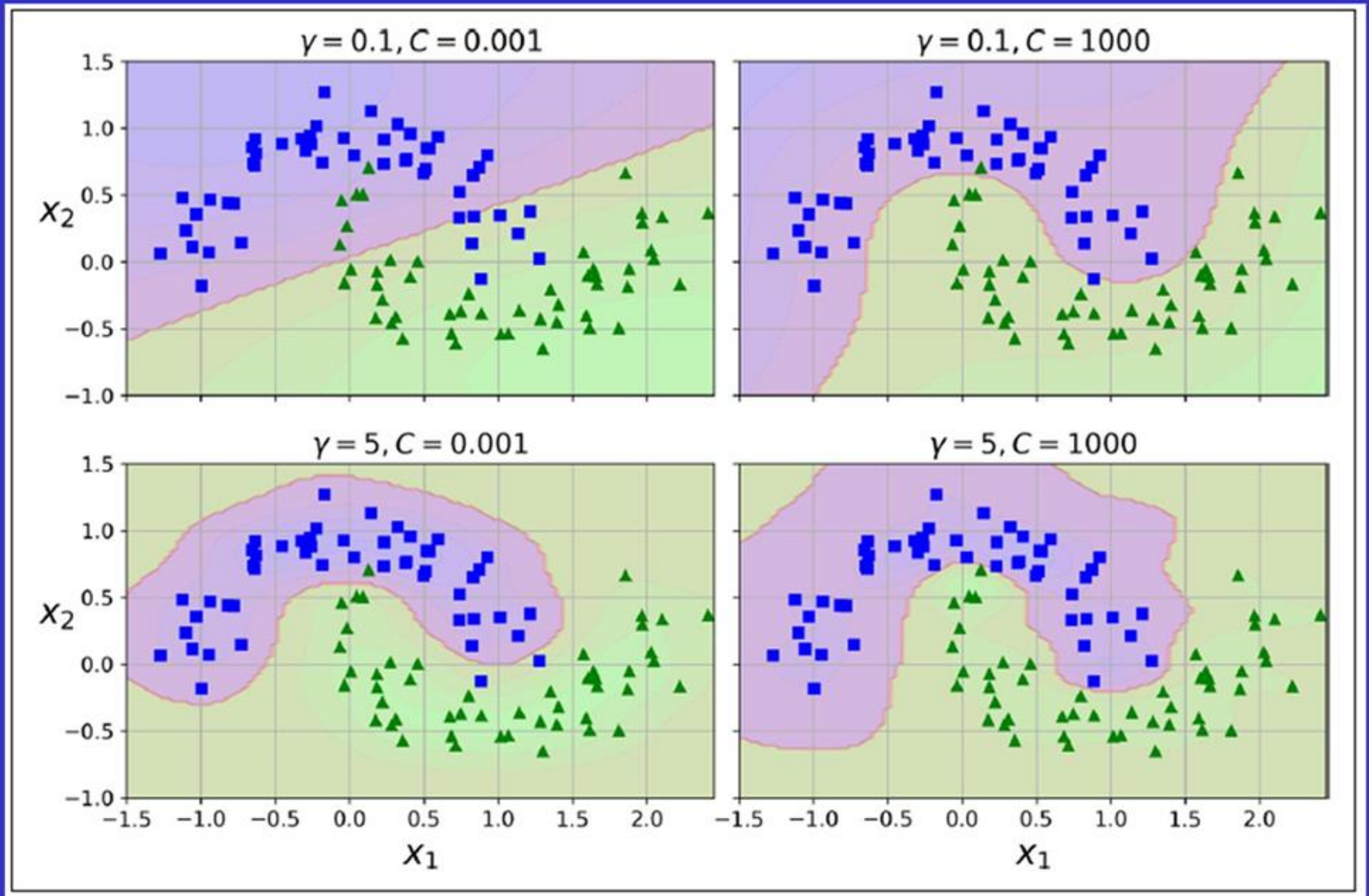


Figure 5-9. SVM classifiers using an RBF kernel

Computational Complexity

LinearSVC class

- Implements an optimized algorithm for linear SVMs.
- It does not support the kernel trick, but it scales almost linearly with the number of training instances and the number of features.
- Its training time complexity is roughly $O(m \times n)$.

SVC class

- Support kernel tricks.
- Time complexity is between: $O(m^2 \times n)$ and $O(m^3 \times n)$.
- Dreadfully slow on large training sets.
- Perfect for complex but small or medium training sets.

m = number of training instances, n = number of features

Nonlinear Support Vector Machine: Kernals

4. **Sigmoid Kernel**:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i^T \mathbf{x}_j + c)$$

- **Parameters**:
  - α: Scaling factor.
  - c: Offset.
- **Use Case**: Mimics neural networks (tanh activation).
- **Advantages**: Flexible and works well in some scenarios.
- **Disadvantages**: Can be sensitive to parameter tuning.

Which kernel to use when?

- Try the linear kernel first (LinearSVC), especially if the training set is very large or too many features.
- If the training set is not too large, try the Gaussian RBF kernel; works well in most cases.
- Have time/resources, try other kernels with CV.

Support Vector Machine: Kernals

Comparison of Kernels

Kernel	Use Case	Strengths	Weaknesses
Linear	Linearly separable data	Fast and interpretable	Poor with non-linear data
Polynomial	Polynomial relationships	Flexible	Computationally expensive
RBF	General non-linear relationships	Handles complex data well	Risk of overfitting
Sigmoid	Neural-network-like behavior	Simple and intuitive	Sensitive to parameters

Difference Between Linear and Non-Linear SVM

Feature	Linear SVM	Non-Linear SVM
Hyperplane	A straight line (in 2D) or flat hyperplane (in higher dimensions).	A curved or complex decision boundary.
Data Separation	Used for linearly separable data.	Used for data that is not linearly separable.
Kernel Function	Uses the linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.	Uses non-linear kernels like RBF, polynomial, or sigmoid.
Computational Cost	Low, as no transformation is required.	Higher, due to kernel computations and transformations.
Interpretability	Easy to interpret and visualize.	Complex and harder to interpret.
Examples	Text classification, linearly separable problems.	Image classification, datasets with complex patterns.
Flexibility	Limited to linear decision boundaries.	Can model highly complex decision boundaries.

SVMs and Multiclass Classification

- SVMs are strictly binary classifiers.
- However, there are various strategies that you can use to perform multiclass classification with multiple binary classifiers.
  - *one-versus-the-rest* (OvR)
  - *one-versus-one* (OvO)

- Multiclass classification refers to the task of predicting one class label from three or more classes for a given input.
- Unlike binary classification, which distinguishes between two categories, multiclass classification extends this to multiple categories.
- Below are some common techniques used in multiclass classification with examples:

1. One-vs-All (OvA) / One-vs-Rest (OvR)

This approach involves training a separate binary classifier for each class. For a dataset with N classes, N classifiers are trained, where each classifier distinguishes one class from the rest.

Example:

- **Dataset**: Animal classification (Cat, Dog, Rabbit)
- **Training**:
  - Classifier 1: Cat vs. (Dog and Rabbit)
  - Classifier 2: Dog vs. (Cat and Rabbit)
  - Classifier 3: Rabbit vs. (Cat and Dog)
- **Prediction**: Each classifier outputs a confidence score, and the class with the highest score is selected.

- Not the best solution for SVMs
  - No predict_proba()
  - Do not scale well on large datasets

2. One-vs-One (OvO)

This method trains $N \times (N-1)/2$ binary classifiers, each distinguishing between two classes. For prediction, a voting mechanism is typically used.

Example:

- **Dataset**: Fruit classification (Apple, Banana, Cherry)
- **Training**:
  - Classifier 1: Apple vs. Banana
  - Classifier 2: Apple vs. Cherry
  - Classifier 3: Banana vs. Cherry
- **Prediction**: Each classifier votes, and the class with the most votes is chosen.

- The **main advantage** of OvO is that each classifier only needs to be trained on the part of the training set for the two classes that it must distinguish.

SVMs and Multiclass Classification

- SVMs scale poorly with the size of the training set.
- For them OvO is preferred because it is faster to train many classifiers on small training sets than to train few classifiers on large training sets.
  - Scikit-Learn detects when you try to use a binary classification algorithm for a multiclass classification task, and it automatically runs OvR or OvO, depending on the algorithm.

Advantages of SVM

1. **Effective in High-Dimensional Spaces**:
    - SVM performs well with datasets having a large number of features, especially when the number of samples is smaller than the number of features.
2. **Versatile Kernel Trick**:
    - The ability to use custom kernels makes SVM effective for both linear and non-linear problems.
3. **Robust to Overfitting**:
    - By maximizing the margin, SVM reduces the chance of overfitting, especially in high-dimensional spaces.
4. **Works Well with Clear Margins**:
    - SVM excels when there is a clear margin of separation between classes.
5. **Flexibility**:
    - SVM can handle classification, regression (SVR), and outlier detection.

Disadvantages of SVM

1. **Computationally Expensive**:
   - Training an SVM can be slow for large datasets, as it involves solving a quadratic optimization problem.
2. **Sensitive to Parameter Tuning**:
   - Performance depends heavily on the choice of kernel, C (regularization parameter), and γ (for RBF kernels).
3. **Ineffective with Noisy Data**:
   - If classes are overlapping or data contains significant noise, SVM may perform poorly.
4. **Lack of Probabilistic Interpretation**:
   - Unlike methods like logistic regression, SVM does not natively provide probabilistic class probabilities (though extensions like Platt scaling exist).
5. **Memory Intensive**:
   - Requires storing all support vectors, which can grow with the complexity of the problem.

Applications of SVM

1. **Image Recognition and Classification**:
    - Handwritten digit recognition (e.g., MNIST dataset).
    - Facial recognition and object detection.
2. **Text Classification and Sentiment Analysis**:
    - Spam email detection.
    - Sentiment classification for reviews or social media data.
3. **Bioinformatics**:
    - Classification of genes or proteins (e.g., cancer detection from microarray data).
4. **Finance**:
    - Stock market trend prediction.
    - Fraud detection in transactions.

5. **Healthcare**:
    - Disease diagnosis (e.g., classifying tumors as benign or malignant).
    - Patient health monitoring systems.
6. **Engineering**:
    - Fault detection in mechanical systems.
    - Predictive maintenance models.
7. **Natural Language Processing (NLP)**:
    - Document categorization.
    - Named entity recognition.
8. **Energy**:
    - Forecasting energy demand or optimizing energy distribution.

Support Vector Regression (SVR) is a regression method derived from Support Vector Machines (SVM), which is primarily used for classification. Unlike traditional regression techniques that aim to minimize the error, SVR tries to fit the data within a specified margin of tolerance, ϵ, around the predicted values.

Key Concepts of SVR

1. **Margin of Tolerance** (ϵ):
    - Defines a margin within which predictions are considered acceptable.
    - No penalty is given for data points within this margin.
2. **Support Vectors**:
    - Data points lying on or outside the margin boundaries (ϵ) influence the model.
    - These points define the regression function.

3. **Kernel Trick**:
    - SVR uses kernels (linear, polynomial, RBF, etc.) to map input data into higher-dimensional space to capture complex relationships.
4. **Objective**:
    - Minimize the error while maintaining the complexity of the model (using a regularization parameter C).

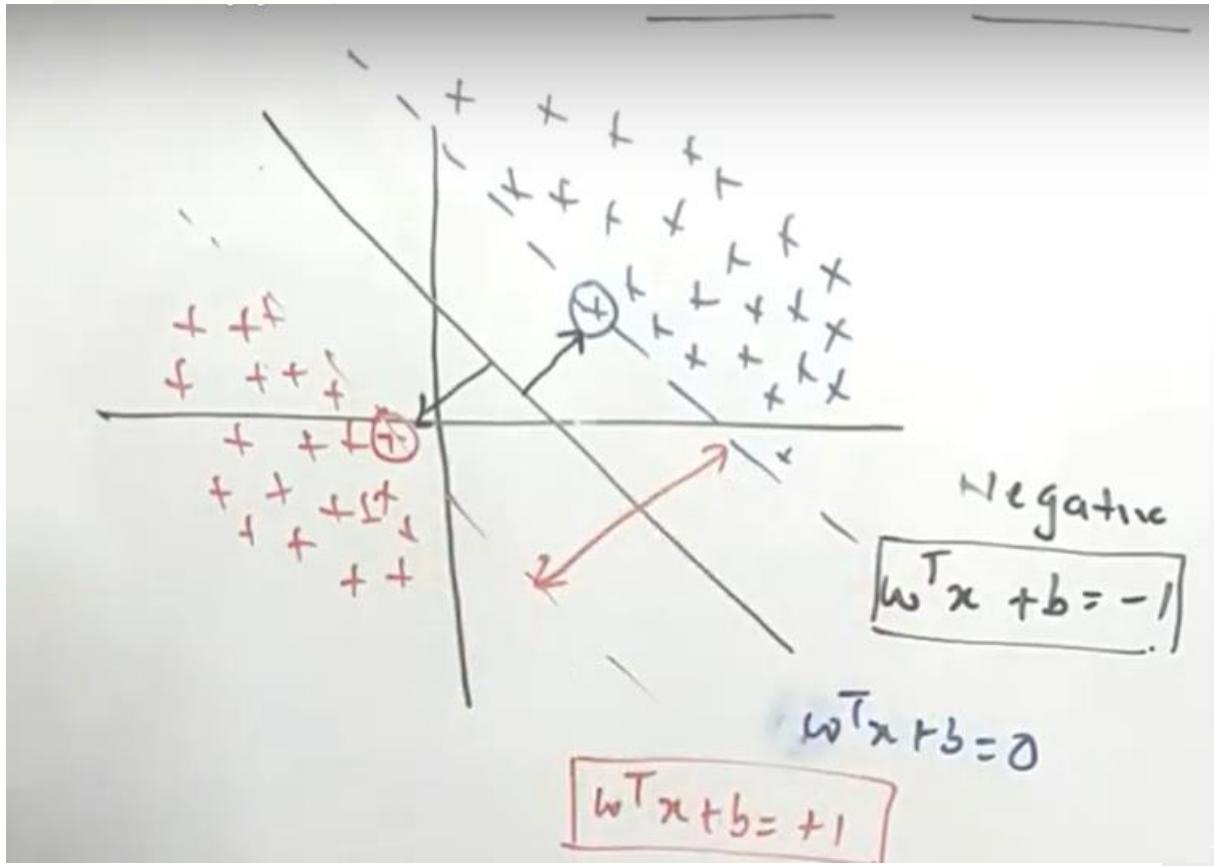
Advantages of SVR

- Works well for small datasets.
- Can capture non-linear relationships with kernels.
- Robust to overfitting (controlled by C).

Disadvantages of SVR

- Computationally expensive for large datasets.
- Requires careful tuning of ϵ, C, and kernel parameters.
- Sensitive to outliers.

SVM MATHS INTUITION



$y = w^T x + b$ — (1)

$m = -1$

c or $b = 0$

$y = w^T x + 0$

$= \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} -4 & 0 \end{bmatrix}$

$= 4$ // $\Rightarrow$ +ve value

$\Rightarrow$ going to be always +ve

+ve

x_1, x_2 (-4, 0)

(4, 4)

x_2

x_1

b = 0

slope = −1

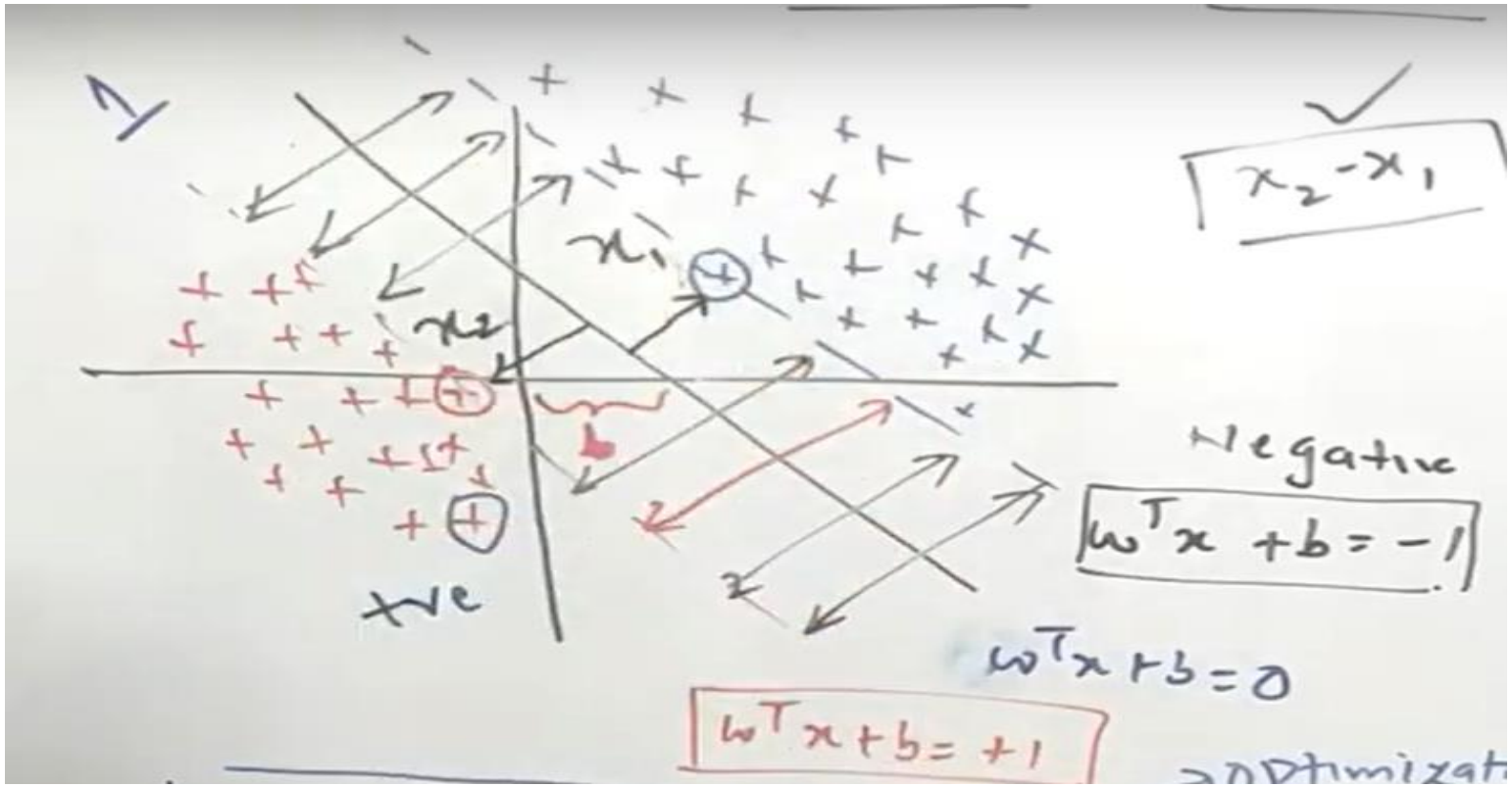
$w^T x + b = 0$

+1

$y = w^T x$

$= \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} 4 & 4 \end{bmatrix}$

$= -4$ // $\Rightarrow$ −ve

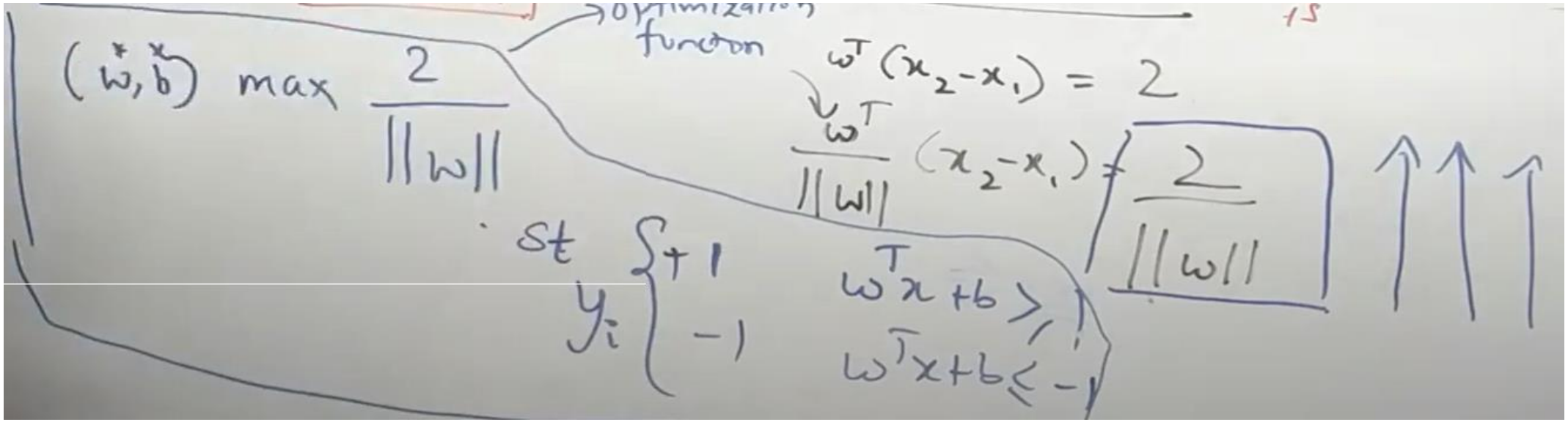


$$w^T x_1 + b = -1$$

$$w^T x_2 + b = 1$$

(-) (-) (-)

$$w^T (x_2 - x_1) = 2$$



$$y_i * w^T x_i + b_i \geq 1$$

$$(w^*, b^*) = \min \frac{\|w\|}{2} + C_i \sum_{i=1}^{n} \xi_i$$

How many errors?

Value of the error