

MACHINE LEARNING

CHAP 6 Support Vector Machine

Jean Tabet
j.tabet@ul.edu.lb

➔ SUPPORT VECTOR MACHINE

Introduction

- A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks.
- It works by finding the optimal hyperplane that best separates different classes in the data.
- The observations closest to the hyperplane are called support vectors – they influence the position and orientation of the hyperplane.
- SVMs aim to maximize the margin between classes to reduce classification errors and improve generalization.
- For data that is not linearly separable, SVMs can use a Soft Margin and Kernel Trick to project it into a higher-dimensional space.

➔ SUPPORT VECTOR MACHINE

Real word applications of SVM

- Text classification (e.g., spam detection, sentiment analysis).
- Image recognition (e.g., digit or face recognition).
- Medical diagnosis based on patient data.
- Financial fraud detection or credit risk analysis.
- Bioinformatics for gene classification.

➔ SUPPORT VECTOR MACHINE

Real word applications of SVM

- Text classification (e.g., spam detection, sentiment analysis).
- Image recognition (e.g., digit or face recognition).
- Medical diagnosis based on patient data.
- Financial fraud detection or credit risk analysis.
- Bioinformatics for gene classification.

→ SUPPORT VECTOR MACHINE

SVM as Linear classifier

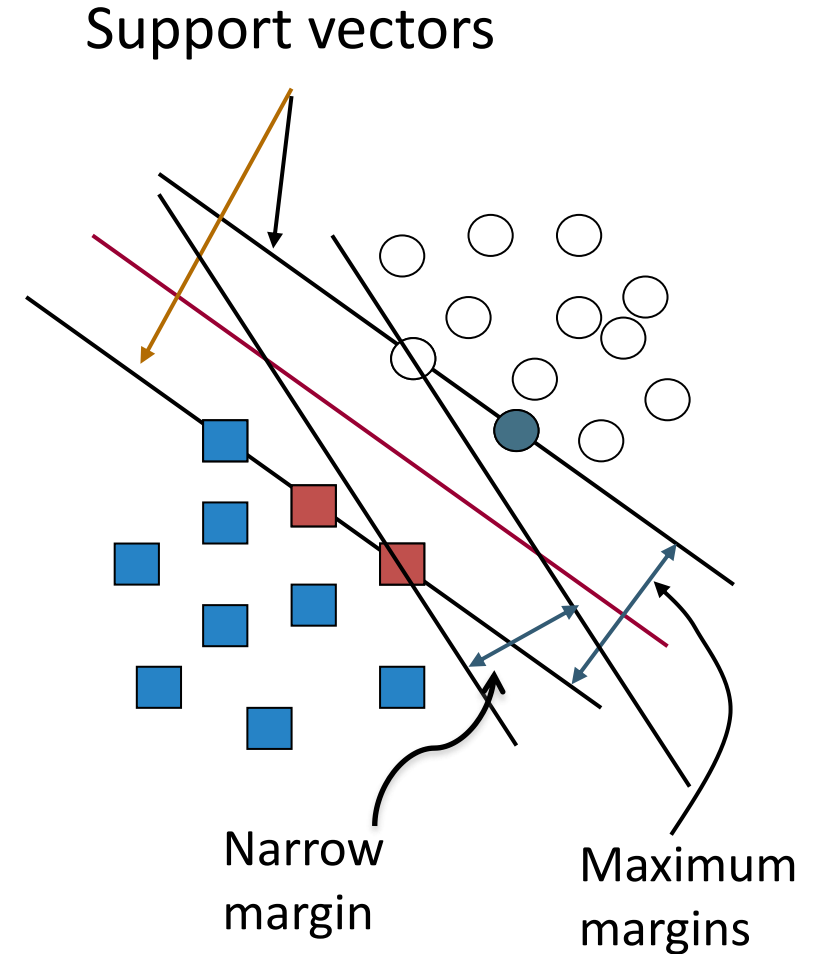
- A Support Vector Machine Classifier searches from a lot of possible solutions of boundaries, the one that separates best the different categories of observations.
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- The optimal hyperplane maximizes the margins.
- The distances between the observations on each side and the threshold in the middle are the same and both reflect the margin.

This line
represents the
decision
boundary:
 $ax + by - c = 0$

➔ SUPPORT VECTOR MACHINE

SVM as Linear classifier

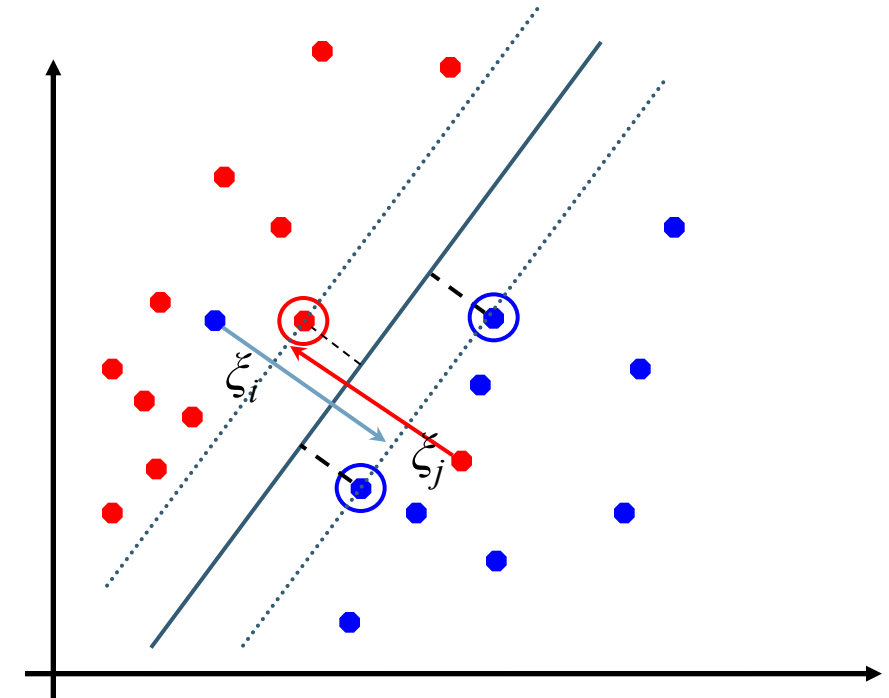
- SVMs maximize the margin around the separating hyperplane.
- We assume for the moment data free from outliers.



➔ SUPPORT VECTOR MACHINE

Slack variable

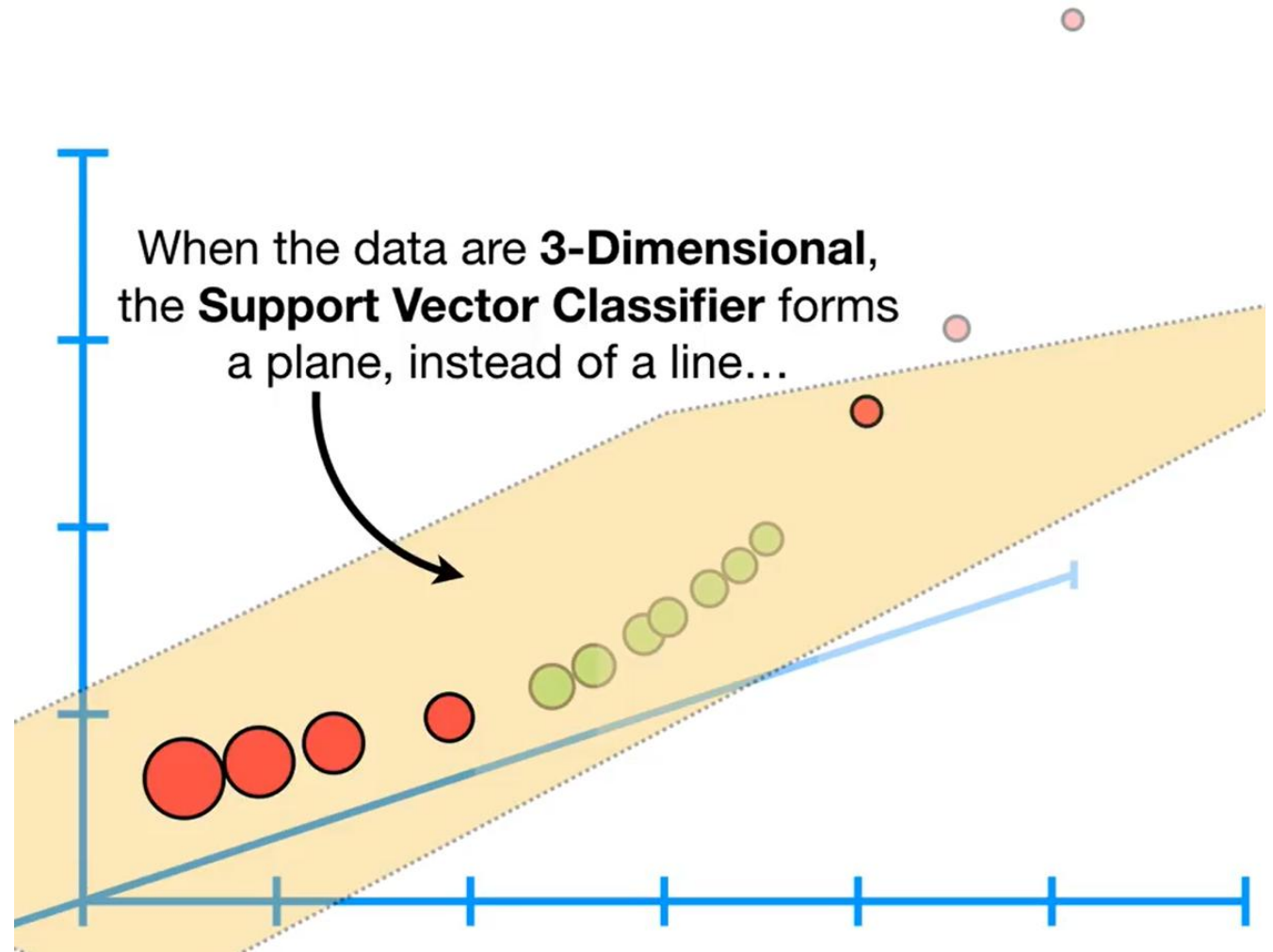
- If the training data is not linearly separable, slack variables ξ_i can be added to allow misclassification allowing some bias.
- In this case, the distance between the observations and the threshold is called a Soft Margin.
- The coefficient C of the slack variables ξ_i is the regularization term. Its optimal value is found by cross validation.
- Small C
 - Smooth decision boundary
 - Model looks at global structure
 - May underfit
- Large C
 - Very complex boundary
 - Model focuses on individual points
 - May overfit



➔ SUPPORT VECTOR MACHINE

SVM in higher dimensions

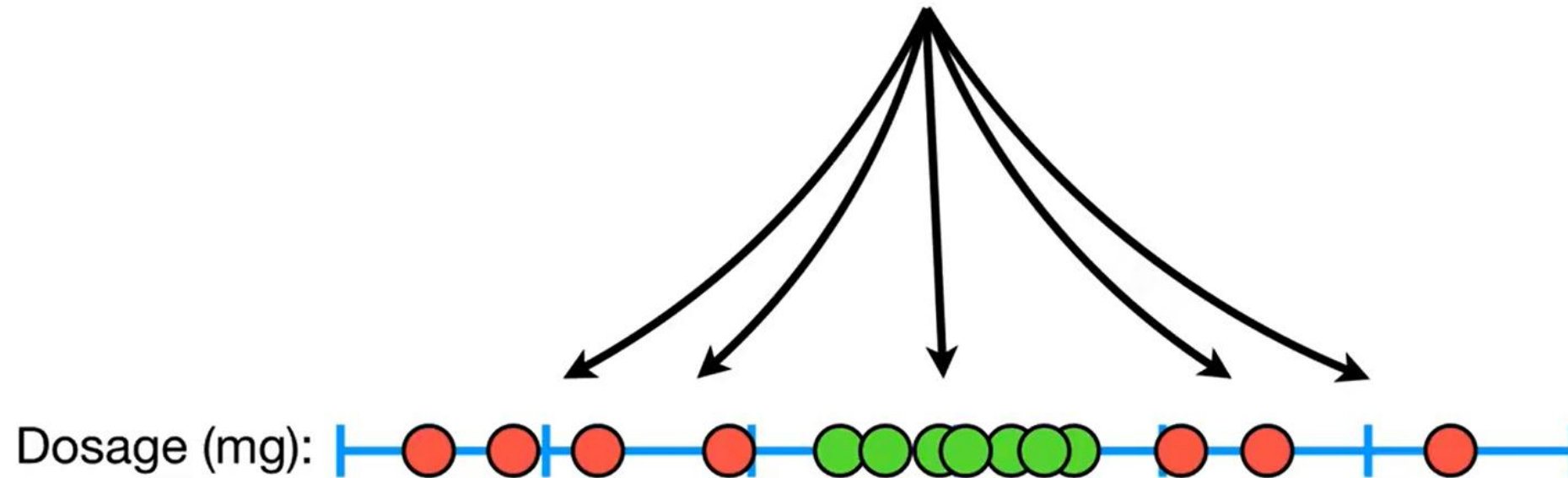
- When the data are 3-Dimensional, the Support Vector Classifier is a 2-Dimensional plane in a 3-Dimensional space.
- And when the data are in 4 or more Dimensions, the Support Vector Classifier is a hyperplane.



→ SUPPORT VECTOR MACHINE

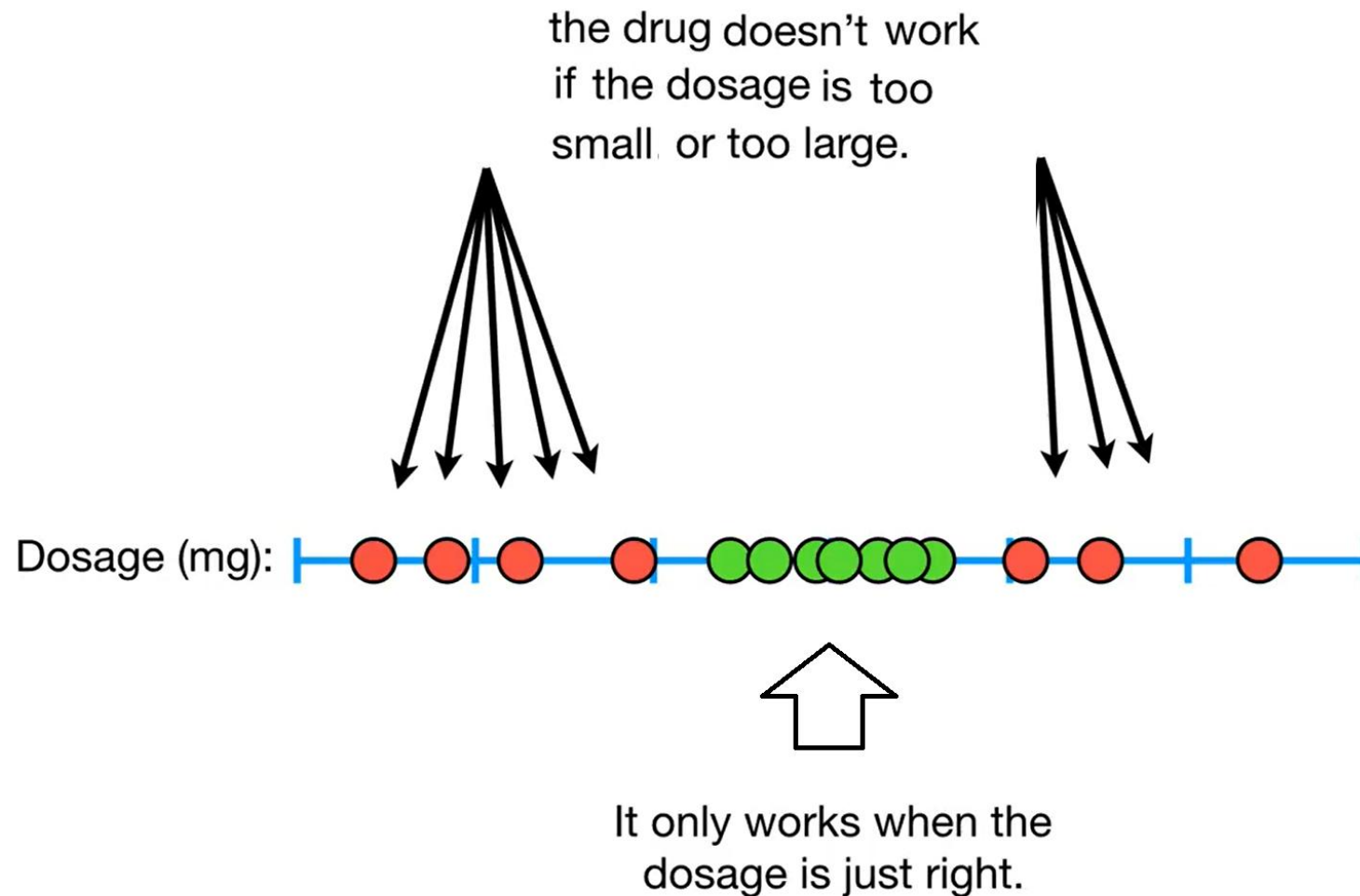
Case of overlapping data

...but what if this was our training data and we had tons of overlap?



→ SUPPORT VECTOR MACHINE

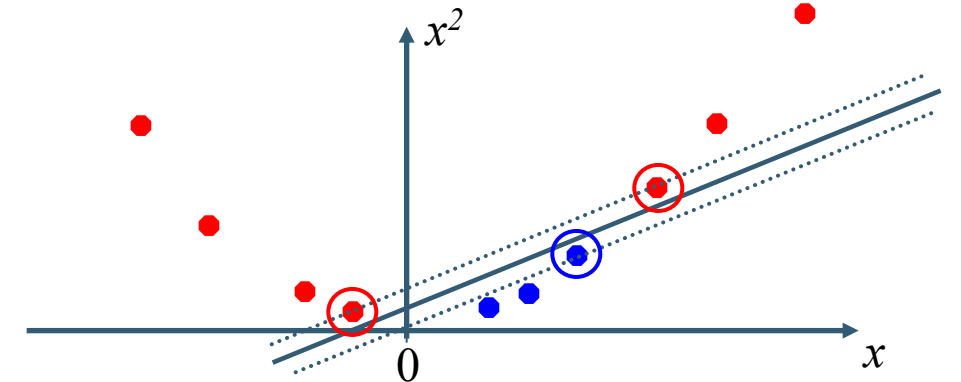
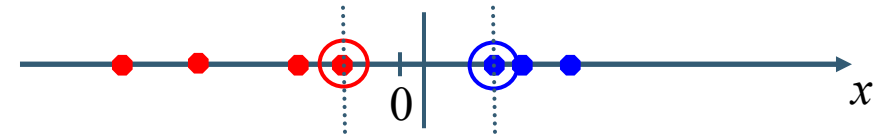
Case of overlapping data



→ SUPPORT VECTOR MACHINE

Case of overlapping data

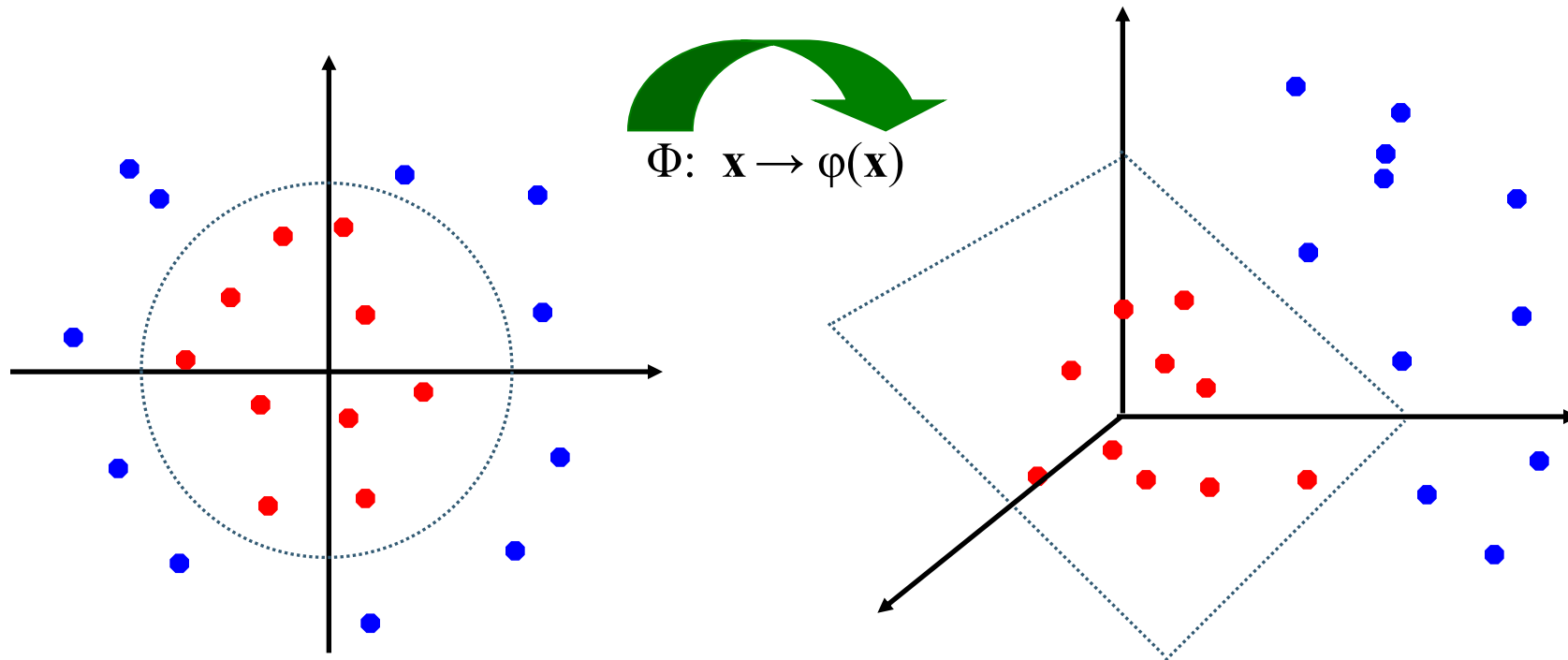
- Datasets that are linearly separable (with some noise) work out great:
- But what are we going to do if the dataset is just too hard?
- How about ... mapping data to a higher-dimensional space:



➔ SUPPORT VECTOR MACHINE

Transformation to higher dimension ➔ Kernel

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



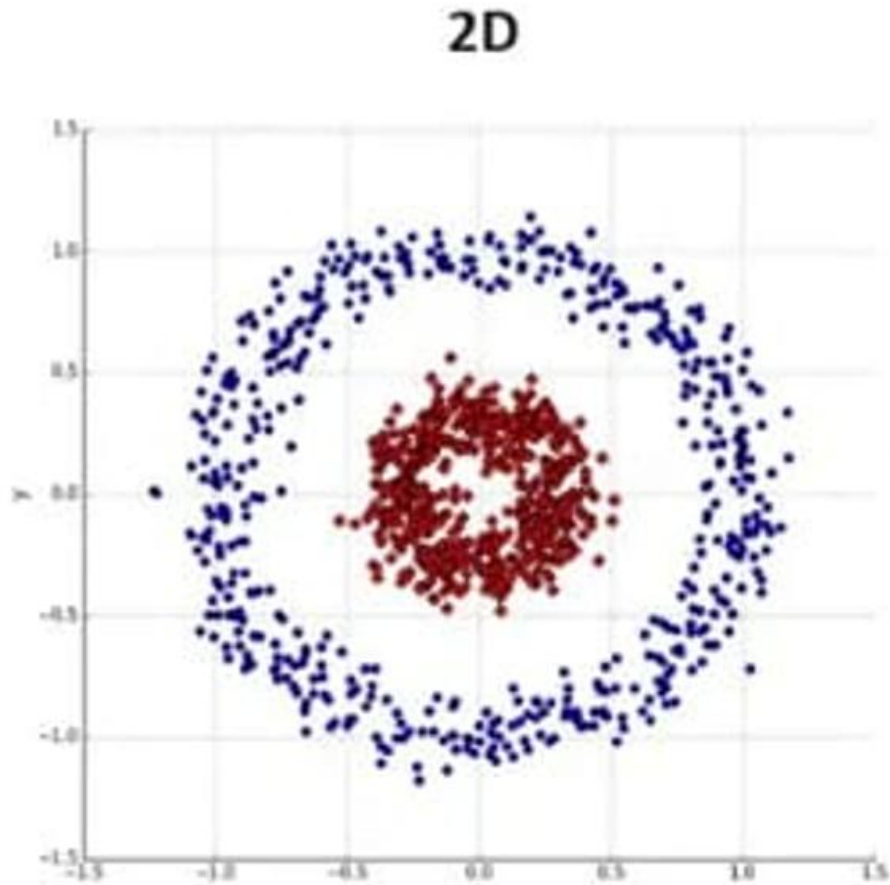
➔ SUPPORT VECTOR MACHINE

Kernel

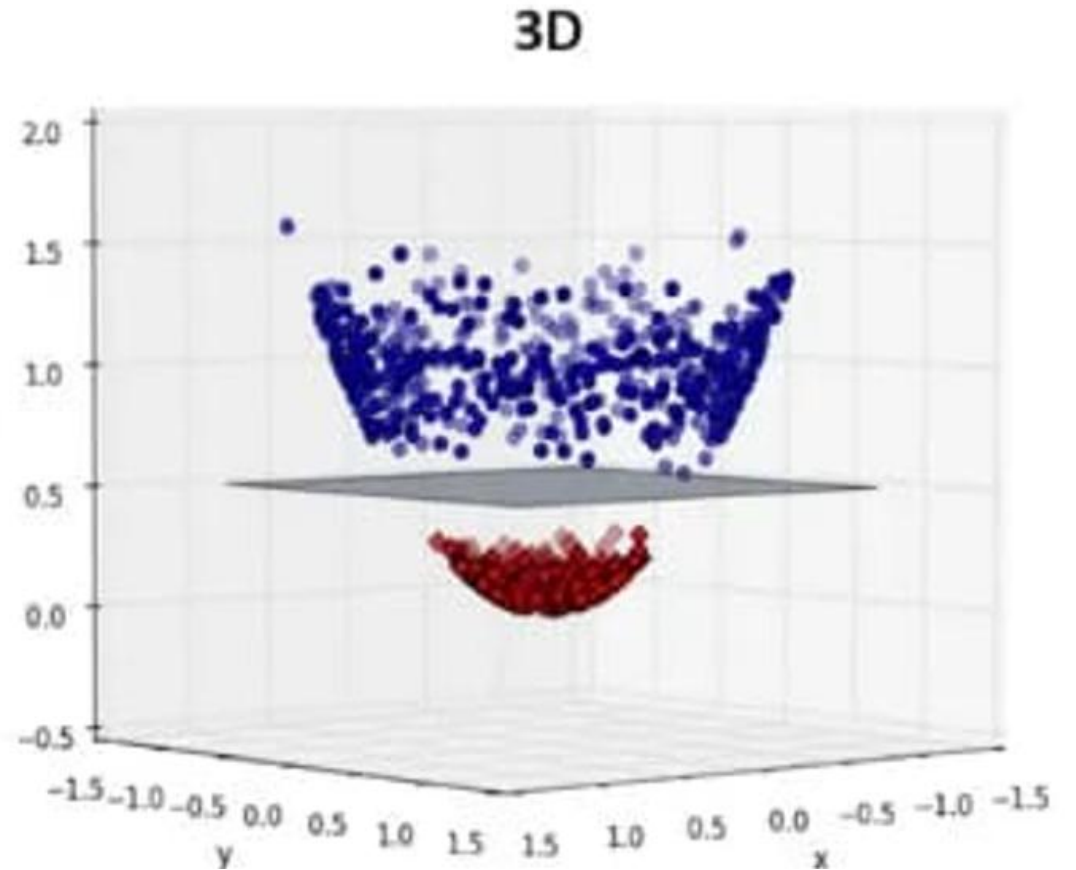
- How do we decide how to transform the data?
- We use a Polynomial Kernel. Its parameter, d , is the degree of the polynomial.
 - $d = 1$, the Polynomial Kernel computes the relationships between each pair of observations in 1-Dimension.
 - $d = 2$, → 2-Dimensional relationships between each pair of observations.
 - $d = 3$, → 3-Dimensional relationships between each pair of observations.
 - $d = 4$ or more, → more dimensions to find a Support Vector Classifier.
- The value for d is found with Cross Validation.

➔ SUPPORT VECTOR MACHINE

2D → 3D kernel

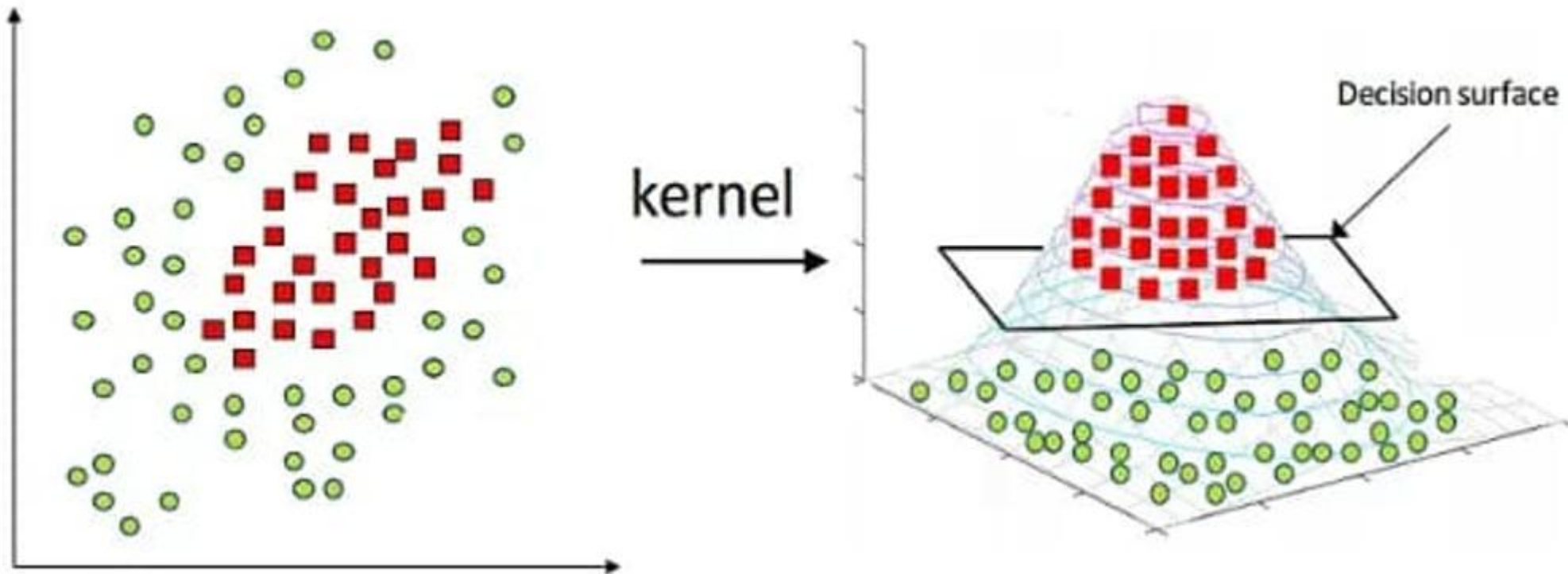


Kernel →



➔ SUPPORT VECTOR MACHINE

2D → 3D kernel



➔ SUPPORT VECTOR MACHINE

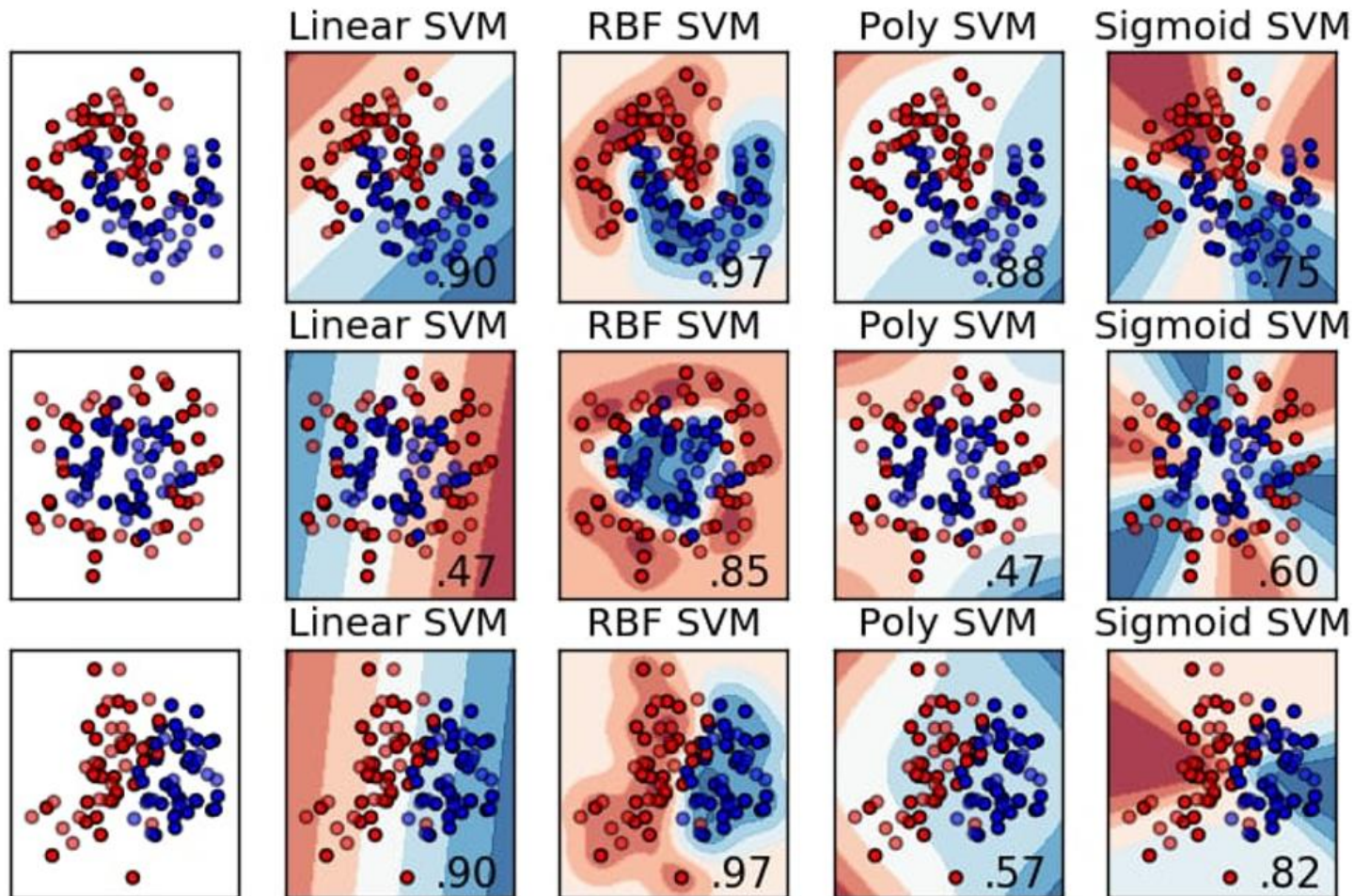
Kernel ['linear', 'rbf', 'poly']

This controls the type of decision boundary the Support Vector Machine (SVM) will use:

- linear: The SVM will try to find a straight-line that separates the data.
 - Use this when the data is linearly separable (can be split with a straight line).
 - Less flexible, but simpler and faster.
- rbf (Radial Basis Function or Gaussian kernel): Allows the SVM to create non-linear decision boundaries by mapping data to a higher-dimensional space.
 - Useful for complex datasets where data points cannot be separated by a straight line.
 - More flexible but can overfit if not tuned well.
- poly (Polynomial) Curved boundaries depending on the degree of the polynomial.
 - degree parameter controls the curve's complexity.
 - Can fit more complex patterns than linear but less flexible than RBF.

➔ SUPPORT VECTOR MACHINE

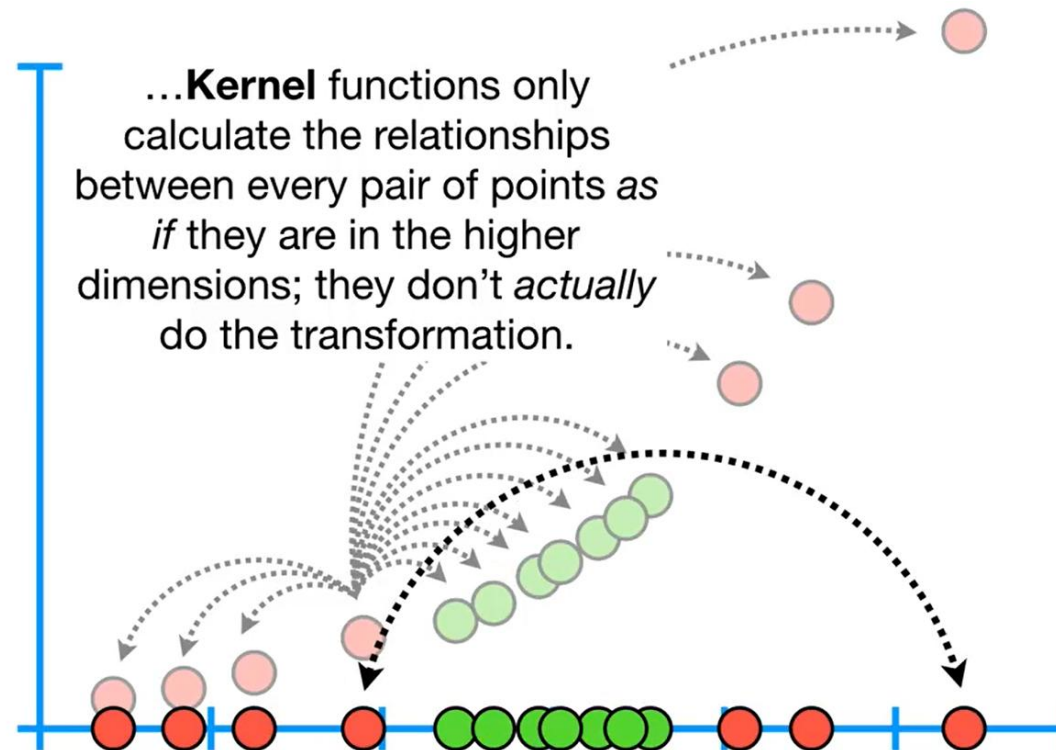
Kernel ['linear', 'rbf', 'poly', 'sigmoid']



→ SUPPORT VECTOR MACHINE

Kernel trick

- The Kernel Trick reduces the amount of computation required for Support Vector Machines by avoiding the math that transforms the data from low to high dimensions.



➔ SUPPORT VECTOR MACHINE

Implementing SVM in Python

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
```

Load dataset

```
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

Split dataset

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Train SVM model

```
model = SVC(kernel='linear', C=1.0)
model.fit(X_train, y_train)
```

Make predictions

```
y_pred = model.predict(X_test)
```

➔ SUPPORT VECTOR MACHINE

Hyperparameter tuning with cross-validation

```
from sklearn.model_selection import GridSearchCV
```

Define parameter grid

```
param_grid = {  
    'C': [0.1, 1, 10],  
    'kernel': ['linear', 'rbf'],  
    'gamma': ['scale', 'auto']  
}
```

Perform grid search

```
grid = GridSearchCV(SVC(), param_grid, refit=True, cv=5)
```

Print best parameters

```
print('Best Parameters:', grid.best_params_)  
print('Best Score:', grid.best_score_)
```

Parameter	Value	Description
kernel	linear	Straight-line decision boundary
kernel	rbf	Curved decision boundary (non-linear)
gamma	scale	Adapts to data variance and feature count
gamma	auto	Uses only feature count (less adaptive)