*Support Vector Machines (SVM)*

* SVM is a supervised machine learning algorithm used for classification (mostly) and regression (less often).
* Its goal is simple yet powerful:
* Find the best possible boundary (aka hyperplane) that separates classes with the maximum margin.
* Core Concepts:
* Hyperplane:
* In 2D: It’s a line
* In 3D: It’s a plane
* In higher dimensions: It’s a “hyperplane”
* Margin: The distance between the hyperplane and the closest points from both classes.
* Support Vectors: The data points lie closest to the hyperplane.
* They’re the most important ones – they “support” the boundary.
* The model is entirely defined by them.
* Maximizing the margin: SVM picks the hyperplane that maximizes this margin to improve generalization (aka better performance on new data).
* What if the data is linearly separable?
* That’s where Kernels come in, they help map your data into a higher-dimensional space where it can be separated by a hyperplane.
* Without actually transforming the data (which would be computationally expensive), the kernel trick computes the inner product in high-dimensional space directly.

|  |  |
| --- | --- |
| Kernel | Use Case |
| Linear | Linearly separable data |
| Polynomial | Curved Boundaries |
| RBF (Radial Basis Function) | Most Common – great for complex patterns |
| Sigmoid | Similar to neural nets |

* Kernel Theory:
* A kernel is a function that transforms your input data into a higher-dimensional space - without explicitly computing that transformation.
* This allows SVM to find a linear boundary in that high-dimensional space, even when the data isn’t linearly separable in the original space.
* We can’t separate red and blue dots on a 2D sheet with a line. But if you lift them into 3D, suddenly a flat plane (hyperplane) can separate them.
* SVM doesn’t actually move the data into higher dimensions.
* Instead, it uses a kernel function to compute the dot product as if the data were in that higher dimensional space – saving mass computation.

|  |  |
| --- | --- |
| Kernel | Best For |
| Linear | Linearly separable data |
| Polynomial | Curved Boundaries |
| RBF / Gaussian | Complex, Wavy patterns |
| Sigmoid | Mimics neutral networks |

Linear Kernel

* K(x, y) = x . y
* Use when:
* Data is linearly separable.
* High Dimensional sparse data (e.g., text classification)
* Pros & Cons:
* Fast and Simple.
* Fewer hyperparameters to tune.
* Doesn’t work for complex, non-linear boundaries.

Polynomial Kernel

* K(x, y) = (x . y + c)^d
* C: constant (controls flexibility)
* D: degree of the polynomial
* Use When:
* Data has interactions between features.
* We want to model curved decision boundaries.
* Pros & Cons:
* Captures more complexity than linear.
* Still interpretable to an extent.
* More sensitive to scale and parameters.
* Can overfit on small datasets.

Radial Basis Function / Gaussian Kernel

K(x, y) = \exp(-\gamma \|x - y\|^2)

* Gamma: defines how far the influence of a single training example reaches.
* Use When:
* You suspect the data is not linearly separable.
* You have complex patterns.
* Pros & Cons:
* Very powerful
* Can handle complex non-linear problems.
* Sensitive to hyperparameters (C, gamma)
* Harder to interpret.

Sigmoid Kernel

K(x, y) = \tanh(\alpha x \cdot y + c)

* Mimics a neural network’s activation function.
* Use When:
* Rarely used.
* Pros & Cons:
* Can add non-linearity
* Not widely used
* Performance is inconsistent and sensitive to parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel | Best For | Common Use Cases | Notes |
| Linear | Linearly Separable Data | Text data (NLP, spam detection) | Fastest |
| Polynomial | Feature Interactions | Bioinformatics, image shape patterns | Tunable |
| RBF | Most non-linear problems | Handwriting, speech, image recognition | Best all-rounder |
| Sigmoid | Experimental only | Rarely used today | Behaves like a neural net |

Hyperparameter Tuning for SVM

C

* Low C = soft margin 🡪 less overfitting
* High C = hard margin 🡪 might overfit noisy data

Gamma

* Too small = underfitting
* Too large = overfitting

Degree – Only for Polynomial Kernel

* Controls the complexity of the decision boundary.
* Higher degrees fit more complex curves.
* But too high 🡪overfitting 🡪slower.