What a data scientist should know about machine learning kernels?

Melanee Group · Follow
Published in MLearning.ai
18 min read · Apr 14

Photo by Robo Wunderkind on Unsplash

In general, a data scientist should have a basic understanding of the following concepts related to kernels in machine learning:

1. What are kernels?

2. Types of kernels.

3. Purpose of kernels.

4. Selection of kernels.

5. Hyperparameter tuning.

6. Limitations of kernels.

Before we discuss the above related to kernels in machine learning, let's first go over a few basic concepts: **Support Vector Machine**, **Support Vectors** and **Linearly vs. Non-linearly Separable Data**.

Support Vector Machine

Support Vector Machine (**SVM**) is a supervised learning algorithm used for classification and regression analysis. The main idea behind **SVM** is to find a hyperplane that separates the data into different classes with the maximum margin. The hyperplane is chosen in such a way that it maximizes the distance between the closest data points (called support vectors) from either class [1].

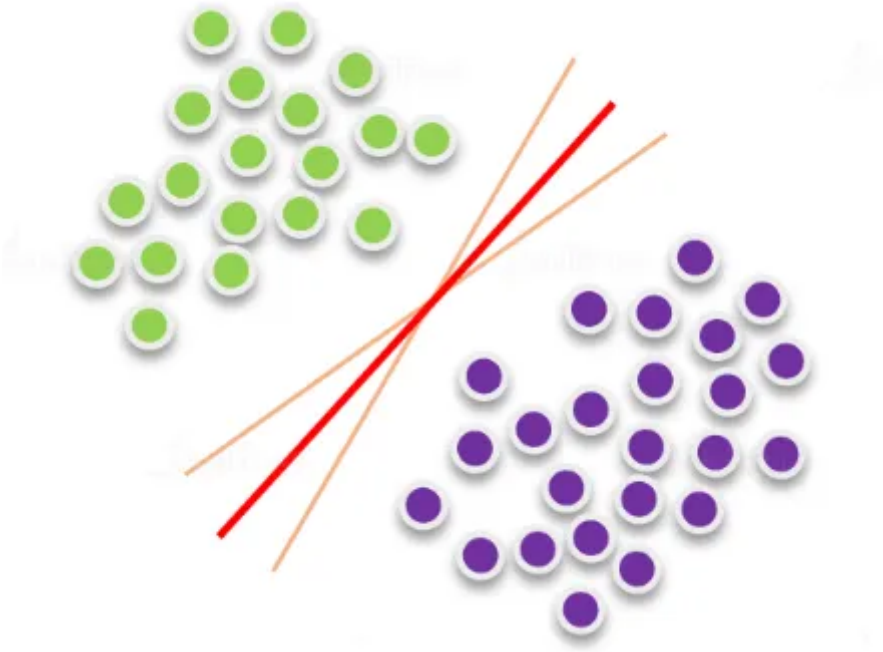

Figure. 1. Data Split by Decision Boundaries [2]

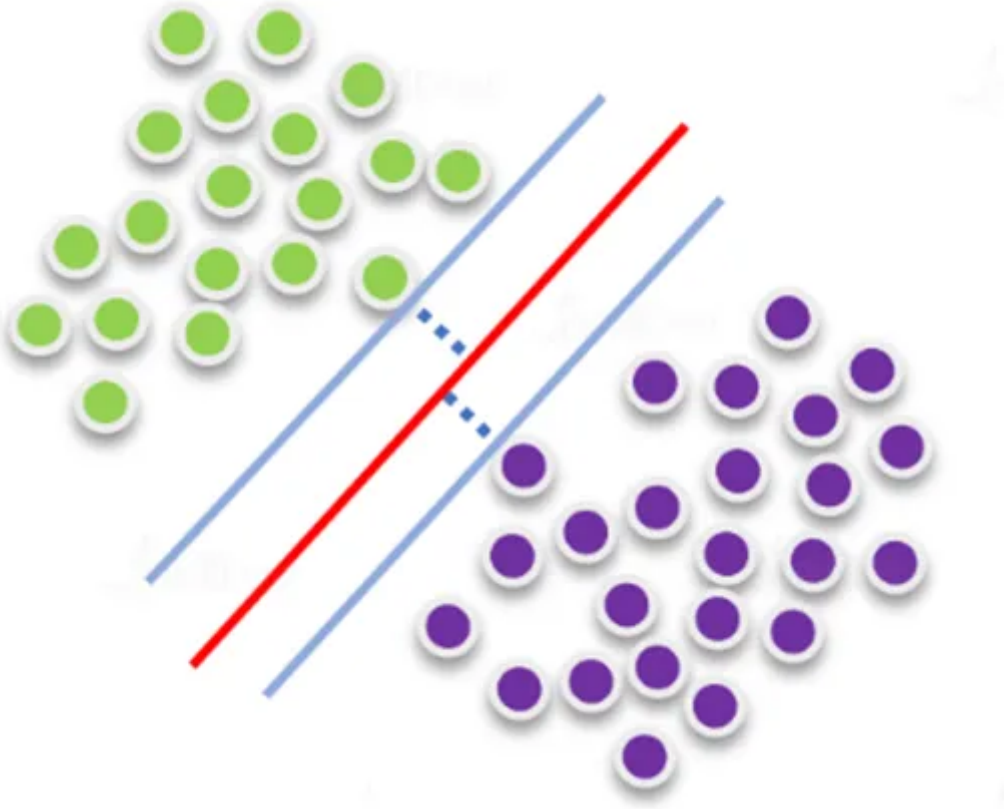

Figure. 2. Margin and Maximum Margin Classifier [2]

Support Vectors

Support Vectors are the data points that lie closest to the hyperplane and are the most critical in defining the hyperplane. If the support vectors are changed, the hyperplane will also change, so support vectors play a crucial role in **SVM**. These support vectors determine the boundary between the classes and, as a result, have the greatest impact on the accuracy of the model [1].

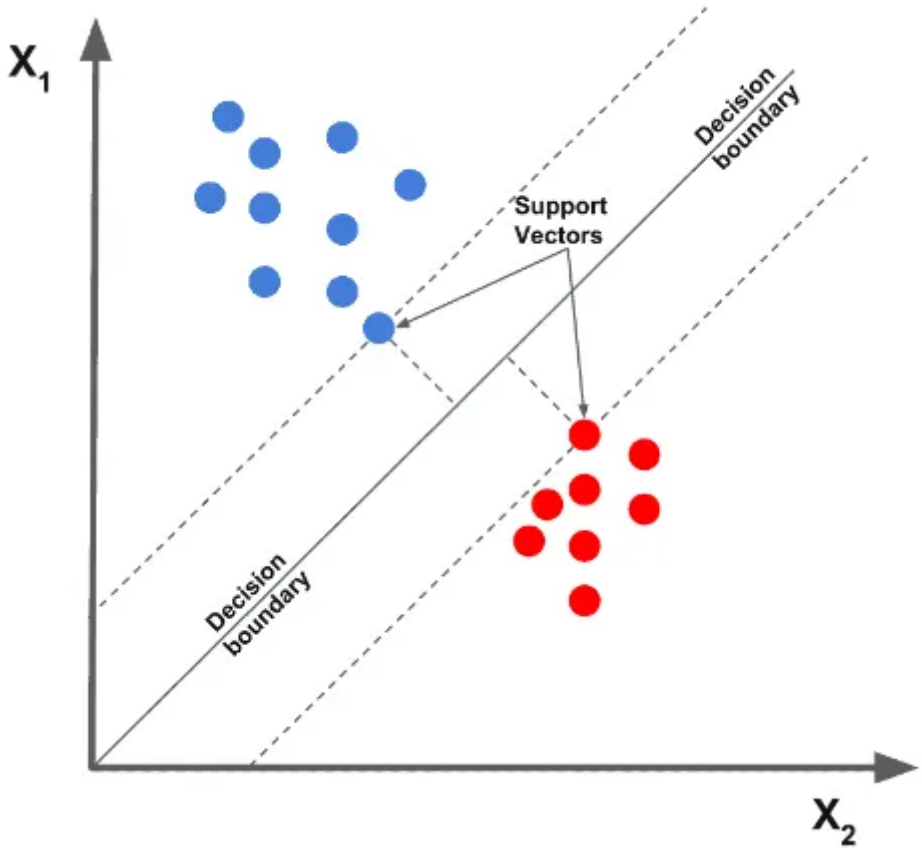

Figure. 3. The points closest to the decision boundary are called support vectors. SVM finds the decision boundary by maximizing its distance from the Support Vectors [3].

Linearly vs. Non-linearly Separable Data

Linearly separable data is data that can be separated into different classes or categories by a single straight line, or hyperplane, in the feature space. That is, the data points belonging to different classes can be separated by a line in such a way that there are no points from different classes on the same side of the line. In other words, the data points can be perfectly separated by a linear boundary [4].

On the other hand, **non-linearly** separable data is data that cannot be separated into different classes by a single straight line or hyperplane in the feature space. This means that the data points belonging to different classes overlap in the feature space and a linear boundary cannot perfectly separate the data points into different classes. In these cases, a non-linear boundary is required to separate the data into different classes [2].

For example, consider a dataset containing two classes of data points, “A” and “B”. If the data points can be separated into two classes by a single straight line, then the data is linearly separable. If the data points cannot be separated into two classes by a single straight line, then the data is non-linearly separable.

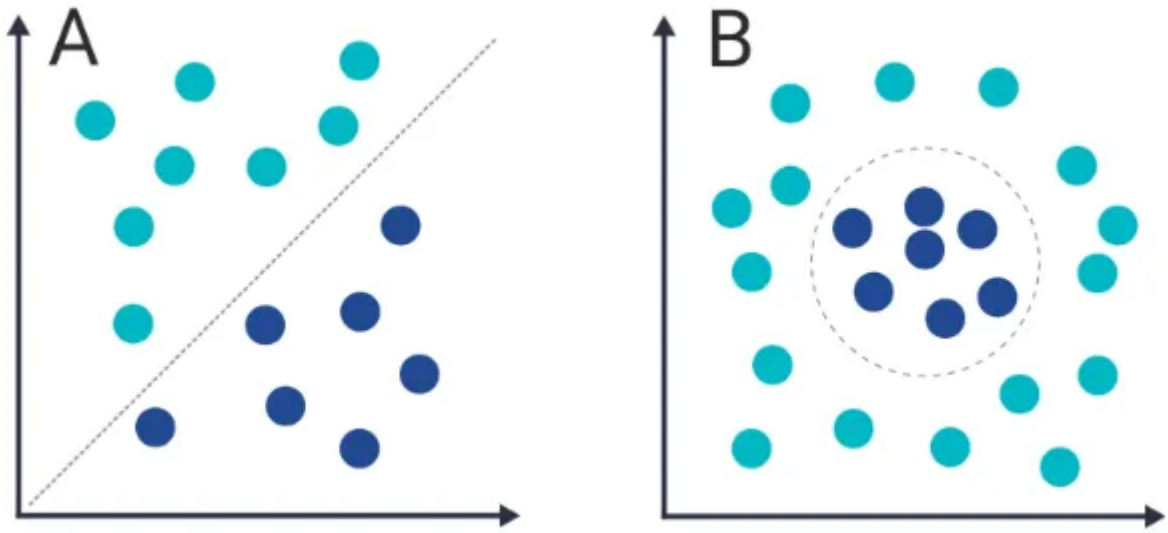

Figure. 4. A: Linearly Separable and B: Non-linearly Separable Data [5]

After comprehending the above concepts, it is now easier to relate and understand the Kernels.

What are kernels?

Machine learning algorithms rely on mathematical functions called “kernels” to make predictions based on input data. A kernel is a mathematical function that maps input data into a higher-dimensional space, where patterns are easier to identify and classify [6].

Kernel Methods:

Kernel methods are a class of machine learning algorithms that make use of kernels to transform the input data into a higher-dimensional space. The transformed data is then used to make predictions about new data or to separate data into different classes. Kernel methods are particularly useful for data that is not linearly separable in its original space. Examples of kernel methods include **SVM**, kernel principal component analysis (**PCA**), and kernel density estimation [7–9].

In summary, kernels and kernel methods are important concepts in machine learning, particularly for problems involving non-linear relationships between variables. By transforming the input data into a higher-dimensional space, kernels allow machine learning algorithms to capture and model non-linear relationships, improving accuracy and performance.

The Kernel Trick:

The kernel trick is a technique used in kernel methods to avoid the need to explicitly compute the mapping of the input data into a higher-dimensional space. Instead of explicitly computing the mapping, the kernel trick uses the dot product between the input data in the original space to calculate the dot product between the mapped data in the higher-dimensional space. This allows the computation to be performed more efficiently, without the need to compute explicit mapping [10–12].

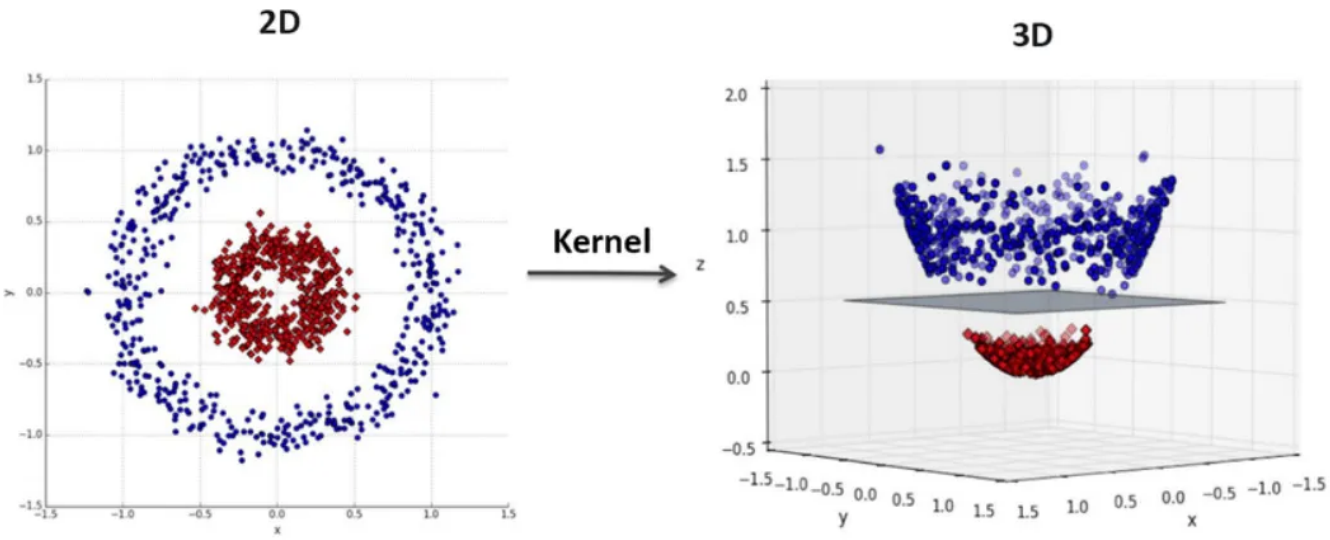

Figure. 5. Non-linear classifier using Kernel trick [13]

The mathematics of the kernel trick is based on the concept of reproducing kernel Hilbert spaces (**RKHS**). An **RKHS** is a Hilbert space that is completely characterized by its inner product, also known as its kernel function. In machine learning, the kernel function maps the input data into a high-dimensional feature space, where a linear boundary can be used to separate the data into different classes.

The key mathematical idea behind the kernel trick is that the dot product between the mapped data in the high-dimensional feature space can be expressed in terms of the dot product between the input data in the original space. This is achieved through the use of a positive definite kernel function, k(x,y), which satisfies the property:

$k(x,y) = \langle \varphi(x), \varphi(y) \rangle$

where ϕ(x) is the mapping of the input data into the high-dimensional feature space and <.,.> is the inner product in the RKHS.

By using the kernel trick, the need to compute the explicit mapping $\phi(x)$ is avoided, and the dot product between the mapped data in the high-dimensional feature space can be calculated using only the dot product between the input data in the original space. This leads to significant computational savings and improved performance for kernel methods [14–17].

Types of kernels

There are many types of kernels that can be used in machine learning, including:

1. Linear Kernels

2. Polynomial Kernels

3. Gaussian Kernels (Radial Basis Function)

4. Sigmoid Kernels

5. Laplacian Kernels

6. Cosine Similarity Kernels

7. Histogram Intersection Kernels

8. Hyperbolic Kernels

In this article, we will discuss the different types of kernels used in machine learning with the mathematics behind them by an example and the scenarios where each is commonly used.

1. Linear Kernels

Linear kernels are the simplest type of kernel, used primarily for linear classification problems. A linear kernel maps the input data into a higher-dimensional space by multiplying the input by a weight matrix. The resulting output is then passed through a linear activation function, which separates the data into two or more classes based on a decision boundary. The linear kernel is ideal for linear problems, such as logistic regression or support vector machines (**SVMs**).

The linear kernel is suitable for use when the data is linearly separable. We can express that a linear kernel is a similarity function that measures the dot product between two vectors in a high-dimensional feature space. Mathematically, it can be expressed as:

$K(x, y) = \langle x, y \rangle$

where <x, y> denotes the dot product of vectors x and y, and K(x, y) is the similarity score between x and y [18].

For example, let's say we have two 2-dimensional vectors x = [1, 2] and y = [3, 4]. Their dot product can be calculated as:

$\langle x, y \rangle = 1 * 3 + 2 * 4 = 3 + 8 = 11$

So, the linear kernel similarity score between x and y would be 11.

2. Polynomial Kernels

Polynomial kernels are similar to linear kernels, but they can capture more complex patterns in the data. They map the input data into a higher-dimensional space by transforming it into a polynomial of a specified degree. The transformed data is then passed through a linear activation function to classify the data. Polynomial kernels are often used for non-linear problems, such as polynomial regression or SVMs.

The equation for the polynomial kernel is given by:

$K(x, x') = (1 + x.x')^d$

where x and x' are the input features, and d is the degree of the polynomial. A higher degree results in a more complex decision boundary [19].

For example, consider a 2-dimensional feature space with input features x = [x1, x2]. The 3rd-degree polynomial kernel function would transform these features into a 6-dimensional space as follows:

$K(x, x') = (1 + x1x'1 + x2x'2)^3 = (1 + x1x'1 + x2x'2)(1 + x1x'1 + x2x'2)(1 + x1x'1 + x2x'2) = 1 + 3(x1x'1 + x2x'2) + 6x1x'1x2x'2 + 3(x1^2x'1^2 + x2^2x'2^2) + x1^3x'1^3 + x2^3x'2^3$

The new features are a combination of the original features and their powers.

3. Gaussian Kernels (Radial Basis Function)

Gaussian kernels, also known as radial basis function (RBF) kernels, are used for non-linear classification problems. They map the input data into a higher-dimensional space by transforming it into a Gaussian distribution. The transformed data is then passed through a non-linear activation function to classify the data. Gaussian kernels are commonly used for classification problems that involve non-linear boundaries, such as decision trees or neural networks.

The mathematical formula for a Gaussian kernel is given by:

$$k(x, x') = exp(-||x - x'||^2 / 2\sigma^2)$$

where x and x' are the two points being compared, ||x — x'|| is the Euclidean distance between the points, and σ is the standard deviation that determines the shape of the Gaussian curve [20].

For example, let's say we have two data points in a two-dimensional space: x = (1, 2) and x' = (3, 4). We can calculate the Gaussian kernel between these two points by using the formula above:

$$k(x, x') = exp(-||(1, 2) - (3, 4)||^2 / 2\sigma^2) = exp(-((3-1)^2 + (4-2)^2) / 2\sigma^2) = exp(-5 / 2\sigma^2)$$

The value of σ determines the shape of the Gaussian curve and the overall value of the kernel. A larger value of σ results in a wider Gaussian curve and a smaller value of the kernel, while a smaller value of σ results in a narrower Gaussian curve and a larger value of the kernel.

In essence, the Gaussian kernel measures how similar the two points are by computing the normal distribution between them. Points that are closer together will result in a larger kernel value, indicating a higher similarity, while points that are further apart will result in a smaller kernel value, indicating a lower similarity.

4. Sigmoid Kernels

Sigmoid kernels are used for binary classification problems, where the goal is to separate the data into two classes. They map the input data into a higher-dimensional space by transforming it into a sigmoid function. The transformed data is then passed through a sigmoid activation function to classify the data. Sigmoid kernels are often used for logistic regression or SVMs.

A sigmoid kernel is a type of radial basis function that measures the similarity between two points in a multi-dimensional space by transforming the dot product between the points into a non-linear space. The mathematical formula for a sigmoid kernel is given by:

$$k(x, x') = tanh(\alpha * x * x' + r)$$

where x and x' are the two points being compared, α is a scalar that determines the steepness of the sigmoid curve, and r is a bias term that shifts the curve left or right [21, 22].

For example, let's say we have two data points in a two-dimensional space: x = (1, 2) and x' = (3, 4). We can calculate the sigmoid kernel between these two points by using the formula above:

$$k(x, x') = tanh(\alpha * (1, 2) * (3, 4) + r) = tanh(\alpha * (1 * 3 + 2 * 4) + r) = tanh(\alpha * 11 + r)$$

The value of α determines the steepness of the sigmoid curve, and the value of r determines the position of the curve. A larger value of α results in a steeper sigmoid curve, while a smaller value of α results in a shallower sigmoid curve. A positive value of r shifts the curve to the right, while a negative value of r shifts the curve to the left.

In essence, the sigmoid kernel measures the similarity between two points by transforming their dot product into a non-linear space. Points that are highly correlated will result in a larger kernel value, indicating a higher similarity, while points that are not correlated will result in a smaller kernel value, indicating a lower similarity.

5. Laplacian Kernels

Laplacian kernels, also known as Laplacian of Gaussian (LoG) kernels, are used in decision trees or neural networks like image processing for edge detection. The Laplacian of the Gaussian function is defined as the second derivative of a Gaussian function. The Laplacian kernel is a 2D filter applied to an image by convolution, where the values in the kernel matrix are the coefficients of the Laplacian of the Gaussian function. The purpose of the Laplacian kernel is to calculate the second derivative of the image intensity, resulting in zero-crossings that correspond to edges in the image. The Laplacian kernel is typically a square matrix with odd

dimensions, for example, 3x3 or 5x5, and is symmetrical around the central element.

Laplacian kernels are used for non-linear classification problems. They map the input data into a higher-dimensional space by transforming it into a Laplacian distribution. The transformed data is then passed through a non-linear activation function to classify the data. The mathematical formula for a Laplacian kernel is given:

$$k(x, x') = exp(-||x - x'|| / \sigma)$$

where x and x' are the two points being compared, ||x — x'|| is the Euclidean distance between the points, and σ is a scalar that determines the scale of the kernel [23, 24].

For example, let's say we have two data points in a two-dimensional space: x = (1, 2) and x' = (3, 4). We can calculate the Laplacian kernel between these two points by using the formula above:

$$k(x, x') = exp(-||(1, 2) - (3, 4)|| / \sigma) = exp(-sqrt((3-1)^2 + (4-2)^2) / \sigma) = exp(-sqrt(5) / \sigma)$$

The value of σ determines the scale of the kernel, with a larger value of σ resulting in a larger kernel value and a smaller value of σ resulting in a smaller kernel value.

6. Cosine Similarity Kernels

Cosine kernels are similarity measures that can be used to compare two vectors in a high-dimensional feature space. Cosine kernels are defined as functions that take two vectors as input and return a scalar value that represents the cosine of the angle between the vectors.

The cosine similarity between two vectors x and y in a high-dimensional feature space is defined as:

$$K(x, y) = (x * y) / (||x|| * ||y||)$$

where (x * y) is the dot product of the vectors x and y, and ||x|| and ||y|| are the Euclidean norms of the vectors.

The cosine similarity ranges from -1 to 1, where 1 indicates that the vectors are identical (have an angle of 0°), 0 indicates that the vectors are orthogonal (have an

angle of 90°), and -1 indicates that the vectors are diametrically opposed (have an angle of 180°) [25–27].

For example, let's consider two vectors x and y in a 3-dimensional feature space, where x = [1, 2, 3] and y = [2, 3, 4]. The cosine similarity between x and y can be computed as follows:

$K(x, y) = (x * y) / (||x|| * ||y||) = (1 * 2 + 2 * 3 + 3 * 4) / (sqrt(1^2 + 2^2 + 3^2) * sqrt(2^2 + 3^2 + 4^2)) = 0.97$

So, in this example, the cosine kernel would compare the two vectors x and y and return a scalar value of 0.97, which indicates a high degree of similarity between the vectors.

7. Histogram Intersection Kernels

Histogram intersection kernels are a type of kernel function that can be used in machine learning to compare histograms, which are representations of the distribution of values in a dataset. The histogram intersection kernel measures the similarity between two histograms by computing the area of overlap between them.

The histogram intersection kernel is defined as follows:

$k(h1, h2) = \sum min(h1(i), h2(i))$

where h1 and h2 are the histograms and h1(i) and h2(i) are the values of the histograms at bin i [28, 29].

For example, let's consider two histograms h1 and h2, where h1 represents the distribution of values for feature A and h2 represents the distribution of values for feature B. The histogram intersection kernel would calculate the similarity between h1 and h2 by computing the area of overlap between them. If the two histograms are very similar, then the area of overlap would be large, indicating that the features are highly correlated. On the other hand, if the two histograms are very different, then the area of overlap would be small, indicating that the features are not highly correlated.

8. Hyperbolic kernels

The mathematics of hyperbolic kernels is based on hyperbolic functions. A hyperbolic function is a type of mathematical function that models many important

physical and biological processes, including the growth and decay of populations, the spread of diseases, and the diffusion of heat and other physical quantities.

In the context of machine learning, hyperbolic kernels are often used as a similarity measure between two instances in a feature space. The kernel function takes as input two instances and outputs a scalar value indicating the similarity between them.

A simple example of a hyperbolic kernel function is the sine hyperbolic kernel, which is defined as:

$K(x, y) = sinh(\alpha x^T y + \beta)$

where α and β are hyperparameters that control the shape of the kernel, and x^Ty is the dot product of the two instances x and y. The dot product measures the similarity between the two instances by counting the number of common features they have. The sine hyperbolic function is then applied to this dot product to produce the final similarity score.

In general, the choice of the hyperbolic kernel will depend on the specific application and the nature of the data being analyzed. Some hyperbolic kernels may be better suited to certain types of data, while others may perform poorly. It's important to carefully consider the choice of kernel function when using machine learning techniques and to experiment with different kernel functions to determine which one works best for a given problem [30, 31].

When and where each kernel is used?

Kernels are commonly used in various machine learning algorithms, particularly in SVM and kernel methods. Here's a brief overview of when and where each kernel is used:

Linear kernel: The linear kernel is the simplest kernel and is used when the data is linearly separable. It is commonly used for simple binary classification problems.

Polynomial kernel: The polynomial kernel is used when the data is not linearly separable. It maps the input data into a higher-dimensional feature space where a linear classifier can be applied.

Radial basis function (RBF) kernel: The RBF kernel is a commonly used non-linear kernel and is equivalent to the Gaussian kernel. It is used for non-linear classification and regression problems.

Sigmoid kernel: The sigmoid kernel is used for binary classification problems and is similar to the logistic regression classifier. It is not commonly used compared to other kernels.

Laplacian kernel: The Laplacian kernel is used for non-linear regression problems and is similar to the Gaussian kernel. It is particularly useful for datasets with complex patterns.

Cosine similarity kernel: The cosine similarity kernel is used for text classification problems, where the similarity between two documents is calculated based on the cosine of the angle between their vectors in a high-dimensional feature space.

Histogram intersection kernel: The histogram intersection kernel is used for image classification problems, where the similarity between two images is calculated based on the intersection of their histograms.

Hyperbolic kernel: The hyperbolic kernel is used for non-linear regression problems and is similar to the Gaussian and Laplacian kernels. It is particularly useful for datasets with complex patterns.

Implement a Kernel Function in Python

As an example, here is a sample code for using the radial basis function (RBF) kernel in Python using the scikit-learn library:

```
import numpy as np
from sklearn.svm import SVC
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
# Load the iris dataset
iris = load_iris ()
X = iris.data
y = iris.target
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split (X, y, test_size=0.2)
# Train the SVM model with the RBF kernel
clf = SVC (kernel='rbf')
clf.fit (X_train, y_train)
# Evaluate the model on the test set
```

```
acc = clf.score (X_test, y_test)
print ("Accuracy:", acc)
```

Selection of Kernels

The choice of kernel depends on the nature of the problem, the type of data, and the computational resources available. It is important to experiment with different kernels to determine the best one for a given problem.

The choice of the kernel in machine learning can greatly impact the performance and accuracy of a model. For instance:

1. **Linear kernels** are ideal for linear problems.

2. **Polynomial kernels** for non-linear problems.

3. **Gaussian kernels** for non-linear classification problems.

4. **Sigmoid kernels** for binary classification problems.

5. **Laplacian kernels** for non-linear classification problems.

It is important to understand the strengths and weaknesses of each type of kernel in order to make an informed decision about which one to use for a given problem.

Hyperparameter Tuning

The performance of a kernel-based algorithm can also be affected by the values of the hyperparameters associated with the kernel. Hyperparameter tuning is the process of finding the optimal values for these hyperparameters to maximize the performance of the algorithm. This is often done using techniques such as cross-validation or grid search.

There are several approaches to hyperparameter tuning for kernel-based algorithms, including:

1. Grid search: In grid search, a set of possible hyperparameter values is defined, and the algorithm is trained and evaluated for each combination of hyperparameter values. The combination with the best performance is then chosen as the final hyperparameter value.

2. Random search: In random search, the hyperparameter values are chosen randomly from a predefined distribution, and the algorithm is trained and evaluated. This process is repeated multiple times, and the combination of hyperparameter values with the best performance is chosen as the final hyperparameter values.

3. Bayesian optimization: In Bayesian optimization, a probabilistic model is used to predict the performance of the algorithm for different hyperparameter values. The hyperparameter values are then chosen based on the model's prediction, and the process is repeated until a satisfactory performance is achieved [32, 33].

Limitations of kernels

Kernels have some limitations, including the choice of kernel, the sensitivity to hyperparameters, and the potential for overfitting. Additionally, kernels can be computationally intensive, especially for large datasets, and can take a long time to train. It's also important to keep in mind that kernel-based methods may not always perform well when dealing with highly imbalanced or noisy data.

Overall, understanding kernels and how to select and tune them is an important aspect of being a data scientist.

I hope this article gives you an overview of what a data scientist needs to know about kernels and helps you to understand kernels just a little bit better.

References

[1] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273–297