**Support Vector Machine Tutorial Video Transcript**

SVM aims to find the optimal decision boundary that separates positive and negative samples using what we call the *widest street approach*. This means the decision boundary is placed to maximise the margin between the two classes.

The key idea is that the margin separates the data points so that positive and negative samples lie on opposite sides of the decision boundary. We have a vector w perpendicular to the boundary and an unknown vector u that we are trying to classify. To classify u, we project it onto w. The larger the projection the further along w the point lies. If it crosses w, we classify it as a positive sample. If the projection lies opposite to w, we classify it as a negative sample.

To build the decision boundary, we need a decision rule. At the start, we don’t know the values of w and b, so to compute them, we introduce constraints based on the classification of points. To simplify the math, we combine these two constraints into a single equation by introducing a variable y\_i. This works because multiplying y\_i with the respective inequality flips the sign for negative samples, effectively combining the rules. This is a bit of the math behind how it works.

We use the value 1 in the margin for simplification purposes. The margin is defined as being 1 unit away from the decision boundary, both for positive and negative samples. Points that lie exactly on the margin are called **support vectors**, and the unified equation for these points is equal to 0. For points correctly classified but not on the margin, the value is greater than 0. Misclassified points fail to satisfy the constraints, so their value becomes less than 0.

The ultimate goal of SVM is to maximize the width of the margin. To do this, we define a point lying exactly on the positive margin and another point lying exactly on the negative margin. The difference between these two vectors gives the width of the margin. Since w is a unit normal vector perpendicular to both the boundary and the margin, we project it onto the difference vector, yielding a scalar value that represents the margin width.

To calculate the margin width, we compute the distance between a positive support vector and a negative support vector by taking their difference vector. We then project this difference vector onto the unit normal vector w. The result is the margin width.

To maximize this margin, we minimize the norm of w, which simplifies to minimizing the squared norm of w. To simplify optimization:

• We drop a factor of 2 because it doesn’t affect the optimization goal.

• We include a factor of  1/2  so that when taking the derivative, it avoids extra factors of 2.

Now we have an optimization problem: minimize the objective function (squared norm of w) subject to constraints. These constraints are represented by the unified equation we derived earlier. When dealing with optimization problems with constraints, we use **Lagrange multipliers**.

The general form of the Lagrangian equation incorporates both the objective function and the constraints. By taking the derivatives with respect to w and b, and setting them to zero, we find the optimal values.

• The derivative with respect to w shows that w is influenced only by support vectors where **α\_i not equal to zero.**

• The derivative with respect to b ensures that positive and negative samples contribute equally to the margin.

The final expression shows that the weight vector w depends only on the dot product of sample vectors.

When we expand the Lagrangian function, we compute the **α\_i** values. If **α\_i > zero**, the corresponding point is identified as a **support vector**, and these vectors directly contribute to calculating w. After performing the substitutions, we obtain the final weight vector w. Substituting w and b, we arrive at the decision boundary equation for the SVM model.

Finally, we can substitute w into the decision rule. This shows that the decision rule depends on the dot product between the support vectors and the unknown vector being classified.

Let’s consider the iris dataset and apply a linear kernel with different values of the parameter:

• When is large, the model behaves like a hard-margin SVM.

• The margin is narrow and doesn’t allow misclassification within the margin. This approach is ideal when there are no outliers in the dataset.

• When is small, the model behaves like a soft-margin SVM.

• The margin is wider, allowing for some misclassification within the margin.

• This makes the model more flexible and ideal when there is noise or outliers in the data.

For the iris dataset, in both scenarios, whether is large or small, the model achieved 100% accuracy.

Importantly, even when points lie inside the margin, they aren’t necessarily misclassified—they simply do not influence the decision boundary. As long as the decision boundary correctly classifies the point, the accuracy remains the same.

Let’s say we have a dataset that isn’t linearly separable. In this scenario, SVM uses an approach called the **kernel trick**.

When data points are not linearly separable in their current feature space, we use a technique called **feature transformation.** This involves mapping the original points into a higher-dimensional space. This space might make the data linearly separable. The optimization and decision rule both rely solely on the dot products of the transformed points. Computing explicitly for every point can be computationally expensive in a very high-dimensional space.

Instead of explicitly calculating, we define a kernel function, which computes the dot product in the transformed space directly. This is known as the **kernel trick.** It allows us to implicitly perform the transformation without actually calculating.

Here are a few popular kernels:

• The **linear kernel.**

• If we change the degree parameter, it becomes a **polynomial kernel.**

• And then there’s the **Gaussian kernel.**

Now, we can define the decision rule based on the kernel function. When classifying a new point , we compute its relationship with the support vectors using the kernel function. This gives us the decision function .

After applying the polynomial kernel of degree 3 to the data points, the points are transformed into a higher-dimensional space. This transformation makes the data linearly separable in this higher-dimensional space, even if it wasn’t linearly separable in the original feature space. In this transformed space, the SVM will compute a decision boundary that maximizes the margin between the two classes.

One of the best things about SVM is that its objective function involves minimizing the squared norm of the weight vector , which is quadratic and convex. This guarantees a unique global minimum, ensuring the optimization always converges to the optimal decision boundary.

This property makes SVM ideal for various real-world classification tasks like sentiment analysis, handwriting recognition, and fraud detection.