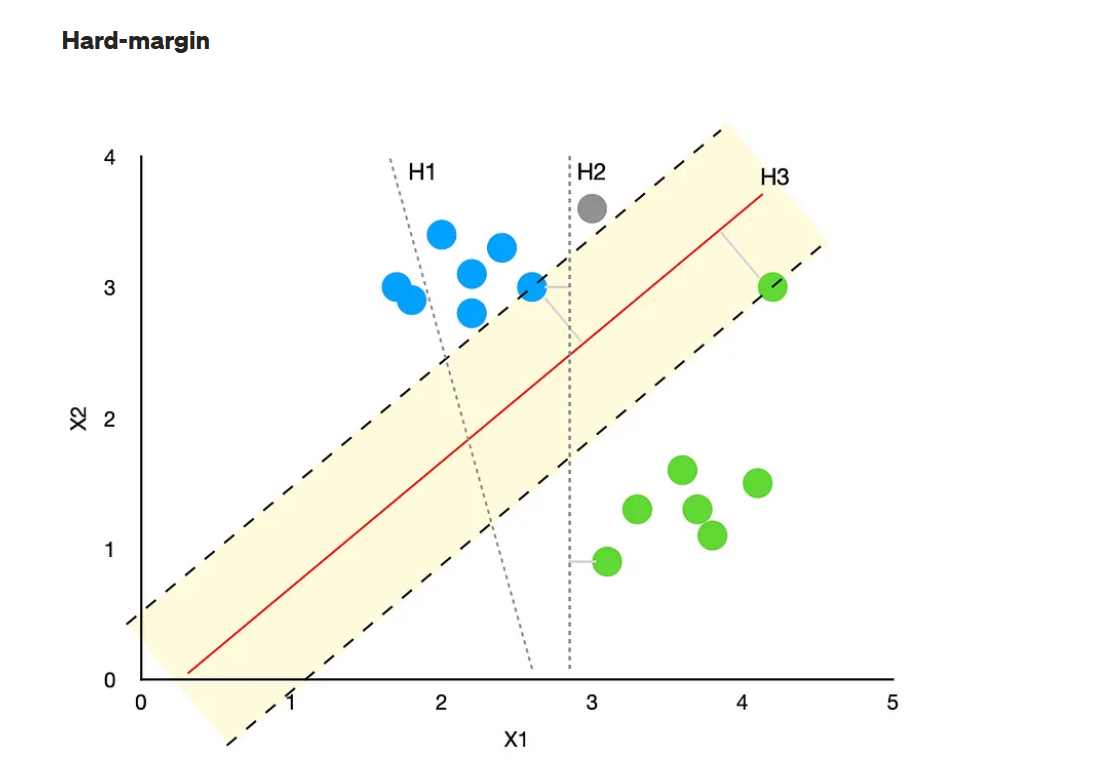
**Support Vector Machine (SVM).**

Let’s assume we have a set of points that belong to two separate classes. We want to separate those two classes in a way that allows us to correctly assign any future new points to one class or the other.

SVM algorithm attempts to find a hyperplane that separates these two classes with the highest possible margin. If classes are fully linearly separable, a **hard-margin** can be used. Otherwise, it requires a **soft-margin**

*Note, the points that end up on the margins are known as****support vectors****.*

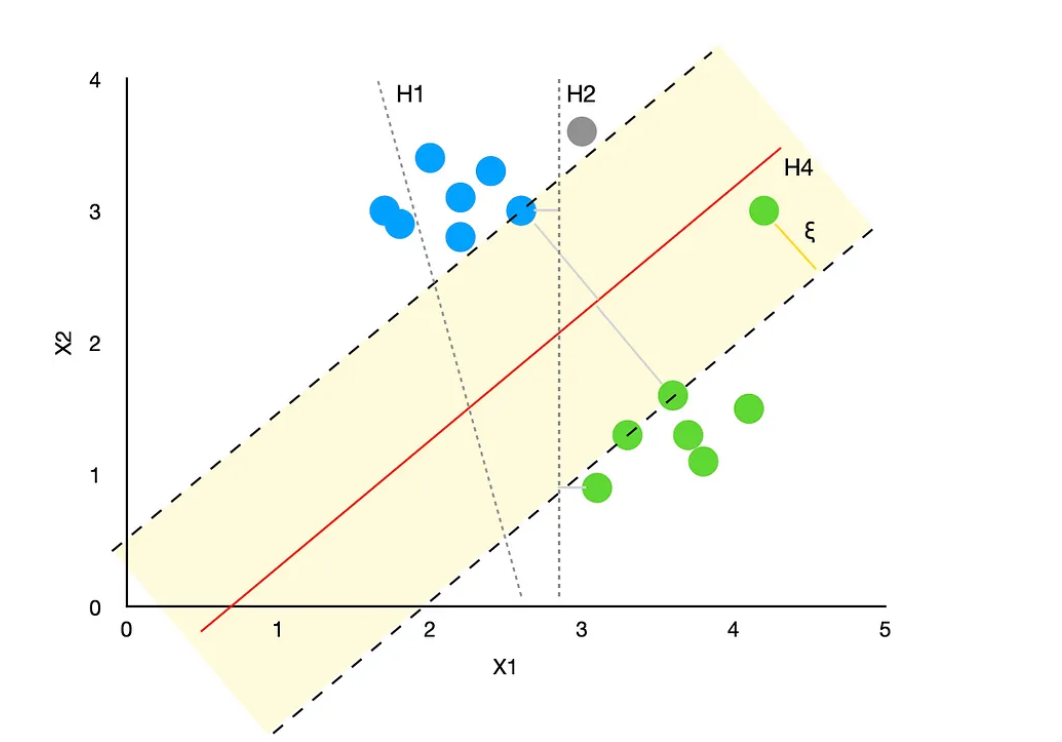
****

* Hyperplane called “**H1**” cannot accurately separate the two classes; hence, it is not a viable solution to our problem.
* The “**H2**” hyperplane separates classes correctly. However, the margin between the hyperplane and the nearest blue and green points is tiny. Hence, there is a high chance of incorrectly classifying any future new points. E.g., the new grey point (x1=3, x2=3.6) would be assigned to the green class by the algorithm when it is obvious that it should belong to the blue class instead.
* Finally, the “**H3**” hyperplane separates the two classes correctly and with the highest possible margin (yellow shaded area). Solution found!

Note, finding the largest possible margin allows more accurate classification of new points, making the model a lot more robust. You can see that the new grey point would be assigned correctly to the blue class when using the “H3” hyperplane

## **Soft-margin**

Sometimes, it may not be possible to separate the two classes perfectly. In such scenarios, a **soft-margin** is used where some points are allowed to be misclassified or to fall inside the margin (yellow shaded area). This is where the “slack” value comes in, denoted by a greek letter ξ (xi).

****

Using this example, we can see that the “H4” hyperplane treats the green point inside the margin as an outlier. Hence, the support vectors are the two green points closer to the main group of green points. This allows a larger margin to exist, increasing the model’s robustness.

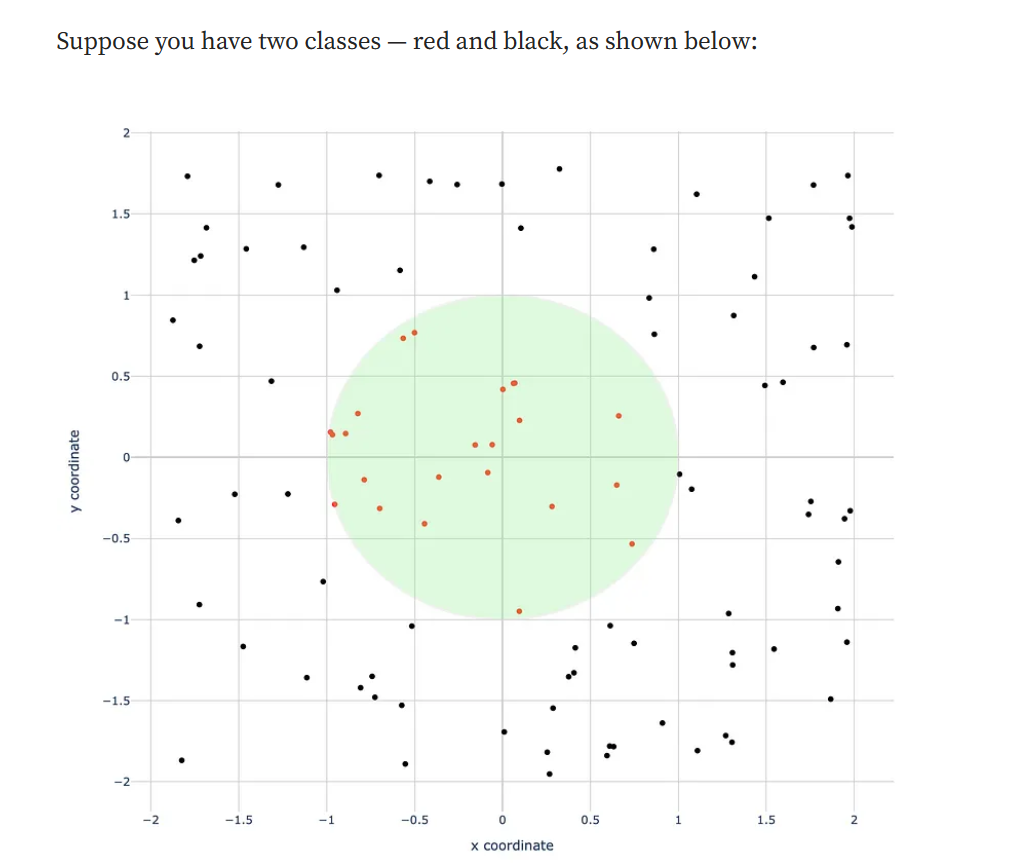
Note, the algorithm allows you to control how much you care about misclassifications (and points inside the margin) by adjusting the hyperparameter C. Essentially, C acts as a weight assigned to ξ. A low C makes the decision surface smooth (more robust), while a high C aims at classifying all training examples correctly, producing a closer fit to the training data but making it less robust.

*Beware, while setting a high value for C is likely to lead to a better model performance on the training data, there is a high risk of overfitting the model, producing poor results on the test data.*

# ****Kernel trick****

The above explanation of SVM covered examples where blue and green classes are linearly separable. However, what if we wanted to apply SVMs to non-linear problems? How would we do that?

This is where the kernel trick comes in. A **kernel is a function** that takes the original non-linear problem and transforms it into a linear one within the higher-dimensional space. To explain this trick, let’s study the below example.

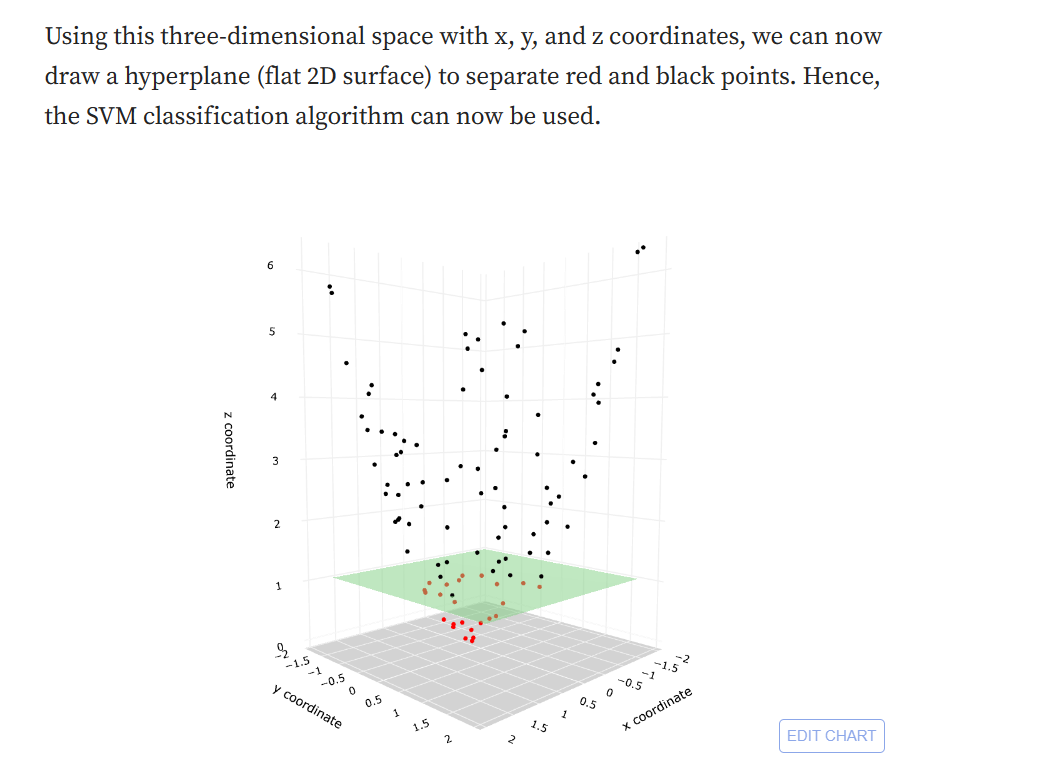
****

As you can see, red and black points are not linearly separable since we cannot draw a line that would put these two classes on different sides of such a line. However, we can separate them by drawing a circle with all the red points inside it and the black points outside it.

## **How to transform this problem into a linear one?**

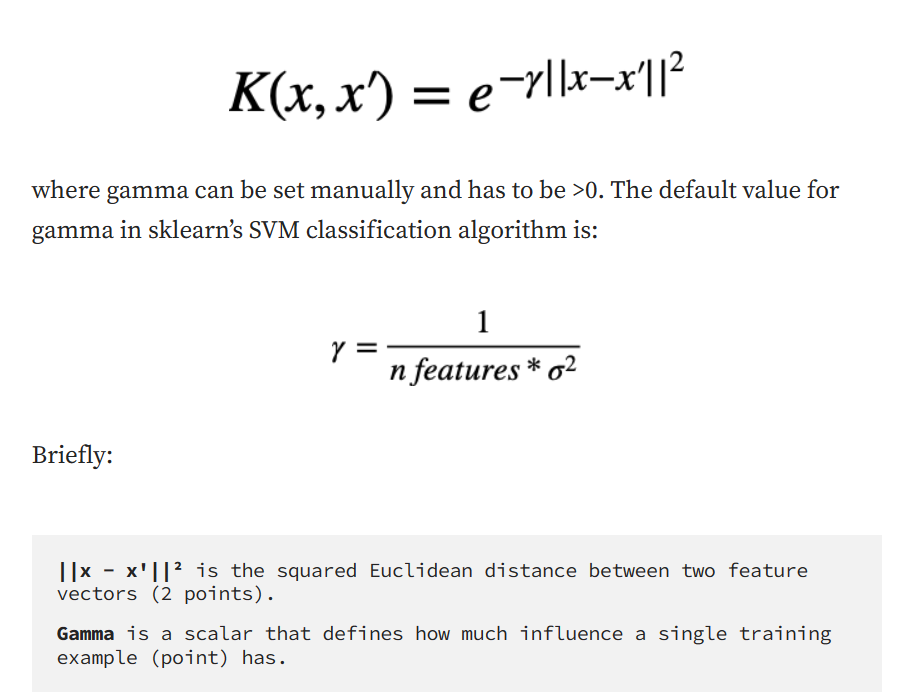
Let’s add a third dimension and make it a sum of squared x and y values:

z = x² + y²

****

# ****Radial Basis Function (RBF) kernel and Python examples****

RBF is the default kernel used within the sklearn’s SVM classification algorithm and can be described with the following formula:

****

So, given the above setup, we can control individual points' influence on the overall algorithm. The larger gamma is, the closer other points must be to affect the model.

 It is a costly operation to actually transform data points to a high-dimensional feature space. The algorithm does not actually transform the data points to a new, high dimensional feature space. Kernelized SVM compute decision boundaries in terms of similarity measures in a high-dimensional feature space without actually doing a transformation. I think this is why it is also called kernel trick.

The C value parameter:

When we increase the C value, the margin gets smaller. **Thus, the models with low C values tend to be more generalized, to avoid overfitting**. The difference becomes clearer with larger datasets.

Gamma Parameter:

Gamma is a hyperparameter used with non-linear SVM. One of the most commonly used non-linear kernels is the radial basis function (RBF). Gamma parameter of RBF controls the distance of the influence of a single training point.

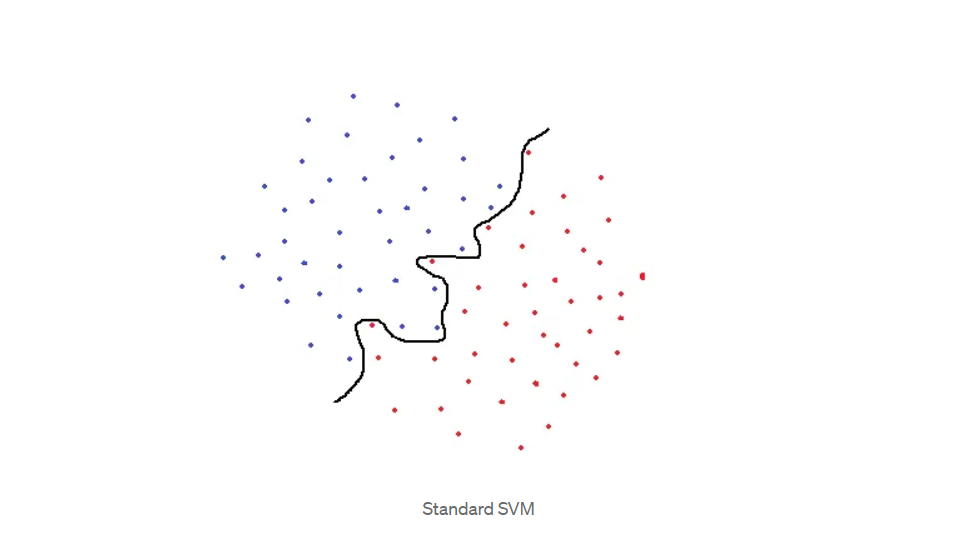
Low values of gamma indicate a large similarity radius which results in more points being grouped together. **For high values of gamma, the points need to be very close to each other in order to be considered in the same group (or class), this could cause overfitting.**

Typical values for c and gamma are as follows. However, specific optimal values may exist depending on the application:

0.0001 < gamma < 10

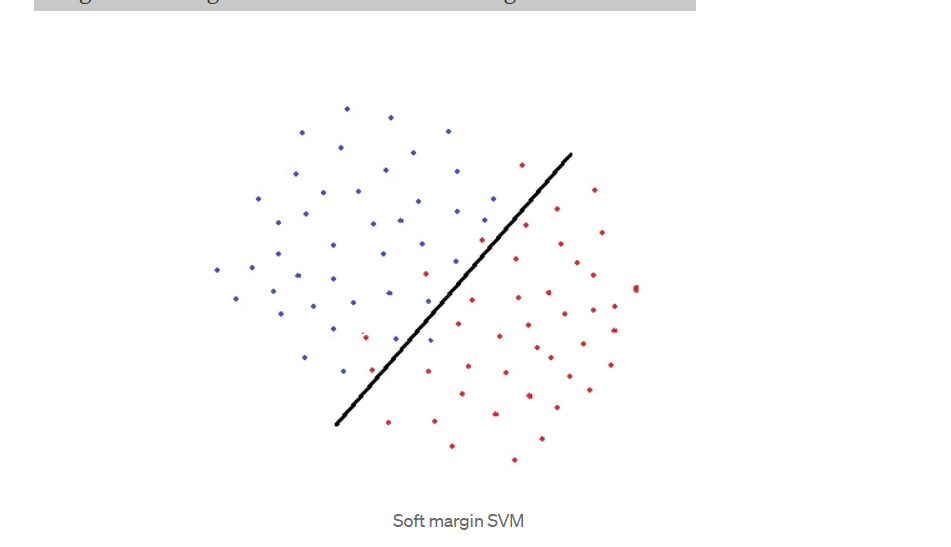
0.1 < c < 100

**High values Of C and Gamma:**

****

A standard SVM would try to separate blue and red classes by using the black curve line as a decision boundary. However, this is a too specific classification and highly likely to end up overfitting. An overfit SVM achieves a high accuracy with training set but will not perform well on new, previously unseen examples. This model would be very sensitive to noise and even very small changes in data point values may change the classification results. The SVM that uses this black line as a decision boundary is not generalized well to this dataset.

To overcome this issue, in 1995, Cortes and Vapnik, came up with the idea of “**soft margin**” SVM which allows some examples to be misclassified or be on the wrong side of decision boundary. Soft margin SVM often result in a better generalized model. In our example, the decision boundary for soft margin SVM might look like the black straight line as below:

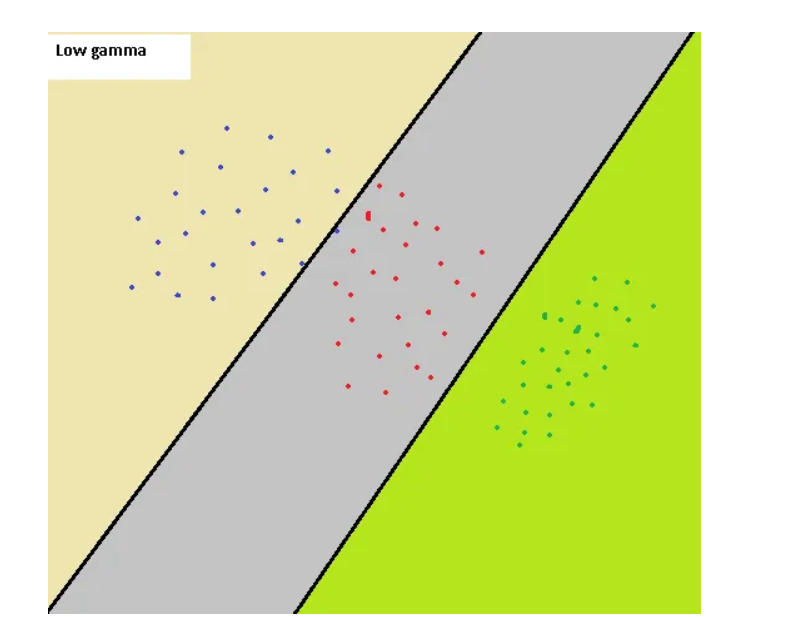
****

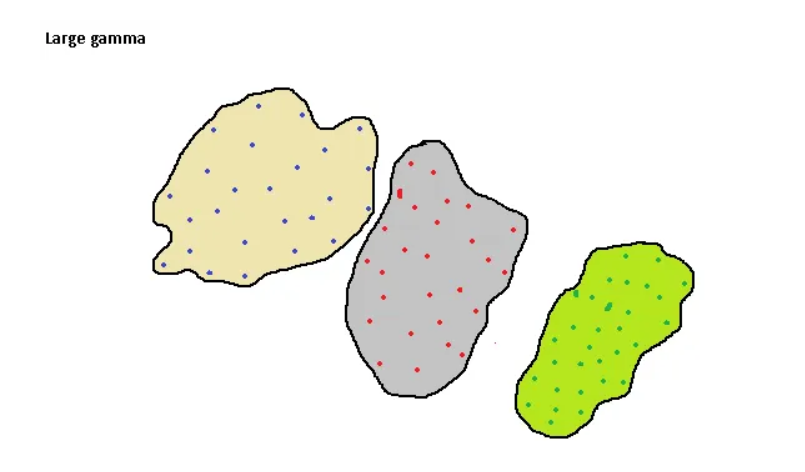
There are some misclassified points but we end up having a more generalized model. When determining the decision boundary, a soft margin SVM tries to solve an optimization problem with the following goals:

* Increase the distance of decision boundary to classes (or support vectors)
* Maximize the number of points that are correctly classified in the training set

There is obviously a trade-off between these two goals. Decision boundary might have to be very close to one particular class to correctly label all data points in training set. However, in this case, accuracy on test dataset might be lower because decision boundary is too sensitive to noise and to small changes in the independent variables. On the other hand, a decision boundary might be placed as far as possible to each class with the expense of some misclassified exceptions. This trade-off is controlled by **c parameter.**

Gamma parameter of RBF controls the distance of influence of a single training point. Low values of gamma indicates a large similarity radius which results in more points being grouped together. For high values of gamma, the points need to be very close to each other in order to be considered in the same group (or class). Therefore, models with very large gamma values tend to overfit. Following visualizations explain the concept better:

****

****

The first image represents the case with a low gamma values. Similarity radius is large so all the points in the colored regions are considered to be in the same class. For instance, if we have a point the right bottom corner, it is classified as “green” class. On the other hand, the second image is the case with large gamma. For data points to be grouped in the same class, they must fall in the tight bounded area. Thus, a small noise may cause a data point to fall out of a class. Large gamma values are likely to end up in overfitting.

As the gamma decreases, the regions separating different classes get more generalized. Very large gamma values result in too specific class regions (overfitting).

# Gamma vs C parameter

For a linear kernel, we just need to optimize the c parameter. However, if we want to use an RBF kernel, both c and gamma parameter need to optimized simultaneously. If gamma is large, the effect of c becomes negligible. If gamma is small, c affects the model just like how it affects a linear model. Typical values for c and gamma are as follows. However, specific optimal values may exist depending on the application:

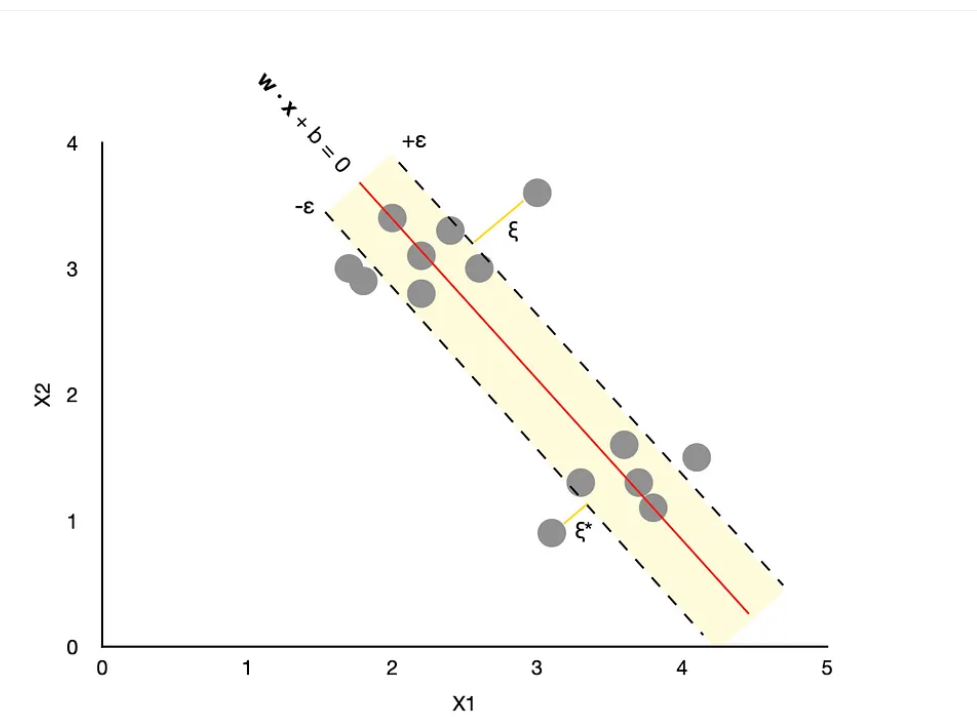
0.0001 < gamma < 10

0.1 < c < 100

**Support Vector Regressor:**

In general, SVR is quite similar to SVM, but there are some notable differences:

* SVR has an additional tunable parameter **ε (epsilon)**. The value of epsilon determines the width of the tube around the estimated function (hyperplane). Points that fall inside this tube are considered as correct predictions and are not penalized by the algorithm.
* The support vectors are the points that fall outside the tube rather than just the ones at the margin, as seen in the SVM classification example.
* Finally, “slack” (ξ ) measures the distance to points outside the tube, and you can control how much you care about it by tuning a regularization parameter C



A simple way to think about SVR is to imagine a tube with an estimated function (hyperplane) in the middle and boundaries on either side defined by ε. **The algorithm's goal is to minimize the error by identifying a function that puts more of the original points inside the tube while at the same time reducing the “slack.”**

While the above explanations focus on linear examples, SVM and SVR algorithms can also handle non-linear situations through a kernel trick. A kernel is a function (you can choose between a few different ones) that takes the original non-linear problem and transforms it into a linear one, which is then handled by the algorithm in a higher-dimensional space.

SVM for classification:

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

SVM for regression:

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html>