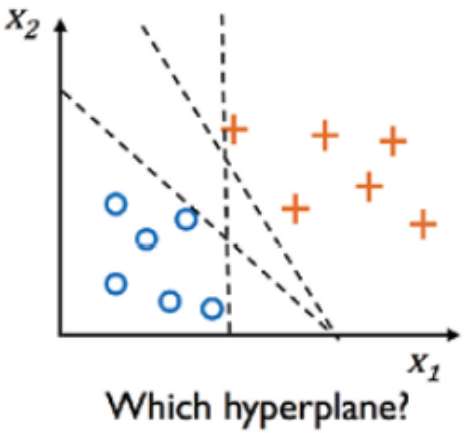
CS5805 : Machine Learning I
Lecture #13

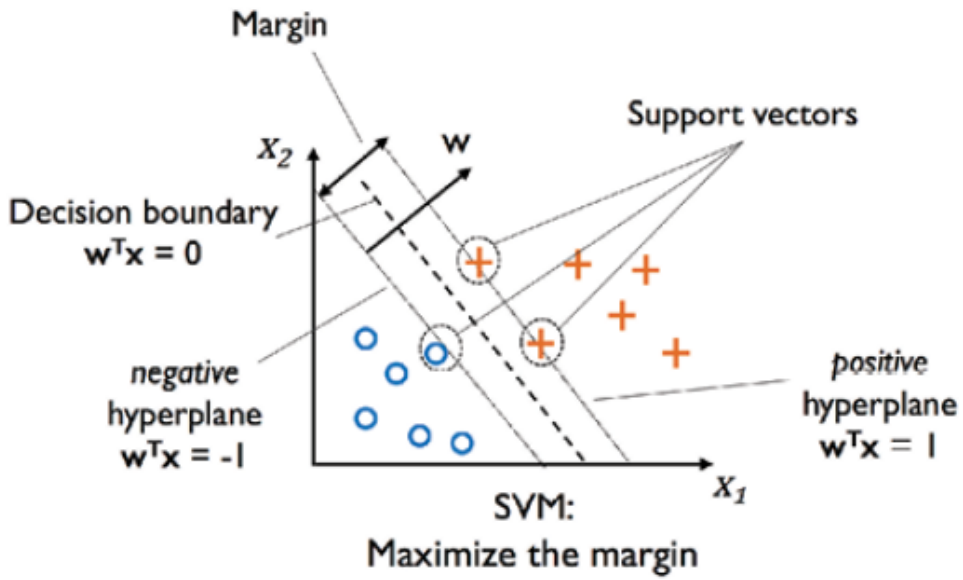
Reza Jafari, Ph.D

Collegiate Associate Professor
rjafari@vt.edu

Support Vector Machine

- A powerful and widely used learning algorithm is the support vector machine (SVM).
- Using the perceptron algorithm, we minimized misclassification errors.
- In SVM the optimization objective is to maximize the margin.
- The margin is defined as the distance between the separating hyperplane (decision boundary) and the training examples that are closest to this hyperplane, which are called **support vectors**.





What is a hyperplane?

- In a p-dimensional space, a hyperplane is a flat affine subspace of dimension $p-1$.
- For instance, in two dimensions, a hyperplane is a flat one-dimensional subspace, in other words, a line.
- In three dimensions, a hyperplane is a flat two-dimensional subspace that is, a plane.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

- In $p > 3$ dimensions, the notation of a $(p-1)$- dimensional flat subspace still applies.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = \boxed{\beta^T \cdot \mathbf{X}} = 0$$

- If a point $X = (X_1, X_2, ..., X_p)$ satisfies the above equation, then X lies on the hyperplane.

Hyperplane...

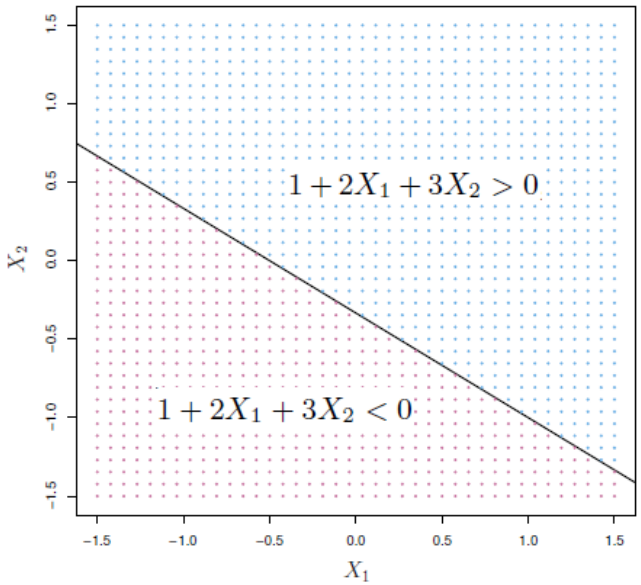
- If **X** does not satisfy the hyperplane equation, then **X** lies to one side of the hyperplane

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p > 0$$

- On the other hand, **X** lies on the other side of the hyperplane.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p < 0$$

- Hyperplane is dividing p-dimensional space into two halves.



Classification Using a Separating Hyperplane

- Suppose the Data matrix X consists of n training observations in p-dimensional space and these observations fall into two classes $y_1, ..., y_n \in -1, 1$, binary classification.

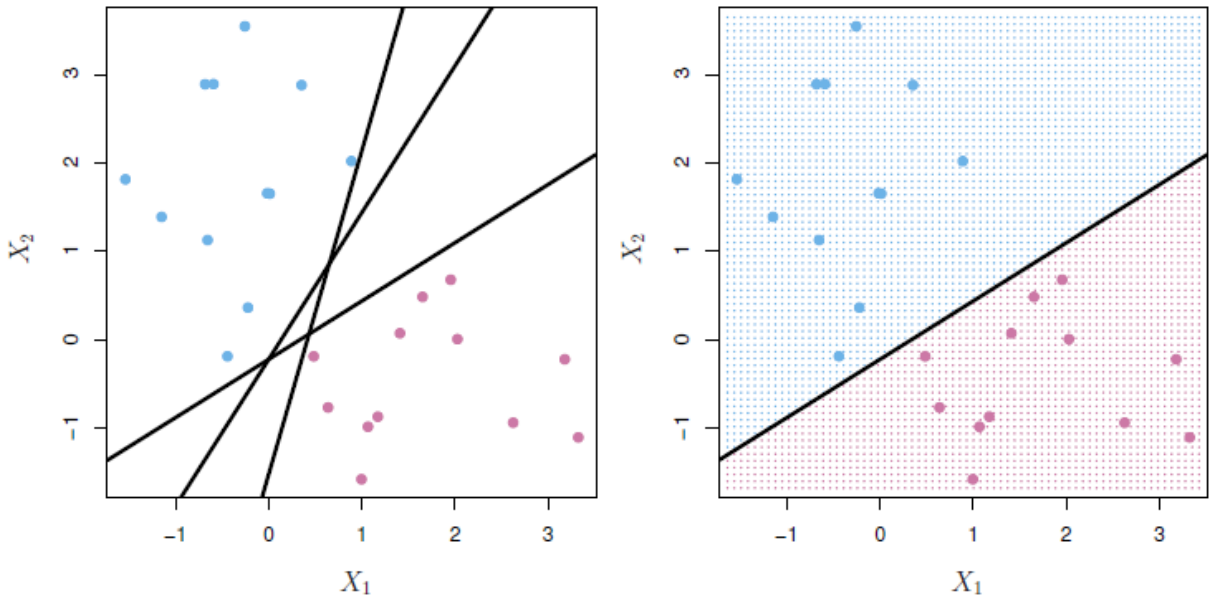
$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, ...x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

- Test observation $x^* = (x_1^*, ..., x_p^*)^T$. Develop a classifier that classifies the test observation using it feature measurements.
- If a separating hyperplane exits then it has the following property:

$$\beta_0 + \beta_1 x_{i1} + \beta_{i2} x_2 + ... + \beta_p x_{ip} > 0 \ if \ y_i = 1$$
$$\beta_0 + \beta_1 x_{i1} + \beta_{i2} x_2 + ... + \beta_p x_{ip} < 0 \ if \ y_i = -1$$
$$\boxed{y_i(\beta_0 + \beta_1 x_{i1} + \beta_{i2} x_2 + ... + \beta_p x_{ip}) \geq 0}$$

Example

- Two classes of observations shown in blue and purple.
- Left : Three separating hyperplanes, out of many possible.
- Right : Separating hyperplane and the two regions. If a test observation falls in the blue portion, the grid will be assigned to blue class.



Notes on separating hyperplane

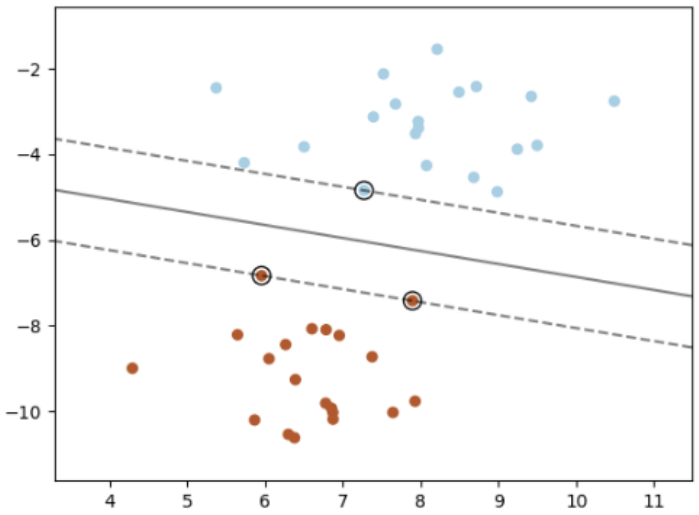
- If a separating hyperplane exists, we can use it to construct a very natural classifier: a test observation is assigned a class depending on which side of the hyperplane it is located.
- That is, we classify the test observations x^* based on the sign of $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + ... + \beta_p x_p^*$.
- If $f(x^*)$ is positive, then we assign the observation to class 1, and if $f(x^*)$ is negative, then we assign to class -1.
- We can also make use of the magnitude of $f(x^*)$, if $f(x^*)$ is far from zero, then this means that x^* lies far from the hyperplane, so we can be confident about the class assignment.
- On the other hand if $f(x^*)$ is close to zero, then x^* is located near hyperplane and so we are less certain about the class assignment.
- A classifier that is based on a separating hyperplane leads to linear decision boundary.

The Maximal Margin Classifier

- If data is lineally separable then there exist infinite number of separating hyperplanes.
- A maximal margin hyperplane is the separating hyperplane that is farthest from the training observations.
- **Margin**? Compute the $\perp$ distance from each training observation to a given separating hyperplane; the smallest such distance is the minimal distance from observations to the hyperplane, and it known as the **margin**.
- The **maximal margin hyperplane** is the separating hyperplane for which the margin is largest, that is the hyperplane that has the farthest minimum distance to the training observations.

The Maximal Margin Classifier

- The dotted lines, parallel to the hyperplane in the following diagram are the margins and the distance between both these dotted lines (Margins) is the Maximum Margin.
- A margin passes through the nearest points from each class; to the hyperplane.
- The angle between these nearest points and the hyperplane is 90°.
- These points are referred to as "Support Vectors". Support vectors are shown by circles in the diagram below.

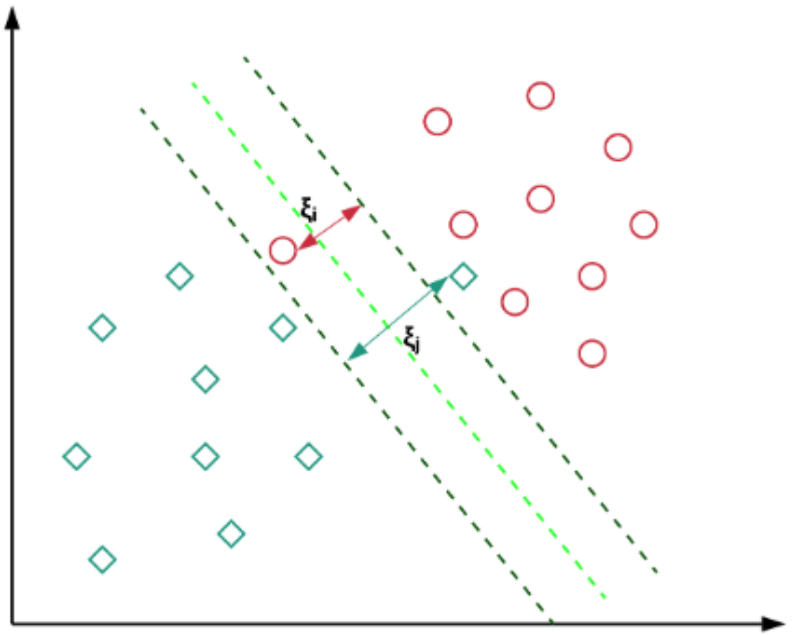


Drawbacks: Maximal Margin Classifier

- This classifier is heavily reliant on the support vector and changes as support vectors change. As a result, they tend to overfit.
- Can't be used for data that isn't linearly separable. Since the majority of real-world data is non-linear. As a result, this classifier is inefficient.
- The maximum margin classifier is also known as a Hard Margin Classifier because it prevents misclassification and ensures that no point crosses the margin. It tends to **overfit** due to hard margin.
- An extension of the Maximal Margin Classifier,Support Vector Classifier was introduced to address the problem associated with it.

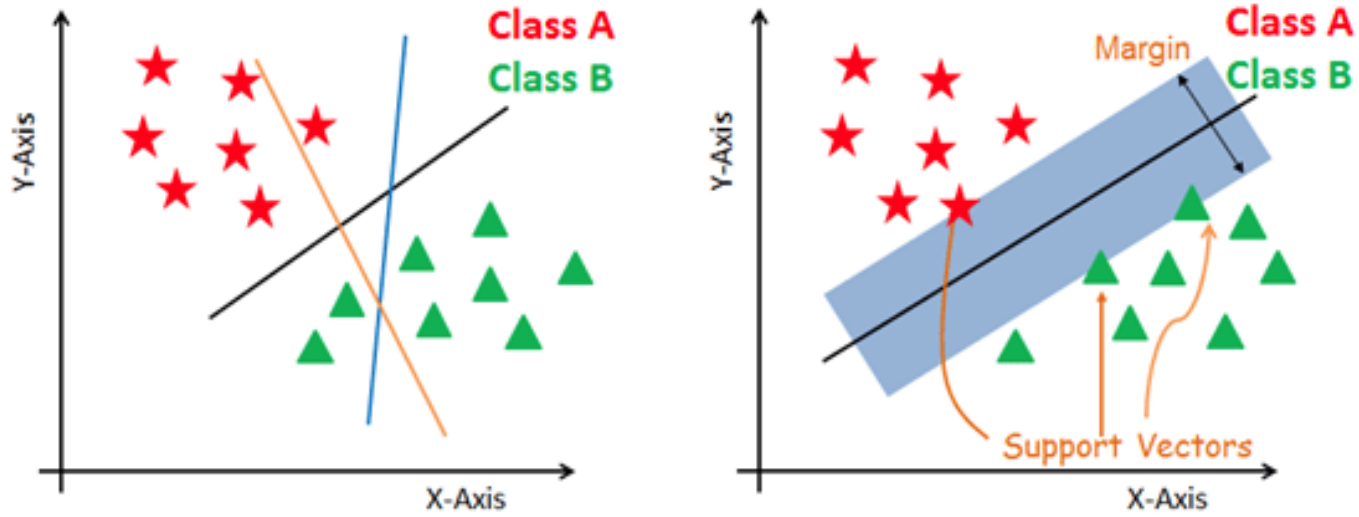
Support Vector Classifier

- Support Vector Classifier is an extension of the Maximal Margin Classifier.
- It is less sensitive to individual data. Since it allows certain data to be misclassified, it's also known as the Soft Margin Classifier.
- It creates a budget under which the misclassification allowance is granted.It allows some points to be misclassified.



How does SVC work?

- The main objective is to segregate the given dataset in the best possible way.
- The distance between the either nearest points is known as the margin.
- The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset.
- SVM searches for the maximum marginal hyperplane as shown below.



Maximum margin intuition

- The rational behind having decision boundaries with large margins is that they tend to have a lower generalization errors.
- Models with smaller margins are more prone to **over-fitting**.
- Considering positive and negative hyperplanes that are parallel to the decision boundary:

$$\left.\begin{aligned} \boldsymbol{\beta}^T \cdot \mathbf{X}^+ &= 1 \\ \boldsymbol{\beta}^T \cdot \mathbf{X}^- &= -1 \end{aligned}\right\} \Rightarrow \boldsymbol{\beta}^T(\mathbf{X}^+ - \mathbf{X}^-) = 2$$

- We can normalize the above equation by the length of the vector $\boldsymbol{\beta}$

$$||\boldsymbol{\beta}|| = \sqrt{\sum_{j=1}^{p} w_j^2}$$

Maximum margin intuition...

- Normalizing both side of the equations yields:

$$\frac{\beta^T(\mathbf{X}^+ - \mathbf{X}^-)}{||\beta||} = \frac{2}{||\beta||}$$

- The left side can be interpreted as the distance between the positive and negative hyperplane, **margin**, that needs to be maximized.
- The objective function of SVM becomes the maximization of the margin by constraint that examples are classified correctly:

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_{i2} x_2 + ... + \beta_p x_{ip}) > 0$$

- In practice, it is easier to minimize the reciprocal term, $\frac{1}{2}||\beta^2||$ which can be solved by quadratic programming.

Learning Model Parameters

- The optimization problem of SVC is commonly represented in the following form:

$$\begin{aligned} & \min \frac{1}{2}||\boldsymbol{\beta}||^2 \\ \textit{subjcet to} \quad & y_i(\beta_0 + \beta_1 x_{i1} + \beta_{i2} x_2 + ... + \beta_p x_{ip}) \geq 1 \\ & \forall i \in 1, 2, ..., n \end{aligned}$$

- Above equation is a constraint optimization with linear inequalities.
- The objective function is convex and quadratic with respect to $\boldsymbol{\beta}$ known as quadratic programming problem (QPP).

Lagrangian primal problem

- Lets rewrite the objective function in a form that takes into account the constraints imposed on its solution.

$$L_p = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{n} \lambda_i (y_i \cdot \beta^T \mathbf{x}_i - 1)$$

  where the $\lambda_i \geq 0$ corresponds to the constraints and are called Lagrange multipliers.

- To minimize the Lagrangian:

$$\frac{\partial L_p}{\partial \beta} = 0$$

In class example

- Find the optimum hyperplane that classifies the following dataset into +1 and -1. The training dataset is:

$$x_1 = [1, 1]^T, class\# - 1$$
$$x_2 = [2, 1]^T, class\# - 1$$
$$x_3 = [1, 2]^T, class\# - 1$$
$$x_4 = [3, 3]^T, class\# + 1$$

- Test set:

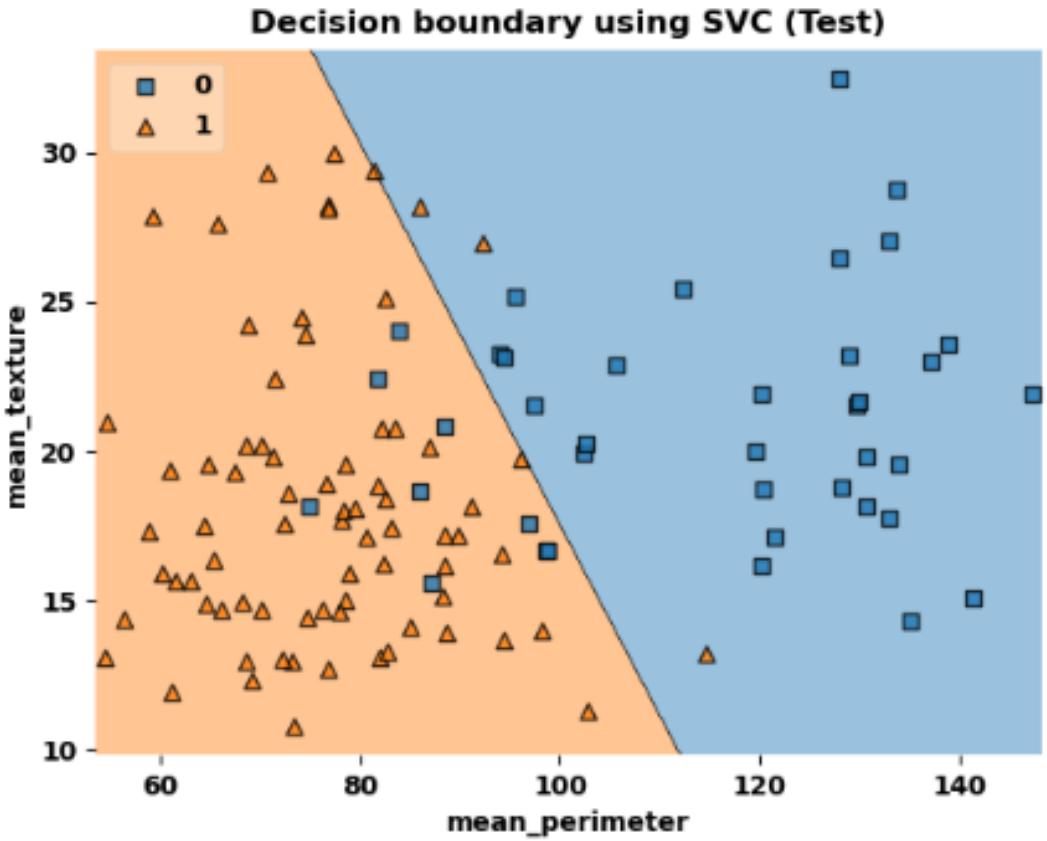
$$x_5 = [4, 4]^T, class\# + 1$$
$$x_6 = [0, 0]^T, class\# - 1$$
$$x_7 = [1.5, 1.5]^T, class\# - 1$$

Breast Cancer classification using linear SVC

- Importing necessary python libraries
- Importing the dataset Breast cancer dataset from scikit-learn. The dataset consists of data related to breast cancer patients and their diagnosis **malignant** or **benign**.
- Separating the features and target variables
- Splitting the dataset into training and test sets, 80-20%
- Fitting the model to the training set. Using SVC() class from scikit-learn.
- Predicting the test results
- Evaluating the model.
- Plotting the decision boundary
- **Develop the python code for above procedures**

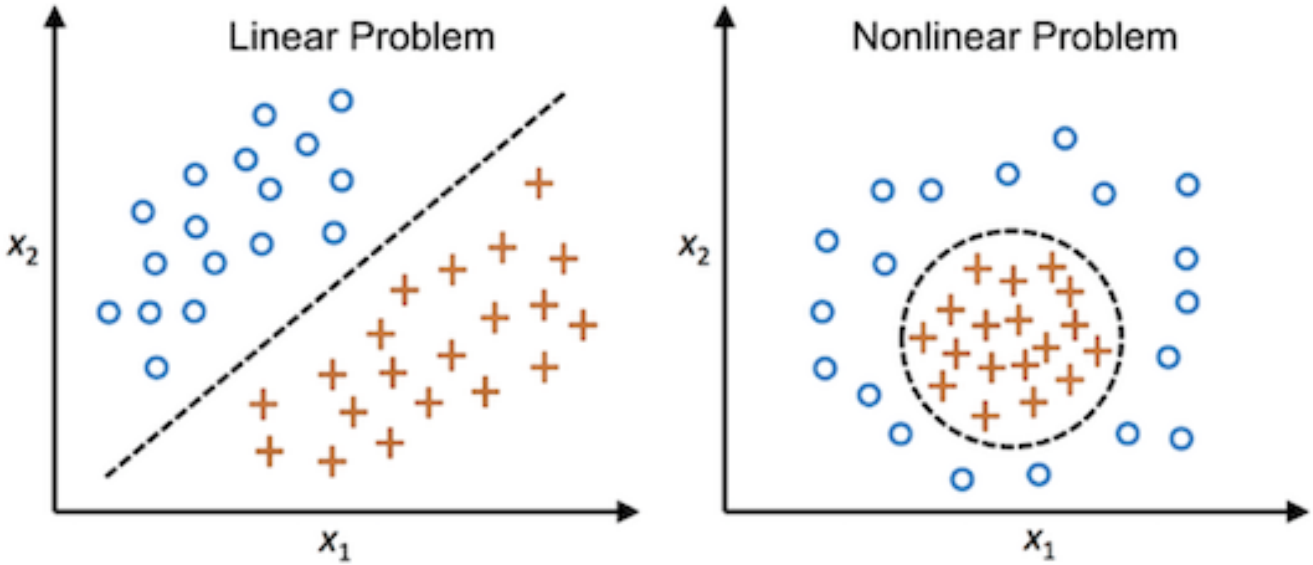
Breast Cancer classification using linear SVC

- The plot shows the distinction between the two classes as classified by the Support Vector Classification algorithm in Python.



Linearly separable and non-linearly separable data

- Maximal Margin Classifier → Hard Margin Classifier.
- Support Vector Classifier → Soft Margin Classifier.
- However, all Maximum-Margin Classifiers and Support Vector Classifiers are restricted to data that can be separated linearly.
- For **nonlinear** classification, Support Vector Machines (SVM) are used.



SVM with Soft Margin

- Soft margin SVM allow misclassification to happen.
- So we'll need to minimize the misclassification error, which means that we'll have to deal with one more constraint.
- Soft margin loss function:

$$\max\{0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\}$$

- The loss of a misclassified point is called a slack variable and is added to the primal problem that we had for hard margin SVM.
- A new regularization parameter λ controls the trade-off between maximizing the margin and minimizing the loss

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + \lambda \sum_{i=1}^{n} \zeta_i$$

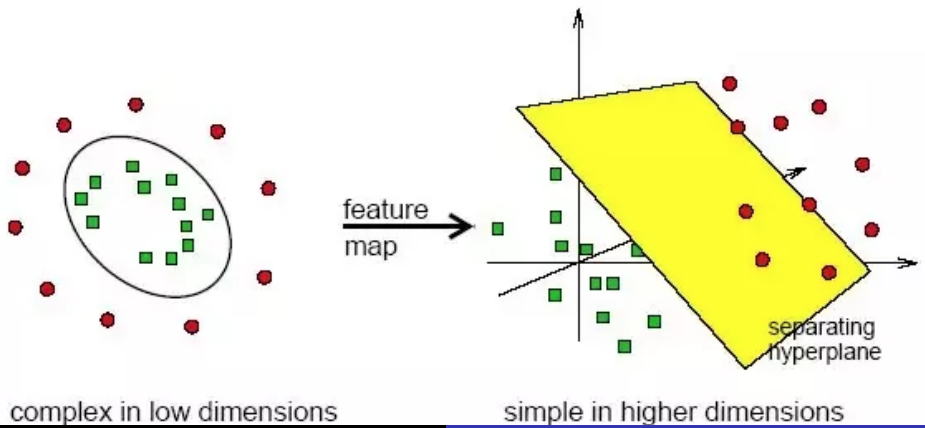
$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \zeta_i \quad \forall i = 1, ..., n, \zeta_i \geq 0$$

Hard Margin vs. Soft Margin

- The difference between a hard margin and a soft margin in SVMs lies in the separability of the data.
- If our data is linearly separable, we go for a hard margin.
- Otherwise soft margin SVM that lest some of the data to be misclassified.
- Sometimes, the data is linearly separable, but the margin is so small that the model becomes prone to overfitting or being too sensitive to outliers.
- Also, in this case, we can opt for a larger margin by using soft margin SVM in order to help the model generalize better.

Support Vector Machines

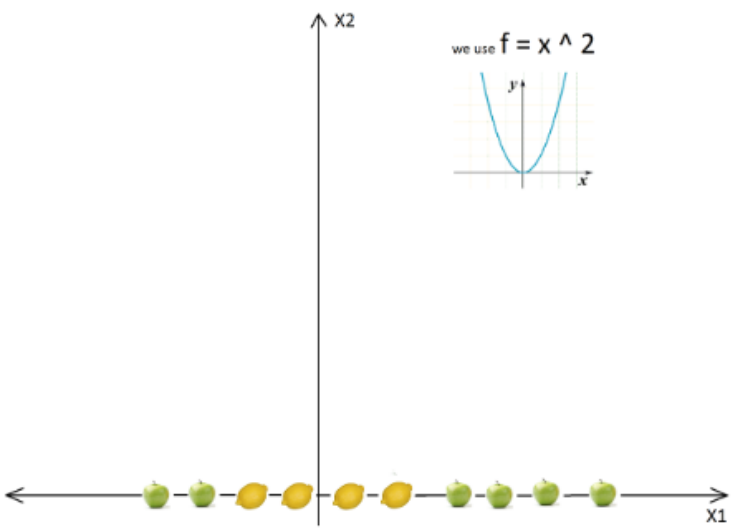
- Support Vector Machines are an extension of Soft Margin Classifier. It can also be used for nonlinear classification by using the kernel.
- SVM performs well in the majority of real-world problem statements. Since in the real world, we will mostly find non-linear separable data, which will necessitate the use of complex classifiers to classify them.
- **Kernel**: It transforms non-linear separable data from lower to higher dimensions to facilitate linear classification. The kernel transforms the data from lower to higher dimensions using mathematical formulas.

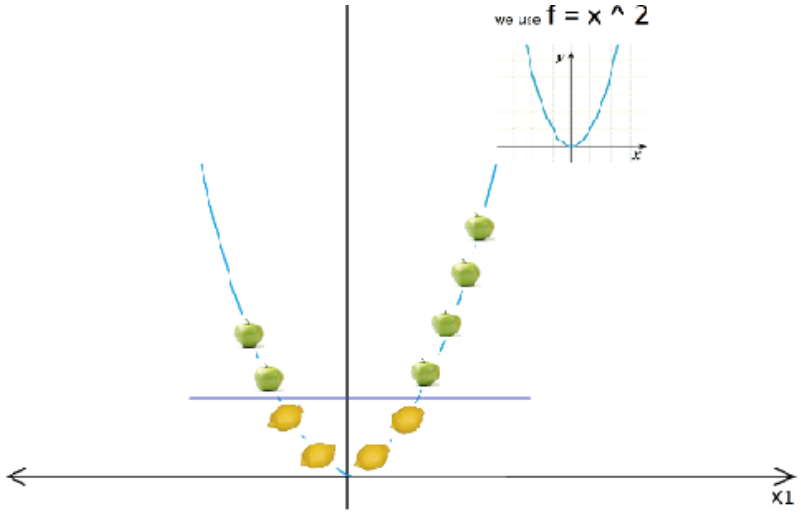


Kernel using simple example

- We want to distinguish apple and lemon on the x-axis. Out model is unable to separate them using a specific point on the x-axis.
- Kernel : Mathematical function
- Kernel will apply the requisite mathematical formula to transform data into higher dimensions, making classification in non-linearly separable data easier.

$$f = x^2$$





SVM Kernels list

- Linear Kernel
- Polynomial Kernel
- Radial Base Function (RBF)
- Sigmoid

Linear Kernel

- Linear Kernel: The linear kernel is equivalnet to Support Vector Classifier.

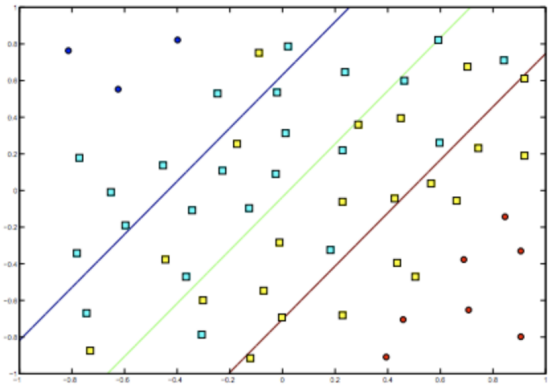
$$K(X_1, X_2) = X_1^T \cdot X_2$$

  where X_1, X_2 represents the data you are trying to classify.
- Linear Kernel is used when the data is Linearly separable.
- Training a SVM with linear Kernel is **faster** than with any other kernel.
- With linear kernel, the only parameter to optimise is λ **Regularization**.
- In python sklearn package can be used accordingly:

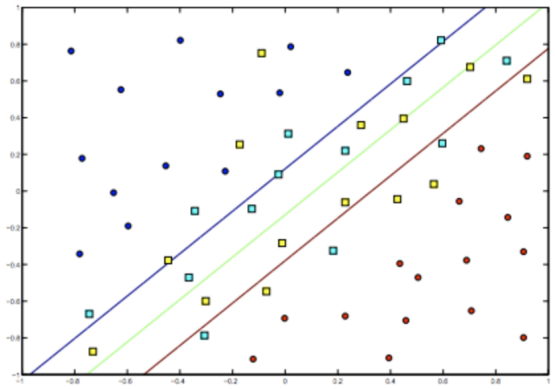
```
from sklearn.svm import SVC
classifier = SVC(kernel ='linear')
classifier.fit(x_train, y_train) # training set in x, y axis
```

SVM with linear kernel- regularization

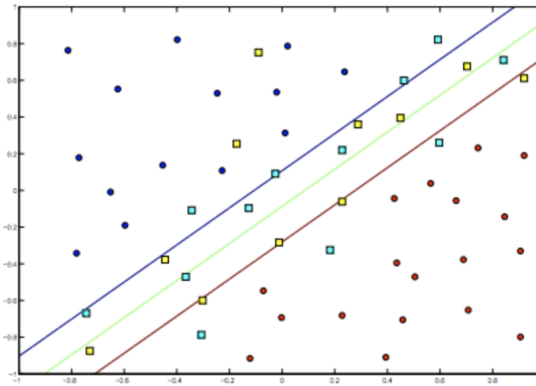
- Regularization parameter serves as a degree of importance to misclassifications. [subject to overfitting]
- When λ grows larger $\Longrightarrow$ less the wrongly classified. When $\lambda \to \infty$ then hard margin.
- When λ tends to 0 (without being 0) the more the miss-classifications are allowed.[subject to under fitting]
- The first figure where λ is lower then more relaxed than the other where data is intended to be fitted more precisely.



($\lambda = 0.1$)



($\lambda = 1$)



($\lambda = 1000$)

Polynomial Kernel

- Polynomial Kernel: The polynomial kernel is more generalized representation of the linear kernel.

$$K(X_1, X_2) = (a + X_1^T \cdot X_2)^b$$

where b is the degree of kernel and a is the constant term.

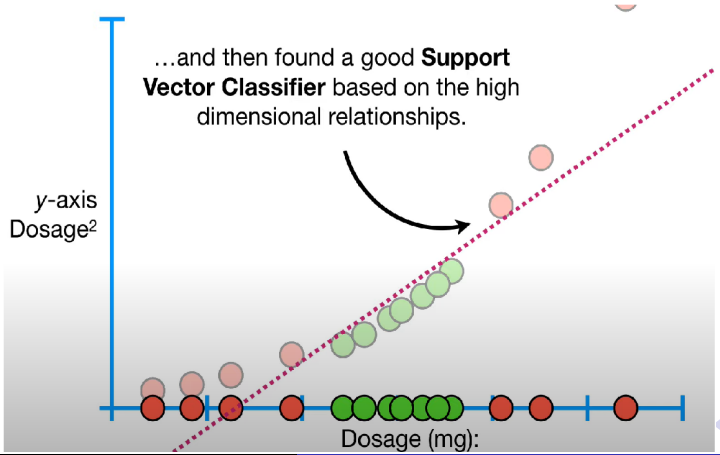
- It is not as preferred as other kernel functions as it is less efficient and accurate.
- In python sklearn package can be used accordingly:

```
from sklearn.svm import SVC
classifier = SVC(kernel ='poly', degree = 4)
classifier.fit(x_train, y_train) # training set in x, y axis
```

Polynomial Kernel-Example

- Let consider 1-D example which is not linearly separable.
- Let $b = 2$ and $a = \frac{1}{2}$, we will have:

$$
\begin{aligned}
(x_1 x_2 + \frac{1}{2})^2 &= (x_1 x_2 + \frac{1}{2})(x_1 x_2 + \frac{1}{2}) \\
&= x_1 x_2 + x_1^2 x_2^2 + \frac{1}{4} \\
&= \left[x_1; x_1^2; \frac{1}{2}\right]^T \cdot \left[x_2; x_2^2; \frac{1}{2}\right]
\end{aligned}
$$



Radial Base Function Kernel

- Gaussian Radial Base Function Kernel is most generalized from of kernelization.
- RBF is most widely used due to similarity to gaussian distribution.
- RBF is a function whose value depends on the distance from the origin or from some point.

$$K(X_1, X_2) = e^{(-\frac{||X_1 - X_2||^2}{2\sigma^2})}$$

where $||.||$ is the Euclidean distance between X_1 and X_2. and σ is the standard deviation and the hyperparameter. It is important to find the right value of σ to decide which points should be considered similar.

- In python sklearn package can be used accordingly:

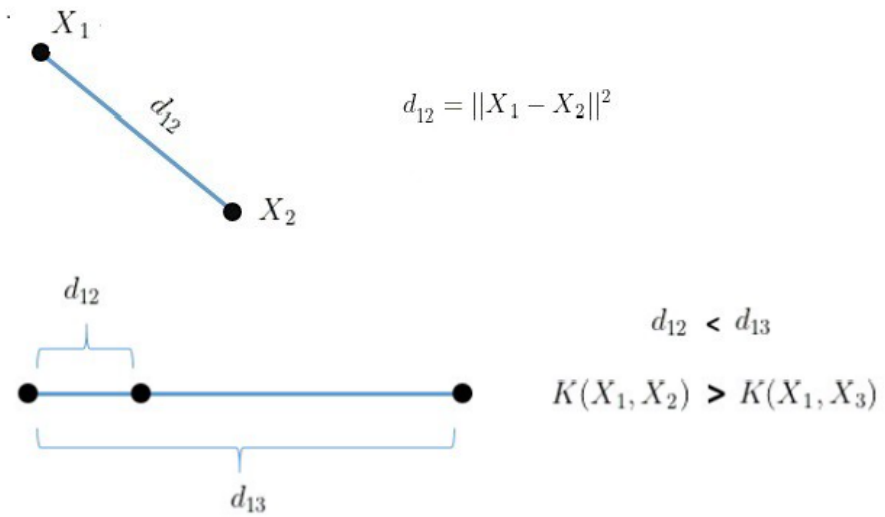
```
from sklearn.svm import SVC
classifier = SVC(kernel ='rbf', random_state = 0)
 # training set in x, y axis
classifier.fit(x_train, y_train)
```

RBF-Example

- Let d_{12} be the distance the two points X_1 and X_2.
- The kernel equation can be re-written as follows:

$$K(X_1, X2) = e^{-\frac{d_{12}}{2\sigma^2}}$$

- The maximum value that RBF kernel can be is 1 and occurs when $d_{12} = 0$ which means the points are similar $(X_1 = X_2)$
- The value that RBF kernel can be close to 0 and occurs when the points are dissimilar and separated by a large distance.

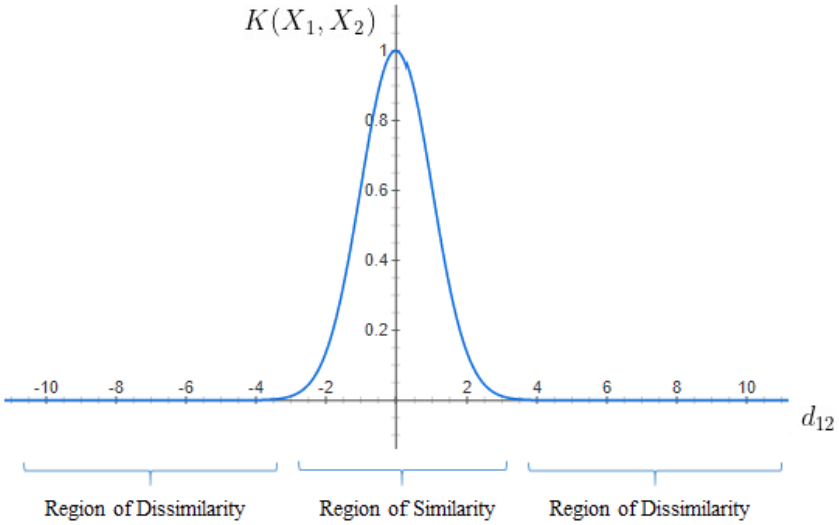


RBF-Choice of σ

- When $\sigma = 1$

$$K(X_1, X2) = e^{(-\frac{||X_1 - X_2||^2}{2})}$$

- As the distance increases, the RBF kernel decreases exponentially and it is 0 for distances greater than 4.
- When $d_{12} = 0$ the similarity is 1.
- If distance is bellow 4, the points can be considered similar and if the distance is greater than 4 then the points are dissimilar.

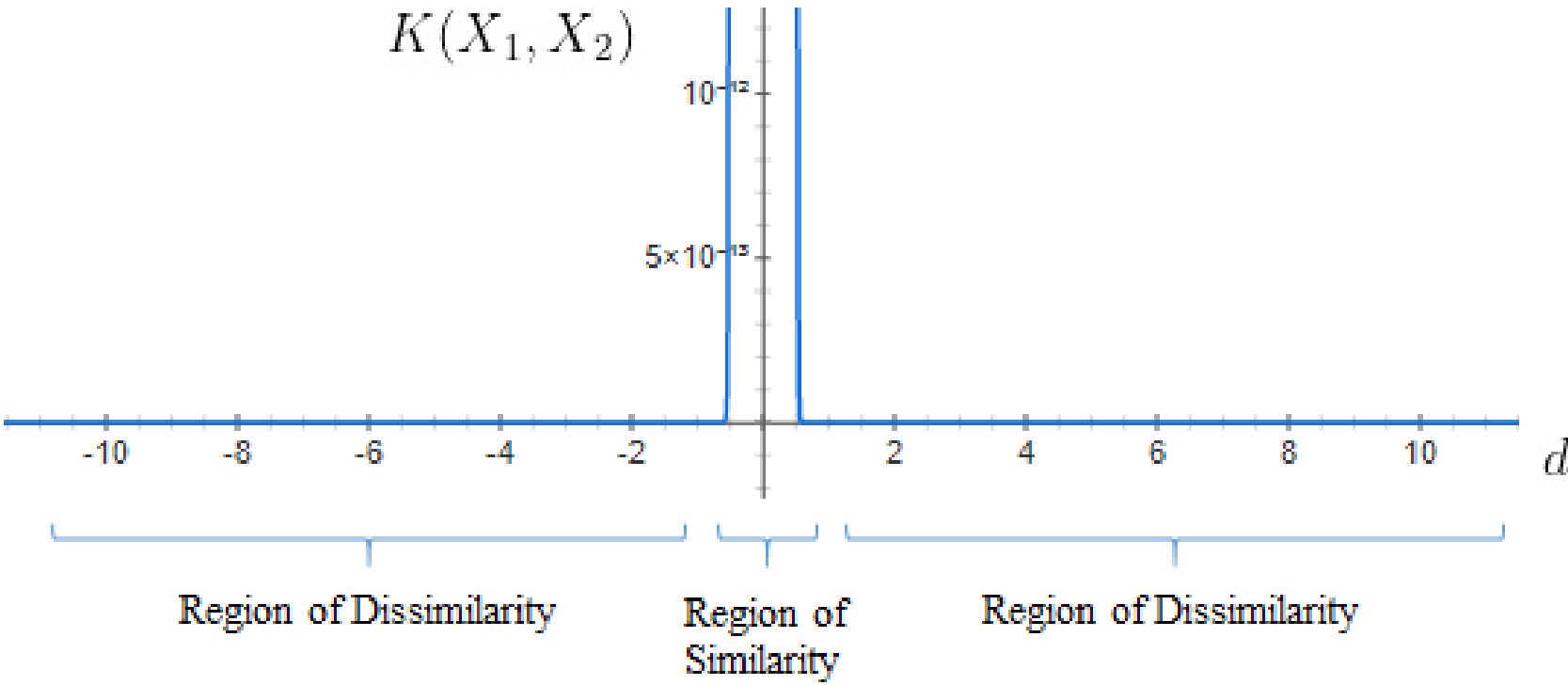


RBF-Choice of σ

- When $\sigma = 0.1$

$$K(X_1, X2) = e^{(-\frac{||X_1 - X_2||^2}{0.02})}$$

- Curve extremely peaked and 0 for distances greater than 0.2.
- Points are considered similar the distance is less than 0.2.



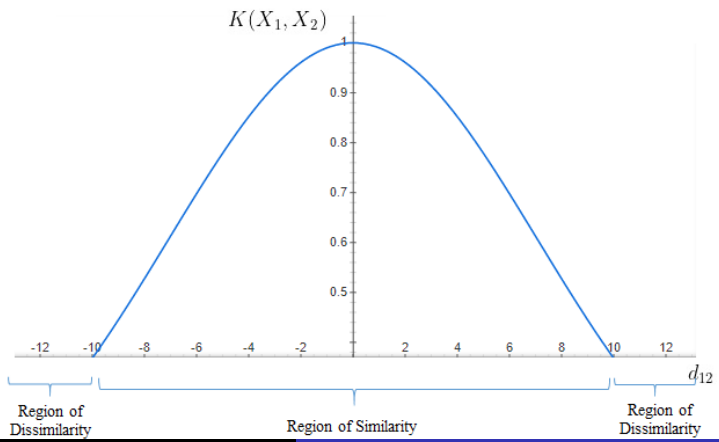
RBF-Choice of σ

- When $\sigma = 10$

$$K(X_1, X2) = e^{(-\frac{||X_1 - X_2||^2}{200})}$$

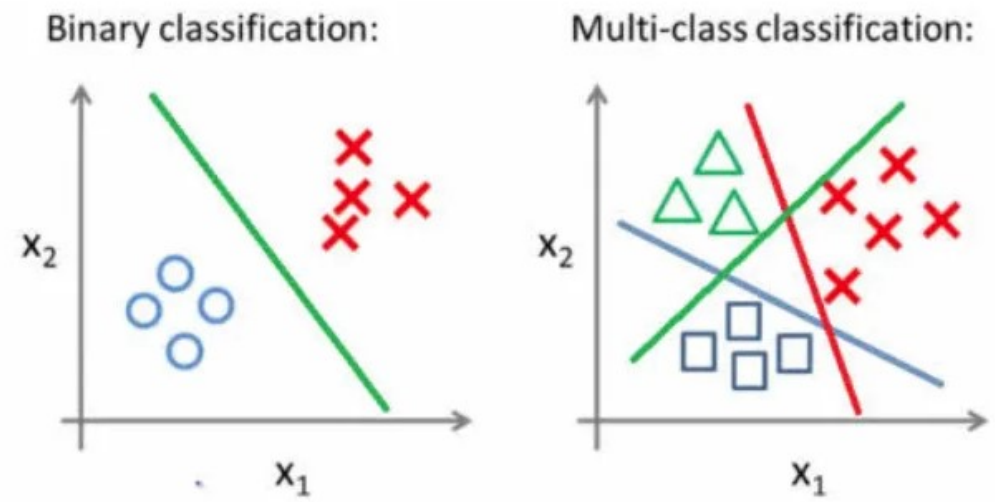
- The width of the curve is large.
- Points are considered similar for distances up to 10 and beyond 10 units they are dissimilar.
- The RBF Kernel Support Vector Machines is implemented in the scikit-learn library and σ can be tuned using γ

$$\gamma \propto \frac{1}{\sigma}$$



Binary classification versus multi-class classification

- Algorithms such as Perceptron, Logistic Regression and Support Vector Machines were designed for binary classification.
- Binary classification when problem have two class labels.
- Multi-class classification when we have more than two class instances.

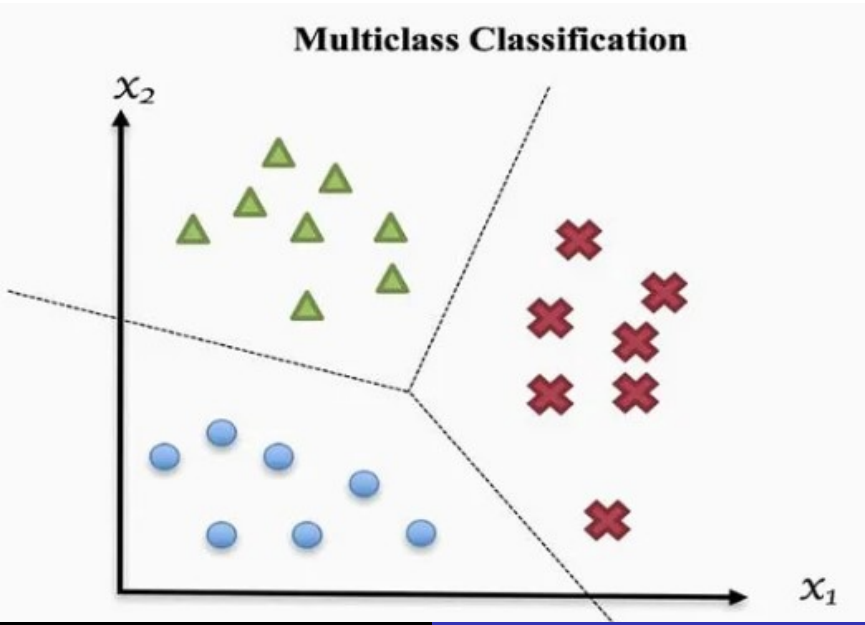


Multi-class classification

- Multi-class classification allows to categorize the test data into multiple class labels in trained data as a model prediction.
- There are mainly two types of multi-class classification techniques:

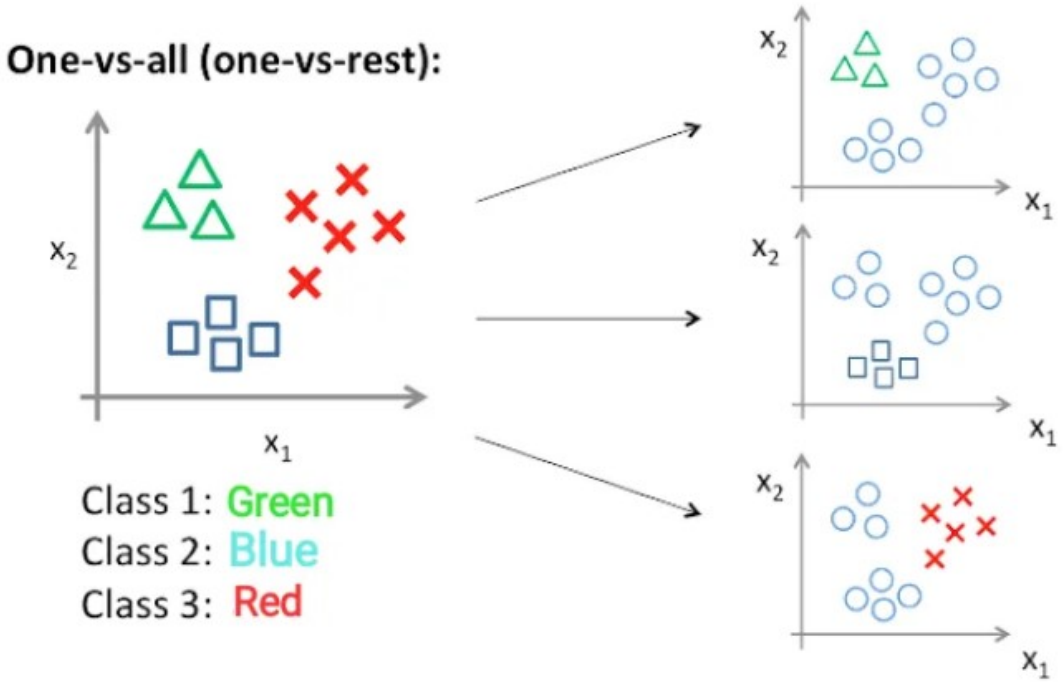
One-vs-all

One-vs-one



One-vs-all(one-vs-rest)

- In one-vs-rest classification, for the N-class instances dataset, we have to generate N-binary classifier models.
- The number of class labels present in the dataset and the number of generated binary classifiers must be the same.
- Generate the same number of classifiers as the class labels in the dataset.



One-vs-all(one-vs-rest)

- **Classifier 1**: [green] vs [red, blue]
- **Classifier 2**: [blue] vs [red, green]
- **Classifier 3**: [red] vs [green, blue]
- Need to create training dataset to train these three classifiers:

Features			Classes
x1	x2	x3	G
x4	x5	x6	B
x7	x8	x9	R
x10	x11	x12	G
x13	x14	x15	B
x16	x17	x18	R

Class 1 :- Green

Class 2 :- Blue

Class 3 :- Red

One-vs-all(one-vs-rest)

- Replace particular class by +1 and put -1 for the remaining classes

Training Dataset 1
Class :- Green

Features			Green
x1	x2	x3	**+1**
x4	x5	x6	-1
x7	x8	x9	-1
x10	x11	x12	**+1**
x13	x14	x15	-1
x16	x17	x18	-1

Training Dataset 2
Class :- Blue

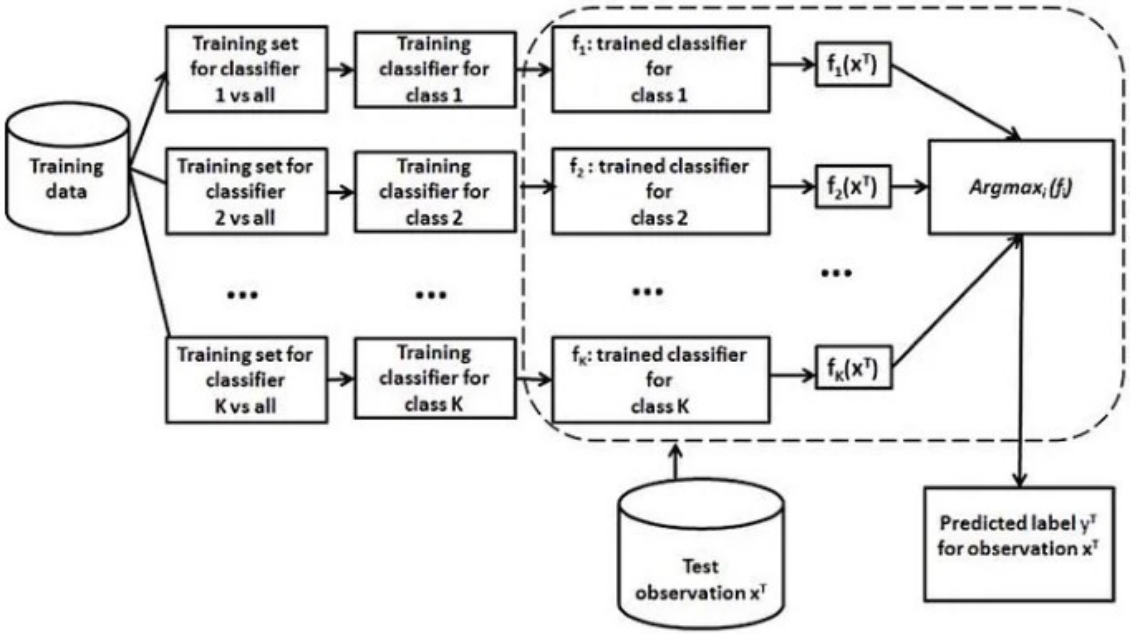
Features			Blue
x1	x2	x3	-1
x4	x5	x6	**+1**
x7	x8	x9	-1
x10	x11	x12	-1
x13	x14	x15	**+1**
x16	x17	x18	-1

Training Dataset 3
Class :- Red

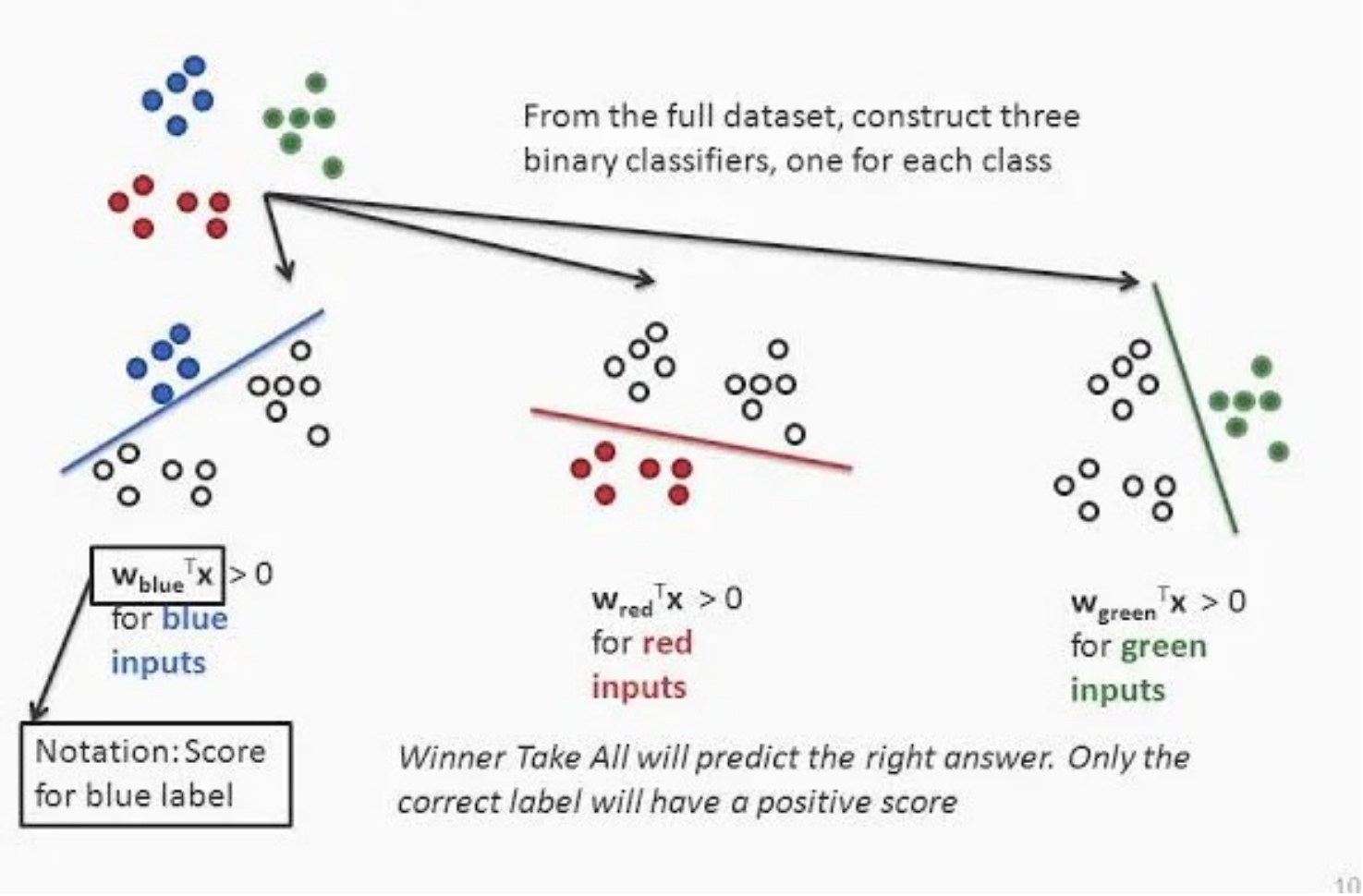
Features			Red
x1	x2	x3	-1
x4	x5	x6	-1
x7	x8	x9	**+1**
x10	x11	x12	-1
x13	x14	x15	-1
x16	x17	x18	**+1**

One-vs-all(one-vs-rest)

- Train k training model. Pass input the test input to all trained model.
- If there is any possibility that the test data belongs to a particular class, then the classifier output positive number close $+1$, or negative number close to -1 otherwise.
- By analyzing the probability scores, we predict the result as the class index having a maximum probability score.

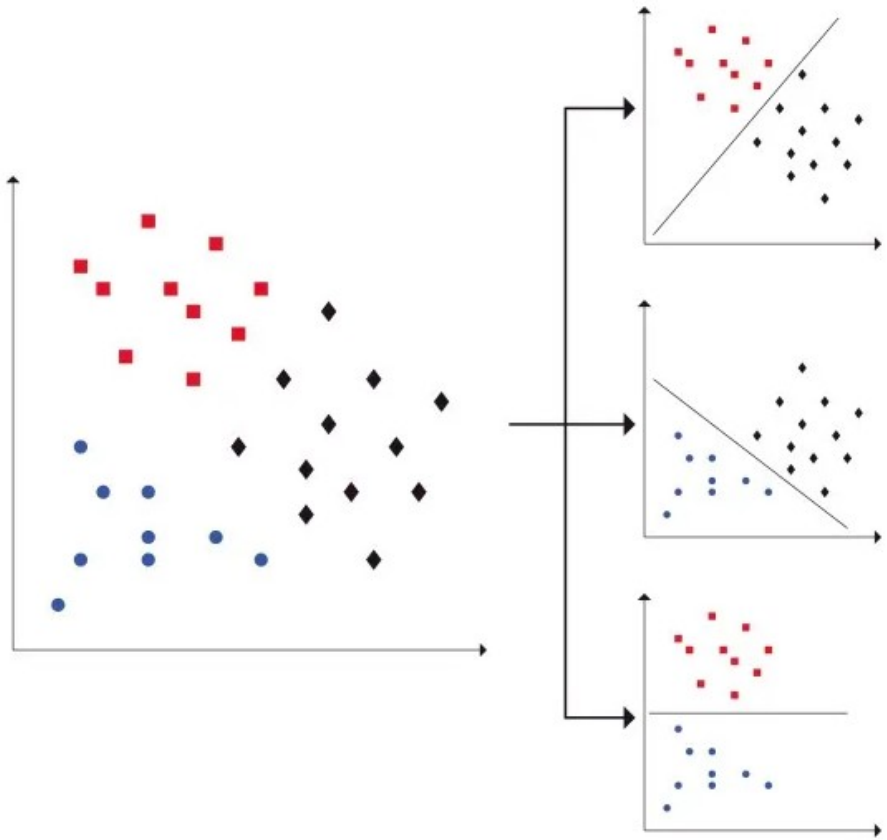


Visualizing one-vs-all



One-vs-one (OvO)

- In One-vs-One classification, for the N-class instances dataset, we have to generate the $\frac{N(N-1)}{2}$ binary classifier models.
- Split the primary dataset into one dataset for each class opposite to every other class.

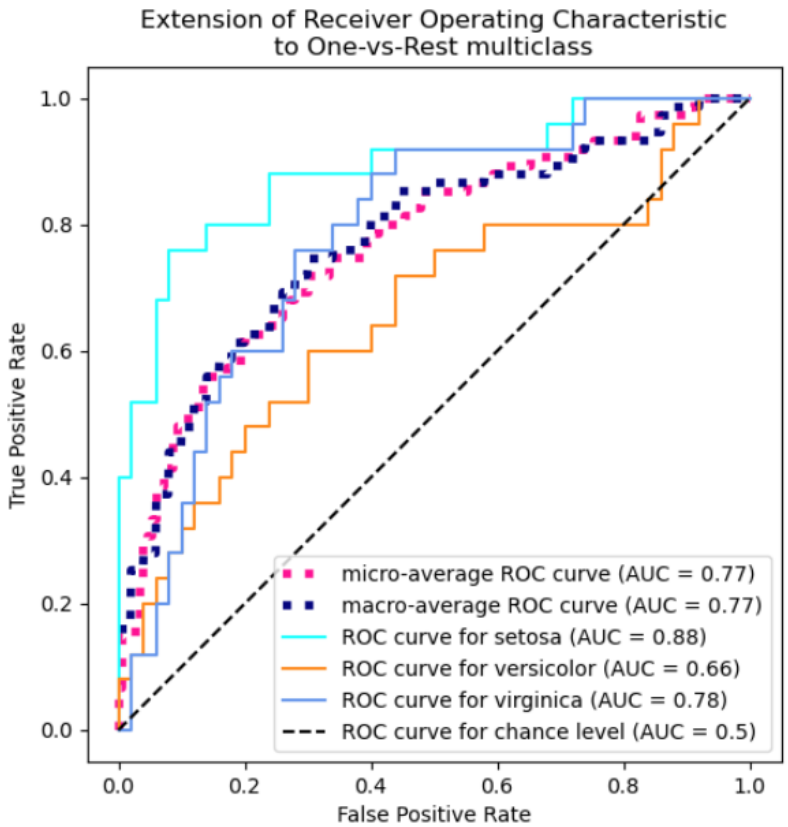


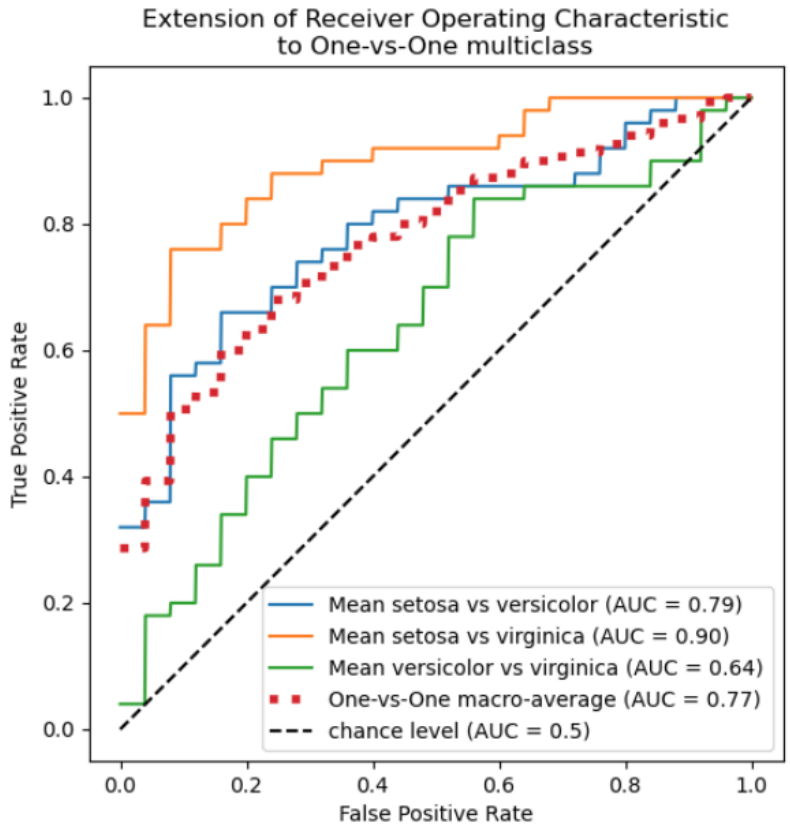
One-vs-one (OvO)

- Consider an example with three classes: Green, Blue and Red.
- We divide this problem into 3*(3-1)/2=3 binary classifier problems:
  - Classifier 1: Green vs Blue
  - Classifier 2: Green vs Red
  - Classifier 3: Blue vs Red
- Each binary classifier predicts one class label.
- When we input the test data to the classifier, then the model with the majority counts is concluded as a result.
- If the binary classification models predict a numerical class such as probability, then the argmax of the sum of the scores is predicted as the class label.
- The OvO strategy is recommended if the user is mainly interested in correctly identifying a particular class or subset of classes at the expense of computational cost for large number of classes.

One-vs-one (OvO) Iris Example

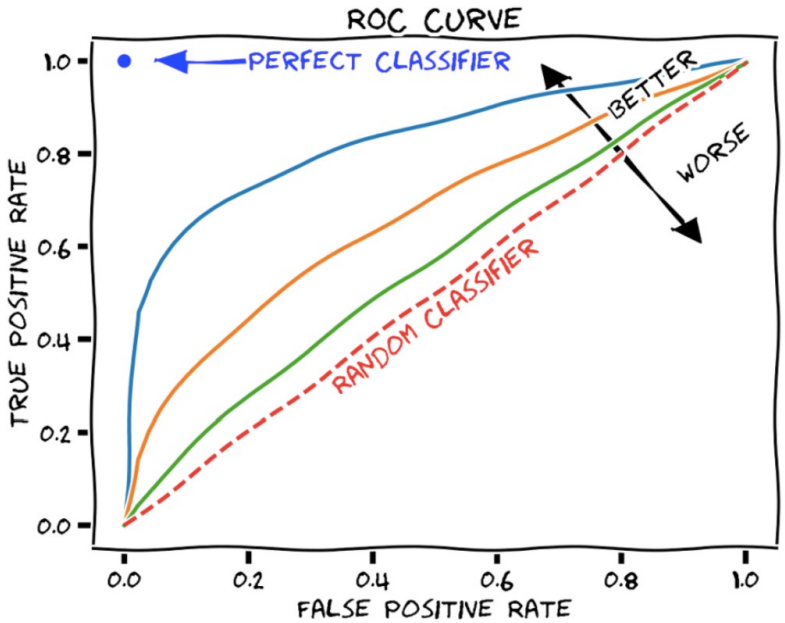
- We confirm that the classes "versicolor" and "virginica" are not well identified by a linear classifier. Notice that the "virginica"-vs-the-rest ROC-AUC score (0.77) is between the OvO ROC-AUC scores for "versicolor" vs "virginica" (0.64) and "setosa" vs "virginica" (0.90).





ROC & AUC/ AUROC

- Receiver operating characteristic (ROC) graphs are used for selecting the most appropriate classification models based on their performance with respect to the false positive rate (FPR) and true positive rate (TPR).
- These metrics are computed by shifting the decision threshold of the classifier.
- ROC curve is used for probabilistic models which predict the probabilities of the class



Performance Multi-class classification

Micro-average precision

- In micro-average method, you sum up the individual true positives, false positives and false negatives.

$$PrecisionMicroAvg = \frac{TP_1 + TP_2 + ... + TP_n}{TP_1 + TP_2 + ... + TP_n + FP_1 + FP_2 + ... + FP_n}$$

Macro-average precision

- In macro-average method, is the arithmetic mean of all precision values for the different classes.

$$PrecisionMacroAvg = \frac{P_1 + P_2 + ... + P_n}{n}$$

Performance Multi-class classification

Micro-average recall

- In micro-average method, is sum of true positives divided by actual positives.

$$RecallMicroAvg = \frac{TP_1 + TP_2 + ... + TP_n}{TP_1 + TP_2 + ... + TP_n + FN_1 + FN_2 + ... + FN_n}$$

Macro-average recall

- In macro-average method, is the arithmetic mean of all recall scores for the different classes.

$$RecallMacroAvg = \frac{R_1 + R_2 + ... + R_n}{n}$$

  - Macro averaging in case of equal weight for each instance.
  - Micro averaging in case of unbalanced class labels.

Example

- Let's suppose we have a multi-class classification system with three unbalanced classes and the following numbers:

Class	TP	FP	FN	Precision	Recall
X	15	11	2	0.57	0.88
Y	10	90	7	0.1	0.58
Z	5	2	1	0.71	0.83

$$PrecisionMicroAvg = \frac{15 + 10 + 5}{15 + 10 + 5 + 11 + 90 + 2} = 0.22$$

$$PrecisionMacroAvg = \frac{0.57 + 0.1 + 0.71}{3} = 0.46$$

$$RecallMicroAvg = \frac{15 + 10 + 5}{15 + 10 + 5 + 2 + 7 + 1} = 0.75$$

$$RecallMacroAvg = \frac{0.88 + 0.58 + 0.83}{3} = 0.76$$

F-score and AUC multi-class classification

F-score

- The Macro-average F-Score will be simply the harmonic mean of the recall and precision.

AUC

- For the AUC score , we can calculate the AUC for specific class and average score of classifier.