Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

JUNAID GIRKAR
60004190057
TE COMPS A4

# EXPERIMENT - 8

**AIM**: Implement Radial Basis function.

**THEORY**:

Radial Basis Kernel is a kernel function that is used in machine learning to find a non-linear classifier or regression line.

**What is Kernel Function**?
Kernel Function is used to transform n-dimensional input to m-dimensional input, where m is much higher than n then find the dot product in higher dimensional efficiently. The main idea to use the kernel is: A linear classifier or regression curve in higher dimensions becomes a Non-linear classifier or regression curve in lower dimensions.

**Mathematical Definition of Radial Basis Kernel:**

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

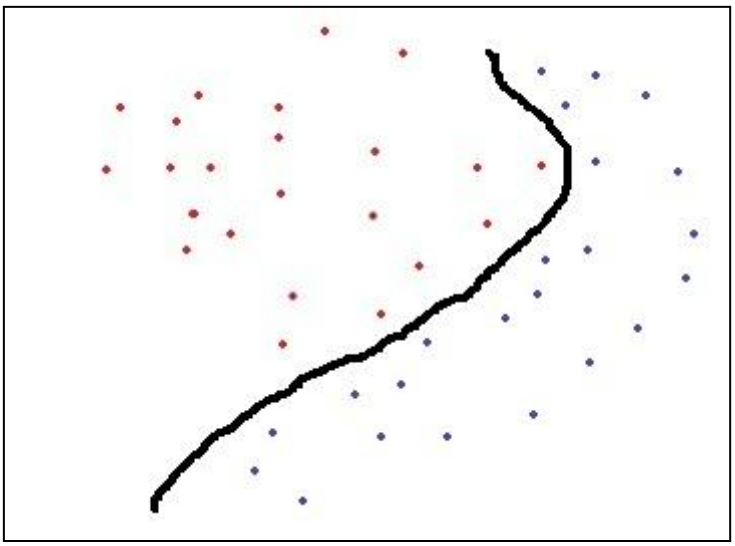
**Radial Basis Kernel**
where x, x' are vector point in any fixed dimensional space.
But if we expand the above exponential expression, It will go upto infinite power of x and x', as expansion of ex contains infinite terms upto infinite power of x hence it involves terms upto infinite powers in infinite dimension.
If we apply any of the algorithms like perceptron Algorithm or linear regression on this kernel, actually we would be applying our algorithm to new infinite-dimensional datapoint we have created. Hence it will give a hyperplane in infinite dimensions, which will give a very strong non-linear classifier or regression curve after returning to our original dimensions.

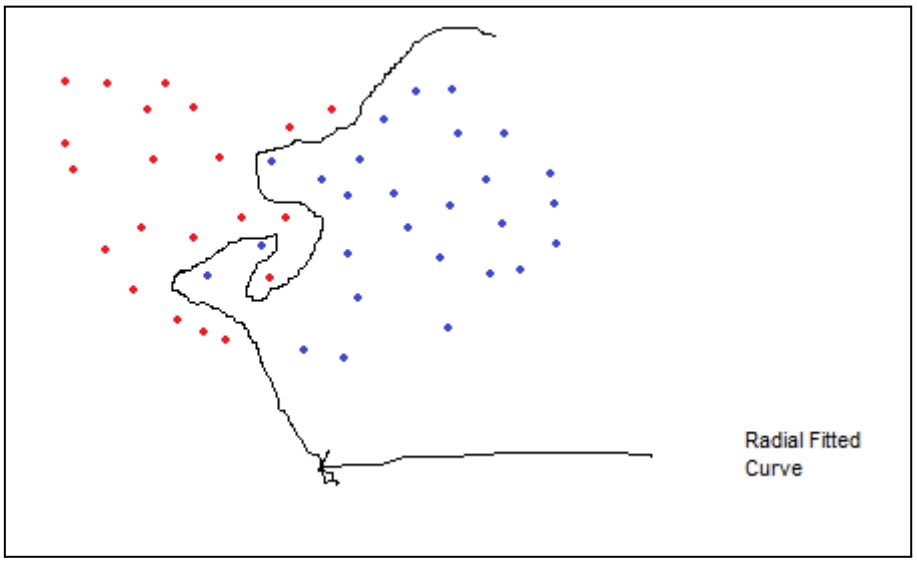
$$a_1 X^{inf} + a_2 X^{inf-1} + a_3 X^{inf-2} + \ldots + a_{inf} X + C$$

polynomial of infinite power
So, Although we are applying linear classifier/regression it will give a non-linear classifier or regression line, that will be a polynomial of infinite power. And being a polynomial of infinite power, Radial Basis kernel is a very powerful kernel, which can give a curve fitting any complex dataset.

**Why is the Radial Basis Kernel so powerful**?
The main motive of the kernel is to do calculations in any d-dimensional space where d > 1, so that we can get a quadratic, cubic or any polynomial equation of large degree for our classification/regression line. Since Radial basis kernel uses exponent and as we know the expansion of e^x gives a polynomial equation of infinite power, so using this kernel, we make our regression/classification line infinitely powerful too.

Some Complex Dataset Fitted Using RBF Kernel easily:






Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

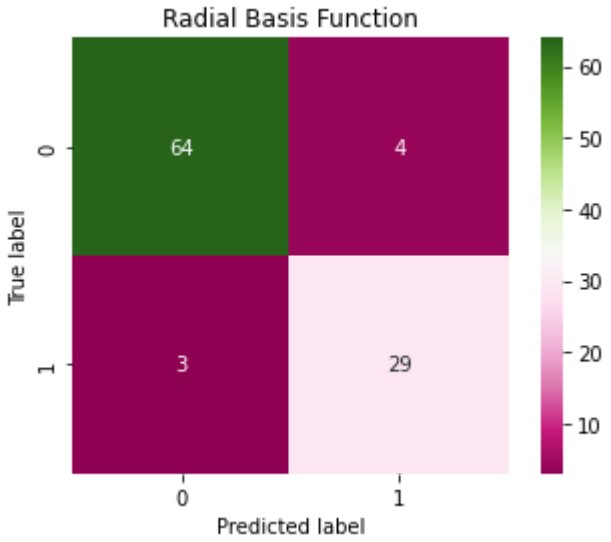
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
confusion_matrix,
accuracy_score,
precision_score,
recall_score,
)
from sklearn.svm import SVC
from matplotlib.colors import ListedColormap
datasets = pd.read_csv("Social_Network_Ads.csv")
print('Radial Basis Function')
print("Dataset used : Social_Networks_Ads.csv")
print("First 5 rows of dataset : ")
print(datasets.head())
x1 = datasets[["Age", "EstimatedSalary"]]
y1 = datasets[["Purchased"]]
X = datasets.iloc[:, [2, 3]].values
Y = datasets.iloc[:, 4].values
print ("Features :", x1.columns.values.tolist())
print ("Labels :", y1.columns.values.tolist())
(X_Train, X_Test, Y_Train, Y_Test) = train_test_split(
X, Y, test_size=0.25, random_state=0
)
sc_X = StandardScaler()
X_Train = sc_X.fit_transform(X_Train)
X_Test = sc_X.transform(X_Test)
classifier1 = SVC(kernel="rbf", random_state=0)
classifier1.fit(X_Train, Y_Train)
Y_Pred1 = classifier1.predict(X_Test)
cm1 = confusion_matrix(Y_Test, Y_Pred1)
print("Radial Basis Function : ")
sns.heatmap(cm1, square=True, annot=True, cmap="PiYG")
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.title("Radial Basis Function")
plt.show()
print ("Radial Basis Function Accuracy :", accuracy_score(Y_Test, Y_Pred1))
print ("Radial Basis Function Precision :", precision_score(Y_Test, Y_Pred1))
print ("Radial Basis Fynction Recall :", recall_score(Y_Test, Y_Pred1))
(X_Set, Y_Set) = (X_Train, Y_Train)
(X1, X2) = np.meshgrid(
np.arange(start=X_Set[:, 0].min() - 1, stop=X_Set[:, 0].max() + 1, step=0.01),
```

```
np.arange(start=X_Set[:, 1].min() - 1, stop=X_Set[:, 1].max() + 1, step=0.01),
)
plt.contourf(
X1,
X2,
classifier1.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
alpha=0.75,
cmap=ListedColormap(("red", "orange")),
)
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for (i, j) in enumerate(np.unique(Y_Set)):
    plt.scatter(
    X_Set[Y_Set == j, 0],
    X_Set[Y_Set == j, 1],
    c=ListedColormap(("red", "orange"))(i),
    label=j,
    )
plt.title("Radial Basis Function (on Training set)")
plt.xlabel("Age")
plt.ylabel("Estimated Salary")
plt.legend()
plt.show()
(X_Set, Y_Set) = (X_Test, Y_Test)
(X1, X2) = np.meshgrid(
np.arange(start=X_Set[:, 0].min() - 1, stop=X_Set[:, 0].max() + 1, step=0.01),
np.arange(start=X_Set[:, 1].min() - 1, stop=X_Set[:, 1].max() + 1, step=0.01),
)
plt.contourf(
X1,
X2,
classifier1.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
alpha=0.75,
cmap=ListedColormap(("green", "blue")),
)
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for (i, j) in enumerate(np.unique(Y_Set)):
    plt.scatter(
    X_Set[Y_Set == j, 0],
    X_Set[Y_Set == j, 1],
    c=ListedColormap(("green", "blue"))(i),
    label=j,
```

```
    )
plt.title("Radial Basis Function (on Test set)")
plt.xlabel("Age")
plt.ylabel("Estimated Salary")
plt.legend()
plt.show()
```

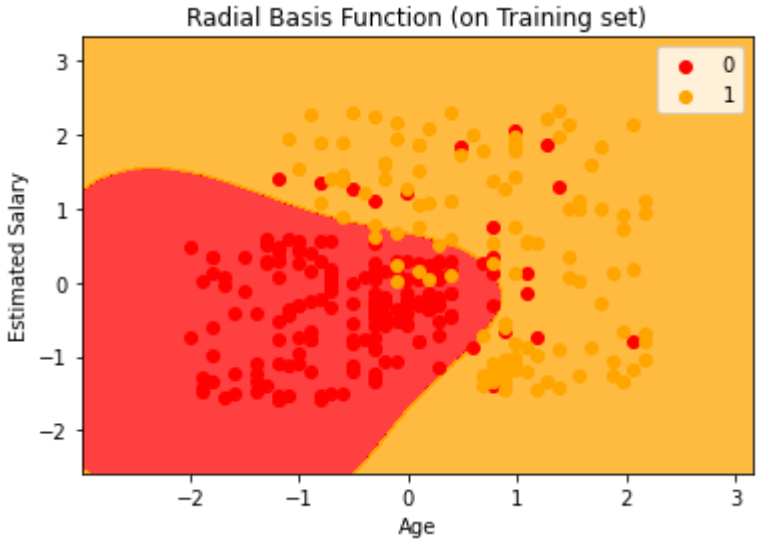
Output:

```
Radial Basis Function
Dataset used : Social_Networks_Ads.csv
First 5 rows of dataset :
   User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0
Features : ['Age', 'EstimatedSalary']
Labels : ['Purchased']
Radial Basis Function :
```

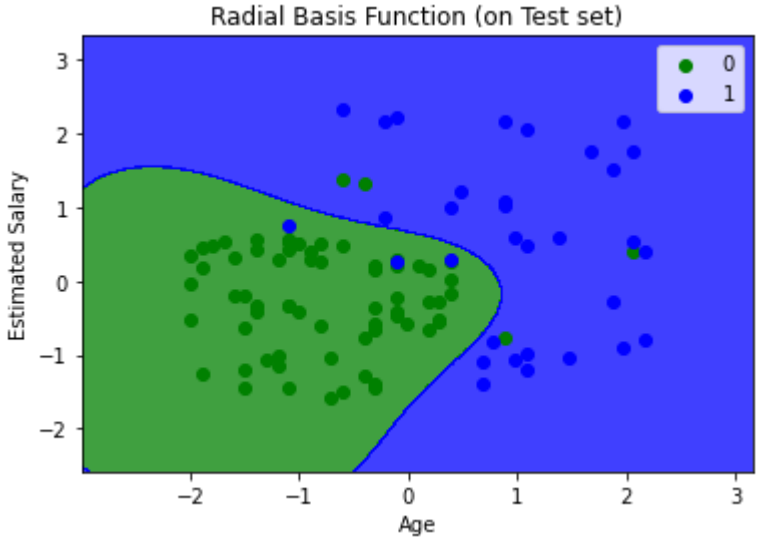



```
Radial Basis Function Accuracy : 0.93
Radial Basis Function Precision : 0.8787878787878788
Radial Basis Fynction Recall : 0.90625
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided
as value-mapping will have precedence in case its length matches with *x* & *y*.  Please
```

use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.




*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.




Conclusion: We learnt about Radial Basis kernel and implemented it in python using the Social_Network_Ads.csv dataset.