ASSIGNMENT – 09

TOPIC: SUPPORT VECTOR MACHINE (SVM)

Q.1. Write notes for sklearn.svm.SVC class focusing more on kernel values – 'linear', 'rbf', 'poly', etc.

Q.2. Write notes for matplotlib library focus on different plots. Ex. 'contourf'.

Q.3. Write notes about numpy – 'meshgrid', 'ravel'.

Ans. 1) sklearn.svm.SVC:

class `sklearn.svm.`**SVC**(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*)

SVM Kernels

The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space. In other words, you can say that it converts non-separable problem to separable problems by adding more dimension to it. It is most useful in non-linear separation problem. Kernel trick helps you to build a more accurate classifier.

- **Linear Kernel:** A linear kernel can be used as normal dot product any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

K(x, xi) = sum(x * xi)

- **Polynomial Kernel:** A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.

K(x, xi) = 1 + sum(x * xi)^d

Where d is the degree of the polynomial. d=1 is similar to the linear transformation. The degree needs to be manually specified in the learning algorithm.

- **Radial Basis Function Kernel:** The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite dimensional space.

K(x, xi) = exp(-gamma * sum(x-xi^2))

Here gamma is a parameter, which ranges from 0 to 1. A higher value of gamma will perfectly fit the training dataset, which causes over-fitting. Gamma=0.1 is considered to be a good default value. The value of gamma needs to be manually specified in the learning algorithm.

Other basic parameters in sklearn.svm.SVC class:

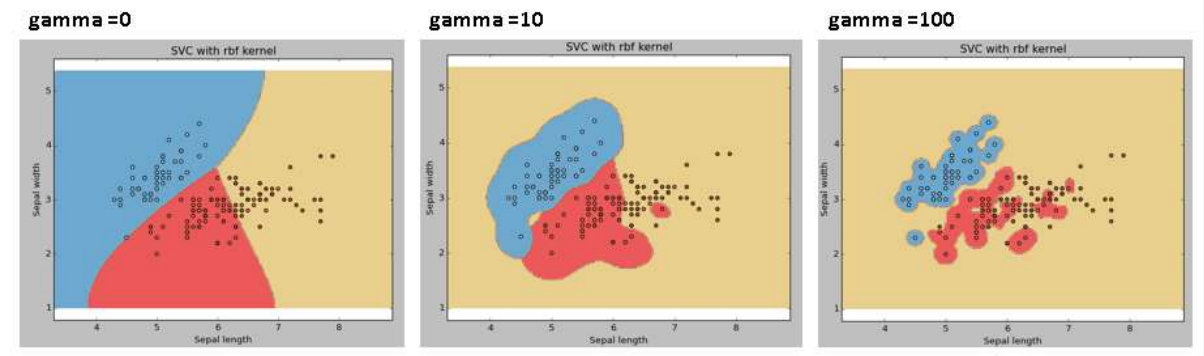
```
svc = svm.SVC(kernel='rbf', C=1,gamma=0)
```

- **Gamma:** Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. Higher the value of gamma, will try to exact fit the as per training data set i.e. generalization error and cause over-fitting problem.
- **C:** Penalty parameter C of the error term. It also controls the trade-off between smooth decision boundaries and classifying the training points correctly.

Example: Comparison on different values of gamma and C using iris dataset.

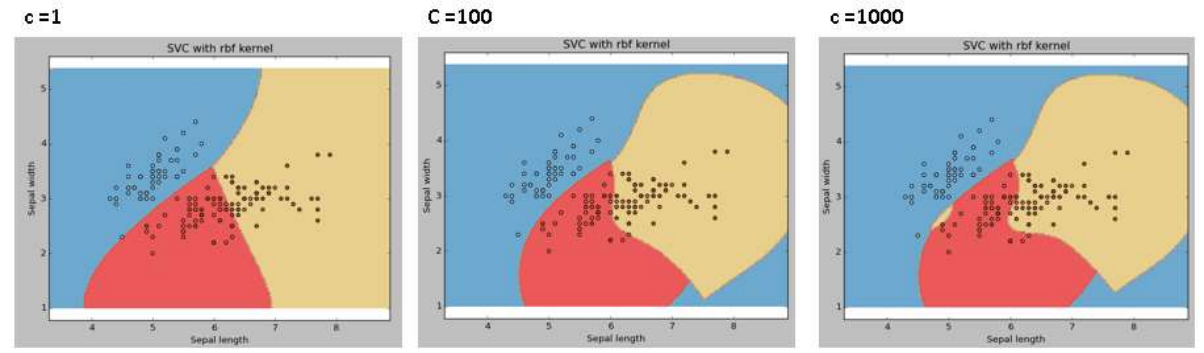
```
1. Difference when using different values of gamma in the above code snippet.

Kernel:  rbf                    rbf                    rbf
gamma:   0                      10                     100
C:       1                      1                      1
```

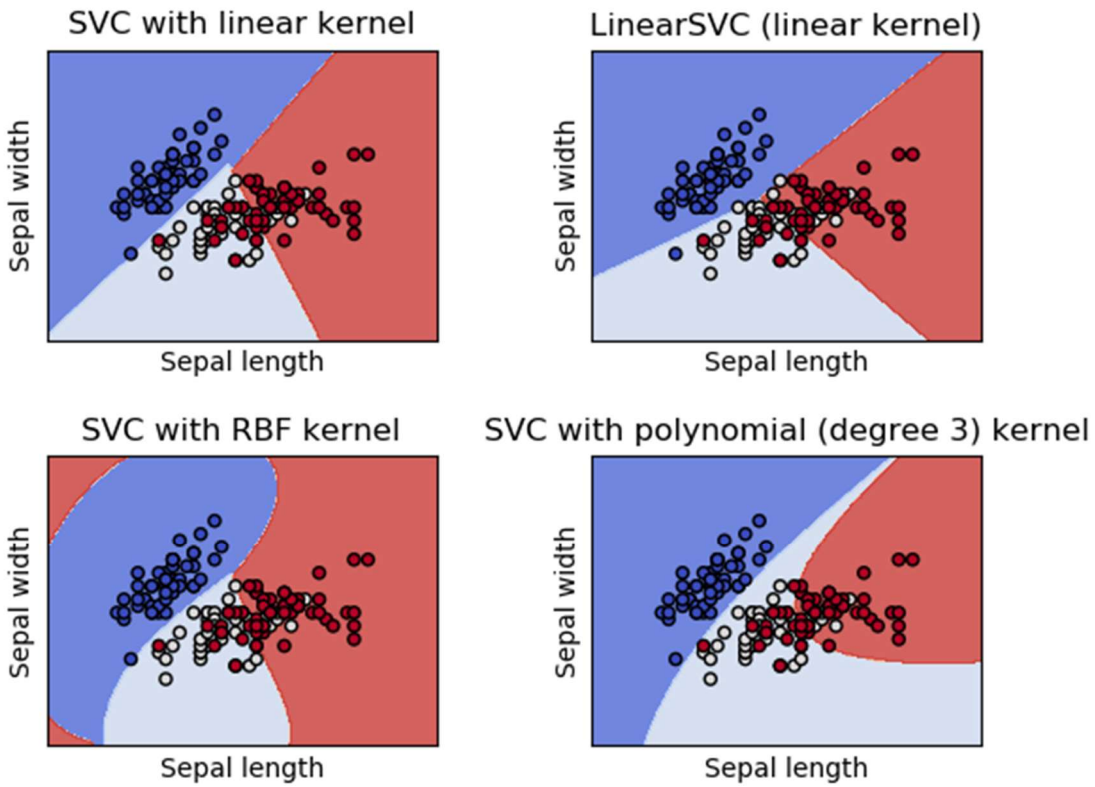



```
2. Difference when using different values of C in the above code snippet.

Kernel:  rbf                    rbf                    rbf
gamma:   0                      10                     100
C:       1                      100                    1000
```



Comparison of different linear SVM classifiers on a 2D projection of the iris dataset using different kernels (source sklearn.org):



Ans. 3) Numpy functions meshgrid and ravel:

meshgrid: numpy.meshgrid(*xi, **kwargs)

The numpy.meshgrid function is used to create a rectangular grid out of two given one-dimensional arrays representing the Cartesian indexing or Matrix indexing.

- Return coordinate matrices from coordinate vectors.
- Make N-D coordinate arrays for vectorised evaluations of N-D scalar/vector fields over N-D grids, given one-dimensional coordinate arrays x1, x2… xn.

Parameters:

- **x1, x2,…, xn (array_like):** This parameter defines the 1-dimensional array, which represents the coordinates of a grid.
- **indexing: {'xy', 'ij'}(optional):** This is an optional argument which defines the Cartesian 'xy'(by default) or matrix ('ij') indexing of output.
- **sparse: bool(optional)**: This parameter is also optional. If we need a sparse grid for conserving memory, we have to set this parameter to True. By default, it is set to False.
- **copy: bool(optional)**: The aim of this optional argument is that it returns a copy of the original array for conserving memory. By default, it is set to False.

- If both **sparse** and **copy** parameters are set to False, then it will return non-contiguous arrays. In addition, more than one element of a broadcast array can refer to a single memory location. If we need to write into the arrays, then we have to make copies first.

Returns: (x1, x2, …, xn) The coordinate length from the coordinate vector is returned from this function.

ravel: `numpy.ravel(a, order='C')`

The numpy module of Python provides a function called numpy.ravel, which is used to change a 2-dimensional array or a multi-dimensional array into a contiguous flattened array. The returned array has the same data type as the source array or input array. If the input array is a masked array, the returned array will also be a masked array.

- Return a contiguous flattened array.
- A 1-D array, containing the elements of the input, is returned. A copy is made only if needed.

Parameters:

- **a (array_like):** This parameter defines the input array, which we want to change in a contiguous flattened array. The array elements are read in the order specified by the order parameter and packed as a 1-D array.
- **order: {'C','F', 'A', 'K'}(optional):** If we set the order parameter to 'C', it means that the array gets flattened in row-major order. If 'F' is set, the array gets flattened in column-major order. The array is flattened in column-major order only when 'A' is Fortran contiguous in memory, and when we set the order parameter to 'A'. The last order is 'K', which flatten the array in same order in which the elements occurred in the memory. By default, this parameter is set to 'C'.

Returns: This function returns a contiguous flatten array with the same data type as an input array and has shape equal to (**x.size**).