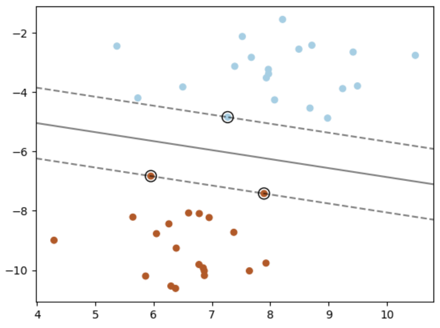
SKLEARN SVM SVC SHORT WRITE UP

Arham Lodha J031

1)SVC



C-Support Vector Classification.

The implementation is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. For large datasets consider using `LinearSVC` or `SGDClassifier`

Code :-

`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*)

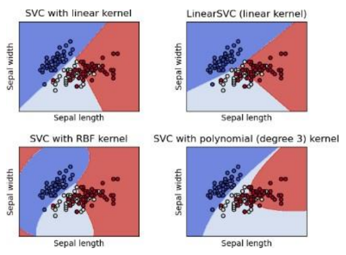
Important hyperparameters are :-

Cfloat, default=1.0

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel*{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default='rbf'*

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.



degree*int, default=3*

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma*{'scale', 'auto'} or float, default='scale'*

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features
-

Some Important Attributes are :-

coef_*ndarray of shape (n_classes * (n_classes - 1) / 2, n_features)*

Weights assigned to the features (coefficients in the primal problem). This is only available in the case of a linear kernel.

intercept_*ndarray of shape (n_classes * (n_classes - 1) / 2,)*

Constants in decision function.

support_*ndarray of shape (n_SV)*

Indices of support vectors.

support_vectors_*ndarray of shape (n_SV, n_features)*

Support vectors.

n_support_*ndarray of shape (n_classes,), dtype=int32*

Number of support vectors for each class.

probA_*ndarray of shape (n_classes * (n_classes - 1) / 2)*
probB_*ndarray of shape (n_classes * (n_classes - 1) / 2)*

Application :-

```
>>> import numpy as np
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.preprocessing import StandardScaler
>>> X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
>>> y = np.array([1, 1, 2, 2])
>>> from sklearn.svm import SVC
>>> clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
>>> clf.fit(X, y)
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('svc', SVC(gamma='auto'))])
>>> print(clf.predict([[-0.8, -1]]))
[1]
```

Methods :-

Fit(X,y)-fit the model according to the given training data

Predict(x)-predict class labels

Score(X,y)-returns mean accuracy on the given test data and label