SVM

implement

- Fit a hyperplane that best classifies the data points while maximizing the margin, which is the distance between the nearest data point of each class and the hyperplane.
- Margin: distance between line and closest points
- Support vectors: points that determine the margin (i.e. lie on the margin, are closest together)
- note:
  Points can be within the margin
  Points can be on the wrong side of the hyperplane
  Points on the right side of the margin have no bearing on the computation
  Some points can be misclassified
  Some points can be within the margin

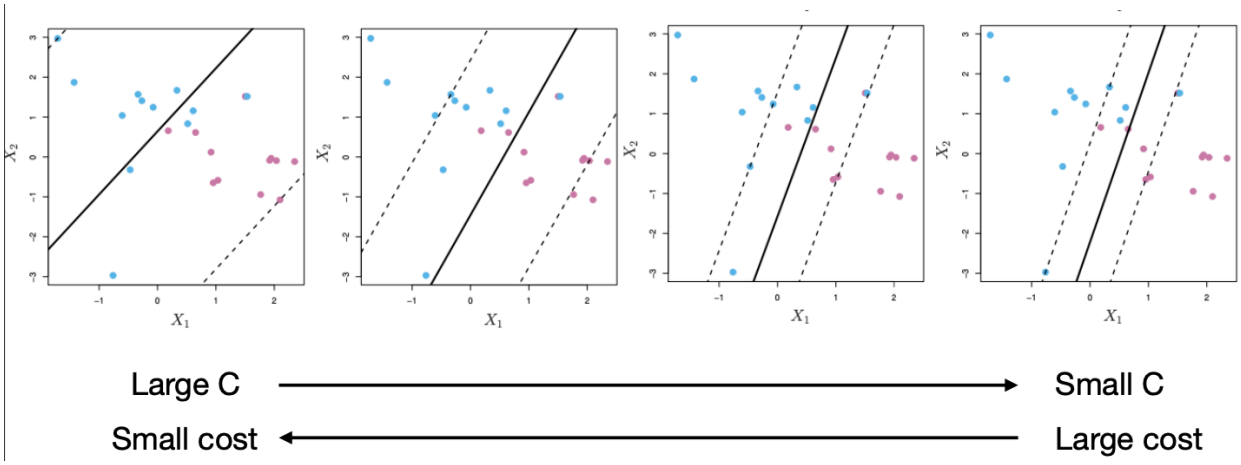
Variations and Extensions

Kernel Functions: Transform data into higher dimensions to make it easier to separate with a hyperplane. Common kernels include linear, polynomial, and radial basis functions.

Parameters

C: “budget” for points on the wrong side of the margin

a larger C value will narrow the range of the minimum misclassification limit, while a smaller C value will expand the range, allowing more misclassified data.



γ Parameter: Applies in kernels like the Radial Basis Function (RBF).

Degree & coef0 : Applies in kernels like the polynomial.

Advantages and Disadvantages

Advantages:

Effective in high-dimensional spaces.

Still effective when the number of dimensions exceeds the number of samples.

Uses a subset of training points (support vectors), making it memory efficient.

Disadvantages:

Not suitable for large datasets as the training time with SVMs is high.

Less effective on noisier datasets with overlapping classes.

The choice of the kernel and regularization can have a large impact on performance.

Plots & Code:

The support vector classifier with two features can be visualized by plotting values of its *decision function*. We have included a function for this in the `ISLP` package (inspired by a similar example in the `sklearn` docs).

‘’’

```
from ISLP.svm import plot as plot_svm
fig, ax = subplots(figsize=(8,8))
plot_svm(X,
         y,
         svm_linear,
         ax=ax)
```

’’’

‘’’

```
svm_linear = SVC(kernel='linear').fit(X, y)
svm_rbf = SVC(kernel='rbf').fit(X, y)
svm_poly= SVC(kernel='poly').fit(X, y)
```

’’’

Cross vaildation

‘’’

```
kfold = skm.KFold(5,
                  random_state=0,
                  shuffle=True)
grid = skm.GridSearchCV(svm_rbf,
                        {'C':[0.1,1,10,100,1000],
                         'gamma':[0.5,1,2,3,4]},
                        refit=True,
                        cv=kfold,
                        scoring='accuracy');
grid.fit(X_train, y_train)
grid.best_params_
```

’’’

ROC Curve and AUC Score: Useful for evaluating the classifier's ability to discriminate between classes.

‘’’

```
fig, ax = subplots(figsize=(8,8))
roc_curve(best_svm,
        X_train,
        y_train,
        name='Training',
        color='r',
        ax=ax);
```

‘’’