SciKit-Learn and SCV

MSXPPDSX2016-V005000

Unlike linear regression, SVC is a very configurable algorithm. When you first start using it, you might feel a bit overwhelmed with the options or lost, not knowing that to tinker with. Just take it a parameter at a time, visualizing you output after each adjustment, and that way, you'll gain a better experiential understanding of each parameter on a per-parameter basis. Later on, you can adjust multiple parameters simultaneously.

Of the many configurable parameters for SciKit-Learn's svm.SVC class, the most important three in order are:

- **kernel** Defines the type of kernel used with your classifier. The default is the radial basis function *rbf*, the most popular kernel used with support vector machines generally. SciKit-Learn also supports linear, poly, sigmoid, and precomputed kernels. You can also specify a user defined function to pre-compute the kernel matrix from your sample's feature space, which should be shaped [n_samples, n_samples].
- **C** This is the penalty parameter for the error term. Do you want your SVC to never miss a single classification? Or is having a more generalized solution important to you? The lower your C value, the smoother and more generalized your decision boundary is going to be. But if you have a large C value, the classifier will attempt to do whatever is in its power to squiggle and wiggle between each sample to correctly classify it.
- **gamma** This parameter's value is inversely proportional to the extent a single training sample's influence extends. Large gamma values result in each training sample having localized effects only. Smaller values result in each sample affecting a larger area. In essence, the gamma values dictate how pronounced your decision boundary is by varying the influence of your support vector samples.

To get started with SVC, import it as usual:

```
>>> from sklearn.svm import SVC
>>> model = SVC(kernel='linear')
>>> model.fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In addition to the regular `.fit()`, `.predict()`, and `.score()` methods, SVC also allows you to calculate the distance of a set of samples to the decision boundary in high-dimensional space using `.decision_function(X)`, where `X` is a series of samples of the form `[n_samples, n_features]`.

In terms of attributes, a few goodies are exposed here to:

- **support_** Contains an array of the indices belonging to the selected support vectors
- **support_vectors_** The actual samples chosen as the support vectors
- **intercept_** The constants of the decision function
- **dual_coef_** Each support vector's contribution to the decision function, on a per classification basis. This has *similarities* to the weights of linear regression