Support Vector Machines

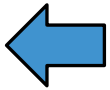
Linear SVM: The Syntax

Import the class containing the classification method

```
from sklearn.svm import LinearSVC
```

Create an instance of the class

```
LinSVC = LinearSVC(penalty='l2', C=10.0)
```



regularization parameters

Linear SVM: The Syntax

Import the class containing the classification method

```
from sklearn.svm import LinearSVC
```

Create an instance of the class

```
LinSVC = LinearSVC(penalty='l2', C=10.0)
```

Fit the instance on the data and then predict the expected value

```
LinSVC = LinSVC.fit(X_train, y_train)
y_predict = LinSVC.predict(X_test)
```

Linear SVM: The Syntax

Import the class containing the classification method

```
from sklearn.svm import LinearSVC
```

Create an instance of the class

```
LinSVC = LinearSVC(penalty='l2', C=10.0)
```

Fit the instance on the data and then predict the expected value

```
LinSVC = LinSVC.fit(X_train, y_train)
y_predict = LinSVC.predict(X_test)
```

Tune regularization parameters with cross-validation.

Kernels

SVMs with Kernels: The Syntax

Import the class containing the classification method

```
from sklearn.svm import SVC
```

Create an instance of the class

```
rbfSVC = SVC(kernel='rbf', gamma=1.0, C=10.0)
```

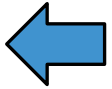
SVMs with Kernels: The Syntax

Import the class containing the classification method

```
from sklearn.svm import SVC
```

Create an instance of the class

```
rbfSVC = SVC(kernel='rbf', gamma=1.0, C=10.0)
```



set kernel and associated coefficient (gamma)

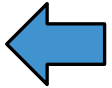
SVMs with Kernels: The Syntax

Import the class containing the classification method

```
from sklearn.svm import SVC
```

Create an instance of the class

```
rbfSVC = SVC(kernel='rbf', gamma=1.0, C=10.0)
```



"C" is penalty associated with the error term

SVMs with Kernels: The Syntax

Import the class containing the classification method

```
from sklearn.svm import SVC
```

Create an instance of the class

```
rbfSVC = SVC(kernel='rbf', gamma=1.0, C=10.0)
```

Fit the instance on the data and then predict the expected value

```
rbfSVC = rbfSVC.fit(X_train, y_train)
y_predict = rbfSVC.predict(X_test)
```

SVMs with Kernels: The Syntax

Import the class containing the classification method

```
from sklearn.svm import SVC
```

Create an instance of the class

```
rbfSVC = SVC(kernel='rbf', gamma=1.0, C=10.0)
```

Fit the instance on the data and then predict the expected value

```
rbfSVC = rbfSVC.fit(X_train, y_train)
y_predict = rbfSVC.predict(X_test)
```

Tune kernel and associated parameters with cross-validation.

Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import Nystroem
```

Create an instance of the class

```
nystroemSVC = Nystroem(kernel='rbf', gamma=1.0,
                       n_components=100)
```

Fit the instance on the data and transform

```
X_train = nystroemSVC.fit_transform(X_train)
X_test = nystroemSVC.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import Nystroem
```

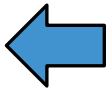
Create an instance of the class

```
nystroemSVC = Nystroem(kernel='rbf', gamma=1.0,
                       n_components=100)
```

Fit the instance on the data and transform

```
X_train = nystroemSVC.fit_transform(X_train)
X_test = nystroemSVC.transform(X_test)
```

Tune kernel parameters and components with cross-validation.



multiple non-linear kernels can be used

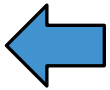
Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import Nystroem
```

Create an instance of the class

```
nystroemSVC = Nystroem(kernel='rbf', gamma=1.0,
                       n_components=100)
```



kernel and gamma are identical to SVC

Fit the instance on the data and transform

```
X_train = nystroemSVC.fit_transform(X_train)
X_test = nystroemSVC.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

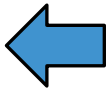
Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import Nystroem
```

Create an instance of the class

```
nystroemSVC = Nystroem(kernel='rbf', gamma=1.0,
                       n_components=100)
```



n_components is number of samples

Fit the instance on the data and transform

```
X_train = nystroemSVC.fit_transform(X_train)
X_test = nystroemSVC.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import RBFsampler
```

Create an instance of the class

```
rbfSample = RBFsampler(gamma=1.0,
                       n_components=100)
```

Fit the instance on the data and transform

```
X_train = rbfSample.fit_transform(X_train)
X_test = rbfSample.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

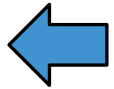
Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import RBFsampler
```

Create an instance of the class

```
rbfSample = RBFsampler(gamma=1.0,
                       n_components=100)
```



RBF is only kernel that can be used

Fit the instance on the data and transform

```
X_train = rbfSample.fit_transform(X_train)
X_test = rbfSample.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

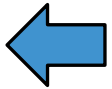
Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import RBFsampler
```

Create an instance of the class

```
rbfSample = RBFsampler(gamma=1.0,
                       n_components=100)
```



parameter names are identical to previous

Fit the instance on the data and transform

```
X_train = rbfSample.fit_transform(X_train)
X_test = rbfSample.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

When to Use Logistic Regression vs SVC

Features	Data	Model Choice
Many (~10K Features)	Small (1K rows)	Simple, Logistic or LinearSVC
Few (<100 Features)	Medium (~10k rows)	SVC with RBF
Few (<100 Features)	Many (>100K Points)	Add features, Logistic, LinearSVC or Kernel Approx.