Support Vector Machines

Dr. Ashish Tendulkar

IIT Madras

Machine Learning Practice

- In this week, we will study how to implement support vector machines for classification tasks with sklearn.

Support Vector Machines

- Support Vector Machines (SVM) are a set of supervised learning methods used for classification, regression and outliers detection.

- SVM constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks.

- In sklearn, we have three methods to implement SVM.

SVC

NuSVC

These are similar methods but, accept slightly different sets of parameters.

Implementation is based on libsvm.

LinearSVC

Faster implementation of linear SVM classification with only linear kernel.

Implementation is based on liblinear.

Training data

Array X : holding the training samples

```
X = [[0, 0], [1, 1]]
```

shape $\rightarrow$ (n_samples, n_features)

Array y : holding the class labels (strings or integers)

```
y = [0,1]
```

shape $\rightarrow$ (n_samples)

How to implement SVC (C-Support Vector Classification)?

Step 1: Instantiate a SVC classifier estimator.

```
from sklearn.svm import SVC
SVC_classifier = SVC()
```

Step 2: Call fit method on SVC classifier object with training feature matrix and label vector as arguments.

```
# Model training with feature matrix X_train and
# label vector or matrix y_train
SVC_classifier.fit(X_train, y_train)
```

How to perform regularization in SVC classifier?

`C` → Regularization parameter

↓

float value

Default:

```
SVC_classifier = SVC(C=1.0)
```

Note:

- strength of the regularization is inversely proportional to C
- strictly positive
- penalty is a squared l2 penalty

How to specify kernel type to be used in the algorithm ?

`kernel` → 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'

Default:

```
SVC_classifier = SVC(kernel = 'rbf')
```

- If `kernel = poly` , set `degree` (any integer value)
- If `kernel = callable` is given it is used to pre-compute the kernel matrix from data matrices

How to set kernel coefficient for *'rbf'*, *'poly'* and *'sigmoid'* kernels?

`gamma`

- 'scale' → value of gamma = $\frac{1}{\text{number of features*X.Var()}}$
- 'auto' → value of gamma = $\frac{1}{\text{number of features}}$
- float value

Default:

```
1 SVC_classifier = SVC(gamma = 'scale')
```

- If `kernel = 'poly'` or `'sigmoid'` , set `coef0` which is an independent term in kernel function (any integer value)

How to view support vectors?

After the classifier is fit on the training data, there are few attributes which reveal the details of support vectors.

```
from sklearn.svm import SVC
SVC_classifier = SVC()
clf = SVC_classifier.fit(X_train, y_train)

#to view indices of the support vectors
clf.support_

#to view the support vectors
clf.support_vectors_

#to view the number of support vectors for each class
clf.n_support_
```

How to implement NuSVC (ν-Support Vector Classification)?

Step 1: Instantiate a NuSVC classifier estimator.

```
from sklearn.svm import NuSVC
NuSVC_classifier = NuSVC()
```

Step 2: Call fit method on NuSVC classifier object with training feature matrix and label vector as arguments.

```
# Model training with feature matrix X_train and
# label vector or matrix y_train
NuSVC_classifier.fit(X_train, y_train)
```

What is the significance of ν in NuSVC?

Instead of C in SVC, ν is introduced in NuSVC to control the number of support vectors and margin errors.

ν is an upper bound on the fraction of margin errors and and a lower bound of the fraction of support vectors.

Value of ν should $\in (0, 1]$

Default: $\nu = 0.5$

Other parameters for NuSVC are same as that of SVC.

How to implement LinearSVC (Linear Support Vector Classification)?

Step 1: Instantiate a LinearSVC classifier estimator.

```
from sklearn.svm import LinearSVC
LinearSVC_classifier = LinearSVC()
```

Step 2: Call fit method on SVC classifier object with training feature matrix and label vector as arguments.

```
# Model training with feature matrix X_train and
# label vector or matrix y_train
LinearSVC_classifier.fit(X_train, y_train)
```

Advantages of LinearSVC

- It has more flexibility in the choice of penalties and loss functions since it is implemented in terms of liblinear.
- Scales better to large numbers of samples.
- Supports both dense and sparse input.

How to provide penalty in LinearSVC classifier?

`penalty`

- $l1$ - adds a L1 penalty term
- $l2$ - adds a L2 penalty term

- $l1$ - leads to `coef_` vectors that are sparse.

Default:
```
LinearSVC_classifier = Linear_SVC(penalty = 'l2')
```

How to choose loss functions in LinearSVC classifier?

`loss` parameter

'hinge' - standard SVM loss

'squared_hinge' - square of the hinge loss

Default:

```
LinearSVC_classifier = Linear_SVC(loss = 'squared_hinge')
```

Combination not supported:

`penalty='l1'` and `loss='hinge'`

Some parameters in LinearSVC classifier

`C`

↓

Regularization parameter

`dual`

↓

- Select the algorithm to either solve the dual or primal optimization problem.
- When n_samples >n_features, prefer dual=False.

`fit_intercept`

↓

To calculate the intercept for the model.

How to perform multi-class classification using SVM?

- SVC and NuSVC implement the "one-versus-one" approach for multi-class classification.

decision_function_shape
- 'ovo'
- 'ovr'

- LinearSVC implements "one-vs-the-rest" approach for multi-class classification.

multi_class
- 'ovr'
- 'crammer_singer'

Advantages of SVM

- Effective in high dimensional spaces.
- Effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel Functions can be specified for the decision function.

Disadvantages of SVM

- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.
- Avoid over-fitting in choosing Kernel functions if the number of features is much greater than the number of samples.