Landon Buell

18 Feb 2020

Breaking Down Sci-kit Learn’s SGDClassifier Object

**Introduction**

Sci-kit learn (sklearn) is a standard, open source tool set module for python that primarily deals in functions, objects and algorithms pertaining to machine learning. There are various submodules within this tool set that give rise to methods for different uses. For this paper, I will break down the SGD Classifier object from the *linear\_model* submodule.  SGDClassifier is short for *Stochastic Gradient Descent Classifier*, which is multi-purpose optimization algorithm commonly used in machine learning practices. By breaking down this module, we will explore all what computations and operations are being performed ‘under the hood’ so that we may possibly manipulate the object to better suite a specific need.

Documentation page for sklearn.linear\_model.SGDClassifier:

<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html>

**The SGDClassifier Class Object**

The SGDClassifier is a pre-created python class object. It operates just as any object, it has a series of methods, attributes and initialization parameters, and its properties can be changed dynamically as well. To understand how the object works, we’ll examine the class a few of the class parameters, and methods. See the documentation page linked above for a more detailed explanations as needed.

We’ll start by loading in a toy dataset, the IRIS set, and the SGDClassifier object from the linear\_model submodule. We can extract the data matrix (called *X* by convention) and the target vector (called *y* by convention). We will also import a few other modules that will be needed later on. Create and instance of the SGD object, and we set a few parameters. See documentation for details.

Documentation page of IRIS dataset:

<https://scikit-learn.org/stable/datasets/index.html#iris-dataset>



Now, we can right click on the *SGDClassifier* text , to open a pop-up window, and then selected *‘Go to definition’*. In most IDE’s (like Visual Studio) , this opens up a new tab, with the source code for that particular class or module. Starting at line 714, we can see the SGD classifier definitions. It is also a child class to the parent class *BaseSGDClassifier*.

**The \_\_init\_\_() method**

By convention, when a class instance is created, the method \_\_init\_\_(), short for initialization is called. This method begins at line 936. It assigns to this particular instance all of the necessary values, most are given by default. If a user sets a parameter manually, then that overrides the default setting. After this method is finished, the instance has been created and the necessary attributes and parameters have been attached to the object.

**The fit() method**

The fit() method is the primary method used to train the classifier object. To use this, we need to create a set of training data and the corresponding training labels. We implement a standard splitting algorithm, also from sklearn, and then use the training set and targets to fit the class instance. This is “training the model”

Documentation for sklearn.model\_selection.train\_test\_split():

<https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>

****

Now we have functionally trained the model and are ready to make predictions of our testing data set or another validation set. But first, we examine the fit() method closer to see what is really happening. This method actually calls the parent class “BaseSGDClassifier” on like 435. The method itself is defined on 679.

The method takes the arguments X and y which are the training data and labels respectively, coef\_init, intercept\_init, and sample\_rate which are all set to *None* types by default. When the fit() methodis called, it returns the \_fit() method on line 518. This is where the operations of the method take place. We can break this method down section by section. For reference, in the documentation it appears as:



**The \_fit() Method**

Most of this method is dedicated to validating the input parameters given to the algorithm. It serves to ensure that all of the arguments passed into the function are reasonable and comply with expected inputs. (See line 122 for self.\_validate\_params() method details). After this is done, the method sets an iteration counter, and then calls the self.\_partial\_fit() method. At this point, it is important to know that even after calling the fit() method on the classifier, we have not yet performed any computations, but merely called other functions that organize the arguments and validate them.

**The \_partial\_fit() Method**

Ths method is defined on line 471.