A4 User Guide

Keycap Guardians

Link to Issue: https://github.com/scikit-learn/scikit-learn/issues/20595

What is the problem?

When the user uses the functions *barycenter_weights* or *barycenter_kneighbors_graph* (which calls *barycenter_weights*), the process of estimating the weights to assign to each point in Y[indices] to recover the point X[i] (where X and Y are input arrays that we want to compute barycenter weights of X from Y along the first axis) is both in of itself very computationally expensive and also is the most computationally expensive step to execute in the function. Currently, it is done sequentially, which takes a long time to execute especially when the input arrays are big or repetitive calls are made to *barycenter_weights*. The issue(linked above) pointed out that executing this process sequentially is suboptimal in terms of runtime.

How did we improve performance?

We parallelized the above computationally expensive step by utilizing python's *multiprocessing* package using the *pool* object which offloads the work to CPU cores to work parallelly depending on the number of cores your computer has. Although for arrays of smaller sizes we observed similar run times and sometimes deprovements (note that runtime of smaller arrays are more unstable), we were able to observe some improvements for arrays of larger sizes. For example, for a 3000 x 3000 input matrix, the runtime is 2.52s for the sequential version and 2.12s for the parallel version, which amounts to a 16% decrease in runtime. This difference can be meaningfully significant if the input gets even larger and repetitive calls to the function are made.

*More references to tests below.

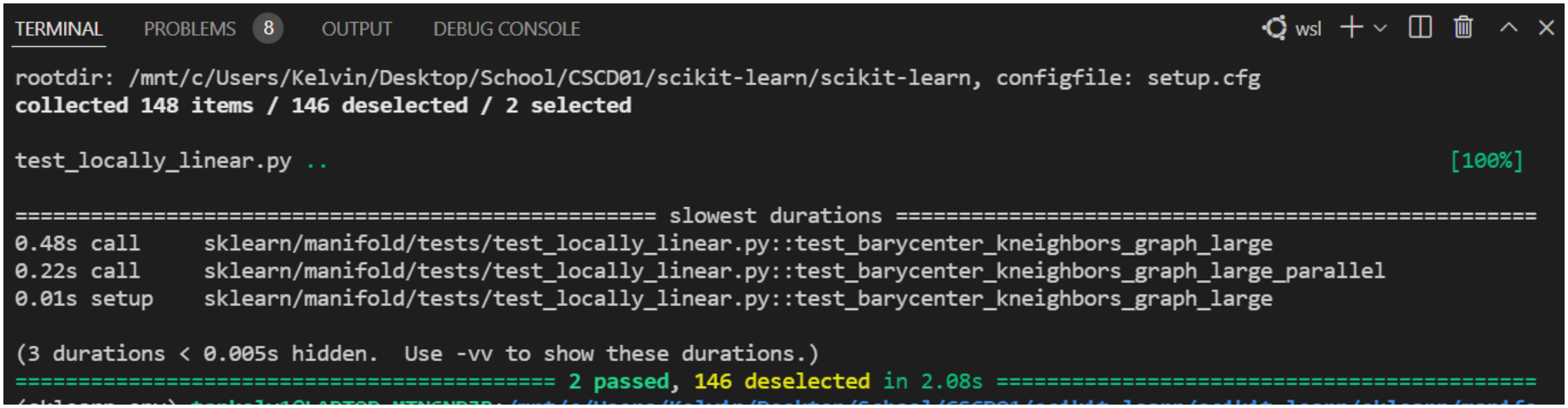
How can the user use the improved function?

We kept the original *barycenter_weights* function as is and added a *barycenter_weights_parallel* function below it which is the same as *barycenter_weights* except that it parallelizes the estimation of weights step. So users can access the parallelized version of this function by calling *barycenter_weights_parallel* with the same parameters.

For the *barycenter_kneighbors_graph* function, we added a *parallel* parameter in its input with the default value equal to True, so the user would, by default, access the parallelized version of the *barycenter_weights* function. However, if the user prefers to use the sequential version of the *barycenter_weights* function, they can do so by setting the *parallel* parameter to False.

Test References

1000 x 1000 matrix inputs | 0.48s for sequential, 0.22s for parallel



3000 x 3000 matrix inputs | 2.52s for sequential, 2.12s for parallel