This package consists of the source codes of streamC3S and streamC3S\_E, which were developed by Dr. Yunlong Mi for dynamic concept learning under the stream datasets. One can use them freely (for academic purposes only) at your own risk. For other purposes, please contact Dr. Yunlong Mi directly.

**Quickstart for streamC3S or streamC3S \_E (work on Eclipse workstation):**

**JDK environment: jdk 1.8 or above.**

**Data format**: The Data of the CSV format file looks like the following:

29.7037,21.3278,1.8359,0

30.4719,5.5551,36.8715,0

33.2494,-3.937,52.1075,0

…

**Step 1:** set the file path in ParametersUtil.java

|  |
| --- |
| **public** **static** String *train\_path* = "./data/TrafficStream1\_initialData[indexNum].csv";  **public** **static** String *test\_path* = "./data/TrafficStream1\_streamData[indexNum].csv"; |

**Step 2:** set some related parameters in ParametersUtil.java, such as the parameters for TrafficStream1 as follows:

|  |
| --- |
| /\*\*  \* 0:static learning; this function is simple implementation in C3S. <br/>  \* 1:partially labeled data stream; note that, this function will be developed in the next version. <br/>  \* 2:labeled data stream, this function is for streamC3S or streamC3S\_E. <br/>  \*/  **public** **static** **int** *U* = 2;  /\*\* U=2, streamC3S or streamC3S\_E: error corrects rate for concept drift detection. \*/  **public** **static** String *methodType* = "streamC3S\_E";  /\*\* Show the results by bachSize or overall accuracies.\*/  **public** **static** String *showResult*="overall";    /\*\* Three required parameters \*/  /\*\* Lambda(i): The $\theta$ value \*/  **public** **static** **int** *thetaT* = 8;// Start: Theta  **public** **static** **int** *thetaE* = 8;// End: Theta  /\*\* MaxSize: The size of concept spaces for each class. \*/  **public** **static** **int** *conceptSZ* =300;  /\*\* Chunk size: The size of each data chunk. \*/  **public** **static** **int** *C* =100;  /\*\* Four optional parameters \*/  /\*\* Alpha: The concept similarity threshold. \*/  **public** **static** **double** *distF* = 0.6;  /\*\* Epsilon: The range of the local $\alpha$-concept neighborhood. default radius=5. \*/  **public** **static** **int** *radius* = 2;  /\*\* Delta\_{d}: For U=2 and fastC3S\_E, when errorCorrect < conceptDriftTheta, concept drift occurs. \*/  **public** **static** **double** *conceptDriftTheta*=-0.2;//[-1,0], default conceptDriftTheta=-0.15.  /\*\* Delta\_{w}: For U=2 and fastC3S\_E, when errorCorrect > errorconceptTheta, concept warning occurs. \*/  **public** **static** **double** *conceptWarningTheta* =-0.2;//[-1,1], selected from [-0.05,0,0.05] |

**Step 3:** run streamC3S or streamC3S\_E in runMethod.java

|  |
| --- |
| /\*\* Load datasets. \*/  **long** s1 = System.*currentTimeMillis*();  Vector<Object> grow\_vec = LoadDataUtil  .*loadData*(ParametersUtil.*train\_path*.replace("indexNum", String.*valueOf*(index)));  Vector<Object> test\_vec = LoadDataUtil  .*loadData*(ParametersUtil.*test\_path*.replace("indexNum", String.*valueOf*(index)));  **long** e1 = System.*currentTimeMillis*();  System.***err***.println("Load dataset：" + (e1 - s1) + "(ms)");  /\*\* Instantiation system, 实例化系统 \*/  C3S c3s = **new** C3S(grow\_vec);  /\*\* Initial system, 系统初始化 \*/  **long** s2 = System.*currentTimeMillis*();  c3s.initialS();  **long** e2 = System.*currentTimeMillis*();  System.***err***.println("Initial system：" + (e2 - s2) + "(ms)");  /\*\* Learning for system, 系统学习 \*/  **long** s3 = System.*currentTimeMillis*();  c3s.trainS();  **long** e3 = System.*currentTimeMillis*();  System.***err***.println("Training system：" + (e3 - s3) + "(ms)");  /\*\* Evaluating and updating system, 系统动态更新与评估 \*/  **long** s4 = System.*currentTimeMillis*();  c3s.evaluateS(test\_vec);  **long** e4 = System.*currentTimeMillis*();  System.***err***.println("Evaluating system：" + (e4 - s4) + "(ms)"); |

**NOTE**: Please cite our work if you use these source condes in any way as follows:

Y. L. Mi. Dynamic Concept Learning with Stream Concept-cognitive Computing System, 2020, submitted to IEEE TPAMI for consideration.