# Laufzeit-Optimierung Shark-Methode

Inhalt

[Laufzeit-Optimierung Shark-Methode 1](#_Toc471891493)

[1. Analyse/Infos - Lange Laufzeit bei Methode “NumericVectorToDataRealVector” 1](#_Toc471891494)

[2. Alternativen 2](#_Toc471891495)

[2.1. std::vector<double> statt Data<RealVector> 2](#_Toc471891496)

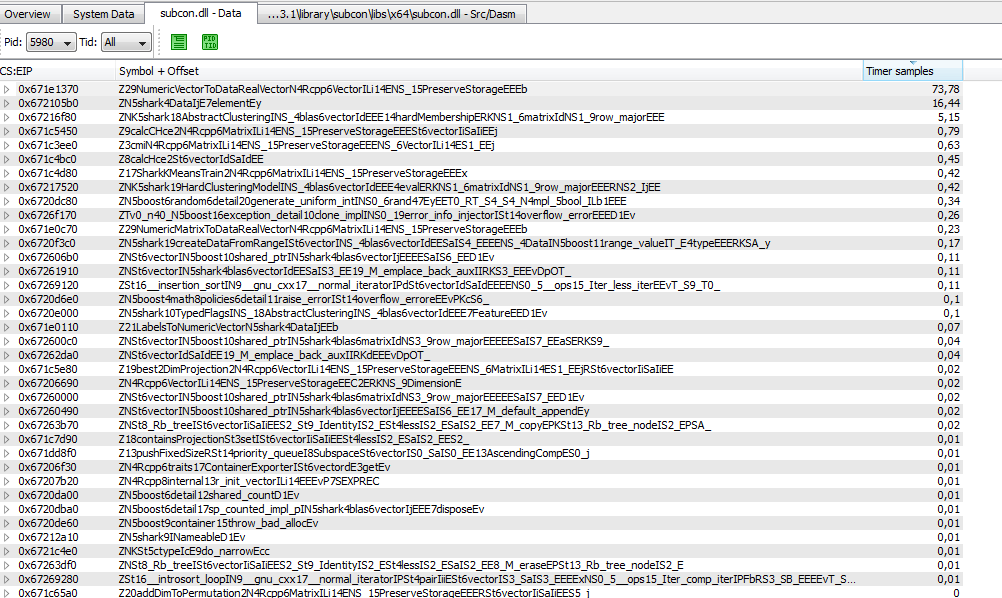
[2.2. RealVector direkt aufbauen statt zuerst eine Copy anzufertigen 3](#_Toc471891497)

[2.3. Typecast von std::vector zu RealVector 3](#_Toc471891498)

[2.4. ALGLIB (C++-Library) einbinden statt Shark 4](#_Toc471891499)

# Analyse/Infos - Lange Laufzeit bei Methode “NumericVectorToDataRealVector”

* Performance-Analyse mit dem AMD CodeAnalyst ergab eine lange Laufzeit bei der Methode „NumericVectorToDataRealVector“:



* Implementierung befindet sich in der Datei utils.cpp

Infos zum Shark-Package

<http://image.diku.dk/shark/doxygen_pages/html/_base_8h.html>

<http://image.diku.dk/shark/doxygen_pages/html/classshark_1_1_data.html>

<http://image.diku.dk/shark/sphinx_pages/build/html/rest_sources/tutorials/algorithms/kmeans.html?highlight=realvector>

KMeans.h liegt unter C:\R\R-3.1.1\library\RcppShark\include\shark\Algorithms

Auszug Callstack:

|  |  |  |
| --- | --- | --- |
| **Datei** | **Methode** | **ruft auf:** |
| CMI.cpp | calcCHce | SharkKMeansTrain2 |
| CMI.cpp | SharkKMeansTrain2 | NumericMatrixToUnlabeledData |
| CMI.cpp | SharkKMeansTrain2 | kMeans |
| CMI.cpp | SharkKMeansTrain2 | LabelsToNumericVector |
|  |  |  |
| utils.cpp | NumericMatrixToUnlabeledData | NumericMatrixToDataRealVector |
| utils.cpp | NumericMatrixToDataRealVector | std::copy-function |
| utils.cpp | NumericMatrixToDataRealVector | createDataFromRange |

# Alternativen

# std::vector<double> statt Data<RealVector>

**NumericMatrixToDataRealVector2**

Direktes Befüllen des vectors mit Inhalten:

std::vector<double> output(size);

for(int i=0; i<X.rows(); i++){

for(int j=0; j<X.cols(); j++){

output[(i\*X.cols())+j] = X(i,j);

}

}

* Problem: kmeans-Meth. erwartet die Daten als Datentyp UnlabeledData<RealVector>

# RealVector direkt aufbauen statt zuerst eine Copy anzufertigen

**NumericMatrixToDataRealVector3**

RealVector tRV(X.cols());

for(int c=0; c<X.cols(); c++){

tRV.push\_back(X( e, c));

}

outputStd.push\_back(tRV);

Ergebnis: 717 sek. (= fast 12 Min.)

> numClusterION <- 10

> topkSearchION <- 500

> topkOutputION <- 100

>

> ####### CMI #######

> startCmiION <- Sys.time()

> CMIResultION = CMISearch(m, numClusterION, topkSearchION, topkOutputION)

number of 2-dim candidates: 496

number of 3-dim candidates: 4960

number of 4-dim candidates: 817

number of 5-dim candidates: 290

number of 6-dim candidates: 83

number of 7-dim candidates: 10

> endCmiION <- Sys.time()

>

> difftime(endCmiION, startCmiION, unit="sec")

Time difference of 717.7345 secs

Vgl.test Originalimplementierung: 559 sec. (=9,3 Min.)

* Keine Performance-Verbesserung

# Typecast von std::vector zu RealVector

**NumericMatrixToDataRealVector4**

outputStd[(i\*X.cols())+j] = (RealVector)X(i,j);

* Problem: Programm kompiliert zwar, aber zur Laufzeit hängt es sich auf
* Offen: weiter ausprobieren

# ALGLIB (C++-Library) einbinden statt Shark

* + Offen:Man könnte noch versuchen die ALGLIB (C++-Library) einzubinden**:**

<http://www.alglib.net/download.php>

* + besitzt auch eine kmeans-Implementierung:

