FlowSOM in Python
Analyse van cytometriedata

Auteur: Janne Cools *12/06/2023*

Abstract

In dit project wordt het FlowSOM algoritme geïmplementeerd in Python. Er wordt besproken hoe dit tot stand is gekomen, samen met ondervonden problemen en wijzigingen. Zowel optimalisaties in het algoritme als de gebruiksvriendelijkheid komen aan bod, alsook de resultaten. Dit laatste bevat grafieken over de snelheden van de gebruikte algoritmes en enkele voorbeelden van de plots die verkregen worden bij het uitvoeren van het FlowSOM algoritme op een dataset. Ten slotte wordt vermeld wat er in de toekomst nog gewijzigd of toegevoegd zou kunnen worden om het algoritme en de gebruiksvriendelijkheid te verbeteren.

In de biologie wordt flowcytometrie gebruikt om verschillende immuuncelpopulaties te bestuderen en zo het immuunprofiel van een patiënt te bepalen. Hiermee kunnen allergische reacties, immuundeficiënties en reacties op pathogenen onderzocht worden.
Het Vlaams Instituut voor Biotechnologie (VIB) heeft voor de analyse van de cytometriedata een algoritme ontworpen, genaamd FlowSOM [1]. Dit algoritme bestaat uit vier fases. Ten eerste wordt de input gelezen en worden eventueel transformaties en herschalingen toegepast op de data. Vervolgens wordt een self-organising map opgesteld en getraind, die weergeeft welke markers het meest aanwezig zijn in samples. Alle clusters van de SOM kunnen aan elkaar verbonden worden via een minimal spanning tree die gelijkaardige elementen dichter bij elkaar plaatst. Als laatste wordt metaclustering gebruikt om verschillende som-samples samen te voegen tot een geheel.
In dit project worden de 3 laatste fases van dit algoritme onder de loep genomen en geïmplementeerd in Python.

Methoden

Implementatie algoritme

Allereerst was het belangrijk om het FlowSOM algoritme te begrijpen. Hiervoor heb ik het artikel [2] gelezen die de algemene principes uitlegde en heb ik de powerpoint meermaals grondig doornomen. Hiermee kreeg ik een vrij goed beeld van de werking van het algoritme en kon ik beginnen met de implementatie ervan.

Voor het genereren van de self-organising map (SOM) heb ik de MiniSOM-bibliotheek [3] gebruikt. Bij het trainen gaf ik enkele belangrijke argumenten mee, zoals de learning rate en de radius. De resultaten van deze SOM konden dan getoond worden via een minimal-spanning tree (MST). Voor het construeren van deze boom heb ik de NetworkX-bibliotheek [4] gebruikt. Hierbij heb ik eerst een graaf gemaakt waarbij er tussen elk paar toppen een boog is met een afstand berekend op basis van hun SOM-gewichten. Deze graaf werd dan door NetworkX omgezet in een minimaal opspannende boom. De bogen ervan liet ik plotten door NetworkX in combinatie met de bibliotheek PyGraphviz [5], die ervoor zorgt dat de boom op een ordelijke manier vertoond wordt zonder kruisende bogen. De toppen zelf heb ik via een andere bibliotheek geplot, namelijk Matplotlib. Ik wilde immers voor gebruiksvriendelijkheid bij elke top zijn SOM-gewichten tonen, die informeert over de aanwezige markers in de top. Hiervoor heb ik per top via Matplotlib-Pyplot een cirkeldiagram [6] gegenereerd die deze informatie duidelijk weergeeft.
Op dit punt had ik echter gemerkt dat de NetworkX-bibliotheek niet aanwezig was in de oplijsting van toegelaten bibliotheken. Daarom heb ik voor de zekerheid dezelfde boom en plots

geconstrueerd via een andere bibliotheek die zeker toegestaan was, namelijk IGraph [7]. Aangezien deze gelijkaardig werkte, ging de implementatie ervan nog relatief vlot.

Al dit werk heb ik verricht voordat de voorbeeldoplossing online is gekomen. Erna heb ik slechts enkele details vergeleken en bij mijn eigen implementatie zeer weinig aangepast. Ook voor de metaclustering had ik op voorhand al mogelijke clusteringmethodes opgezocht en beslist welke ik ging gebruiken, namelijk AgglomerativeClustering van de sklearn-bibliotheek [8]. Aangezien de voorbeeldoplossing ook deze methode gebruikte, gaf dit mij een extra verzekering dat dit een goede keuze was.
Een aspect waarbij ik mij wel op de voorbeeldoplossing gebaseerd heb, is het gebruik van AnnData. De structuur ervan begreep ik nog niet volledig en het voorbeeld heeft mij hier veel bij geholpen. Ik heb hiervan vooral het gebruik van het "uns"-veld overgenomen om belangrijke informatie gemakkelijk bij te houden. Op die manier kan ik in het project volledig met AnnData werken en zet ik verkregen input, zoals numpy arrays en fcs-bestanden, automatisch om naar dit datatype.
Het volgende dat moest gebeuren, was de implementatie van de fit- en predict-methodes. Aangezien deze niet uitgelegd stonden in de voorbeeldoplossing heb ik veel tijd gespendeerd in het begrijpen wat er exact gedaan moest worden. Ik wist immers niet in welke functie de SOM, MST en metaclustering uitgevoerd moesten worden. Na een tijdje uitzoeken heb ik de fit- en predictmethode correct kunnen implementeren.

Optimalisaties

Nadat ik alles geïmplementeerd had, was het tijd om de accuraatheid ervan na te kijken via de v-measure score. Hierbij merkte ik echter dat mijn score zeer laag was door het gebruik van MiniSom (rond 30%). Daarom heb ik deze implementatie vervangen door een betere bibliotheek, namelijk Sklearn-som [9]. Bij het meegeven van dezelfde argumenten behaalde deze bibliotheek al een veel hogere score (rond 65%). Vervolgens heb ik verschillende argumenten wat aangepast om zo een hogere score te verkrijgen. De hoogste score die ik uiteindelijk bereikt heb, is ongeveer 82%.
De implementatie van MiniSom heb ik echter nog niet verwijderd. Ook deze score heb ik kunnen verbeteren door argumenten aan te passen, waardoor de accuraatheid verhoogd is naar 52%.
Bij het aanmaken van een FlowSom-object kan de gebruiker zelf kiezen of het object voor de SOM Sklearn-som of MiniSom moet gebruiken. Hierbij is Sklearn-som de standaard optie.
Bovendien heb ik een poging gedaan om Quicksom te implementeren. Het trainen ervan verliep echter zeer traag en ik kwam nog een paar andere problemen tegen, waardoor ik deze bibliotheek toch heb weggelaten.

Gebruiksvriendelijkheid

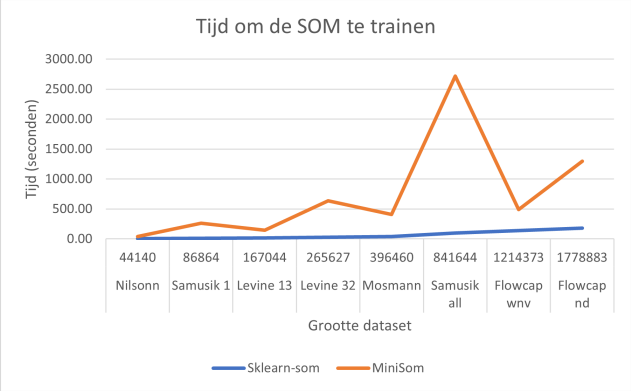
Het laatste dat ik nog wilde verbeteren, was de gebruiksvriendelijkheid. Hierbij heb ik geprobeerd om de visualisaties van de SOM, MST en metaclusters zo duidelijk mogelijk te maken. Om de SOM-clusters te tonen, gebruikte ik eerst cirkeldigrammen, maar heb ik dit later vervangen door een circulaire staafdiagram. Op die manier valt het meer op welke markers het meest aanwezig zijn binnen een cluster. Ook bij de MST heb ik deze wijziging gemaakt bij het tonen van de toppen in het geval dat de boom gegenereerd wordt met NetworkX. In het geval dat IGraph gebruikt wordt, worden wel nog steeds cirkeldiagrammen gebruikt. Dit geeft de gebruiker zelf de keuze om zijn eigen voorkeur te laten visualiseren.
Daarnaast heb ik legendes toegevoegd. Bij de visualisaties van de SOM bevat de legende de kleuren met hun corresponderende markers, terwijl de legendes bij de plots van de metaclusters de kleuren aangeven die horen bij een specifieke metacluster. Bij deze laatste worden de kleuren van de cirkel-/staafdiagrammen dus niet meer verbonden aan de bijhorende marker, aangezien de metaclusters hierbij belangrijker zijn.

Een andere manier om de gebruiksvriendelijkheid te verhogen, is het bereik van mogelijke inputs. De gebruiker kan meerdere datatypes als argument meegeven bij de fit- en predict-methodes, zoals een numpy array, AnnData object, pandas dataframe en een fcs-bestand. Dit heb ik nog uitgebreid door ook het pad naar een map te aanvaarden als input. Het FlowSom object zal dan alle bestanden binnen de map lezen en samenvoegen tot één AnnData object. Op die manier

kan data van meerdere bestanden gecombineerd worden en is het dus bijvoorbeeld niet nodig om elk bestand apart te voorspellen bij de predict-functie, wat de functies veel gebruiksvriendelijker maakt.

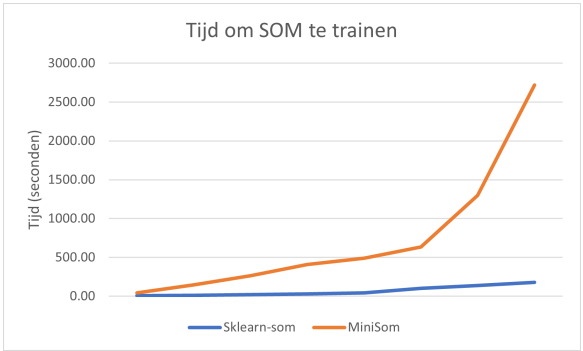
Resultaten

Om de algoritmes van Sklearn-som en MiniSom te vergelijken, zijn ze uitgevoerd op 8 datasets van verschillende groottes, te vinden in de FlowRepository in de map FR-FCM-ZZPH [10]. Hierbij werd enkel het trainen van de self-organising map in rekening gebracht voor de tijd.



Figuur 1: Tijd om SOM te trainen, vergeleken op basis van datasetgrootte

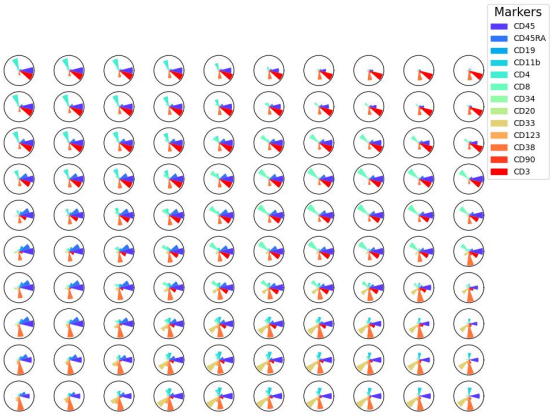
Bij figuur 1 worden de tijden vergeleken op basis van de groottes van de datasets. MiniSom gebruikt een ander aantal iteraties bij het trainen dan Sklearn-som, namelijk de grootte van de dataset vermenigvuldigd met het aantal dimensies (markers). We kunnen zien dat MiniSom dus soms langer traint bij kleinere datasets, terwijl de trainingstijd bij Sklearn-som wel evenredig is aan de datasetgrootte.



Figuur 2: Tijd om SOM te trainen, gesorteerd in oplopende volgorde

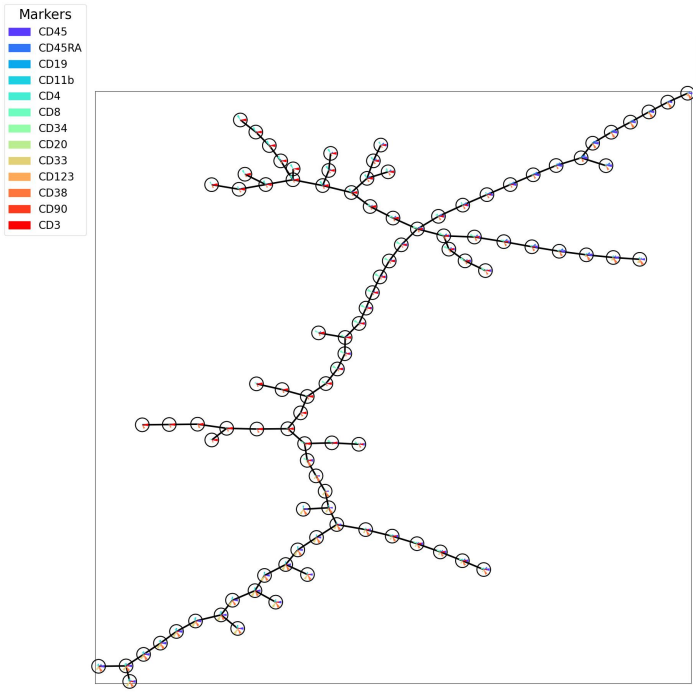
In figuur 2 zijn de tijden gesorteerd in oplopende volgorde. Zo kunnen we makkelijker beide algoritmes vergelijken. De grafiek toont aan dat het algoritme van Sklearn-som veel sneller verloopt dan MiniSom. Hierbij kunnen we ook zien dat MiniSom in het algemeen een exponentiële complexiteit vertoont.

Voor gebruiksvriendelijkheid worden meerdere plots getoond die elk focussen op een ander aspect van het algoritme. Ten eerste worden de SOM-clusters naast elkaar getoond die gedetailleerd tonen welke markers het meest aanwezig zijn per cluster. Hiervoor worden staafdiagrammen gebruikt die omgevormd zijn naar een cirkel. Meer voorkomende markers zullen dus duidelijker zichtbaar zijn op de diagram.

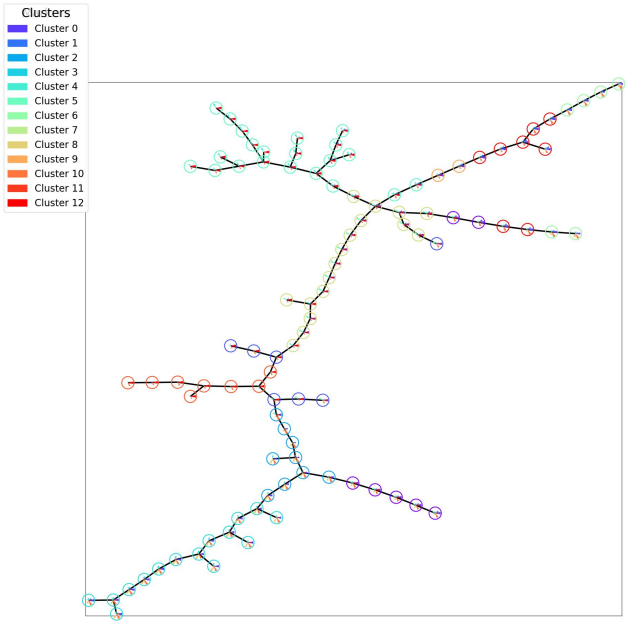


Figuur 3: SOM-clusters

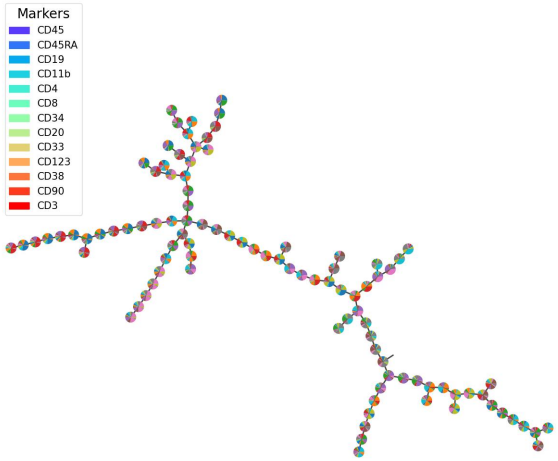
Vervolgens wordt een minimal spanning tree getoond van de SOM-clusters. Op die manier worden relateerde clusters dichter bij elkaar geplaatst en kan de gebruiker hier een structuur terugvinden tussen de SOM-clusters. Bij het gebruik van de NetworkX-bibliotheek worden de toppen voorgesteld met circulaire staafdiagrammen, terwijl deze bij de IGraph-bibliotheek met cirkeldiagrammen getoond worden. Op die manier kan de gebruiker de minimaal opspannende boom via meerdere visualisaties bestuderen, wat de gebruiksvriendelijkheid wat verhoogt.



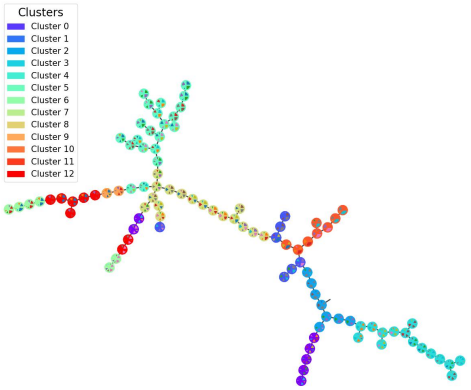
Figuur 4: MST van de SOM-clusters met staafdiagrammen



Figuur 6: MST van de metaclusters met staafdiagrammen



Figuur 5: MST van de SOM-clusters met cirkeldiagrammen



Figuur 7: MST van de metaclusters met cirkeldiagrammen

Voor het construeren van de metaclusters is de AgglomerativeClustering vanuit de Sklearn-bibliotheek een uitstekende keuze. Het trainen neemt minder dan een seconde in beslag en is dus bij alle datasets vergelijkbaar, aangezien het trainen van de SOM veel langer duurt. Bovendien geeft dit algoritme correcte clusters terug en moet het dus niet vergeleken worden met andere mogelijke clusteralgoritmes.

Conclusie

Om snelle en accurate resultaten te krijgen, is het aangeraden om de Sklearn-som bibliotheek te gebruiken. Deze genereert correcte SOM-clusters en heeft hierbij zelfs een hogere trainingssnelheid.

Voor de implementatie van het algoritme (SOM, MST en meta-clustering) heb ik in het algemeen de voorbeeldoplossing niet gebruikt. Hoewel ik hierdoor meer tijd verloren ben voor het begrijpen

van de werking van algoritmes of door het maken van fouten, vond ik dit zelf een betere methode dan te vertrekken van de oplossing. Via mijn methode heb ik immers meer geleerd door het FlowSOM algoritme stap per stap te analyseren en implementeren. Hierdoor begreep ik het beter dan wanneer ik code zou bekeken hebben die mogelijks zelfs nog fouten kon bevatten.

Toekomstig werk

Een aspect waar ik nu te weinig tijd voor had en dus in de toekomst nog had willen doen, is het optimaliseren van MiniSom om deze beter te kunnen vergelijken met Sklearn-som in verband met hun trainingssnelheid. De huidige benchmarks zijn immers momenteel nog niet volledig betrouwbaar, aangezien beide algoritmes nog niet even goed geoptimaliseerd zijn. Pas op het moment dat ongeveer dezelfde accuraatheid verkregen wordt bij beide algoritmes kunnen ze op de correcte manier vergeleken worden op basis van de snelheid. Het is immers mogelijk dat het verbeteren van de juistheid bij MiniSom de snelheid ervan beïnvloedt. Aan de andere kant veronderstel ik dat het optimaliseren van dit algoritme er eerder voor zal zorgen dat het trainen langer zal duren. De tijd voor het trainen van MiniSom zal dus volgens mij afnemen naarmate de accuraatheid ervan toeneemt. We zullen hierbij dus waarschijnlijk nog altijd ondervinden dat Sklearn-som een sneller en beter algoritme is voor het construeren en trainen van een self-organising map dan de MiniSom-bibliotheek.

Wat in de toekomst altijd verbeterd kan worden, is de gebruiksvriendelijkheid. Dit omvat zowel de snelheid van het FlowSOM algoritme als de verkregen plots. Om de snelheid te verhogen, kunnen niet alleen verscheidene bibliotheken onderzocht worden, maar kan ook nog nagedacht worden over de programmeertaal zelf die gebruikt wordt. Python is immers gekend om zijn tragere uitvoeringstijd, terwijl talen zoals C en Rust veel sneller kunnen zijn. Daarom kan het zeker interessant zijn om het FlowSOM algoritme ook eens in andere programmeertalen te implementeren. Hierbij is het zelfs mogelijk om dit slechts voor een deel van het algoritme te doen, zoals bijvoorbeeld het maken en trainen van de self-organising map. Het genereren van de metaclusters verloopt immers veel sneller waardoor deze niet noodzakelijk in een snellere programmeertaal geïmplementeerd moet worden.

Referenties

[1] Saeys Lab. *FlowSOM*. URL: https://github.com/saeyslab/FlowSOM (zie pag. 1).

[2] Sofie Van Gassen. *FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data*. URL: https://onlinelibrary.wiley.com/doi/10.1002/cyto.a.22625 (zie pag. 1).

[3] Giuseppe Vettigli. *MiniSom*. URL: https://github.com/JustGlowing/minisom (zie pag. 1).

[4] *NetworkX: Network Analysis in Python*. URL: https://networkx.org/ (zie pag. 1).

[5] *PyGraphviz documentation*. URL: https://pygraphviz.github.io/documentation/latest/ (zie pag. 1).

[6] URL: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.pie.html (zie pag. 1).

[7] URL: https://python.igraph.org/en/stable/ (zie pag. 2).

[8] URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html (zie pag. 2).

[9] *Sklearn-som v1.1.1 Master Documentation*. URL: https://sklearn-som.readthedocs.io/en/latest/ (zie pag. 2).

[10] *Datasets van Flow*. URL: https://dl01.irc.ugent.be/flow/ (zie pag. 3).