

Implementing unsupervised machine learning algorithms in STOQS (The Spatial Temporal Oceanographic Query System)

Rachel Kahn, Scripps College

Mentor: Mike McCann

Summer 2017

Keywords: machine learning, data mining, oceanography, clustering, Python, Scikit-learn

ABSTRACT

The Monterey Bay Aquarium Research Institute (MBARI) deploys multiple platforms to collect oceanographic data during observational field campaigns. The advancement of technology facilitates the collection of increasingly large and complex data, varying both spatially and temporally. The Spatial Temporal Oceanographic Query System (STOQS) was developed to address the need to efficiently access and visualize MBARI's oceanographic databases, however the issue remains of how to efficiently identify and extract data patterns that may be of interest to scientists. Previous work found that certain types of phytoplankton could potentially be classified by their optical properties via supervised machine learning using salinity as the discriminator for the initial step of labeling the data. This paper describes the first use of unsupervised machine learning using STOQS data. A script was added to the STOQS GitHub repository containing methods for running clustering algorithms on oceanographic data and saving cluster labels back to the database so that they may be viewed in the STOQS user interface. This work demonstrates the potential of unsupervised machine learning to aid in finding and understanding patterns in data, and will serve as a starting point for future efforts to implement unsupervised machine learning to automatically identify features of interest. Above all, this work is the product of my journey transitioning from being a scientist to being a software engineer and a contributor to the STOQS project.

INTRODUCTION

As the development of new ocean technology allows for more oceanographic data acquisition, it also raises the need for tools to process such data. The Monterey Bay Aquarium Research Institute (MBARI) employs ships, moorings, drifters, wave gliders, and autonomous underwater vehicles (AUVs) to collect oceanographic data during observational campaigns. These platforms collect data that vary both spatially and temporally, making visualization and analysis difficult. To address the need to efficiently access, visualize, and analyze the data collected during multi-platform observational campaigns, MBARI created the Spatial Temporal Oceanographic Query System (STOQS). As a completely open-source software package, STOQS uses geospatial relational database technology to provide a means for efficient access and visualization of complex data sets. The STOQS code repository is available on GitHub (https://github.com/stoqs/stoqs), and the public user interface can be found on the MBARI website (Fig. 1). McCann et al. (2014) provide an overview of the structure and capabilities of STOQS (Fig. 2).



Figure 1 Screenshot of the MBARI webpage from which the public STOQS user interface and GitHub repository may be accessed. Available from: http://www.mbari.org/products/data-repository/upper-ocean-data/spatial-temporal-oceanographic-query-system-stoqs-data/



Figure 2 Screenshot of the STOQS user interface. The glider Slocum_260 has been selected for data visualization between 09:01:08 and 13:02:15 on 23 September 2013, during MBARI's September 2013 CANON observational campaign. The glider's position is mapped in the 'Spatial' plot, salinity is shown via color scale in the 'Temporal' plot of depth vs. time, and fluorescence vs. optical backscatter is plotted using salinity as the color scale in the 'Parameter-Parameter' plot.

An investigation of harmful algal bloom ecology in Florida suggests that toxic blooms could potentially be classified by the ratio of chlorophyll fluorescence to optical backscattering (Cannizzaro et al. 2009). McCann et al. (2014) discuss the possibility of machine learning as a tool for recognizing such patterns in observational data. A preliminary use of supervised machine learning labeled data by phytoplankton type based on optical properties, using salinity as the discriminator (Fig. 3; McCann et al. 2014).



Figure 3 Screenshot of the STOQS user interface. Parameter-Parameter plot shows fluorescence vs. optical backscatter measurements using salinity as the color scale for September 2013 CANON data from the AUV dorado, from 17:13:47 on 18 September 2013 to 04:49:15 on 20 September 2013. Initial steps of supervised machine learning have been used to label data by the plankton types diatom, dino[flagellate]1, dino[flagellate]2, and sediment, as shown by the labels in the 'Attributes' section of the user interface.

Because supervised machine learning requires a user to "teach" STOQS software to "understand" the data, the use of unsupervised machine learning could provide a means to recognize patterns in the data without the difficult step of expertly labeling the data. This paper describes the beginnings of unsupervised machine learning implementation in STOQS to classify data. This technique could eventually allow STOQS to autonomously identify features that might be of interest to scientists, without the need of scientists to sort through the data manually.

PROCESS AND RESULTS

MY BACKGROUND

I joined the STOQS project with limited existing background in software and no background in machine learning, but a strong willingness to learn. Having just completed a physics degree with a minor in biology, I was well trained in the scientific process but had no engineering experience. I took one introductory programming course during my undergraduate career, which used Python, and had previous research experience in mathematical modeling and computational fluid dynamics. Upon arriving at MBARI, I had to relearn how to program in Python in addition to learning the basics of machine learning and the workflow of GitHub.

SETTING UP A PERSONAL STOQS DEVELOPMENT SYSTEM

All work was conducted on a 2011 Macbook Pro running OS X El Capitan. In order to run a local STOQS server, it was necessary to set up my own STOQS development system in a virtual Linux machine. Instructions for doing so are found in the README.md document in the STOQS GitHub repository. Due to the large file sizes of the databases, I increased the hard disk space of my virtual machine to 40 GB.

For programming and debugging, I used the integrated development environment PyCharm. PyCharm can be configured to work in a virtual machine via Vagrant, and configuration instructions for can be found in the STOQS GitHub repository under stoqs/doc/instructions/PYCHARM.md.

Once my personal STOQS development system was configured, it was possible to launch STOQS on my local server and load oceanographic observational data by campaign into the server. The data could then be accessed and visualized in my local STOQS user interface. For data loading instructions, see stoqs/doc/instructions/LOADING.md in the STOQS GitHub repository.

Getting to this point took the first half of my internship (except for one week when I was away at sea), as there were many obstacles along the way. The

website Stack Overflow (2017), became instrumental for looking up solutions for errors, and for instructions for workarounds (such as how to increase the hard disk space of my virtual machine).

IMPLEMENTING UNSUPERVISED MACHINE LEARNING

The initial implementation of unsupervised machine learning algorithms in STOQS can be found in the stoqs/stoqs/contrib/analysis/cluster.py script in the 'python3' branch of the STOQS GitHub repository. All code was written in Python 3, and clustering algorithms were used from the Scikit-learn Python module (Pedregosa et al. 2011).

A Python class called Clusterer was created to contain all basic methods pertaining to unsupervised machine learning. Given a platform, two parameters, and a time interval for step size, methods from the Clusterer class were written to identify clusters in the data (VanderPlas 2017), identify clusters sequentially as the method steps through the data at the specified step size, save cluster labels to the database (Fig. 4), and remove cluster labels from the database. The user can also specify one of four clustering algorithms to use (Table 1).

MBARI stoqs_september/2013 Campaign list. Share this view ST0005	250 Sep 13 Sep 16 Sep 19 Sep 22 Sep 25 Sep 28 Oct 1 Time (gmt)
Pictic Cane Session	Measured Parameters Green Selection Parameter Values Green Selection Attributes Green Selection
Metadata: about 4,833,335 data values - 1.594 seconds Parameter-Parameter Come Salecton Measured Parameter Data Access	Measurement Sample Campaign Platform Filter by labeled Measured Parameter (co-located measurements remain in selection) Clustered DBSCAM-Slocum_280 Source selection) SOURCe stop/rottl/slat/slat/slater.pysaveClustersRegdatabase stog, september291platform filter_240signet epital_backsecture?06m filterseeseesealgorithm DBCMclusteredRequiptere BSGAM-filtors_04og.net_comailseinterval house1etep minuteer30start 20109101195930 - ed 3011011(153321 -+)
	A Automated DBSCAN classification of 582/24 datavalues from 2 features Automated DBSCAN classification of 81596 datavalues from 2 features Automated DBSCAN classification of 5880 datavalues from 2 features Automated DBSCAN classification of 380 datavalues from 2 features Automated DBSCAN classification of 380 datavalues from 2 features E Automated DBSCAN classification of 380 datavalues from 2 features
	Automated DBSCAN classification of 66 datavalues from 2 features Automated DBSCAN classification of 12 datavalues from 2 features OUTLER Automated DBSCAN classification of 2788 datavalues from 2 features

Figure 4 Screenshot of the STOQS user interface after running the DBSCAN clustering algorithm on September 2013 CANON fluorescence vs. optical backscatter data from the glider Slocum_260, from

09:39:23 to 13:07:00 on 23 September 2013. Eight clusters (labeled 'A' through 'G' and 'OUTLIER') were identified by the clustering algorithm and saved back to the database, and appear in the 'Attributes' section in the STOQS user interface.

Name	Description
Hierarchical clustering	Identifies clusters based on distance connectivity (Anon 2016)
Density-based spatial clustering of applications with noise (DBSCAN)	Identifies clusters based on data point density (Anon 2016)
Mean Shift	Identifies clusters based on centroids (the means of the points within a region); finds data "blobs" (Anon 2016)
Balanced iterative reducing and clustering using hierarchies (Birch)	Identifies clusters using a Clustering Feature Tree to recursively create subclusters by adding each sample to the subcluster with the closest centroid (Anon 2016)

 Table 1 Scikit-learn clustering algorithms available for Clusterer clustering methods

A side-by-side comparison of the results of the four algorithms (Fig. 5) is included in the Jupyter notebook stoqs/stoqs/contrib/notebooks/compare_clustering_algorithms.ipynb in the STOQS GitHub repository.



Figure 5 Results of clustering algorithms run on STOQS fluorescence vs. optical backscatter data from the glider Slocum_260 from 09:40:38 to 13:06:13 on 23 September 2013. Clusters identified by the algorithms (a) Hierarchical Clustering, (b) DBSCAN, (c) Mean Shift, and (d) Birch are indicated by color.

My process for creating the clustering methods involved learning about relational databases and some basic Structured Query Language (SQL) commands. The culmination of this work was a time-lapse movie showing the clusters over time (Fig. 6).



Figure 6 One frame from the time-lapse movie showing results of the clustering algorithm run on STOQS data over time. Subplots include (a) time-series depth measurements, (b) a map of the glider's location, and (c) results of the clustering algorithm DBSCAN run on September 2013 CANON Slocum_260 fluorescence vs. optical backscatter data with cluster 'A' shown in blue and cluster 'B' shown in orange. The command used to generate the plot is included in the bottom left.

GITHUB WORKFLOW

Due to STOQS being an open-source software package, becoming a contributor involved using GitHub. I worked within my own personal fork from the upstream STOQS repository, which was stored locally on my virtual machine. Each time I added a new method or made edits to my clustering script, I committed the changes to my local repository. Then, I pushed my changes to GitHub, where they appeared online in my forked STOQS repository for anyone to see. When the owner of the upstream STOQS repository pushed changes, I would then pull those changes into my own repository to maintain up-to-date code. Once my clustering script was near complete, I submitted a pull request, which opened up a dialogue between me, the owner, and other contributors to discuss the code I proposed to add to the upstream STOQS repository. The owner then refined my code and integrated it into the upstream repository so that other contributors may pull my script into their own forks.

Because my work is just the first step in implementing unsupervised machine learning in STOQS, the owner opened an issue under which contributors may discuss ways to further develop the code and add more unsupervised machine learning functionality in STOQS. This collaborative process is a hallmark of open-source software development.

CONCLUSIONS

Since initial unsupervised machine learning abilities have now been integrated into STOQS, developers can continue to work toward the goal of autodetection of data patterns. My work demonstrates the potential utility of unsupervised machine learning in STOQS, and will serve as a starting point for future unsupervised machine learning implementations. Next steps could involve training STOQS to track data clusters over time, or identifying thresholds for which a set of data is of interest to scientists and training STOQS to extract these data. The latter would require consultation with scientists who use STOQS. As data collection methods and computational abilities continue to advance, machine learning will provide a useful tool for mining data at large scales. We are just scratching the surface of how machine learning could be used in STOQS to help us learn more from data.

ACKNOWLEDGEMENTS

Thank you to Mike for his mentorship, support, and willingness to both share his knowledge of software with me and to learn about machine learning along with me. Additional thanks to Danelle Cline for getting me set up in PyCharm, George Matsumoto & Linda Kuhnz for their support throughout the internship program, and to the rest of the MBARI staff and interns that I had the pleasure of meeting. This experience would not have been possible without the support of MBARI through funding from the David and Lucile Packard Foundation.

References:

- Anon (2016). API Reference. Scikit-learn: Machine Learning in Python. Available from: http://scikitlearn.org/stable/modules/classes.html#module-sklearn.cluster
- Cannizzaro, J. P., C. Hu, D. C. English, K. L. Carder, C. A. Heil, and F. E. Müller-Karger (2009). Detection of *Karenia brevis* blooms on the west Florida shelf using in situ backscattering and fluorescence data. *Harmful Algae*, 8: 898-909.
- McCann, M., R. Schramm, D. Cline, R. Mchisaki, J. Harvey, and J. Ryan (2014). Using STOQS (the spatial temporal oceanographic query system) to manage, visualize, and understand AUV, glider, and mooring data. *Autonomous Underwater Vehicles (AUV)*, 2014 IEEE/OES: 1-10. https://doi.org/10.1109/AUV.2014.7054414
- Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Duborg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Skikitlearn: machine learning in Python. *Journal of Machine Learning Research*, 12: 2825-2830.
- Anon (2017). Stack Overflow. *Stack Overflow*. Available from: https://stackoverflow.com
- VanderPlas, J. (2017). Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media, Inc., Sebastopol, CA, 331-515.