

**Research Review**

**HyperStream Plugin and Tool Development**

**Supervisors: Dr. Tom Diethe & Professor Peter Flach**

**Author: Yuefeng Li 1619847**

**ACKNOWLEDGEMENTS**

Acknowledgement to my University and those peoples:

* University of Bristol: Affording me the unimaginable opportunity to study here.
* Dr. Tom Diethe and Professor Peter Flach: My project supervisors, support me to write the review of my research project and guide direction for my project.
* Dr. Oliver Ray: Research Skills and MSc Projects Director, for supporting me about how to do the research for my personal individual project.

Contents

[1.Executive Summary 4](#_Toc487378291)

[1.1 Project Type 4](#_Toc487378292)

[1.2 Aims & objectives 4](#_Toc487378293)

[1.3 Deliverables 4](#_Toc487378294)

[2.Introduction 5](#_Toc487378295)

[2.1 Background & Motivations 5](#_Toc487378296)

[2.2 Research Areas 5](#_Toc487378297)

[2.3 Added Value 5](#_Toc487378298)

[2.3.1 The significance of the project: 5](#_Toc487378299)

[2.3.2 What is the project not achieving: 6](#_Toc487378300)

[3. Survey 7](#_Toc487378301)

[3.1 Research area for Time-series Data: 7](#_Toc487378302)

[3.2 Research area for Stream Processing 8](#_Toc487378303)

[3.2.1 Existing stream processing software: 9](#_Toc487378304)

[3.3 HyperStream 10](#_Toc487378305)

[3.4 Online (machine) learning: 13](#_Toc487378306)

[3.4.1 Online (machine) learning is a specific task for time-series data: 14](#_Toc487378307)

[3.4.2 Some typical approaches of online machine learning: 15](#_Toc487378308)

[3.4.3 The Perceptron Training Algorithm: 16](#_Toc487378309)

[3.4.4 The online passive-aggressive algorithm: 18](#_Toc487378310)

[3.5 Research of some typical time-series data resources: 19](#_Toc487378311)

[3.5.1 APIs (Public): 19](#_Toc487378312)

[3.5.2 Statistic datasets: 19](#_Toc487378313)

[4. Project Planning 20](#_Toc487378314)

[4.1 Work Plan 20](#_Toc487378315)

[4.1.1 Expected time-line: 20](#_Toc487378316)

[4.1.2 Gantt Chart of Work Plan: 22](#_Toc487378317)

[4.2 Risk analysis and Contingency Plans 22](#_Toc487378318)

[5.Conclusion 25](#_Toc487378319)

[6. Bibliography: 26](#_Toc487378320)

# 1.Executive Summary

## 1.1 Project Type

Project Type: Total to be 100%. I = Software/hardware development 70%; II = Investigatory, research

20%; III = Theoretical, models/concepts 10%.

## 1.2 Aims & objectives

This project aims to create a plugin for the stream processing software “HyperStream” for the purpose of Online (Machine) Learning, and demonstrate its application to some typical time-series data.

The main objectives of this project are the following:

1. To develop a plugin for HyperStream.
2. To implement several popular online (machine) learning algorithms in python programming language.
3. To develop a tool for importing data into HyperStream.
4. To incorporate the implemented tools and algorithms into the HyperStream plugin.
5. To describe the code in the form of comprehensive documentation.
6. To develop unit tests to ensure the validity of the algorithms.
7. To write a report describing all of the above.

## 1.3 Deliverables

1. The source code and accompanying documentation for the online (machine) learning plugin.
2. The source code and accompanying documentation for the tool which can import data into HyperStream.
3. The source code and accompanying documentation for the tool which implements the perceptron algorithm.
4. The display results of the data.
5. The source code and accompanying documentation for tools which implement other online (machine) learning algorithms. (If time permits)
6. Poster and Thesis.

# 2.Introduction

## 2.1 Background & Motivations

Nowadays data science has been a popular object which is widely used in our daily life. Data scientists can use many tools to do data processing in real life. However, although there are considerable tools for data scientists to use, there are still some problems in some special institutions, such as the SPHERE lab [32] in University of Bristol cannot be resolved. HyperStream software can do stream processing support for SPHERE lab. In this situation, this project aims to create a plugin with (a) tool(s) for the stream processing software “HyperStream” for the purpose of Online (Machine) Learning, and demonstrate its application to some typical time-series data.

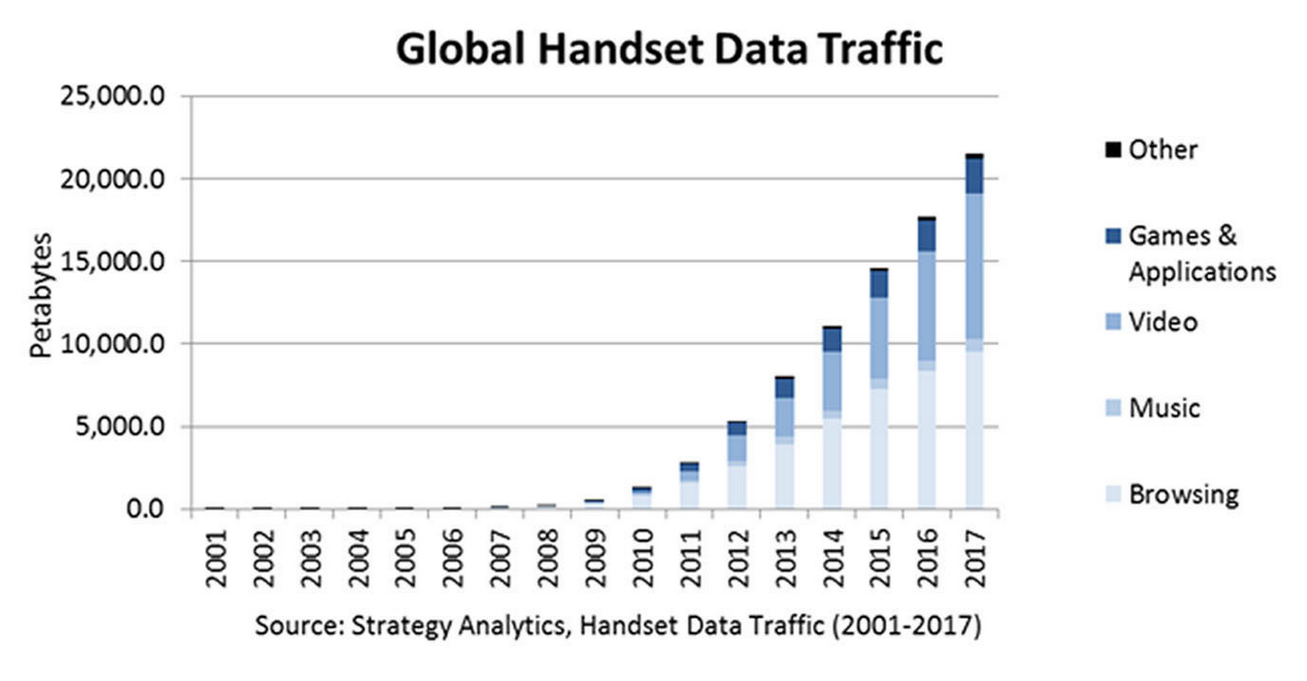
## 2.2 Research Areas

To achieve this goal, first need to learn the background knowledge such as what is time-series data and what is stream processing, the structure of HyperStream and how it works. Then search papers of online (machine) learning algorithms and learn how those algorithms works. Additionally, need to search some typical time-series data resources for the project. Then it would be easier to develop plugin and tools for HyperStream.

## 2.3 Added Value

### 2.3.1 The significance of the project:

* Streaming data is ever increasing. The graph below shows the dramatically increase of streaming data of mobile phones from 2001 to 2017 [33]. As the amount of data increases, we can do more research such as the relationship between the data through analyzing. The analysis of the data requires more efficient tools.



*Figure 1: An example of increasing streaming data. Figure from [33]*

* IID (identically and independently distributed) assumption (characteristics of data, will explain in survey) often taken in machine learning is often violated. Hence the need for online (machine) learning is necessary.
* HyperStream is an attractive solution to these problems as it is designed to be domain independent and easily extensible.

### 2.3.2 What is the project not achieving:

* Not going to introduce fundamentally novel online (machine) learning algorithms. The introductions of online (machine) learning algorithms can be found on the internet.
* Not promising to produce the best results on a given dataset. As this project aims to create plugin and tools for HyperStream.
* Not going to implement all existing online learning algorithms.

# 3. Survey

## 3.1 Research area for Time-series Data:

Time-series data is a series of data points which evolving over time. They are listed or graphed in time order. The most common is that the time-series is a sequence that is taken at successive time intervals.

This occurs in many different applications, such as the number of sunspots, the daily value of the Dow Jones Industrial Average, the heights of ocean tides, the Oxygen isotope levels, the Darwin sea level pressures, etc. [1]

Here discussed the common tasks that time-series involved.

Forecasting: Forecasting is always associated with a time dimension in the future, like estimation for some specific future duration or over a period of time [2]. For example, electricity consumption each month for the next 12 months. With time-series data, it can be forecasted through online learning. Forecasting model is built for aggregate behaviors, in Credit Card spend scenarios, how much spend a customer will have can be forecasted [3].

Prediction: Prediction is an estimation of any event happening, in the past, present or future [4]. For example, the percentage of house owners buying home insurance. Typically, a predicative model is focused on a behavior of objects or customers within a specified time such as performance window. For example, a credit scorecard predicts likelihood of customer defaulting within 12 months [5].

Data mining: This is the process of searching for hidden information from a large number of data through algorithms [6]. Measurements are performed over time in most of the scientific field. Those results lead to a collection of organized data which called time-series [7]. The main purpose of time series data mining is trying to extract most meaningful knowledge from the data. This is a complicated problem for computers, although mankind already has the ability to carry out these tasks [8].

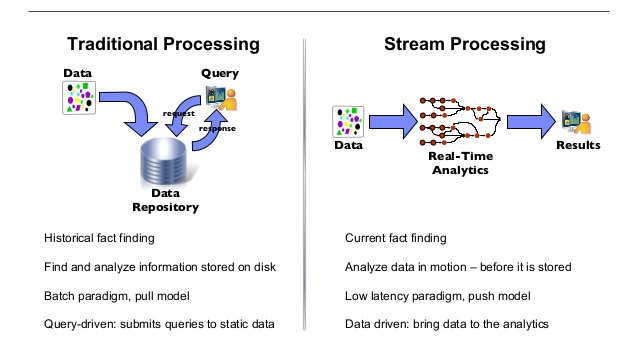
Outlier detection (Anomaly detection): In the production and life, due to the equipment error or man-made operation, the result will inevitably be wrong. At the same time, checking the error for people is a very trivial thing [9]. The operations can be represented by a group of time-series data, using online machine learning to detect outliers can keep people out of trouble. Starting with the labeled data (known whether the data is abnormal), find out some of the normal data as a training set, the remaining normal data and abnormal data as a cross-test set and test set. Using algorithms to do the detection [10].

Time series classification: Time-series classification (TSC) problems including train a classifier on a set of cases, each of which includes a class label and the a real-valued attribute of the ordered set [11]. Time series classification problems can be found in a wide range of areas, including chemometric, machine learning, computational biology, data mining, environmental science, statistics, etc. [12].

## 3.2 Research area for Stream Processing

Stream Processing is a data processing system engine designed to handle infinite data sets. Which means that a steady stream of data flows through the system, the system can continue to calculate [35].

As mentioned in time-series part, data in HyperStream is generally in the form of unbounded data. Unbounded data is a continuous generation, essentially an infinite data set. It is often referred to as “stream data” as well as time-series data. In reality, the essential difference between it and ordinary data is whether it is limited [34]. Such as the daily value of the Dow Jones Industrial Average and the heights of ocean tides mentioned above.

*Figure 2: Differences between traditional processing and stream processing. Figure from [54]*

Stream processing can do many tasks such as combining different streams of data together to create new streams of data, or take a single stream of data and split it into multiple streams [36].

### 3.2.1 Existing stream processing software:

Here are many existing stream processing softwares with their name, license, dependencies and goals:

|  |  |  |  |
| --- | --- | --- | --- |
| Software Name | License | Dependencies | Goals |
| Apache Spark Streaming | Apache | Scala, Java, and Python | Apache spark streaming can let users to build scalable fault-tolerant streaming applications easily [38]. |
| Apache Storm | Apache | Storm development environment, Java,SQL | Storm can let users to reliably process unbounded streams of data and do for real-time stream processing that Hadoop did for batch processing easily [39]. |
| IBM Streams | Apache | IBM SDK Java Technology Edition V7 for x86/POWER7 architecture hardware | IBM streams is an advanced stream processing platform which can ingest, filter, analyze and correlate massive volumes of continuous data streams, it can help users make decisions when things are happening [40]. |

*Table 1: Some examples of existing stream processing software (except HyperStream)*

Those softwares have the same characteristics, which are heavyweight instead, have multiple dependencies, have steep learning curve, and have online mode only. Those characteristics are different from HyperStream (lightweight, have simple dependence, easier to learn and have both online and offline mode), which means they are not optimal for my project.

## 3.3 HyperStream

HyperStream is a stream processing software which has been designed with some key characteristics.

* **HyperStream can operate in online and offline mode:**

In offline mode HyperStream behaves more like traditional data packages (such as pandas), working on pre-existing data (i.e. historical time-series data).

In online mode HyperStream operates continuously, processing streaming data as it arrives. HyperStream has been designed so that the same workflows can be used in both online and offline mode.

* **HyperStream stores the history of computation, so that the same stream won’t be computed twice:**

When users do the data processing of a particular area such as the daily value of the Dow Jones Industrial Average, the computation results will be stored in the database of HyperStream. Then a user tries to do the same computation of the same data, HyperStream will show the computation results instead of doing a new computation.

* **HyperStream designed to be lightweight:**

Lightweight is a software approach. Software approach is a set of rules and conventions for writing computer programs. Lightweight methods have only a few rules and conventions, or that these rules and practices are easy to follow. On the opposite, heavyweight methods have many rules, conventions, and documentation, properly follow them need training and time.

* **HyperStream uses MongoDB as its backend:**

MongoDB is a NoSQL database which its file storage method is based on distributed systems. MongoDB is implemented in C++ (a programming language). The purpose of it is providing a WEB application with extensible high-performance data storage solutions. Its nature is between relational and non-relational databases. It is an abundant method of non-relational database but it is more related to relational databases. It supports many data structures, such as Bson and Json [13], which means the data types it stores can be more complex. The most famous characteristic of MongoDB is that it can support very useful query language. In some parts like the grammar of it shows approximation to the query language which is object-oriented. Similar to almost all the functions of a single-table query can be implemented through MongoDB. It is characterized by high-performance, liable to deploy, liable to use, liable to store data [14]. The main functional characteristics are as follows:

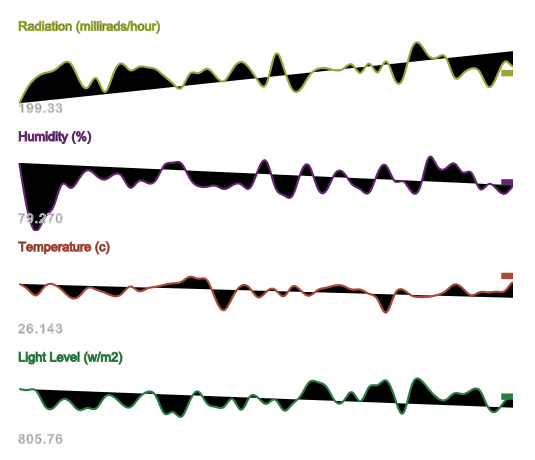
Liable to store the data which is object type for storage collection. Dynamic queries can be supported through schema and it is free. Internal objects can be supported and also the full index. Replication and failover can be supported. Binary data storage is used efficiently, big objects like video and movie are included. In order to support cloud computing’s scalability, using automatic fragmentation. Support programming languages like C#, Java, C++, PHP, python, Ruby, etc. The format of file storage is Bson as mentioned above. Can be accessed through the internet [15]. MongoDB can be used in the following scenes:

First, Website real-time data processing: It is very convenient for timely insert and update. Besides that, it has good scalability and the needed timely data storage site [16]. Second, Cache: The information infrastructure’s buffer layer can be constructed by cache because of its high-performance. The persistent cache layer constructed by the cache avoids overloading the underlying data when the system reboots. Third, highly scalable scenes: Very convenient for thousands of servers composed of the database, its road map already includes the MapReduce engine. Meanwhile, there are some scenes that MongoDB is not suitable for: A system requires a highly transactional, traditional business intelligence applications and complex cross-document (table) cascading query [17].

In conclusion, MongoDB has many characteristics like fast and atomic inserts, it is the first choice for time-series data (timely insertions, updates, and queries). It has the timely data storage needed for site copy and good scalability. HyperStream use MongoDB as it backend is the best choice for now.

* **HyperStream works on two levels:**

Stream level: Stream level works as taking (a) stream(s), using a tool to process the stream(s) and produce (a) new stream(s). Here is a great example of standard real-time data stream for sensor network:



*Figure 3: An example of standard stream. Figure from [55]*

Workflow level: workflows define a graph of computations, linking nodes together by factors. Here the definition of node is that a node is a collection of streams. Streams are objects that use a particular tool for computation, with fixed parameters and filters defined that can reduce the amount of data that needs to be read from the database. The steam can be physically manifested in memory, in the database such as MongoDB (mentioned above), or files, for the time ranges that it has been computed on. Here the definition of factor is that the factor defines the operation of a tool on the node. Tools are the computation elements. They take input data in a standard format and output data in the same format. Meanwhile those tools are version controlled. And they are mostly python classes, other programming languages like Java and C++ can be used with a small amount of wrapper code. Usually, the first node will contain special ‘raw’ streams that pulls in data from a custom data source such as files or database. Workflows can have multiple time ranges, which will cause the streams to be computed on all of the ranges given. HyperSteram supports a plugin system, whereby a set of custom tools and other utilities can be added to the system.

This project including write tools implement online machine learning algorithms such as perceptron algorithm.

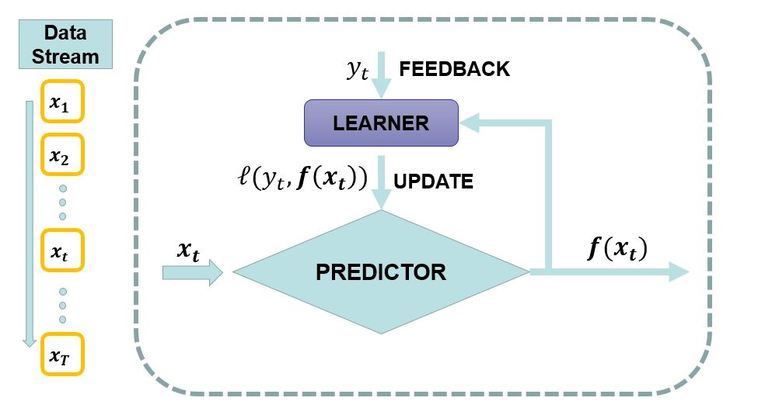
* **Plugins:**

HyperStream has been designed with a simple plugin architecture that lets users extend its core capabilities. These take the form of a python package that is placed in a subdirectory that HyperStream can access, with a small modification to the HyperStream configuration file. Plugins can contain channels, tools, assets, utility functions and data.

This project including create a plugin for online machine learning. With this plugin, users can do the time-series data processing easily through online machine learning algorithms (tools mentioned above).

## 3.4 Online (machine) learning:

Online (machine) learning is defined as a sequence of data for machine learning methods [18]. Every step of it is used to replace our best estimates of the future, rather than learning the whole training data to get the best prediction [19]. Online (machine) learning is a widely-used method around machine learning field. Most of the time use it is because of the data itself can be generated over time, and there is a situation in which the algorithm must be used to dynamically adapt to the current data pattern [20]. For instance, stock price prediction. Stock price could change each second and the prediction of it needs to be done based on the previous step [21].



*Figure 4: An example of how online (machine) learning works. Figure from [22]*

The process of online (machine) learning: first put the data stream into predictor (model), then show the model prediction results to the user. After that, collect the users’ feedback data to train the predictor (model) again. Then update the prediction results. Finally, a loop system formed.

Online (machine) learning is more like an automatic control system, but they are not the same. Here are the differences: the optimization of the online learning goal is to minimize the loss function of the whole, meanwhile, automatic control system requires minimum deviation of the results and expectations. As for the traditional methods, after the model launching, the update cycle of it will be long, it is stable and will not have any interaction with the condition of the situation. Assuming the forecast is wrong, it can only be completed correction in the next update. Online (machine) learning method is different, it is based on the result of online prediction model of the dynamic adjustment. If there is something wrong with the model prediction, it can make a correction in time. As a result, online (machine) learning can reflect changes in a timely manner. Which means it is very suitable for time-series data [37].

### 3.4.1 Online (machine) learning is a specific task for time-series data:

Online (machine) learning is a specific task that users can do on time-series data, the basic premise is that the data items arrive in a sequential fashion. It is often assumed that the data is not necessarily IID (Identically and independently distributed). Identically means that the order of the data matters. We define IID of a group or a sequence of random variables, only when the probability distribution of those variables is the same and all are mutually independent [56].

Here is a mathematical definition:

Let the random variables be defined to assume values in  [57]

We define two random variables  and  are **identically distributed** if 

We define two random variables  and  are **independent** if 

Where  ,  and  are **events,**  ,and  are the **probability of events.**

### 3.4.2 Some typical approaches of online machine learning:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm Name | Date | Authors | Link to the paper | Application |
| Perceptron [41] | 1957 | Frank Rosenblatt | http://psycnet.apa.org/journals/rev/65/6/386/ | This can predict learning curves from neurological variables. Find the separating hyperplanes of datasets. |
| Online Passive-aggressive algorithm [42] | 2006 | Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer. | http://www.jmlr.org/papers/v7/crammer06a | Margin based, Using for various prediction tasks. |
| Sparse online Gaussian processes [43] | 2002 | Lehel Csato and Manfred Opper | http://www.mitpressjournals.org/doi/abs/10.1162/089976602317250933 | Developed in order to overcome the limitations for large datasets. |
| Hedging algorithm [44] | 1997 | M.A. Howe, B. Rustem | https://cseweb.ucsd.edu/~yfreund/papers/nhedge.pdf | Using for decision-theoretic online learning (DTOL) problems [58]. |

*Table 2: Some online (machine) learning algorithms*

There are many sub-methods related to online learning, such as Recursive least squares: an online method of special problems [23]. Stochastic gradient decent (SGD): which is a S approximation of the GD optimization method tries to find out the minimum and maximum by iteration [24], incremental stochastic gradient descent (ISGD). Progressive learning [26]: this learning method is an effective learning model, and it is demonstrated by the human learning process, the process of it is learning continuously from experience, this technique can learn new labels dynamically when it is running. Online sub-gradient descent (OSD) [27]: this is an iterative method to solve convex minimization problems, even though the function is non-differentiable objective, the sub-gradient method is still convergent. Because of the minimization of two differentiable convex functions, OSD is much slower than the Newton method, but it shows that as for nun-differentiable objective, Newton’s method could not converge [28].

### 3.4.3 The Perceptron Training Algorithm:

The perceptron is inspired by the information processing of a single neural cell called a neuron. Neurons accept input signals through their dendrites, which pass the electrical signal down to their cell body. Meanwhile, training data examples give the perceptron signals which were weighted and combined in a linear equation called the activation. The activation is then transformed into an output value of prediction using transfer function, such as step transfer function. Meanwhile, the weight of the Perceptron algorithm must be estimated from the training data through gradient descent (this process can be used to find the weight set in the model that leads to the minimum error of the model on the training data). The perceptron iterates over the training set, updating the weight vector every time it encounters an incorrectly classified example.

The weight update rule is C:\Users\YUEFENG\AppData\Local\Temp\1499448808(1).png, where C:\Users\YUEFENG\AppData\Local\Temp\1499449127(1).png is the new weight vector, C:\Users\YUEFENG\AppData\Local\Temp\1499449221(1).pngis the weight in the previous step, C:\Users\YUEFENG\AppData\Local\Temp\1499449411(1).png is the learning rate, C:\Users\YUEFENG\AppData\Local\Temp\1499449550(1).png is a misclassified positive (negative) example, and C:\Users\YUEFENG\AppData\Local\Temp\1499449633(1).png is  **+1 (-1)**.

Here is the procedure of training a perceptron for linear classification:

Input: labelled training data **D** in homogeneous coordinates; learning rate C:\Users\YUEFENG\AppData\Local\Temp\1499449411(1).png.

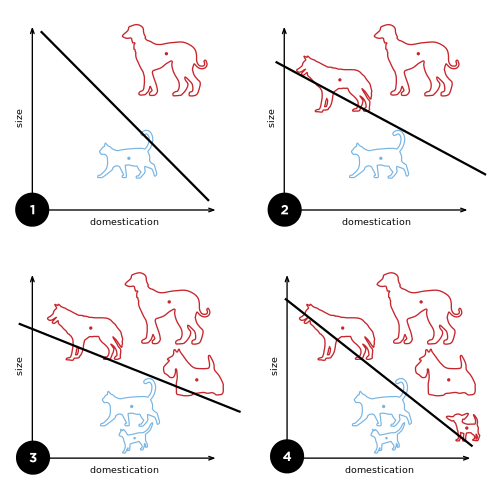
Output: weight vector C:\Users\YUEFENG\AppData\Local\Temp\1499449221(1).png defining classifier C:\Users\YUEFENG\AppData\Local\Temp\1499450863(1).png

Initial C:\Users\YUEFENG\AppData\Local\Temp\1499449221(1).png (weight) to be zero, initial ***converged*** to be ***false***, ‘while’ ***converged*** ***equals to false*** do the following things: make ***converged*** to be ***true,*** and for ***i from 1 to|D| (***a loop does the same thing |D| times***)*** do:

Loop start: if C:\Users\YUEFENG\AppData\Local\Temp\1499451660(1).png, then C:\Users\YUEFENG\AppData\Local\Temp\1499448808(1).png, and here make converged into false. Loop end. Then ‘while’ end.

This perceptron algorithm is iterated over a variety of training samples until all samples are correctly classified. It is easy for us to transform the algorithm into an online algorithm that can handle streaming data: The weight vector is updated only when the most recently received sample is misclassified, comparing to pervious mentioned (The perceptron iterates over the training set, updating the weight vector every time it encounters an incorrectly classified example), this difference is the reason why not everyone thinks of the perceptron algorithm as online ---- it has both online and offline (“batch”) interpretation. The perceptron training algorithm converge when the training data linearly separable, if the data not linearly separable, not necessarily converge [60].

Here is a simple example of how perceptron algorithm working: As the data flow updates, the model is continually trained and the classification plane (linear boundary) is constantly changing.



*Figure 5: A simple example of how perceptron algorithm works. Figure from [59]*

### 3.4.4 The online passive-aggressive algorithm:

Online Passive-Aggressive (PA) algorithm is a margin based online learning algorithm, which is its capability of binary and multiclass categorization [42].

The passive-aggressive algorithm employs an aggressive update strategy by modifying the weight vector by as much as needed to satisfy the constraint imposed by the current example [42].

Imagine that you are listening to the entire twitter every single time and trying to predict something about the tweets. This is the thing that cannot be able to store or keep in memory, then need an algorithm which get an example (tweet), learn from it and throw it away. Here is how this algorithm works: First initializing the weight vector (W) all to 0 and have a 0 for every term in vocabulary. Then look at the stream of data and get a new document (d), it’s a vector over the vocabulary. The do the pre-processing to normalize it to unit length. Normalizing length is critical for the algorithm using here. Then if want to make a prediction, just multiply ‘d transpose’ by weight vector and see if its positive or negative. In the beginning it’s 0, because haven’t learned anything yet. But later on, will get a prediction and sometime later will observe the true class of the document. That’s the reason why it’s either +1 or -1, which means the document is either positive or negative.

Then using the true label to make the weight vector better. Here ‘better’ means: if the document was positive (Y = +1), then we would like to have the dot product bigger than +1; if it’s negative (Y = -1), then we would like the dot product to be smaller than -1. Which means need to have the gap of 1 around the decision boundary line, all the positives to be right and all the negatives to be left. See figure 7.

C:\Users\YUEFENG\AppData\Local\Temp\1499519274(1).png

*Figure 6: Axis of dot product.*

But suppose the document scores on the left side of the decision boundary the prediction is obviously wrong, then need the penalize the decision and change the weight. Suppose the document scores on the left side of the decision boundary the prediction is true, in this case don’t have to change the weight.

This is what need to be enforced. First compute a loss function, let T (temp) = [ ‘d transpose’ multiply W (weight) multiply Y (+1 or -1)] to be over 1 always for all the examples at the beginning. Then if something is smaller than 1, take this as a loss; if something is 1 or bigger than 1, then ignore it. Then got the loss function: L = max (0, 1- T). Finally update the W (weight) through function W(new) = W (now) + YLd (by adding a small proportion of d). Here the new weight vector will correspond to a new decision boundary. Assume the L (loss) is not 0, this means the dot product between the document and the weight vector was “short”. L is the difference between that it should have got and what it did get, which means L is how “short” it fell off the target.

Assume getting a new example and that example wasn’t perfectly classified. If the example is positive and the dot product was smaller than +1, the algorithm moves the weight vector just enough to make the new dot product exactly +1. Which means it moves just enough to be perfect for the example that the algorithm just seen. The passive part is if the dot product was perfect then don’t have to do anything. W (weight) is unconstrained manually, which means with large weights, the loss function can be arbitrarily large and that can lead to certain instabilities, particularly when the data is not nicely separable. So in practice using cap loss function (by introducing a constant C) to solve this: L = min (C, L). When computing the loss function, if L is bigger than C then L will be C.

## 3.5 Research of some typical time-series data resources:

### 3.5.1 APIs (Public):

There are many Open APIs which could be the time-series data resources such as Twitter [49], Facebook [50], Environment agency (UK) [51], BBC Weather [52], Simple Texting [53], etc. [45]. Those open APIs doesn’t need API key to identify users and users do not have to pay it, users can request those APIs easily.

### 3.5.2 Statistic datasets:

Meanwhile, there are many statistic datasets such as Time series classification [46], UCI datasets [47], Data Market [48], etc. These datasets are released with a permissive Authorized Use license, which means that they are readily available for experimentation [31]. In addition, there are existing published results on many of these datasets which makes empirical comparisons easier.

# 4. Project Planning

This part is mainly about the project work plan and the time of tasks which this project included to achieve. This project will begin at the end of June and end in early September.

## 4.1 Work Plan

### 4.1.1 Expected time-line:

**Stage 1 Learn more about Online learning algorithms and HyperStream (7th July)**

Task 1.1: Learn perceptron algorithm and other online learning algorithms (5th July)

* Completion of this project requires a lot of background knowledge. The purpose of this part is to learn more about the online (machine) learning algorithms through searching and reading more papers and to lay the foundation for the online (machine) learning tools.

Task 1.2: Familiar with the structure of HyperStream and how it works (7th July)

* HyperStream is the software developers can update, the online (machine) learning plugin and tools which this project included should run well on HpyerStream. To achieve this, I need to get familiar with the structure of it and need to know how it works.

##### Stage 2 install HyperStream and create a plugin (15th July)

Task 2.1: Install HyperStream use api\_dev branch (9th July)

* After learning the background knowledge, it comes to set up the environment for the creation of plugin and tools. To install HyperStream, first, need to install git, python, python virtual environments, python-pip, MongoDB, and Mqtt broker (e.g. Mosquitto). MongoDB and Mqtt broker can be done with docker (an open source application container engine). Then install HyperStream from github and do some tests to make sure HyperStream and MongoDB can work well on the computer.

Task 2.2: The creation of the online (machine) learning plugin (15th July)

* After installing the HyperStream, the plugin for online (machine) learning could begin. There is a tutorial [30] of how to create a plugin in HyperStream. Follow the instructions and combine the characteristics of online learning to create this plugin.

*Deliverable*: The source code and accompanying documentation for the online (machine) learning plugin.

##### Stage 3 Download data and write a tool to import data into HyperStream (22th July)

Task 3.1: Download an example dataset (17th July)

* Time-series data is necessary for this project. First need to download time-series data from the website such as UCI dataset [29].

Task 3.2: Create a tool to import data into HyperStream (22th July)

* The data download in the previous step need to be imported into HyperStream so that can be used. This task aims to create a new tool to import data into HyperStream.

*Deliverable*: The source code and accompanying documentation for the tool which can import data into HyperStream.

##### Stage 4 Write a tool that implements the perceptron algorithm and display the results (10th August)

Task 4.1: Write a perceptron algorithm tool. (1th August)

* According to the background knowledge of online (machine) learning I learnt in stage 1, a tool can be implemented. This is the hardest part in the whole project.

*Deliverable*: The source code and accompanying documentation for the tool which implements the perceptron algorithm.

Task 4.2: Display results of the data. (6th August)

* After creating the tool, when use the tool to do data processing, the results need to be display, so that users can see the results more intuitive and specific. This part need to do some plotting, either use matplotlib or use the ipython notebook to display high charts plots without installing anything.

*Deliverable*: The display results and accompanying documentation of the data.

Task 4.3: Write tools for other online (machine) learning algorithms. (10th August)

* After finishing the tasks above, if there are still some time before 10th August, other online (machine) learning algorithms tools learnt in stage 1 can be created.

*Deliverable*: The source code and accompanying documentation for tools which implement other online (machine) learning algorithms. (If time permits)

##### Stage 5 Unit tests (12th August)

Task 5.1: Do some unit tests for the plugin and tools (10th August)

* Although the online (machine) learning plugin and tools are created, still need to test the stability and accuracy of the tools. Here use some tests (knowing the outcome should be) to test them.

##### Stage 6 Thesis and poster (28th August)

Task 6.1: Create a poster (16th August)

* The poster will show how the plugin and tools work (such as workflow chart) and the plot of data results and other important creation.

*Deliverable*: Poster

Task 6.2: Write and finish the report of the project (28th August)

* Add more important information and complete the report.

*Deliverable*: Thesis

### 4.1.2 Gantt Chart of Work Plan:

*Figure 7: Gantt Chart of the work plan of the project*

## 4.2 Risk analysis and Contingency Plans

This section will discuss the risks from the perspective of likelihood and severity. And discuss how to prevent and solve.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk** | **Likelihood** | **Severity** | **Prevention** | **Contingency** |
| Mismatch between my version and the main version of HyperStream | 70% | 70% | HyperStream is under continual development. Therefore, it’s easy for my version to become out of sync with the main-line. This is not easy to be prevented because there are other developers update the HyperStream. | Mitigate this by regularly updating. |
| HyperStream doesn’t have the capability to achieve what I want (some complex task) | 50% | 50% | This is hard to prevent because it’s the limit of HyperStream. | Mitigate this by trying to simplify or seek alternative approaches. |
| The development environment in my computer is insufficient. | 50% | 40% | Hard to prevent, only when trying to set up the environment will know. | Mitigate this by using university resources wherever possible. |
| Time constraints and difficulty in achieving the goals in the time given. | 20% | 70% | Study hard without relax during this time. (Have good time management) | Mitigate this by breaking down the large tasks into small achievable tasks and time management. |
| Data loss and hardware damage | 20% | 70% | Take care of my computer and always have a copy. | Mitigate this by using university resources wherever possible. |
| Change personal supervisor at the end of July | 100% | 40% | This may not prevent. But this is a good chance for us to communicate further. | Mitigate this by trying my best to finish the tools and plugin before the change of my supervisor. |
| New supervisor knows less of my project or very busy | 30% | 60% | My supervisor will find a suitable person to be my new supervisor. | Mitigate this by trying my best to finish the tools and plugin before the change of my supervisor. |
| Hard to learn Python or require multiple languages like Java and C++ to compile tools | 30% | 50% | Start the study of Python and other languages now. | Watch the videos online or ask students who are good at Python, Java, and C++. |
| Illness or other personal reasons | 40% | 30% | The timeline mentioned above has 15 days before the actual deadline for dealing with emergencies. | Take care of myself. Complete the tasks ahead of schedule. |

*Table 3: Risk analysis*

# 5.Conclusion

The purpose of this project is to create a plugin for the stream processing software “HyperStream” for the purpose of Online (Machine) Learning, and demonstrate its application to some typical time-series data.

The research review focus on learning the background knowledge of this project including Time-series data (what is time-series data, what applications it occurs, what are the common tasks it involved), Stream processing (what does stream processing means, what can stream processing do, a table of some examples of stream processing software), HyperStream (the key characteristics of it, the layers of it and how it works), Online (machine) learning algorithms (what is online learning, IID assumption, and a table of some existing online learning algorithms), Some typical time-series data resources (open APIs and static datasets).

As the streaming data is increasing every day which means they change over time, this is an important problem for creating plugin and tools. The purpose of this project is not to demonstrate the best results on a particular dataset, but rather demonstrate the whole pipeline working. As there are considerable datasets I cannot test them one by one using different algorithms. The project will not introduce fundamentally novel online learning algorithms and will not implement all existing online learning algorithms, in contrast, this project will implement some typical online learning algorithms such as perceptron algorithm. If there is enough time, I will implement other typical online learning algorithms mentioned above such as Online Passive-aggressive algorithm, Sparse online gaussian processes, and Hedge algorithm.

With the creation of online (machine) learning plugin and tools, HyperStream will become more complete. HyperStream can easily and reliably handle unbounded streams of data for real-time processing, which can greatly facilitate the data scientists in the SPHERE lab.

# 6. Bibliography:

1. Time series introduction. URL: https://en.wikipedia.org/wiki/Time\_series. 2017-7-1.
2. Box, G.E., Jenkins, G.M., Reinsel, G.C. and Ljung, G.M., 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.
3. Brockwell, P.J. and Davis, R.A., 2016. *Introduction to time series and forecasting*. springer.
4. Casdagli, M., 1989. Nonlinear prediction of chaotic time series. *Physica D: Nonlinear Phenomena*, *35*(3), pp.335-356.
5. Brown, R.G., 2004. *Smoothing, forecasting and prediction of discrete time series*. Courier Corporation.
6. Fu, T.C., 2011. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, *24*(1), pp.164-181.
7. Last, M., Kandel, A. and Bunke, H. eds., 2004. *Data mining in time series databases* (Vol. 57). World scientific.
8. Esling, P. and Agon, C., 2012. Time-series data mining. *ACM Computing Surveys (CSUR)*, *45*(1), p.12.
9. Chandola, V., Banerjee, A. and Kumar, V., 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*(3), p.15.
10. Lee, W. and Xiang, D., 2001. Information-theoretic measures for anomaly detection. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on* (pp. 130-143). IEEE.
11. Time series classification. URL: https://www.uea.ac.uk/computing/tsc. 2017-7-1.
12. Wei, L. and Keogh, E., 2006, August. Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 748-753). ACM.
13. MongoDB homepage. URL: https://www.mongodb.com. 2017-7-1.
14. Banker, K., 2011. *MongoDB in action*. Manning Publications Co..
15. Chodorow, K., 2013. *MongoDB: the definitive guide*. " O'Reilly Media, Inc.".
16. Pirtle, M., 2011. *MongoDB for Web Development*. Addison-Wesley.
17. Hawkins, T., Plugge, E. and Membrey, P., *The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing, Apress* (p. 350). ISBN 978-1-4302-3051-9.
18. Introduction of online machine learning. URL: https://en.wikipedia.org/wiki/Online\_machine\_learning. 2017-7-1.
19. Bertsekas, D.P., 2011. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, *2010*(1-38), p.3.
20. L. Rosasco, T. Poggio, Machine Learning: a Regularization Approach, MIT-9.520 Lectures Notes, Manuscript, Dec. 2015. Chapter 7 - Online Learning.
21. Harold, J., Kushner, G. and Yin, G., 1997. Stochastic approximation and recursive algorithm and applications. *Application of Mathematics*, *35*.
22. Online machine learning. URL: http://www.doyensahoo.com/introduction.html. 2017-7-1.
23. Introduction of online machine learning. URL: https://www.analyticsvidhya.com/blog/2015/01/introduction-online-machine-learning-simplified-2. 2017-7-2.
24. Venkatesan, R. and Er, M.J., 2016. A novel progressive learning technique for multi-class classification. *Neurocomputing*, *207*, pp.310-321.
25. Saad, D., 1998. Online algorithms and stochastic approximations. *Online Learning*, *5*.
26. Shawe-Taylor, J. and Cristianini, N., 2004. *Kernel methods for pattern analysis*. Cambridge university press.
27. Kiwiel, K.C., 2006. *Methods of descent for nondifferentiable optimization* (Vol. 1133). Springer.
28. Shor, N.Z., 2012. *Minimization methods for non-differentiable functions* (Vol. 3). Springer Science & Business Media.
29. UCI datasets. URL: https://archive.ics.uci.edu/ml/datasets/Daphnet+Freezing+of+Gait. 2017-7-2.
30. Tutorial of creating plugin in HyperStream. URL: https://github.com/IRC-SPHERE/HyperStream/blob/api\_dev/plugins/data\_importers/tools/csv\_reader/2017-06-20\_v0.0.1.py. 2017-7-2.
31. API license of EBAY. URL: https://go.developer.ebay.com/api-license-agreement. 2017-7-2.
32. SPHERE project homepage. URL: http://www.ircsphere.ac.uk. 2017-7-2.
33. The source of the handset data graph. URL: http://influxis.com/video-streaming-to-lead-the-mobile-data-traffic-growth-of-300-by-2017. 2017-7-2.
34. The introduction of stream processing. URL: http://www.chinacloud.cn/show.aspx?id=24851&cid=17. 2017-7-2.
35. Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P.D. and Klosowski, J.T., 2002. Chromium: a stream-processing framework for interactive rendering on clusters. *ACM transactions on graphics (TOG)*, *21*(3), pp.693-702.
36. Stonebraker, M., Çetintemel, U. and Zdonik, S., 2005. The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, *34*(4), pp.42-47.
37. Diethe, T. and Girolami, M., 2013. Online learning with (multiple) kernels: A review. *Neural computation*, *25*(3), pp.567-625.
38. Home page of Apache Spark Streaming. URL: https://spark.apache.org/streaming/. 2017-7-3.
39. Home page of Apache Storm. URL: http://storm.apache.org/. 2017-7-3.
40. Home page of IBM Streams. URL: https://www.ibm.com/cz-en/marketplace/stream-computing. 2017-7-3.
41. Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), p.386.
42. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S. and Singer, Y., 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, *7*(Mar), pp.551-585.
43. Csató, L. and Opper, M., 2002. Sparse on-line Gaussian processes. *Neural computation*, *14*(3), pp.641-668.
44. Howe, M.A. and Rustem, B., 1997. A robust hedging algorithm. *Journal of Economic Dynamics and Control*, *21*(6), pp.1065-1092.
45. List of open APIs. URL: https://en.wikipedia.org/wiki/List\_of\_open\_APIs. 2017-7-3.
46. Home page of time series classification. URL: http://www.timeseriesclassification.com/. 2017-7-3.
47. Some datasets in UCI datasets. URL: https://archive.ics.uci.edu/ml/datasets.html?format=&task=&att=&area=&numAtt=&numIns=&type=ts&sort=nameUp&view=table. 2017-7-3.
48. Time series data library – Data Market. URL: https://datamarket.com/data/list/?q=provider%3Atsdl. 2017-7-3.
49. API of twitter. URL: https://twitter.com/?lang=en-gb. 2017-7-3.
50. API of facebook. URL: https://developers.facebook.com/. 2017-7-3.
51. API of environment agency (UK). URL: https://www.gov.uk/government/organisations/environment-agency. 2017-7-3.
52. API of BBC weather. URL: http://www.bbc.co.uk/weather/. 2017-7-3.
53. API of simple texting. URL: https://simpletexting.com/api/. 2017-7-3.
54. The source of graph of Differences between traditional processing and stream processing. URL: https://www.slideshare.net/ConfluentInc/kafka-and-stream-processing-taking-analytics-realtime-mike-spicer. 2017-7-3.
55. Example of real-time data stream for sensor network. URL: https://www.pubhub.com/developers/realtime-data-streams/sensor-network/. 2017-7-4.
56. Definition of Identically and independently distributed (IID). URL: https://en.wikipedia.org/wiki/Independent\_and\_identically\_distributed\_random\_variables. 2017-7-5.
57. Gänssler, P. and Stute, W., 1979. Empirical processes: a survey of results for independent and identically distributed random variables. The Annals of Probability, pp.193-243.
58. Chaudhuri, K., Freund, Y. and Hsu, D.J., 2009. A parameter-free hedging algorithm. In Advances in neural information processing systems (pp. 297-305).
59. Figure of how perceptron algorithm works. URL: https://en.wikipedia.org/wiki/Perceptron. 2017-7-7.
60. Flach, P., 2012. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

New part:

由于HyperStream 最初设计是针对以linux 系统为主的 SPHERE 实验室， 并且此软件包还处于完善阶段，因此配置环境时遇到了很多的困难，正如我之前risks里提到的，我的电脑是windows，无法完成配置，因此我使用了虚拟机希望可以在自己电脑上配置，然而测试后又失败了，最终采用ssh lab machine IP 地址 的方法，远程使用学校实验室的一台机器。HyperStream 需要有mongodb的支持，而lab的机器无法使用root权限，只有root之后才可以安装mongodb， 因此我采用使用在线云端mongodb的方法（这里是使用在线云端软件mlab）。

在编译perceptron时，它的测试数据需要time-series data。我使用的测试数据是在UCI dataset里下载的sonar数据 （加上网址）。我拿到的sonar 数据没有time-series序列，并且数据的顺序分类过于明确，（前半部分是一类，后半部分是另外一类）， 因此我利用生成一列新的随机数的方法，将数据的顺序进行打乱，并添加时间序列（time-series）。 我create 的read\_csv\_file tool 可以分别读取csv数据集 文件里的第一列时间序列（eg: 2017-…），中间部分的features，以后最后一列的value。这种分别部分读取而不是整行读取的方式为time-series data而定做，它更加方便用户使用不同部分的数据，方便对数据进行更加细致的操作。

下面来分别讲述我为HyperStream 创造的工具。

1. Csv\_reader：

Csv\_reader是将指定路径的CSV文件通过以“,”隔开的方式读入HyperStream，并以timestamp为key，数据为value的形式展示出来。本文中此tool使用的测试数据是sonar dataset 和 breast-cancer-wisconsin dataset. 这两个数据集的原始形式均是一整个不带有时间序列的数据集，并且数据由features和labels两部分组成，features部分有大量的特征值数据，labels部分则由0和1组成。由labels部分可以看出数据是已经归好类的：即前半部分是1类，后半部分是0类。为了使算法更好的学习以及展示其效能，我采用生成随机数并按大小排列的方法将数据的排列先后顺序进行了随机，并在每行数据前加上时间序列（timestamp），这样做不仅数据的原始分类被打散，也更方便算法进行feature weights的更新，使算法更好的学习，结果更趋近于最优解。由于存在部分时候label不是数字而是字符，因此csv\_reader在设计时采用的存储形式是建立一个dictionary，里面读取的数据是以string的形式存储，这样做无论csv内是阿拉伯数字还是字母或单词都可以使用csv\_reader tool 进行读取，面向对象不仅是处理online learning 的数据，还可以在HyperStream 里任何需要读取csv文件时使用。

1. Perceptron learning algorithm：

Perceptron 算法是模仿单独一层神经网络的效果，可以简单理解为知错就改算法， 每当遇到一个错误的时候就修正算法。并且此算法的输出结果是yes or no。

在测试算法时，我使用了sonar dataset 和 breast-cancer-wisconsin dataset. 对于sonar dataset, 数据的第一列是timestamp，中间部分是feature， 最后一列是特征值 （0/1），使用自己编写的csv\_reader 工具将用“,” 隔开的所有元素都读取进入HyperStream 并展示出来。针对不同的数据列，在perceptron tool 中分别操作。正如上文的csv\_reader tool中提到，由于csv\_reader tool读取数据是以string的类型存储在dictionary中，因此在perceptron tool的开头将string内容全部数字化以方便后续的weight计算和estimate 值计算。

Perceptron算法的初始weight值, 针对不同的数据有不同的初始化方式，我创建的perceptron tool可以选择随机weight 初始化 (0-1之间的随机float数)或者全0初始化。这样做方便用户在使用时进行选择，以达到方便快捷，灵活的特点。经过我大量的测试表明，perceptron算法针对sonar dataset 和 breast-cancer-wisconsin dataset的学习结果对于全0初始化和随机weight初始化的影响并不大。

Sonar data:

Perceptron

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Initial Weight | Accurate while training | Accurate using training weight |
| v0.0.4 | w = 0 | 0.6298 | 0.7596 |
| V0.0.4 | W = random.rand() | 0.6058 | 0.7211 |
| V0.0.4 | W = random.rand() | 0.5961 | 0.7260 |
| V0.0.4 | W = random.rand() | 0.6106 | 0.7644 |
| V0.0.4 | W = random.rand() | 0.5721 | 0.7451 |

Avg\_perceptron (the weight change more stable than normal perceptron)

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Initial weight | Accurate while training | Accurate using training weight |
| V0.0.3 | W = 0 | 0.6298 | 0.7548 |
| V0.0.3 | W = random.rand() | 0.5670 | 0.7452 |
| V0.0.3 | W = random.rand() | 0.6106 | 0.7308 |
| V0.0.3 | W = random.rand() | 0.5961 | 0.7356 |
| V0.0.3 | W = random.rand() | 0.5913 | 0.7644 |

Passive aggressive\_t1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Initial weight | Accurate while training | Accurate using training weight |  |
| V0.0.2 | W = 0 | 0.6298 | 0.6442 |  |
| V0.0.2 | W = random.rand() | 0.6058 | 0.6923 |  |
| V0.0.2 | W = random.rand() | 0.5817 | 0.6875 |  |
| V0.0.2 | W = random.rand() | 0.5577 | 0.7404 |  |

Passive aggressive\_t2

using constant to change loss function L(). We can achieve better results through testing the constant value. I find out that the constant = 0.1 for this test data is the best result for now when the initial weight is 0. But when it comes to random weight for the same constant, results are so bad.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Initial weight | Accurate while training | Accurate using training weight | Constant value |
| V0.0.1 | W = 0 | 0.6298 | 0.7596 | 0.02 |
| V0.0.1 | W = 0 | 0.6298 | 0.6442 | 0.5 |
| V0.0.1 | W = 0 | 0.6154 | 0.7740 | 0.1 |
| V0.0.1 | W = 0 | 0.5913 | 0.6683 | 0.2 |
| V0.0.1 | W = 0 | 0.6298 | 0.6442 | 0.4 |
| V0.0.1 | W = 0 | 0.6298 | 0.6442 | 100 |
| V0.0.1 | W = 0 | 0.5913 | 0.6971 | 0.3 |
| V0.0.1 | W= random.rand() | 0.5721 | 0.5529 | 0.1 |
| V0.0.1 | W= random.rand() | 0.5673 | 0.6058 | 0.1 |

Passive aggressive\_t3

Change constant value can cause different results, and sometimes the accurate results are the same, but the updated weights they have are slightly different. When find a good constant value for the initial weight = 0, but when it comes to random weights, results are really bad.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| V0.0.1 | W = 0 | 0.5913 | 0.6923 | 1 |
| V0.0.1 | W = 0 | 0.6298 | 0.6442 | 10-100 although the accurate are the same, but the weights of each constant task are slightly differently. |
| V0.0.1 | W = 0 | 0.6058 | 0.6923 | 0.5 |
| V0.0.1 | W = 0 | 0.6058 | 0.6637 | 0.1 |
| V0.0.1 | W = 0 | 0.6346 | 0.7452 | 0.001 |
| V0.0.1 | W= random.rand() | 0.4519 | 0.4086 | 0.001 |

OGD\_t1

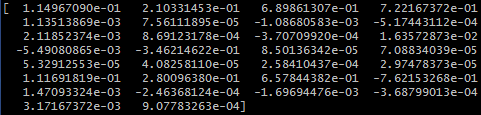
|  |  |  |  |
| --- | --- | --- | --- |
| Version | Initial weight | Accurate while training | Accurate using training weight |
| v.0.0.1 | W = 0 | 0.6202 | 0.6683 |
|  |  |  |  |

Breast-cancer-wisconsin data.

<https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data>

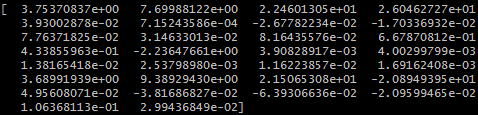
perceptron:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Initial weight | Accurate while training | Accurate using training weight |
| V0.1.0 | W = 0 | 0.6573 | 0.8647 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |



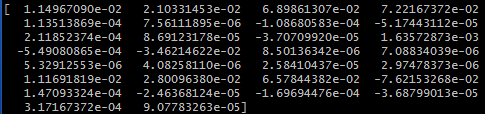
Passiveaggressive\_t1

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Initial weight | Accurate while training | Accurate using training weight |
| v.0.1.0 | W = 0 | 0.6573 | 0.7047 |
|  |  |  |  |



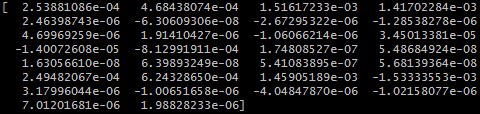
Passiveaggressive\_t2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Initial weight | Accurate while training | Accurate using training weight | Constant value |
| V0.1.0 | W = 0 | 0.6573 | 0.8647 | 0.1 |
|  |  |  |  |  |



Passiveaggressive\_t3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Initial weight | Accurate while training | Accurate using training weight | Constant value |
| V0.1.0 | W = 0 | 0.6239 | 0.8805 | 0.001 |
|  |  |  |  |  |



OGD\_t1

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Initial weight | Accurate while training | Accurate using training weight |
| V0.0.1 | W = 0 | 0.7399 | 0.8629 |
|  |  |  |  |

