Yahoo SAMOA Lab3 --Test Performance of parallel Naive Bayes Classifier

Li Huang

Computer Engineering and Computer Science

J.B. Speed School of Engineering

University of Louisville

l0huan08@gmail.com

2014.5.6

Abstract:

A parallel Naive Bayes Classifier(NB-6) was implemented under Yahoo SAMOA platform. This lab test the performance, including speed and correct rate, of this algorithm under different cluster setting: (1)single machine (2) two machine-cluster (3) three machine-cluster. The source data are from Kdd99[1] and Movie Review[2]. Vertical Hoeffding Decision Tree and a non-parallel version of Naive Bayes(NB-5) classifier are also tested and compared.

[1] Kdd99 data: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[2] Movie Review data: https://www.cs.cornell.edu/people/pabo/movie-review-data/

Contents

1.Purpose:

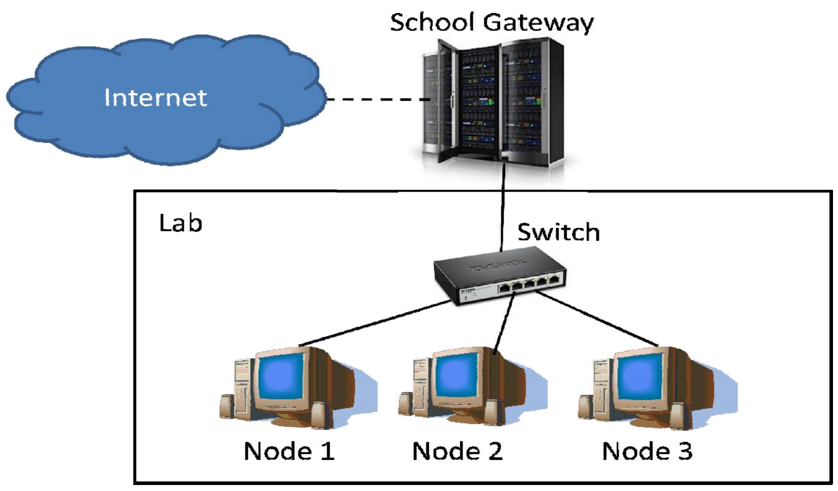
To learn how to implement user-defined data mining algorithm under SAMOA platform, I tried to implement a Naive Bayes Classifier. The previous experiment (Lab 2), I have implemented a non-parallel version called NB-5. In this experiment, I am trying to implement a parallel Naive Bayes algorithm (NB-6).

To test the characteristic and possibility of SAMOA, I test this algorithm on different cluster settings: Single-machine, Two-machine cluster, Three-machine cluster, to see if it is possible to get better performance when the cluster becomes larger(with more computing nodes).

I also compared this algorithm with VHDT(Vertical Hoeffding Decision Tree) which has been integrated into SAMOA, as well as the non-parallel version of my Naive Bayes classifier(NB-5).

2.Test environment:

The test is running on a cluster composed by three computers. They are connected by a switch. The hardware and software configuration are shown below:



Hardware	Software
node1: Pentium4 1.8Ghz, 576 MB RAM	Ubuntu Linux Desktop ver12.04 32-bit
node2: Pentium4 1.8Ghz, 768 MB RAM	Yahoo S4 ver0.6
node3: Pentium4 2.35Ghz, 495 MB RAM	Yahoo SAMOA ver0.0.1

3.Test settings

3 different test data are applied to the implemented Naive Bayes algorithm, the performances(speed and correct rate) are collected under 4 different cluster configuration.

3.1 Algorithm

3.1.1 Introduction of Naive Bayes Classifier

Please see the report of Lab-2[3].

3.1.2 Algorithms used in test

Three algorithms are applied and compared in this test.

(1) Non-Parallel Naive Bayes Classifier (NB-5)

This algorithm is a non-parallel Naive Bayes classifier. It is designed to handle real values and large integer attributes in input data. NB-5 is copy and modified from the source code of Naive Bayes Classifier in MOA(Massive Online Analysis)[4]. The detail of this algorithm please see my Lab-2 report.

Advantages:

Simple; Able to process both numeric attributes and nominal attributes.

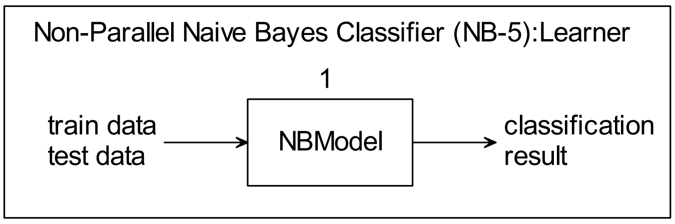
Disadvantages:

Not designed for parallel computing, so it cannot scale up its speed with more nodes.

The streaming processing topology (in SAMOA) of NB-5 is shown below:

[3] Samoa Lab-2 report: https://drive.google.com/file/d/0B0k3wDoweGSZcEVteTVNV3hlTlk/edit?usp=sharing

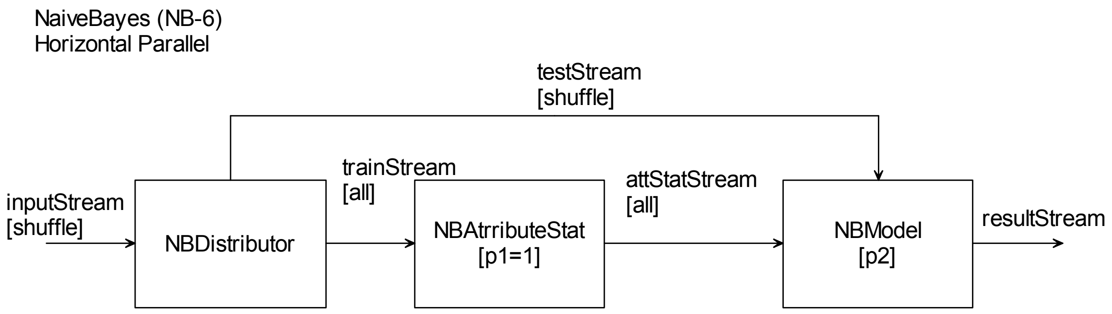
[4] MOA: http://moa.cms.waikato.ac.nz/



In the training phase, the NBModel processor read the training data and update the model. In the test phase, NBModel read the test data and predict their class labels and send the result to the result stream.

(2) Parallel Naive Bayes Classifier (NB-6)

This algorithm is a horizontal parallel classifier. It has multiple NBModel processors to classify the test data. Different NBModel could distribute in different nodes, so they can work parallely. The topology is shown below:



How NB-6 works

Training phase: The training data is sent from inputStream to NBDistributor. Then NBDistributor selects the training data and send them in chunk to the NBAttributeStat. NBAttributeStat stores the statistic model of Naive Bayes classifier, so it updates the model when receiving training data. After received each chunk of data, NBAttributeStat sends the updated Bayisian statistic model to all the NBModel processors, and each NBModel use this new model to substitute it old internal statistic model.

Test phase: The test data is sent from inputStream to NBDistributor. Then NBDistributor selects the test data and send them in chunk to a random NBModel. After received a chunk of test instance, NBModel classify these instances by its internal statistic model, and send the result to the resultStream.

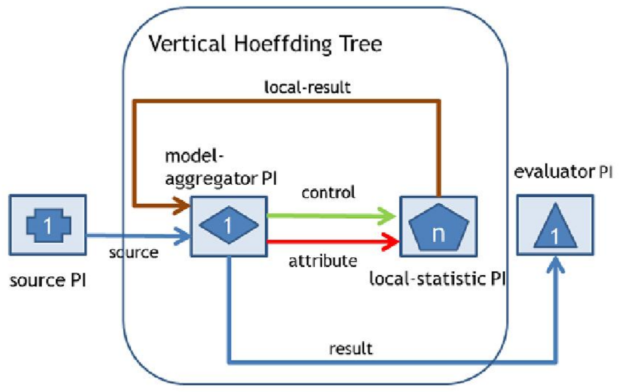
(3) Vertical Hoeffding Decision Tree (VHT)

This algorithm is the default classifier which has been implemented by SAMOA. It is a parallel algorithm that its speed can scale-up when the number of computing nodes increases.

VHDT is tested as a baseline to compare with my own Naive Bayes classifier. This algorithm is a parallel algorithm that would speed up with more computing nodes.

The relevant document could be access at
https://github.com/yahoo/samoa/wiki/Vertical%20Hoeffding%20Tree%20Classifier

The topology diagram is shown below:



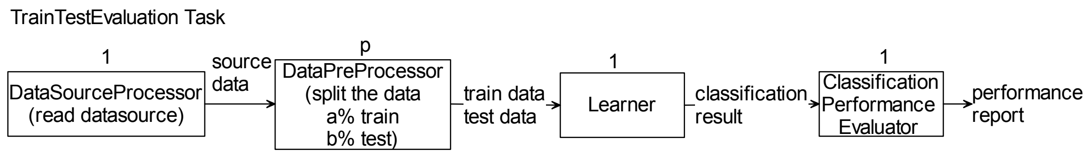
(4) TrainTestEvaluation Task

All the algorithms mentioned above were integrated in one of my custom tasks, "TrainTestEvaluation Task". This task is a framework of classify online data with user-defined classifiers.

I developed *TrainTestEvaluation* because the default task of SAMOA, *PrequentialEvaluation*, could not divide the input data into training part and test part and it considers each instance as both training data and testing data.

This task reads online data, then select part (a%) of the instance to the learner(classifier) for training, and the other part (b%) of data for testing; Finally the classification results are output to a performance evaluator, and the classification performance (such as correct rate, F-measure, etc…) are output.

The topology of TrainTestEvaluation is show below:


TrainTestEvaluation Task
1
DataSourceProcessor (read datasource)
source data
p
DataPreProcessor (split the data a% train b% test)
train data test data
1
Learner
classification result
1
Classification Performance Evaluator
performance report

In conclusion, the algorithms selected to compare are:

(1) Non-parallel Naive Bayes (NB-5)
(2) Parallel Niave Bayes (NB-6)
(3) Vertical Hoeffding Decision Tree (VHT)

3.2 Data

I used 3 different datasets to test the algorithms: MovieReview, NSL Kdd, Kdd 99. The detail introduction of the data could be found in the **Appendix**.

The three datasets are transformed into "ARFF"[5] file format to let SAMOA be able to read. They are:

data	**file**	**total instance number**	**attribute number**	**class number**	**note**
Movie Review data	m10000.arff	10,000	1172	2	
NSL Kdd99 data	n.arff	125,973	42	2 (unbalance 86% anomaly, 14% normal)	NB-4 cannot process because it contains real value attribute.
Kdd99 data	k.arff	4,898,431	42	23 (unbalance, 87% smurf, 12% neptune, 0.48% normal)	NB-4 and VHT cannot process because it contains error values and real value attribute.
In the test , 20% data are used for training, 80% data are used for testing.					

All the data could be downloaded from my Google drive:

[5] http://www.cs.waikato.ac.nz/ml/weka/arff.html

https://drive.google.com/folderview?id=0B0k3wDoweGSZUVlvTmIzSXNQUjQ&usp=sharing

3.3 Cluster modes

I test the different algorithms with different data, and I also need to test them in different cluster settings to see the parallel-computing scalability of the algorithms.

(1) Local mode (Local)

Only run task on node 3, in local mode without S4.

(2) S4 mode, single node (S4-1)

Run task on only node 3, with S4 platform.

(3) S4 mode, two nodes (S4-2)

Run task on cluster with 2 computing nodes (node 3 and node 2), with S4 platform.

(4) S4 mode, three nodes (S4-3)

Run task on cluster with 3 computing nodes (node 3, node2 and node 1), with S4 platform.

In conclusion, the test configurations can format as the combination table below:

Data: **movieReview, NSLKDD, KDD99**				
Algorithm/Cluster	**Local mode(single node)**	**S4, 1 node**	**S4, 2 node**	**S4, 3 node**
NB-5				
NB-6				
VHT				

4. Test Result

The test result are collected from: (1)result files (2) log(debug) information on the screen. Please see the middle test result in Appendix. Here I only show the final rearranged result.

Measurements:

I measured 2 index for each test configuration [dataset + algorithm + cluster mode]:

(1) **Speed** (seconds/X instances):

How many time the algorithm spends on train and testing every X data. X's are different for different dataset. (MovieReview,X=1000; NslKdd,X=10,000; Kdd99,X=100,000)

(2) **Correct rate**.

The percentage of test data that correctly classified (output class=target class).

4.1 Final result

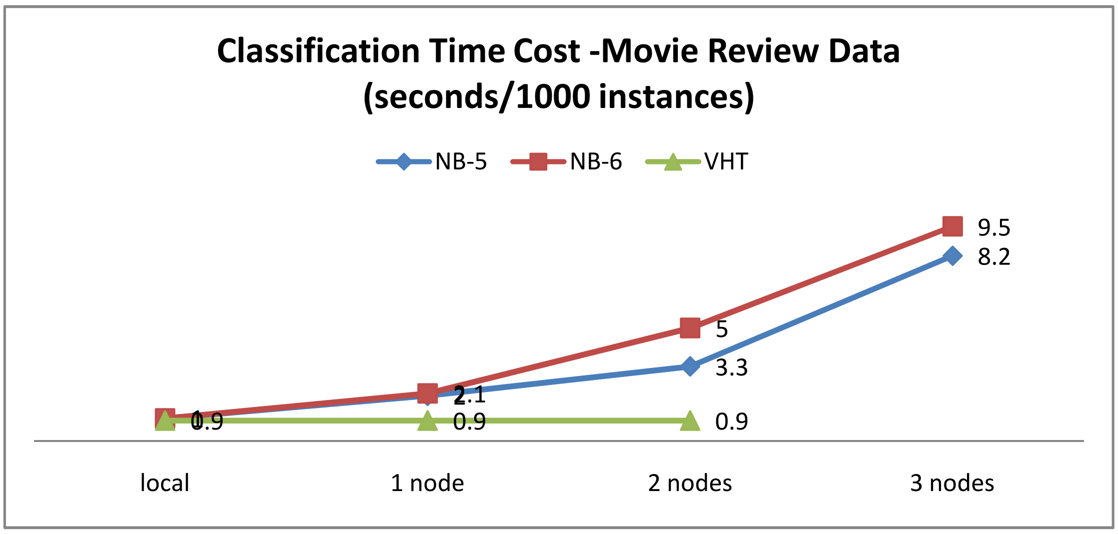
The table below shows the average speed of different algorithms running on different cluster modes and different datasets. Some cells in the table are blank, because the algorithm failed to run in the situation corresponding to this cell (See Appendix).

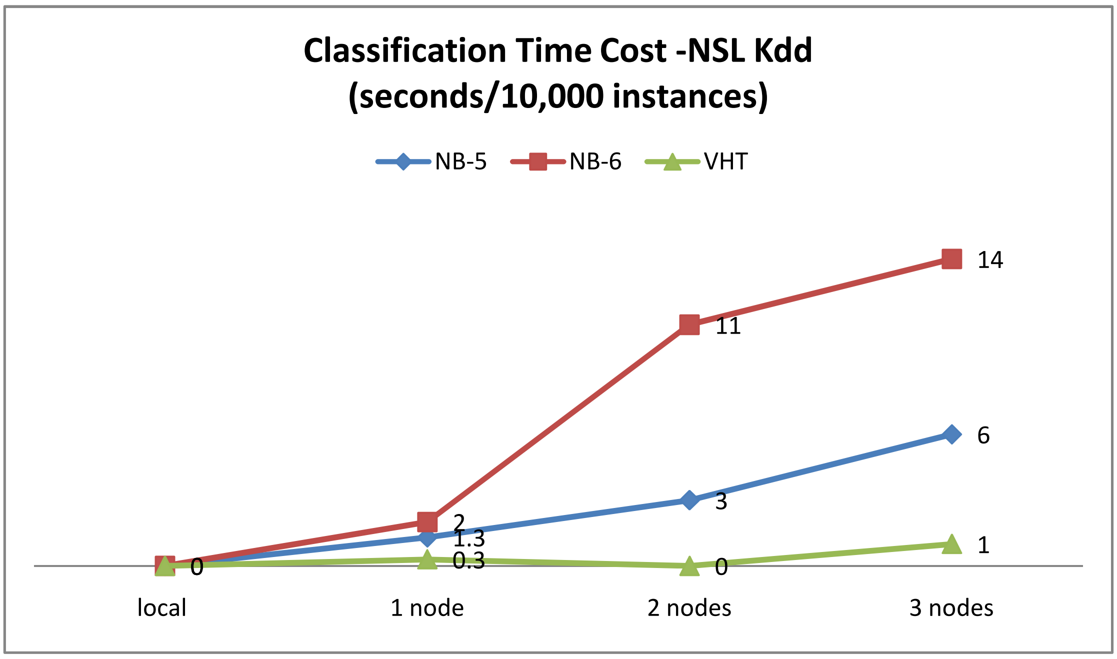
Average Speed (seconds / x instances)
Cells with "X" symbol means errors occurred in this test.

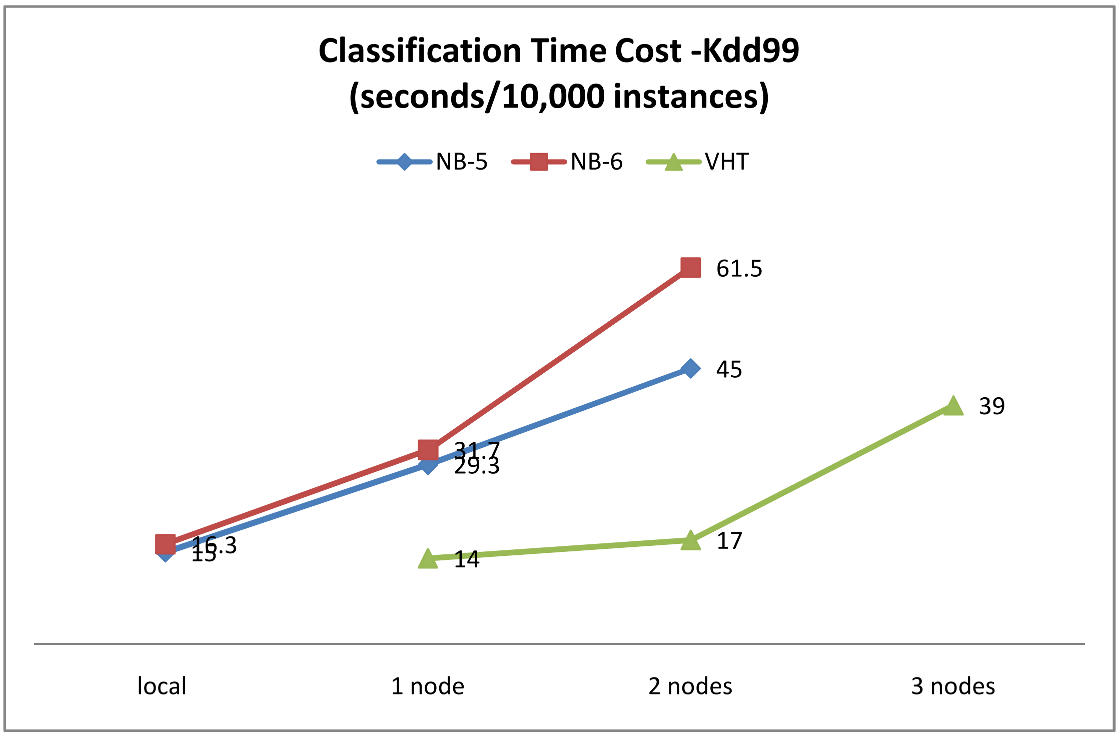
Algorithm/Dataset	**MovieReveiw** (seconds/1000 instances)				**NslKdd** (seconds/10,000 instances)				**KDD99** (seconds/100,000 instances)			
Cluster mode	local	1 node	2 nodes	3 nodes	local	1 node	2 nodes	3 nodes	local	1 node	2 nodes	3 nodes
NB-5	1	2	3.3	8.2	<1	1.3	3	6	15	29.3	45	X
NB-6	1	2.1	5	9.5	<1	2	11	14	16.3	31.7	61.5	X
VHT	0.9	0.9	0.9	X	0	0.3	<1	1	X	14 X	17 X	39 X

We can draw the "speed graphs" from the tables above:

The vertical axis shows the processing speed (Classification time cost of every unit frequency of data).

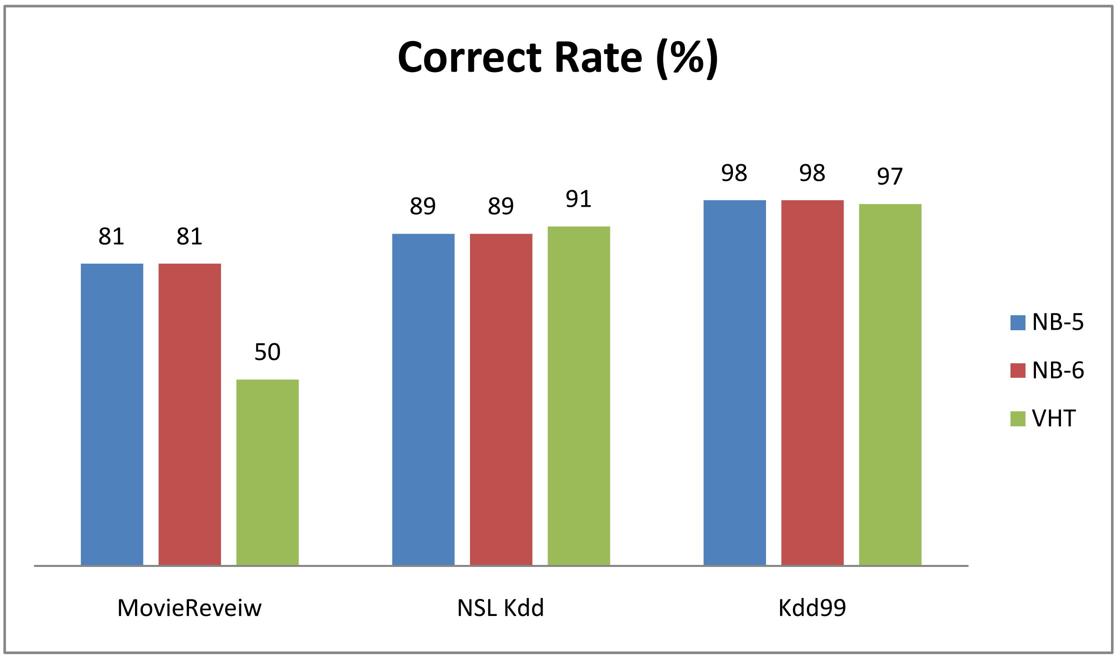

Classification Time Cost -Movie Review Data (seconds/1000 instances)
NB-5
NB-6
VHT
9.5
8.2
5
3.3
2
2.1
1
0.9
0.9
0.9
local
1 node
2 nodes
3 nodes


Classification Time Cost -NSL Kdd (seconds/10,000 instances)
NB-5
NB-6
VHT
14
11
6
3
2
1.3
0.3
0
0
1
local
1 node
2 nodes
3 nodes



The correct rate of different algorithms are basically stable among different cluster mode, so we can draw the correct rates table of different algorithms with different datasets.

Average Correct Rate (%)			
Algorithm/Dataset	**MovieReveiw**	**NSL Kdd**	**Kdd99**
NB-5	81	89	98
NB-6	81	89	98
VHT	50	91	97



5 Conclusion

(1) Comparing the speed of the algorithms:

Briefly, VHT > NB-5 > NB-6. (> means faster). For all the algorithms, the speed become slower with more nodes.

(2) Comparing the correct rate:

NB-5 is as same as NB-6, and VHT has similar correct rate except it has very low correct rate for MovieReview dataset which has a great number of attributes.

(3) Comparing compatibility of the algorithms:

NB-5 and NB-6 can correctly process all the datasets, except they failed to process Kdd99 dataset in 3-nodes mode. On the other side, VHT always throw errors for Kdd99 dataset in every mode.

In summary, in this lab, the processing speed did not get faster with more nodes, no matter using my algorithm or SAMOA's default algorithm. The reason might be:

a) My network communication is not fast, so dispatching the messages through the network and coordinating the nodes takes more time than classification computing.
b) The algorithm is simple and very fast. Multi-nodes mode need exchange information between nodes, and it takes more time than do all the computing in single node.

The SAMOA does not show its scalability as it promiss. However SAMOA is still in version 0.0.1 and the developers are updating the code frequently, SAMOA should have more improvement space in the future.

Appendix

A1. Data

(1) Movie Review data

Movie Review Data are collections of movie-review documents labeled with respect to their overall sentiment polarity (positive or negative) and sentences labeled with respect to their subjectivity status (subjective or objective) or polarity. Basically, a movie review dataset contains many instances, and each instance is a movie review and a class label, "good movie" or "bad movie", of this review.

In Lab 2, I only use the preprocessed data from "polarity dataset 2.0", which contains 1000 positive and 1000 negative processed reviews. The preprocessed data is created by my classmate, Wen Long Sun. Moreover, I shuffled the order of instances that the "pos" and "neg" instances occur alternately. This data contains 2000 instances; each instance has about 1000 attributes and 1 class label, like below:

Movie Review (m10000.arff)

Attribute	&	*	-	1	10	2	…	acc	adam	…	unit	visit	@@class@@
line 1	1	0	0	0	1	0	…	0	0	…	1	1	pos
line 2	0	0	1	0	0	0	…	1	0	…	0	1	neg
line 3	0	1	0	0	0	1	…	0	0	…	0	0	pos
…	…	…											

Each instance stands for a review document, and each attribute is the occurrence of a word in this document. For example, if the word "adam" occurs in this document, then the value of attribute "adam" should set to 1, otherwise set to 0. The last attribute "@@class@@" shows the sentiment polarity of each review document.

In lab 2, this original dataset is called m.arff.

In Lab 3, for achieving enough training instances, I repeat this original dataset 5 times, makeit has 10,000 instances, in which 8,000 are repeated instances. This new dataset called m10000.arff.

Size:

1000(instances) x 1172(attributes)

1000 instances are “pos”, and 1000 are “neg”.

(2) Kdd99 data

Kdd99 data can be found at https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Kdd99 data is created to build a predictive model (i.e. a classifier) capable of distinguishing between ``bad'' connections, called intrusions or attacks, and ``good'' normal connections.

The data contains about 5 Million instances, each with 41 attributes and 1 class label. Each instance is a connection record, including features such as “length (number of seconds) of the connection”, “type of the protocol”, “normal or error status of the connection”, etc. The class label is “normal” connection or 22 types of attacks, such as “buffer_overflow”, “guess_passwd”, etc.

The 22 types of attacks fall into four main categories:

- DOS: denial-of-service, e.g. syn flood;
- R2L: unauthorized access from a remote machine, e.g. guessing password;
- U2R: unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow'' attacks;
- probing: surveillance and other probing, e.g., port scanning.

The data is like:

Kdd99 (k.arff)									
Attribute	duration	protocol_type	service	flag	src_bytes	dst_bytes	…	dst_host_srv_rerror_rate	class
line 1	0	tcp	http	SF	215	45076	…	0	normal
line 2	0	tcp	http	SF	162	4528		0	normal
line 3	0	tcp	http	SF	236	1228		0	smurf

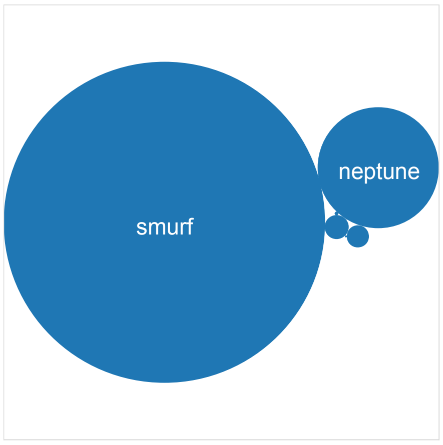
line 4	0	tcp	http	SF	233	2032		0	neptune
…	…	…							…

It's **highly unbalanced** data that most of the instances are labeled "smurf".

Size:

4,898,431(instances) x 42(attributes)

class distribution: Total 23 classes. 87% are "smurf"(DOS attack), 12% are "neptune"(DOS attack), 0.48 % are "normal", as shown below:

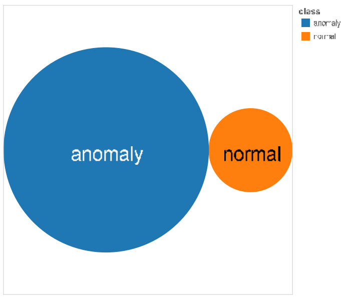


(3) NSL Kdd99 data

NSL Kdd99 data could be found at http://nsl.cs.unb.ca/NSL-KDD/ . It is an improvement of KDD99 that it is more efficient for training classifiers. It remove redundant records; it transforms class labels into only two types: "normal" and "anomaly"; it select the records to make it more balance.

Size:

125,973(instances) x 42(attributes)



class distribution: 86% anomaly,   14% normal

A2. Middle test result

Test Data: m.arff (Movie Review)

measurements:

(1) classification(test) speed: x seconds/1000 instances
(2) classification correct rate: y%

	Local	S4-1 node	S4-2 nodes	S4-3 nodes
NB-5	x=1,1,1,1,1,1,1, y=82,81,81,81,81,81, 81	x=2,2,2,2,2,2,2 y=82,81,81,81,81,81, 81	x=4,4,3,3,3,3,3,3 y=81,82,81,81,81,82, 81	x=7,7,7,7,13 y=81,81,81,81, 81
NB-6	x=1,1,1,1,1,1,1 y=82,81,81,81,81,81, 81	x=2,2,2,2,3,2,2 y=81,81,81,81,81,81, 81	x=5,5,5 y=80,81,82	x=8,11 y=80,80
VHT	x=0,1,1,1,1,1,1 y=48,50,49,50,49,50, 49	x=0,1,1,1,1,1,1 y=49,50,49,50,49,50, 49	x=0,0,2,1,1,1,1 y=51,51,50,50,49,50	x= y= out of memory

Test Data: n.arff (NSL Kdd)

measurements:

(3) classification(test) speed: x seconds/10,000 instances
(4) classification correct rate: y%

	Local	S4-1 node	S4-2 nodes	S4-3 nodes

NB-5	x=0,0,0
	y=89,89,89
		x=1,1,2,1
		y=89,89,89
			x=3,3,3
			y=90,89,89
				x=5,6,7
				y=90,89,89

NB-6	x=0,0,0
	y=89,89,89
		x=2,2,2
		y=89,89,89
			x=4,18
			y=89,89
				x=14
				y=90

VHT	x=0,0,0
	y=91,90,91
		x=0,1,0
		y=92,92,92
			x=0,0,0
			y=90,90,90
				x=0,1,2
				y=90,90,90

Test Data: k.arff (Kdd99)

measurements:

(5) classification(test) speed: x seconds/100,000 instances
(6) classification correct rate: y%

	Local	S4-1 node	S4-2 nodes	S4-3 nodes

NB-5	x=15,15,15
	y=99,98,98
		x=30,29,29
		y=99,98,98
			x=57,49,29
			y=99,98,99
				x=
				y= out of memory

NB-6	x=16,16,17
	y=99,98,98
		x=50,29,16
		y=99,98,98
			x=83,40
			y=98,96
				x=
				y=out of memory

VHT	x= error
	y=
		x=12,19,11
		y=76,69,66
		error
			x=18,16
			y=97,97
			error
				x=39
				y=97
				error