**Related work**

\*Top ing. + J48

^How to choose top num.?

^Best: Top 200, S=0.57; Top 1703, S=?

**New methods**

\*KNN

^How to choose K?

^d(id1,id2) = #{ing(id1) \neq ing(id2)}

^Best: K=21, S=0.67659

\*PCA + SMO

^PCA: linear unsupervised reduction

^How to choose #(PCs)? (Score.pdf)

^Best: PCs 1000, S=0.66020; PCs 1703, S=?

**Discussion and Conclusion**

\*Text mining

\*Compressed sensing.

\*Factorization Machines (2010).

\*Latent Dirichlet Allocation.

----------------------------------------------------------------------------------------------------------------

＊My first try by finding the top 5 ing. => 0.21360 ….

＊KNN with the metric defined by d(id1,id2) = #{ing(id1) \neq ing(id2)} with Condensed Nearest Neighbor or KD Tree.

Step.1 Read data and choose CNN

#(class) = 20

#(train) = 39xxx

#(test) = 9944

#(longest ing.) = 65

#(shortest ing.) = 1

Step.2 Compute CNN

#(same ing.) more => d(id1,di2) shorter

#(CNN) = 19xxx

Step.3 Implement KNN

Data = CNN.json

K=1 => 0.53691

K=3 => 0.55471

K=5 => 0.61394

K=7 => 0.63284

K=9 => 0.63988

K=11 => 0.64853

K=13 => 0.65296

K=15 => 0.65718

K=17 => 0.65426

K=19 => 0.65698 Not good enough…

K=21

K=23

K=25

K=1501

Use parallel computations?

How to choose K?

Data = train.json

K = 21 => 0.67659

K = 23

＊Train directly.

read\_data.py create ingredients.csv

create\_mtx.py create train\_mtx.csv (510M)

create\_weka.py create train\_weka.csv

./weka-csv-arff.sh ./train\_weka.csv > ./train\_weka.arff create train\_weka.arff 🡪 Failed to make arff!!

＊Find top 10 ing. (total 63) to reduce the features (dimension reduction).

create\_top\_ing.py create ing\_top10.csv

create\_mtx.py create train\_weka\_top10.csv (5.3M)

./weka-csv-arff.sh ./train\_weka\_top10.csv > ./train\_weka\_top10.arff Only convert 369 instances 🡪 Failed to make arff!!

weka start to train 🡪 Just a try!!

AutoWeka: NaiveBayes "-K"(37.3984 % = =a)

J48 (59.6206 %)

SimpleLogistic (68.2927 %)

MultilayerPerceptron (84.2818 %)

VotedPerceptron (87.8049 %)

IBk (93.2249 %)

RandomForest (93.2249 %)

RandomTree (93.2249 %)

create\_mtx.py create test\_weka\_top10.csv (1.3M)

./weka-csv-arff.sh ./test\_weka\_top10.csv > ./test\_weka\_top10.arff

change 1001th attribute

weka start to test

create train\_weka\_top10\_IBk.model

create test\_weka\_top10\_IBk.txt

create test\_weka\_top10\_IBk.csv

create test\_weka\_top10\_IBk\_sub.csv

🡪 Only 0.27836 |||

./weka-csv-arff.pl < ./train\_weka\_top10.csv > ./train\_weka\_top10.arff

create train\_weka\_top10\_cnn.arff 🡪 Success!!

weka Classifier (Correct/C-V/Percen66%)

AutoWeka{5hr} (//47.697 %)

🡪 Too low!!

＊Find top 10 ing. (total 63) + CNN to reduce the features (dimension reduction).

create\_top\_ing.py create ing\_top10.csv

create\_cnn.py create train\_cnn.csv 🡪 #(train\_cnn) = 19800

create\_mtx.py create train\_weka\_top10\_cnn.csv (2.7M)

./weka-csv-arff.sh ./train\_weka\_top10\_cnn.csv > ./train\_weka\_top10\_cnn.arff Only convert 271 instances 🡪 Failed to make arff!!

divide\_data.py divide train data into two files

train\_weka\_top10\_cnn\_div1.csv (1.3M)

train\_weka\_top10\_cnn\_div2.csv (1.3M)

It’s very weird.

test\_weka\_top10.csv 🡪 1.3M

train\_weka\_top10\_cnn\_div1.csv 🡪 1.3M

Why can I convert 1st instead of 2nd?

“java.io.IOException: Read uknown nomial value russian…”

Converting csv to arff is sooooooooo difficult!!!

./weka-csv-arff.pl < ./train\_weka\_top10\_cnn.csv > ./train\_weka\_top10\_cnn.arff

create train\_weka\_top10\_cnn.arff 🡪 Success!!

weka Classifier (Correct/C-V/Percen66%)

Auto-weka{2hr} (/34.2828 %/)

NaiveBayes (36.6818 %//)

Logistic (//) ??

MultilayerPerceptron (41.1667 %//)

SMO (//)

VotedPerceptron (//)

IBk{k=} (//)

J48(50.0808 %//)

RandomForest (73.0253 %/Can’t run[P2]/31.8925 %)

RandomTree (//)

🡪 Too low!!

＊Find top 35 ing. (take total 200) to reduce the features (dimension reduction).

create\_top\_ing.py create ing\_top35.csv

create\_mtx.py create train\_weka\_top35.csv (16M)

./weka-csv-arff.pl < ./train\_weka\_top35.csv > ./train\_weka\_top35.arff

create train\_weka\_top35.arff

weka AutoWeka{6hr,2fold} (/61.0172 %/)

Naïve Bayes{4fold} (/57.6683 %/)

SMO{4fold} (/63.9689 %/)

IBk{k=2001,4fold} (/21.8208 %/)

J48{4fold} (/58.8223 %/)

RandomTree{4fold} (/46.6134 % /)

MultiClassClassifier{1-against-all,4fold} (/64.3234 %/)

MultiClassClassifier{1-against-1,4fold} (/Can’t run/)

🡪 Not good enough.

＊Find top 35 ing. (take total 200) + ing\_len (normalized) to reduce the features (dimension reduction).

create\_top\_ing.py create ing\_top35.csv

create\_mtx.py create train\_weka\_top35\_len.csv (16.6M)

./weka-csv-arff.pl < ./train\_weka\_top35\_len.csv > ./train\_weka\_top35\_len.arff

create train\_weka\_top35\_len.arff (16.6M)

weka AutoWeka{2.5hr,2fold} (/51.xxxx %/)

MultiClassClassifier{1-against-all,4fold} (/64.7332 %/)

🡪 Not good enough.

create\_mtx.py create test\_weka\_top35\_len.csv (16.6M)

./weka-csv-arff.sh ./test\_weka\_top35\_len.csv > ./test\_weka\_top35\_len.arff

change 1001th attribute

weka start to test

create train\_weka\_top35\_len\_RandomForest.model

create test\_weka\_top35\_len\_RandomForest.txt

create test\_weka\_top35\_len\_ RandomForest.csv

create test\_weka\_top35\_len\_ RandomForest \_sub.csv

🡪 0.57552

＊ Find top 200 ing. (total 1055) + ing\_len (normalized).

prefixFilter create train.json 🡪 delete special characters

create\_top\_ing.py create ing\_top200.csv

create\_mtx.py create train\_weka\_top200\_len.csv (81M)

./weka-csv-arff.pl < ./train\_weka\_top200\_len.csv > ./train\_weka\_top200\_len.arff

weka MultiClassClassifier{1-against-all} (//out of mem)

MultiClassClassifier{1-against-1} (//)

MultilayerPerceptron (//too long)

IBk{k=1501} (//31.7533 %)

NaïveBayes (//63.3365 %)

J48 (//64.2387 %)

SMO (//)

RandomForest (//out of mem)

RandomTree (//45.7739 %)

Logistic (//)

VotedPerceptron (//)

🡪 Not good..

＊Use all data and PCA to reduce the features (dimension reduction).

do\_pca.cpp Have checked that eigenvectors are o.n.

divide\_into\_vec\_val.pl

create\_pca\_mtx.m create train\_pca\_mtx\_K1000.csv

create\_weka.py create train\_weka\_tol1000\_pca.csv (203M by round)

./weka-csv-arff.pl < ./train\_weka\_tol1000\_pca.csv > ./train\_weka\_tol1000\_pca.arff

weka NaiveBayes (//36.8927 %)

MultilayerPerceptron (//too long)

J48 (//40.176 %)

RandomTree (//24.2624 %)

🡪 Too low..

create\_pca\_mtx.m create train\_pca\_mtx\_K1703.csv (S>=0.95)

create\_weka.py create train\_weka\_tol1703\_pca.csv (339M by round)

./weka-csv-arff.pl < ./train\_weka\_tol1703\_pca.csv > ./train\_weka\_tol1703\_pca.arff

weka NaiveBayes (//36.2198 %)

J48 (//out of mem)

🡪 Too low..

create\_pca\_mtx.m create train\_pca\_mtx\_K1703\_t.csv (S>=0.95)

create\_weka.py create train\_weka\_tol1703\_t\_pca.csv (337M by round)

./weka-csv-arff.pl < ./train\_weka\_tol1703\_t\_pca.csv

> ./train\_weka\_tol1703\_t\_pca.arff

weka NaiveBayes (//44.4035 %)

🡪 Too low..

create\_pca\_mtx.m create train\_pca\_mtx\_K1000\_n.csv

create\_weka.py create train\_weka\_tol1000\_n\_pca.csv (187M by round)

./weka-csv-arff.pl < ./train\_weka\_tol1000\_n\_pca.csv

> ./train\_weka\_tol1000\_n\_pca.arff

weka MultiClassClassifier{1-against-all} (//66.2797 %)

MultilayerPerceptron (//)

IBk{k=1501} (//30.8364 %)

NaïveBayes (//3x.xxx %)

J48 (//40.0281 %)

SMO (//73.2382 %)

SMO{RBF} (//)

RandomForest (//)

RandomTree (//23.8335 %)

Logistic (//)

VotedPerceptron (//)

create train\_weka\_tol1000\_n\_pca\_SMO.model

create\_mtx.py create test\_mtx.csv

create\_pca\_mtx.m create test\_pca\_mtx\_K1000\_n.csv

create\_weka.py create test\_weka\_tol1000\_n\_pca.csv (48.7M)

./weka-csv-arff.pl < ./test\_weka\_tol1000\_n\_pca.csv

> ./test\_weka\_tol1000\_n\_pca.arff

change 1001th attribute

weka create test\_weka\_tol1000\_n\_pca\_SMO.txt

🡪 arff to csv fail!!

./weka-to-kaggle.pl < ./ test\_weka\_tol1000\_n\_pca\_SMO.txt

> ./test\_weka\_tol1000\_n\_pca\_SMO\_sub.csv

🡪 0.66020

create\_pca\_mtx.m create train\_pca\_mtx\_K2000\_n.csv

create\_weka.py create train\_weka\_tol2000\_n\_pca.csv (390M by round)

./weka-csv-arff.pl < ./train\_weka\_tol2000\_n\_pca.csv

> ./train\_weka\_tol2000\_n\_pca.arff

weka MultiClassClassifier{1-against-all} (//)

NaiveBayes (//out of mem)

SMO (//can’t run)

Logistic (//can’t run)

Need xi-u?

Modify the parameters ref. Weka Bible, Lin’s PPT.

Preprocess data ref. Python website.

＊Kernels on Kaggle or others.

Winners’ codes.

Compressed sensing.

Factorization Machines (2010).

Latent Dirichlet Allocation.

Auto Scikit.

**一、主題、研究動機、主要處理問題**

\*Supervised problem

\*Multi-class classification

\* The order of id won’t affect the result.

**二、研究方法、過程、範例說明**

(above)

**三、討論與心得(彙整歸納對此論文的個人看法，包含對於研究題目的認識，研究方法的認知)**

Error rate (micro/macro)

Ein

k-fold Cross-validation

Paired t-test

Confusion matrix

ROC curve

AUC value

**附錄、Weka標準流程**

\*Sketch the steps.

Transform training data to a csv file.

Try several multi-class classifications and choose features (ex. #(ing.) for each cuisine).

Compute the accuracy and cross validation.

Choose a model (accuracy >= 80) to test. ex. J48, KNN, PLA, Bayes...

\*Formal steps.

Create a new csv data file (in a needed form).

Convert it to UTF8 encoding. (use instruction in vim [A]).

Convert it into train and test arff files (Hsin’s shell-script [A]).

Train train.arff by xxx on Weka, analyze the data (MAE, ROC…), and save xxx.model.

Test test.arff by the model, and save result\_xxx.txt.

Convert result\_xxx.txt to result\_xxx.csv.

\*Tried.

Test some of the Cooking data by Weka before the formal training.

Cross-validation.

Use Auto-Weka 2.5 🡪 For the newest Weka 3.8.0 and Auto-Weka 2.5, it needs to install Java and JDK and type the specific instruction to avoid java executable issue. [A]

**附錄、特殊指令**

\*Terminal 指令

sendto16.sh ./test.py Kaggle/test.py

Ctrl+A -> d 斷開遠端桌面

tmux ls 查詢跑code用的tmux序號

tmux a –t 0 連回該tmux畫面

getfrom16.sh Kaggle/submission.csv .

\*轉unicode指令

:set ff=unix

\*使用Auto-Weka指令

#!/bin/bash

java -jar /Applications/weka-3-8-0-oracle-jvm.app/Contents/Java/weka.jar &

\*將csv檔轉成train與test的arff檔指令

weka-csv-arff.sh cuisine2.csv | ./gogo.pl 0 轉為test.arff

weka-csv-arff.sh cuisine2.csv | ./gogo.pl 1 轉為train.arff

\*內文轉unicode

Transform 7xxx ing. into a new-data csv file. 🡪 Failed due to unicode issue..(Solved)

UnicodeEncodeError: 'ascii' codec can't encode character u'\xa0' in position 20: ordinal not in range(128)

a.encode(‘utf-8’)

\*去除特殊字元çèé®™

cd prefixFilter

make

./prefixFilter train.json create train.json.tmp

\*修剪資料指令(weka預測完要擷取結果資料, 網路上的arff to csv因檔案大不能用時)

sed -n 191927,201927p ./test\_weka\_tol1000\_n\_pca\_SMO.txt > ./data.csv

cat ./data.csv | cut -d , -f3|sed 's/.\*://g' > result.txt

**附錄、Multi-class classifiers**

Transform to binary: OvR(1-against-all), OvO(1-against-1)

Extend from binary: NN(MultilayerPerceptron), Extreme learning machines (ELM), kNN(IBk), Naive Bayes, Decision trees(J48), SVM(SMO)

Hierarchical classification

**附錄、Compressed sensing**

\*方法：

把data轉成(0,1)向量x (ex. 用字典)

前乘一個隨機矩陣A得到向量b

重構回x

\*條件：

A滿足restricted isometry property (RIP)

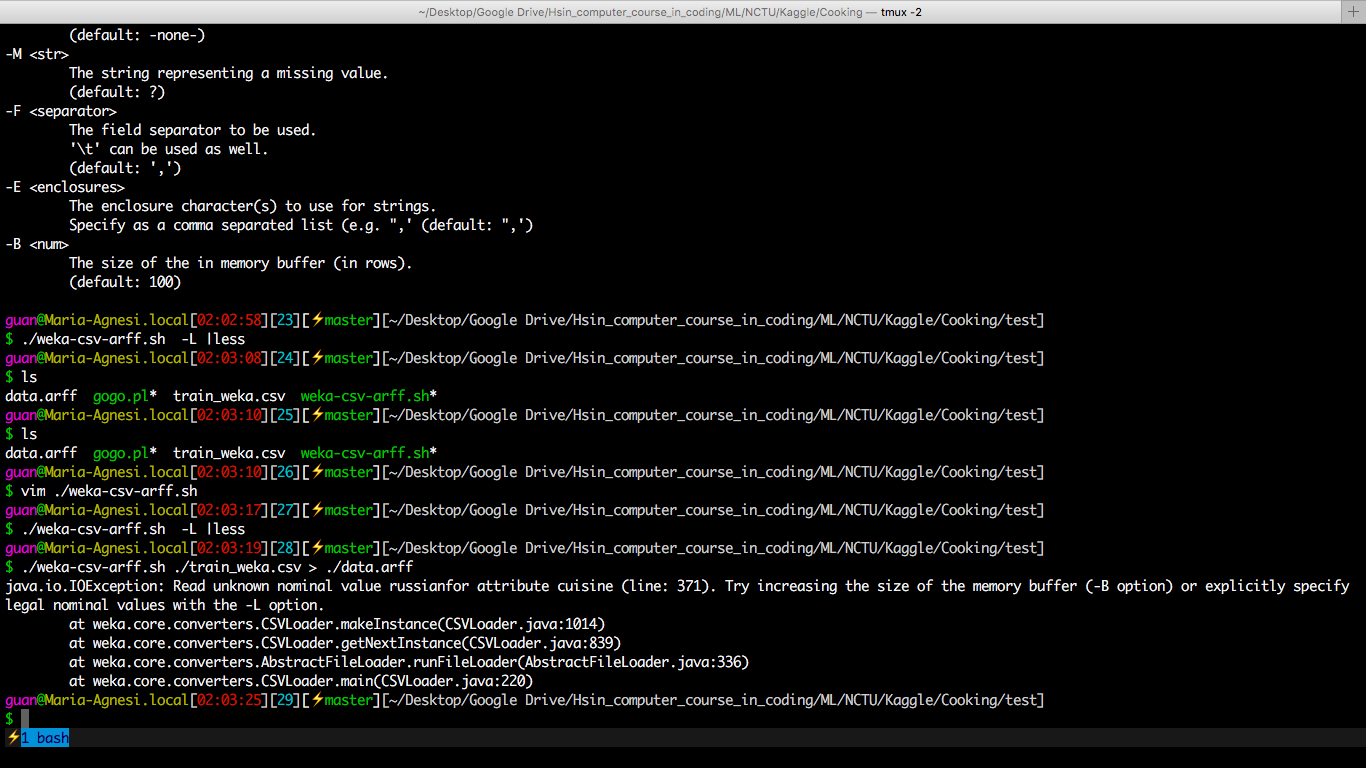
\*定理：

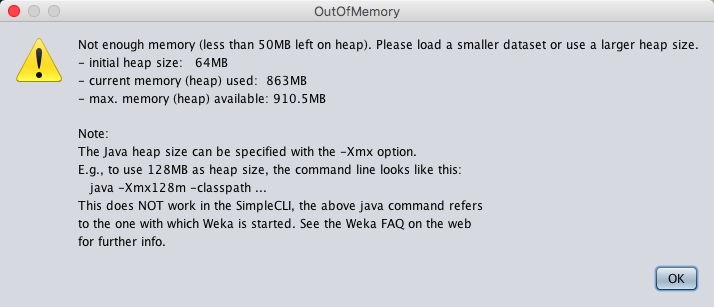
1. A是N(0,1)那種矩陣就高機率滿足RIP

2. A滿足RIP且x稀疏則可用b解回x

**附錄、HMM(隱馬可夫)**

**附錄、錯誤訊息**

****



**附錄、top\_ing總材料數**

減去’cuisine’

top35: 20x

top45: 280

top55: 346

top60: 377

top62: 386

top63: 392

top64: 398

top65: 402

top200: 1055