

## Comp 543 Assignment 5

Kai-Po Lin

### Task 1:

#### 【Result】

applicant: 604

and: 2

attack: 515

protein: 3681

car: 635

```
[15]: # and we'll create a RDD that has a bunch of (word, dictNum) pairs
# start by creating an RDD that has the number 0 thru 20000
# 20000 is the number of words that will be in our dictionary
twentyK = sc.parallelize(range(20000))

# now, we transform (0), (1), (2), ... to ("mostcommonword", 0) ("nextmostcommon", 1), ...
# the number will be the spot in the dictionary used to tell us where the word is located
# A bunch of (word, posInDictionary) pairs
dictionary = twentyK.map(lambda x: (topWords[x][0], x))

# Collect the Rdd to a Dict
localDict = dictionary.collectAsMap()
for inputWord in ["applicant", "and", "attack", "protein", "car"]:
    if inputWord in localDict:
        print(f'{inputWord}: {localDict[inputWord]}')
    else:
        print(f'{inputWord}: -1')
```

#### ► Spark Job Progress

applicant: 604  
and: 2  
attack: 515  
protein: 3681  
car: 635

### Task 2:

#### (a) Writing up your gradient update formula

#### 【Result】

Gradient Function : 
$$-\left(\sum_i y_i \cdot x_i - \frac{x_i \cdot e^{\sum_j x_{ij} \cdot w_j}}{1 + e^{\sum_j x_{ij} \cdot w_j}} - \lambda \times 2 \times W\right)$$
  
(W: weight, x: input data,  
y: label)

## (b) Giving us the fifty words with the largest regression coefficients

### 【Result】

#### Train on Large Dataset:

['five', 'post', 'seems', 'passing', 'experienced', 'copy', 'netherlands', 'rivers', 'demands', 'tons', 'holiday', 'traded', 'wishes', 'denver', 'hughes', 'airing', 'wise', 'religions', 'morocco', 'fiber', 'issuing', 'bells', 'incorporates', 'gaga', 'pornography', 'pulls', 'captures', 'handsome', 'succeeds', 'clarkson', 'dungeons', 'debbie', 'collier', 'pushes', 'angrily', 'kb', 'gravitational', 'introductory', 'confronting', 'embark', 'peugeot', 'trainee', 'hawthorn', 'championed', 'bromwich', 'manipur', 'orr', 'atl', 'thrilled', 'mcdowell']

```
[22]: idx = np.argmax(w, -50)[-50:]
      output = list()

      for key, value in localDict.items():
          if value in idx:
              output.append(key)
      print(output)

VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),...
```

['five', 'post', 'seems', 'passing', 'experienced', 'copy', 'netherlands', 'rivers', 'demands', 'tons', 'holiday', 'traded', 'wishes', 'denver', 'hughes', 'airing', 'wise', 'religions', 'morocco', 'fiber', 'issuing', 'bells', 'incorporates', 'gaga', 'pornography', 'pulls', 'captures', 'handsome', 'succeeds', 'clarkson', 'dungeons', 'debbie', 'collier', 'pushes', 'angrily', 'kb', 'gravitational', 'introductory', 'confronting', 'embark', 'peugeot', 'trainee', 'hawthorn', 'championed', 'bromwich', 'manipur', 'orr', 'atl', 'thrilled', 'mcdowell']

#### Train on Medium Dataset:

['that', 'not', 'any', 'court', 'act', 'mr', 'evidence', 'decision', 'whether', 'tribunal', 'application', 'applicant', 'claim', 'matter', 'reasons', 'appeal', 'appellant', 'orders', 'relevant', 'ltd', 'sought', 'notice', 'circumstances', 'relation', 'hearing', 'proceedings', 'respondent', 'consider', 'matters', 'regard', 'proceeding', 'respondents', 'pty', 'judgment', 'satisfied', 'submissions', 'affidavit', 'magistrate', 'pursuant', 'fca', 'clr', 'hca', 'amp', 'discretion', 'fcr', 'alr', 'jurisdictional', 'relevantly', 'fcafc', 'gummow']

```
[237]: idx = np.argmax(w, -50)[-50:]
      output = list()

      for key, value in localDict.items():
          if value in idx:
              output.append(key)
      print(output)

['that', 'not', 'any', 'court', 'act', 'mr', 'evidence', 'decision', 'whether', 'tribunal', 'application', 'applicant', 'claim', 'matter', 'reasons', 'appeal', 'appellant', 'orders', 'relevant', 'ltd', 'sought', 'notice', 'circumstances', 'relation', 'hearing', 'proceedings', 'respondent', 'consider', 'matters', 'regard', 'proceeding', 'respondents', 'pty', 'judgment', 'satisfied', 'submissions', 'affidavit', 'magistrate', 'pursuant', 'fca', 'clr', 'hca', 'amp', 'discretion', 'fcr', 'alr', 'jurisdictional', 'relevantly', 'fcafc', 'gummow']
```

### Task 3:

### 【Result】

#### (a.) Test on Medium Dataset

TP: 218

TN: 8711

FP: 9636

FN: 159

8929 out of 18724 correct.

Precision: 0.022122995737771465

Recall: 0.5782493368700266

F1 Score: 0.042615580099697

```
[32]: print(f'TP: {tp}')  
      print(f'TN: {tn}')  
      print(f'FP: {fp}')  
      print(f'FN: {fn}')
```

TP: 218  
TN: 8711  
FP: 9636  
FN: 159

```
[33]: precision = tp / (tp + fp)  
      recall = tp / (tp + fn)  
      f1score = 2 * precision * recall / (precision + recall)  
      print("%d out of %d correct." % (tp + tn, len(prediction)))  
      print(f"Precision: {precision}")  
      print(f"Recall: {recall}")  
      print(f"F1 Score: {f1score}\n")
```

8929 out of 18724 correct.  
Precision: 0.022122995737771465  
Recall: 0.5782493368700266  
F1 Score: 0.042615580099697

## (b.) Test on Small Dataset:

TP: 72

TN: 3360

FP: 8

FN: 2

3432 out of 3442 correct.

Precision: 0.9

Recall: 0.972972972972973

F1 Score: 0.935064935064935

```
[236]: print(f'TP: {tp}')  
      print(f'TN: {tn}')  
      print(f'FP: {fp}')  
      print(f'FN: {fn}')  
      print("%d out of %d correct." % (tp + tn, len(prediction)))  
      print(f"Precision: {precision}")  
      print(f"Recall: {recall}")  
      print(f"F1 Score: {f1score}\n")
```

TP: 72  
TN: 3360  
FP: 8  
FN: 2  
3432 out of 3442 correct.  
Precision: 0.9  
Recall: 0.972972972972973  
F1 Score: 0.935064935064935

## (c.) 3 examples of FP => Index 103, 341, and 549. Index 103 is an article talking about “Bankruptcy”,

index 341 is an article talking about “legal systems”, and index 549 is an article talking about “contract and law cases”. I consider that the words used in these articles will somewhat appear in the words used in Australian court cases, and that may be the reason why the model will predict it as positive.