

# CS486 - A3

March 14, 2020

## 1 Part A

### 1.1 (1)

```
[1]: import numpy as np
import copy
```

```
[2]: class Factor:
    def __init__(self, vars, probs):
        self.header = np.array(sorted(vars))
        if type(probs) is np.ndarray:
            self.table = probs
        else:
            order = np.argsort(vars)
            probs.sort(key=lambda x: tuple(x[i] for i in order))
            self.table = np.array([p[-1] for p in probs])
            self.table = self.table.reshape((2, ) * len(vars))

    def restrict(self, var, val):
        if var not in self.header:
            return

        index = 1 if val else 0
        axis = np.where(self.header == var)[0][0]
        self.header = np.delete(self.header, axis)
        self.table = np.take(self.table, index, axis=axis)

    def __mul__(self, other):
        new_header = np.array(sorted(np.union1d(self.header, other.header)))
        a = self.table.copy()
        b = other.table.copy()

        for index, var in enumerate(new_header):
            if var not in self.header:
                a = np.expand_dims(a, axis=index)
            if var not in other.header:
                b = np.expand_dims(b, axis=index)
```

```

        new_table = a * b
        return Factor(new_header, new_table)

    def sumout(self, var):
        if var not in self.header:
            return

        axis = np.where(self.header == var)[0][0]
        self.header = np.delete(self.header, axis)
        self.table = np.sum(self.table, axis=axis)

    def normalize(self):
        self.table /= np.sum(self.table)

    def possibility(self, vars):
        if len(vars) != len(self.header):
            raise Exception("Invalid query size.")

        index = []
        for var in self.header:
            if var in vars:
                index.append(1)
            elif "not " + var in vars:
                index.append(0)
            else:
                raise Exception("Invalid query.")

        return self.table.item(tuple(index))

    def __copy__(self):
        return Factor(self.header, self.table)

    def __str__(self):
        table_format = '{:<15}' * (len(self.header) + 1)
        result = [table_format.format(*(self.header.tolist() +
↪ ['probability']))]
        for index, value in np.ndenumerate(self.table):
            row = ['True' if boolean > 0 else 'False' for boolean in index]
            result.append(table_format.format(*(row + [value])))

        return '\n'.join(result) + '\n'

```

```

[3]: def inference(factors, query, hidden_list, evidence_list, print_step=True):
    def step_printer(factor):
        print(factor)

```

```

factor_list = [copy.copy(factor) for factor in factors]
for factor in factor_list:
    for evidence, value in evidence_list.items():
        factor.restrict(evidence, value)

product = np.prod(factor_list)

if print_step:
    print('Before summing out:')
    step_printer(product)

for hidden in hidden_list:
    product.sumout(hidden)
    print('Eliminate ' + hidden)
    step_printer(product)

product.normalize()

if set(query) == set(product.header):
    return product
else:
    raise Exception("Invalid query size.")

```

## 1.2 (2)

```

[4]: # Fraud / Trav
f1 = Factor(['Fraud', 'Trav'],
            [(0, 0, 1 - 0.004),
             (0, 1, 1 - 0.01),
             (1, 0, 0.004),
             (1, 1, 0.01)])

# Trav
f2 = Factor(['Trav'],
            [(0, 1 - 0.05),
             (1, 0.05)])

# FP / Fraud, Trav
f3 = Factor(['FP', 'Fraud', 'Trav'],
            [(0, 0, 0, 1 - 0.01),
             (0, 0, 1, 1 - 0.9),
             (0, 1, 0, 1 - 0.1),
             (0, 1, 1, 1 - 0.9),
             (1, 0, 0, 0.01),
             (1, 0, 1, 0.9),
             (1, 1, 0, 0.1),

```

```

        (1, 1, 1, 0.9)])

# OC
f4 = Factor(['OC'],
            [(0, 1 - 0.6),
             (1, 0.6)])

# IP / OC, Fraud
f5 = Factor(['IP', 'OC', 'Fraud'],
            [(0, 0, 0, 1 - 0.001),
             (0, 0, 1, 1 - 0.011),
             (0, 1, 0, 1 - 0.01),
             (0, 1, 1, 1 - 0.02),
             (1, 0, 0, 0.001),
             (1, 0, 1, 0.011),
             (1, 1, 0, 0.01),
             (1, 1, 1, 0.02)])

# CRP / OC
f6 = Factor(['CRP', 'OC'],
            [(0, 0, 1 - 0.001),
             (0, 1, 1 - 0.1),
             (1, 0, 0.001),
             (1, 1, 0.1)])

```

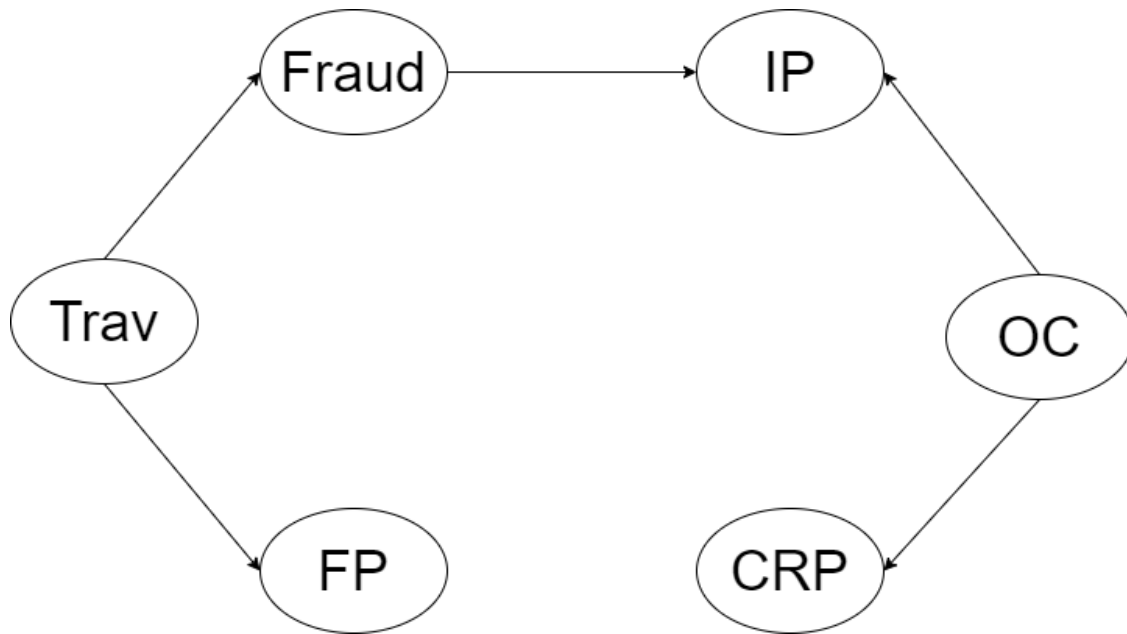
### 1.2.1 a)

```

[16]: from IPython.display import Image
      Image('./A3Q2a.png')

```

[16]:



### 1.2.2 b)

```
[6]: factor_list = [f1, f2, f3, f4, f5, f6]
ans = inference(factor_list, ['Fraud'], ['Trav', 'FP', 'IP', 'OC', 'CRP'],
    ↪dict())
print("P(Fraud) = " + str(ans.possibility(['Fraud'])))

ans = inference(factor_list, ['Fraud'], ['Trav', 'OC'], {'FP': True, 'IP':
    ↪False, 'CRP': True})
print("P(Fraud | FP, ~IP, CRP) = " + str(ans.possibility(['Fraud'])))
```

Before summing out:

CRP	FP	Fraud	IP	OC	Trav
probability					
False	False	False	False	False	False
0.3739461842952					
False	False	False	False	False	True
0.00197604198					
False	False	False	False	True	False
0.5007801348					
False	False	False	False	True	True
0.002646269999999996					
False	False	False	True	False	False
0.0003743205048					
False	False	False	True	False	True
1.97802e-06					

False	False	False	True	True	False
0.0050583851999999995					
False	False	False	True	True	True
2.6729999999999996e-05					
False	False	True	False	False	False
0.0013515990480000002					
False	False	True	False	False	True
1.9760219999999996e-05					
False	False	True	False	True	False
0.001809864					
False	False	True	False	True	True
2.645999999999999e-05					
False	False	True	True	False	False
1.5032952e-05					
False	False	True	True	False	True
2.1977999999999995e-07					
False	False	True	True	True	False
3.6936e-05					
False	False	True	True	True	True
5.399999999999999e-07					
False	True	False	False	False	False
0.0037772341848					
False	True	False	False	False	True
0.017784377820000002					
False	True	False	False	True	False
0.0050583851999999995					
False	True	False	False	True	True
0.023816430000000003					
False	True	False	True	False	False
3.7810152e-06					
False	True	False	True	False	True
1.7802180000000002e-05					
False	True	False	True	True	False
5.10948e-05					
False	True	False	True	True	True
0.00024057000000000004					
False	True	True	False	False	False
0.000150177672					
False	True	True	False	False	True
0.00017784198000000002					
False	True	True	False	True	False
0.00020109600000000003					
False	True	True	False	True	True
0.00023814					
False	True	True	True	False	False
1.670328e-06					
False	True	True	True	False	True
1.97802e-06					

False	True	True	True	True	False
4.104000000000001e-06					
False	True	True	True	True	True
4.86e-06					
True	False	False	False	False	False
0.0003743205048					
True	False	False	False	False	True
1.97802e-06					
True	False	False	False	True	False
0.0556422372					
True	False	False	False	True	True
0.00029403					
True	False	False	True	False	False
3.746952e-07					
True	False	False	True	False	True
1.98e-09					
True	False	False	True	True	False
0.0005620428					
True	False	False	True	True	True
2.97e-06					
True	False	True	False	False	False
1.3529520000000002e-06					
True	False	True	False	False	True
1.977999999999996e-08					
True	False	True	False	True	False
0.000201096					
True	False	True	False	True	True
2.939999999999999e-06					
True	False	True	True	False	False
1.5048e-08					
True	False	True	True	False	True
2.199999999999996e-10					
True	False	True	True	True	False
4.104e-06					
True	False	True	True	True	True
6e-08					
True	True	False	False	False	False
3.7810152e-06					
True	True	False	False	False	True
1.7802180000000002e-05					
True	True	False	False	True	False
0.0005620428					
True	True	False	False	True	True
0.0026462700000000005					
True	True	False	True	False	False
3.7848e-09					
True	True	False	True	False	True
1.7820000000000003e-08					

True	True	False	True	True	False
5.6772000000000005e-06					
True	True	False	True	True	True
2.6730000000000007e-05					
True	True	True	False	False	False
1.50328e-07					
True	True	True	False	False	True
1.7802000000000003e-07					
True	True	True	False	True	False
2.2344e-05					
True	True	True	False	True	True
2.646e-05					
True	True	True	True	False	False
1.672e-09					
True	True	True	True	False	True
1.98e-09					
True	True	True	True	True	False
4.5600000000000006e-07					
True	True	True	True	True	True
5.4e-07					

Eliminate Trav

CRP	FP	Fraud	IP	OC
probability				
False	False	False	False	False
0.3759222262752				
False	False	False	False	True
0.5034264048				
False	False	False	True	False
0.0003762985248				
False	False	False	True	True
0.005085115199999995				
False	False	True	False	False
0.0013713592680000002				
False	False	True	False	True
0.001836324				
False	False	True	True	False
1.5252732e-05				
False	False	True	True	True
3.7476e-05				
False	True	False	False	False
0.0215616120048				
False	True	False	False	True
0.0288748152				
False	True	False	True	False
2.1583195200000003e-05				
False	True	False	True	True
0.0002916648				



False	True	True	False	False
0.00032801965200000003				
False	True	True	False	True
0.000439236				
False	True	True	True	False
3.648348e-06				
False	True	True	True	True
8.964000000000001e-06				
True	False	False	False	False
0.0003762985248				
True	False	False	False	True
0.0559362672				
True	False	False	True	False
3.766752e-07				
True	False	False	True	True
0.0005650128				
True	False	True	False	False
1.3727320000000003e-06				
True	False	True	False	True
0.000204036				
True	False	True	True	False
1.5268e-08				
True	False	True	True	True
4.164e-06				
True	True	False	False	False
2.1583195200000003e-05				
True	True	False	False	True
0.0032083128000000003				
True	True	False	True	False
2.1604800000000003e-08				
True	True	False	True	True
3.2407200000000004e-05				
True	True	True	False	False
3.2834800000000005e-07				
True	True	True	False	True
4.8804e-05				
True	True	True	True	False
3.6520000000000002e-09				
True	True	True	True	True
9.96e-07				
Eliminate FP				
CRP	Fraud	IP	OC	probability
False	False	False	False	0.39748383828
False	False	False	True	0.53230122
False	False	True	False	
0.00039788172000000004				
False	False	True	True	0.0053767799999999999

False	True	False	False	
0.00169937892000000003				
False	True	False	True	
0.00227556000000000003				
False	True	True	False	1.890108e-05
False	True	True	True	
4.6439999999999996e-05				
True	False	False	False	
0.00039788172000000004				
True	False	False	True	0.05914458
True	False	True	False	
3.9828000000000004e-07				
True	False	True	True	0.00059742
True	True	False	False	
1.7010800000000004e-06				
True	True	False	True	0.00025284
True	True	True	False	1.892e-08
True	True	True	True	
5.1600000000000006e-06				

Eliminate IP

CRP	Fraud	OC	probability
False	False	False	0.39788172
False	False	True	0.537678
False	True	False	0.0017182800000000004
False	True	True	0.0023220000000000003
True	False	False	0.00039828000000000004
True	False	True	0.059742
True	True	False	1.7200000000000005e-06
True	True	True	0.000258

Eliminate OC

CRP	Fraud	probability
False	False	0.93555972
False	True	0.00404028
True	False	0.060140280000000004
True	True	0.00025971999999999996

Eliminate CRP

Fraud	probability
False	0.9957
True	0.0043

$P(\text{Fraud}) = 0.0043$

Before summing out:

Fraud	OC	Trav	probability
False	False	False	3.7810152e-06
False	False	True	1.7802180000000002e-05

False	True	False	0.0005620428
False	True	True	0.0026462700000000005
True	False	False	1.50328e-07
True	False	True	1.7802000000000003e-07
True	True	False	2.2344e-05
True	True	True	2.646e-05

Eliminate Trav

Fraud	OC	probability
False	False	2.1583195200000003e-05
False	True	0.0032083128000000003
True	False	3.2834800000000005e-07
True	True	4.8804e-05

Eliminate OC

Fraud	probability
False	0.0032298959952000005
True	4.9132348e-05

$P(\text{Fraud} \mid \text{FP}, \sim\text{IP}, \text{CRP}) = 0.014983813147541077$

### 1.2.3 c)

```
[7]: ans = inference(factor_list, ['Fraud'], ['OC'], {'FP': True, 'IP': False, 'CRP':
    ↪ True, 'Trav': True})
print("P(Fraud | FP, ~IP, CRP, Trav) = " + str(ans.possibility(['Fraud'])))
```

Before summing out:

Fraud	OC	probability
False	False	1.7802180000000002e-05
False	True	0.0026462700000000005
True	False	1.7802000000000003e-07
True	True	2.646e-05

Eliminate OC

Fraud	probability
False	0.0026640721800000005
True	2.663802e-05

$P(\text{Fraud} \mid \text{FP}, \sim\text{IP}, \text{CRP}, \text{Trav}) = 0.009899995919292979$

### 1.2.4 d)

```
[8]: ans = inference(factor_list, ['Fraud'], ['Trav', 'FP', 'OC', 'CRP'], {'IP':
    ↪True})
print("P(Fraud | IP) = " + str(ans.possibility(['Fraud'])))

cond_vars = ['FP', 'CRP']
for enum in range(2 ** 2):
    bin = "{0:02b}".format(enum)
    cond = {var: dig == '1' for var, dig in zip(cond_vars, bin)}
    cond['IP'] = True
    ans = inference(factor_list, ['Fraud'], ['Trav', 'OC'], cond)

    cond_to_print = [var if dig == '1' else '~' + var for var, dig in
    ↪zip(cond_vars, bin)]
    print('P(Fraud | ' + ', '.join(cond_to_print) + ') = ' + str(ans.
    ↪possibility(['Fraud'])))
```

Before summing out:

CRP	FP	Fraud	OC	Trav
probability				
False	False	False	False	False
0.0003743205048				
False	False	False	False	True
1.97802e-06				
False	False	False	True	False
0.0050583851999999995				
False	False	False	True	True
2.6729999999999996e-05				
False	False	True	False	False
1.5032952e-05				
False	False	True	False	True
2.1977999999999995e-07				
False	False	True	True	False
3.6936e-05				
False	False	True	True	True
5.399999999999999e-07				
False	True	False	False	False
3.7810152e-06				
False	True	False	False	True
1.7802180000000002e-05				
False	True	False	True	False
5.10948e-05				
False	True	False	True	True
0.00024057000000000004				
False	True	True	False	False
1.670328e-06				

False	True	True	False	True	
1.97802e-06					
False	True	True	True	False	
4.104000000000001e-06					
False	True	True	True	True	
4.86e-06					
True	False	False	False	False	
3.746952e-07					
True	False	False	False	True	
1.98e-09					
True	False	False	True	False	
0.0005620428					
True	False	False	True	True	
2.97e-06					
True	False	True	False	False	
1.5048e-08					
True	False	True	False	True	
2.199999999999996e-10					
True	False	True	True	False	
4.104e-06					
True	False	True	True	True	6e-08
True	True	False	False	False	
3.7848e-09					
True	True	False	False	True	
1.782000000000003e-08					
True	True	False	True	False	
5.677200000000005e-06					
True	True	False	True	True	
2.673000000000007e-05					
True	True	True	False	False	
1.672e-09					
True	True	True	False	True	
1.98e-09					
True	True	True	True	False	
4.560000000000006e-07					
True	True	True	True	True	
5.4e-07					
Eliminate Trav					
CRP	FP	Fraud	OC	probability	
False	False	False	False	0.0003762985248	
False	False	False	True		
0.005085115199999995					
False	False	True	False	1.5252732e-05	
False	False	True	True	3.7476e-05	
False	True	False	False		
2.1583195200000003e-05					
False	True	False	True	0.0002916648	

False	True	True	False	3.648348e-06
False	True	True	True	
8.964000000000001e-06				
True	False	False	False	3.766752e-07
True	False	False	True	0.0005650128
True	False	True	False	1.5268e-08
True	False	True	True	4.164e-06
True	True	False	False	
2.1604800000000003e-08				
True	True	False	True	
3.2407200000000004e-05				
True	True	True	False	
3.6520000000000002e-09				
True	True	True	True	9.96e-07

Eliminate FP

CRP	Fraud	OC	probability
False	False	False	0.00039788172000000004
False	False	True	0.005376779999999999
False	True	False	1.890108e-05
False	True	True	4.6439999999999996e-05
True	False	False	3.9828000000000004e-07
True	False	True	0.00059742
True	True	False	1.892e-08
True	True	True	5.1600000000000006e-06

Eliminate OC

CRP	Fraud	probability
False	False	0.005774661719999999
False	True	6.534107999999999e-05
True	False	0.00059781828
True	True	5.17892e-06

Eliminate CRP

Fraud	probability
False	0.006372479999999999
True	7.052e-05

$P(\text{Fraud} \mid \text{IP}) = 0.010945211857830204$

Before summing out:

Fraud	OC	Trav	probability
False	False	False	0.0003743205048
False	False	True	1.97802e-06
False	True	False	0.0050583851999999995
False	True	True	2.6729999999999996e-05
True	False	False	1.5032952e-05
True	False	True	2.1977999999999995e-07
True	True	False	3.6936e-05

True	True	True	5.399999999999999e-07
------	------	------	-----------------------

Eliminate Trav

Fraud	OC	probability
False	False	0.0003762985248
False	True	0.0050851151999999995
True	False	1.5252732e-05
True	True	3.7476e-05

Eliminate OC

Fraud	probability
False	0.0054614137247999996
True	5.2728731999999996e-05

$P(\text{Fraud} \mid \sim \text{FP}, \sim \text{CRP}) = 0.009562453711179572$

Before summing out:

Fraud	OC	Trav	probability
False	False	False	3.746952e-07
False	False	True	1.98e-09
False	True	False	0.0005620428
False	True	True	2.97e-06
True	False	False	1.5048e-08
True	False	True	2.1999999999999996e-10
True	True	False	4.104e-06
True	True	True	6e-08

Eliminate Trav

Fraud	OC	probability
False	False	3.766752e-07
False	True	0.0005650128
True	False	1.5268e-08
True	True	4.164e-06

Eliminate OC

Fraud	probability
False	0.0005653894752
True	4.179268e-06

$P(\text{Fraud} \mid \sim \text{FP}, \text{CRP}) = 0.007337600684545429$

Before summing out:

Fraud	OC	Trav	probability
False	False	False	3.7810152e-06
False	False	True	1.7802180000000002e-05
False	True	False	5.10948e-05
False	True	True	0.00024057000000000004
True	False	False	1.670328e-06
True	False	True	1.97802e-06
True	True	False	4.1040000000000001e-06

True	True	True	4.86e-06
------	------	------	----------

Eliminate Trav

Fraud	OC	probability	
False	False	2.1583195200000003e-05	
False	True	0.0002916648	
True	False	3.648348e-06	
True	True	8.964000000000001e-06	

Eliminate OC

Fraud	probability
False	0.0003132479952
True	1.2612348e-05

$P(\text{Fraud} \mid \text{FP}, \sim\text{CRP}) = 0.03870476498043534$

Before summing out:

Fraud	OC	Trav	probability
False	False	False	3.7848e-09
False	False	True	1.7820000000000003e-08
False	True	False	5.6772000000000005e-06
False	True	True	2.6730000000000007e-05
True	False	False	1.672e-09
True	False	True	1.98e-09
True	True	False	4.5600000000000006e-07
True	True	True	5.4e-07

Eliminate Trav

Fraud	OC	probability
False	False	2.1604800000000003e-08
False	True	3.2407200000000004e-05
True	False	3.6520000000000002e-09
True	True	9.96e-07

Eliminate OC

Fraud	probability
False	3.24288048e-05
True	9.99652e-07

$P(\text{Fraud} \mid \text{FP}, \text{CRP}) = 0.02990422220148673$

As is illustrated above, we are supposed to satisfy conditions  $\sim\text{FP}$ ,  $\text{CRP}$  to get the minimal probability. Thus, I should use domestic currency for the payment after making a computer related purchase. This will help reduce the probability from 1.094% to 0.734%, which is approximately 0.36%.



## 2 Part B

```
[9]: import math
```

```
[10]: train_data_path = './dataset/trainData.txt'
train_label_path = './dataset/trainLabel.txt'
test_data_path = './dataset/testData.txt'
test_label_path = './dataset/testLabel.txt'
words_path = './dataset/words.txt'

# Labels must start from 1 since we will minus 1 when storing
actual_labels = [1, 2]

labels = [label - 1 for label in actual_labels]
```

```
[11]: class Document:
    def __init__(self, label, doc_id=0):
        self.id = doc_id
        self.label = label - 1
        # Set labels to start from 0, it makes no difference
        self.word_list = set()

    def __contains__(self, word):
        return word in self.word_list

    def add_word(self, word):
        self.word_list.add(word - 1)
        # We store word - 1 here so that we only need to consider
        # the index issue when mapping word id to its word

def load_train_data():
    return load_data(train_data_path, train_label_path)

def load_test_data():
    return load_data(test_data_path, test_label_path)

def load_data(data_path, label_path):
    docs = []
    with open(label_path, 'r', encoding='utf-8') as file:
        for index, line in enumerate(file):
            doc_id = index + 1
            label = int(line.strip())
            docs.append(Document(label, doc_id))
```

```

with open(data_path, 'r', encoding='utf-8') as file:
    for line in file.readlines():
        [doc_id, word_id] = list(map(int, line.strip().split(' ')))
        docs[doc_id - 1].add_word(word_id)

return docs

def load_words(filename=words_path):
    word_map = dict()

    with open(filename, 'r', encoding='utf-8') as file:
        for index, word in enumerate(file):
            # Similar to above, we minus 1 for the word id
            word_map[index] = word.strip()

    return word_map

```

```

[12]: class NaiveBayes:
    def __init__(self, word_map, labels):
        self.word_set = set(word_map)
        self.poss_labels = labels

    def fit(self, docs):
        num_words = len(self.word_set)
        num_label = len(self.poss_labels)
        stat = np.zeros((num_words, num_label))

        for document in docs:
            for word in document.word_list:
                stat[word][document.label] += 1

        label_sum = np.sum(stat, axis=1, keepdims=True)
        word_sum = np.sum(stat, axis=0, keepdims=True)
        # prob_given_label[word][label] stores the value of Pr(word | label)
        self.prob_given_label = (stat + 1) / (label_sum + num_words)
        # prob_given_word[label][word] stores the value of Pr(label | word)
        self.prob_given_word = np.transpose((stat + 1) / (word_sum + num_label))

    def predict(self, word_list):
        p = dict()
        for label in self.poss_labels:
            p[label] = 1
            for word in self.word_set:
                if word in word_list:
                    p[label] *= self.prob_given_label[word][label]
                else:

```

```

        p[label] *= 1 - self.prob_given_label[word][label]

    return max(p, key=p.get)

    def __diff(self, word, label1, label2):
        return abs(math.log(self.prob_given_label[word][label1]) -
                    math.log(self.prob_given_label[word][label2]))

    def discriminative(self, n, label1=0, label2=1):
        rank = {word: self.__diff(word, label1, label2) for word in self.
        ↪word_set}
        return sorted(rank, key=rank.get, reverse=True)[: n]

```

```

[13]: def test(model, data):
        correct, incorrect = 0, 0
        for doc in data:
            label = doc.label
            pred = model.predict(doc)
            if pred == label:
                correct += 1
            else:
                incorrect += 1

        return correct / (correct + incorrect)

train_data = load_train_data()
test_data = load_test_data()
word_map = load_words()

nb = NaiveBayes(word_map, labels)
nb.fit(train_data)
disc_words = nb.discriminative(10)
print("Top 10 discriminative words:")
print([word_map[word] for word in disc_words])

print("Training accuracy: " + str(test(nb, train_data)))
print("Testing accuracy: " + str(test(nb, test_data)))

```

Top 10 discriminative words:

```
['christian', 'religion', 'atheism', 'books', 'christians', 'library',
'religious', 'libraries', 'novel', 'beliefs']
```

Training accuracy: 0.908

Testing accuracy: 0.7286666666666667

The assumption that all word features are independent is not reasonable. One example is shown above in the most discriminative 10 words that the same words may appear in an article in different forms, say `christian` and `christians`.

To take into account dependencies between words, we would better apply a stemming algorithm first, which will reduce all words to their roots so that different forms of the same words will be counted as only one.