INF4080 – 2019 FALL
NATURAL LANGUAGE PROCESSING

Jan Tore Lønning

Language models, and more

Lecture 4, 12 Sep

Today

- Language models
  - Smoothing Language models
- Naïve Bayes
- Naïve Bayes for text classification
  - Bernoulli model
  - Multinomial model
- Smoothing in Naïve Bayes
- Implementations of Naïve Bayes

Language models

Slides from Jurafsky & Martin 3. ed.

LM_4

Slide 1- 62, 65

Naive Bayes

Slides 41-56 from last week

Today

- Language models
  - Smoothing Language models
- Naïve Bayes
- Naïve Bayes for text classification
  - Bernoulli model
  - Multinomial model
- Smoothing in Naïve Bayes
- Implementations of Naïve Bayes

Laplace-smoothing



- MLE-estimate: $P(w_i) = \frac{c_i}{N}$
- Laplace-estimat: $P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$
- Lidstone-smoothing: add *k*, e.g. *0.5*: $\hat{P}(w_i) = \frac{c_i + k}{N + kV}$
- Nltk.NaiveBayesClassifier uses Lidstone (0.5) as default

Example, Bernoulli, 10,000 terms

Unsmoothed

- $\hat{P}(pitt = True|pos) = \frac{15}{959}$
- $\hat{P}(pitt = True|neg) = \frac{6}{941}$
- $\hat{P}(pitt = False|pos) = \frac{944}{959}$
- $\hat{P}(pitt = False|neg) = \frac{935}{941}$

Smoothed

- $\hat{P}(pitt = True|pos) = \frac{15+1}{959+2} = \frac{16}{961}$
- $\hat{P}(pitt = True|neg) = \frac{6+1}{941+2}$
- $\hat{P}(pitt = False|pos) = \frac{944+1}{959+2}$
- $\hat{P}(pitt = False|neg) = \frac{935+1}{941+2}$

Example, Multinomial, 10,000 terms

Unsmoothed

- $\hat{P}(w = pitt|pos) = \frac{31}{798{,}742}$
- $\hat{P}(w = pitt|neg) = \frac{25}{705{,}726}$

Smoothed

- $\hat{P}(w = pitt|pos) = \frac{31+1}{798{,}742+10{,}000} = \frac{32}{799{,}742}$
- $\hat{P}(w = pitt|neg) = \frac{25+1}{705{,}726+10{,}000} = \frac{26}{706{,}726}$

Doing it ourselves

- Possible to implement Naive Bayes classifiers ourselves
  - (That's not the case for all classifiers)
- Efficiency (and memory space) may be challenging
- Many available implementations. More efficient.
  - E.g. scikit-learn

Available learners

NLTK

- Bernoulli NB
- Decision trees
- (Python inefficient)

Scikit-learn

- Bernoulli NB
- Multinomial NB
- and many, many more
- Much more efficient

Data representation

Assume the following example

	4 different featues				Classes
feature	f1	f2	f3	f4	
type	cat	cat	Bool (num)	num	
Value set	a, b, c	x, y	True, False	0, 1, 2, 3, …	Class1, class2

Representation in NLTK

```
[({'f1': 'a', 'f2': 'z', 'f3': True, 'f4': 5}, 'class_1'),
 ({'f1': 'b', 'f2': 'z', 'f3': False, 'f4': 2}, 'class_2'),
 ({'f1': 'c', 'f2': 'x', 'f3': False, 'f4': 4}, 'class_1')]
```

3 training instances

4 features

class

One-hot encoding

feature 1				feature 2	
a	b	c		x	y
(1,0,0)	(0,1,0)	(0,0,1)		(1,0)	(0,1)

- Represent categorical variables as vectors/arrays of numerical variables

Representation in scikit: "one hot" encoding

NLTK

```
[({'f1': 'a', 'f2': 'z', 'f3': True, 'f4': 5}, 'class_1'),
 ({'f1': 'b', 'f2': 'z', 'f3': False, 'f4': 2}, 'class_2'),
 ({'f1': 'c', 'f2': 'x', 'f3': False, 'f4': 4}, 'class_1')]
```

3 training instances

4 features

class

scikit

```
X_train:
array([[ 1.,  0.,  0.,  0.,  1.,  1.,  5.],
       [ 0.,  1.,  0.,  0.,  1.,  0.,  2.],
       [ 0.,  0.,  1.,  1.,  0.,  0.,  4.]])
```

7 features

```
train_target: ['class_1', 'class_2', 'class_1']
```

3 training instances

3 corresponding classes

One-hot encoding		
a	b	c
[1, 0, 0]	[0, 1,]	[0, 0, 1

Converting a dictionary

- We can construct the data to scikit directly
- Scikit has methods for converting Python-dictionaries/NLTK-format to arrays

```
train_data = [inst[0] for inst in train]
train_target = [inst[1] for inst in train]
v = DictVectorizer()
X_train=v.fit_transform(train_data)

X_test=v.transform(test_data)
```

1. Constructs (=fit) repr. format
2. Transform

Transform
Use same v as for train

Multinomial NB in scikit

- We can construct the data to scikit directly
- Scikit has methods for converting text to bag of words arrays

```
train_data=["en rose er en rose",
                    "anta en rose er en fiol"]
v = CountVectorizer()
X_train=v.fit_transform(train_data)


print(X_train.toarray())
[[0 2 1 0 2]
[1 2 1 1 1]]
```