COMP3220 — Document Processing and Semantic Technologies

Week 03 Lecture 1: Introduction to Text Classification

Diego Mollá

Department of Computer Science
Macquarie University

COMP3220 2022H1

Programme

1. What is Text Classification
2. Statistical Classification with NLTK and Scikit-Learn
3. Advice on Machine Learning

Reading

- NLTK Book Chapter 6 “Learning to Classify Text”
  - Sections 1.1–1.3, 3
- Jurafsky & Martin (draft), Chapter 4. "Naive Bayes and Sentiment Classification".
  - Sections 4.7, 4.8

Some Useful Extra Reading

- Remaining of NLTK book chapter 6, Jurafsky & Martin chapter 4.

Programme

1. What is Text Classification
2. Statistical Classification with NLTK and Scikit-Learn
   - NLTK
   - Scikit-Learn
3. Advice on Machine Learning
   - Over-fitting

Text Classification

What is Text Classification?

Classify documents into one of a fixed predetermined set of categories.

- The number of categories is predetermined.
- The actual categories are predetermined.

Examples

- Spam detection.
- Email filtering.
- Classification of text into genres.
- Classification of names by gender.
- Classification of questions.

Example: Spam Filtering

Distinguish this

```
Date: Mon, 24 Mar 2008
From: XXX YYY <xxx@yahoo.com>
Subj: Re: Fwd: MSc
To: Mark Dras
<madras@ics.mq.edu.au>

Hi, Thanks for that. It would
fit me very well to start
2009, its actually much better
for me and I'm planning to
finish the project in one year
(8 credit points).
```

from this

```
Date: Mon, 24 Mar 2008
From: XXX YYY
<xxx@yahoo.co.in>
Subj: HELLO
To: madras@ics.mq.edu.au

HELLO, MY NAME IS STEPHINE IN
SEARCH OF A MAN WHO
UNDERSTANDS THE MEANING OF
LOVE AS TRUST AND FAITH IN
EACH OTHER RATHER THAN ONE WHO
SEES LOVE AS THE ONLY WAY OF
FUN ...
```

Classification Methods

Manual

- Web portals.
- Wikipedia.

Automatic

Hand coded rules

- e.g. 'Viagra' == SPAM.
- e.g. email filter rules.
- Fragile, breaks on new data.

Supervised learning

- Use an annotated corpus.
- Apply statistical methods.
- Greater flexibility.

Supervised Learning

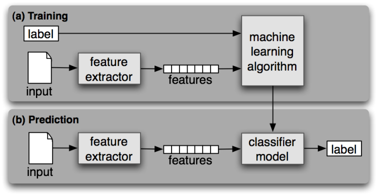
Given

Training data annotated with class information.

Goal

Build a model which will allow classification of new data.

Method



- Feature extraction: Convert samples into vectors.
- Training: Automatically learn a model.
- Classification: Apply the model on new data.

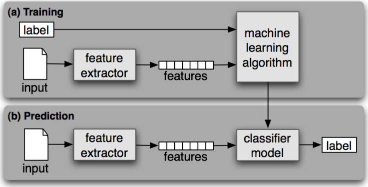
Programme

Programme

1. What is Text Classification
2. Statistical Classification with NLTK and Scikit-Learn
   - NLTK
   - Scikit-Learn
3. Advice on Machine Learning
   - Over-fitting

NLTK Features

- Statistical classifiers are not able to make sense of text.
- We need to feed them with our interpretation of the text.
- NLTK classifiers expect a dictionary of feature names and their values.



Example of a Simple Feature Extractor

```
def gender_features(word):
    return {'last_letter': word[-1]}
```

Example: Gender Classification

```
>>> import nltk
>>> from nltk.corpus import names
>>> import random
>>> random.seed(1234) # Fixed random seed to facilitate replicability
>>> names = ([(name, 'male') for name in m] +
             [(name, 'female') for name in f])
>>> random.shuffle(names)
>>> def gender_features(word):
  return {'last_letter': word[-1]}
>>> featuresets = [(gender_features(n), g) for n, g in names]
>>> train_set, devtest_set, test_set =
               featuresets[1000:], featuresets[500:1000], featuresets[:500]
>>> classifier = nltk.NaiveBayesClassifier.train(train_set)
>>> classifier.classify(gender_features('Neo'))
'male'
>>> classifier.classify(gender_features('Trinity'))
'female'
>>> nltk.classify.accuracy(classifier, test_set)
0.776
>>>
```

Note how the classifier is fed with the gender features and not with the actual names.

The Development Set I

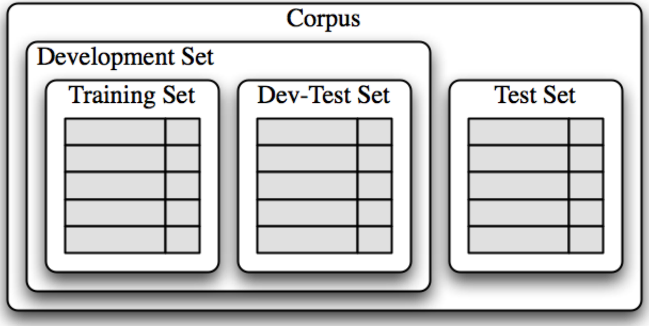
Important

Always test your system with data that has not been used for development (Why . . . ?)

Development and Test Sets

- Put aside a test set and don't even look at its contents.
- Use the remaining data as a development set.
  - Separate the development set into training and dev-test sets.
  - Use the training set to train the statistical classifiers.
  - Use the dev-test set to fine-tune the classifiers and conduct error analysis.
  - Use the test set for the final system evaluation once all decisions and fine-tuning have been completed.

The Development Set II



Error Analysis in our Gender Classifier

We use the dev-test set to conduct error analysis:

```
>>> nltk.classify.accuracy(classifier, devtest_set)
0.756
>>> errors = []
>>> for name, tag in devtest_names:
    guess = classifier.classify(gender_features(name))
    if tag == 'female' and guess == 'male':
        false_males.append(name)
    elif tag == 'male' and guess == 'female':
        false_males.append(name)
>>> len(false_males)
59
>>> len(false_females)
63
>>> for m in false_females[:5]:
        print(m)
Emmery
Winny
Alaa
Nate
Barrie
```

A Revised Gender Classifier

```
>>> def gender_features2(word):
        return {'suffix1': word[-1:],
                'suffix2': word[-2:]}

>>> train_set2 = [(gender_features2(n), g) for n, g in train_names]
>>> devtest_set2 = [(gender_features2(n),g) for n, g in devtest_names]
>>> classifier = nltk.NaiveBayesClassifier.train(train_set2)
>>> nltk.classify.accuracy(classifier, devtest_set2)
0.77
```

Beware of Over-fitting

If there are many features on a small corpus the system may over-fit.

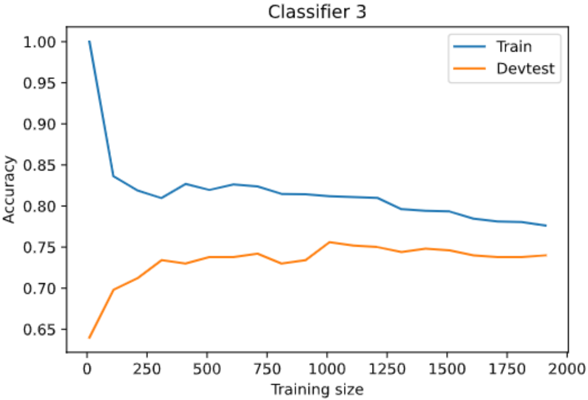
```
>>> def gender_features3(name):
      features = {}
      features['firstletter'] = name[0].lower()
      features['lastletter'] = name[-1]
      for letter in 'abcdefghijklmnopqrstuvwxyz':
         features['count(%s)' % letter] = name.lower().count(letter)
         features['has(%s)' % letter] = (letter in name.lower())
      return features

>>> gender_features3('John')
{'count(u)': 0, 'has(d)': False, 'count(b)': 0, 'count(w)': 0, ...}
>>> train_set3 = [(gender_features3(n), g) for n, g in train_names]
>>> devtest_set3 = [(gender_features3(n), g) for n, g in devtest_names]
>>> classifier = nltk.NaiveBayesClassifier.train(train_set3)
>>> nltk.classify.accuracy(classifier, devtest_set3)
0.758
>>> classifier2b = nltk.NaiveBayesClassifier.train(train_set2)
>>> nltk.classify.accuracy(classifier2b, devtest_set2)
0.77
```

Some types of classifiers are more sensitive to over-fitting than others.

Identifying Over-fitting

(see this week's lecture Jupyter notebook for the code that created this plot)



Programme

1. What is Text Classification
2. Statistical Classification with NLTK and Scikit-Learn
   - NLTK
   - Scikit-Learn
3. Advice on Machine Learning
   - Over-fitting

Text Classification in Scikit-Learn

- Scikit-learn includes a large number of statistical classifiers.
- All of these classifiers have a common interface.
- The features of a document set are represented as a matrix.
  - Each row represents a document.
  - Each column represents a feature.
- Scikit-learn provides several useful feature extractors for text:
  1. CountVectorizer returns a (sparse) matrix of word counts.
  2. TfidfVectorizer returns a (sparse) matrix of tf.idf values.

Gender Classifier in Scikit-Learn - Take 1

```
>>> from sklearn.naive_bayes import MultinomialNB
>>> def gender_features(word):
        "Return the ASCII value of the last two characters"
        return [ord(word[-2]), ord(word[-1])]
>>> featuresets = [(gender_features(n), g) for n, g in names]
>>> train_set, test_set = featuresets[500:], featuresets[:500
>>> train_X, train_y = zip(*train_set)
>>> classifier = MultinomialNB()
>>> classifier.fit(train_X, train_y)
>>> test_X, test_y = zip(*test_set)
>>> classifier.predict(test_X[:5])
array(['female', 'female', 'male', 'female', 'female'],
      dtype='|S6')
```

Gender Classification - Take 2

- In the previous slide we have used this code to encode the last two characters of a name:

```
def gender_features(word):
    "Return the ASCII value of the last two characters"
    return [ord(word[-2]), ord(word[-1])]
```

- This code is not entirely correct since it is representing characters as numbers.
- In general, non-numerical information is best represented using one-hot encoding.
- sklearn provides the following functions to produce one-hot-encoding vectors:
    - preprocessing.OneHotEncoding: from integers to one-hot vectors.
    - preprocessing.LabelBinarizer: from labels to one-hot vectors.

One-hot Encoding

- Suppose you want to encode five labels: 'a', 'b' ,'c', 'd', 'e'.
- Each label represents one element in the one-hot vector.
- Thus:
  - 'a' is represented as (1, 0, 0, 0, 0).
  - 'b' is represented as (0, 1, 0 ,0 ,0).
  - and so on.
- This is also called binarization or categorical encoding.

One-hot Encoding for Gender Classification

```
def one_hot_character(c):
  alphabet = 'abcdefghijklmnopqrstuvwxyz'
  result = [0]*(len(alphabet)+1)
  i = alphabet.find(c.lower())
  if i >= 0:
      result[i] = 1
  else:
      result[len(alphabet)] = 1 # out of the alphab

def gender_features(word):
    last = one_hot_character(word[-1])
    secondlast = one_hot_character(word[-2])
    return secondlast + last
```

Programme

Possible Problems with Machine Learning

- ML methods are typically seen as black boxes.
- Some methods are better than others for specific tasks but people tend to just try several and choose the one with best results.

Possible problems/mistakes you might face

1. Train and test are the same dataset (don't do this!).
2. The results on the test set are much worse than those on the dev-test set.
3. The results of both the test set and the training set are bad.
4. The train/test partition is not random.
5. The results on the test set are good but then the results on your real application problem are bad.

Partition into Training and Testing Set

What's wrong with this partition?

```
from nltk.corpus import names
m = names.words('male.txt')
f = names.words('female.txt')
names = ([(name, 'm') for name in m]+
         [(name, 'f') for name in f])
trainset = names[1000:]
devtest = names[500:1000]
testset = names[:500]
```

Advice

1. Make sure that the train and test sets have no bias.
2. Make sure that the train ant test sets are representative of your problem.

Partition into Training and Testing Set

What's wrong with this partition?

```
from nltk.corpus import names
m = names.words('male.txt')
f = names.words('female.txt')
names = ([(name, 'm') for name in m]+
         [(name, 'f') for name in f])
trainset = names[1000:]
devtest = names[500:1000]
testset = names[:500]
```

Advice

1. Make sure that the train and test sets have no bias.
2. Make sure that the train ant test sets are representative of your problem.

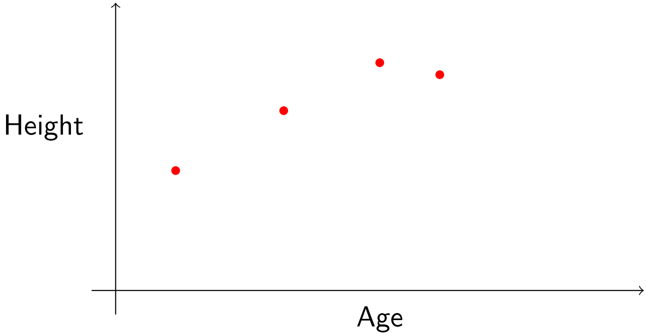
Randomised Partition

A better partition

```
from nltk.corpus import names
import random
m = names.words('male.txt')
f = names.words('female.txt')
names = ([(name, 'm') for name in m]+
         [(name, 'f') for name in f])
random.seed(1234)
random.shuffle(names)
trainset = names[1000:]
devtest = names[500:1000]
testset = names[:500]
```

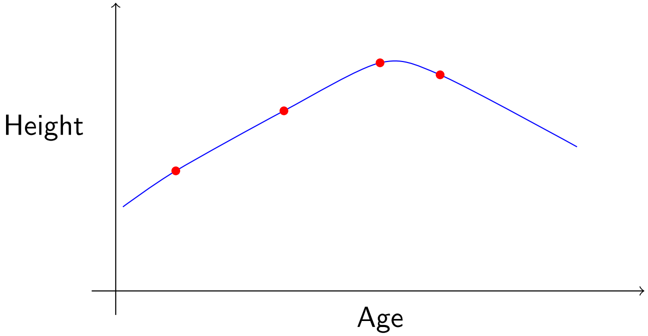
Programme

Why is machine learning hard?



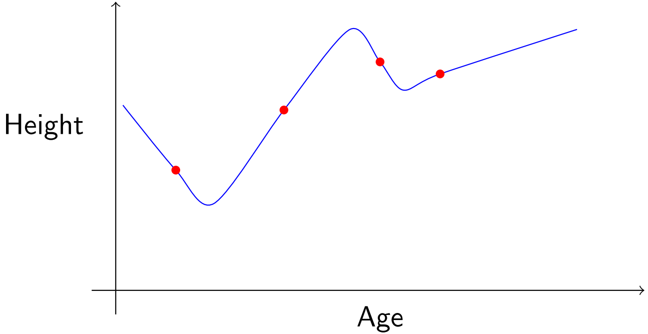
- There are an infinite number of curves that fit the data
  - even more if we don't require the curves to exactly fit (e.g., if we assume there's noise in our data).
- In general, more data would help us identify the correct curve better.

Why is machine learning hard?



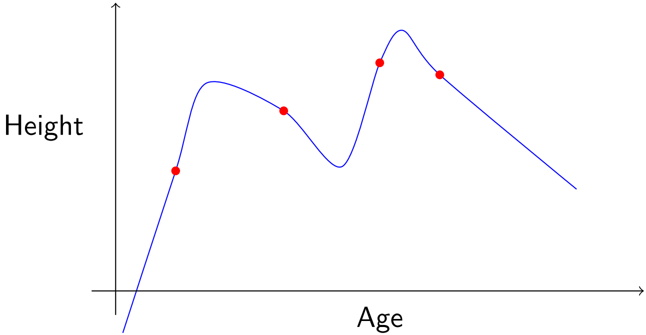
- There are an infinite number of curves that fit the data
  - even more if we don't require the curves to exactly fit (e.g., if we assume there's noise in our data).
- In general, more data would help us identify the correct curve better.

Why is machine learning hard?



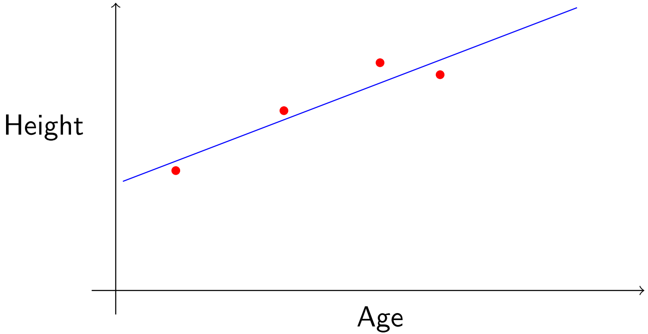
- There are an infinite number of curves that fit the data
    - even more if we don't require the curves to exactly fit (e.g., if we assume there's noise in our data).
- In general, more data would help us identify the correct curve better.

Why is machine learning hard?



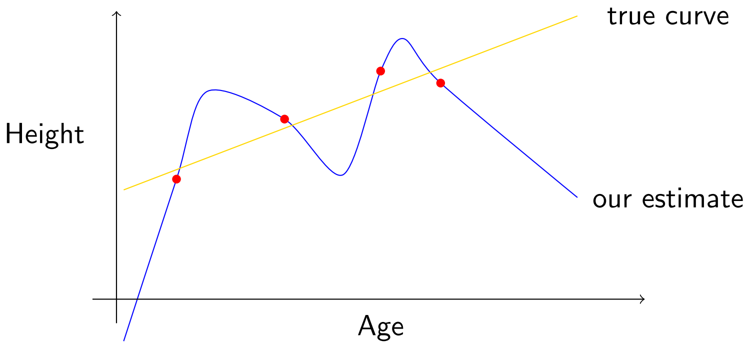
- There are an infinite number of curves that fit the data
  - even more if we don't require the curves to exactly fit (e.g., if we assume there's noise in our data).
- In general, more data would help us identify the correct curve better.

Why is machine learning hard?



- There are an infinite number of curves that fit the data
  - even more if we don't require the curves to exactly fit (e.g., if we assume there's noise in our data).
- In general, more data would help us identify the correct curve better.

Over-fitting the Training Data



- Over-fitting occurs when an algorithm learns a function that is fitting noise in the data.
- Diagnostic of over-fitting: performance on training data is much higher than performance on dev or test data.

Take-home Messages

1. Explain and demonstrate the need for separate training and test set.
2. Implement feature extractors for statistical classifiers in NLTK and Scikit-Learn.
3. Use NLTK's and Scikit-Learn's statistical classifiers.
4. Detect over-fitting.

What's Next

Week 4

- Deep Learning.

Reading

- Deep Learning book chapters 2 and 3.