Handout 12: Classification

1. Examples of classification tasks

- **a.** Spam filtering
- **b.** Assigning documents to topic areas
- **c.** Sentiment analysis
- **d.** Authorship identification
- **e.** Word-sense disambiguation
- **f.** Determining gender of name
- **g.** Part-of-speech tagging
- **h.** Sentence segmentation
- **i.** Dialogue act identification
- **j.** Textual entailment (RTE)

2. Data consists of **instances** with **labels**

```
>>> from nltk.corpus import names
>>> male = names.words('male.txt')
>>> female = names.words('female.txt')
>>> raw = [(n,'M') for n in male] + [(n,'F') for n in female]
>>> import random
>>> random.shuffle(raw)
>>> raw[:3]
[('Lisha', 'F'), ('Amory', 'M'), ('Laura', 'F')]
```

3. Represent **instances** as sets of **features**

a. Feature extractor

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

b. Apply to data

```
>>> data = [(gender_features(x), y) for (x,y) in raw]
>>> data[:3]
[({'last_letter': 'a'}, 'F'), ({'last_letter': 'y'}, 'M'),
({'last_letter': 'a'}, 'F')]
```

4. Split available data into training and test

```
>>> n = int(0.9 * len(data))
>>> train = data[:n]
>>> test = data[n:]
```

5. Train a classifier on the training data

a. Using Naive Bayes

```
>>> from nltk import NaiveBayesClassifier
>>> c = NaiveBayesClassifier.train(train)
```

b. Try it out

```
>>> c.classify(gender_features('Neo'))
'M'
>>> c.classify(gender_features('Trinity'))
'F'
>>> c.classify(gender_features('Morpheus'))
'M'
>>> c.classify(gender_features('Cypher'))
'M'
>>> c.classify(gender_features('Switch'))
'F'
```

6. Evaluate on the test set

```
>>> from nltk.classify import accuracy
>>> accuracy(c, test)
0.76226415094339628
```

7. Development

a. Don't use test set; split off part of training as **dev set**

b. Look at errors: missing features?

c. Informative features (based on training)

```
>>> c.show_most_informative_features()
Most Informative Features
    last_letter = 'k'                M : F      =     45.6 : 1.0
    last_letter = 'a'                F : M      =     43.4 : 1.0
    ...
```

8. Cross-validation

a. Instead of one train-test split, divide the data into 10 sections, let each in turn be the test data, average the performance

b. Also lets you gauge variance (how reliable is the accuracy estimate?)

9. Feature templates

a. Word-valued features

```
def doc_features (doc, vocab):
    docwords = set(doc)
    features = {}
```

```
    for w in vocab:
        name = 'contains({})'.format(w)
        features[name] = (w in docwords)
    return features
```

b. Suffix-valued features

```
name = 'endswith({})'.format(suffix)
features[name] = word.lower().endswith(suffix)
```

10. Words in context

a. Instead of the instance being a word, let it be a pair `(sent, i)`

b. Example feature:

```
if i > 0:
    features['prev-word'] = sent[i-1]
else:
    features['prev-word'] = '<START>'
```

11. Kinds of classifier

a. Naive Bayes

b. Decision tree

```
>>> c = nltk.DecisionTreeClassifier.train(train)
>>> print c.pseudocode(depth=4)
```

12. Sequential classification

a. E.g., part of speech tagging

b. Features: previous and following words, their suffixes

c. **Greedy:** features can include labels from previous instances. Classify to determine part of speech, it is then indelible.

d. **Sequence labeling:** choose best complete label sequence. E.g., Hidden Markov Models.

13. Precision and recall

a. Suppose 1 in 100 documents are good. (The **bias** is .01.) What is the **accuracy** of the trivial classifier that always says "bad"?

b. **Precision:** of the items the classifier returns, what percent are actually good?

c. **Type I error:** false positive, error of precision

d. **Recall:** of the good items, what percent does the classifier find?

e. **Type II error:** false negative, error of recall

f. What are precision/recall for the "just say no" classifier?

g. Just say yes?

h. Flip a coin?

14. Alternative: sensitivity and specificity

a. **Sensitivity:** same as recall. Of the good items, what percent does the classifier find?

b. **Specificity:** of the bad items, what percent does the classifier correctly reject?

c. What are specificity/sensitivity for "just say no"?

d. Just say yes?

e. Flip a coin?

f. Sensitivity/specificity not affected by bias.

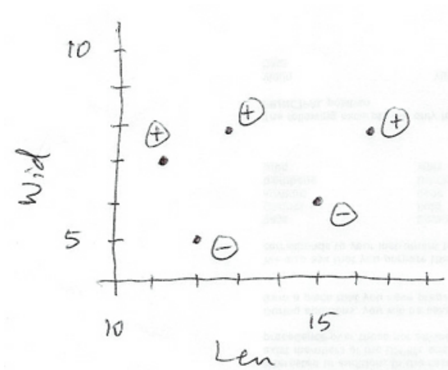
How classifiers work

15. Labeled instances

	Attributes				*Class Label*	
No.	*Length*		*Width*		*Species*	
1	10	*S*	7	*M*	*T*	*(tubifera)*
2	13	*M*	8	*L*	*T*	
3	17	*L*	8	*L*	*T*	
4	12	*M*	5	*S*	*F*	*(formosa)*
5	15	*L*	6	*M*	*F*	

16. Feature space (+ is tubifera, − is formosa)

a.



b. Classifier function is defined over whole space, not just at training points

17. Nearest-neighbor classifier

a. Test instance

	Attributes		*Class Label*
No.	*Length*	*Width*	*Species*
1	13 M	7 M	??

b. **Distance** = sum of squared differences in attribute values

c. Computing distances

$$
\begin{array}{r|lclcll}
1 & (10-13)^2+(7-7)^2 & = & 9+0 & = & 9 & \\
2 & (13-13)^2+(8-7)^2 & = & 0+1 & = & 1 & \leftarrow \\
3 & (17-13)^2+(8-7)^2 & = & 16+1 & = & 17 & \\
4 & (12-13)^2+(5-7)^2 & = & 1+4 & = & 5 & \\
5 & (15-13)^2+(6-7)^2 & = & 4+1 & = & 5 &
\end{array}
$$

d. Prediction = label of #2 = T

18. Naive Bayes classifier

a. Let's extend the training set

	Attributes		*Class Label*
No.	*Length*	*Width*	*Species*
1	S	M	T
2	M	L	T
3	L	L	T
4	M	L	T
5	M	L	T
6	L	M	T
7	M	S	F
8	L	M	F
9	S	S	F
10	S	L	F
11	S	M	F
12	L	S	F
13	M	S	F
14	M	S	F

b. Suppose that we **generate** the data by first choosing the label, then choosing each of the attribute values given the data.

$$p(S, M, T) = p(T) \cdot p(S_{\text{len}}|T) \cdot p(M_{\text{wid}}|T)$$

c. Estimate probabilities by relative frequency in training

c	$p(c)$
T	$6/14$
F	$8/14$

v	$p(v_{\text{len}}\|T)$	$p(v_{\text{wid}}\|T)$	$p(v_{\text{len}}\|F)$	$p(v_{\text{wid}}\|F)$
S	$1/6$	$0/6$	$3/8$	$5/8$
M	$3/6$	$2/6$	$3/8$	$2/8$
L	$2/6$	$4/6$	$2/8$	$1/8$

d. Computing prediction for test case

$$p(M, M, T) = p(T) \cdot p(M_{\text{len}}|T) \cdot p(M_{\text{wid}}|T) = \frac{6}{14} \cdot \frac{3}{6} \cdot \frac{2}{6} = \frac{1}{14} = \frac{4}{56}$$

$$p(M, M, F) = p(F) \cdot p(M_{\text{len}}|F) \cdot p(M_{\text{wid}}|F) = \frac{8}{14} \cdot \frac{3}{8} \cdot \frac{2}{8} = \frac{3}{56}$$

e. Prediction: T