2. Naive Bayes Classification
Machine Learning and Real-world Data (MLRD)

Paula Buttery
(based on slides created by Simone Teufel)

Lent 2019
Last session: we used a sentiment lexicon for sentiment classification

- Movie review sentiment classification was based on information in a sentiment lexicon.
- Possible problems with using a lexicon:
  - built using human intuition
  - required many hours of human labour to build
  - is limited to the words the humans decided to include
  - is static: *bad, sick* could have different meanings in different demographics

Today we will build a machine learning classifier for sentiment classification that makes decisions based on the data that it’s been exposed to.
What is Machine Learning?

- a program that learns from data.
- a program that adapts after having been exposed to new data.
- a program that learns implicitly from data.
- the ability to learn from data without explicit programming.
A Machine Learning approach to sentiment classification

- The sentiment lexicon approach relied on a fixed set of words that we made explicit reference to during classification.
- The words in the lexicon were decided independently from our data before the experiment.
- Instead we want to learn which words (out of all words we encounter in our data) express sentiment.
- That is, we want to implicitly learn how to classify from our data (i.e., use a machine learning approach).
Classifications are made from observations

First some terminology:

- **features** are easily observable (and not necessarily obviously meaningful) properties of the data.
- In our case the features of a movie review will be the words they contain.
- **classes** are the meaningful labels associated with the data.
- In our case the classes are our sentiments: POS and NEG.
- Classification then is function that maps from features to a target class.
- For us, a function mapping from the words in a review to a sentiment.
Probabilistic classifiers provide a distribution over classes

- Given a set of input features a probabilistic classifier returns the probability of each class.
- That is, for a set of observed features $O$ and classes $c_1, ..., c_n \in C$ gives $P(c_i|O)$ for all $c_i \in C$.
- For us $O$ is the set all the words in a review $\{w_1, w_2, ..., w_n\}$ where $w_i$ is the $i$th word in a review, $C = \{\text{POS, NEG}\}$.
- We get: $P(\text{POS}|w_1, w_2, ..., w_n)$ and $P(\text{NEG}|w_1, w_2, ..., w_n)$.
- We can decide on a single class by choosing the one with the highest probability given the features:

$$\hat{c} = \arg\max_{c \in C} P(c|O)$$
Today we will build a Naive Bayes Classifier

- Naive Bayes classifiers are simple probabilistic classifiers based on applying Bayes’ theorem.

Bayes Theorem:

\[
P(c|O) = \frac{P(c)P(O|c)}{P(O)}
\]

\[
c_{NB} = \arg\max_{c \in C} P(c|O) = \arg\max_{c \in C} \frac{P(c)P(O|c)}{P(O)} = \arg\max_{c \in C} P(c)P(O|c)
\]

- We can remove \(P(O)\) because it will be constant during a given classification and not affect the result of \(\arg\max\).
Naive Bayes classifiers assume feature independence

\[ c_{NB} = \arg\max_{c \in C} P(c|O) = \arg\max_{c \in C} \frac{P(c)P(O|c)}{P(O)} = \arg\max_{c \in C} P(c)P(O|c) \]

- For us, \( P(O|c) = P(w_1, w_2, \ldots, w_n|c) \)
- Naive Bayes makes a strong (naive) independence assumption between the observed features.

\[ P(O|c) = P(w_1, w_2, \ldots, w_n|c) \approx P(w_1|c) \times P(w_2|c) \times \cdots \times P(w_n|c) \]

so then:

\[ c_{NB} = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(w_i|c) \]
The probabilities we need are derived during training

\[ c_{NB} = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(w_i | c) \]

- In the **training** phase, we collect whatever information is needed to calculate \( P(w_i | c) \) and \( P(c) \).
- In the **testing** phase, we apply the above formula to derive \( c_{NB} \), the classifier’s decision.
- This is supervised ML because you use information about the classes during training.
A machine learning algorithm has two phases: training and testing.

**Training**: the process of making observations about some known data set

In *supervised* machine learning you use the classes that come with the data in the training phrase

**Testing**: the process of applying the knowledge obtained in the training stage to some new, unseen data

We never test on data that we trained a system on
Task 2: Step 0 – Split the dataset from Task 1

- From last time, you have 1800 reviews which you used for evaluation.
- We now perform a data split into 200 for this week’s testing (actually development) and 1600 for training.
- There are a further 200 reviews that you will use for more formal testing and evaluation in a subsequent session.
- You will compare the performance of the NB classifier you build today with the sentiment lexicon classifier.
- i.e. the NB classifier and the sentiment lexicon classifier will be evaluated on the same 200 reviews.
Task 2: Step 1 – Parameter estimation

- Write code that estimates $P(w_i|c)$ and $P(c)$ using the training data.

Maximum likelihood estimation (MLE) is a method of estimating the parameters of a statistical model given observations

$$\hat{P}(w_i|c) = \frac{\text{count}(w_i, c)}{\sum_{w \in V} \text{count}(w, c)}$$

where $\text{count}(w_i, c)$ is number of times $w_i$ occurs with class $c$ and $V$ is vocabulary of all words.

$$\hat{P}(c) = \frac{N_c}{N_{rev}}$$

where $N_c$ is number of reviews with class $c$ and $N_{rev}$ is total number of reviews

$$\hat{P}(w_i|c) \approx P(w_i|c) \quad \text{and} \quad \hat{P}(c) \approx P(c)$$
Task 2: Step 2 – Classification

In practice we use logs:

\[ c_{NB} = \underset{c \in C}{\operatorname{arg\,max}} \ logP(c) + \sum_{i=1}^{n} \ logP(w_i | c) \]

Problems you will notice:

- A certain word may not have occurred together with one of the classes in the training data, so the count is 0.
- Understand why this is a problem
- Work out what you could do to deal with it
Task 2: Step 3 – Smoothing

Add-one (Laplace) smoothing is the simplest form of smoothing:

\[
\hat{P}(w_i|c) = \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V}(\text{count}(w, c) + 1)} = \frac{\text{count}(w_i, c) + 1}{(\sum_{w \in V}\text{count}(w, c)) + |V|}
\]

where \( V \) is vocabulary of all distinct words, no matter which class \( c \) a word \( w \) occurred with.

See handbook and further reading:
Demonstrator Session today

- Get Task 1 ticked – Sentiment Lexicon Classifier
- Use the white boards to form a ticking queue
- If you miss a deadline you’ll need to ask your DoS to contact us