Supervised Learning in NLP: Bayes and Decision Trees

Daniel Marcu
marcu@isi.edu

1 General framework

Input: We are given a large corpus of texts whose words are manually labeled with word senses.

Goal: We are supposed to use the data in the corpus in order to learn to determine the senses of words in unseen texts.

Intuition: The sense of a word \( w \) is determined by the context, i.e., by the senses of the words that surround \( w \).

2 WSD using Bayes Classification

Each content word in the context of \( w \) contributes potentially useful information about the sense of \( w \).

In general, if a word \( w \) in a context \( c \) has \( k \) senses, \( s_1, s_2, \ldots, s_k \), we are interested in determining the sense \( s' \) so that

\[
s' = \underset{s}{\arg \max} \frac{P(c|s_k)}{P(c)}P(s_k)
\]

\[
= \underset{s}{\arg \max} \frac{P(c|s_k)P(s_k)}{P(c)}
\]

\[
= \underset{s}{\arg \max} \frac{\log P(c|s_k) + \log P(s_k)}{P(c)}
\]

Naive Bayes assumption: the attributes used for context description are conditionally independent.

Let \( c = v_1, v_2, \ldots, v_j \).

Using the Naive Bayes assumption, we get:

\[
P(c|s_k) = P(v_1, v_2, \ldots, v_j | s_k)
\]

\[
= P(v_1 | s_k)P(v_2 | s_k) \ldots P(v_j | s_k)
\]

\[
= \prod_{v_j \text{ in } c} P(v_j | s_k)
\]

1
Consequences of the Naive Bayes Assumption:

- The structure and order of the words is ignored.
- The presence of a word in a context is independent of the presence of another word. Obviously, this is wrong; our intuition tells us that \( P(\text{election} \in c | \text{president} \in c) > P(\text{election} \in c | \text{bread} \in c) \).

Maximum likelihood estimation:

\[
P(v_j | s_k) = \frac{\text{Count}(v_j, s_k)}{\text{Count}(s_k)} \quad P(s_k) = \frac{\text{Count}(s_k)}{\text{Count}(w)}
\]

A Naive Bayes WSD Algorithm

**Training** (Parameter estimation)

for all senses \( s_k \) of \( w \) do  
for all words \( v_j \) in the vocabulary do  
\[
P(v_k | s_k) = \frac{\text{Count}(v_i, s_k)}{\text{Count}(s_k)}
\]
end
end  
for all senses \( s_k \) of \( w \) do  
\[
P(s_k) = \frac{\text{Count}(s_k)}{\text{Count}(w)}
\]
end

**Disambiguation**

for all senses \( s_k \) of \( w \) do  
\[
\text{score}(s_k) = \log P(s_k)
\]
for all words \( v_j \) in the context window \( c \) do  
\[
\text{score}(s_k) = \text{score}(s_k) + \log P(v_j | s_k)
\]
end  
choose \( s' = \arg \max_{s_k} \text{score}(s_k) \) end

**Smoothing is essential**

Example: Laplace smoothing

\[
P(v_k | s_k) = \frac{\text{Count}(v_i, s_k) + 1}{\text{Count}(s_k) + |V|}
\]
Performance: Gale, Church, and Yarowsky (1992)

90% accuracy on duty, dry, land, language, position, sentence.

Examples of good clues (v_j’s with large $P(v_j|s_k)$) for two senses of drug:

- **medication**: prices, prescription, potent, increase, consumer, pharmaceutical;
- **illegal substance**: abuse, illicit, alcohol, cocaine, traffickers.

Bayes classifiers work well when the classes (word senses) to be learned are linearly separable.

**Naive Bayes as example of noisy-channel approach to WSD**

- Estimating $P(s_k | v_j)$ is impossible.
- So we assumed that it is the sense of the word that *generates* the context. Estimating $P(c | s_k) = \prod_{v_j \in c} P(v_j | s_k)$ is trivial.
  - The source is modeled by $P(s_k)$.
  - A noisy channel rewrites the source $s_k$ as a set of words (context $c$) with probability $\prod_{v_j \in c} P(v_j | s_k)$.
  - When we are given an unseen context $c$, we decode, i.e., we find the sense $s_k$ that is most likely to have generated that context ($arg\max_{s_k}[logP(c|s_k) + logP(s_k)]$).
- This is just one instance of noisy-channel application.
  - part of speech tagging
  - speech recognition
  - machine translation
  - summarization
3 Decision-tree-based approaches to WSD

Input: a set $f(\vec{x}) = y$ tuples were $\vec{x}$ are vectors of discrete values and $ys$ are the values to be learned.

Output: a decision tree that approximates the target function $f$.

WSD.names
medication, illegal_substance.

no_prices: 0, 1, 2.
no_prescription: 0, 1, 2.
no_abuse: 0, 1, 2.
no_cocaine: 0, 1, 2.
no_book: 0, 1, 2.
occurs_at_end_sentence: yes, no.
occurs_at_beginning_sentence: yes, no.

WSD.data
2, 0, 0, 0, 0, no, no, medication.
1, 0, 1, 0, no, no, illegal_substance.
1, 1, 0, 0, no, yes, medication.
0, 1, 0, 1, no, no, medication.
1, 0, 1, 0, 1, no, no, medication.
0, 0, 1, 0, no, no, illegal_substance.
0, 2, 0, 1, 0, no, no, medication.
0, 1, 0, 0, 1, no, no, medication.
0, 0, 1, 1, 0, no, no, illegal_substance.
0, 0, 0, 2, 0, no, no, illegal_substance.
0, 0, 0, 2, 1, no, no, illegal_substance.
0, 0, 0, 0, no, no, medication.

DECISION TREE:
no_prescription = 1: medication
no_prescription = 2: medication
no_prescription = 0:
|   no_cocaine = 1: illegal_substance
|   no_cocaine = 2: illegal_substance
|   no_cocaine = 0:
|   |   no_abuse = 0: medication
|   |   no_abuse = 1: illegal_substance
|   |   no_abuse = 2: medication
Appropriate problems

- Instances are represented by attribute-value pairs.
- The target function has discrete output values.
- The training data contains no errors.
- The training data may contain missing attribute values.

The ID3 algorithm

The ID3 algorithm builds a tree top-down, beginning with the question “which attribute should I test at the root of the tree”?

Each attribute is evaluated using a statistical test to determine how well it alone classifies the training examples.

\[
ID3(\text{Examples, Classes, Attributes})
\]

Create a Root node of the tree.

If all Examples are of Class \( \text{Class}_i \) then

- return a single-node Root, with the label \( \text{Class}_i \).

If Attributes is empty then

- return a single-node Root, with the label given by
  
  the most common value of Class in Examples

\( A \leftarrow \) the attribute in Attributes that \text{best classifies} Examples

The decision attribute for Root \( \leftarrow A \)

For each possible value \( v_i \) of \( A \)

Add a new branch below Root, corresponding to the test \( A = v_i \)

Let \( \text{Examples}_{v_i} \) be the subset of Examples that have value \( v_i \) for \( A \)

If \( \text{Examples}_{v_i} \) is empty then

- below this new branch add a leaf node = most common class

else

- below each branch add the subtree \( ID3(\text{Examples}_{v_i}, \text{Classes, Attributes}) \)

Which attribute is the best classifier?

- What is a good measure of the “worth” of an attribute? - The “Information Gain”: it measures how well a given attribute separates the training examples according to their target classification.

Entropy measures the homogeneity of a class of examples

Assume you have only two classes: \( \oplus \) and \( \ominus \).

\[
\text{Entropy}(S) = -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus
\]
\[\text{Entropy}(\text{medication}, 5\text{illegal-substance}) = -7/12\log_2(7/12) - 5/12\log_2(5/12) = 0.979\]

Entropy specifies the minimum number of bits of information needed to encode the classification of an arbitrary number of S-es.

If \(P_{\text{medication}} = 1\), then a receiver knows the result, so no message needs to be sent (\(H = 0\)).

If \(P_{\text{medication}} = 0.8\), a collection of messages can be encoded using on average < 1 bit by using shorter codes to encode “medication” examples and longer codes to encode “illegal-substance” examples.

**Information gain** is the reduction in entropy caused by partitioning the examples according to an attribute

\[\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|}\text{Entropy}(S_v)\]

\[S_v = \{ s \in S | A(s) = v \}\]

\(\text{Gain}(S, A)\) is the information provided about the target function value, given the value of some attribute \(A\).

**Example:**

\(\text{Values(\text{no-prices})} = 0, 1, 2\)

\[S = [7_m, 5_i]\]

\[S_0 = [4_m, 4_i]\]

\[S_1 = [2_m, 1_i]\]

\[S_2 = [1_m, 0_i]\]
\[
\text{Gain}(S, \text{no prices}) = \text{Entropy}(S) - \sum_{v \in \{0, 1, 2\}} \frac{|S_v|}{|S|} \text{Entropy}(S_v)
\]
\[
= \text{Entropy}(S) - \frac{8}{12} \text{Entropy}(S_0) - \frac{3}{12} \text{Entropy}(S_1) - \frac{1}{12} \text{Entropy}(S_2)
\]
\[
= 0.979 - \frac{8}{12}[-4/6\log_4/8 - 4/8\log_4/8] - \frac{3}{12}[-2/3\log_2/3 - 1/3\log_1/3]
- \frac{1}{12}[-1\log_1 - 0\log_0]
= 0.08
\]

Similarly, we can compute:

\[
\text{Gain}(S, \text{no prescription}) = 0.34
\]
\[
\text{Gain}(S, \text{no abuse}) = 0.16
\]
\[
\text{Gain}(S, \text{no cocaine}) = 0.04
\]
\[
\text{Gain}(S, \text{end sentence}) = 0.0
\]
\[
\text{Gain}(S, \text{beginning sentence}) = 0.06
\]

**DECISION TREE:**
no prescription = 1: medication [1/m. 0/i]
no prescription = 2: medication [3/m, 0/i]
no prescription = 0: [3/m, 5/i]

See [Quinlan, 1993, Mitchell, 1997].

**Practical issues**
- Error rates: training on .data files, testing on .test files.
- Cross validation.
- Learning curves.

**References**