BUILDING A SPAM FILTER USING NÀÎVE BAYES

Spam or not Spam: that is the question.

From: "*<lakworld6@hotmail.com>*
Subject: real estate is the only way... gem oalvgkay

Anyone can buy real estate with no money down
Stop paying rent TODAY!

There is no need to spend hundreds or even thousands for similar courses
I am 22 years old and I have already purchased 6 properties using the methods outlined in this truly INCREDIBLE ebook.

Change your life NOW!

=================================================================
Click Below to order:
http://www.wholesaledaily.com/sales/nmd.htm
=================================================================

Categorization/Classification Problems

- **Given:**
  - A description of an instance, \( x \in X \), where \( X \) is the instance language or instance space.
  - (Issue: how do we represent text documents?)
  - A fixed set of categories:
    \[ C = \{ c_1, c_2, \ldots, c_n \} \]

- **Determine:**
  - The category of \( x \): \( c(x) \in C \), where \( c(x) \) is a categorization function whose domain is \( X \) and whose range is \( C \).
  - (We want to automatically build categorization functions ("classifiers").)

EXAMPLES OF TEXT CATEGORIZATION

- **Categories = SPAM?**
  - "spam" / "not spam"

- **Categories = TOPICS**
  - "financial" / "sport" / "asia"

- **Categories = OPINION**
  - "like" / "hate" / "neutral"

- **Categories = AUTHOR**
  - "Shakespeare" / "Marlowe" / "Ben Jonson"
  - The Federalist papers

A Graphical View of Text Classification

Bayesian Methods for Classification

- **Uses Bayes theorem** to build a **generative Naïve Bayes model** that approximates how data is produced

\[
P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)}
\]

Where: \( C \): Categories, \( X \): Instance to be classified

- **Uses prior probability** of each category given no information about an item.

- **Categorization produces a posterior probability** distribution over the possible categories given a description of each instance.
**Maximum a posteriori (MAP) Hypothesis**
- Goodbye to that nasty normalization constant!!

\[
c_{\text{MAP}} = \arg\max_{c \in C} P(c \mid X) = \arg\max_{c \in C} \frac{P(D \mid c)P(c)}{P(X)} = \arg\max_{c \in C} P(X \mid c)P(c)
\]

No need to compute \(P(X)\)!!!

As \(P(X)\) is constant

**Maximum likelihood Hypothesis**

If all hypotheses are a priori equally likely, we only need to consider the \(P(X \mid c)\) term:

\[
c_{\text{ML}} = \arg\max_{c \in C} P(X \mid c)
\]

**Naïve Bayes Classifiers: Step 1**

Assume that instance \(x\) described by \(n\)-dimensional vector of attributes \(X = \{x_1, x_2, \ldots, x_n\}\)

\[
c_{\text{MAP}} = \arg\max_{c \in C} P(c \mid x_1, x_2, \ldots, x_n) = \arg\max_{c \in C} \frac{P(x_1, x_2, \ldots, x_n \mid c)P(c)}{P(x_1, x_2, \ldots, x_n)} = \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n \mid c)P(c)
\]

**Naïve Bayes Classifier: Step 2**

To estimate: \(c_{\text{MAP}} = \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n \mid c)P(c)\)

- \(P(c)\): Can be estimated from the frequency of classes in the training examples
- \(P(x_1, x_2, \ldots, x_n \mid c)\): Problem!!
  - \(O(|X|^n|C|)\) parameters required to estimate full joint prob. distribution

Solution: Naïve Bayes Conditional Independence Assumption:

\[
P(x_1, x_2, \ldots, x_n \mid c) = \prod_{j} P(x_j \mid c)
\]

**Learning the Model**

- First attempt: maximum likelihood estimates
  - Given training data for \(N\) individuals, where \(\text{count}(X=x)\) is the number of those individuals for which \(X=x\), e.g Flu=true
  - For each category \(c\) and each value \(x\) for a variable \(X\)
    \[
    \hat{P}(c) = \frac{\text{count}(C=c)}{N}
    \]
    \[
    \hat{P}(x \mid c) = \frac{\text{count}(X=x, C=c)}{\text{count}(C=c)}
    \]

**Naïve Bayes Classifier for Binary variables**

- Conditional Independence Assumption: features are independent of each other given the class:
  \[
P(X_1, \ldots, X_n \mid C) = P(X_1 \mid C) \cdot P(X_2 \mid C) \cdots P(X_n \mid C)
\]
**Problem with Max Likelihood for Naïve Bayes**

- Max likelihood for Naïve Bayes:
  \[
  \hat{P}(x|c) = \frac{\text{count}(X = x, C = c) + 1}{\text{count}(C = c) + |X|}
  \]
- Slightly better version:
  \[
  \hat{P}(x|c) = \frac{\text{count}(X = x, C = c) + \alpha}{\text{count}(C = c) + \alpha |X|}
  \]
  where \(\alpha\) is the extent of smoothing.

**Add-1” Laplace Smoothing to Avoid Overfitting**

- Zero probabilities overwhelm any other evidence!

**Using Naïve Bayes Classifiers to Classify Text:**

- As a generative model:
  1. Randomly pick a category \(c\) according to \(P(c)\)
  2. For a document of length \(N\), for each word \(i\):
     1. Generate \(w_i\) according to \(P(w|c)\)
     \[
     P(c, D = \langle w_1, w_2, \ldots, w_N \rangle) = P(c) \prod_{i=1}^N P(w_i|c)
     \]
- Note that word order really doesn't matter here
  - Uses same parameters for each position
  - Result is bag of words model
    - Views document not as an ordered list of words, but as a multiset

**Naïve Bayes: Learning (First attempt)**

- From training corpus, extract Vocabulary
- Calculate required estimates of \(P(c)\) and \(P(w|c)\) terms.
  - For each \(i\) in \(C\) do
    \[
    P(c) = \frac{\text{count}_{\text{docs}}(C = c)}{\text{count}_{\text{docs}}(|X|)}
    \]
    where \(\text{count}_{\text{docs}}(x)\) is the number of documents for which \(x\) is true.
  - For each word \(w\) in Vocabulary and \(c \in C\), where \(\text{count}_{\text{tokens}}(x)\) is the number of tokens over all documents for which \(x\) is true of that document and that token...
    \[
    P(w|c) = \frac{\text{count}_{\text{tokens}}(W = w, C = c)}{\text{count}_{\text{tokens}}(C = c)}
    \]

**Naïve Bayes: Learning (Second attempt)**

- Laplace smoothing must be done over the vocabulary items.
  - We can assume we have at least one instance of each category, so we don't need to smooth these.
  - Assume a single new word UNK, that occurs nowhere within the training document set.
  - Map all unknown words in documents to be classified (test documents) to UNK.
  - For \(0 \leq \alpha \leq 1\),
    \[
    P(w|c) = \frac{\text{count}_{\text{tokens}}(W = w, C = c) + \alpha}{\text{count}_{\text{tokens}}(C = c) + \alpha |V| + 1}
    \]

**Naïve Bayes: Classifying**

- Compute \(c_{\text{arg}}\) using either
  \[
  c_{\text{arg}} = \arg \max_c P(c) \prod_i P(w_i|c)
  \]
  \[
  c_{\text{arg}} = \arg \max_c P(c) \prod w \text{ in doc} P(w|c)^{\text{count}(w)}
  \]
  where \(\text{count}(w)\) = the number of times word \(w\) occurs in doc
  (The two are equivalent.)
PANTEL AND LIN: SPAMCOP

- Uses a Naive Bayes classifier
- M is spam if \( P(\text{Spam}|M) > P(\text{NonSpam}|M) \)
- Method
  - Tokenize message using Porter Stemmer
  - Estimate \( P(x_k|C) \) using m-estimate (a form of smoothing)
  - Remove words that do not satisfy certain conditions
  - Train: 160 spams, 466 non-spams
  - Test: 277 spams, 346 non-spams
- Results: ERROR RATE of 4.33%
  - Worse results using trigrams

Naive Bayes is (was) Not So Naive

- Naive Bayes: First and Second place in KDD-CUP 97 competition, among 16 (then) state of the art algorithms
  - Goal: Financial services industry direct mail response prediction model: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.
  - A good dependable baseline for text classification
    - But not the best by itself!
  - Optimal if the Independence Assumptions hold:
    - If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
  - Very Fast:
    - Learning with one pass over the data;
    - Testing linear in the number of attributes, and document collection size
  - Low Storage requirements

Engineering: Underflow Prevention

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since \( \log(xy) = \log(x) + \log(y) \), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

\[
c_{NB} = \arg \max_{c_j \in C} \log c_j + \sum_{i \in \text{positions}} \log P(w_i | c_j)
\]

REFERENCES