A spam classifier based on Bayes network

Alberto Franzin, Fabio Palese

Sistemi Intelligenti

January 15th, 2013

Introduction

The Bayesian approach

- Bayes rule:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

defines the *a posteriori* probability of event A, knowing the event B has already occurred.

The Bayesian approach

- Bayes rule:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

  defines the *a posteriori* probability of event A, knowing the event B has already occurred.
- In other words, we can estimate the probability of an hypothesis, given that we know the consequences.

The Bayesian approach

- Bayes rule:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

  defines the *a posteriori* probability of event A, knowing the event B has already occurred.
- In other words, we can estimate the probability of an hypothesis, given that we know the consequences.
- This has led to two different interpretations of the theorem.

The Bayesian approach

- Bayes rule:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

The Bayesian approach

- Bayes rule:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A|B)$ is the *a posteriori* probability

THE BAYESIAN APPROACH

- Bayes rule:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A|B)$ is the *a posteriori* probability
- $P(B|A)$ is the *likelihood*

The Bayesian approach

- Bayes rule:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A|B)$ is the *a posteriori* probability
- $P(B|A)$ is the *likelihood*
- $P(B|A)P(A)$ is the *prior* probability

The Bayesian approach

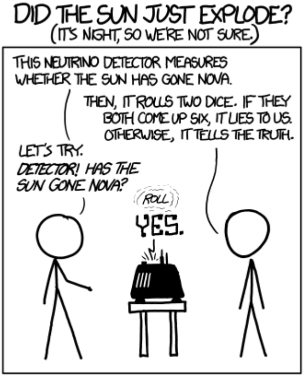
- Bayes rule:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A|B)$ is the *a posteriori* probability
- $P(B|A)$ is the *likelihood*
- $P(B|A)P(A)$ is the *prior* probability
- $P(B) = \sum_{a \in A} P(B|A = a)P(A = a)$ is the *total* probability

THE BAYESIAN APPROACH

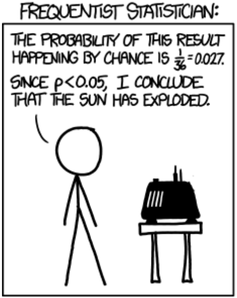
Frequentists vs. Bayesians



from http://xkcd.com/1132, see also
http://en.wikipedia.org/wiki/Sunrise_problem

The Bayesian approach

Frequentists vs. Bayesians



from http://xkcd.com/1132, see also
http://en.wikipedia.org/wiki/Sunrise_problem

The frequentist relies on the theoretical probability of the events.

The Bayesian approach

Frequentists vs. Bayesians

from http://xkcd.com/1132, see also
http://en.wikipedia.org/wiki/Sunrise_problem

The bayesian observes the past events occurred, and adapts the probability accordingly.

What it is

A Bayes network is a way to describe causal relationships between events.

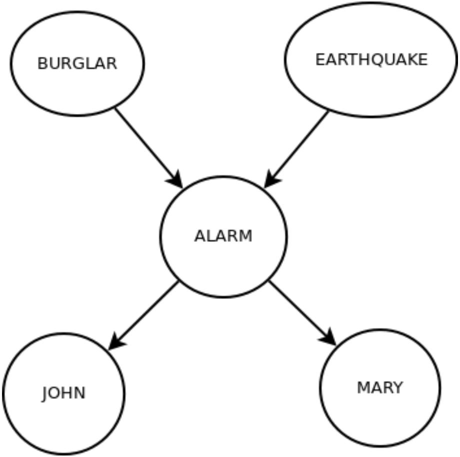
- Nodes = events
- (Directed) Edges = causal relationship

What it is

A Bayes network is a way to describe causal relationships between events.

- Nodes = events
- (Directed) Edges = causal relationship
- Two nodes are connected by an edge: the child of an arc is influenced by its ancestor in a probabilistic way

AN EXAMPLE



Conditional independence

- If
$$P(A|B,C) = P(A|B)$$
then we say that B and C are *conditionally independent*.

Conditional independence

- If
$$P(A|B,C) = P(A|B)$$
then we say that B and C are *conditionally independent*.
- Note that *conditional independence* $\neq$ *independence*

Conditional independence

- If
$$P(A|B,C) = P(A|B)$$
then we say that B and C are *conditionally independent*.
- Note that *conditional independence* $\neq$ *independence*
- Explaining away: if we know that one possible cause of the event has happened, this may *explain away* the event, being all the other causes less probable once we know the one that happened.

Naive Bayes

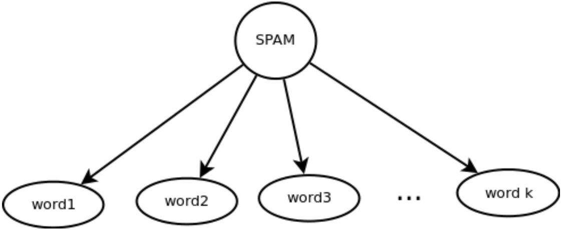
- Computing all the probabilities in a Bayesian network requires exponential time. We introduce the assumption of independence among variables.

Naive Bayes

- Computing all the probabilities in a Bayesian network requires exponential time. We introduce the assumption of independence among variables.
- It is called *naive*, since it's often unrealistic, but it yields good results.

Naive Bayes

- Computing all the probabilities in a Bayesian network requires exponential time. We introduce the assumption of independence among variables.
- It is called *naive*, since it's often unrealistic, but it yields good results.
- In spam classification:



Naive Bayes for spam classification

Algorithm:

- Training

Naive Bayes for spam classification

Algorithm:

- Training
- Validation

Naive Bayes for spam classification

Algorithm:

- Training
- Validation
- Testing

Naive Bayes for spam classification

Formulas:

- For each word:

$$P_{word|spam} = \frac{\text{\# occurrences of word in spam mails}}{\text{\# total occurrences of word}}$$

Naive Bayes for spam classification

Formulas:

- For each word:

$$P_{word|spam} = \frac{\text{\# occurrences of word in spam mails}}{\text{\# total occurrences of word}}$$

- Final for spam is:

$$P_{spam} = \prod_{words \in mail} P_{spam|word}$$

Naive Bayes for spam classification

Formulas:

- For each word:

$$P_{word|spam} = \frac{\text{\# occurrences of word in spam mails}}{\text{\# total occurrences of word}}$$

- Final for spam is:

$$P_{spam} = \prod_{words \in mail} P_{spam|word}$$

- same for ham

Naive Bayes for spam classification

Formulas:

- For each word:

$$P_{word|spam} = \frac{\text{\# occurrences of word in spam mails}}{\text{\# total occurrences of word}}$$

- Final for spam is:

$$P_{spam} = \prod_{words \in mail} P_{spam|word}$$

- same for ham
- Outcome is the class that maximizes the probabilities.

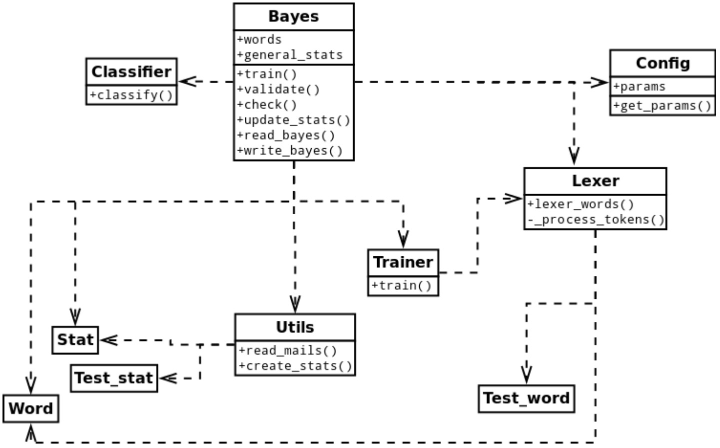
SpamBayes

- SpamBayes: applying NaiveBayes to spam classification
  Python with Ply and BeautifulSoup
  dataset: SpamAssassin archive
  Code and documentation at `http://code.google.com/p/sist-int-2012project/`

SpamBayes

- SpamBayes: applying NaiveBayes to spam classification
  Python with Ply and BeautifulSoup
  dataset: SpamAssassin archive
  Code and documentation at `http://code.google.com/p/sist-int-2012project/`
- spam: undesidered mail
  ham: valid mail

SpamBayes


Bayes
+words
+general_stats
+train()
+validate()
+check()
+update_stats()
+read_bayes()
+write_bayes()
Classifier
+classify()
Config
+params
+get_params()
Lexer
+lexer_words()
-_process_tokens()
Trainer
+train()
Utils
+read_mails()
+create_stats()
Stat
Test_stat
Word
Test_word

Notes on implementation

- Smoothing:

$$P_{word|spam} = \frac{\#\text{ occurrences of word in spam mails} + k}{\#\text{ total occurrences of the word} + |C| \times k}$$

Notes on implementation

- Smoothing:

$$P_{word|spam} = \frac{\#\text{ occurrences of word in spam mails} + k}{\#\text{ total occurrences of the word} + |C| \times k}$$

- Calculations can be simplified: some words bring little contribution to the mail status

Notes on implementation

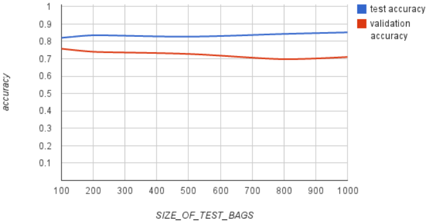
- Smoothing:

$$P_{word|spam} = \frac{\#\text{ occurrences of word in spam mails} + k}{\#\text{ total occurrences of the word} + |C| \times k}$$

- Calculations can be simplified: some words bring little contribution to the mail status
- Several parameters to tune: we describe the more relevant ones

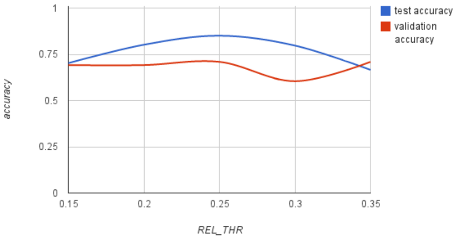
PARAMETERS

Size of training/validation/test sets:



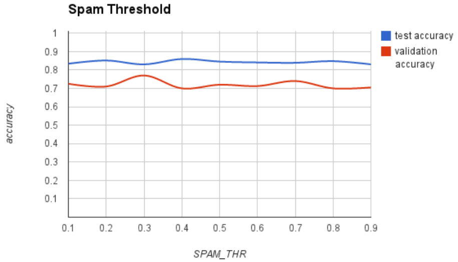
Parameters

Relevance threshold:



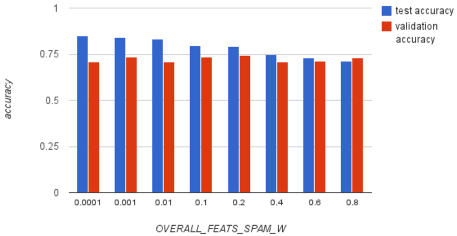
Parameters

"Spamicity" threshold:



PARAMETERS

Feature statistic threshold:



RESULTS

- About the dataset:

RESULTS

- About the dataset:
  - General features

Results

- About the dataset:
  - General features
  - Single words

Results

- About the dataset:
  - General features
  - Single words
- About the classification:

Results

- About the dataset:
  - General features
  - Single words
- About the classification:
  - Accuracy

RESULTS

- About the dataset:
  - General features
  - Single words
- About the classification:
  - Accuracy
  - How to improve?

Final

Any questions?