Klassifikation mit Naive Bayes

Benjamin Roth
(Vielen Dank an Helmut Schmid für Teile der Folien)

Centrum für Informations- und Sprachverarbeitung
Ludwig-Maximilian-Universität München
beroth@cis.uni-muenchen.de

Mathematische Grundlagen

Zufallsexperiment

In der Statistik geht es um die Wahrscheinlichkeit von Ereignissen:

Beispiel: Wie wahrscheinlich ist es, sechs Richtige im Lotto zu haben?

Zufallsexperiment: Experiment (Versuch) mit mehreren möglichen Ausgängen (*Wurf von zwei Würfeln*)

Ergebnis: Resultat eines Experimentes (*3 Augen auf Würfel 1 und 4 Augen auf Würfel 2*)

Ergebnisraum Ω: Menge aller möglichen Ergebnisse

Ereignis: Teilmenge des Ergebnisraumes (*7 Augen auf zwei Würfeln*)

Stichprobe: Folge von Ergebnissen bei einem wiederholten Experiment

Wahrscheinlichkeitsverteilung

Wahrscheinlichkeitsverteilung: Funktion, die jedem Ergebnis einen Wert zwischen 0 und 1 zuweist, so dass

$$\sum_{o \in \Omega} p(o) = 1$$

Die Wahrscheinlichkeit eines Ereignisses ist die Summe der Wahrscheinlichkeiten der entsprechenden Ergebnisse.

Beispiel:

Wahrscheinlichkeit, dass die Zahl der Augen beim Wurf eines Würfels gerade ist

Bedingte und A-priori Wahrscheinlichkeit

Bedingte Wahrscheinlichkeit: Wahrscheinlichkeit eines Ereignisses A, wenn das Ereignis B bekannt ist:

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

Beispiel: Wahrscheinlichkeit, dass die Augenzahl eines Würfels gerade ist, wenn die Augenzahl größer als 3 ist

A Priori Wahrscheinlichkeit $P(A)$**:** Wahrscheinlichkeit des Ereignisses A ohne das Ereignis B zu kennen

Zufallsvariablen

Zufallsvariable: Funktion, welche jedem Ergebnis eine reelle Zahl zuweist.

Beispiel: Abbildung der Noten *sehr gut, gut, befriedigend, ausreichend, mangelhaft, ungenügend* auf die Zahlen 1, 2, 3, 4, 5, 6

Eine Zufallsvariable wird als diskret bezeichnet, wenn sie nur endlich viele oder abzählbar unendlich viele Werte annimmt.

Das obige Beispiel beschreibt also eine diskrete Zufallsvariable.

Wahrscheinlichkeit eines Wertes x der Zufallsvariablen X:

$$P(X = x) = p(x) = P(A_x)$$

Eine Zufallsvariable mit nur den Werten 0 und 1 nennt man **Bernoulliexperiment**.

Gemeinsame Verteilungen und Randverteilungen

Die **gemeinsame Verteilung** zweier Zufallsvariablen X und Y:

$$p(x, y) = P(X = x, Y = y) = P(A_x \cap A_y)$$

Die **Randverteilung** von zwei Zufallsvariablen X und Y:

$$p_X(x) = \sum_y p(x, y) \quad p_Y(y) = \sum_x p(x, y)$$

Unabhängigkeit: Die Zufallsvariablen X und Y sind statistisch unabhängig, falls gilt:

$$p(x, y) = p_X(x) p_Y(y)$$

Beispiel: Beim Wurf zweier Würfel sind deren Augenzahlen statistisch voneinander unabhängig.

Wichtige Regeln

Kettenregel: Einige gemeinsame Wahrscheinlichkeit kann in ein Produkt bedingter Wahrscheinlichkeiten umgewandelt werden.

$$\begin{aligned} P(A_1 \cap A_2 \cap \ldots \cap A_n) &= P(A_1)P(A_2|A_1)\ldots P(A_n|A_1 \cap \ldots \cap A_{n-1}) \\ &= \prod_{i=1}^{n} P(A_i|A_1 \cap \ldots \cap A_{i-1}) \end{aligned}$$

Theorem von Bayes: erlaubt es, eine bedingte Wahrscheinlichkeit "umzudrehen"

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

Wahrscheinlichkeitsschätzung

$$\tilde{p}(x) = \frac{n(x)}{N}$$

Die **Relative Häufigkeit** $n(x)/N$ ist die Zahl der Vorkommen (*counts*) $n(x)$ eines Ereignisses x geteilt durch die Stichprobengröße n.

Für zunehmende Stichprobengröße n, konvergiert die relative Häufigkeit zu der tatsächlichen Wahrscheinlichkeit eines Ereignisses.

genauer: Die Wahrscheinlichkeit, dass die relative Häufigkeit um mehr als ϵ von der tatsächlichen Wahrscheinlichkeit abweicht, konvergiert für zunehmende Stichprobengröße gegen 0.

Wahrscheinlichkeitsschätzung durch relative Häufigkeit

Beispiel:

- Zufallsereignis: Wortvorkommen ist ein bestimmtes Wort
- $n(x)$: Anzahl der Vorkommen (*counts*) des Wortes in einem Corpus
- N: Anzahl aller Wortvorkommen im Corpus.

Wort	n(Wort)	$\tilde{p}$(Wort)
meet		
deadline		
single		
...		

reminder
deadline
meet
thanks

thanks
for
tip

deadline
approaching

hot
stock
tip

meet
hot
single

Wahrscheinlichkeitsschätzung durch relative Häufigkeit

Beispiel:

- Zufallsereignis: Wortvorkommen ist ein bestimmtes Wort
- $n(x)$: Anzahl der Vorkommen (*counts*) des Wortes in einem Corpus
- N: Anzahl aller Wortvorkommen im Corpus.

Wort	n(Wort)	$\tilde{p}$(Wort)
meet	2	$\frac{2}{15} \approx 0.133$
deadline	2	$\frac{2}{15} \approx 0.133$
single	1	$\frac{1}{15} \approx 0.067$
...		

reminder
deadline
meet
thanks

thanks
for
tip

deadline
approaching

hot
stock
tip

meet
hot
single

Relative Häufigkeit für bedingte Wahrscheinlichkeiten

$$\tilde{p}(x|y) = \frac{n(x, y)}{n_y}$$

Auch bedingte Wahrscheinlichkeiten können anhand von relativen Häufigkeiten geschätzt werden.

$n(x, y)$ ist hier die Zahl der gemeinsamen Vorkommen der Ereignisse x und y.

n_y ist die Anzahl aller Vorkommen des Ereignisses y.

Es gilt: $n_y = \sum_{x'} n(x', y)$

Relative Häufigkeit für bedingte Wahrscheinlichkeiten

- Zufallsereignis x: Wortvorkommen ist ein bestimmtes Wort
- Zufallsereignis y: Wortvorkommen ist in Email einer bestimmten Kategorie, z.B. HAM oder SPAM (HAM="kein Spam")
- $n(x, y)$: Anzahl der Wortvorkommen in Emails einer Kategorie im Corpus

Wort	n(Wort, HAM)	$\tilde{p}$(Wort|HAM)	n(Wort, SPAM)	$\tilde{p}$(Wort|SPAM)
meet				
deadline				
single				
...				

reminder
deadline
meet
thanks

thanks
for
tip

deadline
approaching

hot
stock
tip

meet
hot
single

Relative Häufigkeit für bedingte Wahrscheinlichkeiten

- Zufallsereignis x: Wortvorkommen ist ein bestimmtes Wort
- Zufallsereignis y: Wortvorkommen ist in Email einer bestimmten Kategorie, z.B. HAM oder SPAM (HAM=“kein Spam”)
- $n(x, y)$: Anzahl der Wortvorkommen in Emails einer Kategorie im Corpus

Wort	n(Wort, HAM)	$\tilde{p}(\text{Wort}\mid\text{HAM})$	n(Wort, SPAM)	$\tilde{p}(\text{Wort}\mid\text{SPAM})$
meet	1	$\frac{1}{9} \approx 0.111$	1	$\frac{1}{6} \approx 0.167$
deadline	2	$\frac{2}{9} \approx 0.222$	0	0
single	0	0	1	$\frac{1}{6} \approx 0.167$
...				

reminder
deadline
meet
thanks

thanks
for
tip

deadline
approaching

hot
stock
tip

meet
hot
single

Wahrscheinlichkeit für Wortsequenz

- Soweit haben wir nur Wahrscheinlichkeiten von Einzelwörtern ausgedrückt und diese geschätzt.
- Wie können wir die Wahrscheinlichkeiten von ganzen Texten (z.B. Emails) berechnen?
- Anwendung der bedingten Wahrscheinlichkeit:

$$P(w_1, w_2, \ldots, w_n)$$

$$= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\ldots P(w_n|w_1 \ldots w_{n-1})$$

- $\Rightarrow$ löst das Problem nicht wirklich, denn $P(w_n|w_1 \ldots w_{n-1})$ kann nicht gut geschätzt werden

Unabhängigkeitsannahme: Bag of Words

- Eine Lösung: Wir machen die statistische Annahme, dass jedes Wort unabhängig vom Vorkommen anderer Wörter ist.
- Dies nennt man auch Bag-of-words (BOW) Annahme, weil die Reihenfolge der Wörter irrelevant wird.

$$P(w_1, w_2, \ldots, w_n)$$

$$= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\ldots P(w_n|w_1 \ldots w_{n-1})$$

$$\underset{\text{Unabh.}}{=} P(w_1)P(w_2)P(w_3)\ldots P(w_n)$$

Bedingte Unabhängkeit

- Für viele Machine-Learning Algorithmen ist **bedingte Unabhängigkeit** das zentrale Konzept:
  Wenn der Wert einer Zufallsvariable y bekannt ist, sind Zufallsvariablen $x_1, \ldots, x_n$ unabhängig
- Mittelweg zwischen:
  - Keine Unabhängigkeit
  - Unabhängigkeit aller Zufallsvariablen
- In unserem Fall:

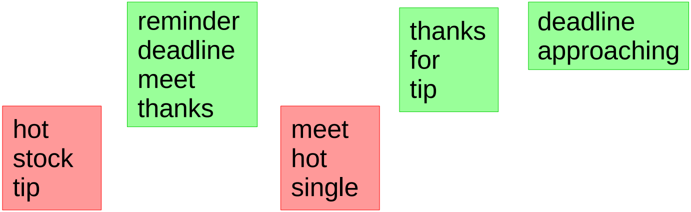
$$P(w_1, w_2, \ldots, w_n|\text{SPAM})$$

$$\underset{\text{bed. Unabh.}}{=} P(w_1|\text{SPAM})P(w_2|\text{SPAM})P(w_3|\text{SPAM})\ldots P(w_n|\text{SPAM})$$

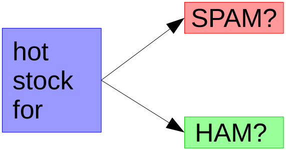
Naive Bayes Classifier

Aufgabenstellung

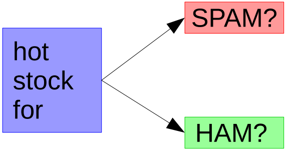
- Gegeben ein Traininscorpus:



- Entscheide ob neue (ungesehene) Emails der Kategorie HAM oder SPAM zugeordnet werden soll:



Entscheidungskriterium



- Gegeben der Inhalt der Email, welche Kategorie ist wahrscheinlicher, SPAM oder HAM?

$$P(HAM|text) > P(SPAM|text)$$

- Warum ist das Entscheidungskriterium nicht:

$$P(text|HAM) > P(text|SPAM)$$

?

Bayes-Regel

$$P(HAM|text) = \frac{P(text|HAM) * P(HAM)}{P(text)}$$

- $P(text|HAM)$: bedingte BOW-Wahrscheinlichkeit
- $P(HAM)$: Prior-Wahrscheinlichkeit, dass eine Email der Kategorie HAM zugeordnet wird (wenn der Inhalt der Email nicht bekannt ist). Schätzung:

$$\tilde{p}(HAM) = \frac{\text{Anzahl HAM-Mails}}{\text{Anzahl alle Mails}}$$

- $P(text)$: BOW-Wahrscheinlichkeit des Inhalts der Email, ohne dass die Kategorie bekannt ist

Entscheidungskriterium

Email ist HAM

$$\Leftrightarrow$$

$$P(HAM|text) > P(SPAM|text)$$

$$\Leftrightarrow$$

$$\frac{P(HAM|text)}{P(SPAM|text)} > 1$$

$$\Leftrightarrow$$

Entscheidungskriterium

Email ist HAM

$$\Leftrightarrow$$

$$P(HAM|text) > P(SPAM|text)$$

$$\Leftrightarrow$$

$$\frac{P(HAM|text)}{P(SPAM|text)} > 1$$

$$\Leftrightarrow$$

$$\frac{\cancel{\frac{1}{P(text)}} P(text|HAM) * P(HAM)}{\cancel{\frac{1}{P(text)}} P(text|SPAM) * P(SPAM)} > 1$$

Was ist Entscheidungsregel für mehr als zwei Kategorien?

Beispiel (Vorläufig)

hot stock tip

reminder deadline meet thanks

meet hot single

thanks for tip

deadline approaching

- $\tilde{p}(HAM) = \frac{3}{5}$
- $\tilde{p}(SPAM) = \frac{2}{5}$
- $p(\text{hot stock for}|HAM)$

$$= \tilde{p}(\text{hot}|HAM)\tilde{p}(\text{stock}|HAM)\tilde{p}(\text{for}|HAM) = \ldots$$

- $p(\text{hot stock for}|SPAM)$

$$= \tilde{p}(\text{hot}|SPAM)\tilde{p}(\text{stock}|SPAM)\tilde{p}(\text{for}|SPAM) = \ldots$$

- ...

Beispiel (Vorläufig)

hot stock tip	reminder deadline meet thanks	meet hot single	thanks for tip	deadline approaching

- $\tilde{p}(HAM) = \frac{3}{5}$
- $\tilde{p}(SPAM) = \frac{2}{5}$
- $p(\text{hot stock for}|HAM)$

$$= \tilde{p}(\text{hot}|HAM)\tilde{p}(\text{stock}|HAM)\tilde{p}(\text{for}|HAM) = \frac{0 \cdot 0 \cdot 1}{9 \cdot 9 \cdot 9} = 0$$

- $p(\text{hot stock for}|SPAM)$

$$= \tilde{p}(\text{hot}|SPAM)p(\text{stock}|SPAM)\tilde{p}(\text{for}|SPAM) = \frac{2 \cdot 1 \cdot 0}{6 \cdot 6 \cdot 6} = 0$$

- Problem: Entscheidungskriterium ist nicht definiert $(\frac{0}{0})$

Addiere-1 Glättung

Addiere-1 Glättung (Laplace-Glättung)

$$\tilde{p}(w) = \frac{n(w) + 1}{N + V}$$

(V = Anzahl der möglichen Wörter; N = Zahl der Tokens)

... ist optimal falls die uniforme Verteilung am wahrscheinlichsten ist, was in bei Textcorpora selten der Fall ist $\Rightarrow$ Zipf'sche Verteilung

... überschätzt daher die Wahrscheinlichkeit ungesehener Wörter.

Addiere-λ Glättung

reduziert das Ausmaß der Glättung

Addiere-λ Glättung

$$\tilde{p}(w) = \frac{n(w) + \lambda}{N + V\lambda}$$

Addiere-λ Glättung für bedingte Wahrscheinlichkeiten

$$\tilde{p}(w|y) = \frac{n(w, y) + \lambda}{n_y + v\lambda}$$

Beispiel (mit Addiere-1 Glättung)

hot stock tip

reminder deadline meet thanks

meet hot single

thanks for tip

deadline approaching

- $\tilde{p}(HAM) = \frac{3}{5}$, $\tilde{p}(SPAM) = \frac{2}{5}$
- Vokabular enthält $v = 10$ unterschiedliche Wörter
- $p(\text{hot stock for}|HAM) = \tilde{p}(\text{hot}|HAM)\tilde{p}(\text{stock}|HAM)\tilde{p}(\text{for}|HAM)$

$$= \frac{(0+1)\cdot(0+1)\cdot(1+1)}{(9+10)\cdot(9+10)\cdot(9+10)} \approx 0.00029$$

- $p(\text{hot stock for}|SPAM) = \tilde{p}(\text{hot}|SPAM)\tilde{p}(\text{stock}|SPAM)\tilde{p}(\text{for}|SPAM)$

$$= \frac{(2+1)\cdot(1+1)\cdot(0+1)}{(6+10)\cdot(6+10)\cdot(6+10)} \approx 0.00146$$

- $\frac{P(text|HAM)*P(HAM)}{P(text|SPAM)*P(SPAM)} = \frac{0.00029\cdot 0.6}{0.00146\cdot 0.4} \approx 0.298 \Rightarrow$ Kategorie?

Beispiel (mit Addiere-1 Glättung)

hot stock tip

reminder deadline meet thanks

meet hot single

thanks for tip

deadline approaching

- $\tilde{p}(HAM) = \frac{3}{5}$, $\tilde{p}(SPAM) = \frac{2}{5}$
- Vokabular enthält $v = 10$ unterschiedliche Wörter
- $p(\text{hot stock for}|HAM) = \tilde{p}(\text{hot}|HAM)\tilde{p}(\text{stock}|HAM)\tilde{p}(\text{for}|HAM)$

$$= \frac{(0+1)\cdot(0+1)\cdot(1+1)}{(9+10)\cdot(9+10)\cdot(9+10)} \approx 0.00029$$

- $p(\text{hot stock for}|SPAM) = \tilde{p}(\text{hot}|SPAM)\tilde{p}(\text{stock}|SPAM)\tilde{p}(\text{for}|SPAM)$

$$= \frac{(2+1)\cdot(1+1)\cdot(0+1)}{(6+10)\cdot(6+10)\cdot(6+10)} \approx 0.00146$$

- $\frac{P(text|HAM)*P(HAM)}{P(text|SPAM)*P(SPAM)} = \frac{0.00029\cdot 0.6}{0.00146\cdot 0.4} \approx 0.298 < 1 \Rightarrow$ Email ist Spam

Rechnen mit Logarithmen

- Bei der Multiplikation vieler kleiner Wahrscheinlichkeiten (z.B. aller Worte in einem langen Text) kann sich das Ergebnis schnell dem Wert 0 annähern, und u.U. nicht mehr korrekt repräsentiert werden.
- Deswegen vermeidet man möglichst immer die Multiplikation von Wahrscheinlichkeiten.
- Man verwendet stattdessen die Summe der logarithmierten Wahrscheinlichkeiten.
- $\log(a \cdot b \cdot c \cdot \ldots) = \log(a) + \log(b) + \log(c) + \ldots$
- Beispiel:

$$0.0001 * 0.001 * 0.00001 * 0.01 = 0.00000000000001$$

$$\log_{10}(0.0001 * 0.001 * 0.00001 * 0.01) =$$

Rechnen mit Logarithmen

- Bei der Multiplikation vieler kleiner Wahrscheinlichkeiten (z.B. aller Worte in einem langen Text) kann sich das Ergebnis schnell dem Wert 0 annähern, und u.U. nicht mehr korrekt repräsentiert werden.
- Deswegen vermeidet man möglichst immer die Multiplikation von Wahrscheinlichkeiten.
- Man verwendet stattdessen die Summe der logarithmierten Wahrscheinlichkeiten.
- $\log(a \cdot b \cdot c \cdot \ldots) = \log(a) + \log(b) + \log(c) + \ldots$
- Beispiel:

$$0.0001 * 0.001 * 0.00001 * 0.01 = 0.00000000000001$$

$$\log_{10}(0.0001 * 0.001 * 0.00001 * 0.01) = -4 + (-3) + (-5) + (-2) = -14$$

- $\log(\frac{a}{b}) = ?$

Entscheidungsregel mit Logarithmen

- Der Logarithmus ist monoton steigend, d.h. wir können ihn bei Ungleichungen auf beiden Seiten anwenden.
- Die Entscheidungsregel ist nun:

$$P(HAM|text) > P(SPAM|text)$$

$$\Leftrightarrow$$

$$\log P(HAM|text) > \log P(SPAM|text)$$

$$\Leftrightarrow$$

Entscheidungsregel mit Logarithmen

- Der Logarithmus ist monoton steigend, d.h. wir können ihn bei Ungleichungen auf beiden Seiten anwenden.
- Die Entscheidungsregel ist nun:

$$P(HAM|text) > P(SPAM|text)$$

$$\Leftrightarrow$$

$$\log P(HAM|text) > \log P(SPAM|text)$$

$$\Leftrightarrow$$

$$\log P(HAM|text) - \log P(SPAM|text) > 0$$

$$\Leftrightarrow$$

$$\log P(text|HAM) + \log P(HAM) - \log P(text|SPAM) - \log P(SPAM) > 0$$

- Den Quotienten der Wahrscheinlichkeiten zweier komplementärer Ereignisse nennt man auch **Odds**.
- Den Logarithmus dieses Quotienten nennt man **Log-Odds**.

Naive Bayes mit anderen Verteilungen

- Je nach Problem kann die Verteilung $P(X|Kategorie)$ verschiedene Formen annehmen.
- Im gerade besprochenen Fall ist die Verteilung die **Multinomialverteilung** (Wahrscheinlichkeit, dass bei Text der Länge n genau die beobachteten Anzahlen der Wörter vorkommen)
- Sind die beobachteten Werte Fließkommawerte (z.B. Werte aus einem Messgerät), kann man z.B. **Gaussverteilungen** verwenden.
  $\Rightarrow$ Smoothing ist auch hier wichtig (z.B. welche Varianz soll bei wenig Daten für eine Kategorie angenommen werden?)
- Bei Machine-learning-Software (z.B. Scikit-learn) kann man die Art der Verteilung als Hyper-Parameter auswählen.

Unbekannte Wörter in den Test-Daten

- Es kann sein, dass Wörter in den Testdaten vorkommen, die in den Trainingsdaten nicht vorgekommen sind.
- Die möglichen Werte der Zufallsvariable wurden aber Anhand der Trainingsdaten gewählt, d.h. die Wahrscheinlichkeit der neuen Wörter ist nicht definiert.
- Zwei häufig verwendete Lösungen:
  - Wörter, die nicht in den Trainingsdaten vorkommen werden ignoriert ($\Rightarrow$ Testdokumente werden kürzer)
  - Wörter, die in den Trainigsdaten nur selten (z.B. 1-2-Mal) bzw. nicht vorkommen, werden (in Training und Test) durch einen Platzhalter `<UNK>` ersetzt.
- Vor- und Nachteile der beiden Methoden?

Implementierung

Trainings- oder Test-Instanz

In unserem Fall:

- Features = Wörter (Tokens)
- Label
  - Binäre Klassifikation: HAM (True) vs SPAM (False)
  - Multi-Klassen Klassifikation (Übungsblatt): String für Kategorie ("work", "social", "promotions", "spam", ...)

```
class DataInstance:
    def __init__(self, feature_counts, label):
        self.feature_counts = feature_counts
        self.label = label
#...
```

Trainings- oder Test-Set

- Menge der möglichen Merkmalsausprägungen ist z.B. für Glättung wichtig.
- Sanity-check: Welche Genauigkeit hätte Vorhersage der Häufigsten Kategorie?
- Manche Lernalgorithmen verlangen mehrere Trainings-Iterationen, zwischen denen das Trainingsset neu permutiert (gemischt) werden sollte.

```
class Dataset:
    def __init__(self, instance_list, feature_set):
        self.instance_list = instance_list
        self.feature_set = feature_set
    def most_frequent_sense_accuracy(self):
        # ...
    def shuffle(self):
        # ...
```

Klassifikator

Welche Informationen benötigen wir, um das Naive-Bayes Modell zu erstellen?

- ...

Klassifikator

Welche Informationen benötigen wir, um das Naive-Bayes Modell zu erstellen?

- Für die Schätzung von $P(w|HAM)$ bzw. $P(w|SPAM)$
  - $n(w, HAM)$ bzw. $n(w, SPAM)$:
    Je ein Dictionary, welches jedes Wort auf seine Häufigkeit in der jeweiligen Kategorie abbildet.
  - n_{HAM} bzw. n_{SPAM}:
    Die Anzahl aller Wortvorkommen pro Kategorie
    (kann aus den Values der Dictionaries aufsummiert werden)
  - Für die Glättung: Parameter λ und Größe des Vokabulars v
- Für die Schätzung von $P(HAM)$ bzw. $P(SPAM)$
  - Jeweils die Anzahl der Trainingsemails pro Kategorie.

Klassifikator: Konstruktor

```
def __init__(self, positive_word_to_count, negative_word_to_count,\
        positive_counts, negative_counts, vocabsize, smoothing):
    # n(word, HAM) and n(word, SPAM)
    self.positive_word_to_count = positive_word_to_count
    self.negative_word_to_count = negative_word_to_count

    # n_HAM and n_SPAM
    self.positive_total_wordcount = \
        sum(positive_word_to_count.values())
    self.negative_total_wordcount = \
        sum(negative_word_to_count.values())

    self.vocabsize = vocabsize

    # P(HAM) and P(SPAM)
    self.positive_prior = \
        positive_counts / (positive_counts + negative_counts)
    self.negative_prior = \
        negative_counts / (positive_counts + negative_counts)

    self.smoothing = smoothing
```

Klassifikator: Übersicht

```
class NaiveBayesWithLaplaceClassifier:
    def log_probability(self, word, is_positive_label):
        # ...
    def log_odds(self, feature_counts):
        # ...
    def prediction(self, feature_counts):
        # ...
    def prediction_accuracy(self, dataset):
        # ...
    def log_odds_for_word(self, word):
        # ...
    def features_for_class(self, is_positive_class, topn=10):
        # ...
```

Berechnung von $P(w|HAM)$ bzw $P(w|SPAM)$

Wahrscheinlichkeitsschätzung ...

- ... geglättet
- ... wird logarithmiert zurückgegeben

```
def log_probability(self, word, is_positive_label):
    if is_positive_label:
        wordcount = self.positive_word_to_count.get(word, 0)
        total = self.positive_total_wordcount
    else:
        wordcount = self.negative_word_to_count.get(word, 0)
        total = self.negative_total_wordcount
    return math.log(wordcount + self.smoothing) \
        - math.log(total + self.smoothing * self.vocabsize)
```

Berechnung der Log-Odds

- Was wird in den zwei Summen jeweils berechnet?

```
def log_odds(self, feature_counts):
    # language model probability
    pos_logprob = sum([ count * self.log_probability(word, True) \
        for word, count in feature_counts.items()])
    # prior probability
    pos_logprob += math.log(self.positive_prior)
    # same for negative case
    neg_logprob = sum([ count * self.log_probability(word, False) \
        for word, count in feature_counts.items()])
    neg_logprob += math.log(self.negative_prior)
    return pos_logprob - neg_logprob
```

Anwenden des Klassifikators, Test-Accuracy

- Vorhersage
  - Anwenden des Modells auf die Feature-Counts einer Test-Instanz
  - Vorhersage einer Kategorie (HAM/True oder SPAM/False) gemäß der Entscheidungsregel

```
def prediction(self, feature_counts):
    # ...
```

- Berechnung der Test-Accuracy
  - Zunächst Vorhersage für alle Instanzen des Dataset
  - Dann Vergleich mit dem richtigen Kategorien-Label

```
def prediction_accuracy(self, dataset):
    # ...
```

Multi-Klassen Klassifikation

Multi-Klassen Klassifikation

- Erweiterung: Klassifikator unterscheidet n verschiedene Kategorien ($n \geq 2$)
- $\Rightarrow$ Übungsblatt
- Entscheidungsregel: wähle Kategorie c^*, die die Wahrscheinlichkeit $p(c^*|text)$ maximiert.

$$c^* = \arg\max_c p(c|text)$$

- $\arg\max_x f(x)$ wählt einen Wert x (aus der Definitionsmenge) aus, für den der Funktionswert $f(x)$ maximal ist.
- Durch Anwendung der Rechenregeln, die bedingte Unabhängigkeitsannahme, und unsere Schätzmethode (Laplace) gilt:

$$c^* = \arg\max_c p(c)p(text|c)$$

$$= \arg\max_c \log[p(c)] + \sum_{w \in text} \log[\tilde{p}(w|c)]$$

Multi-Klassen Klassifikation

- Entscheidungsregel: wähle Kategorie c^*, die die Wahrscheinlichkeit $p(c^*|text)$ maximiert.

$$c^* = \arg\max_c p(c|text)$$

- Gilt die folgende Implikation?

$$c^* = \arg\max_c p(c|text) \Rightarrow \frac{p(c^*|text)}{1 - p(c^*|text)} \geq 1$$

- Gilt die folgende Implikation?

$$\frac{p(c^*|text)}{1 - p(c^*|text)} > 1 \Rightarrow c^* = \arg\max_c p(c|text)$$

Multi-Klassen Klassifikation

- Gilt die folgende Implikation?

$$c^* = \arg\max_c p(c|text) \Rightarrow \frac{p(c^*|text)}{1 - p(c^*|text)} \geq 1$$

Nein. Bei 3 oder mehr Kategorien kann es sein, dass die wahrscheinlichste Kategorie eine WK $p(c^*|text) < 0.5$ hat, und die Odds < 1 sind.

- Gilt die folgende Implikation?

$$\frac{p(c^*|text)}{1 - p(c^*|text)} > 1 \Rightarrow c^* = \arg\max_c p(c|text)$$

Ja. Wenn die wahrscheinlichste Kategorie Odds > 1 hat, ist die WK $p(c^*|text) > 0.5$, und alle anderen Kategorien müssen eine kleinere WK haben.

Multi-Klassen Naive Bayes: Implementierung

- Um die Werte $\tilde{p}(w|c)$ zu berechnen, benötigen wir die Worthäufigkeiten pro Klasse $n(w, c)$
  Lösung: Dictionary
  (str,str) $\rightarrow$ int
- Für die priors $p(c)$ brauchen wir die Anzahl der Instanzen pro Klasse: str $\rightarrow$ int
- Außerdem noch die Vokabulargröße und den Glättungsparameter

```
class NaiveBayesClassifier:
    def __init__(self, word_and_category_to_count, \
        category_to_num_instances, vocabsize, smoothing):
        # ...
```

Log-Odds pro Wort

$\Rightarrow$ Übungsblatt

- Die Log-Odds für eine Kategorie c können auch nur für ein Wort (anstelle eines ganzen Dokuments) berechnet werden.
- Beginne mit $\log \frac{p(c|w)}{1-p(c|w)}$ und wende die Rechenregeln an

$$\log \frac{p(c|w)}{1 - p(c|w)} = ...$$

$$= \log[\tilde{p}(w|c)] + \log[p(c)] - \log[\sum_{c' \neq c} \tilde{p}(w|c')p(c')]$$

- Die Log-Odds pro Wort zeigen an, wie stark ein Wort auf die jeweilige Kategorie hinweist
- Man kann dann alle Wörter anhand ihrer Log-Odds sortieren, und einen Eindruck bekommen, was das Modell gelernt hat (d.h. was für das Modell wichtig ist)

Trainieren und Evaluieren eines Klassifikators

Um einen Klasifikator trainieren und evaluieren zu können, brauchen wir 3 Datensets:

1. Trainingsdaten: Auf diesen Daten schätzt das Modell seine Parameter automatisch. (Z.B. Wortwahrscheinlichkeiten und Kategorien-Priors)
2. Entwicklungsdaten: Auf diesen Daten können verschiedene Model-Architekturen und Hyper-Parameter[1] verglichen werden. **Was z.B. in unserem Fall?**
3. Testdaten: Auf diesen Daten kann, **nachdem durch die Entwicklungsdaten eine Modelarchitektur endgültig bestimmt wurde**, ein Schätzwert gewonnen werden, wie gut das Modell auf weiteren ungesehenen Daten funktioniert.
    - Je nach Beschaffenheit der Daten (Domäne etc) kann diese Schätzung sehr stark von der Realität abweichen.
    - Der Schätzwert kann außerdem in Abhängigkeit der Menge der Test-Daten sehr unzuverlässig sein (⇒ Signifikanztests).
    - **Warum können wir nicht die Performanz auf den Entwicklungsdaten verwenden?**

[1] Parameter, die nicht automatisch gelernt werden

Zusammenfassung

- Wahrscheinlichkeitsrechung
  - Satz von Bayes
  - Bedingte Unabhängigkeit
- Naive Bayes Klassifikator
  - Entscheidungsregel, und “umdrehen” der Formel durch Satz von Bayes
  - Glättung der Wahrscheinlichkeiten
  - Log-Odds
- Fragen?