Data Mining & Pattern Recognition

Versuch 4:
Dokument Klassifikation / Spam Filter

Datum:	
Namen:	
Unterschrift Betreuer:	

Prof. Dr. Johannes Maucher
Medieninformatik Bachelor (MIB)
WS 12/13

Inhaltsverzeichnis

1 Einführung

1.1 Lernziele

In diesem Versuch sollen Kenntnisse in folgenden Themen vermittelt werden:

- **Dokumentklassifikation:** Klassifikation von Dokumenten durch Textanalyse
- **Naive Bayes Classifier:** Weit verbreitete Klassifikationsmethode, welche unter bestimmten Randbedingungen sehr gut skaliert.

Sämtliche Verfahren und Algorithmen werden in Python implementiert.

1.2 Protokoll

Abzugeben sind pro Gruppe spätestens vor Beginn des nächsten Versuchs:

- Antworten auf alle in dieser Anleitung gestellten Fragen. Die Antworten sollten in einem elektronischen Dokument verfasst und als .pdf bereitgestellt werden. Die Antworten sind entsprechend der Kapitelnummerierung in dieser Anleitung zu ordnen.
- Die Antworten auf die Fragen im Kapitel 1.4 müssen **nicht** protokolliert und abgegeben werden. Diese Fragen müssen im Rahmen der Versuchsvorbereitung geklärt werden und werden im Kolloq zu Beginn des Versuchs abgefragt.
- Alle Dateien von denen in dieser Anleitung ein Abspeichern gefordert wird.

Die Durchführung und Dokumentation eigener Experimente, die über die Aufgabenstellungen in diesem Dokument hinausgehen, wird ausdrücklich unterstützt und in der Bewertung entsprechend berücksichtigt.

1.3 Theorie zur Vorbereitung

1.3.1 Parametrische Klassifikation und Naive Bayes Methode

Klassifikatoren müssen zu einer gegebenen Eingabe $\underline{x}$ die zugehörige Klasse C_i bestimmen. Mithilfe der Wahrscheinlichkeitstheorie kann diese Aufgabe wie folgt beschrieben werden: Bestimme für alle möglichen Klassen C_i die bedingte Wahrscheinlichkeit $P(C_i|\underline{x})$, also die Wahrscheinlichkeit, dass die gegebene Eingabe $\underline{x}$ in Klasse C_i fällt. Wähle dann die Klasse aus, für welche diese Wahrscheinlichkeit maximal ist.

Die Entscheidungsregeln von Klassifizierern können mit Methoden des **überwachten Lernens** aus Trainingsdaten ermittelt werden. Im Fall des **parametrischen Lernens** kann aus den Trainingsdaten die sogenannte **Likelihood-Funktion** $p(\underline{x} \mid C_i)$ bestimmt werden[1]. Mithilfe der **Bayes-Formel**

$$P(C_i \mid \underline{x}) = \frac{p(\underline{x} \mid C_i) \cdot P(C_i)}{p(\underline{x})}. \tag{1}$$

kann aus der Likelihood die **a-posteriori-Wahrscheinlichkeit** $P(C_i \mid \underline{x})$ berechnet werden. Darin wird $P(C_i)$ die **a-priori-Wahrscheinlichkeit** und $p(\underline{x})$ die **Evidenz** genannt. Die a-priori-Wahrscheinlichkeit kann ebenfalls aus den Trainingsdaten ermittelt werden. Die Evidenz ist für die Klassifikationsentscheidung nicht relevant, da sie für alle Klassen C_i gleich groß ist.

[1] Allgemein werden mit $p(...)$ kontinuierliche Wahrscheinlichkeitsfunktionen und mit $P(...)$ diskrete Wahrscheinlichkeitswerte bezeichnet

Die Berechnung der Likelihood-Funktion $p(\underline{x} \mid C_i)$ ist dann sehr aufwendig, wenn $\underline{x} = (x_1, x_2, \ldots, x_Z)$ ein Vektor von voneinander abhängigen Variablen x_i ist. Bei der **Naive Bayes Classification** wird jedoch von der vereinfachenden Annahme ausgegangen, dass die Eingabevariabeln x_i voneinander unabhängig sind. Dann vereinfacht sich die bedingte Verbundwahrscheinlichkeits-Funktion $p((x_1, x_2, \ldots, x_Z) \mid C_i)$ zu:

$$p((x_1, x_2, \ldots, x_Z) \mid C_i) = \prod_{i=1}^{Z} p(x_i|C_i) \tag{2}$$

1.3.2 Anwendung der Naive Bayes Methode in der Dokumentklassifikation

Auf der rechten Seite der Gleichung 2 stehen nur noch von den jeweils anderen Variablen unabhängige bedingte Wahrscheinlichkeiten. Im Fall der Dokumentklassifikation sind die einzelnen Worte die Variablen, d.h. ein Ausdruck der Form $P(x_i|C_j)$ gibt an mit welcher Wahrscheinlichkeit ein Wort $x_i = w$ in einem Dokument der Klasse C_j vorkommt. Die Menge aller Variablen $\{x_1, x_2, \ldots, x_Z\}$ ist dann die Menge aller Wörter im Dokument. Damit ist die Bedeutung der linken Seite in Gleichung 2: *Wahrscheinlichkeit, dass die Wörter $\{x_1, x_2, \ldots, x_Z\}$ in einem Dokument der Klasse C_i vorkommen.*

Für jedes Wort w wird aus den Trainingsdaten die Wahrscheinlichkeit $P(w|G)$, mit der das Wort in Dokumenten der Kategorie *Good* und die Wahrscheinlichkeit $P(w|B)$ mit der das Wort in Dokumenten der Kategorie *Bad* auftaucht ermittelt. Trainingsdokumente werden in der Form

$$tD = (String, Category)$$

eingegeben. Wenn

- mit der Variable $fc(w, cat)$ die Anzahl der Trainingsdokumente in Kategorie cat in denen das Wort w enthalten ist
- mit der Variable $cc(cat)$ die Anzahl der Trainingsdokumente in Kategorie cat

gezählt wird, dann ist

$$P(w|G) = \frac{fc(w, G)}{cc(G)} \tag{3}$$

und

$$P(w|B) = \frac{fc(w, B)}{cc(B)}. \tag{4}$$

Wird nun nach der Eingabe von L Trainingsdokumenten ein neu zu klassifizierendes Dokument D eingegeben und sei $W(D)$ die Menge aller Wörter in D, dann berechnen sich unter der Annahme dass die Worte in $W(D)$ voneinander unabhängig sind (naive Bayes Annahme) die a-posteriori Wahrscheinlichkeiten unter Anwendung der Formeln 1 und 2 zu

$$P(G|D) = \frac{\left(\prod_{w \in W(D)} P(w|G)\right) \cdot P(G)}{p(D)} \tag{5}$$

und

$$P(B|D) = \frac{\left(\prod_{w \in W(D)} P(w|B)\right) \cdot P(B)}{p(D)}. \tag{6}$$

Die hierfür notwendigen a-priori-Wahrscheinlichkeiten berechnen sich zu

$$P(G) = \frac{cc(G)}{L}$$

und

$$P(B) = \frac{cc(B)}{L}$$

Die Evidenz $p(D)$ beeinflusst die Entscheidung nicht und kann deshalb ignoriert werden.

1.4 Vor dem Versuch zu klärende Fragen

- Wie wird ein Naiver Bayes Classifier trainiert?
- Wie teilt ein Naiver Bayes Classifier ein neues Dokument ein?
- Welche naive Annahme liegt dem Bayes Classifier zugrunde?
- Ist diese Annahme im Fall der Dokumentklassifikation tatsächlich gegeben?
- Betrachten Sie die Formeln 5 und 5. Welches Problem stellt sich ein, wenn in der Menge $W(D)$ ein Wort vorkommt, das nicht in den Trainingsdaten der Kategorie G vorkommt und ein anderes Wort aus $W(D)$ nicht in den Trainingsdaten der Kategorie B enthalten ist? Wie könnte dieses Problem gelöst werden?

2 Durchführung

2.1 Implementierung

Die folgenden Funktionen und Klassen sind in einer Datei `docclass.py` zu implementieren. Achten Sie auf ausreichende Kommentierung des Quellcodes.

2.1.1 Funktion getwords()

Implementieren Sie eine Funktion `getwords(doc)` der ein beliebiges Dokument in Form einer String-Variablen übergeben wird. In der Funktion soll der String in seine Wörter zerlegt und jedes Wort in *lowercase* transformiert werden. Wörter, die weniger als eine untere Grenze von Zeichen (z.B. 3) oder mehr als eine obere Grenze von Zeichen (z.B. 20) enthalten, sollen ignoriert werden. Die Funktion soll ein dictionary zurückgeben, dessen *Keys* die Wörter sind. Die *Values* sollen für jedes Wort zunächst auf 1 gesetzt werden.
`Tipp:` Benutzen Sie für die Zerlegung des Strings und für die Darstellung aller Wörter mit ausschließlich kleinen Buchstaben die Funktionen `split()` und `lower()` der Klasse String.

2.1.2 Die Klasse `Classifier`

Es bietet sich an die Funktionalität des Klassifikators und das vom Klassifikator gelernte Wissen in einer Instanz einer Klasse `Classifier` zu kapseln. In diesem Fall kann wie folgt vorgegangen werden:

- Im Konstruktor der Klasse wird je ein Dictionary für die Instanzvariablen `fc` und `cc` (siehe oben) initialisiert. Dabei ist `fc` ein verschachteltes Dictionary. Seine Keys sind die bisher gelernten Worte, die Values sind wiederum Dictionarys, deren Keys die Kategorien *Good* und *Bad* sind und deren Values zählen wie häufig das Wort bisher in Dokumenten der jeweiligen Kategorie auftrat. Das

Dictionary `cc` hat als Keys die Kategorien *Good* und *Bad*. Die Values zählen wie häufig Dokumente der jeweiligen Kategorien bisher auftraten.

- Im Konstruktor wird ferner der Instanzvariablen `getfeatures` die Funktion `getwords()` übergeben. Die Funktion `getwords()` wurde bereits zuvor ausserhalb der Klasse definiert. Sinn dieses Vorgehens ist, dass andere Varianten um Merkmale aus Dokumenten zu extrahieren denkbar sind. Diese Varianten könnten dann ähnlich wie die `getwords()`-Funktion ausserhalb der Klasse definiert und beim Anlegen eines `Classifier`-Objekts der Instanzvariablen `getfeatures` übergeben werden.
- Der Methode `incf(self,f,cat)` wird ein Wort `f` und die zugehörige Kategorie `cat` des Dokuments in welchem es auftrat übergeben. In der Methode wird der `fc`-Zähler angepasst.
- Der Methode `incc(self,cat)` wird die Kategorie `cat` des gerade eingelesenen Dokuments übergeben. In der Methode wird der `cc`-Zähler angepasst.
- Die Methode `fcount(self,f,cat)` gibt die Häufigkeit des Worts `f` in den Dokumenten der Kategorie `cat` zurück.
- Die Methode `catcount(self,cat)` gibt die Anzahl der Dokumente in der Kategorie `cat` zurück.
- Die Methode `totalcount(self)` gibt die Anzahl aller Dokumente zurück.
- Der Methode `train(self,item,cat)` wird ein neues Trainingselement, bestehend aus der Betreffzeile (`item`) und der entsprechenden Kategorisierung (`cat`) übergeben. Der String `item` wird mit der Instanzmethode `getfeatures` (Diese referenziert `getwords()`) in Worte zerlegt. Für jedes einzelne Wort wird dann `incf(self,f,cat)` aufgerufen. Ausserdem wird für das neue Trainingsdokument die Methode `incc(self,cat)` aufgerufen.
- Die Methode `fprob(self,f,cat)` berechnet die bedingte Wahrscheinlichkeit $P(f|cat)$ des Wortes `f` in der Kategorie `cat` entsprechend den Formeln (3) und (4), indem sie den aktuellen Stand des Zählers `fc(f,cat)` durch den aktuellen Stand des Zählers `cc(cat)` teilt.
- Die Methode `fprob(self,f,cat)` liefert evtl. ungewollt extreme Ergebnisse, wenn noch wenig Wörter im Klassifizierer verbucht sind. Kommt z.B. ein Wort erst einmal in den Trainingsdaten vor, so wird seine Auftrittswahrscheinlichkeit in der Kategorie in welcher es nicht vorkommt gleich 0 sein. Um extreme Wahrscheinlichkeitswerte im Fall noch selten vorkommender Werte zu vermeiden soll zusätzlich zur Methode `fprob(self,f,cat)` die Methode `weightedprob(self,f,cat)` implementiert und angewandt werden. Der von ihr zurückgegebene Wahrscheinlichkeitswert könnte z.B. wie folgt berechnet werden:

$$wprob = \frac{initprob + count \cdot fprob(self, f, cat)}{1 + count},$$

  wobei *initprob* ein initialer Wahrscheinlichkeitswert (z.B. 0.5) ist, welcher zurückgegeben werden soll, wenn das Wort noch nicht in den Trainingsdaten aufgetaucht ist. Die Variable *count* zählt wie oft das Wort f bisher in den Trainingsdaten auftrat. Wie zu erkennen ist, nimmt der Einfluss der initialen Wahrscheinlichkeit ab, je häufiger das Wort in den Trainingsdaten auftrat.
- Nach dem Training soll ein beliebiges neues Dokument (Text-String) eingegeben werden können. Für dieses soll mit der Methode `prob(self,item,cat)` die a-posteriori-Wahrscheinlichkeit $P(cat|item)$ [2], mit der das Dokument `item` in die Kategorie `cat` fällt berechnet werden. Innerhalb der Methode

[2] Aufgrund der Vernachlässigung der Evidenz handelt es sich hierbei genaugenommen um das Produkt aus a-posteriori-Wahrscheinlichkeit und Evidenz

`prob(self,item,cat)` soll zunächst die Methode `weightedprob(self,f,cat)` für alle Wörter f im Dokument `item` aufgerufen werden. Die jeweiligen Rückgabewerte von `weightedprob(self,f,cat)` werden multipliziert. Das Produkt der Rückgabewerte von `weightedprob(self,f,cat)` über alle Wörter f im Dokument muss schließlich noch mit der a-priori Wahrscheinlichkeit $P(G)$ bzw. $P(B)$ entsprechend den Formel (5) bzw. (6) multipliziert werden. Das Resultat des Produkts wird an das aufrufende Programm zurück gegeben, die Evidenz wird also vernachlässigt (wie oben begründet).

Ein Dokument `item` wird schließlich der Kategorie `cat` zugeteilt, für welche die Funktion `prob(self,item,cat)` den höheren Wert zurück gibt. Da die Rückgabewerte in der Regel sehr klein sind, werden in der Regel folgende Werte angezeigt. Wenn mit g der Rückgabewert von `prob(self,item,cat=G)` und mit b der Rückgabewert von `prob(self,item,cat=B)` bezeichnet wird dann ist die Wahrscheinlichkeit, dass *item* in die Kategorie G fällt, gleich:

$$\frac{g}{g+b}$$

und die Wahrscheinlichkeit, dass *item* in die Kategorie B fällt, gleich:

$$\frac{b}{g+b}$$

2.2 Test

Instanzieren Sie ein Objekt der Klasse `Classifier` und übergeben Sie der `train()` Methode dieser Klasse mindestens 10 kategorisierte Dokumente (Betreffzeilen als Stringvariablen zusammen mit der Kategorie Good oder Bad). Definieren Sie dann ein beliebig neues Dokument und berechnen Sie für dieses die Kategorie, in welches es mit größter Wahrscheinlichkeit fällt. Für den Test bietet es sich an das in `https://www.mi.hdm-stuttgart.de/mib/studium/intern/skripteserver/skripte/Einfuehrung_Kuenstliche_I` ausführlich beschriebene Beispiel zu implementieren.

2.3 Klassifikation von RSS Newsfeeds

Mit dem auf dem Skripteserver bereitgestellten File `parseTechFeed.py` werden Nachrichten verschiedener Newsserver geladen und als String abgespeichert. Trainieren Sie Ihren Naive Bayes Classifier mit allen Nachrichten der in den Listen `trainTech` und `trainNonTech` definierten Servern. Weisen Sie für das Training allen Nachrichten aus `trainTech` die Kategorie `Tech` und allen Nachrichten aus `trainNonTech` die Kategorie `NonTech` zu.

Nach dem Training sollen alle Nachrichten aus der Liste `test` vom Naive Bayes Classifier automatisch klassifiziert werden. Beurteilen und diskutieren Sie die Rate korrekter Zuweisungen.

3 Fragen zum Versuch

1. Was wird mit Evidenz bezeichnet und warum muss diese für die Klassifikation nicht berücksichtigt werden?
2. Wann würden Sie in der Formel für die gewichtete Wahrscheinlichkeit den Wert von *initprob* kleiner, wann größer als 0.5 wählen? (Falls Sie die Möglichkeit haben diesen Wert für jedes Feature und jede Kategorie individuell zu konfigurieren)
3. Was könnten Sie mit dem in dieser Übung implementierten Classifier noch klassifizieren? Geben Sie eine für Sie interessante Anwendung an.

4. Das einmal trainierte, sollte eigentlich persistent abgespeichert werden. Beschreiben Sie kurz wie Sie das für dieses Programm machen würden.

Literatur

[PY] Python Documentation; `http://docs.python.org/index.html/`

[EA] E. Alpaydin; *Maschinelles Lernen*; Deutsche Ausgabe im Oldenbourg Verlag; 2008

[TS] Toby Segaran; *Kollektive Intelligenz*; O'Reilly Verlag; 2008

[MKI] Johannes Maucher; Vorlesung Einführung in die Künstliche Intelligenz - Maschinelles Lernen Teil 1 und 2; `http://www.hdm-stuttgart.de/~maucher/Lecture_KI.html`