1.4 Vor dem Versuch zu klärende Fragen

Wie wird ein Naiver Bayes Classifier trainiert?

Im Fall des überwachten Lernens werden die a-priori Wahrscheinlichkeit und die Likelihood aus den Trainingsdaten geschätzt.

Die a-priori Wahrscheinlichkeit bekommt man durch zählen, wie oft eine Klasse C_i in den Trainingsdaten vorkommt.

$$P(C_i) = |C_i| / N$$

Die bedingte Wahrscheinlichkeit (Likelihood) erhält man durch zählen, wie oft ein Wort in den jeweiligen Klassen vorkommt. Dann teilt man diese Häufigkeit durch die a-priori Wahrscheinlichkeit:

$$P(x_j|C_i) = |(x_j,C_i)| / |C_i|$$

Wie teilt ein Naiver Bayes Classifier ein neues Dokument ein?

Er berechnet zunächst die A-Priori Wahrscheinlichkeit für alle Klassen im Dokument. Anschließend werden für alle Wörter im dokument die bedingte Wahrscheinlichkeit, also die Likelihood bestimmt.

Danach kann für alle Klassen C_i und alle Wörter x_n die bedingte Wahrscheinlichkeit $P(C_i \mid (x_1, \dots, x_N))$ berechnet werden. Die Klasse mit dem maximalen Wert bestimmt anschließend das Dokument.

$$\max\{ P(C_i \mid (x_1,\dots,x_N)) = \prod_{x_j \in |D|} P(x_j|C_i) \cdot P(C_i) \}$$

Welche naive Annahme liegt dem Bayes Classifier zugrunde?

Bei der Naive Bayes Classification wird von der vereinfachenden Annahme ausgegangen, dass die Eingabevariabeln x_i voneinander unabhängig sind. Also alle Wörter unabhängig voneinander sind.

Ist diese Annahme im Fall der Dokumentklassifikation tatsächlich gegeben?

Nein, da manche Wörter häufig gemeinsam vorkommen. Beispielsweise werden Artikel (der, die, das, …) stets mit einem Nomen auftreten.

Betrachten Sie die Formeln 5 und 6. 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?

Kommt Wort nicht in den Trainingsdaten vor und wurde somit nicht klassifiziert, so würde das Produkt aus den bedingten Wahrscheinlichkeiten 0 ergeben (“Veto-Recht”).

Man könnte also schon bei einem fehlenden Wort pro Klasse durch das “Veto-Recht” keine Klassifizierung vornehmen.

Eine Lösung dafür wäre ein gewichteter Ansatz, wie das “Smoothing”.
Hierbei wird den Wörtern jeweils durch die angenommene Wahrscheinlichkeit eines Wortes in einer Kategorie ein Gewicht verpasst. Die Likelihood wird ersetzt durch:

$$P_{weight}(x_j \mid C_i) = (w \cdot P_{ass,i} + |x_j| \cdot P(x_j \mid C_i)) / (w + |x_j|)$$

3 Fragen zum Versuch

Was wird mit Evidenz bezeichnet und warum muss diese für die Klassifikation nicht berücksichtigt werden?

Die Evidenz kann für die Klassifikation vernachlässigt werden, weil sie unabhängig von der jeweiligen Klasse ist. Also bei den a-posteriori-Wahrscheinlichkeiten den gleichen Wert hat.

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)

Wir haben den Wert so belegt, dass er nach Anzahl der möglichen Kategorien variiert.
Bei “n” Kategorien, wird der Wert immer “1/n” annehmen. Somit wird davon ausgegangen, dass die Wahrscheinlichkeit für ein in einer Kategorie nicht trainiertes Wort für alle Kategorien immer gleich ist.

Man könnte diesen Wert aber auch anders belegen, um manchen Kategorien eine größere Wahrscheinlichkeit für untrainierte Worte zu geben.

Was könnten Sie mit dem in dieser Übung implementierten Classifier noch klassifizieren? Geben Sie eine für Sie interessante Anwendung an.

Text-Recommender:
Beim surfen oder lesen von Texten könnte beobachtet werden, ob Texte gelesen oder übersprungen würden. Diese werden dann als “recommended” bzw. “not recommended” klassifiziert. Anschließend kann man beliebige ungelesene Texte testen, einteilen und dem leser vorschlagen.

Eine Abwandlung davon wäre es, den Classifier mit Suchworten zu trainieren und dann nach Texten mit diesen Worten zu suchen. Damit hätte man quasi eine eigene Recherche-Suchmaschine.

Data Mining and Pattern Recognition	**04 Spam Filter**	17.11.2013	Dirk Fritz Steffen Kolb

Trends-Tool:
Vergleichbar mit http://techtrends.mi.hdm-stuttgart.de/ könnte ein rudimentäres Trendstool erstellt werden. Man trainiert mit bekannten Tech-RSS-Feeds und gibt lediglich die am häufigsten genannten Worte aus.

Bibliotheksystem:
In einem Bilbiothektsystem könnten vorhanden Buchtexte (z.B. Inhaltsangabe) zusammen mit den vorhanden Kategorisierungen als Trainingsdaten für den Classifier genutzt werden.
Neue Bücher müssten nur vom Classifier bewertet werden und könnten sofort eingegliedert werden.

Ein Beispiel für diese Anwendung ist unter “genreDetection.py” implementiert.
Dabei werden verschiedene Bücher vom “Project Gutenberg” (http://www.gutenberg.org/) geladen und für das Genre “Adventure” oder “Medicine” trainiert.
Anschließend wird ein kurzes Textstück aus einem Adventure-Buch klassifiziert.

Ergebnis:
scoreAdenture: 1.41809059835e-28
scoreMedicine: 1.27233247138e-32

Sprachen-Erkennung:
Vergleichbar mit dem Bibliothekssystem kann ein Wortschatz für Sprachen gelernt werden und unbekannte Texte dann für diese klassifiziert werden.

Ein Beispiel dazu befindet sich in der datei “languageDetection.py”.
Hierbei wird auf deutsche und englische Bücher trainiert und anschließend ein englischer Text klassifiziert.

Ergebnis:
scoreGerman: 3.60046709373e-114
scoreEnglish: 1.04245170785e-20