# Vergleich verschiedener Klassifikationsverfahren zur Spam-Klassifizierung
# EFME LU WS 2011/12 Übung 4

Philipp Omenitsch, Georg Sperl und Michael Osl

January 25, 2012

**Abstract**

In dieser Arbeit vergleichen wir die Klassifikationsverfahren k-Nearest-Neighbor, Perceptron und Mahalanobis-Distanz hinsichtlich ihrer Laufzeit und Ergebnisse auf einer Datenbank mit Spam-Nachrichten. Zusätzlich untersuchen wir zwei Verfahren, um die Anzahl der Features zu reduzieren.

# Contents


**1 Einführung** **1**
- 1.1 Verwendete Klassifikatoren . . . 1
  - 1.1.1 k-Nearest-Neighbor . . . 1
  - 1.1.2 Perceptron . . . 1
  - 1.1.3 Mahalanobis-Distanz . . . 2
- 1.2 Vorgehensweise . . . 2
- 1.3 Feature-Reduktion . . . 2
  - 1.3.1 Entfernung von Features mit geringer Varianz . . . 2
  - 1.3.2 Schrittweise Entfernung des Features mit dem geringsten Effekt. . . . 3

**2 Ergebnisse** **4**
- 2.1 Verwendung aller Features . . . 4
  - 2.1.1 k-Nearest-Neighbor . . . 4
  - 2.1.2 Perceptron . . . 5
  - 2.1.3 Mahalanobis-Distanz . . . 6
- 2.2 Egebnisse nach Feature-Reduzierung . . . 6
  - 2.2.1 Entfernung von Features mit geringer Varianz . . . 6
  - 2.2.2 Schrittweise Entfernung des Features mit dem geringsten Effekt . . . 6
  - 2.2.3 Auswirkung der Feature-Größe auf die Laufzeit . . . 8

**3 Schlussfolgerungen** **10**

**Bibliography** **11**

# Chapter 1

# Einführung

Das effektive Ausfiltern von unerwünschten E-Mail-Nachrichten (oder kurz Spam-Nachrichten, nachfolgend nur mehr kurz ,,Spam“) ist eine wichtige praktische Aufgabe von maschinellen Lernsystemen. Wichtig ist dabei besonders, die Anzahl der ,,False Positives“ möglichst gering zu halten, da eine legitime E-Mail-Nachricht, welche versehentlich als Spam klassifiziert wird, unangenehme Konsequenzen haben kann.

In unserer Arbeit vergleichen wir die Ergebnisse von drei verschiedenen Klassifikationssystemen: k-Nearest-Neighbor-Algorithmus (kNN), Perceptron und der Mahalanobis-Distanz sowohl hinsichtlich ihrer Ergebnisse als auch ihrer Laufzeit anhan der frei verfügbaren Spambase-Datenbank.

## 1.1 Verwendete Klassifikatoren

In diesem Abschnitt werden die von uns verwendeten Klassifikatoren kurz vorgestellt.

### 1.1.1 k-Nearest-Neighbor

Beim k-Nearest-Neighbor werden die Daten eines Trainings-Sets in einem $n$-dimensionalen Raum aufgeteilt, wobei $n$ die Anzahl der Features ist. Wird ein Element klassifiziert, dann wird es jener Kategorie zugeordet, welcher der Mehrheit der nähesten $k$ Elemente angehören.

### 1.1.2 Perceptron

Das Perceptron ist eine einfache Variante eines neuronalen Netzes. Das Prinzip wurde erstmals im Jahre 1958 von Frank Rosenblatt veröffentlicht[Ros58]. Es handelt sich dabei um eine lineare Diskriminantenfunktion. Während des Lernvorganges wird ein Vektor mit Gewichten erstellt, welcher dann anschließend eine Klassifikation vornimmt. Das Ergebnis wird anschließend durch eine Signum-Funktion dargstellt.

### 1.1.3 Mahalanobis-Distanz

Die Mahalanobis-Distanz ist ein Distanzmaß, welches nach Mahalanobis benannt ist und von diesem erstmals 1936 vorgestellt wurde [Mah36]. Bei diesem Klassifikator wird ein Objekt als jenes Objekt klassifiziert, welches die geringste Mahalnobis-Distanz aufweist. Wir berechnen die Mahalanobis-Distanz mit folgender Formel:

```
distances(i) = (objectvector - trainedSet{i, 1}) *
          inv(trainedSet{i, 2}) * (objectvector - trainedSet{i, 1})';
```

## 1.2 Vorgehensweise

Für den Vergleich der verschiedenen Spam-Klassifikatoren verwenden wir die „Spambase"-Datenbank welche wir vom UC Irvine Machine Learning Repository[1] heruntergeladen haben. Von diesen sind 1813 als Spam gekennzeichnet. Insgesamt haben wir von diesen 4601 Datensätzen 4140 ($\approx 90$ %) für das Trainings-Set $n$ und 461 ($\approx 10$ %) für das Test-Set $t$ verwendet. Jeder Test-Datensatz besitzt jeweils Informationen zu 58 Features $d$. Das Verhältnis $n$ zu $d$ liegt damit bei $71,38$. Vor der Verarbeitung der Daten werden diese noch auf das Traingins-Set normalisiert. Jede Spalte $t_i$ wird dabei durch $max(n_i)$ dividiert.

## 1.3 Feature-Reduktion

Die Anzahl der Features spielt eine wichtige Rolle für die Klassifizerung. Je weniger Features verwendet werden, desto weniger Zeit wird für die Klassifzierung benötigt (wie wir in Abschnitt 2.2.3 noch zeigen werden). Um die Features reduzieren zu können, haben wir haben uns dazu entschieden, die folgenden zwei Vorgehensweisen näher zu untersuchen:

- Entfernung von Features mit geringer Varianz.
- Schrittweise Entfernung des Features mit dem geringsten Effekt.

### 1.3.1 Entfernung von Features mit geringer Varianz

Die Idee hinter dieser Art der Dimensionsreduktion ist, jene Features zu entfernen, die eine geringe Varianz aufweisen und deshalb dazu führen, dass keine eindeutige Zuordnung möglich ist. Dazu wird erst für jede Feature-Spalte des Traings-Sets $n_i$ die Varianz gebildet. Anschließend wird eine Schranke $max_{var}$ definiert, über der die Varianz der Feature-Spalte liegen muss, um berücksichtigt zu werden.

[1] `http://archive.ics.uci.edu/ml/`

### 1.3.2 Schrittweise Entfernung des Features mit dem geringsten Effekt.

Hier war folgender Gedanke ausschlaggebend: Ausgehend von den vorhandenen $n$ Features wird jenes ermittelt, welches den geringsten Effekt hat. Mit ,,geringster Effekt“ ist dabei jenes Feature gemeint, das den Wert für die *false positives* am wenigsten negativ beeinflusst, da wir diesen Wert so gering wie möglich halten wollen. Das durch dieses Verfahren ausgewählte Feature wird anschließend entfernt und aus dem daraus resultierenden Set wird der Schritt dann so lange wiederholt, bis eine definierte Grenze an Iterationen erreicht wurde oder das Verfahren abgebrochen wird.

# Chapter 2

# Ergebnisse

In diesem Abschnitt werden nun die von uns gefundenen Ergebnisse dargestellt und diskutiert.

## 2.1 Verwendung aller Features

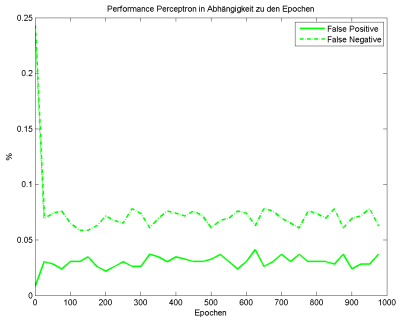
Zuerst betrachten wir die Ergebnisse der einzelnen Klassifikatoren, um optimale Parameter zu finden.

### 2.1.1 k-Nearest-Neighbor

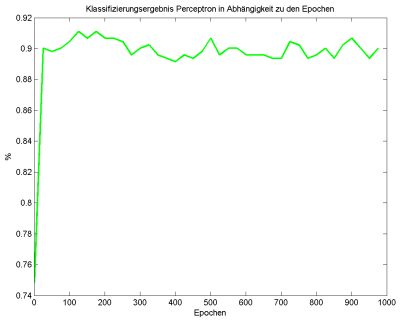
Beim k-Nearest-Neighbor-Klassifikator haben wir empirisch untersucht, wie sich eine Veränderung des $k$-Wertes im Ergebnis und der Laufzeit niederschlägt. Das Ergebnis ist in nachstehender Tabelle dargestellt:

| **k** | **Laufzeit** | **False Positives** | **False Negatives** | Richtige Zuordnung |
|---|---|---|---|---|
| 1 | 7,771272 | 3,9046 % | 3,6876 % | 92,4078 % |
| 2 | 7,683023 | 1,7354 % | 6,7245 % | 91,5401 % |
| 3 | 7,675488 | 3,6876 % | 4,7722 % | 91,5401 % |
| 4 | 7,689970 | 1,7354 % | 6,5076 % | 91,7570 % |
| 5 | 7,711205 | 2,8200 % | 4,9892 % | 92,1909 % |

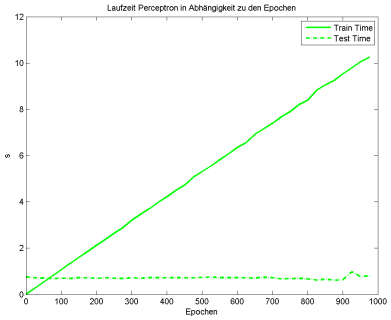
Wie man sieht, hat die Wahl des Wertes von $k$ keinen Einfluss auf die Laufzeit. Dies ist auch intuitiv einleuchtend, da die Ergebnisse ohnehin nach Entfernung sortiert werden müssen und es daher von der Laufzeitkomplexität keinen wesentlichen Unterschied macht, ob nun jene $n$ Ergebnisse herangezogen werden, die am nähesten liegen, oder eben $n+1$ Ergebnisse. Sehr wohl hingegen kann man einen Unterschied bei den Klassifikationsergebnissen ausmachen. Da das Ergebnis für den *False Positive*-Wert bei einem $k$-Wert von 2 am geringsten ist, verwenden wir diesen für den Klassifikator.




(a) Auswirkung auf die False Positives und False Negatives




(b) Auswirkung auf das Gesamtergebnis




(c) Auswirkung auf die Laufzeit

Figure 2.1: Performence des Perceptron-Klassifizierers in Abhängigkeit der Epochen.

### 2.1.2 Perceptron

Beim Perceptron-Klassifikator haben wir untersucht, wie sich die Anzahl der Epochen sowohl auf die Laufzeit als auch das Klassifikationsergebnis auswirkt. Wie sich zeigt ist die Laufzeit, die für das Trainieren des Klassifikators notwendig ist, annähernd linear. Das interessanteste Ergebnis ist jedoch, dass das Verhältnis der False-Positive-Klassifikationen zu der Gesamtanzahl der Daten im Test-Set anfangs sehr stark abnimmt, mit zunehmender Anzahl an Epochen jedoch immer nur noch schwächere Verbesserungen bringt. Auch hinsichtlich des Gesamtergebnisses bringt eine zunehmende Steigerung der Epochen-Anzahl über einen bestimmten Wert hinaus keine signifikanten Veränderungen mehr. Die Ergebnisse sind zusammenfassend in Bild 2.1 dargestellt.

### 2.1.3 Mahalanobis-Distanz

Da es bei dem Klassifikationsverfahren mit der Mahalanobis-Distanz keine weiteren Parameter gibt, die man modifzieren kann, ist hier nur das Gesamtergebnis des Klassifkatiors dargestellt:

| False Positives | False Negatives | Richtige Zuordnung |
|---|---|---|
| 6,07 % | 6,51 % | 87,42 % |

## 2.2 Egebnisse nach Feature-Reduzierung

In diesem Abschnitt werden die Ergebnisse unserer beiden in Abschnitt 1.3 erwähnten Verfahren besprochen, mit der die Anzahl der Features reduziert werden soll.

### 2.2.1 Entfernung von Features mit geringer Varianz

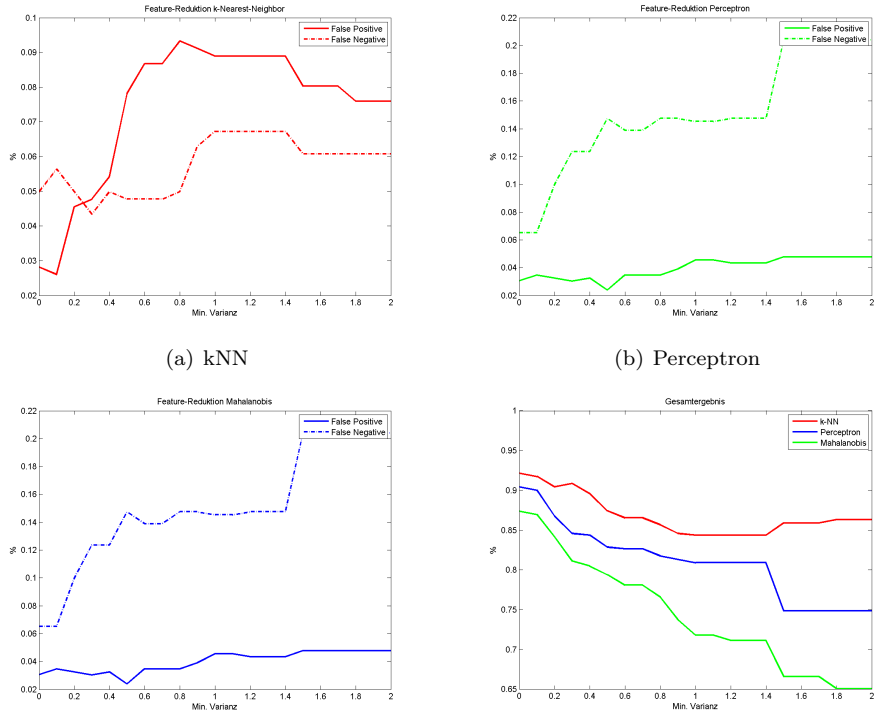
Mit dieser Methode konnten wir recht gute Ergebnisse erzielen. Abhängig von dem gewählen Klassifikationsverfahren lassen sich auch noch akzeptable Ergebnisse mit einer deutlich reduzierten Anzahl an Features erreichen. Die folgende Tabelle gibt eine Übersicht, wie viele Features bei welcher Grenze noch übrig geblieben sind:

| **Grenze für Varianz** | **Anzahl Features** |
|---|---|
| 0 | 58 |
| 0.1 | 44 |
| 0.2 | 31 |
| 0.3 | 23 |
| 0.4 | 19 |
| 0.5 | 16 |
| 1 | 10 |
| 2 | 7 |

Die Werte für die *False Positive*, *False Negative* sowie die Gesamtanzahl der korrekt kategorisierten Ergebnisse sind in Bild 2.2 dargestellt.

### 2.2.2 Schrittweise Entfernung des Features mit dem geringsten Effekt

Diese Methode ist sehr rechenintensiv, da man, um jenes Feature auswählen zu können, welches Entfernt werden soll, das Gesamtergebnis bei $n$ Features $n$ mal berechnen muss. Wir haben uns daher entschlossen, zuerst nur für den Perceptron-Classifier anzuwenden, um abschätzen zu können, wie erfolgsversprechend dieses Verfahren ist.




(a) kNN

(b) Perceptron

(c) Mahalanobis

(d) Zusammengefasst

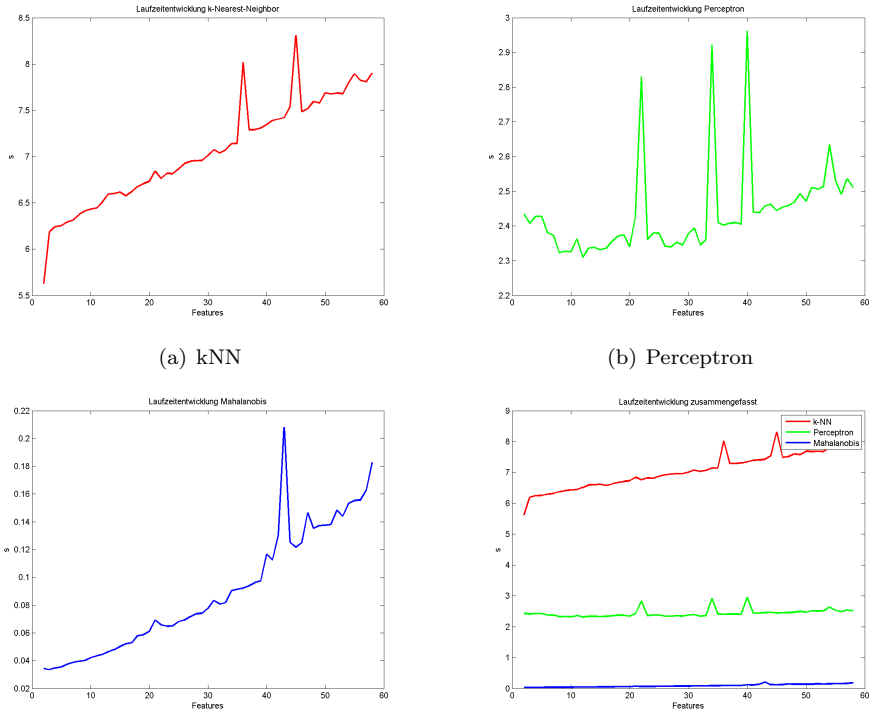
Figure 2.2: Darstellung der Ergebnisse der Feature-Reduktion für die einzelnen Klassifikatoren.

| Entfernte(s) Feature(s) | False Positives | False Negatives | Richtige Zuordnung |
|---|---|---|---|
| 17 | 4,1215 % | 5,4230 % | 90,4555 % |
| 17,44 | 4,3384 % | 5,4230 % | 90,2386 % |
| 17,44,42 | 4,9892 % | 5,2061 % | 89,8048 % |
| 17,44,42,1 | 4,9892 % | 5,4230 % | 89,5879 % |
| 17,44,42,1,2 | 4,9892 % | 4,9892 % | 90,0217 % |

Wie aus der Tabelle ersichtlich ist, kann diese Methode keinen der Werte entscheidend verbessern. Angesichts der Nachteile (Lange Laufzeit und keine signifikante Veränderung der Ergebnisse) haben wir uns entschlossen, diese Methode nicht weiter zu verfolgen.

### 2.2.3 Auswirkung der Feature-Größe auf die Laufzeit

Unabhängig von der konkreten Auswahl der Features haben wir auch untersucht, wie sich eine Reduzierung der Features auf die Laufzeit auswirkt. Die Ergebnisse sind in Bild 2.3 dargestellt. Das wichtigste Ergebnis ist hierbei, dass sich die Laufzeit für alle drei Klassifikationsverfahren in dem von uns untersuchten Bereich annähernd linear entwickelt.




(a) kNN

(b) Perceptron

(c) Mahalanobis

(d) Zusammengefasst

Figure 2.3: Darstellung der Laufzeiten der einzelnen Klassifikationsverfahren im Zusammenhang mit der Anzahl der Features.

# Chapter 3

# Schlussfolgerungen

Der Vergleich der genannten Verfahren hat gezeigt, dass jedes Klassifikationsverfahren seine spezifischen Vor- und Nachteile besitzt, um damit Spam-Nachrichten zu klassifizeren. Der kNN-Klassifkator liefert gute Ergebnisse, hat aber die längste Laufzeit aller drei verglichenen Verfahren. Ein großer Vorteil des Perceptron-Klassifiers ist, dass die Lernphase von der Klassifizerungsphase getrennt ist. Insgesamt liefert der Perceptron-Classifier bei im Vergleich zum kNN-Klassifikators reduzierter Laufzeit ähnlich gute Ergebnisse wie der kNN-Classifier, weist jedoch eine höhere Anzahl an false positives aus. Die Klassifizierung unter Verwendung der Mahalanobis-Distanz hat sich als das schnellste Verfahren herausgestellt, allerding zum Preis eines insgesamt schlechteren Klassifizierungs-Ergebnisses.

Wir konnten ebenfalls zeigen, dass sich mit einem Varianzen-basierten Verfahren die Anzahl der Features zu einem gewissen Grad reduziert werden kann, um dadurch Laufzeit einzusparen und dennoch akzeptable Ergebnisse zu erhalten. Ein Verfahren basierend auf iterativer Entfernung von Features mit geringem Einfluss hat sich hingegen als nicht zielführend erwiesen.

# Bibliography

[Mah36] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, April 1936.

[Ros58] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.