Merkmalsreduzierung und Klassifikation zur Erkennung von Parkin-Mutationen

Kevin Böckler

1 Einleitung

Dieser Bericht befasst sich mit der Vorgehensweise der Erkennung von Parkin-Mutationen aufgrund einer statistischen Datenerhebung. Ziel des Projekts war es, aus einem gegebenem Datensatz zur Klassifizierung verschiedener Testergebnisse einen geeigneten Klassifikator zu entwickeln, der mit einem Bruchteil der Daten auskommt und zudem für unbekannte, folgende Tests Aussagen zur Erkrankung treffen kann.

1.1 Vorgehen

In den folgenden Abschnitten wird auf durchgeführte Methoden und Experimente eingegangen und deren Ergebnisse diskutiert. Zunächst liegt es nahe, einen oder mehrere Klassifikatoren für das gegebene Klassifikationsproblem auszuwählen. Darüber hinaus muss dieser im Entwicklungsprozess auf verschiedene Arten und Zielsetzungen hin geprüft und validiert werden. In diesem Rahmen geschieht eine Reduzierung der Merkmale des gegeben 20-Merkmals-Problems auf ein 6-Merkmals-Problem. Ziel hierbei ist es, die für den gewählten Klassifikator bestmöglichen sechs Merkmale aus dem Trainingsdatensatz zu wählen und für die Mustererkennung zu verwenden.

1.2 Grundlage der Klassifikation

Bei der Bearbeitung des gegebenen Problems stellte sich die Frage, ob mehrere Klassifikatoren betrachtet werden können oder direkt einer als Grundlage verwendet wird. In unserem Fall wurde die Klassifikation auf die Verwendung eines linearen Klassifikators festgelegt, der mittels n-dimensionaler Hyperplanes die Datensätze unterteilt und eine Diskriminanzfunktion hieraus erstellt.

Der hier verwendete Klassifikator erzeugt aus dem Trainingsdatensatz eine feste Klassifikationsmatrix A, die fortan für ausstehende Mustererkennungen mittels einfacher Matrix-Vektor-Multiplikation errechnet werden kann. Die ermittelte Klasse ergibt sich hierbei aus dem Maximum der berechneten Diskrimnanzwerte einer jeden Klasse.

2 Methoden

Die weitere Modifikation des Klassifikators unterliegt nun zum Einen der Validierung seiner Effektivität bezüglich der korrekten Aussagen, zum Anderen der Art der Merkmalsreduktion von zwanzig auf sechs Merkmale, für die die resultierende Effektivität groß-möglichst bleibt.

2.1 Validierung des Klassifikators

Die Validierung lässt sich anhand der gegebenen Trainingsdaten auf zwei sinnvolle Arten durchführen. Die im folgenden näher beschriebene Testset = Trainingsset - Validierung und die Kreuz-Validierung stellen dabei grundsätzliche Ansätze dar, die mit einer kompakten Menge an Trainingsdaten auskommt, um approximativ ausreichende Aussagen über die Effektivität zu formulieren.

2.1.1 Testset = Trainingsset - Validierung (TT)

Die TT[1]-Validierung trainiert den Klassifikator mit 100% der Trainingsdaten und überprüft die Effektivität durch Wiedererken-

1. Testset = Trainingsset

nung derselbigen. Dieses Verfahren ist insofern aussagekräftig, dass der Klassifikator eine möglichst große Menge der zur Verfügung gestellten Daten zum Training nutzt. Allerdings gibt es Gegenbeispiele, für die diese Validierung keine Bedeutung besitzt (z.B. KNN-Klassifikation für $k = 1 \Rightarrow 100\%$).

2.1.2 Kreuz-Validierung

Die Kreuz-Validierung zerlegt die gegebenen Trainingsdaten in Trainings- und Testdaten. Diese Zerlegung ist disjunkt und dient dazu, dass der Klassifikator nicht direkt die Datensätze wiedererkennen kann, die er zuvor gelernt hat (dieses Verfahren bietet sich daher besonders bei KNN-Klassifizierungen an). Bei der in diesem Projekt verwendeten Kreuz-Validierung wird stets eine k-fold Kreuz-Validierung benutzt, die über mehrere Folds eine randomisierte Zerlegung der Daten vornimmt.

Die in den Experimenten verwendete Kreuz-Validierungen verwenden dabei ein Verhältnis von Trainings- zu Testdaten von sowohl 80% als auch 90% der ursprünglich gegebenen Trainingsdaten. Die Ergebnisse werden dabei im folgenden Kapitel gezeigt.

2.2 Merkmalsreduktion

Um die aus zwanzig Merkmalen bestehenden Trainingsdaten in einen Merkmalsraum von sechs Merkmalen zu reduzieren, werden je Test die Trainingsdaten auf eine Auswahl an sechs Merkmalen reduziert und der daraus resultierende Klassifikator trainiert. Dieser wird daraufhin validiert und dieses Ergebnis mit denen von anderen Merkmalsauswahlen verglichen.

Bei unserem Vorgehen wurde die Kreuz-Validierung ebenso wie die TT-Validierung eingesetzt.

3 Experimente

Die zuvor vorgestellten Methoden wurden nun in einigen verschiedenen Experimenten umgesetzt, um eine geeignete Merkmalsreduktion zu ermöglichen. Zunächst zeigt der nächste Abschnitt die Effektivität des Grundklassifikators anhand aller gegebener Trainingsdaten im ganzen Merkmalsraum. Anschließend finden sich zwei Experimente der Merkmalsreduktion mittels der TT-Validierung und der Kreuz-Validierung.

TABELLE 1
Grundklassifikator-Test

TT-Validierung	
⌀	0.8621
Kreuz-Validierung mit 80% Verhältnis	
1	0.6727
2	0.6722
3	0.6727
⌀	0.6725
Kreuz-Validierung mit 90% Verhältnis	
1	0.6984
2	0.6978
3	0.6968
⌀	0.6977

3.1 TT-Validierung des Grundklassifikators

Die Effektivität des Grundklassifikators ist in Tabelle 1 zu sehen. Die Daten zeigen hierbei die korrekten Treffer anteilig im Interval $[0, 1]$ - von 0% bis 100% korrekte Klassifizierung der Testdaten. Wie zu erwarten war, trifft die beste Rate bei der TT-Validierung ein, da der Klassifikator hier jeweils die meisten Datensätze trainieren konnte und zudem alle zu klassifizieren Instanzen bereits in sein Model einbeziehen konnte. Da der lineare Klassifikator allerdings mit Hilfe von Trennungs-Hyperplanes arbeitet, bleibt die Erkennungsrate in der TT-Validierung trotzdem aussagekräftig.

Interessant zu sehen ist, dass der Grundklassifikator bereits mindestens 67% der Validierungstests korrekt zuordnen kann. Im Folgenden wird nun eine Merkmalsreduktion durchgeführt, die diese Werte verändern wird.

3.2 Merkmalsreduktion

Dieses Projekt verwendete zwei verschiedene Arten der Herangehensweise, den Merkmalsraum für den Grundklassifikator zu reduzieren. Zunächst wird ein auf die TT-Validierung basierender Algorithmus unter einer bestimmten Prämisse die Merkmale auf

Aussagekraft prüfen. Anschließend zeigt der für dieses Projekt schließlich gewähltes Brute-Force-Verfahren, wie die TT-Validierung für eine optimale Lösung in Betracht gezogen wurde.

3.2.1 Ein randomisierter Ansatz

Die zugrunde liegende Vermutung besteht darin, dass jedes Merkmal für sich selbst eine gewisse Güte besitze. Ließe sich nun die Güte jedes Merkmals bestimmen, so ließe sich folgern, dass die sechs besten Merkmale bezüglich der Gütefunktion zusammenwirken und das beste Resultat liefern. Diese Annahme wird nun verwendet und später diskutiert werden anhand der Ergebnisse.

Die Güte eines Merkmals errechnet sich daraus, dass die Trainingsdaten auf dieses eine zu betrachtende Merkmal reduziert wird und nun ein k-fold Kreuz-Validierungstest durchgeführt, der mit einer Rate von 100.000 Folds eine jeweils zufällige Zerlegung in Trainings- und Testdaten vornimmt und diese prüft. Das Ergebnis liefert folgende Merkmale:

$$(x_1, x_2, x_7, x_8, x_{13}, x_{15})$$

Der zugehörige Kreuz-Validierungstest wurde mit 10.000 Folds durchgeführt und ergab mit dem auf diese sechs Merkmalen reduzierten Klassifikator eine Trefferrate von

$$\varnothing \approx 0.5755 \tag{1}$$

3.2.2 Bruteforce-Auswahl der Merkmale

Der nächste Ansatz begnügt sich mit der Annahme, dass der Klassifikator mit der höchsten TT-Validierungs-Trefferrate auch der optimale Klassifikator bezüglich der Merkmalsreduktion ist. Zu diesem Zweck wird im Folgenden jede Kombination von sechs Merkmalen aus zwanzig ($\binom{20}{6}$) ausgewählt und in einer TT-Validierung auf die Güte geprüft. Das Merkmalstupel mit der größten resultierenden Güte bzw. Trefferrate liefert den besten Klassifikator. Das Ergebnis sieht wie folgt aus:

$$(x_6, x_7, x_8, x_{10}, x_{12}, x_{18})$$

Diese Auswahl hat intern im Algorithmus eine TT-Validierungs-Trefferate von 0.7701 und ist damit die bestmögliche.

TABELLE 2
Kreuz-Validierung für Bruteforce-Lösung

Kreuz-Validierung mit 80% Verhältnis	
1	0.6498
2	0.6518
⌀	0.6508
Kreuz-Validierung mit 90% Verhältnis	
1	0.6579
2	0.6572
3	0.6583
⌀	0.6578

Im Vergleich zum vorherigen randomisierten Klassifikator wurde auch dieser Merkmalsraum in einer Kreuz-Validierung geprüft und ergab die in Tabelle 2 gezeigten Testergebnisse. Der insgesamt resultierende Durchschnittswert beträgt

$$\varnothing \approx 0.6543 \tag{2}$$

4 Diskussion

Nach der Durchführung der Experimente stellt sich die Frage nach dem Ergebnis der Tests und insbesondere der Antwort auf die Fragestellung, welcher Klassifikator bezüglich welcher Merkmalsreduktion der beste sei. Dem Projekt vorangehend wurde bereits festgelegt, dass der Klassifikator auf den linearen Klassifikator festgelegt ist, wie er bereits in Abschnitt 1.2 erstmals eingeführt wurde. Die ausstehende Frage ist nun, welche Merkmalsreduktion die beste sei. Zwei Verfahren wurden dazu im vorangegangenen Abschnitt durchgeführt und haben jeweils Ergebnisse geliefert. Im Folgenden werden die Experimente kurz gegenseitig verglichen.

4.1 Vergleich Methoden und Experimente

Im Vergleich stehen nun der Merkmalsraum der randomisierten Reduktion (1) und der Bruteforce-Reduktion (2). Es erscheint die Entscheidung sinnvoll, die Bruteforce-Reduktion höher zu bewerten, da mit dieser sichergestellt wurde, dass *alle* möglichen Reduktionen bezüglich einer k-fold Kreuz-Validierung getestet und gegeneinander

abgewogen wurden. Da beide Testverfahren mehrfach intern eine zufällige Zerlegung der Trainingsdaten benutzen, ergeben beide Methoden ein brauchbares Ergebnis bezüglich eines bayes'schen Ansatzes (Durchschnitt durch mehrfache Randomisierung, dafür Verlust der Varianz).

4.2 Ergebnisse und Auswahl

Vergleicht man nun diese beide Validierungswerte der ermittelten Merkmalsräume, so spricht für den optimalen Klassifikator bezüglich Reduktion die Variante der Bruteforce-Lösung. Diese Lösung scheint sowohl durch TT-Validierung (Methode der Reduktion liefert dieses) die größte Güte innezuhaben, als auch in der Kreuz-Validierung den höheren Wert mit $\varnothing \approx 0.6543$ Trefferrate zu erzielen.

Somit fällt unsere Wahl in Bezug auf den optimalen Klassifikator für das Projekt auf den Merkmalsraum

$$(x_6, x_7, x_8, x_{10}, x_{12}, x_{18})$$

5 Zusammenfassung

Der Klassifikator lässt sich nun in dem zuvor ermittelten Merkmalsraum trainieren und kann in Form einer 4x7-Matrix darstellen, die im Anschluss mit dem Testvektor[2] direkt multipliziert werden kann. Der resultierende Vektor gibt zeilenweise die Diskriminanz jeder der vier Klassen an; die Klasse mit der höchsten Diskriminanz gilt als gewählt. Dieser fertige Klassifikator kann einfach im Voraus trainiert werden und wird im abschließenden Wettbewerb direkt in der zu liefernden **erkenner.mat** kodiert.

Die in diesem Bericht beschriebenen Methoden wurden komplett in Matlab implementiert und mit mehreren Testläufen real getestet und deren Werte ermittelt. Alle in diesem Bericht vorkommenden Werte wurden so gemessen. Wie in dem letzten Abschnitt deutlich wurde, lässt sich ein Merkmalsraum von sechs Merkmalen mittels Bruteforce-Ansatz in kurzer Zeit (maximal 1 Minute Rechenzeit auf dem von uns benutzten Testrechner) determinieren. Für das Projekt scheint dieser Ansatz bezüglich der benutzten Methoden optimal.

Offen in diesem Bericht ist die Betrachtung anderer Klassifikatoren, für die eventuell neue Validierungstests nötig wären, um diese gegenseitig zu vergleichen. Es besteht die Vermutung, dass Klassifikatoren wie KNN-Klassifikatoren ebenso geeignet wären für das gestellte Problem.

2. Der Testvektor wird mit einer führenden 1 zu einem Vektor aus dem $\mathbb{R}^7$ erweitert

Anhang A
Die Klassifikator-Matrix A

Die folgende Matrix ist die finale Version, die am Wettbewerb teilnehmen wird:

$$A \approx \begin{bmatrix} -0,1422 & -2,5595 & -1,5778 & 5,2795 \\ -18,5734 & -7,3557 & 6,1550 & 19,7740 \\ -0,2675 & -2,7570 & 3,1695 & -0,1449 \\ 0,1368 & 1,6297 & -1,8884 & 0,1219 \\ -0,0482 & 0,0206 & 0,0060 & 0,0217 \\ 0,0469 & -0,0243 & 0,0079 & -0,0305 \\ 1,5783 & 3,5955 & 1,9943 & -7,1681 \end{bmatrix}^{\mathrm{T}}$$

Dank

Besonderer Dank der Autoren für die Bereitstellung des hier verwendeten LaTeX-Templates geht an http://www.ieee.org.

Kevin Böckler Java-Enthusiast, Webentwicklung.

Für mehr Informationen sei verwiesen auf die Homepage http://kevinboeckler.de.