Ensemble Models - Boosting, Bagging and Stacking

Maximilian Schwinger

3. Februar 2004

Inhaltsverzeichnis

1 Einführung

Unzählige Algorithmen zur Klassifikation, sogenannte 'learning algorithms', sind derzeit verfügbar. Jeder dieser Algorithmen hat seine Stärken und Schwächen, seine optimalen Einsatzgebiete und solche, in denen er schlechte Ergebnisse produziert. Nun liegt die Idee nahe verschiedenartige Methoden oder Klassifikatoren auf irgendeine Art zusammenzufassen, um von jedem das Beste zu erhalten. Solche Methoden werden Ensemble-Methoden genannt.

Nun stellt sich die Frage, auf welche Art die Ergebnisse von Klassifizierungen verbessert werden können. Dabei bemüht man sich den 'mean square error', den durchschnittlichen quadrierten Fehler, zu minimieren. Der MSE kann beschrieben werden als Komposition von Bias und Varianz . Verschiedene Methoden versuchen nun zumeist entweder Bias oder Varianz zu minimieren. Eine gleichzeitige minimierung beider Komponenten erscheint zumeist schwieriger.

$$E[(\hat{\theta} - \theta)^2] = E[(\hat{\theta} - E[\theta] + E[\theta] - \theta)^2]$$

$$= (E[\hat{\theta}] - \theta)^2 + E[(\hat{\theta} - E(\hat{\theta}))^2]$$

Hier sieht man links den quadrierten Bias

$$E[\hat{\theta}] - \theta$$

und rechts die Varianz

$$E[(\hat{\theta} - E(\hat{\theta}))^2]$$

1.1 Überblick

Im Folgenden wollen wir verschiedene Vertreter der Ensemble Methoden näher betrachten. Zunächst eine Methode die bei der Optimierung darauf setzt, den Bias zu minimieren, das Boosting. Daraufhin sehen wir uns die Methode des Bagging an, die hauptsächlich versucht die Varianz zu minimieren und abschliend Stacking, eine der interessantesten, aber auch kompliziertesten Methoden, bei der wohl noch einiges an Forschung nötig sein wird, um ihre Mechanismen zu verstehen.

2 Boosting

Das Boosting wurde das erste mal erwähnt in L.G.Valiant's 'A theory of the learnable' (1984) und basiert auf der Idee verschiedene schwache PAC-Lerner zu einem starken zu 'boosten'. Ein PAC-learner, wobei PAC für Probably Approximatly Correct steht, wird wie folgt definiert.

Sei C eine Konzeptklasse über X. C ist PAC-lernbar, wenn es einen Algorithmus gibt, der

-für jedes epsilon $0 < \epsilon < 0,5$

-für jedes delta $0 < \delta < 0,5$

-für jede Verteilung D über X und

-für jedes c aus C

mit der Wahrscheinlichkeit $1 - \delta$ eine Hypothese h generiert

-die einen Fehler h mit $h < \epsilon$ aufweist.

Wie zu sehen ist sind diese Kriterien sehr streng. Entsprechend schwer ist es somit auch einen, nach dieser Definition, starken PAC-learner zu finden. Viel einfacher ist es einen learner zu finden, der die PAC-Kriterien nur für ein festes Epsilon und ein festes Delta erfüllt. Wenn man nun einen schwachen PAC-Learner, also einen Lerner, der vielleicht nur ein wenig bessere Ergebnisse erzielt als das blosse Raten, hat versucht man seine Ergebnisse zu verbessern, indem man ihn wiederholt auf veränderte Versionen der Daten anwendet und die von den verschiedenen Algorithmen erzeugten gewichteten Hypothesen zu einer Gesamthypothese kombiniert.

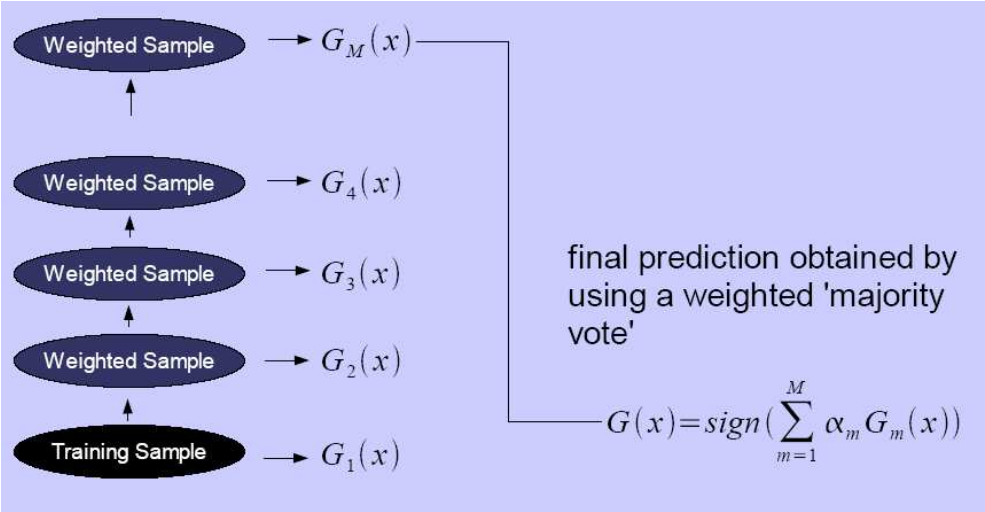

Abbildung 1: Prinzip des Boosting

2.1 Beispiel

Ein interessantes Beispiel für die Implementierung eines Boosting-Algorithm ist AdaBoost. Im folgenden wollen wir die Implementierung M1, die von Freud und Schapire 1997 publiziert wurde untersuchen. Bei dieser Implementierung wird der Bias der Vorhersage minimiert. Eine neuereVersion, AdaBoost.M2 kann sogar Bias undVarianz reduzieren. AdaBoost nimmt fr die Klassifizierung ein zwei - Klassen Problem an. Die Ergebnisse werden auf den diskreten Ergebnisraum -1, 1 abgebildet (Abbildung 1).

Der wichtigste und interessanteste Schritt ist nun der Schritt von einem 'Sample' zum nächsten, gewichteten, Sample. Beim ersten Durchlauf werden allen Trainings-Beobachtungen zunächst das selbe Gewicht zugewisen, welches bei einer menge von N Observations 1/N wäre. Nun wird in jedem darauf folgenden Schritt die Gewichtung an die erzielten Ergebnisse angepasst. Die genaue Anpassung soll nun anhand des PseudoCodes von AdaBoost.M1 genauer dargelegt werden.

2.2 Codebeispiel

1. Initialisierung der Gewichtungen

2. Für m=1 bis M ...

2.1 ...passe einen Classifier G unter Verwendung der Gewichte g an die Trainingsdaten an

2.2 ...Berechne:

$$err_m = \frac{\sum_{i=1}^{N} W_i I(Y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$

2.3 ...Berechne:

$$\alpha_m = log\frac{1 - err_m}{err_m}$$

2.4 ...setze:

$$w \leftarrow w_i e[\alpha_m \times I(y_i \neq G_m(x_i))], i = 1, 2...N$$

3. Gib Ergebnis aus:

$$G(x) = sign[\sum m = 1^M \alpha_m G_m(x)]$$

Nun wollen wir im einzelnen etwas detaillierter auf die durchzuführenden Schritte des Algorithmus eingehen. Der erste Schritt scheint intuitiv verständlich: Hier werden die Gewichte w mit dem Anwangswert, der in diesem Falle $1/overM$ ist initialisiert. Der Schritt 2 ist die sich wiederholende Schleife, in der die modifizierten Datenmengen bearbeitet werden. Im Schritt 2.2 wird zunächst der Fehler des m-ten Klassifikators berechnet. Im nächsten Schritt 2.3 wird aus

den gewonnenen Fehlerdaten ein Faktor alpha berechnet und im Schritt 2.4 daraus der neue Gewichtungsfaktor. Im Schritt 3 wird das Ergebnis, welches durch eine einfache Signum-Funktion über die Klassifikator-Ergebnisse berechnet wird ausgegeben.

2.3 Performance

In Abbildung 2 können wir den Erfolg des Boosings eines 'Single Stump', also eines Baumes mit nur einer Gabelung, sehen. Schon nach etwa 50 Boosting-Iterationen erreichen wir hier bessere Ergebnisse als ein Baum mit 400 Blättern

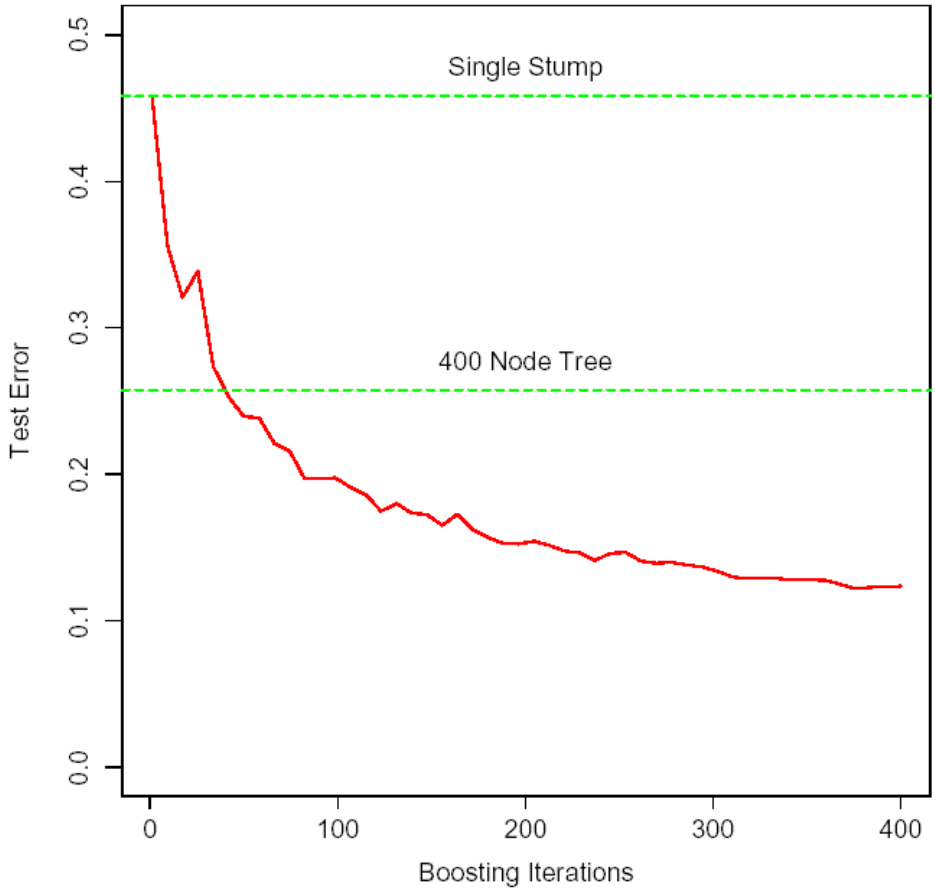

Abbildung 2: Performance

2.4 Anwendung

Nun wollen wir uns noch ein kleines Beispiel einer realen Anwendung ansehen. In Abbildung 4 sehen wir Daten zu Schrifterkennung mit Hilfe von Boosting. Die unterschiedlichen Zeilen sind jeweils der Status nach einer bestimmten Anzahl von Boosting-Iterationen. Die Zahlen unter den Beispieldaten sind folgendermassen

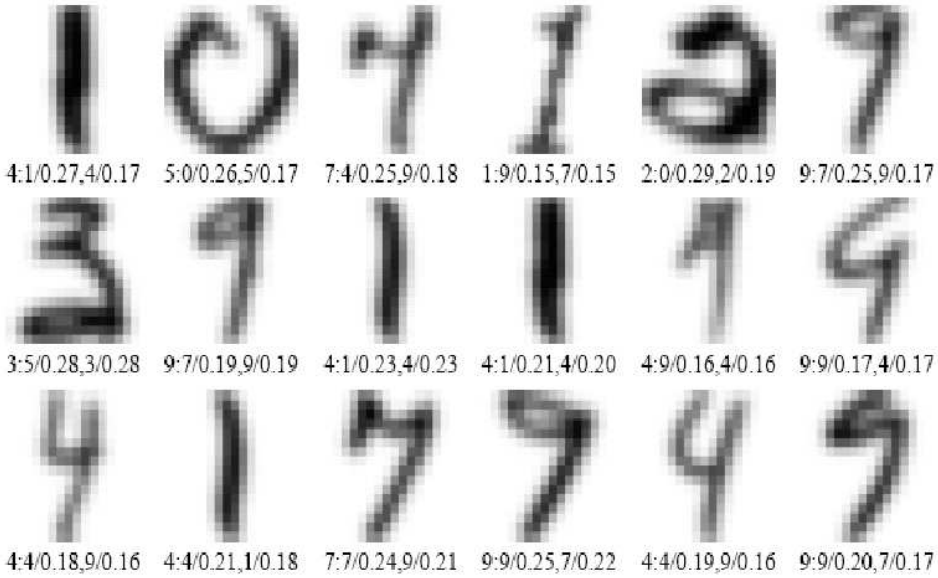

Abbildung 3: Schrifterkennung mit hilfe von geboosteten Klassifikatoren

zu verstehen: Die Struktur ist a:b/c,d/e. a ist das Label der aktuellen Probe. b das Label, das der Zeit das höchste Gewicht, nämlich das Gewicht c hat, d das Label das das zweichthöchste Gewicht, nämlich d hat. Es lässt sich deutlich erkennen wie die Klassifikation in den tieferen Reihen, d.h. nach mehr Boosting-Iterationen bessere Ergebnisse erzielt.

2.5 Resume - Boosting

Für das Boosting spricht, dass wir mit relativ wenig Arbeit ein sehr gutes Ergebnis erzielen können. Gegen das Boosting spricht, dass es Probleme mit verrauschten Datensätzen hat, da in diesem falle fehlklassifizierende Klassifikatoren, die das Rauschen 'richtig' erkennen ein hohes Gewicht erhalten.

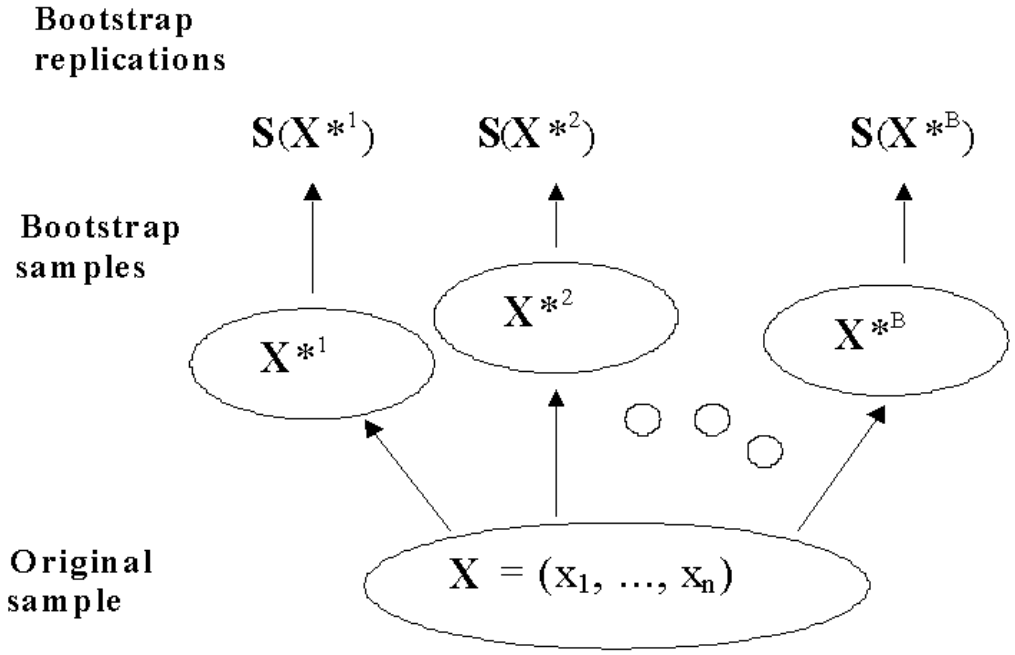

Abbildung 4: Prinzip des Bootstraping

3 Bagging

Das Verfahren des Bagging hat seinen Namen von Bootstrap Aggregation und wurde 1994 von Leo Breiman in 'Bagging Predictors' eingeführt. Die Idee bei der Bootstrap Aggregation ist es, aus einem einzelnen Lern-sample mehrere zu generieren.

Zunächst wollen wir uns das Verfahren des Bootstrappings genauer ansehen. Wie in Abbildung 4 zu erkennen ist haben wir als Grundlage das vorher angesprochene einzelne Lerndatenset. Nun ziehen wir aus dem Datenset so oft (mit zurücklegen) bis wir einen neuen Datensatz generiert haben, welcher die selbe Kardinalität hat wie unser Ursprungsdatensatz. Beim Bagging werden nun aus diesen sogenannten Bootstrapreplikanten jeweils ein neuer Klassifikator generiert. Aus der Gesamtheit der Hypothesen dieser Klassifikatoren wird durch eine majority vote eine Gesamthypothese gebildet.

3.1 Beispiel

Als kleines Beispiel können wir in der folgenden Abbildung Entscheidungsbäume sehen, die auf Bootstrapreplikanten 'gewachsen' sind (Abbildung 5).

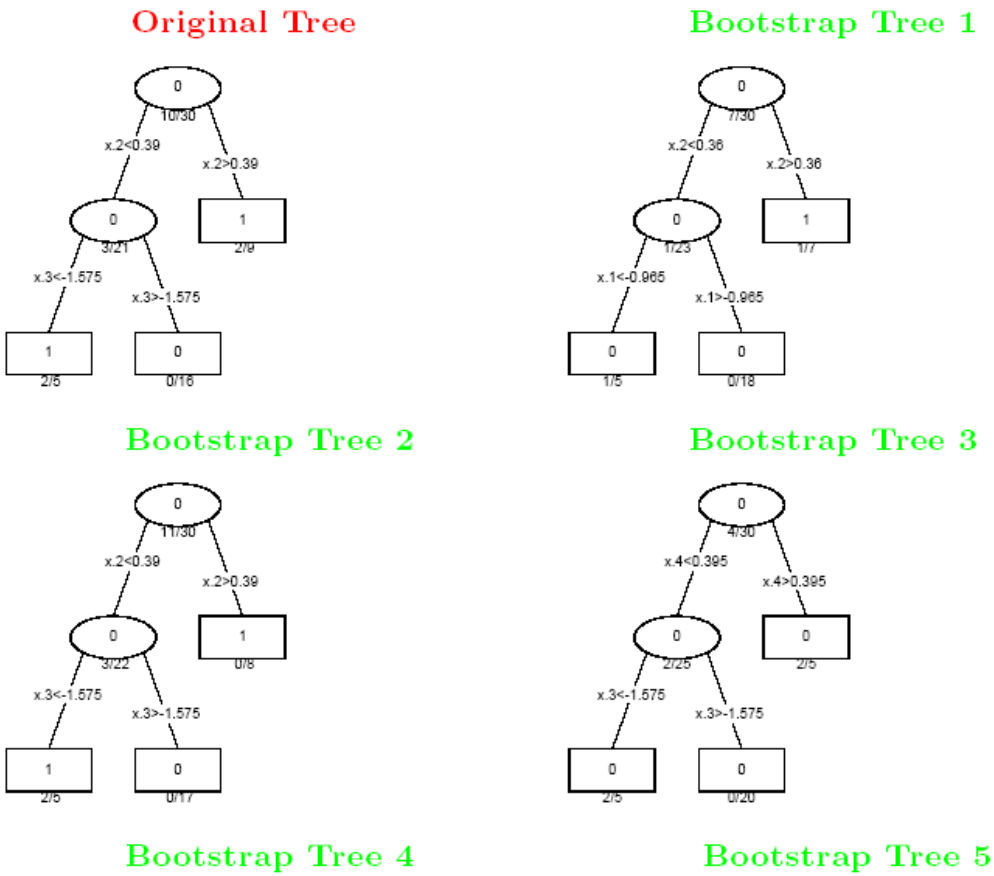

Abbildung 5: decision-trees, 'gewachsen' auf Bootstrapreplikanden

3.2 Resume - Bagging

Durch das Replizieren des Lerndatensatzes ist es möglich insgesamt die Varianz des Ergebnisses zu verkleinern. Ein Problem kann beim Bagging die grosse Anzahl von nötigen Bootstrapreplikanten und damit die performance sein.

4 Stacking

Stacking wurde das erste mal in D.Wolperts 'Stacked generalisation' 1992 erwähnt. Die Idee hinter Stacking ist es, mehrere unterschiedliche Learningalgorithmen mit verschiedenen Stärken und Schwächen auf das selbe Problem anzusetzen und deren Ergebnis dann mit einem weiteren learning algothm dessen Aufgabe es ist zu lernen, welcher Algorithmus in welchen Fällen gute Entscheidungen trifft, zu einem Ergebnis zusammenzufassen. Die einzelnen Klassifikatoren , die auf die Daten selbst angesetzt werden nennt man base - level - classifier oder level - 0 - classifier, den Klassifikator, der die Ergebnisse aller level - 0 - classifier als Input bekommt nennt man meta - classifier oder level - 1 - classifier. Insgesamt ist zu sagen, dass es schwer ist zu entscheiden, welcher Klassifikator gut als Meta-Klassifikator geeignet ist, da man noch nicht die 'Mechanismen' des Stackings verstanden hat.

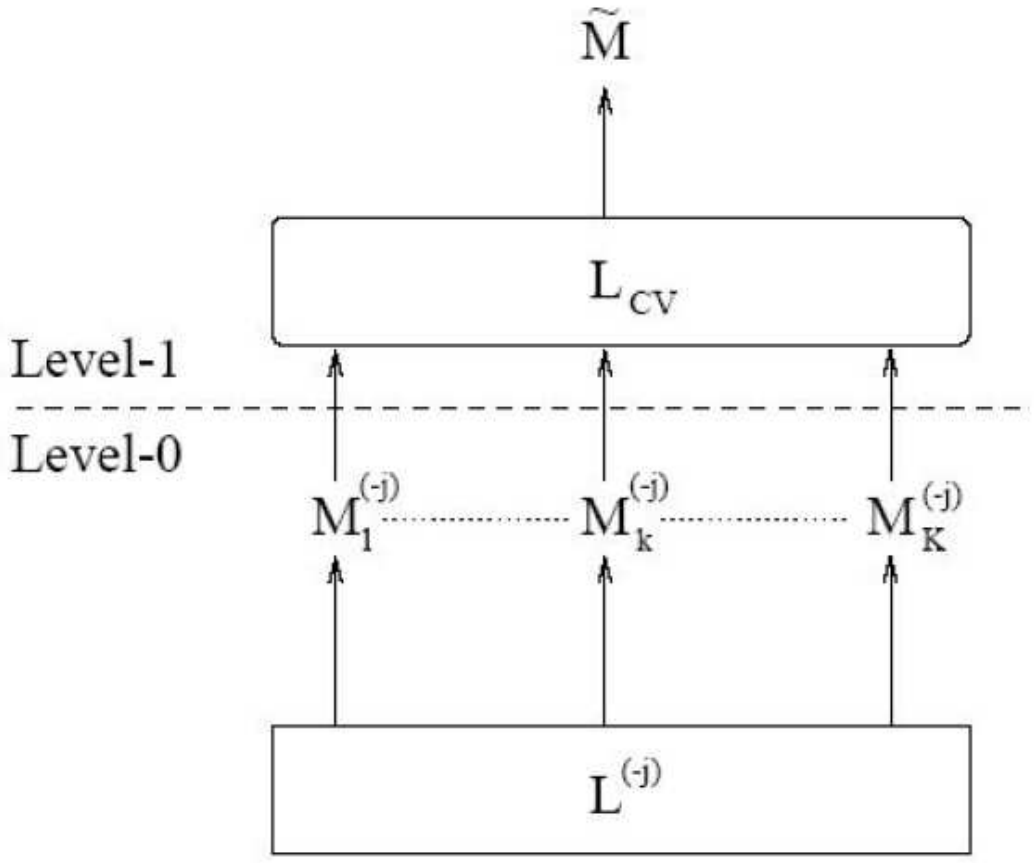

Abbildung 6: Prinzip des Stacking

4.1 Beispiel

Als Beispiel wollen wir das Stacken eines C4.5, eines NB, eines IB1 im level 0 und eines MLR im Level 1, welches in Kai Ming Ting und Ian H. Wittens 'Issues in Stacked Generalisation' (1998) angeführt wird, nennen. Zunächst wollen wir die hier verwendetetn Klassifikatoren kurz vorstellen. e -Der C.4.5-Algorithmus ist ein Algorithmus der aus den lerndaten einen Entscheidungsbaum kreiert.

-Der NB ist der einfache naive Bayessche Algorithmus

-Der IB1 ist ein Instanz-basierter Lernalgorithmus. Fr jede Klasse wird ein Beispiel gespeichert. Wenn nun neue Daten klassifiziert werden sollen ordnet er sie der Klasse zu, deren Repräsentant den neuen Daten am nächsten liegt.

-Der MLR ist eine Variante der least-square linear regression

Um nun mit Hilfe der Level-0-Klassifikatoren unser Level - 0 Modell und unsere Level 1 Daten zu erzeugen benutzen wir die j-fold Cross - Validation, die hier kurz vorgestellt werden soll.

4.2 j-fold Cross Validation

Die Idee bei der j-fold Cross Validation ist es, die uns zur verfügung stehenden Daten in j Partitionen gleicher Grösse zu zerlegen. Nun wird jede der Partitionen einmal als Lerndaten verwendet, während alle anderen als Testdaten zur verfügung stehen. Im nächsten Schritt generieren wir aus den verschiedenen Ergebnissen, die durch die Datenmengen erzeugt wurden durch einen 'majority vote' ein Gesamtergebnis.

4.3 Die eigentliche Prozedur des Stacking

4.3.1 Vorbereitung der Daten

Wir setzen nun jedem der Base-Level-Classifier die durch die j-Fold Cross Validation generierten Lerndatensätze zu Generierung eines Level-0-Modells vor. Beim Test der Daten mit dem Testdatensatz der J-Fold-CrossValidation merken wir uns für jede Instanz x_n aus dem Testdatensatz die Vorhersage des k-ten Modells y_{kn}.

Am Ende des Cross Validation-Prozesses sieht der Datensatz, der aus den K Level-0 Modellen generiert wurde wie folgt aus:

$$\mathcal{L}_{CV} = \{(y_n, z_{1n}, ...z_{Kn}), n = 1, ...N\}$$

Dies sind nun unsere Level-1 Daten auf die wir unseren Level-1 oder Meta-Level Generalisierer anwenden, um das gesamte gestackte Ergebnis zu erhalten.

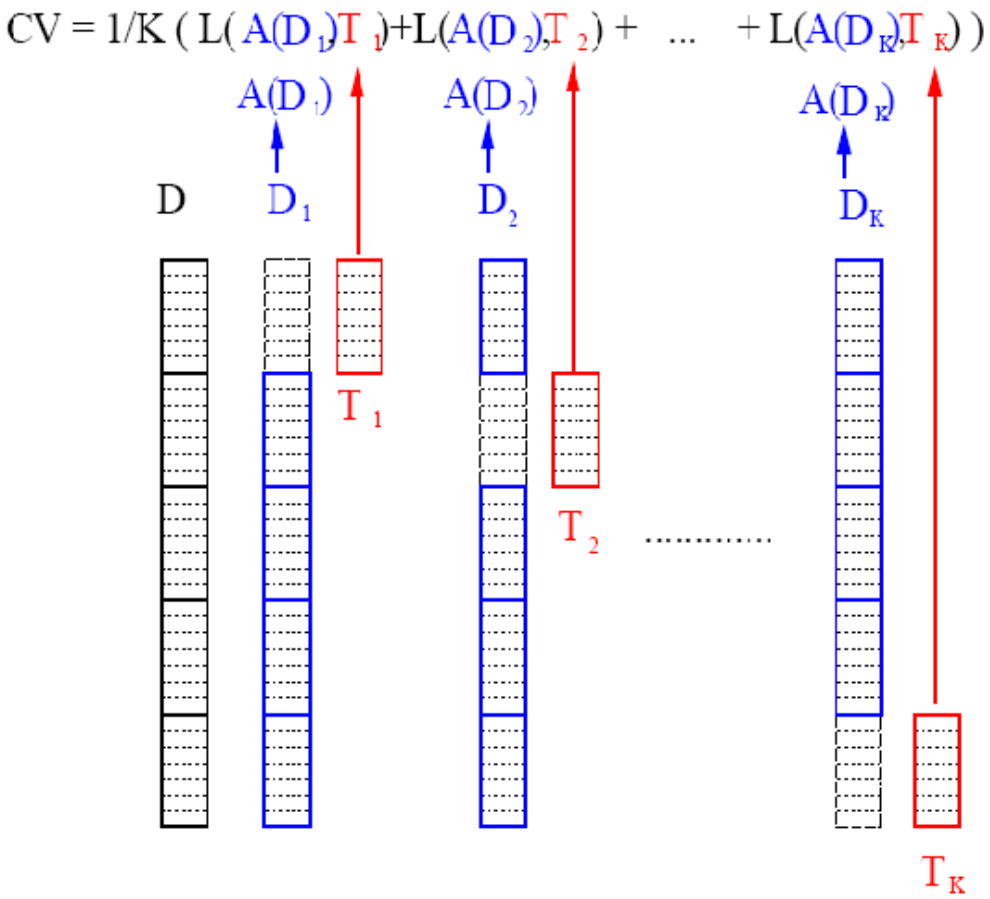

Abbildung 7: Prinzip der j-foold Cross-Validation

4.3.2 Anwendung des Meta - Level - Classifiers MLR

Die Eingabedaten für den Meta-Level-Classifier können nun sowohl Wahrscheinlichkeiten sein, als auch Klassen. Im ersten Fall können wir die lineare Regression wie in der nachstehenden Formel beschrieben einfach berechnet werden.

$$LR_l(x) = \sum_{k}^{K} \alpha_k l P_k l(x)$$

Im zweiten Fall müssen werden die Ergebnisse auf die einfachste denkbare art auf Zahlen abgebildet: Wir ordnen im falle eines treffers dem Ergebnis 1 zu, im Fall einer 'Fahrkarte' 0.

Die alphas wählen wir so, dass der folgende Ausdruck minimal ist:

$$\sum_{j} \sum_{y_n, x_n \in L_j} (y_n - \sum_{k} \alpha_{kl} P_{kl}^{-j}(x_n))$$

Als nächstes berechnen wir die lineare Regression für jeden Klassifikator und wählen für jede Klasse den Klassifikator, der den grössten wert in der linearen

Datasets	# Samples	# Classes	# Attr & Type
Led24	200/5000	10	10N
Waveform	300/5000	3	40C
Horse	368	2	3B+12N+7C
Credit	690	2	4B+5N+6C
Vowel	990	11	10C
Euthyroid	3163	2	18B+7C
Splice	3177	3	60N
Abalone	4177	3	1N+7C
Nettalk(s)	5438	5	7N
Coding	20000	2	15N

N-nominal; B-binary; C-Continuous.

Abbildung 8: Datensätze

Regression hat.

$$LR_l(x) > LR'_l(x) \text{ für alle } l' \neq l$$

4.3.3 Auswertung

Die oben beschriebenen Mechanismen sollen nun an realen Daten getestet werden. Dazu stehen uns die in Abbildung 8 beschriebenen Datensätze zur Verfügung.

Wie in Abbildung 9 zu erkennen ist erzielt unser gestacktes Ensemble exzellente Ergebnisse

4.4 Resume - Stacking

Stacking ist eine der interessantesten Enbsamble-Methoden. Es versucht die vorteile aller bekannten learning algorithms zu vereinen. Dadurch wird aber die Handhabung dieser Methode leicht undhandlich.

Datasets		Level-1 model, $\tilde{\mathcal{M}}$				Level-1 model, $\tilde{\mathcal{M}}'$			
	BestCV	C4.5	NB	IB1	MLR	C4.5	NB	IB1	MLR
Led24	32.8	34.0	32.4	35.0	33.3	41.7	35.7	32.1	**31.3**
Waveform	17.1	17.7	19.2	18.7	17.2	20.6	17.6	17.8	**16.8**
Horse	17.1	16.9	**14.9**	17.6	16.3	18.0	18.5	17.7	15.2
Credit	17.4	18.4	16.1	16.9	17.4	15.4	15.9	**14.3**	16.2
Vowel	2.6	2.6	3.8	3.6	2.6	2.7	7.2	3.3	**2.5**
Euthyroid	**1.9**	**1.9**	**1.9**	**1.9**	**1.9**	2.2	4.3	2.0	**1.9**
Splice	4.5	3.9	3.9	**3.8**	**3.8**	4.0	3.9	**3.8**	**3.8**
Abalone	40.1	38.5	38.5	38.2	38.1	43.3	**37.1**	39.2	38.3
Nettalk(s)	12.7	12.4	11.9	12.4	12.6	14.0	14.6	12.0	**11.5**
Coding	25.0	23.2	23.1	23.2	23.2	22.3	21.2	21.2	**20.7**

Abbildung 9: Auswertung

5 Bibliographie

Elements of Statistical Learning,
Trevor Hastie, Robert Tibshirani, Jerome Friedman,
Springer 2001.

Data Mining: Practical Machine Learning Tools and Techniques with JAVA Implementations,
Ian H. Witten, Eibe Frank, Morgan Kaufmann,
2000

An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants.
Eric Bauer and Ron Kohavi,
Machine Learning, 36(1/2), 105-139

Issues in Stacked Generalization
Kai Ming Ting, Ian H. Witten
1998

An Introduction to Classification and Regression Tree (CART) Analysis
Roger J. Lewis, M.D., Ph.D.,
2000

Rick Higgs und Dave Cummins
Technical Report,
Lilly Research Laboratories, 2003

Tibshirani und Friedman
Elements of Statistical Learning (c) Hastie
2001

Yaochu Jin Future
Technology Research
Honda R and D Europe (Germany), 2000

Yoav Freund and Robert E. Schapire,
Experiments with a New Boosting Algorithm,
AT and T Laboratories 1996

Bernhard Pfahringer
Winning the KDD99 Classification Cup: Bagged Boosting
Austrian Research Institute for AI