VERWENDUNG von ADABOOST in REGELUNGSTECHNISCHEN PROZESSEN

Ronja Güldenring

Hochschule für Angewandte Wissenschaften, Hamburg, Germany,
ronja.gueldenring@haw-hamburg.de

Zusammenfassung. In dieser Recherchearbeit wird AdaBoost auf seinen Einsatz in regelungstechnischen Prozessen untersucht, in denen Echtzeit-Verhalten eine wichtige Rolle spielt. AdaBoost ist ein Klassifizierungsalgorithmus, der eine Menge von schwachen Lernern zu einem starken Klassifikator kombiniert. Aufgrund der unabhängig voneinander rechnenden schwachen Lernern lässt sich AdaBoost gut parallelisieren. Der Einsatz von AdaBoost in Kombination mit der Zustandsschätzung des zu beobachtenden Systems in einem Regelkreis wird als realisierbar eingeschätzt.

1 Einleitung

Die Motivation für die Themenwahl findet ihren Ursprung im Bereich der Robotik. In den letzten Jahren ist die Bedeutung von Servicerobotern/mobilen Robotern sowohl in der Industrie als auch im Dienstleistungsbereich stark gestiegen. Die lang erprobten Industrieroboter befinden sich in einem klar abgegrenzten Bereich (meistens durch Käfige), in dem automatisierte Abläufe stattfinden. In der Service-Robotik ist die Herausforderung viel größer. Sie bewegen sich in einem weitaus größeren Radius mit regelmäßigem Kontakt zu Lebewesen - insbesondere Menschen. Es gibt keine strukturierten Abläufe, d.h. der Roboter muss einen guten Überblick über seine Umgebung haben und situationsbedingt handeln können. Um die Umgebung einschätzen zu können, sind Sensoren wie 2D/3D-Kameras nicht wegzudenken. Mit Hilfe ihrer Daten kann die Umgebung sehr gut erfasst werden und mittels Objektklassifizierung analysiert werden. Um bei zeit-kritischen Situationen eine ausreichend geringe Reaktionszeit vorweisen zu können, ist Echtzeitverarbeitung notwendig. Oftmals sind die Klassifizierungsalgorithmen sehr zeitintensiv und verschlechtern somit die Reaktionszeit. Aus diesem Grund soll die Verwendung von AdaBoost in zeitkritischen Systemen untersucht werden.
In dieser Ausarbeitung wird - inspiriert durch [13] - die Beispielanwendung eines Roboterarms, der ein keulenartiges Objekt fangen soll, betrachtet. Keulenartige Objekte können Flaschen oder Hämmer sein - durch die längliche Form wird das Fangen erschwert, da diese nur in bestimmten Positionen gegriffen werden können. Der Roboter soll mittels eines Regelkreises gesteuert werden, wobei die Lage des Objektes, die durch Objekterkennung ermittelt werden kann, als Eingangsparameter dient. Eine gute Regelung erfordert geringe Zykluszeiten, sodass

die Klassifizierung des Gegenstandes ausreichend schnell passieren muss. Für das erfolgreiche Fangen des Objektes ist eine vorausschauende Lageschätzung nach Kapitel 2.2 notwendig, damit sich der Roboterarm rechtzeitig in die Fangposition bewegen kann und im richtigen Moment zugreift.

2 Grundlagen

In diesem Kapitel werden Ansätze zur Realisierung der gewählten Beispielanwendung vorgestellt. Der Hauptfokus liegt auf der Einführung des AdaBoost-Algorithmus. Dabei wird seine Funktionsweise und sein Generalisierungsverhalten vorgestellt. Abschließend wird ein Überblick über die Zustandsschätzung gegeben, wobei das Kalman-Filter als möglicher Lösungsansatz näher erläutert wird.

2.1 AdaBoost

AdaBoost ist ein Klassifizierungsalgorithmus. Nach [8] bedeutet Klassifizierung aus einer Menge von Daten Objekte zu identifizieren und in entsprechende Klassen einzusortieren. Dabei werden Merkmalsvektoren für die Klassen aufgestellt und anhand dieser die entsprechenden Objekte herausgefiltert. Die Herausforderung ist es einen Klassifikator zu kreieren, der die Klassen vollständig repräsentiert, z.B. soll sowohl ein Stuhl mit als auch ohne Lehne erfolgreich in die Klasse Stuhl einsortiert werden. Insbesondere in der Bildverarbeitung ist zu berücksichtigen, dass Bilder nicht immer die gleiche Qualität an Informationen bieten. Es ist möglich, dass unscharfe Bilder entstehen oder relevante Objekte durch andere verdeckt werden. Solche Abweichungen sollten im besten Fall keine Auswirkungen auf das Endergebnis haben. Für die Klassifizierung gibt es unterschiedliche Verfahren. Bekannte Verfahren sind neben AdaBoost unter anderem Schwellwert-Klassifikatoren, Support Vector Machines oder neuronale Netze.
In [15] gibt Schapire eine Einführung in AdaBoost, die im Folgenden zusammengefasst wird. Bei AdaBoost handelt es sich um einen lernenden Klassifizierungsalgorithmus, der eine Menge von T schwachen Lernern zu einem starken Klassifikator kombiniert. Im Prinzip wird die Klassifizierungsentscheidung durch Abstimmung getroffen, wobei jeder schwache Lerner eine Stimme für oder gegen die gesuchte Klasse abgibt. Es können sowohl Zwei- als auch Mehrklassenprobleme gelöst werden. In den folgenden Unterkapiteln wird sich der Einfachheit halber auf Zweiklassenprobleme bezogen, d.h. ein Objekt wird der Klasse zugeordnet oder nicht. Schwache Lerner sind so definiert, dass ihre Erfolgswahrscheinlichkeit bei der Klassifizierung nicht exakt der Zufallswahrscheinlichkeit = 0.5 entsprechen darf. Es sollten natürlich nur schwache Lerner einbezogen werden, deren Erfolgswahrscheinlichkeit besser als 0.5 ist, weshalb Ausgänge von Lernern, deren Erfolgswahrscheinlichkeit geringer ist, einfach negiert werden können. Sie sollten ressourcenschonend, also einfach aufgebaut sein. AdaBoost wird gemäß des nachfolgenden beschriebenen Trainingsprozesses iterativ geschult und anschließend auf seinen Trainingserfolg getestet.

Trainingsprozess Nach [15] wird AdaBoost - der starke Klassifikator - iterativ trainiert. Gegeben sei eine Menge von T schwachen Lernern und ein möglichst repräsentativer Datensatz $(x_1, y_1), ..., (x_m, y_m)$, wobei $x_i \in X$ die Eingänge und $y_i \in Y = \{-1; 1\}$ die Ausgänge sind. In jedem Iterationsschritt wird der schwache Lerner mit dem kleinsten Klassifizierungsfehler ausgewählt. Es wird eine Gewichtung α_t für den gewählten Lerner berechnet, die angibt mit welchem Anteil dieser am Ende in den starken Klassifikator eingeht. Außerdem wird der Datensatz D neu gewichtet. Dabei werden Gewichtungen falsch klassifizierter Daten hoch gesetzt, sodass diese Daten in den folgenden Iterationsschritten stärker berücksichtigt werden. Aus dieser Anpassung an die Daten entspringt der Name AdaBoost, der eine Kurzform für „adaptive Boosting“ ist. Der Pseudo-Code ist in Algorithmus 1 angegeben.

Gegeben: $(x_1, y_1), ..., (x_m, y_m)$, T schwache Lerner;
Gewichtung des Datensatzes initialisieren: $D_1(i) = 1/m$;
for *t = 1,..., T* **do**

1. Wende schwache Lerner auf Datensatz D_t an;
2. Wähle den schwachen Lerner h_t mit kleinsten Klassifizierungsfehler ε_t:

$$\varepsilon_t = P_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_t) \neq y_i} D_t(i);$$

3. Bestimme Gewichtung für den gewählten schwachen Lerner:

$$\alpha_t = \frac{1}{2} ln(\frac{1 - \varepsilon_t}{\varepsilon_t});$$

4. Passe Gewichtung des Datesatzes an, wobei Z_t der Normalisierungsfaktor ist:

$$D_{t+1} = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & IF\ h_t(x_i) = y_i \\ e^{\alpha_t} & IF\ h_t(x_i) \neq y_i \end{cases};$$

end
return starker Klassifikator $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$;

Algorithm 1: AdaBoost-Algorithmus aus [15]

Zu ergänzen ist, dass die schwachen Lerner nicht unbedingt vor der Durchführung des Algorithmus definiert werden müssen. Oft werden Schwellwert-Klassifikatoren verwendet, die in den einzelnen Iterationsschritten auf Basis des gewichteten Datensatzes erst generiert werden. Schwellwert-Klassifikatoren sind Binärbäume mit einer Höhe von Eins. Je nachdem, ob der Schwellwert unter- oder überschritten wird, werden die Daten in Klasse A (+1) oder Klasse B (-1) eingeordnet (siehe [8]).
In [5] beweisen Freund und Schapire, dass der Trainingsfehler ε exponentiell mit

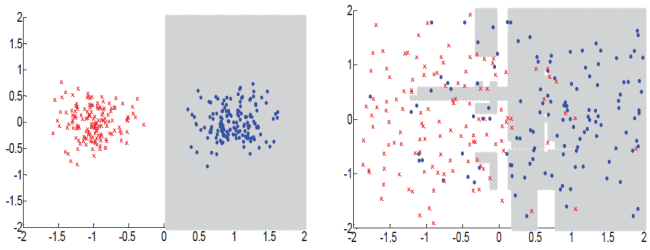
der Anzahl der schwachen Klassifikatoren sinkt.

$$\varepsilon = 0.5 - \gamma_t$$

$$\varepsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t(1-\varepsilon_t)} \leq exp(-2\sum_{t=1}^{T} \gamma_t^2)$$

Generalisierungsfehler Der Generalisierungsfehler wird nach dem Trainingsprozess bestimmt. Der starke Klassifikator wird getestet, indem dieser einen anderen, möglichst repräsentativen Datensatz klassifiziert. Es wird bestimmt in wie vielen Fällen die Daten falsch klassifiziert wurden. Somit ergibt sich der Generalisierungsfehler aus dem Verhältnis der falsch klassifizierten Daten zu der Gesamtanzahl der Daten.
Im Bereich des machinellen Lernens spielt Overfitting bzw. Überanpassung eine große Rolle. Mit Überanpassung beschreibt man einen lernenden Algorithmus, der sich an seine Trainingsdaten anpasst, sich somit zu sehr spezialisiert und nicht mehr allgemeingültig auf andere Daten anwendbar ist.
Overfitting ist bei AdaBoost ein stark diskutiertes Thema. Es gibt diverse Ansätze, wann AdaBoost der Überanpassung unterliegt. Eine endgültige Aussage lässt sich diesbezüglich nicht machen. In den folgenden Absätzen werden die wichtigsten Ansätze vorgestellt.
Nach der Einführung des AdaBoost-Algorithmus von Freund und Schapire haben Wissenschaftler durch Experimente die Erfahrung gemacht, dass für AdaBoost bei der Erhöhung der Anzahl der schwachen Lernern keine Überanpassung auftritt. Insbesondere ist hier die Veröffentlichung [2] von Breimann zu erwähnen; in seinem Versuch sinkt der Generalisierungsfehler mit zunehmender Anzahl an schwachen Klassifikatoren, obwohl der Trainingsfehler bereits bei 0 liegt.
Nach diesen Ergebnissen stellte Schapire und andere [16] die Margin-Theorie auf, die das Generalisierungsverhalten von AdaBoost besser vorhersagen soll. Dabei wird zur Abschätzung der Güte des Algorithmus nicht nur der Trainingsfehler in Betracht gezogen, sondern ein weiterer Wert, der Margin genannt wird. Der Margin besagt mit welcher Sicherheit der Algorithmus die richtige Entscheidung getroffen hat. Der Margin ist also groß, wenn die Entscheidung auf der Grundlage einer großen Mehrheit an schwachen Klassifikatoren getroffen wurde. Wurden die Daten nur mit einer knappen Mehrheit richtig klassifiziert, ist der Margin klein. Diese Theorie ist die weitverbreiteste Theorie um das Überanpassungsverhalten von AdaBoost zu erklären - wurde jedoch bereits in anderen Versuchen widerlegt (siehe [1]).
Des weiteren wurden noch zusätzliche Erkenntnisse bezüglich der Überanpassung von AdaBoost gemacht. Diettrich zeigte in [4], dass AdaBoost sehr viel schlechter abschneidet, sobald reale Datensätze, in denen Rauschen vorhanden ist, vorliegen. Mit Rauschen sind vor allem falsch gekennzeichnete Daten Y in den Datensätzen gemeint. Beispielsweise wird ein Bild, auf dem sich eine Katze befindet, mit der Klasse „Nicht-Katze“ gekennzeichnet. Dadurch wird das Bild im Trainingsprozess zunächst korrekterweise in die Klasse Katze eingeordnet.

Aufgrund der falschen Kennzeichnung, handelt es sich im Trainingsprozess um ein falsch klassifiziertes Bild, sodass die Gewichtung des Bildes hoch gesetzt wird. AdaBoost fokussiert sich also stärker auf dieses Bild, sodass es am Ende falsch „gelernt“ wird. In [17] wurde festgestellt, dass AdaBoost bei sich überschneidenden Klassen für Überanpassung anfällig ist. Überschneidende Klassen sind so definiert, dass sowohl Klasse A als auch Klasse B mit einer positiven Wahrscheinlichkeit in Frage kommen. Zur Verdeutlichung ist in Abbildung 1 ein Datensatz mit überschneidenden Klassen im Vergleich zu klar getrennten Klassen zu sehen.



(a) klar getrennte Klassen (b) überschneidende Klassen

Abb. 1: Veranschaulichung von überschneidenden Klassen im zweidimensionalen Raum aus [11]

2.2 Modellbasierte Zustandsschätzung

Bei der modellbasierten Zustandsschätzung wird der tatsächliche Zustand für ein System oder Objekt auf der Basis von Messwerten geschätzt. Eine Schätzung ist notwendig, da entweder die Messdaten ungenau sind oder eine frühzeitige Vorhersage des Zustandes notwendig ist. Für die Zustandsschätzung wird eine mathematische Beschreibung, das sogenannte Modell, des physikalischen Systems benötigt. Zusätzlich sollten noch Mess- und Systemrauschen berücksichtigt werden. Mit Messrauschen sind ungenaue Sensoren gemeint, dessen Messwerte in einem Intervall um den tatsächlichen Wert schwanken. Systemrauschen beschreibt äußere Einwirkungen im physikalischen System, die im Modell nicht berücksichtigt werden wie z.B. Reibung. In dem Anwendungsbeispiel eines Roboterarms, der einen keulenartigen Gegenstand fangen soll, ist das physikalische System der fliegende Gegenstand. Dieses System lässt sich im Modell mit einer idealen Wurfparabel annähern. Systemrauschen wie Luftwiderstand und -strömung werden in dem Modell vernachlässigt. Als Messeinrichtung würde eine Kamera dienen - auf den Bildern würde der Gegenstand mittels AdaBoost klassifiziert werden,

sodass Messdaten wie Position, Verdrehung und Ausdehnung des Objektes ermittelt werden können.
Das Kalman-Filter ist sehr beliebt im Bereich der Verfolgung von sich bewegenden Objekten. Im Folgenden wird dieses kurz in Anlehnung an [12] erläutert. Das Filter funktioniert so, dass zunächst eine a-priori-Schätzung $\hat{x}_{k+1}^{-}$ für den Zustand gemacht wird. Sobald die benötigten Messwerte vorliegen, wird die Prädiktion mit den gemessenen Werte verglichen und ein korrigierter a-posteriori Systemzustand $\hat{x}_{k+1}^{+}$ geschätzt. Auf Grundlage der neuen Messwerte wird dann ein neuer a-priori-Schätzwert für den nächsten Zustand vorhergesagt und die Iterationsschleife wird erneut durchlaufen. Der Startschätzwert ist sehr ungenau, sodass es zunächst mehrere Iterationsschritte benötigt bis die Prädiktion des Zustandes und der tatsächliche Zustand sehr nah beieinander liegen. Bei den a-priori und a-posteriori-Schätzungen werden jeweils Schätzfehlerkovarianzen P_{k+1}^{-}, P_{k+1}^{+} berechnet. Diese geben die Sicherheit der Schätzung an, d.h. je kleiner der Schätzfehler P desto genauer die Schätzung.

3 Verwandte wissenschaftliche Arbeiten

Anforderungen für einen regelungstechnischen Prozess sind schnelle Zykluszeiten sowie Genauigkeit und Robustheit der Messwerte. Es ist meistens schwer alle Kriterien optimal zu erfüllen, da sie sich gegenseitig behindern. Zum Beispiel sind sehr genaue Klassifizerungsalgorithmen oftmals langsamer, da aufwändige und viele Merkmale verwendet werden. Aus diesem Grund ist es wichtig einen zufriedenstellenden Kompromiss zwischen Geschwindigkeit und Genauigkeit zu finden. In den folgenden Unterkapiteln werden die Bereiche bezüglich AdaBoost genauer untersucht und entsprechende wissenschaftliche Arbeiten vorgestellt.

3.1 Hardware-technische Beschleunigung

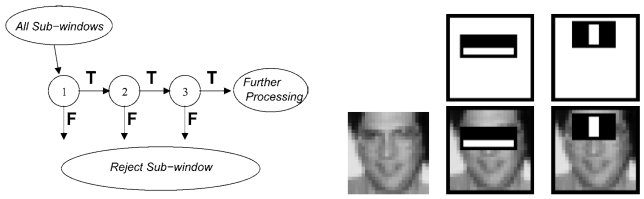
Die Geschwindigkeit, mit der ein Regelkreis durchlaufen werden kann, ist entscheidend für die Regelperformance. Werden komplexe Sensoren wie Kameras in den Regelprozess eingebunden, müssen ihre Daten über Bildverarbeitungsalgorithmen zunächst ausgewertet werden um interessante Messdaten zu extrahieren. Es vergeht also eine gewisse Zeit von Aufnahme des Zustandes bis zur Extraktion der Messdaten. In dem Fall des zu fangenden Objektes muss das Objekt auf der Bildaufnahme zunächst klassifiziert werden, woraus sich Lage und Rotation ermitteln lassen. Diese Daten werden dann in den Regelkreis eingespeist und bei der Lageschätzung verarbeitet. Ziel ist es die Zeit für die Auswertung der Bilddaten zu minimieren, wobei die Klassifizierung einen großen Anteil ausmacht. Für die Klassifizierung wird AdaBoost vorgeschlagen, da sich dieser Algorithmus sehr gut parallelisieren lässt. Er besteht aus einer Menge von unabhängig arbeitenden schwachen Lerner, die alle parallel arbeiten können und deren Endergebnisse am Ende in einer Operation verknüpft werden. Für die Parallelisierung kommen im Moment hardware-technische Lösungen wie Grafikprozessoren(GPU) oder Field

Programmable Array (FPGA) in Frage. Allerdings verfügen auch Zentraleinheiten (CPU) bereits in den meisten Fällen über zwei bis vier Kerne. Die Anzahl der Kerne in CPUs ist in den letzten Jahren stetig gestiegen und der Trend geht weiter in diese Richtung.
In [7] wurde der bekannte AdaBoost-Gesichts-Klassifikator von Viola & Jones, der in Kapitel 3.2 vorgestellt wird, mittels NVIDIA Tesla C1060 GPUs parallelisiert. Dabei werden alle Erkennungsfenster eines Bildes parallel von Threads verarbeitet. Erkennungsfenster sind Rechtecke, die über das Bild geschoben werden; jeder Bereich des Bildes wird in die definierten Klassen eingeordnet. Für Gesichter werden beispielsweise quadratische Erkennungsfenster verschiedener Größen definiert. Die Verarbeitung von kleinen Erkennungsfenstern dauert länger als die von Großen, da sich bei großen Erkennungsfenstern das zu untersuchende Bild in weniger Unterbereiche teilt. Insgesamt wurden vier GPUs eingesetzt. Jede GPU hat eine oder mehrere Erkennungsfenstergrößen bearbeitet. Die Erkennungsfenstergrößen wurden so verteilt, dass ungefähr alle GPUs eine ähnliche Rechenzeit benötigen. Bei den Ergebnissen ergab sich eine Geschwindigkeit von 15.2 Frames per Second. Die Geschwindigkeit entspricht ca. einer FPGA-Geschwindigkeit und ist 50 mal schneller als auf einem gewöhlichem Prozessor. Als Nachteil wird in dieser Veröffentlichung der Energieverbrauch von 190Watt pro GPU aufgeführt, was vor allem Probleme bei mobilen Anwendungen macht. Dazu muss erwähnt werden, dass sich die Grafikprozessoren seit 2010 weiterentwickelt haben. 2014 kam der energieeffiziente Mobilprozessor NVIDIA Tegra K1 mit einer maximalen Leistungsaufnahme von 11 Watt auf den Markt. Er ist mit der leistungsstarken NVIDIA-Kepler-GPU ausgestattet und unterstützt die weitverbreitete GPU-Programmiersprache CUDA.
In [10] wird der Algorithmus durch Verwendung einer FPGA beschleunigt. Der AdaBoost-Algorithmus wird in eine pre-processing und eine post-processing Phase aufgeteilt. Die pre-processing Phase ist mit einem Anteil an schwachen Lernern auf der FPGA implementiert, da diese nur für eine gewisse Anzahl ausgelegt ist. In jedem Taktzyklus kann ein Erkennungsfenster durch die parallel arbeitenden schwachen Lerner in die zugehörige Klasse eingeordnet werden. Die restlichen schwachen Lerner befinden sich auf einer CPU und untersuchen die in der ersten Phase extrahierten Regionen genauer. Durch das Auslagern eines Teil des Klassifikators auf eine FPGA verkürzt sich sowohl die Rechenzeit als auch der Energieverbrauch. Eine FPGA schafft es 22152 mal mehr Erkennungsfenster/Watt und 40 mal mehr Erekennungsfenster/sec als eine CPU zu überprüfen. Die enormen Energieeinsparungen sind insbesondere bei mobilen Anwendungen von Bedeutung.

3.2 Software-technische Beschleunigung

Neben den hardwaretechnischen Lösungen sind auch softwaretechnische Beschleunigungen bei AdaBoost möglich. Dabei wurden verschiedene Ansätze verfolgt: die Wahl der schwachen Lerner, die Kombination verschiedener Methoden oder die Abwandlung des ursprüngliche AdaBoost-Algorithmus.

Eine sehr bekannte softwaretechnische Optimierung ist der kaskadierte AdaBoost-Algorithmus unter der Verwendung von Haar-like-Features von Viola & Jones aus [18]. Zu klassifizierende Daten müssen einen entartenden binären Suchbaum (Kaskade), wie in Abbildung 2a zu sehen ist, durchlaufen. Jedes Blatt stellt einen eigenständigen AdaBoost-Klassifikator dar, wobei die Komplexität des Klassifikators von links nach rechts zunimmt. Wird ein Bild in einem Teilklassifikator als positiv klassifiziert, wird dieses an den folgenden Klassifikator weitergereicht. Das Bild ist endgültig positiv bzw. negativ klassifiziert, wenn dieses die komplette Kaskade durchläuft bzw. an einem der Teilklassifikator aussortiert wird. Die Kaskade bewirkt eine deutliche Verringerung der Rechenzeit, da einfach identifizierbare Negativ-Bilder frühzeitig aussortiert werden und nicht den kompletten Klassifizerungsprozess durchlaufen müssen. Zusätzlich bewirkt die Verwendung von Haar-like Merkmalen eine deutliche Beschleunigung, da es sich hier um sehr einfache und effektive Merkmale handelt. In den Erkennungsfenstern werden 2-4 weiße oder graue Rechtecke definiert, wobei die Anzahl der Pixel in den weißen Rechtecken von der Pixelanzahl der grauen Rechtecken subtrahiert wird. Ein anschauliches Beispiel ist in Abbildung 2b gegeben. Je nach dem, ob das Ergebnis für einen bestimmten Bildbereich über oder unter einem definierten Grenzwert liegt, wird der Bereich in Klasse 1 oder -1 eingeordnet. Um die mehrmalige Berechnung von der Pixelanzahl zu vermeiden, wird das Bild einmalig in ein sogenanntes Integral-Bild umgewandelt. Im Integral-Bild wird für jede Stelle x, y des Bildes die Summe der Pixel, die sich links und über der definierten Stellen befinden, berechnet. Aus den Informationen lassen sich schließlich in einer Operation und über vier Arrayreferenzen Summen für die in den Haar-like Merkmalen definierten Rechtecke berechnen.



(a) Schematische Darstellung des kaskadierten AdaBoost

(b) Haar-like Merkmale zur Identifikation von Augenpartien

Abb. 2: kaskadierter AdaBoost mit Haar-like Merkmalen aus [18]

In [9] wurde im Bereich der Gesichtserkennung durch die Kombination zweier Algorithmen eine Beschleunigung erreicht. Dabei wurden die Skin-Color-Feature-Methode mit AdaBoost unter der Verwendung der Haar-Features als schwache Lerner kombiniert. Bei der Skin-Color-Feature-Methode werden über die Farbe

der Pixel Bereiche mit Hauttönen identifiziert. Sie hat im Gegensatz zu AdaBoost eine geringere Genauigkeit, benötigt allerdings nur die Hälfte der Rechenzeit. Zunächst werden mehrere Qualitätsstufen des zu klassifzierenden Bildes erzeugt. Im ersten Klassifizierungs-Schritt wird die Skin-Color-Feature-Methode auf die gering auflösenden Bilder angewendet, bis Regionen erkannt werden, in denen Farbpixel von Hauttönen vorkommen. Im zweiten Schritt überprüft AdaBoost in den qualitativ besseren Bildern nur die vorher identifizierten Regionen und klassifiziert diese als wahr oder falsch. Durch die Vorfilterung der Gesichter mittels Skin-Color-Feature-Methode ergeben sich bei den Experimenten deutlich bessere Geschwindigkeiten, wobei eine minimale Reduzierung der Erkennungsrate von ca. 3% in Kauf genommen wird.

In dem Fall des fliegenden Objektes bietet sich ebenfalls eine anwendungsspezifische Beschleunigung der Objektklassifizierung an. Sobald das zu fangende Objekt einmal klassifiziert wurde, kann davon ausgegangen werden, dass das Objekt von Bild zu Bild seinen Ort nur um ein maximales Delta ändert. Es bietet sich also an im AdaBoost-Algorithmus zunächst die Erkennungsfenster im Bereich um das zuvor klassifizierte Objekt zu untersuchen. Auch die Größe der Erkennungsfenster ändert sich von dem einen Bild zum Anderen nicht schlagartig. So kann man sich ebenso nach der Größe des zuvor klassifizierten Objektes richten. Inwiefern sich diese Idee realisieren lässt und ob es tatsächlich eine Beschleunigung der Klassifizerung mit sich bringt, sollte im weiteren Projektverlauf geprüft werden.

3.3 Genauigkeit

Für das erfolgreiche Fangen des Gegenstandes spielt auch die Genauigkeit von AdaBoost und daraus folgende Lagebestimmung eine entscheidende Rolle. Die Verbesserung des Generalisierungsfehlers ist seit der Einführung von AdaBoost bis heute ein Thema zu dem viel Forschung betrieben wird und dementsprechend viele Veröffentlichungen resultieren. Die folgenden Absätze sollen einen Überblick geben.
Es wurden diverse Varianten von AdaBoost entwickelt, die in speziellen Fällen eine bessere Genauigkeit erzielen. Beispielhaft wird hier Gentle AdaBoost aus [6] vorgestellt. Die schwachen Lerner geben bei Gentle AdaBoost kontinuierliche Werte zurück, die im Intervall von [-1, 1] liegen. Dieser kontinuierliche Wert gibt an, mit welcher Wahrscheinlichkeit das Datum von dem schwachen Klassifikator in die richtige Klasse einsortiert wurde. Die Entscheidungssicherheit der schwachen Klassifikatoren wird also berücksichtigt, was einen behutsamen (engl. gentle) Einfluss auf die Gewichtung der Trainingsdaten in jeder Runde t hat. Ausreißer werden folglich bei der Gewichtung nicht so stark berücksichtigt, was den Datensatz indirekt glättet.
Die Wahl der schwachen Lerner kann für das Generalisierungsverhalten ebenso ausschlaggebend sein. Insbesondere ist diese sehr anwendungsspezifisch, sodass sich dafür keine allgemeingültige Regel formulieren lässt. Fakt ist, dass man

durch die Wahl der schwachen Lerner einen großen Einfluss auf den resultierenden starken Klassifikator hat. Als Beispiel ist hier die Geschlechtserkennung in [14] zu erwähnen. Für die Klassifizierung wurden hier spezielle Merkmale wie Haarlänge und -kontur, Bartmerkmale und Merkmale in Gesichtszügen verwendet.
Weitere Ansätze zur Verbesserung des Generalisierungsfehlers sind die Kombination verschiedener Algorithmen in [3] oder die Anpassung des Trainingsdatensatzes in [11].

4 Ausblick

Der weitere Projektverlauf gliedert sich in drei Phasen: Projekt 1, Projekt 2/Hauptseminar und die Masterarbeit.
In Projekt 1, was für das kommende Semester angesetzt ist, geht es zunächst darum sich aktiv mit den Grundlagen der Materie auseinander zu setzen. Dafür soll zunächst ein einfacher Zweiklassen-AdaBoost-Klassifikator implementiert, trainiert und getestet werden. Für die Realisierung müssen vorerst die Rahmenbedingungen festgelegt werden. Es stellt sich einerseits die Frage der Datenbeschaffung und andererseits die Wahl der Entwicklungsumgebung.
Für die Datenbeschaffung gibt es mehrere Online-Datenbanken, die Bilder zu verschiedensten Themenbereichen zur Verfügung stellen. Bekannte Datenbanken sind ImageNet, UCI-Repository und PASCAL VOC. Für Keulenartige Gegenstände gibt es jede Menge Datensätze; es kommen zum Beispiel Wasserflaschen oder Hämmer in Frage.
Als Entwicklungsumgebung bietet sich auf den ersten Blick Matlab an. Die Matlab-Lizenz der HAW Hamburg beinhaltet die Statistics and Machine Learning Toolbox, die AdaBoost und diverse Varianten unterstützt. Zusätzlich ist Matlab/Simulink ein beliebtes Werkzeug im Bereich der Regelungstechnik. Es können Regelkreise mittels Blockschaltbildern einfach aufgebaut und simuliert werden. Auch das Kalman-Filter steht zur Verfügung. Matlab scheint für den Einstieg auf jeden Fall passend, es bleibt allerdings die Frage inwieweit man durch vorgefertigte Funktionen eingeschränkt ist.
Für die gewählte Objektklasse sollten mehrere schwache Lerner getestet werden um ein Optimum an Geschwindigkeit und Genauigkeit zu erreichen. Für die Parallelisierung von AdaBoost würde CUDA, eine von NVIDIA entwickelte Programmmiersprache für Grafikprozessoren, die auf C basiert, in Frage kommen. CUDA bietet auch Anbindungen an Matlab, sodass hoffentlich an die Vorarbeiten angeknüpft werden kann.
In Projekt 2 soll der AdaBoost-Algorithmus zunächst mit Video-Daten getestet werden und auf fliegende Objekte spezialisiert werden. Dabei soll der aus Kapitel 3.2 definierte Ansatz untersucht werden. Anschließend werden AdaBoost, die Parallelisierung und die modellbasierte Zustandsschätzung miteinander verknüpft. Ziel ist es geworfene Objekte mittels Kamera zu detektieren und Vorhersagen zu treffen. Am Ende soll nachgewiesen oder widerlegt werden, dass die Vorhersagen ausreichend genau und schnell getroffen wurden. Das fernere Ziel ist

es, das Anwendungsbeispiel an einem realen Roboterarm zu realisieren. Eventuell ist eine Kooperation mit dem Zentrum für industrielle Robotik aus dem Departement Maschinenbau und Produktion möglich. Sie verfügen über ausreichend schnelle Roboterarme wie universal Robots oder Kuka Roboter.
Während der Masterarbeit werden Fragestellungen, die sich im Laufe der Projekte ergeben haben, untersucht, wobei bereits gewonnene Kenntnisse ebenso mit einfließen. Eventuell ist ein Vergleich von AdaBoost mit einem anderen geeigneten Klassifikator möglich. Es könnten auch weitere Sensoren in dem Rahmen untersucht werden.

Literatur

1. Breiman, L.: Prediction games and arcing algorithms. Neural Comput. 11(7), 1493–1517 (Oct 1999), http://dx.doi.org/10.1162/089976699300016106
2. Breimann, L.: Arcing classifiers. In: The Annals of Statistics. University of California, Berkely, USA (1998)
3. Chen, J., Ariki, Y., Takiguchi, T.: Robust facial expressions recognition using 3d average face and ameliorated adaboost. In: Proceedings of the 21st ACM International Conference on Multimedia. pp. 661–664. MM '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2502081.2502173
4. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning 40(2), 139–157 (2000), http://dx.doi.org/10.1023/A:1007607513941
5. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55(1), 119–139 (Aug 1997), http://dx.doi.org/10.1006/jcss.1997.1504
6. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Annals of Statistics 28, 2000 (1998)
7. Hefenbrock, D., Oberg, J., Thanh, N., Kastner, R., Baden, S.: Accelerating viola-jones face detection to fpga-level using gpus. In: Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on. pp. 11–18 (May 2010)
8. Herbert Süße, E.R.: Bildverarbeitung und Objekterkennung. Springer Verlag (2014)
9. Ji, S., Lu, X., Xu, Q.: A fast face detection method combining skin color feature and adaboost. In: Multisensor Fusion and Information Integration for Intelligent Systems (MFI), 2014 International Conference on. pp. 1–5 (Sept 2014)
10. Kadlček, F., Fučík, O.: Fast and energy efficient adaboost classifier. In: Proceedings of the 10th FPGAworld Conference. pp. 2:1–2:5. FPGAworld '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2513683.2513685
11. Kim, D.S., Baek, Y.M., Kim, W.Y.: Reducing overfitting of adaboost by clustering-based pruning of hard examples. In: Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication. pp. 90:1–90:3. ICUIMC '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2448556.2448646
12. Kim, P.: Kalman Filter for Beginners. A-JIN Publishing Company (2010)
13. Kim, S., Shukla, A., Billard, A.: Catching objects in flight. IEEE Transactions on Robotics 30(5), 1049–1065 (Oct 2014)

14. Lee, C.C., Wei, C.S.: Gender recognition based on combining facial and hair features. In: Proceedings of International Conference on Advances in Mobile Computing & Multimedia. pp. 537:537–537:540. MoMM '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2536853.2536933
15. Schapire, R.E.: A brief introduction to boosting. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2. pp. 1401–1406. IJCAI'99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999), http://dl.acm.org/citation.cfm?id=1624312.1624417
16. Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods (1997)
17. Vezhnevets, A., Barinova, O.: Avoiding boosting overfitting by removing 'confusing samples'. In: European Conference on Machine Learning ECML. pp. 430–441. Springer, Springer (2007), http://graphics.cs.msu.ru/en/publications/text/ecml2007vb.pdf
18. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Accepted Conference on Computer Vision and Patter Recognition 2001. pp. 511–518 (2001)