Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Ausarbeitung

Online Machine Learning

vorgelegt von

Tim Pietz

Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen

Studiengang:	Informatik
Matrikelnummer:	6808046
Betreuer:	Eugen Betke

Hamburg, 2018-02-21

Inhaltsverzeichnis

1 Einleitung

Die Anwendungsgebiete des Machine Learning lassen sich in zwei Klassen unterteilen, in Offline und Online Learning.

Bei dem klassischen Offline-, oder auch Batch-Learning, wird das Modell in einer Trainingsphase gebildet, bevor es im Anschluss verwendet werden kann. Die Trainingsphase setzt hierbei daran an, dass ein repräsentativer Datensatz existiert, welcher für die Modellbildung geeignet ist. Vereinfachend wird dieser in drei Mengen partitioniert, welche als Trainings-, Validierungs- und Testdaten bezeichnet werden, und in einem Größenverhältnis von zum Beispiel 50/25/25 stehen könnten. Der Lernalgorithmus iteriert hierbei meist mehrfach über alle Datenpunkte des Trainingsdatensatzes, bevor die Modelle anhand des Validierungsdatensatzes bewertet, das beste zur Verwendung ausgewählt, und eine letzte Abschlussbewertung anhand des Testdatensatzes vorgenommen wird. [Kun17]

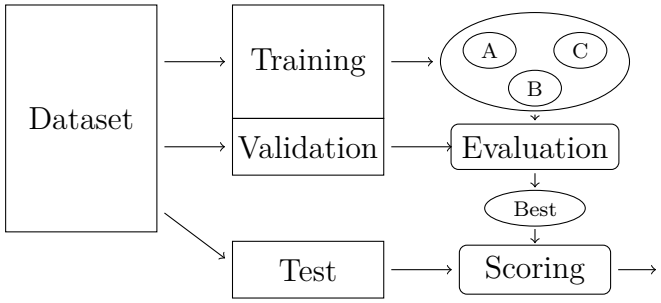

Abbildung 1.1: Schematische Darstellung des Offline Learning Verfahrens
Vgl. [Kun17]

Mit immer steigenden Mengen an kontinuierlich generierten Daten eignet sich dieser Ansatz des Offline Learnings in vielen Anwendungsgebieten jedoch nur bedingt. Um Nutzen aus den durchgängig hinzukommenden Trainingsdaten ziehen zu können, würde dies erfordern, auf dem erweiterten Trainingsdatensatz ein komplett neues Modell zu trainieren. Dies stellt einen wiederholt signifikanten Rechenaufwand dar, und erfordert desweiteren die Speicherung und effiziente Bereitstellung des ständig wachsenden Datensatzes.

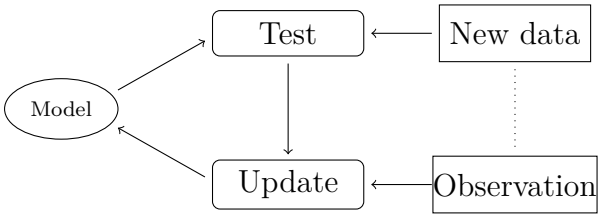

Abbildung 1.2: Schematische Darstellung des Online Learning Verfahrens

Im Gegensatz dazu gibt es bei Online Learning Algorithmen keine klar getrennten Phasen der Modellbildung und -nutzung. Stattdessen liegt ein erweiterbares Modell zugrunde, welches sequentiell auf einzelnen Datenpunkten trainiert werden kann, anstatt in einem Block auf einen vollständigen Datensatz ausgerichtet zu werden. Ein solches Modell kann analog angewendet werden, bietet jedoch zusätzlich die Möglichkeit anhand von neuen Datenpunkten erweitert zu werden. Dies ermöglicht einen kontinuierlichen Wechsel zwischen Nutzung und Training, was die verallgemeinerte Datenstromorientierte Darstellung des Online Learnings aus Listing 4.2 motiviert.

```
1 Initialize model
2
3 Monitor stream:
4     Receive input x
5     Predict p
6     Receive true answer y
7     Update model (x, p, y)
```

Listing 1.1: Datenstromorientierte Darstellung von Online Learning Algorithmen

Um einen Überblick über den Bereich des Online Machine Learnings zu verschaffen, werden im Kommenden zuerst mögliche Anwendungen und Herausforderungen erläutert, bevor im Anschluss ausgewählte Algorithmen betrachtet und verglichen werden.

2 Anwendungsgebiete

Die Charakteristik der effizienten Erweiterbarkeit von Modellen des Online Learnings ermöglicht den Einsatz in breiten Anwendungsgebieten, von denen im Folgenden drei eingeleitet werden. In Kapitel 3 werden die damit einhergehenden Herausforderungen genauer diskutiert.

2.1 Streaming Data

Das grundlegende Konzept der Online Learning Algorithmen sieht die Anwendung in den Fällen vor, in denen die Trainingsdaten zu Beginn noch nicht zugreifbar sind, ggf. noch gar nicht existieren. Dieser Fall tritt unter anderem häufig im Kontext von Nutzergenerierten oder Nutzerorientierten Daten bzw. Zielfunktionen auf. Ein Beispiel, welches in Abschnitt 4.4 genauer behandelt wird, ist das der (personenspezifischen) Handschriftenerkennung. Bei der Nutzung eines solchen Systems enstehen durch die fortlaufende Verwendung und anschließende Korrektur der erkannten Zeichen durch den Nutzer auf natürliche Weise kontinuierlich neue Trainingsdaten, welche zur Verbesserung des vorliegenden Modells verwendet werden können.

2.2 Big Data

Besonders im Hinblick auf sehr große Datenmengen bietet sich die Nutzung von Online Learning Algorithmen an. Da diese auf einzelnen Datenpunkten anstatt ganzen Datensätzen trainiert werden, ist es nicht notwendig, den gesamten Datensatz effizient zugreifbar (d.h. im Arbeitsspeicher) zu halten.

Das Zusammenspiel der Konzepte Big Data und Streaming Data ruft teils das Problem hervor, dass die Datenmenge nicht nur zu groß ist, um effizient mehrfach über diese zu iterieren, sondern überhaupt für längeren Zeitraum zu speichern. Hier bietet es sich an, diese Datenpunkte zum einmaligen Training zu verwenden, und im Anschluss komplett zu verwerfen. Dies resultiert in einem Verhalten, welches mit Offline Learning Algorithmen vergleichbar ist: Offline Learning Algorithmen, speziell Multi-Pass Algorithmen, itertieren mehrfach über den vorhandenen Datensatz, um die Modellparameter möglichst unter Beachtung von over-fitting gut auf diesen anzupassen. Sind ausreichend große Datenmengen vorhanden, so ist es möglich, in jedem Trainingsschritt einen neuen Datenpunkt zu wählen, und das Modell somit zu entwickeln, ohne einen Datenpunkt doppelt zu betrachten.

2.3 Nicht stationäre Zielfunktionen ("Concept drift")

In der Situation, dass sich die zugrundeliegende Verteilung im zeitlichen Verlauf der Trainingsdaten ändert, ist es nur schwer möglich, einen repräsentativen Datensatz für das Training eines Offline Modells zu wählen, welcher die aktuelle Verteilung ausreichend beschreibt [Sri15]. Online Modelle hingegen haben häufig die Möglichkeit, sich an die verändernde Verteilung der Trainingsdaten anzupassen (siehe Abschnitt 3.1).

Ein Anwendungsgebiet in dem dies zur Geltung kommt, liegt in Produktempfehlungen für Kunden bei Onlinehändlern. Diese sollten anpassungsfähig sein u.a. an die zuletzt betrachteten Produkte des Kundens oder anderweitig gesetzte Präferenzen.

Ein weiteres Beispiel liegt in Spam-Filtern. Diese gehören zu dem generellen Bereich des Online-Learnings, da das Modell durch kontinulierlich hinzukommende Spam-Emails erweitert werden kann. Desweiteren handelt es sich hierbei um eine spezielle Form einer veränderlichen Zielfunktion, da zwischen Spam-Filter und Spam-Emails eine Wechselwirkung besteht [Alm16]. Werden zuvor unbekannte Spam-Emails gefunden, so sollte sich der Filter an diese anpassen können. Die Verwendung des erweiterten Filters verringert die Reichweite der Mails, was wiederum zur Folge hat, dass diese von den Absendern verändert werden, um dem entgegen zu wirken. Diese adverseriale Abhängigkeit hat eine nicht stationäre Zielfunktion zur Folge, für die das Modell anpassungsfähig sein muss.

3 Herausforderungen

Mit der Anpassungsfähigkeit an veränderliche Zielfunktionen sowie Datensätze wachsender Größe, besonders bei häufig gleichbleibender Modellkomplexität, kommen entsprechende Schwierigkeiten einher. Abhängig von dem konkreten Anwendungfall handelt es sich um eine Abwägung zwischen verschiedenen Eigenschaften des Modells, welche es Situationsbedingt zu nutzen oder verhindern gilt. Dementsprechend ist es nicht möglich eine allgemeine optimale Lösung zu bieten, weshalb im Folgenden drei solcher Herausforderungen mit möglichen Lösungsansätzen angemerkt werden.

3.1 Stabilität und Sensitivität

Ist ein Modell anpassungfähig an veränderliche Zielfunktionen, so ist es zugleich empfindlicher gegenüber Ausreißern. Während man bei einer sich verändernden Zielfunktion von einem *Concept Drift* spricht, wird das damit einhergehende Problem der Anfälligkeit gegenüber einer Folge von Ausreißern als *Virtual Concept Drift* bezeichnet. Ohne stark einschränkende Annahmen über die Verteilung und die zeitliche Änderung dieser ist es nicht möglich, zwischen einer Änderung der zugrundeliegenden Verteilung und einer Folge von Datenpunkten "outliern" zu unterscheiden, welche dieser veränderten Verteilung entsprechen. Das Problem wird besonders unter Beachtung dessen klar, dass Online Learning Algorithmen nach Listing 4.2 die Trainingdaten genau in der Reihenfolge des Eintreffens und lediglich ein einziges mal anwenden. Dadurch kann im Gegensatz zu Batch Learning Verfahren im Hinblick auf den zeitlichen Verlauf keine Gleichverteilungsannahme der Daten gemacht werden [Alm16]. Werden so zum Beispiel Datenpunkte zweier überlappender Cluster sequentiell eingelesen (erst Klasse A, dann B), so ist nicht klar, ob sich die Verteilung so verändert hat, dass die zuerst eingelesenen Datenpunkte von A "veraltet" sind, und die Überlappungsgebiete eindeuting zu der neuen Klasse B zugeordnet werden sollten, oder noch der vorherigen angehören. Dementsprechend unterscheidet man bei Online Learning Modellen zwischen *Sensitivität* oder auch *Reaktivität* gegenüber sicher ändernder Verteilung, und *Stabilität* gegenüber Ausreißern. Einige Algorithmen wie z.B. der *Passive-Aggressive* Algorithmus (siehe Abschnitt 4.2) stellen hierfür eine veränderliche Lernrate zur Verfügung.

3.2 Modellbewertung

Einhergehend mit einer veränderlichen Verteilung der Daten, und somit erschwerten Bildung eines repräsentativen Datensatzes der aktuellen Verteilung, ist es ebenso nur

erschwert möglich, die aktuelle Modellleistung zu bewerten. In solchen Fällen ist es häufig nur möglich, den Fehler über die letzten k Datenpunkte zu schätzen, was jedoch dementsprechend anfällig gegenüber temporären Abweichungen in der Verteilung der Trainingsdaten ist. [Sri15]

3.3 Metaparameter

Aufgrund des im Vorraus häufig nicht bekannten Datenumfangs und insbesondere auch der zeitlich variablen Komplexität der Verteilung, ist die Abschätzung der optimalen Modellmetaparameter nur bedingt, oder für einen gewissen Zeitraum möglich. Eine hoch angesetzte Modellkomplexität würde somit anfängliches overfitting begünstigen, während eine niedrig angesetzte Modellkomplexität im Zeitlichen Verlauf zu underfitting neigt, und die jeweiligen Zwischenstufen ähnliche Probleme aufweisen, da sie die Verteilung nur zu einem bestimmten Zeitpunkt optimal beschreiben. Dadurch begründet ist eine adaptive Komplexität der Modelle gewünscht. [uBH]

4 Verfahren

4.1 Voting / Expert Advice

Ein mögliches Problem im Bereich des Online Learnings besteht in der Integration mehrerer externer Advisors, um ein gemeinsames Ergebnis zu bilden. Jeder dieser Advisors macht eine unabhängige Vorhersage über die Datenpunkte, wobei die Qualität der Vorhersagen je nach Advisor unterschiedlich ist. Desweiteren ist diese Qualität veränderlich. So hat Advisor A Beispielsweise zu Beginn eine hohe, aber steig abfallende Qualität, während die von Advisor B am Anfang niedriger ist, mit der Zeit jedoch steigt. Ein entsprechender Online-Learning Algorithmus sollte in der Lage sein, derartige Veränderungen zu erkennen, und unter Berücksichtigung derer eine eigene Vorhersage zu entwickeln.

Der Algorithmus aus Listing 4.1 basiert darauf, den Advisors eine Gewichtung zuzuweisen, welche als Glaubwürdigkeit interpretiert wird. Die Vorhersage des Modells ist dementsprechend als das gewichtete Mittel der Vorhersagen zu berechnen. Bei der Anschließenden Modellaktualisierung werden die Gewichtungen um einen Faktor verringert, welcher abhängig von der Qualität der Vorhersage des Advisors ist. Diese Qualität kann anhand einer beliebigen Distanzmetrik gemessen werden, in diesem Fall der Manhattan Distanz.

```
1  Given n predictors, learning rate α
2
3  Initialize:
4      w_i ←1
5
6  Monitor stream:
7      Receive predictions x_1,...,x_n
8      ŷ ←1/L ∑(w_i · x_i)  with  L = ∑(w_i)
9      Submit prediction ŷ
10
11     Receive correct value y
12     w_i ←w_i/(1 + α · Loss(x_i, y))  with e.g.  Loss(a, b) = |a − b|
```

Listing 4.1: Expert Advice Algorithmus

Derartige Umgewichtungen basierend auf der Abweichung der Vorhersage und des korrekten Wertes bilden die Basis des sogenannten *Regret-Based Learning* [BM]. Für

das Modell hat dies eine steigende Vorhersagequalität zur Folge [Sri15]. Desweiteren eignet sich das Verfahren als Ansatz für das Ensemble Learning im Bereich des Online Learnings.

4.2 Passive-Aggressive

Der Passive-Aggressive Algorithmus stellt einen linearen Klassifikator dar. Im Gegensatz zu dem bekannten Perzeptron Algorithmus wird eine möglichst große Distanz zwischen der Entscheidungsgrenze und den Datenpunkten aufgebaut, womit dieser zu den *Large Margin Classifiern* gehört. Dies gewährt eine erhöhte Stabilität der Klassifizierung neuer Datenpunkte [SBSS99].

Da im Gegensatz zu traditionellen Large Margin Classifiern des Batch Learning kein Zugriff auf den gesamten Datensatz, sondern nur auf einzelne Trainingsdatenpunkte besteht, ist es allgemein nicht möglich, die optimale Lage der Einscheidungsgrenze zu berechnen. Stattdessen wird ein Bereich variabler Größe um die Entscheidungsgrenze herum als *Pufferzone* betrachtet, so dass nicht nur bei falsch klassifizierten, sondern auch für Trainingspunkte innerhalb dieses Bereichs eine Gewichtsaktualisierung stattfindet. Als Verlustfunktion wird die folgende *hinge-loss function* verwendet. [CDK+06]

$$l(w, x, y) = \begin{cases} 0 & \text{falls } y(w \cdot x) \geq 1 \\ 1 - y(w \cdot x) & \text{sonst} \end{cases}$$

Der Term $y(w \cdot x)$ mit Gewichtsvektor w, Eingabe x und Zielklassifikation y basiert auf der Klassifikationsfunktion $w \cdot x$ linearer Modelle. Multiplikation mit $y \in \{-1, 1\}$ liefert die Distanz zur Entscheidungsgrenze mit positiven Werten bei korrekter Klassifikation und echt negativen Werten sonst. Das Ziel ist die Bestimmung des Gewichtsvektors, so dass alle Datenpunkte einen Abstand $y(w \cdot x)$ von mindestens 1 zur Entscheidungsgrenze überschreiten. Die Breite der Pufferzone wird in Algorithmus 4.2 implizit durch Skalierung des Gewichtsvektors angepasst.

```
1  Initialize:
2      w ←0⃗
3
4  Monitor stream:
5      Receive input x
6      Predict 1 if x * w > 0 else predict 0
7
8      Receive correct value y
9      l ← max(0, 1 - y * (x * w))
10
11     w ←w + y * l * x
```

Listing 4.2: Passive-Aggressive Algorithmus

Falls ein Datenpunkt falsch klassifiziert wurde oder innerhalb der Pufferzone lag, so nimmt das Updateverfahren genau die minimale Gewichtsänderung vor, die notwendig ist, so dass dieser im Anschluss sowohl korrekt klassifiziert wird, als auch die Pufferzone respektiert [Lav15a]. Ist der Datensatz linear separierbar, so konvergiert das Verfahren [Lav15b]. Andernfalls hat die Updateregel zur Folge, dass die Gewichte beliebig stark verändert werden, um sich dem zuletzt eingelesenen Datenpunkt anzupassen. Dieses Problem wird von dem Passive-Aggressive-II Algorithmus umgangen, indem die Verlustfunktion auf einen konstanten Wert m nach oben beschränkt wird. Entsprechend des Problems der Stabilität und Sensitivität aus Abschnitt 3.1 hat dies jedoch auch eine reduzierte Reaktivität zur Folge.

4.3 Online Random Forests

Das Konzept der Decision Trees und Random Forests aus dem Bereich des Offline Learning lässt sich erweitern, um auch ohne Speicherung von Trainingsdaten und Mehrfachiteration über diese vergleichbare Ergebnisse zu erzielen. Die Modellleistung des Online Random Forest Algorithmus kovergiert hierbei mit steigender Anzahl Trainingsdaten gegen die des Offline Gegenstücks. [SLS$^+$]

Im Batch Learning wird ein Baum rekursiv greedy aufgebaut, indem anhand aller Datenpunkte, die in einem Knoten des Baumes betrachtet werden entschieden wird, welche Testfunktion den lokal größten Informationsgewinn liefert. Derartige Berechnung des Informationsgewinns setzt voraus, dass alle Trainingsdaten zugreifbar sind, um einen möglichen Split auf ihnen zu evaluieren, was im Online Learning nicht gegeben ist. Eine weitere Herausforderung liegt darin, dass eine zuvor gewählte Testfunktion nach hinzukommen weiterer Datenpunkte durch die veränderte Verteilung nicht mehr optimal sein muss. Aufgrund des Top-Down Prinzips von Entscheidungsbäumen ist es nicht möglich eine solche falsche Entscheidung durch Hinzunahme weiterer Knoten rückgängig zu machen. Stattdessen müsste der gesamte (Teil-)Baum neu aufgebaut werden.

Online Random Trees Die direkte Berechnung der Entropie wird dadurch umgangen, dass mit der Erstellung eines Knotens mehrere potentielle Testfunktionen zufällig gewählt, und darauf folgend anhand der kontinuierlich eintreffenden Datenpunkte evaluiert werden. Dadurch ist es nicht notwendig den gesamten Datensatz zu speichern, sondern lediglich Informationen über die Verteilung der zuletzt eingetroffenen Datenpunkte im Bezug auf die Testfunktion. Im einfachsten Fall reicht es hierfür, die Anzahl Datenpunkte je Klasse zu zählen, die durch die Testfunktion positiv oder negativ bewertet werden. Mit diesem Verfahren ist es somit möglich, die gewählten Testfunktionen anhand der seit Erstellung dieses Knotens eingetroffenen Datenpunkten zu evaluieren.

Sind eine Mindestanzahl eingetroffener Datenpunkte erreicht, und liefert eine beliebige Testfunktion einen ausreichen großen Informationsgewinn, so wird diese als Testfunktion des Knotens gewählt und die restlichen verworfen. Die gesammelten Verteilungsinformationen werden an die Kinder weitergegeben und verwendet, um den eintreffenden Datenpunkten anhand des Maximum Likelihood Prinzips eine Klasse zuzuweisen. Die

Datenpunkte werden somit der seit Beginn der Evaluation am häufigsten auftretenden Klasse zugewiesen.

Beispiel. Ein Entscheidungsbaum zur Zuordnung in die Klassen A und B soll gebildet werden. Hierfür werden zwei beliebige Testfunktionen f und g erstellt. Um eine dieser Testfunktionen zu akzeptieren, werden die eintreffenden Trainingsdaten auf diesen ausgewertet. Bezeichne a/b die Klassenzugehörigkeit von Datenpunkten, wobei a bzw. b die Anzahl der Datenpunkte aus Klasse A bzw. B sei. Bietet die Testfunktion f nach 100 ausgewerteten Datenpunkten eine Aufteilung in 30/40 und 20/10, während Testfunktion g die Datenpunkte in 45/15 und 5/35 aufteilt, so erreicht f einen Informationsgewinn von 0.035, während der von g bei 0.296 liegt. Genügt dies den gewählten Splitbedingungen, so wird g als Testfunktion gewählt, ansonsten wird der Knoten vorerst beibehalten und weiter ausgewertet. Anhand der gesammelten Verteilungsinformationen würden die Datenpunkte des ersten Kindes entsprechend der 45/15 Klassenzugehörigkeit vorerst der Klasse A zugeordnet werden, da diese am wahrscheinlichsten eintritt. Zur Wahl oder Evaluation möglicher Testfunktionen können diese einfachen Statistiken jedoch nicht verwendet werden.

Online Random Forests Ähnlich zu Random Forests können auch mehrere Online Random Trees trainiert, und deren Ergebnisse integriert werden, um die Varianz des Gesamtmodells zu reduzieren. Besonders im Hinblick auf die natürliche suboptimalität der zufällig gewählten Entscheidungsfunktionen ist die Ausprägung mehrerer schwacher Lerner und darauf aufbauende Bildung eines starken Lerners sinnvoll.

Zusätzlich zu der Variation durch zufällig generierte Testfunktionen bieten sich die aus Offline Random Forests bekannten Methoden an. Eine Möglichkeit besteht darin, die Traingsdaten zufällig häufig zum Trainieren eines Baumes zu verwenden, zum Beispiel entsprechend einer Poissonverteilung mit Mittelwert $\lambda = 1$. Die Datenpunkte, welche Aufgrund eben dieser Verteilung nicht verwendet werden, um einen Baum zu trainieren, können stattdessen zur Bewertung verwendet werden, und bilden somit einen zufälligen Testdatensatz für den *out-of-bag error*.

Um letzendlich Änderungen der Verteilung behandeln zu können, werden Bäume mit hohen Fehlerquoten vollständig zurückgesetzt, damit diese sich unter kompletter Neubildung an die veränderte Verteilung anpassen.

4.4 Vergleich

Zur Einordnung des Passive Aggressive Algorithmus (Abschnitt 4.2) wurden zwei verwandte Algorithmen hinzugezogen. Dabei handelt es sich zum einen um den ebenfalls linearen Klassifikator *Perzeptron*, welcher jedoch nicht zu den Large Margin Classifiern gehört. Zum anderen wurde aus dem Bereich des Offline Learnings der *SVC* Algorithmus mit linearem Kernel gewählt. Für alle drei dieser Algorithmen wurde die Implementation der Machine Learning Library scikit-learn[1] verwendet. Anschließend wurde außerdem die

[1] `http://scikit-learn.org/`, Version 0.19.1

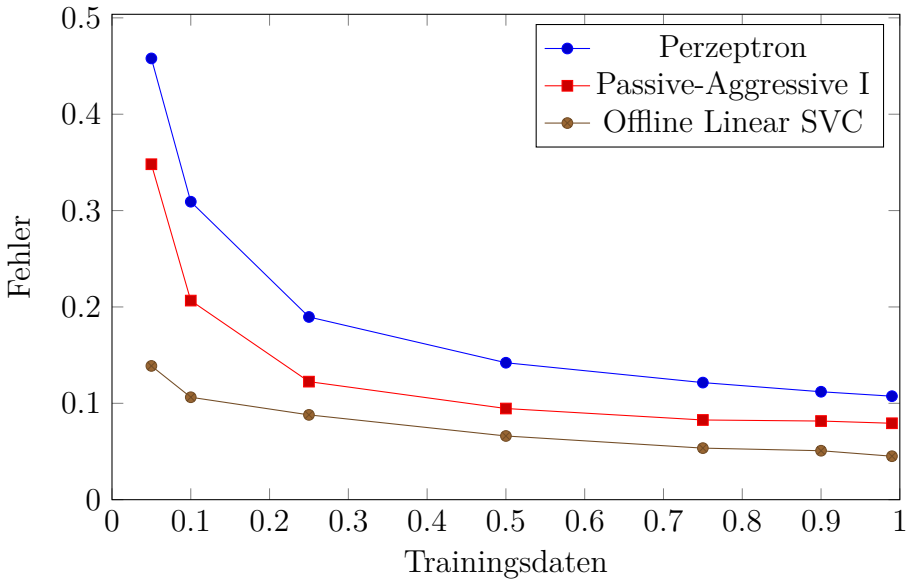

Abbildung 4.1: Vergleich von PA mit verwandten Algorithmen

Online Random Forest Implementation[2] von [SLS+] mit der Scikit-learn Implementation des Offline äquivalents verglichen.

Als Umgebung zur Auswertung wurde aus Abschnitt 2.1 das Beispiel der Handschriftenerkennung herangezogen, basierend auf dem UCI ML "Optical Recognition of Handwritten Digits Data Set"[3]. Dieses besteht aus ca. 1800 Bildern, jeweils 8×8 Pixel mit Helligkeitswerten im Bereich von 0 bis 16. Es wurden verschiedene Train/Test-Splits (holdout 95%, 90%, 75%, 50%, 25%, 10%, 1%) mit jeweils 500 Runden ausgeführt, und die Fehlklassifikationen gemittelt.

Ein erster Vergleich in Abbildung 4.1 zeigt, dass die Maximum Margin Methode des Passive Aggressive Algorithmus eine im Vergleich zu Perzeptron reduzierte Fehlerquote zur Folge hat, gegenüber SVC jedoch besonders bei kleinen Trainingssätzen bedeutend höhere Fehlerraten aufzeigt. Dies lässt sich u.a. darauf zurückführen, dass es sich bei SVC um einen Multi-Pass Offline Learning Algorithmus handelt, während Passive Aggressive lediglich ein einziges mal durch die Trainingsdaten iteriert (vgl. Abbildung 4.2). SVC wurde somit trotz gleicher Anzahl an Trainingspunkten *stärker* trainiert. Anzumerken ist auch, dass die Trainingsphase des SVC mit einem Faktor von ca. 20 eine bedeutend höhere Laufzeit hat.

Zum weiteren Vergleich wurde in Abbildung 4.2 bei allen drei obigen Algorithmen die Anzahl der Iterationen über den Trainingsdatensatz auf 5 begrenzt. Deutlich wird zum einen die Ähnlichkeit der Algorithmen PA und SVC. Bei beiden handelt es sich um lineare Klassifikatoren unter Verwendung der gleichen Verlustfunktion, und bei beiden wurde das *one-vs-rest* Verfahren angewendet. Im Gegensatz zu SVC erweist sich PA jedoch häufig als anfälliger gegenüber Rauschen [His13].

[2] `https://github.com/amirsaffari/online-random-forests`
[3] `http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits`

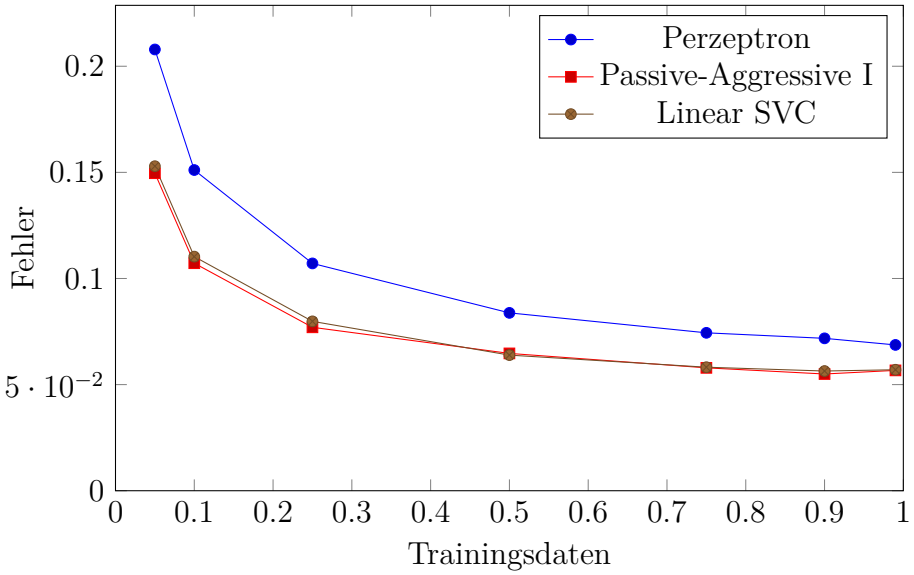

Abbildung 4.2: Vergleich unter gleichgesetzter Iterationszahl

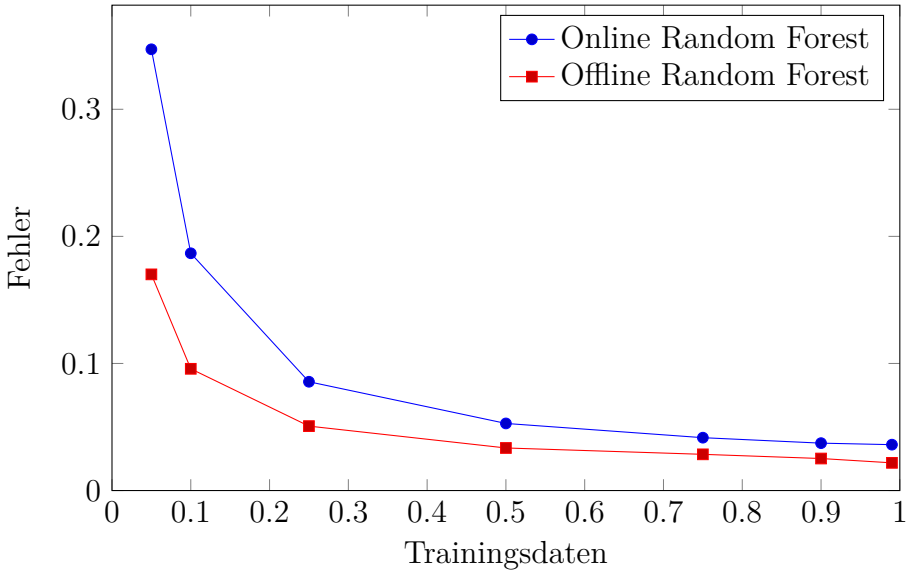

Abbildung 4.3: Vergleich von Offline und Online Random Forest Algorithmen

Der Vergleich der Offline und Online Varianten von Random Forest in Abbildung 4.3 zeigt ein ähnliches Bild wie der Vergleich von SVC mit PA. Das Offline Verfahren hat einen zu Beginn deutlich niedrigeren Fehlerquotienten, jedoch sinkt dieser mit hinzukommenden Trainingsdaten bei beiden Varianten ungefähr gleich schnell. Für die Wahl der Metaparameter wurden verschiedene Werte vergleichen verglichen, und basierend darauf in beiden Varianten die Anzahl der Bäume auf 100, deren maximale Tiefe auf 20 und die Anzahl zufällig generierter Tests pro Baum auf 5 gesetzt.

5 Zusammenfassung

Online Machine Learning Algorithmen bieten eine Möglichkeit, effizient kontinuierlich erweiterbare Modelle zu bilden, um große und echtzeit generierte Datenmengen zu handhaben. Damit einhergehend müssen solche Modelle eine nicht stationäre und ggf. beliebig komplexe Zielfunktion handhaben, um Änderungen der zugrundeliegenden Verteilung behandeln zu können. Hierbei gilt es, einen Kompromiss zwischen Reaktivität im Bezug auf eben solche Änderungen, und Stabilität gegenüber Ausreißern zu finden, um eine stete Modelleistung zu ermöglichen.

Für die gängigen Arten von Offline Learning Algorithmen existieren vergleichbare Implementationen aus dem Bereich des Online Learnings. Unter den linearen Large Margin Classifiern wurde der Passive-Aggressive Algorithmus betrachtet, zu Entscheidungsbäumen die Umsetzung der Online Random Forests, sowie ein Voting Algorithmus zur generellen Online Integration von Ensembles.

Besonders am Beispiel des Passive-Aggressive Algorithmus wurde erkenntlich, dass dieser unter gleichstellenden Bedingungen eine vergleichbare Fehlerquote erreicht wie Linear-SVC. In der Praxis sind jedoch, zumindest auf begrenzten Datensätzen, mehrere Trainingsiterationen notwendig, um tatsächlich ähnliche Leistungen zu gewährleisten [CC06].

Literaturverzeichnis

[Alm16] Felipe Almeida. Online machine learning: introduction and examples, 9 2016.

[Bek04] Ron Bekkerman. Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora, 2004.

[BM] A. Blum and Y. Mansour. Learning, regret minimization, and equilibria.

[BPF+16] Nikolay Burlutskiy, Miltos Petridis, Andrew Fish, Alexey Chernov, and Nour Ali. An investigation on online versus batch learning in predicting user behaviour, 11 2016.

[CC06] Vitor R. Carvalho and William W. Cohen. Notes on single-pass online learning algorithms, 7 2006.

[CDK+06] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, (7):551–585, 3 2006.

[con17a] Wikipedia contributors. Catastrophic interference — Wikipedia, the free encyclopedia, 2017. [Online; zugriff 25.11.2017].

[con17b] Wikipedia contributors. Gradientenverfahren — Wikipedia, the free encyclopedia, 2017. [Online; zugriff 03.12.2017].

[con17c] Wikipedia contributors. Online machine learning — Wikipedia, the free encyclopedia, 2017. [Online; zugriff 23.11.2017].

[con17d] Wikipedia contributors. Out-of-core algorithm — Wikipedia, the free encyclopedia, 2017. [Online; zugriff 20.11.2017].

[con17e] Wikipedia contributors. Perzeptron — Wikipedia, the free encyclopedia, 2017. [Online; zugriff 03.12.2017].

[con17f] Wikipedia contributors. Random forest — Wikipedia, the free encyclopedia, 2017. [Online; zugriff 08.12.2017].

[con17g] Wikipedia contributors. Winnow (algorithm) — Wikipedia, the free encyclopedia, 2017. [Online; zugriff 23.11.2017].

[His13] Sorami Hisamoto. Perceptron, support vector machine, and passive aggressive algorithm, 5 2013.

[KPR+17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks, 2017.

[Kun17] Dr. Julian Kunkel. Vorlesung big data analytics, 12 2017.

[Lav15a] Victor Lavrenko. Ir20.3 passive-aggressive algorithm (pa), 9 2015.

[Lav15b] Victor Lavrenko. Ir20.4 convergence of the pa algorithm, 9 2015.

[LN12] Pavel Laskov and Blaine Nelson. Online and incremental learning; vorlesung big data analytics, 6 2012.

[Par13] Charles Parker. Machine learning from streaming data: Two problems, two solutions, two concerns, and two lessons, 3 2013.

[SBSS99] Alexander J. Smola, Peter Barlett, Bernhard Schölkopf, and Dale Schuurmans. Advances in large margin classifiers, 7 1999.

[SLS+] Amir Saffari, Christian Leistner, Jakob Santer, Martin Godec, and Horst Bischof. On-line random forests.

[SP] Parth Shah and Riju Pahwa. Lecture notes in urban data analytics and machine learning.

[Sri15] Tavish Srivastava. Introduction to online machine learning : Simplified, 1 2015.

[uBH] Alexander Gepperth und Barbara Hammer. Incremental learning algorithms and applications.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Studiengang B.Sc. Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum	Unterschrift

Veröffentlichung

Ich bin damit einverstanden, dass meine Arbeit in den Bestand der Bibliothek des Fachbereichs Informatik eingestellt wird.

Ort, Datum	Unterschrift