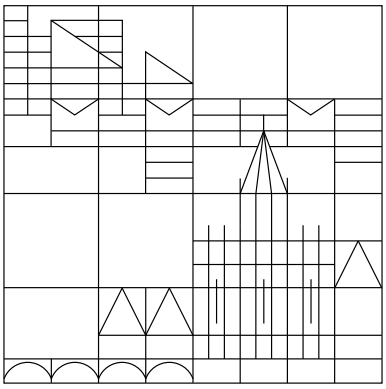
Universität Konstanz



Aktives Lernen zur Klassifikation großer Datenmengen mittels Exploration und Spezialisierung

Dissertation

zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften (Dr. rer. nat.)
an der Universität Konstanz
Fachbereich für Informatik und Informationswissenschaft

vorgelegt von
Nicolas Cebron

Tag der mündlichen Prüfung: 27. Mai 2008
Referenten:
Prof. Dr. Michael R. Berthold
Prof. Dr. Ulrik Brandes

Konstanzer Online-Publikations-System (KOPS)
URL: http://www.ub.uni-konstanz.de/kops/volltexte/2008/5934/
URN: http://nbn-resolving.de/urn:nbn:de:bsz:352-opus-59341

Cebron, Nicolas:
Aktives Lernen zur Klassifikation großer Datenmengen mittels Exploration und Spezialisierung
Universität Konstanz, 2008.

Abstract

The paradigm of active learning is often used to classify large datasets with the help of a human expert in different application areas. The number of examples that need to be classified by the expert in order to build a stable model can be reduced with a selective sampling strategy.

Current state-of-the-art active learning algorithms often deal insufficently with the aspect of exploration. They assume that a stable classification model has been already build and needs to be refined with further carefully selected examples. In this dissertation, two new approaches are introduced that include the aspect of exploration in active learning. In contrast to most of the other active learning methods, the selection strategy is applied from the very first example. After a stable classification model has been build, the selection strategy focuses on the classification boundaries.

The first approach called „Active Learning Vector Quantization“ explorates the data with a global clustering method. In a second phase, the human-classified clusters are further refined with selected examples.

The second approach called „Prototype Based Active Classification“ creates a new prototype for a k-nearest neighbour classification in each learning iteration. The selection of a data-point as a prototype depends on a combination of its representativeness of neighboring unlabeled data-points and the uncertainty of the classifier in predicting its class label. The proposed approach combines the trade-offs of exploration and exploitation via the newly developed uncertainty distribution seamlessly. With each learning iteration the effect of exploration reduces and exploitation increases.

The practical application is demonstrated by a specific application in the field of bioinformatics.

The strategy of first exploring the dataset and subsequently improving the classification model turns out to be beneficial for classification performance and classifier stability.

Zusammenfassung

Das Paradigma des Aktiven Lernens wird häufig in praktischen Anwendungsszenarien angewendet, um große Datenmengen mit Hilfe eines menschlichen Experten zu klassifizieren. Durch eine gezielte Auswahl soll die Anzahl der Muster reduziert werden, die vom Experten klassifiziert werden müssen um ein stabiles Klassifikationsmodell zu lernen.

Bei bisherigen Ansätzen im Bereich des Aktiven Lernens wird oft angenommen, dass ein stabiles Klassifikationsmodell bereits mit zufällig gezogenen Mustern gelernt wurde, welches mit ausgesuchten Beispielen verfeinert werden soll. In dieser Arbeit werden zwei Ansätze zum Aktiven Lernen vorgestellt, die vom ersten Beispiel an versuchen eine sinnvolle Auswahl aus den Daten zu treffen. Zunächst wird bei der Selektion von Mustern der Aspekt der Exploration betont. Nachdem ein stabiles Klassifikationsmodell gelernt wurde, konzentriert sich die Selektionsstrategie auf die Klassengrenzen.

Das erste Verfahren der „Aktiven Lernenden Vektor Quantisierung“ exploriert den Datensatz mittels eines Fuzzy c-means Clusterings. Anschließend werden die vom Experten klassifizierten Cluster mit ausgesuchten Beispielen an den Klassifikationsgrenzen angepasst.

Das zweite Verfahren der „Aktiven Prototypen basierten Klassifikation“ generiert in jeder Lerniteration einen Prototypen für eine k-nächste Nachbarn Klassifikation. Dieser Ansatz kombiniert die Aspekte der Exploration und der Verfeinerung der Klassengrenzen mittels einer neu entwickelten Unsicherheitsverteilung miteinander. Dabei findet mit zunehmender Anzahl von klassifizierten Mustern ein fließender Übergang zwischen Exploration und Verfeinerung der Klassengrenzen statt.

Anhand der spezifischen Anwendung der Verfahren für die Klassifikation von Zellbildern durch einen Biologen wird der praktische Nutzen aufgezeigt.

Die beiden Ansätzen zugrunde liegende Strategie, die Daten zunächst zu explorieren und anschließend das Klassifikationsmodell zu verfeinern stellt sich als vorteilhat für die Performanz und Stabilität gegenüber bisher entwickelten aktiven Lernverfahren heraus.

Danksagungen

Ein besonderer Dank gilt meinem Doktorvater Prof. Dr. Michael Berthold für die Bereitstellung dieses interessanten Themas. Seine Ideen, Kommentare und Anregungen haben mir bei der Erstellung der Arbeit sehr geholfen.
Bedanken möchte ich mich ebenfalls bei Prof. Dr. Ulrik Brandes für die Übernahme des Korreferats und des Prüfungsvorsitzes. Durch Prof. Dr. Frank Klawonn bin ich während meines Studiums auf das Gebiet des Data Mining aufmerksam geworden. Ich bin ihm sehr dankbar für seine Unterstützung und seine Teilnahme an der mündlichen Prüfung.
Besonders danken möchte ich mich auch bei allen Mitgliedern des Lehrstuhls für interessante Diskussionen, Anregungen und nicht zuletzt auch jede Menge Spaß.
Die vorliegende Arbeit entstand im Rahmen eines Stipendiums der Deutschen Forschungsgemeinschaft (DFG) im Graduiertenkolleg „Explorative Analyse und Visualisierung großer Datenräume“. Für die finanzielle Unterstützung und den wissenschaftlichen Austausch im Rahmen dieses Forschungsprogrammes bin ich sehr dankbar.
Meiner Familie und meine Freunde waren mir in der Zeit eine große Hilfe und haben mir viel Unterstützung gegeben.
Mein ganz besonderer Dank gilt meiner Freundin Manuela Stenzel, die mich immer unterstützt hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Der Begriff des Lernens wird in verschiedenen Fachgebieten wie der Biologie, Psychologie oder der Informatik unterschiedlich verwendet. In der Biologie wird der Begriff des Lernens beispielsweise als „durch Erfahrung entstandene Verhaltensänderungen und -möglichkeiten, die Organismen befähigen, aufgrund früherer und weiterer Erfahrungen situationsangemessen zu reagieren“ (Schwachulla, 1998) definiert. Über die einfache Konditionierung (z. B. bei primitiven Lebewesen, welche adaptiv auf ihre Umwelt reagieren) hinaus soll in diesem Abschnitt der Begriff des komplexen Lernens erläutert werden, der in enger Beziehung zu dieser Arbeit steht.

Dieses Lernen beinhaltet die Anwendung einer systematischen Strategie, um ein Problem zu lösen, also auch das Durchspielen mehrerer Möglichkeiten. Dies wiederum setzt die Fähigkeit zur Abstraktion voraus, um mentale Abbilder der Welt zu erstellen, in denen verschiedene Szenarien angewendet werden können. Diese Abstraktion kann beim komplexen Lernen deduktiv oder induktiv erworben werden:

- Beim deduktiven Lernen wird das Spezielle aus dem Allgemeinen erlernt. Der zu lernende Zusammenhang wird in einer Beschreibung vorgegeben und durch den Lernenden analysiert, der daraus eine neue Abstraktion generiert.
- Beim induktiven Lernen wird das Allgemeine aus dem Speziellen gelernt. Eine Reihe von Beispielen (und Gegenbeispielen) werden für die zu lernende Abstraktion vorgegeben. Gemeinsamkeiten und Unterschiede werden vom Lernenden herausgefiltert und bilden den Inhalt der neu gelernten Abstraktion. Das induktive Lernen kann die genaue Grenzziehung einer Abstraktion nicht mit vertretbarem Aufwand lernen. Oft ist eine große Reihe von Beispielen nötig, um die meisten Zweifelsfälle auszuschließen.

Im Kontext des maschinellen Lernens sind die beiden allgemeinen Lernansätze des induktiven und deduktiven Lernens aus dem Bereich der Biologie wiederzufinden. In dieser Arbeit beschäftigen wir uns mit der Version des induktiven Lernens. Anstatt das Wissen explizit in einem Algorithmus oder in einer Formulierung vorzugeben, soll mit Hilfe von gezeigten Mustern eine Gesetzmäßigkeit gelernt werden. Dies geschieht beispielsweise,

weil eine explizite Formulierung des Konzeptes in manchen Fällen unmöglich oder zu aufwändig ist. Oft sind die Muster auch schon in großer Anzahl vorhanden und können direkt verwendet werden. Darüberhinaus ist das Lernen mit Mustern leichter an neue Gegebenheiten anzupassen und flexibler. Es lässt sich daher in realitätsnahen Szenarien leichter und schneller einsetzen.

Im Bereich der Informatik stehen oft riesige Datenmengen aus den unterschiedlichsten Anwendungsbereichen in Form von Mustern für das induktive Lernen zur Verfügung. Um diese Daten zu klassifizieren, ist jedoch in vielen Anwendungsfällen eine manuelle Klassifikation durch einen menschlichen Experten nötig.

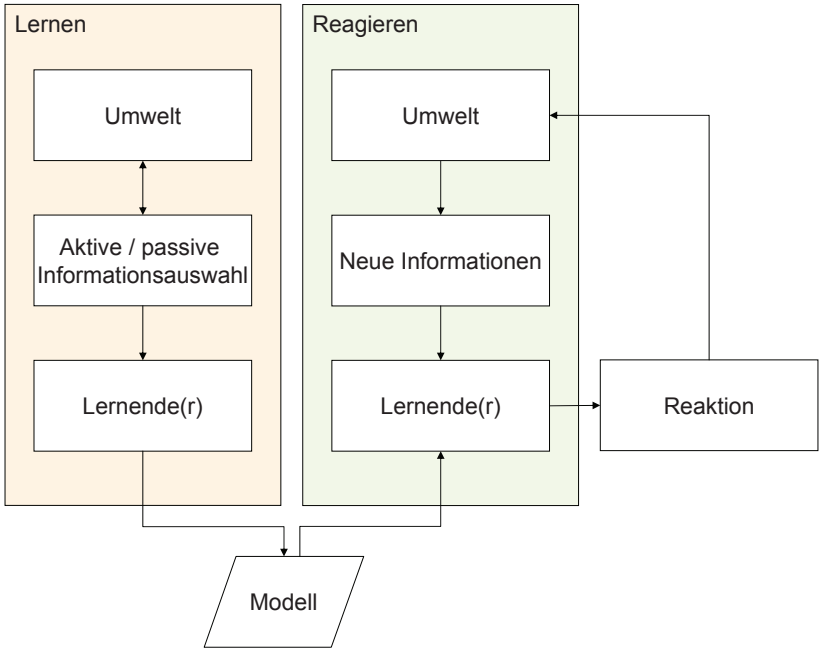
Das Ziel des maschinellen Lernens ist es, mit Hilfe von klassifizierten Mustern automatisch ein Modell zu lernen. Dieses Modell soll sowohl möglichst einfach (z. B. eine möglichst geringe Anzahl von Regeln) als auch möglichst generalisierend sein, d. h. eine gute Performanz auf neuen unbekannten Daten haben. Das Ziel ist es, Gesetzmäßigkeiten in den Daten aufzudecken und mit diesem Modell weitere Daten zu klassifizieren.

Die Anwendungspalette des maschinellen Lernens ist sehr umfangreich und reicht von automatischer Sprach- und Schrifterkennung, Klassifikation von Webseiten, Textdokumenten und DNA-Sequenzen bis hin zur Erkennung von Kreditkartenbetrug.

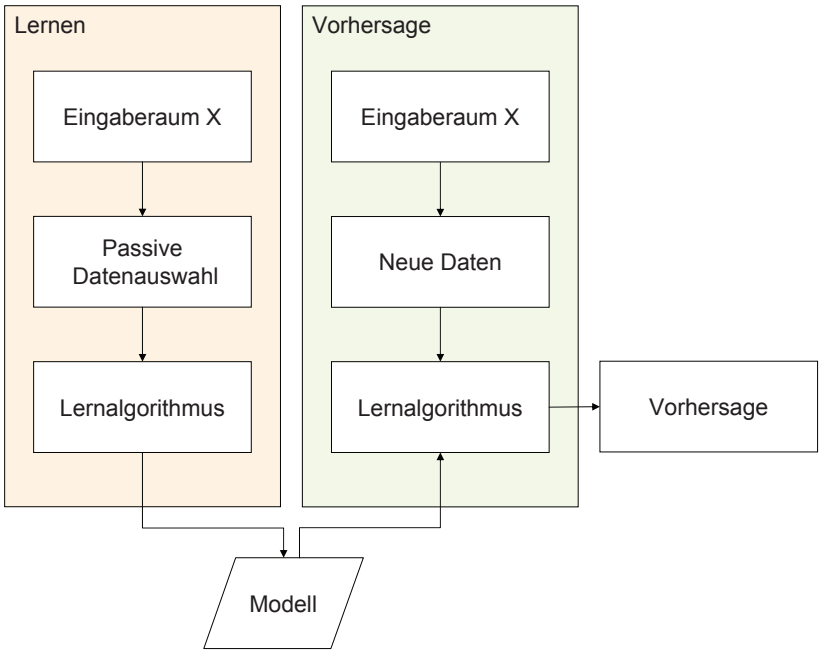
Abbildung 7.4 zeigt die Analogien und Unterschiede zwischen einem generellen Lernszenario und dem maschinellen Lernen. Im generellen Lernszenario kann der Lernende passiv oder aktiv Informationen aus seiner Umwelt beziehen. Basierend auf diesen Informationen versucht er, ein abstraktes (Gedanken)Modell zu bilden. Wird der Lernende anschließend mit neuen Informationen aus seiner Umwelt konfrontiert, so kann er auf das gelernte Modell zurückgreifen und entsprechend reagieren. Das resultierende Verhalten kann wiederum die Umwelt des Lerners aktiv beeinflussen.

Im klassischen maschinellen Lernen ist der Lernvorgang passiver. Der Lernalgorithmus erhält eine Teilmenge der zugrunde liegenden Eingabedaten (Muster und optional deren Klassifikation) und versucht anschließend, ein Modell aus diesen Informationen zu lernen. Mit Hilfe des gelernten Modells können dann neue unbekannte Eingabedaten klassifiziert werden. Diese beeinflussen jedoch den Eingaberaum nicht.

Die Umwelt eines maschinellen Lerners besteht aus Eingabemustern, welche sich aus verschiedenen Typen von Attributen (numerisch, nominal, ordinal) zusammensetzen. So lässt sich beispielsweise eine Kreditkartentransaktion in einem Muster zusammenfassen, das aus Kontonummer, Art der Kreditkarte und Transaktionsbetrag besteht. Oft fallen solche Daten automatisch - wie z. B. bei der Verarbeitung einer Kreditkartentransaktion



(a) Generelles Lernszenario



(b) Maschinelles Lernen

Abbildung 1.1.: Analogien und Unterschiede zwischen abstraktem Lernen und maschinellem Lernen

- an.

In vielen Anwendungsfällen werden auch aus so genannten „Rohdaten“, wie z. B. aus Textdokumenten auf automatisierte Weise Muster erzeugt. So lässt sich eine Webseite beispielsweise durch die Anzahl von häufig vorkommenden Wörtern und der Anzahl von Hyperlinks beschreiben. Die Erzeugung von beschreibenden Attributen kann in vielen Fällen automatisch erfolgen und erfordert nur wenig menschliche Interaktion. Aufgrund ähnlicher Muster können Webseiten in gemeinsame Gruppen eingeteilt werden. Dabei können jedoch z. B. durch Homonyme Inkonsistenzen auftreten. Homonyme sind Wörter mit unterschiedlichen Bedeutungen, so kann das Wort „Bank“ sowohl für eine Sitzgelegenheit als auch für ein Geldinstitut stehen. Eine eindeutige Zuordnung zu einer Kategorie bzw. Klasse ist bei der Gruppierung alleine durch die Ähnlichkeit von Mustern in diesem Fall nicht möglich.

Bei Webseiten versuchen viele Suchmaschinenbetreiber beispielsweise, dem Benutzer bessere Suchergebnisse zu präsentieren, indem sie die Suchergebnisse durch menschliche Experten kategorisieren lassen, um somit doppelte Wortbedeutungen aufzulösen. Durch die Information der Klassifikation von Mustern durch einen menschlichen Experten kann ein Klassifikationsmodell automatisch gelernt werden. Die Klassifikation von Mustern durch einen Menschen verursacht jedoch einen hohen Zeit- und Kostenaufwand.

Je nach Anwendungsfall kann der Eingaberaum für einen maschinellen Lerner demnach aus Eingabemustern bzw. Vektoren $\boldsymbol{x}$ bestehen oder auch aus Mustern mit zusätzlicher Klassifikation durch einem menschlichen Experten $(\boldsymbol{x}, \boldsymbol{y})$.

Das Ziel im maschinellen Lernen ist es, mit Hilfe einer begrenzten Anzahl von Trainingsmustern ein Modell zu lernen, um so in sehr großen Datenmengen interessante Zusammenhänge aufzudecken. Dabei wird grundsätzlich zwischen zwei verschiedenen Szenarien unterschieden: dem überwachten und dem unüberwachten Lernen. Beim überwachten Lernen wird versucht, das Zielattribut $\boldsymbol{y}$ zu bestimmen. Ist das Zielattribut numerisch, spricht man von Regression. So möchte man z. B. mit Hilfe der Angaben zu Größe und Geschlecht das Gewicht einer Person bestimmen. Bei der Klassifikation hingegen ist das Zielattribut nominal, d. h. das Zielattribut nimmt diskrete Werte an, die wir als so genannte Labels bezeichnen.

Beim unüberwachten Lernen fehlen die Anhaltspunkte, wie gut die Leistung des Modells in Bezug zu einer gegebenen Trainingsmenge ist. Hier stehen nur die Eingabedaten aus dem Eingaberaum X ohne Klassenattribut zur Verfügung. Ein großes Teilgebiet im unüberwachten Lernen stellen die Cluster-Verfahren dar, welche dazu dienen, die unter-

liegende Struktur der Daten zu ermitteln. Es wird das Ziel verfolgt, Gruppierungen und Srukturen in den Daten zu finden und zu beschreiben. Dabei muss die (Un)ähnlichkeit von zwei Mustern fallweise definiert werden, z. B. durch die euklidische Metrik.

Sowohl beim überwachten als auch beim unüberwachten Lernen wird üblicherweise eine signifikante Quantität an Daten zufällig aus der zugrunde liegenden Verteilung der Eingabedaten gezogen (wobei häufig angenommen wird, dass die Muster in den Daten unabhängig und gleichverteilt sind), um dann ein Modell der Daten zu lernen. Diese Methode wird als (passives) maschinelles Lernen bezeichnet.

Wir unterscheiden beim maschinellen Lernen zwischen zwei verschiedenen Lernszenarien. Beim so genannten Online-Lernen beobachtet der Lerner eine Sequenz von Mustern und generiert iterativ ein Modell, welches die Daten beschreibt. Beim so genannten Offline-Lernen stehen bereits alle Daten zur Verfügung.

Weiterhin unterscheidet man, ob die Wissensrepräsentation explizit vorliegt (z. B. in Form von Regeln) oder ob sie implizit im Modell (blackbox) vorhanden ist, wie es beispielsweise bei neuronalen Netzen der Fall ist.

1.1. Warum Aktives Lernen?

In vielen Anwendungsszenarien werden die Daten automatisch gewonnen und stehen damit bereits in großer Anzahl zur Verfügung. Die manuelle Klassifikation der Daten ist hingegen ein zeitaufwändiger und kostenintensiver Prozess. Dies liegt in den meisten Fällen daran, dass die Daten von einem menschlichen Experten klassifiziert werden müssen. Hierbei kann die Interpretation des Musters durch den Experten und die darauf folgende Kategorisierung einige Zeit in Anspruch nehmen und damit hohe Kosten verursachen. Es kann aber auch weitere Ursachen geben, wieso eine manuelle Klassifikation teuer und zeitaufwändig ist. In manchen Fällen wird das Klassenlabel beispielsweise durch ein Experiment bestimmt, dessen Durchführung hohe Kosten verursacht.

In vielen Situationen besteht die Möglichkeit, die Ziehung von geeigneten Mustern zu beeinflussen. Unter Berücksichtigung der Verteilung der Daten und des bisher gelernten Modells kann man geeignete Muster wählen, um somit schneller ein Klassifikationsmodell zu lernen. Dabei müssen nicht alle Muster auf einmal ausgewählt werden. Vielmehr kann man den Lernprozess in einzelne Iterationen aufspalten, wobei in jeder Lerniteration ein neues Muster aufgrund der in den vorherigen Iterationen klassifizierten Muster ausgewählt wird. Dieser Prozess der Einflussnahme auf die Selektion von neuen Mustern

wird als Aktives Lernen (Cohn u. a., 1994a) bezeichnet.

Ein typisches Anwendungsbeispiel ist die im vorherigen Abschnitt angesprochene Klassifikation von Textdokumenten. Für ein Textdokument lassen sich automatisch Attribute berechnen, welche z. B. die Häufigkeit von bestimmten Wörtern im Kontext zu anderen Wörtern berechnen. Solche Daten können in fast beliebiger Menge zur Verfügung gestellt werden. Eine Unterscheidung von Dokumenten in verschiedene Themenbereiche kann sich in manchen Fällen jedoch als schwierig gestalten. Dies kann beispielsweise der Fall sein, wenn dieselben Wörter in unterschiedlichen Kategorien häufig in den Dokumenten auftreten. Eine Kategorisierung eines Dokuments durch einen Menschen kann dabei sehr hilfreich sein, ist jedoch im Gegensatz zu den automatisch generierten Attributen aufwändig. So kann die Bewertung durch einen Experten zugleich hohe Kosten verursachen und sehr zeitintensiv sein.

Ein weiteres mögliches Anwendungsfeld kommt aus der bestehenden Industriekollaboration im Bereich der Bioinformatik, in dem der Einsatz von Hochgeschwindigkeitskameras zur Fotografie von Zellbildern neue Einsatzmöglichkeiten im Bereich der Wirkstoffanalyse eröffnet. Diese Geräte sind in der Lage, hunderttausende von Zell-Assay Bildern innerhalb eines einzigen Tages zu produzieren. Die Analyse jedes einzelnen Bildes durch einen Biologen ist dabei kaum noch möglich. Daher werden bisher in aufwändiger Kleinarbeit spezielle, an das Problem angepasste Skripte zur Bildverarbeitung und Klassifikation solcher Bilder entwickelt. Das Ziel unserer Arbeit ist es, den Prozess der Zellbild-Klassifikation zu automatisieren. Dazu werden eine Hand voll interessanter Beispiele aus den Bilddaten selektiert und dem Experten zur Klassifikation gezeigt. Basierend auf diesen wenigen, hand-klassifizierten Beispielen soll ein Modell gelernt werden, welches anschließend in der Lage ist, den Rest der Bilddaten zu klassifizieren.

In dieser Arbeit wollen wir uns dem allgemeinen Problem widmen, eine große Menge von Daten mit Hilfe eines menschlichen Experten zu klassifizieren. Basierend auf einem grob gelernten Modell soll in weiteren Schritten die Klassifikation durch ein so genanntes Orakel (üblicherweise eine menschliche Person) weiter verfeinert werden. Zu diesem Zweck müssen Muster nach einer geeigneten Strategie ausgewählt werden.

In einer relativ kurzen Zeitspanne sind diverse Techniken und Strategien zum Aktiven Lernen entwickelt worden. Bisherige Konzepte im Aktiven Lernen beschäftigen sich aber nur wenig oder unzureichend mit der Exploration der Daten. Meist wird davon ausgegangen, dass zu Beginn eine kleine Menge von klassifizierten Trainingsdaten zur Verfügung steht, um ein Modell zu initialisieren. Desweiteren ist die Selektionsstrategie bei den

bisherigen Verfahren oft unflexibel und passt sich nicht den Erfordernissen während des Lernens an.

Daher wurden in dieser Arbeit Ansätze entwickelt, die den Datenraum zunächst explorieren und in späteren Iterationen die Strategie der Auswahl ändern, indem der Fokus mehr auf die Grenzen zwischen den Klassen gelegt wird.

1.2. Beiträge dieser Forschungsarbeit

Wesentliche Ergebnisse der vorliegenden Arbeit wurden in folgenden Tagungsbänden und Zeitschriften publiziert:

- Nicolas Cebron und Michael R. Berthold. Active Object Classification: From Exploration to Exploitation. *Journal of Data Mining and Knowledge Discovery*, 2008. Akzeptiert.
- Nicolas Cebron und Michael R. Berthold. Adaptive Prototype Based Fuzzy Classification. *Fuzzy Sets and Systems*, 2008. in Druck.
- Nicolas Cebron und Michael R. Berthold. Adaptive Active Classification of Cell Assay Images. *Knowledge Discovery in Databases: PKDD 2006 (PKDD/ECML)*, B. 4213, S. 79-90. Springer Berlin/Heidelberg, 2006.
- Nicolas Cebron und Michael R. Berthold. Adaptive Fuzzy Clustering. *Proc. Conf. North American Fuzzy Information Processing Society (NAFIPS 2006)*, S. 188-193, 2006.
- Nicolas Cebron und Michael R. Berthold. Adaptive Klassifikation von Zellbildern. *Proceedings 16. Workshop Computational Intelligence*, S. 223-234, 2006.
- Nicolas Cebron und Michael R. Berthold. Mining of Cell Assay Images using Active Semisupervised Clustering. *IEEE International Conference on Data Mining, Workshop Computational Intelligence in Data Mining (ICDM 05)*, S. 63-69, 2005.

1.3. Aufbau

Nach einer Einführung in die Thematik des maschinellen Lernens in Kapitel 2 wird in Kapitel 3 näher auf das Teilgebiet des Aktiven Lernens eingegangen. Nach einer Formalisierung und Kategorisierung verschiedener aktiver Lernansätze werden die bisherigen aktiven Lernverfahren vorgestellt.

In dieser Arbeit wird ein Prototypen basierter Ansatz für die Zusammenfassung und Klassifikation von Daten angewendet. Verschiedene Verfahren für die Initialisierung und Anpassung eines Modells von Prototypen werden in Kapitel 4 eingeführt.

Basierend auf diesem Modell und Überlegungen zu bisherigen Defiziten in aktiven Lernverfahren wird in Kapitel 5 ein erstes Verfahren zur aktiven Klassifikation vorgestellt. Hier wird in der ersten Phase ein Fuzzy Clustering durchgeführt, um den Datensatz zusammenzufassen und grob zu klassifizieren. Anschließend helfen ausgesuchte Beispiele, die Klassifikation in der zweiten Phase noch weiter zu verbessern.

In Kapitel 6 wird auf den im vorherigen Kapitel entwickelten Ideen ein erweitertes Verfahren zur Prototypen basierten aktiven Klassifikation eingeführt. Eine Kombination aus lokaler Dichteschätzung und Klassifikator-Unsicherheit bildet ein Kriterium zur aktiven Selektion von Mustern. Durch Reduktion von Dichte-Potentialen findet ein weicher Übergang von der Exploration der Daten auf die Spezialisierung an den Klassengrenzen statt.

Kapitel 7 vergleicht die Ergebnisse der verschiedenen aktiven Lernverfahren auf unterschiedlichen Datensätzen. Zunächst wird auf einem künstlichen Datensatz die Arbeitsweise der unterschiedlichen Lernverfahren verdeutlicht. Auf benchmark-Daten aus dem UCI-Repository wird die Performanz bezüglich der Anzahl der gezeigten und klassifizierten Beispiele analysiert. Ein selbst erzeugter Datensatz aus Bildern von Gesichtern wird auch mit den verschiedenen aktiven Lernverfahren klassifiziert und die Performanz analysiert.

Die praktische Anwendung der aktiven Lernverfahren zur bildgestützten Analyse von Wirkstoffen wird in Kapitel 8 vorgestellt. Hier wird der Prozess der Vorverarbeitung von den Rohdaten der Zell-Assay Bilder über die Segmentierung, Merkmalsberechnung bis hin zur Klassifikation durch den biologischen Experten veranschaulicht.

Eine Zusammenfassung findet sich in Kapitel 9, hier wird auch ein Ausblick auf mögliche Erweiterungen der entwickelten Verfahren gegeben.

1.4. Notation

Vektoren werden als Kleinbuchstaben fettgedruckt dargestellt, z. B. ist $\boldsymbol{x}$ ein d-dimensionaler Vektor $\boldsymbol{x} = (x_1, \ldots, x_d)^T$ (T für transponiert). $\boldsymbol{x}_i$ ist der i-te Vektor, also $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{id})^T$. Vektoren sind immer kleingeschrieben, der Großbuchstabe steht für den entsprechenden Raum. So stammt der Parametervektor $\boldsymbol{w}$ beispielsweise aus dem Parameterraum W.

Tabelle 1.1 stellt die verwendeten Symbole dar.

Symbol	Erklärung
X	d-dimensionaler Eingaberaum $\mathbb{R}^d$
Y	q-dimensionaler Ausgaberaum $\mathbb{R}^q$
$Z = X \times Y$	Eingabe/Ausgaberaum
M	Modell des Lernalgorithmus
U	Menge der nicht klassifzierten Muster aus X
L	Menge der klassifizierten Muster aus Z
D_{train}	Trainingsdaten (aus L oder U)
D_{test}	Testdaten (aus L oder U)
W	Parameter-Raum
$d(\boldsymbol{a}, \boldsymbol{b})$	Distanzfunktion zwischen den Vektoren $\boldsymbol{a}$ und $\boldsymbol{b}$
$E(M, D_{\text{train}})$	Trainingsfehler
$E_i(M, \boldsymbol{x}_i)$	Fehler des Modells M für das Muster $\boldsymbol{x}_i$
$E_{\text{Gen}}(M)$	Generalisierungsfehler
$\hat{y}(\boldsymbol{x})$	Ausgabe des Modells für das Muster $\boldsymbol{x}$
$y(\boldsymbol{x})$	Wahre bzw. gewünschte Ausgabe für das Muster $\boldsymbol{x}$
$p(\cdot)$	Wahrscheinlichkeitsverteilung
$p(y, x)$	Verbundverteilung der Ein- und Ausgabedaten
$P(y\|x)$	Wahrscheinlichkeit für y gegeben x
H	Hypothesenraum
V	Versionsraum
$E_D[\cdot]$	Erwartungswert bezüglich Trainingsmenge D
q	Selektionsfunktion des aktiven Lerners
$\mathit{Utility}(M,\boldsymbol{x}_u)$	Bewertungsfunktion für $\boldsymbol{x}_u$
P	Potentialfunktion
C	Klassifikator-Unsicherheitsfunktion
D	Unsicherheitsverteilung
$N(x_i)$	Nachbarschaft von $\boldsymbol{x}_i$

Tabelle 1.1.

2. Lernen

Wie wir bereits in der Einleitung festgestellt haben, wird beim maschinellen Lernen die Umwelt durch einen abstrakten Eingaberaum X beschrieben. Dieser enthält Muster, welche in einer für den Lernalgorithmus geeigneten Repräsentation vorliegen. In dieser Arbeit beschränken wir uns auf den Fall, dass der Eingaberaum durch eine Menge von n numerischen Attributvektoren $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ beschrieben wird, welche im d-dimensionalen euklidischen Raum $\mathbb{R}^d$ liegen.

Die nicht klassifizierten Muster werden als $U \subseteq X$ bezeichnet. Die Menge der m klassifizierten Muster Z ist eine Teilmenge von X mit den dazugehörigen Klassen (aus einer Menge von möglichen Klassen Y): $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\} \subset X \times Y$. X wird überlichweise als Eingaberaum, Y als Ausgaberaum und Z als Eingabe/Ausgaberaum bezeichnet.

Üblicherweise wird die Menge der nicht klassifizierten Muster U oder die Menge der klassifizierten Muster L in eine Menge von Trainingsdaten D_{train} und in eine Menge von Testdaten D_{test} aufgespalten. Die Trainingsdaten werden verwendet, um ein Klassifikationsmodell M zu trainieren. Die Testdaten werden verwendet, um die Performanz des Modelles für neue unbekannte Daten schätzen zu können. Man spricht hierbei von der Generalisierungsfähigkeit des Modelles, weitere Betrachtungen hierzu finden sich in Abschnitt 2.6.

Der Lernalgorithmus konstruiert - basierend auf den gezeigten Trainingsdaten - ein Modell M. Dieses Modell liefert für ein Muster $\boldsymbol{x}$ eine Ausgabe $\boldsymbol{y}$ aus der Menge aller möglichen Klassen Y. Das Modell M kann auch als eine Funktion $f : X \rightarrow Y, f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{y}$ aufgefasst werden, mit $\boldsymbol{w}$ als Parameter-Vektor des Modells aus dem Parameter-Raum W. Diese Parameter werden während dem Lernprozess angepasst, so dass die gewünschte Funktion approximiert wird. Die Parameter entsprechen den Informationen, welche der Lernalgorithmus aus den Trainingsdaten extrahiert hat. Handelt es sich beispielsweise bei einem Klassifikator um eine lineare Gerade im 2-dimensionalen Raum, sind die Parameter des Modells die Steigung und der y-Achsen Abschnitt. Für alle Muster (Koordinaten im

2-dimensionalen Raum) auf der einen Seite der Gerade ergibt sich dann ein positiver Wert für die Klassifikation, für die anderen Muster ein negativer Wert.

Lernen bedeutet also, ein Modell zu finden, welches die Relation von Eingaben $\boldsymbol{x}_i$ und den dazugehörigen Ausgaben $\boldsymbol{y}_i$ möglichst gut approximiert. Der generelle Ansatz beim maschinellen Lernen besteht darin, den Lernprozess als ein Optimierungsproblem aufzufassen. In den folgenden Abschnitten werden die für diese Arbeit relevanten Ansätze zum Lernen kurz vorgestellt. Diese Ansätze basieren auf unterschiedlichen Ideen, wie der Lernprozess aufgefasst werden kann.

2.1. Statistische Methoden: Bayes'sches Lernen

Das Ziel des statistischen Lernens ist es, Muster aufgrund statistischer Informationen so zu klassifizieren, dass die Wahrscheinlichkeit einer Falschklassifikation möglichst gering ist. Im Folgenden sollen verschiedene Begriffe aus der statistischen Lerntheorie kurz eingeführt werden, welche in dieser Arbeit Verwendung finden.

Die Fähigkeit des Klassifikators, für neue Muster die korrekte Klasse vorherzusagen, hängt von Informationen ab, die im Vorfeld mit zufällig gezogenen Mustern gesammelt wurden. Hierbei wird in der statistischen Lerntheorie von Wahrscheinlichkeiten für die Ausprägungen der Muster und deren Klasse gesprochen. So beschreibt die *a priori* Wahrscheinlichkeit $P(y_c)$ für die Klasse y_c die Anzahl n_{y_c} Beobachtungen von Klasse y_c bezüglich einer unendlichen Anzahl N von Beobachtungen:

$$P(y_c) = \lim_{N \to \infty} \frac{n_{y_c}}{N},\ c = 1, \ldots, m \tag{2.1}$$

welche für eine große Anzahl N von Beachtungen approximiert werden kann mit:

$$\hat{P}(y_c) = \frac{n_{y_c}}{N} \tag{2.2}$$

Dieser Wert beschreibt die (unbedingte) Wahrscheinlichkeit für das Auftreten von Klasse y_c ohne weitere Informationen über das zu klassifizierende Muster. Für eine bessere Übersichtlichkeit nehmen wir im Folgenden an, dass nur zwei Klassen y_0 und y_1 existieren. Die folgenden Betrachtungen lassen sich aber auch auf Mehrklassenprobleme erweitern. Um eine Klassifikationsentscheidung herbeizuführen, muss das Risiko der Falschklassifikation minimiert werden. Das Risiko einer Falschklassifikation ist $P(y_1)$, wenn wir uns für y_0 entscheiden und umgekehrt $P(y_0)$, wenn wir uns für y_1 entscheiden. Um das Risi-

ko einer Falschklassifikation zu minimieren, muss man also folgende Klassifikationsregel anwenden:

$$f(\boldsymbol{x}) = \begin{cases} y_0, & \text{falls } P(y_0) > P(y_1) \\ y_1 & \text{sonst} \end{cases} \tag{2.3}$$

Die Klassifikation mit der Klassenausprägung, welche die höchste Wahrscheinlichkeit besitzt, führt dazu, dass der Klassifikationsfehler möglichst gering ist.

Üblicherweise werden bei einer Klassifikation auch die Ausprägungen der Muster mit in Betracht gezogen, um die Klassifikationsgenauigkeit zu verbessern. Die Verteilung der Zufallsvariable $\boldsymbol{x}$ (in unserem Fall ein mehrdimensionaler Feature-Vektor) bezüglich einer Klasse y_c wird als klassenbedingte Wahrscheinlichkeitsdichtefunktion $p(\boldsymbol{x}|y_c)$ beschrieben. Sie wird auch als *likelihood* der Klasse y_c bezüglich der Ausprägung von $\boldsymbol{x}$ bezeichnet.

Die Wahrscheinlichkeit für das gemeinsame Auftreten von Klasse y_c mit Muster $\boldsymbol{x}$ wird mit der Dichtefunktion $p(y_c, \boldsymbol{x})$ modelliert. Um diese Dichtefunktion zu berechnen, benötigen wir zwei weitere Definitionen: Zum einen die bedingte *a posteriori* Wahrscheinlichkeit $P(y_c|\boldsymbol{x})$, welche die Wahrscheinlichkeit beschreibt, dass die Klassenausprägung y_c für ein gegebenes Muster $\boldsymbol{x}$ auftritt. Weiterhin benötigen wir die Definition der unbedingten Wahrscheinlichkeitsdichteverteilung für $\boldsymbol{x}$, welche mit $p(\boldsymbol{x})$ bezeichnet wird. Sie beschreibt die Häufigkeit der Ausprägung des Musters $\boldsymbol{x}$ und ergibt sich aus der Summe aller Klassenausprägungen y_c des Produktes von *likelihood* für y_c und *a priori* Wahrscheinlichkeit für y_c:

$$p(\boldsymbol{x}) = \sum_c p(\boldsymbol{x}|y_c)P(y_c) \tag{2.4}$$

Nach den Gesetzen der Wahrscheinlichkeitsrechnung (Cios u. a., 1998, Appendix B) gelten folgende Zusammenhänge:

$$\begin{aligned} p(y_c, \boldsymbol{x}) &= P(y_c|\boldsymbol{x})p(\boldsymbol{x}) \\ p(y_c, \boldsymbol{x}) &= p(\boldsymbol{x}|y_c)P(y_c) \end{aligned} \tag{2.5}$$

die sich durch Umstellung in das Bayes'sche Theorem umwandeln lassen:

$$P(y_c|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y_c)P(y_c)}{p(x)} \tag{2.6}$$

Die Besonderheit des Bayes'schen Theorems liegt darin, dass die Bedingungen in den Wahrscheinlichkeiten umgedreht werden können. Somit kann die bedingte *a posterio-*

ri Wahrscheinlichkeit $P(y_c|\boldsymbol{x})$ mit der *a priori* Wahrscheinlichkeit $P(y_c)$ und der klassenbedingten Wahrscheinlichkeitsdichtefunktion (*likelihood*) $p(\boldsymbol{x}|y_c)$ beschrieben werden. Sowohl $P(y_c)$ als auch $p(\boldsymbol{x}|y_c)$ können mit Hilfe von Trainingsdaten geschätzt werden.

In diesem Abschnitt wurde über die Definition von Wahrscheinlichkeitsverteilungen eine konkrete Definition für die Wahrscheinlichkeit der Klasse für ein gegebenes Muster hergeleitet. Diese Klassifikationsregel basiert auf der Idee, den Fehler der Klassifikation zu minimieren. Im nächsten Abschnitt wird diese Idee genauer betrachtet.

2.2. Lernen als Fehlerminimierung

Eine Möglichkeit, den Lernprozess als Optimierungsproblem zu formulieren, besteht darin, eine Funktion zu definieren, welche den Fehler des Lernalgorithmus auf den Trainingsdaten misst. Über die Definition der Fehlerfunktion wird dann eine Regel abgeleitet, um diesen Fehler zu minimieren. Allgemein wird der Fehler auf den Trainingsdaten $E(M, D_{\text{train}})$ folgendermaßen definiert:

$$E(M, D_{\text{train}}) = \frac{1}{n} \sum_{i=1}^{n} E_i(M, \boldsymbol{x}_i) \tag{2.7}$$

$E_i(M, \boldsymbol{x}_i)$ ist der Fehler des Modells M für das Muster $\boldsymbol{x}_i$. $E(M, D_{\text{train}})$ entspricht also dem mittleren Fehler über alle Muster aus den Trainingsdaten und hängt damit sowohl von den Trainingsdaten als auch dem Modell M ab. Sei $y(\boldsymbol{x}_i)$ die Ausgabe für das Muster $\boldsymbol{x}_i$ in den Trainingsdaten D_{train} und $\hat{y}(\boldsymbol{x}_i)$ die Ausgabe des aktuellen Modells. Weit verbeitete Fehlermaße im Bereich der Regression (numerisches Klassenattribut) sind:

$$\text{L1-Fehler: } E_i(M, \boldsymbol{x}_i) = |\hat{y}(\boldsymbol{x}_i) - y(\boldsymbol{x}_i)| \tag{2.8}$$

$$\text{L2-Fehler: } E_i(M, \boldsymbol{x}_i) = (\hat{y}(\boldsymbol{x}_i) - y(\boldsymbol{x}_i))^2 \tag{2.9}$$

Der L1-Fehler misst die absolute Differenz zwischen der Ausgabe in den Trainingsdaten und der Ausgabe des Lerners, beim L2 Fehler wird diese Differenz quadriert, so dass größere Abweichungen stärker bestraft werden.

Im Bereich der Klassifikation (nominales Klassenattribut) kann der Lernalgorithmus, wie im vorherigen Abschnitt beschrieben, für jede mögliche Klasse eine Klassenwahrscheinlichkeit $\hat{P}(y_c|\boldsymbol{x}) \in [0, 1]$ ausgeben. Ist die genaue Wahrscheinlichkeit nicht bekannt, so liegt die Klassifikation in $\{0, 1\}$. Sei $P(y_c|\boldsymbol{x})$ die wahre bzw. gewünschte Wahrschein-

lichkeit für das Auftreten von Klasse y_c für das Muster $\boldsymbol{x}$. Folgende Fehlermaße werden in der Klassifikation oft verwendet:

$$0/1 \text{ loss: } E_i(M, \boldsymbol{x}_i) = \sum_{y_c \in Y} P(y_c|\boldsymbol{x})(1 - \delta(y_c, \arg\max_{y'_c \in Y} \hat{P}(y'_c|\boldsymbol{x})) \tag{2.10}$$

$$\log \text{ loss: } E_i(M, \boldsymbol{x}_i) = -\sum_{y_c \in Y} P(y_c|\boldsymbol{x}) log(\hat{P}(y_c|\boldsymbol{x})) \tag{2.11}$$

Die delta-Funktion $\delta(a, b)$ ist 1 für $a = b$ und 0 sonst. Der 0/1 loss ist demnach 0 für ein richtig klassifiziertes Beispiel und 1 für ein falsch klassifiziertes Beispiel. Beim log-loss hängt die Größe des Fehlers vom Grad der Abeichung zwischen wahrer Ausgabe und Modell-Ausgabe ab. Der Fehlerwert wächst mit zunehmender Differenz zwischen Modell-Ausgabe und wahrer Ausgabe entprechend der Logarithmus-Funktion.

Ist die Fehlerfunktion E im Parameter-Raum W des Modells M differenzierbar, kann man versuchen, das Minimum von E zu finden. Beispielsweise kann man sich die Fehlerfunktion als eine Oberfläche im Parameter-Raum W vorstellen. Das aktuelle Modell stellt einen Punkt auf dieser Oberfläche dar. Um das Minimum der Fehlerfunktion zu finden, wird dieser Punkt in die Richtung des steilsten Abstieges bewegt. Diese Methode ist als Gradientenabstieg bekannt und ist eine der populärsten Methoden zur Minimierung einer Funktion. Der Lernprozess kann gestoppt werden, wenn ein Minimum der Fehlerfunktion gefunden wurde. Da die Fehlerfunktion jedoch eine nicht-lineare Funktion der Parameter $\boldsymbol{w}$ ist, kann nicht garantiert werden, dass dieses lokale Minimum gleichzeitig das gewünschte globale Minimum in der Fehlerfunktion ist.

2.3. Lernen als Maximum-Likelihood Schätzung

Im vorherigen Abschnitt wurde das Lernen aus der Perspektive der Fehlerminimierung vorgestellt. Der Ausgangspunkt hierbei ist die Definition einer geeigneten Fehlerfunktion. Das Ergebnis ist ein - gemäß der Fehlerfunktion optimaler - Parameter-Vektor $\boldsymbol{w}'$. Ein anderer Ansatz des Lernens ist die Maximum-Likelihood Schätzung. Mit $p(\boldsymbol{y}, \boldsymbol{x})$ haben wir die Wahrscheinlichkeitsdichteverteilung über dem Eingabe/Ausgaberaum Z beschrieben. Bei der Maximum-Likelihood Schätzung wird angenommen, dass es sich dabei um die Wahrscheinlichkeit handelt, dass die Trainingsdaten durch ein Modell M mit den Parametern $\boldsymbol{w}$ generiert wurden. Die Lernaufgabe besteht darin, diesen Parametervektor aus den Trainingsdaten zu schätzen. Diese Aufgabe kann auch wieder als

Optimierungsproblem aufgefasst werden: es wird der Parameter-Vektor $\boldsymbol{w}'$ gesucht, welcher mit der höchsten Wahrscheinlichkeit die Trainingsdaten generiert hat. Nimmt man an, dass die Ereignisse statistisch unabhängig sind, so lässt sich die Wahrscheinlichkeitsfunktion über die Einzelwahrscheinlichkeiten $P(M, \boldsymbol{x}_i)$, dass ein Muster $\boldsymbol{x}_i$ durch das Modell M erzeugt wurde, faktorisieren:

$$p(M, X) = \prod_{i=1}^{n} P(M, \boldsymbol{x}_i) \tag{2.12}$$

Bei der Maximum-Likelihood Methode nehmen wir an, dass die Trainingsdaten D_{train} gegeben sind und suchen den Parametervektor $\boldsymbol{w}'$ des Modells M, dessen Wert am wahrscheinlichsten diese Trainingsdaten erzeugt hat.

Um zu betonen, dass $p(M, D_{\text{train}})$ eine Funktion von $\boldsymbol{w}$ für eine gegebene fixe Trainingsmenge D_{train} ist, wird $p(M, D_{\text{train}})$ auch als *Likelihood* bezeichnet. Durch Ableitung nach w und Nullsetzen der Gleichung wird die Funktion optimiert. Oft ist es zur Berechenbarkeit einfacher, die Likelihood Funktion zu logarithmieren, so dass aus dem Produkt eine Summe wird. Dies ist zulässig, da die Logarithmusfunktion monoton und stetig ist, somit wirkt sich die Differenzierung nach den Parametern nicht aus. Formal lässt sich die Suche nach dem optimalen Parameter-Vektor $\boldsymbol{w}'$ folgendermaßen beschreiben:

$$\boldsymbol{w}' = \arg\max_{\boldsymbol{w}} \log p(M, D_{\text{train}}) = \arg\max_{\boldsymbol{w}} \sum_{i=1}^{n} \log P(M, \boldsymbol{x}_i) \tag{2.13}$$

Über die Definition eines Wahrscheinlichkeitsmaßes $P(M, \boldsymbol{x}_i)$ lässt sich so der optimale Parameter-Vektor finden. Beispielsweise kann man annehmen, dass die wahre Ausgabe $y(\boldsymbol{x})$ durch additives Rauschen verzerrt ist, welches sich durch eine Gaußverteilung ϵ beschreiben lässt:

$$y(\boldsymbol{x}) = \hat{y}(\boldsymbol{x}) + \epsilon \tag{2.14}$$

Dann lässt sich $p(M, D_{\text{train}})$ auch über eine Gaußverteilung beschreiben und durch Logarithmieren erhält man:

$$\log p(M, D_{\text{train}}) \propto -\sum_{i=1}^{n} (\hat{y}(\boldsymbol{x}) - y(\boldsymbol{x}))^2 \tag{2.15}$$

Vergleicht man diesen Ausdruck mit dem quadratischen Fehler in Formel 2.9, so lässt sich

feststellen, dass die Maximierung des log-Likelihoods von einem Klassifikationsmodell mit additivem Rauschen, welches durch eine Gaußverteilung beschrieben wird, äquivalent zur Minimierung des quadratischen Fehlers ist.

2.4. Bayes'sches Lernen im Parameterraum

Im vorherigen Abschnitt basierte das Lernen auf einer Wahrscheinlichkeitsverteilung $p(\boldsymbol{y}, \boldsymbol{x})$ über dem Eingabe/Ausgaberaum Z. Das Ziel der Fehlerminimierung und der Maximum-Likelihood Schätzung war es, einen optimalen Parametervektor $\boldsymbol{w}$ zu finden. Beim Bayes'schen Lernen werden diese Ansätze erweitert, indem man auf dem Parameter-Raum W eine Wahrscheinlichkeitsverteilung annimmt. Es gibt nicht mehr nur noch einen Parameter-Vektor, der gefunden werden muss. Stattdessen wird eine Ansammlung von Parameter-Vektoren betrachtet. Das Wahrscheinlichkeitsmaß im Parameter-Raum beschreibt die Wahrscheinlichkeiten für verschiedene Parameter-Vektoren $\boldsymbol{w}$.

Wir nehmen erneut an, dass die Daten unabhängig und identisch verteilt sind. Dann lässt sich für die Trainingsdaten ein Produkt von Datenmodellen für die einzelnen Trainingsmuster aufstellen. Wenn ein Trainingsmuster $\boldsymbol{x}_i$ betrachtet wird, so ändert sich auch die Wahrscheinlichkeitsverteilung im Parameter-Raum, da manche Parameter-Vektoren mit dem Beispiel konsistenter sind als andere. Dies wird mit dem Bayes-Theorem als eine *a-posteriori* Verteilung im Parameter-Raum beschrieben:

$$p(\boldsymbol{w}|D_{\text{train}}) \propto \prod p(\boldsymbol{x}_i|\boldsymbol{w})p(\boldsymbol{w}) \tag{2.16}$$

Die *a-priori* Verteilung $p(\boldsymbol{w})$ ist die Verteilung über W, wenn noch kein Muster aus den Trainingsdaten beobachtet worden ist. Da noch keine Informationen verfügbar sind, wird dies über eine möglichst breite Verteilung beschrieben. Nachdem jeweils ein Muster aus den Trainingsdaten beobachtet worden ist, werden manche Parameter-Vektoren $\boldsymbol{w}$ konsistenter mit diesem Muster sein als andere. Die *a-posteriori* Verteilung wird dann im Lernprozess schmaler als die *a-priori* Verteilung. Dieser Lernfortschritt lässt sich z. B. durch die Abnahme der Shannon-Entropie (Shannon, 2001) beschreiben, welche in Abschnitt 3.4.3 noch genauer erläutert wird.

2.5. Reduktion des Versionsraumes

Der Parameter-Raum W wird auch als Hypothesenraum H bezeichnet. (Mitchell, 1982) definiert den Versionsraum als eine Untermenge aus dem Hypothesenraum H, die mit den Daten konsistent sind:

$$V = \{h \in H | \forall i \in \{1, \ldots, n\}\ \boldsymbol{y} = h(\boldsymbol{x}_i)\} \tag{2.17}$$

Ein Lernproblem kann folglich als Suche nach der besten Hypothese im Versionsraum aufgefasst werden. Für eine Lernstrategie im Versionsraum spielt die partielle Ordnung eine wichtige Rolle. Eine Hypothese h_1 wird als „genereller“ oder gleich hinsichtlich einer Hypothese h_2 bezeichnet (bzw. umgekehrt h_2 spezieller als h_1), in Zeichen $h_1 \geq h_2$, wenn gilt:

$$\forall \boldsymbol{x} \in X : h_2(\boldsymbol{x}) = 1 \Rightarrow h_1(\boldsymbol{x}) = 1 \tag{2.18}$$

h_1 ist streng genereller als h_2, in Zeichen $h_1 > h_2$ genau dann, wenn gilt:

$$(h_1 \geq h_2) \wedge (h_2 \ngeq h_1) \tag{2.19}$$

Um die verschiedenen Hypothesen im Versionsraum zu begrenzen, existieren zwei Mengen S und G: S ist die Menge der speziellsten Hypothesen, also die Menge der Hypothesen, welche keine Hypothesen mehr enthalten, die genereller sind:

$$S = \{h \in H, \nexists h' \in H, h > h'\} \tag{2.20}$$

Analog dazu ist G die Menge der generellsten Hypothesen die Menge, für die keine weitere speziellere Hypothese existiert:

$$G = \{h \in H, \nexists h' \in H, h' > h\} \tag{2.21}$$

Zu Beginn des Lernprozesses enthält S anfangs die speziellste Hypothese, die jedes Zielkonzept negativ klassifiziert, und G die allgemeinste Hypothese, die jedes Zielkonzept positiv klassifiziert. In den nachfolgenden Lerniteration wird über alle Trainingsdaten iteriert und S und G jeweils so angepasst, dass die oben genannten Bedingungen erfüllt sind. Sobald $S = G$ gilt, kann der Lernprozess abgebrochen werden.

2.6. Generalisierung, Bias und Varianz

Die in den vorherigen Abschnitten betrachteten Maße für die Performanz eines Lernalgorithmus basieren auf den Trainingsdaten D_{train}. Unterschiedliche Trainingsdaten führen zu unterschiedlichen Fehlern und unterschiedlichen Modellen. Das Ziel beim Lernen besteht jedoch nicht darin, die Trainingsdaten möglichst korrekt zu klassifizieren[1], sondern eine möglichst gute Performanz auf neuen, ungesehenen Testdaten D_{test} zu erreichen. Die Fähigkeit des Lernalgorithmus, auf bislang ungesehenen Testdaten eine gute Performanz zu erreichen, wird als Generalisierung bezeichnet. Um die Generalisierung zu quantifizieren, lässt sich der Generalisierungsfehler E_{Gen} verwenden, welcher die Wahrscheinlichkeit beschreibt, dass das trainierte Modell eine falsche Ausgabe für ein zufällig gewähltes Beispiel macht:

$$E_{\text{Gen}}(M) = \int_{X \times Y} E_i(M, \boldsymbol{x}) p(\boldsymbol{y}, \boldsymbol{x}) d(\boldsymbol{x}, \boldsymbol{y}) \tag{2.22}$$

$E_i(M, \boldsymbol{x}_i)$ ist hierbei wieder die gewählte Fehlerfunktion. Da die Wahrscheinlichkeitsverteilung für den Eingabe/Ausgaberaum $p(\boldsymbol{y}, \boldsymbol{x})$ unbekannt ist, kann der Ausdruck nicht explizit integriert werden. Somit lässt sich $E_{\text{Gen}}(M)$ nicht bestimmen und kann nicht direkt beim Training verwendet werden, obwohl dieser Ausdruck derjenige ist, der minimiert werden soll. Stattdessen wird das so genannte empirische Risiko (siehe Gleichung 2.7) verwendet, welches sich aus dem arithmischen Mittel des Fehlers auf den Trainingsdaten zusammensetzt. Nach den Gesetzen der Wahrscheinlichkeitsrechnung konvergiert $E(M, D_{\text{train}})$ mit hinreichend großer Anzahl von Mustern n gegen $E_{\text{Gen}}(M)$:

$$E_{\text{Gen}}(M) \equiv E(M, D_{\text{train}}), \lim |D_{\text{train}}| \to \infty \tag{2.23}$$

Neben dem Generalisierungsfehler soll an dieser Stelle die Bias-Varianz Dekomposition des erwarteten Fehlers eines Lernalgorithmus vorgestellt werden. Sie beschreibt die Abhängigkeit eines Modells vom Bias und der Varianz. Rufen wir uns die unbekannte gemeinsame Verteilung $p(\boldsymbol{y}, \boldsymbol{x})$ über $\boldsymbol{x}$ und $\boldsymbol{y}$ und $p(\boldsymbol{x})$ als die Verteilung der Eingabedaten in Erinnerung, $\hat{y}(\boldsymbol{x})$ als die Ausgabe des aktuellen Modells und $y(\boldsymbol{x})$ als die wahre Ausgabe in den Trainingsdaten. Wir betrachten den erwarteten Fehler eines Lerners,

[1] In diesem Fall wäre es völlig ausreichend, eine Tabelle mit den Trainingsdaten zu speichern.

welcher folgendermaßen definiert ist:

$$\int_{\boldsymbol{x}} E_D[(\hat{y}(\boldsymbol{x}) - y(\boldsymbol{x}))^2|\boldsymbol{x}]dx \tag{2.24}$$

wobei $E_D[\cdot]$ den Erwartungswert über $p(\boldsymbol{y}|\boldsymbol{x})$ und Traingsmenge D bezeichnet. Dieser Erwartungswert lässt sich in drei verschiedene Bestandteile (Geman u. a., 1992) aufspalten:

$$\begin{aligned} E_D[(\hat{y}(\boldsymbol{x}) - y(\boldsymbol{x}))^2|\boldsymbol{x}] = E[(y(\boldsymbol{x}) - E[\boldsymbol{y}|\boldsymbol{x}])^2] \\ +(E_D[\hat{y}(\boldsymbol{x})] - E[\boldsymbol{y}|\boldsymbol{x}])^2 \\ +E_D\left[(\hat{y}(\boldsymbol{x}) - E_D[\hat{y}(\boldsymbol{x})])^2\right] \end{aligned} \tag{2.25}$$

Der erste Term auf der rechten Seite der Gleichung ist die Varianz von y gegeben ein Muster $\boldsymbol{x}$ bzw. das Rauschen in der Verteilung und ist nicht abhängig vom Lernalgorithmus oder den Trainingsdaten. An dieser Stelle interessieren wir uns vor allem für die zwei weiteren Terme: Der zweite Term ist der so genannte quadratische Bias; er setzt sich zusammen aus der Differenz des Erwartungswertes des Modells und der tatsächlichen idealen Ausgabe. Er beschreibt die Vorliebe des Lernverfahrens für gewisse Funktionen. Anschaulich ist der Bias groß, wenn das Modell nicht fähig ist, die eigentliche Datenbeziehungen darzustellen. Der dritte Term beschreibt die Varianz des Lernalgorithmus. Die Varianz ist die erwartete Abweichung der Modelle untereinander. Eine große Varianz bedeutet, dass das Modell sehr von den Trainingsdaten abhängt, was bei instabilen Modellerzeugungen der Fall ist.

Will man den erwarteten Fehler minimieren, so müssen also Bias und Varianz minimiert werden. Diese beiden Terme lassen sich jedoch nicht gleichzeitig minimieren, da sie voneinander über die Modellkomplexität abhängig sind. Abbildung 2.1 zeigt das so genannte Bias-Varianz Dilemma. Es lässt sich folgendermaßen beschreiben: Je komplexer ein Modell wird, desto flexibler wird es und desto geringer ist der Bias, d. h. desto besser die Schätzung des Modells. Gleichzeitig steigt die Varianz und das Modell ist nicht mehr in der Lage, die Systematik in den Daten zu repräsentieren (overfitting). Für ein einfaches Modell mit wenigen Parametern gilt dafür umgekehrt, dass die Varianz gering und der Fit schlecht ist, die Daten durch ein solches Modell aber gut generalisiert werden.

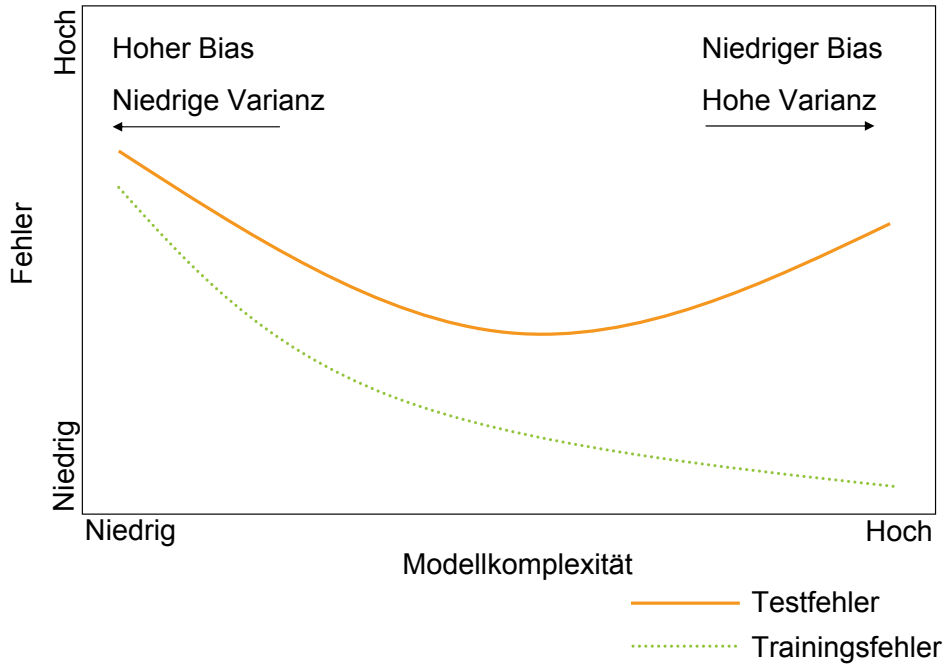


Abbildung 2.1.: Bias-Varianz Dilemma.

2.7. Zusammenfassung

In diesem Abschnitt wurde das Problem des maschinellen Lernens formal eingeführt. Es läst sich feststellen, dass das Lernproblem aus konzeptionell verschiedenen Sichten betrachtet werden kann, die wiederum in manchen Fällen zur selben Lösung führen. Im Speziellen wurde das Lernproblem aus der Sicht der Fehlerminimierung betrachtet, um die Parameter des Modells zu lernen. Hier muss ein geeignetes Fehlermaß zum Lernen ausgewählt werden. Bei der Maximum-Likelihood Schätzung wird der Parametervektor mit der höchsten Wahrscheinlichkeit bezüglich der gegebenen Trainingsdaten gesucht. Mit einer zusätzlichen Wahrscheinlichkeitsverteilung im Modell-Parameter Raum wird beim Bayes'schen Lernen nicht nur ein Parametervektor, sondern ein Ensemble von Parametervektoren gesucht. Dabei wird der durch den Parameterraum definierte Versionsraum in jeder Lerniteration reduziert. Weiterhin wurde aufgezeigt, dass beim Lernen der Generalisierungsfehler minimiert werden soll, der aber wegen seiner Nichtberechenbarkeit durch den Trainingsfehler ersetzt wird. Die Dekomposition des erwarteten Fehlers eines Lernalgorithmus hat das Bias-Varianz Dilemma zwischen einem einfachen und einem komplexen Modell aufgezeigt.

3. Aktives Lernen

3.1. Einleitung

Bislang wurde bei den zuvor beschriebenen Lernstrategien davon ausgegangen, dass es eine Menge von Trainingsdaten D_{train} gibt, mit denen der Lernalgorithmus trainiert wird. Doch wie entstehen diese Daten? Wie in der Einleitung beschrieben, stehen durch Messgeräte, Sensoren und Algorithmen der Merkmalsextraktion oft eine große Menge von nicht klassifizierten Mustern U zur Verfügung. Der Prozess der Klassifikation eines Musters erfordert üblicherweise ein so genanntes Orakel. Das Orakel klassifiziert ein Muster $\boldsymbol{x}_i$, das Resultat ist ein Zweiertupel von Muster und Klassifikation $(\boldsymbol{x}_i, y_i)$. Die Menge der klassifizierten Muster L steht dann dem Lernalgorithmus zum Trainieren zur Verfügung. Der schematische Ablauf ist in Abbildung 3.1 dargestellt.

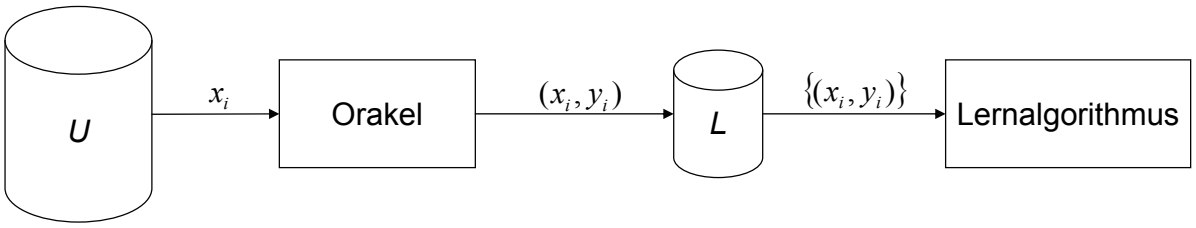


Abbildung 3.1.: Überblick Passives Lernverfahren: Zufällige Auswahl von Daten und Klassifikation durch Orakel.

Diese Abbildung verdeutlicht gleich mehrere Sachverhalte, welche wir genauer betrachten wollen:

- Der Prozess der Datenakquisition ist ein passiver Prozess. Die Muster, welche dem Orakel zur Klassifikation vorgelegt werden, werden nicht durch den Lernalgorithmus bestimmt, sondern zufällig gezogen.
- Damit der Lernprozess statistisch stabil sein kann, muss eine ausreichende Anzahl von Mustern klassifiziert werden. Zusätzlich wird angenommen, dass die Daten

unabhängig und gleichverteilt sind.

- Die Menge der klassifizierten Daten L ist üblicherweise viel kleiner als die Menge der nichtklassifizierten Daten U. Dies kann mehrere Ursachen haben: Zum einen kann die Bestimmung einer Klasse zu einem Muster durch das Orakel sehr aufwändig sein. Wenn das Orakel z. B. ein Mensch ist, so ist dieser nur in der Lage, eine begrenzte Menge von Daten zu klassifizieren. Weiterhin kann die Bestimmung einer Klasse ein sehr zeitaufwändiger Prozess sein, weil beispielsweise zur Bestimmung der Klasse ein aufwändiges Experiment nötig ist.

Greift man den letzten Punkt auf, so wird deutlich, dass in manchen Anwendungsfällen nur eine begrenzte Anzahl von Mustern für den Lernalgorithmus zur Verfügung steht.

Zieht man zufällig Trainingsmuster aus U zur Klassifikation, so fällt der empirische Fehler $E(M, D_{\text{train}})$ (vgl. Gleichung 2.7) monoton und kann mittels einer Potenzfunktion der Form

$$E(M, D_{\text{train}}) = a + \frac{b}{n^{\alpha}} \tag{3.1}$$

beschrieben werden (Duda u. a., 2000). n ist die Anzahl der Muster in der Trainingsmenge, die Parameter a, b und $\alpha \geq 1$ hängen von der Aufgabe und dem zugrunde liegenden Klassifikator ab. Die Form der Fehlerkurve verdeutlicht, dass zu Beginn des Trainierens der Informationsgehalt eines neuen Musters im Mittel noch sehr groß ist. Mit zunehmender Anzahl von Mustern werden neuen Mustern im Mittel nur noch wenig Information entnommen; die Unterschiede im gelernten Modell werden immer geringer.

An dieser Stelle setzen die Verfahren des Aktiven Lernens an: wenn nur wenig klassifizierte Muster zur Verfügung stehen oder die Bestimmung einer Klasse für ein Muster sehr aufwändig ist. Die Grundidee besteht darin, durch gezielte Auswahl von Mustern möglichst viel neue Information für ein Klassifikationsmodell zu erhalten. Dadurch soll der gewünschte maximale Generalisierungsfehler (vgl. Gleichung, 2.22) mit möglichst wenigen Mustern unterschritten werden. Der Unterschied zum klassischen passiven Lernverfahren besteht in der Selektionsfunktion q. Diese Funktion ist für die gezielte Auswahl von Mustern zuständig. Anders als beim passiven Lernen wird in jeder Lerniteration ein Muster aus U durch q ausgewählt, dann durch das Orakel klassifiziert, zur aktuellen Menge der klassifizierten Muster L hinzugefügt und dann der Lernalgorithmus neu trainiert. Jedes neu klassifizierte Muster erzeugt ein neues Modell, und jedes neu erzeugte klassifizierte Muster beeinflusst wiederum die Selektionsfunktion q. Der schematische Ablauf eines aktiven Lernverfahrens ist in Abbildung 3.2 dargestellt.

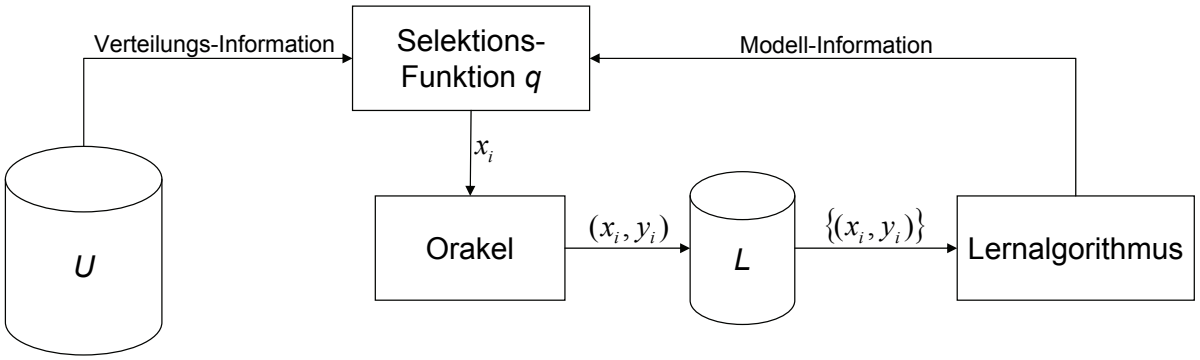


Abbildung 3.2.: Überblick Aktives Lernverfahren: Auswahl mit Selektions-Funktion q, Trainingsbeispiele werden zu L hinzugefügt.

3.2. Selektive Auswahl

Die Selektionsfunktion q ist häufig verknüpft mit einer Bewertungsfunktion $Utility(M, \boldsymbol{x}_u)$, welche den erwarteten Gewinn für eine Klassifikation mit dem Trainingsbeispiel $\boldsymbol{x}_u$ schätzt. Der Selektionsfunktion q stehen zwei Arten von Informationen zur Verfügung: aus den nicht klassifizierten Daten U die Verteilung der Daten und vom Lernalgorithmus das aktuelle Modell, welches auf dem Pool der bislang klassifizierten Daten L beruht.

In der ersten Lerniteration stehen noch keine Modellinformationen zur Verfügung, daher basiert die Selektion von Daten nur auf den unklassifizierten Daten U. Viele aktive Lernalgorithmen wählen zu Beginn zufällig Muster, um ein Modell zu initialisieren. Nur wenige aktive Lernalgorithmen berücksichtigen hierbei auch die Verteilung der Daten. Nach der Initialisierung eines stabilen Modells spielen die Informationen des aktuellen Klassifikationsmodells eine wichtige Rolle. Sie zeigen auf, wo Klassengrenzen (und damit Entscheidungsgrenzen) liegen, und haben einen starken Einfluß auf die Selektion neuer Muster. Wir bezeichnen die erste Phase der Modellinitialisierung als Explorationsphase und die zweite Phase der Modellanpassung als Phase der Verfeinerung bzw. Spezialisierung des Modells (engl.: exploitation).

In Algorithmus 3.1 sind diese beiden Phasen deutlich voneinander getrennt; dies ist zur späteren Unterscheidung der verschiedenen aktiven Lernverfahren erforderlich.

Algorithmus 3.1 Aktives Lernen: Selektive Auswahl

1: $D_{\text{init}} \leftarrow$ Wähle aus U Muster zur Initialisierung des Modells M.
2: **for** $\boldsymbol{x}_u \in D_{\text{init}}$ **do**
3: Frage das Orakel nach der Klasse y des Musters $\boldsymbol{x}_u$.
4: Füge $(\boldsymbol{x}_u, y)$ zur Menge der Trainingsdaten L hinzu.
5: Trainiere ein neues Modell M.
6: **end for**
7: **for** N Iterationen **do**
8: Wähle $\arg\max_{\boldsymbol{x}_u \in U} \mathit{Utility}(M, \boldsymbol{x}_u)$.
9: Frage das Orakel nach der Klasse y von $\boldsymbol{x}_u$.
10: Füge $(\boldsymbol{x}_u, y)$ zur Menge der Trainingsdaten L hinzu.
11: Trainiere ein neues Modell M.
12: **end for**

3.3. Performanz und Komplexität

Damit sich der Einsatz eines aktiven Lernalgorithmus lohnt, sollte der Aufwand für die Selektion von Mustern und das erneute Lernen eines Modells gering sein, da das Orakel beim aktiven Lernen interaktiv in den Klassifikationsprozess eingebunden ist.

Weiterhin sollte die Performanz besser sein als bei einem Algorithmus, der auf zufällig gezogenen Mustern trainiert wird. Dies kann jedoch nicht garantiert werden. Gemäß dem „no free lunch“-Theorem von (Wolpert und Macready, 1997) ist kein Klassifikator besser als ein anderer. Ein Lernprozess ist ohne vorherige Annahmen über das Problem nicht möglich. Die bessere Performanz eines Lernalgorithmus hängt von der Art des Problems ab. Abbildung 3.3 verdeutlicht diesen Sachverhalt. Es kann also durchaus passieren, dass eine zufällige Auswahl von Mustern eine bessere Performanz erzeugt als die aktive Selektion von Daten.

Betrachten wir an dieser Stelle ein positives Beispiel, um das Potential des Aktiven Lernens aufzuzeigen: das Auffinden einer Entscheidungsgrenze auf einem Einheitsintervall (Abbildung 3.4). Der Hypothesenraum $H = \{h_g : g \in \mathbb{R}\}$ wird beschrieben durch Schwellwertfunktionen

$$h_g(x) = \begin{cases} +, & \text{falls } x \geq g \\ - & \text{sonst} \end{cases} \tag{3.2}$$

Es soll eine Hypothesen-Klasse gewählt werden, welche die Datenpunkte korrekt klassifiziert. Um die Grenze g mit einer Genauigkeit ϵ auf dem Intervall zu finden, die beide

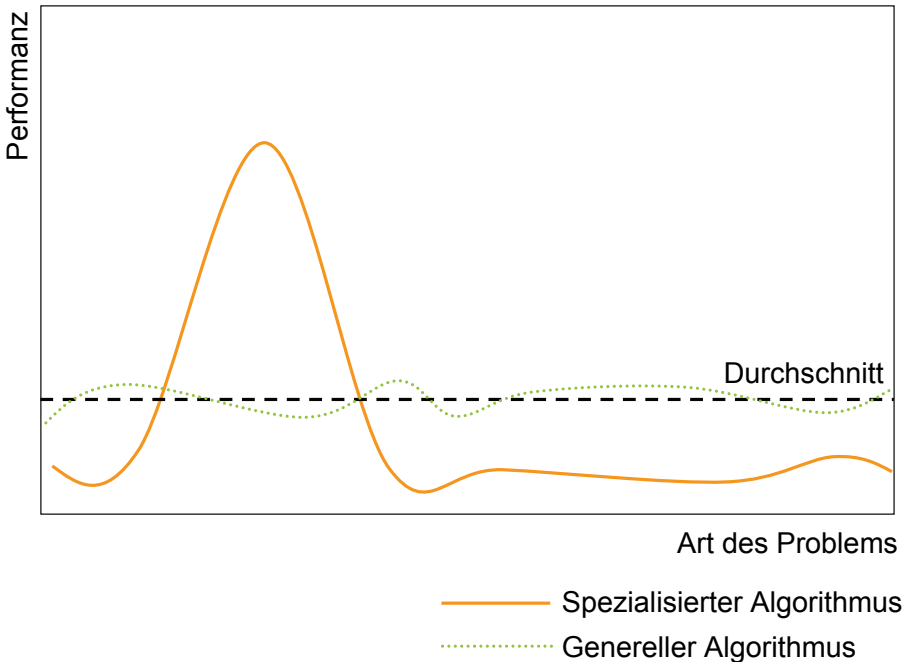


Abbildung 3.3.: No Free Lunch Theorem (frei übernommen aus (Sewell, 2008)).

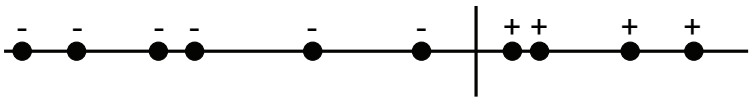


Abbildung 3.4.: Muster auf einem Zahlenstrahl mit den Klassen + und - und der Entscheidungsgrenze.

Klassen voneinander trennt, muss man $m = O(\frac{1}{\epsilon})$ Muster zufällig ziehen und anschließend klassifizieren.

Wenn man jedoch die unklassifizierten Muster verwenden kann und bei jedem Muster nach der Klasse fragt, lässt sich eine binäre Suche durchführen. Hierzu werden nur $\log m$ Muster benötigt, da von den bisherigen Labeln auf die restlichen Label geschlossen werden kann. Dies entspricht einer exponentiellen Verbesserung bezüglich der Anzahl der Labels, die benötigt werden.

Dieses Beispiel wird in der Arbeit von (Dasgupta, 2004) aufgegriffen und für den Fall mit mehreren Dimensionen untersucht. Dabei wird gleich zu Beginn der Arbeit mit Hilfe eines Gegenbeispiels aufgezeigt, dass bei manchen Hypothesen im schlechtesten Fall alle Datenpunkte aus dem Eingaberaum klassifiziert werden müssen, eine aktive

Lernstrategie gegenüber einer zufälligen Auswahl von Mustern also keinen Vorteil hat.

In diesem Beispiel besteht die Hypothesen-Klasse H aus linearen Separatoren in $\mathbb{R}^2$. Eine Menge von m Datenpunkten liegt auf der Kreislinie eines Einheitskreises. Dann existieren Hypothesen, welche nur dann identifiziert werden können, wenn alle m Muster klassifiziert werden. Als Beweis wird folgende Klassenkonfiguration angegeben:

$$L_i(1 \leq i \leq m) : \text{Alle Muster sind negativ, außer } \boldsymbol{x}_i. \tag{3.3}$$

Abbildung 3.5 verdeutlicht dieses Beispiel für die Klassenkonfiguration L_3.

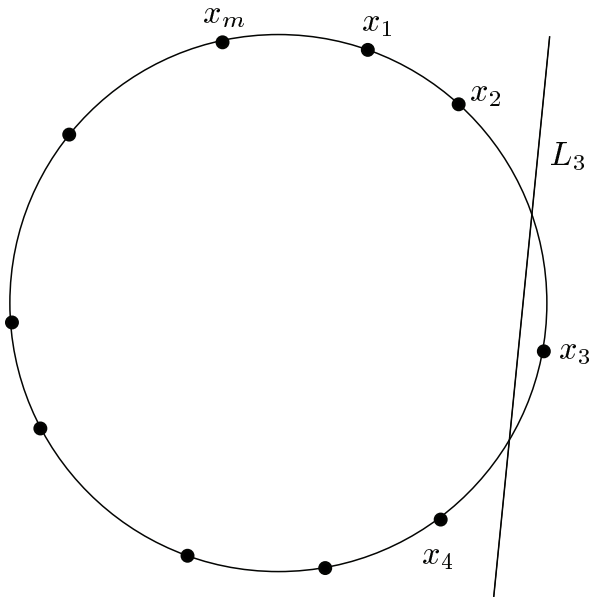


Abbildung 3.5.: Label-Konfiguration, welche nur mit m Abfragen gefunden werden kann (entnommen aus (Dasgupta, 2004)).

Je nach Hypothese und der Verteilung der Daten kann also eine aktive Lernstrategie im idealen Fall die Komplexität einer perfekten binären Suche oder im schlechtesten Fall alle Muster zur Abfrage benötigen. Dies entspricht einem exponentiellen Unterschied zwischen Idealfall und schlechtestem Fall. (Dasgupta, 2004) beschränkt sich daher auf die Analyse im Durchschnittsfall, in dem die Hypothesen gleichmäßig verteilt sind.

In der Arbeit von (Dasgupta, 2004) wird eine populäre *greedy*-Strategie eingesetzt, welche den aktuellen Versionsraum in zwei möglichst gleich große Teile aufspaltet. Der entscheidende Beitrag in dieser Arbeit besteht in dem Nachweis, dass wenn die Hypothesen gleichverteilt sind, die erwartete Anzahl von benötigten Mustern mit dieser

greedy-Strategie höchstens $O(\ln |H|)$ mal so hoch ist wie die erwartete Anzahl für jede andere Strategie. Dies ist eine signifikante Garantie für die Performanz vieler aktiver Lernansätze, die eine ähnliche Strategie zur Selektion von neuen Mustern anwenden. Es ist jedoch keine Garantie dafür, dass die Anzahl der Muster möglichst gering ist. Bei einem Datensatz mit Rauschen wird diese Strategie vor allem die verrauschten Muster selektieren. Dies kann dazu führen, dass die Aufteilung des Versionsraumes überhaupt keine Verbesserung für das Lernen der Zielhypothese bringt.

3.4. Kategorisierung Aktiver Lernverfahren

Allgemein werden verschiedene Formen des Aktiven Lernens unterschieden:

1. Der *query construction* Ansatz (Angluin, 1988) stellt es dem Lerner frei, nützliche Muster selbst zu konstruieren und nach deren Klasse zu fragen. Dies ist jedoch nicht immer sinnvoll oder möglich, da das Orakel nicht unbedingt für jedes künstlich erzeugte Muster eine sinnvolle Klasse zuordnen kann (siehe auch (Lang und Baum, 1992)). Weiterhin berücksichtigt dieser Ansatz nicht die Verteilung der Daten, so dass sich eine Generalisierung als schwierig gestalten kann.
2. Der *selective sampling*-Ansatz wählt aus den nicht klassifizierten Daten U Muster aus. Hier wird weiterhin zwischen zwei Formen unterschieden:
   a) Beim so genannten *stream-based* oder *online active learning* muss der Aktive Lerner aus einem kontinuierlichen Datenstrom ein Muster zur Klassifikation auswählen.
   b) Die meisten Verfahren im Aktiven Lernen lassen sich der Kategorie des *pool based active learning* zuordnen. Hierbei wird in jeder Iteration eine bestimmte Anzahl von Mustern aus einem Pool von Daten ausgewählt und auf den klassifizierten Trainingsdaten ein neues Modell M gelernt.

Wir beschäftigen uns in dieser Arbeit vor allem mit der Form des *pool based active learning*. Die größten Unterschiede der aktiven Lernansätze in diesem Bereich basieren auf der Strategie, mit der die Selektionsfunktion q entworfen wurde. In den folgenden Abschnitten werden verschiedene aktive Lernverfahren vorgestellt, die sich nach ihrer Strategie in folgende Kategorien einordnen lassen:

1. Optimierung einer Zielfunktion: Wie bereits in den Abschnitten 2.2 und 2.3 angesprochen, lässt sich für die Minimierung eines erwarteten Fehlers eines Lernalgorithmus oder zur Maximierung der Likelihood-Funktion eine Zielfunktion aufstellen,

die maximiert werden soll. Über die Definition geeigneter Zielfunktionen lassen sich Kriterien für die optimale Auswahl von Mustern aufstellen.

2. Reduktion des Versionsraumes: Hier wird der Versionsraum des aktuellen Modells betrachtet (siehe Abschnitt 2.4 und 2.5). Die Strategien basieren darauf, den Versionsraum in jeder Lerniteration möglichst stark zu verkleinern.
3. *Uncertainty Sampling*: Diese Heuristik beschränkt sich darauf, direkt an den Klassifikationsgrenzen nach neuen Mustern zu fragen.
4. Aktives Lernen und Clustering: Ansätze, die das aktive Lernen und Clustering miteinander kombinieren, sind bislang noch selten. Hier wird vor allem ein Verfahren vorgestellt, welches das Clustering zur Schätzung der Verteilung der Eingabedaten verwendet.
5. Meta-Heuristiken: Von der Kombination verschiedener aktiver Lernverfahren verspricht man sich eine Verbesserung der Performanz. Hier werden verschiedene Meta-Heuristiken vorgestellt, die auf den zuvor behandelten Algorithmen basieren.

3.4.1. Optimierung einer Zielfunktion

In diesem Abschnitt beschäftigen wir uns mit Ansätzen, die das Aktive Lernen als ein Optimierungsproblem auffassen. Diese Ansätze definieren ein Maß, welches den Nutzen eines Musters für eine Klassifikation durch das Orakel misst.

In einer der ersten Arbeiten auf dem Gebiet des Aktiven Lernens von (MacKay, 1992) werden verschiedene Zielfunktionen für eine aktive Selektion von Daten für ein Interpolationsproblem vorgeschlagen, die auf der Maximierung des Informationsgewinnes beruhen. Zwei verschiedene Maße aus der Informationstheorie werden verwendet, welche die Veränderung der Wahrscheinlichkeitsverteilung im Parameter-Raum W (vgl. Abschnitt 2.4) vor und nach Erhalt eines Musters beschreiben: die Veränderung der Entropie und die Kreuzentropie. Anschaulich beschreiben diese Maße wie sehr die Wahrscheinlichkeitsverteilung im Parameter-Raum W nach Erhalt eines neuen Musters schrumpft (Veränderung der Entropie) bzw. wie sehr sich die Wahrscheinlichkeitsverteilung zusätzlich aufgrund eines neuen Musters verschiebt (Kreuzentropie). Es wird nachgewiesen, dass die beiden Maße in Hinsicht auf den Erwartungswert für den Informationsgewinn äquivalent sind.

Die resultierende Selektionsfunktion q, welche den totalen Informationsgewinn für die

Interpolante maximiert, wählt das nächste Muster an der Stelle aus, wo der erwartete Fehler der Interpolante am höchsten ist. Eine potentielle Schwachstelle der Arbeit von (MacKay, 1992) ist die Annahme, dass das Modell der Interpolante in der aktuellen Lerniteration, welches für die Berechnung des totalen Informationsgewinnes verwendet wird, korrekt ist. Ist dies nicht der Fall, können die berechneten Erwartungswerte „die richtige Antwort auf die falsche Frage“ (MacKay, 1992) sein.

Die Arbeit von (MacKay, 1992) ist eine der ersten Arbeiten, die sich mit der selektiven Auswahl von Daten beschäftigt. Da sich die Arbeit nur auf Regressionsprobleme bezieht und die Suche nach Mustern aufwändig ist (quadratischer Aufwand zur Anzahl der Parameter der Interpolante), ist der Einsatz in praktischen Klassifikationsanwendungen beschränkt.

Die Arbeit von (Roy und McCallum, 2001) beschäftigt sich mit der Minimierung des erwarteten Fehlers (siehe Abschnitt 2.2) mit einem Bayes-Klassifikator. Dazu werden die Fehlermaße 0/1 loss und log loss (siehe Gleichung 2.11) verwendet.

Da die wahre Verteilung über die Eingabedaten und die Klassen $p(\boldsymbol{y}, \boldsymbol{x})$ unbekannt ist, muss sie mit der aktuellen Version des Modells geschätzt werden. Jedes unklassifizierte Muster wird mit jeder möglichen Klasse der aktuellen Trainingsmenge L hinzugefügt und der Lernalgorithmus neu trainiert. Dann wird der erwartete Fehler mit dem aktuell gelernten Modell berechnet. Dieser Fehler wird mit der Ausgabe der Klassenwahrscheinlichkeit (basierend auf dem Modell) für diese Klasse gewichtet. Der Fehlerwert für ein Muster wird über alle Klassen gemittelt. Das Muster mit dem kleinsten erwarteten Fehler wird dann für die Klassifikation durch das Orakel gewählt.

Intuitiv lässt sich der Ansatz von (Roy und McCallum, 2001) so beschreiben, dass solche Muster ausgewählt werden, bei denen das aktuell gelernte Modell unsicher bei der Klassifikation ist, es jedoch nach der Klassifikation dieses Musters möglichst stark bestätigt wird.

Da für jedes Muster mit jeder möglichen Klasse der Einfluss auf den Fehler geschätzt wird, ist eine naive Implementierung sehr ineffizient. Jedoch kann die Laufzeit optimiert werden, wenn der Lernalgorithmus inkrementell trainiert werden kann (wie es bei dem in der Arbeit von (Roy und McCallum, 2001) verwendeten Naive Bayes Klassifikator möglich ist) und der Fehler nur mit Mustern in der unmittelbaren Nachbarschaft geschätzt wird.

Die Arbeit von (Roy und McCallum, 2001) konzentriert sich im Weiteren vor allem auf die effiziente Implementierung des Verfahrens im Bereich der Textklassifikation. Sie ist die

konsequente Umsetzung der Minimierung eines erwarteten Fehlers, der mit dem aktuellen Modell geschätzt wird. Genau wie in der zuvor beschriebenen Arbeit von (MacKay, 1992) ist der größte Schwachpunkt die Annahme, dass das aktuelle Modell, mit dem die Schätzung vorgenommen wird, korrekt ist. In beiden Arbeiten wird das initiale Modell mit zufällig gezogenen Mustern initialisiert.

In der Arbeit von (Cohn u. a., 1994b) wird mit einem statistischen Ansatz ein optimales Auswahlkriterium hergeleitet. Wie in Abschnitt 2.6 beschrieben, wird der erwartete Fehler in seine Einzelkomponenten Rauschen, Bias und Varianz zerlegt. In der Arbeit von (Cohn u. a., 1994b) wird angenommen, dass die Lernalgorithmen keinen Bias haben bzw. dass der Bias im Vergleich zum dritten Term - der Varianz des Lerners - vernachlässigbar ist. Folglich wird in der vorliegenden Arbeit versucht, den Fehler des Lerners zu minimieren, indem seine Varianz minimiert wird:

$$\sigma_{\hat{y}}^2 \equiv \sigma_{\hat{y}}^2(x) = E_D\left[(\hat{y}(\boldsymbol{x}) - E_D[\hat{y}(\boldsymbol{x})])^2\right] \tag{3.4}$$

Wird ein neues Eingabemuster selektiert und klassifiziert, ändert sich der Wert von $\sigma_{\hat{y}}^2$. Der Prozess der Minimierung geschieht folgendermaßen: Es wird angenommen, dass ein Schätzwert für das aktuelle $\sigma_{\hat{y}}^2$ existiert. Falls für eine neue Eingabe $\tilde{\boldsymbol{x}}$ die Verteilung $p(\tilde{y}, \tilde{\boldsymbol{x}})$ bekannt wäre, könnte man die neue Varianz mit dem zusätzlichen Muster $\tilde{\boldsymbol{x}}$ schätzen. Da $p(\tilde{y}, \tilde{\boldsymbol{x}})$ nicht bekannt ist, wird sie mit Schätzwerten von Mittelwert und Varianz des Lerners approximiert. Dabei wird für ein Gauß'sches Mixture Modell und ein lokal gewichtetes Regressionsmodell die Berechnung der Varianz in geschlossener Form hergeleitet und ein Kriterium zur Evaluation eines Eingabemusters präsentiert.

Die Berechnung der Varianz in geschlossener Form, wie sie in der Arbeit von (Cohn u. a., 1994b) vorgestellt wird, ist nicht für beliebige Lernverfahren möglich.

In der Arbeit von (Lindenbaum u. a., 2004) wird der erwartete Fehler für einen k-nächste Nachbarn Klassifikator minimiert[1]. Dabei werden selektiv einzelne Muster (oder eine Sequenz von Mustern) gezogen, welche den erwarteten Fehler in jedem Schritt minimieren.

Wird eine Sequenz von Mustern gezogen, existieren zwei Möglichkeiten: Entweder wird der Fehler jedes neuen Musters mit dem aktuellen Modell geschätzt oder die bisher gezogenen Muster gehen in das aktuelle Modell und damit in die Schätzung für den erwarteten Fehler des nächsten Musters mit ein.

[1] Lindenbaum spricht in dieser Arbeit von einem erwarteten Nutzwert. Dessen Definition und Maximierung entsprechen jedoch der Minimierung des 0/1 loss aus Gleichung 2.11.

Für die Ziehung einer Sequenz von Mustern wird - wie in einem Spielbaum - die Interaktion zwischen Lernalgorithmus und Orakel abgebildet. In jeder Runde wird ein Muster aus der Menge der unklassifizierten Daten gezogen und (in der zweiten Variante) die möglichen Antworten des Orakels mit berücksichtigt. Der Spielbaum ist in Abbildung 3.6 dargestellt. Im so genannten k-deep lookahead Algorithmus wird dann rekursiv

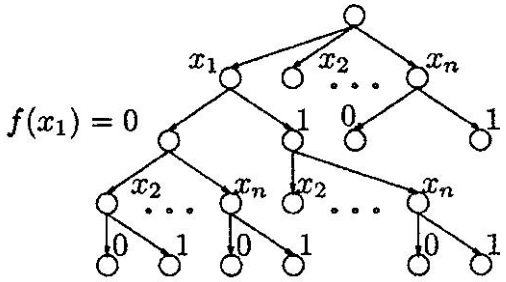


Abbildung 3.6.: Spielbaum mit Daten $x_1, \ldots, x_n$ und Klassenausprägungen $f(x) = 0$ und $f(x) = 1$ (aus (Lindenbaum u. a., 2004)).

die Fehlerfunktion propagiert, um das Muster auszuwählen, welches zu einer Lernsequenz mit dem kleinsten erwarteten Fehler führt. Aufgrund der hohen Komplexität (quadratisch zur Anzahl der Ebenen im Spielbaum) können die Werte nur in begrenzter Tiefe vorberechnet werden.

Um den erwarteten Fehler schätzen zu können, ist vor allem eine Schätzung der Klassenwahrscheinlichkeiten nötig; in dieser Arbeit wird dazu ein so genanntes „random field model“ (Wong und Hajek, 1985) zur Klassifikation verwendet, welches auch Informationen über die Klassenwahrscheinlichkeiten für jedes Muster ausgeben kann. Mathematisch gesehen beschreibt ein „random field model“ die Label der Muster als Zufallsvariablen, welche voneinander abhängig sind. Im Wesentlichen gehen hier die Informationen der klassifizierten Prototypen und die Distanz - welche durch eine Kovarianz-Funktion beschrieben wird - der nicht klassifizierten Datenpunkte zu diesen Prototypen in die Schätzung der Klassenlabel mit ein. Analog zur vorher erwähnten Arbeit von (Roy und McCallum, 2001) wird für jedes unklassifizierte Muster mit jeder möglichen Klassifikation der erwartete Fehler berechnet und das Muster mit dem niedrigsten Fehler zur Klassifikation ausgewählt.

Wird angenommen, dass jedes neu klassifizierte Muster den Fehler des Klassifikators verringert, lässt sich eine weitere Fehlerfunktion aufstellen, indem jeweils ein k-nächster Nachbarn Klassifikator auf den bisher klassifizierten Trainingsdaten L und ein Klassifi-

kator mit dem neuen Muster $L \cup \boldsymbol{x}_i$ betrachtet wird. Für beide Klassifikatoren wird dann der erwartete Fehler berechnet und die Differenz (welche möglichst groß sein sollte) als Kriterium verwendet. Diese offensivere Variante führt zu einer Auswahl von Mustern, welche das aktuelle Modell stark verändern.

Die Arbeit von (Lindenbaum u. a., 2004) zeichnet sich vor allem durch die Idee der vorausschauenden Auswahl von Mustern aus. Leider ist die Anwendung durch die vorausschauende Berechnung und die komplizierte Schätzung der Klassenwahrscheinlichkeiten und Nutzwerte auf 2-Klassen Probleme beschränkt, was bei einem Mehrklassenproblem dazu führt, dass jede Klasse von den anderen Klassen getrennt gelernt werden muss, was zu einer erheblich höheren Anzahl von Mustern führen kann.

Zusammenfassung: Optimierung der Zielfunktion

In diesem Abschnitt haben wir Ansätze für das Aktive Lernen betrachtet, welche basierend auf unterschiedlichen Kriterien - wie z. B. dem erwarteten Fehler - eine Zielfunktion für die Selektion von neuen Daten aufstellen. Aus theoretischer Sicht erleichtert die explizite Definition einer Zielfunktion die Analyse der Stärken und Schwächen eines aktiven Lernverfahrens. Oft werden in solchen Arbeiten Annahmen getroffen, beispielsweise dass der Lernalgorithmus wie in (Cohn u. a., 1994b) keinen Bias hat. In solchen Fällen hängt der Nutzen der Anfrage-Funktion q davon ab, in welchem Ausmaß diese Annahme im Anwendungsfall zutrifft. Die vorgestellten Algorithmen basieren auf der Annahme, dass ein stabiles Modell mit Hilfe von zufällig gezogenen Mustern bereits gelernt wurde.

3.4.2. Reduktion des Versionsraumes

In Abschnitt 2.5 haben wir das Lernproblem als Suche nach der besten Hypothese im Versionsraum aufgefasst. In diesem Abschnitt betrachten wir den Prozess des Aktiven Lernens als Prozess der Verkleinerung des Versionsraumes und stellen verschiedene Ansätze vor, die basierend auf dieser Strategie Daten zur Klassifikation auswählen.

Einer der prominentesten Algorithmen in diesem Bereich ist der Query by Committee-Algorithmus (Seung u. a., 1992). Wie der Name andeutet, wird zunächst ein Kommitee von $2k$ möglichst diversen Hypothesen aufgestellt. In der Arbeit von (Seung u. a., 1992) handelt es sich um Hypothesen, die mit dem Gibbs-Lerner erstellt werden. Ein Gibbs-Lerner wählt zufällig eine Hypothese aus dem Hypothesenraum (also allen möglichen Hypothesen aus dem Versionsraum, die mit den Trainingsdaten konsistent sind, vgl. 2.5)

aus.

Nach der Initialisierung kann dem Query by Committee-Algorithmus ein Strom von Daten angelegt werden. Die Muster werden entweder sofort klassifiziert oder das Orakel wird nach der Klasse des Musters gefragt. Die Auswahl eines Musters zur Klassifikation erfolgt unter der Vorgabe, dass die Entropie der Modell-Parameter nach dem Lernen minimiert werden soll. Ein unklassifiziertes Muster wird dann ausgewählt,wenn k Mitglieder des Komitees das Muster als positiv und k Mitglieder das Muster als negativ klassifizieren, d. h. die Nichtübereinstimmung zwischen den Kommitee-Mitgliedern maximal ist.

Dieser allgemeine Algorithmus wurde in verschiedenen Anwendungsbereichen mit probabilistischen Modellen (Dagan und Engelson, 1995) und speziell mit Naive Bayes Modellen für Textklassifikation (McCallum und Nigam, 1998) verwendet. Der Naive Bayes Klassifikator ermöglicht es, a-priori Wissen mit einfließen zu lassen, was sich als vorteilhaft für die Generierung der verschiedenen Modelle erweist. Der Query by Committee-Algorithmus wurde von (Freund u. a., 1997) auf das Perzeptron erweitert und analysiert. (Liere und Tadepalli, 1997) verwenden den Winnow Algorithmus als Basis-Lerner. Eine Erweiterung des Query by Committee-Algorithmus mit bagging wird im Abschnitt 3.4.5 zu Meta-Heuristiken vorgestellt.

Der Query by Committee Algorithmus gehört zu den so genannten online-Lernverfahren, wird also für Anwendungsfälle eingesetzt, bei denen ein kontinuierlicher Strom von Daten klassifiziert werden muss. Es handelt sich um einen theoretischen Ansatz unter der Annahme, dass kein Rauschen in den Daten existiert und dass Hypothesen aus dem Versionsraum frei zur Verfügung stehen. Unter diesen Voraussetzung weisen (Freund u. a., 1993) nach, dass die Anzahl der benötigten Muster im Verhältnis logarithmisch zur Anzahl der benötigten Muster bei zufälliger Auswahl ist.

Eine der ersten Arbeiten im Bereich des Aktiven Lernens von (Cohn u. a., 1994a) stellt eine weitere aktive Lernstrategie zur selektiven Auswahl von Mustern basierend auf der Idee des Versionsraumes vor.

Die Region der größten Unsicherheit besteht aus der Menge aller Muster $\boldsymbol{x}_i$, für die es zwei Hypothesen gibt, welche zwar konsistent mit den Trainingsdaten sind, aber unterschiedliche Klassen für $\boldsymbol{x}_i$ vorhersagen. Sie wird in dieser Arbeit formal definiert als:

$$R(L) = \{\boldsymbol{x}_i : \exists h_1, h_2 \in V, h_1(\boldsymbol{x}_i) \neq h_2(\boldsymbol{x}_i)\} \tag{3.5}$$

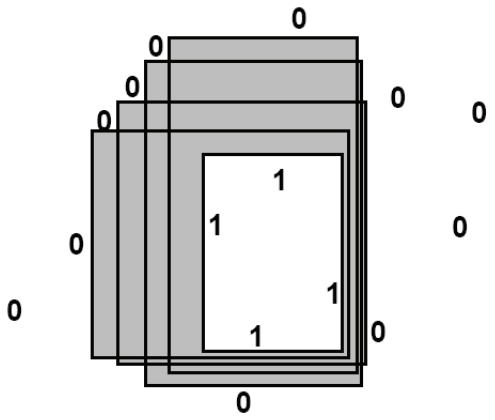


Abbildung 3.7.: Die Region der größten Unsicherheit (grau eingezeichnet) (aus (Cohn u. a., 1994a)).

Alle klassifizierten Muster, welche in diese Region fallen, begrenzen die Größe dieser Region, alle Muster außerhalb dieser Region haben keinen weiteren Einfluss.

Der Ansatz in der Arbeit von (Cohn u. a., 1994a) besteht darin, nach jedem neu klassifizierten Muster die Region der größten Unsicherheit zu berechnen und (zufällig) ein Muster aus dieser Region neu auszuwählen. Da die Berechnung dieser Region sehr aufwändig ist, können pro Iteration auch mehrere Muster ausgewählt werden.

In einem ersten Ansatz wird in der Arbeit von (Cohn u. a., 1994a) ein Multi-Layer-Perzeptron mit einem einzigen Ausgang mit Hilfe des Backpropagation-Algorithmus trainiert. (Cohn u. a., 1994a) setzt die Konfiguration des neuronalen Netzes (d. h. die gelernten Gewichte, nicht die Architektur des Netzes) mit einer Hypothese gleich.[2]

Jede Eingabe $\boldsymbol{x}_i$ wird auf die Ausgabe $[0, 1]$ abgebildet. Bei einem 2-Klassen Problem und einem Ausgabeneuron führt dies zu einer „naiven“ Definition einer Region der größten Unsicherheit. Eine Approximation dieser Region der größten Unsicherheit wären alle Muster, bei denen die Ausgabe beispielsweise zwischen 0.1 und 0.9 liegt, also nicht eindeutig 0 oder 1 ist. Das Problem hierbei ist, dass so nur die Unsicherheit der aktuellen Konfiguration gemessen wird. Dies ist aber nur ein Teil der Region der höchsten Unsicherheit, welche aus allen Unterschieden zwischen allen möglichen Netzkonfigurationen besteht. Verschlimmert wird dies durch den Backpropagation-Algorithmus, welcher dazu

[2]Es wird in der Arbeit nicht näher auf den Sachverhalt eingegangen, dass eine konsistente Hypothese eigentlich **alle** Trainingsdaten korrekt klassifiziert. Ein neuronales Netz kann nicht beliebige Trainingsdaten abbilden, dies ist auch von der Architektur abhängig. Jedoch wird der Fehler auf den Trainingsdaten minimiert.

tendiert, scharfe Klassengrenzen zu ziehen. Daher ist dieser Ansatz sehr anfällig, bei der Initialisierung bestimmte Details im Zielkonzept zu ignorieren.

Ein erweiterter Ansatz in der Arbeit von (Cohn u. a., 1994a) versucht daher, die Region der größten Unsicherheit über die partielle Ordnung im Versionsraum zu definieren. Dabei werden die in Abschnitt 2.5 angesprochenen Mengen der generellsten und speziellsten Hypothesen verwendet. Die Region der größten Unsicherheit liegt dann zwischen der generellsten und der speziellsten Hypothese. Diese Hypothesen werden in der Arbeit von (Cohn u. a., 1994a) mit einem möglichst spezifischen und einem möglichst generellen künstlichen neuronalen Netz aufgrund der bislang klassifizierten Daten gelernt.

Dies geschieht durch einen so gennanten induktiven Bias; dieser Begriff bezeichnet die Auswahlpräferenz einer Hypothese aus dem Versionsraum. In der Arbeit von (Cohn u. a., 1994a) wird das neuronale Netz für alle Muster, die positiv klassifiziert werden, bestraft. Dadurch wird ein sehr spezifisches Konzept gelernt. Die Implementierung eines möglichst generellen neuronalen Netzes erfolgt analog. Die Anfrage-Funktion q beschränkt sich dann darauf, Muster zu selektieren, welche von dem speziellsten und generellsten neuronalen Netz unterschiedlich klassifiziert werden.

In dieser Arbeit von (Cohn u. a., 1994a) wird übrigens der Begriff des Aktiven Lernens zum ersten Mal verwendet und eingeführt. Die Idee eines generellen und speziellen Versionsraumes existiert auch im bekannten Candidate-Elimination Algorithmus von (Mitchell, 1982).

In der Arbeit von (Tong und Koller, 2001) wird eine Strategie basierend auf der Reduktion des Versionsraumes für Support Vector Machines entwickelt. Eine Support Vector Machine (SVM) wird durch eine separierende Hyperebene

$$\boldsymbol{a} \cdot \boldsymbol{x} + b = 0 \tag{3.6}$$

beschrieben, welche die Entscheidungsfunktion

$$f(x) = \text{sign}(\boldsymbol{a} \cdot \boldsymbol{x} + b) \tag{3.7}$$

induziert. Basierend auf den Trainingsdaten $L = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$ wird eine optimale separierende Hyperebene gefunden, welche die Klassen $y \in \{-1, +1\}$ voneinander trennt und deren Spanne zwischen den zwei Klassen maximal ist. Die Einschränkungen der Trainingsdaten für die Hyperebene lassen sich folgendermaßen formu-

lieren:

$$y_i(\boldsymbol{x}_i \cdot a + b) - 1 \geq 0 \; \forall i \tag{3.8}$$

Die Spanne zwischen beiden Klassen beträgt $\frac{2}{\|a\|}$. Indem $\|a\|^2$ unter den Randbedingungen in Gleichung 3.8 minimiert wird, kann die optimale separierende Hyperebene gefunden werden. Anschaulich wird dies in Abbildung 3.8 dargestellt. Das Auffinden einer

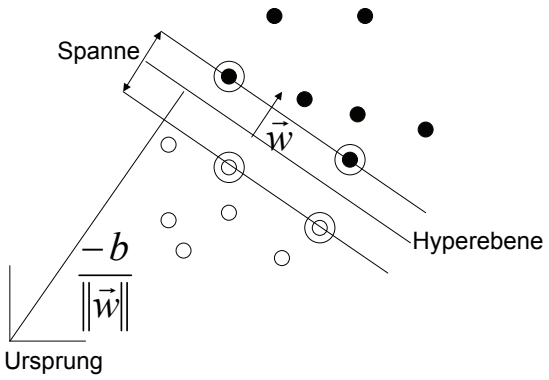


Abbildung 3.8.: Lineare separierende Hyperebenen, die Support-Vektoren sind eingekreist (frei übernommen aus (Burges, 1998)).

optimalen Hyperebene mit den dazugehörigen Support-Vektoren benötigt einen rechnerischen Aufwand, der abhängig von der Anzahl der Trainingsmuster in $\Omega(n^2)$ liegt. Daher ist das Training einer Support Vector Machine für das Aktive Lernen gut geeignet, da hierbei die Anzahl der klassifizierten Trainingsmuster gering gehalten wird.

Wenn die Daten im Eingaberaum X nicht linear trennbar sind, kann man sie mit Hilfe des so genannten „Kernel-Trick“ in einen höher-dimensionalen Merkmalsraum F projezieren. Dies geschieht mittels einer Transformation $\Phi : \boldsymbol{x} \rightarrow \phi(\boldsymbol{x})$. Da man für das Training einer SVM nur das innere Produkt zwischen zwei Vektoren benötigt, reicht die Angabe einer Kernel-Funktion

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j) \tag{3.9}$$

welche das innere Produkt im hochdimensionalen Merkmalsraum F spezifiziert, ohne dass explizit im Merkmalsraum gelernt werden muss.

(Tong und Koller, 2001) nutzen die Dualität zwischen dem Merkmalsraum F und dem Parameterraum W, um eine optimale Strategie im Versionsraum zu entwerfen. Punkte im Merkmalsraum F korrespondieren zu Hyperebenen im Parameterraum W und um-

gekehrt. Die Idee ist dabei, dass ein Trainingsmuster $\boldsymbol{x}_i$ im Merkmalsraum die Menge der separierenden Hyperebenen - und damit den Versionsraum - einschränkt.

Basierend auf dieser Idee wird versucht, Punkte im Merkmalsraum zu finden, die die minimale Distanz zu anderen Hyperebenen maximieren. Damit wird der aktuelle Versionsraum in jeder Iteration halbiert. Drei verschiedene Approximationen für diese Prozedur werden in dieser Arbeit vorgestellt:

Simple Margin Je zentraler eine Hyperebene im Versionsraum platziert ist, umso mehr halbiert sie ihn. Dies geschieht unter der Annahme, dass die SVM bereits zentral im Versionsraum platziert ist und dieser symmetrisch ist. Daraus resultiert folgende Selektionsstrategie: Trainieren einer Support Vector Machine auf den klassifizierten Daten und Auswahl eines Musters, welches im Merkmalsraum möglichst nahe an der separierenden Hyperebene liegt. Der gleiche Ansatz wird auch bei der Strategie des „Uncertainty sampling“ in der Arbeit von (Schohn und Cohn, 2000) verfolgt, welche später in Abschnitt 3.4.3 beschrieben wird.

MaxMin Margin Die relative Größe eines aus der Klassifikation von $\boldsymbol{x}_i$ resultierenden Versionsraumes kann mit der Spanne der Support Vector Machine geschätzt werden. Dabei bezeichnet V^+ den Versionsraum, falls $\boldsymbol{x}_i$ mit $+1$ klassifiziert wird und V^- den Versionsraum einer Klassifikation von $\boldsymbol{x}_i$ mit -1. Da der Versionsraum möglichst gleichmäßig aufgeteilt werden soll, sollte das Minimum der Größe dieser beiden Räume $\min(Area(V^-), Area(V^+))$ möglichst groß sein. Für jedes unklassifizierte Muster $\boldsymbol{x}_i$ werden die Spannen m^- und m^+ der resultierenden Support Vector Machines für $\boldsymbol{x}_i = -1$ und $\boldsymbol{x}_i = +1$ berechnet. Dann wird das Muster gewählt, für das $\min(m^-, m^+)$ am größten ist.

Ratio Margin Diese Methode ist ähnlich zur MaxMin Margin Methode, welche die Spannen m^- und m^+ als Indikator für die Größe des resultierenden Versionsraumes verwendet. Da die Form des Versionsraumes in manchen Fällen lang gestreckt sein kann, können die resultierenden m^- und m^+-Werte auch sehr klein sein. Um dieses Phänomen zu umgehen, wird die relative Größe von m^- und m^+ verwendet: $\min(\frac{m^-}{m^+}, \frac{m^+}{m^-})$.

In den Ergebnissen stellen (Tong und Koller, 2001) fest, dass bei manchen Datensätzen die MaxMin Margin und die Ratio Margin Methoden stabiler sind als die Simple Margin Methode. Dies wird darauf zurückgeführt, dass die Simple Margin Methode weniger den Merkmalsraum exploriert als die anderen beiden Methoden, da nur nach neuen Mustern an der separierenden Hyperebene gefragt wird. Diese Anfrage-Methode setzt voraus, dass

das aktuelle Modell gut initialisiert ist, da sonst manche Cluster unklassifizierter Daten im Eingaberaum ignoriert werden. Die MaxMin Margin und die Ratio Margin Methode beziehen alle Muster in die Auswahl mit ein. Der Nachteil besteht jedoch im hohen Rechenaufwand: bei n Mustern müssen $2n$ Support Vector Machines für jede einzelne Anfrage gelernt werden, was den praktischen Einsatz fast unmöglich macht.

Zusammenfassung: Reduktion des Versionsraumes

In diesem Abschnitt haben wir Ansätze für das Aktive Lernen betrachtet, welche versuchen, den Versionsraum in jeder Lerniteration möglichst stark zu reduzieren. Die direkte Analogie zwischen Versionsraum und Merkmalsraum einer SVM führte direkt zu verschiedenen Auswahlkriterien für Muster aus dem Eingaberaum. Die Idee eines generellen und eines speziellen Versionsraumes führte zu einer Definition einer Region der größten Unsicherheit. Über die Auswahl von Mustern in dieser Region werden die beiden Versionsräume eingeschränkt. Alternativ lässt sich der Versionsraum auch über ein Kommitee von verschiedenen Modellen beschreiben. Durch die Anpassung der Modelle durch ein neues Muster wird der Versionsraum verkleinert. Auch diese Algorithmen basieren auf der Annahme, dass ein stabiles Modell mit Hilfe von zufällig gezogenen Mustern bereits gelernt wurde.

3.4.3. Uncertainty sampling

Eine Heuristik im Aktiven Lernen - das so genannte *uncertainty sampling* - konzentriert sich direkt auf Muster, für die das aktuelle Modell bei der Vorhersage die geringste Konfidenz hat. Dies setzt voraus, dass der Klassifikator in der Lage ist, für ein Muster die Klassenwahrscheinlichkeiten zu bestimmen. Das Ziel dieser Strategie besteht darin, Muster auszuwählen, die an der Grenze zwischen unterschiedlichen Klassen liegen. Um die Unsicherheit des Klassifikators zu messen, können verschiedene Maße verwendet werden. Sei $p(y_c|\boldsymbol{x})$ die Wahrscheinlichkeit für das Auftreten von Klasse y_c für ein Muster $\boldsymbol{x}$. Die Shannon-Entropie (Shannon, 2001) berechnet sich folgendermaßen:

$$H(x) = -\sum_c p(y_c|\boldsymbol{x}) \log p(y_c|\boldsymbol{x}) \tag{3.10}$$

Eine Verteilung der Klassenwahrscheinlichkeiten mit scharfen Spitzen hat eine niedrige Entropie. Eine gleichverteile Wahrscheinlichkeit über alle Klassen hingegen hat eine hohe

Entropie. Daher kann die Shannon-Entropie verwendet werden, um die Unsicherheit eines Klassifikators zu messen.

Eine andere Möglichkeit besteht darin, die Distanz zwischen den zwei Klassen mit der höchsten Wahrscheinlichkeit y_c und y'_c zu messen:

$$|p(y_c|\boldsymbol{x}) - p(y'_c|\boldsymbol{x})| \tag{3.11}$$

Bei einem Klassifikationsproblem mit zwei Klassen sind beide Maße äquivalent. Für n Muster, c Klassen und k Klassifikatoren beträgt der Aufwand für die Selektion $O(nck)$.

Eine der ersten Arbeiten von (Lewis und Gale, 1994) in diesem Bereich stellt eine Strategie für das uncertainty sampling im Bereich der Textklassifikation vor. Mithilfe eines probabilistischen Bayes-Klassifikators werden die Wahrscheinlichkeiten für die Klassenzugehörigkeit (beschränkt auf zwei Klassen 0 und 1) für jedes Muster bestimmt. Das bedeutet, dass der Wert der Ausgabe des Klassifikators bei 0 für die Klasse „0“ liegt und bei 1 für Klasse „1“. Der Bereich der größten Unsicherheit setzt sich aus den Mustern zusammen, für die der Klassifikator eine Ausgabe im Bereich 0,5 hat. In der Arbeit von (Lewis und Gale, 1994) werden $b/2$ Muster mit der Ausgabe des Modells 0.5 und größer und $b/2$ Muster mit 0.5 und kleiner ausgwewählt. Dies soll sicherstellen, dass auf beiden Seiten der Entscheidungsgrenze nach Mustern gefragt wird. Diese Strategie ist sehr rudimentär und auf 2-Klassen Probleme beschränkt.

In der Arbeit von (Schohn und Cohn, 2000) wurde die Heuristik des uncertainty sampling auf einer Support Vector Machine (SVM) umgesetzt. Diese Heuristik deckt sich mit dem im vorherigen Abschnitt behandelten Ansatz der „Simple Margin“ in der Arbeit von (Tong und Koller, 2001). Dieselbe Stragie wurde auch in der Arbeit von (Campbell u. a., 2000) entwickelt. Hier wurde also mit unterschiedlichen Ansätzen fast zeitgleich dieselbe Strategie entworfen. Ausgangspunkt der Arbeit von (Schohn und Cohn, 2000) ist die Idee, die aktuelle Spanne der SVM zwischen zwei Klassen zu minimieren, indem Muster ausgewählt werden, welche möglichst nahe an der separierenden Hyperebene liegen. Die Auswahl von Mustern nach ihrer Distanz zur Hyperebene benötigt mittels des Skalarproduktes nur wenig rechnerischen Aufwand.

Zusammenfassend lässt sich sagen, dass Ansätze mit SVM und Aktivem Lernen in vielen praktischen Anwendungen verwendet werden. Im Bereich des *Image Retrieval* (Tong und Chang, 2001) (Luo u. a., 2005) (Wang u. a., 2003) oder *music retrieval* (Mandel u. a., 2006) oder speziell im Bereich von bildgestützter Arzneimittelforschung (Warmuth u. a., 2003) sind solche Ansätze zum Beispiel oft vertreten.

Zusammenfassung: Uncertainty sampling

In diesem Abschnitt wurde die Heuristik des Uncertainty sampling vorgestellt. Hierbei liegt der Fokus direkt auf den Klassengrenzen des aktuell gelernten Modells. Auch bei dieser Heuristik ist ein stabiles Modell wichtig. Bei den hier vorgestellten Arbeiten werden die Modelle jedoch auch wieder mit zufälligen Mustern initialisiert.

3.4.4. Aktives Lernen und Clustering

Es existieren diverse Erweiterungen zu der im vorherigen Abschnitt behandelten Arbeit von (Schohn und Cohn, 2000), welche ein Clustering in die Selektion von neuen Mustern mit einbeziehen. So wurde in den Arbeiten (Kang u. a., 2004) (Cheng und Shih, 2007) ein k-means Clustering (MacQueen, 1967) zur initialen Selektion von Trainingsmustern verwendet. Basierend auf den Mustern, die am nächsten an den gefundenen Cluster Zentren liegen, wird eine initiale SVM gelernt. Dies wirkt sich positiv auf die Stabilität in den ersten Lerniterationen aus. Ein weiterer Ansatz (Xu u. a., 2004) geht noch einen Schritt weiter und clustert die Daten, die nahe an der Hyperebene liegen, um somit möglichst repräsentative Muster im Bereich der größten Unsicherheit zwischen zwei Klassen zu finden. Diese Ansätze sind nicht unumstritten; in der Arbeit von (Zhang und Oles, 2000) wird argumentiert, dass es bei diskriminativen Modellen wie der SVM nicht hilfreich ist, die Verteilung der Daten zu berücksichtigen. Ergebnisse in der Arbeit von (Xu u. a., 2004) zeigen auf Testdaten aus dem Bereich der Textklassifikation eine bessere Performanz zu Beginn des Lernens. Mit zunehmender Anzahl von Mustern ist die Performanz jedoch generell schlechter als der Ansatz von (Schohn und Cohn, 2000).

Die Arbeit von (Nguyen und Smeulders, 2004) verfolgt einen Ansatz zur Selektion von Daten, der ein Clustering mit einbezieht. Die Daten werden mit Hilfe eines logistischen Regressionsmodells (Menard, 2002) klassifiziert. Die Autoren argumentieren, dass Datenpunkte an der Klassifikationsgrenze informativ sind, jedoch zusätzliche Information über die Verteilung der Daten hilfreich sein können. Die Autoren nehmen an, dass in den Daten X eine Clusterstruktur existiert. Die Arbeit beschränkt sich auf 2-Klassen Probleme $y \in \{0, 1\}$. Zunächst werden die Daten geclustert und die Cluster-Repräsentanten als initiale Muster zur Klassifikation ausgewählt. Für das Clustering wird in dieser Arbeit der k-medoid Algorithmus von (Struyf u. a., 1997) verwendet. Der k-medoid Algorithmus ist sehr ähnlich zum fuzzy c-means Algorithmus, welcher in Abschnitt 4.1.3 beschrieben wird; beide Algorithmen minimieren den quadratischen Fehler zwischen den Datenpunk-

ten, die einem Cluster zugeordnet sind, und dem Cluster-Zentroid. Der Unterschied ist, dass bei fuzzy c-means der Zentroid der Durchschnitt aller zugeordneten Punkte ist; beim k-medoid ist es der Datenpunkt, der diesem Durchschnitt am nächsten liegt.

Die Cluster Label $k \in \{1, 2, \ldots, K\}$ bezeichnen die Cluster-Information. Die Verteilung der Klassenlabel $p(y|k)$ für jeden Cluster werden mit einem logistischen Regressionsmodell berechnet. Die Parameter dieses Modells werden mittels der Maximum-Likelihood Methode (vgl. Abschnitt 2.3) geschätzt. Dabei fließen sowohl Informationen der unklassifizierten Daten U als auch der klassifizierten Daten L in die Schätzung mit ein. Mit $p(k|\boldsymbol{x})$ wird die *a priori* Wahrscheinlichkeit, dass für den Datenpunkt $\boldsymbol{x}$ der Cluster k auftritt, bezeichnet. Diese wird bereits im k-medoid Algorithmus bestimmt.

In der Arbeit von (Nguyen und Smeulders, 2004) wird angenommen, dass sich die Informationen über die Klasse in den Cluster-Labeln befinden. Sobald k bekannt ist, sind y und $\boldsymbol{x}$ unabhängig voneinander. Um das Klassenlabel für ein Muster zu bestimmen, wird die *a priori* Wahrscheinlichkeit des Clusters und dessen Klassenlabel ausgewertet:

$$p(y|\boldsymbol{x}) = \sum_{k=1}^{K} p(y|k)p(k|\boldsymbol{x}) \tag{3.12}$$

$p(k|\boldsymbol{x})$ wird einmal zu Beginn und anschließend immer nur dann, wenn die Daten neu geclustert werden, berechnet. $p(y|k)$ wird hingegen jedes Mal, wenn ein neues Muster zur Trainingsmenge hinzugefügt wird, neu berechnet.

Nachdem die Cluster und die Parameter für das logistische Regressionsmodell bestimmt worden sind, werden neue Muster nach einem speziellen Kriterium ausgewählt. Dieses setzt sich zusammen aus dem erwarteten Fehler für das gegebene Muster und der Dichte des Musters im Eingaberaum X. Dieses Kriterium wird als „Density Weighted Uncertainty Sampling“ (kurz: DWUS) bezeichnet:

$$\arg\max_{\boldsymbol{x} \in U} E[(\hat{y}(\boldsymbol{x}) - y(\boldsymbol{x}))^2|\boldsymbol{x}]p(\boldsymbol{x}) \tag{3.13}$$

$E[(\hat{y}(\boldsymbol{x}) - y(\boldsymbol{x}))^2|\boldsymbol{x}]$ ist der erwartete Fehler (der Unterschied zwischen der „wahren“ Klasse $\hat{y}$ und der Ausgabe des Lerners y) und $p(\boldsymbol{x})$ ist die Dichte des Datenpunktes $\boldsymbol{x}$, welche über die aktuellen Cluster geschätzt wird:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} p(\boldsymbol{x}|k)p(k) \tag{3.14}$$

$p(\boldsymbol{x}|k)$ wird in dieser Arbeit über eine multivariate Gauß'sche Verteilung mit derselben Varianz für alle Cluster beschrieben.

Die Fehlererwartung für einen nicht klassifizierten Datenpunkt wird folgendermaßen bestimmt:

$$E[(\hat{y}(\boldsymbol{x}) - y(\boldsymbol{x}))^2|\boldsymbol{x}] = (\hat{y}(\boldsymbol{x}) - 1)^2 p(y = 1|\boldsymbol{x}) + (\hat{y}(\boldsymbol{x}))^2 p(y = 0|\boldsymbol{x}_i) \tag{3.15}$$

Da die „wahre" Wahrscheinlichkeit $p(y|\boldsymbol{x})$ unbekannt ist, wird sie durch die aktuelle Approximation des Modells ersetzt. Wenn der erwartete Fehler abnimmt, werden die Daten in unregelmäßigen Intervallen neu geclustert. Der Grund dafür liegt darin, dass die Entscheidungsgrenzen nach mehreren Iterationen somit feiner modelliert werden können.

Die Arbeit von (Nguyen und Smeulders, 2004) ist eine der wichtigsten Arbeiten im Bereich des Aktiven Lernens mit Hilfe von Clustering, sie birgt jedoch auch gewisse Nachteile: Erstens beschränkt sich der Algorithmus auf 2-Klassen Probleme. Das initiale Clustering wirkt sich positiv auf die Klassifikation bei den ersten gewählten Mustern aus. Eine ungünstige Anzahl von Clustern kann jedoch dazu führen, dass der Lernprozess verlangsamt wird. Durch die konstante Einbeziehung der (durch die Clusterstruktur vorgegeben sehr groben) Dichteschätzung fokussiert sich der Algorithmus in nachfolgenden Iterationen nicht direkt auf die Klassengrenzen. (Dönmez u. a., 2007) haben nachgewiesen, dass diese Strategie, die Dichte der Datenpunkte konstant in die Auswahl mit einzubeziehen, im Vergleich zu Strategien des *Uncertainty Sampling* in nachfolgenden Iterationen schlechter abschneidet.

Es existieren diverse Arbeiten, die die aktive Selektion von Daten mit einem Clustering verbinden. Allerdings werden hierbei die Daten nicht klassifiziert, sondern nur in Cluster eingeteilt. Daher werden sie an dieser Stelle nur in kurzer Form vorgestellt:

In der Arbeit von (Basu u. a., 2004) wird beispielsweise der k-means Clusteralgorithmus (MacQueen, 1967) mit paarweisen Einschränkungen verknüpft. Die Zielfunktion von k-means wird um einen Term erweitert, welcher die Kosten für die Verletzung von paarweisen Einschränkungen misst. Hierbei handelt es sich um so genannte *must-link* (zwei Muster gehören zur selben Klasse) und *cannot-link* (zwei Muster gehören zu unterschiedlichen Klassen) Einschränkungen. Die Heuristik zur Auswahl von Mustern besteht darin, zunächst mit einem Farthest-First-Traversal das Muster auszuwählen, welches am weitesten von den bisherigen Einschränkunen entfernt ist. Dieses Verfahren wird in Abschnitt 4.1.1 noch genauer vorgestellt. Anschließend werden die Cluster mit zufällig gewählten Mustern konsolidiert. Die Auswahl von Mustern ist anfällig für Ausreißer, und

zufällig gewählte Muster entsprechen nicht direkt der Idee des Aktiven Lernens. Trotzdem ist die Arbeit von (Basu u. a., 2004) eine der ersten Ansätze, welche eine gezielte Auswahl von Mustern mit einem Clustering verbinden.

In der Arbeit von (Grira u. a., 2005) wird wie in der zuvor beschriebenen Arbeit ein aktives halb überwachtes Clustering durchgeführt, indem ein Kostenfaktor (in dieser Arbeit in die Zielfunktion des Fuzzy c-means Algorithmus (Bezdek, 1981)) für die Verletzung von paarweisen Einschränkungen mit eingerechnet wird. Um die Menge der Einschränkungen zu limitieren, wird ein aktives Selektionsschema vorgeschlagen, welches auf der Idee beruht, dass vor allem Cluster redefiniert werden müssen, die nicht kompakt oder gut von den anderen Clustern separiert sind. In jeder Iteration wird der „schlechteste“ Cluster gefunden, indem für jeden Cluster das Fuzzy-Hypervolumen berechnet wird. Das Fuzzy Hypervolumen für einen Cluster gilt als Indikator für die Kompaktheit eines Clusters. Ein einzelner, schlecht definierter Cluster kann bei der Selektionsstrategie von (Grira u. a., 2005) dazu führen, dass nur an den Grenzen dieses Clusters Muster abgefragt werden, was für eine aktive Lernstrategie ineffizient sein kann.

In der Arbeit „Active Data Clustering“ (Hofmann und Buhmann, 1997) wird eine aktive Selektionsstrategie vorgeschlagen, um ein gutes Clustering zu erreichen. Dabei basiert das Clustering nicht auf Attributvektoren, sondern auf paarweise definierten (Un)ähnlichkeiten. Der Vorteil dieser Methode besteht darin, dass solche Unähnlichkeiten fallweise leichter definiert werden können als Attribute zu erzeugen und eine geeignete Metrik zu definieren. Der Nachteil besteht in der hohen Komplexität und der daraus resultierenden schlechten Skalierung. Bei N Datensätzen sind N^2 paarweise Vergleiche nötig. Da es bei großen Datensätzen nicht möglich ist, alle Vergleiche zu berechnen und im Hauptspeicher zu halten, wird auf einer Untermenge von bekannten Unähnlichkeiten gearbeitet. Sequentiell werden neue Muster zur Clusterstruktur hinzugefügt. Die vorläufigen Kenntnisse der Struktur des Clusterings werden verwendet, um neue Muster auszuwählen. Dabei werden Muster gewählt, die das erwartete Risiko einer falschen Vorhersage bezüglich der aktuellen Clusterstruktur minimieren. Diese aktive Selektionsmethode kann auch als Maximierung des erwarteten Informationswertes neuer Muster aufgefasst werden. Die Autoren stellen bei der Auswertung der Ergebnisse fest, dass sich die Selektion von Mustern nach einer kurzen Initialisierungsphase auf solche Muster konzentriert, welche dazu beitragen, zwischen nahen Clustern zu unterscheiden.

Zusammenfassung: Aktives Lernen und Clustering

In diesem Abschnitt wurden diverse Verfahren vorgestellt, die das aktive Lernen mit einem Clustering verbinden. Neben Arbeiten, die eine aktive Selektion von Mustern zur Verbesserung eines Clusterings verwenden, gibt es auch Arbeiten, die ein Clustering dazu verwenden, möglichst repräsentative Muster für eine Klassifikation auszuwählen.

3.4.5. Meta-Heuristiken

In diesem Abschnitt sollen diverse Meta-Heuristiken vorgestellt werden, die im Bereich des Aktiven Lernens entwickelt worden sind. Die ersten zwei vorgestellten Arbeiten beschäftigen sich mit der Auswahl zwischen mehreren aktiven Lernstrategien. Weiterhin wird noch eine Erweiterung des Query-by-Committee Algorithmus mittels boosting und bagging und ein aktive Lernstrategie mit verschiedenen Beschreibungsräumen vorgestellt.

Die Arbeit von (Osugi u. a., 2005) beschäftigt sich mit der richtigen Balance zwischen Exploration und Verfeinerung der Klassengrenzen beim SVM-Lernen. Zur Exploration wird der Kernel-Farthest-First (KFF)Algorithmus verwendet. Hierbei wird ein Muster aus den unklassifizierten Daten U ausgewählt, welches sich am weitesten entfernt von den klassifizierten Mustern L befindet:

$$\arg\max_{\boldsymbol{x}\in U}\left(\min_{\boldsymbol{x}_l\in L}\|\phi(\boldsymbol{x})-\phi(\boldsymbol{x}_l)\|\right) \tag{3.16}$$

wobei

$$\|\phi(\boldsymbol{x})-\phi(\boldsymbol{x}_l)\| = \sqrt{K(\boldsymbol{x},\boldsymbol{x})+K(\boldsymbol{x}_l,\boldsymbol{x}_l)-2K(\boldsymbol{x},\boldsymbol{x}_l)} \tag{3.17}$$

die euklidische Distanz im Merkmalsraum darstellt. Um die Klassengrenzen zu verfeinern, wird dieselbe Strategie wie in der Arbeit von (Schohn und Cohn, 2000) verwendet, welche als *Simple* bezeichnet wird. In jeder Lerniteration wird darüber entschieden, ob der KFF-Algorithmus zur Exploration oder der *Simple*-Algorithmus zur Verfeinerung der Klassengrenzen verwendet wird. Dafür wird die Änderung im Versionsraum verfolgt. Trägt ein Muster im Explorationssschritt maßgeblich zu einer Veränderung bei, so wird die Wahrscheinlichkeit für einen weiteren Explorationsschritt erhöht. Die Ergebnisse dieser Arbeit zeigen eine Verbesserung bei künstlich generierten Daten, welche eine höhere Exploration benötigen. Auf benchmark-Daten ist die Performanz jedoch nicht besser als der *Simple*-Algorithmus.

In der Arbeit von (Baram u. a., 2004) werden verschiedene, bereits in den vorherigen Abschnitten behandelte Algorithmen aufgegriffen und miteinander kombiniert. Nach empirischen Untersuchungen bezüglich der Performanz haben die Autoren den Simple-Algorithmus (Tong und Koller, 2001) und den Algorithmus von (Roy und McCallum, 2001) - der hier als „Self-conf“ bezeichnet wird - ausgewählt. Weiterhin wird der im vorherigen Abschnitt behandelte Kernel-Farthest-First Algorithmus (KFF) eingesetzt, dessen Stärke vor allem darin liegt, so genannte XOR-Strukturen (schachbrettartige Muster in den Daten) zu erkennen. Generell wird empirisch nachgewiesen, dass jeder der vorgestellten Algorithmen je nach zugrunde liegendem Problem eine bessere oder schlechtere Performanz hat. Daher wird das Problem in der Arbeit von (Baram u. a., 2004) darauf reduziert, in jeder Iteration ein Muster des Aktiven Lerners zu verwenden, das die beste erwartete Performanz hat. Da nur sehr wenige klassifizierte Daten zur Verfügung stehen, funktionieren die klassischen Evaluierungstechniken wie z. B. Kreuzvalidierung nicht. Hierbei wird normalerweise der Datensatz in mehrere Partitionen unterteilt, wobei eine Partition zum Lernen und die anderen Partitionen zum Testen verwendet werden. Da beim Aktiven Lernen nicht genügend Daten für solch ein statistisches Verfahren zur Verfügung stehen, wird in dieser Arbeit ein neues Evaluationskriterium vorgestellt.

Zur Auswahl eines Aktiven Lerners wird das Problem in die Spieltheorie übertragen: Hier ist die Aufgabe, aus verschiedenen, nicht-identischen einarmigen Banditen eine Sequenz auszuwählen, so dass der Gewinn maximiert wird. Zur Lösung dieses Problems wird ein bereits entwickelter Algorithmus EXP4 (Auer u. a., 2003) verwendet. Das Ziel in diesem Algorithmus ist es, k Strategien bzw. Orakel so miteinander zu kombinieren, dass n verschiedene einarmige Banditen möglichst gut gespielt werden. In jeder Runde t bestimmt jedes Orakel j, $j = 1, \ldots, k$ eine Wahrscheinlichkeit, die i-te Maschine zu spielen $b^j(t) = (b_1^j(t), \ldots, b_n^j(t))$ mit $\sum_i b_i^j(t) = 1$. Der Vektor $g(t) = (g_1(t), \ldots, g_n(t))$ definiert die Belohnungen aller Maschinen, somit ist der erwartete Gewinn für ein Orakel j in Runde $t = b^j(t) \cdot g(t)$. Die Differenz zwischen dem bestmöglichsten Gewinn $G_{\max}$ und dem erwarteten Gewinn im Algorithmus EXP4 ist beschränkt durch $O(G_{\max} n \ln k)$.

Zur Anwendung des Algorithmus im Kontext des Aktiven Lernens werden die Wahrscheinlichkeiten $b_i^j(t)$ zur Ziehung eines Musters durch den jeweiligen aktiven Lernalgorithmus und die Schätzungen für den erwarteten Gewinn $g(t)$ benötigt. Die Wahrscheinlichkeiten $b_i^j(t)$ lassen sich direkt mit den verwendeten Algorithmen bestimmen: Beim Simple-Algorithmus wird die Kernel-Distanz zur Hyperebene verwendet, bei Self-conf der erwartete Fehler und bei KFF die Kernel-Distanz zur aktuellen Trainingsmenge L.

Für die Schätzung des erwarteten Gewinnes wird das so genannte *Classification Entropy Maximation* (CEM) Kriterium verwendet. Bei einem Zwei-Klassen Problem mit den Klassen $\{0, 1\}$ seien $C^1(U)$ und $C^0(U)$ die jeweils als durch den zugrunde liegenden Lerner positiv bzw. negativ klassifizierten Untermengen von U. Der CEM-Wert ist dann die binäre Entropie:

$$H(\frac{|C^1(U)|}{|U|}) \tag{3.18}$$

Der CEM-Wert ist also hoch, wenn die Klassifikation der Muster für beide Klassen ausgeglichen ist. Ein formaler Zusammenhang zwischen dem CEM-Wert und dem Generalisierungsfehler des Aktiven Lerners besteht nicht. Die Ergebnisse auf verschiedenen Datensätzen legen jedoch nahe, dass dieses Kriterium gut funktioniert.

Ein weiterer Beitrag dieser Arbeit ist die Definition eines Kriteriums zur Schätzung der Performanz eines Aktiven Lerners. Dieses setzt sich aus den folgenden Komponenten zusammen: $\mathrm{Acc}_t(\mathrm{ALG})\, t = 1, \ldots, n$ ist die mittlere Genauigkeit, welche mit dem Algorithmus ALG mit t zufällig und gleichverteilt gezogenen Trainingsdaten erreicht werden kann. ACTIVE ist ein Aktiver Lerner, welcher ALG als Lerner verwendet. $\mathrm{Acc}_t(\mathrm{ACTIVE})$ ist die mittlere Genauigkeit, welche nach t Iterationen mit dem Aktiven Lerner erreicht wurde. Die Defizit-Funktion eines Aktiven Lerners berechnet sich folgendermaßen:

$$\mathrm{Def}_n(\mathrm{ACTIVE}) = \frac{\sum_{t=1}^{n} \mathrm{Acc}_n(\mathrm{ALG}) - \mathrm{Acc}_t(\mathrm{ACTIVE})}{\sum_{t=1}^{n} \mathrm{Acc}_n(\mathrm{ALG}) - \mathrm{Acc}_t(\mathrm{ALG})} \tag{3.19}$$

Der Zähler misst die Fläche zwischen der maximal möglichen Genauigkeit und der Lernkurve des aktiven Lernalgorithmus. Der Nenner normalisiert diesen Wert mit der Fläche zwischen der maximal möglichen Genauigkeit und der Lernkurve des passiven Lernalgorithmus.

Die Arbeit von (Baram u. a., 2004) verdeutlicht, dass eine Anwendung von verschiedenen aktiven Lernstrategien zu unterschiedlichen Zeitpunkten sinnvoll sein kann. Auf verschiedenen Datensätzen wird nachgewiesen, dass die Performanz gegenüber den einzelnen Verfahren stabiler und im Durchschnitt besser ist.

Die Arbeit von (Abe und Mamitsuka, 1998) beschäftigt sich mit der Erweiterung des Query-by-Committee Algorithmus von (Freund u. a., 1997) durch Boosting und Bagging. Bagging (Breiman, 1996) ist eine Meta-Heuristik, um die Stabilität und Performanz eines Klassifikators zu verbessern. Dabei werden aus den Trainingsdaten L mit Kardinalität n durch gleichverteilte Ziehung von Mustern (mit Zurücklegen) j neue Trainingsmengen der Kardinalität $m < n$ erzeugt. Der „Query-by-bagging“ Algorithmus funktioniert

analog zum Query-by-committee-Algorithmus. Der Unterschied besteht darin, dass die zugrunde liegenden Daten zunächst mit dem Bagging-Algorithmus aufgeteilt werden und ein Klassifikator auf jeder Untermenge trainiert wird.

In einem zweiten Ansatz wird in dieser Arbeit der AdaBoost (Adaptive Boosting) Algorithmus (Freund und Schapire, 1995) verwendet. AdaBoost wird ebenso wie Bagging dazu verwendet, um die Performanz eines Klassifikators zu verbessern. Ein Klassifikator wird dabei in mehreren Iterationen auf den Daten trainiert, wobei jedem Muster ein Gewicht zugeordnet ist. Diese Gewichte werden in jeder Iteration angepasst; dabei bekommen Muster, welche falsch klassifiziert werden, ein höheres Gewicht, so dass sich der Klassifikator mehr auf diese Muster konzentriert. Damit ist dieser Algorithmus allerdings auch sehr anfällig für Rauschen und Ausreißer in den Daten. Die im AdaBoost Algorithmus gewonnenen Gewichte werden zur Auswahl von Mustern verwendet. Es werden genau die Muster ausgewählt, für welche die Klassifikatoren uneinig sind und deren Gewichte im AdaBoost Algorithmus möglichst gleich sind.

Als Klassifikator wird in der Arbeit von (Abe und Mamitsuka, 1998) der C4.5 Entscheidungsbaum (Quinlan, 1993) verwendet. Der direkte Vergleich zum Query-by-Committee-Algorithmus wird nicht geführt, jedoch wird auf verschiedenen Datensätzen gezeigt, dass Query-by-boosting und Query-by-bagging eine bessere Performanz haben als ein Entscheidungsbaum, der mit der entsprechenden Anzahl an zufällig gezogenen Mustern trainiert wurde.

Die Arbeit von (Abe und Mamitsuka, 1998) wurde von (Melville und Mooney, 2004) mit dem Decorate-Algorithmus (Melville und Mooney, 2005) erweitert, welcher mit künstlich generierten Mustern mehr Diversität zwischen den Lernern erzeugen soll. Hierbei wird iterativ ein Ensemble von Lernern generiert, welche auf den Trainingsdaten und auf künstlich generierten Daten (basierend auf einem einfachen Modell der Datenverteilung) trainiert werden. Dabei werden die künstlich generierten Muster so ausgewählt, dass ihre Klassenlabel maximal von der aktuellen Klassifikation des Ensembles abweichen. Ein auf den Trainingsdaten und den diversen Mustern trainierter Lerner wird dann zum Ensemble hinzugefügt, wenn er den Ensemble-Trainingsfehler nicht erhöht.

(Muslea u. a., 2006) beschäftigt sich mit dem Fall, dass mehrere Beschreibungsräume bzw. Sichten $V_1, \ldots, V_k$ auf die Daten zur Verfügung stehen. Zunächst wird mit den klassifizierten Trainingsdaten L ein Modell in jedem Beschreibungsraum gelernt. Anschließend wird ein unklassifiziertes Muster ausgewählt, für welches die gelernten Klassifikatoren unterschiedliche Vorhersagen machen (so genannte *contention points*). Nach

einer festgelegten Anzahl von Mustern wird die resultierende Hypothese ausgegeben. Für die Auswahl von Mustern existieren drei verschiedene Strategien:

Naiv: Zufällige Auswahl von einem Muster unter den *contention points*. Diese Strategie ist vor allem dann sinnvoll, wenn der Basis-Lerner nicht in der Lage ist, Klassenwahrscheinlichkeiten auszugeben.

Agressiv: Auswahl des Musters, für welches der Beschreibungsraum mit der geringsten Konfidenz das Muster mit der höchsten Konfidenz klassifiziert. Diese Strategie eignet sich besonders dann, wenn kein oder nur wenig Rauschen in den Daten existiert. In diesem Fall wird mehr als die Hälfte des Versionsraumes reduziert.

Konservativ: Auswahl des Musters, für welches die Konfidenz der verschiedenen Klassifikatoren möglichst gleich ist.

Auch bei der Generierung der resultierenden Klassifikation gibt es verschiedene Möglichkeiten:

Gewichtete Abstimmung: Hierbei wird die Klassifikation in jedem Beschreibungsraum mit seiner Konfidenz gewichtet.

Mehrheitsentscheidung: Hierbei wird die Klassifikation gewählt, welche in den meisten Klassifikationen der Beschreibungsräume aufgetreten ist.

Winner-takes-all: Hierbei wird die Klassifikation des Beschreibungsraumes gewählt, der die wenigsten Fehler in den Lerniterationen gemacht hat. Diese Strategie bietet sich an, wenn der Klassifikator im Beschreibungsraum keine Aussage zu seiner Konfidenz machen kann.

Erweitert wird das Konzept der Beschreibungsräume durch so genannte starke oder schwache Beschreibungsräume. In einem schwachen Beschreibungsraum können nur Konzepte gelernt werden, die genereller oder spezifischer als das Zielkonzept sind. So kann sich ein schwacher Beschreibungsraum z. B. in einer hierarchischen Klassifikation dazu eignen, positive von negativen Mustern grob zu unterscheiden. Eine genauere Kategorisierung der positiven Muster kann nur in einem anderen starken Beschreibungsraum dargestellt werden. Da sich der schwache Beschreibungsraum nicht eignet, um das Zielkonzept zu lernen, werden die *contention points* weiterhin nur von den starken Beschreibungsräumen gewählt.

Der schwache Beschreibungsraum wird zum einen als Zusatzinformation bei der Klassifikation verwendet. Wenn es ein Unentschieden bei der Klassifikation durch die verschiedenen Beschreibungsräume gibt, kann diese Information verwendet werden. Zum anderen kann dieser Raum bei der Selektion von Mustern verwendet werden. Zuerst werden die

Muster gewählt, bei denen die starken Beschreibungsräume eine andere Klasse vorhersagen als der schwache Beschreibungsraum. Unter diesen Mustern wird das Muster mit der höchsten Konfidenz im schwachen Beschreibungsraum ausgewählt. Co-Testing kann als eine Abwandlung des Query by Committee-Algorithmus (vgl. Abschnitt 3.4.2) aufgefasst werden. Der Unterschied besteht vor allem darin, dass die Lerner aus dem Komitee nicht in einem Beschreibungsraum, sondern in unterschiedlichen Beschreibungsräumen gelernt wurden.

Zusammenfassung: Meta-Heuristiken

In diesem Abschnitt wurden verschiedene Meta-Heuristiken für die Auswahl von Mustern vorgestellt. Neben der Nutzung von verschiedenen Beschreibungsräumen und der Erweiterung des Query-by-Committee Algorithmus mittels Boosting und Bagging wurden in manchen Arbeiten verschiedene aktive Lernverfahren miteinander kombiniert, um so eine bessere Performanz zu erreichen. Diese Arbeiten deuten darauf hin, dass eine einzige starre Strategie bei der aktiven Auswahl von Mustern nicht ausreichend ist.

3.5. Fazit

In diesem Kapitel haben wir uns mit der Idee des Aktiven Lernens beschäftigt. Insbesondere haben wir verschiedene Algorithmen aus dem Bereich des *pool-based active learning* betrachtet.

Zwei Probleme treten bei den Verfahren des Aktiven Lernens besonders deutlich zutage: Zum einen die mangelnde theoretische Untersuchung zum Generalisierungsfehler beim Aktiven Lernen. Erste Untersuchungen in den Arbeiten von (Freund u. a., 1997) und (Dasgupta, 2004) beziehen sich auf klar definierte Fälle ohne Rauschen in den Daten.

Der zweite Mangel in den bisherigen Arbeiten zum Aktiven Lernen besteht darin, dass in vielen Arbeiten davon ausgegangen wird, dass ein stabiles Modell mit einer kleinen Anzahl von Daten bereits gelernt wurde. Daher fokussieren die meisten Ansätze aus dem Bereich der Optimierung einer Zielfunktion, Reduktion des Versionsraumes oder des uncertainty sampling die Auswahl auf Muster, welche nützlich für ein bereits gelerntes stabiles Modell sind. Eine Ausnahme bildet die in Abschnitt 3.4.4 angesprochene Arbeit von (Nguyen und Smeulders, 2004), bei der ein initiales Clustering zur Initialisierung dient.

Weiterhin verfolgt eine Großzahl von Strategien zum Aktiven Lernen eine starre Strategie zur Auswahl von Mustern.

In Abschnitt 3.4.4 wurde eine Arbeit mit dem Nachweis angesprochen, dass Selektionsstrategien, welche die Verteilung der Daten berücksichtigen, zu Beginn eine bessere Performanz haben als solche Strategien, welche sich ausschließlich auf die Klassifikationsgrenzen beziehen. In nachfolgenden Iterationen kehrt sich dieser Sachverhalt jedoch um, hier schneiden Selektionsstrategien an Klassifikationsgrenzen besser ab. Dies legt die Schlussfolgerung nahe, dass es besser sein könnte, zunächst repräsentative Muster und in späteren Iterationen Muster an den Klassengrenzen zu selektieren.

Es wird daher in dieser Arbeit eine allgemeine Selektionsstrategie entworfen, welche zunächst die Verteilung der Daten berücksichtigt und sich anschließend auf die Klassifikationsgrenzen konzentriert. Damit wird die Initialisierung des Modelles vom ersten gezeigten Muster an mit einbezogen, so dass das Modell nicht mit einer Menge von zufällig gezogenen Mustern initialisiert werden muss.

Basierend auf dieser Strategie wurden zwei verschiedene Verfahren entwickelt, die in den folgenden Kapiteln vorgestellt werden: Das erste Verfahren „Aktive Lernende Vektor Quantisierung“ (ALVQ) ist in zwei Phasen aufgeteilt: In der ersten Phase werden mittels eines Cluster-Verfahrens repräsentative Muster zur Klassifikation ausgewählt. In der zweiten Phase werden dann Muster an den Klassifikationsgrenzen des in der ersten Phase initialisierten Modelles gewählt.

Das zweite Verfahren „Prototypen-basierte aktive Klassifikation“ (PBAC) kombiniert den Repräsentationsgrad eines Musters und die Unsicherheit des Klassifikators in einem Kriterium, um die klassifizierten Muster als Prototypen für eine k-nächste Nachbarn Klassifikation auszuwählen. Hierbei findet durch Reduktion von Potentialen, welche auf der lokalen Dichte des Musters beruhen, ein fließender Übergang zwischen der Phase der Exploration und der Phase der Verfeinerung der Klassengrenzen statt.

Zunächst wird im nächsten Kapitel das Clustering und die Klassifikation mit Prototypen in allgemeiner Form betrachtet. Diese Verfahren bilden die Grundlage für die zwei aktiven Lernverfahren ALVQ und PBAC.

4. Clustering und Klassifikation mit Prototypen

In dieser Arbeit wird ein Prototypen basierter Ansatz verwendet, um die Daten zu clustern und zu klassifizieren. Dabei wird der Datensatz mit Hilfe von repräsentativen Mustern - so genannten Prototypen - beschrieben. Jeder Prototyp soll dabei eine Ansammlung von Mustern zusammenfassen. Voraussetzung hierfür ist die Definition einer geeigneten Distanzfunktion. Üblicherweise wird - wie in dieser Arbeit auch - die euklidische Distanz verwendet. Basierend auf den Distanzen der Muster zu den Prototypen werden die Muster den Prototypen zugeordnet. Diese Zuordnungen können eindeutig sein, es ist aber auch möglich, dass ein Muster zu mehreren Prototypen gehört.

Der Vorteil einer Prototypen basierten Klassifikation besteht zunächst in der Einfachheit des Modells (welches nur aus den Prototypen besteht). Für einen Menschen ist die Interpretation eines Prototypen-Modells weitaus einfacher als z. B. das Modell einer Support Vector Machine, welches aus Stützvektoren besteht, die semantisch keine Bedeutung haben. Mit Hilfe weniger Prototypen werden die Daten zusammengefasst und ermöglichen so ein leicht interpretierbares Modell für eine Clusterstruktur und/oder eine Klassifikation. Die Zusammenfassung von Informationen zu kleinen überschaubaren Informationseinheiten und die damit einhergende Filterung von redundanter Information lässt sich auch beim menschlichen Lernen beobachten.

Bei einem Clustering oder einer Klassifikation mit Prototypen spielen zwei Schritte eine wichtige Rolle: die Initialisierung der Prototypen und deren Anpassung während des Lernprozesses. Im nächsten Abschnitt 4.1 werden verschiedene Verfahren für die Initialisierung von Prototypen vorgestellt. Weiterhin wird anhand des Fuzzy c-means Algorithmus die Methode der alternierenden Optimierung für die Anpassung der Prototypen im Bereich des Clustering vorgestellt. Abschnitt 4.2 beschäftigt sich mit der Klassifikation mithilfe von Prototypen und stellt die Methode des *competitive learning* zur Anpassung der Prototypen vor.

4.1. Clustering mit Prototypen

Bei der Initialisierung von Prototypen lässt sich unterscheiden zwischen Methoden, welche die Prototypen abhängig oder unabhängig von der Datenverteilung initialisieren. Bei einer unabhängigen Initialisierung (beispielsweise mit zufällig gezogenen Werten) besteht die Möglichkeit, dass mehrere Prototypen für einen Cluster erzeugt werden und nicht alle Cluster gefunden werden. Das Ziel der datenabhängigen Initialisierung besteht also darin, die Prototypen weit über den Datenraum zu verteilen, um somit die Chance zu erhöhen, alle Cluster in den Daten zu finden.

4.1.1. Farthest-First-Traversal

Eine Methode für eine datenabhängige Initialisierung ist das so genannte *Farthest-First-Traversal*-Verfahren von (Hochbaum und Shmoys, 1985). Dabei werden die Prototypen so selektiert, dass der erste Prototyp zufällig gewählt wird und jeder weitere Prototyp mit dem Muster initialisiert wird, dessen minimale Distanz zu allen bisherigen Prototypen am größten ist. Die als *min-max-Distanz* bezeichnete Distanz von einem Datenpunkt $\boldsymbol{x}$ zur Menge der Protoypen S ist definiert als $d(S, \boldsymbol{x}) = \min_{\boldsymbol{y} \in S} d(\boldsymbol{x}, \boldsymbol{y})$. Da die Distanz zwischen den Prototypen maximiert wird, verhindert dieser Ansatz, dass die Prototpyen zu nahe an anderen Prototypen gewählt werden. Fällt die Distanz zur Menge der aktuellen Prototypen im Selektionsprozess signifikant ab, so kann davon ausgegangen werden, dass die geeignete Anzahl von Prototypen gefunden worden ist. Da Ausreißer in den Daten eine große Distanz zur Menge der Prototypen haben, ist bei diesem Verfahren die Wahrscheinlichkeit sehr hoch, dass solche Ausreißer-Muster als Prototypen selektiert werden. Dies ist von Nachteil, da diese Muster nicht repräsentativ für den Datensatz sind.

4.1.2. Subtractive Clustering

Erweiterte Verfahren zur Initialisierung von Prototypen sind das *subtractive clustering* (Chin, 1997) und das *mountain clustering* (Jang u. a., 1997). Beide Verfahren schätzen die Dichte im Eingaberaum, um somit möglichst repräsentative Muster als Prototypen auszuwählen. Beim *mountain clustering* wird die Dichtefunktion an jeder Stelle im Datenraum berechnet, beim *subtractive clustering* hingegen nur bei den Datenpunkten, was die Berechnung beschleunigt. Daher wird im Folgenden der Ansatz des

subtractive clustering näher vorgestellt.

Hierbei wird der Repräsentationsgrad eines Musters $\boldsymbol{x}_i$ geschätzt, indem sein so genanntes Potential P berechnet wird Das Potential $P(\boldsymbol{x}_i)$ eines Musters $\boldsymbol{x}_i$ ergibt sich aus:

$$P(\boldsymbol{x}_i) = \sum_{j=1}^{n} e^{-\alpha d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}, \ \alpha = \frac{4}{r_a^2} \tag{4.1}$$

wobei r_a eine positive Konstante ist, welche den Radius der Nachbarschaft beschreibt.

Alle Muster $\boldsymbol{x}_j$ in der unmittelbaren Nachbarschaft von $\boldsymbol{x}_i$ haben einen großen Einfluss auf das Potential. Das Potential wird mit allen anderen Mustern berechnet, was einen Aufwand von $O(n^2)$ bedeutet.

Nachdem für jeden Datenpunkt das Potential bestimmt worden ist, wird das Muster $\boldsymbol{x}_k^*$ mit dem höchsten Potential $P(\boldsymbol{x}_k^*)$ ausgewählt. Um zu vermeiden, dass ein Muster in der unmittelbaren Nachbarschaft in einer nächsten Iteration ausgewählt wird, werden die Potentiale in der Nachbarschaft des gewählten Musters reduziert.

Sei $\boldsymbol{x}_k^*$ das Muster mit dem höchsten Potential. Dann werden die Potentiale aller anderen Muster folgendermaßen reduziert:

$$P(\boldsymbol{x}_i) \Leftarrow P(\boldsymbol{x}_i) - P(\boldsymbol{x}_k^*) e^{-\beta d(\boldsymbol{x}_i, \boldsymbol{x}_k^*)^2}, \ \beta = \frac{4}{r_b^2} \tag{4.2}$$

wobei r_b eine positive Konstante ist, welche die Nachbarschaft bestimmt, in der die Potentiale reduziert werden.

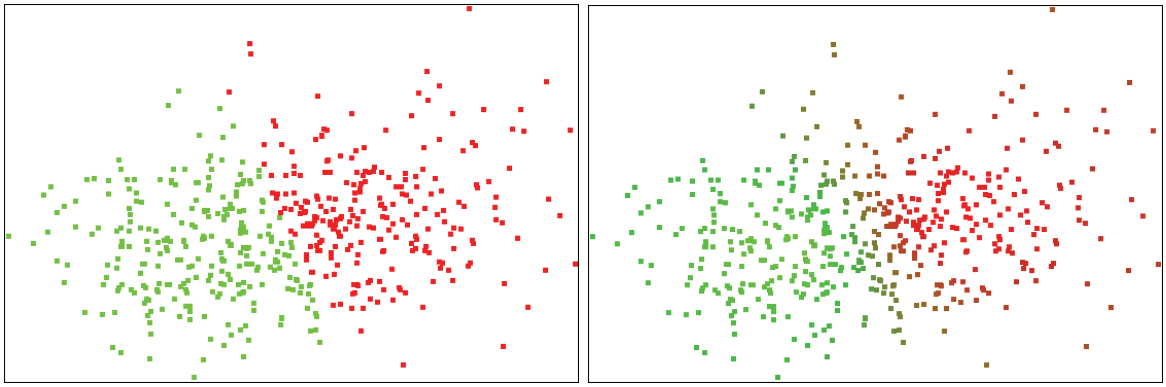
Üblicherweise wird diese Nachbarschaft etwas größer definiert, in der Arbeit von (Chin, 1997) wurde der Wert r_b auf $1.25\, r_a$ gesetzt.

Die Berechnung der Distanzen im *subtractive clustering*-Verfahren ist ein aufwändiger Prozess. Dieses Problem lässt sich jedoch durch geeignete Maßnahmen, wie z. B. lokale Einschränkung des Suchraums der Prototypen, lindern. Solche Verfahren werden in Kapitel 6 vorgestellt. Bei diesem Verfahren kann über die Angabe eines Schwellwertes für das Potential ein Abbruchkriterium festgelegt werden, um die Anzahl der Cluster zu bestimmen.

4.1.3. Fuzzy c-means

Repräsentativ für Prototypen-basierte Clusterverfahren wird an dieser Stelle der Fuzzy c-means (FCM) Algorithmus (Bezdek, 1981) vorgestellt.

Der FCM Algorithmus ist eine bekannte Technik zur Erkennung der zugrunde liegenden Strukturen in den Daten. Fuzzy Clustering zeichnet sich dadurch aus, dass die Cluster überlappen dürfen. Jedes Muster $\boldsymbol{x}_i$ besitzt einen Zugehörigkeitsgrad $w_{i,k} \in [0,1]$ zu jedem Cluster-Prototypen $\boldsymbol{p}_k$. Ein Zugehörigkeitsgrad von 1 bedeutet, dass das Muster mit absoluter Sicherheit dem entsprechenden Cluster zuzuordnen ist, ein Zugehörigkeitsgrad von 0 bedeutet, dass das Muster mit absoluter Sicherheit nicht dem entsprechenden Cluster zuzuordnen ist. Die Werte zwischen 0 und 1 beschreiben dementsprechend graduell die Zuordnung zu einem jeweiligen Cluster. Befindet sich ein Muster z. B. zwischen verschiedenen Clustern, so hat es zu allen denselben Grad der Zugehörigkeit. Der Vorteil von Clustermethoden mit Zugehörigkeitsfunktion besteht darin, dass Übergänge zwischen verschiedenen Clustern besser modelliert werden können und somit für den Menschen intuitiv verständlicher sind. Abbildung 4.1 verdeutlicht den Unterschied zwischen Clustering mit „harten“ ($w_{i,k} \in \{0,1\}$) und „weichen“ ($w_{i,k} \in [0,1]$) Zugehörigkeiten.



(a) Clustering mit „harten“ Zugehörigkeiten (b) Clustering mit „weichen“ Zugehörigkeiten

Abbildung 4.1.: Clustering mit „harten“ und „weichen“ Zugehörigkeiten.

Sei $X = \{\boldsymbol{x}_i\}\ , i = 1, \ldots, n$ die Menge von Eingabemustern und $P = \{\boldsymbol{p}_k\}\, , k = 1, \ldots, c$ die Menge der c Prototypen. W sei die Matrix mit den Koeffizienten $w_{i,k}$, welche die Zugehörigkeit des Musters $\boldsymbol{x}_i$ zu Cluster k bezeichnen. Der fuzzy c-means Algorithmus minimiert schrittweise die folgende Zielfunktion:

$$J(X, P, W) = \sum_{i=1}^{n} \sum_{k=1}^{c} w_{i,k}^{m} d(\boldsymbol{p}_k, \boldsymbol{x}_i)^2 \tag{4.3}$$

$m \in (1, \infty)$ ist der Fuzzifierungs-Parameter, der bestimmt, in welchem Ausmaß die Cluster überlappen dürfen. Ein Wert nahe 1 führt zu einer scharfen, ein größerer Wert zu einer unscharfen Clustereinteilung. $J(X, P, W)$ wird unter den folgenden Randbedingungen minimiert:

$$\forall i : 0 < \sum_{k=1}^{c} w_{i,k} = 1 \tag{4.4}$$

Dies bewirkt, dass jedem Cluster Muster mit dem gleichen Gewicht zugeordnet werden. Die Minimierung der Zielfunktion stellt ein nicht lineares Optimierungsproblem dar. Deshalb werden abwechselnd die Zugehörigsgrade und die Cluster Prototypen optimiert, wobei jeweils der andere Parametersatz als fest angesehen wird. Die Zugehörigkeitsgrade werden folgendermaßen berechnet:

$$w_{i,j} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d(\boldsymbol{x}_i, \boldsymbol{p}_j)}{d(\boldsymbol{x}_i, \boldsymbol{p}_k)} \right)^{\frac{2}{m-1}}} \tag{4.5}$$

Die Cluster-Zentren werden als Schwerpunkt der ihnen zugeordneten Daten berechnet:

$$\boldsymbol{p}_k = \frac{\sum_{i=1}^{n} w_{i,p}^m \boldsymbol{x}_i}{\sum_{i=1}^{n} w_{i,p}^m} \tag{4.6}$$

Die Cluster Prototypen oder die Zugehörigkeitsgrade werden zufällig initialisiert, anschließend werden beide schrittweise optimiert bis das Verfahren konvergiert oder eine vorgegebene Anzahl von Iteration erreicht ist. Da die zufällige Initialisierung dazu führen kann, dass das Optimum der Zielfunktion nicht gefunden wird, wird der Algorithmus oft mit verschiedenen zufälligen Initialisierungen ausgeführt und bewertet.

Ausreißer und Rauschen in den Daten sind ein häufiges Problem in der Datenanalyse. Daher wird in dieser Arbeit eine erweiterte Version des Fuzzy c-means Algorithmus mit Erkennung von Rauschen in den Daten (Dave, 1991) verwendet. Hierbei wird ein zusätzlicher fiktiver Cluster eingefügt, zu dem jedes Muster dieselbe Distanz δ hat. Die Zielfunktion des Fuzzy c-means wird um einen zusätzlichen Term erweitert, welcher die Zuordnungen zum Noise Cluster beschreibt.

$$J(X, P, W) = \sum_{i=1}^{n} \sum_{k=1}^{c} w_{i,k}^m d(\boldsymbol{p}_k, \boldsymbol{x}_i)^2 + \sum_{i=1}^{n} \delta^2 \left(1 - \sum_{k=1}^{c} w_{i,k} \right)^m \tag{4.7}$$

Muster, welche sich nicht den gegebenen normalen c Clustern zuordnen lassen, werden

dem Noise Cluster zugeordnet.

Der Parameter δ hat einen großen Einfluß auf die Zuordnung der Muster zum Noise-Cluster. Wird δ zu klein gewählt, werden zu viele Muster dem Noise Cluster zugeordnet. Ist δ zu groß, wird kein Muster dem Noise Cluster zugeordnet. Daher wird in der Arbeit von (Dave, 1991) vorgeschlagen, δ zu schätzen, indem in jeder Iteration der mittlere Abstand innerhalb der Cluster betrachtet wird:

$$\delta^2 = \lambda \frac{\sum_{i=1}^{n} \sum_{k=1}^{c} d(\boldsymbol{p}_k, \boldsymbol{x}_i)^2}{n \cdot c} \tag{4.8}$$

In diesem Abschnitt haben wir ein Verfahren für ein Clustering mit Prototypen betrachtet. Das Ergebnis ist eine Menge von Prototypen und Zuordnungen der Muster zu diesen Prototypen.

4.2. Klassifikation mit Prototypen

Bei einer Klassifikation mit Prototypen kann man jede Musterklasse $y_i, i = 1, \ldots, c$ durch einen (oder mehrere) repräsentative Prototypen beschreiben. Ein Muster $\boldsymbol{x}$ wird dann klassifiziert, indem ihm die Klasse des nächsten Prototyps $\boldsymbol{p}_i$ zugewiesen wird:

$$y(\boldsymbol{x}) = y_c, \text{ wobei } c = \arg\min_{y(\boldsymbol{p}_i)} d(\boldsymbol{x}, \boldsymbol{p}_i) \tag{4.9}$$

Die Cluster Prototypen werden um ein Klassenlabel erweitert und stellen damit repräsentativ die verschiedenen Klassen in den Daten dar. Die Klassifikation mit dem nächsten Prototyp führt zu einer Partitionierung des Eingaberaumes in Zellen. Allen Mustern innerhalb der Zelle wird die Klassenzugehörigkeit des Prototypen zugewiesen (Voronoi-Mosaik), siehe Abbildung 4.2.

Um die Prototypen für eine Klassifikation anzupassen, stehen verschiedene Verfahren zur Verfügung. Im vorherigen Abschnitt haben wir das Verfahren der alternierenden Optimierung beim FCM Algorithmus für ein Clustering betrachtet. Im folgenden Abschnitt wird ein Verfahren aus dem Bereich des Wettbewerbslernens für die Klassifikation vorgestellt.

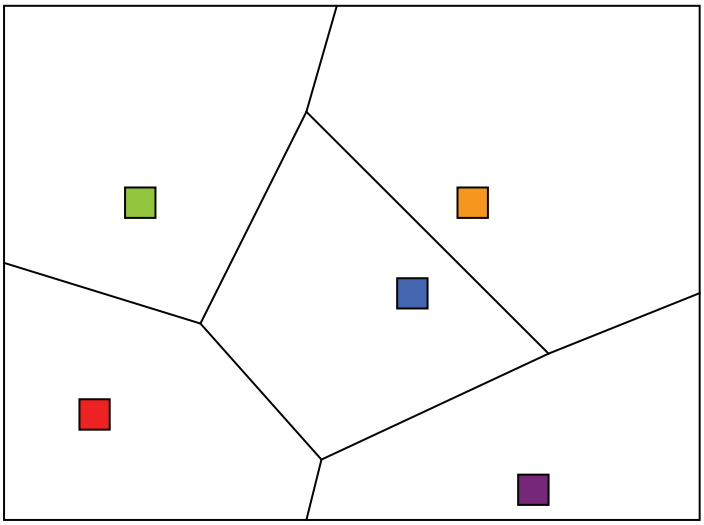


Abbildung 4.2.: Voronoi-Mosaik mit Prototypen und Klassengrenzen.

4.2.1. Lernende Vektor Quantisierung

Lernende Vektor Quantisierung (LVQ) (Kohonen, 2001) gehört zur Kategorie des Wettbewerbslernens in neuronalen Netzen. Beim LVQ Algorithmus handelt es sich um ein 2-Schichten Netzwerk, in dem jedes Input-Neuron mit jedem Output-Neuron verbunden ist. Die Gewichte von den Input-Neuronen zu einem Output-Neuron bilden den Repräsentanten $\boldsymbol{p}_j$, der beim LVQ-Lernen als Codebook-Vektor bezeichnet wird. Für jede Klasse gibt es einen oder mehrere Codebook-Vektoren. In dieser Arbeit bezeichnen wir die Codebook-Vektoren als Prototypen.

Der Pseudocode ist in Algorithmus 4.1 dargestellt. Der Algorithmus arbeitet folgendermaßen: Für jedes Muster wird der nächstgelegene Prototyp bestimmt und dann aktualisiert. Besitzen der Prototyp und das Muster die gleiche Klasse, wird der Prototyp ein kleines Stück in Richtung des Musters bewegt. Falls Prototyp und Muster unterschiedliche Klassen haben, wird der Prototyp ein kleines Stück vom Muster wegbewegt. Dabei wird die Lernrate ϵ in jeder Iteration verringert, so dass sich die Prototypen immer weniger bewegen. Diese Technik ist auch als *Simulated Annealing* (Kirkpatrick u. a., 1983) bekannt.

Zu dem vorgestellten klassischen LVQ-Algorithmus existieren noch weitere Varianten, wie z. B. der LVQ2.1-Algorithmus. Hier werden statt nur einem Sieger die besten zwei Prototypen $\boldsymbol{p}_j$ und $\boldsymbol{p}_k$ ermittelt. Es wird nur dann gelernt, wenn $\boldsymbol{p}_j$ und $\boldsymbol{p}_k$ unterschied-

Algorithmus 4.1 LVQ

1: Wähle c initiale Prototypen für jede Klasse: $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_c$.
2: **repeat**
3: Wähle zufällig ein Muster $\boldsymbol{x}_i$. Sei $\boldsymbol{p}_j$ der nächste Prototyp zu $\boldsymbol{x}_i$. Sei y_i die Klasse von $\boldsymbol{x}_i$ und y_j die Klasse des Prototypen.
4: **if** $y_i = y_j$ **then** // gleiche Klasse
5: Bewege den Prototyp in Richtung des Musters: $\boldsymbol{p}_j \leftarrow \boldsymbol{p}_j + \epsilon(\boldsymbol{x}_i - \boldsymbol{p}_j)$.
6: **end if**
7: **if** $y_i \neq y_j$ **then** // verschiedene Klassen
8: Bewege den Prototyp in die entgegengesetze Richtung des Musters: $\boldsymbol{p}_j \leftarrow \boldsymbol{p}_j - \epsilon(\boldsymbol{x}_i - \boldsymbol{p}_j)$
9: **end if**
10: Verringere die Lernrate ϵ
11: **until** Maximale Anzahl Iterationen erreicht oder keine signifikante Änderung der Prototypen.

liche Klassen haben, das Muster $\boldsymbol{x}_i$ zur Klasse von $\boldsymbol{p}_j$ gehört und zusätzlich in einem Fenster zwischen diesen beiden Klassen liegt:

$$\frac{1+v}{1-v} > \frac{d(\boldsymbol{x}_i, \boldsymbol{p}_j)}{d(\boldsymbol{x}_i, \boldsymbol{p}_k)} > \frac{1-v}{1+v} \tag{4.10}$$

Hier werden also nicht alle Muster zum Lernen verwendet, sondern nur solche Muster, die für die bestehende Klassifikation einen hohen Lerneffekt haben. Diese Idee greifen wir an späterer Stelle für die aktive Lernende Vektor Quantisierung im Kapitel 5 erneut auf.

4.3. Geeignete Attribute und der Fluch der hohen Dimensionen

Wir nehmen an, dass Muster derselben Klasse (bzw. in einem Cluster) ähnlich und Muster unterschiedlicher Klasse (bzw. Cluster) unähnlich sind. Dabei wird die (Un)ähnlichkeit über den Abstand der Muster im Merkmalsraum bestimmt. Dies setzt voraus, dass die beschreibenden Attribute mit dem Zielkonzept kovariieren, d. h. Muster gleicher Klasse sollten im Merkmalsraum möglichst eine geringe Distanz haben und Muster unterschiedlicher Klasse sollten möglichst weit voneinander entfernt sein. Eine gute Klassen-

bzw. Clustereinteilung kann also nur gefunden werden, wenn die verwendeten Attribute zum Zielkonzept passen. Mit zunehmender Anzahl von irrelevanten Attributen, die bezüglich einer Clustereinteilung keine Informationen enthalten, wird es schwieriger, die Clustereinteilung zu lernen. (Döring u. a., 2005) spalten die euklidische Distanz in zwei Terme von relevanten Attributen A_{rel} und irrelevanten Attributen A_{irr} auf:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k \in A_{rel}} d(\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,k}) + \sum_{p \in A_{irr}} d(\boldsymbol{x}_{i,p}, \boldsymbol{x}_{j,p}) \tag{4.11}$$

Anhand dieser Gleichung lässt sich erkennen, dass die Distanzen der irrelevanten Attribute zur Gesamtdistanz addiert werden und somit mit zunehmender Anzahl die Gesamtdistanz erhöhen. Je mehr irrelevante Attribute enthalten sind, umso geringer ist der Anteil der Distanz der relevanten Attribute.

Ein weiteres Problem ist der so genannte „Fluch der hohen Dimensionen“ (Bellman, 1961). Mit zunehmender Anzahl von Attributen (Dimensionen) steigt das Volumen des Eingaberaumes X exponentiell an - die Anzahl der Datenpunkte in X hingegen meistens nicht. Über die Annahme, dass die Datenpunkte gleichmäßig im Datenraum verteilt sind, lässt sich zeigen, dass sich mit zunehmender Anzahl von Dimensionen die Datenpunkte am Rand des Eingaberaumes befinden. Dadurch werden die Distanzen in X immer größer und nähern sich einander an. Somit können sich die Datenpunkte im hochdimensionalen Raum immer weniger zu Clustern zusammenballen.

Für die Anwendung eines Prototypen basierten Cluster- und Klassifikationsschemas ist also zu beachten, dass zum einen möglichst relevante Attribute verwendet werden sollten, und die Anzahl der Attribute möglichst gering ist. Dies lässt sich damit erreichen, dass mit Hilfe von a priori Informationen versucht wird, die relevanten Attribute für das Problem zu bestimmen. Zum Themenbereich der Attributselektion existieren eine Vielzahl von Arbeiten. Eine gute Übersicht findet sich in (Liu und Motoda, 1998). Die Selektion von geeigneten Attributen ist nicht Bestandteil dieser Arbeit. Für die praktische Anwendung in Kapitel 8 wird ein Verfahren zur Selektion von relevanten Attributen vorgestellt.

4.4. Zusammenfassung

In diesem Kapitel haben wir - anhand verschiedener exemplarischer Verfahren für die Initialisierung und die Anpassung - das Clustering und die Klassifikation mit Prototypen

betrachtet. Dabei wurden vor allem die Verfahren betrachtet, welche in den weiteren Kapiteln für die aktiven Lernverfahren verwendet worden sind. Für die Initialisierung von Prototypen existieren noch weitere Verfahren, welche z. B. aus einem probabilistischen Modell der Daten neue Muster selektieren (Langley u. a., 1992) oder ein hierarchisches Clustering durchführen (Johnson, 1967), um aus der Hierarchie eine gewünschte Anzahl von Clustern auszuwählen.

Auch im Bereich der Anpassung der Prototypen existieren noch weitere Verfahren. So kann mit Hilfe eines Gradientenabstiegsverfahrens bzw. mit genetischen oder evolutionären Algorithmen eine Zielfunktion optimiert werden, um somit die Prototypen anzupassen.

Prototypen Modelle eignen sich besonders für einen Überblick des gelernten Modells im Prozess der aktiven Klassifikation. Sind die Prototypen für eine Klassifikation gefunden, bzw. wurden diese aus den Trainingsdaten bestimmt, ist kein weiterer Trainingsaufwand zur Erzeugung des Modells erforderlich.

Prototypen-Modelle sind gut für Mehrklassen-Probleme geeignet, da jede Klasse durch einen Prototyp repräsentiert wird. Im Gegensatz dazu muss bei binären Klassifikatoren ein Klassifikator für jede einzelne Klasse getrennt gelernt werden. Gerade bei aktiven Lernverfahren ist dies von Nachteil, da hier die Selektionsfunktion für jede Klasse ein Muster zur Klassifikation auswählen muss.

Jedoch werden Prototypen-Modelle mit zunehmender Anzahl von Prototypen auch komplexer. Dies kann dazu führen, dass eine Klassifikation eines Musters bei vielen vorhandenen Prototypen mehr Zeit benötigt, da die Abstände zwischen dem Muster und den Prototypen berechnet werden müssen.

5. Aktive Lernende Vektor Quantisierung

In diesem Kapitel stellen wir einen Algorithmus für Aktive Lernende Vektor Quantisierung (ALVQ) vor. Der Algorithmus besteht aus zwei verschiedenen Phasen: Die erste Phase dient der Initialisierung eines stabilen Klassifikators mit Hilfe von Prototypen. Das Auffinden repräsentativer Muster, welche als Prototypen verwendet werden können, geschieht durch ein Clustering mit dem Fuzzy c-means Algorithmus, welcher im vorherigen Kapitel in Abschnitt 4.1 vorgestellt wurde. In Abschnitt 5.1 wird die Initialisierung einer Klassifikation basierend auf den gefundenen Cluster-Zentren beschrieben.

Um die Klassifikation anzupassen, wird in der zweiten Phase des Verfahrens der LVQ-Algorithmus zur Adaption der Prototypen verwendet, welcher in Abschnitt 4.2 beschrieben wurde. Da dieser üblicherweise eine große Menge von klassifizierten Daten voraussetzt, wird in Abschnitt 5.3 ein Verfahren zur aktiven Selektion weiterer Trainingsmuster vorgestellt. Der komplette Algorithmus zum Aktiven Lernen, der sich aus den vorgestellten Verfahren zusammensetzt, wird in Abschnitt 5.4 beschrieben. Abschnitt 5.5 fasst die Vor- und Nachteile des Verfahrens zusammen.

5.1. Fuzzy Clustering und Klassifikation

Da wir in dieser Arbeit davon ausgehen, dass zu Beginn keine klassifizierten Trainingsdaten zur Verfügung stehen, muss der Datensatz alleine mit Strukturinformationen und der Hilfe eines Orakels klassifiziert werden. Als Strukturinformation für besonders repräsentative Muster werden die mit dem im vorherigen Abschnitt 4.1 beschriebenen Fuzzy c-means Clusterverfahren gefundenen Cluster-Zentren verwendet.

Die im Fuzzy c-means gewonnenen Zugehörigkeitsgrade werden direkt für die Klassifikation verwendet. Nachdem für jedes Cluster-Zentrum durch das Orakel ein Klassenlabel zugeordnet wurde, werden die Cluster-Zugehörigkeitsgrade des Musters für jede Klasse

aufsummiert.

Das Cluster-Zentrum, welches aus den ihm zugeordneten Daten berechnet wird, ist streng genommen ein künstlich generiertes Muster, ähnlich zum selective sampling Ansatz (Angluin, 1988). Da in der unmittelbaren Nachbarschaft des Cluster-Zentrums viele Muster verfügbar sind, kann jedoch stattdessen das Muster mit dem kleinsten Abstand zum Cluster-Zentrum gewählt werden.

Die gefundenen Cluster sollten die inhärente Klassifikation der Daten wiedergeben. Jedoch sind Klasse und Cluster nicht in allen Fällen gleichzusetzen - ein Cluster wird durch eine Menge von Mustern charaktisiert, die möglichst homogen ist, wohingegen eine Klasse eine Menge von Mustern mit einem gemeinsamen Attribut darstellt. Daher müssen die Muster in einer Klasse nicht zwangsläufig homogen sein und eine Klasse kann aus mehreren Clustern bestehen.

Diese grobe Kategorisierung ist daher meistens nicht sehr genau, da die Struktur des Clusterings nicht unbedingt der Klassenverteilung im Datensatz entsprechen muss. Um also die aktuelle Cluster-Klassen-Hypothese zu überprüfen, wenden wir ein Verfahren an, welches als *Cluster Mean Selection* (Gabrys und Petrakieva, 2004) bezeichnet wird. Hierbei wird jeder bereits gefundene Cluster erneut in Sub-Cluster aufgespaltet und die Sub-Cluster-Zentren werden klassifiziert. Bestätigen die vergebenen Label die aktuelle Cluster-Hypothese, bleibt der Prototyp des gesamten Clusters erhalten. Ansonsten werden die Sub-Cluster erneut aufgespalten und der Prozess beginnt von vorne, bis eine vorgegebene Rekursionstiefe erreicht ist. Diese Technik kann rekursiv ausgeführt werden, solange die Label innerhalb eines Clusters inhomogen sind. Dabei ist jedoch zu berücksichtigen, dass die Anzahl der Muster pro Ebene exponentiell steigen kann und diese Phase nur der Initialisierung dient. Abbildung 5.1 illustriert den Initialisierungsprozess mit aufgespaltenen Clustern und den korrespondierenden Anfrage-Bäumen. Die letztendlich verwendeten Prototypen sind grau umrandet.

Dieses Verfahren schließt die erste Phase des Algorithmus ab. Durch das Auffinden repräsentativer Muster lässt sich ein stabiles Modell initialisieren, welches zur Klassifikation verwendet werden kann. Allerdings spiegelt dieses Modell nur eine grobe Klassifikation wider. Nachdem man zusätzliche Muster zwischen diesen Prototypen - also in der Region der größten Unsicherheit des Klassifikators - gesehen und klassifiziert hat, müssen die Prototypen angepasst werden, um eine bessere Klassifikation zu ermöglichen. Da der Fuzzy c-means Algorithmus diese Möglichkeit nicht direkt bietet, wird zur Anpassung der Prototypen der LVQ-Algorithmus verwendet, um die Prototypen zu verschieben.

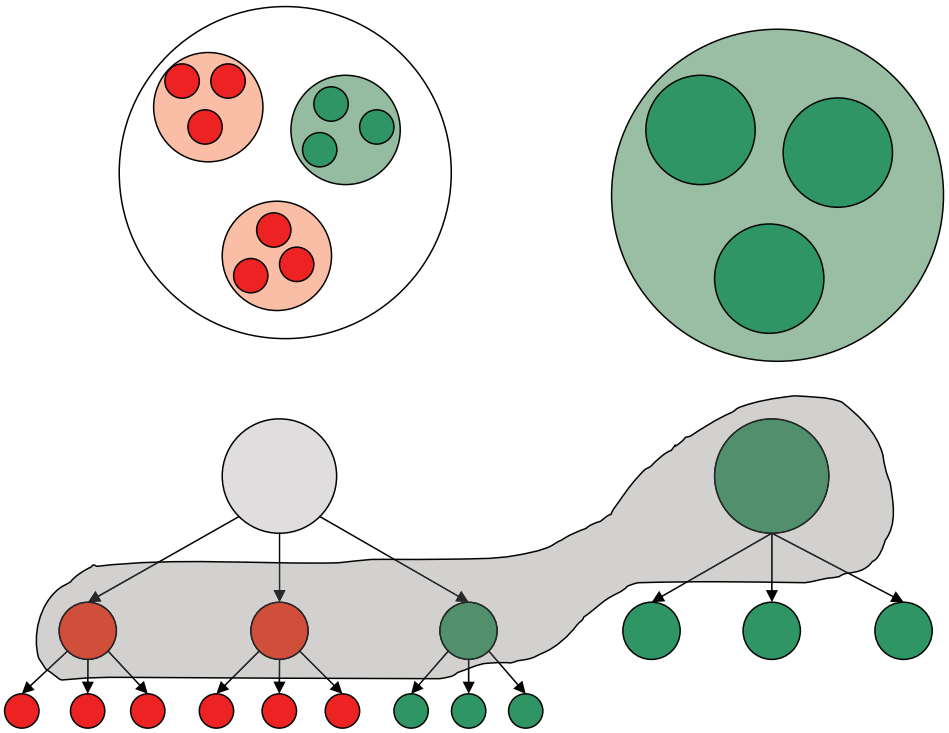


Abbildung 5.1.: Beispiel Initalisierungsprozess ALVQ.

5.2. Lernende Vektor Quantisierung

Zur Anpassung der Prototypen wird der in Abschnitt 4.2 beschriebene LVQ Algorithmus verwendet. Eine wichtige Anforderung im LVQ-Algorithmus ist jedoch die Verfügbarkeit der Klassenlabel für alle Muster.

Beim Aktiven Lernen stehen für die Muster zu Beginn keine Klassenlabel zur Verfügung. Jedoch kann das Orakel für wenige ausgewählte Muster die Klasse bestimmen.

In zwei Punkten muss der LVQ Algorithmus daher an die Bedingungen beim Aktiven Lernen angepasst werden:

1. Für die Initialisierung gibt es normalerweise zwei verschiedene Möglichkeiten: Die erste Möglichkeit besteht darin, die Prototypen zufällig zu initialisieren. Dies birgt jedoch das Risiko, dass manche Prototypen nie Sieger werden und die zufällige Klassenzuordnung nur unzureichend mit der Eingabemenge übereinstimmt. Die zweite Möglichkeit besteht darin, die Prototypen aus den zugrunde liegenden klassifizierten Eingabedaten auszuwählen. Da in diesem aktiven Lernansatz keine oder

nur sehr wenige klassifizierte Muster zur Verfügung stehen, müssen die Prototypen auf eine andere Weise initialisiert werden. In diesem Fall werden die Prototypen mit den in Abschnitt 5.1 gelabelten Cluster-Prototypen initialisiert.

2. Normalerweise stehen im LVQ-Algorithmus für alle Muster aus der klassifizierten Trainingsmenge die Label zur Verfügung. Beim Aktiven Lernen ist die Menge der gelabelten Muster sehr klein und wird zu Beginn für die Initialisierung der Prototpyen verwendet. Die Auswahl von Mustern, die vom Orakel klassifiziert und für das LVQ-Training verwendet werden, muss also mit großer Sorgfalt erfolgen.

5.3. Auswahl von Trainingsmustern

Die Auswahl der Muster spielt eine besondere Rolle beim LVQ-Algorithmus. Diese Muster stellen das Modell der Daten für den Algorithmus dar und müssen deshalb sorgfältig ausgewählt werden. Dabei müssen zwei Aspekte betrachtet werden:

1. Exploration der Daten, d. h. Verifikation der bestehenden Hypothese
2. Spezialisierung (engl: exploitation) der Klassifikation, indem man bestehende Entscheidungsgrenzen verfeinert

Da beim Aktiven Lernen die Klassen von einem Orakel zugewiesen werden, ist es sehr wichtig, dass mit wenigen Mustern so viele Informationen wie möglich gewonnen werden. Die Idee hinter der Selektionsstrategie ist, die Verteilung der Daten zu berücksichtigen. Diese Idee findet sich bereits in der Arbeit von (Cohn u. a., 1994b) wieder, welche schon in Abschnitt 3.4.4 beschrieben wurde. Wir wiederholen die zugrunde liegende Gleichung 3.13:

$$\arg\max_{\boldsymbol{x}\in U} E[(\hat{y}(\boldsymbol{x}) - y(\boldsymbol{x}))^2|\boldsymbol{x}]p(\boldsymbol{x}) \tag{5.1}$$

Zwei Aspekte spielen nach der Gleichung 3.13 eine entscheidende Rolle: der Repräsentationsgrad des Musters, der sich in $p(\boldsymbol{x})$ wiederfindet, und die Unsicherheit des Klassifikators, die sich im erwarteten Fehler $E[(\hat{y}(\boldsymbol{x}) - y(\boldsymbol{x}))^2|\boldsymbol{x}]$ widerspiegelt. Diese zwei Aspekte lassen sich auch als Exploration (Auffinden von Klassen aufgrund repräsentativer Muster) und Ausnutzung von bestehendem Modellwissen (Exploitation) beschreiben.

Dies lässt sich auch in zwei separate Schritte aufteilen: Zuerst wird ein Modell mit besonders repräsentativen Mustern gelernt und anschließend wird das gelernte Modell mit Mustern zwischen den Klassengrenzen verfeinert. Die Initialisierung durch repräsentative Muster wurde bereits weiter oben behandelt, daher befassen wir uns mit der Auswahl

von Mustern zwischen den Klassengrenzen.

Von Mustern, welche in der Region der größten Unsicherheit zwischen zwei Clustern unterschiedlicher Klasse liegen, wird der größte Informationsgewinn erwartet. Da durch das Sub-Clustering in der Initialisierung die Hypothese im Bereich des Prototypen bereits als verifiziert gilt, verspricht man sich von Mustern in der Nähe des Prototypen keinen Informationsgewinn. Wird in jeder Iteration des LVQ Algorithmus der Prototyp bewegt, liefert ein Muster zwischen zwei Prototypen einen maximalen Lerneffekt - gemessen an der Verschiebung der Entscheidungsgrenze durch die Verschiebung des Prototypen.

Abbildung 5.2 stellt die Region der größten Unsicherheit anschaulich dar. Sie liegt an der Klassifikationsgrenze zwischen zwei Prototypen (Kreis und Rechteck) unterschiedlicher Klassen. Die Fragezeichen stellen die bislang unklassifizierten Muster dar. Nimmt man an, dass die unklassifizierten Muster mit geringer Distanz zum Prototypen eine hohe Wahrscheinlichkeit für das Auftreten dieser Prototyp-Klasse besitzen, haben die Muster zwischen den Prototypen einen höheren Lerneffekt.

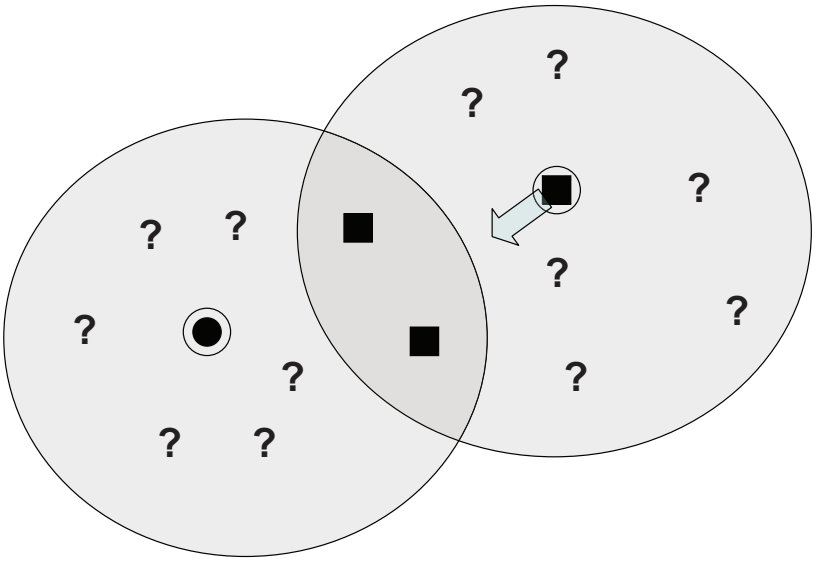


Abbildung 5.2.: Region der größten Unsicherheit zwischen zwei Prototypen.

An dieser Stelle wollen wir ein Verfahren vorstellen, das geeignet ist, Muster zwischen zwei Clustern zu finden, und welches sich besonders einfach einsetzen lässt, wenn die Daten zuvor mit dem Fuzzy c-means Algorithmus geclustert worden sind. Dabei soll in jeder Iteration nicht nur ein Muster, sondern eine Menge von N Mustern ausgewählt werden. Bei der Definition der Zielfunktion des Fuzzy c-means Algorithmus bezeichnet

$w_{i,k}$ die Zugehörigkeit von $\boldsymbol{x}_i$ zum Cluster k. Da die Muster aufgrund ihrer Zugehörigkeitsgrade zu den Cluster Prototypen klassifiziert werden, spiegeln diese auch die Klassifikations(un)sicherheit wider. Wenn also Muster identifiziert werden sollen, die eine hohe Klassifikationsunsicherheit aufweisen, so müssen diese eine fast gleiche Zugehörigkeit zu zwei Clustern unterschiedlicher Klasse haben.

Definition 1. *Für jeden Datenpunkt, der zwischen zwei Clustern unterschiedlicher Klassen liegt, lässt sich sein Rang folgendermaßen berechnen:*

$$Rang(\boldsymbol{x}_i) = 1 - (\min |w_{i,k} - w_{i,l}|) \quad \forall k, l = 1, \ldots c \quad w_{i,k}, w_{i,l} \geq \frac{1}{c} \quad k \neq l \tag{5.2}$$

Wie in Abbildung 5.3 deutlich wird, treten solche interessanten Regionen lokal begrenzt auf und verteilen sich nicht gleichmäßig im Datenraum. Abbildung 5.3a zeigt die Rangordnung der Muster, Abbildung 5.3b die selektierten Muster, welche einen Schwellwert von 0.9 überschreiten. Wird wie in Abbildung 5.3c nur nach Rangordnung selektiert, kann es passieren, dass manche Regionen mit hohem Rang nicht berücksichtigt werden.

Es besteht daher die Gefahr, dass nur Muster aus einer Region im Datenraum ausgewählt werden, wenn die Rangordnung als einziges Kriterium verwendet wird. Um dieses Problem zu umgehen, wird ein weiterer Selektionsschritt ausgeführt, der eine gute Streuung der Muster gewährleisten soll.

Hierzu wird zuerst eine Menge von Mustern mit hohem Rang ausgewählt. Unter diesen Mustern wird anschließend eine Anzahl von möglichst unterschiedlichen Mustern ausgewählt. Dazu wird das in Abschnitt 4.1.1 vorgestellte *Farthest-First-Traversal*-Verfahren von (Hochbaum und Shmoys, 1985) auf den selektierten Daten angewendet. Dabei werden die unterschiedlichsten Muster selektiert, indem das erste Muster zufällig gewählt wird und jedes weitere Muster das am weitesten entfernte von der Menge der bisher selektierten Muster ist. Abbildung 5.3d zeigt die Auswahl von Punkten mit hohem Rang und hoher Diversität.

5.4. Aktive Lernende Vektor Quantisierung

In diesem Abschnitt fassen wir die bisher behandelten Techniken zu einem adaptiven aktiven Klassifikationsschema zusammen. Die einzelnen Schritte sind in Algorithmus 5.1 dargestellt. Der Algorithmus ist in zwei Phasen unterteilt. Die erste Phase umfasst die Exploration der Daten und eine initiale Klassifikation. In der zweiten Phase wird die

Algorithmus 5.1 Aktive Lernende Vektor Quantisierung

1: $L \leftarrow 0$
2: **while** Unterschiedliche Klassen im aktuellen Cluster **do**
3: Teile den Cluster in l Sub-Cluster mit dem Fuzzy c-means Algorithmus auf.
4: Bestimme das Klassenlabel der Cluster-Zentroiden.
5: Füge das klassifizierte Muster zur Menge der Prototypen P hinzu.
6: **end while**
7: **while** Lernprozess durch Evaluation nicht beendet **do**
8: $T \leftarrow$ Wähle m Trainingsmuster an den Clustergrenzen nach Gleichung 5.2.
9: Wähle n voneinander verschiedene Muster aus T mittels dem *Farthest-First-Traversal*-Verfahren.
10: Bestimme das Klassenlabel der Muster und füge diese zu den Trainingsmustern L hinzu.
11: Trainiere die Prototypen P mit Hilfe der Trainingsmuster L.
12: Verringere die Lernrate ϵ.
13: **end while**

initiale Klassifikation durch weitere Muster verfeinert.

Den Ausgangspunkt für eine initiale Klassifikation bildet das FCM Clustering. Wir unterstellen den Strukturen in den Daten eine gewisse Semantik für die Klassifikation, d. h. Muster, die im Datenraum nahe beieinander liegen, sind auch sehr wahrscheinlich in derselben Klasse. Das Fuzzy Clustering fasst die Daten zusammen und verhindert so unnötige Abfragen. Insbesondere die Variante mit der Erkennung von Rauschen hilft dabei, seltene Muster zu erkennen und herauszufiltern, die für die Gesamtklassifikation keine Relevanz besitzen. Der Datensatz wird geclustert und die Prototypen werden durch das Orakel klassifiziert. Anschließend wird überprüft, ob die Klassen in einem Cluster homogen sind.

Nach der ersten initialen Klassifikation wird in der zweiten Phase versucht, diese zu verbessern, indem die Prototypen analog zum LVQ-Verfahren weiter verschoben werden. Die dazu benötigten Muster werden nach dem in Abschnitt 5.3 dargestellten Verfahren selektiert.

Der Vorgang kann beliebig oft wiederholt werden, wobei der Benutzer ihn abbrechen kann, wenn eine akzeptable Klassifikationsgüte erreicht worden ist. Im Folgenden wollen wir noch verschiedene Kriterien zur Evaluation dieser Klassifikationsgüte vorstellen:

Clustergütemaße geben uns Informationen über die Qualität des Clusterings. Wir setzen hier Gütemaße für die Variation innerhalb der Cluster und zwischen den Clustern ein (Windham, 1981). Diese Indikatoren erweisen sich als nützlich, um zu Be-

ginn die richtigen Attribute im Datensatz zu wählen, so dass die Struktur der Daten die Semantik der Klassifikation widerspiegelt. Mit zunehmender Anzahl von Klassifikationsschritten nimmt die Bedeutung dieser Deskriptoren naturgemäß leicht ab.

Gradient die im bisherigen Klassifikationsprozess erhaltenen klassifizierten Muster können zum Testen verwendet werden. Nachdem die Klassen für die Muster in und zwischen den Clustern bestimmt worden sind, werden die Prototypen angepasst. Dadurch ergibt sich eine neue Klassifikation der Daten. Das Verhältnis von korrekt klassifizierten Mustern (auf Basis der bisher klassifizierten Muster) zu falsch klassifizierten Mustern kann als Indikator verwendet werden.

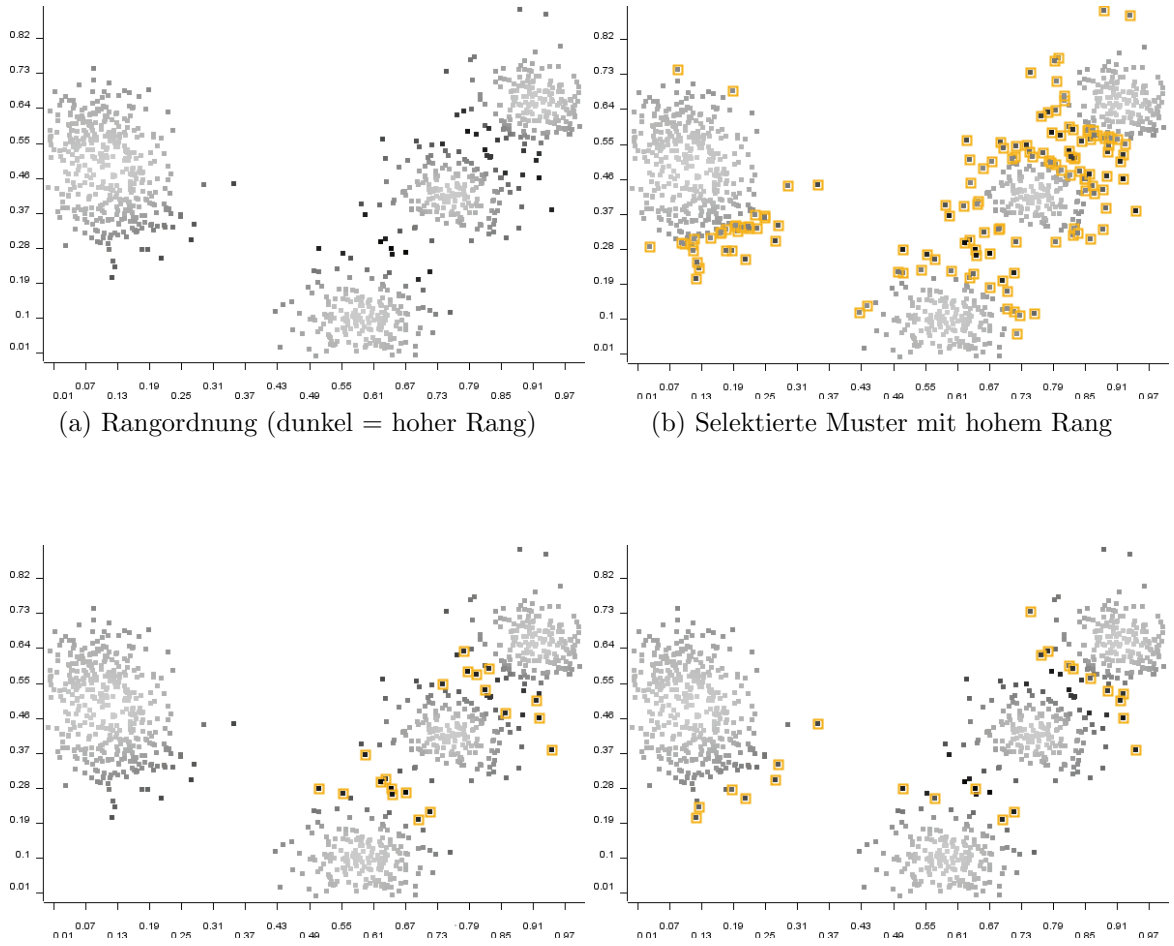
5.5. Fazit

In diesem Kapitel haben wir einen neuen Algorithmus zum Lernen einer Klassifikation mit wenigen klassifizierten Mustern vorgestellt. In einer ersten Explorationsphase werden durch die Berücksichtigung der Verteilung der Eingabedaten repräsentative Muster zur Klassifikation ausgewählt. Dadurch wird das Problem der meisten aktiven Lernverfahren überwunden, ein initiales stabiles Klassifikationsmodell zu lernen. Der Nutzen gegenüber einer zufälligen Initialisierung in anderen Verfahren wird im Ergebnis-Kapitel aufgezeigt.

In der zweiten Phase werden die bereits gewonnenen Fuzzy-Zugehörigkeiten verwendet, um Muster an den Klassifikationsgrenzen aufzufinden. Diese Methode lässt sich in den Bereich des *Uncertainty sampling* (vgl. Abschnitt 3.4.3) einordnen. Da in jeder Iteration mehrere Muster aus dem Pool der unklassifizierten Muster ausgewählt werden, werden die Muster nicht nur nach ihrer Klassifikationsunsicherheit eingeordnet, sondern zusätzlich möglichst gleichmäßig im Datenraum verteilt.

Der Nachteil dieser Methode besteht in der starren Unterteilung in zwei Phasen. Zu Beginn steht noch nicht fest, wie viele Cluster benötigt werden, um eine ausreichende Exploration zu gewährleisten. Dies wird durch das Sub-Clustering kompensiert, erhöht jedoch die Anzahl „unnötiger“ Abfragen, falls die Klassen der Sub-Cluster Prototypen bereits homogen sind.

Im nächsten Abschnitt wird ein Verfahren vorgestellt, dass diese Nachteile kompensiert.



(a) Rangordnung (dunkel = hoher Rang)

(b) Selektierte Muster mit hohem Rang

(c) 20 Muster nach Rang

(d) 23 Muster nach Rang und Diversität

Abbildung 5.3.: Hoher Rang und Diversität.

6. Prototypen Basierte Aktive Klassifikation

Mit dem Verfahren der Prototypen basierten aktiven Klassifikation (PBAC) sollen die Nachteile des Verfahrens der aktiven Lernenden Vektor Quantisierung überwunden werden. Anstelle separater Phasen zur Exploration und zur Verfeinerung der Klassifikation tritt in diesem Ansatz ein einziges Kriterium zur Auswahl von Prototypen, welche für eine k-nächste Nachbarn Klassifikation verwendet werden. Das Kriterium, welches als Unsicherheits-Verteilung D bezeichnet wird, integriert die lokale Dichte mit der Unsicherheit des aktuell gelernten Klassifikators (siehe Abbildung 6.1). In jeder Iteration des Aktiven Lernens reduziert sich der Explorations-Anteil auf natürliche Weise, somit wächst der Einfluss der Klassifikator-Unsicherheit.

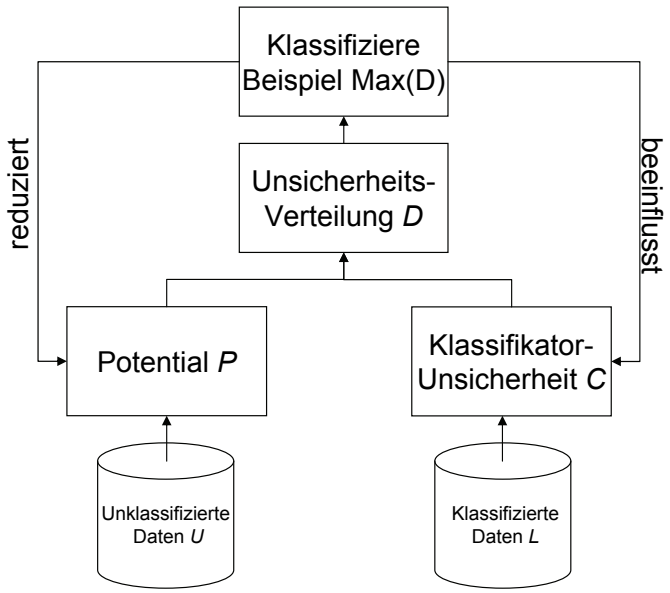


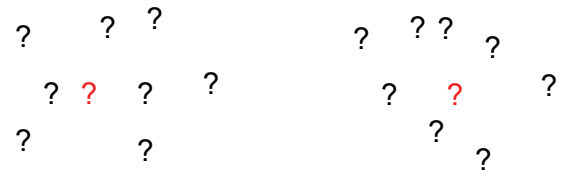
Abbildung 6.1.: Interaktion von Potential, Klassifikator-Unsicherheit und der daraus resultieren Unsicherheits-Verteilung.

6.1. Intuitive Überlegungen

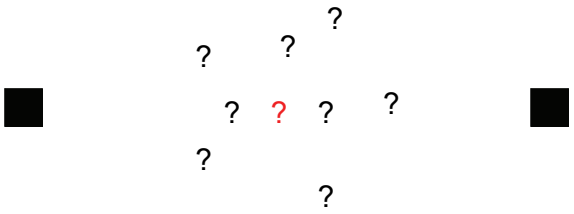
Bevor der Algorithmus zur Prototypen basierten aktiven Klassifikation vorgestellt wird, sollen an dieser Stelle intuitive Überlegungen für verschiedene Fälle dargestellt werden, die beim Klassifikationsprozess auftreten können. Wir gehen von einer Momentaufnahme aus, in der unklassifizierte Muster und evtl. schon klassifizierte Muster in Form von Prototypen vorliegen. In jedem Schritt soll ein Muster gewählt werden, welches dann klassifiziert und als Prototyp eingefügt wird. Wir gehen diesmal nicht davon aus, dass durch ein globales Cluster-Verfahren repräsentative Muster ausgewählt worden sind, sondern beschäftigen uns direkt mit der Auswahl von Mustern aufgrund unterschiedlicher Kriterien.

Abbildung 6.2a zeigt ein 2-dimensionales künstliches Beispiel für den Beginn einer Klassifikation. Die unklassifizierten Muster werden mit einem Fragezeichen dargestellt. Wie auch schon im vorherigen Ansatz in Kapitel 5 verspricht man sich von den Repräsentanten der Cluster-Zentren den größten Informationsgewinn. Wichtig ist dabei, dass verhindert wird, dass der zweite Prototyp in der Nähe des ersten Prototypen platziert wird, um so eine bessere Exploration der Daten zu gewährleisten. Auch wenn bereits zwei klassifizierte Prototypen derselben Klasse existieren, wie in Abbildung 6.2b dargestellt, macht eine Exploration eines Clusters zwischen diesen Klassen Sinn, da ein Prototyp an dieser Stelle einen großen Einfluss auf die Daten hat.

Die Region der größten Unsicherheit liegt zwischen zwei Prototypen unterschiedlicher Klasse. Ein Muster, das genau an dieser Stelle liegt, verspricht den größten Informationsgewinn wie in Abbildung 6.2c dargestellt. Von Mustern, die näher an einem Prototypen liegen, wird erwartet, dass diese mit einer höheren Wahrscheinlichkeit die gleiche Klasse wie der Prototyp haben. Daher verspricht man sich von diesen Punkten einen geringeren Informationsgewinn. Trotzdem lohnt es sich in bestimmten Fällen, wie z. B. in Abbildung 6.2d, nicht nur die Klassifikationsunsicherheit zu verwenden, sondern auch die Verteilung der Muster zu berücksichtigen. Hier spielt wieder der Einfluss des Prototypen auf die unklassifizierten Daten eine wichtige Rolle. Würde man das einzelne Muster direkt an der Klassifikationsgrenze wählen (und dieses zur Klasse „Kreis“ gehören), wäre die Unsicherheit im Cluster nach wie vor hoch.

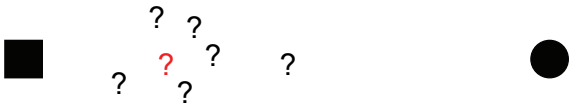


(a) Zwei Cluster zur Exploration



(b) Cluster zwischen Prototypen der gleichen Klasse

(c) Datenpunkt zwischen zwei unterschiedlichen Prototypen



(d) Cluster und Datenpunkt zwischen zwei unterschiedlichen Prototypen

Abbildung 6.2.: Unterschiedliche Konfigurationen von Prototypen und Verteilungen der Muster. Die gewählten Muster sind rot markiert.

6.2. Exploration mit Subtractive Clustering

In dieser Arbeit gehen wir davon aus, dass zu Beginn des Lernprozesses keine klassifizierten Muster zur Verfügung stehen $|L| = 0$. Wir benötigen daher eine Strategie, um aus den unklassifizierten Daten U Muster auszuwählen. Aus der Perspektive der Exploration

sollen bislang unbekannte Regionen im Datensatz erforscht werden, um so die möglichen Klassen zu bestimmen.

Ein Kriterium hierfür ist der Repräsentationsgrad eines Musters, da selten auftretende Muster für die Konstruktion eines globalen Modelles nicht viel Information enthalten[1].

Daher wird für jedes Muster das Potential, wie in Abschnitt 4.1.2 beschrieben, nach Gleichung 4.1 berechnet. Um die Berechnung des Potentials auch für große Datensätze zu ermöglichen, lassen sich drei Maßnahmen anwenden:

1. Alle Prototypen, die komplett von Prototypen der gleichen Klasse umgeben sind, können eliminiert werden (ändert Entscheidungsgrenzen nicht).
2. Die Berechnung des Potentials beschränkt sich nur auf die Muster, die innerhalb des Radius r_a liegen. Muster, welche außerhalb dieses Radius liegen, haben wenig oder keinen Einfluss auf das Potential.
3. Um die unmittelbaren Nachbarn innerhalb einer vorgegebenen Distanz zu finden, lassen sich effiziente Datenstrukturen wie z. B. der KD-Tree (Bentley, 1975) verwenden.

Daraus ergibt sich folgende Definition für das Potential eines Datenpunktes:

Definition 2. *Sei $N(\boldsymbol{x}_i)$ die durch den Radius r_a definierte Nachbarschaft des Datenpunktes $\boldsymbol{x}_i$. Dann wird sein Potential folgendermaßen berechnet:*

$$P(\boldsymbol{x}_i) = \sum_{\boldsymbol{x}_j \in N(\boldsymbol{x}_i)} e^{-\alpha d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2} \tag{6.1}$$

Der Vorteil der hier vorgestellten Methode zur Schätzung der lokalen Dichte im Vergleich zu anderen Verfahren wie z. B. der Parzen-Window Technik (Parzen, 1962) besteht in der schnellen Berechenbarkeit. Die Reduktion der Potentiale spielt in dem hier entwickelten Algorithmus eine große Rolle. Sie erlaubt den Übergang von der Exploration zur Verfeinerung der Klassengrenze, welche in den nächsten Abschnitten beschrieben wird. Zunächst betrachten wir jedoch die k-nächste Nachbarn Klassifikation, welche mit den in diesem Abschnitt gewählten Mustern erstellt werden kann.

[1] Bei realen Applikationen können solche Muster für das Orakel natürlich dennoch interessant sein.

6.3. k-nächste Nachbarn Klassifikation

Das Verfahren der k-nächsten Nachbarn Klassifikation wird in der Literatur als „parameterfreie Methode“ bezeichnet. Das bedeutet, dass die Modellstruktur nicht a priori festgelegt ist, sondern aus den Daten bestimmt wird[2]. Der k-nächste Nachbarn Klassifikator ist ein so genannter *lazy* Lerner, d. h. er lernt indem er die Muster der Trainingsmenge abspeichert und dann zur Klassifikation verwendet.

Streng genommen ist die Methode jedoch nicht parameterfrei, denn der Parameter k beeinflusst die Klassifikationsentscheidung. Zur Bestimmung der Klasse von neuen Mustern wird eine Mehrheitsentscheidung unter den k-nächsten Nachbarn getroffen. Sei $y(\boldsymbol{p}_c)$ die Klasse des Prototypen $\boldsymbol{p}_c$. Dann wird die Klasse $\hat{y}(\boldsymbol{x}_q)$ für ein neues Beispiel $\boldsymbol{x}_q$ folgendermaßen bestimmt:

$$\begin{aligned} \hat{y}(\boldsymbol{x}_q) &= \max_v \sum_{c=1}^{k} \delta(v, y(\boldsymbol{p}_c)) \\ \delta(a,b) &= \begin{cases} 1, & \text{falls } a = b \\ 0 & \text{sonst} \end{cases} \end{aligned} \tag{6.2}$$

Zur Bestimmung der k-nächsten Nachbarn können verschiedene Abstandsmaße verwendet werden. In dieser Arbeit wird der euklidische Abstand verwendet. Falls $k = 1$ ergibt sich für die Klassifikation ein Voronoi-Diagramm. Bei kleinen k besteht die Gefahr, dass ein Rauschen in den Trainingsdaten das Klassifikationsergebnis zu stark beeinflusst. Bei großen k besteht die Gefahr, Prototypen mit zu hohem Abstand in die Klassifikationsentscheidung mit einzubeziehen. Um dieses Problem zu mindern, wird der gewichtete k-nächste Nachbarn Algorithmus verwendet. Dabei werden nähere Prototypen höher gewichtet als weiter entfernte.

$$\hat{y}(\boldsymbol{x}_q) = \max_v \sum_{c=1}^{k} w_c \delta(v, y(\boldsymbol{p}_c)) \tag{6.3}$$

wobei

$$w_c = \frac{1}{d(\boldsymbol{x}_q, \boldsymbol{p}_c)} \tag{6.4}$$

Die Wahl eines geeigneten k basiert üblicherweise auf empirischen Untersuchungen.

[2]Dies bedeutet nicht, dass keine Parameter existieren. Vielmehr werden Art und Anzahl der Parameter erst während des Lernens bestimmt.

Dabei wird z. B. mittels der Methode der Kreuzvalidierung (siehe Abschnitt 3.4.5) versucht, den optimalen Wert für k zu schätzen. Mit verschiedenen Werten für $k = 1, \ldots, K$ bis zu einem vorgegebenen Maximalwert K werden die Trainingsdaten klassifiziert. Der Wert für k, für den die meisten Trainingsmuster korrekt klassifiziert wurden, wird anschließend verwendet.

Im Bereich des Aktiven Lernens steht man vor dem Problem, dass keine klassifizierten Trainingsdaten zur Verfügung stehen, um den optimalen Parameterwert für k zu schätzen. Um ein Unentschieden bei der Klassifikation zu vermeiden, wird ein ungerader Wert für k gewählt, $k = 3$ und $k = 5$ sind häufig verwendete Werte (Manning u. a., 2008). Da die Klassifikation mit den Abständen zum Trainingsmuster gewichtet wird, wird in dieser Arbeit durchgängig der Parameter $k = 5$ verwendet.

Der Vorteil des k-nächsten Nachbarn Klassifikators besteht in seiner Einfachheit des Modells und der genauen Modellierung von Klassengrenzen. Der Nachteil bei der k-nächsten Nachbarn Klassifikation ist überlicherweise die hohe Anzahl von Mustern, die gespeichert werden müssen. Dies kann sowohl zu einem hohen Speicherbedarf als auch zu einer hohen Komplexität bei der Klassifikation von neuen Mustern führen. In dieser Arbeit basiert der k-nächste Nachbar Klassifikator nur auf den gewählten Prototypen, so dass dieser Nachteil ausgeglichen wird.

6.4. Bestimmung der Klassifikationsunsicherheit

Über die Gewichte zu den k nächsten Prototypen lässt sich die Unsicherheit des Klassifikators bestimmen. Wir bestimmen für jede Klassenausprägung y_i die Klassenwahrscheinlichkeit $c(y_i)$ als Summe der Gewichte der Prototypen gleicher Klasse:

$$c(y_i) = \sum_{c=1}^{k} w_c \delta(y_i, f(\boldsymbol{p}_c)) \tag{6.5}$$

Es sollen die Muster ausgewählt werden, welche den größten Beitrag zum erwarteten Fehler haben. Diese Muster haben eine möglichst gleichverteilte Wahrscheinlichkeit, zu unterschiedlichen Klassen zu gehören. Durch eine einfache Überprüfung der Klassenwahrscheinlichkeiten jedes Musters lassen sich diese Punkte identifizieren. Die Klassifikationsunsicherheit C wird über das Histogramm der Klassenwahrscheinlichkeiten berechnet. Eine Verteilung der Klassenwahrscheinlichkeiten mit scharfen Spitzen hat eine niedrige Entropie. Eine gleichverteilte Wahrscheinlichkeit über alle Klassen hingegen hat

eine hohe Entropie.

Definition 3. *Seien $c(y_1), \ldots, c(y_m)$ die Klassenwahrscheinlichkeiten für alle m Klassenausprägungen für das Muster $\boldsymbol{x}_i$. Dann ist die Klassifikationsunsicherheit C für den Datenpunkt $\boldsymbol{x}_i$ definiert als:*

$$C(\boldsymbol{x}_i) = H(c(y_1), \ldots, c(y_m)) = -\sum_{c=1}^{m} c(y_c) \log_2(c(y_c)) \tag{6.6}$$

Dieser Wert der Entropie muss zusätzlich mit der aktuellen Anzahl der Klassen $H_{\max} = \log_2 |Y|$ in jeder Iteration normalisiert werden. Die Entropie wird in dieser Arbeit als Maß für die Unsicherheit des Klassifikators verwendet. Sie steht in direkter Beziehung zur Konfidenz des k-nächsten Nachbarn Klassifikators.

6.5. Prototypen Basierte Aktive Klassifikation

Basierend auf den Potentialen P, welche auf den unklassifizierten Daten U beruhen, und der Klassifikationsunsicherheit C, welche auf den klassifizierten Daten L aufbaut, wird ein neues Kriterium zur Selektion von Mustern gebildet.

Definition 4. *Die Unsicherheitsverteilung D wird definiert als:*

$$D(\boldsymbol{x}_i) = (1 - \epsilon) P(\boldsymbol{x}_i) + \epsilon \; C(\boldsymbol{x}_i) \tag{6.7}$$

Der Paramter $\epsilon \in [0, 1]$ kontrolliert den Einfluss der Klassifikationsunsicherheit auf die Unsicherheitsverteilung.

Basierend auf diesem Kriterium wird der folgende Algorithmus 6.1 zum Aktiven Lernen einer Klassifikation vorgestellt. Die Potentiale können offline vorberechnet werden, um den interaktiven Klassifikationsprozess zu beschleunigen. Anschließend wird für jedes Muster die Klassifikationsunsicherheit berechnet. In jeder Iteration wird das Muster mir der höchsten Unsicherheitsverteilung gewählt. Mit dem erhaltenen Klassenlabel entsteht damit ein neuer Prototyp zur Klassifikation mit dem gewichteten k-nächsten Nachbarn Algorithmus. Durch die Reduktion der Potentiale in jeder Iteration verringert sich ihr Einfluss auf die Unsicherheitsverteilung. Der Einfluss der Klassifikationsunsicherheit nimmt damit auf natürliche Weise zu. Durch die Reduktion der Potentiale kann auch eine Abbruchbedingung hergeleitet werden: Fällt die Summe der Potentiale unter einen vordefinierten Schwellwert t, kann der Lernprozess abgebrochen werden.

Algorithmus 6.1 Prototypen-basierte Aktive Klassifikation (PBAC)

Require: Schwellwert t
Potentialsumme $\leftarrow 0$
for all $\boldsymbol{x}_i \in U$ **do**
  Berechne das Potential $P(\boldsymbol{x}_i)$ gemäß Gleichung 6.1.
  Potentialsumme $\leftarrow$ Potentialsumme $+ P(\boldsymbol{x}_i)$
end for
while Potentialsumme $> t$ **do**
  for all $\boldsymbol{x}_i \in U$ **do**
    Berechne die Klassifikationsunsicherheit $C(\boldsymbol{x}_i)$ gemäß Gleichung 6.6.
    Berechne die Unsicherheitsverteilung $D(\boldsymbol{x}_i)$ gemäß Gleichung 6.7.
  end for
  Wähle das Muster $\boldsymbol{x}_t$ mit dem höchsten Wert der Unsicherheitsverteilung.
  Beziehe das Klassenlabel l für $\boldsymbol{x}_t$.
  $\boldsymbol{p}_t \leftarrow$ Erzeuge einen neuen Prototyp $\boldsymbol{x}_t$ mit Klasse l.
  Füge $\boldsymbol{p}_t$ zur Menge der aktuellen Prototypen P hinzu.
  Klassifiziere den Datensatz mit P.
  Reduziere das Potential um $N(\boldsymbol{x}_t)$ gemäß Gleichung 4.2.
end while

Die Performanz des PBAC Algorithmus ist von zwei Faktoren bzw. von zwei Parametern abhängig: Dies ist zum einen der Radius zur Berechnung der Potentiale r_a und zum anderen der Wert ϵ, welcher den Einfluss der Klassifikationsunsicherheit $C(\boldsymbol{x}_i)$ kontrolliert. Der Radius r_a legt fest, wie groß die Nachbarschaft eines Musters ist, in der die anderen Muster zum Potential beitragen. Ein großer Radius führt zu einer genaueren Schätzung, welche wiederum mehr Rechenzeit benötigt. Da die Reduktion der Potentiale auch an den Radius gekoppelt ist, kann ein zu großer Radius zu weite Auswirkungen auf die restlichen Potentiale haben. Somit besteht die Gefahr, dass interessante Regionen mit hohem Explorationspotential übergangen werden. Ein zu schmaler Radius führt wiederum zu einer ungenauen Schätzung. Zusätzlich kann das Problem entstehen, dass zu viele Muster ein gleich hohes Explorationspotential erhalten. Dies führt zu „unnötigen“ Abfragen in einer Region.

Die Reduktion der Potentiale lässt den Einfluss der Klassifikationsunsicherheit über die Iterationen hinweg wachsen. Durch Veränderung des Wertes ϵ kann der Zeitpunkt des Übergangs nach vorne beeinflusst werden. Ein kleiner Wert kann - je nach zugrunde liegenden Daten - die Klassifikationsgenauigkeit schneller erhöhen. Ein zu großes ϵ kann jedoch eine ausreichende Exploration der Daten auch verhindern.

Abbildung 6.3 vermittelt einen Eindruck über die Arbeitsweise des PBAC Algorith-

mus. In diesem eindimensionalen Beispiel (die Muster sind auf der x-Achse aufgetragen) existieren drei verschiedene Klassen A, B und C. In Abbildung 6.3a sieht man aufgrund der Potentiale, dass es drei Verteilungen gibt, die exploriert werden müssen. Der Faktor ϵ hat in diesem Beispiel den Wert $0,5$, so dass der Maximalwert für die Unsicherheitsverteilung ebenfalls bei $0,5$ liegt. Im ersten Schritt wird ein Prototyp der Klasse B erzeugt und die Potentiale um dieses Muster reduziert (Abbildung 6.3b). Im zweiten Schritt wird - aufgrund des hohen Potentials - der Prototyp mit Klasse A erzeugt. Da nun zwei Prototypen unterschiedlicher Klasse existieren, hängt die Unsicherheitsverteilung nicht mehr alleine vom Potential, sondern auch von der Klassifikator-Unsicherheit ab, siehe Abbildung 6.3c. Auch der dritte Prototyp der Klasse C wird aufgrund seines hohen Potentials gewählt, allerdings wird dies auch durch die hohe Klassifikationsunsicherheit in diesem Bereich verstärkt (Abbildung 6.3d). Im vierten Schritt (Abbildung 6.3e) wird das Muster zwischen Klasse A und Klasse B aufgrund der Klassifikationsunsicherheit und des verbleibenden Potentials gewählt. In diesem Beispiel lässt sich gut erkennen, wie der Datensatz zunächst exploriert wird und alle drei Klassen gefunden werden. Anschließend konzentriert sich der Algorithmus auf die Klassengrenzen.

6.6. Fazit

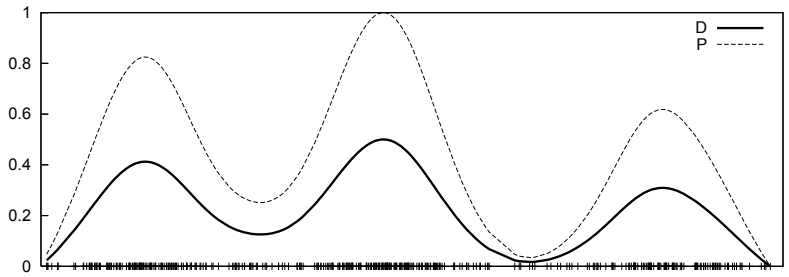
In diesem Kapitel wurde die Methode der Prototypen basierten aktiven Klassifikation (PBAC) vorgestellt. Iterativ werden Prototypen für eine k-nächste Nachbarn Klassifikation erzeugt. Zur Auswahl eines Musters wurde ein neues Kriterium definiert, welches den Repräsentationsgrad des Musters mit der Unsicherheit des Klassifikators für dieses Muster kombiniert. In Regionen, die bereits exploriert worden sind, werden die Potentiale verringert so dass ein fließender Übergang zwischen Exploration und Verfeinerung der Klassengrenzen stattfindet.

Der PBAC-Algorithmus selektiert zu Beginn repräsentative Muster. Das Gesamtpotential, welches in jeder Iteration verringert wird, bietet einen Anhaltspunkt für den Benutzer, zu welchem Zeitpunkt der Datensatz hinreichend exploriert wurde.

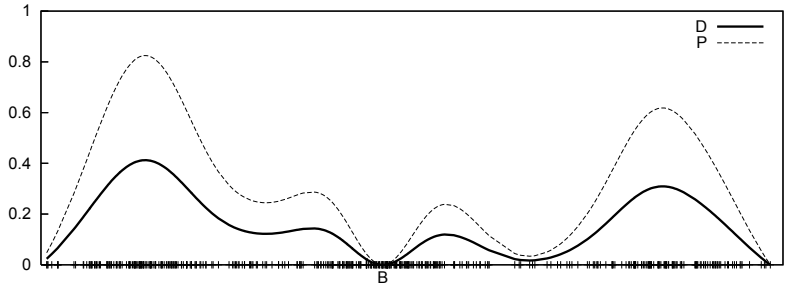
Bisherige Ansätze im Aktiven Lernen ignorieren überwiegend die Verteilung der Daten bei der Initialisierung des Modells ((MacKay, 1992), (Schohn und Cohn, 2000), (Roy und McCallum, 2001), (Cohn u. a., 1994a), (Tong und Koller, 2001)). Bei der Arbeit von (Nguyen und Smeulders, 2004) geht die Dichteschätzung konstant in die Auswahl von Mustern mit ein. Dadurch findet kein richtiger Übergang von repräsentativen Mustern

zu Mustern an den Klassengrenzen statt.

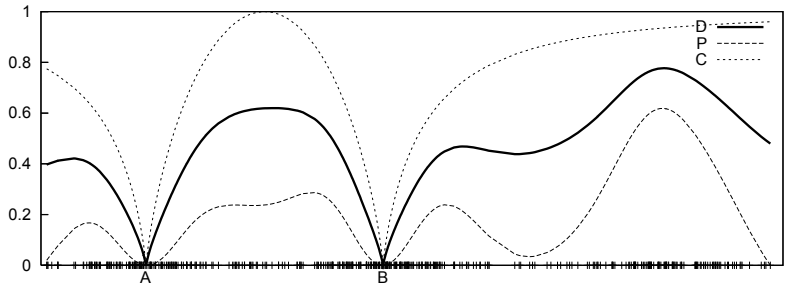
Im folgenden Kapitel vergleichen wir das Verfahren der Aktiven Lernenden Vektor Quantisierung und der Prototypen basierten aktiven Klassifikation mit anderen populären Verfahren des Aktiven Lernens.



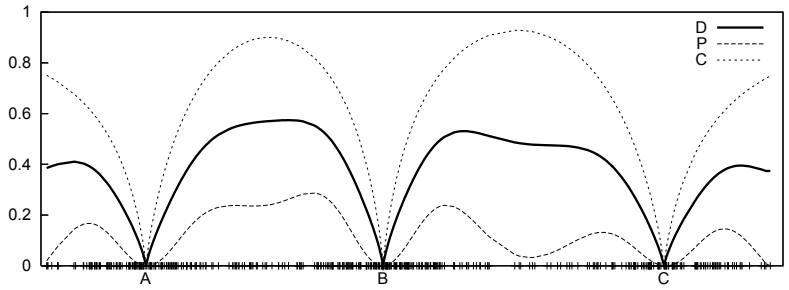
(a) Schritt 0



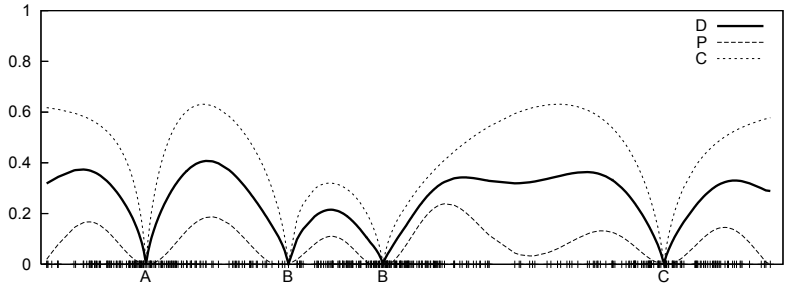
(b) Schritt 1



(c) Schritt 2



(d) Schritt 3



(e) Schritt 4

Abbildung 6.3.: Kennlinien von Potential P, Klassifikator-Unsicherheit C und der daraus resultierenden Unsicherheitsverteilung D in aufeinander folgenden Schritten.

7. Ergebnisse

In diesem Kapitel sollen das in Kapitel 5 entwickelte ALVQ Verfahren und das in Kapitel 6 entwickelte PBAC Verfahren bezüglich der Klassifikationsgüte nach Anzahl gezeigter und klassifizierter Muster untersucht werden. Durch die Einbeziehung des Repräsentationsgrades für jedes Muster wird erwartet, dass diese Verfahren eine bessere Performanz bei der Initialisierung und in den ersten Lerniterationen haben. Weiterhin sollte sich der Wechsel der Selektionsstrategie in späteren Lerniterationen positiv auswirken.

Zunächst wird das Verhalten der Verfahren auf einem künstlich generierten Datensatz analysiert. Auf ausgewählten Datensätzen aus dem UCI Repository of Machine Learning (Asuncion und Newman, 2007), die häufig im Bereich des maschinellen Lernens verwendet werden, wird die Performanz der Verfahren untersucht. Zuletzt wird ein Anwendungsbeispiel mit Bilddaten - wie es auch in der praktischen Anwendung der Zellbild-Klassifikation in Kapitel 8 vorkommt - betrachtet.

Um die neu entwickelten Verfahren sinnvoll in die Vielzahl der unterschiedlichen existierenden aktiven Lernalgorithmen einzuordnen, wurde aus jeder der in Kapitel 3 vorgestellten Kategorien ein populäres (bzw. viel zitiertes) Verfahren implementiert. Aus dem Bereich „Optimierung einer Zielfunktion“ wurde das als *Self-conf* bezeichnete Verfahren von (Roy und McCallum, 2001) implementiert. Aus den Bereichen „Reduktion des Versionsraumes“ bzw. „Uncertainty sampling“ wurde eine Support Vektor Machine mit einer aktiven Selektionsstrategie implementiert. Dieses Verfahren basiert auf den Arbeiten von (Tong und Koller, 2001) und (Schohn und Cohn, 2000) und wird als *aktive SVM* bezeichnet.

Aus dem Bereich „Aktives Lernen und Clustering“ wurde ein Verfahren analog zur Arbeit von (Nguyen und Smeulders, 2004) implementiert, welches als *DWUS* (Kurzform für Density-Weighted Uncertainty Sampling) bezeichnet wird. Da das Verfahren nur für 2-Klassen Probleme konzipiert wurde, ist die Performanz bei Mehrklassen-Problemen deutlich schlechter als bei anderen Verfahren. Dies liegt daran, dass der Algorithmus für

jede einzelne Klasse traininiert werden muss. Bei 10 Klassen und 5 initialen Clustern ergeben sich beispielsweise 50 initiale Trainingsbeispiele. Auch in den nachfolgenden Iterationen muss jeweils ein Muster pro Klasse ausgewählt werden. Aus Gründen der Übersichtlichkeit werden deshalb die Lernkurven des DWUS-Verfahrens nur für 2-Klassen Datensätze aufgeführt.

Die einzelnen Verfahren wurden folgendermaßen initialisiert:

PBAC Keine besondere Initialisierung notwendig.

ALVQ Anzahl der Cluster = Anzahl der Klassen * 2.

DWUS Anzahl der Cluster = Anzahl der Klassen * 2.

Aktive SVM Wie in der Arbeit von (Schohn und Cohn, 2000) beschrieben, werden so lange zufällig Muster gezogen, bis von jeder Klasse 4 Muster zur Verfügung stehen. Erst dann wird die query-Strategie angewendet.

Self-conf Dieser Algorithmus benötigt einen initialisierten Naive Bayes Klassifikator. Daher werden bei diesem Verfahren gezielt Muster aus jeder Klasse gezogen und für das Training eines initialen Modells verwendet.

Bei der Initialisierung der anderen aktiven Lernverfahren wird eine paradoxe Situation deutlich, die in den meisten Arbeiten nicht angesprochen wird: Auf der einen Seite soll durch gezielte Auswahl von Mustern der Aufwand für eine Klassifikation möglichst gering gehalten werden, auf der anderen Seite wird davon ausgegangen, dass eine genügend große Anzahl an initialen Mustern vorhanden ist. Ohne die Information, welche Klassen in dem Datensatz enthalten sind, lassen sich die meisten Verfahren überhaupt nicht initialisieren. Insofern ist ein direkter Vergleich der Verfahren kritisch zu betrachten, da bei der aktiven SVM und dem Self-conf-Algorithmus mehr a-priori Informationen in die Klassifikation mit einfließen.

Die Performanz der aktiven SVM hängt erheblich vom verwendeten Kernel ab. Das Auffinden eines optimalen Kernels ist nach wie vor Gegenstand der Forschung. Bei den vorliegenden Versuchen wurde daher empirisch vorgegangen: Aus drei weit verbreiteten Kernelfunktionen (Polynomiell, RBF und Hypertangential) mit sinnvollen Parameterbelegungen wurde der Kernel gewählt, der die beste Performanz bezüglich der Testdaten aufwies. Anschließend wurden mit diesem Kernel sinnvolle Parameterbelegungen getestet.

7.1. Künstliche Daten

Um ein besseres Verständnis der eingesetzten Verfahren zu bekommen, soll in diesem Abschnitt das Verhalten auf künstlichen Daten analysiert werden. Es handelt sich hierbei um einen 2-dimensionalen Datensatz, bestehend aus 10435 Mustern. In diesem Datensatz existieren zwei verschiedene farblich markierte Klassen, siehe Abbildung 7.1a.

Abbildung 7.1 zeigt die von den jeweiligen Verfahren selektierten Muster in gelber Markierung und die daraus resultierende Klassifikation.

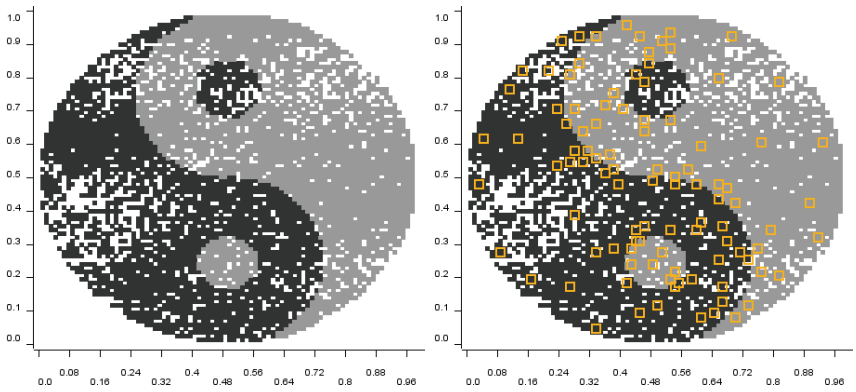
Beim PBAC-Verfahren (Abbildung 7.1b) lässt sich erkennen, dass in dichten Datenregionen wenige repräsentative Muster ausgewählt wurden. Der Datenraum wird großflächig abgedeckt und exploriert. Anschließend fokussiert sich der Algorithmus speziell auf die Klassengrenzen. Dadurch werden redundante Abfragen - speziell in weniger dichten Datenregionen - vermieden und der Fokus auf die interessanten Bereiche im Datensatz gelegt.

Der LVQ-Algorithmus konzentriert sich nach der Initialisierung mehr auf die Grenzen, die durch die Cluster induziert werden, siehe Abbildung 7.1c. Der Explorationsanteil ist durch die Bildung von größeren Clustern etwas geringer und führt auch zu einer stärkeren Konzentration auf die größte s-förmige Klassengrenze.

Bei der Selektion der Muster mit der aktiven SVM (Abbildung 7.1d) ist die Strategie des uncertainty sampling deutlich zu sehen. Nach einer zufälligen Ziehung von Mustern, unabhängig vom Repräsentationsgrad, werden Muster an der Grenze zur Hyperebene im Parameterraum gewählt, der durch den gewählten Kernel (in diesem Fall ein RBF-Kernel) induziert wird. Nur durch den RBF-Kernel ist in diesem Datensatz eine korrekte Klassifikation möglich. Die große Abhängigkeit vom gewählten Kernel einer SVM zeigt auch Abbildung 7.2. Hier wurde die SVM mit einem linearen Kernel trainiert, der nicht in der Lage ist, das Zielkonzept zu lernen.

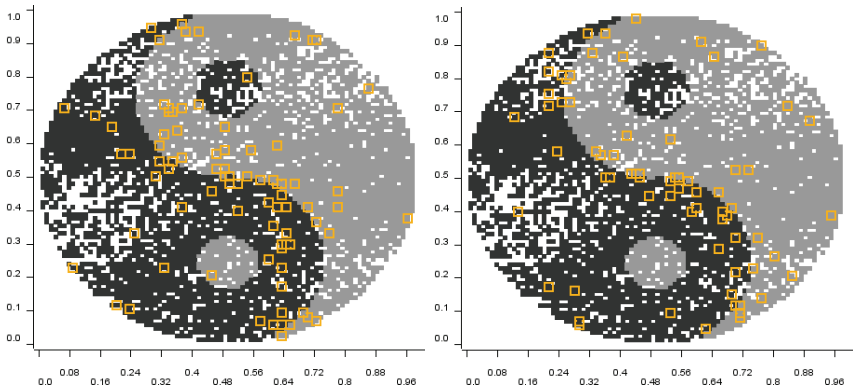
In Abbildung 7.1e sind die selektierten Muster des Self-conf Algorithmus zu sehen. Viele Muster werden zunächst zufällig zur Initialisierung des Modells gezogen. Zwei „Anhäufungen“ von selektierten Mustern sind deutlich zu erkennen, welche die Klassengrenze des Naiven Bayes Klassifikators beeinflussen (siehe auch Abbildung 7.3e mit Klassifikation).

Beim DWUS-Verfahren (Abbildung 7.1f) ist deutlich die Konzentration der selektierten Muster auf die durch die Cluster induzierten Klassengrenzen zu sehen. Hier sind an den Klassengrenzen Häufungen von selektierten Mustern zu beobachten. Wenige Muster



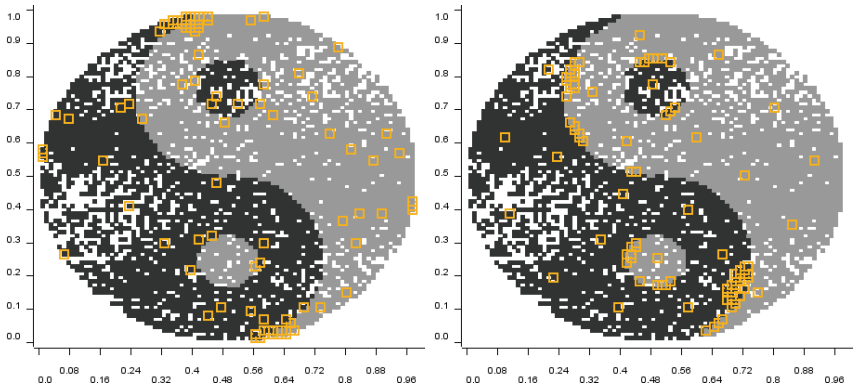
(a) “Wahre“ Klassifikation

(b) PBAC



(c) LVQ

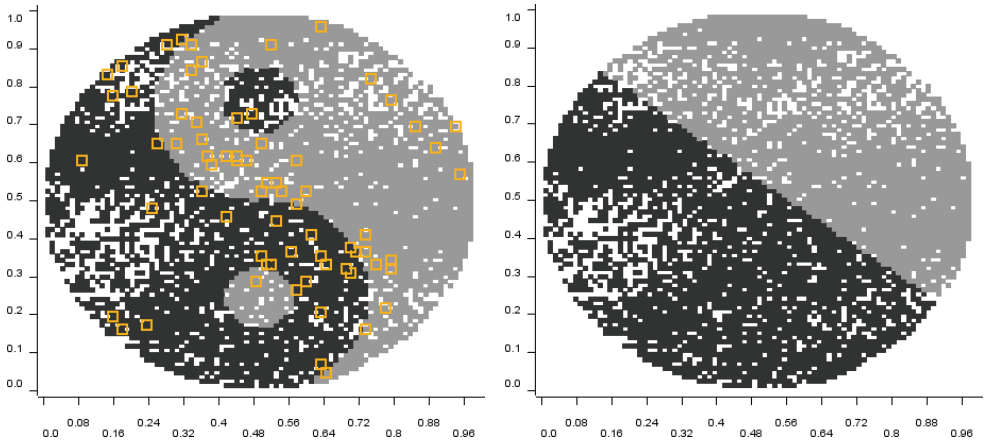
(d) SVM



(e) Naive Bayes

(f) DWUS

Abbildung 7.1.: Selektierte Daten (gelb markiert) der verschiedenen Algorithmen.



(a) Selektierte Muster (b) Klassifikation

Abbildung 7.2.: Selektierte Muster und Klassifikation mit linearer SVM.

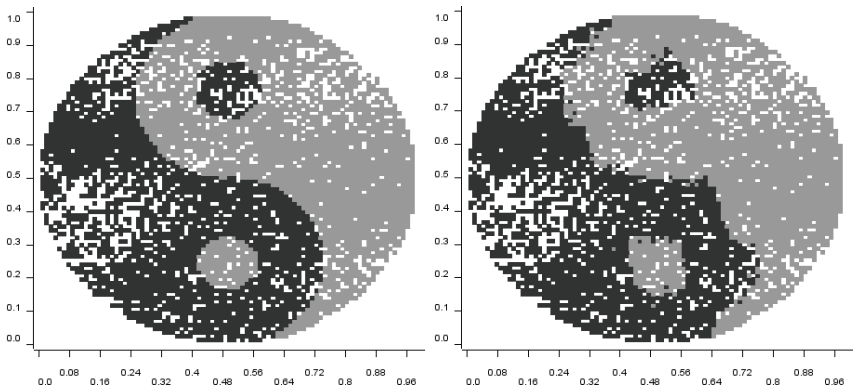
in den Cluster-Zentren decken den Datenraum großflächig ab.

Aus den verschiedenen Abbildungen wird deutlich, dass sich die Selektionsstrategien zum Teil deutlich voneinander unterscheiden. Beim ALVQ, PBAC und DWUS-Verfahren werden die ersten Muster nicht zufällig gezogen, sondern nach ihrem Repräsentationsgrad ausgewählt. Dies führt zu einer gleichmäßigen Abdeckung des Datenraumes und erlaubt somit eine bessere Exploration. Somit ist in diesem Beispiel sichergestellt, dass alle relevanten Bereiche des Datensatzes betrachtet werden. Das zufällige Ziehen von Mustern bei der aktiven SVM und dem Self-Conf Algorithmus führt dazu, dass relevante Stellen im Datensatz ignoriert werden, was sich negativ auf das Klassifikationsergebnis auswirkt.

Abbildung 7.3 zeigt die resultierende Klassifikation nach 100 gezeigten Mustern. Das PBAC-Verfahren (Abbildung 7.3b) findet alle Klassen in der Zielabbildung. Die runden, geschwungenen Formen können aufgrund der Prototypen basierten Klassifikation nur grob abgebildet werden. Insgesamt wird die Abbildung jedoch vollständig wiedergegeben.

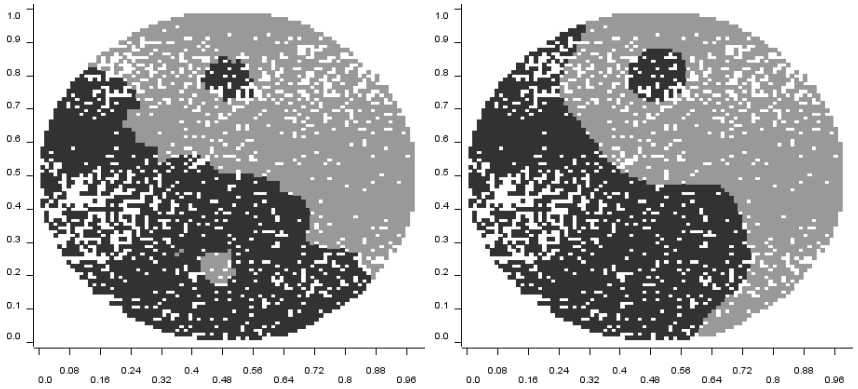
Dasselbe gilt für die Klassifikation durch das ALVQ-Verfahren (Abbildung 7.3c) Hier können die schmalen Bereiche links oben und rechts unten durch die Verschiebung der Prototypen nicht korrekt wiedergegeben werden.

Durch die zufällige Initialisierung der SVM wird der kleine graue Bereich im unteren Teil des Bildes nicht gefunden (siehe Abbildung 7.3d). (Nur) durch den RBF-Kernel können die geschwungenen Formen der Abbildung gut wiedergegeben werden.



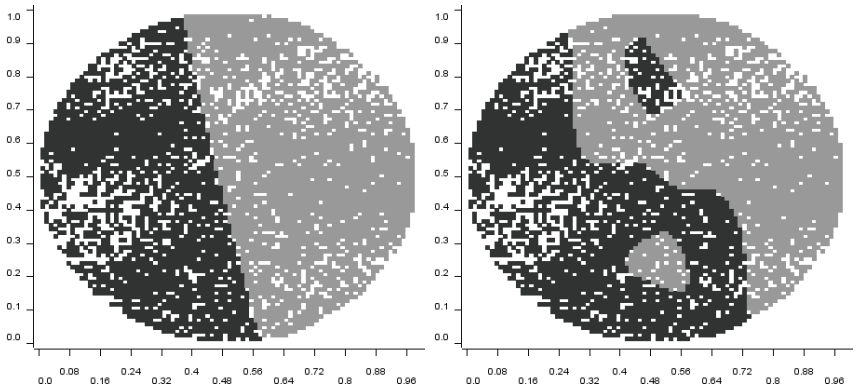
(a) “Wahre“ Klassifikation

(b) PBAC



(c) LVQ

(d) SVM



(e) Naive Bayes

(f) DWUS

Abbildung 7.3.: Resultierende Klassifikation.

Datensatz	PBAC	ALVQ	ASVM	ANB	Durchschnitt
segment	**86,89** %	84,04 %	82,97 %	71,21 %	81,28 %
satimage	**82,48** %	78,13 %	79,15 %	79,93 %	80,83 %
diabetes	73,60 %	73,90 %	73,28 %	**75,74** %	74,13 %
spam	78,77 %	72,13 %	66,42 %	**80,82** %	74,54 %
vehicle	60,30 %	54,56 %	**73,16** %	42,29 %	58,68 %
pendigits	**89,76** %	67,39 %	79,38 %	64,80 %	75,16 %
faces	**93,36** %	75,88 %	87,14 %	80,54 %	83,26 %
Abweichung Durchschnitt	+5,45 %	-3,00 %	+2,07 %	-4,53 %	-

Tabelle 7.1.: Ergebnisse der aktiven Lernverfahren nach 100 klassifizierten Mustern.

Auch der Naive Bayes Klassifikator hat Bereiche der Abbildung aufgrund der zufälligen Initialisierung nicht finden können. Dadurch resultiert eine zu grobe Abbildung des zu lernenden Zielkonzeptes (Abbildung 7.3e).

Durch das initiale Clustering mit einer ausreichend großen Anzahl von Clustern wurden im DWUS-Verfahren alle relevanten Bereiche der Abbildung aufgedeckt (Abbildung 7.3f) . Die Abbildung wird auch hier grob wiedergegeben.

7.2. UCI Repository Daten

Das UCI Repository of Machine Learning (Asuncion und Newman, 2007) besteht aus ca. 160 verschiedenen Datensätzen, welche für die empirische Analyse von Algorithmen des maschinellen Lernens verwendet werden. Aus diesen Datensätzen wurden sechs verschiedene Datensätze ausgewählt, welche den folgenden Bedingungen entsprechen:

1. Alle Attribute sind numerisch.
2. Das Klassenattribut ist nominal.
3. Die Anzahl der Muster ist größer als 500.

Alle Datensätze wurden in einem Verhältnis von 30 % zu 70 % in Trainings- und Testdaten aufgeteilt.

Die Ergebnisse nach jeweils 100 klassifizierten Mustern für alle Datensätze sind in Tabelle 7.1 zusammengefasst.

Die Parameterbelegungen der einzelnen Verfahren sind in Tabelle 7.2 zusammengefasst. Für das PBAC-Verfahren sind die Parameter bei allen Datensätzen gleich. Der verwendete Radius für die Exploration und Potentialreduzierung r_a ist 0,4, der Parame-

Datensatz	PBAC	ALVQ	ASVM	Self-conf
segment	$r_a = 0.4$, $\epsilon = 0.4$, $k = 5$	RT = 1	RBF, $\sigma = 1.0$	$\|L\| = 7$
satimage	$r_a = 0.4$, $\epsilon = 0.4$, $k = 5$	RT = 2	RBF, $\sigma = 1.0$	$\|L\| = 12$
diabetes	$r_a = 0.4$, $\epsilon = 0.4$, $k = 5$	RT = 1	Polyn., Bias 1, Potenz 2	$\|L\| = 2$
spam	$r_a = 0.4$, $\epsilon = 0.4$, $k = 5$	RT = 2	Polyn., Bias 1, Potenz 2	$\|L\| = 2$
vehicle	$r_a = 0.4$, $\epsilon = 0.4$, $k = 5$	RT = 3	Polyn., Bias 1, Potenz 2	$\|L\| = 12$
pendigits	$r_a = 0.4$, $\epsilon = 0.4$, $k = 5$	RT = 2	RBF, $\sigma = 1.0$	$\|L\| = 20$
faces	$r_a = 0.4$, $\epsilon = 0.4$, $k = 5$	RT = 1	Polyn., Bias 1, Potenz 2	$\|L\| = 20$

Tabelle 7.2.: Parameterbelegungen der aktiven Lernverfahren.

ter ϵ für die Spezialisierung beträgt 0,4 und die Anzahl der Nachbarn k für die Klassifikation ist 5 (siehe Abschnitt 6.3). Auf dem ersten Datensatz (den Segment Daten) wird in Abschnitt 7.2.1 die Wahl der Parameterwerte für r_a und ϵ erörtert.

Für das ALVQ-Verfahren ist jeweils die verwendete Rekursionstiefe (RT) für die Phase der Exploration zu Beginn angegeben. Die Parameter für das Training im LVQ-Algorithmus beim ALVQ-Verfahren sind bei allen Versuchen gleich: Die initiale Lernrate beträgt 0,1, die Abklingkonstante beträgt 0,92. Für die aktive SVM ist der verwendete Kernel mit den entsprechenden Parameterwerten angegeben und für den Self-Conf Algorithmus die Anzahl der initial klassifizierten Trainingsdaten $|L|$.

Zu jedem der aktiven Lernverfahren existiert auch ein entsprechendes passives Lernverfahren. Die Performanz der passiven Lernverfahren, die auf *allen* Mustern gelernt wurden, ist in Tabelle 7.3 aufgeführt. Für das PBAC-Verfahren ist die k-nächste Nachbarn Klassifikation (KNN) das korrespondierende passive Lernverfahren, beim ALVQ-Verfahren ist es der normale LVQ-Algorithmus. Für die aktive SVM ist eine normale SVM mit den gleichen Kernel-Parametern und für den Self-Conf Algorithmus ein Naiver Bayes Lerner jeweils das passive Lernverfahren. Bei der Klassifikationsgüte der aktiven Lernverfahren ist daher auch immer die Performanz des korrespondierenden passiven Lernverfahrens zu beachten, da es hier große Unterschiede geben kann.

In den folgenden Abschnitten werden die unterschiedlichen Verfahren auf verschiedenen Datensätzen direkt miteinander verglichen. Um die Performanz der aktiven Lernverfahren im Einzelnen zu beurteilen, sind im Anhang für jeden Datensatz und jedes Verfahren weitere Ergebnisse bezüglich der Performanz zu finden. Betrachtet werden hierbei die Performanz mit gezogenen Mustern gemäß der aktiven Lernstrategie, die Performanz mit zufällig gezogenen Mustern und als Obergrenze die Performanz eines

Datensatz	KNN	LVQ	SVM	NB
segment	93,45 %	86,89 %	89,98 %	81,39 %.
satimage	88,36 %	77,48 %	87,51 %	79,67 %
diabetes	71,93 %	74,90 %	75,65 %	73,42 %
spam	74,4 %	74,28 %	76,01 %	77,13 %
vehicle	65,2 %	52,68 %	76,86 %	44,43 %
pendigits	97,77 %	69,69 %	98,14 %	81,96 %
faces	94,97 %	85,12 %	97,48 %	91,76 %

Tabelle 7.3.: Ergebnisse der korrespondierenden passiven Lernverfahren auf allen Mustern.

korrespondierenden passiven Lernverfahrens mit allen Mustern. Dies entspricht der Idee des Performanz-Kriteriums aus der in Abschnitt 3.4.5 vorgestellten Arbeit von (Baram u. a., 2004). Um die Güte eines aktiven Lernalgorithmus zu beurteilen, kann man die Fläche zwischen der passiven und der aktiven Lernstrategie betrachten.

7.2.1. Segment Daten

Die Segment Daten aus dem UCI Repository (Asuncion und Newman, 2007) bestehen aus Aufnahmen von verschiedenen Bildklassen (Ziegel, Himmel, Laub, Zement, Fenster, Weg, Gras). Jedes Bild wurde von Hand segmentiert und 19 numerische Merkmale der Segmente berechnet, z. B. die Anzahl der Pixel und durchschnittliche RGB-Werte.

Bevor die verschiedenen aktiven Lernverfahren miteinander verglichen werden, soll die Wahl der Parameterwerte für den PBAC Algorithmus erörtert werden. In Abschnitt 6.5 haben wir bereits den Einfluß der Parameter auf die Performanz des Algorithmus betrachtet. Die praktische Auswirkung der Parameterwahl wird auf den Segment Daten untersucht.

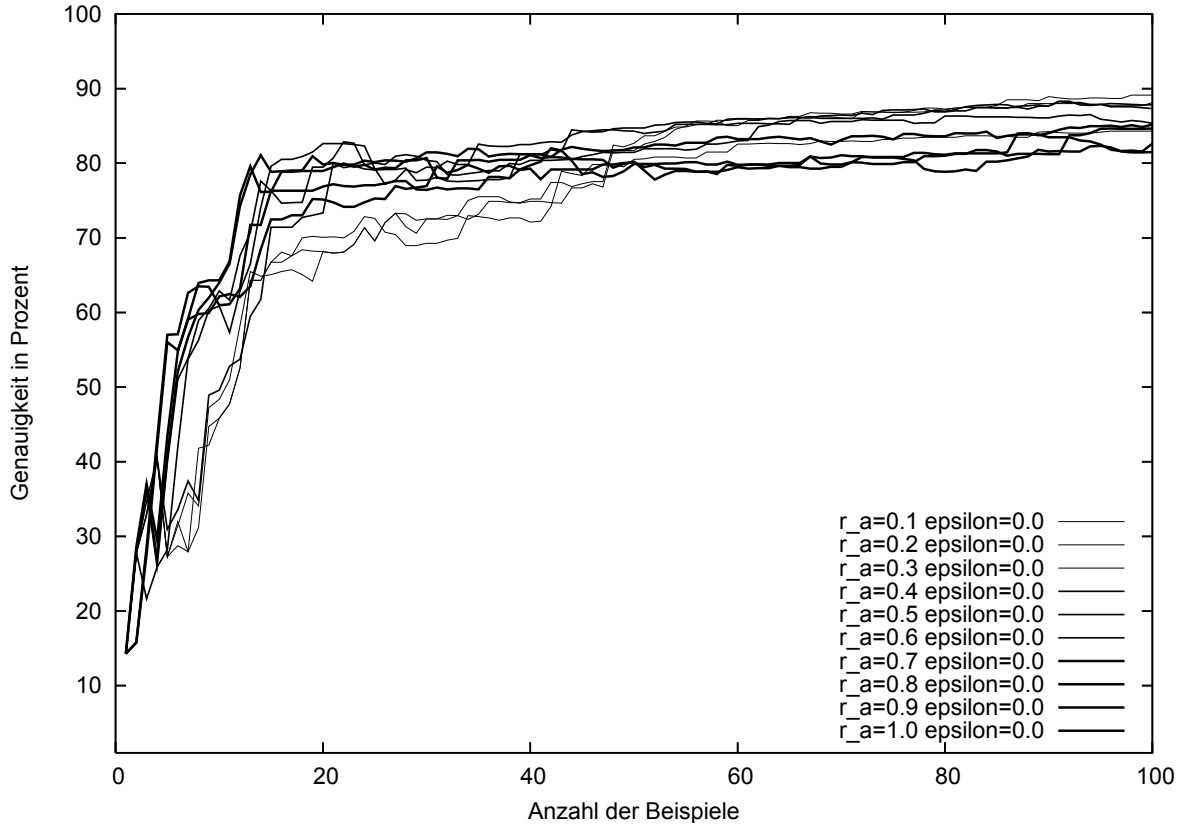
Im Folgenden betrachten wir in allen Abbildungen die Anzahl der gezeigten Muster (auf der x-Achse) gegenüber der Klassifikationsgenauigkeit (y-Achse).

In Abbildung 7.4a sind die Kennlinien für verschiedene Werte von r_a dargestellt. Hierbei ist deutlich zu erkennen, dass mit einem großen Radius das Modell besser initialisiert. Mit einem kleinen Radius werden mehr Regionen mit hohem Potential gefunden, dementsprechend wird der Datensatz mehr exploriert. Dies hat zur Folge, dass mehr Muster benötigt werden. Mit zunehmender Anzahl von Mustern ist dies aber von Vorteil, die Performanz mit kleinem Radius liegt in späteren Iterationen über der Performanz der Modelle mit großem Radius.

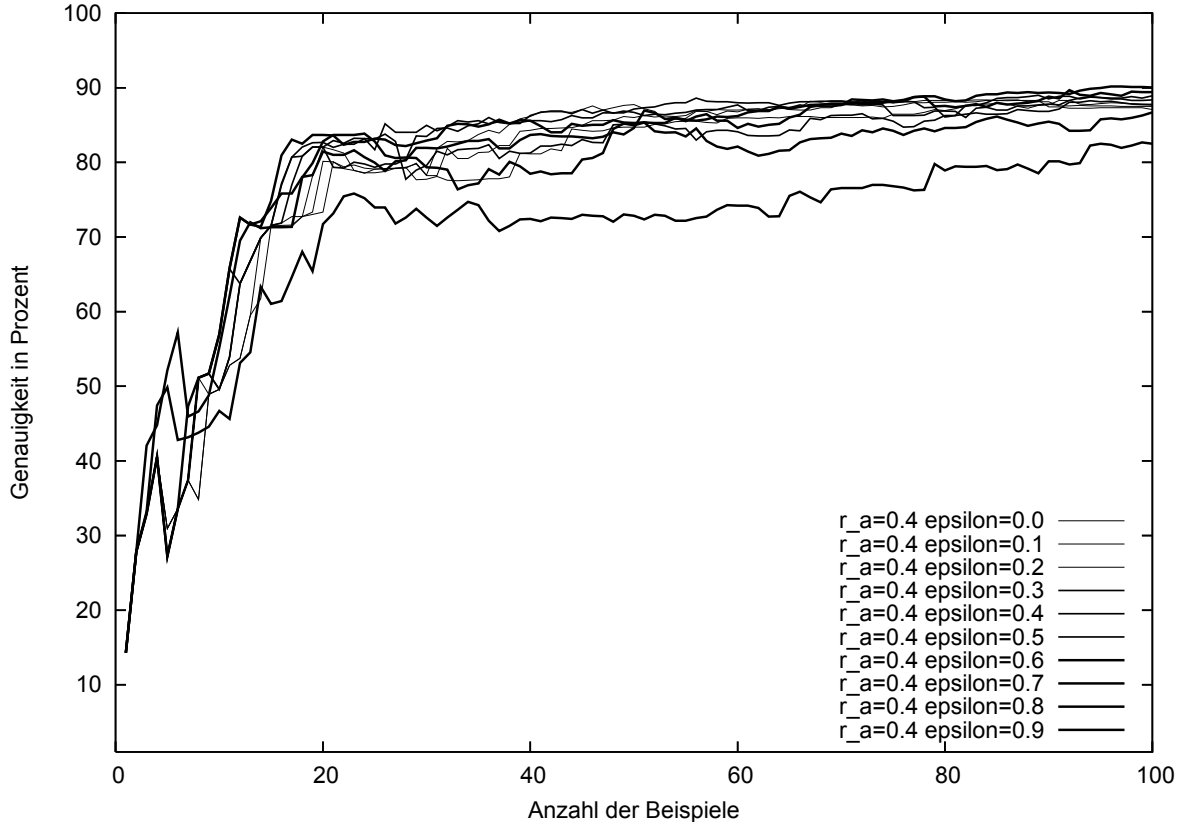
In Abbildung 7.4b wurde r_a aufgrund der vorherigen Betrachtung auf einen „moderaten“ Wert von 0,4 fixiert. Für verschiedene Werte von ϵ soll der Einfluß des Parameters untersucht werden. Hier zeigt sich, dass ein hoher Wert für ϵ zu starken Einbußen in der Performanz führt. Dies liegt an der zu hohen Gewichtung der Klassifikationsunsicherheit, bevor der Datensatz ausreichend exploriert wurde. Ein moderater Wert im Bereich von 0,4 zeigt gegenüber einem geringen Wert von 0,1 eine leichte Verbesserung der Performanz. Es kann also durchaus von Vorteil sein, den Aspekt der Verfeinerung der Klassengrenzen in Betracht zu ziehen, bevor der Datensatz komplett durch die Reduktion der Potentiale exploriert wurde.

Das Zusammenspiel der beiden Parameter r_a und ϵ ist weitaus komplexer. Abbildung 7.5 zeigt einen Überblick für ausgesuchte Kombinationen von Werten (niedrig, mittel, hoch). Hier fallen besonders die Kennlinien mit großem ϵ auf, die von der durchschnittlichen Performanz abweichen. Ansonsten scheint der Einfluß der Parameter auf die Performanz keinen sehr großen Einfluß zu haben. Aus diesem Grund wird in dieser

Arbeit für r_a der Wert 0,4 verwendet, um eine ausreichende Exploration der Daten zu gewährleisten. Um die Performanz zu verbessern, wird der Wert für ϵ auf 0,4 gesetzt. Dies garantiert natürlich nicht die beste Performanz auf allen Datensätzen, da der Bedarf an Exploration je nach Datensatz unterschiedlich ausfällt. Da die Unterschiede in der Performanz jedoch nicht sehr groß ausfallen ist die vorgeschlagene Festlegung der Parameter relativ unproblematisch.



(a) Segment Daten: Einfluß des Parameters r_a



(b) Segment Daten: Einfluß des Parameters ϵ

Abbildung 7.4.: Segment Daten: Einzelner Einfluß der Parameter r_a und ϵ.

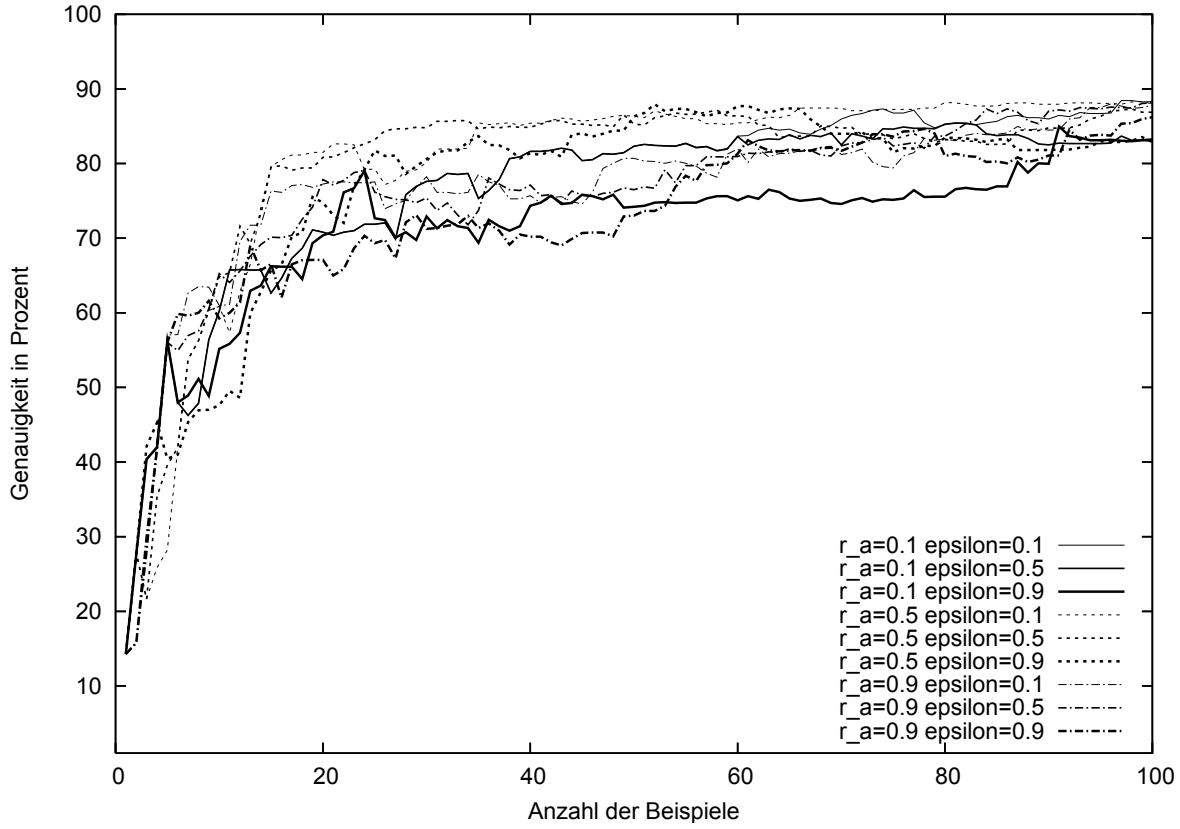


Abbildung 7.5.: Segment Daten: Einfluß der Parameter r_a und ϵ.

Ein weiterer Aspekt bei den Experimenten ist die variierende Performanz der aktiven SVM durch die zufällige Auswahl von Mustern zu Beginn des Lernens. Abbildung 7.6 soll die Stabilität des PBAC-Verfahrens gegenüber der aktiven SVM aufzeigen. In fünf unterschiedlichen Testläufen kann die Performanz der aktiven SVM stark schwanken oder zuweilen auch massiv einbrechen. In den folgenden Ergebnisgrafiken ist daher zur

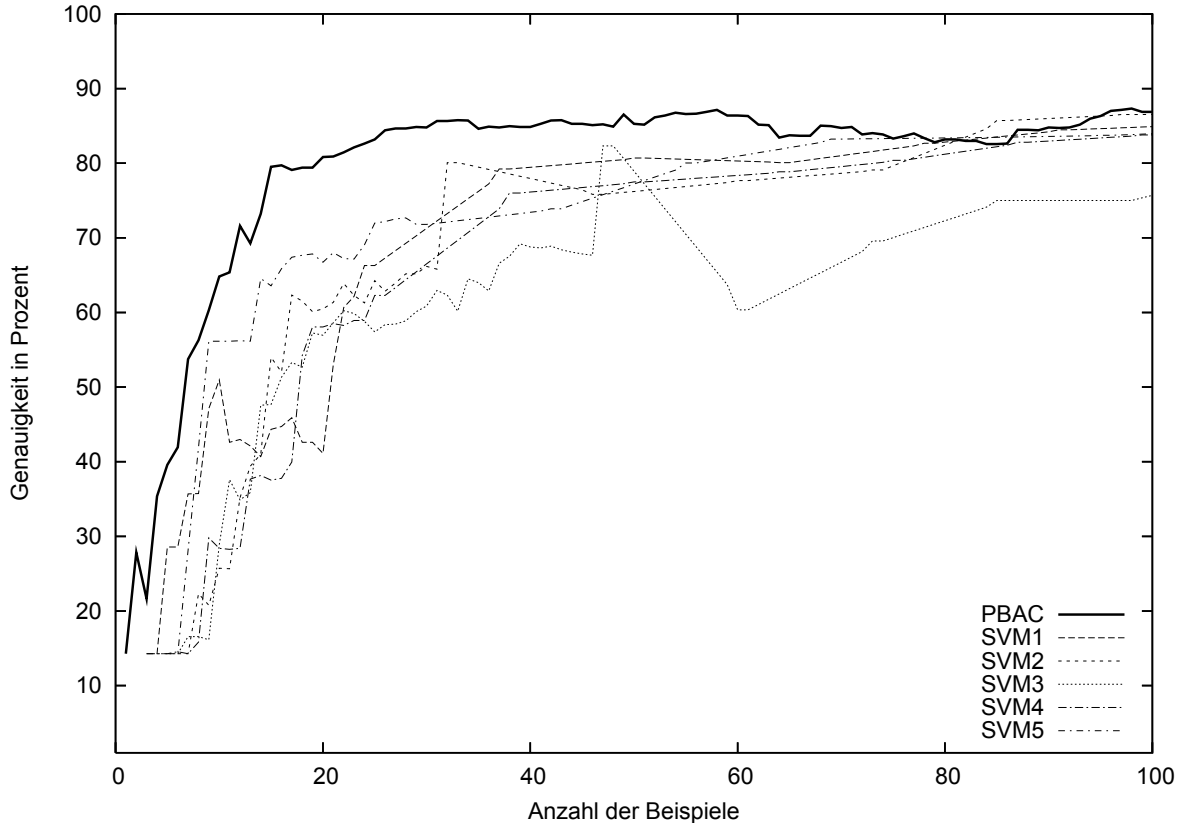


Abbildung 7.6.: Segment Daten: Stabilität.

Übersichtlichkeit die durchschnittliche Performanz der aktiven SVM dargestellt, mit den jeweiligen positiven und negativen Abweichungen.

Abbildung 7.7 zeigt die Kennlinien der verschiedenen Verfahren. Hierbei wird deutlich, dass sowohl das ALVQ Verfahren als auch das PBAC Verfahren deutlich schneller an Genauigkeit gewinnen und im Vergleich zur aktiven SVM deutlich stabiler sind. Die aktive SVM hat zu Beginn eine große Varianz in der Genauigkeit, verschuldet durch die zufällige Initialisierung mit zufällig gewählten Mustern. Aber auch zwischen dem 60. bis 80. gezeigten Muster ist die Varianz relativ hoch, bevor sie sich mit zunehmender Anzahl von Mustern stabilisiert. Wie im Anhang zu sehen ist, hat die aktive Selektionsstrategie

keinen Einfluss auf die Performanz der SVM. Der Self-conf Algorithmus ist im Bereich der ersten 30 Muster besser als die aktive SVM, danach steigert er sich allerdings kaum noch und schließt nach 100 Mustern deutlich schlechter ab als die anderen Verfahren (siehe Tabelle 7.1). Gemessen an der Grundperformanz des Naiven Bayes Klassifikators und der Performanz mit zufällig gezogenen Mustern (siehe Anhang) schneidet die aktive Selektionsstrategie im Self-conf Algorithmus bei den ersten 40 Mustern recht gut ab.

7.2.2. Satimage Daten

Die Satimage Daten aus dem UCI Repository (Asuncion und Newman, 2007) bestehen aus multispektralen Werten (aus dem roten und grünen Bereich und im Infrarot-Bereich) von einer 3×3 Nachbarschaft von Pixeln in Satellitenaufnahmen. In diesem Datensatz existieren sechs verschiedene Klassen (rotes Erdreich, Baumwollanbau, grauer Boden, feuchter grauer Boden, Boden mit wenig Vegetation und sehr feuchte graue Erde).

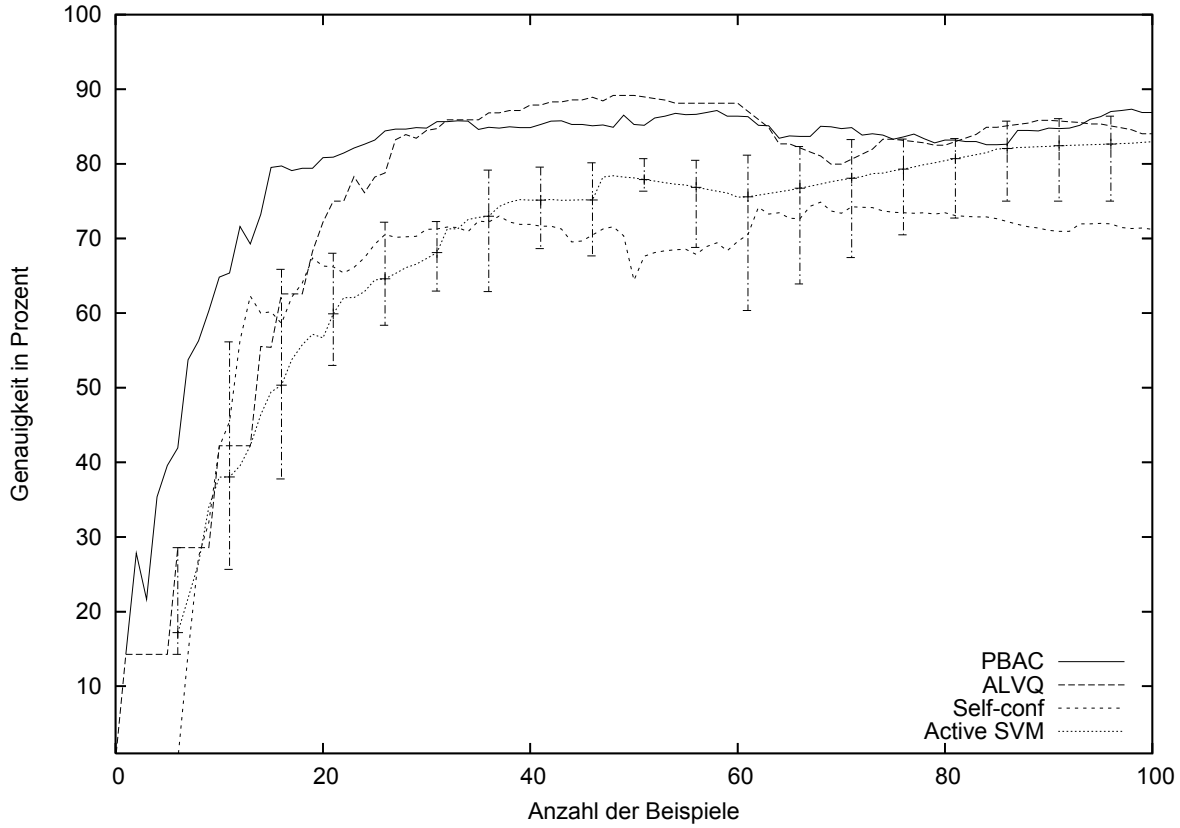


Abbildung 7.7.: Segment Daten.

Auch bei diesem Datensatz sind das PBAC und das ALVQ-Verfahren deutlich schneller und stabiler als der Self-conf Algorithmus und die aktive SVM, siehe Abbildung 7.8. Mit zunehmender Anzahl von gezeigten Mustern erreichen fast alle Verfahren die gleiche Klassifikationsgenauigkeit, der PBAC Algorithmus liegt um ca. 3 % leicht vorne.

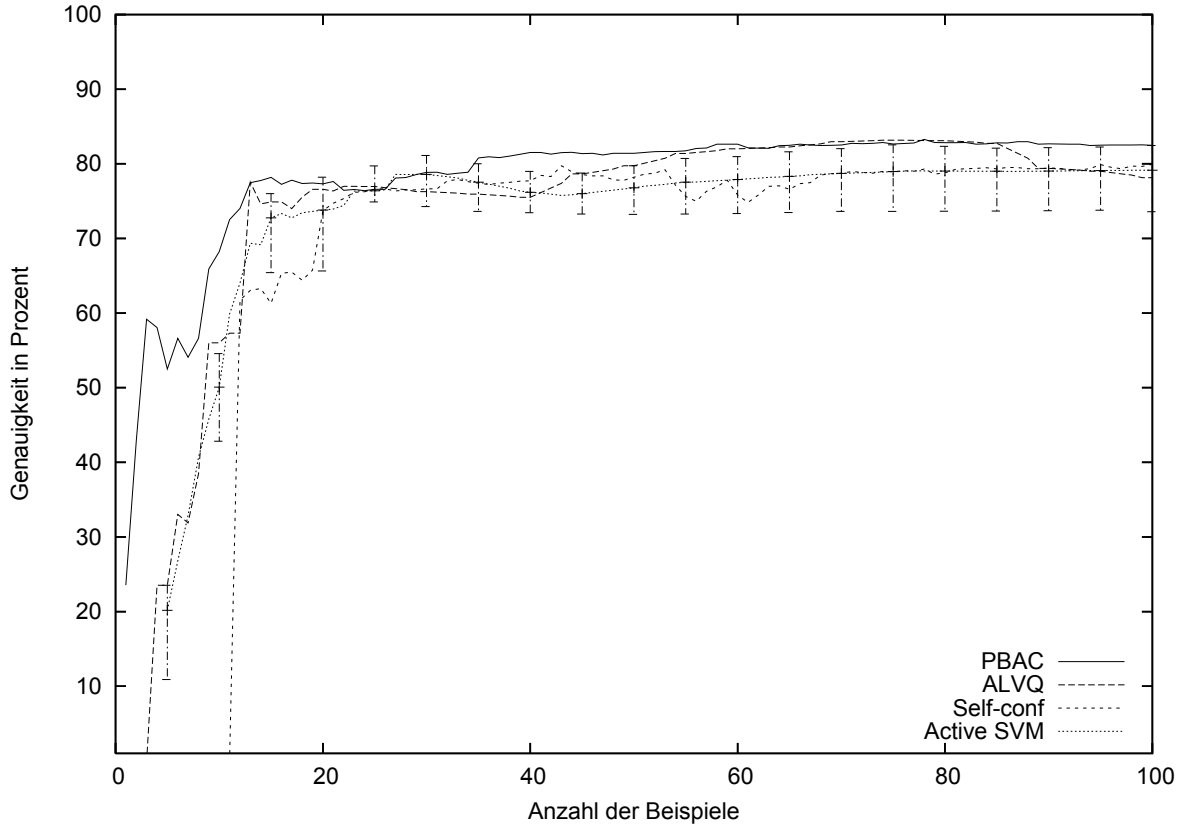


Abbildung 7.8.: Satimage Daten.

7.2.3. Diabetes Daten

Die Diabetes Daten aus dem UCI Repository (Asuncion und Newman, 2007) bestehen aus Datensätzen von ambulanten Diabetes Patienten. Acht verschiedene numerische Merkmale beschreiben den Lebensstil des Patienten beispielsweise mit Glukose- und Insulinwert. Es existieren zwei verschiedene Klassen.

Die Lernkurven in Abbildung 7.9 zeigen, dass alle Verfahren sehr schnell eine gute Klassifikationsgenauigkeit erreichen, offensichtlich lässt sich dieser Datensatz recht leicht explorieren. Bei den ersten 50 gezeigten Mustern verhalten sich der Self-conf Algorithmus das PBAC-Verfahren besser als die anderen Verfahren. Besonders auffällig bei den

Diabetes-Daten ist die instabile Performanz der aktiven SVM im Bereich der ersten 30 gezeigten Muster. Da es sich bei diesem Datensatz um ein Problem mit zwei Klassen handelt, wurde auch das DWUS-Verfahren angewendet. Dieses verhält sich (auch) sehr stabil bei der Initialisierung, weitere gezeigte Muster bewirken jedoch kaum noch eine Änderung. Die Performanz liegt nach ca. 50 gezeigten Mustern gleichauf mit der Performanz der anderen Verfahren. Die Klassifikationsgenauigkeit aller Verfahren liegt nach 100 gezeigten Mustern sehr eng beieinander, der Self-conf Algorithmus liegt geringfügig vorn. Auch der Vergleich mit einer zufälligen Selektionsstrategie im Anhang zeigt, dass dieser Datensatz leicht zu explorieren ist. Bei der aktiven SVM und dem self-conf Algorithmus wirkt sich die aktive Lernstrategie positiv auf die Stabilität bei den ersten 20 gezeigten Mustern aus.

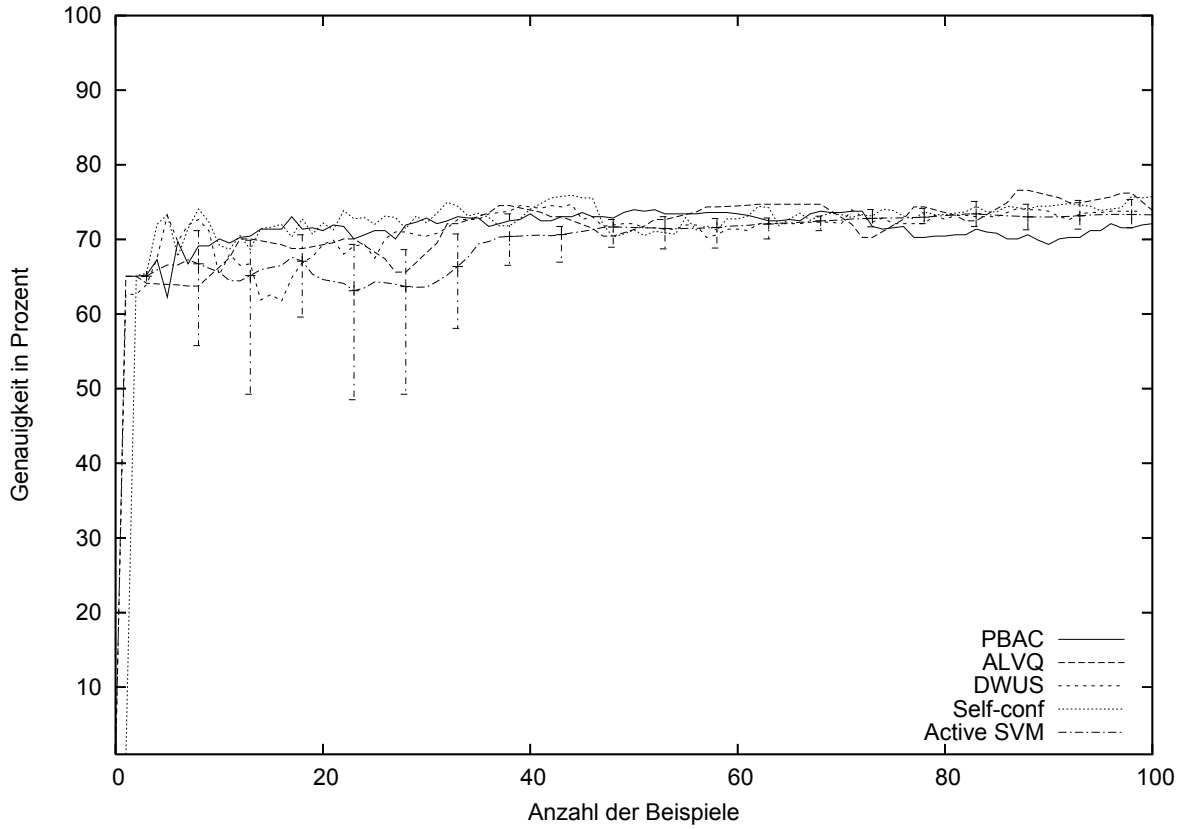


Abbildung 7.9.: Diabetes Daten.

7.2.4. Spam Daten

Die Spambase Daten aus dem UCI Repository (Asuncion und Newman, 2007) bestehen aus normalen und spam-E-Mails. Die E-Mails werden durch 58 numerische Attribute beschrieben, welche hauptsächlich die Häufigkeit eines bestimmten Wortes wiedergeben. Zusätzlich werden noch die Anzahl der Großbuchstaben gemessen. Es existieren zwei Klassen für „spam“ und „kein spam“.

Sowohl der PBAC-Algorithmus, der ALVQ Algorithmus als auch die aktive SVM gewinnen schnell an Klassifikationsgenauigkeit, fallen dann jedoch um ca. 10 % ab. Beim PBAC Algorithmus wird der Abfall jedoch schneller wieder ausgeglichen. Der self-conf Algorithmus benötigt deutlich mehr Muster, um eine stabile Klassifikation zu erreichen. Er liegt in folgenden Iterationen jedoch zusammen mit dem PBAC Algorithmus deutlich über der Performanz von ALVQ und der aktiven SVM. Die gute Performanz wird allerdings nicht durch die aktive Selektionsstrategie errreicht, wie die erweiterten Experimente im Anhang zeigen. Offensichtlich ist ein Naiver Bayes Klassifikator für diesen Datensatz gut geeignet. Der DWUS Algorithmus initialisiert sehr stabil, verzeichnet mit weiteren gezeigten Mustern aber auch Sprünge in der Klassifikationsgenauigkeit von $\pm 10\,\%$. Nach 100 gezeigten Mustern liegt die Performanz des DWUS Algorithmus im Mittelfeld.

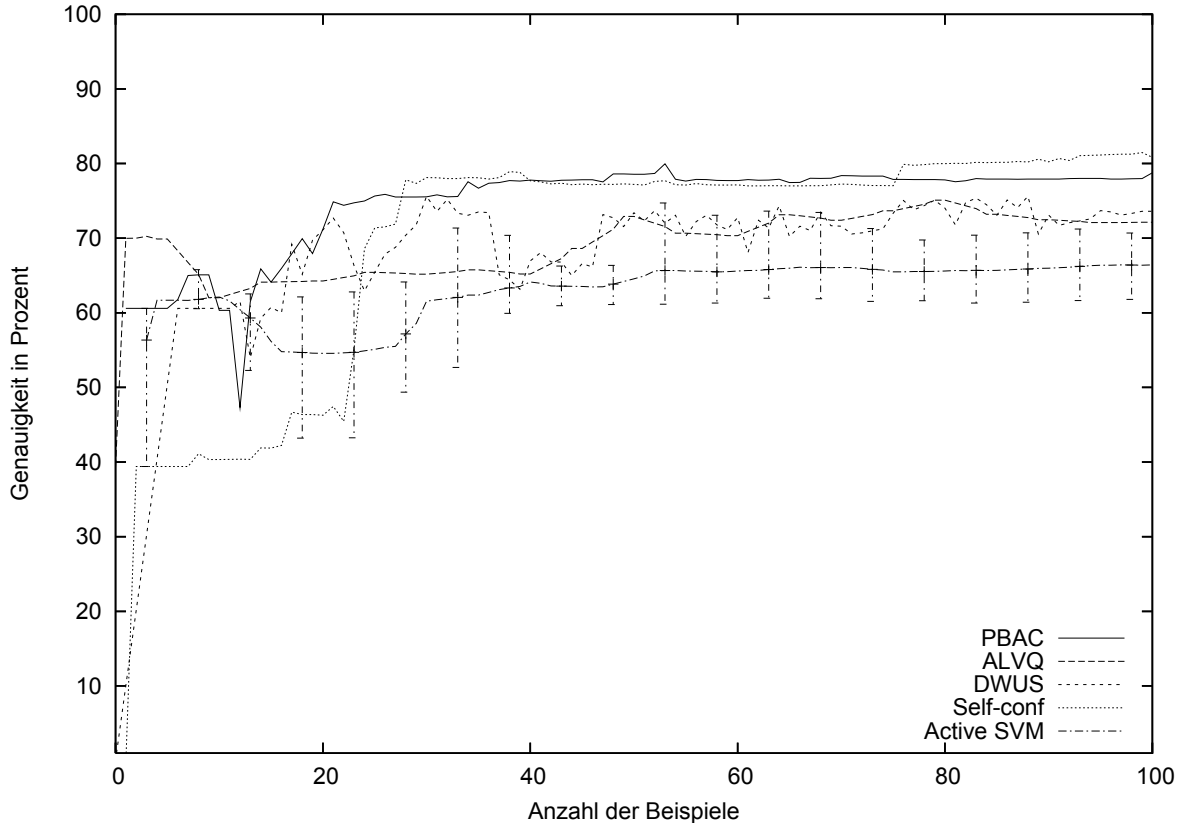


Abbildung 7.10.: Spam Daten.

7.2.5. Vehicle Daten

Die Vehicle Daten aus dem UCI Repository (Asuncion und Newman, 2007) bestehen aus 18 Merkmalen von der Silhouette von vier verschiedenen Fahrzeugen. Die Fahrzeuge wurden fotografiert und anschließend wurde ein Binärbild der Aufnahme erzeugt. Als Merkmale dienen dann Kompaktheit, Rundheit und verschiedene Verteilungsmerkmale bezüglich der Hauptachse im Binärbild.

Die Initialisierung mit den ersten 50 Mustern (Abbildung 7.11) verläuft bei allen Algorithmen ähnlich, lediglich bei der aktiven SVM treten große Schwankungen auf. Die Performanz der aktiven SVM liegt jedoch mit zunehmender Anzahl von gezeigten Mustern deutlich über der Performanz der restlichen Verfahren und wird stabiler. Die erweiterten Experimente im Anhang zeigen, dass die Performanz der SVM auf diesem Datensatz generell besser ist als die der anderen Klassifikationsverfahren. Das PBAC-Verfahren liegt ca. 8 % hinter der Genauigkeit der aktiven SVM, gefolgt vom ALVQ-Algorithmus. Der Self-conf Algorithmus schneidet deutlich schlechter als die anderen Verfahren ab.

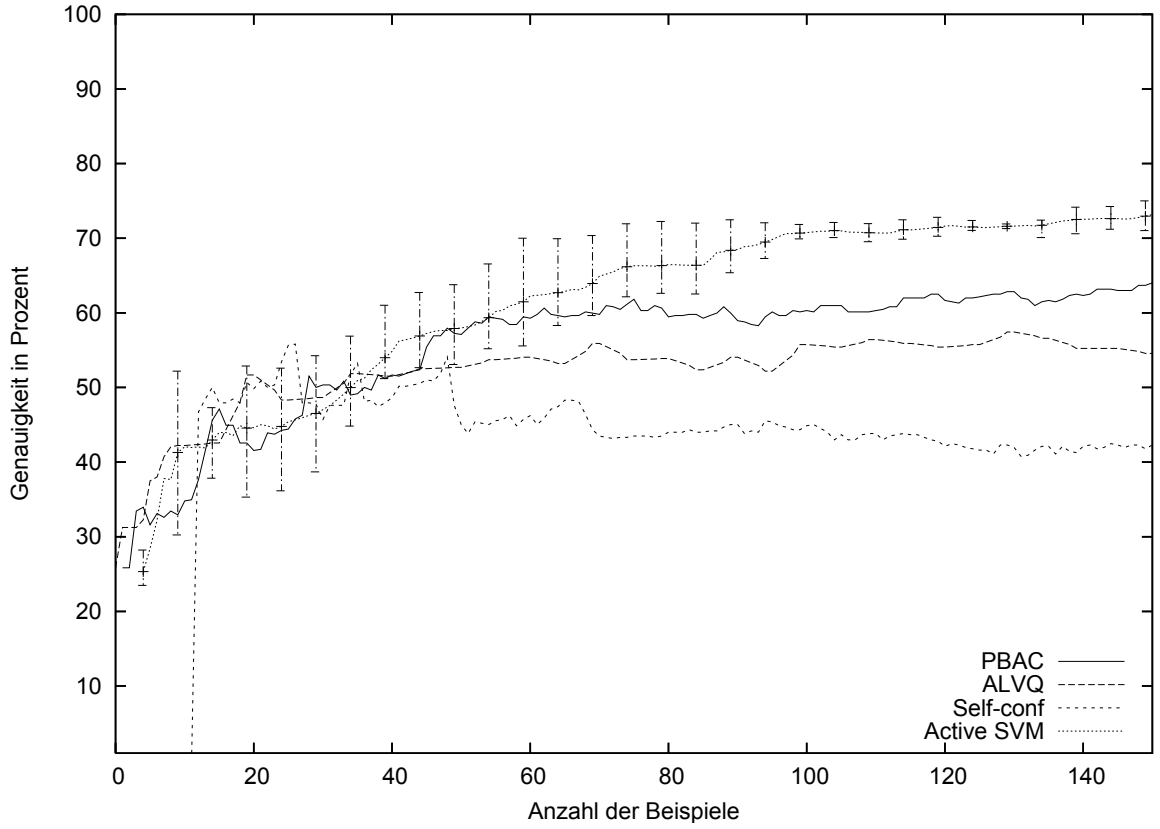


Abbildung 7.11.: Vehicle Daten.

7.2.6. Pendigits Daten

Die Pendigits Daten aus dem UCI Repository (Asuncion und Newman, 2007) bestehen aus x,y-Koordinaten von handgeschriebenen Zahlen. Es existieren 10 verschiedene Klassen in diesem Datensatz (die Ziffern 0-9).

Abbildung 7.12 zeigt die stabile und schnelle Initialisierung des PBAC und des ALVQ Algorithmus. Der ALVQ-Algorithmus fällt mit zunehmender Anzahl von gezeigten Mustern jedoch deutlich ab. Der Self-conf Algorithmus benötigt viele Muster zur Initialisierung und bleibt deutlich hinter den anderen Verfahren zurück. Im Gegensatz zu allen anderen Verfahren wirkt sich die aktive Lernstrategie negativ auf die Performanz des Klassifikators aus (siehe Anhang). Die Performanz der aktiven SVM pendelt sich mit zunehmender Anzahl von gezeigten Mustern zwischen die Performanz von PBAC und ALVQ ein. Der PBAC Algorithmus hat nach ca. 70 gezeigten Mustern die deutlich beste Performanz von allen Verfahren.

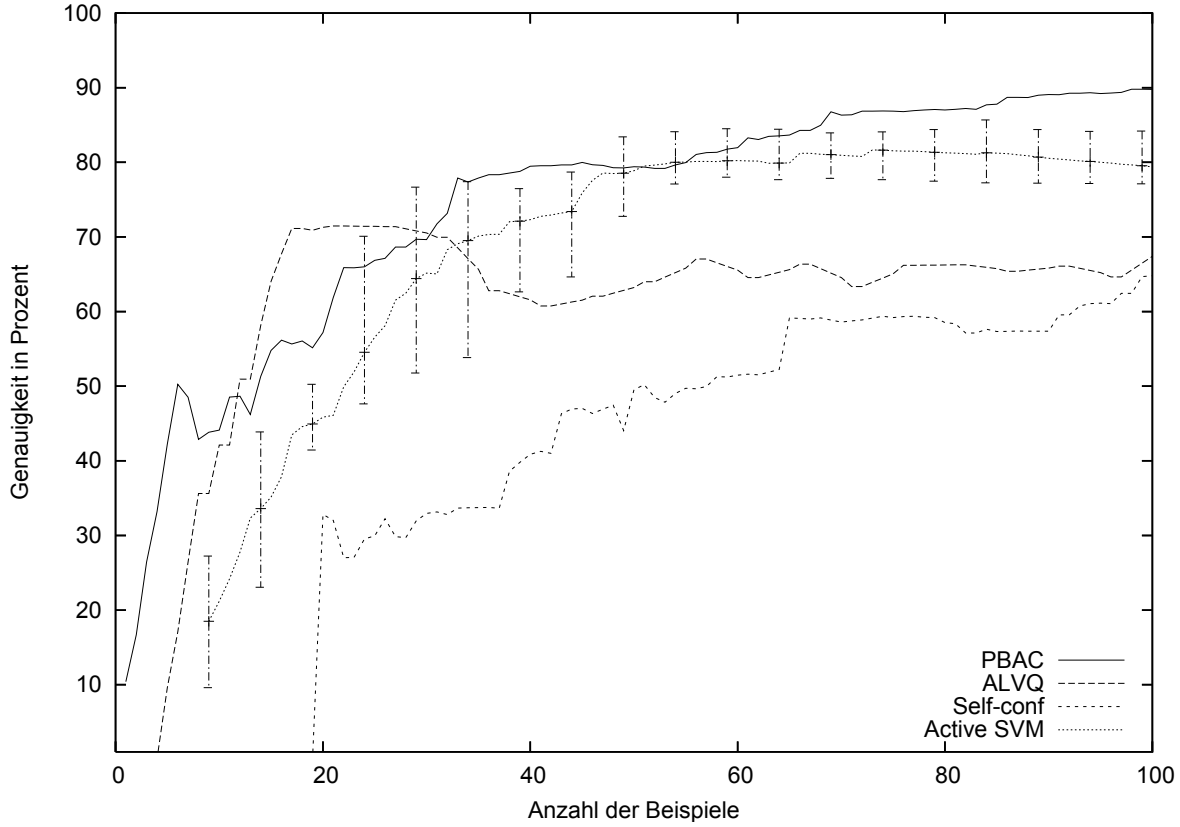


Abbildung 7.12.: Pendigits Daten.

7.3. Faces Daten

Die Faces Daten von (Mitchell und Blum, 1994) bestehen aus 640 Aufnahmen von Gesichtern von 20 verschiedenen Studenten und Mitarbeitern der Informatik an der Carnegie Mellon Universität. Pro Person existieren 32 Aufnahmen, welche in der Pose (nach links/rechts/oben schauend), Ausdruck (fröhlich/traurig) und mit/ohne Sonnenbrille variieren. Die Bilddaten wurden mit Hilfe der in Kapitel 8 vorgestellten Module eingelesen und die Zernike-Merkmale 5. Ordnung berechnet. Von den 29 komplexen Momenten wurde der Betrag genommen und normalisiert. Das Ziel dieser Klassifikation besteht darin, die richtige Person auf einem Bild vorherzusagen. 30 Prozent der Daten wurden zum Lernen verwendet, die restlichen 70 Prozent zum Testen. Abbildung 7.13 zeigt die

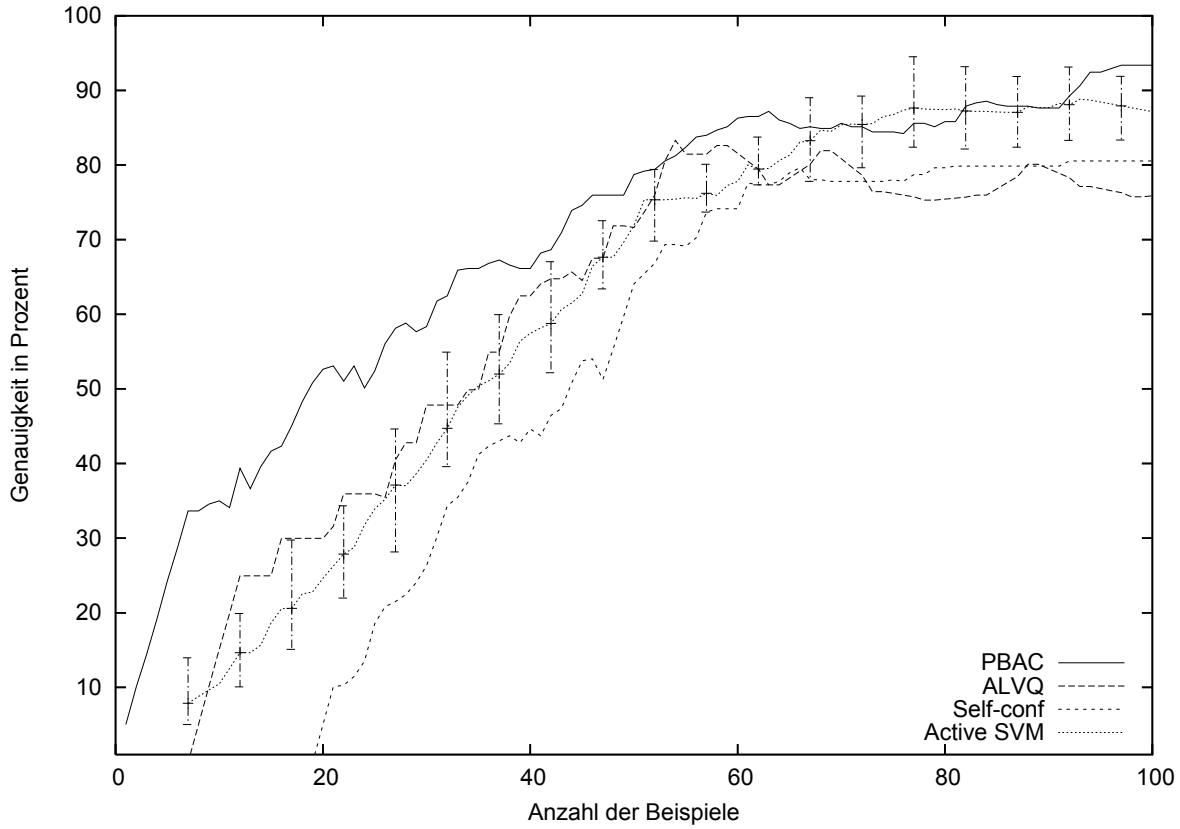


Abbildung 7.13.: Faces Daten.

Lernkurven der unterschiedlichen Verfahren für dieses Klassifikationsproblem. Die hohe Anzahl der Klassen bewirkt bei allen Verfahren, dass viele Muster benötigt werden, um eine akzeptable Klassifikationsgenauigkeit zu erreichen. Die Performanz des PBAC Verfahrens liegt in den ersten Iterationen deutlich höher als bei den anderen Verfahren. Nach

ca. 45 gezeigten Mustern gleicht sich die Performanz der aktiven SVM im Durchschnitt dem PBAC Verfahren an. Nach 100 gezeigten Mustern liegt die Performanz des PBAC Verfahrens bei 93,36 %, die aktive SVM erreicht durchschnittlich 87,14 %. Gemessen an der Performanz eines Klassifikators mit allen gezeigten Mustern liegt die aktive SVM damit hinter dem PBAC Algorithmus (siehe Anhang). Der ALVQ Algorithmus verhält sich in der Initialisierung sehr stabil. Mit weiteren gezeigten Mustern steigt die Klassifikationsgenauigkeit nicht weiter und bleibt mit 75,88 % unter der Performanz der anderen Verfahren. Jedoch ist dieser Wert nahe an der Performanz des LVQ Algorithmus mit allen gezeigten Mustern. Der Self-conf Algorithmus bleibt zu Beginn deutlich hinter den anderen Verfahren zurück, die Performanz nach 100 gezeigten Mustern beträgt 80,54 %.

7.4. Fazit

In diesem Kapitel wurden die populärsten aktiven Lernverfahren bezüglich ihrer Performanz mit den in dieser Arbeit vorgestellten Ansätzen ALVQ und PBAC verglichen.

Grundsätzlich hängt die Performanz der Klassifikationsverfahren vom verwendeten Datensatz ab. Die Stärken des Naive Bayes Klassifikators liegen vor allem in der Textklassifikation (wo er auch häufig verwendet wird) wie z. B. bei den Spam-Daten. Die SVM eignet sich besonders, um mit Hilfe eines geeigneten Kernels die Daten zu separieren - dies ist deutlich bei den Vehicle-Daten zu sehen. Das Prototypen basierte Klassifikationsverfahren LVQ hat im Vergleich zu den anderen Klassifikationsverfahren im Durchschnitt eine schlechtere Performanz. Das k-nächste Nachbarn Verfahren profitiert (im Gegensatz zum LVQ-Verfahren) von vielen klassifizierten Referenzmustern, mit denen die Klassifikationsgrenzen genauer abgebildet werden können. Dieser Vorteil geht bei einer aktiven Klassifikationsstrategie verloren, durch eine geeignete Auswahl von Mustern kann dies jedoch ausgeglichen werden.

Zunächst lässt sich feststellen, dass das ALVQ und PBAC-Verfahren mit wenigen Parametern generell alle Datensätze mit wenigen initialen Mustern stabil klassifizieren. Im Gegensatz zu allen anderen Verfahren sind keine a-priori Informationen über den Datensatz (wie z. B. Anzahl der Klassen, gewählte Muster aus jeder Klasse, Tuning der Kernel-Funktion am Datensatz) notwendig.

Als problematisch zu sehen ist die hohe Varianz der aktiven SVM vor allem bei den ersten gezeigten Mustern und die spezielle Auswahl einer geeigneten Kernel-Funktion. Ansonsten ist die Performanz nach einer genügend großen Anzahl von Mustern stabil

und liegt meistens nahe oder über dem Durchschnitt aller aktiven Lernverfahren. Jedoch liegt dies nicht immer an der aktiven Selektionsstrategie, wie die erweiterten Experimente im Anhang aufzeigen. Bei manchen Datensätzen hat eine zufällige Auswahl von Mustern eine ähnlich gute Performanz.

Das DWUS-Verfahren zeigt in der Initialphase durch das Clustering eine hohe Stabilität. Bei weiteren gezeigten Mustern, in denen sich die anderen Verfahren stärker auf Muster an den Klassengrenzen fokussieren, steigt die Performanz durch das dichtebasierte sampling nur langsam. Problematisch ist vor allem die Begrenzung auf 2-Klassen Probleme.

Der Self-conf Algorithmus benötigt eine initiale Anzahl von Mustern aus jeder Klasse. Dies ist vor allem bei Datensätzen mit mehreren Klassen ein Problem. Vor allem zu Beginn liegt die Performanz meistens leicht unter den anderen Verfahren. Nach mehreren gezeigten Mustern liegt die Performanz im Durchschnitt unter der Performanz der anderen aktiven Lernverfahren. Im Vergleich zur zufälligen Ausahl von Mustern zeigt sich bei den unterschiedlichen Datensätzen eine hohe Variabilität. Manchmal liegt die Performanz der aktiven Lernstrategie deutlich über der passiven Auswahl, manchmal deutlich darunter.

Der ALVQ-Algorithmus zeigt durch das initiale Clustering ebenfalls eine stabile Performanz in den ersten Lerniterationen. Das nachfolgende LVQ-Lernen mit ausgewählten Mustern bringt oft nur eine leichtere Verbesserung des Klassifikationsergebnisses, zum Teil fällt die Performanz auch ab. Die Verschiebung der Prototypen ermöglicht nur eine grobe Anpassung der Klassifikation. Sinnvoll wird dies erst, wenn ausreichend viele Prototypen existieren. Im Vergleich zur zufälligen Auswahl von Mustern schneidet das Verfahren besser ab. Dies liegt an der wesentlich stabileren Initialisierung durch das Clustering-Verfahren in der ersten Phase des Algorithmus.

Der PBAC-Algorithmus liegt sowohl bei der Initialisierung als auch bei der Performanz nach mehreren gezeigten Mustern im Durchschnitt über alle Datensätze klar vorn. Gegenüber einer zufälligen Auswahl von Mustern ist die vorgestellte aktive Selektionsstrategie immer vorzuziehen, wie die erweiterten Experimente im Anhang zeigen. Vor allem bei Mehrklassenproblemen ist die Performanz deutlich besser. Sie hängt vom Explorations-Radius und dem Spezialisierungs-Faktor ϵ ab. Mit den hier vorgestellten Parameter-Einstellungen lassen sich in allen Fällen gute Ergebnisse erzielen. Die Abhängigkeit der Performanz von den Parametern ist weitaus niedriger als z. B. bei der aktiven SVM. Auch sonst sind keine Vorkenntnisse über die Daten und die möglichen enthaltenen Klassen

nötig, was dieses Verfahren ideal für den industriellen Einsatz macht.

8. Klassifikation von Zell-Assays

In diesem Kapitel wird die Anwendung der entwickelten Algorithmen in einer realen Umgebung demonstriert. Dazu wird zunächst die Problemstellung erläutert und die nötigen Schritte der Vorverarbeitung aufgezeigt, die nötig sind, um das Problem mit Hilfe eines aktiven Lernverfahrens zu lösen.

8.1. Problemstellung

In dieser Problemstellung widmen wir uns dem Bereich der Bioinformatik, in dem die Entwicklung von Hochgeschwindigkeitskameras zur Analyse von Zellbildern neue Einsatzmöglichkeiten im Bereich der Wirkstoffanalyse eröffnet. Diese Geräte sind in der Lage, hunderttausende Zell-Assay Bilder innerhalb eines einzigen Tages zu produzieren.

Abbildung 8.1 zeigt ein typisches Gerät zur Hochdurchsatzanalyse. Die Pippetierung der einzelnen Mikrotiterplatten erfolgt durch einen Roboter. In jedem Loch auf der Platte (welches als *well* bezeichnet wird) befindet sich eine kleine Zellkultur. Oft handelt es sich um fluoreszierende Zellen, welche auf eine Substanz aus einer Testreihe reagieren. Diese Daten werden genutzt, um potentielle Effekte der Substanzen zu testen.

Die Analyse jedes einzelnen Bildes durch einen Biologen ist dabei kaum noch möglich. Daher werden bisher in aufwändiger Kleinarbeit spezielle, an das Problem angepasste Skripte zur Klassifikation solcher Bilder verwendet. Das Ziel unserer Arbeit ist es, den Prozess der Zellbild-Klassifikation zu automatisieren. Dazu wird eine Hand voll interessanter Muster in den Bilddaten selektiert und dem Orakel (in diesem Fall dem Biologen)[1] präsentiert. Basierend auf diesen wenigen, hand-klassifizierten Beispielen soll ein Modell gelernt werden, das anschließend in der Lage ist, den Rest der Bilddaten zu klassifizieren.

[1] In diesem Kapitel verwenden wir durchgängig den Begriff Biologe für das Orakel, da dieser die Klassifikation der Zellbilder vornimmt.

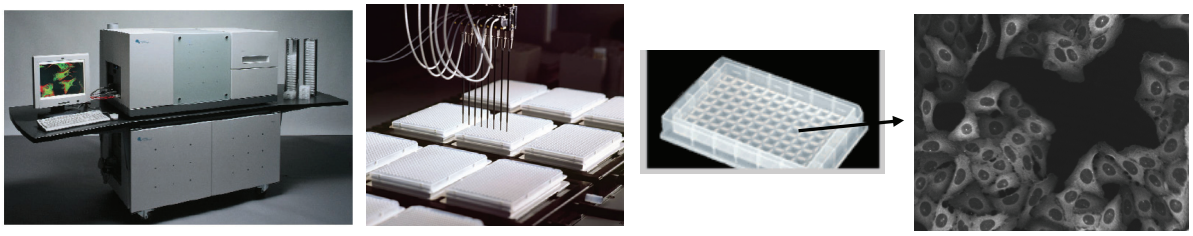


Abbildung 8.1.: Geräte und Ausstattung zur Hochdurchsatzanalyse.

8.2. Vorverarbeitung

Um ein Zell-Assay Bild im Ganzen zu klassifizieren, sind einige Vorverarbeitungsschritte notwendig. Abbildung 8.2 stellt den Ablauf schematisch dar. Zunächst werden die Bilder eingelesen und über Informationen in den Dateinamen den entsprechenden Koordinaten auf der Platte zugeordnet. Der Biologe kann weitere Informationen einfließen lassen, z. B. welche Bilder von so genannten Referenzplatten kommen (hier wird ein bekannter Wirkstoff in hoher Konzentration verwendet, um gezielt eine gewünschte Reaktion in den Zellen hervorzurufen) oder welche Bilder nicht für die Klassifikation verwendet werden sollen. Referenzbilder lassen sich gut dazu verwenden, die relevanten Merkmale für eine Klassifikation zu bestimmen, siehe Abschnitt 8.3. In der Vorverarbeitung werden numerische Merkmale über das Gesamtbild berechnet (siehe auch Abschnitt 8.2.2). Diese numerischen Eingabemuster werden mit Hilfe des in Abschnitt 4.1 vorgestellten Fuzzy c-means Algorithmus mit Erkennung von Rauschen geclustert. Die Bilder, welche als „Rauschen“ erkannt werden, werden direkt aussortiert und müssen vom Biologen gesondert betrachtet werden. Üblicherweise handelt es sich hierbei um grobe Ausreißer wie z. B. Bilder, die stark überbelichtet sind.

Die weiteren Schritte der Segmentierung und Merkmalsextraktion werden in den nächsten Abschnitten genauer erläutert.

Abbildung 8.3 zeigt ein typisches Zellbild aus einem so genannten Translokations-Assay. Der Wirkstoff ist vom Zellkern in das Zytoplasma eingedrungen und fluoresziert dort.

8.2.1. Segmentierung

Da die Reaktionen der einzelnen Zellen auf den Wirkstoff im Bild unterschiedlich ausfallen können, muss jede einzelne Zelle im Bild gesondert betrachtet werden. Die Gesamt-

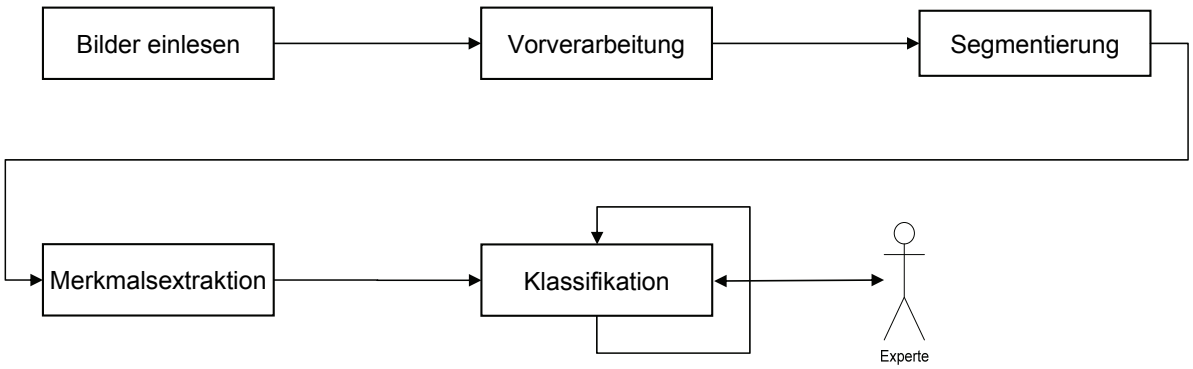


Abbildung 8.2.: Schematischer Ablauf für die Klassifikation der Bilder.

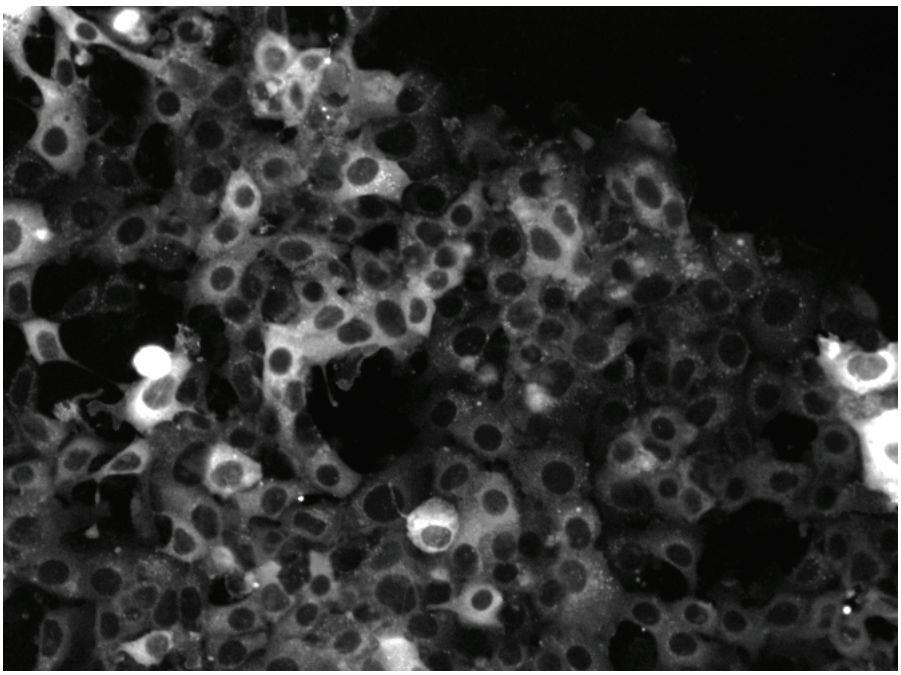


Abbildung 8.3.: Typisches Zellbild aus einem Translokations-Assay.

klassifikation des Bildes erfolgt durch eine Mehrheitsentscheidung über die Einzelbilder der Zellen in einem Bild. Die Bildsegmentierung dient dem Übergang von Pixelwerten zu einer symbolischen Darstellung des Bildes in Teilbereiche mit ähnlichen Eigenschaften. Sie dient der Erkennung von Objekten (in diesem Anwendungsfall Zellen), eine Klassifikation der Objekte erfolgt in diesem Fall erst zu einem späteren Zeitpunkt.

In dieser Arbeit wurden verschiedene Segmentierungsalgorithmen entwickelt, welche im Folgenden kurz vorgestellt werden sollen:

Schwellwert-Verfahren stellen eines der einfachsten Verfahren zur Segmentierung eines Bildes dar. Sie sind besonders gut geeignet, um Vorder- und Hintergrund in einem Grauwertbild voneinander zu trennen. Abhängig von einem Schwellwert t werden alle Pixel in einem Bild mit Grauwert $\leq t$ auf die Hintergrundfarbe und alle Pixel mit Grauwert $> t$ auf die Vordergrundfarbe gesetzt. Die Schwelle t muss abhängig von den zugrunde liegenden Bildern definiert werden. Dies kann zum einen durch manuelle Schwellwertanpassung für ein spezielles Zell-Assay geschehen oder durch adaptive Verfahren wie z. B. das Otsu-Thresholding (Otsu, 1978). Hierbei wird angenommen, dass eine bimodale Verteilung der Grauwerte im Histogramm existiert und versucht, die Varianz zwischen beiden Klassen zu maximieren. Ein häufig auftretendes Problem bei Schwellwert-Verfahren besteht darin, dass ein Schwellwert nicht global für das ganze Bild gilt, z. B. bei ungleichmäßiger Ausleuchtung. Daher kann der Schwellwert in jedem $m \times n$-Ausschnitt des Bildes separat berechnet werden. Abbildung 8.4a zeigt das Ergebnis einer Schwellwert-Segmentierung.

Künstliche Neuronale Netze lassen sich auch für die Segmentierung von Zellbildern einsetzen, obwohl sich dieses Verfahren eher den Verfahren der Mustererkennung zuordnen lässt. Der Benutzer markiert in einigen Trainingsbildern per Mausklick die Position von Zellen. Anschließend wird ein $m \times n$ großes Fenster pixelweise von links nach rechts und von oben nach unten über das Bild bewegt. Die Grauwerte der Pixel stellen den Eingabevektor für ein künstliches neuronal Netz dar. Befindet sich eine Markierung im zentralen Bereich des Fensters, so wird der Bildausschnitt als „Zelle“ markiert, ansonsten als „Hintergrund“. Abbildung 8.4b zeigt das Ergebnis einer Segmentierung mithilfe eines künstlichen neuronalen Netzes. Dieses Verfahren ist besonders gut geeignet, wenn dieselben Grauwerte in Objekten und im Hintergrund auftreten.

Flächen-Wachstum Algorithmen gehören zu den so genannten regionenbasierten Segmentierungsverfahren. Eine so genannte seed region wird aufgrund eines Ähnlichkeitsmaßes zu den Pixeln in ihrer unmittelbaren Nachbarschaft erweitert. Wir verwenden zur Segmentierung der Zellbilder ein erweitertes Verfahren von (Jones u. a., 2005). Hierbei wird als Ähnlichkeitsmaß der Gradient auf einer weich gezeichneten Version des Bildes verwendet. Zusätzlich fließt die euklidische Distanz in das Ähnlichkeitsmaß ein. Hiermit lässt sich beeinflussen, ob die Grenze zwischen zwei Zellen mehr auf den lokalen Bildmerkmalen oder auf der euklidischen Distanz zwischen den Zellen beruht. Das Verfahren wurde um ein Abbruchkriterium erweitert, welches das Wachstum in eine Richtung

stoppt, wenn der Gradient einen vordefinierten Schwellwert überschreitet. Somit lässt sich verhindern, dass der Hintergrund in die Segmentierung mit einbezogen wird. Um die seed regions zu erhalten, wird die Kernfärbung eines Bildes verwendet und das im ersten Punkt vorgestellte Schwellwert-Verfahren angewendet. Anschließend wird der Algorithmus auf dem Signalbild angewendet, Abbildung 8.4c zeigt das Ergebnis. Die gelb umrandeten Objekte sind die seed regions (Zellkerne), die grün umrandeten Objekte das Ergebnis des Flächenwachstums (Zytoplasma).

8.2.2. Extraktion von Merkmalen

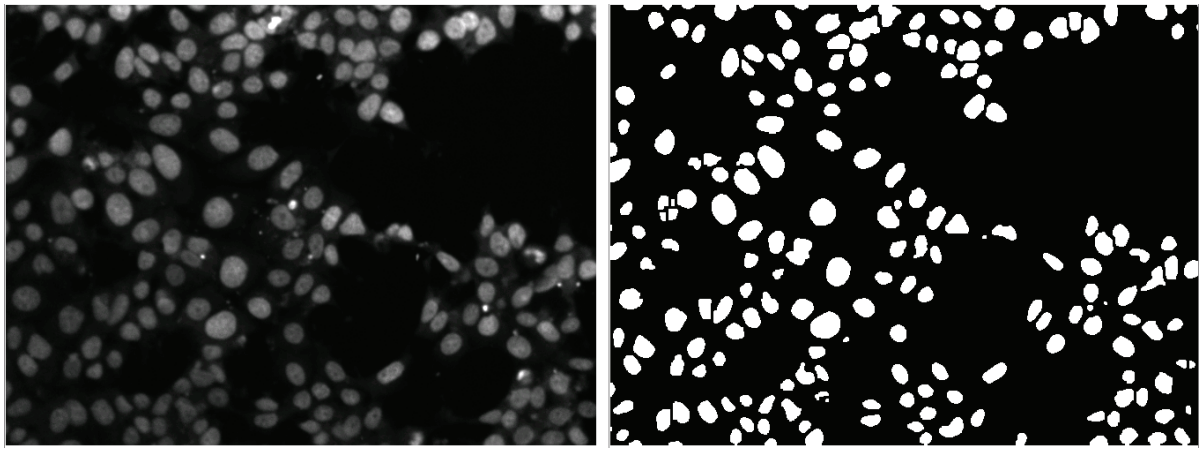
Um die einzelnen Zellbilder aus verschiedenen Sichten zu beschreiben, stehen verschiedene Module zur Merkmalsextraktion zur Verfügung, welche im Folgenden kurz erläutert werden. Die numerischen Merkmale eines Bildes werden in einem Merkmalsvektor zusammengefasst und für die Klassifikation verwendet, siehe Abbildung 8.5. Mit Hilfe dieser Merkmale lassen sich die Zellbilder bezüglich ihrer Helligkeitsverteilung, Textur und Form beschreiben.

Histogramm-Merkmale

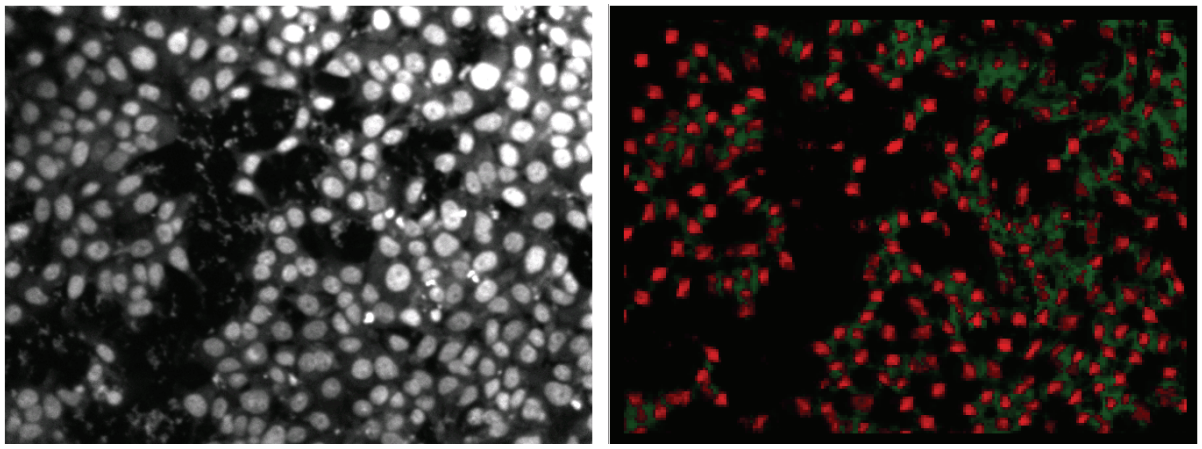
Ein Histogramm dient der grafischen Darstellung der Häufigkeitsverteilung quantitativer Merkmale; in einem 8-bit Grauwertbild sind dies z. B. die Pixel aus 256 verschiedenen Klassen (Grauwerten). Der Flächeninhalt jeder Klasse ist proportional zur relativen Häufigkeit der auf diese Klasse entfallenden Elemente. Basierend auf dieser Häufigkeitsverteilung lassen sich verschiedene statistische Momente berechen, wie z. B. Erwartungswert (Mittelwert), Varianz, Schiefe und Wölbung.

Textur-Merkmale

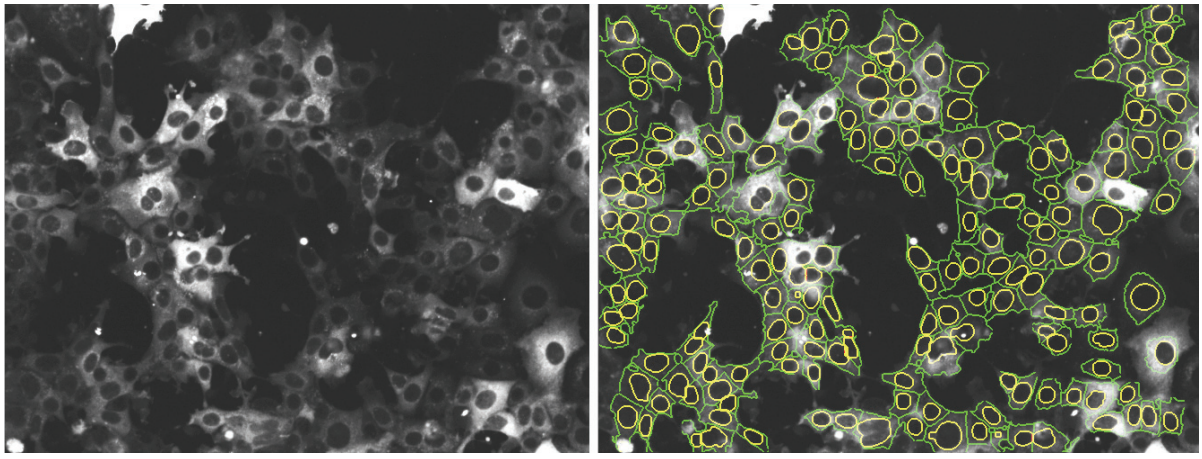
Die Textur-Merkmale beschreiben die Struktur im Bild (Oberflächenbeschaffenheit) anhand der Struktur der Grauwerte. Eine weit verbreitete Methode zur Berechnung der Texturmerkmale ist die Berechnung von so genannten *Cooccurence-Matrizen*. Hier wird das paarweise Auftreten von Grauwerten über einen Abstand d gezählt. Die *Cooccurence-Matrizen* in horizontaler, vertikaler, diagonaler und antidiagonaler Richtung werden gemittelt, um eine gewisse Rotationsinvarianz für die Merkmale zu erreichen. Auf der resultierenden Matrix lassen sich verschiedene statistische Merkmale berechnen. In diesem Modul werden 14 verschiedene Textur-Merkmale berechnet, die in der Arbeit von



(a) Schwellwert-Segmentierung



(b) Segmentierung mit künstlichen neuronalen Netzen



(c) Flächen-Wachstum mit Voronoi-Begrenzung

Abbildung 8.4.: Eingesetzte Verfahren der Segmentierung im Überblick.

Key	Mask	000|0	000|1	D ASM	D Contrast	D Correla...	D Variance	D IDFM	D SumAv...	D SumEnt...	D Entropy	D DiffEntr...	D ICM1	D ICM2	D Mean
001|001_1				0.003	40.951	8,278,475.228	865.402	0.437	58.379	3,853,049,9...	7.219	3.357	-0.349	0.963	27.476
001|001_2				0.003	43.955	14,312,165....	1,442.186	0.412	74.935	3,850,449,5...	7.199	3.355	-0.395	0.976	35.807
001|001_3				0.006	24.457	3,298,290.137	567.331	0.542	46.609	3,858,187,6...	6.79	3.097	-0.358	0.955	21.262
001|001_4				0.001	95.142	67,850,007....	3,482.665	0.297	115.29	3,828,314,9...	7.798	3.702	-0.424	0.988	54.682

Abbildung 8.5.: Einzelne Zellen und ihre assoziierten numerischen Merkmale.

(Haralick, 1973) vorgeschlagen werden, wie z. B. Entropie, Kontrast, Homogenität und Korrelation.

Zernike-Momente

Zernike-Polynome (Zernike, 1934) existieren schon seit längerer Zeit in der theoretischen Optik. Basierend auf diesen Polynomen wurden von (Teague, 1980) orthogonale komplexe Momente entwickelt. Die Zernike-Momente entstehen durch die Projektion der Bildfunktion $B(x, y)$ auf die Zernike-Polynome. Rotationsinvarianz der Merkmale wird erreicht, indem nur jeweils der Betrag der Momente genommen wird.

Masken-Merkmale

Basierend auf der binären Maske des Zellbildes lassen sich diverse Merkmale erzeugen, z. B. Flächeninhalt, Zirkularität, Anzahl von Regionen etc.

8.3. Klassifikation

Durch die Berechnung und Kombination verschiedener Merkmale auf den Bildern entsteht ein hochdimensionaler Merkmalsvektor. Wie in Kapitel 4 bereits angesprochen, stellen hochdimensionale Merkmalsräume eine Schwierigkeit bei der Klassifikation dar (Yang und Pedersen, 1997) (Liu und Motoda, 1998).

Das Problem der Auswahl geeigneter Attribute für eine Klassifikation wurde in bisherigen Arbeiten unter der Voraussetzung betrachtet, dass ein klassifizierter Trainingsdatensatz zur Verfügung steht. Da diese Voraussetzung in diesem aktiven Lernszenario nicht gegeben ist, wird eine Behelfslösung angewandt, welche auf den klassifizierten Referenzplatten (siehe Abschnitt 8.2) basiert. Hierbei wird für das Gesamtbild vom Biologen

eine Klassifikation vorgegeben. Diese Klassifikation wird auf alle einzelnen Zellbilder im Bild angewendet, womit ein ausreichend großer Datensatz entsteht. Auch wenn dieser Datensatz nur sehr grob klassifiziert (und damit verrauscht) ist, hat sich diese Methode in verschiedenen Anwendungsszenarien als stabil erwiesen.

Es existieren diverse Maße, um den Informationsgehalt eines Attributes zu bewerten, die jedoch an dieser Stelle nicht alle vorgestellt werden sollen. Eine vollständige Übersicht findet sich in (Liu und Motoda, 1998). In dieser Arbeit wird das Maß des Informationsgewinns (Mitchell, 1997) verwendet.

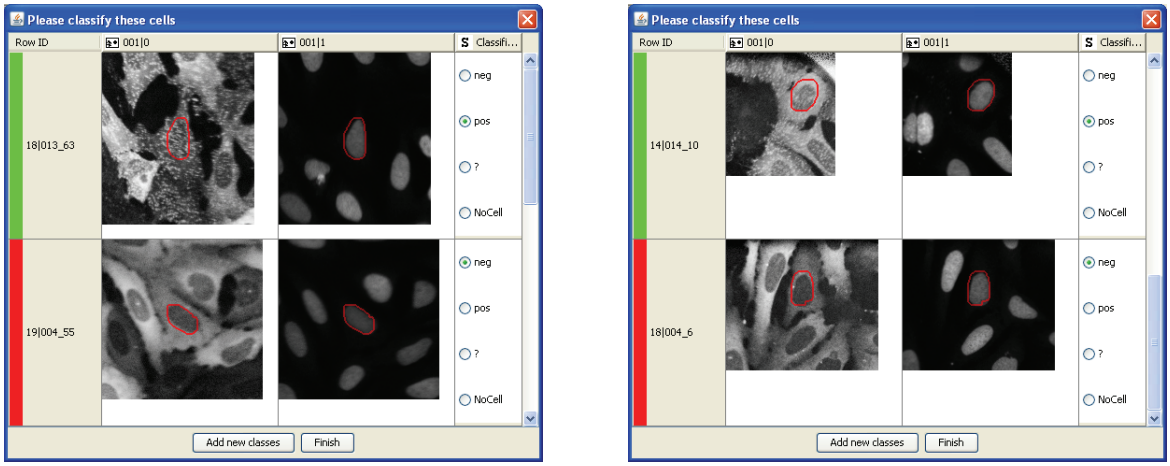
Durch den Informationsgewinn wird für jedes Attribut ermittelt, wie viele Informationen für die Unterscheidung zwischen verschiedenen Klassen gewonnen werden oder verloren gehen, wenn dieses Feature weggelassen wird. Mithilfe des Informationsgewinnes lassen sich die einzelnen Attribute in eine Rangfolge bringen, um somit nur die „besten“ Attribute in einem niedrigdimensionalen Merkmalsraum zu verwenden.

Durch den Umstand, dass jedes Muster mit einem Zellbild verknüpft ist, können die Zellbilder direkt bei der Klassifikation verwendet werden. Der Biologe wird in den Klassifikationsprozess interaktiv eingebunden. Beim PBAC Algorithmus werden die Potentiale aller Beispiele vor der Klassifikation berechnet. Somit müssen in jeder Lerniteration nur die Klassifikationsunsicherheiten berechnet werden, so dass der Biologe ohne größere Latenzzeiten in den Klassifikationsprozess eingebunden werden kann.

In jeder Lerniteration werden mehrere Muster für die Klassifikation durch den Biologen anhand des aktiven Lernalgorithmus ausgewählt. Zu jedem einzelnen Muster wird das zugehörige Bild und die zugehörigen Bilder einer kleinen Anzahl von Beispielen in der direkten Nachbarschaft dem Biologen zur Klassifikation gezeigt. Die Beispiele in der Nachbarschaft dienen zur Stabilisierung des Lernverfahrens. Macht der Biologe bei diesen Mustern widersprüchliche Angaben (klassifiziert er beispielsweise vier Muster positiv und eines negativ), so kann hier erneut nachgefragt werden, ob er sich bei der Klassifikation sicher ist. Ohnehin gleicht die k-nächste Nachbarn Klassifikation einzelne falsch klassifizierte Muster aus. Weiterhin hat der Biologe bei mehreren Beispielen die Möglichkeit, die Klassifikation von manchen Mustern auszusetzen, wenn er sich unsicher ist.

In Abbildung 8.6 sind typische Anfragen für einzelne Zellbilder zu sehen. Bei diesen Daten spielt die Körnigkeit der Textur eine große Rolle. Hier ist zu sehen, dass zu Beginn besonders unterschiedliche Beispiele gezeigt werden, wohingegen in späteren Iterationen die Klassengrenze verfeinert wird.

Basierend auf der Menge der klassifizierten Muster (Prototypen) erfolgt in jeder Lerni-



(a) Zu Beginn: Hohe Diversität (b) Weitere Iterationen: Klassengrenzen

Abbildung 8.6.: Ausgewählte Beispiele von Zellen zu unterschiedlichen Zeitpunkten.

teration eine Klassifikation aller Muster. Somit hat der Biologe eine unmittelbare Rückmeldung, wie sich die Klassifikation auf den gesamten Datensatz auswirkt. In der Anwendung des Zell-Assay Klassifikators wird dazu eine spezielle Sicht auf die Platten generiert, siehe Abbildung 8.7. Sie erlaubt es, die Klassifikation über das Gesamtbild auf der Platte

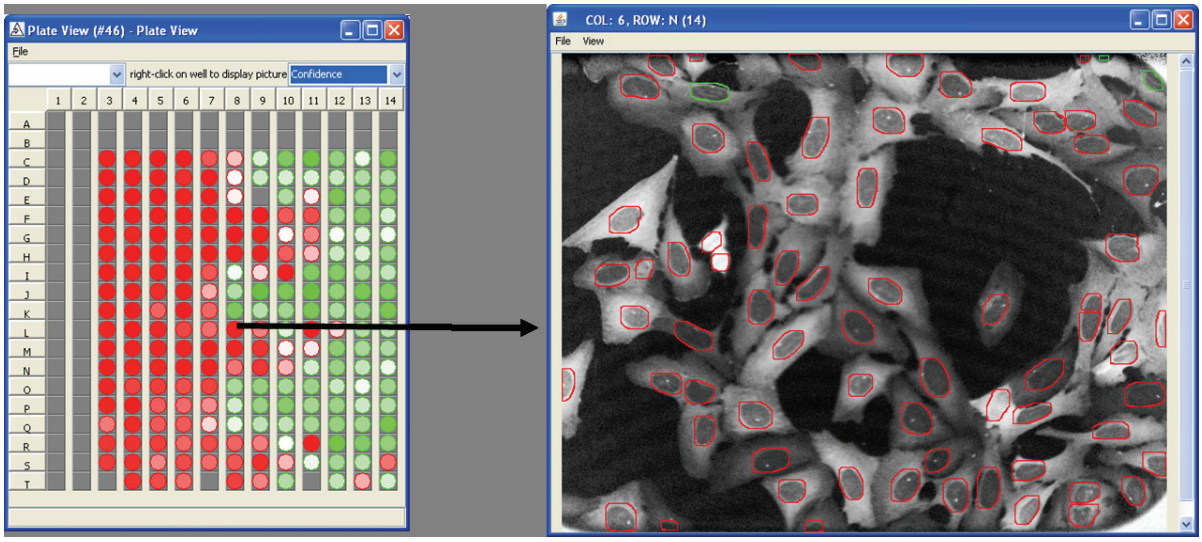


Abbildung 8.7.: Eine Sicht auf das Klassifikationsergebnis mit zwei Klassen auf der Platte aus der Konfidenz-Perspektive.

zu betrachten. Die Konfidenz ergibt sich aus dem Verhältnis der Anzahl der Zellen aus

der jeweiligen Klasse. Der Biologe hat zusätzlich die Möglichkeit, die Klassifikation der einzelnen Zellen im Gesamtbild durch Auswahl eines *wells* zu betrachten.

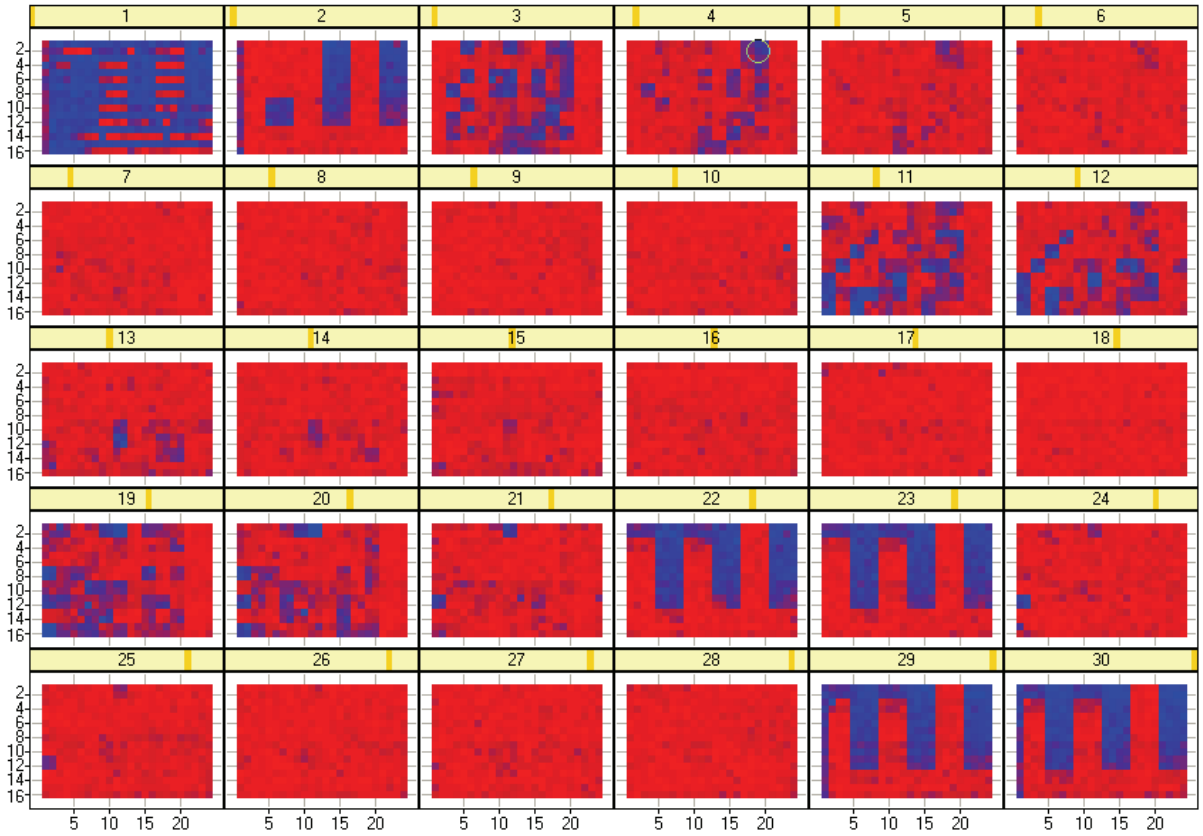
8.4. Cellminer Software Ergebnisse

Die Cellminer Software soll es dem Biologen ermöglichen, ohne Kenntnisse der Informatik bzw. der Bildverarbeitung, Zell-Assays zu klassifizieren. Die Abläufe der Bildverarbeitung können jedoch recht komplex sein, je nach Art der Bilder sind verschiedene Schritte der Vorverarbeitung, spezielle Segmentierungsalgorithmen oder spezielle Merkmalsberechnungen nötig.

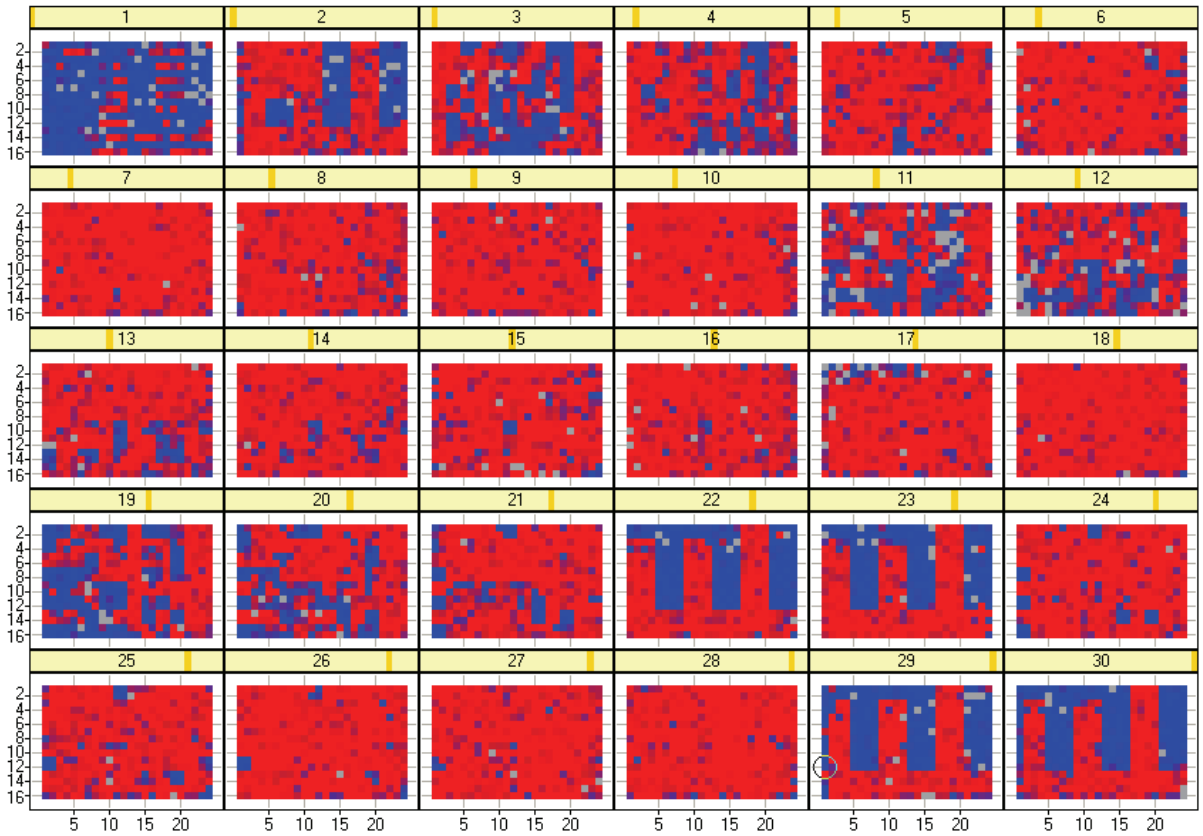
Daher stehen dem Biologen verschiedene *workflows* zur Verfügung, welche die Vorverarbeitungsschritte automatisch ausführen, je nach Typ des Experiments. Dazu werden die notwendigen Abläufe von einem Bildverarbeitungsexperten mithilfe eines *workflow-tools* definiert und parametrisiert. Diese vordefinierten *workflows* können anschließend direkt in der Cellminer Software verwendet werden.

Der Biologe wird nur in die notwendigen Prozesse eingebunden, dazu gehört die Auswahl der Bilddaten, die Angaben zur Plattenbelegung und die Klassifikation.

Ein klassifizierter Datensatz mit Testdaten steht bei der Cellminer-Anwendung nicht zur Verfügung. Jedoch wurde ein Zell-Assay mit Hilfe eines Skriptes analysiert und die Ergebnisse auf Plattenebene festgehalten. Es handelt sich hierbei um ein 2-Klassen Problem (Zellen reagieren auf den Wirkstoff oder nicht). Ein direkter Vergleich der Auswertung des Skriptes und der Klassifikation durch die Cellminer Software in Abbildung 8.8 zeigt deutliche Gemeinsamkeiten bei der Auswertung. Insbesondere bei den Kontrollplatten Nr. 2, 22, 23, 29, 30 lassen sich dieselben Belegungen erkennen. In der Auswertung der Cellminer-Softare sind die in der Vorverarbeitung aussortierten Bilder in grauer Farbe gekennzeichnet.



(a) Plattenergebnisse Skript



(b) Plattenergebnisse Cellminer

Abbildung 8.8.: Auswertung des Zell-Assays durch Skript und durch Cellminer Software.

9. Fazit

In dieser Arbeit haben wir uns mit dem Problem beschäftigt, eine große Anzahl von Daten unter der Randbedingung zu klassifizieren, dass zu Beginn nur wenige oder keine klassifizierten Trainingsmuster verfügbar sind.

Das Paradigma des Aktiven Lernens im Bereich des maschinellen Lernens setzt sich mit dieser Thematik auseinander. In relativ kurzer Zeit wurde in diesem Gebiet eine Vielzahl verschiedener Strategien entwickelt, die meistens speziell für einen bestimmten Lernalgorithmus entworfen wurden. Im Bereich des Aktiven Lernens existieren verschiedene Vorgehensweisen: Ein Konzept beschäftigt sich mit der Schätzung des erwarteten Fehlers eines Lerners und dessen Minimierung. Ein weiterer Ansatz besteht darin, den Versionsraum in jeder Lerniteration möglichst stark zu reduzieren. Die Methode des „Uncertainty sampling“ versucht direkt an der Klassifikationsgrenze weitere Muster auszuwählen. Bislang existieren nur wenig Ansätze, die das Aktive Lernen mit einem Clustering verbinden.

Aktives Lernen ist ein Szenario, welches in vielen realen Anwendungsfällen nützlich ist. Insbesondere in industriellen Anwendungen, wo Zeit und Kosten eine wichtige Rolle spielen, werden solche Verfahren eingesetzt. In dieser Arbeit haben wir einen speziellen Anwendungsfall aus dem Bereich der Wirkstoffanalyse in der bildgestützten Arzneimittelforschung betrachtet. Hierbei standen einer großen Menge von unklassifizierten Messdaten ein menschlicher Experte für die Klassifikation gegenüber. In dieser Anwendung muss von Beginn an ein stabiles Modell zur Klassifikation der Zellbilder gelernt werden, ohne dass a priori Informationen über die mögliche Anzahl und Ausprägungen der Klassen verfügbar sind. Die bisher entwickelten Verfahren im Bereich des Aktiven Lernens ignorieren solche Anwendungsfälle und betrachten den Anwendungsfall meist erst ab dem Zeitpunkt, zu dem bereits ein initiales Modell trainiert worden ist.

Um den Prozess der Datenexploration besser in den aktiven Lernprozess zu integrieren, wurde in einem ersten Ansatz ein Verfahren entwickelt, welches den Datensatz durch ein initiales Clustering mit Erkennung von Rauschen kompakt zusammenfasst. Eine initiale

Klassifikation wird dadurch gewonnen, dass jeder Cluster in seine Subcluster unterteilt wird und nach der Klasse der Cluster-Zentren gefragt wird. Sind die Klassen innerhalb eines Clusters homogen, stoppt der Prozess im aktuellen Cluster, ansonsten wird er bis zu einer vorgegebenen maximalen Rekursionstiefe fortgeführt. Anhand der Fuzzy-Zugehörigkeitsgrade zu den Prototypen wird jeder Datenpunkt klassifiziert. Basierend auf dieser initialen Klassifikation wird dann versucht, das durch die Prototypen erhaltene Konzept noch näher an die Semantik des Orakels anzupassen. Hierfür werden die Prototypen mit Hilfe des LVQ-Verfahrens mit wenigen klassifizierten Mustern trainiert. Die hier vorgestellte Strategie, die möglichst sinnvolle Muster aus der Menge der nicht klassifizierten Daten auswählt, ermöglicht eine schnellere Adaption der Prototypen als sie z. B. bei einer Zufallsauswahl möglich wäre.

Der zweite Ansatz besteht aus einem Prototypen basierten aktiven Klassifikationsschema, in dem in jeder Lerniteration ein neuer Prototyp kreiert wird, um die Klassifikation zu verfeinern. Basierend auf dieser wachsenden Ansammlung von Prototypen werden für jedes Muster die Klassenwahrscheinlichkeiten bestimmt, um somit die Muster mit hoher Klassifikationsunsicherheit zu identifizieren. Kombiniert mit einer lokalen Schätzung der Dichte wurde ein neues Selektionskriterium entwickelt. Durch die Reduktion der Dichte-Potentiale ergibt sich ein Übergang von einer initialen Exploration hin zu einer Spezialisierung und Verfeinerung der Klassengrenzen. Dadurch werden zunächst besonders repräsentative Muster ausgewählt. Erst wenn der Datensatz ausreichend exploriert worden ist, konzentriert sich die Selektion auf Muster an den Klassengrenzen.

Die wesentlichen Vorteile dieser beiden vorgestellten Verfahren gegenüber den anderen Verfahren im Bereich des Aktiven Lernens sind:

- Es wird vom ersten Muster an ein Modell gelernt. Dies wirkt sich positiv auf die Stabilität und die Performanz in den ersten Lerniterationen aus und ermöglicht eine schnelle und stabile Klassifikation mit nur wenigen Mustern.
- Die Performanz des PBAC Algorithmus garantiert im Durchschnitt eine höhere Stabilität als die anderen Verfahren.
- Es ist nicht erforderlich, dass die Anzahl der möglichen Klassen im Datensatz zu Beginn bekannt sein muss.
- Die Klassifikation beschränkt sich nicht auf 2-Klassen Probleme.

Im Bereich des Aktiven Lernens wird der Ansatz mit einer aktiven SVM sehr häufig angewendet. Wie man im Ergebnis-Kapitel sehen konnte, ist die Performanz sehr gut. Es sprechen jedoch einige Gründe dafür, einen Prototypen basierten Ansatz vorzuziehen:

Der wichtigste Grund ist, dass die Performanz der SVM sehr stark vom verwendeten Kernel abhängt. In den in dieser Arbeit durchgeführten Experimenten wurde jeweils der beste Kernel speziell für jeden Datensatz ausgewählt, die Performanz mit anderen Kernel(parametern) war zum Teil deutlich schlechter. Das Auffinden eines guten Kernels erfordert eine Vielzahl von klassifizierten Mustern, die in dieser Problemstellung eigentlich nicht zur Verfügung stehen. Als binärer Klassifikator mit Komplexität $O(n^2)$ benötigt die SVM weitaus mehr Zeit zum Trainieren, was in einer zeitkritischen interaktiven Anwendung von Nachteil ist. Weiterhin lässt sich ein Modell von Support-Vektoren im Modell-Parameter Raum für einen Benutzer nur schwer visualisieren, ein Prototypen-Modell ist für den Benutzer einfacher zu interpretieren.

Durch das globale Clustering hat das ALVQ-Verfahren gegenüber der lokal begrenzten Dichteschätzung im PBAC-Verfahren einen leichten Vorteil. Da jedoch die bestehende Cluster-Hypothese durch Sub-Clustering mit mehreren Mustern verifiziert werden muss, kann dies bei homogener Klassenverteilung im Subcluster zu „unnötigen“ Abfragen führen. Generell unterscheiden sich das ALVQ und das PBAC-Verfahren vor allem dadurch, dass die Phasen der Exploration und der Spezialisierung auf die Klassengrenzen im ALVQ-Verfahren klar voneinander getrennt sind, wohingegen der Übergang im PBAC-Verfahren weicher formuliert ist. Die feine Modellierung der Klassengrenzen über mehrere Prototypen im PBAC-Verfahren ermöglicht in nachfolgenden Iterationen eine bessere Klassifikation als die Verschiebung der Prototypen im ALVQ-Verfahren. Weiterhin ist das PBAC-Verfahren weniger anfällig für Rauschen in den Daten, da über die gewichtete k-nächste Nachbarn Klassifikationen falsch klassifizierte Muster ausgeglichen werden können.

Eine mögliche Erweiterung des PBAC-Verfahrens besteht darin, statt fest vorgegebener Parameterwerte, an das Klassifikationsproblem angepasste Parameterwerte zu verwenden. Die Performanz des Verfahrens ist abhängig von den Parametern r_a für die Schätzung der Potentiale, ϵ für die Gewichtung der Klassifikationsunsicherheit und k für die Anzahl der Nachbarn, die die Klassifikationsentscheidung beeinflussen. Auch wenn der Radius r_a das Ergebnis beim subtractive clustering beeinflusst, existieren nach Wissen des Autors keine Verfahren, um einen optimalen Radius aus den Daten zu schätzen.

In manchen Fällen kann es sinnvoll sein den Aspekt der Verfeinerung der Klassengrenzen mit dem Parameter ϵ zu betonen, bevor dies automatisch durch die Reduktion der Potentiale erfolgt. Eine Möglichkeit besteht z. B. darin, den erwarteten Fehler in jeder Lerniteration zu schätzen. Verändert sich der Fehler in aufeinander folgenden Lernite-

rationen nicht mehr stark, so könnte man ϵ erhöhen. Ein Effekt dieser Maßnahme wäre dann wahrscheinlich in der Mitte zwischen Explorationsphase und späteren Iterationen zu beobachten.

Verschiedene Arbeiten beschäftigen sich mit der Schätzung eines optimalen Wertes für k bei der k-nächsten Nachbarn Klassifikation. Die meisten Verfahren verwenden hierfür klassifizierte Testdaten und sind damit nicht für das Aktive Lernen geeignet. Ein interessanter Ansatz findet sich in der Arbeit von (Voulgaris und Magoulas, 2008). Hier wird versucht, mit verschiedenen Werten von k die resultierende Klassifikationssicherheit für ein Muster zu maximieren. Anschließend wird das harmonische Mittel der Distanzen der Nachbarn einer Klasse berechnet, und die Klasse mit dem kleinsten Wert zugewiesen. Auch in dieser Arbeit werden klassifizierte Testdaten verwendet. Es wäre evtl. denkbar, dieses Verfahren einzusetzen, nachdem eine gewisse Anzahl von Prototypen im PBAC-Verfahren generiert worden ist (beispielsweise nach der Phase der Exploration, wenn die Potentialsumme einen gegebenen Schwellwert unterschritten hat).

Eine im Hinblick auf die Cellminer-Applikation nützliche Erweiterung des PBAC-Verfahrens besteht darin, dieses Verfahren in ein Meta-Verfahren für mehrere verschiedene Beschreibungsräume (wie z. B. in der Arbeit von (Muslea u. a., 2006)) einzubinden. Bisher werden die relevanten Attribute aufgrund einer groben Vorklassifikation auf der Referenzplatte geschätzt. Durch den Einsatz mehrerer Beschreibungsräume könnten relevante numerische Beschreibungen der Zellbilder herausgefiltert werden und somit die für den Biologen relevante Beschreibung verwendet werden. Um die Qualität eines Beschreibungsraumes zu schätzen, könnten die mittleren Abstände der Prototypen zu Prototypen der anderen Klasse verwendet werden. Auch Cluster-Qualitätsmerkmale könnten ein Indikator für eine gute Qualität des Beschreibungsraumes sein.

Literaturverzeichnis

[Abe und Mamitsuka 1998] Abe, Naoki ; Mamitsuka, Hiroshi: Query Learning Strategies Using Boosting and Bagging. In: *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1998, S. 1–9. – ISBN 1-55860-556-8

[Angluin 1988] Angluin, D.: Queries and concept learning. In: *Machine Learning* 2 (1988), Nr. 3, S. 319–342

[Asuncion und Newman 2007] Asuncion, A. ; Newman, D.J.: *UCI Machine Learning Repository*. 2007. – URL `http://mlearn.ics.uci.edu/MLRepository.html`

[Auer u. a. 2003] Auer, Peter ; Cesa-Bianchi, Nicolò ; Freund, Yoav ; Schapire, Robert E.: The Nonstochastic Multiarmed Bandit Problem. In: *SIAM J. Comput.* 32 (2003), Nr. 1, S. 48–77. – ISSN 0097-5397

[Baram u. a. 2004] Baram, Yoram ; El-Yaniv, Ran ; Luz, Kobi: Online Choice of Active Learning Algorithms. In: *J. Mach. Learn. Res.* 5 (2004), S. 255–291. – ISSN 1533-7928

[Basu u. a. 2004] Basu, Sugato ; Banerjee, Arindam ; Mooney, Raymond J.: Active Semi-Supervision for Pairwise Constrained Clustering. In: Berry, Michael W. (Hrsg.) ; Dayal, Umeshwar (Hrsg.) ; Kamath, Chandrika (Hrsg.) ; Skillicorn, David B. (Hrsg.): *SDM*, SIAM, 2004. – ISBN 0-89871-568-7

[Bellman 1961] Bellman, Richard: *Adaptive COntrol Processes: A Guided Tour*. Princeton : Princeton University Press, 1961

[Bentley 1975] Bentley, Jon L.: Multidimensional binary search trees used for associative searching. In: *Commun. ACM* 18 (1975), Nr. 9, S. 509–517. – ISSN 0001-0782

[Bezdek 1981] Bezdek, James C.: *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA : Kluwer Academic Publishers, 1981. – ISBN 0306406713

[Breiman 1996] Breiman, Leo: Bagging predictors. In: *Mach. Learn.* 24 (1996), Nr. 2, S. 123–140. – ISSN 0885-6125

[Burges 1998] Burges, Christopher J. C.: A Tutorial on Support Vector Machines for Pattern Recognition. In: *Data Mining and Knowledge Discovery* 2 (1998), Nr. 2, S. 121–167

[Campbell u. a. 2000] Campbell, Colin ; Cristianini, Nello ; Smola, Alex J.: Query Learning with Large Margin Classifiers. In: *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2000, S. 111–118. – ISBN 1-55860-707-2

[Cheng und Shih 2007] Cheng, Shouxian ; Shih, Frank Y.: An improved incremental training algorithm for support vector machines using active query. In: *Pattern Recogn.* 40 (2007), Nr. 3, S. 964–971. – ISSN 0031-3203

[Chin 1997] Chin, Stephen L.: An Efficient Method for Extracting Fuzzy Classification Rules from High Dimensional Data. In: *JACIII* 1 (1997), Nr. 1, S. 31–36

[Cios u. a. 1998] Cios, Krzysztof ; Pedrycz, Witold ; Swiniarski, Roman: *Data Mining: Methods for Knowledge Discovery*. Kluwer Academic Publishers, 1998

[Cohn u. a. 1994a] Cohn, D. A. ; Atlas, L. ; Ladner, R. E.: Improving Generalization with Active Learning. In: *Machine Learning* 15 (1994), Nr. 2, S. 201–221

[Cohn u. a. 1994b] Cohn, David A. ; Ghahramani, Zoubin ; Jordan, Michael I.: Active Learning with Statistical Models. In: Tesauro, Gerald (Hrsg.) ; Touretzky, David S. (Hrsg.) ; Leen, Todd K. (Hrsg.): *NIPS*, MIT Press, 1994, S. 705–712

[Dagan und Engelson 1995] Dagan, Ido ; Engelson, Sean P.: Committee-Based Sampling For Training Probabilistic Classifiers. In: *ICML*, 1995, S. 150–157

[Dasgupta 2004] Dasgupta, S.: Analysis of a greedy active learning strategy. In: *NIPS*, 2004

[Dave 1991] Dave, Rajesh N.: Characterization and detection of noise in clustering. In: *Pattern Recogn. Lett.* 12 (1991), Nr. 11, S. 657–664

[Dönmez u. a. 2007] Dönmez, Meryem P. ; Carbonell, Jaime G. ; Bennett, Paul N.: Dual Strategy Active Learning. In: Kok, Joost N. (Hrsg.) ; Koronacki, Jacek (Hrsg.) ; Mántaras, Ramon L. de (Hrsg.) ; Matwin, Stan (Hrsg.) ; Mladenic, Dunja (Hrsg.) ; Skowron, Andrzej (Hrsg.): *ECML* Bd. 4701, Springer, 2007, S. 116–127. – ISBN 978-3-540-74957-8

[Döring u. a. 2005] Döring, Christian ; Borgelt, Christian ; Kruse, Rudolf: Effects of Irrelevant Attributes in Fuzzy Clustering. In: *Proc. 15th IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'05, Reno, NV), CD-ROM.* Piscataway, NJ, USA : IEEE Press, 2005

[Duda u. a. 2000] Duda, Richard O. ; Hart, Peter E. ; Stork, David G.: *Pattern Classification (2nd Edition).* Wiley-Interscience, 2000. – ISBN 0471056693

[Freund u. a. 1997] Freund, Y. ; Seung, H. S. ; Shamir, E. ; Tishby, N.: Selective sampling using the query by committee algorithm. In: *Machine Learning* 28 (1997), Nr. 2-3, S. 133–168

[Freund und Schapire 1995] Freund, Yoav ; Schapire, Robert E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory.* London, UK : Springer-Verlag, 1995, S. 23–37. – ISBN 3-540-59119-2

[Freund u. a. 1993] Freund, Yoav ; Seung, H. S. ; Shamir, Eli ; Tishby, Naftali: Information, Prediction, and Query by Committee. In: *Advances in Neural Information Processing Systems 5, [NIPS Conference].* San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1993, S. 483–490. – ISBN 1-55860-274-7

[Gabrys und Petrakieva 2004] Gabrys, B. ; Petrakieva, L.: Combining labelled and unlabelled data in the design of pattern classification systems. In: *Int. J. Approx. Reasoning* 35 (2004), Nr. 3, S. 251–273

[Geman u. a. 1992] Geman, Stuart ; Bienenstock, Elie ; Doursat, Rene: Neural networks and the bias/variance dilemma. In: *Neural Comput.* 4 (1992), Nr. 1, S. 1–58. – ISSN 0899-7667

[Grira u. a. 2005] Grira, Nizar ; Crucianu, Michel ; Boujemaa, Nozha: Semi-supervised image database categorization using pairwise constraints. In: *ICIP (3)*, 2005, S. 1228–1231

[Haralick 1973] Haralick, R. M.: Textural features for image classification. In: *IEEE Trans. System, Man and Cybernetics* 3 (1973), Nr. 6, S. 610–621

[Hochbaum und Shmoys 1985] Hochbaum ; Shmoys: A best possible heuristic for the k-center problem. In: *Mathematics of Operations Research* 10 (1985), Nr. 2, S. 180–184

[Hofmann und Buhmann 1997] Hofmann, Thomas ; Buhmann, Joachim M.: Active Data Clustering. In: Jordan, Michael I. (Hrsg.) ; Kearns, Michael J. (Hrsg.) ; Solla, Sara A. (Hrsg.): *NIPS*, The MIT Press, 1997. – ISBN 0-262-10076-2

[Jang u. a. 1997] Jang, J. S. R. ; Sun, C. T. ; Mizutani, E.: *Neuro-Fuzzy and soft Computing: a computational approach to learning and machine intelligence*. 1st. Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1997

[Johnson 1967] Johnson, Stephen: Hierarchical clustering schemes. In: *Psychometrika* 32 (1967), Nr. 3, S. 241–254

[Jones u. a. 2005] Jones, Thouis R. ; Carpenter, Anne ; Golland, Polina: Voronoi-Based Segmentation of Cells on Image Manifolds. In: Liu, Yanxi (Hrsg.) ; Jiang, Tianzi (Hrsg.) ; Zhang, Changshui (Hrsg.): *CVBIA* Bd. 3765, Springer, 2005, S. 535–543. – ISBN 3-540-29411-2

[Kang u. a. 2004] Kang, Jaeho ; Ryu, Kwang R. ; Kwon, Hyuk-Chul: Using Cluster-Based Sampling to Select Initial Training Set for Active Learning in Text Classification. In: *Advances in Knowledge Discovery and Data Mining* Bd. 3056, Springer Berlin / Heidelberg, 2004, S. 384–388

[Kirkpatrick u. a. 1983] Kirkpatrick, Scott ; Jr., D. G. ; Vecchi, Mario P.: Optimization by Simmulated Annealing. In: *Science* 220 (1983), Nr. 4598, S. 671–680

[Kohonen 2001] Kohonen, Teuvo ; Kohonen, T. (Hrsg.) ; Schroeder, M. R. (Hrsg.) ; Huang, T. S. (Hrsg.): *Self-Organizing Maps*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2001. – ISBN 3540679219

[Lang und Baum 1992] Lang, K.J. ; Baum, E.B.: Query Learning Can Work Poorly When a Human Oracle Is Used. In: *Proceedings of the International Joint Conference on Neural Networks*, 1992, S. 335–340

[Langley 2000] Langley, P. (Hrsg.): *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Standord, CA, USA, June 29 - July 2, 2000*. Morgan Kaufmann, 2000. – ISBN 1-55860-707-2

[Langley u. a. 1992] Langley, Pat ; Iba, Wayne ; Thompson, Kevin: An Analysis of Bayesian Classifiers. In: *National Conference on Artificial Intelligence*, 1992, S. 223–228

[Lewis und Gale 1994] Lewis, David D. ; Gale, William A.: A Sequential Algorithm for Training Text Classifiers. In: Croft, W. B. (Hrsg.) ; Rijsbergen, C. J. van (Hrsg.): *SIGIR*, ACM/Springer, 1994, S. 3–12. – ISBN 3-540-19889-X

[Liere und Tadepalli 1997] Liere, Ray ; Tadepalli, Prasad: Active Learning with Committees for Text Categorization. In: *AAAI/IAAI*, 1997, S. 591–596

[Lindenbaum u. a. 2004] Lindenbaum, Michael ; Markovitch, Shaul ; Rusakov, Dmitry: Selective Sampling for Nearest Neighbor Classifiers. In: *Machine Learning* 54 (2004), Nr. 2, S. 125–152

[Liu und Motoda 1998] Liu, Huan ; Motoda, Hiroshi: *Feature Selection for Knowledge Discovery and Data Mining*. Norwell, MA, USA : Kluwer Academic Publishers, 1998. – ISBN 079238198X

[Luo u. a. 2005] Luo, T. ; Kramer, K. ; Goldgof, D. B. ; Hall, L. O. ; Samson, S. ; Remsen, A. ; Hopkins, T.: Active Learning to Recognize Multiple Types of Plankton. In: *Journal of Machine Learning Research* 6 (2005), S. 589–613

[MacKay 1992] MacKay, David J. C.: Information-based objective functions for active data selection. In: *Neural Comput.* 4 (1992), Nr. 4, S. 590–604. – ISSN 0899-7667

[MacQueen 1967] MacQueen, J. B.: Some Methods for Classification and Analysis of MultiVariate Observations. In: Cam, L. M. L. (Hrsg.) ; Neyman, J. (Hrsg.): *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* Bd. 1, University of California Press, 1967, S. 281–297

[Mandel u. a. 2006] Mandel, Michael I. ; Poliner, Graham E. ; Ellis, Daniel P. W.: Support vector machine active learning for music retrieval. In: *Multimedia Syst.* 12 (2006), Nr. 1, S. 3–13

[Manning u. a. 2008] Manning, Christopher D. ; Raghavan, Prabhakar ; Schütze, Hinrich: *Introduction to Information Retrieval*. Cambridge University Press, 2008

[McCallum und Nigam 1998] McCallum, Andrew ; Nigam, Kamal: Employing EM and Pool-Based Active Learning for Text Classification. In: Shavlik, Jude W. (Hrsg.): *ICML*, Morgan Kaufmann, 1998, S. 350–358. – ISBN 1-55860-556-8

[Melville und Mooney 2004] Melville, Prem ; Mooney, Raymond J.: Diverse ensembles for active learning. In: *ICML '04: Proceedings of the twenty-first international conference on Machine learning*. New York, NY, USA : ACM, 2004, S. 74. – ISBN 1-58113-828-5

[Melville und Mooney 2005] Melville, Prem ; Mooney, Raymond J.: Creating diversity in ensembles using artificial data. In: *Information Fusion* 6 (2005), Nr. 1, S. 99–111

[Menard 2002] Menard, S.: *Applied logistic regression analysis (2nd Ed.)*. Sage University Papers, 2002 (106)

[Mitchell und Blum 1994] Mitchell, Tom ; Blum, Avrim: *A Neural Network Face Recognition Assignment*. 1994. – URL `http://www.cs.cmu.edu/afs/cs.cmu.edu/user/avrim/www/ML94/face_homework.html`

[Mitchell 1982] Mitchell, Tom M.: Generalization as Search. In: *Artif. Intell.* 18 (1982), Nr. 2, S. 203–226

[Mitchell 1997] Mitchell, Tom M.: *Machine Learning*. New York : McGraw-Hill, 1997

[Muslea u. a. 2006] Muslea, Ion ; Minton, Steven ; Knoblock, Craig A.: Active Learning with Multiple Views. In: *J. Artif. Intell. Res. (JAIR)* 27 (2006), S. 203–233

[Nguyen und Smeulders 2004] Nguyen, Hieu T. ; Smeulders, Arnold W. M.: Active learning using pre-clustering. In: Brodley, Carla E. (Hrsg.): *ICML*, ACM, 2004

[Osugi u. a. 2005] Osugi, Thomas ; Kun, Deng ; Scott, Stephen: Balancing Exploration and Exploitation: A New Algorithm for Active Machine Learning. In: *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*. Washington, DC, USA : IEEE Computer Society, 2005, S. 330–337. – ISBN 0-7695-2278-5

[Otsu 1978] Otsu, N.: A Threshold Selection Method from Gray-Level Histogram. In: *IEEE Trans. Systems, Man, and Cybernetics* 8 (1978), S. 62–66

[Parzen 1962] Parzen, E.: On the estimation of a probability density function and the mode. In: *Annals of Mathematical Statistics* 33 (1962), S. 1065–1076

[Quinlan 1993] Quinlan, J. R.: *C4.5: programs for machine learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1993. – ISBN 1-55860-238-0

[Roy und McCallum 2001] Roy, N. ; McCallum, A.: Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In: Brodley, C. E. (Hrsg.) ; Danyluk, A. P. (Hrsg.): *ICML*, Morgan Kaufmann, 2001, S. 441–448. – ISBN 1-55860-778-1

[Schohn und Cohn 2000] Schohn, G. ; Cohn, D.: Less is More: Active Learning with Support Vector Machines. In: (Langley, 2000), S. 839–846. – ISBN 1-55860-707-2

[Schwachulla 1998] Schwachulla, Wolfram (Hrsg.): *Der Brockhaus*. Leipzig, Deutschland : F.A. Brockhaus GmbH, 1998. – ISBN 3-7653-1678-4

[Seung u. a. 1992] Seung, H. S. ; Opper, M. ; Sompolinsky, H.: Query by committee. In: *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*. New York, NY, USA : ACM, 1992, S. 287–294. – ISBN 0-89791-497-X

[Sewell 2008] Sewell, Martin: *No Free Lunch Theorems*. 2008. – URL `http://www.no-free-lunch.org`

[Shannon 2001] Shannon, C. E.: A mathematical theory of communication. In: *SIGMOBILE Mob. Comput. Commun. Rev.* 5 (2001), Nr. 1, S. 3–55. – ISSN 1559-1662

[Struyf u. a. 1997] Struyf, Anja ; Hubert, Mia ; Rousseeuw, Peter J.: Integrating robust clustering techniques in S-PLUS. In: *Comput. Stat. Data Anal.* 26 (1997), Nr. 1, S. 17–37. – ISSN 0167-9473

[Teague 1980] Teague, M. R.: Image analysis via the general theory of moments. In: *Journal of the Optical Society of America* 70 (1980), 8, S. 920–930

[Tong und Koller 2001] Tong, S. ; Koller, D.: Support vector machine active learning with applications to text classification. In: *Journal of Machine Learning Research* 2 (2001), November, S. 45–66. – Full version of paper in ICML 2000

[Tong und Chang 2001] Tong, Simon ; Chang, Edward: Support vector machine active learning for image retrieval. In: *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*. New York, NY, USA : ACM, 2001, S. 107–118. – ISBN 1-58113-394-4

[Voulgaris und Magoulas 2008] Voulgaris, Z. ; Magoulas, G.D.: Extensions of the k Nearest Neighbour Methods for Classification Problems. In: *Proc. of the 26th IASTED International Conference on Artificial Intelligence and Applications (AIA), Innsbruck, Austria, February 11-13*, 2008, S. 23–28

[Wang u. a. 2003] Wang, L. ; Chan, K. L. ; Zhang, Z.: Bootstrapping SVM Active Learning by Incorporating Unlabelled Images for Image Retrieval. In: *CVPR (1)*, IEEE Computer Society, 2003, S. 629–634. – ISBN 0-7695-1900-8

[Warmuth u. a. 2003] Warmuth, M. K. ; Liao, J. ; Rätsch, G. ; Mathieson, M. ; Putta, S. ; Lemmen, C.: Active Learning with Support Vector Machines in the Drug Discovery Process. In: *Journal of Chemical Information and Computer Sciences* 43 (2003), Nr. 2, S. 667–673

[Windham 1981] Windham, M.P.: Cluster Validity for Fuzzy Clustering Algorithms. In: *Fuzzy Sets and Systems* 5 (1981), S. 177–185

[Wolpert und Macready 1997] Wolpert, David H. ; Macready, William G.: No Free Lunch Theorems for Optimization. In: *IEEE Transactions on Evolutionary Computation* 1 (1997), April, Nr. 1, S. 67–82

[Wong und Hajek 1985] Wong, E. ; Hajek, B.: *Stochastic Processes in Engineering Systems*. New York, U.S.A. : Springer, 1985

[Xu u. a. 2004] Xu, Zhao ; Yu, Kai ; Tresp, Volker ; Xu, Xiaowei ; Wang, Jizhi: Representative Sampling for Text Classification Using Support Vector Machines. In: *ECIR 2003* Bd. 2633, Springer Berlin / Heidelberg, 2004, S. 393–407

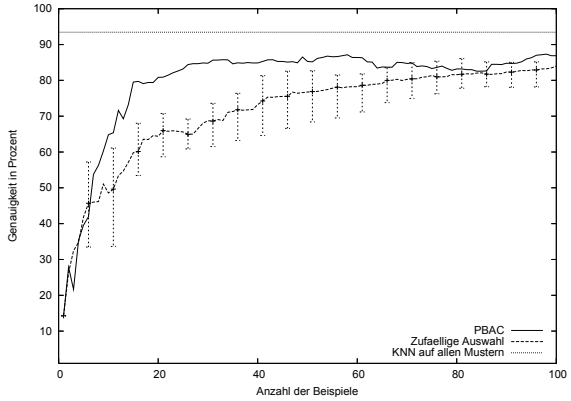
[Yang und Pedersen 1997] Yang, Yiming ; Pedersen, Jan O.: A Comparative Study on Feature Selection in Text Categorization. In: *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1997, S. 412–420. – ISBN 1-55860-486-3

[Zernike 1934] Zernike, F.: Beugungstheorie des Schneideverfahrens und seiner Verbesserten Form der Phasenkontrastmethode. In: *Physica* 1 (1934), S. 689–704

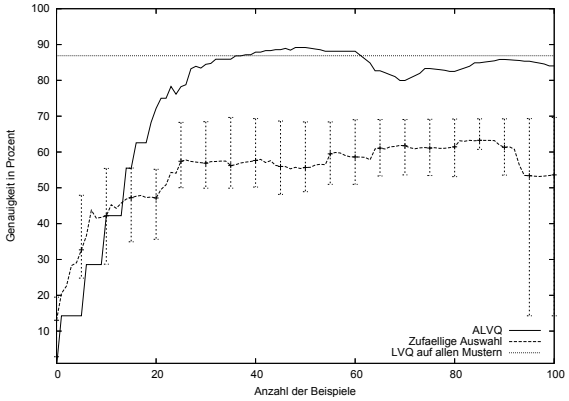
[Zhang und Oles 2000] Zhang, Tong ; Oles, Frank J.: A probability analysis on the value of unlabeled data for classification problems. In: (Langley, 2000), S. 1191–1198. – ISBN 1-55860-707-2

A. Erweiterte Experimente

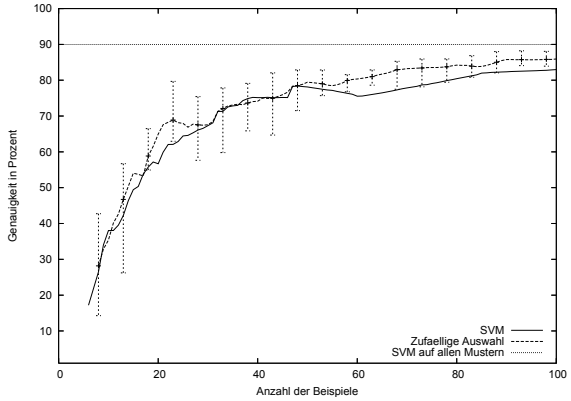
Segment Daten



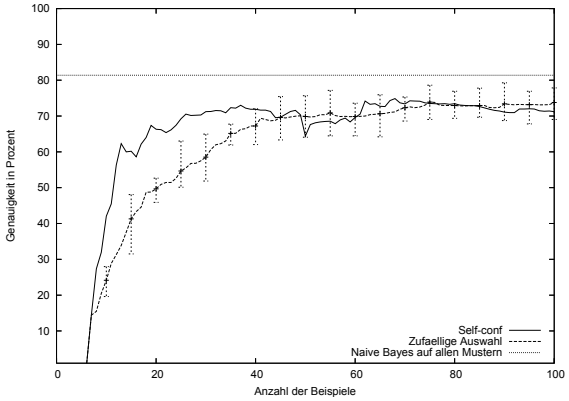
(a) PBAC



(b) ALVQ



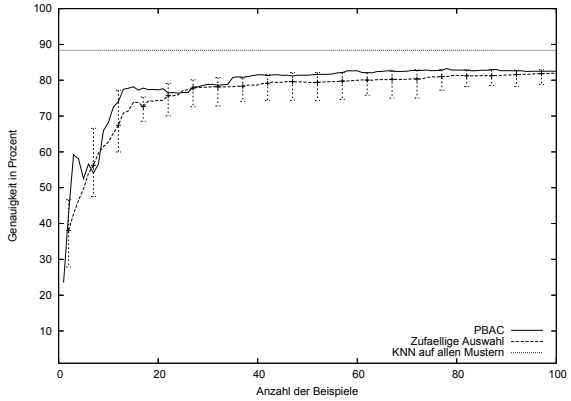
(c) Aktive SVM



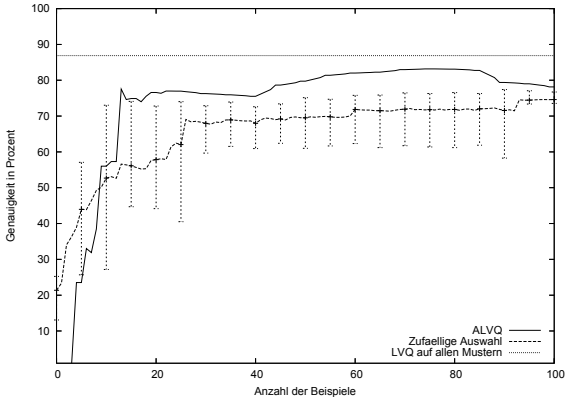
(d) Self-conf

Segment Daten.

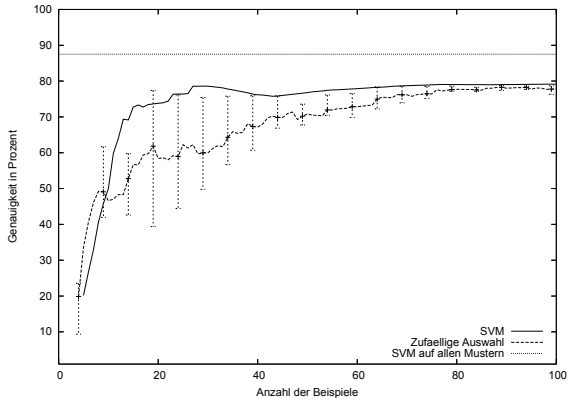
Satimage Daten



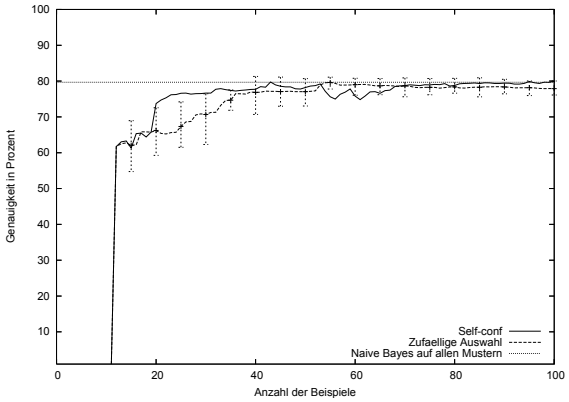
(a) PBAC



(b) ALVQ



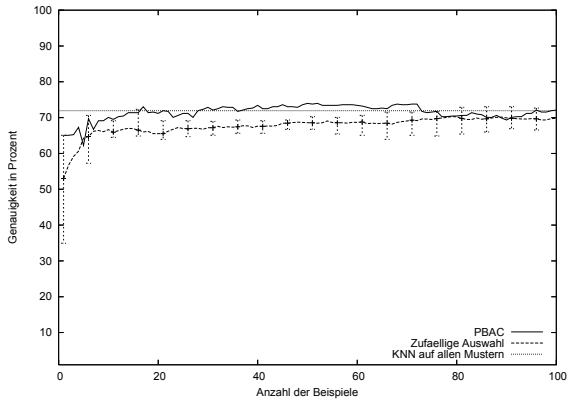
(c) Aktive SVM



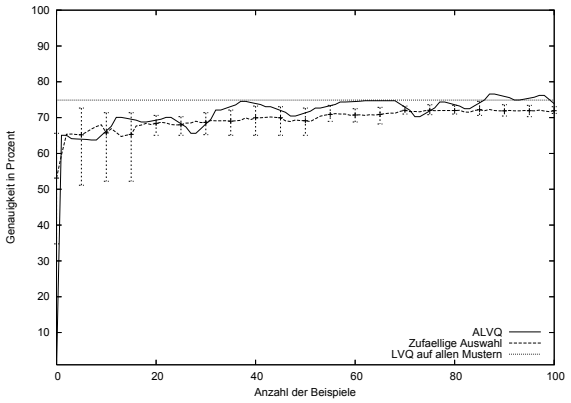
(d) Self-conf

Satimage Daten.

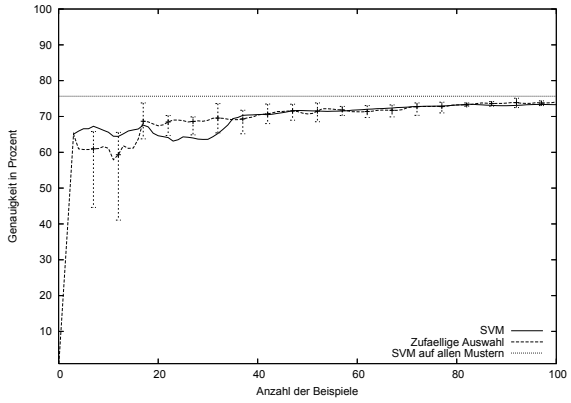
Diabetes Daten



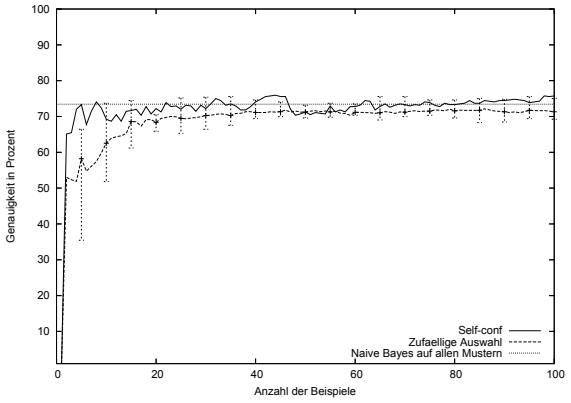
(a) PBAC



(b) ALVQ



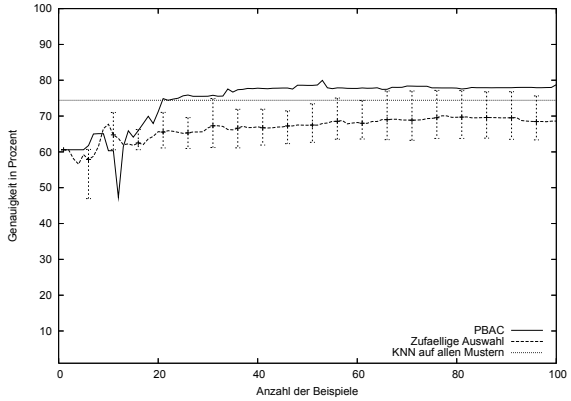
(c) Aktive SVM



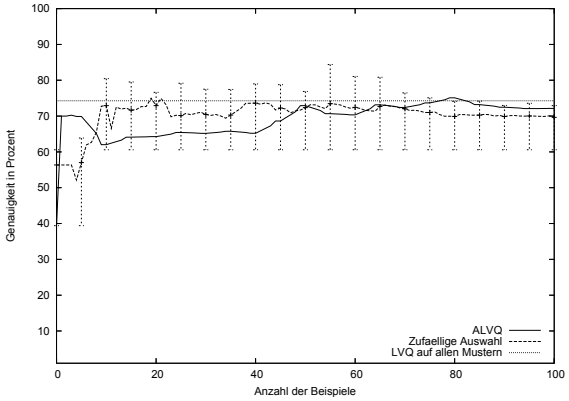
(d) Self-conf

Diabetes Daten.

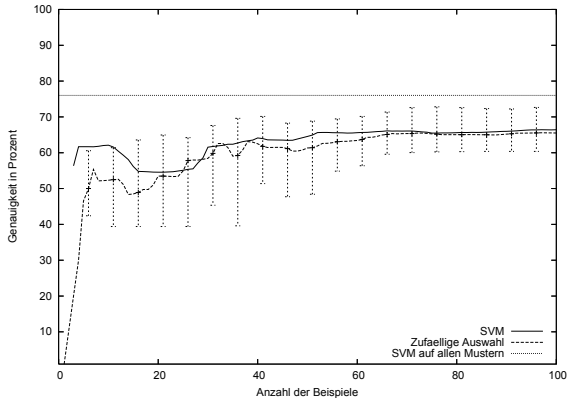
Spam Daten



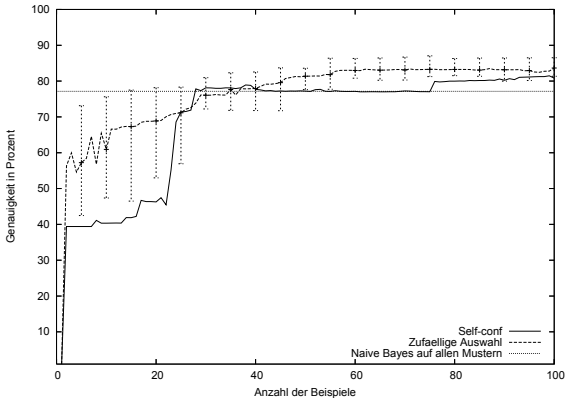
(a) PBAC



(b) ALVQ



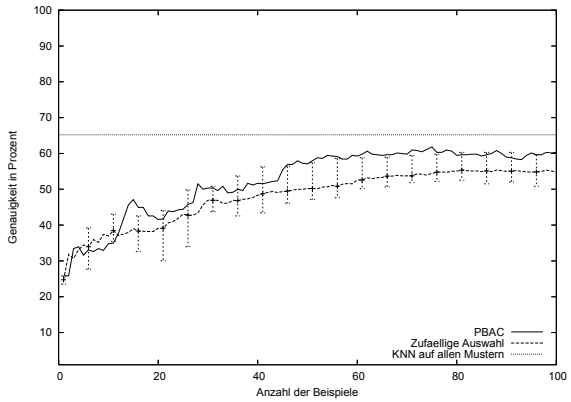
(c) Aktive SVM



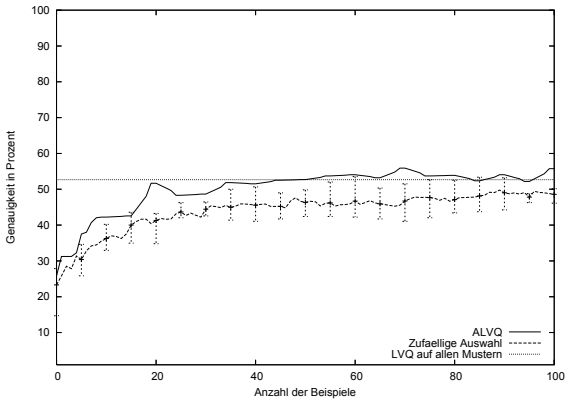
(d) Self-conf

Spam Daten.

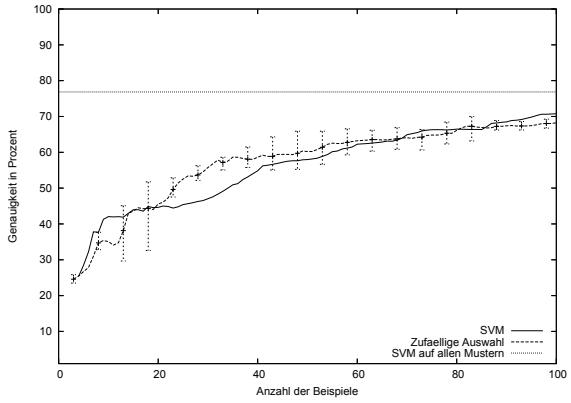
Vehicle Daten



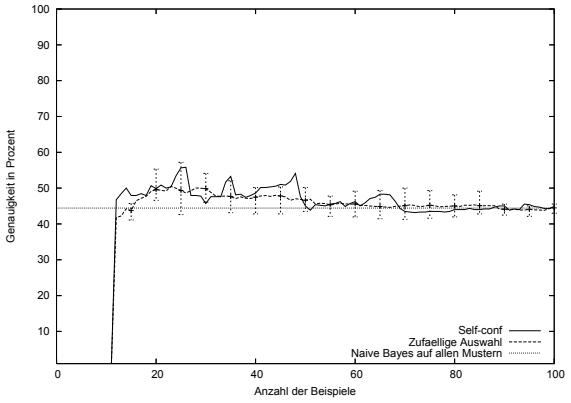
(a) PBAC



(b) ALVQ



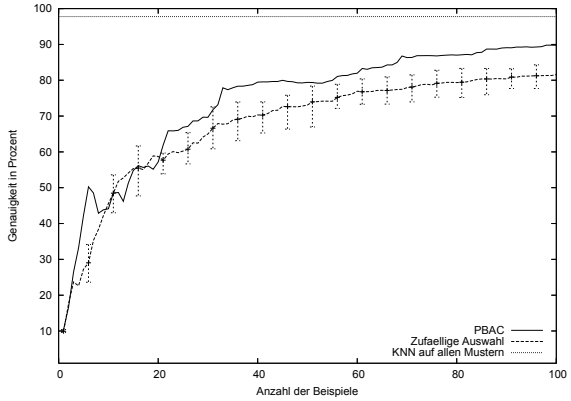
(c) Aktive SVM



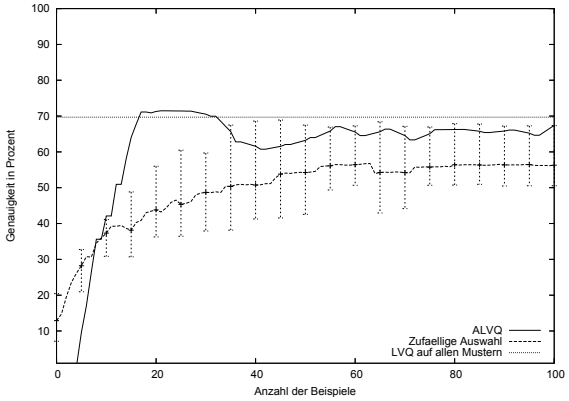
(d) Self-conf

Vehicle Daten.

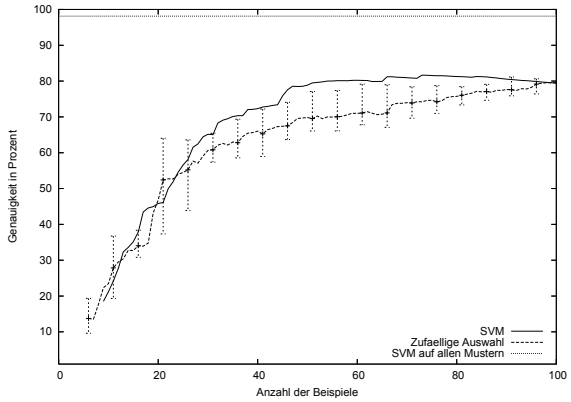
Pendigits Daten



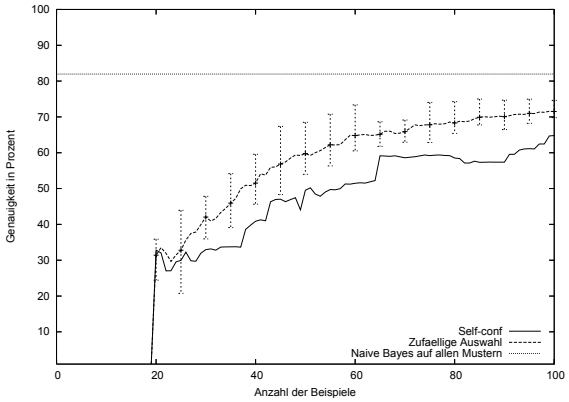
(a) PBAC



(b) ALVQ



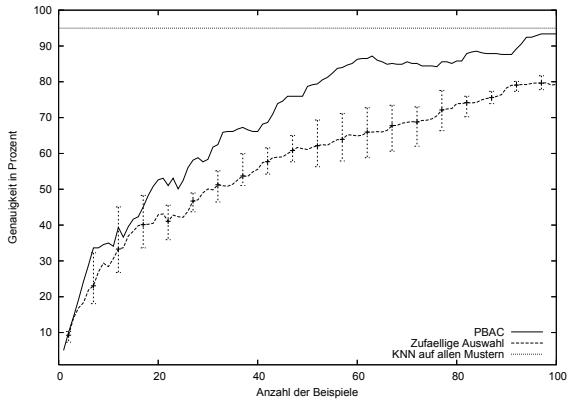
(c) Aktive SVM



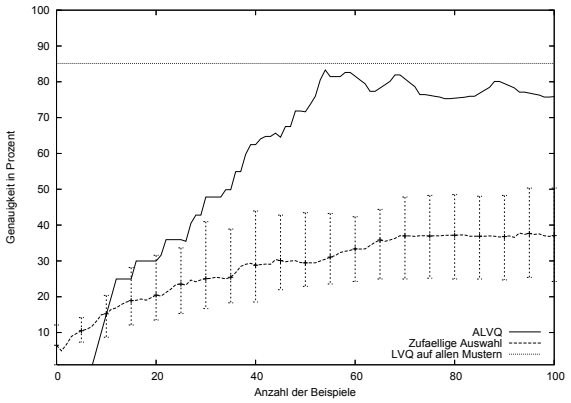
(d) Self-conf

Pendigits Daten.

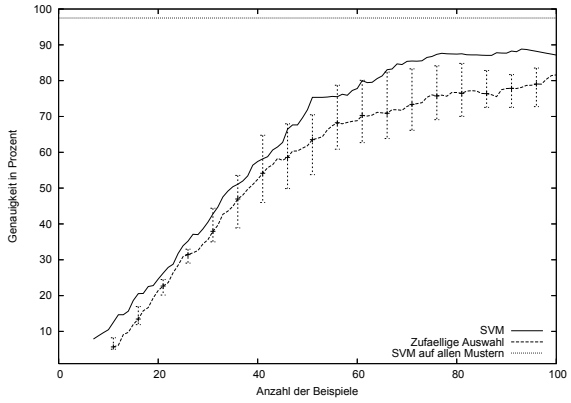
Faces Daten



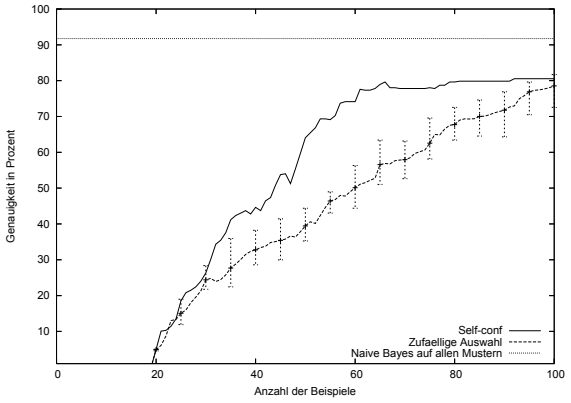
(a) PBAC



(b) ALVQ



(c) Aktive SVM



(d) Self-conf

Faces Daten.