

A Multi-dimensional Measure Function for Classifier Performance

Adrian Schröter
schroeter@st.cs.uni-sb.de

Lehrveranstaltung: AI Tools

Ort: Universität des Saarlandes

Dozenten: Michael Kipp
Alassane Ndiaye
Dominik Heckmann
Michael Feld

Zusammenfassung

Heutzutage verwenden wir Cross Validation, um das beste Modell für unsere Problemstellung zu finden. Leider wird bei Cross Validation einige wichtige Aspekte, wie die Komplexität eines Modells. Damit wir nun alle wichtigen Aspekte mit in unsere Evaluation einfließen lassen können, entwickelten Niklas Lavesson und Paul Davidsson „Measure Functions“. Diese Funktionen beziehen die wichtigsten drei Aspekte, Ähnlichkeit, Komplexität und Genauigkeit ein. Hiermit sind wir nun in der Lage, eine genauere Evaluation zu erstellen, um somit das für uns beste Modell zu finden.

Inhaltsverzeichnis

1	Einleitung	4
2	Grundlagen	4
2.1	Machine Learning	5
2.2	Weitere Begriffe	5
3	Modelle	6
3.1	k-Nearest-Neighbor	6
3.1.1	Idee	6
3.1.2	Formale Beschreibung	7
3.1.3	Parameter	7
3.2	Entscheidungsbaum	8
3.2.1	Idee	8
3.2.2	Formale Beschreibung	8
3.2.3	Parameter	9
3.3	Support Vector Machine	9
3.3.1	Idee	9
3.3.2	Formale Beschreibung	10
3.3.3	Parameter	11
4	Cross Validation	12
4.1	Idee	12
4.2	Vorteile	12
4.3	Nachteile	13
4.4	Ausprägungen	13
5	Evaluations Aspekte	13
5.1	Sub-Set-Fit	13
5.2	Similarity	14
5.3	Simplicity	14
6	Measure Function	16
6.1	Interpretation	16
6.2	Vorteile	16
6.3	Nachteile	16
7	Fazit	16

1 Einleitung

Wenn wir versuchen, mittels einer Datenmenge ein Vorhersagemodel zu erstellen, stellen sich uns zunächst folgende Fragen:

1. Welches Model ist für unsere Daten am besten geeignet?
2. Was ist die best mögliche Parametrisierung für dieses Model?
3. Wie evaluieren wir unser Model?

Wenn man zunächst diese drei Fragen betrachtet kommt man oftmals zu dem falschen Schluss, dass die Antworten häufig zu den ersten beiden Fragen am aufwändigsten und wichtigsten sind. Aber aufgrund von vorhandenen Automatisierungsmöglichkeiten wird die Suche nach den ersten beiden Antworten auf ein zeitliches Problem herunter gebrochen. Die Suche ist nun zusätzlich von einem weitem Faktor abhängig, nämlich von der Antwort auf die letzte Frage.

Da sich die Cross Validation als Standard zur Evaluierung durchgesetzt hat, entstand das Problem, dass verschiedene wichtige Entscheidungsaspekte ignoriert werden, und somit eine genaue Einschätzung der Modelle nicht immer möglich ist. Dies wirkt sich direkt auf die Suche nach dem optimalen Model für unsere Daten aus.

Somit ist eine gute Antwort auf die dritte Frage am wichtigsten. Denn von ihr hängt letztlich die Auswahl des finalen Models aus. Doch warum ist es die Frage, der in der Regel die geringste Bedeutung zu gute kommt?

Lavesson und Davidson [4] haben eine verbesserte Evaluierungsmethode vorgeschlagen, welche alle wichtigen Aspekte mit in die Bewertung einfließen lässt.

Im Hinblick auf diese Arbeit werden wir mit einigen Grundlagen anfangen (siehe Abschnitt 2). Anschließend widmen wir uns in Abschnitt 3 einigen Modellen, anhand derer die Bedeutung der Evaluierung veranschaulicht werden soll. Danach stellen wir die Cross Validationen (siehe Abschnitt 4) vor, gefolgt von den drei wichtigsten Entscheidungsaspekten (siehe Abschnitt 5). Zum Schluss stellen wir die Measure Function (siehe Abschnitt 6) von Lavesson und Davidson vor und schließen mit unserer Schlussfolgerung (siehe Abschnitt 7).

2 Grundlagen

In diesem Abschnitt führen wir zuerst den Begriff des *Machine Learning* ein, gefolgt von verschiedenen Grundbegriffen des *Machine Learning*.

2.1 Machine Learning

Wir bezeichnen mit *Machine Learning* den maschinellen Vorgang aus vorhandenen Daten Wissen zu generieren. In anderen Worten, es wird ein Vorhersagemodell erstellt, welches anhand des generierten Wissens Vorhersagen für verschiedene Ereignisse erstellt. Grundsätzlich unterscheiden wir zwischen zwei Arten der Vorhersage:

Klassifizierung. Hier werden Ereignisse anhand des generierten Wissens und deren Eigenschaften entsprechenden Klassen zugeteilt. Jedoch ist die Anzahl und Art der Klassen durch die anfangs vorhandenen Daten beschränkt.

Beispiel: Geht Alice morgen schwimmen?

Regression. In diesem Fall ist das generierte Wissen eine Funktion mit kontinuierlichem Wertebereich. Diese Funktion stellt eine Approximation der vorgegebenen Daten dar. Typischer Weise hat die Funktion den Typ: $\{\text{Eigenschaften}\} \rightarrow R$, in anderen Worten sie ordnet den Eigenschaften eines Ereignisses einen reellen Wert zu.

Beispiel: Wie lange geht Alice schwimmen?

2.2 Weitere Begriffe

Wir werden nun weitere Grundbegriffe, die wir benutzen werden, definieren. Um die Grundbegriffe zu verdeutlichen erläutern wir sie zusätzlich anhand eines Beispiels:

Unsere Freundin Alice geht gerne Schwimmen, jedoch ist es vom Wetter des jeweiligen Tages abhängig. Einfacher: Wenn der Tag sonnig ist, geht Alice in der Regel schwimmen.

Observation. Eine **Observation** beschreibt eine gemachte Beobachtung. Sie besteht aus der abhängigen Variable und verschiedenen Eigenschaften (Features).

Beispiel: An einem sonnigen Tag(Observation) geht Alice schwimmen.

Feature. **Features** stellen die Eigenschaften einer Observation dar.

Beispiel: An einem sonnigen(Feature) und windigen(Feature) Tag geht Alice schwimmen.

Abhängige Variable. Die **abhängige Variable** wird durch die Features einer Observation bestimmt. Sie kann kategorisch oder kontinuierlich sein.

Beispiel: Alice geht schwimmen(abhängige Variable), wenn es sonnig ist.

Model. Das statistische **Model** verarbeitet Observationen, deren abhängige Variable bekannt sind. Das hierbei erzeugte Wissen wird benutzt,

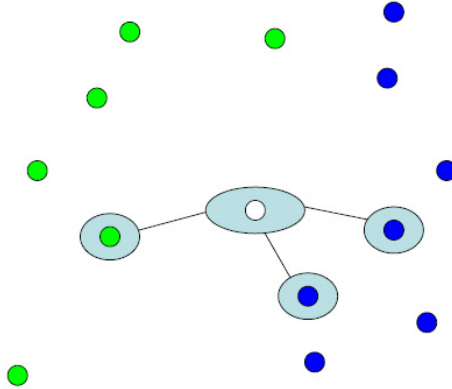


Abbildung 1: Im Fall von $k=3$ wird der neue Punkt in diesem Fall blau gefärbt, da zwei der drei nächsten Observationen(Punkte) blau sind.

um für neue Observationen, deren abhängige Variable unbekannt ist, Vorhersagen zu treffen.

Beispiel: Wir(Model) sagen anhand der vergangenen Tage vorher, ob Alice morgen schwimmen geht.

Training. Durch das **Training** wird das Modell anhand von Observationen erstellt und im Zuge dessen werden auch modellspezifische Parameter optimiert.

Beispiel: Bevor wir vorhersagen, ob Alice morgen schwimmen geht, sehen wir uns die vergangenen Tage an(Training).

3 Modelle

Bevor wir über Bewertungsarten zu diskutieren führen wir uns einige der wichtigsten statistischen Modelle vor Augen, um einen Eindruck von der vorhandenen Vielfalt zu erhalten. Am Anfang stellen wir intuitive Modelle vor und arbeiten uns zu weniger intuitiven vor.

3.1 k-Nearest-Neighbor

3.1.1 Idee

Stellen wir uns als Beispiel vor, dass wir eine Alice gerne schwimmt. Zusätzlich ist uns bekannt an welchen Tagen Alice schwimmen gegangen ist und an welchen nicht. Um vorherzusagen, ob Alice am nächsten Tag schwimmen geht oder nicht vergleichen wir den kommenden Tag mit den vergangenen Tagen und suchen den ähnlichsten Tag zum Kommenden. Unsere Vorhersage fällt nun so aus wie Alice's Entscheidung an diesem ähnlichsten Tag.

3.1.2 Formale Beschreibung

Mittels einer Abstandsfunktion werden die k nächsten Observationen(Punkte) gesucht. Anschließend werden bei einer Regression die abhängigen Variablen benutzt, um eine entsprechende Vorhersage zu erhalten, z.B. Berechnung Mittelwert oder Median über den k abhängigen Variablen. Bei einer Klassifizierung werden Votings (Wahlen) abgehalten, um die vorher zu sagende Klasse zu ermitteln. Zum Beispiel wählen wir die Klasse, die am häufigsten unter den k nächsten Observationen ist.

Wie wir in Abbildung 1 sehen, haben wir für $k=3$ entsprechend die drei nächsten Nachbarn bestimmt. Nun färben wir aufgrund eines einfachen Mehrheitsvotums den neuen Punkt blau.

3.1.3 Parameter

Das k -Nearest-Neighbor-Verfahren hat einige offensichtliche Parameter wie z.B k aber auch einige weniger offensichtliche. Diese hängen oft von der Art der Vorhersage ab, d.h. ob es eine Regression oder eine Klassifikation ist.

Allgemeine Parameter

k. k stellt die Anzahl der Nachbarn, die betrachtet werden, dar.

Abstand. Um die k nächsten Observationen (Punkte) zu bestimmen, muss man auf eine vorher definierte Abstandsfunktion zurückgreifen wie z.B. die Euklidische oder Manhattan Distanz.

Klassifikationspezifisch Parameter

Voting. Nachdem wir die k nächsten Observationen (Punkte) ermittelt haben, müssen wir nun die Klasse für den vorherzusagenden Punkt finden. Dies geschieht durch so genanntes Voting. Es gibt verschiedene Votingverfahren, wie z.B. einfaches Mehrheitsvotum oder gewichtetes Voting.

Regressionsspezifisch Parameter

Verrechnung. Nachdem wir die k nächsten Observationen (Punkte) ermittelt haben, müssen wir nun deren abhängige Variablen miteinander verrechnen. Hier kann man zum Beispiel den Median oder den Mittelwert der k nächsten Nachbarn heranziehen. Eine weitere Möglichkeit besteht auch darin, einen am Abstand gewichteten Mittelwert zu berechnen.

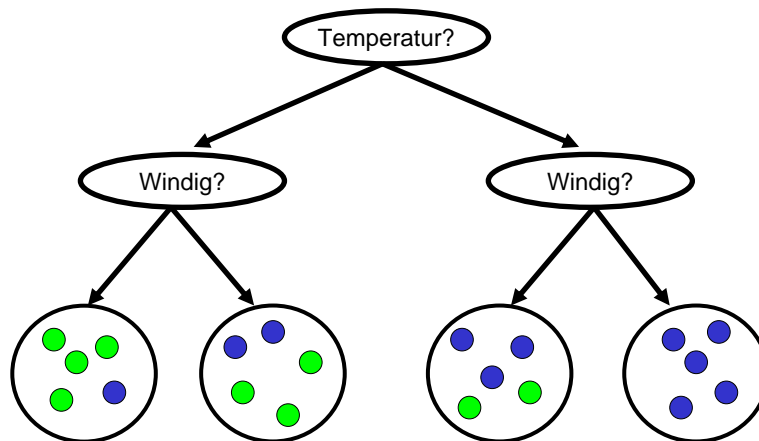


Abbildung 2: Dieser Entscheidungsbaum wurde über zwei binäre Features gebildet und enthält 20 Observationen, die entweder blau oder grün gefärbt sind.

3.2 Entscheidungsbaum

3.2.1 Idee

Stellen wir uns als Beispiel vor, dass wir eine Alice gerne schwimmt. Zusätzlich ist uns bekannt an welchen Tagen Alice schwimmen gegangen ist und an welchen nicht. Eine anderes, als das zuvor beschriebene Verfahren, ist das Erstellen eines Entscheidungsbaums anhand der verschiedenen Eigenschaften der vergangenen Tage. Anschließend fügen wir den kommenden Tag ein und sagen, dass Alice am kommenden Tag schwimmen geht, wenn der Knoten, zu dem der neue Tag zugeordnet wurde, hauptsächlich Tagen entspricht, an denen Alice schwimmen gegangen ist.

3.2.2 Formale Beschreibung

In einem Entscheidungsbaum beschreiben wir mit jedem inneren Knoten ein Feature. Die einzelnen Kanten des Baumes entsprechen einem Wert (bei kategorischen Variablen) oder einem Wertebereich (bei kontinuierlichen Variablen). Die Blätter eines Baumes hingegen repräsentieren einen möglichen Wert, der sich durch die entsprechenden Observationen zusammensetzt. Die-

se Zusammensetzung kann durch verschiedene Votings oder Mittelungen berechnet werden. Im Training teilen wir die Observationen der Features in Untermengen. Dies wird für jedes Feature und jede Untermenge rekursiv wiederholt, solange bis jede Untermenge von jedem Feature geteilt wurde.

In Abbildung 2 sehen wir, dass wir unsere Observationen in vier Untermengen geteilt haben, die jeweils ein Blatt des Baumes darstellen. Nun können wir neue Observationen mit einem entsprechenden Voting in den Blättern klassifizieren.

3.2.3 Parameter

Wie auch schon bei dem k-Nearest-Neighbor Model gibt es auch beim Entscheidungsbaum einige Parameter, welche man optimieren kann.

Allgemein Parameter

Splitting. Die Art des Aufteilens der Observationen in Untermengen kann entscheidend sein. Es gibt verschiedene Arten des Aufteilens, darunter sind Guillotine-Cuts für Binär-Bäume (CART) oder nicht binäre Unterteilungen wie bei C4.5 Bäumen (wie in WEKA [2]).

Pruning. Oftmals sind detaillierte Bäume zu komplex oder auch zu fehleranfällig. Deshalb kann man Bäume auch beschneiden, indem Blätter und damit auch Knoten verschmolzen werden. Dies steigert oftmals die Vorhersagekraft als auch die Robustheit.

Klassifikationspezifisch Parameter

Voting. Das Voting bezieht sich auf das Verhalten der Blätter des Baumes, d.h. wie jede neue Observation welcher Klasse zugeteilt wird. Die Votingmöglichkeiten sind ähnlich wie die des k-Nearest-Neighbor Verfahrens.

Regressionspezifisch

Verrechnung. Auch die Verrechnung passiert wie bei dem k-Nearest-Neighbor Verfahren, und sogar die Abstandsgewichtungen können im Fall eines vorher erfolgten Prunings auf die Blätter angewendet werden.

3.3 Support Vector Machine

3.3.1 Idee

Betrachten wir wieder unsere Bekannte Alice und ihre Schwimmgewohnheiten. Diesmal stellen wir uns ein zweidimensionales Koordinatensystem vor,

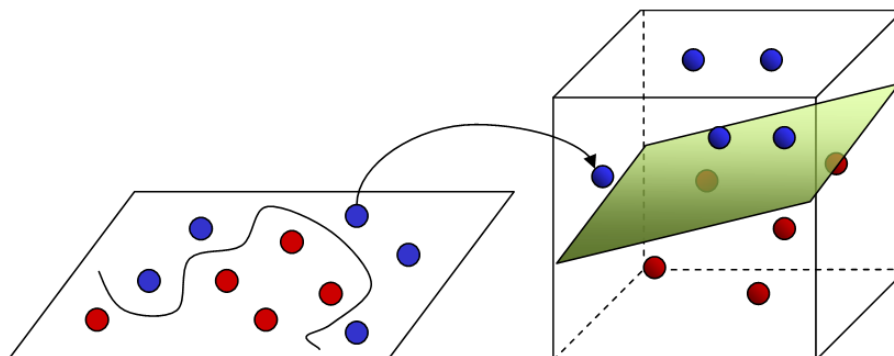


Abbildung 3: Damit die Support Vector Machine eine ideale Trennung der Punkte erreichen kann, werden die Punkte in eine höhere Dimension transformiert, in der sie besser separierbar sind.

in das wir die Tage entsprechend der Temperatur und der Windstärke eintragen. Anschließend färben wir die verschiedenen Punkte wie in Abbildung 3 rot wenn sie nicht schwimmen geht und blau, wenn sie geht. Nun sehen wir, dass die roten und blauen Punkte durcheinander gewürfelt sind und nicht mittels einer Geraden getrennt werden können.

Damit wir nun die Punktemengen trennen können, transformieren wir unsere Tage in eine höhere Dimension (Abbildung 3), indem wir eine neue dritte Koordinate einführen, die sich aus den bereits bekannten Koordinaten zusammensetzt. Nun können wir in der höheren Dimension die Tage, an denen Alice schwimmen geht von den Tagen, an denen sie nicht schwimmen geht, durch eine Ebene trennen.

3.3.2 Formale Beschreibung

Die Support Vector Machine transformiert den bekannten Featureraum in einen höherdimensionalen Raum. In diesem Raum wird eine trennende Ebene zwischen die Klassen beschrieben durch die abhängige Variable gelegt. Diese Ebene hat den maximalen Abstand zu den nächsten Punkten der unterschiedlichen Klassen. Durch diese nächsten Punkte wird jeweils eine Ebene parallel zur trennenden Ebene gelegt. Diese Ebenen nennt man auch Support-Vektoren.

Die zugrunde liegende Annahme ist, je weiter die parallelen Ebenen auseinander liegen, desto besser ist die Vorhersagekraft der Support Vector Machine. Deswegen ist die Suche nach den besten Support-Vektoren die ei-

gentliche Aufgabe einer Support Vector Machine, denn die trennende Ebene wird einfach zwischen die Support-Vektoren gelegt.

3.3.3 Parameter

Wie auch schon bei der kNN Methode und bei den Entscheidungsbäumen gibt es auch bei der Support Vector Machine einige Parameter, welche man an seine Datenmenge anpassen sollte.

Allgemein Parameter

Kernel. Der Kernel beschreibt, wie die Ebenen im höher dimensional Raum verlaufen. Ursprünglich besaßen Support Vector Machines nur lineare Kernels, welche einfache Hyperplanes darstellen. Dies wurde jedoch erweitert, um komplexere Zusammenhänge darzustellen.

Bestrafung. Da es oftmals nicht möglich ist, die Punktmengen zu trennen bzw. man bei einer Regression oftmals Abweichungen in Kauf nehmen muss, kann man die Stärke des Bestrafungsterms regulieren. Diese Regulierung erweist sich oft als sinnvoll, wenn man die Ergebnisse zu Gunsten einer bestimmten Klasse oder Art der Abweichung verschieben will.

Klassifikationspezifisch Parameter

ν -Class. Diese Art der Klassifikation versucht zwei Klassen voneinander zu separieren. Zusätzlich gibt diese Art der Klassifikation uns die Möglichkeit, die Anzahl der Support-Vektoren zu kontrollieren. D. h. während der Berechnung der idealen trennenden Ebene werden alle möglichen Paare von Support-Vektoren gegeneinander getestet. Mit der Kontrolle über diese Anzahl an zu testenden Paaren kann die benötigte Rechenzeit reduziert werden.

Multi-Class. Da wir oft Observationen in mehr als nur zwei Klassen einteilen wollen, kam die Idee der Multi-class Support Vector Machine auf. Das Grundprinzip basiert auf der „jeder gegen jeden“ Idee. Hier wird für jede Konstellation eine Support-Vektor-Machine trainiert und durch Voting die entsprechende Vorhersage ermittelt.

Regressionsspezifisch Parameter

ϵ -Regression. Die trennende Ebene fungiert als Näherungsfunktion zu den Werten der Datenmenge. Hierbei fungieren die Support-Vektoren zur Unterdrückung von Ausreißern, die normalerweise Modelle negativ beeinflussen würden. D.h. die trennende Ebene wird nur an die Datenpunkte zwischen den beiden Support-Vektoren angepasst.

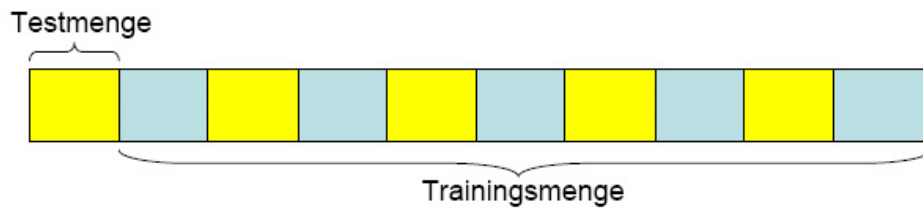


Abbildung 4: Unsere Datenmenge ist in zehn Partitionen unterteilt und abwechselnd wird jede Partition als Testmenge benutzt.

v -Regression. Die v -Regression erweitert die ϵ -Regression um die schon in der v -Class Klassifikation erwähnte Eigenschaft.

4 Cross Validation

Nachdem wir nun einige Machine Learning Modelle kennen gelernt haben, kommen wir zu der *State of the Art* Evaluierungsmethode, Cross Validation. In diesem Abschnitt erläutern wir was Cross Validation ist.

4.1 Idee

Die zugrunde liegende Idee der Cross Validation ist recht unkompliziert. In der Regel reichen die vorhandenen Daten nicht aus, um sie in eine Trainings- und Testmenge zu unterteilen.

Um die vorhandenen Daten optimal zu nutzen, werden sie in n Partitionen unterteilt. Nun werden $n - 1$ Partitionen als Trainingsmenge und die verbleibende Partition als Testmenge verwendet. Nachdem jede der n Partitionen einmal als Testmenge fungiert hat, werden alle Ergebnisse aus den einzelnen Durchgängen gemittelt. Diese einzelnen Ergebnisse liegen in verschiedenen Formen wie z.B. Spearman Rang Korrelationskoeffizienten vor.

Wie wir in Abbildung 4 sehen, haben wir die Datenmenge in 10 Partitionen unterteilt und verwenden jede Partition einmal als Testmenge. Die erhaltenen Ergebnisse werden im Anschluss an die zehn Durchläufe gemittelt.

4.2 Vorteile

Die Vorteile liegen klar auf der Hand. Zum einen können wir die gesamte Datenmenge benutzen, um unsere Modell zu bewerten, und zum anderen ist die Cross Validation unabhängig von den Eigenschaften der einzelnen Modelle.

4.3 Nachteile

Die Nachteile sind nicht offensichtlich. Wie schon genannt ist der größte Nachteil die Vernachlässigung verschiedener wichtiger Aspekte, die man untersuchen sollte. Um genauer zu sein, die Cross Validation fokussiert lediglich den *Sub-Set-Fit* (siehe Abbildung 5.1).

Ein weiterer Nachteil besteht in der Bewertung der einzelnen Durchläufe. Es ist nicht immer klar, welche Bewertungsart wir wählen sollten, um ein optimales Ergebnis zu erzielen. Zudem wird Cross Validation auch oft in Situationen benutzt, in denen sie keine wirkliche Aussagekraft besitzt, aber aufgrund ihrer geringen Voraussetzungen dennoch anwendbar ist.

4.4 Ausprägungen

Es gibt verschiedene Arten der Cross Validation(CV). Wir stellen im Folgenden die wichtigsten Ausprägungen vor:

10-Fold CV. Bei einer 10-Fold Cross Validation unterteilen wir unsere vorhandenen Daten in zehn Partitionen(siehe Abb. 4) und gehen wie oben beschrieben vor. Die 10-Fold Cross Validation hat den Vorteil, dass man eine angemessene Testmengengröße während der Evaluierung nutzen kann.

Leave One Out CV. Eine Leave One Out oder auch Jack Knife Cross Validation teilt die Daten in n Partitionen auf, hierbei ist n die Anzahl an vorhandenen Daten. Durch die großen Trainingsmengen kann man den größtmöglichen Informationsgewinn für die Modelle erzielen.

5 Evaluations Aspekte

Nachdem wir nun die Cross Validation und ihre Vor- bzw. Nachteile kennen gelernt haben, widmen wir uns nun allen Aspekten, die für die Auswahl eines Models von Bedeutung sind. Wir diskutieren zuerst den bereits erwähnten *Sub Set Fit*, gefolgt von der *Similarity*, die die Vorhersagen von ähnlichen Daten beurteilt, und schließen mit der *Simplicity*, die die Komplexität eines Models beschreibt.

5.1 Sub-Set-Fit

Der *Sub-Set-Fit* beschreibt die Idee, die hinter der Cross Validation steht. Wir trainieren ein Model mit einer Untermenge(sub set) der vorhanden Daten und evaluieren die Performance des Models auf einer gegebenen Testmenge. Dies ist ein wichtiger Aspekt, da wir immer nur einen Teil der Daten zum Trainieren benutzen können, wenn man bedenkt, dass die meisten Daten noch unbekannt sind. Die Beurteilung eines *Sub-Set-Fits* wird häufig

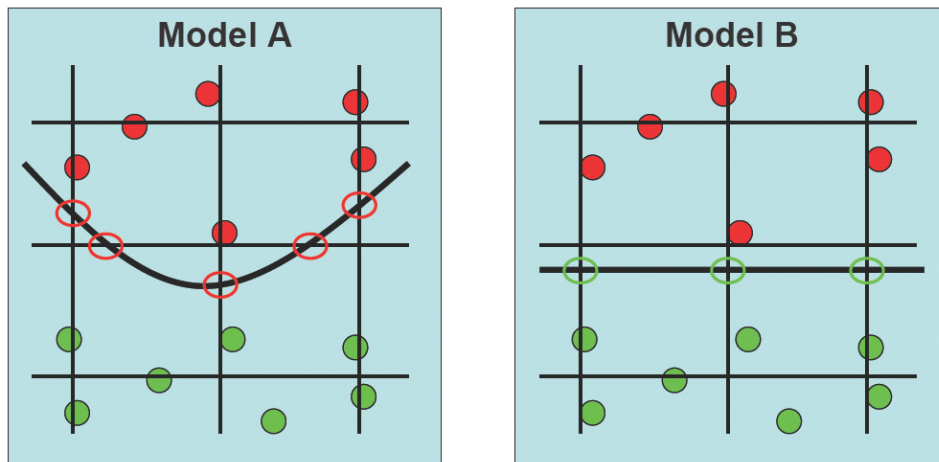


Abbildung 5: Model A hat einen Simplicity Wert von 5 wohingegen Model B mit einem Wert von 3 einfacher ist.

durch den Accuracy Wert bestimmt, kann aber auch über Precision und Recall oder auch durch Korrelationskoeffizienten bestimmt werden.

Um den *Sub-Set-Fit* zu bestimmen sollte man eine Cross Validation oder ggf. eine passendere Validierungsmethode benutzen.

5.2 Similarity

Die *Similarity* kann man aus dem k-Neighrest-Neighbor (kNN) Verfahren ableiten. Beim kNN Verfahren werden Datenpunkte wie ihre Nachbarn klassifiziert. Um nun die Similarity zu messen, können wir Datenpunkte in verschiedene Mengen einteilen. Hierzu gruppieren wir die Daten entsprechend eines angemessenen Koeffizienten, wie zum Beispiel Pearsons Korrelationskoeffizienten oder dem Euklidischen Abstand. Nun können wir unsere Modelle trainieren und mit unserem neuen Testdaten, welche nur ähnliche Daten enthalten, evaluieren.

5.3 Simplicity

Die *Simplicity* beschreibt die Einfachheit eines Modells. Im Machine Learning wird nach der folgenden Grundregel entschieden:

Erzielen zwei Modelle ähnlich gute Ergebnisse, entscheiden wir uns für das einfachere.

Der Hintergrund für diese Grundregel liegt an dem möglichen *Overfitting*, unter dem oftmals komplexere Modelle leiden. Wir werden nun Overfitting und Underfitting erläutern.

Overfitting. Komplexe Modelle haben die Möglichkeit, sich sehr stark an die vorhandenen Daten anzupassen. Da aber gemessenen Daten oft mit Messfehlern durchsetzt sind, liegt in den Daten oftmals ein hohes Maß an Rauschen vor. Dieses Rauschen und ggf. Ausreißer können dazu führen, dass Modelle sich zu stark von Anomalien beeinflussen lassen. Wenn ein Modell sich sehr stark an diese Daten anpassen kann spricht man von einer Gefahr des Overfittings.

Overfitting kann auch eintreten, falls sich zwei Punktemengen räumlich überlappen. In einem solchen Fall kann die Grenze, die zwischen den beiden Klassen gezogen wird, übermäßig komplex ausfallen.

Underfitting. Underfitting stellt, wie schon im Namen, das Gegenteil zum Overfitting dar. In der Regel leiden sehr einfache Modelle unter Underfitting. Es tritt oft ein wenn nur wenige Daten vorhanden sind. Wenn zusätzlich ein Modell Ausreißer ignoriert, können einzelne Datenpunkte außer Betracht fallen, obwohl diese einen wichtigen Teil der Daten repräsentieren und ausschlaggebend sein sollten. Dies fördert dies die Gefahr von Underfitting.

Die größte Schwierigkeit liegt oftmals darin eine geeignete Balance zwischen Over- und Underfitting zu finden. Um dazu in der Lage zu sein, müssen wir die Komplexität eines Modells messen können.

Andersson et al. [1] haben eine Methode vorgeschlagen, um die Komplexität eines Modells zu beurteilen. Allerdings ist diese Methode nichtdeterministisch und um die Wiederholbarkeit von Experimenten zu garantieren haben Lavesson und Davidsson eine deterministische Methode vorgestellt.

Wie wir in Abbildung 5 sehen legen wir in unser Koordinatensystem ein Gitternetz, welches in höher dimensional Räumen durch ein Ebenennetz ersetzt wird. Nun zählen wir einfach, wie oft in einem Fenster fester Größe das Gitternetz von unserer Entscheidungsgrenze bzw. Funktion geschnitten wird. Diese Anzahl an Schnittpunkten ist der Indikator für die Komplexität unseres Modells. Entsprechend der Beschreibung bedeutet ein niedriger Wert eine geringe Komplexität und ein hoher Wert entspricht einer hohen Komplexität.

Die Grenzen dieser Methode liegt darin, dass wir diese immer nur auf einen begrenzten Bereich anwenden können. Dies liegt daran, dass dem Training nur endliche viele Observationen zu grunde liegen und deshalb das erstellte Modell nur in einem begrenzten Bereich einsetzbar ist. Zudem hängt der Messwert für die ein Modell auch von dessen Lage ab, dies kann zu irreführenden Komplexitätswerten führen.

6 Measure Function

Da wir nun gesehen haben, dass die Cross Validation zwei von drei wichtigen Aspekten zur Beurteilung eines Vorhersagemodells ignoriert, werden wir nun die *Measure Function* von Lavesson und Davidsson vorstellen.

6.1 Interpretation

Da wir uns die drei Aspekte vor Augen geführt haben, können wir die Aufstellung der eigentlichen Measure Funktion direkt herleiten:

$$a_0 \cdot subset + a_1 \cdot simi - a_2 \cdot simp$$

Subset steht für die Qualität des Sub-Set-Fits, *simi* repräsentiert die Similarity und *simp* entspricht der Komplexität des Modells. Die Variablen a_i verleihen uns die Möglichkeit, einzelne Aspekte zu betonen und die Modelle von verschiedenen Gesichtspunkten aus zu beurteilen.

Die Werte für Sub-Set-Fit und Similarity werden addiert, da wir hier möglichst hohe Werte erzielen, wohingegen die Simplicity aufgrund der gegensätzlichen Interpretation abgezogen wird. Das heißt je höher der erzielte Wert ist, desto besser ist die Performance unsers Modells.

6.2 Vorteile

Die Vorteile der Measure Funktion liegen auf der Hand. Sie ermöglicht uns alle drei Aspekte eines Modells in einem Wert zusammenzufassen und somit alle vorhandenen Eigenschaften zur Beurteilung heranzuziehen. Dies hat zur Folge, dass wir die Auswahl und Ermittlung optimaler Parametrisierung der Modelle automatisieren können ohne befürchten zu müssen, ein nicht optimales Model zu erhalten, das zum Beispiel unter Overfitting leidet.

6.3 Nachteile

Der Nachteil liegt in der Berechnung der einzelnen Werte wie der Similarity. Denn es ist oft nicht klar wie die entsprechend beste Messmethode für die gegebene Datenmenge auszusehen hat.

7 Fazit

Cross Validation ist die State of the Art Evaluierungsmethode für Machine Learning Modelle. Wie wir gesehen haben ignoriert Corss Validation zwei von drei wichtigen Beurteilungskriterien für ein Model. Diese zwei Aspekte sind die Komplexität und die Art der Klassifizierung von ähnlichen Datenpunkten.

Durch diese Vernachlässigung und der riesigen Auswahl an Modellen und deren Parametrisierung kann es leicht zu einer falschen Entscheidung kommen. Das heißt, wir können zum Beispiel Gefahr laufen ein Model auszuwählen, das unter Overfitting leidet.

Deshalb haben wir die Measure Function von Lavesson und Davidsson vorgestellt, um eine bessere Alternative zur Cross Validation aufzuzeigen. Sie vereinen alle drei Aspekte in sich und bietet gleichzeitig die Option, einzelne Aspekte stärker zu gewichten. Daher ziehen wir den Schluss, dass sobald man in der Lage ist Measure Functions zu benutzen sie auch verwenden sollte.

Literatur

- [1] A. Andersson, P. Davidsson, and J. Linden. Measure-based classifier performance evaluation. In *Pattern Recognition Letters*, pages 1165–1173, November 1999.
- [2] M. Kaufmann. *Data Mining: Practical machine learning tools and techniques*. 2005.
- [3] N. Lavesson. Evaluation of classifier performance and the impact of learning algorithm parameters. Master's thesis, 2003.
- [4] N. Lavesson and P. Davidsson. A multi-dimensional measure function for classifier performance. In *Proceedings of the 2nd IEEE International Conference on Intelligent Systems*, pages 508–513, 2004.