Analyse des Dockingverhalten von PARP10 und PARP14

von

Immanuel Bayer

Lucas Beyer

Supinya Manodumrongthum

Betreuer: Prof. Dr. Andreas Schuppert

Aachen, April 2011

I. Inhaltsverzeichnis

I.Inhaltsverzeichnis

I.Inhaltsverzeichnis

I.Inhaltsverzeichnis

II. Einfürung

L. Beyer

1. Motivation und Ziel der Arbeit

PARP ist eine Proteinfamilie die in der Reparatur der Zellen und in dem programmierten Zelltod eine wichtige Rolle spielt. PARP-Inhibitoren bilden eine Klasse pharmazeutischer Inhibitoren die insbesondere in der Krebsbekämpfung zu gelten kommen. Deshalb ist es notwendig zu Analysieren, welche Proteine sich gut an PARP binden und aus welchem Grund .sie das tun.

Solch eine Analyse wird in dieser Arbeit auf der Primärstruktur von Proteinen durchgeführt, die in einem Microarray auf Bindungen mit PARP10 und PARP14 untersucht wurden.

2. Datenaufbereitung

Wir haben eine Excel-Tabelle bekommen, in der eine Liste von Proteinen mitsamt deren Namen, ID und Z-score [CHE03] mit jeweils PARP10 und PARP14 enthalten sind.

Nach einer ersten explorativen Datenanalyse mittels Orange [ORANGE] ist uns aufgefallen, dass die Z-score Werte einen anormalen Wertebereich decken, wie auf Abb 2, und zu sehen ist.

Dadurch haben wir einen Fehler in der Datenkonvertierung entdeckt, woraufhin wir die Originaldaten im HTML-Format bekamen. Diese haben wir mittels regulärer Ausdrücke [AHO90] in ein für uns geeignetes CSV-Format [CSV05] umgewandelt.

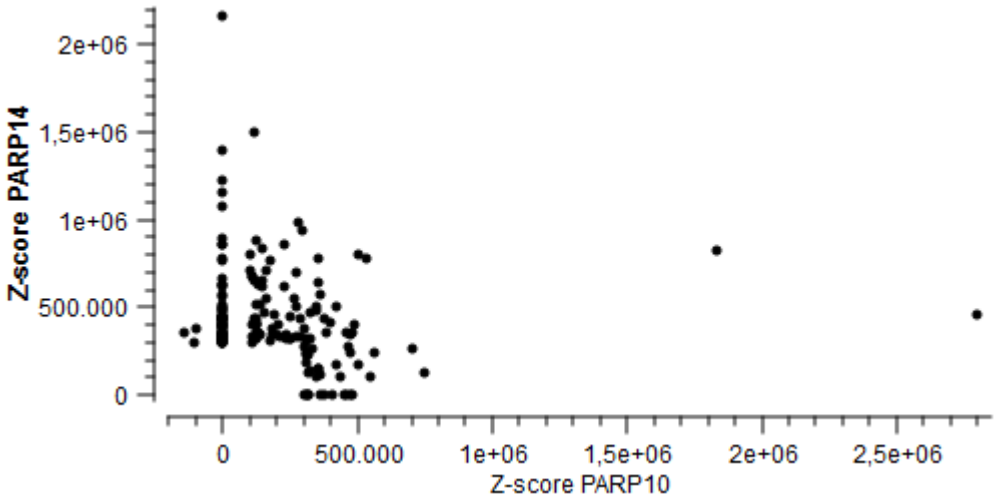


Abb 1: Scatterplot der PARP10 und PARP14 original Z-scores

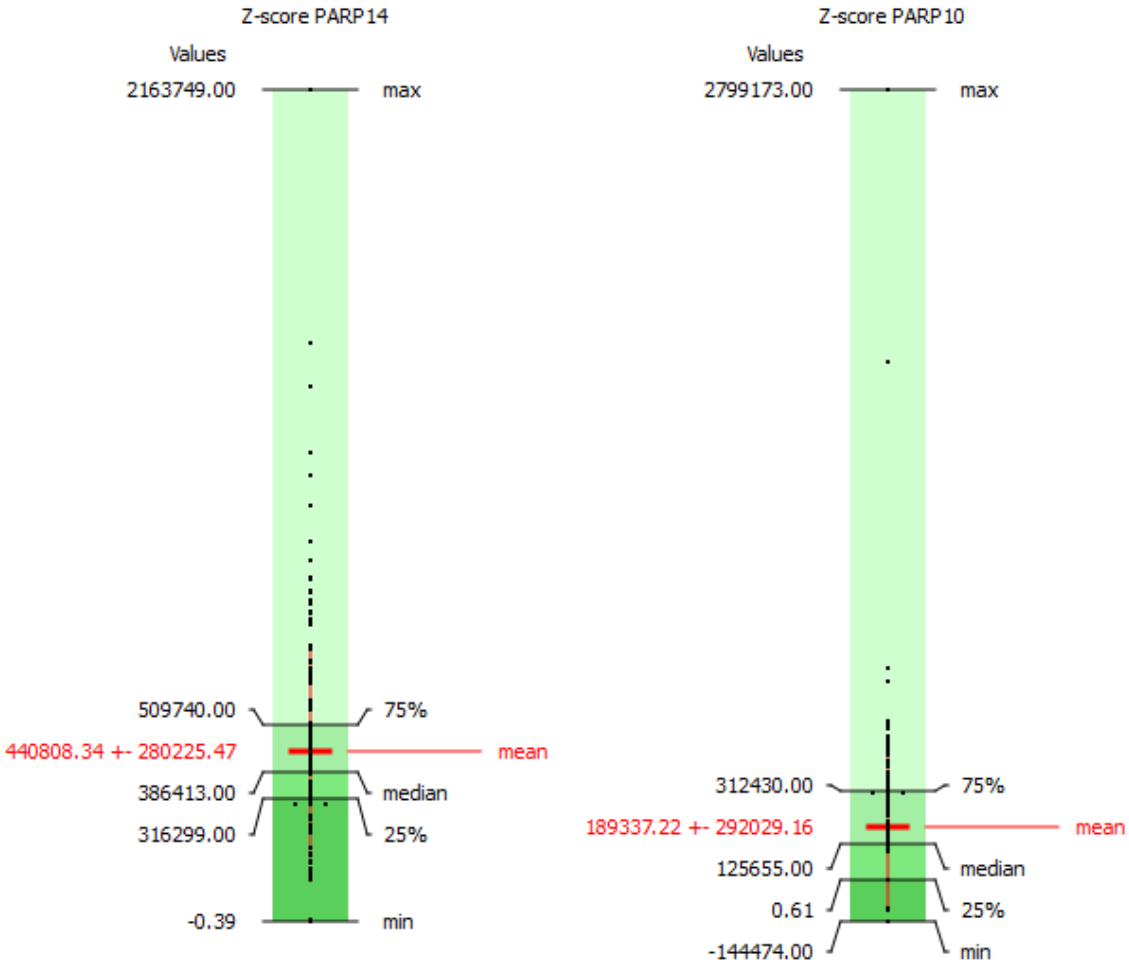


Abb 2: Statistiken zu den PARP10 und PARP14 original Z-scores.

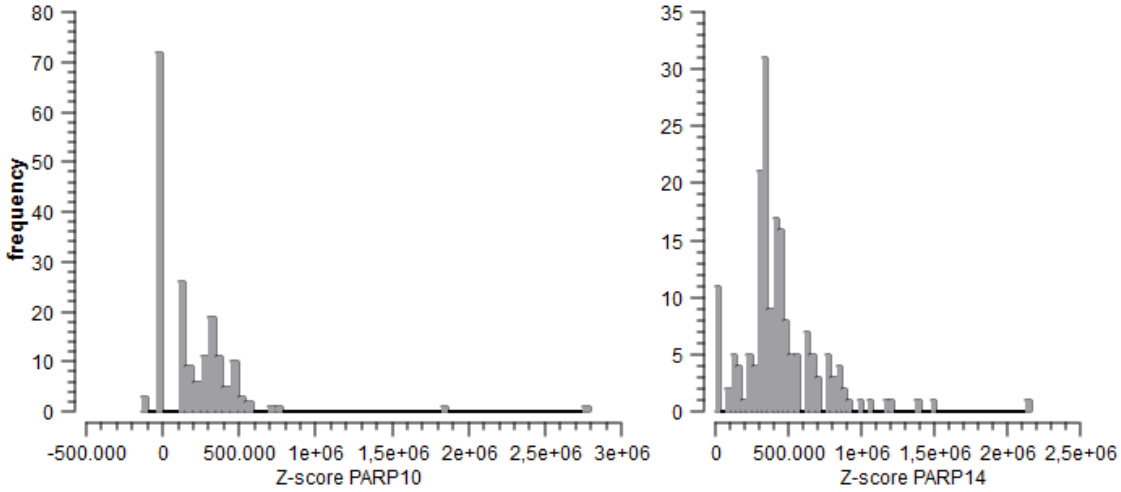


Abb 3: Häufigkeitsverteilung der PARP10 und PARP14 original Z-scores

Somit standen uns folgende Datensätze zur verfügung:

- 61 PARP10-*Positive* Proteinsequenzen mit Z-score,
- 8215 PARP10-*Negative* Proteinsequenzen mit Z-score,
- 149 PARP14-*Positive* Proteinsequenzen mit Z-score und
- 8125 PARP14-*Negative* Proteinsequenzen mit Z-score.

Eine PARP10-Positive bzw. PARP10-Negative Proteinsequenz ist eine Proteinsequenz deren Z-score aussagt, dass diese sich höchstwahrscheinlich mit PARP10 eine bzw. keine Verbindung eingegangen ist, also mit PARP10 clustert bzw. Nicht clustert. Analoges gilt für PARP14.

Weiterhin haben wir sowohl die Aminosäuren der Proteinsequenzen als auch deren Gruppierung, basierend auf der chemischen Charakterisierung verwendet:

- Basisch (B): R, H, K
- Sauer (J): D, E
- Neutral & Polar (Z): S, T, N, Q, C, Y
- Neutral & Unpolar (X): G, A, V, L, I, P, M, F, W

Zusätzlich wurde uns gesagt, dass wenn ein Fenster für die Reaktion verantwortlich ist, dieses mit hoher Wahrscheinlichkeit mindestens eine saure Aminosäure (also Gruppe J, Aminosäure D oder E) enthält.

3. Übersicht

Während dieser Arbeit haben wir das Problem aus zwei grundsätzlich verschiedenen Ansichten betrachtet. Daher haben wir auch zwei grundsätzlich verschiedene Lösungswege befolgt.

Die eine Sichtweise ist die des Dataminings. Das Ziel dieser Herangehensweise war es, eine SVM zu trainieren, welche dann für beliebige neue Proteinsequenzen vorhersagen kann, ob diese mit PARP10 oder PARP14 clustern oder nicht.

Die andere Sichtweise ist die der Mustersuche. Das Ziel dieser Herangehensweise war es, in den Sequenzen ein Muster zu entdecken, welches in möglichst vielen Positiven und in möglichst wenigen Negativen Proteinsequenzen auftritt. Ein solches Muster sollte für Menschen intuitiv deutbar sein.

III. Dockingvorhersage mittels SVM

I. Bayer

1. Einführung in die Klassifizierung von Sequenzen

Das Klassifizieren von Sequenzen mittels Machine Learning (ML) birgt eine Besonderheit zu anderen Problemstellungen in diesem Bereich. Diese liegt darin, dass keine expliziten *Features* vorhanden sind. Mit Features werden die Eingenschaften einer Instanz des Datensatzes beschrieben. Wollte man z.B. das Wetter für morgen vorhersagen, könnte der Datensatz aus bestimmten Messungen des Vortages, wie Luftdruck, Bewölkungsgrad etc. bestehen. Jeder gemessene Tag ist dann eine *Instanz* des Datensatzes und die verschiedenen Messungen die *Features*. Wenn wir also diese Begriffe auf unser Proteinklassifizierungsproblem anwenden, wird klar, dass wir pro Instanz nur ein Feature haben, nämlich die Sequenz selbst. Für Supervised Classification benötigen wir noch ein *Label*, also für jede Instanz die Information, welchen Wert die vorauszusagende Größe in diesem Fall angenommen hat. Da die meisten Machine Learning Algorithmen ihre Klassifizierungsentscheidungs auf Grundlage von Features lernen, können wir unseren Datensatz in der sequenziellen Form so nicht verwenden. Im nachfolgenden wird kurz auf zwei Möglichkeiten eingegangen, wie man diese Inkompatibilität umgehen kann.

1.1. Distanz basierte Klassifizierung

Die erste Möglichkeit ist, eine Metrik für die Distanz zwischen zwei Instanzen zu definieren. Dadurch kann die paarweise Distanz aller Instanzen des Datensatzes als Featurematrix interpretiert und als Eingang für den Klassifizierungsalgorithmus verwendet werden.

1.2. Feature basierte Klassifizierung

Es ist jedoch auch möglich die Features aus dem ursprünglichen Datensatz zu generieren und somit Eigenschaften der Sequenz als Feature zu verwenden, z.B. die Häufigkeit mit der ein bestimmtes Element vorkommt. Eine mögliche Umsetzung dieses Ansatzes werden wir in Abschnitt Feature Konstruktion noch genauer betrachten. Dieser Ansatz führt jedoch Häufig dazu, dass eine sehr große Anzahl von Features generiert wird, die nur einem sehr kleinen Teil der Instanzen

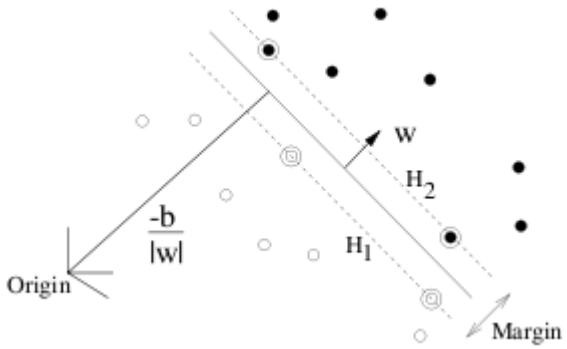
vorkommt. Wir erhalten also ein dünn besetztes und mehrdimensionales Problem. Viele ML-Ansätze können jedoch nicht gut mit hochdimensionalen Problemstellungen umgehen oder erfordern nicht realisierbare Rechenlaufzeiten. Es gilt daher einen Algorithmus zu wählen, der mit solchen Problemstellungen umgehen kann oder eine Reduktion auf die wesentlichen Features durch zu führen; eine Aufgabe die für sich selbst eine Herausforderung darstellt. Im folgenden wird der Support Vector Algorithmus beschrieben welcher sich für dünn besetzte und hochdimensionale Probleme bewährt hat und in She et al. (2003) als besonders effektiv für die Klassifizierung von Proteinen bewertet wird. Xing, Pei, & Keogh (2010) bietet eine gute Zusammenfassung über die Klassifizierung von Sequenzen.

2. Grundlagen SVM

Im Folgenden wird eine kurze Zusammenfassung über die Funktionsweise von Support Vektor Machinen, anhand des binären Klassifizierungsproblems dargelegt. Eine ausführliche Einführung bietet Burges (1998).

Large margin Seperation

Die Grundidee ist, dass ein Datensatz durch eine Trennebene aufgeteilt wird, so dass der größtmögliche Abstand (Margin) zwischen zwei beiden Klassen entsteht.



Grafik 1: Quelle:(Burges, 1998)

Die Entscheidungsfunktion Eq. 1 berechnet einen Wert für jede zu klassifiziere Instanz und weist je nach Vorzeichen die eine oder die andere Klasse zu.

$$f(x)=\langle w, x\rangle+b \tag{1}$$

Hard Margin

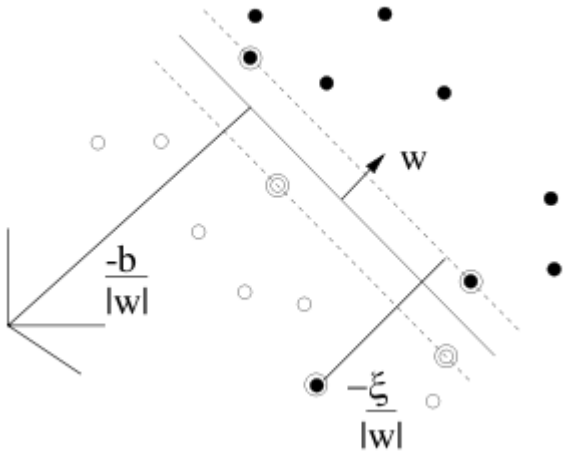
Um den Abstand zwischen den beiden Klassen zu minimieren wird das in Eq. 2 definierte Optimierungsproblem gelöst, wobei diese Formulierung die Bedingung enthält das die beiden Klassen perfekt trennbar sind.

$$\begin{aligned} &\underset{\alpha,b}{minimize}\ \frac{1}{2}\|w\|^2 \\ &subject\ to:\ y_1(\langle w, x_i\rangle+b)\geq 1, \\ &for\ \ i = 1,...,n \end{aligned} \tag{2}$$

Soft Margin

Für Klassifizierungsaufgaben bei denen eine perfekte Trennung nicht möglich ist, wird die Bedingung aus Eq. 2 durch Eq. 3 ersetzt. Sogenannte Slack variables ermöglichen auch Lösungen mit falsch klassifizierten Instanzen.

$$y_i(\langle w, x_i \rangle + b) \geq 1 - \xi, \quad for\ i = 1,...,n, \tag{3}$$



Grafik 2: Quelle: (Burges, 1998)

Dual Formulierung

Eine Umformulierung des Optimierungsproblems mit Soft-Margin ist in Eq. 4 zu sehen, diese Formulierung ermöglicht eine effizientere Berechnung, auch für sehr große Datenmengen.

$$\begin{aligned} &\underset{\alpha}{maximize} \sum_{i=1}^{n} \alpha_i \\ &\frac{-1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \\ &subject\ to: \sum_{i=1}^{n} y_i \alpha_i = 0, 0 \leq \alpha_i \leq C \end{aligned} \tag{4}$$

In Eq. 5 ist die Lösung des Problem zu sehen, es gehen nur die Instanzen ein für die x ungleich Null ist.

$$w = \sum_{i=1}^{n} y_i \alpha_i x_i \tag{5}$$

Kernels

Wichtig ist das die Formulierung in Eq. 6 Nur von dem Skalarprodukt im Featureraum abhängt.

$$f(x)=\sum_{i=1}^{n} y_i \alpha_i \langle \phi(x_i), \phi(x) \rangle + b \tag{6}$$

Man definiert daher eine Kernelfunktion (Eq. 7), welche es erlaubt das Problem zu lösen ohne ein Mapping in dem möglicherweise sehr hochdimensionalen Raum durchzuführen.

$$k(x, x') = \langle \phi(x), \phi(x') \rangle \tag{7}$$

3. Spektrum Kernen für Sequenzen

Eine besondere Kernelfunktion für Sequenzen ist der Spektrumkernel, welcher in Eq. 8 gegeben ist.

$$k_l^{spectrum}(\overline{x}, \overline{x}') = \langle \phi_l^{spectrum}(\overline{x}), \phi_l^{spectrum}(\overline{x}') \rangle \tag{8}$$

Hier sind $\overline{x}, \overline{x}'$ zwei Sequenzen über dem Alphabet Σ. Die Anzahl der Elemente des Alphabets ist $|\Sigma|$. Damit ergibt sich für Sequenzen der Länge l , ein $|\Sigma|^l$ dimensionaler Featureraum, der bereits für l > 5 sehr schnell zu Laufzeiten führ die nicht realisierbar sind. Eine Lösung dafür ist nur die tatsächlich in den Sequenzen auftretenden l-mere zu nutzen, dies ermöglicht es in einem viel niedrigdimensionaleren Raum die Skalarprodukte zu bestimmen. Leslie, Eskin, Cohen, Weston, und Noble (2004) beschreiben einen Algorithmus der genau dies tut: Durch das Vorberechnen der Kernelmatrix wird dabei eine Laufzeit erreicht, die linear mit der Länge der Sequenzen ist.

3.1. ν-SVM

Der Strafterm C, in der in Eq. gewählte Formulierung erlaubt die Gewichtung zwischen Trainings und Generalisierungsfehler, lässt aber grundsätzlich für die Soft-Margin jeden positiven Wert zu. Die ν SVM-Formulierung garantiert, dass die Soft-Margin zwischen Eins und Null liegt. ν bekommt dadurch auch eine neue Interpretation und zwar die Doppelrolle Obergrenze für die Anzahl der Magin

Errors und die Untergrenze für die Anzahl der Support Vectoren. Eine genauere Beschreibung bieten Chen, Lin, und Schölkopf (2005).

4. SVM Wahrscheinlichkeitsschätzung

Eine von Platt vorgeschlagene Möglichkeit Klassenwahrscheinlichkeiten für SVM's zu erhalten, ist das Fitten eines parametrischen Modells. Empirische Betrachtungen zeigen, dass die bedingten Klassenwahrscheinlichkeiten zwischen der Margin der SVM annähernd exponentiell verteilt sind. Mit Hilfe der Bayes's Regel und zwei Exponentialfunktionen erhält man die parametrische Sigmoid Form.

$$P(x=1|f)=\frac{1}{1+\exp(Af+B)} \tag{9}$$

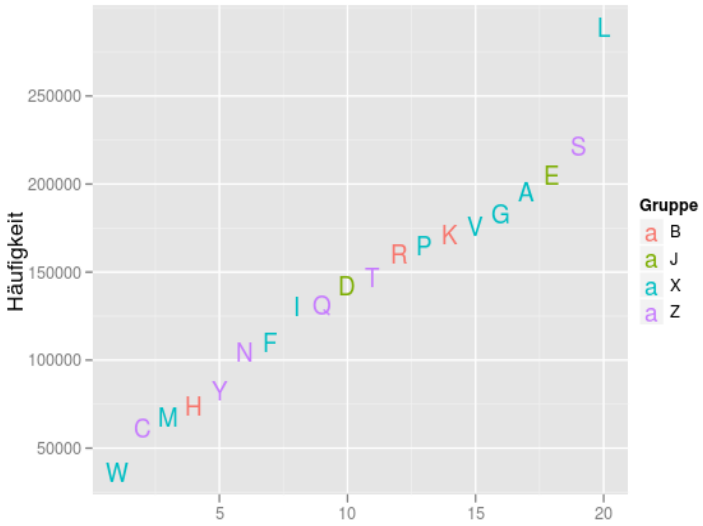
Die Parameter A und B können mit einer Maximum Likelihood Methode geschätzt werden. Ein ausführliche Herleitung bieten Platt (n.d.) und H. T. Lin, Lin, und Weng, (2007).

5. Umsetzung

In diesem Abschnitt wird der Datensatz genauer betrachtet, um einen Zugang zu den Daten zu finden. Besonders grafische Darstellungen ermöglichen es hier einen schnellen Überblick zu bekommen und eventuelle Datenfehler oder Besonderheiten zu erkennen. Ziel der Datenuntersuchung ist es, eine Möglichkeit zu finden, die als Sequenzen vorliegenden Daten in eine andere, für Machine Learning Algorithmen einfacher zugängliche Form, zu bringen. Im letzten Schritt vor der Modellierung wird der Datensatz aufgeteilt, so dass später eine allgemeine Aussage über die Prognosefähigkeit des Modells getroffen werden kann.

5.1. Daten Inspektion

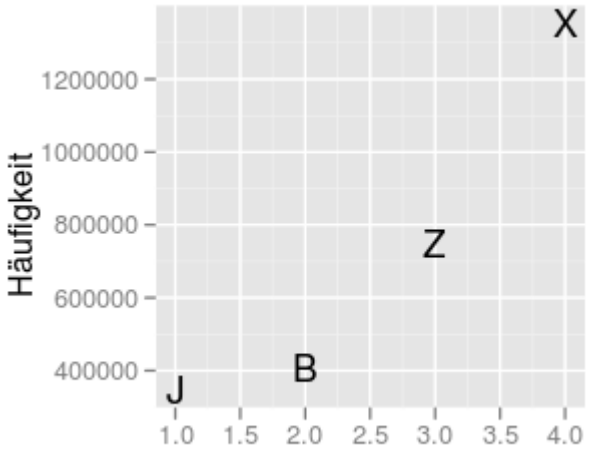
Die Proteinsequenzen liegen in einem Einbuchstabencode vor. Dies bedeutet, dass jeder Buchstabe in der Sequenz für eine Aminosäure steht. Insgesamt treten in dem gesamten Datensatz zwanzig verschiedene Aminosäuren auf. In der Grafik 3 ist die Häufigkeit jeder Aminosäure in dem Datensatz dargestellt. Die Aminosäuren wurden in der Grafik nach ihrer Häufigkeit sortiert. Auch wenn sich L und W etwas von dem Rest der Aminosäuren absetzen, ist keine Auffälligkeit zu erkennen, die einen Datenfehler vermuten lässt. Die einzelnen Aminosäuren lassen sich vier verschiedenen Gruppen zuordnen. Die Zuordnung der Aminosäuren zu den einzelnen Gruppen lassen sich an der Farbe der Buchstaben ablesen.



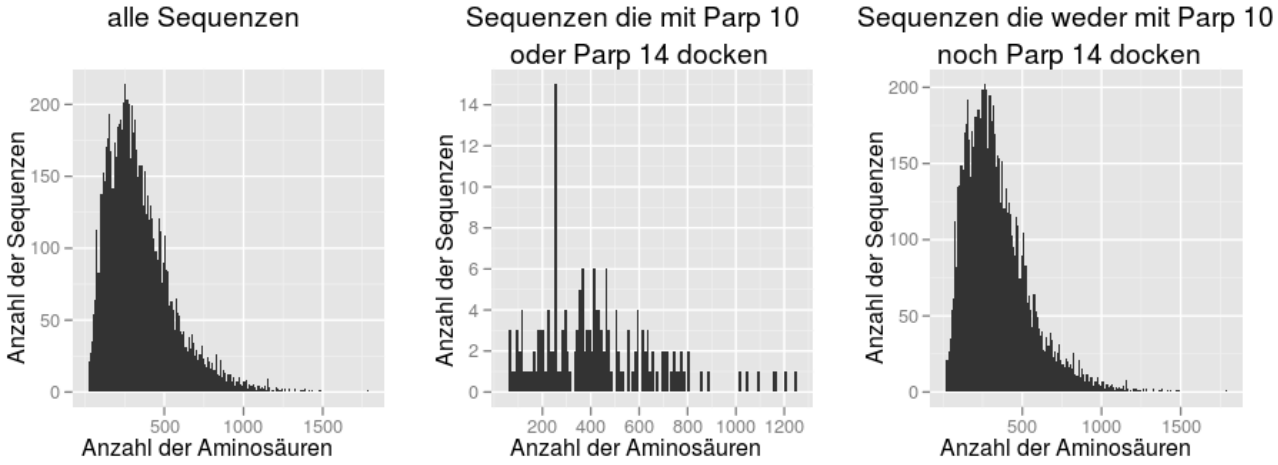
Grafik 3: Häufigkeit der Aminosäuren in dem vollständigen Datensatz.

Ein weiterer Datensatz wurde generiert, in welchem die Buchstaben der Aminosäuren durch den Buchstaben der Gruppe, zu der sie zugeordnet werden können, ersetzt wurde.

Die Häufigkeit mit der die für die vier Gruppen stehenden Buchstaben auftreten, sind in Grafik 4 zu sehen. Im Folgenden werden oft die beiden verschieden Kodierungen im Vergleich dargestellt. Grafik 5 gibt eine Übersicht darüber, wie häufig Sequenzen einer bestimmten Länge auftreten, wobei die Länge äquivalent zu Anzahl der Aminosäuren ist. Die linke Darstellung in Grafik 5 zeigt ein Histogramm über die Häufigkeit aller Sequenzlängen. In der Mitte sind nur die Längen der Sequenzen berücksichtigt, welche entweder mit Parp10 oder mit Parp14 eine Verbindung



Grafik 4: Häufigkeit der aus den Aminosäuren gebildeten Gruppen

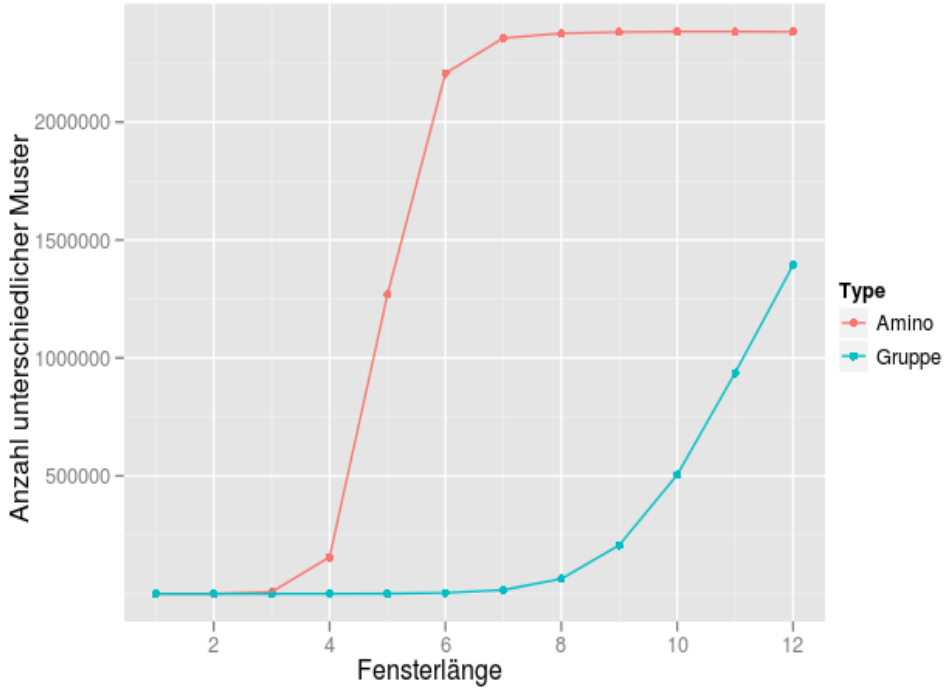


Grafik 5: Histogram übe die Länge der Sequenzen

eingegangen sind. Rechts ist nur die Länge aller Proteine berücksichtigt, die keine Verbindung eingegangen sind. Die bindenden Sequenzen sind über das vollständige Spektrum der vorkommenden Längen vertreten, Daher erscheint die Länge keine besondere Bedeutung zu haben. Weiter ist auffällig, dass es weit mehr nicht bindende Proteine gibt, als andere. Darauf werden wir später noch einmal zurückkommen.

5.2. Feature Konstruktion

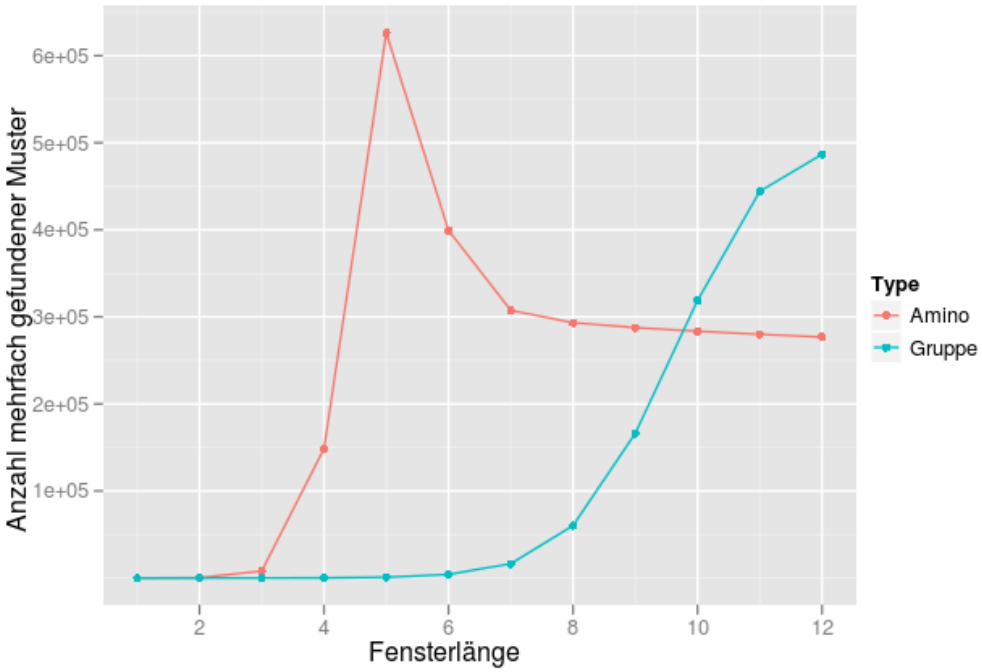
Wie bereits Eingangs diskutiert eignen sich die in Sequenzen vorliegenden Daten nur bedingt als Eingang für Machine Learning Algorithmen. Wir werden daher die Sequenzen in eine numerische Form überführen. Dafür schieben wir ein Fester der Länge l über alle Sequenzen und bestimmen so alle verschieden Fenster der Länge l, die in den Daten auftreten. Anschließend wird für jede Sequenz bestimmt, wie Häufig jedes einzelnes Fenster auftritt. Diese Häufigkeiten können nun als Features der jeweiligen Sequenz verwendet werden.



Grafik 6: Anzahl der Features mit Länge l im Datensatz

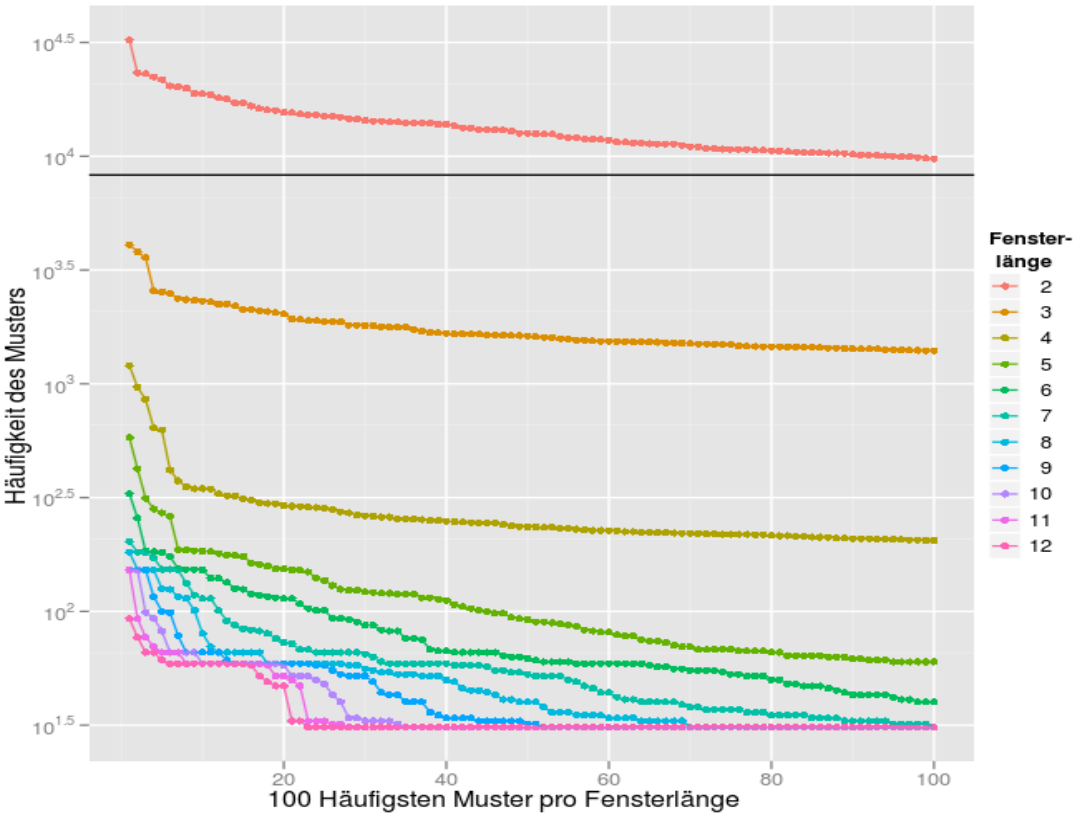
Intuitiv ist klar, dass mit der Länge des Fensters auch die Anzahl der verschiedenen Fenster steigt. Als Folge ist zu erwarten, dass sehr viele Features in den meisten Proteinen überhaupt nicht vorkommen. In solchen Fällen spricht man von einem stark dünn besetzten Problem. Um einen interessanten Parameterbereich für die Fensterlänge zu bekommen, betrachten wir Grafik 6. Die rote Kurve zeigt deutlich, dass ab einer Fensterlänge von drei die Anzahl der unterschiedlichen Fenster sprunghaft ansteigt. Die weiteren Untersuchungen werden daher besonders auf die Fensterlänge drei bis fünf konzentriert. Die blaue Kurve steht für die Anzahl der verschieden Fenster, die in dem Datensatz gefunden wurden, indem Aminosäuren durch ihre Gruppe ersetzt wurden. Wie durch die

geringere Anzahl der Buchstaben zu erwarten, sind hier deutlich längere Fenster noch mit einer moderaten Anzahl an verschieden Fenstern vertreten. Hier ist jedoch nur ein kontinuierlicher Anstieg zu beobachten, so dass wir hier unsere Untersuchungen auf die Längen fünf bis elf konzentrieren werden.

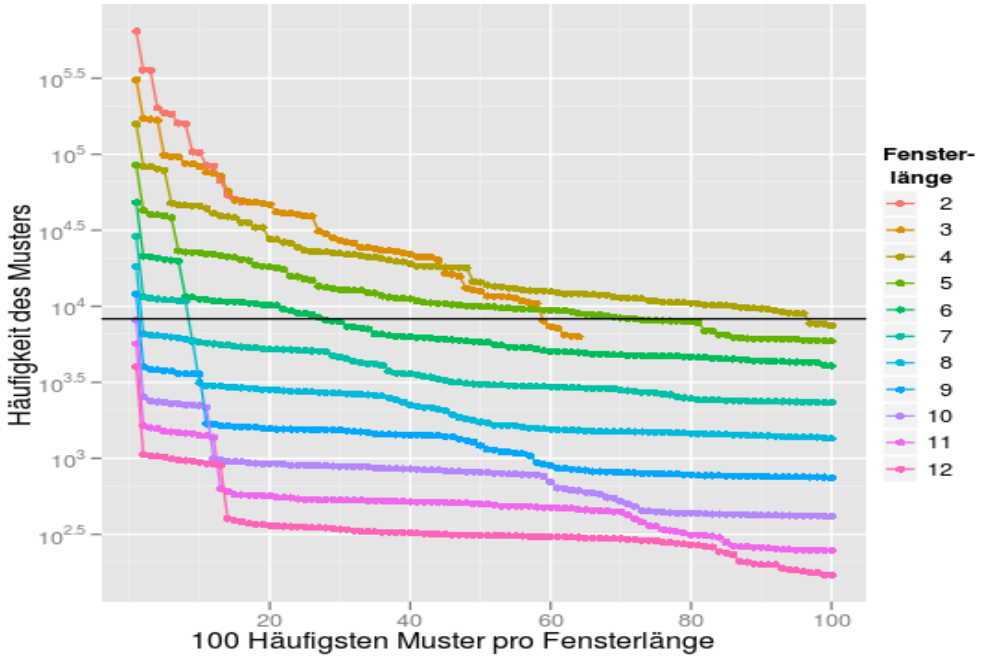


Grafik 7: Anzahl der Features die Häufiger als einmal auftreten

In Grafik 7 wurde die Anzahl der Fenster, die häufiger als einmal gefunden wurden, gegen die Fensterlänge aufgetragen. Ein Fenster, dass nur einmal Auftritt, kann kein Unterscheidungskriterium sein und ist daher für unsere Zwecke nutzlos. Wir suchen Features die Häufig genug vorkommen, damit durch diese Regeln für unseren Datensatz generiert werden können. Es ist daher von höchstem Interesse, dass für die Amino Sequenzen eine Fensterlänge größer als fünf zu niedrigeren Werten führt. Dies bedeutet, dass wenn die Fensterlänge länger als fünf gewählt wird, keine weiteren Information dazu gewonnen werden können, jedoch die dünn-Besetztheit des Problems verstärkt wird. Bei den Gruppen Sequenzen ist auch in diesem Diagramm nur ein kontinuierlicher Übergang zu sehen. In den Grafiken 8 und 9 sind die Häufigkeiten der 100 häufigsten Features dargestellt. Die schwarze horizontale Linie stellt die Anzahl der verschieden Sequenzen dar.



Grafik 8: 100 häufigsten Features für Aminosequenzen



Grafik 9: 100 häufigsten Features für Gruppensequenzen

5.3. Datenaufteilung

Nachdem wir eine Möglichkeit diskutiert haben, Features für unseren Datensatz zu generieren, betrachten wir nun die Response oder Target Daten. In Tabelle 1 ist aufgeführt, wie viele Sequenzen mit Parp14 und Parp10 gebunden (TRUE) und nicht gebunden (FALSE) haben. Es fällt auf, dass sehr viel mehr negative Beispiele vorhanden sind, als positive und dass Parp14 fast drei mal soviel positive Beispiele hat als Parp10. Im Folgenden werden wir daher unsere Betrachtungen auf Parp14 beschränken und die Ergebnisse auf Parp10 übertragen. Des Weiteren werden wir aus den negativen Beispielen die gleiche Anzahl Sequenzen ziehen (ohne zurücklegen), wie wir positive Beispiele haben. Dies ermöglicht uns Berechnungen durchzuführen, die sonst sehr lange Rechenlaufzeiten erfordern würden. Darüber hinaus ist das Interpretieren eines balancierten Datensatzes oft einfacher.

Total : 8273	**TRUE**	**FALSE**
Parp 10	59	8214
Parp 14	149	8124
Parp 10 oder Parp 14	176	8097
Parp 10 und Parp 14	32	8241

Table 1: Klassenhäufigkeiten

Jetzt muss noch eine Aufteilung des Datensatzes durchgeführt werden, da wir, um unsere Prognosefähigkeit zu überprüfen, Vorhersagen auf vorher nicht von dem Algorithmus gesehen Daten machen möchten. Wir legen daher 30 Prozent der Daten zur Seite um darauf die zu erwartende Performance unseres Modells festzustellen. Diesen Datensatz nennen wir im folgenden Testset, siehe auch Hastie, Tibshirani, und Friedman (2009, p. 222).

In Tabelle 2 und Tabelle 3 ist genau aufgeführt ,wie viele Beispiele wir in den einzelnen Datensätzen haben.

Parp 14 (70/30)	**TRUE**	**FALSE**	**Total**
Trainingsset	105	105	210
Testset	44	44	88

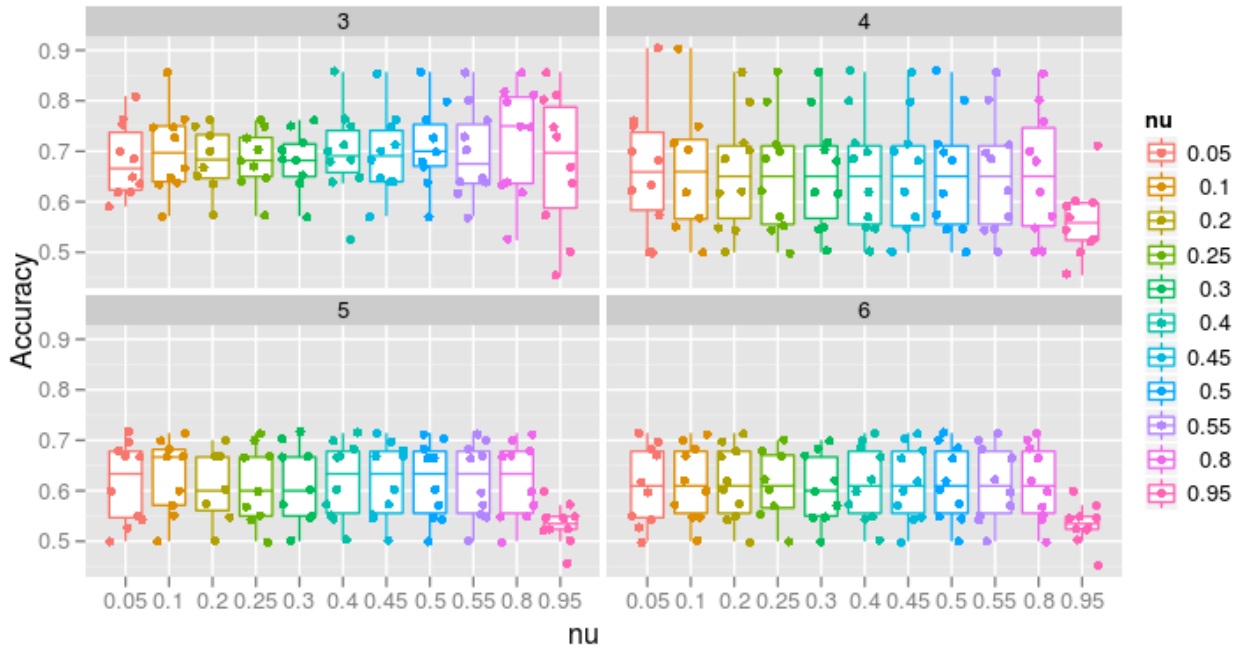
Table 2: Parp 14

Parp 10 (70/30)	**TRUE**	**FALSE**	**Total**
Trainingsset	42	42	84
Testset	17	17	34

Table 3: Parp 10

5.4. Parameteroptimierung

Als Vorhersagemodell wurde eine v-SVM mit einem Spektrumkernel gewählt. Es stehen daher zwei Parameter zur Verfügung um das Modell an die Daten anzupassen. Zum einen den Strafterm **v** für die SVM und **k** für die Länge des Spekrumkernels. Zur Auswertung wurde das in Tabelle 2 definierte balancierte Dataset verwendet. Um diesen relativ kleinen Datensatz effizient zu nutzen, wurde eine 10-fache Crossvalidation durchgeführt (Hastie et al., 2009, p. 242) .



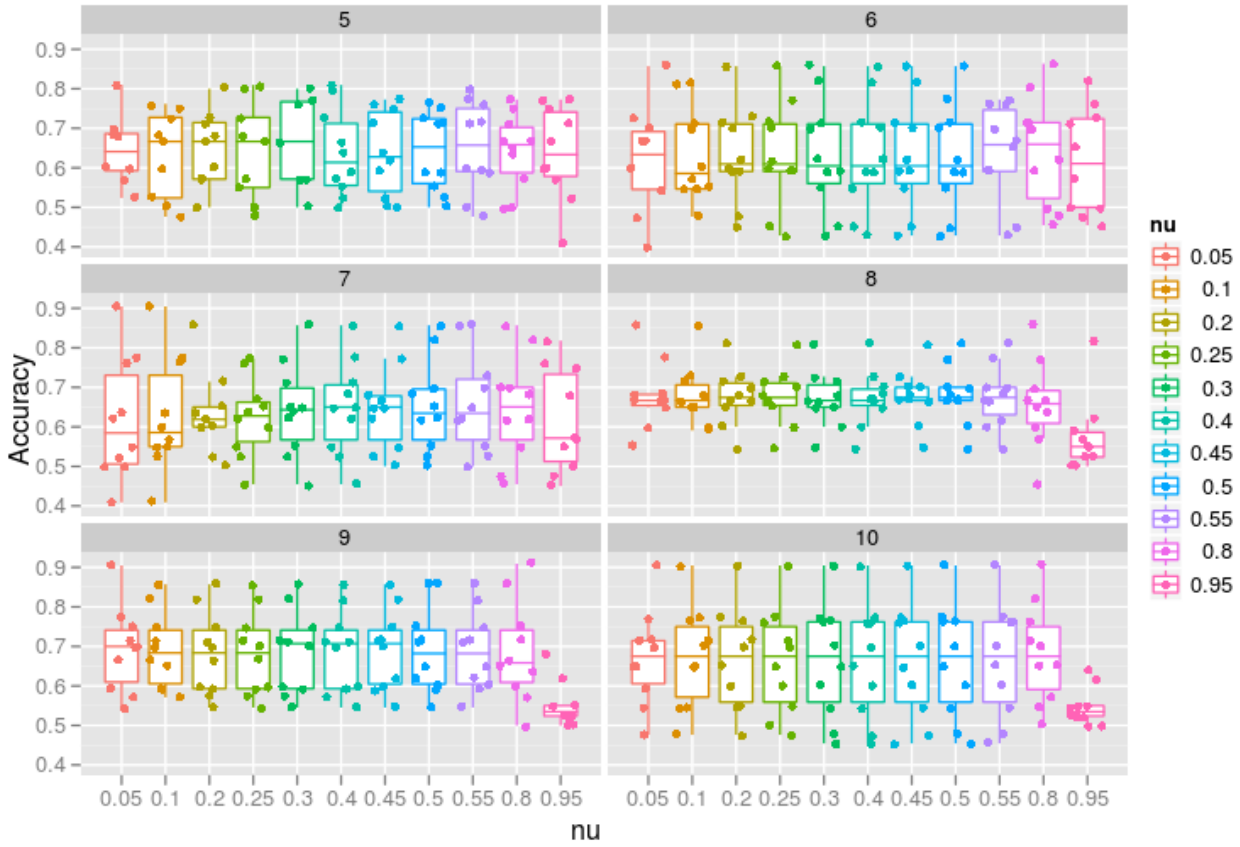
Grafik 10: Parameteroptimierung für Parp14 auf Aminosequenzen

In Grafik 10 und 11 sind die Ergebnisse der Parametersuche für die Amino- und Gruppendaten grafisch dargestellt. Als Maß für die Qualität des Modells wurde die Accuracy gewählt.

$$Accuracy = \frac{(true\ positive + true\ negative)}{(positive + negative)} = \frac{(TP + TN)}{(P + N)} \tag{10}$$

Die Zahl in dem grau hinterlegten Balken, in den Grafiken, gibt die Länge des verwendeten Spektrumkernels an. Für jede Länge wurde der Parameter **v** jeweils von 0.05 bis 0.95 mit einer Schrittweite von 0.05 variiert. Jeder Boxplot fast die

jeweiligen Werte der Crossvalidierung zusammen. Die einzelnen Datenpunkte zeigen die jeweiligen Accuracy der Crossvalidierungsläufe auf.



Grafik 11: Parameteroptimierung für Parp14 auf Gruppensequenzen

Eine visuelle Auswertung von Grafik 10 zeigt für k = 3 die durchschnittlich höchste Accuracy. Dies trifft mit unseren Beobachtungen von Grafik 6 zusammen. Es kann daher angenommen werden, dass auch die nicht-getesteten Werte größer drei nicht interessant sind. Für den SVM Parameter v zeigt der Bereich von 0.2 bis 0.3 für v die geringste Varianz. Wir wählen daher v = 0.25 für das Modell.

In Grafik 11 sind die Ergebnisse für die Crossvalidierungsläufe auf den Gruppendaten zu sehen, hier zeigt sich ein anderes Bild als in Grafik 10. Die beste Durchschnitte Accuracy wird für eine Fensterlänge von k= 9 erreicht. Auch dieses Ergebnis ist konsistent mit den in Grafik 11 aufgeführten Überlegungen. Der Parameter v scheint für k = 9 keine besondere Rolle zu spielen. Wir legen ihn daher auf 0.4 fest.

6. Modell Auswertung

Im Folgenden werden wir genauer auf die Crossvalidierungsergebnisse der zwei besten Modelle eingehen. Es ist hier mit einem Selektionsbias zu rechnen da wir im Nachhinein die Performance der besten Modelle betrachten. Die zu erwartende Prognosefähigkeit wird später auf dem Testset festgestellt. Zu dem v-SVM Modell wurde ein Wahrscheinlichkeitsmodel trainiert, wie es in Abschnitt SVM Wahrscheinlichkeitsschätzung vorgestellt wurde. Dies ermöglicht es, die Bindungswahrscheinlichkeit, die die Modelle der jeweiligen Sequenz zugeordnet haben, zu vergleichen. Anhand der Farbe der Datenpunkte sind die echten Klassenzugehörigkeiten zu erkennen.

Grafik 12: Scatterplot Model Amino vs. Model Gruppen

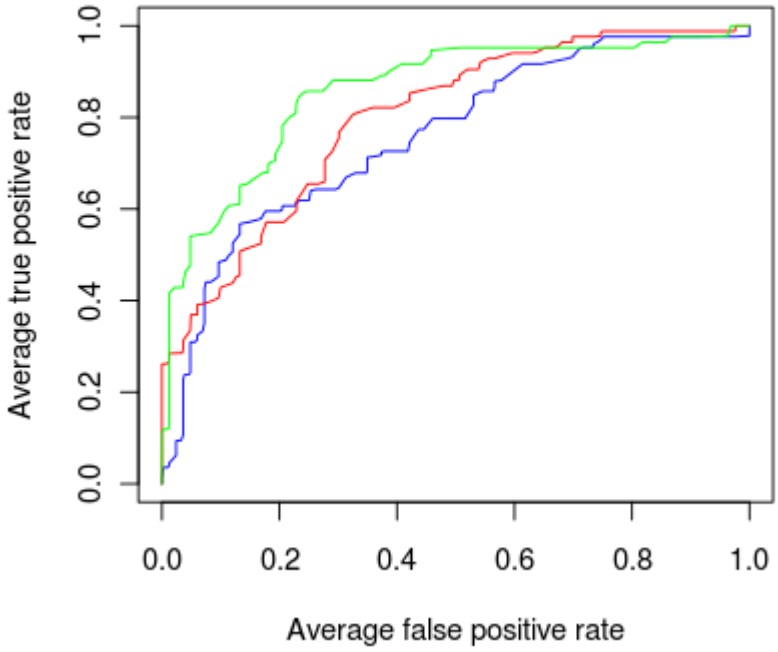
Eine qualitative Betrachtung und die im Nachfolgenden aufgeführten Statistiken zeigen, dass die Modelle für sich betrachtet eine Vorhersagekraft haben die

deutlich über dem *no information ratio* liegen. Des Weiteren fällt auf, dass die beiden Modelle eine geringe Abhängigkeit von 0.249 bei einem P-Wert (*Pearson's product-moment correlation*) von 0.001157 haben. Dies macht es attraktiv die beiden Modelle zu kombinieren, mit dem Ziel, dass falsche Aussagen abgeschwächt und Richtige verstärkt werden. In der Tat ist es genau diese Eigenschaft, die man bei Ensemble Methoden zu erreichen versucht.

```
Parp 14 Amino
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE    60   30
     TRUE     23   54

               Accuracy : 0.6826
                 95% CI : (0.6063, 0.7524)
    No Information Rate : 0.503
    P-Value [Acc > NIR] : 1.953e-06


Parp14 Gruppe

Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE    61   26
     TRUE     22   58

               Accuracy : 0.7126
                 95% CI : (0.6376, 0.7799)
    No Information Rate : 0.503
    P-Value [Acc > NIR] : 2.94e-08
```

Wenn wir also die Klassenwahrscheinlichkeiten als Posteriori Wahrscheinlichkeiten interpretieren und eine stochastische Unabhängigkeit annehmen, können wir das Produkt der Wahrscheinlichkeiten bilden und als Gesamtklassenwahrscheinlichkeit betrachten. Andere Möglichkeiten, wie das *Klassifikator staking* währen auch möglich (für mehr Details zu anderen Verfahren siehe Polikar, 2006).

Anhand der ROC Kurve REF können die Eigenschaften genauer analysiert werden. Fawcett (2003) und Flach (2003) sind gute Quellen zu Anwendung und Interpretation der ROC Kurve.)



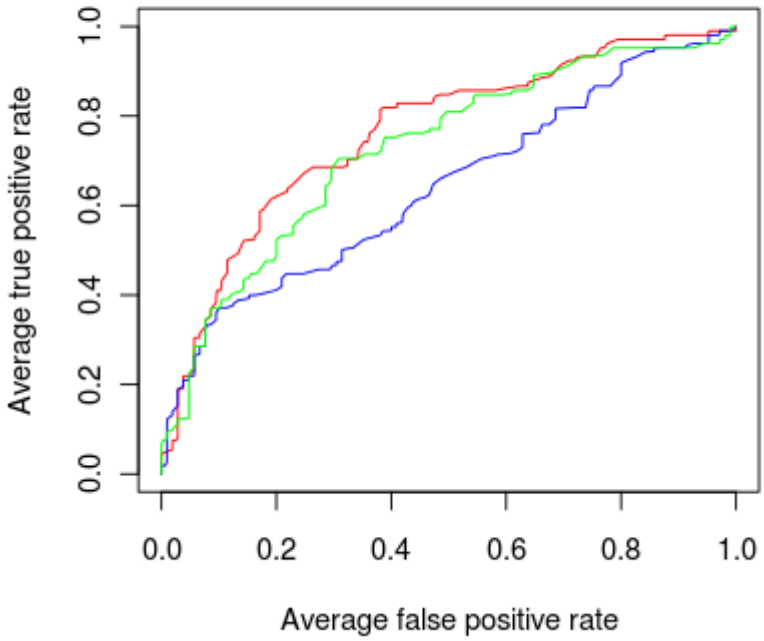
Grafik 13: Rot Amino, Blau Gruppe und Grün: Kombination

7. Dimension Reduktion

Wie bereits mehrfach diskutiert, hat die Feature Generierung durch das schieben von Fenstern den Nachteil, dass eine sehr hohe Anzahl an Features entsteht und diese zudem Großteils dünn besetzt sind. Hochdimensionale Daten stellen grundsätzlich eine Schwierigkeit für die Interpretation der Ergebnisse da, den der Einfluss der einzelnen Features ist schwierig nachzuvollziehen. Um die Dimension der Daten zu reduzieren, wurden daher zwei verschiedene Ansätze getestet. Damit Einfluss auf die verwendeten Fenster genommen werden konnte, wurde diesmal nicht der Spectrum Kernel verwendet, sondern die Feature Generierung durch das verschieben eines Fenster selbst implementiert. Dafür wurde wieder ein v-SVM (v = 0.25) mit einem linearen Kernel (vanilladot) verwendet. Die Reduzierung wurde nur auf der original Kodierung, also mit den Aminosäuren getestet.

7.1. Häufigkeit der Features

In dem ersten Ansatz haben wir nur Features zugelassen die eine gewissen Mindesthäufigkeit und Maximalhäufigkeit auf dem Trainingsset aufweisen. Die rote ROC Kurve in Grafik 14 zeigt die Ergebnisse für eine 10x Crossvalidation bei einer Mindesthäufigkeit von 10 und Maximalhäufigkeit von 25. Die Ergebnisse stellen keine Verbesserung zu dem Modell mit dem vollen Featuresatz da, liefern jedoch eine interessante Interpretation. Die Prognosequalität wird nicht beeinflusst, wenn nur Features verwendet werden, die nur 25 mal im gesamten Datensatz auftreten. Berücksichtigt man, dass sich 105 positive Beispiele in dem Datensatz befinden, beutet dies, dass es kein einzelnes Feature gibt, welches den ganzen Datensatz erklärt.



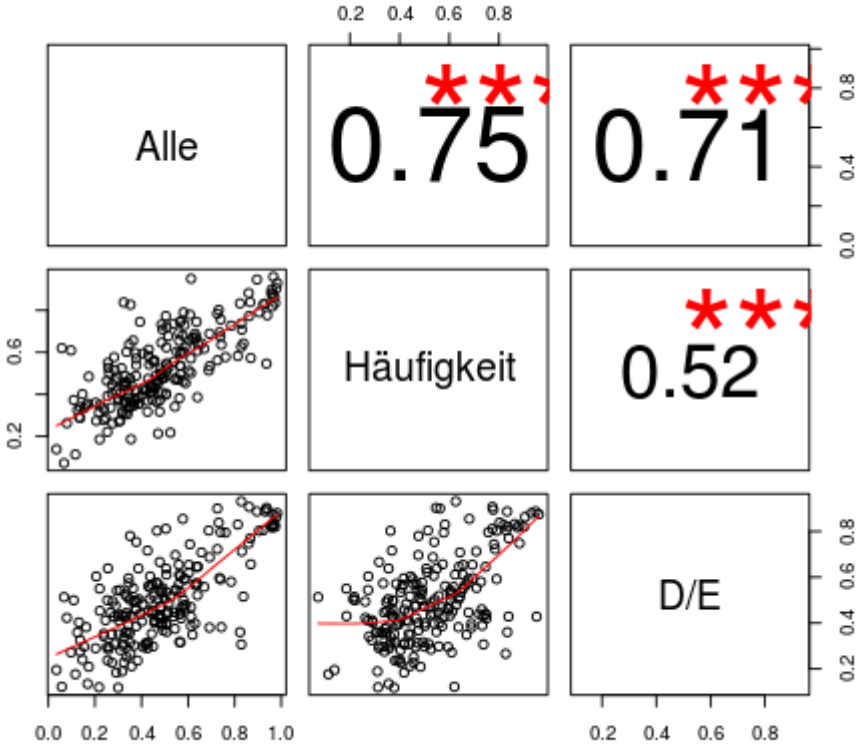
Grafik 14: Rot = Original, Grün = Häufigkeit, Blau = DE

7.2. Features mit D/E

Durch ein Gespräch mit dem Urheber der Daten, wurde die Überlegung geäußert, dass Features die ein D oder E enthalten biologisch besondere Relevanz für die Bindungsfähigkeit haben. Es wurden daher in einem weiteren Versuch nur Features verwendet, die auch die mit D oder E kodierte Aminosäure enthalten. Die

Ergebnisse sind in Grafik 14 als blaue Kurve zu sehen. Die Vorhersagbarkeit hat im Vergleich mit dem vollständige Featuresatz deutlich nachgelassen.Für unseren Ansatz macht somit eine Reduktion nur auf Features, die ein D oder E enthalten kein Sinn.

```
Confusion Matrix and Statistics Rot

          Reference
Prediction FALSE TRUE
     FALSE    80   39
     TRUE     25   66

               Accuracy : 0.6952
                 95% CI : (0.6282, 0.7567)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 7.546e-09


Confusion Matrix and Statistics Grün (min = 10,max = 25)

          Reference
Prediction FALSE TRUE
     FALSE    74   33
     TRUE     31   72

               Accuracy : 0.6952
                 95% CI : (0.6282, 0.7567)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 7.546e-09

Confusion Matrix and Statistics Blau

          Reference
Prediction FALSE TRUE
     FALSE    67   49
     TRUE     38   56

               Accuracy : 0.5857
                 95% CI : (0.5159, 0.6531)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.007762
```



Grafik 15: Korrelation der Prognosen, basierend auf unterschiedlicher Features-Auswahl

8. Ergebnisse auf dem Testset

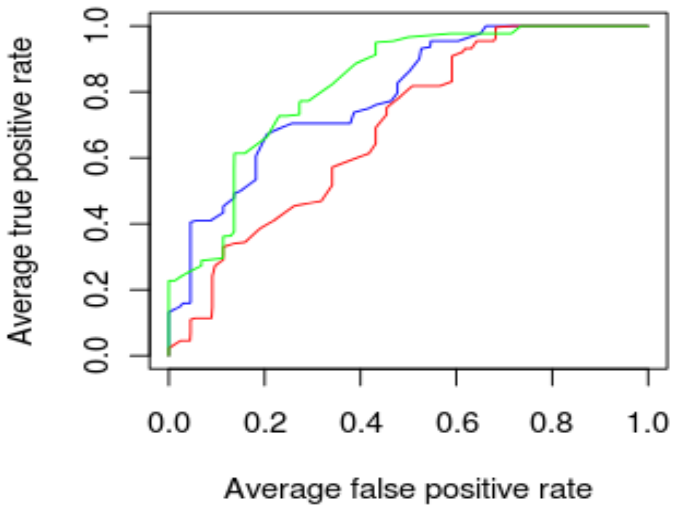
In diesem Abschnitt werden die Modelle auf den am Anfang zur Seite gelegten Daten getestet. Es gilt zu beachten, dass besonders für Parp10 Daten der Testset recht klein ist und daher eine größere Abweichungen zu den hier erreichten Ergebnissen zu erwarten sind. Tabelle 4 zeigt eine Zusammenfassung der Ergebnisse. Im Folgenden wird die Performance im Detail anhand von ROC Kurven diskutiert. Dabei steht blau wieder für die Amino-Kodierung, rot für Gruppen-Kodierung und grün für die Kombination der Modelle.

AUC	**Amino (Blau)**	**Gruppe (Rot)**	**Gemeinsam(Grün)**
Parp10	0.8408304	0.7474048	0.8269896
Parp14	0.8580868	0.879345	0.8652686
Parp10trainParp14	0.7892562	0.6921488	0.8254132

Table 4: Testset AUC Übersicht

8.1. Ergebnisse für Parp14

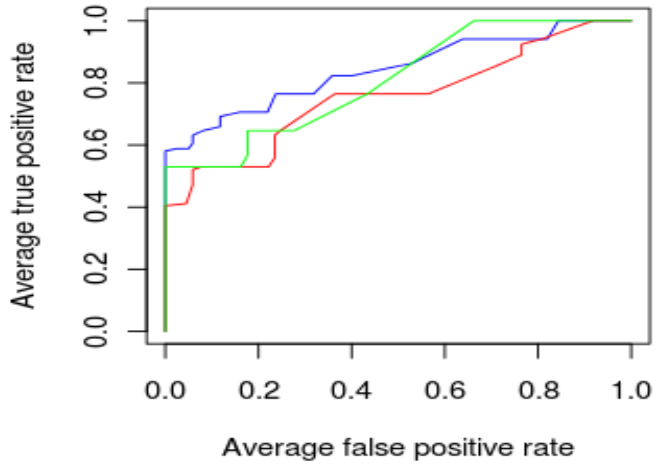
Die gesamt Performance entspricht den Werten, die wir durch die CV Auswertung erwarten konnten. Auch wenn hier die AUC-Werte in Tabelle 4 für die Kombination der Modelle unter denen der Einzelergebnisse liegen, so lässt ich an der ROC Ḱurve in Grafik 16 über weite Teile eine Verbesserung der Performance feststellen. Interessant ist, dass im oberen Teil der Kurve ein vollständig horizontaler Bereich zu sehen ist. Dies bedeutet, dass das Modell für ein Teil der Daten keine Vorhersage treffen kann.



Grafik 16: ROC Kurve für die Performance von Parp 14 auf dem Testset

8.2. Ergebnisse für Parp10

Wie wir in Tabelle 3 sehen können, beinhaltet das Testset für Parp10 nur 34 Sequenzen. Daher ist eine detaillierte Betrachtung von der ROC Kurve nur eingeschränkt aussagekräftig. Es ist jedoch dem vertikalen Anstieg am Anfang der Kurve zu entnehmen, dass ein Teil der Proteine sehr genau vorhergesagt werden kann. Die Vorteile durch die Kombination der Modelle sind hier weniger ausgeprägt im Vergleich zu Parp 14.



Grafik 17: ROC Kurve für die Performance von Parp 10 auf dem Testset

9. Modell auf Parp14 trainiert und auf Parp10 getestet

Hier wurde das Modell auf dem Trainings- und Testset von Parp14 aus Tabelle 2 trainiert und auf dem Trainings- und Testset von Parp10 ausgewertet. Dies kann uns als Interpretation dienen, wie sehr sich die Bindungsmechanismen bzgl. unseres Modells für Parp14 und Parp10 unterscheiden. Auffällig ist, dass der AUC Wert der Modellkombination in Tabelle 4 sich quasi nicht von dem Wert auf dem reinen Parp10 Datensatz unterscheidet. Interessant ist auch, dass hier zum ersten Mal die ROC Kurve des Gruppenmodells über der des Aminomodells liegt. Im Ganzen betrachtet, ist die erzielte Performance jedoch schlechter, als die der Modelle, die für das gleiche Protein trainiert und getestet wurden. Für die hier untersuchten Modelle sind die Bindungsmechanismen in Parp14 und Parp10 ähnlich, aber nicht gleich.

10. Fazit und Ausblick

Es wurde gezeigt, dass die Bindungsfähigkeit von Proteinsequenzen mit Parp 10 und Parp 14 mit Hilfe von Stringkernel-SVM Modellen vorhersagbar ist. Es wurde deutlich, dass die Umkodierung der Aminosäuren in Aminosäuregruppen vorteilhaft sein kann und eine Kombination von Modellen, basierend auf unterschiedlichen Kodierungen, lohnenswert ist. Es ist anzunehmen, dass das Modell für die Vorhersage von Parp 10, durch eine höhere Anzahl von bindenden Proteinsequenzen, verbessert werden kann. Eine weitere Kodierungsmöglichkeit für die Proteinsequenzen wurde in Heider & Hoffmann (2011) vorgestellt. Für weitere Analysen wäre es interessant, damit eine weiteres Modell zu erstellen und dann auf der Ausgabe aller drei Modelle einen Metaklassifier zu trainieren. Des Weiteren sollte der Einfluss des in Yu et al. (2010) beschriebenen Problems von Hub Proteinen auf die Ergebnisse untersucht werden.

11. Implementierung

Alle Auswertungen wurden mit der Skript Sprache R (R Development Core, 2010) programmiert für die einzelnen Aufgaben wurden zahlreiche Pakete benutzt. Für die Modellierung wurden die SVM Implementierungen von Paket kernlab (Karatzoglou, Smola, Hornik, & Zeileis, 2004) verwendet, für die Auswertung und ROC Kurven wurden die Packete caret (Wing, Weston, Williams, Keefer, & Engelhardt, 2011) und ROCR (Sing, Sander, Beerenwinkel, & Lengauer, 2009) benutzt. Für die meisten Grafiken wurde das Paket ggplot2 (Wickham, 2009) genutzt.

11.1. R und Paket Versionen

```
R version 2.12.0 (2010-10-15)
Platform: i486-pc-linux-gnu (32-bit)

locale:
 [1] LC_CTYPE=en_US.utf8          LC_NUMERIC=C
 [3] LC_TIME=en_US.utf8           LC_COLLATE=en_US.utf8
 [5] LC_MONETARY=en_US.utf8       LC_MESSAGES=en_US.utf8
 [7] LC_PAPER=en_US.utf8          LC_NAME=en_US.utf8
 [9] LC_ADDRESS=en_US.utf8        LC_TELEPHONE=en_US.utf8
[11] LC_MEASUREMENT=en_US.utf8    LC_IDENTIFICATION=en_US.utf8

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] ggplot2_0.8.9   proto_0.3-9.2   caret_4.88      cluster_1.13.2
 [5] reshape_0.8.3   plyr_1.5.2      lattice_0.19-13 ROCR_1.0-4
 [9] gplots_2.8.0    caTools_1.10    bitops_1.0-4.1  gdata_2.8.0
[13] gtools_2.6.2    kernlab_0.9-10  rj_0.5.0-5

loaded via a namespace (and not attached):
[1] rJava_0.8-7  tools_2.12.0
```

Amino acid - Wikipedia, the free encyclopedia. (n.d.). Retrieved September 14, 2011, from http://en.wikipedia.org/wiki/Amino_acid

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), 121–167.

Chen, P. H., Lin, C. J., & Schölkopf, B. (2005). A tutorial on ν-support vector machines. *Applied Stochastic Models in Business and Industry*, *21*(2), 111-136.

Fawcett, T. (2003). ROC graphs: Notes and practical considerations for data mining researchers. *HP Laboratories technical report*.

Flach, P. A. (2003). The geometry of ROC space: understanding machine learning metrics through ROC isometrics. *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-* (Vol. 20, p. 194).

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning : data mining, inference, and prediction* (2nd ed.). New York: Springer.

Heider, D., & Hoffmann, D. (2011). Interpol: An R package for preprocessing of protein sequences. *BioData mining*, *4*(1), 16.

Karatzoglou, A., Smola, A., Hornik, K., & Zeileis, A. (2004). kernlab – An S4 Package for Kernel Methods in R. *Journal of Statistical Software*, *11*(9), 1–20.

Leslie, C. S., Eskin, E., Cohen, A., Weston, J., & Noble, W. S. (2004). Mismatch string kernels for discriminative protein classification. *Bioinformatics*, *20*(4), 467.

Lin, H. T., Lin, C. J., & Weng, R. C. (2007). A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, *68*(3), 267-276.

Platt, J. (n.d.). Probabilistic Outputs for SVMs and Comparisons to Regularized Likelihood Methods.

Polikar, R. (2006). Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE*, *6*(3), 21–45.

R Development Core, T. (2010). *R: A Language and Environment for Statistical Computing*. Vienna, Austria. Retrieved from http://www.R-project.org/

She, R., Chen, F., Wang, K., Ester, M., Gardy, J. L., & Brinkman, F. S. L. (2003). Frequent-subsequence-based prediction of outer membrane proteins. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 436–445).

Sing, T., Sander, O., Beerenwinkel, N., & Lengauer, T. (2009). *ROCR: Visualizing the performance of scoring classifiers.* Retrieved from http://CRAN.R-project.org/package=ROCR

Wickham, H. (2009). *ggplot2: elegant graphics for data analysis*. Springer New York. Retrieved from http://had.co.nz/ggplot2/book

Wing, M. K. C. from J., Weston, S., Williams, A., Keefer, C., & Engelhardt, A. (2011). *caret: Classification and Regression Training*. Retrieved from http://CRAN.R-project.org/package=caret

Xing, Z., Pei, J., & Keogh, E. (2010). A brief survey on sequence classification. *ACM SIGKDD Explorations*, *12*(1), 40–48.

Yu, J., Guo, M., Needham, C. J., Huang, Y., Cai, L., & Westhead, D. R. (2010). Simple sequence-based kernels do not predict protein–protein interactions. *Bioinformatics*, *26*(20), 2610.

IV. Suche eines Dockingmusters

In diesem Teil wird die suche eines Dockingmusters erklärt. Dockingmuster haben wir auf zwei verschiedene Arten gesucht: mittels genetischer Algorithmen und mittels vollständiger Suche. In beiden Fällen geht es darum, ein oder mehrere Fenster in der Proteinsequenz zu finden, welche besonders häufig in den positiven Sequenzen und besonders selten in den negativen Sequenzen auftreten.

Solche Fenster werden durch Muster beschrieben. Ein Muster besteht aus einer Kette von Elementen. Ein solches element kann eine Aminosäure, eine Aminogruppe, eine oder-Kombination mehrerer Aminosäuren oder ein Wildcard (_) sein. Ein Wildcard bedeutet, dass an dieser Stelle des Fensters eine beliebige Aminosäure stehen darf. Diese Elemente können beliebig kombiniert werden. Im folgenden werden einige Beispielmuster zur Verdeutlichung erklärt:

- `GNDRHL`: Dieses Muster besteht ausschließlich aus Aminosäuren, entspricht also nur dem Fenster `GNDRHL`. Wir nennen solch ein Muster Muster der ersten Form.
- `GNJRHL`: Dieses Muster ist das selbe wie das Obige, jedoch wurde das `D` durch ein `J` ersetzt. `J` ist die Gruppe der sauren Aminosäuren, steht also sowohl für `D` als auch für `E`. Somit entspricht dieses Muster den Fenstern `GNDRHL` und `GNERHL`. Wir nennen solch ein Muster Muster der zweiten Form.
- `GN_RHL`: In diesem Muster wurde das `D` durch ein Wildcard ersetzt. Das bedeutet, dass an der Stelle des Wildcards eine beliebige Aminosäure stehen darf. Somit entspricht dieses Muster insgesamt 20 verschiedenen Fenstern. Wir nennen solch ein Muster Muster der dritten Form.
- `GN[A,T,D]RHL`: In diesem Muster wurde das `D` durch eine oder-Kombination der Aminosäuren `A`, `T` und `D` ersetzt. Somit entspricht dieses Muster den Fenstern `GNARHL`, `GNTRHL` und `GNDRHL`. Wir nennen solch ein Muster Muster der vierten Form.

1. Mittels vollständiger Mustersuche

L. Beyer

In der vollständigen Mustersuche haben wir versucht, alle möglichen Muster einer bestimmten Länge aufzuzählen, zu bewerten und nach dieser Bewertung zu sortieren.

1.1. Suchraumanalyse

Im folgenden wird analysiert, wie sich der Suchraum in Bezug auf die Musterkomplexität und die Fensterlänge verhält.

Muster der ersten Form

Betrachtet man nur die Muster, die ausschließlich aus Aminosäuren bestehen, stellt man fest, dass die Anzahl möglicher Muster durch

$$N = n_a{}^w \tag{11}$$

gegeben ist, wobei $n_a = 20$ für die Anzahl verschiedener Aminosäure und w für die Fensterlänge stehen. Tabelle 1 zeigt die Anzahl möglicher Muster für einige Fensterlängen an.

w	3	4	5	6	7	8	9	10	11	12
N	8000	1.6e5	3.2e6	6.4e7	1.28e9	6.4e10	1.28e12	2.56e13	5.12e14	1.024e16

Table 5: Mögliche Muster für verschiedene Fensterlängen

Daraus wird ersichtlich, dass es nicht möglich ist, alle Kombinationen aufzuzählen.

Muster anderer Formen

Weiterhin ist es von Interesse, Muster der zweiten und dritten Form zu betrachten. In diesem fall, gibt es für jedes Muster der ersten Form genau 2^w mögliche Kombinationen, die Aminosäuren durch ihre Gruppen oder ein Wildcard zu ersetzen. Also ist die gesamte Musteranzahl für Muster der zweiten und dritten Form gegeben durch:

$$N_2 = N_3 = N_1 \cdot 2^w = (2 \cdot n_a)^w . \quad (12)$$

Möchte man nun Muster der zweiten und dritten Form kombinieren, ergibt sich die gesamte Musteranzahl als:

$$N_{2,3} = N_1 \cdot 3^w = (3 \cdot n_a)^w . \quad (13)$$

Weiterhin ist die Anzahl möglicher Muster der vierten Form durch

$$N_4 = \prod_{i=0}^{w} \frac{n_a !}{(n_a - s_i) !} \quad (14)$$

gegeben, wobei s_i für die größe der i-ten Gruppe steht.

1.2. Einschränkungen

Nach näherer Betrachtung ist es jedoch nicht nötig, all diese Muster aufzuzählen, da folgende Einschränkungen gelten:

- es kommen aus biologischen Gründen nur Muster in Frage, die mindestens ein D oder ein E enthalten und
- es brauchen nur die Muster betrachtet zu werden, die mehr als einmalig in den positiven Proteinen vorkommen.

Nach Anwendung dieser beiden Betrachtungen könnte sich der Suchraum, abhängig von dem Dateninhalt, stark reduzieren.

1.3. Bewertung eines Musters

Zur Bewertung der Muster haben wir die Trennfähigkeit wie folgt definiert:

$$t(m) = f_p - f_n = P(m|p) - P(m|n) , \quad (15)$$

wobei $f_p = P(m|p)$ ausdrückt, mit welcher Häufigkeit das Muster m in den positiven Proteinen vorkam, also wie wahrscheinlich mit der ein positives Protein das Muster m enthält. Analog ist $f_n = P(m|n)$ für die negativen Proteine definiert.

Der Wertebereich dieser Bewertungsfunktion ist durch $\mathrm{Im}(t) = [-1, 1]$ gegeben. Die bestmögliche Bewertung liegt somit bei $t(m) = 1$ was bedeutet, dass dieses Muster in jedem positiven Protein und in keinem negativen Protein vorhanden ist. Eine Bewertung von $t(m) = -1$ ist genauso interessant denn sie bedeutet, dass

dieses Muster ausschließlich in negativen Proteinen vorkam. Eine Bewertung von $t(m) \approx 0$ bedeutet jedoch, dass dieses Muster irrelevant ist.

1.4. Umsetzung

Umgesetzt wurde die vollständige Mustersuche in einem C++11-Programm, welches folgende Grundstruktur aufweist:

```
1. pos = Positive Fenster der Länge N um D/E
2. neg = Negative Fenster der Länge N um D/E
3. kombis = Kartesisches Produkt aus g,a,_ der Länge N
4. For kombination in kombis:
5.      npos = transformiere (pos, kombination)
6.      nneg = transformiere (neg, kombination)
7.      For Muster in npos:
8.          p = Anteil positiver Proteine in denen das Muster vorhanden ist
9.          n = Anteil negativer Proteine in denen das Muster vorhanden ist
10.         scores[Muster] = p - n
11. Sort scores
```

Listing 1: Grundstruktur des Programms zur vollständigen Mustersuche

Wobei die Funktionalität der `transformiere`-Funktion durch folgenden Pseudocode beschrieben werden kann:

```
1. Funktion transformiere(fensterliste, transformationsmuster):
2.      neuefenster = []
3.      For fenster in fensterliste:
4.          neuesfenster = ''
5.          For i in length(transformationsmuster):
6.              If transformationsmuster[i] == g:
7.                  neuesfenster += aminogruppe(fenster[i])
8.              Elif transformationsmuster[i] == _:
9.                  neuesfenster += _
10.             Else:
11.                 neuesfenster += fenster[i]
12.         add neuesfenster to neuefenster
13.     return neuefenster
```

Listing 2: Pseudocode für die transformationsfunktion

Dadurch wird eine Datenstruktur aufgebaut, in der man für eine bestimmte Fensterlänge schnell herausfinden kann, wie häufig ein Muster in einem Satz Fenster vorkommt.

Wir haben aus Zeit- und Laufzeitgründen nur nach Mustern des ersten, zweiten und für kleine w des zweiten und dritten Typs gesucht.

Als weitere Suchraumreduktion haben wir nur die (transformierten) Fenster betrachtet, die in mehr als nur einem positiven Protein vorkamen. Aus diesem Grund können wir keine Werte nahe -1 für t erwarten.

1.5. Ergebnisse

Da uns gesagt wurde, dass aus biologischen Gründen Fenster der Länge 8 bis 12 am sinnvollsten sind, haben wir die Ergebnisse auf diese Längen beschränkt. Andere Längen sind jedoch ohne weiteres durch das Programm berechenbar. Als Ergebnisse haben wir die bestbewerteten Muster für alle Fensterlängen von 8 bis 12 in den Teilen VI bis IX (Seiten 60 bis 72) zusammengefasst.

Es ist zu erkennen, dass ein gewisser Anteil der Muster deutlich häufiger in den positiven als in den negativen Proteinen vorkommt.

Um diese Ergebnisse besser deuten zu können, leiten wir ein weiteres Maß anhand der Bayes-Regel her:

$$P(p|m)=\frac{P(m|p)\cdot P(p)}{P(m)}=\frac{P(m|p)\cdot P(p)}{P(m|p)\cdot P(p)+P(m|n)\cdot P(n)}\,. \qquad (16)$$

$P(p|m)$ beschreibt die Wahrscheinlichkeit, mit der ein Protein welches das Muster m enthält positiv ist, d.H. clustert. $P(p)$ und $P(n)$ sind jeweils die Häufigkeit positiver respektive negativer Proteine.

Subsampling

Da bei der Verwendung des gesamten negativen Datensatzes $P(p)=0.007489$ und $P(n)=0.99251$ für PARP10 gelten (ähnliche Zahlen für PARP14), ist dieses Maß nur dann sinnvoll, wenn ungefähr die gleiche Menge negativer wie auch positiver Proteine verwendet werden, so dass $P(n)\approx P(p)\approx 0.5$ gilt. Um dies zu erreichen haben wir zufällig so viele negative Proteine aus dem Datensatz (ohne Widerholung) gezogen wie positive Proteine vorhanden waren.

Erstaunlicherweise sind die Ergebnisse bis auf geringe Schwankungen gleich geblieben. Somit ist gezeigt, dass das Bewertungsmaß $t(m)$ stabil ist. Das maß $P(p|m)$ ist weniger stabil, da wenn ein Muster in keinem der zufällig gewählten negativen Proteine vorkam $P(p|m)=1$ berechnet wurde, was jedoch mit einer anderen zufälligen Wahl einen anderen Wert hätte.

Betrachtet man z.B. das durch $t(m)$ bestbewertete Muster der Länge 8 im PARP10 Subsampling Datensatz (siehe VIII.Bestbewertete Parp10 Muster mit subsampling, Seiten 68ff.) sieht man, dass eine Wahrscheinlichkeit von über 80% erreicht werden kann.

Diese Werte sind jedoch mit Vorsicht zu betrachten, da aus einer rein Machine-Learning bezogenen Sichtweise hier nicht zwischen Trainings- und Testset unterschieden wurde. Dies ist jedoch bewusst gemacht worden, da Vorhersage kein ziel dieses Ansatzes ist.

Diskussion der Ergebnisse

Aus Platzgründen können wir in diesem Dokument nur Ausschnitte der Ergebnisse zeigen, die gesamten Ergebnisse und die Programme um diese zu berechnen werden jedoch diesem Dokument beigefügt.

PARP10

Auffällig ist bei den Ergebnissen für PARP10, dass bei einer Musterlänge 12 eine Musterfamilie die Ergebnisse mit einem konstanten Wert $t(m)=0.14$ dominiert. Es würde sich sicherlich lohnen, die Redundanz der Muster dieser Familie zu untersuchen und diese Muster mit `and` bzw. `or` zu verknüpfen.

Verringert man die Musterlänge kristallisieren sich einige Muster heraus die eine bessere Bewertung als die Muster der oben genannten Familie haben.

Weiterhin deuten viele der gut bewerteten Muster darauf hin, dass sie Teil eines größeren Musters sind, da sie « verschoben » mehrmals auftauchen, wie z.B. an dem Muster in Table 2 deutlich zu erkennen ist.

Muster	P(m|p)	P(m|n)	t(m)
JXXZXXXX	0.316667	0.115699	0.200967
XXJXXZXX	0.333333	0.127673	0.205661
XXXJXXZX	0.333333	0.124007	0.209326
DXXZXXXX	0.283333	0.0631643	0.220169
XXXEXXZX	0.3	0.07416	0.22584

Table 6: « Verschobenes » Muster der Länge 8 in den PARP10 Ergebnissen ohne subsampling

PARP14

Auch hier können die selben Beobachtungen wie bei PARP10 gemacht werden.

Viele der Muster (wie z.B. `XXXJXXZXX`) die bei PARP10 besonders gut abgeschnitten haben, haben auch bei PARP14 sehr gut abgeschnitten. Dies lässt zu vermuten, dass die für das Clustering verantwortlichen Muster ähnlich, wenn nicht sogar gleich sind.

1.6. Fazit und Ausblick

Es wurde gezeigt, dass sich mit dem hier eingeführten Verfahren möglich ist, Muster zu entdecken, die sehr häufig in den positiven Proteinen jedoch nur selten in den negativen Proteinen vorkommen zu entdecken.

Es wurde in dieser Arbeit nicht nach Mustern mit $t(m)=-1$ gesucht, was jedoch noch viele Möglichkeiten eröffnet.

Da eine Vielzahl interessanter Muster gefunden wurden, bietet es sich an diese noch weiter zu untersuchen um mögliche Geistermuster und Redundanzen zu entdecken und eliminieren.

Weiterhin könnte man diese Muster mittels `and`, `or` bzw. `not` verknüpfen und somit ein Regelwerk schaffen, welches eine hohe Aussagekraft und einfache Deutung haben könnte. Eine Möglichkeit hierfür bieten Decision-Trees.

Noch eine weitere möglichkeit wäre es, Muster unterschiedlicher Längen miteinander zu vergleichen, z.B. um zu erkennen ob diese sich überlappen oder gegenseitig beinhalten.

2. Mittels genetischer Algorithmen

S. Manodumrongthum, L. Beyer

Ziel dieses Teils ist es, mithilfe von genetischen Algorithmen die Muster zu finden, die entsprechend einem gewissen Gütemaß die besten sind. Daraufhin haben wir diese besten Muster mittels und und oder verknüpfüngen verbunden und diese meta-muster nochmals bewertet.

2.1. Grundlagen der genetischen Algorithmen

Die Genetischen Algorithmen [LUG02] bilden eine Familie von Algorithmen, die auf die Darwinsche Evolutionstheorie [DAR59] basieren. Die Suche nach einem Optimum wird als ein Wettbewerb unter Individuen einer Population betrachtet. Diese Individuen (manchmal auch Chromosome genannt) werden durch eine Fitnessfunktion bewertet, wobei die Bewertung eine große Rolle in der Entscheidung spielt, inwiefern dieses Individuum in die nächste Lösungseneration Einfluss nimmt.

Es handelt sich hierbei um iterative Algorithmen bei denen eine Iteration aus zwei Hauptschritten besteht: die Bewertung und der Generationswechsel.

Die Bewertung

Während der Bewertungsphase wird jedes Individuum einem Eignungstest unterzogen. In diesem Test berechnet eine Eignungsfunktion (Engl: evaluation function) die Lösungsgüte (den score) des Individuums, also wie nahe dieses Individuum an der Lösung liegt bzw. wie « gut » dieses Individuum ist. Die Eignungsfunktion ist für jedes Problem neu zu entwerfen.

Anhand dieses scores, den scores der anderen Individuen und ggf. den scores der vorherigen Iterationen berechnet eine Fitnessfunktion dann die Fitness des Individuums. Hierfür gibt es einige Standardverfahren, u.A. Lineare [SUS04] und exponentielle[FAR04] Skalierung.

Es ist auch möglich mehrere Eignungsfuntionen zur berechnung der Fitness zu verwenden.

Der Generationswechsel

In dem Generationswechsel von der aktuellen Generation G_i zur neuen Generation G_{i+1} wird ein gewisser Anteil der Individuen unveränder übernommen, während auf einem (manchmal anderen) Anteil der Individuen genetische Operatoren angewendet werden, um somit neue Individuen zu generieren.

2.2. Genetische Operatoren

S. Manodumrongthum

Es wird zwischen drei Arten genetischer Operatoren unterschieden: Initialisatoren, Mutatoren und Crossover-Operatoren.

Initialisatoren

Die Initialisatoren werden verwendet um die Individuen der Anfangsgeneration G_0 zu generieren. Die Initialisatoren können rein zufällige Individuen generieren, durch sie kann aber auch ein eventuelles vorwissen über die Problemdomaine eingebracht werden. Ist kein solches Vorwissen vorhanden wird empfohlen, die Individuen gleichmäßig über die Problemdomäne zu verteilen[CHOU00].

Mutatoren

Mutatoren fügen, wie auch in der Genetik, zufällige veränderungen in die Individuen ein. Dadurch wird erreicht, dass die Individuen der Population im Lösungsraum « springen » können und somit die Gesamtpopulation nicht in einem lokalen Optimum « hängen bleibt ». Dabei ist es wichtig zu beachten, dass diese Sprünge immer innerhalb des Lösungsraums bleiben.

Üblicherweise wird der Mutation eine feste Wahrscheinlichkeit zugeordnet. Das heißt, es wird anfangs eine konstante Wahrscheinlichkeit gewählt, mit der eine Mutation stattfindet. Eine mögliche Erweiterung hierzu ist die adaptive Mutation, in der die Mutationsrate variabel, abhängig von der Fitness eines Individuums ist[LIB00].

Crossover-Operatoren

Diese Operatoren sind dafür verantwortlich, aus einem Elternpaar ein oder mehrere neue Individuen (die Kinder) zu generieren, welche aus kombinierten

eigenschaften des Elternpaars bestehen. Diese Operation entspricht der Paarung in der Natur. Hiermit wird das sogenannte « hill-climbing » erreicht, also das nähern an ein Optimum. Auch hier gilt es darauf zu achten, dass die Kinder den Lösungsraum nicht verlassen. Weiterhin gibt es auch hier die Möglichkeit einer adaptiven Crossoverrate[SRI94].

2.3. Eignungsfunktion als Gütemaß

L. Beyer

Die Eignungsfunktion, auch Zielfunktion genannt, steuert den genetischen Algorithmus in Richtung eines Ziels indem sie die Individuen bewertet. Individuen mit besseren Bewertungen überleben, bekommen Nachfahren und somit verschiebt sich die Population in Richtung eines Optimums dieser Zielfunktion.

Die meisten Frameworks für genetische Algorithmen sind darauf ausgelegt, Chromosome zu finden welche die Eignungsfunktionen maximieren. Daraus folgt, dass die Eignungsfunktion « guten » Individuen eine hohe Eignung zuweisen muss und « schlechten » Individuen eine niedrige Eignung.

Die Eignungsfunktion ist nicht nur auf die direkte nähe eines Individuums zur Lösung beschränkt. Weitere gewüschte Kriterien können eingebaut werden, indem diese gut oder schlecht bewertet werden. Somit kann man komplexere Individuen bestrafen, wenn man an einer einfachen Lösung interessiert ist.

2.4. Umsetzung

S. Manodumrongthum

Implementiert haben wir diesen Teil in der Programmiersprache Python 2.6 [PY11] anhand des genetische Algorithmen-Frameworks Pyevolve 0.6rc1 [PYEV09].

Darstellung der Individuen

Zur Darstellung der Individuen haben wir eine Kombination der Muster vierter und zweiter Form verwendet. Das bedeutet, dass das Individuum aus einer Liste von N « Sets » besteht, jedes dieser Sets beinhaltet alle Aminosäuren oder Aminosäuregruppen, die an dieser Stelle im Muster stehen dürfen.

Wir haben uns für diese Darstellung mit sets entschieden, da bei anderen Darstellungen crossover-operatoren nicht sinnvoll definiert werden können.

Initialisatoren

Wir haben zwei verschiedene Initialisatoren implementiert, von denen immer zufällig einer zur Initialisierung eines Individuums verwendet werden kann.

Zur Initialisierung eines Individuums wird zufällig eines der Fenster, die häufiger als einmal in den positiven Proteinen vorkommen verwendet. Daraufhin wird einer zufälligen anzahl an Sets des Individuums eine zufällige Aminosäure hinzugefügt.

In einer anderen Variante wird ein Individuum mit drei zufälligen Fenstern aus den häufigeren Fenstern besetzt. Somit hat jedes Set des Individuums eine größe von bis zu 3.

Mutatoren

Folgende Mutatoren haben wir verwendet:

Un/Gruppieren

Dieser Mutator ersetzt zufällige Aminosäuren durch ihre Gruppe bzw. ersetzt eine Aminosäurengruppe durch ein zufällig gewähltes Mitglied.

Das Ziel dieses Mutators ist es, zu bestimmen ob an einer bestimmten Stelle ein Amino oder seine Gruppe besser geeignet ist.

ÄndereAmino

Dieser Mutator ersetzt zufällige Sets durch ein Set welches aus nur einem Amino besteht, welches vorher nicht in dem Set war und wenn möglich von einer anderen Gruppe stammt.

Das Ziel dieses Mutators ist es, im Lösungsraum zu springen und somit viele verschiedene Muster auszuprobieren.

Reinitialisierung

Dieser Mutator tritt nur mit einer sehr geringen Wahrscheinlichkeit ein. Er löscht ein Individuum komplett und initialisiert es wieder mit einem zufälligen häufigen positiven Fenster.

Dieser Mutator hat zwei Ziele: Erstens erlaub er es, in dem Lösungsraum zu springen, zweitens springt er nur an Stellen, bei denen ein relativ hoher score fast

garantiert ist. Dadurch hilft soll er dem Algorithmus helfen wieder « auf den Weg zu kommen », wenn er nur noch Lösungen mit niedrigem score besucht.

Seterweiterung

Dieser Mutator fügt zufälligen Sets zufällige Aminosäuren hinzu oder entfernt zufällige Einträge des Sets. Dieser Mutator hat das selbe Ziel wie der ÄndereAmino Mutator.

Crossover-Operatoren

In PyEvolve wird eine Crossover Operation immer auf zwei Eltern-Individuen angewendet und ergibt immer zwei Kind-Individuen.

Folgende Crossover-Operatoren haben wir verwendet:

CrossoverGerecht

Dieser Crossover-Operator bestimmt an jeder Stelle des Musters die Gemeinsamkeiten und die Unterschiede zwischen dem Vater und der Mutter. Dies entspricht den folgenden Mengenoperationen:

$$\begin{aligned} G &= M \cap V \\ U &= (M \cup V) \setminus G \end{aligned}. \tag{17}$$

Dabei steht G für Gemeinsam, U für Unterschied, M für Mutter und V für Vater.

Beiden Kindern werden die gemeinsamen Einträge gegeben, während die unterschiedlischen Einträge zufällig aber gerecht auf die Kinder verteilt werden.

Dieser Operator soll nützen, aus zwei guten Individuen zwei weitere gute alternativen zu bilden, von denen eins vielleicht sogar besser ist, als die Eltern.

CrossoverUngerecht

Dieser Crossover-Operator verwendet auch die Mengen G und U, die oben definiert wurden. Im gegensatz zu CrossoverGerecht bekommt hier jedoch ein Kind die komplette Menge G und das andere Kind die komplette Menge U. Wenn eine der Mengen leer ist, wird sie zufällig gefüllt.

Dieser Operator erlaubt es, aus zwei guten Eltern ein hoffentlich noch besseres, aber einfacheres Kind und ein eher zufälliges Kind zu erstellen.

Eignungsfunktionen

L. Beyer

Die verwendete Eignungsfunktion besteht aus zwei Termen. Der erste Term, der die Trenngüte des Individuums definiert, bestimmt den größten Anteil der Fitness eines Individuums. Da wir auf der Suche nach einem für Menschen verständlichen Muster sind, bestraft der zweite Term besonders komplexe Individuen und belohnt einfachere Individuen.

Um die Trenngüte eines Individuums zu messen haben wir das in 1.3. definierte Maß $t(m)$ verwendet.

Für den zweiten Term haben wir folgende Formel verwendet:

$$S=\frac{w}{\sum_{i=1}^{w} L_i}, \tag{18}$$

wobei L_i die Länge des Sets an der i-ten Stelle des Individuums darstellt.

Da $min(L_i)=1$ gilt, ist der Wertebereich dieser Funktion durch Im(S)=(0,1) gegeben.

Ein Individuum in dem jeder Set die Länge 1 hat ist das einfachst mögliche Individuum. Für solch ein Individuum gilt $S=1$. Das kompliziertest mögliche Individuum besteht aus sets maximaler Länge. Mit der vereinfachenden Annahme, dass die maximale Länge eines Sets unendlich ist, gilt im Grenzfall $S=0$.

Um eine bessere Kompatibilität zu den Anzeigefunktionalitäten des PyEvolve Frameworks zu erreichen, haben wir die Eignungswerte mit 100 multipliziert. Es handelt sich hierbei um eine reine konstante Skalierung für die Darstellung.

2.5. Ergebnisse

S. Manodumrongthum, L. Beyer

In den Ergebnissen konnten wir feststellen, dass die genetischen Algorithmen mit den oben vorgestellten Operatoren für kleine Mustergrößen äußerst erfolgreich

sind. Bei jedem Lauf fanden Sie eines der Fenster, die auch in der vollständigen Suche am besten abgeschnitten haben.

Die algorithmen konvergierten nach schon wenigen Generationen und liefen somit deutlich schneller als die vollständige Suche.

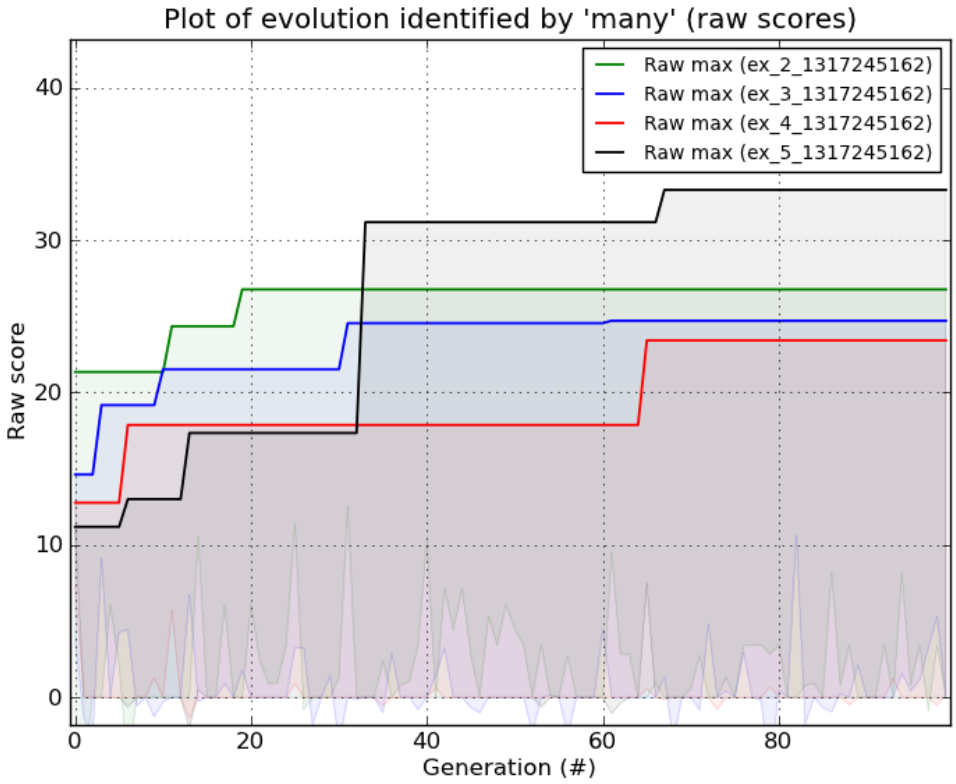


Abb 4: Eignung der Individuen verschiedener Musterlängen über die Generationen im Vergleich. Datensatz: PARP10

Abb 4 Zeigt den Graphen der Punkte, die die Eignungsfunktion den Individuen gegeben hat. Die Linien kennzeichnen jeweils die Bewertung des besten Individuums einer Generation, die Fläche darunter ist bis zu den Punkten des schlechtesten Individuums der Generation mit einer halbtransparenten Farbe gefüllt. Der Graph hat die erwartete typische Form eines Generationengraphens genetischer Algorithmen.

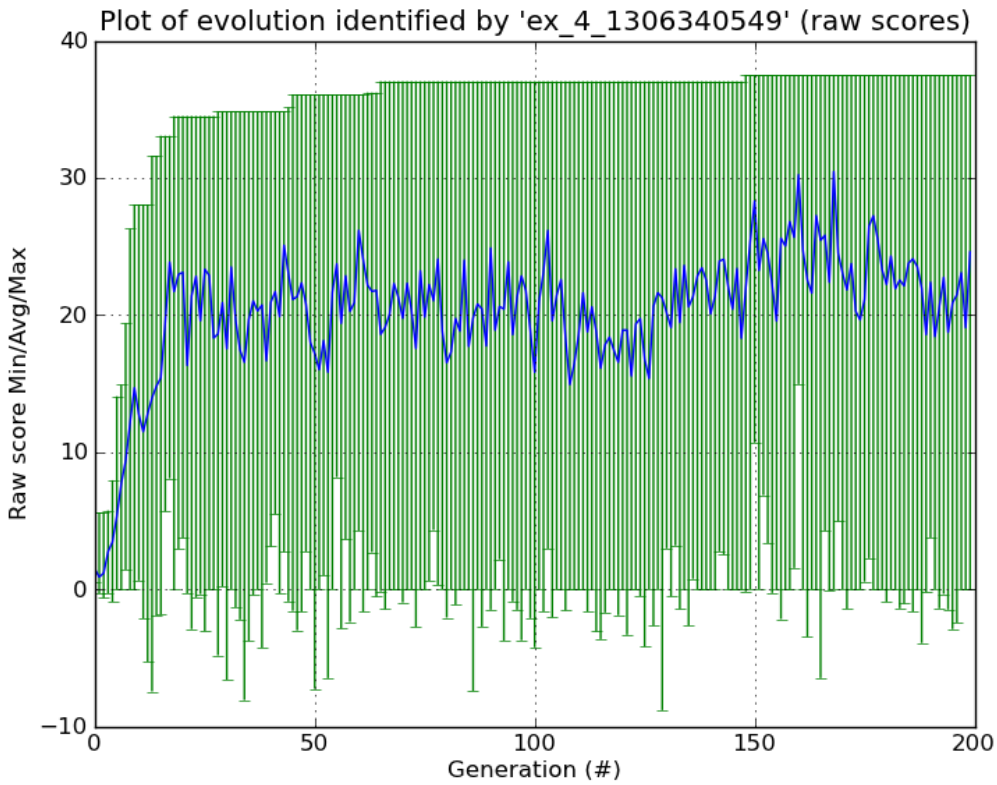


Abb 5: Generationenplot eines Laufes mit der Musterlänge w=4 auf PARP10 Daten

Abb 5 zeigt nochmals, dass für geringe Fenstergrößen nach ca 100 Generationen schon ein sehr gutes Ergebnis zur verfügung steht. Dieser Plot entspricht dem Muster `[L],[P,G,V],[X,R,H],[I,J,T],` welches einen Score von 37.487 (bzw. 0.3748 unskaliert) erreichen konnte. Muster in dieser Form können nicht durch die hier vorgestellte vollständige Mustersuche gefunden werden.

In einem weiteren Beispiel zeigt Abb 6, dass es sich Lohnen kann sogar bis zu 500 Generationen zu warten. Diese Abbildung entspricht dem Fenster `[B,D], [A,V,Z,F],[J,M,N],[Q,I,T,W],` welches einen Score von 42.571665 (bzw. 0.42571665 unskaliert) erreicht hat, wenn der Strafterm für die Komplexität nicht mit eingerechnet wird.

Wenn man beachtet, dass das beste Muster welches durch die vollständige Mustersuche gefunden wurde `DFXX` mit einem Score von 0.376196 ist, zeigt sich hier die Stärke dieser Darstellung (Muster der 2. und 4. Form kombiniert).

Es ist zu bemerken, dass die (unskalierten) Scores des genetischen Algorithmus ohne Strafterm durchaus mit den Scores der vollständigen Mustersuche zu vergleichen sind, da die selbe Bewertungsfunktion verwendet wurde.

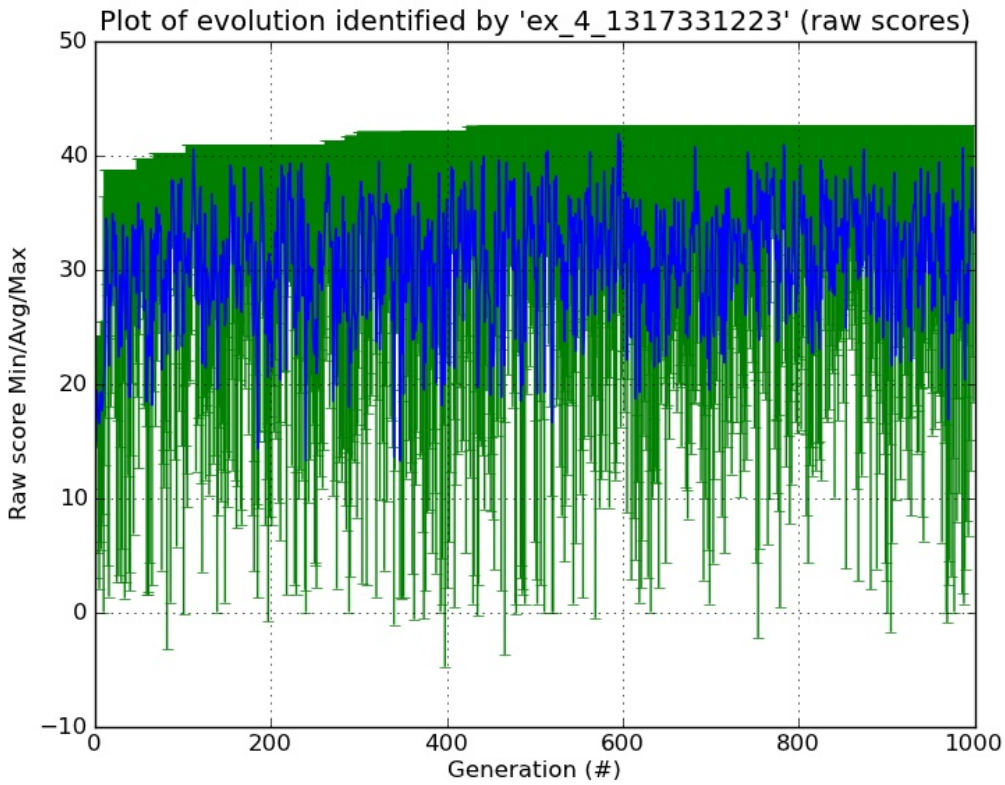


Abb 6: Generationenplot über 1000 Generationen eines Musters mit w=4 für PARP10 Daten

Leider ließen sich diese vielversprechenden Ergebnisse nicht auf große Muster erweitern. Dies kann entweder daran liegen, dass sich hier gute Lösungen erst in viel späteren Generationen zeigen, für die uns jedoch nichtmehr genug Rechenzeit zur Verfügung stand oder aber, dass andere Parametereinstellungen benötigt werden, die wir leider nicht finden konnten. Dies ist ein bekannter Nachteil genetischer Algorithmen und wurde unter anderem in [ROT00] erwähnt und in [LOBO99] untersucht.

Ein Beispiel hierfür zeigt Abb 7, in der ein Generationenplot eines Musters mit $w=10$ zu sehen ist. Das Muster ist `[A,J],[Y,H,V],[J],[I,X],[P,R,H],[T,W],[P,T,L],[M,S,E,C],[Z],[Y,C,R,L]`. Es ist in dem Plot eindeutig zu erkennen, dass dank der starken Initialisatoren eine gute Anfangsgeneration entstanden ist, diese sich jedoch nicht verbessern konnte. Dies bedeutet, dass die meisten Kinder (einer Crossover-Operation) bzw. Mutationen schlechter wurden.

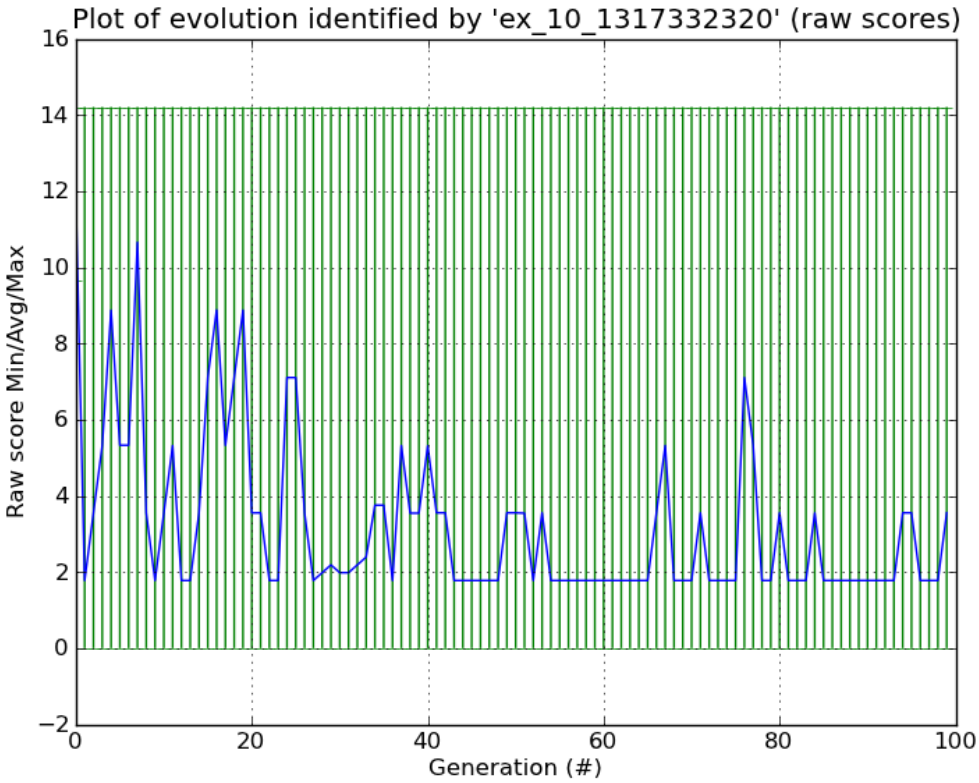


Abb 7: Generationenplot über 100 Generationen eines Musters mit w=10 für PARP10 Daten

Abb 8 zeigt der Vollständigkeit halber die Ergebnisse des genetischen Algorithmus mit den selben Einstellungen wie in Abb 4, nur die Musterlänge w wurde vergrößert. Man kann auch hier sehen, dass bis auf einen Sprung nicht viel passiert ist.

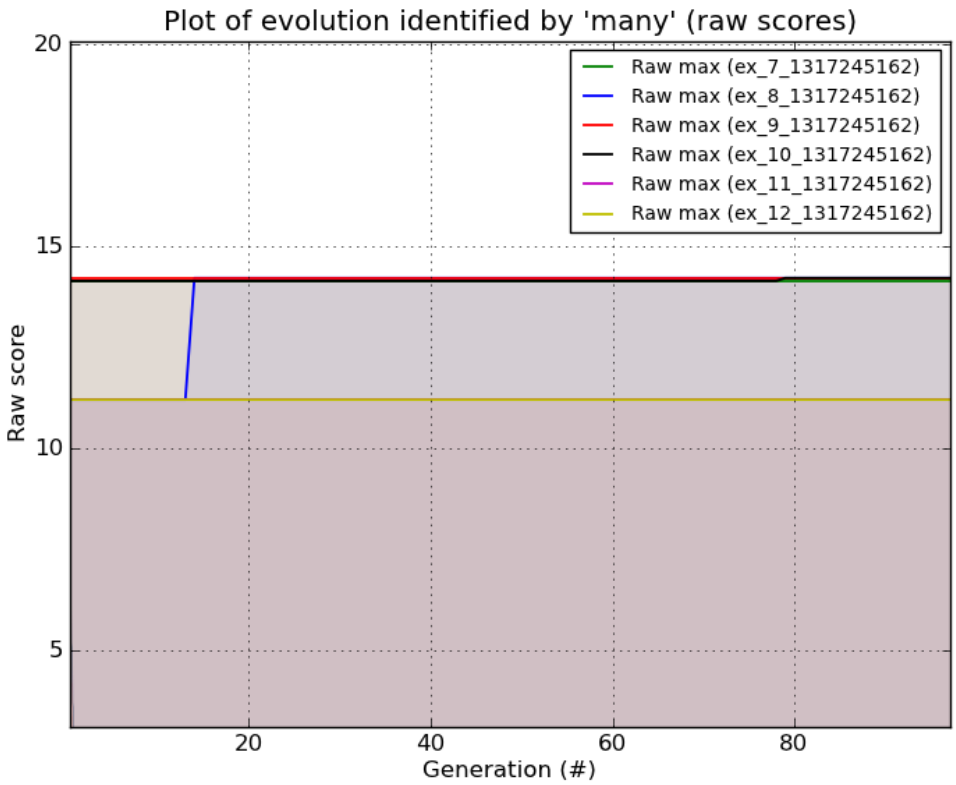


Abb 8: Eignung der Individuen verschiedener großer Musterlängen über die Generationen im Vergleich. Datensatz: PARP10

Diese Feststellungen lassen sich jedoch gut damit vereinigen, dass auch in der vollständigen Mustersuche für größere Fenster eine Musterfamilie, in der jedes Muster den selben Score hatte, die Ergebnisse dominiert hat.

2.6. Fazit und Ausblick

S. Manodumrongthum, L. Beyer

In diesem Abschnitt haben wir gezeigt, dass die hier vorgestellten genetischen Algorithmen für kleine Mustergrössen (ca. $w \leq 7$) schnell sehr gute Lösungen finden. Es können auch Muster der vierten Form gefunden werden, was durch die hier vorgestellte vollständige Mustersuche nicht möglich ist. Diese Muster der vierten Form können bessere Scores als die Anderen erreichen.

Leider ließen sich diese Erfolge nicht wie erhofft auf größere Muster übertragen.

Eine Möglichkeit besteht darin, die Bewertungsfunktion zu beschleunigen (z.B. in C zu implementieren), um somit eine viel höhere Generationenzahl erreichen zu können.

Eine interessante Fortführung dieser Arbeit wäre auch hier die `and`, `or` und evlt. `not` Verknüpfung der erfolgreichen Muster kleiner Länge.

V. Zusammenfassung

I. Bayer, L. Beyer, S. Manodumrongthum

Im ersten Teil wurden ausgehend von Proteinsequenzen in Einbuchstabencode und Bindungsergebnissen für Parp14 und Parp10 Vorhersagemodelle für die Bindungsfähigkeit erstellt. Dazu wurde zum einen die original Einbuchstabencode Darstellung verwendet und zum anderen die Buchstaben der Einbuchstabenkodierung durch Gruppen ersetzt. Aus den beiden Darstellungen wurden dann Features generiert. Mit diesen wurden dann Support Vector Machine Modelle trainiert und deren Parameter optimiert. Anschließenden wurden eine Modellkombination erstellt wodurch eine Verbesserung der Performance erreicht wurde.

Im Zweiten Teil wurde nach Mustern gesucht, welche so häufig wie möglich in den Positiven Proteinsequenzen *und* so selten wie möglich in den Negativen Proteinsequenzen vorkommen. Dazu wurden verschiedene Musterarten entworfen. In diesen Musterarten wurde eine vollständige Suche durchgeführt, wobei Einschränkungen und Annahmen getroffen werden mussten um den Suchraum zu bendigen. Um auch nach komplexeren Mustern und ohne Einschränkungen suchen zu können, wurden genetische Algorithmen entworfen.

VI. Bestbewertete Parp10 Muster ohne subsampling

L. Beyer

Muster	P(m|p)	P(m|n)	t(m)
JXXZXXXX	0.316667	0.115699	0.200967
XXJXXZXX	0.333333	0.127673	0.205661
XXXJXXZX	0.333333	0.124007	0.209326
DXXZXXXX	0.283333	0.0631643	0.220169
XXXEXXZX	0.3	0.07416	0.22584
XBXXDXXX	0.283333	0.0568112	0.226522
XBXXJFXX	0.266667	0.0323763	0.23429
XKXXJXXX	0.283333	0.0489921	0.234341
XBXXDFXX	0.266667	0.0295663	0.2371
XBXXJFGX	0.266667	0.0283445	0.238322
XBXXDFGX	0.266667	0.0278558	0.238811
XKXXJFXX	0.266667	0.0277337	0.238933
XKXXDFXX	0.266667	0.0265119	0.240155
XKXXJFGX	0.266667	0.0263897	0.240277
XKXXDFGX	0.266667	0.0261454	0.240521
XBXXJXGX	0.283333	0.0370189	0.246314
XKXXDXXX	0.283333	0.0361637	0.24717
XBXXDXGX	0.283333	0.0331093	0.250224
XKXXJXGX	0.283333	0.0298106	0.253523
XKXXDXGX	0.283333	0.0281002	0.255233
total average score was: 0.0185476			

Table 7: PARP10 Ergebnisse für Muster der Länge 8 ohne subsampling

VI.Bestbewertete Parp10 Muster ohne subsampling

Muster	P(m|p)	P(m|n)	t(m)
XBXXJXXXX	0.2	0.0444716	0.155528
JXBXJZXXX	0.183333	0.0262676	0.157066
DXBXJZXXX	0.183333	0.0254123	0.157921
XKXXJXXXX	0.183333	0.0252902	0.158043
JXBXJNXXX	0.183333	0.0250458	0.158288
XEXZXXJXX	0.166667	0.00818571	0.158481
JXKXJZXXX	0.183333	0.0248015	0.158532
DXBXJNXXX	0.183333	0.0245571	0.158776
DXKXJZXXX	0.183333	0.0244349	0.158898
JXKXJNXXX	0.183333	0.0243128	0.159021
DXKXJNXXX	0.183333	0.0241906	0.159143
XZXXXXXDS	0.166667	0.00720831	0.159458
XXJBBZXXX	0.166667	0.00696396	0.159703
XXZXXXXXJ	0.216667	0.0551008	0.161566
XKXXDXXXX	0.183333	0.0208919	0.162441
XBXXJXGXX	0.183333	0.0207697	0.162564
XBXXDXGXX	0.183333	0.0190593	0.164274
XKXXJXGXX	0.183333	0.0166158	0.166718
XKXXDXGXX	0.183333	0.0158827	0.167451
JXXXZXJXX	0.183333	0.0152718	0.168061
XJXXXZXJX	0.183333	0.0134392	0.169894
LXXEXXZXX	0.183333	0.0131949	0.170138
LXXJXXZXX	0.2	0.0186927	0.181307
XXXEXXZXX	0.233333	0.0392181	0.194115
XXXJXXZXX	0.266667	0.063653	0.203014
total average score was: 0.0185086			

Table 8: PARP10 Ergebnisse für Muster der Länge 9 ohne subsampling

VI.Bestbewertete Parp10 Muster ohne subsampling

Muster	P(m|p)	P(m|n)	t(m)
BBBJZJXXZZ	0.15	0.00549786	0.144502
ZZBBBBBBJZ	0.15	0.00549786	0.144502
BBBBBBJZJX	0.15	0.00549786	0.144502
BBBBBJZJXX	0.15	0.00549786	0.144502
XXXZPXXXXJ	0.15	0.00525351	0.144746
XXJXXXXDXB	0.15	0.00488699	0.145113
BXJXZXXJXX	0.15	0.00488699	0.145113
XXEXXXXJXB	0.15	0.00452046	0.14548
ZXXJBBZXXX	0.15	0.00415394	0.145846
RXJXZXXJXX	0.15	0.00403177	0.145968
XXEXXXXDXB	0.15	0.00403177	0.145968
BXEXZXXJXX	0.15	0.00390959	0.14609
RXEXZXXJXX	0.15	0.00366524	0.146335
XXJXXXZXJX	0.166667	0.0091631	0.157504
XJXXXZXJXX	0.183333	0.00843005	0.174903
total average score was: 0.0184423			

Table 9: PARP10 Ergebnisse für Muster der Länge 10 ohne subsampling

Muster	P(m|p)	P(m|n)	t(m)
XZZZBBBHHBJ	0.15	0.00537569	0.144624
XZZZBBBHBHD	0.15	0.00537569	0.144624
XZZZBBBHBHJ	0.15	0.00537569	0.144624
XZZZBBBHBBD	0.15	0.00537569	0.144624
XZZZBBBHBBJ	0.15	0.00537569	0.144624
XZZZBBBBHHD	0.15	0.00537569	0.144624
XZZZBBBBHHJ	0.15	0.00537569	0.144624
XZZZBBBBHBD	0.15	0.00537569	0.144624
XZZZBBBBHBJ	0.15	0.00537569	0.144624
XZZZBBBBBHD	0.15	0.00537569	0.144624
XZZZBBBBBHJ	0.15	0.00537569	0.144624
XZZZBBBBBBD	0.15	0.00537569	0.144624
XZZZBBBBBBJ	0.15	0.00537569	0.144624
XXJXXXZXJXX	0.166667	0.00610874	0.160558
total average score was: 0.0183937			

Table 10: PARP10 Ergebnisse für Muster der Länge 11 ohne subsampling

Muster	P(m|p)	P(m|n)	t(m)
XZZZBBBBHBJY	0.15	0.00537569	0.144624
XZZZBBBBHBJZ	0.15	0.00537569	0.144624
XZZZBBBBBHDY	0.15	0.00537569	0.144624
XZZZBBBBBHDZ	0.15	0.00537569	0.144624
XZZZBBBBBHJY	0.15	0.00537569	0.144624
XZZZBBBBBHJZ	0.15	0.00537569	0.144624
XZZZBBBBBBDY	0.15	0.00537569	0.144624
XZZZBBBBBBDZ	0.15	0.00537569	0.144624
XZZZBBBBBBJY	0.15	0.00537569	0.144624
XZZZBBBBBBJZ	0.15	0.00537569	0.144624
total average score was: 0.0183367			

Table 11: PARP10 Ergebnisse für Muster der Länge 12 ohne subsampling

VII. Bestbewertete Parp14 Muster ohne subsampling

L. Beyer

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
XXEXXXXJ	0.134228	0.028042	0.106186	0.827189
XJXXXZXJ	0.134228	0.0270537	0.107174	0.832258
XXXBZXXJ	0.14094	0.0332304	0.107709	0.809207
XXXZXJBZ	0.127517	0.0182829	0.109234	0.874603
XZXXZZXJ	0.147651	0.037554	0.110097	0.79723
XEXXXZXJ	0.127517	0.0168005	0.110716	0.883587
XZXXJXXX	0.228188	0.116862	0.111326	0.661319
XKXXJFXX	0.14094	0.0274243	0.113515	0.837113
XXEXXBXX	0.154362	0.0403953	0.113967	0.792586
XZXXEXXX	0.181208	0.0667078	0.1145	0.730926
GXXXXJXX	0.161074	0.0443484	0.116725	0.784111
XBXXJFXX	0.154362	0.0318715	0.122491	0.828863
LXXEXXZL	0.134228	0.00827671	0.125951	0.94192
XXEXXZLX	0.147651	0.0190241	0.128627	0.885861
XXXEXXZL	0.147651	0.0164299	0.131221	0.899867
XXEXXZXX	0.208054	0.0760964	0.131957	0.732197
XXJXXZLX	0.167785	0.035701	0.132084	0.824553
XXXJXXZL	0.161074	0.0287832	0.132291	0.848396
XXJXXZXX	0.261745	0.126745	0.1350	0.67375
LXXJXXZL	0.147651	0.0117356	0.135915	0.92637
LXXEXXZX	0.167785	0.0193947	0.148391	0.896385
XXXEXXZX	0.228188	0.073008	0.15518	0.757606
LXXJXXZX	0.187919	0.0321186	0.155801	0.854031
XXXJXXZX	0.288591	0.122545	0.166046	0.701936
total average score was: 0.00708246				

Table 12: PARP14 Ergebnisse für Muster der Länge 8 ohne subsampling

VII.Bestbewertete Parp14 Muster ohne subsampling

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
JXXXJXXZX	0.114094	0.0164299	0.0976641	0.874123
EXXXJXXZX	0.107383	0.00926498	0.0981176	0.920573
XXJXXXZXJ	0.114094	0.0154416	0.0986523	0.880793
JXXZXXZZX	0.120805	0.0208771	0.0999283	0.852648
XXXXBZXXD	0.114094	0.0101297	0.103964	0.918456
XXXXBZXXJ	0.134228	0.0174182	0.11681	0.885139
LXXEXXZLX	0.134228	0.00679432	0.127434	0.951821
LXXJXXZLX	0.14094	0.00827671	0.132663	0.944532
XXXJXXZLX	0.154362	0.0171711	0.137191	0.899896
XXXEXXZLX	0.147651	0.0101297	0.137521	0.935799
LXXEXXZXX	0.154362	0.0118592	0.142503	0.928654
LXXJXXZXX	0.161074	0.0174182	0.143656	0.902415
XXXEXXZXX	0.187919	0.0379246	0.149995	0.832076
XXXJXXZXX	0.214765	0.0623842	0.152381	0.774908
total average score was: 0.00709332				

Table 13: PARP14 Ergebnisse für Muster der Länge 9 ohne subsampling

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
BXJXXXZJAX	0.0872483	0.00222359	0.0850247	0.975148
XXJXXXZXJX	0.0939597	0.00877085	0.0851889	0.914623
JXXXJXXZXX	0.0939597	0.00877085	0.0851889	0.914623
XJXXZXJXXX	0.0939597	0.00840025	0.0855595	0.917934
XXXJXXZLXZ	0.0939597	0.00765905	0.0863007	0.92463
LXXJXXZXXZ	0.0939597	0.00617665	0.0877831	0.938318
XZXDXXXBXX	0.0939597	0.00592959	0.0880301	0.940638
XXXJXXZXXZ	0.107383	0.0190241	0.0883585	0.849501
XDXXZXJXXX	0.0939597	0.00506485	0.0888949	0.948853
ZXXXXBZXXJ	0.0939597	0.00457072	0.089389	0.953611
XJLXXJXXZX	0.0939597	0.00432366	0.0896361	0.956008
XXJBBZXXXX	0.0939597	0.00407659	0.0898831	0.958418
XELXXJXXZX	0.0939597	0.00358246	0.0903773	0.963273
XXJKBZXXXX	0.0939597	0.00321186	0.0907479	0.966946
XJXXXJXXZX	0.100671	0.00852378	0.0921474	0.92194
XJXXXZXJXX	0.100671	0.00802965	0.0926415	0.926131
XEXXXJXXZX	0.100671	0.00568252	0.0949886	0.94657
XJXXZXXZZX	0.107383	0.0108709	0.0965116	0.908072
total average score was: 0.00707954				

Table 14: PARP14 Ergebnisse für Muster der Länge 10 ohne subsampling

VII.Bestbewertete Parp14 Muster ohne subsampling

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
IZXXEXXXLDX	0.0872483	0.00222359	0.0850247	0.975148
IZXXEXXXLJX	0.0872483	0.00222359	0.0850247	0.975148
IZXXEXXXXDX	0.0872483	0.00222359	0.0850247	0.975148
DBXXKZJXXXB	0.0872483	0.00222359	0.0850247	0.975148
IZXXEXXXXJX	0.0872483	0.00222359	0.0850247	0.975148
DBXXBZDXXXB	0.0872483	0.00222359	0.0850247	0.975148
IZXXJXXXLDX	0.0872483	0.00222359	0.0850247	0.975148
IZXXJXXXLJX	0.0872483	0.00222359	0.0850247	0.975148
DXXXBXXZXKK	0.0872483	0.00222359	0.0850247	0.975148
IZXXJXXXXDX	0.0872483	0.00222359	0.0850247	0.975148
DXXXBXXZXBK	0.0872483	0.00222359	0.0850247	0.975148
IZXXJXXXXJX	0.0872483	0.00222359	0.0850247	0.975148
DBXXBZJXXXB	0.0872483	0.00222359	0.0850247	0.975148
JXFXBXXZXKK	0.0872483	0.00222359	0.0850247	0.975148
JXFXBXXZXKB	0.0872483	0.00222359	0.0850247	0.975148
JXFXBXXZXBK	0.0872483	0.00222359	0.0850247	0.975148
JXFXBXXZXBB	0.0872483	0.00222359	0.0850247	0.975148
JBXXKZDXXXB	0.0872483	0.00222359	0.0850247	0.975148
XJXZXLEXXXL	0.0872483	0.00222359	0.0850247	0.975148
XJXZXLEXXXX	0.0872483	0.00222359	0.0850247	0.975148
JXXXBXXZXKK	0.0872483	0.00222359	0.0850247	0.975148
JXXXBXXZXBK	0.0872483	0.00222359	0.0850247	0.975148
XXXJXXZXXZZ	0.0939597	0.00741198	0.0865478	0.926883
XXJXXZXXZZX	0.0939597	0.00704138	0.0869184	0.930284
XXJXXXZXJXX	0.0939597	0.00568252	0.0882772	0.942971
total average score was: 0.00706358				

Table 15: PARP14 Ergebnisse für Muster der Länge 11 ohne subsampling

VII.Bestbewertete Parp14 Muster ohne subsampling

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
XZXXXJBXXBZ	0.0805369	0.00222359	0.0783133	0.973132
ZXXXJBXXBZJ	0.0805369	0.00222359	0.0783133	0.973132
XXXEXXZXXZZX	0.0872483	0.00358246	0.0836659	0.960559
DXXXBXXZXKBB	0.0872483	0.00247066	0.0847777	0.972462
DXXXBXXZXBBB	0.0872483	0.00247066	0.0847777	0.972462
XKXDXXSKXXJX	0.0872483	0.00247066	0.0847777	0.972462
XKXDXXSBXXJX	0.0872483	0.00247066	0.0847777	0.972462
XKXDXXZKXXJX	0.0872483	0.00247066	0.0847777	0.972462
XKXDXXZBXXJX	0.0872483	0.00247066	0.0847777	0.972462
XKXJXXSKXXJX	0.0872483	0.00247066	0.0847777	0.972462
XKXJXXSBXXJX	0.0872483	0.00247066	0.0847777	0.972462
XKXJXXZKXXJX	0.0872483	0.00247066	0.0847777	0.972462
XKXJXXZBXXJX	0.0872483	0.00247066	0.0847777	0.972462
XBXDXXSKXXJX	0.0872483	0.00247066	0.0847777	0.972462
XBXDXXSBXXJX	0.0872483	0.00247066	0.0847777	0.972462
XBXDXXZKXXJX	0.0872483	0.00247066	0.0847777	0.972462
XXXEXXZLXZZX	0.0872483	0.00247066	0.0847777	0.972462
XBXDXXZBXXJX	0.0872483	0.00247066	0.0847777	0.972462
XBXJXXSKXXJX	0.0872483	0.00247066	0.0847777	0.972462
XBXJXXSBXXJX	0.0872483	0.00247066	0.0847777	0.972462
XXXJXXZLXZZX	0.0872483	0.00247066	0.0847777	0.972462
XBXJXXZKXXJX	0.0872483	0.00247066	0.0847777	0.972462
JXXXBXXZXKBB	0.0872483	0.00247066	0.0847777	0.972462
JXXXBXXZXBBB	0.0872483	0.00247066	0.0847777	0.972462
XBXJXXZBXXJX	0.0872483	0.00247066	0.0847777	0.972462
DXFXBXXZXKKB	0.0872483	0.00222359	0.0850247	0.975148
DXFXBXXZXKBB	0.0872483	0.00222359	0.0850247	0.975148
DXFXBXXZXBKB	0.0872483	0.00222359	0.0850247	0.975148
DXFXBXXZXBBB	0.0872483	0.00222359	0.0850247	0.975148
DXXXBXXZXKKB	0.0872483	0.00222359	0.0850247	0.975148
DXXXBXXZXBKB	0.0872483	0.00222359	0.0850247	0.975148
JXFXBXXZXKKB	0.0872483	0.00222359	0.0850247	0.975148
JXFXBXXZXKBB	0.0872483	0.00222359	0.0850247	0.975148
JXFXBXXZXBKB	0.0872483	0.00222359	0.0850247	0.975148
JXFXBXXZXBBB	0.0872483	0.00222359	0.0850247	0.975148
JXXXBXXZXKKB	0.0872483	0.00222359	0.0850247	0.975148
JXXXBXXZXBKB	0.0872483	0.00222359	0.0850247	0.975148
XXXJXXZXXZZX	0.0939597	0.00481779	0.0891419	0.951226
total average score was: 0.00704473				

Table 16: PARP14 Ergebnisse für Muster der Länge 12 ohne subsampling

VIII. Bestbewertete Parp10 Muster mit subsampling

L. Beyer

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
XZXXXJLX	0.2	0	0.2	1
XJXXZBXX	0.2	0	0.2	1
ZXXXJXXZ	0.216667	0.0163934	0.200273	0.92966
ZXXEXXXX	0.233333	0.0327869	0.200546	0.876797
XZXXXEXX	0.25	0.0491803	0.20082	0.835617
XZXXXXXJ	0.266667	0.0655738	0.201093	0.802632
XZXXXJXX	0.283333	0.0819672	0.201366	0.775617
LXXJXXZX	0.216667	0	0.216667	1
XKXXDFGX	0.266667	0.0491803	0.217486	0.844291
XKXXDFXX	0.266667	0.0491803	0.217486	0.844291
XKXXJFGX	0.266667	0.0491803	0.217486	0.844291
XKXXJFXX	0.266667	0.0491803	0.217486	0.844291
XBXXDFGX	0.266667	0.0491803	0.217486	0.844291
XBXXDFXX	0.266667	0.0491803	0.217486	0.844291
XBXXJFGX	0.266667	0.0491803	0.217486	0.844291
XBXXJFXX	0.266667	0.0491803	0.217486	0.844291
XXJXXZXX	0.333333	0.114754	0.218579	0.743902
DXXZXXXX	0.283333	0.0491803	0.234153	0.852095
XKXXDXGX	0.283333	0.0491803	0.234153	0.852095
XKXXDXXX	0.283333	0.0491803	0.234153	0.852095
XKXXJXGX	0.283333	0.0491803	0.234153	0.852095
XKXXJXXX	0.283333	0.0491803	0.234153	0.852095
XBXXDXGX	0.283333	0.0491803	0.234153	0.852095
XBXXDXXX	0.283333	0.0491803	0.234153	0.852095
XBXXJXGX	0.283333	0.0491803	0.234153	0.852095
XBXXJXXX	0.283333	0.0491803	0.234153	0.852095
XXXEXXZX	0.3	0.0491803	0.25082	0.859155
XXXJXXZX	0.333333	0.0655738	0.26776	0.835616
total average score was: 0.0187109				

Table 17: PARP10 Ergebnisse für Muster der Länge 8 mit subsampling

VIII.Bestbewertete Parp10 Muster mit subsampling

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
XJXXXXJXB	0.15	0	0.15	1
XXJXXXXJX	0.15	0	0.15	1
XZXZXXXXJ	0.15	0	0.15	1
XXXZXJBZX	0.15	0	0.15	1
BXJXXZBXX	0.15	0	0.15	1
ZXXJBBZXX	0.15	0	0.15	1
XZXXXXXDS	0.166667	0.0151515	0.151515	0.916667
XZXXXXXDZ	0.166667	0.0151515	0.151515	0.916667
XXZXXXXLJ	0.166667	0.0151515	0.151515	0.916667
XXZBXXJXL	0.166667	0.0151515	0.151515	0.916667
XZXXXXXJS	0.166667	0.0151515	0.151515	0.916667
XXZBXXJXX	0.166667	0.0151515	0.151515	0.916667
XZXXXXXJZ	0.166667	0.0151515	0.151515	0.916667
XJXZXXJXX	0.166667	0.0151515	0.151515	0.916667
XKXXDXGXX	0.183333	0.030303	0.15303	0.858156
XKXXDXXXX	0.183333	0.030303	0.15303	0.858156
XKXXJXGXX	0.183333	0.030303	0.15303	0.858156
XKXXJXXXX	0.183333	0.030303	0.15303	0.858156
JXXXZXJXX	0.183333	0.030303	0.15303	0.858156
XJXXXZXJX	0.183333	0.030303	0.15303	0.858156
JXXZXXXXX	0.216667	0.0606061	0.156061	0.781421
XEXZXXJXX	0.166667	0	0.166667	1
XXJBBZXXX	0.166667	0	0.166667	1
LXXEXXZXX	0.183333	0.0151515	0.168182	0.923664
XXJXXZXXZ	0.183333	0.0151515	0.168182	0.923664
XXZXXXXXJ	0.216667	0.0454545	0.171212	0.82659
LXXJXXZXX	0.2	0.0151515	0.184849	0.929578
XXXEXXZXX	0.233333	0.030303	0.20303	0.885057
XXXJXXZXX	0.266667	0.0454545	0.221212	0.854369
total average score was: 0.0185592				

Table 18: PARP10 Ergebnisse für Muster der Länge 9 mit subsampling

VIII.Bestbewertete Parp10 Muster mit subsampling

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
ZZBBBBBBJZ	0.15	0	0.15	1
BBBBBBJZJX	0.15	0	0.15	1
BBBBBJZJXX	0.15	0	0.15	1
BBJZJXXZZJ	0.15	0	0.15	1
BJZJXXZZJZ	0.15	0	0.15	1
JZJXXZZJZX	0.15	0	0.15	1
XXZZJZXZXZ	0.15	0	0.15	1
XZZJZXZXZX	0.15	0	0.15	1
XXJXXXJXB	0.15	0	0.15	1
BXJXZXXJXX	0.15	0	0.15	1
ZXXJBBZXXX	0.15	0	0.15	1
XJXXXBXXZX	0.15	0	0.15	1
XXJXXXZXJX	0.166667	0.0140845	0.152582	0.922078
XXXZXXXXJ	0.166667	0.0140845	0.152582	0.922078
XJXXXZXJXX	0.183333	0.0140845	0.169249	0.928656
total average score was: 0.0184521				

Table 19: PARP10 Ergebnisse für Muster der Länge 10 mit subsampling

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
XZZZBBBBBBD	0.15	0	0.15	1
ZZBBBBBBJZD	0.15	0	0.15	1
ZBBBBBBJZJI	0.15	0	0.15	1
BBBBBBJZJXP	0.15	0	0.15	1
BBBBBJZJXXT	0.15	0	0.15	1
BBBJZJXXZZE	0.15	0	0.15	1
BBJZJXXZZJN	0.15	0	0.15	1
BJZJXXZZJZL	0.15	0	0.15	1
JZJXXZZJZXY	0.15	0	0.15	1
XZZZBBBBBBJ	0.15	0	0.15	1
ZZZBBBBBBJZ	0.15	0	0.15	1
ZZBBBBBBJZJ	0.15	0	0.15	1
ZBBBBBBJZJX	0.15	0	0.15	1
BBBBBBJZJXX	0.15	0	0.15	1
BBBBBJZJXXZ	0.15	0	0.15	1
BBBBJZJXXZZ	0.15	0	0.15	1
BBBJZJXXZZJ	0.15	0	0.15	1
BBJZJXXZZJZ	0.15	0	0.15	1
BJZJXXZZJZX	0.15	0	0.15	1
JZJXXZZJZXZ	0.15	0	0.15	1
XZZJZXZXZXX	0.15	0	0.15	1
XXJXXXZXJXX	0.166667	0	0.166667	1
total average score was: 0.0184405				

Table 20: PARP10 Ergebnisse für Muster der Länge 11 mit subsampling

VIII.Bestbewertete Parp10 Muster mit subsampling

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
XZZZBBBBBBJY	0.15	0.0151515	0.134848	0.908257
ZZZBBBBBBJZD	0.15	0.0151515	0.134848	0.908257
ZZBBBBBBJZJI	0.15	0.0151515	0.134848	0.908257
ZBBBBBBJZJXP	0.15	0.0151515	0.134848	0.908257
BBBBBJZJXXZT	0.15	0.0151515	0.134848	0.908257
BBBBJZJXXZZE	0.15	0.0151515	0.134848	0.908257
BBBJZJXXZZJN	0.15	0.0151515	0.134848	0.908257
BJZJXXZZJZXY	0.15	0.0151515	0.134848	0.908257
ZJXXZZJZXZXQ	0.15	0.0151515	0.134848	0.908257
JXXZZJZXZXZG	0.15	0.0151515	0.134848	0.908257
ZZZBBBBBBJZJ	0.15	0.0151515	0.134848	0.908257
BBBBBJZJXXZZ	0.15	0.0151515	0.134848	0.908257
BBBBJZJXXZZJ	0.15	0.0151515	0.134848	0.908257
ZZBBBBBBJZJX	0.15	0.0151515	0.134848	0.908257
XZZZBBBBBBJZ	0.15	0.0151515	0.134848	0.908257
ZBBBBBBJZJXX	0.15	0.0151515	0.134848	0.908257
BBBBBBJZJXXZ	0.15	0.0151515	0.134848	0.908257
BBBJZJXXZZJZ	0.15	0.0151515	0.134848	0.908257
BBJZJXXZZJZX	0.15	0.0151515	0.134848	0.908257
BJZJXXZZJZXZ	0.15	0.0151515	0.134848	0.908257
JZJXXZZJZXZX	0.15	0.0151515	0.134848	0.908257
ZJXXZZJZXZXZ	0.15	0.0151515	0.134848	0.908257
JXXZZJZXZXZX	0.15	0.0151515	0.134848	0.908257
XXZZJZXZXZXX	0.15	0.0151515	0.134848	0.908257
total average score was: 0.0184932				

Table 21: PARP10 Ergebnisse für Muster der Länge 12 mit subsampling

IX. Bestbewertete Parp14 Muster mit subsampling

L. Beyer

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
XJXXBXXJ	0.120805	0.00662252	0.114183	0.948029
XXXZXJBZ	0.127517	0.013245	0.114272	0.905905
XKXXJFXX	0.14094	0.0264901	0.11445	0.841784
XXXXEZXX	0.147651	0.0331126	0.114538	0.816818
XXJXXBXX	0.167785	0.0529801	0.114805	0.760016
XXEXXZXX	0.208054	0.0927152	0.115338	0.69174
XXJXXZXX	0.261745	0.145695	0.11605	0.642414
XXXJBXXB	0.120805	0	0.120805	1
XEXXXZXJ	0.127517	0.00662252	0.120894	0.95063
XJXXZBXX	0.127517	0.00662252	0.120894	0.95063
XJXXXZXJ	0.134228	0.013245	0.120983	0.910187
XBXXJFXX	0.154362	0.0331126	0.12125	0.823376
XXEXXXXJ	0.134228	0.00662252	0.127606	0.952982
XXEXXBXX	0.154362	0.0264901	0.127872	0.853526
XXXXJBXX	0.154362	0.0264901	0.127872	0.853526
GXXXXJXX	0.161074	0.0331126	0.127961	0.829481
XXXEXXZX	0.228188	0.0993377	0.12885	0.696703
LXXEXXZL	0.134228	0	0.134228	1
XXEXXZLX	0.147651	0.00662252	0.141028	0.957073
LXXEXXZX	0.167785	0.0264901	0.141295	0.863646
XXJXXZLX	0.167785	0.0264901	0.141295	0.863646
LXXJXXZL	0.147651	0	0.147651	1
XXXEXXZL	0.147651	0	0.147651	1
LXXJXXZX	0.187919	0.0397351	0.148184	0.825458
XXXJXXZX	0.288591	0.13245	0.15614	0.685423
XXXJXXZL	0.161074	0	0.161074	1
total average score was: 0.0070597				

Table 22: PARP14 Ergebnisse für Muster der Länge 8 mit subsampling

IX.Bestbewertete Parp14 Muster mit subsampling

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
JXXXZJXXJ	0.0939597	0	0.0939597	1
JXXZBXXJX	0.0939597	0	0.0939597	1
JXJZXBXJX	0.0939597	0	0.0939597	1
JXXBXXJJB	0.0939597	0	0.0939597	1
XJXXBXXJJ	0.0939597	0	0.0939597	1
XXJXXZXXZ	0.120805	0.026738	0.0940674	0.818778
JLXXJXXZX	0.100671	0.00534759	0.0953235	0.94956
XDXXXZJXX	0.100671	0.00534759	0.0953235	0.94956
BXXEXXBXX	0.100671	0.00534759	0.0953235	0.94956
XXJXXXZXD	0.100671	0.00534759	0.0953235	0.94956
BXXJXXBXX	0.100671	0.00534759	0.0953235	0.94956
XJXXXZXJX	0.100671	0.00534759	0.0953235	0.94956
EXXXJXXZX	0.107383	0.0106952	0.0966874	0.909423
XXZXXZZXJ	0.107383	0.0106952	0.0966874	0.909423
XXXXBZXXJ	0.134228	0.0374332	0.096795	0.781936
JXXXJXXZX	0.114094	0.0160428	0.0980512	0.876724
BXJXXZBXX	0.100671	0	0.100671	1
XXJXXXZXJ	0.114094	0.0106952	0.103399	0.914294
XXXJXXZXX	0.214765	0.101604	0.113161	0.678843
XJXXZXXZZ	0.114094	0	0.114094	1
JXXZXXZZX	0.120805	0	0.120805	1
LXXJXXZXX	0.161074	0.0320856	0.128988	0.833891
LXXEXXZLX	0.134228	0	0.134228	1
XXXEXXZXX	0.187919	0.0534759	0.134444	0.778471
LXXJXXZLX	0.14094	0	0.14094	1
LXXEXXZXX	0.154362	0.0106952	0.143667	0.935203
XXXEXXZLX	0.147651	0	0.147651	1
XXXJXXZLX	0.154362	0	0.154362	1
total average score was: 0.00699703				

Table 23: PARP14 Ergebnisse für Muster der Länge 9 mit subsampling

IX.Bestbewertete Parp14 Muster mit subsampling

Muster	P(m|p)	P(m|n)	t(m)	p(p|m)
RDXKXJNXXX	0.100671	0.0112994	0.0893717	0.899086
RDXKXJZXXX	0.100671	0.0112994	0.0893717	0.899086
RDXBXJNXXX	0.100671	0.0112994	0.0893717	0.899086
RDXBXJZXXX	0.100671	0.0112994	0.0893717	0.899086
RJXKXJNXXX	0.100671	0.0112994	0.0893717	0.899086
RJXKXJZXXX	0.100671	0.0112994	0.0893717	0.899086
RJXBXJNXXX	0.100671	0.0112994	0.0893717	0.899086
RJXBXJZXXX	0.100671	0.0112994	0.0893717	0.899086
BDXKXJNXXX	0.100671	0.0112994	0.0893717	0.899086
BDXKXJZXXX	0.100671	0.0112994	0.0893717	0.899086
BDXBXJNXXX	0.100671	0.0112994	0.0893717	0.899086
BDXBXJZXXX	0.100671	0.0112994	0.0893717	0.899086
BJXKXJNXXX	0.100671	0.0112994	0.0893717	0.899086
BJXKXJZXXX	0.100671	0.0112994	0.0893717	0.899086
BJXBXJNXXX	0.100671	0.0112994	0.0893717	0.899086
BJXBXJZXXX	0.100671	0.0112994	0.0893717	0.899086
XXXJXXZXXZ	0.107383	0.0169492	0.0904334	0.863678
XELXXJXXZX	0.0939597	0	0.0939597	1
XJLXXJXXZX	0.0939597	0	0.0939597	1
XZXDXXXBXX	0.0939597	0	0.0939597	1
XXXEXXZXXZ	0.0939597	0	0.0939597	1
XXJKBZXXXX	0.0939597	0	0.0939597	1
XXJBBZXXXX	0.0939597	0	0.0939597	1
ZXXXXBZXXJ	0.0939597	0	0.0939597	1
XZXJXXXBXX	0.0939597	0	0.0939597	1
XJXXXJXXZX	0.100671	0.00564972	0.0950214	0.946862
XXZXZXXXXJ	0.100671	0.00564972	0.0950214	0.946862
XJXXZXXZZX	0.107383	0.0112994	0.0960831	0.904793
XEXXXJXXZX	0.100671	0	0.100671	1
total average score was: 0.00710139				

Table 24: PARP14 Ergebnisse für Muster der Länge 10 mit subsampling

X. Literatur

CHE03: Chris Cheadle, Marquis P. Vawter, William J. Freed, Kevin G. Becker, Analysis of Microarray Data Using Z ScoreTransformation, 2003

ORANGE: Blaz Zupan, Gregor Leban, Janez Demsar, Tomaz Curk, Widgets and Visual Programming,

AHO90: Aho, Alfred V., Algorithms for finding patterns in strings, 1990

CSV05: Y. Shafranovich, Common Format and MIME Type for Comma-Separated Values (CSV) Files, 2005

LUG02: George F. Luger, Künstliche Intelligenz, 2002

DAR59: Darwin, Charles, On the Origin of Species, 1859

SUS04: Sushil J. Louis, Scaling in Genetic Algorithms, 2004, http://www.cse.unr.edu/~sushil/class/gas/notes/scaling/index.html

FAR04: Farzad A. Sadjadi, Comparison of fitness scaling functions in genetic algorithms withapplications to optical processing, 2004

CHOU00: Chih-Hsun Chou, Jou-Nan Chen, Genetic Algorithms: initialization schemes and genes extraction, 2000

LIB00: S. Marsili Libelli, P. Alba, Soft Computing 4, 2000

SRI94: M. Srinivas, L. M. Patnaik, Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms, 1994

PY11: Various Authors, Python v2.6.7 documentation, 2011, http://docs.python.org/release/2.6.7/

PYEV09: Christian S. Perone, Welcome to Pyevolve documentation !, 2009, http://pyevolve.sourceforge.net/

XI. Abbildungsverzeichnis