# 20.10.2000 - Computational Genefinding

Hier ist meine Zusammenfassung vom Computational Genefinding.
Sie enthaelt ein paar mehr Details als ich im Vortrag angegeben
habe. Genefinding koennte, wenn die Einfuehrung vorbei ist,
ein interessantes Projekt sein, wenn sich genug Leute finden,
die daran arbeiten wollen.

1. Ziel:
Man moechte "frisch sequenzierte" genomische DNA annotieren, d.h.
Sequenzabschnitte gewissen Klassen zuordnen, z.B.
- codierende / nichtcodierende Sequenz
(codierend := kann in Proteine uebersetzt werden),
d.h. wo sind wirklich die Gene im Genom?
- Promoter / Exons / Introns:
Hat man den Start eines Gens (hinter einem Promoter) gefunden,
wird nicht immer eine zusammenhaengende Sequenz wirklich in ein
Protein uebersetzt. Oft gibt es dabei Luecken, sog. Introns
(=nichtcodierende DNA in einem Genbereich).
Wo beginnen / enden diese Introns?
- Regulatorische Signale:
Z.B. zur Uebersetzung von DNA in Proteine werden bestimmte Enzyme
benoetigt, diese muessen an die DNA andocken. Dies kann nur an
bestimmten Stellen geschehen (z.B. wenn dort eine bestimmte relativ
feste DNA Sequenz vorliegt)
Wo sind solche Stellen?

2.Methoden

a) Zum Auffinden von sogenannten Signalen (das sind z.B. Start- und
Stoppcodons; Stellen, an die sich Enzyme binden, etc.) benutzt man
sogenannte signalbasierte Methoden. Das sind
- eine Konsensussequenz (d.h. man geht davon aus, dass eine feste Sequenz
an einer solchen stelle vorliegen muss, wie beim Startcodon)
- positionsspezifische Scorematrizen (Verallgemeinerung der
Konsensussequenz: es gibt fuer manche Sequenzen hoehere Scores als fuer
andere; solche mit hohen Scores deuten auf das Vorhandensein des gesuchten
Signals hin)

b) Zum Bestimmen des Inhaltes / der Funktion eines laengeren Stueckes von
DNA verwendet man sog. inhaltsbasierte Methoden. Das sind z.B.
- Markov-Modelle: Z.B. unterscheiden sich Exons und Introns in ihrem
Aufbau. Man koennte beobachten, dass in Exons oft A auf G folgt; in
Introns haeufiger T auf G (dies ist voellig hypothetisch und aus der Luft
gegriffen, aber man kann versuchen, solche "Folgewahrscheinlichkeiten"
aus bekannten Exons und Introns herzuleiten und sehen, inwieweit sie

sich unterscheiden). Bei einem unbekannten Stueck DNA schaut man nun, ob sie besser zum Exon oder Intronmodell passt und klassifiziert dieses dann entsprechend.
- Neuronale Netzwerke: Darueber wollen wir noch genauer sprechen. Prinzip: Man trainiert das NN mit bekannten Beispielen z.B. von codierender / nichtcodierender DNA und laesst es die Eigenschaften "lernen". Nach dem Training praesentiert man dem NN unbekannte DNA und laesst es nach codierend/nichtcodierend klassifizieren.
Probleme: Es ist nicht immer klar, welche Eigenschaften das NN wirklich als relevant einstuft und ob es auf die unbekannte DNA gut generalisieren kann.

c) integrative Methoden
Im Idealfall moechte man folgendes erreichen: Man praesentiert einem Programm ein komplettes Genom, und das Programm erledigt alle moeglichen Annotationen automatisch (das wird in den seltensten Faellen funktionieren). Hier besteht eine Verwandtschaft zur grammatischen Analyse ("Parsing") natuerlicher Sprachen: Es ist ein Satz gegeben, und man moechte ihn in seine Bestandteile aufgliedern und deren Funktion bestimmen (Subjekt, Praedikat, Objekt, ...). In der Tat ist das Genefinding stark von Linguisten beeinflusst worden. Man hat z.B. versucht, (probabilisitische) Grammatiken fuer DNA anzugeben.
Das Problem besteht dann eben gerade im "Parsing" einer gegebenen Sequenz. Am beliebtesten hier sind HMMs (Hidden Markov Models): Man hat fuer jede "Klasse" von DNA-Sequenz ein Markov-Model (MM), und das Problem besteht darin, fuer eine gegebene Sequenz zu bestimmen, in welchem MM man sich gerade (mit grosser Wahrscheinlichkeit) befindet, d.h. in welcher "Klasse" (Exon / Intron / ...).

**ACHTUNG:** Die meisten der heute verfuegbaren Methoden / Programme sind zwar relativ gut, aber eben noch lange nicht perfekt. Man muss mit Fehlklassifikationen rechnen. Dies koennte, wie oben angedeutet, ein interessantes Projekt werden (mit bestehenden Methoden experimentieren, neue Ideen entwickeln, die Ergebnisse mit den bestehenden Programmen vergleichen, etc.). Das ist mit Sicherheit nicht einfach und kann sehr arbeitsaufwendig sein, aber hier gibt es noch viel zu tun.