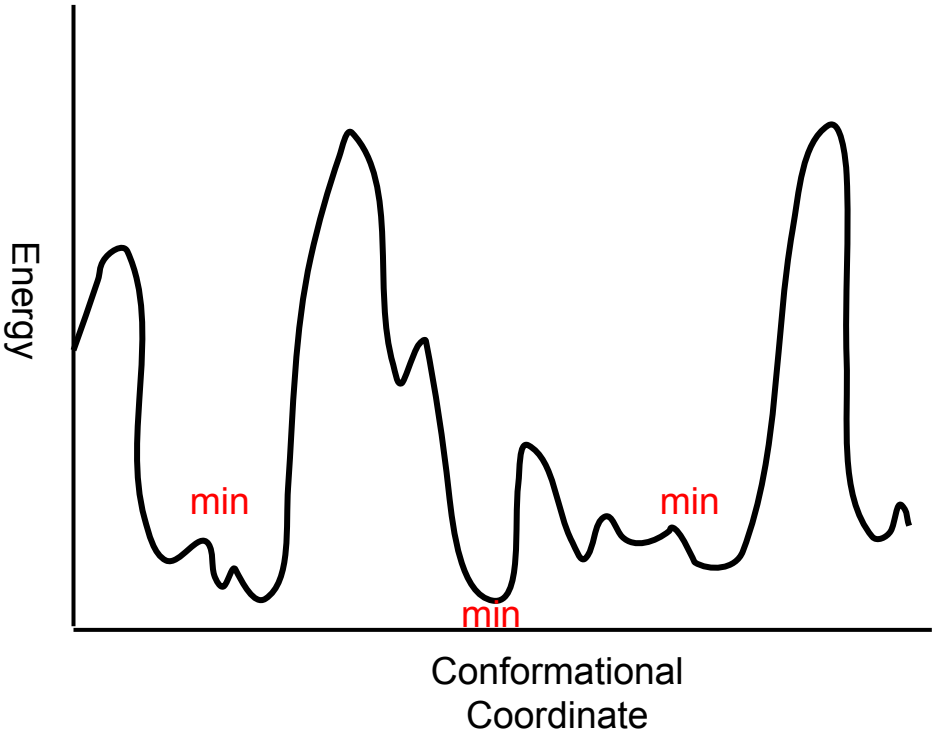
Learning Molecular Dynamics with CRF

Josh Wheeler & Yotam Barnoy

Background



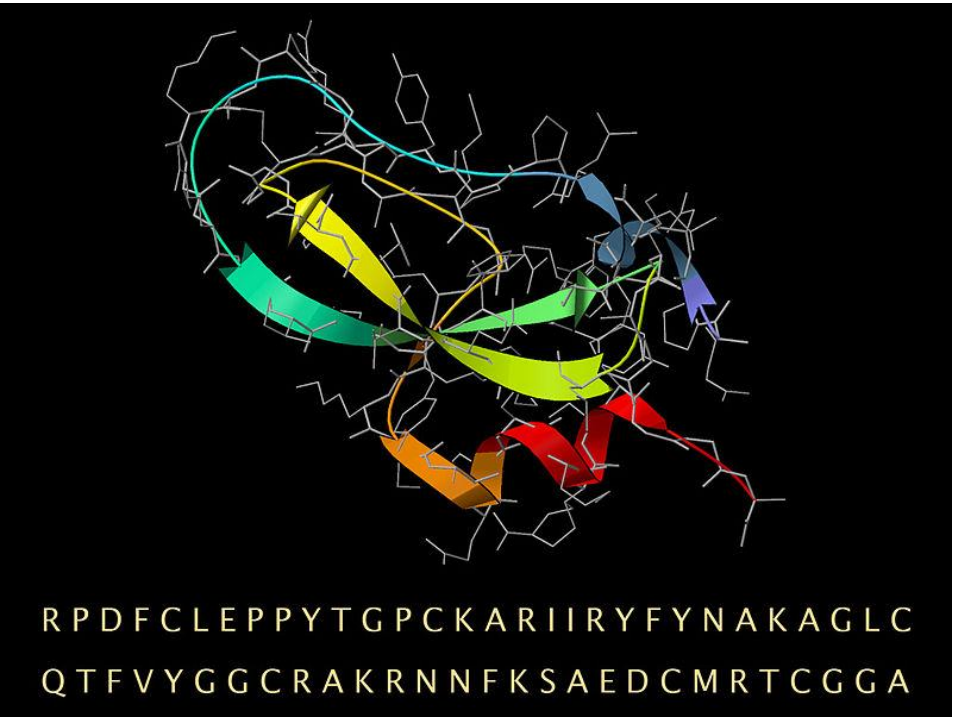
Science benefits greatly from knowing how proteins interact with one another. Molecular Dynamics simulations are computationally intensive simulations used to track the positions of atoms in a protein over time. The longest known simulation to date has covered 1ms of time[4], which is considered extremely long in MD terms.

As proteins moves over time, they may eventually enter different conformational states, which are low energy states.

We are interested in identifying features that are associated with different conformational states. Finding features associated with certain states is useful for separating state transitions from noise.

For future work, we are also interested in attempting to predict state-transitions, that is, places in the data where the protein leaves one state and enters another. These events are very rare due to how short simulations tend to be.

Big Data



Our dataset consists of a large MD Trajectory for the BPTI[5] (bovine pancreatic trypsin inhibitor) protein. This protein is relatively small, containing 568 atoms, and is therefore often used in molecular dynamics. While it's difficult to say with absolute certainty, it appears that there are 5 possible conformational states for this protein.

Our data comes in the form of a time series, with several million timesteps of data.

In its raw form, the data is simply (x,y,z) coordinates for each of the atoms in the protein, per timestep. However, (x,y,z) tuples are not an ideal choice for model features, as atoms tend to make small movements regularly, and pure positioning data neglects the relationship between atoms.

Related Work

Since this area is at the forefront of biophysics, there is much uncertainty about how to advance the state of the art. Many papers are therefore exploratory in nature.

One avenue of research has been the simulation of molecules. To date, the most impressive effort has been that of D.E.Shaw research[4], who used their Anton supercomputer to calculate frames of BPTI every 250ps for a total of 1ms.

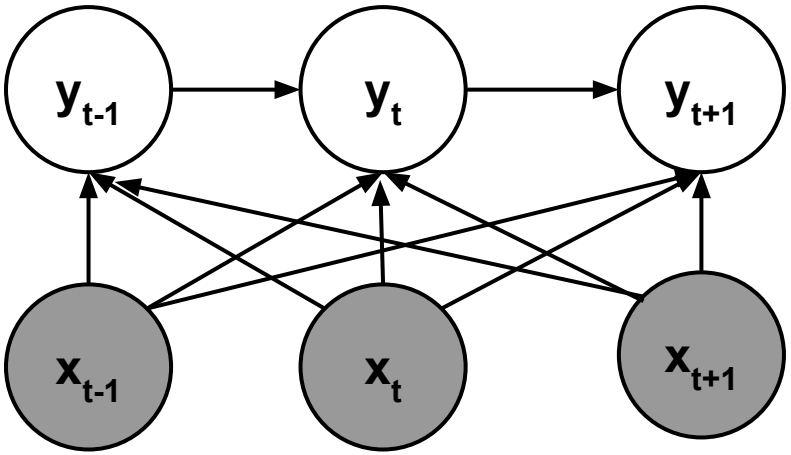
Another direction is performing analysis on the large amounts of data produced by such simulations. The raw (x,y,z) coordinates produced by simulations are a poor choice for features, due to the fact that they are not rotation and translation invariant. Feature engineering is therefore a large part of any analysis of the data. It's not entirely clear, though, which features are most useful.

Derived features, such as dihedral and side-chain angles, and the presence of hydrogen bonds, appear to correlate better with conformational states. The angular features are can be binned effectively, and hydrogen bonds can be boolean valued, allowing for a slightly lower dimensional representation of the data. Nevertheless, the dimensionality is still quite high, and is difficult to deal with in general.
Some papers have shown that some common dimensionality reducing techniques such as PCA are not valid for MD simulations, as the data violates some of the necessary assumptions [3].

Finally, there is the matter of applying machine learning technique to actually find patterns in the data and/or predict states. This carries its own set of difficulties due to the abundant data and the relatively few state transitions present in current simulation data. This field is sparse, with the exceptions being [2] and [7].

Linear-Chain Conditional Random Fields

States



Observations

CRFs are a variation on the traditional Bayes Nets/Markov Models.

In terms of our project:
- Each y_t represents the state label of the protein at time t.
- Each x_t represents the set of observed features at time t.

Motivation:

- Linear-Chain Model for modelling time-series data.
- We have a large set of observations. Modelling the joint **P(X,Y)** would be difficult. CRFs allow us only to model the conditional distribution **P(Y|X)** so that distributions of observations are not modelled explicitly.
- Our observations are strongly-correlated (groups of atoms move together)
- CRF's can be useful for feature-selection in high-dimensional data.
- CRF's are ideal for labeling sequence data

Linear-Chain CRF CPDs

y_{t-1}	y_t	Factor
1	1	$\exp(\sum_i \lambda_i f_i(1, 1, x_{1:t}))$
1	2	$\exp(\sum_i \lambda_i f_i(1, 2, x_{1:t}))$
2	1	$\exp(\sum_i \lambda_i f_i(2, 1, x_{1:t}))$
2	2	$\exp(\sum_i \lambda_i f_i(2, 2, x_{1:t}))$

The CRF handles heterogeneous observation data well since it doesn't model its probability distribution at all.

Our CPDs are relatively simple. Each entry is in log-linear form. We create a set of feature functions **F.** Each function f_k is associated with a weight $lambda_i$. We will discuss our choice of feature functions in another section.

Each factor's scope is 2 consecutive timesteps (and implicitly, all observations).

The probability of an entire label sequence is simply a (normalized) product of all factors in the network.

During Parameter Estimation, we will attempt to find the set of weights that maximizes the likelihood of the data

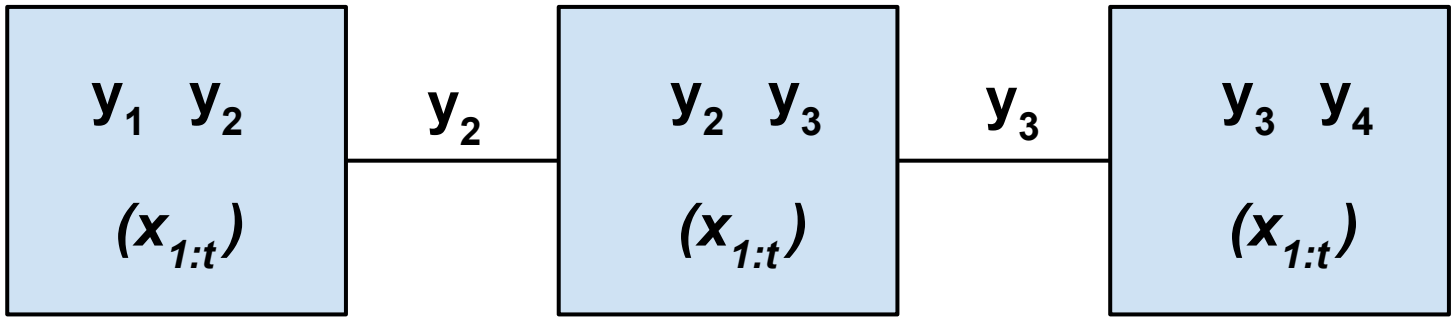
Inference

Exact inference is tractable in a linear-chain CRF using several algorithms.

We chose to use Sum-Product Message Passing in a Clique Tree for inference.

- We initialize each factor using the log-linear equations above.
- The clique tree can be calibrated in 2 passes, upward and downward.

We can also use Max-Product Message Passing for decoding. (Assigning labels to data)



Example Clique Tree

Parameter Estimation

We used gradient ascent on the log-likelihood of the data, with an extra regularization term that placed a Gaussian Prior on lambda.

Log Likelihood: $$l(\lambda) = \sum_{n=1}^{N} \sum_{t=2}^{T} \sum_{i=1}^{I} \lambda_i f_i(y_{t-1}^n, y_t^n, x_{1:t}^n) - \sum_{n=1}^{N} log(Z(x^n)) - \sum_{i=1}^{I} \frac{\lambda_i^2}{2\sigma^2}$$

Gradient:

$$\frac{\partial l}{\partial \lambda_i} = \sum_{n=1}^{N} \sum_{t=1}^{T} f_k(y_{t-1}^n, y_t^n, x_{1:t}^n) - \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{y,y'} p(y, y' | x_{i:t}^n) f_k(y, y', x_{1:t}^n) - \frac{\lambda_i}{\sigma^2}$$

The first sum is the expected value of f_k under the empirical distribution of the data.

The second sum is the expected value of f_k under the current model parameters.

The last term simply penalizes large parameter values to help prevent over-fitting.

It is worth noting that there is a probability query in this expression. This means that we need to run inference once for every piece of data during gradient ascent. (Since the query is conditioned on the current observations).

Implementation / Features

We used our own ocaml-based message-passing framework, extended to support gradient ascent and feature functions. Ocaml is a high-performance functional language, and as such, allows calculation at speeds often nearing that of C while allowing for type-safe high-level constructs.

The framework is flexible and allows for easy creation of new feature functions to be incorporated into the model.

Since the dataset is very large, time to convergence on the whole dataset is on the order of hours. We performed experiments with varying trajectory lengths, training set sizes, and hyper-parameters. Due to issues of scale, we were only able to explore smaller chunks of the dataset at a time.

All feature functions have the form $f(y_{t-1}, y_{t,}$ **x**)

It is common practice to use indicator functions inside the feature functions. For example, functions might take the following form.

$f(y_{t-1},y_t,$ **x)** = if $y_t = 1$ AND x.chi2[34] in bin 1

then 1

else 0

A strong weight for this function suggests a correlation between state 1 and finding the chi2 angle number 32 in bin 1.

We enumerated functions for all possible current states, and all possible bins of each angle. We also created feature functions for each of the hydrogen bonds available in the data. This leads to large number of features for the CRF to learn.

Results and Analysis

hbond[485]: 0.001139
hbond[456]: 0.001139
hbond[398]: 0.001139
hbond[381]: 0.001139
hbond[360]: 0.001139
hbond[352]: 0.001139
hbond[340]: 0.001139
hbond[315]: 0.001139
hbond[307]: 0.001139
chi2[9] bin[90.0-270.0]: 0.001139
chi2[8] bin[90.0-270.0]: 0.001139
chi1[37] bin[90.0-270.0]: 0.001139
chi1[31] bin[90.0-270.0]: 0.001139
chi1[30] bin[90.0-270.0]: 0.001139
chi1[25] bin[90.0-270.0]: 0.001139
chi1[17] bin[90.0-270.0]: 0.001139
chi1[16] bin[90.0-270.0]: 0.001139
chi1[6] bin[90.0-270.0]: 0.001139
chi1[3] bin[270.0-330.0]: 0.001139

hbond[168]: 0.000956
hbond[524]: 0.000956
hbond[307]: 0.000956
hbond[705]: 0.000957
hbond[680]: 0.000957
hbond[516]: 0.000957
hbond[485]: 0.000957
hbond[381]: 0.000957
hbond[360]: 0.000957
hbond[340]: 0.000957
hbond[315]: 0.000957
chi2[9] bin[90.0-270.0]: 0.000957
chi2[8] bin[90.0-270.0]: 0.000957
chi1[37] bin[90.0-270.0]: 0.000957
chi1[30] bin[90.0-270.0]: 0.000957
chi1[25] bin[90.0-270.0]: 0.000957
chi1[17] bin[90.0-270.0]: 0.000957
chi1[16] bin[90.0-270.0]: 0.000957
chi1[6] bin[90.0-270.0]: 0.000957

We spent most of our time working on the implementation of our model, but we were able to achieve some initial results. With more work, we believe we can dig deeper and get more meaningful results.

As a starting point, we ran our learning procedure on many different pieces of the data set with different hyper-parameters. After each run, we sorted the weights of the model according to their strength. This allows us to get an idea about correlations between features and states. For example, features with a strong positive weight that are only active during state 1 are more strongly correlated with state 1 than other features in the model.

For example, looking at the top 15 weights from two different runs of our code on different pieces of our data set. Each of these are features that are only activated when the protein is labeled as state 1. It seems that certain features have a strong correlation with state 1.

This analysis is only partial, and we need to spend more time exploring the data to gain more significant insight into the features associated with each state.

Future Work

There are several directions we would like to explore:

- We would like to spend more time exploring choices of feature functions. We spent time building our framework for (x,y,z) tuples and then only had time to switch to angular data (chi) and hydrogen bonds.
- Scale: we want to be able to work with our entire dataset. We will have to implement the ability to stream data from files as it will not be able to fit into memory.
- Parallelize: many of the computations required for this model are easy to parallelize -- in particular, the dot-products required for computing log-linear probabilities and gradient ascent. This should reduce the time needed to run inference on the entire data set. We attempted to perform parallelization using the ocaml parmap library, but the results were unimpressive and require significant investigation to improve.
- Distribute: just as we can parallelize on one machine, we can distribute the calculation on many machines. Apache Spark, or our own K3 framework[6] would be good choices.
- Advanced gradient ascent techniques. Second order methods will help speed up convergence.
- L1 regularization of parameters. This will be helpful for feature selection, though it is not differentiable so we will need to explore new options for parameter estimation.

Bibliography

[1] Sutton, Charles, and Andrew McCallum. **"An introduction to conditional random fields."** *Machine Learning* 4.4 (2011): 267-373.

[2] Sarana Nutanong, Nick Carey, Yanif Ahmad, Alex S Szalay, and Thomas B Woolf. **Adaptive exploration for large-scale protein analysis in the molecular dynamics database.** In Proceedings of the 25th International Conference on Scientific and Statistical Database Management, page 45. ACM, 2013.

[3] Arvind Ramanathan, Andrej Savol, Virginia Burger, Shannon Quinn, Pratul K Agarwal, and Chakra Chennubhotla. **Statistical inference for big data problems in molecular biophysics**. Technical report, Oak Ridge National Laboratory (ORNL); Center for Computational Sciences, 2012.

[4] David E Shaw, Paul Maragakis, Kresten Lindorff-Larsen, Stefano Piana, Ron O Dror, Michael P Eastwood, Joseph A Bank, John M Jumper, John K Salmon, Yibing Shan, et al. **Atomic-level characterization of the structural dynamics of proteins.** Science, 330(6002):341–346, 2010.

[5]"Aprotinin" *Wikipedia: The Free Encyclopedia*. Wikimedia Foundation, Inc. 22 July 2004. Web. 19 Dec. 2013. <http://en.wikipedia.org/wiki/Aprotinin>

[6] Shyamshankar, P. C., Zachary Palmer, and Yanif Ahmad. "K3: Language design for building multi-platform, domain-specific runtimes." *First International Workshop on Cross-model Language Design and Implementation (XLDI)*. 2012.

[7] Razavian, Narges S., Hetunandan Kamisetty, and Christopher J. Langmead. "Learning generative models of molecular dynamics." *BMC genomics* 13.Suppl 1 (2012): S5.