

# Unifying secondary-structure, fold-recognition, and new-fold methods for protein structure prediction

Kevin Karplus, Rachel Karchin, Richard Hughey

`karplus@soe.ucsc.edu`

Center for Biomolecular Science and Engineering

University of California, Santa Cruz



# Outline of Talk

- 🦖 The folding problem
- 🦖 Iterative search and alignment (SAM-T2K)
- 🦖 Local structure prediction (predict-2nd)
- 🦖 Multi-track HMMs (SAM)
- 🦖 Fold-recognition (SAM-T02)
- 🦖 Fragment-packing (undertaker)
- 🦖 Results



# Folding Problem

The *Folding Problem*:

Given a protein expressed as a string  $A$  over the alphabet of 20 amino acids

( $A \in \{a, c, d, e, f, g, h, i, k, l, m, n, p, q, r, s, t, v, w, y\}^*$ ),  
figure out how it folds up in 3-space.

---

MTMSRRNTDA ITIHSILDWI EDNLESPLSL EKVSERSGYS KWHLQRMFKK  
ETGHSLGQYI RSRKMTEIAQ KLKESNEPIL YLAERYGFES QQTLTRTFKN  
YFDVPPHKYR MTNMQGESRF LHPLNHYNS



# Fold-recognition problem

The *Fold-recognition Problem*:

Given a protein expressed as a string  $A$  over the alphabet of amino acids (the *target* sequence) and a library of proteins with known 3-D structures (the *template* library), figure out which templates  $A$  match best, and align the target to the templates.

- 🦖 The backbone for the target sequence is predicted to be very similar to the backbone of the chosen template.



# Remote-homology Problem

The *Homology Problem*:

Given a protein expressed as a string  $A$  over the alphabet of amino acids (the *target* sequence), and a library of protein *sequences*, figure out which sequences  $A$  is similar to and align them to  $A$ .

- 🦖 This problem is fairly easy for recently diverged, very similar sequences, but difficult for more remote relationships.
- 🦖 No structure information is used, just sequence information.



# Secondary structure Prediction

- 🦖 Instead of predicting the entire structure, we can predict local properties of the structure.
- 🦖 What local properties do we choose?
- 🦖 We want properties that are well-conserved through evolution, easily predicted, and useful for finding and aligning templates.
- 🦖 One popular choice is a 3-valued helix/strand/other alphabet—we are investigating others.



# New-fold prediction

- 🦖 What if there is *no* template we can use?
- 🦖 We can try to generate many conformations of the protein backbone and try to recognize the most protein-like of them.
- 🦖 Search space is huge, so we need a good conformation generator and a cheap score function to evaluate conformations.



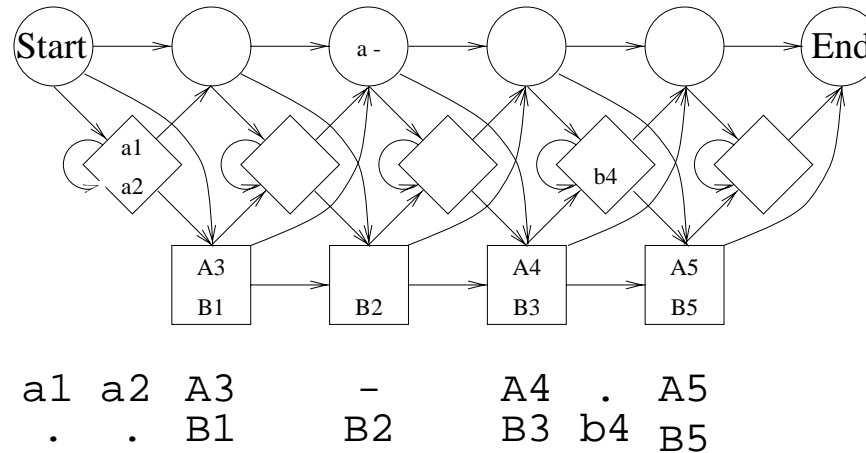
# Hidden Markov Models

- 🦖 A *hidden Markov Model* (HMM) is a finite-state machine with a probability for emitting each letter in each state, and with probabilities for making each transition between states.
- 🦖 Probabilities of letters sum to one for each state.
- 🦖 Probabilities of transitions out of each state sum to one for that state.
- 🦖 We also include *null states* that emit no letters, but have transition probabilities on their out-edges.





# Profile Hidden Markov Model



- 🦖 Circles are null states.
- 🦖 Squares are *match states*, each of which is paired with a null *delete state*. We call the match-delete pair a *fat state*.
- 🦖 Each fat state is visited exactly once on every path from Start to End.
- 🦖 Diamonds are *insert states*, and are used to represent possible extra amino acids that are not found in most of the sequences in the family being modeled.



# How is HMM built?

Overview of method for building a target HMM, given a single sequence (or a seed alignment):

**loop:** Construct a profile HMM with one fat state for each letter of sequence (or column of multiple alignment).

**find:** Find sequences in a large database of protein sequences that score well with  $M$ . This is the *training set*.

Retrain  $M$  (using forward-backward algorithm) to re-estimate all probabilities, based on the training set.

Make a multiple alignment (using Viterbi algorithm) of all sequences in the training set. The multiple alignment has one alignment column for each fat state of the HMM.

Repeat from *loop*, with thresholds in step *find* loosened.



# Predicting Local Structure

- 🦖 Want to predict some local property at each residue.
- 🦖 Local property can be emergent property of chain (such as being buried or being in a beta sheet).
- 🦖 Property should be conserved through evolution (at least as well as amino acid identity).
- 🦖 Property should be somewhat predictable (we gain information by predicting it).
- 🦖 Predicted property should aid in fold-recognition and alignment.
- 🦖 For ease of prediction and comparison, we look only at discrete properties (alphabets of properties).



# Using Neural Net

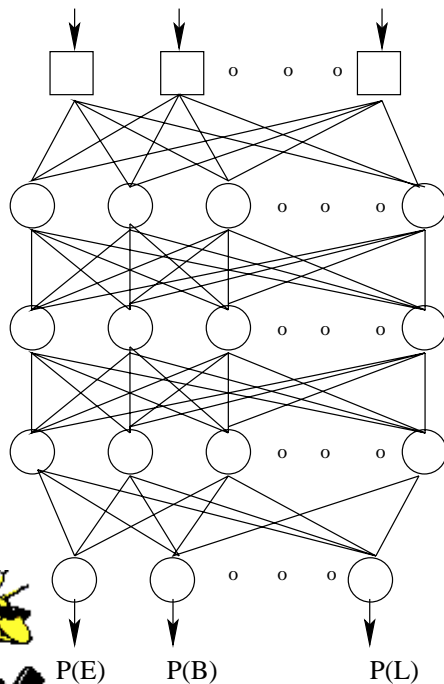
- 🦖 We use neural nets to predict local properties.
- 🦖 Input is profile with probabilities of amino acids at each position of target chain, plus insertion and deletion probabilities.
- 🦖 Output is probability vector for local structure alphabet at each position.
- 🦖 Each layer takes as input windows of the chain in the previous layer and provides a probability vector in each position for its output.



# Neural Net

Typical net has 4 layers and 6471 weight parameters:

input/pos	window	output/pos	weights
22	5	15	1665
15	7	15	1590
15	9	15	2040
15	13	6	1176



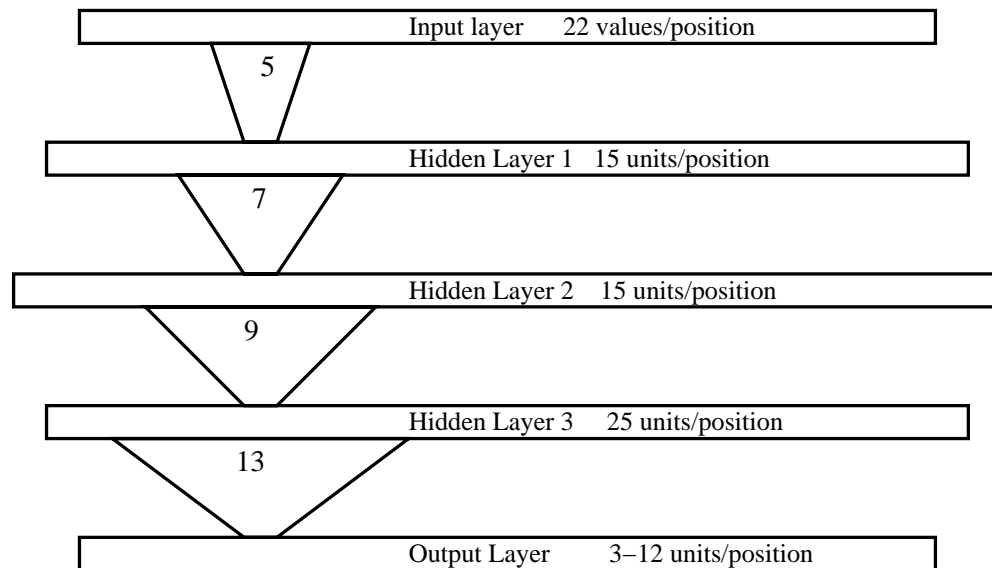
**Inputs**

**Hidden Layer 1**

**Hidden Layer 2**

**Hidden Layer 3**

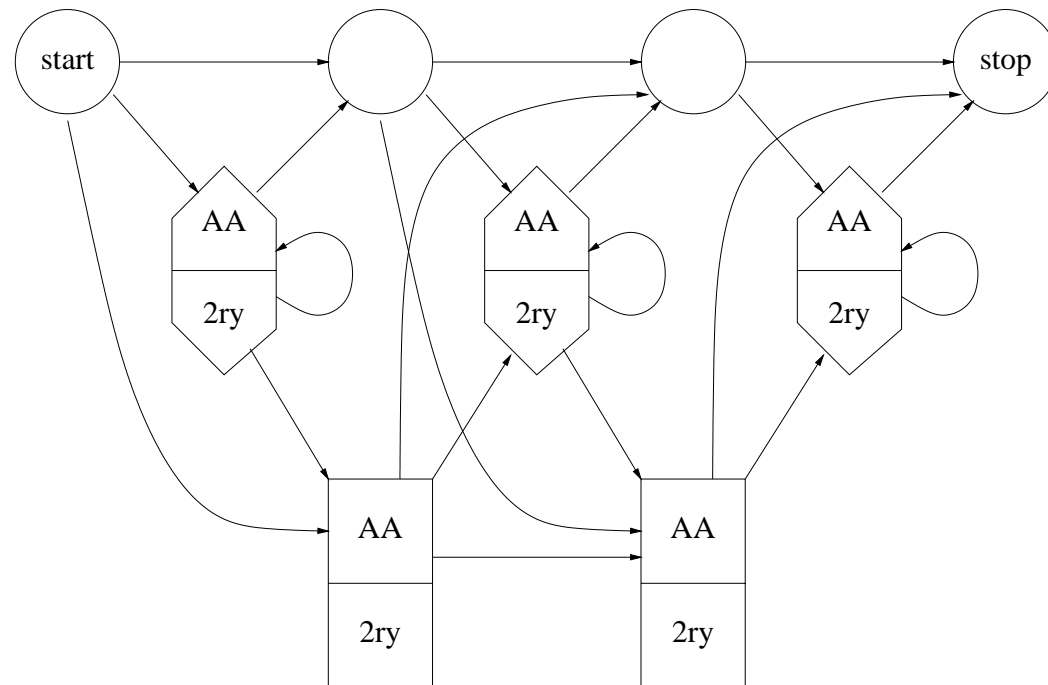
**Output Layer**



# Multi-track HMMs

We can also use alignments to build a two-track target HMM:

- 🦖 Amino-acid track (created from the multiple alignment).
- 🦖 Local-structure track (probabilities from neural net).
- 🦖 Can align template (AA+local) to target model.



# Target-model Fold Recognition

- 🦖 Find probable homologs of target sequence and make multiple alignment.
- 🦖 Make secondary structure probability predictions based on multiple alignment.
- 🦖 Build an HMM based on the multiple alignment and predicted 2ry structure (or just on multiple alignment).
- 🦖 Score sequences and secondary structure sequences for all proteins that have known structure.
- 🦖 Select the best-scoring sequence(s) to use as templates.



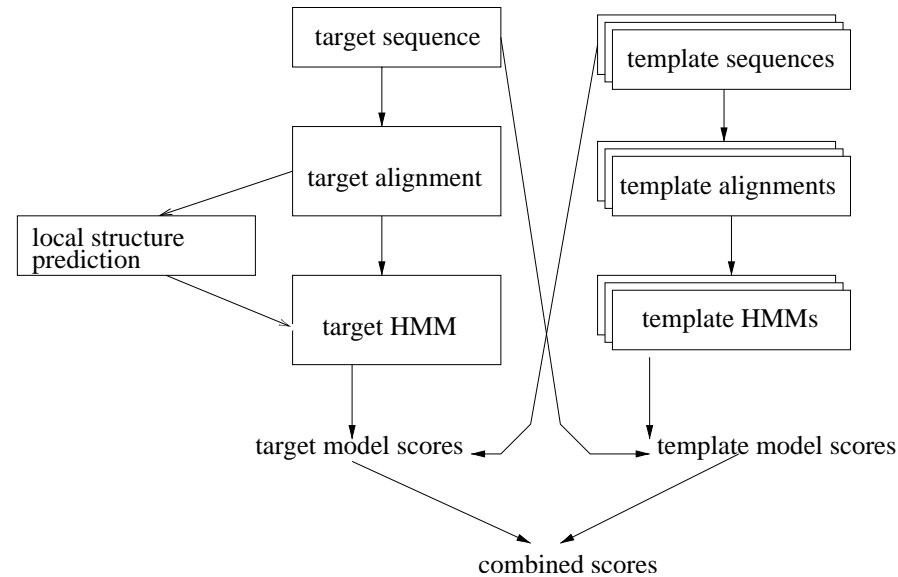
# Template-library Fold Recognition

- 🦖 Build an HMM for each protein in the template library, based on the template sequence (and any homologs you can find).
- 🦖 The library currently has over 7000 templates from PDB.
- 🦖 For the fold-recognition problem, structure information can be used in building these models (though we currently don't).
- 🦖 Score target sequence with all models in the library.
- 🦖 Select the best-scoring model(s) to use as templates.






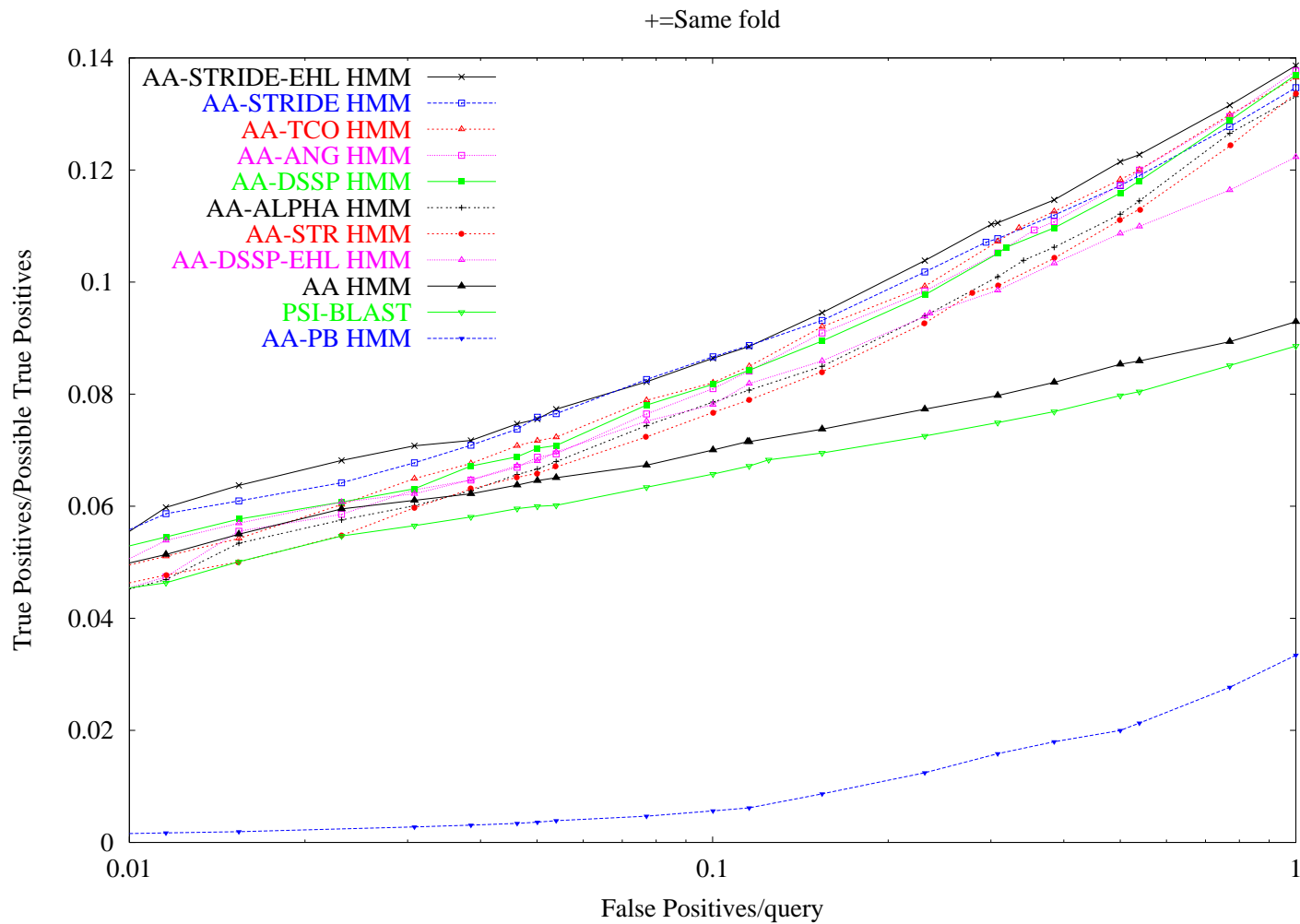
# Combined SAM-T02 method



- 🦖 Combine the scores from the template library search and the target library searches using different local structure alphabets.
- 🦖 Choose one of the many alignments of the target and template (whatever method gets best results in testing).

 <http://www.cse.ucsc.edu/research/compbio/HMM-apps/T02-query.html>

# Fold recognition results



# Fragment Packing

- 🦖 Fragment packing was introduced by Simon and Baker's Rosetta program.
- 🦖 It provides intelligent conformation generation for new folds.
- 🦖 Rosetta conformation is contiguous chain.
- 🦖 New conformations are created by randomly replacing fragment of backbone with different fragment (from library), keeping chain contiguous.
- 🦖 Stochastic search by simulated annealing.



# Undertaker

- 🦖 Undertaker is UCSC's attempt at a fragment-packing program.
- 🦖 Named because it optimizes burial.
- 🦖 Representation is 3D coordinates of all heavy atoms.
- 🦖 Can insert fragments (a la Rosetta) or full alignments—chain need not remain contiguous.
- 🦖 Conformations can borrow heavily from fold-recognition alignments, without having to lock in a particular alignment.
- 🦖 Use genetic algorithm with many conformation-change operators to do stochastic search.



# Fragfinder

Fragments are provided to undertaker from 3 sources:

- 🦖 Generic fragments (2-4 residues, exact sequence match) are obtained by reading in 500–1000 PDB files, and indexing all fragments.
- 🦖 Long specific fragments (and full alignments) are obtained from the various target and template alignments generated during fold recognition.
- 🦖 Medium-length fragments (9–12 residues long) for every position are generated from the HMMs with `fragfinder`, a new tool in the SAM suite.



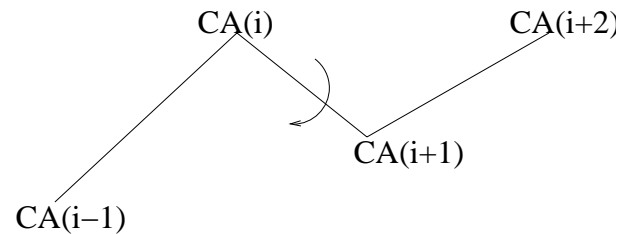
# Cost function

- 👉 Cost function is modularly designed—easy to add or remove terms.
- 👉 Main components are variants on burial cost:
  - *Burial* is the number of atoms whose centers are in a particular sphere.
  - We define points for each residue where burial is checked.
  - We use histograms of burial conditioned on residue type to convert burial to cost ( $-\log \text{Prob}$ ).
- 👉 Cost function can include predictions of local properties by neural nets.
- 👉 There are currently about 20 other cost function components (clashes, disulfides, contact order, radius of gyration, constraints, ...) that can be used.

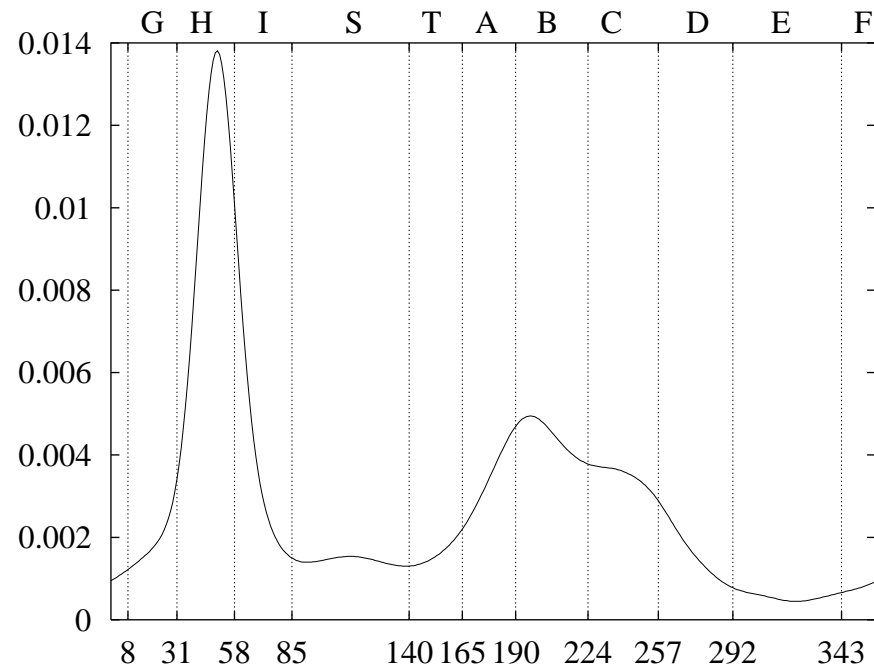


# Predicted $\alpha$ angle in cost

🦖 Current cost function includes a neural-net prediction of  $\alpha$  angle:

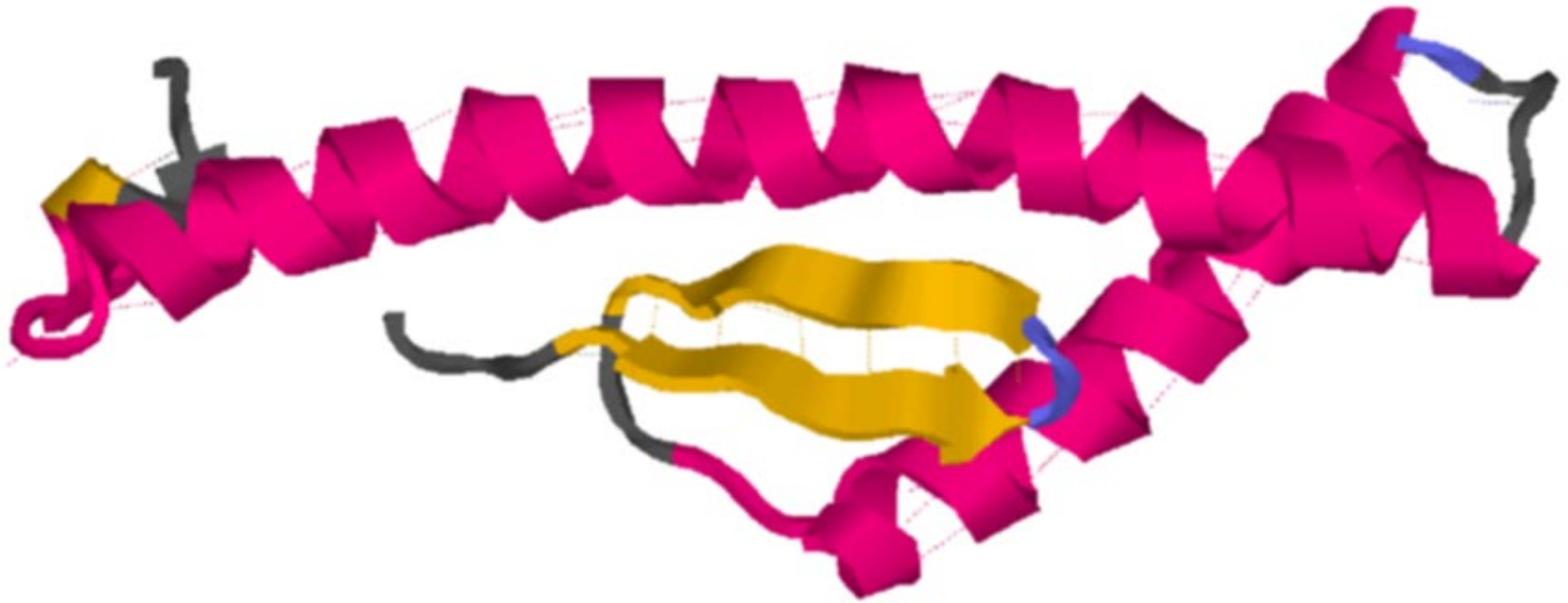


🦖 Neural net predicts discrete alphabet:



# Undertaker example: T0131

Ab-initio prediction:





# Undertaker example: T0147

Fold-recognition plus ab-initio prediction:



# Web sites

## **UCSC bioinformatics (research and degree programs) info:**

<http://www.soe.ucsc.edu/research/compbio/>

## **SAM tool suite info:**

<http://www.soe.ucsc.edu/research/compbio/sam.html>

**HMM servers:** <http://www.soe.ucsc.edu/research/compbio/HMM-apps/>

## **SAM-T02 prediction server:**

<http://www.soe.ucsc.edu/research/compbio/HMM-apps/T02-query.html>

## **These slides:**

<http://www.soe.ucsc.edu/~karplus/papers/sam-undertaker.pdf>

