Mining and Learning with Graphs (MLG 2008)
Helsinki, July 4-5, 2008

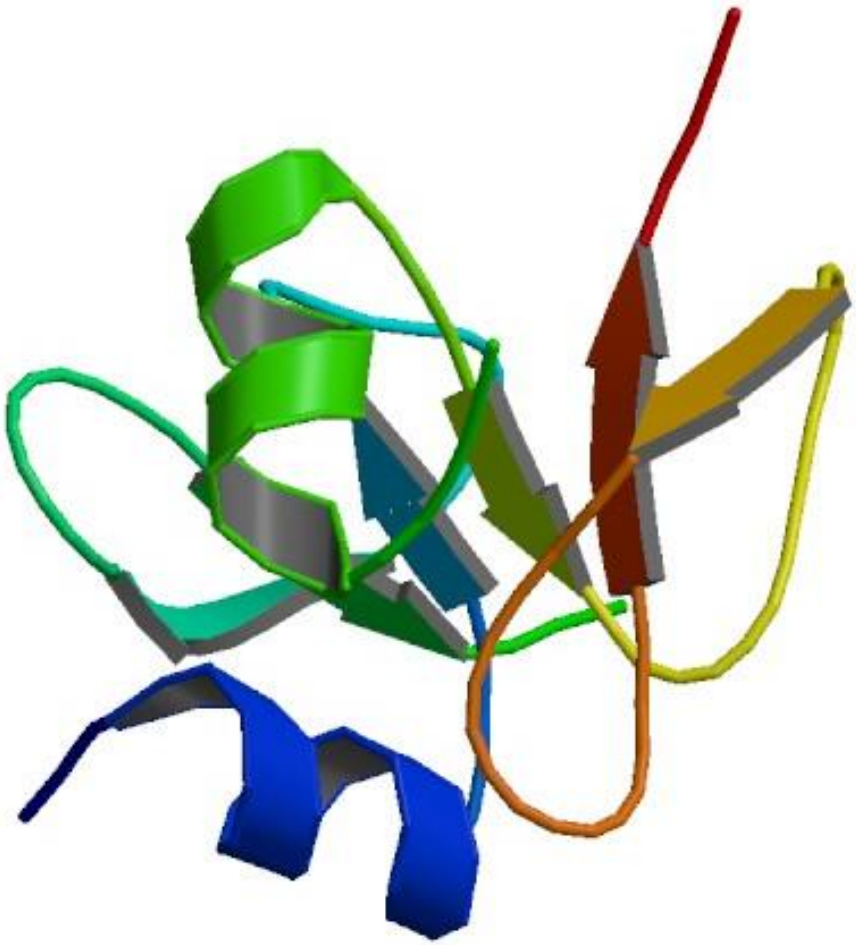
Markov Logic improves protein β-partners prediction

Marco Lippi, Paolo Frasconi

Machine Learning & Neural Networks group
Università degli Studi di Firenze

3D Protein structure prediction

- PDB entry 1FD4



- Very complex structured and relational problem
- Many **link prediction** tasks

$\Downarrow$

- Contact maps

  [Casadio et al. 2000, Pollastri 2006]

- Cysteine connectivity

  [Vullo & Frasconi 2004,

  Taskar et al. 2005]

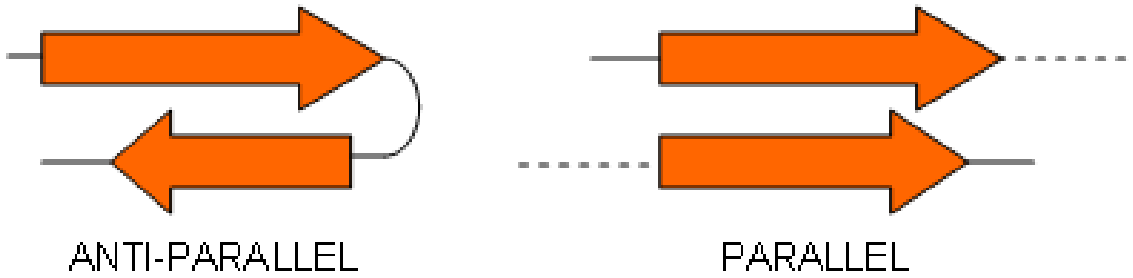
- **β-partners prediction**

  [Baldi & Cheng 2005]

β-partners: a link prediction task

β-sheets:

- flat conformations of two or more extended strands (may be parallel or anti-parallel)



β-partners prediction within a protein sequence:

- **sub-problem** of contact map prediction, where contact matrix is restricted to residues belonging to β-sheets.
- **link-prediction** problem in a graph, in which β-residues are nodes, and the edges (to be predicted) represent the contacts.

β-partners: a link prediction task

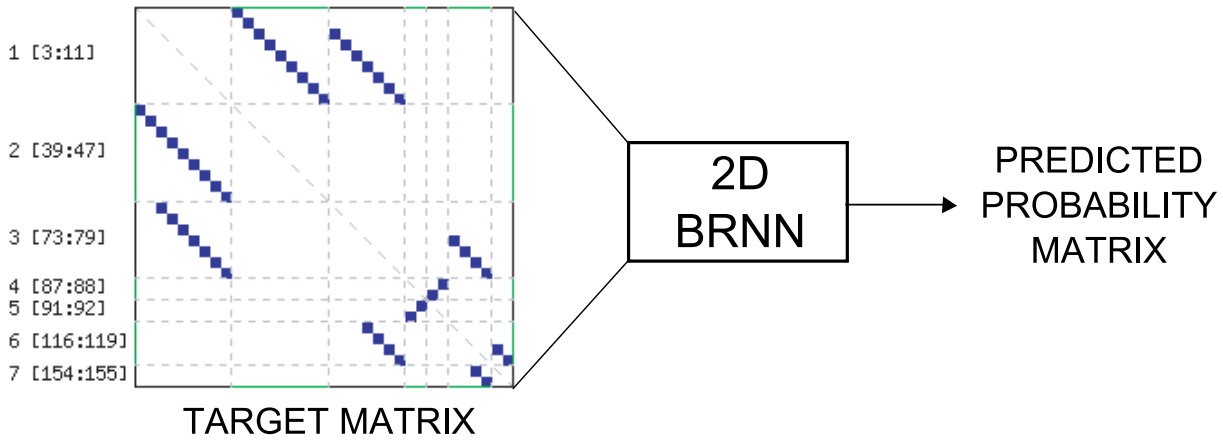
Early approach by [Baldi et al. 2000]:

- feedforward neural networks trained as binary classifiers on residue pairs (i, j)
- window around residues as input to the network
- data set highly **imbalanced**: 37,000 positive cases vs. 44,000,000 negative ones
- not taking into account relations between targets

State-of-the-art architecture: BetaPro

[Baldi & Cheng 2005] set up a two-stage architecture (BetaPro):

- a 2D recursive neural network (2D-RNN) is trained using a grid structure with the binary contact matrix as target
- a post-processing non-adaptive phase rearranges β-links by graph matching and pseudoenergy minimization
- secondary structure is assumed to be **known**: residues are already assigned to one of three classes (α-helix, β-sheet, coil)



β-partners: a link prediction task

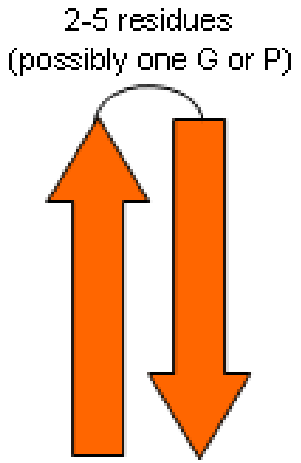
β-partners follow common conformation patterns

- non independent predictions $\longrightarrow$ collective classification
- discriminative learning is appropriate
- first-order logic is a straightforward way to build the model
- background knowledge (with noise) $\longrightarrow$ uncertainty

  $\Rightarrow$ use of **Markov Logic**

Example: β-**hairpin motif**

$$\text{LastOfStrand(r1,s1)} \wedge \text{FirstOfStrand(r2,s2)} \wedge \text{DistanceLessThanSix(s1,s2)} \wedge \text{GlycineWithin(s1,s2)} \Rightarrow \text{Partners(r1,r2)}$$



Markov Logic

Markov Logic Networks (MLNs), introduced by [Domingos & Richardson 2006], combine in a single representation:

- first-order logic
- probabilistic graphical models

A MLN can be seen as a **template** for constructing Markov Random Fields, given:

- a set of first-order formulae $f_1, \ldots, f_N$
- a database of constants $C_1, \ldots, C_K$

Uncertainty is modeled attaching **weights** to first-order formulae.

Markov Logic

In the setting of **discriminative** learning, an MLN is a model for the conditional distribution of a set of **query atoms** Y given a set of **evidence atoms** X, expressed by a log-linear function:

$$P(Y = y|X = x) = \frac{exp\left(\sum_{F_i \in F_y} w_i n_i(x, y)\right)}{Z_x}$$

where

- w_i : real-valued weight attached to formula F_i
- F_y : set of formulas that contain query atoms
- $n_i(x, y)$: number of groundings of F_i satisfied in world (x, y)
- Z_x : normalization factor

Markov Logic

Discriminative learning:

- maximizes the conditional log-likelihood (CLL) $\log P(y|x)$
- requires inference on the Markov Random Field generated by the database of constants.

Inference algorithms:

- **MC-SAT** $\longrightarrow$ compute conditional probabilities of query atoms
- **MaxWalkSAT** $\longrightarrow$ compute maximum probability configuration of query atoms (MAP)

Curse of dimensionality

Dilemma

Curse of dimensionality...

Nonlinear model with number of parameters exponential in k

```
Feature_1(x,+f_1) ^ ... ^ Feature_k(x,+f_k) => QueryPredicate(x)
```

$$\Downarrow$$

```
Feature_1(x,c_11) ^ ... ^ Feature_k(x,c_1k) => QueryPredicate(x)

...

Feature_1(x,c_n1) ^ ... ^ Feature_k(x,c_mk) => QueryPredicate(x)
```

Expressivity of the model

Dilemma

Curse of dimensionality...

Linear model with number of parameters linear in k

```
Feature_1(x,+f_1) => QueryPredicate(x)
                  ...
Feature_k(x,+f_k) => QueryPredicate(x)
```

Markov Logic Networks with grounding-specific weights

Solution: grounding-specific weights

We propose a **re-parametrization** of MLNs by computing each weight w_i as a function of variables of each specific grounding c_{ij}:

Standard MLN

$$P(Y = y|X = x) = \frac{exp\left(\sum_{F_i \in F_y} w_i n_i(x,y)\right)}{Z_x}$$

MLNs with grounding-specific weights

$$P(Y = y|X = x) = \frac{exp\left(\sum_{F_i \in F_y} \sum_j w_i(c_{ij},\theta_i) n_{ij}(x,y)\right)}{Z_x}$$

Markov Logic Networks with grounding-specific weights

The weights $w_i(c_{ij}, \theta_i)$ can be computed in several ways

- using Multi-Layered Perceptrons (MLPs), by taking as input an encoding of the grounding c_{ij}

- **Inference** algorithms do not change.
- **Learning** algorithm can implement gradient descent:

$$\frac{\partial P_w(y|x)}{\partial \theta_k} = \frac{\partial P_w(y|x)}{\partial w_i} \frac{\partial w_i}{\partial \theta_k}$$

where the **first** term is computed by MLN inference and the **second** one is computed by backpropagation.

Markov Logic Networks with grounding-specific weights

In the case of MAP inference:

$$\frac{\partial P_w(y|x)}{\partial w_i} = n_i(x, y) - n_i(x, y_w^*)$$

where $n_i(x, y_w^*)$ is the number of satisfied groundings in maximum probability world (x, y_w^*).

The gradient is equal to **0** if the maximum probability state of the grounding matches its target, and **+1** or **-1** if they disagree.

MAP inference **actively selects** examples for the MLP training.

The data set

We used the same data set as in [Baldi & Cheng 2005]:

- 916 sequences
- 48,996 β-residues
- 31,638 β-residue pairs ($\sim$ 3,000,000 negative examples)
- 10-fold cross validation

Four query predicates:

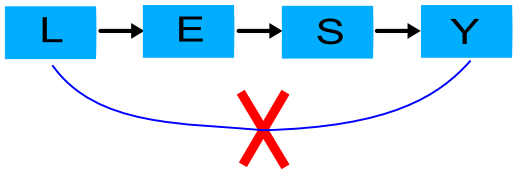
- Partners(residue,residue)
- StrandContact(strand,strand)
- ParallelContact(strand,strand)
- AntiParallelContact(strand,strand)

The model: hard rules

Some basic properties...

- Anti-reflexivity:
  $$!Partners(r,r)$$
- Symmetry:
  $$Partners(r1,r2) \Rightarrow Partners(r2,r1)$$

- No partners belonging to same strand:
  $$BelongsToStrand(r1,s) \wedge BelongsToStrand(r2,s) \Rightarrow !Partners(r1,r2)$$



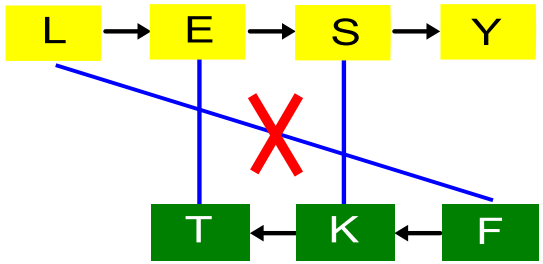
- A residue can't have two partners belonging to same strand:
  $$Partners(r1,r2) \wedge BelongsToStrand(r2,s) \wedge BelongsToStrand(r3,s) \Rightarrow !Partners(r1,r3)$$

The model: more complex rules

Modeling more complex patterns...

- No crossing edges:

  $$\text{Partners}(i,j) \wedge \text{Partners}(i+1,j+1) \Rightarrow \text{!Partners}(i-1,j+2)$$



  - Anti-transitivity of coarse contacts:

    $$\text{StrandContact}(si,sj) \wedge \text{StrandContact}(sj,sk) \Rightarrow \text{!StrandContact}(si,sk)$$

  - Adjacency in parallel sheets:

    $$\text{Partners}(i,j) \wedge \text{ParallelContact}(si,sj) \Rightarrow \text{Partners}(i+1,j+1)$$

  - Adjacency in anti-parallel sheets:

    $$\text{Partners}(i,j) \wedge \text{AntiParallelContact}(si,sj) \Rightarrow \text{Partners}(i+1,j-1)$$

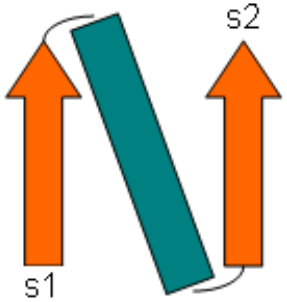
The model: more complex rules

- β-hairpin motif:

$$\begin{aligned}&\text{LastOfSeg(r1,s1)} \wedge \text{FirstOfSeg(r2,s2)} \wedge \\ &\quad \text{DistanceLessThanSix(s1,s2)} \wedge \\ &\quad\quad \text{GlycineWithin(s1,s2)} \\ &\quad\quad\quad \Rightarrow \text{Partners(r1,r2)}\end{aligned}$$

- β-α-β motif:

$$\begin{aligned}&\text{Length(s1,n)} \wedge \text{Length(s2,n)} \wedge \\ &\quad \text{HelixWithin(s1,s2)} \wedge \text{FirstOfStr(f1,s1)} \wedge \\ &\quad\quad \text{FirstOfStr(f2,s2)} \Rightarrow \text{Partners(f1,f2)}\end{aligned}$$



Plugging in BetaPro probabilities

In our experiments we plugged-in grounding-specific weights from BetaPro first-stage (2D-RNN) for the basic rule:

$$\text{Window(i,wi)} \wedge \text{Window(j,wj)} \Rightarrow \text{Partners(i,j)}$$

- $w_i(c_{ij}, \theta_i) = \text{logit}(p_{ij})$
- $p_{ij} \in [0, 1]$ is the probability computed by BetaPro.

The other weights were learned by stochastic gradient ascent

- each protein produces a different Markov Random Field
- the total number of weights (rules) of the MLN is 66

The model: a second-stage MLN

Problem

- Some rules satisfied by making the antecedent false:

$$\text{Partners(i,j)} \wedge \text{ParallelContact(si,sj)} \Rightarrow \text{Partners(i+1,j+1)}$$

- This can produce an **under-prediction** of partners (low recall)

Solution

- Second refinement MLN: links predicted at first level become evidence (introduced as new "CandidatePartners" predicate)

$$\text{CandidatePartners(i,j)} \wedge \text{ParallelContact(si,sj)} \Rightarrow \text{Partners(i+1,j+1)}$$

Results obtained on 10-fold cross validation

- Measuring performance is not easy
- [Baldi & Cheng 2005] use $F_1 = \frac{2PR}{P+R}$ at residue level
- Need more detailed measures
- Protein-level scores are usual in these tasks

We consider **coarse** (strand-strand) predictions and measure the percentage C_x of correct **proteins** with less than x% missed edges

Results obtained on 10-fold cross validation

Residue-level predictions

	BetaPro	MLN
F_1	40.9	43.0

Coarse-level predictions

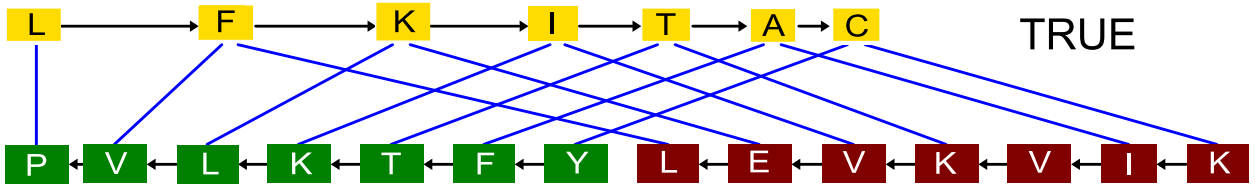
	BetaPro	MLN
C_{10}	46.6	54.8
C_{20}	84.3	87.3
C_{50}	100.0	100.0

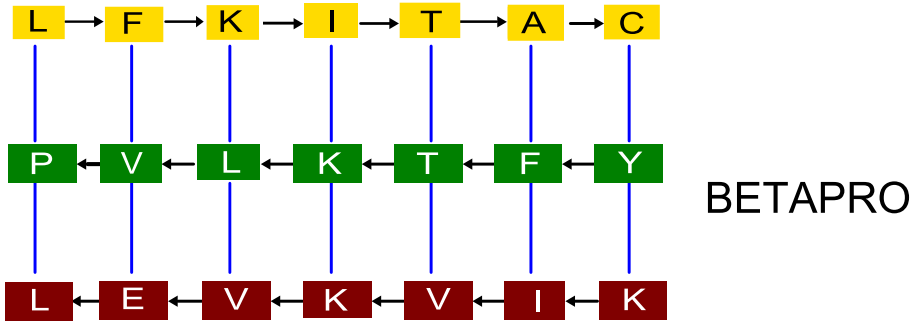
Differences are statistically significant with p-value < 0.01.

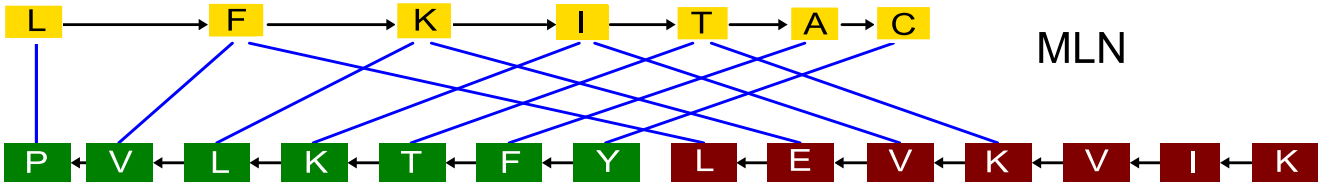
Examples: comparison with BetaPro

Mistakes in coarse map (1)

- PDB entry 1B33N



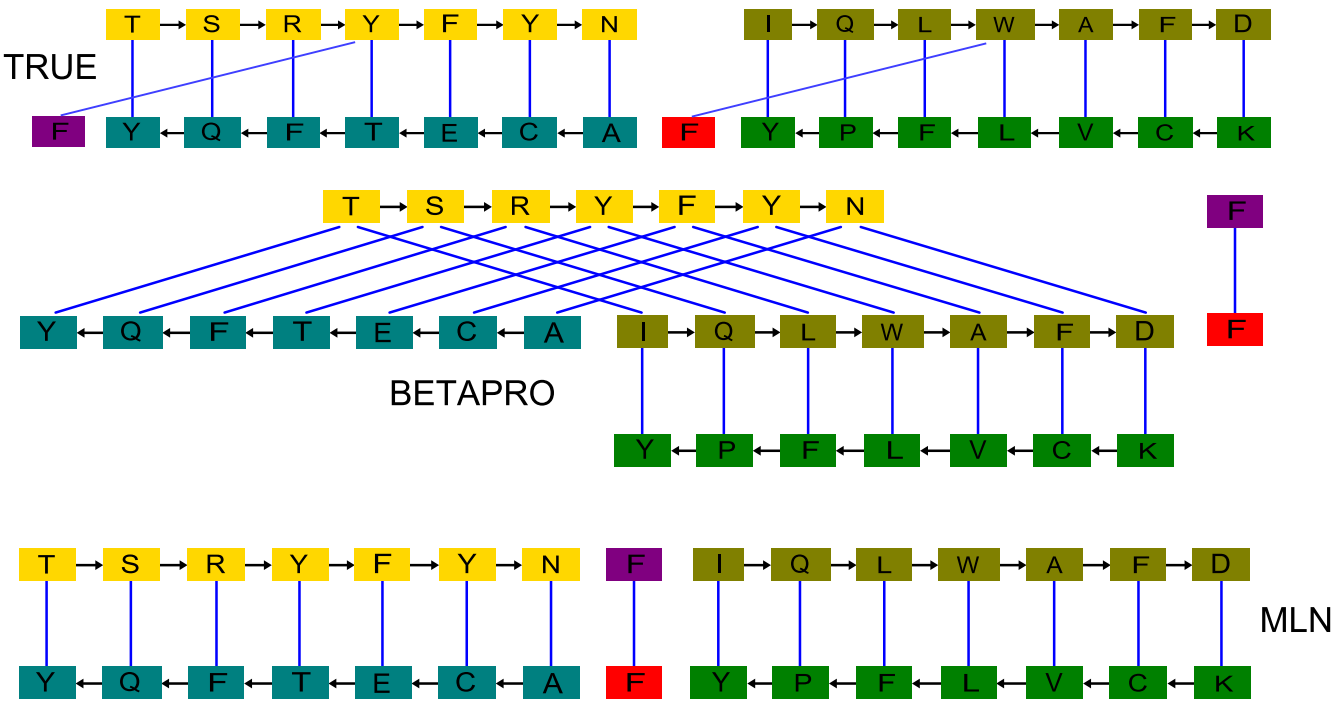




Examples: comparison with BetaPro

Mistakes in coarse map (2)

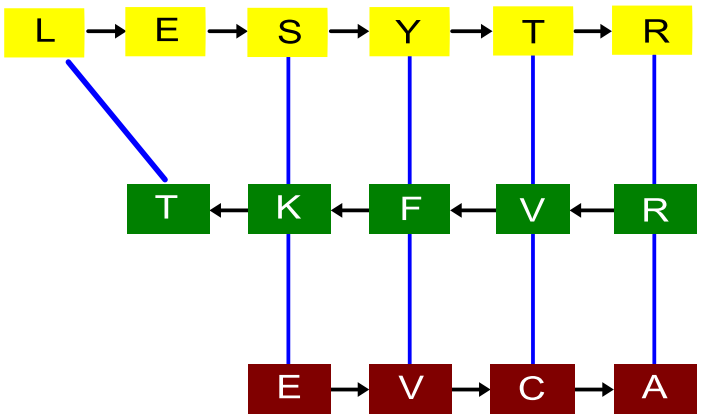
- PDB entry 1BIKA



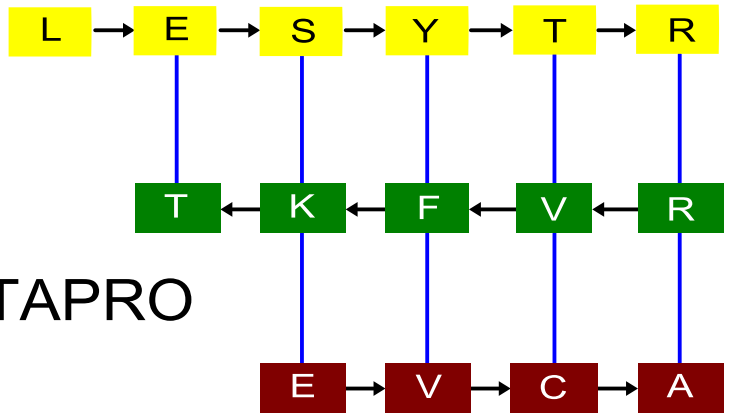
Examples: comparison with BetaPro

Gaps

- PDB entry 1ESRA



TRUE and MLN



BETAPRO

Conclusions and Future Work

- Encouraging results, but there is still a lot of work to be done
- Use of Multi-Layered Perceptrons for predicting ground-specific weights, performing **joint training** with MLN
- **Multitask learning** scheme: β-partners jointly predicted with secondary structure and/or solvent accessibility
- Measure improvement on 3D reconstruction
- Application to many other bioinformatics problems (e.g. metal binding sites)