Ensemble methods for gene/protein function prediction

Giorgio Valentini
valentini@dsi.unimi.it

DSI - Dipartimento di Scienze dell’Informazione

Università degli Studi di Milano
Outline

• Gene Function Prediction (GFP)
• The Gene Ontology
• Computational approaches to GFP
• Ensemble methods
• Ensemble methods for GFP
• A case-study
Gene function prediction can be formalized as a supervised machine learning problem.
Motivation

- Increasing sets of genomes available, but functions of most genes/gene products are unknown or only partially known
- Many biological questions can be answered if we understand the role of a protein in a biological process, how it interacts with other proteins or where it operates within a cell
- This biological problem raises challenging problems from a machine learning standpoint.
Computational prediction supports biological gene function prediction

Biological genome-wide gene function prediction through direct experimental assays is costly and time-consuming

Computational prediction methods assist the biologist to:

• Suggest a restricted set of candidate functions that can be experimentally verified
• Directly generate new hypotheses
• Guide the exploration of promising hypotheses
Characteristics of the gene function prediction problem

• Large number of functional classes: hundreds (FunCat) or thousands (Gene Ontology (GO)) : large multi-class classification
• Multiple annotations for each gene: multilabel classification
• Different level of evidence for functional annotations: labels at different level of reliability
• Hierarchical relationships between functional classes (tree forest for FunCat, direct acyclic graph for GO): hierarchical relationships between classes (structured output)
• Class frequencies are unbalanced, with positive examples usually largely lower than negatives: unbalanced classification
• The notion of “negative example” is not univocally determined: different strategies to choose negative examples
• Multiple sources of data available: each type captures specific functional characteristics of genes/gene products: multi-source classification
• Data are usually complex (e.g. high-dimensional) and noisy: classification with complex and noisy data
The Gene Ontology

The Gene Ontology (GO) project began as a collaboration between three model organism databases, FlyBase (Drosophila), the Saccharomyces Genome Database (SGD) and the Mouse Genome Database (MGD), in 1998. Now it includes several of the world's major repositories for plant, animal and microbial genomes.

The GO project has developed three structured controlled vocabularies (ontologies) that describe gene products in terms of their associated biological processes, cellular components and molecular functions in a species-independent manner.
The Gene Ontology (GO) is actually three Ontologies

1) Molecular Function
GO term: Malate dehydrogenase activity
GO id: GO:0030060
(S)-malate + NAD(+) = oxaloacetate + NADH.

2) Biological Process
GO term: tricarboxylic acid cycle
Synonym: Krebs cycle
Synonym: citric acid cycle
GO id: GO:0006099

3) Cellular Component
GO term: mitochondrion
GO id: GO:0005739
Definition: A semiautonomous, self replicating organelle that occurs in varying numbers, shapes, and sizes in the cytoplasm of virtually all eukaryotic cells. It is notably the site of tissue respiration.

(Slide downloaded from www.geneontology.org)
**GO term: tricarboxylic acid cycle**

**GO Accession : GO:0006099**

**Ontology : Biological Process**

**Definition**

A nearly universal metabolic pathway in which the acetyl group of acetyl coenzyme A is effectively oxidized to two CO2 and four pairs of electrons are transferred to coenzymes. The acetyl group combines with oxaloacetate to form citrate, which undergoes successive transformations to isocitrate, 2-oxoglutarate, succinyl-CoA, succinate, fumarate, malate, and oxaloacetate again, thus completing the cycle. In eukaryotes the tricarboxylic acid is confined to the mitochondria.

998 annotated gene products
TCA cycle in the GO DAG
Relationships between terms in the GO

The ontologies of GO are structured as a directed acyclic graph (DAG) \( G=\langle V,E \rangle \)

\[ V = \{ t \mid \text{terms of the GO} \} \quad E = \{ (t, u) \mid t \in V \text{ and } t \in V \} \]

Relations between GO terms are also categorized and defined:

- *is a* (subtype relations)
- *part of* (part-whole relations)
- *regulates* (control relations)
**Is a relation**

*If we say A is a B, we mean that node A is a subtype of node B.*

For example, mitotic cell cycle is a cell cycle, or lyase activity is a catalytic activity.

The *is a* relation is **transitive**, which means that if A is a B, and B is a C, we can infer that A is a C.

E.g.:
Part of relation

The relation part of represents *part-whole relationships in the GO.* A is part of B means that wherever B exists, it is as part of A, and the presence of the B implies the presence of A. However, given the occurrence of A, we cannot say for certain that B exists:

The part of relation is **transitive:**
Regulates relation

If we say that \( A \) regulates \( B \) we mean that \( A \) directly affects the manifestation of \( B \), i.e. the former regulates the latter.

For example, the target of the regulation may be another process—for example, regulation of a pathway or an enzymatic reaction—or it may be a quality, such as cell size or pH.

Analogously to part of, this relation is used specifically to mean necessarily regulates:

In general regulates is not transitive
A visualization of the GO DAG through OBO-Edit
GO DAG of the BP ontology (*S. cerevisiae*)

1074 GO classes (nodes) connected by 1804 edges

Graph realized through *HCGene* (Valentini, Cesa-Bianchi, *Bioinformatics* 24(5), 2008)
Evidence codes

Evidence codes indicate how the annotation to a particular term is supported:

Experimental Evidence Codes:
an experimental assay has been used for the annotation

Author statement codes:
indicate that the annotation was made on the basis of a statement made by the author(s) in the reference cited.

Curatorial evidence codes:
an annotations inferred by a curator from other GO annotations

Computational analysis evidence codes:
based on an in silico analyses manually reviewed

Automatically-assigned Evidence Codes:
based on an in silico analyses not manually reviewed
Groups of evidence codes

**Experimental Evidence Codes**
- EXP: Inferred from Experiment
- IDA: Inferred from Direct Assay
- IPI: Inferred from Physical Interaction
- IMP: Inferred from Mutant Phenotype
- IGI: Inferred from Genetic Interaction
- IEP: Inferred from Expression Pattern

**Author Statement Evidence Codes**
- TAS: Traceable Author Statement
- NAS: Non-traceable Author Statement

**Curator Statement Evidence Codes**
- IC: Inferred by Curator
- ND: No biological Data available

**Computational Analysis Evidence Codes**
- ISS: Inferred from Sequence or Structural Similarity
- ISO: Inferred from Sequence Orthology
- ISA: Inferred from Sequence Alignment
- ISM: Inferred from Sequence Model
- IGC: Inferred from Genomic Context
- RCA: inferred from Reviewed Computational Analysis

**Automatically-assigned Evidence Codes**
- IEA: Inferred from Electronic Annotation

**Obsolete Evidence Codes**
- NR: Not Recorded
Computational approaches to GFP

- Flat (e.g. by homology-based or machine learning methods) (Tetko et al. 2008; Chitale et al. 2009)
- Functional association (linkage) networks (Karaoz et al, 2004; Tsuda et al, 2005; Chua et al, 2007)
- Methods based on the joint kernelization of both the input variables and the output (tree or DAG structured) (Astikainen et al. 2008, Sokolov and Ben-Hur, 2010)
- Hierarchical ensemble methods (Barutcuoglu et al. 2006; Obozinski et al, 2008; Schietgat et al. 2010)
A brief parenthesis on ensemble methods

Ensembles are sets of learning machines that work together to solve a machine learning problem

E.g.:

Ensemble methods are one of the main topics in machine learning research
Why should we use ensembles?

From *empirical studies*: ensembles are often much more accurate than individual learning machines (Freund & Schapire (1995), Bauer & Kohavi (1999), Dietterich (2000), …).

Different *theoretical explanations* proposed to justify their effectiveness (Kittler (1998), Schapire et al. (1998), Kleinberg(2000), Allwein et al. (2000), …).

Very fast development of *computer technology*: availability of very fast computers and networks of workstations at a relatively low cost.
An example: majority voting ensembles

A dichotomous classification problem and L classifiers with error < 0.5

The resulting majority voting ensemble has an error lower than the single classifier

For instance, 21 classifiers, \( p < 0.3 \), probability of error of each classifier

\[
P_{\text{error}} = \sum_{i=\lfloor L/2 \rfloor}^{L} \binom{L}{i} p^i (1 - p)^{L-i} \Rightarrow P_{\text{error}} = 0.026 \ll p
\]

Condorcet Jury Theorem (XVIII century): the judgment of a committee is superior to those of individuals, (if their competence is reasonable, e.g. \( p < 0.5 \))
A lot of methods …

- Majority and weighted voting (Perrone and Cooper, 1993, Lam & Sue, 1997)
- Minimum, maximum, average and OWA aggregating operators (Kittler, 1998, Kuncheva, 1997)
- Bayesian (Naïve-Bayes) decision rule (Xu, 1992)
- Fuzzy aggregation (Cho & Kim, 1995, Wang et al., 1998)
- Decision templates (Kuncheva et al., 2001)
- Bagging (Breiman, 1998)
- Boosting (Freund & Schapire, 1998)
- Random forests (Breiman, 2001)
- ECOC ensembles (Dietterich and Bakiri, 1995)

See *L. Kuncheva Combining Pattern Classifiers, Wiley, 2004* for a good review book on ensemble methods
Hierarchical ensemble methods

*They are in general characterized by a two-step strategy:*

1. Flat learning of the protein function on a per-term basis (a set of independent classification problems)
2. Combination of the predictions by exploiting the relationships between terms that govern the hierarchy of the functional classes.

The term *ensemble* raises from the fact that a set of learning machines in someway combine their output.

In principle any supervised learning algorithm can be used for step 1.

Step 2 requires a proper combination of the predictions made at step 1.
Bayesian hierarchical multi-label prediction of gene function
(Barutcuoglu, Schapire and Troyanskaya, 2006)

Main ideas:

- *Flat prediction* of each term/class (possibly inconsistent)

- *Bayesian hierarchical combination* scheme to allow collaborative error-correction over all nodes

Basic notation:

\[ y_i : \text{binary membership to class } i \]

\[ \hat{y}_i : \text{classifier output for class } i, \quad 1 \leq i \leq N \]
Bayesian correction of classifier outputs

Goal: given a set of (possibly inconsistent) $\hat{y}_i$
find the set of consistent $y_i$ that maximize:

$$P(y_1, \ldots, y_N | \hat{y}_1, \ldots, \hat{y}_N) = \frac{P(\hat{y}_1, \ldots, \hat{y}_N | y_1, \ldots, y_N)P(y_1, \ldots, y_N)}{Z}$$

Direct solution is too hard … (exponential in time w.r.t to the number of nodes)

Proposed solution: a Bayesian network structure that exploits the relationships between functional classes.
The proposed Bayesian network

1. $y_i$ nodes conditioned to their children (structure constraints)
2. $\hat{y}_i$ nodes conditioned on their label $y_i$ (Bayes rule)
3. $\hat{y}_i$ are independent from both $\hat{y}_j, j \neq i$ and $y_j, j \neq i$ given $y_i$

This allows us to simplify the Bayesian equation:

- from 1: $P(y_1, \ldots, y_N) = \prod_{i=1}^{N} P(y_i | ch(y_i))$
- from 2,3: $P(\hat{y}_1, \ldots, \hat{y}_N, | y_1, \ldots, y_N) = \prod_{i=1}^{N} P(\hat{y}_i | y_i)$
Estimation of the probabilities

Estimation of

\[ P(y_1, \ldots, y_N) = \prod_{i=1}^{N} P(y_i \mid ch(y_i)) \]

Can be inferred from training labels by counting

Estimation of

\[ P(\hat{y}_1, \ldots, \hat{y}_N, \mid y_1, \ldots, y_N) = \prod_{i=1}^{N} P(\hat{y}_i \mid y_i) \]

Can be inferred by validation during training, by modeling the distribution of \( \hat{y}_i \) outputs over positive and negative examples.
E.g.: a parametric gaussian model:
Implementation of the method

- **Bagged ensemble of SVMs** (10 SVMs) trained at each node (see next slide …)
- Median values of their outputs on out-of-bag examples have been used to estimate means and variances for each class.
- Mean and variances have been used as parameters of the gaussian models used to estimate the conditional probabilities $P(\hat{y}_i | y_i = 1)$ and $P(\hat{y}_i | y_i = 0)$

The prediction of the label for each class $i$ is then computed as follows:

$$y_i^* = \arg \max_{y_i \in \{0,1\}} P(y_i | \hat{y}_1, \ldots, \hat{y}_N) = \frac{\prod_{j=1}^N P(\hat{y}_j | y_i) P(y_i | \text{child}(y_i))}{Z}$$
Bagging (Bootstrap aggregating)  
\textit{(Breiman, 1996)}

Input: \( Z = \{ (x_1, y_1),..., (x_m, y_m) \} \) \hspace{1em} \( y_i \in Y = \{ 1, ..., k \} \) \hspace{1em} LearnAlg

\textbf{Do for} \( t=1 \) to \( T \):

1. Bootstrap replicate \( Z_t \) from \( Z \)  \hspace{2em}  \text{(random sampling with replacement)}
2. Get back an hypothesis \( h_t: X \rightarrow Y \)
   \( h_t = \text{LearnAlg}(Z_t) \)

\textbf{end for}

Output the final hypothesis by aggregation and majority voting:

\[
h_{\text{fin}}(x) = \arg \max_{y \in Y} \sum_{t=1}^{T} \begin{cases} 
1 & \text{if} \quad h_t(x) = y \\
0 & \text{otherwise}
\end{cases}
\]

- Effective with unstable algorithms
- It reduces the variance component of the error
Results on a sub-hierarchy of the BP GO ontology

- 105 terms/nodes of the GO BP (model organism *S. cerevisiae*)
- 4 types of data integrated through Vector Space Integration
- Hierarchical approach improves AUC results on 93 of the 105 GO terms
- Darker blue: improvements; darker red: deterioration; white: no change.
Hierarchical corrections provide consistent predictions

Prediction of the gene YNL261W: subunit of the origin recognition complex that binds to replication origin and directs DNA replication (Bell, 2002).
Improvements of the algorithm

Guan, Myers, Hess, Barutcuoglu, Caudy and Troyanskaya, 2008

• Two variants of the Bayesian integration:
  – HIER-MB: Hierarchical Bayesian combination involving nodes in the Markov Blanket
  – HIER-BFS: Hierarchical Bayesian combination involving nodes the 30 first nodes visited through a Breadth-First-Search (BFS) in the GO graph

• Integration of 3 classifiers selected through held-out examples

• Application to the prediction of *M. musculus* (mouse) gene functions
HIER-MB: Hierarchical Bayesian combination involving nodes in the Markov Blanket
HIER-BFS: Hierarchical Bayesian combination using the first 30 BFS nodes
Ensemble of 3 classifiers
selected through held-out examples

Data pre-processing
- Gene expression
  Microarray
- Protein sequence and domain information
- Protein-protein interactions
- Phenotype profiles
- Phylogenetic profiles
- Disease profiles (using homology)

Three classifiers
1. Single SVM for GO term X
2. Hierarchical correction
3. Bayesian integration of individual datasets

Ensemble
- Held-out set validation to choose the best classifier for each GO term

Performance metrics:
- True positive rate
- False positive rate
- Final ensemble predictions for GO term X

G.Valentini, DSI - Univ. Milano
Main limitations of the Princeton group approach

Main drawbacks:

- **Hierarchical integration is local** (limited to the Markov blanket and the first 30 BFS nodes)

- **Integration strategy**: other works showed that methods other than VSI work better (e.g. Kernel fusion (*Lanckriet et al.*, 2004), ensemble methods (*Re and Valentini*, 2010)).

- The approach does not take into account the unbalance between positive and negative examples.
Conclusions

• Hierarchical ensembles improve results over simple “flat” methods
• The approach proposed by the Princeton group (Troyanskaya and collaborators) is very convincing, but there are also some drawbacks
• Several other nice approaches have been recently proposed (e.g. Obozinski et al, 2008, Schietgat et al. 2010)
• Considering the complexity of the gene function prediction problem, there is room for new research …