Enhancing the prediction of disease-gene associations with multimodal deep learning

1. Introduction

- protein-protein interaction (PPI) networks. using topological structure of PPI networks contain lots of false positives and is not accurate.
- PPI + clinical data: limited by amount and quality of data; tend to generate results that do not include less mutated gene.
- PPI + non-clinical data (generic models) + matrix factorization: too slow to converge, use limited types of data.
- Deep Belief Networks(DBN): number of known disease genes is too small to train a deep model. GBRBM is too har to train.

2. Key Takeaway

Two submodels are trained based on PPI networks and GO(gene ontology) terms respectively. Then a joint DBN is used to coombine the two submodels to learn cross modality representations.

3. Prerequisites

3.1 RBM: Restricted Boltzmann Machine

A fully connected bipartite graph unit with a visible layer and a hidden layer.

3.1.1 Binary-Binary RBM(BBRBM)

Inputs:
- Visible layer: v_i is a boolean variable representing the state of the visible unit i (the real data).
- Hidden layer: h_i is a boolean variable representing the state of the hidden unit i (the learned data to simulate visible features)
- Biases: a_i and b_j are the biases of visible unit i and hidden unit j respectively. It's a feature to be

learned
- Weight Matrix: W_{ij} represents the weight between v_i and h_j. It's a feature to be learned.

Energy Function for configuration vectors v and h:

$$E(v, h) = -(a^T v + b^T h + v^T W h)$$

Partition Function: Ths sum of $e^{-E(v,h)}$ over all possible configurations,
which is a normalization.

$$Z = \sum_v \sum_h e^{-E(v,h)}$$

Probability Distribution:
Joint Distribution $P(v, h) = \frac{1}{Z} \exp(-E(v, h))$

Marginal over v: $P(v) = \frac{1}{Z} \sum_h \exp(-E(v, h))$

Learning Objective:
Objective for trainining set $V = \{v_1, v_2, \cdots\}$:

$$\arg\max_W \prod_{v \in V} P(v) \Leftrightarrow \arg\max_W \sum_{v \in V} \log P(v) \Leftrightarrow \mathbb{E}_{v \in D}[\log P(v)]$$

for data distribution D.

We maximize this because it is a **log-likelihood** problem(given real data observations, the maximization objective is used to measure how likely the model can generate the true observations).

Training Algorithm (Contrast Divergence(CD)-1)
See `train_RBM.pdf` for details.

1. For a mini batch B of training samples V:

- Take training sample v, compute $P(h|v)$

$$P(h_j = 1|v) = \sigma(b_j + \sum_{i=1}^{n} v_i W_{ij})$$

- sample h from it (i.e. for a uniformly sampled r if $P(h_j = 1|v) > r$ then $h_j = 1$ else $h_j = 0$). Get the corresponding output $h(v)$

1. Calculate the "positive gradient" $\langle v_i h_j \rangle_{data}$ over the batch B:

$$\langle v_i h_j \rangle_{data} = \frac{1}{|B|} \sum_{v \in B} v_i h(v)_j$$

2.
   - For the constructed $h(v)$, compute $P(v|h)$:

$$P(v_i = 1|h) = \sigma(a_i + \sum_{j=1}^{m} h_j W_{ij})$$

   - sample reconstructed v' from it. Get the corresponding output $v'(h)$
   - calculate $P(h|v')$ to get reconstructed hidden h' from it. Get the corresponding output $h'(v')$.
3. Calculate the negative gradient $\langle v_i h_j \rangle_{recon}$ over the batch B:

$$\langle v_i h_j \rangle_{recon} = \frac{1}{|B|} \sum_{v' \in B} v'_i h(v')_j$$

4. Update Weights using stochastic gradient descent (This is called gradient descent because the following forms are the gradients of the log-likelihood function)
   - $\Delta W_{ij} = \eta \cdot (\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon})$
   - $\Delta a_i = \eta \cdot \frac{1}{|B|} \sum_{v \in B} (v_i - v'_i)$
   - $\Delta b_j = \eta \cdot \frac{1}{|B|} \sum_{v \in B} (h_j - h'_j)$

Optimizations:
- L2 regularization for weights:

$$\Delta W_{ij} = \eta \cdot (\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon}) - \lambda W_{ij}$$

- h' is not sampled but set to $P(h|v')$ directly to reduce noise.

3.1.2 Gaussian-Binary RBM(GBRBM)

Visible layer is continuous while hidden layer is binary.
Now the energy function becomes:

$$E(v, h) = \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} W_{ij} h_j$$

where σ_i is the standard deviation of visible unit i.

4. Methods

4.1 Multimodel DBN(Deep Belief Network)

DBNs can be seen as stacked RBMs where the hidden layer of one RBM is the visible layer of the next.

In the thesis, two DBN submodels are trained based on PPI network and GO terms respectively. Then we concatenate the outputs of the respective layers.

A joint DBN is used to combine the two submodels to learn cross modality representations, where the first layer is a GBRBM and the remains are BBRBMS.

4.2 Raw Feature Extraction

In each submodel,...

1. Construct a disease similarity network and a gene similarity network.
2. Features are extracted from these respective networks using `node2vec`. It performs random walk on a network and captures both local topological information and global structural equivalent properties to extract features. It's chosen because what it generates are **independent** features, which is important for RBM training.

4.2.1 Network Construction

4.2.1.1 PPI

- Gene similarity network: gene-gene inetraction network projected from PPI network. 也就是说，把PPI中的蛋白质节点换成编码它们的基因
- Disease similarity network: using the **disease module theory**. We define the distance between two disease modules in the PPI network:

  $$\mathrm{SIM}^{d}_{\mathrm{ppi}}(d_1, d_2)$$

  and use **KNN** to construct the disease similarity network.

Disease Module: A disease module of a disease d is a subgraph $M = (V, E)$, where V is a set of genes associated with d, and E is the set of interactions among genes in V.

Similarities:

- For two diseases d_1 and d_2, the similarities of them is:

$$\mathrm{SIM}_{\mathrm{ppi}}^{d}(d_1, d_2) = \frac{2 \cdot s(M_1, M_2)}{s(M_1, M_1) + s(M_2, M_2)}$$

- where the similarity of their disease modules M_1, M_2 is:

$$s(M_1, M_2) = \frac{\left(\sum_{g \in V(M_1)} F_{M_2}(g)\right)\left(\sum_{g \in V(M_2)} F_{M_1}(g)\right)}{|V(M_1)| + |V(M_2)|}$$

- Where for gene g and disease module M,

$$F_M(g) = \mathrm{avg}\left(\sum_{g_i \in V(M)} \mathrm{sim}(g, g_i)\right)$$

  simulates the simularity between gene g and genes in module M.
- where:

$$\mathrm{sim}(g_1, g_2) = \begin{cases} 1 & g_1 = g_2 \\ \exp\left(-\mathrm{sp}(g_1, g_2)\right) & g_1 \neq g_2 \end{cases}$$

Where $\mathrm{sp}(g_1, g_2)$ is the shortest path between gene g_1 and g_2 in the PPI network.

4.2.1.2 GO(Gene Ontology)

Ontologies provided: biological process, cellular component, molecular function.

GO raw database: For a gene, there are a set of **GO terms** that are related to it. The database is a DAG where nodes are terms, e.g. "cell division" and edges are semantic relationships between terms, e.g. "is_a", "part_of".

Successor graph of a term: $\mathrm{DAG}_A = (T_A, E_A)$ where T_A is the set of all **ancestor** terms of term A (including A itself) and E_A is the set of edges among terms in T_A, where we define S**-values**:

$$S_A(t) = \begin{cases} 1 & t = A \\ \max\{w_e(t, t') \cdot S_A(t') \mid t' \in \text{children of } t\} & t \neq A \end{cases}$$

By this recursion we can derive S_A from A to the root.

> Note we need to distinguish between **children** and **successors** here. Children are the direct descendants while successors include all descendants.

where w_e is the weight of edge (manually defined for each semantic)

Similarities:

- For two diseases d_1 and d_2 associated with gene sets G_{d_1} and G_{d_2} respectively, their similarities is defined as:

$$\mathrm{SIM}^d_{\mathrm{GO}}(G_{d_1}, G_{d_2}) = \frac{\sum_{g \in G_{d_1}} \mathrm{SG}(g, G_{d_2}) + \sum_{g \in G_{d_2}} \mathrm{SG}(g, G_{d_1})}{|G_{d_1}| + |G_{d_2}|}$$

- where

$$\mathrm{SG}(g, G) = \max_{g' \in G} \mathrm{SIM}_{\mathrm{GO}}(g, g')$$

- where the functional similarities of two genes g_1, g_2, represented by their GO term sets T_{g_1} and T_{g_2} respectively, is defined as:

$$\mathrm{SIM}_{\mathrm{GO}}(g_1, g_2) = \frac{\sum_{t \in T_{g_1}} \mathrm{sim}_{\mathrm{GO}}(t, T_{g_2}) + \sum_{t \in T_{g_2}} \mathrm{sim}_{\mathrm{GO}}(t, T_{g_1})}{|T_{g_1}| + |T_{g_2}|}$$

- where the semantic similarity between a GO term t and a set of GO terms T is:

$$\mathrm{sim}_{\mathrm{GO}}(t, T) = \max_{t' \in T} \mathrm{SGO}(t, t')$$

- where, for two GO terms A and B, their similarity is defined as:

$$\mathrm{SGO}(A, B) = \frac{\sum_{t \in T_A \cap T_B} (S_A(t) + S_B(t))}{\sum_{t \in T_A} S_A(t) + \sum_{t \in T_B} S_B(t)}$$

4.2.2 input Construction

We use `node2vec` to extract features from the gene similarity network and disease similarity network respectively, and concatenate them as the input for one submodel. Now Ψ is the input feature vector set representing relations of disease i and gene j:

$$\psi_{ij} = (\phi^d_i, \phi^g_j)$$

where ϕ^d_i is the feature vector of disease d_i and ϕ^g_j is the feature vector of gene g_j.

4.3 Evaluation

- Use AUC Curve: Area under ROC Curve

- **ROC**: Receiver Operating Characteristic Curve. It is a function where :
  - x-axis: False Positive Rate(FPR) = $\frac{FP}{FP+TN}$
  - y-axis: True Positive Rate(TPR) = $\frac{TP}{TP+FN}$
- T/F; True/False; P/N: Positive/Negative
- Higher AUC means better performance.

4.4 Dataset

4.4.1 Source

OMIM database, three labels:

- (3): known
- ?: provisional
- []: abnormal library test values

4.4.2 Preproceesing

- Only consider (3) as positive samples.
- Merging similar disease entries into one using string match and manual verification.
- Delete disease associated with only one gene that is not connected to other diseases in the PPI network.

4.4.3 Constructing Negative samples

Note that the database only contains positive samples (known disease-gene associations). We need to construct negative samples (unknown disease-gene associations) for training.

- ψ^p_{avg} : average feature vector of all positive samples.

For each unlabeled sample u:

- d^p_u : Euclidean distance between u and ψ^p_{avg}. Average it to get d^p_{avg}.
- If $d^p_u > d^p_{\text{avg}}$, sample u is considered as a reliable negative sample.

4.5 HyperParameters

- LR(BBRBM): 0.01
- LR(GBRBM): 0.0005
- Dim for each submodel: 256
- Batch Size: 4
- Epochs: 30 (convergence threshold)

5. Performance

- Better than other proposed algorithms trained on similar datasets
- Predicts unlearned disease-gene associations that are verified in literature.