Gene expression inference with Deep Learning

POLITECNICO DI MILANO

Andrea Sessa

Yifei Chen, Yi Li, Rajiv Narayan,
Aravind Subramanian, Xiaohui Xie
Bioinformatics, 2016
Volume 32, Issue 12, pages: 1832-1839

Motivations

- A fundamental problem in molecular biology: Characterize gene expression pattern under certain biological states
- **Gene expression profiling** has been historically used as a tool to capture the gene expression patterns in response to **diseases**
- Unfortunately, whole-genome gene expression profiling is still **too expensive** to be used in a typical academic lab to generate a compendium over thousand of samples
- Human genome: ~ 22000 genes!

Problem Statement - 1

In the following we will indicate:

- L the number of landmark genes
- T the number of target genes
- N the number of training samples

$x^i \in R^L$ denotes the expression value of the landmark genes in the i-th sample

$y^i \in R^T$ denotes the expression value of the target genes in the i-th sample Our goal is to infer the function:

$$F : R^L \rightarrow R^T \tag{1}$$

Problem Statement - 2

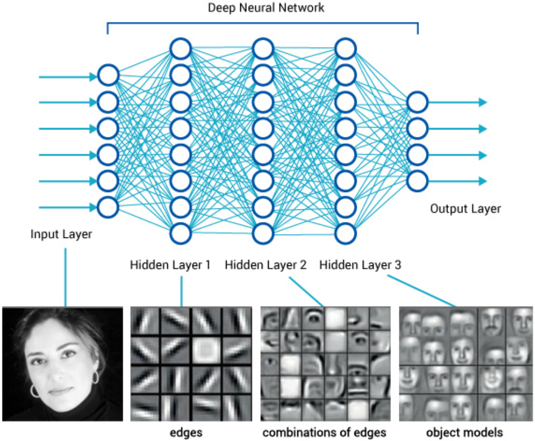
- Computationally complex to infer the expression profiles of target genes based on landmark genes
- A large-scale multi-task supervised learning problem!
- We need also to consider that the target dimension(22000) is significantly **greater** thant the feature dimension(1000)
  - LINCS uses linear regression, high scalability but it ignores **non-linearity patterns** inside the data
  - Others(*Ye et al. 2006*) have tried to use kernel machines(ie. SVM), scalability problem!
  - Seems natural to adopt **Deep Learning** approach, high scalability + high representability

Previous Works

- **Main idea**: The expression profile of genes are known to be **highly** correlated!
- Connectivity **MAP** project: Creation of a large reference collection gene expression patterns of the human genome (only 564 genome-wide gene expr. profiles)
- **LINCS**: PCA analysis over the CMAP data; only ~ 1000 genes of 22000 capture the 80% of the variance!
  - The chosen 1000 genes are called **landmark genes**
  - The remaining part are called **target genes**
  - Measure gene expression of the landmark genes under certain biological condition(low cost!)
  - Infer the expression of the target genes from the landmark genes and other expression profile
  - LINCS program currently use **linear regression** to infer the expression of the target genes

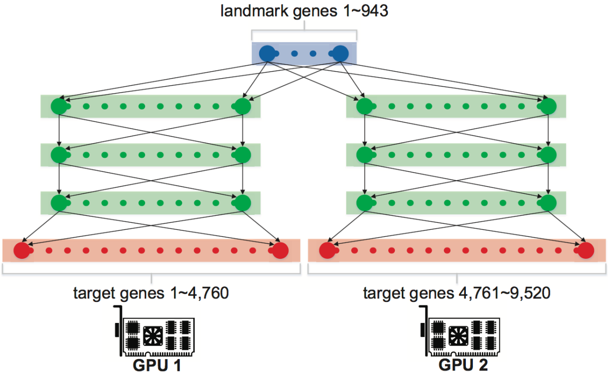
Deep Learning Architecture

Idea: learn a **hierarchical** rappresentation of the data through multiple layers



D-GEX - 1

D-Gex: A mult-task multi-layed feedforward neural network



953 hidden units(one for each landmark gene)

9520 output units, divided in two due to capicity constraints

D-GEX - 2

The ouput of the hidden unit j in the hidden layer l is given by:

$$o_j^l = f(\sum_{i=1}^{H} \omega_{i,j}^{l-1} o_i^{l-1} + b_j^{l-1})$$

$f(x)$ is a non-linear activation function that in our case has been chosen as the hyperbolic tangent

We try the learn w_{ij}, b_j by minimizing the sum of mean squared errors for each output unit(t)

$$E = \sum_{t=1}^{T} \left[\frac{1}{N} \sum_{i=1}^{N} (y_{i(t)} - \hat{y}_{i(t)})^2 \right]$$

D-GEX - Training

The training follows the standard back-propagation algorithm, with some tricks:

- **Dropout**: Regularization and model averaging
- **Momentum gradient descent**: Use velocity to update parameters
- **Normalized Initialization:** Initial parameters sampled from a uniform distribution
- **Learning Rate**: $5 * 10^{-4}$ or $3 * 10^{-4}$ and decreased according to the training error
- **Model Selection**: 200 epochs and evaluated against the dataset after each epoch

Linear Regression

Idea: Train a **linear** model **for each** target gene t:

$$F_t(x) = w_t^T x + b_t$$

where $w_t^T x$ and b_t are the model parameters associated to t:

$$(w_t, b_t) = \underset{w,b}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} (y_{it} - w_t^T x_i - b_t)^2 \tag{2}$$

For regularization we introduce the L1 and L2 norms:

$$(w_t, b_t) = \underset{w,b}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} (y_{it} - w_t^T x_i - b_t)^2 + \lambda ||w_t||_1 \tag{3}$$

$$(w_t, b_t) = \underset{w,b}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} (y_{it} - w_t^T x_i - b_t)^2 + \lambda ||w_t||_2 \tag{4}$$

L(1) is currently used in LINCS

λ is tuned on the GEO-va and 1000G-va datasets.

KNN-GE

- Non-parametric, instance based algorithm
- For any testing data, the *k-nearest* neighbors based on a certain distance metric(euclidean) are used for the prediction(average)
- **Problem:** Bias due to duplicated samples in the data(DGE)
- **Solution:** Do not query the k nearest sample but the k nearest genes
- **Pros:** No prior assumption on the model, high capability to model non linear pattern
- k is tuned on the GEO-va and 1000G-va datasets.

Datasets - 1

The datasets used are:

- **GEO Expression Data**
  - 129158 gene expression profiles from **Affymetrix microarray platform**
  - Normalization: quantile normalization(joint) + duplicate removal
- **GTEx Expression Data**
  - 2921 gene expression profiles from **Illumina RNA-Seq platform**
  - Normalization: quantile normalization(joint)
- **1000 Genome expression data**
  - 462 gene expression profiles from **Illumina RNA-Seq platform**
  - Normalization: quantile normalization(joint)

Datasets - 2

For the **microarray** platform(Affymetrix):

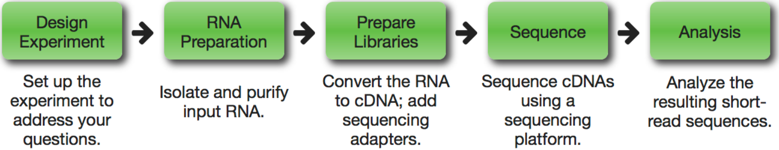
- **GEO** dataset
- 80 % for training(GEO-tr), 10 % validation(GEO-va), 10 % testing(GEO-te)

For the **RNA-Seq** platform:

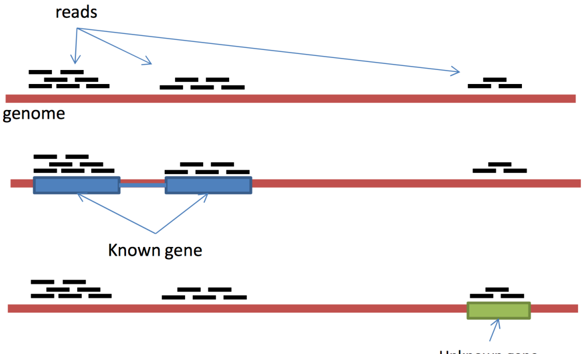
- The **GEO-tr** dataset was used for training
- The **1000G**-va dataset was used for validation
- The **GTEx**-te was used for testing

Something on RNA-Seq - 1

- Developed in **2008**
- Affymetrix Oligonucleotides microarray $\sim$ 2000
- cDNA microarray $\sim$ 1995


Design Experiment
RNA Preparation
Prepare Libraries
Sequence
Analysis
Set up the experiment to address your questions.
Isolate and purify input RNA.
Convert the RNA to cDNA; add sequencing adapters.
Sequence cDNAs using a sequencing platform.
Analyze the resulting short-read sequences.

Something on RNA-Seq - 2

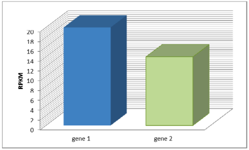


reads

genome

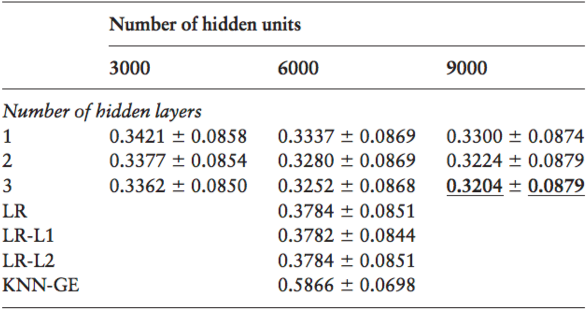
Known gene

Unknown gene



Performance on GEO - 1

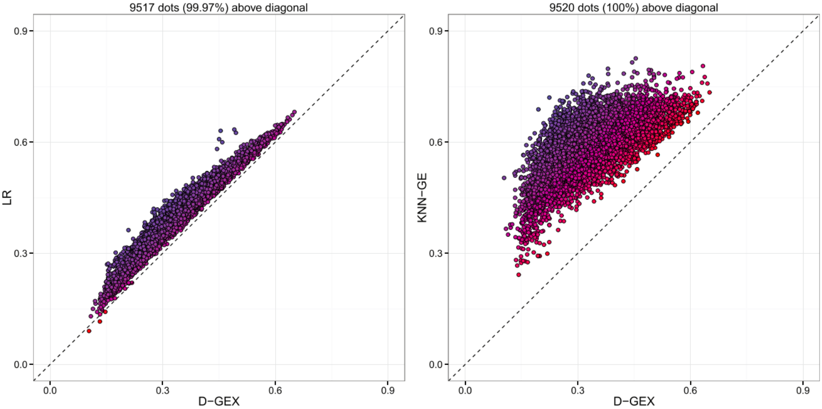
The best performance of DGEx is achieved with a dropout rate of 10 %



	Number of hidden units		
	3000	6000	9000
Number of hidden layers			
1	0.3421 ± 0.0858	0.3337 ± 0.0869	0.3300 ± 0.0874
2	0.3377 ± 0.0854	0.3280 ± 0.0869	0.3224 ± 0.0879
3	0.3362 ± 0.0850	0.3252 ± 0.0868	0.3204 ± 0.0879
LR		0.3784 ± 0.0851	
LR-L1		0.3782 ± 0.0844	
LR-L2		0.3784 ± 0.0851	
KNN-GE		0.5866 ± 0.0698	

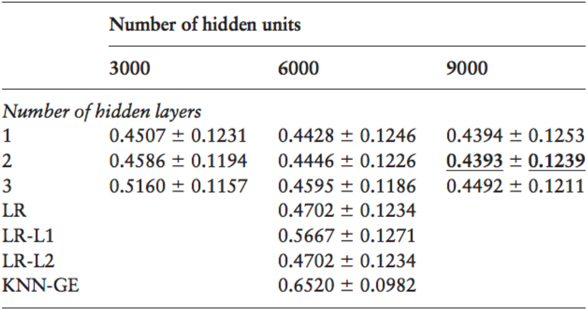
Relative improvement: 15.33% vs LR and 45.38% vs KNN

Performance on GEO - 2



Performance on GTEx

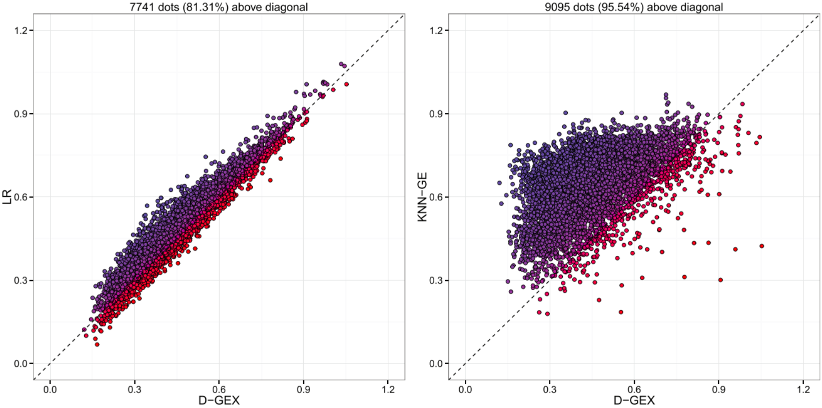
Cross-platform scenario: reduced predictive power!



	Number of hidden units		
	3000	6000	9000
Number of hidden layers			
1	0.4507 ± 0.1231	0.4428 ± 0.1246	0.4394 ± 0.1253
2	0.4586 ± 0.1194	0.4446 ± 0.1226	0.4393 ± 0.1239
3	0.5160 ± 0.1157	0.4595 ± 0.1186	0.4492 ± 0.1211
LR		0.4702 ± 0.1234	
LR-L1		0.5667 ± 0.1271	
LR-L2		0.4702 ± 0.1234	
KNN-GE		0.6520 ± 0.0982	

Relative improvement: 6.57% vs LR and 32.63% vs KNN

Performance on GTEx - 2



Interpretation

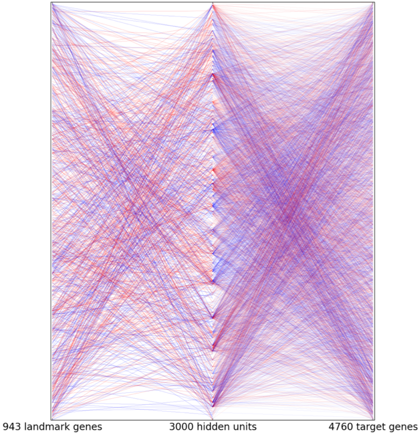
Interpreting the linear model produced by **LR** is **easy**: big coefficients indicate a strong dependency between the target and the landmark genes.

On the other hand interpreting the model learned by DGEx is much more complex!
No consolidated method exist, the author proposed:

- Major weight visualization
- Analysis of the non-linearity captured by the hidden units

Major weights visualization

Following the idea of LR, try to visualize the more **important** connection between artificial neurons.



Non-Linearity Analysis

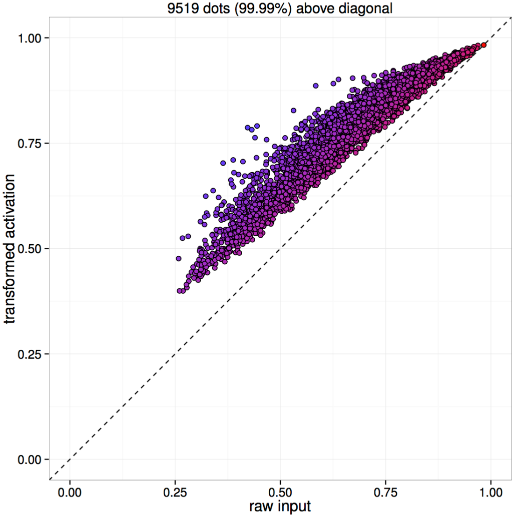
Try to understand if some of the hidden units have captured some non-linearity pattern.
Problem: many neurons in DGEx!

Idea: each hidden unit represent a feature, that is a non-linear transformation of the expression of the landmark genes.
Can a LR model based on this feature achieve better performances than a simple LR?
The author have estimated the (adjusted)R^2 between:

- The last hidden layer and the target (features based LR)
- The raw input and the target (Simple LR)

In the first case we achieve a larger R^2!

Non-Linearity Analysis - Results



The Software

- Available at: https://github.com/uci-cbcl/D-GEX
- **Not** user friendly, it a collection of script that model the neural network.
- The software permits just to train the network and to evaluate the performance.
- Poor documentation: comments in the code???.
- Usage of the learned model is up to the final user!

Conclusions

- DGEx outperforms LR in a **single-platform** scenario
- DGEx outperforms LR in a **cross-platform** scenario(dropout)
- Interpretation of the learned model lead to the conclusion that the network has capture non-linearity pattern in the data(LR probably underfit the data)
- Model interpretation in LR is easy while it is very complicated in deep learning settings

Possible **improvements**:

- Target genes have been randomly partitioned and each set was trained separately using different GPUs
- First improvement: perform clustering and the train the model
- Target genes sharing similar expression profiles share weights in the NN
- Other possibilities: use a larger memory GPU(or multi-GPU techniques)

Thank you!
Questions?