GENIE3: GENE REGULATORY NETWORK INFERENCE — AN INTUITIVE UNDERSTANDING

Songting Shi
songtingstone@gmail.com

December 9, 2020

Uncovering the secret of the biologic process is the dream of each biologist, and the genes regulatory network helps to reveal this big figure in the key position. As the rapidly development of the sequencing technologies, we now accumulate a huge mount of the gene expression data, which give us a valuable opportunity to putative the possible regulatory network by data mining and then verifying them by experiments or some other way.

Currently, SCENIC(Aibar et al. (2017), Van de Sande et al. (2020)) is one of the best tools to achieve this goal. The core element is based on the GINIE3 model. So we decide to begin with it firstly. In this tutorial, we will explain the basic principle of the GINIE3 to do the gene regulatory network inference.

GENIE3(Huynh-Thu et al. (2010)) is a tree based model to solve the gene regulatory network. It use the regression tree random forest to build the relation between the target gene and the regulators. With the importance score comes from its contribution to the variance reduction of the MSE error. And the importance score final becomes the the ranking score.

To understand the random forests(Breiman (2001)) model based on the regression trees, we should down to the simple, understand the regression tree model. We can easy to do this since it something like the reality tree (Fig 1(Bleier (2018))), but it is even more simple, it is a binary tree and can be represent in the panel like this (Fig 2), and now remember this figure in your head, we now will put the details how this binary tree is built and tell you why it is helpful to capture the gene regulatory network.

Suppose that we have the sequencing data

$$LS := \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \tag{1}$$

, where

$$\mathbf{x}_k := (x_k^1, x_k^2, \ldots, x_k^p)^T \tag{2}$$

is the measured expression values k-th sample(cells) with all p genes.

Suppose now we want to inference the regulatory genes of gene j, we first split the samples into target gene j and all other genes as the candidate regulatory genes. Denoting the expression value of all genes except gene j in the k-th sample(cell) as

$$\mathbf{x}_k^{-j} := (x_k^1, \ldots, x_k^{j-1}, x_k^{j+1} \ldots, x_k^p)^T \tag{3}$$

Then the learning sample can be denoted by

$$LS^j := \{(x_k^j, \mathbf{x}_k^{-j}),\ i = 1, \ldots, N\} \tag{4}$$

We want to learn a function $f_j(\cdot)$ such that it minimizes the mean square error

$$\frac{1}{N}\sum_{i=1}^{N}(x_k^j - f_j(\mathbf{x}_k^{-j}))^2 \tag{5}$$

In the process of learning, we hope that we can learn a rank score $w_{ij}, i \neq j$ which measures the regulatory influence of regulator gene i on target gene j. Moreover, we hope that $w_{ij}, i \neq j$ is comparable for all pairs $(i, j),\ i \neq j$, so that we can use it to rank all pairs of genes to find the most confident regulatory relations.

Form the probability theory, we know that the optimal solution of the problem

$$\mathbb{E}_{(x^j, \mathbf{x}^{-j}) \sim p(x^j, \mathbf{x}^{-j})}(x^j - f_j(\mathbf{x}^{-j}))^2 \tag{6}$$

Fig 1

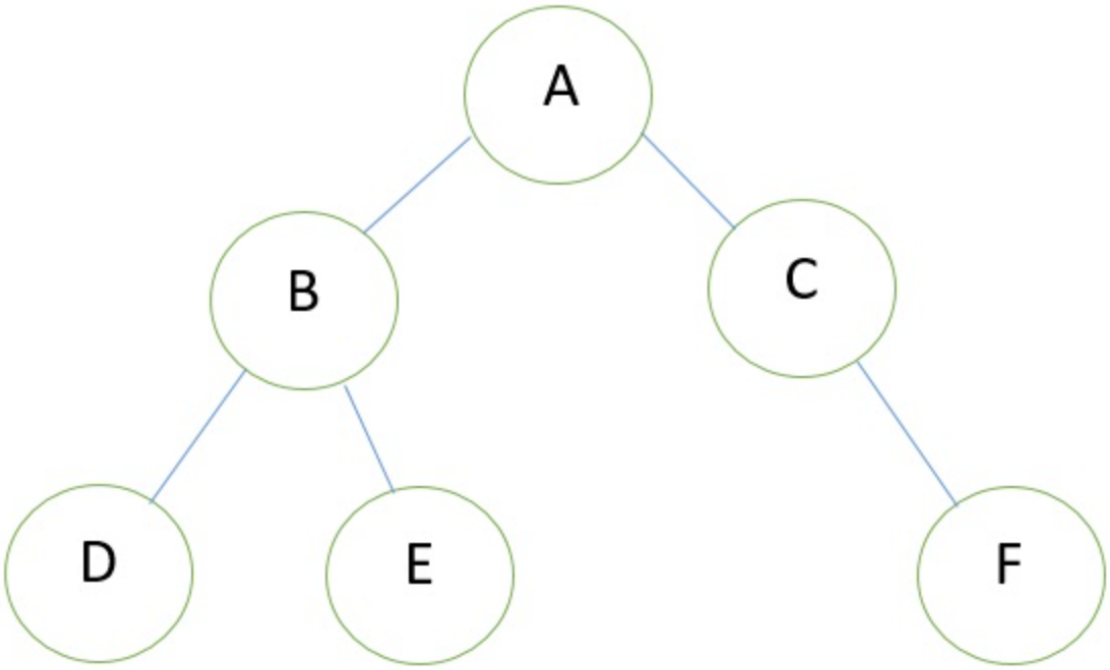

Fig 2

is given by

$$f_j(\mathbf{x}^{-j}) := \mathbb{E}_{(x^j,\mathbf{x}^{-j})\sim p(x^j,\mathbf{x}^{-j})}(x^j|\mathbf{x}^{-j}) \tag{7}$$

, where $\mathbb{E}(\cdot)$ stands for expectation and $p(x^j, \mathbf{x}^{-j})$ is the probability distribution of genes $(x^j, \mathbf{x}^{-j})$.

When given the learning sample LS^j, the sample estimation of loss (6) can be estimate by (5), so the optimal solution of the loss (5) can be estimated by $\mathbb{E}_{(x^j,\mathbf{x}^{-j})\sim\tilde{p}(x^j,\mathbf{x}^{-j})}(x^j|\mathbf{x}^{-j})$, where $\tilde{p}(x^j, \mathbf{x}^{-j})$ is the empirical probability distribution of genes $(x^j, \mathbf{x}^{-j})$ given by the samples LS^j.

The regression tree model basically learn the condition expectation $\mathbb{E}_{(x^j,\mathbf{x}^{-j})\sim\tilde{p}(x^j,\mathbf{x}^{-j})}(x^j|\mathbf{x}^{-j})$ by use the rectangle-like set to split the Euclidean space of $\mathbf{x}^{-j}$ based on the distribution of target gene x^j.

To make the concepts become specific, we consider a simple example. Supposing that there are four genes $1, 2, 3, 4$, the true regulatory genes for target gene 1 are genes $2, 3, 4$. and we have the following relation on the state state of transcription process.

$$x^1 = x^2 + x^3 + x^4 \tag{8}$$

We obtained samples in 9 experiments, with the expression values given by

$$X := \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 2 & 0 \\ 2 & 0 & 2 & 0 \\ 2 & 0 & 2 & 0 \\ 6 & 0 & 0 & 6 \\ 6 & 0 & 0 & 6 \\ 6 & 0 & 0 & 6 \end{bmatrix} \tag{9}$$

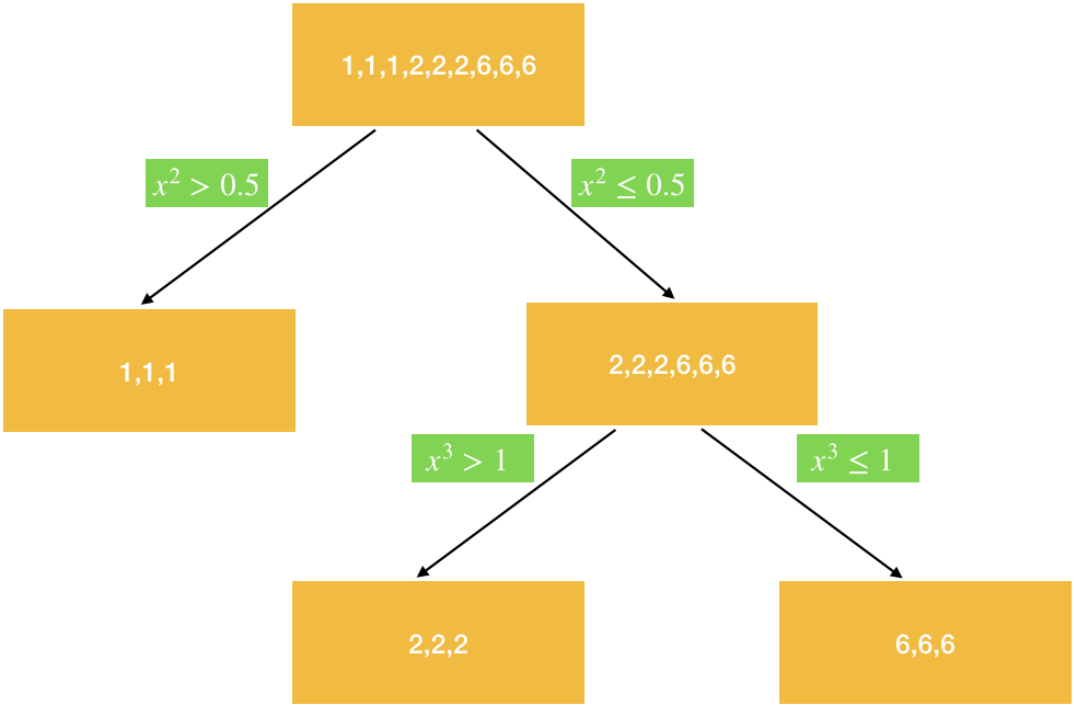

Fig 3

, where the columns of X stands for genes, and rows for samples. The probability distribution of genes (x^1, x^2, x^3, x^4) is given by

$$p(x^1, \mathbf{x}^{-1}) = p(x^1, x^2, x^3, x^4) := \begin{cases} 1/3 & \text{if } (x^1, x^2, x^3, x^4) \in \{(1,1,0,0),\ (2,0,2,0),\ (6,0,0,6)\} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Note that, the empirical distribution based on the specific samples X are the same as the true probability distribution, i.e.

$$\tilde{p}(x^1, \mathbf{x}^{-1}) = p(x^1, \mathbf{x}^{-1}) \tag{11}$$

And the optimal solution has the following form

$$\begin{aligned} f_1(\mathbf{x}^{-1}) &= \mathbb{E}_{(x^1,\mathbf{x}^{-1})\sim p(x^1,\mathbf{x}^{-1})}(x^1|\mathbf{x}^{-1}) = \mathbb{E}_{(x^1,\mathbf{x}^{-1})\sim \tilde{p}(x^1,\mathbf{x}^{-1})}(x^1|\mathbf{x}^{-1}) \\ &= x^2 + x^3 + x^4 \\ &= I_{x^2=1} + 2I_{x^3=2} + 6I_{x^4=6} \\ &= I_{x^2>0.5} + 2I_{x^2\leq 0.5,\ x^3>1} + 6I_{x^2\leq 0.5,\ x^3<1} \\ &= I_{x^2>0.5} + 2I_{x^2\leq 0.5,\ x^3>1} + 6I_{x^2\leq 0.5,\ x^3<1} \\ &= 6I_{x^4>3} + 2I_{x^4\leq 3,\ x^2\leq 0.5} + I_{x^4\leq 3,\ x^2>0.5} \\ &= 6I_{x^4>3} + 2I_{x^4\leq 3,\ x^3>1} + I_{x^4\leq 3,\ x^3\leq 1} \end{aligned} \tag{12}$$

, where I is the indictor function. The above function can be represented by the regression trees Fig 3, Fig 4, Fig 5 or Fig 6.

In these figures, the yellow box (node) contains the sample values of the target gene 1, the green box contains the test condition. The root node contains all the sample values of gene 1, and the intermediate and leaf node contains the samples values of gene 1 whose corresponding expression values of gene $2, 3, 4$ satisfy the conditions in the green boxes which reside on the path from the root node to the current node. The last four equalities in equation (12) correspond to the for figures Fig 3, Fig 4, Fig 5 or Fig 6, respectively.

In this simple example, we see that the optimal solution can be exactly captured by the regression tree, which means that the rectangle-like representation of the conditional expectation can be represented by the binary regression

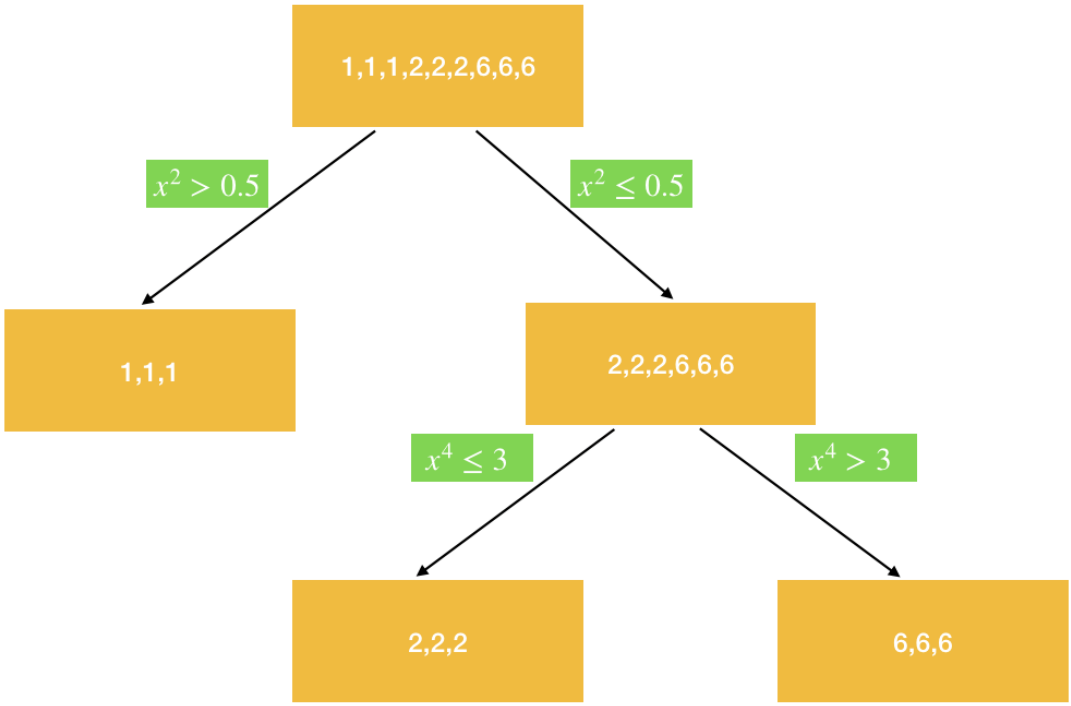

Fig 4

trees with sequentially choosing the split variable with its split point. In generally, this is true, since every event in the probability space be represented by the union of the basic rectangle-like sets.

Now, we have the basic intuition that the optimal solution which given by the conditional expectation can be approximated by the regression tree. Next, we want to see how to build the regression tree efficiently. We want to find way to build the regression tree with less times of split variable, and to reduce the mean square error as fast as possible.

The key to the split is to chose which leaf node to split which will reduce the mean square error in the maximum amount. GENIE3 use the total reduction of the variance of the output target variable as the indictor to select the split node and split point, which is defined by

$$\mathbf{I}(\mathcal{N}) = \#S\mathrm{Var}(S) - \#S_t\mathrm{Var}(S_t) - \#S_f\mathrm{Var}(S_f) \tag{13}$$

, where S is the target gene values of samples that reach node $\mathcal{N}$, and S_t (resp. S_f) is the subset for which the test is true (resp. false). More specifically, supposing that at the node $\mathcal{N}$, we chose the one split variable gene $i \neq j$ and the split value at s, denoting

$$\mathbf{I}(\mathcal{N}, i, s) := \sum_{k\in S}(x_k^j - \frac{1}{\#S}\sum_{k\in S}x_k^j)^2 - \sum_{k\in S_t}(x_k^j - \frac{1}{\#S_t}\sum_{k\in S_t}x_k^j)^2 - \sum_{k\in S_f}(x_k^j - \frac{1}{\#S_f}\sum_{k\in S_f}x_k^j)^2 \tag{14}$$

where $S_t := \{k|k \in S, x_k^i > s\}$ and $S_f := \{k|k \in S, x_k^i \leq s\}$. Then

$$\mathbf{I}(\mathcal{N}) = \max_{\{i|i\neq j\},s} \mathbf{I}(\mathcal{N}, i, s) \tag{15}$$

The regression tree can be built as in the Algorithm 1.

Using the Algorithm 1 on our simple example, we can get the regression tree which is represented in Fig 5 or Fig 6, which can be verified easily. On the root node, we selected the split variable gene 4 and split value at $3 = (0 + 6)/2$, and we have the reduction on the total reduction of the variance of the target gene 1 is

$$\mathbf{I}(\text{root node}) = \max_{\{i|i\neq j\},s} \mathbf{I}(\mathcal{N}, i, s) = 40.5 \tag{16}$$

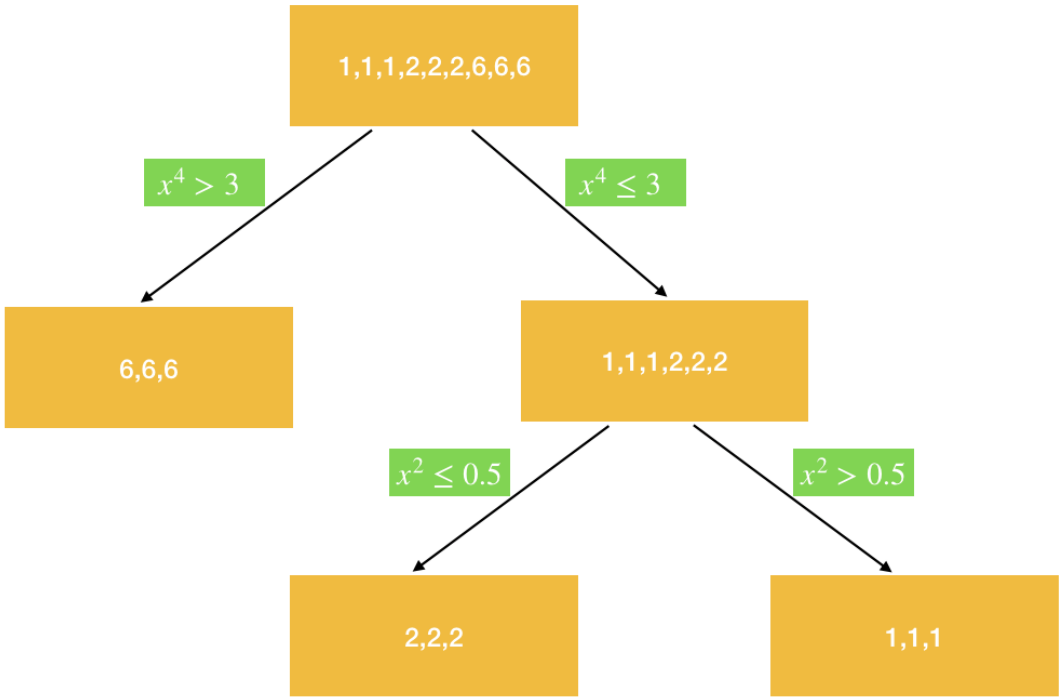

Fig 5

Algorithm 1 Regression Tree

1: **function** REGRESSIONTREE(LS^j)
2: **repeat**
3: For each leaf node of the current tree, compute $\mathbf{I}(\mathcal{N})$ if this node has not been computed before.
4: Find the the largest $I(\mathcal{N})$ among the leaf nodes, and denoting the optimal split variable as i and optimal split value as s and the largest value $\mathbf{I}(\mathcal{N})$ as $\mathbf{I}$. Using the optimal node as the split node and the corresponding test condition $x^i > s$ as the test function, growing up the tree by splitting the optimal node to two leaf nodes with $x^i > s$ as the test condition.
5: **until** $\mathbf{I}$ less than the pre-specified threshold

To verify it indeed achieves the maximum reduction, we can write a small python program (Fig 7) to check it. After running the program, we can get the results, Fig 8. From the result, we see that the split $[1, 1, 1, 2, 2, 2, 6, 6, 6]$ into $[6, 6, 6]$ and $[1, 1, 1, 2, 2, 2]$ indeed give the maximum reduction.

After we slit the root node with the values of gene 1, $S = \{1, 1, 1, 2, 2, 2, 6, 6, 6\}$, we get the split nodes $S_t = \{6, 6, 6\}$ and $S_f = \{1, 1, 1, 2, 2, 2\}$. The variance of S_t is 0, so there is no need to split, we will predict the expression of gene 1 with value 6 if it satisfies that $x^4 > 3$. And $\#S_f \text{Var}(S_f) = 1.5$, we can split with the gene 2(resp. gene 3), with test condition $x^2 \leq 0.5$ (resp. $x^3 > 1$), and finally, we get the regression tree Fig 5 (resp. Fig 6).

For each regression tree, we can measure the contribution of each regulator gene to the target gene, this is achieved summing the $\mathbf{I}$ values of all tree nodes where this variable is used to split, and we denote the contribution of the regulator gene i to the target gene j by $w_{i,j}$. For example, for the regression tree Fig 5 (resp. Fig 6), we will get $w_{2,1} = 1.5$, $w_{3,1} = 0$, $w_{4,1} = 40.5$ (resp. $w_{2,1} = 0$, $w_{3,1} = 1.5$, $w_{4,1} = 40.5$). Note that $\#S\text{Var}(S) = 42 = w_{2,1} + w_{3,1} + w_{4,1}$. In general, if the samples values of the target gene which reach to leaf

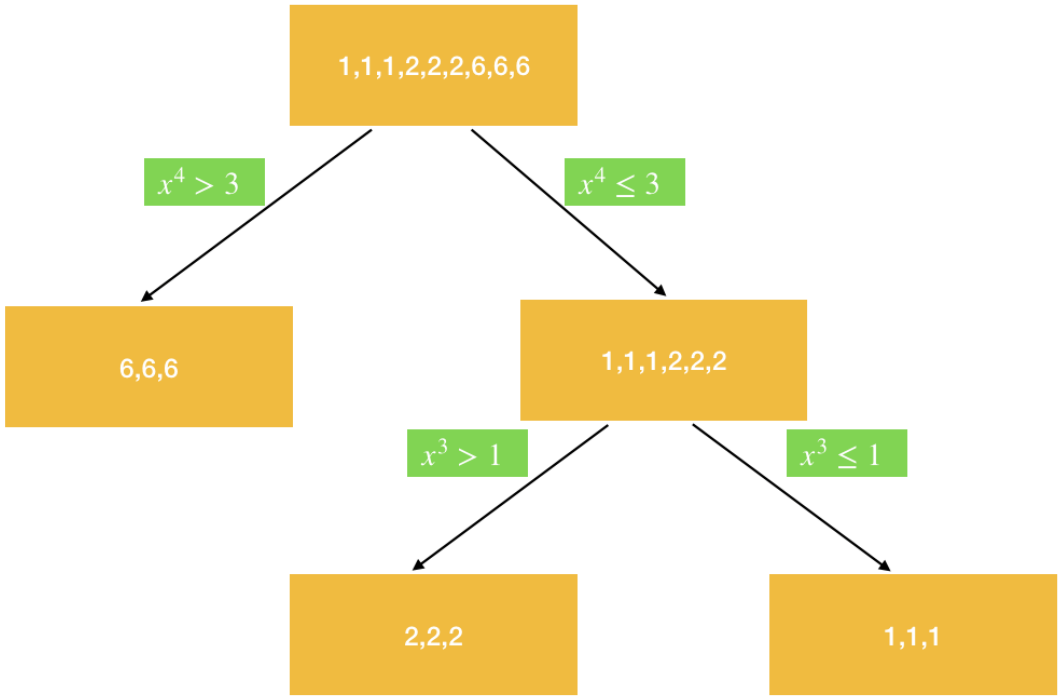

Fig 6

node of the regression tree have the zero variance, then

$$\#S\mathrm{Var}(S) = \sum_{i|i\neq j} w_{i,j} \tag{17}$$

The zero variance of a variable means that it take a constant value. In general case, the samples at the leaf node may have small variance, then we have

$$\#S\mathrm{Var}(S) \approx \sum_{i|i\neq j} w_{i,j} \tag{18}$$

The regression tree will predict the value of the target gene with means of samples reaching the leaf node, if all the the regulatory genes satisfy the test conditions along the root node to the leaf node, which is just the intuitive definition of the condition expectation.

As we can see form the above example, the regression tree has some drawbacks, it sometime will miss the importance of true regulators, e.g. , in the Fig 5, it miss the importance of the gene 4 which regulates the target gene 1 (resp. in the Fig 6, it miss the importance of the gene 4 which regulates the target gene 1).

Random forest will overcomes this drawbacks by building a collection of regression trees which are independent identically distributed, combining them into a random forest to do the prediction and also output the importance score by averaging the importance scores of each tree. Each regression tree is building on the bootstrap samples of LS^j and on the tree growing process, each split will randomly select a subset of K variables to do split.

As we can see our example, if we ensemble the regression tree of Fig 5 and Fig 6, then we can get a better result, the importance score will becomes $w_{2,1} = 0.75,\ w_{3,1} = 0.75,\ w_{4,1} = 40.5$.

Now we give the algorithm of the random forest in Algorithm 2.

In the random forest, there values of K can be chosen as $\sqrt{p-1}$ or $p-1$ by the traditional experience. For our simple example, if we choose $k = 2$ and the number of trees $m_0 = 100$, then the ensemble the regression

```
import numpy as np

def mean(x):
    if(len(x)==1):
        return x
    return np.sum(x)/np.prod(np.shape(x))

def var_sum(x):
    m = mean(x)
    return np.sum(np.square(x-m))

def I(x, x_t, x_f):
    x= np.asarray(x)
    x_t = np.asarray(x_t)
    x_f = np.asarray(x_f)
    return var_sum(x) - var_sum(x_t) - var_sum(x_f)
x = np.array([1,1,1,2,2,2,6,6,6])

def ci(i):
    return 3-i
def x_t_g(i1,i2,i6):
    return [1,] *i1 + [2,] *i2 + [6,]*i6
def x_f_g(i1,i2,i6):
    return [1,] *ci(i1) + [2,] *ci(i2) + [6,]*ci(i6)
def print_res(i1,i2,i6):
    x_t = x_t_g(i1,i2,i6)
    x_f = x_f_g(i1,i2,i6)
    if(len(x_t)>0 and len(x_f)>0 and len(x_t)+ len(x_f) ==9):
        print( I(x, x_t, x_f),x_t,x_f)
for i1 in range(0,4):
    for i2 in range(0,4-i1):
        for i6 in range(0, 4-i1 -i2):
            print_res(i1,i2,i6)
        i6=0
        print_res(i1,i2,i6)
    i2,i6=0,0
    print_res(i1,i2,i6)
```

Fig 7

```
10.125 [6] [1, 1, 1, 2, 2, 2, 6, 6]
23.142857142857142 [6, 6] [1, 1, 1, 2, 2, 2, 6]
40.5 [6, 6, 6] [1, 1, 1, 2, 2, 2]
1.125 [2] [1, 1, 1, 2, 2, 6, 6, 6]
2.5714285714285765 [2, 6] [1, 1, 1, 2, 2, 6, 6]
12.5 [2, 6, 6] [1, 1, 1, 2, 2, 6]
1.125 [2] [1, 1, 1, 2, 2, 6, 6, 6]
2.5714285714285694 [2, 2] [1, 1, 1, 2, 6, 6, 6]
0.49999999999999645 [2, 2, 6] [1, 1, 1, 2, 6, 6]
2.5714285714285694 [2, 2] [1, 1, 1, 2, 6, 6, 6]
4.5 [2, 2, 2] [1, 1, 1, 6, 6, 6]
4.5 [2, 2, 2] [1, 1, 1, 6, 6, 6]
4.5 [1] [1, 1, 2, 2, 2, 6, 6, 6]
0.6428571428571459 [1, 6] [1, 1, 2, 2, 2, 6, 6]
8.000000000000004 [1, 6, 6] [1, 1, 2, 2, 2, 6]
4.5 [1] [1, 1, 2, 2, 2, 6, 6, 6]
5.785714285714285 [1, 2] [1, 1, 2, 2, 6, 6, 6]
0.0 [1, 2, 6] [1, 1, 2, 2, 6, 6]
5.785714285714285 [1, 2] [1, 1, 2, 2, 6, 6, 6]
8.0 [1, 2, 2] [1, 1, 2, 6, 6, 6]
8.0 [1, 2, 2] [1, 1, 2, 6, 6, 6]
4.5 [1] [1, 1, 2, 2, 2, 6, 6, 6]
10.285714285714288 [1, 1] [1, 2, 2, 2, 6, 6, 6]
0.5 [1, 1, 6] [1, 2, 2, 2, 6, 6]
10.285714285714288 [1, 1] [1, 2, 2, 2, 6, 6, 6]
12.500000000000007 [1, 1, 2] [1, 2, 2, 6, 6, 6]
12.500000000000007 [1, 1, 2] [1, 2, 2, 6, 6, 6]
10.285714285714288 [1, 1] [1, 2, 2, 2, 6, 6, 6]
18.0 [1, 1, 1] [2, 2, 2, 6, 6, 6]
18.0 [1, 1, 1] [2, 2, 2, 6, 6, 6]
18.0 [1, 1, 1] [2, 2, 2, 6, 6, 6]
```

Fig 8

Algorithm 2 Random Forest

function REGRESSIONTREE(LS^j, K, m_0, n)
  Initialize the number of tress $m = m_0$.
  for $l \leftarrow 1, \ldots, m$ **do**
    Generating a bootstrap sample from LS^j with n samples, and denoting it by the LS^j_l
    {Building the regression tree l with learning sample LS^j_l}
    repeat
      For each leaf node of the current tree, compute $\mathbf{I}(\mathcal{N})$ on a random select K variables for genes $\{1, \ldots, j-1, j+1, p\}$ if this node has not been computed before.
      Find the the largest $I(\mathcal{N})$ among the leaf nodes, and denoting the optimal split variable as i and optimal split value as s and the largest value $\mathbf{I}(\mathcal{N})$ as $\mathbf{I}$. Using the optimal node as the split node and the corresponding test condition $x^i > s$ as the test function, growing up the tree by splitting the optimal node to two leaf nodes with $x^i > s$ as the test condition.
    until $\mathbf{I}$ less than the pre-specified threshold
    Compute the importance scores of K variables and storing it in $w^l_{i,j}$
  Compute the overall importance score $w_{i,j} = \frac{1}{m}\sum_{l=1}^{m} w^l_{i,j}$. And the predictors as the average of the m regression tree predictors.

trees will composite with Fig 5 and Fig 6 in equal numbers approximately, the importance score will approach to $w_{2,1} = 0.75,\ w_{3,1} = 0.75,\ w_{4,1} = 40.5$. And this improve the result from the a single regression tree.

Now we can give the GENIE3 algorithm 3

Algorithm 3 Gene Networks Inference with Ensemble of Trees

function GENIE3(LS, K, m_0, n)
  Initialize the number of tress $m = m_0$. Normalize the the gene expressions samples LS such that each gene has a unit variance, denoting it still as LS for simplicity.
  for $j \leftarrow 1, \ldots, p$ **do**
    Generating the learning sample of inout-output pairs for gene j:

$$LS^j := \{(x^j_k, \mathbf{x}^{-j}_k),\ i = 1, \ldots, N\}$$

    Running RegressionTree(LS^j, K, m_0, n) to compute the confidence levels $w_{i,j}, \forall i \neq j$, for all genes except gene j itself.
  Aggregate the p individual gene rankings in get a global ranking of all regulatory links.

In the algorithm 3, we normalize the the gene expressions samples LS such that each gene has a unit variance to make that the $w_{i,j}$ can be comparable for all gene j. This can be seen from equation 18, if we un-normalize the expression value, then $w_{i,j}$ will has a relative higher value for the target gene j which has a larger variance than those have a lower variance. To reduce this positive bias to the highly variable genes, we normalize each gene has a unit variance.

The graphical abstract of GENIE3 (Huynh-Thu et al. (2010)) is given in Fig 9

Finally, I hope you read the original paper GENIE3 (Huynh-Thu et al. (2010)) to get a deeper understanding.

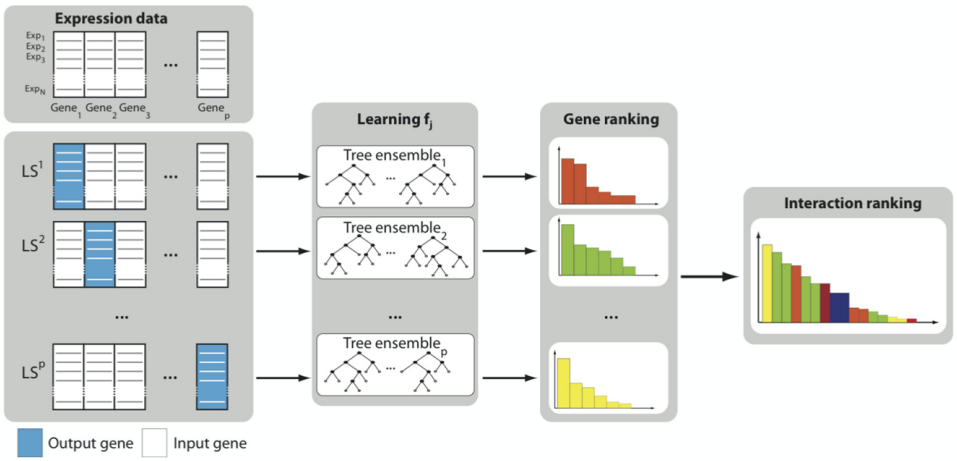

Fig 9

References

Aibar, S., González-Blas, C. B., Moerman, T., Huynh-Thu, V. A., Imrichova, H., Hulselmans, G., Rambow, F., Marine, J.-C., Geurts, P., Aerts, J., van den Oord, J., Atak, Z. K., Wouters, J., and Aerts, S. (2017). Scenic: single-cell regulatory network inference and clustering. *Nature Methods*, 14(11):1083–1086.

Bleier, A. (2018). Coding regression trees in 150 lines of r code https://www.r-bloggers.com/2018/11/coding-regression-trees-in-150-lines-of-r-code/.

Breiman, L. (2001). Random forests. In *Machine Learning*, pages 5–32.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., and Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PLOS ONE*, 5(9):1–10.

Van de Sande, B., Flerin, C., Davie, K., De Waegeneer, M., Hulselmans, G., Aibar, S., Seurinck, R., Saelens, W., Cannoodt, R., Rouchon, Q., Verbeiren, T., De Maeyer, D., Reumers, J., Saeys, Y., and Aerts, S. (2020). A scalable scenic workflow for single-cell gene regulatory network analysis. *Nature Protocols*, 15(7):2247–2276.