TGPred User Guide

TGPred v.0.1.0 Python Version

TGPred: Efficient methods for predicting target genes of a transcription factor by integrating statistics, machine learning, and optimization.

Below is documentation of each function in the APGD module. If you have any question, feel free to contact Ling (lingzhan@mtu.edu).

Functions:

1. ConstructNetwork(n_genes, structure)

Construct the network structure from either Hierarchical Network or Barabasi-Albert Network in simulation studies.

- Input:
  - n_genes: the number of genes.
  - structure: "HN": Hierarchical Network or "BAN": Barabasi-Albert Network.
- Output:
  - adj_all: n_genes * n_genes dimensional symmetric adjacency matrix of network structure.

Example:

```
N = 200
Adj = ConstructNetwork(n_genes=N, structure="BAN")
Adj = ConstructNetwork(n_genes=N, structure="HN")
```

2. GraphicalModel(adj, a1=-0.7, a2=-0.1, b1=0.1, b2=0.7)

Simulate a covariance matrix from the specific graph (Adj) based on a Gaussian graphical model.

- Input:
  - adj: the adjacency matrix of network structure.
  - a1, a2, b1, b2: parameters for constructing domain [a1, a2] union [b1, b2].
    - default: a1 = -0.7, a2 = -0.1, b1 = 0.1, b2 = 0.7
- Output:
  - sigma: covariance matrix of target genes according to network structure.

Example:

```
N = 200
Adj = ConstructNetwork(n_genes=N, structure="BAN")
Sigma1 = GraphicalModel(Adj)
```

3. SimulationData(n_samples, n_genes, adj, sigma, method, beta0=None, beta_true=None)

Simulate y and X from a given network structure.

- Input:

  - n_samples: the number of sample size.
  - n_genes: the number of target genes.
  - adj: the adjacency matrix of network structure. Adjacency matrix must be a n_genes * n_genes dimensional.
    * symmetric matrix, the elements equal 1 indicates two genes are connected. If you consider Barabasi-Albert.
    * Network or Hierarchical Network in the article, you can directly use "ConstructNetwork" function to get the adjacency matrix.the adjacency matrix of network structure. directly use "ConstructNetwork" function to get the adjacency matrix.
  - sigma: the covariance matrix of target genes according to network structure. You can directly use "GraphicalModel" function to get the covariance matrix.
  - method: "HN": by Hierarchical Network, "BAN": by Barabasi-Albert Network or "DIY": by user designed.
  - beta0: numeric value of effect size in simulation settings.
    * default: None; if method is "HN" or "BAN", input a numerical value.
  - beta_true: numeric matrix with the dimension of n_genes * 1 in simulation settings.
    * default: None; if method is "DIY", input a numerical matrix (n_genes * 1).
- Output:
  - y: expression levels of a transcription factor (TF).
  - X: expression levels of n_genes target genes (TGs).
  - beta: true regulated effect beta for n_genes TGs.

Example:

```
N_samples = 300
N_genes = 200
Adj = ConstructNetwork(n_genes=N_genes, structure="BAN")
Sigma1 = GraphicalModel(Adj)
# Set up a true regression coefficient for simulated data (beta0=1)
res = SimulationData(N_samples, N_genes, Adj, Sigma1, "BAN", beta0=1)
y = res[0]
X = res[1]
beta1 = res[2]
```

4. CalculateAdj(annotated_matrix)

Calculate adjacency matrix from an annotation file.

- Input:
  - annotated_matrix: n_genes * n_pathways dimensional matrix that indicates the annotation of genes within pathways information.
- Output:
  - adj: the adjacency matrix of network structure.

Example:

```
Annoted_df = read_file(matrix_path)
Adj = CalculateAdj(Annoted_df)
```

5. CalculateLaplacian(adj)

Calculate Laplacian matrix and symmetric normalized Laplacian matrix from an adjacency matrix.

- Input:
  - adj: the adjacency matrix of network structure.
- Output:
  - l: the Laplacian matrix for network structure.

- l_norm: the symmetric normalized Laplacian matrix for network structure.

Example:

```
Annoted_df = read_file(matrix_path)
Adj = CalculateAdj(Annoted_df)
```

6. Lambda_grid(X, y, n_lambda, alpha, loss_func, ratio=1e-2)

Simulate a grid set of lambdas for a given alpha in penalized regression.

- Input:
  - X: expression levels of n_genes target genes (TGs).
  - y: expression levels of a transcription factor (TF).
  - n_lambda: the number of lambdas. Positive integers.
  - alpha: the proportion of l1 norm affects (the numerical values of nonzero coefficients), it's in range (0,1].
  - loss_func: either "Huber" or "MSE".
  - If flag = "Huber", the loss function in penalized regression model is Huber function.
  - If flag = "MSE", the loss function in penalized regression model is mean squared errors.
  - ratio: the ratio of the smallest lambda.
    - default: 0.01
- Output:
  - lambdas: n_lambda length vector of lambdas according to the alpha you provided.

Example:

```
alpha = 0.5
n_lambda = 10
lambda_set = Lambda_grid(X, y, n_lambda, alpha, loss_func = "Huber")
lambda_set = Lambda_grid(X, y, n_lambda, alpha, loss_func = "MSE")
```

7. HuberNet_Beta(X, y, adj, lambda0, alpha0, method="APGD", gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, quiet=False, scales=False)

Estimate beta_hat using HuberNet function.

- Input:
  - X: expression levels of n_genes target genes (TGs).
  - y: expression levels of a transcription factor (TF).
  - adj: the adjacency matrix of network structure.
  - lambda0: one of parameters in HuberNet regression, which controls the number of nonzero coefficients.
  - alpha0: one of parameters in HuberNet regression, which controls the numerical values of nonzero coefficients.
  - method: The current methods must be 'APGD' or 'CVX'.
  - gamma: initial value of gamma in APGD.
    - default: 1000
  - niter: the maximum number of APGD to solve HuberNet regression.
    - default: 2000
  - crit_beta: converge criterion of change of beta.
    - default: 1e-4
  - crit_obj: converge criterion of change of objective function.
    - default: 1e-8
  - quiet: decide if exist the output report.
    - default: False
  - scales: decide if scale the expression levels.

        * default: False
- Output:
  - beta_hat: n_genes length vector of estimated regulated effect sizes, where beta_j != 0 indicates j th gene is not selected in HuberNet regression.

Example:

```
lambda0 = 200
alpha0 = 0.5
beta_hat_APGD = HuberNet_Beta(X, y, Adj, lambda0, alpha0, method="APGD", scales=True)
beta_hat_CVX = HuberNet_Beta(X, y, Adj, lambda0, alpha0, method="CVX", scales=True)
```

8. HuberNet_SP(X, y, adj, alphas, n_lambda, ratio=1e-2, B=500, gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, timer=True)

Estimate selection probability using HuberNet function solving by APGD.

- Input:
  - X: expression levels of n_genes target genes (TGs).
  - y: expression levels of a transcription factor (TF).
  - adj: the adjacency matrix of network structure.
  - alphas: the grid sets of alpha (in [0,1]) used to calculate selection probabilities of genes.
  - n_lambda: the number of lambdas.
  - ratio: the ratio of the smallest lambda.
    * default: 0.01
    * B: the number of half-sample resampling used to calculate selection probabilities of genes.
      · default: 500
    * gamma: initial value of gamma in APGD.
      · default: 1000
    * niter: the maximum number of APGD to solve HuberNet regression.
      · default: 2000
    * crit_beta: converge criterion of change of beta.
      · default: 1e-4
    * crit_obj: converge criterion of change of objective function.
      · default: 1e-8
    * timer: decide if exist the output report.
      · default: True
- Output:
  - sp_hubernet: n_genes length vector of selection probability.

Example:

```
alphas = [0.1, 0.9]
n_lambda = 10
B0 = 100
ratio = 0.01
SP_HuberNet = HuberNet_SP(X, y, Adj ,alphas, n_lambda, ratio, B=B0, gamma=1000, niter=2000, timer=False)
```

9. HuberLasso_Beta(X, y, lambda0, method="APGD", gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, quiet=False, scales=False)

Estimate beta_hat using Huber Lasso function.

- Input:
  - X: expression levels of n_genes target genes (TGs).
  - y: expression levels of a transcription factor (TF).
  - lambda0: one of parameters in Huber Lasso regression, which controls the number of nonzero coefficients.

  - method: the current methods must be 'APGD' or 'CVX'.
  - gamma: initial value of gamma in APGD.
    * default: 1000
  - niter: the maximum number of APGD to solve Huber Lasso regression.
    * default: 2000
  - crit_beta: converge criterion of change of beta.
    * default: 1e-4
  - crit_obj: converge criterion of change of objective function.
    * default: 1e-8
  - quiet: decide if exist the output report.
    * default: False
  - scales: decide if scale the expression levels.
    * default: False
- Output:
  - beta_hat: n_genes length vector of estimated regulated effect sizes, where beta_j != 0 indicates j th gene is not selected in HuberLasso regression.

Example:

```
lambda0 = 200
beta_hat_APGD = HuberLasso_Beta(X, y, lambda0, method="APGD", scales=True)
beta_hat_CVX = HuberLasso_Beta(X, y, lambda0, method="CVX", scales=True)
```

10. HuberLasso_SP(X, y, n_lambda, ratio=1e-2, B=500, gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, timer=True)

Estimate selection probability using HuberLasso function solving by APGD.

- Input:
  - X: expression levels of n_genes target genes (TGs).
  - y: expression levels of a transcription factor (TF).
  - alphas: the grid sets of alpha (in [0,1]) used to calculate selection probabilities of genes.
  - n_lambda: the number of lambdas.
  - ratio: the ratio of the smallest lambda.
    * default: 0.01
    * B: the number of half-sample resampling used to calculate selection probabilities of genes.
      · default: 500
    * gamma: initial value of gamma in APGD.
      · default: 1000
    * niter: the maximum number of APGD to solve Huber Lasso regression.
      · default: 2000
    * crit_beta: converge criterion of change of beta.
      · default: 1e-4
    * crit_obj: converge criterion of change of objective function.
      · default: 1e-8
    * timer: decide if exist the output report.
      · default: True
- Output:
  - sp_huberlasso: n_genes length vector of selection probability.

Example:

```
n_lambda = 50
B0 = 100
ratio = 0.01
SP_HuberLasso = HuberLasso_SP(X, y, n_lambda, ratio, B=B0, gamma=1000, niter=2000, timer=False)
```

11. HuberENET_Beta(X, y, lambda0, alpha0, method="APGD", gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, quiet=False, scales=FALSE)

Estimate beta_hat using Huber Elastic Net function

- Input:
  - X: expression levels of n_genes target genes (TGs).
  - y: expression levels of a transcription factor (TF).
  - lambda0: one of parameters in Huber Elastic Net regression, which controls the number of nonzero coefficients.
  - alpha0: one of parameters in Huber Elastic Net regression, which controls the numerical values of nonzero coefficients.
  - method: the current methods must be 'APGD' or 'CVX'.
  - gamma: initial value of gamma in APGD.
    - default: 1000
  - niter: the maximum number of APGD to solve Huber Elastic Net regression.
    - default: 2000
  - crit_beta: converge criterion of change of beta.
    - default: 1e-4
  - crit_obj: converge criterion of change of objective function.
    - default: 1e-8
  - quiet: decide if exist the output report.
    - default: False
  - scales: decide if scale the expression levels.
    - default: False
- Output:
  - beta_hat: n_genes length vector of estimated regulated effect sizes, where beta_j != 0 indicates j th gene is not selected in Huber Elastic Net regression.

Example:

```
lambda0 = 200
alpha0 = 0.5
beta_hat_APGD = HuberENET_Beta(X, y, lambda0, alpha0, method="APGD", scales=True)
beta_hat_CVX = HuberENET_Beta(X, y, lambda0, alpha0, method="CVX", scales=True)
```

12. HuberENET_SP(X, y, alphas, n_lambda, ratio=1e-2, B=500, gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, timer=True)

Estimate selection probability using HuberENET function solving by APGD.

- Input:
  - X: expression levels of n_genes target genes (TGs).
  - y: expression levels of a transcription factor (TF).
  - alphas: the grid sets of alpha (in [0,1]) used to calculate selection probabilities of genes.
  - n_lambda: the number of lambdas.
  - ratio: the ratio of the smallest lambda.
    - default: 0.01
    - B: the number of half-sample resampling used to calculate selection probabilities of genes.
      - default: 500
    - gamma: initial value of gamma in APGD.
      - default: 1000
    - niter: the maximum number of APGD to solve Huber Elastic Net regression.
      - default: 2000
    - crit_beta: converge criterion of change of beta.
      - default: 1e-4
    - crit_obj: converge criterion of change of objective function.

        - default: 1e-8
      * timer: decide if exist the output report.
        - default: True
- Output:
  - sp_huberenet: n_genes length vector of selection probability.

Example:

```
alphas = [0.1, 0.9]
n_lambda = 10
B0 = 100
ratio = 0.01
SP_HuberENET = HuberENET_SP(X, y, alphas, n_lambda, ratio, B=B0, gamma=1000, niter=2000, timer=False)
```

13. MSEENET_Beta(X, y, lambda0, alpha0, method="APGD", gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, quiet=False, scales=False)

Estimate beta_hat using MSE loss along with Elastic Net penalty function.

- Input:
  - X: expression levels of n_genes target genes (TGs).
  - y: expression levels of a transcription factor (TF).
  - lambda0: one of parameters in Elastic Net regression, which controls the number of nonzero coefficients.
  - alpha0: one of parameters in Elastic Net regression, which controls the numerical values of nonzero coefficients.
  - method: the current methods must be 'APGD' or 'CVX'.
  - gamma: initial value of gamma in APGD.
    * default: 1000
  - niter: the maximum number of APGD to solve Elastic Net regression.
    * default: 2000
  - crit_beta: converge criterion of change of beta.
    * default: 1e-4
  - crit_obj: converge criterion of change of objective function.
    * default: 1e-8
  - quiet: decide if exist the output report.
    * default: False
  - scales: decide if scale the expression levels.
    * default: False
- Output:
  - beta_hat: n_genes length vector of estimated regulated effect sizes, where beta_j != 0 indicates j th gene is not selected in Elastic Net regression.

Example:

```
lambda0 = 200
alpha0 = 0.5
beta_hat_APGD = MSEENET_Beta(X, y, lambda0, alpha0, method="APGD", scales=True)
beta_hat_CVX = MSEENET_Beta(X, y, lambda0, alpha0, method="CVX", scales=True)
```

14. MSEENET_SP(X, y, alphas, n_lambda, ratio=1e-2, B=500, gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, timer=True)

Estimate selection probability using MSEENET function solving by APGD.

- Input:
  - X: expression levels of n_genes target genes (TGs).

  - y: expression levels of a transcription factor (TF).
  - alphas: the grid sets of alpha (in [0,1]) used to calculate selection probabilities of genes.
  - n_lambda: the number of lambdas.
  - ratio: the ratio of the smallest lambda.
    * default: 0.01
    * B: the number of half-sample resampling used to calculate selection probabilities of genes.
      · default: 500
    * gamma: initial value of gamma in APGD.
      · default: 1000
    * niter: the maximum number of APGD to solve Elastic Net regression.
      · default: 2000
    * crit_beta: converge criterion of change of beta.
      · default: 1e-4
    * crit_obj: converge criterion of change of objective function.
      · default: 1e-8
    * timer: decide if exist the output report.
      · default: True
- Output:
  - sp_enet: n_genes length vector of selection probability.

Example:

```
alphas = [0.1, 0.9]
n_lambda = 10
B0 = 100
ratio = 0.01
SP_ENET = MSEENET_SP(X, y, alphas, n_lambda, ratio, B=B0, gamma=1000, niter=2000, timer=False)
```

15. MSELasso_Beta(X, y, lambda0, method="APGD", gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, quiet=False, scales=False)

Estimate beta_hat using MSE loss along with Lasso penalty function.

- Input:
  - X: expression levels of n_genes target genes (TGs).
  - y: expression levels of a transcription factor (TF).
  - lambda0: one of parameters in HuberNet regression, which controls the number of nonzero coefficients.
  - gamma: initial value of gamma in APGD.
    * default: 1000
  - niter: the maximum number of APGD to solve Lasso regression.
    * default: 2000
  - crit_beta: converge criterion of change of beta.
    * default: 1e-4
  - crit_obj: converge criterion of change of objective function.
    * default: 1e-8
  - quiet: decide if exist the output report.
    * default: False

  - scales: decide if scale the expression levels.
    * default: False
- Output:
  - beta_hat: n_genes length vector of estimated regulated effect sizes, where beta_j != 0 indicates j th gene is not selected in Lasso regression.

Example:

```
lambda0 = 200
beta_hat_APGD = MSELasso_Beta(X, y, lambda0, method="APGD", scales=True)
beta_hat_CVX = MSELasso_Beta(X, y, lambda0, method="CVX", scales=True)
```

16. MSELasso_SP(X, y, n_lambda, ratio=1e-2, B=500, gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, timer=True)

Estimate selection probability using MSELasso function solving by APGD.

- Input:
  - X: expression levels of n_genes target genes (TGs).
  - y: expression levels of a transcription factor (TF).
  - n_lambda: the number of lambdas.
  - ratio: the ratio of the smallest lambda.
    - default: 0.01
    - B: the number of half-sample resampling used to calculate selection probabilities of genes.
      - default: 500
    - gamma: initial value of gamma in APGD.
      - default: 1000
    - niter: the maximum number of APGD to solve HuberNet regression.
      - default: 2000
    - crit_beta: converge criterion of change of beta.
      - default: 1e-4
    - crit_obj: converge criterion of change of objective function.
      - default: 1e-8
    - timer: decide if exist the output report.
      - default: True
- Output:
  - sp_lasso: n_genes length vector of selection probability.

Example:

```
n_lambda = 50
B0 = 100
ratio = 0.01
SP_Lasso = MSELasso_SP(X, y, n_lambda, ratio, B=B0, gamma=1000, niter=2000, timer=False)
```

17. MSENet_Beta(X, y, adj, lambda0, alpha0, method="APGD", gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, quiet=False, scales=False)

Estimate beta_hat using MSE loss along with Net penalty function.

- Input:
  - X: expression levels of n_genes target genes (TGs).
  - y: expression levels of a transcription factor (TF).
  - adj: the adjacency matrix of network structure.
  - lambda0: one of parameters in Net regression, which controls the number of nonzero coefficients.
  - alpha0: one of parameters in Net regression, which controls the numerical values of nonzero coefficients.
  - method: the current methods must be 'APGD' or 'CVX'.
  - gamma: initial value of gamma in APGD.
    - default: 1000
  - niter: the maximum number of APGD to solve HuberNet regression.
    - default: 2000
  - crit_beta: converge criterion of change of beta.
    - default: 1e-4
  - crit_obj: converge criterion of change of objective function.

        * default: 1e-8
    - quiet: decide if exist the output report.
        * default: False

    - scales: decide if scale the expression levels.
        * default: False
- Output:
    - beta_hat: n_genes length vector of estimated regulated effect sizes, where beta_j != 0 indicates j th gene is not selected in Net regression.

Example:

```
lambda0 = 200
alpha0 = 0.5
beta_hat_APGD = MSENet_Beta(X, y, Adj, lambda0, alpha0, method="APGD", scales=True)
beta_hat_CVX = MSENet_Beta(X, y, Adj, lambda0, alpha0, method="CVX", scales=True)
```

18. MSENet_SP(X, y, adj, alphas, n_lambda, ratio=1e-2, B=500, gamma=1000, niter=2000, crit_beta=1e-4, crit_obj=1e-8, timer=True)

Estimate selection probability using MSENet function solving by APGD.

- Input:
    - X: expression levels of n_genes target genes (TGs).
    - y: expression levels of a transcription factor (TF).
    - adj: the adjacency matrix of network structure.
    - alphas: the grid sets of alpha (in [0,1]) used to calculate selection probabilities of genes.
    - n_lambda: the number of lambdas.
    - ratio: the ratio of the smallest lambda.
        * default: 0.01
        * B: the number of half-sample resampling used to calculate selection probabilities of genes.
            · default: 500
        * gamma: initial value of gamma in APGD.
            · default: 1000
        * niter: the maximum number of APGD to solve Net regression.
            · default: 2000
        * crit_beta: converge criterion of change of beta.
            · default: 1e-4
        * crit_obj: converge criterion of change of objective function.
            · default: 1e-8
        * timer: decide if exist the output report.
            · default: True
- Output:
    - sp_net: n_genes length vector of selection probability.

Example:

```
alphas = [0.1, 0.9]
n_lambda = 10
B0 = 100
ratio = 0.01
SP_Net = MSENet_SP(X, y, Adj ,alphas, n_lambda, ratio, B=B0, gamma=1000, niter=2000, timer=False)
```

Support Functions:

1. read_file(file_path)

Read file path to get data.

- Input:
  - file_path: the path of the file (.txt .csv) and separator by tab (''̂).
- Output:
  - df: data frame.

2. Huber_gradient(delta, m)

Calculate the gradient of Huber function for an input value delta.

- Input:
  - delta: Input value delta.
  - m: Shape parameter, which is defaulted to be one-tenth of the interquartile range (IRQ).
- Output:
  - value: The gradient of Huber function for an input value delta.

3. Huber_Mz(z, m)

Calculate the Huber function for an input value z.

- Input:
  - z: Input value z
  - m: Shape parameter, which is defaulted to be one-tenth of the interquartile range (IRQ).
- Output:
  - value: The Huber function for an input value z.

4. count_time(start, end)

Calculate running time.

- Input:
  - start: start point.
  - end: end point.
- Output:
  - show the running time by hours, minutes and seconds.