Toolbox "DEEPGLM"

May 2018

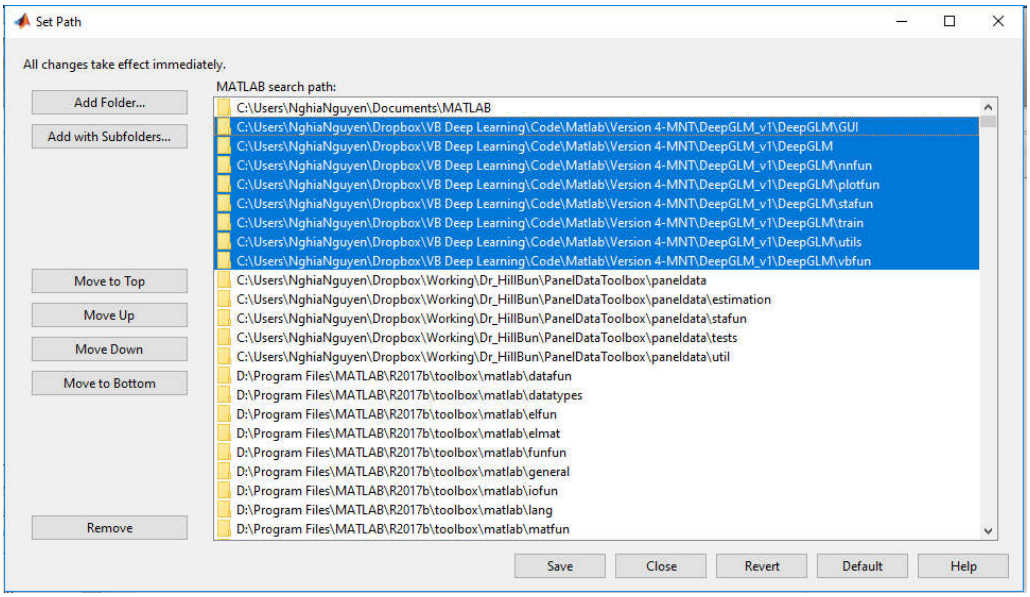
Type	Toolbox
Title	Bayesian Deep Net Generalized Linear Model (**deepGLM**)
Version	1.0
Released	May 2018
Description	Matlab Toolbox for paper **Bayesian Deep Net GLM and GLMM with Natural Gradient Factor-Gaussian Variational Approximation** (DeepGLM, 2018) by *Minh-Ngoc Tran*, *Nghia Nguyen*, *David J. Nott* and *Robert Kohn*. DeepGLM is a flexible version of Generalized Liner Model where Deep Feedforward Network is used to automatically choose transformations for the raw covariates.
Language	Matlab (2016a or later)
Author	Nghia Nguyen (nghia.nguyen@sydney.edu.au) Minh-Ngoc Tran (minh-ngoc.tran@sydney.edu.au)

Contents

Download and Install

Download and use deepGLM Matlab package

- **Step 1**: Clone or download the Matlab package from GitHub link: https://github.com/VBayesLab/deepGLM
- **Step 2**: Add the deepGLM package to Matlab path as following:
  - In Matlab window, choose **Home → Set path**
  - In Set path dialog, add the folder (Add with Subfolders) named deepGLM inside deepGLM package downloaded directory



  - **Save** and **Close**

Preparing Data

Preparing data for deepGLM

To fitting a deepGLM model, prepare your data in the form that is accepted by fitting function **deepGLMfit**. The data for training and testing must be divided into two parts, a design matrix *X* for predictor variables and a column *y* for response variable.

- Predictor variables *X*, specified as an *n*-by-*p* matrix, where *n* is the number of observations and *p* is the number of predictor variables. Each column of *X* represents one variable, and each row represents one observation.
- Response variable *y*, specified as an *n*-by-1 vector, where *n* is the number of observations. Each entry in *y* is the response for the corresponding row of *X*.

deepGLMfit

Fit a deepGLM model

Syntax

```
mdl = deepGLMfit (X, y)
```

mdl = deepGLMfit (__ , Name, Value)

Description

mdl = deepGLMfit (X, y) fits a deepGLM model using the design matrix *X* and response vector *y*, and returns an output structure **mdl** to make prediction on a test data. By default, if *'distribution'* option is not specified, deepGLMfit will treat response variable *y* as normal distributed variable.

mdl = deepGLMfit (__ , Name, Value) fit a deepGLM model with additional options specified by one or more of the following name/value pairs:

'Distribution' Name of the distribution of the response, chosen from the following:

'normal'	Normal distribution (for continuous response)
'binomial'	Binomial distribution (for binary response)
'poisson'	Poisson distribution (for counting response)

Example 'Distribution', 'binomial'
Default 'normal'
Data type single | string

'Network' Neuron Network structure for deepGLM. In the current version, deepGLM supports only 1 node for the output layer, users just need to provide a structure for hidden layers in an array where each element in the array is the number of nodes in the corresponding hidden layer.

Example 'Network', [10,10,10] → deepGLM uses 3 hidden layers with 10 nodes for each.
Default [10,10]
Data type array | positive integer

'Lrate' The fix learning rate that is used for training. If the learning rate is too small, training will take a long time, but if it is too high, the training is likely to get stuck at a suboptimal result.

Example 'Lrate', 0.001
Default 0.01
Data type single | double

'Momentum' Momentum weight for stochastic gradient ascend. The momentum determines the contribution of the gradient step from the previous iteration to the current iteration of training. It must be a value between 0 and 1, where 0 will give no contribution from the previous step, and 1 will give a maximal contribution from the previous step. Must be between 0 and 1.

Example 'Momentum, 0.9
Default 0.6
Data type single | double (0 to 1)

'BatchSize' The size of the mini-batch used for each training iteration. For deepGLM, batch size should be a large number (e.g. 5000, 10000) compared to batch size in deep learning literature (e.g. 128, 256).

Must be a positive integer equal or smaller than number of observations of training data
Example 'BatchSize', 128
Default 5000
Data type single | positive integer

'MaxEpoch' The maximum number of epochs that will be used for training. An epoch is defined as the number of iterations needed for optimization algorithm to scan entire training data. Must be a positive integer.
Example 'MaxEpoch', 1000
Default 100
Data type single | positive integer

'Patience' Number of *consecutive* times that the validation loss is allowed to be larger than or equal to the previously smallest loss before network training is stopped, used as an early stopping criterion. Must be a positive integer.
Example 'Patience, 20
Default 100
Data type single | positive integer

'LrateFactor' Down-scaling factor that is applied to the learning rate every time a certain number of iterations has passed. Must be a positive integer
Example 'LrateFactor', 100 → After the first 100 iterations, learning rate will be multiplied with $\frac{100}{t}$ in each iteration, where *t* is the current number of iteration.
Default 500
Data type single | positive integer

'S' The number of samples needed for Monte Carlo approximation of gradient of lower bound. Must be an positive integer
Example 'S', 20
Default 10
Data type single | positive integer

'WindowSize' Size of moving average window that used to smooth the VB lowerbound. Must be an positive integer.
Example 'WindowSize', 50
Default 200
Data type single | double

'Intercept' Set true (default) to add a column of 1 to predictor observation X matrix (play the role as intercept). If the data have already included the first '1' column, set *'Intercept'* to false.
Example 'Intercept', false
Default true
Data type single | logical

'Verbose' Number of iterations that information on training progress will be printed to the command window each time. Set to 0 to disable this options.

Example 'Verbose', 10 → Display training progress after each 5 iterations.
Default 0
Data type single | integer

'Monitor' Display monitor window showing the training process on a user interface. This is a useful tool to visualize training metrics at every iteration. However, using this option will slow down training progress because of graphical related tasks.
Example 'Monitor', true
Default false
Data type single | logical

'Isotropic' Set to true if you want to use Isotropic structure on Sigma (Variational Covariance matrix). By default, deepGLM uses Diagonal structure to factorize Sigma
Example 'Isotropic', true
Default false
Data type single | logical

'Seed' Seeds the random number generator using the nonnegative integer. Must be a nonnegative integer.
Example 'Seed', 500
Default NaN
Data type single | nonnegative integer

Output

deepGLMfit returns a structure including all input setting by users and output training information with a combination of the following fields:

out.weights Estimated Mean of weights of Deep Neuron Network from input layer to the last hidden layer. This will be used to quickly doing point estimation for new data.

out.beta Estimated Mean of weights from the last hidden layer to output layer. This will be used to quickly doing point estimation for new data.

out.shrinkage Matrix storing estimated values of group Lasso coefficients during training phase.

out.lb A vector storing values of VB lowerbound calculated in each iteration. In the training algorithm, the convergence of lowerbound is used as early stopping rule.

out.lbBar A vector storing values of VB lowerbound after smoothed by a moving average window.

out.vbMU Estimated Mean of Variational Distribution. Used to do prediction interval estimation for new data.

out.vbSIGMA	Estimated Covariance matrix of Variational Distribution. Used to do prediction interval estimation for new data.
out.b	Estimated factor loading vector. Used to calculate the estimated Covariance matrix of Variational Distribution vbSIGMA
out.c	Estimated vector of idiosyncratic noise standard deviation. Used to calculate the estimated Covariance matrix of Variational Distribution vbSIGMA
out.sigma2Alpha	Estimated shape parameter of Invert Gamma Variation Approximation for variance of noise
out.sigma2Beta	Estimated scale parameter of Invert Gamma Variation Approximation for variance of noise
out.sigma2Mean	Estimated mean of covariance of noise in each iteration during training phase
out.nparams	Total number of parameters in deepGLM have to be trained.
out.indexTrack	A vector to keep track the indexes of all weights in Deep Neuron Network. Used to reconstruct Deep Neuron Network from vbMU when doing prediction on new data.
out.CPU	Total amount of training time (in second)

deepGLMpredict

Predict responses using a trained deepGLM structure

Syntax

Pred = deepGLMpredict (mdl, Xtest)
Pred = deepGLMpredict (__ , Name, Value)

Description

Pred = deepGLMpredict (mdl, Xtest) predict responses for new data *Xtest* using trained deepGLM structure *mdl* (output from deepGLMfit)

Pred = deepGLMpredict (__ , Name, Value) predicts responses with additional options specified by one or more of the following name/value pairs:

'ytest' Specify column of test responses. If this option is specified with true response column of new observations, deepGLMpredict will return prediction scores (PPS, MSE or Classification Rate) using true responses column vector *ytest*

'Interval' Return prediction interval estimation for observations in test data *Xtest*. By default, this predictive interval capability is disable ('Interval' is 0). Must be an positive number.

Example 'Interval', 1 → Return one standard deviation predictive interval for new observations.

Default 0

Data type single | double

'Nsample' Number of samples generated from posterior distribution of model parameters used to make prediction interval estimation for test data. Must be a positive integer

Example 'Nsample', 200
Default 1000
Data type single | positive integer

Output

deepGLMpredict returns a structure of prediction results on test data with a combination of the following fields:

yhat Vector of point estimation for *Xtest*. The length of *yhat* is equal to the number of observation of *Xtest.*

yNN Vector of output values when pass Xtest through trained Deep Neuron Network. The length of *yNN* is the number of observation in *Xtest.* If responses are normal, *yNN* is equal to *yhat*.

yProb Vector of estimated probabilities assigned for each observation in *Xtest* belong to class 1. This in only available for binary response data.

pps Partial Predictive Score loss of trained deepGLM on *Xtest*. This is only available when *ytest* is specified.

mse Mean Square Error of trained deepGLM on *Xtest*. This is only available for Normal or Poisson responses and *ytest* is specified.

accuracy Classification rate of trained deepGLM classifier on *Xtest*. This is only available for binary responses and *ytest* is specified.

interval Prediction interval estimation of trained deepGLM on *Xtest*. *Interval* is a two columns matrix where each row is an prediction interval of the corresponding observation of *Xtest*. This is only available when '*Interval*' option is specified with a non-zero value.

deepGLMplot

Plot analytic figures for deepGLM

Syntax

deepGLMplot (type, data)
deepGLMplot (type, data, Name, Value)

Description

deepGLMplot (type, data) Plots data specified in *data1* according the type specified by *type*. Type can be one of the following options:

'Shrinkage' Plot stored values of group Lasso coefficient during training phase. If this type is specify, *data* is the output structure *mdl* from **deepGLMfit** or user can manually extract *mdl.out.shrinkage* field from *mdl* an use as input argument *data.*

'Interval' Plot prediction interval estimation for test data. If this type is specified, *data* is the output structure *Pred* from **deepGLMpredict**.

'ROC' Plot ROC curve for prediction from binary test data. If this type is specify, *data* is a matrix where the first column is a vector of target responses and the second column is the predicted vector *yProb* extract from output structure *Pred* of **deepGLMpredict**. If you want to plot different ROC curves, add more probability columns to *data*.

deepGLMplot (type, data, Name, Value) Plots data specified in *data* according the type specified by *type* with one of the following name/value pairs:

'Nsample' This option only available when *'type'* is *Interval*. *'Nsample'* specifies number of test observations randomly selected from test data to plot prediction intervals.

'Title' Title of the figure

'Xlabel' Label of X axis

'Ylabel' Label of Y axis

'LineWidth' Line width of the plots

'Legend' Legend creates a legend with descriptive labels for each plotted data series

Practical Recommendation

To make it easy to use deepGLM, we have already set the default settings for deepGLM that users can use in the first attempt of training a deepGLM model. However, these default setting can be inappropriate for some dataset that need a more sophisticated tuning of hyperparameters to effectively train a deepGLM model.

Training deepGLM is hard because there are a lot of possible combinations of hyper-parameters can be fed into ***deepGLMfit***. There is always a trade-off between reducing training time and improving predictive performance of the trained model. The following recommendations give some heuristic guidelines when users want to improve the performance of deepGLM in each of the case:

Improving predictive performance of deepGLM model

Smoothing lowerbound

- DeepGLM uses the lowerbound of KL divergence between variational distribution and true posterior distribution of model parameters to make early stopping criterion and model selection. For this reason, we have to make sure that the model

settings have to produce a smooth lowerbound (clear increasing trend) during training phase.

- If the lowerbound does not have increasing trend or significantly fluctuate, then we can modify these hyperparameters to make a smooth lowerbound:
  - Increase batch size by modifying *'BatchSize'* option of ***deepGLMfit***. Increasing batch size helps reducing the variance of Monte Carlo estimation of log-likelihood contribution of lowerbound.
  - Increase moving average window size by modifying *'WindowSize'* option of ***deepGLMfit***. Increasing window size helps smoothing fluctuated lowerbound.
  - Reducing learning rate by modifying *'Lrate'* option. This helps lowerbound not 'overshoot' when it moves closely to saturated region.
  - Increase *'S'* option of deepGLMfit (e.g. 'S', 20). This helps to reduce the variance of MC estimation of log-likelihood contribution of lowerbound

Try various neuron network structures

- Modify option *'Network'* of ***deepGLMfit***
- An inappropriate structure may cause high bias on test data.
- Start with a simple network (e.g. 'Network', [5,5]) then increasing the network complexity until the predictive performance gets worse on test data.
- If the network is too complicated, it will take longer time for training. Additionally, increasing network complexity (adding more layers) does not ensure an improvement in predictive performance of deepGLM

Decreasing learning rate

- Decrease *'Lrate'* option of ***deepGLMfit***
- A too small learning rate will slow down training process.
- If the total number of iterations needed for training is smaller than *'LrateFactor'*, then reducing *'LrateFactor'* could help to decrease learning rate.

Increase number of training iterations

- Increase *'Patience'* option of ***deepGLMfit***. This is to relax the early stopping criteria.
- Increase *'MaxEpoch'* option of ***deepGLMfit*.** This is to relax the early stopping criteria.

Others

- Try different valid values of *'Momentum, BatchSize'* options may help to improve prediction accuracy on test data.
- Increase *'S'* option of deepGLMfit (e.g. 'S', 20). A small number of S may cause high variation in the Monte Carlo estimation of gradient of lower bound.

Reduce training time

Increase learning rate

- Increase *'Lrate'* option of ***deepGLMfit*** (e.g. 'Lrate', 0.1)
- If learning is too high, training algorithm may be failed to converge

Decrease number of training iterations

- Decrease *'Patience'* option of ***deepGLMfit*** (e.g. 'Patience', 10).
- Increase *'MaxEpoch'* option of ***deepGLMfit*.**

Simplify Neuron Network structure

- Set simpler structure for *'Network'* option of ***deepGLMfit***.
- A too simple Neuron Network structure does not imply that deepGLM will have worse performance. The complexity of Neuron structure depends on the complexity of training data.

Use Isotropic structure for very complex neuron network structure

- Enable *'Isotropic'* option of ***deepGLMfit*** to *true* (e.g. 'Isotropic', true)

Others

- Disable *'Monitor'* option of ***deepGLMfit*** to *'false'* (e.g.' Monitor', false)
- Decrease *'S'* option of deepGLMfit (e.g. 'S', 5). A small number of S may cause high variation in the Monte Carlo estimation of gradient of lower bound.