Package 'deepLearnR'

March 12, 2016

Title Interface to TensorFlow Deeplearning Framework

Version 0.0.0.9000

Date 2016-03-13

Description Package provides Classifier with different architectures - simple Linear Neural Network, Deep Neural Network and Recursive Neural Network (rnn,gru & lstm). The package interfaces with Tensorflow via the skflow python package.

Depends R (>= 3.2.3)

SystemRequirements Python pckages TensorFlow, skflow and Pandas

License GPL-3

LazyData true

Imports rPython, RJSONIO

RoxygenNote 5.0.1

Repository CRAN

Suggests knitr, rmarkdown, testthat, MASS

VignetteBuilder knitr

NeedsCompilation no

Author Krishna Sankar [aut, cre],
Billy Vreeland [aut],
Tej Azad [aut]

Maintainer Krishna Sankar <ksankar42@gmail.com>

R topics documented:

deepLearnR *Deep Learnning interface to TensorFlow from R Leverage the distributed multicore capabilites of TensorFlow*

Description

Functions to create deepLearning architectures and associated datasets. Requires tensorFlow 0.7.0, skflow and rPython installed. Works with default python, not anaconda python installations. See examples for functions `TensorFlow.Classifier`, `TensorFlow.predict`, `TensorFlowDNNRegressor`, `TensorFlow.regressorEval` and `TensorFlow.SystemLinReg`.

References

[1] rPython and data in and out of pandas https://statcompute.wordpress.com/2013/10/13/rpython-r-interface-to-python/

[2] some python code refactored from skflow examples in Tutorials (1,2 & 3) by Illia Polosukhin

https://medium.com/@ilblackdragon/tensorflow-tutorial-part-1-c559c63c0cb1#.njjgnw8yh

https://medium.com/@ilblackdragon/tensorflow-tutorial-part-2-9ffe47049c92#.xxksiy8gg

https://medium.com/@ilblackdragon/tensorflow-tutorial-part-3-c5fc0662bc08#.md7qum553

[3] python code from skflow examples https://github.com/tensorflow/skflow/tree/master/examples

See Also

`TensorFlow.Classifier`

`TensorFlow.predict`

`TensorFlowDNNRegressor`

`TensorFlow.regressorEval`

`TensorFlow.SystemLinReg`

GeneratePythonScript *Generate a TensorFlow Python script, internal function used by deepLearnR*

Description

Generate a TensorFlow Python script, internal function used by deepLearnR

Usage

```
GeneratePythonScript(epochs = 1e+05, learning.rate = 1e-04)
```

Arguments

epochs	number of epochs to use in the model
learning.rate	learning rate to use in the model

GetSerializedTFData — *Get serialized data values back following python execution, internal function used by deepLearnR*

Description

Get serialized data values back following python execution, internal function used by deepLearnR

Usage

```
GetSerializedTFData(file.name)
```

Arguments

file.name	file name from which to read data

SerializeTFData — *Serialize data to JSON and write to file, internal function used by deepLearnR*

Description

Serialize data to JSON and write to file, internal function used by deepLearnR

Usage

```
SerializeTFData(data, file.name)
```

Arguments

data	data to serialize
file.name	output file name

TensorFlow.Classifier — *Create Classifier model based on the parameters*

Description

Create Classifier model based on the parameters

Usage

```
TensorFlow.Classifier(modelTag, XTrain, YTrain, nClasses = 2,
  miniBatchSize = 128, steps = 500, optimizer = "SGD",
  learningRate = 0.05, hiddenUnits = c(10, 20, 10), rnnSize = 100,
  nnType = "linear", netType = "ReLU", cellType = "lstm")
```

Arguments

modelTag	Tag for this model - can be referenced in other calls like prediction
XTrain	The X Matrix for training
YTrain	The Y Matrix for training
nClasses	The number of classes
miniBatchSize	Batch Size for the mini batch for optimization algorithms like SGD
steps	Number of epochs for training
optimizer	The Optimizer algorithm = "SGD", "Adam","Adagrad" (only "SGD" tested, others ignored)
learningRate	The learning rate for optimize algorithm
hiddenUnits	The number and architecture of hidden unit layers for dnn e.g. [10,20,10]
rnnSize	The size of the rnn cell, e.g. size of your word embeddings
nnType	The network type = "linear", "dnn", "rnn" ("rnn" is not implemented, but included for completeness of the interface & future implementation)
netType	The network type for the final round = "ReLU","tanh"
cellType	The cell type for rnn network = "rnn","gru","lstm" (not implemented, but included for completeness of the interface & future implementation)

See Also

TensorFlow.predict

Examples

```
{
 Y <- deepLearnR::titanic.data$Survived
 X <- deepLearnR::titanic.data[,c("Age","SibSp","Fare","Pclass")]
 X$Age[is.na(X$Age)] <- mean(X$Age,na.rm=TRUE)
 set.seed(512)
 inTrain <- sample(1:nrow(X), trunc(nrow(X)*0.8))
 X.Train <- X[inTrain,]
 Y.Train <- Y[inTrain]
 X.Test <- X[-inTrain,]
 Y.Test <- Y[-inTrain]
 deepLearnR::TensorFlow.Classifier(modelTag="tflr-03",X=X.Train,Y=Y.Train,steps=5000)
 pred <- deepLearnR::TensorFlow.predict(modelTag="tflr-03",X=X.Test,Y=Y.Test)
 accuracy <- sum(pred == Y.Test)/length(Y.Test)
 print(accuracy) # Should be ~ 0.6312849
 pred <-  deepLearnR::TensorFlow.predict(modelTag="tflr-03",X=X,Y=Y)
 accuracy <- sum(pred == Y)/length(Y)
 print(accuracy) # Should be ~ 0.6397306
}
```

`TensorFlow.predict` *Predict using a model(modelTag) the Yvalues for the X Matrix*

Description

Predict using a model(modelTag) the Yvalues for the X Matrix

Usage

```
TensorFlow.predict(modelTag, XTest, YTest = NULL, calculateAccuracy = TRUE)
```

Arguments

`modelTag`	Tag for this model - referenced in the model ceate calls like TensorFlow.Classifier
`XTest`	The X Matrix for test or prediction
`YTest`	The Y Matrix for test, to calculate the accuracy
`calculateAccuracy`	Yes/No to calculate the accuracy from skflow. As a check

See Also

`TensorFlow.Classifier`

`TensorFlow.regressorEval` *Predict using a model(modelTag) the Yvalues for the X Matrix*

Description

Predict using a model(modelTag) the Yvalues for the X Matrix

Usage

```
TensorFlow.regressorEval(modelTag, calculateMSE = TRUE, calculateR2 = TRUE)
```

Arguments

`modelTag`	Tag for this model - referenced in the model ceate calls like TensorFlow.Classifier
`calculateMSE`	Yes/No to calculate the MSE
`calculateR2`	Yes/No to compute R2

TensorFlow.SystemLinReg

TensoFlow linear regression implementation

Description

TensoFlow linear regression implementation

Usage

```
TensorFlow.SystemLinReg(X, Y, epochs = 1e+05, learning.rate = 1e-04)
```

Arguments

X	the dependent variables in the model (currently only supports 1-D numeric vector)
Y	the independent variable in the model
epochs	number of epochs to use in the model
learning.rate	learning rate to use in the model

Examples

```
{
x.vals <- seq(1:100)
y.vals <- 0.3 + 0.5 * x.vals
lm.tf.fit <- deepLearnR::TensorFlow.SystemLinReg(X = x.vals, Y = y.vals,
                          epochs = 100000, learning.rate = .00005)
lm.tf.fit
}
```

TensorFlowDNNRegressor

Create Classifier model based on the parameters

Description

Create Classifier model based on the parameters

Usage

```
TensorFlowDNNRegressor(modelTag, X, y, test_size = 0.2, steps = 5000,
  learningRate = 0.1, batchSize = 1)
```

Arguments

modelTag	Tag for this model - can be referenced in other calls like prediction
X	The X Matrix for training
y	The y Matrix for training
test_size	The division of the dataset into training vs test
steps	Number of epochs for training
learningRate	The learning rate for optimize algorithm
batchSize	Batch Size for the mini batch for optimization algorithms like SGD

Examples

```
{
library(MASS)
data(Boston)
X <- Boston[,2:14]
y <- Boston[,1]

TensorFlowDNNRegressor(modelTag="tfdnnr-01", X=X, y=y, steps=5000)
pred <- TensorFlow.regressorEval(modelTag="tfdnnr-01")
mse <- rPython::python.get("mse")
r2 <- rPython::python.get("r2")
}
```

---

titanic.data *The titanic Dataset*

Description

A test data set of 12 variables

- PassengerId. Id of the passenger
- Survived. Whether they survived or not
- Pclass. Class (1,2,3)
- Name. Name of the passenger
- Sex.
- Age.
- Sibsp. Number of Siblings/Spouses Aboard
- Parch. Number of Parents/Children Aboard
- Ticket. Ticket Number
- Fare.
- Cabin.
- Embarked. Port of Embarkation

Index