Package ‘RSofia’

March 12, 2013

Type Package

Title Port of sofia-ml (http://code.google.com/p/sofia-ml/) to R

Version 1.1

Date 2011-09-06

Author Michael King <wmichaelking1@gmail.com> and Fernando Cela Diaz <fcela@sloan.mit.edu>. Original sofia-ml code by D. Sculley <dsculley@google.com>.

Maintainer Michael King <wmichaelking1@gmail.com>

Description sofia-ml is a suite of fast incremental algorithms for machine learning that can be used for training models for classification or ranking

License Apache License 2.0

LazyLoad yes

Depends methods, Rcpp (>= 0.9.6)

Suggests RUnit

LinkingTo Rcpp

RcppModules sofia

Repository CRAN

Date/Publication 2011-12-18 00:23:49

R topics documented:

RSofia-package *Train and Test Suite of Models from Sofia-ml*

Description

Sofia-ml is a suite of fast incremental algorithms for machine learning that can be used for training models for classification or ranking

Details

Package:	RSofia
Type:	Package
Version:	1.0
Date:	2011-09-06
License:	apache
LazyLoad:	yes

Author(s)

W. Michael King <wmichaelking1@gmail.com> and

Fernando Cela Diaz <<fcela@sloan.mit.edu>.

Original sofia-ml code by D. Sculley <dsculley@google.com>.

Maintainer: W. Michael King <wmichaelking1@gmail.com>

References

D. Sculley. *Combined Regression and Ranking*. Proceedings of the 16th Annual SIGKDD Conference on Knowledge Discover and Data Mining, 2010.

D. Sculley. *Web-Scale K-Means Clustering*. Proceedings of the 19th international conference on World Wide Web, 2010.

D. Sculley. *Large Scale Learning to Rank*. NIPS Workshop on Advances in Ranking, 2009. Presents the indexed sampling methods used learning to rank, including the rank and roc loops.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. *Online passive-aggressive algorithms*. J. Mach. Learn. Res., 7, 2006. Presents the Passive-Aggressive Perceptron algorithm.

T. Joachims. *Optimizing search engines using clickthrough data*. In KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002. Presents the RankSVM objective function, a pairwise objective function used by the rank loop method in sofia-ml.

Y. Li and P. M. Long. *The relaxed online maximum margin algorithm.* Mach. Learn., 46(1-3), 2002. Presents the ROMMA algorithm.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: *Primal estimated sub-gradient solver for SVM*. In ICML '07: Proceedings of the 24th international conference on Machine learning, 2007. Presents the Pegasos SVM solver.

T. Zhang. *Solving large scale linear prediction problems using stochastic gradient descent algorithms*. In ICML '04: Proceedings of the twenty-first international conference on Machine learning, 2004. Presents SGD SVM.

`http://leon.bottou.org/projects/sgd` Leon Bottou's SGD page, including experiments with SGD SVM.

W. Krauth and M. M'ezard. *Learning algorithms with optimal stability in neural networks*. Journal of Physics A, 20(11):745-752, 1987. Presents Perceptron with Margins.

`irismod` *A Slight Modification to Edgar Anderson's Iris Data*

Description

The famous iris data set, but with the species column altered from factor to binary and renamed `Is.Virginica`. 1's represent the Virginica Species whereas the (-1)'s represent "veriscolor" or "virginica".

Usage

```
irismod
```

Format

A dataframe

Source

Fischer, R. A. (1935) The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**, Part II, 179–188

`parse_formula` *parse a Dataframe for use with* `"sofia.fit"`

Description

Function for parsing Dataframe for use with `"sofia.fit"`.

Usage

```
parse_formula(formula, data)
```

Arguments

`formula` an object of class `"formula"`

`data` a data frame

Value

`"parse_formula"` returns a list containing the following components:

`data` a numeric matrix. The explanatory variables for the model

`labels` a numeric vector. The response variable

Examples

```
data(irismod)
parse_formula(Is.Virginica ~ . , irismod)
```

`predict.sofia` *Predict Method for Sofia-ml Model Fits*

Description

Predicted values based on Sofia-ml model object.

Usage

```
## S3 method for class 'sofia'
predict(object, newdata, prediction_type, ... )
```

Arguments

`object` sofia-ml object

`newdata` If the model was fit with `sofia.formula`, a data frame; if it was fit with `sofia.fit`, a new set of X's

`prediction_type` `linear`: use standard linear dot product <w,x> for predictions; `logistic`: use prediction function of exp(<w,x>) / (1 + exp(<w,x>)) for prediction, in the manner of logistic regression

`...` unused

Value

`predict.sofia` produces a numeric vector of predictions

See Also

sofia

Examples

```
data(irismod)
i.TRAIN <- sample(1:nrow(irismod), 100)

model.logreg <- sofia(Is.Virginica ~ ., data=irismod[i.TRAIN,], learner_type="logreg-pegasos")
p <- predict(model.logreg, newdata=irismod[-1*i.TRAIN,], prediction_type = "logistic")
table(predicted=p>0.5, actual=irismod[-1*i.TRAIN,]$Is.Virginica)

model.pegasos <- sofia(Is.Virginica ~ ., data=irismod[i.TRAIN,], learner_type="pegasos")
d <- predict(model.pegasos, newdata=irismod[-1*i.TRAIN,], prediction_type = "linear")
table(predicted=d>0, actual=irismod[-1*i.TRAIN,]$Is.Virginica)
```

---

read.svmlight *Read Files in SVM-Light Format*

Description

Read datasets in SVM-Light sparse data format:

```
<class-label> <feature-id>:<feature-value> ... <feature-id>:<feature-value>\n
```

```
<class-label> qid:<optional-query-id> <feature-id>:<feature-value> ... <feature-id>:<feature-value>\
```

```
<class-label> <feature-id>:<feature-value> ... <feature-id>:<feature-value># Optional comment or extr
```

Usage

```
read.svmlight(file)
```

Arguments

`file`	a character string giving the name of the file to read.

Details

I don't believe this implementation of "read.svmlight" to be particularly robust and should be used with caution.

Value

read.svmlight returns a list containing the following components:

data a numeric matrix. The explanatory variables for the model

labels a numeric vector. The response variable

no_bias_term NULL

See Also

write.svmlight

Examples

```
data(irismod)

x <- parse_formula(Is.Virginica ~ ., irismod)

tmp <- tempfile()

write.svmlight(x$labels, x$data, file = tmp)

irismod.svmlight <- read.svmlight(tmp)

unlink(tmp)
```

sofia *Fitting sofia-ml models*

Description

sofia is used to fit classification and regression models provided by D. Sculley's sofia-ml.

Usage

```
 sofia(x, ...)

## S3 method for class 'formula'
sofia(x, data, random_seed = floor(runif(1, 1, 65535)), lambda = 0.1,
    iterations = 1e+05, learner_type = c("pegasos", "sgd-svm",
        "passive-aggressive", "margin-perceptron", "romma", "logreg-pegasos"),
    eta_type = c("pegasos", "basic", "constant"), loop_type = c("stochastic",
        "balanced-stochastic", "rank", "roc", "query-norm-rank",
        "combined-ranking", "combined-roc"), rank_step_probability = 0.5,
    passive_aggressive_c = 1e+07, passive_aggressive_lambda = 0,
    perceptron_margin_size = 1, training_objective = FALSE, hash_mask_bits = 0,
    verbose = FALSE, reserve = 0, ...)

## S3 method for class ’character’
sofia(x, random_seed = floor(runif(1, 1, 65535)), lambda = 0.1,
    iterations = 1e+05, learner_type = c("pegasos", "sgd-svm",
        "passive-aggressive", "margin-perceptron", "romma", "logreg-pegasos"),
    eta_type = c("pegasos", "basic", "constant"), loop_type = c("stochastic",
        "balanced-stochastic", "rank", "roc", "query-norm-rank",
        "combined-ranking", "combined-roc"), rank_step_probability = 0.5,
    passive_aggressive_c = 1e+07, passive_aggressive_lambda = 0,
   perceptron_margin_size = 1, training_objective = FALSE,no_bias_term = FALSE, dimensionality=150000,
    verbose = FALSE, buffer_mb = 40, ...)
```

Arguments

`x` a `formula` object or a character with a path to a file

`data` data to parse formula on, when model is specified via a formula

`random_seed` an integer. Makes algorithm use this seed. Can be useful in testing and parameter tuning

`lambda` a numeric scalar. Value of lambda for SVM regularization, used by both Pegasos SVM and SGD-SVM.

`iterations` an integer. Number of stochastic gradient steps to take.

`learner_type` a character string indicating which type of learner to use. One of `"pegasos"` (default), `"sgd-svm"`, `"passive-aggressive"`, `"margin-perceptron"`, `"romma"`, `"logreg-pegasos"`

`eta_type` a character string indicating the type of update for learning rate to use. One of `"pegasos"` (default), `"basic"`, `"constant"`

`loop_type` a character string indicating the type of sampling loop to use for training. One of

`"stochastic"` - Perform normal stochastic sampling for stochastic gradient descent, for training binary classifiers. On each iteration, pick a new example uniformly at random from the data set.

`"balanced-stochastic"` - Perform a balanced sampling from positives and negatives in data set. For each iteration, samples one positive example uniformly at random from the set of all positives, and samples one negative example uniformly at random from the set of all negatives. This can be useful for training binary classifiers with a minority-class distribution.

`"rank"` - Perform indexed sampling of candidate pairs for pairwise learning to rank. Useful when there are examples from several different qid groups.

`"roc"` - Perform indexed sampling to optimize ROC Area.

`"query-norm-rank"` - Perform sampling of candidate pairs, giving equal weight to each qid group regardless of its size. Currently this is implemented with rejection sampling rather than indexed sampling, so this may run more slowly.

`"combined-ranking"` - Performs CRR algorithm for combined regression and ranking. Alternates between pairwise rank-based steps and standard stochastic gradient steps on single examples. Relies on `"rank_step_probability"` to balance between these two kinds of updates.

`"combined-roc"` - Performs CRR algorithm for combined regression and ROC area optimization. Alternates between pairwise roc-optimization-based steps and standard stochastic gradient steps on single examples. Relies on `"rank_step_probability"` to balance between these two kinds of updates. This can be faster than the combined-ranking option when there are exactly two classes.

`rank_step_probability`
a numeric scalar. Probability that we will take a rank step (as opposed to a standard stochastic gradient step) in a combined ranking or combined ROC loop.

`passive_aggressive_c`
a numeric scalar. Maximum size of any step taken in a single passive-aggressive update

`passive_aggressive_lambda`
a numeric scalar. Lambda for pegasos-style projection for passive-aggressive update. When set to 0 (default) no projection is performed.

`perceptron_margin_size`
Width of margin for perceptron with margins. Default of 1 is equivalent to unregularized SVM-loss

`training_objective`
logical. When `TRUE`, computes the value of the standard SVM objective function on training data, after training.

`dimensionality` integer. Index id of largest feature index in training data set, plus one.

`hash_mask_bits` an integer. When set to a non-zero value, causes the use of a hased weight vector with hashed cross product features. This allows learning on conjunction of features, at some increase in computational cost. Note that this flag must be set both in training and testing to function properly. The size of the hash table is set to `2^hash_mask_bits`. default value of 0 shows that hash cross products are not used.

`verbose` logical.

`no_bias_term` logical. When set, causes a bias term x_0 to be set to 0 for every feature vector loaded from files, rather than the default of x_0 = 1. Setting this flag is equivalent to forcing a decision threshold of exactly 0 to be used. The same setting of this flag should be used for training and testing. Note that this flag as no effect for rank and roc optimzation. Default: not set. To set this flag using the formula interface use (Y ~ -1 + .)

`reserve` integer. experimental, should vector be explicity reserved for data?

`buffer_mb` integer. Size of buffer to use in reading/writing to files, in MB.

`...` items passed to methods.

Value

`sofia` returns an object of class "sofia".

An object of class "sofia" is a list containing at least the following components:

`par` a list containing the parameters specified in training the model

`weights`	a numeric vector of the parameter weights (the model)
`training_time`	time used to fit the model (does not include io time)

If the method was called via the formula interface, it will additionally include:

`formula`	formula with the specification of the model

References

D. Sculley. *Combined Regression and Ranking*. Proceedings of the 16th Annual SIGKDD Conference on Knowledge Discover and Data Mining, 2010.

D. Sculley. *Web-Scale K-Means Clustering*. Proceedings of the 19th international conference on World Wide Web, 2010.

D. Sculley. *Large Scale Learning to Rank*. NIPS Workshop on Advances in Ranking, 2009. Presents the indexed sampling methods used learning to rank, including the rank and roc loops.

See Also

`http://code.google.com/p/sofia-ml/`

Examples

```
data(irismod)

model.logreg <- sofia(Is.Virginica ~ ., data=irismod, learner_type="logreg-pegasos")
```

`write.svmlight` *Write Files in SVM-Light Format*

Description

Write datasets in SVM-Light sparse data format

Usage

```
write.svmlight(labels, data, file, ...)
```

Arguments

`labels`	numeric. labels for dataset
`data`	numeric matrix. explanatory variables
`file`	a character string giving the name of the file to be written to.
`...`	unused

See Also

read.svmlight

Examples

```
data(irismod)

x <- parse_formula(Is.Virginica ~ . , irismod)

tmp <- tempfile()

write.svmlight(x$labels, x$data, tmp);

readLines(tmp)

unlink(tmp)
```

Index