Package ‘sgd’

August 29, 2016

Type Package

Title Stochastic Gradient Descent for Scalable Estimation

Version 1.1

Maintainer Dustin Tran <dustin@cs.columbia.edu>

Description A fast and flexible set of tools for large scale estimation. It features many stochastic gradient methods, built-in models, visualization tools, automated hyperparameter tuning, model checking, interval estimation, and convergence diagnostics.

URL https://github.com/airoldilab/sgd

BugReports https://github.com/airoldilab/sgd/issues

License GPL-2

Suggests bigmemory, gridExtra, R.rsp, testthat

Imports ggplot2, MASS, methods, Rcpp (>= 0.11.3)

LinkingTo BH, bigmemory, Rcpp, RcppArmadillo

VignetteBuilder R.rsp

NeedsCompilation yes

Author Dustin Tran [aut, cre],
Panos Toulis [aut],
Tian Lian [ctb],
Ye Kuang [ctb],
Edoardo Airoldi [ctb]

Repository CRAN

Date/Publication 2016-01-05 21:12:16

R topics documented:

`plot.sgd` *Plot objects of class* `sgd`.

Description

Plot objects of class sgd.

Usage

```
## S3 method for class 'sgd'
plot(x, ..., type = "mse", xaxis = "iteration")

## S3 method for class 'list'
plot(x, ..., type = "mse", xaxis = "iteration")
```

Arguments

`x`	object of class sgd.
`type`	character specifying the type of plot: `"mse"`, `"clf"`, `"mse-param"`. See 'Details'. Default is `"mse"`.
`xaxis`	character specifying the x-axis of plot: `"iteration"` plots the y values over the log-iteration of the algorithm; `"runtime"` plots the y values over the time in seconds to reach them. Default is `"iteration"`.
`...`	additional arguments used for each type of plot. See 'Details'.

Details

Types of plots available:

`mse` Mean squared error in predictions, which takes the following arguments:

- `x_test` test set
- `y_test` test responses to compare predictions to

`clf` Classification error in predictions, which takes the following arguments:

- `x_test` test set
- `y_test` test responses to compare predictions to

`mse-param` Mean squared error in parameters, which takes the following arguments:

- `true_param` true vector of parameters to compare to predict.sgd *Predict for objects of class* sgd

Description

Form predictions using the estimated model parameters from stochastic gradient descent.

Usage

```
## S3 method for class 'sgd'
predict(object, x_test, ...)

predict_all(object, x_test, ...)
```

Arguments

object	object of class sgd.
x_test	design matrix to form predictions on
...	further arguments passed to or from other methods.

Details

A column of 1's must be included if the parameters include a bias, or intercept, term.

print.sgd *Print objects of class* sgd.

Description

Print objects of class sgd.

Usage

```
## S3 method for class 'sgd'
print(x, ...)
```

Arguments

x	object of class sgd.
...	further arguments passed to or from other methods.

sgd *Stochastic gradient descent*

Description

Run stochastic gradient descent in order to optimize the induced loss function given a model and data.

Usage

```
sgd(x, ...)

## S3 method for class 'formula'
sgd(formula, data, model, model.control = list(),
  sgd.control = list(...), ...)

## S3 method for class 'function'
sgd(x, ...)

## S3 method for class 'matrix'
sgd(x, y, model, model.control = list(),
  sgd.control = list(...), ...)

## S3 method for class 'big.matrix'
sgd(x, y, model, model.control = list(),
  sgd.control = list(...), ...)
```

Arguments

`x,y` a design matrix and the respective vector of outcomes.

`formula` an object of class "`formula`" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details can be found in "`glm`".

`data` an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm is called.

`model` character specifying the model to be used: "`lm`" (linear model), "`glm`" (generalized linear model), "`cox`" (Cox proportional hazards model), "`gmm`" (generalized method of moments), "`m`" (M-estimation). See 'Details'.

`model.control` a list of parameters for controlling the model.

`family` ("`glm`") a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See `family` for details of family functions.)

`rank` ("`glm`") logical. Should the rank of the design matrix be checked?

- fn ("gmm") a function $g(\theta, x)$ which returns a k-vector corresponding to the k moment conditions. It is a required argument if gr not specified.
- gr ("gmm") a function to return the gradient. If unspecified, a finite-difference approximation will be used.
- nparams ("gmm") number of model parameters. This is automatically determined for other models.
- type ("gmm") character specifying the generalized method of moments procedure: "twostep" (Hansen, 1982), "iterative" (Hansen et al., 1996). Defaults to "iterative".
- wmatrix ("gmm") weighting matrix to be used in the loss function. Defaults to the identity matrix.
- loss ("m") character specifying the loss function to be used in the estimating equation. Default is the Huber loss.
- lambda1 L1 regularization parameter. Default is 0.
- lambda2 L2 regularization parameter. Default is 0.

sgd.control an optional list of parameters for controlling the estimation.

- method character specifying the method to be used: "sgd", "implicit", "asgd", "ai-sgd", "momentum", "nesterov". Default is "ai-sgd". See 'Details'.
- lr character specifying the learning rate to be used: "one-dim", "one-dim-eigen", "d-dim", "adagrad", "rmsprop". Default is "one-dim". See 'Details'.
- lr.control vector of scalar hyperparameters one can set dependent on the learning rate. For hyperparameters aimed to be left as default, specify NA in the corresponding entries. See 'Details'.
- start starting values for the parameter estimates. Default is random initialization around zero.
- size number of SGD estimates to store for diagnostic purposes (distributed log-uniformly over total number of iterations)
- reltol relative convergence tolerance. The algorithm stops if it is unable to change the relative mean squared difference in the parameters by more than the amount. Default is 1e-05.
- npasses the maximum number of passes over the data. Default is 3.
- pass logical. Should tol be ignored and run the algorithm for all of npasses?
- shuffle logical. Should the algorithm shuffle the data set including for each pass?
- verbose logical. Should the algorithm print progress?

... arguments to be used to form the default sgd.control arguments if it is not supplied directly.

Details

Models: The Cox model assumes that the survival data is ordered when passed in, i.e., such that the risk set of an observation i is all data points after it.

Methods:

sgd stochastic gradient descent (Robbins and Monro, 1951) `implicit` implicit stochastic gradient descent (Toulis et al., 2014)

`asgd` stochastic gradient with averaging (Polyak and Juditsky, 1992)

`ai-sgd` implicit stochastic gradient with averaging (Toulis et al., 2015)

`momentum` "classical" momentum (Polyak, 1964)

`nesterov` Nesterov's accelerated gradient (Nesterov, 1983)

Learning rates and hyperparameters:

`one-dim` scalar value prescribed in Xu (2011) as

$$a_n = scale * gamma/(1 + alpha * gamma * n)^(-c)$$

where the defaults are `lr.control = (scale=1, gamma=1, alpha=1, c)` where `c` is `1` if implemented without averaging, `2/3` if with averaging

`one-dim-eigen` diagonal matrix `lr.control = NULL`

`d-dim` diagonal matrix `lr.control = (epsilon=1e-6)`

`adagrad` diagonal matrix prescribed in Duchi et al. (2011) as `lr.control = (eta=1, epsilon=1e-6)`

`rmsprop` diagonal matrix prescribed in Tieleman and Hinton (2012) as `lr.control = (eta=1, gamma=0.9, epsilon=1e-6`

Value

An object of class `"sgd"`, which is a list containing the following components:

`model`	name of the model
`coefficients`	a named vector of coefficients
`converged`	logical. Was the algorithm judged to have converged?
`estimates`	estimates from algorithm stored at each iteration specified in `pos`
`pos`	vector of indices specifying the iteration number each estimate was stored for
`times`	vector of times in seconds it took to complete the number of iterations specified in `pos`
`model.out`	a list of model-specific output attributes

Author(s)

Dustin Tran, Tian Lan, Panos Toulis, Ye Kuang, Edoardo Airoldi

References

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121-2159, 2011.

Yurii Nesterov. A method for solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372-376, 1983.

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1-17, 1964.

Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838-855, 1992.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pp. 400-407, 1951.

Panos Toulis, Jason Rennie, and Edoardo M. Airoldi, "Statistical analysis of stochastic gradient methods for generalized linear models", In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Panos Toulis, Dustin Tran, and Edoardo M. Airoldi, "Stability and optimality in stochastic gradient descent", arXiv preprint arXiv:1505.02417, 2015.

Wei Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. arXiv preprint arXiv:1107.2490, 2011.

Examples

```
## Linear regression
set.seed(42)
N <- 1e4
d <- 10
X <- matrix(rnorm(N*d), ncol=d)
theta <- rep(5, d+1)
eps <- rnorm(N)
y <- cbind(1, X) %*% theta + eps
dat <- data.frame(y=y, x=X)
sgd.theta <- sgd(y ~ ., data=dat, model="lm")
sprintf("Mean squared error: %0.3f", mean((theta - as.numeric(sgd.theta$coefficients))^2))

## Wine quality (Cortez et al., 2009): Logistic regression
set.seed(42)
data("winequality")
dat <- winequality
dat$quality <- as.numeric(dat$quality > 5) # transform to binary
test.set <- sample(1:nrow(dat), size=nrow(dat)/8, replace=FALSE)
dat.test <- dat[test.set, ]
dat <- dat[-test.set, ]
sgd.theta <- sgd(quality ~ ., data=dat,
               model="glm", model.control=binomial(link="logit"),
               sgd.control=list(reltol=1e-5, npasses=200),
                 lr.control=c(scale=1, gamma=1, alpha=30, c=1))
sgd.theta
```

`winequality` *Wine quality data of white wine samples from Portugal*

Description

This dataset is a collection of white "Vinho Verde" wine samples from the north of Portugal. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

Usage

```
winequality
```

Format

A data frame with 4898 rows and 12 variables

- fixed acidity.
- volatile acidity.
- citric acid.
- residual sugar.
- chlorides.
- free sulfur dioxide.
- total sulfur dioxide.
- density.
- pH.
- sulphates.
- alcohol.
- quality (score between 0 and 10).

Source

https://archive.ics.uci.edu/ml/datasets/Wine+Quality

Index