SGDinference: An R Vignette

Introduction

SGDinference is an R package that provides estimation and inference methods for large-scale mean and quantile regression models via stochastic (sub-)gradient descent (S-subGD) algorithms. The inference procedure handles cross-sectional data sequentially:

(i) updating the parameter estimate with each incoming "new observation",
(ii) aggregating it as a Polyak-Ruppert average, and
(iii) computing an asymptotically pivotal statistic for inference through random scaling.

The methodology used in the SGDinference package is described in detail in the following papers:

- Lee, S., Liao, Y., Seo, M.H. and Shin, Y., 2022. Fast and robust online inference with stochastic gradient descent via random scaling. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 36, No. 7, pp. 7381-7389). https://doi.org/10.1609/aaai.v36i7.20701.
- Lee, S., Liao, Y., Seo, M.H. and Shin, Y., 2023. Fast Inference for Quantile Regression with Tens of Millions of Observations. arXiv:2209.14502 [econ.EM] https://doi.org/10.48550/arXiv.2209.14502.

We begin by calling the SGDinference package.

```
library(SGDinference)
set.seed(100723)
```

Case Study: Estimating the Mincer Equation

To illustrate the usefulness of the package, we use a small dataset included in the package. Specifically, the *Census2000* dataset from Acemoglu and Autor (2011) consists of observations on 26,120 nonwhite, female workers. This small dataset is constructed from "microwage2000_ext.dta" at https://economics.mit.edu/people/faculty/david-h-autor/data-archive. Observations are dropped if hourly wages are missing or years of education are smaller than 6. Then, a 5 percent random sample is drawn to make the dataset small. The following three variables are included:

- ln_hrwage: log hourly wages
- edyrs: years of education
- exp: years of potential experience

We now define the variables.

```
    y = Census2000$ln_hrwage
  edu = Census2000$edyrs
  exp = Census2000$exp
 exp2 = exp^2/100
```

As a benchmark, we first estimate the Mincer equation and report the point estimates and their 95% heteroskedasticity-robust confidence intervals.

```
mincer = lm(y ~ edu + exp + exp2)
inference = lmtest::coefci(mincer, df = Inf,
                           vcov = sandwich::vcovHC)
results = cbind(mincer$coefficients,inference)
colnames(results)[1] = "estimate"
```

```r
print(results)
#>                estimate       2.5 %      97.5 %
#> (Intercept)  0.58114741  0.52705757  0.63523726
#> edu          0.12710477  0.12329983  0.13090971
#> exp          0.03108721  0.02877637  0.03339806
#> exp2        -0.04498841 -0.05070846 -0.03926835
```

Estimating the Mean Regression Model Using SGD

We now estimate the same model using SGD.

```r
 mincer_sgd = sgdi_lm(y ~ edu + exp + exp2)
 print(mincer_sgd)
#> Call:
#> sgdi_lm(formula = y ~ edu + exp + exp2)
#>
#> Coefficients:
#>             Coefficient    CI.Lower    CI.Upper
#> (Intercept)  0.58714627  0.51899447  0.65529806
#> edu          0.12651235  0.12290359  0.13012112
#> exp          0.03152331  0.02788511  0.03516150
#> exp2        -0.04601193 -0.05566846 -0.03635539
#>
#> Significance Level: 95 %
```

It can be seen that the estimation results are similar between two methods. There is a different command that only computes the estimates but not confidence intervals.

```r
 mincer_sgd = sgd_lm(y ~ edu + exp + exp2)
 print(mincer_sgd)
#> Call:
#> sgd_lm(formula = y ~ edu + exp + exp2)
#>
#> Coefficients:
#>             Coefficient
#> (Intercept)  0.58621823
#> edu          0.12658176
#> exp          0.03152287
#> exp2        -0.04599148
```

We compare the execution times between two versions and find that there is not much difference in this simple example. By construction, it takes more time to conduct inference via `sgdi_lm`.

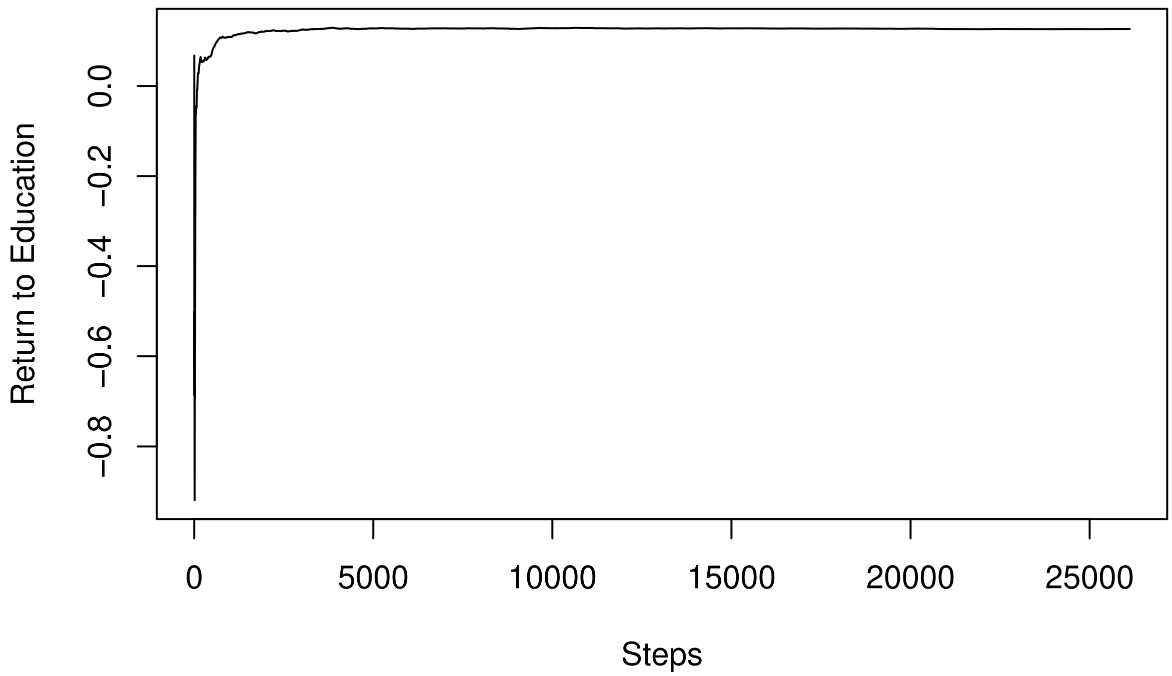
```r
library(microbenchmark)
res <- microbenchmark(sgd_lm(y ~ edu + exp + exp2),
                      sgdi_lm(y ~ edu + exp + exp2),
                      times=100L)
print(res)
#> Unit: milliseconds
#>                           expr      min       lq     mean   median       uq       max neval
#>   sgd_lm(y ~ edu + exp + exp2) 3.557775 3.770770 5.861999 3.857219 4.289072 111.11102   100
#>  sgdi_lm(y ~ edu + exp + exp2) 4.268018 4.537244 5.544736 4.617830 5.028220  11.06135   100
```

To plot the SGD path, we first construct a SGD path for the return to education coefficients.

```r
mincer_sgd_path = sgdi_lm(y ~ edu + exp + exp2, path = TRUE, path_index = 2)
```

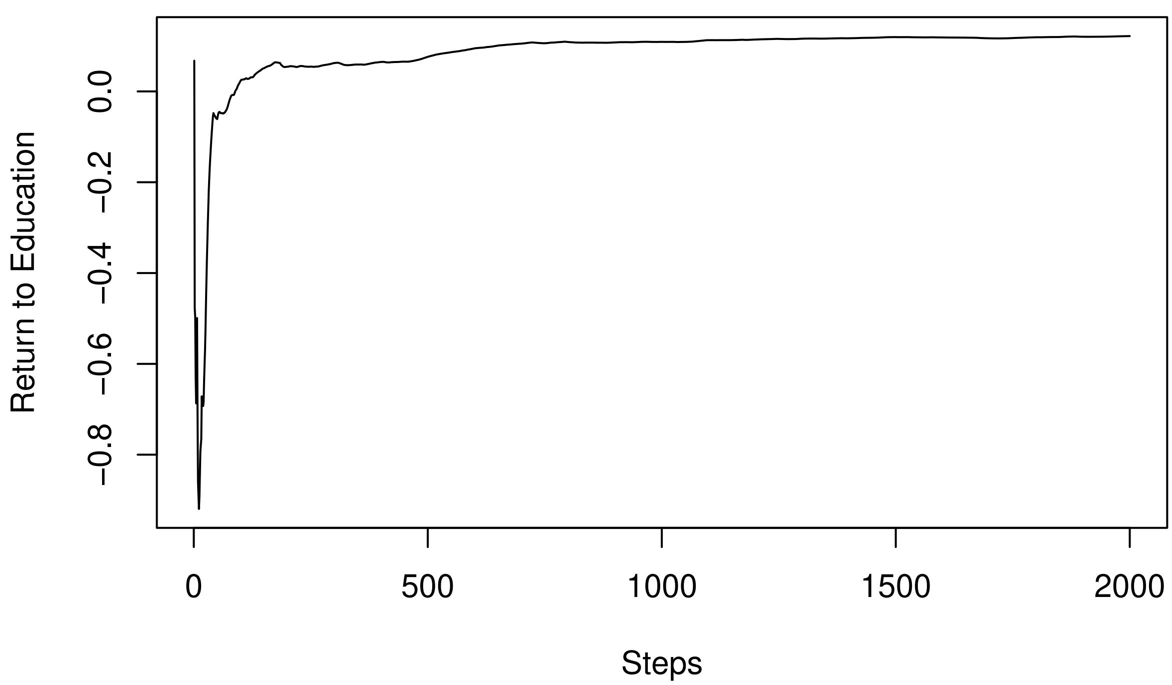
Then, we can plot the SGD path.

```r
plot(mincer_sgd_path$path_coefficients, ylab="Return to Education", xlab="Steps", type="l")
```



To observe the initial paths, we now truncate the paths up to 2,000.

```r
plot(mincer_sgd_path$path_coefficients[1:2000], ylab="Return to Education", xlab="Steps", type="l")
```




```r
print(c("2000th step", mincer_sgd_path$path_coefficients[2000]))
#> [1] "2000th step"       "0.121832196962998"
```

```r
print(c("Final Estimate", mincer_sgd_path$coefficients[2]))
#> [1] "Final Estimate"    "0.126481851251926"
```

It can be seen that the SGD path almost converged only after the 2,000 steps, less than 10% of the sample size.

Estimating the Quantile Regression Model Using S-subGD

We now estimate a quantile regression version of the Mincer equation.

```r
mincer_sgd = sgdi_qr(y ~ edu + exp + exp2)
print(mincer_sgd)
#> Call:
#> sgdi_qr(formula = y ~ edu + exp + exp2)
#>
#> Coefficients:
#>              Coefficient    CI.Lower    CI.Upper
#> (Intercept)   0.38554518  0.34997815  0.42111221
#> edu           0.14179972  0.13871229  0.14488716
#> exp           0.03070496  0.02817208  0.03323784
#> exp2         -0.04446399 -0.04992280 -0.03900519
#>
#> Significance Level: 95 %
```

The default quantile level is 0.5, as seen below.

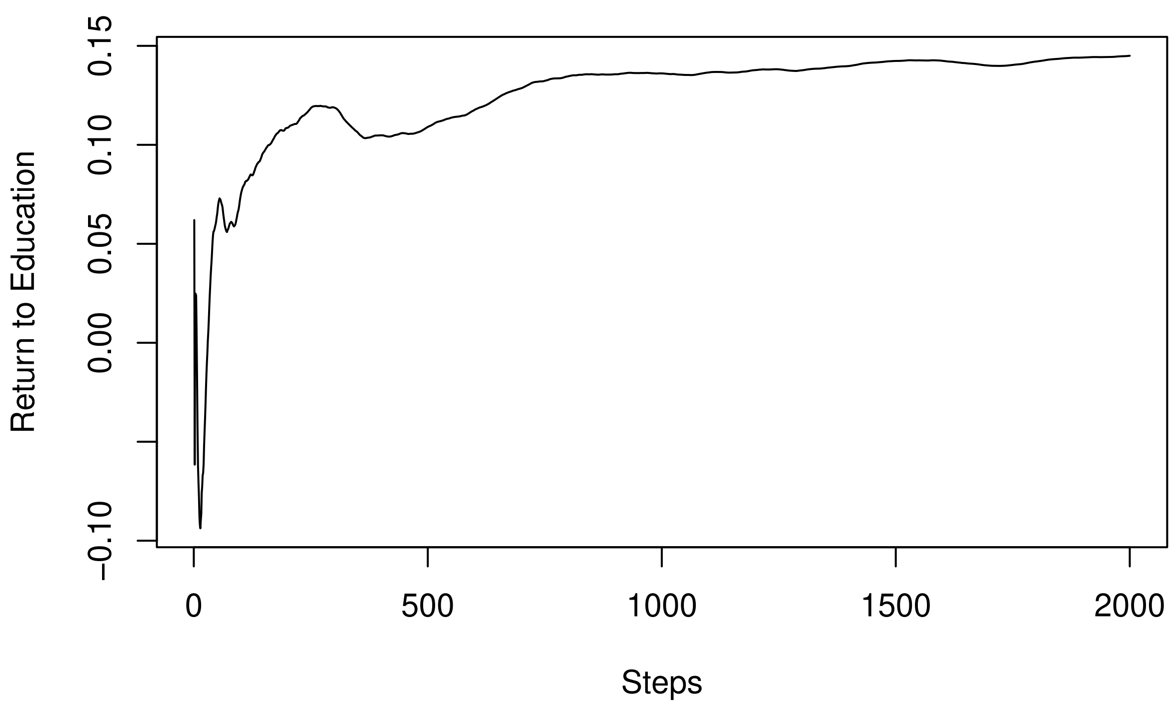
```r
mincer_sgd = sgdi_qr(y ~ edu + exp + exp2)
print(mincer_sgd)
#> Call:
#> sgdi_qr(formula = y ~ edu + exp + exp2)
#>
#> Coefficients:
#>              Coefficient    CI.Lower    CI.Upper
#> (Intercept)   0.38950568  0.35471614  0.42429523
#> edu           0.14093267  0.13888052  0.14298482
#> exp           0.03162466  0.02957803  0.03367129
#> exp2         -0.04682869 -0.05391537 -0.03974201
#>
#> Significance Level: 95 %
mincer_sgd_median = sgdi_qr(y ~ edu + exp + exp2, qt=0.5)
print(mincer_sgd_median)
#> Call:
#> sgdi_qr(formula = y ~ edu + exp + exp2, qt = 0.5)
#>
#> Coefficients:
#>              Coefficient    CI.Lower    CI.Upper
#> (Intercept)   0.39411891  0.35611944  0.43211839
#> edu           0.14093688  0.13831053  0.14356323
#> exp           0.03093505  0.02835594  0.03351416
#> exp2         -0.04491385 -0.05045572 -0.03937199
#>
#> Significance Level: 95 %
```

We now consider alternative quantile levels.

```r
mincer_sgd_p10 = sgdi_qr(y ~ edu + exp + exp2, qt=0.1)
print(mincer_sgd_p10)
#> Call:
#> sgdi_qr(formula = y ~ edu + exp + exp2, qt = 0.1)
#>
#> Coefficients:
#>              Coefficient    CI.Lower    CI.Upper
#> (Intercept) -0.25196933 -0.31035456 -0.19358410
#> edu          0.13065520  0.12507978  0.13623062
#> exp          0.03324245  0.02421412  0.04227079
#> exp2        -0.05336690 -0.07399920 -0.03273460
#>
#> Significance Level: 95 %
mincer_sgd_p90 = sgdi_qr(y ~ edu + exp + exp2, qt=0.9)
print(mincer_sgd_p90)
#> Call:
#> sgdi_qr(formula = y ~ edu + exp + exp2, qt = 0.9)
#>
#> Coefficients:
#>               Coefficient    CI.Lower    CI.Upper
#> (Intercept)  1.568552430  1.45559730 1.681507562
#> edu          0.114339739  0.10430657 0.124372907
#> exp          0.015915492  0.01069968 0.021131310
#> exp2        -0.004314836 -0.01861263 0.009982955
#>
#> Significance Level: 95 %
```

As before, we can plot the SGD path.

```r
mincer_sgd_path = sgdi_qr(y ~ edu + exp + exp2, path = TRUE, path_index = 2)
plot(mincer_sgd_path$path_coefficients[1:2000], ylab="Return to Education", xlab="Steps", type="l")
```



```r
print(c("2000th step", mincer_sgd_path$path_coefficients[2000]))
#> [1] "2000th step"       "0.144993066450143"
print(c("Final Estimate", mincer_sgd_path$coefficients[2]))
#> [1] "Final Estimate"    "0.141593421862818"
```