Accompanying code for "The Proximal Robbins–Monro Method"

This document describes the code accompanying "The Proximal Robbins–Monro Method" paper. Three R script files are necessary: `Fig1_2.R` and `Fig_3.R`, and `lib.R`, and they can be used to reproduce all the figures in the paper.

Figure 1 (`Fig1_2.R`)

The three stochastic procedures for Figures 1 and 2 are implemented through the `sgd, isgd, sfp` functions in `Fig1_2.R` for either the normal model (Section 5.1 of main paper) or the Poisson model (Section 5.2). For instance, to run SGD on normal data using a learning rate $\gamma_1 = 10$ we can run the following code:

```
> source("Fig1_2.R")
> Data = gen_data("normal")
> out = sgd(Data, gamma1=10, verbose=FALSE)
> out
            mean_level      max       avg      last
(Intercept)  -1.877628 14.72283 -1.866649 -2.215374
```

The results show log(MSE) for the SGD iterates; `max` refers to the maximum log(MSE) over all iterates; `avg` refers to their average value; `last` refers to the log(MSE) of the last SGD iterate.

To run ISGD in the same setting:

```
> out = isgd(Data, gamma1=10, verbose=FALSE)
> out
            mean_level       max       avg      last
(Intercept)  -1.918245 0.6037228 -1.903004 -1.994473
```

To run the stochastic fixed point algorithm (SFP) of Eq. (16) in the same setting:

```
> out = sfp(Data, gamma1=10, verbose=FALSE)
> out
            mean_level        max       avg      last
(Intercept)  -1.533026 -0.9272963 -1.506299 -1.772039
```

We can see that all methods are comparable with respect to average log(MSE). However, standard SGD has clearly the largest max log(MSE), indicating that it is less robust than the other methods.

To generate the full version of Figure 1 (in the paper) is computationally intensive. So, we can generate a simplified version of Figure 1 through the following code:

```
> run_experiment(num_gamma=5, nreps=10)  # this saves a .rda file locally.
> g1 = results_boxplot(whatval="mean_level")
> g2 = results_boxplot(whatval="max")
> multiplot(g1, g2)
```

In a conventional laptop this will take between 1-2 minutes of wall clock time. The output is shown in the figure below. To fully reproduce the experiment in Figure 1 of the main paper we just need to scale up the above code:

```
> run_experiment(num_gamma=20, nreps=50)
```

The execution could be done in a super computing cluster, since the experiments are independent across the learning rates γ_1.

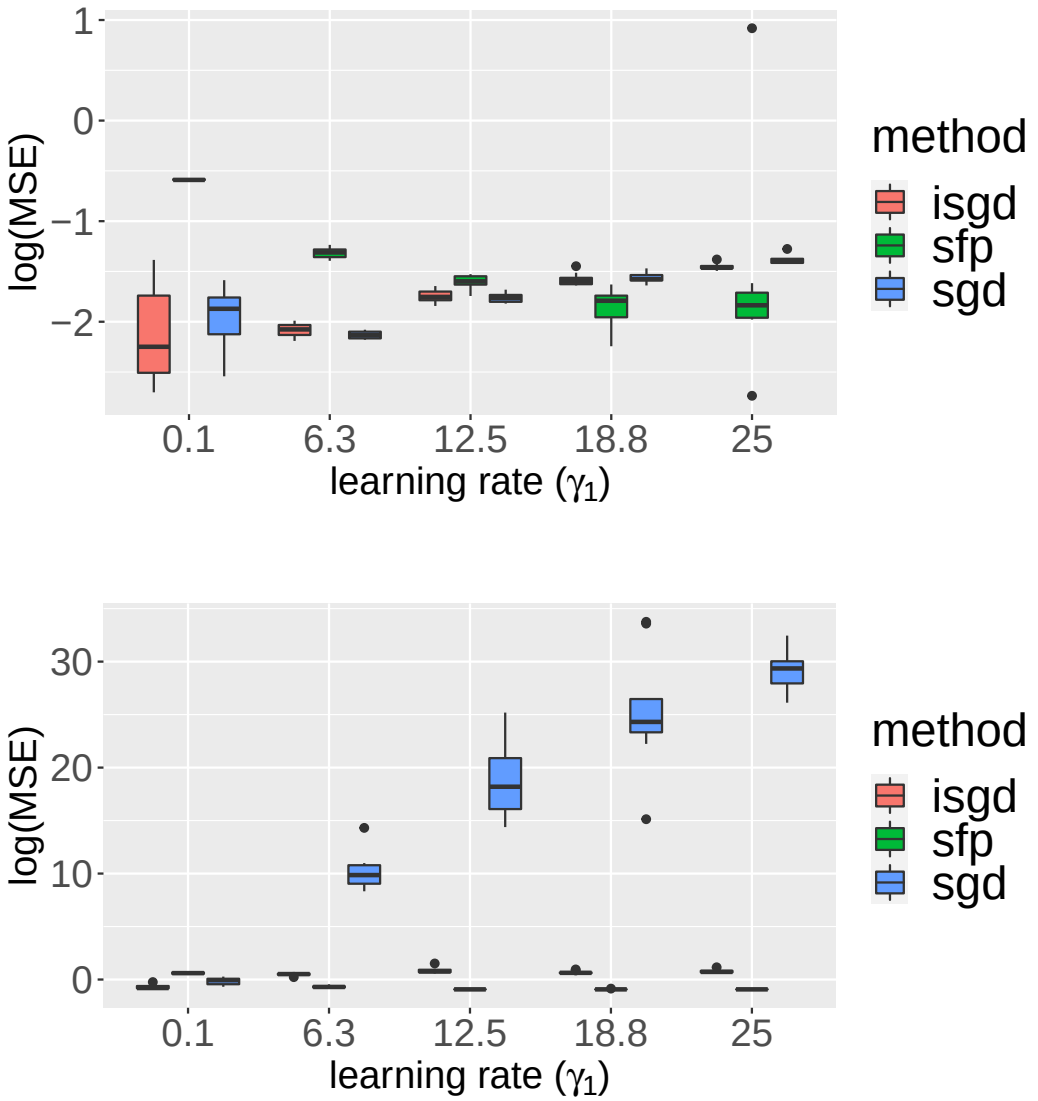

Figure 1: Output from simplified experiment with normal data, as described above. The resulting figure is a simplified version of Figure 1 in the main paper.

Figure 2 (Fig1_2.R)

To run SGD on Poisson data using a learning rate $\gamma_1 = 1$ we can run the following code:

```
> Data = gen_data("poisson")
> out = sgd(Data, gamma1=1, verbose=FALSE)
> out
            mean_level       max        avg       last
(Intercept) -0.4701483 0.4659749 -0.4706964 -0.4621388
```

To run ISGD in the same setting:

```
> out = isgd(Data, gamma1=1, verbose=FALSE)
> out
            mean_level        max       avg      last
(Intercept)  -3.031638 -0.2779868 -2.899444 -3.349216
```

To run SFP in the same setting:

```
> out = sfp(Data, gamma1=1, verbose=FALSE)
> out
            mean_level        max        avg       last
(Intercept) -0.4735927 -0.2147034 -0.4832646 -0.4648346
```

We can see that all methods are comparable with respect to average log(MSE), with ISGD being slightly better overall. However, standard SGD will perform much worse as we increase the learning rate.

To get a simplified version of Figure 2 (in the paper) we can execute the following code:

```
> run_experiment(model="poisson", num_gamma=5, max_gamma1=5, nreps=10)
> g1 = results_boxplot(model="poisson", whatval="mean_level")
> g2 = results_boxplot(model="poisson", whatval="max")
> multiplot(g1, g2)
```

In a conventional laptop this will take between 3-4 minutes of wall clock time. The output is shown in the figure below. To fully reproduce the experiment in Figure 2 of the main paper we just need to scale up the above code:

```
> run_experiment(model="poisson", num_gamma=20, max_gamma1=5, nreps=50)
```

As in the normal model, this could be done in a super computing cluster.

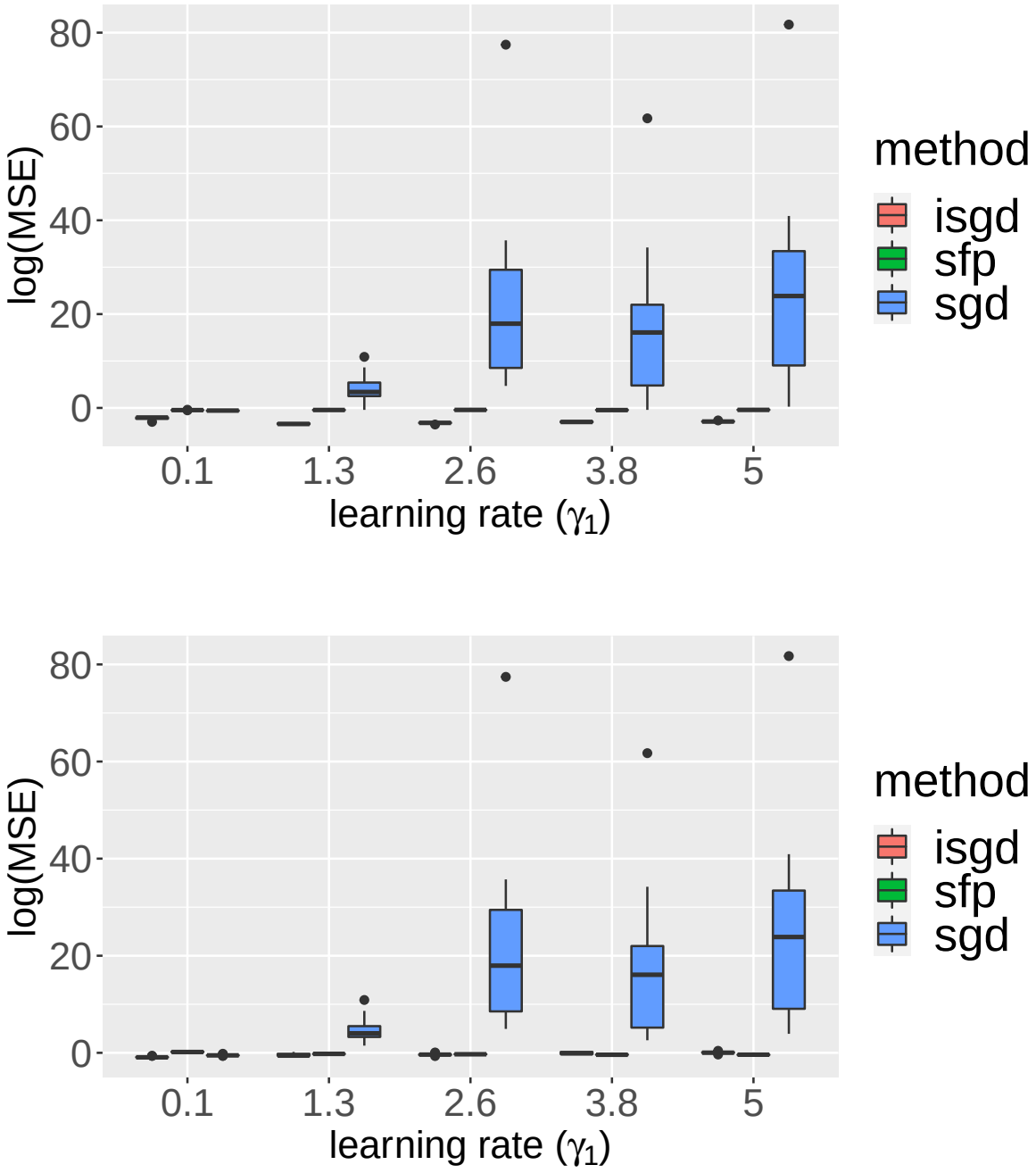

Figure 2: Output from simplified experiment with Poisson data, as described above. The results are similar to Figure 2 of main paper.

Figure 3

To reproduce Figure 3 we first have to generate the results through the following command:

```
> total_experiment(nreps=100)
```

The total execution time for the entire experiment is about 20 minutes of wall clock time. The results will be saved in a file "`QuantileRegression.rda`". The results can then be loaded and plotted through the command:

```
> gen_Fig3()
```

This will reproduce Figure 3 in the main paper.