STSCI 6520 Final Report

Minjie Jia, Joia Zhang

Fall 2023

1 Introduction

Online learning (OL) refers to the situation where data is available in a sequential order and is used to update the best predictor for future data at each step. Compared to batch learning, where an algorithm is trained on the entire data set at once, OL is used when it is computationally infeasible to train over the entire dataset, when there's a need to adapt to new patterns of data in the future, or when the data itself is generated as a function of time, e.g., stock price prediction.

Online Learning is often designing and analyzing algorithms to minimize:

Regret over a sequence of loss functions with respect to an arbitrary competitor $u \in V$

$$\text{Regret}_T(u) := \sum_{t=1}^{T} l_t\left(x_t\right) - \sum_{t=1}^{T} l_t(u)$$

Here, l_t is the loss function at time t. The loss functions are always restricted to e.g. convex / lipschiz convex for convergence analysis.

2 Existing Algorithms

Definitions:

$l_t(x)$: Realization of stochastic objective at time step t. For simplicity l_t.

$g_t(x)$: The gradient of $l_t(x)$ with respect to x, formally $\nabla_x l_t(x)$. For simplicity, g_t.

x_t : Parameters at time step t.

G_t : The outer product of all previous subgradients, given by $\sum_{r=1}^{t} g_\tau g_\tau^\top$

2.1 Online Gradient Descent (OGD)

Instead of updating the estimates using the gradient of the full dataset iteratively, Online gradient descent (OGD) updates each time a new data point comes in. Formally, let $\nabla l_t\left(\beta_t\right)$ be the gradient of loss function, and η_t be

the learning rate for each iteration, then the OGD algorithm updates in the following way:

For $t = 1, 2, \ldots T$

$$\boxed{\beta_{t+1} = \beta_t - \eta_t \nabla l_t \left(\beta_t\right)}$$

Algorithm 1 OGD

Require: η : Stepsize ;
$l(\beta)$: Stochastic objective function (loss) ;
β_1 : Initial parameter vector;
for $t = 1$ `to` T **do**
`Evaluate` $l_t\left(\beta_t\right)$`;`
`Get and save` $g_t = \nabla_\beta l_t(\beta)$`;`
$\beta_{t+1} \leftarrow \beta_t - \eta g_t$
end for

For **convergence analysis**, assume that the norm of the gradients is bounded by L, that is $\|\boldsymbol{g}_t\|_2 \leq L$. Also assume a bounded diameter, we can upper bound $\|\boldsymbol{u} - \boldsymbol{x}_1\|$ by D. Then, we have $\eta^\star = \underset{\eta}{\operatorname{argmin}} \frac{D^2}{2\eta} + \frac{\eta L^2 T}{2} = \frac{D}{L\sqrt{T}}$ which gives a regret bound of $DL\sqrt{T} \implies$ **OGD has a regret that is sublinear in time**

2.2 Adaptive Gradient Descent (AdaGrad)

Adaptive Gradient Descent (AdaGrad) extends OGD by adaptively setting the step size. Specifically, define $G_t = \sum_{\tau=1}^{t} \nabla l_\tau \left(\beta_\tau\right) \nabla l_\tau \left(\beta_\tau\right)^\top$.

The update now becomes:

For $t = 1, 2, \ldots T$

$$\boxed{\beta_{t+1} = \beta_t - \eta \left(G_t\right)^{-\frac{1}{2}} g_t}$$

Here we used the diagonal of G_t to update the learning rate.

It can be seen that AdaGrad assigns different learning rates to different features. In particular, it assigns low learning rate to frequent occurring features, and high learning rate to infrequent features.

Algorithm 2 AdaGrad Iterative

Require: η : Stepsize
$l(\beta)$: Objective function (loss)
β_1 : Initial parameter vector

for $t = 1$ to T **do**
$g_t = \nabla_\beta l_t(\beta)$
$G_t \leftarrow \sum_{\tau=1}^{t} g_\tau g_\tau^\top$
$\beta_{t+1} \leftarrow \beta_t - \eta G_t^{-1/2} g_t$
end for

However, one problem with AdaGrad is that it assigns equal weights to recent and old gradients in the computation of G, and as time increases, the learning rate is rapidly diminishing.

2.3 Root Mean Square Propagation (RMSProp)

Update rule:

$$\boxed{\beta_t = \beta_{t-1} - \alpha \frac{\nabla L_t\left(\beta_{t-1}\right)}{\sqrt{R_t}}}$$

where

$$\boxed{R_t = \gamma R_{t-1} + (1-\gamma)\nabla L_t\left(\beta_{t-1}\right)^2}$$

RMSProp is similar to AdaGrad but with an exponential moving average controlled by $\gamma \in [0, 1)$. With smaller γ, there will be more emphasis on recent gradients.

2.4 Adaptive Moment Estimation (Adam)

Adam combines the advantages of both AdaGrad–works well with sparse gradients and RMSProp–works well in non-stationary settings.

Algorithm 3 Adam

Require: η : Stepsize
$l(\beta)$: Stochastic objective function (loss)
β_1 : Initial parameter vector
for $t = 1$ `to` T **do**
$M_t = \alpha_1 M_{t-1} + (1-\alpha_1)\nabla L_t(W_{t-1})$ (1st moment estimate)
$R_t = \alpha_2 R_{t-1} + (1-\alpha_2)\nabla L_t(W_{t-1})^2$ (2nd moment estimate)
$\hat{M}_t = M_t / \left(1-(\alpha_1)^t\right)$ (1st moment bias correction)
$\hat{R}_t = R_t / \left(1-(\alpha_2)^t\right)$ (2nd moment bias correction)
$\beta_t = \beta_{t-1} - \eta \frac{\hat{M}_t}{\sqrt{\hat{R}_t + \epsilon}}$ (Update)
end for

3 Experiments

3.1 Data Generation

We generated data for linear regression and logistic regression. Our data had $n = 1000$ observations and $p = 2000$ predictors. Our true β was a k-sparse vector where the $k = 0.05p$ nonzero values were sampled from $N(100, .01)$. Our data $X \sim N(0, 5)$. For regression, $\epsilon \sim N(0, 1)$ and $Y = X\beta + \epsilon$. For classification, $\pi = \frac{1}{1+\exp(-X\beta)}$ and $Y \sim Ber(\pi)$.

3.2 Performance Comparison

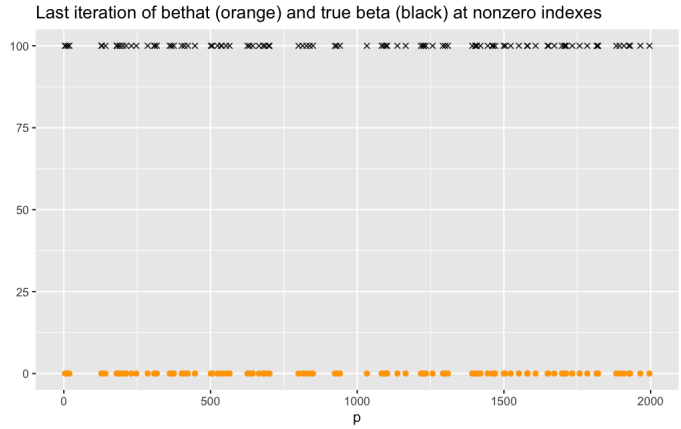
We used the following criterion to compare performance across OGD, Adagrad, and ADAM. First, we used prediction error given by $(\beta_t' x_t - y_t)^2$ for regression and misclassification rate $\frac{\sum_{i-1}^{t} \mathbf{1}\{\hat{Y}_t \neq Y_t\}}{t}$ for classification. Second, both regression and classification, we used estimation error given by $\|\beta_t - \beta\|_2$. Third, we evaluated runtime measured in seconds and finally, we observed convergence of prediction error across runtime.

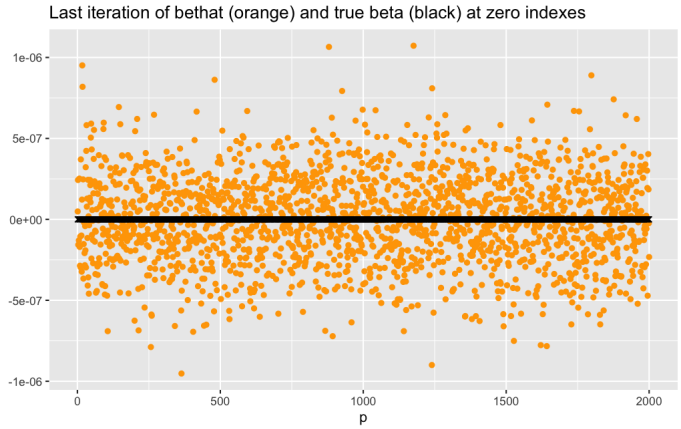
We will present some instances of our exploration for how the methods perform for different learning rates, weight initializations, and dimensions of the data. Here, our base case for comparison has learning rate of 10^{-7}, $\hat{\beta}_0 = 0$, and $n = 1000, p = 2000$.

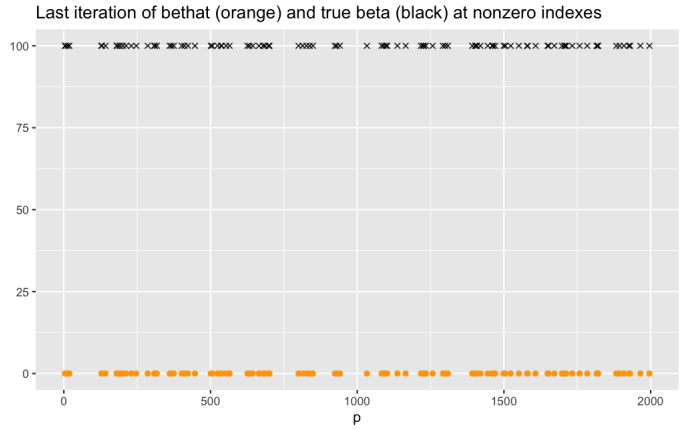
3.3 Behavior of $\hat{\beta}$

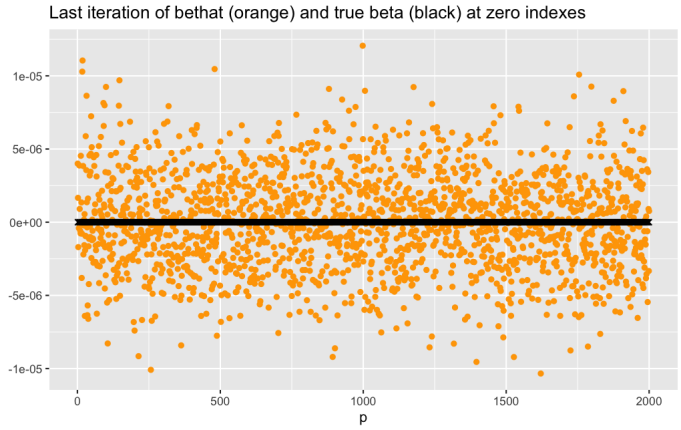
For both classification and regression, the $\hat{\beta}$'s tended to be zero, thereby failing to capture the k nonzero values of the true k-sparse vector β. The plots are for OGD, Adagrad, and Adam respectively.

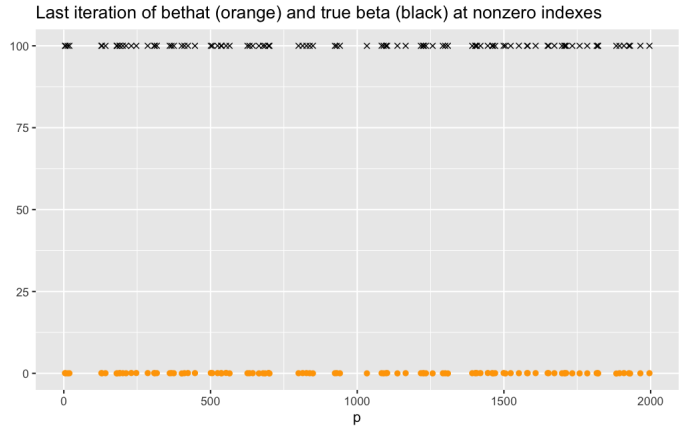
Regression:

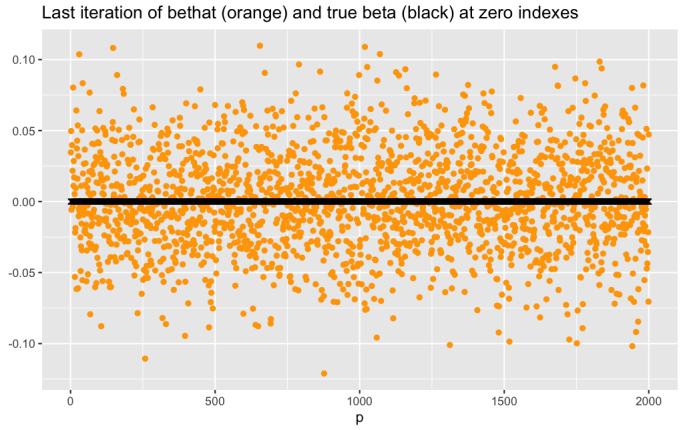




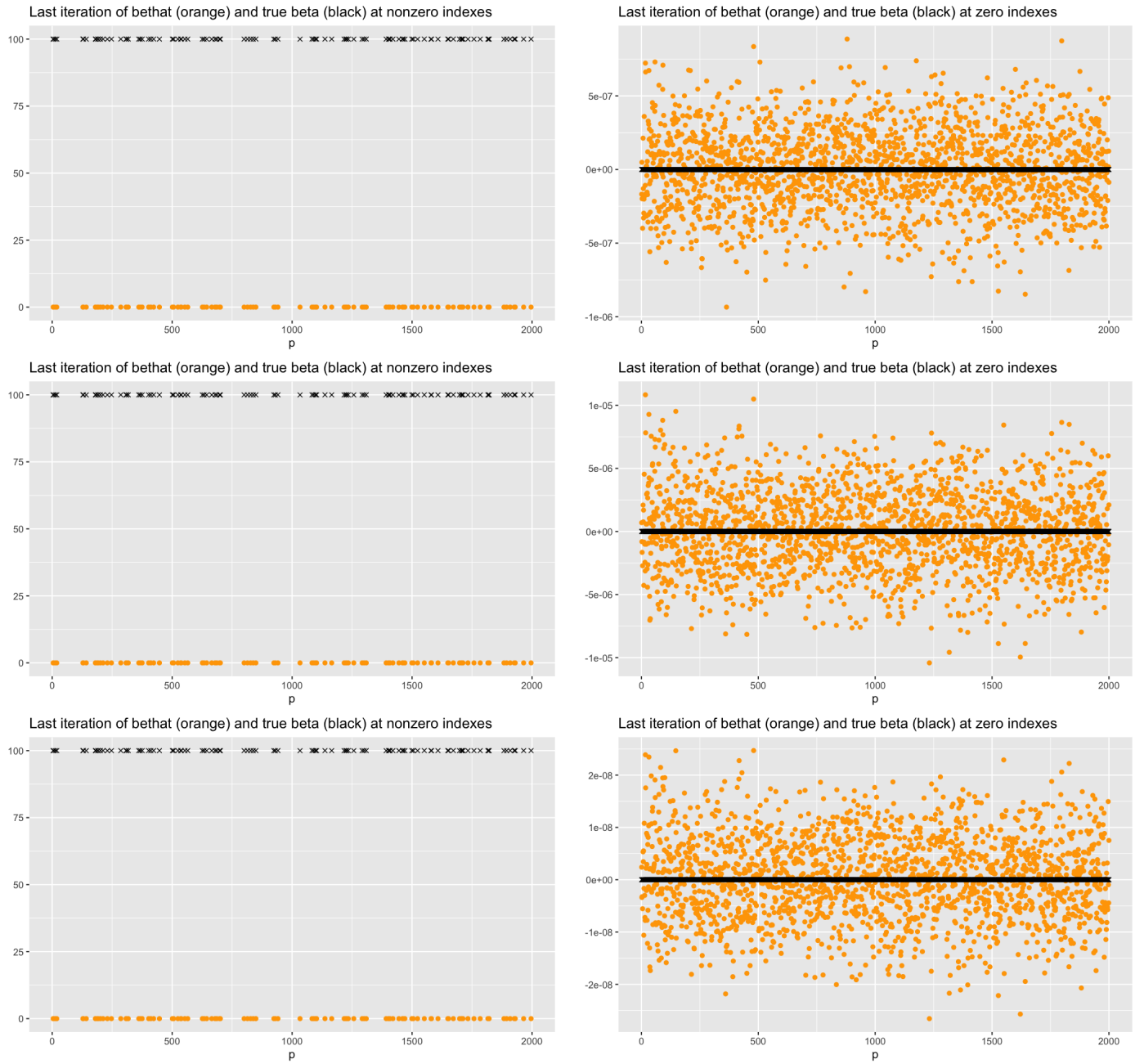








Classification:

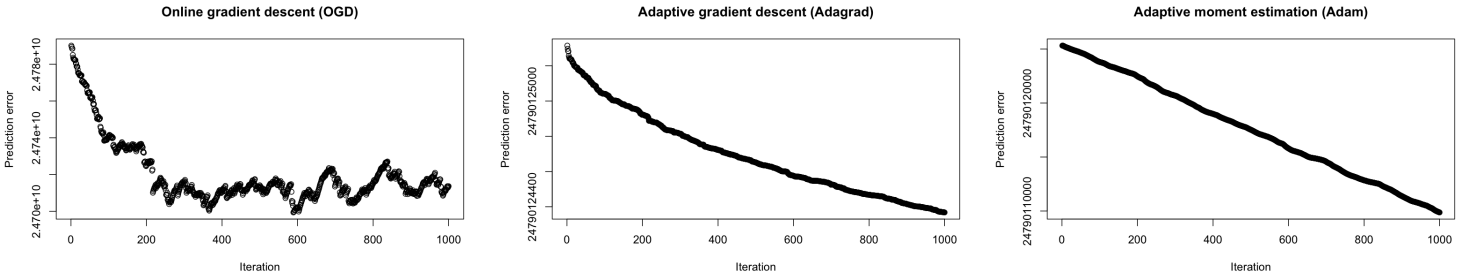


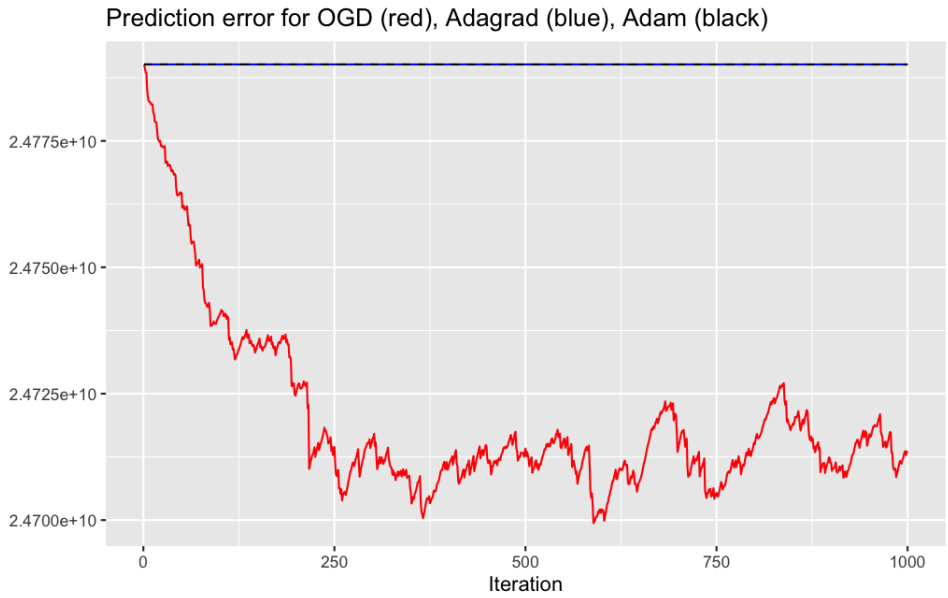
The observation holds for OGD, Adagrad, and Adam for both classification and regression.

3.4 Prediction Error

Regression:

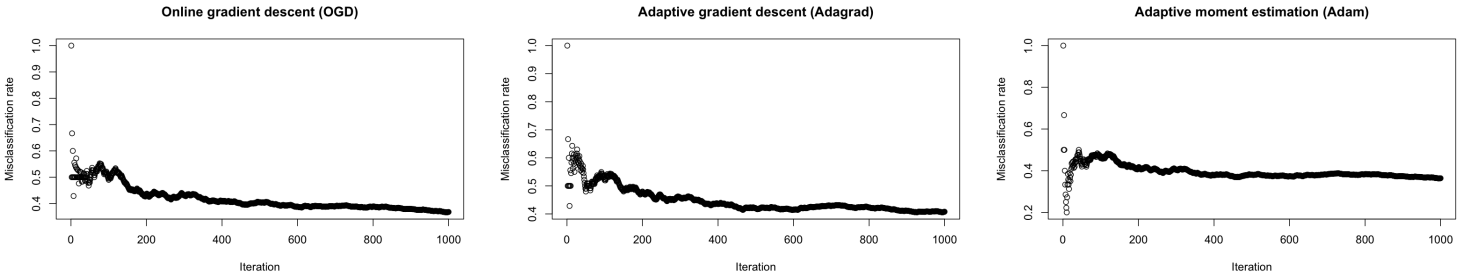
Prediction error decreases for the three methods, although at the current value of the hyperparameters, OGD converges better and faster than its competitors under the regression setting, which can be seen when the three methods are plotted together.

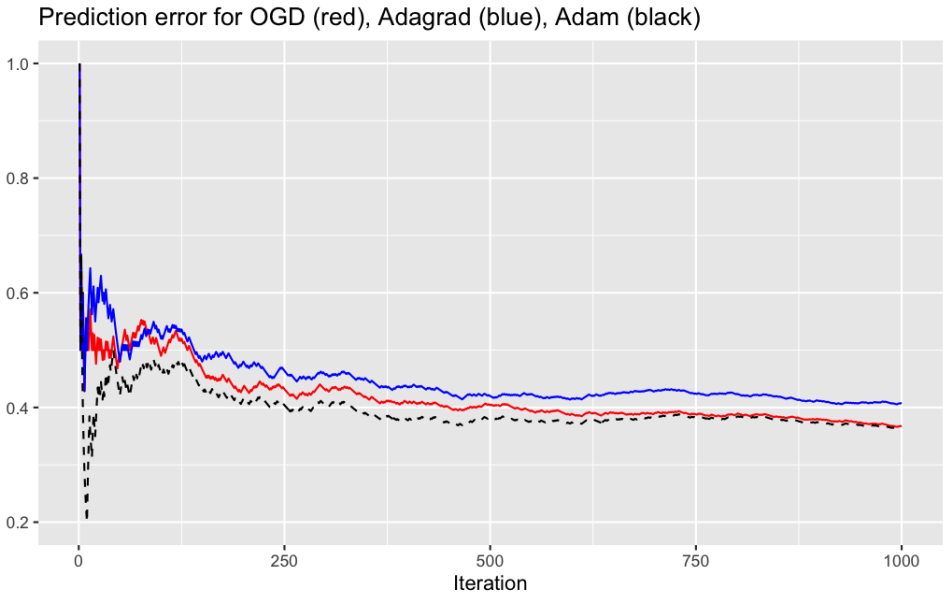




Classification:

Unlike the regression setting, the three methods perform comparably in classification. Adam performs the best with the lowest misclassification rate, although OGD and Adagrad are comparable.

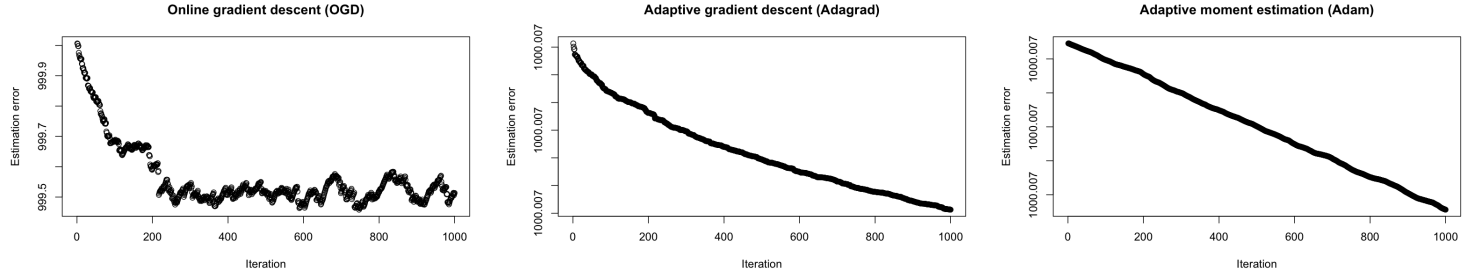


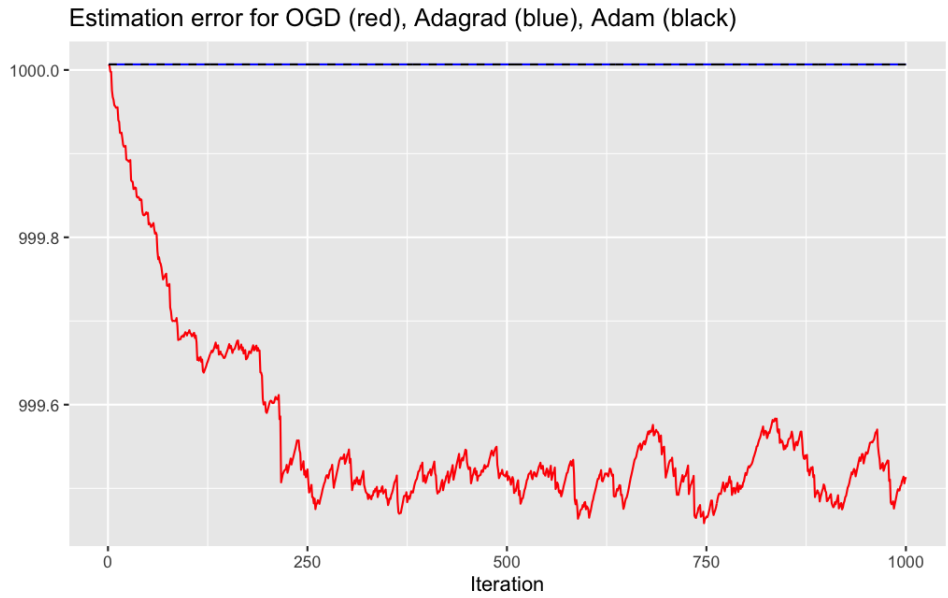


3.5 Estimation Error

Regression:

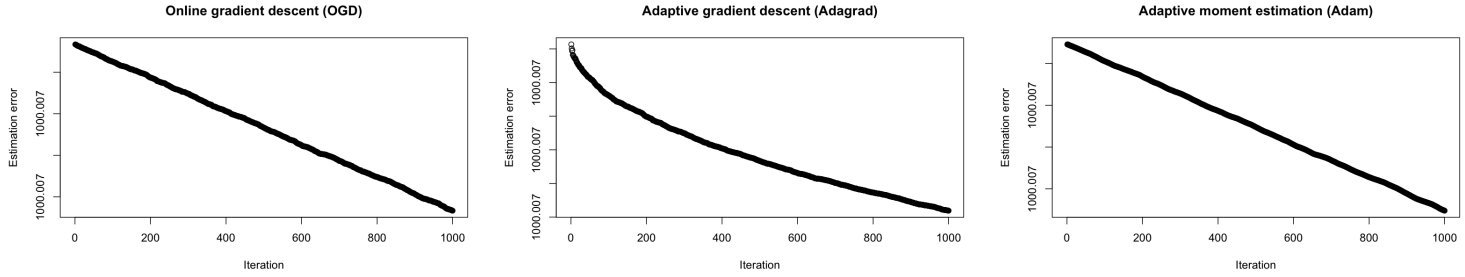
Similar observations can be made for estimation error in the regression setting, where OGD converges faster and more than the other optimization methods.

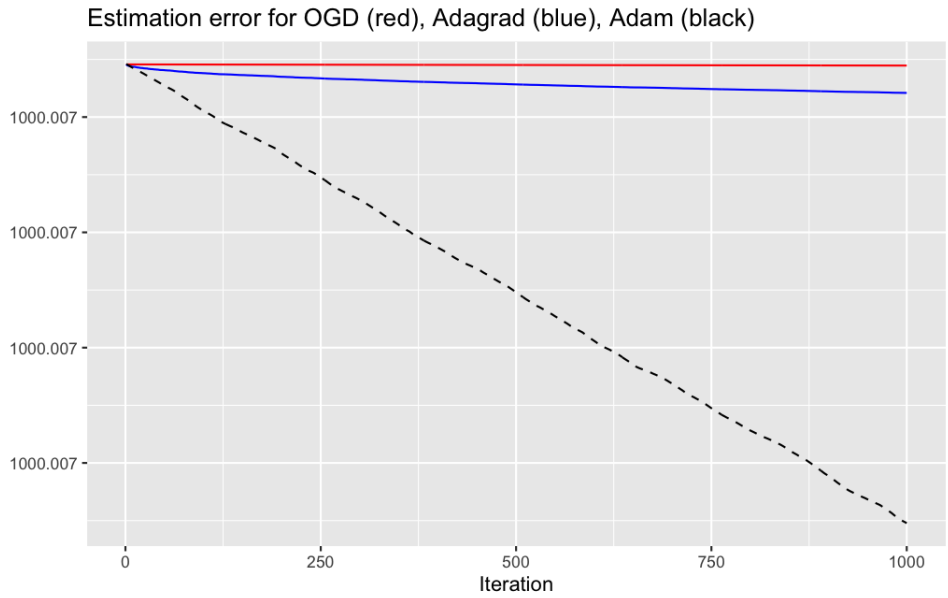




Classification:

Unlike regression, Adam has the best estimation error and converges more and faster than OGD and Adagrad.

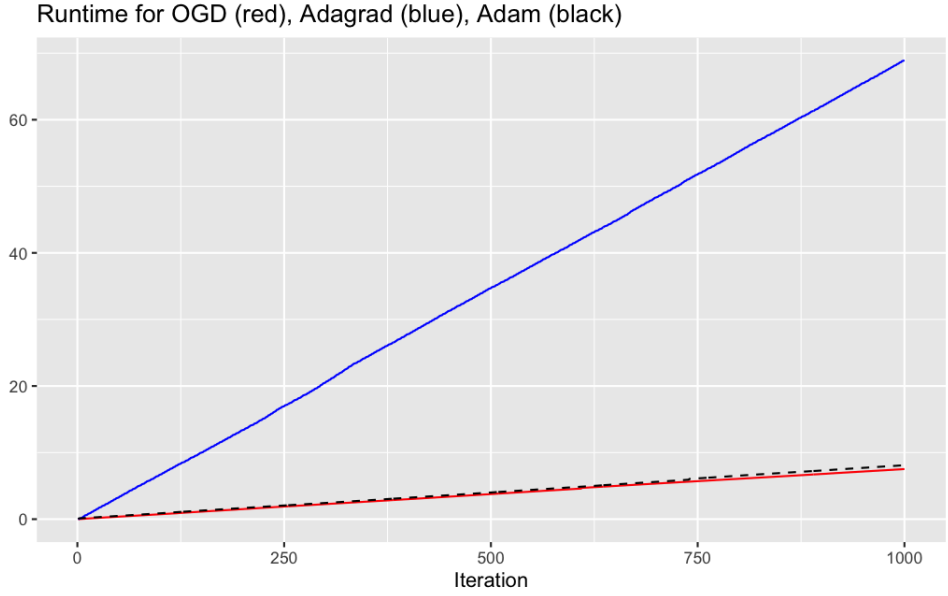




3.6 Runtime

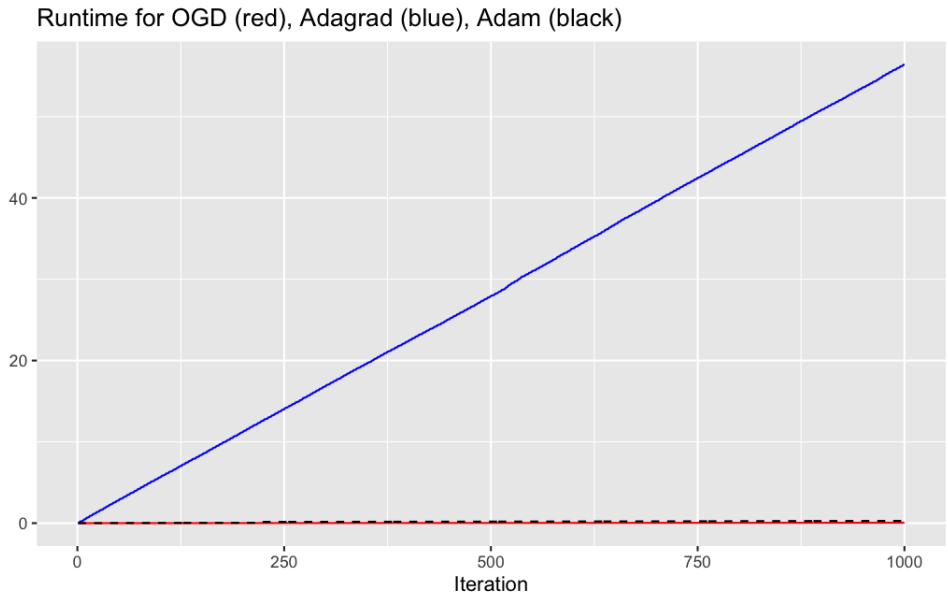
Regression:

Adagrad takes longer to compute whereas OGD and Adam have comparable runtimes.



Classification:

Runtimes are similar to those of regression where Adagrad requires longer runtime in R.

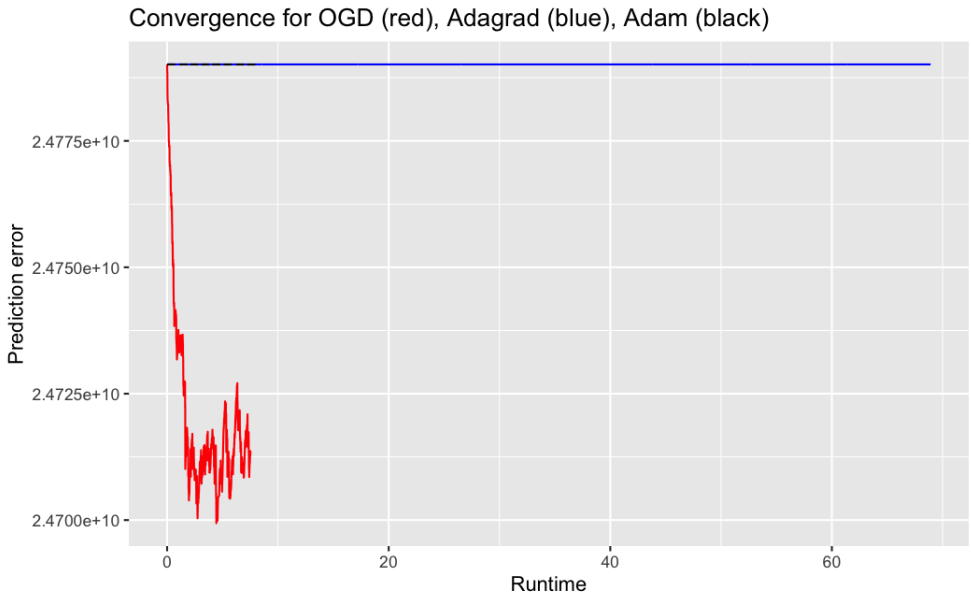


3.7 Convergence

Now we show prediction error against runtime to observe convergence or lack thereof for the three methods.

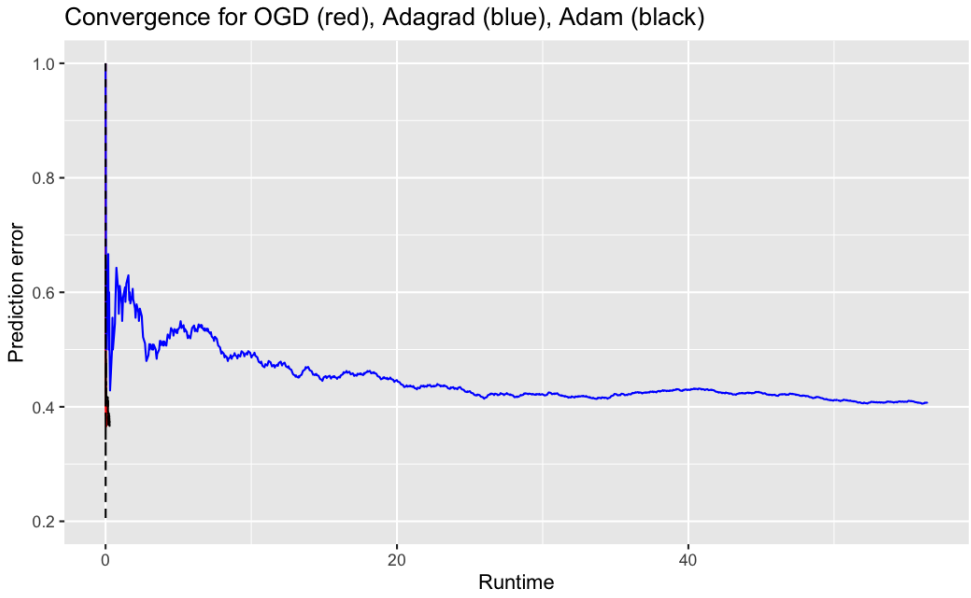
Regression:

Adagrad takes the longest to run while OGD converges the most for regression.



Classification:

For classification, Adagrad still takes the longest to run but Adam seems to be performing the best and converging the most.

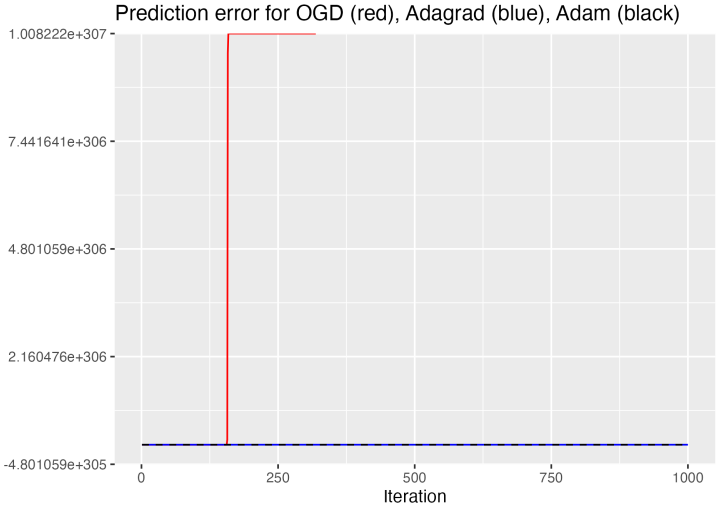


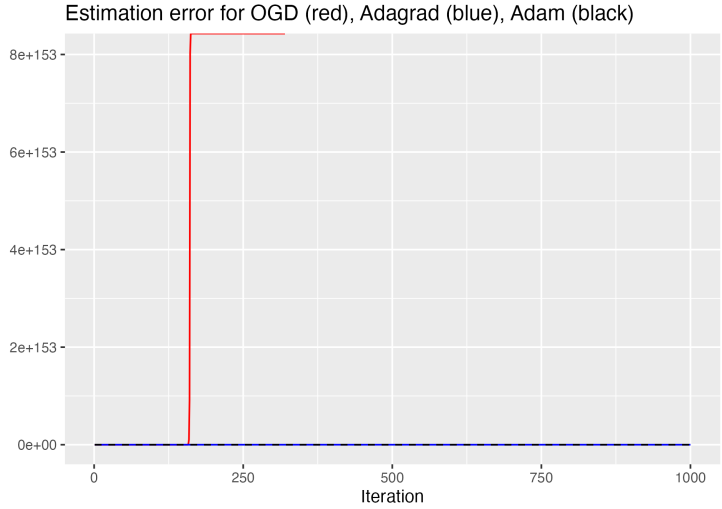
3.8 Learning Rate

Here, we keep the zero initialization and dimensions, but we use a larger learning rate of 10^{-4} instead of 10^{-7}.

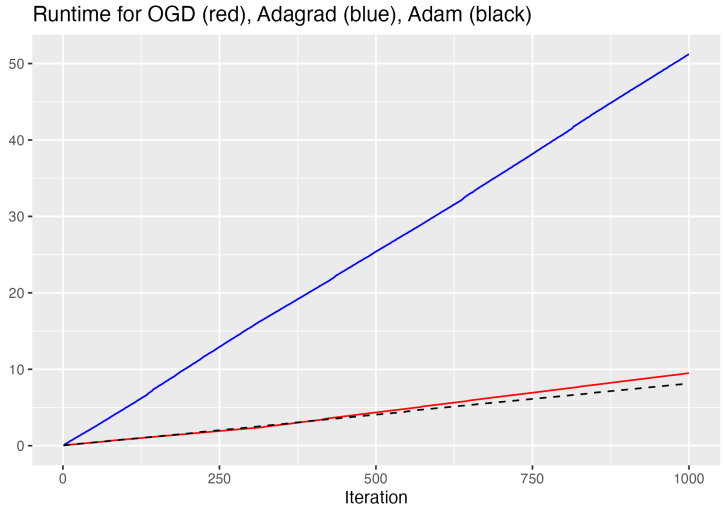
Regression:

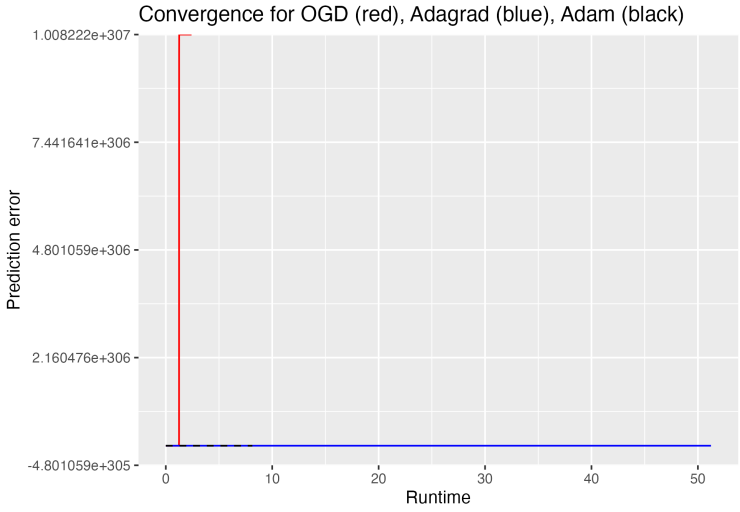
Now with the smaller learning rate, Adagrad and Adam both converge more than OGD which actually diverges, for both prediction and estimation error.





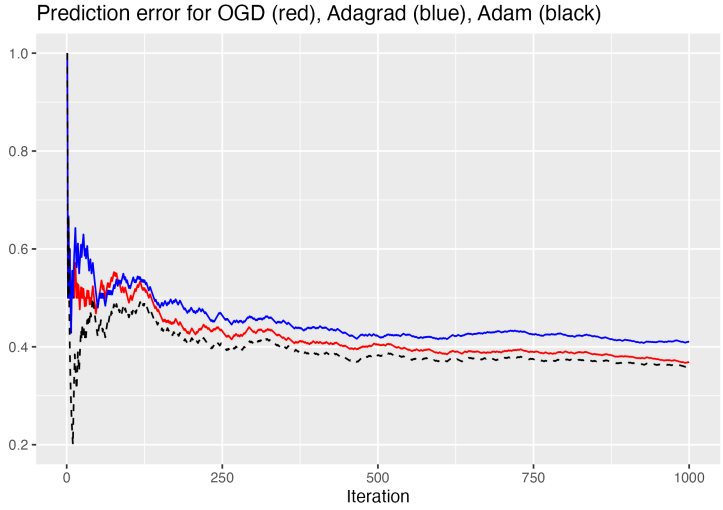
Runtime is similar to the original case of settings where Adagrad has the slowest runtime while OGD and Adam are comparable. We can see that with a larger learning rate and everything else constant, OGD no longer performs the best for regression and actually diverges. This further emphasizes the importance of learning rate selection, where OGD has a fixed learning rate and Adagrad and Adam have adaptive learning rates that are not entirely dependent on the global learning rate we increased here.

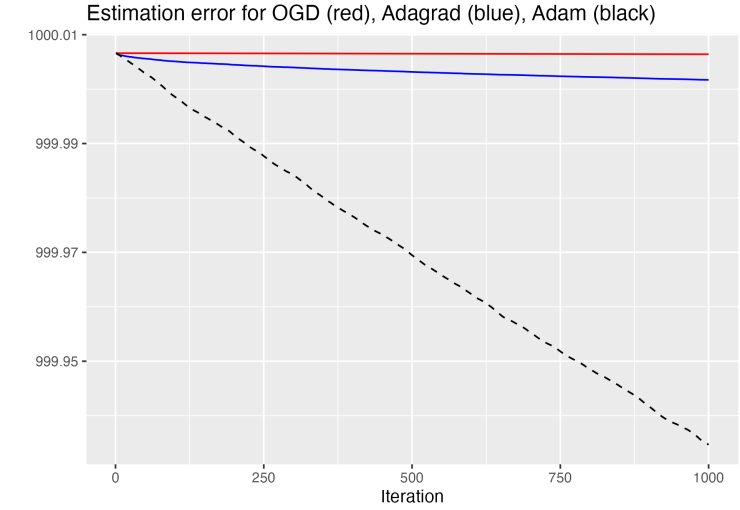


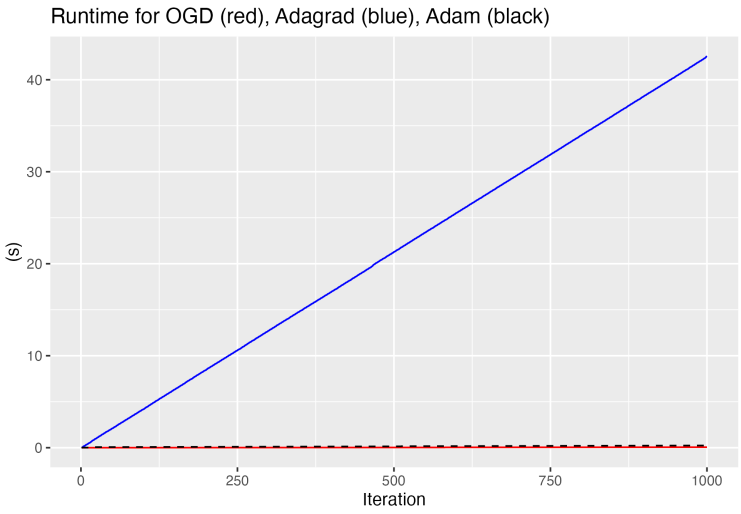


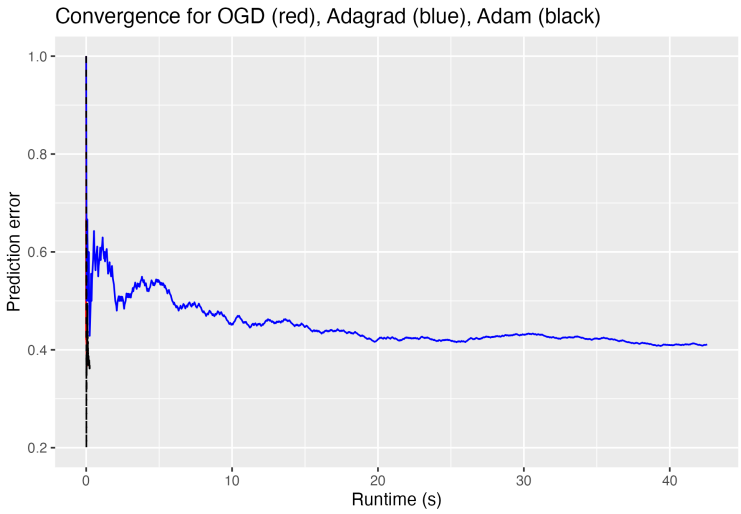
Classification:

In classification, with the higher learning rate OGD does not diverge like in regression for prediction error or estimation error. We still see that Adam is performing the best and that Adagrad has the slowest runtime.







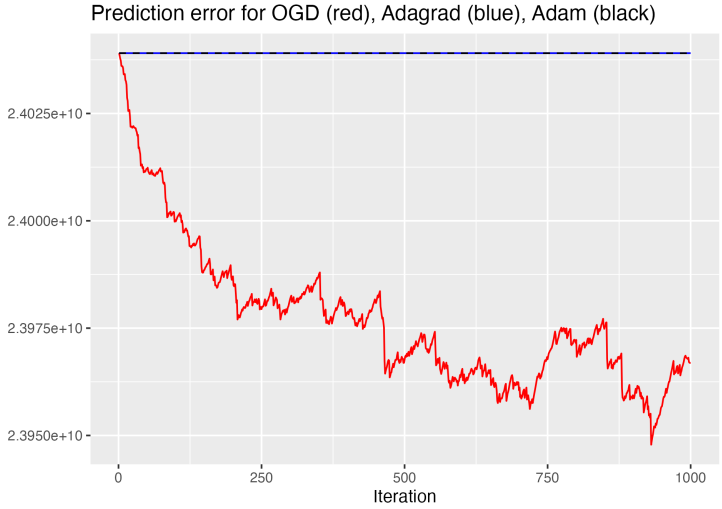


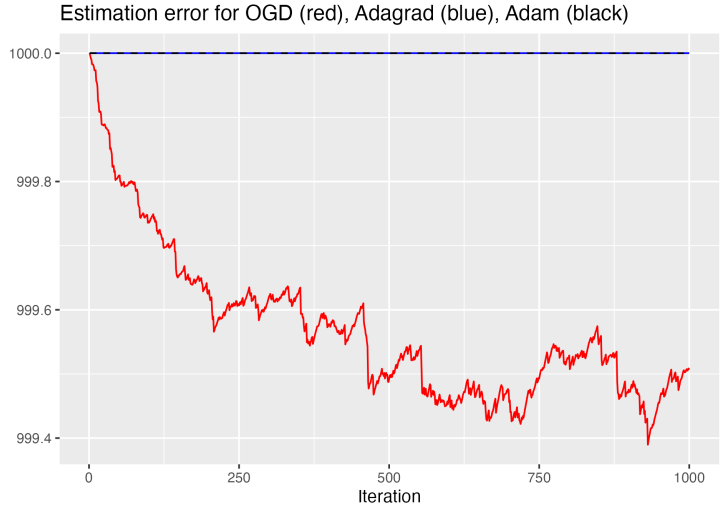
3.9 Weight Initialization

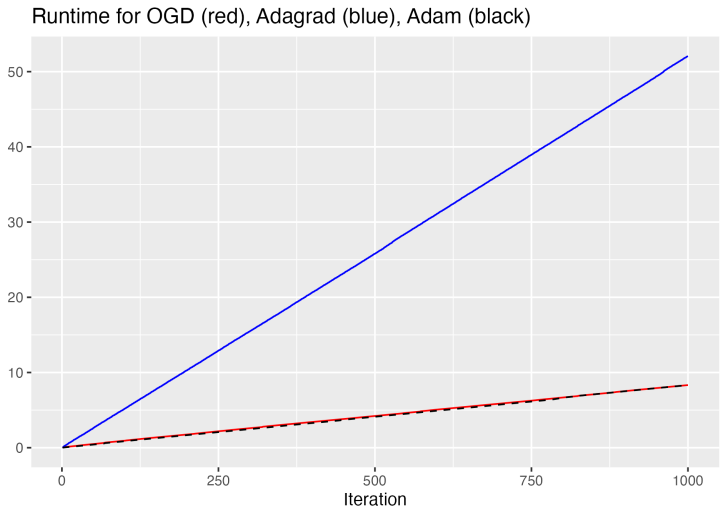
Here, we keep the same learning rate and dimensions, but we use uniform $U(0,1)$ initialization instead of the zero initialization.

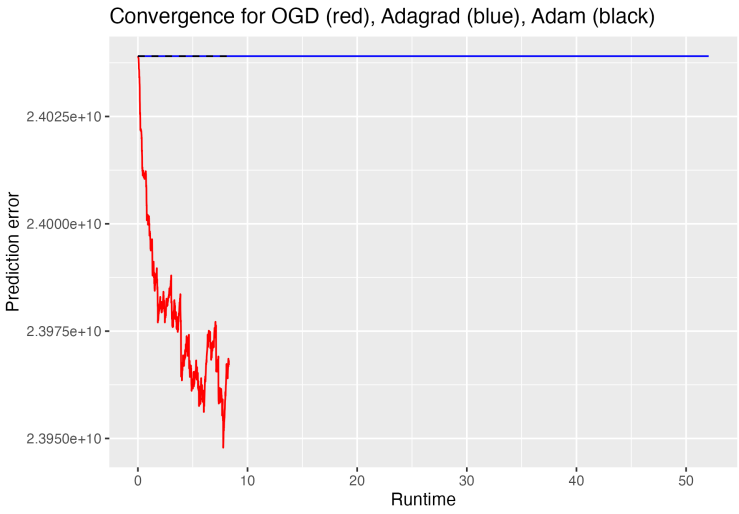
Regression:

Results are similar to the original case where OGD performs the best in regression while Adagrad and Adam are comparable.



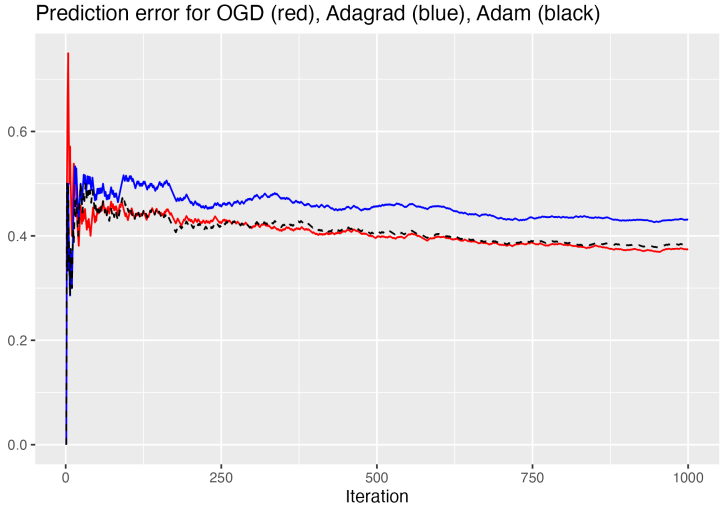


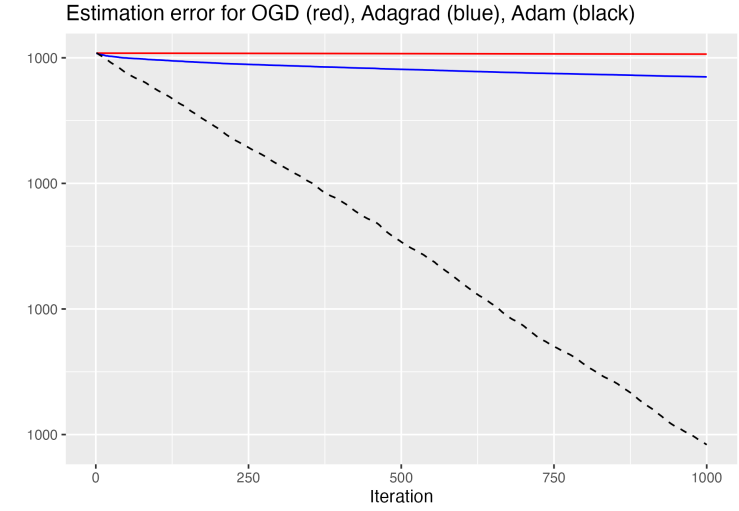


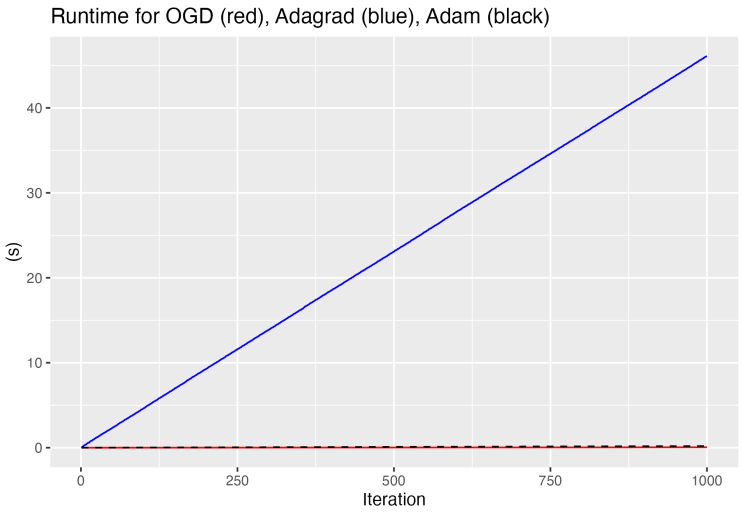


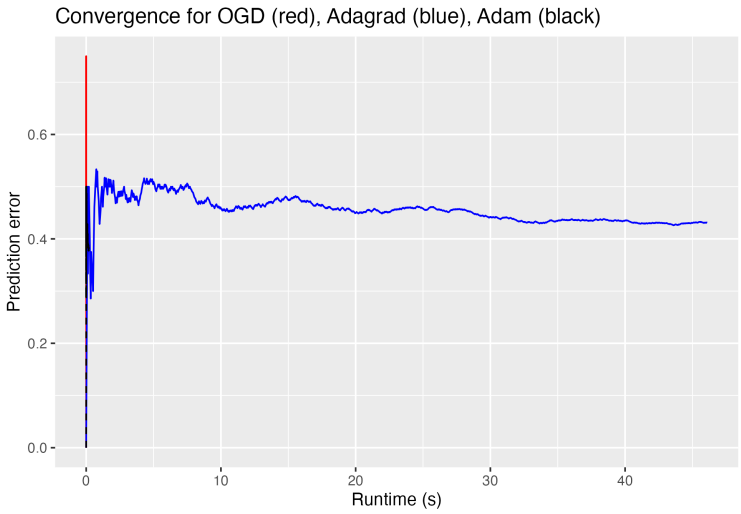
Classification:

With the different weight initialization, results for classification are similar to the original case as well, except possibly that OGD seems to be performing even more comparably to Adam for prediction error. Although for estimation error, Adam still performs the best.







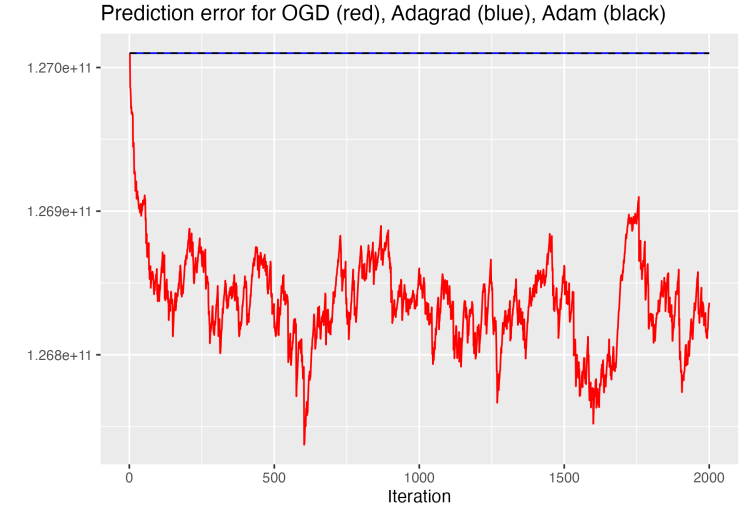


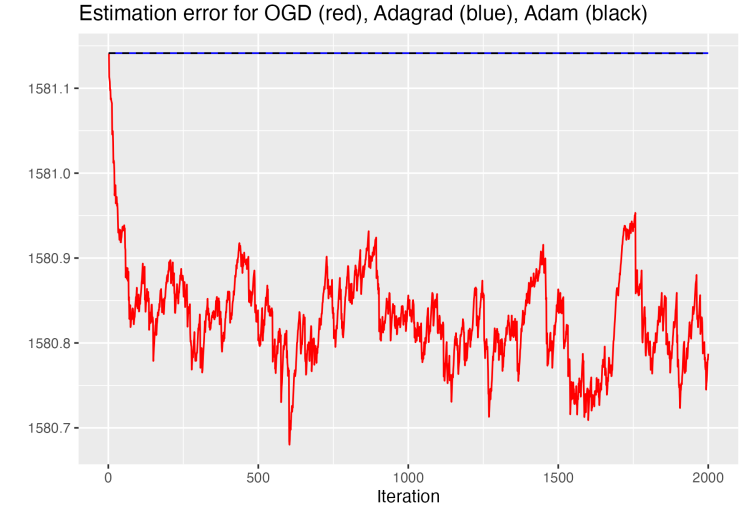
3.10 Dimensionality

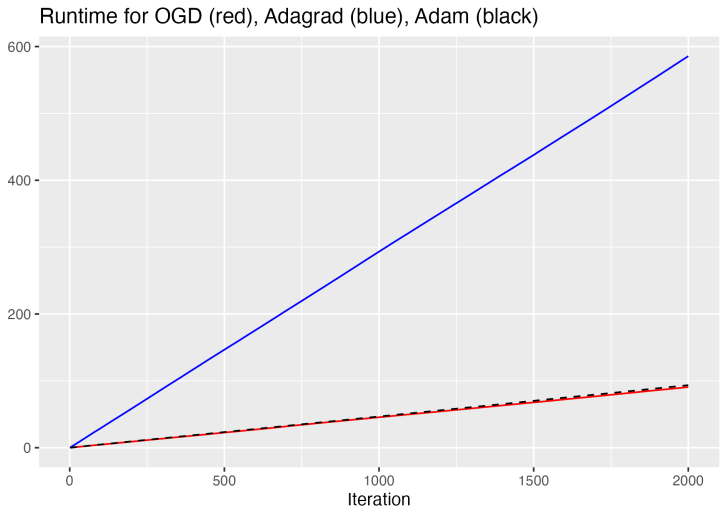
Here, we keep the learning rate and zero initialization and dimensions, but we use higher dimensions of the data of $n = 2000, p = 5000$.

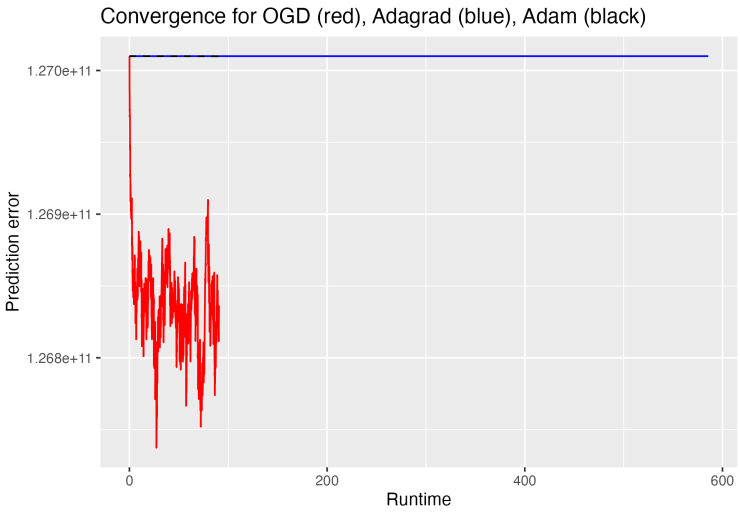
Regression:

Similar to the original case, OGD converges the most for both prediction and estimation error.



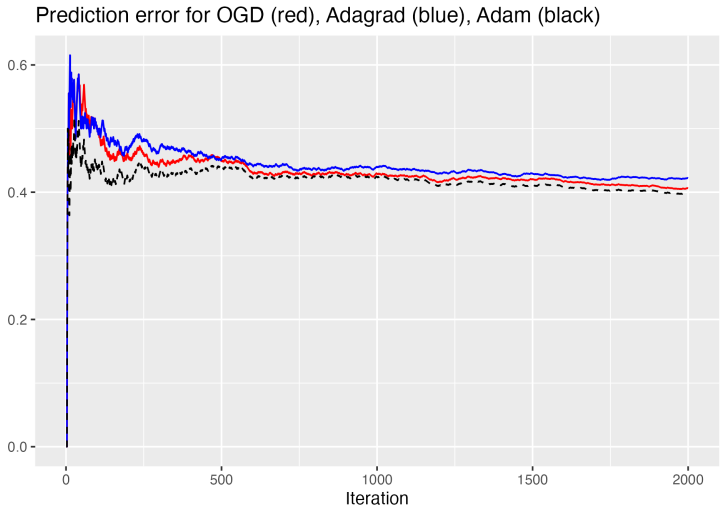


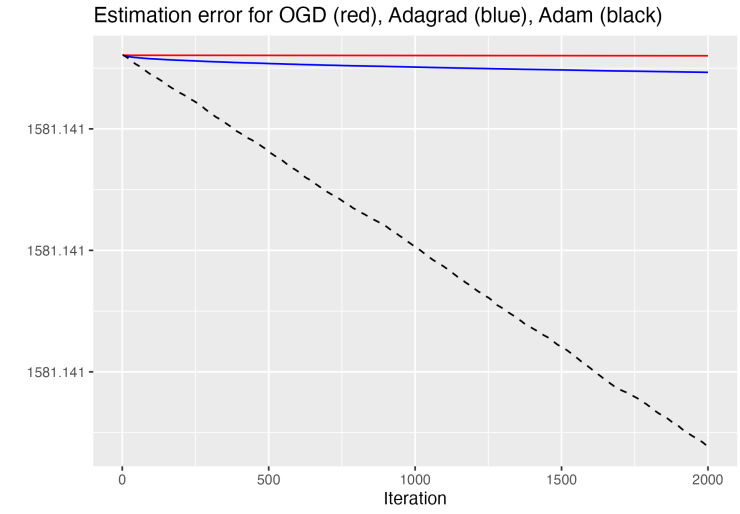


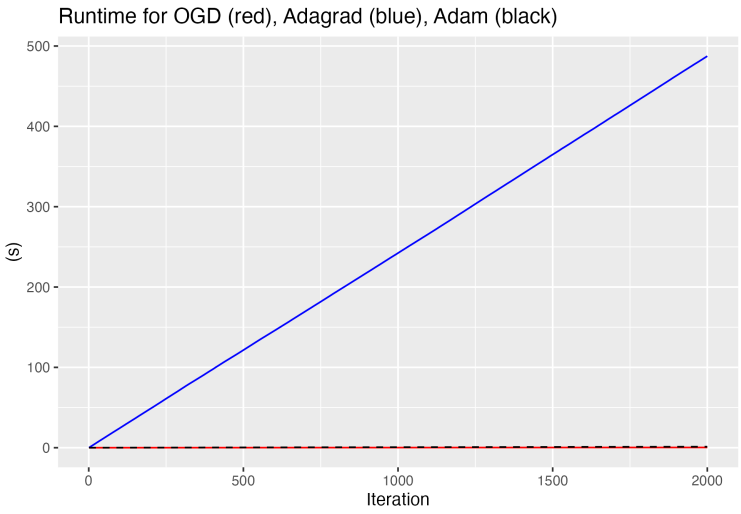


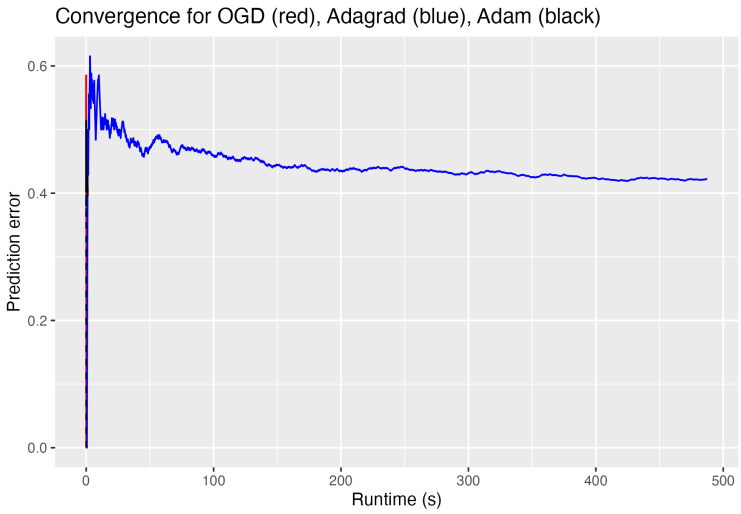
Classification:

Also similar to the original case, Adam performs best in terms of prediction error and especially estimation error.









4 Possible Improvements

Based on Adam, there are a few possible improvements.

First, by modifying the l_2 norm to ∞ norm, we have the following AdaMax algorithm:

Algorithm 4 AdaMax

Require: η : Stepsize
$l(\beta)$: Stochastic objective function (loss)
β_1 : Initial parameter vector
for $t = 1$ `to` T **do**
$M_t = \beta_1 M_{t-1} + (1-\beta_1)\nabla L_t(W_{t-1})$ (1st moment estimate)
$U_t = \max\{\beta_2 U_{t-1}, |\nabla L_t(W_{t-1})|\}$ (" ∞ " moment estimate)
$\hat{M}_t = M_t / \left(1-(\beta_1)^t\right)$ (1st moment bias correction)
$W_t = W_{t-1} - \alpha \frac{\hat{M}_t}{U_t}$ (Update)
end for

Compared to Adam, there is no need to have second moment bias correction which reduces run time, and l_∞ is as stable as l_1 and l_2.

Second, Adam and these adaptive moment methods in general may not converge in the case of irregular loss functions, e.g., not convex and varies with time.

5 Conclusion

Our experiments testing different learning rates, weight $\hat{\beta}_0$ initializations, and data dimensions show that learning rate seems to affect the performance of the optimization methods the most. Depending on the learning rate, OGD can have much worse or better prediction errors than Adagrad and Adam, however Adagrad and Adam tend to have similar prediction errors. Adam tends to have the lowest estimation error. Adagrad has been shown to have the longest runtime, whereas OGD and Adam have similar run times. Overall, from our experiments, we conclude that Adam is the best optimizer out of the three due to its adaptability to different learning rates compared to OGD, superior estimation error, and comparable runtime lower than Adagrad.

References

[1] John Duchi, Elad Hazan, and Yoram Singer.

Adaptive subgradient methods for online learning and stochastic optimization.

Journal of machine learning research, 12(7), 2011.

[2] Pavel Dvurechensky, Shimrit Shtern, and Mathias Staudigl.

First-order methods for convex optimization.

EURO Journal on Computational Optimization, 9:100015, 2021.

[3] Diederik P. Kingma and Jimmy Ba.

Adam: A method for stochastic optimization, 2017.

[4] Yi Qin, Tao Xie, Chang Xu, Angello Astorga, and Jian Lu.

Comid: Context-based multi-invariant detection for monitoring cyber-physical software, 2018.

[5] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar.

On the convergence of adam and beyond.

CoRR, abs/1904.09237, 2019.

[6] Sebastian Ruder.

An overview of gradient descent optimization algorithms.

CoRR, abs/1609.04747, 2016.

[7] Matthew D. Zeiler.

Adadelta: An adaptive learning rate method, 2012.