Online learning optimization algorithms

Minjie Jia, Joia Zhang

STSCI 6520 Project

December 7, 2023

Proposal Summary

- Problem statement: batch learning trains the entire dataset in every iteration but in online learning, data becomes available in sequential order and aims to **minimize** the regret over loss functions with respect to a competitor $\boldsymbol{u} \in V \subseteq \mathbb{R}^d$: $\text{Regret}_T(\boldsymbol{u}) := \sum_{t=1}^{T} \ell_t(\boldsymbol{x}_t) - \sum_{t=1}^{T} \ell_t(\boldsymbol{u})$
- Significance: Improvements can be made to existing methods such as an adaptive learning rate in a sparse setting
- Research strategy:
  1. Implement benchmark methods (did OGD)
  2. Implement AdaGrad (done)
  3. Construct "ML" method (used linear & logistic regression)
  4. Implement AdaGrad using our adjusted loss function (used OLS and log loss)
  5. Compare methods on classification tasks (generated sparse data for classification and regression)
  6. Write an R package with the above functions (done)
  7. Additional: implemented ADAM which builds on Adagrad
- Validation: benchmark comparison, prediction error, estimation error, runtime

Online Gradient Descent (OGD)

Assuming the loss function L is invariant to t,
Update rule:

$$\beta_{t+1} = \beta_t - \eta \nabla L(\beta_t)$$

Algorithm 1 Online Gradient Descent

Require: η : Stepsize

$l(\beta)$: Objective function (loss)

β_1 : Initial parameter vector

for $t = 1$ *to* T **do**

$g_t = \nabla_\beta l_t(\beta)$

$\beta_{t+1} = \beta_t - \eta g_t$

end

Adaptive Gradient Descent (AdaGrad)

Update rule:

$$\beta_{t+1} = \beta_t - \eta \frac{\nabla L\left(\beta_t\right)}{\sqrt{\sum_{t'=1}^{t+1} \nabla L\left(\beta_{t'-1}\right)^2}}$$

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

Root Mean Square Propagation (RMSProp)

Update rule:

$$\beta_t = \beta_{t-1} - \alpha \frac{\nabla L_t(\beta_{t-1})}{\sqrt{R_t}} \text{ where}$$

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

Similar to AdaGrad but with an exponential moving average controlled by $\gamma \in [0,1)$ (smaller $\gamma \implies$ more emphasis on recent gradients).

Adaptive Moment Estimation (Adam)

- Combine the advantages of:
  - AdaGrad - works well with sparse gradients
  - RMSProp - works well in non-stationary settings
- - Maintain exponential moving averages of gradient and its square
  - Update proportional to $\frac{\text{average gradient}}{\sqrt{\text{average squared gradient}}}$

Adaptive Moment Estimation (Adam)

Algorithm 3 Adam

$M_0 = \mathbf{0}, R_0 = \mathbf{0}$ (Initialization)

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

$$M_t = \alpha_1 M_{t-1} + (1 - \alpha_1) \nabla L_t (W_{t-1}) \quad \text{(1st moment estimate)}$$

$$R_t = \alpha_2 R_{t-1} + (1 - \alpha_2) \nabla L_t (W_{t-1})^2 \quad \text{(2nd moment estimate)}$$

$$\hat{M}_t = M_t / \left(1 - (\alpha_1)^t\right) \quad \text{(1st moment bias correction)}$$

$$\hat{R}_t = R_t / \left(1 - (\alpha_2)^t\right) \quad \text{(2nd moment bias correction)}$$

$$\beta_t = \beta_{t-1} - \eta \frac{\hat{M}_t}{\sqrt{\hat{R}_t + \epsilon}} \quad \text{(Update)}$$

Return W_T

- $\alpha > 0$ - learning rate
- $\beta_1 \in [0, 1)$ - 1 st moment decay rate (typical choice: 0.9)
- $\beta_2 \in [0, 1)$ 2nd moment decay rate (typical choice: 0.999)

[1]

Data Generation

- Dimensions: $n = 1000, p = 2000$
- β is a k-sparse vector where $k = 0.05p \sim N(100, .01)$
- $X \sim N(0, 5)$

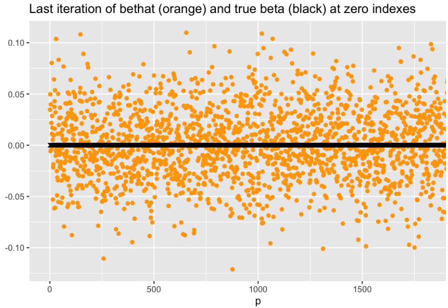
Regression

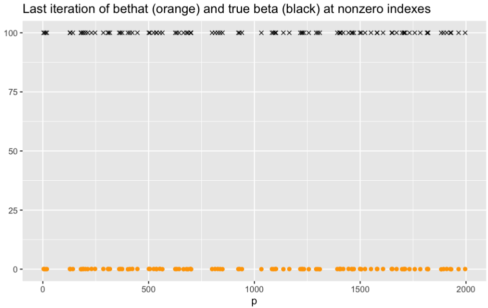
- $\epsilon \sim N(0, 1)$
- $Y = X\beta + \epsilon$

Classification

- $\pi = \frac{1}{1+\exp(-X\beta)}$
- $Y \sim Ber(\pi)$

Behavior of $\hat{\beta}$



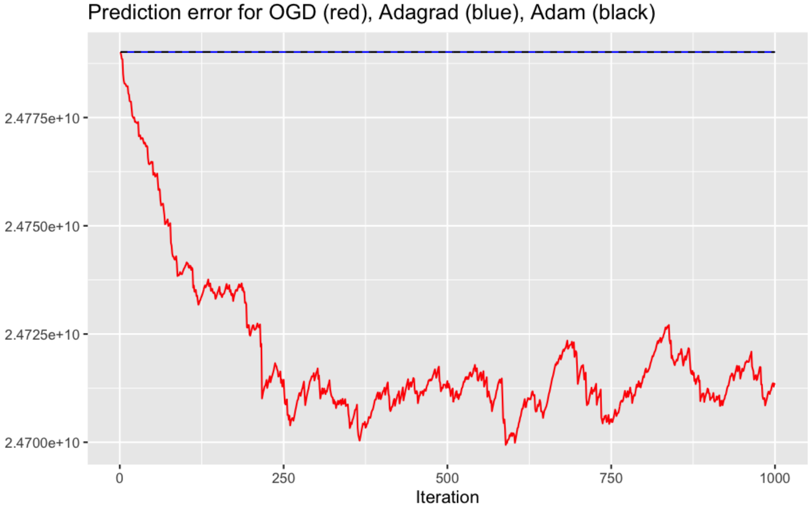


- OGD, regression
- Predicts all zero, thereby failing to capture the k nonzero coefficients

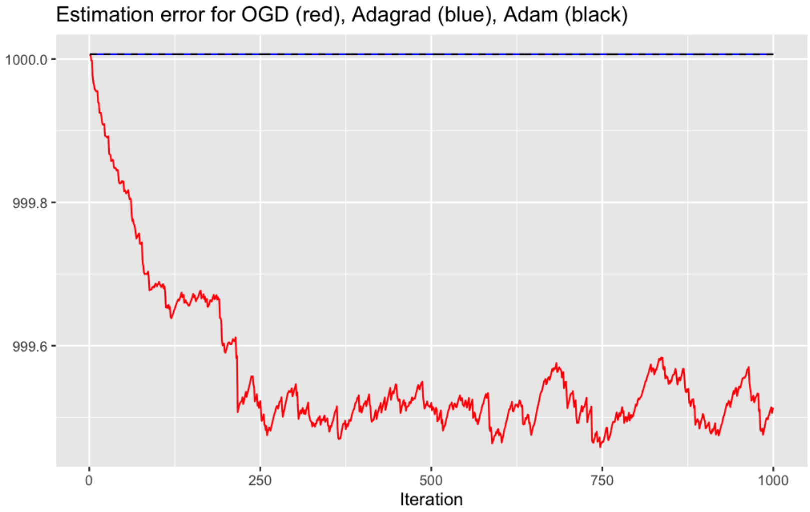
Performance Comparison

- Prediction error $(\beta_t' x_t - y_t)^2$, $\frac{\sum_{i-1}^{t} \mathbb{1}\{\hat{Y}_t \neq Y_t\}}{t}$
- Estimation error $\|\beta_t - \beta\|_2$
- Runtime (s)
- Convergence

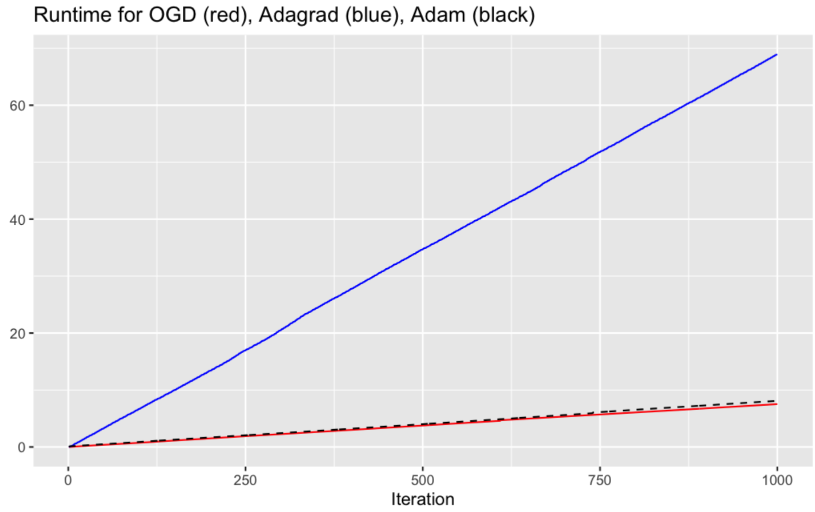
Regression: Prediction Error



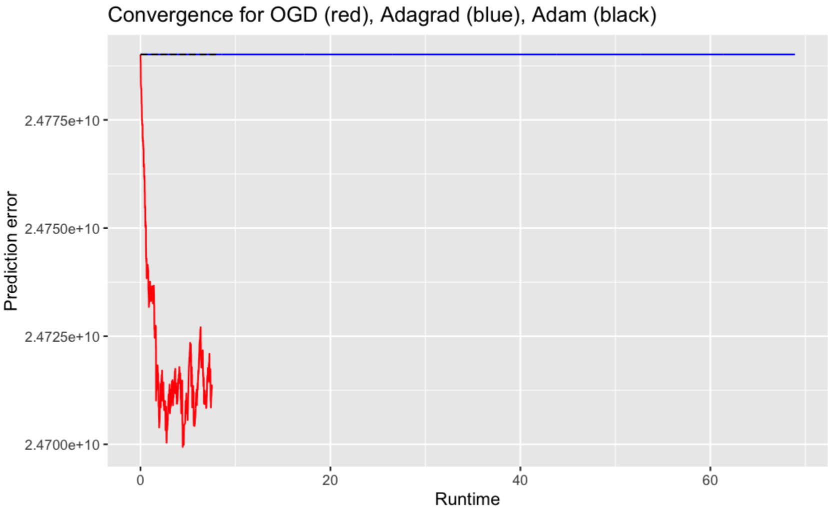
Regression: Estimation Error



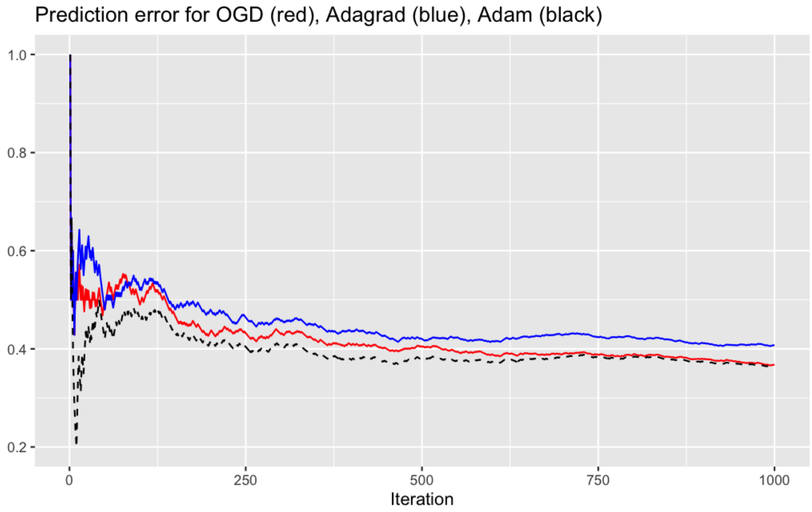
Regression: Runtime



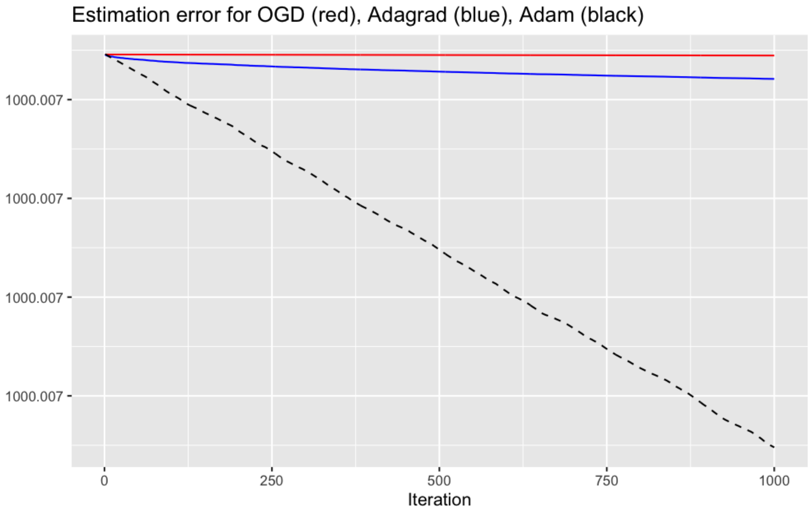
Regression: Convergence



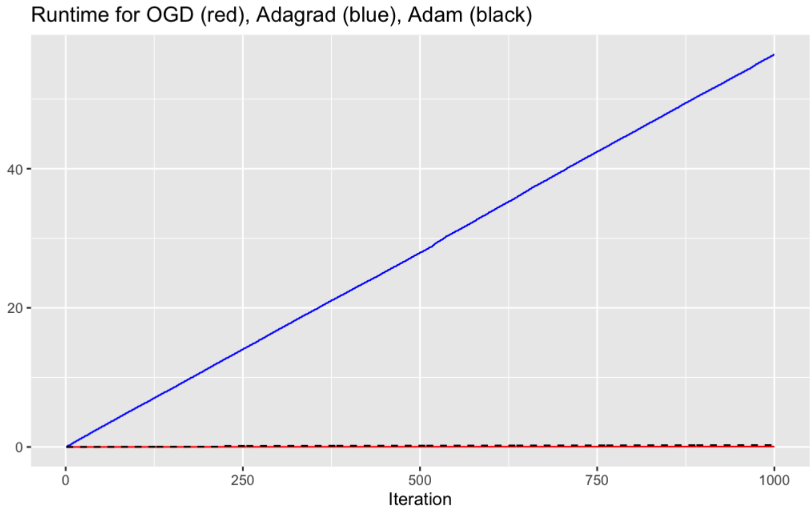
Classification: Misclassification rate



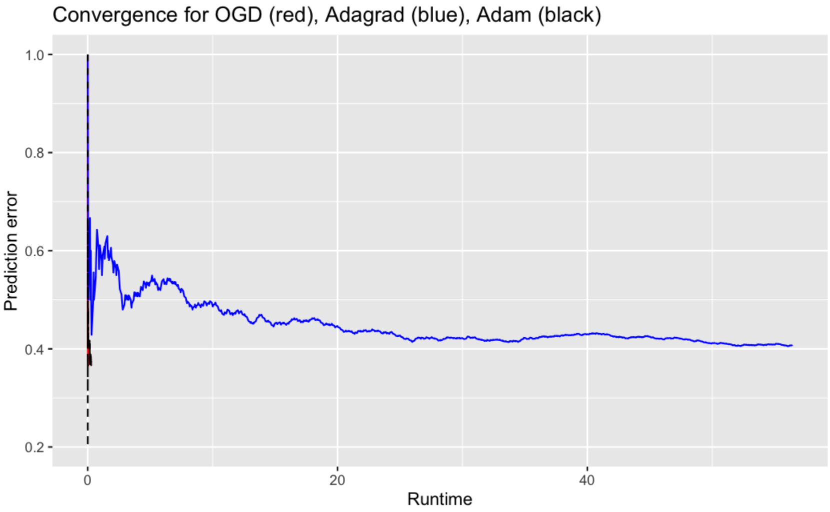
Classification: Estimation Error



Classification: Runtime



Classification: Convergence



Possible Improvements

- AdaMax: l_2 norm $\to$ l_p norm, $p \to \infty$ [3]
  - no need to have second moment bias correction, reduce run time
  - l_∞ is as stable as l_1 and l_2

AdaMax

$M_0 = \mathbf{0}, U_0 = \mathbf{0}$ (Initialization)

For $t = 1, \dots, T$:

$$M_t = \beta_1 M_{t-1} + (1-\beta_1)\nabla L_t(W_{t-1}) \quad \text{(1st moment estimate)}$$

$$U_t = \max\{\beta_2 U_{t-1}, |\nabla L_t(W_{t-1})|\} \quad \text{(" } \infty \text{ " moment estimate)}$$

$$\hat{M}_t = M_t / \left(1 - (\beta_1)^t\right) \quad \text{(1st moment bias correction)}$$

$$W_t = W_{t-1} - \alpha \frac{\hat{M}_t}{U_t} \quad \text{(Update)}$$

Return W_T

- Adam on adversarial loss functions may not converge [2]

References

Diederik P. Kingma and Jimmy Ba.
Adam: A method for stochastic optimization, 2017.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar.
On the convergence of adam and beyond.
CoRR, abs/1904.09237, 2019.

Sebastian Ruder.
An overview of gradient descent optimization algorithms.
CoRR, abs/1609.04747, 2016.

Thank you!

Questions/comments?