Small Batch or Large Batch?
Gaussian Walk with Rebound Can Teach

Presented by Fangcheng Fu

2018/09/17

Intro & Motivation

Works on accelerating mini-batch SGD

- Parallelism: architecture (MR, PS); synchronization (SSP, ASP); ADMM
- Modify stochastic gradient: momentum, Nesterov, EASGD
- Learning rate: second-order, AdaGrad, AdaDelta, Adam
- Variance reduction: SAG, SDCA, SVRG

To be studied: trade-offs in batch size

- Large batch size: more accurate
- Small batch size: more efficient

This work proposes a model for learning process that illustrates the relationship between batch size and object value decrement. With the help of this model, dynamic batch size can be achieved.

Model of Stochastic Derivative

Optimization problem

$$\vec{\theta}^* = \arg\min_{\vec{\theta}} \mathrm{F}(\vec{\theta}|\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}_i \in \mathcal{X}} \mathrm{G}^i(\vec{\theta})$$

Gradient descent

$$\vec{\theta}' = \vec{\theta} - \eta \cdot \mathrm{f}_{\vec{\theta}} = \vec{\theta} - \eta \cdot \frac{\sum_{\mathbf{x}_i \in \mathcal{X}} \mathrm{g}^i_{\vec{\theta}}}{|\mathcal{X}|}$$

Mini-batch SGD

$$\mathrm{f}_{\vec{\theta}} \approx \hat{\mathrm{f}}_{\vec{\theta}} = \frac{\sum_{\mathbf{y}_j \in \mathcal{Y}} \mathrm{g}^j_{\vec{\theta}}}{|\mathcal{Y}|} \qquad \{\mathcal{Y} \subset \mathcal{X} || \mathcal{Y}| << |\mathcal{X}|, \}$$

Model of Stochastic Derivative

Mini-batch SGD

$$\hat{\mathrm{f}}_{\vec{\theta}} = \frac{\sum_{\mathbf{y}_j \in \mathcal{Y}} \mathrm{g}^{j}_{\vec{\theta}}}{|\mathcal{Y}|} = \frac{\sum_{\mathbf{y}_j \in \mathcal{Y}} (\mathrm{f}_{\vec{\theta}} + \vec{\xi}_j)}{|\mathcal{Y}|} = \mathrm{f}_{\vec{\theta}} + \frac{\sum_{\mathbf{y}_j \in \mathcal{Y}} \vec{\xi}_j}{|\mathcal{Y}|} \qquad \sum_{\mathbf{x}_i \in \mathcal{X}} \vec{\xi}_i = \mathbf{0}$$

The second term can be treated as a random vector following a multi-dimensional normal distribution (*Central Limit Theorem*)

$$\frac{\sum_{\mathbf{y}_j \in \mathcal{Y}} \vec{\xi}_j}{|\mathcal{Y}|} \sim \mathcal{N}(\mathbf{0}, \frac{\Sigma}{|\mathcal{Y}|})$$

The estimation of gradient is effected by batch size (inversely proportional), which explains the high fluctuation in learning with SGD

Dynamically update the batch size will be a wiser strategy

Model of Stochastic Learning Process

Decrease of object value

$$\mathrm{F}(\vec{\theta}_0 - \eta \hat{\mathrm{f}}_{\vec{\theta}_0}) - \mathrm{F}(\vec{\theta}_0) \approx \mathrm{f}_{\theta_0}^T(\theta_0 - \eta \cdot \hat{\mathrm{f}}_{\theta_0} - \theta_0) = -\eta \mathrm{f}_{\vec{\theta}_0}^T \hat{\mathrm{f}}_{\vec{\theta}_0}$$

$$= -\eta \mathrm{f}_{\vec{\theta}_0}^T \cdot (\mathrm{f}_{\vec{\theta}_0} + \frac{\sum_{\mathrm{y}_j \in \mathcal{Y}} \vec{\xi}_j}{|\mathcal{Y}|}) = -\eta \cdot \mathrm{f}_{\vec{\theta}_0}^T \mathrm{f}_{\vec{\theta}_0} - \eta \cdot \mathrm{f}_{\vec{\theta}_0}^T \cdot \vec{\varepsilon}_{\mathcal{Y}}$$

Since $\vec{\varepsilon}_{\mathcal{Y}} \sim \mathcal{N}(\mathbf{0}, \frac{\Sigma}{|\mathcal{Y}|})$, we have $\vec{\varepsilon}_{\mathcal{Y}}^T \cdot \hat{\mathrm{f}}_{\vec{\theta}_0} \sim \mathcal{N}(0, \frac{\hat{\mathrm{f}}_{\vec{\theta}_0}^T \cdot \Sigma \cdot \hat{\mathrm{f}}_{\vec{\theta}_0}}{|\mathcal{Y}|})$, where Σ's unbiased estimation

$$\hat{\Sigma} = \frac{1}{|\mathcal{Y}| - 1} \sum_{\mathrm{y}_j \in \mathcal{Y}} \left(\mathrm{g}_{\vec{\theta}_0}^j - \frac{\sum_{\mathrm{y}_j \in \mathcal{Y}} \mathrm{g}_{\vec{\theta}_0}^j}{|\mathcal{Y}|} \right) \left(\mathrm{g}_{\vec{\theta}_0}^j - \frac{\sum_{\mathrm{y}_j \in \mathcal{Y}} \mathrm{g}_{\vec{\theta}_0}^j}{|\mathcal{Y}|} \right)^T = \frac{\sum_{\mathrm{y}_j \in \mathcal{Y}} (\mathrm{g}_{\vec{\theta}_0}^j - \hat{\mathrm{f}}_{\vec{\theta}_0})(\mathrm{g}_{\vec{\theta}_0}^j - \hat{\mathrm{f}}_{\vec{\theta}_0})^T}{|\mathcal{Y}| - 1}$$

The decrease of object value satisfies a Gaussian distribution

Model of Stochastic Learning Process

Game of Gaussian Walk with Rebound

Abstract the learning process as a random walk game with a Gaussian dice

Domain of game states: $[S^*, +\infty)$

At state S_i, player pays m for a moving step $\triangle s_i \sim \mathcal{N}\left(\mu_i, \frac{\sigma_i^2}{m}\right)$

Connecting with SGD

S^*: optimal objective value

m: batch size

η: transfer speed/learning rate

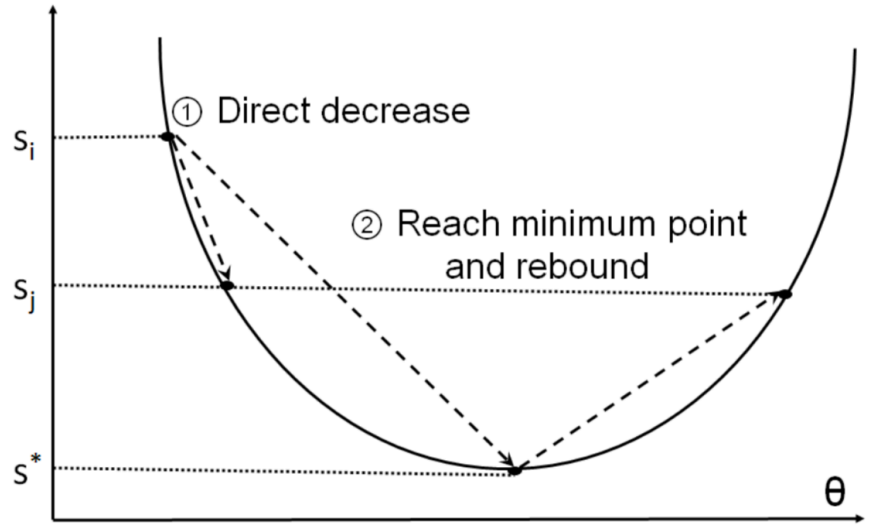
Gaussian dice is solely determined by current state

Model of Stochastic Learning Process

Game of Gaussian Walk with Rebound

Abstract the learning process as a random walk game with a Gaussian dice

State transfer: two paths



$$s_{t+1} = |s_t - S^* - \eta \triangle s_t| + S^*$$

$$= \begin{cases} s_t - \eta \triangle s_t, & \text{if } s_t - \eta \triangle s_t \geq S^* \\ 2S^* + \eta \triangle s_t - s_t, & \text{otherwise} \end{cases}$$

Model of Stochastic Learning Process

Game of Gaussian Walk with Rebound

Abstract the learning process as a random walk game with a Gaussian dice

$p_m^{s_t}(\triangle s_t)$ is the PDF for a random moving step $\triangle s_t \sim \mathcal{N}\left(\mu_t, \frac{\sigma_t^2}{m}\right)$

Expected state transfer

$$\mathbf{E}_m^{s_t}(s_{t+1}) = \int_{-\infty}^{+\infty} p_m^{s_t}(\triangle s_t)(|s_t - S^* - \eta \cdot \triangle s_t| + S^*) d\triangle s_t$$

$$= \eta \int_{-\infty}^{+\infty} p_m^{s_t}(\triangle s_t) \left| \triangle s_t - \frac{s_t - S^*}{\eta} \right| d\triangle s_t + S^*$$

$$= (s_t - S^* - \eta\mu_t)\{\Phi(a) - \Phi(-a)\} + \frac{\eta\sigma_t}{\sqrt{m}} \sqrt{\frac{2}{\pi}} e^{-\frac{a^2}{2}} + S^*$$

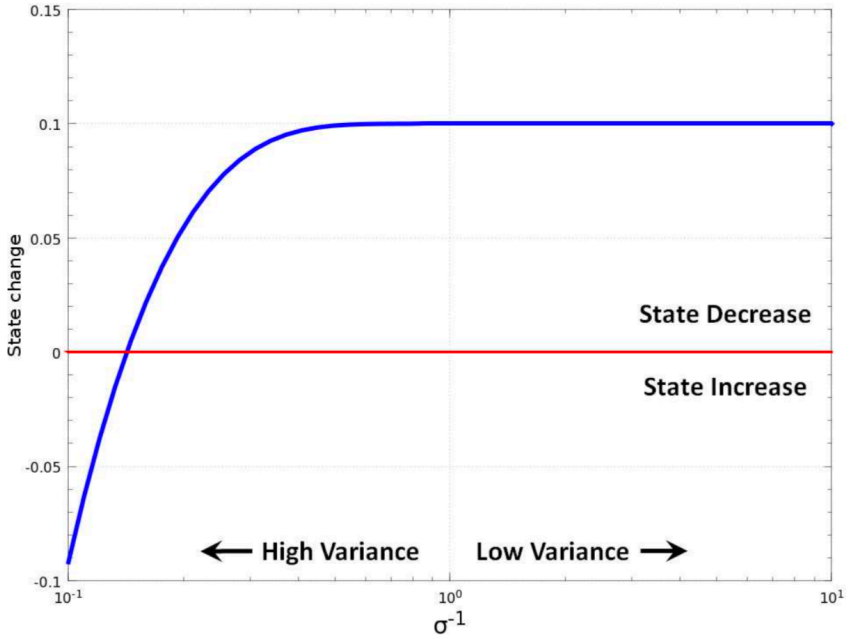
$$\text{where } a = \frac{s_t - S^* - \eta\mu_t}{\eta\sigma_t} \sqrt{m}$$

Full-batch $\lim_{m \to +\infty} \mathbf{E}_m^{s_t}(s_{t+1}) = \mathtt{sign}(a)(s_t - S^* - \eta\mu_t) + S^* = |s_t - S^* - \eta\mu_t| + S^*$

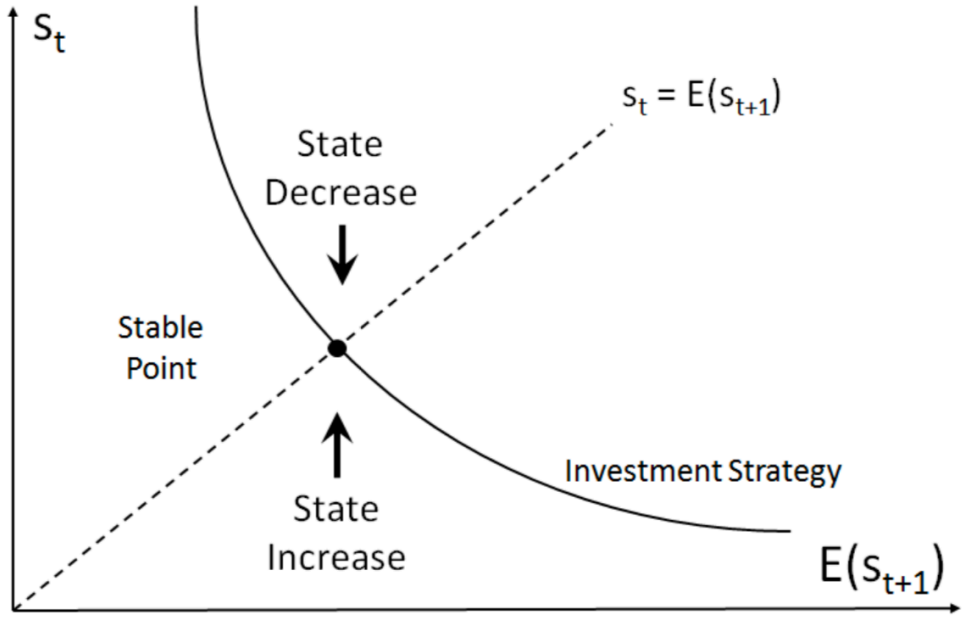
Model of Stochastic Learning Process

Game of Gaussian Walk with Rebound

Abstract the learning process as a random walk game with a Gaussian dice



Correlation between state change and variance



Three scenarios of an investment strategy

Batch-Adaptive SGD

Given a limited budget (total number of samples), what is the best strategy of quota allocation to achieve minimum expected state?

Mini-batch SGD: use same amount of samples

Full-batch GD: use all samples

Let $\mathrm{Q}(s_t, M)$ denote the lowest achievable state given state and budget

$$\mathrm{Q}(s_t, M) = \min\{\mathrm{Q}(\mathrm{E}_m^{s_t}(s_{t+1}), M - m) | m = 1, 2, \cdots, M\}$$

Solve with dynamic programming? -- Not practical

1) No knowledge of future states
2) Time consuming when M is large
3) Solution is only valid for current state

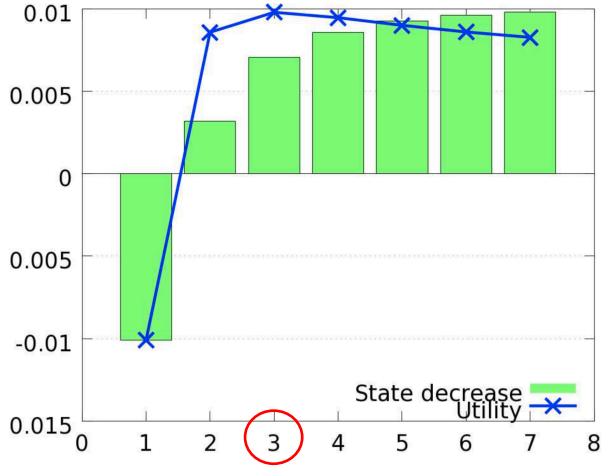
A heuristic strategy that maximizes the efficiency of decreasing object value is required

Batch-Adaptive SGD

MaxDec-Efficiency Strategy

Game setting	$\mu_t = s_t, \sigma_t \equiv 1, \eta = 0.1, S^* = 0, M = 7$					
$s_0 = 0.1$	Round 1		Round 2		Round 3	
	m_1	$\mathrm{E}^{s_0}_{m_1}(s_1)$	m_2	$\mathrm{E}^{s_1}_{m_2}(s_2)$	m_3	$\mathrm{E}^{s_2}_{m_3}(s_3)$
I	7	0.0902	-	-	-	-
II	2	0.0968	2	0.0946	2	0.0930*
III	3	0.09296	4	0.0856	-	-

* The remaining of 1 budget can not make the state decrease based on Equation (7). So it stops at 3rd round.



Consider the efficiency of investment: $\mathrm{u}(m, s_t) = \dfrac{s_t - \mathrm{E}^{s_t}_m(s_{t+1})}{m}$

➔ $m_{\mathrm{opt}} = \arg\max_m \mathrm{u}(m, s_t)$

Batch-Adaptive SGD

Algorithm Design

Algorithm 1 Batch-Adaptive SGD

procedure BA-SGD($\mathcal{X}, \vec{\theta}, \eta, M, m_0$)
  while $M > 0$ **do**
    $\mathcal{Y} \leftarrow \emptyset$
    repeat
      random sample $\mathcal{Z}$ from $\mathcal{X} - \mathcal{Y}$ with $|\mathcal{Z}| = m_0$
      $\mathcal{Y} \leftarrow \mathcal{Y} \bigcup \mathcal{Z}$
      calculate $\hat{\mathrm{f}}_{\vec{\theta}}, \hat{\Sigma}$ with $\mathcal{Y}$ ▷ Equation (2) and (5)
      $s_t \leftarrow \mathrm{F}(\vec{\theta}|\mathcal{Y}), \mu_t \leftarrow \hat{\mathrm{f}}_{\vec{\theta}}^T \hat{\mathrm{f}}_{\vec{\theta}}, \sigma_t \leftarrow \sqrt{\hat{\mathrm{f}}_{\vec{\theta}}^T \hat{\Sigma} \hat{\mathrm{f}}_{\vec{\theta}} / |\mathcal{Y}|}$
      $m^* \leftarrow \arg\max_m \mathrm{u}(m, s_t)$ where $S^* = 0$
    until $|\mathcal{Y}| \geq \min\{m^*, |\mathcal{X}|\}$
    $\vec{\theta} \leftarrow \vec{\theta} - \eta \cdot \hat{\mathrm{f}}_{\vec{\theta}}$
    $M \leftarrow M - |\mathcal{Y}|$
  end while
  return $\vec{\theta}$
end procedure

Optimizations

Avoid redundant computation

$$\hat{\mathrm{f}}_{\vec{\theta}} = \frac{\sum_{\mathrm{y}_j \in \mathcal{Y}_1 \cup \mathcal{Y}_2} \mathrm{g}_{\vec{\theta}}^j}{|\mathcal{Y}_1| + |\mathcal{Y}_2|} = \frac{\sum_{\mathrm{y}_j \in \mathcal{Y}_1} \mathrm{g}_{\vec{\theta}}^j + \sum_{\mathrm{y}_k \in \mathcal{Y}_2} \mathrm{g}_{\vec{\theta}}^k}{|\mathcal{Y}_1| + |\mathcal{Y}_2|}$$

$$\hat{\Sigma} = \frac{\sum_{\mathrm{y}_j \in \mathcal{Y}_1 \cup \mathcal{Y}_2} \mathrm{g}_{\vec{\theta}}^{j\,T} \mathrm{g}_{\vec{\theta}}^j - (|\mathcal{Y}_1| + |\mathcal{Y}_2|) \hat{\mathrm{f}}_{\vec{\theta}}^T \hat{\mathrm{f}}_{\vec{\theta}}}{|\mathcal{Y}_1| + |\mathcal{Y}_2| - 1}$$

Reduce space cost

Assume parameters are independent of each other and achieve diagonal covariance matrix

Evaluation

Experimental Setup

Baselines: GD, SGD, SGD with fixed lr, mini-batch SGD
Datasets: #instances >> #features
Model: MLP with 2 hidden layers, 5 neurons in each layer

Data	Class	Feature*	size	Name in UIC Repo
Adult	2	7	48,842	Adult
Bank [24]	2	24	41,188	Bank Marketing Data Set
Connect-4	3	42	67,557	Connect-4
Cover Type	7	14	581,012	Covertype
Credit [36]	2	23	30,000	default of credit card clients
Letter Recog	26	16	20,000	Letter Recognition
Sensor	11	48	58,509	Dataset for Sensorless Drive Diagnosis
Skin [3]	2	3	100,000+	Skin Segmentation
Teacher	4	16	10,800	Firm-Teacher_Clave-Direction_Classification

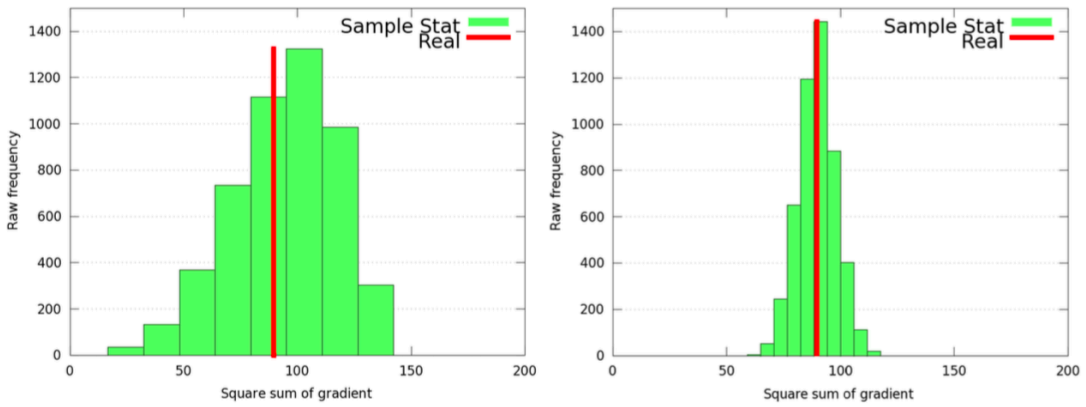
*Feature dimension may be different from raw data since we unfold categorical attributes and delete some which have too many values but for each value there is only a few number of instances.
+We remove some duplicate synthetic data records.

Evaluation

Model Verification

Gaussian Distribution of Gradient's Square Sum $\mathrm{f}_{\vec{\theta}}^{T} \cdot \mathrm{f}_{\vec{\theta}}$

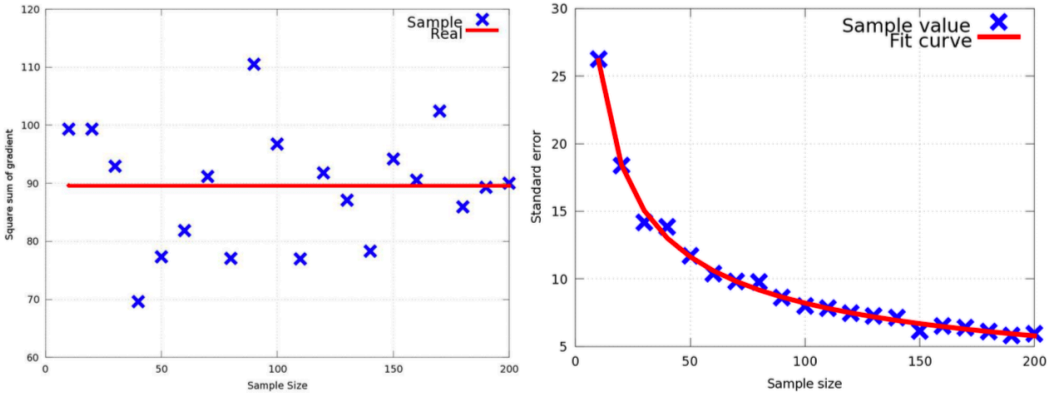
Look similar to normal distribution



(a) Sample size 10 (b) Sample size 100

Larger sample size incurs smaller variance

Size & std error in the form of $y \sim \frac{1}{\sqrt{x}}$

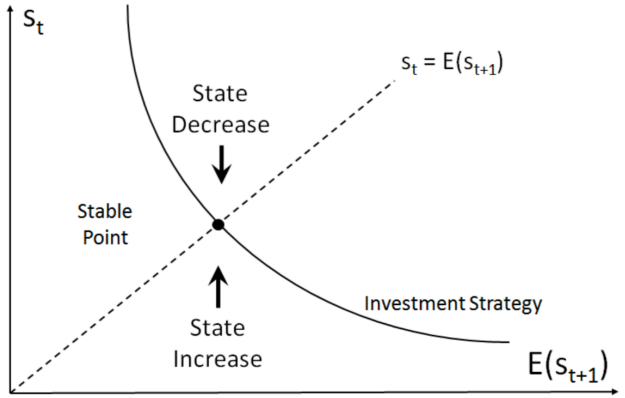


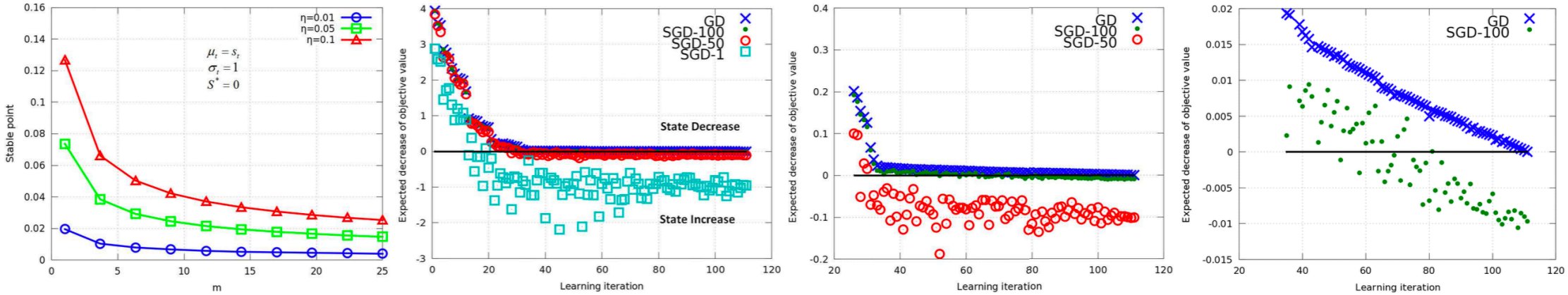
(a) Average (b) Standard Error

Evaluation

Model Verification
Stable Point of SGD

All SGD-based algorithms have negative decrease
Larger batch size has a smaller stable point





(a) Simulation (b) Original (c) Zoom-in (d) Further Zoom-in

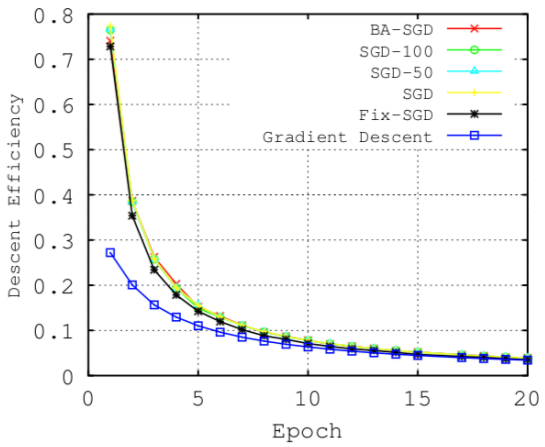
Figure 7: Verification of Stable Point for SGD. The x-axis represents the learning iterations of gradient descent (GD). The y-axis represents the decrease of objective value for a single iteration.

Evaluation

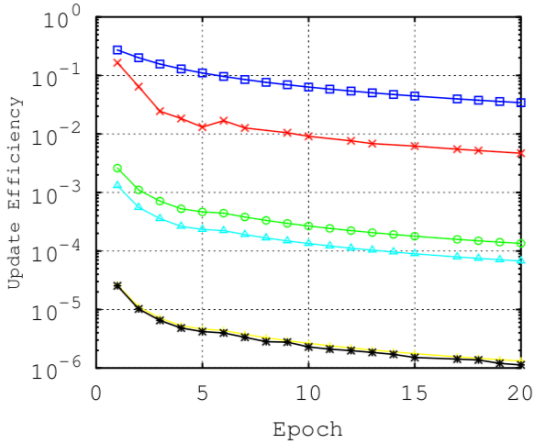
General Comparison

Descent Efficiency & Update Efficiency

Descent Efficiency: decrease of object value w.r.t. #scanned data
Update Efficiency: decrease of object value w.r.t. #iteration



(a) Descent Efficiency



(b) Update Efficiency

Evaluation

General Comparison

Object Value after 20 Epochs

Skin dataset has low noise in data, which may make Fix-SGD more efficient
Thus the impact of noise should be studied

	Data								
	Adult	Bank	Connect-4	Cover Type	Credit	Letter Rec.	Sensor	Skin	Teacher
BA-SGD	**0.5040**	**0.3534**	**0.3116**	**2.0070**	0.1683	**3.3375**	**1.9249**	0.1464	**0.9084**
SGD-100	0.5404	0.3562	0.4708	2.0773	0.1658	3.7909	3.3680	0.1469	1.3659
SGD-50	0.5378	0.3557	0.4681	2.0639	0.1651	3.7976	3.3246	0.1451	1.3556
SGD	0.5795	0.3543	0.4713	2.3864	0.1628	3.8857	3.6366	0.1433	1.7368
Fix-SGD	0.6624	0.4190	0.4339	6.0445	**0.1577**	4.1610	3.2110	**0.0030**	3.8869
Gradient Descent	0.9569	0.7078	0.7916	4.2286	0.3700	10.9911	5.8926	0.5546	2.9549

Evaluation

Impact of Noise

As noise scale increases, strategies of bigger constant investment performs better

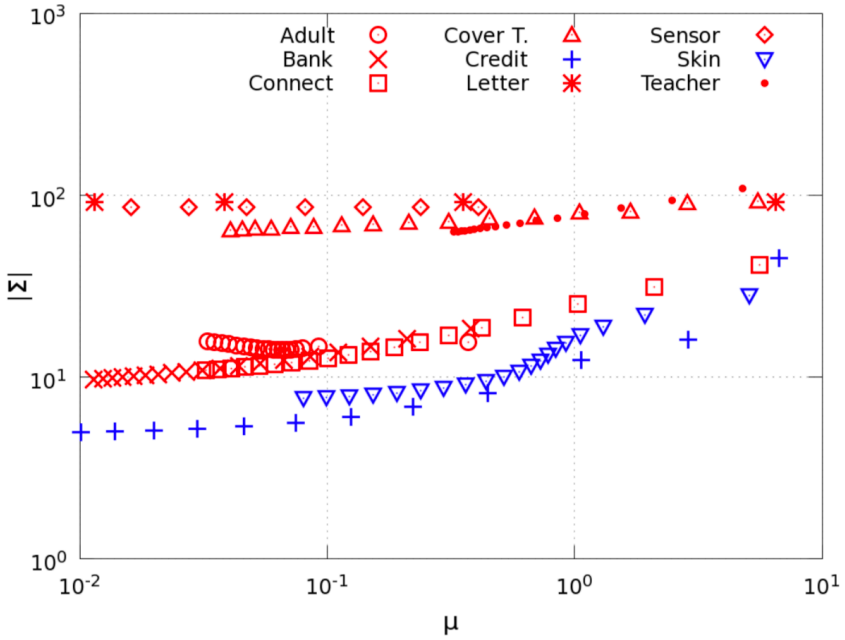


Figure 9: Noise Demonstration in Real Data

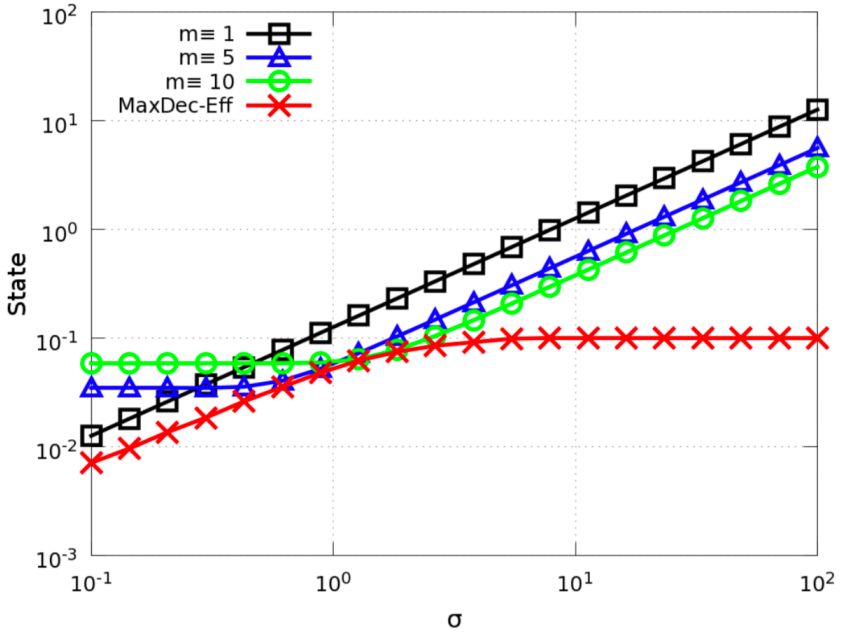


Figure 10: Impact of Noise in Simulation