Optimization Algorithms in Deep Learning

Stochastic Gradient Descent and ADAM

Xiaoxi Shen and Jialong Li

Texas State University

November 17, 2023

ERM Framework for Deep Learning

- Under ERM, given the training data set $\{(\boldsymbol{X}_i, \boldsymbol{Y}_i)\}_{i=1}^n$ and a loss function $\mathcal{L}$, training a deep neural network can be formulated as

$$\hat{f} = \mathrm{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(\boldsymbol{X}_i), \boldsymbol{Y}_i), \tag{1}$$

where

$$\mathcal{F} = \left\{ f : \mathbb{R}^d \to \mathbb{R} : f(\boldsymbol{x}) = \sigma_L(\boldsymbol{W}_L \cdots \sigma_2 \boldsymbol{W}_2(\sigma_1(\boldsymbol{W}_1 \boldsymbol{x}))) \right\}$$

ERM Framework for Deep Learning

- Under ERM, given the training data set $\{(\boldsymbol{X}_i, \boldsymbol{Y}_i)\}_{i=1}^n$ and a loss function $\mathcal{L}$, training a deep neural network can be formulated as

$$\hat{f} = \mathrm{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(\boldsymbol{X}_i), \boldsymbol{Y}_i), \tag{1}$$

where

$$\mathcal{F} = \left\{ f : \mathbb{R}^d \to \mathbb{R} : f(\boldsymbol{x}) = \sigma_L(\boldsymbol{W}_L \cdots \sigma_2 \boldsymbol{W}_2(\sigma_1(\boldsymbol{W}_1 \boldsymbol{x}))) \right\}$$

- Each member in $\mathcal{F}$ is parameterized by weight matrices $\Theta = (\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L)$, so problem (1) is equivalent to

$$\hat{\Theta} = \mathrm{argmin}_{\Theta} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i).$$

Gradient Descent

- Gradient Descent Updating Scheme:

$$\Theta^{(k+1)} = \Theta^{(k)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i).$$

Gradient Descent

- Gradient Descent Updating Scheme:

$$\Theta^{(k+1)} = \Theta^{(k)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_{\Theta}(\boldsymbol{X}_i), \boldsymbol{Y}_i).$$

- Two natural questions for GD are
  1. When GD converges, does it converge to a local minimum or a saddle point?
  2. How long does it take to converge (i.e. convergence speed)?

A Recap from Calculus – Saddle Points

Let $g : \mathbb{R}^p \to \mathbb{R}$ and our goal is to minimize $g(\boldsymbol{x}) \in C^2$.

- (*2nd Order Necessary Condition*) If $\boldsymbol{x}^*$ is a local minimizer, then
  - $\nabla g(\boldsymbol{x}^*) = 0$. (*1st Order Necessary Condition*)
  - $\nabla^2 g(\boldsymbol{x}^*)$ is positive semi-definite, i.e. $\lambda_{\min}(\nabla^2 g(\boldsymbol{x}^*)) \geq 0$.

A Recap from Calculus – Saddle Points

Let $g : \mathbb{R}^p \to \mathbb{R}$ and our goal is to minimize $g(\boldsymbol{x}) \in C^2$.

- (*2nd Order Necessary Condition*) If $\boldsymbol{x}^*$ is a local minimizer, then
  - $\nabla g(\boldsymbol{x}^*) = 0$. (*1st Order Necessary Condition*)
  - $\nabla^2 g(\boldsymbol{x}^*)$ is positive semi-definite, i.e. $\lambda_{\min}(\nabla^2 g(\boldsymbol{x}^*)) \geq 0$.
- A critical point $\boldsymbol{x}^*$ of g ($\nabla g(\boldsymbol{x}^*) = 0$) can be categorized as follow:

$$\lambda_{\min}(\nabla^2 g(\boldsymbol{x}^*)) \begin{cases} > 0 & \text{local minimum} \\ = 0 & \text{local minimum or non-strict saddle point} \\ < 0 & \text{strict saddle point} \end{cases}$$

A Recap from Calculus – Saddle Points

Let $g : \mathbb{R}^p \to \mathbb{R}$ and our goal is to minimize $g(\boldsymbol{x}) \in C^2$.

- (*2nd Order Necessary Condition*) If $\boldsymbol{x}^*$ is a local minimizer, then
  - $\nabla g(\boldsymbol{x}^*) = 0$. (*1st Order Necessary Condition*)
  - $\nabla^2 g(\boldsymbol{x}^*)$ is positive semi-definite, i.e. $\lambda_{\min}(\nabla^2 g(\boldsymbol{x}^*)) \geq 0$.
- A critical point $\boldsymbol{x}^*$ of g ($\nabla g(\boldsymbol{x}^*) = 0$) can be categorized as follow:

$$\lambda_{\min}(\nabla^2 g(\boldsymbol{x}^*)) \begin{cases} > 0 & \text{local minimum} \\ = 0 & \text{local minimum or non-strict saddle point} \\ < 0 & \text{strict saddle point} \end{cases}$$

- Strict saddle points require that there is at least one direction along which the curvature is strictly negative.

A Recap from Calculus – Saddle Points

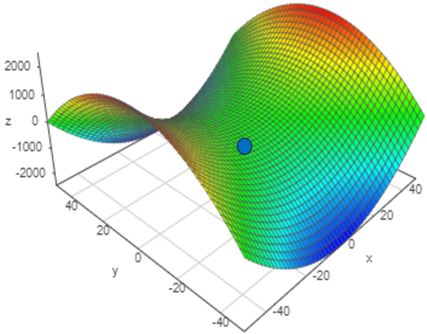
Let $g : \mathbb{R}^p \to \mathbb{R}$ and our goal is to minimize $g(\boldsymbol{x}) \in C^2$.

- (*2nd Order Necessary Condition*) If $\boldsymbol{x}^*$ is a local minimizer, then
  - $\nabla g(\boldsymbol{x}^*) = 0$. (*1st Order Necessary Condition*)
  - $\nabla^2 g(\boldsymbol{x}^*)$ is positive semi-definite, i.e. $\lambda_{\min}(\nabla^2 g(\boldsymbol{x}^*)) \geq 0$.
- A critical point $\boldsymbol{x}^*$ of g ($\nabla g(\boldsymbol{x}^*) = 0$) can be categorized as follow:

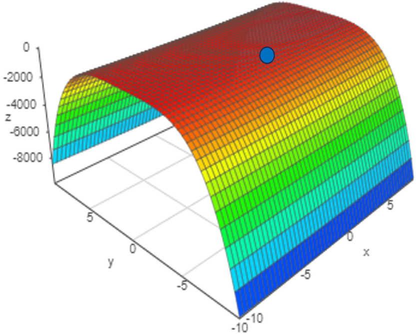
$$\lambda_{\min}(\nabla^2 g(\boldsymbol{x}^*)) \begin{cases} > 0 & \text{local minimum} \\ = 0 & \text{local minimum or non-strict saddle point} \\ < 0 & \text{strict saddle point} \end{cases}$$

- Strict saddle points require that there is at least one direction along which the curvature is strictly negative.
- In general, distinguishing local minima and non-strict saddle points is NP-hard.

A Recap from Calculus – Saddle Points



Strict Saddle Point
$f(x, y) = x^2 - y^2$



Non-Strict Saddle Point
$f(x, y) = x^2 - y^4$

Theoretical Guarantee of Gradient Descent (Lee et al. (2016)

Suppose we minimize a differential function $g(\boldsymbol{x})$ via gradient descent:
$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \eta \nabla g(\boldsymbol{x}^{(k)})$

Theoretical Guarantee of Gradient Descent (Lee et al. (2016)

Suppose we minimize a differential function $g(\boldsymbol{x})$ via gradient descent: $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \eta \nabla g(\boldsymbol{x}^{(k)})$

(A1) $\|\nabla g(\boldsymbol{x}) - \nabla g(\boldsymbol{y})\| \leq \gamma \|\boldsymbol{x} - \boldsymbol{y}\|$.

Theoretical Guarantee of Gradient Descent (Lee et al. (2016)

Suppose we minimize a differential function $g(\boldsymbol{x})$ via gradient descent: $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \eta \nabla g(\boldsymbol{x}^{(k)})$

(A1) $\|\nabla g(\boldsymbol{x}) - \nabla g(\boldsymbol{y})\| \leq \gamma \|\boldsymbol{x} - \boldsymbol{y}\|$.

(A2) The initial point $\boldsymbol{x}_0$ is sampled from a distribution of a continuous random variable.

Theoretical Guarantee of Gradient Descent (Lee et al. (2016)

Suppose we minimize a differential function $g(\boldsymbol{x})$ via gradient descent: $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \eta \nabla g(\boldsymbol{x}^{(k)})$

(A1) $\|\nabla g(\boldsymbol{x}) - \nabla g(\boldsymbol{y})\| \leq \gamma \|\boldsymbol{x} - \boldsymbol{y}\|$.

(A2) The initial point $\boldsymbol{x}_0$ is sampled from a distribution of a continuous random variable.

Theorem

If $g \in C^2$ and $\boldsymbol{x}^*$ is a strit saddle point, then under (A1), (A2) and the assumption that $0 < \eta < 1/\gamma$,

$$\mathbb{P}\left(\lim_k \boldsymbol{x}^{(k)} = \boldsymbol{x}^*\right) = 0.$$

Theoretical Guarantee of Gradient Descent (Lee et al. (2016)

Suppose we minimize a differential function $g(\boldsymbol{x})$ via gradient descent: $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \eta \nabla g(\boldsymbol{x}^{(k)})$

(A1) $\|\nabla g(\boldsymbol{x}) - \nabla g(\boldsymbol{y})\| \leq \gamma \|\boldsymbol{x} - \boldsymbol{y}\|$.

(A2) The initial point $\boldsymbol{x}_0$ is sampled from a distribution of a continuous random variable.

Theorem

If $g \in C^2$ and $\boldsymbol{x}^*$ is a strit saddle point, then under (A1), (A2) and the assumption that $0 < \eta < 1/\gamma$,

$$\mathbb{P}\left(\lim_k \boldsymbol{x}^{(k)} = \boldsymbol{x}^*\right) = 0.$$

- With probability 1, gradient descent with a random initialization will escape saddle points eventually.

Theoretical Guarantee of Gradient Descent (Lee et al. (2016)

Suppose we minimize a differential function $g(\boldsymbol{x})$ via gradient descent:
$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \eta \nabla g(\boldsymbol{x}^{(k)})$

(A1) $\|\nabla g(\boldsymbol{x}) - \nabla g(\boldsymbol{y})\| \leq \gamma \|\boldsymbol{x} - \boldsymbol{y}\|$.

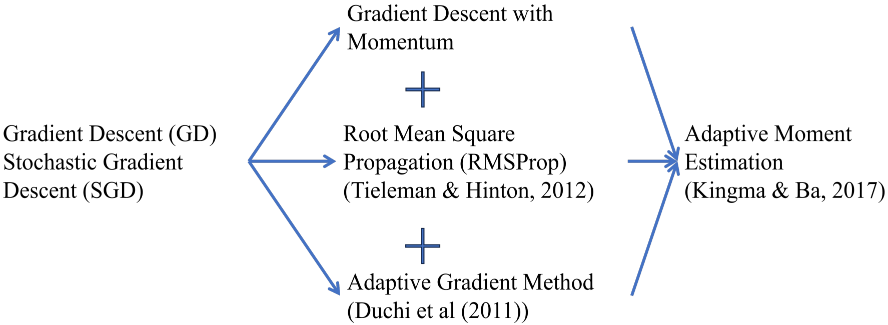
(A2) The initial point $\boldsymbol{x}_0$ is sampled from a distribution of a continuous random variable.

Theorem

If $g \in C^2$ and $\boldsymbol{x}^*$ is a strit saddle point, then under (A1), (A2) and the assumption that $0 < \eta < 1/\gamma$,

$$\mathbb{P}\left(\lim_k \boldsymbol{x}^{(k)} = \boldsymbol{x}^*\right) = 0.$$

- With probability 1, gradient descent with a random initialization will escape saddle points eventually.
- It may take exponential time to escape (Du et al. 2017).


Gradient Descent (GD)
Stochastic Gradient Descent (SGD)
Gradient Descent with Momentum
Root Mean Square Propagation (RMSProp) (Tieleman & Hinton, 2012)
Adaptive Gradient Method (Duchi et al (2011))
Adaptive Moment Estimation (Kingma & Ba, 2017)

Stochastic Gradient Descent (SGD)

- $\Theta^{(k+1)} = \Theta^{(k)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_{\Theta}(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.

Stochastic Gradient Descent (SGD)

- $\Theta^{(k+1)} = \Theta^{(k)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.
- Observation

$$\begin{aligned}
\frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i) &= \mathbb{E}_{\mathbb{P}_n}[\nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}), \boldsymbol{Y})] \\
&= \int \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}), \boldsymbol{Y}) d\mathbb{P}_n.
\end{aligned}$$

Stochastic Gradient Descent (SGD)

- $\Theta^{(k+1)} = \Theta^{(k)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.
- Observation

$$\frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i) = \mathbb{E}_{\mathbb{P}_n}[\nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}), \boldsymbol{Y})]$$
$$= \int \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}), \boldsymbol{Y}) d\mathbb{P}_n.$$

- The Idea of SGD is to replace $\mathbb{E}_{\mathbb{P}_n}[\nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}), \boldsymbol{Y})]$ by an unbiased estimator and a typical choice is

$$\frac{1}{B} \sum_{i \in \mathcal{S}_j} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i), \quad j = 1, \ldots, \lceil n/B \rceil.$$

where $\mathcal{S}_j$ is chosen uniformly at random among the set of all subsets of size B from $\{1, \ldots, n\}$.

Stochastic Gradient Descent (SGD)

- $\Theta^{(k+1)} = \Theta^{(k)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.
- Observation

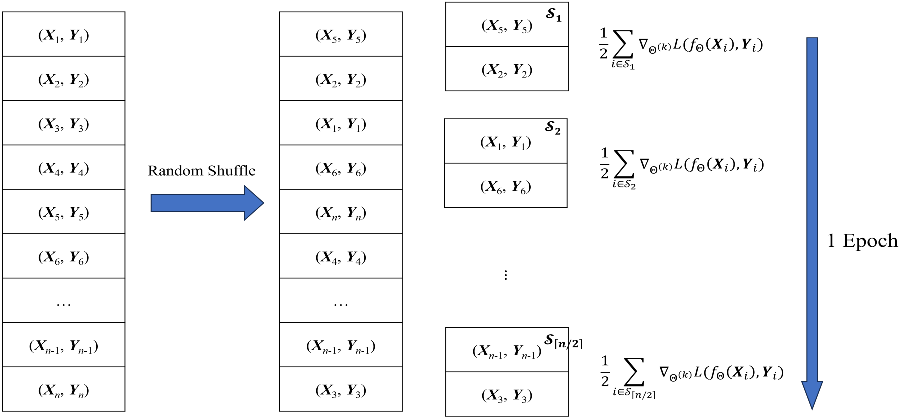
$$\begin{aligned} \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i) &= \mathbb{E}_{\mathbb{P}_n}[\nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}), \boldsymbol{Y})] \\ &= \int \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}), \boldsymbol{Y}) d\mathbb{P}_n. \end{aligned}$$

- The Idea of SGD is to replace $\mathbb{E}_{\mathbb{P}_n}[\nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}), \boldsymbol{Y})]$ by an unbiased estimator and a typical choice is

$$\frac{1}{B} \sum_{i \in \mathcal{S}_j} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i), \quad j = 1, \ldots, \lceil n/B \rceil.$$

  where $\mathcal{S}_j$ is chosen uniformly at random among the set of all subsets of size B from $\{1, \ldots, n\}$.
- In deep learning, SGD refers to the case $B = 1$. For $B > 1$, this is known as the mini-batch gradient descent.

Stochastic Gradient Descent (SGD)



Stochastic Gradient Descent (SGD)

- **Reasons for SGD: memory constraint and faster convergence**.
  - A GPU with memory size 11Gb can only process 512 samples at one time when using AlexNet for ImageNet.
  - SGD is not necessarily faster than GD if all samples can be processed in a single machine in a parallel way, but in the memory-constraint system SGD is often much faster than GD.

Stochastic Gradient Descent (SGD)

- **Reasons for SGD: memory constraint and faster convergence**.
  - A GPU with memory size 11Gb can only process 512 samples at one time when using AlexNet for ImageNet.
  - SGD is not necessarily faster than GD if all samples can be processed in a single machine in a parallel way, but in the memory-constraint system SGD is often much faster than GD.
- Convergence of SGD:
  - Under some general assumptions, convergence of SGD is guaranteed for SGD if $\eta_k = 1/k^\alpha$ for $\alpha \in (1/2, 1]$.
  - For constant step size, the gradient does not converge to zero. However, nowadays, SGD with a constant learning rate works quite well in many cases. So there is a gap between theory and applications.

GD with Momentum

- $\boldsymbol{p} = m\boldsymbol{v}$. Momentum is a measure of the amount of motion that an object has. An object with a high momentum will be harder to stop or change direction than an object with a low momentum.

GD with Momentum

- GD update: $\Theta^{(k+1)} = \Theta^{(k)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_{\Theta}(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.

GD with Momentum

- GD update: $\Theta^{(k+1)} = \Theta^{(k)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.
- GD with momentum update:
  - Momentum: $m^{(k+1)} = \beta m^{(k)} + (1 - \beta) \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i)$
  - Update: $\Theta^{(k+1)} = \Theta^{(k)} - \eta_k m^{(k)}$.

GD with Momentum

- GD update: $\Theta^{(k+1)} = \Theta^{(k)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.
- GD with momentum update:
  - Momentum: $m^{(k+1)} = \beta m^{(k)} + (1-\beta) \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i)$
  - Update: $\Theta^{(k+1)} = \Theta^{(k)} - \eta_k m^{(k)}$.
- $m^{(0)} = 0$ and
  $m^{(k+1)} = (1-\beta) \sum_{t=0}^{k} \beta^t \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k-t+1)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.

GD with Momentum

- GD update: $\Theta^{(k+1)} = \Theta^{(k)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_{\Theta}(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.
- GD with momentum update:
  - Momentum: $m^{(k+1)} = \beta m^{(k)} + (1-\beta) \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_{\Theta}(\boldsymbol{X}_i), \boldsymbol{Y}_i)$
  - Update: $\Theta^{(k+1)} = \Theta^{(k)} - \eta_k m^{(k)}$.
- $m^{(0)} = 0$ and
  $m^{(k+1)} = (1-\beta) \sum_{t=0}^{k} \beta^t \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k-t+1)}} \mathcal{L}(f_{\Theta}(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.
- A common choice for β in practice is 0.9.

GD with Momentum

- GD update: $\Theta^{(k+1)} = \Theta^{(k)} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_{\Theta}(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.
- GD with momentum update:
  - Momentum: $m^{(k+1)} = \beta m^{(k)} + (1-\beta) \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_{\Theta}(\boldsymbol{X}_i), \boldsymbol{Y}_i)$
  - Update: $\Theta^{(k+1)} = \Theta^{(k)} - \eta_k m^{(k)}$.
- $m^{(0)} = 0$ and
  $m^{(k+1)} = (1-\beta) \sum_{t=0}^{k} \beta^t \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k-t+1)}} \mathcal{L}(f_{\Theta}(\boldsymbol{X}_i), \boldsymbol{Y}_i)$.
- A common choice for β in practice is 0.9.
- GD with momentum almost always works faster than GD. However, this is not true for the naive stochastic version.

Adaptive Gradient Method (AdaGrad), Duchi et al, 2011

- At the k-the iteration, update the parameter as

$$\Theta^{(k+1)} = \Theta^{(k)} - \eta_k \boldsymbol{G}^{(k)^{-1/2}} \boldsymbol{g}^{(k)}, \quad k = 0, 1, 2, ...,$$

where

$$\boldsymbol{g}^{(k)} = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_{\Theta}(\boldsymbol{X}_i), \boldsymbol{Y}_i)$$

$$\boldsymbol{G}^{(k)} = \sum_{j=1}^{k} \boldsymbol{g}^{(j)} \boldsymbol{g}^{(j)^T}.$$

Adaptive Gradient Method (AdaGrad), Duchi et al, 2011

- At the k-the iteration, update the parameter as

$$\Theta^{(k+1)} = \Theta^{(k)} - \eta_k \boldsymbol{G}^{(k)^{-1/2}} \boldsymbol{g}^{(k)}, \quad k = 0, 1, 2, ...,$$

  where

$$\boldsymbol{g}^{(k)} = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i)$$

$$\boldsymbol{G}^{(k)} = \sum_{j=1}^{k} \boldsymbol{g}^{(j)} \boldsymbol{g}^{(j)^T}.$$

- Caculating $\boldsymbol{G}^{(k)^{-1/2}}$ may be hard. In practice, using $\mathrm{diag}(\boldsymbol{G}^{(k)})^{-1/2}$ works well enough.

Adaptive Gradient Method (AdaGrad), Duchi et al, 2011

- At the k-the iteration, update the parameter as

$$\Theta^{(k+1)} = \Theta^{(k)} - \eta_k \boldsymbol{G}^{(k)^{-1/2}} \boldsymbol{g}^{(k)}, \quad k = 0, 1, 2, \ldots,$$

where

$$\boldsymbol{g}^{(k)} = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\Theta^{(k)}} \mathcal{L}(f_\Theta(\boldsymbol{X}_i), \boldsymbol{Y}_i)$$

$$\boldsymbol{G}^{(k)} = \sum_{j=1}^{k} \boldsymbol{g}^{(j)} \boldsymbol{g}^{(j)^T}.$$

- Caculating $\boldsymbol{G}^{(k)^{-1/2}}$ may be hard. In practice, using $\mathrm{diag}(\boldsymbol{G}^{(k)})^{-1/2}$ works well enough.
- AdaGrad is shown to exhibit a convergence rate similar to SGD for convex problems and non-convex problems. After T iterations, the error is of the order $1/\sqrt{T}$.

Root Mean Squre Propagation (RMSProp), Tieleman & Hinton, 2012

- One drawback of AdaGrad is that it treats all past gradients equally, and it is natural to use exponentially decaying weights for the bast gradients.

Root Mean Squre Propagation (RMSProp), Tieleman & Hinton, 2012

- One drawback of AdaGrad is that it treats all past gradients equally, and it is natural to use exponentially decaying weights for the bast gradients.
- RMSProp update:

$$\boldsymbol{G}^{(k)} = \beta \boldsymbol{G}^{(k-1)} + (1-\beta)\boldsymbol{g}^{(k)}{\boldsymbol{g}^{(k)}}^T$$
$$\Theta^{(k+1)} = \Theta^{(k)} - \eta_k \mathrm{diag}(\boldsymbol{G}^{(k)})^{-1/2}\boldsymbol{g}^{(k)}.$$

Root Mean Squre Propagation (RMSProp), Tieleman & Hinton, 2012

- One drawback of AdaGrad is that it treats all past gradients equally, and it is natural to use exponentially decaying weights for the bast gradients.
- RMSProp update:

$$\boldsymbol{G}^{(k)} = \beta \boldsymbol{G}^{(k-1)} + (1-\beta)\boldsymbol{g}^{(k)}{\boldsymbol{g}^{(k)}}^T$$
$$\Theta^{(k+1)} = \Theta^{(k)} - \eta_k \mathrm{diag}(\boldsymbol{G}^{(k)})^{-1/2}\boldsymbol{g}^{(k)}.$$

- In deep learning, a small number $\epsilon = 10^{-8}$ is often added to each component in $\mathrm{diag}(\boldsymbol{G}^{(k)})$ to reduce numerical instability.

Root Mean Squre Propagation (RMSProp), Tieleman & Hinton, 2012

Theorem

Assume that the empirical risk function is gradient Lipshitz continuous and lower bounded by R^*. Then RMSProp with diminishing step size $\eta_k = \eta_1/\sqrt{k}$ and any $\beta \in (0, 1)$,

$$\min_{k\in(1,T]} \left\|\boldsymbol{g}^{(k)}\right\|_1 \leq \mathcal{O}\left(\frac{\log T}{\sqrt{T}}\right),$$

where $T > 0$ us the total iteration number.

Adaptive Moment Estimation (ADAM), Kingma & Ba, 2017

- The paper "ADAM: A Method for Stochastic Optimization" has been cited 160,205 times based on Google Scholar despite that there is an error in the proof of the paper.

Adaptive Moment Estimation (ADAM), Kingma & Ba, 2017

- The paper "ADAM: A Method for Stochastic Optimization" has been cited 160,205 times based on Google Scholar despite that there is an error in the proof of the paper.
- ADAM is the combination of RMSProp and the gradient descent with momentum. Here is the ADAM update

$$
\begin{aligned}
\text{(Momentum)}: \quad & m^{(k)} = \beta_1 m^{(k-1)} + (1-\beta_1)\boldsymbol{g}^{(k)} \\
\text{(RMSProp)}: \quad & \boldsymbol{G}^{(k)} = \beta_2 \boldsymbol{G}^{(k-1)} + (1-\beta_2)\boldsymbol{g}^{(k)}\boldsymbol{g}^{(k)^T} \\
\text{(Bias Correction)} \quad & \hat{m}^{(k)} = m^{(k)}/(1-\beta_1^k) \\
\text{(Bias Correction)} \quad & \hat{\boldsymbol{G}}^{(k)} = \boldsymbol{G}^{(k)}/(1-\beta_2^k) \\
\text{(Update)} \quad & \Theta^{(k+1)} = \Theta^{(k)} - \eta \mathrm{diag}(\hat{\boldsymbol{G}}^{(k)})^{-1/2}\hat{m}^{(k)}
\end{aligned}
$$

Adaptive Moment Estimation (ADAM), Kingma & Ba, 2017

- The paper "ADAM: A Method for Stochastic Optimization" has been cited 160,205 times based on Google Scholar despite that there is an error in the proof of the paper.
- ADAM is the combination of RMSProp and the gradient descent with momentum. Here is the ADAM update

$$
\begin{aligned}
\text{(Momentum)}: \quad & m^{(k)} = \beta_1 m^{(k-1)} + (1-\beta_1)\boldsymbol{g}^{(k)} \\
\text{(RMSProp)}: \quad & \boldsymbol{G}^{(k)} = \beta_2 \boldsymbol{G}^{(k-1)} + (1-\beta_2)\boldsymbol{g}^{(k)}\boldsymbol{g}^{(k)^T} \\
\text{(Bias Correction)} \quad & \hat{m}^{(k)} = m^{(k)}/(1-\beta_1^k) \\
\text{(Bias Correction)} \quad & \hat{\boldsymbol{G}}^{(k)} = \boldsymbol{G}^{(k)}/(1-\beta_2^k) \\
\text{(Update)} \quad & \Theta^{(k+1)} = \Theta^{(k)} - \eta \mathrm{diag}(\hat{\boldsymbol{G}}^{(k)})^{-1/2}\hat{m}^{(k)}
\end{aligned}
$$

- Common choices of β_1 and β_2 are 0.9 and 0.99 resp. in practice.

Adaptive Moment Estimation (ADAM), Kingma & Ba, 2017

Theorem

Assume that the empirical risk function is gradient Lipshitz continuous and lower bounded by R^*. Then ADAM with diminishing step size $\eta_k = \eta_1/\sqrt{k}$ and any $\beta_1 < \sqrt{\beta_2} < 1$,

$$\min_{k\in(1,T]} \left\|\boldsymbol{g}^{(k)}\right\|_1 \leq \mathcal{O}\left(\frac{\log T}{\sqrt{T}}\right),$$

where $T > 0$ us the total iteration number.

Numerical Experiment

- Linear regression
- Find local minimum

The experiments are conducted on Python 3.9.16, we developed our own algorithms instead of using an existing library/Package.

Numerical Experiment : Linear Regression

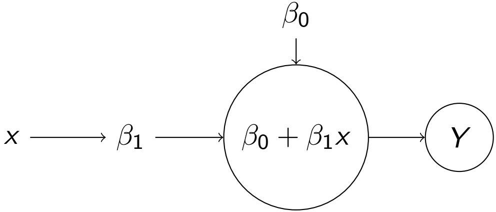
Given

$$
\begin{aligned}
& X \sim \mathcal{N}(0,1) \quad \varepsilon \sim \mathcal{N}(0,1) \\
& Y = \beta_0 + \beta_1 x + \varepsilon \\
& \text{Loss} = \frac{1}{n} \sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 x_i))^2
\end{aligned}
$$

And the gradient of the loss function respect to β_0 and β_1 will be

$$
\begin{aligned}
\frac{\partial}{\partial \beta_0} \text{Loss} &= -\frac{2}{n} \sum_{i=1}^{n} Y_i - (\beta_0 + \beta_1 x_i) \\
\frac{\partial}{\partial \beta_1} \text{Loss} &= -\frac{2}{n} \sum_{i=1}^{n} Y_i - (\beta_0 + \beta_1 x_i) x_i
\end{aligned}
$$

We generate the (x_i, Y_i), for all $i \in [n]$ with $\beta_0 = 2$ and $\beta_1 = 3$.


β_0
x
β_1
$\beta_0 + \beta_1 x$
Y

$$
\begin{tikzcd}
 & & \beta_0 \arrow[d] & \\
x \arrow[r] & \beta_1 \arrow[r] & \beta_0 + \beta_1 x \arrow[r] & Y
\end{tikzcd}
$$

$\boxed{\text{Now we switch to PPT}}$.

Numerical Experiment : local/global minimum

Use function

$$z = f(x, y) = \frac{1}{2}x^2 + \frac{1}{4}y^4 - \frac{1}{2}y^2$$

Numerical Experiment : local/global minimum

Use function

$$z = f(x, y) = \frac{1}{2}x^2 + \frac{1}{4}y^4 - \frac{1}{2}y^2$$

By using "Second partial derivative test", critical points are $(0,0), (0,1), (0,-1)$ and local minimum at $(0,1)$ and $(0,-1)$, saddle point at $(0,0)$.

Numerical Experiment : local/global minimum

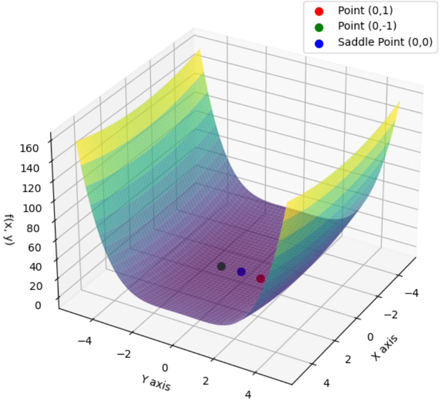
Use function

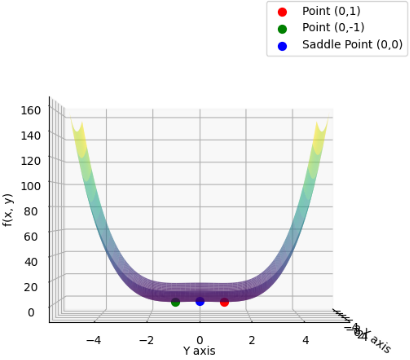
$$z = f(x, y) = \frac{1}{2}x^2 + \frac{1}{4}y^4 - \frac{1}{2}y^2$$

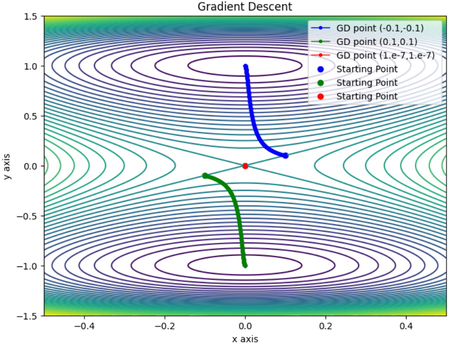
By using "Second partial derivative test", critical points are $(0,0), (0,1), (0,-1)$ and local minimum at $(0,1)$ and $(0,-1)$, saddle point at $(0,0)$. We will use the point nearby the saddle point as initial value such as $(0.1, 0.1)$, $(-0.1, -0.1)$ and $(10^{-7}, 10^{-7})$, and

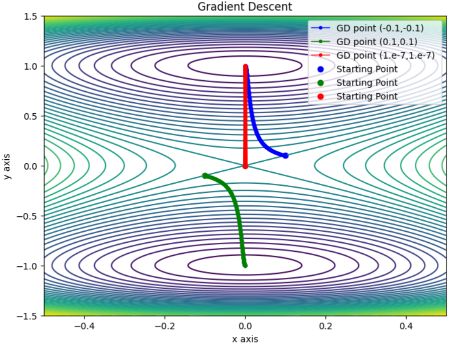
$$\nabla z = \begin{bmatrix} x \\ y^3 - y \end{bmatrix}$$

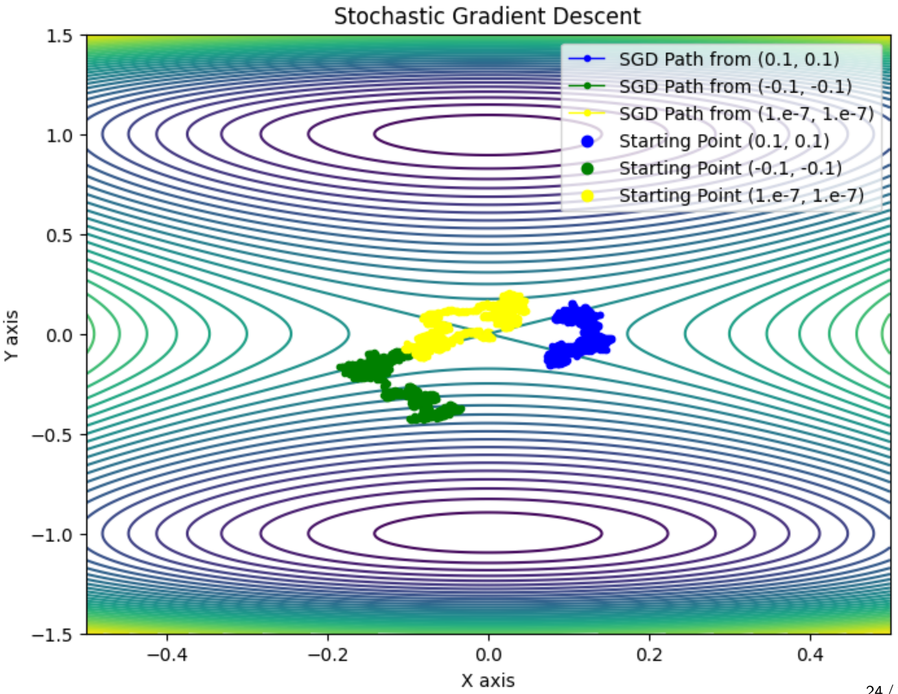
Function Overview

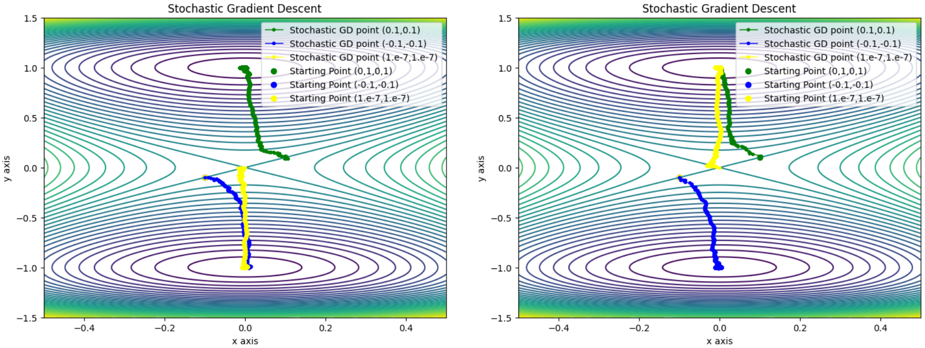


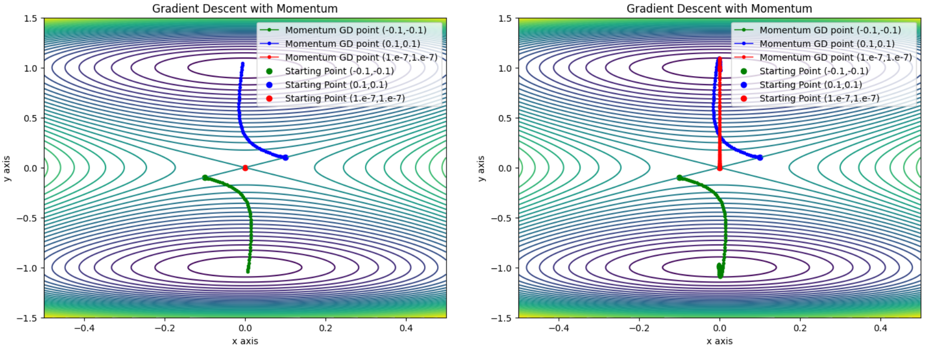


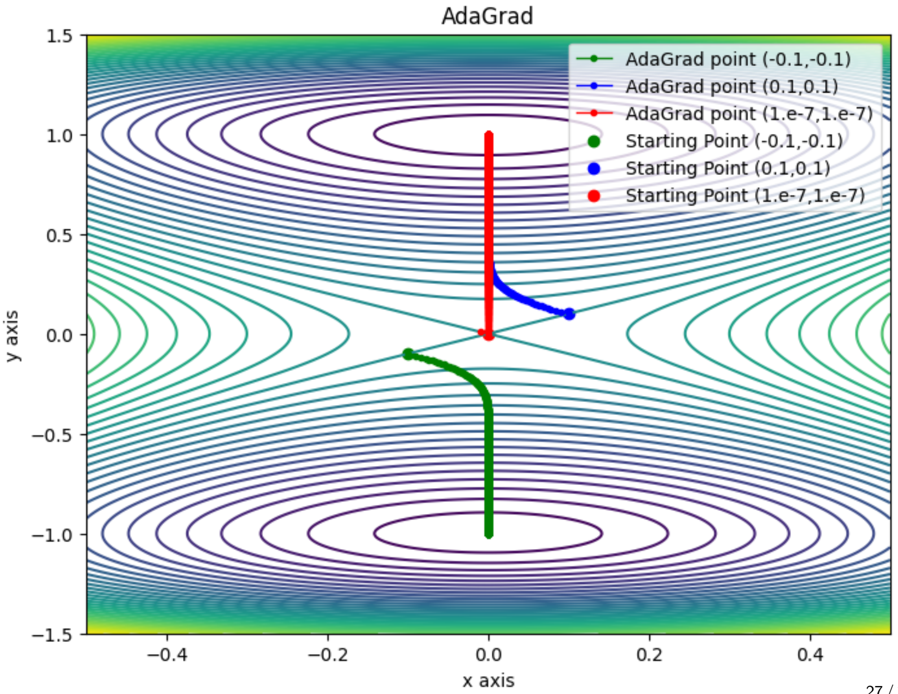

Gradient Descent
GD point (-0.1,-0.1)
GD point (0.1,0.1)
GD point (1.e-7,1.e-7)
Starting Point
Starting Point
Starting Point
y axis
x axis
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−0.4
−0.2
0.0
0.2
0.4


Gradient Descent
GD point (-0.1,-0.1)
GD point (0.1,0.1)
GD point (1.e-7,1.e-7)
Starting Point
Starting Point
Starting Point
y axis
x axis
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−0.4
−0.2
0.0
0.2
0.4

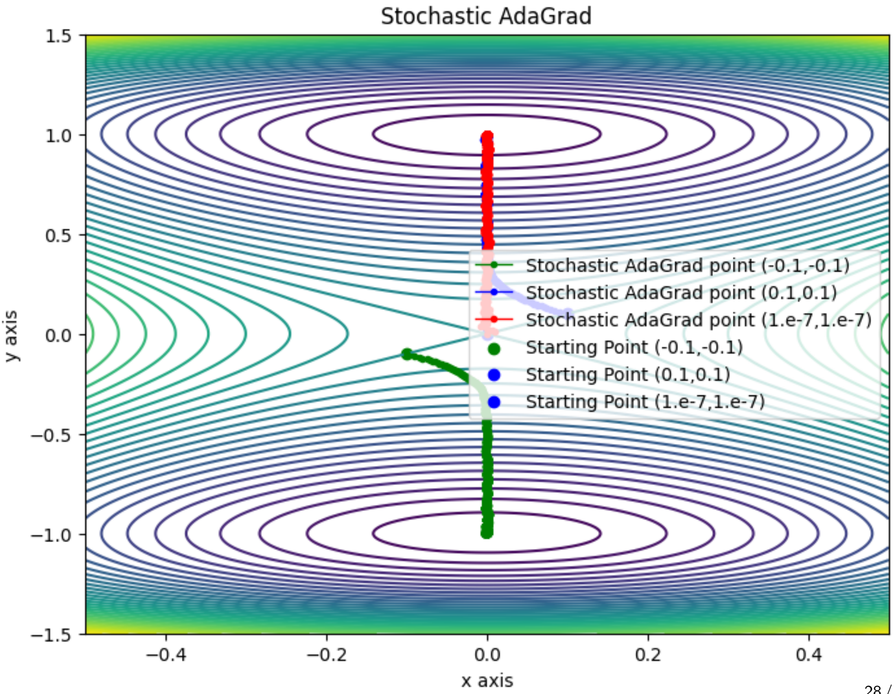

Stochastic Gradient Descent
SGD Path from (0.1, 0.1)
SGD Path from (-0.1, -0.1)
SGD Path from (1.e-7, 1.e-7)
Starting Point (0.1, 0.1)
Starting Point (-0.1, -0.1)
Starting Point (1.e-7, 1.e-7)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−0.4
−0.2
0.0
0.2
0.4
Y axis
X axis

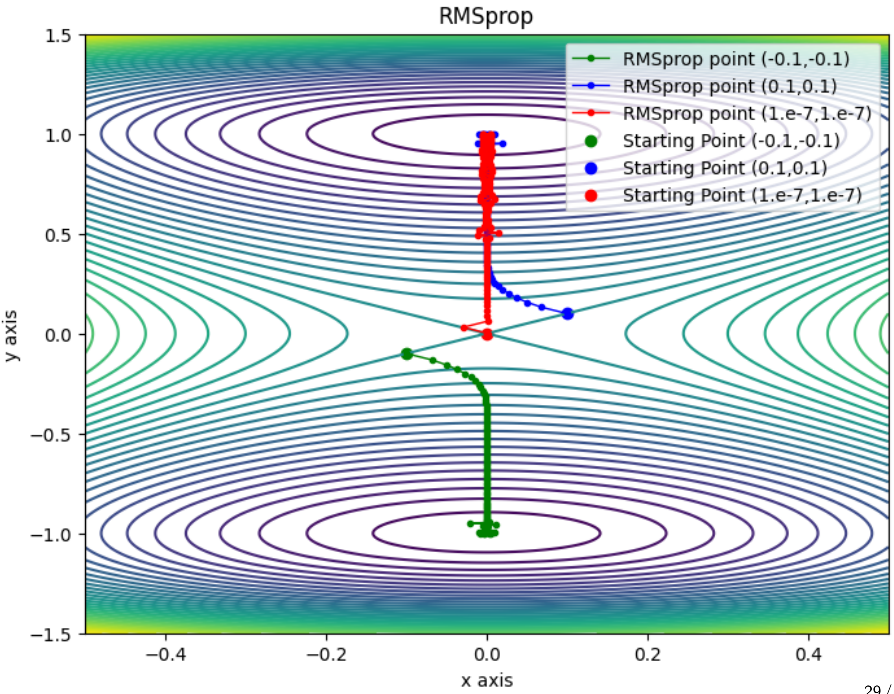

Stochastic Gradient Descent
Stochastic GD point (0.1,0.1)
Stochastic GD point (-0.1,-0.1)
Stochastic GD point (1.e-7,1.e-7)
Starting Point (0,1,0,1)
Starting Point (-0.1,-0.1)
Starting Point (1.e-7,1.e-7)
y axis
x axis
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−0.4
−0.2
0.0
0.2
0.4
Stochastic Gradient Descent
Stochastic GD point (0.1,0.1)
Stochastic GD point (-0.1,-0.1)
Stochastic GD point (1.e-7,1.e-7)
Starting Point (0,1,0,1)
Starting Point (-0.1,-0.1)
Starting Point (1.e-7,1.e-7)
y axis
x axis

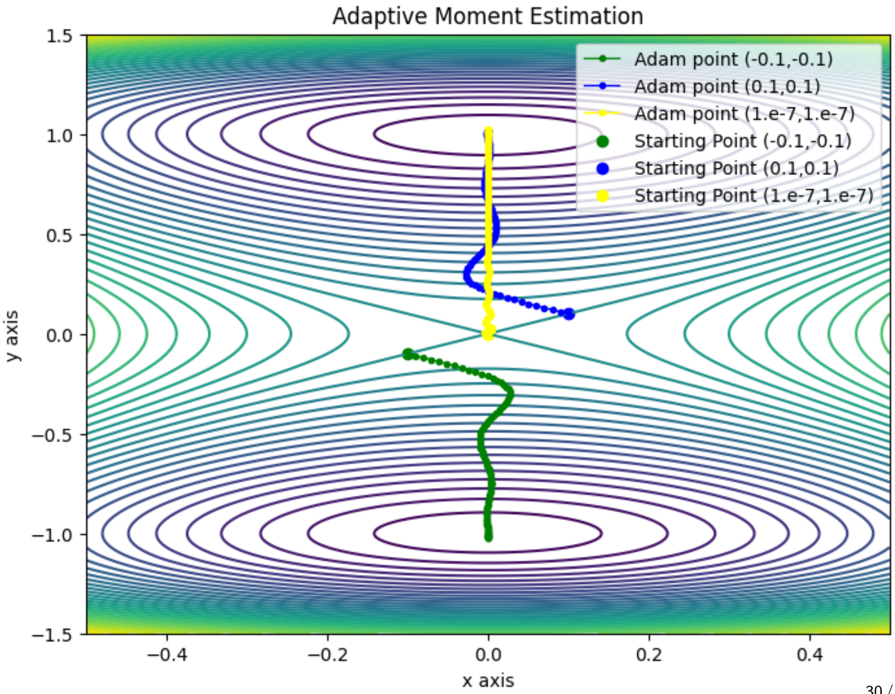

Gradient Descent with Momentum
Momentum GD point (-0.1,-0.1)
Momentum GD point (0.1,0.1)
Momentum GD point (1.e-7,1.e-7)
Starting Point (-0.1,-0.1)
Starting Point (0.1,0.1)
Starting Point (1.e-7,1.e-7)
x axis
y axis
Gradient Descent with Momentum
Momentum GD point (-0.1,-0.1)
Momentum GD point (0.1,0.1)
Momentum GD point (1.e-7,1.e-7)
Starting Point (-0.1,-0.1)
Starting Point (0.1,0.1)
Starting Point (1.e-7,1.e-7)
x axis
y axis



AdaGrad
AdaGrad point (-0.1,-0.1)
AdaGrad point (0.1,0.1)
AdaGrad point (1.e-7,1.e-7)
Starting Point (-0.1,-0.1)
Starting Point (0.1,0.1)
Starting Point (1.e-7,1.e-7)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y axis
−0.4
−0.2
0.0
0.2
0.4
x axis


Stochastic AdaGrad
Stochastic AdaGrad point (-0.1,-0.1)
Stochastic AdaGrad point (0.1,0.1)
Stochastic AdaGrad point (1.e-7,1.e-7)
Starting Point (-0.1,-0.1)
Starting Point (0.1,0.1)
Starting Point (1.e-7,1.e-7)
y axis
x axis
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−0.4
−0.2
0.0
0.2
0.4


RMSprop
RMSprop point (-0.1,-0.1)
RMSprop point (0.1,0.1)
RMSprop point (1.e-7,1.e-7)
Starting Point (-0.1,-0.1)
Starting Point (0.1,0.1)
Starting Point (1.e-7,1.e-7)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−0.4
−0.2
0.0
0.2
0.4
x axis
y axis


Adaptive Moment Estimation
Adam point (-0.1,-0.1)
Adam point (0.1,0.1)
Adam point (1.e-7,1.e-7)
Starting Point (-0.1,-0.1)
Starting Point (0.1,0.1)
Starting Point (1.e-7,1.e-7)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y axis
−0.4
−0.2
0.0
0.2
0.4
x axis

References

1 Lee, Jason D., et al. "Gradient descent only converges to minimizers." Conference on learning theory. PMLR, 2016.

2 Du, Simon S., et al. "Gradient descent can take exponential time to escape saddle points." Advances in neural information processing systems 30 (2017).

3 Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of machine learning research 12.7 (2011).

4 Tieleman, Tijmen and Hinton, Geoffrey. "Lecture 6.5-rmsprop: Divede the gradient by a running average of its recent magnitude." COURSERA: Neural networks for machine learning, 4(2):26-31, 2012

5 Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

References

6 Sun, Ruoyu. "Optimization for deep learning: theory and algorithms." arXiv preprint arXiv:1912.08957 (2019).

7 Shi, Naichen, et al. "RMSprop converges with proper hyper-parameter." International Conference on Learning Representations. 2020.

8 Staib, Matthew, et al. "Escaping saddle points with adaptive gradient methods." International Conference on Machine Learning. PMLR, 2019.

Thank you!