Homework 2-report

Ma Xiaoqi
308932

Consider the function $f: R^2 \to R$ where $f(x,y) = xy$

1. For function $f(x,y) = xy$

$\frac{\partial f(x,y)}{\partial x} = y \quad \frac{\partial f(x,y)}{\partial y} = x$ Let $\nabla f(x,y) = 0$

so the first order stationary point is $(0,0)$

The Hessian Matrix of the function:

$H = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. the engivalue of the Matrix H is $1, -1$

So H is indefinite matrix

So $(0,0)$ is saddle point.

2. When $(x^*, y^*) = (0,0)$; $f(x^*, y^*) = 0$

① $f(x^*, y) = f(x^*, y^*) = 0 = f(y, x^*) = 0$

② Then for $f(x^*, y^*) \geq f(x^*, y)$ We need prove $x^* y^* \geq x^* y$ for all y

$x^*(y^* - y) \geq 0$

proof by contradiction: if $x^* > 0$ $y < y^*$, It's not for all y

if $x^* < 0$, $y > y^*$, It's not for all y. So $x^* = 0$

Similarly, $y^* = 0$; So $(x^*, y^*) = (0,0)$ is the solution

Actually, we can rotate the coordinate system to fix problem

Try to rotate $45°$ of system by multiple matrix $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

$\therefore \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2}x - \frac{\sqrt{2}}{2}y \\ \frac{\sqrt{2}}{2}x + \frac{\sqrt{2}}{2}y \end{bmatrix}$ (new coordinates)

(old coordinate)

So the function in new ordinate system: $f^* = \frac{1}{2}x^2 - \frac{1}{2}y^2$

the saddle point is $(0,0) = (x^*, y^*)$

$\therefore f(x^*, y^*) = 0 \geq -\frac{1}{2}y^2 = f(x^*, y)$ It's $\min_x \max_y f(x,y)$

$f(x^*, y^*) = 0 \leq \frac{1}{2}x^2 = f(x, y^*)$

3. (a) proof by contradiction:

We suppose there exist limit of $\{x_k, y_k\}_{k=0}^{\infty}$

That means $\lim_{k\to\infty} x_{k+1} = x_k \quad \lim_{k\to\infty} y_{k+1} = y_k$

$\therefore x_k - \gamma \nabla_x f(x_k, y_k) = x_k - \gamma y_k = x_k$

$\because \gamma > 0 \quad \therefore y_k = 0$ Similarly, $x_k = 0 \quad (x_k, y_k) = (0,0)$

~~However, $(x_0, y_0) \neq (0,0)$~~

$\because \begin{cases} x_k = x_{k-1} - \gamma y_{k-1} \\ y_k = y_{k-1} + \gamma x_{k+1} \end{cases} \Rightarrow y_{k-1} = -\gamma^2 y_{k-1}$

if $y_{k-1} \neq 0$ $-\gamma^2 = 1$ It is impossible so $(x_{k-1}, y_{k-1}) = (0,0)$

However, $(x_0, y_0) \neq (0, 0)$ is the condition
So the sequence of iterates diverges

(b): Consider the distance of $\{x_k, y_k\}$ in the space

$$\sqrt{(x_{k+1})^2 + (y_{k+1})^2} = \sqrt{(x_k^2 - 2\gamma x_k y_k + \gamma^2 y_k^2) + (y_k^2 + 2\gamma x_k y_k + \gamma^2 x_k^2}$$

$$= \sqrt{1+\gamma^2}\ \sqrt{x_k^2 + y_k^2}$$

So the rate is $\sqrt{1+\gamma^2}$

Consider the GAN: $G := \{g : g(z) = Wz + b\}$

$F := \{f : f(x) = v^T x\}$

[1] $F := \{f(x) = v^T x\}$

$$\|f(x) - f(y)\| = \|v^T x - v^T y\| = \|v^T (x-y)\| \leq \|v^T\| \|x - y\|$$

So the Lipschitz constant is $\|v^T\|$ $\quad \vec{v} = (v_1, v_2)$

$\vec{v} \in R^2 \quad \therefore \|(v_1, v_2)\|_2^2 = v_1^2 + v_2^2$

The set of function in F whose Lipschitz constant is at most 1 means the the norm ~~value~~ value of the gradient of the function is at most 1

[F] Given two distributions μ, ν

$F := \{f : f(x) = v^T x\}$

$$\max_{f \in F} E_{x \sim \mu}[f(x)] - E_{Y \sim \nu}[f(Y)]$$

$$= \max_{f \in F} f[E_{x \sim \mu}(x)] - f[E_{Y \sim \mu}(Y)]$$

$$= \max_{f \in F} f[E_{x \sim \mu}(x) - E_{Y \sim \mu}(Y)]$$

Let $E_{x \sim \mu}(x) - E_{Y \sim \mu}(Y) = \vec{c}$

$\therefore \max_{f \in F} f[E_{x \sim \mu}(x) - E_{Y \sim \mu}(Y)] = \vec{c}^T c \geq 0$ (let $\vec{c} = \vec{v}$)

Another:

As we enforce the 1-Lipschitz constraint

$$\max_{\|F\|_L \le 1} E_{x\sim\mu}[f(x)] - E_{Y\sim\nu}[f(Y)] \quad (1)$$

According to the duality

(1) equals to:

$$\min_{\gamma\sim\Pi(\mu,\nu)} E_{(x,y)\sim\gamma}[\|x-y\|]$$

$\gamma\sim\Pi(\mu,\nu)$ is the union distribution of μ, ν

$\therefore \|x-y\| \ge 0$

$\therefore \max_{\|F\|_L \le 1} E_{x\sim\mu}[f(x)] - E_{Y\sim\nu}[f(Y)] \ge 0$

5 $\vec{x}$ is a vector $\in R^2$ $\vec{x} = (x_1, x_2)$

The expected value of the vector $\vec{x}$:

$$E[\vec{x}] = \begin{pmatrix} E[x_1] \\ E[x_2] \end{pmatrix}$$

$\because E_{(x_1,x_2)\sim\mu}[x_1] = E_{(x_1,x_2)\sim\nu}[x_1]$

$E_{(x_1,x_2)\sim\mu}[x_2] = E_{(x_1,x_2)\sim\nu}[x_2]$

$$\therefore E_{\vec{x}\sim\mu}[\vec{x}] = \begin{pmatrix} E_{\sim\mu}[x_1] \\ E_{\sim\mu}[x_2] \end{pmatrix} = \begin{bmatrix} E_{\sim\nu}[x_1] \\ E_{\sim\nu}[x_2] \end{bmatrix} = E_{\vec{x}\sim\nu}[\vec{x}]$$

$$\therefore \max E_{\vec{x}\sim\mu}[f(\vec{x})] - E_{\vec{Y}\sim\nu}[f(\vec{Y})]$$

$$= \max f(E_{\vec{x}\sim\mu}[\vec{x}] - E_{\vec{Y}\sim\nu}[\vec{Y}])$$

$$= 0$$

According to the GAN examples: we set the true mean and noise mean equals to [0., 0.] and the true covariance and noise covariance is different. So the "red" distribution always circles around the "blue" distribution.

1 Minimax problems and GANs

Comments about GANs

Compared with the traditional GAN which used KL divergence to described the difference of different distributions, this experiment used another distance to describe the difference of true sample and gen sample.

Then, after using the different stochastic gradient methods: simultaneous and alternating to update the gradient(the GIF and parts of picture are in the file folder), I observed the difference of the two SG methods by the motion of data points sequence, the alternating SG is better than simultaneous SG because the trajectory of the data points(red distribution) in alternating SG is more concentrated(stable) and closer to the blue section than that in the simultaneous SG after some iterations.

We focus on the update of g_{k+1} ,the simultaneous method updates according to the $f_k$andg_k,but the alternating method updates according to the f_{k+1}andg_k, the generated points considering present value of f_{k+1}performs better because function f plays the role as classifier. In the alternating method, the generated data points obtain more information about the 'true' or 'false', so the data points are more close to the true distribution.

2.Optimizers of Neural Network

In the Experiment of optimizers of neural network, I used the MNIST dataset and different optimizers to train the neural network. To show the difference of these optimizers, I used the step-size 0.5, 1e-2, 1e-5 to train each optimizer with 15 epochs.

Firstly, give the plot of training loss and training accuracy of five optimizers at different step-size(after the comments)

Comments on optimizers:

1. Vanilla Minibatch SGD

At the largest step-size 0.5, SGD performs the best with the high training accuracy and faster convergence rate of loss. However, at the smallest step-size 1e-5, it performs worst after 15 epochs, which may be caused by inadequate epochs, so I try to set the epochs as 100 and observe that the efficiency of the SGD optimizers improves.

2.Minibatch SGD with Momentum

At the largest step-size 0.5, SGD with momentum performs badly with the decreasing accuracy and unchanged loss, but it performs well at 1e-2. In my opinion, compare with SGD, SGD with momentum increased stability per update because it is based on the momentum of the previous update, so large step-size may cause less opportunity to search more different directions, then make the loss unchanged.

3 AdaGrad

AdaGrad changes the learning rate at each update(adaptive learning rate) according to the previous of occurrence of parameters, it performs better than SGD, but it use all the history

gradients to update the learning rate, it may convergence earlier especially compared with RMSProp.

4 RMSProp

Based on the Adagrad, RMSProp just bases the part of history accumulated gradient, It performs worst in 0.5 may because trap into the local minimum. It performs best in 1e-5 compared with other step-sizes. .

5 Adam

Combined with the method of momentum and adaptive learning rate, Adam performs than other optimizers in 1e-2,1e-5.

Overall, when we choose the appropriate step-size(not too big, not too small), we find the adaptive learning rate method is better than SGD and SGD with momentum. For the above five optimizers, Adam is the best which combined the momentum and adaptive learning rate.

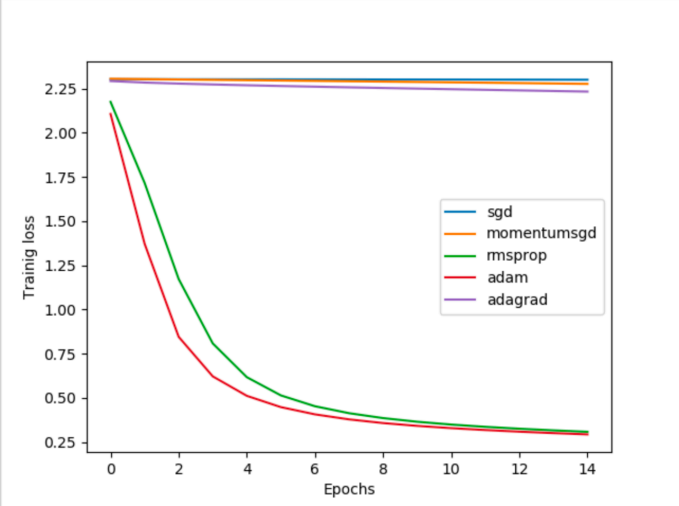

Fig2.1 training loss (step-size: 10^-5 epochs:15)

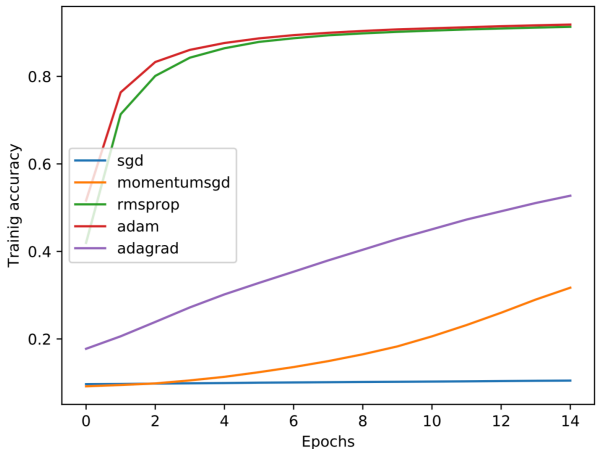

Fig2.2training accuracy (step-size: 10^-5 epochs:15)

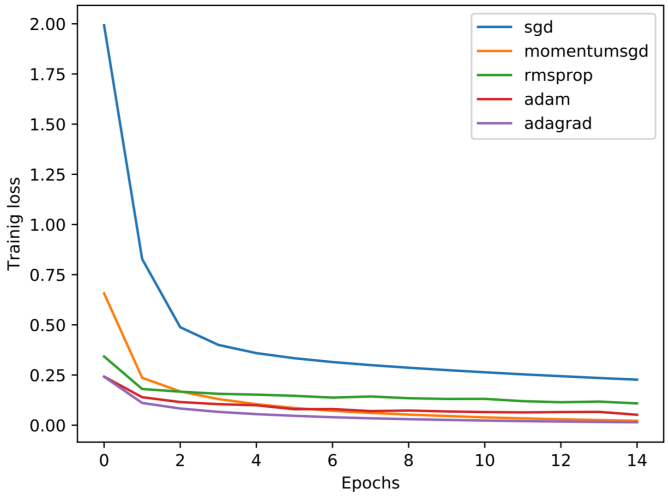

Fig2.3 training loss (step-size: 10^-2 epochs:15)

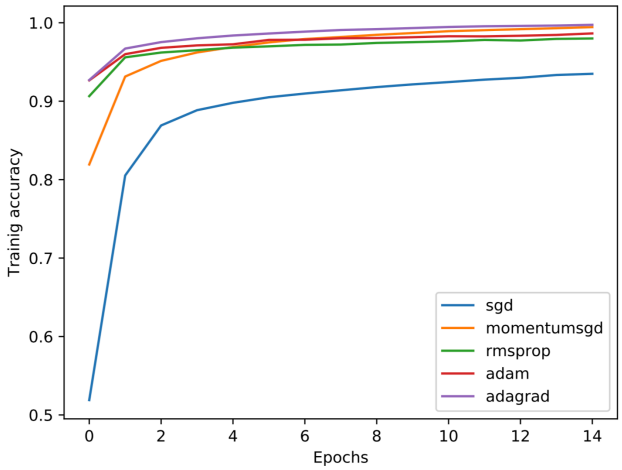

Fig2.4training accuracy (step-size: 10^-2 epochs:15)

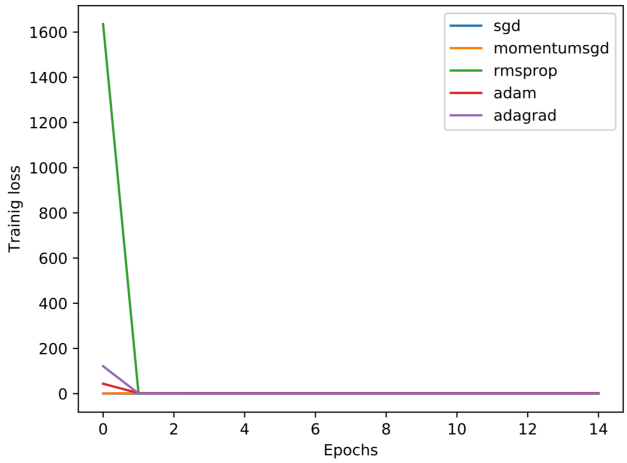

Fig2.5 training loss (step-size: 0.5epochs:15)

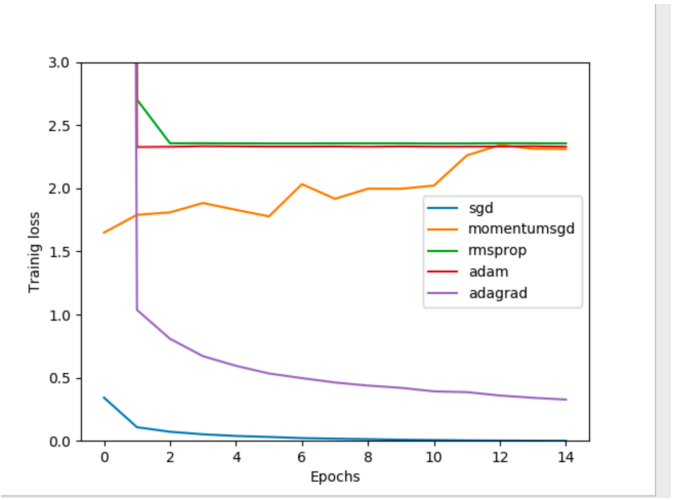

Fig2.5(1) training loss (step-size: 0.5 epochs:15)

Separate the optimizers by changing scale of y

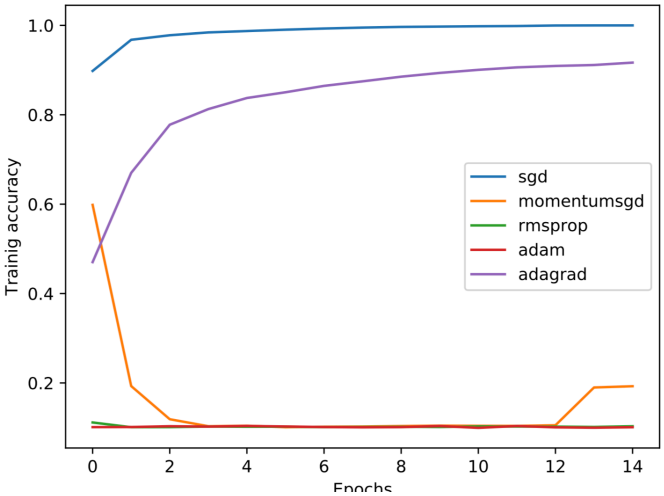

Fig2.6training accuracy (step-size: 0.5 epochs:15)