HW 2

Part 1

1.1

1.1.1

Take the gradient w.r.t. f and setting it to zero,

$$\nabla_x f = y - 1 = 0, \nabla_y f = x - 1$$

we get the stationary point is $(1, 1)$. Since the Hessian is $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Since $D(x, y) = \det \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = -1$, according to Second partial derivative test, it is a saddle point.

1.1.2

Define $g(x) = max_y f(x, y)$, we would have $g(x) = 0$ if $x = 1$; $g(x) = +\infty$ if $x \neq 1$. Hence, $g(x)$ attains its minimum 0 when $x = 1$. Hence, $(x^\star, y^\star) = (1, 1)$ is the solution to $\min_x \max_y f(x, y)$.

Besides, it's clear that $f(x^\star, y^\star) = f(x^\star, y) = f(x, y^\star) = 0$. Hence, it satisfies saddle point inequality $f(x^\star, y) \leq f(x, y) \leq f(x, y^\star)$. It is thus the solution to the min-max problem.

1.1.3

We can rewrite the gradient decent:

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

i.e.

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & -\gamma \\ \gamma & 1 \end{pmatrix} \left[\begin{pmatrix} x_k \\ y_k \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right]$$

Hence

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

Taking the square and sum it up

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

i.e.

$$(x_k - 1)^2 + (y_k - 1)^2 = (1 + \gamma^2)^k((x_0 - 1)^2 + (y_0 - 1)^2)$$

which tends to ∞ as k tends to ∞ if $(x_0, y_0) \neq (1, 1)$. And the rate of distance is $\alpha = \sqrt{1 + \gamma^2}$, where

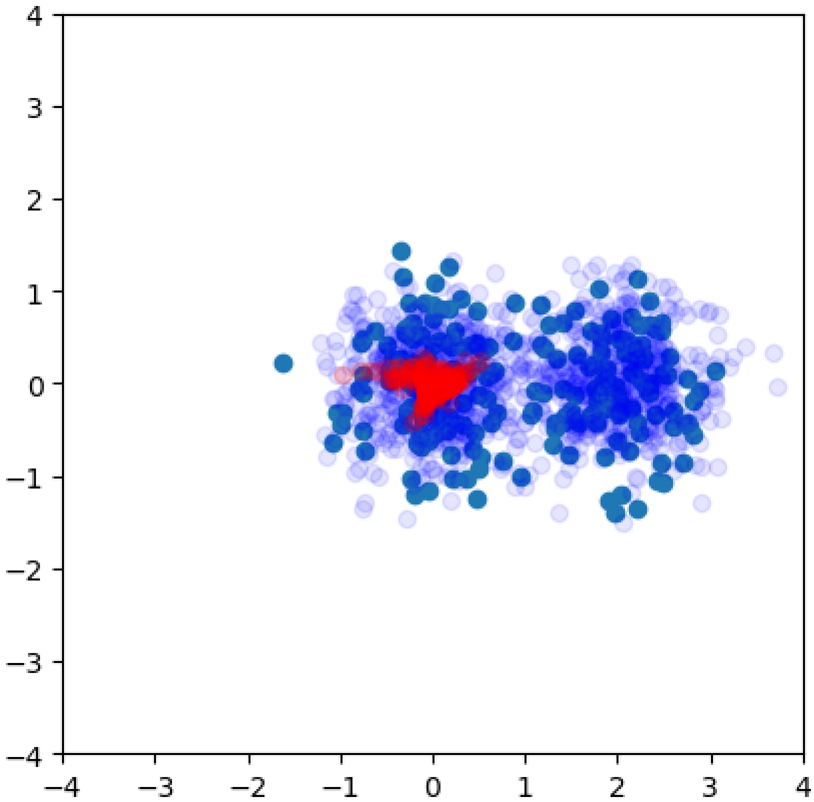
$$\sqrt{(x_{k+1} - 1)^2 + (y_{k+1} - 1)^2} = \alpha^k \sqrt{(x_k - 1)^2 + (y_k - 1)^2}$$

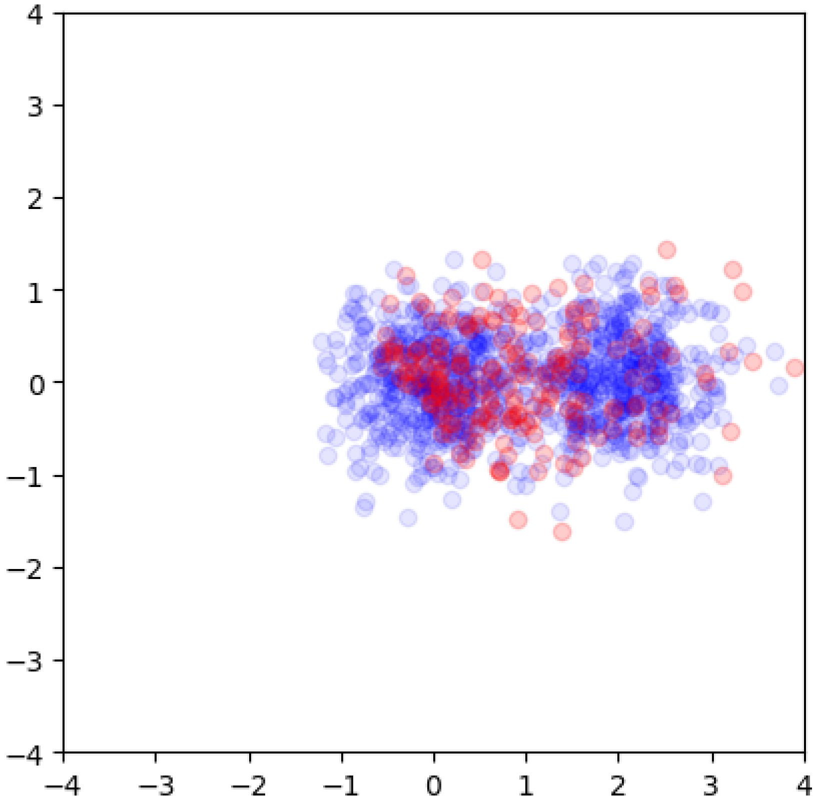
1.2.2

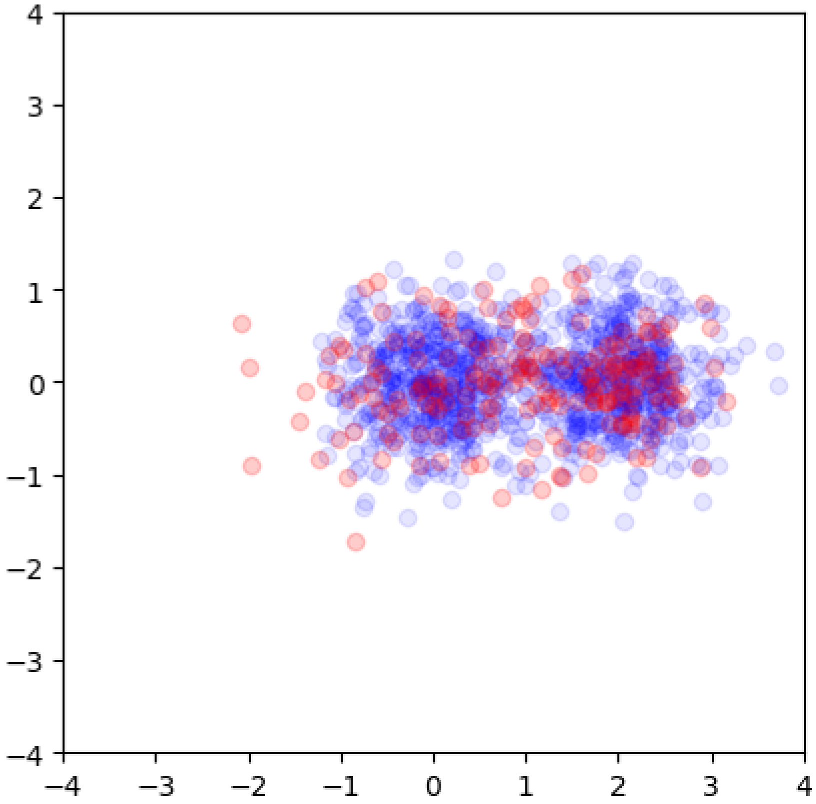
In order to visualize the effects of GAN better, I modify the number of steps to be 3000. The results are in code/gan/figs/

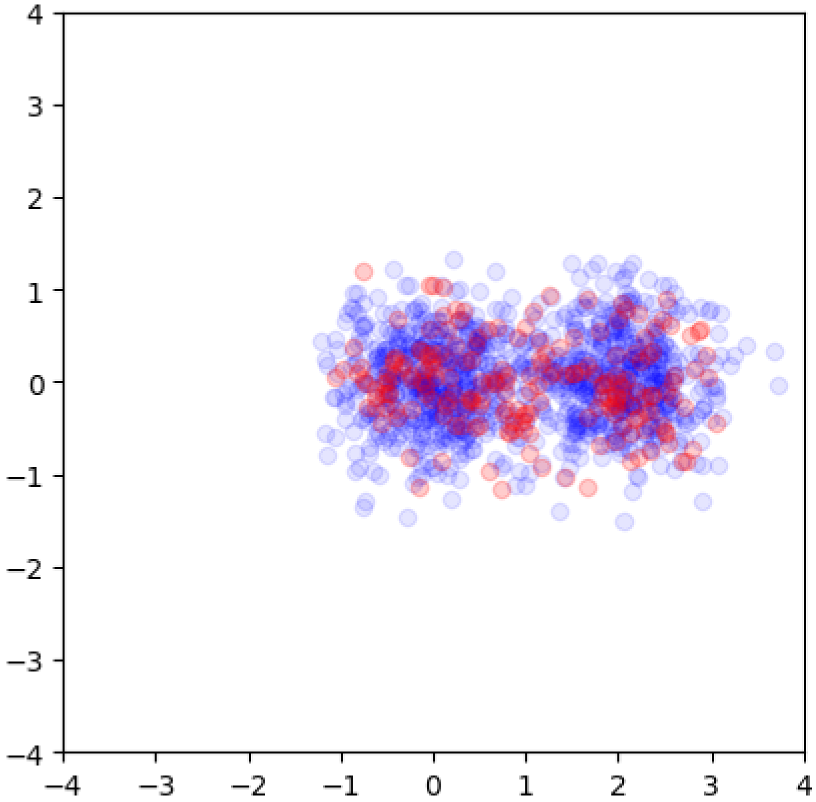
The initial stage of WGAN-SN (spectral normalization) and WGAN-GP are similar, but longer training disable the regularization. WGAN-SN fits the distribution better than weight clipping, weight clipping is unable to generate the full distribution, implying clipping might not the best choice.

- WGAN-SN

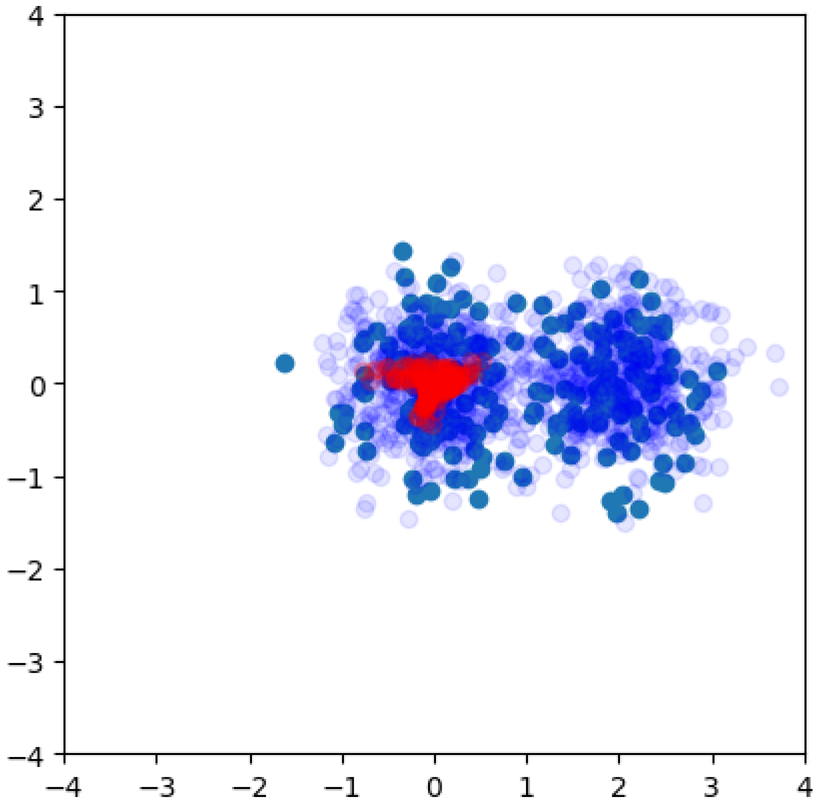


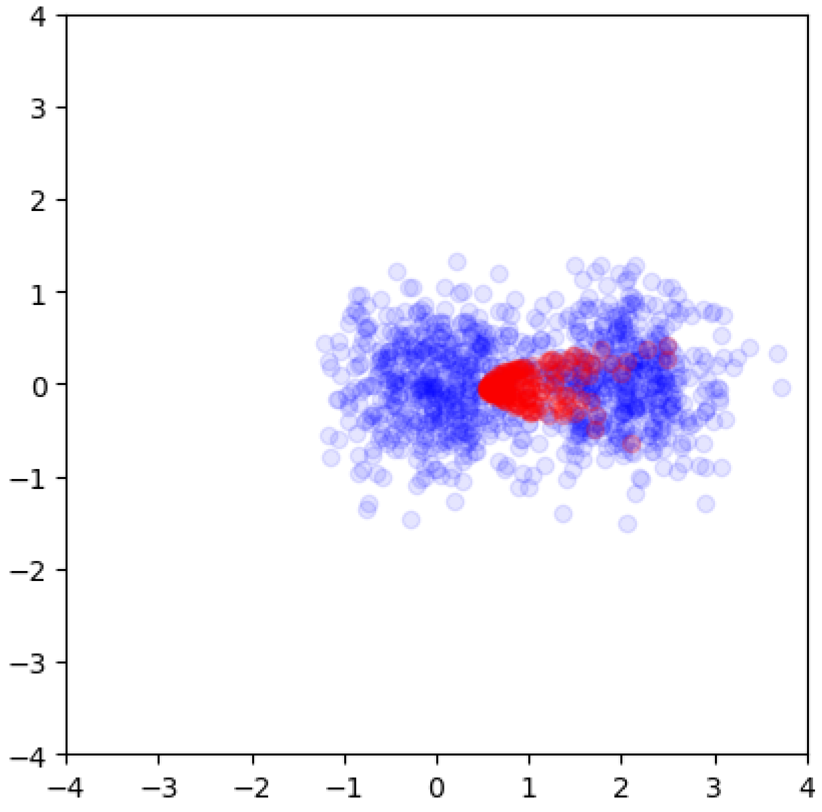


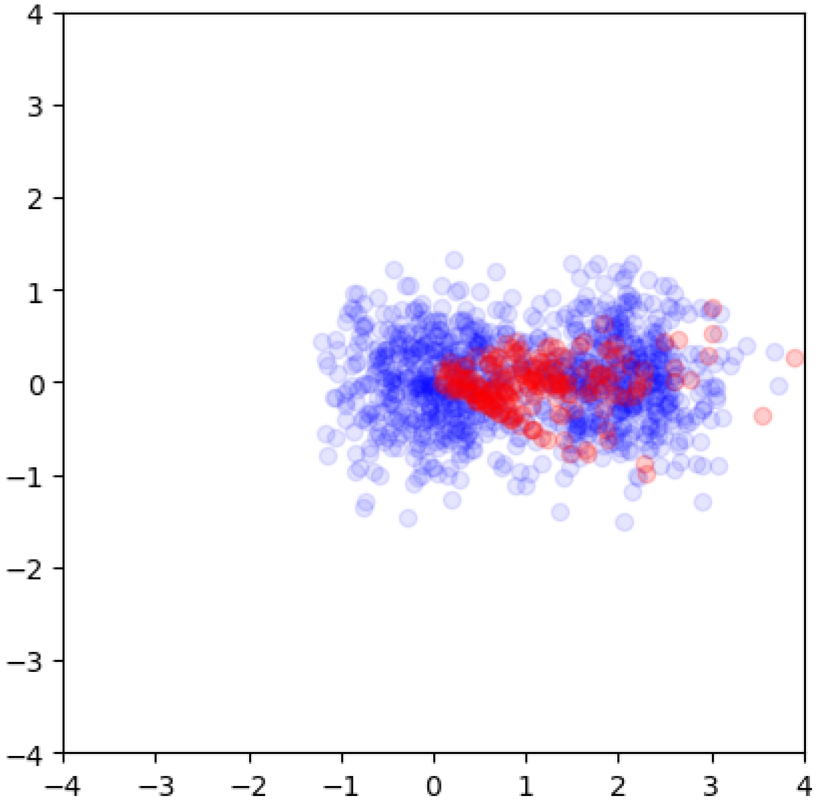

4
3
2
1
0
−1
−2
−3
−4
−4 −3 −2 −1 0 1 2 3 4

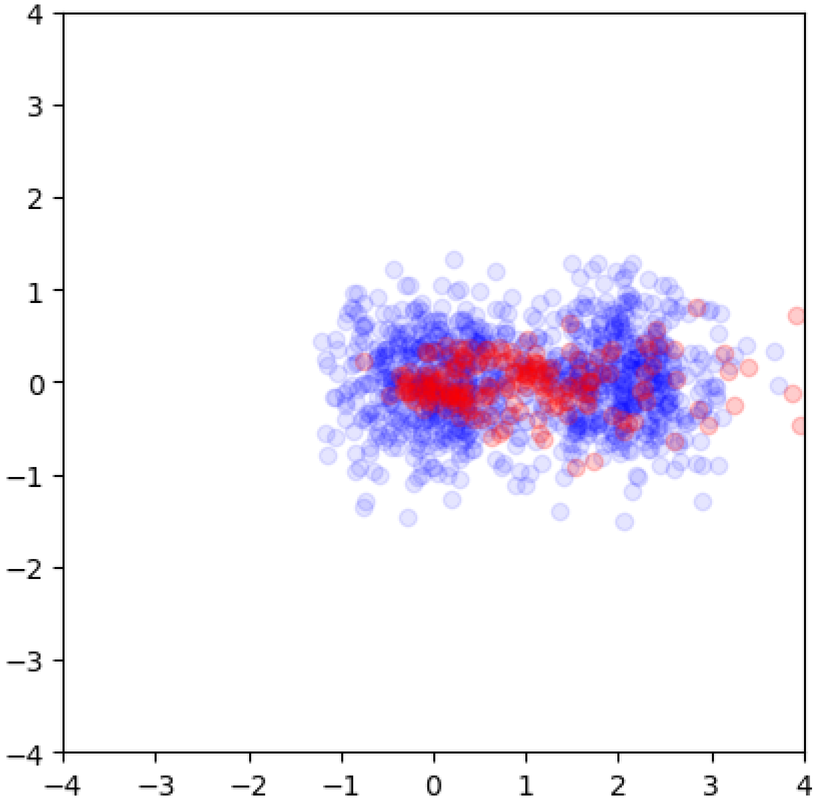

4
3
2
1
0
−1
−2
−3
−4
−4 −3 −2 −1 0 1 2 3 4

- Weight Clipping

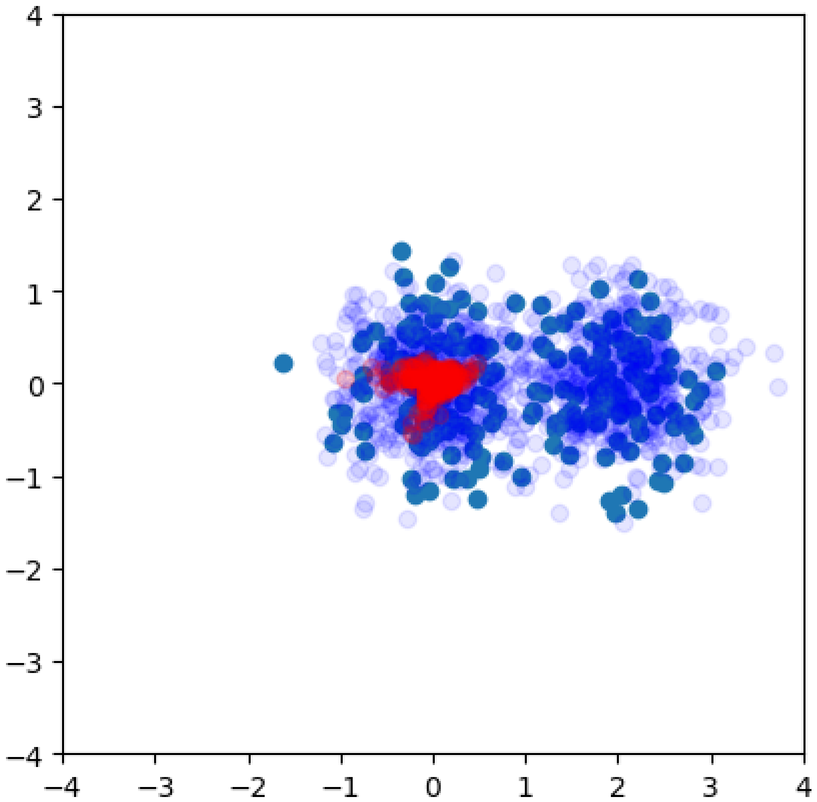


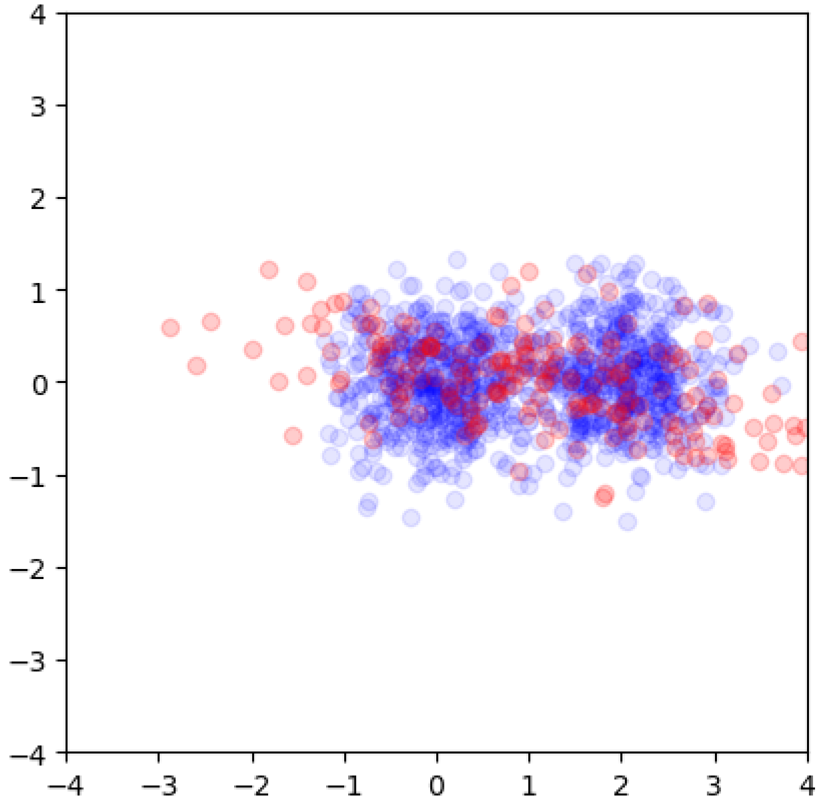


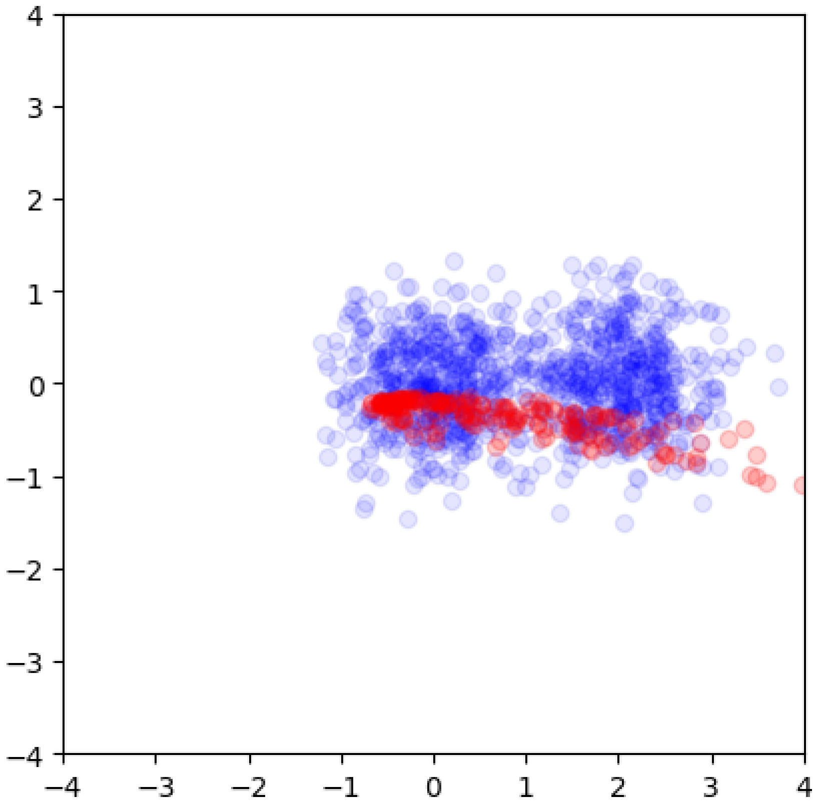

4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

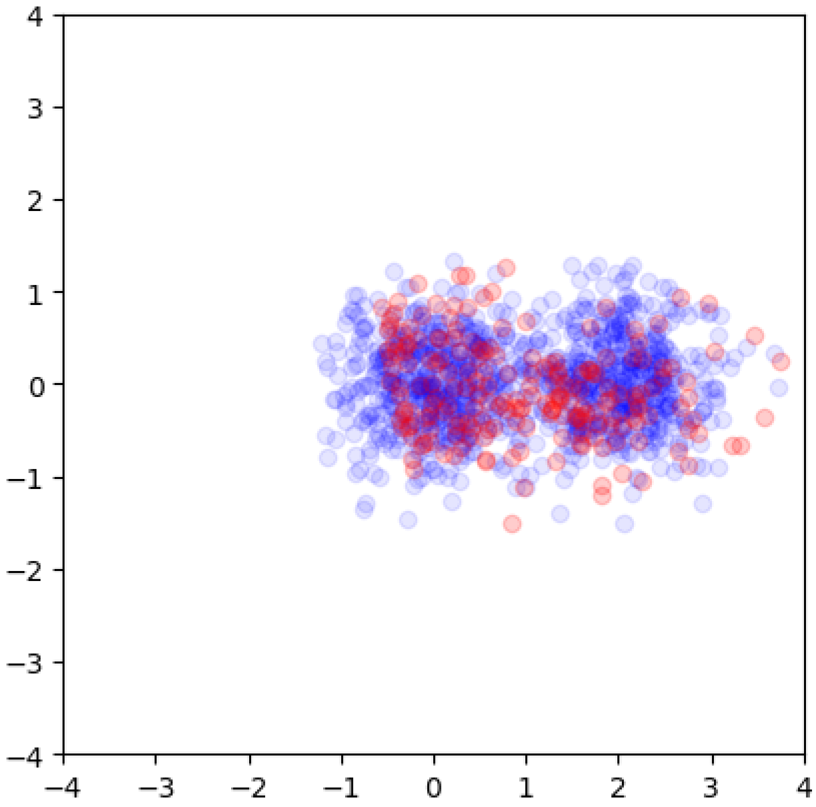

4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

- WGAN-GP









The differences are summarized below:

- **WGAN-GP**: augment the objective function with the regularizer that rewards the function for having local 1-Lipschitz constant $\nabla_{\tilde{x}} D(\tilde{x})$ at discrete sets of points $\tilde{x}$ generated by linearly interpolating a sample from generative distribution $\bar{x}$ and a sample from the data distribution x. Sample data independent regularization.

The penalty can be written as:

$$L = \mathbb{E}_{\bar{x}} D(\bar{x}) - \mathbb{E}_{x} D(x) + \lambda \cdot \mathbb{E}_{\tilde{x}}(\|\nabla_{\tilde{x}} D_{\tilde{x}}(\tilde{x})\|_2 - 1)^2$$

we set $\lambda = 1$. The original paper set $\lambda = 10$ in real world datasets. But for the synthetic datasets, I empirically find $\lambda = 1$ is better.

Pros: does not suffer from effective dimension of the feature space.

Cons: (1) heavily dependent on the support of the current generative distribution, since the generative distribution and its support gradually changes, and this can destabilize the effect of such regularization. (2) High computational cost.

- **WGAN-SN** normalizes the spectral norm of the weight matrix W so that it satisfies the Lipschitz constraint $\sigma(W) = 1$. Augmenting the cost function with a sample data dependent regularization function. **Pros**: (1) regularizes the function the operator space, and the effect of the regularization is more stable with respect to the choice of the batch or aggressive learning rate. (2) unlike the weight normalization and gradient clipping, spectral normalization allows the parameter matrix to use as many features as possible while satisfying local 1-Lipschitz constraint. (3) Low computational cost.
- **Gradient Clipping** directly clip weights into $[-c, c]$, where c is the threshold.

  Pros: easy to implement, low computational budgets. **Cons** (1) leads to optimization difficulties. (2) suffers from effective dimension of the feature space, only models very simple approximations to the optimal functions.

Part 2

The numerical results below are presented below, where column "AVG" presents the mean and std of 3 seed trials for different methods. The accuracy is reported on the last epoch.

LR=10^{-2}	**SEED=0**	**1**	**2**	**AVG**
SGD	0.9383	0.938	0.9363	0.9375 ± 0.0009
MOMENTUMSGD	0.9776	0.9783	0.9786	0.9782 ± 0.0004
RMSPROP	0.965	0.9684	0.9618	0.9651 ± 0.0027
AMSGRAD	0.9796	0.9766	0.9787	0.9783 ± 0.0013

LR=10^{-5}	**SEED=0**	**1**	**2**	**AVG**
SGD	0.0814	0.1042	0.1134	0.0997 ± 0.0135
MOMENTUMSGD	0.2699	0.1318	0.2531	0.2183 ± 0.0615
RMSPROP	0.9166	0.9177	0.9145	0.9163 ± 0.0013

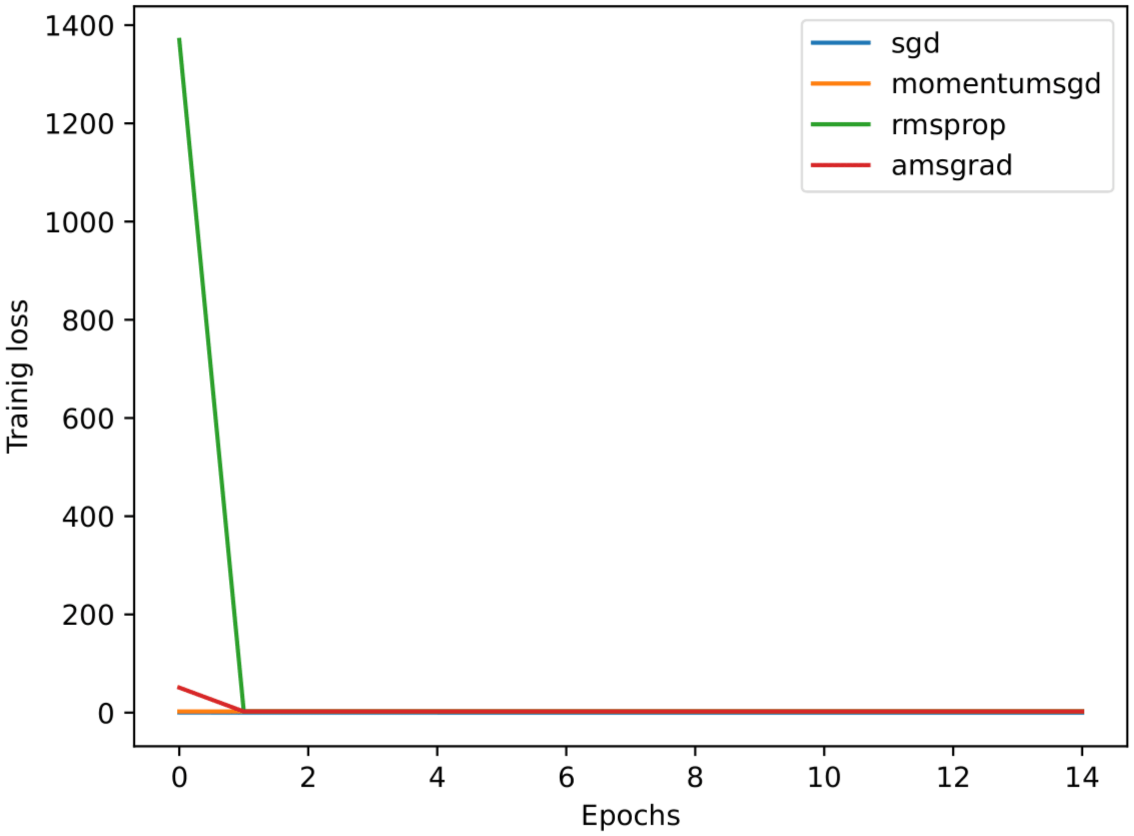
LR=10^{-5}	**SEED=0**	**1**	**2**	**AVG**
AMSGRAD	0.9189	0.9185	0.9174	0.9183 ± 0.0006

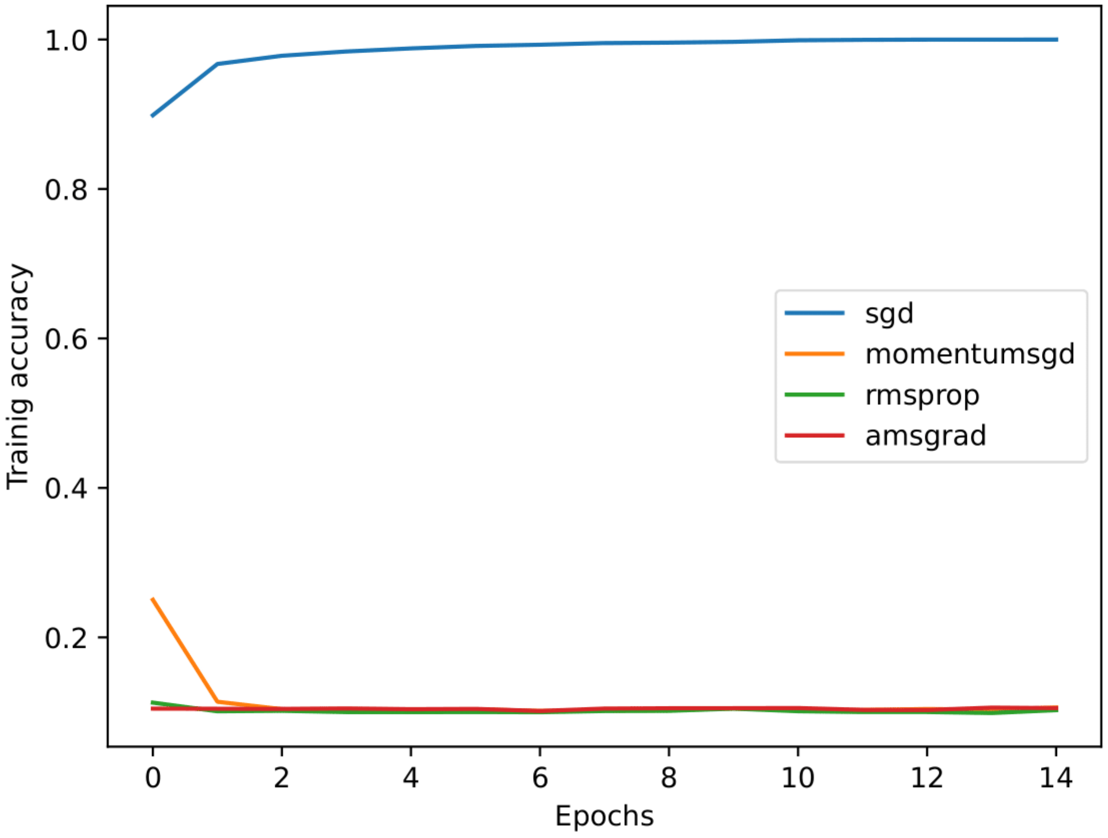
LR=0.5	**SEED=0**	**1**	**2**	**AVG**
SGD	0.9836	0.9852	0.9841	0.9843 ± 0.0007
MOMENTUMSGD	0.1135	0.1028	0.1135	0.1099 ± 0.0050
RMSPROP	0.1028	0.0956	0.0974	0.0986 ± 0.0030
AMSGRAD	0.1028	0.1028	0.1135	0.1064 ± 0.0050

The figures are in the folder code/optimizer/figs/$LR/seed_$SEED/, where LR denotes the learning rate, in $\{0.5, 10^{-2}, 10^{-5}\}$, SEED denotes the random seed, in $\{0, 1, 2\}$.

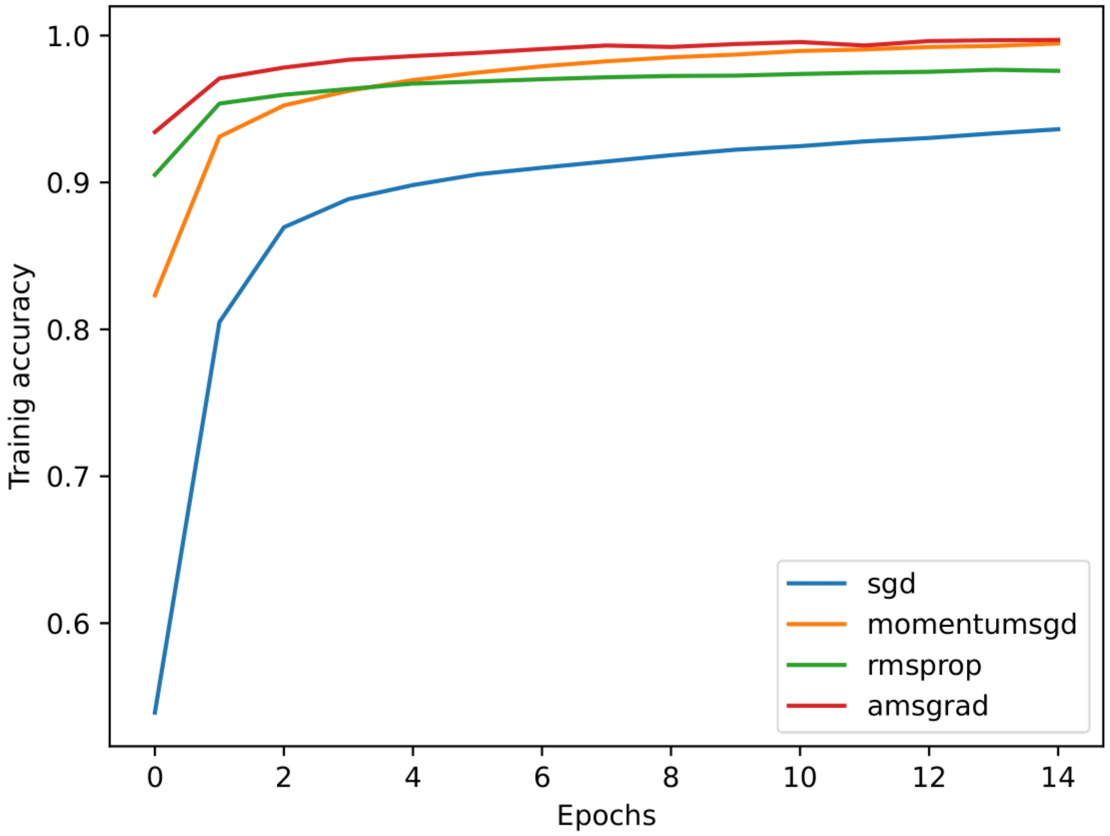
We find that for regular learning rate 10^{-2}, the adaptive learning-rate method have their advantages, for small learning rate 10^{-5}, SGD and SGD with momentum cannot converge under given number of steps, but adaptive lr method will be able to find a good classifer. However, for large stepsize, only the vanilla SGD works, adaptive lr cannot deal with large stepsize, it together with SGD with momentum, unable to converge to a minima with good generalization ability. In all, adaptive learning rate method favors smaller stepsize, while SGD favors larger stepsize.

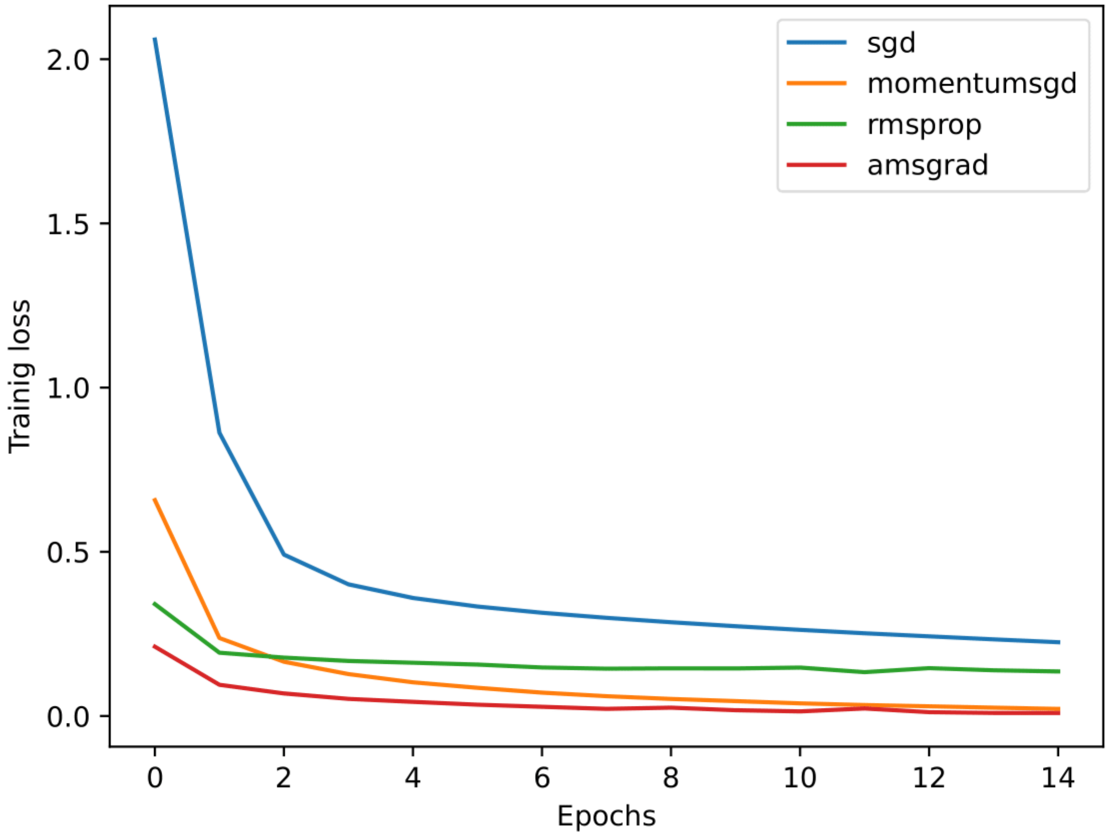
- STEPSIZE 0.5



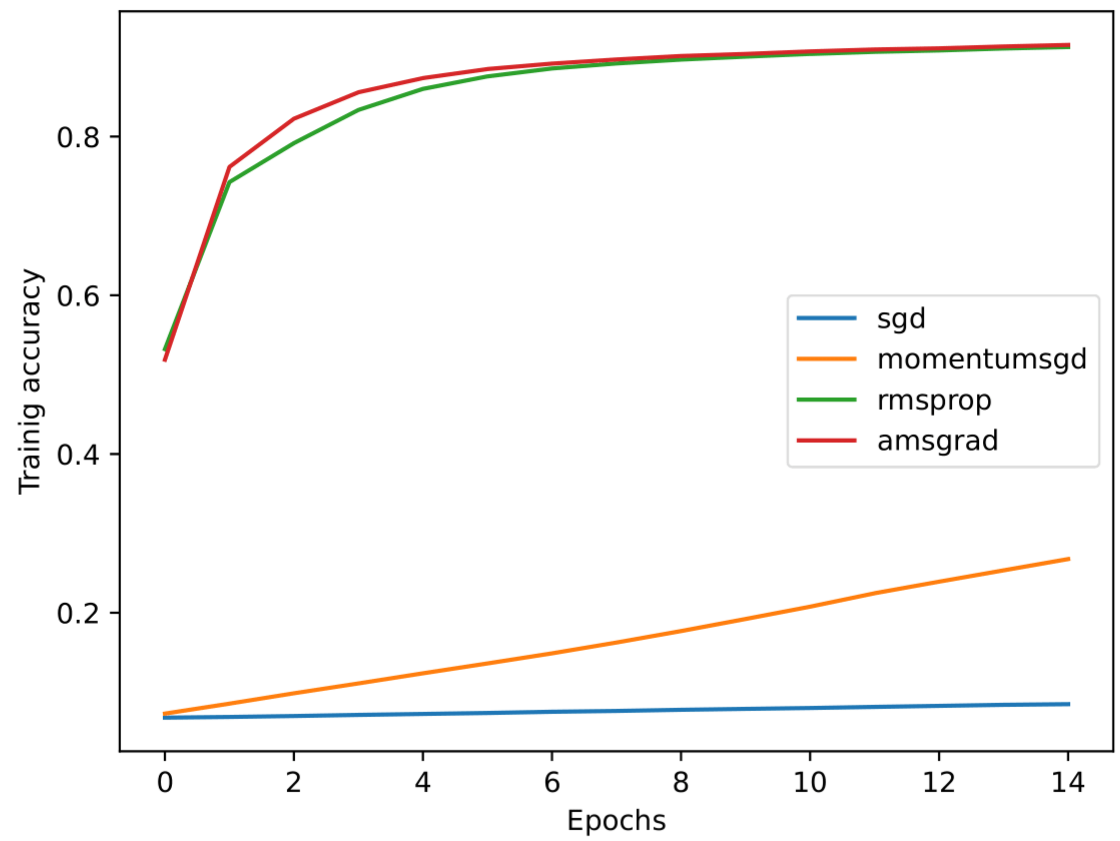


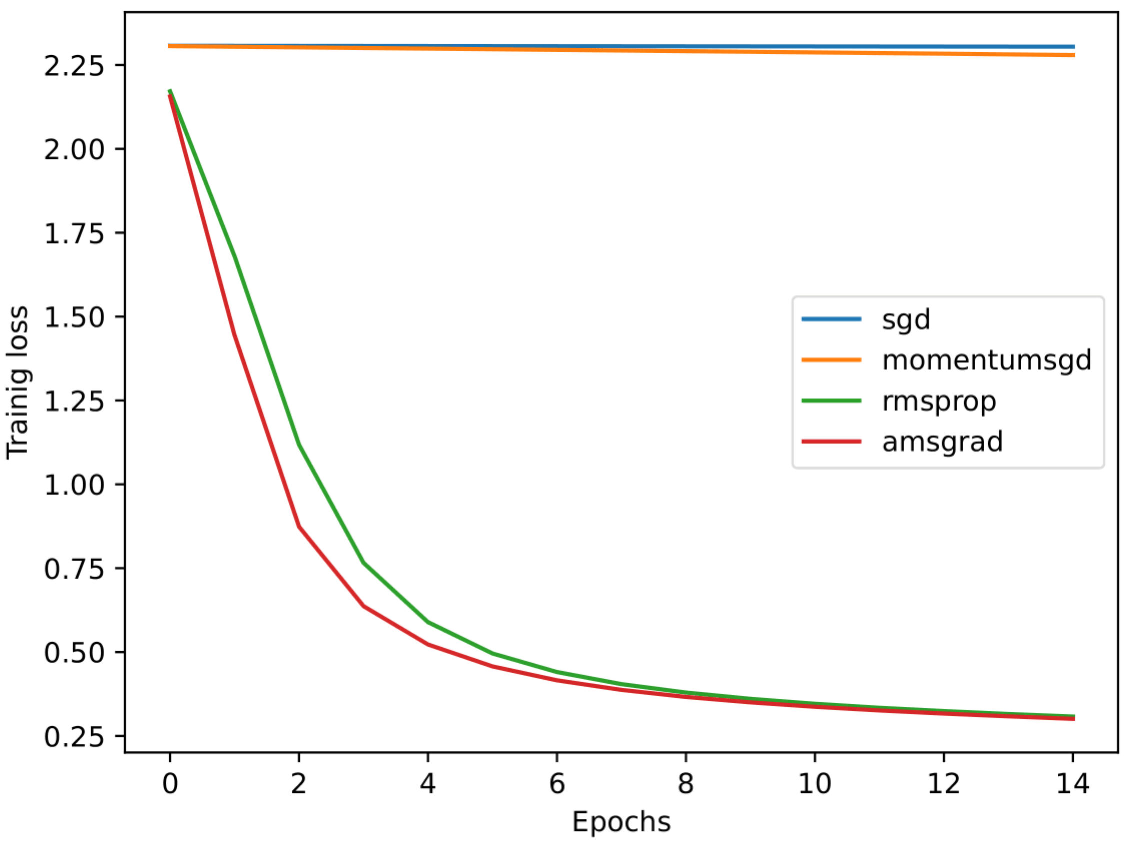
- STEPSIZE 1e-2





- STEPSIZE 1e-5


0.8
0.6
0.4
0.2
Trainig accuracy
0
2
4
6
8
10
12
14
Epochs
sgd
momentumsgd
rmsprop
amsgrad


2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
Trainig loss
0
2
4
6
8
10
12
14
Epochs
sgd
momentumsgd
rmsprop
amsgrad