EE-556: Mathematics of Data **Fall 2019**

Homework #2 - Due date: 15th November 2019

Student: Oriol Barbany Mayor

Problem 1 - Minimax problems and GANs

1.

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix} \tag{1}$$

The first order stationary points are $\{(x,y) : \ \nabla f(x,y) = \mathbf{0}\} = \{(x,y) : x = 0, y = 0)\}$, so there's actually only one of such.

$$\nabla^2 f(x,y) = \begin{bmatrix} \frac{\partial^2}{\partial x^2} f(x,y) & \frac{\partial^2}{\partial x \partial y} f(x,y) \\ \frac{\partial^2}{\partial y \partial x} f(x,y) & \frac{\partial^2}{\partial y^2} f(x,y) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2}$$

The Hessian gives information about the type of first order stationary points. In this case, it has eigenvalues $\lambda = [0, 1]$ in all domain, so in particular $(0, 0)$ is a saddle point since the Hessian is not positive definite nor negative definite.

2. For $(x^*, y^*) = (0, 0)$, it's trivial to see that $0 = f(x^*, y^*) \geq f(x^*, y) = 0$ and $0 = f(x^*, y^*) \leq f(x, y^*) = 0$, so indeed (x^*, y^*) is a solution to the problem $\min_x \max_y f(x, y)$.

3. The distance in squared L2-norm to the optimal solution in iteration $k + 1$ is

$$\begin{aligned} \|(x_{k+1}, y_{k+1}) - (x^*, y^*)\|^2 &:= \|(x_k - \gamma y_k, y_k - \gamma x_k)\|^2 && (3)\\ &:= x_k^2 - 2\gamma x_k y_k + \gamma^2 y_k + y_k^2 + 2\gamma y_k x_k + \gamma^2 x_k^2 && (4)\\ &= (1+\gamma^2)(x_k^2 + y_k^2) := (1+\gamma^2) \|(x_k, y_k)\|^2 = (1+\gamma^2)^{k+1} \|(x_0, y_0)\|^2 && (5)\\ &= (1+\gamma^2)^{k+1} \|(x_0, y_0) - (x^*, y^*)\|^2 && (6) \end{aligned}$$

so if $(x_0, y_0) \neq \mathbf{0}$, the iterates diverge as $\mathcal{O}((1+\gamma^2)^k)$ in terms of the distance (in termss of the squared L2-norm) to the optimum in iterate k from the initial distance to the optimum.

4.

$$|f(\mathbf{x}) - f(\mathbf{y})| = |\mathbf{v}^T(\mathbf{x} - \mathbf{y})| := |\langle \mathbf{x} - \mathbf{y}, \mathbf{v}^* \rangle| \leq \|\mathbf{v}\| \|\mathbf{x} - \mathbf{y}\| := B \|\mathbf{x} - \mathbf{y}\| \tag{7}$$

where the inequality follows from Cauchy-Schwarz. So the set of functions that are $B-$Lipschitz with $B \leq 1$ is $\{f \in \mathcal{F} : \ \sqrt{v_1^2 + v_2^2} \leq 1\}$.

5. Note that in order to enforce that a function $f \in \mathcal{F}$ is $1-$Lipschitz, it's enough to normalize its vector $\mathbf{v}$ so as it has unit norm, i.e. $\mathbf{v} \leftarrow \frac{\mathbf{v}}{\|\mathbf{v}\|}$.

6. The performance with alternating updates is better than that with simultaneous updates. The estimates at some iterations for this latter approach can be seen in Figure 1. Both show an oscillatory behavior that is discussed in the last section of this exercise, but in the case of the alternating updates, the mean of the distribution estimated at each iterate is closer than the true mean. We could say that the alternating updates allow to more rapidly adapt to the first moment of the true distribution. For further comparison, see the `gif` files included in the deliverable.

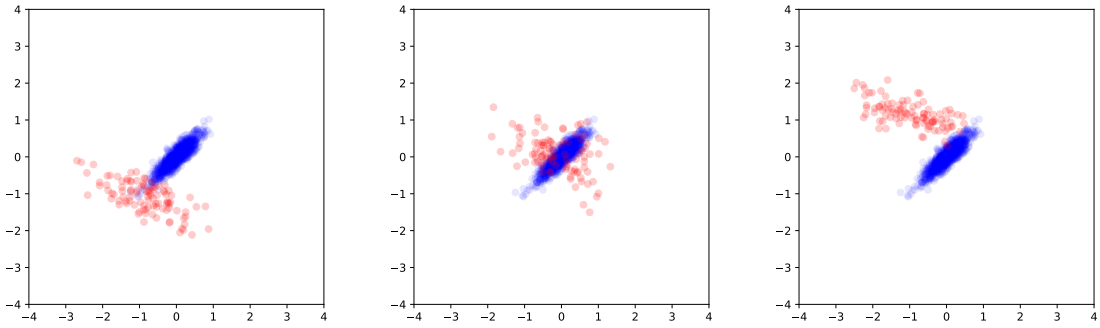

Figure 1: Snapshots of the evolution with simultaneous updates

7. By definition of f, linearity of the expectation and given that $\mathbf{v}$ is deterministic,

$$\max_{f \in \mathcal{F}} \mathbb{E}_{\mathbf{X} \sim \mu}[f(\mathbf{X})] - \mathbb{E}_{\mathbf{Y} \sim \nu}[f(\mathbf{Y})] = \max_{\mathbf{v} \in \mathbb{R}^2} \mathbf{v}^T (\mathbb{E}_{\mathbf{X} \sim \mu}[\mathbf{X}] - \mathbb{E}_{\mathbf{Y} \sim \nu}[\mathbf{Y}]) \geq 0 \tag{8}$$

   where the inequality follows by setting $\mathbf{v} = \mathbf{0}$, which is not necessarily the maximum.

8. In the case that the first moment coincide for both distributions μ and ν, we have that $\mathbb{E}_{\mathbf{X} \sim \mu}[\mathbf{X}] = \mathbb{E}_{\mathbf{Y} \sim \nu}[\mathbf{Y}]$, so (8) is satisfied with equality in this case.

   This explains the observed behavior in the GAN experiment, since it can estimate the first moment but not higher order moments such as the covariance defining the shape of the true distribution. Hence, the obtained distribution keeps oscillating around a zero mean distribution but doesn't converge to the true one.

Problem 2 - Optimizers of Neural Network

The training accuracy and cross-entropy loss for several optimizers and learning rates can be seen in Figure 2, where the mean and variance over three runs is depicted.

- `sgd:` We can see that SGD is the best optimizer for the learning rate of 0.5, but its performance decreases for the lower learning rate of 0.01 and it barely improves with the lowest step size, where we can see that its loss and accuracy remains flat over all the epochs.

- `momentumsgd:` When we add momentum to SGD, we overshoot for large step sizes but get better performance than the naive SGD when the learning rate decreases. This is because the momentum effectively increases the learning rate when the descent direction remains stable in consecutive iterations and thus improves with small learning rates but decreases the performance for large learning rates.

- `adagrad:` This optimizer adapts its learning rates for every direction but yet we can observe the influence of the initial step size, specially for the value of 10^{-5}. It is the best performing optimizer for the step size of 0.01 and its worst performance is attained at the lowest of the tested learning rates.

- `rmsprop:` Similarly to AdaGrad, we have adapting learning rate, but in this case, the updates of the effective step size from one iteration to the following one, are in a moving average fashion instead of a simple hard assignment. This is specially beneficial for the learning rate of 10^{-5}, where we have a rather flat region where AdaGrad slowly converges. Nevertheless, we can see that for a large initial step size, the moving average updates don't allow to rapidly change the effective step size as it happens with AdaGrad.

- `adam:` In addition to RMSProp, ADAM uses second order moment estimation. Hence, we observe the same flat behavior for the large step size as RMSProp and as discussed earlier, the momentum doesn't help in this case. With lower learning rates, Adam achieves the best performance since it can adapt

its learning rate and further accelerate if the descent direction remains the same from one iteration to another with the momentum term.

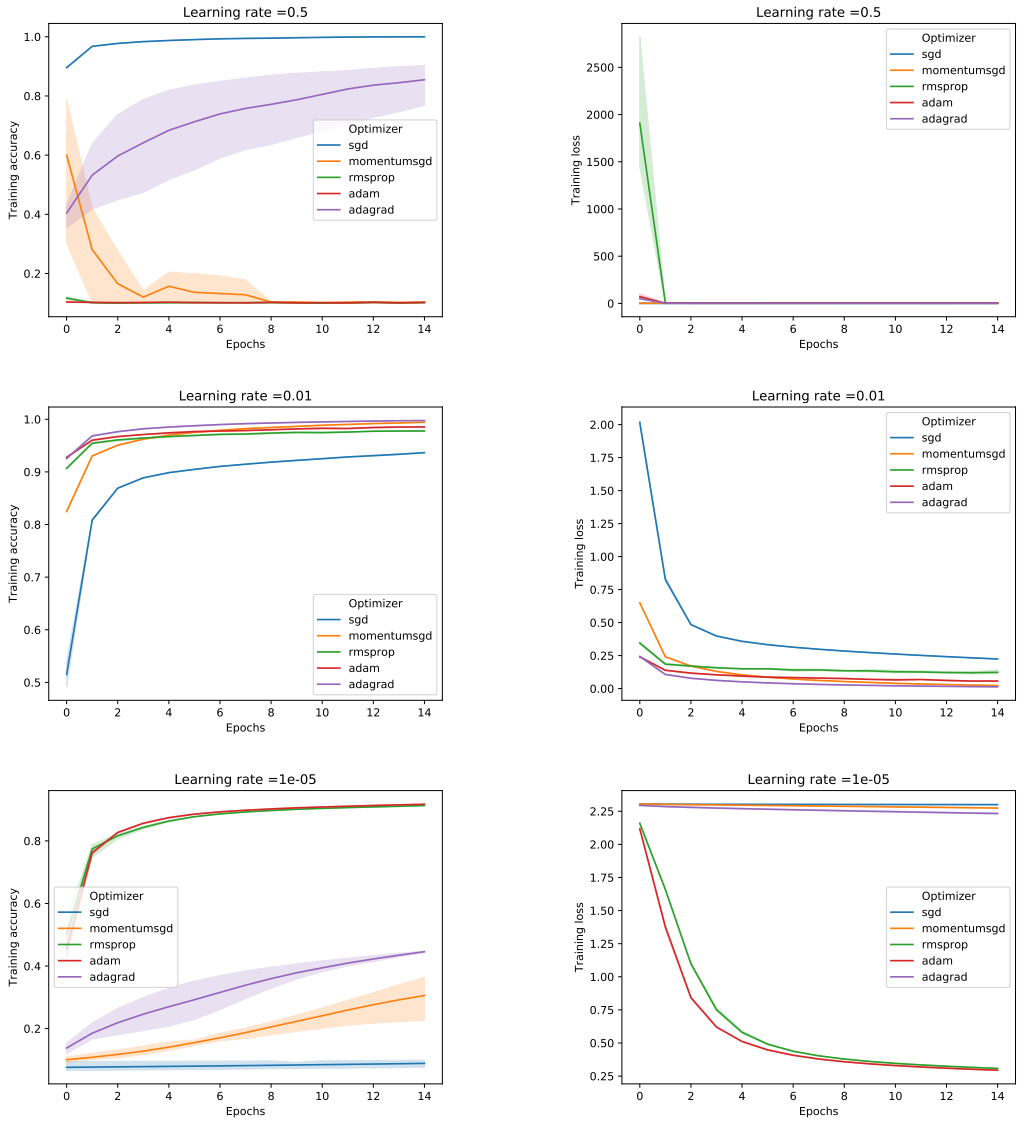

Figure 2: Comparison of the implemented optimizers for various learning rates over 3 runs