ECE-GY 9123 / Spring 2021

Homework 6

1. **(4 points)** In this problem we will see how minimax optimization (such as the one used for training GANs) is somewhat different from. For illustration, consider a simple problem, consider a simple function of 2 scalars:

$$\min_x \max_y f(x, y) = 4x^2 - 4y^2.$$

You can try graphing this function in Python to visualize this (there is no need to include the graph in your answer).

   a. Determine the saddle point of this objective function. A saddle point is a point (x, y) for which f attains a local minimum along one direction and a local maximum along an orthogonal direction.

   b. Write down the gradient descent/ascent equations for this objective function starting from any arbitrary initial point.

   c. Determine the range of allowable step sizes for the ascent and descent required to ensure that the gradient descent/ascent converges to the saddle point.

   d. What if, instead of solving a minimax problem, you just did regular gradient *descent* for both x and y? Comment on the dynamics of the updates, and whether or not one would converge to the saddle point.

Solutions

   a. Actually this particular problem is easy (finding saddle points for general functions is not trivial). If we graph the function, we can see that this literally looks like a saddle:

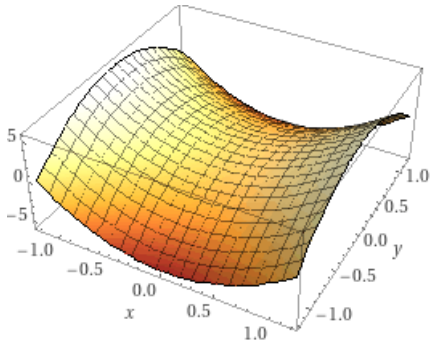


Figure 1: Saddle points

So we can guess that there is a saddle point at (0,0). To check this, we compute the partial derivatives:

$$\frac{\partial f}{\partial x} = 8x, \qquad \frac{\partial f}{\partial y} = -8y,$$

and verify that both derivatives go to zero at $x = 0, y = 0$. Finally, we can check (either graphically or by computing the second derivative) that (0,0) is a local minimum in the x-direction and a local maximum in the y-direction.

b. We can write gradient descent/ascent as follows. Starting at an arbitrary (x_0, y_0), we get the following pair of recurrences:

$$x_{t+1} = x_t - \eta_x \left.\frac{\partial f}{\partial x}\right|_{x_t},$$
$$y_{t+1} = y_t + \eta_y \left.\frac{\partial f}{\partial y}\right|_{y_t},$$

for step sizes η_x, η_y. It is standard to assume positive step sizes. Plugging in the partial derivatives, we get:

$$x_{t+1} = (1 - 8\eta_x)x_t,$$
$$y_{t+1} = (1 - 8\eta_y)y_t.$$

c. Staring at the above equations a bit; these are both geometric progressions. These sequences will converge if the contraction ratio is less than 1 in magnitude. Therefore, for η_x (and identically for η_y), we get:

$$|1 - 8\eta_x| < 1,$$

which gives the range of allowable step sizes:

$$0 < \eta_x, \eta_y < \frac{1}{4}.$$

Observe that under the above choice of step size, this will indeed converge to (0,0).

d. This requires some care to answer correctly. Regular gradient descent looks very similar to the equations written above, but with a $-$ instead of a $+$ for the y gradient-update. (Basically we are descending along the negative gradient for both x and y). So we get:

$$x_{t+1} = (1 - 8\eta_x)x_t,$$
$$y_{t+1} = (1 + 8\eta_y)y_t.$$

So with positive step sizes, the y updates will *generally diverge*. This makes sense; if we imagine rolling a ball down the saddle, it will descend down to $-\infty$ in *most cases*.

Here, note the emphasis on the word *generally* and in *most cases*. This sequence does converge in the very special case where $y_0 = 0$, and we choose $0 < \eta_x < 1/4$. (It does not matter what η_y is.) This corresponds to the case where the ball is exactly placed in the $y = 0$ axis, so the only effective updates are along x.

2. **(6 points)** In this problem, the goal is to train and visualize the outputs of a simple Deep Convolutional GAN (DCGAN) to generate realistic-looking (but fake) images of clothing.

a. Use the FashionMNIST training dataset to train the DCGAN. APIs for downloading it are available in both PyTorch and TensorFlow. Images are grayscale and size 28×28.

b. Use the following discriminator architecture (kernel size = 5×5 with stride = 2 in both directions):

- 2D convolutions ($1 \times 28 \times 28 \rightarrow 64 \times 14 \times 14 \rightarrow 128 \times 7 \times 7$)
- each convolutional layer is equipped with a Leaky ReLU with slope 0.3, followed by Dropout with parameter 0.3.
- a dense layer that takes the flattened output of the last convolution and maps it to a scalar.

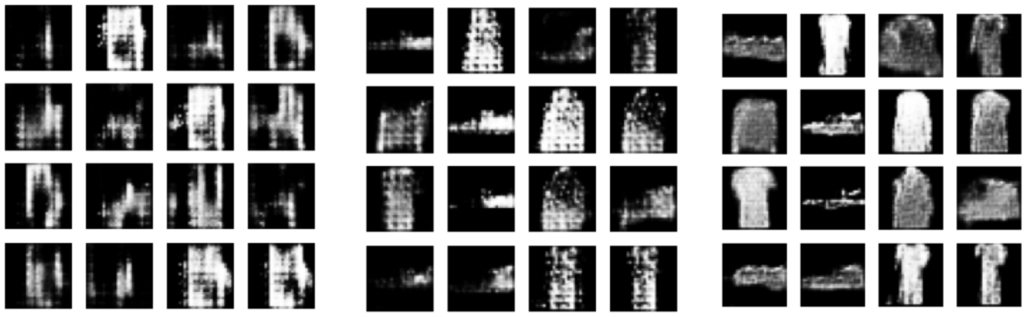
Here is a link that discusses how to appropriately choose padding and stride values in order to desired sizes.

c. Use the following generator architecture (which is essentially the reverse of a standard discriminative architecture). You can use the same kernel size. Construct:

- a dense layer that takes a unit Gaussian noise vector of length 100 and maps it to a vector of size $7 * 7 * 256$. No bias terms.
- several transpose 2D convolutions ($256 \times 7 \times 7 \rightarrow 128 \times 7 \times 7 \rightarrow 64 \times 14 \times 14 \rightarrow 1 \times 28 \times 28$). No bias terms.
- each convolutional layer (except the last one) is equipped with Batch Normalization (batch norm), followed by Leaky ReLU with slope 0.3. The last (output) layer is equipped with tanh activation (no batch norm).

d. Use the cross-entropy loss for training both the generator and the discriminator. Use the Adam optimizer with learning rate 10^{-4}.

e. Train it for 50 epochs. You can use minibatch sizes of 16, 32, or 64. Training may take several minutes (or even up to an hour), so be patient! Display intermediate images generated after $T = 10$, $T = 30$, and $T = 50$ epochs. If the random seeds are fixed throughout then you should get results of the following quality:



f. Report loss curves for both the discriminator and the generator loss over all epochs, and qualitatively comment on their behavior.

Solutions

A notebook has been posted on NYUClasses.