HW 6

Hunter Busick

11/19/2024

What is the difference between gradient descent and *stochastic* gradient descent as discussed in class? (*You need not give full details of each algorithm. Instead you can describe what each does and provide the update step for each. Make sure that in providing the update step for each algorithm you emphasize what is different and why.*)

Both gradient descent and stochastic gradient descent seek to minimize an error function by iterating through different combinations of parameters and updating said parameters with each iteration. Where the two differ is the amount of data that is used in this process. In gradient descent this process is performed on the entire dataset in which every observation is considered before a parameter is updated. This update step looks like

$$\theta_{i+1} = \theta_i - \alpha\Lambda f(\theta_i, x, y)$$

where x and y are not a subset. Contrarily, stochastic gradient descent use a random subset of the data in updating the parameters, rather than using the entire set. This can be helpful when working with very large datasets in which it would be difficult to iterate over the entire set for each parameter update. This will increase the variability in gradients, but will help us prevent getting stuck in local extremes presented in the entire set. Our stochastic update step is

$$\theta_{i+1} = \theta_i - \alpha\Lambda f(\theta_i, x_i, y_i)$$

where x_i and y_i are *subsets*.

Consider the `FedAve` algorithm. In its most compact form we said the update step is $\omega_{t+1} = \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$. However, we also emphasized a more intuitive, yet equivalent, formulation given by $\omega_{t+1}^k = \omega_t - \eta \nabla F_k(\omega_t); w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$.

Prove that these two formulations are equivalent.
(*Hint: show that if you place ω_{t+1}^k from the first equation (of the second formulation) into the second equation (of the second formulation), this second formulation will reduce to exactly the first formulation.*)

From the second equation of the second formulation we are given

$$\omega_{t+1}^k = \omega_t - \eta \nabla F_k(\omega_t)$$

plugging ω_{t+1}^k into the second formulation we then have

$$\omega_{t+1} = \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} (\omega_t - \eta \nabla F_k(\omega_t))$$

we can then distribute $\frac{n_k}{n}$

$$\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$$

We can then simplify each summation. The first one:

$$\sum_{k=1}^{K} \frac{n_k}{n} \omega_t = \omega_t \sum_{k=1}^{K} \frac{n_k}{n}$$

We can further simplify as we know our weights sum to 1. Our final equation for the first summation is

$$\sum_{k=1}^{K} \frac{n_k}{n} \omega_t = \omega_t$$

Our second summation is then

$$-\eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$$

Thus we can substitute back such that

$$\omega_{t+1} = \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$$

where ω_{t+1} is equal to the simplified versions of separating out the earlier summations.

Now give a brief explanation as to why the second formulation is more intuitive. That is, you should be able to explain broadly what this update is doing.

The second formulation is more explainable than the first. Though we have two separate parts they are easily distinguishable. The first represents the local update for each client k. The second formulation then takes the weighted average of each local update and applies it to the global model. In this way we are able to continually update the global model without ever exchanging the raw data across models.

Prove that randomized-response differential privacy is ϵ-differentially private.

If a subset of responses consists of

$$truth = (TruthHH, TruthHT, YesTH, NoTT)$$

Where our output space is

$$Output = (Yes, No)$$

We can then calculate the probability of having output "Yes" when our input is "Yes" divided by the probability our output is "Yes" when our input is no. In our example this value will be $(3/4)/(1/4) = 3 = e^{ln(3)}$. Thus $\epsilon = 3$ We can then divide the probability that the truth is "No" and they answer "Yes" divided by the truth being "Yes" and they answer "Yes" as 1/3. probability of dividing the probability of random guess "No" = "Yes" divided by "Yes" = "Yes" is $e^{ln(3)}$. Thus we have shown that randomized response differential privacy is ϵ - differentially private.

Define the harm principle. Then, discuss whether the harm principle is *currently* applicable to machine learning models. (*Hint: recall our discussions in the moral philosophy primer as to what grounds agency. You should in effect be arguing whether ML models have achieved agency enough to limit the autonomy of the users of said algorithms.*)

J.S. Mill's harm principle states that agents should have the ability to act as they choose, as long as their actions do not impose harm among another agent. In essence it argues that the right to swing your fist ends at someone else's nose, and that state interference should only occur when agents are likely to harm others. I believe that machine learning models do not have agency enough to be held accountable to the harm principle. The blame should fall on the one utilizing/ creating the model. The birth of a machine learning model is the data it is trained on, and a newborn model will not be able to tell whether the data it is fed is biased or inaccurate. In this sense current machine learning models have next to no agency as they only work off of what they are initially fed and make their next movements off of that. For instance, it is more common to hold the team at NorthPointe accountable for the COMPAS algorithm outcomes than holding the algorithm itself accountable, because it is understood that the model was built on or fed information that prompted biased outcomes. Developers and those who irresponsibly use machine learning models are not exempt from the harm principle, however holding models themselves to the same standard is not a good use of our time.