HW 6

Mackie Jackson

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.*)

Gradient descent minimizes a federated learning loss function by updating the basic gradient rule to follow the direction opposite of the gradient. It utilizes all data which often causes it to get stuck in local rather than global minima, which is why stochastic gradient descent is often utilized instead. Stochastic gradient descent uses a subset of data for each step, which allows it to calculate a global minimum more quickly.

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.*)

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

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

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

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 intuitive because it shows the updated step of each K client's step being averaged to attain a global update.

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

As discussed in class, randomized response differential privacy is ϵ - differentially private where $\epsilon = ln(3)$.

If D and $S \in Yes, No$, consider the case $S = Yes$:

$\frac{(P[A(Yes)=Yes]}{P[A(No)=Yes]} = \frac{3/4}{1/4} = 3 = e^{ln(3)}$

and

$\frac{(P[A(No)=Yes]}{P[A(Yes)=Yes]} = \frac{1}{3}$

Thus,

$\frac{(P[A(D_1)\in S]}{P[A(D_2)\in S]} \leq 3 = e^{ln(3)}$

We can conclude that the randomized response is $ln(3)$ - 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.*)

The harm principle holds that personal autonomy ends at the point where exercising said autonomy causes objective harm to another moral agent. I would argue that it is not machine learning models that this principle is currently applicable to, but the developers of these models. Ml models do not currently meet the requirements of moral autonomy (to give oneself moral laws) or personal autonomy (to make decisions/take actions for oneself). They do not possess the capability to create their own moral code or take independent action/thought because they make "decisions" by classifying data as directed by a developer. Suggesting that ML models are in any way separated from their developers' direction removes moral responsibility from developers for harm caused by implementation of their models.