HW 6

Student Name

1/21/2024

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.*)

SGD uses a mini-batch or a single data point whereas gradient descent uses the entire dataset. This allows SGD to escape local minima and find the optimal solution much faster than gradient descent.

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.*)

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

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

$w_{t+1} = \omega_t - \sum_{k=1}^{K} \frac{n_k}{n} \eta \nabla F_k(\omega_t)$ Separate sums, factor out ω_t and sum $\sum_{k=1}^{K} \frac{n_k}{n} = 1$

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

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 first equation shows that each client K performs its own update and the second equation shows the global update which averages all K updates based on the proportion of the data $\frac{n_k}{n}$

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

Since D and S are simply elements of {Yes, No}, let's first consider the case where S is Yes.

P(Output = Yes | Input = Yes) = 3/4 and P(Output = Yes | Input = No) = 1/4

This leads to

$\frac{P[A(Yes)=Yes]}{P[A(No)=Yes]} = \frac{P(Output=Yes|Input=Yes)}{P(Output=Yes|Input=No)} = \frac{3/4}{1/4} = 3 \leq e^{\epsilon}$

Now let's consider the case where S is No

P(Output = No | Input = Yes) = 1/4 and P(Output = No | Input = No) = 3/4

This leads to

$\frac{P[A(Yes)=No]}{P[A(No)=No]} = \frac{P(Output=No|Input=Yes)}{P(Output=No|Input=No)} = \frac{1/4}{3/4} = 1/3 \leq e^{\epsilon}$

As a result, $\epsilon \leq ln(3)$ and satisfies ϵ-differential privacy

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 states that the only reason to limit one's autonomy is to prevent harm to others. The harm principle is not currently applicable to machine learning models because the models do not have autonomy or moral responsibility. There is no intention in its actions; it simply processes data and produces an output based an a preprogrammed algorithm. Simply put, a model does what its told. Therefore, the models do not have agency and therefore the harm principle is not applicable.