HW 6

Holden Wright

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.*)

Both algorithms are used to minimize a loss function. In each step we take derivatives and for a function f() we are considering the gradient: $\nabla f = (\partial f/\partial x1, ..., \partial f/\partial xd)$. *For gradient descent the update step is:* $\theta i{+}1 = \theta i - \alpha \nabla f(\theta i, X, Y)$ *and for stochastic gradient descent the update step is:* $\theta i + 1 = \theta i - \alpha \nabla f(\theta i, Xi', Yi')$ *The difference in these algorithms is that gradient descent all data is used in computing the gradient and for stochastic gradient descent a random subset of the data is used in every step.*

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 first step of the intuitive formulation: $\omega_{t+1}^k = \omega_t - \eta \nabla F_k(\omega_t)$

Substitute into second step: $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$ $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} (\omega_t - \eta \nabla F_k(\omega_t))$

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

Simplify first term: Since ω_t is independent of k, it can be factored out of the first summation: $\omega_t \sum_{k=1}^{K} \frac{n_k}{n}$

The weights $\frac{n_k}{n}$ sum to 1

Thus: $\omega_t \sum_{k=1}^{K} \frac{n_k}{n}$ is equal to ω_t

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

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 of the FedAve algorithm is more intuitive because it breaks the update into two separate steps that show the local and global processes in federated learning. In the first step each client k computes its own update to the model based on its local data. This is the standard gradient descent update, where each client adjusts its local model $\omega_{t+1}^k = \omega_t - \eta \nabla F_k(\omega_t)$. The update here is local to the client, meaning each client performs its own gradient descent update without any communication with others. After each client performs its local update, the global model w_{t+1} is obtained by aggregating the individual updates from all the clients. This aggregation is weighted by the size of the clients' data sets, where the term $\frac{n_k}{n}$ represents the proportion of data each client holds relative to the total data across all clients.

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

In the randomized-response mechanism for a binary variable x $\in$ {0,1}:

With probability p, the true value x is reported. With probability 1-p, the opposite value 1-x is reported.

The probability of reporting y $\in$ {0,1} given the true value x is: P(y=x) = p, P(y$\neq$x) = 1-p

Let D and D' be two neighboring data sets, differing only in the response of a single individual. Assume the differing individual has true value x in D and x' in D', with x$\neq$x'.

For x=0 and output y=1: P(M(D)=y) = P(y=1 | x=0)= 1-p P(M(D')=y) = P(y=1 | x'=1)=p

For the case where x=0, x'=1, and y=1, we have: (P(M(D)=y) / P(M(D')=y)) = ((1-p) / p)

To satisfy differential privacy, this ratio must be bounded by e^ϵ: (1-p/p) $\leq$ e^ ϵ

p $\geq$ (1 / (e^ ϵ + 1))

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 is defined as: personal autonomy is checked at the point where exercising personal autonomy causes objective harm to another moral agent. ML models lack agency because they operate off of programmed instructions and learned patterns, without intentionality or moral understanding. They do not possess autonomy or self-awareness, both of which are key components of agency. ML models themselves cannot restrict autonomy but can enable restrictions when misused i.e. biased algorithms in approving loans Therefore, the harm principle applies indirectly, focusing on human developers and users rather than the models themselves.