HW 6

Anna Fetter

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.*)

Student Input

Both gradient descent and stochastic gradient descents are optimization algorithms. A gradient is the vector direction of the steepest ascent or descent; it's a multivariable calculus concept.

Gradient descent calculates the average gradient descent for the entire dataset using the loss function. In contrast, stochastic gradient descent calculates the gradient using the derivative based on a single point or batch. For stochastic gradient descent, the size of the batch can't be too large or you risk overshooting the extrema.

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.)

Student Input

1. *place* ω_{t+1}^k from the first equation (of the second formulation) into the second equation (of the second formulation)

start:$\omega_{t+1} = \sum_{k=1}^{k} n_k/n * \omega_{t+1}^k$

substitute

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

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

2. "split" the summation & move constant η outside summation
   $\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$
3. simplifying, recognizing that $\sum_{k=1}^{K} \frac{n_k}{n} \omega_t = \omega_t$
   $\omega_{t+1} = \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$
   which is the most compact form of FedAve, thus completing the proof.

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.

Student Input

The second formulation is more intuitive because it shows how the local client updates through each iteration.

FedAve is an example of the broader concept of federated learning, which describes a decentralized approach to model training across multiple nodes/clients. Federated learning is important to data privacy because it updates the model piece by piece so that there is less risk of the data being exposed if one part is compromised.

Omega represents the model weights, which are updated through each iteration as the model trains. The second formulation explicitly shows how the weighted averages of the locally added omega parameters are summed through each iteration.

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

Student Input

Differential privacy is a category of techniques where noise is added to raw data so that outside observers can't distinguish between points used in the training set and not. Randomized-response differential privacy involves individuals either reporting the true response of a random response determined by some outside influence, such as a coin flip. An algorithm is said to be e-deferentially private if for a subset s outputs

$(P[A(D_1) \in S)/P(P[A(D_2) \in S) <= e^{\epsilon}$

In class, we determined that we would be using binary data (so yes/no).

1. starting with differential privacy example (yes is 1, no is 0)
   $\frac{P(1)=1}{P(1)=0} = \frac{P}{1-P}$
2. set $= e^{\epsilon}$ to prove that it's ϵ - differentially private
   $\frac{P}{1-P} <= e^{\epsilon}$
3. move ϵ down by taking natural log to prove that randomized response is e-differentially private
   $ln\frac{P}{1-P} <= e$

This proof is the generalization of the example we did in class.

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.*)

Student Input

The harm principle as theorized by philosopher JS Mill states that the personal autonomy of a moral agent should be restricted when using said personal autonomy would result in objective harm to another moral agent. Machine learning models have not achieved enough agency or individual autonomy to have the harm

principle be fully applicable to them at the current moment. Machine learning models and algorithms are trained on previous data and while they may be able to mimic a human response to a prompt based on its training data, they currently lack the sentience required to take full moral responsibility for the harm that their actions may cause other moral agents (i.e. human). Since the harm principle isn't directly applicable to ML models but the ML models may still cause harm to other moral agents based on their actions, it is then the responsibility of the developers and implement of these ML models to take responsibility for the harm of these ML models and take steps to ensure the safety of the users of these models.