HW 6

Pranav Kallem

11/17/2024

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.*)

The primary difference between gradient descent (GD) and stochastic gradient descent (SGD) lies in how they compute and utilize the gradient during optimization. Gradient descent calculates the gradient of the cost function using the entire dataset and updates the model parameters accordingly, with the update step given by $\theta = \theta - \eta \cdot \nabla_\theta f(\theta)$, where $\nabla_\theta f(\theta)$ is the gradient based on all data points. This ensures accurate updates but can be computationally expensive for large datasets since each iteration requires processing the full dataset. In contrast, stochastic gradient descent uses a single randomly chosen data point to approximate the gradient, updating the parameters as $\theta = \theta - \eta \cdot \nabla_\theta f_i(\theta)$, where $f_i(\theta)$ is the cost function for one data point. This approach introduces randomness, which can make SGD computationally faster and better at escaping local minima, but the updates are noisier, leading to more fluctuations near the optimal solution. Consequently, GD is precise and stable but computationally heavy, whereas SGD is faster and more scalable but less smooth in convergence.

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.)

We start with the second formulation of the `FedAve` algorithm:

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

Substitute ω_{t+1}^k from the first equation into the second equation:

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

Expand the summation:

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

Since ω_t is independent of k, the first term simplifies to:

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

Thus, the equation reduces to:

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

This matches the compact form of the update step:

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

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 is more intuitive because it explicitly separates the update process into two steps. First, each client k computes a local update

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

reflecting the effect of the gradient on the model parameters for their local data. Then, the server aggregates these updates, weighted by the relative sizes of the clients' datasets, $\frac{n_k}{n}$, to produce the global update

$$\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_{t+1}^k.$$

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

To show that randomized response achieves ϵ-differential privacy, we need to prove that the output probabilities for any two inputs x_1 and x_2 satisfy $\frac{P(Output=y|x_1)}{P(Output=y|x_2)} \leq e^{\epsilon}$. In the randomized-response mechanism, given a binary input $x \in \{0,1\}$, the mechanism outputs $y = x$ with probability p and $y \neq x$ with probability $1-p$. Thus, the probabilities of outputs are $P(y=1 \mid x=1) = p$, $P(y=0 \mid x=1) = 1-p$, $P(y=1 \mid x=0) = 1-p$, and $P(y=0 \mid x=0) = p$. For any output $y \in \{0,1\}$, the privacy ratio is $\frac{P(y=1|x=1)}{P(y=1|x=0)} = \frac{p}{1-p}$. If we set $p = \frac{e^{\epsilon}}{1+e^{\epsilon}}$, this ratio simplifies to $\frac{p}{1-p} = \frac{\frac{e^{\epsilon}}{1+e^{\epsilon}}}{\frac{1}{1+e^{\epsilon}}} = e^{\epsilon}$. Similarly, for $y=0$, the ratio remains bounded by e^{ϵ}. Therefore, randomized response 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 applies to machine learning models if they have enough agency to impact users' autonomy. Agency involves the ability to act independently and make meaningful choices, which requires self-awareness and reasoning. Currently, ML models do not have true agency because they lack self-awareness and intentionality. They work based on programmed algorithms, data, and patterns, without understanding the ethical consequences of their actions. However, ML models can indirectly affect user autonomy if they are biased, unclear, or used by people to restrict freedoms. While ML models themselves are not subject to the harm principle, the responsibility lies with the people who design and deploy them to follow the principle by minimizing harm and protecting user autonomy.