HW 6

Neel Singh

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.*)

Student Response:

Gradient descent computes the gradient using the entire dataset at each step of the algorithm. This ensures that the vector direction of steepest descent is most precise based on all of the data.

The update step is given as

$$\theta_{i+1} = \theta_i - \alpha \nabla f(\theta_i, X, Y)$$

In gradient descent, X and Y represent the entire dataset. However, depending on the value of the step size α, it is possible for the gradient descent algorithm to get "stuck" in a local minima.

This issue can be somewhat alleviated with *stochastic gradient descent*, which increases variability in calculating gradients to reduce the chances of the algorithm getting stuck in local minima. Stochastic gradient descent achieves this by only using a random subset of the data to increase variability.

The update step for stochastic gradient descent is given as

$$\theta_{i+1} = \theta_i - \alpha \nabla f(\theta_i, X_{i'}, Y_{i'})$$

Here, $X_{i'}$ and $Y_{i'}$ represent a random subset of the data. The fact that the algorithm runs on only a subset of the data versus the full dataset X and Y result in stochastic gradient descent's ability to possibly escape local minima and find the global minima more efficiently than regular 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.*)

Student Response:

Whole form:

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

Substitute $\omega_t - \eta \nabla F_k(\omega_t)$ for ω_{t+1}^k:

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

Distribute the summation:

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

Simplify:

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

This is equivalent to the compact form $\omega_{t+1} = \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$

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.

Student Response:

The second formulation splits the update process into two steps, which makes it easier to understand how the algorithm works.

The step $\omega_{t+1}^k = \omega_t - \eta \nabla F_k(\omega_t)$ shows that each client k computes a local model based on the local data. This shows the local gradients $\nabla F_k(\omega_t)$.

The step $\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_{t+1}^k$ shows the averaging of the local updates to create the global update. The local updates are weighted by the proportion of data $\frac{n_k}{n}$ coming from each client k.

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

Student Response:

A randomized algorithm A is ϵ-differentially private if the following is satisfied:

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

Where D_1 and D_2 are datasets differing in exactly one element and subset $S \in im(A)$.

Randomized-response differential privacy has the following outputs:

- Truth (HH)

- Truth (HT)
- Yes (TH)
- No (TT)

The following proof shows that randomized-response differential privacy is ϵ-differentially private for an epsilon of $ln(3)$.

D and $S \in$ {Yes, No}. Assume $S = Yes$.

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

Thus randomized-response differential privacy is ϵ-differentially private for an epsilon of $ln(3)$.

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 Response:

The *harm principle* holds that an individual's personal autonomy ends at the point where their actions cause harm to another individual or group. People are free to act as they like as long as they do not infringe on the wellbeing of others. Agency is the ability of some entity to act with intention and make autonomous decisions. In order to apply the harm principle to assess the reach of the autonomy of an entity, said entity much be able to exhibit agency because an entity cannot be autonomous without agency.

Current machine learning (ML) models, though highly skilled at making decisions based on a stimulus, lack agency. They operate without consciousness, intention, or moral reasoning. A ML model's ability to make decisions depends solely on how another entity trained said model, including choice of training data and type of model. Thus, for a given input, the ML model does not *choose* what decision it makes. It is predisposed to make a decision based on the parameters chosen for it to operate on. As such, ML models operate without consciousness, intention, or moral reasoning, and do not possess sufficient agency for the harm principle to be applied to assess the reach of the autonomy of ML models.

However, ML models can significantly impact others through their applications. The choice to use a ML model is made by a thinking and reasoning (and most likely human) entity. The harm principle *can* be applied to the rational entities commanding the development and use of ML models. For instance, ML models that exhibit algorithmic bias (like COMPAS) can perpetuate discrimination within human society. The autonomous choice to use these biased models (and the choice to design biased models) can be judged by the harm principle. These models are harmful to individuals, and thus, the deployment of said models would not be justified under the harm principle. Restricting the autonomy of those deploying and building harmful models can be justified when it prevents harm to individuals and groups, like with restricting the deployment of COMPAS in courthouses.