HW 6

Logan Schmitt

4/10/2024

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.*)

Logan Input: Gradient descent and stochastic gradient descent (SDG) are both optimization algorithms.

The update step for gradient descent is as follows: $\theta_{i+1} = \theta i - \alpha \nabla F(\theta_i, x, y)$. It computes the gradient using the entire dataset and the current parameters, updates the model's parameters, and passes the results again into the next update step until an objective is met, such as risk minimization. Gradient descent is useful for non-convex functions, such as those with many local minimums, as it converges toward a singular optimization point.

The update step for stochastic gradient descent is as follows: $\theta_{i+1} = \theta_i - \alpha \nabla F(\theta_i, X_i, Y_i)$. Unlike gradient descent, SGD uses a randomly-selected subset of the data instead of the entire dataset. It computes the gradient using this subset, updates the model's parameters, and passes the results again into the next update step until risk is minimized. This is especially useful for more convex functions and not getting captured in local minimums. Whereas gradient descent converges towards a local minimum, SGD avoids this due to the different subsets of data being used, allowing it to approach and find the global minimum.

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.*)

Logan Input:

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

Step 2: Distribution $\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_t - \sum_{k=1}^{K} \eta \frac{n_k}{n} \nabla F_k(\omega_t)$

Step 3: Factorization $\omega_{t+1} = \frac{\omega_t}{n} \sum_{k=1}^{K} n_k - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$

Step 4: Simplification $\omega_{t+1} = \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$

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.

Logan Input: With the second formulation, local update steps are first calculated using the current parameters as well as a random subset of data. In the next update step, these outcomes are passed through the model into the next update step until an objective is optimized (i.e., risk being minimized). As a result, the new update is based on the previous update. With the second formulation, this relationship between the new iteration and the previous iteration is clearly visualized. At the local level, you can see how and where the local update is taking place as well as the stochastic gradient descent being calculated and then passed onto the global update step.

Explain how the harm principle places a constraint on personal autonomy. 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.*)

Logan Input: The harm principle outlines that a moral agent's personal autonomy is exercisable up until the point it causes objective harm to another moral agent. On the surface, it would appear that machine learning models should have to adhere to the harm principle. For example, the COMPAS algorithm specifically influences criminal defendants and their ability to be granted parole. The results from this model can have a significant influence on the defendant's lives, including the ability to negatively affect them or cause harm.

However, a key term within the harm principle definition is "moral agents." A key distinguishing feature of moral agents is having sentience. Sentience encompasses feelings, which includes an awareness of how one's actions can affect another. With this comes the knowledge of moral understanding and responsibility. Since machine learning algorithms lack sentience, the harm principle cannot be extended to these algorithms as they lack this sensation and understanding. Alternatively, machine learning algorithms, therefore, cannot limit the autonomy of their users.

While the harm principle cannot apply to machine learning models, it can apply to the creators of these models. These developers, in effect, have a responsibility to adhere to the harm principle and must not design machine learning models that have the potential to cause harm to other moral agents. For example, the developer cannot use a user's data if it harms the user, therefore limiting the autonomy of the developer in this scenario. Likewise, the develops, not the machine learning models, have the agency to limit the autonomy of the users of their algorithms.