Stratejiler ve Detayları

Stratejiler, Federe Öğrenmede server tarafında kullanılan optimizerlardır. Bir takım kıstaslara göre strateji seçimi yapılabilir. Bu kıstaslar, özellikle sistem heterojenliği ve istatistiksel heterojenliği içerir.

FedAVG

FedAVG, eğitime katılan istemcileri rastgele seçer. Ancak ağ yapısı karmaşık olduğunda ve veriler bağımsız ve aynı şekilde dağıtılmadığında, FedAvg iyi performans gösteremeyebilir. FedAvg toplaması, her yerel model için model parametrelerinin eşit dağılımından oluşur. Parametrelerin, yüklenen tüm modeller arasında ortalaması alınır.

FedProx

Diğer bir strateji, veri ve sistemlerin heterojenliğini ele almak için bazı değişikliklerle FedAvg'nin bir genellemesi olan FedProx'tur. Teorik olarak, aynı olmayan dağılımlardan (istatistiksel heterojenlik) verileri öğrenirken ve katılan her cihazın değişken miktarda iş yapmasına (sistem heterojenliği) izin vererek cihaz düzeyinde sistem kısıtlamalarına bağlı kalırken çerçevemiz için yakınsama garantileri sağlarız. Pratik olarak, FedProx'un bir dizi gerçekçi birleşik veri kümesinde FedAvg'den daha güçlü yakınsamaya izin verdiğini gösteriyoruz. Özellikle, son derece heterojen ortamlarda, FedProx, FedAvg'ye göre önemli ölçüde daha kararlı ve doğru yakınsama davranışı gösterir --- mutlak test doğruluğunu ortalamada %22 artırır. FedAvg, heterojen ortamlarda deneysel başarı göstermiş olsa da, heterojenlikle ilişkili temel zorlukları tam olarak ele almıyor. Sistem heterojenliği bağlamında, FedAvg, katılımcı cihazların temel sistem kısıtlamalarına dayalı olarak değişken

miktarlarda yerel çalışma gerçekleştirmesine izin vermez; bunun yerine, belirli bir zaman aralığında E dönemlerini hesaplayamayan cihazları basitçe bırakır yani elimine eder. **FedProx'u geliştirirken sahip olduğumuz önemli bir kavrayış, birleşik öğrenmede sistemler ve istatistiksel heterojenlik arasında bir etkileşimin var olduğudur. Gerçekten de, hem geride kalanları düşürmek (FedAvg'de olduğu gibi) hem de geriye kalanlardan alınan kısmi bilgileri safça birleştirmek (yakın terim 0'a ayarlanmış FedProx'ta olduğu gibi) örtük olarak istatistiksel heterojenliği arttırır ve yakınsama davranışını olumsuz etkileyebilir.**
Ayrıca, heterojen sistem kaynaklarına sahip federe ağlarda, yerel dönemlerin sayısının yüksek olarak ayarlanması, cihazların belirli bir iletişim turunda eğitimi tamamlamama riskini artırabilir ve bu nedenle prosedürden çıkmak zorunda kalabilir (Bonawitz ve diğerleri, 2019).
.
Pratikte, bu nedenle, yerel dönemleri yüksek olacak şekilde (iletişimi azaltmak için) ayarlamanın ve aynı zamanda sağlam yakınsamaya izin vermenin bir yolunu bulmak önemlidir. Daha temel olarak, yerel dönemlerin sayısı için "en iyi" ayarın, hem yerel verilerin hem de mevcut sistem kaynaklarının bir işlevi olarak, her yinelemede ve her cihazda muhtemelen değişeceğini not ediyoruz. Gerçekten de, sabit sayıda yerel dönemi zorunlu kılmaktan daha doğal bir yaklaşım, dönemlerin ağın özelliklerine göre değişmesine izin vermek ve bu heterojenliği hesaba katarak çözümleri dikkatli bir şekilde birleştirmektir. Önerilen çerçevemiz FedProx (Algoritma 2), her turda bir cihaz alt kümesinin seçilmesi, yerel güncellemelerin yapılması ve daha sonra bu güncellemelerin bir global güncelleme oluşturmak için ortalamasının alınması bakımından FedAvg'a benzer. Bununla birlikte, FedProx, önemli deneysel iyileştirmelerle sonuçlanan ve aynı zamanda yöntem için yakınsama garantileri sağlamamıza izin veren aşağıdaki basit ama kritik değişiklikleri yapar.

Kısmi çalışmayı tolere etmek. Daha önce tartışıldığı gibi, federe ağlardaki farklı cihazlar genellikle bilgi işlem donanımı, ağ bağlantıları ve pil seviyeleri açısından farklı kaynak kısıtlamalarına sahiptir. Bu nedenle, FedAvg'de olduğu gibi her aygıtı tek tip bir iş yapmaya zorlamak (yani, aynı sayıda yerel dönem çalıştırmak,E) gerçekçi değildir. FedProx'ta, mevcut sistem kaynaklarına dayalı olarak cihazlar arasında yerel olarak gerçekleştirilecek değişken miktarlarda işlere izin vererek FedAvg'yi genelleştirir ve ardından (bu cihazları düşürmeye kıyasla) geride kalanlardan gönderilen kısmi çözümleri toplarız. Diğer bir deyişle, eğitim süreci boyunca tüm cihazlar için tek tip bir γ varsaymak yerine, FedProx, farklı cihazlar ve farklı yinelemelerde değişken γ'leri örtük olarak barındırır.

PROXIMAL TERM ÖNEMLİ

Proximal Term iki açıdan faydalıdır: (1) Yerel dönemlerin sayısını manuel olarak ayarlamaya gerek kalmadan yerel güncellemeleri ilk (küresel) modele daha yakın olacak şekilde kısıtlayarak istatistiksel heterojenlik sorununu ele alır. (2) Sistem heterojenliğinden kaynaklanan değişken miktarlardaki yerel çalışmanın güvenli bir şekilde dahil edilmesini sağlar.

Fed-CS

Protocol 2 Federated Learning with Client Selection. K is the number of clients, and $C \in (0, 1]$ describes the fraction of random clients that receive a resource request in each round.

1: `Initialization` in Protocol 1.
2: `Resource Request`: The MEC operator asks $\lceil K \times C \rceil$ random clients to participate in the current training task. Clients who receive the request notify the operator of their resource information.
3: `Client Selection`: Using the information, the MEC operator determines which of the clients go to the subsequent steps to complete the steps within a certain deadline.
4: `Distribution`: The server distributes the parameters of the global model to the selected clients.
5: `Scheduled Update and Upload`: The clients update global models and upload the new parameters using the RBs allocated by the MEC operator.
6: `Aggregation` in Protocol 1.
7: All steps but `Initialization` are iterated for multiple rounds until the global model achieves a desired performance or the final deadline arrives.

FedCS'yi Protokol 2'de sunuyoruz. Protokolümüzün ana fikri, Protokol 1'in orijinal İstemci Seçimi adımında rastgele istemciler seçmek yerine, aşağıdaki iki aşamalı istemci seçim şemasını önermemizdir.

İlk olarak, yeni Kaynak Talebi adımı rastgele istemcilerden kablosuz kanal durumları, hesaplama kapasiteleri ve mevcut eğitim göreviyle ilgili veri kaynaklarının boyutu gibi kaynak bilgileri hakkında MEC operatörüne bilgi vermelerini ister.

Ardından operatör, Dağıtım ve Planlanmış Güncelleme ve Yükleme adımları için gereken süreyi tahmin etmek ve hangi istemcilerin bu adımlara gideceğini belirlemek için sonraki İstemci Seçimi adımında bu bilgilere başvurur (programlama istemcileri için özel algoritmalar daha sonra açıklanacaktır).

Dağıtım adımında, aynı içeriğin (yani küresel modelin) istemci popülasyonlarına iletilmesi için bant genişliği etkili olduğu için, BS'den çok noktaya yayın yoluyla bir küresel model seçilen istemcilere dağıtılır.

Planlanmış Güncelleme ve Yükleme adımında, seçilen istemciler modeli paralel olarak günceller ve MEC operatörü tarafından tahsis edilen RB'leri kullanarak sunucuya yeni parametreler yükler.

Sunucu, Protokol 1'i izleyerek istemci güncellemelerini toplar ve belirli doğrulama verileriyle model performanslarını ölçer. Model belirli bir istenen performansı elde edene kadar (örneğin, %90'lık bir sınıflandırma doğruluğu) veya son teslim tarihi gelene kadar, Başlatma dışındaki tüm adımlar birden fazla tur için yinelenir.

FedAdam, FedYOGI, FedAdaGrad

** Not : Adaptif yöntemlerde, her bir ağırlık için ayrı bir öğrenme oranı uyarlanır.

Client tarafında SGD veya Adam kullanırlar. Her turda aynı miktarda client hesaplaması ve iletişimi gerçekleştirirler. ClientOpt'u SGD olarak tutarken sunucu tarafı momentum veya uyarlanabilir yöntemler kullanmanın bir başka yararı, istemcilerin hesaplama karmaşıklığını veya tur başına iletişim maliyetini artırmamasıdır. Ayrıca son derece düşük müşteri örnekleme oranı (%1'den az) ile uyumludur. Yukarıdaki faydaların tümü, cihazlar arası FL ayarlarında sunucu tarafı momentumun veya uyarlanabilir yöntemlerin faydasını vurgular. Sunucu tarafı momentum ve uyarlanabilir yöntemler göz önüne alındığında, ortaya çıkan doğal bir soru, momentumu veya uyarlanabilirliği doğrudan istemcilere nasıl uygulayabiliriz? Sezgisel olarak, her adımda birinci ve ikinci dereceden anları kullanmak, sunucu tarafı uyarlamalı yöntemlerde olduğu gibi her τ adımında kullanmaktan daha iyi olabilir. Ancak, istemciler yerel olarak momentum veya uyarlamalı optimizasyon yöntemleri uyguladığında, optimize edici durumları ayrı olarak güncellenir ve bu nedenle, IID olmayan veri dağılımları nedeniyle birbirinden sapabilir. Bu sorunu çözmek için, birkaç çalışma, hem yerel modellerin hem de istemci optimize edici durumlarının sunucu tarafından her turda senkronize edildiği senkronize durumlar stratejisini önerdi. Örnekler arasında, her ikisi de senkronize stratejiyi kullanarak ve tüm müşterilerin katılımını varsayarak müşterilere ivme uygulayan [288] ve [291] sayılabilir. Son zamanlarda, Wang ve ark. [263], istemci iyileştirici durumları her turun başında varsayılan değerlere sıfırlandığında bile, istemci uyarlamalı yöntemlerin faydalı olabileceğini ve performansı daha da artırmak için sunucu uyarlamalı yöntemlerle birleştirmenin mümkün olduğunu göstermektedir. Ayrıca, bu sıfırlama durumları stratejisi, sunucu ve istemciler arasında herhangi bir ek iletişime neden olmaz.

Momentum ve uyarlanabilirlik için tembel güncelleme çerçeveleri

Sunucu ve istemci optimizasyon çerçevesinin ötesinde, uyarlamalı optimizasyon yöntemlerini FL'ye uygulamak için başka yaklaşımlar da vardır.
İstemci optimize edici durumları arasındaki sapmaları önlemek için (yani, uyarlamalı optimize edicilerde birinci ve ikinci dereceden anlar), yerel eğitim sırasında optimize edici durumlarını düzeltmek ve bunları her turun sonunda tembelce sunucuda güncellemek yaygın bir fikirdir.

Yukarıda bahsedilen sunucu tarafı uyarlamalı yöntemlerle karşılaştırıldığında, iki önemli fark vardır: (i) Tembel güncelleme algoritmaları, istemcilerin yerel olarak momentum veya uyarlamalı yöntemler kullanmasına izin verir, ancak optimize edici durumları sabittir veya

senkronize tutulur; (ii) Bu tembel güncelleme algoritmalarında, optimize edici durumları, tüm istemcilerin yerel optimize edici durumlarını senkronize ederek veya mevcut global modelde değerlendirilen toplu gradyan kullanılarak sunucuda güncellenir.

FedBN

FedBN olarak adlandırılan verimli ve etkili bir öğrenme stratejisi öneriyoruz. FedAvg'ye benzer şekilde, FedBN yerel güncellemeler gerçekleştirir ve yerel modellerin ortalamasını alır. Ancak FedBN, yerel modellerin BN katmanlarına sahip olduğunu varsayar ve parametrelerini ortalama alma adımından hariç tutar. Algoritmanın tamamını Ek C'de sunuyoruz. Bu basit değişiklik, iid olmayan ortamlarda önemli deneysel iyileştirmelerle sonuçlanır.
Gayri resmi olarak, FedBN, her model için yerel parametreleri korumak yerine toplu normalleştirme parametrelerini ortalama alma adımından hariç tutarak FederatedAveraging'i genişletir. Toplu normalleştirme [44], bir katmanın aktivasyonlarını standart hale getirmek, daha hızlı ve daha düzgün eğitim, aktivasyonları yeniden merkezleme ve yeniden ölçeklendirme yoluyla daha iyi genelleme yeteneği elde etmek için bir yöntemdir.

FedOpt

FL'de iletişim verimliliğini ve gizliliğin korunmasını teşvik etmek için yeni bir yaklaşım, yani Federated Optimization (FedOpt) öneriyoruz. FedOpt'u uygulamak için, verimli iletişim için Seyrek Sıkıştırma Algoritması (SCA) adlı yeni bir sıkıştırma algoritması tasarlıyoruz ve ardından verilerin sızmasını önlemek için ek homomorfik şifrelemeyi diferansiyel gizlilikle entegre ediyoruz. Böylece, önerilen FedOpt, öğrenme görevini benimsemek için iletişim verimliliğini ve gizliliğin korunmasını sorunsuz bir şekilde dengeler. Deneysel sonuçlar, FedOpt'un en gelişmiş FL yaklaşımlarından daha iyi performans gösterdiğini göstermektedir. Özellikle üç farklı değerlendirme kriterini ele alıyoruz; model doğruluğu, iletişim verimliliği ve hesaplama yükü. Ardından, önerilen FedOpt'u, bu üç değerlendirme kriterinin tamamında temel yapılandırmalar ve son teknoloji yaklaşımlarla, yani Birleşik Ortalama (FedAvg) ve paillier şifreleme tabanlı gizliliği koruyan derin öğrenme (PPDL) ile karşılaştırırız. Deneysel sonuçlar, FedOpt'un daha az eğitim dönemi ve daha küçük bir gizlilik bütçesi içinde bir araya gelebildiğini gösteriyor.

FedCurv

Federated Learning için, EWC algoritmasını sıralı bir algoritmadan paralel bir algoritmaya uyarlıyoruz. Bu senaryoda, FedAvg'de olduğu gibi yerel modellerle iletişim kurmaya ve ortalamalarını almaya devam ediyoruz, ancak her yerel modeli diğer tüm cihazların bilgisini

korumaya zorlamak için EWC cezasını da ekliyoruz. İletişim sırasında her cihaz kendi modelini ve modelin Fisher bilgi matrisini diyagonal olarak gönderir.

FedRad

Tipik olarak model güncellemelerini toplayan işbirlikçi öğrenme çerçevesi, rakip istemcilerden gelen model zehirlenmesi saldırılarına karşı savunmasızdır. Global sunucu ve katılımcılar arasında paylaşılan bilgiler sadece model parametreleri ile sınırlı olduğundan hatalı model güncellemelerini tespit etmek zordur. Ayrıca, gerçek dünya veri kümeleri genellikle heterojendir ve bağımsız değildir ve katılımcılar arasında özdeş olarak dağıtılmaz (IID olmayan), bu da bu tür sağlam FL boru hattının tasarımını zorlaştırır. Bu çalışmada, rakipleri tespit etmek ve medyan istatistiğin özelliklerine dayalı olarak yerel modelleri sağlam bir şekilde toplamak ve ardından topluluk Bilgi Damıtma'nın uyarlanmış bir sürümünü gerçekleştirmek için yeni bir sağlam toplama yöntemi olan Federated Robust Adaptive Distillation (FedRAD) öneriyoruz. Önerilen yöntemi yakın zamanda yayınlanan eserlere karşı değerlendirmek için kapsamlı deneyler yapıyoruz. Sonuçlar, FedRAD'ın düşmanların varlığında ve heterojen veri dağıtımlarında diğer tüm toplayıcılardan daha iyi performans gösterdiğini gösteriyor.