Popularity Prediction with Federated Learning for Proactive Caching at Wireless Edge

Kaiqiang Qi and Chenyang Yang
Beihang University, Beijing, China
Email: {kaiqiangqi, cyyang}@buaa.edu.cn

***Abstract*—File popularity prediction plays an important role in proactive edge caching. The widely-used methods for popularity prediction are based on centralized learning, which needs to collect the request information and even more personal information from users, incurring the privacy-disclosure risk. In this paper, we propose a method of predicting file popularity with federated learning to address the privacy issue, where the request data of each user for each file is only employed for local training at each user. To facilitate the popularity prediction at the base station (BS) without information disclosure as well as the supervised training of a neural network at each user, we let each user upload a weighted sum of its own preference and file popularity to the BS. The neural network is employed to predict the weighted sum at each user by training with the user's historical request records for each file. The local models and uploaded results of the users are aggregated at the BS. We show the convergence of the proposed popularity prediction method with a synthetic dataset. We use simulation results with a real dataset to show that the proposed method performs closely to the centralized learning based method in terms of caching performance.**

***Index Terms*—Federated learning, file popularity prediction, proactive caching at wireless edge.**

I. Introduction

Proactive caching at the wireless edge, say at base stations (BSs) and users, has been shown as an effective technique to alleviate traffic load and improve user experience [1, 2], if file popularity is known a priori. In practice, the future file popularity is never known in advance, and hence popularity prediction is a key task to achieve the promising gain of proactive edge caching.

Centralized machine learning has been used to make the prediction, where the file popularity is predicted either at a BS or a central processor (CP). In [2–5], the file popularity was respectively predicted with collaborative filtering, linear regression and neural network (NN) by using the historical numbers of requests for each file gathered from all users. In [6], multi-arm bandit was used to predict the popularity by using context information, such as age and gender of a user. The centralized learning framework needs to collect the file request information (say the number of requests of each user for a file [2]) or even more personal information (such as age and gender [6]). The users may not be willing to share such information with the BS or CP, with which the user preference for contents can be inferred. This concern will hinder the application of content caching at the wireless edge. Therefore, there is an urgent need to design a distributed learning framework where the data with personal information is only stored and used at the user side.

Federated learning (FL) is a distributed framework that can learn a global model at a server while protecting the privacy of the participants, which allows each participant to share the locally-trained model parameters instead of its local data with the server [7]. Recently, federated learning has been introduced to address wireless problems [8–12]. In [10], a deep reinforcement learning framework with federated learning was designed to make the decisions for proactive caching at the BS (and for computing offloading at each user), where each BS is a participant to optimize the caching policy using user demands data and the server in the cloud aggregates the uploaded models from the BSs. In [11], a FL-based deep echo state network was used at a BS to predict the mobility and orientation of each user for proactive virtual reality steaming, where each BS is a participant to predict user mobility with user locations and a server covering multiple cells is used for model aggregation. Since the BS is a participant for making the cache decision in [10] and for predicting user mobility in [11], the request or location data from each user is stored at the BS, which still faces the risk of privacy disclosure. In [12], federated learning was introduced for proactive caching, where the caching decision is made at the BS. Except the model parameters of each participant (i.e., each user), other information (i.e., a recommended list of the preferred files of each user) is also uploaded to the BS for making the decision [12]. Yet such a file list discloses the preference of a user.

In this paper, we use federated learning to predict file popularity for proactive edge caching without disclosing user preference. We take the request data as an example to illustrate the FL-framework for popularity prediction. To hide the individual preference of each user meanwhile to facilitate the popularity prediction at the BS and the local training at each user, we first determine what information of each user should be uploaded to the BS. We show that if each user uploads a weighted sum of user preference and file popularity, as well as the total number of requests for all files, the file popularity can be predicted at the BS if the file popularity, user preference and activity level are static. We then design a NN for each user to predict the preference-weighted popularity, which is locally trained with its own historical request records and is aggregated at the BS. We demonstrate that the cache-hit ratio of the FL-based caching policy is close to that of the

This work is supported by National Natural Science Foundation of China (NSFC) under Grants 61731002 and 61671036.

978-1-7281-3106-1/20/$31.00 ©2020 IEEE

centralized learning based policy by using a real dataset with dynamic popularity for training the NN at each user.

II. System Model

Consider a network where each BS is equipped with a cache and is connected to the core network via backhaul. In each cell, there are K users, each initiates requests for files from a library with F files. A server for mobile edge computing (MEC) is co-located at each BS, which predicts the file popularity of the cell periodically to update a proactive caching policy.

Time is discretized into time periods, each with cache update duration. File popularity of a cell depends on the individual preference and activity level of every user in the cell [13]. Denote the *file popularity* of a cell in the tth time period as $\mathbf{p}^t = [p_1^t, \cdots, p_F^t]$, $p_f^t \in [0,1]$ and $\sum_{f=1}^F p_f^t = 1$, where p_f^t is the probability that the fth file is requested by all the K users in the cell. Denote the *user preference* of the kth user in the tth time period as $\mathbf{q}_k^t = [q_{k,1}^t, \cdots, q_{k,F}^t]$, $q_{k,f}^t \in [0,1]$ and $\sum_{f=1}^F q_{k,f}^t = 1$, where $q_{k,f}^t$ is the probability that the kth user sends a request for the fth file. In practice, some users are very active and others are not. The user *activity level* at the tth time period is denoted as $\mathbf{a}^t = [a_1^t, \cdots, a_K^t]$, $a_k^t \in [0,1]$ and $\sum_{k=1}^K a_k = 1$, where a_k^t is the probability that a request is sent from the kth user. Then, the file popularity of a cell in the tth time period can be obtained from the activity level and preference of the users initiating requests in the cell as [13],

$$\mathbf{p}^t = \sum_{k=1}^{K} a_k^t \mathbf{q}_k^t. \tag{1}$$

We call the number of requests of each user for each file in each time period as a request record. When using FL for file popularity prediction, the request records of each user are only stored in the user's own device [7]. At the beginning of each time period, the MEC server broadcasts the aggregated file popularity and model parameters of machine learning (say a NN) to the users, and then each user trains a local model using its own historical request records with the aggregated model parameters as initialization. At the end of each time period, each user sends the updated model parameters as well as other required information to the MEC server, and then the MEC server predicts the file popularity by aggregation, aggregates the model parameters, and optimizes the caching policy with the predicted popularity. This completes one round of communication between each user and the server. The procedure of predicting file popularity with FL is shown in Fig. 1. In the rest of the paper, we consider one cell for illustration, and call the MEC server as the BS.

III. Predicting Popularity with Federated Learning

In this section, we propose a method of predicting file popularity for proactive caching with federated learning.

A. Uploaded Information of Each User

If only the model parameters are sent to the BS, which do not contain any user request information, the privacy can be protected but the file popularity cannot be predicted. To facilitate popularity prediction at the BS without user information disclosure, we need to determine what kind of information should be uploaded to the BS.

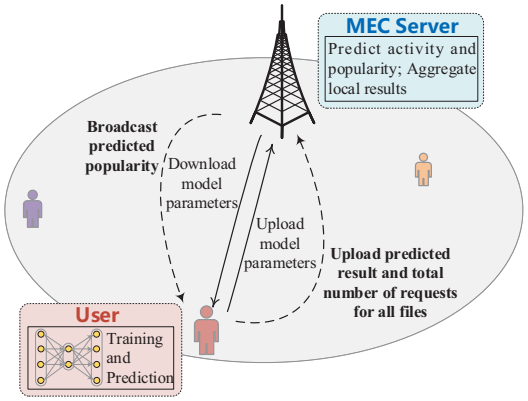

Fig. 1. Procedure of predicting file popularity for caching with FL.

A straightforward way is to let each user upload the predicted popularity. However, it is impossible for each user to obtain the label for supervising the training procedure, since each user only has its own request records. If each user employs the aggregated popularity broadcasted by the BS as the label, then finally only the initialized popularity can learned, as to be clear in the following two remarks.

Another way is to let each user upload the predicted user preference to the BS. This is technically viable because the user preference estimated from the request records of the user can be used as the label for supervised learning. With the predicted user preference as well as other required information able to reflect user activity level, the BS can predict file popularity. Alternatively, each user can upload a recommendation list of the most preferred files according to the predicted user preference, and the BS can obtain a caching policy by aggregating the recommended files from multiple users. However, both ways for uploading disclose the preferred files of the user.

To hide the personal information, we let each user predict and upload a weighted sum of user preference and file popularity. For the kth user, the *preference-weighted popularity* in the tth time period is,

$$\mathbf{s}_k^t = \omega \mathbf{q}_k^t + (1-\omega)\hat{\mathbf{p}}^t, \quad \omega \in (0,1), \tag{2}$$

where $\hat{\mathbf{p}}^t$ is the predicted popularity available at every user after the BS broadcasts it at the beginning of the tth communication round, ω is a weighting coefficient that can be adjusted to balance convergence and accuracy, and $\sum_{f=1}^F s_{k,f}^t = 1$.

In what follows, we show that the file popularity can be predicted at the BS by aggregating the preference-weighted popularity predicted at every user, if file popularity, user preference and activity level are static.

To this end, we first assume that each user knows the ground truth of its own preference (say, the kth user knows $\mathbf{q}_k$) and the BS knows the ground truth of the user activity

level $[a_1,\cdots,a_K]$. Then, the ground truth of file popularity is $\mathbf{p}=\sum_{k=1}^{K}a_k\mathbf{q}_k$ according to (1). At the end of the tth communication round, each user (say the kth user) uploads the predicted preference-weighted popularity $\hat{\mathbf{s}}_k^{t+1}$ (i.e., the weighted sum of $\mathbf{q}_k$ and the broadcasted popularity by the BS in the tth period $\hat{\mathbf{p}}^t$). Then, the BS can predict the popularity in the $(t+1)$th period by aggregating the uploaded results as,

$$\begin{aligned}\hat{\mathbf{p}}^{t+1}=\sum_{k=1}^{K}a_k\hat{\mathbf{s}}_k^{t+1}&=\sum_{k=1}^{K}a_k(\omega\mathbf{q}_k+(1-\omega)\hat{\mathbf{p}}^t)\\&=\omega\mathbf{p}+(1-\omega)\hat{\mathbf{p}}^t.\end{aligned}\tag{3}$$

If in the first communication round the predicted popularity $\hat{\mathbf{p}}^1$ is initialized as a random distribution, then from (3) the predicted popularity in the $(t+1)$th round is,

$$\hat{\mathbf{p}}^{t+1}=(1-(1-\omega)^t)\mathbf{p}+(1-\omega)^t\hat{\mathbf{p}}^1.\tag{4}$$

Remark 1. When $\omega\in(0,1)$, we can obtain $\lim_{t\to\infty}\hat{\mathbf{p}}^{t+1}=\mathbf{p}$ from (4), i.e., the predicted file popularity can converge to the ground truth of the file popularity. If $\omega=0$, i.e., each user uploads the predicted popularity, then $\lim_{t\to\infty}\hat{\mathbf{p}}^{t+1}=\hat{\mathbf{p}}^1$, i.e., the finally predicted file popularity at the BS is irrelevant to the ground truth of the popularity.

Next, we remove the assumption on knowing the ground truth. To predict the file popularity, the BS needs to predict the user activity level. To predict the preference-weighted popularity, each user needs to predict its own preference implicitly. Since the activity level and preference of each user are assumed static, the BS can use the estimated activity level as the predicted value, and each user can employ the estimated preference as the predicted preference. At the end of the tth communication round, each user uploads the predicted preference-weighted popularity, as well as its total number of requests for all files in the round to assist the BS in estimating activity level. Then, the BS can predict the popularity in the $(t+1)$th period by aggregating the uploaded results as,

$$\begin{aligned}\hat{\mathbf{p}}^{t+1}=\sum_{k=1}^{K}\hat{a}_k^{t+1}\hat{\mathbf{s}}_k^{t+1}&=\sum_{k=1}^{K}\bar{a}_k^t(\omega\bar{\mathbf{q}}_k^t+(1-\omega)\hat{\mathbf{p}}^t)\\&\overset{(b)}{=}\omega\bar{\mathbf{p}}^t+(1-\omega)\hat{\mathbf{p}}^t,\end{aligned}\tag{5}$$

where $\bar{\mathbf{q}}_k^t=\mathbf{r}_k^t/\|\mathbf{r}_k^t\|_1$ is the estimated preference of the kth user in the tth time period, $\mathbf{r}_k^t=[r_{k,1}^t,\cdots,r_{k,F}^t]$, $r_{k,f}^t$ is the number of requests for the fth file of the kth user in the tth period, $\|\cdot\|_1$ is the l_1 norm, $\bar{\mathbf{p}}^t$ in step (b) is the estimated popularity using (1) and $\bar{a}_k^t=\|\mathbf{r}_k^t\|_1/\sum_{k=1}^{K}\|\mathbf{r}_k^t\|_1$ is the estimated activity level of the kth user in the tth period. From (5), the predicted popularity can be derived as,

$$\hat{\mathbf{p}}^{t+1}=\omega\sum_{i=1}^{t}(1-\omega)^{t-i}\bar{\mathbf{p}}^i+(1-\omega)^t\hat{\mathbf{p}}^1,\tag{6}$$

Remark 2. When $\omega\in(0,1)$, we can obtain $\lim_{t\to\infty}\hat{\mathbf{p}}^{t+1}=\omega\sum_{i=1}^{t}(1-\omega)^{t-i}\bar{\mathbf{p}}^i$ from (6). This indicates that the popularity is predicted by linear regression with the historically estimated popularity, and the regression coefficient exponentially decays with the time period where the popularity is estimated. If $\omega=1$, then $\lim_{t\to\infty}\hat{\mathbf{p}}^{t+1}=\bar{\mathbf{p}}^t$, i.e., the predicted popularity is simply the estimated popularity in previous time period. If $\omega=0$, then $\lim_{t\to\infty}\hat{\mathbf{p}}^{t+1}=\hat{\mathbf{p}}^1$, i.e., the predicted popularity is still the initialized random distribution.

In practice, file popularity is dynamic, which needs more complex models to make the prediction. Besides, uploading the preference-weighted popularity predicted as in (5) may still face the privacy risk, since the BS can guess the preference of each user according to (2) by subtracting the broadcasted popularity. In what follows, we employ NN at each user as an illustrative technique to predict the preference-weighted popularity. In this way, dynamic popularity can be predicted, more user information such as age and gender can be incorporated, and the preference of each user is hard to be obtained by the BS. For illustration, we still employ the request record as the input of the NN.

B. Predicting Preference-weighted Popularity at Each User

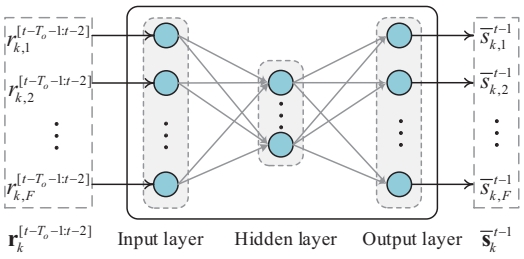

Fig. 2. NN at each user for predicting the preference-weighted popularity with the past records of numbers of requests.

1) Structure, Input and Output of NN: We adopt the multi-layer perception as an illustrative NN to make the prediction at each user using the historical numbers of requests in an observation window with T_o time periods, as shown in Fig. 2. We select the `sigmoid` function (i.e., $y=\frac{1}{1+e^{-x}}$ [14]) as the activation function for the hidden layer, which performs similarly to other activation functions such as tanh and ReLU, according to our simulations. To yield the probability distribution, we select the `softmax` function (i.e., $y_i=\frac{e^{x_i}}{\sum_{f=1}^{F}e^{x_f}}, i=1,\cdots,F$ [14]) in the output layer.

A sample to train the NN at the kth user consists of a F-dimension input vector $\mathbf{x}_k$ and the F-dimension expected output vector $\mathbf{y}_k$ (i.e., the label). The sample used for training in the tth communication round is updated at the end of the $(t-1)$th communication round, since the request data in the $(t-1)$th time period can be measured and used to generate the label for training. Specifically, the training sample for the kth user in the tth round is $\{\mathbf{x}_k,\mathbf{y}_k\}=\{\mathbf{r}_k^{[t-T_o-1:t-2]},\bar{\mathbf{s}}_k^{t-1}=\omega\bar{\mathbf{q}}_k^{t-1}+(1-\omega)\hat{\mathbf{p}}^{t-1}\}$, if there exists at least one request for a file in the library both in the observation window and the $(t-1)$th time period, i.e., $\|\mathbf{r}_k^{[t-T_o-1:t-2]}\|_1>0$ and $\|\mathbf{r}_k^{t-1}\|_1>0$, where the fth element of $\mathbf{r}_k^{[t-T_o-1:t-2]}$ is $r_{k,f}^{[t-T_o-1:t-2]}=\sum_{i=t-T_o-1}^{t-2}r_{k,f}^i$. Otherwise, $\{\mathbf{x}_k,\mathbf{y}_k\}=\emptyset$.

In the beginning and end of each communication round, each user re-trains its NN with the updated sample and predicts the preference-weighted popularity during the next communication round, respectively. Note that both the training and prediction phases can be executed at the off-peak hours. The procedure for training and prediction is shown in Fig. 3.

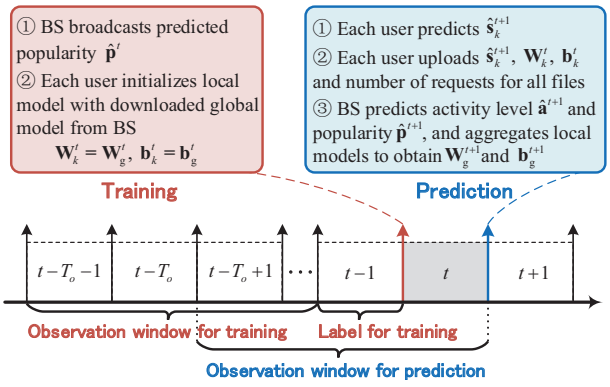

Fig. 3. Training and prediction in the tth communication round.

2) Training in Each Round: At the beginning of the tth communication round, the NN at each user (say the kth user) is initialized with the downloaded global model parameters $\mathbf{W}_g^t$ and $\mathbf{b}_k^t$ from the BS, i.e., $\mathbf{W}_k^t = \mathbf{W}_g^t$ and $\mathbf{b}_k^t = \mathbf{b}_g^t$.

Then, the NN at the kth user is trained to minimize the cross entropy between its output and expected output, i.e.,

$$J(\mathbf{W}_k^t, \mathbf{b}_k^t) = \sum_{f=1}^{F} y_{k,f} \log(\hat{y}_{k,f}), \tag{7}$$

where $y_{k,f}$ and $\hat{y}_{k,f}$ are the fth element of the expected output and the output of the NN, respectively. The stochastic gradient descent (SGD) algorithm [14] is used to update the model parameters, which can be obtained as follows in the nth iteration,

$$\begin{aligned} \mathbf{W}_k^{(n+1)} &= \mathbf{W}_k^{(n)} - \eta \nabla_{\mathbf{W}_k^t} J(\mathbf{W}_k^{(n)}, \mathbf{b}_k^{(n)}), \\ \mathbf{b}_k^{(n+1)} &= \mathbf{b}_k^{(n)} - \eta \nabla_{\mathbf{b}_k^t} J(\mathbf{W}_k^{(n)}, \mathbf{b}_k^{(n)}), \end{aligned} \tag{8}$$

where $\nabla_{\mathbf{W}_k^t}$ and $\nabla_{\mathbf{b}_k^t}$ are respectively the gradients with respect to $\mathbf{W}_k^t$ and $\mathbf{b}_k^t$, which can be obtained by back-propagation, and η is the learning rate.

3) Prediction in Each Round: At the end of the tth communication round, we input the numbers of requests of the kth user for every file in the observation window, $\mathbf{r}_k^{[t-T_o+1:t]}$, into the trained NN. Then, the preference-weighted popularity in the $(t+1)$th round can be predicted at the kth user as,

$$\hat{\mathbf{s}}_k^{t+1} = f(\mathbf{r}_k^{[t-T_o+1:t]}, \mathbf{W}_k^t, \mathbf{b}_k^t), \tag{9}$$

where $f(\cdot)$ is the mapping function of the NN.

C. Predicting File Popularity at the BS

At the end of each communication round, the BS predicts the file popularity and aggregates the local models. To this end, each user uploads the trained model parameters, the predicted preference-weighted popularity, and the total numbers of requests for all files in the observation window.

1) Activity Level and Popularity Prediction: To predict the file popularity, the BS needs to predict the user activity level. A simple predictor is to use the estimated activity level in the observation window as the predicted value, i.e.,

$$\hat{a}_k^{t+1} = \frac{\|\mathbf{r}_k^{[t-T_o+1:t]}\|_1}{\sum_{k=1}^{K} \|\mathbf{r}_k^{[t-T_o+1:t]}\|_1}, \quad k = 1, \cdots, K, \tag{10}$$

where $\|\mathbf{r}_k^{[t-T_o+1:t]}\|_1$ is the total number of requests for all files in the observation window uploaded by the kth user at the end of the tth round. We can also use a NN to predict the activity level using the estimated values in the past at the BS.

With the predicted activity level, the BS can predict the file popularity by aggregating the predicted preference-weighted popularity uploaded from every user as,

$$\hat{\mathbf{p}}^{t+1} = \sum_{k=1}^{K} \hat{a}_k^{t+1} \hat{\mathbf{s}}_k^{t+1} \overset{(c)}{=} \omega \sum_{k=1}^{K} \hat{a}_k^{t+1} \hat{\mathbf{q}}_k^{t+1} + (1-\omega) \sum_{k=1}^{K} \hat{a}_k^{t+1} \hat{\mathbf{p}}_k^{t+1}, \tag{11}$$

where step (c) is obtained from (9) and (2), from which we can observe that the finally predicted popularity at the BS is a weighted sum of two kinds of aggregated results. The first is the predicted popularity with the implicitly predicted preference $\hat{\mathbf{q}}_k^{t+1}$. The second is the aggregated popularity with the implicitly predicted popularity at each user $\hat{\mathbf{p}}_k^{t+1}$.

2) Model Aggregation: Considering that the model can be better trained for the more active user, the BS aggregates the trained models of all users as,

$$\mathbf{W}_g^{t+1} = \sum_{k=1}^{K} \hat{a}_k^{t+1} \mathbf{W}_k^t, \quad \mathbf{b}_g^{t+1} = \sum_{k=1}^{K} \hat{a}_k^{t+1} \mathbf{b}_k^t. \tag{12}$$

The procedure of predicting the popularity with FL is summarized in **Algorithm 1**.

IV. Simulation Results

In this section, we first evaluate the learning performance of the proposed method for predicting file popularity via a synthetic dataset, since the ground truth of file popularity, user preference and activity level is unknown in the real dataset. Then, we evaluate the caching performance achieved by a simple proactive caching policy, which periodically caches the files with the largest predicted request probabilities until the cache space is full, via a real dataset with dynamic popularity.

We adopt the *Lastfm-1K* dataset [15], which is widely used for evaluating the performance of music recommendation algorithms. This dataset records more than 19 million requests in each second from 992 users during the year of 2005 to 2009. To capture the main trends of user behaviour in requesting files, we pre-process the dataset by choosing the most popular 500 songs and the most active 100 users that generate totally $376,407$ requests for these songs during $1,216$ days from Jan. 1, 2006 to Apr. 30, 2009. The cosine similarity of user preference (defined in [13]) is 0.17. The file popularity and

Algorithm 1 FL-based algorithm to predict the file popularity.

Input: Weighting coefficient ω, number of communication rounds T, number of iterations E, leaning rate η of SGD algorithm

Initialization:
  Initialize the model parameters (i.e., $\mathbf{W}_{\mathrm{g}}^1$ and $\mathbf{b}_{\mathrm{g}}^1$) of the NN at the BS
  Initialize the predicted popularity in the first round $\hat{\mathbf{p}}^1$ at the BS
Iteration:
for each round $t = 1, \cdots, T$ **do**
  The BS broadcasts the predicted popularity $\hat{\mathbf{p}}^t$
  ⋆ **Users:**
  for each user $k = 1, \cdots, K$ independently **do**
    Download $\mathbf{W}_{\mathrm{g}}^t$ and $\mathbf{b}_{\mathrm{g}}^t$ from the BS, and then initialize model parameters as $\mathbf{W}_k^t = \mathbf{W}_{\mathrm{g}}^t$ and $\mathbf{b}_k^t = \mathbf{b}_{\mathrm{g}}^t$
    if training sample $\{\mathbf{x}_k, \mathbf{y}_k\} = \emptyset$ **then**
      for each iteration $e = 1, \cdots, E$ **do**
        Update $\mathbf{W}_k^t$ and $\mathbf{b}_k^t$ with SGD algorithm
      end for
    end if
    $\hat{\mathbf{s}}_k^{t+1} \leftarrow f(\mathbf{r}_k^{[t-T_o+1:t]}, \mathbf{W}_k^t, \mathbf{b}_k^t)$
    Upload $\mathbf{W}_k^t$, $\mathbf{b}_k^t$, $\hat{\mathbf{s}}_k^{t+1}$ and $\|\mathbf{r}_k^{[t-T_o+1:t]}\|_1$
    Update $\{\mathbf{x}_k, \mathbf{y}_k\}$:
    if $\|\mathbf{r}_k^{[t-T_o:t-1]}\|_1 > 0$ and $\|\mathbf{r}_k^t\|_1 > 0$ **then**
      $\mathbf{x}_k \leftarrow \mathbf{r}_k^{[t-T_o:t-1]}$, $\mathbf{y}_k \leftarrow \omega \frac{\mathbf{r}_k^t}{\|\mathbf{r}_k^t\|_1} + (1-\omega)\hat{\mathbf{p}}^t$
    else $\{\mathbf{x}_k, \mathbf{y}_k\} \leftarrow \emptyset$
    end if
  end for
  ⋆ **BS:**
  Predict activity level $\hat{a}_k^{t+1}$ by (10) for $k = 1, \cdots, K$
  Predict file popularity $\hat{\mathbf{p}}^{t+1}$ by (11)
  Aggregate model parameters $\mathbf{W}_{\mathrm{g}}^{t+1}$ and $\mathbf{b}_{\mathrm{g}}^{t+1}$ by (12)
end for

activity level can be fitted as Zipf distribution with skewness parameters of 0.29 and 0.54, respectively. The number of daily requests of all the selected users is 310.

To evaluate the accuracy of popularity prediction, we synthesize a static dataset, where user preference is obtained by the algorithm proposed in [13] with given popularity, activity level and user similarity. We set the number of users, library size, skewness parameters of file popularity and activity level, user similarity and average request arrival rate the same as the pre-processed *Lastfm-1K* dataset, and generate the request data of each user over $1,216$ days according to these statistics.

To apply Algorithm 1, $\mathbf{W}_{\mathrm{g}}^1$ is initialized as Gaussian random variables with zero mean and standard deviation of 0.1, and $\mathbf{b}_{\mathrm{g}}^1$ is initialized as zeros. The popularity in the first round $\hat{\mathbf{p}}^1$ is initialized as uniform distribution such that all the files are requested with the same probability. The number of iterations for training in each round is set as 100, the number of communication rounds is set as 100, the cache update duration is set as one day, and the duration of the observation window is set as ten days. The cache size of BS is 100 files. The fine-tuned NN is with one hidden layer and one hidden node. Unless otherwise specified, this setting is used in the sequel.

We employ the synthetic dataset to evaluate the prediction error of file popularity and the resulting performance loss of caching achieved by proactive caching in each communication round, where the procedures for training the local model, predicting the preference-weighted popularity at each user, as well as predicting the file popularity at the BS can be found in Secs. III-B2, III-B3 and III-C, respectively. The prediction error in the ith round is defined as $\sum_{f=1}^{F}(\hat{p}_f^i - p_f)^2$, i.e., the total squared error for all files between the predicted popularity $\hat{\mathbf{p}}^i$ and the ground truth of popularity $\mathbf{p}$. The cache performance loss is measured by the loss of cache-hit ratio in the ith round achieved by the caching policy using $\hat{\mathbf{p}}^i$ from the caching policy using $\mathbf{p}$. The results are obtained from 100 Monte-Carlo trails. In each trail, the dataset is randomly generated with the same statistics.

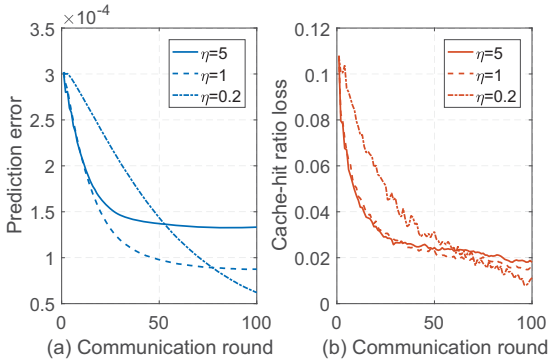

Fig. 4. Convergence of popularity prediction, synthetic dataset, $\omega = 0.1$, 100 users.

In Fig. 4, we show the popularity prediction error and the cache-hit ratio loss in each round under different learning rates. We can see that the convergence rates of the prediction error and cache-hit ratio loss are similar but not the same. With the decreasing of η, the convergence becomes slow, while the steady prediction error and cache-hit ratio loss are small.

In the sequel, we compare the proposed method of popularity prediction (with legend "*FedLrn*") with a method using centralized learning [5] (with legend "*CenLrn*"), which assumes that the BS can obtain the request records from all users in a cell. With "*CenLrn*", the future popularity is obtained by predicting the number of requests for each file with a multi-layer perception using the historical numbers of requests for the file in the observation window.

We employ the data from the 1st day to the 50th day in the real dataset to train the models of the federated and centralized learning, and use the data in the following 50 days to evaluate the caching performance. Same conclusions can be obtained when using the data in other days for training and evaluation. The metric is the cache-hit ratio in the evaluation period, i.e., the ratio of the number of requests for the cached files to the total number of requests during 50 days for evaluation. After fine-tuning under the real dataset, we set $\eta = 0.1$ for $\omega = 1$, $\eta = 1$ for $\omega = 0.1$ and $\omega = 0$.

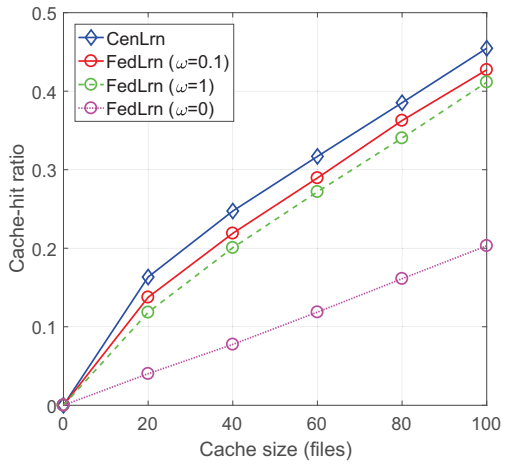

Fig. 5. Cache performance, real dataset, 100 users.

In Fig. 5, we compare the performance achieved by the proactive caching with the predicted popularity using federated learning and centralized learning. We can see that the caching policy using "*FedLrn*" with $\omega = 0.1$ performs closely to the policy using centralized learning for the real dataset where the popularity is dynamic. It can be observed that "*FedLrn*" with $\omega = 1$ performs a little worse than "*FedLrn*" with $\omega = 0.1$ despite of facing high risk of privacy leaks by using the estimated user preference as the label. This is because the request data of a user in a cache update duration is very sparse, leading to the inaccurate (implicit) estimation of the user preference. "*FedLrn*" with $\omega = 0$ performs the worst, since the corresponding caching policy is to cache the files in the library randomly as implied in Remark 1.

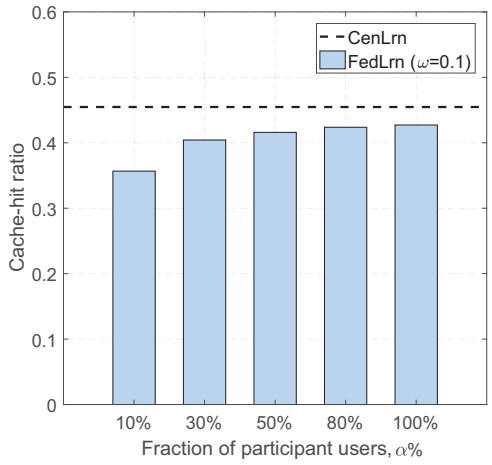

Fig. 6. Impact of the number of participant users, real dataset.

In Fig. 6, we show the impact of the number of users participating in federated learning on the caching performance, where $\alpha\%$ of users are randomly chosen to participate in learning. As expected, with the decreasing of α, the caching performance degrades. Nevertheless, we can see that even with only 30% of participant users, the cache-hit ratio using the FL-based prediction can achieve over 80% of that using the centralized learning based prediction.

V. Conclusion

In this paper, we proposed a file popularity prediction method with federated learning for proactive caching at wireless edge. To facilitate popularity prediction at the BS while hiding the user preference information, each user uploads the predicted preference-weighted popularity and the total numbers of requests for all files to the BS. A neural network was designed for each user to predict the preference-weighted popularity with the historical request records, where the information of the request for each file is only used locally at the user. With the uploaded information from all users in a cell, the BS predicts the file popularity of the cell and aggregates the local models. We evaluated the convergence of the prediction with federated learning using a synthetic dataset. Simulation results with a real dataset showed that the proposed predictor with federated learning achieves a close cache-hit ratio to the predictor with centralized learning.

References

[1] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 142–149, Apr. 2013.

[2] E. Zeydan, E. Bastug, M. Bennis, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data caching for networking: Moving from cloud to edge," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 36–42, Sep. 2016.

[3] N. Zhang, K. Zheng, and M. Tao, "Using grouped linear prediction and accelerated reinforcement learning for online content caching," in *Proc. IEEE ICC Workshops*, 2018.

[4] K. N. Doan, T. V. Nguyen, T. Q. S. Quek, and H. Shin, "Content-aware proactive caching for backhaul offloading in cellular network," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3128–3140, May. 2018.

[5] K. Qi, S. Han, and C. Yang, "Learning a hybrid proactive and reactive caching policy in wireless edge under dynamic popularity," *IEEE Access*, vol. 7, pp. 120 788–120 801, 2019.

[6] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, 2016.

[7] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. PMLR AISTATS*, 2017.

[8] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency V2V communications," in *Proc. IEEE GLOBECOM*, 2018.

[9] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *arXiv preprint arXiv:1909.11875*, 2019.

[10] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep. 2019.

[11] M. Chen, O. Semiari, W. Saad, X. Liu, and C. Yin, "Federated echo state learning for minimizing breaks in presence in wireless virtual reality networks," *arXiv preprint arXiv:1812.01202*, 2018.

[12] Z. Yu, J. Hu, G. Min, H. Lu, Z. Zhao, H. Wang, and N. Georgalas, "Federated learning based proactive content caching in edge computing," in *Proc. IEEE GLOBECOM*, 2018.

[13] D. Liu and C. Yang, "Caching at base stations with heterogeneous user demands and spatial locality," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1554–1569, Feb. 2019.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[15] Ò. Celma, *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer, 2010.