November 14, 2018

LEARNING ONE UNIVERSAL MACHINE LEARNING MODEL FOR WI-FI UNDER DIVERSE DEVICES AND ENVIRONMENTS

Dan Tan

Rob Liston

Xiaoqing Zhu

Herb Wildfeuer

LEARNING ONE UNIVERSAL MACHINE LEARNING MODEL FOR WI-FI UNDER DIVERSE DEVICES AND ENVIRONMENTS

AUTHORS:
Dan Tan
Rob Liston
Xiaoqing Zhu
Herb Wildfeuer

ABSTRACT

Techniques are provided for associating similar devices and environments together so they can be effectively learned. Furthermore, a new device (e.g., smartphone) can be associated quickly with behaviors of other similar observed smartphones to avoid learning from scratch. Since wireless performance depends strongly on device and environments types, any machine learning method also needs to be conditioned on device and environment types.

DETAILED DESCRIPTION

Applying machine learning to optimize transmission parameters at a Wi-Fi® Access Point (AP) is a promising approach to improving Wi-Fi performance. However, the same transmission parameter may lead to different outcomes for different client types (e.g., between different types of smartphones) and different environments (e.g., conference room versus airport). There are two problems with directly learning separate models at each AP. First, it is ineffective in that contributions from client and environments are tangled and is not transferrable to a new client or a new location for the AP. Second, it is not scalable to train and deploy unique models for every client and environment combination. Instead, methods are needed to characterize client effects and environment effects to allow training one model that can be universally deployed.

As such, it is desirable to train a universally deployable machine learning model which accounts for the different influences of devices types (e.g., between different types of smartphones) and environments (e.g., conference room versus airport). Techniques described herein involve capturing the influence of device and environment on wireless network performance via a unified model, which can be trained using the aggregate of data gathered across different sites. These techniques further involve representing device and

Published by Technical Disclosure Commons, 2018

environment inputs as embedding vectors, so that proximity in the embedding space generally corresponds to the similarity in device/environment characteristics. This way, the model can easily incorporate new device types via continuous updates.

Wireless network telemetry data may be collected from multiple deployment sites and fed to a central location for learning a universal deployment model. Each data entry corresponds to a single packet transmission, and includes the following: observations, actions, performance, client device Identifier (ID) and type, and environment ID. The observations may be of Channel State Information (CSI), Signal-to-Noise Ratio (SNR), etc. The chosen actions may be in the form of transmission parameters such as the number of spatial streams, multi-user versus single-user (MU/SU) mode, etc. The measured performance may include the resulting effective throughput, PHY rate, packet error rate, etc. Client device ID and type may include, e.g., the Media Access Control (MAC) address of the client, prior knowledge pertaining to the device type (e.g., manufacturer name, Operating System (OS) version, etc.), etc. The environment ID may be an identifier of the deployment environment (e.g., MAC address of the AP).

One goal may be to train a universally deployable Machine Learning (ML) model for optimizing wireless network performance, e.g., selecting the best transmission parameters to maximize throughput that accounts for the device and environment variabilities. Generally, the ML model can be either a classifier or reinforcement learning policy.

It is not scalable to train a ML model with individual device and environment identifiers as direct input, since only a subset of devices is observed at each location. Whenever a new device shows up at a location, the ML model will not be able to make an intelligent decision for this new device-environment combination before it spends some time collecting sufficient data for this new scenario. In contrast, the system described herein can still map the new device to the embedding vector, and its existing knowledge about similar devices at the same location. This way, the system can still perform inference on a new device using an existing model, while quickly updating the model for the new device via continuous training.

Figure 1 below illustrates the overall system diagram. Basically, training and inference of the universal model take as input the separately trained device and environment embedding vectors.

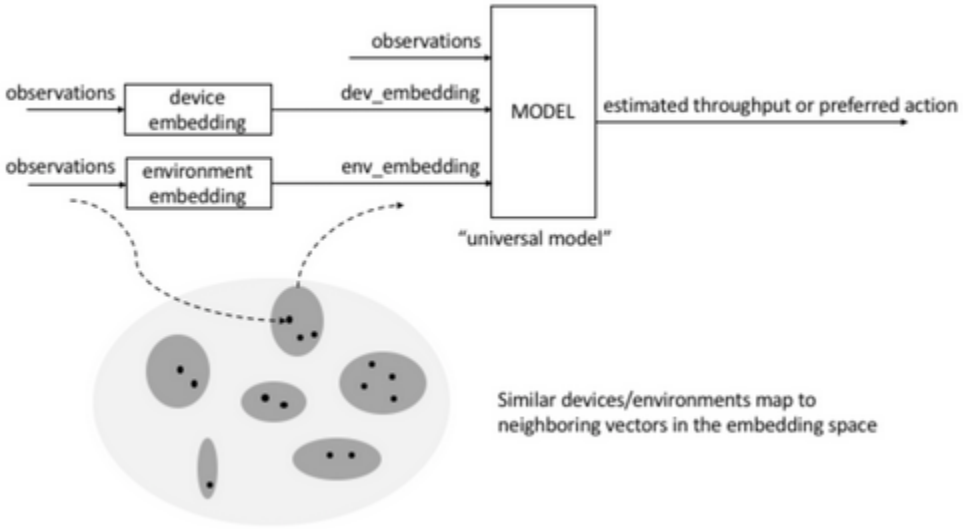

Figure 1

Figure 2 below illustrates how the device and environment embeddings are trained separately.

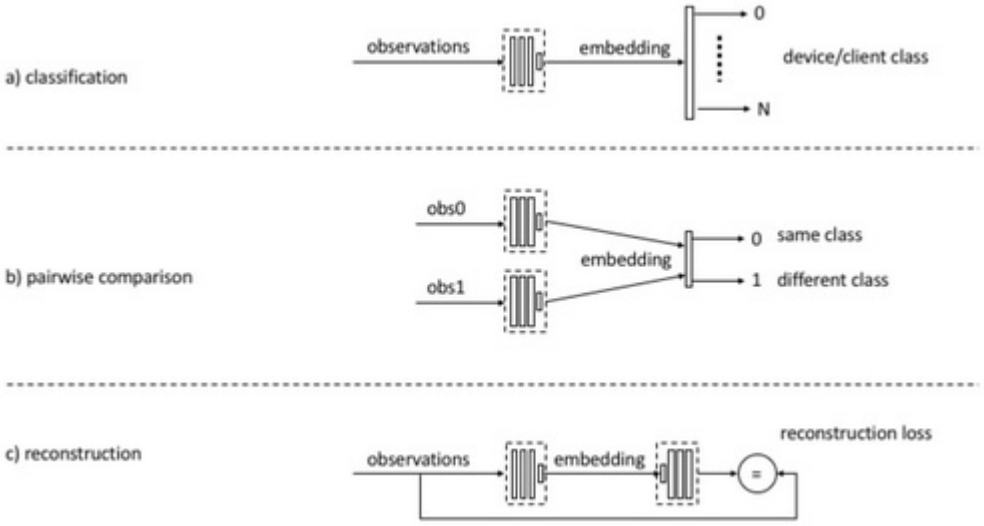

Figure 2

Using device embedding as an example, the observation vectors are chosen so that they reflect device characteristics (e.g., features used in Radio Frequency (RF) fingerprinting for devices). One or more candidate embedding learning techniques may be

employed. In a first example, one can train a device type classifier from the observation vectors. The final feature layer of the classifier can be used as the device embedding.

In a second example, by leveraging prior knowledge on whether two devices belong to the same class/type, a pair of neural networks sharing identical architecture and parameters can be trained using the loss function derived from device similarity. The embedding vector corresponds to the final layer of the neural network.

In a third example, an auto-encoder can be trained using observations that are known to reflect device characteristics. The resulting encoded observation may also serve as a device embedding. In this method explicit knowledge of device type may not be leveraged, but observation vectors that carry device information implicitly may still be relied on.

As data accumulate over time, both the embedding networks and the universal model may be periodically updated in training and redeployed for inference.

In practice, instead of repeatedly calculating the embedding from per-packet observations, one can further save computation by computing the embedding for the encountered device once and then caching the mapping from the device ID to embedding for future observations of the same device.

Although the descriptions above focus on the device embeddings as an example, the same operations also apply to environment embeddings as described herein.

The method described herein may be applied to various use cases in optimizing wireless networks. One specific example is optimizing the transmission parameters (e.g., SU/MU mode selection, number of spatial streams, PHY rate selection, etc.) at the AP for different clients based on their reported CSI over both uplink and downstream directions, SNR, etc. Another specific example is adjusting the receive power threshold for determining the handoff from one AP to another neighbor given the client's location and channel observations, so that each client can minimize its handoff time or the probability of dropping an ongoing call at a given location.

For both problems, the optimal decision needs to adapt to both the client device type and deployment environment. Provided is a method to learn a unified model by decomposing the device/environment factors as separately learned embeddings, without having to a separate the model per device-environment combination.

For the first use case on wireless transmission parameter optimization, using MU/SU Multiple-Input Multiple-Output (MIMO) mode selection as a concrete example, one embodiment is based on training a Deep Q Network (DQN) as the unified model (i.e., the policy engine). State (s) includes bi-directional CSIs, SNRs, device embeddings, and environment embeddings. Action (a) includes a binary choice whether to include the client in MU-MIMO groups. Reward (r) includes effective throughput, measured as R_PHY (1-Packet Error Rate (PER)). Here, R_PHY indicates the PHY link rate corresponding to the chosen index, and PER denotes the resulting measured packet error rate.

In other words, the goal is to learn a function Q(s,a) as the estimated reward (throughput) from the DQN network using measurement r as the reward. A standard implementation choice for the loss function for training the parameters in the DQN is given by L = sum_i [(Q(s_i, a_i)-r_i)]^2, where <s_i, a_i, r_i> are experience tuples collected at the AP, indexed by i.

Figures 3 and 4 below illustrates the input/output of the DQN network, as well as how training/inference of the unified model is carried out in this embodiment.

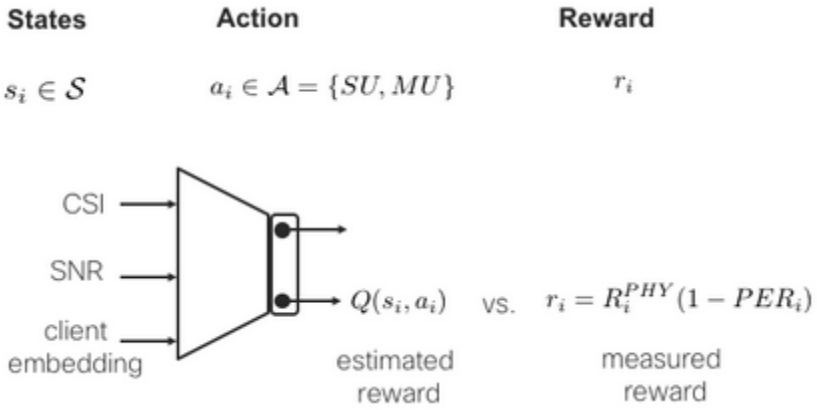

Figure 3

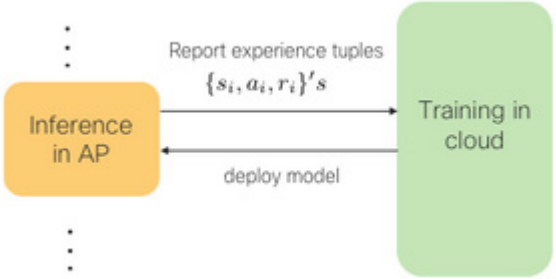

Figure 4

Training of the device and environment embedding neural networks is carried out separately (see Figure 2 above). These two neural networks only need to run inference during training of the unified model. After centralized training, the unified model for the policy engine can be deployed back to individual sites for running inference locally. The key advantage of the scheme is that data may be collected from two sites, a library and a cafeteria. The use of embedding allows the data (<s_i, a_i, r_i>) collected from the two different sites to be interpreted together for central training. An alternative is to train a separate model for each site, but one is then left with the daunting task of merging knowledge from multiple models.

In another embodiment, instead of employing neural networks for device/environment embeddings, one can directly calculate hand-crafted RF signatures of the device and environments based on intuitions. For instance, the device signature can contain power ramp-up profiles, frequency offset estimation as an indicator of the device's oscillator stability, etc.

The environment signature can contain known physical information pertaining to the AP's deployment, such as actual location, room size, ceiling height, expected client density, etc.

Techniques described herein provide the ability to train a unified model for data gathered across multiple sites, which automatically adapts its decision to different device types and deployment environments. This avoids the need for the unscalable approach of training separate models for different device-environment combinations. This also supports continuous training of the model and incremental learning of the embeddings, as new devices and new deployment environment are added to the system.

In summary, techniques are provided for associating similar devices and environments together so they can be effectively learned. Furthermore, a new device (e.g., smartphone) can be associated quickly with behaviors of other similar observed smartphones to avoid learning from scratch. Since wireless performance depends strongly on device and environments types, any machine learning method also needs to be conditioned on device and environment types.