Using Google Cloud Machine Learning to Predict Clicks at Scale

CSE 5449
Autumn ‘21

Presented by Nawras Alnaasan
alnaasan.1@osu.edu
Outline

• **Click Prediction Problem**
  • Criteo Click Logs Dataset
  • Google Cloud Machine Learning
  • GCML to Predict Clicks
    • Linear Classifier
    • DNN Model
  • Summary/Critique
Click Prediction Problem

CTR = \frac{\text{Number click - throughs}}{\text{Number of impressions}}

- CTR (click-through rate) is critical for online advertisement to maximize revenue.
- Rely on Machine Learning models to decide which ad to display.

Source: https://outsideroi.co/do-click-through-rates-matter/
Criteo Click Logs Dataset

• 1TB of click logs data over 24 days (4 billion data instances)

• Data instances correspond to a display of ad and whether it was clicked or not

• Label, 13 integer features, and 26 categorical features (hashed for privacy)

  <label> <integer feature 1> ... <integer feature 13> <categorical feature 1> ... <categorical feature 26>

• Semantic of data is unknown

• Rows are chronologically ordered
Outline

• Click Prediction Problem
  • Criteo Click Logs Dataset

• Google Cloud Machine Learning
  • GCML to Predict Clicks
    • Linear Classifier
    • DNN Model
  • Summary/Critique
Google Cloud Machine Learning (1)

- Machine learning products, services, and infrastructure on cloud.
- A lot of automated services with minimal effort/expertise in ML.
- Targeted for industries and enterprises.
- Products:
  - Vertex AI: Build pre-trained models with 80% less code. Deploy and use API.
  - AutoML: No code required, just upload data to fine-tune your model.
  - AI Infrastructure: Access to TPUs, CPUs, and GPUs.
  - Human labeling services.

Google Cloud Machine Learning (2)

AutoML Vision use case:

Source: https://cloud.google.com/automl
Google Cloud Machine Learning (3)

Source: https://cloud.google.com/automl
Outline

• Click Prediction Problem
  • Criteo Click Logs Dataset
• Google Cloud Machine Learning
  • GCML to Predict Clicks
    • Linear Classifier
    • DNN Model
• Summary/Critique
GCML to Predict Clicks

Dataset: Criteo Click Logs Dataset

Preprocess the data using Cloud Dataflow:

Tab-separated-values $\rightarrow$ Compressed TFRecords

```
PYTHON=\$(gcloud config list project --format "value(core.project)"
BUCKET=gs://\${PROJECT}-ml"
GCS_PATH="\${BUCKET}/\${USER}/large\_clicks"
python preprocess.py --training\_data \"\${GCS\_PATH}/day\_+\" \n   --eval\_data \"\${GCS\_PATH}/eval\_day\_\+\" \n   --output\_dir \"\${GCS\_PATH}/preproc\" \n   --project\_id \$PROJECT \n   --cloud
```

Cloud Dataflow features: dynamic resources autoscaling, anomaly detection, SQL/Firebase queries, data encryption, ...
Linear Classifier (1)

- Use a linear function to predict the value (click or no click)
- Linear classification scales well with large datasets.
- They use Stochastic Dual Coordinate Ascent (SDCA) for training.
- No code needed! Just specify config files and run command.

```java
01 JOB_ID="largeclicks_linear_${USER}_$(date +%Y%m%d_%H%M%S)"
02 gcloud beta ml jobs submit training "$JOB_ID" \
03  --module-name trainer.task \
04  --package-path trainer \
05  --staging-bucket "$BUCKET" \
06  --region us-central1 \
07  --config config-large.yaml \
08  --async \
09  -- \
10  --dataset large \
11  --model_type linear \
12  --ignore_crosses \
13  --l2_regularization 1000 \
14  --output_path "${GCS_PATH}/output/${JOB_ID}" \
15  --metadata_path "${GCS_PATH}/preproc/metadata.json" \
16  --eval_data_paths "${GCS_PATH}/preproc/features_eval" \
17  --train_data_paths "${GCS_PATH}/preproc/features_train"
```

Figure 1: Graph of the loss for the linear model
Linear Classifier (2)

• What are feature crosses?

Source: https://www.linkedin.com/pulse/why-feature-crosses-still-important-machine-learning-rakesh-sharma
Linear Classifier (2)

• What are feature crosses?

\[ y = \text{sign}(b + w_1 x_1 + w_2 x_2) \]

Source: https://www.linkedin.com/pulse/why-feature-crosses-still-important-machine-learning-rakesh-sharma
Linear Classifier (2)

- What are feature crosses?

\[ y = \text{sign}(b + w_1 x_1 + w_2 x_2) \]

Define \( x_3 = x_1 x_2 \)

\[ y = \text{sign}(b + w_1 x_1 + w_2 x_2 + w_3 x_3) \]

Source: https://www.linkedin.com/pulse/why-feature-crosses-still-important-machine-learning-rakesh-sharma
Linear Classifier (3)

- Adding crosses decreases loss by 1.5%
- Training time increases from 70 minutes to 142 minutes.
- In this example, crosses are added empirically 40 features $\rightarrow$ 89 features.

```python
01   column = tf.contrib.layers.crossed_column(
02       [columns[index - 1] for index in cross],
03       hash_bucket_size=int(1e6),
04       combiner='sum')
```

[Link](https://cloud.google.com/blog/products/gcp/using-google-cloud-machine-learning-to-predict-clicks-at-scale)
DNN Model

- DNNs can handle non-linearity (no need for feature crosses)
- In this example, crosses are added empirically 40 features → 89 features.
- 11 fully connected layers with 1,062 neurons each.

```bash
01 gcloud beta ml jobs submit training "$JOB_ID" \
02   --module-name trainer.task \ 
03   --package-path trainer \ 
04   --staging-bucket "$BUCKET" \ 
05   --region us-central1 \ 
06   --config config-large.yaml \ 
07   --async \ 
08   -- \ 
09   --dataset large \ 
10   --model_type deep \ 
11   --hidden_units 1062 1062 1062 1062 1062 1062 1062 1062 1062 \ 
12   --batch_size 512 \ 
13   --num_epochs 1 \ 
14   --output_path "/gs://{GCS_PATH}/output/$JOB_ID" \ 
15   --metadata_path "/gs://{GCS_PATH}/preproc/metadata.json" \ 
16   --eval_data_paths "/gs://{GCS_PATH}/preproc/features_eval" \ 
17   --train_data_paths "gs://{GCS_PATH}/preproc/features_train"
```

https://cloud.google.com/blog/products/gcp/using-google-cloud-machine-learning-to-predict-clicks-at-scale
Comparing Results

<table>
<thead>
<tr>
<th>Modeling technique</th>
<th>Training time</th>
<th>Loss</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear model</td>
<td>70 minutes</td>
<td>0.1293</td>
<td>0.7721</td>
</tr>
<tr>
<td>Linear model with crosses</td>
<td>142 minutes</td>
<td>0.1272</td>
<td>0.7841</td>
</tr>
<tr>
<td>Deep neural network, 1 epoch</td>
<td>26 hours</td>
<td>0.1257</td>
<td>0.7963</td>
</tr>
<tr>
<td>Deep neural network, 3 epochs</td>
<td>78 hours</td>
<td>0.1250</td>
<td>0.8002</td>
</tr>
</tbody>
</table>

Graph of loss for the different models vs training time in hours.

https://cloud.google.com/blog/products/gcp/using-google-cloud-machine-learning-to-predict-clicks-at-scale
Outline

• Click Prediction Problem
  • Criteo Click Logs Dataset
• Google Cloud Machine Learning
• GCML to Predict Clicks
  • Linear Classifier
  • DNN Model

Summary/Critique
Summary/Critique

• Click prediction is a critical challenge that can be solve with ML techniques.

• Google Cloud Machine Learning provides products, services, and infrastructure to allow users to easily integrate machine learning in their projects.

• In the Google Cloud article, they use data from the first 23 days for training and data from day 24 for evaluation. Any potential issues?

• Use of 11 fully connected layers each 1,062 neurons. Better DNN architectures to tackle such problem?