Embedding and Scaling AI Models in Healthcare Applications

BrainX 11/3/2018

Ghaith Habboub MD
PGY 6 Neurosurgery
Spine Fellow
Cleveland Clinic
ABOUT ME:

https://www.dentonacooley.org/blog/2016/8/11/photo-gallery
Embedding and Scaling AI Models in Healthcare Applications

BrainX 11/3/2018

1. Introduction
2. The Path for Every Data Scientist
3. The Reality of Machine Learning System
4. Existed Model Serving Methods
5. Scaling Machine Learning Servers
1. Introduction
2. The Path for Every Data Scientist
3. The Reality of Machine Learning System
4. Existed Model Serving Methods
5. Scaling Machine Learning Servers
MAKE YOUR OWN NEURAL NETWORK

A gentle journey through the mathematics of neural networks, and making your own using the Python computer language.

TARIQ RASHID
Goals:
1. Understand why it is important to create an infrastructure to serve machine learning.
2. Understand what options are currently available.
3. Understand how can we scale our models to serve our enterprise and even external clients?
Embedding and Scaling AI Models in Healthcare Applications

BrainX 11/3/2018

1. Introduction
2. The Path for Every Data Scientist
3. The Reality of Machine Learning System
4. Existed Model Serving Methods
5. Scaling Machine Learning Servers
THE PATH FOR EVERY DATA SCIENTIST

Classification
- Infection vs. No-Infection
- Pathology vs. Other Pathology

Regression
- Days
- Costs
MACHINE LEARNING:

TensorFlow

sas
App 1

VS.

App 2

© 2018 Cleveland Clinic
Realtime AI Decision Making in Surgical Spine Patients

Input Layer
- EMR Data
- OR Data
- Financial Data
- Quality & Functional
- Outcome Data
- Imaging
- Imaging Reports

Database Layer
- The Spine Database

Artificial Intelligence Layer
- Machine Learning Algorithm

Management Algorithm Layer
- Decision Algorithm to Optimize Patients Prior to Final Model Recommendations.

Outcome Layer
- Probability of Surgical Success.
- Prediction of Costs and Payments.

Continuous Learning by Collecting More Data and Retrain The Models

AI: Artificial Intelligence, EMR: Electronic Medical Record, OR: Operating Room
Realtime AI Decision Making in Sepsis Patients

Input Layer

EMR Data (Chronics triggered by either time or input)

Database Layer

The Sepsis Database

Artificial Intelligence Layer

Machine Learning Algorithm

Outcome Layer

Probability of Sepsis

Management Algorithm Layer

EMR alert
Sepsis Care Bundle
Vital Scout alert

Continuous Learning by Collecting More Data and Retrain The Models

AI: Artificial Intelligence, EMR: Electronic Medical Record
What can I do after I create a machine learning model?
After deploying the model how would you incorporate the application in the workflow without interrupting the client routine?
Fact!

In 2012 0% of Google applications utilized Machine Learning! In 2017 100% of Google applications utilized Machine Learning!
Embedding and Scaling AI Models in Healthcare Applications

BrainX 11/3/2018

1. Introduction
2. The Path for Every Data Scientist
3. The Reality of Machine Learning System
4. Existed Model Serving Methods
5. Scaling Machine Learning Servers
So what is Model Serving???
Embedding and Scaling AI Models in Healthcare Applications

BrainX 11/3/2018

1. Introduction
2. The Path for Every Data Scientist
3. The Reality of Machine Learning System
4. Existed Model Serving Methods
5. Scaling Machine Learning Servers
Training and Serving ML models with tf.keras
GENERAL CONCEPT: ONE SERVER

Electronic Medical Record Server (EMR)

Machine Learning Model

Probability Of surgical Success 0.82
GENERAL CONCEPT: **APPLICATION PROGRAMMING INTERFACE**

Electronic Medical Record Server (EMR) → Machine Learning Model Server

Data Communication Interface

*Probability of surgical success 0.82*
# Model Serving Methods

## Calling Function Directly

<table>
<thead>
<tr>
<th>Predictive Model Markup Language (PMML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Server Application Programming Interface (API)</td>
</tr>
<tr>
<td>Containerized Application Programming Interface (API) with Container-Orchestration System</td>
</tr>
</tbody>
</table>

## Cloud Services
CALLING FUNCTION DIRECTLY

\[ \hat{y} = \beta_0 + \beta_1 X \]
Example 2-5. Random Forest Classifier using **SparkML with PMML export**

```scala
goobject WineQualityRandomForestClassifierPMML {
  def main(args: Array[String]): Unit = {
      ...
      // Load and parse the data file
      ...
      // Decision Tree operates on feature vectors
  }
```
Requests per Hour in a single threaded server:

4000 requests

* 1 second

/ 60 seconds

= 67 minutes
DISADVANTAGES OF THE ABOVE METHOD

1. Limited by the offered models (logistic regression, decision trees…)
2. Platform dependent.
3. Difficulty with scalabilities
4. Difficulty with replication
5. Difficulty with model updates
6. No support for continuous learning
Scaling horizontally and vertically

- 50 passengers want to travel from LA to Vegas
- We have only one bus with capacity of 25

**Scale Up**

Vertical scaling

- Single bus with capacity 50

Horizontal scaling

- Two bus with capacity of 25 (total 50 passengers)

1. Introduction
2. The Path for Every Data Scientist
3. The Reality of Machine Learning System
4. Existed Model Serving Methods
5. Scaling Machine Learning Servers
CONTAINERIZED APPLICATION PROGRAMMING INTERFACE (API) WITH CONTAINER-ORCHESTRATION SYSTEM

1. Capabilities to build any deep learning models including convolutional neural network (CNN) for image analysis, recurrent neural network for sequence-to-sequence analysis.
2. Scalable.
3. Continuous Learning.
4. Easy to upgrade models to newer versions.
5. Easy to license to third party without exposing our models.
6. Non-platform dependent (does not rely on specific EMR since it is hiding behind an API) which is favorable from commercial standpoint.
7. Easy to expand to the cloud.
8. Able to have multi-input and connect to the outside world as opposed to be restricted to EMR.
Distributed Training and Serving Models with Microservices

Client PC
- Docker
- Kubernetes
- Ksonnet
- KubeFlow

Jupyter Hub

Kubectl

Machine Learning Training and Serving Cluster
- Master Node
  - Docker
  - Kubernetes
  - Kubeadm

- Worker
  - Docker
  - Kubernetes
  - Kubeadm
Try it at home, Maybe!

Creating Kubernetes Cluster with Kubeadm and Using
GOOGLE CLOUD AND OTHER CLOUDS
CONTINUOUS LEARNING

Activate HAL

Training

Query

Prediction
PERFORMANCE MONITORING
What to get out of all of this?

1. There is far more to create a full machine learning system than just building a machine learning model.

2. Scalability is a key for machine learning systems due to the high computation power required to appropriately serve these models.

3. Appropriate infrastructure is an absolute necessity when deciding to embed these models into health care applications.

4. There is a reason why Google, Netflix, openAI, CapitalOne, eBay, Sling... all use containerized application to provide services including machine learning based services and we should do the same!
Scalable and accurate deep learning with electronic health records

Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M. Dai, Nissan Hajaj, Michaela Hardt, Peter J. Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, Patrik Sundberg, Hector Yee, Kun Zhang, Yi Zhang, Gerardo Flores, Gavin E. Duggan, Jamie Irvine, Quoc Le, Kurt Litsch, Alexander Mossin, Justin Tansuwan, De Wang, James Wexler, Jimbo Wilson, Dana Ludwig, Samuel L. Volchenboum, Katherine Chou, Michael Pearson, Srinivasan Madabushi, Nigam H. Shah, Atul J. Butte, Michael D. Howell, Claire Cui, Greg S. Corrado and Jeffrey Dean
The collected healthcare data are reaching the genomic scale. Previous models utilizing healthcare data have limited features included due to the complexity of the data. Typically used models are even simpler, with only a few variables included and are usually single center. Fast Healthcare Interoperability Resources (FHIR) represents clinical data in a consistent, hierarchical, and extensible container format regardless of the healthcare system.
METHODS

➤ Dataset: Two hospitals USCF, UCM. Data from UCM also contained text data

➤ Outcomes: Inpatient mortality, 30-day readmission, length of stay at least 7 days, discharge diagnoses

➤ ML: Ensemble of (1. LSTM, 2. Attention-Decision Time-Aware LSTM, 3. Boosted Time-Based Decision Stumps)

➤ Control: Standard known scales including: Early Warning System Score, Modified Hospital Score for Readmission, and Modified Liu Score for Length of Stay
Fig. 1 This boxplot displays the amount of data (on a log scale) in the EHR, along with its temporal variation across the course of an admission. We define a token as a single data element in the electronic health record, like a medication name, at a specific point in time. Each token is considered as a potential predictor by the deep learning model. The line within the boxplot represents the median, the box represents the interquartile range (IQR), and the whiskers are 1.5 times the IQR. The number of tokens increased steadily from admission to discharge. At discharge, the median number of tokens for Hospital A was 86,477 and for Hospital B was 122,961.
<table>
<thead>
<tr>
<th>Table 2. Prediction accuracy of each task made at different time points</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Hospital A</strong></td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Inpatient mortality, AUROC (^a) (95% CI)</td>
</tr>
<tr>
<td>24 h before admission</td>
</tr>
<tr>
<td>At admission</td>
</tr>
<tr>
<td>24 h after admission</td>
</tr>
<tr>
<td>Baseline (aEWS(^b)) at 24 h after admission</td>
</tr>
</tbody>
</table>

| 30-day readmission, AUROC (95% CI) | | |
|-----------------------------------|-----------------|
| At admission | 0.73 (0.71–0.74) | 0.72 (0.71–0.73) |
| At 24 h after admission | 0.74 (0.72–0.75) | 0.73 (0.72–0.74) |
| At discharge | **0.77 (0.75–0.78)** | **0.76 (0.73–0.77)** |
| Baseline (mHOSPITAL\(^c\)) at discharge | 0.70 (0.68–0.72) | 0.68 (0.67–0.69) |

| Length of stay at least 7 days, AUROC (95% CI) | | |
|-----------------------------------------------|-----------------|
| At admission | 0.81 (0.80–0.82) | 0.80 (0.80–0.81) |
| At 24 h after admission | **0.86 (0.86–0.87)** | **0.85 (0.85–0.86)** |
| Baseline (Liu\(^d\)) at 24 h after admission | 0.76 (0.75–0.77) | 0.74 (0.73–0.75) |

| Discharge diagnoses (weighted AUROC) | | |
|-------------------------------------|-----------------|
| At admission | 0.87 | 0.86 |
| At 24 h after admission | 0.89 | 0.88 |
| At discharge | **0.90** | **0.90** |

\(^a\)Area under the receiver operator curve
\(^b\)Augmented Early Warning System score
\(^c\)Modified HOSPITAL score for readmission
\(^d\)Modified Liu score for long length of stay

The bold values indicate the highest area-under-the-receiver-operator-curve for each prediction task.
Fig. 2  The area under the receiver operating characteristic curves are shown for predictions of inpatient mortality made by deep learning and baseline models at 12 h increments before and after hospital admission. For inpatient mortality, the deep learning model achieves higher discrimination at every prediction time compared to the baseline for both the University of California, San Francisco (UCSF) and University of Chicago Medicine (UCM) cohorts. Both models improve in the first 24 h, but the deep learning model achieves a similar level of accuracy approximately 24 h earlier for UCM and even 48 h earlier for UCSF. The error bars represent the bootstrapped 95% confidence interval.
Fig. 3 The patient record shows a woman with metastatic breast cancer with malignant pleural effusions and empyema. The patient timeline at the top of the figure contains circles for every time-step for which at least a single token exists for the patient, and the horizontal lines show the data type. There is a close-up view of the most recent data points immediately preceding a prediction made 24h after admission. We trained models for each data type and highlighted in red the tokens which the models attended to—the non-highlighted text was not attended to but is shown for context. The models pick up features in the medications, nursing flowsheets, and clinical notes relevant to the prediction.
Health systems collect and store electronic health records in various formats in databases.

All available data for each patient is converted to events recorded in containers based on the Fast Healthcare Interoperability Resources (FHIR) specification.

The FHIR resources are placed in temporal order, depicting all events recorded in the EHR (i.e., timeline). The deep learning model uses this full history to make each prediction.

Data from each health system were mapped to an appropriate FHIR (Fast Healthcare Interoperability Resources) resource and placed in temporal order. This conversion did not harmonize or standardize the data from each health system other than map them to the appropriate resource. The deep learning model could use all data available prior to the point when the prediction was made. Therefore, each prediction, regardless of the task, used the same data.
LIMITATIONS

➤ Retrospective study
➤ Each center analyzed separately
➤ Selecting Outcome Measures.
➤ Code not fully shared.
➤ Did not include neurosurgery patients!
QUESTIONS?