Project Flogo™

Edge Machine Learning with Flogo

Matt Ellis – Product Management and Strategy
Dan Rope – Director, Data Science
Drivers for Lightweight Integration

Project Flogo

Machine Learning

Bringing Intelligence to the Edge
What (hyper)connectivity looks like ...

In 2015, we had 4.9 billion connected things, by 2020, the number of Internet-connected things will reach or even exceed 50 billion.

GE believes that the “Industrial Internet” (their term for IoT) will add $10 to $15 trillion to global GDP in the next 20 years.

According to estimations by the McKinsey Global Institute, the IoT will have a total economic impact of up to $11 trillion by 2025.
Consumer & Industrial Use Cases

Predictive Maintenance

Connected Cars

Self Optimizing Production

Smart Meters

Automated Inventory Management

Fleet Management

Track & Trace

Remote Patient Monitoring

Consumer & Industrial Use Cases

Predictive Maintenance

Connected Cars

Self Optimizing Production

Smart Meters

Automated Inventory Management

Fleet Management

Track & Trace

Remote Patient Monitoring
The architecture of today

- Cloud-centric, Centralized
- Data transported from Devices to Cloud over network
- Processing happens in the Cloud
Embracing edge computing

- Massive amount of data needs to be processed in real-time
- Cloud-centric IoT is not reliable
- Push computing, analytics & decision making to **EDGE!!**
Drivers for Lightweight Integration

Project Flogo

Machine Learning

Bringing Intelligence to the Edge
Introducing Project Flogo™

Project Flogo
Ultralight Edge Microservices Framework
IoT: Edge and Gateway Use Cases

*Bloated, oversized IoT frameworks are a thing of the past...*

**Industrial IoT**

**Consumer IoT**

- Edge Gateway
- Device
Flogo Apps are even Lighter

Announced at TIBCO Now Berlin, in June

Flogo Edge Apps now run on the tiniest microcontrollers
- 100x lighter than before
- 1,000x lighter than NodeJS
- 10,000x lighter than Java

Natively compile and run on microcontrollers
- Contribution model for device extension
- Custom activities and triggers for sensors
So, what’s the challenge with IoT integration?

Typical cloud-only IoT Integration Scenario

The Issues:
- Unreliable connectivity
- Non-trivial hop latency
- Increased connectivity TCO
A better IoT integration blueprint

Where appropriate, move application logic and integration out to the devices

- Devices
  - Edge microservices
    - Logic could also run here
  - Control
  - Sensor Events

- Gateway
  - Edge microservices
    - Go here
  - Business microservices
    - Go here
  - Control
  - Events

- Cloud

Benefits of edge integration & event processing

- Local control = more reliable
- Less bandwidth & lower TCO
- Fewer hops, less latency
Why you can’t retrofit old technology

Limited Compute Resources

Low/Intermittent Connectivity*
How did we get so small?

Java, NodeJS are great, but too large for resource constrained environments

Why **Golang** for Project Flogo:

- Complies natively and runs natively.
- Only the required dependencies are built into the application.
- Static linking enables zero OS dependencies.
Event-driven by design

IoT App = Trigger + Actions

Action = Flow of Activities

Flow = Activities + Transitions + Error Handling

Event Processing
- Triggers emit events
- Activities collect events, process them and emit output events for further processing
Flogo Web UI
A modern development environment for modern apps

- Low friction web-native UX
  - Express app logic using rich flows,
  - not just data or request pipelines
  - Inline data transformations
  - Built-in web-based debugger
  - Build for target platform directly from UI

- Available on Docker Hub or Flogo.io
Flogo Apps: Deployment – Beyond the Edge

Design and debug flows in web UI

Package using CLI or CI/CD pipeline

Deploy to PaaS, Serverless, Edge Device or run locally

Docker and the Docker logo are trademarks or registered trademarks of Docker, Inc. in the United States and/or other countries. Docker, Inc. and other parties may also have trademark rights in other terms used herein.

This document (including, without limitation, any product roadmap or statement of direction data) illustrates the planned testing, release and availability dates for TIBCO products and services. It is for informational purposes only and its contents are subject to change without notice. © Copyright 2000-2017 TIBCO Software Inc. All rights reserved. TIBCO Proprietary Information.
Drivers for Lightweight Integration

Project Flolog

Machine Learning

Bringing Intelligence to the Edge
Data Flow: Event – Data – Insight – Action

1/ All Data Begin as Real-Time Events

2/ Analytics on Accumulated Data

3/ Insights are Perishable => Take Action !!
Why Analytics at the Edge?

• Data Volume, Generation
  – Data collection exceeds ability to transport

• Intelligent Aggregation
  – Reduces transfer & storage costs
  – Smarter, more efficient networks

• Predictions
  – Smarter Device Actions, Less Network Latency
  – Actions resilient to network connectivity issues

The Issues:
• Prediction Lag
• Massive Data Transfers
• Connectivity Requirements

ML Challenges Amplify The Issues of IoT Integration!
Model Training

Data Exploration, Analysis and Modeling
Data Storage, Aggregation, and Granularity

• **Time Granularity:** Hours, Minutes, Seconds,..
  - Statistical tests are useful, autocorrelations
  - Different device measurements may require different granularities

• **Historical Time Horizon**
  - Power plant may be 5 years, hospital patient data 2 weeks
  - The “actionable time interval” (how quickly can you respond?)

• **Data Aggregation**
  - Median, mean, time weighted averages, variability/robustness
  - Different data channels must be aligned to common granularity

Correlogram: Correlation v Lag
Analytic Approaches and Use Cases

- **Time Series Data, SPC, Process Monitoring Models**
  - Control charts SPC, MSPC, Process Monitoring, RCA
  - Watch “everything” to predict trouble (it’s hard to predict outliers...)

- **Machine Learning: Unsupervised**
  - Anomaly Detection: Identify outliers in multi-dimensions
  - Model-based identification of “good”; virtual sensors (Autoencoders, Single class SVM, Cluster-distances)
  - Identification of states, clustering

- **Machine Learning: Supervised**
  - Predicting response, failure probability, state, robustness, ...
  - Model Train and Test: Control false positives
  - Time lags for forward predictions; include static predictors
TensorFlow™

• Open Source Deep Learning Framework
• Python, C++, Java, Go Deployment APIs
• Model Training: tf.estimator
• Model Deployment tf.estimator.export
TIBCO Statistica

Comprehensive Stats and Machine Learning

- 1000’s of stats, models and methods; all data sources
- Drag and drop data prep/blending/ETL, model creation
- Open Source: R, Python, C#, Spark, H2O, CNTK Deep NN

Model + Rule Management

- Metadata Managed - inbuilt models and rules, R and Python models
- Champion / Challenger rolling bake-offs and promotion cycling
- Security & Governance: repeatable, auditable, version control

Embed Everywhere

- Model + rule transport and embedding
- Native Project Flogo Activities

© Copyright 2000-2017 TIBCO Software Inc.
Project Flogo and Edge/Serverless ML

- Embedded Tensorflow inferencing capabilities
- Contribution model to backend inferencing frameworks
- Native streaming constructs
  - Time and block aggregations
  - Data collection and emission
- All capabilities exposed inline, within Flows
- Deploy to Device or Serverless: Same Code!

This document (including, without limitation, any product roadmap or statement of direction data) illustrates the planned testing, release and availability dates for TIBCO products and services. It is for informational purposes only and its contents are subject to change without notice. © Copyright 2000-2017 TIBCO Software Inc. All rights reserved. TIBCO Proprietary Information.
Device Data Aggregation Using Flogo

- Time-based aggregation in a Flogo Activity:
  - Overlapping & Non-Overlapping Windows
  - Configurable Time Duration Window
  - Time Weighted Averages
- Data Aggregated In-Memory
Deploying a TensorFlow Model Using Flogo

1. Train a TensorFlow model in the cloud/local computer/etc.
2. Save the trained model as a .pb
3. Flogo Model Lifecycle will either
   a) pull the model from a remote location or b) load the model from a local copy
4. Leverage the native Flogo Inference activity to prep data & execute the model. The input data is mapped during Flow creation.
1. Train the model in Statistica.
2. Flogo Activity Generator Node exports C code within a Flogo Activity
3. Leverage the native Flogo Inference activity to execute the model. The input data is mapped during Flow creation.
4. Add any data preparation activities that may be required
Demo

Exploration, Modeling and Deployment

Predictions in Flogo using TensorFlow or TIBCO Statistica Models
ML Deployment With Flogo: Key Takeaways

• Predictions are deployed and executed on the device with minimal overhead!

• Accelerometer example is specific--but the concept & approach is general

• Deep Learning Models vs. Classic Predictive Modeling
TIBCO Community Spotfire Wiki: ‘how to’ support

https://community.tibco.com/wiki/tibco-spotfire-community-wiki
What about Open Source Software?
Benefits for All

**OSS Project**

- Greater level of innovation
- Quicker release cycles

**OSS Consumer**

- Project transparency
- Greater control over product direction
Project Flogo on GitHub

- GitHub projects
  - flogo-lib
  - flogo-contrib
  - flogo-services
  - flogo-cli
- BSD License
- Found a bug? Feature enhancement?

Be an OSS hero and Contribute!
Q&A
This document (including, without limitation, any product roadmap or statement of direction data) illustrates the planned testing, release, and availability dates for TIBCO products and services. It is for informational purposes only and its contents are subject to change without notice. © Copyright 2000-2017 TIBCO Software Inc. All rights reserved. TIBCO Proprietary Information.
Human Activity Detection using TIBCO Spotfire and TensorFlow

Author: Venkata Jagannath, October 17, 2017

In [1]: import IPython

Deep neural networks and TensorFlow

Introduction to neural networks

Neural networks are part of a specialized category of machine learning called deep learning. These advanced models are used in supervised and unsupervised learning tasks to find non-linear patterns between a set of variables.

TensorFlow

TensorFlow is an open source deep learning library first released by Google in Nov 2015. It has since become the most popular library used for both development and production tasks. The backend allows users to deploy tasks to multiple CPUs or GPUs. TensorFlow models can be outputted as protocol buffer files.

Introduction

Problem

Sensors such as accelerometers produce several records of data every second. As sensors become more commonplace, having these devices communicate with web servers will give rise to latency and bandwidth issues. Data privacy is also a big concern. To address these problems, there is a need to let the data remain on the device and make decisions on the device itself. This report has the following sections...
Sample

The data collected from an accelerometer contains three columns - ‘x’, ‘y’ & ‘z’. The accelerometer outputs 20 records of data every second. An initial sample of 21 seconds of data for each activity is collected.

The below Spotfire analysis shows us three overlay line charts using data from accelerometer readings. The three activities - Jogging, Walking & Standing are represented in different colours.
Evaluate

Predict on the 20% unseen dataset

```python
In [14]: pred = clf.predict_classes(input_fn=pred_fn)

prediction = pd.DataFrame(np.array(list(pred)), dtype='str')
label = pd.DataFrame(text['activity'], dtype='str')

print("Accuracy of the model on testing data is %.2f\% \" \) %
(metrics.accuracy_score(label, prediction) * 100)
```

Accuracy of the model on testing data is 99.32%

Output .pb file

Create a protobuf file and save it to the model output location specified above

```python
In [20]: feature_spec = create_feature_spec_for_parsing(feature_cols)

serving_input_fn = input_fn_utils.build_serving_input_fn(feature_spec)
servingable_model_path = clf.export_savedmodel(model_output_loc, serving_input_fn, as_text=False)

print("Tensorflow model saved at : " + model_output_loc)

Tensorflow model saved at : models/
```

In [ ]: