A Stock Prediction System using open-source software

Fred Melo
fmelo@pivotal.io
@fremelo_br

William Markito
wmarkito@pivotal.io
@william_markito
It's all about **DATA**

Data Sources → Look for patterns → Prediction
Machine Learning is the answer
Applying Machine Learning

Train with historical dataset

Apply model to the new input
Why so hard?

**Hard** to scale

**Hard** to make it real-time

**Hard** to add *new* data sources

Why?
Traditional models are reactive and static

No real-time information
ETL based
Data-source specific

Hard to change
Labor intensive
Inefficient
Stream-based, real-time closed-loop analytics are needed

Multiple Data Sources
Real-Time Processing
Store Everything

In-Memory Real-Time Data
Expert System / Machine Learning

Continuous Learning
Continuous Improvement
Continuous Adapting
How can it be addressed?

- Look at past trends (for similar input)
- Evaluate current input
- Score / Predict

Info

Analysis

Neural Network
How can it be addressed?

Analysis

Filter

[ json ]

Info

Neural Network
How can it be addressed?
How can it be addressed?
How can it be addressed?
How can it be addressed?

Info

Analysis

Filter ➔ Enrich ➔ Transform ➔

Transform ➔

Neural Network

Pivotal
How can it be addressed?

Neural Network

In-Memory Data Grid

Real-time scoring

Train
How can it be addressed?
Streaming real-time analytics architecture

Ingest ➔ Transform ➔ Sink

Distributed Computing
Spark

SpringXD

Store / Analyze
Hadoop HDFS

Other Sources and Destinations
JMS
MQTT
kafka
mongoDB

Fast Data
GEOIDE

Predict / Machine Learning
MADlib
python
R
Demo Architecture

Extensible
Open-Source
Fault-Tolerant
Horizontally Scalable
Cloud-Native
**Data Stream Pipelining**

<table>
<thead>
<tr>
<th>INGEST / SINK</th>
<th>PROCESS</th>
<th>ANALYZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little or no coding required</td>
<td>Call Spark, Reactor or RxJava</td>
<td>Import and invoke PMML jobs easily</td>
</tr>
<tr>
<td>Dozens of built-in connectors</td>
<td>Built-in configurable filtering, splitting and transformation</td>
<td>Call Python, R, Madlib and other tools</td>
</tr>
<tr>
<td>Seamless integration with Kafka, Sqoop</td>
<td>Out-of-box configurable jobs for batch processing</td>
<td>Built-in configurable counters and gauges</td>
</tr>
<tr>
<td>Create new connectors easily using Spring</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scale-Out and HA Architecture

SpringXD

Stream Deployment

XD admin

Ingest ➔ Split ➔ Filter ➔ Transform ➔ Sink

SpringXD

XD Nodes

Ingest ➔ Split ➔ Filter ➔ Transform ➔ Sink

XD Nodes

ZooKeeper

Messaging
Demo Architecture

Extensible
Open-Source
Fault-Tolerant
Horizontally Scalable
Cloud-Native
Geode client-server architecture

GemFire Client
- Local Cache
- Connection pool

GemFire Server
- Partitioned Region

GemFire Locator
- Request server information from locator.
  Locator responds with least loaded server address.

Send address and load information to locator

Send, receive cache data.
Receive server events
Partitioned Regions

GemFire Server1
- Region A
- Region B

Primary to redundant replication

GemFire Server2
- Region A
- Region B

Primary to redundant replication
Event handling

The pool propagates the event to the cache server, where the region is updated.

The server distributes the event to its peers and also places it into the subscription queue for Client 2.

The pool propagates the event to the cache server, where the region is updated.

The server distributes the event to its peers and also places it into the subscription queue for Client 2.
Demo Architecture

- Extensible
- Open-Source
- Fault-Tolerant
- Horizontally Scalable
- Cloud-Native

- Split ➔ Filter ➔ Transform ➔ Sink
- SpringXD
- Predict ➔ Sink
- Machine Learning
- Fast Data
- GEODE

- HTTP
- Dashboard
- Push
Neural Networks

Input nodes layer

Hidden nodes layer

Output nodes layer

Input $x_1$

Input $x_2$

Input $x_3$

Output $y_1$

Output $y_2$

Neuron
Neural Network

- price(x)
- medium avg (x)
- relative strength (x)

medium avg (x+1)
Neural Network

# R sample source code for NN using RSNNS project
# http://cran.r-project.org/web/packages/RSNNS/
# (…)

inputs <- techIndicators[,inputColumnns(techIndicators)]
targets <- techIndicators[,outputColumnns(techIndicators)]
patterns <- splitForTrainingAndTest(inputs, targets, ratio = 0.15)

model <- jordan(patterns$inputsTrain, patterns$targetsTrain,
+    size = c(8), learnFuncParams = c(0.1), maxit = 5000,
+    inputsTest = patterns$inputsTest, targetsTest = patterns$targetsTest,
+    linOut = FALSE)
Demo Architecture

Extensible
Open-Source
Fault-Tolerant
Horizontally Scalable
Cloud-Native
SO YOU MEAN TO TELL ME...

...IT'S THAT SIMPLE?
Demo Time

LET'S MAKE SOMETHING HAPPEN

- Start Time: 2015.01.19 08:19:00
- Expire Time: 2015.01.19 08:15:00
- Last Close: 1.16121
- Predicted Close Direction: Down
- Accuracy: 70.7%
ANY questions?
Pivotal
A NEW PLATFORM FOR A NEW ERA