TigerGraph

Using Graphs to Improve Machine Learning and Produce Explainable AI

Connected Data London 2019

Victor Lee

Your Guide Today

Victor Lee
Director of Product Management, TigerGraph

- BS in Electrical Engineering and Computer Science from UC Berkeley, MS in Electrical Engineering from Stanford University
- PhD in Computer Science from Kent State University focused on graph data mining
- 15+ years in tech industry

Disclaimer: I'm a graph "expert" who knows about machine learning, not a machine learning expert who knows about graphs.

TigerGraph

OUTLINE

1. Why Graphs for Machine Learning and AI?
2. Logistics: Online account for the workshop
3. Unsupervised Learning with Graph Algorithms
   Hands-on exercise
4. Using a Graph Database as a Neural Network
   Hands-on exercise
5. Extracting Graph Features for Supervised Learning
   Hands-on exercise

Part 1 - Why Graph?

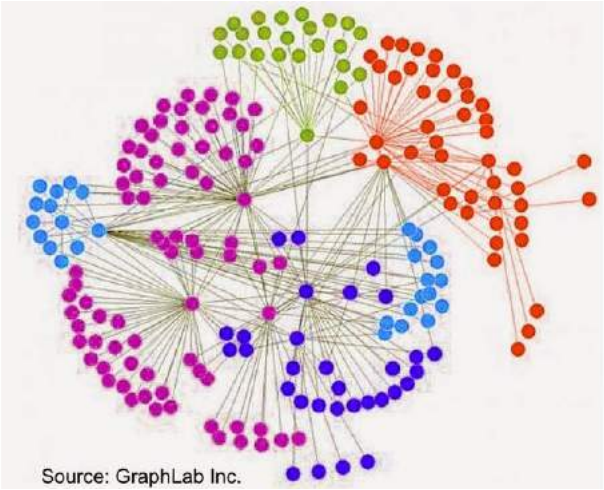
1. **Why Graphs for Machine Learning and AI?**
2. Logistics: Online account for the workshop
3. Unsupervised Learning with Graph Algorithms
   Hands-on exercise
4. Using a Graph Database as a Neural Network
   Hands-on exercise
5. Extracting Graph Features for Supervised Learning
   Hands-on exercise

Why Graphs?

- Richer Data
- Richer Analytics
  - Algorithms and Pattern Matching
  - Neural Networks, Deep Learning
  - Explainable Results

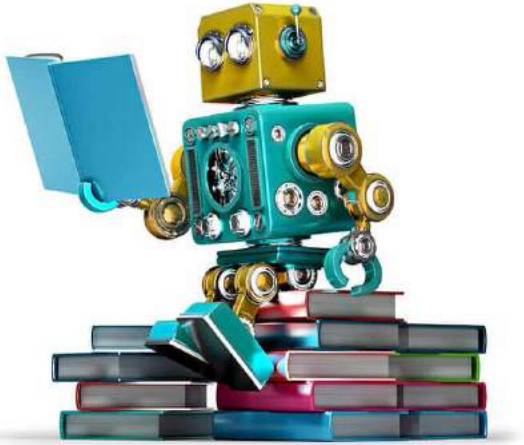
Two Hot Items: A Good Match?

Graph Database



New, exciting way to represent information and to query it.

Machine Learning

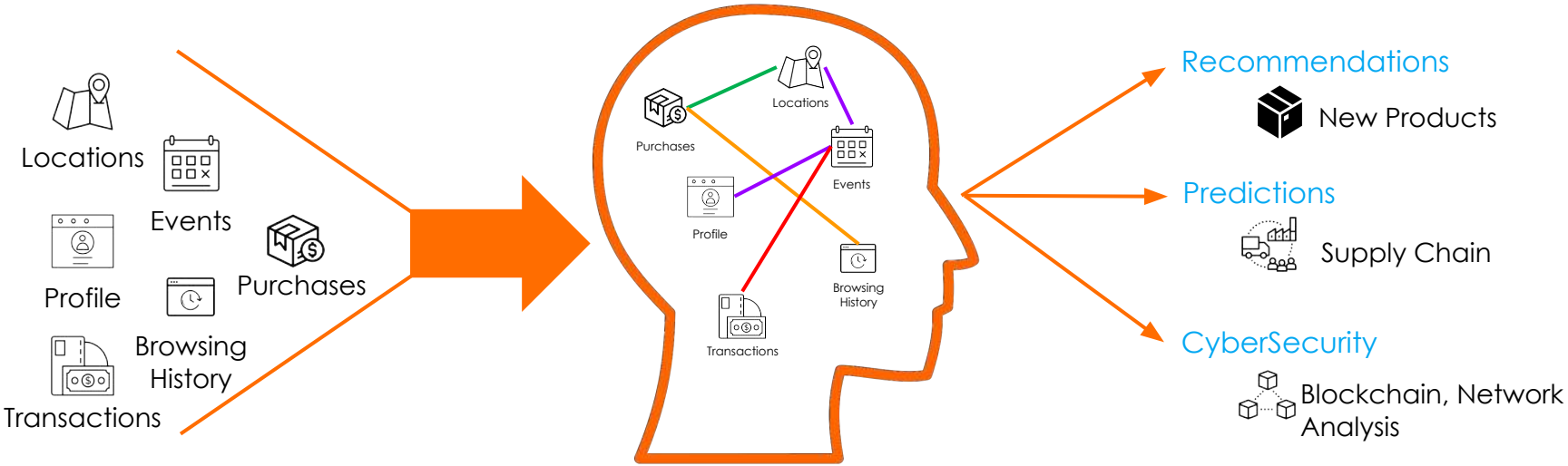


Established powerhouse for predictions and "smart" systems, but still tricky to use.

TigerGraph

Graph Is How WE THINK

Natural and organic approach for representing all kinds of things, ideas, and **how they relate to one another.**



Identify key data and process massive amounts of data

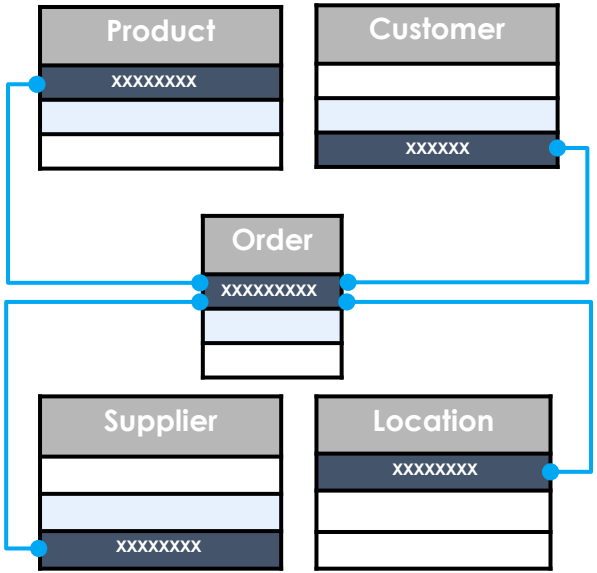
Use the power of relationships and deep analytics to provide insights

TigerGraph

Graphs Provide Richer Data

Relational Database

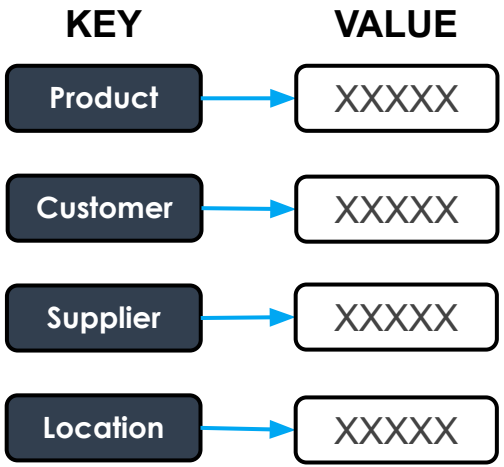
Collection of structured tables



- Each table is an attributed entity type
- High performance for transactions
- Poor performance

Tabular NoSQL Database

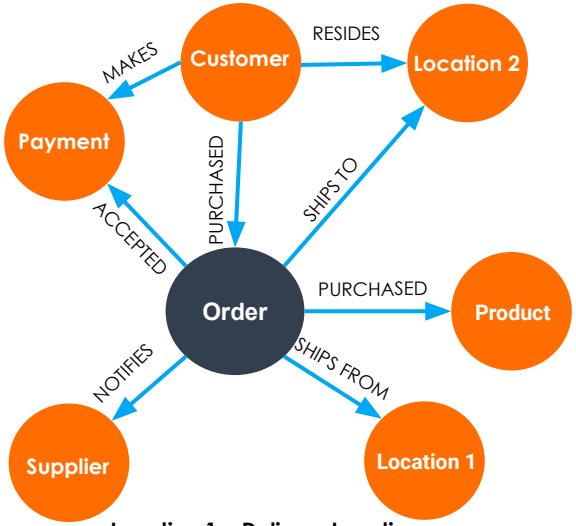
Simplifies to a single table



- Includes Key-Value and Columnar
- Less rich than Relational DB
- Advantage is scalability, aggregational analytics

Graph Database

Both nodes and connections are attributed entities



Location 1 = Delivery Location
Location 2 = Warehouse

- Flexible schema
- High performance for complex transactions
- High performance for deep analytics

TigerGraph

We Use Graphs Every Day

Social Media

Graphs are used to analyze relationships for social media

Social Graph

Website Search

PageRank is a graph algorithm used by Google Search to rank web pages in their search engine results

Knowledge Graph

Product Recommendations

Graphs are used to understand the behavior and preferences of online buyers

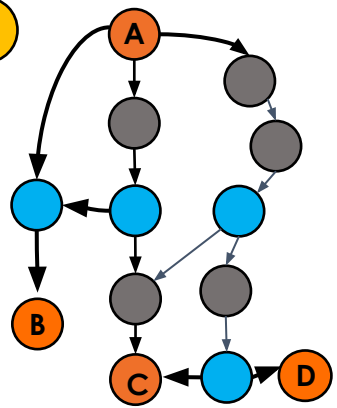
Customer 360 Graph

GRAPHS = RELATIONSHIPS

TigerGraph

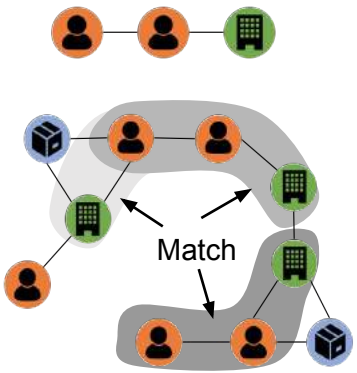
7 Key Data Science Capabilities Powered By a Native Parallel Graph

1 Deep Link Analysis



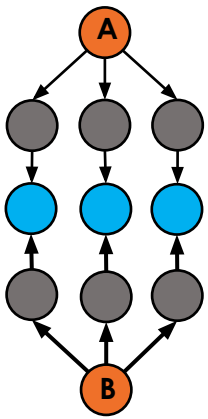
From a set of entities (e.g. devices, customers, accounts, doctors), show all links o connections

2 Multi-dimensional Entity & Pattern Matching



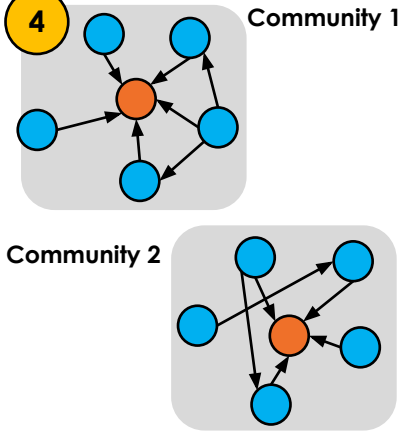
Given a pattern (e.g. referring business to a relative), find similar patterns in the graph

3 Relational Commonality Discovery and Computation



Given 2 entities (e.g. customers, businesses), follow their relationships to find commonality

4 Hub & Community Detection



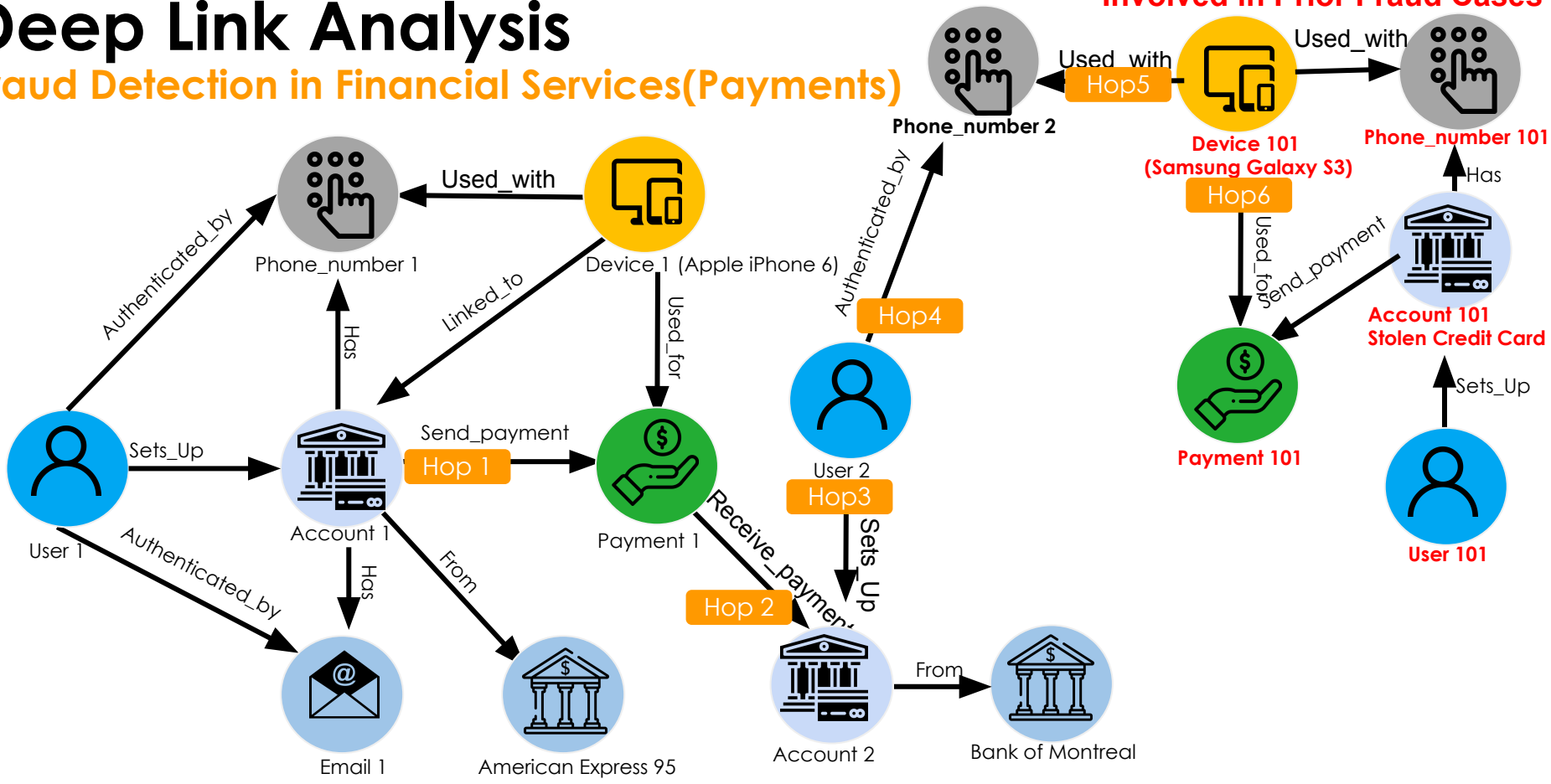
Find most influential members of a group (customers, doctors) & detect community around them

5	**Geospatial Graph Analysis**	Analyze changes in entities & relationships with location data
6	**Temporal (Time-Series) Graph Analysis**	Analyze changes in entities & relationships over time
7	**Machine Learning Feature Generation & Explainable AI**	Extract graph-based features to feed as training data for machine learning; Power Explainable AI

TigerGraph

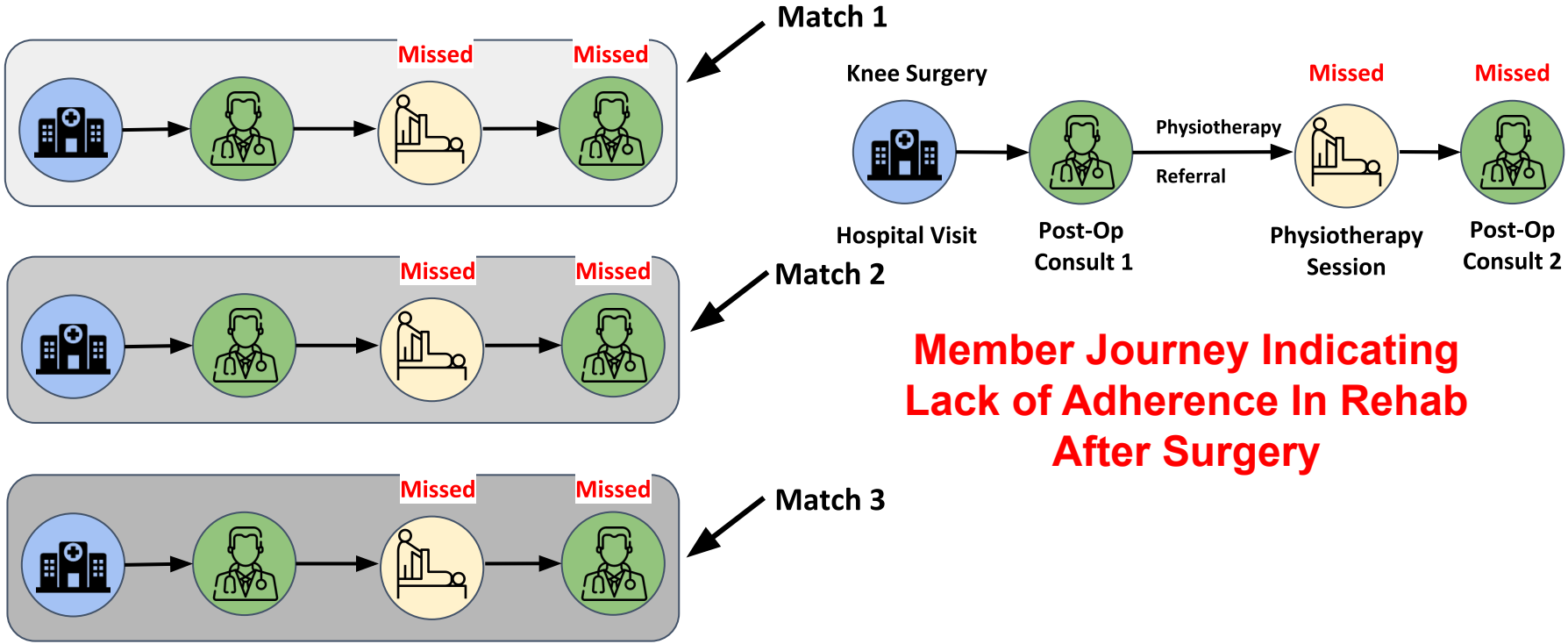
Deep Link Analysis

Fraud Detection in Financial Services(Payments)



TigerGraph

Multi-Dimensional Pattern Matching - Richer Features
Example: Finding Lack of Adherence in Member Wellness Journey

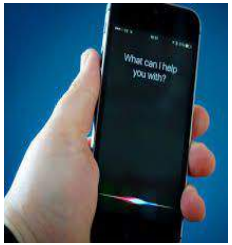


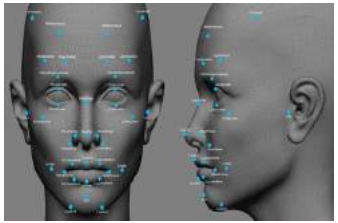
Additional details at https://info.tigergraph.com/tigergraph-2.4 & https://info.tigergraph.com/graph-gurus-14

TigerGraph

Some Uses of Machine Learning

- Virtual Personal Assistants
  - Voice-to-text
  - Semantic analysis
  - Formulate a response
- Real-time route optimization
  - Waze, Google Maps
- Image Recognition
  - Facial recognition
  - X-ray / CT scan reading
- Effective spam filters





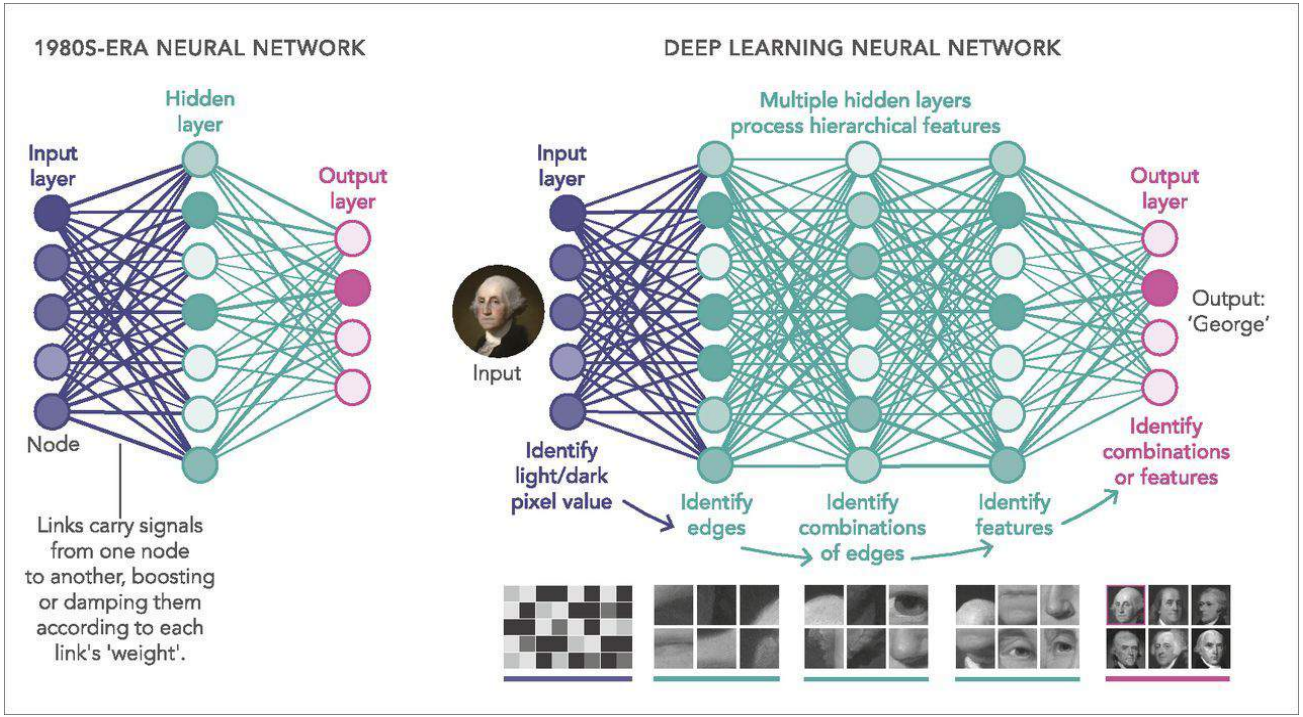
TigerGraph

Some Machine Learning Techniques

- **Supervised Learning**
  Humans provide "training data" with known correct answers
  - Decision Trees
  - Nearest Neighbor
  - Hidden Markov Model,Naïve Bayesian
  - Linear Regression
  - Support Vector Machines (SVM)
  - Neural Networks
- **Unsupervised Learning**
  No "correct" answer; detect patterns or summarize
  - Clustering, Partitioning, Outlier detection
  - Neural Networks
  - Dimensionality reduction

Graph Analytics - Neural Network & Deep Learning

- Neural Networks ARE graphs!

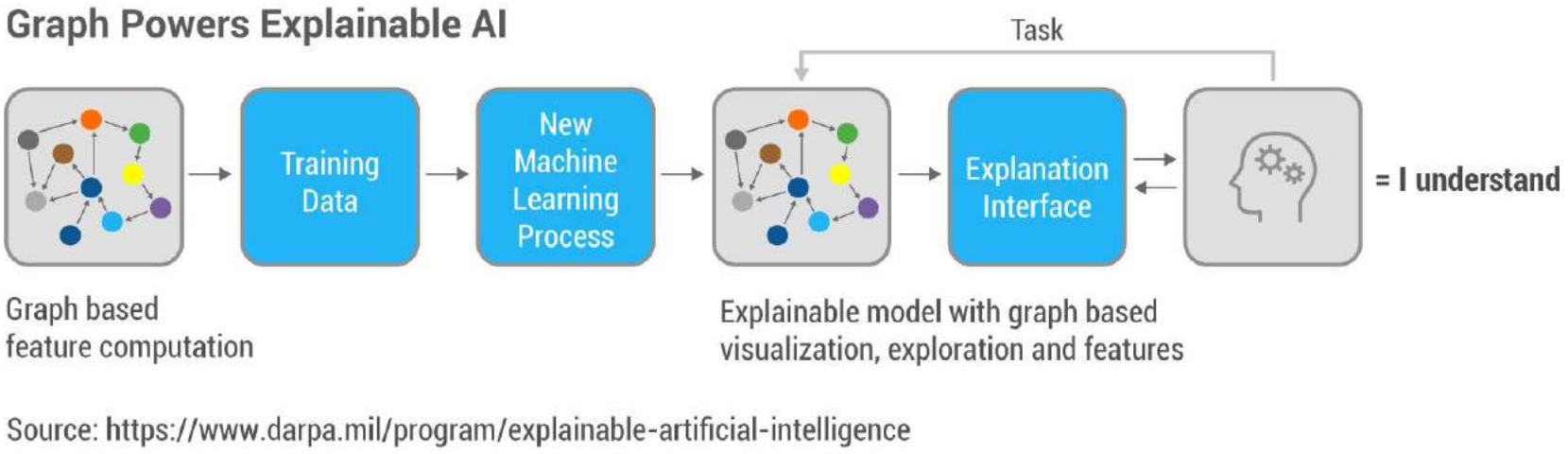


https://www.pnas.org/content/116/4/1074 **"What are the limits of deep learning?"**

TigerGraph

Graph Analytics - Explainable Results

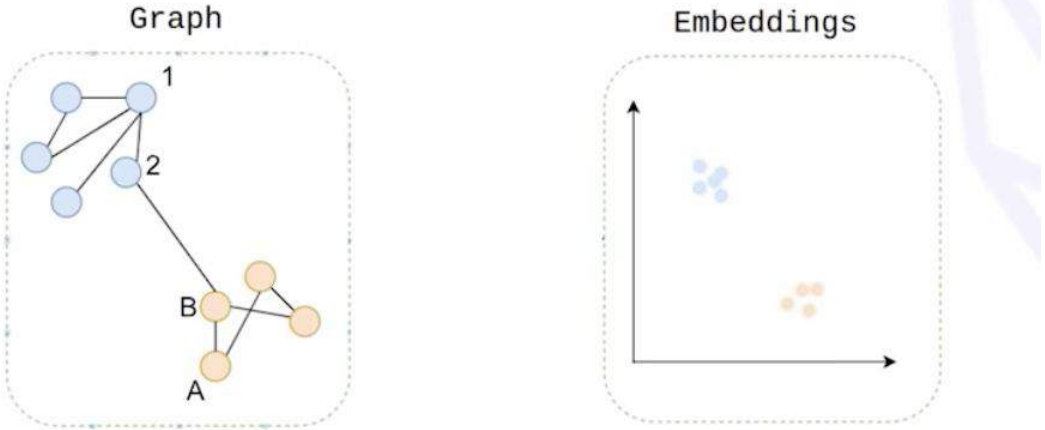
- The data model itself is explanatory
- Graph Analytics = Connecting the Dots



TigerGraph

Node2Vec

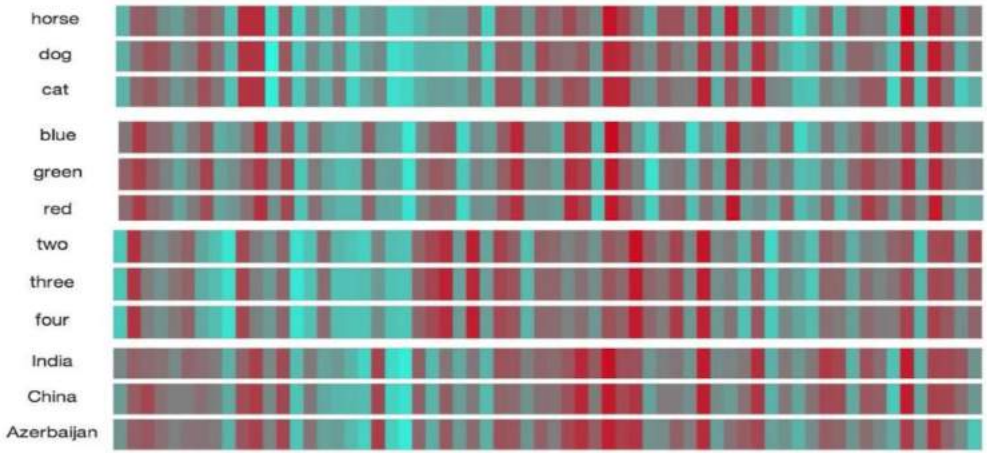
Construct a feature vector representation(embedding) for nodes and edges



TigerGraph

Node2Vec

	Searching,	Incredibles 2,	Project Gutenberg,	Crazy Rich Asians
Neol (	6,	9,	8,	5)
Kat (	0,	7,	0,	8)



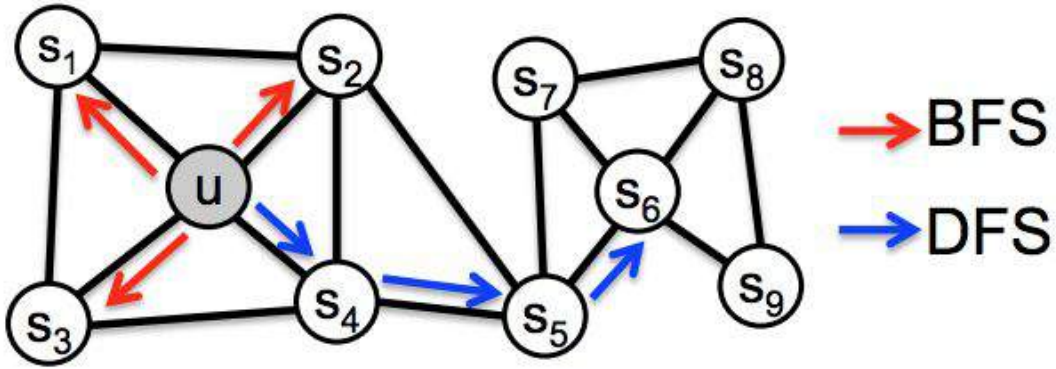
The dimensions don't really have names/features to associate with

TigerGraph

Node2Vec

1. Nodes from the same network community and have lots of interconnections

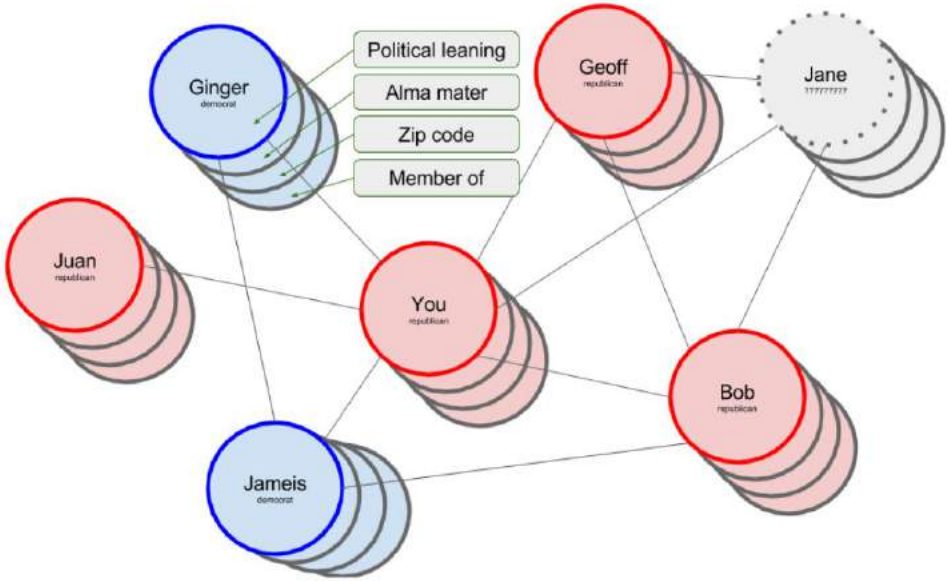
2. nodes that share similar roles in the community



TigerGraph

Graph Convolutional Network

Node Classification: giver nodes, how to provide a l node



TigerGraph

Summary for "Why Graph for ML/AI?"

- Natural Data Model - Graph is how we think
- Richer data - connections between entities, graph-based features
- Graphs have always had a natural role in machine learning:
  - Unsupervised learning through graph algorithms, frequent pattern mining
  - Learning through neural networks and deep learning
- Graph data models are uniquely qualified to provide explanatory AI.

Summary for "Why Graph for ML/AI?"

- Natural Data Model - Graph is how we think
- Richer data - connections between entities, graph-based features
- Graphs have always had a natural role in machine learning:
  - Unsupervised learning through graph algorithms, frequent pattern mining
  - Learning through neural networks and deep learning
- Graph data models are uniquely qualified to provide explanatory AI.

Part 2 - Setting up your account

1. Why Graphs for Machine Learning and AI?
2. **Logistics: Online account for the workshop**
3. Unsupervised Learning with Graph Algorithms
   Hands-on exercise
4. Using a Graph Database as a Neural Network
   Hands-on exercise
5. Extracting Graph Features for Supervised Learning
   Hands-on exercise

Using Your TigerGraph Workshop Instance

- Workshop home page: docs.tigergraph.com/workshop/cdl-19
  - These slides, Machine access, Exercise descriptions
- Need: Browser (Preferred: Chrome. Not supported: IE)
- Accessing remote cloud instance:
  - Instructor will assign you an instance number
  - Shell access: Use ssh (or PuTTY for Windows)
  - Exact URL and passwords will be given class.

Part 3 Graph Algorithms

1. Why Graphs for Machine Learning and AI?
2. Logistics: Online account for the workshop
3. **Unsupervised Learning with Graph Algorithms**
   Hands-on exercise
4. Using a Graph Database as a Neural Network
   Hands-on exercise
5. Extracting Graph Features for Supervised Learning
   Hands-on exercise

Graph Algorithms for ML/AI?

- Best-known graph algorithms
  - Shortest path
  - PageRank
  - Are they considered "learning?"

Types of Graph Algorithms

- Path Finding
- Clustering / Community Detection
  - Lenient clustering - connected component: one connection
  - Strict clustering - clique detection: every possible connection
  - **Relative density** - more connections in-group than between-group
- **Ranking**
  - PageRank, HITS
  - SimRank, RoleSim
- **Frequent Pattern Discovery**
  - Agglomerative search

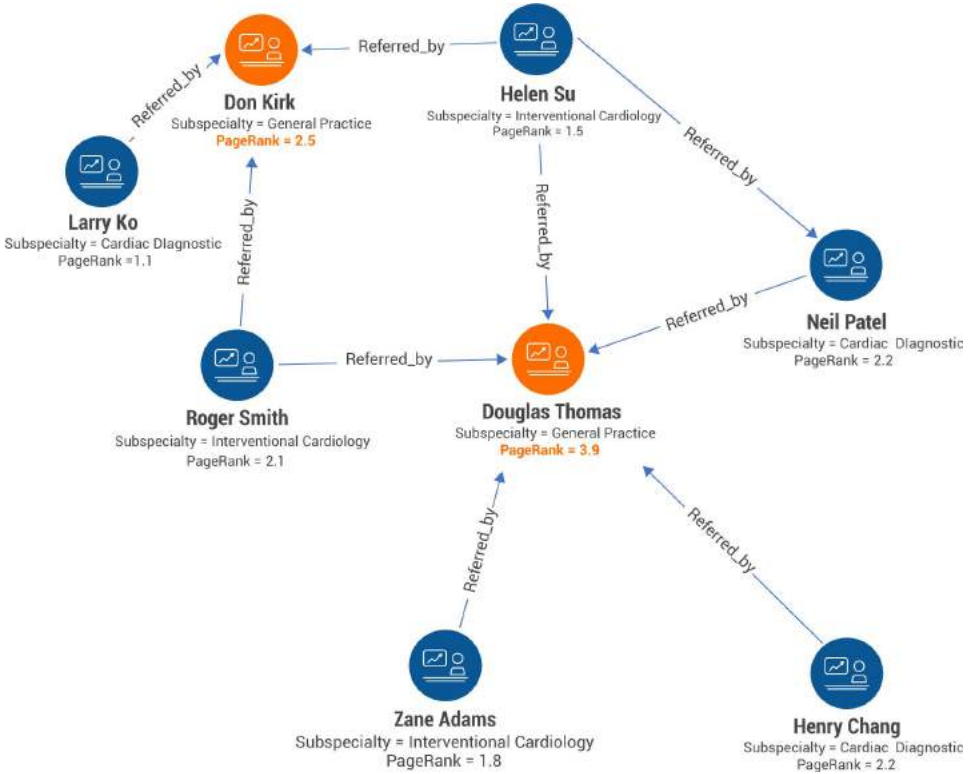
Graph Problem vs Algorithm vs Implementation

- Strictly speaking, "Finding the shortest weighted path from A to B" is a **graph problem.**

- Dijkstra discovered an **efficient algorithm** for solving the more global problem of "shortest weight paths from A to every other vertex", based on breadth first search from A.

- An **implementation** is written in a particular programming language and selects what data structures to use to hold temporary and final results. A parallel processing platform will generally be much faster.

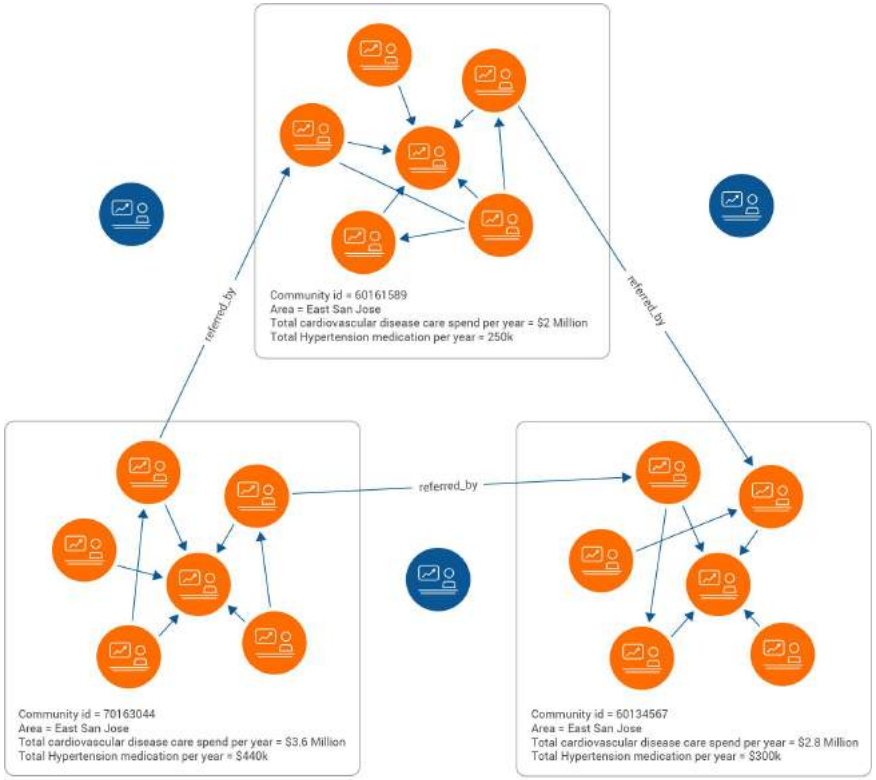
Hub Detection/Ranking - Finding the Influencers



Additional details at https://www.tigergraph.com/solutions/product-service-marketing/

TigerGraph

Community Detection with TigerGraph



Additional details at https://www.tigergraph.com/solutions/product-service-marketing/

TigerGraph

Computational Complexity

- Graph Algorithms arise from theoretical mathematics and computer science.
- Using simplified abstract models of a computer, each algorithm has a Memory Complexity (how much disk do I need?) and Computational Complexity (how many CPU cycles do I need)?
- The complexity is typically written as O(<expression>)
  - Expression in terms of #Vertices (V), #Edges (E), #iterations. etc.
  - Can have different values for worst case and for "expected" case

Practical Graph Algorithm Computation

Problem	Best or Most Common Implementation
Positive Weighted Shortest Path	O(E*V)
PageRank	O(E*k), k = num of iterations
Closeness Centrality	O(E*k), k = num of iterations
Betweenness Centrality	O(E*V) unweighted graph, $O(V^3)$ weighted, dense graph

- If your graph is HUGE, then $O(V^3)$ or $O(E^2)$ might not be reasonable.
- Use the best algorithms
- Use the best implementations and platform:
  - Massively Parallel Processing
  - Highly programmable
    - Data structures
    - Control flow
    - User-customizaable

Schema and Data for Algorithms

- Many classical graph algorithms are designed for monotype graphs:
  - One vertex type
  - One edge type

- Individual algorithms are designed for certain specific schema types:
  - Directed or undirected edges
  - With/without edge weights or vertex weights

Understanding Louvain Community Detection

- For monotype graphs with undirected edges
  - One vertex type, one undirected edge type with weights
- Uses "modularity" as the objective function
  - The fraction of the edges that fall within the given groups minus the expected fraction if edges were distributed at random.
  - The value of the modularity lies in the range [-1 : +1]
  - Researchers from the University of Louvain discovered an efficient computational approach.

Exercise: Using Louvain Community Detection to find/confirm Communities

- Exercise 1 at https://docs.tigergraph.com/workshop/cdl-19
- View the Results
  - Vertices in the same community have the same community ID.

Enhancing the Output for Louvain

- GraphStudio lets you color vertices programmatically. E.g., for all the vertices with a given community number value!

Summary for Graph Algorithms

- Graph Algorithms provide specific information about a graph's structure, either globally or with respect to certain vertices.

- Certain algorithms can be considered learning algorithms.
  - Ranking
  - Community Detection
  - Frequent Pattern Discovery

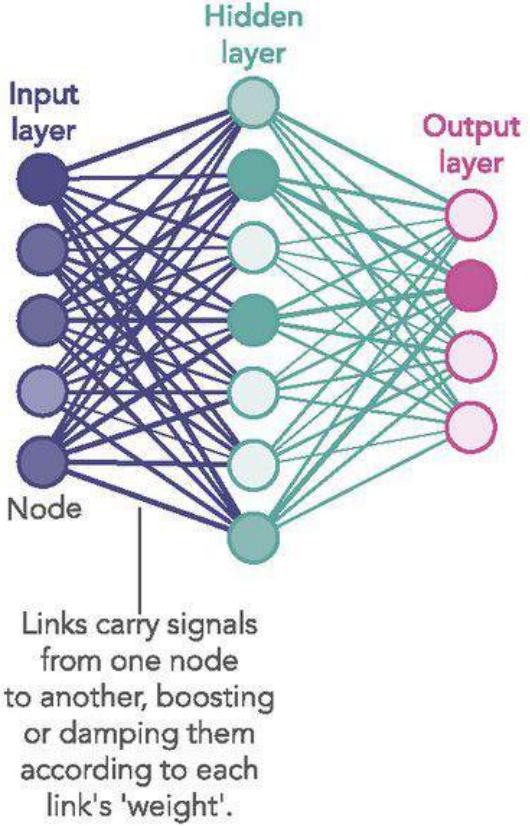
- The computational requirements can be intense:
  - Consider the theoretical memory and CPU demands (big O).
  - Consider the parallelism and programmability of the graph platform.

Part 4 Neural Networks

1. Why Graphs for Machine Learning and AI?
2. Logistics: Online account for the workshop
3. Unsupervised Learning with Graph Algorithms
   Hands-on exercise
4. **Using a Graph Database as a Neural Network**
   Hands-on exercise
5. Extracting Graph Features for Supervised Learning
   Hands-on exercise

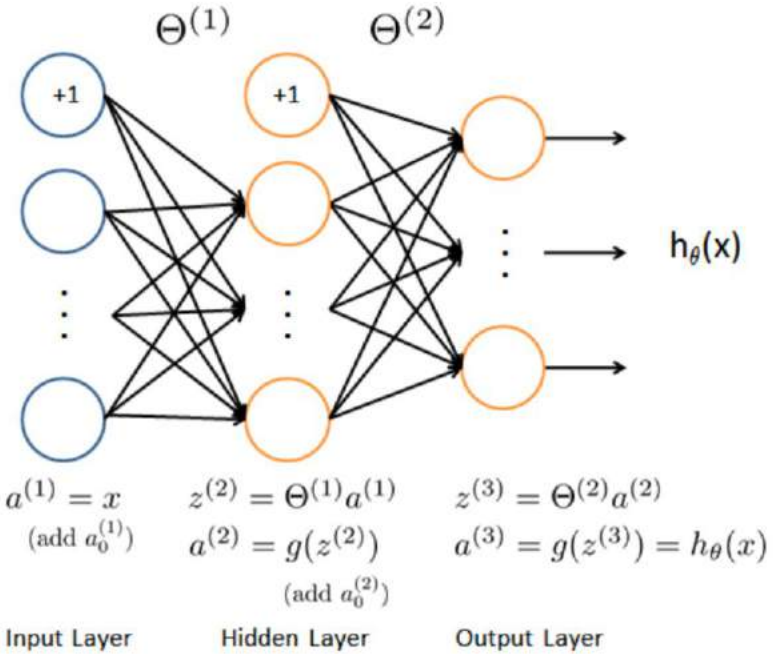
What is a Neural Network?

- Inspired by how we believe(d) biological neurons work.
- Supervised or Semi-Supervised
  - Depends on getting a response to each set of stimuli, assessing if that result was desirable or not, and then making adjustment to the network accordingly.
- Assumes the output can be computed as a linear combination of the inputs.
- ML Training: Learning the best weights on the edges



TigerGraph

Feed-Forward Neural Network Model



Left figure shows the structure of a three-layer neural network.

First layer is the input layer with the activation function $a^{(1)}$ equal to the input x.

At the hidden layer, $z^{(2)}$ is computed from $\Theta^{(2)}$ and $a^{(1)}$. $\Theta^{(2)}$ is a l_2 by l_1 weight matrix where l_2 and l_1 are the number of neurons in the first and second layer respectively. $a^{(2)}$ is then computed by a sigmoid function $g(z^{(2)}) = 1/(1+\exp(-z^{(2)}))$.

At the output layer, the prediction $h_\theta(x) = a^{(3)}$ is in the similar fashion as $a^{(2)}$ in the hidden layer.

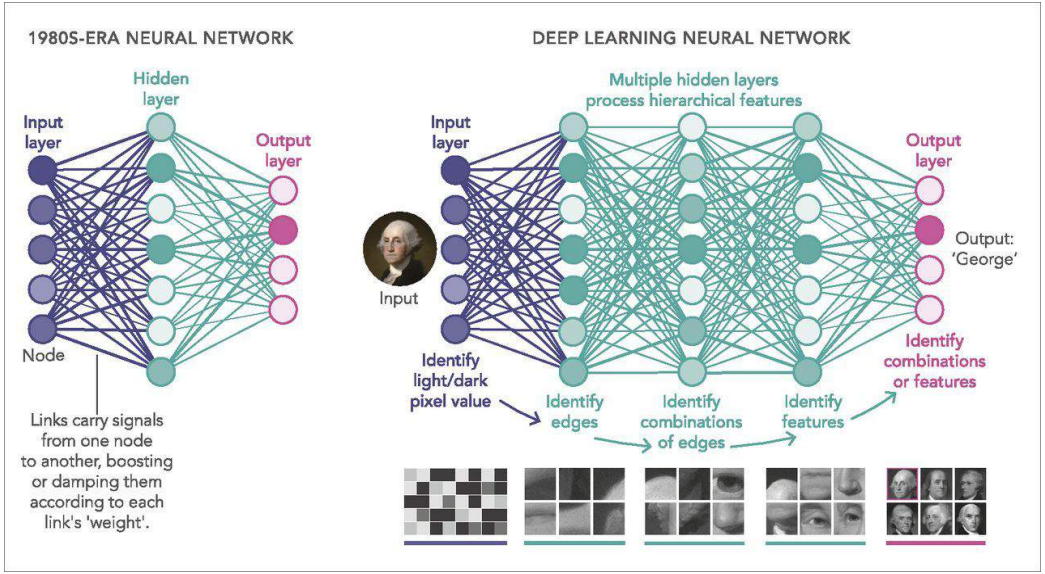
TigerGraph

Should I Use a Graph DB for Deep Learning?

- Is your data now or will be in a graph?
- Is your graph DB good at high-performance user-specified analytics?
  - Need to implement neuron computation
  - Need efficient parallel and iterative computation
- Do you need explainable results?

Deep Learning

- Deep Learning is just learning from a neural network which has multiple hidden layers.
- Is it better than single-layer neural networks?
- Clearly more expensive computationally



TigerGraph

Neural Network Example

Problem: Recognizing Handwritten Digits

- Classical machine learning multi-class classification problem
- Given a 20 x 20 grayscale image showing a handwritten digit (vector of length 400), output 0,1,2,3,4,5,6,7,8 or 9.
- We will solve it using a three-layer feed forward neural network.
- The neural network can be trained using a backpropagation algorithm implemented in GSQL.

examples of training dataset

Example taken from a Coursera.org course on Machine Learning by Andrew Ng.

Training Data Format

Table with 402 columns

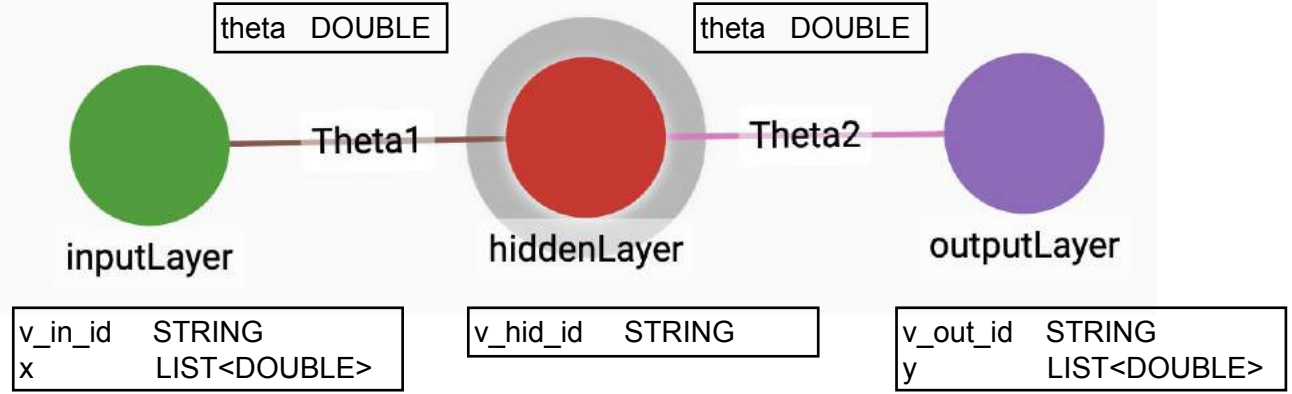
- Column 0 = ID
- Columns 1 to 400: number representing the gray scale intensive of pixel n
- Column 401 = Label (0,1,2,3,4,5,6,7,8 or 9)

n-Fold Leave-one-out Cross Validation Training

1. Split the data randomly into n equally-sized sets
2. For k = 1 to n
   a. Use all the groups EXCEPT group k as the training data, and run the learning algorithm.
   b. After the model has been used, apply the model to each member of set k, and record whether the model predicted the correct label or not.
3. Compute the aggregate average correctness percentage.

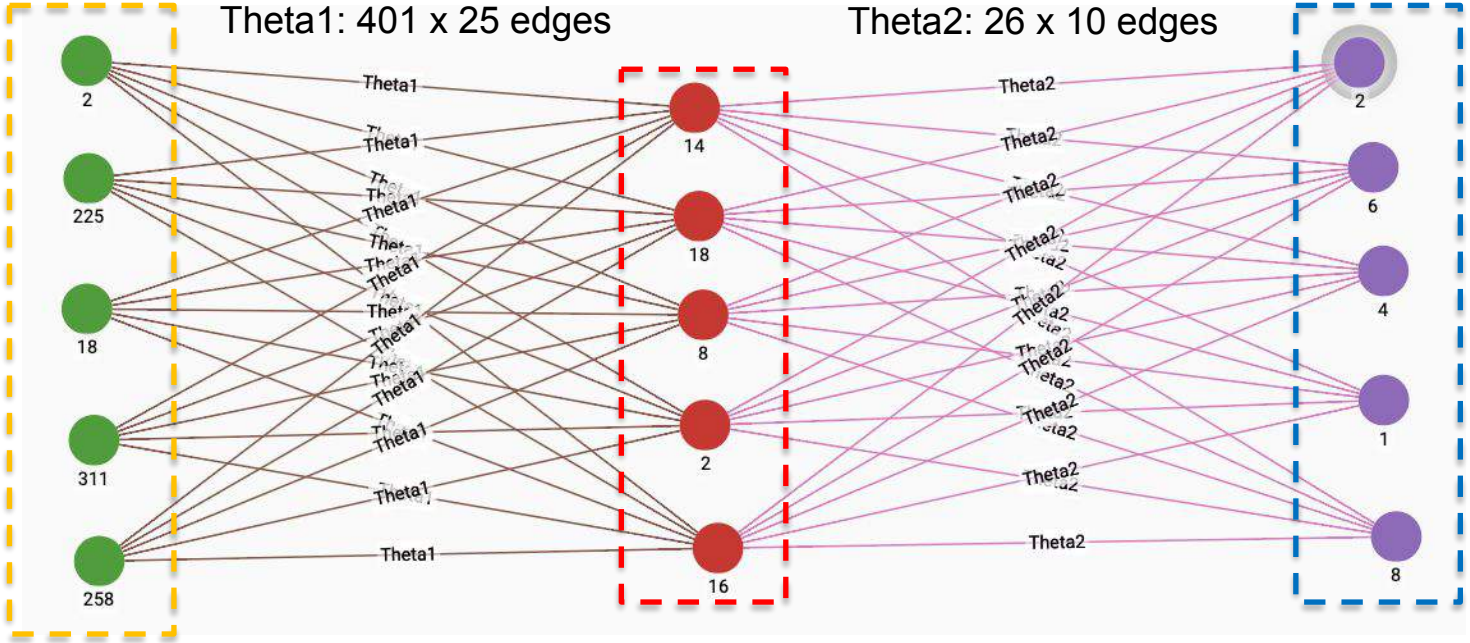
Schema and Queries

- Training data (*x, y*) are loaded into the *inputLayer* and *outputLayer* vertices.
- The weights (*theta*) in the neural network are trained by running **backpropagation.gsql**.
- After training, the neural network can make prediction by running **prediction.gsql**



TigerGraph

Neural Network



Input layer
- 401 vertices
- 20 x 20 image + 1 bias

Hidden layer
- 26 vertices
- 25 from the input layer + 1 bias

Output layer
- 10 vertices
- Representing 0 to 9

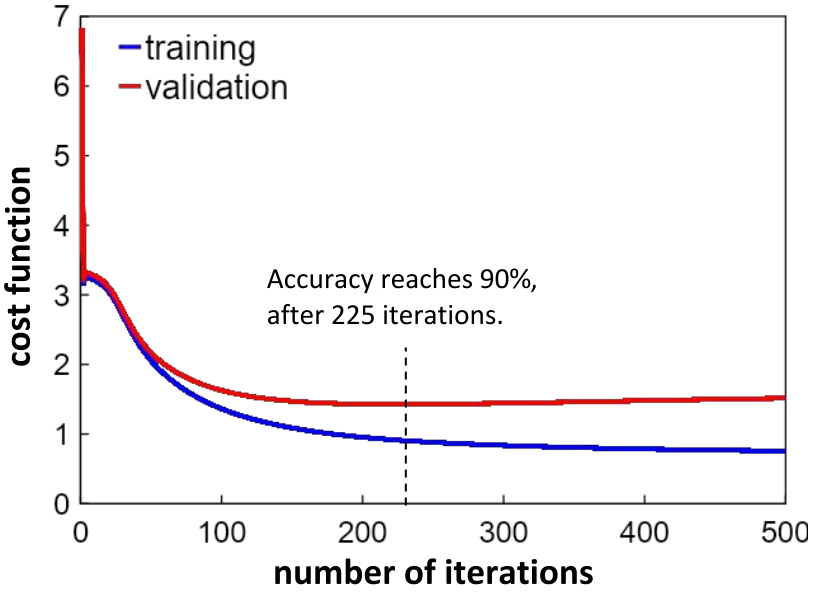
TigerGraph

Memory Cost

- Memory needed for the neural-net graph
  - Neural-net graph: $O(N_e)$, N_e = number of edges
  - Training the neural-net graph: $O(N_e+N_v*m)$, N_v = number of vertices, m = number the training data sets.
- Messages passed around per layer?
  - $O(N_{vi}*m)$, N_{vi} = number of vertices in i^{th} layer.

Time Cost

- 5000 data sets were split into training set (4000) and validation set (1000).
- The neural network were trained on the training set with a constant learning rate.
- The prediction accuracy on the validation set reaches 90% after 225 iterations, where the cost on validation is minimized.
- Each iteration takes ~0.77 s, total 225 iterations takes less than 3 min.



TigerGraph

Using a Neural Network to recognize handwritten digits

- Exercise 2 at https://docs.tigergraph.com/workshop/cdl-19

- We will use https://sketch.io/sketchpad/ top draw some digits test our model.

Part 5 Graph Feature Extraction

1. Why Graphs for Machine Learning and AI?
2. Logistics: Online account for the workshop
3. Unsupervised Learning with Graph Algorithms
   Hands-on exercise
4. Using a Graph Database as a Neural Network
   Hands-on exercise
5. **Extracting Graph Features for Supervised Learning**
   Hands-on exercise

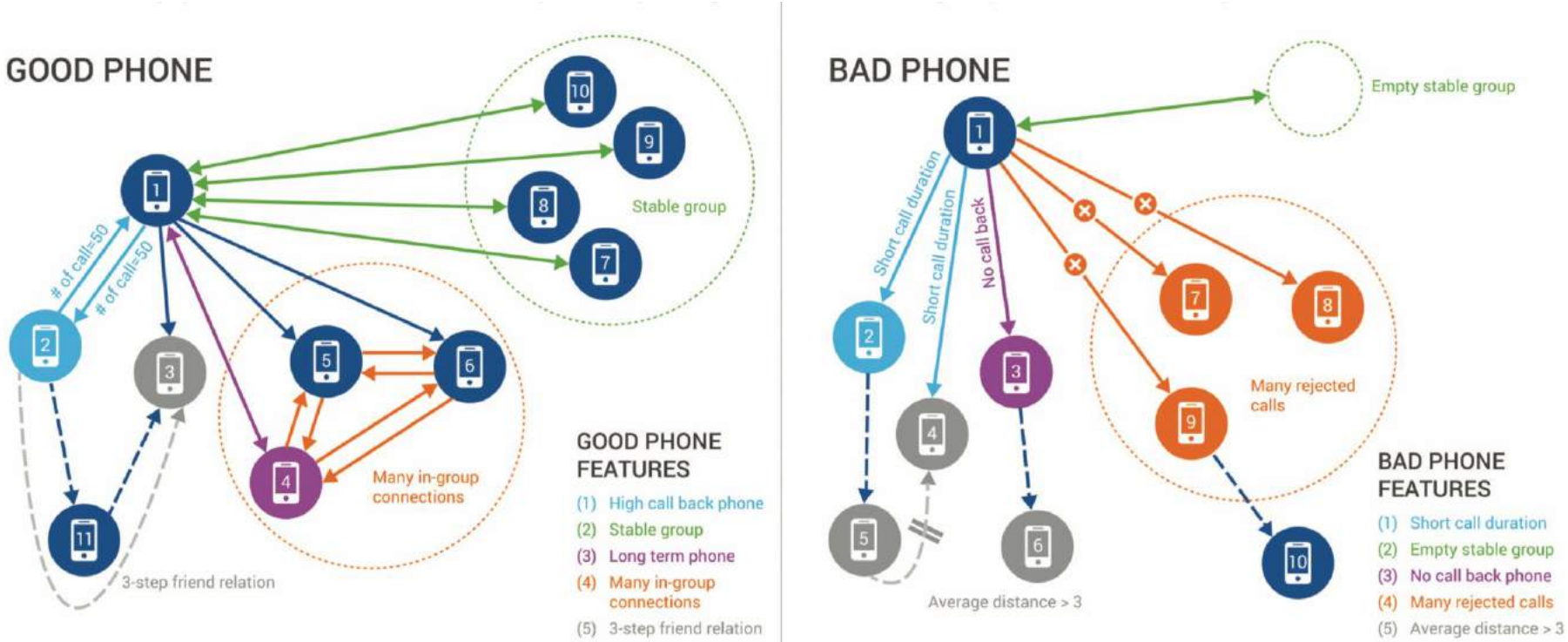
Graph Feature Extraction and Explainable AI

- Most/All ML methods try to correlate *features* (properties/attributes) with the target result.

	Feat 1	Feat 2	Feat 3	**Result**
Item 1	X	0	0	**X**
Item 2	X	X	0	**0**
Item 3	X	0	X	**?**

- Quality of modeling depends on quality of features
- Graph provides features not available in tabular data ⇒ **Richer feature set ⇒ Better AI models**

Detecting Phone-Based Fraud by Analyzing Network or Graph Relationship Features at China Mobile

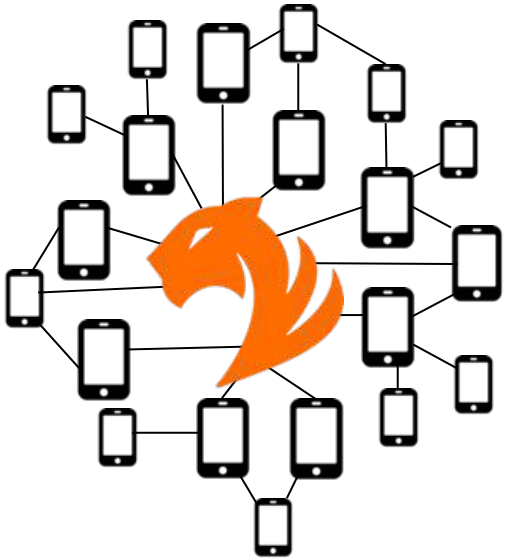


Download the solution brief at - https://info.tigergraph.com/MachineLearning &
Explore solution details at https://www.tigergraph.com/solutions/ai-and-machine-learning/

TigerGraph

Generating New Training Data for Machine Learning to Detect Phone-Based Scam

- Graph with 600 Million phones and 15 Billion calls,1000s of new calls per second.
  Feed Machine Learning with new training data with 118 features per phone every 2 hours



Tens – Hundreds of Billions of calls

Phone 1 Features

(1) High call back phone
(2) Stable group
(3) Long term phone
(4) Many in-group connections
(5) 3-step friend relation

Phone 2 Features

(1) Short term call duration
(2) Empty stable group
(3) No call back phone
(4) Many rejected calls
(5) Avg. distance > 3

Training Data

118 per phone x 600 Million phones = 70 Billion new features

Machine Learning Solution

Download the solution brief at - https://info.tigergraph.com/MachineLearning

TigerGraph

Explainability of a Graph Feature-based Model

- Case 1: Weighted sum of Graph Features
  - Example:
  - Low call back phone 30%
  - No status group 20%
  - Short calls 15%
  - Many in-group connections 15%
  - 3-step friend relation 10%
  - "These are the factors that lead to phone fraud."

- Case 2: Connecting the Dots & Link Prediction
  - Learning model says the presence of certain types of paths between A and B is indicative of the target type.

Use Case Example: Bank Fraud

- Using Banksim synthetic data generator where 2% of the transactions are fraudulent
  - data available on Kaggle
  - data originally structured as one table, but we transform it into a graph
- Exercise 3 at https://docs.tigergraph.com/workshop/cdl-19
- We will extract graph features, then export them to a data science workbook.

Summary

1. Graphs improve Machine Learning and AI
2. Machine Learning can be either in graph or out of graph
3. Graph's natural data model makes the AI models very explainable
4. Native Parallel Graphs like TigerGraph enable large scale feature extraction and in-graph analytics

Additional Resources

TigerGraph Cloud Page
https://www.tigergraph.com/cloud/

Download the Developer Edition
https://www.tigergraph.com/download/

Developer Portal
https://www.tigergraph.com/developers/

Guru Scripts
https://github.com/tigergraph/ecosys/tree/master/guru_scripts

Join our Developer Forum
https://groups.google.com/a/opengsql.org/forum/#!forum/gsql-users


@TigerGraphDB


youtube.com/tigergraph


facebook.com/TigerGraphDB


linkedin.com/company/TigerGraph

TigerGraph

Appendix

Neural Nework Cost function

The cost function J with regularization is defined as

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

where m is the number of training sets, K is the dimension of the output vector (number neurons in the output layer); y and $h_\Theta(x)$ are the labeled output and the predicted output respectively. The regularization parameter is specified by λ. L is the number of layers in the neural network, and s_l is the number of neurons in l^{th} layer.

Backpropagation and Gradient Descent

To train the values for the weight matrix $\Theta_{ji}^{(l)}$, the partial derivatives $\frac{\partial J}{\partial \Theta_{ji}^{(l)}}$ are needed. Each iteration $\Theta_{ji}^{(l)}$ are updated by $\Theta_{ji}^{(l)} = \Theta_{ji}^{(l)} - \alpha \frac{\partial J}{\partial \Theta_{ji}^{(l)}}$, where α is called the **learning rate**. The $\Theta_{ji}^{(l)}$ to minimize the cost function J are found where $\frac{\partial J}{\partial \Theta_{ji}^{(l)}} = 0$. The partial derivatives are computed by

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)} = \frac{1}{m} \Delta_{ij}^{(l)} \qquad \text{for } j = 0$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)} = \frac{1}{m} \Delta_{ij}^{(l)} + \frac{\lambda}{m} \Theta_{ij}^{(l)} \qquad \text{for } j \geq 1$$

where $\Delta^{(l)} = \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$ for all m training data sets.

For a three-layer neural network,

$$\delta_k^{(3)} = (a_k^{(3)} - y_k)$$

$$\delta^{(2)} = \left(\Theta^{(2)}\right)^T \delta^{(3)} .* g'(z^{(2)}), \quad g'(z) = \frac{d}{dz} g(z) = g(z)(1 - g(z))$$

TigerGraph

THANK YOU!