An Introduction to Graph Neural Networks: Models and Applications

Miltos Allamanis

AI Residency, Microsoft Research, Cambridge
Background
Machine Learning

"All models are wrong, some are useful"
– George Box

Machine Learning Model
Designed by humans

Model Parameters
Learned from data
Supervised Machine Learning

- Given an i.i.d. dataset \{ (x_1, y_1), ..., (x_N, y_N) \}
- Pick \( \theta \) that minimize Loss \( \mathcal{L}(\theta) = \frac{1}{N} \sum_i L(f_{\theta}(x_i), y_i) \)
Gradient Descent: Learning Model Parameters $\theta$

while not converged

• Compute (estimate of) derivative $g \approx \nabla_\theta \mathcal{L}(\theta)$
• Update $\theta \leftarrow \theta - q(g)$
Generalization

Underfitting

Overfitting

graphs from http://antianti.org/?p=175
Distributed Vector Representations

Local representation (1-hot)

Distributed representation

\[ \mathbf{r} = E \mathbb{I}_w \] with \( E \) a \( D \times V \) matrix

“vocabulary”
Graph Notation

- Nodes/Vertices
- Edges/Links

\[ G = (V, E) \]
Graph Neural Networks

and Neural Message Passing
Graph Neural Networks

Graph Representation of Problem

Initial Representation of each node
Graph Neural Networks

Initial Representation of each node

GNN

Output Representations of each Node

Task Specific Stuff + Loss
Graph Neural Networks

Initial Representation of each node

Output Representations of each Node

Task Specific Stuff + Loss
Graph Neural Networks
Neural Message Passing

Current Neighbor States

Prepare “Message”

Summarize Received Information

Next Node State

Current Node State
\[ h_t^n = q_t \left( h_{t-1}^n, \bigcup_{n_j : n_j \rightarrow n} f_t \left( h_{t-1}^n, k, h_{t-1}^{n_j} \right) \right) \]
Graph Neural Networks: Message Passing

t=0  t=1  t=2  t=3  t=4
GNNs: Synchronous Message Passing (All-to-All)
Graph Neural Networks: Output

- node selection
- node classification
- graph classification

https://github.com/microsoft/tf-gnn-samples/
Example: Node [Binary] Classification

\[ x_n = \sigma(w^T h^n_t + b) \]

Binary cross entropy

\[ \mathcal{L}(x_n, y_n) = y_n \cdot \log x_n + (1 - y_n) \log(1 - x_n) \]
Gated GNNs

\[ m = \sum_{n_j : n_j \rightarrow n} E_k h_{t-1}^{n_j} \]

\[ h_t^n = \sigma \left( \frac{1}{\text{numNeighbors} + 1} W_t \left( h_{t-1}^n + \sum_{n_j : n_j \rightarrow n} h_{t-1}^{n'} \right) \right) \]
Trick 1: Backwards Edges
Expressing GGNNS as Matrix Operations
Graph Notation (2) — Adjacency Matrix

\[ A = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \]
Graph Notation (2) — Adjacency Matrix

\[
A = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad N = \begin{bmatrix} a \\ b \\ c \end{bmatrix}
\]

\[
A \cdot N = \begin{bmatrix} 0 \\ a \\ a + b \end{bmatrix}
\]
Graph Notation (2) — Adjacency Matrix

\[
A_0 = \begin{bmatrix}
0 & 1 & 1 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix},
\]

\[
A_1 = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\]
If we used a vanilla RNN instead

$$H_{t+1} = \sigma(UH_t + WR_t)$$
Expressing GNN Matrix Operations as Code
\[ C = \text{np.einsum('bd,qd->bq', A, B)} \quad \# \quad C_{b,q} = \sum_d A_{b,d} B_{q,d} \]

\[ D = \text{np.einsum('abc,be,abq->cqe', A, B,C)} \]

\[ \# \quad D_{c,q,e} = \sum_b \sum_a A_{a,b,c} B_{b,e} C_{a,b,q} \]
GGNN as Pseudocode

```python
def GGNN(initial_node_states, adj, num_steps):
    node_states = initial_node_states  # [N, D]

    for i in range(num_steps):
        messages = {}
        for k in range(num_message_types):
            messages[k] = einsum('nd,dm->nm', node_states, edge_transform[k])  # [N, M]

        received_messages = zeros(num_nodes, M)  # [N, M]
        for k in range(num_message_types):
            received_messages += einsum('nm,nl->lm', messages[k], adj[k])

        node_states = GRU(node_states, received_messages)

    return node_states
```
GGNN as Pseudocode

def GGNN(initial_node_states, adj, num_steps):
    node_states = initial_node_states  # [N, D]

    for i in range(num_steps):
        messages = {}  
        for k in range(num_message_types):
            messages[k] = einsum('nd, dm->nm', node_states)

        received_messages = zeros(num_nodes, M)  # [N,]
        for k in range(num_message_types):
            received_messages += einsum('nm, nl->lm', messages[k], received_messages)

        node_states = GRU(node_states, received_messages)

    return node_states

Node States

\[ H_t = \begin{bmatrix} h_t^{n_0} \\ \vdots \\ h_t^{n_K} \end{bmatrix} \] (num_nodes x D)

Messages to-be sent

\[ M_t^k = E_k H_t \] (num_nodes x M)

Received Messages

\[ R_t = \sum_k A M_t^k \] (num_nodes x M)

Update \( H_{t+1} = GRU(H_t, R_t) \)
def GGNN(initial_node_states, adj, num_steps):
    node_states = initial_node_states # [N, D]
    for i in range(num_steps):
        messages = {}
        for k in range(num_message_types):
            messages[k] = einsum('nd,dm->nm', node_states, edge_transform[k]) # [N, M]
        received_messages = zeros(num_nodes, M) # [N, M]
        for k in range(num_message_types):
            received_messages += einsum('nm,nl->lm', messages[k], adj[k])
        node_states = GRU(node_states, received_messages)
    return node_states
Where are there the parameters?

def GGNN(initial_node_states, adj, num_steps):
    node_states = initial_node_states  # [N, D]

    for i in range(num_steps):
        messages = {}
        for k in range(num_message_types):
            messages[k] = einsum('nd, dm->nm', node_states, edge_transform[k])  # [N, M]

        received_messages = zeros(num_nodes, M)  # [N, M]
        for k in range(num_message_types):
            received_messages += einsum('nm, nl->lm', messages[k], adj[k])

        node_states = GRU(node_states, received_messages)

    return node_states

Find the parameters!
def GGNN(initial_node_states, adj, num_steps):
    node_states = initial_node_states  # [N, D]

    for i in range(num_steps):
        messages = {}
        for k in range(num_message_types):
            messages[k] = einsum(‘nd,dm->nm’, node_states, edge_transform[k])  # [N, M]

        received_messages = zeros(num_nodes, M)  # [N, M]
        for k in range(num_message_types):
            received_messages += einsum(‘nm,nl->lm’, messages[k], adj[k])

        node_states = GRU(node_states, received_messages)

    return node_states
Two Sample Applications

...with graph neural networks.
Variable Misuse Task

```csharp
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first=classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(first);

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Possible type-correct options: clazz, first
int SumPositive(int[] arr, int lim) {
    int sum = 0;
    for (int i = 0; i < lim; i++)
        if (arr[i] > 0)
            sum += arr[i];
    return sum;
}
Assert.NotNull(clazz);
Programs as Graphs: Data Flow

```
(x, y) = Foo();

while (x > 0)
    x = x + y;
```
Representing Program Structure as a Graph

Additional Edge Types:
• ReturnsTo

```c
int foo(int sum) {
    ...
    return x;
}
```
Representing Program Structure as a Graph

Additional Edge Types:
- ReturnsTo
- FormalArgName

```
void foo(int sum) {
    b = foo(result);
    sum
}
```
Programs as Graphs

```c
int SumPositive(int[] arr, int lim) {
    int sum = 0;
    for (int i = 0; i < lim; ++i) {
        if (arr[i] > 0)
            sum += arr[i];
    }
    return sum;
}
```

~900 nodes/graph  ~8k edges/graph
Initial Node Representations

Label: outFilePrefix
Type: string

Split to subtokens

out, file, prefix

Embed

Average

Concat

Max Pool

All implemented types

string, object, ...

Embed


Initial Node Representations
public readonly static Thickness MultLinePadding = new Thickness(0.0, 1.0, 0.0, 0.0);

public static IList<Rect> GetRectanglesFromBounds(IList<TextBounds> bounds)
{
    var newBounds = new List<Rect>(bounds.Count);
    foreach (var b in bounds)
    {
        double x1 = b.Left - padding.Left;
        double x2 = b.Right + padding.Right;
        if (x1 < x2)
        {
            double y1 = b.TextTop - padding.Top;
            double y2 = b.TextBottom + padding.Bottom;

            newBounds.Add(new Rect(x1, y1, x2 - x1, y2 - y1));
        }
    }

    return newBounds;
}

public static Rect GetRectangularBound(IList<Rect> rects)
{
    if (rects == null || rects.Count == 0)
    {
        return default(Rect);
    }

    Rect result = rects[0];
    foreach (var rect in rects)
    {
        double x1 = Math.Min(rect.Left, result.Left);
        double x2 = Math.Max(rect.Right, result.Right);
        double y1 = Math.Min(rect.Top, result.Top);
        double y2 = Math.Max(rect.Bottom, result.Bottom);

        result = new Rect(x1, y1, x2 - x1, y2 - y1);
    }

    return result;
}

// Set up the initial geometry.
Graph Representation for Variable Misuse

```csharp
var clazz = classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first = classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(first);

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Possible type-correct options: clazz, first
Graph Representation for Variable Misuse

```csharp
var clazz = classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first = classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(SLOT);

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

**Goal**: make the representation of SLOT as close as possible to the representation of the correct candidate node

\[ f \left( h_T^{SLOT}, h_T^{first} \right) \gg f \left( h_T^{SLOT}, h_T^{clazz} \right) \]
<table>
<thead>
<tr>
<th>Name</th>
<th>Git SHA</th>
<th>kLOCs</th>
<th>Slots</th>
<th>Vars</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akka.NET</td>
<td>719335a1</td>
<td>240</td>
<td>51.3k</td>
<td>51.2k</td>
<td>Actor-based Concurrent &amp; Distributed Framework</td>
</tr>
<tr>
<td>AutoMapper</td>
<td>2ca7c2b5</td>
<td>46</td>
<td>3.7k</td>
<td>10.7k</td>
<td>Object-to-Object Mapping Library</td>
</tr>
<tr>
<td>BenchmarkDotNet</td>
<td>1670ca34</td>
<td>28</td>
<td>5.1k</td>
<td>6.1k</td>
<td>Benchmarking Library</td>
</tr>
<tr>
<td>BotBuilder</td>
<td>190117c3</td>
<td>44</td>
<td>6.4k</td>
<td>8.7k</td>
<td>SDK for Building Bots</td>
</tr>
<tr>
<td>choco</td>
<td>93985688</td>
<td>36</td>
<td>3.8k</td>
<td>5.2k</td>
<td>Windows Package Manager</td>
</tr>
<tr>
<td>commandline†</td>
<td>09677b16</td>
<td>11</td>
<td>1.1k</td>
<td>2.3k</td>
<td>Command Line Parser</td>
</tr>
<tr>
<td>CommonMark.NETDev</td>
<td>f3d54530</td>
<td>14</td>
<td>2.6k</td>
<td>1.4k</td>
<td>Markdown Parser</td>
</tr>
<tr>
<td>Dapper</td>
<td>931c700d</td>
<td>18</td>
<td>3.3k</td>
<td>4.7k</td>
<td>Object Mapper Library</td>
</tr>
<tr>
<td>EntityFramework</td>
<td>fa0b7ec8</td>
<td>263</td>
<td>33.4k</td>
<td>39.3k</td>
<td>Object-Relational Mapper</td>
</tr>
<tr>
<td>Hangfire</td>
<td>ffc4912f</td>
<td>33</td>
<td>3.6k</td>
<td>6.1k</td>
<td>Background Job Processing Library</td>
</tr>
<tr>
<td>Humanizer†</td>
<td>cc11a77e</td>
<td>27</td>
<td>2.4k</td>
<td>4.4k</td>
<td>String Manipulation and Formatting</td>
</tr>
<tr>
<td>Lean†</td>
<td>f574bdf7</td>
<td>190</td>
<td>26.4k</td>
<td>28.3k</td>
<td>Algorithmic Trading Engine</td>
</tr>
<tr>
<td>Nancy</td>
<td>72e1f614</td>
<td>70</td>
<td>7.5k</td>
<td>15.7</td>
<td>HTTP Service Framework</td>
</tr>
<tr>
<td>Newtonsoft.Json</td>
<td>6057d9b8</td>
<td>123</td>
<td>14.9k</td>
<td>16.1k</td>
<td>JSON Library</td>
</tr>
<tr>
<td>Ninject</td>
<td>7006297f</td>
<td>13</td>
<td>0.7k</td>
<td>2.1k</td>
<td>Code Injection Library</td>
</tr>
<tr>
<td>NLog</td>
<td>643e326a</td>
<td>75</td>
<td>8.3k</td>
<td>11.0k</td>
<td>Logging Library</td>
</tr>
<tr>
<td>Opserver</td>
<td>51b032e7</td>
<td>24</td>
<td>3.7k</td>
<td>4.5k</td>
<td>Monitoring System</td>
</tr>
<tr>
<td>OptiKey</td>
<td>7d35c718</td>
<td>34</td>
<td>6.1k</td>
<td>3.9k</td>
<td>Assistive On-Screen Keyboard</td>
</tr>
<tr>
<td>Orleans</td>
<td>e0d6a150</td>
<td>300</td>
<td>30.7k</td>
<td>35.6k</td>
<td>Distributed Virtual Actor Model</td>
</tr>
<tr>
<td>Polly</td>
<td>0afdbc32</td>
<td>32</td>
<td>3.8k</td>
<td>9.1k</td>
<td>Resilience &amp; Transient Fault Handling Library</td>
</tr>
<tr>
<td>quartznet</td>
<td>b33e6f86</td>
<td>49</td>
<td>9.6k</td>
<td>9.8k</td>
<td>Scheduler</td>
</tr>
<tr>
<td>RavenDB.Dev</td>
<td>55230922</td>
<td>647</td>
<td>78.0k</td>
<td>82.7k</td>
<td>Document Database</td>
</tr>
<tr>
<td>RestSharp</td>
<td>70de357b</td>
<td>20</td>
<td>4.0k</td>
<td>4.5k</td>
<td>REST and HTTP API Client Library</td>
</tr>
<tr>
<td>Rx.NET</td>
<td>2d146fe5</td>
<td>180</td>
<td>14.0k</td>
<td>21.9k</td>
<td>Reactive Language Extensions</td>
</tr>
<tr>
<td>scriptcs</td>
<td>f3cc8bcb</td>
<td>18</td>
<td>2.7k</td>
<td>4.3k</td>
<td>C# Text Editor</td>
</tr>
<tr>
<td>ServiceStack</td>
<td>6d59da75</td>
<td>231</td>
<td>38.0k</td>
<td>46.2k</td>
<td>Web Framework</td>
</tr>
<tr>
<td>ShareX</td>
<td>718d711</td>
<td>125</td>
<td>22.3k</td>
<td>18.1k</td>
<td>Sharing Application</td>
</tr>
<tr>
<td>SignalR</td>
<td>fa88089e</td>
<td>53</td>
<td>6.5k</td>
<td>10.5k</td>
<td>Push Notification Framework</td>
</tr>
<tr>
<td>Wox</td>
<td>cdafe6272</td>
<td>13</td>
<td>2.0k</td>
<td>2.1k</td>
<td>Application Launcher</td>
</tr>
</tbody>
</table>
Quantitative Results – Variable Misuse

<table>
<thead>
<tr>
<th>Accuracy (%)</th>
<th>BiGRU</th>
<th>BiGRU+Dataflow</th>
<th>GGNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.0</td>
<td>73.7</td>
<td>85.5</td>
<td></td>
</tr>
</tbody>
</table>

Seen Projects: 24 F/OSS C# projects (2060 kLOC): Used for train and test

3.8 type-correct alternative variables per slot (median 3, \( \sigma = 2.6 \))
// Create or update the document.
var newDocument = await cosmosClient.UpsertDocumentAsync(cosmosDbCollectionUri, document);

if (updateRecord)
{
    logger.WriteLog("Updated {existingDocument} to {newDocument}");
}
else
{
    logger.WriteLog("Added {existingDocument}");
}

Based on this repo's code patterns, did you intend to use 'newDocument' (confidence 92%) rather than 'existingDocument' (confidence 7%) here? Review is recommended by Research bot's Variable Misuse analysis.

JK ++1
bool TryFindGlobalDirectivesFile(string baseDirectory, string fullPath, out string path) {
    baseDirectory = baseDirectory.TrimEnd(Path.DirectorySeparatorChar);
    var directivesDirectory = Path.GetDirectoryName(baseDirectory)
        .TrimEnd(Path.DirectorySeparatorChar);
    while (directivesDirectory != null && directivesDirectory.Length >= baseDirectory.Length) {
        path = Path.Combine(directivesDirectory, GlobalDirectivesFileName);
        if (File.Exists(path)) return true;

        directivesDirectory = Path.GetDirectoryName(directivesDirectory)
            .TrimEnd(Path.DirectorySeparatorChar);
    }
    path = null;
    return false;
}

What the model sees...
Other Models as Special Cases of GNNs
Special Case 1: Convolutions (CNN)
Special Case 2: “Deep Sets”

Set of Objects

Represent a variable-sized set of objects
Special Case 2: “Deep Sets”
“Advanced” GNNs
GGNs: Asynchronous Message Passing

- Define Schedule
- Send Messages
GGNs: Asynchronous Message Passing
RNNs are a special case of AsyncGNNs!
Machine Learning in Practice
Common Architecture of Deep Learning Code

1. Data Extraction
2. Compute Metadata
3. Convert to Tensors
4. Create Minibatches
5. Build ML Model
6. Training Loop

hyperparameters
This is your machine learning system?

Yup! You pour the data into this big pile of linear algebra, then collect the answers on the other side.

What if the answers are wrong?

Just stir the pile until they start looking right.
Practical (?) Tips on Debugging Machine Learning Models

Model Capacity (\textit{what can the model learn?})
- Overtrain on a small dataset
- Synthetic data

Optimization Issues (\textit{can we make the model learn?})
- Look at learning curves
- Monitor gradient update ratios
- Hand-pick parameters for synthetic data

Other model “bugs” (\textit{is the model doing what I want it to do?})
- Generate samples from your model (if you can)
- Visualize learned representations (\textit{e.g.} embeddings, nearest neighbors)
- Error analysis (examples where the model is failing, most “confident” errors)
- Simplify the problem/model
- Increase capacity, sweep hyperparameters (\textit{e.g.} increase size of $h$ in LSTM)

See also: https://youtu.be/oMB24_a05A
Learning Curve
Graph Neural Networks

\[ h^n_t = q_t \left( h^n_{t-1}, \bigcup_{n_i:n_i \rightarrow n} f_t(h^n_{t-1}, k, h^n_{t-1}) \right) \]