Supervised Learning vs. Reinforcement Learning: A Comparative Analysis for Designing Intelligent FutureG Networks

Martha Cash, Alexander Wyglinski

Worcester Polytechnic Institute

November 13^{th}, 2023

Tutorial Outline

Table of Contents

Speaker Introductions

B.S., M.S. Electrical Engineering, LSU
2^{nd} year Ph.D. student at WPI

Associate Dean of Graduate Studies, Professor of Electrical Engineering at WPI

Thank You To...

US Army Devcom 4.4 and 6.5

MIT Lincoln Laboratory

Worcester Polytechnic Institute

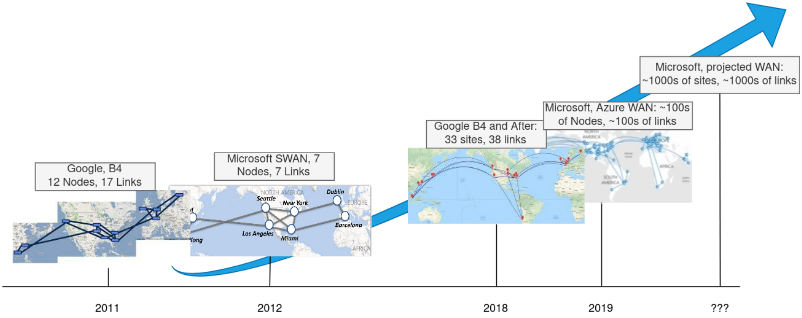
Table of Contents

Why This Tutorial?

Wireless networks are growing in size and complexity

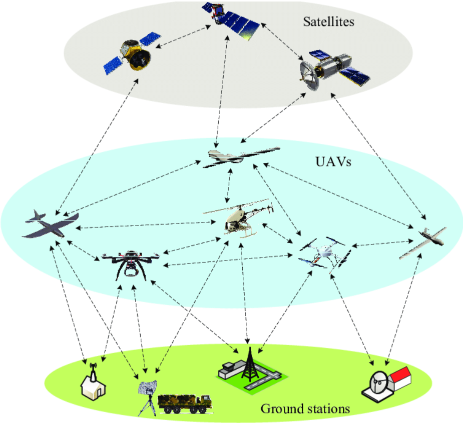
- OSPF struggling to meet increasing demands of high speed and low latency

Why This Tutorial?



Growth of Wide Area Networks [13]

Why This Tutorial?



Complex Mesh Network [8]

Why This Tutorial?

Resulting Challenges

- Need fast traffic allocation of routes in large networks

Why This Tutorial?

Resulting Challenges

- Need fast traffic allocation of routes in large networks
- Difficulty in predicting traffic demands across the network

Why This Tutorial?

Resulting Challenges

- Need fast traffic allocation of routes in large networks
- Difficulty in predicting traffic demands across the network
- Time-variant topologies makes routing challenging

Why This Tutorial?

How Should Routing Protocols Adapt?

- **Adaptability**: wireless networks are increasingly dynamic and subject to environmental changes

Why This Tutorial?

How Should Routing Protocols Adapt?

- **Adaptability**: wireless networks are increasingly dynamic and subject to environmental changes
- **Complex Decision Making**: handle networks with large number of nodes and diverse traffic patterns

Why This Tutorial?

How Should Routing Protocols Adapt?

- **Adaptability**: wireless networks are increasingly dynamic and subject to environmental changes
- **Complex Decision Making**: handle networks with large number of nodes and diverse traffic patterns
- **Traffic Prediction**: analyze historical traffic demands and predict future demands to make more efficient routing decisions

Why This Tutorial?

How Do We Make 'Intelligent' Routing Decisions?

Leverage Machine Learning (ML)!

Why This Tutorial?

How Do We Make 'Intelligent' Routing Decisions?

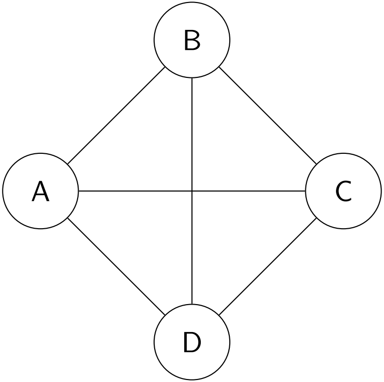
- Leverage information about **past** traffic conditions to learn good routing configurations for **future** conditions – **Supervised Learning** [12]
- Train an agent to learn to generate routing decisions from a history of observed traffic patterns – **Reinforcement Learning** [12]

Table of Contents

Preliminaries

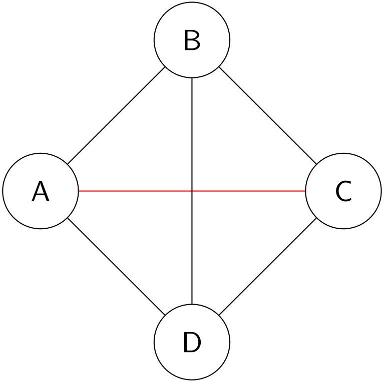
- Routing as network flows
- Representing traffic demands
- The MILP Problem

Routing as Network Flows



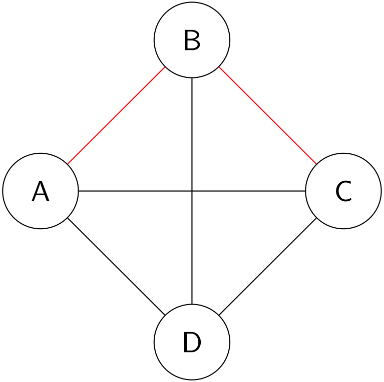
- Information sent over a network is divided into packets
- *Routing policy*: to which adjacent node should the current node send its packet in order to get it as quickly as possible to its eventual destination?
- *Policy*: minimize congestion, maximize throughput, minimize latency, etc.

Routing as Network Flows



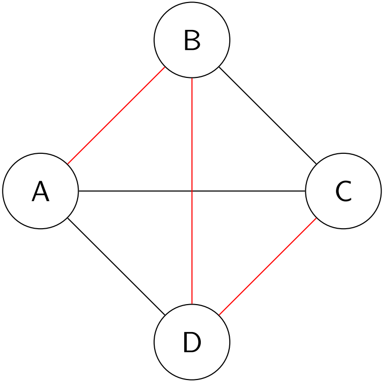
- Information sent over a network is divided into packets
- *Routing policy*: to which adjacent node should the current node send its packet in order to get it as quickly as possible to its eventual destination?
- *Policy*: minimize congestion, maximize throughput, minimize latency, etc.

Routing as Network Flows



- Information sent over a network is divided into packets
- *Routing policy*: to which adjacent node should the current node send its packet in order to get it as quickly as possible to its eventual destination?
- *Policy*: minimize congestion, maximize throughput, minimize latency, etc.

Routing as Network Flows



- Information sent over a network is divided into packets
- *Routing policy*: to which adjacent node should the current node send its packet in order to get it as quickly as possible to its eventual destination?
- *Policy*: minimize congestion, maximize throughput, minimize latency, etc.

Routing as Network Flows

Let $G = (V, E, c)$ be a capacitated directed graph
Routing Policy $\rightarrow$ Optimization Problem [12]

- $c : E \rightarrow \mathbb{R}^+$

Routing as Network Flows

Let $G = (V, E, c)$ be a capacitated directed graph
Routing Policy $\rightarrow$ Optimization Problem [12]

- $c : E \rightarrow \mathbb{R}^+$
- $s =$ source node

Routing as Network Flows

Let $G = (V, E, c)$ be a capacitated directed graph
Routing Policy $\rightarrow$ Optimization Problem [12]

- $c : E \rightarrow \mathbb{R}^+$
- s = source node
- t = destination node

Routing as Network Flows

Let $G = (V, E, c)$ be a capacitated directed graph
Routing Policy $\rightarrow$ Optimization Problem [12]

- $c : E \rightarrow \mathbb{R}^+$
- $s =$ source node
- $t =$ destination node
- $n = |V|$ and $\Gamma(\nu)$ denotes node ν neighbors in G

Routing as Network Flows

Let $G = (V, E, c)$ be a capacitated directed graph
Routing Policy $\rightarrow$ Optimization Problem [12]

- $c : E \rightarrow \mathbb{R}^+$
- s = source node
- t = destination node
- $n = |V|$ and $\Gamma(\nu)$ denotes node ν neighbors in G
- $R_{\nu,(s,t)} : \Gamma(\nu) \rightarrow [0, 1]$

Routing as Network Flows

Let $G = (V, E, c)$ be a capacitated directed graph
Routing Policy $\rightarrow$ Optimization Problem [12]

- $c : E \rightarrow \mathbb{R}^+$
- $s =$ source node
- $t =$ destination node
- $n = |V|$ and $\Gamma(\nu)$ denotes node ν neighbors in G
- $R_{\nu,(s,t)} : \Gamma(\nu) \rightarrow [0, 1]$
  - R, defines how traffic from a source node, s, and a destination node, t, traverses ν and is split among $\Gamma(\nu)$

Routing as Network Flows

- We don’t route packets one by one, but in groups

Routing as Network Flows

- We don't route packets one by one, but in groups
- **Flow**: A sequence of packets that share common characteristics between a source and destination in a network [6]

Routing as Network Flows

- We don't route packets one by one, but in groups
- **Flow**: A sequence of packets that share common characteristics between a source and destination in a network [6]
  - Denoted by $x_{(i,j)}$

$$\sum_{j:(i,j)\in E} x^{s,t}_{(i,j)} - \sum_{j:(i,j)\in E} x^{s,t}_{(j,i)} = \begin{cases} \Lambda_{s,t}, & \text{if } i = s \\ -\Lambda_{s,t}, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases}$$

Routing as Network Flows

- We don't route packets one by one, but in groups
- **Flow**: A sequence of packets that share common characteristics between a source and destination in a network [6]
  - Denoted by $x_{(i,j)}$

$$\sum_{j:(i,j)\in E} x^{s,t}_{(i,j)} - \sum_{j:(i,j)\in E} x^{s,t}_{(j,i)} = \begin{cases} \Lambda_{s,t}, & \text{if } i = s \\ -\Lambda_{s,t}, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases}$$

- **Capacity**: Every link has a maximum data rate. The flow across a link *must not* exceed the capacity

Routing as Network Flows

- We don't route packets one by one, but in groups
- **Flow**: A sequence of packets that share common characteristics between a source and destination in a network [6]
  - Denoted by $x_{(i,j)}$

$$\sum_{j:(i,j)\in E} x^{s,t}_{(i,j)} - \sum_{j:(i,j)\in E} x^{s,t}_{(j,i)} = \begin{cases} \Lambda_{s,t}, & \text{if } i = s \\ -\Lambda_{s,t}, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases}$$

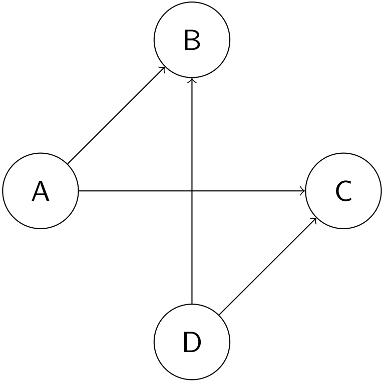
- **Capacity**: Every link has a maximum data rate. The flow across a link *must not* exceed the capacity
  - Denoted by c_{ij}

$$\sum_{s,t\in N} x^{s,t}_{(i,j)} \leq c_{(i,j)} z_{(i,j)}$$

Representing Traffic Demands

- Packets $\rightarrow$ Flows
- Packets originate from some demand $\rightarrow$ Traffic Demand Matrix
- **Demand Matrix**: $n \times n$ matrix where the $(i, j)^{th}$ entry, $d_{(i,j)}$, specifies the amount of traffic sourcing at node i destined for j [12]
- $R : d_{(i,j)} \rightarrow x_{(i,j)}$

Representing Traffic Demands



$$\begin{array}{c c} & \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \end{array} & \begin{pmatrix} 0 & \mathrm{d}_{AB} & \mathrm{d}_{AC} & 0 \\ \mathrm{d}_{AB} & 0 & 0 & \mathrm{d}_{DB} \\ \mathrm{d}_{AC} & 0 & 0 & \mathrm{d}_{DC} \\ 0 & \mathrm{d}_{DB} & \mathrm{d}_{DC} & 0 \end{pmatrix} \end{array}$$

Routing as an Optimization Problem

Routing Policy Objectives

- Maximize Throughput
- Minimize (Max) Link Utilization
- Maximize Flow Minimize Cos

Maximize Throughput [9]

$$
\begin{aligned}
\text{Maximize} \quad & \rho \\
\text{subject to:} \quad & \sum_{j:(i,j)\in E} x^{s,t}_{(i,j)} - \sum_{j:(i,j)} x^{s,t}_{j,i} = \begin{cases} \rho\Lambda_{s,t}, & \text{if } i = s \\ -\rho\Lambda_{s,t}, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases} \\
& \sum_{s,t\in N} x^{s,t}_{(i,j)} \leq c_{(i,j)} z_{(i,j)} \\
& \sum_{j} z_{(i,j)} \leq T, \ \ \forall i, T = m \\
& z_{(i,j)} = z_{(j,i)} \ \ \forall (i,j) \, \forall (s,t) \\
& z_{ij} \in 0, 1 \ \ \forall (i,j) \\
& x^{s,t}_{i,j} \geq 0 \\
& \rho \geq 0
\end{aligned}
$$

Minimize (Max) Link Utilization [3]

$$\text{Minimize} \quad \frac{\sum_{s,t \in N} x_{(i,j)}^{s,t}}{c_{(i,j)}} \quad \forall (i,j) \in E$$

$$\text{subject to:} \quad \sum_{j:(i,j) \in E} x_{(i,j)}^{s,t} - \sum_{j:(i,j)} x_{j,i}^{s,t} = \begin{cases} \Lambda_{s,t}, & \text{if } i = s \\ -\Lambda_{s,t}, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases}$$

$$\sum_{s,t \in N} x_{(i,j)}^{s,t} \leq c_{ij} z_{ij}$$

$$\sum_{j} z_{(i,j)} \leq T, \ \ \forall i, T = m$$

$$z_{(i,j)} = z_{(j,i)} \ \ \forall (i,j) \forall (s,t)$$

$$z_{ij} \in 0, 1 \ \ \forall (i,j)$$

$$x_{i,j}^{s,t} \geq 0$$

A Road Map

$$
\begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,n} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m,1} & t_{m,2} & \cdots & t_{m,n} \end{pmatrix}
\longrightarrow
\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix}
\longrightarrow
\begin{array}{ll} \text{Maximize} & f(x) \\ \text{Subject to} & g_i(x) \geq 0 \end{array}
$$

Traffic Demands **Network Flows** **Routing Policy**

Where does machine learning come into play?

A Road Map

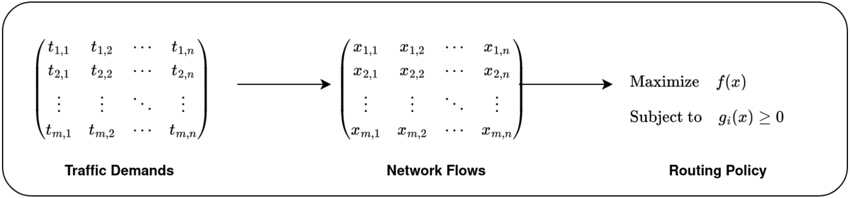
$$\begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,n} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m,1} & t_{m,2} & \cdots & t_{m,n} \end{pmatrix} \longrightarrow \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix} \longrightarrow \begin{array}{ll} \text{Maximize} & f(x) \\ \text{Subject to} & g_i(x) \geq 0 \end{array}$$

Traffic Demands | **Network Flows** | **Routing Policy**

Traffic Demands & Network Flows

- Predict future traffic demands
- Predict network flows
- Supervised Learning

A Road Map



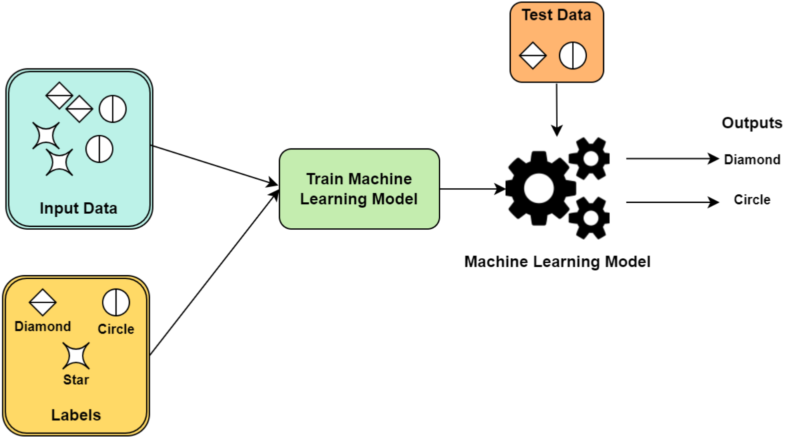
Routing Policy

- Replace optimizer with ML model
- Teach ML model to replace optimizer
- Supervised Learning & Reinforcement Learning

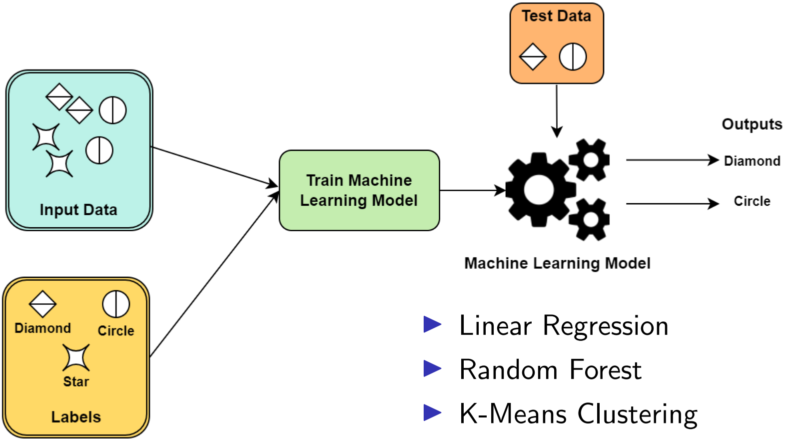
Table of Contents

Supervised Learning – Learning By Example

A Machine Learning (ML) training approach that uses labeled data to predict outcomes or categorize data

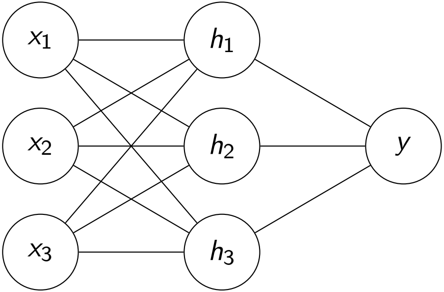


Supervised Learning – Learning By Example



- Linear Regression
- Random Forest
- K-Means Clustering
- **Neural Networks**

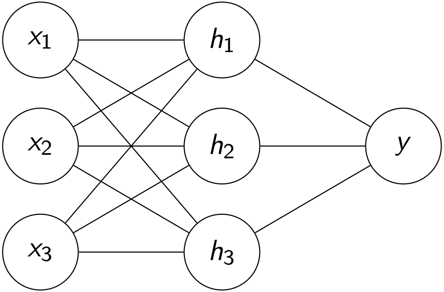
Supervised Learning – Neural Networks [2]



Simple Feed Forward Neural Network

- Approximate a function f^*

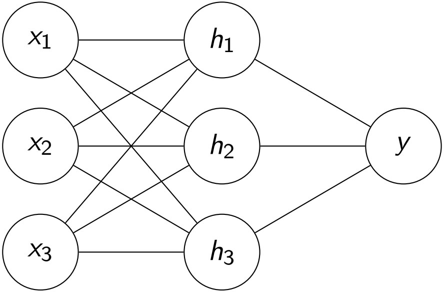
Supervised Learning – Neural Networks [2]



Simple Feed Forward Neural Network

- Approximate a function f^*
- Defines a mapping for $y = f(\boldsymbol{x}; \boldsymbol{\theta})$

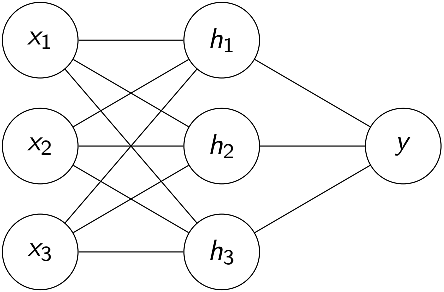
Supervised Learning – Neural Networks [2]



Simple Feed Forward Neural Network

- Approximate a function f^*
- Defines a mapping for $y = f(\boldsymbol{x}; \boldsymbol{\theta})$
- Learns parameters $\boldsymbol{\theta}$

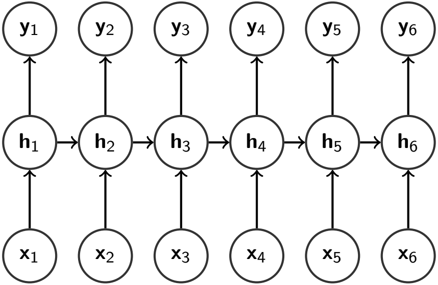
Supervised Learning – Neural Networks [2]



Simple Feed Forward Neural Network

- Approximate a function f^*
- Defines a mapping for $y = f(\boldsymbol{x}; \boldsymbol{\theta})$
- Learns parameters $\boldsymbol{\theta}$
- Minimize error $f^*(\boldsymbol{x})$ and $f(\boldsymbol{x})$ via SGD

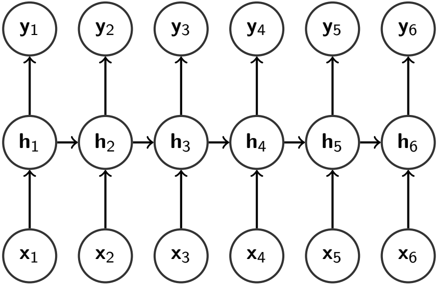
Supervised Learning – Neural Networks [2]



A simple Recurrent Neural Network

- RNNs are designed to process a sequence of inputs

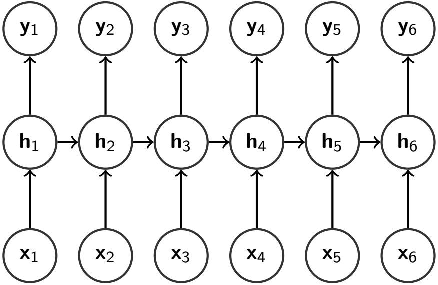
Supervised Learning – Neural Networks [2]



A simple Recurrent Neural Network

- RNNs are designed to process a sequence of inputs
- RNNs take the output of the previous cell as input to the current cell

Supervised Learning – Neural Networks [2]

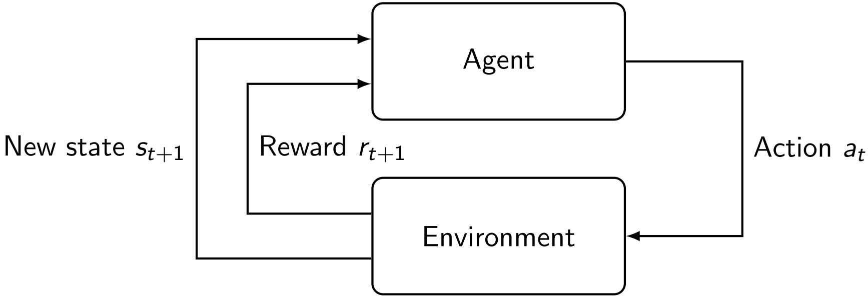


A simple Recurrent Neural Network

- RNNs are designed to process a sequence of inputs
- RNNs take the output of the previous cell as input to the current cell
- RNNs can learn temporal patterns in data

Reinforcement Learning – Learning What to Do

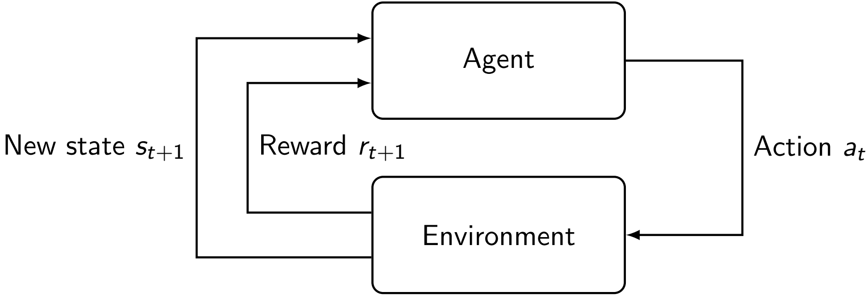
How to map situations to actions to maximize some numerical reward [10]



Action-Reward Feedback Loop [10]

Reinforcement Learning – Learning What to Do

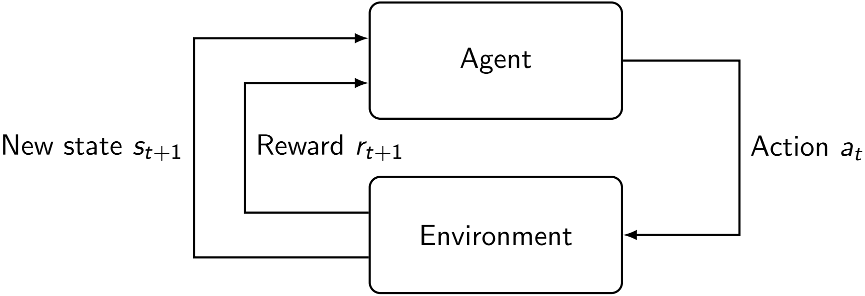
- $t = 1, 2, 3, \ldots$



Action-Reward Feedback Loop [10]

Reinforcement Learning – Learning What to Do

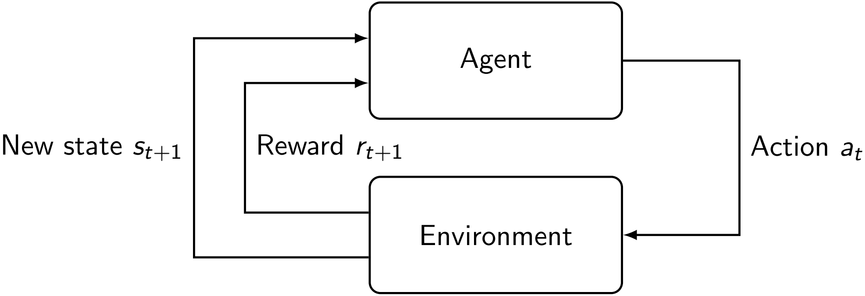
- $t = 1, 2, 3, ...$
- Observe S_{t-1} and select $a_t \in A$



Action-Reward Feedback Loop [10]

Reinforcement Learning – Learning What to Do

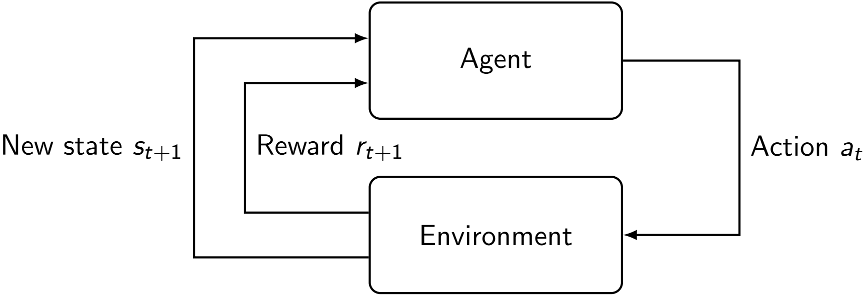
- $t = 1, 2, 3, \ldots$
- Observe S_{t-1} and select $a_t \in A$
- $a_t \to s_t$ and receives r_t



Action-Reward Feedback Loop [10]

Reinforcement Learning – Learning What to Do

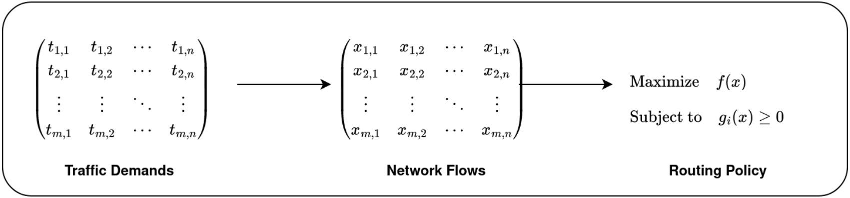
- $t = 1, 2, 3, \ldots$
- Observe S_{t-1} and select $a_t \in A$
- $a_t \rightarrow s_t$ and receives r_t
- $\pi : S \rightarrow A$ w.r.t. $E[\sum_t \gamma^t r_t]$



Action-Reward Feedback Loop [10]

Table of Contents

Predicting Traffic Demands



Road Map

Traffic Demands & Network Flows

- Predict future traffic demands
- Supervised Learning

Predicting Traffic Demands

Synthetic Traffic Demands

- Gravity Model

Predicting Traffic Demands

Synthetic Traffic Demands

- Gravity Model
- Bimodal Model

Predicting Traffic Demands

Gravity Model [5]

- We can model traffic from a source node, i, to a destination node, j, as a random process

Predicting Traffic Demands

Gravity Model [5]

- We can model traffic from a source node, i, to a destination node, j, as a random process
- Packets from a source node to a destination node are independent

Predicting Traffic Demands

Gravity Model [5]

- We can model traffic from a source node, i, to a destination node, j, as a random process
- Packets from a source node to a destination node are independent
- Generate traffic following an exponential random variable

$$x_{(i,j)} = \frac{x_i^{in} x_j^{out}}{\sum_{k=1}^{n} x_k^{out}}$$

Predicting Traffic Demands

Gravity Model [5]

- We can model traffic from a source node, i, to a destination node, j, as a random process
- Packets from a source node to a destination node are independent
- Generate traffic following an exponential random variable

$$x_{(i,j)} = \frac{x_i^{in} x_j^{out}}{\sum_{k=1}^{n} x_k^{out}}$$

- Assume $x^{total} = \sum_{k=1}^{n} x_k^{out} = \sum_{k=1}^{n} x_k^{in}$

Predicting Traffic Demands

Bimodal Model [7]

- Generated via a mixture of two Gaussian distributions

Predicting Traffic Demands

Bimodal Model [7]

- Generated via a mixture of two Gaussian distributions
- N_1 (μ_1, σ_1) and N_2 (μ_2, σ_2), specified in same units as data rate

Predicting Traffic Demands

Bimodal Model [7]

- Generated via a mixture of two Gaussian distributions
- $N_1\ (\mu_1, \sigma_1)$ and $N_2\ (\mu_2, \sigma_2)$, specified in same units as data rate
- For each (i, j) pair, $P(x_{(i,j)} = N_1) = p$ and $P(x_{(i,j)} = N_1) = 1 - p$

Predicting Traffic Demands

Synthetic Traffic Demand Generation Framework [12]

- Control DM size, ie: 9×9, 23×23, etc...

Predicting Traffic Demands

Synthetic Traffic Demand Generation Framework [12]

- Control DM size, ie: 9×9, 23×23, etc...
- Consider "sparsficiation" of DMs, ie: sample a p-fraction of the communicating pairs, $p \in [0, 1]$

Predicting Traffic Demands

Synthetic Traffic Demand Generation Framework [12]

- Control DM size, ie: 9×9, 23×23, etc...
- Consider "sparsficiation" of DMs, ie: sample a p-fraction of the communicating pairs, $p \in [0, 1]$
- Mimic regularity by creating a cycle of DMs:

Predicting Traffic Demands

Synthetic Traffic Demand Generation Framework [12]

- Control DM size, ie: 9×9, 23×23, etc...
- Consider "sparsficiation" of DMs, ie: sample a p-fraction of the communicating pairs, $p \in [0, 1]$
- Mimic regularity by creating a cycle of DMs:
  - $t - 1 : D^0, \cdots, D^{q-1}$

Predicting Traffic Demands

Synthetic Traffic Demand Generation Framework [12]

- Control DM size, ie: 9×9, 23×23, etc...
- Consider "sparsficiation" of DMs, ie: sample a p-fraction of the communicating pairs, $p \in [0, 1]$
- Mimic regularity by creating a cycle of DMs:
  - $t - 1 : D^0, \cdots, D^{q-1}$
  - $D^{(j)} \in t - 1$

Predicting Traffic Demands

Synthetic Traffic Demand Generation Framework [12]

- Control DM size, ie: 9×9, 23×23, etc...
- Consider "sparsficiation" of DMs, ie: sample a p-fraction of the communicating pairs, $p \in [0, 1]$
- Mimic regularity by creating a cycle of DMs:
  - $t - 1 : D^0, \cdots, D^{q-1}$
  - $D^{(j)} \in t - 1$
  - $D^{(j+1 modulo\ q)} \in t$

Predicting Traffic Demands

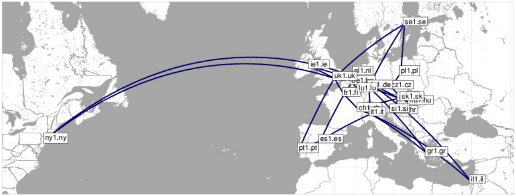
Real Network Traffic Datasets
Generating synthetic datasets leads to inherent biases in validating the SL approach

Abeline Dataset

- 48096 Demand Matrices
- Granularity: 5min
- Time Horizon: 6 months

Predicting Traffic Demands

Real Network Traffic Datasets
Generating synthetic datasets leads to inherent biases in validating the SL approach



GÉANT Dataset [11]

- 11460 Demand Matrices
- Granularity: 15min
- Time Horizon: 4 months

Predicting Traffic Demands

Traffic Demand Prediction – Work Flow [1], [4], [6]

- Let $D = n \times n$ demand matrix

Predicting Traffic Demands

Traffic Demand Prediction – Work Flow [1], [4], [6]

- Let $D = n \times n$ demand matrix
- We form a training set of $n \times n \times T \times M$ matrices

Predicting Traffic Demands

Traffic Demand Prediction – Work Flow [1], [4], [6]

- Let $D = n \times n$ demand matrix
- We form a training set of $n \times n \times T \times M$ matrices
  - $T =$ time series dimension

Predicting Traffic Demands

Traffic Demand Prediction – Work Flow [1], [4], [6]

- Let $D = n \times n$ demand matrix
- We form a training set of $n \times n \times T \times M$ matrices
  - T = time series dimension
  - M = number of time series in the training set

Predicting Traffic Demands

Traffic Demand Prediction – Work Flow [1], [4], [6]

- Let $D = n \times n$ demand matrix
- We form a training set of $n \times n \times T \times M$ matrices
  - $T =$ time series dimension
  - $M =$ number of time series in the training set
- Learn to predict D_t given $D_{t-T}, D_{t-T+1}, \ldots, D_{t-2}, D_{t-1}$

Predicting Traffic Demands

Traffic Demand Prediction – Work Flow [1], [4], [6]

- Let $D = n \times n$ demand matrix
- We form a training set of $n \times n \times T \times M$ matrices
  - T = time series dimension
  - M = number of time series in the training set
- Learn to predict D_t given $D_{t-T}, D_{t-T+1}, \ldots, D_{t-2}, D_{t-1}$
- Evaluation Metric: Pair-wise MSE

Predicting Traffic Demands

Traffic Demand Prediction – Work Flow [1], [4], [6]

- Let $D = n \times n$ demand matrix
- We form a training set of $n \times n \times T \times M$ matrices
  - $T =$ time series dimension
  - $M =$ number of time series in the training set
- Learn to predict D_t given $D_{t-T}, D_{t-T+1}, \ldots, D_{t-2}, D_{t-1}$
- Evaluation Metric: Pair-wise MSE
- $D_t \rightarrow R_t$

Predicting Traffic Demands

Traffic Demand Prediction – Work Flow [1], [4], [6]

- Let $D = n \times n$ demand matrix
- We form a training set of $n \times n \times T \times M$ matrices
  - T = time series dimension
  - M = number of time series in the training set
- Learn to predict D_t given $D_{t-T}, D_{t-T+1}, \ldots, D_{t-2}, D_{t-1}$
- Evaluation Metric: Pair-wise MSE
- $D_t \rightarrow R_t$
- Relevant ML Models: **RNN**

Predicting Traffic Demands + Learning Routing Policy

$$\begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,n} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m,1} & t_{m,2} & \cdots & t_{m,n} \end{pmatrix} \longrightarrow \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix} \longrightarrow \begin{array}{ll} \text{Maximize} & f(x) \\ \text{Subject to} & g_i(x) \geq 0 \end{array}$$

Traffic Demands **Network Flows** **Routing Policy**

Road Map

Traffic Demands & Routing Policy

- Predict future traffic demands
- Train ML model to replace optimizer

Predicting Traffic Demands + Learning Routing Policy

Train two separate NNs

- Train a **RNN** to predict future traffic demands (as previously described)

Predicting Traffic Demands + Learning Routing Policy

Train two separate NNs

- Train a **RNN** to predict future traffic demands (as previously described)
- Train a **FFNN** to learn to match traffic demands to routing paths

Predicting Traffic Demands + Learning Routing Policy

Train two separate NNs

- Train a **RNN** to predict future traffic demands (as previously described)
- Train a **FFNN** to learn to match traffic demands to routing paths
  - Recall previous optimization problems: maximize throughput, minimize congestion

Predicting Traffic Demands + Learning Routing Policy

Train two separate NNs

- Train a **RNN** to predict future traffic demands (as previously described)
- Train a **FFNN** to learn to match traffic demands to routing paths
  - Recall previous optimization problems: maximize throughput, minimize congestion
  - Take DM training set $\rightarrow$ get MILP output

Predicting Traffic Demands + Learning Routing Policy

Train two separate NNs

- Train a **RNN** to predict future traffic demands (as previously described)
- Train a **FFNN** to learn to match traffic demands to routing paths
  - Recall previous optimization problems: maximize throughput, minimize congestion
  - Take DM training set $\rightarrow$ get MILP output
  - DMs are input to FFNN, routing paths are output

Predicting Traffic Demands + Learning Routing Policy

Train two separate NNs

- Train a **RNN** to predict future traffic demands (as previously described)
- Train a **FFNN** to learn to match traffic demands to routing paths
  - Recall previous optimization problems: maximize throughput, minimize congestion
  - Take DM training set $\rightarrow$ get MILP output
  - DMs are input to FFNN, routing paths are output
  - Compare FFNN routing paths with MILP output

Predicting Traffic Demands + Learning Routing Policy

Train two separate NNs

- Train a **RNN** to predict future traffic demands (as previously described)
- Train a **FFNN** to learn to match traffic demands to routing paths
  - Recall previous optimization problems: maximize throughput, minimize congestion
  - Take DM training set $\rightarrow$ get MILP output
  - DMs are input to FFNN, routing paths are output
  - Compare FFNN routing paths with MILP output
- At time t, generate D^{t+1} and R^{t+1}

Learning Routing Policy via Reinforcement Learning

Do not need to train two separate models

Train one agent to learn a routing strategy

- Dataset consists of tuples: (S, A, P_a, R_a)

Learning Routing Policy via Reinforcement Learning

Do not need to train two separate models

Train one agent to learn a routing strategy

- Dataset consists of tuples: (S, A, P_a, R_a)
- Action space is too large: $|V|^2 \cdot |E|$

Learning Routing Policy via Reinforcement Learning

Do not need to train two separate models
Train one agent to learn a routing strategy

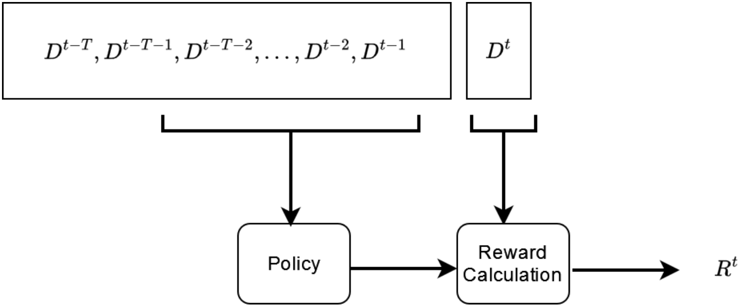
- Dataset consists of tuples: (S, A, P_a, R_a)
- Action space is too large: $|V|^2 \cdot |E|$
- *destination-based routing*: splitting ratios at each node, ν, w.r.t a destination, d, are the same across all sources, s

Learning Routing Policy via Reinforcement Learning

Do not need to train two separate models
Train one agent to learn a routing strategy

- Dataset consists of tuples: (S, A, P_a, R_a)
- Action space is too large: $|V|^2 \cdot |E|$
- *destination-based routing*: splitting ratios at each node, ν, w.r.t a destination, d, are the same across all sources, s
- Action space: $|V| \cdot |E|$

Learning Routing Policy via Reinforcement Learning



Previous *n* demands are given to the policy. The policy maps the set of demands and current network state to a set of actions. The reward calculator selects the set of actions that maximizes [3]

Table of Contents

Conclusions

- Need intelligent routing solutions for future networks

Conclusions

- Need intelligent routing solutions for future networks
- Supervised Learning & Reinforcement Learning

Conclusions

- Need intelligent routing solutions for future networks
- Supervised Learning & Reinforcement Learning
- Supervised Learning

Conclusions

- Need intelligent routing solutions for future networks
- Supervised Learning & Reinforcement Learning
- Supervised Learning
  - Predict traffic demands + routing policy

Conclusions

- Need intelligent routing solutions for future networks
- Supervised Learning & Reinforcement Learning
- Supervised Learning
  - Predict traffic demands + routing policy
  - Predict traffic demands + learn routing policy

Conclusions

- Need intelligent routing solutions for future networks
- Supervised Learning & Reinforcement Learning
- Supervised Learning
  - Predict traffic demands + routing policy
  - Predict traffic demands + learn routing policy
- Reinforcement Learning

Conclusions

- Need intelligent routing solutions for future networks
- Supervised Learning & Reinforcement Learning
- Supervised Learning
  - Predict traffic demands + routing policy
  - Predict traffic demands + learn routing policy
- Reinforcement Learning
  - Directly learn to routing policy

Conclusions

- Need intelligent routing solutions for future networks
- Supervised Learning & Reinforcement Learning
- Supervised Learning
  - Predict traffic demands + routing policy
  - Predict traffic demands + learn routing policy
- Reinforcement Learning
  - Directly learn to routing policy
  - Agent is making routing decisions

Conclusions

When to use each approach?

- Supervised Learning

Conclusions

When to use each approach?

- Supervised Learning
  - Historical data with labeled examples

Conclusions

When to use each approach?

- Supervised Learning
  - Historical data with labeled examples
  - Predictable and stable environments

Conclusions

When to use each approach?

- Supervised Learning
  - Historical data with labeled examples
  - Predictable and stable environments
  - Explicit knowledge of optimal routes

Conclusions

When to use each approach?

- Supervised Learning
  - Historical data with labeled examples
  - Predictable and stable environments
  - Explicit knowledge of optimal routes
- Reinforcement Learning

Conclusions

When to use each approach?

- Supervised Learning
  - Historical data with labeled examples
  - Predictable and stable environments
  - Explicit knowledge of optimal routes
- Reinforcement Learning
  - Dynamic and Unknown Environments

Conclusions

When to use each approach?

- Supervised Learning
  - Historical data with labeled examples
  - Predictable and stable environments
  - Explicit knowledge of optimal routes
- Reinforcement Learning
  - Dynamic and Unknown Environments
  - Networks that need to adapt persistently

Conclusions

When to use each approach?

- Supervised Learning
  - Historical data with labeled examples
  - Predictable and stable environments
  - Explicit knowledge of optimal routes
- Reinforcement Learning
  - Dynamic and Unknown Environments
  - Networks that need to adapt persistently
  - Don't have explicit knowledge

Thank You, Questions?

[1] Abdelhadi Azzouni and Guy Pujolle. "NeuTM: A neural network-based framework for traffic matrix prediction in SDN". In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. 2018, pp. 1–5. DOI: 10.1109/NOMS.2018.8406199.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[3] Oliver Hope and Eiko Yoneki. "GDDR: GNN-based Data-Driven Routing". In: *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. 2021, pp. 517–527. DOI: 10.1109/ICDCS51616.2021.00056.

[4] Duc-Huy Le et al. "An AI-based Traffic Matrix Prediction Solution for Software-Defined Network". In: *ICC 2021 - IEEE International Conference on Communications*. 2021, pp. 1–6. DOI: 10.1109/ICC42927.2021.9500331.

[5] A. Medina et al. "Traffic Matrix Estimation: Existing Techniques and New Directions". In: *Proceedings of the*