# Learning heuristics for graph algorithms

Benjamin Negrevergne, kickoff Esigma

# Tools for data analysis

## Historically

- Focus on generality
  - Data mining algorithms that can work with different algebraic structures
  - Data mining algorithms that can work with different constraints
- Focus on efficiency
  - parallel and distributed computing

## Nowadays

- Topic 1: Detecting salient events in game data
- Topic 2: Recognizing style automatically (painting/music)
- Topic 3: Distributed deep learning

## Next

- Learning solving strategies for CSP solving

# Supervised learning

## Problem

- $f$ : unknown, hard to specify
- $E$ : a set of examples $(x, f(x))$

▶ Goal, find $\hat{f}$ such that

$$\hat{f} \sim f$$

## Approach

- We choose a family of function $\hat{f} = f_\theta$ parameterized with $\theta$
- We optimize the parameters $\theta$ such that
  - $f_\theta$ minimize the error on the given examples
  - $f_\theta$ generalizes well on new examples drawn from the same distribution

# Supervised learning

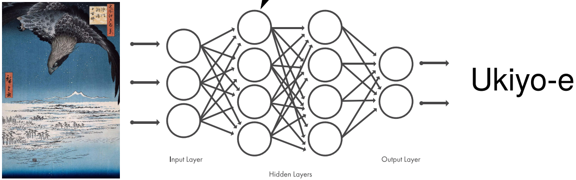
- We can learn complex relations between inputs and outputs

$f\Bigg($  $\Bigg) \mapsto$ *Baroque*

# Deep learning

We represent $f_\theta$ as a neural network organized in layers.

$$h_i = g(\sum_{j=1}^{n} x_j \cdot \theta_{ij})$$

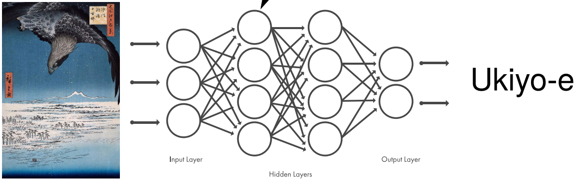



- Each layer learns a vector representation tuned for the next layer.

# Deep learning

We represent $f_\theta$ as a neural network organized in layers.

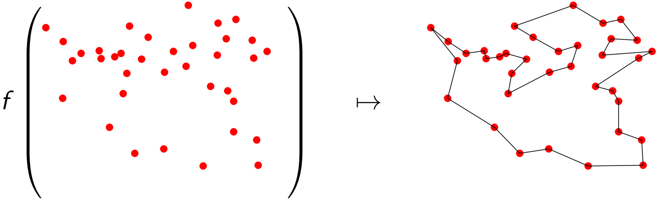
$$h_i = g(\sum_{j=1}^{n} x_j \cdot \theta_{ij})$$




- Each layer learns a vector representation tuned for the next layer.

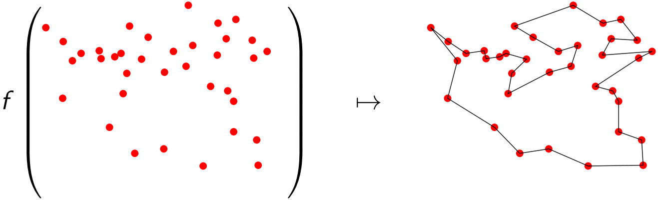
**Main takeaway:** do not try to specify the features yourself.

# Learning in graph algorithmics



Can we learn $f$ ?

# Learning in graph algorithmics




Can we learn $f$ ?

- What is there to learn?
- How to formalize the learning task?

# Learning Combinatorial Optimization Algorithms over Graphs

H. Dai et al., NIPS, 2017

# Learning Combinatorial Optimization Algorithms over Graphs

H. Dai et al., NIPS, 2017

**What is there to learn?**

▶ Given a graph optimization problem $G$ and a distribution $\mathbb{D}$ of problem instances, can we learn a heuristic that generalizes to unseen instances of $\mathbb{D}$

# How to formalize the learning task ?

As a regression task

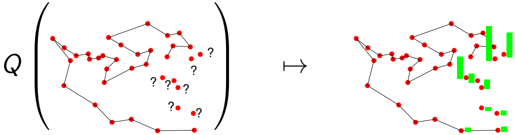
**Given**

- A set $X$ of points in $\mathbb{R}^2$, with $|X| = n$
- a path $p = (x_1, \ldots, x_m)$ with $m < n$

  We denote $P = (X, p)$ is a partially solved TSP

**Learn**

$$Q : (P, x) \mapsto \textit{minimal-final-length}(P \sqcup x)$$




**TSP- ml (P, p)**:
if $\exists x \in X \setminus P$
$\quad TSP\text{-}ml(P \sqcup argmax_x(Q(P, x))$

# How can we learn $Q$

▶ Machine learning algorithms works with vectors

**Problem:** how to represent a vertex $v$ as a vector $\mu_v$?

- $\mu_v \ = (x_v, y_v)$ ▶ only captures local properties of $v$

# How can we learn $Q$

▶ Machine learning algorithms works with vectors
**Problem:** how to represent a vertex $v$ as a vector $\mu_v$?

- ${\mu_v}^0 = (x_v, y_v)$
- $\mu_v^1 = F(\mu_0, \{\mu_u^0\}_{u \in N(v)}, \{w(v,u)\}_{u \in N(v)})$
- ...
- $\mu_v^{(t+1)} = F(\mu_t, \{\mu_u^t\}_{u \in N(v)}, \{w(v,u)\}_{u \in N(v)})$

Where $F$ is the aggregation function

# How can we learn $Q$

▶ Machine learning algorithms works with vectors
**Problem:** how to represent a vertex $v$ as a vector $\mu_v$?

- ${\mu_v}^0 = (x_v, y_v)$
- $\mu_v^1 = F(\mu_0, \{\mu_u^0\}_{u \in N(v)}, \{w(v,u)\}_{u \in N(v)})$
- $\ldots$
- $\mu_v^{(t+1)} = F(\mu_t, \{\mu_u^t\}_{u \in N(v)}, \{w(v,u)\}_{u \in N(v)})$

Where $F$ is the aggregation function

▶ After $T$ iteration, each node embedding $\mu_v^T$ will contain information about its T-hop neighbors

# How to choose $F$

How to choose the aggregation function $F$ ?

- should include some notion of the degree other local vs. global statistics
- should probably be highly non-linear
- should depend on the problem were are looking at

# How to choose $F$

How to choose the aggregation function $F$ ?

- should include some notion of the degree other local vs. global statistics
- should probably be highly non-linear
- should depend on the problem were are looking at

**Take away from before:** do not try to specify the features yourself!
▶ End-to-end learning together with $Q$ for the task at hand

# How to choose $F$

How to choose the aggregation function $F$ ?

- should include some notion of the degree other local vs. global statistics
- should probably be highly non-linear
- should depend on the problem were are looking at

**Take away from before:** do not try to specify the features yourself!
▶ End-to-end learning together with $Q$ for the task at hand

$$\mu^{(t+1)}(v) = relu(\theta_1 x_v + \theta_2 \sum_{u \in N(v)} \mu_u(t) + \theta_3 \sum_{u \in N(v)} relu(\theta_4 w(v,u)))$$

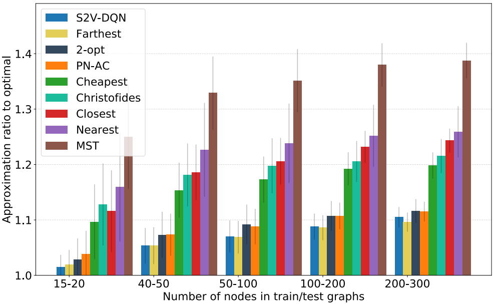
Model parameters: $\theta_1 \in \mathbb{R}^p, \theta_2, \theta_3 \in \mathbb{R}^{p \times p}, \theta_4 \in \mathbb{R}^p$

# Reinforcement learning

1. Solve a TSP
2. Observe reward
3. Update parameters of $Q$ and $F$

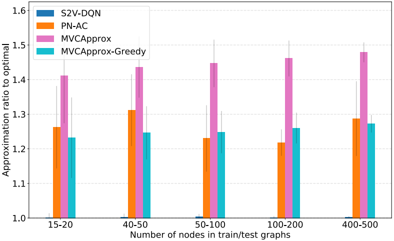
$\varepsilon -$ *greedy* algorithm to balance exploration / exploitation

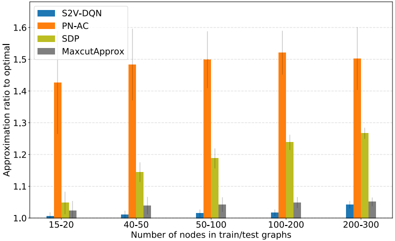
# Quality of the approximation



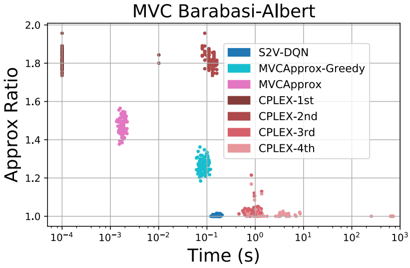

(c) TSP random

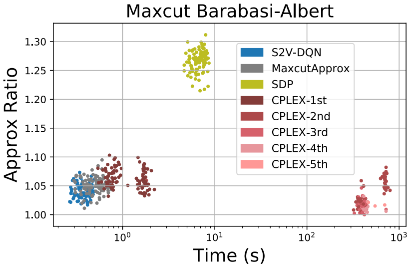
# Minimum vertex cover
# Maximum cut



# Minimum vertex cover
# Maximum cut



# Interesting perspectives

## Vertex embeddings

- if $d(\mu^T(u), \mu^T(v))$ is small, then $u$, $v$ are “similar” in terms of TSP solving
- $\mu_v^T$ capture the properties of $v$ and its neighborhood that are relevant to solve a TSP
  - Does this similarity measure corresponds to any natural property? (e.g. in a social network?)
  - Does it correspond to any known graph property?
  - What about if we optimize $\mu_v$ for graph coloring, maxcut ...?

# Interesting perspectives

## Vertex embeddings

- if $d(\mu^T(u), \mu^T(v))$ is small, then $u$, $v$ are "similar" in terms of TSP solving
- $\mu_v^T$ capture the properties of $v$ and its neighborhood that are relevant to solve a TSP
  - Does this similarity measure corresponds to any natural property? (e.g. in a social network?)
  - Does it correspond to any known graph property?
  - What about if we optimize $\mu_v$ for graph coloring, maxcut ...?

## Discovering new algorithms

- Dai et al. were able to discover a new algorithm for MAXCUT that hasn't been analyzed before

# Interesting perspectives

## Vertex embeddings

- if $d(\mu^T(u), \mu^T(v))$ is small, then $u$, $v$ are "similar" in terms of TSP solving
- $\mu_v^T$ capture the properties of $v$ and its neighborhood that are relevant to solve a TSP
  - Does this similarity measure corresponds to any natural property? (e.g. in a social network?)
  - Does it correspond to any known graph property?
  - What about if we optimize $\mu_v$ for graph coloring, maxcut ...?

## Discovering new algorithms

- Dai et al. were able to discover a new algorithm for MAXCUT that hasn't been analyzed before

## CSP solving

▶ some problems need both learning and reasoning (e.g. CPC)

- How to combine constraint propagation and learning?

Thanks