Graph Neural Networks for Data Management

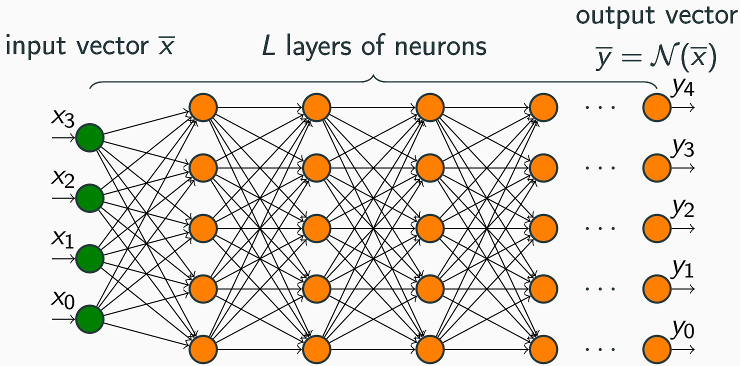
IN3020/4020 Database Systems

Egor V. Kostylev

April 30, 2021

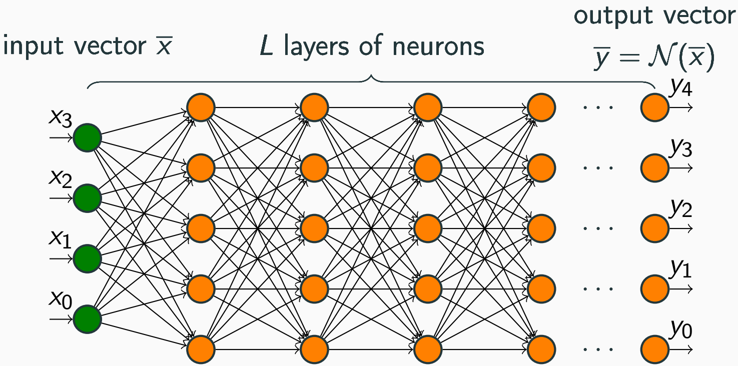
University of Oslo

Fully Connected Feed-Forward Neural Networks (NNs)



A fully connected neural network $\mathcal{N}$

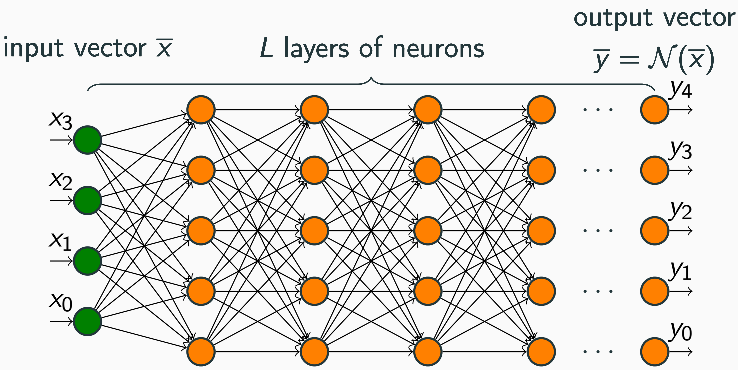
Fully Connected Feed-Forward Neural Networks (NNs)



A fully connected neural network $\mathcal{N}$

- Weight $w_{n' \to n}$ between two consecutive neurons

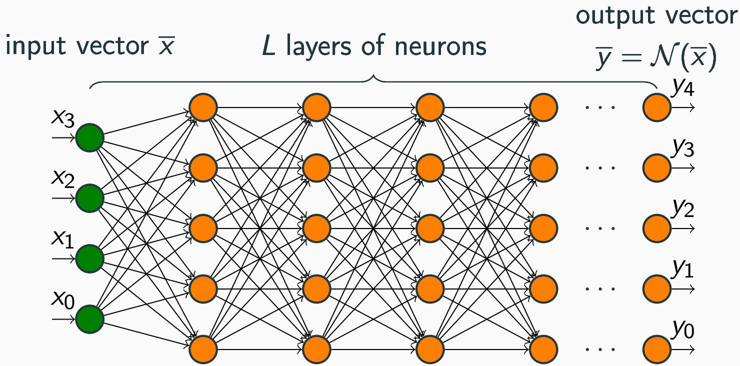
Fully Connected Feed-Forward Neural Networks (NNs)



A fully connected neural network $\mathcal{N}$

- Weight $w_{n' \to n}$ between two consecutive neurons
- Compute left to right $\lambda(n) := f(\sum w_{n' \to n} \times \lambda(n'))$

Fully Connected Feed-Forward Neural Networks (NNs)



A fully connected neural network $\mathcal{N}$

- Weight $w_{n' \to n}$ between two consecutive neurons
- Compute left to right $\lambda(n) := f(\sum w_{n' \to n} \times \lambda(n'))$
- Goal: find the weights that "solve" your problem (classification, clustering, regression, etc.)

Finding the weights

- Goal: find the weights that "solve" your problem

$\rightarrow$ minimize $\mathrm{Dist}(\mathcal{N}(\overline{x}), g(\overline{x}))$, where g is what you want to learn

Finding the weights

- Goal: find the weights that "solve" your problem

$\rightarrow$ minimize $\mathrm{Dist}(\mathcal{N}(\overline{x}), g(\overline{x}))$, where g is what you want to learn

$\rightarrow$ use backpropagation algorithms

Finding the weights

- Goal: find the weights that "solve" your problem

$\rightarrow$ minimize $\mathrm{Dist}(\mathcal{N}(\overline{x}), g(\overline{x}))$, where g is what you want to learn

$\rightarrow$ use backpropagation algorithms

- **Problem**: for fully connected NNs, when a layer has many neurons there are a lot of weights...

Finding the weights

- Goal: find the weights that "solve" your problem

$\rightarrow$ minimize $\mathrm{Dist}(\mathcal{N}(\overline{x}), g(\overline{x}))$, where g is what you want to learn

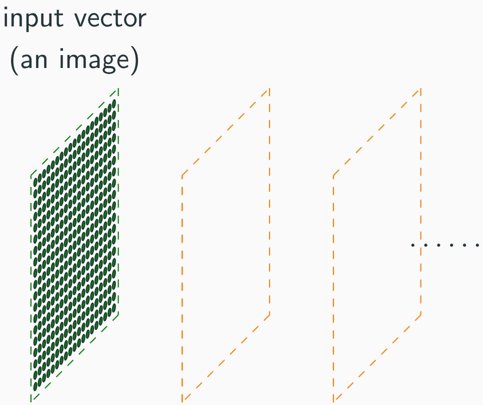
$\rightarrow$ use backpropagation algorithms

- **Problem**: for fully connected NNs, when a layer has many neurons there are a lot of weights. . .

$\rightarrow$ example: input is a 250×250 pixels image, and we want to build a fully connected NN with 500 neurons per layer

$\rightarrow$ between the first two layers we have $250 \times 250 \times 500 = 31,250,000$ weights

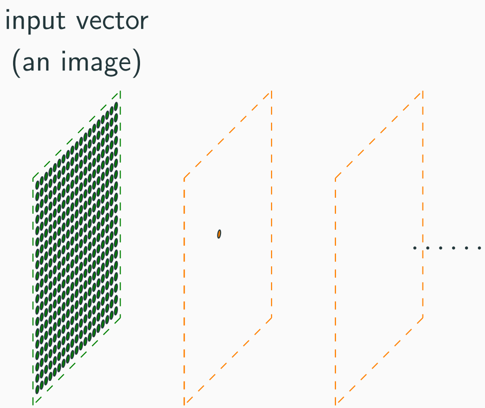
Convolutional Neural Networks



A convolutional neural network

- **Idea**: use the structure of the data (here, a grid)

Convolutional Neural Networks

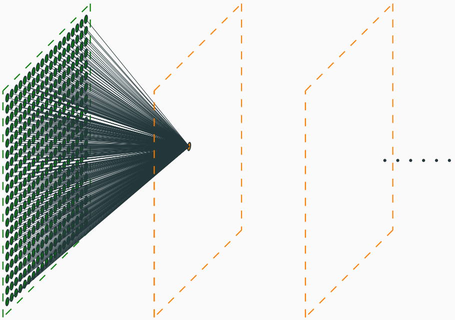


A convolutional neural network

- **Idea**: use the structure of the data (here, a grid)

Convolutional Neural Networks

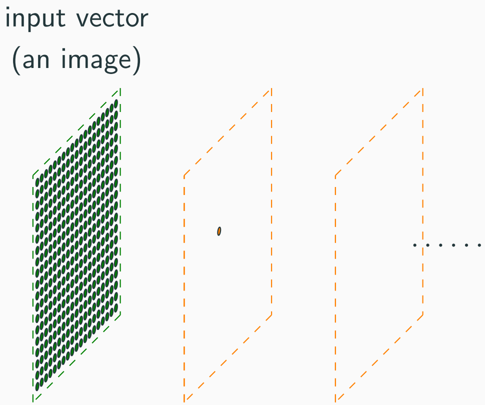
input vector
(an image)



A convolutional neural network

- **Idea**: use the structure of the data (here, a grid)

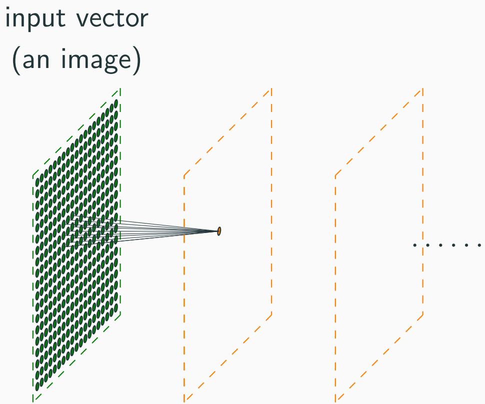
Convolutional Neural Networks



A convolutional neural network

- **Idea**: use the structure of the data (here, a grid)

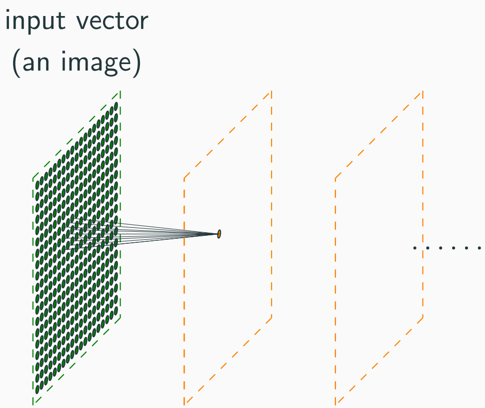
Convolutional Neural Networks



A convolutional neural network

- **Idea**: use the structure of the data (here, a grid)

Convolutional Neural Networks

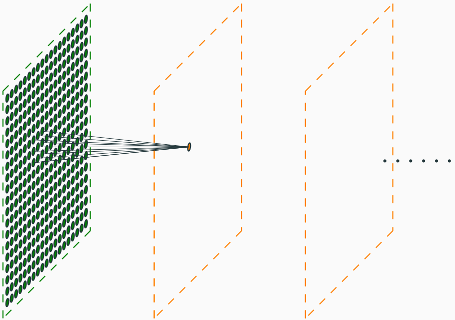


A convolutional neural network

- **Idea**: use the structure of the data (here, a grid)

$\rightarrow$ fewer weights to learn (e.g, $500 * 9 = 4,500$ for the first layer)

Convolutional Neural Networks



A convolutional neural network

- **Idea**: use the structure of the data (here, a grid)

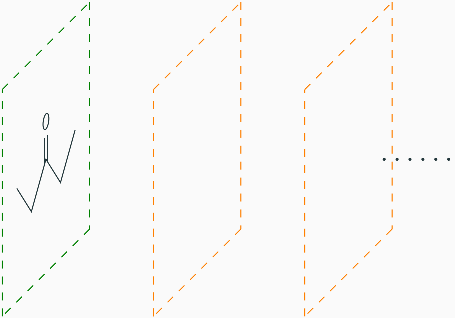
$\rightarrow$ fewer weights to learn (e.g, $500 * 9 = 4,500$ for the first layer)

$\rightarrow$ other advantage: recognize patterns that are local

Graph Neural Networks (GNNs)

input vector
(a molecule)

output:
is it poisonous? (e.g., [?])

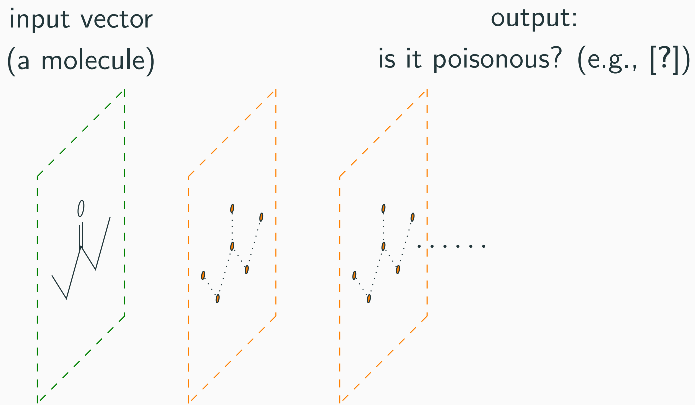


A (convolutional) graph neural network

- **Idea**: use the structure of the data

$\rightarrow$ GNNs generalize this idea to allow *any* graph as input

Graph Neural Networks (GNNs)



A (convolutional) graph neural network

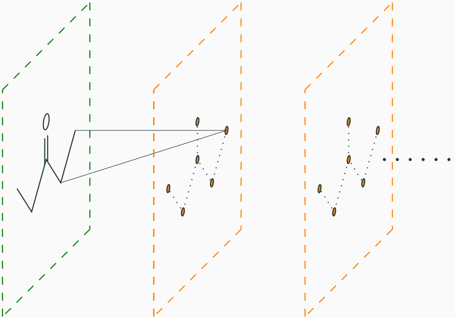
- **Idea**: use the structure of the data

$\rightarrow$ GNNs generalize this idea to allow *any* graph as input

Graph Neural Networks (GNNs)

input vector (a molecule)

output: is it poisonous? (e.g., [?])

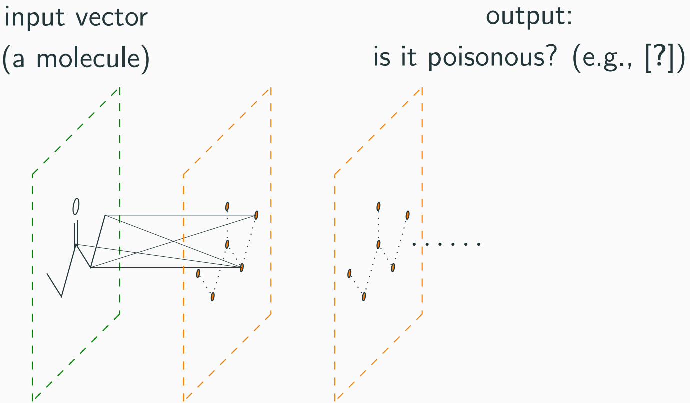


A (convolutional) graph neural network

- **Idea**: use the structure of the data

$\rightarrow$ GNNs generalize this idea to allow *any* graph as input

Graph Neural Networks (GNNs)



A (convolutional) graph neural network

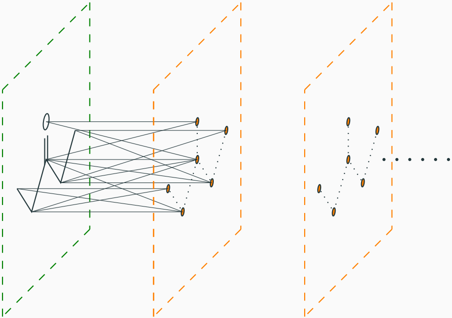
- **Idea**: use the structure of the data

$\rightarrow$ GNNs generalize this idea to allow *any* graph as input

Graph Neural Networks (GNNs)

input vector
(a molecule)

output:
is it poisonous? (e.g., [?])



A (convolutional) graph neural network

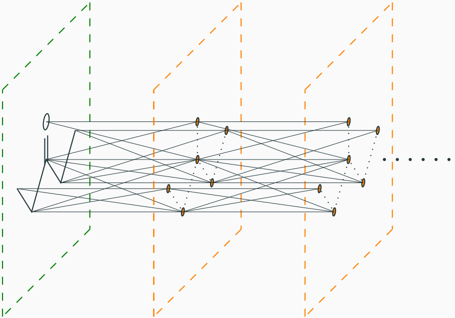
- **Idea**: use the structure of the data

→ GNNs generalize this idea to allow *any* graph as input

Graph Neural Networks (GNNs)

input vector
(a molecule)

output:
is it poisonous? (e.g., [?])



A (convolutional) graph neural network

- **Idea**: use the structure of the data

$\rightarrow$ GNNs generalize this idea to allow *any* graph as input

Basic GNNs Formalisation

- Simple, undirected, node-labeled graph $G = (V, E, \lambda)$, where $\lambda : V \to \mathbb{R}^d$

Basic GNNs Formalisation

- Simple, undirected, node-labeled graph $G = (V, E, \lambda)$, where $\lambda : V \to \mathbb{R}^d$
- Run of a GNN with L layers on G: iteratively compute $\boldsymbol{x}_u^{(i)} \in \mathbb{R}^d$ for $0 \leq i \leq L$ as

Basic GNNs Formalisation

- Simple, undirected, node-labeled graph $G = (V, E, \lambda)$, where $\lambda : V \to \mathbb{R}^d$
- Run of a GNN with L layers on G: iteratively compute $\boldsymbol{x}_u^{(i)} \in \mathbb{R}^d$ for $0 \leq i \leq L$ as

$\rightarrow$ $\boldsymbol{x}_u^{(0)} := \lambda(u)$

Basic GNNs Formalisation

- Simple, undirected, node-labeled graph $G = (V, E, \lambda)$, where $\lambda : V \to \mathbb{R}^d$
- Run of a GNN with L layers on G: iteratively compute $\boldsymbol{x}_u^{(i)} \in \mathbb{R}^d$ for $0 \leq i \leq L$ as

$\to$ $\boldsymbol{x}_u^{(0)} := \lambda(u)$

$\to$ $\boldsymbol{x}_u^{(i+1)} := \mathit{ReLU}(\mathbf{A}_i \boldsymbol{x}_u^{(i)} + \mathbf{B}_i \sum_{v \in \mathcal{N}_G(u)} \boldsymbol{x}_v^{(i)} + \mathbf{c}_i)$

- where $\mathbf{A}_i$ and $\mathbf{B}_i$ are trainable matrices and $\mathbf{c}_i$ are such vectors
- ReLU is non-linearity function

Basic GNNs Formalisation

- Simple, undirected, node-labeled graph $G = (V, E, \lambda)$, where $\lambda : V \to \mathbb{R}^d$
- Run of a GNN with L layers on G: iteratively compute $\boldsymbol{x}_u^{(i)} \in \mathbb{R}^d$ for $0 \leq i \leq L$ as

$\rightarrow$ $\boldsymbol{x}_u^{(0)} := \lambda(u)$

$\rightarrow$ $\boldsymbol{x}_u^{(i+1)} := ReLU(\mathbf{A}_i \boldsymbol{x}_u^{(i)} + \mathbf{B}_i \sum_{v \in \mathcal{N}_G(u)} \boldsymbol{x}_v^{(i)} + \mathbf{c}_i)$

- where $\mathbf{A}_i$ and $\mathbf{B}_i$ are trainable matrices and $\mathbf{c}_i$ are such vectors
- ReLU is non-linearity function

- Generalisation to ordered graphs with different types of edges is straightforward
- Knowledge graphs (RDF, Neo4j)?

Question:
Can we make use of GNNs in data management?

One problem: Supervised learning queries from examples

- Input:
  - set of positive examples (G, v) (graph-node pairs)
  - set of negative examples (G, v) (graph-node pairs)
- Question:
  - give me a simple query (SPARQL, Cypher) that mostly give these answers

One problem: Supervised learning queries from examples

- Input:
  - set of positive examples (G, v) (graph-node pairs)
  - set of negative examples (G, v) (graph-node pairs)
- Question:
  - give me a simple query (SPARQL, Cypher) that mostly give these answers
- Mostly, because input may be noisy
- Query (not a NN or other algorithm), because we want efficient evaluation on big graphs
- Simple, because we need something understandable (explainable!)

What can we use for query learning?

- Logic-based methods? (Symbolic AI)

What can we use for query learning?

- Logic-based methods? (Symbolic AI)
  - high complexity
  - unclear what to do with noise

What can we use for query learning?

- Logic-based methods? (Symbolic AI)
    - high complexity
    - unclear what to do with noise
- Standard feed-forward NNs?

What can we use for query learning?

- Logic-based methods? (Symbolic AI)
    - high complexity
    - unclear what to do with noise
- Standard feed-forward NNs?
    - assume input of fixed size, but KGs may be arbitrarily big
    - unclear how to extract a query from a trained NN

What can we use for query learning?

- Logic-based methods? (Symbolic AI)
  - high complexity
  - unclear what to do with noise
- Standard feed-forward NNs?
  - assume input of fixed size, but KGs may be arbitrarily big
  - unclear how to extract a query from a trained NN
- RNNs? (Recurrent NNs, used a lot for NLP)

What can we use for query learning?

- Logic-based methods? (Symbolic AI)
  - high complexity
  - unclear what to do with noise
- Standard feed-forward NNs?
  - assume input of fixed size, but KGs may be arbitrarily big
  - unclear how to extract a query from a trained NN
- RNNs? (Recurrent NNs, used a lot for NLP)
  - may be applied to some encodings of KGs as strings, but will depend of the exact encoding
  - "destroy" the structure
  - unclear how to extract a query from a trained RNN

What can we use for query learning?

- Logic-based methods? (Symbolic AI)
  - high complexity
  - unclear what to do with noise
- Standard feed-forward NNs?
  - assume input of fixed size, but KGs may be arbitrarily big
  - unclear how to extract a query from a trained NN
- RNNs? (Recurrent NNs, used a lot for NLP)
  - may be applied to some encodings of KGs as strings, but will depend of the exact encoding
  - "destroy" the structure
  - unclear how to extract a query from a trained RNN

GNNs for query learning

- GNNs have potential to overcome these problems:
    - can process inputs of various size
    - work directly on graph structures
    - noise-tolerant

GNNs for query learning

- GNNs have potential to overcome these problems:
  - can process inputs of various size
  - work directly on graph structures
  - noise-tolerant
- Question: How to translate a (trained) GNN to a (SPARQL) query?
  - much less hopeless than for usual NNs
  - GNNs "keep" a lot of structure
  - some GNN architectures may be easier to translate than others

GNNs for data and knowledge management

- There are two paradigms for data and knowledge management
  - Symbolic (logic-based) methods (SQL, SPARQL, OWL)
  - Syb-Symbolic (statistic-based) methods (NNs)
- Both have strengths and weaknesses
- A major challenge is to deeply integrate these paradigms

GNNs for data and knowledge management

- There are two paradigms for data and knowledge management
  - Symbolic (logic-based) methods (SQL, SPARQL, OWL)
  - Syb-Symbolic (statistic-based) methods (NNs)
- Both have strengths and weaknesses
- A major challenge is to deeply integrate these paradigms

- I believe that GNNs can be a bridge between these paradigms
  - work on structured data of varying size
  - have a neural capabilities

GNNs for data and knowledge management

- There are two paradigms for data and knowledge management
  - Symbolic (logic-based) methods (SQL, SPARQL, OWL)
  - Syb-Symbolic (statistic-based) methods (NNs)
- Both have strengths and weaknesses
- A major challenge is to deeply integrate these paradigms

- I believe that GNNs can be a bridge between these paradigms
  - work on structured data of varying size
  - have a neural capabilities
- Can be potentially used to solve many problems as above:
  - incorporating logical knowledge into network-based models
  - knowledge graph completion
  - query answering over incomplete knowledge graphs
  - ontology learning
  - etc.