Neural networks and structures

Neuro-symbolic modeling

THE UNIVERSITY OF UTAH

Lecture outline

- Neural networks & structured outputs
- Examples of structured outputs
- Computational questions with structured prediction
- Inference

Lecture outline

- Neural networks & structured outputs
- Examples of structured outputs
- Computational questions with structured prediction
- Inference

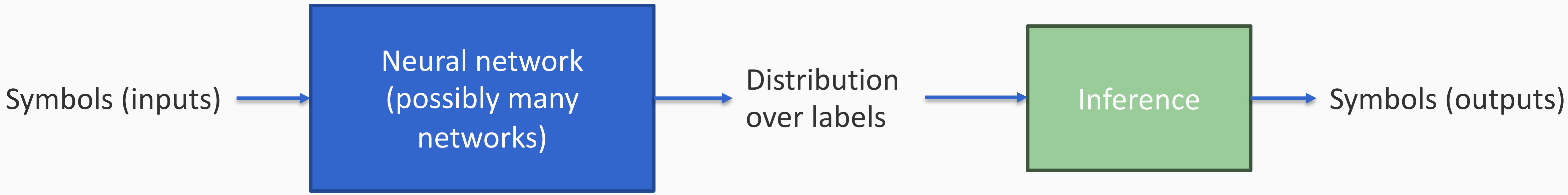
Neural network + symbolic programs

Neural networks involve vectors and differentiable functions

Rules are symbolic objects (i.e. consisting of discrete decisions)

How do they interface with each other?

Neuro-symbolic interfaces



Convert symbolic inputs into vectors, operate with distributions over labels

Apply probabilistic inference (with symbolic knowledge) over the top of network outputs to produce final outputs: *Structured prediction*

Structured inference

The neural network produces probabilities over its output space. Suppose our rules are constraints about the outputs

Structured inference

The neural network produces probabilities over its output space. Suppose our rules are constraints about the outputs

If our constraints are purely about outputs, then we could set up a constrained optimization problem whose solution is guaranteed to satisfy the constraints

Structured inference

The neural network produces probabilities over its output space. Suppose our rules are constraints about the outputs

If our constraints are purely about outputs, then we could set up a constrained optimization problem whose solution is guaranteed to satisfy the constraints

The neural network creates the optimization problem for an input instance, the symbolic solver solves it

Structured inference

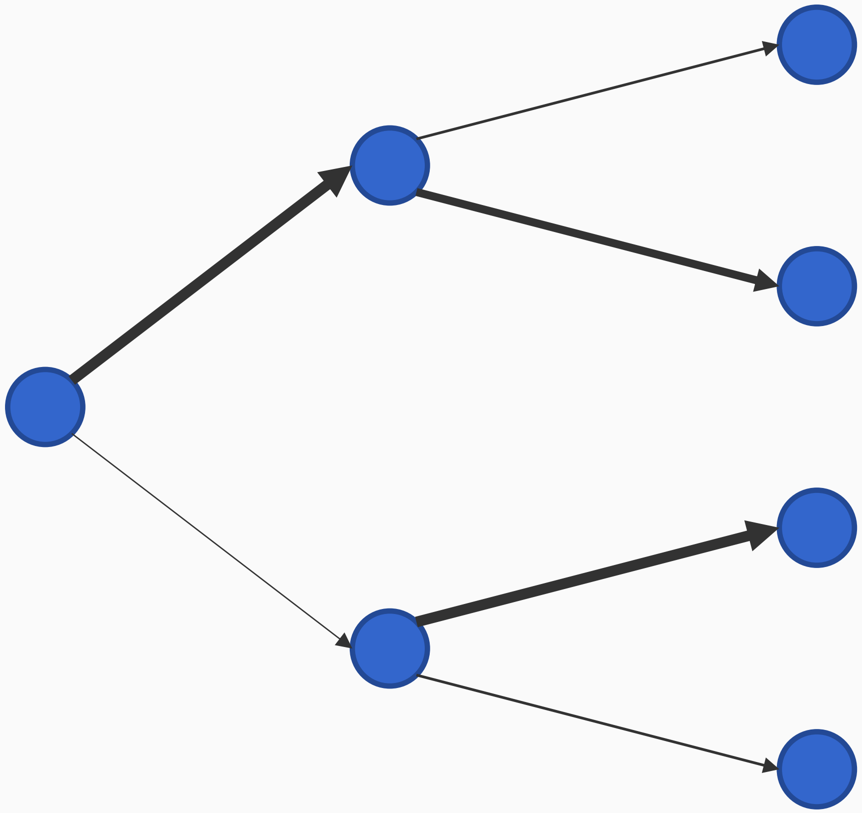
The neural network produces probabilities over its output space. Suppose our rules are constraints about the outputs

If our constraints are purely about outputs, then we could set up a constrained optimization problem whose solution is guaranteed to satisfy the constraints

The neural network creates the optimization problem for an input instance, the symbolic solver solves it

This could provide supervision at training time and/or can be used only at deployment

Neuro-symbolic interfaces



Use a neural network to guide a program that navigates a symbolic space by scoring paths (e.g. game playing, combinatorial inference, etc)

A variant: Neural network guided symbolic problem solver

The neural network produces probabilities over its output space

The network probabilities could guide a search system that explores a richer (compositional) output space

The output space may involve symbolic constraints that the search system will incorporate

Lecture outline

- Neural networks & structured outputs
- Examples of structured outputs
- Computational questions with structured prediction
- Inference

The simplest example: Multiclass classification

Collect a training set (hopefully with correct labels)

Define feature representations for inputs ($\boldsymbol{x} \in \Re^n$)

And, maybe also the labels $\mathbf{y} \in \{book, \ \ dog, \ \ penguin\}$

Train models functions to score labels

$$\underset{\mathbf{y} \in \{book, dog, penguin\}}{\operatorname{argmax}} \ \mathrm{score}(\mathbf{x}, \mathbf{y})$$

Recipe for multiclass classification

Train weights so that it scores examples correctly

e.g., for an input of type $book$, we want

$$\text{score}(\mathbf{x}, book) > \text{score}(\mathbf{x}, penguin)$$
$$\text{score}(\mathbf{x}, book) > \text{score}(\mathbf{x}, dog)$$

Prediction: $\underset{\mathbf{y} \in \{book, dog, penguin\}}{\text{argmax}} \ \text{score}(\mathbf{x}, \mathbf{y})$

- Easy to predict
- Iterate over the output list, find the highest scoring one

Recipe for multiclass classification

Train weights so that it scores examples correctly

e.g., for an input of type *book*, we want

$$\text{score}(\mathbf{x}, book) > \text{score}(\mathbf{x}, penguin)$$
$$\text{score}(\mathbf{x}, book) > \text{score}(\mathbf{x}, dog)$$

The cross-entropy loss takes care of this. Why?

Prediction: $$\underset{\mathbf{y} \in \{book, dog, penguin\}}{\text{argmax}} \ \text{score}(\mathbf{x}, \mathbf{y})$$

– Easy to predict
– Iterate over the output list, find the highest scoring one

Recipe for multiclass classification

Train weights so that it scores examples correctly

e.g., for an input of type *book*, we want

$$\mathrm{score}(\mathbf{x}, book) > \mathrm{score}(\mathbf{x}, penguin)$$
$$\mathrm{score}(\mathbf{x}, book) > \mathrm{score}(\mathbf{x}, dog)$$

The cross-entropy loss takes care of this. Why?

Prediction: $$\operatorname*{argmax}_{\mathbf{y} \in \{book, dog, penguin\}} \mathrm{score}(\mathbf{x}, \mathbf{y})$$

- Easy to predict
- Iterate over the output list, find the highest scoring one

What if the space of outputs is much larger?
Say trees, or in general, graphs. Let's look at examples.

Example 2: Information extraction

Predicting entities and relations from text

$\boxed{\text{Colin}}$ went back home to $\boxed{\text{Ordon Village}}$.

Entities can be `Person`, `Location`, `Organization`, `None`

Directed edges between them can be `Kill`, `LiveIn`, `WorkFor`, `LocatedAt`, `OrgBasedIn`, `None`

Example 2: Information extraction

Predicting entities and relations from text

$\boxed{\text{Colin}}$ went back home to $\boxed{\text{Ordon Village}}$.

Colin is a `Person`

Ordon Village is a `Location`

Colin → OrdonVillage: `LiveIn`

Ordon Village → Colin `None`

Prediction: a structure

Structured prediction…

… requires an exploration of the combinatorial space of possible outputs to find the best one

Colin is a Location
Ordon Village is a Organization
Colin → OrdonVillage: LiveIn
Ordon Village → Colin LiveIn

Colin is a Person
Ordon Village is a Organization
Colin → OrdonVillage: WorkFor
Ordon Village → Colin LiveIn

Colin is a Person
Ordon Village is a Location
Colin → OrdonVillage: LiveIn
Ordon Village → Colin None

Colin is a Organization
Ordon Village is a Person
Colin → OrdonVillage: Kill
Ordon Village → Colin WorkFor

Colin is a Person

Colin is a Person

Structured prediction…

… requires an exploration of the combinatorial space of possible outputs to find the best one

Colin is a `Location`
Ordon Village is a `Organization`
Colin → OrdonVillage: `LiveIn`
Ordo

Colin is a `Person`
Ordon Village is a `Organization`
Colin → OrdonVillage: `WorkFor`
n

Modern neural models (e.g. LLMs) seem to simulate this process very well.

Yet, it is not too hard to find instances where their outputs violate domain constraints, especially for large problems

Colin
Ordo
Colin → OrdonVillage: `LiveIn`
Ordon Village → Colin `None`

Colin → OrdonVillage: `Kill`
Ordon Village → Colin `WorkFor`

Colin is a `Person`

Colin is a `Person`

Example 3: Object detection

Photo by Andrew Dressel - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0

Example 3: Object detection

Photo by Andrew Dressel - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0

Example 3: Object detection

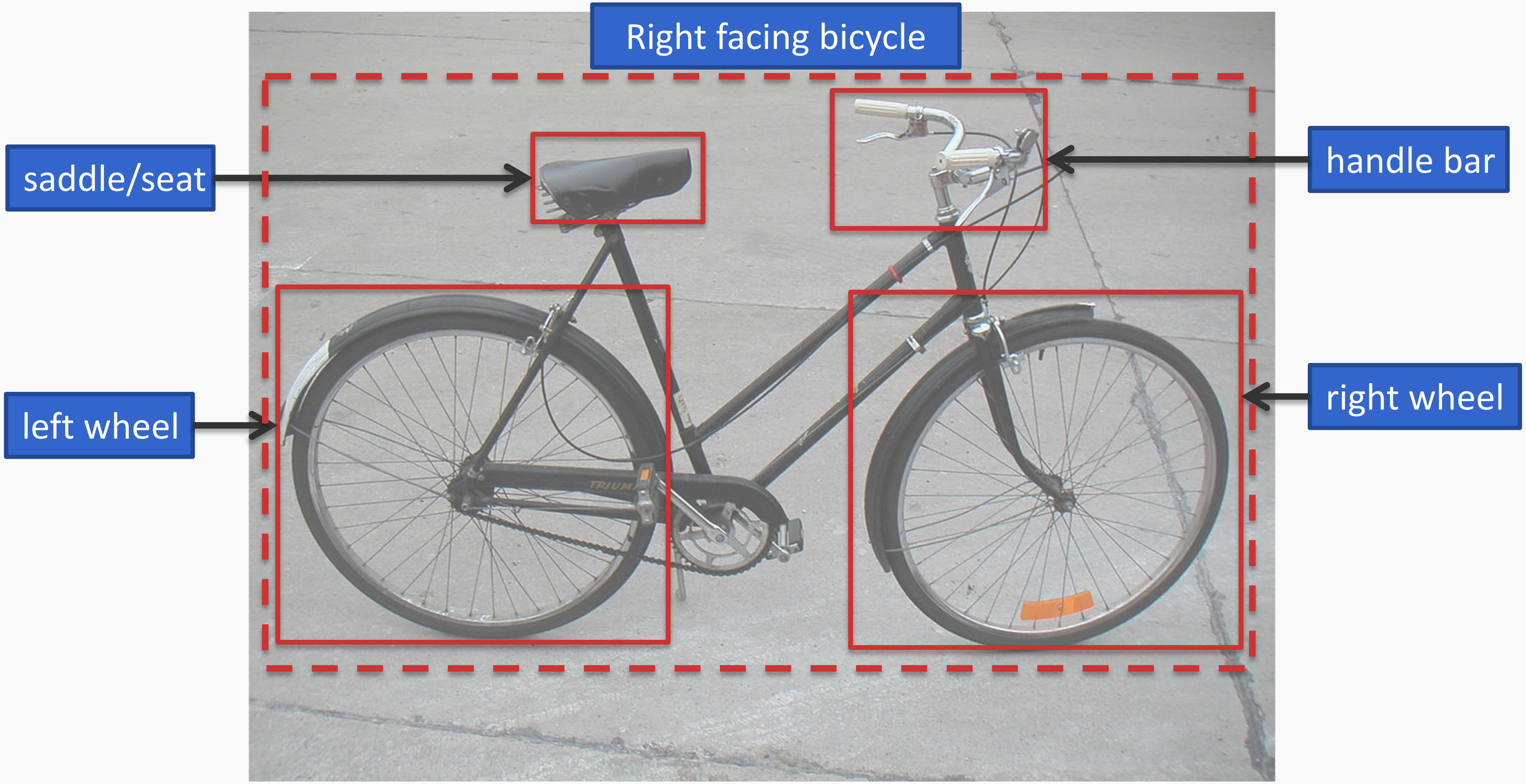
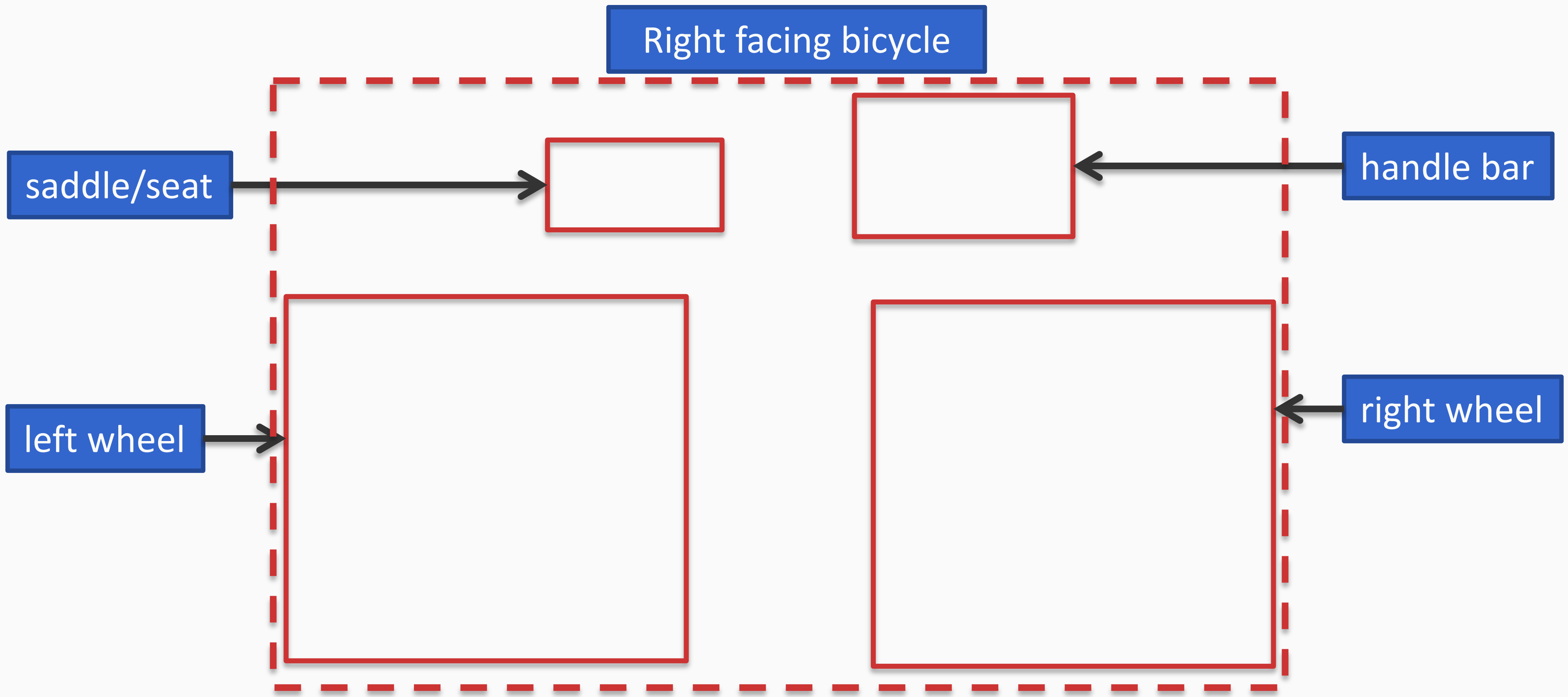


Photo by Andrew Dressel - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0

The output: A schematic showing the parts and their relative layout



Once again, a structure

Example 3: Object detection

How would you design a predictor that labels all the parts using the tools we have seen so far?

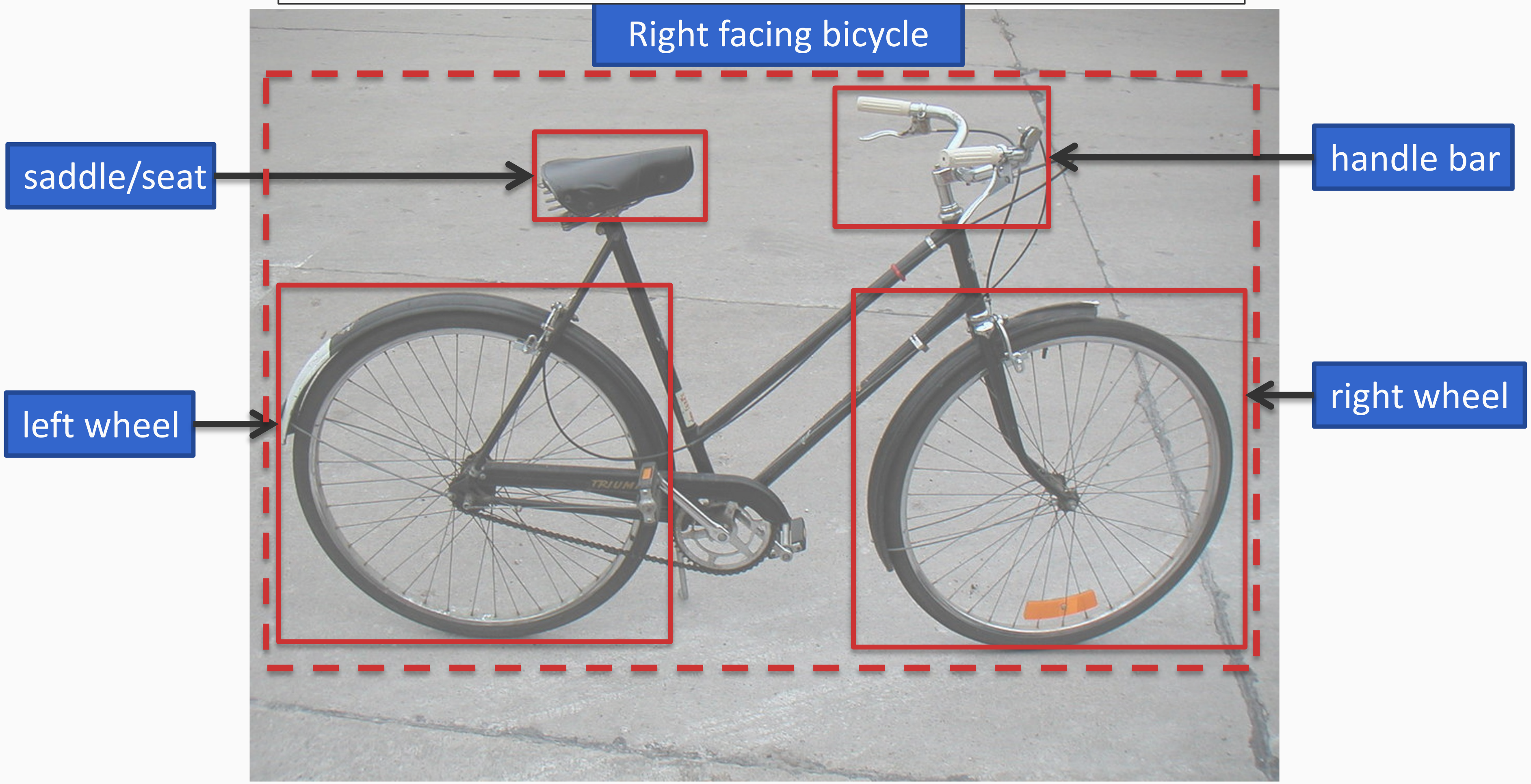


Photo by Andrew Dressel - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0

One approach to build this structure

Left wheel detector: Is there a wheel in this box? Binary classifier

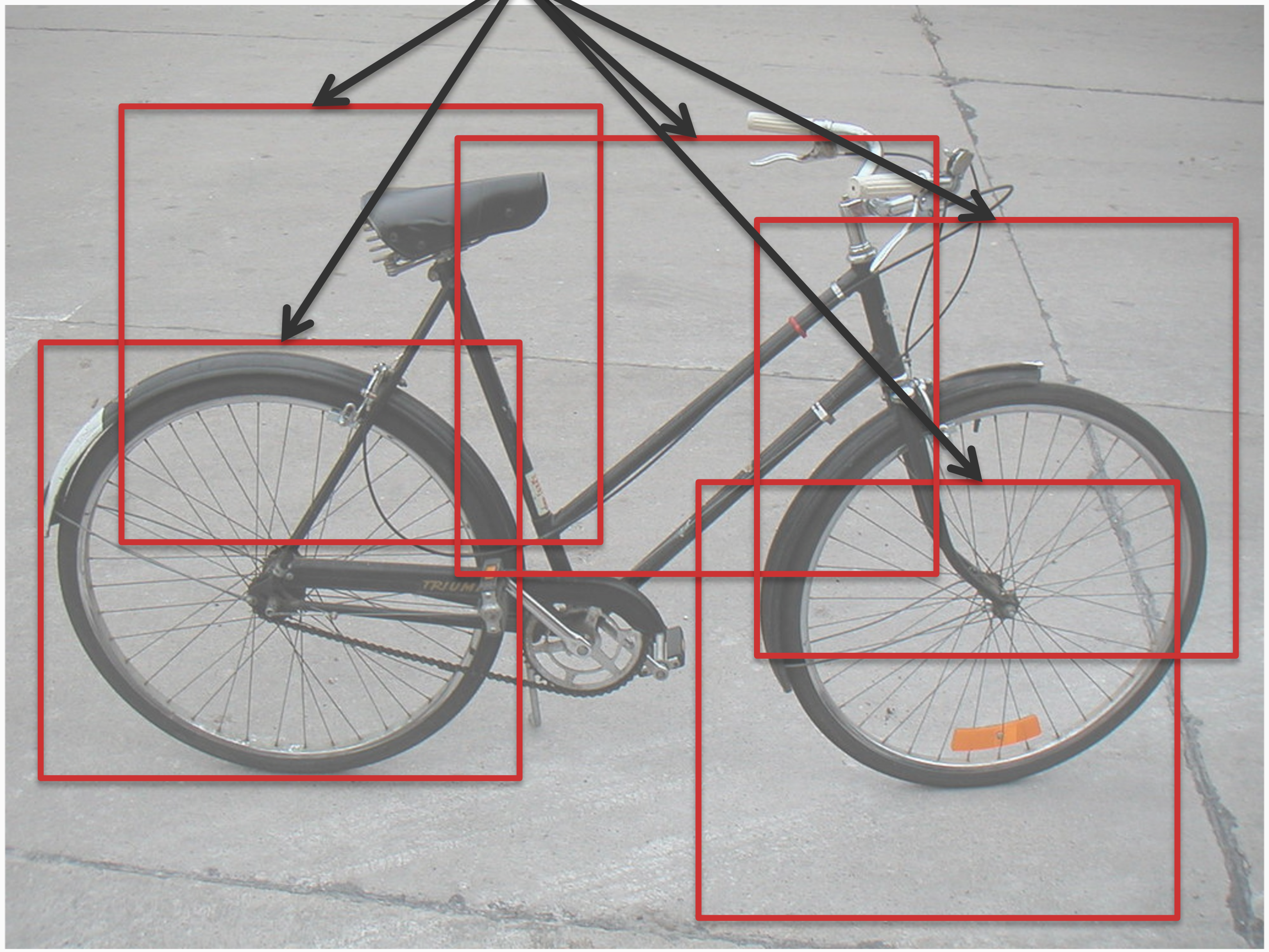


Photo by Andrew Dressel - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0

One approach to build this structure

Handle bar detector: Is there a handle bar in this box? Binary classifier

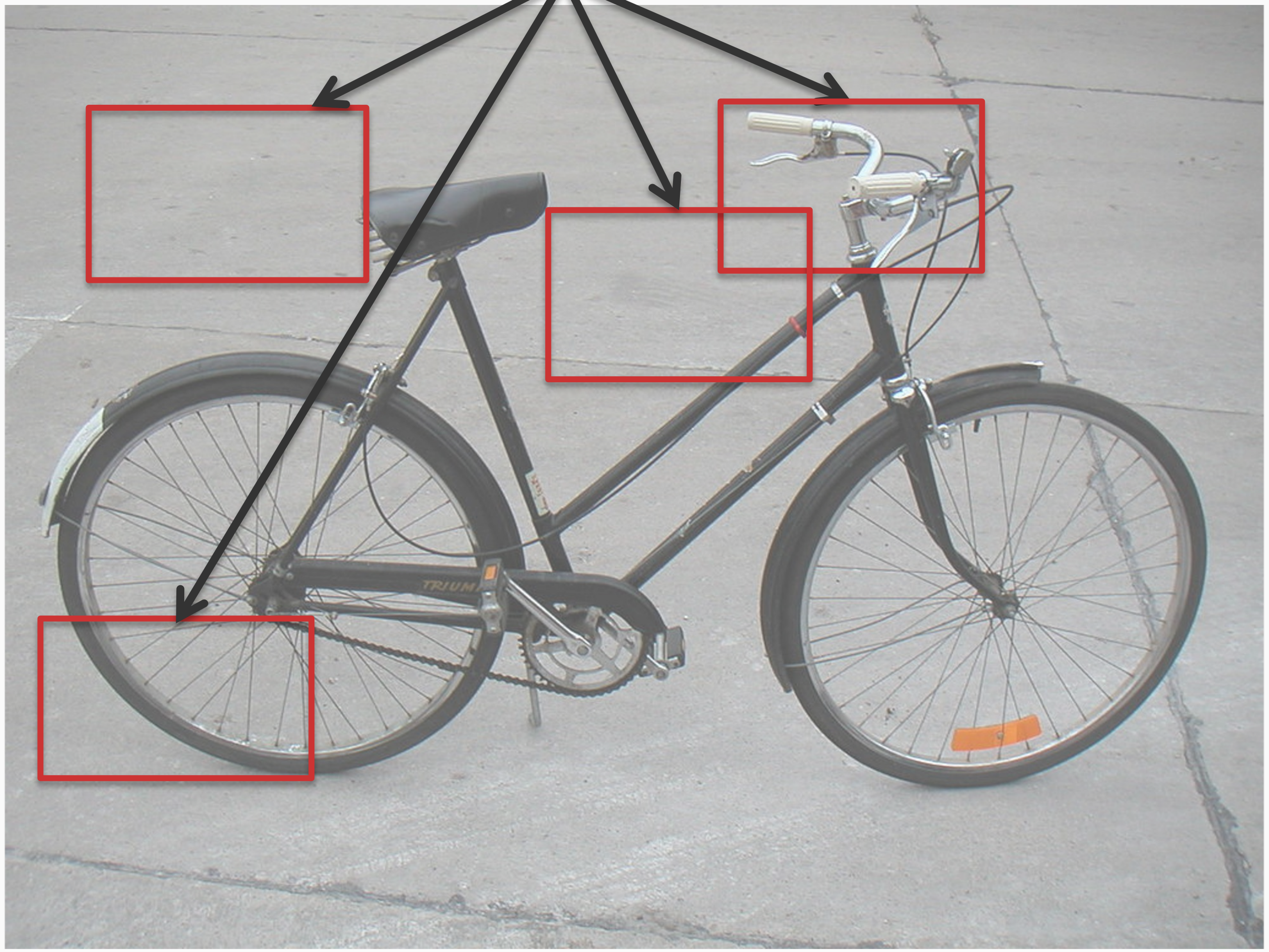


Photo by Andrew Dressel - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0

One approach to build this structure

1. Left wheel detector

2. Right wheel detector

3. Handle bar detector

4. Seat detector

Photo by Andrew Dressel - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0

One approach to build this structure

1. Left wheel detector

2. Right wheel detector

3. Handle bar detector

4. Seat detector

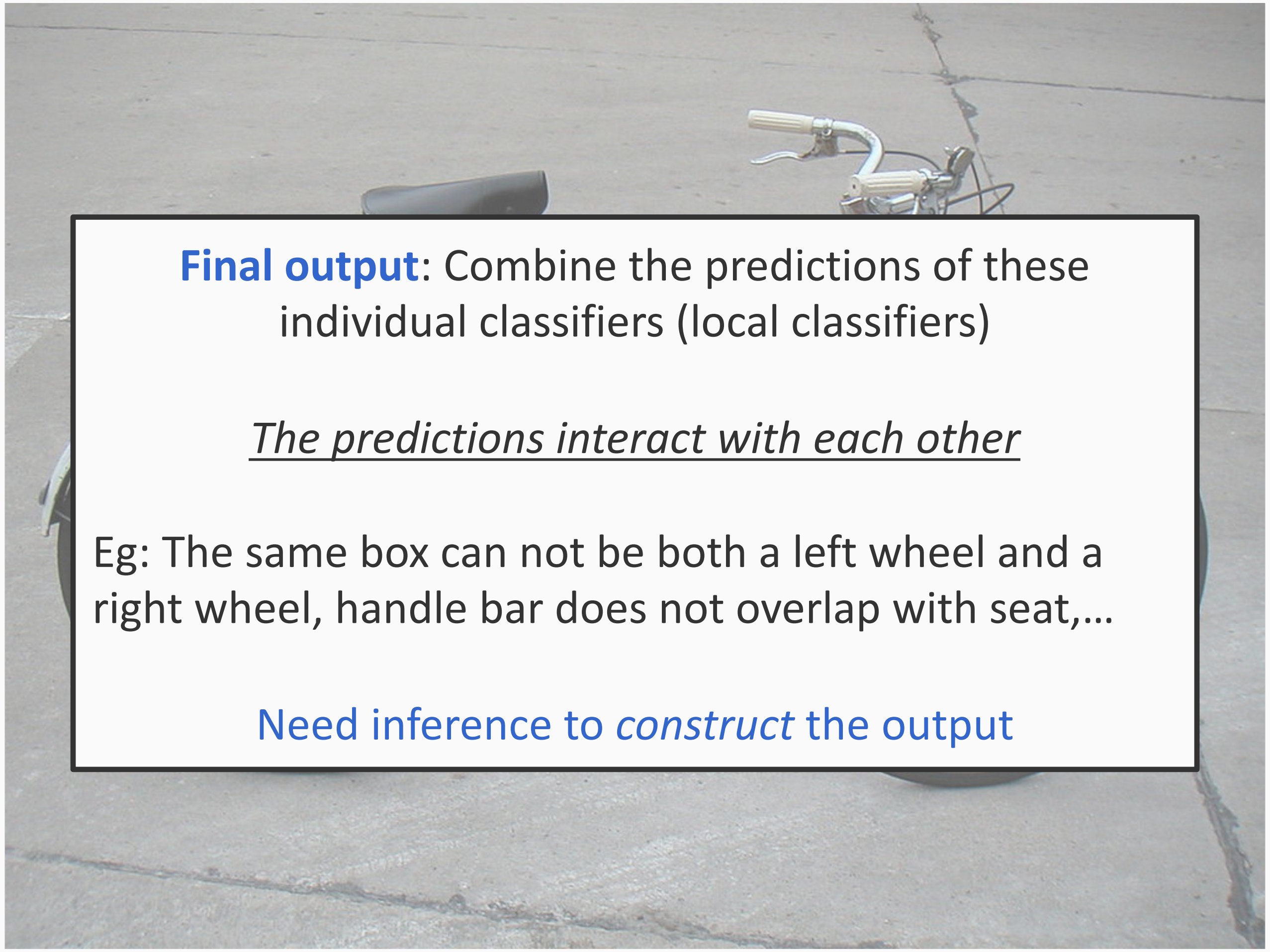


Photo by Andrew Dressel - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0

More examples

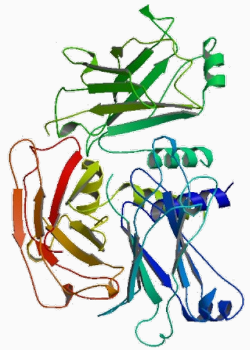
Protein 3D structure prediction

```
AMADLEQKVL EMEASTYDGV FIWKISDFAR KRQEAVAGRI PAIFSPAFYT
 HHHHHHHHH HHHH  SSSE EEEEEESHHH HHHHHHTTSS  EEE   EES

SRYGYKMCLR IYLNGDGTGR GTHLSLFFVV MKGPNDALLR WPFNQKVTLM
STTS EEEEE EETT  GGGT TTEEEEEEEE E  TTGGGS  SS    EEEE

LLDQNNREHV IDAFRPDVTS SSFQRPVNDM NIASGCPLFC PVSKMEAKNS
E  TTSS  E EEEE   TTS GGGS  SSSB    EEEEEEE ETTTTTTSTT

YVRDDAIFIK AIVDLTGL
T  SSEEEEE EEE  TT
```



Inferring layout of a room

Image from [Schwing et al 2013]

Structured output is...

- A graph, possibly labeled and/or directed
  - Possibly from a restricted family, such as chains, trees, etc.
  - A discrete representation of input
  - Eg. A table, the information extraction output, a sequence of labels etc

- A collection of inter-dependent decisions
  - Eg: The sequence of decisions used to construct the output

- The result of a combinatorial optimization problem

$$\underset{\mathbf{y} \in \, all\ outputs}{\operatorname{argmax}} \; score(\mathbf{x}, \mathbf{y})$$

Structured output is…

- A graph, possibly labeled and/or directed
  - Possibly from a restricted family, such as chains, trees, etc.
  - A discrete representation of input
  - Eg. A table, the information extraction output, a sequence of labels etc

Representation

- A collection of inter-dependent decisions
  - Eg: The sequence of decisions used to construct the output

Procedural

- The result of a combinatorial optimization problem

$$\underset{\mathbf{y} \in \, all\ outputs}{\operatorname{argmax}} \; score(\mathbf{x}, \mathbf{y})$$

We have seen something similar before in the context of multiclass

Structured output is…

- A graph, possibly labeled and/or directed
  - Possibly from a restricted family, such as chains, trees, etc.
  - A discrete representation of input
  - Eg. A table, the information extraction output, a sequence of labels etc

Representation

- A collection of inter-dependent decisions
  - Eg: The sequence of decisions used to construct the output

Procedural

- The result of a combinatorial optimization problem

$$\underset{\mathbf{y} \in \, all\ outputs}{\operatorname{argmax}} \; score(\mathbf{x}, \mathbf{y})$$

We have seen something similar before in the context of multiclass

There are a countable number of graphs
Question: Why can't we treat each output as a label and train/predict as multiclass?

Challenges with structured output

Two challenges

1. We cannot train a separate set of parameters for each possible inference outcome
   - For multiclass, we could train one scoring function for each label
2. We cannot enumerate all possible structures for inference
   - Inference for multiclass was easy

Challenges with structured output

Two challenges

1. We cannot train a separate set of parameters for each possible inference outcome
   - For multiclass, we could train one scoring function for each label
2. We cannot enumerate all possible structures for inference
   - Inference for multiclass was easy

Key point

We have multiple decisions that interact with each other

Each decision may be the result of a neural network…

…but we can't take the network predictions independently because they may create an invalid output (i.e. violate structural constraints)

Challenges with structured output

Two challenges

1. We cannot train a separate set of parameters for each possible inference outcome
   - For multiclass, we could train one scoring function for each label
2. We cannot enumerate all possible structures for inference
   - Inference for multiclass was easy

Solution

- Decompose the output into parts that are labeled
- Define
  - how the parts interact with each other
  - how these labeled interacting parts are scored
  - an inference algorithm to assign labels to all the parts

Challenges with structured output

Two challenges

1. We cannot train a separate set of parameters for each possible inference outcome
   - For multiclass, we could train one scoring function for each label
2. We cannot enumerate all possible structures for inference
   - Inference for multiclass was easy

Solution

- Decompose the output into parts that are labeled
- Define
  - how the parts interact with each other
  - how these labeled interacting parts are scored
  - an inference algorithm to assign labels to all the parts

Lecture outline

- Neural networks & structured outputs
- Examples of structured outputs
- Computational questions with structured prediction
- Inference

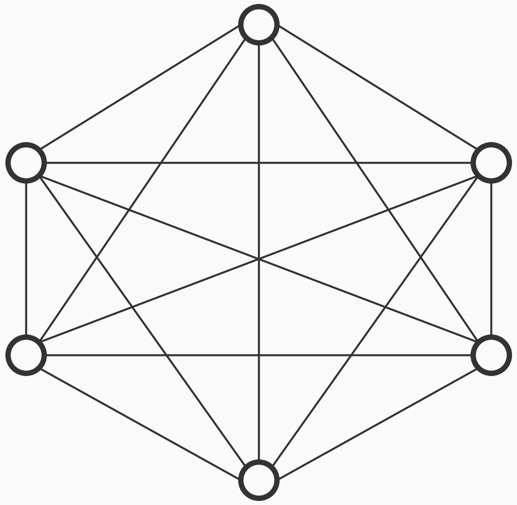
Decomposing the output

- We need to produce a graph
  - We cannot enumerate all possible graphs for the argmax

- Solution: Think of the graph as combination of many smaller parts
  - The parts should agree with each other in the final output
  - Each part has a score
  - The total score for the graph is the sum of scores of each part

- Decomposition of the output into parts also helps generalization
  - Why?

Decomposing the output: Example



3 possible node labels

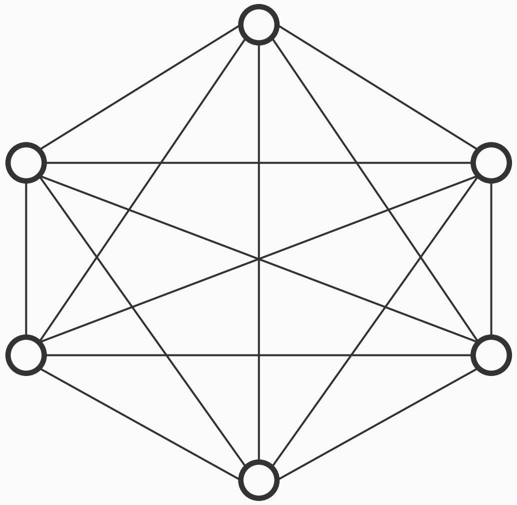
3 possible edge labels

Setting

Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree

Decomposing the output: Example



3 possible node labels

3 possible edge labels

Setting

Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree

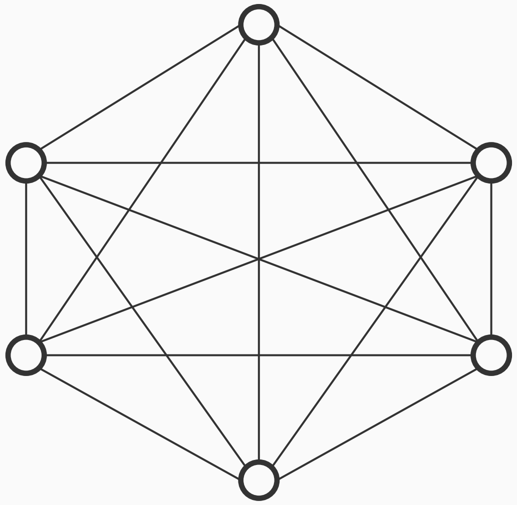
The scoring function scores outputs

For generalization and ease of inference, break the output into parts and score each part

The score for the structure is the sum of the part scores

What is the best way to do this decomposition? Depends….

Decomposing the output: Example



3 possible node labels 

3 possible edge labels 

Setting

Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree

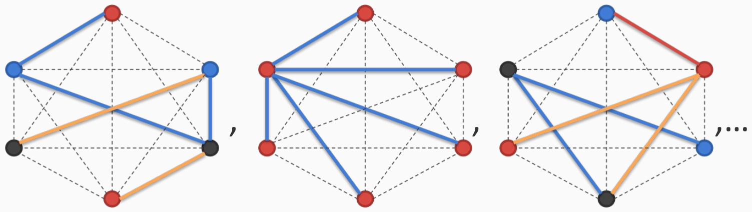
The scoring function scores outputs

For generalization and ease of inference, break the output into parts and score each part

The score for the structure is the sum of the part scores

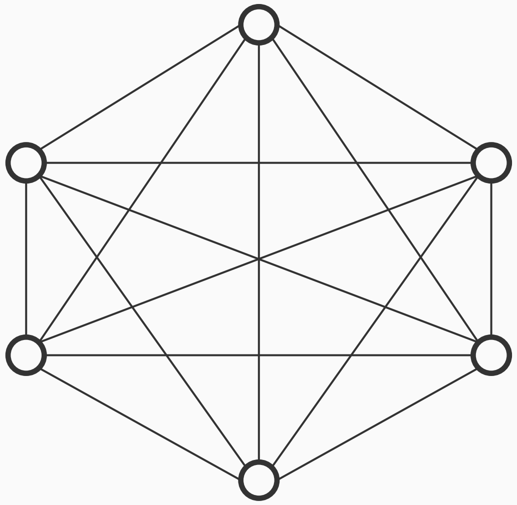
What is the best way to do this decomposition? Depends….

Note: The output **y** is a labeled assignment of the nodes and edges



The input **x** not shown here

Decomposing the output: Example



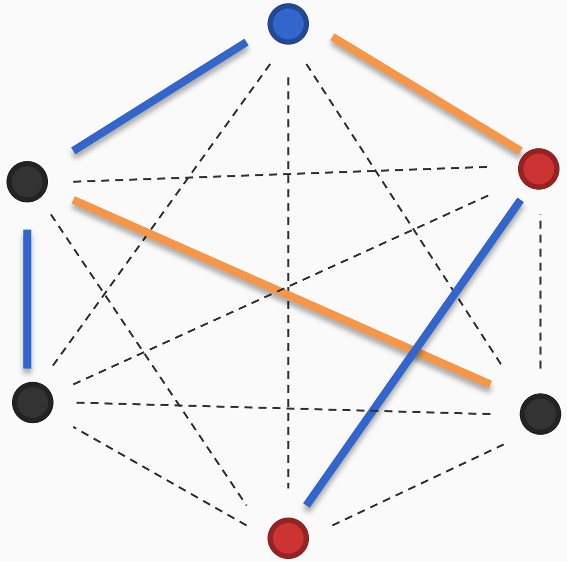
3 possible node labels

3 possible edge labels

Setting
Output: Nodes and edges are labeled and the blue and orange edges form a tree

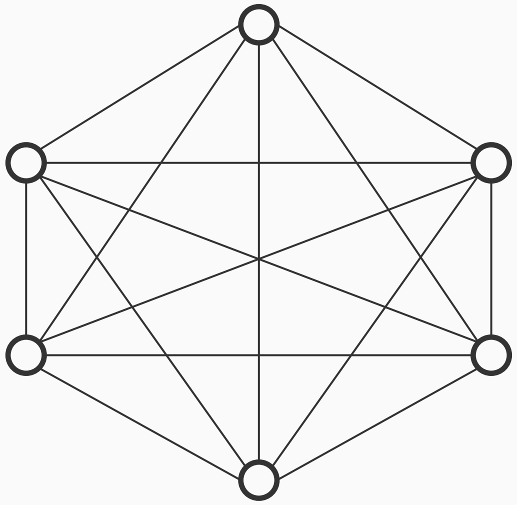
Goal: Find the highest scoring labeling such that the edges that are colored form a tree

One option: Decompose fully. *All nodes and edges are independently scored*



$$\text{score}(\mathbf{x}, \mathbf{y}) = \sum_{n \in \text{ nodes}(\mathbf{x},\mathbf{y})} \text{score}(n) + \sum_{e \in \text{ edges}(\mathbf{x},\mathbf{y})} \text{score}(e)$$

Decomposing the output: Example



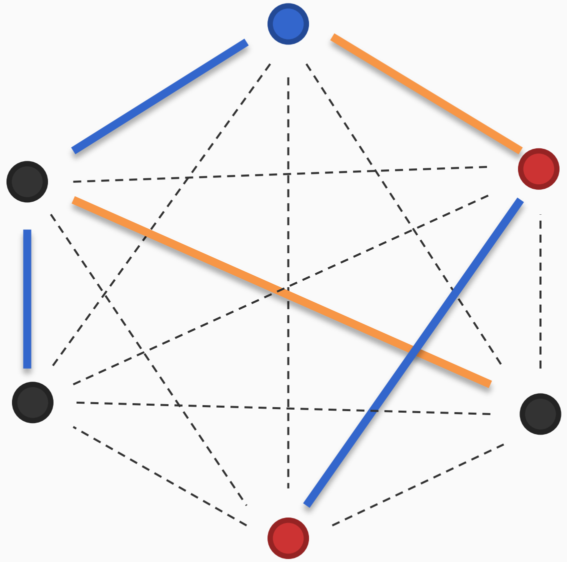
3 possible node labels

3 possible edge labels

Setting
Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree

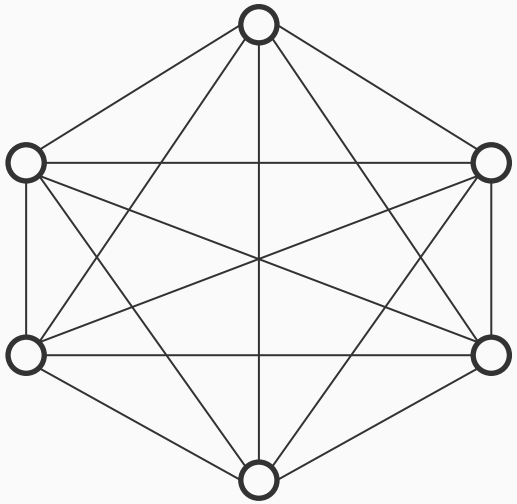
One option: Decompose fully. *All nodes and edges are independently scored*



Produced by neural models

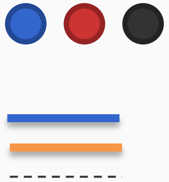
$$\text{score}(\mathbf{x}, \mathbf{y}) = \sum_{n \in \text{ nodes}(\mathbf{x},\mathbf{y})} \text{score}(n) + \sum_{e \in \text{ edges}(\mathbf{x},\mathbf{y})} \text{score}(e)$$

Decomposing the output: Example



3 possible node labels

3 possible edge labels

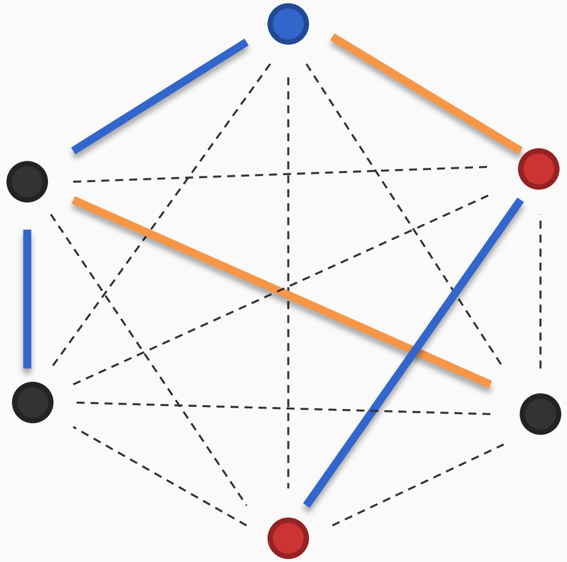


Setting

Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree

One option: Decompose fully. *All nodes and edges are independently scored*



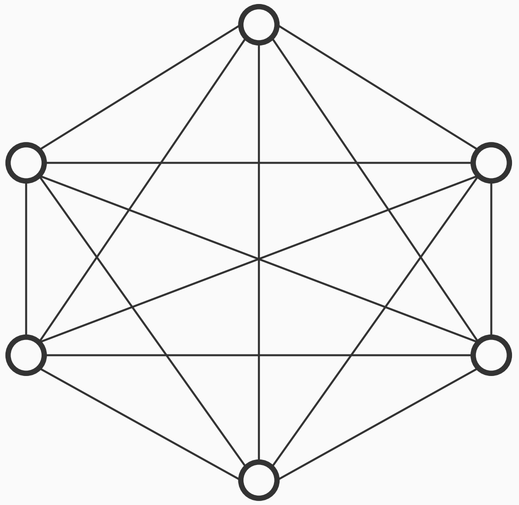
Still need to ensure that the colored edges form a valid output (i.e. a tree)

$$\text{score}(\mathbf{x}, \mathbf{y}) = \sum_{n \in \text{ nodes}(\mathbf{x},\mathbf{y})} \text{score}(n) + \sum_{e \in \text{ edges}(\mathbf{x},\mathbf{y})} \text{score}(e)$$

Prediction:

$$\arg\max_{\mathbf{y}} \ \text{score}(\mathbf{x}, \mathbf{y}) \quad \text{s.t.} \quad \mathbf{y} \text{ forms a tree}$$

Decomposing the output: Example



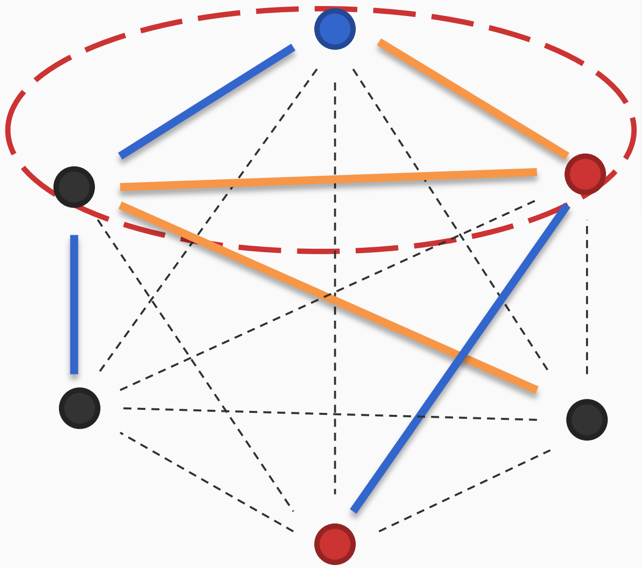
3 possible node labels

3 possible edge labels

Setting
Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree

One option: Decompose fully. *All nodes and edges are independently scored*



Still need to ensure that the colored edges form a valid output (i.e. a tree)

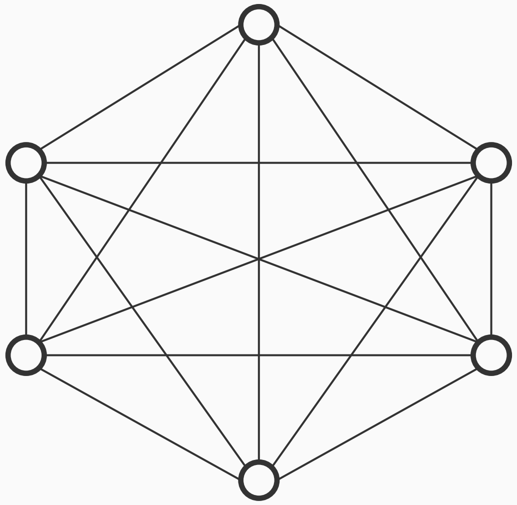
This is invalid output!
Even this simple decomposition requires ***inference*** to ensure validity

$$\mathrm{score}(\mathbf{x}, \mathbf{y}) = \sum_{n \in \text{ nodes}(\mathbf{x},\mathbf{y})} \mathrm{score}(n) + \sum_{e \in \text{ edges}(\mathbf{x},\mathbf{y})} \mathrm{score}(e)$$

Prediction:
$$\arg\max_{\mathbf{y}} \quad \mathrm{score}(\mathbf{x}, \mathbf{y})$$
$$\text{s.t.} \quad \mathbf{y} \text{ forms a tree}$$

Decomposing the output: Example

Another possibility: Score each edge and its nodes together



3 possible node labels

3 possible edge labels

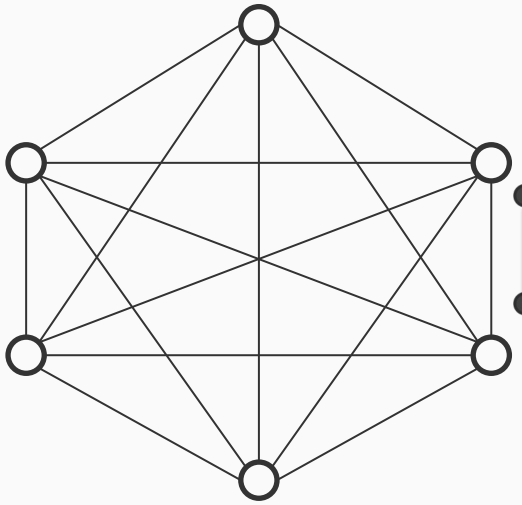
Setting

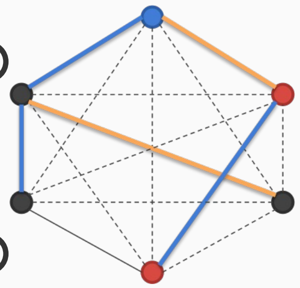
Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree

Decomposing the output: Example

Another possibility: Score each edge and its nodes together





3 possible node labels

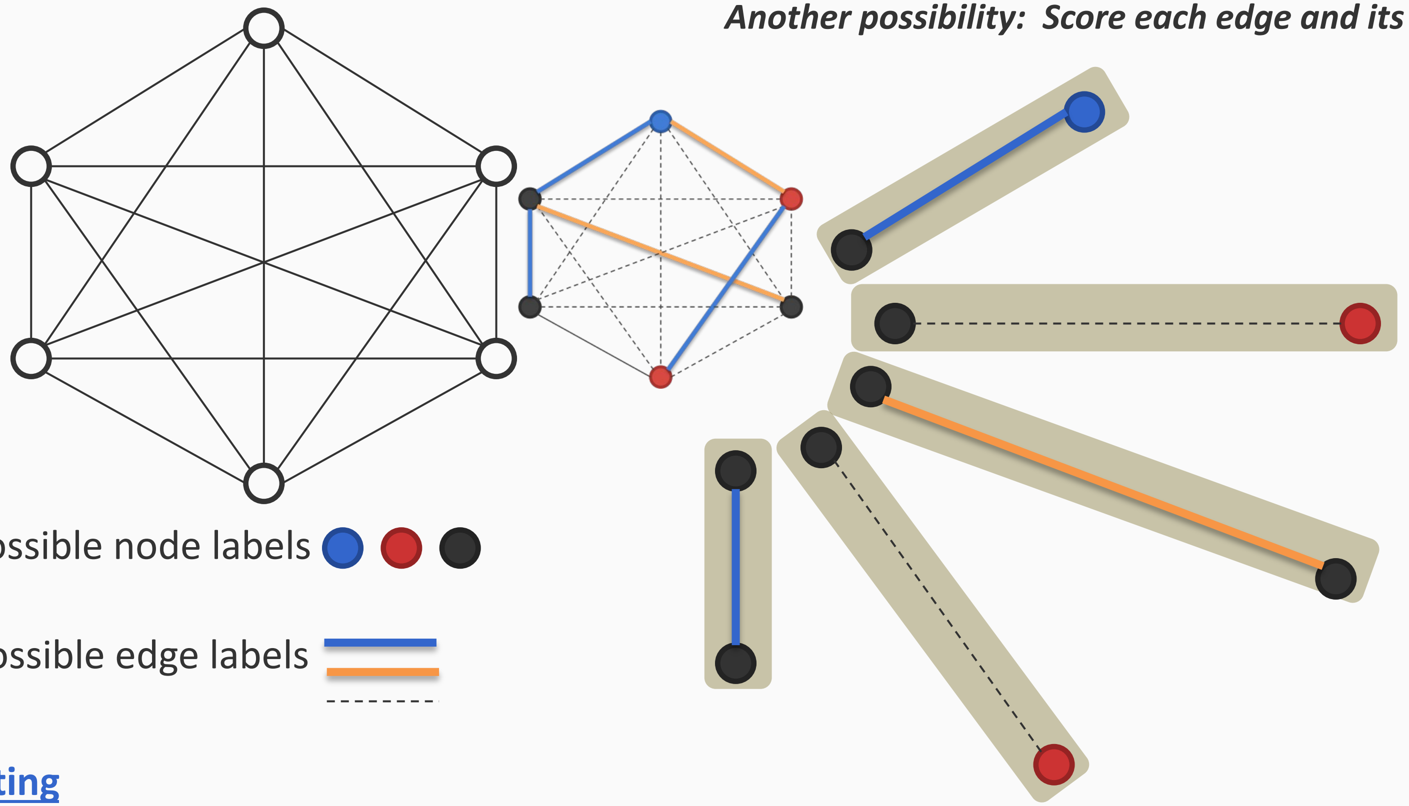
3 possible edge labels

[Setting]()

Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree

Decomposing the output: Example



Another possibility: Score each edge and its nodes together

And many other edges…

Each patch represents piece that is scored independently

3 possible node labels

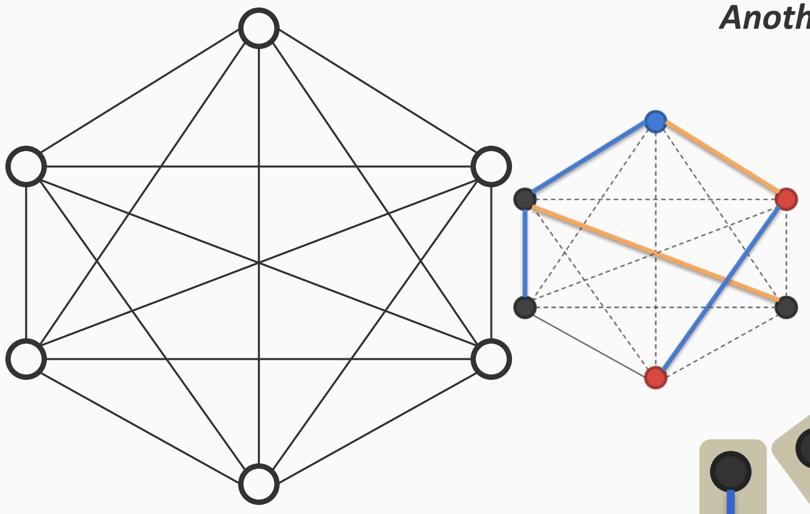
3 possible edge labels

Setting

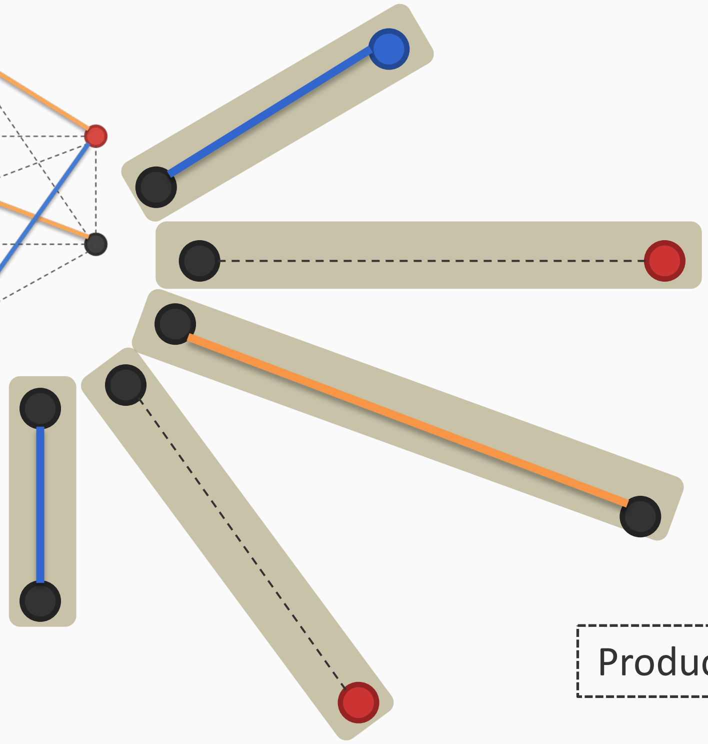
Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree

Decomposing the output: Example



Another possibility: Score each edge and its nodes together



And many other edges…

Each patch represents piece that is scored independently

3 possible node labels

3 possible edge labels

Setting

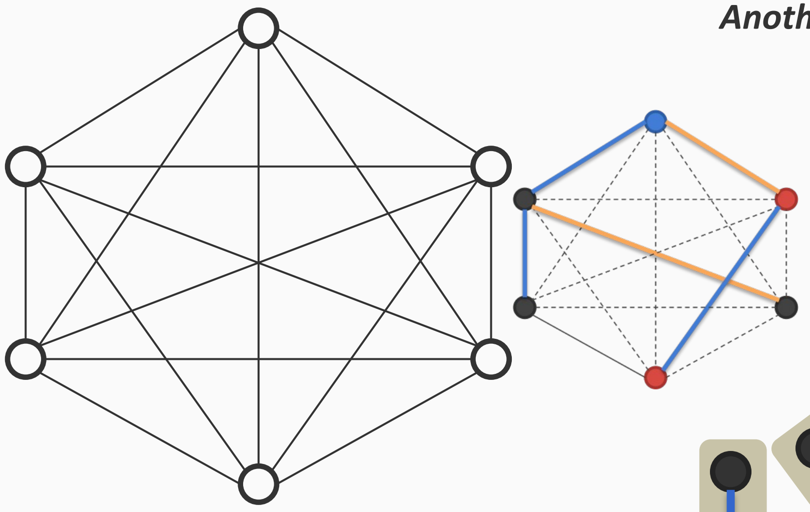
Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree

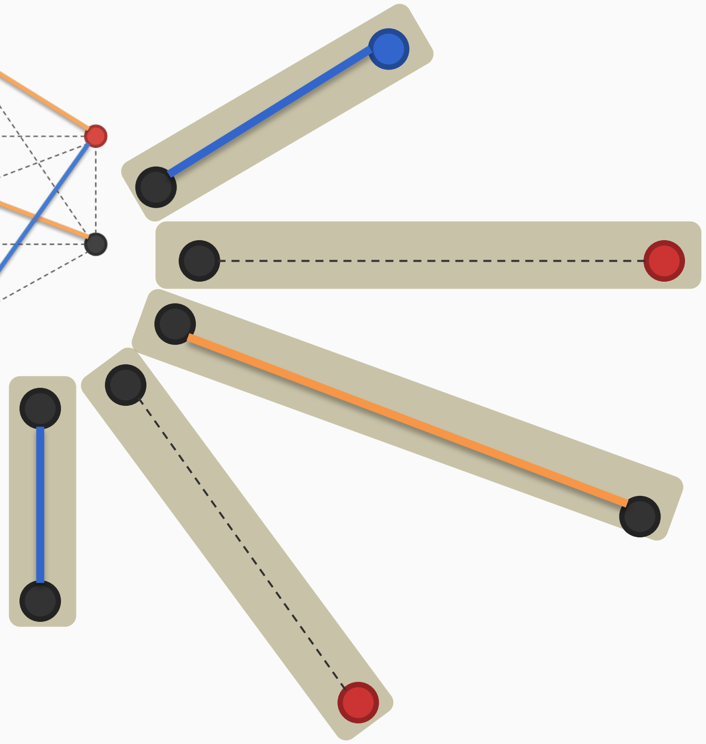
Produced by neural models

$$\text{score}(\mathbf{x}, \mathbf{y}) = \sum_{\substack{n_1, n_2 \in \text{ nodes}(\mathbf{x},\mathbf{y}) \\ e \in \text{ edges}(\mathbf{x},\mathbf{y})}} \text{score}(n_1, n_2, e)$$

Decomposing the output: Example



Another possibility: Score each edge and its nodes together



And many other edges…

Each patch represents piece that is scored independently

3 possible node labels 

3 possible edge labels 

Setting

Output: Nodes and edges are labeled and the blue and orange edges form a tree

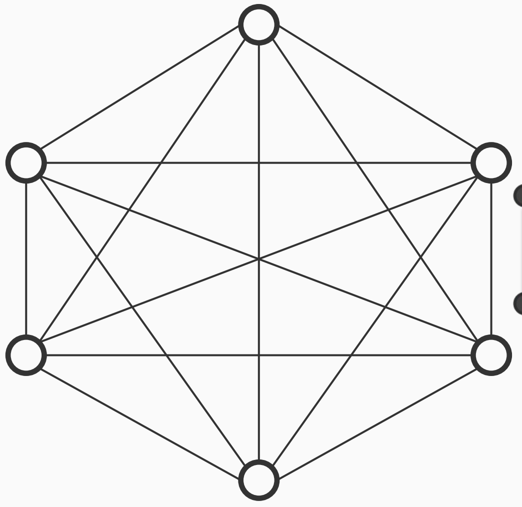
Goal: Find the highest scoring labeling such that the edges that are colored form a tree

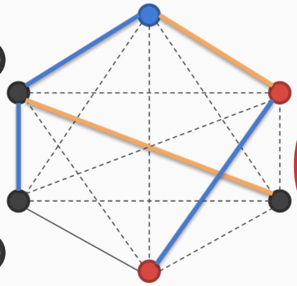
Inference should ensure that

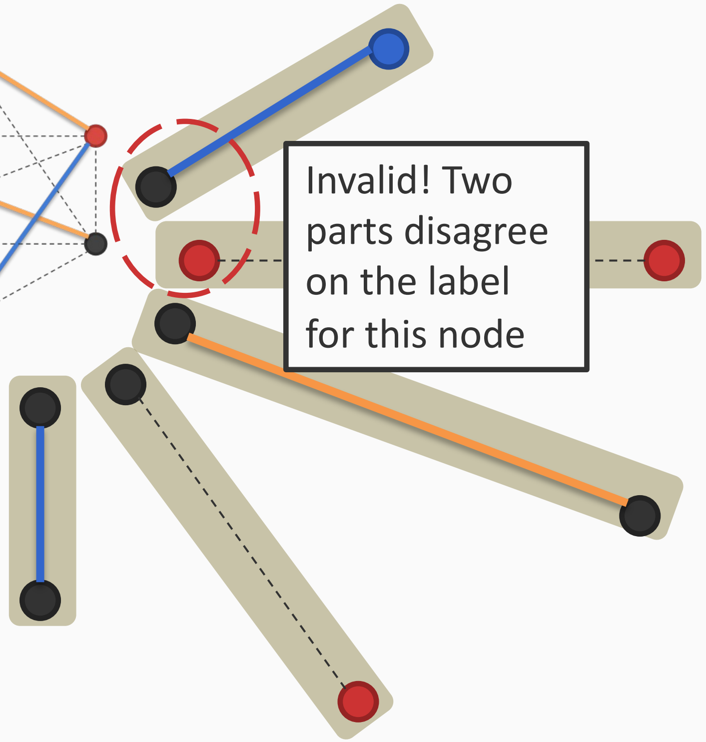
1. The output is a tree, and
2. Shared nodes have the same label in all the pieces

Decomposing the output: Example

Another possibility: Score each edge and its nodes together







And many other edges…

Each patch represents piece that is scored independently

3 possible node labels

3 possible edge labels

Setting

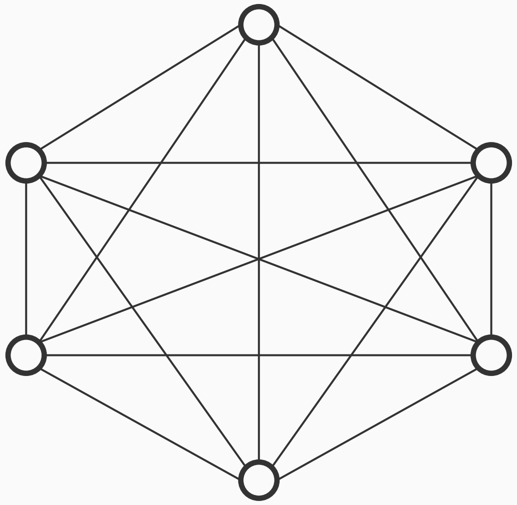
Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree

Inference should ensure that

1. The output is a tree, and
2. *Shared nodes have the same label in all the parts*

Decomposing the output: Example

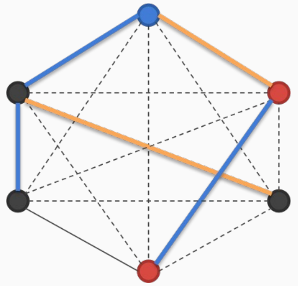


3 possible node labels

3 possible edge labels

Setting
Output: Nodes and edges are labeled and the blue and orange edges form a tree

Goal: Find the highest scoring labeling such that the edges that are colored form a tree



We have seen two examples of decomposition

Many other decompositions possible…

Structured inference

- Each part is scored independently
  - Key observation: Number of possible inference outcomes for each part may not be large
    - Even if the number of possible structures might be large
- Inference: How to glue together the pieces to build a valid output?
  - Depends on the "shape" of the output
- Computational complexity of inference is important
  - Worst case: intractable
  - With assumptions about the output, polynomial algorithms exist.
    - Predicting sequence chains: Viterbi algorithm
    - To parse a sentence into a tree: CKY algorithm
    - In general, might have to either live with intractability or approximate

Questions?

Computational issues

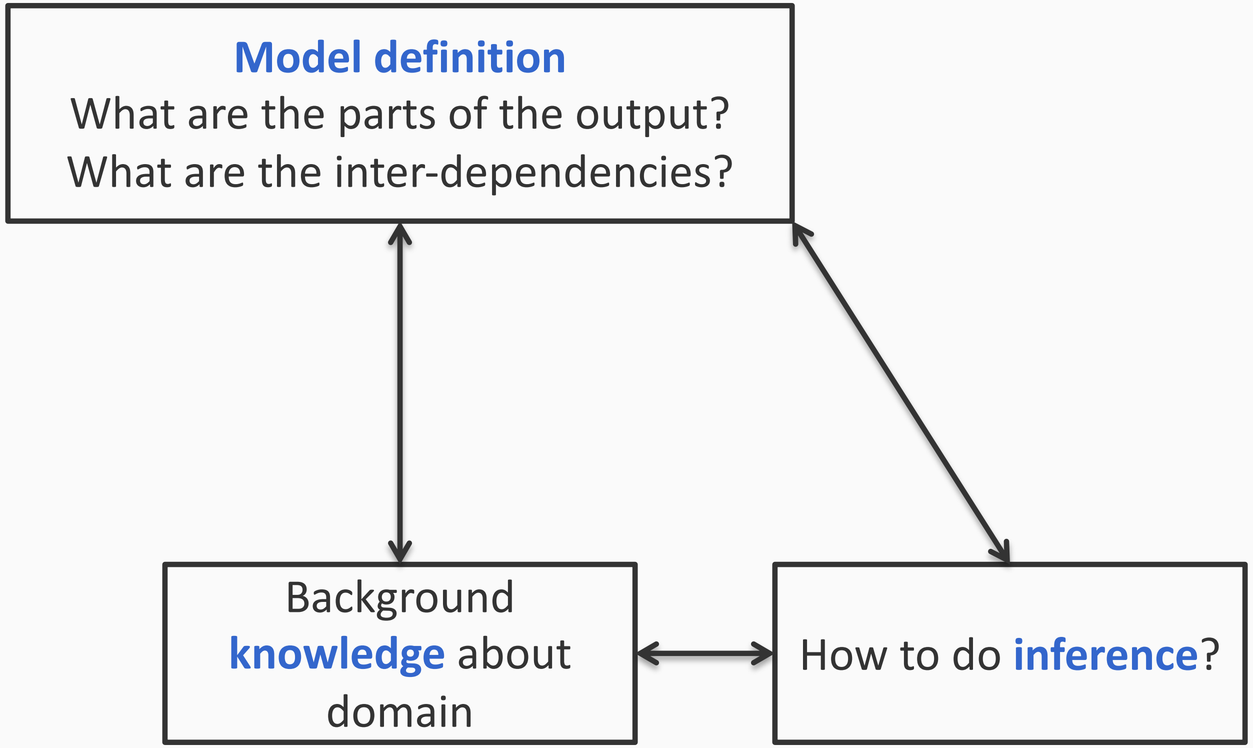
Background **knowledge** about domain

Computational issues

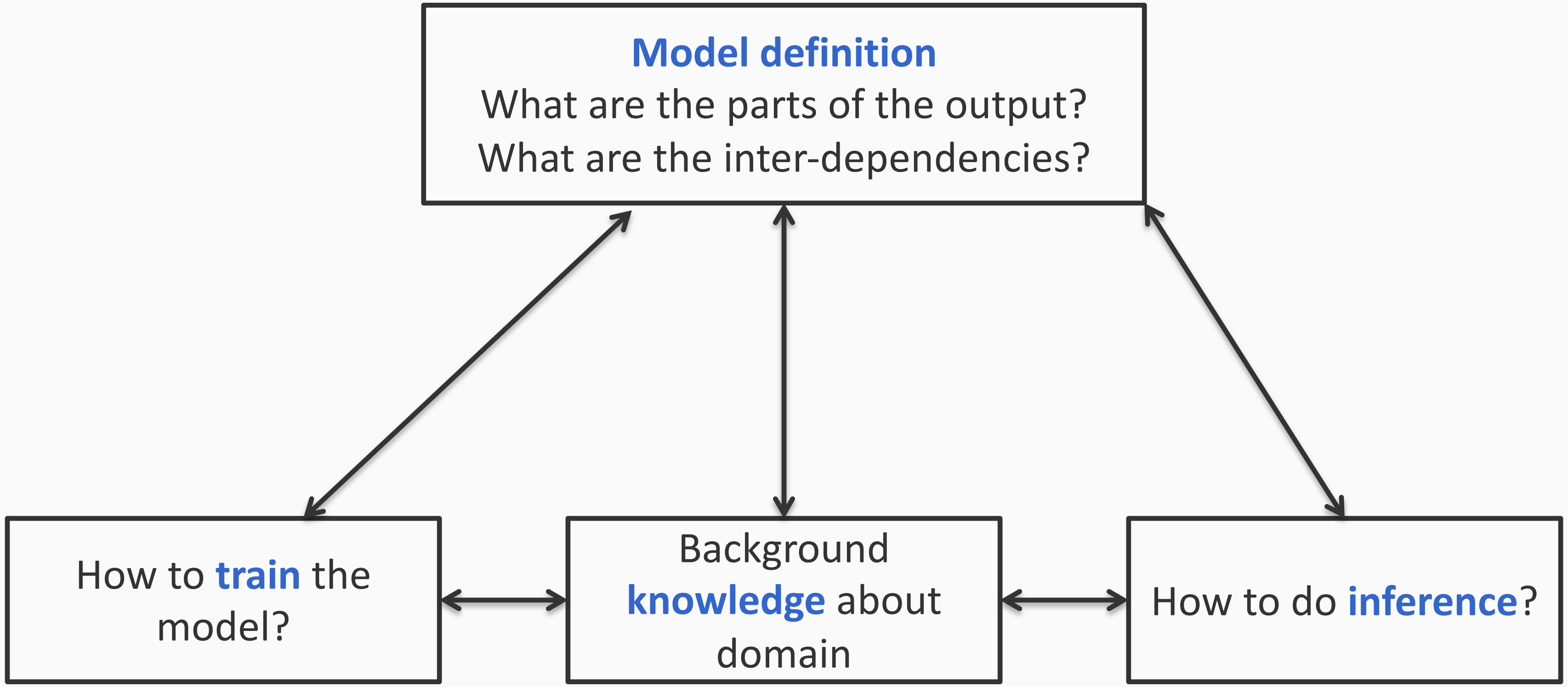
Model definition
What are the parts of the output?
What are the inter-dependencies?

Background
knowledge about
domain

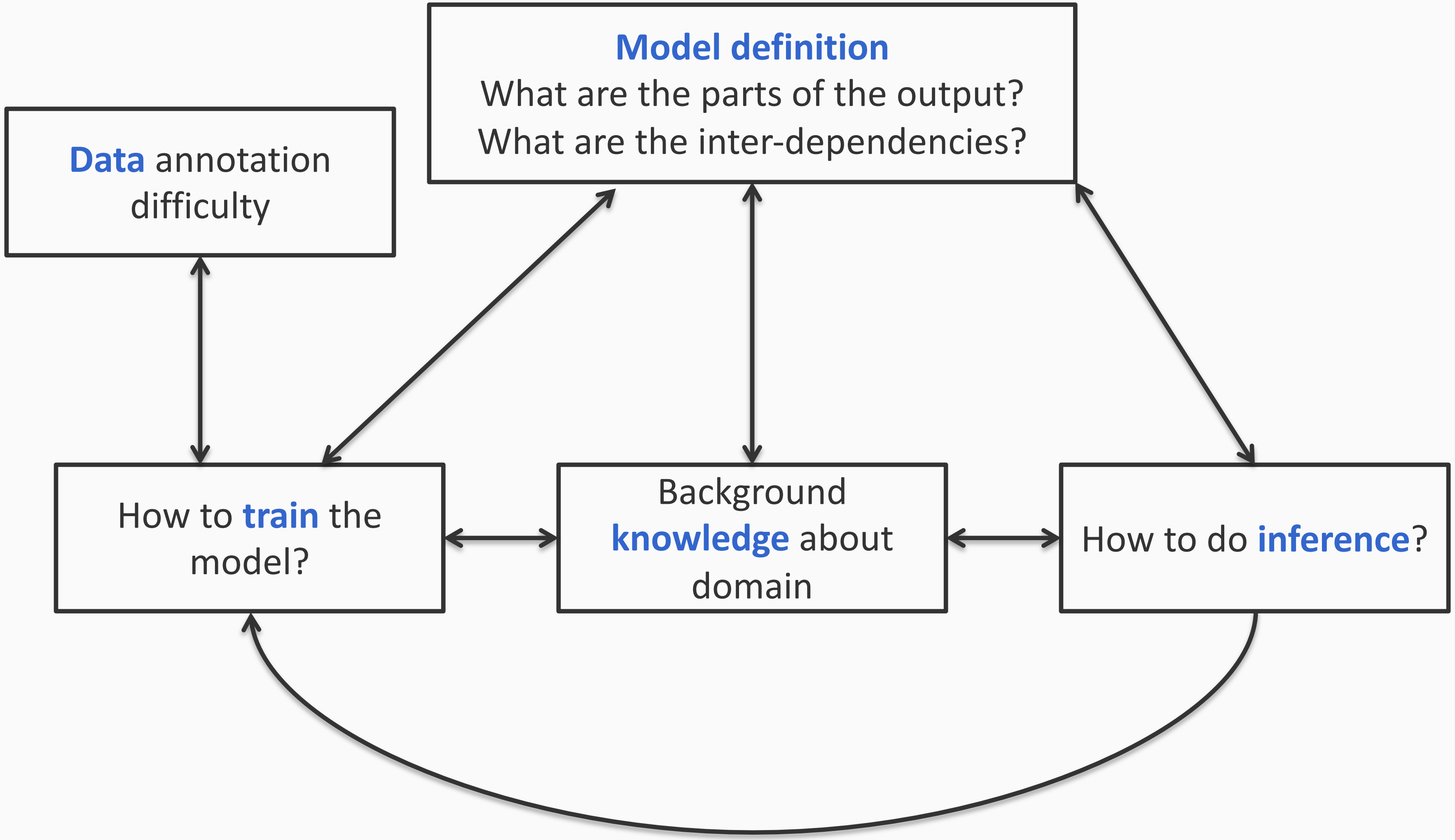
Computational issues



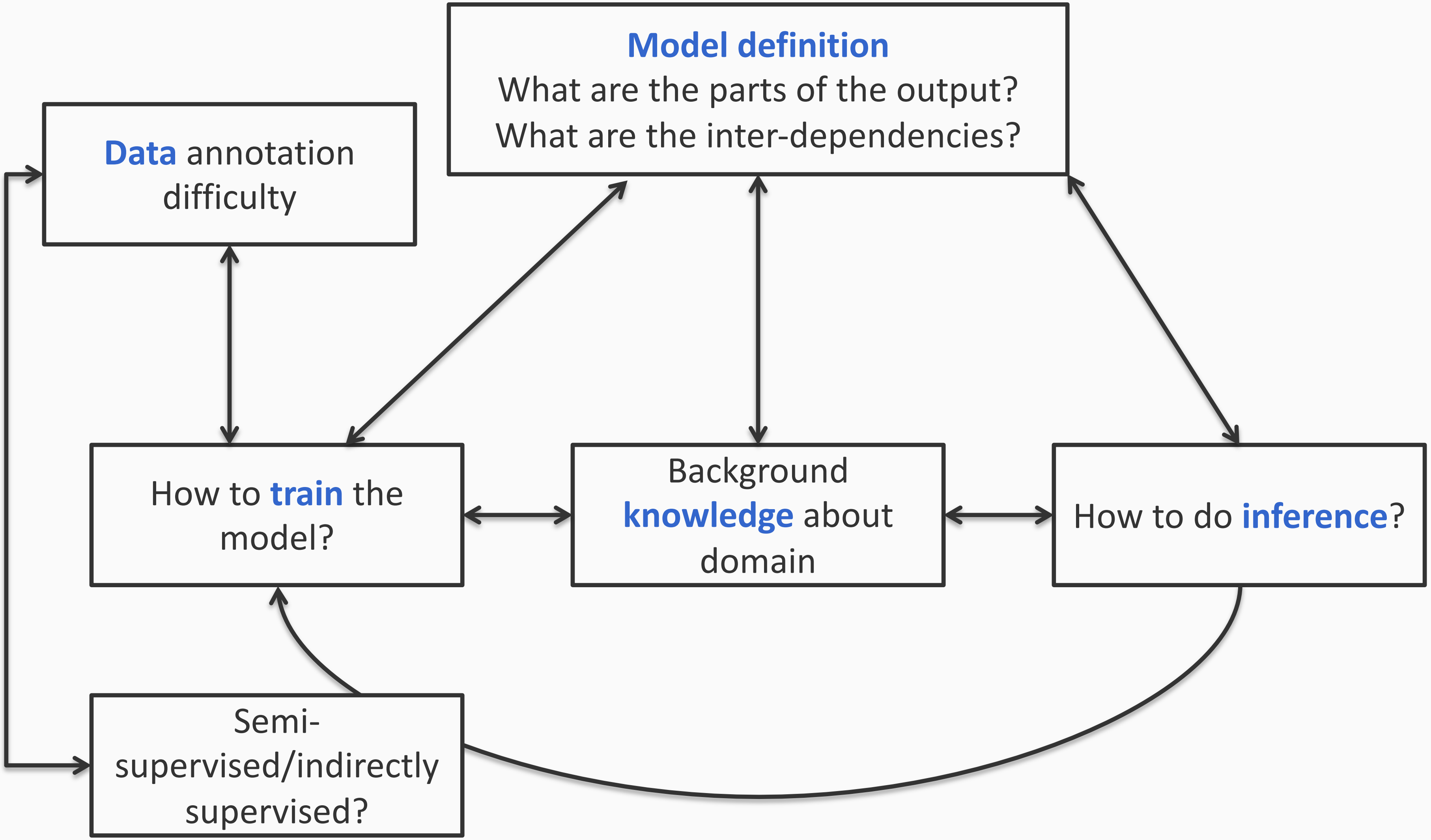
Computational issues



Computational issues



Computational issues



Lecture outline

- Neural networks & structured outputs
- Examples of structured outputs
- Computational questions with structured prediction
- Inference

The bigger picture

- The goal of structured prediction: Predicting a graph

- Modeling: Defining probability distributions over the random variables
  - Involves making independence assumptions

- Learning creates functions that score predictions

- Inference: The computational step that actually constructs the output
  - Also called *decoding* in some papers

Inference questions

- This class:
  - Mostly we use inference to mean "*What is the highest scoring assignment to the output random variables for a given input?*"
  - Maximum A Posteriori (MAP) inference (if the score is probabilistic)

Inference questions

- This class:
  - Mostly we use inference to mean "*What is the highest scoring assignment to the output random variables for a given input?*"
  - Maximum A Posteriori (MAP) inference (if the score is probabilistic)

- Other inference questions

Inference questions

- This class:
  - Mostly we use inference to mean "*What is the highest scoring assignment to the output random variables for a given input?*"
  - Maximum A Posteriori (MAP) inference (if the score is probabilistic)

- Other inference questions
  - What is the highest scoring assignment to *some* of the output variables given the input?

Inference questions

- This class:
  - Mostly we use inference to mean "*What is the highest scoring assignment to the output random variables for a given input?*"
  - Maximum A Posteriori (MAP) inference (if the score is probabilistic)

- Other inference questions
  - What is the highest scoring assignment to *some* of the output variables given the input?
  - Sample from the posterior distribution over the output space

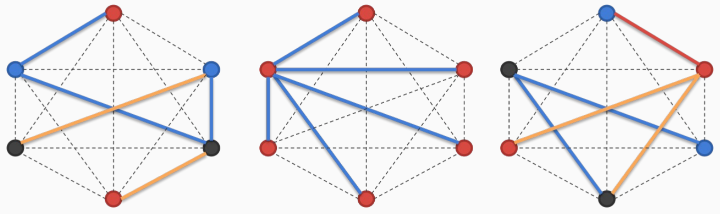
Inference questions

- This class:
  - Mostly we use inference to mean "*What is the highest scoring assignment to the output random variables for a given input?*"
  - Maximum A Posteriori (MAP) inference (if the score is probabilistic)

- Other inference questions
  - What is the highest scoring assignment to *some* of the output variables given the input?
  - Sample from the posterior distribution over the output space
  - Computing marginal probabilities over output space

MAP inference

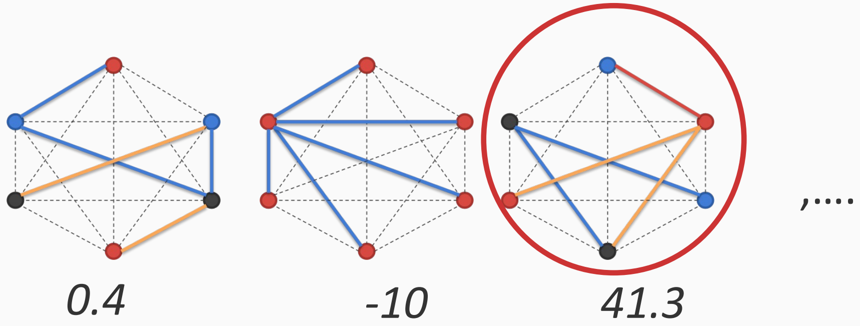
- A combinatorial problem



0.4 -10 41.3 ,....

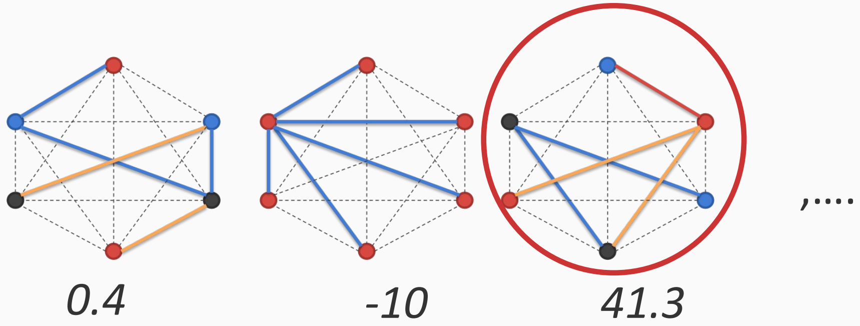
MAP inference

- A combinatorial problem



MAP inference is discrete optimization

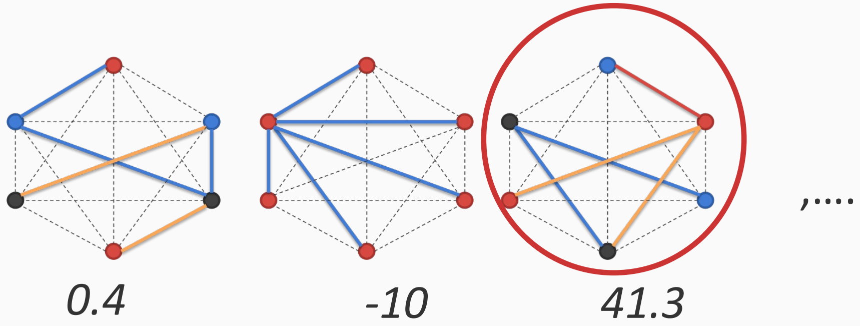
- A combinatorial problem



- Computational complexity depends on
  - The size of the input
  - The *factorization* of the scores
  - More complex factors generally lead to expensive inference

MAP inference is discrete optimization

- A combinatorial problem



- Computational complexity depends on
  - The size of the input
  - The factorization of the scores
  - More complex factors generally lead to expensive inference

- A generally bad strategy in most but the simplest cases: *"Enumerate all possible structures and pick the highest scoring one"*

MAP inference is search

- We want a collections of decisions that has highest score

$$\underset{\mathbf{y}}{\operatorname{argmax}}\, w^T \phi(\mathrm{x}, \mathbf{y}) \qquad\qquad \mathbf{y} = (y_1, y_2, \cdots y_n)$$

- No algorithm can find the max without considering every possible structure
  - Why?

- Key question to consider: How should we solve this computational problem?
  - Exploit the structure of the search space and the cost function
  - That is, exploit decomposition of the scoring function

Approaches for inference

- Exact vs. approximate inference
  - Should the maximization be performed exactly?
    - Or is a close-to-highest-scoring structure good enough?
  - **Exact**: Search, dynamic programming, integer linear programming, ….
  - **Heuristic**: Gibbs sampling, belief propagation (some cases), beam search, linear programming relaxations (some cases), …
    - Also called *approximate inference*

- Randomized vs. deterministic
  - Relevant for approximate inference: If I run the inference program twice, will I get the same answer?