First look at structures

CS 6355: Structured Prediction

THE UNIVERSITY OF UTAH

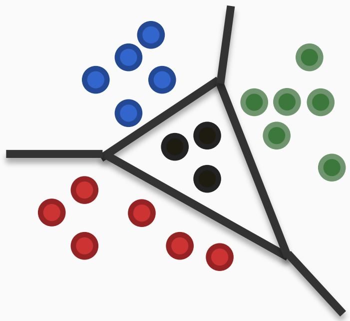
So far…

- Binary classifiers
  - Output: 0/1
- Multiclass classifiers
  - Output: one of a set of labels
- Linear classifiers for both
  - Learning algorithms
- Winner-take-all prediction for multiclass

What we have seen: Training multiclass classifiers

- Label belongs to a set that has more than two elements

- Methods
  - Decomposition into a collection of binary (*local*) decisions
    - One-vs-all
    - All-vs-all
    - Error correcting codes
  - Training a single (*global*) classifier
    - Multiclass SVM
    - Constraint classification



Questions?

This lecture

- What is structured output?
- Multiclass as a structure
- Constructing outputs from pieces

Where are we?

- What is structured output?
  - Examples

- Multiclass as a structure

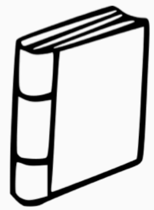
- Constructing outputs from pieces

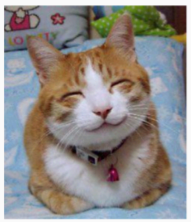
Recipe for multiclass classification

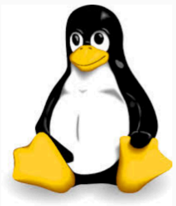
– Collect a training set (hopefully with correct labels)


book


penguin


book


dog


penguin

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

And, $\mathbf{y} \in \{book, \;\; dog, \;\; penguin\}$

– Define linear functions to score labels

$$\underset{\mathbf{y} \in \{book, dog, penguin\}}{\operatorname{argmax}} \mathbf{w}_{\mathbf{y}}^T \mathbf{x}$$

Natural extension to non-linear scoring functions too

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

Recipe for multiclass classification

- Train weights so that it scores examples correctly

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

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

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

  – 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

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

- Prediction: $\underset{\mathbf{y} \in \{book, dog, penguin\}}{\mathrm{argmax}} \; \mathrm{score}(\mathbf{x}, \mathbf{y})$
  - 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 1: 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 1: Information extraction

Predicting entities and relations from text

Colin went back home to 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`

Example 2: Semantic Role Labeling

The task: Given a sentence, identify who does what to whom, where and when.

The bus was ***heading*** for Nairobi in Kenya

Example 2: Semantic Role Labeling

The task: Given a sentence, identify who does what to whom, where and when.

The bus was ***heading*** for Nairobi in Kenya

Relation: *`to head`*
Mover[**A0**]: the bus
Destination[**A1**]: Nairobi in Kenya

Example 2: Semantic Role Labeling

The task: Given a sentence, identify who does what to whom, where and when.

The bus was ***heading*** for Nairobi in Kenya

Predicate

Arguments

Relation: *to head*

Mover[**A0**]: the bus

Destination[**A1**]: Nairobi in Kenya

Predicting verb arguments

The bus was **heading** for Nairobi in Kenya.

Predicting verb arguments

The bus was **heading** for Nairobi in Kenya.

1. Identify candidate arguments for verb using parse tree
   - Filtered using a binary classifier

Predicting verb arguments

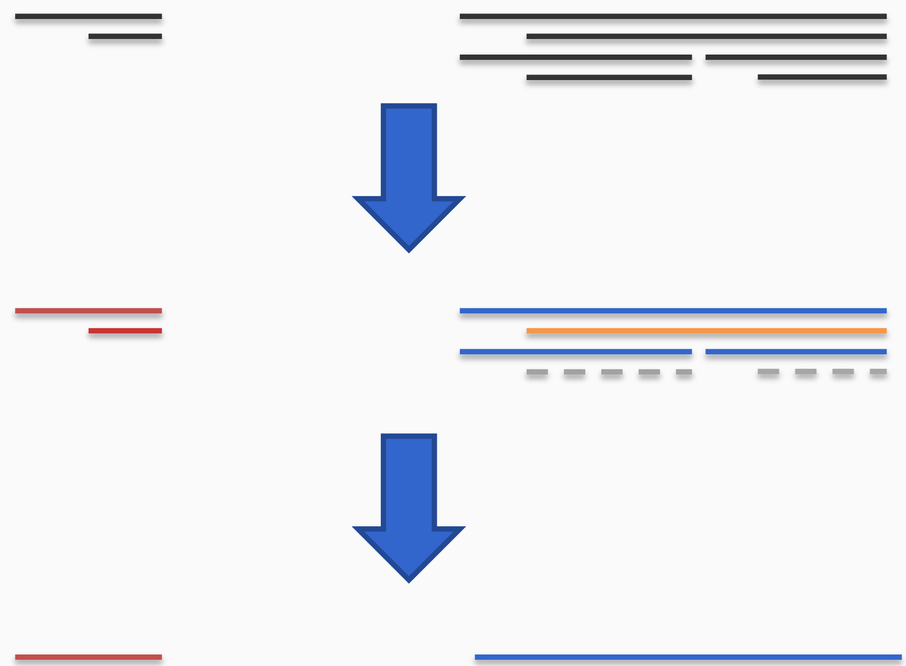
The bus was **heading** for Nairobi in Kenya.

1. Identify candidate arguments for verb using parse tree
   – Filtered using a binary classifier
2. Classify argument candidates
   – Multi-class classifier (one of multiple labels per candidate)

Each color is a different label here

Predicting verb arguments

The bus was **heading** for Nairobi in Kenya.



1. Identify candidate arguments for verb using parse tree
   – Filtered using a binary classifier
2. Classify argument candidates
   – Multi-class classifier (one of multiple labels per candidate)
3. Inference
   – Using probability estimates from argument classifier
   – Must respect structural and linguistic constraints
      • Eg: The same word can not be part of two arguments

Inference: verb arguments

The bus was **heading** for Nairobi in Kenya.

Suppose we are assigning colors to each span

Special label, meaning "Not an argument"

Inference: verb arguments

The bus was **heading** for Nairobi in Kenya.

Inference: verb arguments

The bus was **heading** for Nairobi in Kenya.

Inference: verb arguments

The bus was **heading** for Nairobi in Kenya.

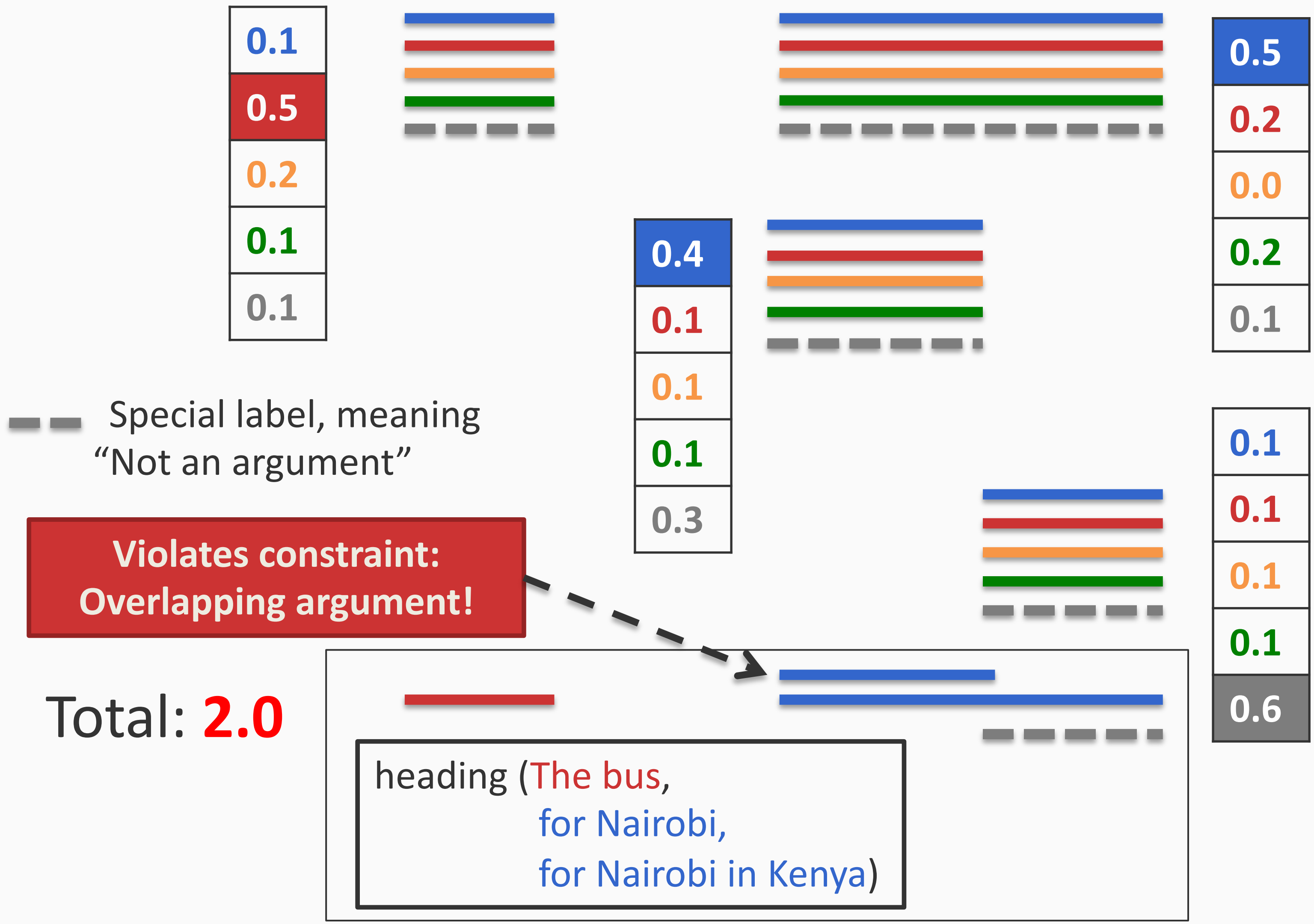
Special label, meaning "Not an argument"

Total: **2.0**

heading (The bus,
for Nairobi,
for Nairobi in Kenya)

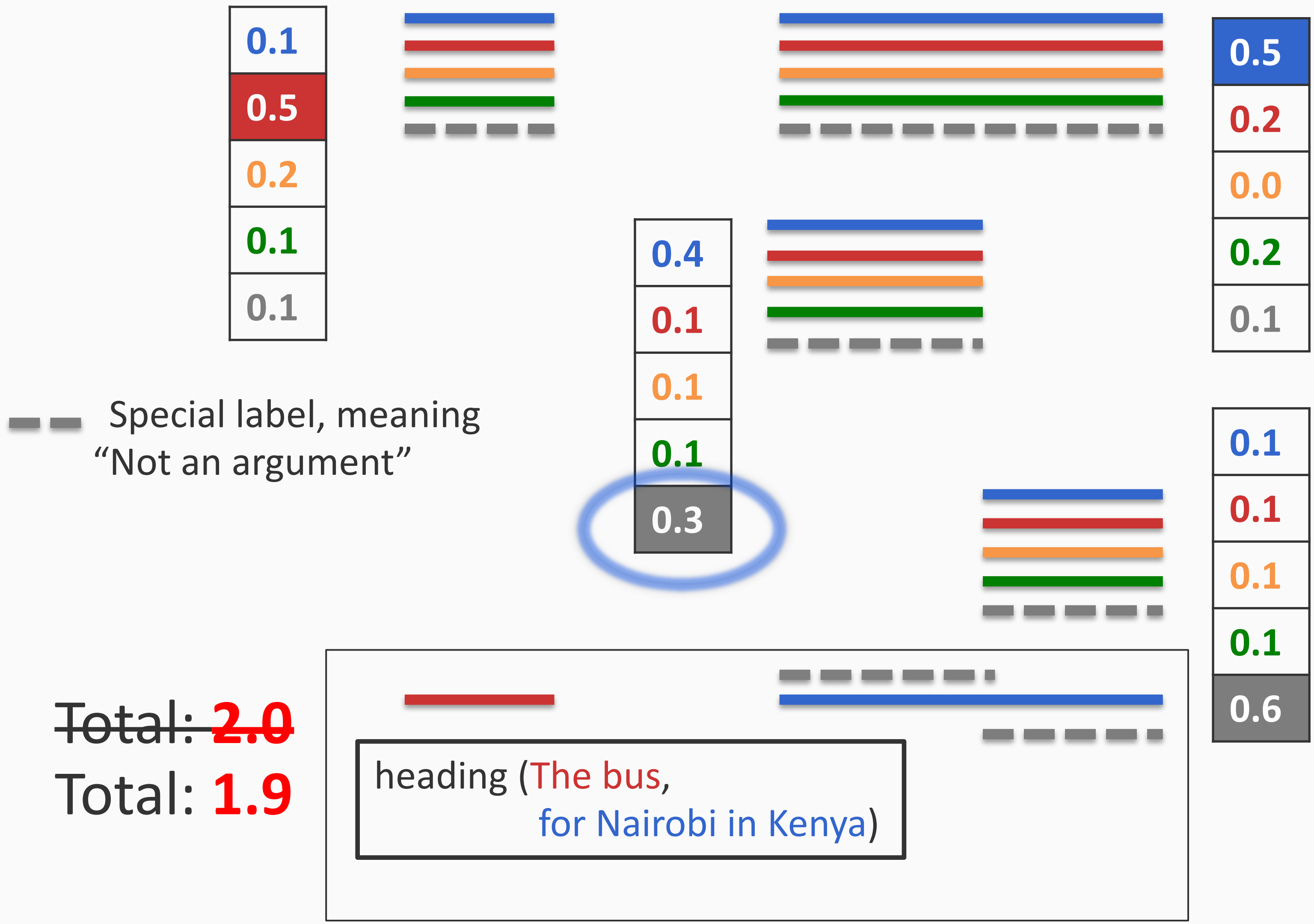
Inference: verb arguments

The bus was **heading** for Nairobi in Kenya.



Inference: verb arguments

The bus was **heading** for Nairobi in Kenya.



Inference: verb arguments

The bus was **heading** for Nairobi in Kenya.

Input $\mathbf{x}$ Text with pre-processing

Output Five possible decisions for each candidate (●●●●●)

Create a binary variable for each decision, only one of which is **true** for each candidate. Collectively, a "structure"

$$\mathbf{y} = \{y^1, y^2, \cdots, y^n\} \in \{0,1\}^n$$

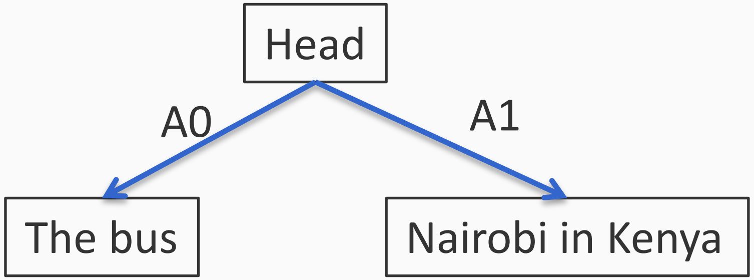
heading (The bus,
for Nairobi in Kenya)

Structured output is…

- A *data structure* with a pre-defined schema
  - Eg: SRL converts raw text into a record in a database

Predicate	A0	A1	Location
Head	The bus	Nairobi in Kenya	-

- Equivalently, a *graph*
  - Often restricted to be a specific family of graphs: chains, trees, etc

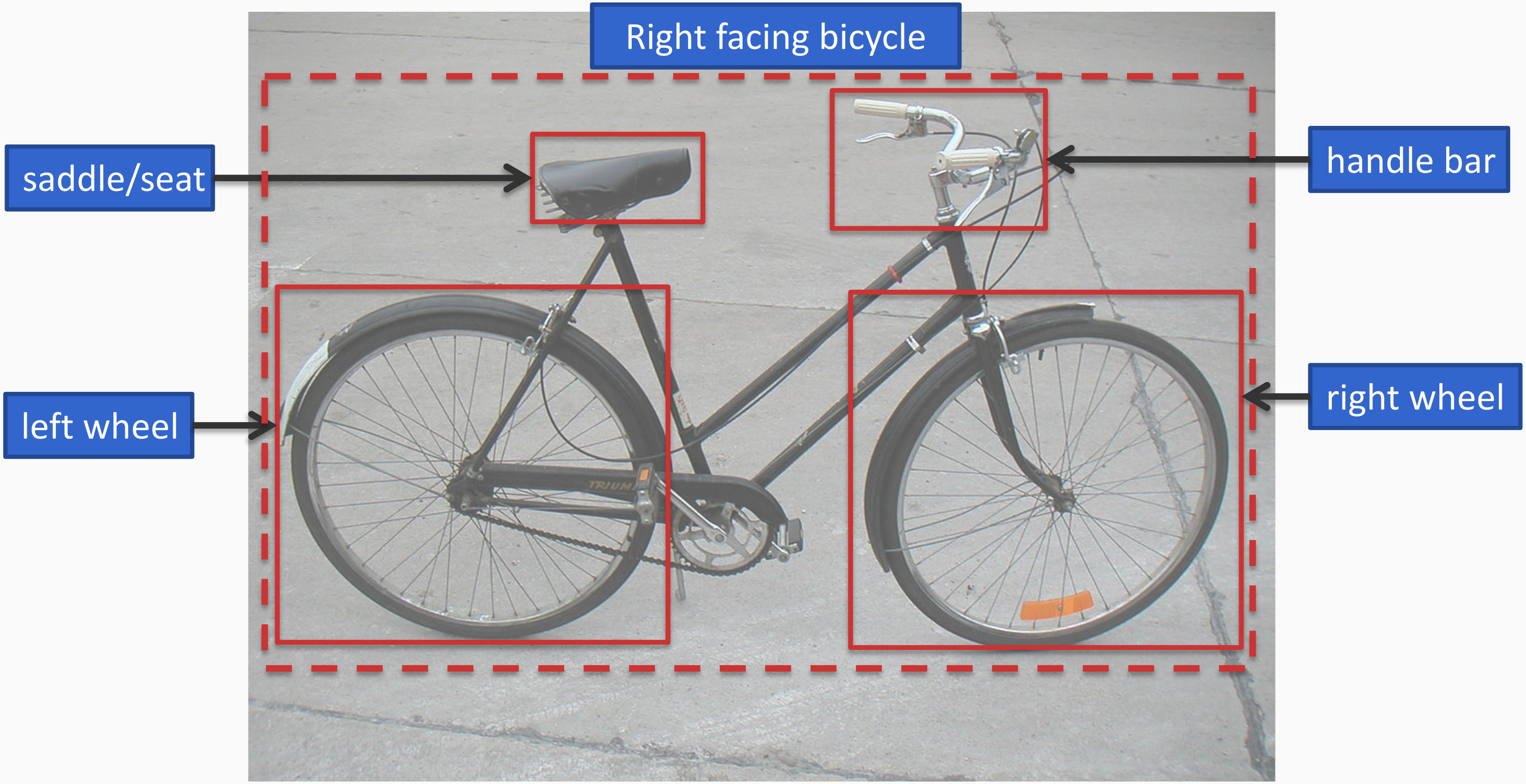


Questions/comments?

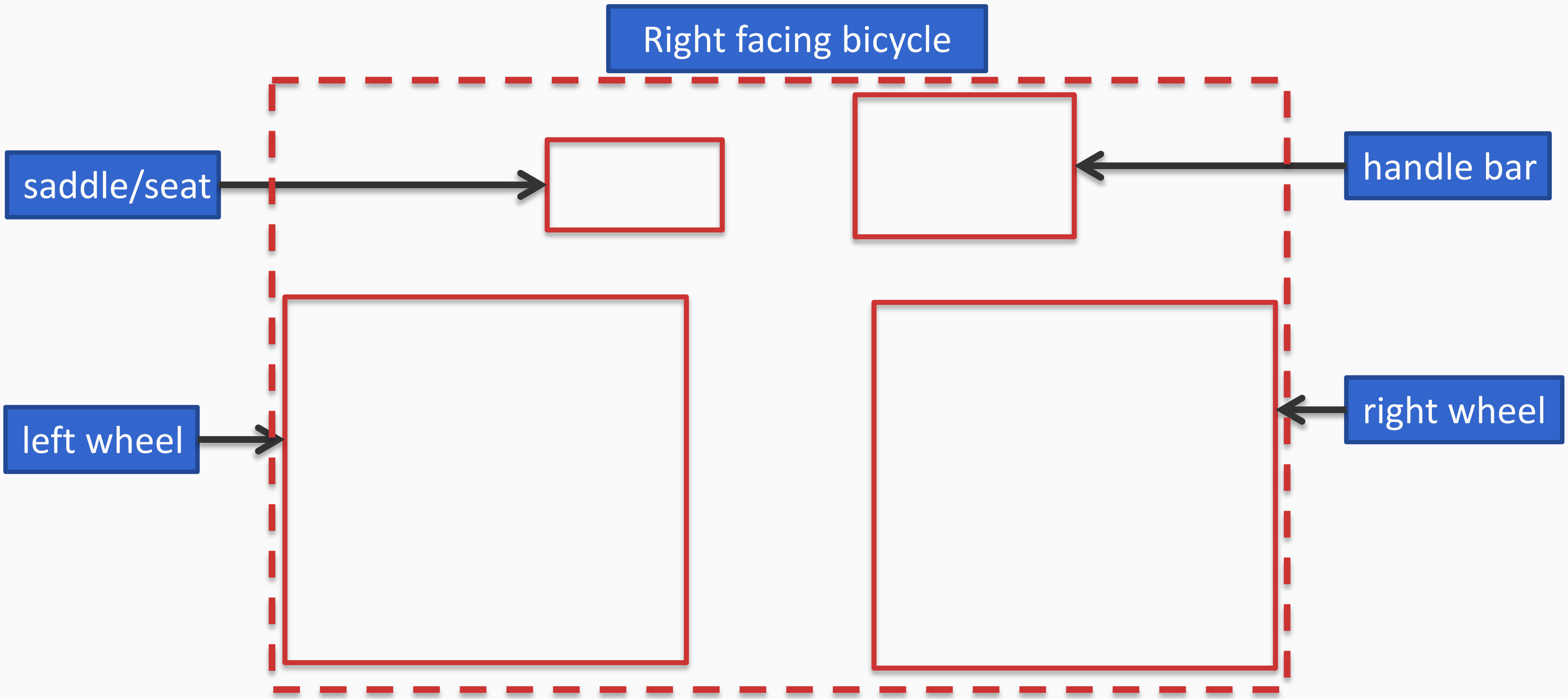
Example 3: Object detection

Example 3: Object detection

Example 3: Object detection



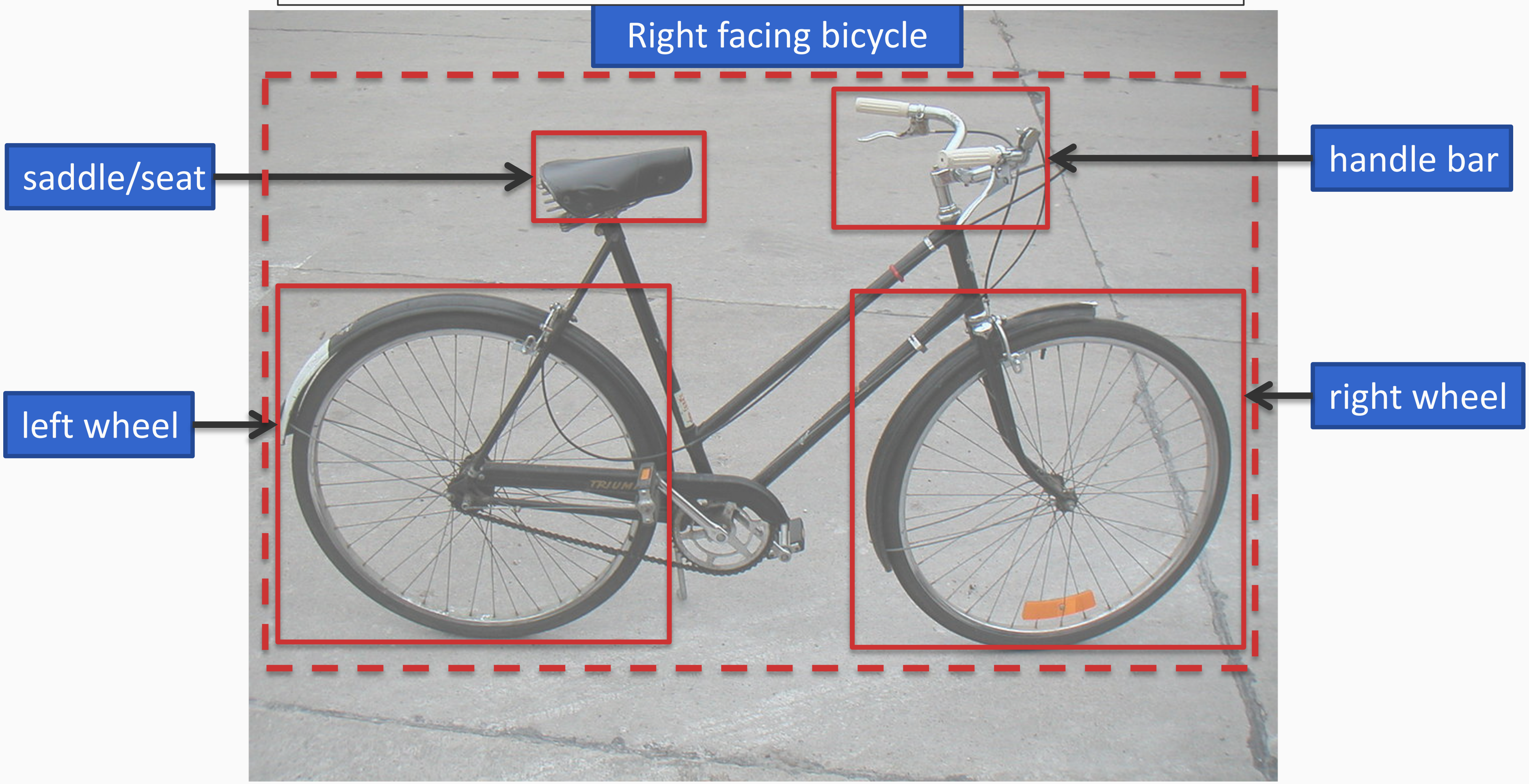
The output: A schematic showing the parts and their relative layout



Once again, a structure

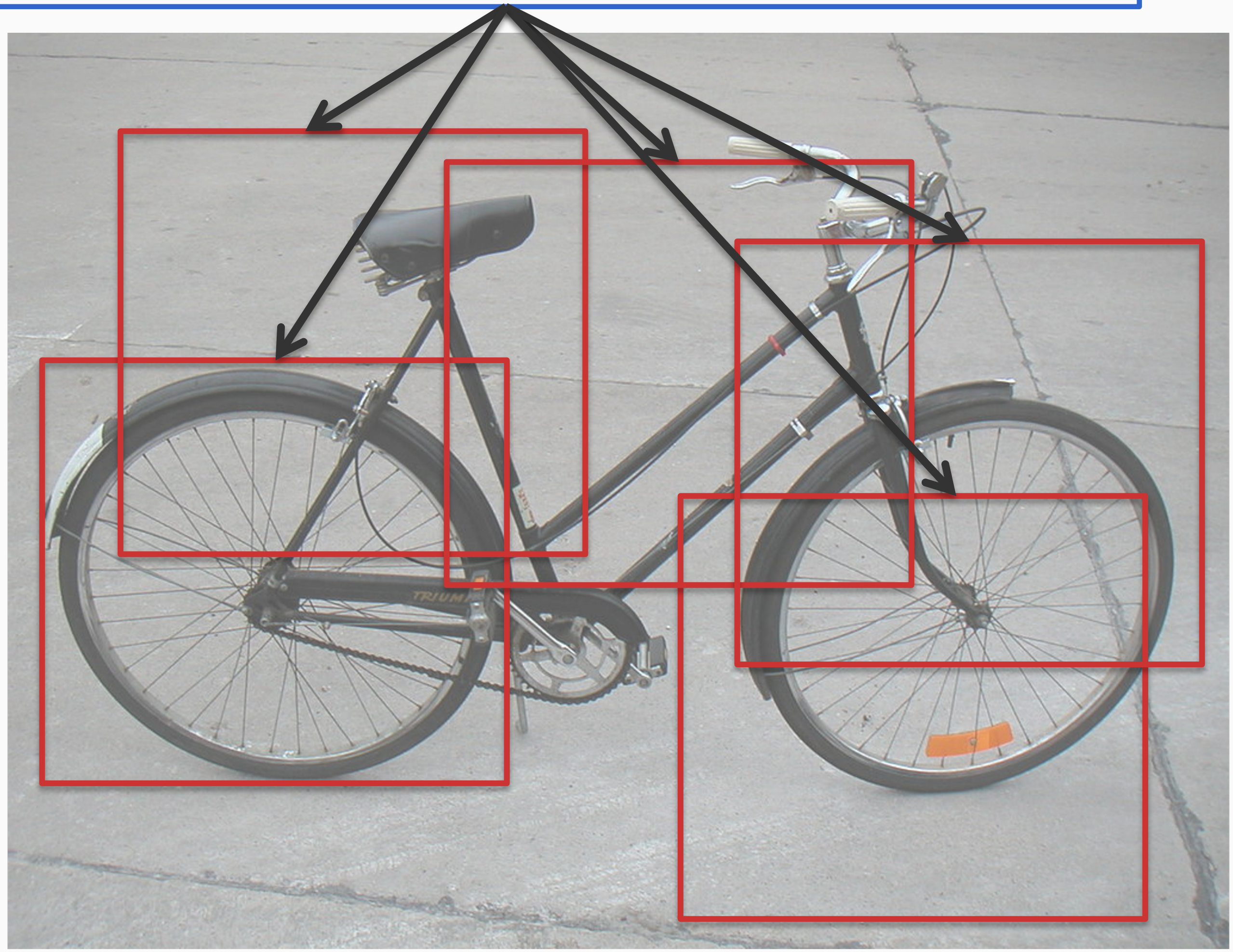
Object detection

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



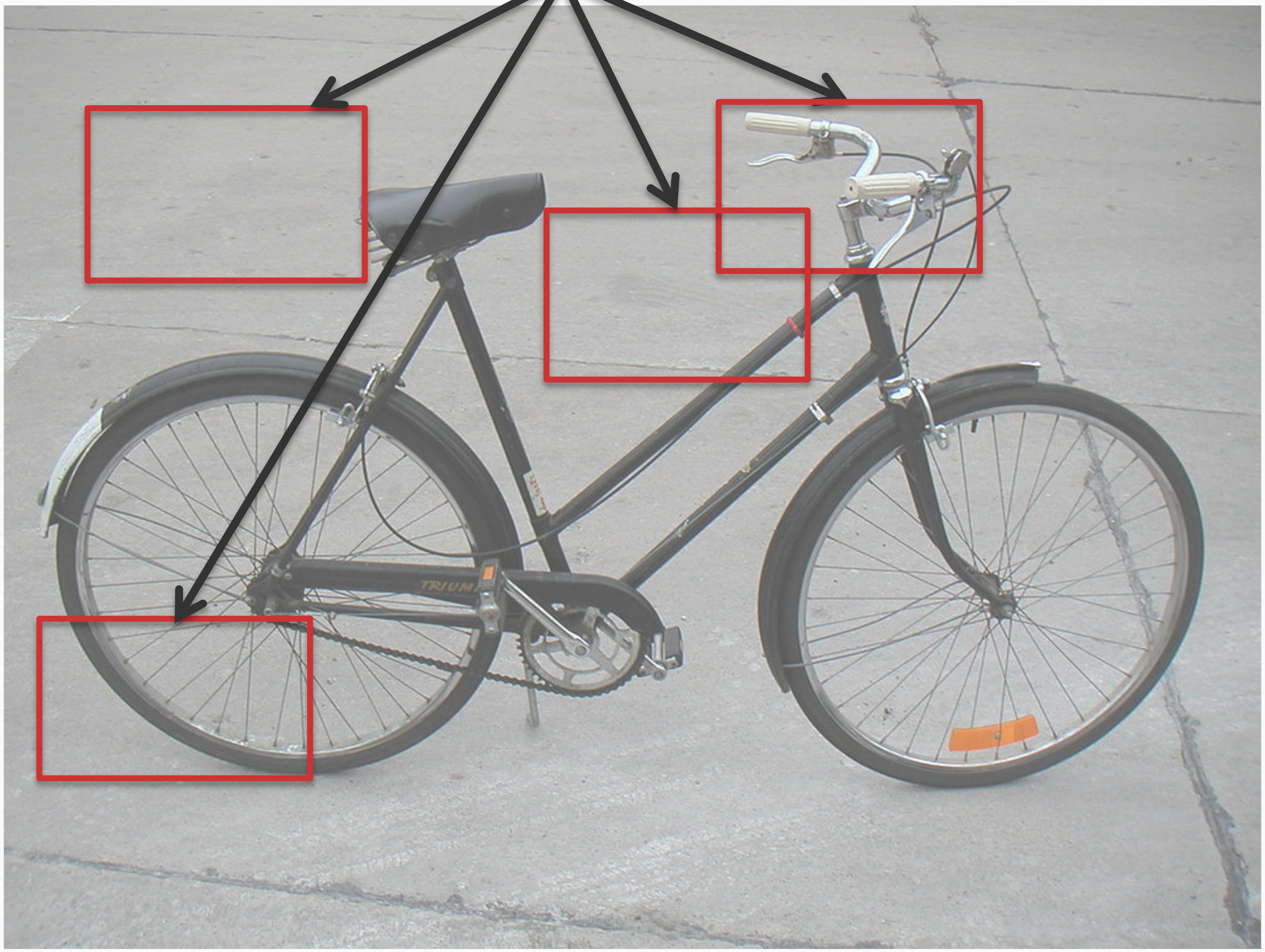
One approach to build this structure

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



One approach to build this structure

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



One approach to build this structure

1. Left wheel detector

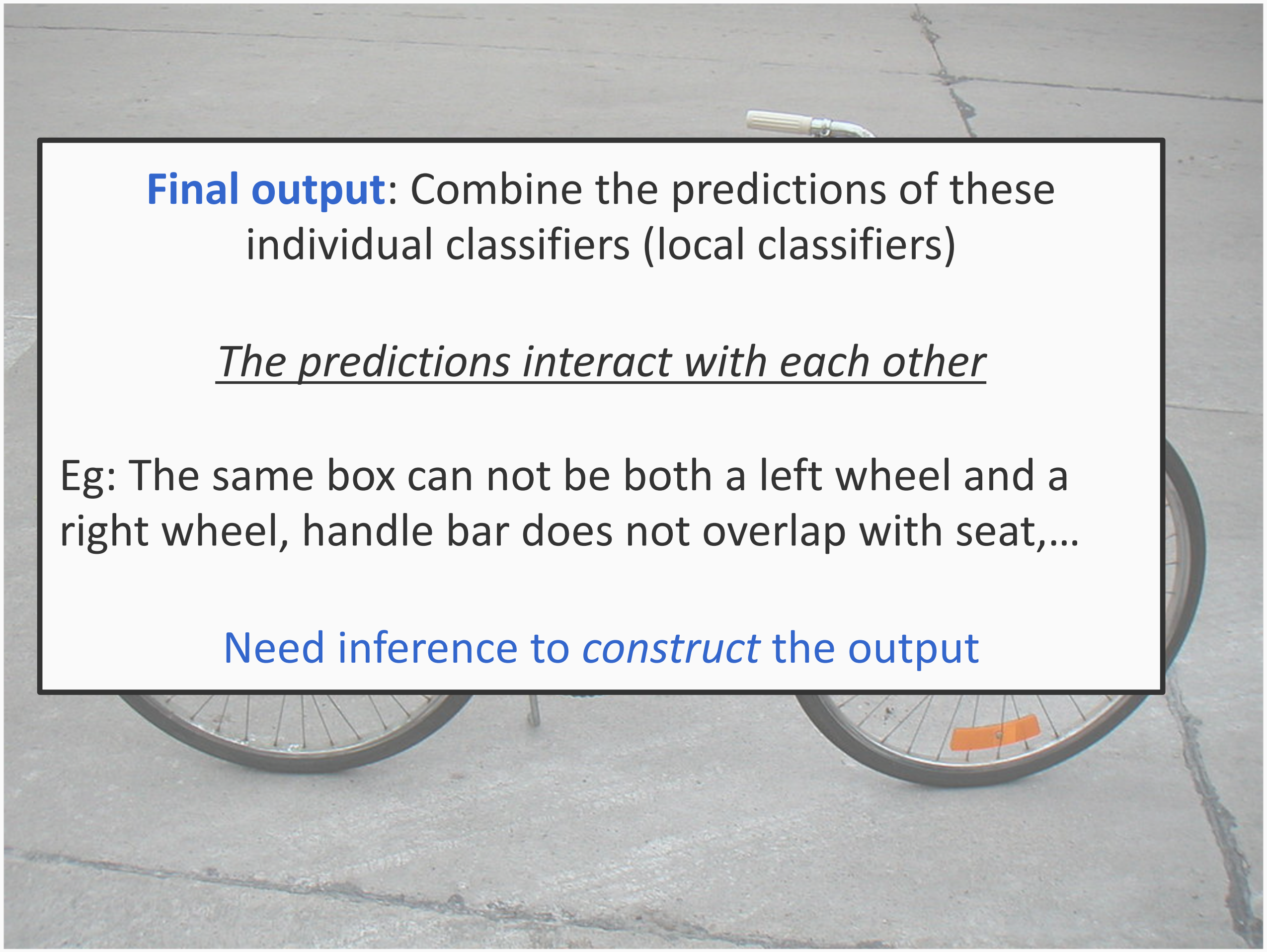
2. Right wheel detector

3. Handle bar detector

4. Seat detector

One approach to build this structure

1. Left wheel detector
2. Right wheel detector
3. Handle bar detector
4. Seat detector



Final output: Combine the predictions of these individual classifiers (local classifiers)

The predictions interact with each other

Eg: The same box can not be both a left wheel and a right wheel, handle bar does not overlap with seat,…

Need inference to *construct* the output

Example 4: Sequence labeling

More on this in next lecture

- Input: A sequence of *tokens* (like words)
- Output: A sequence of labels of same length as input

Eg: Part-of-speech tagging:

Given a sentence, find parts-of-speech of all the words

Example 4: Sequence labeling

More on this in next lecture

- Input: A sequence of *tokens* (like words)
- Output: A sequence of labels of same length as input

Eg: Part-of-speech tagging:

Given a sentence, find parts-of-speech of all the words

The	Fed	raises	interest	rates

Example 4: Sequence labeling

More on this in next lecture

- Input: A sequence of *tokens* (like words)
- Output: A sequence of labels of same length as input

Eg: Part-of-speech tagging:

Given a sentence, find parts-of-speech of all the words

The	Fed	raises	interest	rates
`Determiner`	`Noun`	`Verb`	`Noun`	`Noun`

Example 4: Sequence labeling

More on this in next lecture

- Input: A sequence of *tokens* (like words)
- Output: A sequence of labels of same length as input

Eg: Part-of-speech tagging:

Given a sentence, find parts-of-speech of all the words

The	Fed	raises	interest	rates
`Determiner`	`Noun`	`Verb`	`Noun`	`Noun`
Other labels are possible in different contexts	`Verb` (I *fed* the dog)		`Verb` (Poems don't *interest* me)	`Verb` (He *rates* movies online)

Part-of-speech tagging

Given a word, its label depends on :

- The identity and characteristics of the word

  E.g. Raises is a `Verb` because it ends in –es (among other reasons)

- Its grammatical context

  - Fed in “The Fed” is a `Noun` because it follows a `Determiner`
  - Fed in “I fed the..” is a `Verb` because it follows a `Pronoun`

Part-of-speech tagging

Given a word, its label depends on :

- The identity and characteristics of the word
- Its grammatical context

One possible modeling assumption:

Each output label is dependent on its neighbors in addition to the input word

Part-of-speech tagging

Given a word, its label depends on :

- The identity and characteristics of the word
- Its grammatical context

One possible modeling assumption:

Each output label is dependent on its neighbors in addition to the input word

Two kinds of scoring functions for labels

1. Score for label associating with a particular word in context
2. Score for a pair of labels following each other

Part-of-speech tagging

Given a word, its label depends on :

- The identity and characteristics of the word
- Its grammatical context

One possible modeling assumption:

Each output label is dependent on its neighbors in addition to the input word

Two kinds of scoring functions for labels

1. Score for label associating with a particular word in context
2. Score for a pair of labels following each other

What we want: Find a sequence of labels that *maximizes* the sum/product of these scores

More examples

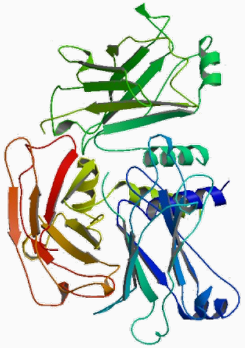
Protein 3D structure prediction

```
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 SRL frame 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…

Representation

- 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 SRL frame output, a sequence of labels etc

Procedural

- A collection of inter-dependent decisions
  - Eg: The sequence of decisions used to construct the output
- The result of a combinatorial optimization problem

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

We have seen something similar before in the context of multiclass

Structured output is…

Representation

- 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 SRL frame output, a sequence of labels etc

Procedural

- A collection of inter-dependent decisions
  - Eg: The sequence of decisions used to construct the output
- The result of a combinatorial optimization problem

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

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 weight vector for each possible inference outcome
   - For multiclass, we could train one weight vector 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 weight vector for each possible inference outcome
   - For multiclass, we could train one weight vector 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 so that the whole is meaningful

Where are we?

- What is structured output?
- Multiclass as a structure
  - A very brief digression
- Constructing outputs from pieces

Multiclass as a structured output

- A structure is…
  - A graph (in general, hypergraph), possibly labeled and/or directed
  - A collection of inter-dependent decisions
  - The output of a combinatorial optimization problem

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

- Multiclass
  - A graph with one node and no edges
    - Node label is the output
  - Can be composed via multiple decisions
  - Winner-take-all

$$\underset{i}{\operatorname{argmax}} \, \mathbf{w}^{\mathbf{T}} \phi(\mathbf{x}, i)$$

Multiclass is a structure: Implications

1. A lot of the ideas from multiclass *may* be generalized to structures
   - Not always simple, but useful to keep in mind

2. Broad statements about structured learning must apply to multiclass classification
   - Useful for sanity check, also for understanding

3. Binary classification is the most “trivial” form of structured classification
   - Multiclass with two classes

Questions/comments?

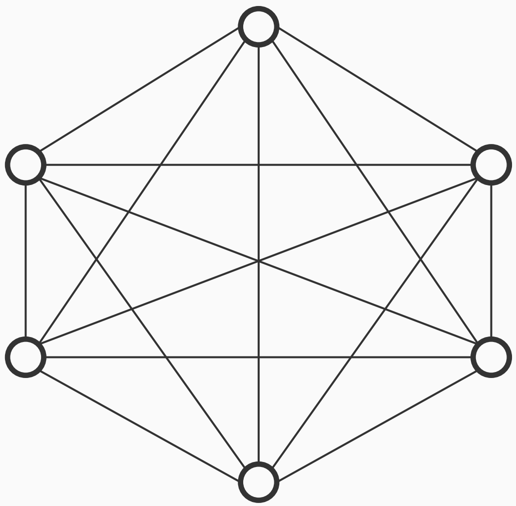
Where are we?

- What is structured output?
- Multiclass as a structure
- Constructing outputs from pieces

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

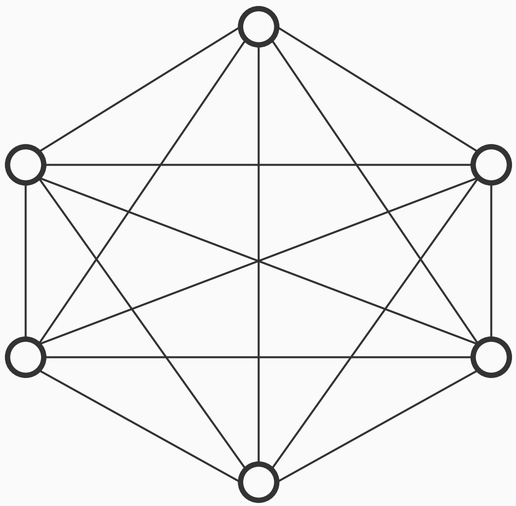
3 possible edge labels

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

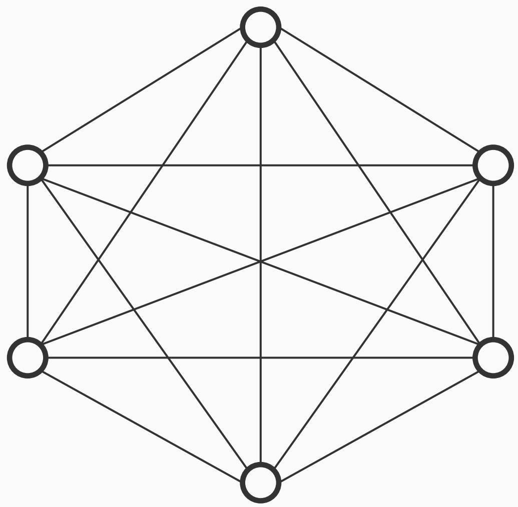
The scoring function scores outputs

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

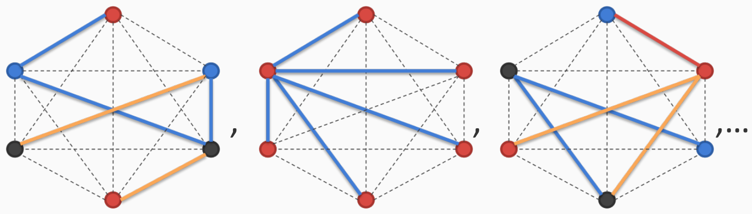
The scoring function scores outputs

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

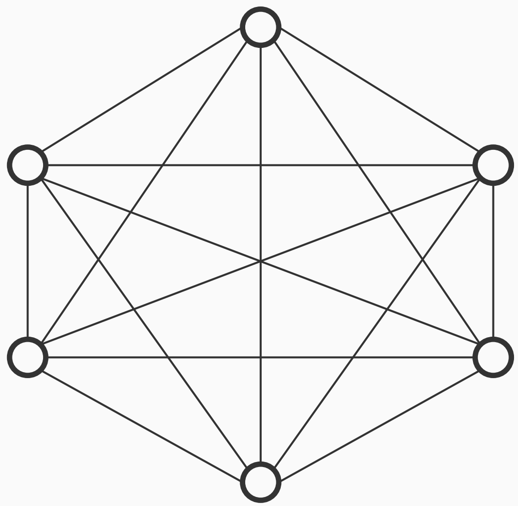
What is the best way to do this decomposition? Depends….

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

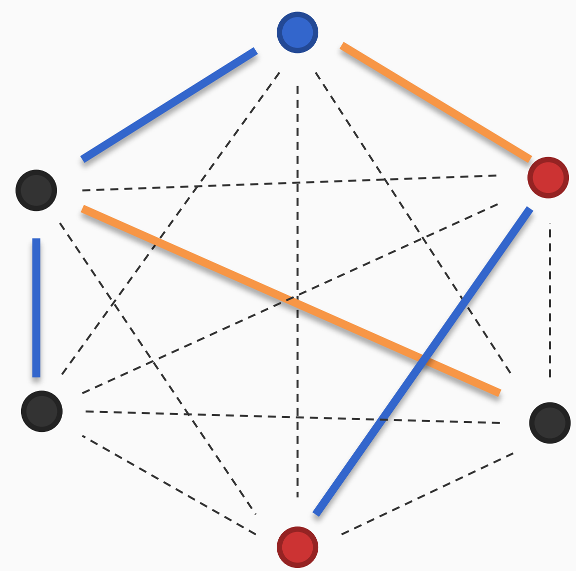
3 possible edge labels

Setting

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

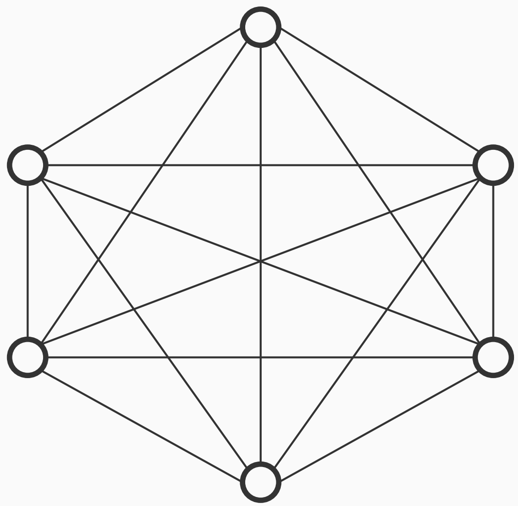
Goal: Find the highest scoring labeling such that the edges that are colored form a tree

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 

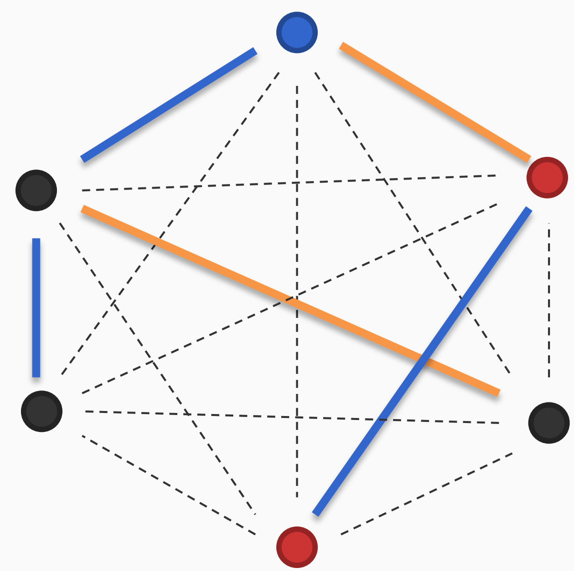
3 possible edge labels 

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

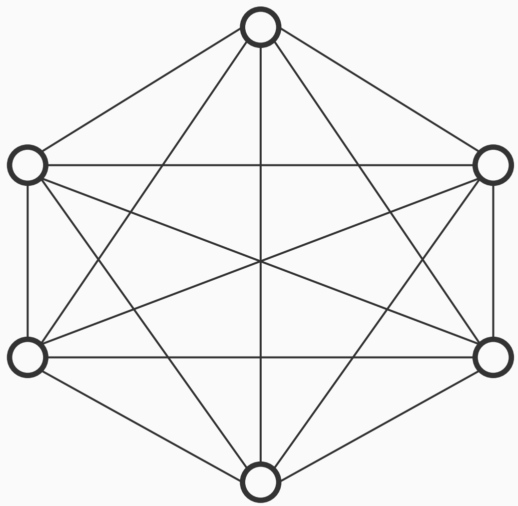
One option: Decompose fully*. All nodes and edges are independently scored*



Could be linear functions

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

Decomposing the output: Example



3 possible node labels 

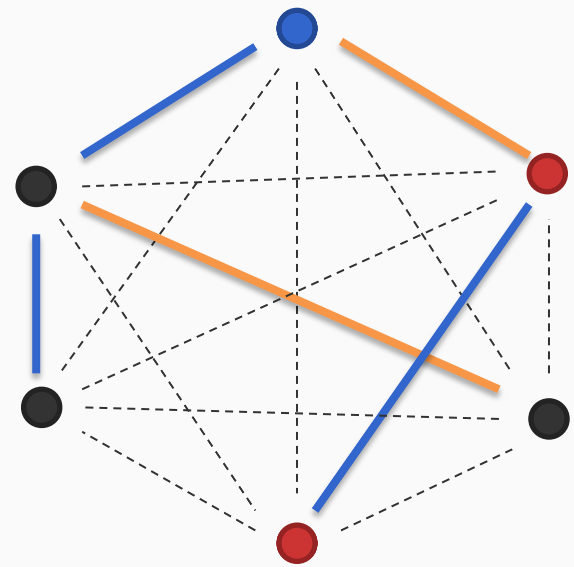
3 possible edge labels 

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*



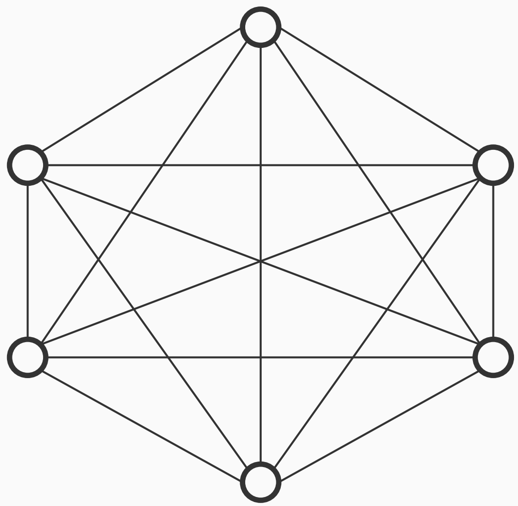
Still need to ensure that the colored edges form a valid output (i.e. a tree)

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

Prediction:

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

Decomposing the output: Example



3 possible node labels

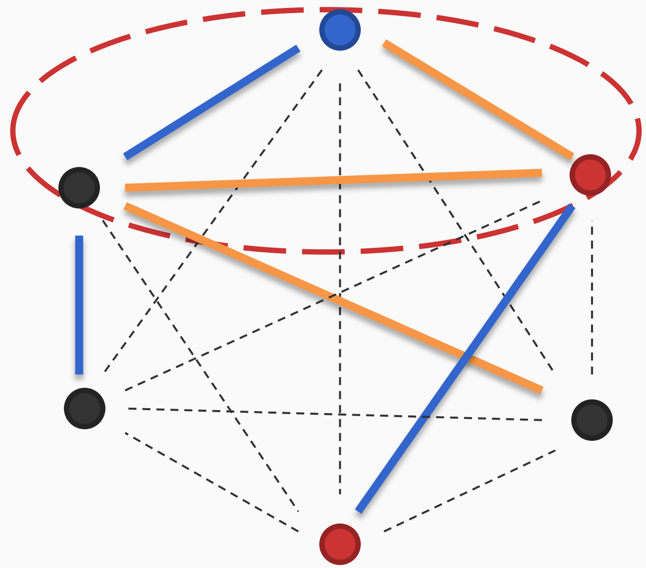
3 possible edge labels

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)

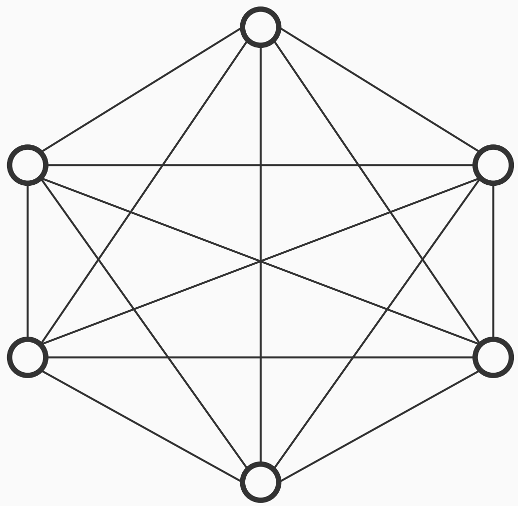
This is invalid output!

Even this simple decomposition requires ***inference*** to ensure validity

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

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

Decomposing the output: Example



Another possibility: Score each edge and its nodes together

3 possible node labels

3 possible edge labels

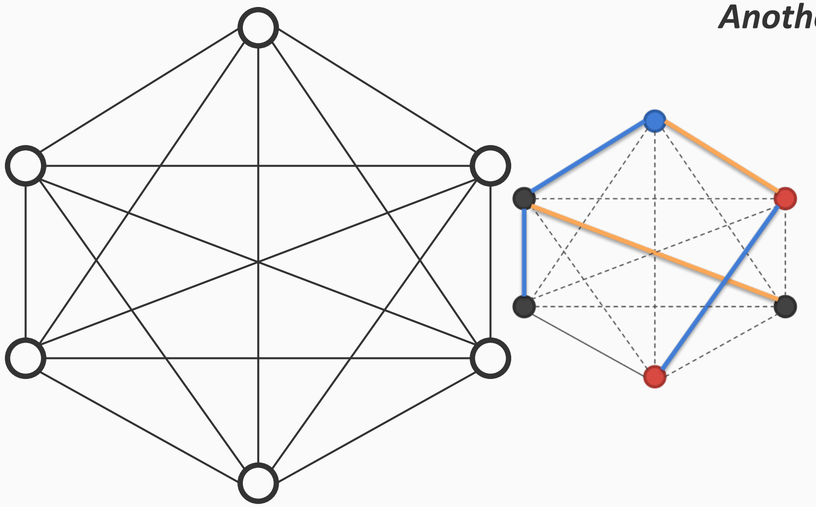
Setting

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 

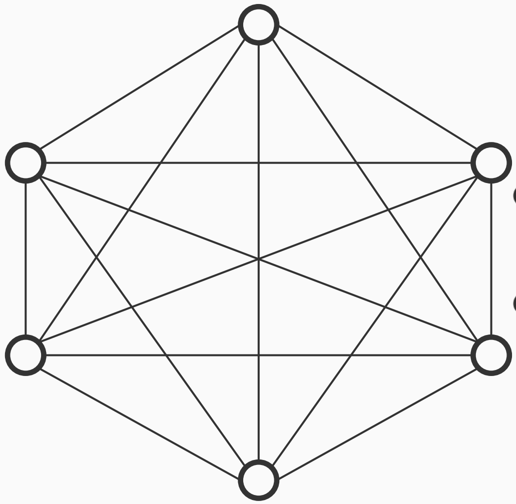
3 possible edge labels 

Setting

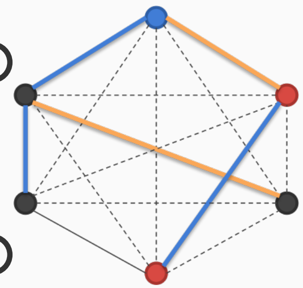
Output: Nodes and edges are labeled and the blue and orange edges form a tree

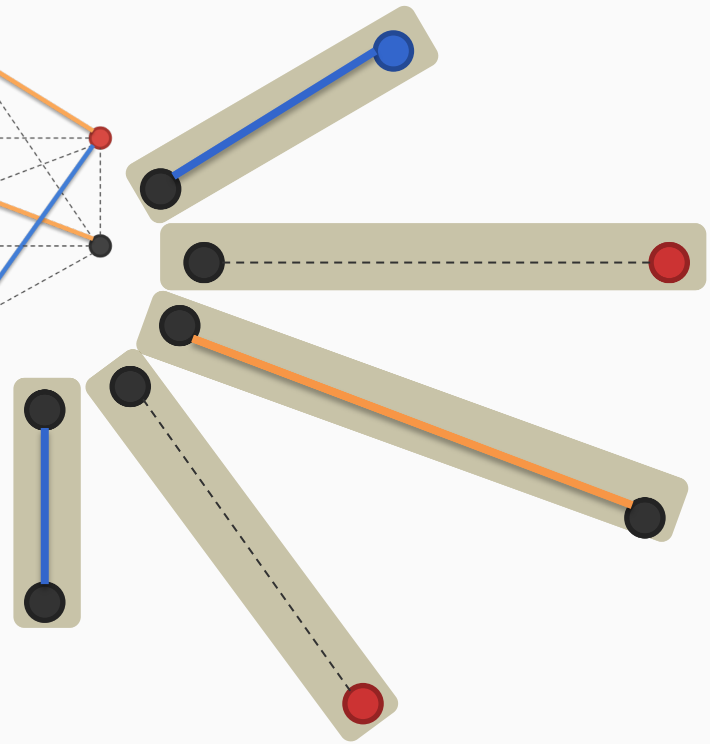
Goal: Find the highest scoring labeling such that the edges that are colored form a tree

Decomposing the output: Example



Another possibility: Score each edge and its nodes together

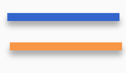




And many other edges…

Each patch represents piece that is scored independently

3 possible node labels 

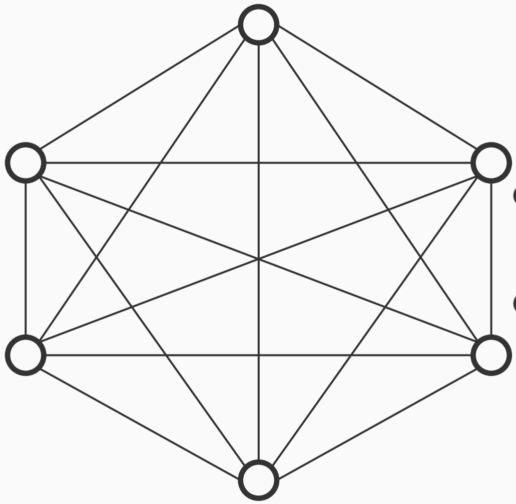
3 possible edge labels 

Setting

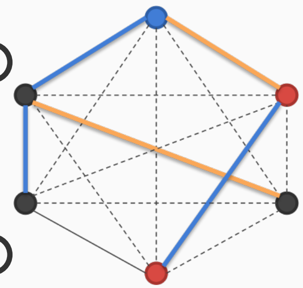
Output: Nodes and edges are labeled and the blue and orange edges form a tree

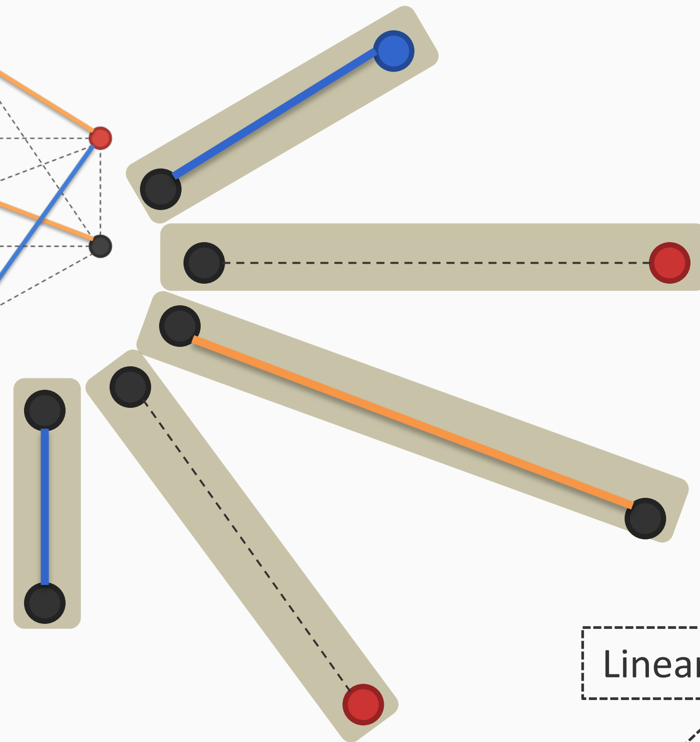
Goal: Find the highest scoring labeling such that the edges that are colored form a tree

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

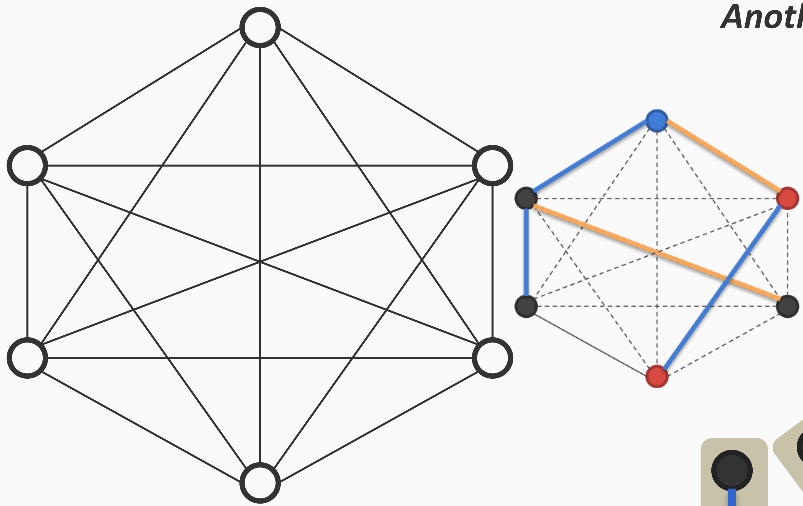
Output: Nodes and edges are labeled and the blue and orange edges form a tree

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

Linear function

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

Decomposing the output: Example



3 possible node labels

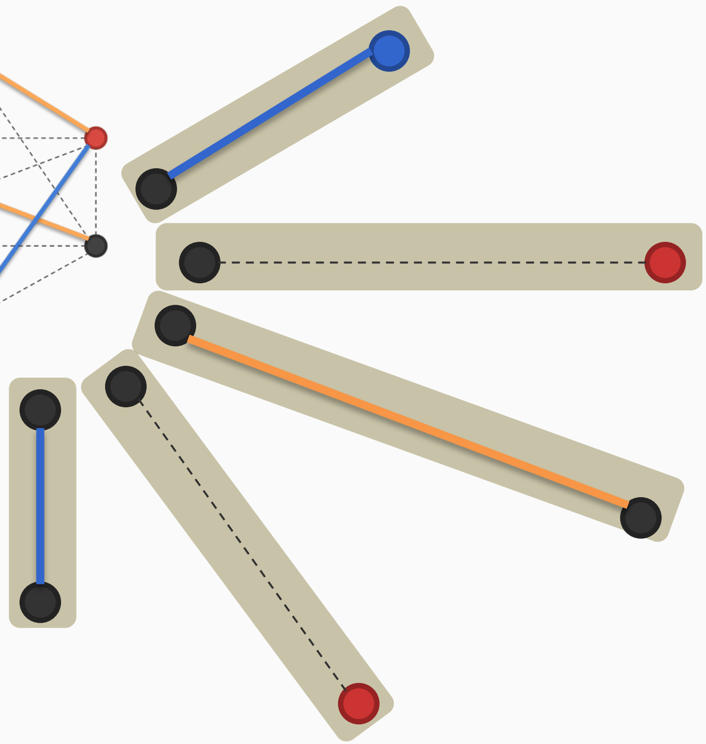
3 possible edge labels

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

Another possibility: Score each edge and its nodes together



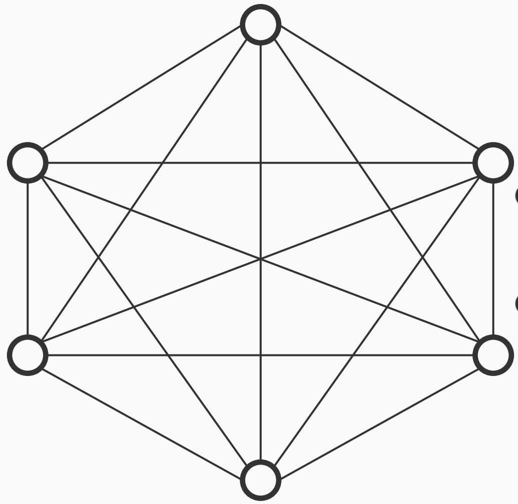
And many other edges…

Each patch represents piece that is scored independently

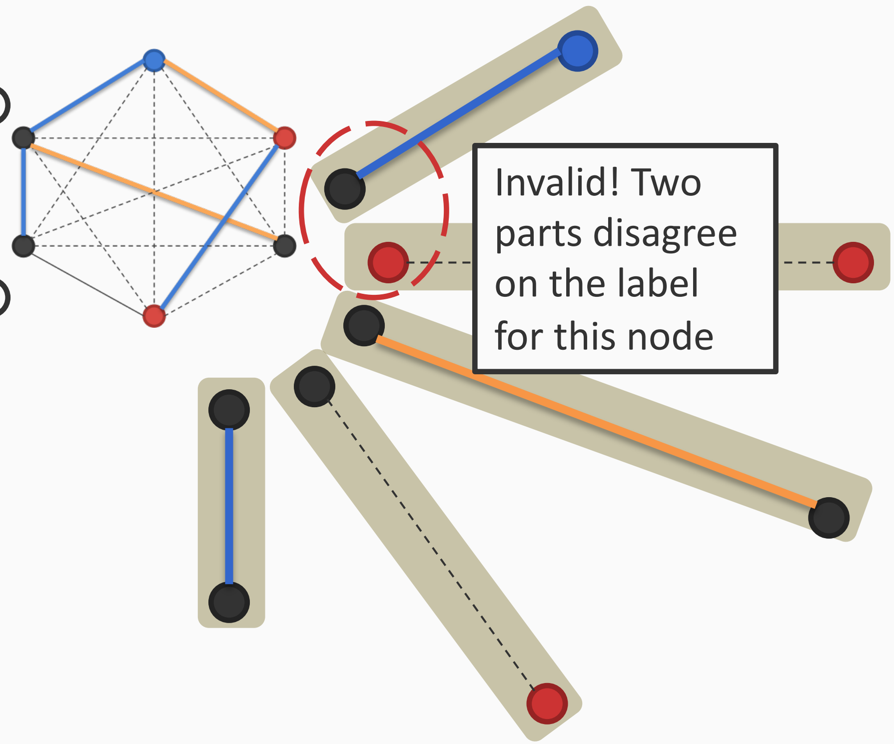
Inference should ensure that

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

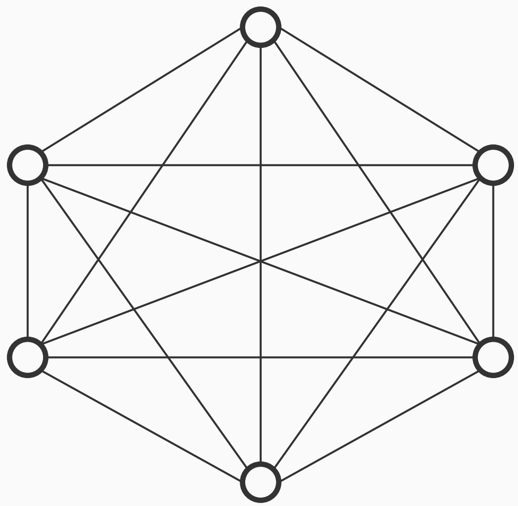
Output: Nodes and edges are labeled and the blue and orange edges form a tree

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



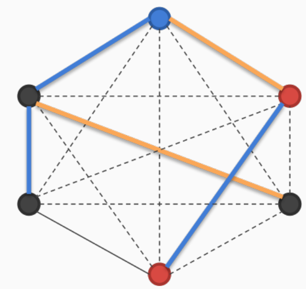
3 possible node labels 

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…

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.
    - We may encounter some examples in more detail:
    - 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?

Training regimes

- Decomposition of outputs gives two approaches for training
  - Decomposed training/Learning without inference
    - Learning algorithm does not use the prediction procedure during training
  - Global training/Joint training/Inference-based training
    - Learning algorithm uses the final prediction procedure during training
- Similar to the two strategies we had before with multiclass
- Inference complexity often an important consideration in choice of modeling and training
    - Especially so if full inference plays a part during training
    - Ease of training smaller/less complex models could give intermediate training strategies between fully decomposed and fully joint

Computational issues

Background **knowledge** about domain

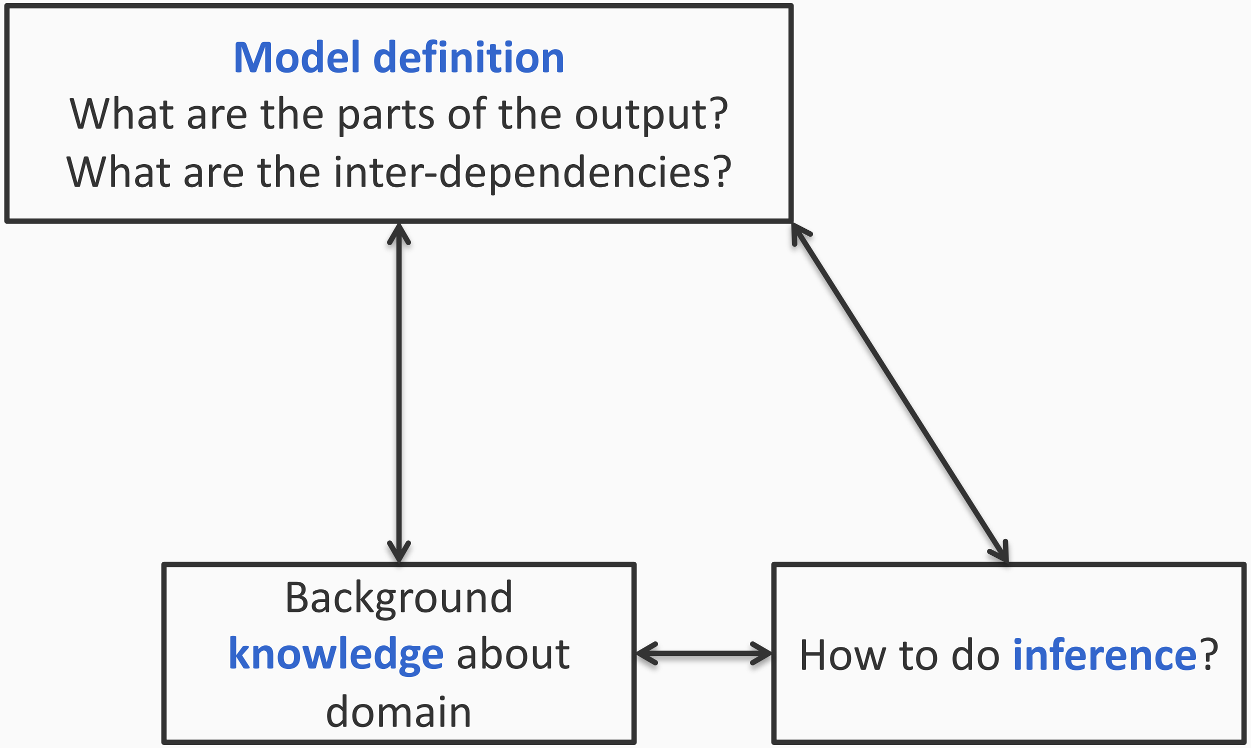
Computational issues

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

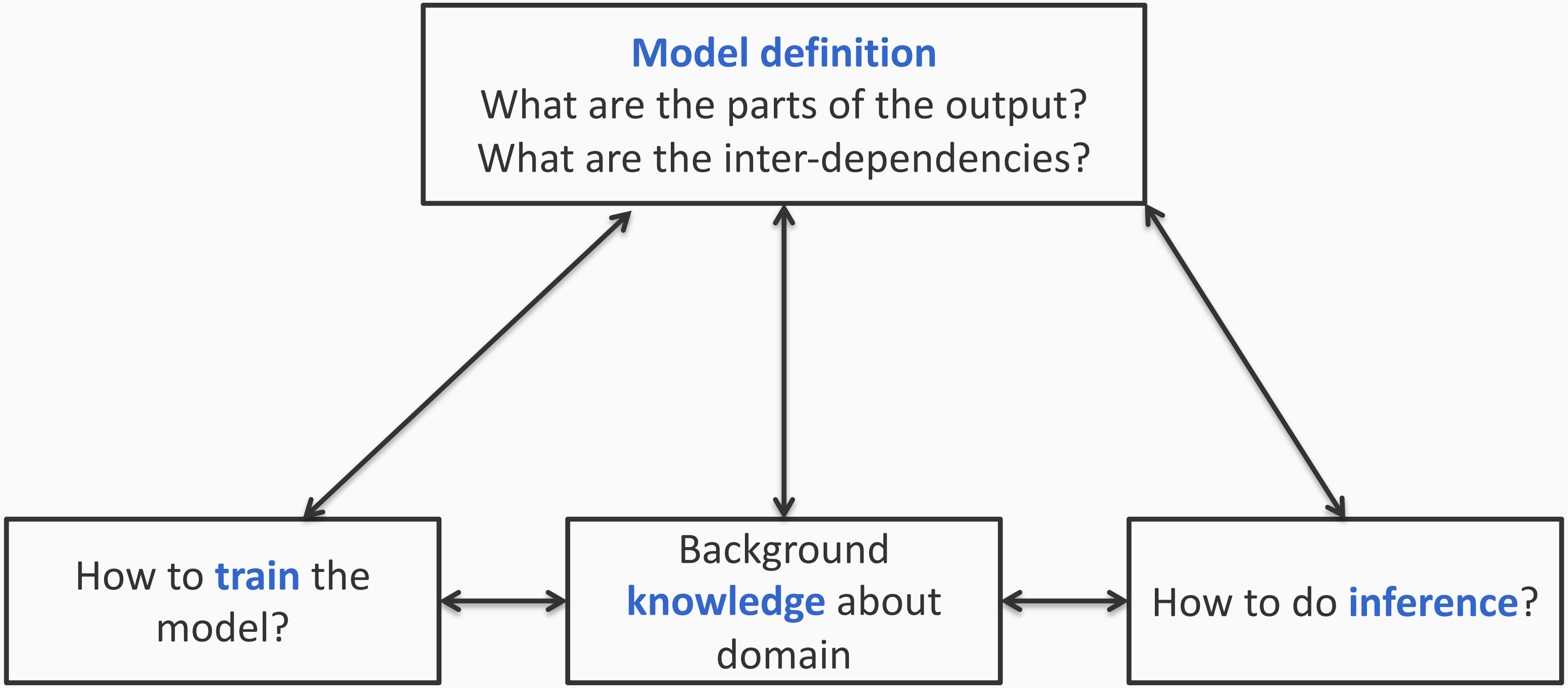
↕

Background
knowledge about
domain

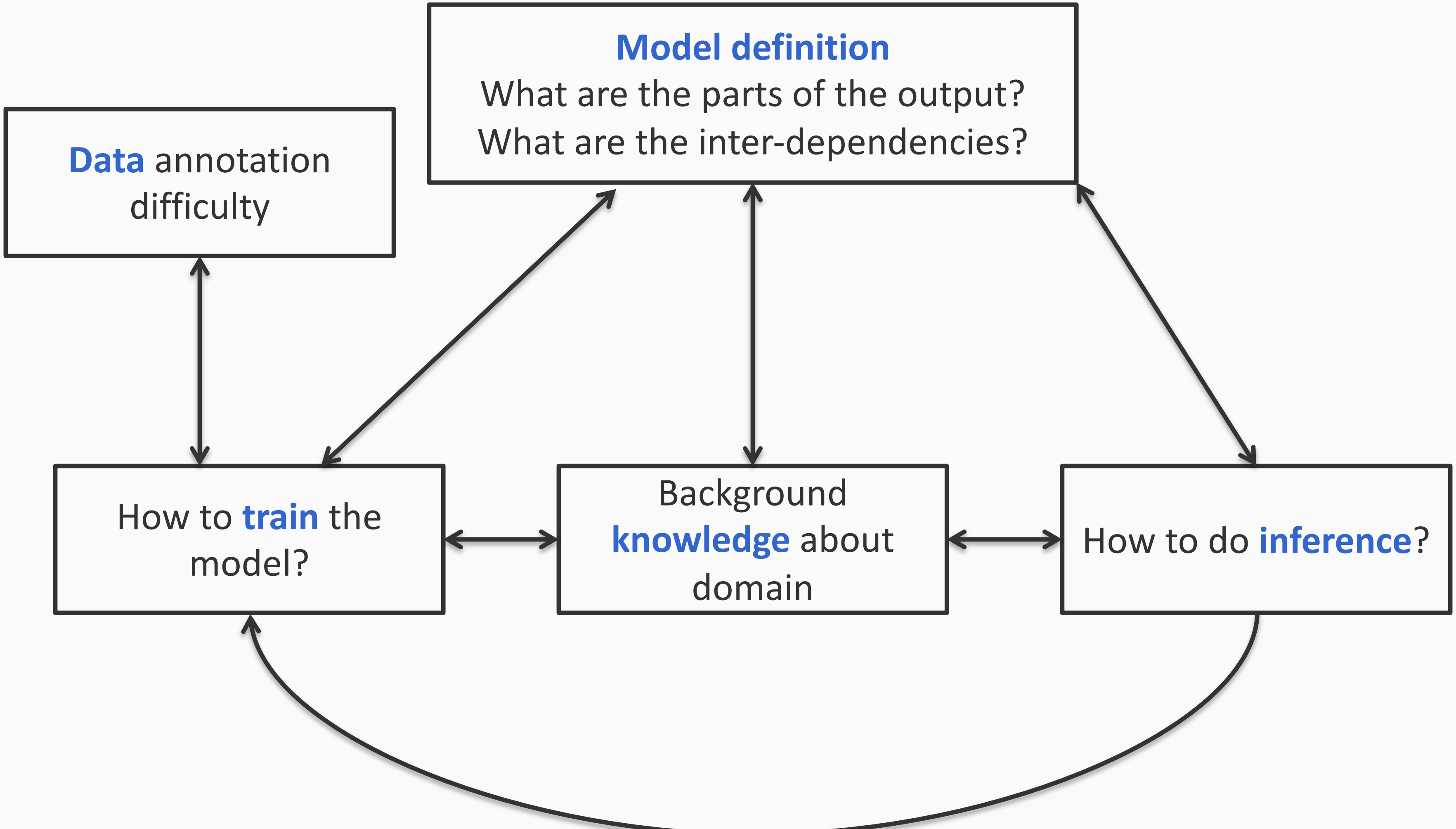
Computational issues



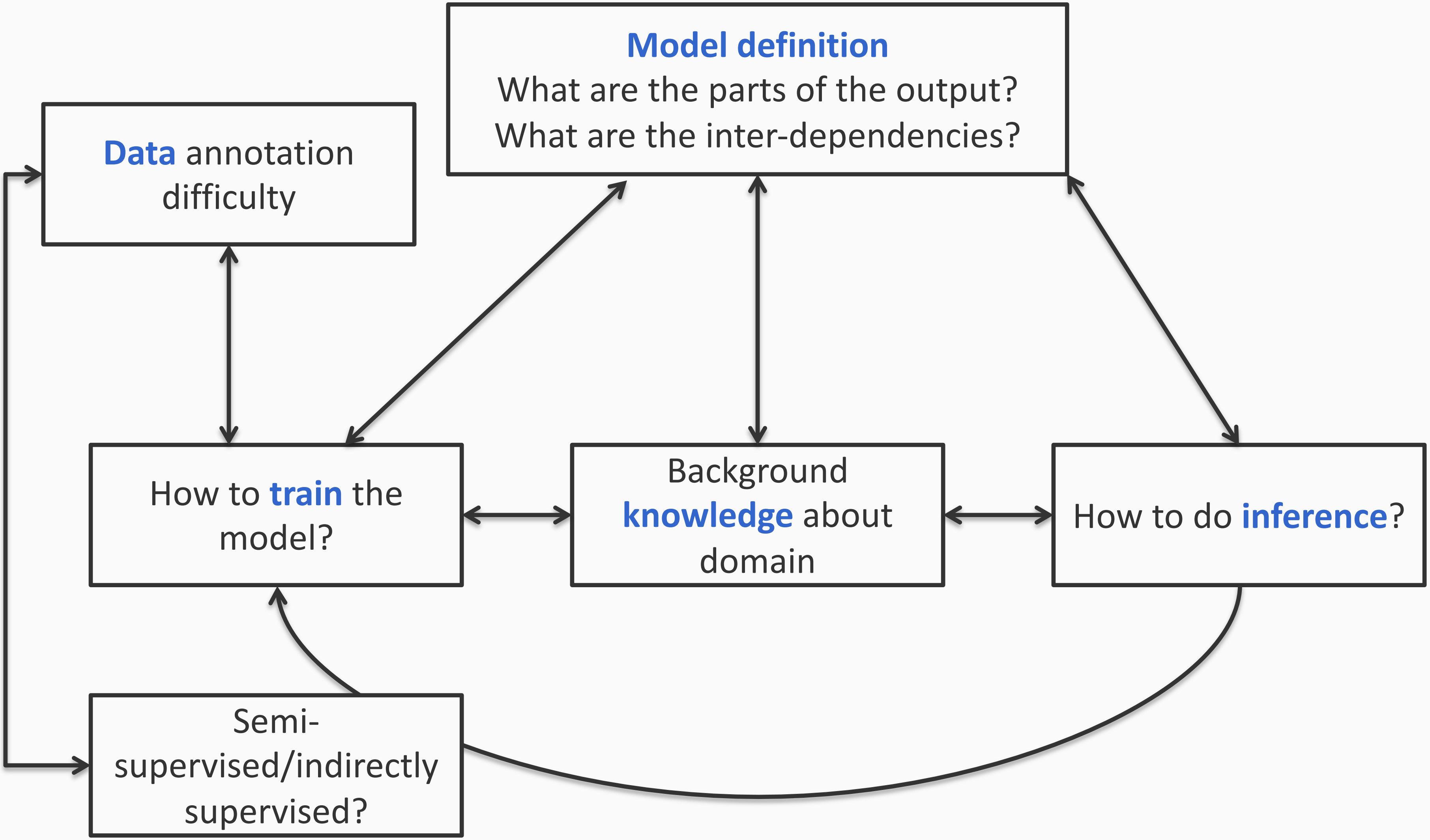
Computational issues



Computational issues



Computational issues



Summary

- We saw several examples of structured output
  - Structures are graphs
    - Sometimes useful to think of them as a sequence of decisions
    - Also useful to think of them as data structures
- Multiclass is the simplest type of structure
  - Lessons from multiclass are useful
- Modeling outputs as structures
  - Decomposition of the output, inference, training

Questions?

Next steps…

- Sequence prediction
  - Markov model
  - Predicting a sequence
    - Viterbi algorithm
  - Training
    - MEMM, CRF, structured perceptron for sequences
- After sequences
  - General representation of probabilistic models
    - Bayes Nets and Markov Random Fields
  - Generalization of global training algorithms to arbitrary conditional models
  - Inference techniques
  - More on Conditional models, constraints on inference