CS 546
Machine Learning in NLP

A First Look on Structures

Dan Roth
Department of Computer Science
University of Illinois at Urbana-Champaign

Augmented and modified by Vivek Srikumar
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**
Last Lecture: 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

• Discussion about structured prediction
Where are we?

- What is structured output?
  - Examples

- Multiclass as a structure

- Discussion about structured prediction
Recipe for multiclass classification

– Collect a training set (hopefully with correct labels)

– Define feature representations for inputs ($\mathbf{x} \in \mathbb{R}^n$)
  • And, $\mathbf{y} \in \{\text{book, penguin, dog}\}$

– Linear functions to score labels

$$\arg \max_{\mathbf{y} \in \{\text{book, dog, penguin}\}} w^T_{\mathbf{y}} \mathbf{x}$$
Recipe for multiclass classification

• Train weight vectors $w_y$ so that it scores correctly
  eg, for an input of type “book”, we want
  $w_{\text{book}}^T x > w_{\text{dog}}^T x$ and $w_{\text{book}}^T x > w_{\text{penguin}}^T x$, and so on...

• Prediction:
  $\arg \max_{y \in \{\text{book, dog, penguin}\}} w_y^T x$

  – 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: Semantic Role Labeling

• Based on the dataset PropBank [Palmer et. al. 05]
  – Large human-annotated corpus of verb semantic relations
• The task: To predict arguments of verbs

Given a sentence, identifies who does what to whom, where and when.

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

```
Predicate
```

```
Arguments
```

```
Relation: Head
```

```
Mover[A0]: the bus
```

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

Predicting verb arguments

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: No overlapping 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”
Inference: verb arguments

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

Total: **2.0**

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

**Special label, meaning “Not an argument”**

**Violates constraint: Overlapping argument!**
The bus was **heading** for Nairobi in Kenya.

**Total:** 2.0

**Total:** 1.9

Special label, meaning “Not an argument”

heading (The bus, for Nairobi in Kenya)
Inference: verb arguments

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

**Input**  \( x \)  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”

\[
y = \{y^1, y^2, \ldots, y^n\} \in \{0, 1\}^n
\]

Total: **1.9**

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

<table>
<thead>
<tr>
<th>Predicate</th>
<th>A0</th>
<th>A1</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head</td>
<td>The bus</td>
<td>Nairobi in Kenya</td>
<td>-</td>
</tr>
</tbody>
</table>

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

Questions/comments?
Example 2: 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

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, etc

Need inference to compose the output
Example 3: 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

<table>
<thead>
<tr>
<th>The</th>
<th>Fed</th>
<th>raises</th>
<th>interest</th>
<th>rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determiner</td>
<td>Noun</td>
<td>Verb</td>
<td>Noun</td>
<td>Noun</td>
</tr>
</tbody>
</table>

Other possible tags 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
    • Eg. 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

One possible model:

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

• Protein 3D structure prediction

• 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
  - \( \arg\max_{y \in \text{all outputs}} \text{score}(x, y) \)

There are a countable number of graphs

**Question:** Why can’t we treat each output as a label and train/predict as multiclass?

We have seen something similar before in the context of 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
  1. 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 labels are scored for each part
     • an inference algorithm to assign labels to all the parts
Where are we?

• What is structured output?

• Multiclass as a structure
  – A very brief digression

• Discussion about structured prediction
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
    \[ \arg\max_{y \in \text{all outputs}} \text{score}(x, y) \]

- Multiclass
  - A graph with one node and no edges
    - Node label is the output
  - Can be composed via multiple decisions
  - Winner-take-all
    \[ \arg\max_{i} w^T \phi(x, i) \]
Multiclass is a structure: Implications

1. A lot of the ideas from multiclass *may* be generalized to structures
   - Not always trivial, 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

• Discussion about structured prediction
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

The scoring function (via the weight vector) 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.

**Setting**

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

**Goal:** Find the highest scoring tree.

The input $x$ not shown here.
Decomposing the output: Example

**Setting**

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

**Goal:** Find the highest scoring 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).

Prediction:

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

\[
\text{arg max}_y \text{ score}(x, y) \text{ s.t. } y \text{ forms a tree}
\]
Decomposing the output: Example

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

Prediction:
\[
\text{arg max}_y \quad \text{score}(x, y) \\
\text{s.t. } y \text{ forms a tree}
\]
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

Linear function

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

Goal: Find the highest scoring tree

3 possible node labels

3 possible edge labels

score\((x, y)\) = \[\sum_{n_1, n_2 \in \text{nodes}(x,y)} \sum_{e \in \text{edges}(x,y)} \text{score}(n_1, n_2, e)\]
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 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

Invalid! Two parts disagree on the label for this node

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 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 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: Reprise

- **Data annotation difficulty**

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

- **Background knowledge about domain**

- **How to train the model?**
  - Different algorithms, joint vs decomposed learning
  - Semi-supervised/indirectly supervised?

- **How to do inference?**
  - Different algorithms, Exact vs approximate inference

- Task dependent, of course

Training and inference concerns drive this too.
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