Neuro-Symbolic Modeling: Overview

A taxonomy of approaches

This lecture

- The Two Systems of Thinking
- Learning & Reasoning
- History: Statistical relation learning
- Some examples of neural-symbolic integration
- Technical challenges for neural-symbolic integration
- A taxonomy of approaches

This lecture

- The Two Systems of Thinking
- Learning & Reasoning
- History: Statistical relation learning
- Some examples of neural-symbolic integration
- Technical challenges for neural-symbolic integration
- A taxonomy of approaches

Approach 1: Just train the neural network

Suppose we have a task where we require that any model satisfies some properties P1, P2,…

For example, the unique core role constraint from semantic role labeling (SRL)

Imagine that we have a *massive* amount of training data, covering a diverse set of domains for the task

Because the training data is correct, presumably the data satisfies the properties.

For example, an SRL dataset satisfies the task constraint, except for annotation errors

Approach 1: Just train the neural network

Suppose we have a task where we require that any model satisfies some properties P1, P2,…

For example, the unique core role constraint from semantic role labeling (SRL)

Imagine that we have a *massive* amount of training data, covering a diverse set of domains for the task

Because the training data is correct, presumably the data satisfies the properties.

For example, an SRL dataset satisfies the task constraint, except for annotation errors

Can we just train a model to fit the data? In doing so, its predictions will satisfy the required properties because the data does

Approach 1: Just train the neural network

Suppose we have a task where we require that any model satisfies some properties P1, P2,…

For example, the unique core role constraint from semantic role labeling (SRL)

Imagine that we have a *massive* amount of training data, covering a diverse set of domains for the task

Because the training data is correct, presumably the data satisfies the properties.

For example, an SRL dataset satisfies the task constraint, except for annotation errors

Can we just train a model to fit the data? In doing so, its predictions will satisfy the required properties because the data does

What are some difficulties with this strategy?

Approach 1 (extension): Data augmentation

Suppose we have a constraint that is not explicitly applicable to the task

Example: the MNIST data doesn't involve addition, so the constraint doesn't apply at all

Approach 1 (extension): Data augmentation

Suppose we have a constraint that is not explicitly applicable to the task

- Example: the MNIST data doesn't involve addition, so the constraint doesn't apply at all

We can create a synthetic dataset where the constraint is satisfied

- For the addition case, create a new dataset with two random images as inputs and their sum as the label. The constraint is trivially satisfied in the data

Approach 1 (extension): Data augmentation

Suppose we have a constraint that is not explicitly applicable to the task

- Example: the MNIST data doesn't involve addition, so the constraint doesn't apply at all

We can create a synthetic dataset where the constraint is satisfied

- For the addition case, create a new dataset with two random images as inputs and their sum as the label. The constraint is trivially satisfied in the data

Now, we can train a model on this new synthetic data. Because the data satisfies the constraint, so will the model.

Approach 1 (extension): Data augmentation

Suppose we have a constraint that is not explicitly applicable to the task

- Example: the MNIST data doesn't involve addition, so the constraint doesn't apply at all

We can create a synthetic dataset where the constraint is satisfied

- For the addition case, create a new dataset with two random images as inputs and their sum as the label. The constraint is trivially satisfied in the data

Now, we can train a model on this new synthetic data. Because the data satisfies the constraint, so will the model.

What are some difficulties with this strategy?

Approach 2: Logic to design networks

Suppose we have a neural network and a constraint that involves a few nodes in the network

Can we somehow re-architect the network so that the resulting architecture (by construction) satisfies the constraint? Or almost satisfies the constraint?

If we could do this, *what are some difficulties with this strategy?*

Approach 3: Logic to design loss functions

Most neural networks are opaque and the only interfaces we have are at the inputs and outputs

This means that most constraints will also about them

Approach 3: Logic to design loss functions

Most neural networks are opaque and the only interfaces we have are at the inputs and outputs

This means that most constraints will also about them

Can we write loss functions about the outputs that encourage the model to satisfy the constraints? Each constraint will be mapped to its own loss

We can then use any learning algorithm/optimizer

Approach 3: Logic to design loss functions

Most neural networks are opaque and the only interfaces we have are at the inputs and outputs

This means that most constraints will also about them

Can we write loss functions about the outputs that encourage the model to satisfy the constraints? Each constraint will be mapped to its own loss

We can then use any learning algorithm/optimizer

What are some difficulties with this strategy?

Approach 4a: Structured inference

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

Approach 4a: 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

Approach 4a: 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

Approach 4a: 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

Approach 4a: 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

What are some difficulties with this strategy?

Approach 4b: 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

What are some difficulties with this strategy?

Approach 5: Opaque symbolic programs

Suppose we have a black-box symbolic program that operates over the predictions of a model

Some Examples:

- The program executes a python program created by a model
- The network calls a search engine to fetch documents on which the network continues working with
- The network informs a game engine to run a simulation, which in turn produces the output

Can the outputs of these black box programs provide supervision to train the network? Can the eventual deployment of the network involve such programs?

Approach 5: Opaque symbolic programs

Suppose we have a black-box symbolic program that operates over the predictions of a model

Some Examples:

- The program executes a python program created by a model
- The network calls a search engine to fetch documents on which the network continues working with
- The network informs a game engine to run a simulation, which in turn produces the output

Can the outputs of these black box programs provide supervision to train the network? Can the eventual deployment of the network involve such programs?

What are some difficulties with this strategy?

Summary

Learning and reasoning are two important aspects of intelligent behavior

- Neuro-symbolic models try to combine both
- There is a history that we can build with: statistical relation learning/structured prediction

Two key challenges with integrating neural and symbolic systems

- Mapping symbols to neural network elements
- Making symbolic systems amenable with differentiable learning

There are different families of approaches to address this integration

- We will be looking at most of these for the rest of the semester