UNIWERSYTET
WARSZAWSKI

Wydział Matematyki, Informatyki i Mechaniki
Uniwersytet Warszawski

Artificial Intelligence in Theorem Proving

(Sztuczna inteligencja w dowodzeniu)

Cezary Kaliszyk

MIMUW, March 2020

Overview

Last Lecture

- Theorem Proving Overview
- Proof Assistants
- Comparison with other tools

Today

- Mini-rehearsal on machine learning
- Machine Learning problems in proving
- Lemma selection
- Statistical methods
- k-nearest neighbours for premise selection

Administration

Only one reminder:

Please remember to subscribe to the chat:

`https://chat.mimuw.edu.pl/channel/sid2020-lecture`

(the subscribe button is unintuitively at the usual send location)

Summary of Last Lecture

- Proofs in mathematics and computer are getting more complex, some of them are beyond humans to verify. This is still rare in mathematics, but when it comes to large software, checking that it matches a specification becomes a large effort.
- This increases the attractivity of proof assistants. Definition:
  - a computer program to assist a mathematician
    - (how: keep track of theories, definitions, assumptions, check individual steps, provide decision procedures)
  - in the production of a proof
  - that is mechanically checked
    - (means: completely verifiable in a formal logical system)
- Most of work with a proof assistant consists of translating a human proof accurately enough and then filling in the gaps to the extent that the proof assistant automation can help.
- What are the places that machine learning can help in this process? This will be the topic of todays lecture.

Recently we had a fast progress in machine learning

What is Machine Learning?

- Tuning a large number of parameters

Algorithms that improve their performance based on data

- Face detection
- Recommender systems
- Speech recognition
- Stock prediction
- Spam detection
- Molecule modeling
- Automated translation
- ...

Tasks related to proofs and reasoning

Tasks involving logical inference

- Natural language question answering [*Sukhbaatar+2015*]
- Knowledge base completion [*Socher+2013*]
- Automated translation [*Wu+2016*]

Games

AlphaGo (Zero) problems similar to proving [*Silver+2016*]

- Node evaluation
- Policy decisions

Problems

Main machine learning problems

- Regression: estimate a parameter (1 function)
- Classification: estimate a class (point in plane, digit)
- Clustering

Correspond to

- Supervised learning
- Unsupervised learning
- ...

AI theorem proving techniques (1/3)

We will separate the AI theorem proving techniques in three levels

High-level AI guidance

- Problems where machine learning helps predict and suggest actions that a human normally selects
- First such problem is premise selection: given a statement that we are trying to prove and a very large library of facts (for example all the mathematical books in existence), find the parts of the knowledge that are most useful. For example if we have a fact that looks like topology, a human can find books on topology in a library. A machine learning algorithm can process thousands of books and facts in them - can it select the most useful ones for our goal?
- Second such problem is tactic selection: again given a conjecture we have a number of known proof techniques. Predicting that a particular statement should be proved by induction, or for example by integration by parts is a task that a machine learning system could be trained on.
- For both these tasks we need a sufficiently good characterization of the goals, proof states, lemmas etc. As such it is necessary to find suitable features and learning techniques. And gather sufficient learning data.
- We often have 5–10 seconds to perform the actual prediction, with more the

AI theorem proving techniques (2/3)

Mid-level AI guidance

- Problems where the learning can select a strategy for an automated prover, for a tactic, choose a heuristic
- Selecting lemmas that can be reused, proposing intermediate lemmas
- Suggesting new conjectures
- We often have 0.1–3 seconds to perform the actual prediction, with more time running a tool for longer will likely do better human will likely do better.

AI theorem proving techniques (3/3)

Low-level AI guidance

- AI can also be used to directly guide the actions of an automated theorem prover
- This means selecting (almost) every inference step by previous knowledge
- Depending on a calculus this may mean hundreds to tens of thousands of inferences per second
- In some ATPs this may be the selection of a clause to expand, in some ATPs this may be the selection of substitutions to apply, in some cases a branch etc.
- In order to make very fast decisions it is necessary to have very good proof-state characterizations and fast relevance: typically minimal statistical methods
- Most recently also random forest predictions, despite slowing the prover down 2-5 times can allow having overall higher performance

All Problems for Machine Learning

- Is my conjecture true?
  - Be careful: "Is $a^n + b^n = c^n$ true?" is a tough one...
- Is a statement is useful?
  - Optionally: For a conjecture
- What are the dependencies of statement? (premise selection)
- Should a theorem be named? How?
- What should the next proof step be?
  - Tactic? Instantiation?
- What new problem is likely to be true?
  - Intermediate statement for a conjecture

Premise selection

In the next part of the lecture we will focus on premise selection, the first problem for which historically machine learning helped.

Intuition

Given:

- set of theorems T (together with proofs)
- conjecture c

Find: minimal subset of T that can be used to prove c

More formally

$$\arg\min_{t \subseteq T} \{|t| \mid t \vdash c\}$$

(or $\emptyset$ if not provable)

Note: implicit assumption on a proving system. ATP in practice.

In machine learning terminology

Multi-label classification

Input: set of samples $\mathbb{S}$, where samples are triples $s, F(s), L(s)$

- s is the sample ID
- $F(s)$ is the set of features of s
- $L(s)$ is the set of labels of s

Output: function f : features $\rightarrow$ labels

Predicts n labels (sorted by relevance) for set of features

Sample features

Sample add_comm ($a + b = b + a$) characterized by:

- F(add_comm) = {"+", "=", "num"}
- L(add_comm) = {num_induct, add_0, add_suc, add_def}

Not exactly the usual machine learning problem

Labels correspond to premises and samples to theorems

- Very often same, a theorem is usually proved so it is both something that can be predicted and is a label of a training example. If a is proved using b and b is proved using c should we predict c for a as well? Should we predict a for a itself? Few learning problems have this issue...

Similar theorems are likely to be useful in the proof

- Also likely to have similar premises

Theorems sharing logical features are similar

- Theorems sharing rare features are very similar

Temporal order

- Recently considered theorems and premises are important
- Also in evaluation

Not exactly for the usual machine learning tools

Needs efficient learning and prediction

- Frequent major data updates: Whenever a user changes some definition or statement does it mean we have to relearn everything?
- Automation cannot wait more than 10 seconds, often less, the user / reasoning can do more in that time...

Multi-label classifier output

- Often asked for thousands of most relevant lemmas (most tools do 3–10)

Easy to get many interesting features

- Complicated feature relations
- Both linguistic techniques and feature space reduction techniques: PCA / LSA / ...?

Premise Selection: A presentation of major methods tried

- Syntactic methods
  - Lemmas that are neighbours of the conjectures using various metrics
  - Most successful: Recursive methods. We will discuss SInE and MePo later
- Naive Bayes classifier and k-Nearest Neighbours (today)
- Linear / Logistic Regression
  - Needs feature and theorem space reduction
  - Kernel-based multi-output ranking
- Decision Trees (Random Forests)
- Neural Networks
  - Winnow, Perceptron
  - Deep learning

[*SNoW*, *MaLARea*]

Adapting Basic Machine Learning Algorithms

Basic k-Nearest Neighbours:

- Given a set of training examples and a sample to evaluate
- Find k nearest neighbors to the sample among the training examples
- Return their average (or weighted average)

kNN Properties

- Very easy to implement
- Almost no learning (build a simple index)

What does this mean for premise selection?

- finds a fixed number (k) of proved facts nearest to the conjecture c
- weight the dependencies each such fact f by the distance between f and c
- relevance is the sum of weights across the k nearest neighbors

k-NN (1/3)

We start with the basic definition of distance between two facts (similarity) characterized by binary features. We then add feature weight and add a scaling factor for these weights:

Definition: Distance of two facts (**similarity**)

$$s(a, b) = \sum_{f \in F(a) \cap F(b)} 1$$

k-NN (1/3)

We start with the basic definition of distance between two facts (similarity) characterized by binary features. We then add feature weight and add a scaling factor for these weights:

Definition: Distance of two facts (**similarity**)

$$s(a,b) = \sum_{f \in F(a) \cap F(b)} w(f)$$

k-NN (1/3)

We start with the basic definition of distance between two facts (similarity) characterized by binary features. We then add feature weight and add a scaling factor for these weights:

Definition: Distance of two facts (**similarity**)

$$s(a,b) = \sum_{f \in F(a) \cap F(b)} w(f)^{\tau_1}$$

k-NN (1/3)

We start with the basic definition of distance between two facts (similarity) characterized by binary features. We then add feature weight and add a scaling factor for these weights:

Definition: Distance of two facts (**similarity**)

$$s(a,b) = \sum_{f \in F(a) \cap F(b)} w(f)^{\tau_1}$$

To estimate the relevance we start with the similarities. We then realize that if a proof relies on more dependencies, each of them is less valuable to we divide by the number of dependencies. We additionally add a factor for the dependencies themselves.

Relevance of fact a for goal g

$$\left(\sum_{b \in N | a \in D(b)} \frac{s(b,g)}{|D(b)|} \right)$$

k-NN (1/3)

We start with the basic definition of distance between two facts (similarity) characterized by binary features. We then add feature weight and add a scaling factor for these weights:

Definition: Distance of two facts (**similarity**)

$$s(a,b) = \sum_{f \in F(a) \cap F(b)} w(f)^{\tau_1}$$

To estimate the relevance we start with the similarities. We then realize that if a proof relies on more dependencies, each of them is less valuable to we divide by the number of dependencies. We additionally add a factor for the dependencies themselves.

Relevance of fact a for goal g

$$\left(\sum_{b \in N | a \in D(b)} \frac{s(b,g)}{|D(b)|} \right) + \begin{cases} s(a,g) & \text{if } a \in N \\ 0 & \text{otherwise} \end{cases}$$

k-NN (1/3)

We start with the basic definition of distance between two facts (similarity) characterized by binary features. We then add feature weight and add a scaling factor for these weights:

Definition: Distance of two facts (**similarity**)

$$s(a,b) = \sum_{f \in F(a) \cap F(b)} w(f)^{\tau_1}$$

To estimate the relevance we start with the similarities. We then realize that if a proof relies on more dependencies, each of them is less valuable to we divide by the number of dependencies. We additionally add a factor for the dependencies themselves.

Relevance of fact a for goal g

$$\left(\tau_2 \sum_{b \in N | a \in D(b)} \frac{s(b,g)}{|D(b)|}\right) + \begin{cases} s(a,g) & \text{if } a \in N \\ 0 & \text{otherwise} \end{cases}$$

k-NN (2/3)

- Ok, we have adapted k-Nearest Neighbors to premise selection, but is this already good enough? What k to choose?
- In 2013 in an experimental evaluation it came out that values of k between 40 and 800 are good and multiple values are complementary.
- But actually the complementarity only says that we need different k for different cases.
- Can we thus choose k automatically?
- On the next slide we have the code for k-NN from multiple state-of-art ITPs, that is currently used in tools. As the core is rather small, you can study it, see how weights are used and how k is adjusted.

k-NN (3/3)

```
let knn_eval csyms (sym_ths, sym_wght) deps maxth no_adv =
  let neighbours = Array.init maxth (fun j -> (j, 0.)) in
  let ans = Array.copy neighbours in

  (* for each symbol, increase the importance of the theorems
     which contain the symbol by a given symbol weight *)
  List.iter (fun sym ->
      let ths = sym_ths sym and weight = sym_wght sym in
      List.iter (fun th ->
          if th < maxth then map_snd neighbours th ((+.) (weight ** 6.0))) ths) csyms;

  Array.fast_sort sortfun neighbours;

  let no_recommends = ref 0 in
  let add_ans k i o =
    if snd (ans.(i)) <= 0. then begin
      incr no_recommends;
      map_snd ans i (fun _ -> float_of_int (age k) +. o))
    end else map_snd ans i ((+.) o) in

  (* Additionally stop when given no_recommends reached *)
  Array.iteri (fun k (nn, o) ->
    add_ans k nn o;
    let ds = deps nn in
    let ol = 2.7 *. o /. (float_of_int (List.length ds)) in
    List.iter (fun d -> if d < maxth then add_ans k d ol) ds;
  ) neighbours;

  Array.fast_sort sortfun ans;
```

Additional Literature (not required)

The first paper has a more detailed description of k-NN.

The second one describes a number of the learning problems in detail on a particular dataset.

Jasmin Christian Blanchette, David Greenaway, Cezary Kaliszyk, Daniel Kühlwein, and Josef Urban.
A learning-based fact selector for Isabelle/HOL.
J. Autom. Reasoning, 57(3):219–244, 2016.

Cezary Kaliszyk, François Chollet, and Christian Szegedy.
Holstep: A machine learning dataset for higher-order logic theorem proving.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Summary

This Lecture

- Machine Learning problems
- Lemma selection
- k-Nearest Neighbours

Next

- other statistical methods
- kernels
- decision trees
- random forests