# Idea for a project: KR for ML

December 19, 2016

**Abstract**

Hypothesis: ML has focuses on solving concrete learning/prediction tasks. In order to harness this technology, it needs to be imbued with a KR philosophy.

## 1 Next steps

1. What is the killer-ap? can combination of ontology and ML help guide ML techniques? what does one need ontologies to do? `http://dl-learner.org/about/ontology-learning/`
2. `http://www.ttic.edu/SNL2017/`
3. Find and engage the best ML people interested in CAV/KR.
4. Do a literature search to find out what has been done at the interface of ML and KR (see Section 5).
   (a) `https://sites.google.com/site/krr2015/home/schedule` was a workshop on symbolic and neural approaches to KRR.
   (b) `http://www1.icsi.berkeley.edu/~shastri/shruti/` has work on modeling logic programs as NN and running the NN to evaluate queries.
   (c) `http://www.ntu.edu.sg/home/asahtan/` has worked on integrating domain knowledge into NNs.
   (d) Lise Getoor: "we need machine learning for graphs"
5. Can one solve synthesis using ML? Yes, genetic programming.
6. The five tribes of ML: `https://www.youtube.com/watch?v=B8J4uefCQMc`
7. `https://en.wikipedia.org/wiki/Markov_logic_network`
8. Pour resoudre des jeux, j'avais parle de Daniel Neider : `https://www.lii.rwth-aachen.de/en/18-mitarbeiter/wissenschaftliche-mitarbeiter/team-prof-grohe/121-publications-of-daniel-neider.html` Celui-la par ex : `https://arxiv.org/abs/1507.05612`

   Sur l'application de l'algo de Angluin dans la verif compositionnelle : Cobleigh, J.M., Giannakopoulou, D., and Pasareanu, C.S. "Learning Assumptions for Compositional Verification", TACAS 2003. Voir ici : `https://ti.arc.nasa.gov/profile/dimitra/`

P Madhusudan fait des choses aussi avec D Neider entre autres : `http://madhu.cs.illinois.edu/pub.html`

9. `http://ti.tuwien.ac.at/cps/teaching/practicals/?id=178` Neurons controlling locomotion.

# Glossary

- NN = artificial neural network
- KR = knowledge representation
- ML = machine learning

# 2 NNs 101

An *activation function* is a function $F : \mathbb{R} \to \mathbb{R}$. Two common activation functions are

$$F_1(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise.} \end{cases}$$

and

$$F_2(z) = \frac{1}{1+e^{-z}}.$$

A *neuron* is a function $G : \mathbb{B}^n \to \mathbb{B}$ where $G(\overline{x}) = F(\overline{x} \cdot \overline{w})$ for some *weights* $w_i \in \mathbb{R}$. Neurons with activation functions $F_1$ are called *perceptrons*, and with activation functions $F_2$ are called *sigmoid neurons*.

A *neural network (NN)* is a DAG whose nodes are neurons. A NN with $N$ inputs and $M$ outputs computes a function $F : \mathbb{B}^N \to \mathbb{R}^M$. One may round the output of $F$ to the nearest integer 0 or 1 to get a function $F : \mathbb{B}^N \to \mathbb{B}^M$.

**Fact 1.** *NN built from perceptrons can compute arbitrary Boolean functions. E.g., $N = 3, M = 1, w_1 = w_2 = 1, w_3 = -1$ determines $x_1 \wedge x_2$ if we fix $x_3 = 1$; and $N = 2, M = 1, w_1 = -2, w_2 = 1$ determines $\neg x_1$ if we fix $x_2 = 1$.*

One might be tempted to consider the synthesis problem is: given a function $G$, find a NN that computes it. The problem is that $G$ is typically not known so one can't know if a NN actually computes $G$. Instead, some tuples $(x, G(x))$ are known. This data can be used as training and validation data, and there are measures of how well a NN represents $G$ (e.g., cross-validation). Thus from a ML point of view, this is a *optimisation* problem which one can think of as minimising some error.

The typical approach in NN seems to be: a) get a bunch of data, b) take an educated guess at the DAG, c) apply an optimisation technique (e.g., gradient descent) to *learn* the weights that minimise the error.

**Remark 1.** *I read that there are no good learning algorithms for NNs using $F_1$ unless the DAG has very special form (e.g., consists of a single neuron).*

# 3 KR for NN

Suppose you've trained a bunch of NN to do certain tasks in the same domain. Are these NNs consistent with each other? Can one compose these NNs to do more complex tasks?

**Example 1.** *Suppose $N_1$ recognises pictures of birds and $N_2$ recognises pictures of animals.*

1. consistency*: check if every input recognised by $N_1$ is also recognised by $N_2$?*

2. composition*: build a NN that recognises all animals that are not birds.*

**Definition 1.** *Given a finite directed graph $(V, E)$ and a NN $N_v$ for every $v \in V$. This setting is* consistent *if $(v, w) \in E$ implies $N_v(\overline{x}) \to N_w(\overline{x})$ for all $\overline{x}$ (intuitively, $E$ represents the "is a" relation).*

**Theorem 1.** *Assume we are using the activation function $F_1$. Consistency is decidable.*

*Proof.* Recall that the FO-theory of $\mathcal{R} := (\mathbb{R}, +, <)$ is decidable, a fact that can be proved using automata on infinite words. But every NN $N(\overline{x})$ can be written as a FO formula $A(\overline{x})$ over $\mathcal{R}$. Thus we can decide consistency, i.e., decide if $\forall \overline{x}. A(\overline{x}) \to B(\overline{x})$. □

Actually, $A(\overline{x})$ is an existential formula, i.e., of the form $\exists \overline{y}. \phi(\overline{x}, \overline{y})$ where $\phi$ is a quantifier free formula in the signature of $\mathcal{R}$. So, consistency can be checked in EXPTIME. Can it be done in PSPACE?

What about composition? We can build a formula for the composition, e.g., $C(\overline{x}) = A(\overline{x}) \wedge \neg B(\overline{x})$. But, how does one turn $C$ into a NN? Does one need to turn it into a NN?

Can we say anything if the activation function is $F_2$?

## 3.1 DL for ML

NN can be used to represent binary relations between objects, e.g., "friend" relation, "like" relation, etc.

So, the description logic $FL^{-1}$ seems like a start for doing basic reasoning about the objects and their relationships. We describe this logic. Given a set of unary predicates $C_i$ and binary predicates $R_j$, a *concept* is generated by the unary predicates, and closed under the following operations: $C_i \cap C_j$, $\{x : \forall y. R_j(x, y) \to C_i(y)\}$, and $\{x : \exists y. R_j(x, y)\}$.

In DL, one basic question is "subsumption", i.e., $\forall x. C(x) \to D(x)$. This is what we called consistency before.

Issue: One problem with subsumption/consistency is that $\forall x$ is probably too strong. Perhaps one needs something like "almost all x". Or, assuming that the data is labeled consistently, what is the error/probability that a data point $x$ is classified as being in $C$ but not in $D$.

# 4 Wild ideas

1. Can FOL deduction be implemented in a NN? See Smolensky 1988 ”On the proper treatment of Connectionism”.

# 5 Literature

#### 5.0.1 Some basic machine terminology

#### 5.0.2 Towell and Shavlik 1994, KR for ANN

#### 5.0.3 Markov Logic Networks