ON-THE-FLY CONFIGURATION OF MACHINE LEARNING SERVICES

On the Evolution of Intelligent Systems Design

Eyke Hüllermeier
Heinz Nixdorf Institute
Department of Computer Science
Paderborn University

eyke@upb.de

LMU Munich, 21-MAR-2018
Essentially based on **machine learning** technology, makes use of deep neural networks and combines different types of learning (supervised, reinforcement, MCTS).

**Deep Blue beats Garry Kasparov (1997)**

Brute force **computing power** (massively parallel system, evaluation of 200 million positions per second), **systematic search**, structured domain.

**Watson wins Jeopardy! (2011)**

Massive information **retrieval** (four terabytes of structured and unstructured content), yet little **reasoning** and **learning**.

**AlphaGo beats Lee Sedol (2016)**
MILESTONES OF AI

Data + learning

AlphaGo beats Lee Sedol (2016)

Knowledge + retrieval

Watson wins Jeopardy! (2011)

Algorithmics + programming

Deep Blue beats Garry Kasparov (1997)
classical programming

function GetMin(var a: TList)
  var
    i, min, mini: integer;
  begin
    min := MaxInt;
    mini := 0;
    for i := 1 to a.len do
      if a.arr[i].G < min
        begin
          min := a.arr[i].G
          mini := i;
        end;
    GetMin := mini;
  end;

mann(adam).
mann(tobias).
mann(frank).
frau(eva).
frau(daniela).
frau(ulrike).
vater(adam,tobias).
vater(tobias,frank).
vater(tobias,ulrike).
mutter(eva,tobias).
mutter(daniela,frank).
mutter(daniela,ulrike).

# Spot Check Algorithms
models = []
models.append('LR', LogisticRegression)
models.append('LDA', LinearDiscriminantAnalysis)
models.append('KNN', KNeighborsClassifier)
models.append('CART', DecisionTreeClassifier)
models.append('NB', GaussianNB)
models.append('SVM', LinearSVC)
results = []
for name, model in models:
  kfold = model_selection.Clf
  cv_results = model_selection.cross_val_score(model, X, y)
  results.append(cv_results)
  names.append(name)
msg = "%s: %f (%f)" % (name, scores.mean(), scores.std())
print(msg)
THE ALGORITHMIC APPROACH

computer scientist

algorithm

start

end
**THE ALGORITHMIC APPROACH**

**algorithm**

**ALGORITHM** shortest-path(V,T)

W := {v1}
ShortDist[v1] :=0

FOR each u in V - {v1}
    ShortDist[u] := T[v1,u]

WHILE W /= V

    MinDist := INFINITE
    FOR each v in V - W
        IF ShortDist[v] < MinDist
            MinDist = ShortDist[v]
            w := v
        END {if}
    END {for}

    W := W U {w}

END {while}

END {for}

W := W U {w}

FOR each u in V - W
    ShortDist[u] := Min(ShortDist[u], ShortDist[w] + T[w,u])

END {while}
The Algorithmic Approach

Requires a comprehensive understanding and adequate formalization, not only of the problem, but also of the solution process.
COMPLEX PROBLEMS

GAME PLAYING

state vector describing the environment

ROBOT SOCCER

action vector

IMAGE RECOGNITION

MALE

AUTONOMOUS CARS

The End of Driving?
A chorus of carmakers has declared that they expect autonomous cars to reach commercial viability by 2020. Computer systems and sensors that handle parking, braking, and to a limited degree, steering are already giving us a glimpse of a future in which machines not only drive unassisted but do so better than any human can. Now the Tesla Motors, maker of the electric luxury sports car that debuted to rave reviews, has tipped the ante. Tesla's CEO, Elon Musk, says that within the next three years, his company aims to produce systems capable of safely taking the helm for 90 percent of miles driven.

technology and science news 19 September 2013
EVOLUTION OF INTELLIGENT SYSTEMS

classical programming

knowledge-based programming

... is difficult for truly complex problems
Knowledge-based systems

- Representation of problem-specific knowledge, such as facts and rules about a domain. „What“ but not „how“!

- Generic control structure implemented by the inference engine.
- Programs = theories of a formal logic, computations = deductions.
- Closely connected to declarative programming languages such as PROLOG.
- Appealing if it‘s difficult to explain HOW the problem is solved.
KNOWLEDGE-BASED SYSTEMS

- Generic control structure implemented by the inference engine.
- programs = theories of a formal logic, computations = deductions.
- Closely connected to declarative programming languages such as PROLOG.
- Appealing if it’s difficult to explain HOW the problem is solved.

Representation of problem-specific knowledge, such as facts and rules about a domain. „What“ but not „how“!
EVOLUTION OF INTELLIGENT SYSTEMS

- **classical programming**
  - is difficult for truly complex problems

- **knowledge-based programming**
  - suffers from knowledge acquisition bottleneck

- **“implicit” programming**

```haskell
function GetMin(var a: TList) var
  i, min, mini: integer;
begin
  min := MaxInt;
  mini := 0;
  for i := 1 to a.len do
    if a.arr[i].G < min then
        min := a.arr[i].G
        mini := i;
    end;
end;
GetMin := mini;
```

```
mann(adam).
mann(tobias).
mann(frank).
frau(eva).
frau(daniela).
frau(ulrike).
vater(adam,tobias).
vater(tobias,frank).
vater(tobias,ulrike).
mutter(eva,tobias).
mutter(daniela,frank).
mutter(daniela,ulrike).
```

```
# Spot Check Algorithms
models = []
models.append('LR', LogisticRegression)
models.append('LDA', LinearDiscriminantAnalysis)
models.append('KNN', KNeighborsClassifier)
models.append('CART', DecisionTreeClassifier)
models.append('SVM', SVC)
models.append('GBM', GradientBoostingClassifier)
models.append('RF', RandomForestClassifier)

# evaluate each model in order
results = []
names = []
for name, model in models:
    kfold = model_selection.
    cv_results = model_selection.
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" %

print(msg)
```
Human skills are not always easy to explain!

\[ x \in \mathbb{R}^N \]

\[ y \in \{0, 1\} \]
IMPLICIT SKILLS

**Human skills are not always easy to explain!**

For example, a reduction of the search space does not immediately imply better solutions.

Eine Beschränkung des Suchraums führt beispielsweise nicht unmittelbar zu besseren Lösungen.
**Implicit Skills**

**Human skills are not always easy to explain!**

---

**Abstract**

Given a sample of instances with binary labels, the top ranking problem is to produce a ranked list of instances where the head of the list is dominated by positives. Popular existing approaches to this problem are based on surrogates to a performance measure known as the fraction of positives of the top (PTop). In this paper, we show that the measure and its surrogates have an undesirable property: for certain noisy distributions, it is optimal to trivially predict the same score for all instances. We propose a simple rectification of the measure which avoids such trivial solutions, while still focussing on the head of the ranked list and being as easy to optimise.
Instead of providing a complete and consistent description of domain knowledge, or designing a model by hand, it is easier to ...

- give **examples** and let the system **generalize**
- let the system **explore** and provide **feedback**
- demonstrate and let the system **imitate**

→ supervised learning
→ reinforcement learning
→ imitation learning
“Machine learning is the science of getting computers to act without being explicitly programmed.”

Andrew Ng, 2013
LEARNING FROM DATA

computer scientist

DATA ➔ LEARNER ➔ \( f \) ➔ \( y \)
Learning does not mean turning data into knowledge, but revising prior knowledge in the light of observed data.
FEATURE ENGINEERING

symmetry

intensity

\[ x \rightarrow [x^T w > \theta] \]

9

3
FEATURE ENGINEERING
AUTOMATED MACHINE LEARNING

function GetMin(var a: TList)
var
  i, min, mini: integer;
begin
  min := MaxInt;
  mini := 0;
  for i := 1 to a.len do
    if a.arr[i].G < min then
      min := a.arr[i].G;
      mini := i;
  end;
  GetMin := mini;
end;

mann(adam).
mann(tobias).
mann(frank).
frau(eva).
frau(daniela).
frau(ulrike).
vater(adam,tobias).
vater(tobias,frank).
vater(tobias,ulrike).
mutter(eva,tobias).
mutter(daniela,frank).
mutter(daniela,ulrike).

# Spot Check Algorithms
models = []
models.append('LR'). Logistic models.append('LDA'). Linear models.append('KNN'). KNearest
models.append('CART'). Decision models.append('NB'). Gaussian models.append('SVM'). SV
# evaluate each model in turn
results = []
for name, model in models:
  kfold = model_selection
  cv_results = model_selection
  results.append(cv_resu
  names.append(name)
  msg = "%s: %f (±%f)"
  print(msg)

classical programming
knowledge-based programming
“implicit” programming
automated machine learning

... is difficult for truly complex problems
... suffers from knowledge acquisition bottleneck
... still requires a lot of ML expertise
The computer/ML/data scientist is not supposed to solve the actual problem (provide an algorithm) but the problem to learn how to solve that problem.

That’s not necessarily an easy task either …
ML Paradigms

- Active learning
  and experiment design
- Cost-sensitive learning
- Inverse reinforcement learning
- Meta learning
- Multi-task learning
- Online learning
- Reinforcement learning
- Semi-supervised learning
- Transductive learning
- Structured output prediction
- Transfer learning
- ...

ML Methodologies

- Deep learning
- Gaussian processes
- Graphical models
  and Bayesian networks
- Inductive logic programming
- Kernel-based methods
  and support vector machines
- Latent variable and topic models
- Markov networks
- Preference learning and ranking
- Relational learning
- Rule and decision tree learning
- Sparsity and compressed sensing
- ...

ML PARADIGMS AND METHODOLOGIES
Objective of the learning problem
- specify the type of problem and prediction task to be solved
- success criteria (accuracy/loss function, model complexity, ...)
- ...

Specifying the model induction problem
- feature description
- kernel functions
- ...

Solving the model induction problem
- preprocessing, including feature selection, normalization, etc.
- model selection, choice of the model class
- choice of the learning algorithm
- estimation of generalization performance (e.g., cross-validation)
- tuning of hyper-parameters
- interpreting and reacting to feedback gathered from experiments
- postprocessing of models
- ...
Example of an ML pipeline for image classification
A deep (convolutional) neural net (determining network structure and training) may have more than 40 hyper-parameters:

- number of hidden units
- activation function
- convolution kernel width
- implicit zero padding
- weight decay coefficient
- loss function
- weight initialization
- learning rate
- batch size
- dropout rate
- ...

*When solving a practical problem, an ML scientist explicitly or implicitly fixes thousands of degrees of freedom …*
Several *AutoML* tools already exist (Auto-WEKA, auto-sklearn, TPOT, RECIPE, RapidMiner, …).

Essentially, these tools realize a systematic search in the space of ML pipelines, assessing each candidate in terms of an estimated performance.
On-the-Fly (OTF) Computing is a novel computing paradigm that aims at the provision of individually configured software services in a market environment that comprises so-called OTF providers, service providers, and end-users as main participants.

The service requested by an end-user is automatically constructed by an OTF provider in an on-the-fly manner, and then executed in an OTF compute center.

The OTF provider relies on existing services made available by service providers, which are freely traded on a global service market and flexibly combined in the course of a service composition process.
ON-THE-FLY COMPUTING

Clients

Service Provider

Provider
OTF Compute Center
OTF Software Provider

Technical and economic market infrastructure

Provision of IT-Services

Organization of the market
Functionality specified in terms of pre- and postconditions (logical predicates) \( \langle P_{pre}, P_{post} \rangle \)
OTF COMPUTING AND ML

Automated composition of ML services

Improving efficiency and quality of service composition through ML
On-the-Fly Machine Learning (OTF-ML) as an instantiation of OTF computing: On-the-fly selection, configuration, provision, and execution of machine learning and data analytics functionality.
OTF-ML SCENARIOS

TASK

TRAIN 0 1 1

TEST ? ? ?

OTF-ML

TEST 1 0 1

predictions

f : X → Y

predictor

customized software
Existing approaches optimize parameters of a **fixed ML pipeline**. The parameter space is structured, each “point” defines algorithm selection (model) and configuration (hyper-parameters). Essentially restricted to (binary) classification. No backtracking (e.g., due to overall insufficient quality) and no user interaction.
Hierarchical planning (Hierarchical Task Networks, HTN) as a more flexible and expressive formalism to create ML pipelines.

Recursive reduction of complex tasks to (complex or simple) subtasks.
- Algorithmically solved using **graph search** algorithms.
- A node is a **goal node** if all remaining tasks are simple.
- HTN via **forward-decomposition**: one successor is created for each possible decomposition of the first unsolved task in the list of remaining tasks.
ML-Plan implements **best-first search** with node evaluation.

- Problem: cost of a solution (e.g., expected loss of a classifier) cannot be computed from the descriptions of the plan elements.
- Default node evaluation based on **random path completion** as also used in Monte Carlo Tree Search, combined optimistically (minimum).
- Specific strategy to prevent **over-fitting**.
Pipeline created for the page-blocks benchmark data
**Summary & Conclusion**

Data is extremely useful, and its increased availability enables AI applications beyond reach so far.

Yet, we cannot get rid of knowledge, nor of algorithms: Knowledge is needed to make sense of data, and algorithms to exploit it.

With the trend toward data-driven design of systems, the knowledge required becomes more abstract, and algorithms more generic.
SUMMARY & CONCLUSION

ALGORITHM

ML ALGORITHM

AutoML ALGORITHM

MetaML ALGORITHM

{MALE, FEMALE}

algorithm (predictor)

ML algorithm (learner)

AutoML algorithm

data

ML problem (data)

ML problem (data)
SUMMARY & CONCLUSION

Instruct the computer how to solve the problem

Instruct the computer how to learn how to solve the problem

Instruct the computer how to find a good way to learn how to solve the problem

{MALE, FEMALE}

algorithm (predictor)

ML algorithm (learner)

AutoML algorithm

data

ML problem (data)

ML problem (data)
The “ML as a service” idea comes with a number of interesting new challenges, both scientifically and from an application point of view.