UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

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

MILESTONES OF AI

INTELLIGENT SYSTEMS

*Essentially based on **machine learning** technology, makes use of deep neural networks and combines different types of learning (supervised, reinforcement, MCTS)*

AlphaGo beats Lee Sedol (2016)

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

Watson wins Jeopardy! (2011)

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

Deep Blue beats Garry Kasparov (1997)

MILESTONES OF AI

Data
+ learning

AlphaGo beats Lee Sedol (2016)

Knowledge
+ retrieval

Watson wins Jeopardy! (2011)

Algorithmics
+ programming

Deep Blue beats Garry Kasparov (1997)

```
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 t
        begin
            min := a.arr[i].G
            mini := i;
        end;

    GetMin := mini;
end;
```

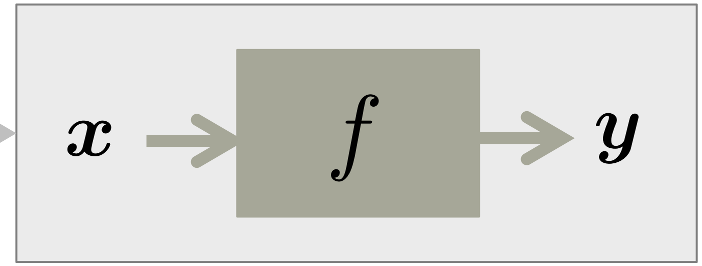
```
mann(adam).
mann(tobias).
mann(frank).
frau(eva).
frau(daniela).
frau(ulrike).
vater(adam,tobias).
vater(tobias,frank).
vater(tobias,ulrike).
mutter(eva,tobias).
mutter(daniela,frank).
mutter(daniela,ulrike).
```

```
# Spot Check Algorithms
models = []
models.append(('LR', Logis
models.append(('LDA', Line
models.append(('KNN', KNei
models.append(('CART', Dec
models.append(('NB', Gauss
models.append(('SVM', SVC(
# evaluate each model in t
results = []
names = []
for name, model in models:
    kfold = model_selectio
    cv_results = model_sel
    results.append(cv_resu
    names.append(name)
    msg = "%s: %f (%f)" %
    print(msg)
```

classical programming

INTELLIGENT SYSTEMS

computer scientist



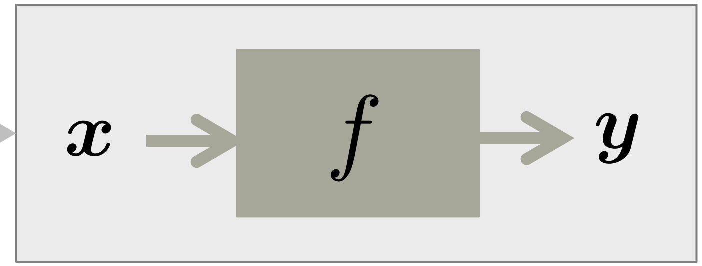
algorithm

start

end

THE ALGORITHMIC APPROACH

computer scientist

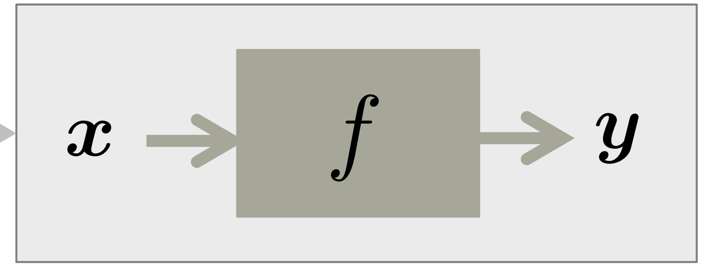


algorithm

```
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}
       FOR each u in V - W
           ShortDist[u] := Min(ShorDis[u],ShortDist[w] + T[w,u])
 END {while}
```

THE ALGORITHMIC APPROACH

computer scientist

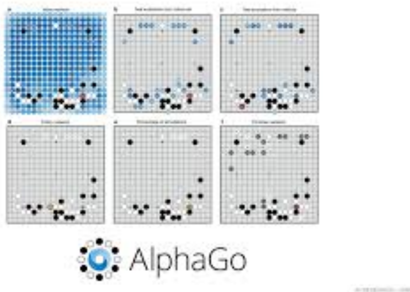


algorithm

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

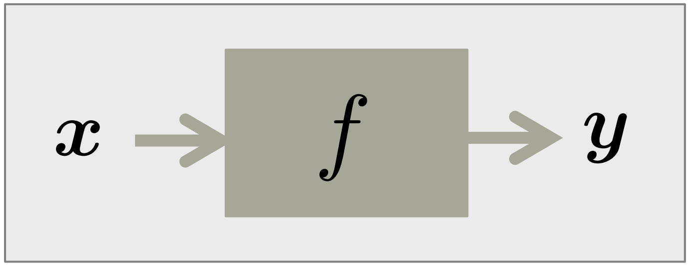
COMPLEX PROBLEMS

GAME PLAYING



ROBOT SOCCER

state vector describing the environment



action vector

MALE

technology and science news

19 September 2013

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 Tesla Motors, maker of the eponymous electric luxury sports car that debuted to rave reviews, has upped 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.

IMAGE RECOGNITION

AUTONOMOUS CARS

EVOLUTION OF INTELLIGENT SYSTEMS


```
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 t
        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).
```

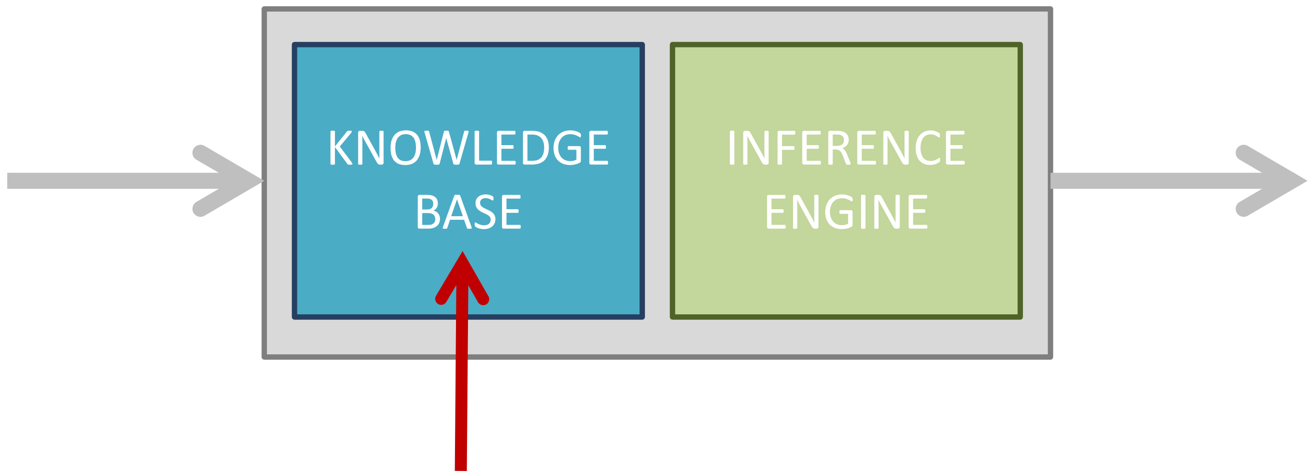
```
# Spot Check Algorithms
models = []
models.append(('LR', Logis
models.append(('LDA', Line
models.append(('KNN', KNei
models.append(('CART', Dec
models.append(('NB', Gauss
models.append(('SVM', SVC(
# evaluate each model in t
results = []
names = []
for name, model in models:
    kfold = model_selectio
    cv_results = model_sel
    results.append(cv_resu
    names.append(name)
    msg = "%s: %f (%f)" %
    print(msg)
```

classical programming

knowledge-based programming

… is difficult for truly complex problems

KNOWLEDGE-BASED SYSTEMS

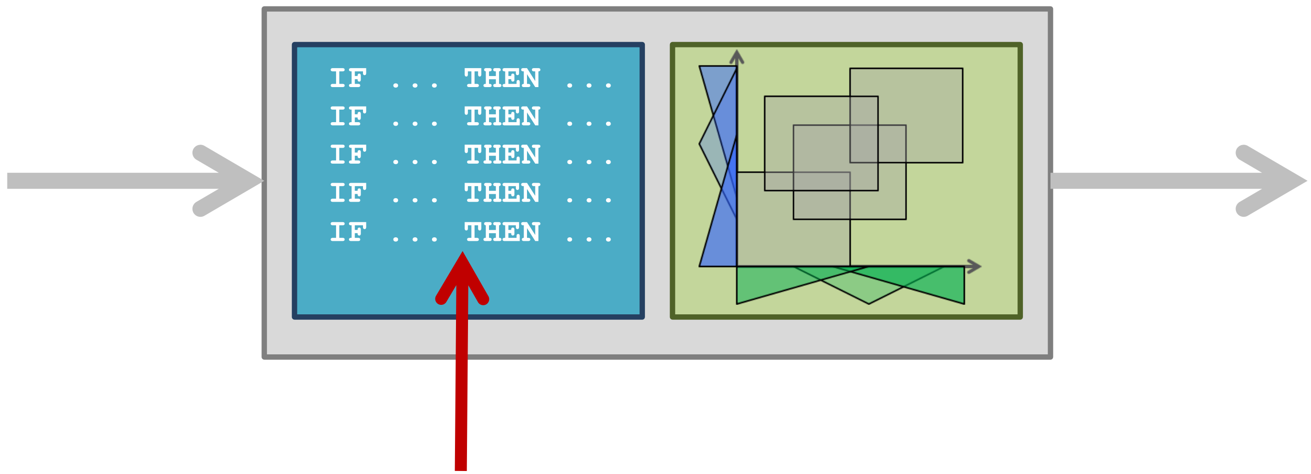


EXPERT

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



EXPERT

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.

EVOLUTION OF INTELLIGENT SYSTEMS

INTELLIGENT SYSTEMS

```
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 t
        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', Logis
models.append(('LDA', Line
models.append(('KNN', KNei
models.append(('CART', Dec
models.append(('NB', Gauss
models.append(('SVM', SVC(
# evaluate each model in t
results = []
names = []
for name, model in models:
    kfold = model_selectio
    cv_results = model_sel
    results.append(cv_resu
    names.append(name)
    msg = "%s: %f (%f)" %
    print(msg)
```

classical programming

knowledge-based programming

"implicit" programming

… is difficult for truly complex problems

… suffers from knowledge acquisition bottleneck

IMPLICIT SKILLS

Human skills are not always easy to explain!

$\longrightarrow \boxed{f} \longrightarrow$ MALE OR FEMALE

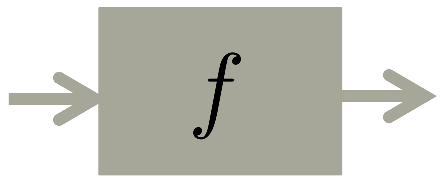
$\boldsymbol{x} \in \mathbb{R}^N$

$\boldsymbol{y} \in \{0, 1\}$

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!

Optimal Sample Complexity of M-wise Data for Top-K Ranking

Algorithm 1 Rank Centrality (Negahban et al., 2012)

Input the collection of statistics $\boldsymbol{s} = \left\{ s_{\mathcal{I}} : \mathcal{I} \in \mathcal{E}^{(M)} \right\}$.
Convert the M-wise sample for each hyper-edge $\mathcal{I}$ into M pairwise samples:

1. Choose a circular permutation of the items in $\mathcal{I}$ uniformly at random,
2. Break it into the M pairs of adjacent items, and denote the set of pairs by $\phi(\mathcal{I})$,
3. Use the (pairwise) data of the pairs in $\phi(\mathcal{I})$.

Compute the transition matrix $\hat{\boldsymbol{P}} = [\hat{P}_{ij}]_{1 \le i,j \le n}$:

$$\hat{P}_{ij} = \begin{cases} \frac{1}{2d_{\max}} y_{ij} & \text{if } i \neq j; \\ 1 - \sum_{k:k \neq j} \hat{P}_{kj} & \text{if } i = j; \\ 0 & \text{otherwise,} \end{cases}$$

where $d_{\max}$ is the maximum out-degree of vertices in $\mathcal{E}$.
Output the stationary distribution of matrix $\hat{\boldsymbol{P}}$.

$$y_{ij} := \sum_{\mathcal{I}:\{i,j\} \in \phi(\mathcal{I})} \frac{1}{L} \sum_{\ell=1}^{L} y_{ij,\mathcal{I}}^{(\ell)}. \tag{16}$$

In an ideal scenario where we obtain an infinite number of samples per M-wise comparison, i.e., $L \to \infty$, sufficient statistics $\frac{1}{L}\sum_{l=1}^{L} y_{ij,\mathcal{I}}^{(\ell)}$ converge to $\frac{w_i}{w_i + w_j}$, as the PL model is a natural generalized version of the BTL model. Then, the constructed matrix $\hat{\boldsymbol{P}}$ defined in Algorithm 1 becomes a matrix $\boldsymbol{P}$ whose entries $[P_{ij}]_{1 \le i,j \le n}$ are defined as

$$P_{ij} = \begin{cases} \frac{1}{2d_{\max}} \sum_{\mathcal{I}:\{i,j\} \in \phi(\mathcal{I})} \frac{w_i}{w_i + w_j} & \text{for } \mathcal{I} \in \mathcal{E}^{(M)}; \\ 1 - \sum_{k:k \neq j} P_{kj} & \text{if } i = j; \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

The entries for observed item pairs represent the relative likelihood of item i being preferred over item j. Intuitively, random walks of $\boldsymbol{P}$ in the long run visit some states more often, if they have been preferred over other frequently-visited states and/or preferred over many other states.

The random walks are reversible as $w_i P_{ji} = w_j P_{ij}$ holds, and irreducible under the connectivity assumption. Once we obtain the unique stationary distribution, it is equal to $\boldsymbol{w} = \{w_1, \ldots, w_n\}$ up to some constant scaling.

It is clear that random walks of $\hat{\boldsymbol{P}}$, a noisy version of $\boldsymbol{P}$, will give us an approximation of $\boldsymbol{w}$. The algorithm adopts a power method, known to be computationally efficient in obtaining the leading eigenvalue of a sparse matrix (Meirovitch, 1997), to obtain the stationary distribution.

3.2. Proof outline

To outline the proof of Theorem 2, let us introduce Theorem 3. We show that Theorem 3 leads to Theorem 2.

Theorem 3. *When* Rank Centrality *is employed, with high probability, the ℓ_∞ norm estimation error is upper-bounded by*

$$\frac{\|\hat{\boldsymbol{w}} - \boldsymbol{w}\|_\infty}{\|\boldsymbol{w}\|_\infty} \lesssim \sqrt{\frac{n \log n}{\binom{n}{M} pL}} \sqrt{\frac{1}{M}}, \tag{18}$$

where $p \ge c_1 (M-1) \sqrt{\frac{\log n}{\binom{n-1}{M-1}}}$, and c_1 is some numerical constant.

Let $\|\boldsymbol{w}\|_\infty = w_{\max} = 1$ for ease of demonstration. Suppose $\Delta_K = w_K - w_{K+1} \gtrsim \sqrt{\frac{\log n}{\binom{n}{M} pL}} \sqrt{\frac{1}{M}}$. Then,

$$\begin{aligned} \hat{w}_i - \hat{w}_j &\ge w_i - w_j - |\hat{w}_i - w_i| - |\hat{w}_j - w_j| \\ &\ge w_K - w_{K+1} - 2\|\hat{\boldsymbol{w}} - \boldsymbol{w}\|_\infty > 0, \end{aligned} \tag{19}$$

for all $1 \le i \le K$ and $j \ge K+1$. That is, the top-K items are identified as desired. Hence, as long as $\Delta_K \gtrsim \sqrt{\frac{\log n}{\binom{n}{M} pL}} \sqrt{\frac{1}{M}}$, i.e., $\binom{n}{M} pL \gtrsim \frac{n \log n}{\Delta_K^2} \frac{1}{M}$, reliable top-$K$ ranking is achieved with the sample size of $\frac{n \log n}{\Delta_K^2} \frac{1}{M}$.

Now, let us prove Theorem 3. To find an ℓ_∞ error bound, we first derive an upper bound on the point-wise error between the score estimate of item i and its true score, which consists of three terms:

$$\begin{aligned} |\hat{w}_i - w_i| \le |\hat{w}_i - w_i| \hat{P}_{ii} &+ \sum_{j:j \neq i} |\hat{w}_j - w_j| \hat{P}_{ij} \\ &+ \left| \sum_{j:j \neq i} (w_i + w_j) \left(\hat{P}_{ji} - P_{ji} \right) \right|. \end{aligned} \tag{20}$$

This can be obtained applying $\hat{\boldsymbol{w}} = \hat{\boldsymbol{P}} \hat{\boldsymbol{w}}$ and $\boldsymbol{w} = \boldsymbol{P}\boldsymbol{w}$. We obtain upper bounds on these three terms as follows.

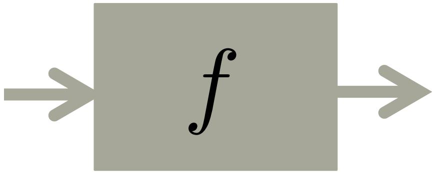
$$\hat{P}_{ii} < 1, \tag{21}$$

$$\left| \sum_{j:j \neq i} (w_i + w_j) \left(\hat{P}_{ji} - P_{ji} \right) \right| \lesssim \sqrt{\frac{n \log n}{\binom{n}{M} pL}} \sqrt{\frac{1}{M}}, \tag{22}$$

$$\sum_{j:j \neq i} |\hat{w}_j - w_j| \hat{P}_{ij} \lesssim \sqrt{\frac{n \log n}{\binom{n}{M} pL}} \sqrt{\frac{1}{M}}, \tag{23}$$

with high probability (see Lemmas 1, 2 and 3 in the supplementary for details). One can see that the inequalities (21)

et al., 2013) directly follows the ordering evaluated in each sample; if it is $1 \prec 2 \prec \cdots \prec M-1 \prec M$, it is broken into pairs of adjacent items: $1 \prec 2$ up to $M-1 \prec M$. Our method turns out to be consistent, i.e., $\frac{\Pr[y_{ij}=1]}{\Pr[y_{ji}=0]} = \frac{w_i}{w_j}$ (see (17)), whereas the adjacent breaking method is not (Azari Soufiani et al., 2013).



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**

MALE

→ *supervised learning*

– let the system **explore** and provide **feedback**

→ *reinforcement learning*

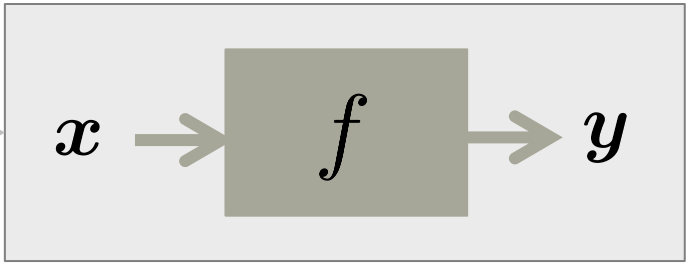
– **demonstrate** and let the system **imitate**

→ *imitation learning*

LEARNING FROM DATA

INTELLIGENT SYSTEMS

computer scientist



"Machine learning is the science of getting computers to act without being explicitly programmed."

Andrew Ng, 2013

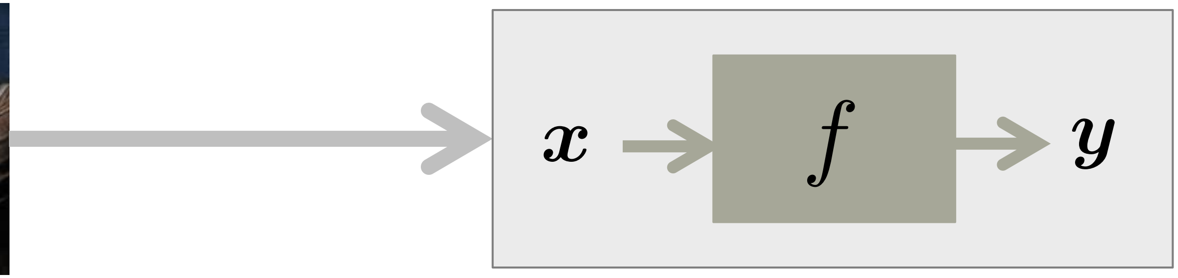
LEARNING FROM DATA

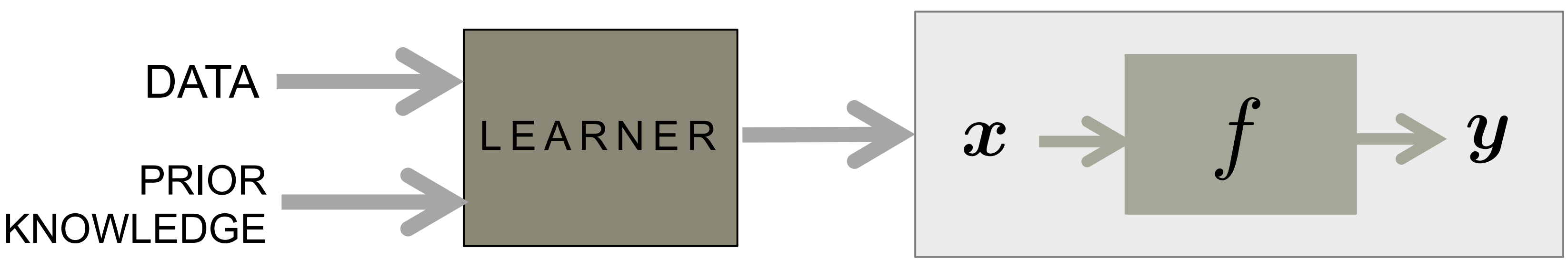
computer scientist $\rightarrow$ $x \rightarrow \boxed{f} \rightarrow y$

DATA $\rightarrow$ LEARNER $\rightarrow$ $x \rightarrow \boxed{f} \rightarrow y$

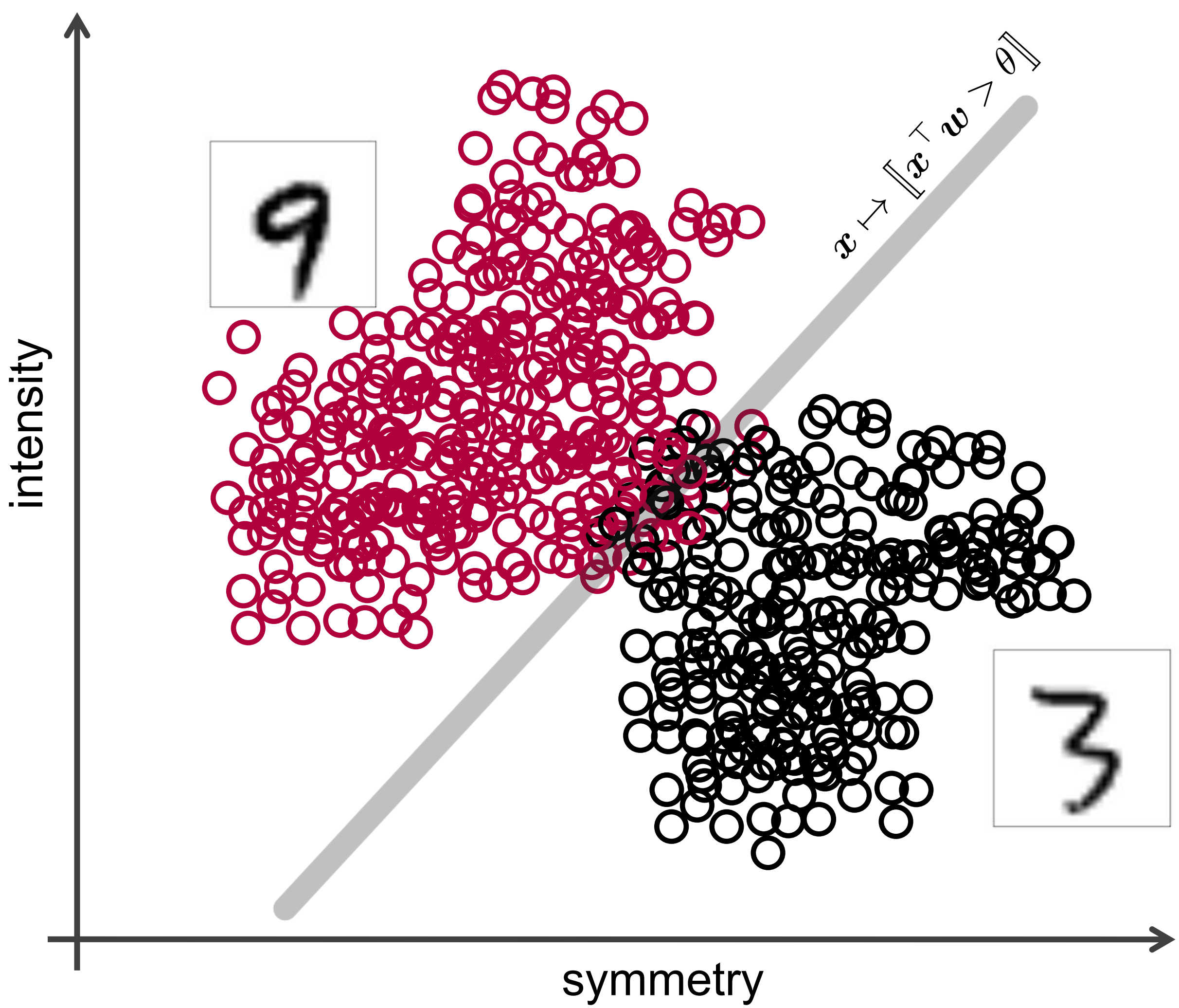
LEARNING FROM DATA

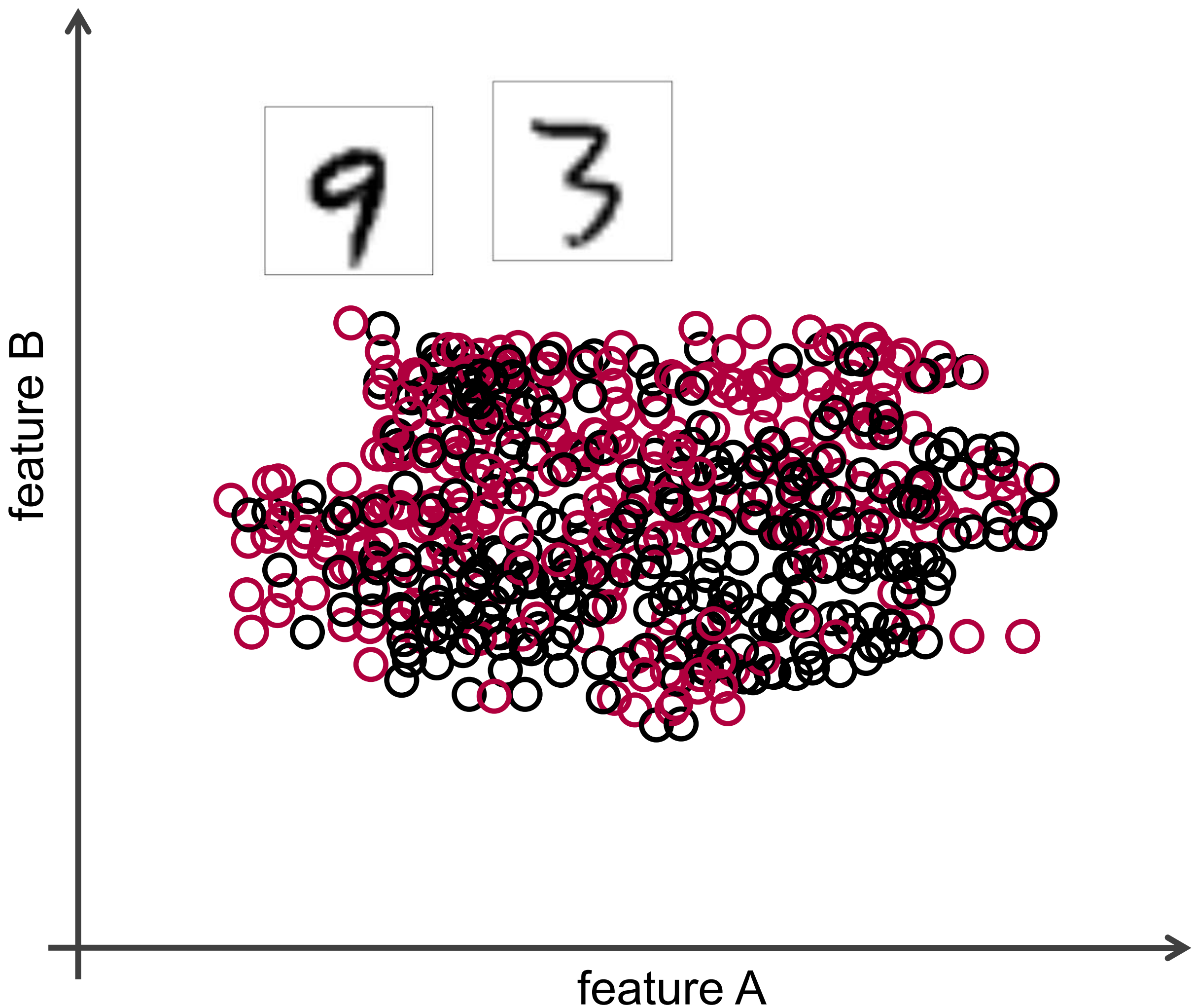
computer scientist



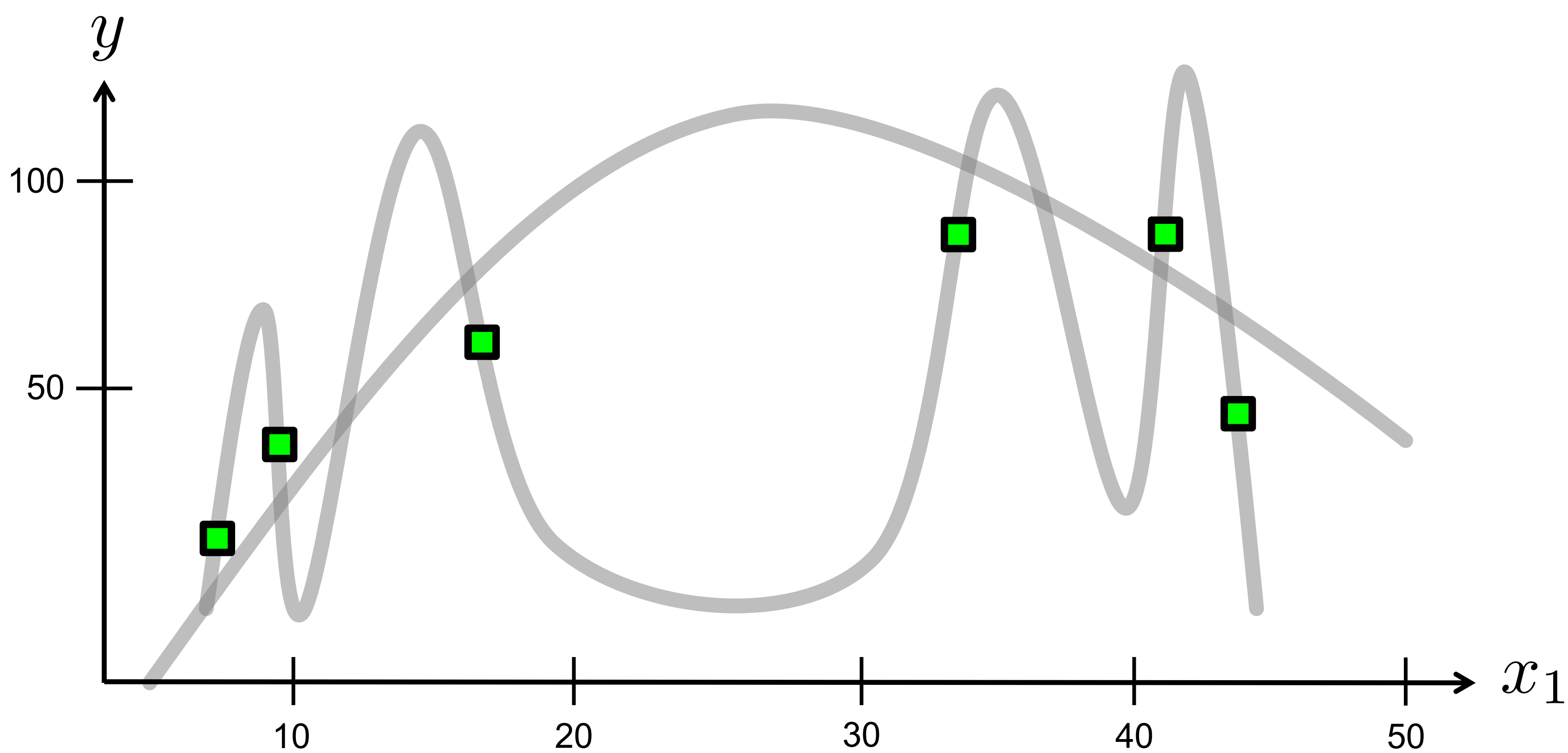


Learning does not mean turning data into knowledge, but revising prior knowledge in the light of observed data.


$\boldsymbol{x} \mapsto [\![\boldsymbol{x}^\top \boldsymbol{w} > \theta]\!]$
intensity
symmetry



9
3
feature B
feature A



y
100
50
10
20
30
40
50
x_1

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 t
        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', Logis
models.append(('LDA', Line
models.append(('KNN', KNei
models.append(('CART', Dec
models.append(('NB', Gauss
models.append(('SVM', SVC(
# evaluate each model in t
results = []
names = []
for name, model in models:
    kfold = model_selectio
    cv_results = model_sel
    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

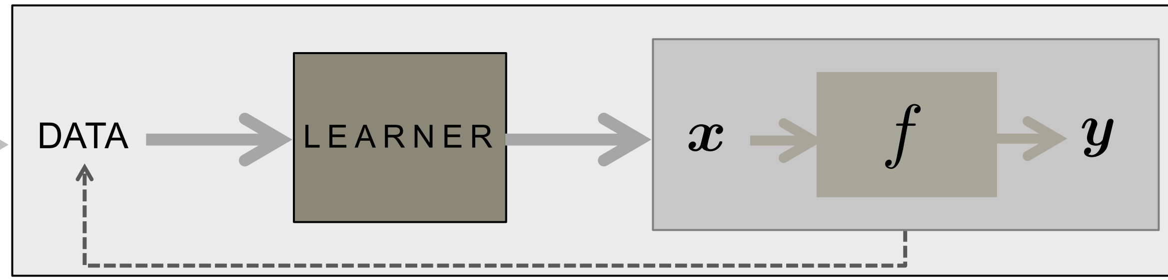
… suffers from knowledge acquisition bottleneck

… still requires a lot of ML expertise

AUTOMATED MACHINE LEARNING

INTELLIGENT SYSTEMS

Replacing the programmer by a learning algorithm …



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 …

INTELLIGENT SYSTEMS

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
- …

INTELLIGENT SYSTEMS

Objective of the learning problem

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

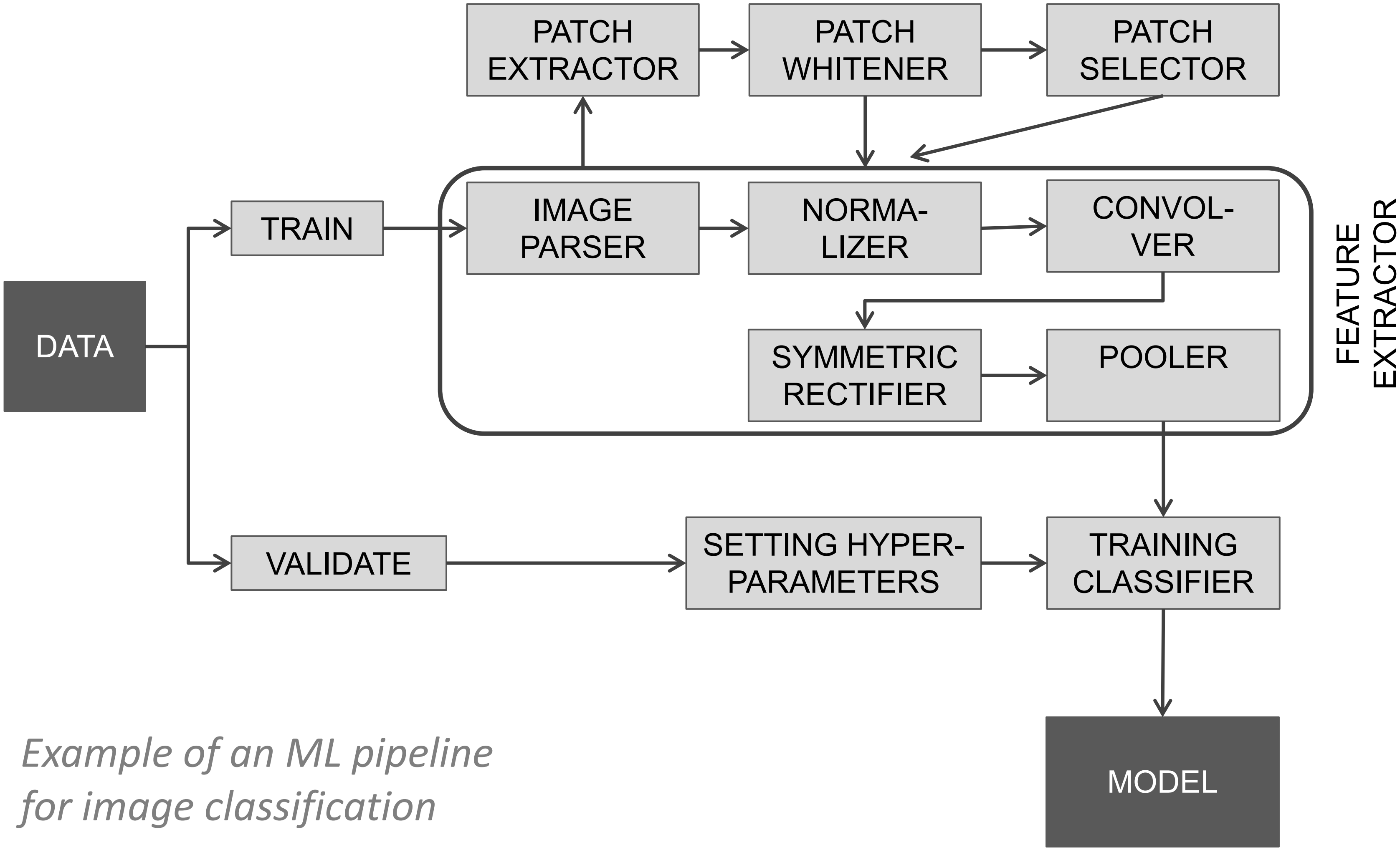
Specifying the model induction problem

- 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
- ...

ML PIPELINES



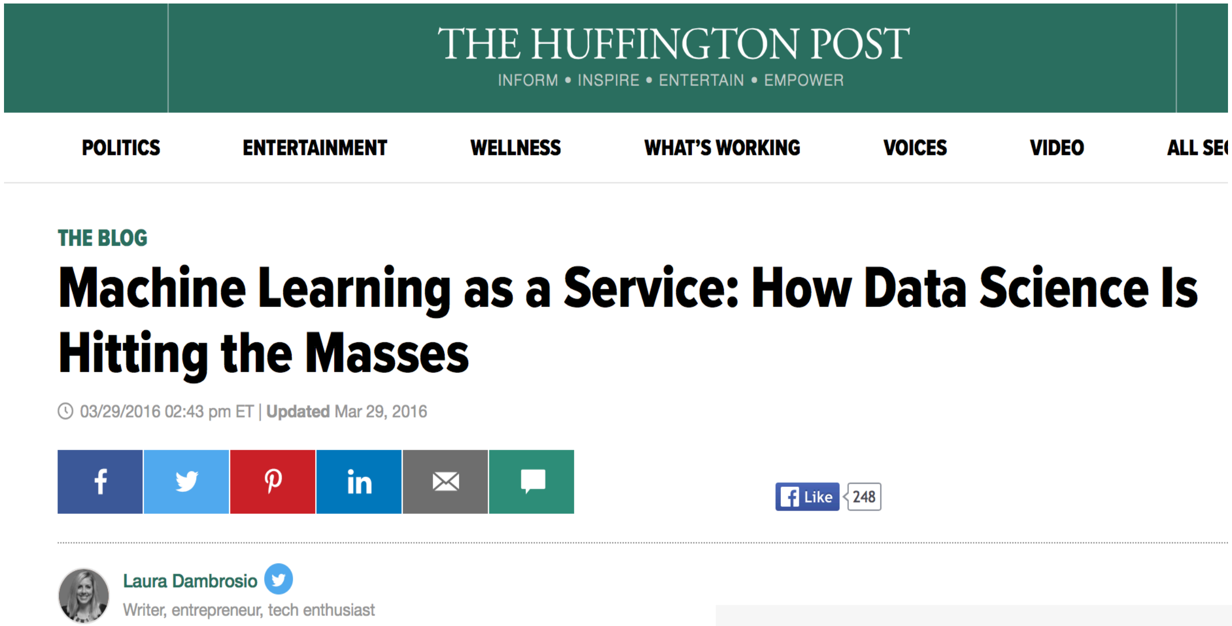
Example of an ML pipeline for image classification

INTELLIGENT SYSTEMS

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 …



THE HUFFINGTON POST

INFORM • INSPIRE • ENTERTAIN • EMPOWER

POLITICS ENTERTAINMENT WELLNESS WHAT'S WORKING VOICES VIDEO ALL SE

THE BLOG

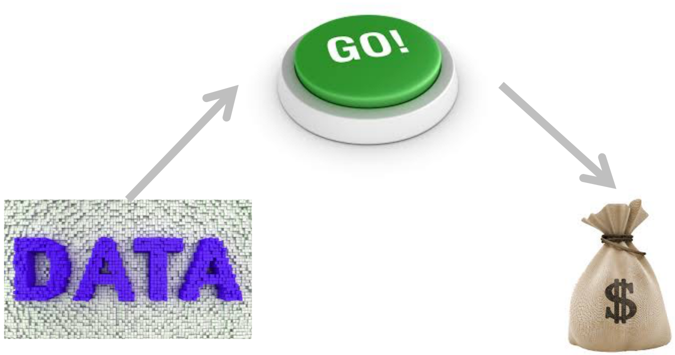
Machine Learning as a Service: How Data Science Is Hitting the Masses

03/29/2016 02:43 pm ET | Updated Mar 29, 2016

Like 248

Laura Dambrosio

Writer, entrepreneur, tech enthusiast



- 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.

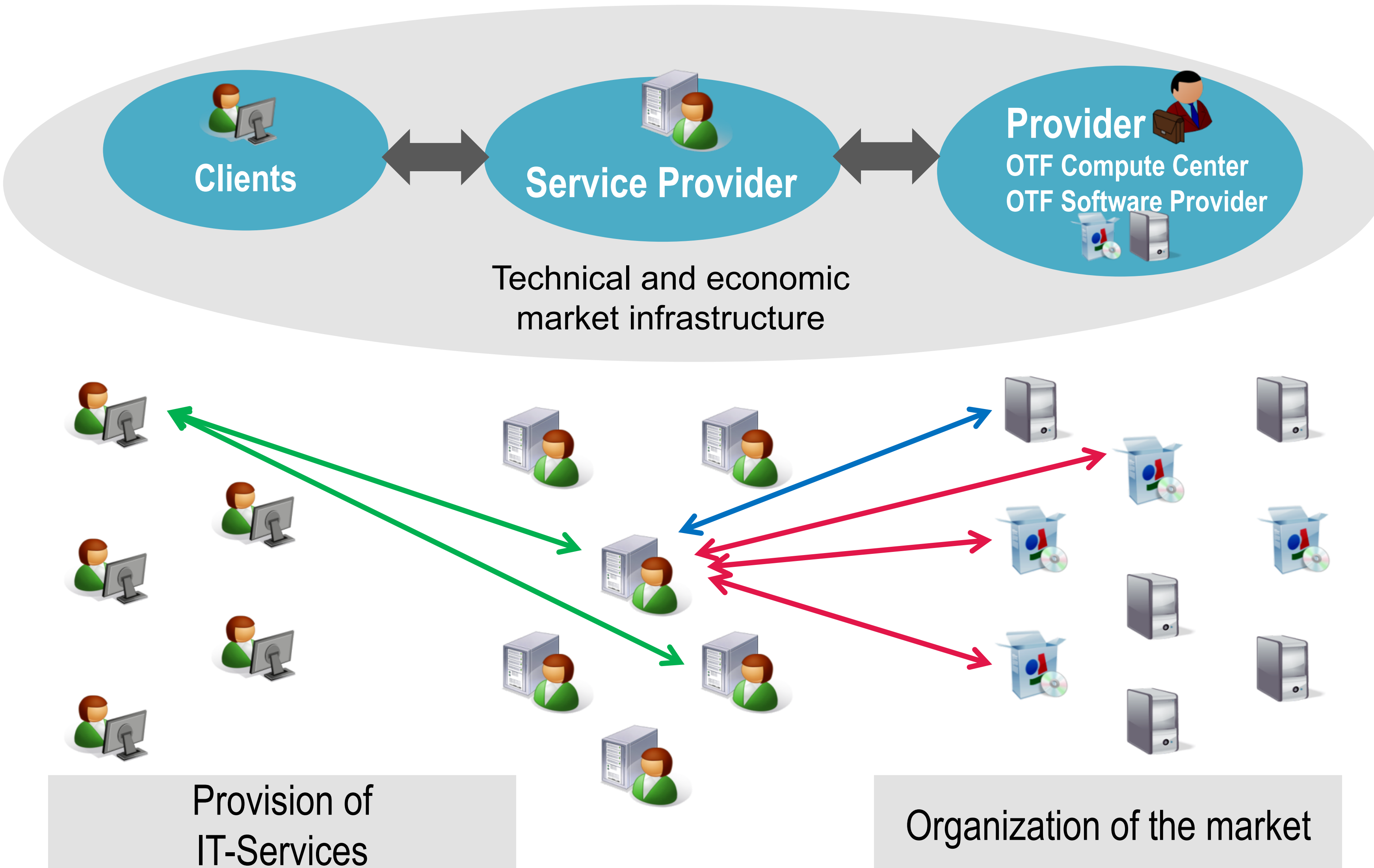
SFB 901

ON - THE - FLY COMPUTING

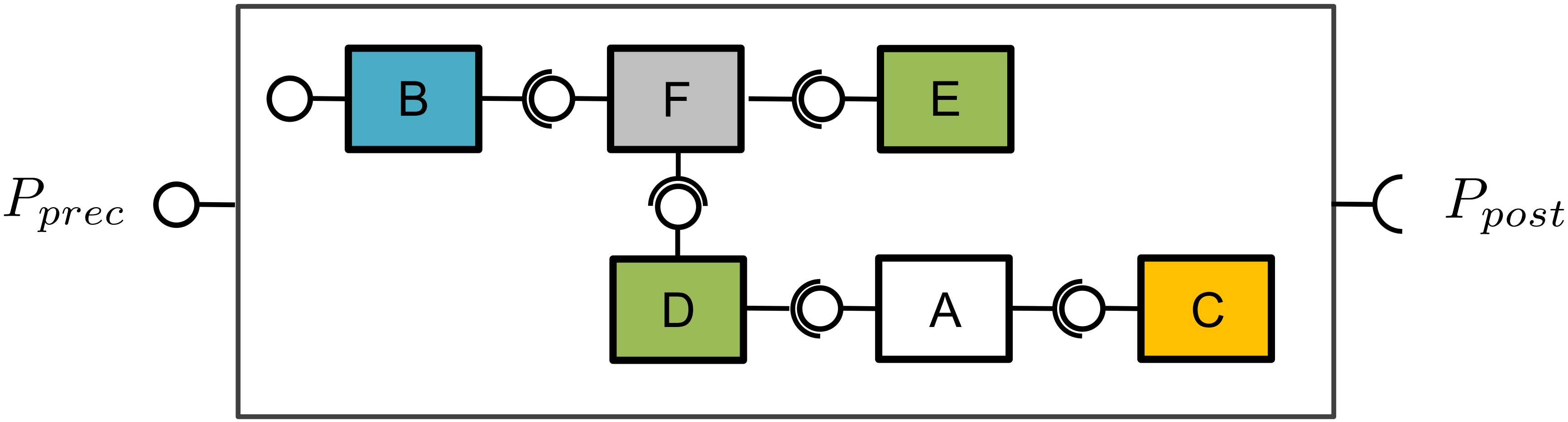
- On-the-Fly (OTF) Computing is a novel computing paradigm that aims at the provision of individually configured **software services in a market environment** that comprises so-called **OTF providers, service providers, and end-users** as main participants.

- 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**.


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) $\langle P_{pre}, P_{post} \rangle$

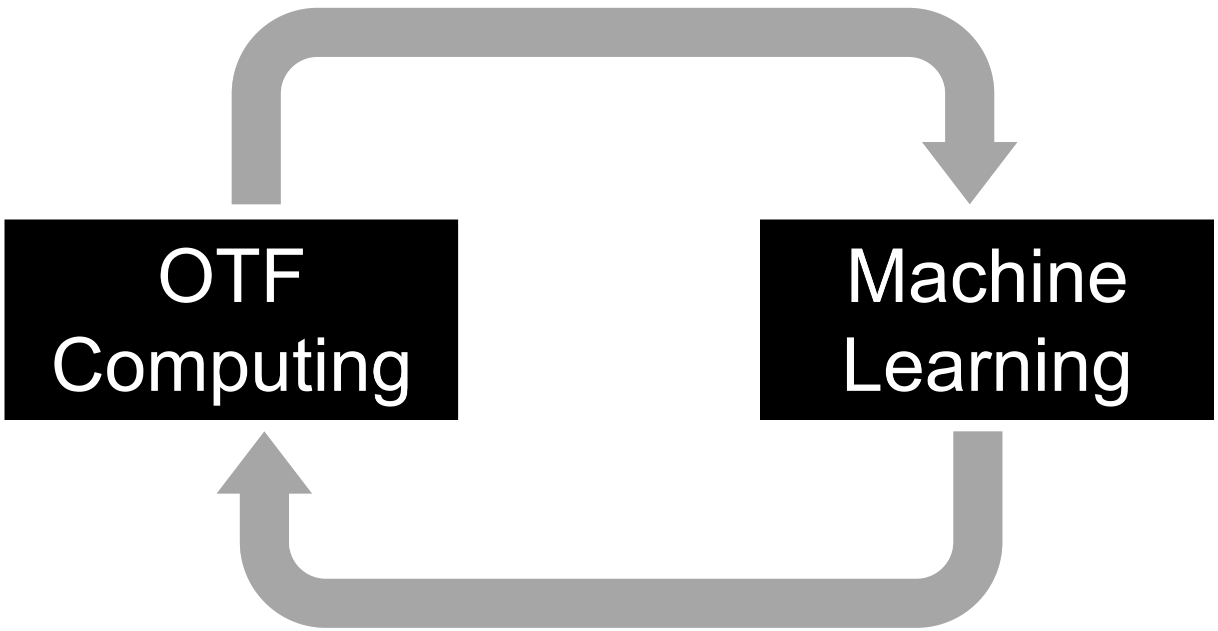


software service composition

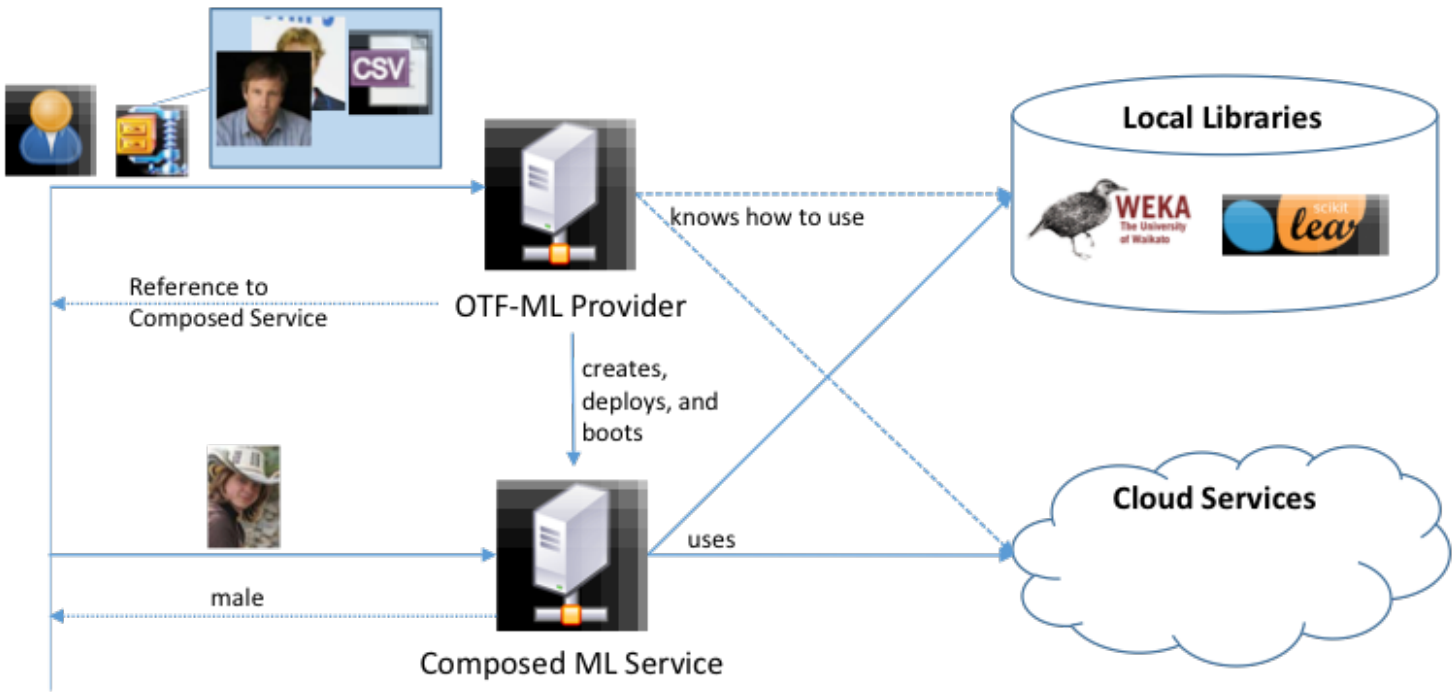
OTF COMPUTING AND ML

INTELLIGENT SYSTEMS

Automated composition of ML services

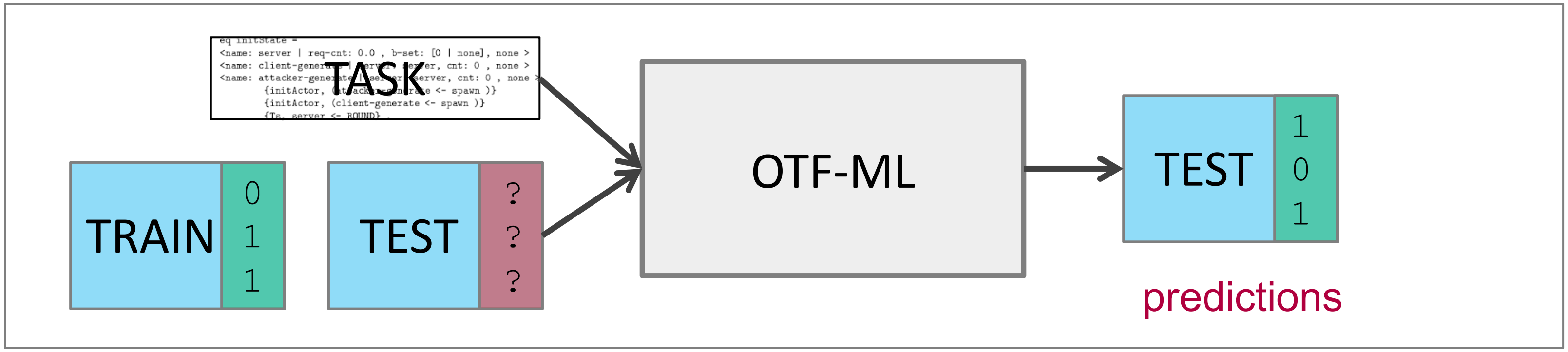


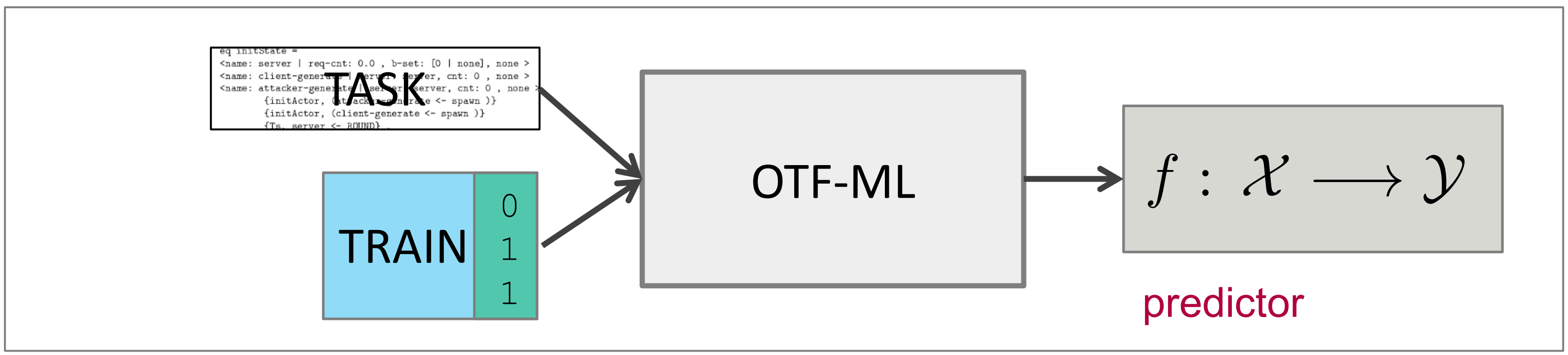
Improving efficiency and quality of service composition through ML

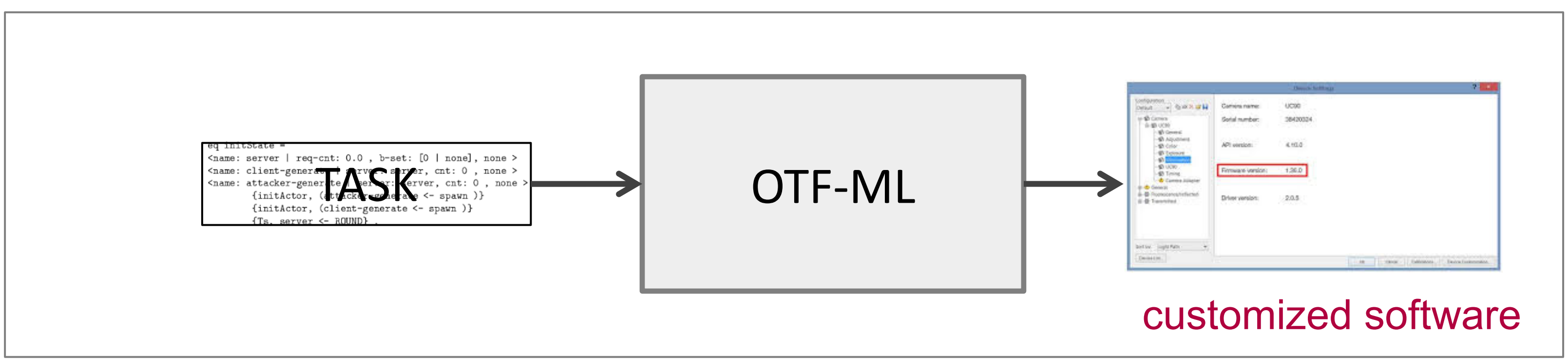


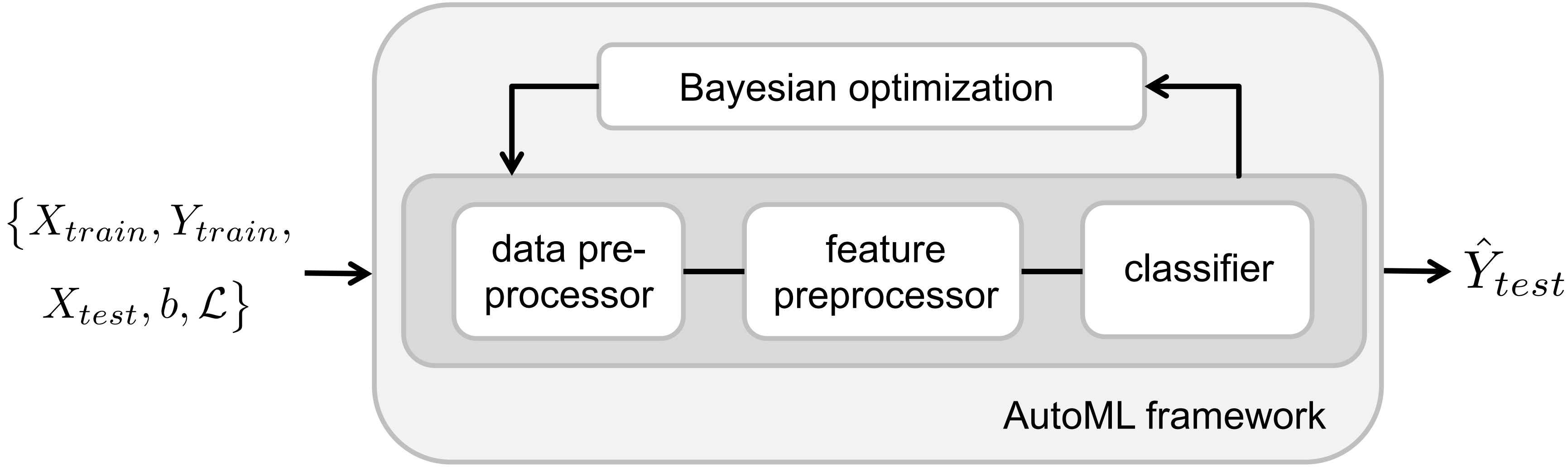
On-the-Fly Machine Learning *(OTF-ML) as an instantiation of OTF computing: On-the-fly selection, configuration, provision, and execution of machine learning and data analytics functionality.*

OTF-ML SCENARIOS



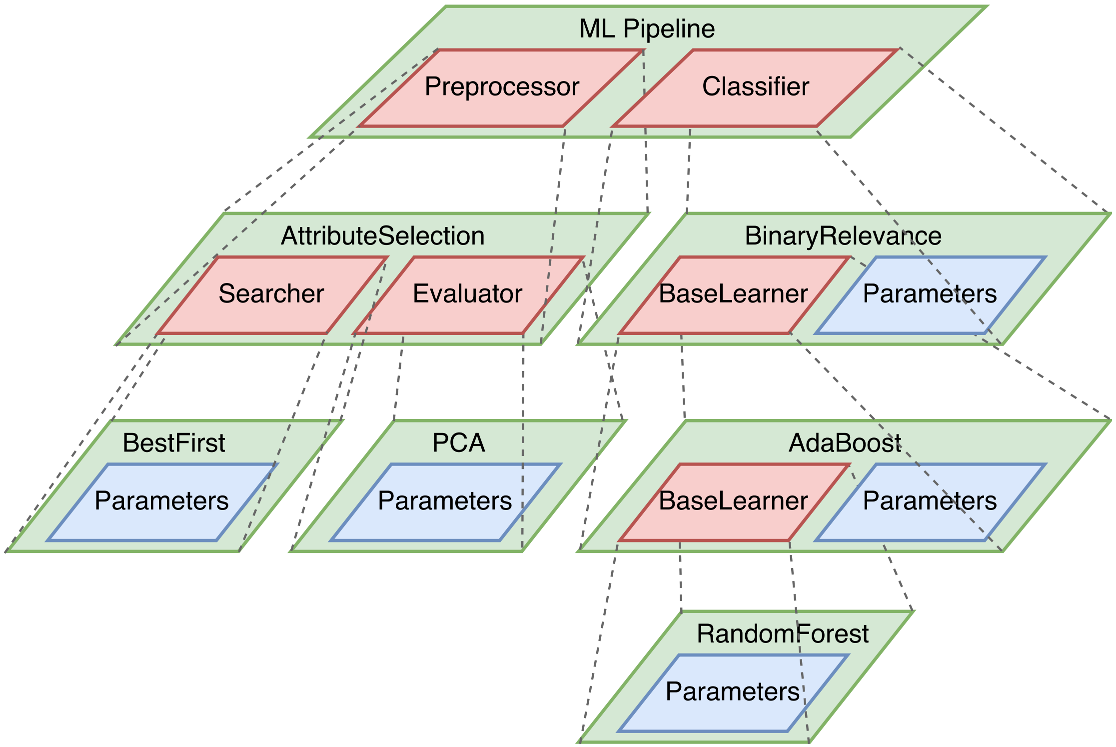




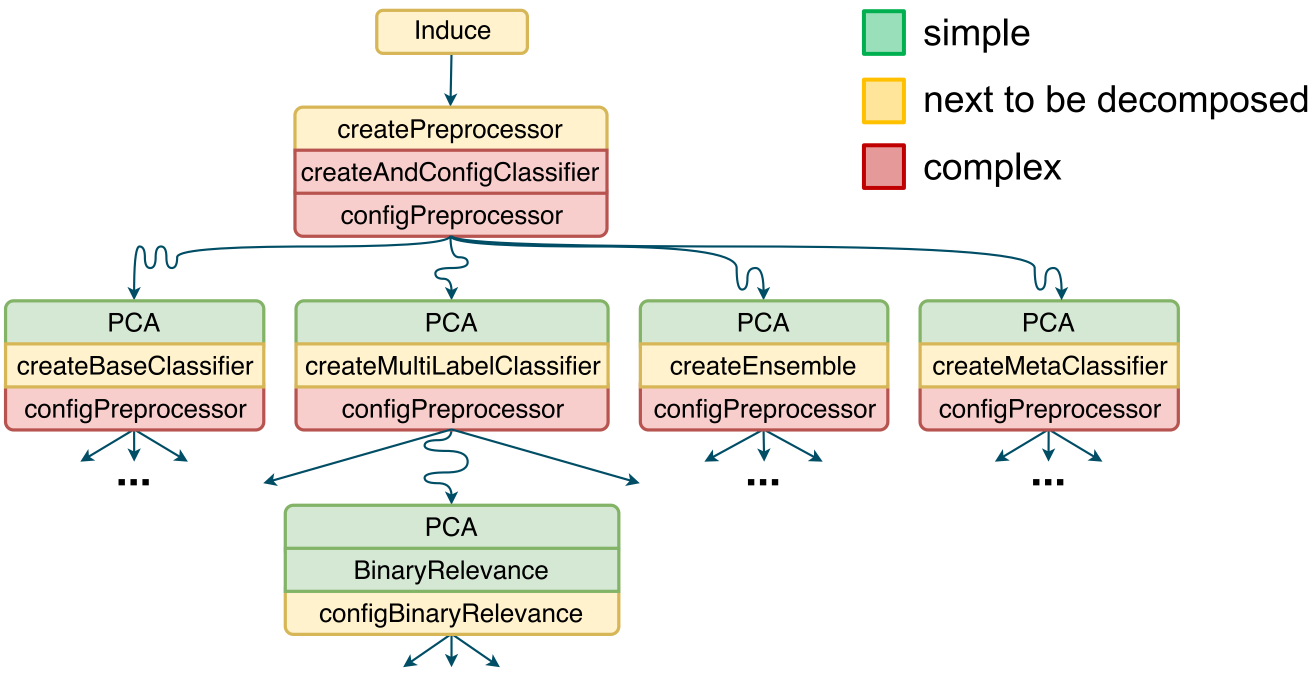


*Existing approaches optimize parameters of a **fixed ML pipeline.** The parameter space is structured, each "point" defines algorithm selection (model) and configuration (hyper-parameters). Essentially restricted to (binary) classification. No backtracking (e.g., due to overall insufficient quality) and no user interaction.*

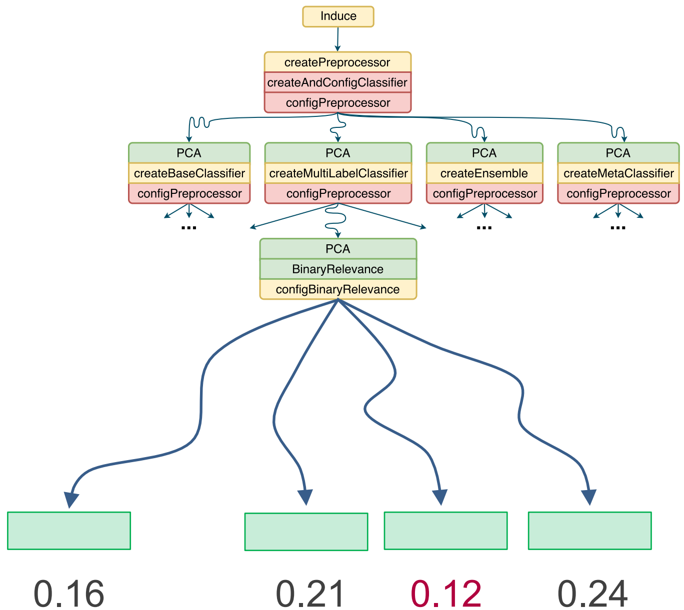
- **Hierarchical planning** (Hierarchical Task Networks, HTN) as a more flexible and expressive formalism to create ML pipelines.
- **Recursive reduction** of complex tasks to (complex or simple) subtasks.

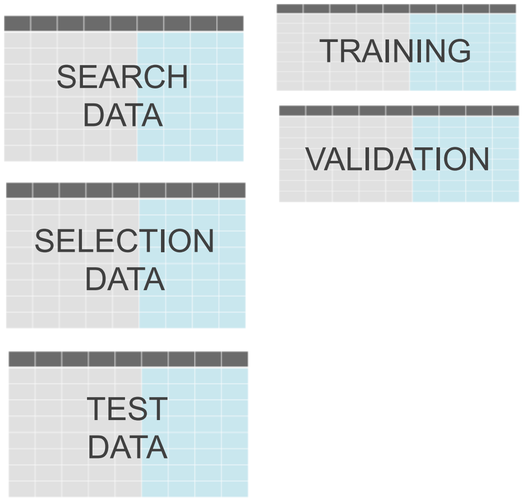


- Algorithmically solved using **graph search** algorithms.
- A node is a **goal node** if all remaining tasks are simple.
- HTN via **forward-decomposition**: one successor is created for each possible decomposition of the first unsolved task in the list of remaining tasks.



- 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**.





ML-PLAN


Visualizer for jaicore.search.structure.core.GraphEventBus@4082ba93
TechSmith
ERSTELLT MIT CAMTASIA TESTVERSION

ML PIPELINES


```
PolynomialFeatures

{degree: 2,
Include_bias: false,
Interaction_only:
false}
```

$$\begin{pmatrix} .21, .3, \dots \\ .53, .4, .. \\ \dots \\ .02, .21 \dots \end{pmatrix}$$

```
StackingEstimator
{estimator:
    BernoulliNB
    {alpha=0.1,
    fit_prior=false}
}
```

$$\begin{pmatrix} 1 \\ 3 \\ \dots \\ 2 \end{pmatrix}$$

```
RandomForestClassifier
{bootstrap: true,
criterion:: gini,
Max_features: 0.55,
Min_samples_leaf:3
Min_samples_split: 5,
N_estimators: 100}
```

Pipeline created for the page-blocks benchmark data

SUMMARY & CONCLUSION

```
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 t
        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', Logis
models.append(('LDA', Line
models.append(('KNN', KNei
models.append(('CART', Dec
models.append(('NB', Gauss
models.append(('SVM', SVC(
# evaluate each model in t
results = []
names = []
for name, model in models:
    kfold = model_selectio
    cv_results = model_sel
    results.append(cv_resu
    names.append(name)
    msg = "%s: %f (%f)" %
    print(msg)
```

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

{MALE, FEMALE}

data

ML ALGORITHM

algorithm (predictor)

ML problem (data)

AutoML ALGORITHM

ML algorithm (learner)

ML problem (data)

MetaML ALGORITHM

AutoML algorithm

SUMMARY & CONCLUSION

INTELLIGENT SYSTEMS

 → *Instruct the computer how to solve the problem* → {MALE, FEMALE}

data → *Instruct the computer how to learn how to solve the problem* → algorithm (predictor)

ML problem (data) → *Instruct the computer how to find a good way to learn how to solve the problem* → ML algorithm (learner)

ML problem (data) → *Instruct the computer how to learn how to learn how to solve the problem* → AutoML algorithm

SUMMARY & CONCLUSION

Instruct the computer how to solve the problem

{MALE, FEMALE}

data

Instruct the computer how to learn how to solve the problem

algorithm (predictor)

ML problem (data)

Instruct the computer how to find a good way to learn

ML algorithm (learner)

ML problem (data)

Instruct the computer how to learn how to learn how to solve the problem

AutoML algorithm

The "ML as a service" idea comes with a number of interesting new challenges, both scientifically and from an application point of view.