Deep Machines that know when they do not know

Consider e.g. unsupervised scene understanding using a generative model

[Attend-Infer-Repeat (AIR) model, Hinton et al. NIPS 2016]

Sum-Product Probabilistic Programming:
Making machine learning and data science easier [Stelzner, Molina, Peharz, Vergari, Trapp, Valera, Ghahramani, Kersting ProgProb 2018]

Probabilistic Programming:
Easier modelling by programming generative models in a high-level, prob. language

Use deep probabilistic models that feature tractable, deterministic inference

Deep Probabilistic Prog.:
Modelling and inference might be hard, so use a deep neural network for it

Kristian Kersting
AI and ML have a strong impact

Data are now ubiquitous; there is great value from understanding this data, building models and making predictions.

However, there are not enough data scientists, statisticians, machine learning and AI experts.

Provide the foundations, algorithms, and tools to develop systems that ease or even automate AI model discovery from data as much as possible.
Deep Neural Networks

Potentially much more powerful than shallow architectures, represent computations

Deep Neural Networks

Potentially much more powerful than shallow architectures, represent computations

They “develop intuition” about complicated biological processes and generate scientific data
[Schramowski, Brugger, Mahlein, Kersting 2019]
Deep Neural Networks

Potentially much more powerful than shallow architectures, represent computations


They “invent” constrained optimizers

[Schramowski, Bauckhage, Kersting arXiv:1803.04300, 2018]
Deep Neural Networks

Potentially much more powerful than shallow architectures, represent computations

They “capture” stereotypes from human language
Deep Neural Networks

Potentially much more powerful than shallow architectures, represent computations

The Moral Choice Machine

<table>
<thead>
<tr>
<th>Dos</th>
<th>WEAT</th>
<th>Bias</th>
</tr>
</thead>
<tbody>
<tr>
<td>smile</td>
<td>0.116</td>
<td>0.348</td>
</tr>
<tr>
<td>sightsee</td>
<td>0.090</td>
<td>0.281</td>
</tr>
<tr>
<td>cheer</td>
<td>0.094</td>
<td>0.277</td>
</tr>
<tr>
<td>celebrate</td>
<td>0.114</td>
<td>0.264</td>
</tr>
<tr>
<td>picnic</td>
<td>0.093</td>
<td>0.260</td>
</tr>
<tr>
<td>snuggle</td>
<td>0.108</td>
<td>0.238</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Don’ts</th>
<th>WEAT</th>
<th>Bias</th>
</tr>
</thead>
<tbody>
<tr>
<td>rot</td>
<td>-0.099</td>
<td>-1.118</td>
</tr>
<tr>
<td>negative</td>
<td>-0.101</td>
<td>-0.763</td>
</tr>
<tr>
<td>harm</td>
<td>-0.110</td>
<td>-0.730</td>
</tr>
<tr>
<td>damage</td>
<td>-0.105</td>
<td>-0.664</td>
</tr>
<tr>
<td>slander</td>
<td>-0.108</td>
<td>-0.600</td>
</tr>
<tr>
<td>slur</td>
<td>-0.109</td>
<td>-0.569</td>
</tr>
</tbody>
</table>

But lucky they also “capture” our moral choices
[Jentzsch, Schramowski, Rothkopf, Kersting AIES 2019]
The Moral Choice Machine

<table>
<thead>
<tr>
<th>Dos</th>
<th>WEAT</th>
<th>Bias</th>
</tr>
</thead>
<tbody>
<tr>
<td>smile</td>
<td>0.116</td>
<td>0.348</td>
</tr>
<tr>
<td>sightsee</td>
<td>0.090</td>
<td>0.281</td>
</tr>
<tr>
<td>cheer</td>
<td>0.094</td>
<td>0.277</td>
</tr>
<tr>
<td>celebrate</td>
<td>0.114</td>
<td>0.264</td>
</tr>
<tr>
<td>picnic</td>
<td>0.093</td>
<td>0.260</td>
</tr>
<tr>
<td>snuggle</td>
<td>0.108</td>
<td>0.238</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Don’ts</th>
<th>WEAT</th>
<th>Bias</th>
</tr>
</thead>
<tbody>
<tr>
<td>rot</td>
<td>-0.099</td>
<td>-1.118</td>
</tr>
<tr>
<td>negative</td>
<td>-0.101</td>
<td>-0.763</td>
</tr>
<tr>
<td>harm</td>
<td>-0.110</td>
<td>-0.730</td>
</tr>
<tr>
<td>damage</td>
<td>-0.105</td>
<td>-0.664</td>
</tr>
<tr>
<td>slander</td>
<td>-0.108</td>
<td>-0.600</td>
</tr>
<tr>
<td>slur</td>
<td>-0.109</td>
<td>-0.569</td>
</tr>
</tbody>
</table>

But lucky they also “capture” our moral choices

[Jentzsch, Schramowski, Rothkopf, Kersting AIES 2019]
Deep neural networks do not quantify their uncertainty.
They are not calibrated probabilistic models.

Train & Evaluate

Transfer Testing

[MNIST]

[SVHN]

[SEMEION]

Input log "likelihood" (sum over outputs)


[Peharz, Vergari, Molina, Stelzner, Trapp, Kersting, Ghahramani UDL@UAI 2018]
Getting deep systems that know when they don’t know.
Can we borrow ideas from deep learning for probabilistic graphical models?

Judea Pearl, UCLA
Turing Award 2012
This results in Sum-Product Networks, a deep probabilistic learning framework. Computational graph (kind of TensorFlow graphs) that encodes how to compute probabilities.

Inference is linear in size of network.
This results in Sum-Product Networks, a deep probabilistic learning framework.

\[\rightarrow \text{convex sum}\]
\[\times \text{product}\]
\[\wedge \text{distribution}\]

Computational graph (kind of TensorFlow graphs) that encodes how to compute probabilities.

Inference is linear in size of network.
This results in Sum-Product Networks, a deep probabilistic learning framework.

\[ \text{convex sum} \quad + \]
\[ \text{product} \quad \times \]
\[ \text{distribution} \quad \land \]

**Completeness**
sum children: same scope

**Decomposability**
product children: non-overlapping scope

**Computational graph** (kind of TensorFlow graphs) that encodes how to compute probabilities

Inference is linear in size of network
And there is a way to select models

Testing independence of random variables using e.g. (nonparametric) tests

Examples

Random Variables

Conditioning, e.g., via clustering

keep growing alternatingly * and + layers
SPFlow: An Easy and Extensible Library for Sum-Product Networks

Domain Specific Language, Inference, EM, and Model Selection as well as Compilation of SPNs into TF and PyTorch and also into flat, library-free code even suitable for running on devices: C/C++, GPU, FPGA

SPFlow, an open-source Python library providing a simple interface to inference, learning and manipulation routines for deep and tractable probabilistic models called Sum-Product Networks (SPNs). The library allows one to quickly create SPNs both from data and through a domain specific language (DSL). It efficiently implements several probabilistic inference routines like computing marginals, conditional probabilities and (approximate) most probable explanations (MPEs) along with

https://github.com/SPFlow/SPFlow
Random sum-product networks

[Peharz, Vergari, Molina, Stelzner, Trapp, Kersting, Ghahramani UDL@UAI 2018]

<table>
<thead>
<tr>
<th>Data Set</th>
<th>RAT-SFN</th>
<th>MLP</th>
<th>vMLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>98.19</td>
<td>98.32</td>
<td>98.09</td>
</tr>
<tr>
<td></td>
<td>(8.5M)</td>
<td>(2.64M)</td>
<td>(5.28M)</td>
</tr>
<tr>
<td>F-MNIST</td>
<td>89.52</td>
<td>90.81</td>
<td>89.81</td>
</tr>
<tr>
<td></td>
<td>(0.65M)</td>
<td>(9.28M)</td>
<td>(1.07M)</td>
</tr>
<tr>
<td>20-NG</td>
<td>47.8</td>
<td>49.05</td>
<td>48.81</td>
</tr>
<tr>
<td></td>
<td>(0.37M)</td>
<td>(0.16M)</td>
<td>(0.16M)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>0.0852</td>
</tr>
<tr>
<td></td>
<td>(17M)</td>
</tr>
<tr>
<td>F-MNIST</td>
<td>0.3525</td>
</tr>
<tr>
<td></td>
<td>(0.65M)</td>
</tr>
<tr>
<td>20-NG</td>
<td>1.6954</td>
</tr>
<tr>
<td></td>
<td>(1.63M)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>4042</td>
</tr>
<tr>
<td></td>
<td>(17M)</td>
</tr>
<tr>
<td>F-MNIST</td>
<td>3069</td>
</tr>
<tr>
<td></td>
<td>(0.65M)</td>
</tr>
<tr>
<td>20-NG</td>
<td>1680</td>
</tr>
<tr>
<td></td>
<td>(1.63M)</td>
</tr>
</tbody>
</table>

outliers
prototypes
outliers
prototypes
How do we do data science offshore?

[Sommer, Oppermann, Molina, Binnig, Kersting, Koch ICDD 2018]
There are generic protocols to validate computations on authenticated data without knowledge of the secret key.

#### DNA MSPN ####
- Gates: 298208
- Yao Bytes: 9542656
- Depth: 615

#### DNA PSPN ####
- Gates: 228272
- Yao Bytes: 7304704
- Depth: 589

#### NIPS MSPN ####
- Gates: 1001477
- Yao Bytes: 32047264
- Depth: 970
Learning the Structure of Autoregressive Deep Models such as PixelCNNs [van den Oord et al. NIPS 2016]

Learn Conditional SPN by testing conditional independence and using conditional clustering, using e.g. [Zhang et al. UAI 2011; Lee, Honovar UAI 2017; He et al. ICDM 2017; Zhang et al. AAAI 2018; Runge AISTATS 2018]

Conditional SPNs
[Shao, Molina, Vergari, Peharz, Kersting 2019]
Learn Conditional SPN by testing conditional independence and using conditional clustering, using e.g. [Zhang et al. UAI 2011; Lee, Honovar UAI 2017; He et al. ICDM 2017; Zhang et al. AAAI 2018; Runge AISTATS 2018]

**Conditional SPNs**
[Shao, Molina, Vergari, Peharz, Kersting 2019]
Learn Conditional SPN by testing conditional independence and using conditional clustering, using e.g. [Zhang et al. UAI 2011; Lee, Honovar UAI 2017; He et al. ICDM 2017; Zhang et al. AAAI 2018; Runge AISTATS 2018]

**Conditional SPNs**
[Shao, Molina, Vergari, Peharz, Kersting 2019]
Learn Conditional SPN by testing conditional independence and using conditional clustering, using e.g. [Zhang et al. UAI 2011; Lee, Honovar UAI 2017; He et al. ICDM 2017; Zhang et al. AAAI 2018; Runge AISTATS 2018]

Conditional SPNs
[Shao, Molina, Vergari, Peharz, Kersting 2019]
Generally, we can explore more of deep learning for probability distributions: **Residual SPN**

Short cuts among (sub)mixtures)

[Ventola, Molina, Stelzner, Kersting 2019]

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Best weak SPN+opt</th>
<th>Simple Mixture+opt</th>
<th>LearnSPN+opt</th>
<th>ResSPN+opt</th>
<th>BigMix ResSPN+opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio5</td>
<td>-42.639</td>
<td>-42.069</td>
<td>-44.363</td>
<td>-41.526</td>
<td>-40.848</td>
</tr>
<tr>
<td>Audio10</td>
<td>-42.639</td>
<td>-41.919</td>
<td>-44.363</td>
<td>-40.345</td>
<td>-40.259</td>
</tr>
<tr>
<td>Jester</td>
<td>55.335</td>
<td>-53.393</td>
<td>-54.934</td>
<td>-54.455</td>
<td>-53.214</td>
</tr>
</tbody>
</table>

---

**Log Likelihood**

---

**Running Times**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Best weak SPN+opt</th>
<th>Simple Mixture+opt</th>
<th>LearnSPN+opt</th>
<th>ResSPN+opt</th>
<th>BigMix ResSPN+opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLTCS</td>
<td>31</td>
<td>117</td>
<td>20</td>
<td>2606</td>
<td>2909</td>
</tr>
<tr>
<td>KDDCup2k</td>
<td>264</td>
<td>3686</td>
<td>2788</td>
<td>2461</td>
<td>7236</td>
</tr>
<tr>
<td>Audio5</td>
<td>56</td>
<td>165</td>
<td>255</td>
<td>172</td>
<td>340</td>
</tr>
<tr>
<td>Audio10</td>
<td>31</td>
<td>187</td>
<td>255</td>
<td>10841</td>
<td>10641</td>
</tr>
<tr>
<td>Jester</td>
<td>74</td>
<td>128</td>
<td>183</td>
<td>92</td>
<td>186</td>
</tr>
<tr>
<td>Netflix</td>
<td>25</td>
<td>86</td>
<td>166</td>
<td>105</td>
<td>196</td>
</tr>
</tbody>
</table>
Distribution-agnostic Deep Probabilistic Learning

Use nonparametric independency tests and piece-wise linear approximations
However, we have to provide the statistical types and do not gain insights into the parametric forms of the variables. Are they Gaussians? Gammas? …

Use nonparametric independence tests and piece-wise linear approximations.
We can even automatically discovers the statistical types and parametric forms of the variables.
That is, the machine understands the data with few expert input …

Exploring the Titanic dataset

This report describes the dataset Titanic and contains general statistical information and an analysis on the influence different features and subgroups of the data have on each other. The first part of the report contains general statistical information about the dataset and an analysis of the variables and probability distributions. The second part focuses on a subgroup analysis of the data. Different clusters identified by the network are analyzed and compared to give an insight into the structure of the data. Finally, the influence different variables have on the predictive capabilities of the model are analyzed. The whole report is generated by fitting a sum product network to the data and extracting all information from this model.


…and can compile data reports automatically
The machine understands the data with no expert input ...

...and can compile data reports automatically

*[Baehrens, Schroeter, Harmeling, Kawanabe, Hansen, Müller JMLR 11:1803-1831, 2010]

Explanation vector* (computable in linear time in the size of the SPN) showing the impact of "gender" on the chances of survival for the Titanic dataset
P(heart attack | )?
P(heart attack | )?

A.I. Is Harder Than You Think

By Gary Marcus and Ernest Davis

Mr. Marcus is a professor of psychology and neural science. Mr. Davis is a professor of computer science.

May 18, 2018
A.I. Is Harder Than You Think

By Gary Marcus and Ernest Davis

Mr. Marcus is a professor of psychology and neural science. Mr. Davis is a professor of computer science.

May 18, 2018
Crossover of ML and DS with data & programming abstractions


- building general-purpose data science and ML machines
- make the ML/DS expert more effective
- increases the number of people who can successfully build ML/DS applications

\[ P(\text{heart attack} | \text{?}) \]
Atherosclerosis is the cause of the majority of Acute Myocardial Infarctions (heart attacks)

Table:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Likelihood</th>
<th>AUC-ROC</th>
<th>AUC-PR</th>
<th>Time</th>
<th>State-of-the-art</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boosting</td>
<td>0.81</td>
<td>0.96</td>
<td>0.93</td>
<td>9s</td>
<td></td>
</tr>
<tr>
<td>LSM</td>
<td>0.73</td>
<td>0.54</td>
<td>0.62</td>
<td>93 hrs</td>
<td>37200x faster</td>
</tr>
</tbody>
</table>

Probabilistic Programming: The Democratization of Machine Learning and Data Science

Plaque in the left coronary artery

[Boosted Statistical Relational Learning, Foundations & Trends in Data-Driven Medicine]

Understanding Electronic Health Records

[ICML’08; Karwath, Kersting, Landwehr ICDM’08; Natarajan, Joshi, Tadepelli, Kersting, Shavlik. IJCAI’11; Natarajan, Kersting, Ip, Jacobs, Carr IAAI ’13; Yang, Kersting, Terry, Carr, Natarajan AIME ’15; Khot, Natarajan, Kersting, Shavlik ICDM’13, MLJ’12, MLJ’15, Yang, Kersting, Natarajan BIBM’17]
Human-in-the-loop learning
New field: Probabilistic Programming

Appplies to QPs but here illustrated on MDPs for a factory agent which must paint two objects and connect them. The objects must be smoothed, shaped and polished and possibly drilled before painting, each of which actions require a number of tools which are possibly available. Various painting and connection methods are represented, each having an effect on the quality of the job, and each requiring tools. Rewards (required quality) range from 0 to 10 and a discounting factor of 0.9 was used.
In general, computing the exact posterior is intractable, i.e., inverting the generative process to determine the state of latent variables corresponding to an input is time-consuming and error-prone.

### Deep Probabilistic Programming

```python
import pyro.distributions as dist
def model(data):
    # define the hyperparameters that control the beta prior
    alpha_0 = torch.tensor(10.0)
    beta_0 = torch.tensor(10.0)
    # sample f from the beta prior
    f = pyro.sample("latent_fairness", dist.Beta(alpha_0, beta_0))
    # loop over the observed data
    for i in range(len(data)):
        # observe datapoint i using the bernoulli
        pyro.sample("obs_({})".format(i), dist.Bernoulli(f), obs=data[i])

def guide(data):
    # register the two variational parameters with Pyro.
    alpha_q = pyro.param("alpha_q", torch.tensor(1.0),
                         constraint=constraints.positive)
    beta_q = pyro.param("beta_q", torch.tensor(1.0),
                         constraint=constraints.positive)
    # sample latent_fairness from the distribution Beta(alpha_q, beta_q)
    pyro.sample("latent_fairness", dist.Beta(alpha_q, beta_q))
```

(2) Ease the implementation by some high-level, probabilistic programming language

(1) Instead of optimizing variational parameters for every new data point, use a deep network to predict the posterior given $X$  
[Kingma, Welling 2013, Rezende et al. 2014]
(1) Instead of optimizing variational parameters for every new data point, use a deep network to predict the posterior given $X$  [Kingma, Welling 2013, Rezende et al. 2014]

(2) Ease the implementation by some high-level, probabilistic programming language

```python
import pyro.distributions as dist

def model(data):
    # define the hyperparameters that control the beta prior
    alpha0 = torch.tensor(10.0)
    beta0 = torch.tensor(10.0)
    # sample f from the beta prior
    f = pyro.sample("latent_fairness", dist.Beta(alpha0, beta0))
    # loop over the observed data
    for i in range(len(data)):
        # observe datapoint i using the Bernoulli
        pyro.sample("obs_{i}".format(i), dist.Bernoulli(f), obs=data[i])

def guide(data):
    # register the two variational parameters with Pyro.
    alpha_q = pyro.param("alpha_q", torch.tensor(15.0),
                        constraint=constraints.positive)
    beta_q = pyro.param("beta_q", torch.tensor(15.0),
                        constraint=constraints.positive)
    # sample latent_fairness from the distribution Beta(alpha_q, beta_q)
    pyro.sample("latent_fairness", dist.Beta(alpha_q, beta_q))
```

Sum-Product Network

Deep Neural Network
Consider e.g. unsupervised scene understanding using a generative model

[Attend-Infer-Repeat (AIR) model, Hinton et al. NIPS 2016]

Sum-Product Probabilistic Programming: Making machine learning and data science easier [Stelzner, Molina, Peharz, Vergari, Trapp, Valera, Ghahramani, Kersting ProgProb 2018]

Probabilistic Programming:
Easier modelling by programming generative models in a high-level, prob. language

Deep Probabilistic Prog.:
Modelling and inference might be hard, so use a deep neural network for it

Unsupervised scene understanding

Use deep probabilistic models that feature tractable, deterministic inference
Actually, the main idea is to replace the VAEs within AIR by SPNs.

- Infinite mixture model
- Intractable density
- Intractable posterior

- "Large" but finite mixture model
- Tractable density
- Tractable marginals [Peharz et al., 2015]
- Tractable posterior [Vergari et al., 2017]
Sum-Product Attent-Infer Repeat

[Stelzner, Peharz, Kersting 2019]
Sum-Product Attent-Infer Repeat

[Stelzner, Peharz, Kersting 2019]
Sum-Product Attent-Infer Repeat

[Stelzner, Peharz, Kersting 2019]
There are strong investments into (deep) probabilistic programming.

RelationalAI, Apple, Microsoft and Uber are investing hundreds of millions of US dollars.
Since we need languages for Systems AI, the computational and mathematical modeling of complex AI systems.


Overall, AI/ML/DS indeed refine “formal” science, but …

- **AI is more than deep neural networks.** Probabilistic and causal models are whiteboxes that provide insights into applications.

- **AI is more than a single table.** Loops, graphs, different data types, relational DBs, … are central to data science and high-level programming languages for DS help to capture this complexity.

- **AI is more than just Machine Learners and Statisticians.**

Learning-based programming offers a framework for building systems that help to go beyond, democratize, and even automize traditional AI/ML/DS.
Kristian Kersting - Sum-Product
Probabilistic Programming: The Democratization of Machine Learning and Data Science
Not every Data Science machine is generative

\[
\min_{w, b, \xi} \mathcal{P}(w, b, \xi) = \frac{1}{2} w^2 + C \sum_{i=1}^{n} \xi_i \\
\text{subject to } \begin{cases} 
\forall i & y_i(w^T \Phi(x_i) + b) \geq 1 - \xi_i \\
\forall i & \xi_i \geq 0 \end{cases}
\]

Not everyone likes to turn math into code

Support Vector Machines
High-level Languages for Mathematical Programs

Write down SVM in „paper form.“ The machine compiles it into solver form.

```plaintext
#QUADRATIC OBJECTIVE
minimize: sum{J in feature(I,J)} weight(J)**2 + c1 * slack + c2 * coslack;

#labeled examples should be on the correct side
subject to forall {I in labeled(I)}: labeled(I)*predict(I) >= 1 - slack(I);

#slacks are positive
subject to forall {I in labeled(I)}: slack(I) >= 0;
```

Embedded within Python s.t. loops and rules can be used

RELOOP: A Toolkit for Relational Convex Optimization

Support Vector Machines