Uncertainty in Recurrent Decision Tree Classifiers

Stefan Wezel

Explainable Machine Learning

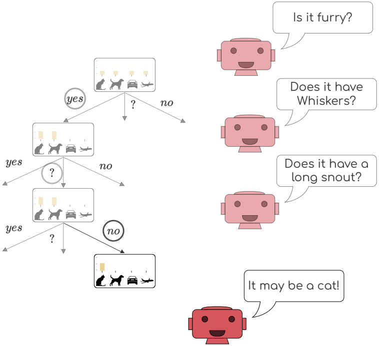
October 19, 2020

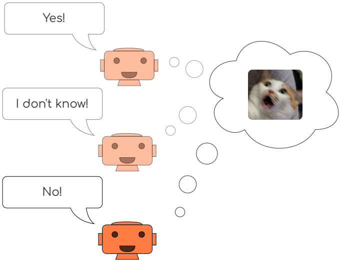
- There are a lot of architectures that perform great on image classification tasks
- Maybe, most prominently: ResNet
- However, they only yield a classification
- In many settings a classification is not worth much without the reasoning behind it

- Two agents
- One is asking questions and one is answering them
- The unfolding decision process is an interpretable tree

What?

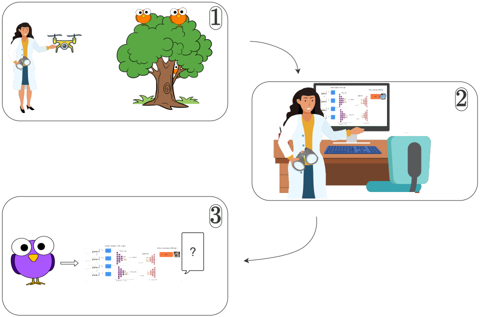
Recap





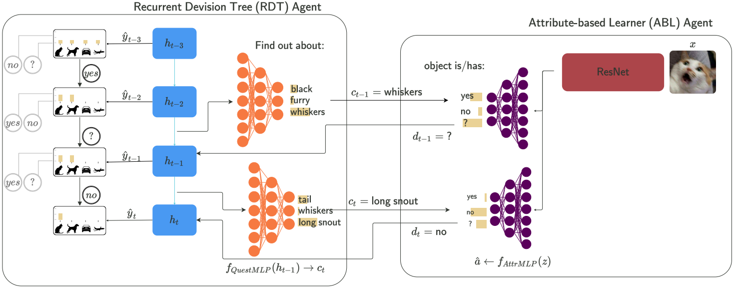
- Two agents
- One is asking questions and one is answering them
- The unfolding decision process is an interpretable tree

Why do we need uncertainty?
A Practical Example...



- The ornithologist is tasked to survey bird species, which she automates using a drone and computer vision software
- She uses our model to go through the vast amount of collected data
- Some bird species unknown to the model appear in the data. The model yields high uncertainty and the ornithologist can classify them manually

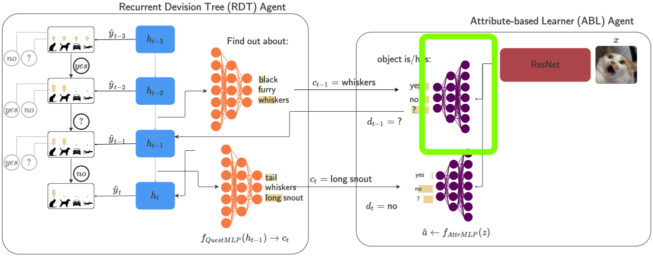
- some point



- The RDT can not see the images but only ask whether an attribute is present in the image
- The AbL can see the image and based on features answer the RDT's questions

How?

Do we get uncertainty information?



- We use dropout uncertainty estimation to retrieve uncertainty (more on that later)
- We want the AbL to say 'I don't know in the case of high uncertainty
- However, we want to keep the ResNet as feature extractor
- Thus, we use dropout in $f_{AttrMLP}$ (more on that later)
- After extracting features, we do n forward passes and compute variance

- We extract features from a given image using a ResNet
- A MLP maps extracted features to 'yes-no' answers indicating absence/presence of attributes
- It returns a tensor with the of shape number of attributes $\times$ decision size
- We get a discrete answer from our AbL though applying

$$TempSoftmax(log\ \pi) = \frac{exp((log\ \pi_i)/\tau)}{\sum_{j=1}^{K} exp((log\ \pi_j)/\tau)} = d_t \text{ on } log\ \pi$$

  which are the logit values for either 'Yes', or 'No' per attribute.

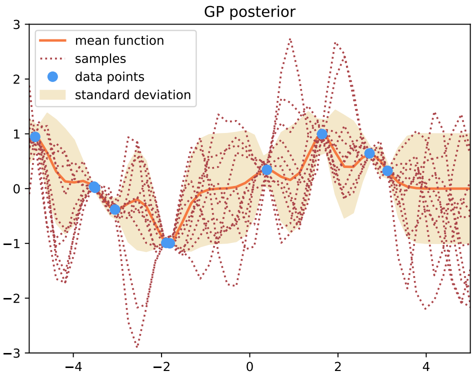
- The TempSoftmax serves as differentiable approximation to a one argmax returning a one-hot encoding

- LSTM
  - Hidden state based on previous hidden states and new answers
- Explicit Memory
  - Stores all questions and corresponding answers
  - This is our decision tree
- $f_{QuestMLP}$
  - Find next question to ask based on LSTM's hidden state
- $f_{ClassMLP}$
  - Make classification based on LSTM's hidden state

Training
the two agents

- We optimize for class and attribute accuracy
- $$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T}\left[(1-\lambda)\mathcal{L}_{CE}(y, \hat{y}_t) + \lambda\mathcal{L}_{CE}(\alpha_{y,c_t}, \hat{\alpha}_{c_t})\right]$$
- λ can be used to balance the two loss terms
- For all of our experiments, we use $\lambda = 0.2$

- summary bullet point 1

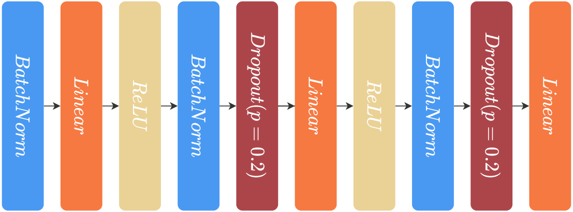


- Data points can be described by (infinitely) many functions
- A GP is a PDF over these functions
- Intuition: → a GP yields a probability for function values at any given index
- Parameterized by mean function and covariance function
- The variance resembles the model uncertainty where no data is given

- A GP can be seen as an infinitely wide neural network
- In turn, we can view a neural network with a finite amount of layers as an approximation to a GP
- To compute the posterior in GP's, methods of variational inference are used
  - GP objective gets turned into a minimization objective
  - For computing covariance matrix, we use Monte-Carlo integration
- This allows us to rewrite a GP's objective as to objective of a dropout neural network

Getting Uncertainty Information

through dropout uncertainty information



- We include dropout layers in our $f_{AttrMLP}$
- We tested different configurations and a combination of batchnorm and dropout worked best

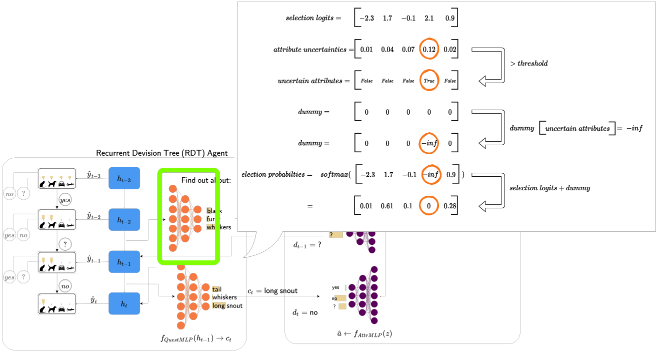
Using Uncertainty Information

as an inductive bias

- This allows us to get uncertainty information from our model
- But now, we need a way to use it
- we use two different strategies
  - We prevent the model from asking questions regarding uncertain attributes $\rightarrow$ remRDTC
  - We give the model the ability to answer with 'I don't know' as extended vocabulary $\rightarrow$ extRDTC
- For both strategies, we need to make sure the model does not use any gradients coming from uncertain attributes
- This ensures that the uncertainty information only remains an inductive bias and the model does not misuse it

Using Uncertainty Information

Removing uncertain attributes



- The output from $f_{QuestMLP}$ is an index that indicates the attribute in question
- In case, an attribute is deemed uncertain by the AbL, we replace selection logits at those indices with $-\infty$ This prevents the Gumbel softmax from picking such attributes as index

Using Uncertainty Information

Extending the vocabulary

$$
\begin{array}{cc} Y & N \end{array}
\qquad
?_{initial}
$$

$$
\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\begin{bmatrix} 0 \\ 0 \end{bmatrix}
$$

$$\Downarrow$$

$$
\begin{array}{c} \begin{array}{ccc} Y & N & ?_{initial} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{array}
\qquad
\begin{array}{c} ? \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{array}
$$

$$\Downarrow \qquad \searrow$$

$$
\begin{array}{c} \begin{array}{ccc} Y & N & ?_{initial} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{array}
*
\begin{array}{c} ! \\ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{array}
\quad \& \quad
\begin{array}{c} \begin{array}{ccc} Y & N & !_{initial} \end{array} \\ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array}
*
\begin{array}{c} ? \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{array}
$$

$$\Downarrow$$

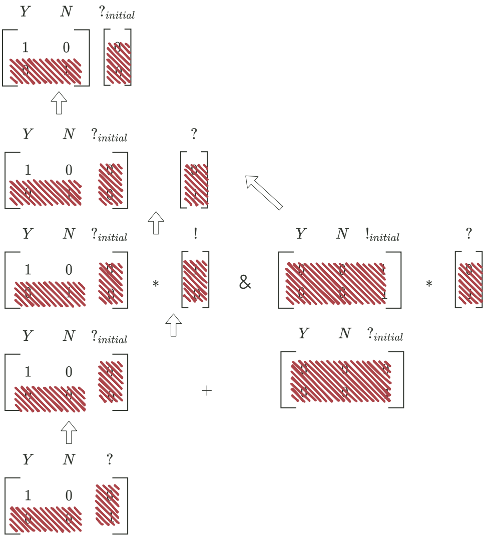
$$
\begin{array}{c} \begin{array}{ccc} Y & N & ?_{initial} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array}
\quad + \quad
\begin{array}{c} \begin{array}{ccc} Y & N & ?_{initial} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}
$$

$$\Downarrow$$

$$
\begin{array}{c} \begin{array}{ccc} Y & N & ? \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}
$$

Using Uncertainty Information

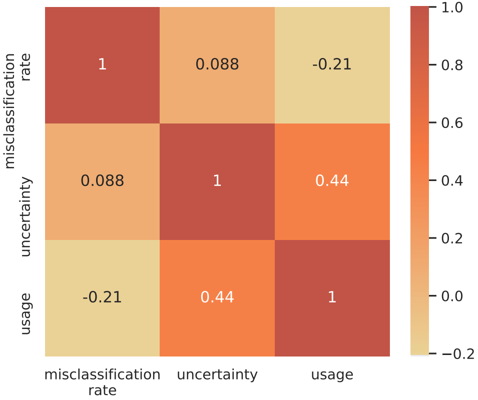
Extending the vocabulary



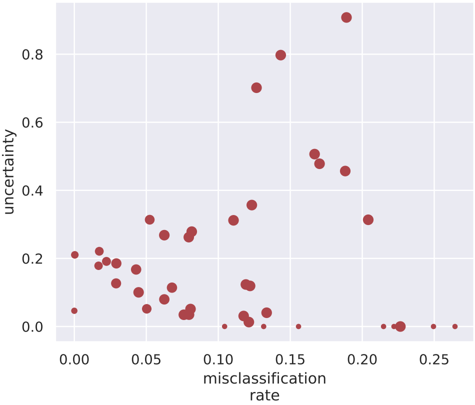
- some point

Experiments

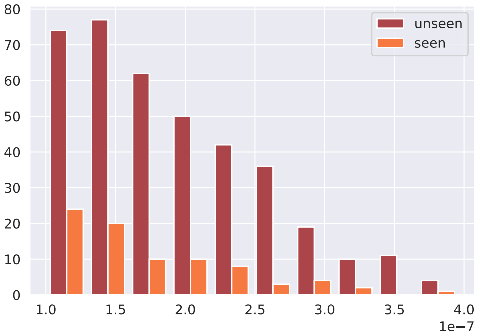
- Now, we can make our model aware of, and express its uncertainties
- We use this in our experiments to:
  - Investigate uncertainty and its relationship to other variables
  - Test our model on OOD data
  - Test the model's performance on benchmark datasets



misclassification rate
uncertainty
usage
1
0.088
-0.21
0.088
1
0.44
-0.21
0.44
1
misclassification rate
uncertainty
usage
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



0.8
0.6
0.4
0.2
0.0
uncertainty
0.00
0.05
0.10
0.15
0.20
0.25
misclassification rate



unseen
seen
80
70
60
50
40
30
20
10
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e−7

	AWA2	aPY	CUB
ResNet [HZRS16]	98.2± 0.0	85.1± 0.6	79.0± 0.2
DT	78.0± 0.4	64.3± 0.6	19.3± 0.3
dNDF[KFCRB15]	97.6± 0.2	85.0± 0.6	73.8± 0.3
RDTC[AA19]	98.0± 0.1	85.7± 0.7	78.1± 0.2
XDT	73.9± 0.9	59.9± 1.5	4.9± 1.3
aRDTC[AA19]	98.6	86.1	77.9± 0.6
remRDTC(ours)	98.7	86.4	77.7
extRDTC(ours)	98.7	85.4	77.8

	aRDTC [AA19]	Random Baseline	remRDTC	extRDTC
AWA2				
Class	98.6	98.5	98.7	98.7
Attribute	80.4	84.6	87.5	82.31
aPY				
Class	86.1	86.5	86.4	85.4
Attribute	86.4	86.2	87.6	87.12
CUB				
Class	77.9	76.8	77.7	77.8
Attribute	68.6	70.0	77.4	82.6

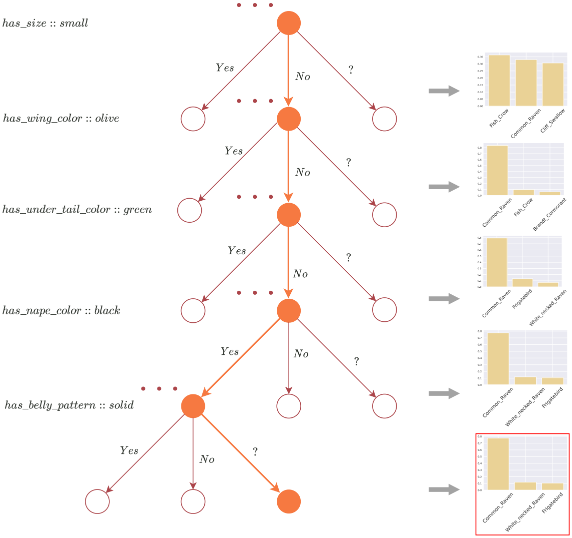
- We have seen that the RDTC uses only a small set of attributes that it rarely misclassifies
- Unseen classes yield higher uncertainty
- Both our proposed strategies perform well on benchmark datasets

Discussion

Looking back...

- The right kind of uncertianty?
- Uncertainty versus risk
- Beyond attribute uncertainty
-

Conclusions
A qualitative Example



[AA19] Stephan Alaniz and Zeynep Akata. Explainable observer-classifier for explainable binary decisions. arXiv preprint arXiv:1902.01780, 2019.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[KFCRB15] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In Proceedings of the IEEE international conference on computer vision, pages 1467–1475, 2015.