Training of mixed deep classifiers of symbolical rules and neural networks

TECHNISCHE UNIVERSITÄT DARMSTADT

Andreas Straninger – Masters Thesis

Supervisors: Prof. Dr. J. Fürnkranz, Dr. E. Loza Mencia

Motivation


Source: Deep Mind‘s work in 2016
https://deepmind.com/blog/deepmind-round-up-2016/

Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

Popularity of Deep Neural Networks

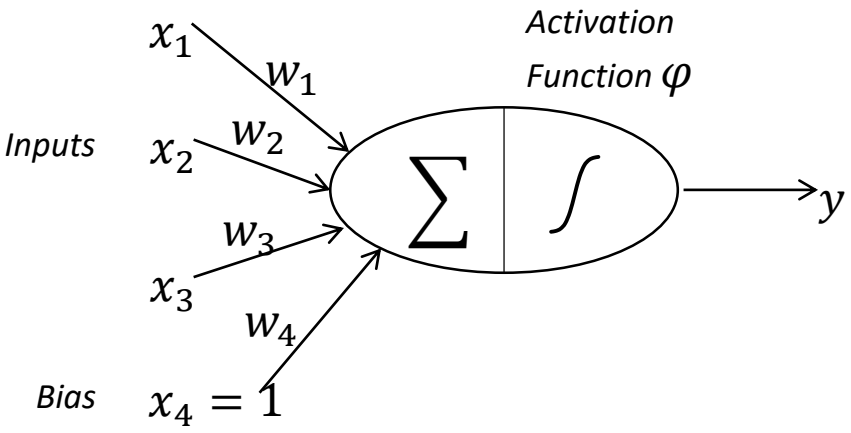
- High-Dimensional Problems
- Itermediate Concepts (Hidden Layers)

Symbolical Methods

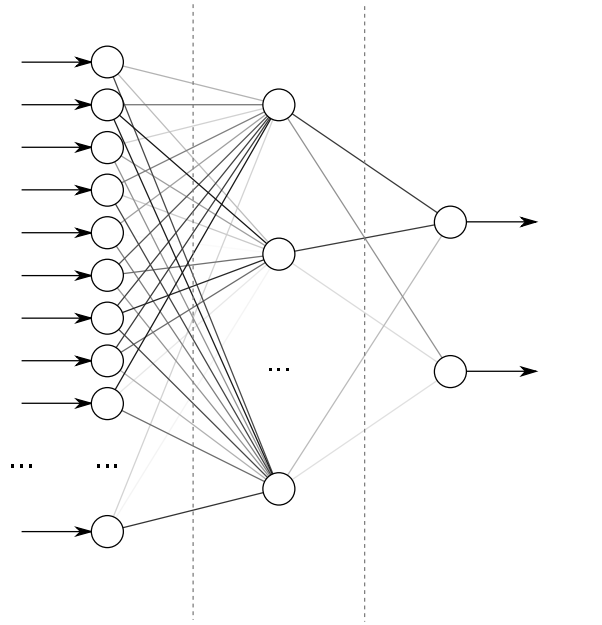
- Better interpretability
- But: No hierarchical topology

Goal: Train hierarchical symbolical Models

Neuron / Neural Network

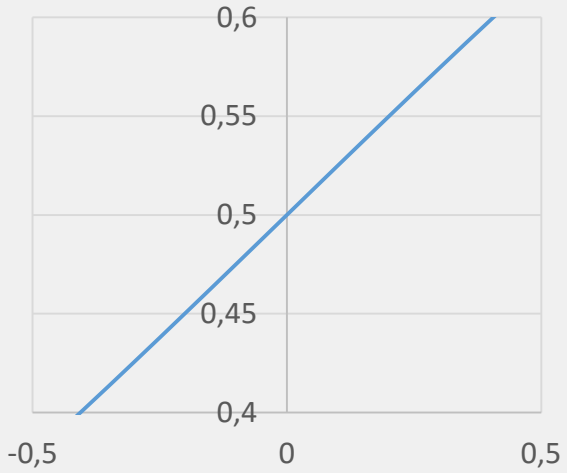
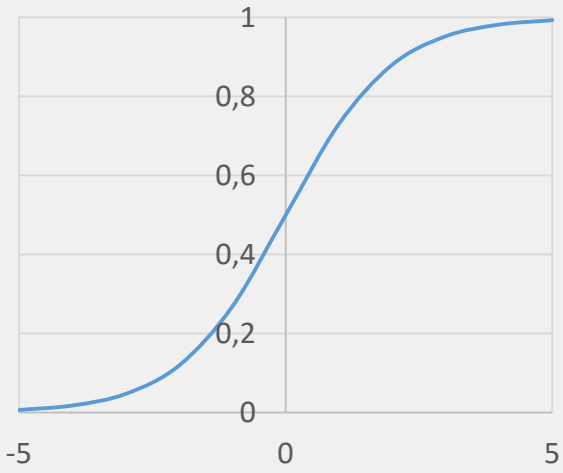
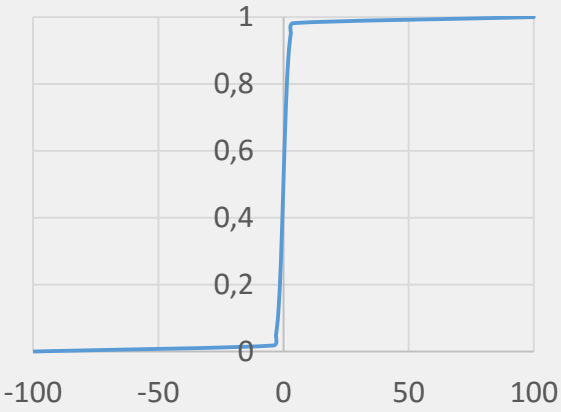


$$y = \varphi\left(\sum_{i=1}^{n} w_i x_i\right)$$



Softmax Activation Function

TECHNISCHE UNIVERSITÄT DARMSTADT

Linear approximation	Softmax	Step function approximation
	 $$\varphi(x) = \frac{1}{1 + e^{-x}}$$	
Example: Addition	Example: Sine	Example: Logical Operations
f(a, b) = a + b a, b ∈ [0, 1]	f(x) = sin(x) x ∈ [0, 2π]	f(a, b) = a & b a, b ∈ {0, 1}

Backpropagation

- Determine error at inner nodes
- Error at weights:

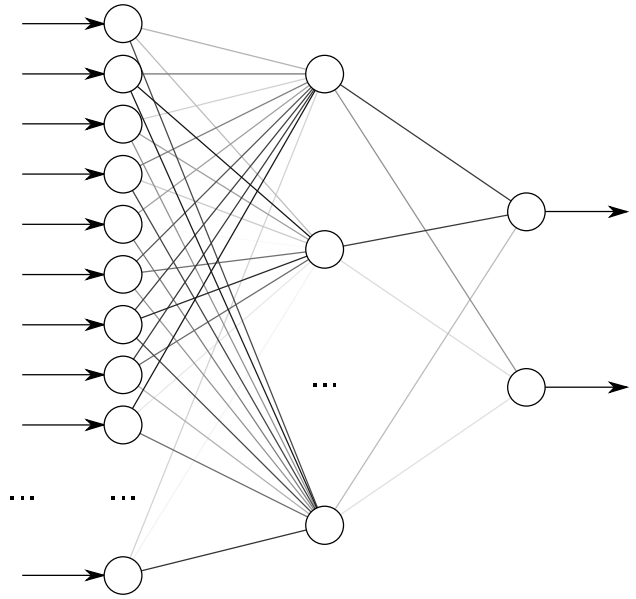
$$\frac{\partial E}{\partial w_{ij}} = in_i \delta_j$$

- Improve model by reducing the error of the weights:

$$\Delta w_{ij} = \alpha \frac{\partial E}{\partial w_{ij}}$$

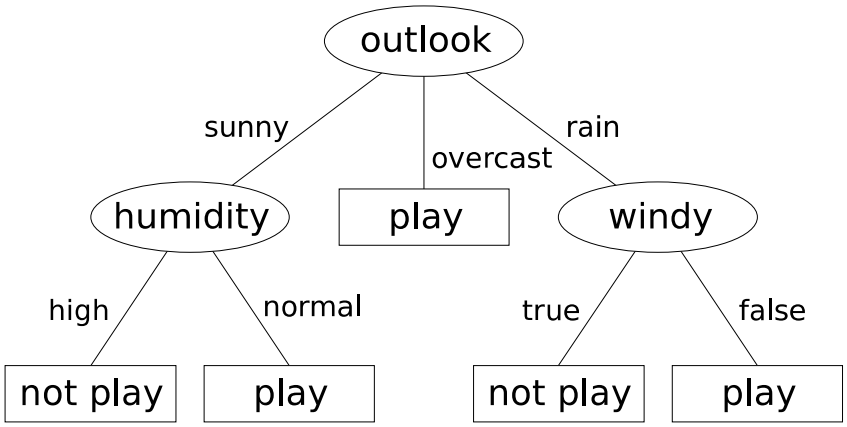
with lerning rate $\alpha \approx 10^{-2}$

$$in = x \rightarrow h_1 = \varphi(W \cdot in) \rightarrow out = \varphi_2 (W_2 \cdot h_1)$$



$$\delta_1 = \varphi'_1(W_1 in) W_1^T \delta_2 \leftarrow \delta_2 = \varphi'_2(W_2 h_1) W_2^T err$$

Decision Trees



Prediction:

- Follow a path from root node to a leaf
- Check attributes of a data sample at each node

Construction:

- Start with one leaf node
- Grow: Split a leaf nodes (Use feature that minimizes Entropy)

Reduce Overfitting (after training):

- Prune: Remove children of inner node
- Increases generalization of the model

Decision Lists

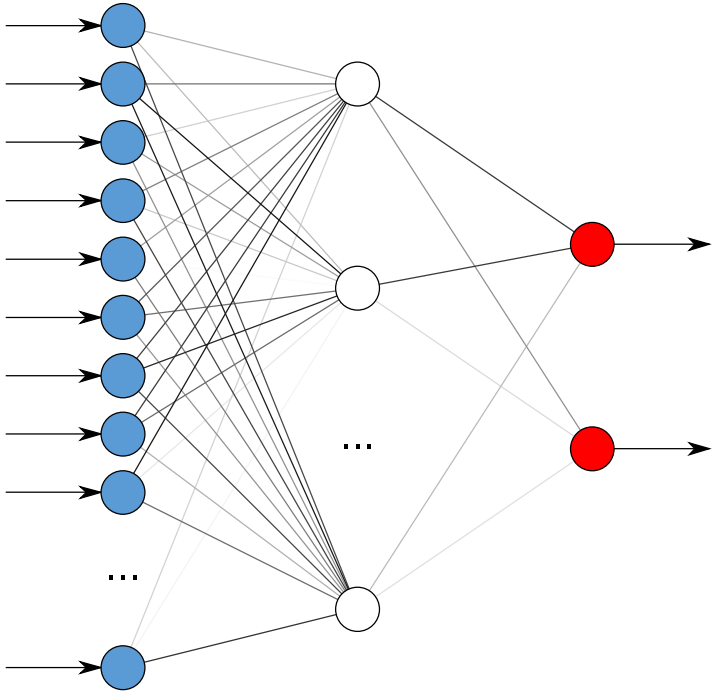
TECHNISCHE UNIVERSITÄT DARMSTADT

Example:
if outlook = sunny **and** humidity = high **then** not play
else if outlook = rain **and** windy = true **then** not play
otherwise play

Training: Iterative Method

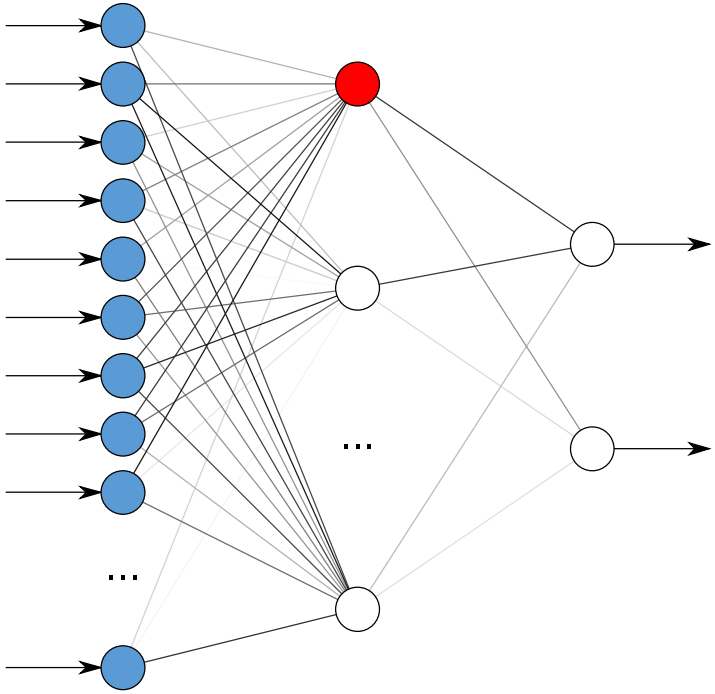
1. Grow Tree with 1 Leaf
2. Prune Tree
3. Tree → Rule
4. Remove covered training samples
5. If uncovered training samles exist, continue with 1

Generate Hierarchies (pedagogical)



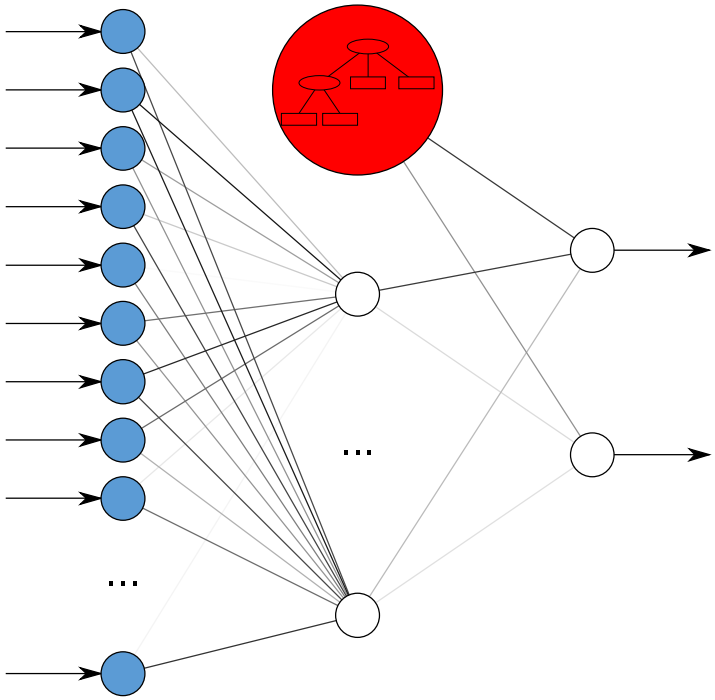
Attr_1	Attr_2	...	Attr_n	Class
0.65	0		1	2
0	1		0	1
0.4	1		0	2
1	1		.57	1
0	0		1	1
0	.33		0	1
0	.02		.98	2
1	1		0	2
...	...		...	...
1	0		1	2

Generate Hierarchies (pedagogical)



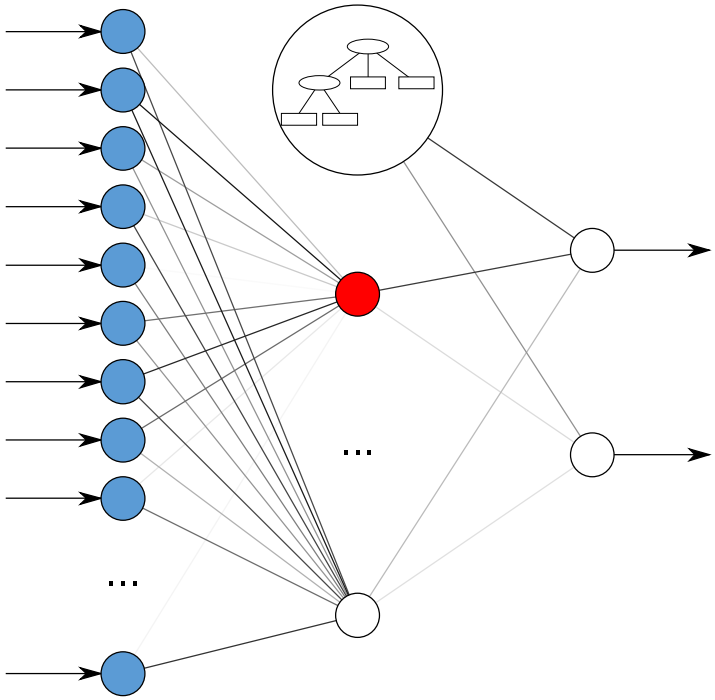
Attr_1	Attr_2	...	Attr_n	H_1
0.65	0		1	.28
0	1		0	.08
0.4	1		0	.19
1	1		.57	.44
0	0		1	.37
0	.33		0	.18
0	.02		.98	.09
1	1		0	.91
...	...		...	...
1	0		1	.67

Generate Hierarchies (pedagogical)



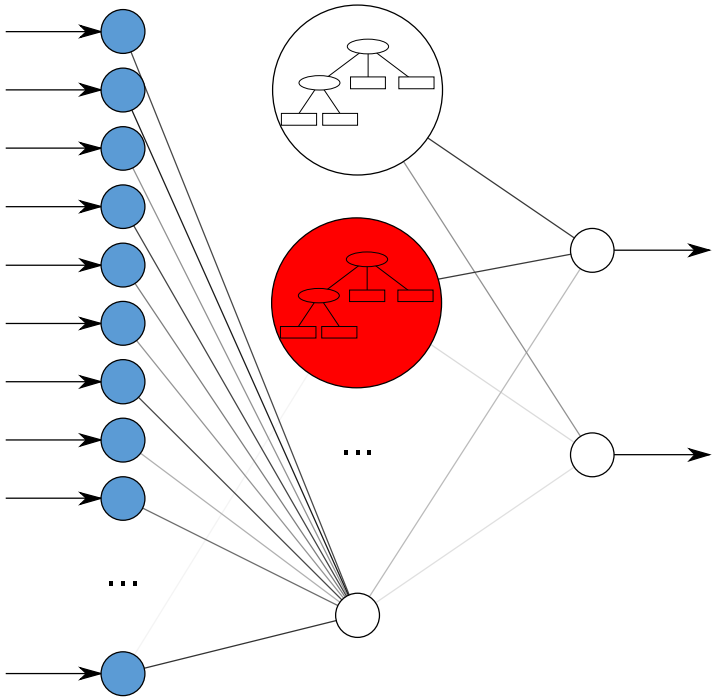
Attr_1	Attr_2	...	Attr_n	H_1
0.65	0		1	.28
0	1		0	.08
0.4	1		0	.19
1	1		.57	.44
0	0		1	.37
0	.33		0	.18
0	.02		.98	.09
1	1		0	.91
...	...		...	...
1	0		1	.67

Generate Hierarchies (pedagogical)



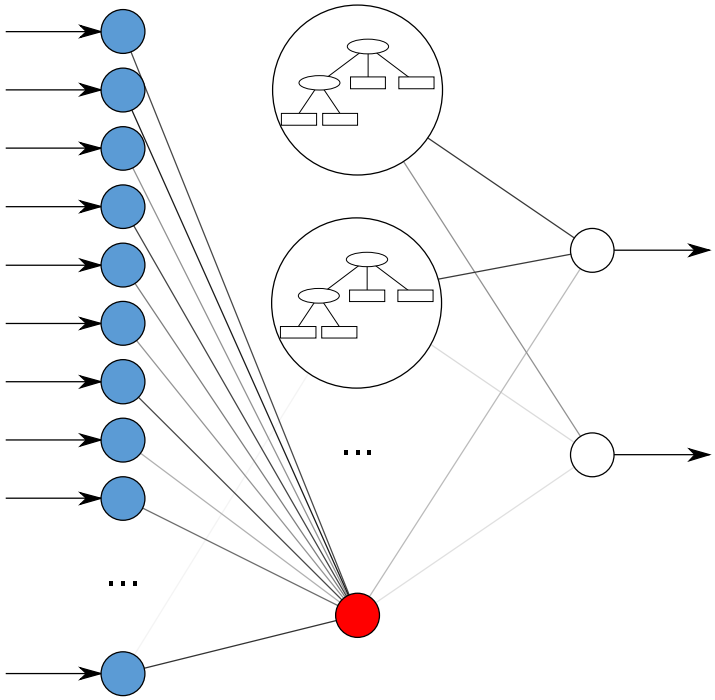
Attr_1	Attr_2	…	Attr_n	H_2
0.65	0		1	.1
0	1		0	.74
0.4	1		0	.14
1	1		.57	.65
0	0		1	.5
0	.33		0	.11
0	.02		.98	.98
1	1		0	.89
…	…		…	…
1	0		1	.95

Generate Hierarchies (pedagogical)



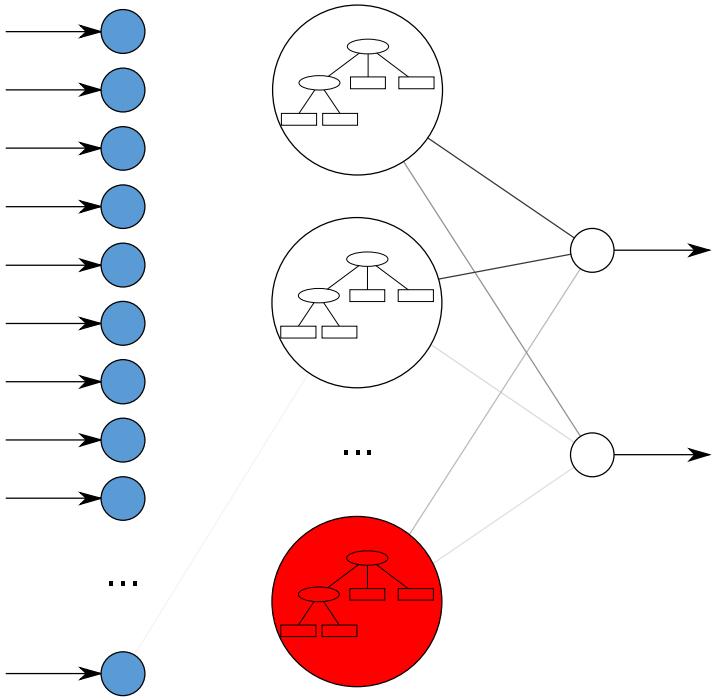
Attr_1	Attr_2	…	Attr_n	H_2
0.65	0		1	.1
0	1		0	.74
0.4	1		0	.14
1	1		.57	.65
0	0		1	.5
0	.33		0	.11
0	.02		.98	.98
1	1		0	.89
…	…		…	…
1	0		1	.95

Generate Hierarchies (pedagogical)



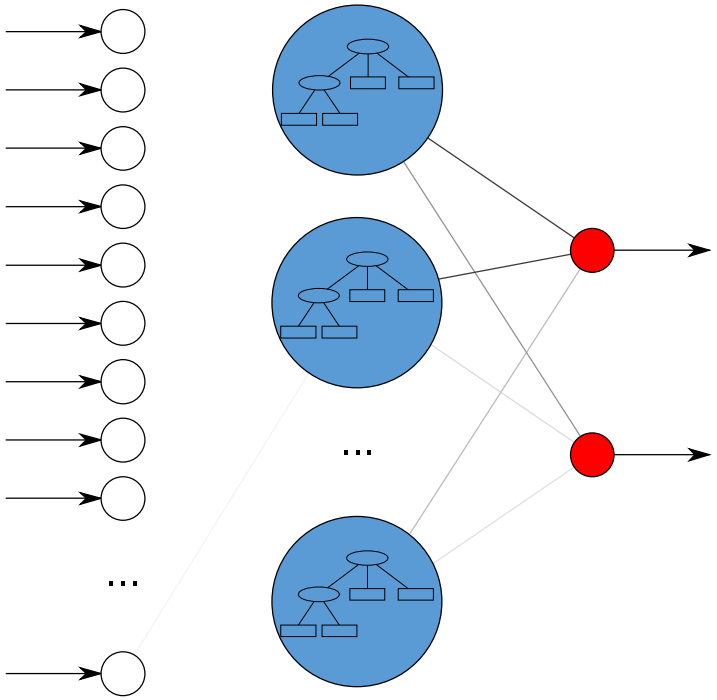
Attr_1	Attr_2	...	Attr_n	H_n
0.65	0		1	.24
0	1		0	.15
0.4	1		0	.27
1	1		.57	.17
0	0		1	.88
0	.33		0	.58
0	.02		.98	.41
1	1		0	.73
...	...		...	...
1	0		1	.68

Generate Hierarchies (pedagogical)



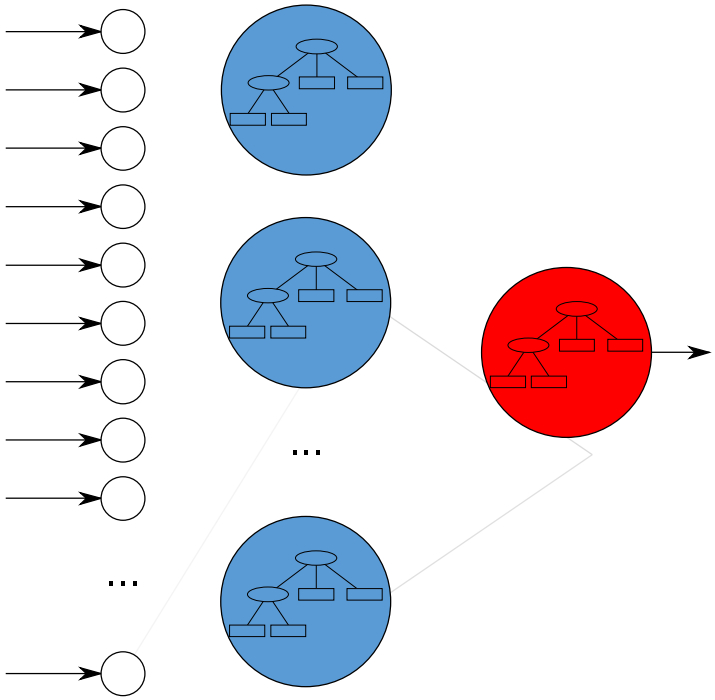
Attr_1	Attr_2	…	Attr_n	H_n
0.65	0		1	.24
0	1		0	.15
0.4	1		0	.27
1	1		.57	.17
0	0		1	.88
0	.33		0	.58
0	.02		.98	.41
1	1		0	.73
…	…		…	…
1	0		1	.68

Generate Hierarchies (pedagogical)



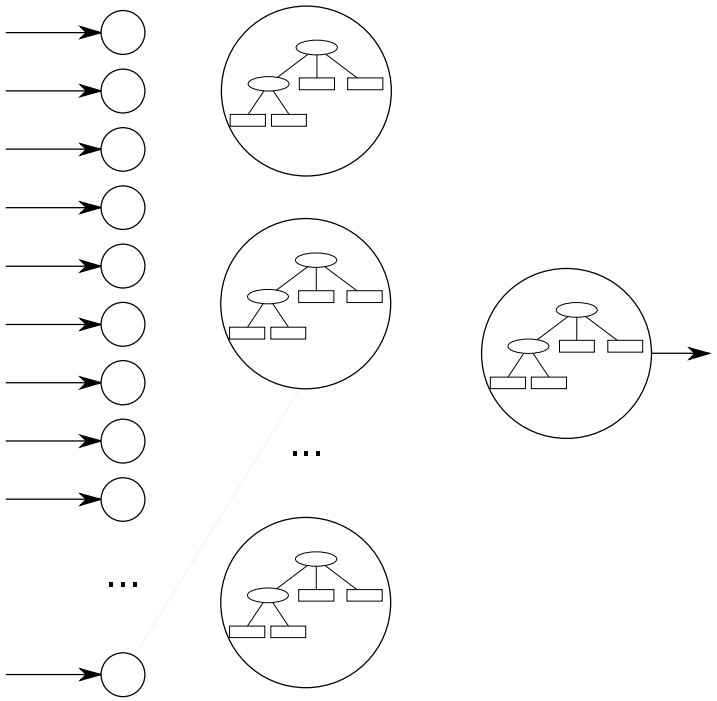
H_1 (DL)	H_2 (DL)	…	H_n (DL)	Class
.28	.1		.24	2
.08	.74		.15	1
.19	.14		.27	2
.44	.65		.17	1
.37	.5		.88	1
.18	.11		.58	1
.09	.98		.41	2
.91	.89		.73	2
…	…		…	…
.67	.95		.68	2

Generate Hierarchies (pedagogical)



H_1 (DL)	H_2 (DL)	…	H_n (DL)	Class
.28	.1		.24	2
.08	.74		.15	1
.19	.14		.27	2
.44	.65		.17	1
.37	.5		.88	1
.18	.11		.58	1
.09	.98		.41	2
.91	.89		.73	2
…	…		…	…
.67	.95		.68	2

Generate Hierarchies (pedagogical)



Result

- Hierarchical model
- Outputs from one layer are passed to the next layer

Drawbacks

- Isolated training of symbolical units
- Large symbolical models for approximating neurons

Related: Boosting

TECHNISCHE UNIVERSITÄT DARMSTADT

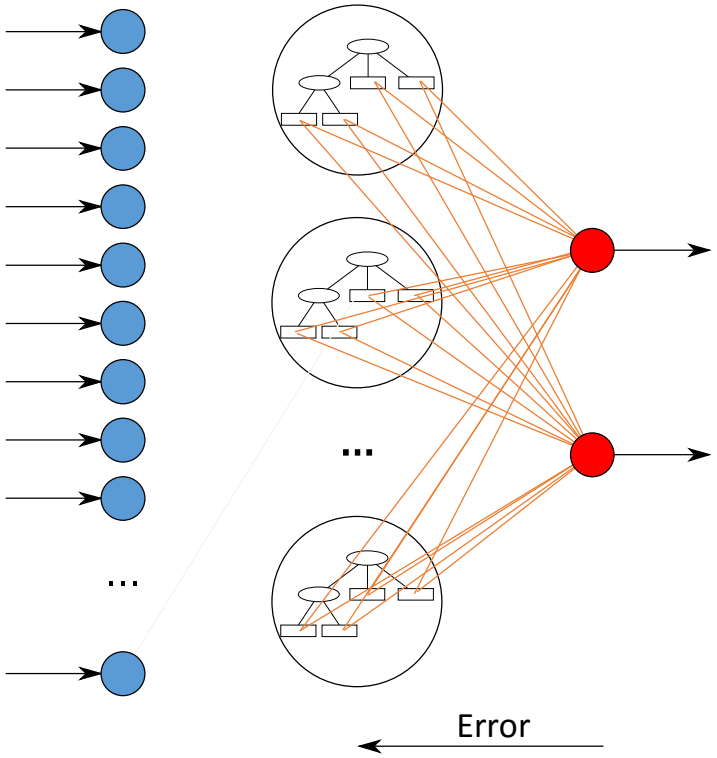
Boosting

- Train set of classifiers and perform a voting
- New Classifiers are trained with training samples that have been misclassified

Gradient Boosting

- Two-Layer Model
  - Layer 1: Classifiers
  - Layer 2: Connection Layer
- Advantage
  - Weighted voting in connection layer
  - Use gradient descent to determine error of Classifiers

Gradient Tree Boosting



H_1	H_2	...	H_n	dL_1	dL_2	...	dL_n
0.65	0		1	.47	.42		.16
0	1		0	.07	-.26		-.15
0.4	1		0	.16	.17		.47
1	1		.57	.02	.06		-.24
0	0		1	-.10	.25		.46
0	.33		0	.17	.25		-.13
0	.02		.98	.27	.46		-.43
1	1		0	.23	.42		-.25
...	...		...	...	...		...
1	0		1	-.08	.01		-.26

On-line Training

TECHNISCHE UNIVERSITÄT DARMSTADT

Idea: Train a mixed network

- Neural Layers: Backpropagation
- Successively replace Neural Layers by Symbolical layers

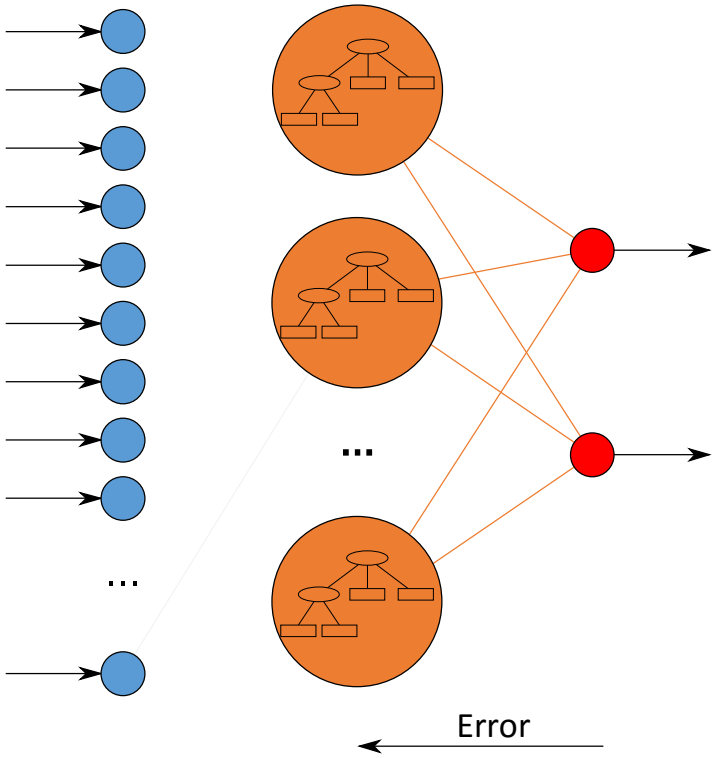
Backpropagation to…

- … Neural Layers: Determine error at the weights
- … Symbolical Layers: Determine error at the output of the symbolical units:

$$\frac{\partial E}{\partial h_j} = \delta_j$$

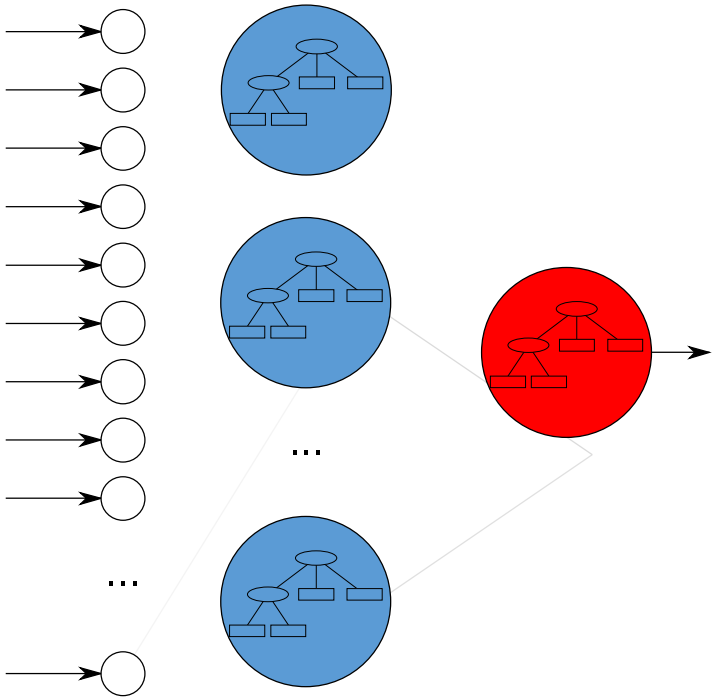
- Grow operation: Determine average ∂E for the implied ∂h

On-line Training



H_1	H_2	...	H_n	dH_1	dH_2	...	dH_n
0.65	0		1	.47	.42		.16
0	1		0	.07	-.26		-.15
0.4	1		0	.16	.17		.47
1	1		.57	.02	.06		-.24
0	0		1	-.10	.25		.46
0	.33		0	.17	.25		-.13
0	.02		.98	.27	.46		-.43
1	1		0	.23	.42		-.25
...	...		...	...	...		...
1	0		1	-.08	.21		-.26

On-line Training



H_1 (DL)	H_2 (DL)	...	H_n (DL)	Class
1	1		1	2
0	1		0	1
0	0		1	2
0	0		1	1
0	1		0	1
1	0		1	1
1	1		1	2
0	1		1	2
...	...		...	...
1	1		0	2

On-line Training with Decision Lists

TECHNISCHE UNIVERSITÄT DARMSTADT

Final model:

- Outputs of a layer are used as inputs of the next layer
  - Therefore: Binary outputs to make range estimation of next layer obsolete

Decision Tree:

- Choice of root node is crucial
  - Cannot be pruned
  - May become bad if target output changes

Simple Realization:

- Use Decision Lists
- Grow / prune condition if improvement can be archieved
  - No rule-wise training / misclassification error instead of information gain

Stochastic grow operations

TECHNISCHE UNIVERSITÄT DARMSTADT

Expensive to calculate all possible refinements

- Find the best grow operation in $\theta(|Inputs| \cdot |Hidden\ units| \cdot |Samples|)$
  - For $|Conditions| \sim |Inputs|$:
    Training of the best rules in $\theta(|Inputs|^3 \cdot |Samples|)$
  - Neural Networks:
    Training of one epoch in $\theta(|inputs|^2 \cdot |Samples|)$

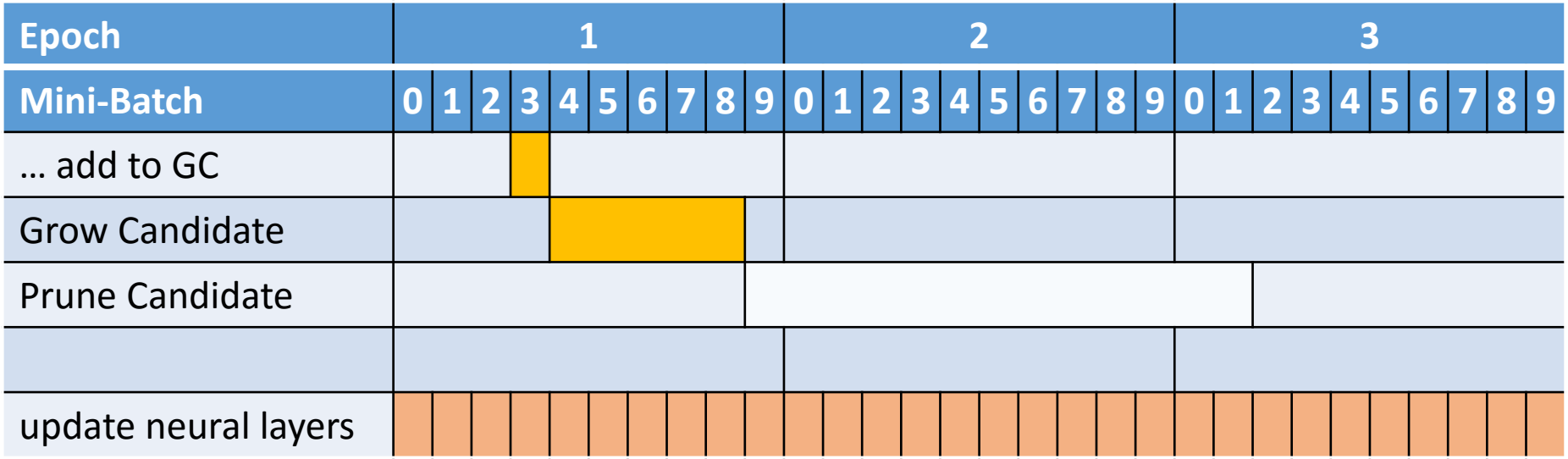
Idea: Find a good grow operation instead of the best

- Grow Candidates: Measure most promising refinements for some time
- $p_{Add\ to\ GC} \sim imp(g, s)$ for $g \in Grow\ Operations, s \in Samples$

Timing

TECHNISCHE UNIVERSITÄT DARMSTADT

Order of adding and removing a condition



Number of Conditions:

- Each epoch, add one Condition to each rule (average)

Normalization

TECHNISCHE UNIVERSITÄT DARMSTADT

Refinement

- Neural Netwok: all weights
- DL Layer: only most promising

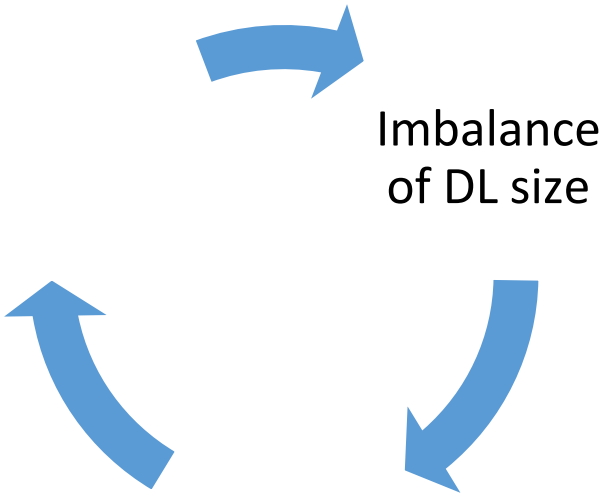
  → Imbalance

Normalization

TECHNISCHE UNIVERSITÄT DARMSTADT

Refinement

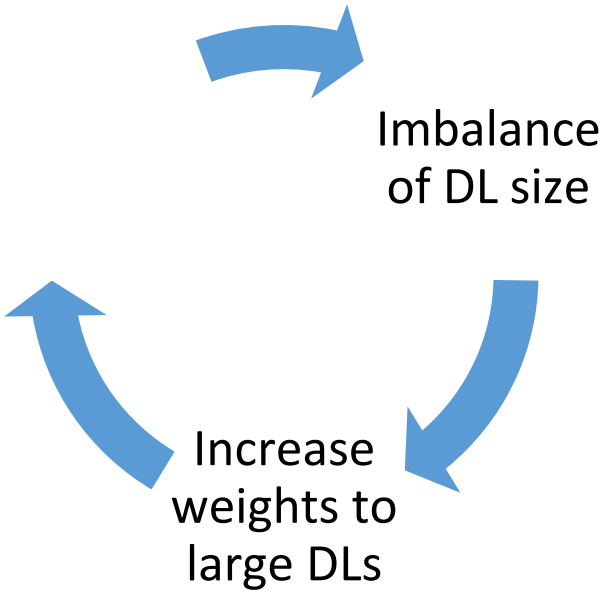
- Neural Netwok: all weights
- DL Layer: only most promising
  - Imbalance



Normalization

Refinement

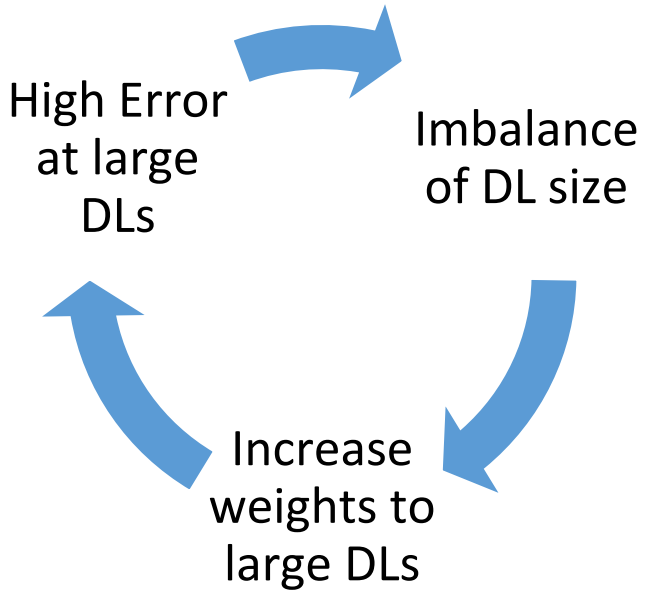
- Neural Netwok: all weights
- DL Layer: only most promising
  - Imbalance



Normalization

Refinement

- Neural Netwok: all weights
- DL Layer: only most promising
  - Imbalance



Normalization

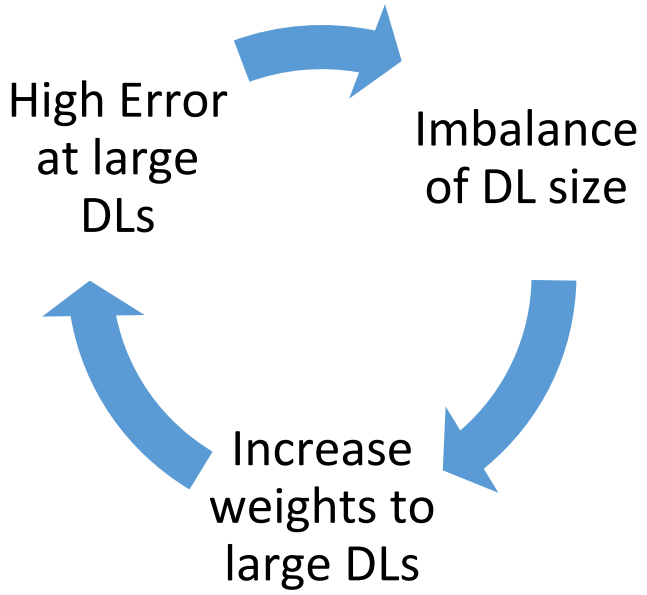
TECHNISCHE UNIVERSITÄT DARMSTADT

Refinement

- Neural Netwok: all weights
- DL Layer: only most promising
  - Imbalance

Countermeasures:

- Normalize Error with sum of incoming weights



Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

Dataset MNIST:

- 70000 labeled hand-written digits (28x28 Pixels)
- Provided Training and Test Sets

Method:

- Pedagogical approach
  - Regression at hidden units
  - Classification at output units
- On-line approach
  - On-line training at hidden units
  - Off-line classification at outputs

Results

TECHNISCHE UNIVERSITÄT DARMSTADT

Algorithm	Topology	Number of Conditions	Number of Conditions (last layer)	Accuracy
Jrip		3451		89.65%
Neural Network	784-64-10			96.84%
	784-256-10			97.51%
	784-256-64-10			97.82%
	784-256-128-64-10			97.3%
Pedagogical (Jrip)	784-64-10	128000	1661	91.35%
	784-256-10	512000	1530	92.28%
	784-256-64-10	640000	798	93.29%
	784-256-128-64-10	1120000	562	93.52%
Mixed Network + Jrip	784-64-10	1038	2674	90.13%
	784-256-10	3978	2092	91.81%
	784-256-64-10	3856-895	2363	90.44%
	784-256-128-64-10	3722-1433-769	1987	89.12%

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

Contribution:

- On-line training scheme for hierarchical symbolical models
- Reduced model size compared to off-line training
- Problems: Scaling with num

Future Works:

- Use continuous regression at hidden units (need to implement range esimation)
- Train with different algorithms and optimization metrics
  (e.g. decision trees and information gain)
- Implement optimization techniques for decision lists (see RIPPER algorithm)

End

End

TECHNISCHE UNIVERSITÄT DARMSTADT

End

End

Symbolical Backpropagation

Idea:

- Activate hidden symbolical units
- How does output j change, if input i changes?
  - $f(0) = 0, f(1) = 1 \rightarrow f' = 1$
  - $f(0) = 1, f(1) = 0 \rightarrow f' = -1$
  - Otherwise: $f' = 0$

Realization:

- Voting of backpropagated error of rule layers and neural layers