UNIVERSITÄT OSNABRÜCK

University of Osnabrück, Institute of Cognitive Science

Machine Learning

3 – Decision Tree Learning

SS 2018

Gunther Heidemann

Overview

1. Basics of decision trees:
   - Decision tree representation
   - ID3 learning algorithm:
     - Entropy, information gain
     - Selecting the next attribute
   - Comments on ID3:
     - Hypothesis search by ID3
     - Inductive bias in ID3, Occam’s razor
2. Extensions of ID3
3. Application example

UNIVERSITÄT OSNABRÜCK

Idea

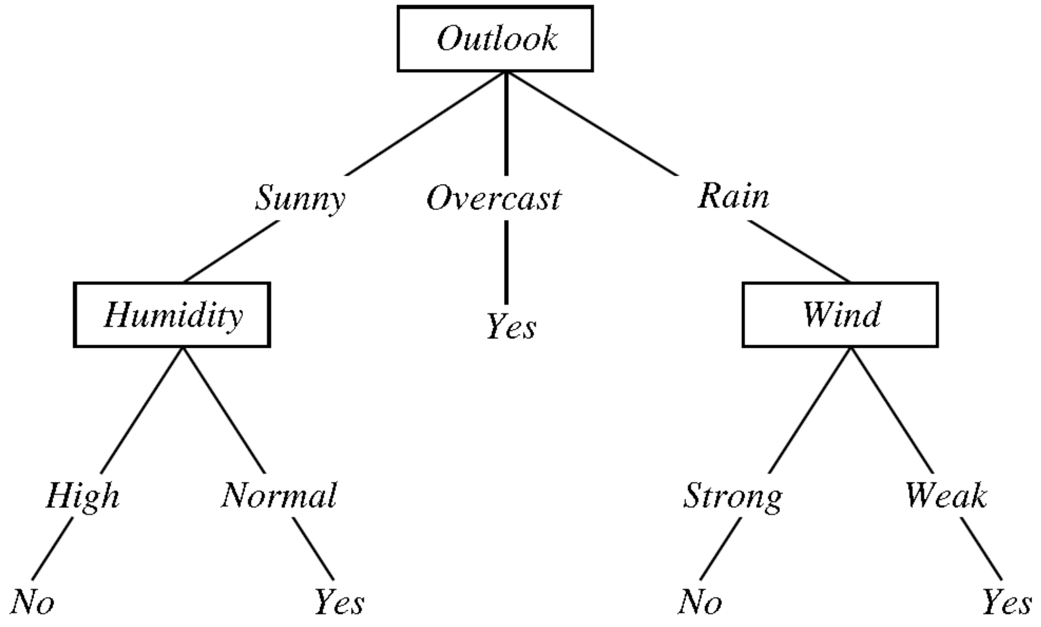
- Decision trees for learning discrete valued target functions.
- Classifies data by a sequence of interpretable steps.
- Decision tree representation:
  - Each internal node tests an attribute
  - Each branch corresponds to one attribute value
  - Each leaf node assigns a classification
- A decision tree
  - can be written as
  - is a nice form of

  } disjunction of conjunctions of attribute values
- So a decision tree can be written as a set of rules.
- Widely used in many domains.
- There is a simple core algorithm for building a decision tree plus refinements, most well known are ID3 and C4.5.

Example

Decision tree for concept *PlayTennis* for given attributes of the day (*Outlook, Temperature, Humidity, Wind*)

To classify a day, start at root, go to branch according to the value of attribute *Outlook* etc. until leaf node with classification is reached.



[M]

Tree corresponds to

$\neg($

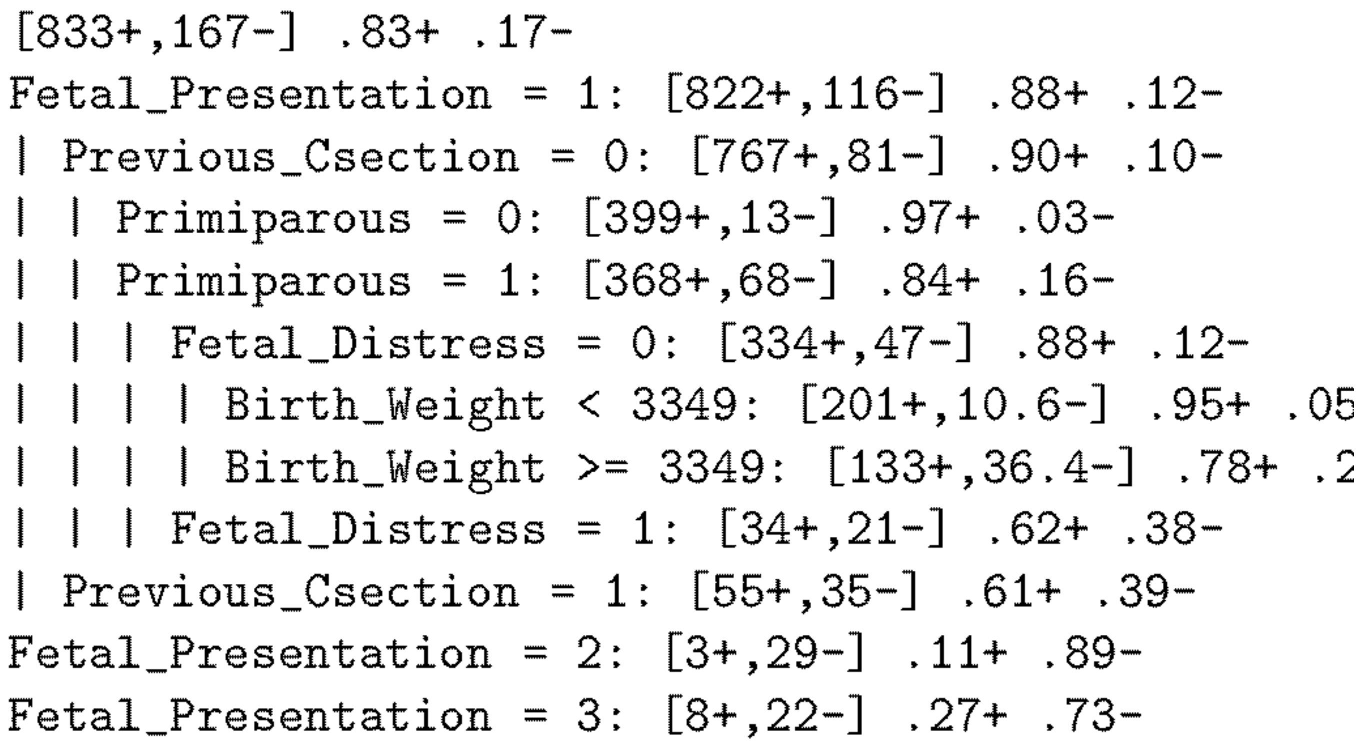
$(Outlook=Sunny \wedge Humidity=High)$

$\vee\ (Outlook=Rain \wedge Wind=Strong)$

$)$

Example

A decision tree to predict C-section risk, learned from medical records of 1000 women, expressed as rules. Negative examples are C-sections. Tree gives insight into importance of attributes.



```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .2
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

[M]

When to consider decision trees

Features of the problem

- Instances are describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

Top-down induction of decision trees

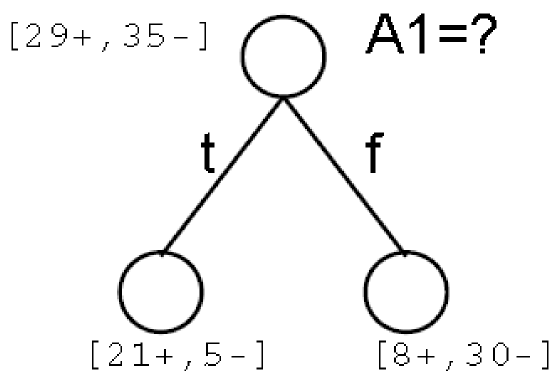
Learning a decision tree means finding the best order to ask the attribute values.

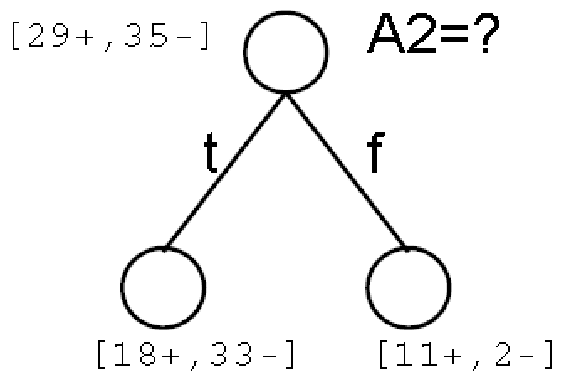
- "Best" means we want a small tree.
- Most algorithms employ top down, greedy search among possible decision trees.
- Core algorithm is **ID3** (Iterative Dichotomizer 3) (Quinlan, 1986).
- Idea: Find best attribute by the distribution of its values over the examples.
- Put best attribute at the root, one branch for each of its values.
- Then search second best attribute for each of the branches and so on.

Top-down induction of decision trees

Main loop of core algorithm:

1. *A* ← the "best" decision attribute for next *node*.
2. Assign *A* as decision attribute for node.
3. For each value of *A*, create new descendant of *node*.
4. Sort training examples to leaf nodes.
5. If the training examples are perfectly classified then STOP
   else iterate over new leaf nodes.





[M]

UNIVERSITÄT OSNABRÜCK

Entropy

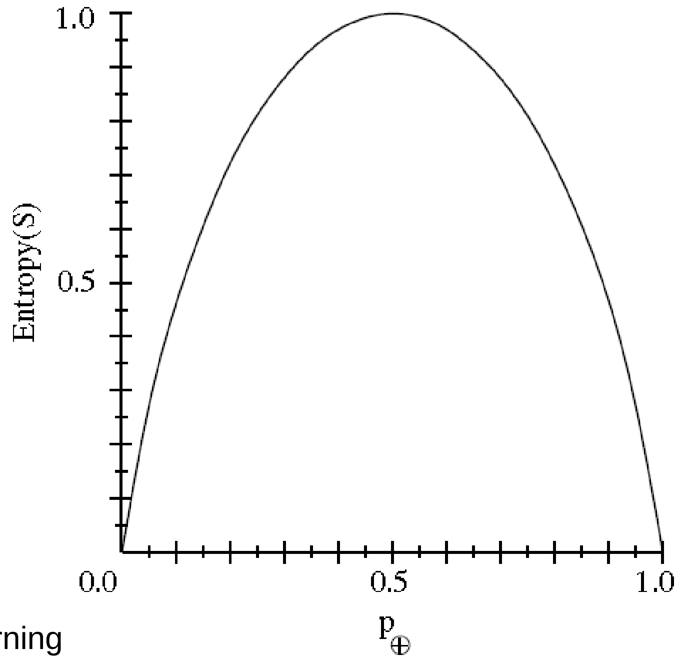
The decision which attribute to choose is based on entropy:

- S is a set of training examples.
- p_+ is the proportion of positive examples in S.
- p_- is the proportion of negative examples in S, $p_+ + p_- = 1$.
- Entropy E measures the “impurity” of S :

$$E(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

- Define $0 \log 0 = 0$.
- For c different values:

$$E(S) = -\sum_{i=1\ldots c} p_i \log_2 p_i$$



[M]

Entropy

$E(S)$ = expected number of bits needed to encode class membership (**+** or **–**) of a randomly drawn member of S (using the optimal, shortest-length code)

Why?

Information theory: optimal length code assigns $-\log_2 p$ bits to a message having probability p.

So, the expected number of bits to encode **+** or **–** of a random member of S is

$$E(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Information gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

$$= E(S) - \sum_{v \in Values(A)} E(S_v) \cdot |S_v| / |S|$$

where S_v is the subset of S for which A has value v.

Explanation:

$E(S)$ = Entropy before evaluation of A.

$\sum \ldots$ = Entropy after evaluation of A.

$|S_v| / |S|$ = Weighting by the fraction of examples that belong to S_v.

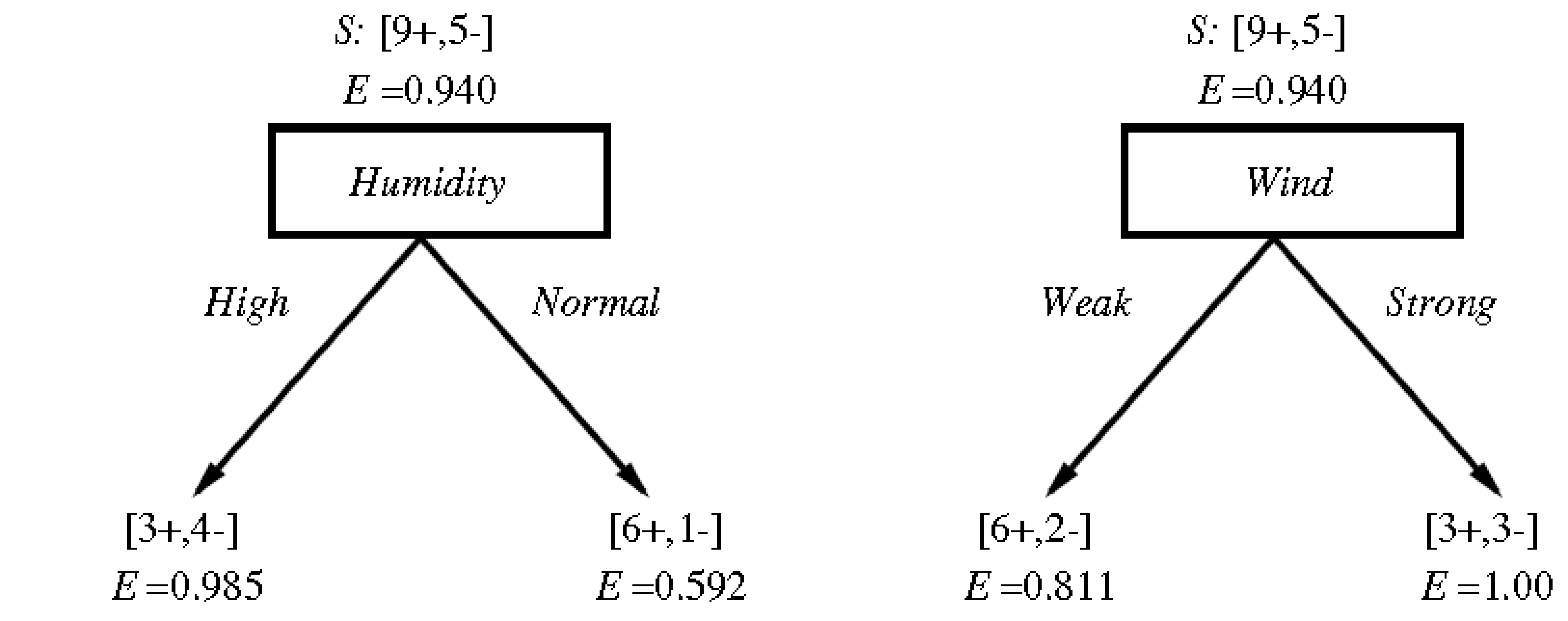
Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

[M]

Selecting the next attribute

Which attribute is the better classifier ?



Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

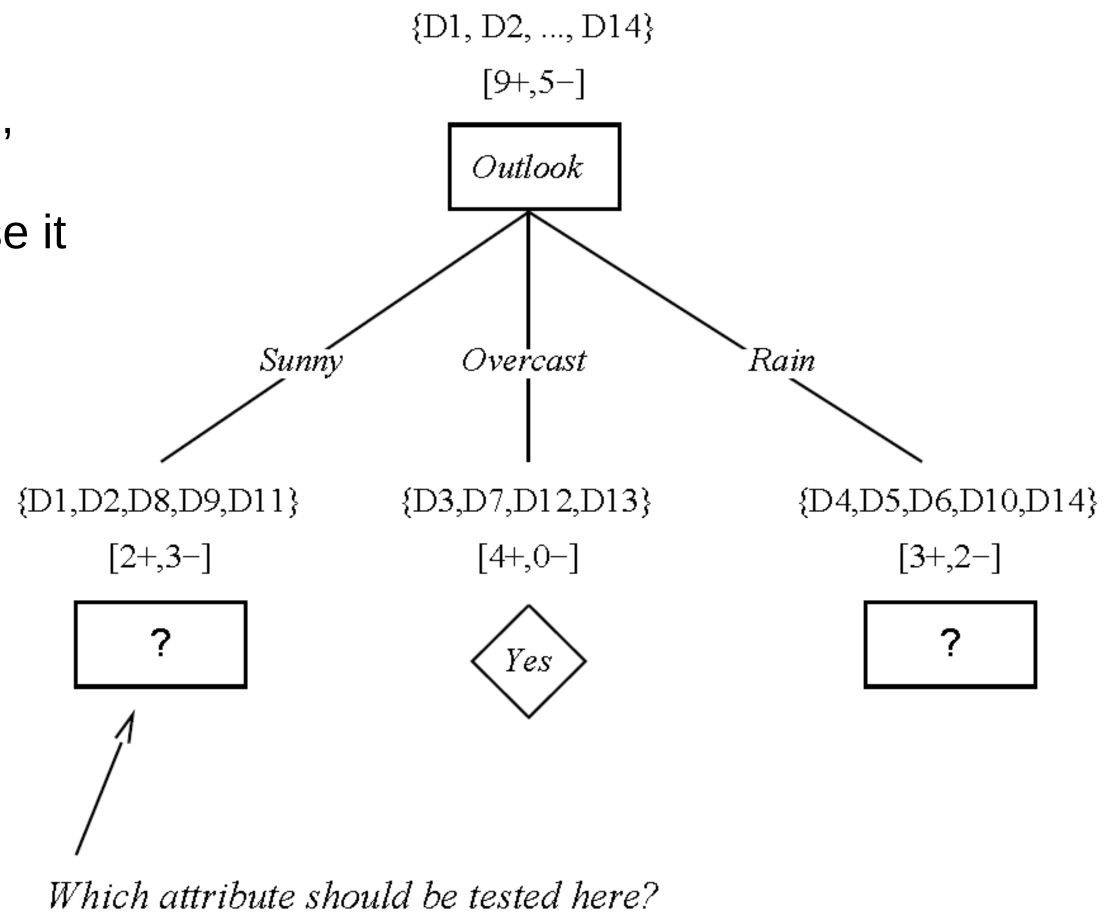
= .940 - (8/14).811 - (6/14)1.0
= .048

[M]

Humidity gains more information (relative to the target classification).

Selecting the next attribute

Of the four attributes, *Outlook* gains most information → choose it as the root node.



$S_{sunny} = \{D1,D2,D8,D9,D11\}$

$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

$Gain\ (S_{sunny}, Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

[M]

Hypothesis space search by ID3

- ID3 searches the tree that builds up possible decision trees.
- Hypothesis space is complete!

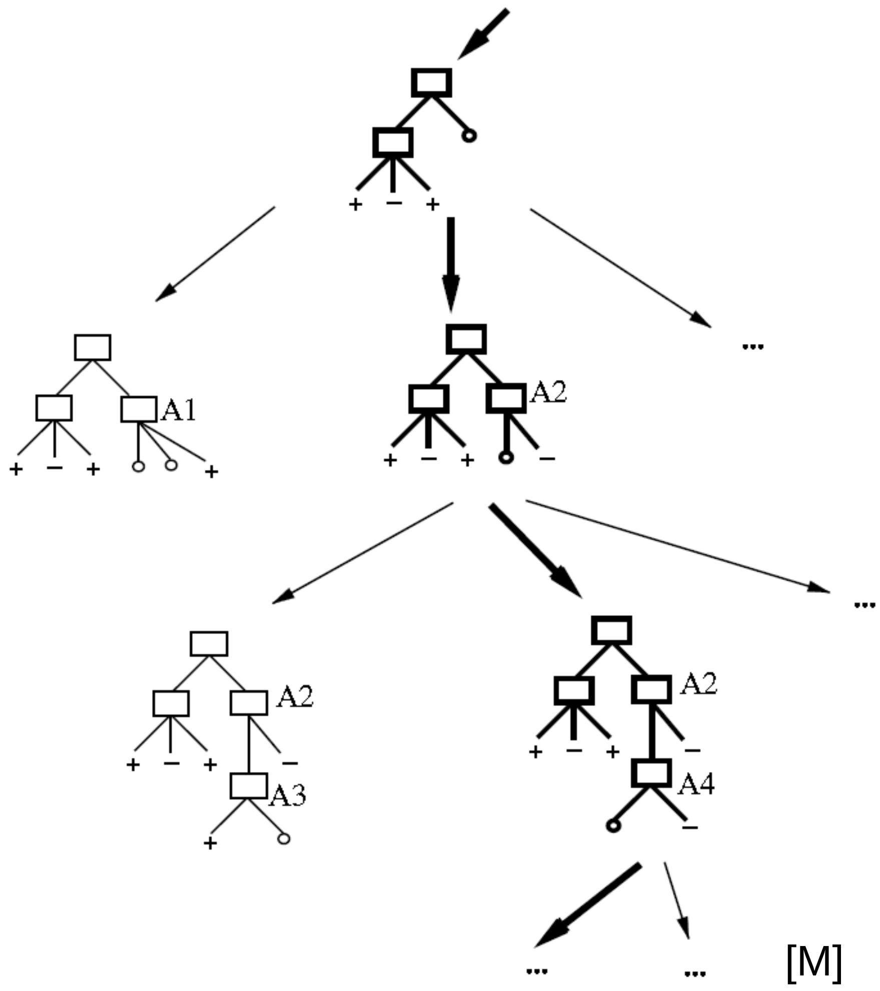
  → Target function is surely in there
- Outputs a single hypothesis, not a version space

  → No queries to resolve among competing hypotheses.
- No backtracking (reconsidering earlier choices)

  → Ends up in local optima
- Statistically-based search choices

  → Robust to noisy data

  → Needs all examples for every choice, no incremental learning.



[M]

Inductive bias in ID3

How does ID3 generalize from examples?

H is the power set of instances *X* → Unbiased search ?

No, because

- short trees are preferred,
- high information gain attributes near root are preferred.
- Bias is a *preference* for some hypotheses, rather than a *restriction* of hypothesis space *H*.
- Conforms to Occam's razor

As for many other algorithms, the bias of ID3 is difficult to describe exactly, because it results from the strategy, not the definition of the hypotheses space.

Occams's razor

William of Occam, ~ 1320 :

Prefer the shortest hypothesis that fits the data !

Argument in favor:

- There are fewer short hypotheses than long hypotheses (combinatorics).
- Short hypothesis that fits data is unlikely to be coincidence.
- Long hypothesis that fits is more likely to be coincidence.

Argument opposed:

- There may be many ways to define small sets of hypotheses, e.g., all trees with a prime number of nodes that use attributes in numerical order. If one of these fits, it would be likely to be “good”!
- What is so special about small sets based on *size* of hypothesis? Size depends on the representation!

Occams's razor

Remark:

Occam´s razor is also used to decide between equivalent descriptions of different complexity:

"Sun is the center, planets on ellipses"

versus

"Earth is the center, sun and planets on cycles + epicycles"

Extensions of ID3

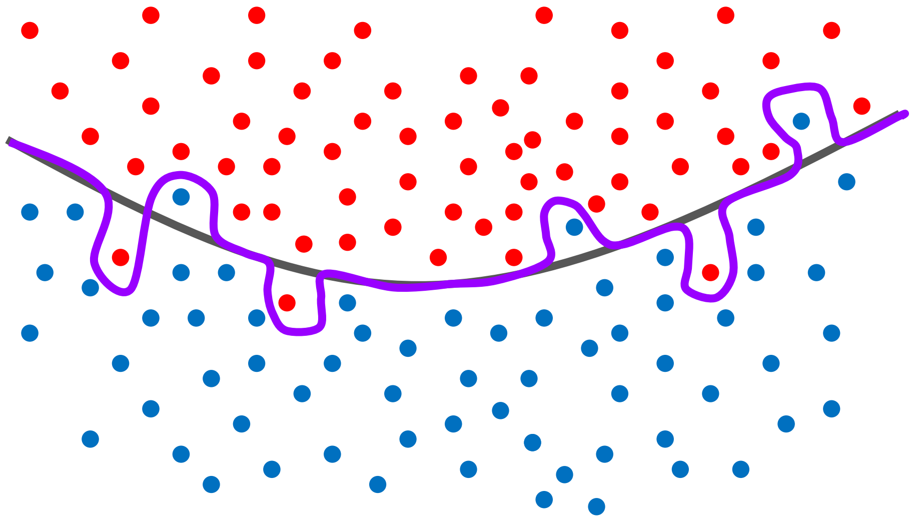
Extensions of ID3

Most of the extensions are part of the C4.5 algorithm (Quinlan, 1993).

Overview:

- Avoiding overfitting
  - reduced error pruning
  - rule post pruning
- Continuous valued attributes
- Attributes with many values
- Attributes with costs
- Missing attribute values

Overfitting

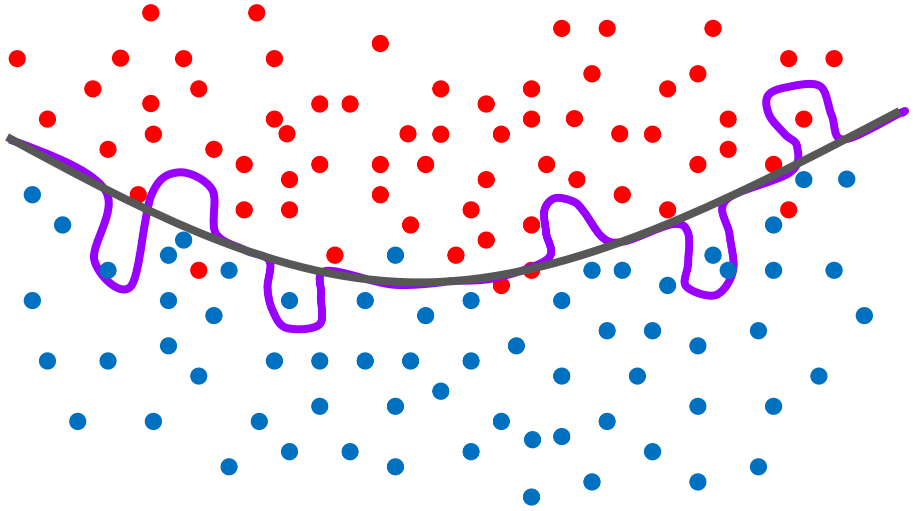


[H]

Which of the two boundaries separates the classes better?

Occams razor suggests the simple one.

Overfitting

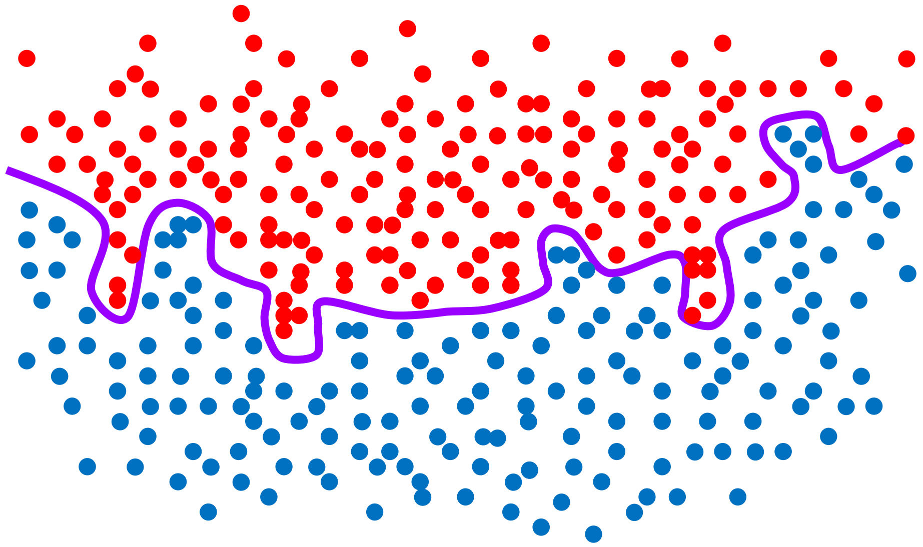


[H]

Another set of examples might exhibit the same minor deviations from the simple boundary, but in different places. So the complex boundary overfits the first example set.

But it is provided we have enough samples. Otherwise …

Overfitting



[H]

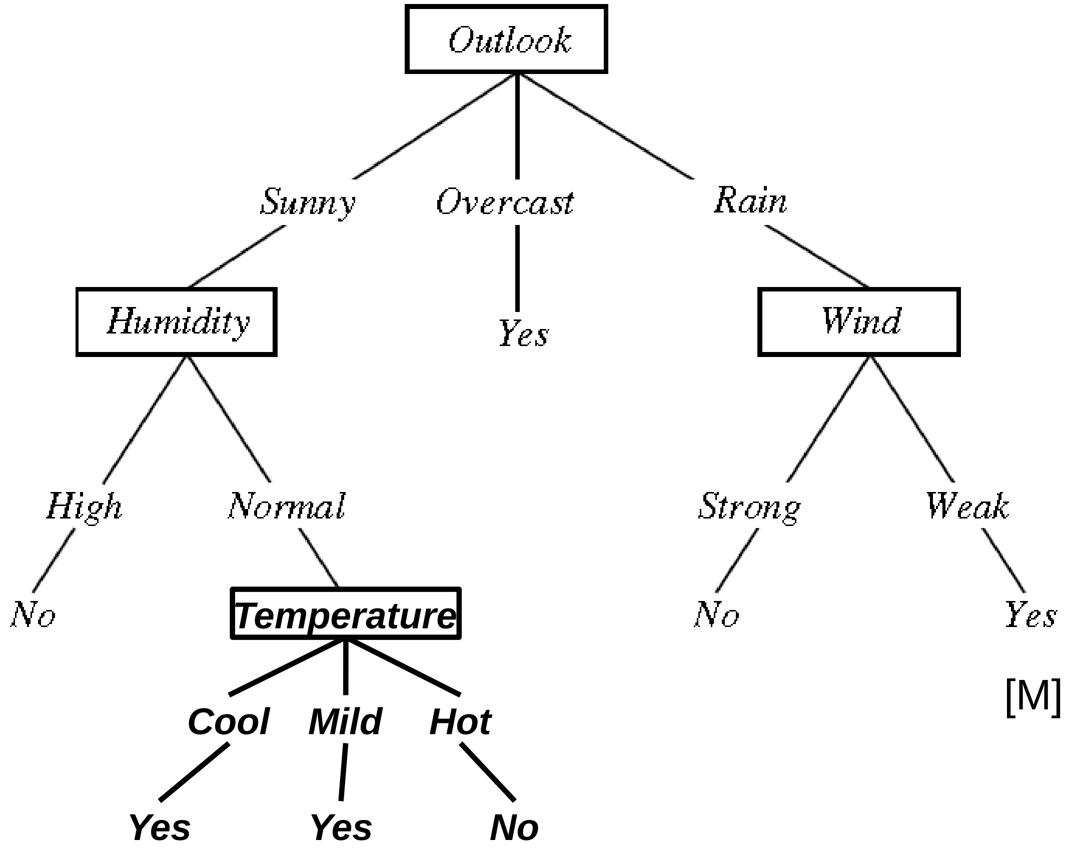
… further examples might prove the “noise” of the first example set to be an effect!

Overfitting

Add noisy training example D15: *< Sunny, Hot, Normal, Strong >* **–**

Earlier tree *h* becomes *h'* :

Other than *h*, *h'* fits training data perfectly. But probably the original *h* fits subsequent data better!



[M]

Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution D of data: $error_D(h)$

Hypothesis $h \in H$ overfits the training data if there is an alternative hypothesis $h' \in H$ such that

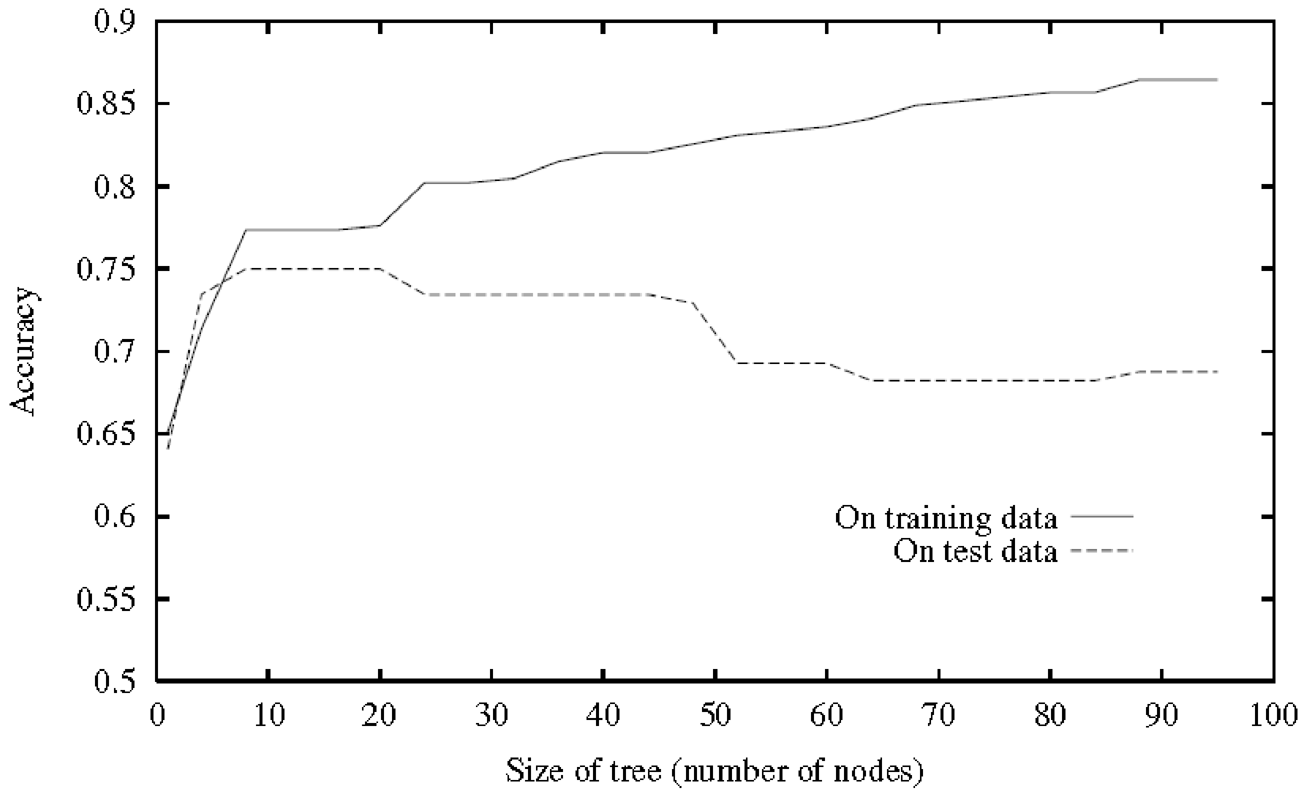
$$error_{train}(h) < error_{train}(h')$$

and

$$error_D(h) > error_D(h').$$

Problem: D is usually not accessible.

Overfitting



[M]

Overfitting: For the training data, adding nodes always results in better or equal performance. For independent test data, adding nodes increases performance only up to a certain point, then the tree specializes too much on the training data and is incapable of generalizing to the test data.

Avoiding overfitting

How can we avoid overfitting?

- Stop growing when data split is not statistically significant
- Grow full tree, then post-prune

How to select best tree:

- Measure performance over training data
- Measure performance over separate validation data set

- MDL: Minimize

  size(*tree*) + *size*(*misclassifications* (*tree*))

Reduced error pruning

Reduced error pruning (Quinlan 1987) removes nodes to achieve better generalization on validation set.

Pruning a decision node *n* means:

- Remove subtree of *n*.
- Make *n* a leaf node and assign most common classification of its affiliated training examples.

Algorithm:

Do until further pruning decreases performance on validation set below original tree:

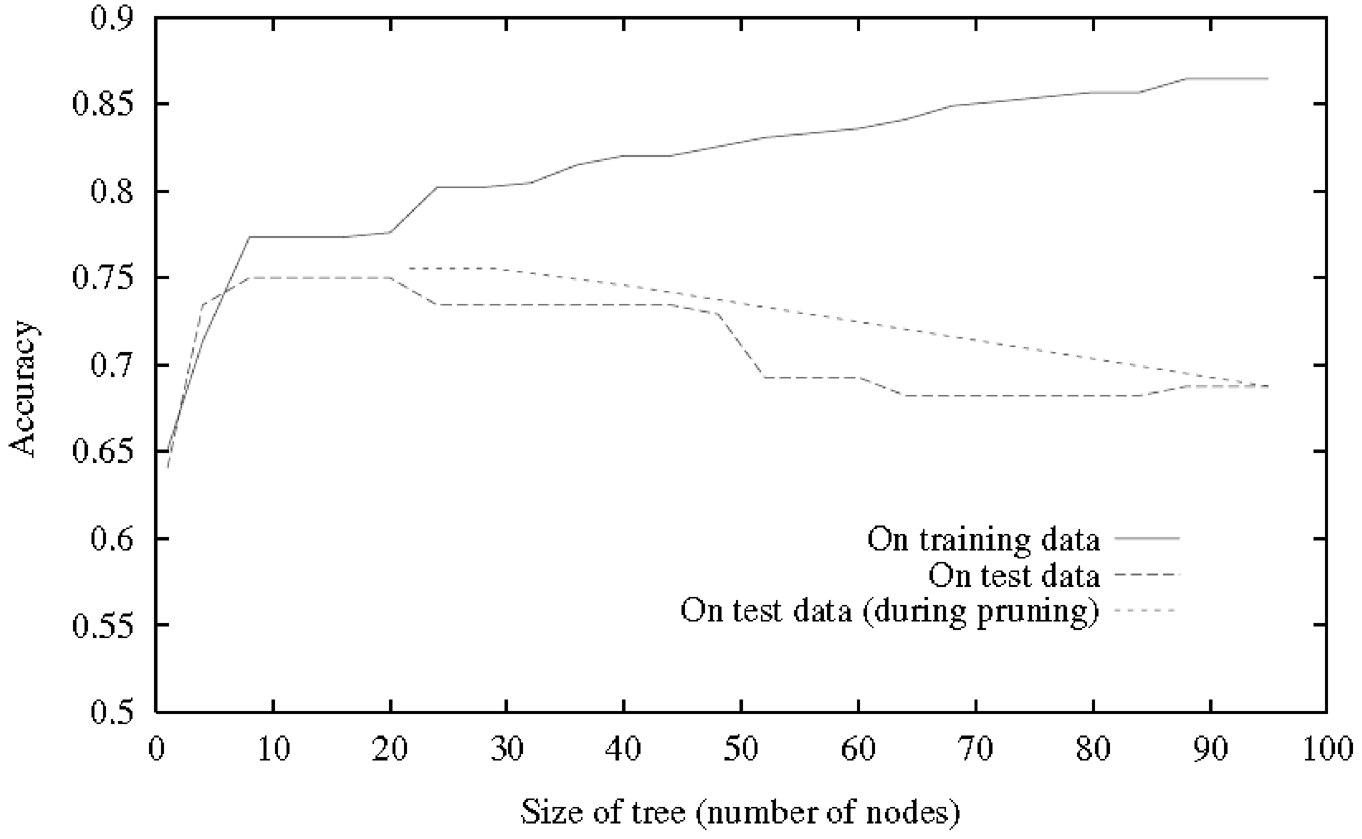
1. Evaluate impact on validation set of pruning each possible node.
2. Greedily remove the one that most improves validation set accuracy.

Reduced error pruning

Properties:

- Produces smallest version of most accurate subtree.
- Removes nodes produced by “noise” (because noise in the training data should by definition not be present in the validation data, otherwise it’s a regularity).
- If data is limited, use rule post pruning instead.

Effect of reduced error pruning



[M]

Reduced error pruning reduces the effect of overfitting.

Here, in addition to the validation set used for pruning, a third, independent test data set has been used to obtain the above curve.

Rule post pruning

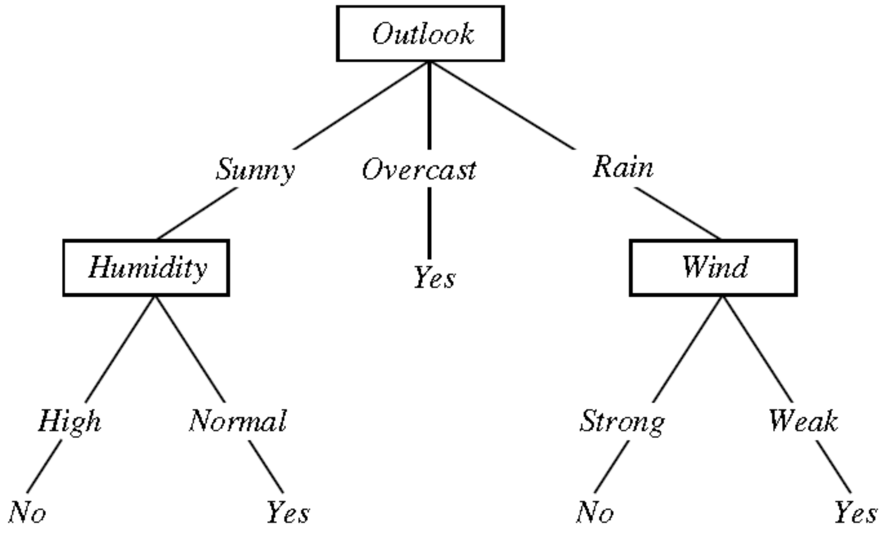
Rule post pruning is similar to the pruning method employed by C4.5 (Quinlan 1993).

Algorithm:

1. Build up decision tree to fit training set as well as possible, allowing overfitting.
2. Convert tree into equivalent set of rules: One rule for each path from root to leaf.
3. Prune each rule (independently of others) by removing any preconditions that result in improving its accuracy on the validation set.
4. Sort final rules by their accuracy on validation set. Apply them in this order when classifying new instances.

Rules from decision trees

Example:



[M]

IF (*Outlook = Sunny*) ∧ (*Humidity = High*) // *precondition* or *antecedent*

THEN *PlayTennis = No* // *postcondition* or *consequent*

For pruning, the preconditions (*Outlook = Sunny*) or (*Humidity = High*) would be removed if performance on the validation set increases.

A useful trick allows to substitute the validation set by the training set.

Rule post pruning

Why is pruning based on rules better than pruning nodes?

- Only certain paths from root to leaves are pruned, not entire subtrees (as with reduced error pruning). This allows pruning sensitive to the “context” in which a node is used.
- There is no hierarchy of rules (most important rule at the root) any more. Even the root node may be pruned but subtrees are retained.
- Better readability. Note decision trees are not only used for classification but also to understand the data.

Continuous valued attributes

To include a continuous valued attribute, thresholds must be defined to obtain a binary attribute.

Thresholds should be chosen to gain information:

1. Sort attribute values over example set.
2. Find "boundaries" within the values where classification changes.
3. Thresholds should be set at boundaries.

Example:

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

[M]

So $Temperature_{binary} = \theta(Temperature - 54) \cdot \theta(85 - Temperature)$

Attributes with many values

Problem:

- If an attribute has many values, *Gain* tends to select it.
- Attribute *Date = Jun_3_2011* would become root node for the *PlayTennis* example, classifying perfectly without yielding any real information.

One solution:

Use *GainRatio* (Quinlan 1986) instead of *Gain*:

$$GainRatio(S, A) = Gain(S, A) \,/\, SplitInformation(S, A)$$

$$SplitInformation(S, A) = -\sum_{i=1\ldots n} |S_i| / |S| \cdot \log_2 (|S_i| / |S|)$$

where S_i is the subset of S for which A has value v_i and A has n different values in total. *SplitInformation* is the entropy with respect to the attribute values.

Attributes with many values

Properties of *GainRatio* :

- If two attributes yield identical *Gain* and for both the distribution of attribute values over *S* is uniform, then *GainRatio* favors the attribute with fewer values.
- *GainRatio* is undefined for attributes with the same value for all examples (zero denominator), but such attributes are no use anyway.
- *GainRatio* is very large if $|S_i| \approx |S|$.

Question: What's the *SplitInformation(S, A)* for the following distributions of attribute values?

½ ½	¼ ¼ ¼ ¼
1	**2**

Attributes with costs

Consider a task where decision requires costly acquisition of attribute values. So design of the tree should take into account not only the information gain but also the acquisition cost of attributes.

Examples:

- Medical diagnosis: "Decision" = diagnosis, attributes are, *BloodTest* (150 Eur), *Pulse* (1,20 Eur) etc.;
- Robotics: "Decision" = object movable?, attributes are *EvalCameraImage* (1 sec), *TouchWithManipulator* (40 sec).

➔ Cost of attributes must be taken into account relative to information gain !

Attributes with costs

How to learn a consistent tree with *low expected cost*?

One approach: replace gain by

- $(2^{Gain(S,A)} - 1) / (Cost(A) + 1)^w$,

  where $w \in [0,1]$ determines the importance of cost.

  Proposed by Nunez (1988) for medical diagnosis rules.

- $Gain^2(S, A) / Cost(A)$

  Proposed by Tan & Schlimmer (1990) for object classification using robot manipulator.

Approaches are highly task dependent !

Missing attribute values

What if some examples are missing values of *A* ?

Use training example anyway, sort through tree using one of the following methods:

- If node *n* tests *A*, assign most common value of *A* among other examples sorted to node *n*.
- Assign most common value of *A* among other examples with same target value.
- Assign probability p_i to each possible value v_i of *A*, where p_i is estimated from the value distribution of the examples assigned to node *n*. Assign fraction p_i of the examples with missing values to each descendant in tree.

Classify new examples in same fashion.

Application example:

Tactile sensing in robotics

Motivation

Trees are not only used for classification / decision making, but also to understand

- how classifications / decisions come about from
- the structure of the data,
- and its representation by the attributes.

Tactile sensing in robotics

Major challenge in robotics: Tactile sensing

- So far, sensors in manipulator robotics are mainly restricted to
  - optical sensors (cameras, laser scanners),
  - simple mechanical sensors such as force/torque sensors (e.g., wrist sensors) and simple pressure sensors.
- By contrast, tactile sensing is highly important in human grasping
- Artificial skin providing tactile sensing with sufficient
  - resolution
  - speed
  - range of pressure values
  - flexibility

is not yet available.

Human skin

Receptors in the skin:

- Temperature
- Pain
- Pressure
- Vibration

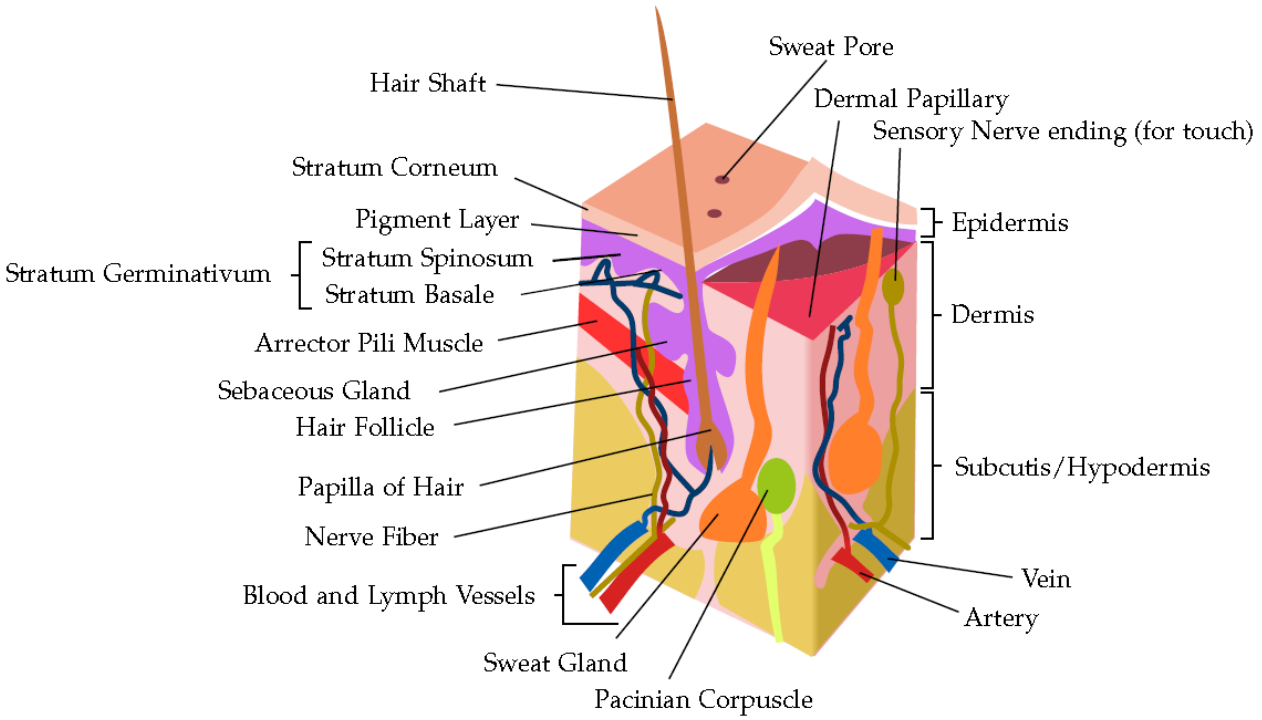


Figure 2: Schematic drawing of the human skin. Source: US Government

Receptor Type	Afferent Fiber	Receptive Field [mm^2]	Frequency Range	Receptors [$1/cm^2$]	Sensitive to
Meissner's Corpusle	FA I (RA)	1 – 100	10 - 200	140	Dynamic light touch, Motion & Vibration
Pacinian Corpuscle	FA II (PC)	10 – 1000	40 – 800	21	Dynamic touch, Vibration
Merkel Disk	SA I	2 – 100	0.4 – 100	70	Static light touch, Texture
Ruffini Endings	SA II	10 – 500	7	49	Static touch, Skin stretch

[Sch]

Tactile sensing in robotics

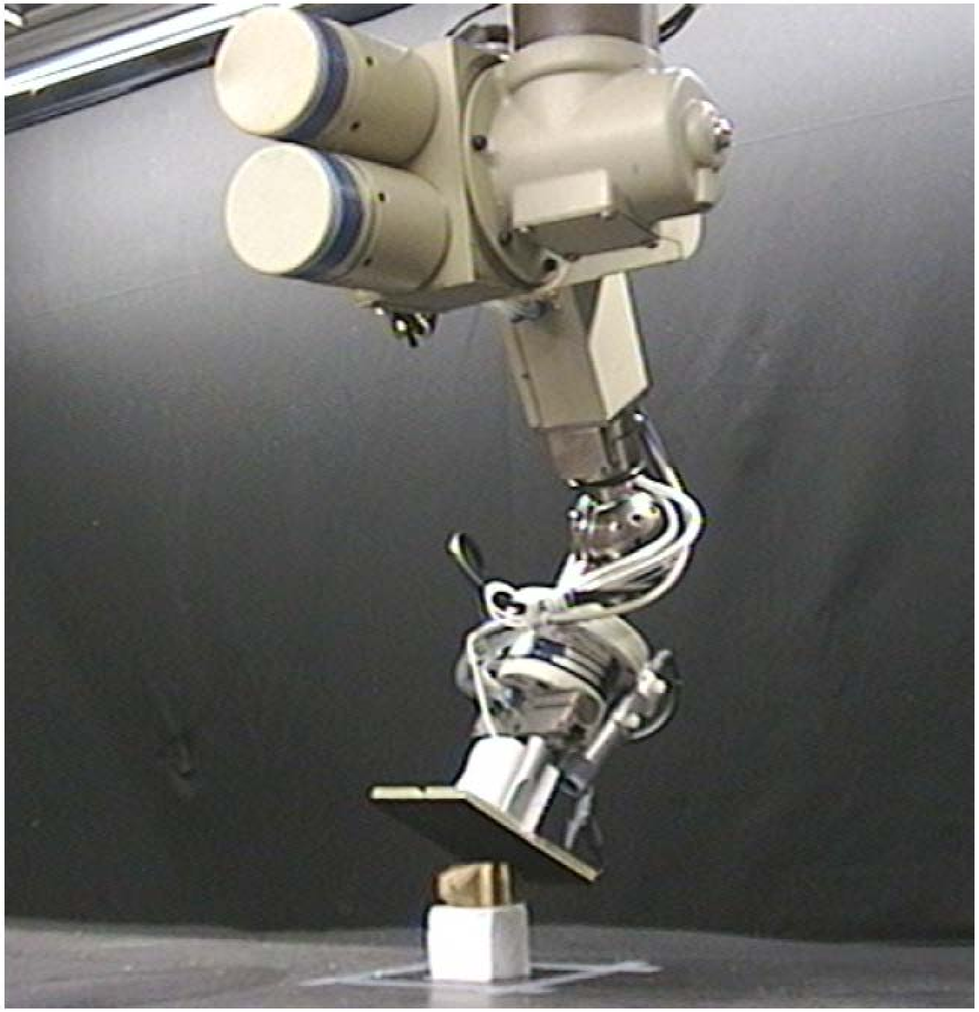
Recent work by Matthias Schöpfer, Bielefeld University:

Object recognition using a tactile sensor array (matrix sensor).

Experimental setup:

- Puma 260 robot arm with 6 DOF.
- 16x16 pressure sensor as end-effector.
- Sensor is moved according to perceived pressure on object and according to predefined scanning scheme.
- Real time evaluation of pressure patterns and real time control of manipulator is essential.

Robot setup



[Sch]

Pressure sensor

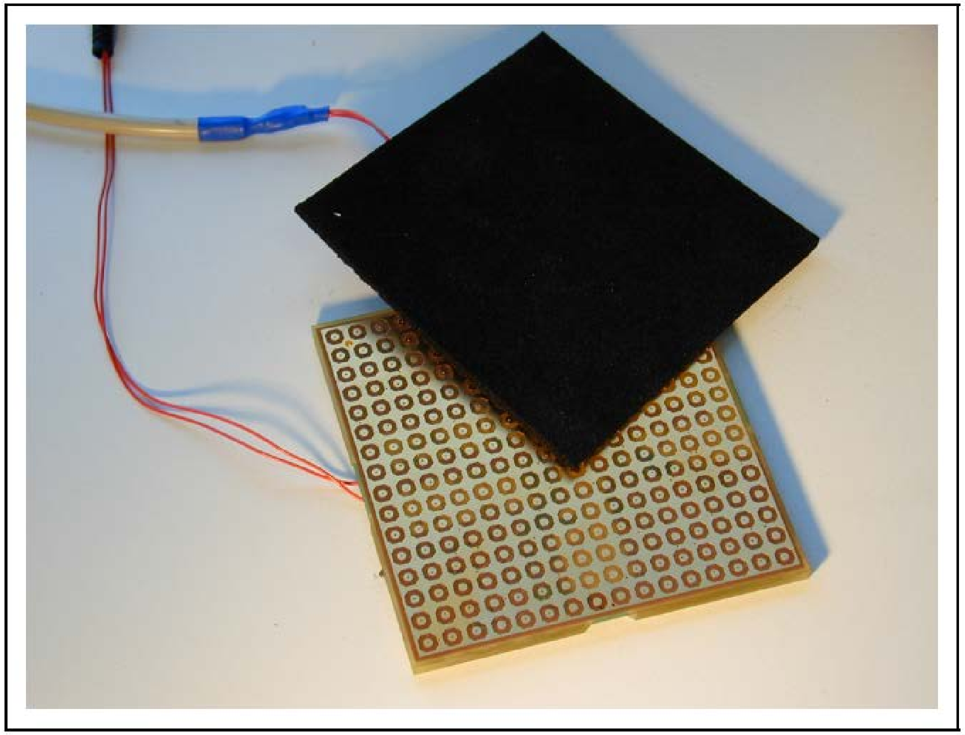


Figure 18: DSA100-256IS Tactile Sensor [Sch]

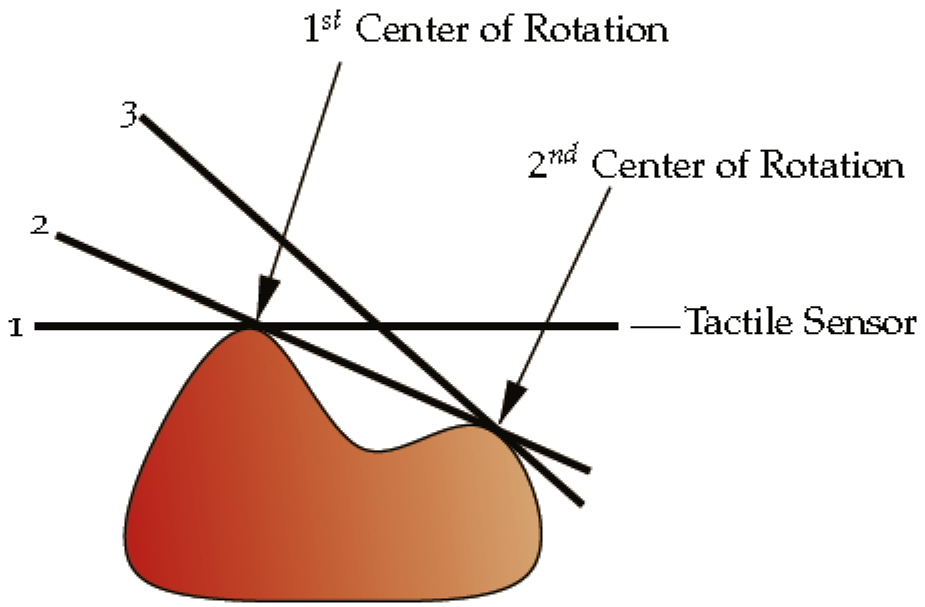
UNIVERSITÄT OSNABRÜCK

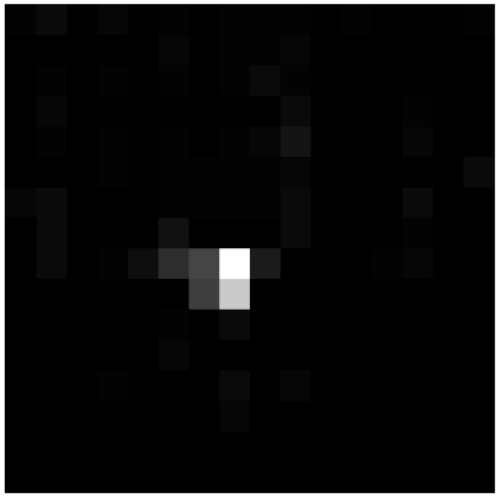
Objects

Objects were probed in different poses.

[Sch]

Probing strategy





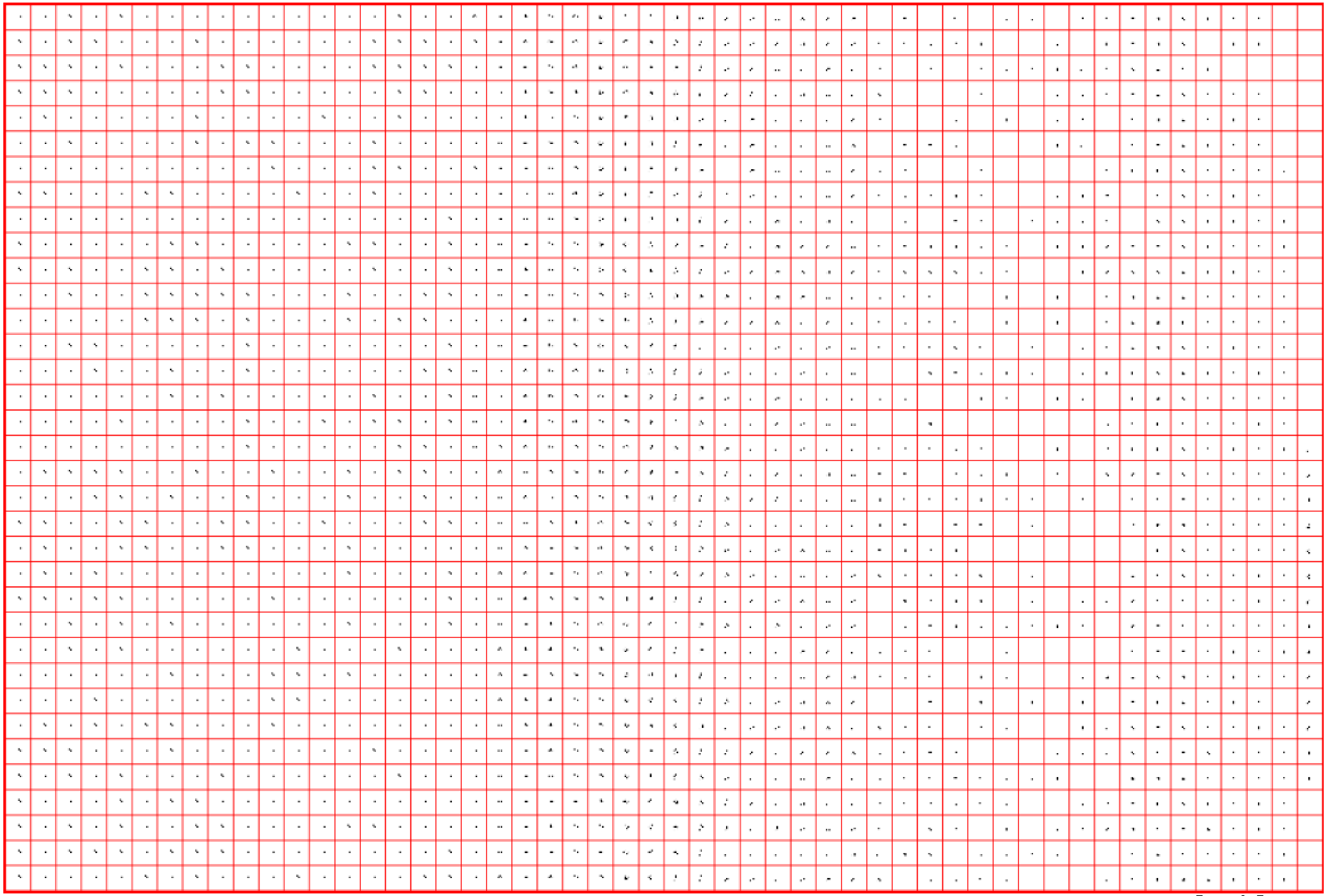
[Sch]

Left: 2D illustration of the probing strategy. The sensor is rolled over the object with the contact point as the center of rotation. A new contact point becomes the next center.

Right: 16x16 pressure profile encoded as gray values.

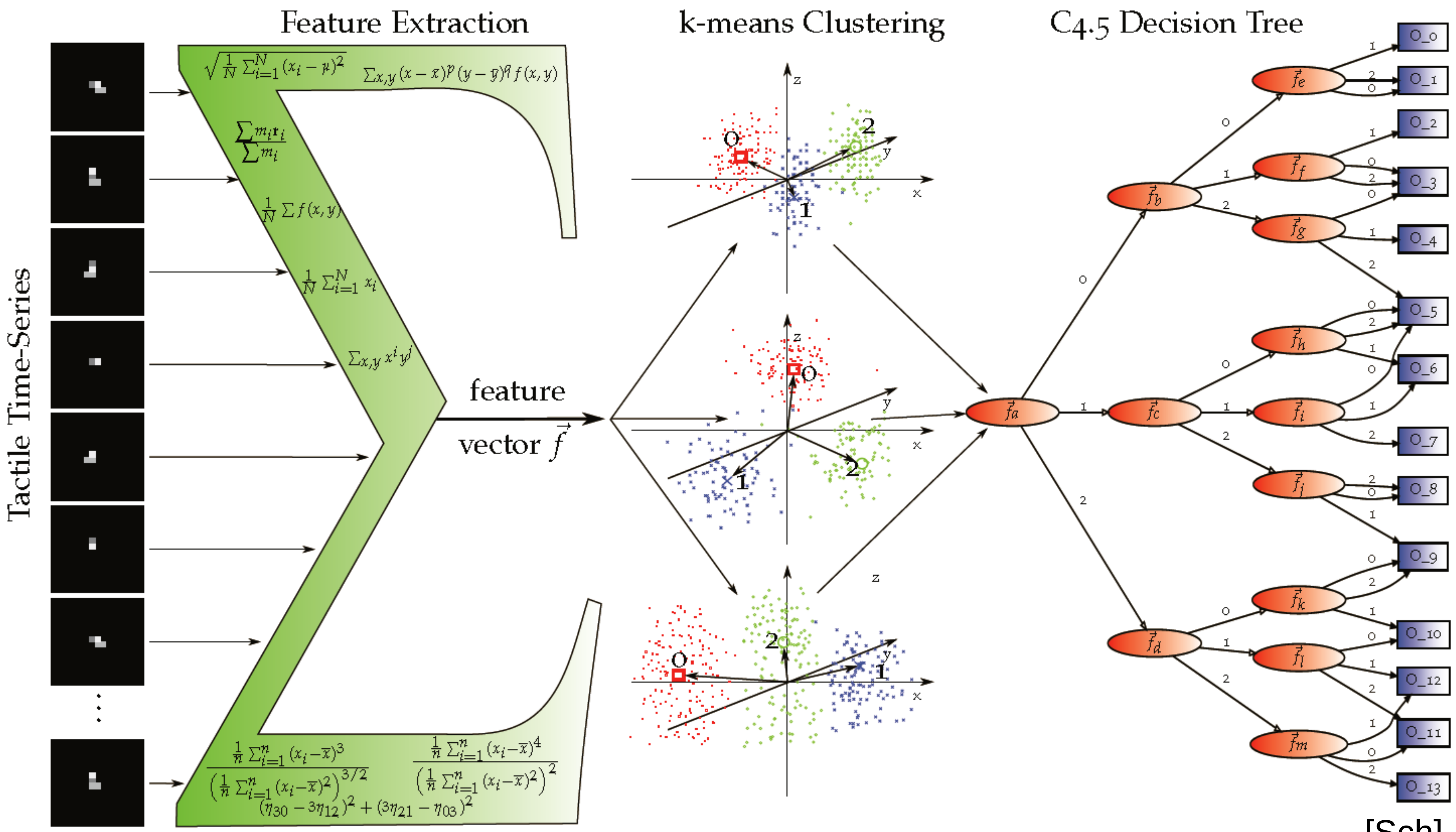
UNIVERSITÄT OSNABRÜCK

Pressure profile time series



[Sch]

Pattern recognition


Feature Extraction
k-means Clustering
C4.5 Decision Tree
Tactile Time-Series
feature vector $\vec{f}$
$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$
$\sum_{x,y}(x-\bar{x})^p(y-\bar{y})^q f(x,y)$
$\frac{\sum m_i r_i}{\sum m_i}$
$\frac{1}{N}\sum f(x,y)$
$\frac{1}{N}\sum_{i=1}^{N} x_i$
$\sum_{x,y} x^i y^j$
$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2\right)^{3/2}}$
$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2\right)^2}$
$(\eta_{30}-3\eta_{12})^2+(3\eta_{21}-\eta_{03})^2$
O_0
O_1
O_2
O_3
O_4
O_5
O_6
O_7
O_8
O_9
O_10
O_12
O_11
O_13

[Sch]

From the input (pressure profile time series), a variety of features is extracted. Feature vectors of the example set are clustered to obtain prototypes.

Feature extraction

The tactile recognition system breaks the raw input down to features for dimension reduction and to gain robustness (like most other pattern recognition systems):

- 16x16 sensor points x 2200 frames ≈ 600000 real values
- Feature extraction → 920 real valued features
- Clustering for discretization → 226 discrete valued attributes as input to C4.5

Design problem (again for most pattern recognition systems):

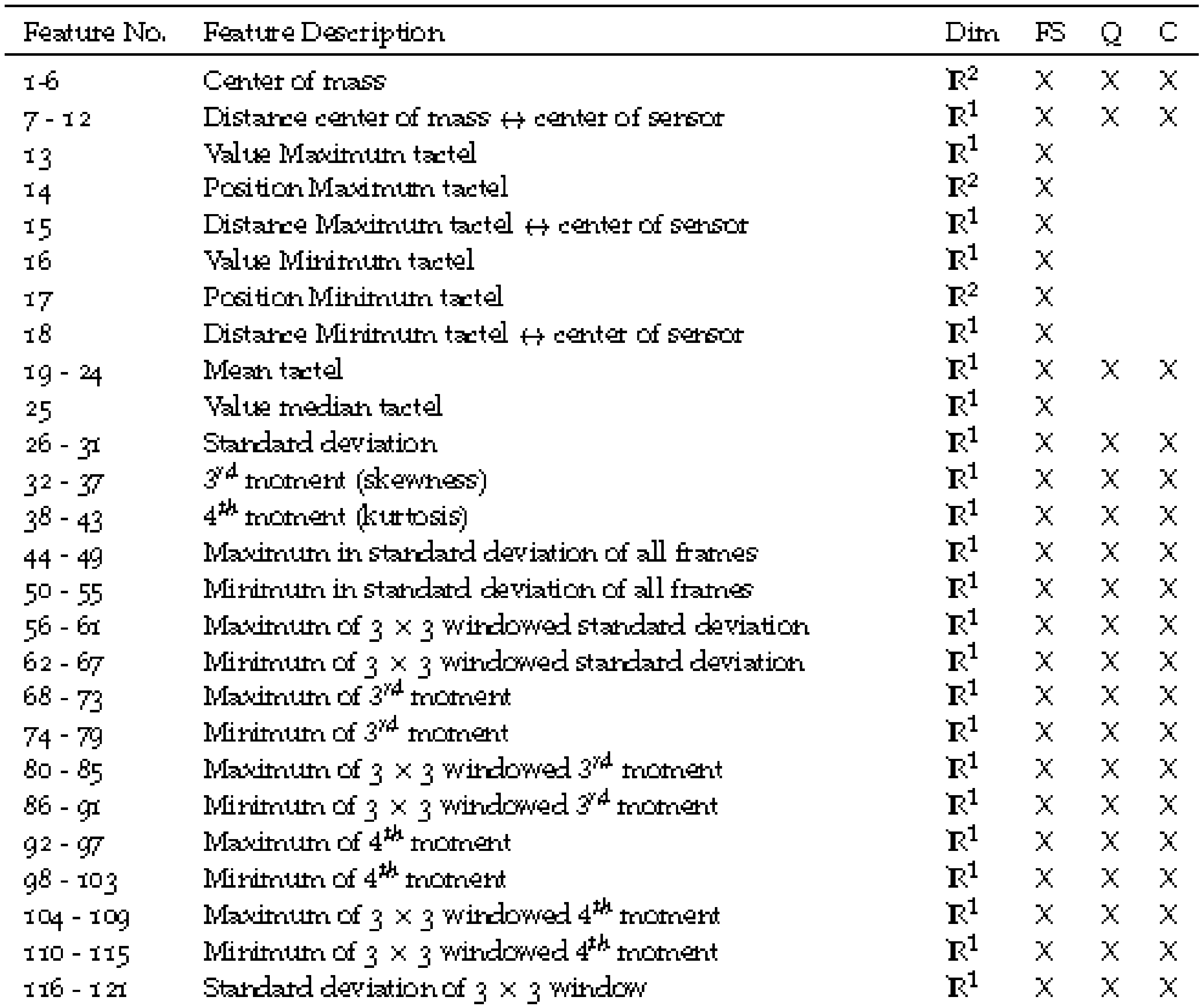
Which features?

→There are many commonly used features, but performance for a particular method is unknown !

Solution: Use C4.5 for building classification tree.

Analyze tree to find relevant features.

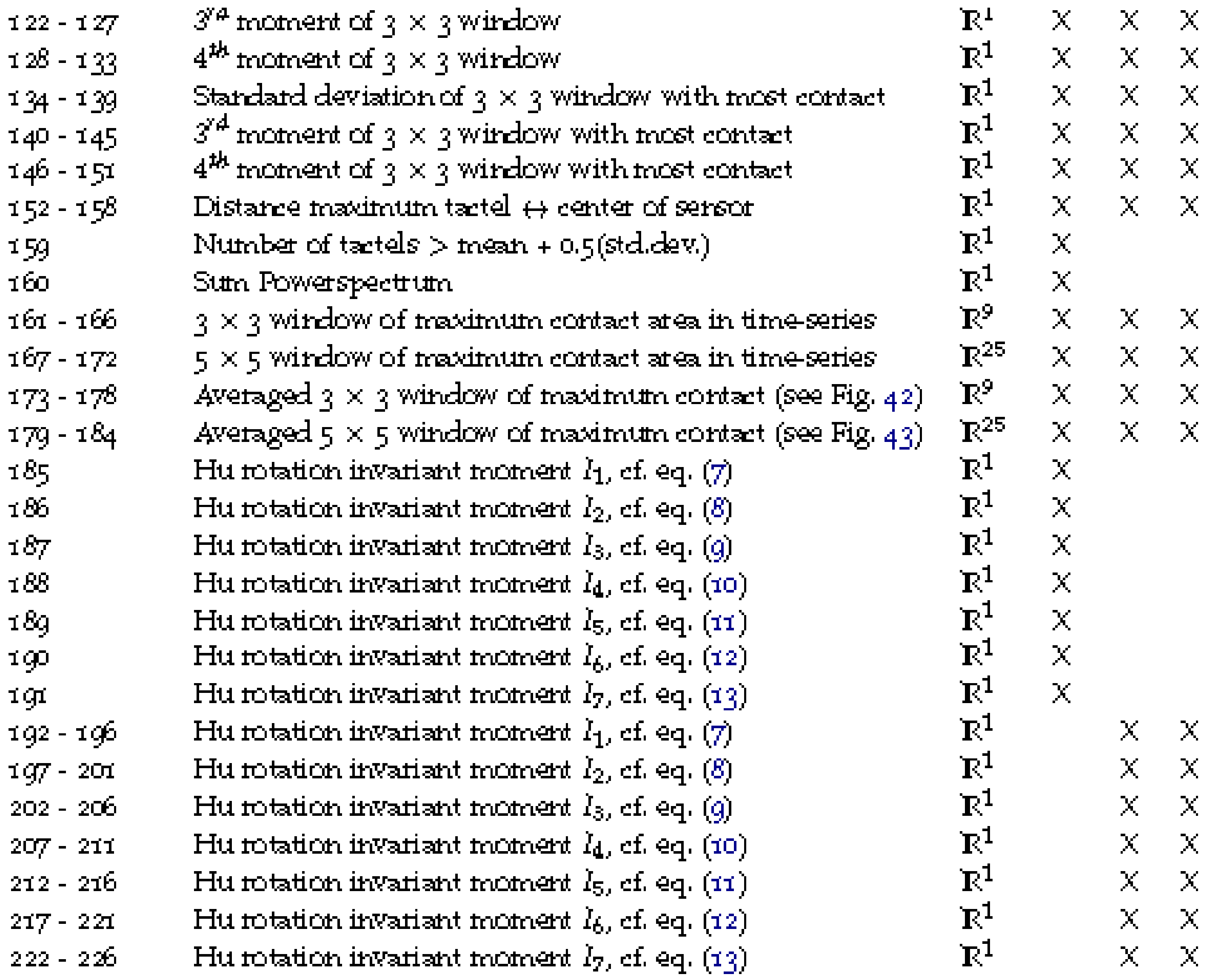
Features



Feature No.	Feature Description	Dim	FS	Q	C
1-6	Center of mass	$\mathbb{R}^2$	X	X	X
7 - 12	Distance center of mass $\leftrightarrow$ center of sensor	$\mathbb{R}^1$	X	X	X
13	Value Maximum tactel	$\mathbb{R}^1$	X		
14	Position Maximum tactel	$\mathbb{R}^2$	X		
15	Distance Maximum tactel $\leftrightarrow$ center of sensor	$\mathbb{R}^1$	X		
16	Value Minimum tactel	$\mathbb{R}^1$	X		
17	Position Minimum tactel	$\mathbb{R}^2$	X		
18	Distance Minimum tactel $\leftrightarrow$ center of sensor	$\mathbb{R}^1$	X		
19 - 24	Mean tactel	$\mathbb{R}^1$	X	X	X
25	Value median tactel	$\mathbb{R}^1$	X		
26 - 31	Standard deviation	$\mathbb{R}^1$	X	X	X
32 - 37	3rd moment (skewness)	$\mathbb{R}^1$	X	X	X
38 - 43	4th moment (kurtosis)	$\mathbb{R}^1$	X	X	X
44 - 49	Maximum in standard deviation of all frames	$\mathbb{R}^1$	X	X	X
50 - 55	Minimum in standard deviation of all frames	$\mathbb{R}^1$	X	X	X
56 - 61	Maximum of 3×3 windowed standard deviation	$\mathbb{R}^1$	X	X	X
62 - 67	Minimum of 3×3 windowed standard deviation	$\mathbb{R}^1$	X	X	X
68 - 73	Maximum of 3rd moment	$\mathbb{R}^1$	X	X	X
74 - 79	Minimum of 3rd moment	$\mathbb{R}^1$	X	X	X
80 - 85	Maximum of 3×3 windowed 3rd moment	$\mathbb{R}^1$	X	X	X
86 - 91	Minimum of 3×3 windowed 3rd moment	$\mathbb{R}^1$	X	X	X
92 - 97	Maximum of 4th moment	$\mathbb{R}^1$	X	X	X
98 - 103	Minimum of 4th moment	$\mathbb{R}^1$	X	X	X
104 - 109	Maximum of 3×3 windowed 4th moment	$\mathbb{R}^1$	X	X	X
110 - 115	Minimum of 3×3 windowed 4th moment	$\mathbb{R}^1$	X	X	X
116 - 121	Standard deviation of 3×3 window	$\mathbb{R}^1$	X	X	X

[Sch]

Features



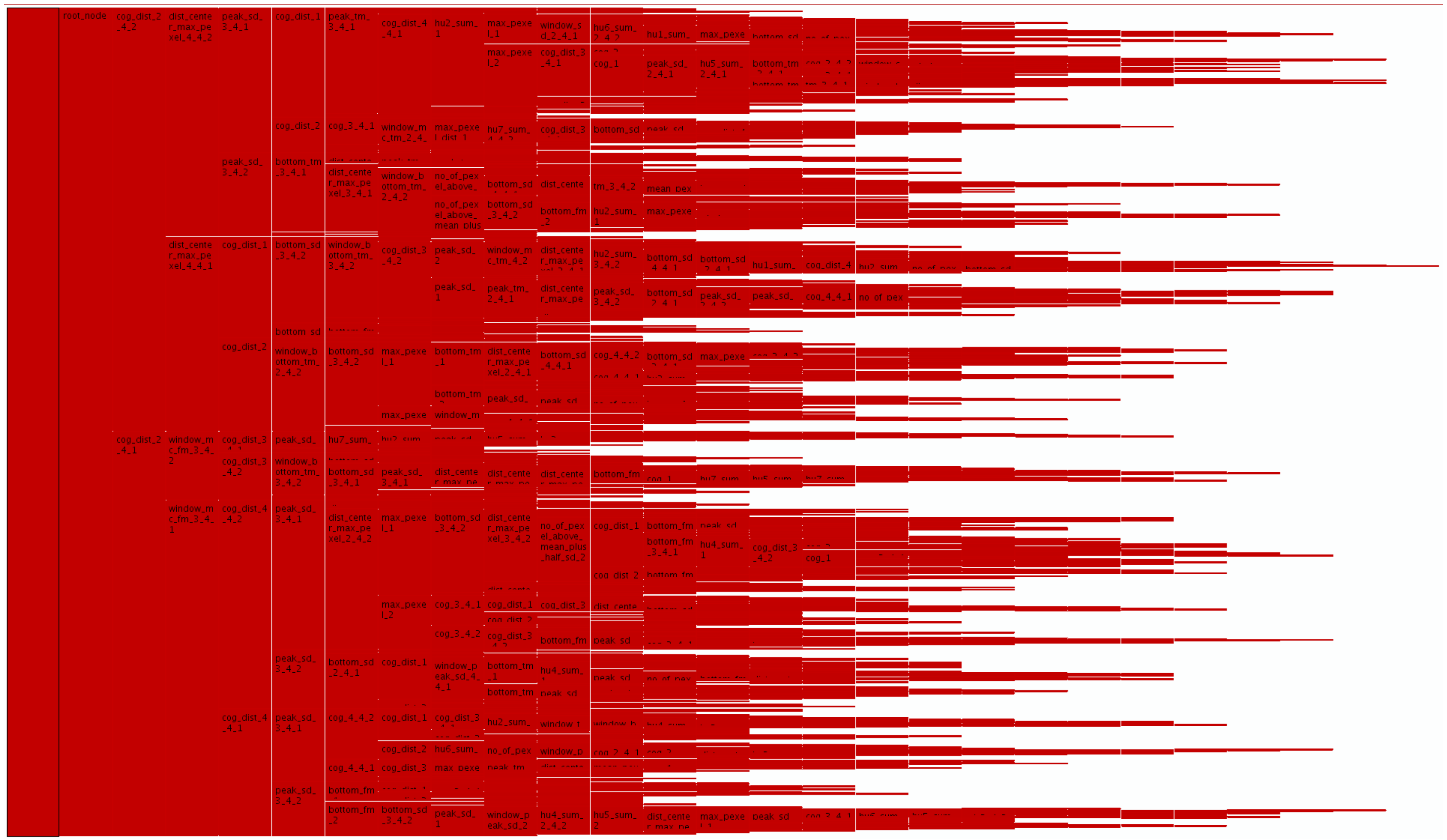
122 - 127	3rd moment of 3×3 window	$\mathbb{R}^1$	X	X	X
128 - 133	4th moment of 3×3 window	$\mathbb{R}^1$	X	X	X
134 - 139	Standard deviation of 3×3 window with most contact	$\mathbb{R}^1$	X	X	X
140 - 145	3rd moment of 3×3 window with most contact	$\mathbb{R}^1$	X	X	X
146 - 151	4th moment of 3×3 window with most contact	$\mathbb{R}^1$	X	X	X
152 - 158	Distance maximum tactel $\leftrightarrow$ center of sensor	$\mathbb{R}^1$	X	X	X
159	Number of tactels > mean + 0.5(std.dev.)	$\mathbb{R}^1$	X		
160	Sum Powerspectrum	$\mathbb{R}^1$	X		
161 - 166	3×3 window of maximum contact area in time-series	$\mathbb{R}^9$	X	X	X
167 - 172	5×5 window of maximum contact area in time-series	$\mathbb{R}^{25}$	X	X	X
173 - 178	Averaged 3×3 window of maximum contact (see Fig. 42)	$\mathbb{R}^9$	X	X	X
179 - 184	Averaged 5×5 window of maximum contact (see Fig. 43)	$\mathbb{R}^{25}$	X	X	X
185	Hu rotation invariant moment I_1, cf. eq. (7)	$\mathbb{R}^1$	X		
186	Hu rotation invariant moment I_2, cf. eq. (8)	$\mathbb{R}^1$	X		
187	Hu rotation invariant moment I_3, cf. eq. (9)	$\mathbb{R}^1$	X		
188	Hu rotation invariant moment I_4, cf. eq. (10)	$\mathbb{R}^1$	X		
189	Hu rotation invariant moment I_5, cf. eq. (11)	$\mathbb{R}^1$	X		
190	Hu rotation invariant moment I_6, cf. eq. (12)	$\mathbb{R}^1$	X		
191	Hu rotation invariant moment I_7, cf. eq. (13)	$\mathbb{R}^1$	X		
192 - 196	Hu rotation invariant moment I_1, cf. eq. (7)	$\mathbb{R}^1$		X	X
197 - 201	Hu rotation invariant moment I_2, cf. eq. (8)	$\mathbb{R}^1$		X	X
202 - 206	Hu rotation invariant moment I_3, cf. eq. (9)	$\mathbb{R}^1$		X	X
207 - 211	Hu rotation invariant moment I_4, cf. eq. (10)	$\mathbb{R}^1$		X	X
212 - 216	Hu rotation invariant moment I_5, cf. eq. (11)	$\mathbb{R}^1$		X	X
217 - 221	Hu rotation invariant moment I_6, cf. eq. (12)	$\mathbb{R}^1$		X	X
222 - 226	Hu rotation invariant moment I_7, cf. eq. (13)	$\mathbb{R}^1$		X	X

[Sch]

Analyzing feature contributions

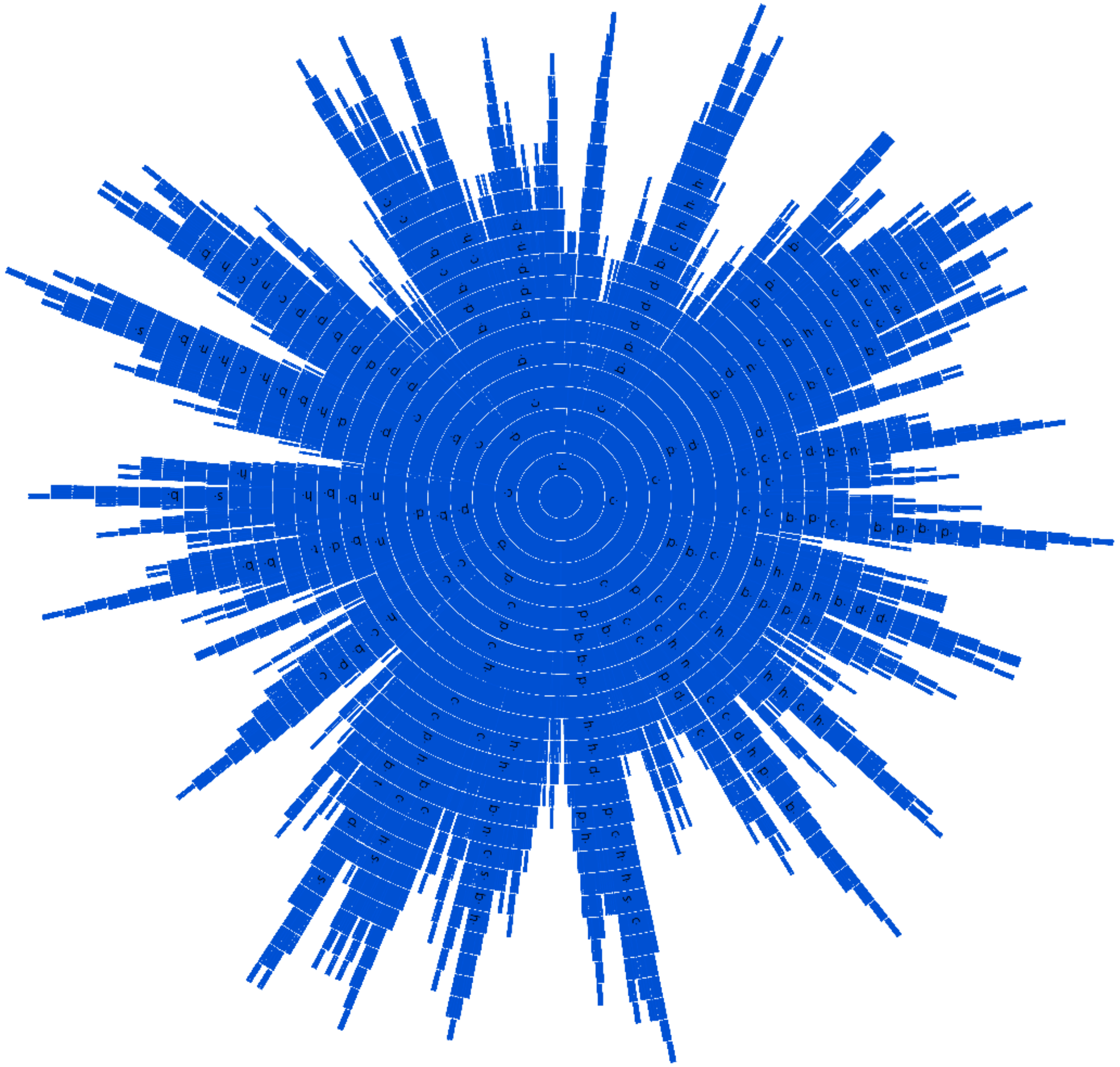
Problem: How to analyze the tree?

Rectangle tree visualization



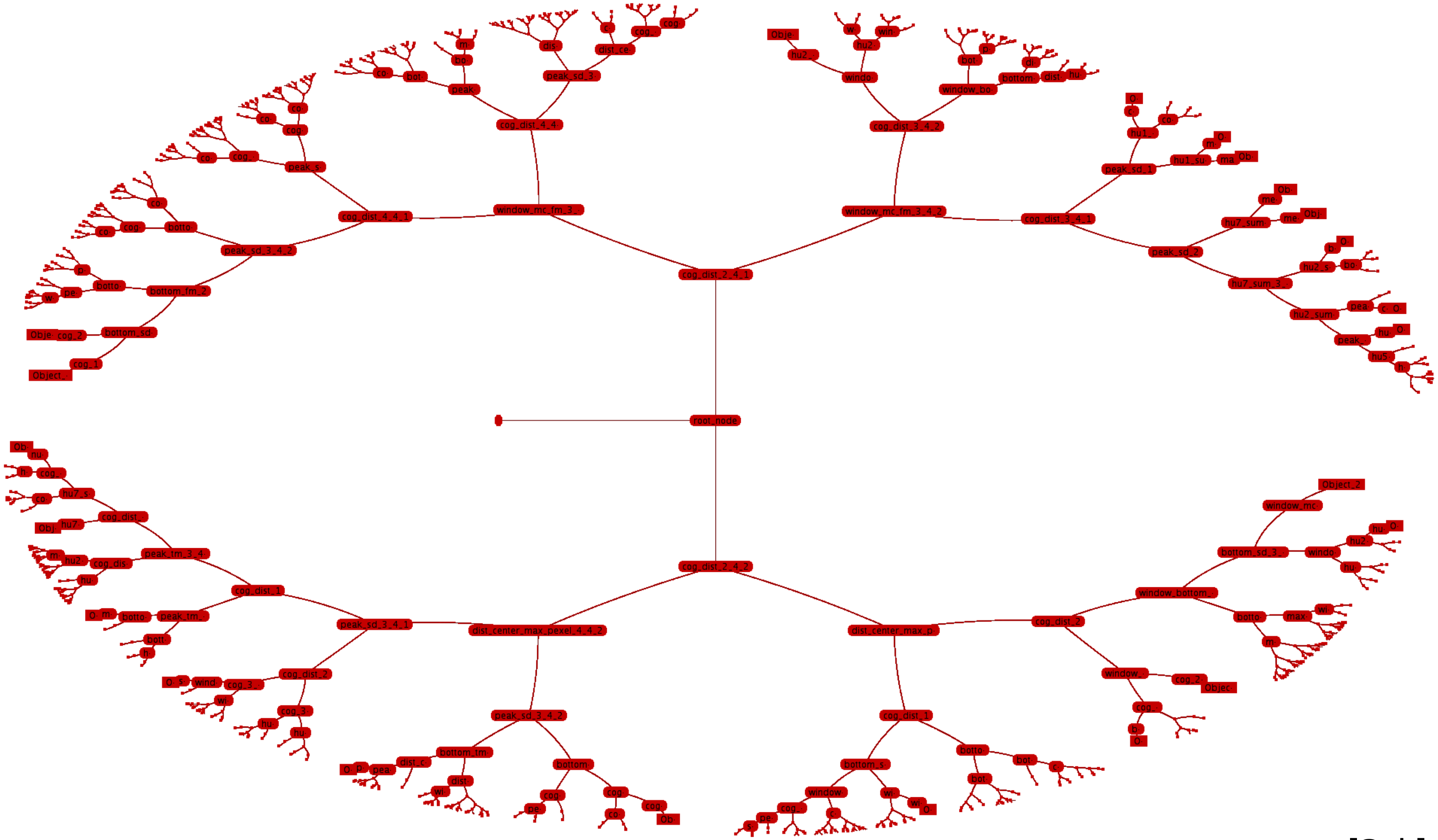
[Sch]

Sunburst tree visualization



[Sch]

Hyperbolic tree visualization



[Sch]

Analyzing feature contributions

Problem: How to analyze the tree?

- Tree is huge, visualization requires special tools
- Exact shape depends on random elements in C4.5 (random subset selection in *iterative mode*)

Solution: Use combination of measures to judge relevance of a feature *f*

- Level at which *f* appears in tree.
- # examples partly classified by *f*.
- Class entropy
- Mutual information of feature and classification

Results:

- Several features proved to be useless
- Overall, contribution is broadly distributed among features
- List of best features
- Influence of probing phase was analysed.

Summary

- Decision trees are well established classifiers for a wide range of tasks.
- Classification in intuitive steps.
- ID3 builds tree based on information gain achieved by attributes.
- Many extensions of the core algorithm.
- Related algorithms for clustering.
- For real tasks, trees may still be difficult to analyze.
- Visualization techniques and additional measures required for large trees, i.e., data of high complexity.

Image sources

[M] Online material available at www.cs.cmu.edu/~tom/mlbook.html for the textbook: Tom M. Mitchell: *Machine Learning*, McGraw-Hill

[H] Gunther Heidemann, 2012.

[Sch] Matthias Schöpfer, *Tactile Perception of Cognitive Robots*, Dissertation, Universität Bielefeld, Technische Fakultät, 2011.