

Künstliche Intelligenz (Sommersemester 2024)

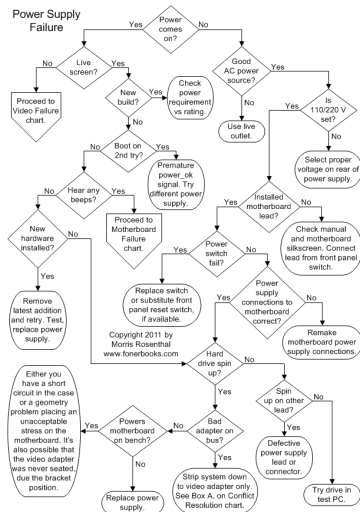
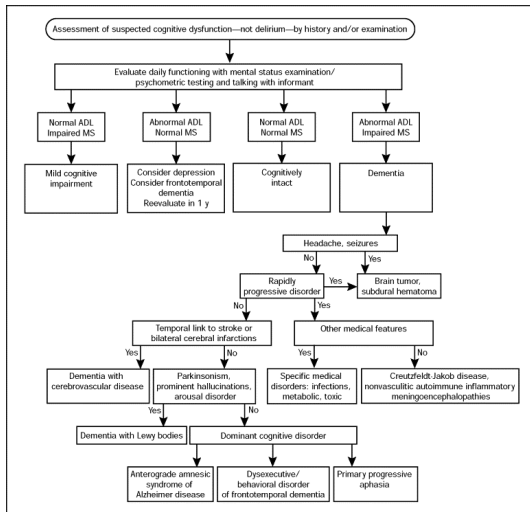
Kapitel 04: Entscheidungsbäume

Prof. Dr. Adrian Ulges

Entscheidungsbäume: Popularität Bild: [5]



Entscheidungsbäume wurden als der **meistverwendete ML-Modelltyp** bezeichnet [6].





1. Grundlagen
2. Exkurs: Informationstheorie und Information Gain
3. Training und Pruning
4. Random Forests

Entscheidungsbäume: Einführung

Ansatz

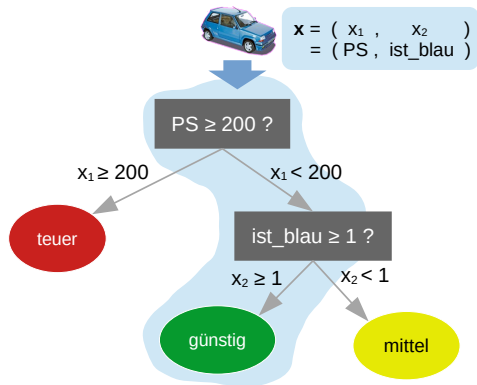
- ▶ Treffe Vorhersage $f_{\theta}(\mathbf{x})$ basierend auf einfachen **rekursiven Entscheidungen** (oder *Regeln*).

Vorteile

- ▶ **Einfachheit** und **Geschwindigkeit**.
- ▶ **Transparenz** der Entscheidungen des Modells.

Hauptfrage






- ▶ **Lernen**: Wie bauen wir einen “guten” Baum basierend auf einem (gelabelten) Trainingsdatensatz?

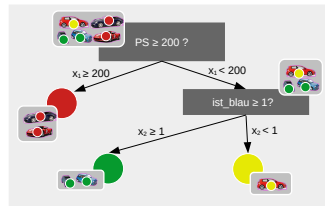
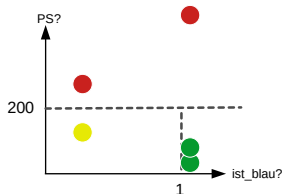


Beispiel: Autopreis-Klassifikator

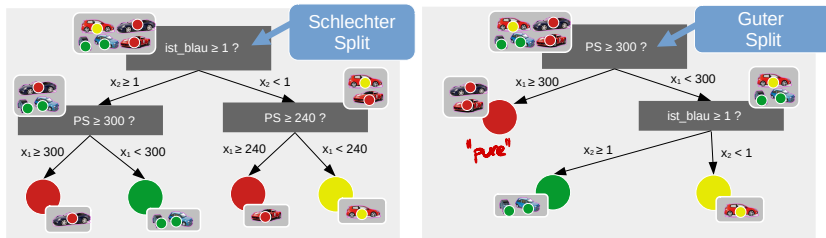
Rekursives Training

- ▶ gegeben: ein Trainingsdatensatz $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots (\mathbf{x}_n, y_n)\}$.
- ▶ Wir beginnen mit der **Wurzel** des Baums, und dem kompletten Datensatz \mathcal{D} .
- ▶ In jedem Knoten **teilen** wir den aktuellen Datensatz nach einer Dimension i des Merkmalsvektors und einer Schwelle t auf.
- ▶ Wir erhalten zwei Kindknoten (mit zwei Teilmengen $\mathcal{D}_1, \mathcal{D}_2 \subseteq \mathcal{D}$) und verfahren rekursiv weiter.

Objekt	(PS) x_1	(ist_blaue) x_2	y
	87	1	günstig
	560	1	teuer
	110	0	mittel
	280	0	teuer
	30	1	günstig



Wie finden wir “gute” Splits?



- ▶ Der Schlüsselteil des Trainings besteht darin, die Knoten “gut” zu splitten, so dass wir **möglichst kleine Bäume** erhalten 😊.
- ▶ Übliche Strategie: Split so wählen, dass die **Labels y in den beiden Kindknoten möglichst homogen** werden!
- ▶ Es existieren viele **Split-Kriterien** (engl. “Gains”) die diese Homogenität messen. Je höher der Gain, desto besser der Split.
- ▶ Hier: Der sogenannte **Information Gain**.



1. Grundlagen
2. Exkurs: Informationstheorie und Information Gain
3. Training und Pruning
4. Random Forests

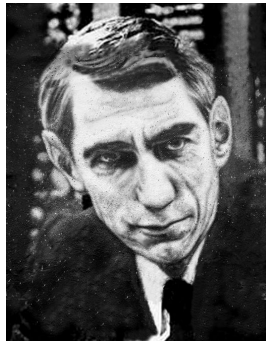
Split-Kriterium für Klassifikation: Informationsgewinn Bild: [3]



Um unser Split-Kriterium *Information Gain* herzuleiten, machen wir einen Ausflug in die Informationstheorie.

Informationstheorie¹: Anwendungsbereiche

- ▶ **Kompression**: Wie stark kann **zip** einen englischsprachigen Text komprimieren?
- ▶ **Kryptographie**: Wie leicht ist es, eine verschlüsselte Nachricht zu entziffern?
- ▶ **Statistische Inferenz**: Wie messen wir die "Unsicherheit" in einem Zufallsprozess?



¹Claude E. Shannon: "A Mathematical Theory of Communication" (1948)

Präfix-Codes²



- ▶ Stellen Sie sich eine **Sprache** mit vier Buchstaben a,b,c,d vor.
Eine **Nachricht** in dieser Sprache könnte lauten: “abaadbaabcbacda”.
- ▶ Stellen Sie sich vor, diese Nachricht in **Bits** zu übertragen. Wir **kodieren** jedes Zeichen separat:

Zeichen x	a	b	c	d
Code $c(x)$	00	01	10	11

- ▶ Dadurch wird die Nachricht zu: 00.01.00.00.11.01.00.00.01.10.00.01.00.10.11.00
- ▶ Dieser Code ist ein sogenannter **Präfixcode**. In einem Präfixcode ist kein Codewort ein Präfix eines anderen Codewortes. Dadurch benötigen wir keine zusätzlichen Trennzeichen zwischen den codierten Zeichen.
- ▶ Die Nachricht ist 32 Bits lang. Dies macht im Durchschnitt **2 Bits je Zeichen**.

²**Sehr lesenswert:** Christopher Olah: “Visuelle Informationstheorie”.
<https://colah.github.io/posts/2015-09-Visual-Information/>



- **Idee: Häufige Elemente sollten kürzere Codes erhalten!**

Zeichen x	a	b	c	d
Wahrscheinlichkeit $P(x)$	1/2	1/4	1/8	1/8
Code $c(x)$	0	10	110	111

- Dies verwandelt die Nachricht in: 0.10.0.0.111.10.0.0.10.110.0.10.0.110.111.0
- Die Nachricht ist 28 Bits lang. Im Durchschnitt macht das **je Zeichen 1.75 Bits**. 😊
- Offensichtlich hängt das Potenzial zum "Bits sparen" davon ab, wie unausgewogen die Verteilung P ist...

Die Entropie



- ▶ Könnten wir im Beispiel der letzten Folie eine **effizientere Codierung** wählen?
- ▶ Dies beantwortet das zentrale Konzept der Informationstheorie, die **Entropie**!

Definition (Entropie)

Seien x_1, \dots, x_m die Realisierungen (Zeichen, Ereignisse, Klassen, ...) einer diskreten Zufallsvariablen X mit Verteilung $P = (p_1, \dots, p_m)$. Wir nennen

$$H(X) (= H(P)) = - \sum_i p_i \cdot \log_2(p_i)$$

die **Entropie** von X (oder P).

Beispiel

Zeichen x_1, \dots, x_4	a	b	c	d
Wahrscheinlichkeiten p_1, \dots, p_4	1/2	1/4	1/8	1/8
$-\log_2(p_i)$	1	2	3	3

$$\rightarrow \text{Entropie } H(P) = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = 1.75$$



- ▶ In der obigen Definition ist $0 \cdot \log_2(0) = 0$ (d.h., ein nie auftretendes Zeichen trägt nicht zur Gesamtcodewort-Länge bei).
- ▶ Die Entropie ist eine **untere Schranke** für die **durchschnittliche Zeichencodewort-Länge**, die durch **einen beliebigen Präfixcode c** erreichbar ist (Beweis: [1]):

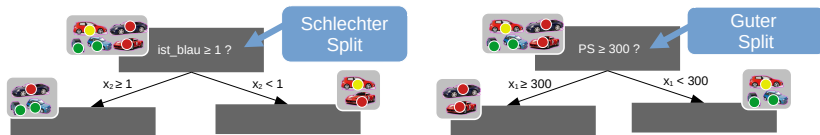
$$H(X) \leq \sum_i p_i \cdot \text{length}(c(x_i)) \quad \text{für alle Präfixcodes } c$$

- ▶ Die Entropie ist ein Maß für die **Zufälligkeit** einer Wahrscheinlichkeitsverteilung:
 - ▶ $P = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}) \rightarrow H(P) = 2$
 - ▶ $P = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}) \rightarrow H(P) = 1.75$
 - ▶ $P = (1, 0, 0, 0) \rightarrow H(P) = 0$

Anwendung auf Entscheidungsbäume ...?

- ▶ Wichtige Beobachtung: Die Entropie der Klassenverteilung eines Knotens ist **niedrig**, wenn der Knoten "pure" ist (d.h. eine bestimmte Klasse dominiert)!
- ▶ Der Split mit maximalem **Informationsgewinn** reduziert die Entropie der Kindknoten am stärksten. ☺

Zurück zu **Entscheidungsbäumen** – wir vergleichen die beiden Splits nach ihrer **Reinheit der Klassenlabels**!



- ▶ Wir überprüfen die **Klassenverteilung** im obersten Knoten und den Kindknoten
- ▶ Welcher Split führt zum **stärkeren Rückgang der Entropie**?

Beispiel: Aufteilung nach PS

- ▶ oben: $H^{oben} = H(p_{guenstig}, p_{mittel}, p_{teuer}) = H(\frac{2}{5}, \frac{1}{5}, \frac{2}{5}) = 1.52$
- ▶ links: $H^{links} = H(p_{guenstig}, p_{mittel}, p_{teuer}) = H(0, 0, 1) = 0$
- ▶ rechts: $H^{rechts} = H(p_{guenstig}, p_{mittel}, p_{teuer}) = H(\frac{2}{3}, \frac{1}{3}, 0) = 0.92$
- ▶ Informationsgewinn: $H^{oben} - \left(\frac{2}{5} \cdot H^{links} + \frac{3}{5} \cdot H^{rechts} \right) = 0.968$

Definition: Informationsgewinn



Gegeben ein Datensatz $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, sei $H(\mathcal{D})$ die Entropie der Verteilung der Labels y_1, \dots, y_n . Wir definieren den **Information Gain** eines Splits als:

$$Gain = H(\mathcal{D}) - \left(\frac{\#\mathcal{D}_1}{\#\mathcal{D}} \cdot H(\mathcal{D}_1) + \frac{\#\mathcal{D}_2}{\#\mathcal{D}} \cdot H(\mathcal{D}_2) \right)$$

wobei \mathcal{D} die Menge des Elternknotens und $\mathcal{D}_1, \mathcal{D}_2$ die Mengen der Kindknoten sind.

Bemerkungen

- Der Informationsgewinn ist immer ≥ 0 .



1. Grundlagen
2. Exkurs: Informationstheorie und Information Gain
3. Training und Pruning
4. Random Forests

Entscheidungsbaum-Classifizier: Pseudo-Code

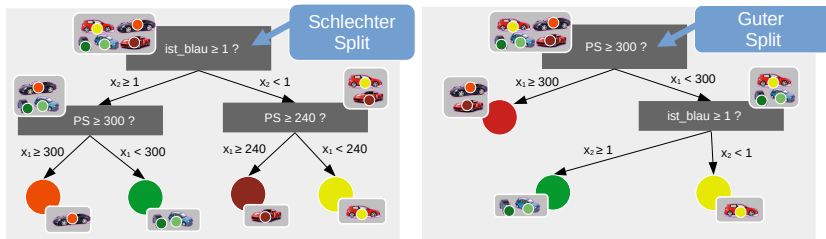


Ein **Knoten** im Baum ist ein Tupel (**Wert**, **Split**, **Kinder**), wobei

- ▶ **Wert** die Prognose des Modells \hat{y} ist (*nur definiert für Blattknoten*)
- ▶ **Split** Dimension und Schwellenwert, an dem geteilt wird, sind (*nur für innere Knoten*)
- ▶ **Kinder** sind Referenzen auf die beiden Kindknoten (*nur definiert für innere Knoten*)

```
1 function build_tree( $\mathcal{D}$ ):
2
3     # check *all* dimensions  $i$  and thresholds  $t$ , and
4     # pick the one with highest information gain
5      $i^*, t^* := \operatorname{argmax}_{i,t} \operatorname{Gain}(\mathcal{D}, i, t)$ 
6
7     # no improvement? -> build a leaf node
8     if  $\operatorname{Gain}(\mathcal{D}, i^*, t^*) == 0$ :
9         let  $y_1, \dots, y_n$  be the labels in  $\mathcal{D}$ 
10         $\hat{y} := \operatorname{mode}(y_1, y_2, \dots, y_n)$     # prediction = the leaf's most frequent label
11        return ( $\hat{y}$ , -, {})    # value= $\hat{y}$ , no split, no children
12
13    # else -> build an inner node
14    use  $i^*, t^*$  to split  $\mathcal{D}$  into subsets  $\mathcal{D}_1, \mathcal{D}_2$ 
15    return ( -, ( $i^*, t^*$ ), {build_tree( $\mathcal{D}_1$ ),
16                                build_tree( $\mathcal{D}_2$ ) } )    # recursion!
```


Split-Kriterium für Regression: Varianzreduktion



Gute Splits finden: Beispiel (Varianzreduktion)

Gegeben sei ein Datensatz $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, wobei $Var(\mathcal{D})$ die Varianz der Labels y_1, \dots, y_n von \mathcal{D} ist. Wir definieren den **Gain** eines Splits als:

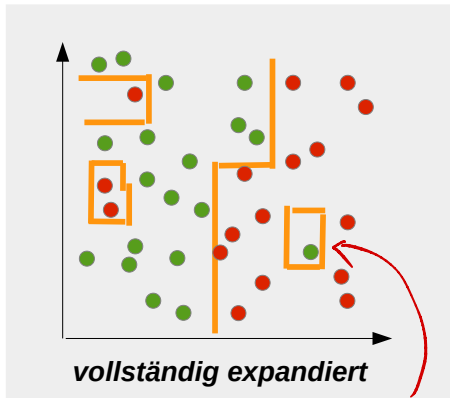
$$Gain = Var(\mathcal{D}) - \left(\frac{\#\mathcal{D}_1}{\#\mathcal{D}} \cdot Var(\mathcal{D}_1) + \frac{\#\mathcal{D}_2}{\#\mathcal{D}} \cdot Var(\mathcal{D}_2) \right)$$

wobei \mathcal{D} die Menge des Elternknotens und $\mathcal{D}_1, \mathcal{D}_2$ die Mengen der Kindknoten sind.

Entscheidungsbäume und Overfitting



Herausforderung: Kleine Bäume neigen zu **Underfitting**, vollständig expandierte Bäume zu **Overfitting**.

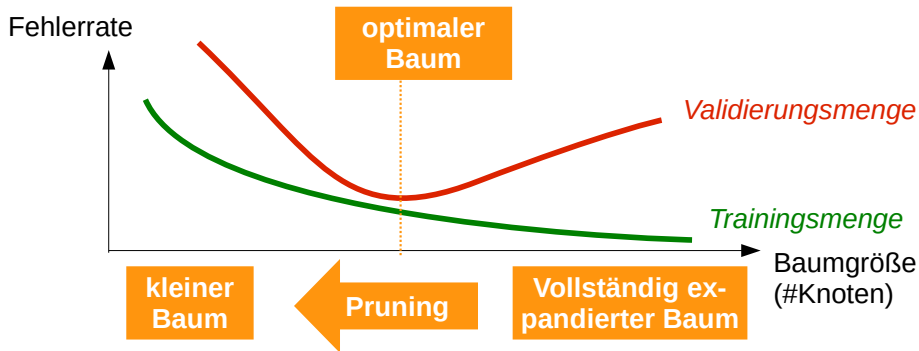


*Is2
ist auch
eine Insel*

Pruning: Die optimale Baumgröße finden

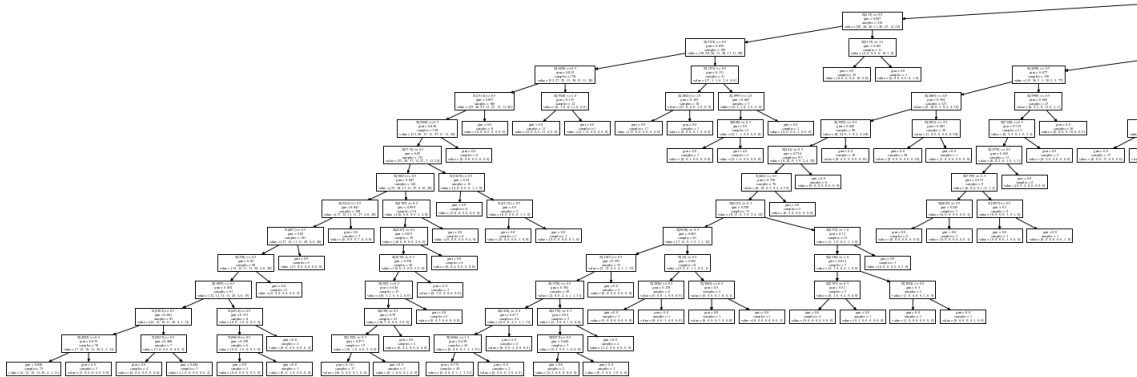


1. Wir teilen unseren Datensatz in einen **Trainingsdatensatz** und einen **Validierungsdatensatz** auf.
2. Wir trainieren einen **vollständig expandierten** Baum mit dem Trainingsdatensatz.
3. Wir entfernen sukzessive Blattknoten, solange der Fehler auf dem **Validierungsdatensatz** abnimmt.





Sind Entscheidungsbäume gut für unsere Bag-of-Words-Nachrichtenklassifizierung?





1. Grundlagen
2. Exkurs: Informationstheorie und Information Gain
3. Training und Pruning
4. Random Forests

- ▶ Wie wir gesehen haben, kann **Overfitting** eine große Herausforderung bei Entscheidungsbäumen sein.
- ▶ Ein Ansatz ist, den Baum auf eine begrenzte Größe zu **beschneiden**.
- ▶ Ein weiterer üblicher Ansatz ist, **mehrere Bäume** zu einem sogenannten **Ensemble** zu kombinieren.

Ensembles

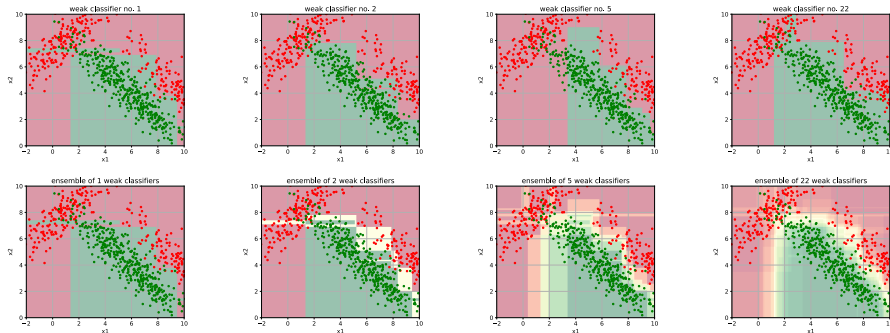
- ▶ Es gibt viele Ansätze, um Ensembles zu erstellen.
- ▶ Sehr beliebt sind **Boosting**-Methoden, insbesondere **Gradient Boosting** (z.B. *XGBoost*).
- ▶ Wir werden uns eine einfachere Methode ansehen, sogenannte **Random Forests**.



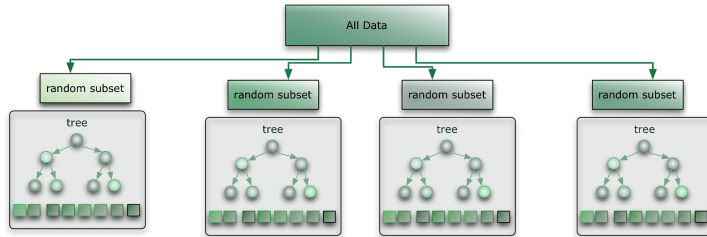
Random Forests: Beispiel



- **Entscheidungsbäume** werden zu einem Random Forest kombiniert, indem man ihre Entscheidungen **mittelt**.



- Durch die Kombination der Bäume wird die Klassifikation stabiler. Wir haben die **Varianz verringert!** 😊



- Einfacher Ansatz: Der Input wird mit jedem Baum klassifiziert und die Ergebnisse der einzelnen Bäume werden mittels Mehrheits-Voting oder Mittlung kombiniert.

Wieso sind die Bäume des Random Forests nicht identisch?

Wir *randomisieren* die Bäume:

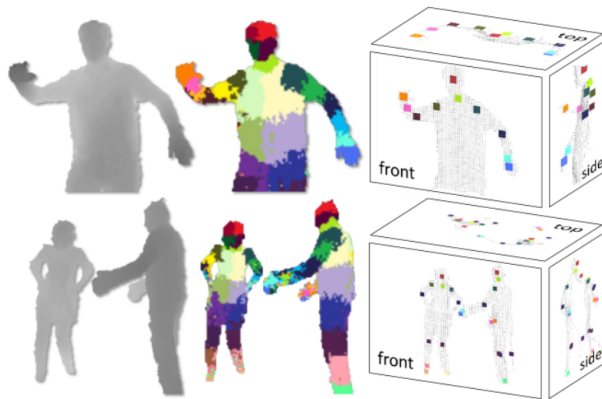
1. Die Bäume werden auf zufälligen Teilmengen der Trainingsmenge trainiert.
2. Beim Aufteilen jedes Knotens im Training verwenden wir nur eine **zufällig ausgewählte Teilmenge von Merkmalen!**



Beispiel-Evaluation [4]: Fehlerquoten auf einer Vielzahl von Standarddatensätzen aus dem bekannten *UCI Machine Learning Repository*.

Data set	Adaboost	Selection	Forest-RI single input	One tree
Glass	22.0	20.6	21.2	36.9
Breast cancer	3.2	2.9	2.7	6.3
Diabetes	26.6	24.2	24.3	33.1
Sonar	15.6	15.9	18.0	31.7
Vowel	4.1	3.4	3.3	30.4
Ionosphere	6.4	7.1	7.5	12.7
Vehicle	23.2	25.8	26.4	33.1
German credit	23.5	24.4	26.2	33.3
Image	1.6	2.1	2.7	6.4
Ecoli	14.8	12.8	13.0	24.5
Votes	4.8	4.1	4.6	7.4
Liver	30.7	25.1	24.7	40.6
Letters	3.4	3.5	4.7	19.8
Sat-images	8.8	8.6	10.5	17.2
Zip-code	6.2	6.3	7.8	20.6
Waveform	17.8	17.2	17.3	34.0
Twonorm	4.9	3.9	3.9	24.7
Threenorm	18.8	17.5	17.5	38.4
Ringnorm	6.9	4.9	4.9	25.7

Kinect Body Part Recognition³



³Shotton et al.: Real-Time Human Pose Recognition in Parts from Single Depth Images (Microsoft Research), CVPR 2011.

References I



- [1] Michael Langer: Entropy is a lower bound on average code length (lecture slides, Jan 2008).
<http://www.cim.mcgill.ca/~langer/423/lecture5.pdf> (retrieved: Oct 2016).
- [2] RandomForest.
<http://randomforest2013.blogspot.de/2013/05/randomforest-definicion-random-forests.html> (retrieved: Oct 2016).
- [3] USA mathematician and electronic engineer Claude Shannon.
https://commons.wikimedia.org/wiki/File:Claude_Shannon_graffiti.jpg (retrieved: Oct 2016).
- [4] Leo Breiman.
Random forests.
Mach. Learn., 45(1):5–32, October 2001.
- [5] M. Rosenthal.
Computer Repair with Diagnostic Flowcharts: Troubleshooting PC Hardware Problems from Boot Failure to Poor Performance.
Foner Books, 2003.
- [6] G. Seni and J. Elder.
Ensemble Methods in Data Mining: Improving Accuracy through Combining Predictions.
Morgan and Claypool, 2010.