Datenkompression:

Vektorquantisierung

H. Fernau

email: fernau@uni-trier.de

WiSe 2008/09

Universität Trier

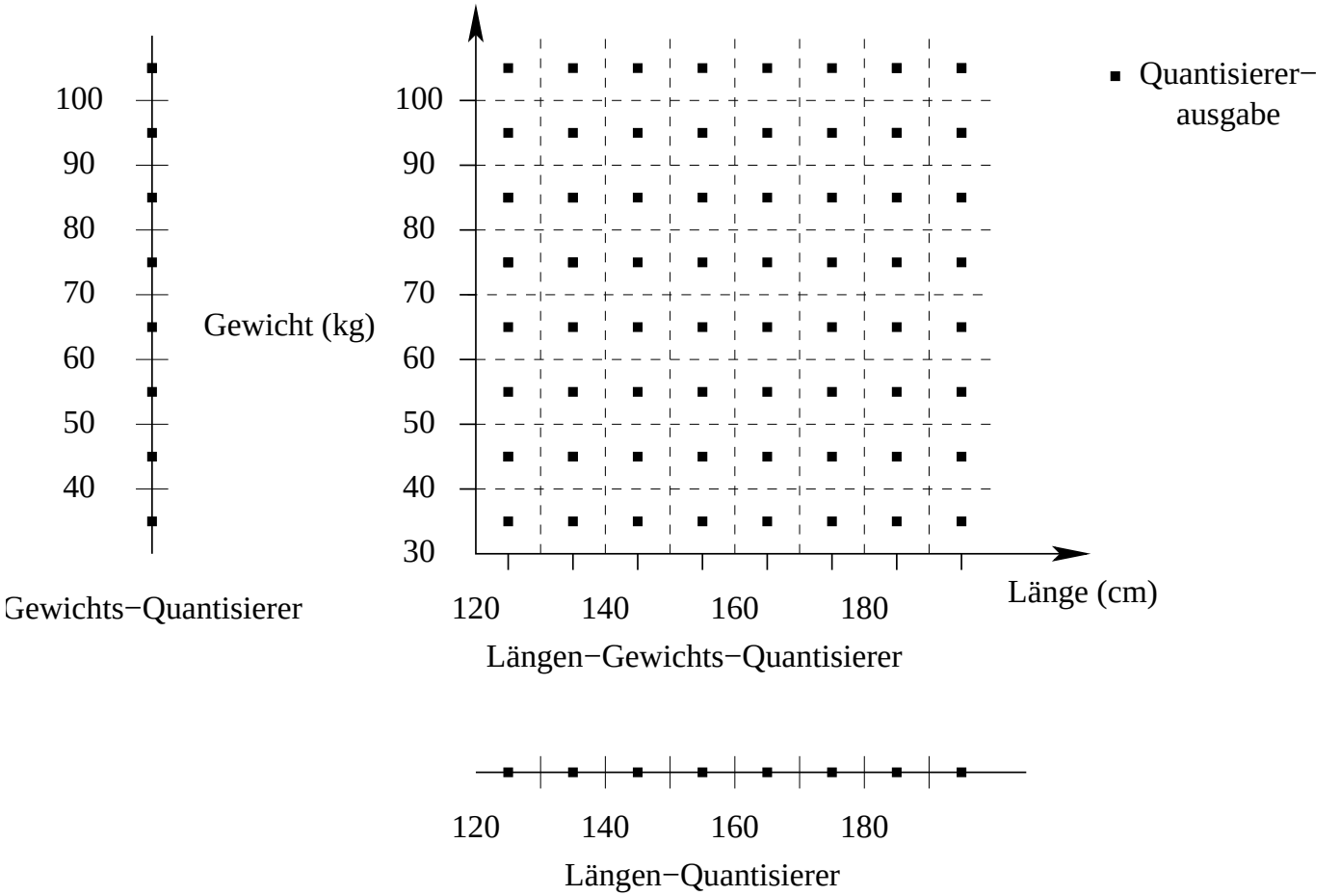
Vektorquantisierung—Allgemeines

Bislang: Versuch, einzelne Quellenausgaben (also skalare Werte) zu quantisieren *skalare Quantisierung*

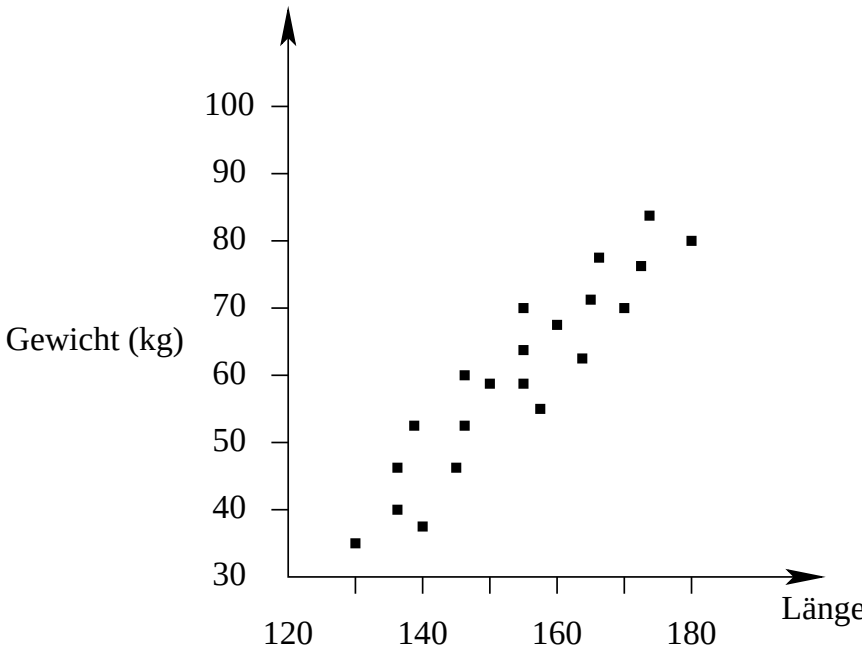
Daher unberücksichtigt: räumliche (Bsp.: Bilder) oder zeitliche (Bsp.: Sprache) Abhängigkeiten zwischen aufeinander folgenden Quellenausgaben.

(schon Grundidee bei adaptiver Quantisierung)

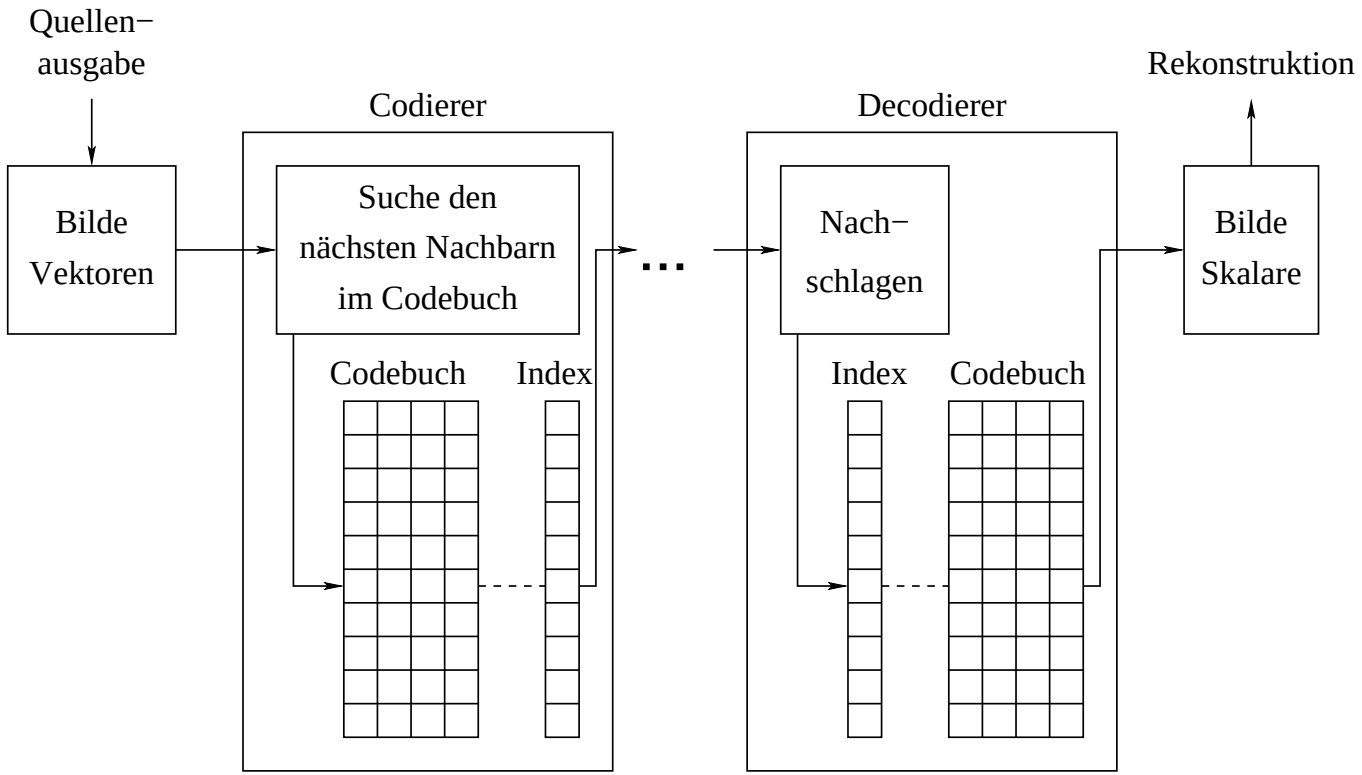
Beispiel 1: Länge und Gewicht von Personen.



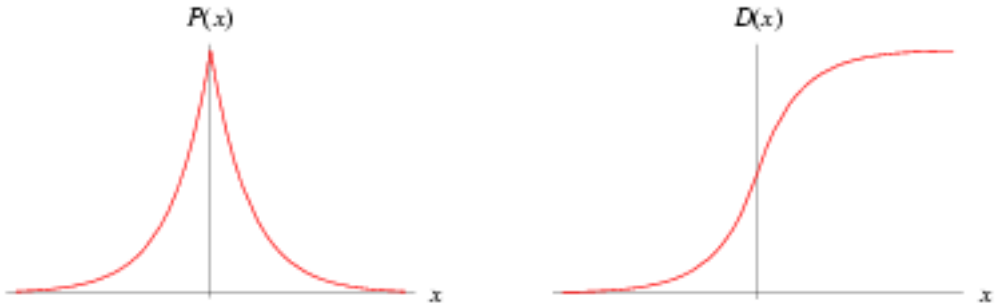
Eine Stichprobe von Länge und Gewicht von Personen



Die Arbeitsweise eines Vektorquantisierers



Beispiel 2: Diskussion einer Laplace-verteilten Quelle

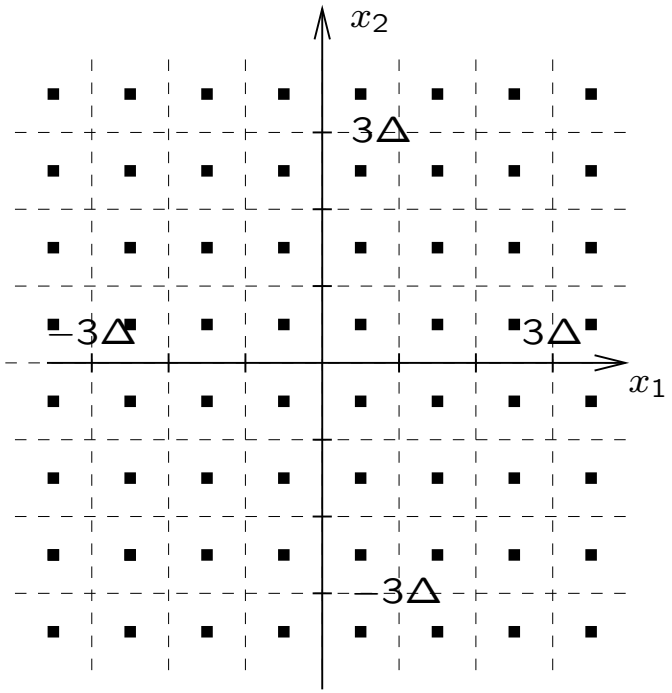


Die Laplace-Verteilung (Quelle: `mathworld.wolfram.com`)

Wahl der Schrittweite

Erinnerung: 3-Bit-Gleichquantisierer für Laplace-verteilte Quelle:

$\rightsquigarrow$ Optimum $\Delta = 0,7309$.



Für $\Delta = 0,7309$ bekommen wir: SNR=11,44 dB.

Erinnerung: 3-Bit-Gleichquantisierer für Laplace-verteilte Quelle: $\rightsquigarrow$ Optimum $\Delta = 0,7309$.

Optimale Schrittweiten für Gleichquantisierer

Alph.-größe	Gleichverteilung Schrittweite	SNR	Gaußverteilung Schrittweite	SNR	Laplaceverteilung Schrittweite	SNR
2	1.732	6.02	1.596	4.40	1.414	3.00
4	0.866	12.04	0.9957	9.24	1.0873	7.05
6	0.577	15.58	0.7334	12.18	0.8707	9.56
8	0.433	18.06	0.5860	14.27	0.7309	11.39
10	0.346	20.02	0.4908	15.90	0.6334	12.81
12	0.289	21.60	0.4238	17.25	0.5613	13.98
14	0.247	22.94	0.3739	18.37	0.5055	14.98
16	0.217	24.08	0.3352	19.36	0.4609	15.84
32	0.108	30.10	0.1881	24.56	0.2799	20.46

Hier gilt aber:

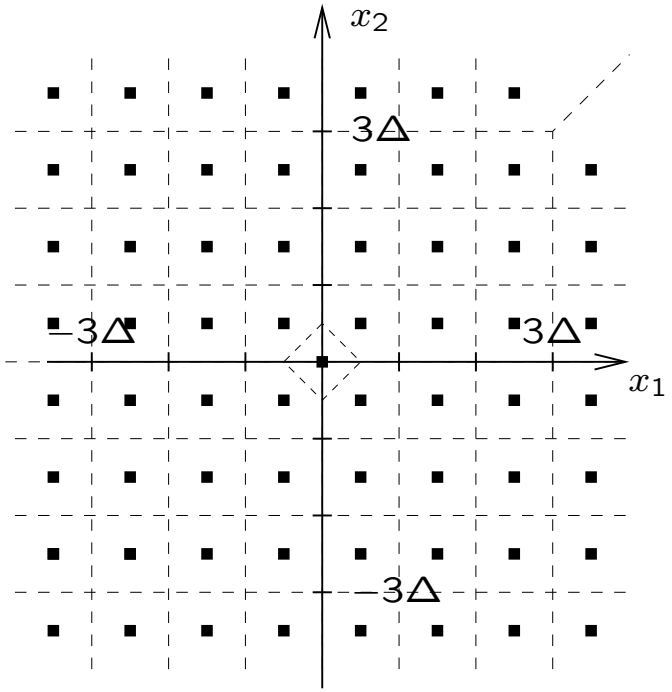
$$P(x \in [0, \Delta)) = 0,3242 \; und \; P(x \in [3\Delta, \infty)) = 0,0225.$$

Dann (unter Annahme der Unabhängigkeit aufeinander folgender Ereignisse)

$$P((x, y) \in [3\Delta, \infty) \times [3\Delta, \infty)) = P(x \in [3\Delta, \infty)) \cdot P(y \in [3\Delta, \infty)) = 0,0005\,.$$

$\rightsquigarrow$ Idee: Lege unwahrscheinliche Bereiche zusammen und unterteile dafür wahrscheinlichere

64-elementiges Codebuch eines 2-dimensionalen Vektorquantisierers



Für den Quantisierer ist SNR=11,75 dB (vorher SNR=11,44 dB)

Problem: Entwurf guter Codebücher

$\rightsquigarrow$ Modifikation des Lloydschen Algorithmus

Das *Lindo-Buzo-Gray-Verfahren I* ist auf der nächsten Folie erläutert.

An die Stelle der Entscheidungsgrenzen treten die *Quantisier-Regionen*, und die Verzerrung bzw. Zentroide sind entsprechend kompliziertere Integrale.

1. Wähle Anfangsrepräsentanten $\{Y_i^{(0)}\}_{i=1}^M$ & Abbruchschwellwert ε für die Verzerrungsdifferenz. Setze $D^{(-1)} = 0$, $k = 0$.
2. Berechne neue Quantisierungsregionen mit:

$$V_j^{(k)} = \{X \mid \forall i \neq j : d(X, Y_j) < d(X, Y_i)\}\,.$$

3. Berechne die neue Verzerrung

$$D^{(k)} = \sum_{i=1}^{M} \int_{V_i^{(k)}} \left\| \vec{x} - Y_i^{(k)} \right\|^2 f_X(\vec{x}) d\vec{x}\,.$$

4. Falls $\frac{\left| D^{(k)} - D^{(k-1)} \right|}{D^{(k)}} < \varepsilon$, STOP.
5. Sonst: Setze $k = k + 1$ und berechne neue Repräsentanten als Zentroide:

$$Y_j^{(k)} = \frac{\int_{V_j^{(k-1)}} \vec{x} f_X(\vec{x}) d\vec{x}}{\int_{V_j^{(k-1)}} f_X(\vec{x}) d\vec{x}}\,.$$

   Gehe zu Schritt 2.

Das Lindo-Buzo-Gray-Verfahren II

Kompliziertheit der auftretenden Integrale

$\rightsquigarrow$ vereinfachte, sozusagen diskretisierte Version

Bei ihr wird mit einer *Trainingsmenge* $\{X_n\}_{n=1}^{N}$ gearbeitet.

Dieses Verfahren kommt ohne Annahmen über die Quellenstatistik aus;

Einzelheiten siehe nächste Folie.

Es steht eine Trainingsmenge $\{X_n\}_{n=1}^N$ zur Verfügung.

1. Wähle Anfangsrepräsentanten $\{Y_i^{(0)}\}_{i=1}^M$ und Abbruchschwellwert ε für die Verzerrungsdifferenz. Setze $D^{(-1)} = 0$, $k = 0$.

2. Berechne neue Quantisierungsregionen mit:

$$V_j^{(k)} = \{X_n \mid \forall i \neq j : d(X_n, Y_j) < d(X_n, Y_i)\}\,.$$

   Sollte ein $V_j^{(k)}$ leer sein, so teile man die größte Region in zwei "Hälften".

3. Berechne die neue Verzerrung

$$D^{(k)} = \sum_{i=1}^{M} \sum_{X_n \in V_i^{(k)}} \|X_n - Y_i\|^2\,.$$

4. Falls $\frac{|D^{(k)} - D^{(k-1)}|}{D^{(k)}} < \varepsilon$, STOP.

5. Sonst: Setze $k = k+1$ und berechne neue Repräsentanten als Mittelwerte:

$$Y_j^{(k)} = \frac{\sum_{X_n \in V_j^{(k)}} X_n}{\left|\{X_n \in V_j^{(k)}\}\right|}\,.$$

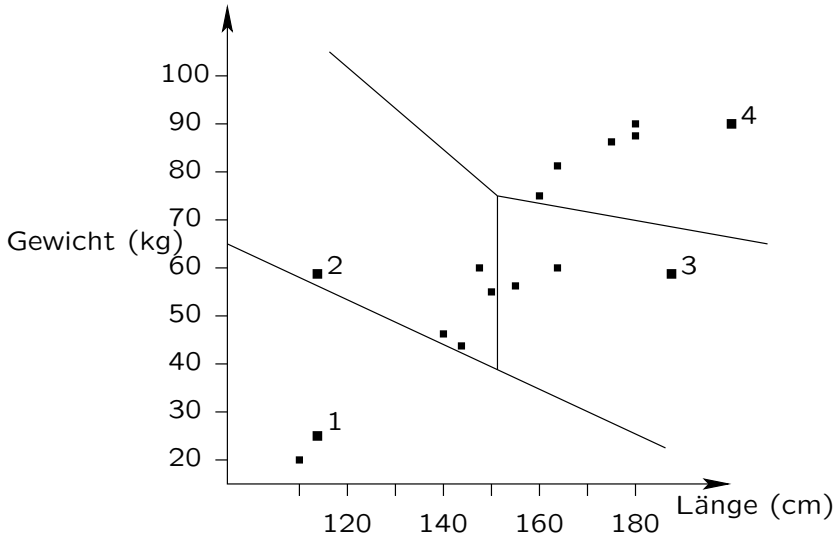
   Gehe zu Schritt 2.

Länge	Gewicht
180	90
180	88
163	60
110	20
148	60
155	57
160	75
150	55
163	81
140	46
143	44
175	86

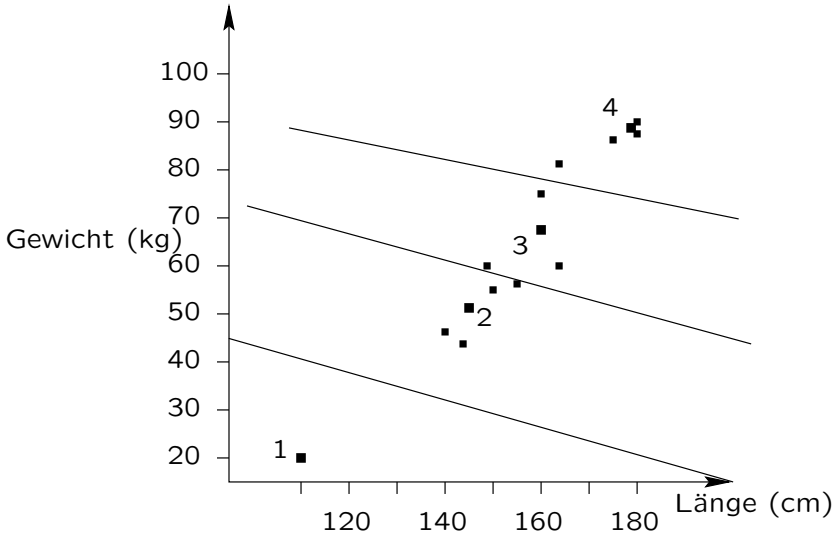
Trainingsmenge für den Codebuchentwurf

Länge	Gewicht
113	25
113	59
188	59
200	90

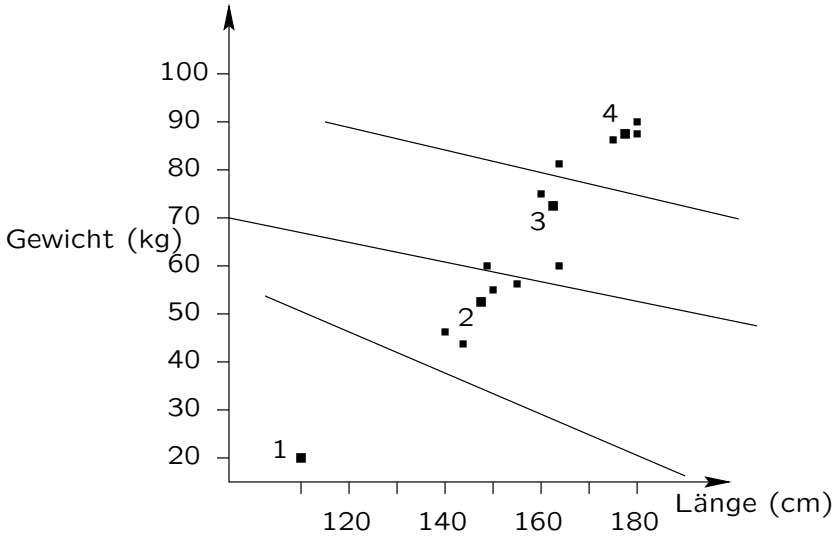
Anfangscodebuch



Anfängliche Quantisierungsregionen



Regionen nach einem Durchlauf des LBG-Verfahrens



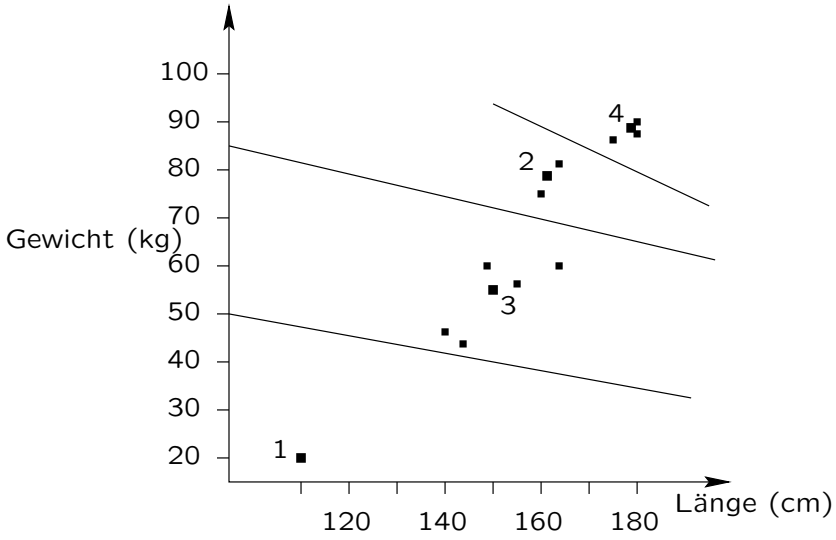
Regionen nach zwei Durchläufen des LBG-Verfahrens

Länge	Gewicht
188	25
188	64
188	59
200	90

Ein anderes Anfangscodebuch. . .

Länge	Gewicht
110	20
161,5	78
149,8	53,7
178,3	88

. . . führt zu einem anderen Endcodebuch. . .



. . . mit den angegebenen Regionen.

Problem: Entwurf eines guten *Anfangscodebuchs*.

1. Lindo, Buzo und Gray schlugen die *Aufspalttechnik* vor.
   Der Abbruchschwellwert ε und die *Vektorstörung* δ sowie die Trainingsmenge sind als vorweg gegebene Parameter aufzufassen.

2. Eine quasi entgegengesetzte Idee zum Entwurf des Anfangscodebuchs hatte Equitz (*Equitz-Verfahren*).
   Ausgehend von einer Trainingsmenge werden solange paarweise nächstbenachbarte Cluster vereinigt, angefangen mit der Clustermenge $\{\{X_n\}\}_{n=1}^{N}$, bis der erwünschte Codebuchumfang erreicht ist.

3. Häufig am einfachsten und besten: bestimme Anfangscodebuch durch Würfeln als Teilmenge der Trainingsmenge.
   Man kann dann den LBG-Algorithmus auf mehreren solchen Anfangscodebüchern laufen lassen und schließlich das beste Codebuch sich heraussuchen.

Das Lindo-Buzo-Gray-Verfahren III

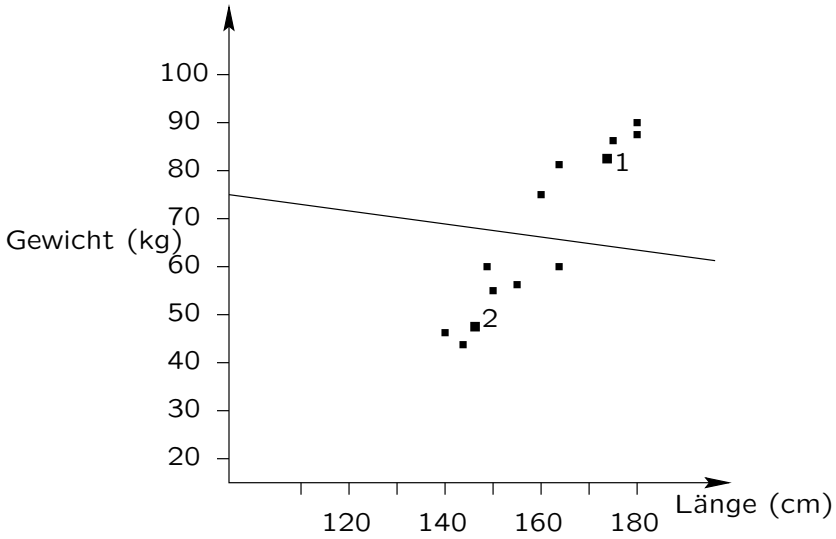
Entwirf K Bit großes Anfangscodebuch:

1. Falls $K = 0$: Nimm Mittelwert der Trainingsmenge $\{X_n\}_{n=1}^{N}$ als Codevektor.
2. Falls $K > 0$: Entwirf (rekursiv) $K - 1$ Bit großes Anfangscodebuch;
   Initialisiere K-Bit Codebuch durch 2^{K-1} so erhaltene Vektoren sowie um fixes δ gestörte Vektoren;
   Starte LBG-Algorithmus zur Ermittlung eines K-Bit-Anfangscodebuchs.

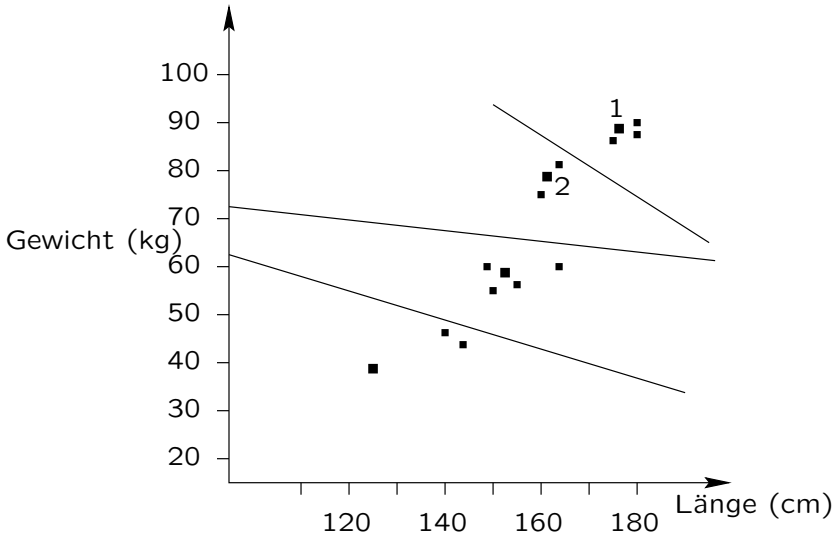
Die Aufspalttechnik am Beispiel

Codebuch	Länge	Gewicht
0 Bit	155	64
1 Bit Anfangs-	155	64
	180	69
1 Bit End-	145	49
	173	84
2 Bit Anfangs-	145	49
	171	54
	173	84
	198	89
2 Bit End-	130	37
	155	58
	163	78
	178	88

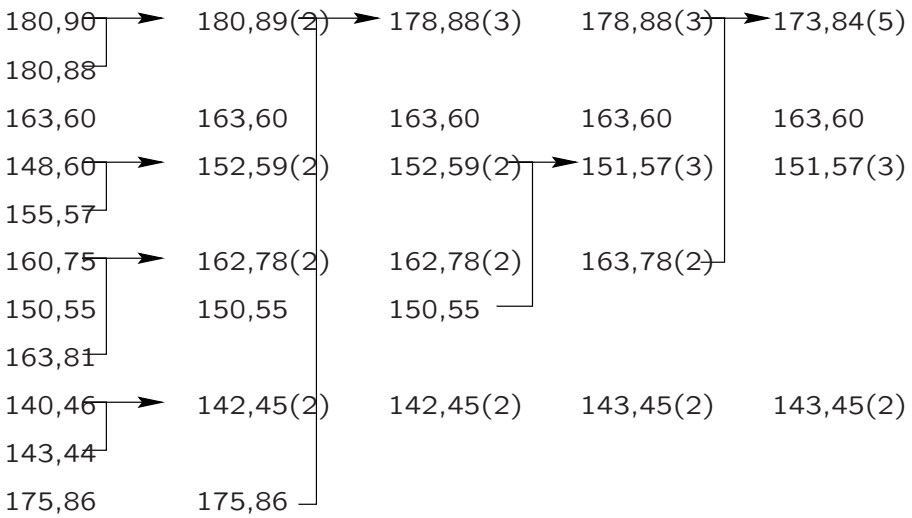
Ein-Bit Vektorquantisierer



Zwei-Bit Vektorquantisierer



Der Ansatz von Equitz



Zwei grundsätzliche Probleme birgt das LBG-Verfahren noch in sich:

1. Woher bekommt man "gute" Trainingsmengen für eine Anwendung?

   Bilder haben häufig unterschiedliche Charakteristika; ein auf ein Bild optimiertes Codebuch mag für ein anderes Bild nur bedingt brauchbar sein. Umgekehrt mindert ein als Begleitinformation übertragenes komplettes Codebuch natürlich die (bei Vektorquantisierern ansonsten hervorragende) Kompressionsrate, s. Tabellen auf der nächsten Folie.

2. Das Auffinden des nächsten Codevektors kostet bei großen Codebüchern viel Zeit.

Verschiedene Kompresssionsmaße für ein Grauwertbild mit 8 Bit pro Pixel Auflösung für ein 16D Quantisierer (4×4 Blöcke)

Codebuchumfang K	$\log_2 K$	Bits pro Pixel	Kompressionsrate
16	4	0,25	32:1
64	6	0,375	21,33:1
256	8	0,50	16:1
1024	10	0,625	12.8:1

Overhead für die Codebuchübertragung

Codebuchumfang K	Overhead (Bits pro Pixel)
16	0,03125
64	0,125
256	0,50
1024	2,0