ROZDZIAŁ 10. KWANTYZACJA WEKTOROWA

Przedstawiona w tym rozdziale technika stratnej kompresji jako główny element zawiera schemat efektywnej kwantyzacji wektorowej. Opisany został tutaj sposób konstrukcji zbioru wektorów kodu, przy pomocy których przybliżane są kolejne wektory danych, oraz zasadnicze ograniczenia implementacji tejże metody kompresji. Wektorowa kwantyzacja traktowana jest w tym rozdziale jako samodzielna technika kompresji, jakkolwiek może ona stanowić jedynie element bardziej złożonych technik kompresji, poprzedzona różnymi algorytmami dekompozycji danych oryginalnych i formowania wektorów, a zakończona różnymi sposobami kodowania indeksów wektorowej kwantyzacji, łączenia ich z innymi strumieniami danych, itd. Kilka przykładów bardziej złożonych zastosowań kwantyzacji wektorowej zostało przedstawionych w ostatnim podrozdziale.

10.1. Wprowadzenie

Pierwszym krokiem w technice kompresji zwanej kwantyzacją wektorową (ang. vector quantization - VQ) jest dekompozycja oryginalnego zbioru danych na *n*-wymiarowe wektory danych. Wektory mogą być tworzone na wiele różnych sposobów. Do najczęstszych należy formowanie trójwymiarowych wektorów trzech składowych koloru, np. RGB, YUV, pojedynczych pikseli obrazu czy też *n*-wymiarowych wektorów z uporządkowanych kolejno wartości pikseli bloku o wymiarach $n = k \cdot l$, gdzie wartości k i l określają rozmiar bloku. Można także składać wektory z kilku równoległych strumieni danych generowanych w pewnym procesie fizycznym. Ponadto, zbiór danych może zostać podzielony na bloki, w których ulega dekompozycji przy pomocy np. transformaty Fouriera czy kosinusowej, a współczynniki transformaty z każdego bloku stają się wektorami. Postać wektorów powinna jednoznacznie wynikać z charakteru informacji reprezentowanej w kompresowanym zbiorze danych i uwzględniać jej strukturę oraz rodzaj zależności pomiędzy danymi w zbiorze.

Każdy wektor danych jest następnie porównywany z pewnym zbiorem reprezentatywnych wzorców, zwanych wektorami kodu, który pochodzi z utworzonej wcześniej *n*-wymiarowej księgi kodów (wektorów kodu) $\mathbf{K}$. Najodpowiedniejszy wektor jest dobierany na podstawie zdefiniowanej reguły minimalnego zniekształcenia. Miara tego zniekształcenia powinna być z jednej strony prosta obliczeniowo, z drugiej zaś dobrze odpowiadać rzeczywistej degradacji informacji w procesie zastępowania danego wektora poprzez wybrany wektor kodu. Najczęściej stosowaną w VQ miarą zniekształceń (błędu kwantyzacji) jest błąd średniokwadratowy (MSE), który odpowiada kwadratowi Euklidesowej odległości pomiędzy dwoma wektorami. W niektórych aplikacjach stosowany jest ważony błąd średniokwadratowy, uwzględniający np. perceptualną ocenę poziomu zniekształcenia przy pomocy modelu HVS (patrz poprzedni rozdział) w kompresji obrazów.

Ostatni etap procesu kompresji polega na zapisaniu do pliku (lub transmisji) indeksów wybranych wektorów kodu, które najlepiej przybliżają wektory danych. Bitowa długość indeksu wynika jednoznacznie z ilości wektorów kodu, czyli rozmiaru książki kodowe j N_{K} i wynosi $\log_2 N_{\mathrm{K}}$ bitów. Przy dekompresji indeksy te służą do odczytania odpowiednich wektorów z księgi kodów, z których składany jest rekonstruowany zbiór danych, co czyni algorytm dekompresji bardzo prostym. Algorytm kwantyzacji w schemacie kompresji, w którym poszczególnym indeksom kwantyzacji przypisywane słowa kodowe o ustalonej, równej długości nazywany jest kwatyzatorem o ustalonej średniej bitowej (ang. fixed-rate

quantizer). Są przypadki, kiedy korzystniej jest zakodować ten zbiór indeksów metodą kodów o zmiennej długości. Zostaną one przedstawione później. We wstępnych rozważaniach przyjmijmy model kwantyzacji o ustalonej średniej bitowej.

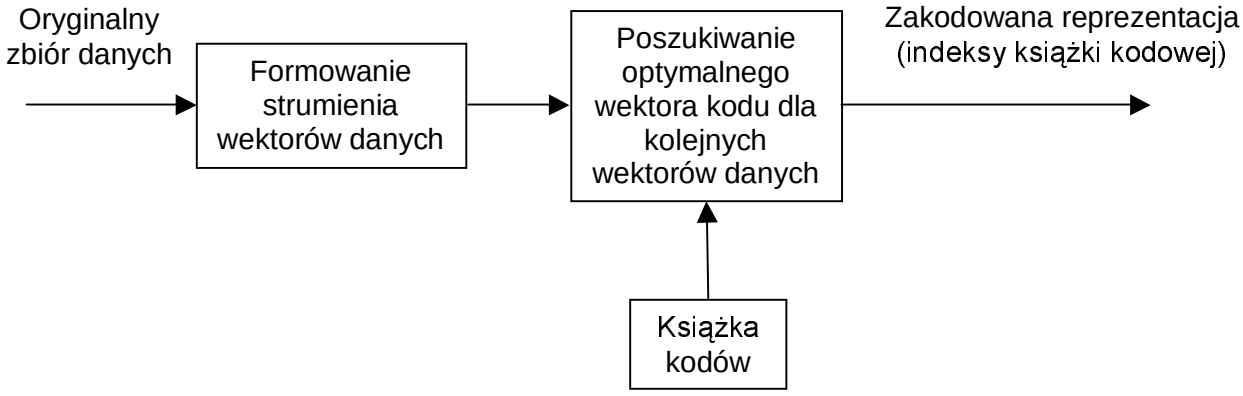
Zasadniczo kompresja w VQ osiągana jest w przypadku, gdy liczba wektorów księgi kodów jest mniejsza od ilości różnorodnych wektorów danych, czyli analogicznie jak w kwantyzacji skalarnej większą dynamikę zbioru danych, w tym przypadku reprezentowanego poprzez wektory, zastępujemy mniejszą dynamiką wybranych reprezentantów. Dokładniej rzecz biorąc uzyskana średnia bitowa reprezentacji kodowej jest równa:

$$BR = \frac{\log_2 N_{\text{K}}}{n} \text{ bitów/symbol,} \qquad (10.1)$$

gdzie n- wymiar wektorów danych. Teoretycznie VQ może osiągnąć efektywność bliską granicznej krzywej $BR(D)$, jeśli $n \to \infty$. W rzeczywistości wzrost wymiarowości wektorów, jakkolwiek zmniejsza średniokwadratowy błąd kwantyzacji, to jednak prowadzi do problemów związanych z konstrukcją, zapisaniem i przeszukiwaniem ogromnej księgi kodów.

10.2. Schemat metody

Zasadniczym etapem algorytmu kwantyzacji wektorowej jest tworzenie efektywnej książki kodów, odpowiednika słownika w bezstratnych metodach słownikowego kodowania. Od jakości tej książki zależy powodzenie całej metody. Jest to jednak etap wstępny i nie jest ujęty w schemacie samego procesu kompresji. Po przygotowaniu książki kodowej podstawowy schemat algorytmu kompresji metodą wektorowej kwantyzacji wygląda jak na rys. 10.1.

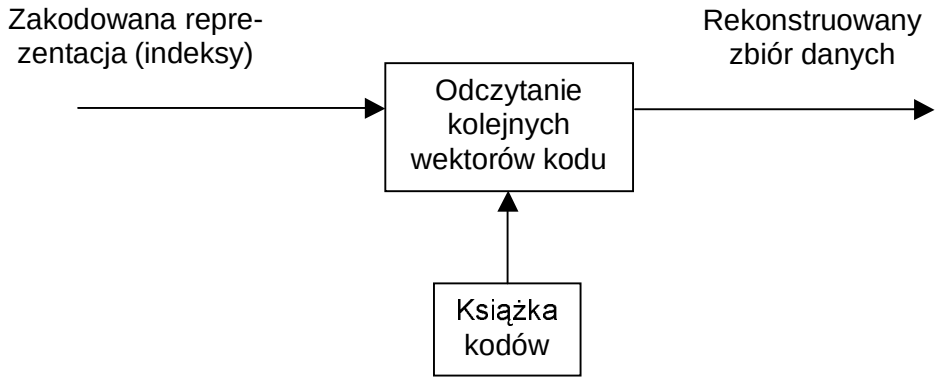


Rys. 10.1. Podstawowy schemat blokowy kompresji danych metodą wektorowej k wantyzacji. Schemat ten może być uzupełniony o blok odwracalnego kodera, który dodatkowo skraca długość reprezentacji wyjściowej kodując indeksy metodą kodu o zmiennej długości.

Zasadniczym elementem tego schematu jest środkowy blok wyszukiwania optymalnego wektora kodu ze zbioru, jakim jest książka kodów. Jest to proces czasochłonny,

decydujący o stopniu złożoności całej metody. Optymalny wektor kodu znaczy tutaj najbardziej podobny lub najmniej odległy od wektora danych w sensie przyjętego kryterium.

Metoda wektorowej kwantyzacji jest silnie niesymetryczna jako metoda kompresji. Czas kompresji, właśnie ze względu na konieczność wielokrotnego przeszukiwania książki kodów, jest znacznie dłuższy od czasu dekompresji, kiedy to jedynie na podstawie odczytywanych indeksów pobierane są kolejno z książki kodów (realizowanej najczęściej w konwencji 'look-up table') odpowiednie wektory kodu. Schemat dekompresji pokazany został na rys. 10.2.



Rys. 10.2. Podstawowy schemat blokowy algorytmu dekompresji danych w metodzie wektorowej kwantyzacji. Gdyby indeksy wejściowe były kodowane, wówczas należałoby jako pierwszy element schematu umieścić dekoder.

Bardzo istotnym etapem w technice VQ jest konstrukcja księgi kodowej zwana fazą klasteryzacji. Jest ona głównym czynnikiem różnicującym wiele opracowanych w ostatnich latach algorytmów VQ, a w jej skład wchodzą dwa zasadnicze procesy:

- generacja księgi kodów, ustalająca jakie wektory kodu winny się znaleźć w zbiorze wektorów przybliżającym wektory danych;
- projektowanie struktury księgi kodowej, która umożliwi szybkie przeszukiwanie i efektywną reprezentację wektorów kodu.

Oczywiście te dwa elementy są wzajemnie zależne i zazwyczaj przyjęta jako założenie pewna struktura książki kodów warunkuje sposób generacji tejże księgi. Możliwych rozwiązań jest tutaj bardzo wiele - przedstawione zostaną jedynie metody podstawowe, najczęściej stosowane.

10.3. Generacja książki kodowej

Spośród znanych technik budowania książki kodowej najbardziej znanym i stosowanym w licznych aplikacjach stratnej kompresji jest algorytm opracowany przez zespół Linde, Buzo i Gray w roku 1980 [1], nazwany od pierwszych liter autorów algorytmem LBG. Algorytm ten wykorzystuje treningowe zbiory danych, reprezentatywne dla klasy danych, które są przedmiotem kompresji. W skrajnym przypadku można wygenerować optymalną książkę kodów dla pojedynczego zbioru danych, ale jest to kosztowne czasowo i istnieje konieczność

dołączenia księgi kodowej do skompresowanej reprezentacji, co w praktyce przekreśla w wielu przypadkach jakąkolwiek efektywność kompresji.

Inne metody budowania książki kodowej to przede wszystkim algorytmy oparte na koncepcji sieci neuronowych, w tym głównie uczenia sieci bez nauczyciela. Do najbardziej znanych należy algorytm Kohonena [2] [3] bazujący na samoorganizujących się mapach cech (ang. self organizing feature map - SOFM) czy też algorytm propagacji wstecznej. Metody te, podobnie jak algorytm LBG, pozwalają znaleźć rozwiązane zbieżne jedynie do lokalnego minimum błędu kwantyzacji, dlatego też dużej wagi nabiera forma inicjalizacji algorytmu. Istnieją również metody, które pozwalają bardziej skutecznie wyznaczyć optymalny w sensie globalnym zbiór wektorów kodu, ale są to zazwyczaj schematy zbyt złożone obliczeniowo, by mogły znaleźć zastosowanie w praktycznych algorytmach kompresji. Jedna z nich to metoda deterministycznego chłodzenia (ang. deterministic annealing) polegająca na konstrukcji i rozwiązaniu całej rodziny problemów optymalizacyjnych odpowiednio dobranych dla zadanego problemu optymalizacji książki kodów [4]. Wprowadza się model kwantyzatora rozmytego narzucając spełnienie warunku maksymalnej entropii, po czym następuje redukcja entropii do zera, co oczywiście prowadzi do deterministycznego modelu kwantyzacji. Inną techniką jest symulowane chłodzenie (stochastyczna relaksacja), sprowadzająca się do statystycznego rozmywania modelu, a następnie redukcji tego rozmycia. Przykładowo, po wykryciu lokalnego minimum błędu kwantyzacji wykonuje się losowy skok lub dodaje się szum biały, po czym następuje redukcja tego zaburzenia. Żadna z tych metod nie daje pełnej gwarancji dotarcia do globalnego optimum. Założenia wykorzystywanych teorii są warunkami koniecznymi, ale nie wystarczającymi dla znalezienia globalnego minimum dla dowolnej postaci sygnału wejściowego.

Opis podstawowej wersji algorytmu LBG został przedstawiony poniżej. Aby rozpocząć budowanie książki należy w pierwszej kolejności zdefiniować wektory danych, ich postać i wymiar. Należy także określić parametry inicjalizacji, o których będzie mowa nieco później.

Algorytm LBG

Algorytm ten jest uogólnioną, wielowymiarową wersją algorytmu nierównomiernej kwantyzacji Lloyda-Maxa. Zasadniczą ideą LBG jest podział wielowymiarowej przestrzeni na regiony decyzyjne, rozłożone wokół wektorów kodu l - tej iteracji oraz przypisanie wszystkich wektorów treningowych do tych regionów. Na podstawie wektorów treningowych przypisanych do poszczególnych regionów następuje wyznaczenie środków ciężkości tych regionów, które stają się nowymi wektorami kodu, tworzącymi wokół siebie nowe regiony decyzyjne. Ten iteracyjny proces modyfikacji regionów i wektorów kodu zostaje zatrzymany w momencie, kiedy kolejne zmiany nie wnoszą już żadnych istotnych korekt do postaci wektorów kodu.

Algorytm 10.1. Metoda LBG budowania książki kodowej w wektorowej kwantyzacji

1. Inicjalizacja:
   - dobór znaczącego zbioru n-wymiarowych wektorów treningowych T określonej postaci;
   - ustalenie początkowej postaci książki kodów $\mathbf{K}^{(1)} = \left\{ \boldsymbol{Y}_j^{(1)} \right\}_{j=1}^{N_\kappa}$, gdzie $N_\mathbf{K}$ oznacza zakładany rozmiar książki kodów;

- zdefiniowanie miary zniekształceń $d(\boldsymbol{X}_i, \widetilde{\boldsymbol{X}}_i)$, która określa błąd przybliżenia kolejnego wektora danych $\boldsymbol{X}_i = (x_{i1}, x_{i2}, ..., x_{in})$ w procesie rekonstrukcji najbliższym (w sensie przyjętej miary) wektorem kodu $\widetilde{\boldsymbol{X}}_i = (\widetilde{x}_{i1}, \widetilde{x}_{i2}, ..., \widetilde{x}_{in}) = \boldsymbol{Y}_j$, stanowiącym element aktualnej książki kodowej; najczęściej jest to błąd średniokwadratowy: $d(\boldsymbol{X}_i, \widetilde{\boldsymbol{X}}_i) = \frac{1}{n}\sum_{k=1}^{n}(x_{ik} - \widetilde{x}_{ik})^2 = \frac{1}{n}\sum_{k=1}^{n}(x_{ik} - y_{jk})^2$, gdzie $\widetilde{\boldsymbol{X}}_i = \boldsymbol{Y}_j$ jeśli $\boldsymbol{X}_i$ należy do regionu przynależności (decyzyjnego) wektora $\boldsymbol{Y}_j$;
- założenie wartości ε określającej progową wartość względnej zmiany średniej wartości zniekształcenia (kwantyzacji);
- ustalenie początkowej średniej wartości zniekształcenia (przybliżenia, kwantyzacji) dla wszystkich wektorów treningowych $D^{(0)} = \infty$ (bardzo duża wartość numeryczna),
- ustawienie licznika iteracji l=1;

2. Określenie regionów przynależności wektorów treningowych oraz średniego poziomu zniekształceń.
   Należy przyporządkować wszystkie wektory treningowe zbioru T do odpowiednich (najbliżej odległych) wektorów kodu $\boldsymbol{Y}_j^{(l)}$ książki $\mathbf{K}^{(l)}$. W tym celu określane są regiony przynależności (decyzyjne) $\{\boldsymbol{R}_j^{(l)}\}_{j=1}^{N_\kappa}$ wokół poszczególnych wektorów kodu z kryterium minimalnego zniekształcenia d: $\boldsymbol{R}_j^{(l)} = \{\boldsymbol{X} : d(\boldsymbol{X}, \boldsymbol{Y}_j^{(l)}) \le d(\boldsymbol{X}, \boldsymbol{Y}_i^{(l)}) \ \forall i \ne j\}$, gdzie $\boldsymbol{Y}_i^{(l)} \in \mathbf{K}^{(l)}$. Następnie obliczana jest wartość średniego poziomu zniekształceń w danej iteracji na zbiorze wszystkich wektorów treningowych $D^{(l)} = \sum_{j=1}^{N_\kappa} \frac{1}{|\boldsymbol{R}_j^{(l)}|} \sum_{\boldsymbol{X} \in \boldsymbol{R}_j^{(l)}} d(\boldsymbol{X}, \boldsymbol{Y}_j^{(l)})$. Jeśli spełniony jest warunek:

$$\frac{D^{(l-1)} - D^{(l)}}{D^{(l-1)}} \le \varepsilon, \tag{10.2}$$

   wówczas następuje zakończenie algorytmu. Osiągnięta została zbieżność do lokalnego minimum z określonym błędem ε. Jeśli natomiast warunek nie jest spełniony, wówczas wykonywany jest następny punkt.

3. Uaktualnienie książki kodowej.
   Każdy wektor kodu $\boldsymbol{Y}_j^{(l)}$ książki $\mathbf{K}^{(l)}$, z przyporządkowanym mu regionem decyzyjnym $\boldsymbol{R}_j^{(l)}$ zastępowany jest przez nowy wektor $\boldsymbol{Y}_j^{(l+1)}$, który minimalizuje błąd kwantyzacji w tym regionie. I tak dla średniokwadratowej miary zniekształceń będzie to środek ciężkości wektorów treningowych z regionu decyzyjnego $\boldsymbol{R}_j^{(l)}$. Ponadto inkrementowany jest licznik iteracji: l=l+1 i następuje skok do punktu 2.

Inicjalizacja książki kodowej

W algorytmie tym znaczącą rolę odgrywa początkowa postać książki kodowej ze względu na zbieżność do lokalnych minimów błędu kwantyzacji. Spośród wielu metod inicjalizacji książki kodowej można wyróżnić trzy zasadnicze, w zależności od rozmiaru początkowego

zbioru wektorów kodu (równy docelowemu rozmiarowi książki kodowej, większy lub mniejszy), takie jak:

- losowa
- grupowania (algorytm PNN - ang. pairwise nearest neighbour)
- rozdzielania (ang. splitting)

Losowa metoda inicjalizacji polega na losowym wybieraniu N_K wektorów ze zbioru wektorów treningowych. Oczywiście takie rozwiązanie jest sensowne tylko wtedy, kiedy nie mamy żadnej wiedzy a piori na temat charakteru danych wektorowych. Wszelką dostępną na wstępie informację trzeba wykorzystać do zawężenia obszaru losowego wyboru, gdyż przypadkowy start od mało korzystnej postaci wektorów kodu może dać w efekcie postać książki kodowej znacznie odległą od optymalnej. Przy takiej bowiem inicjalizacji, kiedy startujemy od rozmiaru książki takiej samej jak rozmiar docelowy, istnieje niewielka możliwość korekty początkowych wektorów - jedynie do najbliższego minimum błędu kwantyzacji.

Drugi sposób przygotowania początkowej postaci wektorów kodu jest bardziej czasochłonny, lecz pozwala uzyskać zazwyczaj lepsze rozwiązanie w skali globalnej. Polega on na tworzeniu coraz liczniejszych grup wektorów danych poczynając od grup jednoelementowych, związanych z każdym wektorem sekwencji treningowej. W kolejnych iteracjach grupowane są najbliższe siebie wektory, czyli liczone są odległości pomiędzy pojedynczymi wektorami oraz środkami ciężkości poszczególnych grup wektorów (które stanowią zbiór wektorów kodu na danym etapie tworzenia księgi) w poszukiwaniu odległości minimalnej. Następnie dwie najbliższe siebie grupy (ew. pojedyncze wektory) są łączone w jedną, tak że sukcesywnie zmniejsza się liczba grup aż do momentu, gdy liczba grup pokrywa się z założonym rozmiarem książki kodowej. Wtedy środki ciężkości tych grup wyznaczają początkową postać wektorów kodu dla algorytmu LBG.

Kolejna metoda szukania optymalnej książki wektorów kodu jest jeszcze bardziej złożona i wymaga wielokrotnego stosowania algorytmu LBG, przy czym wynik jednej optymalizacji LBG staje się, po zastosowaniu odpowiednio dobranej reguły rozdziału wektorów, początkowym zbiorem wektorów kodu kolejnej realizacji LBG ze zwiększoną liczbą elementów książki kodowej. W metodzie rozdzielania, budowę książki kodów rozpoczynamy od jednego wektora, wybieranego najczęściej jako środek ciężkości (centroid) wszystkich wektorów treningowych, który potem jest rozdzielany na dwa wektory według pewnej reguły (np. każda ze składowych wektora zaburzana jest gaussowskim generatorem losowym o zerowej wartości średniej i odchyleniu standardowym estymowanym na podstawie analizy wektorów treningowych). Do otrzymanych w ten sposób dwu wektorów stosuje się algorytm LBG optymalizujący dwuelementową książkę kodową. Wektory te są następnie dzielone według tej samej reguły i optymalizowane LBG, przy czym proces ten jest powtarzany aż do uzyskania założonej wielkości książki kodowej . Jakkolwiek jest to metoda najbardziej czasochłonna ze wspomnianych, to jednak potencjalnie pozwala w większości przypadków uzyskać rozwiązanie najbliższe optymalnej średniokwadratowo książce kodów w skali całego zbioru wektorów treningowych. Nieco zmodyfikowana metoda rozdzielania umożliwia konstrukcję książki kodowej o bardzo użytecznej strukturze drzewa.

Struktura książki kodowej

W celu odnalezienia wektora kodu przybliżającego wektor danych z najmniejszym zniekształceniem konieczne jest przeszukiwanie księgi kodów z kryterium minimalnego

błędu kwantyzacji. Towarzyszą temu procesowi wielokrotne obliczenia błędu dla kolejnych wektorów książki, które są powtarzane dla wszystkich wektorów danych, jakie pojawiają się w strumieniu wejściowym. Czasochłonność i nieefektywność tego procesu, zwłaszcza w przypadku dużej księgi kodów, znacznie ogranicza obszar zastosowań technik VQ. Dlatego też dużej wagi nabiera nie tylko sama zawartość księgi kodów, ale także jej struktura, umożliwiająca możliwie szybkie i skuteczne dobieranie optymalnych wektorów kodu dla analizowanych wektorów danych.

Najprostsza postać książki kodowej, zawierająca nieuporządkowane wektory kodu lub też kolejne wektory według porządku związanego z ich wyznaczaniem (np. w algorytmie LBG), pokryciem przestrzeni według pewnej orientacji lub też zgodnie z innymi przesłankami (np. wiedzą a priori), nie jest optymalna z punktu widzenia szybkości jej przeszukiwań. Konieczność porównania z każdym wektorem w celu wyznaczenia najbardziej podobnego (według przyjętego kryterium) pozwala uzyskać minimalny błąd kwantyzacji dla danego zbioru wektorów, wymaga jednak olbrzymiego nakładu obliczeniowego i czyni takie rozwiązanie w większości przypadków niepraktycznym. Taka organizacja algorytmu wektorowej kwantyzacji nazywana jest metodą pełnego przeszukiwania, a książka kodowa może być nazwana księgą bez struktury.

Aby zwiększyć prędkość przeszukiwania księgi wykorzystywano różne techniki akceleracji obliczeń. Ich zasada działania koncentruje się wokół następującej zasady: Euklidesowe odległości pomiędzy wszystkimi parami wektorów kodu są wstępnie obliczane i zapisywane w tablicy (faza kwantyzacji wstępnej). Mając teraz na wejściu kwantyzatora wektor danych $\boldsymbol{X}_i$, wybierany jest początkowy wektor kodu $\boldsymbol{Y}_j$. Następnie wszystkie wektory kodu, których odległość od $\boldsymbol{Y}_j$ jest większa niż $2\left\|\boldsymbol{X}_i - \boldsymbol{Y}_j\right\|$ są eliminowane z dalszych przeszukiwań, gdyż nie mogą być bliżej wejściowego wektora danych niż $\boldsymbol{Y}_j$. Te wektory kodu, które nie zostały wyeliminowane, są kolejno porównywane z $\boldsymbol{X}_i$ aż do momentu, kiedy znajdzie się wektor bliższy $\boldsymbol{X}_i$ niż $\boldsymbol{Y}_j$. Wtedy zastępuje on wektor kodu $\boldsymbol{Y}_j$ i proces zacieśniania poszukiwań optymalnego wektora kodu jest kontynuowany. Sposób organizacji kwantyzacji wstępnej, algorytmu zawężania zbioru rozwiązań, itd. były przedmiotem optymalizacji w licznych badaniach [5].

Inna klasa rozwiązań koncentruje się na organizacji struktury księgi kodowej. By usprawnić proces przeszukiwania buduje się księgę kodów z wykorzystaniem struktur, które mają ułatwić ten proces możliwie małym kosztem sprzętowym (chodzi o ograniczone wymagania dotyczące dodatkowej pamięci, niezbędnej do przechowywania struktury wektorów księgi kodowej).

Ważnym przyczynkiem do kształtowania struktur księgi kodowej jest analiza statystycznych modeli źródeł informacji w przestrzeni *n*-wymiarowych wektorów. Modele te przybliżają źródła informacji rzeczywistej. Rozważmy wejściowy wektor danych $\boldsymbol{X}$ generowany ze stacjonarnego źródła bez pamięci S_X. Zgodnie z pewną własnością, tj. asymptotyczną własnością podziału AEP (ang. asymptotic equipartition property) – definiowaną np. w [6], jeśli n staje się duży, wówczas n-wymiarowa funkcja gęstości prawdopodobieństwa $p(\boldsymbol{X})$, modelująca to źródło, jest coraz silniej zlokalizowana w pewnym regionie $\boldsymbol{R}_n \in R^n$, wewnątrz którego $p(\boldsymbol{X})$ jest prawie równomierna. Poza tym regionem wektory danych nie są prawie generowane przez źródło. Dla źródeł o rozkładzie Gaussa region $\boldsymbol{R}_n$ jest n-wymiarową sferą, podczas gdy dla źródeł o rozkładzie Laplace'a jest on powłoką n-wymiarowej piramidy. Konsekwencją jest struktura książki kodów, która w

regionie $\boldsymbol{R}_n$ zawiera skoncentrowaną dużą liczbę równomiernie rozłożonych wektorów kodu, bo tam jest duże prawdopodobieństwo pojawienia się kolejnych wektorów danych, podczas gdy pozostałe obszary dziedziny danych pokrywane są zaś bardzo niewielką, jeżeli w ogóle, liczbą wektorów rekonstrukcji.

10.4. Strukturalna kwantyzacja wektorowa

Książka kodowa formowana jest tutaj jako struktura o pewnej regularności, której parametry są dobierane według prostej reguły, na podstawie wiedzy a priori lub też analizy własności zbioru wektorów treningowych. Drzewiasta struktura książki kodowej umożliwia znaczące zmniejszenie czasochłonności algorytmów wektorowej kwantyzacji. Odbywa się to jednak kosztem zwiększenia rozmiaru struktury księgi w stosunku do metod z pełnym przeszukiwaniem. W strukturalnej kwantyzacji wektorowej chodzi także o rozwiązanie problemu zbyt dużego rozmiaru książki kodowej w stosunku do efektywności kwantyzacji. Konstruowane są więc struktury książek kodowych o możliwie małej objętości przy zachowaniu niskich kosztów obliczeniowych. Spośród wielu takich rozwiązań przytoczono kilka przykładowych schematów kwantyzacji, które mają istotne znaczenie z punktu widzenia praktycznych aplikacji. Więcej o różnych schematach strukturalnej kwantyzacji, np. wykorzystujących estymację obszarów jednakowego prawdopodobieństwa narzucających poprzez statystyczne modelowanie przestrzeni wektorów pewną strukturę oszczędnej książki kodowej, a także innych jeszcze koncepcjach usprawnienia algorytmu wektorowej kwantyzacji można przeczytać w [7][8].

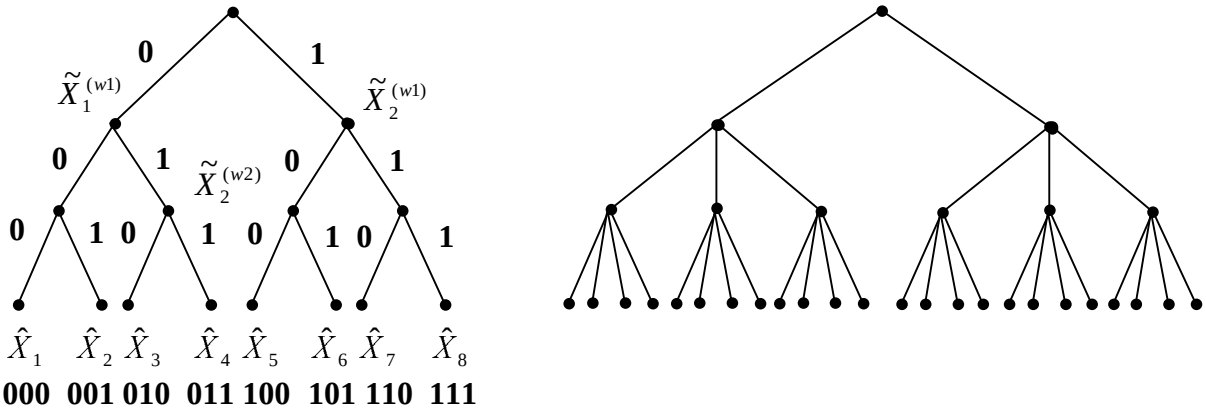
Struktura drzewa

Jedną z najczęściej stosowanych jest struktura drzewa (rys. 10.3), gdzie każdy węzeł zawiera *m* gałęzi, a węzły z kolei zgrupowane są na $p = \log_m N_{\text{K}}$ poziomach drzewa. W poszukiwaniu właściwego wektora przeglądane są tylko niektóre gałęzie drzewa, co zmniejsza liczbę koniecznych porównań z $\text{P}(n \cdot N_{\text{K}})$ (czytaj proporcjonalnie do $n \cdot N_{\text{K}}$), w przypadku przeglądania całej księgi, do $\text{P}(nm \log_m N_{\text{K}})$. Zwiększeniu szybkości przeszukiwań towarzyszą jednak dodatkowe koszty, związane z koniecznością zarezerwowania większej ilości pamięci do przechowywania struktury księgi. Pojemność książki kodów rośnie od wartości $\text{P}(n \cdot N_{\text{K}})$ przy pełnym przeszukiwaniu do wartości $\text{P}(nm(N_{\text{K}} - 1)/(m - 1))$, ze względu na konieczność gromadzenia dodatkowych wektorów w węzłach wewnętrznych drzewa, które reprezentują wektory umieszczone w węzłach potomnych. Te dodatkowe wektory są wykorzystywane do wyznaczenia drogi przejścia przez drzewo od korzenia do wektora kodu dającego najmniejszy błąd kwantyzacji w konkretnym przypadku.

Księga kodów o strukturze drzewa jest tworzona poprzez zastosowanie algorytmu LBG dla każdego węzła drzewa mającego wewnętrzne węzły potomne, w celu wyznaczenia wektorów dla bezpośrednich węzłów potomnych. Generowana jest więc księga kodów o rozmiarze *m* w sposób bardzo zbliżony do metody inicjalizacji poprzez rozdzielanie. Początkowo środek ciężkości wszystkich wektorów treningowych jest rozdzielany na dwa wektory według przyjętej reguły. Wektory te są następnie optymalizowane algorytmem LBG, potem rozdzielane itd. aż do momentu uzyskania *m* wektorów. Są to reprezentanci węzłów wewnętrznych połączonych z korzeniem drzewa. Będą one wykorzystane w pierwszej fazie poszukiwania optymalnego wektora kodu dla wejściowych wektorów danych (zobacz wektory $\tilde{X}_1^{(w1)}$ i $\tilde{X}_2^{(w1)}$ w drzewie binarnym z rys. 10.3a). Ten z nich, który jest najbliższy

wektorowi danych wskazuje na skojarzony węzeł i jego węzły potomne jako na tę część drzewa, która będzie następnie przeszukiwana. Pozostałych *m*-1 węzłów wewnętrznych pierwszego poziomu wraz z podległymi im w strukturze drzewa rozgałęzieniami i węzłami aż do liści zostanie zignorowana.

a) b)



Rys. 10.3. Przykłady struktur drzewa wykorzystywanych przy budowie książki kodowej: a) drzewo ze stałą liczbą gałęzi w węzłach (drzewo binarne) oraz b) drzewo stożkowe z rosnącą liczbą gałęzi w węzłach na kolejnych poziomach drzewa (kolejno dwie, trzy, cztery gałęzie). Na drzewie z lewej strony zaznaczono sposób tworzenia słów kodowych dla wektorów kodu znajdujących się w węzłach zewnętrznych, a nawet wewnętrznych.

Tych *m* wektorów węzłów wewnętrznych pierwszego poziomu drzewa, wyznaczonych z centroidu wszystkich wektorów treningowych, skupia wokół siebie wektory treningowe w *m* regionach decyzyjnych. Następnie proces rozdzielania każdego z wektorów węzłów wewnętrznych wykonywany jest niezależnie na podstawie jedynie tych wektorów treningowych, które znalazły się w regionie decyzyjnym skupionym wokół niego. Proces ten jest kontynuowany aż do momentu zbudowania *p* poziomów drzewa i uzyskania N_{K} węzłów zewnętrznych.

Efektywność kompresji prostego algorytmu VQ, wykorzystującego księgę kodów o strukturze drzewa jest zazwyczaj mniejsza od skuteczności kompresji analogicznej metody z księgą kodów z pełnym przeszukiwaniem, ze względu na mniejszy zbiór, w którym poszukuje się optymalnego wektora kodu po wyborze określonych węzłów wewnętrznych na kolejnych poziomach drzewa. Jednak przedstawione niżej rozwiązania pozwalają zwiększyć efektywność kompresji metod wykorzystujących książkę kodową o strukturze drzewa. Ponadto, taka struktura książki kodowej umożliwia konstrukcję kodera o bardzo użytecznych własnościach tworzenia kodu progresywnego i zagnieżdżonego, zwłaszcza z punktu widzenia zastosowań transmisyjnych. Przykładowo, dla binarnego drzewa z rys. 10.3a) przesyłanie słowa kodowego **011** oznacza odtworzenie czwartego wektora kodu $\tilde{X}_4$ jako przybliżenie oryginalnego wektora danych. Jednak już po pierwszym bicie **0** może zostać odtworzony przez dekompresor wektor $\tilde{X}_1^{(w1)}$ przypisany do węzła wewnętrznego na pierwszym poziomie jako przybliżenie ostatecznej postaci wektora kodu. Potem następny bit **1** pozwala przybliżyć wektor danych poprzez wektor $\tilde{X}_2^{(w2)}$, który jest już zazwyczaj bardziej podobny do wektora $\tilde{X}_4$, by wreszcie po kolejnym bicie **1** uzyskać pełną informację o właściwym wektorze kodu i uzyskać jeszcze dokładniejszą rekonstrukcję oryginalnego wektora danych.

Aby zwiększyć skuteczność struktury drzewa możliwych jest kilka rozwiązań. Obok wspomnianej struktury z równą liczbą gałęzi w węźle stosowane są także struktury drzewa z niejednakową liczbą gałęzi w węźle. W drzewie stożkowym (ang. tapered tree) liczba gałęzi w węzłach drzewa rośnie w miarę przesuwania się w dół drzewa (patrz. rys. 10.3b), co daje nieco większą efektywność algorytmów wykorzystujących tę strukturę kosztem większej złożoności. Z kolei w strukturze okrojonego drzewa (ang. pruned tree) usuwa się z dużego drzewa, utworzonego w momencie inicjalizacji, pewne wektory kodu zaczynając od podstawy tak, aby uzyskać założoną średnią długość kodu przy minimalnym zniekształceniu, lub też najmniejszą średnią długość kodu dla danego zniekształcenia.

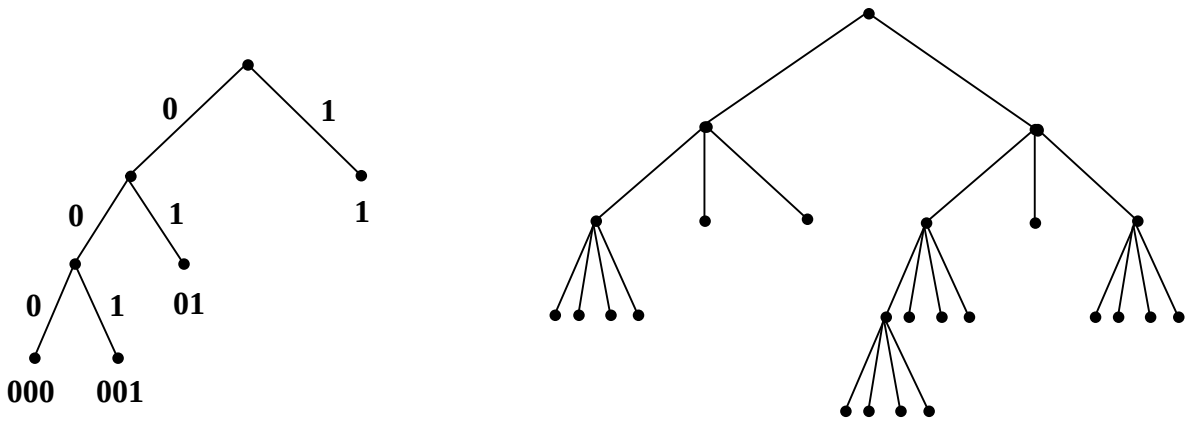
Okrajanie drzewa polega na szukaniu poddrzewa T, które minimalizuje następujące wyrażenie:

$$\lambda(\mathrm{T}) = \frac{\Delta D_{\mathrm{T}}}{\Delta BR_{\mathrm{T}}}, \tag{10.3}$$

gdzie ΔD_{T} oznacza zmianę błędu kwantyzacji po okrojeniu do T, a ΔBR_{T} - zmianę średniej bitowej po okrojeniu do T.

Pomysł polega więc na usuwaniu tych gałęzi, których nieobecność w księdze kodów powoduje powstanie najmniej znaczących zniekształceń rekonstruowanego obrazu. W pierwszej kolejności usuwane są więc gałęzie z węzłami, w których znajdują się bardzo podobne wektory. Reprezentujący te węzły wektor na wyższym poziomie drzewa (skojarzony z węzłem od którego odchodzą wspomniane gałęzie) jest w takim przypadku bardzo do nich podobny, czyli zastąpienie tym wektorem wszystkich wektorów danych trafiających do regionu decyzyjnego rozważanego fragmentu drzewa wprowadzi stosunkowo niewielkie dodatkowe zniekształcenie, a zmniejszenie liczby wektorów kodu da istotny zysk w postaci zmniejszenia bitowego rozmiaru indeksów. Optymalizację stosunku średniej bitowej do wielkości zniekształceń przeprowadza się badając, o ile zwiększy się zniekształcenie kosztem zmniejszenia długości kodu na skutek usunięcia danego fragmentu drzewa. W efekcie powstaje nieregularne drzewo o dłuższych i krótszych gałęziach (rys. 10.4), które generuje zmienną długość kodu dla poszczególnych wektorów kodu.

a) b)



Rys.10.4. Przykłady struktur książek kodowych uzyskane w wyniku algorytmów okrajania początkowej postaci drzewa regularnego: a) ze stałą liczbą gałęzi, b) stożkowego. Słowa kodowe przypisane poszczególnym wektorom kodu - liściom tworzą kod o zmiennej długości. Oczywiście, można w tym przypadku przyjąć inny sposób tworzenia kodu binarnego wektorów.

Inną metodą projektowania księgi kodów jest tworzenie struktury iloczynowej, w której końcowa postać książki jest formowana jako iloczyn kartezjański kilku mniejszych ksiąg kodów. Jeśli osobne wektory danych dotyczą kilku niezależnych cech źródła informacji, np. położenia w przestrzeni trójwymiarowej pewnych obiektów oraz koloru tych obiektów określonego przez wektory RGB, dla każdej z tych cech wyznaczana jest oddzielna księga kodów. Jeśli utworzone niezależnie książki zawierają odpowiednio N_{K1} i N_{K2} wektorów kodu określających położenie i kolor, wtedy efektywny rozmiar księgi iloczynowej wynosi $N_{\text{K1}} \cdot N_{\text{K2}}$, podczas gdy koszty obliczeniowe oraz pamięciowe są proporcjonalne do $N_{\text{K1}} + N_{\text{K2}}$. Ostateczny wektor kodu jest złożeniem wektorów z obu ksiąg opisujących poszczególne cechy.

Księga kodów ze strukturą iloczynową ma potencjalną skuteczność w kompresji wektorów danych zbliżoną do księgi z pełnym przeszukiwaniem, przy takim samym efektywnym rozmiarze. Pozwala jednak uzyskać wyraźnie wyższą jakości rekonstrukcji przy jednakowej złożoności obliczeniowej i średniej bitowej kodu (średniej długości słów kodowych przypisanych poszczególnym wektorom kodu). Wiąże się to z faktem, że przy tych samych czasowych kosztach przeglądania księgi efektywny rozmiar księgi kodów ze strukturą iloczynową jest znacznie większy i może pokryć szerszy obszar *n*-wymiarowej przestrzeni danych przy jednakowym poziomie średniej bitowej wyjściowego strumienia danych.

Przykładem kwantyzatora wykorzystującego książkę kodów ze strukturą iloczynową (ang. product quantizer) jest wektorowy kwantyzator kształtu i wzmocnienia (ang. shape -gain quantizer), który wykorzystuje oddzielne księgi kodowe kształtu i wzmocnienia. Kształt jest definiowany jako oryginalny wektor danych obrazu (np. blok 4×4 dający wymiar wektorów n=16) znormalizowany względem wskaźnika wzmocnienia, takiego jak energia lub wariancja wektora kodu. Książka wzmocnienia jest więc zbiorem dodatnich wartości skalarnych: $\mathbf{K}_w = \{\tilde{w}_i; i = 1,...,N_w\}$, natomiast książka kształtu o jednostkowej normie n-wymiarowych wektorów przyjmuje postać $\mathbf{K}_{\boldsymbol{K}} = \{\tilde{\boldsymbol{K}}_j; j = 1,...,N_{\boldsymbol{K}}\}$. Finalny wektor kodu jest definiowany jako: $\tilde{\boldsymbol{Y}} = \tilde{w}\tilde{\boldsymbol{K}}$. Mając te dwie księgi określanie optymalnego wektora kodu dla kolejnych wektorów danych przebiega w sposób następujący:

- dobierany jest wektor z księgi kształtu $\tilde{\boldsymbol{K}}_j$, dla którego uzyskuje się maksimum korelacji z wejściowym wektorem danych $\boldsymbol{X}$, tj. maksymalną wartość iloczynu skalarnego $\boldsymbol{X}^T \tilde{\boldsymbol{K}}_j$;
- mając dobrany wektor kształtu, wyszukiwany jest taki wektor z księgi wzmocnienia, który minimalizuje wartość wyrażenia $\left|\tilde{w}_i - \boldsymbol{X}^T \tilde{\boldsymbol{K}}_j\right|$.

Powyższy algorytm pozwala określić finalny wektor kodu, który przybliża wektor danych z najmniejszym możliwym błędem średniokwadratowym, bez konieczności normalizowania wejściowego wektora danych (co może być nieraz kosztowne obliczeniowo).

Innym przykładem wykorzystania księgi o strukturze iloczynowej jest piramidowa VQ (ang. pyramid VQ), także mieszcząca się w konwencji kwantyzatora kształtu i wzmocnienia. Wektory księgi kształtu rozmieszczone są tutaj na powierzchni n-wymiarowej piramidy, która jest zbiorem wszystkich wektorów ze składowymi o wartościach dających w sumie 1. Piramidowa VQ jest bardzo dobrze dostosowana do źródeł losowych i.i.d. Laplace'a (opisanych rozkładem Laplace'a) i zawiera najczęściej efektywną metodę indeksowania wektorów kształtu. Odpowiednikiem piramidowej VQ dla źródeł Gaussa (opisanych rozkładem Gaussa) jest sferyczna (ang. spherical) lub biegunowa (ang. polar) VQ. Wektory kształtu rozmieszczane są tutaj według konturów obszarów stałego prawdopodobieństwa,

które przyjmują kształt okręgów na płaszczyźnie, sfer w przestrzeni trójwymiarowej czy hipersfer w przestrzeni o wyższym wymiarze.

Struktura kraty

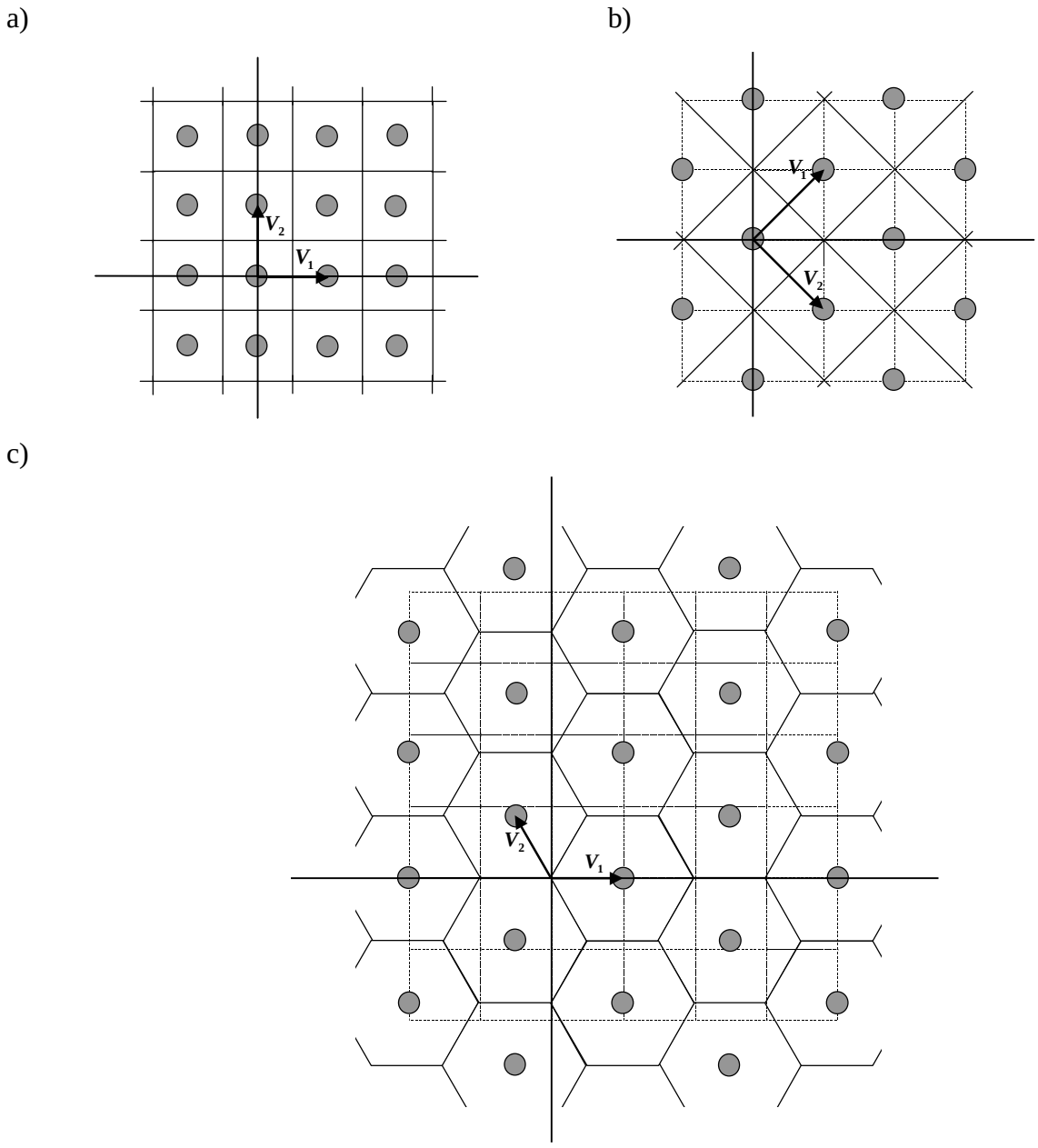
Jest to najprostsza postać wektorowej kwantyzacji, kiedy to można uniknąć czasochłonnych algorytmów budowania książki kodowej i jej optymalizacji. W tym przypadku książka kodowa jest podzbiorem przestrzeni o regularnej strukturze kraty. Kratą w przestrzeni R^n jest zbiór wszystkich wektorów regularnej postaci $\sum_{i=1}^{n} a_i \boldsymbol{V}_i$, gdzie a_i są liczbami całkowitymi, a $\boldsymbol{V}_i$ są liniowo niezależnymi wektorami *n*-wymiarowymi (przypadek nie zdegenerowany). Z kratą nierozerwalnie związany jest kształt regionów decyzyjnych, skupiających wektory danych wokół wektorów kodu ułożonych regularnie w przestrzeni *n*-wymiarowej. Znaczy to, że wybrany na podstawie analizy wektorów danych fragment *n*-wymiarowej przestrzeni pokryty jest przestrzenną siatką o regularnym kształcie oczka - regionu decyzyjnego.

Przykłady regularnych krat wykorzystywanych najczęściej w strukturalnej metodzie wektorowej kwantyzacji na płaszczyźnie przedstawiono na rys. 10.5.

Na płaszczyźnie, przy wektorach bazy $\boldsymbol{V}_1 = (1,0)$ oraz $\boldsymbol{V}_2 = (0,1)$ krata składa się ze wszystkich punktów płaszczyzny o całkowitych współrzędnych. Punkty te stanowią zbiór wektorów kodu, wokół których zostaje rozpięta równomierna siatka kwadratowych regionów decyzyjnych (zobacz rys. 10.5a). Oczywiście kształt tej kraty, zarówno co do rozmiarów oczka (skala kraty), jak i jego kształtu, może być optymalizowany w zależności od charakterystyki kodowanego strumienia danych. Ten rodzaj wektorowej kwantyzacji jest więc rozszerzeniem (uogólnieniem) pojęcia równomiernej kwantyzacji skalarnej na przestrzeń wielowymiarową. Prostokątny nośnik odpowiada dokładnie skalarnej kwantyzacji równomiernej, realizowanej niezależnie na każdej składowej przestrzeni wektorowej.

Inna siatka na płaszczyźnie powstaje dla wektorów bazy $\boldsymbol{V}_1 = (1,1)$ i $\boldsymbol{V}_2 = (1,-1)$ - rys. 10.5b. Wszystkie punkty tej siatki, tworzącej nośnik typu D, mają parzystą sumę współrzędnych. Jeśli natomiast ustalimy wektory bazy jako $\boldsymbol{V}_1 = (1,0)$ i $\boldsymbol{V}_2 = (-\frac{1}{2}, \frac{\sqrt{3}}{2})$, wówczas otrzymamy sześciokątną siatkę nośnika typu A.

Do opisu książki kodowej zbudowanej na równomiernym nośniku potrzeba więc takich parametrów jak: skala, kształt oczka kraty oraz lokalizacja dziedziny kraty w przestrzeni *n*-wymiarowej. Znane są algorytmy (np. najbliższego sąsiada) do szybkiego wyznaczania wektora kodu dla kolejnych wektorów danych [9], a także tworzenia efektywnej reprezentacji bitowej dla poszczególnych wektorów kodu [10]. Efektywność takich ksiąg kodowych z punktu widzenia błędu kwantyzacji w relacji do średniej bitowej i czasochłonności algorytmu kompresji jest stosunkowo wysoka, a nawet niekiedy bliska optymalnej w przypadku kodowania indeksów wyjściowych przy pomocy metod entropijnych. Zaletą w stosunku do równomiernej kwantyzacji skalarnej poszczególnych składowych wektorów jest możliwość ustalenia różnego kształtu nośnika poprzez dobór wektorów bazy rozpinających siatkę.



Rys.10.5. Przykłady regularnych krat stosowanych w wektorowej kwantyzacji na płaszczyźnie: równomierna prostokątna siatka regionów decyzyjnych (a), siatka rombów (b) i sześciokątów (c).

Księga o strukturze kraty pozwala uzyskać skuteczność bliską optymalnej dla wysokorozdzielczej kwantyzacji o zmiennej długości kodowanych indeksów (np. koderem arytmetycznym), a także dla wysokorozdzielczej kwantyzacji o ustalonej średniej bitowej (ang. fixed-rate) źródeł o równomiernym rozkładzie generowanych wektorów danych. Jeśli źródło ma skończony nośnik, kwantyzator o strukturze kraty jest dobierany tak, by mieć ten sam nośnik. Jeśli zaś nośnik jest nieskończony, jak chociażby dla źródeł i.i.d. Gaussa (o rozkładzie Gaussa), wówczas współczynnik skalujący kraty oraz wzór kraty są tak dobierane, aby region nośnika kwantyzatora był pokrywany przez wektory danych z dużym prawdopodobieństwem. Pozostaje oczywiście do rozwiązania problem, jak kwantyzować wektory danych, które leżą poza tym obszarem. Jednym ze sposobów jest przeskalowanie takich wektorów danych, aby znalazły w obszarze nośnika kwantyzatora, a następnie

poszukanie najbliższego wektora kodu. Jednak ta prosta metoda nie zawsze pozwala znaleźć wektor najbliżej leżący wejściowego wektora danych. Nie ma niestety algorytmów o małej złożoności, przy pomocy których można by rozwiązać ten problem. Ponadto, potrzebna jest efektywna technika indeksowania-kodowania wybieranych w trakcie kodowania wektorów kodu.

Adaptacyjna książka kodowa

W przypadku dużych zbiorów danych pochodzących z silnie niestacjonarnych źródeł, korzystnie jest niekiedy zbudować adaptacyjny algorytm modyfikacji książki kodowej w zależności od lokalnej statystyki zbioru. Jednym z możliwych rozwiązań jest wykorzystywanie jedynie podzbiorów globalnej księgi kodów do kompresji poszczególnych fragmentów strumienia wejściowego. Na leży jedynie określić sposób wyboru lokalnej książki kodowej jako podzbioru księgi globalnej, a następnie zapisu tej informacji w strumieniu wyjściowym jako informacji dodatkowej, aby dekoder mógł zdekodować dany fragment zbioru na podstawie właściwego fragmentu dostępnej globalnej książki kodowej.

Konstrukcję adaptacyjnego algorytmu wykorzystującego powyższy algorytm należałoby rozpocząć od budowy dużej książki kodowej, uwzględniającej różnorodną charakterystyka kompresowanego zbioru informacji. Książka ta, zarówno co do treści, jak i struktury winna być znana tak w algorytmie kompresji jak i dekompresji. Sekwencja wektorów danych na wejściu kodera jest przeglądana z pewnym wyprzedzeniem (wprowadzenie pewnego opóźnienia) pod kątem lokalizacji serii wektorów w przestrzeni pokrywanej przez globalną księgę kodu. Na tej podstawie wybierany jest jej fragment, który zostanie wykorzystany do zakodowania przejrzanej partii wektorów. Ponieważ tak określona lokalna książka kodowa ma często znacznie mniejsze rozmiary od księgi globalnej, indeksy lokalizujące poszczególne wektory kodu mogą być wyraźnie mniejsze. Informację o wybranej lokalnie książce należy dodać do strumienia wyjściowego, najlepiej na początku reprezentacji kodowej wspomnianej serii wektorów danych, przy pomocy dodatkowego symbolu z indeksem lokalnego podzakresu księgi kodów. Przykładową realizację takiej książki adaptacyjnej przedstawiono poniżej.

PRZYKŁAD 10.1. Realizacja adaptacyjnej księgi kodowej.

Jeśli globalna księga kodowa zawiera 65536 wektorów, co daje 16-bitowe indeksy tych wektorów, to można przesłać dodatkową daną 16-to bitową, gdzie każdy z bitów definiuje kolejny sektor 4096 wektorów księgi (1-oznacza włączenie sektora do lokalnej książki, 0-oznacza absencję danego sektora). Każdy z wybranych sektorów może być jeszcze bardziej dookreślony przez kolejną sekwencję 16-to bitową. Załóżmy, że przesłano w celu określenia aktualnej lokalnej książki kodów dwie sekwencje 16-to bitowe następującej postaci: 4000H oraz 00C0H poprzedzone symbolem sygnalizującym informację dodatkową w strumieniu (jeden bajt). Oznacza to, że książka zlokalizowana jest w piętnastym sektorze (licząc od najmniej znaczących bitów), w dwu (siódmym i ósmym) jego pod -sektorach, czyli składa się na nią 512 wektorów kodu możliwych do opisania przez 9-cio bitowe indeksy. Jeśli liczebność grupy analizowanych wektorów jest 40, daje to oszczędność długości kodu równą 6 bitów na symbol (indeks), ponieważ zamiast indeksów 16-to bitowych mamy 40 indeksów 9-cio bitowych plus 16+16+8 bitów informacji dodatkowej. Daje to $16+16+8+40\cdot 9$ bitów na 40 indeksów, czyli 10 bitów/symbol. Dla kolejnej partii wektorów danych książka ta może ulec modyfikacji, jeśli zmienia się charakterystyka danych wejściowych. Optymalizacja takiego algorytmu może dotyczyć zarówno wielkości kolejnych sekwencji wektorów danych, dla których testowana jest skuteczność aktualnej książki kodów, jak też sposobu kodowania obszaru globalnej księgi pokrywanej przez książki lokalne.

Możliwe jest także rozwiązanie, kiedy początkowa postać lokalnej książki kodowej jest modyfikowana w przypadku, kiedy pojawi się wektor danych leżący poza obszarem decyzyjnym obejmowanym przez tę książkę (skupionym wokół jej wektorów kodu). Należy wówczas podać pełny indeks właściwego wektora księgi globalnej (lub oszczędniej ale z większym błędem indeks lokalny najbliższego wektora obszaru decyzyjnego aktualnej książki), a następnie przesunąć odpowiednio lokalny obszar decyzyjny modyfikując książkę. Takie rozwiązanie nie wymaga przesyłania dodatkowej informacji do dekodera (przy założeniu zgodnej inicjalizacji i mechanizmu modyfikacji lokalnego obszaru decyzyjnego), jest jednak mało skuteczne dla silnie zmiennych strumieni danych.

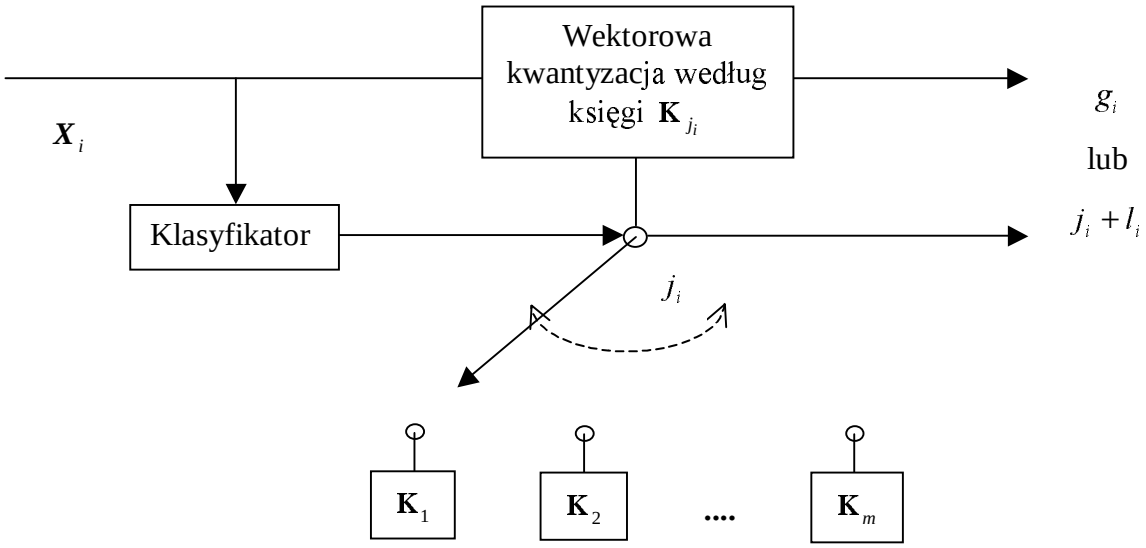
Wykorzystywane są też w praktyce metody z adaptacyjną książką kodów, które bardziej przypominają techniki rozbudowy słownika, czy też modyfikacji schematu predykcji omawianych wcześniej metod. Otóż początkowa postać książki kodowej o określonym rozmiarze jest używana do kodowania kolejnych wektorów danych aż do mementu, kiedy błąd kwantyzacji przekroczył pewien założony próg. Jest to sygnał do zmiany bieżącej postaci książki kodowej. Materiałem na nowy wektor kodu jest ów wektor danych, który właśnie podlega procesowi kwantyzacji. Poprzez np. skalarną kwantyzację każdej jego składowej lub też prostą postać wektorowej kwantyzacji o wektorach rekonstrukcji rozmieszczonych w równomiernej siatce wielowymiarowej dziedziny uzyskujemy *n*-wymiarową skwantowaną postać wektora, który jest dodawany do książki kodowej zarówno kodera jak i dekodera, a więc musi być przesłany pomiędzy kolejnymi indeksami jako informacja dodatkowa. Aby rozmiar książki kodów nie uległ zmianie trzeba jednocześnie usunąć jeden wektor spośród wektorów kodu dotąd stosowanych. Wykorzystane mogą być tutaj różne sposoby, jak chociażby zastąpienie najbliższego nowemu wektorowi w sensie przyjętej miary odległości reprezentanta książki kodowej, usunięcie najrzadziej dotąd stosowanego wektora kodu, itp.

Bardziej złożone schematy kwantyzacji wektorowej z elementami adaptacji zostały przedstawione w następnej części tego rozdziału.

Wektorowa kwantyzacja z klasyfikacją

Jest to koncepcja łącząca pomysł wykorzystania zbioru mniejszych ksiąg kodowych z klasyfikacją wektorów danych, będącą pewną odmianą adaptacji wprzód, zwłaszcza kiedy książki te budowane są jako wstępny etap kompresji zbiorów danych pewnej klasy. Schemat blokowy metody przedstawia rys. 10.6.

W metodzie tej wyznaczanych jest kilka różnych ksiąg kodowych dla poszczególnych partii kodowanego strumienia danych o wyraźnie różnych własnościach (aktywny, nieaktywny, tekstura, krawędź z orientacją, tło, itd.). O wyborze odpowiedniej książki kodowej dla kolejnych wektorów danych decyduje pewna reguła klasyfikacji, rozróżniająca cechy tychże wektorów w znaczeniu kilku zdefiniowanych klas. Informacja o wyborze musi być przesłana do dekodera. Nie musi być to jednak słowo kodowe składające się z numeru książki kodowej i indeksu wybranego wektora w tej książce. Wystarczy jedynie ponumerować razem wszystkie wektory kodu z małych książek, aby wystarczył jedynie sam indeks wektora (pozwoli on dekoderowi sięgnąć do odpowiedniej książki kodowej). Racjonalnym zdaje się wybór nawet kilkunastu małych książek kodowych opisujących wektory danych o określonych dość szczegółowo własnościach, które mogą być znacznie szybciej przeszukiwane niż jedna wielka księga globalna, przy zachowaniu jakości rekonstruowanego sygnału. Poszczególne książki mogą być różnych rozmiarów i każda z nich może używać innej miary zniekształceń przy selekcji najbardziej podobnego wektora.



Rys. 10.6. Schemat blokowy wektorowej kwantyzacji z klasyfikacją. Oznaczenia: $\mathbf{K}_1, \mathbf{K}_2, ..., \mathbf{K}_m$ to małe książki kodowe dla wektorów określonej klasy, j_i - indeks książki kodowej dobranej dla *i*-tego wektora danych na wejściu, g_i -globalny indeks wektora kodu, l_i-lokalny (dla książki o indeksie j_i) indeks wektora kodu. Proces odwrotny polega na odczytywaniu indeksów globalnych lub lokalnych i sięganiu po odpowiednie wektory kodu z książek $\mathbf{K}_1, \mathbf{K}_2, ..., \mathbf{K}_m$.

Wektorowa kwantyzacja z klasyfikacją znajduje często zastosowanie w schematach kompresji wykorzystujących jako wstępny etap predykcji. Na bazie analizy danych oryginalnych konstruowany jest model predykcji, następnie liczone są wartości błędów predykcji, które podlegają wektorowej kwantyzacji. Dane wykorzystywane do predykcji są kwantowane natomiast skalarnie. Przykładem może być prosty algorytm usuwania średniej z każdego wektora danych. Ponieważ często składowa stała w poszczególnych częściach sygnału wejściowego jest bardzo zróżnicowana, po jej odjęciu okazuje się, że znacznie mniej wektorów kodu jest koniecznych do opisu wektorów błędu predykcji. Wartości średnie kolejnych wektorów są kwantowane skalarnie i równomiernie np. do 6-ciu bitów, kodowane i następnie transmitowane do dekodera. Oczywiście aproksymowane z 6 bitowych poziomów kwantyzacji wartości średnie są odejmowane od wektorów w koderze i dodawane do zrekonstruowanych wektorów w dekoderze. Wektory te są rekonstruowane na podstawie wektorów kodu w schemacie kwantyzacji wektorowej z klasyfikacją.

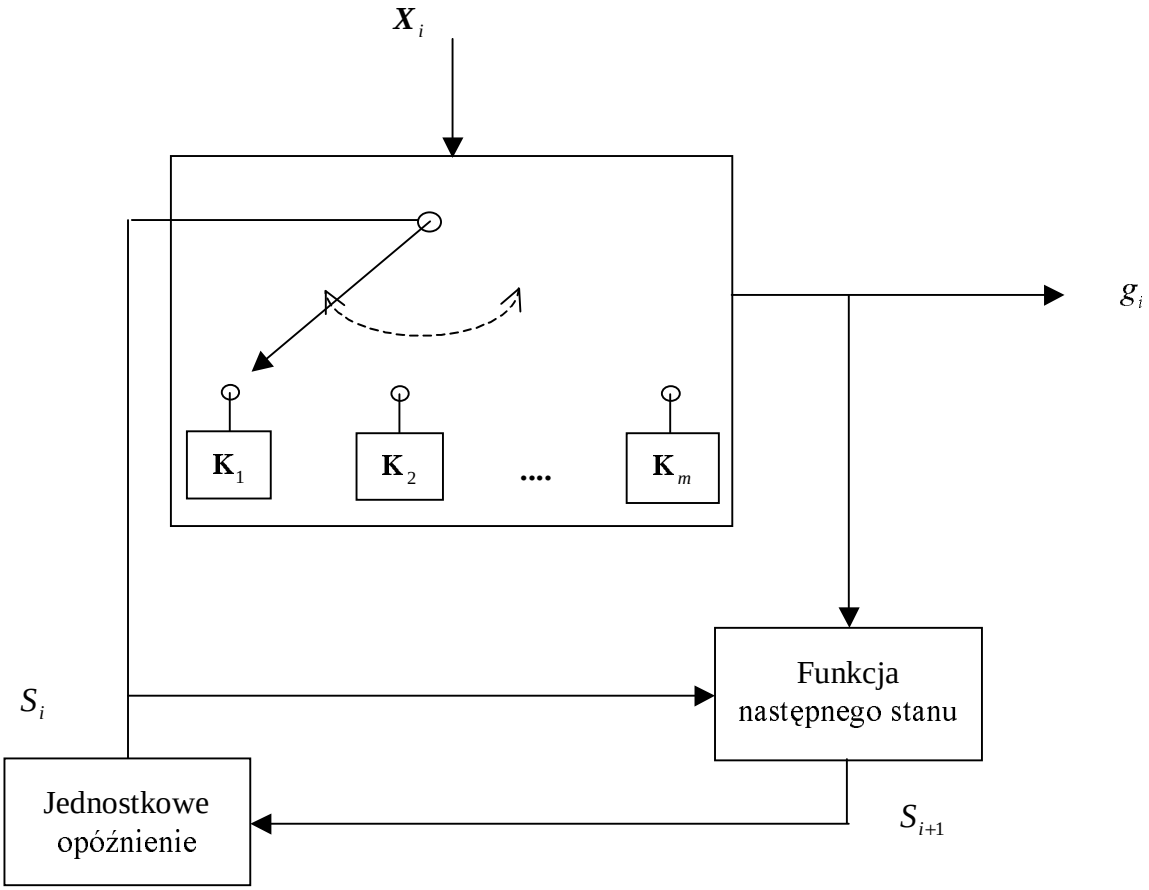
Wektorowa kwantyzacja o skończonej liczbie stanów

Kolejny schemat wektorowej kwantyzacji jest przykładem algorytmu z pamięcią, który może być modelowany jako maszyna o skończonej liczbie stanów (ang. finite state machine). Można przy tym zauważyć nawiązanie do koncepcji adaptacji wstecz. Podobnie jak metoda wykorzystująca klasyfikację posługuje się zbiorem efektywnych małych książek kodowych dopasowanych do konkretnych cech wektorów danych. Pomysł polega na przełączaniu małych książek kodowych w zależności od stanu, w jakim znajduje się koder. Aby wektorowa kwantyzacja w takim schemacie była możliwa, musi zostać zdefiniowana funkcja przejścia od jednego stanu do drugiego. Przejście do następnego stanu i związany z tym wybór właściwej książki kodowej odbywa się na podstawie aktualnego stanu oraz

wyemitowanego właśnie indeksu wektora kodu. Proces ten może być wiernie odtworzony poprzez dekoder bez żadnej dodatkowej informacji.

Odmiana tego schematu znalazła zastosowanie w popularnej ostatnio metodzie kwantyzacji opartej na tzw. kodowaniu kraty (ang. trellis coded quantization), wykorzystywanej w najnowszych algorytmach kompresji przygotowanych w ramach standardu JPEG 2000. Schemat kodera według kwantyzacji wektorowej o skończonej liczbie stanów przedstawiony został na rys. 10.7.

Powyższy schemat kwantyzacji wykorzystując zależności pomiędzy kolejnymi wektorami danych może zwiększyć efektywność kwantyzacji stosunkowo małych wektorów. Z teorii informacji wynika, że dla większych rozmiarów wektorów algorytmy kwantyzacji wektorowej z pamięcią i bez pamięci ją jednakowo skuteczne. Wadą algorytmu z pamięcią jest duża wrażliwość na zakłócenia w transmisji danych, kiedy to nawet niewielkie przekłamanie pojedynczego słowa kodowego spowodować zaburzenie całego procesu dekompresji i bardzo duże błędy rekonstrukcji przesyłanego zbioru danych.



Rys.10.7. Schemat blokowy algorytmu kompresji opartej na wektorowej kwantyzacji o skończonej liczbie stanów. Oznaczenia: to poprzedni i następny stan, w jakim znajduje się kwantyzator z pamięcią, $\mathbf{K}_1, \mathbf{K}_2, ..., \mathbf{K}_m$ to małe książki kodowe dopasowane do konkretnych cech wektorów danych, g_i - globalny indeks wektora kodu. Proces odwrotny powtarza schemat przejść pomiędzy kolejnymi stanami odczytując sukcesywnie indeksy książek kodowych i wystawiając na wyjście odpowiednie wektory kodu.

Inne techniki VQ

Sygnalizując jedynie szeroką gamę innych rozwiązań schematu wektorowej kwantyzacji należy wspomnieć o technikach SVQ i TCQ, łączących w sobie kilka różnych koncepcji kwantyzatorów skalarnych i wektorowych. Piramidowa i sferyczna VQ są szczególnymi przypadkami strukturalnej VQ o ustalonej średniej bitowej, zwanej skalarno-wektorową kwantyzacją (ang. scalar-vector quantization - SVQ). Skalarno-wektorowy kwantyzator jest szczególnym przypadkiem VQ, w którym struktura książki kodowej jest wyprowadzona ze skalarnego kwantyzatora o zmiennej długości kodowanych indeksów [11]. Jest to próba połączenia optymalnego kwantyzatora skalarnego wymuszonego entropią z małej złożoności algorytmem strukturalnej wektorowej kwantyzacji o ustalonej średniej bitowej. Pozwala na nierównomierne rozłożenie wektorów kodu w przestrzeni, co może dać poprawę rekonstrukcji danych rzeczywistych.

Ważnym schematem kwantyzacji w koderach falkowych jest kwantyzacja z kodowaniem kraty (ang. trellis coded quantization - TCQ) [12][13], będąca właściwie wersją kwantyzacji wektorowej (ang. vector quantization VQ) z ustalona strukturą kraty, gdzie określone są przejścia pomiędzy stanami, co daje możliwość minimalizacji długości binarnego kodu wyjściowego. TCQ jest w przybliżeniu odmianą algorytmu VQ z pamięcią, który może być modelowany jako maszyna o skończonej ilości stanów (ang. finite state machine).

W uproszczeniu można stwierdzić, że algorytm TCQ wykorzystuje zbiór efektywnych małych książek kodowych, dopasowanych do konkretnych cech wektorów danych. Pomysł polega na przełączaniu małych książek kodowych w zależności od stanu, w jakim znajduje się koder. Aby wektorowa kwantyzacja w takim schemacie była możliwa, musi zostać zdefiniowana funkcja przejścia z jednego stanu do drugiego. Przejście do następnego stanu i związany z tym wybór właściwej książki kodowej odbywa się na podstawie aktualnego stanu oraz wyemitowanego właśnie indeksu wektora kodu. Proces ten może być wiernie odtworzony poprzez dekoder bez żadnej dodatkowej informacji.

Ogólniej, kwantyzacja z kodowaniem kraty może być interpretowana jako VQ o znaczącej długości bloków danych i księgą kodową tak zorganizowaną, aby realizowała funkcje nieliniowego filtru lub kwantyzatora o skończonej liczbie stanów lub też wektorowego kwantyzatora o mniejszej wymiarowości wektorów. Tworzone są długie słowa kodowe z pomocą struktury kraty (ang. trellis), tj. sukcesywna rekonstrukcja symboli naznacza gałęzie kraty i kodowanie polega de facto na realizacji algorytmu przeszukiwania z minimum zniekształcenia, takiego jak algorytm Viterbiego [14].

Bardziej szczegółowy opis tych ważnych w praktyce kompresji algorytmów można znaleźć w [15][16][17].

Bibliografia:

1. Y. Linde, A. Buzo, R.M. Gray, *An Algorithm for Vector Quantization Design*, IEEE Trans. on Communications, COM-28:84-95, 1980.
2. T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verag, Berlin, 1988.
3. R. Hecht-Nielsen, *Applications of Counterpropagation Networks*, 1:131-139, 1988.
4. K. Rose, E. Gurevitz, G.C. Fox, *Vector Quantization by Deterministic Annealing*, IEEE Trans. on Inform. Theory, 38(4):1249-1257.

5. R. M. Gray, D. L. Neuhoff, *Quantization*, IEEE Trans. Inform. Theory, 44(6):2325-2384, 1998.
6. K. Sayood, *Introduction to data compression*, Morgan Kaufmann Publishers, Inc., San Francisco, str. 243-244, 1996.
7. H. Abut, editor, *Vector Quantization*, IEEE Press, 1990.
8. A. Gersho, and R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Press, 1992.
9. J.H. Conway, and N.J.A. Sloane, *Fast Quantizing and Decoding Algorithms for Lattice Quantizers and Codes*, IEEE Trans. Inform. Theory, IT-28:227-232, 1982.
10. J.H. Conway, and N.J.A. Sloane, *A Fast Encoding Method for Lattice Codes and Quantizers*, IEEE Trans. Inform. Theory, IT-29:820-824, 1983.
11. R. Laroia and N. Farvardin, *A structured fixed-rate vector quantizer derived from a variable-length scalar quantizer. I. Memoryless sources, II Vector sources*, IEEE Trans. Inform. Theory, 39:851--876, May 1993.
12. Z. Xiong, and X. Wu, *Wavelet image coding using trellis coded space-frequency quantization*, IEEE Signal Process. Letters, 6:158-161, July 1999.
13. P. Sriram, and M. W. Marcellin, *Image coding using wavelet transforms and entropy-constrained trellis coded quantization*, IEEE Trans. Image Process., 4:725-733, June 1995.
14. G. D. Forney, Jr., *The Viterbi Algorithm*, Proc. IEEE, 61: 268-278, March 1973.
15. R. Laroia, N. Farvardin, *Trellis-based scalar-vector quantizer for memoryless sources*, ISR Technical Research Report TR 92-82, 1992.
16. M. W. Marcellin and T. R. Fischer, *Trellis coded quantization of memoryless and Gauss-Markov sources*, IEEE Trans. Comm., 38:82-93, Jan. 1990.
17. M. W. Marcellin, *On entropy-constrained trellis-coded quantization*, IEEE Trans. Comm., 42:14-16, Jan. 1994.