CUANTIZARE VECTORIALA

Unul din rezultatele fundamentale ale cercetarilor lui Shannon privind relatia debit-distorsiune demonstreaza ca se obtin performante optime codând vectori în loc de scalari, chiar si în cazul în care sursa vectoriala are k componente statistic independente, rezultat care justifica superioritatea cuantizarii vectoriale. Vom trata cuantizarea vectoriala, tratarea fiind identica (caz particular) pentru cuantizarea scalara.

Cuantizarea vectoriala (CV) în sensul sau cel mai general este aproximarea unui semnal de amplitudine continua printr-un semnal de amplitudine discreta. În cazul aplicatiei noastre CV consta în a reprezenta un vector x de dimensiune k printr-un vector y de aceiasi dimensiune apartinând unui set finit numit ***dictionar***.

Din punct de vedere matematic, putem defini cuantizorul vectorial în urmatoarea maniera:

Fie I multimea indicilor corespunzatori dictionarului Y si y_i, i<I setul de vectori din Y

$$I = \{ 1, 2, \ldots , N \} \quad (1)$$

Procesul de codare este definit prin:

$$C : R \rightarrow I \quad (2)$$
$$x \rightarrow C(x) \quad (3)$$

iar procesul de decodare prin:

$$D : I \rightarrow Y \quad (4)$$
$$i \rightarrow y_i \quad (5)$$

În operatia de comprimare anumite informatii se pierd si semnalul original nu poate fi refacut. De aceea cuantizarea vectoriala este o operatie **ireversibila**.

Un cuantizor vectorial (CV) se descompune de obicei în doua componente: un codor si un decodor. Vom folosi abrevierea CV atât pentru "Cuantizare Vectoriala" cât si pentru "Cuantizor Vectorial" urmând a rezulta din context despre care din notiuni este vorba.

Rolul **cuantizorului** consta în a cauta în dictionar pentru toti vectorii x ai semnalului de intrare (bloc al imaginii) vectorul cel mai "apropiat" de vectorul sursa x. Pentru aprecierea apropierii (distantei) vom folosi în acest caz distanta euclidiana între doi vectori. Adresa vectorului y ales este cea care se va transmite în locul vectorului x. Acesta este momentul compresiei propriu-zise. În cazul simplu al unui dictionar implementat sub forma de tablou adresa este de fapt tocmai indicele vectorului ales în acel tablou.

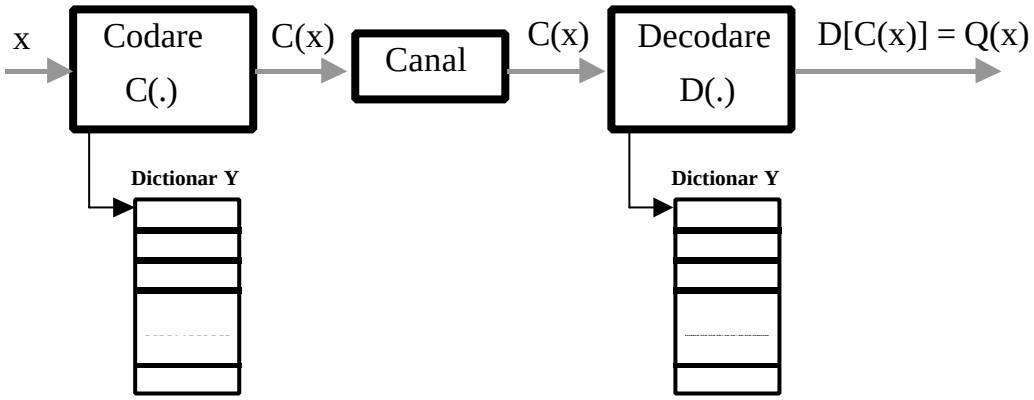

Figura 1 Schema generala a unui cuantizor vectorial

Decodorul dispune de o copie a dictionarului pe care îl consulta pentru a obtine vectorul corespunzator adresei receptionate. În acest fel se realizeaza operatia de decompresie. Daca dimensiunea vectorului este mare tabloul se înlocuieste cu un arbore binar în care se face cautarea bit cu bit (pe masura ce bitii adresei se receptioneaza).

Schema generala a unui cuantizor vectorial poate fi reprezentat ca în Figura 1. Din analiza schemei CV rezulta rolul esential al dictionarului utilizat în operatiile de codare si decodare.

Un dictionar, independent de modul sau de constructie si de modul de cautare ales, este definit de doi parametrii principali: numarul de vectori din cadrul dictionarului si de dimensiunea spatiului vectorial (dimensiunea vectorilor din cadrul dictionarului). Cei doi parametrii apar strâns legati si este utila alegerea celor mai bune valori pentru acesti parametrii, alegere care va fi posibila prin construirea unui dictionar optim. De obicei se lucreaza în cadrul unui spatiu transformat iar spatiul nostru vectorial va fi un subspatiu al lui R_k cu k = {1, 4, 16}. Vectorii din acest spatiu reprezinta blocuri ale imaginii în domeniul transformat.

Înainte de a analiza care este cea mai buna valoare a acestor parametrii este util de a fixa ordinul lor de marime:

- Dimensiunea vectorilor dictionarului este fixata de obicei între 1 si 16. Fixând o dimensiune mai mare de 16 efectul de bloc devine important si dictionarul este puternic dependent de imaginea din care el a fost construit. La limita ajungem a considera un vector având dimensiunea egala cu cea a secventei de antrenament.
- Numarul vectorilor dictionarului este limitat de doua constrângeri distincte:

$\Rightarrow$ Debitul unui CV este egal cu $\log_2 N / k$ biti/pixel unde N este numarul de vectori ai dictionarului iar k este dimensiunea acestor vectori. Pentru ca cuantizorul sa prezinte interes debitul trebuie sa fie inferior debitului semnalului original (în practica se fixeaza un debit maxim mult inferior debitului semnalului original). De asemenea, daca facem o cuantizare scalara (k=1)

aplicând formula anterioara si considerând ca si debit initial 8 biti/pixel, numarul maxim de vectori ai dictionarului este 256. Daca însa folosim blocuri de dimensiune 16 numarul maxim de vectori este $2^{16*8} = 2^{128.}$

$\Rightarrow$ A doua problema este aceea a spatiului ocupat de dictionar, fiind de neconceput ca dictionarele sa contina mai multe blocuri decât imaginea originala. În cazul unei imagini de 512 x 512 numarul reprezentantilor din dictionar trebuie sa fie (mult) sub 16384 în cazul blocurilor de dimensiune 16.

Pentru zone ale imaginii care necesita o calitate ridicata la refacere sau care contin o energie mare (necesitând astfel un debit important) se impune utilizarea cuantizarii scalare sau a unui numar de blocuri de dimensiune mai mica pentru a nu construi un dictionar de dimensiuni exagerate.

Pentru restul de zone ale imaginii se pot utiliza blocuri de dimensiune mai mare fara a determina o crestere exagerata a numarului de vectori din dictionar utilizarea acestora, pentru un debit dat, facând CV superioara cuantizarii scalare.

1. Constructia dictionarului în cuantizarea vectoriala

Constructia unui CV performant în toate cazurile este o problema dificila fiind legata de imposibilitatea gasirii unui dictionar universal. În continuare se va trata constructia dictionarului stiind ca, în mod esential, calitatea unui CV depinde de algoritmul folosit pentru constructia acestuia.

Elaborarea dictionarului se face pornind de la o secventa de antrenament. Vectorii ce apartin acestei secvente corespund blocurilor imaginii (sau coeficientilor în cazul în care se lucreaza într-un spatiu transformat).

1.1. Algoritmul LBG

Acest algoritm propus de Linde, Buzo si Gray corespunde unei extensii a algoritmului Lloyd-Max utilizat pentru elaborarea dictionarului în cazul cuantizarii scalare. Rolul sau este, pentru un dictionar initial dat, de a optimiza codorul si decodorul.

pasul 0: Initializarea

- se fixeaza

  un dictionar initial $Y^{(0)}$ de dimensiune N

  un prag $\delta > 0$

  un numar maxim de iteratii p

o sursa de vectori x de distributie cunoscuta

- se initializeaza

  $D^{(-1)} = \infty$

  $m = 0$

pasul 1:

- pentru dictionarul curent $Y^{(m)} = \{ y_i^{(m)} \}_{\{i = 1,..., N\}}$, se determina partitia optimala

  $R^{(m)} = \{ R_i^{(m)} \}_{\{i = 1,..., N\}}$ care minimizeaza distorsiunea medie

  $$x \in R_i^{(m)} \quad \text{daca} \quad d(x, y_i^{(m)}) < d(x, y_j^{(m)}), \ \forall j \mathrel{!}= i \tag{6}$$

- se calculeaza distorsiunea medie

  $$D^{(m)} = \sum_{i=1}^{N} E [d(x, y_i^m) \mid x \in R_i^{(m)}] \tag{7}$$

pasul 2:

- daca $\frac{D^{(m-1)} - D^{(m)}}{D^{(m)}} \leq \delta$ sau $m = p$ (8)

  oprire algoritm: CV este descris de $R^{(m)}$ si $Y^{(m)}$

- daca nu continua

pasul 3:

- pentru partitia $R^{(m)}$, se calculeaza dictionarul optimal

  $$Y^{(m+1)} = y_i^{(m+1)}{}_{\{i=1,..,N\}} \quad \text{unde} \quad y_i^{(m+1)} = \text{cent}(R_i^{(m)}) \tag{9}$$

- se incrementeaza m
- se revine la pasul 1

Abrevierea cent utilizata în algoritm corespunde centroidului. Centroidul unei regiuni R_i este definit prin

$$y_i = \text{cent}(R_i) = E[x \mid x \in R_i] \tag{10}$$

Acest algoritm de optimizare iterativa porneste de la un dictionar initial. Problema principala este alegerea acestui dictionar initial deoarece experienta a dovedit ca el contribuie esential la performantele algoritmului LBG. Fiecare iteratie produce o schimbare locala a dictionarului iar algoritmul converge la minimul local cel mai apropiat de dictionarul initial. Numarul maxim de iteratii a fost introdus în scopul reducerii timpului de obtinere a dictionarului. Limita numarului maxim de iteratii (de obicei 20) se atinge doar în cazuri rare si nu introduce o distorsiune semnificativa. Problema care ramâne este aceea a construirii dictionarului initial.

1.2 Alegerea dictionarului initial. Metode de dihotomie vectoriala

Cel mai simplu dictionar initial este cel care contine primii N vectori ai secventei de antrenament sau N vectori alesi în mod aleator din secventa de antrenament. Vectorii pot fi bineînteles nereprezentativi si rezultatele sunt mediocre. O alta solutie ar fi sa fixam o distanta minima între vectorii dictionarului initial. Aceasta metoda permite obtinerea unei reprezentativitati mai bune ca în cazurile anterioare dar este în continuare nesatisfacatoare, determinarea pragului fiind dificil de realizat deoarece depinde de complexitatea secventei de antrenament.

Solutia consacrata are la baza ***dihotomia vectoriala*** si a fost propusa în versiunea initiala a algoritmului LBG sub numele de metoda de "splitting". Se impune un debit R = $\log_2(N)$ întreg. În acest caz este generata o succesiune de CV având debitele crescând de la 0 la R. Procedeul folosit pentru constructie este urmatorul:

pasul 0: initializarea

- initializarea dictionarului Y considerând Y[0] = y_0 unde y_0 este centroida secventei de antrenament luata în ansamblul ei.
- se noteaza u vectorul din R_k având toate componentele sale egale cu 1.
- se fixeaza o perturbatie scalara ε.
- se fixeaza un debit R.
- se fixeaza un prag al distorsiunii D.
- se atribuie r = 0.

pasul 1:

- se perturba dictionarul curent $Y = \{y_i\}_{\{i=0,..,2r-1\}}$ astfel

$$\begin{cases} Y[2i+1] = Y[i] - \varepsilon u \\ Y[2i+0] = Y[i] + \varepsilon u \end{cases} \qquad \forall i = 0,\ldots,2^r - 1 \tag{11}$$

- se incrementeaza r

pasul 2:

- se considera noul dictionar $Y = \{y_i\}_{\{i=0,...,2r-1\}}$
- se executa algoritmul LBG asupra dictionarului Y
- daca r / k = R sau $D^{(r)} < D$

  oprire si retinere $Y^{(r)}$

  daca nu

  întoarcere la pasul 1

O organigrama simpla a algoritmului este data în Figura 2. Un exemplu de executie este dat în Figura 3.

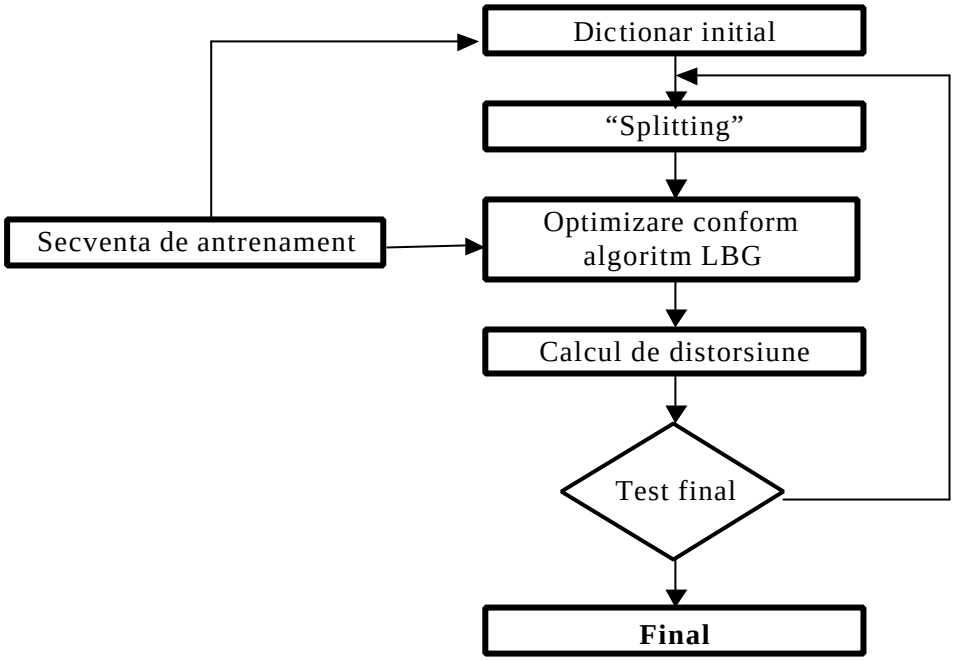

Figura 2 Organigrama algoritmului LBG

Calculele necesare constructiei dictionarului sunt semnificative dar nu sunt considerate un dezavantaj deosebit deoarece un dictionar este construit o singura data. Deoarece cautarea exhaustiva în cadrul dictionarului pentru determinarea reprezentantului optim este mare consumatoare de timp de calcul este necesar a se lua în considerare noi tehnici care sa permita obtinerea rezultatelor cu o precizie acceptabila cu o complexitate a calculelor la codare rezonabila. Acest compromis este obtinut prin utilizarea metodelor arborescente.

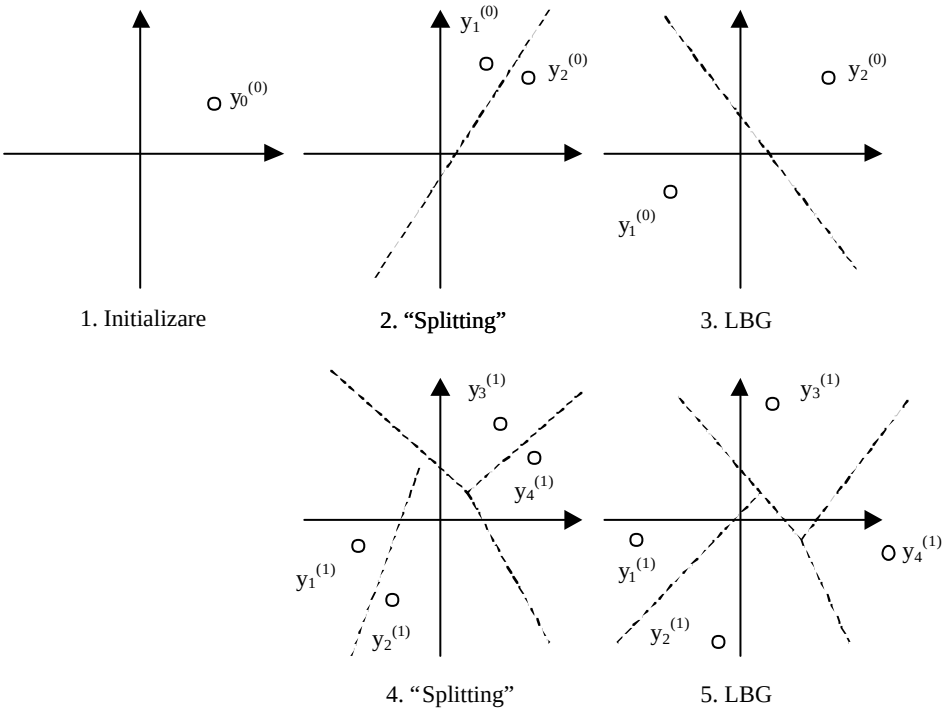

Figura 3 Exemplu de executie a algoritmului LBG

2. Cuantizare vectoriala cu cautare arborescenta

Odata dictionarul construit trebuie gasita o metoda de cautare rapida. Cautarea exhaustiva fiind foarte costisitoare, d.p.d.v. al timpului, metodele de cautare arborescenta sunt foarte des folosite. Trebuie semnalat ca, desi aceste metode nu dau obligatoriu cel mai apropiat reprezentant ci un reprezentant apropiat, ele permit reducerea considerabila a timpului de codare si apar ca un bun compromis între precizia obtinuta la refacerea imaginii si viteza de codare a acesteia.

Se disting 2 mari metode de CV cu cautare arborescenta (notate în continuare **CVca**):

- prima posibilitate consta în integrarea constructiei arborelui în constructia dictionarului.
- a doua posibilitate este de a construi arborele atunci când dictionarul este deja constituit. Se porneste de la vectorii obtinuti prin algoritmul LBG (frunze în arbore) care nu prezinta nici o structura apriori.

Codorul, pentru a efectua cautarea, dispune de arbore si porneste cautarea începând cu radacina acestuia. În fiecare etapa, se calculeaza distanta fata de cei doi fii ai nodului curent si se alege cel care contribuie cu o distorsiune cât mai mica. Procesul continua considerând subarborele având acel nod ca radacina pâna când se atinge un nod terminal al arborelui (o frunza). Vectorul asociat acelui nod terminal este considerat ca si reprezentant al blocului sursa.

Decodorul nu dispune în general de ansamblul nodurilor intermediare ale arborelui ci numai de nodurile finale deoarece codorul îi transmite direct indicele vectorului care reprezinta vectorul sursa. Totusi în cazul particular al reconstructiei progresive decodorul utilizeaza noduri intermediare. Transmisia se face de catre codor la fiecare nivel în arbore iar pe masura ce acesta înainteaza în cautarea sa vectorul reprezentativ al vectorului sursa devine din ce în ce mai precis si este interpretat corespunzator de decodor.

2.1. CVca cu arbori obtinuti ulterior constructiei dictionarului

Constructia arborelui se face dupa ce dictionarul este constituit. Dictionarul este construit prin metode clasice (de exemplu algoritmul LBG) si nu prezinta a priori nici o structura particulara. Realizarea unei clasificari ierarhice ascendente consta în a forma, pornind de la clase mici foarte omogene, clase din ce în ce mai putin omogene pâna la obtinerea unei clase unice. Pentru aceasta definim 2 functii:

- o functie de distanta d() care determina distanta între doi vectori
- o functie de disimilaritate $\delta(\ ,\ )$ care masoara disimilaritatea între doua clase de vectori

Distanta d este cea utilizata si pentru constructia dictionarului. Pentru strategia de fuziune se pot utiliza mai multe functii de disimilaritate, doua dintre acestea fiind date în continuare:

- masura distantei minime

$$\delta_1(Ri, Rj) = \min_{x_i \in R_i, x_j \in R_j} d(x_i, x_j) \tag{12}$$

- masura distantei medii

$$\delta_2(R_i, R_j) = \frac{1}{L_i L_j} \sum_{x_i \in R_i, x_j \in R_j} d(x_i, x_j) \tag{13}$$

unde L_i = card[R_i] si L_j = card[R_j]

Algoritmul utilizat pentru construirea arborelui este algoritmul **Clasificarii Ierarhice Ascendente** si este descris în continuare.

pasul 0: initializarea

- pornind de la N clase disjuncte R_i se calculeaza $\delta(R_i,R_j)$

pasul 1:

- se determina în ierarhia curenta perechea cea mai similara (R_i,R_j) în sensul disimilaritatii $\delta(\ ,\ )$.
- se calculeaza centroida pentru $R_i \cup R_j$.

pasul 2:

- se actualizeaza distantele $\delta(R_k, R_i \cup R_j)$, $\forall k \neq i, k \neq j$

pasul 3:

- daca a ramas o singura clasa algoritmul se termina
- daca sunt mai multe clase se reia de la pasul 1

2.2. CVca cu arbori echilibrati obtinuti simultan constructiei dictionarului

În acest caz nodurile terminale sunt situate pe acelasi nivel si adâncimea este data de relatia

$$H = \log_2(N) \tag{14}$$

La fiecare nivel de cautare în arbore se calculeaza distorsiunea asociata reprezentarii vectorului sursa prin fiecare din fii nodului curent (în cazul nostru se utilizeaza arbori binari deci numarul fiilor este 2). Complexitatea algoritmului de cautare este 2H în loc de 2^H corespunzatoare unei cautari exhaustive rezultând un câstig de $\frac{2^H}{2H} = 2^{H-1}H^{-1}$.

Pentru a aprecia ordinul de marime sa consideram un dictionar de adâncime 10 ceea ce este echivalent cu un dictionar de 1024 elemente:

- numarul calculelor de distorsiune în cazul cautarii exhaustive este de 1024
- numarul calculelor de distorsiune în cazul unei cautari arborescente este $2 \times 10 = 20$
- complexitatea scade în acest caz de $1024/20 = 51.2$ ori

Dimensiunea dictionarului este marita, în acest caz trebuind stocati atât vectorii cât si întreaga arborescenta. Numarul nodurilor neterminale este egal cu 2^H-1, acestea adaugându-se celor terminale care sunt 2^H. Numarul de noduri care trebuie memorate în acest caz este $2^H-1+2^H = 2^{H+1}-1$ dupa cum se observa dublu fata de cazul cautarii exhaustive. La dimensiunea actuala ale memoriei acest surplus nu constituie un handicap semnificativ.

Constructia dictionarului respecta principiul **dihotomiei vectoriale** prezentat anterior (aplicarea *sistematica* a metodei de "splitting"). Când lucram la o adâncime h fiecare nod va fi divizat în 2 fii care constituie arborele de adâncime h + 1.

CVca obtinut folosind arborescente echilibrate este sub-optimal în raport cu CV exhaustive din urmatoarele doua motive:

sub-optimalitatea dictionarului - acesta contine mai putini reprezentanti adecvati decât în cazul metodei exhaustive datorita constrângerilor din momentul constructiei.

sub-optimalitatea codarii - datorita modului de cautare în arbore vectorul obtinut nu este neaparat cel mai apropiat de vectorul sursa.

2.3. CVca cu arbori neechilibrati obtinuti simultan constructiei dictionarului

Posibilitatea de a avea un debit variabil (arbori dezechilibrati) prezinta avantajul de a distribui eficient bitii. Codorul poate atribui mai multi biti regiunilor "active" si poate "economisi" un numar semnificativ de biti la codarea regiunilor omogene.

În cazul constructiei unui CVca, pentru a avea un debit variabil este suficient de a construi un arbore neechilibrat, adica nodurile terminale nu se afla toate la aceeasi adâncime în arbore. Lungimea cuvântului de cod asociat fiecarei frunze este egala cu adâncimea ei în cadrul arborelui.

Tehnicile de constructie a arborescentelor descendente neechilibrate se pot împarti în doua mari categorii:

- Algoritmi prin divizare - constructia arborelui neechilibrat se face odata cu constructia dictionarului prin aplicarea *nesistematica* a metodei de "splitting".
- Algoritmi prin taierea ramurilor ("pruning") - Se construieste întâi un arbore echilibrat prin una din metodele clasice iar acestui arbore i se taie anumite ramuri pentru a asigura un compromis mai bun între distorsiune si debit.

Pentru constructia arborelui se foloseste o secventa de antrenament. În cadrul codarii secventei, presupusa de lungime L, definim R_n ca si "celula" asociata nodului n, adica R_n desemneaza ansamblul vectorilor din secventa de antrenament care, prin codare, apartin nodului n. Definim de asemenea:

- probabilitatea p(n) a nodului n ca si

$$p(n) = \frac{card[R_n]}{L} \tag{15}$$

- distorsiunea $\Delta(n)$ asociata unui nod ca si

$$\Delta(n) = \frac{1}{card[R_n]} \sum_{x \in R_n} \left\| x - y_n \right\|^2 \tag{16}$$

unde y_n este centroida celulei R_n.

2.3.1. Algoritmi prin divizare

Spre deosebire de cazul constructiei arborilor echilibrati, în loc de a diviza fiecare nod de la adâncimea h în cei doi fii ai sai, divizam numai nodul care prezinta cel mai mare interes din punct de vedere al unui criteriu care va fi definit în continuare. Acest nod nu prezinta nici o caracteristica particulara singura restrictie fiind aceea ca la acel moment el este un nod terminal.

Pot fi definite mai multe criterii de a alege nodul de maxim interes:

- criteriul cel mai natural este acela de a alege nodul care prezinta cea mai mare scadere a distorsiunii.
- un alt criteriu convenabil din punct de vedere logic este de a diviza nodul care contribuie cel mai mult la distorsiunea globala.
- un al treilea criteriu consta în a alege nodul care asigura cel mai bun compromis "scadere a distorsiunii / crestere a debitului".

Pentru nodul terminal care este "împartit" în n_1 si n_2 calculam

- variatia (scaderea) distorsiunii egala cu:

$$p(n)\Delta(n) - (p(n_1)\Delta(n_1) + p(n_2)\Delta(n_2)) \tag{17}$$

- variatia (cresterea) debitului egala cu:

$$p(n)h(n) - (p(n_1)h(n_1) + p(n_2)h(n_2)) \tag{18}$$

Aceste doua variatii sunt antagoniste astfel încât trebuie gasit un compromis între ele. În acest scop masuram raportul între scaderea distorsiunii si cresterea debitului rezultate prin "divizarea" nodului n, raport pe care îl numim "***raport marginal***" si îl notam $\lambda(n)$. Prin definitie $\lambda(n)$ este dat de

$$\lambda(n) = -\frac{p(n)\Delta(n) - (p(n_1)\Delta(n_1) + p(n_2)\Delta(n_2))}{p(n)h(n) - (p(n_1)h(n_1) + p(n_2)h(n_2))} \quad (19)$$

Frunza (nodul terminal) care determina **cea mai mare valoare** pentru "raportul marginal" $\lambda(n)$ este cea care va fi "**divizata**".

2.3.2. Algoritm prin taierea optima a ramurilor

Algoritmul este cunoscut ca **BFOS** dupa numele autorilor sai Breiman, Friedman, Olshen si Stone. Prima etapa, înainte de aplicarea propriu-zisa a algoritmului, consta în construirea unui dictionar de dimensiune importanta, dictionar (arborescent) caruia apoi i se vor taia ramuri. Scopul algoritmului este de a suprima parti ale arborelui care nu au o contributie importanta în scaderea distorsiunii raportat la contributia lor în cresterea ratei. Acesta revine la suprimarea decuparilor zonelor omogene realizata în cadrul constructiei arborelui. Ca si algoritmul de divizaree si algoritmul BFOS se bazeaza pe "raport marginal".

Principiul algoritmului **BFOS** este relativ simplu: pentru fiecare nod n neterminal al arborelui se calculeaza "raportul marginal" $\lambda(n)$. **Taiem** apoi **ramurile** începând din nodul n_i având **cea mai mica valoare** a "raport marginal" $\lambda(n)$ adica înlocuim în arbore subramura Sn_i provenind din nodul n_i cu nodul n_i. Actualizam apoi raportul $\lambda(n)$ asociat tuturor ascendentilor lui n_i (pentru alte noduri din arbore acest raport ramâne neschimbat) si reiteram aceasta procedura pâna obtinem un debit dat sau o distorsiune maxima fixata.

3. Alegerea spatiului de reprezentare

O imagine este de obicei reprezentata sub forma unei matrici de pixeli, spatiul de reprezentare fiind cel al luminantei (pentru imaginile color trebuie considerata si crominanta). În acest spatiu sunt multe redundante statistice luminanta unui pixel fiind puternic corelata de cea a pixelilor vecini. De aceea aplicarea CV direct pe spatiul luminantei (spatiul primar) nu da rezultate la fel de bune ca aplicarea CV într-un spatiu transformat – de exemplu al Transformatei Cosinus Discreta (DCT) sau al componentelor principale.

Atunci când lucram în domeniu frecventa putem organiza matricea transformata în doua moduri distincte în asa fel încât sa rezulte blocuri de date de natura diferita. Astfel putem descompune imaginea în sub-benzi si grupa coeficientii unei sub-benzi de frecventa si sa realizam CV pe fiecare din aceste sub-benzi (cuantizare **"intra-banda"**). Alternativa este de a pastra corespondenta spatiala între blocurile imaginii originale si blocurile spatiului transformat (cuantizare "**inter-banda**").