Predavanje III

Tags	Done

Srenja duzina koda je mera performanse koda, odnos izmedju ukupne duzine kodiranog teksta i samog broja simbola. Cilj je da ukupna duzina poruke bude sto manja i samim tim srenja duzina koda meri te performanse.

Donja granica je entropija i ona je za datu slucajnu promenljivu koja predstavlja izvor simbola koje kodiramo, entopija te slucajne promeljive je srednja mera informativnosti.

Da li postoje algoritmi koji odredjuju kod koji ima optimalnu srednju duzinu kodne reci.

Optimalni kodovi

Ako je V optimalni kod i pi je verovatnoca pojavljivanja i-tog simbola, a ni odgovarajuca duzina kodne reci, sto je simbol verovatniji to je kodna rec manja.

Verovatniji simbolima je kodirana manja kodna rec.

Potreban uslov optimalnosti za bilo koju bazu koda.

```
Pretpostavljamo da postoje Vi i Vj za koje je p_i > p_j
i n_i > n_j. Verovatniji simbol kodiran duzom kodnom reci.
n_v = p_i n_i + ... p_j n_j = min

ako vazi uslov p_i > p_j i n_i > n_j, zelimo da dokazemo
n_v mozemo da smanjimo i dobijamo kontradikciju da je v optim

n_v' = p_j n_j + ... + p_i n_i =>
n_v - n_v' = (p_i - p_j) + (n_i - n_j) > 0
n_v < n_v' -> dolazimo do kontadikcije
```

Pretpostavljamo da je kod binarni. Ukoliko neki kod koji je jednoznacno dekodabilan zadovoljava kraftovu jednakost, znaci da postoji prefiksni kod sa tim sitim duzinama kodnih reci tako da dovoljno je da posmatramo prefiksne kodove.

Koristimo cinjenicu da je kod prefiksni (pretpostavka).

Ako je V prefiksni kod i ako su verovatnoce pojavljivanja sortiranje tada su dve poslednje reci iste duzine.

```
V je optimalni kod.
- - - -
- - - - - -
Trebamo da dokazemo da su ove dve kodne reci iste duzine.
Ako je v_a > v_a-1. Potrebno je da skratimo v_a, tada uvodimo
koji je prefik v_a i jednak je duzini v_a-1.
Posto smo promenili v_a', ona se pojavlje samo jednom i
nije jednaka sa ni sa jednim delovim v_i, ako bi bila, znaci
je prefiks nekog dela kodne reci, a samim tim ni ona nije pre

Osobina je da ima manju srednju vrednost duzine kodne reci.

n_v = p_i[a_v]
n_v' = p_i[a_v'] < n_v

dolazimo do kontradikcije da je v optimalne duzine, samim tim
zakljucujemo da v_a i v_a-1 moraju da budu iste duzine.
```

Za slucajnu promenljivu postoji prefiksni optimalni kod, tako da dve poslednje reci su iste duzine i mogu se razlikovati samo u poslednjem slovu.

```
- - -a
- - -b

Pretpostavljamo da to nije tacno i bez gubitka opstosti, pret
da se zanja rec zavrsava sa 0.

Menjamo kod tako da zadrzavamo srednju duzinu reci.

Poslednju kodnu rec izbacujemo i postavljamo 1.
Ukoliko takva rec postoji i bila je u kodu, postavljamo je na
i time se ne menja srednja duzina reci. Ako nije bila, tada k
prefiksni jer ako je bilo sta prefiks od nje, onda je samim t
prefiks i od prve reci sto je nemoguce
jer je polazni kod prefiksni.
```

Pitanje koje se postavlja kada posmatramo potrebne uslove, da li kod koji zadovoljava sva ova tri uslova obavezno i optimalan?

E pa nazalost, ispostavlja se da to nije tacno.

Shannon-Fano kod

Shannon-Fano kod zadovoljava uslove optimalnosti, ali nije optimalan. I on je bio kontruisan da pravi optimalne kodove.

```
Ako je A = {a_1….a_n} skup simbola alfabeta i
neka su p_1..p_n odgovarajuce verovatnoce gde je p_i = p(a_i)
Kontruisemo kod tako da skup simola podelimo na dva dela
tako da su njihove verovantnoce sto priblizniје. Apsolutna ra
minimaln p_1...p_i i p_i+1...p_a.
Gde i dodeljujemo 0, a i+1 do a dodeljujemo 1.
To znaci da pocinu sa 0 ili 1.
Rekurzivno primenjujemo siti postupak.
Ako je skup jendoelementarni, onda je kraj.


Pimer algoritma:
V = {0,100,101,110,1110,1111}
I = {0}
II = {100,101,110,1110,1111}
_____________________________________
I = {100,101}
II = {110,1110, 1111}
_____________________________________
I = {100, 110}
II = {101, 1110, 1111}
_____________________________________
I = {1110}
II = {1111}
_____________________________________
```

Huffmanov algoritam

Sastavljamo optimalni kod koriscenjem metode bottom-up. Uocavamo dva simbola koja se najmanje verovarna simbola i od njih pravimo jedan simbol koji

ima ukupnu verovatnocu pojavanja koda.

Na taj nacin trazimo optimalnu duzinu.

Ako na kraju dobijamo dva simbola, jedan kodiramo 0, jedan 1.

Sada dobijamo bottom-up pristup. A optimalni kod citamo top-down, od korena do lista.

Moguca je primena i za b≥3, tada se b najmanjih simbola tada kombinuje u svakom koraku.

```
Potrebno je da dokazemo da Huff algoritam daje optimalni kod.
Dokazujemo ukoliko je V optimalan kod za neku slucajnu promen
X sa raspodelom p i ako p_j mozemo da razdvojimo kao q_0+q_1
pri cemu su q+0 i q_1 manji od p_a.
Onda je kod V' koji dobijamo kada na kodnu rec Vj
dodamo 0 ili 1, optimalan za kod
koji ima slucajnu raspodelu.

Ukoliko je kod koji smo dobili primenom HUff metoda za jednu
bice optimalan i za ostale.

Posmatramo kod V koji je optimalna za raspodelu b.
Posmatramo kodne reci v_j0 i v_j1. Posto je V prefisan kod,
znaci da se ove dve reci ne pojavljuju nigde u kodu.
Jer ako bi bilo jednako
V_k onda bi bio prefiks od neke reci.

Dobijamo da je kod V' dobro definisan i prefiksan.
Njegovu srednju duzinu mozemo da izracunamo:
n_vj = sum[n_ip_i + (n_j+1)(q_0+q_1) = sum[n_ip_i+p_j] = n_v+



Sada W' zelimo da dokazemo da je optimalna za raspodelu:
p_1 >= p_2 >= ... p_j-1 >= p_j-2 >= ... p_n >= q_0 >= q_1

W' = {w_1..w_2...w_j-1...w_a, w_a0, w_a1}
Kao potrebne uslove smo dokazali da postoji optimalni kod
tako da se dve kod reci razlikuju u jednom simbolu (0 ili 1).
```

```
Konstruisamo kod W gde izbacujemo poslednje dve reci:
W = {w_1...w_j-1, w, w_j+1,...w_a}

za nju postoji optimalni kod za nju vazi da je w_a0 = w_0 i w_
gde je ovaj kod prefiksni, a w nije prefiks nijedne kodne rec
sto ako bi bio prefiks neke druge, bio bi duzi od neke ili is
Najkraci koliko w moze da bude je w_0 ili w_1. Samim tim
nijedna rec ne moze da bude prefiks w.

n_w = n_w + p_j

Posto je W optimalni kod tada vazi
n_v <= n_w

i dobijamo isto za n_v'

n_v' = n_v + p_h <= n_w + p_j = n_w' -> v' je optimlani kod

Primenjujemo transformaciju koda, da zbog uslova mozemo na ob
da dokazemo optimalno. Ako je v optimalnan onda je i v' optim

Svaki od konstruisanih kodava je optimalna, pa je samim tim i
polanzi kod optimalan kod.
```

Imamo algoritam koji na osnovu verovatnoce pojavljivanja simbola konstruisemo optimalni kod koji ima najmanju srednju vrednost duzine kodne reci i primenom tog koda, dobijamo najkracu kodnu poruku i samim tim je to najbolja moguca kompresija.

Verovatnoca pojavljivanje je podjednaka sta god mi dobijali, ali to ne pije vodu.

Pretpostavka u nezavisnosti simbola, najcesce nije ispunjena. Tako da ako primenimo Huff algoritam ne dobijamo neke idealne rezultate.

Ako stavimo pretpostavku da su grupe od dva, tri slova medjusobno nezavisne.

Za svaku grupu slova, racunamo verovatnocu pojavljivanja, primenjujemo HUff algoritam.

Ako imamo N slova i specijalne karaktere i grupisemo ih u k grupa, znaci imamo n^k simbola. Za velike poruke imamo problem tacnosti i optimalnosti samom

koda.

Algoritmi koji koriste predhodno znanje i dinamicni adaptiraju sam kod.

Huff algoritam se koristi onog trenutka kada dobijemo podatke, gde su pojedinacni simboli potpuno ili delimicno nezavisni. Prvo se korisni neki algoritam koji odredjuje tu nezavisnost, pa nakon toga se primenjuje Huff algoritam. → tema prvog projekta.