Föreläsning 9

MA1508

Sannolikhetsträd

Om vi har ett binärt träd kan vi välja att göra en sannolikhetsfördelning på löven. Sannolikheterna för alla löven ska givetvis summera till 1.

Har man sannolikheter på löven får man naturligt sannolikheter på noderna. Sannolikheten för en nod blir summan av sannolikheterna för dess två barn. (Det gäller även för noder inne i trädet.) Sannolikheterna vid noderna kommer inte vara sannolikheten att vi väljer den noden, för vi väljer bara löv, utan det blir sannolikheten att vi passerar den noden när vi går från roten till det slumpmässigt valda lövet.

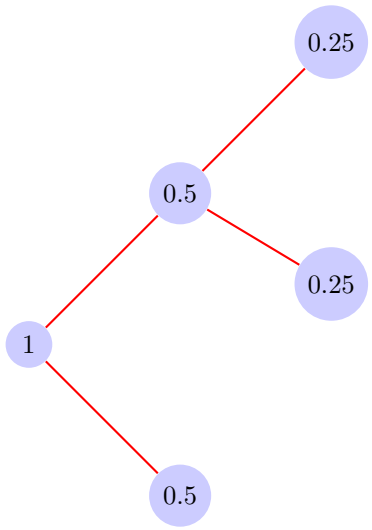
På grund av kopplingen mellan binära träd och prefixfria koder kommer vi vara intresserade av det genomsnittliga djupet i ett träd. Om löven har sannolikheten $p_1, p_2, \ldots, p_n$ och djupen $d_1, d_2, \ldots, d_n$ definierar vi det det genomsnittliga djupet som

$$\sum_{i=1}^{n} p_i d_i.$$

Det är samma sak som väntevärdet av djupet för ett löv i trädet.

Hjälpsats. *I ett binärt träd med sannolikheter är det genomsnittliga djupet lika med summan av sannolikheterna i alla noderna.*

Exempel 1. Betrakta trädet nedan, där jag skrivit sannolikheterna i noderna. Summan av sannlikheterna i noderna är 1.5. Och det är också genomsnittliga djupet, vilket ni kan kontrollera.



△

Bevis. Man kan göra ett induktionsbevis. Det är sant för trädet med bara två löv. Om vi ersätter ett löv som har sannolikheten p och djupet d med en nod vars barn har sannolikheten q, r där $q + r = 1$ så ökar genomsnittliga längden med $(q+r)\cdot 1 = p$. Vi har fått en ny nod vars sannolikhet är p så även summan av sannolikheterna i alla noder ökar med p. Ni kan fylla i detaljerna i beviset om ni vill.

□

Huffmankoder

Nu kommer vi in på Huffmankoder, vilket ni nog väntat på. Anta vi r stycken olika symboler som vi ska koda. Låt sannolikheten för symbol nummer i vara p_i. Vi kan anta att $p_1 \geq p_2 \geq \ldots \geq p_r$. Vi vill välja en prefixfri kod vars genomsnittliga längd är så liten som möjligt.

I en optimal kod så kan inte en mer sannolik symbol ha ett längre kodord än en mindre sannolik symbol. Det känns uppenbart, och kan lätt bevisas genom att vi inser att om vi bara byter kodorden med varandra så får vi en bättre kod annars.

Vi såg ett exempel förra gången där om vi skapade ett träd för en kod där längderna var 2,2,2 och 3 så fick vi ett tomt löv. Det kan inte hända i ett träd som kommer från en optimal kod.

Hjälpsats. *I det binära trädet för en optimal kod så finns det inga tomma löv.*

Bevis. Om vi har ett tomt löv kan vi stryka det och låta dess syskon ta förälderns plats. Vi kan fortsätta så tills det inte finns några tomma löv kvar. □

Vi fortsätter att stegvis fundera ut hur trädet för en optimal kod måste se ut. Vi inser att hitta det optimala trädet är precis samma sak som att hitta den optimala koden.

Hjälpsats. *Det finns en optimal prefixfri kod så att de två minst sannolika kodorden (alltså kodorden för symbolerna med nummer $r-1$ och r) bara skiljer i sista biten.*

Bevis. Vad kan r ha för syskon i trädet? Den måste ha ett syskon eftersom det inte finns några tomma löv. Det finns inga löv som har större djup än vad r har. Om det är inte är $r-1$ som är syskon så kan vi byta plats på $r-1$ och den andra symbolen utan att öka genomsnittliga djupet. □

Vi kommer använda denna hjälpsats för att bygga trädet för den optimala koden baklänges. Kan anta att de två minst sannolika symbolerna är syskon vid en nod.

Anta vi låtsas slå ihop de två minst sannolika symbolerna till en ny symbol med sannolikheten $p_{r-1}+p_r$, låt oss kalla den $\boxed{\text{S}}$, och hittar ett träd för den nya uppsättningen av symboler. Om vi sedan återersätter lövet för $\boxed{\text{S}}$ med en nod med symbolerna $r-1$ och r som barn så ökar trädets genomsnittliga djup med precis $p_{r-1}+p_r$, oberoende av hur resten av trädet ser ut. Alltså om vi hittar det optimala trädet med symbolerna 1,2, ... r-2, $\boxed{\text{S}}$ kommer det ge oss det optimala trädet för den ursprungliga listan av symboler med en enkel ersättningsoperation.

Så Huffmans procedur för att hitta en optimal prefixfri kod är att följande

1. Skapa varsitt hörn för alla symbolerna, som kommer vara löv i det klara trädet

2. Om det finns bara två hörn som inte har föräldrar, skapa en rot med dem som barn. Trädet du skapat är det optimala.

3. Om det finns fler än två hörn som inte har föräldrar, ge de två minst sannolika hörnen utan en förälder samma förälder och ge den noden deras gemensamma sannolikhet. Gå sedan tillbaka till steg 2.

Se t ex sidan 70 i boken för ett exempel.

Exempel 2. Om vi har symbolerna a, b med sannolikheterna 0.9 och 0.1 så kan inte blir kodorden 0 och 1. Ingen komprimering sker.

△

Exempel 3. Låt oss återkomma till vårt föregående exempel. Vi kodar nu två symboler i taget. Nu är det aa, ab, ba och bb som kommer stå i löven i vårt träd. Sannolikheterna är (vi antar oberoende) 0.81, 0.09, 0.09 och 0.01. Nu blir kodorden i en optimal kod 0, 10, 110 och 111. Genomsnittliga längden av kodorden är 1.29. Men om vi delar med 2 för att ta hänsyn till att vi nu kodar två symboler åt gången blir genomsnittliga längden 0.645.

Vi skulle kunna fortsätta att ta längre block i taget, t ex 3 eller 4 symboler. Men beräkningarna blir mer omfattande för varje steg. Nästa vecka kommer vi diskutera hur lågt vi äntligen kan få antalet bitar per kodord om vi tar extremt långa block.

△

Komplexiteten hos Huffmanalgoritmen

Huffmanalgoritmen har tidskomplexitet $O(n \log n)$ med n symboler. Det är bra, men om antalet symboler ökar exponentiellt så ökar också Huffmanalgoritmens tidsåtgång exponentiellt. Och det är precis vad som händer när vi kodar allt längre och längre block av symboler i taget.

Eftersom man kan tänkas vilja hantera långa block i taget, för att få så mycket komprimering som möjligt, har man uppfunnit alternativ till Huffmankoder som används i många sammanhang.

Unikt avkodningsbara koder

Det är värt att notera att du inte kan få något bättre än Huffmankoder om du släpper kravet att koden är prefixfri men fortfarande kräver den är unikt avkodningsbar. Bokens bevis för det är rätt roligt, men definitivt överkurs att förstå.