Jak mierzyć wartość informacji

Pewne zdarzenie może zajść na N sposobów. Możliwości te mają prawdopodobieństwa $(p_1, \ldots, p_N)$. Jak zmierzyć ilość posiadanej informacji o tym, na jaki sposób zaszło zaszło zdarzenie?

Uwaga: Od tej pory tak numerujemy możliwe sposoby zajścia zdarzenia, że ich prawdopodobieństwa tworzą ciąg nierosnący

Propozycja 1: Ile minimalnie pytań TAK/NIE trzeba zadać, by zdobyć tę informację?

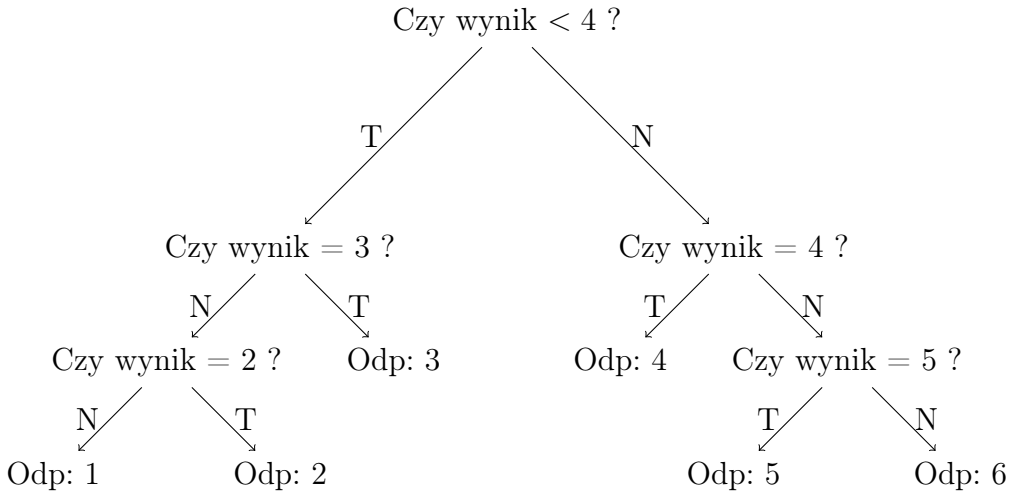
Zauważmy, że jeżeli zdarzenie mogło zajść na trzy sposoby, to jedną z możliwości zidentyfikujemy po jednym pytaniu, a pozostałe po zadaniu dwóch pytań (zakładamy, że nie zadajemy pytań oczywistych, na które jest tylko jedna odpowiedź).

Ilość pytań może zatem zależeć od ostatecznej odpowiedzi. Żeby liczba ta nie zależała od odpowiedzi, bierzemy średnią liczbę pytań które zadamy.

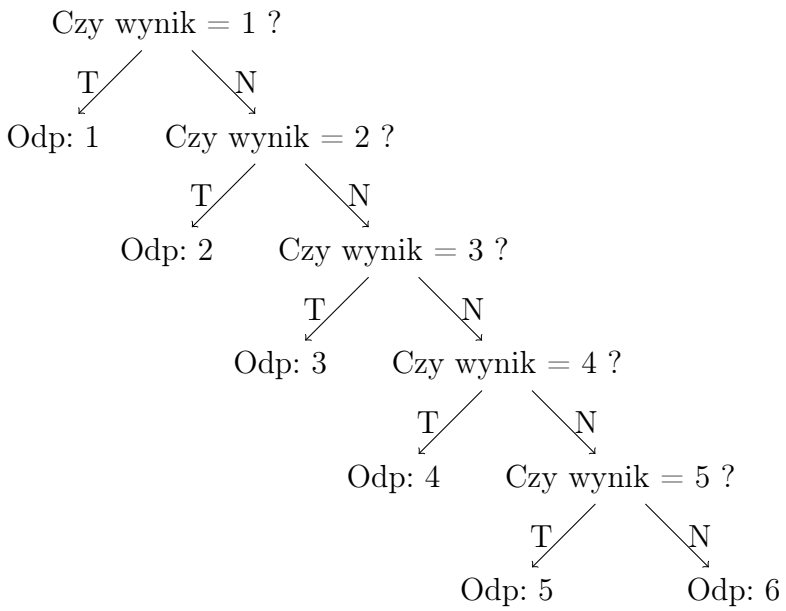
Propozycja 2: Ile średnio pytań TAK/NIE zadamy, by zdobyć informację.

Załóżmy, że znamy wynik rzutu sześcienną kostką o równych prawdopodobieństwach wszystkich wyników. Informację o tym zdarzeniu można zdobyć konstruując wiele systemów identyfikacji (zbioru pytań, które prowadzą nas do odpowiedzi). Rozważmy dwa z nich.

System 1:



System 2:



Zauważmy, że w systemie pierwszym zadamy średnio $\frac{8}{3}$ pytań by uzyskać odpowiedź, a w drugim $\frac{10}{3}$. Nawet gdy nie zadajemy oczywistych pytań, średnia ilość pytań potrzebna do identyfikacji odpowiedzi może być różna dla różnych systemów identyfikacji. Zmodyfikujemy zatem naszą propozycję miary posiadanej informacji:

Propozycja 3: Ile średnio pytań TAK/NIE musimy zadać, by zdobyć informację, przy użyciu optymalnego systemu identyfikacji.

Drzewa binarne

Żeby można było sformalizować pojęcie *systemu identyfikacji* musimy wprowadzić pojęcie drzewa binarnego.

Definicja: *Drzewem* nazywamy graf spójny bez cykli ($\Leftrightarrow$ każde dwa wierzchołki są połączone dokładnie jedną drogą)

Definicja: *Drzewem ukorzenionym* nazywamy graf spójny z wyróżnionym wierzchołkiem - *korzeniem.*

- W drzewie ukorzenionym *rodzicem* wierzchołka A jest pierwszy wierzchołek na drodze łączącej go z korzeniem. Tylko korzeń nie ma rodzica.
- Zbiór wierzchołków, których rodzicem jest wierzchołek A, nazywamy *dziećmi* wierzchołka A.
- Wierzchołki nie posiadające dzieci (stopnia 1) nazywamy *wierzchołkami końcowymi (liśćmi).* Pozostałe wierzchołki nazywamy *pośrednimi.*
- Długość drogi łączącej wierzchołek z korzeniem nazywamy *rzędem wierzchołka.*
- *Rzędem drzewa* nazywamy maksymalny rząd jego wierzchołków.

Definicja: *Drzewem binarnym* nazywamy drzewo ukorzenione, w którym każdy wierzchołek ma co najwyżej 2 dzieci.

Definicja: Drzewo binarne w którym wszystkie liście mają ten sam rząd i w którym każdy wierzchołek pośredni ma dokładnie 2 dzieci nazywamy *drzewem binarnym pełnym (regularnym).*

Udowadniamy prosty:

Lemat: Drzewo binarne rzędu k ma co najwyżej 2^k wierzchołków końcowych. Równość zachodzi dla drzewa binarnego pełnego.

Dowód: indukcja ze względu na rząd drzewa.

W dalszej części wykładu będziemy potrzebować następującego twierdzenia o drzewach binarnych:

Twierdzenie (Nierówność Krafta):

1. Rzędy wierzchołków końcowych w drzewie binarnym spełniają nierówność:
$$\sum_i 2^{-n_i} \leqslant 1$$
(gdzie i numeruje wierzchołki a n_i oznacza rząd i-tego wierzchołka).
2. Dla każdego zbioru liczb naturalnych dodatnich spełniających powyższą nierówność istnieje drzewo binarne, o rzędach wierzchołków równych tym liczbom.

Dowód: 1 - Wybieramy liczbę N większą lub równą rzędowi drzewa. W każdym wierzchołku końcowym o rzędzie $n_i < N$ dobudowujemy drzewo pełne rzędu $N-n_i$. W powstałym grafie wszystkie wierzchołki końcowe mają rząd N. Ilość jego wierzchołków końcowych jest równa $\sum_i 2^{N-n_i}$ i na mocy lematu jest ona mniejsza lub równa 2^N.

2 - Przepisujemy nierówność jako $\sum_j k_j 2^{-m_j}$ grupując wierzchołki o tych samych rzędach. Licznik j przebiega nie po wierzchołkach, ale po wartościach rzędu, a k_j oznacza krotność j-tej wartości rzędu.

Ograniczenie na liczbę f_2 dostępnych węzłów rzędu m_2 wynosi:

$$f_2 = 2^{m_2} - k_1 * 2^{m_2-m_1} = 2^{m_2}(1 - k_1 2^{-m_1}).$$

Analogicznie, dla wyższych poziomów:

$$f_n = 2^n(1 - \sum_{j<n} k_j 2^{-m_j}).$$

Zauważmy, że jeżeli liczby $k_1, \ldots, k_n$ spełniają nierówność Krafta, to $\forall i\ k_i < f_i$, zatem możemy skonstruować drzewo. Gdy nierówność jest ostra, mamy więcej węzłów końcowych niż potrzeba, możemy zmniejszyć ich liczbę zmniejszając liczbę dzieci niektórych wierzchołków pośrednich $\square$

Zauważmy, że system identyfikacji odpowiedzi to dokładnie drzewo binarne. Nierówność Krafta pozwala powiązać drzewo binarne z pewnym ciągiem liczb spełniających nierówność Krafta. Te liczby, to ilość pytań potrzebnych do identyfikacji różnych odpowiedzi.

Definicja: Systemem identyfikacji zbioru N wyników zachodzących z prawdopodobieństwami $(p_1, \ldots, p_N)$ jest ciąg liczb naturalnych dodatnich $S = (n_1, \ldots, n_N)$ spełniających nierówność Krafta. Dla systemu identyfikacji S definiujemy jego wartość oczekiwaną $\mathbb{E}(S) = \sum_i p_i n_i$.

Optymalne systemy identyfikacji

Definicja: System identyfikacji N wyników S nazywamy *optymalnym*, jeżeli dla dowolnego innego systemu identyfikacji N wyników S' zachodzi $\mathbb{E}(S) \leqslant \mathbb{E}(S')$.

Optymalne systemy identyfikacji mają następujące własności:

O1: Jeżeli $p_i > p_j$, to $n_i \leqslant n_j$.

Dowód: Załóżmy, że dla pewnego systemu identyfikacji znajdziemy parę indeksów (i, j) dla której zachodzi: $p_i > p_j$ i $n_i > n_j$. Wtedy możemy rozważyć system S' w którym węzły końcowe (i, j) są zamienione miejscami. Porównajmy wartości oczekiwane obu systemów:

$$\mathbb{E}(S) - \mathbb{E}(S') = p_i n_i + p_j n_j - p_i n_j - p_j n_i = (p_i - p_j)(n_i - n_j) > 0$$

zatem system S zadaje średnio więcej pytań niż system S' i nie może być optymalny.

O2: Z każdego węzła pośredniego wychodzą dwie krawędzie (nie ma w systemie pytań oczywistych)

Dowód: Usunięcie pytania oczywistego zmniejszy o jeden liczbę pytań potrzebnych do identyfikacji pewnej liczby odpowiedzi, co poprawi wartość oczekiwaną systemu. System z pytaniem oczywistym nie jest zatem optymalny.

O3: Dwa węzły końcowe o najmniejszych prawdopodobieństwach mają maksymalny rząd.

Dowód: Konsekwencja O1 i O2

O4: Jeżeli dla zbioru N wyników o prawdopodobieństwach $(p_1, \ldots, p_N)$ system identyfikacji $S = (n_1, \ldots, n_N)$ (rzędy węzłów w kolejności malejących prawdopodobieństw) jest optymalny, to dla zbioru $N - 1$ wyników o prawdopodobieństwach $(p_1, \ldots, p_{N-1} + p_N)$ system identyfikacji $S' = (n_1, \ldots, n_{N-1} - 1)$ jest również optymalny.

Dowód: Jeżeli S' nie jest optymalny, to istnieje T' od niego lepszy. Z T' konstruujemy system identyfikacji N wyników $T = (n_1, \ldots, n_{N-2}, n_{N-1}, n_N)$. Obliczamy jego wartość oczekiwaną i wychodzi niższa niż dla systemu S, czyli system S z którego pochodzi S' nie mógł być optymalny.

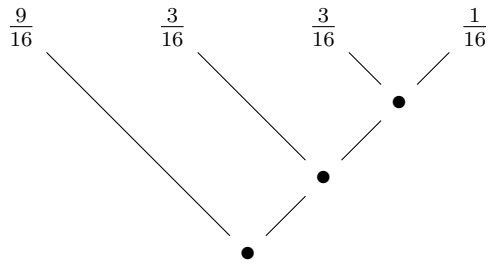
Własności O3 i O4 dają już metodę konstrukcji systemu optymalnego dla N wyników. Wiemy, że odpowiedzi o najmniejszych prawdopodobieństwach muszą być połączone węzłem nadrzędnym. Jeżeli połączymy te dwie odpowiedzi w jedną, dostaniemy znów system optymalny dla $N - 1$ wyników, dla którego znów dwie najmniej prawdopodobne odpowiedzi łączy węzeł nadrzędny, itp. Algorytm ten jest dobrze określony, bo w każdym przebiegu pętli maleje o jeden rozmiar systemu identyfikacji, więc skończy się po skończonej liczbie przebiegów. Nosi on nazwę **konstrukcji Huffmana**.

Do każdej odpowiedzi w systemie identyfikacji możemy przypisać ciąg odpowiedzi na poszczególne pytania TAK/NIE (1 dla TAK, 0 dla NIE). Jeżeli system jest optymalny, to kod tak nazywa się *kodem Huffmana* i jest to spośród wszystkich możliwych kodów kod o najkrótszym średnim słowie kodowym.

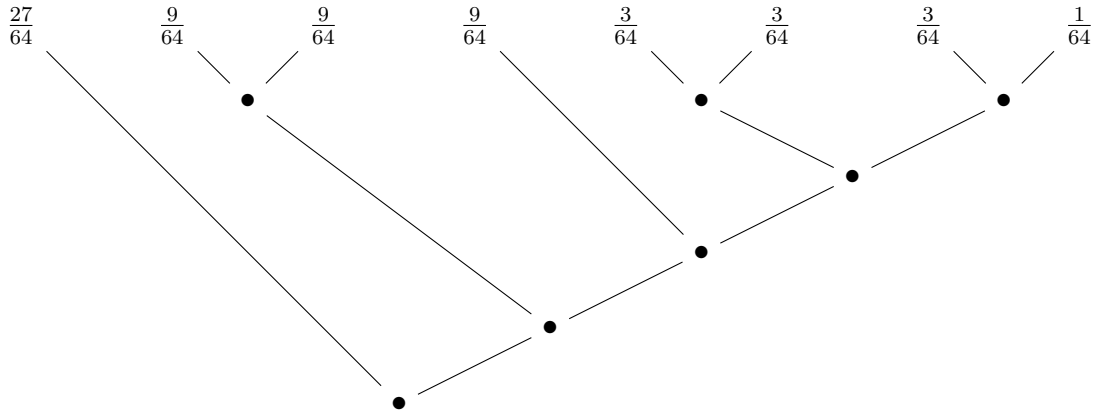
Jeżeli rzucamy monetą dającą z równym prawdopodobieństwem orła i reszkę, do zidentyfikowania odpowiedzi potrzebujemy jednego pytania. Gdyby wypadał zawsze tylko jeden wynik, to potrzebowalibyśmy 0 pytań. Powinno być tak, że gdy jedna z możliwości jest bardziej prawdopodobna, potrzebujemy średnio mniej niż jednego pytania by zidentyfikować wynik. Moneta tak jest bardziej przewidywalna niż uczciwa moneta, więc informacja o wyniku rzutu powinna być mniej warta. Jak to możliwe, skoro do identyfikacji jej wyniku wciąż potrzebujemy dokładnie jednego pytania?

Wyobraźmy sobie, że rzucamy dwa razy monetą o prawdopodobieństwach (0.25,0.75). Wyniki ta-

kiego eksperymentu mają prawdopodobieństwa $(9/16, 3/16, 3/16, 1/16)$. Optymalnie identyfikujemy je za pomocą systemu $(1, 2, 3, 3)$, który daje średnią liczbę pytań $27/16$, czyli ≈ 0.84 pytania na jeden rzut monetą.



Jeżeli rzucamy trzy razy taką monetą, mamy rozkład prawdopodobieństwa wyników $(27/64, 9/64, 9/64, 9/64, 3/64, 3/64, 3/64, 1/64)$. Identyfikujemy go optymalnie za pomocą systemu $(1, 3, 3, 3, 5, 5, 5, 5)$:



który potrzebuje średnio $158/64$ pytań do identyfikacji wyniku, czyli ≈ 0.82 pytania na jeden rzut monetą.

Widzimy, że średnia liczba pytań na jeden rzut spada gdy identyfikujemy coraz większe grupy monet. Za miarę wartości informacji o wyniku pomiaru zmiennej losowej powinniśmy brać wartość graniczną tego ciągu.

Ograniczenia na średnią liczbę pytań

Dla rozkładu prawdopodobieństwa $X = (p_1, \ldots, p_N)$ definiujemy **entropię Shannona**:

$$H(X) = -\sum_i p_i \log_2 p_i,$$

z ciągłości przyjmujemy, że $0 \log_2 0 = 0$.

Łatwo udowodnić, że dla rozkładu prawdopodobieństwa dwóch zdarzeń niezależnych X, Y mamy $H(X,Y) = H(X) + H(Y)$:

$$H(X,Y) = -\sum_{i,j} p_i q_j \log_2 p_i q_j = -\sum p_i q_j (\log_2 p_i + \log_2 q_j) = -\sum_i p_i \log_2 p_i - \sum_j q_j \log_2 q_j = H(X) + H(Y)$$

Nierówność Kleina: $\log_2 x \leqslant \log_2 e \cdot (x-1)$, równość $\iff x = 1$.

Dowód: Znajdź styczną w $x = 1$ do wykresu $\log_2 x$ i sprawdź że jest to funkcja wklęsła.

Przy jej pomocy udowadniamy następujące twierdzenie o informacji Shannona:

Twierdzenie: $H(X) \leqslant \mathbb{E}S$ dla dowolnego systemu identyfikacji S

Dowód: $H(X) - \mathbb{E}(S) = -\sum_i p_i \log_2 p_i - \sum_i p_i n_i = \sum_i p_i \log_2(2^{-n_i}/p_i) \leqslant \sum_i p_i \log_2 e(2^{-n_i}/p_i - 1) = \log_2 e(\sum_i 2^{-n_i} - 1) \leqslant 0$ □

O wyjątkowości funkcji informacji Shannona mówi następujące twierdzenie:

Twierdzenie: Istnieje taki system identyfikacji S, że $H(X) \leqslant \mathbb{E}S < H(X) + 1$

Dowód: $-\sum_i p_i \log_2 p_i \leqslant \sum_i p_i n_i \leqslant -\sum_i p_i(\log_2 p_i - 1)$. Udowodnimy, że nierówność zachodzi dla odpowiednich składników sum: $-\log_2 p_i \leqslant n_i \leqslant -\log_2 p_i + 1$. Dla każdej liczby p_i istnieje liczba naturalna w przedziale $[p_i, p_i + 1)$. Pozostaje sprawdzić, czy liczby te spełniają nierówność Krafta □

System identyfikacji o takich liczbach pytań nazywamy systemem Shannona. W przedziale $[H(X), H(X)+1)$ siedzi oczywiście optymalny system identyfikacji zwracany przez konstrukcję Huffmana, ale jest on na tyle szeroki by pomieścić również inne systemy. Dla rozważanych trzech rzutów monetą system identyfikacji Shannona ma następujące liczby pytań:

$$\begin{array}{ll} 1\times & 6 \\ 3\times & \lceil 6 - \log_2 3 \rceil = 5 \\ 3\times & \lceil 6 - 2\log_2 3 \rceil = 3 \\ 1\times & \lceil 6 - 3\log_2 3 \rceil = 2 \end{array}$$

zauważmy, że system ten nie wysyca nawet nierówności Krafta (zawiera pytania oczywiste), a mimo to wciąż jest wystarczająco blisko dolnego ograniczenia na średnią liczbę pytań.

Z twierdzenia wynika, że jeżeli będziemy identyfikowali po n wyników na raz, to średnia liczba pytań będzie w przedziale $[H(X), H(X) + \frac{1}{n})$. Kodując wyniki w odpowiednio dużych pakietach, możemy się zbliżyć ze średnią długością słowa kodowego dowolnie blisko wartości $H(X)$ i tę wielkość uznajemy za miarę informacji o zdarzeniu X.

Entropia Shannona jest średnią z funkcji $p \mapsto -\log_2(p)$, która jest wartością informacyjną zdarzenia elementarnego (funkcja niespodzianki). Mało prawdopodobne zdarzenia zaskakują nas bardziej niż te bardziej prawdopodobne. Funkcja ta powinna być funkcją ciągłą o naturalnych własnościach:

- $f : [0, 1] \to [0, \infty]$, $f(0) = \infty$, $f(1) = 0$
- powinna być addytywna na zdarzeniach niezależnych: $f(pq) = f(p) + f(q)$

Jedyna funkcją spełniającą te własności jest $-\log$, z dokładnością do podstawy.

Entropia Shannona

Własności entropii Shannona

Entropia Shannona, jako funkcja H:$\Delta^n \to \mathbb{R}$ ma następujące własności

1-3 Jest funkcją dodatnią, symetryczną i ciągłą na rozkładach prawdopodobieństwa.

4 Wklęsłość

Dowód: Wystarczy udowodnić, że $p \mapsto -p \log_2 p$ jest wklęsła.

5 Subaddytywność Dla dwóch zmiennych losowych X, Y zachodzi $H(XY) \leqslant H(X) + H(Y)$, a równość zachodzi wtedy i tylko wtedy, gdy zdarzenia są niezależne (addytywność).

Dowód: $H(X)+H(Y)-H(XY) = \sum_i p_i \log_2 p_i + \sum_j p_j \log_2 p_j - \sum_{ij} p_{ij} \log_2 p_{ij} = \sum_{ij} p_{ij} \log_2 \frac{p_i p_j}{p_{ij}} \leqslant \log_2 e \sum_{ij} p_{ij} (\frac{p_i p_j}{p_{ij}} - 1) = 0$.
Równość zachodzi, gdy $\forall i \; p_{ij} = p_i p_j$ $\square$

Subaddytywność jest intuicyjnie jasna - informacja o całości nie może przekraczać informacji o częściach składowych. Później zobaczymy, że własność ta nie zachodzi w świecie kwantowym.

6 $H(X, Y) \geqslant H(X), H(Y)$

Dowód: Udowodni się samo później

Informacja warunkowa

Załóżmy że wiemy, że przy pomiarze zmiennej Y zmierzono j-ty wynik. Wtedy rozkładem prawdopodobieństwa zmiennej X jest $\{p_{i|Y=j} = \frac{p_{ij}}{p_j}\}$. Przy znanym j-tym wyniku Y, informacja o X jest równa:

$$\sum_i \frac{p_{ij}}{p_j} \log_2 \frac{p_{ij}}{p_j} = \frac{1}{p_j} \sum_i p_{ij} (\log_2 p_{ij} - \log_2 p_j)$$

a $j-ty$ wynik Y pojawia się z prawdopodobieństwem p_j, więc średnio gdy znamy wynik Y, informacja o X wynosi:

$$\sum_{ij} p_{ij} \log_2 p_{ij} - \sum_{ij} p_{ij} \log_2 p_j = H(X, Y) - H(Y)$$

Czyli jest dokładnie równa informacji o obu zmiennych - informacja o zmiennej Y. Wielkość tę nazywamy informacją warunkową i oznaczamy jako $H(X|Y)$.

Własności informacji warunkowej

1. $H(X|Y) \leqslant H(X)$, równość zachodzi dla zmiennych niezależnych
2. $H(X|Y) \geqslant 0$ (co udowadnia szóstą własność entropii Shannona)

Dowód 1: wynika z subaddytywności entropii Shannona

Dowód 2: wynika wprost z definicji informacji warunkowej

Informacja wzajemna

Znajomość wyniku zdarzenia Y, w ogólności zmniejsza wartość informacji o wyniku zdarzenia X (nie zmniejsza tylko gdy X i Y są niezależne). Możemy zatem zapytać, ile informacji o wyniku zdarzenia X dostarcza informacja o wyniku zdarzenia Y. Oznaczamy tę wielkość jako $I(X:Y)$. Jest to różnica:

$$I(X:Y) = H(X) - H(X|Y) = H(X) + H(Y) - H(X,Y).$$

Ostatnia równość pokazuje, że jest to wielkość symetryczna. Znajomość Y dostarcza tyle samo informacji o X, co znajomość X o Y.

Reguły łańcucha

Znajomość wartości pewnej zmiennej Z wpływa w następujący sposób na informację o innych dwóch zmiennych losowych X, Y:

$$H(X,Y|Z) = H(X|Z) + H(Y|X,Z)$$

Dowód: $H(X,Y|Z) - H(X|Z) = H(X,Y,Z) - H(Z) - H(X,Z) + H(Z) = H(Y|X,Z)$ $\square$

Własność $H(X,Y) = H(X|Y) + H(Y)$ uogólnia się do większej liczby zmiennych losowych:

$$H(X_n, X_{n-1}, \ldots, X_1) = \sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1)$$

Dowód: $H(X_n, X_{n-1}, \ldots, X_1) = H(X_n|X_{n-1}, \ldots, X_1) + H(X_{n-1}, \ldots, X_1) = H(X_n|X_{n-1}, \ldots, X_1) + H(X_{n-1}|X_{n-2}, \ldots, X_1) + H(X_{n-2}, \ldots, X_1) = \ldots = H(X_n|X_{n-1}, \ldots, X_1) + \ldots + H(X_2|X_1) + H(X_1)$ $\square$

Podobną własność mamy dla informacji wzajemnej:

$$I(X_n, X_{n-1}, \ldots, X_1 : Y) = \sum_{i=1}^{n} H(X_i : Y|X_{i-1}, \ldots, X_1)$$

Dowód:

$$I(X_n, X_{n-1}, \ldots, X_1 : Y) =$$

$$-H(X_n, X_{n-1}, \ldots, X_1, Y) + H(X_n, X_{n-1}, \ldots, X_1) + H(Y) =$$
$$-\sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1, Y) - H(Y) + \sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1) + H(Y) =$$
$$\sum_{i=1}^{n} H(Y|X_{i-1}, \ldots, X_1) + \sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1) - \sum_{i=1}^{n} H(X_i, Y|X_{i-1}, \ldots, X_1) =$$
$$\sum_{i=1}^{n} I(X_i : Y|X_{i-1}, \ldots, X_1)$$

Wszystkie te relacje i definicje dobrze ilustruje diagram Venna.

Kanały informacyjne

Przypomnienie: kombinacja wypukła, to kombinacja liniowa o współczynnikach dodatnich sumujących się do 1.

Definicja: Kanałem informacyjnym nazywamy odwzorowanie liniowe $S : \Delta^n \to \Delta^m$ (przeprowadzające rozkłady prawdopodobieństwa w rozkłady prawdopodobieństwa).

Macierz kanału ma kolumny o wyrazach dodatnich sumujących się do 1 - rozkłady pewne też przechodzą na rozkłady prawdopodobieństwa.

W drugą stronę, macierz o takiej własności jest kanałem - kombinacja wypukła rozkładów prawdopodobieństwa też jest rozkładem prawdopodobieństwa. (sprawdzić!)

Macierze o takiej własności nazywamy *macierzami stochastycznymi*. Macierz stochastyczna to rozkład prawdopodobieństw warunkowych $p(Y|X)$, gdzie Y to wyjście z kanału a X to wejście.

Z tych samych powodów, kombinacja wypukła kanałów dalej jest kanałem, złożenie kanałów dalej jest kanałem (jeżeli zgadzają się rozmiary).

Najczęściej rozważamy kanały przekazujące znaki pewnego n-znakowego alfabetu, wejście i wyjście są tym samym zbiorem, a kanały są wtedy automorfizmami zbioru rozkładów prawdopodobieństwa Δ^n na tym alfabecie.

Taki kanał odwzorowuje sympleks Δ^n w samego siebie, zatem posiada przynajmniej jeden punkt stały (z tw. Brouwera o punkcie stałym).

$$p_0 = Kp_0$$

Dodatkowo, twierdzenie Perona-Frobeniusa zapewnia, że nieredukowalna macierz stochastyczna ma dokładnie jeden punkt stały w sympleksie. Znajdujemy go startując z dowolnego punktu i iterując kanał.

Szczególnymi kanałami są kanały dane przez macierze bistochastyczne, gdzie nie tylko wartości w kolumnach sumują się do 1, ale tak samo wartości we wierszach. Punktem stałym takich kanałów jest stan maksymalnie mieszany (rozkład równomierny), nie zmniejszają one zatem wartości informacji o przesłanym znaku.

Ilość infomacji przesłanej przez kanał

Twierdzenie: Dla dowolnego kanału zachowującego stan maksymalnie mieszany K, $H(KX) \geqslant H(X)$ (po przesłaniu znaku zwiększa się nasza niewiedza o nim i informacja jest więcej warta).

Dowód: Z faktu, że funkcja $x \mapsto x \log_2 x$ jest funkcją wklęsłą.

Uwaga: Nie jest to prawdą dla ogólnych kanałów. Jako przykład wystarczy rozważyć kanał o macierzy

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

Informacja o wejściu kanału K wynosi $H(X)$. Informacja łączna wejścia i wyjścia wynosi $H(X, KX)$. Informacja stracona w kanale, to $H(X|KX)$. Informacja przesłana przez kanał, to $I(X : KX)$.

Pojemność kanału (chanel capacity)

Definicja: Pojemnością kanału informacyjnego nazywamy liczbę $C(K) = \max_X I(X : KX)$. To największa ilość informacji, którą możemy przesłać przy jednokrotnym użyciu kanału.

Niech wejście ma rozkład $\{p_i\}$, a wyjście $q_j = \sum_j K_{ji} p_i$.

$$I(p : q) = -\sum_i p_i h_i + H(q),$$

gdzie h_i oznacza entropię i-tej kolumny macierzy kanału. Przez h oznaczać będziemy wektor wierszowy zbudowany z h_i.

Obliczmy ekstremum warunkowe tej funkcji przy warunku $\sum_i p_i = 1$:

$$\partial_{p_i}(I(p : q) - \lambda \sum_i p_i) = -h_i - \sum_j \log_2(q_j e) \cdot K_{ji} - \lambda = 0$$

$$h_i + \sum_j \log_2(q_j) K_{ji} = -\lambda - \log_2 e = -I(p : q)$$

Ostatnią równość otrzymamy mnożąc przedostatnią równość przez p_i i sumując po i. Wynika stąd, że $\forall i \ h_i + \sum_j [\log_2(q_j) + I(p : q)] K_{ji} = 0$ ($I(p : q)$ weszło do nawiasu dzięki stochastyczności K), zatem ekstremum lokalne istnieje wtedy i tylko wtedy gdy h^T należy do obrazu K^T. Wtedy też $I(p : q) = C(K)$. Ograniczmy się do macierzy K odwracalnych.
W zapisie wektorowym: $[\log_2(q_j)] = -C(K)\mathbb{I} - hK^{-1}$, przy warunku $\sum_j q_j = 1$. Otrzymujemy:

$$C = \log_2 \sum_j 2^{-[hK^{-1}]_j}$$

Przykład: Przepustowość symetrycznego kanału jednobitowego z prawdopodobieństwem błędu ϵ.

$C(K) = 1 - H(\epsilon)$

Przykład: Przepustowość dla dowolnego kanału symetrycznego (to kanał bistochastyczny, w którym wiersze i kolumny różnią się tylko o permutację) nad źródłem N-elementowym.

$C(K) = \log_2(N) - H(\epsilon)$

Przykład: Przepustowość dla kanału, w którym tylko jeden znak podlega błędom transmisji:

$C(K) = \log_2(N - 1 + 2^{-\frac{H}{p_1}})$, gdzie H - entropia wyjścia gdy nadawany jest znak podlegający błędom

Twierdzenie Shannona

Przy rozważaniu symetrycznego kanału jednobitowego dostaliśmy wyrażenia na przepustowość kanału zawierające prawdopodobieństwo błędnego odczytu bitu. Ogólniej związek ten przedstawia **nierówność Fano**:

$$1 + P_e \log_2(N-1) \geqslant H(P_e) + P_e \log_2(N-1) \geqslant H(X|Y),$$

gdzie P_e jest prawdopodobieństwem błędnego odczytania nadanego bitu. W przypadku bitu, wzór redukuje się do $H(P_e) \geqslant H(X|Y)$.

Dowód: Wprowadzamy zmienną losową E o wartościach $0, 1$ mówiącą o tym, czy wystąpił błąd odczytu.

$$\begin{aligned} H(E,X|Y) &= H(X|Y) + H(E|X,Y) = H(X|Y) \\ &= H(E|Y) + H(X|E,Y) \\ &\leqslant H(P_e) + P(E=0)H(X|Y,E=0) + P(E=1)H(X|Y,E=1) \\ &\leqslant H(P_e) + P_e \log_2(N-1) \end{aligned}$$

Chcemy przesyłać pakiety po n znaków. Na zbiorze N^n elementowym $X^{\times n}$ działa teraz kanał $K^{\times n}$ nazywany n-tym rozszerzeniem kanału K. Z reguły łańcucha dla informacji warunkowej mamy następujący:

Wniosek: $C(K^{\times n}) = nC(K)$.

Dowód: $I(X^n, Y^n) = H(Y^n) - H(Y^n|X^n) = H(Y^n) - \sum_{i=1}^n H(Y_i|Y_{i-1}, \ldots, Y_1, X^n) = H(Y^n) - nH(Y|X) \leqslant nH(Y) - nH(Y|X) = nC(K)$. W drugą stronę, wybierając dla każdej zmiennej ten sam rozkład na którym realizujemy przepustowość K, osiągamy równość $\square$

Wynik ten mówi nam, że kodując informację w blokach i przesyłając rozszerzeniem kanału, nie prześlemy więcej niż przy przesyłaniu znak po znaku. Załóżmy kodujemy informację w M słowach kodowych o długości n bitów. Rozpisujemy entropię źródła:

$$H(X^n) = H(X^n|Y^n) + I(X^n, Y^n) \leqslant 1 + P_e^{(n)} H(X^n) + nC(K)$$

Nierówność zachodzi również dla supremum obu stron wynoszącego $H(X^n) = \log_2 M$ (liczba bitów logicznych). Wprowadzamy pojęcie współczynnika transmisji $R = \frac{1}{n} \log_2 M$ (stosunek liczby bitów logicznych do liczby bitów fizycznych), które mówi ile bitów informacji jest przesyłane przy przesłaniu jednego bitu fizycznego (znaku). Mamy wtedy nierówność:

$$R \leqslant \frac{1}{n} + P_e^{(n)} R + C$$

Załóżmy, że konstruujemy ciąg kodów o tym samym współczynniku transmisji i długości słów kodowych $n \to \infty$. Warunkiem koniecznym, by $P_e^{(n)} \to 0$ jest $R \leqslant C$. W drugą stronę, jeżeli $R > C$, to zawsze istnieje graniczna wartość prawdopodobieństwa błędu transmisji $1 - \frac{C}{R}$, poniżej której nie da się zejść.

Fakt ten znany jest jako odwrotność twierdzenia Shannona. Samo twierdzenie Shannona mówi, że dla $R < C$ można skonstruować kod o dowolnie małym prawdopodobieństwie błędu.

Szkic dowodu twierdzenia Shannona

Dla ciągu $\vec{x}$ niezależnych wartości zmiennej losowej X o znanym rozkładzie $\{p_i\}$ i entropii $H(X)$ możemy zdefiniować *entropię serii* wzorem $-\frac{1}{n}\sum_i \log_2 p_i$. Na mocy słabego prawa wielkich liczb:

$$Pr\left(\left|\frac{1}{n}\sum_i X_i - \bar{X}\right| < \epsilon\right) \stackrel{n\to\infty}{\longrightarrow} 1$$

wiemy, że dla dużych n entropia serii będzie z dużym prawdopodobieństwem w przedziale $H(X) - \epsilon, H(X) + \epsilon$. Ciągi o takiej własności nazywamy ϵ-typowymi.

Podobnie możemy rozważać pary ciągów na wejściu i wyjściu kanału i rozważać pary ϵ - *łącznie typowe*.

Definicja: załóżmy, że wejście X i wyjście Y kanału mają rozkład łączny $p(x, y)$. Przesyłamy przez kanał słowo kodowe $\vec{x}$ długości n i otrzymujemy na wyjściu słowo kodowe $\vec{y}$. Słowa te są ϵ - *łącznie typowe*, jeżeli spełnione są poniższe warunki:

$$\begin{aligned} |-\frac{1}{n}\sum_i \log_2 p(x_i) - H(X)| &< \epsilon \\ |-\frac{1}{n}\sum_i \log_2 p(y_i) - H(Y)| &< \epsilon \\ |-\frac{1}{n}\sum_i \log_2 p(x_i, y_i) - H(X,Y)| &< \epsilon \end{aligned}$$

Zbiór par ϵ - *łącznie typowych* będziemy oznaczać jako $A_\epsilon^{(n)}$. Ma on następujące własności:

1. $Pr((\vec{x}, \vec{y}) \in A_\epsilon^{(n)}) \stackrel{n\to\infty}{\longrightarrow} 1$ (dla ustalonego ϵ - maksymalnej dopuszczalnej różnicy między entropią ciągu a entropią źródła, prawdopodobieństwo że para losowana zgodnie z rozkładem łącznym wejścia i wyjścia jest łącznie typowa dąży do 1).

   Dowód: Konsekwencja słabego prawa wielkich liczb.

2. $\#A_\epsilon^{(n)} \leqslant 2^{n(H(X,Y)+\epsilon)}$ (ograniczenie na ilość par ϵ - łącznie typowych)

   Dowód: $1 = \sum_i p(\vec{x}, \vec{y}) \geqslant \sum_{A_\epsilon^{(n)}} p(\vec{x}, \vec{y}) \geqslant \#A_\epsilon^{(n)} 2^{-n(H(X,Y)+\epsilon)}$.

3. Załóżmy, że słowa $\vec{x}$ i $\vec{y}$ generujemy niezależnie od siebie (pochodzą z różnych par). Pytamy o prawdopodobieństwo, że stworzą parę ϵ - łącznie typową:

   $$Pr((\vec{x}, \vec{y}) \in A_\epsilon^{(n)}) \leqslant 2^{-n(I(X:Y)-3\epsilon)}$$

   Dowód: $Pr((\vec{x}, \vec{y}) \in A_\epsilon^{(n)}) = \sum_{A_\epsilon^{(n)}} p(\vec{x})p(\vec{y}) \leqslant 2^{n(H(X,Y)+\epsilon)} 2^{-n(H(X)-\epsilon)} 2^{-n(H(Y)-\epsilon)}$

By udowodnić twierdzenie Shannona powinniśmy wskazać ciąg kodów o współczynniku transmisji $R < C$, w którym prawdopodobieństwo błędu będzie malało do zera gdy $n \to \infty$.

1. Wybieramy pewien dowolny ustalony rozkład prawdopodobieństwa znaków na wejściu.

2. Zgodnie z tym rozkładem prawdopodobieństwa generujemy $m = 2^{nR}$ słów kodowych długości n. Prawdopodobieństwo wylosowania konkretnego kodu $\mathcal{C}$ wynosi $\prod_{w=1}^{m}\prod_{i=1}^{n} p(x_i^{(w)})$.

3. Słowa kodowe będziemy wysyłać z równymi prawdopodobieństwami 2^{-nR}. Po nadaniu słowa $\vec{x}(w)$ otrzymujemy na wyjściu ciąg $\vec{y}$.

4. Zamiast szukać najbliższego słowa kodowego, wprowadzamy następującą procedurę dekodowania: Szukamy wszystkich w takich że para $(\vec{x}(w), \vec{y})$ jest ϵ - łącznie typowa. Błąd popełniamy, jeżeli takich w nie ma lub jest więcej niż jedno lub jeżeli jedyne w jest różne od tego które nadał nadawca.

   Jako E_i oznaczamy zdarzenie, że i-te słowo kodowe tworzy wraz z wyjściem parę ϵ - łącznie typową. Prawdopodobieństwo że popełnimy błąd nadając słowo kodowe o numerze 1 jest równe $Pr(E_1^c \cup E_2 \cup \ldots \cup E_m)$.

Oszacujmy prawdopodobieństwo błędu uśrednione po wszystkich słowach kodowych i po wszystkich kodach:

$$Pr(e) = \sum_{\mathcal{C}} Pr(\mathcal{C}) P(e|\mathcal{C}) = \sum_{\mathcal{C}} Pr(\mathcal{C}) \frac{1}{m} \sum_{w=1}^{m} Pr(e|w, C)$$
$$= \frac{1}{m} \sum_{w=1}^{m} \sum_{\mathcal{C}} Pr(\mathcal{C}) Pr(e|w, C) = \frac{1}{m} \sum_{w=1}^{m} Pr(e|w) = Pr(e|w = 1)$$
$$= Pr(E_1^c \cup E_2 \cup \ldots \cup E_m | w = 1) \leqslant Pr(E_1^c | w = 1) + \sum_{i=2}^{m} Pr(E_i | w = 1)$$

Prawdopodobieństwo, że wyjście tworzy z wejściem parę ϵ - łącznie typową dąży do 1 dla $n \to \infty$. Pierwszy składnik prawej strony będzie więc mniejszy od δ dla odpowiednio dużego n, powiedzmy $n > n_1$. Prawdopodobieństwo że po nadaniu pierwszego słowa kodowego wyjście będzie tworzyło parę ϵ - łącznie typową z innym słowem wejściowym nie przekracza $2^{-n(I(X:Y)-3\epsilon)}$, zatem możemy oszacować drugi składnik prawej strony przez $(2^{nR}-1)2^{-n(I(X:Y)-3\epsilon)} < 2^{-n(I(X:Y)-R-3\epsilon)}$. Jeżeli $R \leqslant I(X:Y)-3\epsilon$ to wyrażenie to będzie zbieżne do 0 i dla pewnego $n > n_2$ mamy $\sum_{i=2}^{m} Pr(E_i|w=1) < \delta$ i łącznie, dla każdego $n > N = \max\{n_1, n_2\}$ prawdopodobieństwo błędnego dekodowania jest mniejsze niż 2δ.

ϵ i δ możemy dowolnie zmniejszać (w zamian za zwiększanie N - długości słowa kodowego). Jeżeli współczynnik transmisji nie przekracza przepustowości kanału, możemy (kosztem długich słów kodowych) zejść z prawdopodobieństwem błędu dowolnie blisko zera.

Wynik dotyczy błędu uśrednionego po wszystkich kodach. Daje to pewność istnienia kodu o interesujących nas własnościach (jeżeli średnia liczb jest mniejsza niż 2δ, to przynajmniej jeden ze składników średniej musi być $\leqslant 2\delta$), natomiast nic nie mówi jak go szukać.

Binarne kody liniowe korygujące błędy

Jeżeli dla kodu długości n nad alfabetem $\{0, 1\}$ słowa kodowe tworzą m-wymiarową podprzestrzeń liniową przestrzeni $\mathbb{Z}_2^n$, to kod nazywamy *binarnym kodem liniowym*. Każdą podprzestrzeń można wyciąć przy pomocy warunku: $Hx = 0$, gdzie liniowo-niezależne kolumny macierzy H są prostopadłe do wszystkich słów kodowych. Macierz H nazywamy *macierzą kontroli parzystości*. Ilość słów wynosi 2^m. W przypadku błędu, dekodujemy przez odnalezienie najbliższego słowa kodowego.

Procesy stochastyczne

Proces stochastyczny $\mathcal{X}$ to rodzina $\{X_t\}_{t\in T}$ zmiennych losowych o wartościach w przestrzeni S indeksowana zbiorem indeksów T. Dla nas proces przyjmował będzie wartości w pewnym zbiorze skończonym, a $T = \mathbb{N}$ (dyskretny proces stochastyczny). Zmienne te nie są na ogół niezależne. Proces w praktyce charakteryzują wszystkie rozkłady łączne zmiennych $\{X_t\}_{t\in T}$.

Proces stochastyczny jest stacjonarny (przesuwalny w czasie), jeżeli jego rozkłady łączne nie zmieniają się przy przesunięciach w czasie. Proces nazywamy ergodycznym, jeżeli jego rozkłady brzegowe da się wyznaczyć z odpowiednio długiej próbki. Ograniczymy zainteresowanie do procesów stacjonarnych i ergodycznych.

Dla procesów stochastycznych będziemy definiować współczynnik entropii na dwa sposoby:

$$H(\mathcal{X}) = \lim_{n\to\infty} \frac{1}{n} H(X_n, \ldots, X_1)$$
$$H(\mathcal{X}) = \lim_{n\to\infty} H(X_n|X_{n-1}, \ldots, X_1)$$

Twierdzenie: Dla procesów stacjonarnych obie definicje są równoważne.

Dowód: Z reguły łańcucha: $\frac{1}{n}H(X_n, \ldots, X_1) = \frac{1}{n}\sum_i H(X_i|X_{i-1}, \ldots, X_1)$.
Mamy $H(X_i|X_{i-1}, \ldots, X_1) \leqslant H(X_i|X_{i-1}, \ldots, X_2) = H(X_{i-1}|X_{i-2}, \ldots, X_1)$. Jest to nierosnący ciąg liczb nieujemnych, zatem jest zbieżny. Ciąg średnich tego ciągu jest zbieżny do tej samej granicy □

Najprostszym przykładem jest mało interesujący proces niezależnych zmiennych losowych o tym samym rozładzie. Współczynnik entropii jest równy entropii zmiennej.

Prostą do analizy, ale już ciekawą klasą procesów są procesy Markowa. W procesie Markowa przyszłość zależy od przeszłości tylko przez teraźniejszość: $P(X_{i+1}|X_i, X_{i-1}, \ldots) = P(X_{i+1}|X_i)$. Dyskretny, stacjonarny proces Markowa jest dany przez macierz przejścia (kanał informacyjny) pomiędzy następującymi w czasie rozkładami prawdopodobieństwa. Dla takiego procesu istnieje rozkład stacjonarny - wniosek z twierdzenia Perona-Frobeniusa.

Proces, którego wartość w chwili zależy od dwóch poprzednich chwil, możemy traktować jako proces Markowa na parach.

Przykład: Niech wartością procesu X_i będzie:

$$\begin{cases} 1 & \text{gdy} \quad X_{i-2} = 0 \\ 0 & \text{gdy} \quad X_{i-2} = 1 \end{cases} \tag{1}$$

Trajektorią takiego procesu jest ciąg: 001100110011 .... Macierzą przejścia dla par jest:

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Macierz przejścia jest w tym przypadku macierzą rzadką - w każdej kolumnie mogą być najwyżej dwie wartości niezerowe.

Podobnie, każdy proces *o pamięci długości* n o wartościach w zbiorze M-elementowym można sprowadzić do procesu Markowa na M^{n+1} elementowej przestrzeni krotek długości $n + 1$. Znów w każdej kolumnie macierzy przejścia może być najwyżej M wartości niezerowych.

Dla ergodycznego stacjonarnego procesu stochastycznego $X = \{X_i\}_{i\in\mathbb{Z}}$ definiujemy jego k-te przybliżenie markowowskie jako proces stochastyczny o rozkładach brzegowych:

$$Q_k(x_{-(k-1)}, \ldots, x_0, x_1, \ldots, x_n) = P(x_{-(k-1)}, \ldots, x_0) \cdot P(x_1|x_{-(k-1)}, \ldots, x_0) \cdot P(x_2|x_{-(k-2)}, \ldots, x_1) \cdot \ldots \cdot P(x_n|x_{n-k}, \ldots, x_{n-1}),$$

gdzie P są prawdopodobieństwami dla procesu X. Aproksymacja zerowego rzędu to ciąg niezależnych zmiennych losowych o rozkładach prawdopodobieństwa zgodnych z rozkładami prawdopodobieństwa dla wyjściowego procesu. Aproksymacja pierwszego rzędu to proces Markowa o prawdopdodobieństwach warunkowych zgodnych z z prawdopodobieństwami warunkowymi dla wyjściowego procesu. Aproksymacja n-tego rzędu to proces Markowa dla krotek długości n.

Przy pomocy przybliżeń markowowskich możemy obliczyć w granicy entropię procesu:

$$\begin{aligned} & -\frac{1}{n}\log_2 Q_k(X_1, \ldots, X_n|X_{-(k-1)}, \ldots, X_0) \\ &= -\frac{1}{n}\left(\log_2 P(x_{-(k-1)}, \ldots, x_0) + \sum_j \log_2 P(x_j|x_{j-1}, \ldots, x_{j-k})\right) \\ &= -\frac{1}{n}\left(\log_2 P(x_{-(k-1)}, \ldots, x_0) + \sum_j \log_2 P(x_0|x_{-1}, \ldots, x_{-k})\right) \\ & \overset{n\to\infty}{\longmapsto} H(X_0|X_{-k}, \ldots, X_{-1}) \overset{k\to\infty}{\longmapsto} H(\mathcal{X}) \end{aligned}$$

Przybliżenia markowowskie języka naturalnego

Pobierzemy duży tekst w języku polskim, wyrzucimy słowa zawierające znaki inne niż polskie litery i zmienimy duże litery na małe:

```
znaki="aąbcćdeęfghijklłmnńoóprsśtuwyzźż "
def oczysc(s):
        s=s.lower()
        s=filter(lambda i:i in znaki,s)
        return ''.join(s)

with open('opowiadania.txt') as f:
        s=f.read()
s=oczysc(s)
```

Ciąg s zawiera teraz N znaków. Wyznaczamy słownik, który ciągom warunkującym przypisuje słowniki, przypisujące następującym po nich literom ilości zliczeń:

```
def wystapienia(n,s):
        d={}
        for i in range(n,len(s)):
                x=s[i-n:i];
                y=s[i]
                if x in d:
                        if y in d[x]:   d[x][y]+=1
                        else:           d[x][y]=1
                else:                   d[x]={y:1}
        return d
```

Dla każdego ciągu warunkującego, normujemy liczby zliczeń i liczymy entropie rozkładu. Następnie, liczymy średnią z tych wartości, z wagami będącymi sumami zliczeń:

```
from math import log
```

```
def ent(l):
        s=0
        for i in l:
                s+=i*log(i)
        N=sum(l)
        return (log(N)-s/N)/log(2)

def ent_war(d):
        s=0; h=0
        for i in d:
                l=d[i].values()
                N=sum(l)
                h+=N*ent(l)
                s+=N
        return h/s
```

Zdefiniujemy dwie kolejne funkcje, które pozwolą nam generować *realizację procesu*, zgodnie z jego macierzą prawdopodobieństw warunkowych:

```
from random import randint

def losujz(d):
        v=randint(0,sum(d.values()))
        s=0
        for i in d:
                s+=d[i]
                if v<=s: return i

def sym(d,n):
        dk=list(d.keys())
        s=dk[randint(0,len(dk))]
        st=len(s)
        for i in range(n-st):
                s+=losujz(d[s[-st:]])
        return s
```

Pierwsza funkcja pozwala losować ze zbioru, dla którego prawdopodobieństwa są zadane przez liczby zliczeń. Druga funkcja na podstawie słownika zliczeń warunkowych losuje kolejny znak z prawdopodobieństwem zależnym od poprzedzającego ciągu znaków. Jest to realizacja procesu stochastycznego.

Dla każdego n (ilość liter warunkujących) generujemy słownik przypisujący ciągowi warunkującemu słownik zliczeń, oraz obliczamy: entropię na znak i jej błąd, entropię kolejnego znaku i jej błąd, następnie generujemy realizację procesu o długości 500 znaków:

```
for n in range(1,5):
        d=wystapienia(s,n)
        l=list(map(lambda i:sum(i.values()),d.values()))
        print(n, ent(l)/n, ent_war(d))
        print(sym(d,500))
```

Algorytmy kompresji

Konstrukcja kodu Huffmana

Teraz traktujemy nasz ciąg znaków jako ciąg krotek n-znaków:

```
def nki(s,n):
        return [s[i*n:(i+1)*n] for i in range(len(s)//n)]
```

Obliczamy statystykę zliczeń dla tego ciągu:

```
def statystyka(s):
        d={}
        for i in s:
                if i in d:
                        d[i]+=1
                else:
                        d[i]=1
        return d
```

By skonstruować kod Huffmana, najpierw konstruujemy system identyfikacji wyników, w której każdy węzeł jest reprezentowany jako krotka długości 2, której elementami są dwa węzły potomne. Startujemy ze słownika zliczeń i jego listy kluczy. W każdym przebiegu pętli łączymy dwa elementy o najmniejszej liczbie zliczeń krotką je zawierającą (długość listy zmniejsza się o 1). Jednocześnie do słownika dopisujemy łączną liczbę zliczeń dla tej krotki. Postępujemy tak, aż lista skróci się do dwóch elementów:

```
def huff(d):
        l=d.keys()
        while len(l)>2:
                m1=min(l,key=d.get); l.remove(m1);
                m2=min(l,key=d.get); l.remove(m2);
                l.append((m1,m2)); d[(m1,m2)]=d[m1]+d[m2]
        return dict(kod(tuple(l)))
```

Po wyjściu z pętli konstruujemy słownik przypisujący każdemu elementowi słowo kodowe za pomocą funkcji rekurencyjnej:

```
def kod(l,pref=''):
        if type(l)==tuple:
                return kod(l[0],pref+'0')+kod(l[1],pref+'1')
        else:
                return [(l,pref)]
```

Teraz wygenerujmy ciąg bitów kodujący nasz ciąg znaków:

```
nki_list=nki(s,n)
h=huff(statystyka(nki_list))
dh=dict(kod('',dh))
c = ''.join(map(h.get,nki_list))
```

Wypiszmy, ile średnio bitów przypada na jeden znak:

```
print("Liczba␣bitów␣na␣znak:␣",len(c)/len(s))
```

Jest to ciąg malejący, ale z rosnącym n zwiększa się rozmiar słownika, który musimy wysłać wraz z naszymi danymi. Jego rozmiar to suma rozmiarów wartości plus suma rozmiarów kluczy:

```
print "Rozmiar␣słownika:␣",sum(map(len,h.values()))+sum(map(len,h.keys()))*8
```

Teraz możemy policzyć rzeczywistą (z uwzględnieniem narzutu) liczbę bitów na znak:

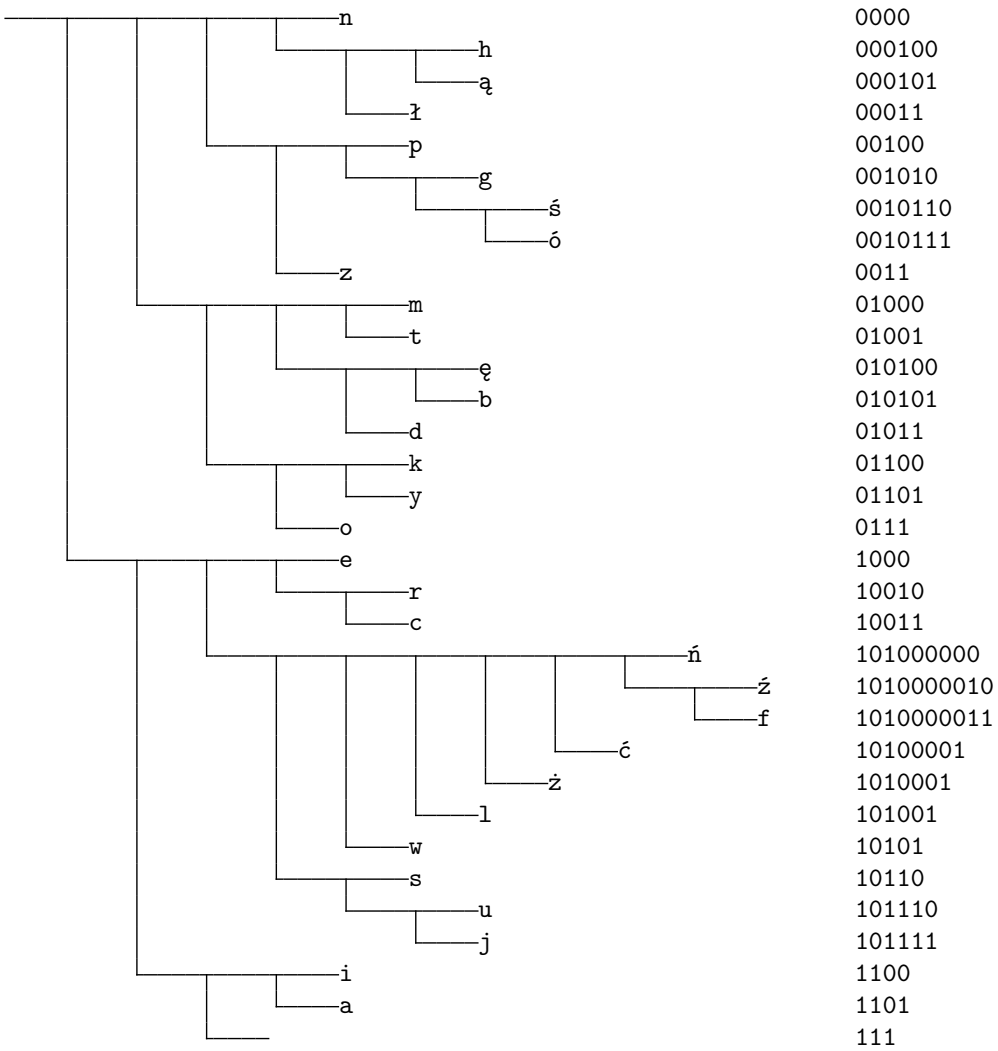
```
print "Liczba␣bitów␣na␣znak␣(z␣narzutem):␣", \
1.0*(len(c)+sum(map(len,h.values()))+sum(map(len,h.keys()))*8)/len(s)
```

Do narysowania reprezentacji graficznej systemu decyzyjnego odpowiadającego kodowi, użyjemy funkcji rekurencyjnej:

```
def drzewo(kod):
        N=4
        if len(kod)==1:
                return [N*chr(9472)+list(kod.keys())[0]]
        kod0={}
        kod1={}
        for k in kod.keys():
                if kod[k][0]=='0':
                        kod0[k]=kod[k][1:]
                else:
                        kod1[k]=kod[k][1:]
        l0=drzewo(kod0)
        l1=drzewo(kod1)
        return [N*chr(9472)+chr(9516)+l0[0]]
                +[N*'␣'+chr(9474)+i for i in l0[1:]]
                +[N*'␣'+chr(9492)+l1[0]]
                +[(N+1)*'␣'+i for i in l1[1:]
                ]

nki_list=nki(s,1)
d=statystyka(nki_list)
dh=huff(d)
dh=kod('',dh)
dh=dict(dh)
print('\n'.join(['%-60s␣%s' % i for i in zip(drzewo(dh),sorted(dh.values()))]))
```

otrzymamy w wyniku:



0000
000100
000101
00011
00100
001010
0010110
0010111
0011
01000
01001
010100
010101
01011
01100
01101
0111
1000
10010
10011
101000000
1010000010
1010000011
10100001
1010001
101001
10101
10110
101110
101111
1100
1101
111

Transformata Burrowsa-Wheelera

Dla ciągu znaków długości n:

1. Dodajemy unikalny znak końca ciągu
2. Wyznaczamy wszystkie rotacje i zapisujemy we wierszach macierzy $(n+1) \times (n+1)$
3. Sortujemy wiersze leksykograficznie
4. wynikiem jest ostatnia kolumna macierzy

Transformata jest odwracalna. Startujemy z macierzy pustej i $(n+1)$ krotnie powtarzamy kroki

1. Do istniejącej macierzy dodaj ciąg wejściowy jako kolumnę z lewej strony
2. Posortuj wiersze
3. wynikiem jest wiersz kończący się znakiem końca.

Transformata grupuje powtarzające się znaki blisko siebie. Przykład:

```
metacentrum jest punktem geometrycznym statku którego położenie jest ważne dla stat
eczności statku w statku siły ciężkości i wyporu przyłożone są do różnych punktów g
dy statek nie jest przechylony obie siły są przyłożone na jednej prostej prostopadł
ej do poziomu wody gdy statek przechyli się punkt w którym przyłożona jest siła wyp
oru zostaje przesunięty w kierunku tej burty na którą statek się przechylił linia p
ionowa prostopadła do poziomu wody i przechodząca przez punkt przyłożenia siły wypo
ru przecina się wtedy z osią statku punkt tego przecięcia to jest właśnie metacentr
um odległość od środka ciężkości do punktu metacentrycznego to wysokość metacentryc
zna gdy metacentrum znajduje się ponad środkiem ciężkości to wysokość metacentryczn
a jest dodatnia gdy poniżej  ujemna gdy wysokość metacentryczna jest dodatnia siły
wyporu i ciężkości sprowadzają statek do pionu gdy ujemna  statek może przechylić s
ię i zatonąć zadaniem konstruktorów statków jest obliczyć optymalną wartość wysokoś
ci metacentrycznej tak by statek nie był za sztywny i jego okres kołysań był optyma
lny do założonej nośności zgodnie z tymi obliczeniami wyprofilować kadłub statku

ajjkńeayimekjaiąyttaawyuamuyyięaiaawemaeoćwmsuwałeyćććuikeykkjyitćmołćzaoęyooojajik
eętuoeatmąuzuętohoćekiatueaaiaą_yyyąmąwibwaeyjtuaoizy_ytuątuuęyyiaićooytyełćioimudd
unkłninnnnnnniwcniilziłttttttttnzwppkntztmmidwttttttttttttttddzwzsłwuooo____ąaaaaaaaa
yeeeeeśśśśśśśęee____iyyyyyeyioaaoooo_oeo________joogggggggeaoaaainiiinżjjinizzzzzzzt
jtjrtnlżtłnntnttttttitijjżzżccccccccgirjjjjjjjjzmmmmmmmmmzo_______eeeezeccccccc_cc_c
m_lccc_m_cnnnncnllnnnnbnknkfclzzppslsssssscncccclsssssneeeeeeeeaau__uu________aaeeeee
ed_d___żżożżoooonnnnun___ntttntttddybb_yyaayieeueuyyuyy^_______oyaee_oomzm_zzo_iozż
oozodztitee__eśdauouuuuuuuozś_oeeeeeeeeolwożzlotgdddggddgttd____ddrr_gwwhrssss_eiiż
pżżżpikilttt__ppppt_zrrrnrlpppkkkkknknnkkkłtmłłłłłłoo____yyyy___y___s___________o
o______ókśśpppppauooootttttettttttóppppppppppppppo_óeyo__________yyyy_eeeeeeeeo______
_______ooone___ssssksskksseeeeeeeees_sssssssssssssaw__saaaaaakaaaaaaaa___asskrsnnnnn
nnnekrępp_zkkkknrktrrkkkkmmrkłd__rrrrsppppppprbóó___óoo_o_y___________łndnddtndbdd
łdtłłddnrrrrrrhhhhrntttwwwwwłwwwwtzbbzzzz_e___d__rrrrrrrcrr_ooccc_cccc_scrrrrdttttk
rrssjrinznaśśśśyiśąiiiiiiiiiiiiyydiwdgyoyyyadiiiioaooooooooao__ooooooiooęęęęaóoooo
```

Implementacja:

```
def BW(s):
        l=[s[i:]+s[:i] for i in range(len(s))]
        l=sorted(l)
        return ''.join([i[-1] for i in l])
def IBW(l,z):
        l1=l
        for i in range(len(l)-1):
                l1=list(map(''.join,zip(l,sorted(l1))))
        l1=l1[0]
        z=l1.index(z)
        return l1[z+1:]+l1[:z]
```

Move to front

Transformata Burrowsa-Wheelera jedynie przestawia znaki nie zmieniając ich statystyki, zatem kod Huffmana dla tego ciągu będzie miał dokładnie taką samą średnią długość słowa kodowego. Zakładamy, że znaki alfabetu mają reprezentację binarną określonej długości. Ideą jest, by kodować nie sam przebieg, ale jego pochodną (w przypadku dyskretnym jest to ciąg różnic). Teraz różnice kodów znaków o wiele częściej będą wynosiły 0 i na poziomie różnic średnia długość słowa kodowego się zmniejszy.

Dla naszego przykładu:

- Entropia tekstu: 4.47
- Entropia pochodnej tekstu: 4.96
- Entropia pochodnej tekstu po transformacie: 3.67 - przesunięcie 0 stanowi teraz prawie połowę wszystkich przesunięć.

Implementacja:

```
znaki="aąbcćdeęfghijklłmnńoóprsśtuwyzźż␣^"

def mtf(s):
        return [znaki.index(s[0])]+[(znaki.index(s[i])-znaki.index(s[i-1])) % len(znaki)
        for i in range(1,len(s))]

def imtf(l):
        i=0
        s=''
        for j in l:
                i=(i+j) % len(znaki)
                s+=znaki[i]
        return s
```

Run-length encoding

Dalej w naszym ciągu zera stoją w długich ciągach. Wprowadzając do alfabetu dwa dodatkowe znaki specjalne: 0, 1 wprowadzamy umowę: jeżeli zero powtarza się, zapisujemy binarnie liczbę jego wystąpień za pomocą dodatkowych znaków (pierwsza cyfra tej liczby napewno jest 1, więc jej nie piszemy).

Implementacja:

```
def RLE(s,z0,z1): #dwa znaki specjalne
        s=list(s)
        l=[s[0:1]]
        for i in range(1,len(s)):
                if s[i]!=l[-1][-1]:
                        l+=[s[i:i+1]];
                else:
                        l[-1]+=s[i:i+1]
        l=[i if len(i)<2 and i[0]!=0 else [z1 if j=='1' else z0 for j in '{0:b}'.format(len(i))[1:]]
           for i in l]
        return sum(l,[])
```

Pozwala nam to zejść z entropią na znak do 3.27 bita.

Algorytm bzip2

Algorytm bzip2 składa się z następujących kroków:

1. Dla bloków ustalonej długości wykonujemy: transformatę BW, MTF, RLE
2. Dla bloku wyznaczamy kod Huffmana.

Algorytmy Lempel-Ziv

LZ77: Dla znaku tekstu, szukamy najdłuższego ciągu (<długość maksymalna) zaczynającego się od tego znaku w poprzednich 32 kB. Jeżeli znajdziemy, to kodujemy go jako para (odległość, długość). Jeżeli nie, kodujemy go jako (0,znak):

`ABRAKADABRA -> (0,A)(0,B)(0,R)(3,1)(0,K)(2,1)(0,D)(7,2)(7,2)`

LZ78: Tniemy ciąg na najkrótsze podciągi, które jeszcze nie wystąpiły:

`ABRAKADABRA -> A,B,R,AK,AD,AB,RA`

następnie kodujemy każdy ciąg jako pozycja prefiksu + ostatni znak: `A,B,R,AK,AD,AB,RA -> (0,A)(0,B)(0,R)(3,K)(4,D)(5,B)(4,A)`

Algorytmy te są optymalne - dla procesu stochastycznego o wartościach w alfabecie asymptotycznie osiągają one entropię procesu. Algorytmy LZ są używane m.in w gzip, png, gif

Twierdzenie (Lemat Kaca): Niech $\{U_i\}_{i\in\mathbb{Z}}$ będzie stacjonarnym ergodycznym[1] procesem stochastycznym o wartościach w przeliczalnym alfabecie. Jeżeli $U_0 = u$ i $p(u) > 0$, to średni czas oczekiwania na ponowne pojawienie się tego samego znaku $T(u)$ (średni czas powrotu) wynosi $1/p(u)$.

Dowód: Wybierzmy pewne u z alfabetu, które pojawia się ze skończonym prawdopodobieństwem $p(u)$. Dla $j \geqslant 1, k \geqslant 0$ zdefiniujmy zdarzenie $A_{jk} = \{U_{-j} = U_k = u \wedge \forall l \in (-j, k)\ U_l \neq u\}$. Zauważmy, że zdarzenia te są rozłączne, a ich sumą jest zdarzenie polegające na wystąpieniu przynajmniej jednego u na lewo od 0 i przynajmniej jednego u na prawo od 0. Jest to zdarzenie pewne. Mamy zatem:

$$1 = \Pr\left(\bigcup_{j,k} A_{jk}\right) = \sum_{j=1}^{\infty}\sum_{k=0}^{\infty}\Pr\left(A_{jk}\right) = \sum_{j=1}^{\infty}\sum_{k=0}^{\infty}\Pr\left(U_{-j} = U_k = u\ \wedge\ \forall l \in (-j,k)\ U_l \neq u\right)$$

na mocy stacjonarności procesu:

$$= \sum_{j=1}^{\infty}\sum_{k=0}^{\infty}\Pr\left(U_{j+k} = U_0 = u\ \wedge\ \forall l \in (0, j+k)\ U_l \neq u\right)$$

$$= \sum_{j=1}^{\infty}\sum_{k=0}^{\infty}\Pr\left(U_0 = u\right)\Pr\left(U_{j+k} = u\ \wedge\ \forall l \in (0, j+k)\ U_l \neq u | U_0 = u\right).$$

Zdarzenie $\{U_{j+k} = u\ \wedge\ \forall l \in (0, j+k)\ U_l \neq u | U_0 = u\}$ to pierwszy powrót do wartości początkowej po czasie dokładnie $j+k$, przy warunku że tą wartością jest u. Oznaczmy je jako Q_{j+k}. Mamy zatem:

$$1 = \sum_{j=1}^{\infty}\sum_{k=0}^{\infty} p(u) Q_{j+k} = p(u)\sum_{i=1}^{\infty} i Q_i = p(u)T(u).$$

Zatem $T(u) = 1/p(u)$ $\square$

Lemat: Dowolnie dużą liczbę naturalną n można zakodować na $\log_2 n + 2\log_2(\log_2 n + 1) + 4$ bitach.

Dowód: Przed przesłaniem liczby n w zapisie binarnym trzeba wysłać wiadomość o tym, ile bitów ma zostać odczytane ($k = \lceil \log_2 n \rceil$). Liczbę k (również nieograniczoną) kodujemy w następujący sposób: $0 \ldots 01$ + zapis binarny liczby. Liczba zer jest równa liczbie bitów w zapisie liczby k. Liczba k jest zakodowana na $2\lceil \log_2 k \rceil + 1 \leqslant 2\log_2 k + 3$ bitach, zatem do zakodowania liczby n potrzebujemy najwyżej $\log_2 n + 2\log_2(\log_2 n + 1) + 4$ bitów $\square$

Wykres funkcji wypukłej $\mathbb{R} \to \mathbb{R}$ leży pod sieczną łączącą punkty. Podobny fakt zachodzi dla funkcji wielu zmiennych. Formalizujemy do jako nierówność Jensena: Dla funkcji wypukłych $f(\sum_i p_i x_i) \leqslant \sum_i p_i f(x_i)$, dla funkcji wklęsłych $f(\sum_i p_i x_i) \geqslant \sum_i p_i f(x_i)$.

Lemat: Średnia liczba bitów potrzebnych do zakodowania n kolejnych znaków w procesie stochastycznym $\{X_i\}_{i\in\mathbb{Z}}$ nie przekracza $H(X_1, \ldots, X_n) + 2\log_2(H(X_1, \ldots, X_n) + 1) + \log_2 n + 5$

Dowód: Dla ciągu $\vec{x}$ długości n oczekiwane przesunięcie do miejsca gdzie taki sam ciąg wystąpił w przeszłości wynosi $1/p(\vec{x})$, zatem średnia liczba bitów potrzebna do zakodowania tego przesunięcia nie przekracza $\log_2 \frac{1}{p(\vec{x})} + 2\log_2\left(\log_2 \frac{1}{p(\vec{x})} + 1\right) + 4$. Uśredniając ten wynik po wszystkich możliwych ciągach dostaniemy $\sum_{\vec{x}} p(\vec{x}) \log_2 \frac{1}{p(\vec{x})} + 2\sum_{\vec{x}} p(\vec{x}) \log_2 \log_2\left(\frac{1}{p(\vec{x})} + 1\right) + 4 \leqslant \sum_{\vec{x}} p(\vec{x}) \log_2 \frac{1}{p(\vec{x})} + 2\log_2 \sum_{\vec{x}} p(\vec{x})\left(\log_2 \frac{1}{p(\vec{x})} + 1\right) + 4 = H(X_0, \ldots, X_n) + 2\log_2(H(X_0, \ldots, X_n) + 1) + 4$ (oszacowanie z nierówności Jensena).

[1] proces ergodyczny to proces, w którym każda, odpowiednio długa realizacja procesu ma te same własności statystyczne. Założenie to nam jest potrzebne, by można było dobrze zdefiniować prawdopodobieństwo wystąpienia znaku

Do tego dochodzi $\lceil \log_2 n \rceil \leqslant \log_2 n + 1$ bitów na zakodowanie długości $\square$

Wniosek: Gdy rozmiar okna w algorytmie LZ77 rośnie, wtedy kodujemy w postaci (przesunięcie długość) coraz dłuższe ciągi. W granicy $n \to \infty$ dostajemy na znak:

$$\lim_{n\to\infty} \frac{1}{n} \left(H(X_1, \ldots, X_n) + 2 \log_2 H(X_1, \ldots, X_n) + \log_2 n + 5 \right) = \lim_{n\to\infty} \frac{1}{n} H(X_1, \ldots, X_n)$$

czyli do entropii procesu. Algorytm LZ77 jest asymptotycznie optymalny.

Algorytm LZ78 tworzy ciąg najkrótszych sekwencji, które nie wystąpiły wcześniej. Poniższy lemat daje ograniczenie na ich ilość:

Lemat: Rozbijamy ciąg n znaków nad alfabetem m-znakowym na parami różne podciągi. Ilość elementów w rozbiciu $c(n)$ nie przekroczy $\frac{n}{\log_m n(1-\delta_n)}$, gdzie $\delta_n = \frac{2+\frac{1}{m-1}-\log_m(m-1)+\log_m(\log_m n+1)}{\log_m n}$ jest ciągiem zbieżnym z góry do 0.

Dowód: Wszystkich sekwencji o długości nie przekraczającej k jest $\sum_{j=1}^{k} m^j = \frac{m}{m-1}\left(m^k - 1\right)$, a ich łączna długość wynosi $\sum_{j=1}^{k} j m^j = \frac{m}{m-1}\left(m^k\left(k - \frac{1}{m-1}\right) + \frac{1}{m-1}\right)$. Liczba $c(n)$ osiąga maksimum, jeżeli w rozkładzie wykorzystujemy wszystkie najkrótsze dostępne sekwencje. Dla $n = n_k$ mamy ograniczenie:

$$c(n_k) = \frac{m}{m-1}\left(m^k - 1\right) < \frac{m^{k+1}}{m-1} < \frac{n_k}{k - \frac{1}{m-1}}$$

Biorąc dowolne $n \in [n_k, n_{k+1})$ pierwsze n_k znaków rozbijamy na sekwencje długości $\leqslant k$, a kolejne Δ znaków rozbijamy na $\lfloor \Delta/(k+1) \rfloor$ sekwencji długości $k+1$ (ostatnia z nich musi być dłuższa). Mamy ograniczenie na liczbę sekwencji:

$$c(n) = c(n_k) + \lfloor \Delta/(k+1) \rfloor < \frac{n_k}{k - \frac{1}{m-1}} + \frac{\Delta}{k+1} \leqslant \frac{n_k + \Delta}{k - \frac{1}{m-1}} = \frac{n}{k - \frac{1}{m-1}} \tag{2}$$

Szacujemy liczbę n:

$$\begin{aligned} \frac{m}{m-1}\left(m^k\left(k - \frac{1}{m-1}\right) + \frac{1}{m-1}\right) = n_k \quad &\leqslant n < \quad n_{k+1} = \frac{m}{m-1}\left(m^{k+1}\left(k+1 - \frac{1}{m-1}\right) + \frac{1}{m-1}\right) \\ m^k \quad &\leqslant n < \quad \frac{m^{k+2}}{m-1}(k+1) < \frac{m^{k+2}}{m-1}(\log_m n + 1) \end{aligned}$$

Dostajemy stąd $k + 2 \geqslant \log_m \frac{n(m-1)}{\log_m n + 1} \Rightarrow k - \frac{1}{m-1} \geqslant \log_m \frac{n(m-1)}{\log_m n+1} - 2 - \frac{1}{m-1} =$
$\log_m n \left(1 - \frac{2+\frac{1}{m-1}-\log_m(m-1)+\log_m(\log_m n+1)}{\log_m n}\right) = \log_m n(1-\delta_n)$. Pozwala to oszacować mianownik (2):

$$c(n) < \frac{n}{\log_m n(1-\delta_n)} \qquad \square$$

Niech s będzie ciągiem znaków długości k. Niech $c_{l,s}(n)$ oznacza liczbę sekwencji poprzedzonych ciągiem s i o długości l. Mamy $\sum_{l,s} c_{l,s}(n) = c(n)$, $\sum_{l,s} l c_{l,s}(n) = n$. Mamy następującą nierówność

Lemat (nierówność Ziv): $\log_2 Q_k(x_1, \ldots, x_n | s_1) \leqslant -\sum_{l,s} c_{l,s} \log_2 c_{l,s}$.

Dowód: Ciąg znaków $x_1, \ldots x_n$ zapisujemy jako ciąg sekwencji $y_1, \ldots, y_{c(n)}$. Zachodzi
$Q_k(x_1, x_2, \ldots, x_n | s_1) = Q_k(y_1, \ldots, y_{c(n)} | s_1) = \prod_{i=1}^{c(n)} P(y_i | s_i)$ (bo pamięć w k-tym przybliżeniu markowowskim nie sięga dalej niż k poprzedzających znaków), zatem

$$\log_2 Q_k(x_1, \ldots, x_n | s_1) = \sum_{i=1}^{c(n)} \log_2 P(y_i | s_i) = \sum_{l,s} \sum_{i:|y_i|=l, s_i=s} \log_2 P(y_i | s_i)$$

$$= \sum_{l,s} c_{l,s} \sum_{i:|y_i|=l,s_i=s} \frac{1}{c_{l,s}} \log_2 P(y_i|s_i) \leqslant \sum_{l,s} c_{l,s} \log_2 \frac{\sum_{i:|y_i|=l,s_i=s} P(y_i|s_i)}{c_{l,s}} \leqslant -\sum_{l,s} c_{l,s} \log_2 c_{l,s} \quad \square$$

Lemat: Jeżeli dla zmiennej losowej U mamy $\mathbb{E}(U) = \mu$, to $H(U) \leqslant (\mu+1)\log_2(\mu+1) - \mu \log_2 \mu = \mu((1+\mu^{-1})\log_2(1+\mu^{-1}) - \mu^{-1}\log_2 \mu^{-1})$.

Dowód: W dalszej części wykładu.

Twierdzenie: Gdy $n \to \infty$ ilość bitów potrzebnych do zakodowania znaku w algorytmie LZ78 zbiega do entropii procesu.

Dowód:

$$-\frac{1}{n} \log_2 Q_k(x_1, \ldots, x_n|s_1) \geqslant \frac{c}{n} \sum_{l,s} \frac{c_{l,s}}{c} \log_2 \frac{c_{l,s}}{c} + \frac{c}{n} \log_2 c \tag{3}$$

$c_{l,s}/c$ to łączny rozkład prawdopodobieństwa zmiennych losowych U i V, gdzie wartością pierwszej jest długość sekwencji, a wartością drugiej jest ciąg długości k poprzedzający sekwencję. Pierwszy wyraz drugiej strony to $-\frac{c}{n}H(U,V)$. Ponieważ znamy $\mathbb{E}U = n/c$, a o drugiej zmiennej nie wiemy nic, możemy oszacować ich łączną entropię::

$$\begin{aligned} \frac{c}{n}H(U,V) &\leqslant \frac{c}{n}\left(k\log_2 m + \left(1+\frac{c}{n}\right)\log_2\left(1+\frac{c}{n}\right) - \frac{c}{n}\log_2\frac{c}{n}\right) \\ &\leqslant \frac{c}{n}\left(k\log_2 m + \left(1+\frac{c}{n}\right)\frac{c}{n}\log_2 e - \frac{c}{n}\log_2\frac{c}{n}\right) \end{aligned}$$

Pamiętamy, że c/n zachowuje się jak $1/\log_m n$, zatem $\epsilon_n = \frac{c}{n}H(U,V)$ jest ciągiem zbieżnym z góry do 0. Z (3) mamy:

$$\begin{aligned} \frac{c(n)\log_2 c(n)}{n} &\leqslant -\frac{1}{n}\log_2 Q_k(x_1,\ldots,x_n|s_1) + \epsilon_n \\ \limsup_{n\to\infty} \frac{c(n)\log_2 c(n)}{n} &\leqslant \lim_{n\to\infty} -\frac{1}{n}\log_2 Q_k(x_1,\ldots,x_n|s_1) = H(X_0|X_{-1},\ldots,X_{-k}) \overset{k\to\infty}{\longmapsto} H(X) \end{aligned}$$

Każda sekwencja jest kodowana za pomocą pary (pozycja sufiksu w ciągu przeszłych sekwencji, ostatni znak sekwencji). Do zakodowania pozycji potrzeba najwyżej $\log_2 c(n) + 2\log_2(\log_2 c(n)+1) + 4$ bitów, do zakodowania znaku $\log_2 m$ bitów. W przeliczeniu na znak przy kodowaniu n znaków potrzebujemy

$$\frac{c(n)}{n}(\log_2 c(n) + 2\log_2(\log_2 c(n)+1) + 4 + \log_2 m)$$

bitów. W granicy przeżywa tylko pierwszy wyraz, a jak pokazaliśmy ta granica jest równa entropii procesu $\square$

Rozkłady ciągłe

Entropia rozkładu ciągłego

Dla zmiennej losowej X o rozkładzie ciągłym o gęstości f (na początek na zwartym nośniku) entropię definiujemy jako: $h(X) = h(f) = -\int f(x)\log_2 f(x)dx$.

Analogicznie definiujemy entropię warunkową i informację wzajemną:

$$\begin{aligned} h(X|Y) &= -\int f(x,y)\log_2 \frac{f(x,y)}{f(y)} = h(X,Y) - h(Y) \\ I(X:Y) &= \int f(x,y)\log_2 \frac{f(x,y)}{f(y)f(x)} = h(Y) + h(X) - h(X,Y) \geqslant 0 \end{aligned}$$

Przykład: Entropia rozkładu równomiernego na odcinku o długości a wynosi $\log_2(a)$.

Zauważmy, że wartość entropii może być ujemna! - ponieważ nośnik może być dowolnie wąski, nie istnieje rozkład pewny do którego można by wycechować entropię.

Fakt: Jest to rozkład maksymalizujący entropię wśród funkcji o zadanym nośniku.

Dowód: Szukamy ekstremum funkcjonału $H(f) = -\int_a^b f(x)\log_2 f(x)dx$ przy warunku $\int_a^b f(x)dx = 1$.

Zmienną losową możemy potraktować jako kombinację liniową dwóch zmiennych losowych: $X = i\Delta + \delta x$, gdzie Δ jest szerokością przedziału duskretyzacji, i - numerem przedziału dyskretyzacji, oraz $\delta x \in [0, \Delta)$. Zachodzi $H(X) = H(i) + H(\delta x) \leqslant H(i) + \log_2 \Delta = \sum_i \Delta f(i\Delta)\log f(i\Delta)$. Gdy $\Delta \to 0$, zgodnie z definicją całki Riemanna prawa strona zbiega z góry do strony lewej.

Przykład: Entropia rozkładu Gaussa o wariancji σ: $\frac{1}{2}\log(2\pi e\sigma)$.

Wniosek: Dla wielowymiarowego rozkładu Gaussa o macierzy kowariancji K entropia wynosi $\frac{1}{2}\log((2\pi e)^n \det K)$.

Fakt: Dla zmiennych o ciągłym rozkładzie zachodzi: $h(X + a) = h(X), h(aX) = h(X) + \log_2 a$.

Twierdzenie: Dla dowolnej zmiennej losowej X o $\mathbb{E}X^2 < \sigma$ zachodzi $h(X) \leqslant \frac{1}{2}\log_2(2\pi e)\sigma^2$, a równość zachodzi tylko dla rozkładu Gaussa o wariancji σ^2.

Dowód: Szukamy ekstremum funkcjonału $H(f) = -\int_a^b f(x)\log_2 f(x)dx$ przy warunkach $\int_a^b f(x)dx = 1$, $\int_a^b x^2 f(x)dx = 1$.

Wniosek: Podobnie, dla wielowymiarowej zmiennej losowej o macierzy kowariancji K maksimum entropii jest osiągane dla rozkładu Gaussa o macierzy kowariancji K.

Twierdzenie Wienera-Chinczyna

Definicja: Funkcja autokorelacji dla stacjonarnego procesu ergodycznego X_t:

$$C(t) \stackrel{df}{=} \mathbb{E}\langle X_0 | X_t \rangle$$

Definicja: Spektralna gęstość mocy (widmo sygnału) dla stacjonarnego procesu stochastycznego X_t:

$$P(f) \stackrel{df}{=} \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left|\int_{-T/2}^{T/2} x(t)e^{-i2\pi ft}dt\right|^2,$$

gdzie $x(t)$ jest realizacją procesu X_t

Twierdzenie Wienera-Chinczyna funkcja autokorelacji sygnału jest transformatą Fouriera widmowej gęstości mocy sygnału.

Dowód:

$$\begin{aligned}\mathbb{E}\left|\int_{-T/2}^{T/2} x(t)e^{-i2\pi ft}dt\right|^2 &= \int_{-T/2}^{T/2}\int_{-T/2}^{T/2} \mathbb{E}x(t)x(s)e^{-i2\pi f(t-s)}dtds = \int_{-T/2}^{T/2}\int_{-T/2}^{T/2} C(t-s)e^{-i2\pi f(t-s)}dtds \\ &= \int_{-T}^{T}(T-|u|)C(u)e^{-i2\pi fu}du\end{aligned}$$

dzieląc obie strony przez T i wykonując granicę $T\to\infty$ dostajemy

$$\mathbb{E}\left|\int_{-\infty}^{\infty} x(t)e^{-i2\pi ft}dt\right|^2 = \int_{-\infty}^{\infty} C(u)e^{-i2\pi f(u)}du = C(f) \quad \square$$

Kanały Gaussowskie

Kanał Gaussowski to kanał dany wzorem $Y = X + Z$, gdzie Z jest szumem o rozkładzie $\mathcal{N}(0,\sigma_Z)$. Rozważać będziemy tylko jednowymiarowe kanały Gaussowskie. W przypadku wielowymiarowym należy zastąpić wariancję macierzą kowariancji.

Jeżeli zmienna X jest nieograniczona, przepustowość kanału jest nieskończona. Dlatego będziemy liczyć przepustowość przy ograniczeniu na moc sygnału: $\mathbb{E}X^2 = \frac{1}{n}\sum_i x_i^2 < P$, gdzie P oznacza maksymalną moc (energia na jednostkę czasu).

$$I(X:Y) = h(Y) - h(Y|X) = h(Y) - h(Z|X) = h(Y) - h(Z)$$

Wariancja $\sigma_Y = \sigma_X + \sigma_Z$ (ponieważ są to zmienne niezależne) i stąd $h(Y) \leqslant \frac{1}{2}\log_2(2\pi e)(\sigma_X+\sigma_Z)$, zatem $I(X:Y) \leqslant \frac{1}{2}\log_2(2\pi e)(\sigma_X+\sigma_Z) - \frac{1}{2}\log_2(2\pi e)\sigma_Z = \frac{1}{2}\log_2(1+\frac{\sigma_X}{\sigma_Z})$, a równość jest osiągana dla sygnału wejściowego o rozkładzie Gaussa i maksymalnej dopuszczalnej mocy. Przepustowość kanału Gaussa wynosi:

$$\frac{1}{2}\log_2(1+SNR),$$

gdzie SNR oznacza stosunek mocy sygnału do mocy szumów.

Kanał z białym szumem o ograniczonym paśmie

Załóżmy, że przysyłamy informację kanałem ciągłym w czasie. W naszym kanale dodaje się szum gaussowski, którego spektralna gęstość mocy jest stała (szum biały) i wynosi N_0. W kanale są filtrowane wszystkie częstotliwości powyżej pewnej częstotliwości granicznej W.

Na podstawie twierdzenia Kotelnikowa, wartość funkcji jest całkowicie wyznaczona przez jej wartości próbkowane z częstotliwością $2W$. Częstsze próbkowanie dawałoby zależności między próbkami,

rzadsze prowadziło do aliasingu. Szum biały ograniczony do pasma szerokości W ma funkcję autokorelacji proporcjonalną do $\mathrm{sinc}(2\pi Wt)$, zatem jeżeli próbkujemy z częstotliwością $2W$, każda próbka podlega niezależnemu szumowi Gaussa. W jednostce czasu mamy zatem $2W\Delta t$ niezależnych kanałów Gaussowskich (kolejne próbki). Na każdą próbkę przypada $P/(2W\Delta t)$ mocy i $N_0/(2\Delta t)$ mocy szumów. (Dlaczego optymalnie jest podzielić całkowitą moc równo pomiędzy próbki?)

Mamy zatem $2W\Delta t$ niezależnych kanałów Gaussowskich połączonych równolegle. Całkowita przepustowość wynosi:

$$2W\Delta t\frac{1}{2}\log_2\left(1+\frac{\frac{P}{2W\Delta t}}{\frac{N_0}{2}}\right)=W\Delta t\log_2\left(1+\frac{P}{N_0W}\right)$$

a przepustowość na jednostkę czasu wynosi

$$W\log_2\left(1+\frac{P}{N_0W}\right)\left[\frac{\text{bit}}{\text{s}}\right]$$

W przypadku małego SNR (np. szerokie pasmo), możemy wielkość tę przybliżyć jako:

$$\log_2 e\frac{P}{N_0}$$

W przypadku dużego SNR (np. wąskie pasmo), możemy wielkość tę przybliżyć jako:

$$W\log_2\left(\frac{P}{N_0W}\right)=W\log_2 10\log_{10}SNR\approx 3.32W\log_{10}SNR\ \left[\frac{\text{bit}}{\text{s}}\right]$$

Przykład: SNR dla linii telefonicznej wynosi 33.1 dB. Szerokość pasma wynosi 3.3 kHz. Przepustowość wynosi 36.26 kb/s.

Kanały Gaussowskie z szumem kolorowym

Załóżmy teraz że dzielimy pasmo na n podpasm o równej szerokości W, a szum Gaussa nie jest szumem białym - ma niepłaskie widmo, a co za tym idzie różną od zera funkcję autokorelacji dla próbek w różnych chwilach czasu. W i-tym podpaśmie przesyłamy w jednostce czasu $2W$ próbek kanałem Gaussowskim o wariancji szumu σ_i^2. Powstaje pytanie, jak optymalnie podzielić moc pomiędzy podpasma i jaką przepustowość jesteśmy w stanie uzyskać.

Podobnie jak dla pojedyńczego kanału Gaussowskiego liczymy $I(\vec{X},\vec{Y})=h(\vec{Y})-h(\vec{Z})$ $\leqslant\frac{1}{2}\log_2((2\pi e)^n\det(K_X+K_Z))-\frac{1}{2}\log_2((2\pi e)^n\det K_Z)$, przy czym równość jest osiągana gdy X ma rozkład Gaussa. Chcemy znaleźć K_X które maksymalizuje $\det(K_X+K_Z)$ przy warunku $\mathrm{Tr}K_X\leqslant P$ (ograniczenie na łączną moc sygnału).

Lemat: Wyrażenie $\det(A+B)$ osiąga maksimum przy warunku $\mathrm{Tr}A=const$ dla $A+B\sim I$.

Wniosek: Przepustowość kanału osiąga maksimum, gdy suma macierzy kowariancji sygnału i szumu jest proporcjonalna do identyczności (lub gdy moc na to nie pozwala, jej maksymalny blok diagonalny).

Gdy dzielimy pasmo na coraz węższe podpasma, gęstość spektrum macierzy kowariacji szumu zmierza do gęstości mocy szumu w paśmie. Optymalnie podzielona moc sygnału pomiędzy podpasma powstaje z "zalewania objętości nad wykresem mocą do równego poziomu".

Silnik Szilárda

Na początek przypomnijmy jedno ze sformułowań drugiej zasady termodynamiki:

II zasada termodynamiki w sformułowaniu Kelvina: *Nie jest możliwy proces odwracalny, którego jedynym skutkiem byłoby pobranie pewnej ilości ciepła ze zbiornika i zamiana go w równoważną ilość pracy (perpetum moblie II rodzaju)*

Policzmy, ile pracy wykona gaz rozprężający się dwukrotnie w przemianie izotermicznej:

$$W^G = \int_V^{2V} pdV = Nk_BT \int_V^{2V} \frac{dV}{V} = Nk_BT \ln 2,$$

Rozważmy cylinder z jednym atomem gazu, który po obu stronach ma tłoki z popychaczami, a w środku opuszczaną przegródkę. Załóżmy, że mamy informację w której połowie cylindra znajduje się atom. Wtedy opuszczamy przegródkę i bez nakładu pracy przesuwamy jeden z tłoków do środka cylindra. Podnosimy przegródkę i gaz się rozpręża, wykonując pracę $k_BT \ln 2$ kosztem ciepła pobranego z rezerwuaru. Jeżeli będziemy to powtarzać, złamiemy II zasadę termodynamiki. Istotę obserwującą położenie atomu i sterującą przegródką i tłokami nazywamy demonem Maxwella.

Przemiany energii związane ze zmianą informacji o układzie

Rozważmy układ dwupoziomowy, o dwóch poziomach o równej energii rozdzielonych barierą energetyczną. Jeżeli nie znamy jego stanu, oba poziomy mają prawdopodobieństwa równe $\frac{1}{2}$. Rozważmy operację ustawienia go w stanie 0 niezależnie od tego w jakim stanie był na początku. Taka operacja byłaby nieodwracalna. Ponieważ na poziomie mikrostanów fizyka jest odwracalna, żeby to zrealizować sprzęgniemy nasz układ z innym układem dwupoziomowym (otoczeniem) w stanie 0. Jeżeli nic nie wiemy o naszym układzie, stan obu układów będzie równy: $\begin{array}{|c|c|}\hline \frac{1}{2} & 0 \\ \hline \frac{1}{2} & 0 \\ \hline \end{array}$. Stan naszego układu uzyskujemy wysumowując wartości we wierszach, stan otoczenia uzyskamy wysumowując wartości w kolumnach. Wykonajmy teraz transpozycję tej macierzy. Jest to kanał ekstremalny, permutujący stany czyste ($00 \to 00$, $01 \to 10$, $10 \to 01$, $11 \to 11$) , zatem odwracalny. Przekształci on stan układu złożonego w: $\begin{array}{|c|c|}\hline \frac{1}{2} & \frac{1}{2} \\ \hline 0 & 0 \\ \hline \end{array}$. Stanem układu będzie 0, a stan otoczenia będzie maksymalnie mieszany. Na poziomie mikroskopowym wykonanie operacji w której tracimy bit informacji o układzie jest związany ze zwiększeniem entropii rezerwuaru o 1 bit. Entropia termodynamiczna rezerwuaru w stanie Gibbsa wynosi:

$$S = nk_B \left(\ln Z + \beta \bar{E}\right) = nk_B \left(\ln Z - \beta \partial_\beta \ln Z\right) \tag{4}$$

Oznacza, to że przyrost entropii rezerwuaru (wymieniamy z nim energię tylko poprzez przepływ ciepła, nie poprzez wykonywanie pracy) wynosi:

$$dS = k_B \beta d(n\bar{E}) = dQ_D/T \tag{5}$$

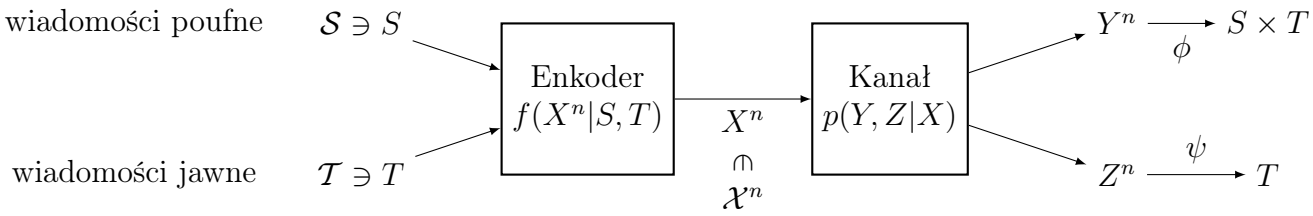
Przyrost entropii Shannona rezerwuaru o 1 bit odpowiada wzrostowi jego entropii termodynamicznej o $k_B \ln 2$, co oznacza przepływ do rezerwuaru ciepła $k_BT \ln 2$. W ten sposób udowadniamy zasadę Landauer'a:

Zasada Landauer'a: Operacja nieodwracalna, w której tracimy bit informacji o układzie sprzężonym z rezerwuarem o temperaturze T powoduje rozproszenie $k_BT \ln 2$ ciepła.

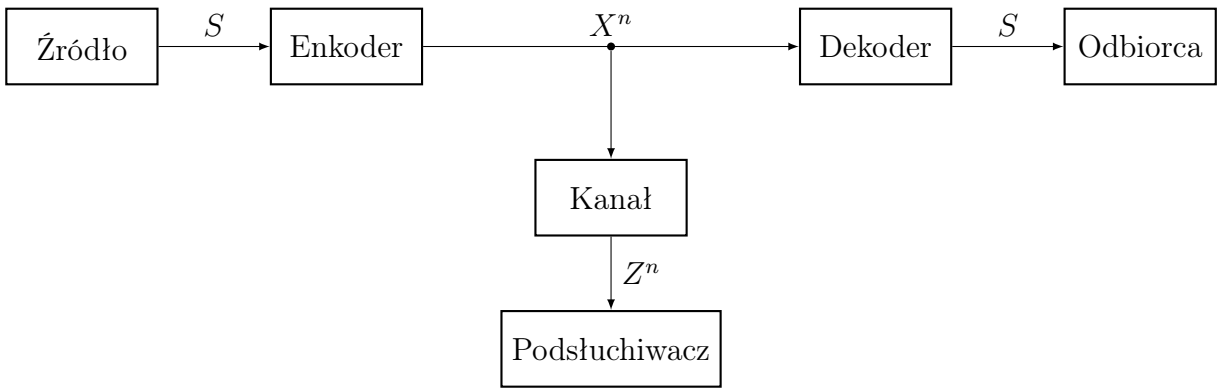
Zasada Landauer'a tłumaczy paradoks demona Maxwella - demon startuje z rejestrem pamięci ustawionym na 0. Pozyskując informację w jednym przypadku na dwa zmienia stan rejestru bez wkładu energii, a żeby proces był odwracalny na koniec musi usunąć wszelkie ślady informacji o poprzednim przebiegu procesu. W tym celu zeruje swój jednobitowy rejestr (niezależnie od tego co było tam zapisane) i rozprasza do otoczenia $k_B T \ln 2$ ciepła. Ponieważ energia wewnętrzna rejestru pozostaje bez zmian, odbywa się to kosztem pracy dostarczonej do układu. W ten sposób suma ciepła wymienionego z układem podczas cyklu i całkowita praca w cyklu wynoszą zero.

Przesyłanie wiadomości poufnych w kanałach

Załóżmy, że mamy dwa zbiory wiadomości: zbiór $\mathcal{S}$ wiadomości poufnych i zbiór $\mathcal{T}$ wiadomości jawnych. Załóżmy, że kodujemy parę $S \times T$ takich wiadomości jako słowo długości n nad alfabetem $\mathcal{X}$. Kodowanie może przypisywać jednej parze więcej niż jedno słowo, w sposób losowy. Enkoder charakteryzuje macierz prawdopodobieństw warunkowych $f(X^n|S,T)$. Słowo następnie przesyłane jest kanałem o dwóch wyjściach: Y i Z. Kanał charakteryzuje macierz prawdopodobieństw warunkowych $p(Y^n, Z^n|X^n)$. Każde wyjście ma dekoder: $\phi : \mathcal{Y}^n \to \mathcal{S} \times \mathcal{T}$, $\psi : \mathcal{Z}^n :\to \mathcal{S}$ - odbiorca wyjścia Y jest uprawniony do odczytu wiadomości poufnej, drugi odbiorca powinien odczytywać bezbłędnie tylko wiadomość jawną. Nie ma sensu rozważać stochastycznych dekoderów. Dekodery są deterministyczne.



W szczególnym przypadku kanał może być *zdegradowany* (— wyjście Z to wyjście Y podsłuchane za pomocą innego kanału) i przesyłamy tylko wiadomość poufną. Wtedy diagram redukuje się do:



Mówimy, że trójka (f, ϕ, ψ) umożliwia transmisję z błędem nie przekraczającym ϵ, jeżeli dla pary (S,T) prawdopodobieństwo prawidłowego odczytu wyjść (S,T) i T przekracza $1-\epsilon$. Określamy „poziom niewiedzy drugiego odbiorcy o wiadomości poufnej" poprzez $H(S|Z^n)$. Liczności obu zbiorów wiadomości możemy zdefiniować poprzez ich współczynniki transmisji: $\#\mathcal{S} = 2^{nR_1}$, $\#\mathcal{T} = 2^{nR_2}$. Chcemy przy tym, by $H(S|Z^n) \geqslant R_e$. Jesteśmy zainteresowani jednoczesną maksymalizacją tych trzech liczb. Chcemy scharakteryzować zbiór $\mathcal{R}$ dopuszczalnych trójek (R_1, R_e, R_2). Charakteryzuje go następujące twierdzenie:

Twierdzenie (Csiszár - Körner): Trójka (R_1, R_e, R_2) należy do $\mathcal{R}$, wtedy i tylko wtedy jeżeli istnieje łańcuch Markowa zmiennych losowych $U \to V \to X \to Y \times Z$, takich że rozkłady prawdopodobieństw warunkowych $p(Y|X)$ i $p(Z|X)$ są zgodne z rozkładami brzegowymi kanału i spełnione są warunki:

1. $R_e \leqslant R_1$
2. $R_e \leqslant I(V:Y|U) - I(V:Z|U)$
3. $R_1 + R_2 \leqslant I(V:Y|U) + \min\{I(U:Y), I(U:Z)\}$
4. $R_2 \leqslant \min\{I(U:Y), I(U:Z)\}$

Wniosek: Załóżmy, że kodujemy blokowo wiadomości tajne i jawne w blokach długości k, jako słowa długości n nad alfabetem $\mathcal{X}$. Wiadomości jawne są od siebie niezależne, podobnie wiadomości tajne. Zakładamy, że dla każdego ϵ istnieją k i n i trójka (f, ϕ, ψ) taka, że:

- $\frac{k}{n} \geqslant R - \epsilon$ (współczynnik transmisji jest ϵ-blisko współczynnika R).
- $\frac{1}{k} H(S^k|Z^n) \geqslant \Delta - \epsilon$ (jesteśmy ϵ-blisko pewnego poziomu poufności Δ).
- $\frac{1}{k}\mathbb{E}d_H(S^n \times T^n, \phi(Y_n)) \leqslant \epsilon$ i $\frac{1}{k}\mathbb{E}d_H(T^n, \phi(Z_n)) \leqslant \epsilon$ (błędy transmisji są ϵ-małe).

Taki ciąg kodów istnieje wtedy i tylko wtedy gdy: $(RH(S^k|T^k), R\Delta, RH(T^k)) \in \mathcal{R}$.

Zawsze $\Delta \leqslant H(S^k|T^k)$ (o wiadomości poufnej odbiorca Z wie przynajmniej tyle, ile można wywnioeskować z wiadomości jawnej T^k). Maksymalną poufność uzyskamy, gdy S^k i T^k są niezależne i gdy nierówność się wysyci, czyli gdy $\Delta = H(S)$. W takim przypadku poszukujemy trójek $(RH(S^k), RH(S^k), RH(T^k)) \in \mathcal{R}$, a warunki z twierdzenia redukują się do:

2. $R_1 \leqslant I(V:Y|U) - I(V:Z|U)$
4. $R_2 \leqslant \min\{I(U:Y), I(U:Z)\}$

Dalej, możemy założyć że wiadomość jawna nie jest przesyłana. Warunek 4. jest wtedy spełniony zawsze, a maksymalny współczynnik transmisji który możemy osiągnąć przy przesyłaniu tylko wiadomości poufnej i przy założeniu pełnej poufności ($H(S) = \Delta$) to:

$$\max_{V \to X \to Y \times Z} I(V:Y) - I(V:Z)$$

Wielkość tę oznaczamy jako C_s (*pojemność poufności — secrecy capacity*).

Załóżmy, że $C(X \to Y) \geqslant C(X \to Z)$. (warunek ten spełniają m.in. wszystkie kanały zdegradowane). Wtedy:

$$C_s = \max\{I(X:Y) - I(X:Z)\}.$$