# Data Compression

León Illanes F.

10 de abril de 2012

# Introducción

# Medición de información

Buscamos una forma razonable de medir la información de una variable $X$

- Podemos medir el espacio necesario para guardar $X$.
  - Espacio promedio, depende de la distribución de $X$
  - Si $X$ es un string de $n$ bits y distribuye uniforme, se necesitan $n$ bits
  - Si hay un string $a$ que aparece con probabilidad $1/2$, mientras el resto distribuye uniforme: $n/2 + 1$ bits

# Medición de información (cont.)

- Podemos medir la predictibilidad de $X$.
  - Obviamente también depende de la distribución de $X$
  - Para $X$ uniforme la probabilidad de adivinar el resultado es $1/2^n$
  - Para el otro caso la probabilidad de adivinar es $1/2$

# Medición de información (cont.)

- Podemos medir el tamaño esperado del programa más corto que imprime $X$
  - Esta es la Complejidad de Kolmogorov
  - Se puede considerar el promedio sobre la distribución de $X$, o se puede aplicar sobre un string particular
    - "127351430923412452300234 12..." contiene más información que "11111111111111111111111..."

# Entropía

## Definición

*Entropía de Shannon (de primer orden):*

$$H(X) = \sum_x p(x) \log \left( \frac{1}{p(x)} \right)$$

Convención: $0 \log(1/0) = 0$ ($\lim_{x \to 0} x \log(1/x) = 0$)

## Definición

$$H(X) = \mathbb{E} \left[ \log \left( \frac{1}{p(X)} \right) \right]$$

# Justificación

La justificación para la definición de entropía es axiomática. Si tomamos los siguientes axiomas para una función de información $f$, se puede demostrar $f = cH$:

- Si $X$ es uniformemente distribuida sobre un conjunto de tamaño $M_X$ e $Y$ es uniformemente distribuida sobre un conjunto de tamaño $M_Y > M_X$, entonces $f(Y) > f(X)$.
- Si $X$, $Y$ son independientes, entonces $f(X, Y) = f(X) + f(Y)$.
- Si $B$ es una variable con distribución de Bernoulli, con probabilidad de éxito $p$, entonces $f(B)$ es una función continua de $p$.
- Si $X$ y $B$ son variables aleatorias y $B$ tiene distribución de Bernoulli con probabilidad de éxito $p$, entonces $f(BX) = f(B) + p\, f(X|B = 1) + (1 - p)\, f(X|B = 0)$.

# Propiedades

La verdadera justificación es que cumple con propiedades interesantes que se ajustan a la intuición

- $H(X) \geq 0$
- $H(0) = H(1) = 0$
- $H(1/2) = 1$
- $H(B_q) = H(B_{1-q})$

# Entropía conjunta

## Definición

*Entropía conjunta:*

$$H(X, Y) = \sum_x \sum_y p(x, y) \log \left( \frac{1}{p(x, y)} \right)$$

## Definición

$$H(X, Y) = \mathbb{E} \left[ \log \left( p(X, Y) \right) \right]$$

# Entropía condicional

## Definición

$$H(X|\,Y=y)=\sum_x p(x|y)\ \log\left(\frac{1}{p(x|y)}\right)$$

## Definición

*Entropía condicional:*

$$\begin{aligned} H(X|\,Y) &= \sum_y p(y)\ H(X|\,Y=y) \\ &= \sum_y \sum_x p(x,y)\log\left(p(x|y)\right) \\ &= \mathbb{E}\left[\log\left(\frac{1}{p(X|\,Y)}\right)\right] \end{aligned}$$

# Entropía condicional (cont.)

Funciona como se espera (ie. regla de la cadena):

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

Demostración (pizarra)

# Entropía relativa

También se conoce como Divergencia de Kullback-Leibler.
Es una medida de la diferencia entre dos distribuciones.

## Definición

*Entropía relativa:*

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

$$= \mathbb{E}\left[\log \frac{p(X)}{q(X)}\right]$$

Convención: $0 \log 0/q = 0$ y $p \log p/0 = \infty$

# Entropía relativa (cont.)

Relación entre entropía relativa y entropía:

$$\begin{aligned} H(X) &= \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)} \\ &= \log |\mathcal{X}| - \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{\frac{1}{|\mathcal{X}|}} \\ &= \log |\mathcal{X}| - D(p||u) \end{aligned}$$

Donde $u$ se refiere a la distribución uniforme.

# Información mutua

Es la entropía relativa entre la distribución conjunta y el producto de las distribuciones.

## Definición

*Información mutua:*

$$I(X;Y) = D(p(x,y)||p(x)p(y))$$
$$= \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

# Información mutua (cont.)

$I(X;Y)$ cumple con la siguiente propiedad:

$$I(X;Y) = H(X) - H(X|Y)$$

Demostración (pizarra)

$I(X;Y)$ corresponde a la reducción en incerteza de $X$ debido a conocer $Y$.

# Información mutua (cont.)

Otras propiedades interesantes:

- $I(X; Y) = I(Y; X)$
- $I(X; Y) = H(X) + H(Y) - H(X, Y)$
- $I(X; X) = H(X)$

# Compresión de datos

El objetivo es generar algoritmos que reciben un mensaje y retornan el mensaje codificado de una forma que usa una menor cantidad de bits.

¿Se puede hacer esto, en general?

No.

# Argumento de conteo

- La cantidad de mensajes (binarios) de tamaño $n$ que se pueden generar son $2^n$.
- La cantidad de mensajes (binarios) de tamaño estrictamente menor a $n$ es $2^n - 1$.
- Luego, por el Principio del Palomar, es imposible comprimir todos los mensajes de tamaño $n$ a codificaciones de largo menor.

Dado que no se puede comprimir todo, es indispensable suponer un cesgo en la distribución de los mensajes y comprimir para reducir el promedio ponderado del tamaño.

Con esto el problema de la compresión se divide en dos partes:

- Modelamiento (definir la distribución de probabilidad de los mensajes)
- Codificación

# Codificación

El objetivo es generar algoritmos que reciben un mensaje y retornan el mensaje codificado de una forma que usa **en promedio** una menor cantidad de bits.

Dada una distribución, buscamos codificar de modo que los mensajes más frecuentes tengan los códigos más cortos.

En la práctica no se suele trabajar con mensajes completos, si no que se dividen en símbolos. Técnicamente se puede considerar cada uno de estos símbolos como mensajes aparte.

De este modo, funcionamos con la distribución de probabilidad de los símbolos (y no de los mensajes completos).

## Definición

*Código:*
*Un código $C$ para una variable aleatoria $X$ es un mapping de $\mathcal{X}$ a $\mathcal{D}^\star$. Donde $\mathcal{D}$ es un alfabeto de tamaño $D$.*

## Definición

*Largo esperado de un código $C$:*

$$L(C) = \sum_{x \in \mathcal{X}} p(x) \; |C(x)|$$

## Definición

*Un código $C$ se dice **no singular** si mapea cada elemento de $\mathcal{X}$ a un string distinto en $\mathcal{D}^\star$:*

$$x_i \neq x_j \Rightarrow C(x_i) \neq C(x_j)$$

## Definición

*La extensión $C^\star$ de un código $C$ corresponde al mapeo de $\mathcal{X}^\star$ a $\mathcal{D}^\star$:*

$$C^\star(x_1 x_2 \cdots x_n) = C(x_1)C(x_2)\cdots C(x_n)$$

**Definición**

*Un código $C$ se dice* ***únicamente decodificable*** *si su extensión $C^\star$ es* ***no singular****.*

Todo string codificado por un código únicamente decodificable tiene un único string fuente que lo produce.

**Definición**

*Un código $C$ es un* ***código prefijo*** *(o código instantáneo) si ninguna palabra codificada es prefijo de otra.*

Un código prefijo se puede decodificar en "tiempo real". Si termino de leer un símbolo codificado, puedo traducirlo inmediatamente.

# Código prefijo

Puede verse como un árbol de decisión con ramificación $D$. Para $\mathcal{D} = \{0, 1\}$ se trata de un árbol de decisión binario.

# Códigos óptimos

## Teorema

***Desigualdad de Kraft***. *Para todo código prefijo sobre un alfabeto de tamaño $D$, los largos de las palabras codificadas ($l_1$, $l_2$, ..., $l_n$) deben satisfacer:*

$$\sum_i D^{-l_i} \leq 1$$

*Además, dado un conjunto de largos de palabras codificadas que satisfacen la desigualdad, se sabe que existe un código prefijo con estos largos.*

# Desigualdad de Kraft: Demostración

- Sea $l_{max}$ el largo de la palabra codificada más larga. Consideremos el árbol de decisión $D$-ario completo (algunos nodos son palabras, algunos van en camino a formar palabras y otros son descendientes de palabras).
- Si tomamos un nodo $v_i$ que corresponde a una palabra de largo $l_i$, podemos definir $A_i$ como el conjunto de hojas del subárbol con raíz en $v_i$.
- Es fácil ver que $|A_i| = D^{l_{max}-l_i}$.
- Dado que el código es prefijo, necesariamente se tiene que para dos palabras (nodos $v_i$ y $v_j$) se tiene $A_i \cap A_j = \varnothing$
- Sumando sobre todas las palabras:

$$\sum_i D^{l_{max}-l_i} \leq D^{l_{max}} \Rightarrow \sum_i D^{-l_i} \leq 1$$

# Códigos óptimos

Sabemos que todo código prefijo satisface la desigualdad de Kraft, y que esta desigualdad es condición suficiente para la existencia de un código con los largos de palabras especificados.

Queremos encontrar códigos que minimizan el largo esperado:

$$\text{mín} \sum_i p_i l_i$$

$$s.a. \sum_i D^{-l_i} \leq 1$$

$$\text{mín} \sum_i p_i l_i$$

$$s.a. \sum_i D^{-l_i} \leq 1$$

Solución:

$$l_i^\star = \log \frac{1}{p_i}$$

De esto se obtiene:

$$L^\star = \sum_i p_i l_i^\star = \sum_i p_i \log \frac{1}{p_i} = H(X)$$

# Códigos de Huffman

- Huffman (1952) diseñó un algoritmo para hacer asignación óptima de códigos: genera el árbol de decisión
  - Aproximadamente óptimo. . . , hay un problema al hacer aproximación a cantidades enteras de bits.
  - Es verdaderamente óptimo si las probabilidades son todas potencias de 1/2. Se hace más malo mientras más lejos se está de esto.

# Algoritmo de Huffman (para código binario)

- Partimos con un bosque en que cada símbolo es un nodo suelto con una etiqueta de probabilidad (que corresponde a la distribución dada)
- En cada iteración tomamos los dos árboles cuyas raíces tienen menor probabilidad: $p_i$ y $p_j$.
- Agregamos un nuevo nodo al que asignamos probabilidad $p_i + p_j$. Ponemos los otros dos árboles como los descendientes de este nodo.
- Iteramos hasta que queda un único árbol.

Es fácil de extender a un alfabeto más grande, haciendo que se acerque más al óptimo, pero se va haciendo más lento de ejecutar.

# Codificación aritmética

- Rissanen (1976). Es un algoritmo que se aproxima mejor al óptimo.
- No se usó mucho hasta hace poco porque es relativamente lento (float) y estaba protegido por patentes...

# Codificación aritmética (cont.)

- Dado un orden $<$ para los símbolos del alfabeto y un modelo $P$ (ie. $P(x)$ es la probabilidad de que aparezca el símbolo $x$), definimos $P_<(x)$ como $\sum_{y<x} P(y)$.
- Además, definimos $P_\leq(x) = P_<(x) + P(x)$.
- El código aritmético para un string $x$ es el número (float) binario $b$ más corto tal que $P_<(x) \leq b < P_\leq(x)$.
- Este número $b$ siempre existe y es tal que nunca es más de un bit más largo que $\log 1/p_x$.

El algoritmo sigue los siguientes pasos:

- Inicialmente se definen rangos de probabilidad acumulada para los símbolos. ($P_<(x)$, $P_\leq(x)$)
- Definimos $low = 0$, $high = 1$.
- Loop sobre símbolo $x$ de entrada:
  - Definimos $range = high - low$.
  - Definimos $high = low + range * P_<(x)$.
  - Definimos $low = low + range * P_\leq(x)$.
  - Retorna un número entre $low$ y $high$.

# ¿Qué falta para completar un curso express en compresión de datos?

- Codificación por transformadas: substitución o diccionario
- Modelos
- Compresión con distorsión (ie. lossy compression)