Information and Coding Theory: Quick reference

vishvAs vAsuki

March 28, 2012

Part I

Introduction

1 Notation

Hamming distance d(x,y).

2 Themes

Designing efficient and reliable data transmission, for compression and error correction. Suitability of codes for particular purposes.
For Cryptography, see cryptography ref.

3 Information

3.1 Self Information of an event

Aka surprisal. Measure of information content associated with event e: rarer the event, more the info, and in case of independence $\perp (e, f) : h(e, f) = h(e) + h(f)$. In the latter case, Pr(e,f) = Pr(e)Pr(f); thence get derivation: $h(e) = h(X = x) = \log(\frac{1}{Pr(e)})$.

3.1.1 As code-length for recording event

3.1.1.1 Coding problem

Suppose that we wanted to record information that an event occurred, but we wanted to use as few bits in expectation as possible. We want to satisfy this: the more common the event, fewer the bits one would need to transmit the event's occurrence.

3.1.1.2 Coding algorithm

We observe that there can be at most $1/p$ events with probability p. So, assigning $\left\lceil \log(\frac{1}{Pr(e)}) \right\rceil$ bits to communicate the occurrence of an event ensures that we have a way of encoding all possible events, while using fewer bits to encode commoner events.
This is a code with the least expected code-length, as shown in the entropy section.

3.1.2 Unit

Inspired by the code-length interpretation of surprisal. Depending on whether $\log_2$ or ln is used in definition: bits or nats.

3.2 Entropy of an RV X

3.2.1 Definition

3.2.1.1 Desired properties

Uncertainty associated with an RV: Should not change if probability rearranged for different values of X: symmetry; should increase with number of values X can take; if $X \perp Y$, uncertainty of (X, Y) should be sum of uncertainties.

3.2.1.2 As expected surprisal

$H(X) = E[h(X)] = E_X[-log(Pr(X = x))]$
$= -\sum Pr(X = x_i) \log(Pr(X = x_i))$; is the only measure which satisfies this [**Find proof**].

3.2.1.3 Extension to 0 values

Extend definition for $Pr(X = x_i) = 0$:
$lt_{Pr(X=x_i)\to 0} Pr(X = x_i) \log(Pr(X = x_i)) = 0$, so set $Pr(X = x_i) \log(Pr(X = x_i)) = 0$: so expansibility property: No change in entropy due to adding 0 probability events $X = x_i$.

3.2.2 Expected Information/ code-length

Entropy of X is the average amount of information/ surprisal communicated by the corresponding random process.
It is the least expected number of bits required to transmit the value of the random process.

Proof. : Non negativity of Information divergence.

3.2.2.1 Cross entropy

Even though X may have distribution D, an alternative code appropriate for random variable corresponding to distribution E can potentially be used to encode events $X = x$. But, the expected code length is higher if this is done. This inspires a way of measuring divergence between distributions - Information (KL) Divergence/ Code-length divergence $KL(E||D)$. This is described in probability theory survey.

3.2.3 As cross entropy relative to U

$H(X) = \log |ran(X)|$ if $X \sim U$. $KL(X||U) = \log |ran(X)| - H(X)$; but $KL(X,U) \geq 0$, so U has max entropy, reduction in entropy is $KL(X,U)$.
Non uniform distribution has less entropy than uniform distribution. Can use this to reduce the number of bits needed to transmit information.

3.2.4 Concavity in case of discrete distribution p

$H(p) = \sum_i p_i \log(1/p_i)$: concave in p_i as $\nabla^2 H(p) \succeq 0$. Consider RV X $\sim$ bernoulli(p): entropy cup shaped, with max at p=0.5.

3.2.5 Asymptotic equipartition property (AEP)

Take binary distribution with entropy H, iid sample $\{X_i\}$, get sequence (X_i). Then, sequences will either have probability 2^{-nH}, or ≈ 0. So, need only nH bits, rather than n bits. Pf: Set $Y_i = \log \frac{1}{Pr(X_i)}$; By law of large numbers $n^{-1} \sum Y_i \to H$; so $-Pr((X_i) = (x_i)) \to nH$.

3.3 Joint and cross entropy

3.3.1 Joint entropy

$H(X,Y) = E_{x,y}[-log(Pr(X = x, Y = y))]$.
Additivity, as requried: If $X \perp Y : H(X,Y) = H(X) + H(Y)$; subadditivity: $H(X,Y) \leq H(X) + H(Y)$.

3.3.2 Cross entropy

$H_C(X,Y) = E_x[-log(Pr(Y = y))]$: avg bits required to transmit X using protocol designed for Y. Compare with information divergence: that is the number of extra bits required to transmit X using a protocol designed for Y.

3.4 Conditional entropy of X given Y

$H(X|Y) = E_y[H(X|Y = y)] = E_y[E_x[-log(Pr(X = x|Y = y))]] = H(X,Y) - H(Y)$: Aka equivocation; Avg uncertainty in X, after seeing Y.

3.5 Mutual information of X wrt Y

$I(X;Y) = E_{x,y}\log[\frac{Pr(X=x,Y=y)}{Pr(X=x)Pr(Y=y)}] = H(X) - H(X|Y) = H(X) + H(Y) - H(X,Y)$ - visualize with a venn diagram!: reduction in uncertainty about X due to knowledge of Y. It is symmetric.
This is the expected value of the information gain / code-length divergence: $E_x[H(Y) - H(Y|X=x)]$; and is therefore loosely called information gain when considered in the context of classification problems in machine learning.

3.5.1 As deviation from independent distribution

$I(X;Y) = K(Pr(X=x,Y=y)||Pr(X=x)Pr(Y=y))$; so $I(X;Y) \neq 0$ iff $X \perp Y$. So, it is non negative.

3.5.2 Conditional Mutual information wrt Z

$I(X;Y|Z) = E_z[I(X;Y|Z=z)]$.

3.6 Other information metrics

Hamming weight of x: wt(x). Hamming distance: $d(x,y) = wt(x \oplus y)$.

3.7 Communication complexity

3.7.1 The problem

A talks to B; A knows a; B knows b; want to find f(a, b) with min communication and even ∞ local computation. a, b are n bit numbers.
Easy solution is to send a and b. But these may be large. So want to use some protocol depending on f.

3.7.2 Applications

VLSI, scenarios where communication is very costly.

3.7.3 The communication protocol tree

$A \leftrightarrow B$ communication can be represented as this: A and B take turns sending messages, the message sent at step i is $m_i = f_i(a,b)$. Maybe distriubution M over (a, b) specified and want to minimize expected communication, maybe want min worst case communication.
So, can look at all possible communication sequences using a protocol tree.

3.7.4 Deterministic vs randomized protocols

Bits transmitted by deterministic protocol, for worst possible (a, b) := $D(f)$.
If distribution M specified: $D_M(f)$: avg bits used.

Randomized protocols may use public randomness or private random bits. Bits used by them for worst (a, b) := $R(f)$. Randomized protocols much more powerful than deterministic ones: See equality testing example.
Having public random bits is not much more powerful: you can replace public random bit using protocol with private random bit using protocol with only $+\log n$ bits penalty.

3.7.5 Computing f for k input pairs

Want to do better than $kD(f)$ from trivial algorithm. Deterministic protocol: $\Omega(k\sqrt{D(f)})$. Randomized protocol: $\tilde{\Omega}(R(f)\sqrt{k})$.

3.7.6 Examples

3.7.6.1 Checking equality

$f(a,b) : b \stackrel{?}{=} a$. Any det protocol needs n bits. So use fingerprinting (see Randomized algs ref).
A uses rand r, sends fingerprint (F(a, r), r) to B.
To show that F is good: Make $\hat{F}(a) = ((F(a,r_1),r_1),..(F(a,r_s),r_s))$; pick rand element and send. For all $a \neq b$, show Hamming dist $\delta(\hat{F}(a),\hat{F}(b))$ large.

Part II

Coding

4 Fingerprinting

This codes can also be used as error detection codes.

4.1 Chinese reminder code

Codes which use a mod p, with rand p. $\hat{F}(a)$ elements will use diff fields; so not preferred.

4.1.1 Checking equality

A picks rand prime p between 1 and $k = n^3$; Sends (a mod p, p) to B; B says '=' if $a \equiv b \mod p$.
$Pr_p(a \equiv b \mod p | a \neq b) \leq n^{-1}$: num(p with $a \equiv bmodp$ when $b \neq a$) or, $num(p|(a-b)) \leq n^{-1}$ as $a-b \in [0, 2^n-1]$; so $Pr(p|a-b) < \frac{n}{\Pi(n^3)} =\leq \frac{n\ln n}{n^3} \leq \frac{1}{n}$ Using Prime number theorem.

4.2 Univar polynomial code

(Reed Solomon) Codes which make univar polynomial p_a over $\mathbb{F}_p$, $(deg \leq n)$, from a, prime p, with a's bits representing coefficients.

4.2.1 Checking equality

Fix p. A picks rand r from F_p, sends $(p_a(r), r)$ to B, B accepts if $p_b(r) = p_a(r)$. $Pr((p_b - p_a)(r) = 0) \leq \frac{n}{p}$: max n roots.

4.3 Multivar polynomial code

(Reed Muller). [**Incomplete**]

5 Source coding

Compression. See the example about checking equality.

6 Channel Codes

6.1 Code design

In most cases, this is an art, rather than a science. Not many things are proved; instead one runs long simulations to show goodness of a code.

6.2 Modelling a channel

Transmitted x is transmuted to y; want to model this process.

6.2.1 Channel capacity

Aka Shannon limit or capacity. The tightest upper bound on the amount of information that can be reliably transmitted over a communications channel.

6.2.2 Binary symmetric channel

$\Pr(x_i \neq y_i) = \text{p}$.

6.2.3 Erasure channel

$Pr(y_i = x_i) = p$, with 1-p probability, $y_i =?$ (erased). This can model packet loss.

6.3 Model message distribution

Usually assume uniform distribution over messages.

6.4 Tolerating errors

Design codes and protocols for error detection and correction.

6.4.1 ARQ

If error detected, ask for retransmission.

6.4.2 Forward error correction

Receiver never sends any message back to transmitter. Error correcting code (ECC) attached with data used to fix errors.

6.4.3 Decoding

Set of codes $C\left\{F\right\}_2^n$. x is received. Select $c \in C$ closest to x.

List decoding Output a list of codes within a certain distance of the mangled code.

6.4.4 Joint source-channel coding

Encoding of a redundant information source for transmission over a noisy channel, and the corresponding decoding. [**Incomplete**]

6.5 Properties

6.5.1 Minimum Hamming Distance d

Aka distance of the code, Hamming metric. Closely related to the error correcting ability of the code.
More efficient encoding and decoding. [**Find proof**]

6.5.2 Code rate

Code rate k/n. High rate code if this is high.

6.6 Types

6.6.1 Block vs Convolutional codes

Block codes: k-bit info to n-bit code. Block length n.
Convolutional code: k bit info to n bit code.

6.6.2 Bound on code size of block codes

(Gilbert-Varshamov). Take code with length n, distance d, size (not dimension) of the code $A_q(n, d)$. Then $A_q(n, d) \geq \frac{q^n}{\sum_{j=0}^{d-1} \binom{n}{j}(q-1)^j}$ [**Find proof**].
The best rate vs distance tradeoff.

6.6.3 w error correcting code

A code which can correct w erroneous bits.

w error correcting linear code Given n*m generator matrix G and m bit y, find n bit x such that $d(xG, y) \leq w$, if it exists. This is possible only if corresponding $d \geq 2w + 1$.

6.6.4 Cyclic code

Right shifting a code $c \in C$ also yields a code in C.

7 [n, k, d] Linear code C

A type of block code. Block Length n, message length k, min Hamming distance d: encode k bit msg in n bit message.
d is the min Hamming wt of any non zero code vector [**Find proof**].

7.1 A linear subspace of valid codes

A linear subspace C with dimension k of vector space F_q^n over finite field F_q. The channel takes you away from this subspace, find the vector closest to the received message in the subspace.
All vector and scalar ops are in F_q. For Binary linear codes, use the field F_2.

7.1.1 Basis codes

Can be represented as span of basis codes. Basis codes form rows of k*n generating matrix G. Standard form of G: G is of the augmented matrix $[I_k : A]$, with k*(n-k) A. To encode x, find xG.

7.1.2 Random [n, k] code

k vectors chosen randomly from $\{0, 1\}^n$. Or, full rank G is chosen randomly. Achieves whp Gilbert Varshmov bound on rate vs distance tradeoff.

7.2 Decoding

Check/ parity check matrix H: n*(n-k), with left kernel C; $H = [-A^T : I_{n-k}]$ in std form. GH=0. To check y, verify: $yH = 0$.

For corrupted y, there is an error vector e with $wt(e) \leq \frac{d-1}{2}$ such that $y \oplus e = xG$ for some x. To decode, look at its syndrome: yH = (x + e)H = eH. Then solve for e or look it up in a table. Then find x.

7.2.1 [p, q] regular code

Make a bipartite graph: bits in variable x on one hand, and nodes corresponding to parity checks in H on the other. If this is a [p, q] regular graph, you have a [p, q] regular code.

7.2.2 Decoding

Avg case hardness unknown. Worst case decoding is NP hard. Even finding d is belived hard.

7.2.3 As inference over factor graph

Make a factor graph Make nodes for the transmitted codeword bits x, and for the corresponding received/ corrupted codeword bits y. Make factors corresponding to the parity checks for y: eg: if H contains a check which says $\oplus_{i \in S} x_i$, make a factor f_S, such that any x where this is not satisfied has probability 0. Relationship between x_i and y_i can be modelled using a symmetric error: maybe y_i is corrupted with probability p.

The inference problem y is observed, x is unobserved - to be inferred. Can use loopy belief propogation for doing this.

Guarantees for [p, q] regular codes As the block size n increases, can be sure that loopy belief propogation properly decodes: shown using the 'density evolution' argument. Loopy belief propogation gets into trouble because of cycles; but if you consider the computation tree corresponding to a node, maybe convergence achieved well before a cyclical message is received!