CSc 8370 fall '14

Practicing Statistics without a License.

Classical Frequentist Approach:

Can we show that two things are identical with frequency-based statistics? Or, is it easier to show that two things are different?

Sample size, frequencies and sufficiency.

Given a sample of size N, frequencies are estimated by

f('x') = #times symbol is 'x' / total # of symbols
where 'x' is whatever we think is a symbol (it can be a string for example).

In the unobtainable limit of N-> ∞ the frequency converges to the probability f('x') -> p('x'). This is a consequence the law of large numbers.

$$\mu = \lim_{N \to \infty} \frac{1}{N}(X_0 + ... + X_N)$$ (the mean converges in the limit of very large numbers)

Defining X as f_N('x') – p('x') with a mean value of 0 shows that the errors in frequency vs. probability eventually disappear with large samples.

The problem is how fast does this converge? The answer is not very quickly. It is governed by the Central Limit Theorem (CLT). In its simplest version the convergence of errors follows a binomial distribution (coin flips of +/-) which is well approximated by a Gaussian with $\sqrt{N}$ as a standard deviation. The ratio between deviation and N then goes as $\frac{1}{\sqrt{N}}$ (Note to the cognoscenti – this is a very approximate approach, and there are much better "real" proofs and definitions).

The consequence of this is that infrequent events may not be observed very accurately relative to more frequent events. (for example the letter 'q' is seen in less than 1% of English letters (0.17%) – so it would be surprising if exactly 17 were seen in 10000 letters, and it might have an anomalous frequency in unusual texts "Queuing queer queen moves quickly wins in chess").

The problem in cryptanalysis is that we don't have good control (if any) of sample size. We have what we have. While there are historical examples of generating "traffic" to get larger samples, and there are attacks which involve generating "chosen ciphertext", we cannot, in general, assume that we can get arbitrarily large samples.

Sufficiency. The idea of sufficiency is that a "sufficient statistic" is a function that captures enough of the information in a distribution from samples to reliably estimate the parameters of the distribution.

For example with a normal distribution:

$$N(x,\mu,\sigma)=\frac{1}{\sigma\sqrt{(2\pi)}}e^{-(\frac{x-\mu}{\sigma})^2}$$

the estimated mean and standard deviation are sufficient statistics to reproduce the distribution. Note that there are other combinations of statistics that would work too. These would be a little harder to use, but would work.

Using sufficiency to decide. The key idea is that we calculate a sufficient statistic on our sample and compare it to the values we estimate from our model distribution. Or we calculate the sufficient statistics on two samples and compare them. If, in some general sense, they are the 'same', then we say that the two underlying distributions are the 'same'. The central limit theorem can be used to provide sample error bounds on what is mean by being 'the same', and a probability of being an accidental agreement can be derived using a number of standard statistical tests (chi-squared, Student's T-test etc).

The Big Problem. Complicated distributions require complicated models and many more parameters.

$Nasty(x,\mu_0,\sigma_{0,}..,\mu_n,\sigma_n)=\frac{1}{Z}Polynomial(e^{-(\frac{x-\mu_i}{\sigma_i})^2})$ Will require 2(n+1) statistics to be sufficient.

What happens if there isn't enough data do find all 2(n+1) statistics with any pretense of accuracy?

Negation Rules! We can always generate a simple model and show that our distribution is different from it. Then we only need enough sample to calculate a sufficient statistic for the simple model. The uniform distribution (uniform prior) is usually a good model for a non-solution of a cryptosystem, since the goal of encryption is to remove information from the message in the absence of a key. It is also relatively easy to analytically calculate expected values of statistics for uniform distributions.

$$\mu(U)=\frac{1}{N};\sigma(U)=0$$

$2\text{nd}\ moment(U)=N(\frac{1}{N})(\frac{1}{N})=\frac{1}{N}$ We will see this again and again so don't forget it!

Another example can be seen with binary ciphers that take the space of 8-bit bytes (Z^8) to itself. If the plaintext is in ASCII characters which are all in the 7-bit space, then there is one bit of the plaintext per character that is always known (the “sign” bit) and is 0. Any decryption that does not produce all of the “sign” bits being 0 is wrong and the amount of sample I need to make this decision is smaller than the amount I need to make the decision based on a more complete model of the language. If the probability of the “sign” bit being 0 is 0.5 then the probability of an accidental string of 0's is 0.5^N which converges to zero rather quickly. (Negation is “almost” Bayesian)

Statistical Measures of Languages

We need to be able to identify when a string is in a (human) language. It is also important to be able to recognize when two strings have been encrypted by the same algorithm in the same key. We've seen that doing this by hand is rather tiresome. Therefore we need to examine computable measures that we can use. We assume that if two strings are characterized by identical distributions, then they come from the same source. The measures listed below describe several ways of characterizing the distributions.

1) Approximate Measure of Probability.
   Given a sample of size N, the probability that an event (eg. the occurrence of a given symbol) occurs is approximated by :
   $\sum \partial(t_i, x)/N$
   where ∂ is 1 when the two symbols are the same and 0 otherwise
2) Incidence of Coincidence or κ

   $$k(t,t') = \sum_{i=1}^{N} \partial(t_i, t'_i)/N$$

   where ∂ is 1 when t equals t' and 0 otherwise.
   $k \leq 1$ and is only equal to 1 when the two texts are identical. If the sources are characterized by a probability distribution p, then the expected value for k is the sum of p^2. If the source is random (or if the sources are random with respect to each other) then the expected value is just $1/n_{symbol}$.
3) Measures of smoothness for the distribution (Φ,Ψ)
   Φ,Ψ measure the smoothness of the distribution. Uniformly distributed symbols are smooth, and meaningfully distributed symbols are not (to first approximation). For both of these functions one calculates the empirical distribution m_i which is just the number of times the symbol is symbol i divided by the total number of symbols in the string (i.e. the size of the string).

   $$Psi(T) = \sum_{i=0}^{N_{symbols \in the\ text}} (m_i - m_{average})^2 = \sum (m_i^2 - 2 m_i m_{average} + m_{average}^2)$$

   $$Psi(T) = \sum_{i=0}^{N_{symbols \in the\ text}} (m_i - m_{average})^2 = \sum (m_i^2) + constant$$

   so we just drop the constant.

   $$Psi(T) = \sum_{i=0}^{N_{symbols \in the\ text}} m_i^2$$

   (some authors normalize in this formula, in which case it is m/M where M is the size of the string).

   $$Phi(T) = \sum_{i=0}^{N_{symbols \in message}} m_i(m_i - 1/N)$$

$$Phi(T)=\sum_{i=0}^{N_{symbols\in message}} m_i(m_i-m_{average})$$

(again this is for normalized m, for un-normalized the m_i M would be just m_i)

These formulas appear to be "magic", but are actually based in approximations to the entropy of a distribution (information theory). The information contained in a distribution can be defined as -log(p). This is "simply" a useful way of describing how many "bits" are needed to represent the distribution at a given fidelity. It is also a very useful way to handle joint distributions because the logarithms add when the probabilities multiply. The expected value of the information (-p ln p) is identified as the entropy and this is maximized when the distribution is mostly flat. Chernoff noted that $\ln(x)\le x-1$ and therefore x ln(x) < x(x-1).

4) Chi function.

$$Chi(T,T')=\frac{\sum_{i=1}^{M} m_i m_i'}{M^2}$$ For unnormalized or

$$Chi(T,T')=\sum_{i=1}^{M} m_i m_i'$$ for normalized.

Note that Chi(T,T) is just Psi(T)

5) Kappa Psi or Kappa Chi equivalence.

$$Chi(T,T')=\frac{1}{M}\sum_{n=0}^{M-1} Kappa(T^{(i)}T')$$ Where $T^{(i)}$ is T shifted circularly by i positions. This gives you an estimate of what Kappa should be if the two texts are drawn from the same source which is sort of important if you are not sure that they are.

6) Measures of errors in the distribution (χ^2)

An empirical distribution m can always be compared to a theoretical distribution p by a χ^2 test.

$$X^2=1/(N-1)\sum_{i=0}^{N_{number\ of\ symbols}} (m_i-p_i)^2$$

(unscaled version would replace p by p*M).

This will be minimal at a correct decryption.

7) How much text is enough? (unicity distance).

The unicity distance is the size of the message which has a unique key. It can be estimated by using the entropy of the distribution p. It depends both on the underlying clear language and the cipher system. For single substitution in English it is about 25 characters.

$$U=\frac{1}{3.5}lg\,Z$$ where Z is the number of possible cipher combinations. 3.5 is a

constant chosen to make U= 25 when Z = 26!. With Ceaser ciphers Z = 26 (for N=26 letters), for monoalphabetic Z = 26!, for dialphabetic Z = (26^2!). With block keyed ciphers like DES where there are 2^n keys then Z is 2^n.

6) Patterns in the plaintext

In addition to the frequencies of characters, analysis can be made of the repeat distances between symbols or the "period" or format of the message. In WW2 the enigma system was analyzed with a special purpose machine (a "bombe") that explicitly looked for patterns in the text (for example the distances between the 'e's in "neider mit den Englander" (it was actually more complicated than this and we'll treat it in detail later.)) Similarly, knowing that a message is mostly ascii letters is enough to detect a good key with most modern cryptosystems because of the number of bits that must be zero. Can you estimate the probability of a pattern using the tools already described?

Source Modeling

The empirical tests we've just discussed are useful, but can be improved if there is a good model for the source of the message. In fact without a model we can't apply tools like chi-squared.

Basic Idea:

We find a statistical model that reflects some properties of a language. When these are met then we assume we are likely to have a solution.

Generalized Machine Decryption Algorithm:

Assume (i.e. Make a guess) a keyed mapping between Cipher (C) and plaintext (P).

Search the space of keys for keys which maximize the expected correspondence between decryption and the statistical model of the language. The search can be intelligent and use information from the statistical model, or just be brute force.

Report the best correspondence solutions.

Good modern cryptosystems use a large key space to make this search non-trivial. Keyspaces the size of 2^{40} -2^{56} are readily searched with modern machines.
Much of the research in cryptanalysis is in finding methods to reduce the key search to a tractable level either by restricting the space of keys by analysis of a set of related messages or by finding algebraically equivalent problems that are easier to solve..

Some Source Models:

1) uniform distribution over all symbols. This is a model of C for a good modern cipher. It is not a good model of a human language. So we maximize the difference between this model and our decryption. **This is an example of disproving a null hypothesis.**
$$P(symbol)=\frac{1}{N_{symbol}} \qquad P(symbol_1, symbol_2)=(\frac{1}{N_{symbol}})^2$$
2) Empirical Individual Distribution. Take a sample text and count the distribution of symbols.
3) Empirical Multiple Symbol. Take a sample text and count the distribution of pairs, triples, etc. This is a good model, but most entries will be unobserved.
4) Empirical Transition Probability. Calculate (3) above, but normalize based on path. $P(symbol_1, symbol_2, ...)=P(symbol_1)\cdot P(symbol_2|symbol_1)\cdot ...$ Where P(1 | 2) means P(1 given 2).
   This is a fairly compact and order dependent representation of strings.
5) Known Text (aka a “Crib”). Specific strings that are likely to exist in the text.

Using the models

1) Individual.
   P(are) = P(a)P(r)P(e)
   P(rea) = P(a)P(r)P(e)
2) Multiple
   P(are) = P(ar)P(e) or P(a)P(re) or P(are) depending on modeling
   P(rea) = P(re) P(a) or P(r)P(ae) or P(rea)
3) Transition.
   P(are) = P(a)P(r|a)P(e|r)
   P(rea) = P(r)P(e|r)P(a|e)

Not that the values in 2,3 are not always the same. Even though the mathematical formulations should be equivalent, the observed empirical values may not give identical answers.

Using the Models.

ENTROPY

$$S = \int dP\, P \ln P \approx \int dp\, P(1-P) \approx \sum P(1-P) \approx \sum P \ln P$$

This measures the difference from a uniform distribution.

Conditional entropy

$$S = \int dP\, P \ln \frac{P}{X} \approx \int dp\, P\left(1-\frac{P}{X}\right) \approx \sum P\left(1-\frac{P}{X}\right) \approx \sum P \ln \frac{P}{X}$$

where X is the prior distribution or the distribution we want to measure against.

Relative Entropy

$$S = \int dP\, P \ln \frac{P}{X} - \int dX \ln \frac{X}{P}$$

and many, many more.

Problem Set, In class and out.
Write programs to derive monomer probabilities and pair probabilities. It would also be good to write a program to calculate Psi or Phi. The file 3boat10.txt is a relevant source of plaintext.

The files are on my area (~cscrwh or ~cscrwh/crypto) on snowball so you already have access to them.

0) We've been describing alphabetic languages like English, Italian and Arabic. What happens if we look at a language like Chinese where there may be 4000 symbols? Would there be more or less information about the source on a computer representation of the language?
1) Calculate the phi statistic on single letters and pairs for ciphers 1,2,3,4 on the first handout. One should be unusual, why? (Phi is better here because the samples are small).

2) (the file classcipher) is this a mono alphabetic substitution? Prove it.
(note that the file itself contains some Unix/C end of string '\0' characters in the middle so DO NOT just transcribe this handout – it won't work – they are important).

vvgfxxdfaaavaxdfdvdgvvdgfxdvdxvxgaavfgdfvaagfvfggdgavfdfafgxdvafgfgaxdagfggxdd
vxfxvvaxavvvggxdgvggvxvxvgdgvxdxgdvxgxdvvvxgdaaagavxddaxgdafdvvdaxgxdggav
vfxgxdaddadxggvfavxfaxvgxgaavxvgvgffxdgagdgaxxggaaadxvvdvdgvgdadxvdfddafgdaa
dvvvdvgdfdxvxdvfavxgadvdfxxgxaavvaffgxxfdvggfvgdvgvvavvgadvavvgavfavvvgagddv
xvagxdvdffgaavxvxgxgxddxdgfagggaxavdxaadgxdddgxxffgdggaadavvffxgfvaaavavaxgv
avxxaxgfgddxavxgdavvggxdddaavfddddvagagggvgggaxdfffdgadagvvadaffggfagxavfgxdf
vxfvadaggfdvgxaggfaxdavxvvaaaaaafdvvxxadxfdvxdvxgfagagafvgdvdgddvvagggdddvdf
gagaaadadgadffvaaafxdvgadgdggffffadgfdggvvdadagadxgafaagfadgaavgvfaffvavagfvvfdv
axgvdaggaagfxaxdgxvxfxgaaffgfxgxgxvgxvvvgfvggfgdgavavaadxvvxaxxvfvdvaxdvdgaa
adgxgvadavdaafdvaxdaaggxafdvxxgavvggvfxfavxxvvafvgvvgfdvfddgfvaaggfgvvvvfdvfv
ddvxgdgggvvggggdfxfagfffvgvdxvvfgddxaaadvavgaaaafdvgfdvaxdfgvaxgxafavvadvggdf
gxvfgavvagggavxxfgdgfvvavvdvfddavaaxxdaxxfvxagfdvvvvgxgvgfaaxvfgfxgddvdgavxfa
davvgvvafaxgvgxvvdfvgvadavfdaaggxavdffvadvvgddxgvfvafddaxdvadvgdxdvdxdddavv
dadvaadvxvgdvavvvvagdagfvxdvfgaxaxgfffffv

3) It could be a variant of adfgvx. Is it permuted or not? Prove it.

4)Side information tells you that if it is a transposition then it is an incomplete columnar transposition. Use the Psi test to find which period (i.e. what size column or row) was used.

5) It may have come from 3boat10.txt. How would you check this without solving the cipher? Write a program and try it. (There are several fast algorithms that will identify too many places and a slower algorithm that will only find correct answers).