Chapter 1

Information Theory and Statistics

1.1 Preliminaries

The entropy of a discrete random variable, X, with distribution P on alphabet $\mathcal{A}$ is defined as:

Entropy

$$H[X] = \mathbb{E}_X[-\log P(X)] \tag{1.1}$$

The relative entropy between two distributions, P and Q, is defined as:

Relative Entropy

$$D(P||Q) = \sum_{a \in A} P(a) \log \frac{P(a)}{Q(a)} \tag{1.2}$$

Let $X_1, X_2, \ldots$ be a sequence of iid random variable with mean μ and let $S_n = \frac{1}{n}\sum_{i=1}^{n} X_i$. Then, for any $\epsilon > 0$:

Weak Law of Large Numbers

$$\lim_{n\to\infty} P(|S_n - \mu| \geq \epsilon) = 0. \tag{1.3}$$

Thus S_n converges *in probability* to μ. .

Prove this, starting from Markov

If the sequence of random variables have a common distribution defined by probability mass function, P, the *Asymptotic Equipartition Property (AEP)* states that the probability of a random string decays exponentially with n with rate given by the entropy:

AEP

$$\lim_{n\to\infty} (|\frac{-1}{n} \log P^n(X_1^n) - H[X]| \geq \epsilon) = 0 \tag{1.4}$$

The AEP follows from the weak law of large numbers directly. Equivalently, $P^n(X_1^n) \simeq 2^{-nH[X]}$ for large n with high probability. The typical set of strings for some ϵ is defined as:

Try bounding the size of the typical set.

$$A_{\epsilon,n} = \{2^{-n(H[x]+\epsilon)} \leq x_1^n : P^n(x_1^n) \leq 2^{-n(H[x]-\epsilon)}\} \tag{1.5}$$

1.2 Fixed-Rate Lossless Data Compression

For a source $\mathbf{x} = \{X_n\}$, A fixed-rate lossless compression code consists of a sequence of codebooks $\{B_n\}$, where each B_n is a set of source strings of length n.

Fixed-Rate Code

A simple way of implementing such a code, assuming that both encoder and decoder have access to the codebooks is by, for some message x_1^n:

1. If $x_1^n \in B_n$, send a 1 followed by an index, taking $1 + \lceil \log |B_n| \rceil$ bits.
2. If $x_1^n \notin B_n$, send a 0 followed by the uncompressed message, taking $1 + \lceil \log |A^n| \rceil$ bits.

The rate and error probability of this code are:

$$R_n = \frac{1}{n}(1 + \lceil \log |B_n| \rceil) \text{ bits / symbol} \qquad P_e^{(n)} = Pr(X_1^n \notin B_n) \tag{1.6}$$

Fixed Rate Coding Theorem

Theorem 1.2.1 (Fixed Rate Coding Theorem) *For a memoryless source* **X** *with entropy* $H[X_i]$*:*

(a): For any $\epsilon > 0$*, there exists some fixed-rate code* $\{B_n^*\}$ *with vanishing error of probability* $P_e^{(n)} \to 0$ *as* $n \to \infty$ *with rate:*

$$R_n \leq H[X] + \epsilon + \frac{2}{n} \tag{1.7}$$

for all n.

(b): Let $\{B_n\}$ *be a fixed-rate code with* $P_e^{(n)} \to 0$ *as* $n \to \infty$*. Then, for any* $\epsilon > 0$*:*

$$R_n \geq H[X] - \epsilon \tag{1.8}$$

eventually.

Prove the Fixed Rate Coding Theorem

(a) can be proved by taking B_n^* to be the set of typical sequences and using the AEP. To prove (b), note that

$$\begin{aligned} P^n(B_n \cap B_n^*) &= P^n(B_n) + P^n(B_n^*) - P^n(B_n \cup B_n^*) \\ &= 2 - P_e^{(n)} - P_*^{(n)} - P^n(B_n \cup B_n^*) \\ &\geq 1 - P_e^{(n)} - P_*^{(n)} \end{aligned} \tag{1.9}$$

where $P_e^{(n)}$ is the error for some code with vanishing error probability and $P_*^{(n)}$ is the error for code defined by the typical set. Note that the above error probabilities are vanishing. Thus, eventually, $P^n(B_n \cap B_n^*) > \frac{1}{2}$ which combined with typical inequalities used when considering the typical set, produces the desired result.

If a rate-optimal code is desired i.e. the asymptotic rate is exactly equal to the entropy, the error probability can be exponentially small i.e. $P_e^{(n)} \simeq 2^{-nD^*}$ and it is possible to calculate the largest possible value of D^*. **Action:** *Come back to this - what is this optimal value!*

1.3 Hypothesis Testing

A simple hypothesis test is considered: given data x_1^n produced by some memoryless source and two candidate distributions, P and Q on the finite alphabet A, we must decide whether the data was produced by P or Q. Any hypothesis test can be computing some region, $B_n \in A^n$ in which, if the data is lies in this region, we declare P and otherwise Q is declared. We seek to choose B_n to minimise the error probability.

$$e_1^{(n)} = Pr(\text{declare } P | X_1^n \sim Q^n) = Q^n(B_n) \tag{1.10}$$

$$e_2^{(n)} = Pr(\text{declare } Q | X_1^n \sim P^n) = P^n(B_n^c) \tag{1.11}$$

Error Probability: Tension

Note that the 'cost' of these errors is often asymmetry i.e. a false negative could be far more significant than a false negative. There is a tension between this error probabilities; minimising $e_1^{(n)}$ requires making B_n smaller but reducing $e_2^{(n)}$ requires making B_n^c smaller.

Stein's Lemma

Theorem 1.3.1 (Stein's Lemma) *Let* P *and* Q *be two different distributions over* A *such that* $D = D(P||Q)$ *is neither* 0 *nor* ∞*. Suppose that* **X** *is a memoryless source on* A*, distributed either to* P *or* Q*.*

(a): For any $\epsilon > 0$*, there is a hypothesis test with regions* $\{B_n^*\}$ *such that*

$$e_1^{(n)} \leq 2^{-n(D-\epsilon)} \tag{1.12}$$

for all n and with $e_2^{(n)} \to 0$ *as* $n \to \infty$*.*

(b): For a hypothesis test with regions $\{B_n\}$ *such that* $e_2^{(n)} \to 0$ *as* $n \to \infty$*, for any* $\epsilon > 0$*:*

$$e_1^{(n)} \geq 2^{-n(D+\epsilon+\frac{1}{n})} \tag{1.13}$$

The average log likelihood ratio can be expressed as

Prove Stein's Lemma.

$$\frac{1}{n}\log\frac{P^n(X_1^n)}{Q^n(X_1^n)} = \frac{1}{n}\sum_{i=1}^{n}\underbrace{\log\frac{P(X_i)}{Q(X_i)}}_{Z_i} \xrightarrow{P} D(P||Q) \tag{1.14}$$

The weak law of large numbers thus states that the above sum convergences in probability to the mean of Z_i, which (under P), is $D(P||Q)$. The likelihood-ratio typical set can be defined as:

$$B_n^* = \left\{x_1^n \in A^n : 2^{n(D-\epsilon)} \leq \frac{P^n(x_1^n)}{Q^n(x_1^n)} \leq 2^{n(D+\epsilon)}\right\} \tag{1.15}$$

This can be used to prove (a), taking the decision regions defined by the likelihood-ratio typical set. (Note that these regions are asymptotically optimal but may not be optimal for finite n; this proof is only about existence however). A bound on $e_1^{(n)}$ is obtained by applying typical bounds related to typical sets. The proof of part (b) follows the same format as for the fixed rate coding theorem.

Note that the assumptions that $D \neq 0$ is trivial, as it is equivalent to assuming that P and Q are different distributions. Assuming that $D \neq \infty$ means that every symbol possible under distribution P is also possible under Q; if this wasn't true, it would be very easy to perform the hypothesis test (e.g. waiting until we see symbols in one alphabet but not the other).

Are the assumptions valid?

Stein's lemma tells that if one of the error probability vanishes, the fastest rate at which the other error can decrease is given by $D(P||Q)$. Thus the larger the 'distance' between P and Q, the fewer data points are required to give a certain error probability and the easier it is to distinguish between the distributions. This motivates the interpretation of the relative entropy as a natural distance measure for discrete distributions.

Interpret Stein's Lemma

Theorem 1.3.2 (Neyman-Pearson Lemma) *For a hypothesis test between two distributions, P and Q, based on n data samples, consider the 'likelihood-ratio' decision region:*

Neyman-Pearson Lemma

$$B_{NP} = \left\{x_1^n \in A^n : \frac{P^n(x_1^n)}{Q^n(x_1^n)} \geq T\right\} \tag{1.16}$$

for some threshold $T > 0$ with error probabilities:

$$e_{1,NP}^{(n)} = Q^n(B_{NP}) \qquad e_{2,NP}^{(n)} = P^n(B_{NP}^c)$$

This decision region is optimal in the following sense. For any other decision region, $B_n \subset A^n$, if one the error probabilities, $e_2^{(2)}$, is better than that of B_{NP}, then the other one is worse:

$$e_2^{(n)} \leq e_{2,NP}^{(b)} \Rightarrow e_1^{(n)} \geq e_{1,NP}^{(b)}$$

Proof: Note the following inequality always holds:

Prove the Neyman-Pearson Lemma

$$\left[\mathbb{I}_{B_{NP}}(x_1^n) - \mathbb{I}_{B_n}(x_1^n)\right]\left[P^n(x_1^n) - TQ^n(x_1^n)\right] \geq 0 \tag{1.17}$$

It can be seen that this inequality holds simply by considering possible regions which x_1^n may be in. Then summing over all $x_1^n \in A^n$ yields, substituting and rearranging yields the desired result.

The empirical distribution induced by x_1^n on A^n is simply the empirical frequencies with with a appears in x_1^n. Formally:

Empirical Distribution

$$\hat{P}_n(a) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}_{\{a\}}(x_i) \tag{1.18}$$

The Neyman-Pearson decision region can equivalent be expressed as:

Relative Entropy Formulation

$$B_{NP} = \left\{x_1^N \in A^n : D(\hat{P}_n||Q) \geq D(\hat{P}_n||P) + T'\right\} \tag{1.19}$$

with $T' = \frac{1}{n}\log T$. Informally, this test computes the empirical distribution and chooses P if the distribution is closer to P (including a margin). By expanding the form for the logarithm of the likelihood, it can be shown that:

Prove the following line

$$\log\frac{P^n(x_1^n)}{Q^n(x_1^n)} = n[D(\hat{P}_n||Q) - D(\hat{P}_n||P)]$$

from which the reformulation directly follows.

1.4 Relative Entropy and Convexity

Convexity and Concavity

Let $F : E \to \mathbb{R}$ be a function defined on the convex set $E \subset \mathbb{R}^n$. Then:

1. f is convex if, for every pair of points, $x, y \in E$ with $x \neq y$, and every $\lambda \in [0,1]$,

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \tag{1.20}$$

2. f is concave if the inequality above is reversed, or equivalently if $-f$ is convex.

Prove link between convexity and the second derivative

A function is *strictly* convex/concave if the above inequality is strict. Convex functions have a positive second derivative, and a strictly positive second derivative implies strict convexity.

Jensen's Inequality

Theorem 1.4.1 (Jensen's Inequality) *For any convex function $f : R \to \mathbb{E}$ and any RV, X, taking values in E:*

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)] \tag{1.21}$$

If f is strictly convex, equality is achieved iff X is constant.

Log-Sum Inequality

Theorem 1.4.2 (Log-Sum Inequality) *For arbitrary non negative constants, $\{a_i, b_i\}_{i=1}^n$,*

$$\sum_i a_i \log \frac{a_i}{b_i} \geq \Big(\sum_i a_i\Big) \log \frac{\sum_i a_i}{\sum_i b_i} \tag{1.22}$$

Equality is achieved if the ratios a_i/b_i are the same $\forall i$.

Prove the Log-Sum Inequality

Let $A = \sum_i a_i$ and $B = \sum_i b_i$. Define a random variable, X as follows:

$$X = \frac{a_i}{b_i} \text{ with probability } \frac{b_i}{B} \forall i$$

Let $f(x) = x \log x$ with the convention that $f(0) = 0$, and note this is a strictly convex function. Simply applying Jensen's inequality, and considering the condition for equality in Jensen's inequality yields the desired result.

Prove the Data Processing Inequality

Theorem 1.4.3 (Data Processing Inequality) *Let $X \sim P_X$ and $X' \sim Q_X$ be two RVs over the same alphabet, A, defined by their PMFs. Let $f : A \to B$ be an arbitary function, and denote the corresponding output PMFs as $P_{f(X)}$ and $Q_{f(X)}$. Then:*

$$D(P_{f(X)} || Q_{f(X)}) \leq D(P_X || Q_X) \tag{1.23}$$

i.e. data processing cannot possibly make the two distributions more distinguishable.

The proof is straightforward by applying the log-sum inequality and writing the transformed distributions.

Definition 1.4.1 (Total Variation) *The total variation distance between two PMFs, P and Q, defined on the same alphabet, A, is*

$$||P - Q||_{TV} = \sum_{x \in A} |P(x) - Q(x)| \tag{1.24}$$

Alternative Total Variation Interpretation

Define the set $B = \{x \in A : P(x) > Q(x)\}$. The total variation can alternatively be interpreted as:

$$||P - Q||_{TV} = 2(P(B) - Q(B)) \tag{1.25}$$

The proof of the above is straightforward, and follows by definition of the set B.

Pinsker's Inequality

Theorem 1.4.4 (Pinsker's Inequality) *For any two PMFs, P and Q, on the same alphabet, A,*

$$||P - Q||_{TV}^2 \leq (2 \ln 2) D(P||Q) \tag{1.26}$$

Consider $P = \text{Bern}(p)$ and $Q = \text{Bern}(q)$. Without loss of generality, it can be assumed that $q \leq p$, since otherwise labels can be exchanged. Define:

Pinsker's Inequality: Bernouli Proof

$$\Delta(p,q) = (2\ln 2)D(P||Q) - ||P-Q||_{TV}^2 \tag{1.27}$$

Pinsker's inequality can be proved for this case by showing that $\Delta(p,q) > 0$. Since $\Delta(p,p) = 0$, this is equivalent to showing that:

$$\frac{\partial \Delta(p,q)}{\partial q} \leq 0$$

since q would be decreased from p. For the Bernoulli case, this can be shown by substituting in, expanding and considering the bounds on q and p.

How to expand to general random variables

For general random variables, the data-processing inequality is applied with $f(x) = \mathbb{I}_B(x)$ with $B = \{x \in A : P(x) > Q(x)\}$. Then, if $X \sim P$ and $X' \sim Q$, $f(X) \sim \text{Bern}(P(B))$ and $f(X') \sim \text{Bern}(Q(B))$, and $Q(B) \leq P(B)$. Then, applying the data processing inequality:

$$\begin{aligned}(2\ln 2)D(P||Q) &\geq (2\ln 2)D(P_f||Q_f)\\ &\geq ||P_f - Q_f||_{TV}^2\\ &= (2(P(B)-Q(B)))^2\\ &= ||P-Q||_{\text{TV}}^2\end{aligned} \tag{1.28}$$

Convexity of Relative Entropy

Theorem 1.4.5 (Convexity of Relative Entropy) *The relative entropy, $D(P||Q)$, is jointy convex. For any P_1, P_2, Q_1, Q_2 and $\lambda \in [0,1]$:*

$$D(\lambda P_1 + (1-\lambda)P_2||\lambda Q_1 + (1-\lambda)Q_2) \leq \lambda D(P_1||Q_1) + (1-\lambda)D(P_2||Q_2) \tag{1.29}$$

Prove Convexity

The proof for the above is similar to the proof of the data-processing inequality. .

Note that if P is a distribution on an alphabet, A, and U is the uniform distribution over A, then:

$$D(P||U) = \log|A| - H[P] \tag{1.30}$$

This, with the convexity of relative entropy, directly implies that the following corollary.

Corollary 1.4.5.1 (Concavity of Entropy) *The entropy, $H[P]$, is a concave function on the set of all PMFs, P, on a finite alphabet.*

1.5 Poisson Approximation

Note: In this section, it is convenient to use entropies defined by the natural log, denoted by subscript e.

Binomial to Poisson

Suppose that λ events, on average, occur in some time window. Let $X_1, \ldots, X_n$ be a sequence of RVs with $X_i \overset{iid}{\sim} \text{Bern}(\frac{\lambda}{n})$. Then, their sum converges to a Poisson distribution.

$$Pr(S_n = k) \overset{n\to\infty}{\to} \exp[-\lambda]\frac{\lambda^k}{k!} \tag{1.31}$$

In general, let $\{X_i\}_{i=1}^N$ be a sequence of RVs with $X_i \sim \text{Bern}(p_i)$. Then, the distribution of the sum of the RVs, S_n is close to $\text{Po}(\lambda)$ with $\lambda = \mathbb{E}[S_n]$, provided that each p_i is 'small' and the X_i are 'weakly' dependent.

Theorem 1.5.1 (Poisson Approximation I) *Suppose $S_n = X_1 + \cdots + X_n$ with $X_i \overset{iid}{\sim} Bern(p_i)$. Then*

$$D_e(P_{S_n}||Po(\lambda)) \leq \sum_i p_i^2 \tag{1.32}$$

with $\lambda = \mathbb{E}[S_n] = \sum_i p_i$.

Show this proves the Binomial-to-Poisson Convergence

Note that this theorem implies the binomial-to-Poisson convergence. Additionally, this theorem is not asymptotic and we form an equality that holds for all n. It also provides an easily computable, quantitative bound on the difference between P_{S_n} and the limiting Poisson distribution, without assuming that each p_i is equal.

Theorem 1.5.2 (Poisson Approximation II) *Suppose $S_n = X_1 + \cdots + X_n$ with $X_i \sim Bern(p_i)$, allowing dependence between the Bernoulli RVs. Then*

$$D_e(P_{S_n}||Po(\lambda)) \leq \sum_i p_i^2 + \sum_i H_e[X_i] - H_e(X_1^n) \tag{1.33}$$

with $\lambda = \mathbb{E}[S_n] = \sum_i p_i$.

Poisson Proof

Proof: $A = \{0, 1, \ldots\}$ and $Z_i \overset{iid}{\sim} \text{Po}(p_i)$, meaning that the distribution of the sum $T_n = \sum_i Z_i$ is $\text{Po}(\lambda)$. Define $f : A^n \to A$ as follows:

$$f(x_1^n) = \sum_{i=1}^n x_i \tag{1.34}$$

meaning that $S_n = f(X_1^n)$ and $T_n = f(Z_1^n)$.

Poisson Approximation Proof

$$D_e(P_{S_n}||\underbrace{\text{Po}(\lambda)}_{P_{T_n}}) = D_e(P_{f(X_1^n)}||P_{f(Z_1^n)}) \leq D_e(P_{X_1^n}||P_{Z_1^n}) \tag{1.35}$$

Next, note that for every $p \in [0, 1]$:

$$D_e(\text{Bern}(p)||\text{Po}(p)) \leq p^2 \tag{1.36}$$

Substituting in for the definition of relative entropy, and multiplying and dividing by the product of the marginal distributions yields the desired result.

This results gives a precise quantitative definition of what small p_i corresponds to i.e. p_i must be small enough to have a small sum of squares. Additionally, 'weakly dependent' is also quantified. Note that the term $\sum_i H_e[X_i] - H_e(X_1^n)$ is non-negative and only 0 if independent.

1.6 Estimation and Information

Problem Setup

IID data, X_1^n is generated by some unknown distribution, P, which is assumed to belong to some parametric family of distributions, $\mathcal{P} = \{P_\theta : \theta \in \Theta\}$. The parameter space, Θ, is usually some interval of real numbers or a subset of $\mathbb{R}^d$. Assume that the true distribution lies within this parameter space, $P = P_{\theta^*}$. We hope to find a good estimator for θ^*, $\hat{\theta} = \hat{\theta}(X_1^n)$. Note that this estimator is a function of the data i.e. $\hat{\theta} : A^n \to \Theta$, and it is a random variable.

Bias-Variance Tradeoff

A reasonable approach is to try and minimise the mean squared error.

$$\begin{aligned} \text{MSE}(\hat{\theta}; \theta) &= \mathbb{E}_{P_\theta}[(\hat{\theta} - \theta)^2] \\ &= \underbrace{\mathbb{E}_{P_\theta}[(\hat{\theta} - \mathbb{E}_{P_\theta}[\hat{\theta}])^2]}_{\text{var}[\hat{\theta}]} + \underbrace{(\mathbb{E}_{P_\theta}[\hat{\theta} - \theta])^2}_{\text{bias}(\hat{\theta};\theta)^2} \end{aligned} \tag{1.37}$$

In order to have a small mean squared error, we need both a small bias and small variance. Typically, there is a tension between the two terms known.

1.6.1 Unbiasedness and Maximum Likelihood

It may seem sensible to restrict attentions to unbiased estimators and choose the estimator with the smallest possible variance, but in certain cases (such as for the entropy), unbiased estimators do not exist or may lead to implausible estimates.

Maximum Likelihood

The *Maximum Likelihood Principle* states that the best estimate is given by the parameter that maximises the likelihood of the data:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta \in \Theta} P_\theta^n(x_1^n) \tag{1.38}$$

$$= \arg\min_{\theta \in \Theta} D(\hat{P}_n||P_\theta) \tag{1.39}$$

Proof: MLE and Relative Entropy

The MLE estimate can equivalently be expressed as as the closest parametric distribution to the empirical distribution. This can be proved by rewriting the negative log-likelihood.

1.6.2 Fisher Information

When forming maximum likelihood estimates, the relative entropy is differentiated with respect to θ. The behaviour of the relative entropy, $D_e(P_\theta||P_{\theta'})$, is now investigated for θ close to θ'. Define $\Delta(x) = P_\theta(x) - P_{\theta'}(x)$.

$$\begin{aligned} D_e(P_\theta||P_{\theta'}) &= \sum_x (P_{\theta'}(x) + \Delta(x)) \ln\left[1 + \frac{\Delta(x)}{P_{\theta'}(x)}\right] \\ &\simeq \sum_x (P_{\theta'}(x) + \Delta(x))\left[\frac{\Delta(x)}{P_{\theta'}(x)} - \frac{1}{2}\left(\frac{\Delta(x)}{P_{\theta'}(x)}\right)^2\right] \\ &\simeq \frac{1}{2}\sum_x P_{\theta'}(x)\left(\frac{P_\theta(x) - P_{\theta'}(x)}{P_{\theta'}(x)}\right)^2 \end{aligned}$$

assuming that $\theta - \theta'$ is small and thus that Δ is small and noting that $\sum_x \Delta(x) = 0$. Then, assuming that the derivatives of $P_\theta(x)$ exist with respect to θ, $P_\theta(x) - P_{\theta'}(x) \simeq (\theta - \theta')P'_\theta$. Therefore:

$$D_e(P_\theta||P'_\theta) \simeq \frac{1}{2}(\theta - \theta')^2 \mathbb{E}\left[\left(\frac{P'_\theta(X)}{P_\theta(X)}\right)^2\right] \tag{1.40}$$

Thus $D_e(P_\theta||P'_\theta)$ is locally quadratic in θ.

Fisher Information

Definition 1.6.1 (Fisher Information) *Let $\mathcal{P} = \{P_\theta : \theta \in [a,b]\}$ be a one-dimensional parametric family such that the derivatives P'_θ of P_θ with respect to θ exist for all $\theta \in [a,b]$. The Fisher Information, $J(\theta)$, for this parametric family is defined as:*

$$J(\theta) = \mathbb{E}_{P_\theta}\left[\left(\frac{P'_\theta(X)}{P_\theta(X)}\right)^2\right] \tag{1.41}$$

Alternative Fisher Information Expressions

The Fisher Information can also be expressed as:

$$J(\theta) = \mathbb{E}_{P_\theta}\left[\left(\frac{\partial}{\partial\theta}\{\ln P_\theta(X)\}\right)^2\right] \tag{1.42}$$

Alternatively:

Show Alternative Expression

$$J(\theta) = \mathrm{var}_{P_\theta}\left[\underbrace{\frac{\partial}{\partial\theta}\{\ln P_\theta(X)\}}_{\rho(\theta;X)}\right] \tag{1.43}$$

where $\rho(\theta;X)$ is known as the score function. To prove, consider the mean of the score function.

Cramér-Rao Bound

Theorem 1.6.1 (Cramér-Rao Bound for Any Estimator) *The MSE of any estimator, $\hat{\theta}$, satisfies:*

$$var_{P_\theta}(\hat{\theta}) \geq \frac{[1 + bias'(\hat{\theta};\theta)]}{J(\theta)} \tag{1.44}$$

where $bias'(\hat{\theta};\theta)$ is the derivative of the bias with respect to θ.

For an unbiased estimator, the derivative of the bias is zero.

Cramér-Rao Bound Proof

Proof: Consider the following:

$$J(\theta)\,\mathrm{var}_{P_\theta}(\hat{\theta}) = \mathrm{var}_{P_\theta}(\rho(\theta;X))\,\mathrm{var}_{P_\theta}(\hat{\theta}) \leq \mathbb{E}\left[(\rho(\theta;X) - \underbrace{\mathbb{E}_{P_\theta}[\rho(\theta;X)]}_{0})(\theta(X) - \mathbb{E}_{P_\theta}[\hat{\theta}])\right] \tag{1.45}$$

since the correlation coefficient between two RVs is bounded by 1 i.e.

$$\mathrm{corr}(X,Y) = \frac{\mathrm{cov}(X,Y)}{\sqrt{\mathrm{var}(X)\mathrm{var}(Y)}} = \frac{\mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]}{\sqrt{\mathrm{var}(X)\mathrm{var}(Y)}} \leq 1 \tag{1.46}$$

Suppose that X_1^n is IID according to some distribution, P_θ. It can be shown that:

Proof: Fisher Information for IID Samples

$$J_n(\theta) = nJ(\theta) \tag{1.47}$$

This is straightforward to show by the independence of each datapoint. This gives a natural extension of the Cramér-Rao Bound.

Chapter 2

Practical Number Theory & Algebra

2.1 Number Theory

Theorem 2.1.1 (Euclid's Division Theorem) *For any integers $n, d \neq 0$ there exist unique integers, q, r such that:*

Euclid's Division Theorem

$$n = qd + r \text{ with } 0 \leq r < |d| \tag{2.1}$$

This theorem can be shown by showing that there exists a pair, (q', r') with $n = qd + r$, and that this can be used to recursively generate pairs until one is found which satisfies the condition. Additionally, it is shown that this pair is unique (try showing this yourself).

Proof Outline

Theorem 2.1.2 (Properties of Remainders) *Renote the remainder of n when it is divided by d as $R_d(n)$. Remainders have the following properties:*

Properties of Remainders

Try proving these!

1. $R_d(n + kd) = R_d(n)$
2. $R_d(k + n) = R_d(R_d(k) + R_d(n))$
3. $R_d(kn) = R_d(R_d(k) \cdot R_d(n))$

Definition 2.1.1 (Greatest Common Divisor) *The greatest common divisor between integers n_1, n_2 is the largest integer d which divides both n_1 and n_2. Denote the greatest common divisor as $gcd(n_1, n_2)$.*

Note that signs are irrelevant for the GCD.

Theorem 2.1.3 (Fundamental Property of GCDs) *For any integers, n_1, n_2 and k:*

$$gcd(n_1 + kn_2, n_2) = gcd(n_1, n_2) \tag{2.2}$$

Any number that divides both n_1 and n_2 will also divide $n_1 + kn_2$ by the property of remainders, giving us this fundamental property.

Imagine picking k as the negative quotient of n_1/n_2. Then, the above property implies that:

Euclid's Algorithm

$$\gcd(n_1, n_2) = \gcd(R_{n_2}(n_1), n_2) \tag{2.3}$$

which provides an algorithm for computing the GCD of two numbers:

1. Label the numbers such that $n_1 > n_2$.
2. Compute $r = R_{n_2}(n_1)$ and assign $n_1 \leftarrow n_2$, $n_2 \leftarrow r$.
3. If $n_2 = 0$, terminate and output n_1. Otherwise, repeat.

Theorem 2.1.4 (Even & Odd Properties of GCDs) *For any integers, n_1 and n_2:*

1. *If n_1, n_2 are both even, then $gcd(n_1, n_2) = 2\, gcd(n_1/2, n_2/2)$.*

2. *If n_1 is even and n_2 is odd, then $gcd(n_1, n_2) = gcd(n_1/2, n_2)$.*
3. *If n_1, n_2 are odd, then $gcd(n_1, n_2) = gcd(\frac{n_1 - n_2}{2}, n_2)$.*

Stein's Algorithm

Applying the above properties gives *Stein's Algorithm* which is more efficient that Euclid's algorithm.

Theorem 2.1.5 (Greatest Common Divisor Theorem) *For any integers n_1 and n_2 not both zero, there exist (not necessarily unique) integers such that:*

$$gcd(n_1, n_2) = an_1 + bn_2 \tag{2.4}$$

Try Proving this Yourself

The proof of this theorem considers $d' = an_1 + bn_2$ which is defined as the smallest possible integer combination of n_1 and n_2. It can be shown that d' divides both n_1 and n_2 exactly, but also that the GCD divides d', meaning that d' must be the GCD. In many cases, we are very interested in calculating the values of a and b.

Extended Euclid's Algorithm

Both Stein's Algorithm and Euclid's algorithm can be extended to also compute the pair (a, b). The key idea is to maintain two pairs (a_1, b_1) and (a_2, b_2) such that, after k iterations:

$$a_1^{(k)} n_1^{(0)} + b_1^{(k)} n_2^{(0)} = n_1^{(k)} \tag{2.5}$$
$$a_2^{(k)} n_1^{(0)} + b_2^{(k)} n_2^{(0)} = n_2^{(k)} \tag{2.6}$$

Show Recursive Relationships

The pairs are initialised as $(a_1^{(0)}, b_1^{(0)}) = (1, 0)$ and $(a_2^{(0)}, b_2^{(0)}) = (0, 1)$. For Euclid's algorithm, the following recursive relationships hold:

$$a_1^{(k+1)} = a_2^{(k)} \tag{2.7}$$
$$b_1^{(k+1)} = b_2^{(k)} \tag{2.8}$$
$$a_2^{(k+1)} = a_1^{(k)} - q a_2^{(k)} \tag{2.9}$$
$$b_2^{(k+1)} = b_1^{(k)} - q b_2^{(k)} \tag{2.10}$$

The algorithm ends by setting $(a, b) = (a_1, b_1)$ since $n_1^{(k^*)} = \gcd(n_1^{(0)}, n_2^{(0)})$ and $n_2^{(k^*)} = 0$.

A number, $p > 1$, is prime iff it is only divisible by 1 and itself. If $n < p$, then $\gcd(p, n) = 1$. Note that two numbers are called co-prime if $\gcd(n_1, n_2) = 1$.

Theorem 2.1.6 (Fundamental Theorem of Arithmetic) *Every integer, $n > 1$, is either prime or can be **uniquely** expressed as a product of prime numbers.*

Any non-prime number, can be expressed as a product of two integers less than the factor. This can be repeatedly endlessly for any non-prime factor, and given that there are a finite number of integers, that we can express any number as a product of primes.

Uniqueness of Prime Decomposition

If n has two distinct prime factorisations, either there is prime in one factorisation which is not present in the other or a factor appears an unequal number of times in each factorisation. Denote the unique factor as p'. p' must divide n, meaning that it must divide one of the factors in the other factorisation or be expressible as a product of factors, meaning that p' is not prime or one of the other factors is prime. If the factor p' appears k_1 times in one factorisation and k_2 times in the other, the same argument can be made for $p^{\max(k_1, k_2)}$, meaning that the factorisation of primes must be unique.

Theorem 2.1.7 *There are infinitely many primes.*

Theorem 2.1.8 (Prime Number Theorem) *For any integer, let $\pi(n)$ be the number of primes up to and including n. Then:*

$$\lim_{n \to \infty} \frac{\pi(n)}{n / \ln(n)} = 1 \tag{2.11}$$

i.e. the number of primes grows roughly as $n / \ln n$

Chinese Remainder Theorem

Theorem 2.1.9 (Chinese Remainder Theorem) *Let $\{m_i\}_{i=1}^k$ be pairwise co-prime integers (or moduli). Let $m = \prod_i m_i$. Then, for any choice of residues, $\{r_i\}_{i=1}^k$ such that $0 \leq r_k < m_j$ there exists a **unique** non-negative integer, $n < m$, for which:*

$$R_{m_j}(n) = r_j \tag{2.12}$$

i.e. for co-prime moduli, the residues uniquely define a number.

Uniqueness can be shown by assuming there are two integers, n and n' which satisfy the condition. Considering $R_{m_j}(n-n')$ shows that $n-n'$ is divided by all the of relatively prime module, meaning that $n-n'$ can be expressed as some multiple of the product of the moduli. However, $n-n' < m$, meaning that $n-n'=0$.

Uniqueness

There are m choices for n and there are $m = \prod_j m_j$ choices for the residue vector. Since each residue vector uniquely corresponds to a number, and each number has residues, there must be an integer for every choice of residues.

Existence

It is clearly easy to calculate residues from a number n, but what about the other way around? Define $u_j = \frac{m}{m_j}$ and note that $\gcd(u_j, m_j) = 1$ since all the moduli are co-prime. Then, we can write:

From Residues to n

$$\gcd(m_j, u_j) = a_j m_j + b_j u_j = 1 \tag{2.13}$$

Additionally, $R_{m_j}(a_j m_j + b_j u_j) = R_{m_j}(b_j u_j) = 1$. Then, $R_{m_j}(\alpha b_j u_j) = \alpha$ for $\alpha < m_j$. Also note that $R_{m_t}(b_j u_j) = 0$ since m_t divides u_t. Therefore:

$$n = R_m\Big[\sum_{j=1}^{k} r_j b_j u_j\Big] \tag{2.14}$$

has the correct residues, which can easily be verified by considering $R_{m_j}(n)$.

Verify Correct Residues

2.2 Algebra

A single operation algebraic system consists of a set, A, and operation $\star$ and is denoted as $\langle S, \star \rangle$.

Definitions of Systems

Conditions	**Statement**	Example	$\langle S, \star \rangle$
Closure	$\forall a, b \in S,\ c = a \star b \in S$	$\langle \mathbb{R}^n, \times \rangle$	
Associative Law	$\forall a, b, c \in S,\ (a \star b) \star c = a \star (b \star c)$	$\langle \mathbb{N}^*, + \rangle$	Semi-Group
Neural Element	$\exists e \in S$ s.t. $e \star a = a \star e = a$.	$\langle \mathbb{Z}_m, \odot \rangle$	Monoid
Inverse	$\forall a \in S,\ \exists b \in S$ s.t. $a \star b = b \star a = e$		Group
Commutativity	$\forall a, b \in S,\ a \star b = b \star a$	$\langle \mathbb{Z}_p, \oplus \rangle$	Abelian Group

Table 2.1: Algebraic Systems with one operation.

Denote $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ and $\mathbb{Z}_m^* = \{1, \ldots, m-1\}$. In this course, we are mostly interested in operations modulo m on $\mathbb{Z}_m$:

$$a \oplus b \triangleq R_m(a+b) \qquad a \odot b \triangleq R_m(a \cdot b) \tag{2.15}$$

Note that $\langle \mathbb{Z}_p, \oplus \rangle$, $\langle \mathbb{Z}_p^*, \odot \rangle$ are both abelian groups when p is prime.

$\langle S, + \rangle$	$\langle S, \cdot \rangle$	**Conditions**	$\langle S, +, \cdot \rangle$
Abelian Group	Abelian Group	Distributive Law	Ring
		$\forall a, b, c \in S, a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ and $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$	
Abelian Group	Abelian Group	$\langle S \setminus \{e_+, \cdot\} \rangle$ is an Abelian Group	Field

$\langle \mathbb{Z}_m, \oplus, \odot \rangle$ is a ring while $\langle \mathbb{Z}_p, \oplus, \odot \rangle$ is a field since the prime ensures that non-zero element has an inverse. It is easy to show that the distributive law holds. A finite ring or field has a finite number of elements in S.

Proof Distributive Law

Finite Ring/Field

2.2.1 Properties of Monoids

Theorem 2.2.1 (Uniqueness of the Neutral Element) *The neural element, e, in a monoid must be unique.*

Prove Uniqueness of e

Uniqueness is easy to show since $e_1 = e_1 \star e_2 = e_2$.

Theorem 2.2.2 (Uniqueness of the Inverse) *Consider the monoid $\langle S, \star \rangle$. If the element $a \in S$ has inverse $a^{-1} = b$, then the inverse is unique.*

Show Monoid's have Unique Inverses

This can be proven by assuming that a has two inverses, b and c, and that b and c must be equal.

Theorem 2.2.3 (Invertible Elements of $\langle \mathbb{Z}_m, \odot \rangle$) *An element, $a \in \mathbb{Z}_m$, is invertible iff $gcd(u, m) = 1$.*

Assume that a has an inverse, b and recall the definition of the inverse:

$$a \cdot b = 1 = R_m(ab) = ab - qm \tag{2.16}$$

Proof

Note that $\gcd(a, m)$ divides $ab - qm = 1$, and thus $\gcd(a, m) = 1$ since 1 is the only non-negative number that divides 1. Then, $xa + ym = 1$ by the GCD theorem, for some x, y. Therefore

$$R_m(xa + ym) = R_m(xa) = R_m(x) \odot a = 1 \tag{2.17}$$

meaning that the extended Euclid/Stein algorithm can be used to compute the inverse of an element.

Euler's Function

Definition 2.2.1 (Euler's Function) *Euler's function gives the number of elements of $\langle \mathbb{Z}_m, \odot \rangle$ which are invertible:*

$$\varphi(m) \triangleq |\{k : 0 \leq k < m,\ gcd(k, m) = 1\}| \tag{2.18}$$

$$= \prod_{j=1}^{k} (p_j - 1) p_j^{e_j - 1} \tag{2.19}$$

for the composite $m = p_1^{e_1} \dots p_k^{e_k}$.

For Primes and Simple Composites

It is clear that $\varphi(p) = p - 1$ since all elements other than 0 are invertible. For $\varphi(p^e)$, we can express $k = qp + r$ with $q \in \{0, 1, \dots, p^{e-1}\}$ and $r \in \{0, 1, \dots, p - 1\}$. We require $\gcd(k, p^e) = 1$ and since the only factor of p^e is p, there are p^{e-1} choices of q and $p - 1$ choices for r giving $\varphi(p^e) = p^{e-1}(p - 1)$.

Multiplication and Inversion CRT

Theorem 2.2.4 (Multiplication and Inversion Property of the CRT) *For a set of co-prime module, $\{m_i\}$, and $m = \prod m_i$, let $a, b \in \mathbb{Z}_m$ have residuals $\{a_i\}$ and $\{b_i\}$. Then, the residuals of the product, $a \odot_m b$ are the products of the residuals:*

$$R_{m_j}(a \odot_m b) = a_j \odot_{m_j} b_j \tag{2.20}$$

i.e. the j-th residual of the product is the product (in $\mathbb{Z}_{m_j}$) of the residuals.

Product Proof & Euler's Function Proof

The statement is easy to verify via substitution. **Note:** this can be used to show the form of Euler's Function for composite m. For element $a \in \mathbb{Z}_m$, the inverse, b satisfies $a \odot_m b = 1$. Let $m_j = p_j^{e_j}$, Then, by the above theorem $R_{m_j}(a \odot_m b) = 1\ \forall j \Rightarrow a_j \odot_{m_j} b_j = 1\ \forall j$. Thus, a has an inverse iff a_j has an inverse in $\mathbb{Z}_{m_j}\ \forall j$. The composite equation follows since there are $\varphi(p_j^{e_j}) = p_j^{e_j - 1}(p_j)$ possible values of a_j.

Additionally, if a prime decomposition of m is known, it is easy to check if an element of $\mathbb{Z}_m$ is invertible by computing the residuals and checking if they are invertible.

2.2.2 Properties of Groups

The difference between monoids and groups is that every element in a group is invertible. Thus, clearly for any monoid $\langle M, \star \rangle$, the system $\langle M^\star, \star \rangle$ where $M^\star$ is the set of invertible elements in M is a group; associativity is inherited from the monoid, the neutral element is invertible and is in $M^\star$. Thus, we need only verify closure. Consider $a, b \in M^\star$:

Verify Closure

$$b^{-1} \star a^{-1} \star a \star b = e \tag{2.21}$$

Then, $b^{-1} \star a^{-1}$ is the inverse of $a \star b$. Since the monoid satisfies closure, $b^{-1} \star a^{-1} \in M$, and thus $a \star b$ has an inverse in M, meaning that $a \star b \in M^\star$, verifying closure. Note that for a group, $\langle G, \star \rangle$, the equation:

$$a \star x = b \tag{2.22}$$

for $a, b \in G$ always has a unique solution, which is not the case for a monoid. Additionally, define exponentiation of an element in a group as:

$$a^n \triangleq \underbrace{(a \star a \star \cdots \star a)}_{n \text{ times}} \tag{2.23}$$

and so the usual rules for exponentiation apply.

Order of an Element

The order of an element, $a \in G$, is denoted as $\text{ord}(a)$ and is the *smallest* number n such that $a^n = e$. In a finite group, each element has a well defined order since there are a finite supply of elements in the group meaning that if a^n is computed, eventually elements must repeat. Let $n_1 < n_2$ such that $a^{n_1} = a^{n_2}$. Then, $a^{n_2 - n_1} = e$ meaning there exists $n = n_2 - n_1$ such that $a^n = e$, and thus there must be a smallest value for the exponent.

Cyclic Group

Consider the algebraic system $\langle \{a^1, a^2, \ldots, a^n\}, \star \rangle$. This is a group since it contains the neural element and it is closed since:

$$a^{n_1} \star a^{n_2} = a^{qn + R_n(n_1+n_2)} = e^q \star a^{R_n(n_1+n_2)} = a^{R_n(n_1+n_2)} \tag{2.24}$$

This is called the *cyclic group* generated by the *generator*, a. Consider the cyclic group generated by (a^m). The order of this cyclic group, α is the smallest integer such that:

$$\alpha = \arg\min_{u,q} mu = qn \tag{2.25}$$

If $\gcd(m, n) = 1$, then the order of the group is n and the same cyclic group will be generated. Therefore, there are $\varphi(n)$ generators in a cycle group. In general, for an element $b = a^m$ in the cycle group generated by a,

$$\text{ord}(b) = \frac{n}{\gcd(m, n)} \tag{2.26}$$

Order of a Group

which can be seen by considering the prime factor of m and u. Since the size of the group generated by a is $\text{ord}(a)$, the order of a group refers to the number of elements in a group.

Cauchy's Theorem

Theorem 2.2.5 (Cauchy's Theorem) *Let G be a finite group and let p be a prime number which divides the order of G. Then, there exists an element in G with order p.*

For a group $\langle G, \star \rangle$ and a subset, $H \subseteq G$, if $\langle H, \star \rangle$ is a group, then it is called a sub-group of $\langle G, \star \rangle$. It must be verified that this sub-group satisfies closure under $\star$ (and inversion), and thus it must contain the neutral element. For any element, $a \in G$, the cyclic group generated by a is a sub-group of G.

Langrange's Theorem

Theorem 2.2.6 (Lagrange's Theorem) *The order of any subgroup of a finite group divides the order of the group.*

Proof

Let H be a subgroup of $\langle G, \star \rangle$. Consider an element, $a_i \notin H$, and consider the set $a_i \star H = \{a_i \star h : h \in H\}$. Then, note that:

(a): $a_i \in a_i \star H$ since $e \in H$ (since H is a sub-group).

(b): $|a_i \star H| = |H|$ since a_i has an inverse, so $a_i \star h_1 = a_i \star h_2 \Rightarrow h_1 = h_2$ for $h_1, h_2 \in H$.

Why don't the subsets intersect?

(c): If $a_i \star H$ intersects $a_j \star H$, then $a_i \star H = a_j \star H$. If there is an intersection, then $a_i \star h_1 = a_j \star h_2$ for some $h_1, h_2 \in H$. Therefore, $a_j = a_i \star h_1 \star h_2^{-1}$ and $a_j \star H = a_i \star h_1 \star h_2^{-1} \star H = a_j \star H$ since H satisfies closure and each element of h has an inverse.

Therefore, G can be partitioned into unique non-intersecting subsets, H, $a_1 \star H, \ldots$ and more subsets can be generated as long as elements of G remain. Since each subset has order $|H|$, and the subsets must in total have the same number of elements as G, $|G| = k|H|$.

A consequence of this is that the order of an element must divide the order of the group it resides in. Additionally, the following theorems are corollaries of Lagrange's theorem.

Euler's Theorem

Theorem 2.2.7 (Euler's Theorem) *For any **invertible** element $a \in \langle \mathbb{Z}_m, \odot \rangle$:*

$$a^{\varphi(m)} = 1 \tag{2.27}$$

since the order of invertible elements is $\varphi(m)$ and $\varphi(m) = k \cdot ord(a)$.

Fermat's Theorem

Theorem 2.2.8 (Fermat's Theorem) *For any prime number, p, and any $a < p$*

$$R_p(a^{p-1}) = 1 \tag{2.28}$$

In fact, this is true for any a which is not divisible by p. This can be shown by considering $b > p$ such that $R_p(b) = a$, and a is necessarily smaller than p.

2.3 Finite Fields and Vector Spaces

Galois Field

Definition 2.3.1 (Galois Field) *The algebraic system $\langle \mathbb{Z}_p, \oplus, \odot \rangle$ is called the* Galois Field *of order p, and is denoted as GF(p).*

There are no concerns with numerical stability when operating in finite fields, which is a crucial reason that excellent error correction codes exist in these fields. Additionally, the concept of vector spaces and dimensions translate over directly from real numbers, but a vector space in a finite field has a finite number of elements.

Can we get a non-prime number of elements?

There are a number of other finite fields with prime order, but these are all effectively the same as the Galois field. We can get a non-prime number of elements by *extending* the base field, GF(p), to form an *extension field*, GF(q) where $q = p^k$.

Polynomials over GF(p)

One simple way to form the extension field, GF($q = p^k$) is to consider a polynomial. An element $a(X) \in \text{GF}(p^k)$ is:

$$a(X) = a_0 + a_1 X + a_2 X^2 + \ldots + a_{k-1} X^{k-1} \tag{2.29}$$

where $a_i \in \text{GF}(p)$. The symbol X has no meaning, and is simply an auxiliary variable used to define our operations. Addition in this extension field is simply the addition of polynomials, where the addition of the coefficients occurs over GF(p).

Multiplication in the Extension Field

Multiplication in the extension field is defined as polynomial multiplication *modulo an irreducible polynomial, $\pi(X)$, of degree k*. An irreducible polynomial cannot be expressed as a product of two polynomials of a lesser degree i.e. it is the polynomial equivalent of a prime. If there exists a polynomial $\alpha(X)$ of order $p^k - 1$ under multiplication module $\pi(X)$, then $\pi(X)$ is known as a *primitive polynomial*. Whilst we could use any irreducible polynomial to define an extension field, using a primitive polynomial gives a cyclic field. We can find an irreducible polynomial in a similar way to finding a prime number; consider increasing degrees of polynomial, and eliminate those which are obtained by polynomials of a lower degree. **Note:** polynomials which do not have a leading 1 are clearly divisible by X, and thus can be eliminated immediately. To verify that a polynomial is indeed primitive, we only need to find an element with order, but recall Langrange's theorem which says that the order of a sub-group must divide the order of the group which makes this easier to check.

Primitive Polynomial

Finding an Irreducible Polynomial

Verifying a Polynomial is Primitive

Multiplication Tricks

The main trick when multiplying polynomials is to multiply one power at a time, subtracting factors of $\pi(X)$ to get back within the correct range. Another easy way of performing the multiplication is to pick a generator (an element α with maximum order $p^k - 1$) and to pre-compute all of it's powers. Given a look-up table with these powers, then:

$$a(X) \cdot B(X) = \alpha^{k_a} \alpha^{k_b} = \alpha^{R_{p^k-1}(k_a + k_b)} \tag{2.30}$$

Note that if the order of the multiplicative abelian group is prime, then any element is a generator. We typically pick primitive polynomials such that $\alpha = X$ is a generator.

Vectors over GF(p)

The second approach to defining operations in extension fields is considering elements of GF(p^k) as k-ary row vectors where each element is in GF(p). Note that it is assumed that X is a generator. Addition is performed component-wise but to perform multiplication, we required the *companion matrix*, **B**:

$$\mathbf{a} \cdot \mathbf{b} = \begin{bmatrix} a_1 & \ldots & a_k \end{bmatrix} \cdot \underbrace{\begin{bmatrix} \mathbf{b} \\ X\mathbf{b} \\ \vdots \\ X^{k-1}\mathbf{b} \end{bmatrix}}_{\mathbf{B}} \tag{2.31}$$

so the rows of the companion matrix are the pre-computed produces of **b** with powers of X. Note that:

$$\mathbf{a} \cdot \mathbf{b} \cdot \mathbf{c} = (\mathbf{aB})\mathbf{c} = \mathbf{a}(\mathbf{BC}) \tag{2.32}$$

so **BC** must be the companion matrix of $\mathbf{b} \cdot \mathbf{c}$.

The companion matrix of X, **X**, always has the form: *Companion Matrix of X*

$$\mathbf{X} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ & & -\pi^* & & \end{bmatrix} \tag{2.33}$$

This provides an easy way to compute the multiplication table; the companion matrix for any element, X^k is $\mathbf{X}^k$ modulo p. If extended to a field where X is not a generator, if we express all field elements as polynomials in the generator polynomial the same companion matrix notation may be applied.

Any matrix operation in GF(p^k) is in fact a matrix operation in $GF(p)$ where each element in a GF(p^k) matrix is replaced by its companion matrix. As a result, linear codes over GF(p^k) are essentially codes over GF(p). *Matrix Operations*

Chapter 3

Linear Codes over GF(q)

The task of coding is to provide arbitary reliability at data rates approaching capacity for discrete-input memoryless channels, distinct from the task of error correction. However, in certain situations, the aim is indeed to correct errors with some provable error performance. For capacity achieving codes, their error correcting performance must often be calculated through simulation and thus for very small error probabilities, layers of coding which provide guarantees on reducing errors are often included. Whilst these codes are no longer rate-optimal, provable error performance can be worth the compromise.

3.1 Linear Coding Fundamentals

(N,K) Error Correction Code

Definition 3.1.1 (q-ary (N,K) Error Correction Code) *A q-ary (N,K) error correct code is a set, $\mathcal{C}$, of q^K q-ary row vectors, $\mathbf{x}$, of length N with $N > K$.*

The principle is to add redundancy to allow errors to be connected. We distinguish between the code itself, which is simply defined by $\mathcal{C}$, and the encoding operation. *Linear Codes* can be generated using matrix operations:

Linear Codes

$$\mathbf{x} = \mathbf{uG} \tag{3.1}$$

$\mathbf{G}$ is a $K \times N$ matrix known as the *encoder matrix*. The code is then a vector subspace of $GF(q)^N$, and the rows of $\mathbf{G}$ are the basis vectors for $\mathcal{C}$. Clearly, any K linearly independent codewords could serve as a basis for this vector space, defining a different encoding for the same code. Additionally, switching rows, multiplying rows by a constant and adding rows will not affect the code. Thus, by using elementary row operations, we can create a *systematic encoder matrix*:

Systematic Encoder

$$\mathbf{G}_s = \begin{bmatrix} \mathbf{I}_K & \mathbf{P}_{K\times(N-K)} \end{bmatrix} \tag{3.2}$$

Thus the first K symbols of a codeword are the information symbols. Note that the probability that a given word is decoded to the wrong codeword (or not decoded) at all is independent of the encoding, provided each codeword is equally likely. However, this form gives a better *symbol error rate* as the wrong codeword will be fairly similar to the transmitted codeword; for systematic encoding, small errors in codewords result in small errors in information words.

A code can also be define using a parity check matrix, a set of $N-K$ vectors which each codeword is orthogonal too i.e.

Parity Check Matrix

$$\mathbf{xH}^T = 0 \tag{3.3}$$

where each row of $\mathbf{H}$ is a vector in GF$(q)^N$ and the $(N-K)\times K$ matrix $\mathbf{H}$ is known as the *parity check matrix*. Note if $\mathbf{h}_1$ and $\mathbf{h}_2$ are orthogonal to all codewords, then any linear combination is also orthogonal to all codewords so $\mathbf{H}$ forms a basis for this subspace, $\mathcal{C}^\perp$, which is known as the *dual code*. The *dimension* of a code is defined as the number of basis vectors it has, and the dimension of the code and it's dual code sum to N.

Dual Code

Consider a non-systematic encoder matrix, $\mathbf{G}$ and form the matrix $\mathbf{M}$ by picking $N-K$

From Encoder to Parity Check Matrix

linear independent rows that are not codewords to form $\tilde{\mathbf{G}}$. Define:

$$\mathbf{M} = \begin{bmatrix} \mathbf{G} \\ \tilde{\mathbf{G}} \end{bmatrix} \qquad \mathbf{M}^{-1} = \begin{bmatrix} \tilde{\mathbf{H}}^T & \mathbf{H}^T \end{bmatrix} \tag{3.4}$$

$\mathbf{M}$ must have an inverse since it is an $N \times N$ matrix with linearly independent rows. $\mathbf{H}$ is obtained by taking the last $N-K$ columns of $\mathbf{M}^{-1}$ and transposing them. Therefore:

$$\mathbf{M}\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{G}\tilde{\mathbf{H}}^T & \mathbf{G}\mathbf{H}^T \\ \tilde{\mathbf{G}}\tilde{\mathbf{H}}^T & \tilde{\mathbf{G}}\mathbf{H}^T \end{bmatrix} \tag{3.5}$$

meaning that $\mathbf{G}\mathbf{H}^T = \mathbf{0}_{K\times(N-K)}$, so $\mathbf{H}$ is the parity check matrix since $\mathbf{x}\mathbf{H}^T = 0 \ \forall \mathbf{x} \Rightarrow \mathbf{u}\mathbf{G}\mathbf{H}^T = 0 \ \forall \mathbf{u} \Rightarrow \mathbf{G}\mathbf{H}^T = 0$. This provides a process for converting an encoder matrix into a parity check matrix, and can be inverted to go from $\mathbf{H}$ to $\mathbf{G}$.

Systematic Encoder to Parity Check

For a systematic encoder, the parity check matrix can be found by applying the above process to yield:

$$\mathbf{H}_s = \begin{bmatrix} -\mathbf{P}^T & \mathbf{I}_{N-K} \end{bmatrix} \tag{3.6}$$

this is known as *systematic*, though it's only property is allowing for easy construction of the systematic encoder.

Theorem 3.1.1 *For any of the N positions in an (N,K) code, there are an equal number of codewords which have any of the q symbols in this position.*

Proof

Consider the j-th position of the codeword, and take any non-zero element in the j-th column of the parity check matrix. By varying the corresponding information symbol through its possible q values, the j-th position of the crossword cycles through all of its values as every element is invertible in multiplication and addition. Thus when a linear encoder is applied to uniformly distributed data symbols, the probability of any code symbol will also be uniform. Therefore, linear codes are only useful for channels with uniform capacity achieving distributions.

Hamming Distance

Definition 3.1.2 (Hamming Distance) *The Hamming distance is the number of positions in which two word differ.*

When error correction/recovery is targeted for a symmetric channel, decoding rule chooses the codeword with minimum Hamming distance to the received codeword, following the *Maximum Likelihood* probabilistic rule. Note that the Hamming distance satisfies the triangle inequality:

$$d(\mathbf{a},\mathbf{b}) + d(\mathbf{b},\mathbf{c}) \geq d(\mathbf{a},\mathbf{b}) \tag{3.7}$$

as the differences may be overlapping i.e. if $a_i \neq b_i$ and $b_i \neq c_i$, $a_i = c_i$ is not precluded. Note that for a received word, $\mathbf{r}$, and codeword $\mathbf{x}$ such that $d(\mathbf{x},\mathbf{r}) < d_{\min}/2$, the received word will be uniquely decoded to $\mathbf{x}$ as:

Proof

$$d(\mathbf{x}',\mathbf{r}) \geq d(\mathbf{x}',\mathbf{x}) - d(\mathbf{r},\mathbf{x}) > \frac{d_{\min}}{2} \tag{3.8}$$

For an Erasure Channel

Every codeword has a hamming sphere of words that can be uniquely decoded to it. Note that any number of errors smaller than $d_{\min}/2$ is guaranteed to be decoded correctly. For an erasure channel, all codewords are different in *at least* $d_{\min}$ positions. Therefore, the code can recover from at least $d_{\min} - 1$ erasures (consider comparing a target codeword with every other possible codeword).

Theorem 3.1.2 *A code with minimum distance d_{min} can correct at least $\lfloor \frac{d_{min}}{2} \rfloor$ errors and recover from at least $d_{min} - 1$ errors.*

Hamming Weight

Definition 3.1.3 (Hamming Weight) *The Hamming Weight of a codeword is defined as the Hamming distance between the codeword and the all-zero codeword:*

$$w(\mathbf{x}) = d(\mathbf{0}_N, \mathbf{x}) \tag{3.9}$$

Note that clearly $d(\mathbf{x},\mathbf{x}') = w(\mathbf{x}-\mathbf{x}')$ since the difference will be zero in all positions that the codewords differ.

Theorem 3.1.3 (Weight-Distance Equivalent of Linear Codes) *For any two codewords in a linear code at distance d from each other, there exists a codeword of weight d. Equivalently, listing the number of codewords at every distance from any particular codeword produces the same list no matter which codeword is picked. For a linear codeword, the minimum distance is the minimum weight of all non-zero codewords i.e.* $d_{min} = w_{min}$.

These properties follow from the fact that the difference of two codewords is a codeword. The last statement is particularly powerful as it enables us to examine the weight of every codeword rather than examining every pair of codewords.

The minimum weight is the minimum number of columns of $\mathbf{H}$ that are linearly dependent. Since $\mathbf{x}\mathbf{H}^T = 0$, this can be written as:

Proof: Weights and the Parity Check Matrix

$$x_1\mathbf{h}_1^T + \ldots + x_N\mathbf{h}_N^T = 0 \tag{3.10}$$

where $\mathbf{h}_i$ represents the i-th column of $\mathbf{H}$. This can be proved by contradiction; if it was possible to find $w < w_{\min}$ columns which were a linear combination (and thus could be combined to give 0), then a codeword with weight w would be a valid codeword, and then the minimum distance would not be $w_{\min}$.

Theorem 3.1.4 (Singleton Bound) *The minimum distance,* d_{min} *of an* (N,K) *linear code satisfies:*

$$d_{min} \leq N - K + 1 \tag{3.11}$$

Codes that satisfy this bound with equality are known as maximum distance separable (MDS) codes.

The parity check matrix is an $(N-K)\times N$ matrix, and thus has maximum rank $N-K$. The column and row rank of a matrix are the same, and thus $\mathbf{H}$ contains at most $N-K$ independent columns, meaning that at most $N-K+1$ columns are linearly dependent. The rank may in fact be less than $N-K$ and thus this provides an upper bound on the minimum distance of the code. It is not easy to design matrices that satisfy the Singleton bound with equality (or design a matrix with any selection of $N-K$ columns being linearly independent) other than the simple examples of a single parity check code or repetition code.

Why?

Consider erasures and a $K\times N$ encoder matrix. If m code symbols are erased, if the any subset of the remaining $N-m$ columns gives an invertible $K\times K$ matrix, we can recover from the erasure. Clearly, if $m > (N-K)$, we cannot form this matrix, and number of erasures that can be recovered from is less than or equal to $N-K$. The maximum number of erasures that can be decoded is greater than or equal to $d_{\min}-1$ and rearranging yields the singleton bound. Designed an MDS code is equivalent to ensuring that any K columns of the generator matrix yields an invertible matrix.

Alternative Explanation: Encoder Matrices

3.2 The MacWilliams Identity

Let A_k be the weight profile of an (N,K) linear code, $\mathcal{C}$. A_k counts the number of codewords of weight k in the code. This is often written as a polynomial:

Weight Polynomial

$$A(x) = \sum_{k=0}^{N} A_k x^k \tag{3.12}$$

Consider $X_1,\ldots,X_N \overset{iid}{\sim} \text{Bern}(p)$ and denote the probability that the resulting vector is a codeword as $P(E)$. The probability of any codeword with weight w is $p^w(1-p^w)$ and therefore:

P(E)

$$P(E) = \sum_{k=0}^{N} A_k p^k (1-p)^{N-k} = (1-p)^N A\left(\frac{p}{1-p}\right) \tag{3.13}$$

since there are A_k codewords of weight k. We now seek to express $P(E)$ by considering the dual code. Let $\mathbf{S} = \mathbf{x}\mathbf{H}^T$ be the syndrome given by multiplying the random vector with the parity check matrix. Codewords of the dual code are obtained by forming linear combinations of the parity check matrix:

$$\mathbf{c}_\perp = \lambda_1 \mathbf{h}_1 + \ldots + \lambda_{N-K}\mathbf{h}_{N-K} \tag{3.14}$$

The random vector must be orthogonal to every single possible codeword. Therefore:

$$\mathbf{x}\mathbf{c}_\perp^T = \sum_{k=1}^{N-K} \lambda_k \mathbf{x}\mathbf{h}_k^T = \begin{cases} 0 & \mathbf{S} = \mathbf{0} \\ \sum_{k=1}^{N-1} \lambda_k s_k & \text{otherwise} \end{cases} \tag{3.15}$$

Second Case: Key Argument

with $s_k = \mathbf{x}\mathbf{h}_k^T$. Note that $s_k \in \{0,1\}$ and $\lambda_k \in \{0,1\}$. Considering k corresponding to $s_k = 1$, each term is included in half of the linear combinations. A recursive argument where an additional bit is added at each step suggests that this term is 0 for half of the code-words of the dual code and 1 for the other half. Therefore, the probability that random vector is in the dual code (and thus is not a codeword) is:

$$\underbrace{1 - P(E)}_{\text{prob. in } \mathcal{C}^\perp} = \frac{1}{2^{N-K-1}} \sum_{\mathbf{c}_\perp \in \mathcal{C}^\perp} P(\mathbf{x}\mathbf{c}_\perp^\mathbf{T} \neq 0) \tag{3.16}$$

since there are 2^{N-K-1} codewords (half the codewords of the dual code) which give $\mathbf{x}\mathbf{c}_\perp^\mathbf{T} = 1$.

Individual Terms

For a codeword of weight w, the probability that the dot product with $\mathbf{x}$ is non-zero is the probability that an odd number of indices corresponding to the non-zero positions in $\mathbf{c}$ are ones in $\mathbf{x}$. It can be shown that:

$$P(\text{odd number out of } k \text{ Bernoulli RVs}) = \frac{1}{2} - \frac{1}{2}(1-2p)^k \tag{3.17}$$

$$\therefore\ p(\mathbf{x}\mathbf{c}_\perp^T \neq 0) = \frac{1}{2} - \frac{1}{2}(1-2p)^w \tag{3.18}$$

Therefore:

$$1 - P(E) = 1 - \frac{1}{2^{N-K}} B(1-2p) \tag{3.19}$$

where $B(\cdot)$ is the weight polynomial for the dual code. Rearranging gives the *MacWilliams Identity*.

MacWilliams Identity

Theorem 3.2.1 (MacWilliams Identity) *Let $A(x)$ be the weight polynomial for an (N,K) binary linear code and let $B(x)$ be the weight polynomial for its dual code. Then,*

$$A(x) = \frac{(1+x)^N}{2^{N-K}} B\left(\frac{1-x}{1+x}\right) \tag{3.20}$$

The MacWilliams identity can be used to investigate properties of codes, and how the distribution of the weights of the dual code (which are set by the parity check matrices) affect the weights of the original code.

3.3 Reed Solomon Coding

Properties of Good Code

A 'good code' ought to add redundancy but also ought to introduce strong dependencies between symbols, so that if symbols are missing (or wrong), the errors can be recovered from.

The Discrete Fourier Transform

The Discrete Fourier Transform can be generalised to fields. For a DFT of length N in the finite field GF(q), the complex exponent used in the DFT matrix is replaced with a field element, α, which has order N (and since the order of α divides $q-1$, N must divide $q-1$).

$$\mathbf{X} = \mathbf{x}\mathcal{F} \qquad \mathbf{x} = \frac{1}{N^*}\mathcal{F}^{-1} \tag{3.21}$$

$$\tag{3.22}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are known as the *DFT Matrix* and *Inverse DFT Matrix* respectively. There elements are:

DFT Matrix

$$[\mathcal{F}]_{ij} = \alpha^{ij} \qquad [\mathcal{F}^{-1}]_{ij} = \alpha^{-ij} \tag{3.23}$$

Note that the division in the inverse transform is not be N but rather by:

N^*

$$N^* = \sum_{k=0}^{N-1} 1 \tag{3.24}$$

if q is prime, this yields N but this it not true in an extension field. For example, in GF(8), the sum of $N = 7$ ones is 1.

The DFT has the following properties:

Properties of the DFT

- *Cyclic Convolution*

  Cyclic Convolution

  Try Showing This

  $$z_k = \sum_{n=0}^{N-1} x_n y_{R_N(k-n)} \Leftrightarrow Z_k = X_k Y_k \tag{3.25}$$

  i.e. cyclic convolution in the time domain is equivalent to point-wise multiplication in the frequency domain. Note that also cyclic convolution in the frequency domain is equivalent to point-wise multiplication in the time domain.

- *Frequency Shift*

  Frequency Shift

  $$\tilde{x}_k = x_k \alpha^{kn} \Leftrightarrow \tilde{X}_k = X_{R_N(k-n)} \tag{3.26}$$

- *Time Shift*

  Time Shift

  $$\tilde{X}_k = X_k \alpha^{kn} \Leftrightarrow \tilde{x}_k = x_{R_n(k-n)} \tag{3.27}$$

The shift properties follow from the cyclic convolution property since the transform of the vector $\begin{bmatrix} 1 & \alpha & \dots & \alpha^{N-1} \end{bmatrix}$ is a 1 in position k and zero elsewhere. **Action:** *Not sure this checks out to be honest...*

Prove the Shift Properties

A *recurrent relation* or *difference equation* takes the form:

Recurrence Relation

$$x_k = c_1 x_{k-1} + \dots + c_L x_{k-L} \tag{3.28}$$

Definition 3.3.1 (Linear Complexity) *The linear complexity of a sequence,* $\mathbf{x}$*, is written* $\mathcal{L}(\mathbf{x})$ *and is the length of the* shortest *recurrence relation that generates the sequence.*

Linear Complexity

Theorem 3.3.1 *For a sequence,* $\mathbf{x}$*, with linear complexity,* $\mathcal{L}(\mathbf{x}) = L$*, observing any* $2L$ *consecutive elements of* $\mathbf{x}$ *suffices to reconstruct the recurrence relation that generates the sequence.*

The first $L+1$ symbols provides the first simultaneous equation to solve, and each additional symbol yields an additional equation. Given the complexity of a sequence, it can be verified that there is no shorter recurrence relation that generates the sequence by trying to solve the simultaneous equations and finding there is no solution.

Why?

Recalling the definition of the $\mathcal{Z}$-Transform and it's useful time-shift property, a recurrence relation translates to:

$$\bar{x}(z) = c_1 z^{-1} \bar{x}(z) + \dots + c_L z^{-L} \bar{x}(z) + P(z) \tag{3.29}$$

where $P(z)$ captures the influence of initial terms (from the time shift property) of the recurrence relation, the largest negative power in $P(z)$ is $L-1$. Then

$$\bar{x}(z) = \frac{P(z)}{C(z)} \text{ with } C(z) = 1 - c_1 z^{-1} - \dots - c_L z^{-L} \tag{3.30}$$

Thus, the linear complexity is the smallest degree of polynomial $C(z)$ such that $\bar{x}(z)$ can be written as above where $P(z)$ is any polynomial of smaller degree. Note when working with DFTs, sequences have fixed length N and we typically work with the linear complexity of the periodic continuation of $\mathbf{x}$.

Linear Complexity: Z-Transform View

Theorem 3.3.2 (Blahut's Theorem) *Let* $\mathbf{x}$ *be a vector of length* N *and of Hamming weight* $w(\mathbf{x}) < N/2$ *over any field. The Hamming weight of* $\mathbf{x}$ *equals the linear complexity of its DFT:*

$$w(\mathbf{x}) = \mathcal{L}(\mathbf{x}) \tag{3.31}$$

Intuition

The linear complexity measures the strength of dependences in a sequence and the Hamming weight measures its sparsity. Intuitively, the sparser the sequence, the simpler the linear dependencies in its DFT. Note that this theorem also holds in reverse, as the DFT and inverse DFT are very similar operations. If we wish to correct up to t errors, the error sequence has a DFT with linear complexity t.

Relevance

Proof

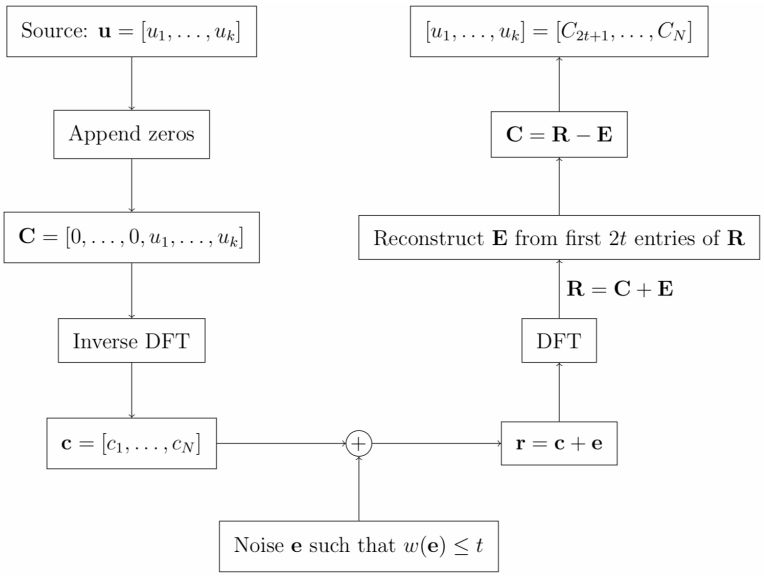
Consider the $\mathcal{Z}$-transform of the periodic repetition of $\mathbf{X}$, $\tilde{\mathbf{X}}$ where the capital denotes the DFT. It can be shown that:

$$\mathcal{Z}[\tilde{\mathbf{X}}] = \sum_{n=0}^{N-1} \frac{x_n}{1-\alpha^n z^{-1}} \tag{3.32}$$

meaning that we have expressed the sequence in the form $P(x)/C(x)$ where $P(z)$ has degree at most $w(\mathbf{x})-1$ and $C(z)$ has degree $w(\mathbf{x})$, showing the linear complexity of the DFT is $w(\mathbf{x})$.

Reed Solomon Schematic

A t error correcting Reed-Solomon code is a set of words in GF$(q)^N$ whose DFT is zero in the first $2t$ positions and N is a length for which the DFT exists in GF(q).



Note that Blahut's Theorem is essential; the first $2t$ entries of $\mathbf{R}$ correspond to the error vector only, and since the weight of $\mathbf{e} \leq t$, the linear complexity of $\mathbf{E} \leq t$.

The Parity Check Matrix

$\mathbf{H}$ consists of the first $2t = N - K$ rows (or columns, since symmetric) of the DFT matrix, since any $\mathbf{c}$ must have zeros in the first $2t$ positions once decoded.

The Encoder Matrix

The encoder matrix, $\mathbf{G}$, is the last $N - 2t$ rows of the inverse DFT matrix (the first $2t$ rows are not used due to the zero padding. A systematic encoder can be created by performing elementary row operations on $\mathbf{G}$ to bring it into systematic form. The above scheme is then modified by removing the inverse DFT on the transmitted and instead applying the inverse DFT to $\mathbf{C}$, then outputting the systematic part only. The first $2t$ entries of $\mathbf{R}$ correspond to $\mathbf{e}$ only still as the elementary row operations change the terms in $\mathbf{C}$ corresponding only to the u.

MDS

A Reed-Solomon code can correct t errors when $N-K = 2t$. Therefore, the minimum distance must be greater than $2t+1$, **which fufils the Singleton bound with equality**.

Chapter 4

Introduction to Cryptology

4.1 Introduction

Cryptanalysis aims to find out the secrets which have been hidden using cryptographic methods. Cryptanalysis can attempt to do this in a number of ways:

Cryptanalysis Attacks

1. Ciphertext only attack: the cryptanalyst has access to the ciphertext only.
2. Known plaintext attack: the cryptanalyst has access to a plaintext ciphertext pair.
3. Chosen plaintext attack: the cryptanalyst can *choose* the plaintext to be encrypted. This is common in modern attacks where the enemy cryptanalysis use impersonation.
4. Chosen ciphertext attack: the cryptanalyst can cause chosen encrypted messages to be decrypted.

Cryptography aims to ensure both *secrecy* (ensuring that only the intended recipient can read the message) and *authenticity* (ensuring the reader is confident about the message author) between communication parties. These aims are dual, and protocols exist that satisfy one but not the other. Often, two processes may be required to ensure that both aims are met. Cryptosystems are classified according to a number of criteria.

Aims of Cryptography

Secret key cryptosystems rely on the existence between a shared secret key known only to origin and destination whilst *public key cryptosystems* do not assume the existence of this key, and aim to establish a secret channel in full view of the enemy cryptanalyst. These techniques rely on the availability of authenticity to guarantee secrecy.

Cryptosystems: Key Types

Unconditional security concerns itself with methods for which there are mathematical proofs of safety (and thus the enemy cryptanalyst could be unlimited computational resources). *Computational security* concerns itself with algorithms that ensure security against **adversaries with limited computational resources, using all currently known and published mathematical knowledge**. Most practical methods are computationally secure, though there has been an increasing research focus on unconditional security.

Security

Principle 4.1.1 (Kerchkoff's Principle) *The cipher should be designed to be secure when the enemy cryptanalyst knows all details of the enciphering process and deciphering process* ***except the secret key****.*

Kerchoff's Principle

4.2 Public Key Cryptography

Public key cryptography relies on the concept of *one-way functions* which can be computed in one direction easily but whose inverse is very hard to compute or *trapdoor one-way functions* which are similar, but whose inverse can be computed easily if one knows the secret 'trapdoor', typically a secret number.

Trapdoor One-Way Functions

4.2.1 The Diffie Hellman Key Distribution System

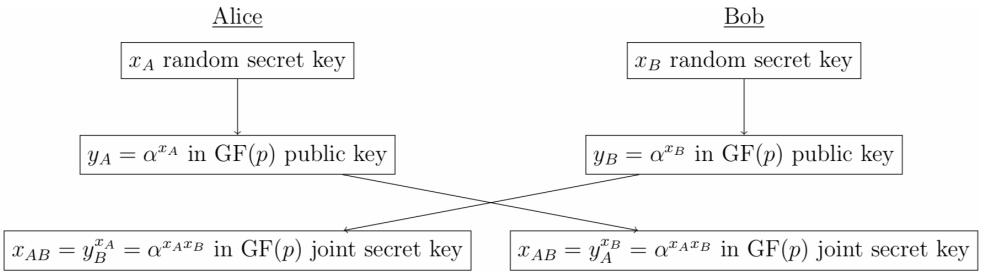
Let α be the generator for a cyclic group. It is difficult to invert discrete exponentiation, α^x, for an unknown x for groups whose order has a large prime factor e.g. GF(p) where

One-Way Function

$p-1$ has a large prime factor. The inversion is known as the *discrete logarithm* which satisfies the usual properties of logarithms:

$$\log_\alpha(n_1 \odot n_2) = \log_\alpha n_1 \oplus \log_\alpha n_2 \tag{4.1}$$

$$\log_\alpha(n_1^{n_2}) = n_2 \odot \log_\alpha n_1 \tag{4.2}$$



Components of the Diffie Hellman System

The system components are:

(a): Modulus p and generator α: published and define the system.

Caveat

(b): Secret keys x_A and x_B. Chosen using a strong random number generator. Note that if the random number generation is not strong (i.e. not truly random), this protocol is vulnerable to attacks.

Practical Consideration

(c): Public keys $y_A = \alpha^{x_A}$ and $y_B = \alpha^{x_B}$. Fully accessible to all parties. In practice, a trusted authentication authority is requires to publish keys in a manner that cannot be tampered with and vouches for their authenticity.

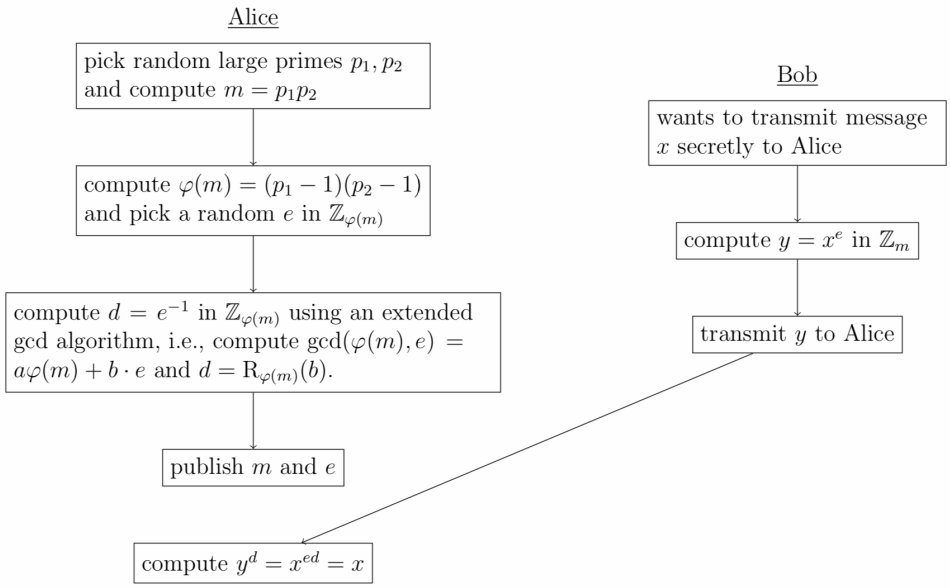
(d): Joint secret key $\alpha^{x_A x_B}$.

This technique allows Alice and Bob to agree on a secret key which is known only to them and relies on the difficulty of computing the discrete logarithm.

4.2.2 The Rivest-Shamir-Adelman Public-Key Cryptosystem

RSA Schematic

RSA is an encryption protocol which relies on the difficulty to factoring a large number, m, into it's prime factors.



Note that when operating in $\mathbb{Z}_m$, operations in the exponent occur modulo $\varphi(m)$ i.e.:

$$x^{e_1} \odot_m x^{e_2} = x^{e_1 \oplus_{\varphi(m)} e_2} \tag{4.3}$$

$$(x^{e_1})^{e_2} = x^{e_1 \odot_{\varphi(m)} e_2} \tag{4.4}$$

Only somebody who knows $\varphi(m)$ (which can be obtained by the secret prime) can find $d = e^{-1}$ in $\mathbb{Z}_{\varphi(m)}$ and hence invert the operation x^e in $\mathbb{Z}^m$. The protocol enables Alice to publish a public key which allows anyone to transmit messages that only Alice can read. In reality, it is essential that the public key is published in a manner that it cannot be tampered with.The components of the system are:

Real World Consideration
System Components

(a): Secret keys p_1, p_2 which are the initial secrets which must be chosen using reliable, unpredictable generators of large random primes; predictability can reduce the search space in an attack.

(b): The published key, $m = p_1 p_2$ and e. e must be invertible in $\mathbb{Z}_{\varphi(m)}$. It could be generated using the CRT if a factor of $\varphi(m)$ is available.

Generating e

(c): Alice's generated secrets are $\varphi(m)$ and d. d is computed using an extended GCD algorithm such as Stein's extended algorithm.

Note that RSA also relies on the difficulty of computing the discrete logarithm; in a known plaintext attack, the adversary could compute $d = \log_y x$.

RSA Works for Any x

In this section, we show that RSA decryption works for any value of x. The modulo used is:

$$m = p_1 p_2 \qquad \varphi(m) = (1-p_1)(1-p_2) \tag{4.5}$$

We choose e, d such that $e \odot_{\varphi(m)} d = 1$. Therefore:

$$ed - k\varphi(m) = ed - k(p_1-1)(p_2-1) = 1 \tag{4.6}$$

Therefore:

$$e \odot_{p_1-1} d = 1 \text{ and } e \odot_{p_2-1} d = 1 \tag{4.7}$$

A message, $x \in \mathbb{Z}_m$, is chosen and encrypted using $y = R_m(x^e)$. The decoding rule is $\tilde{x} = R_m(y^d)$.

$$\tilde{x} = R_m(y^d) = R_m(R_m(x^e)^d) = R_m(x^{ed}) \tag{4.8}$$

Note that $e \odot_x d = 1 \Leftrightarrow ed - kx = 1$. Now, consider:

$$\begin{aligned} x^{ed} &= x^{1+k(p_1-1)} \\ &= x(x^{p_1-1})^k \\ &= x(x^{p_2-1})^k \end{aligned} \tag{4.9}$$

Fermat's Theorem implies that:

$$\begin{aligned} R_{p_1}(x^{ed} - x) &= R_{p_1}(R_{p_1}(x) \cdot \underbrace{R_{p_1}(x^{p_1-1})^k}_{1} - R_{p_1}(x)) \\ &= R_{p_1}(x - x) \\ &= 0 \end{aligned} \tag{4.10}$$

Similarily, $R_{p_2}(x^{ed} - x) = 0$. Therefore,

$$x^{ed} - x = kp_1p_2 = km \tag{4.11}$$

Therefore:

$$R_m(\tilde{x} - x) = 0 \tag{4.12}$$