Problem 1. (Min-entropy and Statistical Difference)

(a). Prove that for every two random variables $X$ and $Y$,
\[
\Delta(X, Y) = \max_f |E[f(X)] - E[f(Y)]| = \frac{1}{2} \cdot |X - Y|_1,
\]
where the maximum is over all $[0,1]$-valued functions $f$. (Hint: first identify the functions $f$ that maximize $|E[f(X)] - E[f(Y)]|$.)

(b). Suppose that $(W, X)$ are jointly distributed random variables where $W$ takes values in $\{0,1\}^\ell$ and $(W, X)$ is a $k$-source. Show that for every $\varepsilon > 0$, with probability at least $1 - \varepsilon$ over $w \overset{R}{\leftarrow} W$, we have $X|w = w$ is a $(k - \ell - \log(1/\varepsilon))$-source.

(c). Suppose that $X$ is an $(n - \Delta)$-source taking values in $\{0,1\}^n$, and we let $X_1$ consist of the first $n_1$ bits of $X$ and $X_2$ the remaining $n_2 = n - n_1$ bits. Show that for every $\varepsilon > 0$, $(X_1, X_2)$ is $\varepsilon$-close to some $(n_1 - \Delta, n_2 - \Delta - \log(1/\varepsilon))$ block source.

Problem 2. (Extractors vs. Samplers) One of the problems we have revisited several times is that of randomness-efficient sampling: Given oracle access to a function $f : \{0,1\}^m \rightarrow [0,1]$, approximate its average value $\mu(f)$ to within some small additive error. All of the samplers we have seen work as follows: they choose some $n$ random bits, use these to decide on some $D$ samples $z_1, \ldots, z_D \in \{0,1\}^m$, and output the average of $f(z_1), \ldots, f(z_D)$. We call such a procedure a $(\delta, \varepsilon)$-(averaging) sampler if, for any function $f$, the probability that the sampler’s output differs from $\mu(f)$ by more than $\varepsilon$ is at most $\delta$. In this problem, we will see that averaging samplers are essentially equivalent to extractors.

Given $\text{Ext}: \{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m$, we obtain a sampler $\text{Smp}$ which chooses $x \overset{R}{\leftarrow} \{0,1\}^n$, and uses $\{\text{Ext}(x, y) : y \in \{0,1\}^d\}$ as its $D = 2^d$ samples. Conversely, every sampler $\text{Smp}$ using $n$ random bits to produce $D = 2^d$ samples in $\{0,1\}^m$ defines a function $\text{Ext}: \{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m$. 


(a). Prove that if Ext is a \((k - 1, \varepsilon)\)-extractor, then Smp is a \((2^k / 2^n, \varepsilon)\)-averaging sampler.

(b). Prove that if Smp is a \((2^k / 2^n, \varepsilon)\)-sampler, then Ext is a \((k + \log(1/\varepsilon), 2\varepsilon)\)-extractor.

(c). Suppose we are given a constant-error BPP algorithm which uses \(r = r(n)\) random bits on inputs of length \(n\). Show how, using Part (a) and the extractor of Theorem 8 from Lecture Notes 12, we can reduce its error probability to \(2^{-\ell}\) using \(O(r) + \ell\) random bits, for any polynomial \(\ell = \ell(n)\). (Note that this improves the \(r + O(\ell)\) given by expander walks for \(\ell \gg r\) .) Conclude that every problem in BPP has a randomized algorithm which only errs for \(2^{\log^{0.01}}\) choices of its \(q\) random bits!

Problem 3. (Encryption and Deterministic Extraction) A (one-time) encryption scheme with key length \(n\) and message length \(m\) consists of an encryption function \(\text{Enc}: \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^\ell\) and a decryption function \(\text{Dec}: \{0,1\}^\ell \times \{0,1\}^m \rightarrow \{0,1\}^m\) such that \(\text{Dec}(k, \text{Enc}(k, u)) = u\) for every \(k \in \{0,1\}^n\) and \(u \in \{0,1\}^m\). Let \(K\) be a random variable taking values in \(\{0,1\}^n\). We say that \((\text{Enc}, \text{Dec})\) is \((\text{statistically}) \varepsilon\)-secure with respect to \(K\) if for every two messages \(u, v \in \{0,1\}^m\), we have \(\Delta(\text{Enc}(K, u), \text{Enc}(K, v)) \leq \varepsilon\). For example, the one-time pad, where \(n = m = \ell\) and \(\text{Enc}(k, u) = k \oplus u = \text{Dec}(k, u)\) is \(0\)-secure (aka perfectly secure) with respect to the uniform distribution \(K = U_m\). For a class \(\mathcal{C}\) of sources on \(\{0,1\}^n\), we say that the encryption scheme \((\text{Enc}, \text{Dec})\) is \(\varepsilon\)-secure with respect to \(\mathcal{C}\) if \(\text{Enc}\) is \(\varepsilon\)-secure with respect to every \(K \in \mathcal{C}\).

(a). Show that if there exists a deterministic \(\varepsilon\)-extractor \(\text{Ext}: \{0,1\}^n \rightarrow \{0,1\}^m\) for \(\mathcal{C}\), then there exists an \(2\varepsilon\)-secure encryption scheme with respect to \(\mathcal{C}\).

(b). Conversely, use the following steps to show that if there exists an \(\varepsilon\)-secure encryption scheme \((\text{Enc}, \text{Dec})\) with respect to \(\mathcal{C}\), where \(\text{Enc}: \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^\ell\), then there exists a deterministic \(2\varepsilon\)-extractor \(\text{Ext}: \{0,1\}^n \rightarrow \{0,1\}^{m - 2\log(1/\varepsilon) - O(1)}\) for \(\mathcal{C}\), provided \(m \geq \log n + 2\log(1/\varepsilon) + O(1)\).

(i) For each fixed key \(k \in \{0,1\}^n\), define a source \(X_k\) on \(\{0,1\}^\ell\) by \(X_k = \text{Enc}(k, U_m)\), and let \(\mathcal{C}'\) be the class of all these sources (i.e., \(\mathcal{C}' = \{X_k : k \in \{0,1\}^n\}\)). Show that there exists a deterministic \(\varepsilon\)-extractor \(\text{Ext}' : \{0,1\}^\ell \rightarrow \{0,1\}^{m - 2\log(1/\varepsilon) - O(1)}\) for \(\mathcal{C}'\), provided \(m \geq \log n + 2\log(1/\varepsilon) + O(1)\).

(ii) Show that if \(\text{Ext}'\) is a deterministic \(\varepsilon\)-extractor for \(\mathcal{C}'\) and \(\text{Enc}\) is \(\varepsilon\)-secure with respect to \(\mathcal{C}\), then \(\text{Ext}(k) = \text{Ext}'(\text{Enc}(k, 0^m))\) is a deterministic \(2\varepsilon\)-extractor for \(\mathcal{C}\).

Thus, a class of sources can be used for secure encryption iff it is deterministically extractable.

Problem 4. (The Building-Block Extractor) Assume the condenser stated in Theorem 7 from Lecture Notes 12. Show that for every constant \(t > 0\) and all positive integers \(n \geq k\) and all \(\varepsilon > 0\), there is an explicit \((k, \varepsilon)\)-extractor \(\text{Ext}: \{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m\) with \(m = k/2\) and \(d = k/t + O(\log(n/\varepsilon))\). (Hint: convert the source into a block source with blocks of length \(k/O(t) + O(\log(n/\varepsilon))\).)