5370 Midterm

Ritchie Cai

November 5, 2015

Instructions.

- The exam is due before class on October 15, 2015.
- Your solution is to be written in LaTeX. The LaTeXfile and the corresponding pdf file is to be emailed to `Robert_Marks@Baylor.edu` by the deadline.
- Show your work. Clarity of presentation counts.
- No human resource will be consulted in the execution of the exam.
- Problem is really difficult. A successful solution might result in a publication. Be a Huffman!

1. A random variable X has the same probability of one third at the points $x = -1, 0, 1$. Its entropy is thus $H_X = \log_2 3$. Let $g(\cdot)$ denote a continuous function and define the random variable $Y = g(X)$.

1. Find a continuous function, $g(\cdot)$, that reduces the entropy of Y to zero.
   Solution:
   $g = \sin(\pi x)$

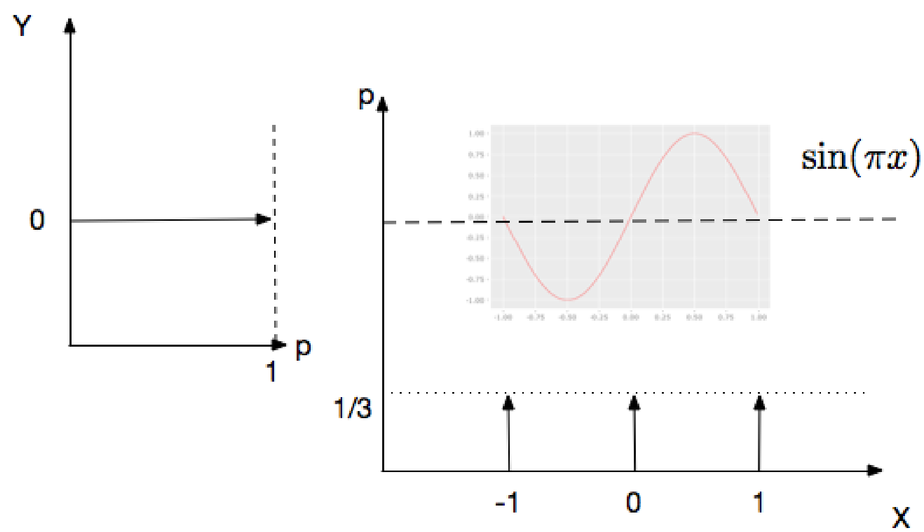

Figure 1: Figure for problem1a

2. Find another continuous function, $g(\cdot)$, that reduces the entropy of Y below the entropy of X, but not to zero.
   Solution:
   $g = |x|$

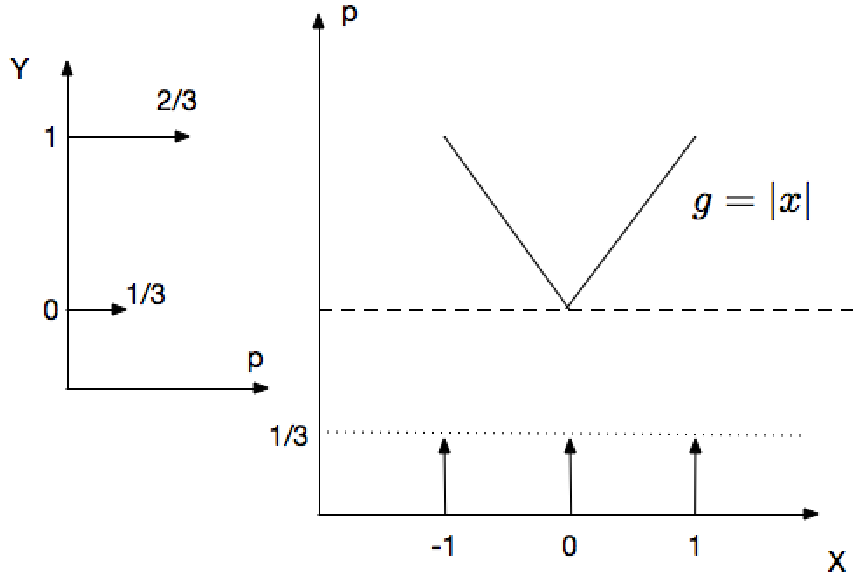

Figure 2: Figure for problem1b

3. Find a third function , $g(\cdot)$, that generates an entropy of Y equal to the entropy of X.
   Solution:
   $g = x + 1$

For all three answers, please sketch the nonlinearity and label your axes.

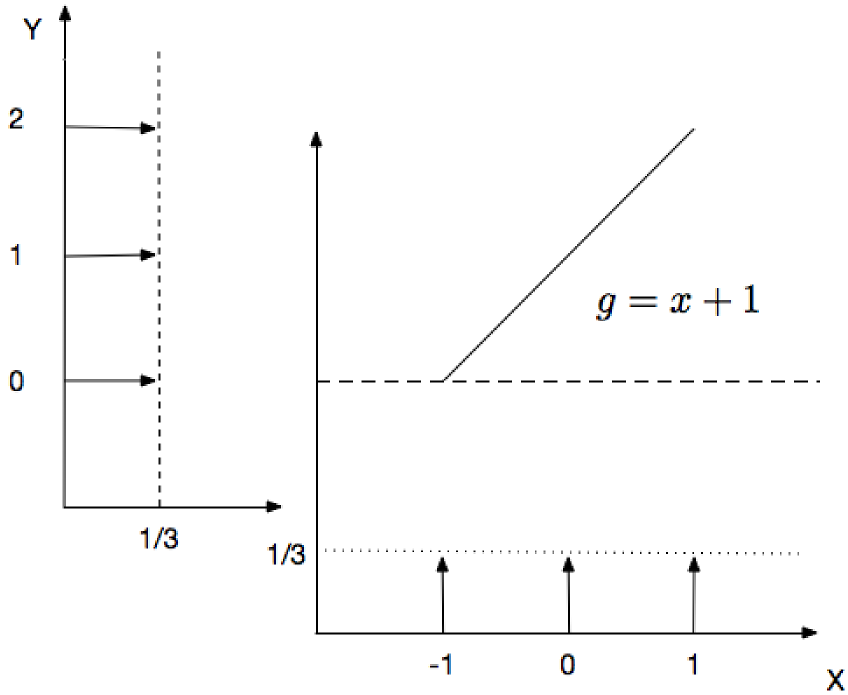

Figure 3: Figure for problem1c

2. Random variable are defined by the conditional probabilities

- $\Pr[Y = 1|X = 0] = q$
- $\Pr[Y = 0|X = 1] = p$

where both p and q are fixed probabilities. Assume $\Pr[X = 0] = \pi$ where π is some probability.

1. Calculate the mutual information between X and Y.
   Solutions:

Assuming both Y and X are binary

$$
\begin{aligned}
\Pr[Y=0|X=0] &= 1-q \\
\Pr[Y=1|X=1] &= 1-p \\
\Pr[Y=0] &= \Pr[Y=0, X=0] + \Pr[Y=0, X=1] \\
&= \Pr[Y=0|X=0]\Pr[X=0] + \Pr[Y=0|X=1]\Pr[X=1] \\
&= (1-q)\pi + p(1-\pi) \\
&= p + \pi(1-q-p) \\
\Pr[Y=1] &= \Pr[Y=1, X=0] + \Pr[Y=1, X=1] \\
&= \Pr[Y=1|X=0]\Pr[X=0] + \Pr[Y=1|X=1]\Pr[X=1] \\
&= q\pi + (1-p)(1-\pi) \\
&= q\pi + 1 - \pi - p + p\pi \\
&= 1 - p + \pi(q+p-1)
\end{aligned}
$$

$$
\begin{aligned}
I(X;Y) &= H(Y) - H(Y|X) \\
H(Y) &= -\Pr[Y=0]\log\Pr[Y=0] - \Pr[Y=1]\log\Pr[Y=1] \\
&= -(p+\pi(1-q-p))\log(p+\pi(1-q-p)) \\
&\quad - (1-p+\pi(q+p-1))\log(1-p+\pi(q+p-1)) \\
H(Y|X) &= \sum_{x,y}\Pr[X=x, Y=y]\log\Pr[Y=y|X=x] \\
&= \sum_{x,y}\Pr[X=x]\Pr[Y=y|X=x]\log\Pr[Y=y|X=x] \\
&= \Pr[X=0]\Pr[Y=0|X=0]\log\Pr[Y=0|X=0] \\
&\quad + \Pr[X=0]\Pr[Y=1|X=0]\log\Pr[Y=1|X=0] \\
&\quad + \Pr[X=1]\Pr[Y=0|X=1]\log\Pr[Y=0|X=1] \\
&\quad + \Pr[X=1]\Pr[Y=1|X=1]\log\Pr[Y=1|X=1] \\
&= \pi(1-q)\log(1-q) + \pi q\log q \\
&\quad + (1-\pi)p\log p + (1-\pi)(1-p)\log(1-p) \\
I(X;Y) &= -(p+\pi(1-q-p))\log(p+\pi(1-q-p)) \\
&\quad - (1-p+\pi(q+p-1))\log(1-p+\pi(q+p-1)) \\
&\quad - \pi(1-q)\log(1-q) - \pi q\log q \\
&\quad - (1-\pi)p\log p - (1-\pi)(1-p)\log(1-p)
\end{aligned}
$$

2. What value of π maximizes the mutual information?
Solution:

$$
\begin{aligned}
\frac{d}{d\pi}I(X;Y) &= -(1-q-p)\log(p+\pi(1-q-p)) - 1 + q + p \\
&\quad - (q+p-1)\log(1-p+\pi(q+p-1)) - q - p + 1 \\
&\quad - (1-q)\log(1-q) - q\log q + p\log p + (1-p)\log(1-p) \\
&= -(1-q-p)\log(p+\pi(1-q-p)) \\
&\quad - (q+p-1)\log(1-p+\pi(q+p-1)) \\
&\quad - (1-q)\log(1-q) - q\log q + p\log p + (1-p)\log(1-p) = 0
\end{aligned}
$$

3. The average word in English is $\lambda = 5.1$ letters.

1. If letters are drawn randomly from an $N = 27$ character alphabet (A through Z and a space),[1] then when is a thick novel with W words typical? Explain your reasoning.

   Solution:
   When
   $$p\left(X^{\lceil W*(5.1+1)\rceil}\right) \approx 2^{-W*(5.1+1)*(5.1+1)} \tag{1}$$
   Because when n is very large, $E\left[\frac{1}{n}l(X^n)\right] \approx H(X)$, which means $H(X) \approx \lambda$. The number of alphabets here, including space, should be about $\lceil W*(\lambda+1)\rceil$, where "+1" is due the space character. Since we are adding space at the end of each words, the entropy for words with space should be $\lambda + 1$. So we have (1), satisfying which makes the sequence at least weakly typical.

2. Here is a web page of 40,000 sampled words

   `http://www.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html`

   (a) Complete the **Count** column in table by including the *space*. Call the resulting empirical distribution $q(x)$ where $x = 0, 1, 2, \cdots$ corresponds to (space, e, t, a, o, $\cdots$).
   Solution:
   Since sampled words are 40000, there should be the same amount of spaces, one space per word.

Index	Letters	Count	Probability
0	Space	40000	0.1799
1	E	21912	0.09857
2	T	16587	0.07461
3	A	14810	0.06662
4	O	14003	0.06300
5	I	13318	0.05991
6	N	12666	0.05698
7	S	11450	0.05151
8	R	10977	0.04938
9	H	10795	0.04856
10	D	7874	0.03542
11	L	7253	0.03263
12	U	5246	0.02360
13	C	4943	0.02224
14	M	4761	0.02142
15	F	4200	0.01889
16	Y	3853	0.01733
17	W	3819	0.01718
18	G	3693	0.01661
19	P	3316	0.01492
20	B	2715	0.01221
21	V	2019	0.009082
22	K	1257	0.005654
23	X	315	0.001417
24	Q	205	0.0009222
25	J	188	0.0008457
26	Z	128	0.0005758

[1] We are not considering punctuation, capitalization or numbers.

After experiment, I found that beta-binomial distribution with $n = 26, \alpha = 0.68, \beta = 2.7$ fit the data the best as shown in fig 4. Exponential distribution also fits well, but not as well as beta-binomial distribution, also exponential distribution is continuous, and beta-binomial is discrete which also make beta-binomial distribution a better fit.

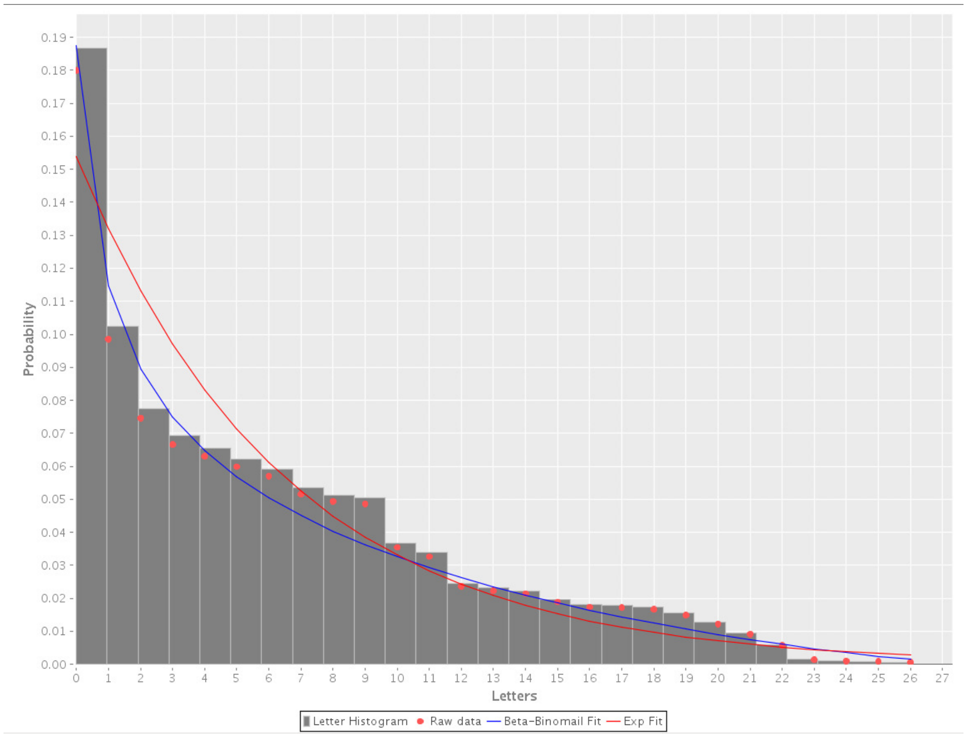

Figure 4: Beta-binomial pmf fitting with $n = 26, \alpha = 0.68, \beta = 2.7$

(b) Find the value of α that minimizes the Kullback-Liebler distance between the distribution corresponding to this augmented table and

$$p(x) = (1 - \alpha)\alpha^x \tag{2}$$

Solutions:
Beta-binomial distribution pmf:

$$q(x|n, \alpha, \beta) = \binom{n}{x} \frac{B(x + \alpha, n - x + \beta)}{B(\alpha, \beta)}$$
$$p(x) = (1 - \alpha)\alpha^x$$

Since we are assuming $q(x)$ as our true distribution and trying to make $p(x)$ as close to $q(x)$ as possible, by reducing KLIC, thus Kullback-Liebler distance should be expressed as:

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

Let

$$C_x = \binom{n}{x}$$
$$B_x = B(x + \alpha, n - x + \beta)$$
$$B = B(\alpha, \beta)$$

then we have:

$$D(q||p) = \sum_x \binom{n}{x} \frac{B(x+\alpha, n-x+\beta)}{B(\alpha,\beta)} \log \left(\frac{\binom{n}{x} \dfrac{B(x+\alpha, n-x+\beta)}{B(\alpha,\beta)}}{(1-\alpha)\alpha^x} \right)$$

$$= \frac{1}{B} \sum_x C_x B_x \log \left(\frac{\frac{C_x B_x}{B}}{(1-\alpha)\alpha^x} \right)$$

$$= \frac{1}{B} \sum_x \left[C_x B_x \log \left(\frac{C_x B_x}{B} \right) - C_x B_x \log((1-\alpha)\alpha^x) \right]$$

$$= \frac{1}{B} \sum_x C_x B_x \log \left(\frac{C_x B_x}{B} \right) - \frac{1}{B} \sum_x C_x B_x \log((1-\alpha)\alpha^x)$$

As we can see, the function of $D(q||p)$ contains a inverted log function with respect to α. Since log is a concave function, if there is a critical point exist, it must be a global minima. To find the critical point, we need to set $\frac{d}{d\alpha} D(q||p) = 0$

$$\frac{d}{d\alpha} D(q||p) = \frac{d}{d\alpha} \left(\frac{1}{B} \sum_x C_x B_x \log \left(\frac{C_x B_x}{B} \right) - \frac{1}{B} \sum_x C_x B_x \log((1-\alpha)\alpha^x) \right) = 0$$

$$= -\frac{1}{B} \sum_x \frac{C_x B_x (x\alpha^{(x-1)} - (x+1)\alpha^x)}{(1-\alpha)\alpha^x} = 0$$

$$= -\frac{1}{B} \sum_x \frac{C_x B_x (x\alpha^{-1} - x - 1)}{(1-\alpha)} = 0$$

$$= \sum_x C_x B_x (x\alpha^{-1} - x - 1) = 0$$

$$= \sum_x C_x B_x x - \sum_x C_x B_x \alpha x - \sum_x C_x B_x \alpha = 0$$

$$\alpha = \frac{\sum_{x=0}^{26} C_x B_x x}{\sum_{x=0}^{26} C_x B_x (x+1)} = \frac{\sum_{x=0}^{26} \binom{n}{x} B(x+\alpha, n-x+\beta) x}{\sum_{x=0}^{26} \binom{n}{x} B(x+\alpha, n-x+\beta)(x+1)}$$

with $n = 26, \alpha = 0.68, \beta = 2.7$

$$\alpha \approx 0.8395$$

(c) Compare this to the α that minimizes

$$\sum_{x=0}^{\infty} |p(x) - q(x)|^2$$

Solution:

Since we only have $0 - 26$, i.e., 26 alphabets with one space character, the summation would only go up to 26, reset of terms would be all 0

$$
\begin{aligned}
\sum_{x=0}^{26} |p(x) - q(x)|^2 &= \sum_{x=0}^{26} \left| (1-\alpha)\alpha^x - \frac{C_x B_x}{B} \right|^2 \\
&= \sum_{x=0}^{26} \left| a^x - a^{x+1} - \frac{C_x B_x}{B} \right|^2 \\
&= \sum_{x=0}^{26} \Bigg(a^{2x} - a^{2x+1} - \frac{C_x B_x}{B} a^x \\
&\qquad - a^{2x+1} + a^{2x+2} + \frac{C_x B_x}{B} a^{x+1} \\
&\qquad - \frac{C_x B_x}{B} a^x + \frac{C_x B_x}{B} a^{x+1} + \left(\frac{C_x B_x}{B}\right)^2 \Bigg) \\
&= \sum_{x=0}^{26} \left(a^{2x+2} - 2a^{2x+1} + a^{2x} + \frac{2C_x B_x}{B} a^{x+1} - \frac{2C_x B_x}{B} a^x + \left(\frac{C_x B_x}{B}\right)^2 \right) \qquad (3) \\
&= \sum_{x=0}^{26} \left(a^{2x+2} - 2a^{2x+1} + a^{2x} \right) + \sum_{x=0}^{26} \left(\frac{2C_x B_x}{B} a^{x+1} - \frac{2C_x B_x}{B} a^x + \left(\frac{C_x B_x}{B}\right)^2 \right) \qquad (4)
\end{aligned}
$$

Since this is a polynomial with even degree and positive leading coefficient, $a^{2\times 26+2} = a^{54}$, the opening of the polynomial should be upward and it should have at least one global minimum.

With first and second derivative of (3):

$$
\begin{aligned}
\frac{d}{da}\left(\sum_{x=0}^{26} |p(x) - q(x)|^2 \right) &= \sum_{x=1}^{26} \left((2x+2)a^{2x+1} - (4x+2)a^{2x} + 2xa^{2x-1} \right) \\
&\quad + \sum_{x=1}^{26} \left((x+1)\frac{2C_x B_x}{B} a^x - x\frac{2C_x B_x}{B} a^{x-1} \right) + 2a - 2 + \frac{2C_0 B_0}{B} \\
\frac{d^2}{da^2}\left(\sum_{x=0}^{26} |p(x) - q(x)|^2 \right) &= \sum_{x=2}^{26} \left((2x+2)(2x+1)a^{2x} - (2x+2)2xa^{2x-1} + 2x(2x-1)a^{2x-2} \right) \\
&\quad + \sum_{x=2}^{26} \left(x(x+1)2\frac{C_x B_x}{B} a^{x-1} - x(x-1)2\frac{C_x B_x}{B} a^{x-2} \right) \\
&\quad + 12a^2 - 12a + 2 + \frac{4C_1 B_1}{B} + 2
\end{aligned}
$$

we can numerically find the global minimum(s) with Newton's method[2] which gives me $\alpha \approx 0.83495$. Code used for newton's method is shown in listing (1) and (2)

Listing 1: Functions for using newton's method

```
(ns elc5370.optimization
  (:use clojure.core
        elc5370.binomial)
  (:import [org.apache.commons.math3.special Beta]))

(def beta-binomial (new-beta-binomial-fn 26 0.68 2.7))

(defn newton
  [f df & {:keys [x0 iter tol] :or {x0   (double 0.0)
                                    iter (long   100)
```

[2] https://en.wikipedia.org/wiki/Newton%27s_method

```
                                       tol   (double 0.000001)}}]
  (loop [i ^Long   (long 0)
         x ^Double (double x0)]
    (if (< i (long iter))
      (let [fx (f x)]
        (if (> (Math/abs fx) tol)
          (recur (unchecked-inc i) (- x (/ fx (df x))))
          x))
      x)))

(defn f
  [a]
  (loop [x ^Long   (long 0)
         s ^Double (double 0.0)]
    (if (<= x 26)
      (let [_2x   ^Long   (long (* x 2))
            CxBxB ^Double (beta-binomial x)]
        (recur (unchecked-inc x)
               (+    s
                     (Math/pow a (+ _2x 2))
                  (- (* 2 (Math/pow a (inc _2x))))
                     (Math/pow a _2x)
                     (* 2 CxBxB (Math/pow a (inc x)))
                  (- (* 2 CxBxB (Math/pow a x)))
                     (Math/pow CxBxB 2))))
      s)))

(defn df
  [a]
  (loop [x ^Long   (long 1)
         s ^Double (double 0.0)]
    (if (<= x 26)
      (let [_2x     ^Long   (long (* x 2))
            _2CxBxB ^Double (double (* 2 (beta-binomial x)))]
        (recur (unchecked-inc x)
               (+    s
                     (* (+ _2x 2) (Math/pow a (inc _2x)))
                  (- (* (+ (* 4 x) 2) (Math/pow a _2x)))
                     (* _2x (Math/pow a (unchecked-dec _2x)))
                     (* (inc x) _2CxBxB (Math/pow a x))
                  (- (* x _2CxBxB (Math/pow a (unchecked-dec x)))))))
      (+ s (* 2 a) (- 2) (* 2 (beta-binomial 0))))))

(defn ddf
  [a]
  (loop [x ^Long   (long 2)
         s ^Double (double 0.0)]
    (if (<= x 26)
      (let [_2x     ^Long   (long (* x 2))
            _2CxBxB ^Double (double (* 2 (beta-binomial x)))]
        (recur (unchecked-inc x)
               (+    s
                     (* (+ _2x 2) (+ _2x 1) (Math/pow a _2x))
                  (- (* (+ _2x 2) _2x       (Math/pow a (dec _2x))))
                     (* _2x (dec _2x)       (Math/pow a (- _2x 2)))
                     (* x (inc x) _2CxBxB (Math/pow a (dec x)))
                  (- (* x (dec x) _2CxBxB (Math/pow a (- x 2)))))))
      (+ s
         (* 12 (Math/pow a 2))
         (- (* 12 a))
         4
         (* 4 (beta-binomial 1))))))

(newton df ddf :x0 1 :iter 10000 :tol 0.000001)
```

Listing 2: Functions for beta binomial distribution

```
(ns elc5370.binomial
```

```
    (:use clojure.core)
    (:import [org.apache.commons.math3.special Beta]))

  (def comb-cache
    (let [cache (atom {})]
      (fn ! ([^BigInteger n ^BigInteger k]
            (or (@cache [n k])
                (cond
                  (or (= n k) (= k 0)) (bigint 1)
                  (or (= k 1) (= k (dec n))) (bigint n)
                  :else (let [v ^BigInteger (+ (! (unchecked-dec n) (unchecked-dec k))
                                                (! (unchecked-dec n) k))]
                          (swap! cache assoc [n k] v)
                          v))))
        ([] @cache))))

  (def comb-mem
    (memoize
     (fn [^BigInteger n ^BigInteger k]
       (cond
         (or (= n k) (= k 0)) (bigint 1)
         (or (= k 1) (= k (dec n))) (bigint n)
         (= [n k] [1 0]) (bigint 1)
         (= [n k] [1 1]) (bigint 1)
         :else (+ (comb-mem (- n 1) (- k 1))
                  (comb-mem (- n 1) k))))))

  (defn new-beta-fn
    [n alpha beta]
    (memoize (fn [^Long x]
               (if (< n x)
                 (throw (Exception. (format "Beta: Combination of range, x > n: x = %d,
                     n = %d" x n)))
                 (Math/exp (Beta/logBeta (+ x alpha) (+ (- n x) beta)))))))

  (def fact-cache
    (let [cache (atom {})]
      (fn ! ([n]
            (let [n ^BigInteger (bigint n)]
             (cond
               (< n 0) (throw (Exception. (format "Factorial only accept integer greater
                    than 0: n = %d" n)))
               (<= n 1) (bigint 1)
               :else (or (@cache n)
                         (let [v (* n (! (dec n)))]
                           (swap! cache assoc n v)
                           v)))))
        ([] @cache))))


  (defn new-beta-binomial-fn
    [n alpha beta]
    (let [cache   (atom {})
          beta-fn (new-beta-fn n alpha beta)
          B       (Math/exp (Beta/logBeta alpha beta))]
      (fn ([^Long x]
          (let [bb-fn (fn [x]
                        (cond
                          (< x 0) (throw
                                   (Exception.
                                    (format "Beta-binomial: Index must be non negative
                                        integer: x = %d" x)))
                          (< n x) (throw
                                   (Exception.
                                    (format "Beta-binomial: Combination values of range:
                                         n = %d, x = %d" n x)))
                          :else (or (@cache x)
                                    (let [val (* (comb-cache n x)
```

```
                                                    (/ (beta-fn x) B))]
                                        (swap! cache assoc x val)
                                        val))))]
           (if (coll? x) (map bb-fn x) (bb-fn x))))
       ([] @cache))))
```

(d) What is the average word length in English based on $q(x)$? On $p(x)$?
Solution:
Ideally,

$$\text{average word length} = \frac{\text{total length (or non-space characters count)}}{\text{number of words (or spaces)}} = \frac{182303}{40000} \approx 4.56$$

However, average length based on a distribution $f(x)$, in general, should be

$$\text{average word length} = \frac{\sum_{x=1}^{26} f(x)}{f(x=0)}$$

So, for $q(x)$, we have:

$$\text{average word length} = \frac{\sum_{x=1}^{26} q(x)}{q(0)} \approx \frac{0.8066}{0.1935} \approx 4.17$$

where

$$q(x) = \binom{n}{x} \frac{B(x+\alpha, n-x+\beta)}{B(\alpha, \beta)} \quad \text{with } n = 26, \alpha = 0.68, \beta = 2.7$$

For $p(x)$, we have:

$$\text{average word length} = \frac{\sum_{x=1}^{26} p(x)}{p(0)} \approx \frac{0.8306}{0.1605} \approx 5.18$$

where

$$p(x) = (1-\alpha)\alpha^x \quad \text{with } \alpha = 0.8395$$

4. Problem 2.10b in the text:

$$
\begin{aligned}
X &= \begin{cases} X_1 & \text{with probability } \alpha \\ X_2 & \text{with probability } 1-\alpha \end{cases} \\
H(X) &= -\alpha \log \alpha - (1-\alpha)\log(1-\alpha) + \alpha H(X_1) + (1-\alpha)H(X_2) \qquad (5)
\end{aligned}
$$

Maximize over α to show that $2^{H(X)} \leq 2^{H(X_1)} + 2^{H(X_2)}$ and interpret using the notion that $2^{H(X)}$ is the effective alphabet size.

Solution:

$$
\begin{aligned}
\frac{d}{d\alpha}H(X) &= \frac{d}{d\alpha}\left[-\alpha \log \alpha - (1-\alpha)\log(1-\alpha) + \alpha H(X_1) + (1-\alpha)H(X_2)\right] = 0 \\
&= -\left(\log \alpha + \alpha\frac{1}{\alpha}\right) - \left(-\log(1-\alpha) - (1-\alpha)\frac{1}{1-\alpha}\right) + H(X_1) - H(X_2) \\
&= -\log \alpha - 1 + \log(1-\alpha) + 1 + H(X_1) - H(X_2) \\
&= \log(1-\alpha) - \log \alpha + H(X_1) - H(X_2) \\
&= \log\left(\frac{1-\alpha}{\alpha}\right) + H(X_1) - H(X_2) = 0 \qquad (6) \\
\Rightarrow \log\left(\frac{1-\alpha}{\alpha}\right) &= H(X_2) - H(X_1) \\
\frac{1-\alpha}{\alpha} &= 2^{H(X_2)-H(X_1)} \\
1 &= \alpha 2^{H(X_2)-H(X_1)} + \alpha \\
1 &= \alpha\left(2^{H(X_2)-H(X_1)} + 1\right) \\
\alpha &= \frac{1}{2^{H(X_2)-H(X_1)} + 1} \qquad (7)
\end{aligned}
$$

From (5), we have:

$$
\begin{aligned}
H(X) &= -\alpha \log \alpha - (1-\alpha)\log(1-\alpha) + \alpha H(X_1) + (1-\alpha)H(X_2) \\
&= -\alpha \log \alpha - \log(1-\alpha) + \alpha\log(1-\alpha) + \alpha H(X_1) + (1-\alpha)H(X_2) \\
&= \alpha \log\left(\frac{1-\alpha}{\alpha}\right) - \log(1-\alpha) + \alpha H(X_1) - \alpha H(X_2) + H(X_2) \\
&= \alpha\left(\log\left(\frac{1-\alpha}{\alpha}\right) + H(X_1) - H(X_2)\right) - \log(1-\alpha) + H(X_2)
\end{aligned}
$$

Using (6), we get:

$$
H(X) = -\log(1-\alpha) + H(X_2)
$$

Thus,

$$
\begin{aligned}
2^{H(X)} &= 2^{-\log(1-\alpha)+H(X_2)} \\
&= 2^{-\log(1-\alpha)}2^{H(X_2)} \\
&= (1-\alpha)^{-1}2^{H(X_2)}
\end{aligned}
$$

using (7), we have:

$$
\begin{aligned}
2^{H(X)} &= \left(1 - \frac{1}{2^{H(X_2)-H(X_1)}+1}\right)^{-1} 2^{H(X_2)} \\
&= \frac{2^{H(X_2)-H(X_1)}+1}{2^{H(X_2)-H(X_1)}} 2^{H(X_2)} \\
&= 2^{H(X_2)} + 2^{H(X_2)-H(X_2)+H(X_1)} \\
&= 2^{H(X_2)} + 2^{H(X_1)} && \text{for max } \alpha
\end{aligned}
$$

5. Continuation from Problem . The distribution of English word length, k, is a shifted Poisson random variable. At the end of each word, we will add a space to separate words. There are no spaces allowed in the middle of a word. Under this assumption word length follows the distribution

$$w(k) = \frac{e^{-\lambda}\lambda^{k-2}}{(k-2)!}; k = 2, 3, 4, \cdots \tag{8}$$

Assuming there are about a million words in the English language.

1. What is the probability of generating a word in the English language randomly choosing letters
   - assuming the letters are chosen uniformly.
   - assuming the letters are chosen according to $p(x)$.[3]

   Solution:

$$\text{Let} \begin{cases} K & = \text{probability of generating a sequence of non-space alphabets followed by a space character} \\ E & = \text{probability of generating a valid English word} \\ N & = \text{total number of English words}, 1000000 \end{cases}$$

Using conditional probability relation we can express $\Pr\{E\}$ as:

$$\Pr\{E\} = \sum_k \Pr\{E \cap K\} = \sum_k \Pr\{E|K = k\} \Pr\{K = k\}$$

where

$$\Pr\{K = k\} = (1 - \Pr\{\text{space}\})^{k-1} \Pr\{\text{space}\}$$

Since $w(k) * N \approx$ number of English words with length k, we can express $\Pr\{E|K = k\}$ as

$$\Pr\{E|K = k\} \approx \frac{w(k) * N}{26^{k-1} \times 1}$$

However, when $k = 2, 3, 4$, using (8) gives us unexpected word counts:

$$\begin{aligned} w(2) &= \frac{e^{-6.1}6.1^0}{0!} \approx 0.0060967 \\ w(2) * 1000000 &\approx 2242 > 26 \\ w(3) &= \frac{e^{-6.1}6.1^1}{1!} \approx 0.013681 \\ w(3) * 1000000 &\approx 13681 > 26^2 \\ w(4) &= \frac{e^{-6.1}6.1^2}{2!} \approx 0.041729 \\ w(4) * 1000000 &\approx 41728 > 26^3 \end{aligned}$$

Note that I am using $\lambda = 6.1$ since we are adding an additional space at the end of each word.

For this reason, in order to get a reasonable estimation, I will use some data I found online for $k = 2, 3, 4$ terms.

According to Wikipedia,

- Number of 1 letter-words = 3
  source: `https://en.wiktionary.org/wiki/Category:English_one-letter_words`

[3]See Problem .

- Number of 2 letter-words = 114
  source: `https://en.wiktionary.org/wiki/Category:English_two-letter_words`
- Number of 3 letter-words = 172
  source: `https://en.wiktionary.org/wiki/Category:English_three-letter_words`

Thus, my estimation for generating a word in English is:

$$\begin{aligned}
\Pr\{E\} &= \sum_{k=2}^{N}\left(\frac{w(k)*N}{26^{k-1}}(1-\Pr\{\text{space}\})^{k-1}\Pr\{\text{space}\}\right) \\
\Pr\{E\} &= \frac{3}{26}(1-\Pr\{\text{space}\})\Pr\{\text{space}\}+ \\
&\frac{114}{26^2}(1-\Pr\{\text{space}\})^2\Pr\{\text{space}\}+ \\
&\frac{172}{26^3}(1-\Pr\{\text{space}\})^3\Pr\{\text{space}\}+ \\
&\sum_{k=5}^{N}\left(\frac{w(k)*N}{26^{k-1}}(1-\Pr\{\text{space}\})^{k-1}\Pr\{\text{space}\}\right)
\end{aligned}$$

So,

- If letters are chosen uniformly,

$$\begin{aligned}
\Pr\{\text{space}\} &= \frac{1}{27} \\
\Pr\{E\} &= \sum_{k=2}^{N}\left(\frac{w(k)*N}{26^{k-1}}\left(\frac{26}{27}\right)^{k-1}\frac{1}{27}\right) \\
&= \sum_{k=2}^{N}\left(\frac{w(k)*N}{27^k}\right) \\
&= \frac{3}{27^2}+\frac{114}{27^3}+\frac{172}{27^4}+\sum_{k=5}^{N}\left(\frac{w(k)*N}{27^k}\right) \\
&\approx 0.01649
\end{aligned} \tag{9}$$

- If letters are chosen according to $p(x)$,

$$\begin{aligned}
\Pr\{\text{space}\} &= \frac{40000}{222303} \approx 0.17993 \\
\Pr\{E\} &= \frac{3}{26}\frac{182303}{222303}\frac{40000}{222303}+\frac{114}{26^2}\left(\frac{182303}{222303}\right)^2\frac{40000}{222303}+\frac{172}{26^3}\left(\frac{182303}{222303}\right)^3\frac{40000}{222303}+ \\
&\sum_{x=5}^{N}\left(\frac{w(k)*N}{26^{k-1}}\left(\frac{182303}{222303}\right)^{k-1}\frac{40000}{222303}\right) \\
&\approx 0.054269
\end{aligned} \tag{10}$$

Listing 3: Code used to compute the probability in (9) and (10)

```
(defn w [k & {:keys [lambda] :or {lambda 6.1}}]
  (let [p (fn [x] (/ (* (Math/exp (- lambda)) (Math/pow lambda (- x 2)))
                   (fact-cache (- x 2))))]
    (if (coll? k)
      (map p k)
      (p k))))
```

```
(defn Pr_E|K
  [Pr_space k & {:keys [N] :or {N 1000000}}]
  (* (/ (* (w k) N) (Math/pow 26 (dec k)))
     (Math/pow (- 1 Pr_space) (dec k))
     Pr_space))

(defn Pr_E
  [Pr_space & {:keys [K] :or {K 100}}]
  (let [Pr_letter (- 1.0 Pr_space)]
   (+ (* (/ 3 26) Pr_letter Pr_space)
      (* (/ 114 (Math/pow 26 2)) (Math/pow Pr_letter 2) Pr_space)
      (* (/ 172 (Math/pow 26 3)) (Math/pow Pr_letter 3) Pr_space)
      (loop [k 5
             s 0.0]
        (if (<= k K)
          (recur (inc k) (+ s (Pr_E|K Pr_space k)))
          s)))))
```

2. What is the probability of randomly generating W words in a row under each distribution?
Solution:

   - If letters are chosen uniformly: 0.01649^W
   - If letters are chosen according to $p(x)$: 0.054269^W

3. How many times must we randomly sample before getting W words in a row under each distribution?
Solution:
Given a event's probability of occurrence is p, the expected number of trial to the first success should be $\frac{1}{p}$. Thus

   - Uniform distribution: we need $\lceil 0.01649^{-W} \rceil$ number of trials.
   - For $p(x)$, we need $\lceil 0.054269^{-W} \rceil$ number of trials.

4. If a word is w_k characters long (including a space at the end), assume it requires $k \log_2 N$ bits to form. How many bits do we use, on average, to query to the point we have W words in a row?
Solution:
Number of bits require to encode a sequence is $W \cdot H(E)$

Average length of random word can be calculated by:

$$\sum_{k=2}^{\infty} k(1 - \Pr\{\text{space}\})^{k-1} \Pr\{\text{space}\}$$

For both cases, they are computed numerically using code shown in (4)

Listing 4: Code used to compute expected random word length

```
(defn E_l
  [p & {:keys [K] :or {K 1000}}]
  (let [q (- 1 p)]
    (loop [k 2
           s 0.0]
      (if (<= k K)
        (recur (inc k)  (+ s (* k (Math/pow q (dec k)) p)))
        s))))
```

Computed results are:

   - Expected random word length using uniform distribution is $26.8433 \approx 27$.
   - Expected random word length using $p(x)$ is $5.378 \approx 6$.

So for uniform distribution we need $W(27\log_2 N)\left\lceil 0.01649^{-W}\right\rceil$ bits, and for $p(x)$ we would need $W(6\log_2 N)\left\lceil 0.054269^{-W}\right\rceil$ bits.