Homework 3

Team:

Rithu Anand Krishnan
Aman Bhardwaj
Manvi Mahajan

Communication consists of the following components:

1) Information Source: It produces a message or a sequence of characters as a function of time.
2) Transmitter: encode or convert the message. It is also responsible for compression and adapting to channels.
3) Channel: Physical transmission
4) Receiver: It is the reverse function of the transmitter
5) Destination

Communication: reproduce at one point a message selected at another point. Information should depend on the *possibilities*, e.g. the set of messages one can choose from. Information should be a monotonic function of the possible number of messages N that *could* have been sent.

Formula: I = $\log_2(N)$

Adding one bit in a memory doubles the possible states. Two punched cards square the number of possible messages.
Log simplifies the maths a lot. The minimal amount of information (N=2) is 1 (*bit*).

How much information we can transmit per unit of time is called the capacity of the channel

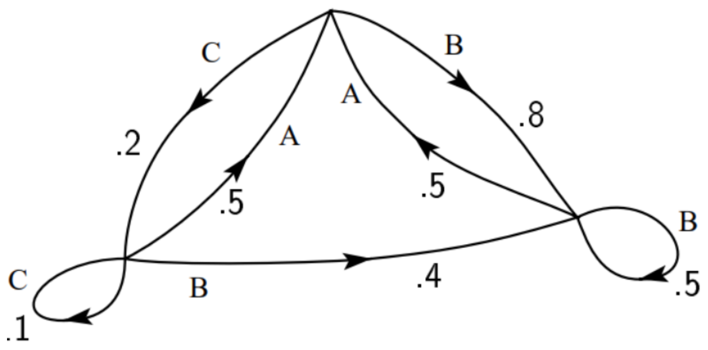
$$C = \lim_{t \to \infty} \frac{\log(N(t))}{t}$$

Sources

A *source* generates messages, one symbol after the other. By considering all messages that a source *may* have sent, along with the probabilities of occurrence, we can see a source as a stochastic symbol generator. Real life messages come from natural written language (NL). The underlying stochastic process is almost impossible to define, not to say compute properly. Shannon thesis: relatively simple artificial stochastic processes can provide a reasonable approximation of NL.

Markov sources

1) Consider a finite set of n states
2) For each set i,
    - pi(j): probability to go to state j
    - pi(s): probability to output symbol s
    - right example: states and symbols are identified
3) This defines a Markov source



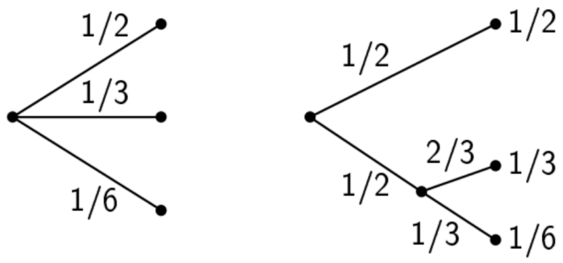
Ergodic and mixed sources

Ergodicity is a very common property.When the graph is not strongly connected, we have a mixture along the recurrent strongly connected components: this is called a mixed source. Example of mixed source: the source can possibly speak in two languages (but each message is in one language) Take-away: the statistical properties of natural language can be approximated by an ergodic Markov source. Any long enough message will have the same statistical properties.

Entropy of a probability distribution

- Remind: information should mix possibilities and probabilities

- Consider a distribution p=p1,…,pn
- We would like a measure H(p) that verifies:
  - H is continous
  - H is monotonic:
    if p1=1/n1,…,1/n1 and p2=1/n2,…,1/n2 with n1<n2,
    then H(p1)<H(p2)
  - H is *associative*, i.e. for example
  - H(1/2,1/3,1/6) = H(1/2,1/2) + 1/2H(2/3,1/3)



Entropy of a probability distribution

Theorem: The only H that checks continuity, monotonicity and associativity is of the formH=−K∑ipilog(pi)

Properties of entropy

1) H=0 iff pi=δi0i for some i0 (certainty).
2) For fixed n, max(H)=log(n), reached uniquely for the uniform distribution.
3) If x and y are two events (distributions living in the same universe), max(H(x),H(y))≤H(x,y)≤H(x)+H(y)
   - l.h.s. is reached when x and y are fully correlated (knowing one is knowing the other)
   - r.h.s. is reached when x and y are independent (p(i,j)=p(i)p(j))
4) Any smoothing/averaging/convolution of the distribution increases the entropy

Conditional entropy

For x and y, the conditional entropy of y (w.r.t x) is the weighted average of the entropy of y when x is known.

$H_x(y)=\sum_i p_i \sum_j -p_i(j)\log(p_i(j))=-\sum_{i,j} p(i,j)\log(p_i(j))$
Properties:

- $H(x,y)=H(x)+H_x(y)$
- $H_x(y)\le H(y)$ (equality iff independence)

Entropy of a Markov source

1) Remind: n states, $p_i(j)$, $p_i(s)$, P_i.
2) Each state has a symbol entropy H_i
3) Entropy of the source: $H=\sum_i P_i H_i=-\sum_{i,s} P_i p_i(s)\log(p_i(s))$
4) Unit: bits per symbol produced
5) Can be converted into bits per second

Links between entropy of a Markov source and produced messages

1) Let $p(B_i,s)$ be the probability of a sequence B_i+s, $p_{B_i}(s)$ the conditional probability
   - $F_N:=-\sum_{|B_i|=N-1,s} p(B_i,s)\log(p_{B_i}(s))$
   - F_N decreases, $F_N\le G_N$, and $\lim F_N=H$ (F is a better approximation than G)
   - F_N is the conditional entropy of a symbol when the $N-1$ previous ones are known (Nth order approximation).
2) It's a recipe to estimate H from a large corpus (compute probabilities using a sliding window of size N).
3) Seed of joint complexity

Entropy of English

1) Using F8 estimatation, Shannon proposes 50% for English
2) Corroborrated with other experiments (e.g. how many letters can we suppress until a human cannot understand the text)
3) Application to crosswords:
   - Crosswords are possible because redundancy is less than 50%.
   - 3D crossword would require redundancy less than 33%.
4) Note: actual compression rates are greater (5%-20%)
   - Better long term estimates
   - Bytes are not letters

Transducers

1) Transduction (encoding/decoding at transmitter/receiver) is modeled by a device with internal memory (finite).
2) When a new input symbol x_n is fed to a transducer in state α_n, two deterministic operations are performed:
   - An output symbol $y_n=f(x_n,\alpha_n)$ is produced;
   - A new state $\alpha_{n+1}=g(x_n,\alpha_n)$ is set.
   - A transducer is *non-singular* is there exists an *inverse* transducer that can recover the input from the output.

Transducers

Theorem: *The output of a finite state transducer driven by a finite state statistical source is a finite state statistical source, with entropy (per unit time) less than or equal to that of the input. If the transducer is non-singular they are equal.*

Proof:

1) trivial for stateless (one state) transducers
2) multi-state: sum over the product-state space

Interpretation: the best thing a transducer can do is not to destroy information

The fundamental theorem for a noiseless channel

Let a source have entropy H (bits per symbol) and a channel have a capacity C (bits per second). Then it is possible to encode the output of the source in such a way as to transmit at the average rate $CH-\epsilon$ symbols per second over the channel where ϵ is arbitrarily small. It is not possible to transmit at an average rate greater than C/H

Problem 2: Scraping, Entropy and ICML papers.

Scraping all pdfs from the link - http://proceedings.mlr.press/v70/ with BeautifulSoup

In [1]:
```
import requests
from bs4 import BeautifulSoup
import io
from PyPDF2 import PdfFileReader
```

In [2]:
```
url = "http://proceedings.mlr.press/v70/"
read = requests.get(url)
html_content = read.content
soup = BeautifulSoup(html_content, "html.parser")
```

In [3]:
```
#Getting a list of all the pdfs from the link above
list_of_pdf = set()
for i in soup.find_all('a'):
    pdf_list = (i.get('href'))
    if "http" not in pdf_list or ".pdf" not in pdf_list:
        continue
    print(pdf_list)
    list_of_pdf.add(pdf_list)
```

```
http://proceedings.mlr.press/v70/achab17a/achab17a.pdf
http://proceedings.mlr.press/v70/achab17a/achab17a-supp.pdf
http://proceedings.mlr.press/v70/acharya17a/acharya17a.pdf
http://proceedings.mlr.press/v70/acharya17a/acharya17a-supp.pdf
http://proceedings.mlr.press/v70/achiam17a/achiam17a.pdf
http://proceedings.mlr.press/v70/achiam17a/achiam17a-supp.pdf
http://proceedings.mlr.press/v70/agarwal17a/agarwal17a.pdf
http://proceedings.mlr.press/v70/agarwal17a/agarwal17a-supp.pdf
http://proceedings.mlr.press/v70/akrour17a/akrour17a.pdf
http://proceedings.mlr.press/v70/aksoylar17a/aksoylar17a.pdf
http://proceedings.mlr.press/v70/aksoylar17a/aksoylar17a-supp.pdf
http://proceedings.mlr.press/v70/alaa17a/alaa17a.pdf
http://proceedings.mlr.press/v70/alaa17a/alaa17a-supp.pdf
http://proceedings.mlr.press/v70/ali17a/ali17a.pdf
http://proceedings.mlr.press/v70/ali17a/ali17a-supp.pdf
http://proceedings.mlr.press/v70/allamanis17a/allamanis17a.pdf
http://proceedings.mlr.press/v70/allamanis17a/allamanis17a-supp.pdf
http://proceedings.mlr.press/v70/allen-zhu17a/allen-zhu17a.pdf
http://proceedings.mlr.press/v70/allen-zhu17b/allen-zhu17b.pdf
http://proceedings.mlr.press/v70/allen-zhu17c/allen-zhu17c.pdf
http://proceedings.mlr.press/v70/allen-zhu17d/allen-zhu17d.pdf
http://proceedings.mlr.press/v70/allen-zhu17e/allen-zhu17e.pdf
http://proceedings.mlr.press/v70/amos17a/amos17a.pdf
```

```
http://proceedings.mlr.press/v70/zilly17a/zilly17a-supp.pdf
http://proceedings.mlr.press/v70/zoghi17a/zoghi17a.pdf
http://proceedings.mlr.press/v70/zoghi17a/zoghi17a-supp.pdf
```

In [6]:
```
def info(pdf_path):
    # used get method to get the pdf file
    response = requests.get(pdf_path)

    # response.content generate binary code for
    # string function
    with io.BytesIO(response.content) as f:

        # initialized the pdf
        try:
            pdf = PdfFileReader(f, strict=False)
        except Exception:
            return ""
        # all info about pdf
        information = pdf.getDocumentInfo()
        number_of_pages = pdf.getNumPages()
        content = ""
        for page_number in range(number_of_pages):
            page = pdf.pages[page_number]
            content = content + page.extract_text()
        #print(page.extract_text())

    return content
```

In [7]:
```
# Copying all the contents of these pdf files into pdf_content dict
j = 0
pdf_content = {}
for i in list_of_pdf:
    pdf_content[j] = info(i)
    j = j+1
```

In [23]:
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import string

#Converting this pdf_content into a dataframe to perform some data cleani
pdf = pd.DataFrame(pdf_content.items(), columns=['id','content'])
pdf['content'] = pdf['content'].str.replace('[{}]'.format(string.punctuat
pdf['content'] = pdf['content'].str.encode("ascii", "ignore")
pdf['content'] = pdf['content'].str.decode("ascii")
pdf['content'] = pdf['content'].str.encode("utf-8", "ignore")
pdf['content'] = pdf['content'].str.decode("utf-8")
pdf = pdf.replace('\n',' ', regex=True)
pdf = pdf.replace('\d+',' ', regex=True)
```

```
/var/folders/gp/xs_xkf814zx898m2wdld0hw40000gn/T/ipykernel_25469/42364348
97.py:8: FutureWarning: The default value of regex will change from True
to False in a future version.
  pdf['content'] = pdf['content'].str.replace('[{}]'.format(string.punctu
ation), '')
```

In [24]:
```
pdf.head(2)
```

Out[24]:

	id	content
0	0	The Loss Surface of Deep and Wide Neural Netwo...
1	1	GramCTC Automatic Unit Selection and Target De...

In [25]:
```
#Writing this pdf content into a text file
with open("pdf_content.txt", 'a') as f:
    dfAsString = pdf.to_string(header=False, index=False)
    f.write(dfAsString)
```

In [56]:
```
# Reading the text file back to perform some more pre-processing
import nltk
nltk.download('punkt')
raw = open('pdf_content.txt').read()
```

```
[nltk_data] Downloading package punkt to /Users/ri/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [57]:
```
from nltk import pos_tag, word_tokenize, wordpunct_tokenize
nltk.download('words')
from nltk.corpus import words

# Removing non english words from the pdfs
def removeNonEnglishWordsFunct(x):
    words = set(nltk.corpus.words.words())
    filteredSentence = " ".join(w for w in nltk.word_tokenize(x) \
                                if w.lower() in words)
    return filteredSentence

fresh_data = removeNonEnglishWordsFunct(raw)
```

```
[nltk_data] Downloading package words to /Users/ri/nltk_data...
[nltk_data]   Package words is already up-to-date!
```

In [45]:
```
# Writing the content back to a sample file, which will be used further f
text_file = open("sample.txt", "wt")
n = text_file.write(fresh_data)
text_file.close()
```

1. What are the top 10 common words in the ICML papers?

In [46]:
```
from nltk.probability import FreqDist

raw = open('sample.txt').read()

tokens = word_tokenize(raw)
words = [w.lower() for w in tokens]

# Finding the frequency of all words
fdist = FreqDist(words)
word_freq_df = pd.DataFrame(fdist.items(), columns=['word', 'frequency'])
```

In [47]:
```
# Printing the top 10 most common words - stopwords are not removed
word_freq_df.sort_values(by='frequency', ascending=False).head(10)
```

Out[47]:

	word	frequency
0	the	205775
3	of	100241
5	and	74580
47	a	71322
16	in	67867
34	to	62905
39	we	50858
46	for	47449
13	is	44338
18	that	35409

2. Let Zbe a randomly selected word in a randomly selected ICML paper. Estimate the entropy of Z.

In [48]:
```
# Entropy or H is the minimum no of bits that are needed to be sent acros
# H = - sum ((probability of i) * log (probability of i))
import math
# total number of words
total_words = word_freq_df["frequency"].sum()
word_freq_df["per_word"] = word_freq_df["frequency"]/total_words

word_freq_df["log_entropy"] = word_freq_df["per_word"].map(lambda x: math
word_freq_df["entropy"] = word_freq_df["per_word"] * word_freq_df["log_en
```

In [59]:
```
total_entropy = -1 * word_freq_df["entropy"].sum()
# Priting the entropy value
print("The entropy value of Z is ", total_entropy)
```

```
The entropy value of Z is  8.896772673091544
```

3. Synthesize a random paragraph using the marginal distribution over words.

In [50]:
```
para = np.random.choice(word_freq_df["word"], 100, p=word_freq_df["per_wo
full_para = ' '.join([str(elem) for elem in para])
full_para
```

```
Out[50]: 'or in our in achieve conference estimator a share xi while b is with of
s test for result two handle greedy we it a substance any per j to operat
or the and the thus of by plane as of provided noise many that yang g lea
rning al target of regression state sense also mathematical and propose p
rior sense state learning algorithm extended not particular and it stocha
stic that the the early of and as appendix sub randomly of gradient in ro
bust me problem an our result learn is with paired convex forb loss appro
ach task optimization the the and'
```

4. (Extra credit) Synthesize a random paragraph using an n-gram model on words. Synthesize a random paragraph using any model you want.

```
In [60]: text_file = open ("sample.txt","r",encoding="utf-8")
pdf_text = text_file.read()
text_file.close()
```

```
In [65]: #Creating the ngram dictionary
ngrams = {}
#Our n is 3 here
pair_words = 3

# Tokenize all the words in the sample
words_tokens = nltk.word_tokenize(pdf_text)

# 3 words are used as key and value consists of all words that are most l
for i in range(len(words_tokens)-pair_words):
    seq = ' '.join(words_tokens[i:i+pair_words])
    if  seq not in ngrams.keys():
        ngrams[seq] = []
    ngrams[seq].append(words_tokens[i+pair_words])
```

In [69]:
```
import random

# Randomly generate a Paragraph by taking random words as first pair
number= random.randint(1,len(words_tokens)-pair_words)

first_pair = ' '.join(words_tokens[number:number+pair_words])
para = first_pair

for i in range(150):
    # Check if this pair is a key
    if first_pair not in ngrams.keys():
        break
    #If yes then take the value of this key
    list_of_words = ngrams[first_pair]
    # Randomly pick the next words
    nxt_words = list_of_words[random.randrange(len(list_of_words))]
    para += ' ' + nxt_words
    seq_words = nltk.word_tokenize(para)
    # Repeat this process with a new set of random pairs
    first_pair = ' '.join(seq_words[len(seq_words)-pair_words:len(seq_wor

print(para,"\n")
```

```
much of an issue as such are unlikely to occur in parallel As a remedy we
introduce the Schema Network and robust generalization beyond these Deep
Related Work The recent empirical success of about the here are qualitati
ve due to that m By the union bound in similar respectively h hi T T wher
e varied depending on C following with least n C Remark us a density leve
l set clustering In th Annual Conference on Learning Theory COLT Benjamin
D and e Global in ten factorization deep learning and demonstrate that op
timization algorithm good L Bonsai was shown to be universal capable of a
rbitrarily complex given capacity al Furthermore recent work shown that c
an preserve of the problem When is used as the context vector into two ti
The t are at every step The dissipation inequality that the dynamics are
the same used in the algorithm Since are chosen as the class
```

In [ ]: