Lecture 3: Compression and Flows

Scribed by: Eugene Lim, Ang Ming Liang, Ng Wen Xian
Lecture presented by: Daniel Maung, Terence Neo, Eloise Lim

Lossless Compression (by Daniel Maung and Terence Neo)

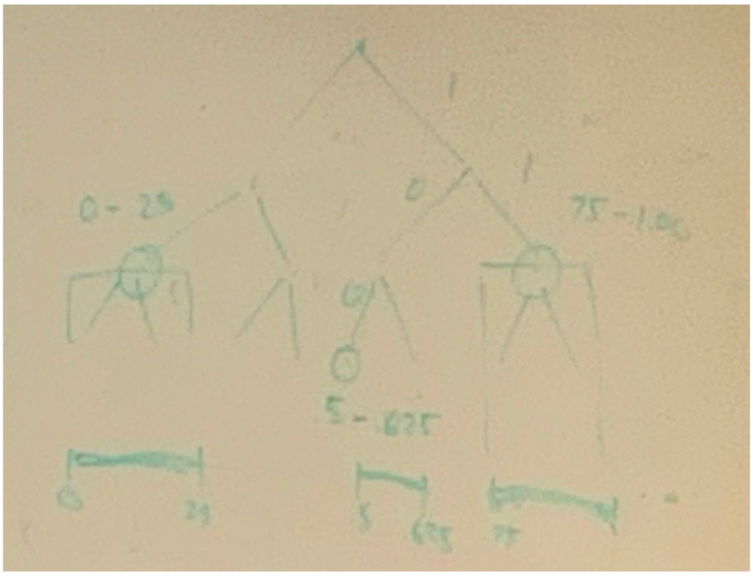
- Universal Compression Scheme
    - No algorithm can compress every bitstream
    - Can be proven by pigeonhole principle and the fact that lossless compression must be bijection (i.e. one to one).
    - Another argument is that if every bitstream can be compressed we can recursively compress each bitstream to 0 which is absurd.
- Coding Symbols
    - Naive coding: cannot exploit structure.
    - Variable-length codes: need to disambiguate. Can be done by stopword (e.g. Morse) or prefix-free code (e.g. Binary Tries).
    - However prefix-free code is sensitive to errors.
    - Huffman coding: optimal prefix-free code
- Theoretical Limits
    - Shannon entropy: expected negative log probability (see [1, 2] for intuition.
        - Why take log in entropy?
            - https://www.quora.com/Why-is-there-a-logarithmic-function-in-the-entropy-formula
            - https://stats.stackexchange.com/questions/87182/what-is-the-role-of-the-logarithm-in-shannons-entropy
            - We can take out the inverse, since negative log is its reciprocal log, hence often entropy is presented as

$$H = -\sum_{i=1}^{n} p_i \log p_i$$

        - Shannon theorem: The length of the code is equal or larger than the entropy of the message.
            - Proof uses Jensen's inequality[3]

    - Kraft McMilan Inequality:
        - For any uniquely decodable code C, we have that $\sum_{(s,w) \in C}$ 2^(-l(w)) <= 1, l(w) is the length of codeword w for symbol s.
        - For any set of lengths L, if $\sum_{l \in L}$ 2^(-l) <= 1, there is a prefix code C of same size such that l(wi) = li (i=1,..., |L|)

        - The same proof on the slide is on Wikipedia (a little more verbose) https://en.wikipedia.org/wiki/Kraft%E2%80%93McMillan_inequality
- Cross entropy and proof on how it is related to KL-divergence (visual intuition at [2])
- Conditional entropy and conditional entropy has a lower entropy than the original entropy.
- Coding Considerations
    - High entropy - long code length
    - Conditional entropy: Decrease the entropy by using context as a constraint
- Arithmetic Coding
    - Encode a sequence of characters with an interval between 0 and 1
    - Naive attempt: use binary fractional notation and use them as code
    - The probability distribution we use can be obtained from a generative model like an autoregressive model
    - Issues: might straddle, assumes infinite precision



- LZ77
    - Adaptive model
    - Use a sliding window of the history of message to compress
    - Bad if there's no pattern
    - **Pseudo code:**

```
while input is not empty do
    prefix := longest prefix of input that begins in window

    if prefix exists then
        i := distance to start of prefix
        l := length of prefix
        c := char following prefix in input
    else
        i := 0
        l := 0
        c := first char of input
    end if

    output (i, l, c)

    s := pop l+1 chars from front of input
    discard l+1 chars from front of window
    append s to back of window
repeat
```

Cut-aways:

- This is a nice explanation of the compression method, “Byte-Pair Encoding” to compress no. of “words” in the vocabulary used for modern deep learning based NLP tasks) https://pdfs.semanticscholar.org/1e94/41bbad598e181896349757b82af42b6a6902.pdf
- http://www.infinitepartitions.com/art001.html
- One way of making the LZ77 encoding more effective is to first do a BWT transform[4] then apply the LZ77 compression algorithm. This is trick is often used by computational biologist to compress large genomic data file.
- Visualization of different compressions:
    - https://people.ok.ubc.ca/ylucet/DS/Huffman.html

Related Links

[1] https://math.stackexchange.com/questions/331103/intuitive-explanation-of-entropy
[2] https://colah.github.io/posts/2015-09-Visual-Information/
[3] https://www.quora.com/What-is-an-intuitive-explanation-of-Jensens-Inequality
[4] https://en.wikipedia.org/wiki/Burrows%E2%80%93Wheeler_transform

Normalizing Flow (by Eloise Lim and Amit Prusty)

- AR cannot use for continuous distributions
- Naive: Discretize the space and use AR
- Can use GMM, but doesn't work for high dimensional data
- Think in CDF to make sampling simple.
    - Sample from Uni(0, 1)
    - Put it through the inverse CDF
- Mapping to z(noise)
    - Called ‘noise’ because it doesn’t make sense without the mapping from the original data

- Need a formula relating p(x) and p(z). Unfortunately, it involves the determinant of the Jacobian. Thus, need to make sure determinant of the Jacobian of f is easy (i.e. take an upper or/and lower triangular matrix) to calculate and differentiate.
- The model is learnt using maximum likelihood.
- One interesting feature of flow is the ability to stack flows together and compose flow models to form larger flow models. This allows you to break up a complex flow model and learn them each input feature at a time, allowing you to capture non-linearities better. However you lose the “O(1)” sample and inference features of flows.
    - Might be relevant: https://en.wikipedia.org/wiki/Universal_approximation_theorem (talks about the expressive power of networks and the ability to represent a wide variety of functions with universal approximators)
- In order to make the model learn using any arbitrary neural network model one way is to split the input data i.e. RealNVP [1,2]

Keywords Definitions:

- Quantization (into different bins):
    - sort of compression nearby information to the same point
- Gaussian distribution, aka. Normal distribution.
    - Known as Gaussian because it’s discovered by Carl Friedrich Gauss

Related Links

[1] **https://arxiv.org/abs/1605.08803**
[2] https://lyusungwon.github.io/generative-models/2018/07/18/realnvp.html
https://www.shortscience.org/paper?bibtexKey=journals/corr/1605.08803#hlarochelle