

# ADVANCED MACHINE LEARNING

András Kornai

AML 2024/11/27

# PLAN FOR TODAY

- Speech emotion recognition: *Gedeon Kövér*
- 
- Algorithmic compression of weighted languages

# ALGORITHMIC COMPRESSION OF WEIGHTED LANGUAGES

- Goal: minimize sum of model length and residual data length (error)
- Both are measured in bits
- For a finite example, such as the Hungarian proquants discussed last time, the baseline is brute force: encode the strings, encode the probabilities, build a simple automaton – about 10k bits
- Where can we save?
- First, we don't need *exact* precision – a corpus is just a sample from an infinite population, our interest is with population probabilities
- Every sample has some noise, how do we measure how much?
- Ideally: get another sample, compare the two

# COMPARING SAMPLES

- Major measures:  $L_2$  (Euclidean distance) not very useful
- $L_p$  in general not very useful
- “Earth mover” sometimes useful, but hard to compute
- And the winner is cross-entropy:
- Kullback-Leibler (KL) *approximation error*  $Q$  of  $q$  relative to  $p$  is  $\sum_{\alpha \in \mathcal{S}(q)} p(\alpha) \log(p(\alpha)/q(\alpha))$ .
- The case when  $q = 0$  will be discussed later

# INTERNAL NOISE

- When we can't take another sample, we randomly divide the existing sample in two, and compute the cross-entropy
- When you do this with proquants, you get  $KL \sim 7 \cdot 10^{-5}$
- This means there is no reason to approximate the probabilities better than  $10^{-5}$
- This is very general: no need for better precision than what's inherent in the data!
- Our computations are finite precision anyway, but surely you don't need 64 bits
- How many bits do you actually need?

# UNIFORM QUANTIZATION

- As a first approximation, let's divide the  $[0,1]$  interval in  $2^b$  small intervals, this is known as *uniform quantization to  $b$  bits*
- For any probability  $p$  we could just use the center-point of the interval it fits in. Transmitting the interval takes  $b$  bits
- For the entire table this is  $160b$  bits
- What is the error of this method? We estimate both worst case and expected
- Can we improve on this, seeing a lot of zeros (24% of the data)?
- Yes, and we can amortize the compression trick (not transmitting the zero probabilities)

# ERROR OF UNIFORM QUANTIZATION

- Since we are not transmitting the zeros, we have a smallest probability  $p_{min} > 0$  and we use  $b$  large enough for  $P_{min} > 1/2^{b-2}$  (leaving the first four bins empty).
- Usually 32 bits suffice for this, SRILM uses 64 bit quantities
- Sum of reconstructed values (interval midpoints) need not be 1, so receiver (Bob) renormalizes: if  $\sum q_i = r$ , he will use  $\bar{q} = q_i/r$  instead of the  $q_i$  that were transmitted by Alice.
- When  $b$  is large, the  $p_i$  will be distributed uniformly mod  $2^{-b}$ . In this case, the expected values  $E(p_i - q_i)$  are zero for all  $i$ , so  $E(\sum q_i) = \sum E(q_i) = \sum E(p_i) = E(\sum p_i) = 1$  or, in other words,  $E(r) = 1$ . Since  $\text{Var}(r - 1) = \sum_i \text{Var}(p_i - q_i) = k/12n^2$  is on the order  $1/n^2$ , renormalization can be ignored, and the the KL approximation error is
- $Q = \sum_{i=0}^{n-1} \sum_{i/n \leq p_j \leq (i+1)/n} p_j \Delta(p_j^i)$  where  $\Delta(p_j^i) = \log(2np_j/(2i + 1))$

# ERROR OF UNIFORM QUANTIZATION CONT'D

- $\Delta$  is maximal if  $p_j$  is at the low end of the interval,  $\log(\frac{2i+1}{2^i})$ . Using  $\log(1+x) \leq x$  this will be less than  $\frac{1}{2^i} \leq 1/8$  since  $i \geq 4$
- Therefore  $Q_n \leq \frac{1}{8 \log 2} \sim 0.18$  independent of  $n$  (as long as the first four bins are empty).
- This can be improved as  $b$  grows, but more important than the maximum error is the *expected* error, for which we have 
$$E(Q_n) \leq \frac{1}{8n \log 2}$$



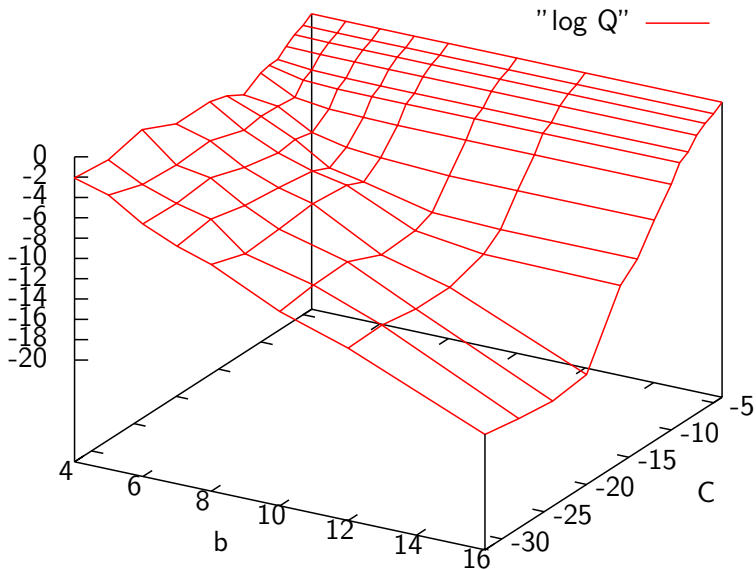
# LOG QUANTIZATION

- Probabilities are not distributed uniformly (many small values) so it's better to deal with log probabilities
- Two-parameter quantization scheme, whereby first  $\log p_j$  are cut off at  $-C$ , and the rest, which are on the  $(-C, 0)$  interval, are sorted in  $n = 2^b$  bins 'b-bit quantization'
- The expected value of the binning error is no longer zero, but

rather 
$$\int_{-C \log \frac{i+1}{n}}^{-C \log \frac{i}{n}} e^x - e^{-C \frac{i+0.5}{n}} dx \sim C^3/8n^3$$

- For  $C, n$  sufficiently large for the first 4 bins to remain empty, the approximation error  $L_n^C$  of log-uniform quantization with cutoff  $-C$  into  $n = 2^b$  bins  $[-C(i+1)/n, -Ci/n)$  is bounded by  $L_n^C \leq \frac{C}{2n \log 2}$  and the expected value  $E(L_n^C)$  is bounded by  $C^2/4n^2 \log 2$ .

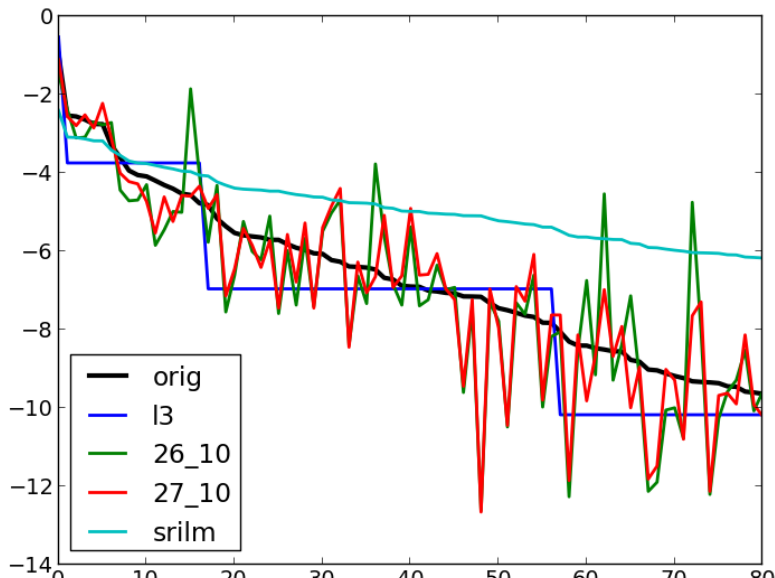
# OBSERVED QUANTIZATION ERROR



## MORE ON THE TREATMENT OF ZEROS

- A string will be deemed *ungrammatical* or *structurally excluded* iff every generation path includes at least one zero weight in the above sense
- This is NOT about measurement error! We are willing to say  $p(\text{teh})=0$  even if 'teh' is frequently observed in the corpus (34,174th most frequent word)
- We will need to model typos, and it turns out that the log price of the /the/teh/ substitution is about -9.8
- This predicts not just the observed frequency of *teh*, but also those of *weatehr*, *otehr*, *tehy*, *tehre*, *tehft*, *Lutehr*, *tehn*, *tehn*, *anotehr*, *leatehr*, *eitehr*, *whetehr*, *clotehs*, *otehrs*, *ratehr*, *tehse*, ... without adding these to the lexicon.
- Cannot build entire the language model in a single sweep directly on the data.

# MODEL FIT WITH MORE COMPLEX MODELS



# SUMMARY SO FAR

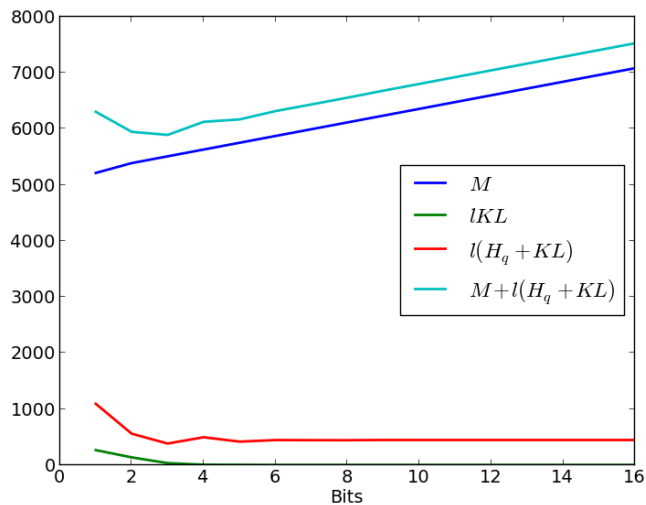
- We transmit probabilities on arcs, strings from states With  $s$  states, and  $b$  bits for probability, an arc requires  $2 \log_2 s + b$  bits.
- WFSA can be assumed to be trimmed
- Zero probs NOT transmitted
- Actual machine costs dominated by strings/architecture not by cost of probs. Assume the character frequencies of Hungarian with entropy  $H$  are shared between Alice and Bob: encoding a string  $\alpha$  costs simply  $|\alpha|H$
- The baseline is always a 'list' automaton
- Here we can obviously do better!

# LIST MODELS WITH CHARACTER-BASED STRING ENCODING

$b$	$l$	$M$	$c_s$	$c_a$	KL	$H_q$
1	121	5210	4306	904	2.1883	6.833
2	121	5386	4306	1080	1.1207	3.487
3	121	5507	4306	1201	0.268	2.889
4	121	5628	4306	1322	0.041436	4.044
5	121	5749	4306	1443	0.016117	3.424
6	121	5870	4306	1564	0.002409	3.667
7	121	5991	4306	1685	0.000676	3.653
8	121	6112	4306	1806	0.000288	3.647
9	121	6233	4306	1927	5.905e-5	3.681
10	121	6354	4306	2048	8.003e-6	3.678
11	121	6475	4306	2169	3.999e-6	3.678
12	121	6596	4306	2290	1.387e-6	3.678
16	121	7080	4306	2774	4.660e-9	3.676

$b$  is the number of bits,  $l$  is the number of trainable parameters (weights associated to arcs),  $c_a$  is the cost of transmitting the arcs.  $c_s$  is the cost of transmitting the emissions, and the total model cost is  $M = c_a + c_s$ . KL gives the KL divergence between the model and the training data. This measures the expected extra message length per arc weight, so that the error residual  $E$  is  $k$  times this value, where  $k$  is the number of values being modeled. We emphasize that  $k = l$  only in the listing format, where all values are treated as independent – in the ‘hub’ model we shall discuss shortly  $l$  is only 26 (10 prefix and 16 suffix weights) but  $k$  is still 121.

# MAIN COMPONENTS OF THE TOTAL MDL COST





# HOGY

- For a weighted language  $p$  a model transform  $X$  is *learnable in principle* (LIP) if (i) both  $\mathcal{M}$  and  $X(\mathcal{M})$  are part of the hypothesis space and (ii) the total MDL cost of describing  $p$  by  $X(\mathcal{M})$  is significantly below that of describing  $p$  by  $\mathcal{M}$
- *hogy* is ambiguous between 'how' and 'that'
- It also provides 40% of the total data
- We consider simple 'hub' models
- And add an extra arc for 'hogy'

# HUB/HOGY MODELS

Lines 1-3: list models; Lines 4-6: hub models (lines 4-6);

Lines 7-9: hubs w/ hogy

$b$	$l$	$M$	$c_s$	$c_a$	KLe5	$M+E$
3	121	1907	705	1202	26800	2289
10	121	2754	705	2049	0.8	3199
12	121	2999	705	2290	0.14	3441
3	26	473	81	392	42343	1305
10	26	662	81	581	32593	1201
12	26	716	81	635	29827	1249
3	27	480	81	400	23094	1052
10	27	676	81	596	11268	1161
12	27	733	81	652	10022	1198

# SINGULARITIES

- Example: Hungarian stem-internal morphotactics. Data from the Analytic Dictionary of Hungarian (Kiss et al 2011)
- Each stem like *beleilleszt* ‘fit in’ is analyzed, here as preverb *bele* + root *ill* + verb-forming suffix *eszt*.
- The analytic categories are Stem  $S$ ; sUffix  $U$ ; Preverb  $P$ ; rooT  $T$ ; Modified  $M$ ; and forelgn  $I$
- Each stem is analyzed as a string over  $\Sigma = \{S, U, P, T, M, I\}$ . We have two weighted languages: the *tYpe-weighted* language  $Y$  and *tOken-weighted* language  $O$  (tokens counted in the Hungarian webcorpus, Halácsy et al., 2004)
- Inherent noise of  $O$  is 0.0474 bits, of  $Y$  0.011 bits. The two languages are very different
- We find types in the analytic dictionary that are not in the corpus, and we find tokens in the corpus whose type is not listed in the dictionary. This makes it theoretically impossible to compute the KL divergence in either direction

# WHAT CAN WE DO?

- Criticize the data! In this case: add the missing entries to the Analytic Dictionary manually.
- Ignore the singular terms (obtain 2.11 bits, not correct but gives an idea)
- Censor the data (classical statistical method)
- Recategorize the data (merge categories until singularities disappear)
- In NLP practice, tokens with no dictionary type are either collected in a single 'unknown' type or are silently discarded.

# THE UNIGRAM DATA

	$f_O$	$P_O$	$KL_O$	$f_Y$	$P_Y$	$KL_Y$
Stem	.7967	.9638	4.7887	.5342	.9122	3.5092
sUffix	.1638	.1464	0.2284	.3443	.5699	1.2174
Modified	.0083	.0103	0.0149	.0255	.0623	0.0928
rooT	.0114	.0141	0.0205	.0331	.0804	0.1209
Preverb	.0198	.0248	0.0362	.0623	.1531	0.2397
forelgn	.0001	.0001	0.0002	.0006	.0010	0.0006

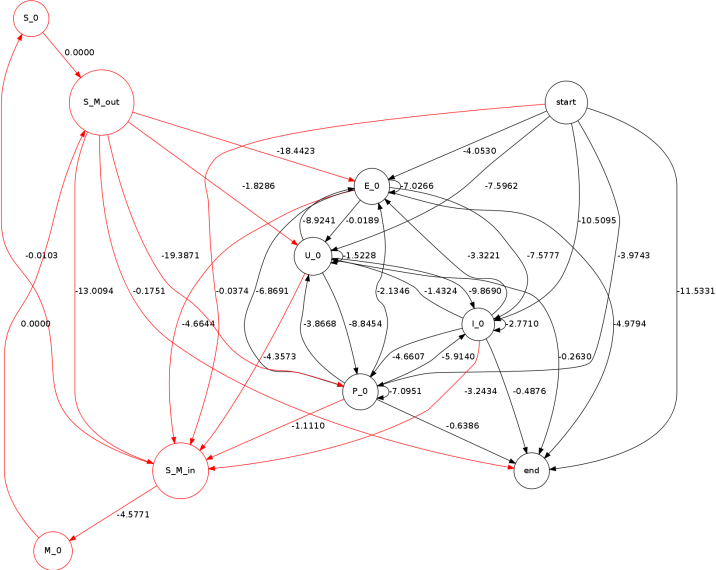
# THE UNSEEN DATA

- If coverage is high, say  $P(\text{unseen}) < 0.05$ , the model that covers the seen data should also be good for predicting the distribution of the unseen data.
- This means 5% or 5% (less than a quarter percent) is really 'unseen unseen' or, as the military likes to say, *unknown unknown* (order of  $P^2$  not order of  $P$ )
- In general we may consider two distributions  $\{p_i\}$  and  $\{q_i\}$  and compute  $P = \sum_{q_i=0} p_i$ , the proportion of  $q$ -singular data in  $p$ .
- The total cost  $L$  of transmitting an item from the  $p$ -distribution is bound by  $L \leq (1 - P)(KL(p, q) + H_q) + P(1 + \log_2 n)$
- Why? We use, with probability  $(1 - P)$ , the  $q$ -based codebook: this will have cost  $H_q$  plus the modeling loss  $KL(p, q)$ . In the remaining cases (probability  $P$ ) we resort to uniform coding at cost  $\log_2 n$ , where  $n$  is the number of singular cases. We need to transmit some information as to which codebook is used: this requires an extra  $H(P, 1 - P) \leq P$  bits

## DISCARDING DATA

- When  $P$  is small, the second term  $P(1 + \log_2 n)$  can be absorbed in the noise
- Consider  $I$  (foreign). By unigram freq, this is 0.06% of  $O$  and 0.013% of  $Y$ . Columns  $KL_O$  and  $KL_Y$  show the KL divergence of  $O$  and  $Y$  from models obtained by discarding words containing the letter in question, columns  $P_O$  and  $P_Y$  show the weight of the strings that are getting discarded.
- For  $Y$ , only  $I$  can be discarded while keeping below the inherent noise of the data, but for  $O$  we have three other symbols  $M$ ,  $E$ , and  $P$ , that could be removed! Further, removing both letters  $M, E$  only produces a KL loss of 0.036 bits; removing  $M, I$  a loss of 0.015 bits;  $E, I$  0.021 bits;  $P, I$  0.036 bits; and even removing all three of  $M, E, I$  only 0.036 bits.
- Discarding  $I$  helps with ignorance of lexicographer (paper+back, base+ball)

# MERGING









# MERGE-SPLIT

- *XY merge-split* transform in two steps: first we replace all letters (or strings)  $X$  by  $Y$  and train a model, and next we split up the emission states of  $Y$  in the merged model to  $X$  and  $Y$ -emissions according to the relative proportions of  $X$  and  $Y$  in the original data.
- Transmission cost is composed of two parts: transmission of the merged model plus transmitting the pair  $X, Y$  and the probability of the split **but this is just the cost of a single arc**
- We can systematically investigate all 6·5 merge-split possibilities. The best deal is to discard I and merge M into S (smallest model 349 bits, about half of best model without these steps, see Kornai, Zséder, and Recski, 2013)

-  Halácsy, Péter et al. (2004). “Creating open language resources for Hungarian”. In: *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*. ELRA, pp. 203–210.
-  Kiss, Gábor et al. (2011). “A Magyar szóelemtár megalkotása és a Magyar gyökszótár előkészítő munkálatai”. In: *MSZNY 2012*. Ed. by A. Tanács and V. Vincze, pp. 102–112.
-  Kornai, András, Attila Zséder, and Gábor Recski (2013). “Structure Learning in Weighted Languages”. In: *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 72–82. URL: <http://www.aclweb.org/anthology/W13-3008>.
-  Leeuw, Karel de et al. (1956). “Computability by probabilistic machines”. In: *Automata studies*. Ed. by C.E. Shannon and J. McCarthy. Princeton University Press, pp. 185–212.

-  Makhoul, John, Salim Roucos, and Herbert Gish (1985). “Vector quantization in speech coding”. In: *Proceedings of the IEEE* 73.11, pp. 1551–1588.
-  Solomonoff, Ray J. (1964). “A formal theory of inductive inference”. In: *Information and Control* 7, pp. 1–22, 224–254.
-  Vitanyi, Paul M. B. and Ming Li (2000). “Minimum description length induction, Bayesianism, and Kolmogorov complexity”. In: *IEEE Transactions on Information Theory* 46.2, pp. 446–464.
-  Widrow, Bernard and István Kollár (2008). *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge University Press.