

# ADVANCED MACHINE LEARNING, LECTURE 12 (WEEK 13)

András Kornai

BME 2020 Dec 3

# ALGORITHMIC COMPLEXITY

- Theory built reductively (in order to make contact with logic)
- First reduction: instead of classification or regression, we are predicting a single function. If we know how to do that, we can order the instances to be classified as  $x_i$ , and the true values  $y_i$ , what we need to predict is  $x_{n+1}, y_{n+1}$
- Second reduction: all you need to predict is a bitstring
- What we require is an algorithm  $A$  that produces the  $i$ -th bit on input  $i$ .
- Initial definition: let the *algorithmic complexity*  $A(s)$  of a bitstring  $s$  be the length (in bits) of the shortest program that computes it

# PROBLEMS WITH THE DEFINITION

- This depends on programming language!
- We can solve this by “programming” Turing machines instead of ordinary hardware
- On ordinary hardware, the closest concept is the “self-extracting archive” e.g. RAR (Windows) or StuffIt (Mac)
- In the Turing machine setup, we fix a universal TM, and prepend the program it runs “prefix complexity”
- Key observation: choosing another UTM just adds a constant factor  $O(1)$
- There remains but one problem:  $A(s)$  is not computable
- Indirect proof: Chaitin’s Theorem: there exists a limit  $L$  such that we can’t prove that  $A(s) > L$  for any  $s$

# PRACTICAL PROBLEMS

- Central issue: there are *incompressible* strings
- Obviously, if we could compress all strings of length  $n$  to at most  $n - 1$  bits we will get collisions by the pigeonhole principle because there are only  $1 + 2 + \dots + 2^{n-1} = 2^n - 1$  of these.
- In fact, for large  $n$  *most* things will be incompressible (Theorem: as  $n \rightarrow \infty$ , proportion of compressible strings  $\rightarrow 0$ )
- Rational numbers have finite algorithmic complexity
- Do algebraic numbers have finite ac? Yes. Why?
- There are even transcendental numbers that have finite ac!
- These are the *constructible* numbers, and there is only measure zero of them: randomly chosen real is incompressible

# ALGORITHMIC COMPLEXITY IN ML

- There is an effective version due to Rissanen called Minimum Description Length (MDL)
- Setup identical to ML: we have some data  $D$  and a hypothesis space  $\mathcal{H}$
- Given some hypothesis  $H \in \mathcal{H}$  we measure the length  $I(H)$  of the hypothesis (called model length) and the length  $I(D|H)$  (called residual data length)
- The view that individual information objects have algorithmic complexity is hard to maintain (think of specialized compression schemes)
- But individual objects have a description length (upper bound on algorithmic complexity)
- It is quite easy to keep the compression scheme constant across models

# AN APPLICATION: HUNGARIAN PROQUANTS

	∅	a	akár	bár	egyvala	más	másvala	minden	se	vala
hány	72383	9502	2432	55					21	4584
hogy	7781539	213687	3173	1839		4570		123	4138	31873
hol	117231	399052	1037	9845		16066		16009	20521	34081
honnan	24777	18628	296	1205		2482		1321	627	4274
honnét	1598	1197	12	25		78		33	23	236
hová	17589	21073	486	1753	1	5073	1	1859	2249	3966
hova	17360	10591	309	1166		1788		1381	2105	3036
ki	1309618	1464744	3933	60923	884		814	308508	165230	221175
meddig	11879	8171	189	225					74	252
mely	761277	1586913	166	74262	3				4	40601
melyik	68051	47564	1996	34477	2				939	48274
mennyi	76429	25805	657	1415					517	96184
mi	1626013	1303820	6500	52480	1337		161		275773	355690
miért	251120	20672	58	205	4				1810	13552
mikor	173652	555325	679	33516		15892		11288	206	18235
milyen	343643	38921	8217	68033		1618	1		55603	81155

# MDL FOR PROQUANTS

- Given some finite alphabet  $\Sigma$ , a *weighted language*  $p$  over this alphabet is defined as a mapping  $p : \Sigma^* \rightarrow \mathbb{R}$  taking non-negative values such that  $\sum_{\alpha \in \Sigma^*} p(\alpha) = 1$
- A WFSA  $\mathcal{M}$  is defined by a square transition matrix  $M$  whose element  $m_{ij}$  give the probability of transition from state  $i$  to state  $j$ , an emission list  $h$  that gives a string  $h_i \in \Sigma^*$  for each  $i \neq 0$ , and an acceptance vector  $\vec{a}$  whose  $i$ -th component is 1 if  $i$  is an accepting state and 0 otherwise. There is a unique initial state which starts the state numbering at 0, and we permit states with empty outputs. Rows of  $M$  must sum to 1.
- Hypothesis space: normalized probability-weighted nondeterministic Moore machines
- Data: a weighted language (divide by total number of words)
- Standard method: SRILM, takes 12 kbytes
- We (Kornai Zséder Recski 2013) minimize  $I(D|H) + I(H)$ , obtain 1052 bits