Advanced Machine Learning, Lecture 5

András Kornai

BME 2020 Oct 8

Homeworks

- Lot of good stuff, some of it exemplary!
- 17 people submitted, pretty skewed distribution: 2 people got 5 points; 2 people got 8; 2 got 9; 2 got 9.5; 9 people got 10
- Lots of people did more than was strictly required to get 10
- Some people used Bartlett's test, great!
- Please start looking at other people's work
- Leaderboard lists some typical errors

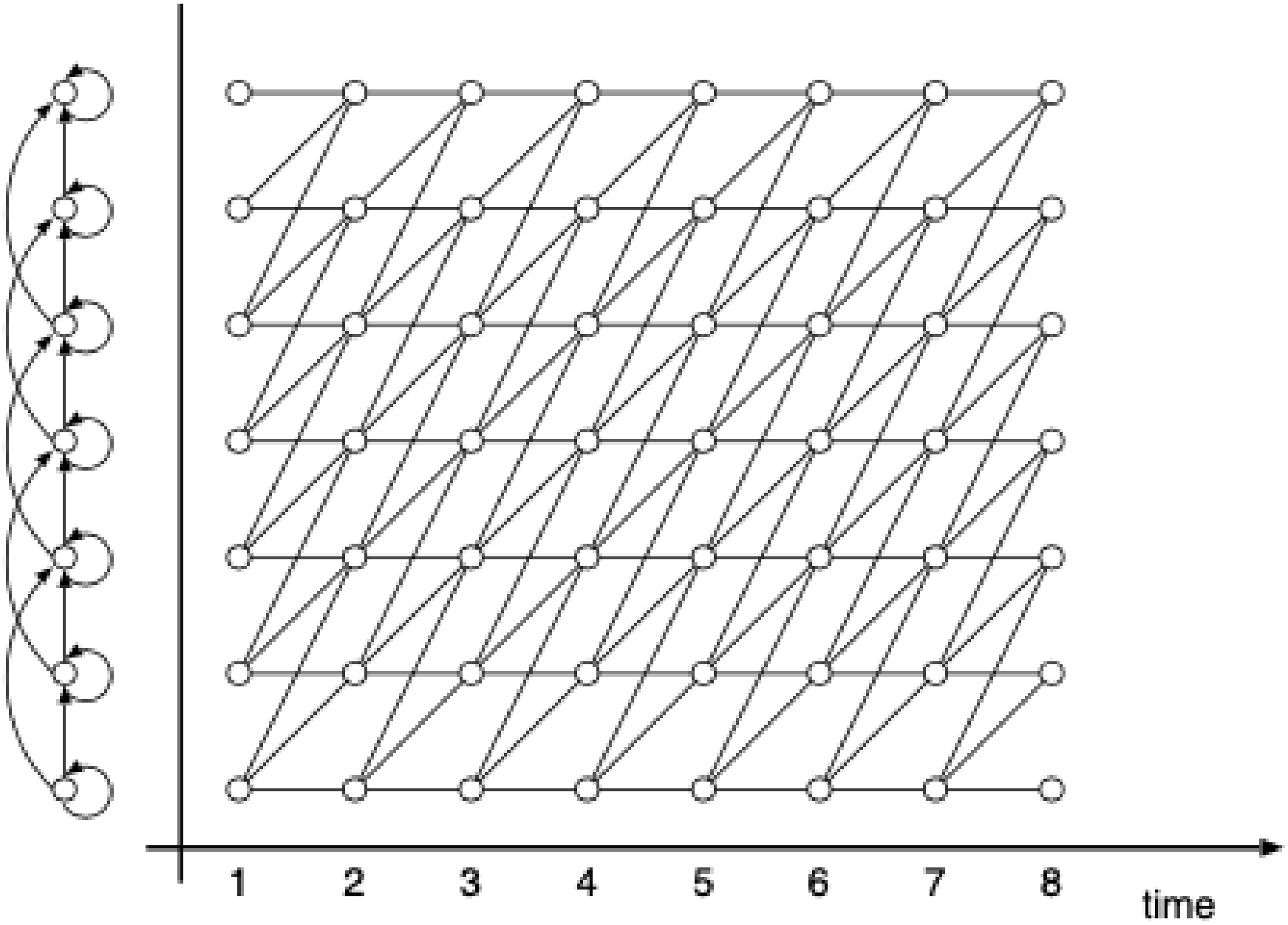
Markov modeling refresher

- Not a single model, but a rich family
- Relevant everywhere where we see Markovian dependency
- Data stream $d_1, d_2, \ldots$ is *first order Markovian* if $p(d_t|d_1, \ldots, d_{t-1}) = p(d_t|d_{t-1})$
- k-th order if it depends on previous k, not just previous 1
- Original example (Markov): probability of a letter in (natural language) text depends on probability of previous few letters

Hidden MMs

- Assume a set of *hidden* (unobservable) states $s_1, \ldots, s_l$
- These are linked by probailistic *transitions* given by a matrix T whose ij element gives the probability of moving from state i to state j
- Each state has its own *emission* function E_i that describes the probability of observing data d if the model is in state i
- Emitted signal can be discrete (from a finite set) or continuous (vectors in Euclidean space)
- Given a model with fixed transition and emission parameters, compute the probability that the model will emit $d_1, d_2, \ldots d_t$. We sum over all the paths of length t. For one path $s_{i_1}, \ldots s_{i_t}$ we have the transition probabilities $\prod_{i=1}^{t-1} T_{i,i+1}$ multiplied with the emission probabilities $\prod_{k=1}^{t} E_{i_k}(d_k)$
- This is l^t paths, very expensive, but there is a clever data structure, the *trellis*, that makes this linear in t

The trellis for an IBM model



Viterbi, EM

- Given an observation sequence $d_1, d_2, \ldots d_t$, find the most likely sequence of hidden states $s_{i_1}, \ldots s_{i_t}$ that could have generated the sequence. This is the *recognition problem*, solved by the Viterbi algorithm
- Given lots of observation sequences, find the model parameters most likely to generate them as Viterbi solutions. This is the *training problem*
- Solved by the expectation maximzation algorithm (Wikpedia has great visualization)
- These algorithms (and other key ones) are available for student presentation
- "Academic" project: give 20-25 minutes presentation on some of these algorithms

Other material for academic projects

- Maximum entropy methods, decision trees
- Genetic/evolutionary methods, boosting
- Nearest neighbor, tangent distance methods
- Algorithmic information theory, Kolmogorov complexity, minimum description length.
- Neural nets (NN), backpropagation.

Feature engineering

The key idea

Replace measurements $m_1, \ldots m_k$ by a set of features $f_i, \ldots, f_r$ computed from the measurements

- Particularly salient in speech recognition (heavily used in NN approaches to ASR as well)
- Slowly (but not entirely) disappearing from NLP
- Gone from vision
- Simplest (linear) version: PCA $k > r$
- A clever nonlinear vesion: kernel trick $k < r$
- Nonlinear but $k > r$: signal preprocessing. Requires solid domain knowledge
- In ASR, $k >> r$: input k is 44.1k stereo 16 bit PCM = 1.411 megabit/sec, output 2 kilobit/sec

What is happening in NLP?

- There is an ongoing "Intro to Python and NLP" course: https://python-nlp.github.io It's more practical than this one, and takes you further in NLP
- If you assume 128k words, at worst you'd need 17 bits to encode a word (in practive 12-15 bits suffice since word frequencies are not uniform. This assumption is compatible with the idea that there are infinitely many words.)
- Word vectors give you 300 dimensional real encodings (higher dimensions and higher precision are often used these days) for 9.6 kilobits/word
- But the word vectors are nice: similar words will have similar vectors (we measure this by cosine similarity more than Euclidean distance, because vector length are proportional to log word frequency)
- What does it mean for words to be similar? That they appear in the same or similar contexts (similar sentences, docs, . . .)

Multimodal Recurrent Neural Network

Our Multimodal Recurrent Neural Architecture generates sentence descriptions from images. Below are a few examples of generated sentences:

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

""little girl is eating piece of cake."

"baseball player is throwing ball in

"woman is holding bunch of bananas."

"black cat is sitting on top of suitcase."

From Visual Question-Answering to Visual Reasoning

Together with the increased performance on the typical computer vision tasks, computer vision transitions into more holistic reasoning systems. One such study is visual question answering, where the visual system is exposed to questions about images. However, even the most challenging existing tasks can still be handled by systems with limited reasoning capabilities. For instance, the state-of-the-art on VQA, the most popular visual question answering dataset, relies heavily on pre-trained visual features. Yet, other elements that are associated with human intelligence like memory, step-by-step planning, compositional thinking, or symbolic manipulation, are often ignored. We want to close the gap between that "fast thinking", which is often impulsive and in this context responsible for a quick interpretation of the visual scene, and "slower thinking" that is more algorithmic. To achieve such goals, we need to think, build, think again, and build suitable benchmarks, architectures and algorithms. Can you create the first system that connects vision with the reasoning in the next three or four years?

Antol et al. VQA: Visual Question Answering. ICCV'15 & CVPR'18

Research directions

In the following research programme, you will

- define what visual reasoning is by building various datasets and tasks
- build architectures that deal with basic reasoning tasks such as analogies, counting, intuitive physics, memory, all grounded in perception
- understand the limitations of the current systems
- draw inspirations from biological systems
- move beyond the standard paradigm of learning from pixels towards reasoning about pixels

Word vector precursors

- Discrete (partial) decomposition of meanings into finite bit vectors is old hat, for example *brother* = '+sibling +male' *sister* = '+sibling –male'
- Continuous begins with Osgood et al. (1975) who asked for judgements on a scale of -3 to +3 and performed PCA on the results
- Next big thing was Landauer, Dumais, etc. who took term-document cooccurrence data
- Create 'term-document matrix' T where T_{ij} counts the number of times term i appears in document j
- Landauer, Dumais etc. applied SVD, reduced T to a few hundred principal components, called it "Latent semantic indexing" and patented it (thereby slowing down developments by 15 years or so – Microsoft didn't get rich on the patents)

Success has many fathers

- Idea first suggested by Schütze (1990)
- First implementation that really worked Bengio and Ducharme (2001)
- NLP "almost from scratch" POS, CHUNK, NER, role labeling Collobert (2011)
- Has linear structure (king–queen=man–woman) Mikolov (2013)
- Why? Pennington et al (2014), Arora (2015), Gittens (2017)

Word embeddings

- Let us define "context as "within a window of $\pm n$ words (typically, $n = 5$). We define PMI(x,y)=log $\frac{p(x,y)}{p(x)p(y)}$
- Actually we tend to ignore negative evidence, and use PPMI = max(0, PMI)
- The big thing is that *supervised* data is obtained cheaply (teraword scale)
- T is now term-term cooccurrence (PPMI) matrix, and we again do SVD for dimension reduction. This way we assign a relatively short vector $\vec{word} \in \mathbb{R}^d$ to each word. This assignment is called the (static) embedding
- Dynamic embeddings (ELMO, BERT) don't have this kind of clean math yet
- In fact, the static was not fully understood until Levy and Goldberg 2014a It worked first, made sense later

Project (midterm) discussion

1. Download your data (not on GitHub, just some local disk)
2. Unless it comes with predefined train/dev/test cut, create one (add filelist to github project repo, not the files themselves)
3. Extend your writeup with a reproducible description of train/dev/test data, your motivation/goals, and SOTA (if there is any – if there is none, say so)
4. By week 7, you will need a baseline system
5. To give you time for these tasks, there is no homework
6. Exception: People who have 10 points or less in columns K+P need to improve their scores