ADVANCED MACHINE LEARNING

András Kornai

AML 2024/10/16

Groups

- NLP: *Acevedo* Aktan Karoiu Tatrishvili
- TrafficSigns: Bodai Oroszki *Szőke* Szűcs
- Fingerprint: Bárdos-Deák Boros Czakó *Kránitz*
- Flower: *Békési* Sooomro Szecskás Wiederschiz
- MRI: *Hermán* Kovács Nguyen Varga
- Simulation: *Máth* Nemes
- SignLg: Barta Nagy Oroszki Szimonenk
- Speech: *Gedeon* Kövér
- Punctuation: Gómez, *Gallego*
- WP: *Juhász*

More HW, project discussion

- You can still pick an ‘academic presentation’ project
- The goal of this is to give a coherent 15-30 minute lecture
- Important for those who didn’t do some part of the homework
- Also for those who want to continue in academia rather than industry
- In earlier years some people were permitted to do this in Hungarian, but not this year

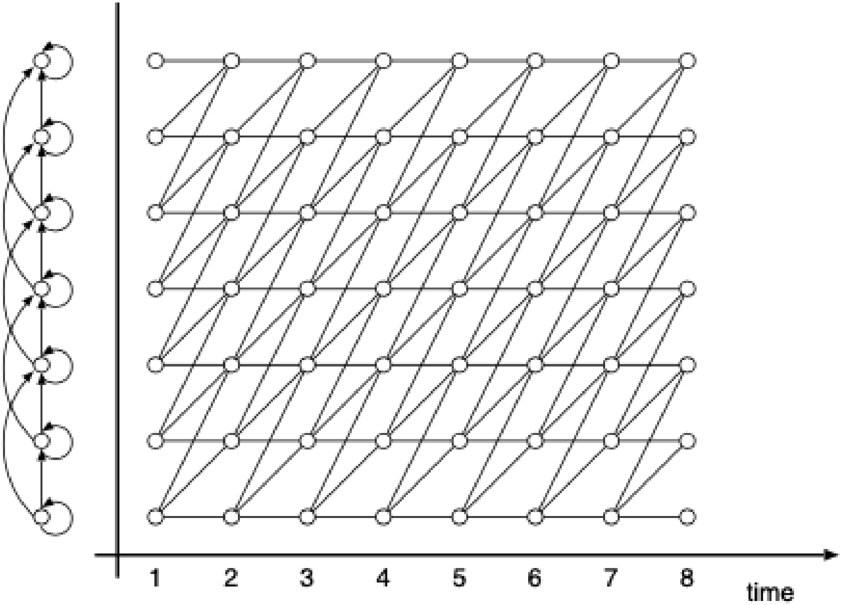
Markov processes

- 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 $s_1, \dots, s_l$
- 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)
- Problem I: given a model with fixed transition and emission parameters, compute the probability that the model will emit $d_1, d_2, \dots d_t$. We sum over all the paths of length t. For one path $s_i 1, \dots s_i t$ 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

Maximum entropy

- If you set up n random linear equations in n unknowns, there will be exactly one solution (with probability 1)
- If you have *more* equations than unknowns, you have no solutions (with probability 1)
- Gauss to the rescue! You will have a unique *best* solution (in the least squares sense)
- What to do when you have *fewer* equations than unknowns?
- There is an infinite number of solutions (with probability 1)
- E.T. Jaynes to the rescue!

Looking for a probability distribution

- We want to estimate $p_1, \ldots, p_n$ such that $\sum_i p_i = 1$
- We have some overall knowledge about events A_i e.g. that $f(A_i) = F_i$ and we know (e.g. from observation) that $\mathbb{E}(f) = c$
- Numerical example: $c = 1.75$ and

event	prob	f
A_1	p	1
A_2	q	2
A_3	r	3

- We have two equations, $p + q + r = 1$ and $p + 2q + 3r = 1.75$, what to do?
- $H = -p \log p - q \log q - r \log r$ is the entropy of the distribution: choose the one for which this is maximal!
- When there is exactly one less equation than unknown, this can be solved analytically

Looking for a pd (2)

- $q + 2r = 0.75$ so $q = 0.75 - 2r$ and $p = 1 - q - r = 0.25 + r$
- Maximize H, minimize -H:
  $(0.25 + r)\log(0.25 + r) + (0.75 - 2r)\log(0.75 - 2r) + r\log(r)$
- After differentiation, we have
  $\log(0.25 + r) - 2\log(0.75 - 2r) + \log(r) = 0$
- Gives rise to quadratic equation with root at $\frac{13-\sqrt{61}}{24} = 0.2162$
- Yes, but what do we do when there are far more equations than variables?
- What is the basic technique of constrained optimization?

Lagrange multipliers

- Our primary interest is not with modeling the distribution p as with joint modeling of distribution classes for best classification. We have a bunch of samples (n-dim vectors whose components are the direct measurements, plus one *class variable* c), and we can use any $\mathbb{R}^{n+1} \to \mathbb{R}$ function f_i as an *indicator* or *feature* to distinguish the classes
- When we have k classes, the ideal features $f_1, \ldots f_k$ would be the indicator functions that take 1 on class j and 0 elsewhere
- We are looking for a distribution with maximum entropy H among all distributions that satisfy constraints given to us in the form $p(f_i) = \tilde{p}(f_i)$ (p is the expected value, and $\tilde{p}$ is the measured value)
- The constraints are $\frac{1}{N}\sum_{d \in D} f_i(d, c(d)) = \frac{1}{N}\sum_{d \in D}\sum_c P(c|d)f_i(d, c)$
- Altogether, we want to maximize the Lagrangian $\Lambda(p, \lambda) = H(p) + \sum_i \lambda_i(p(f_i) - \tilde{p}(f_i))$

Looking for features

- There is a unique distribution p with maximum entropy, and it always has the form $P(c|d) = \frac{1}{Z(d)} \exp(\sum_i \lambda_i f_i(d,c))$
- Here $Z(d)$ is the *partition function* $\sum_c \exp(\sum_i \lambda_i f_i(d,c))$ (required to make sure probabilities sum to 1)
- The maxent feature weights were originally obtained by (Improved) Iterative Scaling, nowadays we use L-BFGS
- A key issue is feature selection, dropping features that are not helpful
- In *extrinsic filtering* we check e.g. correlations with features first
- In *outer loop* or *wrapper* methods we just recompute the model with some features dropped and see if it gets better
- In *embedded methods* we drop features that receive low λ_i weights

An early application

- English-French Machine Translation (Berger et al 1996)
- certain “N de N” phrases are inverted: bureau de poste – post office; compagnie d’assurance – insurance company; ...
- others stay put: pays d’origin – country of origin; somme d’argent – sum of money; ...
- predict which is which based on one of the words or both
- clear tendencies: système de X typically inverts, mois de X typically stays put
- But there are at minimum 50k nouns to be considered, potentially giving rise to 2.5g N de N constructions.

Berger et al (2)

- We can't collect enough data, and can't expect to memorize it!
- What we have is a small sample, maybe 10k pairs labeled for invert/stay
- Let's assume *all* words are relevant, in 1st place, 2nd, or that both words are relevant
- These all contribute to the model before feature selection
- But we keep less than 400
- Generalizes remarkably well to unseen data