Q&A – CIL 2020

Thomas Hofmann

August 18, 2020

Lecture 1

Linear Autoencoder

no questions

Lecture 2

Principal Component Analysis

no questions

Lecture 3

Matrix Approximation & Reconstruction

Shrinkage for Matrix Reconstruction

The most interesting fact about the SVD shrinkage algorithm is that it maintains a sequence of sparse matrices $\mathbf{B}_t$, which is computationally inexpensive compared to maintaining a full matrix (as we assume high degree of sparseness in pratical applications).

$$\mathbf{B}_{t+1} = \underbrace{\mathbf{B}_t}_{\text{sparse}} + \eta_t \underbrace{\Pi(\underbrace{\mathbf{A}}_{\text{sparse}} - \underbrace{\text{shrink}(\mathbf{B}_t)}_{\text{dense}})}_{\text{sparse}} \tag{3.1}$$

The matrix $\mathbf{B}_t$ does not in general converge to $\mathbf{A}$! What we are interested in is to find a sparse matrix $\mathbf{B}$ such that after shrinking it with $\text{shrink}_\tau(\mathbf{B})$ it agrees with $\mathbf{A}$ and is regularized with regard to a combination of nuclear norm and squared Frobenius norm. This is the significance of the theorem on slide 3.33.

$$\lim_{t\to\infty} \text{shrink}_\tau(\mathbf{B}_t) = \underset{\mathbf{B}:\Pi(\mathbf{B})=\mathbf{A}}{\text{argmin}} \left\{ \|\mathbf{B}\|_* + \frac{1}{2\tau}\|\mathbf{B}\|_F^2 \right\} \tag{3.2}$$

So after running the algorithm long enough with a valid step-size sequence, we will get that $\Pi(\text{shrink}_\tau(\mathbf{B}_t)) \approx \mathbf{A}$ which is different from $\mathbf{B}_t \approx \mathbf{A}$. From a computational perspective, what makes the shrink-operator interesting is that it is non-linear, yet can be solved by running an SVD (+ some simple transformation of the singular values).

Lecture 4

Non-Negative Matrix Factorization

pLSA and LDA: Generative Model

There has been some discussion around pLSA not being a generative model. Let me try to clarify this point. Assume that you have fitted a pLSA model to a document collection, resulting in parameters that estimate $p(w_j|z)$ and $p(z|d_i)$. pLSA then gives you a smoothed version of the empirical word distribution

$$\text{empirical:} \quad \hat{p}(w|d_i) = \frac{n(w, d_i)}{n(d_i)} \tag{4.1}$$

$$\text{smoothed:} \quad p(w|d_i) = \sum_z p(z|d_i) p(w|z) \tag{4.2}$$

If a new document d_0 is added (e.g. after the pLSA model has been fitted). We can introduce and fit new parameters $p(z|d_0)$, keeping all other parameters fixed. This gives us a smoothed conditional word distribution for d_0. This process is very fast and is sometimes called 'folding-in'. So pLSA can be useful on new documents by smoothing their empirical distribution. This is good!

However, pLSA does not make a prediction on what words to expect in a new document. This is a shortcoming that LDA addresses by keeping the pLSA model structure, yet by putting a distribution on the topic mixture weights. Whether one needs a generative model for new documents is application dependent. Note that LDA considers the vocabulary fixed and one could also require a model that accounts for new/unseen word types occurring.

As a side remark: LDA also performs some form of Bayesian regularization and smoothing, which avoids overfitting of the topic mixtures. This is a benefit that is independent (and in addition) to its generative semantics.

Lecture 5

Embeddings

GloVe: Correction & Clarification

GloVe uses a squared error objective, which conceptually brings it close to standard matrix decomposition techniques. However the squared error is computed on the (i) log-scale of counts and (ii) comes with a weighting function $f : \mathbb{Z}_{\geq 0} \to [0; 1]$. Because of the log-scale, entries with zero counts have to be discarded. The objective can be written as

$$\mathcal{H}(\theta, \mathbf{N}) = \sum_{i,j:n_{ij}>0} f(n_{ij})(\log n_{ij} - \log \tilde{p}_\theta(w_i, w_j))^2, \tag{5.1}$$

$$\text{where} \quad \log \tilde{p}_\theta(w_i, w_j) = \langle x_i, y_j \rangle + b_i + c_j\,, \tag{5.2}$$

which models the joint probability of occurrences of a word w_i with a context word w_j.[1] Note that $\mathbf{N} = (n_{ij})$ may or may not be symmetric, dependent on how one defines a context.

[1] In the lecture notes, I had written $\log \tilde{p}_\theta(w_i|w_j)$, which is incorrect.

Lecture 6

Data Clustering and Mixture Models

no questions

Lecture 7

Neural Networks

no questions

Lecture 8

Generative Models

KL Divergence in ELBO

The appearance of the KL divergence term in the ELBO sometimes causes questions on its significance and interpretation. Writing schematically in most simple terns

$$\begin{aligned}
\log p(x) &= \log \int p(x,z)\; dz = \log \int \frac{p(x,z)}{q(z|x)} q(z|x)\; dz && (8.1)\\
&\geq \mathbf{E}_q[\log p(x,z) - \log q(z|x)] && (8.2)\\
&= \mathbf{E}_q[\log p(x|z) + \log p(z) - \log q(z|x)] && (8.3)\\
&= \underbrace{\mathbf{E}_q[\log p(x|z)]}_{\text{predictiveness}} - \underbrace{\mathrm{KL}(q(z|x)||p(z))}_{\text{regularization}} && (8.4)
\end{aligned}$$

First, note that the KL term comes out naturally, when applying Jensen's inequality to the marginal log-likelihood. So it is **not** an explicit design choice. Students sometimes wonder, whether this is sensible as the inference model $q(z|x)$ may be very different from the prior $p(z)$ at times. That is certainly the case, yet no argument against the KL term.

One illuminating way to think about this is in terms of information theory (due to Hinton). Assume we use the following encoding scheme:

1. given x, compute $q(z|x)$
2. sample $z \sim q(z|x)$ and encode it with high numerical accuracy
3. encode x using z and the generative model $p(x|z)$

The naïve coding of z relative to some coding scheme $p(z)$ requires an average coding length of

$$L(q) = \mathbf{E}_q\left[-\log p(z)\right] + \text{const.} \qquad (8.5)$$

Encoding the input requires

$$L(x|q) = \mathbf{E}_q\left[-\log p(x|z)\right] + \text{const.} \tag{8.6}$$

The step from the cross entropy to the (smaller) KL divergence comes from the so-called 'bits-back' argument, which amounts to the differential entropy of q[1]

$$H(q) = \mathbf{E}_q\left[-\log q(z|x)\right] \tag{8.7}$$

The total encoding length is then

$$L = L(x|q) + L(q) - H(q) + \text{const.} \tag{8.8}$$

which is a different view on the negative ELBO.

[1]cf. `http://users.ics.aalto.fi/harri/thesis/valpola_thesis/node30.html`

Lecture 9

Sparse Coding

no questions

Lecture 10

Dictionary Learning

no questions