Probabilistic Transformers

Javier R. Movellan & Prasad Gabbur
Apple

Abstract

We show that Transformers are Maximum Posterior Probability estimators for Mixtures of Gaussian Models. This brings a probabilistic point of view to Transformers and suggests extensions to other probabilistic cases.

This technical note presents a novel probabilistic interpretation of Transformers [1, 2]. This interpretation may help gain a better understanding of the conditions under which standard Transformers are optimal and suggest ways to extend them to more general probabilistic cases.

Database Generative Model. As is commonly done in the literature we ground the presentation of Transformers as database systems that output values to input queries. We model a database as a collection of query, value pairs, parameterized as a joint probability distribution $p(q, v)$. Querying the database corresponds to finding the most probable value given a query. We model the database distribution as a collection of memory units, each of which generate queries and values. The overall distribution of queries and values is the sum over the conditional distributions across units:

$$p(q, v) = \sum_u p(u)\, p(q, v \mid u) \tag{1}$$

Inference. For a given query q we want to compute the most probable value associated with that query, i.e.

$$\hat{v} = \operatorname*{argmax}_v p(v|q) \tag{2}$$

To do so we use an Expectation Maximization (EM) [8] approach: Given the current estimate v_t we look for a new estimate v_{t+1} that increases the standard EM auxiliary function Q. Maximizing Q with respect to v guarantees maximization of $p(v|q)$. Let

$$Q(v_t, v_{t+1}) = \sum_u p(u \mid q, v_t) \log p(u, q, v_{t+1}) \tag{3}$$

Taking the gradient and setting it to zero we get the EM maximization equation

$$\nabla_{v_{t+1}} Q(v_t, v_{t+1}) = \sum_u w_u \nabla_{v_{t+1}} \log p(q, v_{t+1}, u) = 0 \tag{4}$$

where

$$w_u = p(u \mid q, v_t) = \frac{p(u)\, p(q, v_t \mid u)}{\sum_u p(u)\, p(q, v_t \mid u)} \tag{5}$$

and $\nabla_{v_{t+1}} \log p(q, v_{t+1}, u)$ is the Fisher Score for unit u.

Transformers. Here we show that Transformers [1, 2] solve (4) for Gaussian mixture models. Let

$$p(q, v \mid u) = p(q \mid u)\, p(v \mid u) \tag{6}$$

$$p(q \mid u) = \Big(\frac{2\pi}{\alpha}\Big)^{-d/2} e^{-\frac{\alpha}{2}\|q - \xi_u\|^2} \tag{7}$$

$$p(v \mid u) = \Big(\frac{2\pi}{\beta}\Big)^{-m/2} e^{-\frac{\beta}{2}\|v - \mu_u\|^2} \tag{8}$$

$\xi_u \in R^d$, $\mu_u \in R^m$ are the key and expected value vectors for unit u. Thus

$$p(q, v) = \frac{1}{z} \sum_u p(u) e^{-\frac{\alpha}{2}\|q - \xi_u\|^2} e^{-\frac{\beta}{2}\|v - \mu_u\|^2} \tag{9}$$

$$z = \Big(\frac{2\pi}{\alpha}\Big)^{d/2} \Big(\frac{2\pi}{\beta}\Big)^{m/2} \tag{10}$$

and the Fisher score takes the following form

$$\nabla_{v_{t+1}} \log p(q, v_{t+1}, u) = \beta(\mu_u - v_{t+1}) \tag{11}$$

Thus

$$\nabla_{v_{t+1}} Q(v_t, v_{t+1}) = \beta \sum_u w_u(\mu_u - v_{t+1}) \tag{12}$$

and the EM maximization equation becomes as follows:

$$v_{t+1} = \sum_u w_u \, \mu_u \tag{13}$$

$$w_u = \frac{p(u) \, e^{-\frac{\alpha}{2}\|q-\xi_u\|^2} \, e^{-\frac{\beta}{2}\|v_t-\mu_u\|^2}}{\sum_u p(u) \, e^{-\frac{\alpha}{2}\|q-\xi_u\|^2} \, e^{-\frac{\beta}{2}\|v_t-\mu_u\|^2}} \tag{14}$$

To get the Transformer equation we link the priors of each unit to the length of the key and expected value vectors

$$ap(u) = \frac{1}{k} e^{\frac{\alpha}{2}\|\xi_u\|^2} e^{\frac{\beta}{2}\|\mu_u\|^2} \tag{15}$$

$$k = \sum_u e^{\frac{\alpha}{2}\|\xi_u\|^2} e^{\frac{\beta}{2}\|\mu_u\|^2} \tag{16}$$

In this case the optimal inference equation simplifies as follows:

$$\hat{v} = \sum_u w_u \mu_u \tag{17}$$

$$w_u = \frac{e^{\alpha \xi_u' q} \, e^{\beta \mu_u' v_t}}{\sum_u e^{\alpha \xi_u' q} \, e^{\beta \mu_u' v_t}} \tag{18}$$

and as $\beta \to 0$ we obtain the original Transformer equation:

$$v_{t+1} = \sum_u w_u \mu_u \tag{19}$$

$$w_u = \frac{e^{\alpha \xi_u' q}}{\sum_u e^{\alpha \xi_u' q}} \tag{20}$$

Off-line Supervised Parameter Learning. The EM inference equation is differentiable and thus it can be embedded on a deep network with the parameters trained off-line via Stochastic Gradient Descent. Traditionally in Transformers only the key vectors ξ and expected value vectors μ are trained. However, with the right parameterization is also possible to train the precision parameters α, β as well as the priors π.

Inference-time Unsupervised Parameter Learning. In most cases at inference time Transformers receive a batch of queries , e.g. one query per word in a sentence or one query per pixel in an image. In such cases it is possible to adapt the database parameters (e.g, keys, unit priors) to the observed queries in an unsupervised manner. Thus, at inference time first we would update the database parameters to the current queries in an unsupervised manner, followed by the optimal value inference process given the adapted parameters. To prevent parameter overfit to the current batch of queries, we include a prior over the parameter that discourages extreme parameter changes. Let $q_1, q_2, \ldots q_s \in R^d$ be the batch of s

sample queries. Our goal is to find the model parameters that are the most probable given the observed sample of queries

$$\underset{\lambda}{\operatorname{argmax}}\, p(\lambda \mid q_1 \cdots q_s) \tag{21}$$

This can by done by using an iterative EM approach: Given the current parameter estimate λ^t we look for a new estimate λ^{t+1} that maximizes the EM auxiliary function Q:

$$Q(\lambda^t, \lambda^{t+1}) = \sum_{i=1}^{s}\sum_{j=1}^{n} w_j^i \log p(q_i, u_j, \lambda^{t+1}) \tag{22}$$

$$w_j^i = p(u_j \mid q_i, \lambda^t) = \frac{p(u_j \mid \lambda_t)p(q_i \mid u_j, \lambda_t)}{\sum_k p(u_k \mid \lambda^t)p(q_i \mid u_k, \lambda^t)} \tag{23}$$

Taking gradient with respect to the candidate parameter value λ^{t+1} and setting it to zero we get the EM update equation:

$$\nabla_{\lambda^{t+1}} Q(\lambda^t, \lambda^{t+1}) = \sum_{i=1}^{s}\sum_{j=1}^{n} w_j^i \nabla_{\lambda^{t+1}} \log\Big(p(\lambda^{t+1})p(u_j \mid \lambda^{t+1})p(q_i, \mid u_j, \lambda_{t+1})\Big) = 0 \tag{24}$$

Example 1: Unsupervised Adaptation of Keys. Here we adapt $\xi_k \in R^d$, the key vector of unit k. To avoid overfitting to the batch of queries we use a Gaussian prior centered on the initial value of the key vector

$$p(\xi_k^t) \propto e^{-\gamma\frac{1}{2}\|\xi_k^t - \xi_k^0\|^2} \tag{25}$$

In which case the EM update equation (24) takes the following form

$$\nabla_{\xi_k^{t+1}} Q(\xi_k^t, \xi_k^{t+1}) = \sum_{i=1}^{s} w_k^i\Big(\gamma(\xi_k^0 - \xi_k^{t+1}) + \alpha(q_i - \xi_k^{t+1})\Big) = 0 \tag{26}$$

$$\tag{27}$$

resulting in the following solution

$$\xi_k^{t+1} = \frac{\gamma}{\alpha+\gamma}\xi_k^0 + \frac{\alpha}{\alpha+\gamma}\bar{q}_k \tag{28}$$

$$\bar{q}_k = \frac{\sum_i w_k^i q_i}{\sum_i w_k^i} \tag{29}$$

Example 2: Unsupervised Adaptation of Unit Priors. In this case we want to adapt $\pi = [p(u_1), \cdots, p(u_n)]'$. We use a Dirichlet o distribution with parameter $\eta \in R^n$ to discourage extreme changes from the initial unit priors $\pi^0 \in R^n$ we let $\eta \propto \pi^0$. In this case

$$p(q_i, u_j, \pi) = p(\pi)\, \pi_j\, p(q_i \mid u_j) \propto \pi_j^{\eta_j} \pi_j e^{-\frac{\alpha}{2}\|q_i - \xi_k\|^2} \tag{30}$$

and the componet of the Q function that depends on π^{t+1} looks as follows

$$Q(\pi^t, \pi^{t+1}) = \sum_i \sum_k w_k^i(\eta_k + 1)\log(\pi_k^{t+1}) \tag{31}$$

Since the elements of $\pi \in R^n$ are constrained to sum up to one, we add a Lagrangian term to the original Q function. Taking the gradient and setting it to zero

$$\frac{\partial Q(\pi^t, \pi^{t+1}) + \lambda(\sum_i \pi_i^{t+1} - 1)}{\partial \pi_k^{t+1}} = \sum_i w_k^i (\eta_k + 1) \frac{1}{\pi_k^{t+1}} + \lambda = 0 \tag{32}$$

which has the following solution

$$\pi_k^{t+1} = \frac{\bar{w}_k(1 + \eta_k)}{\sum_j \bar{w}_j(1 + \eta_j)} \tag{33}$$

$$\bar{w}_j = \sum_i w_j^i \tag{34}$$

Extensions. The probabilistic interpretation proposed here clarifies the conditions under which standard Transformers are optimal and suggests different versions that may be optimal in more general cases.

Contrary to the standard Transformer equation, the general Gaussian form (13) does not couple the prior probability of a unit to the length of the query vector, it uses a squared difference similarity metric, rather than the inner product metric, and allows for $\beta \neq 0$. There are good reasons to believe that this may benefit the performance of Transformers [5, 6]. As this note shows, the standard Transformer equation is optimal under a Mixture of Gaussians model with a constant diagonal covariance matrix. It may be beneficial to use Gaussian mixtures with non-constant, non-diagonal covariance matrices, use sparse probability models, like mixtures of T-distributions, or to learn sparsity parameters using Generalized Gaussian distributions. Multilayer mixtures of Gaussians in which the value of one layer becomes the query of the next layer can be used the same way as it is done in standard Transformers. These could be trained using the EM algorithm. Mixtures of Gaussians are popular observation models in HMMs. In a similar vein, system dynamics could also be applied to Transformers in deep networks, to enforce temporal consistency of values across time, or across layers. The inference time adaptation equations are differentiable so they can be included as additional layers in a deep network trainable via Stochastic Gradient Descent.

Prior Work. Transformers [2] are an extension of attention mechanisms for deep networks commonly used in NLP problems [1] and more recently in CV problems [4]. The idea that Transformers are related to Mixtures of Gaussians was inspired on recent work showing the relationship between Transformers and Hopfield Networks [3]. While this note was under review [7] published a probabilistic interpretation of Transformers, similar to ours. However in [7] the objective is to maximize the marginal probability of queries, which as the paper mentions, it is not a natural way to do probabilistic inference. In our formulation the goal is to find the most probable value for a given query, and we show how the EM solution to this problem becomes the standard Transformer equation under special cases. The adaptation of parameters at inference time was inspired by the Double Normalized Attention (DNA) algorithm in [7]. DNA can be seen as a special case of on-line adaptation of the key parameters when using uninformative priors ($\gamma \to 0$), and unit proportions linked to the length of the key parameters. However our interpretation of the role of DNA is quite different than in [7]. In [7] DNA is justified by optimizing the likelihood of the keys with respect to the queries. Here DNA surfaces as part of unsupervised MAP estimation of the key parameters. We hope for this note to contribute to the ongoing conversation to gain a better theoretical understanding of Transformers.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2014) Neural machine Translation by Jointly learning to Align and Translate. CoRR, abs/1409.0473.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin (2017) Attention is All You Need, arXiv:1706.03762

[3] Lukas Gruber, Markus Holzleitner, Milena Pavlovic ,Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp Gunter Klambauer, Johannes Brandstetter, Sepp Hochreiter (2020) Hopfield Networks is All You Need, arXiv:2008.02217.

[4] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, Liang-Chieh Chen (2020) Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation, arXiv:2003.07853.

[5] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, Han Hu (2020) Disentangled Non-Local Neural Networks, arXiv:2006.06668.

[6] Hyunjik Kim, George Papamakarios, Andriy Mnih (2020) The Lipschitz Constant of Self-Attention, arXiv:2006.04710.

[7] Nan Ding, Xinjie Fan, Zhenzhong Lan, Dale Schuurmans, Radu Soricut (2020) Attention that does not Explain Away, arXiv:2009.14308

[8] Arthur Dempster, Nan Laird, Donald Rubin (1977) Maximum Likelihood from Incomplete Data via the EM Algorithm, JRSS, Vol. 39, No. 1, pp. 1-38