SCIENCES SORBONNE UNIVERSITÉ

Statistical Learning Project

Bayesian Attention Modules

Lise Le Boudec
Yunfei ZHAO

January 12, 2022

In this project we aim at developing a course about attention mecanisms and the latest improvements related. We will focus on Bayesian theory and Attention mechanism which is introduced in **Bayesian Attention Modules**. The paper is accepted in NeurIPS 2020 and we are going to introduce and study essential elements in this paper and related papers.

Contents

List of Figures

Introduction

In this course, we are going to make an introduction of attention mechanism which is firstly mentioned by Bahdanau et al. (2014) [4], and which takes an important role in artificial intelligence (in tasks such as nature language language processing, computer vision and reinforcement learning). Further more, we are going to explain how stochastic theory can be introduced into this mechanism to make it has a better generalisation and more robust to noise.

Deep neural network models based on attention mechanism have achieve state-of-art results in many domains with a better interpretablility. The paper **Bayesian Attention Modules** [5] which we are going to study is a bridge connecting attention models with stochastic models and the Bayesian framework. It incurs an improvement in graph node classification, visual question, answering, image captioning, machine translation, and language understanding.

Before starting this course, we need to have a basic understanding of deep neural network and notions in Bayesian inference theory. After this course, you will have a good understanding of **Transformers-based Encoder-Decoder Models**, how **Bayesian Framework** improves the performance of these models as well as some state-of-art in AI research field using these elements. This course will include intuitive explications and theoretical demonstrations.

This course is going to focus firstly on attention mechanisms to introduce the principle and interest of this field. Then, an introduction to Bayesian theory and more precisely the related variational inference is proposed. In the last part, we study how to introduce the Bayesian theory into the Attention modules.

Chapter 1

Attention in neural network

In the context of neural networks, attention is a technique that mimics cognitive attention. It enhances the important parts of the input data and fades out the rest – the thought being that the network should devote more computing power on a small but important part of the data. Which part of the data is more important than others depends on the context and is learned through training data by gradient descent.

The article we are studying is based on the Transformer which is a specific model structure of attention mechanism. Transformer has its own attention mechanism called **Scaled Dot-Product Attention** which is introduce in the following section.

1.1 Background on Attention Mechanism

Self-attention, is a mechanism first used for nature language processing, such as language translation, text content summary, etc. Self-attention, sometimes called intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence, the sequence can be a phrase in NPL task (Like in the Google Brain's original paper **Attention is all you need** [6]). This mechanism have already become an integral part of compelling sequence modeling and transduction models in various tasks. However such attention mechanism are used in conjunction with a recurrent network (RNN) or a convolutonal network (CNN). Given a sequence of words, if we see the word "eating", then we will take more attention to the following name of food.

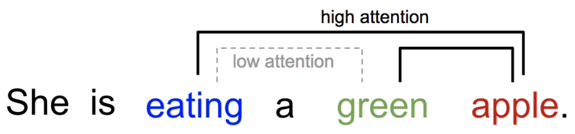


Figure 1.1: An example on attention when human read a sentence

1.1.1 Seq2Seq Model

One example of application of the attention mechanism is the famous "Sequence to Sequence Learning with Neural Networks" [7] model. Broadly speaking, it aims to transform an input sequence (source) to a new one (target) and both sequences can be of arbitrary lengths. Examples of transformation tasks include machine translation between multiple languages in either text or audio, or tasks like adding video caption. The main structure Seq2seq model uses is the Encoder-Decoder structure.

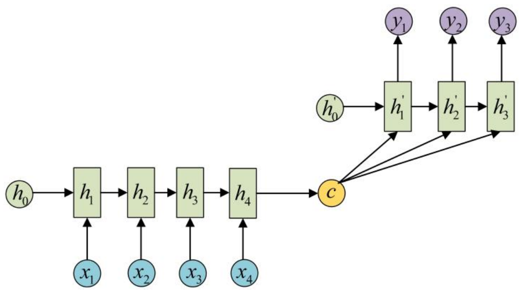


Figure 1.2: RNN Seq2Seq model Encoder Decoder structure.

Definition 1.1: Encoder

Encoder processes the input sequence and compresses the information into a context vector. This vector is expected to be the encoded information or meaning of the input sequence.

Definition 1.2: Decoder

Decoder is initialized with the context vector to emit the transformed output. The early work only used the last state of the encoder network as the decoder initial state.

In this case, both the encoder and decoder are recurrent neural networks (they can use LSTM or GRU units for example).

Disadvantage

However, there are some disadvantages of RNN for this task. Fixed-length context vector design are not able to remember long sentences. As an example, a RNN has forgotten the first parts of a given sentence once it has completely processed the whole sequence. These modules are also hard to parallel, as the layers are run in sequence.

1.1.2 CNN Model

People try to use CNN to solve the parallel issue caused by RNN. But as show in figure 1.3, the receptive field [1] of one CNN layer is very limited and it is very hard to capture the global information without using deep networks.

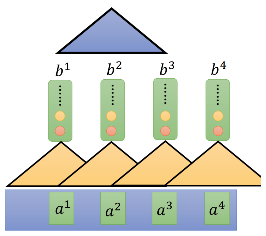
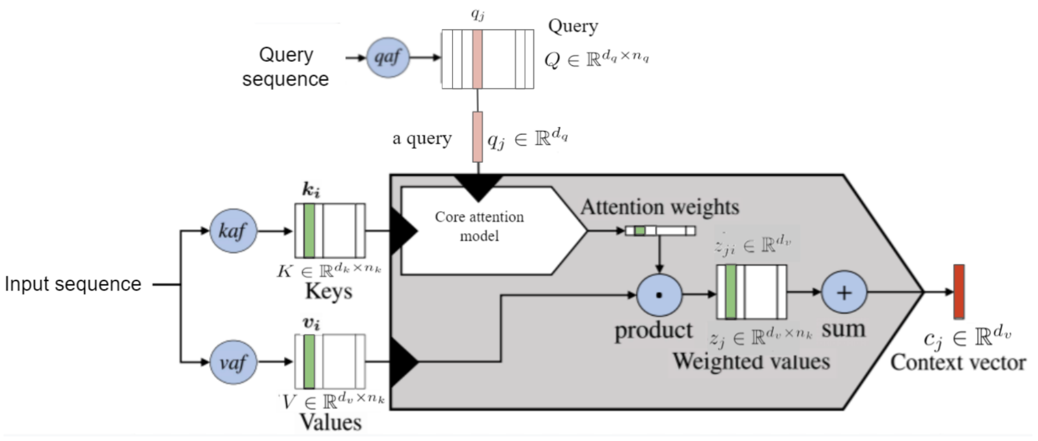


Figure 1.3: Using CNN layer to replace RNN to get attention[1]

[1]An explanation of receptive field https://distill.pub/2019/computing-receptive-fields/

1.2 Transformer

The Transformer is a model proposed in **Attention is all you need**[6]. It's entirely based on attention. Nowadays, almost all tasks that use a RNN to perform attention mechanism can be replaced by a Transformer. The transformer is the new stat-of-art structure in 2020 and it outperforms the Google's Neural Machine Translation System "GNMT" [8] in specific tasks. Its biggest benefit, comes from how The Transformer lends itself to parallelization. It is in fact Google Cloud's recommendation to use The Transformer as a reference model to use their Cloud TPU offering. The general attention model is shown below (it's proposed in a great review of attention mechanism **Attention in Natural Language Processing** [9]).



Query sequence
qaf
q_j
Query
$Q \in \mathbb{R}^{d_q \times n_q}$
a query
$q_j \in \mathbb{R}^{d_q}$
k_i
kaf
Input sequence
$K \in \mathbb{R}^{d_k \times n_k}$
Keys
v_i
vaf
$V \in \mathbb{R}^{d_v \times n_k}$
Values
Core attention model
Attention weights
$z_{ji} \in \mathbb{R}^{d_v}$
product
$z_j \in \mathbb{R}^{d_v \times n_k}$
Weighted values
sum
$c_j \in \mathbb{R}^{d_v}$
Context vector

Figure 1.4: Schema of general attention model

1.2.1 Basic Elements of Attention

Definition 1.3: Three meta elements of Attention

$$\textit{Key Matrix} \quad \mathbf{K} \in \mathbb{R}^{n \times d_k}, \quad \textit{Query Matrix} \quad \mathbf{Q} \in \mathbb{R}^{m \times d_k}, \quad \textit{Value matrix} \quad \mathbf{V} \in \mathbb{R}^{n \times d_v}$$

Definition 1.4: Energy scores

$$\mathbf{E} = f(\mathbf{K}, \mathbf{Q}) \in \mathbb{R}^{m \times n}$$

Where f is called a **Compatibility function**.

Table 1.1: Commonly used compatibility functions

Name	Equation
similarity	$f(k_i, q_j) = sim(k_i, q_j)$
multiplicative or dot	$f(k_i, q_j) = q_j^T k_i$
scaled dot	$f(k_i, q_j)=\frac{q_j^T k_i}{\sqrt{d_k}}$
general bilinear	$f(k_i, q_j) = q_j^T W k_i$
activated general	$f(k_i, q_j) = act(q_j^T W k_i + b)$

Definition 1.5: Attention Weights

$$\mathbf{W} = g(\mathbf{E}) \in \mathbb{R}^{m \times n}$$

Where g is a distribution function, we often use a Softmax function so we have

$$\mathbf{W} = softmax(\mathbf{E}) \in \mathbb{R}^{m \times n}$$

In figure 1.5, the elements of matrix **E** are represented as α and the elements of **W** are represented as $\hat{\alpha}$.

Definition 1.6: Context Vector

The output feature is represented as

$$\mathbf{C} = \mathbf{WV} \in \mathbb{R}^{m \times d_v}$$

Where

$$C_j = \sum_i Z_{ij} = W$$

A generalized attention model works with a set of key-value pairs $(K, V) = \{(k_1, v1), (k_2, v2), \ldots, (\{k_{n_k}, v_{n_k})\}$. For a query $q_j \in Q$, the attention model can be written as:

$$\begin{aligned} Attention(q_j, K, V) &= c_j \\ &= \sum_{i=1}^{n_k} a_{ji} v_i \\ &= \sum_{k_i \in \Omega} g(f(q_j, k_i)) v_i \end{aligned} \tag{1.1}$$

Definition 1.7: Attention Field

The set Ω is called **attention field**. It contains all the input keys observed by the query state q_j to calculate the corresponding attention weights. There are three kinds of attention models, depending on the keys we expose to a given query.

- When $\Omega = K$, namely $|\Omega| = n_k$, we have **global attention**. It takes all keys into account for a given query q_j.
- When $\Omega \subset K$, namely $|\Omega| < n_k$ **local attention**, take a part of keys into account for a given query q_j.
- When query, key and value comes from same hidden states that is to say, they all come from a transformation of given variable **X**, the attention model is called **self attention**.
- When key and value comes from come from a transformation of given variable **X** and query comes from different transformation of given hidden states variable **Y** , the attention model is called **cross attention**.

There is a table to resume all notations in attention mode which can be found in 3.1

1.2.2 Hard Attention VS Soft Attention

There are generally two types of attention : hard attention and soft attention. This course, and the article [9], use soft attention as a starting point.

soft attention : The attention model can be described as a mapping with a probabilistic interpretation. Indeed, we use of a softmax normalization on energy scores to obtain the attention weights, so these weights can be seen as a distribution. If the attention weights are deterministically selected by a compatibility function, the model is called deterministic (soft) attention. It uses all values $\mathbf{V}$ with their attention weights $\mathbf{W}$ to get the attention.

hard attention : It uses probability distribution over weights to hardly sample a single input as context vector, in particular following a reinforcement learning strategy. Sometimes, the attention weights encode a probability distribution. In this case, the stochastic attention model differs from the general only by the value $\mathbf{V}$ selected (we pick the $\mathbf{V}$ which is associated to the maximum attention weight). This kind of method like argmax is not differentiable, and nudging attention weights a little bit will not change the index we selected (in this case, more complex techniques are employed).

1.2.3 Scaled Dot-Product Attention

Attention model aims at generating a mask (attention weight) on the encoder hidden states, to force the decoder to pay attention to the important elements, as shown in Figure 1.3. More generally, attention model can be considered as a mapping from a sequence of vector $\{h_1, h_2, \ldots, h_T\}$ (also called **Keys**) to a distribution (attention weight) according to another sequence of vector $\{s_1, s_2, \ldots, s_{T'}\}$ (also called **Queries**).

The core part of attention mechanism maps a matrix of Keys ($K \in \mathbb{R}^{d_k \times n_k}$) to attention weights , given a query vector q_j with size d_q. The scalar elements (α_{ji}) represent the importance of the corresponding k_i to q_j. K encodes the data features on which attention is computed. For instance, K may be a word or a character embeddings of a document, or the internal states of a recurrent architecture. In some cases, K could include multiple features or representations of the same object (e.g., both one-hot encoding and embedding of a word) or even — if the task calls for it — representations of entire documents.

Another input element q_j (a query), is used as a reference when computing the attention distribution. In that case, the attention mechanism will give emphasis to the input relevant elements for the task, according to q_j. If no query is defined, attention will give emphasis to the elements inherently relevant to the task at hand. In RNN, for instance, q_j is a single element, namely, the RNN hidden state s_{t-1}. In other architectures, q_j may represent different entities: embeddings of actual textual query, contextual information background knowledge... From the keys K and query q_j, a vector α_j of n_k energy scores denoted α_{ji} is computed through a compatibility function f:

$$\alpha_{ji} = f(k_i, q_j) \tag{1.2}$$

The compatibility function defines how keys and queries are matched or combined. Commonly used compatibility functions are shown in table 1.1. Energy scores are then transformed into attention weights α_j using a distribution function g, as shown in equation (1.3). Such weights are the outcome of the core attention mechanism. The commonest distribution function is the softmax function, which normalizes all the scores to a probability distribution. Weights represent the relevance of each element for the given task, with respect to q_j and $k_i \in K$.

$$\hat{\alpha}_{ji} = g(\alpha_{ji}) \tag{1.3}$$

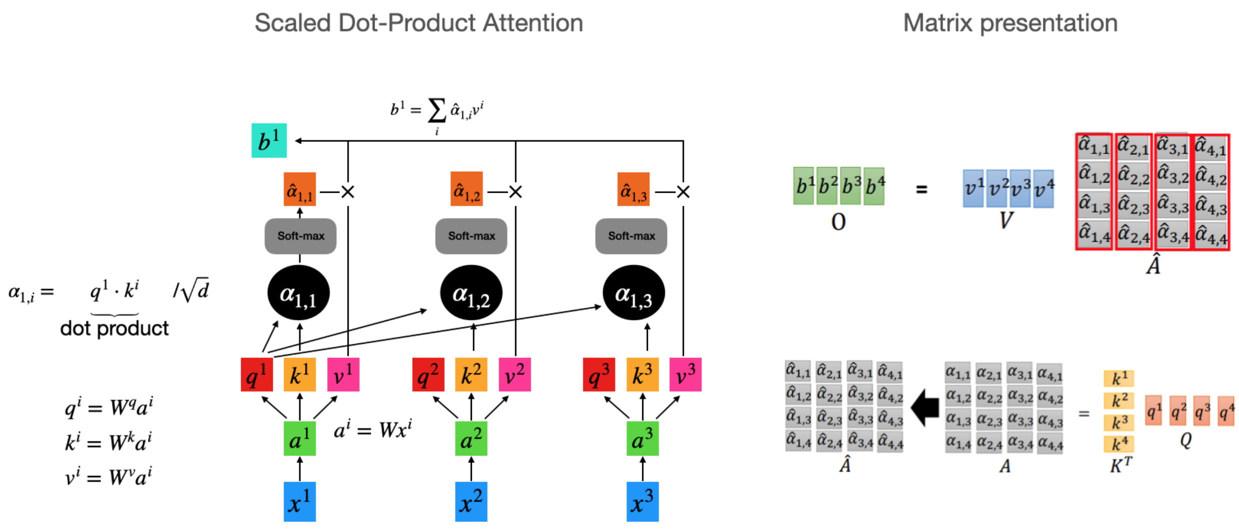


Figure 1.5: Schema of Scaled Dot-Product Attention, created by Yunfei ZHAO based on M. Hung-yi Lee's course slide [2]

1.2.4 Multi-head Attention

The authors of the paper about Transformers found that it is beneficial to linearly project the queries, keys, and values h times with different and learned linear projections into some d_k, d_k and d_v dimensions, respectively. It generates h d_v-dimensional output values in parallel and these are concatenated and once again projected, resulting in the final values.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^o$$

where W^o is a linear transformation

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

1.2.5 Structure of Transformer: Encoder and Decoder

As the paper we studied include many experiments in visual question answering, image captioning, we take an classical paper which use Transformer to do object detection as an example, to explain the structure of Transformer. The name of this paper is **End-to-End Object Detection with Transformers (DETR)**[10]

Firstly, an input RGB image tensor with size $[3, H_0, W_0]$ (H_0 for height, W_0 for width) is fed to a CNN fitted to extract feature tensor with size $[C, H, W]$ (C for channels, H for height, W for width). A 1×1 CNN is used to reduce the channels from C to d, and thus the feature tensor becomes $[d, H, W]$. A transformer requires a sequence as input, so that the tensor $[d, H, W]$ is flattened into $[d, H \times W]$ before being sent to encoder. The encoder in transformer is responsible for extracting and encoding input features based on multi-head self attention modules. In DETR, the encoder can be considered as a feature refiner to add global information to the previous CNN features that contains local information. The output of encoder is a sequence tensor $[d_1, H \times W]$ including $H \times W$ elements with d_1 dimension for each of them, as shown in figure 1.6.

The decoder has two inputs: the output features from the encoder, and the object queries. Object queries are a set of random vectors and its cardinality coresponds to the maximum number of objects the model can find in an image (One query correspond to an object). Object queries can be considered as the initial random guess of bounding boxes which can be learned directly and automatically. In DETR, the number of object queries is set to 100, indicating that the model has to learn and find 100 bounding boxes and has to predict class label for each of them. These 100 boxes are all kept in the final decision so that no NMS (Non Maximum Suppression) process is required. For each object query in input, the decoder will learn a corresponding feature tensor which then connects with two separate FFN (Feed Forward Network) heads: one for class prediction, the other for bounding box prediction. DETR uses a loss based on bipartite matching between a set of predicted pairs $< class, box >$ and the ground truth, which forces each pair to be assigned to only one object without duplication in the ground truth. To satisfy this constraint, a 'no object' class is introduced for those pairs that do not match any object.

In DETR, the decoder takes a more important role than the encoder. Details are shown in Figure 1.7. There are two types of attention modules in decoder: **multi-head self attention** and **multi-head cross attention**.

Multi-head self attention takes the object queries (the first decoder layer), and output from the previous decoder layer as input and revels the importance of each position in the sequence, so that the multi-head cross attention can focus on more important positions.

Multi-head cross attention uses the output from self-attention module as query Q to find important features from the output of encoder. Output features from each decoder layer can connect with two FFN heads to predict classes and boxes, so that the loss can be calculated for each decoder layer.

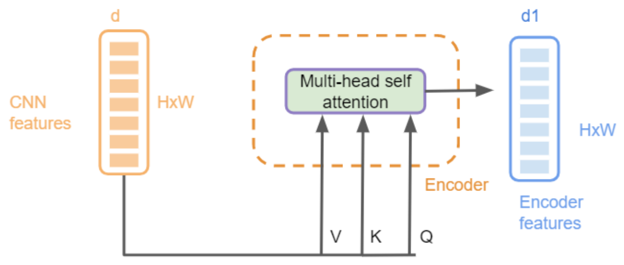


Figure 1.6: Encoder in DETR framework.

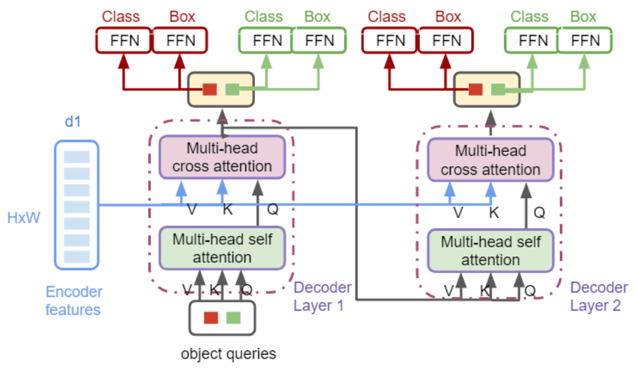


Figure 1.7: Decoder in DETR framework.

Chapter 2

Variational Bayesian method

We consider a supervised learning problem with some observed variables $\mathcal{X} := \{x_i\}_{i=1}^N$, where N is the size of the dataset, and some hidden variables $\mathcal{Z} := \{z_i\}_{i=1}^N$.

2.1 Set-up

Definition 2.1: Latent Variables

A latent variable is a variable that cannot be observed. The presence of latent variables, however, can be detected by their effects on variables that are observable.

In Bayesian models, the latent variables help govern the distribution of the data. A Bayesian model draws the latent variables from a **prior** density $p(z)$ and then relates them to the observations through the **likelihood** $p(x|z)$. Inference in a Bayesian model amounts to conditioning on data and computing the **posterior** $p(z|x)$. To complete the vocabulary, we call **evidence** the distribution $p(x)$ of the observations.

Though, we consider that the Z_i's are some latent variable related to the X_i's, on which we would like to compute the distribution. We are interested in finding the posterior distribution of our unobserved variables $p(z|x)$. Unfortunately, this posterior distribution is often very difficult to compute [11], [12].

$$\begin{aligned} p(z|x) &= \frac{P(z,x)}{P(x)} \\ &= \frac{P(z,x)}{\int P(x,z)dz} \end{aligned}$$

The normalization term is the problem, because this integral can be unavailable in closed form or requires exponential time to be computed. That's why we approximate it with a simpler distribution denoted $q(z)$, ie we claim that there exists $q(z)$ such that $p(z|x) \approx q(z)$ which we suppose to belong to a given family of known distributions. We call this the **Variational distribution**.

2.2 KL Divergence

To quantify the quality of our approximation, we need a criteria.

Definition 2.2: Divergence [13]

Let p and q be two probability distributions over a space Ω. We suppose that p is absolutely continuous with respect to q. Then, for a convex function f such that $f(1) = 0$, the f-divergence of p from q is defined as :

$$D_f(p||q) := \int_{\Omega} f\left(\frac{dp}{dq}\right) dq$$

Property 2.1: Property of divergence

Let p and q be two probability distributions and let $D_f(p||q)$ be the divergence associated. We have the following propertied :

- $D_f(p||q) \geq 0$
- $D_f(p||q) = 0$ iff $p = q$ almost everywhere.

Proof. Application of the Jensen inequality □

Several divergence measures exists [13], [14], but the most commonly used divergence is the Kullback-Leibler (KL) divergence, which is the one we are going to use.

Definition 2.3: Kullback-Leibler divergence

Let p and q be two distribution. The Kullback-Leibler divergence is defined as :

$$D_{KL}(p||q) = \mathbb{E}_{p(x)}\left[\log \frac{p(x)}{q(x)}\right]$$

It's corresponding function f in 2.2 is $f(t) = tlog(t)$.

The **Variational Inference** aims at finding the distribution $q(z)$ that minimizes the KL-divergence. In the best case, we would like it to be 0. This is rarely possible because of the approximation done on the true posterior distribution (see an illustration on Figure 2.1). Indeed, if $D_{KL} = 0$, the variational distribution matches the posterior thanks to property 2.1.

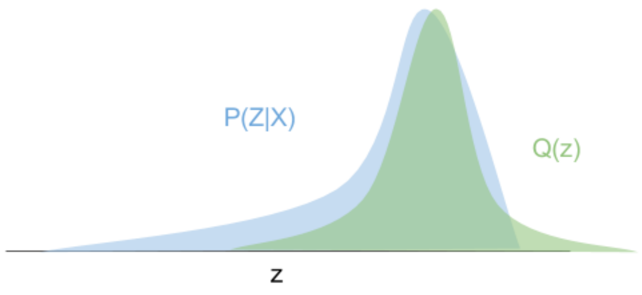


Figure 2.1: Approximation of probability distributions [3].

Back to our distribution estimation, we now want to solve the following problem :

$$\min_q D_{KL}\left(q(z)||p(z|x)\right) = \min_q \mathbb{E}_q\left[\log \frac{q(z)}{p(z|x)}\right] \tag{2.1}$$

2.3 ELBO

Once again, we face a problem to estimate directly the KL-divergence because of $p(z|x)$ which the unknown distribution that we want to estimate. Let's work on the KL-divergence expression [12] :

$$
\begin{aligned}
D_{KL}\left(q(z)||p(z|x)\right) &= \mathbb{E}_q\left[\log\left(\frac{q(z)}{p(z|x)}\right)\right] \\
&= \mathbb{E}_q\left[\log(q(z))\right] - \mathbb{E}_q\left[\log(p(z|x))\right] \\
&= \mathbb{E}_q\left[\log(q(z))\right] - \mathbb{E}_q\left[\log(p(z,x))\right] + \mathbb{E}_q\left[\log(p(x))\right]
\end{aligned}
$$

Setting $ELBO := \mathbb{E}_q\left[p(z,x)\right] - \mathbb{E}_q\left[\log q(z)\right]$, and noticing that $\mathbb{E}_q\left[p(x)\right]$ is constant with respect to q, we show that minimizing the KL-divergence is equivalent to maximizing the ELBO .

Remark : Interpretation of ELBO
Rewritting ELBO, we have :

$$
\begin{aligned}
ELBO &= \mathbb{E}_q\left[\log(p(z,x))\right] - \mathbb{E}_q\left[\log q(z)\right] \\
&= \mathbb{E}_q\left[\log(p(x|z))\right] + \mathbb{E}_q\left[\log(p(z))\right] - \mathbb{E}_q\left[\log q(z)\right] \\
&= \mathbb{E}_q\left[\log(p(x|z))\right] - D_{KL}(q(z)||p(z)) \qquad (2.2)
\end{aligned}
$$

In this expression, the first term is an expected likelihood whereas the second term is the KL-divergence between the variational density and the prior. Though, the ELBO shows the trade-off between finding the latent variables that explain data and finding a distribution that is close to the prior.
Moreover, we have :

$$
\begin{aligned}
ELBO + D_{KL}\left(q(z)||p(z|x)\right) &= \mathbb{E}_q\left[\log(p(x))\right] \\
ELBO &\leq \mathbb{E}_q\left[\log(p(x))\right] \qquad \text{because } D_{KL} \geq 0
\end{aligned}
$$

Though, the ELBO lower-bounds the log-evidence for any q, which explains its name (**E**vidence **L**ower **BO**und).

Chapter 3

Bayesian model in Attention Module

In this last chapter, we are going to see the last improvements in term of Stochastic Attention by developing the construction of a "Bayesian Attention Modules" taken from the article of the same name [5].

3.1 The model

We consider a supervised learning problem with some training data $\mathcal{D} := \{x_i, y_i\}_{i=1}^N$, where N is the size of the dataset. We aim at modelling the conditional probability $p_\theta(y_i|x_i)$ using a neural network parametrized by θ. This NN is supposed to have attention modules (see Chapter 1 for more details on these modules). Moreover, we suppose that the attention weights, denoted as $\{W^{l,h}\}_{l=1:L,h=1:H}$, are stochastic weights sampled from a distribution q_ϕ. These weights are not observed, and though can be seen as latent variables that depends on our data. More precisely, they depends on the queries (Q) and keys (K) of the attention module.

A few notations on the Bayesian model used :

- we denote $p(W)$ the prior distribution of the W's.
- $p(y|x, W)$ is the likelihood.
- $p(W|x, y)$ is the posterior distribution (that we want to estimate).
- $q(W)$ is the variational distribution used to approximates the posterior.

As explained in chapter 2, we are solving the following problem :

$$\min_q D_{KL}(q(W)||p(W|x, y)) \tag{3.1}$$

Which is equivalent to maximize the ELBO related to our problem (using expression 2.2):

$$\max_q \mathbb{E}_q [p(W, x, y)] - \mathbb{E}_q [\log q(W)] = \max_q \mathbb{E}_q [\log(p(y|x, W))] - D_{KL} (q(W)||p(W)) \tag{3.2}$$

These problems are still very difficult to solve, and we need to add hypothesis. We suppose that our distribution belong to well-known distribution families so that the problem reduces now at finding the parameter of those distributions. We will call ϕ the parameters of $q(W)$ and η the parameters of $p(W)$.

3.2 Backpropagation through attention modules

Let's first recall the expression of the ELBO :

$$ELBO = \max_{q_\phi} \mathbb{E}_{q_\phi}\left[\log(p_\theta(y|x,W))\right] - D_{KL}\left(q_\phi(W)||p_\eta(W)\right) \tag{3.3}$$

The first therm is the likelihood, that we can easily compute. However, we need to work a little on the second part of the expression to obtain an analytic form and its gradient. Indeed, with some good hypothesis on our distribution, we can compute the KL-divergence. Furthermore, we recall that these distribution are used in a Neural Networks, and so, it is desirable to have a method that allows backpropagation through the network.

Among the existing distribution, two of them are studied in [5] and provides explicit expressions of the quantities involved : the Weibull distribution and the Lognormal distribution. They are both well-known distributions that we can easily optimize to solve our problem as we will see in the next sections 3.2.1 and 3.2.2.

3.2.1 Log-Normal distribution

Definition 3.1: Log-Normal

Let S be a random variable following the Log-Normal distribution. Then $S \sim Lognormal(\mu, \sigma^2)$ admits the following probability density function (pdf), for all $s \in \mathbb{R}_+$:

$$p(s|\mu,\sigma) = \frac{1}{s\sigma\sqrt{2\pi}} \exp - \frac{(\log(s)-\mu)^2}{2\sigma^2}$$

Moreover, if $S \sim Log-Normal(\mu, \sigma^2)$, then $log(S) \sim \mathcal{N}(\mu, \sigma^2)$.

Property 3.1: KL divergence with log-Normale distribution

Let X, Y be two random variables following respectively two Log-Normal distributions, ie $X \sim Log-Normal(\mu_1, \sigma_1^2)$ and $Y \sim Log-Normal(\mu_2, \sigma_2^2)$. Then the KL divergence has an analytic expression which is :

$$D_{KL}(Weibull(k,\lambda)||Gamma(\alpha,\beta)) = \log(\frac{\sigma_2}{\sigma_1}) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - 0.5$$

Proof. Let X, Y be two random variables following respectively two Log-Normal distributions, ie $X \sim Log-Normal(\mu_1, \sigma_1^2)$ (p(x)) and $Y \sim Log-Normal(\mu_2, \sigma_2^2)$ (q(y)). Then with a direct computation of the KL divergence we have :

$$D_{KL}(p(x)|q(x)) = \mathbb{E}_p\left[\log(\frac{p(x)}{q(x)})\right] = \mathbb{E}_p\left[\log(p(x))\right] - \mathbb{E}_p\left[\log(q(x))\right]$$

Noticing that if $z \sim Log-Normal(\mu, \sigma^2)$, then $log(z) \sim \mathcal{N}(\mu, \sigma^2)$.

$$\begin{aligned} D_{KL}(p(x)|q(x)) &= \mathbb{E}_p\left[\log(p(x))\right] - \mathbb{E}_p\left[\log(q(x))\right] \\ &= \mathbb{E}_p\left[-\log(x) - \log(\sigma_1\sqrt{2\pi}) - \frac{(\log(x)-\mu_1)^2}{2\sigma_1^2}\right] \\ &- \mathbb{E}_p\left[-\log(x) - \log(\sigma_2\sqrt{2\pi}) - \frac{(\log(x)-\mu_2)^2}{2\sigma_2^2}\right] \end{aligned}$$

Simply developing the two terms (by using the linearity of $\mathbb{E}[.]$) and the expression of the pdf of a Log-Normal distribution in Definition 3.1, we obtain :

$$\begin{aligned} D_{KL}(p(x)|q(x)) &= -\mu_1 - \log(\sigma_1\sqrt{2\pi}) - 0.5 + \log(\sigma_2\sqrt{2\pi}) + \mu_1 + \frac{1}{2\sigma_2^2}(\sigma_1^2 + (\mu_1-\mu_2)^2) \\ &= \log(\frac{\sigma_2}{\sigma_1}) + \frac{1}{2\sigma_2^2}(\sigma_1^2 + (\mu_1-\mu_2)^2) - 0.5 \end{aligned}$$

Which is the desired expression. □

3.2.2 Weibull distribution

Definition 3.2: Weibull

Let S be a random variable following the Weilbull distribution. Then $S \sim Weibull(k, \lambda)$ admits the following probability density function (pdf), for all $s \in \mathbb{R}_+$:

$$p(s|k,\lambda) = \frac{k}{\lambda^k} s^{k-1} e^{-(\frac{s}{\lambda})^k}$$

Definition 3.3: Gamma

Let S be a random variable following the Gamma distribution. Then $S \sim Gamma(\alpha, \beta)$ admits the following probability density function (pdf), for all $s \in \mathbb{R}_+$:

$$p(s|\alpha,\beta) = \frac{1}{\Gamma(\alpha)} \beta^\alpha s^{\alpha-1} e^{-\beta s}$$

where $\Gamma(z) = \int_0^{+\infty} t^{z-1} e^{-t} dt$ is the Euler function.

Property 3.2: KL divergence with Weibull and Gamma distributions

Let X, Y be two random variables following respectively the Weibull and Gamma distributions, ie $X \sim Weibull(k, \lambda)$ and $Y \sim Gamma(\alpha, \beta)$. Then the KL divergence has an analytic expression which is :

$$\begin{aligned} &D_{KL}(Weibull(k,\lambda)||Gamma(\alpha,\beta)) \\ &\quad = \frac{\gamma\alpha}{k} - \alpha\log(\lambda) + \log(k) + \beta\lambda\Gamma(1+\frac{1}{k}) - \gamma - 1 - \alpha\log(\beta) + \log(\Gamma(\alpha)) \end{aligned}$$

3.2.3 Analytic expression of the KL-divergence in BAM

In this section, we suppose that the attention weight are now sampled from either a Weibull distribution 3.2.2 or a Log-Normal distribution 3.2.1. We then apply a normalization step (a

SoftMax for example), in order to insure that the weight remain in $[0;1]$. Note that after the normalization step, we once again don't have an analytic expression of the weights. In the following, we will denote S the weights directly sampled from the distribution, and W the weights after normalization step.

Remark

If the weights S's aren't sampled from a distribution, but taken from their expectation, i.e. they are deterministic, the problem is equivalent to the classic deterministic soft attention [4]. This can be achieved by letting the hyper-parameter k in the Weibull distribution go to infinity, or σ in the log-normal distribution go to zero. Then, the variance tends to zero and the distribution becomes a point centered on its mean.

Last step before obtaining the expression of our objective function ELBO , is to compute the expression of the regularization term, which we will compute for the S's variable. Moreover, there are some dependencies between attention weights from different layers 1 i.e., we have :

$$q_\phi(S) = \prod_{l=1}^{L} q_\phi(S_l|S_{1:l-1})$$

where L in the number of attention layers. The hypothesis from the previous sections gives us an analytic expression for $q_\phi(S_l|S_{1:l-1})$. Fortunately, thanks to

Property 3.3: Semi-analytic expression for the KL

The KL-divergence from the prior to variational distribution is semi-analytic :

$$D_{KL}(q_\phi(S)||p_\eta(S)) = \sum_{l=1}^{L} \mathbb{E}_{q_\phi(S_{1:l-1})} D_{KL}(q_\phi(S_{1:l-1})||p_\eta(S_{1:l-1})) \tag{3.4}$$

Proof.

$$\begin{aligned}
D_{KL}(q_\phi(S)||p_\eta(S)) &= \mathbb{E}_{q_\phi(S)}\left[\sum_{l=1}^{L} \log(q_\phi(S_{1:l-1})) - \log(p_\eta(S_{1:l-1}))\right] \\
&= \sum_{l=1}^{L} \mathbb{E}_{q_\phi(S)}\left[\log(q_\phi(S_{1:l-1})) - \log(p_\eta(S_{1:l-1}))\right] \text{ linearity of } \mathbb{E}[.] \\
&= \sum_{l=1}^{L} \mathbb{E}_{q_\phi(S_{1:l-1})}\mathbb{E}_{q_\phi(S_l|S_{1:l-1})}\left[\log(q_\phi(S_{1:l-1})) - \log(p_\eta(S_{1:l-1}))\right] \text{ by conditionning} \\
&= \sum_{l=1}^{L} \mathbb{E}_{q_\phi(S_{1:l-1})}\left[D_{KL}(q_\phi(S_{1:l-1})||p_\eta(S_{1:l-1}))\right]
\end{aligned}$$

□

Finally, we obtain the following expression for our objective function (combining 3.3 and 3.4):

$$\max_{q_\phi} \mathcal{L}(x, y, S) = \max_{q_\phi} \mathbb{E}_{q_\phi} \left[\log(p_\theta(y|x, S))\right] - \lambda \sum_{l=1}^{L} \mathbb{E}_{q_\phi(S_{1:l-1})} \left[D_{KL}(q_\phi(S_{1:l-1})||p_\eta(S_{1:l-1}))\right] \quad (3.5)$$

Where λ can be seen as a reguralization parameter to balance the likelihood of the data from the metric between the two distributions. As we have seen in the previous section, the expression of D_{KL} is analytic and its gradients can be computed thanks to traditionnal NN techniques, as well as the likelihood. In [5], they add a reparametrization step to sample the data, and use a Monte-Carlo estimation to estimate the gradient (see more details on gradient estimation with Monte Carlo method in [15]). All these computation leads to pseudo-code for Bayesian Attention Modules presented in Annex.

Conclusion

This course presented a very recent method to introduce a Bayesian estimation method in attention modules, which can be seen as a generalization of classical deterministic attention modules. To do so, we presented attention mechanisms, and the basics of variational Bayesian theory.

In the reference paper, many applications show the interest of a such method. Indeed, in the example presented in [5], BAM (**B**ayesian **A**ttention **M**odules) show interesting increases of the performances in tasks such as image captioning or pretrained language models with a very small increase of the computational cost.

Other method exists to introduce randomness in attention modules such as [16], [17] or [18]. This article has lead to new publication using bayesian transformers ([19] or [20] for example). More generaly, Attention and transformers have opened many possibilities in term of performance in various fields such as image processing or NLP for example.

Annex

Recap of the notation in Attention Models

Table 3.1: Notations in attention model

Symbol	Name	Definition
K	Keys	Matrix of n_k vectors (k_i) of size d_k, whereupon attention weights are computed. $K \in \mathbb{R}^{d_k \times n_k}$.
V	Values	Matrix of n_k vectors (v_i) of size d_k, whereupon attention weights are applied. Each v_i and its corresponding k_i offer two, possibly different, interpretations of the same entity. $V \in \mathbb{R}^{d_v \times n_k}$.
Q	Queries	Matrix of n_q vectors (q_j) of size d_q, in which respect attention weights are computed for each q_j. $Q \in \mathbb{R}^{d_q \times n_q}$, $q_j \in \mathbb{R}^{d_q}$.
kaf vaf qaf	Annotation functions	Functions that encode the input and query sequences, producing K, Q, and V respectively.
e_j	Energy scores	$e_j \in \mathbb{R}^{n_k}$ Vector of size n_k, whose scalar elements (e_{ji}) represent the relevance the corresponding k_i given q_j, according to the compatibility function.
w_j	Attention weights	Vector of size n_k, whose scalar elements (w_{ji}) represent the importance of the corresponding k_i to q_j, according to the attention model. $w_j \in \mathbb{R}^{n_k}$.
f	Compatibility function	Function that evaluates the relevance of K with respect to q_j, returning a vector of energy of scores. $w_j = f(K, q_j)$.
g	Distribution function	Function that computes the attention weights from the energy scores. $w_j = g(e_j)$.
z_j	Weighted values	Matrix of n_k vectors (z_{ji}) of size d_v, respecting the application of w_j to V. $z_j \in \mathbb{R}^{d_v \times n_k}$, $z_{ji} \in \mathbb{R}^{d_v}$.
c_j	Context vector	Vector of size d_v, offering a compact representation of z_j. c_j is a mapping of q_j by K and V. $c_j \in \mathbb{R}^{d_v}$.

Psuedo-code of Bayesian Attention Modules

Algorithm 1 Bayesian Attention Modules

```
Require: $\rho$ learning rates
  Initialization of $\theta, \eta, \phi$ and $t = 0$
  while $\theta, \eta, \phi$ have not converged do
      Sample random minibatches of M points from the dataset $\{x_i, y_i\}_{i=1}^{M}$
      Sample M samples $\{\epsilon_i\}_{i=1}^{M}$
      $\lambda = sigmoid(t * \rho)$
      Compute gradients $\frac{1}{M}\nabla_{\theta,\eta,\phi}\sum_{i=1}^{M}\mathcal{L}_{\lambda}(x_i, y_i, \epsilon_i)$ using equation 3.5
      Update $\theta, \eta, \phi$ with a gradient step
      t = t + 1
  end while
  return $\theta, \eta, \phi$
```

Bibliography

[1] H. yi Lee, "Transformer course." [Online]. Available: http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html

[2] ——, "Transformer course slide." [Online]. Available: http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2019/Lecture/Transformer%20(v5).pdf

[3] A. Kushwaha, "Inférence variationnelle: modèle de mélange gaussien." [Online]. Available: https://ichi.pro/fr/inference-variationnelle-modele-de-melange-gaussien-90543555355771

[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2015.

[5] X. Fan, S. Zhang, B. Chen, and M. Zhou, "Bayesian attention modules," 2020.

[6] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *ArXiv*, vol. abs/1706.03762, 2017.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014.

[8] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016.

[9] A. Galassi, M. Lippi, and P. Torroni, "Attention in natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, p. 4291–4308, Oct 2021. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2020.3019893

[10] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020.

[11] C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt, "Advances in variational inference," 2018.

[12] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, p. 859–877, Apr 2017. [Online]. Available: http://dx.doi.org/10.1080/01621459.2017.1285773

[13] Unknown, "f-divergence." [Online]. Available: https://en.wikipedia.org/wiki/F-divergence

[14] ——, “Divergence (statistics).” [Online]. Available: https://en.wikipedia.org/wiki/Divergence_(statistics)

[15] S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih, “Monte carlo gradient estimation in machine learning,” 2020.

[16] Y. Deng, Y. Kim, J. Chiu, D. Guo, and A. M. Rush, “Latent alignment and variational attention,” 2018.

[17] A. Martin, C. Ollion, F. Strub, S. Le Corff, and O. Pietquin, “The Monte Carlo Transformer: a stochastic self-attention model for sequence prediction,” Dec. 2020, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02896961

[18] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” 2016.

[19] S. Zhang, X. Fan, H. Zheng, K. Tanwisuth, and M. Zhou, “Alignment attention by matching key and query distributions,” 2021.

[20] J.-T. Chien and H.-W. Tien, “Variational disentangled attention for regularized visual dialog,” 2022. [Online]. Available: https://openreview.net/forum?id=ZocWLFKDN3a