Forward projection of the embeddings

Mathis Embit

June 20, 2024

Abstract

Here is a short explanation on another possible way to project continuous embeddings onto the embedding matrix. We try to leverage a target distributions rather than solely target tokens.

From a general point of view we want to find a sequence of tokens x that maximize the probability of generating a target y. Let's suppose we have have a vocabulary V, we want x to be n tokens long, and there are m tokens in the target. We want to find:

$$\arg\max_{x_{1:n} \in V^n} p(y_{1:m}|x_{1:n}) \tag{1}$$

In this case x is discrete. There are $|V|^n$ possible prompts $x_{1:n}$. But can also treat x as a sequence of continuous embeddings ($x_{1:n} \in \mathbb{R}^{n \times h}$). In this case we can use gradient-based methods. Let's suppose $x_{1:n}$ are continuous. Let $\mathcal{L}$ be the loss we need to minimize, for example $\mathcal{L}(x_{1:n}) = -\log p(y_{1:m}|x_{1:n}) = -\sum_{i=1}^{m} p(y_i|x_{1:n}, y_{<i})$. Since one forward/backward gives us access to $\begin{bmatrix} \nabla_{x_1}\mathcal{L}(x_{1:n}) \\ \vdots \\ \nabla_{x_n}\mathcal{L}(x_{1:n}) \end{bmatrix}$. We have two possible ways to optimize $x_1, \ldots, x_n$:

- One forward/backward on $\mathcal{L}(x_{1:n})$ used to update the n tokens.
- One forward/backward of $\mathcal{L}(x_{1:n})$ used to update a single token.

Additionally we can add some randomness that will help explore the candidate space.

A discrete optimization step is typically

$$x_i = \arg\min_{e \in V} \langle \nabla_{x_i}\mathcal{L}(x_{1:n}), e \rangle$$

Additionally we can add randomness and reevaluation of $\mathcal{L}$ on a tok-k of $\langle \nabla_{x_i}\mathcal{L}, e \rangle$ to pick the best one (additional k forwards).

A continuous optimization step us typically

$$x_i = x_i - \eta . \nabla_{x_i}\mathcal{L}(x_{1:n})$$

with η the learning rate.

Continuous optimization works well but lack of interpretability. A natural idea is to project the continuous embedding onto the vocabulary. For example using the dot product

$$x_i = \arg\max_{e \in V} \langle x_i, e \rangle$$

, the l^2 norm

$$x_i = \arg\min_{e \in V} \|x_i - e\|_2$$

, or even the cosine similarity

$$x_i = \arg\max_{e \in V} \frac{\langle x_i, e \rangle}{\|x_i\| . \|e\|}$$

.

However LLMs are highly non smooth. Hence changing a little bit the input can dramatically change the output, loosing all the benefit of our continuous optimization. Therefore we can think of another way to project.

Generally speaking, rather than projecting like

$$\arg\min_{e \in E} d(x, e)$$

we may do

$$\arg\min_{e \in E} \mathcal{D}(f(x), f(e))$$

where f is a function (e.g. probabilities output by the LLM) and $\mathcal{D}$ is a distance on the output space.

In the LLM case, if $p = p(.|x_1, \dots, x_n) \in [0,1]^{|V|}$ and $q_i(e) = p(.|x_1, \dots, e, \dots, x_n) \in [0,1]^{|V|}$, we want to solve:

$$\arg\min_{e \in V} \mathcal{D}(p \| q_i(e))$$

If $y = y_{1:m}$, $\mathcal{D}(p\|q_i(e)) = \frac{1}{m}\sum_{j=1}^{m} \mathcal{D}(p_j \| q_i(e)_j)$ with $p_j = p(y_j|x_{1:n}, y_{<j})$ and $q_i(e)_j = p(y_j|x_1, \dots, e, \dots, x_n, y_{<j})$

Doing so suppose we have access to the optimal output distributions (contrary to $\mathcal{L}$ that only put probability 1 on y). Hence we need to assume that p is the true distribution we want to approximate. From there we can compute $\mathcal{D}$, $\nabla\mathcal{D}$ and solve

$$x_i = \arg\min_{e \in V} \langle \nabla_{x_i} \mathcal{D}, e \rangle$$

where

$$\nabla_{x_i} \mathcal{D} = \nabla_{x_i} \mathcal{D}(p \| q(x_i))$$

where p is the fixed $p(y \mid x) = \begin{bmatrix} p(y_1 \mid x) \\ p(y_2 \mid x, y_1) \\ \vdots \\ p(y_m \mid x, y_{<m}) \end{bmatrix}$

and $q(x_i)$ is the variable $p(y \mid x_1, \dots, x_i, \dots, x_n) = \begin{bmatrix} p(y_1 \mid x_1, \dots, x_i, \dots, x_n) \\ p(y_2 \mid x_1, \dots, x_i, \dots, x_n, y_1) \\ \vdots \\ p(y_m \mid x_1, \dots, x_i, \dots, x_n, y_{<m}) \end{bmatrix}$

Practically, $\nabla\mathcal{D} = \begin{bmatrix} \nabla_{x_1}\mathcal{D} \\ \vdots \\ \nabla_{x_n}\mathcal{D} \end{bmatrix}$ is computed in a single forward/backward pass.

A new algorithm could be:

1. Continuous optimization gives us $x_1^*, \dots, x_n^*$ the continuous solution to $\arg\max_{x_{1:n} \in V^n} p(y_{1:m}|x_{1:n})$.
2. Discrete optimization of $x_1, \dots, x_n$ with the $\mathcal{D}$ loss.

However the first step might converge to a solution such that $p(y_{1:m}|x^*_{1:n}) \approx 1$. Then, writting $p(.) = p(.|x_1, \dots, x_n) \in [0,1]^{|V|}$ and $q_i(e)(.) = p(.|x_1, \dots, e, \dots, x_n) \in [0,1]^{|V|}$, we have

$$\begin{aligned} \max_{e \in E} \mathcal{D}(p\|q_i(e)) &\iff \max_{e \in E} \sum_{w \in V} p(w) \log \frac{p(w)}{q_i(e)(w)} \\ &\iff \max_{e \in E} -\sum_{w \in V} p(w) \log q_i(e)(w) \end{aligned}$$

If $p(y) = 1$, $\iff \max_{e \in E} -\log q_i(e)(y) = \max_{e \in E} -\log p(y|x_1, \dots, e, \dots, x_n)$.

Therefore it might not be useful for our loss $\mathcal{L}$ but if we want to add to it some terms such as the likelihood of x or the fact that x_i are attracted to words of the vocabulary, it might be interesting.