Dual form of SGD via Attention

Skorik Sergey

MIPT, 2022

February 21, 2023

Motivation

Introduction

Despite the broad success of neural nets (NNs) in many applications, much of their internal functioning remains obscure. Naive visualisation of their activations or weight matrices rarely shows human-interpretable patterns. In this paper authors propose to revisit the dual form of the perceptron (Aizerman et al., 1964) and apply it in the modern context of deep NNs, with the objective of better understanding how training datapoints relate to test time predictions in NNs.

Preliminaries

Attention

Definition(Unnormalized Dot Attention) let $\mathbf{K} = (\mathbf{k}_1, \dots \mathbf{k}_T) \in \mathbb{R}^{d_{\text{in}} \times T}$ and $\mathbf{V} = (\mathbf{v}_1, \dots \mathbf{v}_T) \in \mathbb{R}^{d_{\text{out}} \times T}$ denote matrices representing T key and value vectors. Let $\mathbf{q} \in \mathbb{R}^{d_{\text{in}}}$ denote a query vector. An unnormalised linear dot attention operation $\mathit{Attention}(\mathbf{K}, \mathbf{V}, \mathbf{q})$ computes the following weighted average of value vectors $\mathbf{v}_t$:

$$\mathit{Attention}(\mathbf{K}, \mathbf{V}, \mathbf{q}) = \sum_{t=1}^{T} \alpha_t \mathbf{v}_t \tag{1}$$

where the weights $\alpha_t = \mathbf{k}_t^\top \mathbf{q} \in \mathbb{R}$ are dot products between key $\mathbf{k}_t$ and query $\mathbf{q}$ vectors, and are called attention weights.

Preliminaries

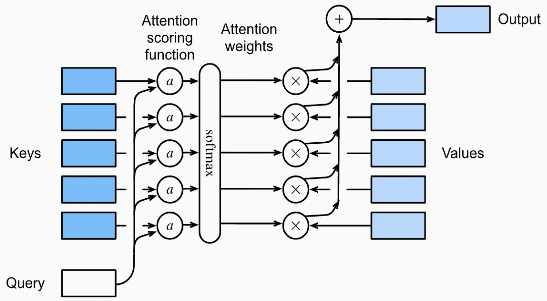

Fig. 11.3.1 Computing the output of attention pooling as a weighted average of values.

Preliminaries

Matrix form

Corollary: Attention computation defined above can be expressed as:

$$Attention(\mathbf{K}, \mathbf{V}, \mathbf{q}) = \mathbf{V}\mathbf{K}^\top \mathbf{q} \tag{2}$$

Proof:

$$Attention(\mathbf{K}, \mathbf{V}, \mathbf{q}) = \sum_{t=1}^{T} \alpha_t \mathbf{v}_t = \sum_{t=1}^{T} \mathbf{v}_t \alpha_t = \sum_{t=1}^{T} \mathbf{v}_t \mathbf{k}_t^\top \mathbf{q} = \left(\sum_{t=1}^{T} \mathbf{v}_t \mathbf{k}_t^\top \right) \mathbf{q}$$

Equivalent System

Two systems S_1 and S_2 defined over the same input and output domains $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$, are said to be *equivalent* if and only if for any input, their outputs are equal, i.e., for any $\mathbf{x} \in \mathcal{D}_i n$, the following holds

$$S_1(\mathbf{x}) = S_2(\mathbf{x}) \tag{3}$$

Idea

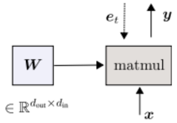

Figure 1. The primal form of a linear layer to be contrasted with the dual form in Figure 2.

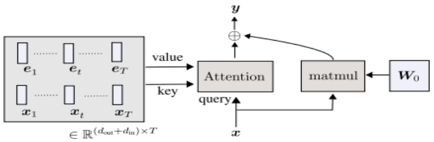

Figure 2. The dual form of a linear layer trained by gradient descent is a key-value memory with attention storing the entire training experience. Compare to the primal form in Figure 1.

Duality

Corollary

The following systems S_1, S_2 are equivalent:

- S_1 (*Primal form*): A linear layer in a neural network trained by gradient descent in some error function using T training inputs to this layer $(\mathbf{x}_1, \dots, \mathbf{x}_T)$ with $\mathbf{x}_t \in \mathbb{R}^{d_{in}}$ and corresponding (backpropagation) error signal $(\mathbf{e}_1, \dots, \mathbf{e}_T)$ with $\mathbf{e}_t \in \mathbb{R}^{d_{out}}$ obtained by gradient descent. Its weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ is thus:

$$\mathbf{W} = \mathbf{W}_0 + \sum_{t=1}^{T} \mathbf{e}_t \otimes \mathbf{x}_t \qquad (4)$$

Where $\mathbf{W}_0 \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ in is the initialisation. The layer transforms input $\mathbf{x} R^{d_{\text{in}}}$ to output $S_1(\mathbf{x}) \in \mathbb{R}^{d_{\text{out}}}$ as:

$$S_1(\mathbf{x}) = \mathbf{W}\mathbf{x} \qquad (5)$$

Corollary

- S_2 (*Dual form*): A layer which stores T key-value pair $(\mathbf{x}_1, \mathbf{e}_1), \dots (\mathbf{x}_T, \mathbf{e}_T)$ i.e., a key matrix $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_T) \in \mathbb{R}^{d_{\text{in}} \times T}$ and a value matrix $\mathbf{E} = (\mathbf{e}_1, \dots, \mathbf{e}_T) \in \mathbb{R}^{d_{\text{out}} \times T}$, and a weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ which transforms input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ to output $S_2(\mathbf{x}) \in \mathbb{R}^{d_{\text{out}}}$ as:

$$S_2(\mathbf{x}) = \mathbf{W}_0\mathbf{x} + Attention(\mathbf{X}, \mathbf{E}, \mathbf{x}) \tag{6}$$

Proof:

$$S_1(\mathbf{x}) = \mathbf{W}\mathbf{x} = \mathbf{W}_0\mathbf{x} + \sum_{t=1}^{T} \mathbf{e}_t \otimes \mathbf{x}_t \cdot \mathbf{x} = \mathbf{W}_0\mathbf{x} + Attention(\mathbf{X}, \mathbf{E}, \mathbf{x})$$

Remarks

Remark 1

Nothing is "forgotten":
The entire life of an NN is recorded and stored as a key matrix X and value matrix E. Roughly speaking, the only limitation of the model's capability to "remember" something is the limitation of the retrieval process

Remark 2

Unlimited memory size is not necessarily useful:
Systems which store everything in memory by increasing its size for each new event (S_2) are not necessarily better than those with a fixed size storage (S_1).

Remark 3

Non-Uniqueness:
The expression of a linear layer as an attention system is not unique.

Main Setup

Common settings

$$Attention(\mathbf{X}, \mathbf{E}, \mathbf{x}) = \mathbf{E}\mathbf{X}^\top \mathbf{q} = \sum_{t=1}^{T} \mathbf{x}_t \mathbf{x} \cdot \mathbf{e}_t = \sum_{t=1}^{T} \alpha_t \mathbf{e}_t$$

We posit that the dual formulation of linear layers offers a possibility to visualise how different training datapoints (i.e., a set of input/error signal pairs seen during training) contribute to some NN's test time predictions. In the paper considering three different scenarios: single-task training, multi-task joint training, and multi-task continual training for image classification using feedforward NNs with two hidden layers. Analogous experiments also are conducted with language modelling using a one-layer LSTM recurrent NN.

How to conduct the experiments?

- Train a neural network using SGD and backprop.
- Save all x_t and e_t for each layer on each iteration.
- Take a test sample $\hat{\mathbf{x}}$, pass it through the neural network, get a test query and calculate $Attention(\mathbf{X}, \mathbf{E}, \hat{\mathbf{x}})$.

Limitations

Since storing the inputs to each linear layer for the entire training experience can be highly demanding in terms of disk space, the authors works with small a datasets: MNIST, Fashion-MNIST (for image classification) and small domain book, WikiText-2 for language modelling.

Common model

Our model for image classification has two hidden layers with 800 nodes, each using relu activation functions after each layer. The total storage of training patterns is thus roughly $3 \times 800 \times T$ units, where T is the "number of training datapoints.

Single-Task Case

Setup

The model is trained on MNIST dataset for $3K$ updates using a batch size of 128 which correspond to $384K$-long training key/value memory slots. The resulting model achieves 97% accuracy on the test set.

How to interpreter attention weights?

- Take test sample with some target $i = \overline{0, 9}$, run through all memory slots and get attention weights
- Sorting attention weights according to the training samples $\mathbf{x}_t$ classes.
- The resulting "groups" of weights are compared in two ways:
  - **Attention weights distribution:** Get top 500 attention weights in each group by weight value
  - **Total attention weights:** Sum attention weights in each group.

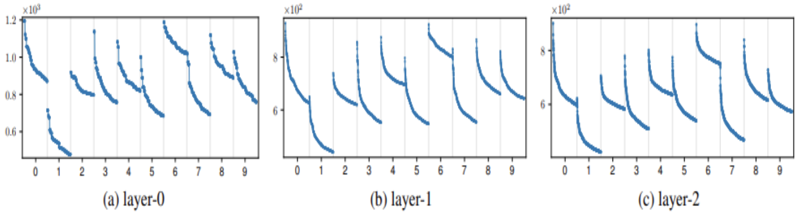

Figure 4. Attention weights over training examples for the input test example from class 6 (Figure 3a) for the single task case on MNIST. The x-axis is partitioned by class, and for each class, top-500 datapoints sorted in descending order are shown.

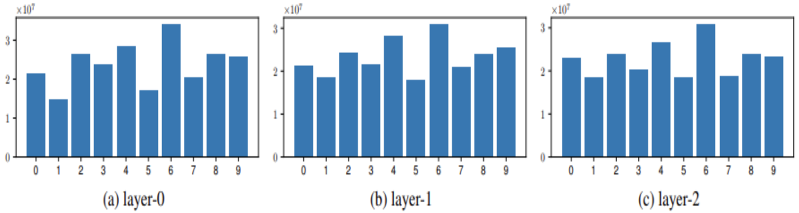

Figure 5. Total scores per class for the input test example from class 6 (Figure 3a) for the single task case on MNIST.

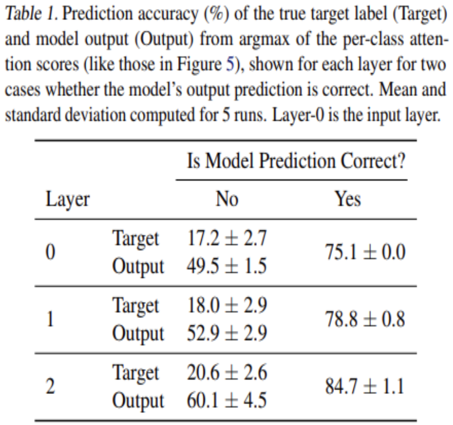

Table 1. Prediction accuracy (%) of the true target label (Target) and model output (Output) from argmax of the per-class attention scores (like those in Figure 5), shown for each layer for two cases whether the model's output prediction is correct. Mean and standard deviation computed for 5 runs. Layer-0 is the input layer.

		Is Model Prediction Correct?	
Layer		No	Yes
0	Target	17.2 ± 2.7	75.1 ± 0.0
	Output	49.5 ± 1.5	
1	Target	18.0 ± 2.9	78.8 ± 0.8
	Output	52.9 ± 2.9	
2	Target	20.6 ± 2.6	84.7 ± 1.1
	Output	60.1 ± 4.5	

Multi-Task Case

Setup

The model is trained on MNIST and F-MNIST (at the same time) dataset for $5K$ updates and achieves 97% and 87% test accuracy on MNIST and F-MNIST, respectively. The output dimension remains 10, so, samples in other domain, but in the same targets are joined. using a batch size of 128 which correspond to $384K$-long training key/value memory slots. The resulting model achieves 97% accuracy on the test set.

How to interpreter attention weights?

- Calcluate attention weigths in the same manner as in Single-Task Case.
- In opposite to target classes, we split attention weigths into 20 classes.
- The main observation in cross-task attention weights.

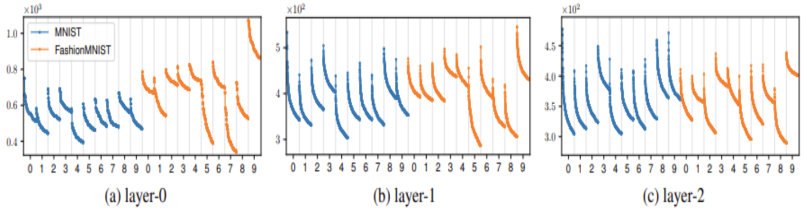

Figure 6. Attention weights over training examples for the input test example from class 9 of F-MNIST (Figure 3b) in the ***joint training*** *case. The x-axis is partitioned by class (for each task), and for each class, top-500 datapoints sorted in descending order are shown.*

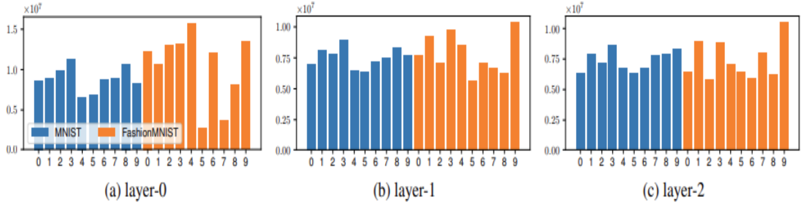

Figure 7. Total scores per class for the input test example from class 9 of F-MNIST (Figure 3b) in the ***joint training*** *case.*

Continual Learning Case

Setup

The model is trained on MNIST and F-MNIST subsequence: first on MNIST for 3,000 steps, then on F-MINST for 3,000 steps. The final test accuracies are 85% on F-MNIST and 45% on MNIST (down from 97% after training on MNIST only).

How to interpreter attention weights?

- Calcluate attention weigths in the same manner as in Single-Task Case.
- Analogous to Multi-Task Case, we have 20 classes of attention weights.
- If we used an attention mask on the F-MNIST part of training keys for the attention computation, the model would be able to reproduce the good performance on MNIST obtained after the first part of training.

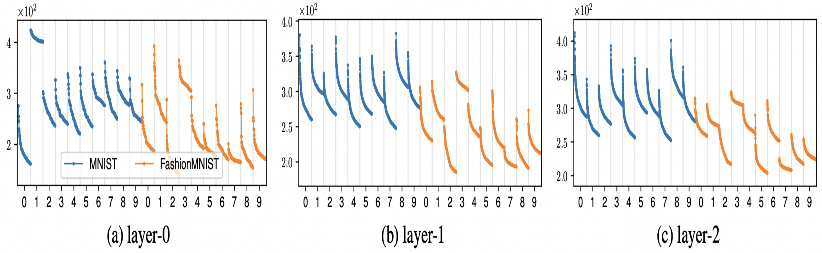

Figure 8. Plots analogous to Figure 6 for the input test example from class 1 of MNIST (Figure 12d) in the ***continual learning case****.*

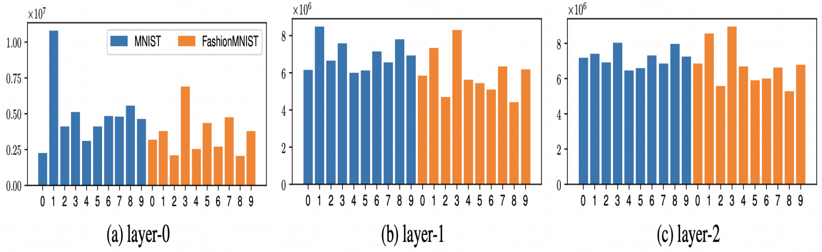

Figure 9. Total scores per class for the input test example from class 1 of MNIST (Figure 12d) in the ***continual training case****.*

Language modelling

Setup

In the experiment using one-layer LSTM LM on two small datasets: a tiny public domain book, “Aesop’s Fables”, by J. H. Stickney and the standard WikiText-2 dataset.

How to interpreter attention weights?

- Train LSTM, store inputs for each linear layer
- At test time, we forward the trained LM on a prompt (short text segment) and get the test query from the last token from the prompt.
- Grouping attention weights by training token position.

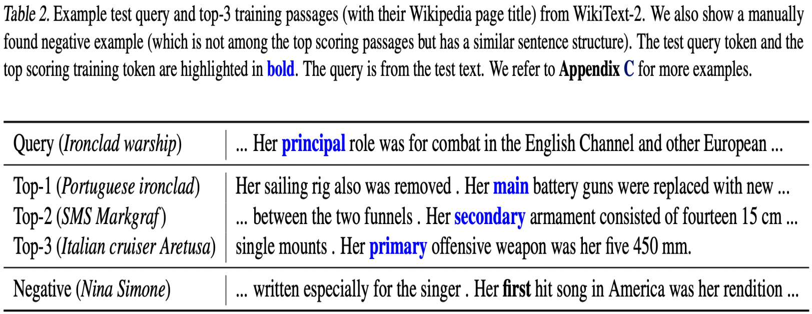

Table 2. Example test query and top-3 training passages (with their Wikipedia page title) from WikiText-2. We also show a manually found negative example (which is not among the top scoring passages but has a similar sentence structure). The test query token and the top scoring training token are highlighted in **bold**. The query is from the test text. We refer to **Appendix C** for more examples.

Query (*Ironclad warship*)	... Her **principal** role was for combat in the English Channel and other European ...
Top-1 (*Portuguese ironclad*)	Her sailing rig also was removed . Her **main** battery guns were replaced with new ...
Top-2 (*SMS Markgraf*)	... between the two funnels . Her **secondary** armament consisted of fourteen 15 cm ...
Top-3 (*Italian cruiser Aretusa*)	single mounts . Her **primary** offensive weapon was her five 450 mm.
Negative (*Nina Simone*)	... written especially for the singer . Her **first** hit song in America was her rendition ...

Discussion and Limitations

- explicit visualization of attention weights. New point of view on NN's
- memory storage requirement
- analysis is not applicable to models which are already trained
- it is limited to a study of attention weights, in line with traditional visualisations of attention-based systems, and can only show which training datapoints are combined. It does not tell how the combined representations can be converted to a meaningful output, e.g., in the case of large generative NNs.