MY360/459 Quantitative Text Analysis: Neural Network Based Language Models

Friedrich Geiecke

March 25, 2024

Course website: lse-my459.github.io

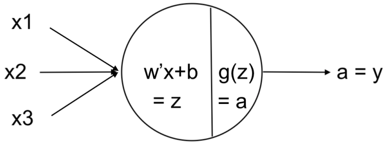
Today

- Given the limited time available in one lecture, we will focus on generative large language models (LLMs) and their alignment which are arguably the most salient topics in neural network based natural language processing at the moment
- Before that, we will review some fundamentals which help with understanding many aspects of more complex architectures later as well:
  - Set up a simple neural network architecture for classification
  - Define its loss function
  - Train the network by minimising that loss function

Outline

- Neural network fundamentals
- Transformers and large language models
- Alignment through supervised learning and reinforcement learning from human feedback
- Coding

Simplest architecture: Single layer perceptron

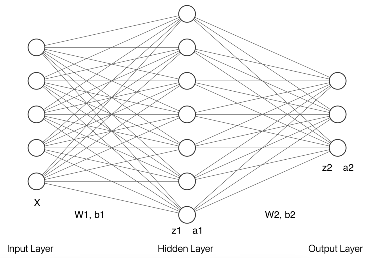


- **Single layer perceptron (SLP)** here: A (**feedforward**) neural network with no hidden layer and an output layer with a single neuron and activation
- $z = x'w + b = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$ with 3-dimensional **weights** vector $w = (w_1, w_2, w_3)$ and **bias** b
- $a = g(z)$ where $g(z)$ is called **activation function**
- Circles in figures commonly depict both z and a values
- Note: With **sigmoid activation** $g(z) = \frac{1}{1+e^{-z}}$, this is just logistic regression!

Discussion

- Note: Conceptually, the neural networks discussed here are simply function approximators that fit the functional relationship between inputs X and outputs y, just like the text regression and classification models you have studied in this course before
- Adding further neurons, hidden layers, etc. allows neural networks to fit increasingly complex functions
- Activations introduce non-linearities
- Stacking just linear layers could only create a linear model

Example of a more flexible network: Classification MLP with one hidden layer



Drawn with `http://alexlenail.me/NN-SVG/index.html`

- Forward pass (discussed in more detail on whiteboard):
  $x \underbrace{\rightarrow}_{\cdot W^{(1)},\ +b^{(1)}} z^{(1)} \underbrace{\rightarrow}_{g(\cdot)} a^{(1)} \underbrace{\rightarrow}_{\cdot W^{(2)},\ +b^{(2)}} z^{(2)} \underbrace{\rightarrow}_{s(\cdot)} a^{(2)} = p$
- Activation middle layer: E.g. ReLu $g(z_i) = max\{0, z_i\}$
- Activation last layer: Softmax $s(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

Loss function

- Cross-entropy loss for a single observation:

$$L(y_i, p_i) = -\sum_{k=1}^{K} y_{i,k} \log(p_{i,k})$$

- Summarise all weights, $W^{(1)}, W^{(2)}, \ldots$ and biases $b^{(1)}, b^{(2)}, \ldots$ in one vector θ

- For full training dataset:

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^{n} L(y_i, p(x_i, \theta))$$

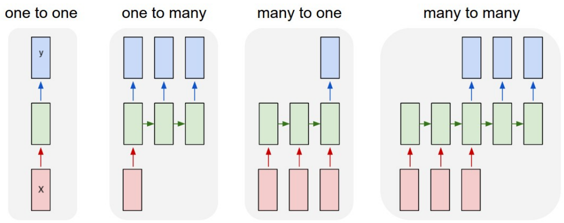
Training

- Compute derivative of loss w.r.t all weights and biases with the chain rule to obtain $\nabla_\theta L\left(y_i, p(x_i, \theta)\right)$ (backpropagation)
- Randomly initialise weights; select a learning rate α. Then repeat the following for E epochs or until approximate convergence:
  1. Shuffle all observations in the training dataset
  2. For each batch with $B \ll n$ observations:
     - Update $\theta \leftarrow \theta - \alpha \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta L\left(y_i, p(x_i, \theta)\right)$
- Current optimisers such as Adam additionally add features like momentum

Common variant for textual data: Recurrent neural networks

- Recurrent neural networks (RNNs) can process sequences of inputs and predict sequences of outputs
- RNNs are e.g. used in machine translation, sentence completion, sentiment analysis, image captioning, etc.
- Common types of RNNs such as Long Short-Term Memory (LSTM) or those with Gated Recurrent Units (GRUs) are based on cells which improve the model's ability to remember long term dependencies
- Potential challenges: Vanishing/exploding gradients and limited parallelisability of sequential computation
- More recently, transformer neural networks have taken over a lot of these tasks (discussed in more detail next)

Common variant for textual data: Recurrent neural networks



Source: From Andrej Karpathy's blog; slightly edited

- Arrows are functions/transformations, rectangles are vectors, green rectangles hold states
- One to one: Standard feed forward neural network
- One to many: RNN that e.g. takes an image as input and then outputs a sentence describing it
- Many to one: RNN that e.g. inputs a sequence of words and outputs a sentiment label
- Many to many: RNN that e.g. inputs a sentence in one language and outputs it in another language

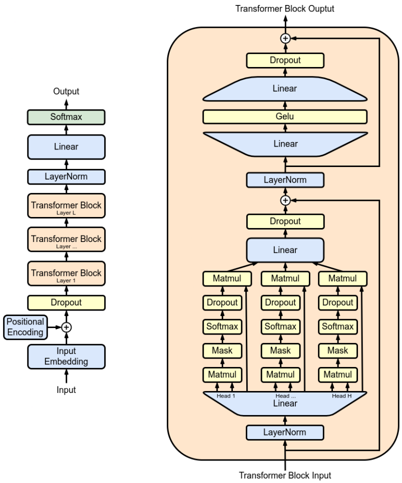
Outline

- Neural network fundamentals
- Transformers and large language models
- Alignment through supervised learning and reinforcement learning from human feedback
- Coding

Transformers

- Introduced by Vaswani et al. (2017)
- Original application was in machine translation with encoder and decoder blocks
- So an input "Machine learning is interesting" was encoded into a numerical representation and then e.g. decoded into "Maschinelles Lernen ist interessant"
- Today encoders and decoders are often used separately
- We will discuss some broad features of GPT-1 and GPT-2 (Radford et al., 2018, 2019) in the following as an example
- From GPT-3.5 onwards, much less details are available

Original GPT architecture



Source `https://en.wikipedia.org/wiki/Generative_pre-trained_transformer`

Some key concepts (1/2)

- At a high level, the model is a function that predicts the next token iteratively Pr(next_token | sequence of previous tokens)
- Tokenisation: A text such as "Natural language processing is interesting" might become [4, 42, 2, 17, 100] with each integer referring to a token in an overall vocabulary (assume for simplicity here that tokenisation is done at the word level). Thus, the model just inputs sequences of integers and also outputs probabilities over the next integer, and these have to be translated back into text
- Embeddings: Each token is represented by a numerical vector / embedding of some chosen dimension (there are also additional encodings representing positions of tokens)

Some key concepts (2/2)

- Attention and context: Attention blocks allow to update each embedding as a linear combination of the embeddings in the context. So the updated vector for “natural” may depend more on/communicate with/pay attention to/have a higher weight on the vector of “language” than on that of “is”
- Feedforward neural network/MLP: Architecture discussed previously; creates nonlinear transformations of vectors
- Attention weights and other network parameters are learned through training on very large amounts of text and with variants of gradient descent minimising a cross-entropy loss

Attention and neural network blocks in more detail

- Discussion on whiteboard

Discussion

- The first pre-training stage that yields a base model can be conceptualised as a kind of lossy compression of internet text data
- As an example, a Llama 2 model variant with $\sim$70 billion parameters may be trained on $\sim$10 terabyte of internet text data. Its resulting parameters can be stored in a $\sim$140GB file. Its training requires $\sim$6,000GPUs for $\sim$12 days and costs around $2M (see Karpathy, 2023, for further details)
- Scaling laws: For now, there has been a relatively predictable improvement in a model's loss just by increasing dataset size and model parameters
- Albeit not the task researchers ultimately care about, that loss of the base model is strongly correlated with many downstream capabilities

Excursus: Encoder transformers

- You may have previously heard of/used transformer based models such as BERT (Devlin et al., 2018) and wonder how these fit into the discussion here
- GPT is based on a decoder architecture: Attention is backward-looking - positions only attend to earlier positions
- BERT is based on an encoder architecture: Attention is backward- and forward-looking - weights relative to tokens before and after a given token can all be non-zero
- Thus, the embedding for "language" in our example could depend on "natural" as well as "interesting" after a decoder attention block
- Model parameters are as usual learned by minimising some loss function, e.g. arising from a task such as masked language modelling or next sentence prediction (in case of BERT)
- Bidirectional contextual embeddings from pre-trained encoders are frequently used in other downstream tasks

Excursus: Document embeddings with encoders

- Such encoder transformers often yield much better document embeddings (e.g. as averages in some last layers) than our previous bag-of-words models
- Briefly returning to our discussion of topic models – could these embeddings be used to find topics?
- One simple approach is to use more recent clustering algorithms (e.g. `https://scikit-learn.org/stable/modules/clustering.html`) on these embeddings
- Yet, unless documents are very short, for this to work well, it may still require to break the documents down into individual topics before embedding and clustering to represent the mixture nature of topics

Outline

- Neural network fundamentals
- Transformers and large language models
- Alignment through supervised learning and reinforcement learning from human feedback
- Coding

Alignment

- Returning to our main example of a generative LLM
- So far the next token prediction objective does not fit that of a helpful assistant. It might auto-complete a question with another question, generate toxic content, etc.
- A key question is how to align its behaviour
- This is currently often achieved with a mix of supervised and reinforcement learning
- We will study these topics by discussing parts of the paper Ouyang et al. (2022)
- Before that, we need to review some fundamentals of reinforcement learning

Review: Reinforcement learning (RL) basics

Figure: Based on Human-level control through deep reinforcement learning Mnih et al. (2015) and on Mnih et al. (2016), Screenshot from: https://youtu.be/xXP77QiHFTs

- Examples of RL applications: Playing Go, Chess, Shogi, or Starcraft (Silver et al., 2017, 2018; Vinyals et al., 2019), robotics (Abbeel et al., 2007), nuclear fusion (Degrave et al., 2022), aligning LLMs such as the GPTs or Llama2 (Ouyang et al., 2022; Touvron et al., 2023)

Reinforcement learning in a nutshell

- In essence, RL is a tool to control dynamic environments
- Learns policy function $\pi_\theta(s) : s \to a$ and value function $v_\nu(s) : s \to \mathbb{R}$

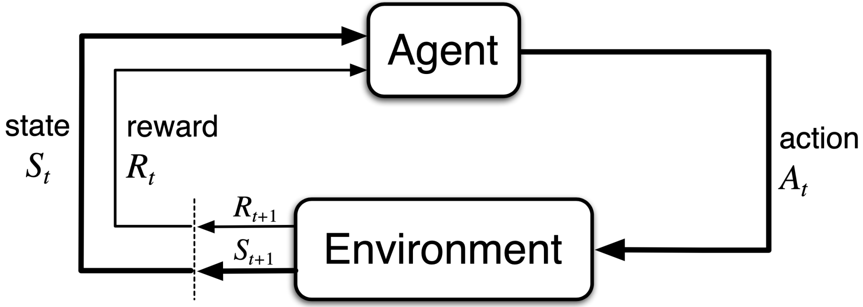


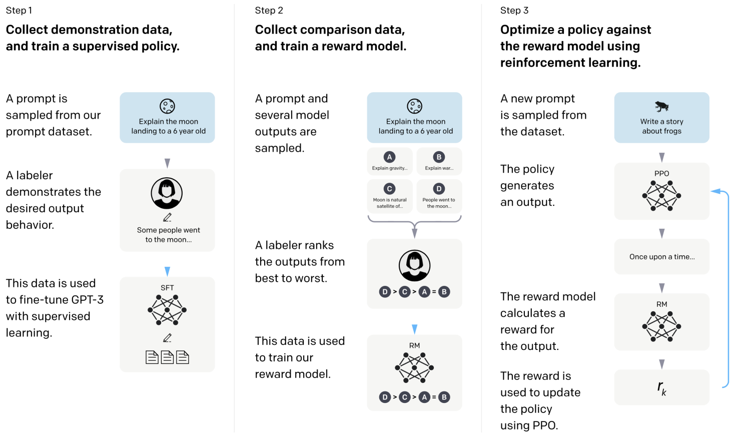
Figure: From Sutton and Barto (2018)

Learning policy functions in RL environments

- Observe state, choose action, observe reward and new state, chose new action, observe new state and reward, etc.
- In regular intervals, update policy and value functions to reinforce actions that yielded high rewards while continuing to explore other actions (typically with decreasing probability)
- Fix policy and value function after learning
- Why so often mixed with discussion of neural networks? Networks are simply used as function approximators for $\pi(\cdot)$ and $v(\cdot)$

Back to Ouyang et al. (2022)

- Three key stages:



Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

Explain the moon landing to a 6 year old
Some people went to the moon...
SFT
Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
D > C > A = B
RM
D > C > A = B
Write a story about frogs
PPO
Once upon a time...
RM
r_k

Step 1: Supervised fine tuning (SFT)

- Paper uses three different GPT-3 model sizes throughout: 1.3B, 6B, and 175B parameters
- Dataset of 13k training prompts (from the API and labeler written)
- Create different types of prompt-answer pairs: Plain (labelers come up with task), few-shot (labelers give examples in prompt), user-based (labelers create prompt-answers matching desired use cases submitted to API + write answers to general prompts from API sandbox)
- Train model using the same objective as before, but now on prompt-answer texts (potentially zero out gradients of prompt parts in some variants of SFT)

Step 2: Reward model

- Dataset of 33k training prompts (from the API and labeler written)
- Let LLMs generate different potential answers for each prompt. Then let labelers score the answers
- Challenge: Labelers would give incomparable scalar scores/rewards, so instead let them rank different potential answers to the same prompt and transform rankings into scalars
- Reward model: Starting from the SFT model (6B) with the final layer removed, train a model to take in a prompt and response, and predict the scalar reward

Step 3: Reinforcement learning from human feedback (RLHF)

- Store copy of LLM as RL policy function
- Conceptualise answer generation as RL environment: Take an initial prompt as state, sample next token as action from policy function (i.e. LLM), combine token with previous state into new state, sample next token as action, etc.
- Episode ends once answer is completed. Run answer text through reward model, obtain reward, and repeat for a batch of prompts. Then optimise policy function (i.e. LLM) with RL (algorithm used: PPO Schulman et al., 2017)
- Note: This creates text in a forward-looking manner rather than greedily with 1-step look-ahead

Discussion

- Very expensive to gather instruction data, but relative to training data of base model, these datasets used for fine-tuning are very small
- For behaviour to change the way it does after fine-tuning on limited instruction data, the findings suggests that models are able to generalize the notion of “following instructions”

Many limitations and challenges

- Alignment to preferences of the group of people creating instructions and training the model, but of course not necessarily to those of others
- Labelers disagree with each other frequently (found the inter-labeler agreement to be about 73%)
- RLHF requires creating expensive instruction datasets and suffers from slow and at times brittle additional training with RL (sparse rewards only at end of answer)
- Whether the reward model is working well or not, RL will move the policy to maximise that reward
- Aligned models might still be tricked
- Approaches could be misused

Outlook

- RLHF is increasingly amended or substituted by reinforcement learning from AI feedback (RLAIF) (see e.g. Bai et al., 2022; Lee et al., 2023)
- RLAIF can allow to align base models also without the resources to create new human labeled feedback datasets
- Inherently creates a feedback loop of AI output with unclear dynamics
- Very recent work such as Direct Preference Optimization (DPO) Rafailov et al. (2023) tries to avoid the separate reward model and and RL steps; Azar et al. (2023) generalises such considerations

Outline

- Neural network fundamentals
- Transformers and large language models
- Alignment through supervised learning and reinforcement learning from human feedback
- Coding

Coding

- 01-nn-text-classification.Rmd
- 02-named-entity-recognition.Rmd
- 03-llms-via-apis.Rmd

Resources for further study

- For very good introductions to transformers, e.g. have a look at `https://jalammar.github.io/illustrated-transformer/` and `https://peterbloem.nl/blog/transformers`
- To learn building neural networks and language models from scratch, an excellent free lecture series is this one by Andrej Karpathy (requires knowledge of Python up to classes)
- It discusses and builds many key parts, e.g. automatic differentiation, computation graphs, or backpropagation, and then ultimately the transformer architecture behind GPT-2

References I

Abbeel, Pieter, Adam Coates, Morgan Quigley, and Andrew Y. Ng, "An Application of Reinforcement Learning to Aerobatic Helicopter Flight," in B. Schölkopf, J. C. Platt, and T. Hoffman, eds., *Advances in Neural Information Processing Systems 19*, MIT Press, 2007, pp. 1–8.

Azar, Mohammad Gheshlaghi, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos, "A general theoretical paradigm to understand learning from human preferences," *arXiv preprint arXiv:2310.12036*, 2023.

Bai, Yuntao, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon et al., "Constitutional ai: Harmlessness from ai feedback," *arXiv preprint arXiv:2212.08073*, 2022.

Degrave, Jonas, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas et al., "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, 2022, *602* (7897), 414–419.

References II

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

Karpathy, Andrej, "Lecture: Intro to Large Language Models," 2023. `https://www.youtube.com/watch?v=zjkBMFhNj_g`.

Lee, Harrison, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi, "Rlaif: Scaling reinforcement learning from human feedback with ai feedback," *arXiv preprint arXiv:2309.00267*, 2023.

Mnih, Volodymyr, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *arXiv e-prints*, Feb 2016, p. arXiv:1602.01783.

References III

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis, "Human-Level Control through Deep Reinforcement Learning," *Nature*, feb 2015, *518* (7540), 529–533.

Ouyang, Long, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray et al., "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, 2022, *35*, 27730–27744.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever et al., "Language models are unsupervised multitask learners," *OpenAI blog*, 2019, *1* (8), 9.

__ **, Karthik Narasimhan, Tim Salimans, Ilya Sutskever et al.**, "Improving language understanding by generative pre-training," 2018.

References IV

Rafailov, Rafael, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn, "Direct preference optimization: Your language model is secretly a reward model," *arXiv preprint arXiv:2305.18290*, 2023.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton et al., "Mastering the game of go without human knowledge," *Nature*, 2017, *550* (7676), 354.

— **, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel et al.**, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, 2018, *362* (6419), 1140–1144.

References V

Sutton, Richard S and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

Touvron, Hugo, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale et al., "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, 2017, *30.*

Vinyals, Oriol, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, 2019, *575* (7782), 350–354.