Aprendizagem Profunda

Perceptron

Cada input é um vetor(weight também) e dá se update 1 a 1.

1. Adicionar bias ao 1º termo dos weights e de todos os inputs.
2. $\hat{y^{(i)}} = \text{Act. func}(w^T x^{(i)})$
3. Se $\hat{y^{(i)}} = y^{(i)}$, próximo input.
4. Caso contrário, $w \leftarrow w + y^{(i)} x^{(i)}$

Linear regression

Meter bias como primeira coluna do X!

$w = (X^T X)^{-1} X^T y$

Feed-Forward

$h^{[0]} = x$
$z^{[i]} = W^{[i]} h^{[i-1]} + b$
$W \rightarrow (\text{Out} \times \text{In})$, $h^{[i-1]} \rightarrow (\text{In} \times \text{N. inputs})$,
$b \rightarrow (\text{Out} \times 1)$
$h[i] = \text{Act. func}(z^{[i]})$

Backpropagation

$\frac{\partial L(y,\hat{y})}{\partial z^{[l]}} = \frac{\partial L(y,\hat{y})}{\partial y}$
$\frac{\partial L(y,\hat{y})}{\partial \mathbf{W}^{[l]}} = \frac{\partial L(y,\hat{y})}{\partial z^{[l]}} \mathbf{h}^{[l-1]^T} \wedge \frac{\partial L(y,\hat{y})}{\partial b^{[l]}} = \frac{\partial L(y,\hat{y})}{\partial z^{[l]}}$
$\frac{\partial L(y,\hat{y})}{\partial \mathbf{h}^{[l]}} = \mathbf{W}^{[l+1]^T} \frac{\partial L(y,\hat{y})}{\partial z^{[l+1]}}$
$\frac{\partial L(y,\hat{y})}{\partial \mathbf{z}^{[l]}} = \frac{\partial L(y,\hat{y})}{\partial h^{[l]}} \odot g'(z^{[l]})$
$\mathbf{W}^{[l]} = \mathbf{W}^{[l]} - \eta \frac{\partial L(y,\hat{y})}{\partial \mathbf{W}^{[l]}} \wedge \; b^{[l]} = b^{[l]} - \eta \frac{\partial L(y,\hat{y})}{\partial b^{[l]}}$

Activation Functions and derivatives

$ReLU(x) = max(0,x) \wedge ReLU'(x) = \begin{cases} 1 & if \;\; x > 0 \\ 0 & if \;\; x \leq 0 \end{cases}$
$\sigma(x) = \frac{1}{1+e^{-x}} \wedge \sigma'(x) = \sigma(x)(1-\sigma(x))$
$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{2}{1+e^{-2x}} - 1$
$tanh'(x) = 1 - tanh^2(x)$
$Softmax(\mathbf{z}) = \frac{exp(z_i}{\sum_{k=1}^{d} exp(z_k)}$

Loss functions and derivatives

$MSE = \frac{1}{2} \sum_{i=1}^{n} (\hat{y_i} - y_i)^2$ (n = size of set/batch)
$MSE' = \hat{y} - y \cdot \hat{y}'$
$BCE = -\sum_{i=1}^{N} [y^{(i)} \cdot log(\hat{y}^{(i)}) + (1 - y^{(i)}) log(1 - \hat{y}^{(i)})]$
$BCE' = \frac{-y}{\hat{y}} \cdot \hat{y}' + \frac{1-y}{1-\hat{y}} \cdot -\hat{y}'$
$CE = -\sum_{i=1}^{N} y_k ln(p_k)$, se softmax $\frac{\partial L(y,p)}{\partial z} = -y + p$
$y = ln(x) \implies y' = \frac{x'}{x}$

Fórmulas para NN e CNN

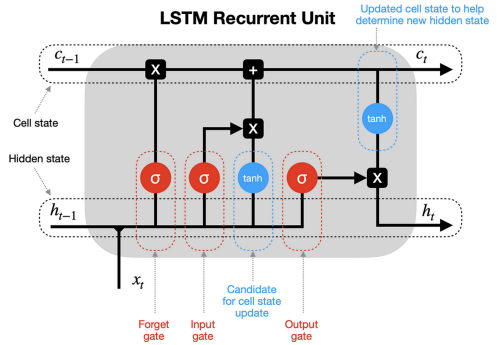
CNN:
Size = $\frac{Input - Kernel + 2 \cdot padding}{stride} + 1$
NumUnits = $Out_w \cdot Out_h \cdot C_{out}$
NumWeights = $C_{out} \cdot (C_{in} \cdot Kernel^2)$
NN:
NumWeights = $C_{out}(C_{in})$
Ambos:
NumBias = C_{out}
NumParams = NumWeights + NumBias

Decoders teoria

Positional Encondings - Uma solução feita de forma a que os transformers sejam capazes de ter em consideração a ordem com que os tokens são colocados em sequência dando a solução tendo isso em mente
Auto-encoders - Auto-encoders are networks that are trained to learn the identify function, i.e.,to reconstruct in the output what they see in the input. This is done by imposing some form of constraint in the hidden representations (e.g. lower dimensional or sparse). They are useful to learn good representations of data in an unsupervised manner, for example to capture a lower-dimensional manifold that approximately contains the data
Masking - É o método usado para controlar que partes do input são observados enquanto o calculo dos *scores*. Isto usa-se quando podemos ter inputs de tamanhos diferentes preservando a causalidade dos inputs.

Attention/RNN Teoria

LSTM - Servem para resolver o problema de vanishing gradient presente nas RNN. Elas resolvem este problema através do uso de uma long e short memory de forma a controlar a quantidade de informação entre estados próximos e estados mais distantes. Se o input for mesmo muito grande não resolve nada! :(
Self attention vantagens - São mais rápidos porque permitem fazer processos em paralelo e conseguem detetar melhor as relações entre tokens mais distantes já que não faz processos de forma sequencial.



Fórmulas para RNN

$z_t^{[1]} = W_{hh} h_{t-1} + W_{hx} x_t + b_h$
$h_t = \text{Act. func}(z_t^{[1]})$
$z_t^{[2]} = W_{yh} h_t + b_y$
$p_t = Probabilidades(z_t^{[2]})$
Caso de cross entropy loss:
$l_t = log[p_t]_{y_t} = -e_{y_t} \cdot ln(p_t)$
$e_y \rightarrow$ one-hot vector of gold outputs(true y)

Fórmulas para Attention

Scaled dot product: $\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_K}}, K \in \mathbb{R}^{L \times d_K}$
· Cada linha de Q, K e V é um vetor vetor associado a query, key e value, respetivamente.
· $Q = XW_Q, K = XW_K, V = XW_V$, onde X têm os seus inputs por linha.
$\mathbf{S} = \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_K}} \in \mathbb{R}^{L \times L}$
$\mathbf{P} = softmax(\mathbf{S}) \in \mathbb{R}^{L \times L}$
$\mathbf{Z} = \mathbf{P}\mathbf{V} \in \mathbb{R}^{L \times d_V}$

1. An RNN-based sequence-to-sequence model with an attention mechanism translates an input sentence of M words into an output sentence with N words. How does the number of computational operations (algorithmic complexity) increase as a function of M and N?
   Solution: $O(MN)$

2. A transformer-based sequence-to-sequence model translates an input sentence of M words into an output sentence with N words. How does the number of computational operations (algorithmic complexity) increase as a function of M and N?
   Solution: $O(max(M, N)^2)$

3. Which activation function best handles the problem of vanishing gradients
   Solution: Relu.

4. Assume we train an LSTM model on English to Portuguese translation. We ensure the training dataset contains the same amount of masculine/feminine pronouns. The resulting model:
   Solution: The marginal frequency of male/female pronouns is not sufficient to ensure that there is no bias. It is important to check the co-occurrence with non-gendered words.

5. Explain how dropout regularization works.
   Solution: At training time, for each example neurons are dropped randomly with probability p (i.e. their activations are masked to become zero) and the remaining activations are scaled by $\frac{1}{(1-p)}$. This forces each neuron to depend less on other neurons' activations.

6. Explain the role and need for positional encoding in transformers.
   Solution: Without positional encodings, the self-attention in transformers is insensitive to the word positions being queried: permuting the words leads to a similar permutation in the self-attention responses. In order for transformers to be sensitive to the word order, each word embedding is augmented with a positional embedding.

7. Mention one advantage of contextualized word embeddings (e.g. BERT) over static word embeddings (e.g. word2vec or GloVe).
   Solution: Contextualized word embeddings can assign different representations to the same word being used in different contexts; this is particularly useful for polysemic words (such as "bank" which can be a river bank or a financial institution).

8. Briefly explain in 1-2 sentences why a CNN is an adequate choice of architecture for Yolanda and Zach's task (image classification).
   Solution: CNNs take advantage of the spacial structure of the image, unlike standard feed-forward networks. Moreover, convolutional and pooling layers exploit the fact that the same feature may appear in different parts of the image, enabling the network to process those occurrences in a similarly way.

9. Give one example of a pretrained model that Bart could use for this task(Generate shorter texts) and the necessary steps to use it.
   Solution: Bart could use a pretrained decoder-only (e.g., GPT) or encoder-decoder model (e.g., T5, BART) and fine-tune it on the data he has available. Alternatively he could use the model without any fine-tuning and use prompting at test time. A possible prompt would be "<document> TL;DR: <answer>". Note: an encoder-only model (e.g. BERT) would not be suitable, since this an auto-regressive generation task.

10. What is exposure bias in sequence-to-sequence models and what causes it?
Solution: Exposure bias is the tendency of sequence-to-sequence models to be exposed only to correct target sequence prefixes at training time, and never to their own predictions. This makes them having trouble to recover from their own incorrect predictions at test time, if they are produced early on in the sequence. Exposure bias is caused by auto-regressive teacher forcing, where models are trained to maximize the probability of target sequences and are always assigned the previous target symbols as context.

11. In sequence models for natural language processing, why are sentences usually sorted by length when batching?
Solution: Within the same batch, all sequences must have the same length, and for this reason they must be padded with padding symbols for making them as long as the longest sentence in the batch. Since in NLP sentences can have very different lengths, if we don't sort sentences by length, we can end up with very unbalanced batches, where some sentences are very short and others are very long, which makes it necessary to add a lot of padding symbols. This process is inefficient and can make training more time consuming. For this reason, sentences are usually sorted by length, which makes each batch more balanced.

12. Mention two advantages of self-attention encoders over RNN encoders.
Solution: They can better capture long-range relations between elements of the sequence, since information does not propagate sequentially (words can be compared with a constant number of operations). This usually reflects in more accurate models. Also, the training is usually faster, since the self-attention computation can be parallelized, unlike RNNs, that require sequential computation.

13. The inception block includes some 1×1 convolutional layers. What does such a layer do, and why may it be useful?
Solution: A 1×1 convolutional layer maps each input "pixel" (with all its channels) to one output pixel in a nonlinear way. It can be thought of as applying a fully connected layer to each input "pixel". This can be used, for example, to reduce or enlarge the number of channels in an image in a "pixelwise" manner, performing a nonlinear transformation on each of them.

14. Explain which problem long-short term memories (LSTMs) try to solve and how they do it.
Solution: LSTMs solve the vanishing gradient problem of RNNs. They do it by using memory cells (propagated additively) and gating functions that control how much information is propagated from the previous state to the current and how much input influences the curreent state.

15. What is an auto-encoder and why it can be useful?
Solution: Auto-encoders are networks that are trained to learn the identify function, i.e., to reconstruct in the output what they see in the input. This is done by imposing some form of constraint in the hidden representations (e.g. lower dimensional or sparse). They are useful to learn good representations of data in an unsupervised manner, for example to capture a lower-dimensional manifold that approximately contains the data.

16. Explain why transformers need causal masking at training time.
Solution: The self-attention in transformers allows any word to attend to any other word, both in the source and on

the target. When the model is generating a sequence left-to-right it cannot attend at future words, which have not been generated yet. At training time, causal masking is needed in the decoder self-attention to mask future words, to reproduce test time conditions.

17. Briefly explain why using a ReLU activation may be preferable over a sigmoid activation in very deep networks.
    Solution: ReLUs are significantly simpler and have a much simpler derivative than the sigmoid, leading to faster computation times. Also, sigmoids are easy to saturate and, when that happens, the corresponding gradients are only residual, making learning slower. ReLUs saturate only for negative inputs, and have constant gradient for positive inputs, often exhibiting faster learning.

18. Give one example of a transformer-based pretrained model that Ada could use for this task(quote attribution) and the necessary steps to use it.
    Solution: Ada could use a pretrained encoder-only (e.g., BERT) or encoder-decoder model (e.g., T5, BART) and fine-tune it on the data she has available. Note: a decoder-only model(e.g. GPT-3) would not be the best choice, since this a classification task.

19. Explain how a bidirectional RNN works and when they are useful.
    Solution: Bidirectional RNNs combine a left-to-right RNN with a right-to-left RNN. After propagating the states in both directions, the states of the two RNNs are concatenated. Bidirectional RNNs are used as sequence encoders, their main advantage is that each state contains contextual information coming from both sides. Unlike unidirectional RNNs, they have the same focus on the beginning and on the end of the input sequence.

20. Explain the difference between self-attention and masked self-attention, as well as why and where the masked version is used.
    Solution: The self-attention in transformers allows any word to attend to any other word, both in the source and on the target. When the model is generating a sequence left-to-right it cannot attend at future words, which have not been generated yet. At training time, causal masking is needed in the decoder self-attention to mask future words, to reproduce test time conditions.

Random theory:
Convolutional layers are equivariant to translations, but not rotations.

Neural networks with a single hidden layer with linear activations are equivalent to linear classifiers, which are not universal approximators.

Auto-encoders with linear activations would correspond to PCA.

Gradient clipping can prevent exploding gradients, not vanishing gradients.

Variational auto-encoders are an instance of a latent variable model. They max- imize a lower bound of the data log-likelihood, GANs do not.

GPT-3 is a decoder-only model. BERT is an encoder-only model trained on a masked language modeling objective.

The family of encoder-decoder large language models is suitable to both clas- sification and generation.

Decoder-only models, including GPT, are trained with a causal language modeling objective.

Prompting is a more efficient way to adapt models than fine-tuning, but it is usually done with models such as GPT, which are trained with a causal language modeling objective.