Dry Part

1. In the above wet assignment we specified you should use latent vectors of dimension 128. Imagine we'd ask you to experiment with a larger range of latent dimensions, where for each dimension you'd be training a new latent space and then try to classify over that learned space. Assuming only the latent dimension changes in each experiment, how do you think the classification accuracy would change with respect to the hidden dimension? Namely, would it decrease up to a certain h value and then increase with growth in h, increase up to a certain h value and then decrease with growth in h, or would it be monotonically increasing/decreasing? Explain your answer.

Answer :

The classification accuracy will increase up to a certain h value and then will decrease as h continues to grow , this happens because while a too small latent space cannot capture sufficient information , too large a latent space can cause overfitting or redundant features . Thus the relationship is not monotonic , there is an optimal latent dimension that balances both expressiveness and generalization but using a much larger latent dimension will just lower the accuracy of the classification .

2. You want to use a 5-layer MLP to perform some regression task. Your friend claims that according to the Universal Approximation Theorem (UAT) for deep neural networks, any MLP with a single hidden layer can achieve optimal error over any dataset and loss criterion. He therefore concludes that there's never really a reason to use MLPs with more than one hidden layer.

   (a) Briefly explain what the UAT means, and how it supports the claim that you can achieve optimal error over any dataset and loss criteria.

   (b) Explain why his conclusion (never use more than one hidden layer) is wrong.

Answer :

a) UAT states that a neural network with one hidden layer and a sufficient number of neurons can approximate any continuous function on a bounded domain to any desired accuracy.
This supports the claim that in theory a single MLP hidden layer if large enough can achieve low approximation error on many types of datasets and loss functions , which means it can achieve optimal error for regression .

b) The friend's conclusion is not correct , like we said in 'a' , that in theory it works however in practice using only one hidden layer is often inefficient , complex functions may require huge number of neurons in a shallow network , while a deeper network can learn the same function more efficiently with better generalization.
Deeper MLPs can model compositional patterns and are easier to train most of the time , its also more scalable for real-world tasks.
In short , having more layers is actually necessary .

3. Alice and Bob want to perform image classification over a large, annotated dataset they found online. Alice suggests using an MLP, while Bob thinks it's better to use a CNN architecture.

   (a) Help Bob convince Alice - what is the main advantage of using a CNN over an MLP in this case? What are the main difficulties Alice might encounter of she uses an MLP?

   (b) Alice heard your statements from the previous section, but is still not convinced. She claims that the convolution operation is in any case linear. Therefore, there shouldn't be a difference between using linear layers between activations (MLP) or convolutional operations between activations (CNN). Do you agree with her? Explain.

Answer :

a) CNNs are better than MLP for image classification , because they preserve spatial structure , use fewer parameters due to weight sharing , and are more robust to position changes in images , also generalize better .
   Using MLP will be challenging , having high parameter counts leads to overfitting and inefficient training , also the loss of spatial information when flattening image input , and poor scalability to high-resolution images .

b) Even though convolution itself is a linear operation , CNNs use non-linear activation functions after convolution so the overall layer is not linear .
   At the same time CNNs architecture has structural advantages such as local connectivity , weight sharing and spatial hierarchy , which MLP lacks so MLP can't exploit image structure as effectively .
   Overall , CNNs are far more efficient and effective for image-related tasks .

4. In the optimization tutorial we learned about several optimization algorithms performing Exponential Moving Average (EMA). What would happened if instead of EMA, we'd be using linear averaging of accumulated gradients? Would you expect the algorithm's efficiency be compromised? Explain

Answer :

In optimization algorithms , EMA is used to smooth out the gradients by giving exponentially decreasing weights to older gradients which allows the optimizer to be more responsive to recent updates while at the same time considering past information , if instead of EMA we use linear averaging which gives all past gradients the same weight , the optimizer would become less adaptive and more sluggish in responding to recent changes in the loss landscape.

Which could cause a slower convergence , reduced stability and higher susceptibility to noise.
In conclusion, replacing EMA with linear averaging would compromise the algorithm's efficiency and effectiveness in training deep learning models.

5. Based on what we learned in the automatic differentiation mechanism tutorial, explain why Pytorch demands the loss tensor to be a scalar in order to perform backpropagation (i.e. run *loss.backward()*).

Answer :
PyTorch requires the loss tensor to be a scalar for loss.backward() because automatic differentiation is designed to compute gradients of scalar outputs with respect to model parameters.
A scalar loss ensures that the backward pass produces a clear gradient vector using the chain rule. If the loss were not a scalar, the resulting gradient would be a more complex structure like a Jacobian, which PyTorch does not handle automatically without additional input.

6. (a) Explain the term *Inductive Bias*
   (b) What is the inductive bias of CNNs? Give at least 2 potential "biases".
   (c) Generally, what are the pros and cons of training a model with inductive bias? In what cases is more inductive bias good? In what cases would we prefer avoiding it?

Answer :
a) Inductive Bias refers to the set of assumptions a learning algorithm makes to generalize from the training data to unseen data , the fact that models can't learn from limited data alone raises the need for a built-in preferences or assumptions aka biases that guide the learning process .

b) Two potential biases for CNN :
   1. Translation invariance : the assumption that the object of interest can possibly appear anywhere in the input and still be recognizable.
   2. Local Connectivity : the assumption that nearby pixels are more related than distant ones , allowing CNNs to focus on local features using small filters.

c) Pros of Inductive Bias and when its useful to use :
   Inductive bias can improve generalization, especially when the assumptions align with the structure of the data. It also makes learning more efficient and requires less training data.
   It is helpful or useful to use when we have prior knowledge about the data .

   Cons and when to avoid using Inductive Bias :
   If the bias is too strong or mismatched with the task, it can limit the model's flexibility and performance.
   In tasks with highly unstructured or unfamiliar data it is recommended to avoid using Inductive Biases because using a strong one may misguide learning , so more flexible and data-driven models would be preferred .

7. You are trying to solve some sequence modeling problem using self-attention. Your input sequence has n tokens, encoded to key, query and value matrices $Q, K, V \in \mathbb{R}^{n \times d}$.

   (a) What is the computational time complexity of computing the output of the attention layer - $Softmax(\frac{QK^T}{\sqrt{d}}) \cdot V$?

   (b) Assuming $d << n$, Bob offers to reduce the computational complexity by changing the attention operation to $Softmax(\frac{Q}{\sqrt{d}})(K^T \cdot V)$ (note the parenthesis around the $K^T \cdot V$, indicating this product is computed first). Why does this help reduce computational time complexity? Does this preserve the function of the original attention formulation?

Answer :

a) The time needed to compute QK^T is $O(n^2d)$, applying softmax over each row takes $O(n^2)$, multiplying the result with V also takes $O(n^2d)$ so overall the complexity is $O(n^2d)$.

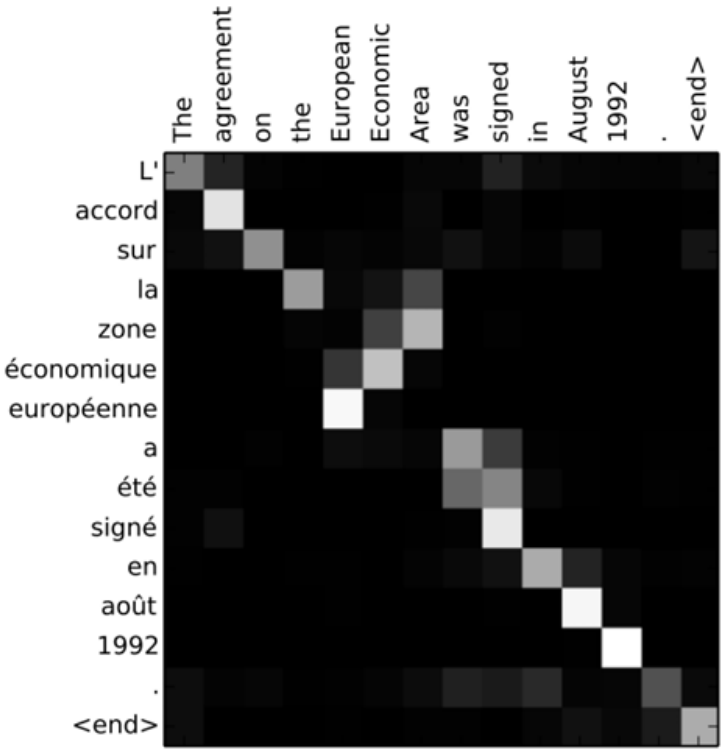
b) Regarding the complexity , Bob's version does it efficiently , he computes K^TV first which is R^{dxd} $instead$ of R^{nxd} which makes the complexity $O(nd^2)$ and the new complexity is more efficient since $d \ll n$.

However , the modified formulation doesn't preserve the original function of attention , In the original form, attention weights are computed using pairwise similarities between all query-key pairs QK^T which is then applied to V .

In Bob's version the attention becomes linear and doesn't compute pairwise token interactions which makes it lose the advantage of self-attention and its ability to model relationships between tokens .

8. Consider the following attention map, received during inference of a well-trained model performing English-French machine translation:

Explain the received attention map:



(a) What does each pixel in the map represent?

(b) What is the meaning of having rows with only one non-zero pixel?

(c) What is the meaning of rows that have several non-zero pixels?

(d) Explain why only rows with one non-zero pixel have a white pixels, while the rest have only gray pixels.

Answer :

a) Each pixel represents the attention weight between a source word in English and a target word in French during translation, A brighter pixel means a higher attention weight, which means the target word is focusing more on that specific source word when being generated.

b) A row with only one non-zero pixel means that the target word is attending strongly and almost exclusively to a single source word , this happens while doing 1 to 1 word translations.

c) Rows with multiple non-zero pixels mean the target word is attending to multiple source words, this usually happens when translating phrases, compound expressions, or when the target word's meaning depends on broader context.

d) White pixels means very high attention weights which is close to 1 , it appears when a target word attends to one source word only , however when the attention is distributed between multiple words , each pixel has a smaller value resulting in gray rather than white (like we mentioned in 'a' a brighter color means a higher weight).
From what we mentioned in 'b' one non zero means that the target word is attending strongly , thus why its colored in white , and in 'c' mean the target word is attending to multiple source words thus why its colored in gray .

9. In the tutorial we've formulated:

$$\log p_\theta(x_0) = \mathbb{E}_q[\log(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)} \cdot \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)})] = D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0)) + \mathbb{E}_q[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}]$$

.
In training we drop the KL-divergence term, and only optimize the ELBO term $\mathbb{E}_q[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}]$.

(a) What is the mathematical basis we rely on when we ignore the KL-divergence term?

(b) Why can't we compute the KL-divergence term?

(c) In the tutorial we further develop the ELBO term to

$$\mathcal{L}(\theta; x_0) = -D_{KL}(q(x_T|x_0)||p_\theta(x_T)) - \sum_{t>1} \mathbb{E}_q[D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}] + \mathbb{E}_q[\log p_\theta(x_0|x_1)]$$

. In training we ignore the term $-D_{KL}(q(x_T|x_0)||p_\theta(x_T))$. Explain why.

Answer :

a) We ignore the KL-divergence term based on the variational inference identity, which expresses the log-likelihood as the ELBO plus a KL term. Since the KL-divergence is always non-negative, the ELBO serves as a lower bound. Ignoring the KL assumes that our approximate posterior is close to the true posterior, making the KL term negligible during optimization.

b) We can't compute the KL-divergence because it requires access to the true posterior $p_\theta(x_{1:T}|x_0)$which is generally intractable. This is due to complex dependencies and high-dimensional integrals that we cannot evaluate directly, especially in latent variable models.

c) The KL term is often ignored because $p_\theta(x_T)$ is either unknown or difficult to compute, also this term usually doesn't depend on the model parameters, so excluding it doesn't affect the training gradients, allowing for simpler optimization.

10. When working on the wet assignment, Bob made the following distinction - *"In the self-supervised autoencoder trained over MNIST, the decoder component learns to map latent vectors onto real MNIST images. We can randomly draw random vectors, feed them to the decoder to get a generative model of handwritten digit images!"*.

    (a) Explain why Bob is wrong.

    (b) In spite of the issue with Bob's idea, his partner Alice became very excited about the notion of optimizing an autoencoder to pose as generative model. She came across *Variational Autoncoders* (VAEs) - explain what those are, how they differ from "standard" autoencoders (like the one you trained in the wet assignment), and how the modifications applied to them allow them to pose as generative models.
    You may use any source of information, however we specifically recommend you review our course's VAE tutorial (not covered this semester)

Answer :

a) Bob is wrong because a standard autoencoder doesn't learn a structured latent space suitable for random sampling. The decoder is only trained to reconstruct inputs from specific latent codes produced by the encoder, not from arbitrary random vectors. So, feeding random vectors to the decoder usually won't generate meaningful images.

b) VAEs are modified autoencoders that treat the latent space probabilistically. Instead of encoding inputs to a fixed point, the encoder outputs a distribution, and we

sample from it during training. A KL-divergence term encourages these distributions to align with a known prior (like a normal distribution), which makes the latent space smooth and suitable for generation. Unlike standard autoencoders, VAEs can generate new data by sampling from this structured latent space and decoding it.