GMLQC
Summer Semester 2024

Exercise Sheet 11

Due: 09.07.24 14h00

Fred Hamprecht
Gerrit Gerhartz, Christof Gehrig

General Regulations.

- Please hand in your solutions in groups of up to two people.
- Your solutions to theoretical exercises can be either handwritten notes (scanned), or typeset using LaTeX. In case you hand in handwritten notes, please make sure that they are legible and not too blurred or low resolution.
- For the practical exercises, always provide the (commented) code as well as the output, and don't forget to explain/interpret the latter. Please hand in an exported PDF of your notebook.
- Submit all your files in the Übungsgruppenverwaltung, only once for your group.

1 Paper reading

- M-OFDFT [1]: Please read A.4.

Exercise: Why is density fitting needed? What metric do they use to fit the densities? (4 pts)

2 Equivariant graph network

In this exercise, we will predict energies from the `MD17` dataset, which contains molecular dynamics simulations for specific molecules (we will take a look at ethanol). Here the relative distance between the atoms is important and therefore we can't use the GCNConv layer from before, because it does not take into account the atomic positions.

(a) The first task will be to load the `MD17` dataset from `pytorch_geoemtric` (specifically the part where `name="revised ethanol"`). In this exercise we will use a cutoff graph with $r_{cutoff} = 5.0\text{Å}$, therefore you need to specify a transformation when loading the dataset. You can use `torch_geometric.transforms.RadiusGraph`. Split up the dataset into train, validation and test split, with 1000, 100 and 100 samples respectively.

(1 pt)

(b) The input to our model will be the atom coordinates R_j and the atomic numbers Z_j. We need to embed these features to be used efficiently by the model. The atomic numbers can be embedded by the `torch.nn.Embedding` class into a scalar feature vector.

(1 pt)

(c) Next, we will embed the relative distance of two neighboring atoms. For this we will use a so called Gaussian-Embedding, where the distance will be embedded by evaluating different gaussians,

$$r_{ij,k} = \frac{1}{\sigma_k\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(||R_i - R_j|| - \mu_k)^2}{\sigma_k^2}\right).$$

Here, k specifies the gaussian. We will set the number of gaussians to 32. The gaussians should be uniformly placed in $[0, r_{\text{cutoff}}]$. σ_k should be chosen in a way, that the gaussians overlap a bit. Plot the gaussians.

(2 pts)

(d) Now the relative distance vector needs to be embedded. For this, we use the `e3nn.o3.spherical_harmonics`, where we feed in the relative distance. You can set $l_{\max} = 2$. In the end, you can concatenate the distance and the angular embedding to one edge attribute feature vector e_{ij}.

(2 pts)

(e) We want to keep track of how our features transform: the embedded charges and distances are invariant (scalars), while the angular par of the difference transform according to their l. Specify the transformation behavior of the node and edge features using `e3nn.o3.Irreps` such that `e3nn` can build a tensor product. Also choose sensible irreps for the nodes in the hidden layers.

(2 pts)

(f) Write an equivariant convolutional layer:

$$x_i' = \sigma\left(\sum_{j\in\mathcal{N}_i} x_j \otimes e_{ij}\right).$$

The layers are made up of a tensor product (using `e3nn.o3.FullyConnectedTensorProduct`) and a non-linearity. You can use `torch_geometric.nn.MessagePassing` as a base class, call `propagate` in the `forward` method and implement the actual calculations in `message(self, x_j, edge_attr)`. The `x_j` signifies that this is the node feature of the neighbor that will be summed over. The tensor product is calculated between the `x_j` and the edge attributes e_{ij}. For the non-linearity to be equivariant, we can't apply an activation function on every element. Instead, we will use the so-called gated non-linearity `e3nn.nn.Gate`, where the scalar features are split up into two parts: The first part is fed through a `SiLU` non-linearity, the second part is fed through a `sigmoid` non-linearity and then multiplied with the higher-l features. Note that you need to change the output irreps of the tensor product to account for the extra scalars needed in the gated non-linearity.

(4 pts)

(g) Combine the embedding and two equivariant convolution layers to obtain an equivariant model. Apply a linear layer to the scalar node features and add them to obtain a prediction for the energy. You can use `torch_geometric.nn.global_add_pool`.

(2 pts)

(h) Train your model and plot the loss and the validation MAE. State the test MAE. You can use your train loop from exercise sheet 09.

(2 pts)

References

[1] He Zhang et al. "Overcoming the barrier of orbital-free density functional theory for molecular systems using deep learning". In: *Nature Computational Science* 4.3 (Mar. 2024), pp. 210–223. ISSN: 2662-8457. DOI: 10.1038/s43588-024-00605-8. URL: https://uebungen.physik.uni-heidelberg.de/c/image/d/vorlesung/20241/1883/material/zhang_24_overcoming.pdf.