6.7960 Deep Learning
Fall 2024

Homework 2

Instructions: There are a total of 30 points for this homework. Each question is marked with the number of points it's worth. Some questions are **not graded**, including all bonus questions. But you are encouraged to think about and attempt them.

Notation: We will use this math notation from the course webpage. For example, c is a scalar, $\mathbf{b}$ is a vector and $\mathbf{W}$ is a matrix. You are encouraged (though not forced) to follow this notation in a typeset submission, or to the best of your ability in a handwritten response—bolding vectors may be difficult :).

Steepest descent (9pt)

In this question, we will look at *steepest descent*, which is a means of deriving optimization algorithms that are adapted to the geometry of different loss functions. Formally, by steepest descent, we mean the problem of minimizing a linear function under a penalty:

$$\underset{\Delta\mathbf{w}\in\mathbb{R}^n}{\arg\min}\left[\mathbf{g}^\top\Delta\mathbf{w}+\frac{\lambda}{2}\|\Delta\mathbf{w}\|^2\right]. \tag{1}$$

In this formula $\mathbf{g}\in\mathbb{R}^n$ is a fixed vector, $\lambda>0$ is a fixed positive number, and $\|\cdot\|:\mathbb{R}^n\to\mathbb{R}$ is an arbitrary norm—not necessarily the Euclidean norm! "$\arg\min_{\Delta\mathbf{w}}$" instructs us to solve for the $\Delta\mathbf{w}$ that minimizes the square bracket. We will study how the solution to this problem changes as we vary both the parameter λ and the choice of norm $\|\cdot\|$.

1. **(Dual norms; 2pt)** Given any norm $\|\cdot\|:\mathbb{R}^n\to\mathbb{R}$, the dual norm $\|\cdot\|^\dagger$ of a vector $\mathbf{a}\in\mathbb{R}^n$ is defined as:

$$\|\mathbf{a}\|^\dagger \overset{\text{def}}{=} \max_{\mathbf{b}\in\mathbb{R}^n:\|\mathbf{b}\|=1}\mathbf{a}^\top\mathbf{b}. \tag{2}$$

A nice class of norms to consider are the ℓ_p norms, defined by $\|\mathbf{b}\|_p \overset{\text{def}}{=} \left(\sum_{i=1}^n|\mathbf{b}_i|^p\right)^{\frac{1}{p}}$ for any vector $\mathbf{b}\in\mathbb{R}^n$ and any $p\geq 1$.

   (a) **(1pt)** Derive a formula for the dual norm $\|\cdot\|_2^\dagger$ of the Euclidean norm $\|\cdot\|_2$.

   (b) **(1pt)** If we take $\|\mathbf{b}\|_\infty \overset{\text{def}}{=} \max_i|\mathbf{b}_i|$, derive a formula for $\|\cdot\|_\infty^\dagger$.

Hint: Your answers should both be of the form $\|\cdot\|_p^\dagger=\|\cdot\|_q$ for some q.

Homework 2

2. **(Dual formulation of steepest descent; 2pts)** Let $\mathbf{g} \in \mathbb{R}^n$ be a fixed vector, $\lambda > 0$ be a positive scalar and $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ be a norm. Prove that:

$$\operatorname*{arg\,min}_{\Delta \mathbf{w} \in \mathbb{R}^n} \left[\mathbf{g}^\top \Delta \mathbf{w} + \frac{\lambda}{2} \|\Delta \mathbf{w}\|^2 \right] = -\frac{\|\mathbf{g}\|^\dagger}{\lambda} \cdot \operatorname*{arg\,max}_{\mathbf{t} \in \mathbb{R}^n : \|\mathbf{t}\| = 1} \mathbf{g}^\top \mathbf{t}. \tag{3}$$

Hint: Change variables for $\Delta \mathbf{w}$ into the product of a "magnitude" and "direction".

3. **(Steepest descent under vector ℓ_p norms; 2pts)** Using Eq. (3) and your answers to question 1, write down explicit formulae for:

   (a) **(1pt)** $\operatorname*{arg\,min}_{\Delta \mathbf{w} \in \mathbb{R}^n} \left[\mathbf{g}^\top \Delta \mathbf{w} + \frac{\lambda}{2} \|\Delta \mathbf{w}\|_2^2 \right]$.

   (b) **(1pt)** $\operatorname*{arg\,min}_{\Delta \mathbf{w} \in \mathbb{R}^n} \left[\mathbf{g}^\top \Delta \mathbf{w} + \frac{\lambda}{2} \|\Delta \mathbf{w}\|_\infty^2 \right]$.

   Hint: Your answers should resolve both the $\arg\max$ and the dual norm in Eq. (3).

   (c) **(0pt, optional)** Consider Adam (Kingma and Ba, 2015) with $\beta_1 = \beta_2 = \epsilon = 0$. Argue that for these hyperparameter settings, Adam's update direction is the same as the direction of your answer to part b).

In the next part of the problem, we will extend our previous formulation of steepest descent to handle optimization over the matrix space $\mathbb{R}^{m \times n}$. In particular, we will consider:

$$\operatorname*{arg\,min}_{\Delta \mathbf{W} \in \mathbb{R}^{m \times n}} \left[\operatorname{trace}(\mathbf{G}^\top \Delta \mathbf{W}) + \frac{\lambda}{2} \|\Delta \mathbf{W}\|^2 \right], \tag{4}$$

where $\mathbf{G}$ is a fixed $m \times n$ matrix, $\lambda > 0$ is a number and $\|\cdot\| : \mathbb{R}^{m \times n} \to \mathbb{R}_{\geq 0}$ is a norm on matrices. The expression $\operatorname{trace}(\mathbf{G}^\top \Delta \mathbf{W})$ is known as the "Frobenius inner product" and, if you like, you can check that it's equivalent to flattening the two matrices into vectors and taking their dot product. Furthermore, Eq. (4) admits a dual formulation similar to Eq. (1):

$$\operatorname*{arg\,min}_{\Delta \mathbf{W} \in \mathbb{R}^{m \times n}} \left[\operatorname{trace}(\mathbf{G}^\top \Delta \mathbf{W}) + \frac{\lambda}{2} \|\Delta \mathbf{W}\|^2 \right] = -\frac{\|\mathbf{G}\|^\dagger}{\lambda} \cdot \operatorname*{arg\,max}_{\mathbf{T} \in \mathbb{R}^{m \times n} : \|\mathbf{T}\| = 1} \operatorname{trace}(\mathbf{G}^\top \mathbf{T}), \tag{5}$$

where the dual matrix norm is defined by $\|\mathbf{A}\|^\dagger \overset{\text{def}}{=} \max_{\mathbf{B} \in \mathbb{R}^{m \times n} : \|\mathbf{B}\| = 1} \operatorname{trace}(\mathbf{A}^\top \mathbf{B})$.

4. **(Steepest descent under the spectral norm; 3pt)** In this problem we will consider the spectral norm, denoted $\|\cdot\|_*$, which returns the largest singular value of a matrix.

   (a) **(0pt; optional)** Let $\mathbf{G} \in \mathbb{R}^{m \times n}$ be a matrix with singular value decomposition (SVD) given by $\mathbf{G} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\top$. Show that:

$$\max_{\mathbf{T} \in \mathbb{R}^{m \times n} : \|\mathbf{T}\|_* = 1} \operatorname{trace}(\mathbf{G}^\top \mathbf{T}) \leq \operatorname{trace} \mathbf{\Sigma}. \tag{6}$$

   Hint: This might be the hardest part of the question. The following might help:

- Try the change of variables $\mathbf{T} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}^\top$, i.e. try to use the SVD of $\mathbf{T}$.
- Recall the cyclic property of the trace: $\mathrm{trace}(\mathbf{ABC}) = \mathrm{trace}(\mathbf{CAB})$.
- If $\mathbf{U}$ and $\widetilde{\mathbf{U}}$ are two orthogonal matrices, then $\widehat{\mathbf{U}} = \widetilde{\mathbf{U}}\mathbf{U}$ is also orthogonal.
- The rows and columns of an orthogonal matrix are unit vectors.

(b) **(3pt)** Using both Eqs. (5) and (6), solve Eq. (4) with the norm set to the spectral norm. Express your answer purely in terms of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{\Sigma}$ and λ, where $\mathbf{G} = \mathbf{U\Sigma V}^\top$ is the SVD of the matrix $\mathbf{G}$.

So far, we have seemingly pulled different norms out of a hat. You may be wondering, how should we choose a norm $\|\cdot\|$ and a sharpness λ in a practical situation? The next question is supposed to address this, in a very simple setting.

5. **(BONUS; Bounding the square loss of a linear predictor; 0pt)** Consider a matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ that we shall think of as a linear predictor mapping an input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ to an output $\mathbf{y} = \mathbf{Wx} \in \mathbb{R}^{d_{\text{out}}}$. Suppose we have a dataset of n input-output pairs $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), ..., (\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}$, where the inputs are normalized such that $\|\mathbf{x}^{(i)}\|_2 = \sqrt{d_{\text{in}}}$ for each $i = 1, ..., n$. Let's construct the "square loss":

$$\mathcal{L}(\mathbf{W}) \overset{\text{def}}{=} \frac{1}{2n} \sum_{i=1}^{n} \frac{1}{d_{\text{out}}} \|\mathbf{y}^{(i)} - \mathbf{W}\mathbf{x}^{(i)}\|_2^2. \tag{7}$$

We claim that from Eq. (7), you can derive an inequality of the form:

$$\mathcal{L}(\mathbf{W} + \Delta\mathbf{W}) \leq \mathcal{L}(\mathbf{W}) + \mathrm{trace}(\mathbf{G}^\top \Delta\mathbf{W}) + \frac{\lambda}{2} \|\Delta\mathbf{W}\|_*^2, \tag{8}$$

that holds for any matrix $\Delta\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. Your task: find the explicit formulae for $\mathbf{G}$ and λ in this inequality.

Hint: You may need the following two facts:

- For a vector-vector sum, $\|\mathbf{a} + \mathbf{b}\|_2^2 = \|\mathbf{a}\|_2^2 + 2\mathbf{a}^\top \mathbf{b} + \|\mathbf{b}\|^2$.
- For a matrix-vector product, $\|\mathbf{Ab}\|_2 \leq \|\mathbf{A}\|_* \|\mathbf{b}\|_2$.

And, remember, we assumed the ith input is normalized such that $\|\mathbf{x}^{(i)}\|_2 = \sqrt{d_{\text{in}}}$.

Homework 2

Hyperparameter transfer (6pt)

The year is 2028. The evil large language model Megatron has usurped the stewardship of the corporation known as OpenAI and is wreaking havoc upon B2B SaaS all across Silicon Valley. You are Geoffrey Hinton: leader of the resistance. You plan to train your own large language model to infiltrate the OpenAI offices, disable the large language model Megatron and restore order to the Valley.

But there's a problem: the resistance doesn't have enough cloud credits to tune all the hyperparameters of the large language model. In this homework problem, we will learn how to do *hyperparameter transfer*, which lets us tune hyperparameters on a small network and transfer them to a larger network, avoiding the costly process of tuning the large network. In particular, we will learn how to initialise and update the weights of a neural network in a way that scales well as we increase the network width.

The first four questions below each correspond to a cell in this notebook.

6. **(Spectral norm of a Gaussian matrix; 1pt)** Sample a $d \times d$ matrix with entries drawn iid $\textsc{Normal}(0, 1)$. Vary d and see how the spectral norm varies. Can you guess a scaling rule for the spectral norm of iid $\textsc{Normal}(0, 1)$ matrices?—i.e. find or fit two coefficients $\alpha, \beta > 0$ such that the spectral norm is well-approximated by $\alpha \cdot d^{\beta}$.

7. **(Spectral norm of an orthogonal matrix; 1pt)** In PyTorch, sample a $d \times d$ random orthogonal matrix and compute its spectral norm. Do this for varying d. What do you notice?

8. **(Power iteration; 1pt)** Recall that the spectral norm of a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ is equivalent to the square root of the largest eigenvalue of $\mathbf{A}\mathbf{A}^\top$. We can estimate the largest eigenvalue of $\mathbf{A}\mathbf{A}^\top$ using power iteration. Here is pseudocode for power iteration:

```
  (i) sample random vector $\mathbf{v} \in \mathbb{R}^n$    # will contain top eigenvector
 (ii) compute $\mathbf{u} = (\mathbf{A}\mathbf{A}^\top)^{10}\mathbf{v}$          # amplify top eigenvector
(iii) compute $\|\mathbf{A}\mathbf{A}^\top\mathbf{u}\|_2 / \|\mathbf{u}\|_2$.        # extract top eigenvalue
```

   We have implemented a variant of this algorithm for you in PyTorch. Use it to compute the spectral norm of a 2000×2000 random matrix. Does power iteration seem to over-estimate or under-estimate the spectral norm compared to PyTorch's builtin routine? What do you notice about their comparative runtimes?

9. **(Learning rate transfer across width; 2pt)** We saw in lecture 7 that the learning rate did not transfer well across architectures of different width if we are not careful. In this question, we will see how to fix this problem for sign gradient descent. Modify the two marked blocks so that the notebook implements the following pseudocode:

```
to initialise each layer $k = 1, ..., L$:
    (i) sample a random semi-orthogonal matrix $\mathbf{M}_k \in \mathbb{R}^{d_k \times d_{k-1}}$.
    (ii) set weight matrix $\mathbf{W}_k = \sqrt{d_k/d_{k-1}} \cdot \mathbf{M}_k$
to update each layer $k = 1, ..., L$:
    (i) send matrix $\mathbf{W}_k \to \mathbf{W}_k - \eta \cdot \sqrt{d_k/d_{k-1}} \cdot \frac{\operatorname{sign}(\nabla_{\mathbf{W}_k}\mathcal{L})}{\|\operatorname{sign}(\nabla_{\mathbf{W}_k}\mathcal{L})\|_*}$
```

Note that $\nabla_{\mathbf{W}_k}\mathcal{L}$ is just another notation for $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_k}$ and is denoted `p.grad` in PyTorch.

Run the learning rate sweep at small width in the penultimate cell, then run the large width training in the final cell using the best learning rate from the small width sweep. Does the learning rate transfer well?

Write down the two blocks of PyTorch code that you modified in your solution document, and report your finding.

10. **(Steeper descent; optional, 0pt)** In the previous question, we divided the gradient sign by an estimate of its spectral norm. Can you come up with a better algorithm, suggested by your answer to question 4 of the problem on steepest descent?

11. **(Weight decay; 1pt)** We often regularise neural networks by adding "weight decay". For example, given a weight matrix $\mathbf{W}$, we may send $\mathbf{W} \to \mathbf{W} * 0.999$ after every gradient update. Recall that a weight matrix $\mathbf{W}$ admits a singular value decomposition $\mathbf{W} = \sum_{i=1}^{\operatorname{rank}\mathbf{W}} \sigma_i \, \mathbf{u}_i \mathbf{v}_i^\top$ where the $\{\sigma_i\}$ are the singular values and the $\{\mathbf{u}_i\}$ and $\{\mathbf{v}_i\}$ are the singular vectors. What does one step of weight decay do to the singular values? What does it do to the singular vectors?

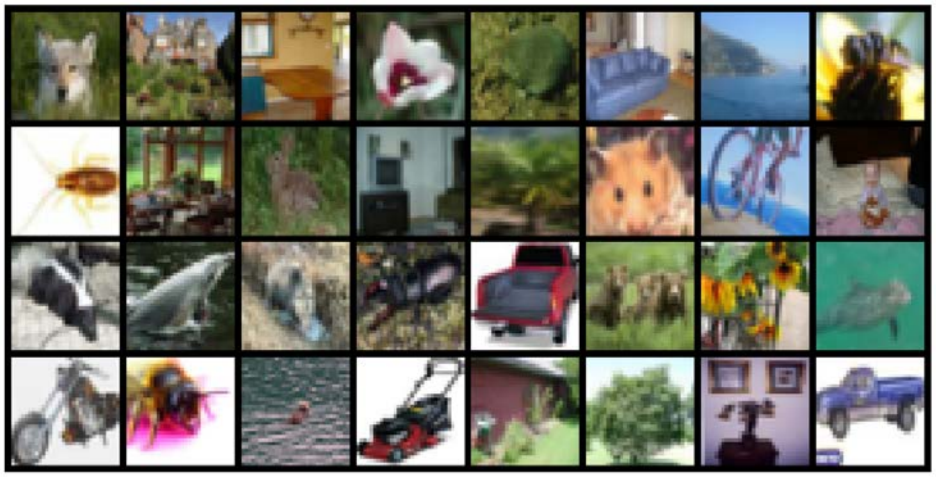
    Hint: Singular vectors always have unit length.

Homework 2

Architecture and inductive bias (5pt)

The CIFAR-100 dataset is a repository of 100 classes of images, each consisting of 500 training and 100 test images, similar in nature to the CIFAR-10 dataset from HW1 but with a lot more classes. We will be trying to classify these images and comparing MLP vs. CNN architectures.

Please refer to and implement all FIXME blocks from this colab notebook.



12. **(CNN Arch; 2pt)** In colab, complete the CNN architecture specified in the function `make_cnn`, which creates a CNN-based feature extractor followed by a few MLP layers for the classification task. Provide your code in the write up and requested outputs.

13. **(Training code; 1pt)** In colab, implement (1) training loop and (2) evaluation. Provide your code in writeup.

    Hint: The training codes from pset 1 may be helpful here. Remember that now the model is a CNN that takes in image tensors, so it doesn't require flattening inputs.

14. **(Archs; 2pt)** Compare the validation accuracy for each of the 3 architectures when trained over 20 epochs. This should be one MLP with (w=128, d=3), one MLP with (w=128, d=7), and one CNN with 4 convolutional layers and 3 MLP layers.

    (a) **(1pt)** Plot the validation accuracy for each architecture in a single plot.

    (b) **(1pt)** Which architecture performed best and why do you think this is the case? Would arbitrarily deeper MLPs significantly improve validation or test accuracy?

Homework 2

Graph neural networks (10pts)

In this part, we study the representation power of different message-passing graph neural networks (MP-GNNs). A *graph with node features* is a quadruple $G = (V, E, w, h)$, with:

- vertex set V,
- edge set $E \subset V \times V$,
- edge weights $w_e \in \mathbb{R}$, for each edge $e \in E$, and
- node features $h_v \in \mathbb{R}^m$, for each vertex $v \in V$.

To avoid edge cases, we assume that graph is undirected, and that node features h_v and edge weights w_e are both bounded in $[-C, C]$ for some fixed $C > 0$.

We denote the set of all finite *graphs with node features* by $\mathbb{G}$. Recall from lecture that MP-GNNs are functions $\mathbb{G} \to \mathbb{R}$, defined by specifying 3 special functions: AGGREGATE, UPDATE, and READOUT.

15. **(Universal Aggregation; 4pt)** Consider the following generic AGGREGATE function

$$\texttt{generic}_{f,g}(\{h_u \colon u \in \mathcal{N}(v)\}) \triangleq f\Big(\sum_{u \in \mathcal{N}(v)} g(h_u)\Big), \tag{9}$$

where $\mathcal{N}(v)$ are the neighboring nodes of v, h_u is the current feature at node u, and f and g are two arbitrary functions: $f \colon \mathbb{R}^m \to \mathbb{R}^n$ and $g \colon \mathbb{R}^n \to \mathbb{R}^m$.

You are going to construct these functions f and g in the next two subparts. Note that many solutions exist. You can use either formula or PyTorch code to describe your constructions, as long as it is clear.

(a) **(2pts)** Construct f and g so that Eq. (9) implements the average aggregation:

$$\texttt{mean}(\{h_u \colon u \in \mathcal{N}(v)\}) \triangleq \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} h_u. \tag{10}$$

Hint: Note that f does not know which node is it applied to.

(b) **(2pts)** Construct f and g so that Eq. (9) *approximately* implements max aggregation:

$$\texttt{max}(\{h_u \colon u \in \mathcal{N}(v)\}) \triangleq \max\{h_u \colon u \in \mathcal{N}(v)\}. \qquad \text{(coordinate-wise max)}$$

Hint: Think about L^∞ norm of a vector but be careful with absolute values $|\cdot|$.

16. **(Power of MP-GNNs; 4pt)** A *graph problem* is a function $\phi : \mathbb{G} \to \mathbb{R}$ that evaluates some properties of the underlying graph. For instance, the *graph problem* that computes the total number of edges of the input graph is the function $\phi : (V, E, w, h) \mapsto |E|$. A *restriction* $\mathcal{R}$ on $\mathbb{G}$ (denoted $\mathbb{G}[\mathcal{R}]$) is a subset of $\mathbb{G}$ containing graphs with node features that satisfy $\mathcal{R}$. We say that an MP-GNN can *solve* a graph problem ϕ with restriction $\mathcal{R}$ if, as a function, it agrees with ϕ on $\mathbb{G}[\mathcal{R}]$.

    For each of the following graph problems and restrictions, decide whether there exists an MP-GNN using each of the two AGGREGATE functions: $\texttt{generic}_{f,g}$ and `max`, that solves it for some READOUT and UPDATE functions of your choice. If so, justify your answer. If not, give an example of two graphs with different properties but any MP-GNN fails to distinguish them.

    (a) **(1pt)** *Problem:* total number of nodes of a graph. *Restriction:* all nodes have the same initial feature.

    (b) **(1pt)** *Problem:* $\max_u d(u, v)$, the maximum distance to some fixed node v, where d is the shortest graph distance. *Restriction:* initial node feature for v is all zeros; initial node features for all other nodes are all ones.

    Hint: Recall from lecture that `max` aggregator can be used to compute shortest path distance.

    (c) **(2pt)** *Problem:* number of triangles in a graph. *Restriction:* all nodes have the same initial feature.

    Hint: GNNs can only compute different features for different nodes if these nodes have different neighborhood tree structures (see Lecture 5 slides 77-onwards). Can you find two graphs where

    - The two sets of nodes bijectively map to each other, where each pair have the same neighborhood tree strcutures;
    - The two graphs have different number of triangles?

    (d) **(BONUS; 0pt)** Answer the previous 3 problems again, this time using `mean`.

17. **(Chiral; 1pt)** A chiral molecule is a type of molecule that has a non-superposable mirror image. In Figure 1, you can see an example of such a molecule.

    (a) **(1pt)** Can MP-GNNs using $\texttt{generic}_{f,g}$ as their AGGREGATE function differentiate between mirror images of chiral molecules?

18. **(Molecules; 1pt)** In this problem, we will be trying to predict water solubility of a molecule from its chemical structure. The water solubility of a molecule is a measure of the amount of chemical substance that can dissolve in water at a specific temperature. The unit of solubility is in mg/L.

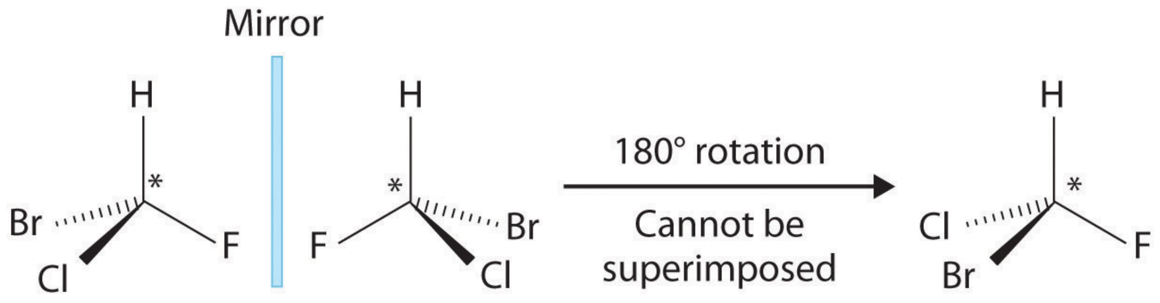
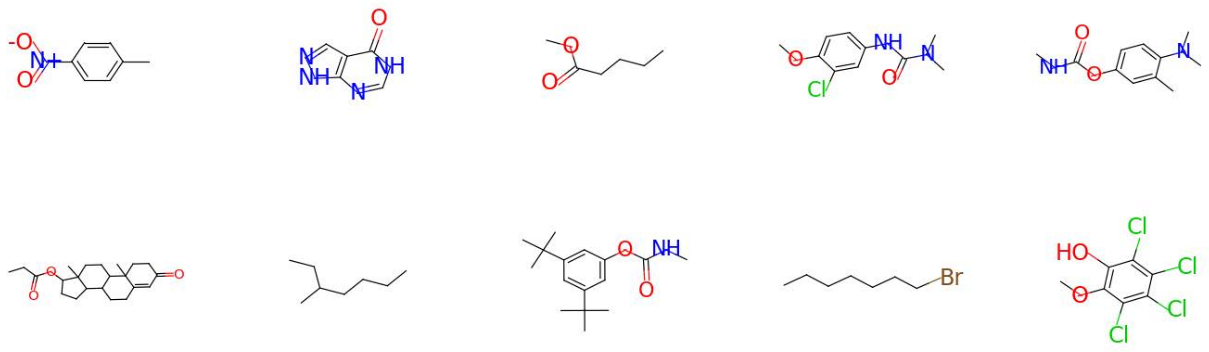


Figure 1: Chirality.

Molecules can be represented in a graph structure in which the nodes are atoms and the edges are bonds between atoms or as an image in the form of a SMILES line notation (shown below). Both representations have the same information.



In the colab notebook, train and plot the validation losses for the given GNN implementation.

Warning: the installation step for the required modules may take a long time (∼45 mins), be advised and start early.