Mathematics behind Machine Learning

Week 4 Exercise Sheet: Neural Networks and Large Language Models

Brief recap

- A neural network is built from **composed functions**.
- Each layer takes an input, transforms it, and passes it to the next layer.
- A simple layer often looks like
$$h = \sigma(Wx + b)$$
where W is a weight matrix, b is a bias vector, and σ is an activation function.
- Deep learning means using **many layers** rather than only one.
- Composition means applying one function after another such as:
$$f(x) = f_3(f_2(f_1(x))).$$
- Common activation functions include ReLU, sigmoid, and tanh; they add non-linearity to the model.
- Large language models (LLMs) turn words into vectors, apply many transformations, and output probabilities for the next token.
- The key idea for LLMs is:
$$\text{input sequence} \longrightarrow \text{many transformations} \longrightarrow \text{probability distribution over tokens.}$$

Exercises

1. A single layer with an activation function

Consider the input vector
$$x = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$
and the layer
$$z = Wx + b, \qquad W = \begin{pmatrix} 1 & -2 \\ 3 & 1 \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ -2 \end{pmatrix}.$$
The output of the layer is then
$$h = \sigma(z),$$
where σ is applied elementwise.

1.1. Compute $z = Wx + b$.

1.2. Compute the output h when $\sigma = \text{ReLU}$, where

$$\text{ReLU}(z) = \max(0, z).$$

1.3. Compute the output h when $\sigma = \tanh$.

1.4. Explain in one sentence why the activation function changes the behaviour of the layer.

2. A three-layer composition

Suppose a network has three layers:

$$f(x) = f_3(f_2(f_1(x))).$$

Let

$$f_1(x) = Wx + b, \qquad f_2(u) = \text{ReLU}(u), \qquad f_3(v) = Pv,$$

where

$$W = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad b = \begin{pmatrix} -1 \\ 2 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 3 \end{pmatrix}.$$

Use the input

$$x = \begin{pmatrix} 3 \\ 5 \end{pmatrix}.$$

2.1. Compute $f_1(x)$.

2.2. Compute $f_2(f_1(x))$.

2.3. Compute $f(x)$.

3. From words to probabilities

In a language model, each token is first converted into a vector called an *embedding*. These embeddings are then processed by several layers of the model, which combine information from the input sequence and produce a vector of *logits*.

A *logit* is a raw score for each possible next token in the vocabulary. The model does not directly output a word. Instead, it outputs one score per possible next token. These scores are then converted into probabilities using the *softmax* function. Softmax turns a list of real numbers into a probability distribution, so that all probabilities are nonnegative and add up to 1.

In this exercise, we will use a very small toy vocabulary with only two possible next tokens:

$$\{\text{dog}, \text{mouse}\}.$$

We will also use very simple embeddings:

$$\text{cat} \mapsto \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \text{dog} \mapsto \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad \text{mouse} \mapsto \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Consider the input sequence

$$\text{cat, mouse.}$$

We will use the following matrix to represent a simplified final layer of the language model:

$$M = \begin{pmatrix} 2 & -1 \\ 1 & 1 \end{pmatrix}.$$

3.1. Add the two embedding vectors together to form a simple sequence representation. In real life it would be some sort of weighted sum, based on prior knowledge and attention.

3.2. Apply the matrix

$$M = \begin{pmatrix} 2 & -1 \\ 1 & 1 \end{pmatrix}$$

to your sequence representation. Interpret the result as the logits for the next token being *dog* or *mouse*.

3.3. Apply the softmax function

$$\text{softmax}(z_i) = \frac{e^{z_i}}{e^{z_1} + e^{z_2}}$$

to convert the logits into probabilities.

3.4. What probabilities do you get for *dog* and *mouse*?

3.5. What does the softmax output mean in the context of language modelling.