Exercise 4-1 ‘The Simpsons’ Characters Classification in Tensorflow

The provided notebook on the website leads you through a classification task of the main characters of the TV-series ‘The Simpsons’. Therefore, we will use the ‘The Simpsons Characters Data’ from kaggle.com, provided by the user alexattia. The original source of the dataset can be found here: https://www.kaggle.com/alexattia/the-simpsons-characters-dataset

In a previous exercise, you implemented and trained a neural network for this task using the high-level API Keras. This time we will implement the classifier in Tensorflow.

Exercise 4-2 Manifold Learning - Locally linear embedding

In locally linear embedding, we assume that data lies on a manifold, where each sample and its neighbors lie on an approximately linear subspace. The idea is to approximate the data by a set of linear patches. This local geometry of these patches can be characterized by linear coefficients that reconstruct each data point from its neighbors. The algorithm consists of three steps:

- For each \( x_i \), find the \( k \) nearest neighbors
- Find the weight matrix \( \mathbf{W} \) minimizing the error for reconstructing each \( x_i \) from its neighbors. The cost function is given by:

\[
\mathcal{E}(\mathbf{W}) = \sum_{i=1}^{n} ||x_i - \sum_{j \neq i} w_{ij} x_j||^2
\]  

where \( w_{ij} = 0 \) unless \( x_j \) is one of \( X_i \)'s \( k \)-nearest neighbors. For each \( i \), \( \sum_{j} w_{ij} = 1 \).

- Find the coordinates \( \mathbf{Y} \) minimizing the reconstruction error using the weights calculated in step 2, i.e.,

\[
\Phi(\mathbf{Y}) = \sum_{i=1}^{n} ||y_i - \sum_{j \neq i} w_{ij} y_j||^2
\]  

subject to the constraint that \( \sum_{i} y_{ij} = 0 \) for each \( j \) and that \( \mathbf{Y}^T \mathbf{Y} = \mathbf{I} \).

(a) Given the local linearity of a manifold, i.e., suppose that the manifold was exactly linear around \( x_i \) and forms a linear subspace. Describe what happens to the error as \( n \) grows. What is the conclusion for the low-dimensional and high-dimensional embedding space w.r.t to the weights?

(b) Show that the constraint of \( \sum_{j} w_{ij} = 1 \) is sufficient to be invariant towards translations w.r.t each input vector \( x_i \).
(c) What happens if \( k \), the number of neighbors, is greater than \( p \), the number of features?

(d) Formulate a optimization problem on how to stabilize the problem in the case on \( k > p \).

**Exercise 4-3**  Generative Adversarial Networks (GANs)

(a) Provide a short and intuitive, non-formal description of a GAN. What is the main idea? What are the individual components of the model and how do they interact?

(b) What are possible applications for a GAN?

(c) Name at least two different variations of the GAN model and provide a possible application for each.

**Exercise 4-4**  Optimal Solution of a GAN

Recall from the lecture the idea of Generative Adversarial Networks (GANs):

A generator \( g \) is trained to produce samples \( x = g(h; \theta_g) \). The idea is that the generator performs especially well, i.e. closely approximates the real data distribution \( p_{\text{data}}(x) \), if a discriminator \( d(x; \theta_d) \) is not able to separate the samples produced by \( g \) from real samples from \( p_{\text{data}}(x) \). Both, generator and discriminator, are usually multi-layer perceptrons, such the \( \theta_g \) and \( \theta_d \) represent their respective weights. We will omit these parameters in the following for brevity. In a GAN, both networks are trained jointly to perform their respective tasks as good as possible:

\[
(\theta_g^*, \theta_d^*) = \arg\min_g \arg\max_d V(g, d)
\]

where

\[
V(g, d) = \mathbb{E}_{x \sim p_{\text{data}}} [\log d(x)] + \mathbb{E}_{h \sim p_h} [\log(1 - d(g(h)))]
\]

Intuitively, the discriminator attempts to classify the samples correctly as real or fake by maximizing the negative cross-entropy. The generator on the other hand attempts to minimize the negative cross-entropy with the goal of fooling the discriminator into believing that its generated samples are real.

In the following, we want to partly proof the following statement:

At the unique global optimum, \( p_g(x) = p_{\text{data}}(x) \) and \( d(x) = 1/2 \) everywhere.

Though the full proof is not particularly difficult, we skip the last part due to time reasons. A nicely worked out, detailed proof is also provided by Scott Rome.\(^1\) Note that in practice the GAN is trained on finite batches of training samples and generated samples using a gradient-based learning rule. One can show that training converges given enough training data and certain other conditions. However, training GANs can be difficult in practice and is actively researched.

\(^1\)https://srome.github.io/An-Annotated-Proof-of-Generative-Adversarial-Networks-with-Implementation-Notes/
(a) Given a fixed $g$, what is the optimal discriminator $d^*_g$?

Hint: We can write $V$ as

$$V(g, d) = \int_x p_{data}(x) \log d(x)dx + \int_h p_h(h) \log(1 - d(g(h)))dh$$

$$= \int_x p_{data}(x) \log d(x) + p_g(x) \log(1 - d(x))dx$$

Further, for any $a, b \in \mathbb{R}$, $a + b > 0$, the function

$$y \rightarrow a \log y + b \log(1 - y)$$

attains its maximum in $[0, 1]$ at $\frac{a}{a+b}$.

(b) Given a fixed generator $g^*$ such that $p_{g^*} = p_{data}$, what is the optimal discriminator $d^*$? What is the value of $V(g^*, d^*)$?

(c) What is left to show for a full proof of the above statement?