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 $W$ minimizing the error for reconstructing each $x_i$ from its neighbors. The cost function is given by:

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

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 $Y$ minimizing the reconstruction error using the weights calculated in step 2, i.e.,

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

subject to the constraint that $\sum_i y_{ij} = 0$ for each $j$ and that $Y^TY = I$.

(a) Under which conditions can we achieve an error of $\mathcal{E}(W) = 0$? 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 an optimization problem on how to stabilize the problem in the case on $k > p$. 
Possible Solution:

(a) Let’s assume that the manifold was exactly linear around $x_i$, i.e., that the point $x_i$ and its neighbors relates to a $q$-dimensional linear subspace. Since $q + 1$ points in generally define a $q$-dimensional subspace, there would be some combination of the neighbors reconstructing $x_i$ exactly, i.e., some set of weights $w_{ij}$ exists, such that

$$x_i = \sum_j w_{ij} x_j$$  \hspace{1cm} (3)

Conversely, if there are such weights, then $x_i$ and some of its neighbors do form a linear subspace.

Further, every manifold is locally linear if we consider a small enough region around each point. In particular, the above equations hold in the limit as $n$ approaches infinity and each point has infinitely many other points in its neighborhood. In this case, the error approaches zero. Hence, the same weights would work to reconstruct $x_i$ both in the high-dimensional embedding space and the low-dimensional subspace. Therefore, the weights around a given point which characterize what the manifold looks like nearby $x_i$. This is why, we will only need the weights, not the original vectors in the last step.

(b) The condition $\sum_j w_{ij} = 1$ give us invariance under translation, i.e., by adding any vector $c$. That is, if we add any vector $c$ to $x_i$ and all of its neighbors, nothing happens to the function we’re minimizing:

$$(x_i + c) - \sum_j w_{ij} (x_j + c) = (x_i + c) - (\sum_j w_{ij} x_j) - c = x_i - \sum_j w_{ij} x_j$$ \hspace{1cm} (4)

Since we are looking at the same shape of manifold no matter how we move it around in space, translational invariance is a constraint we want to impose.

(c) Let $k$ be the number of neighbors which is greater than $p$, the number of features. Then (in general) the space spanned by $k$ distinct vectors is the whole space. Consequently, $x_i$ can be written exactly as a linear combination of its $k$-nearest neighbors. In fact, if $k > p$, then there is not only a solution to $x_i = \sum_j w_{ij} x_j$, but alas there are generally infinitely many solutions! This is because there are more unknowns ($k$) than equations ($p$). In the case of $k > p$, the optimization problem is **irregular**.

(d) One way on how to stabilize the optimization problem is to use regularization. One common way to handle this is to use the $L_2$ regularization. That is to say, instead of minimizing:

$$||x_i - \sum_j w_{ij} x_j||^2$$ \hspace{1cm} (5)

we can pick a regularization parameter $\alpha > 0$ and minimize:

$$||x_i - \sum_j w_{ij} x_j||^2 + \alpha \sum_j w_{ij}^2$$ \hspace{1cm} (6)

The meaning of this is the following: pick the weights minimizing a combination of reconstruction error and the sum of the squared weights. As $\alpha \to 0$, this gives us back the least-squares problem. Let’s take the opposite limit $\alpha \to \infty$: the first term becomes negligible, and we just want to minimize the $L_2$ norm of the weight vector $w_{ij}$. Since the weights are constrained to add up to 1, we can best achieve this by making all the weights equal - so some of them can’t be vastly larger than the others, and they stabilize at a definite preferred value. Typically, $\alpha$ is set to be small, but not zero, so we allow some variation in the weights if it really helps improve the fit.
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?

**Possible Solution:**
The main idea is to train a generator that is able to produce samples which are similar to the training data observed by the model. In particular, the generator is considered to perform well, if a discriminator is not able to tell the generated samples apart from the training data, i.e. if the generator can successfully fool the discriminator. The GAN consists of the generator and discriminator, which are usually multi-layer perceptrons and are trained jointly.

(b) What are possible applications for a GAN?

**Possible Solution:**
Possible applications include:
- Similarly as with encoders, the convolutional layers of a trained discriminator can be used to initialize a discriminator for another task, such as image classification, by exchanging the last layer with a linear layer for prediction. This is particularly useful, if only relatively few labeled samples are available for this task.
- The generator can be used to produce high-quality artificial data samples. This is also useful in a setting with few labeled samples, since the generator can be used to generate more synthetic training samples.

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

**Possible Solution:**
- Conditional GAN (cGAN): Use the class label as an additional input to the generator and discriminator. This makes it possible to produce samples conditioned on a certain label combination. For instance, given labels extracted from a textual description of a bird such as “white with some black on its head and wings and a long orange beak”, the generator can produce plausible images matching that description (reverse captioning).
- InfoGAN: Similar to the cGAN, but the discriminator additionally predicts the class label. A possible application is to not only synthesize novel images, but to modify existing ones, e.g. adding a beard or sunglasses to a face. The InfoGAN improves this task by disentangling different concepts in the latent space. This is achieved by letting the discriminator predict the class label of the samples produced by the generator.
- CycleGAN: Used for unpaired image-to-image translation, e.g. turn horses into zebras without having aligned pairs of horse and zebra images. Train two generators, for mapping horses to zebras and zebras to horses respectively, such that if a zebra is turned into a horse and back into a zebra (analogously for a horse), the original and the resulting zebra are similar. This is called cycle consistency. Additionally, two discriminators are trained which separate real from fake horses and real from fake zebras, respectively. A CycleGAN can be used for many different style transfer tasks, e.g. summer to winter, photo to style of a particular artist and so on.
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

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

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. 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.

(a) Given a fixed $g$, what is the optimal discriminator $d_g^*$?

Hint: We can write $V$ as

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

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

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 $\frac{a}{a + b}$.

Possible Solution:

Using the first hint, we have

\[ V(g, d) = \int_x p_{\text{data}}(x) \log d(x) + p_g(x) \log(1 - d(x)) \, dx \]
\[ \leq \int_x \max_y \{ p_{\text{data}}(x) \log y + p_g(x) \log(1 - y) \} \, dx \]

Using the second hint, the unique maximum is achieved at

\[ d^*_g(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \]

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

Possible Solution:

Let \( g^* \) such that \( p_{g^*} = p_{\text{data}} \), then

\[ d^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{data}}(x)} = 1/2 \]

Plugging this into \( V \), we get

\[ V(g^*, d^*) = \int_x p_{\text{data}}(x) \log d^*(x) + p_{g^*}(x) \log(1 - d^*(x)) \, dx \]
\[ = \int_x p_{\text{data}}(x) \log 1/2 + p_{\text{data}}(x) \log(1 - 1/2) \, dx \]
\[ = \int_x p_{\text{data}}(x)(\log 1/2 + \log 1/2) \, dx \]
\[ = \log 1/4 \int_x p_{\text{data}}(x) \, dx \]
\[ = - \log 4 \]

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

We still need to show that \((g^*, d^*)\) is the unique global optimum of \(V\), i.e. \(V(g^*, d^*) \leq V(g, d^*)\) for all \(g\) with equality if and only if \(g = g^*\). For time reasons we skip this, but if you are interested, here is the last part of the proof:

To see that \(g^*\) is optimal, consider \(V(g^*, d^*)\) for an arbitrary \(g\):

\[
V(g, d^*_g) = \int x \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} dx + \int x p_g(x) \log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} dx
\]

Next, we add zero under each integral to cheat \(V(g^*, d^*)\) into the equation:

\[
V(g, d^*_g) = \int (\log 2 - \log 2) p_{\text{data}}(x) + p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} dx
\]

\[
+ \int (\log 2 - \log 2) p_g(x) + p_g(x) \log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} dx
\]

\[
= - \log 4 + \int p_{\text{data}}(x) \log \left( \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right) dx + \int p_g(x) \log \left( \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right) dx
\]

\[
= - \log 4 + D_{KL} \left( p_{\text{data}} \mid \frac{p_{\text{data}} + p_g}{2} \right) + D_{KL} \left( p_g \mid \frac{p_{\text{data}} + p_g}{2} \right)
\]

In the above equation,

\[
D_{KL}(p \mid q) = \int x p(x) \log \frac{p(x)}{q(x)} dx
\]

denotes the Kullback-Leibler divergence, which can be seen as a distance function for probability distributions. It is always non-negative and zero if and only if \(p = q\). Thus, both KL-divergences above are non-negative and zero if and only if \(g = g^*\) in both cases. Thus, \(g^*\) is the unique global minimum.