Assignment 1 (ML with noisy labels). We want to learn a binary classifier \( q(k \mid x; \theta) \) with classes \( k = \pm 1 \). It is defined as a neural network with parameters \( \theta \) and with the sigmoid logistic distribution in the output.

The true labels \( k_i \) of the images \( x_i \) are however unknown. Instead we are given training pairs \((x_i, t_i)\) with “noisy labels” \( t_i = \pm 1 \). They might have been incorrectly assigned by the person who annotated the data. More specifically, let us assume that the label \( t_i \) is correct \((t_i = k_i)\) with probability \( 1 - \varepsilon \) and incorrect \((t_i = -k_i)\) with probability \( \varepsilon \).

a) Formulate the conditional maximum likelihood learning of the parameters \( \theta \).

*Hint*: the conditional likelihood of the training data sample \((x_i, t_i)\) is obtained by marginalizing over the unknown true label

\[
p(t_i \mid x_i) = \sum_{k \in \{-1, 1\}} p(t_i \mid k) q(k \mid x_i; \theta),
\]

where \( p(t \mid k) \) is the labelling noise model.

b) A popular practical solution is to minimize the cross-entropy loss

\[
-\sum_i \sum_k p_i(k) \log q(k \mid x_i; w),
\]

where \( p_i(k) \) denote "softened 1-hot labels": \( p_i(k) = 1 - \varepsilon \) for \( k = t_i \) and \( \varepsilon \) otherwise. Prove that the negative cross-entropy (1) is a lower bound of the log likelihood in a). Use Jensen’s inequality for \( \log \).

Assignment 2. Let \( q(x) \) and \( p(x) \) be two factorizing probability distributions for random vectors \( x \in \mathbb{R}^n \), i.e.

\[
p(x) = \prod_{i=1}^{n} p(x_i) \quad \text{and} \quad q(x) = \prod_{i=1}^{n} q(x_i).
\]

Prove that their KL-divergence decomposes into a sum of KL-divergences for the components, i.e.

\[
D_{KL}(q(x) \parallel p(x)) = \sum_{i=1}^{n} D_{KL}(q(x_i) \parallel p(x_i))
\]

Assignment 3. Compute the KL-divergence of two univariate normal distributions.

Assignment 4 (Smooth AP). In this exercise we will see the relation between average precision and triplet loss and consider also an alternative smoothing technique.
Let \( a \) be a given anchor or query. Let \( P \) be the set of all positive examples for \( a \) and \( N \) be the set of all negative examples (so that \( P \cup N \) is a disjoint partition of the whole dataset). Let \( d_x = d(f(a), f(x)) \) be the distance in the feature space between the anchor \( a \) and another example \( x \in P \cup N \). It can be shown that the average precision defined in Lab 6 can be expressed as

\[
AP = 1 - \frac{1}{T} \sum_{p \in P} \frac{1}{k(p)} \sum_{n \in N} [d_n < d_p],
\]

where \( T = |P| \) is the total number of positive examples and \( k(p) = \sum_{x \in P \cup N} [d_x \leq d_p] \). In this expression the inner sum counts the number of negative examples which have a smaller distance to the query than \( p \), i.e. they will be incorrectly listed earlier in a sorted list of retrieved items. The function \( k(p) \) expresses the position of \( p \) in the sorted list of all examples. Efficiently \( 1/k(p) \) gives a higher relative weights to errors in the beginning of the retrieval list and discounts errors towards the end of the retrieval list.

\[a)\] Can we train a neural network \( f \) by gradient descent to maximize \( AP \) directly?

\[b)\] Consider minimizing just \( \sum_{p \in P} \sum_{n \in N} [d_n < d_p] \), i.e. ignoring the weights \( 1/k(p) \). What is the relation between this objective and the triplet loss \( l(a) \) proposed in the lab?

\[c)\] Another variant to make the function \( [d_n < d_p] \) differentiable is to replace it by some smooth function. Let’s give it a latent variable interpretation. Assume \( Z \) is a noise with logistic distribution with scale \( \tau \). Consider the loss with injected noise \( Z \) in each term, modeling imprecise descriptors:

\[
[d_n - d_p + Z < 0].
\]

Compute its expectation in \( Z \).

Applying such smoothing to all indicator functions in (2), including those occurring in \( k(p) \) results in the method of A. Brown et al.: Smooth-AP: Smoothing the Path Towards Large-Scale Image Retrieval (2020).