From Label Smoothing to Label Relaxation*
- Supplementary Material -

Julian Lienen, Eyke Hüllermeier
Heinz Nixdorf Institute and Department of Computer Science
Paderborn University, Germany
{julian.lienen,eyke}@upb.de

A Appendix

A.1 Analytical Determination of Equation (5)

Proposition 1. *Let $Q^\alpha := Q_i^\alpha$ be the target set as defined in Equation (3) for the true class $y := y_i \in \mathcal{Y}$. Moreover, let $\hat{p} \notin Q^\alpha$ be a probability distribution over the classes $\mathcal{Y}$ and L^* denote the label relaxation loss as defined in Equation (4). Then, $p^r \in \min_{p \in Q^\alpha} D_{KL}(p||\hat{p})$.*

Proof. The statement of this proposition can also be rephrased as follows: $\forall p \in Q^\alpha$ with $p \neq p^r$, $L^*(p, \hat{p}) \geq L^*(p^r, \hat{p})$. To prove the statement, let us assume that $L^*(p, \hat{p}) < L^*(p^r, \hat{p})$, which would be a contradiction.

We know that

$$\begin{aligned}
L^*(p^r, \hat{p}) &= (1-\alpha) \log \frac{1-\alpha}{\hat{p}(y)} + \sum_{y' \in \mathcal{Y}: y' \neq y} \frac{\alpha \cdot \hat{p}(y')}{\sum_{\hat{y} \in \mathcal{Y}: \hat{y} \neq y} \hat{p}(\hat{y})} \log \frac{\frac{\alpha \cdot \hat{p}(y')}{\sum_{\hat{y} \in \mathcal{Y}: \hat{y} \neq y} \hat{p}(\hat{y})}}{\hat{p}(y')} \\
&= (1-\alpha) \log \frac{1-\alpha}{\hat{p}(y)} + \sum_{y' \in \mathcal{Y}: y' \neq y} \frac{\alpha \cdot \hat{p}(y')}{1-\hat{p}(y)} \log \frac{\alpha}{1-\hat{p}(y)} \\
&= (1-\alpha) \log \frac{1-\alpha}{\hat{p}(y)} + \frac{\alpha}{1-\hat{p}(y)} \log \frac{\alpha}{1-\hat{p}(y)} \sum_{y' \in \mathcal{Y}: y' \neq y} \hat{p}(y') \\
&= (1-\alpha) \log \frac{1-\alpha}{\hat{p}(y)} + \frac{\alpha \cdot (1-\hat{p}(y))}{1-\hat{p}(y)} \log \frac{\alpha}{1-\hat{p}(y)} \\
&= (1-\alpha) \log \frac{1-\alpha}{\hat{p}(y)} + \alpha \log \frac{\alpha}{1-\hat{p}(y)} \quad .
\end{aligned}$$

*To appear in *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, February 2021.

Thus, the loss simplifies to the Kullback-Leibler divergence on the two cases of the class being y or any other class. Without loss of generality, we will use this simplification in the further course.

Together with

$$L^*(p,\hat{p}) = p(y)\log\frac{p(y)}{\hat{p}(y)} + \sum_{y'\in\mathcal{Y}:y'\neq y} p(y')\log\frac{p(y')}{\hat{p}(y')} \ ,$$

we get

$$p(y)\log\frac{p(y)}{\hat{p}(y)} + \sum_{y'\in\mathcal{Y}:y'\neq y} p(y')\log\frac{p(y')}{\hat{p}(y')} - (1-\alpha)\log\frac{1-\alpha}{\hat{p}(y)} - \alpha\log\frac{\alpha}{1-\hat{p}(y)} < 0 \ .$$

Using the log sum inequality[1], we infer

$$\begin{aligned}\sum_{y'\in\mathcal{Y}:y'\neq y} p(y')\log\frac{p(y')}{\hat{p}(y')} &\geq \left(\sum_{y'\in\mathcal{Y}:y'\neq y} p(y')\right)\log\frac{\sum_{y'\in\mathcal{Y}:y'\neq y} p(y')}{\sum_{y'\in\mathcal{Y}:y'\neq y}\hat{p}(y')} \\ &= (1-p(y))\log\frac{1-p(y)}{1-\hat{p}(y)} \ .\end{aligned}$$

Due to $\hat{p}(y) < (1-\alpha)$ and $p(y) \geq (1-\alpha)$, we get

$$\begin{aligned}& p(y)\log\frac{p(y)}{\hat{p}(y)} + (1-p(y))\log\frac{1-p(y)}{1-\hat{p}(y)} - (1-\alpha)\log\frac{1-\alpha}{\hat{p}(y)} - \alpha\log\frac{\alpha}{1-\hat{p}(y)} \\ =\ & p(y)\log p(y) - p(y)\log\hat{p}(y) + (1-p(y))\log(1-p(y)) - (1-p(y))\log(1-\hat{p}(y)) \\ & \quad - (1-\alpha)\log(1-\alpha) + (1-\alpha)\log\hat{p}(y) - \alpha\log\alpha + \alpha\log(1-\hat{p}(y)) \\ =\ & p(y)\log p(y) + (1-p(y))\log(1-p(y)) - (1-\alpha)\log(1-\alpha) - \alpha\log\alpha \\ & \quad + \underbrace{[(1-\alpha)-p(y)]}_{\leq 0}\log\underbrace{\hat{p}(y)}_{<(1-\alpha)} + \underbrace{[\alpha-(1-p(y))]}_{\geq 0}\log\underbrace{(1-\hat{p}(y))}_{>\alpha} \\ >\ & p(y)\log p(y) + (1-p(y))\log(1-p(y)) - (1-\alpha)\log(1-\alpha) - \alpha\log\alpha \\ & \quad + [(1-\alpha)-p(y)]\log(1-\alpha) + [\alpha-(1-p(y))]\log\alpha \\ =\ & p(y)\log p(y) + (1-p(y))\log(1-p(y)) \\ & \quad + [(1-\alpha)-p(y)-(1-\alpha)]\log(1-\alpha) + [\alpha-(1-p(y))-\alpha]\log\alpha \\ =\ & p(y)\log p(y) + (1-p(y))\log(1-p(y)) - p(y)\log(1-\alpha) - (1-p(y))\log\alpha\end{aligned}$$

With Gibb's inequality[2], we can infer

$$\begin{aligned}&[p(y)\log p(y) + (1-p(y))\log(1-p(y))] \\ &\qquad - [p(y)\log(1-\alpha) + (1-p(y))\log\alpha] \geq 0 \ ,\end{aligned}$$

which contradicts the assumption. □

[1] For $a_1,\ldots,a_n,b_1,\ldots,b_n \in \mathbb{R}_+$, the log sum inequality states that $\sum_{i\in[n]} a_i\log\frac{a_i}{b_i} \geq a\log\frac{a}{b}$, whereby $a := \sum_{i\in[n]} a_i$ and $b := \sum_{i\in[n]} b_i$.

[2] Gibb's inequality states that for two arbitrary discrete probability distributions $q_1, q_2 \in \mathbb{P}(\mathcal{S})$ over a discrete set $\mathcal{S}$, $\sum_{s\in\mathcal{S}} q_1(s)\log q_1(s) \geq \sum_{s\in\mathcal{S}} q_1(s)\log q_2(s)$ holds.

Proposition 2. *Let Q^α be defined as in Proposition 1 for the true class $y \in \mathcal{Y}$. Furthermore, let $\hat{p}$ be an arbitrary probability distribution over $\mathcal{Y}$. Then, using the Kullback-Leibler divergence D_{KL} as loss L within L^*, the loss as defined in Equation (4) is given by Equation (5).*

Proof. To prove the proposition, one can consider the two cases whether $\hat{p}$ is in Q^α or not. Let $p, q \in \mathbb{P}(\mathcal{Y})$ be any arbitrary probability distributions over the class space $\mathcal{Y}$. Then, the divergence properties of D_{KL} yield $D_{KL}(p||q) = 0$ if and only if $p = q$, as well as $D_{KL}(p||q) \geq 0$, $\forall\, p, q \in \mathbb{P}(\mathcal{Y})$.

For the first case, namely $\hat{p} \in Q^\alpha$, the first property of D_{KL} delivers a zero loss for $\hat{p}$ taken as target within Q^α. The latter case is an implication of Proposition 1. Hence, $L^*(Q^\alpha, \hat{p}) = D_{KL}(p^r||\hat{p})$ for $\hat{p} \notin Q^\alpha$. □

Proposition 3. *Given the fixed target set Q^α for the true class $y \in \mathcal{Y}$ as defined before, $L^*(Q^\alpha, \hat{p})$ is convex with regard to its second parameter $\hat{p}$, i.e.,*

$$\forall p_1, p_2 \in \mathbb{P}(\mathcal{Y}), \forall \lambda \in [0,1]: \\ L^*(Q^\alpha, \lambda p_1 + (1-\lambda)p_2) \leq \lambda L^*(Q^\alpha, p_1) + (1-\lambda) L^*(Q^\alpha, p_2) \ .$$

Proof. Let p_1, p_2 and λ be fixed, and let $\tilde{p} = \lambda p_1 + (1-\lambda)p_2$ denote the considered combination of p_1 and p_2. Then, in order to prove the statement, one can distinguish three cases:

Case 1: $p_1, p_2 \in Q^\alpha$. Due to $\lambda \in [0,1]$, we have $\tilde{p} \in Q^\alpha$ and thus, per definition, $L^*(Q^\alpha, \tilde{p}) = 0$. With the non-negativity of the Kullback-Leibler divergence, we can infer $L^*(Q^\alpha, \cdot) \geq 0$, which implies the convexity.

Case 2: $p_1, p_2 \notin Q^\alpha$. As a consequence, $\tilde{p} \notin Q^\alpha$ holds. With the application of the log sum inequality, we can follow:

$$\begin{aligned}
L^*(Q^\alpha, \tilde{p}) = \ & (1-\alpha)\log\frac{1-\alpha}{\tilde{p}(y)} + \alpha\log\frac{\alpha}{1-\tilde{p}(y)} \\
= \ & [\lambda(1-\alpha) + (1-\lambda)(1-\alpha)]\log\frac{\lambda(1-\alpha)+(1-\lambda)(1-\alpha)}{\lambda p_1(y) + (1-\lambda)p_2(y)} \\
& + [\lambda\alpha + (1-\lambda)\alpha]\log\frac{\lambda\alpha + (1-\lambda)\alpha}{\lambda(1-p_1(y)) + (1-\lambda)(1-p_2(y))} \\
\leq \ & \lambda(1-\alpha)\log\frac{\lambda(1-\alpha)}{\lambda p_1(y)} + (1-\lambda)(1-\alpha)\log\frac{(1-\lambda)(1-\alpha)}{(1-\lambda)p_2(y)} \\
& + \lambda\alpha\log\frac{\lambda\alpha}{\lambda(1-p_1(y))} + (1-\lambda)\alpha\log\frac{(1-\lambda)\alpha}{(1-\lambda)(1-p_2(y))} \\
= \ & \lambda\left[(1-\alpha)\log\frac{1-\alpha}{p_1(y)} + \alpha\log\frac{\alpha}{1-p_1(y)}\right] \\
& + (1-\lambda)\left[(1-\alpha)\log\frac{1-\alpha}{p_2(y)} + \alpha\log\frac{\alpha}{1-p_2(y)}\right] \\
= \ & \lambda L^*(Q^\alpha, p_1) + (1-\lambda)L^*(Q^\alpha, p_2)
\end{aligned}$$

Case 3: Either $p_1 \in Q^\alpha$ and $p_2 \notin Q^\alpha$ or vice versa (the proof is analogous). Obviously, $\tilde{p}$ can be an element of Q^α or not.

For the first case, namely that $\tilde{p} \in Q^\alpha$, we have $L^*(Q^\alpha, \tilde{p}) = 0$ and

$$\lambda L^*(Q^\alpha, p_1) + (1-\lambda) L^*(Q^\alpha, p_2) = (1-\lambda) L^*(Q^\alpha, p_2) \geq 0 \ .$$

For the latter case, we know that $L^*(Q^\alpha, \tilde{p})$ simplifies to the KL divergence D_{KL} on the two cases whether the class is y or any other class, whereby $D_{KL}(q||\cdot)$ is known to be convex for a fixed distribution q and non-negative. Then, with $1-\alpha$ being the lower bound probability for y of distributions in Q^α, $L^*(Q^\alpha, \tilde{p})$ converges to 0 for $\tilde{p}(y) \to 1-\alpha$.

Taking both cases together, $L^*(Q^\alpha, \cdot)$ comes down to the (convex) KL divergence for distributions $\tilde{p} \notin Q^\alpha$ up to the boundary of $\tilde{p}(y) = 1-\alpha$, where the loss then degenerates to 0 in a continuous manner. As a consequence, the loss is convex for any $\tilde{p}$ in this case. □

A.2 Experimental Details

In the following, a more comprehensive overview over the experimental setting and chosen hyperparameters within our empirical studies is presented. In all experiments, SGD was used as optimizer with a Nesterov momentum of 0.9, while the batch size was set to 64. The experimental runs were conducted on 10 NVIDIA RTX 2080 Ti and 20 NVIDIA GTX 1080 Ti.

A.2.1 Simple Dense Architecture

For MNIST and Fashion-MNIST, a simple neural network with two hidden layers, each consisting of 1024 ReLu-activated neurons, was used. In the case of training on MNIST, 25 epochs were used, while the model training on Fashion-MNIST lasted for 50 epochs. Here, an initial learning rate of 0.05 has been used for both datasets, which was multiplied by $\sqrt{0.1}$ after 30 epochs for Fashion-MNIST. No further regularization was applied.

A.2.2 ResNet

To train models using the ResNet-56 (V2) architecture, our implementation is based on the publicly available code in [3]. Due to the model size, we augmented the training data as described in Subsection A.2.5. For CIFAR-10, we trained for 200 epochs with an initial learning rate of 0.1 multiplied by $\sqrt{0.1}$ after 50, 100 and 150 epochs. For CIFAR-100, 300 epochs were trained with the same initial learning rate multiplied by $\sqrt{0.1}$ after 75, 150 and 225 epochs. For this architecture, no additional Dropout layers were used within our experiments, while BatchNormalization followed each convolutional layer of the model.

A.2.3 VGG

For VGG16, we adopted the implementation as provided in [4], where some minor adaptions in order to be able to derive reasonable models for CIFAR-10 and

[3] `https://keras.io/examples/cifar10_resnet/`
[4] `https://keras.io/api/applications/vgg/`

CIFAR-100 were made. Due to the high amount of parameters, we were required to add BatchNormalization and Dropout (ranging from $p = 0.3$ to $p = 0.5$ for increasing depth) to the architecture. Furthermore, we used augmented data as described in Subsection A.2.5. For all losses, we used the same techniques and parameters. The initial learning rate was set to 0.01 in all experiments. To induce models on CIFAR-10, we trained for 200 epochs, whereby we divided the learning rate by 10 after 50, 100 and 150 epochs. In the case of CIFAR-100, our model is trained for 300 epochs with the learning rate divided by 10 after 150 and 225 epochs.

A.2.4 DenseNet

Within our experiments, we used the more efficient variant DenseNet-BC with a depth of 100 and a growth rate of 12 based on the original implementation as provided in [5]. For both CIFAR-10 and CIFAR-100, we used the same augmentation scheme as before, while we did not use Dropout in our runs. In both dataset cases, we used an initial learning rate of 0.1. While we trained on CIFAR-10 for 200 epochs with dividing the learning rate by 10 after 50, 100 and 150 epochs, we used 300 epochs for CIFAR-100 with the same divisor after 150 and 225 epochs.

A.2.5 Preprocessing and Data Augmentation

For preprocessing, we used a rather simple approach: Each image pixel is divided by 255 to get values in $[0, 1]$, followed by subtracting the pixel means of the training data. Furthermore, as described for the individual architectures, data augmentation is used for some models as an additional regularization mean. When applied, we used the implementation of [6], where we flipped horizontally and shifted in width and height (each with a fraction of 0.1).

[5] `https://github.com/liuzhuang13/DenseNet`

[6] `https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator`

A.3 Barycentric Loss Visualization

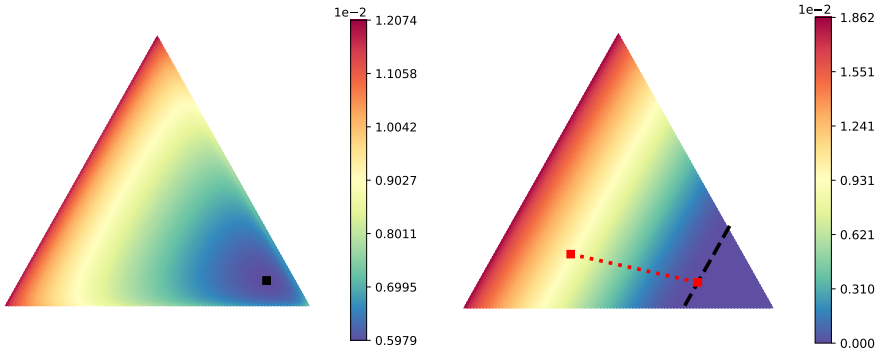

Figure 1: Barycentric visualizations of label smoothing (left plot) and L^* based on the Kullback-Leibler divergence (right plot) for three classes with $\alpha = 0.25$: The lower right corner represents the degenerate (one-point) distribution for the true target. In the case of label smoothing, the black point represents the smoothed target. For L^*, the border of the induced set Q^α (cf. Equation (3)) is sketched by the black dashed line. The red points indicate an arbitrary prediction outside Q^α (left point), which is projected onto the set boundary according to its distribution (right point).