Self-certified Tuple-Wise Deep Learning

Sijia Zhou[1(✉)], Yunwen Lei[2], and Ata Kabán[1]

[1] University of Birmingham, Birmingham B15 2TT, UK
sxz115@student.bham.ac.uk, a.kaban@bham.ac.uk
[2] University of Hong Kong, Pokfulam, Hong Kong, China
leiyw@hku.hk

Abstract. Tuple-wise learning maps a tuple of input points to a label. A typical application is object re-identification, for which empirically successful algorithms have been recently proposed. However, individual tuples do not bring independent information, as their component points participate in multiple tuples. Hence, one may expect needing a larger sample size to learn effectively. To make the most of the available labelled tuples, we turn to the idea of learning with self-certification based on PAC-Bayes bounds. While existing results are not applicable directly to our case, we generalize the self-certified learning paradigm to tuple-wise neural networks, by using U-statistics. The obtained new PAC-Bayes bound confirms the increasing sample complexity for tuple-wise learning as a function of the tuple size. We then conduct an empirical study to evaluate the tuple-wise objective functions obtained from the bound. As an illustrative example, we train the PAC-Bayes posterior distribution of a stochastic neural network using pairwise stochastic gradient descent. Our results demonstrate non-vacuous risk bounds in tuple-wise deep learning on the task of person re-identification (Re-ID), using several real-world datasets.

Keywords: Tuple-wise Learning · PAC-Bayes · Generalization

1 Introduction and Related Work

Tuple-wise learning aims to learn a mapping from an input tuple consisting of $m \geq 2$ examples to an output. There has been considerable recent interest in applications of tuple-wise learning, starting with pairwise problems such as metric learning, ranking, AUC maximization, and object recognition based on their similarity [1,7,9,10]. In metric learning the aim is to learn an appropriate distance or similarity to compare pairs of examples. This can be applied to problems such as face verification and person re-identification (Re-ID) [7,15,18,37], which have been further enhanced by deploying powerful neural networks [38,41]. Beyond pairwise loss functions, various triplet-losses have been proposed [25]. For instance, if positive pairs and negative pairs are compared jointly, a triplet loss can distinguish the similarity more efficiently [34]. By considering

A. Bifet et al. (Eds.): ECML PKDD 2024, LNAI 14942, pp. 303–320, 2024.
https://doi.org/10.1007/978-3-031-70344-7_18

the distance between positive pairs or negative pairs, some works use contrastive loss [16,22] and verification loss [25].

The empirical performance gain obtained with such triplet losses then motivated the idea of tuple-wise loss functions [39] in the hope to capture and more fully exploit the information contained in the relative position of a query relative to a larger subset of the data. However, these newer developments, beyond the pairwise case, have not been theoretically analysed, and it is unclear to what extent increasing the size of the input tuple to the learning algorithm will yield accuracy improvements. The issue is that, larger and larger tuples bring less and less independent information to the learner, as their constituent points participate in multiple other tuples. Therefore, we want to make the most of the available labelled data.

In this work, we address the above issue by considering self-certified learning based on the PAC-Bayesian framework. Our motivation is that self-certified learning makes it possible to use the whole data for training a predictor, as it also gives an upper bound on the risk of the obtained predictor [30], removing the need for a validation set. It can be generalized to tuple-wise deep learning and trains the distribution of stochastic neural networks using tuple-wise stochastic gradient descent (SGD). Here, we consider the application of pairwise learning in metric learning. We use convolutional siamese networks to extract features from tuple-wise inputs and evaluate their similarities [41].

Our contributions are summarized as follows:

- We analyze the generalization performance of tuple-wise learning, which can be applied to neural networks, yielding a computable generalization bound that also highlights the effect of the tuple size.
- We devise a new tuple-wise learning algorithm that implements the above generalization guarantee, extending a recent trend of self-certified learning from point-wise to tuple-wise case. We do this to mitigate the diminishing returns of increasing the tuple size, as self-certified learning allows us to use all available labelled data for training.
- We apply and test our algorithm in several Re-ID datasets, starting with pairwise inputs, demonstrating that the previously reported advantages of self-certified learning carry over to this setting.

The remainder of the paper is organized as follows. We briefly review the related work on PAC-Bayes generalization analysis in Sect. 1.1. We discuss the use of U-statistics and our notations in Sect. 2. Our main results are presented in Sect. 3. We conduct experiments in the special case of metric learning based on Re-ID, using an algorithm obtained from our PAC-Bayesian analysis in Sect. 4, and conclude the paper in Sect. 5.

1.1 Related Work

The PAC-Bayes framework of generalization analysis is one of the most successful algorithm-dependent approaches currently. One may specify a prior distribution

and a parametrised posterior distribution over the parameter domain, inducing distributions over predictors. The prior must not depend on the training sample, but it may be learned from an independent sample for increased tightness [30]. The PAC-Bayes bounds obtained through the framework hold uniformly for all posterior distributions of the specified form. Techniques date back to works by Shawe-Taylor, McAllester [26,36], further improved by Catoni [8] and others. It should be noted that the terms 'prior' and 'posterior' have a different meaning in the PAC-Bayes bounding context than in Bayesian inference.

The typical use of PAC-Bayes methodologies is to bound the generalization error of stochastic learning algorithms, with computable data-dependent quantities: the empirical risk, and a divergence between the PAC-Bayes prior and posterior, most commonly the Kullback-Leibler (KL) divergence or the Rényi divergence [4].

The PAC-Bayes theory has been developed to give non-vacuous bounds for neural networks [13,30]. Tighter numerical bounds are computed based on the learned data-dependent prior [2,12,14,27,28,33], and the suggestive term 'self-certified learning' has been coined [29,30] to emphasize the remarkable ability of these analyses to yield learning algorithms with generalization guarantees estimated on the training sample, without the need of a hold-out set.

However, all of these works assume i.i.d. examples, and hence do not apply to non-i.i.d. settings such as tuple-wise, or even pairwise learning. In non-i.i.d. settings, [32] gave PAC-Bayes bounds by using fractional covers, which allows for handling the statistical dependencies within a dataset. Their work is fairly complex, in turn we will simply exploit the observation that the tuple-wise empirical risk resembles the form of U-statistics.

One of the most well-known examples of tuple-wise learning is pairwise learning, such as metric learning. Traditional approaches to metric learning involve two separate steps: feature extraction and metric learning. A specific method is learning a Mahalanobis metric [3,18,37]. The traditional metric learning methods have been applied to practical problems, including face verification and Re-ID [15,18,37]. With the development of deep learning methods, the siamese neural network was introduced [6], which accomplishes feature learning simultaneously with learning a metric. This architecture contains two models with shared weights, and it was further used in deep metric learning [38,41]. The siamese network is trained to extract feature vectors of the inputs, as well as learn the similarity between input pairs. Based on their similarity, this approach can be used to determine whether two images contain the same class or not.

2 Preliminaries

Let $\mathcal{X}$ by an input domain, and $\mathcal{Y}$ a set of labels. We will use the notations $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, and $S = \{z_1, \ldots, z_n\}$ drawn i.i.d. from some unknown distribution $\mathcal{D}$ defined on $\mathcal{Z}$.

In classical pointwise learning, S is a training set and the goal is to learn a hypothesis that maps an element of $\mathcal{X}$ to an element of $\mathcal{Y}$. In turn, in m-tuple-wise learning, the aim is to learn a hypothesis of the form $h_{\mathbf{w}} : \mathcal{X}^m \to \{0, 1\}$, i.e.

a mapping from m elements of $\mathcal{X}$ (an m-tuple) to an element of $\{0, 1\}$. The label of a tuple is a function of the individual labels of the m inputs. Take pairwise learning as an example ($m = 2$) – an input pair $x_i, x_j, i \neq j$, is assigned label 1 if $y_i = y_j$ and label 0 otherwise.

We apply a learning algorithm $A : \mathcal{Z}^n \to \mathcal{H}$ on the labelled training tuples to obtain a predictor (hypothesis) $h_{\mathbf{w}} \in \mathcal{H}$ parameterised by $\mathbf{w} \in \mathcal{W}$, where $\mathcal{W} \subseteq \mathbb{R}^d$ is a parameter domain and $\mathcal{H} = \{h_{\mathbf{w}} | \mathbf{w} \in \mathcal{W}\}$ is the hypothesis set.

To measure the quality of predictions of a predictor $h_{\mathbf{w}}$, we will use the 0–1 loss function $\ell : \mathcal{H} \times \mathcal{Z}^m \to \{0, 1\}$ on m-tuple examples, rather than a single example as in the classic pointwise case. For tuple-wise learning, the generalization error is defined as the expected loss of the learned predictors applied on an unseen tuple of inputs drawn from $\mathcal{D}^m$. The risk is defined as

$$R(h_{\mathbf{w}}) = \mathbb{E}_{z_1,\ldots,z_m \sim \mathcal{D}^m}[\ell(h_{\mathbf{w}}, z_1, \ldots, z_m)]. \tag{1}$$

We approximate the true risk with the empirical risk instead

$$R_S(h_{\mathbf{w}}) = \frac{1}{C(n,m)} \sum_c \ell(h_{\mathbf{w}}, z_{i_1}, \ldots, z_{i_m}), \tag{2}$$

for $m \geq 2$, $n \geq m$, and $C(n, m) := \binom{n}{m}$, where $c = \{i_1, \ldots, i_m\} \subset \{1, \ldots, n\}$ and the summation runs over all the $C(n, m)$ combinations of m different sample indices.

In order to use the PAC-Bayes analysis methodology, we define a PAC-Bayes prior distribution $\mathbb{P}$ on $\mathcal{W}$ before seeing the training data. We also define a parametrised PAC-Bayes posterior distribution $\mathbb{Q}$ whose parameters are allowed to depend on the training data. The expected (true) risk w.r.t the posterior is a measure of the quality of a predictor, and is defined as

$$\Re(\mathbb{Q}) = \mathop{\mathbb{E}}_{\mathbf{w}\sim\mathbb{Q}}[R(h_{\mathbf{w}})]. \tag{3}$$

Its empirical counterpart is $\Re_S(\mathbb{Q}) = \mathop{\mathbb{E}}_{\mathbf{w}\sim\mathbb{Q}}[R_S(h_{\mathbf{w}})]$.

In sequel, we analyze the generalization problems, by upper bounding the difference $\Re(\mathbb{Q}) - \Re_S(\mathbb{Q})$.

2.1 Tuple-Wise Learning via U-Statistics

The difficulty posed by working with the tuple-wise empirical loss in Eq. (2) is that the tuples are no longer independent of each other, even if the individual points are i.i.d. Instead, $R_S(h_{\mathbf{w}})$ is an m-th order U-statistic.

There is a basic and useful representation of U-statistics [35] in terms of the average of (dependent) random variables

$$U_n = \frac{1}{C(n,m)} \sum_c \ell(h_{\mathbf{w}}, z_{i_1}, \ldots, z_{i_m}) = \frac{1}{n!} \sum_\pi \frac{1}{\lfloor \frac{n}{m} \rfloor} \sum_{i=1}^{\lfloor \frac{n}{m} \rfloor} \ell(h_{\mathbf{w}}, z_{\pi(i_i)}, \ldots, z_{\pi(i_m)}), \tag{4}$$

where π runs through all permutations of $\{1, \ldots, n\}$. The reason this is useful is that the inner summation on the r.h.s. is of i.i.d. random variables. A special case of this representation was used to extend bounds from pointwise to pairwise learning in the literature (see e.g. [21,32]). It will play a key role in obtaining our results, in particular Lemma 4.

3 Main Results

In Theorem 1 we give our main result. We first introduce two useful lemmas, which are the tuple-wise extensions of Lemma 5.2 and Lemma 5.3 in [20]. Our proof extends the classic PAC-Bayes bound (see e.g. [20]) to tuple-wise learning, considering the non i.i.d. nature of the problem. The proof is given in Appendix A. We denote by r_S the observed number of errors on the training set S for the hypothesis $h_{\mathbf{w}}$, defined as $r_S := C(n,m)R_S(h_{\mathbf{w}})$. For $p, q \in (0,1)$, we will define $\mathrm{KL}_+(q\|p) := q \log \frac{q}{p} + (1-q) \log \frac{1-q}{1-p}$ for $p > q$ and 0 otherwise.

Lemma 1. *For all $\mathcal{D}$, any $S, S' \sim \mathcal{D}^n$, for all $\mathbb{P}$ and any $\delta \in (0,1]$, we have*

$$\Pr_{S \sim \mathcal{D}^n} \left\{ \mathbb{E}_{\mathbf{w} \sim \mathbb{P}} \frac{1}{\Pr_{S' \sim \mathcal{D}^n}\{r_S = r_{S'}\}} \leq \frac{C(n,m)+1}{\delta} \right\} \geq 1 - \delta.$$

Lemma 2. *For any datasets $S, S' \sim \mathcal{D}^n$ and all $\mathbb{Q}$, we have*

$$\lfloor n/m \rfloor \cdot \mathrm{KL}_+(\Re(\mathbb{Q}) || \Re_S(\mathbb{Q})) \leq \mathbb{E}_{\mathbf{w} \sim \mathbb{Q}} \ln \left(\frac{1}{\Pr_{S' \sim \mathcal{D}^n}\{r_S = r_{S'}\}} \right).$$

Based on the above lemmas, we obtain the tuple-wise PAC-Bayes bound [32].

Theorem 1. *For any prior $\mathbb{P}$ independent of S and any $\delta \in (0,1]$, with probability at least $1-\delta$, we have*

$$\forall \mathbb{Q}, \mathrm{KL}_+ \left(\Re(\mathbb{Q}) \| \Re_S(\mathbb{Q}) \right) \leq \frac{1}{\lfloor n/m \rfloor} \left(\mathrm{KL}(\mathbb{Q}\|\mathbb{P}) + \ln \frac{C(n,m)+1}{\delta} \right). \tag{5}$$

Remark 1. Theorem 1 implies that sample complexity increases with the tuple size. Indeed, considering m tuple-wise learning from n i.i.d. points – then, according to Theorem 1, these n points only amount to $\lfloor n/m \rfloor$ independent inputs.

Next, we consider deriving the tuple-wise objective function based on the main result for stochastic optimization.

3.1 Optimization of the Bound w.r.t. Posterior $\mathbb{Q}$

We will convert the bound Eq. (5) into an easily computable objective function, as it will need to be evaluated repeatedly during iterative optimization. For this purpose, we recall Pinsker's inequality, for $p > q$, $\mathrm{KL}_+(q\|p) \geq 2(p-q)^2$. We use this to lower bound the KL_+ term on the LHS to isolate the expected risk $\Re(\mathbb{Q})$.

We will only use this during optimization, since Pinsker's inequality slightly loosens the bound. So for any $\delta \in (0,1)$, with probability at least $1-\delta$, we have

$$\forall \mathbb{Q},\ \Re(\mathbb{Q}) \leq \Re_S(\mathbb{Q}) + \frac{1}{2\lfloor n/m \rfloor}\left[\mathrm{KL}(\mathbb{Q}\|\mathbb{P}) + \ln \frac{C(n,m)+1}{\delta}\right]. \tag{6}$$

The $\mathrm{KL}(\mathbb{Q}||\mathbb{P})$ term will have a closed form with our choice of $\mathbb{P}, \mathbb{Q}$ distributions, discussed in Sect. 4.2. It remains to compute $\Re_S(\mathbb{Q})$, for which we use Monte Carlo sampling.

As in [30], we will employ a surrogate loss to avoid having to work with the 0–1 loss – e.g. the cross-entropy loss, which upper bounds the 0–1 loss. We denote by ℓ^{CE} the cross-entropy loss, and by $\Re_S^{CE}(\mathbb{Q})$ the associated expected empirical risk. Based on Eq. (6), we take the objective function f_{obj}

$$f_{obj} = \Re_S^{CE}(\mathbb{Q}) + \frac{1}{2\lfloor n/m \rfloor}\left[\mathrm{KL}(\mathbb{Q}\|\mathbb{P}) + \ln \frac{C(n,m)+1}{\delta}\right], \tag{7}$$

where $\Re_S^{CE}(\mathbb{Q}) = \mathop{\mathbb{E}}_{\mathbf{w}\sim\mathbb{Q}}[\frac{1}{C(n,m)}\sum_c \ell(h_{\mathbf{w}}, z_{i_1}, \ldots, z_{i_m})]$.

Remark 2. In practice, we will compute unbiased estimates of the gradients w.r.t. the above objective functions, as done by [5,30]. The optimization of the above objective function only involves the optimization of posterior $\mathbb{Q}$. Therefore, with our choice of prior and posterior distributions, we will discuss the corresponding estimates of the gradients in Sect. 4.2, using certain efficient optimization methods, e.g. stochastic gradients.

3.2 Plugging the Obtained $\mathbb{Q}$ Into Theorem 1

After the optimization process, we estimate the expected empirical risk $\Re_S(\mathbb{Q})$ by Monte Carlo sampling. The following Eq. (8) gives a convergence bound for an i.i.d. Monte Carlo average to approach the true expectation (Theorem 2.5 in [19]). For hypothesis $h_1, \ldots, h_k$ drawn i.i.d. from distribution $\mathbb{Q}$, we have Monte Carlo approximation $\hat{\mathbb{Q}}_k = \sum_{j=1}^{k} \delta_{h_j}$ for $\mathbb{Q}$. Then for 0–1 loss and any $\delta' \in (0,1)$, with probability at least $1-\delta'$ we have

$$\mathrm{KL}_+\left(\Re_S(\hat{\mathbb{Q}}_k)\|\Re_S(\mathbb{Q})\right) \leq \frac{\log\frac{2}{\delta'}}{k}. \tag{8}$$

We consider the inversion of the KL_+ term w.r.t. the second argument. This will upper bound the expected risk $\Re_S(\mathbb{Q})$. For $q \in [0,1]$, $p > q$, $a \geq 0$, we define

$$f^{kl}(q,a) = \sup\{p \in [0,1] : \mathrm{KL}_+(q\|p) \leq a\}.$$

In practice, we can use some numerical methods to compute f^{kl}. This gives the computable upper bound on the expected risk, which holds w.p. $1-\delta'$:

$$\Re_S(\mathbb{Q}) \leq f^{kl}\left(\Re_S(\hat{\mathbb{Q}}_k), \frac{\log\frac{2}{\delta'}}{k}\right). \tag{9}$$

Based on the inequalities above, we compute the final form of the risk bound as the following.

$$\Re(\mathbb{Q}) \leq f^{kl}\left(f^{kl}\left(\Re_S(\hat{\mathbb{Q}}_k), \frac{\log\frac{2}{\delta'}}{k}\right), \frac{1}{2\lfloor n/m \rfloor}\left[\mathrm{KL}(\mathbb{Q}\|\mathbb{P}) + \ln\frac{C(n,m)+1}{\delta}\right]\right). \tag{10}$$

4 Experiments with Application to Metric Learning

In this section, we illustrate the use of our PAC-Bayes bounds in an example application of tuple-wise learning in metric learning (i.e. with $m = 2$). It aims to keep examples that have the same label close to each other, while separating examples that have different labels. This can be applied to practical problems, including face verification and Re-ID [15,18,37].

Here we consider this using the siamese network [6]. This architecture contains two models with shared weights, and it was introduced in deep metric learning, and Re-ID [38,41]. We will use convolutional siamese networks, trained to learn feature vectors, as well as the similarity between them, in Re-ID tasks on several benchmark datasets. The architecture is a simplified version of the architecture employed in [41]. Our code is publicly available[1].

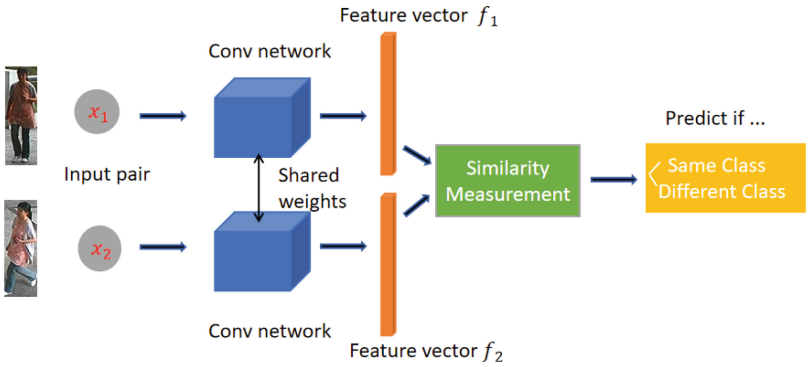


Fig. 1. Within the convolutional siamese networks structure, if an input pair is given, two models with shared weights extract feature embeddings f_1 and f_2 respectively. These outputs are used to evaluate the similarities between the input pair.

4.1 Dataset and Model

We train the convolutional siamese network on three Re-ID datasets, including CUHK01 [23], CUHK03 [24] and Market-1501 [40] (see Table 1). The CUHK01

[1] Code available at https://github.com/git0405/Self-certified-Tuple-wise.

dataset contains 3,884 images 971 persons, captured by two cameras. The CUHK03 dataset contains 14,097 images of 1,467 persons, captured by at most 5 cameras. The Market-1501 dataset contains 32,668 images of 1,501 persons, captured by at most 6 cameras.

We follow the instructions in [37,40,42] to divide the data into training and testing sets. Specifically, in the CUHK03 and Market-1501 dataset, images in the file 'bounding-box-train' are chosen for training, and images in the file 'bounding-box-test' are chosen for testing. The dataset CUHK01 is randomly divided in two subsets, so the test set is randomly selected using half of the available persons. We re-scale the images to size $3 \times 224 \times 224$ ($c \times h \times w$).

The siamese model is given in Fig. 1, built on the work of [25,41]. The models are trained to learn representations of the given pairs and the corresponding similarity. This framework has two CNN modules which extract features from the input data (x_i, x_j), sharing the same weights.

We use the ResNet-18 network [17] as the CNN module, pre-trained on ImageNet [11]. The outputs of the two CNN modules, each of size $512 \times 1 \times 1$, are flattened into two one-dimensional feature vectors (f_1 and f_2 in Fig. 1). The component-wise square difference between two feature vectors f_1, f_2 is fed into a fully-connected layer with softmax activation function, giving an estimate of the probability that the pair of input images represents the same person. We ensure the boundedness of the loss function, as required in our bounds, during training by restricting the minimum of the softmax outputs to a threshold (10^{-5}) in the same way as in [30].

Table 1. Properties of person Re-ID datasets used in this paper.

Dataset	CUHK01 [23]	CUHK03 [24]	Market-1501 [40]
Identities	971	146	1,501
Cameras	2	5	6
Train images	1,944	7,368	12,936
Test images	1,940	5,328	13,115

4.2 PAC-Bayes Bound Minimization Algorithm

In our PAC-Bayes bounds, we choose i.i.d. Gaussian distributions over weights, for both the prior and the posterior. Specifically, for Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with mean μ and variance $\sigma \geq 0$

$$\mathcal{N}(\mu, \sigma)(x) = (2\pi\sigma)^{-1/2} \exp\left(-\frac{(x-\mu)^2}{2\sigma}\right),$$

we have the KL divergence between two Gaussian distributions $\mathcal{N}(\mu_0, \sigma_0)$ and $\mathcal{N}(\mu, \sigma)$ as follows.

$$\mathrm{KL}\left(\mathcal{N}(\mu, \sigma) \| \mathcal{N}(\mu_0, \sigma_0)\right) = \frac{1}{2}\left(\log\left(\frac{\sigma_0}{\sigma}\right) + \frac{(\mu - \mu_0)^2}{\sigma_0} + \frac{\sigma}{\sigma_0} - 1\right).$$

This can be easily extended to multivariate case, as the KL divergence will still have a closed form. Other choices of prior and posterior with known closed form KL divergence include the Laplace distribution; this would suit situations where sparse connectivity is sought.

We will turn our bounds into learning algorithms by minimizing the bounds. To ensure that our bounds are tight for this purpose, we consider data-dependent priors. The priors are trained through empirical risk minimization (ERM) on a subset of the training data, as done by [28,30]. In practice, the training set is divided into a prior train set and a disjoint risk sampling set (i.e. there is no intersection between the prior train set and the risk sampling set). The latter is used to compute the risk bounds through Monte Carlo methods to approximate the expectation w.r.t. $\mathbb{Q}$ in the first term $\Re_S(\mathbb{Q})$ of the bound [28].

Algorithm 1. PAC-Bayes Pairwise Learning

Inputs: μ_0, ρ_0, $(z_i, z_j)_{i \neq j}^n$, δ, η and T
Optimize : μ, ρ
$\mu \leftarrow \mu_0$, $\rho \leftarrow \rho_0$, $t \leftarrow 1$
repeat
  Sample $\phi \sim \mathcal{N}(0, I)$
  Compute the random weights: $\mathbf{w} = \mu + \log(1 + \exp(\rho)) \odot \phi$
  Given the training objective function in Eq. (7) $f_{\text{obj}}\left((z_i, z_j), \mathbf{w}, \mu, \rho, \mu_0, \rho_0, \delta\right)$, use SGD to update μ and ρ, according to Eq. (11)
  $t = t + 1$
until $t > T$
return μ, ρ

We denote by μ_0 the trained weights of the priors. In practice, we set $\sigma = \log(1 + \exp(\rho))$ and update ρ during the process [13,30]. This gives non-negative trained σ. The variance σ_0 of priors is set to a constant [13] and this gives the constant ρ_0 based on the definition. the prior parameters μ_0 and σ_0 are used to initialise the mean and variance of the posterior Gaussian $\mathbb{Q}$.

To learn the posterior $\mathbb{Q}$, we consider unbiased estimates of the gradients w.r.t. the objective functions, as done by [5,30], as follows. We compute the gradients w.r.t. μ and σ. We sample the weights following the distribution $\mathcal{N}(\mu, \sigma)$, by computing $\mathbf{w} = \mu + \sigma \odot \phi$, where $\phi \sim \mathcal{N}(0, I)$ is sampled i.i.d. in every iteration. Based on this, we have the SGD update rule for parameters:

$$\mu = \mu - \eta\left(\frac{\partial f_{obj}}{\partial \mathbf{w}} + \frac{\partial f_{obj}}{\partial \mu}\right), \quad \rho = \rho - \eta\left(\frac{\partial f_{obj}}{\partial \mathbf{w}} \cdot \frac{\phi}{1 + \exp(-\rho)} + \frac{\partial f_{obj}}{\partial \rho}\right). \quad (11)$$

We denote the training pairs by $\{(z_i, z_j), i \neq j, i, j \in [n]\}$, the confidence parameter by $\delta \in (0, 1)$, learning rate by $\eta \in (0, 1)$ and number of iterations by T. Algorithm 1 gives the optimization of the pairwise PAC-Bayes bound.

4.3 Results

The training and the test set results on the datasets CUHK01, CUHK03 and Market-1501 are given in the Table 2. We have set $\delta = 0.025$ in Eq. (7), and $\delta' = 0.01, k = 150$ in Eq. (9). We use SGD with learning rate 0.01 and momentum 0.9 when training both the prior and the posterior. We run 40 epochs to learn the priors and 80 epochs for the posteriors. We experiment with σ_0 set to 0.005 and 0.01. The batch size is 16, and we use 50% of the training set for learning the priors. The parameter settings follow those in [13,25,30].

The learned data-dependent PAC-Bayes priors for each dataset are obtained with dropout-regularised ERM on 50% of three Re-ID training sets separately. The test errors w.r.t. the 0–1 loss of the siamese models with weights set to the mean of the trained prior on CUHK01, CUHK03 and Market-1501 are 0.1286, 0.1638 and 0.1298 respectively. We explored the prior distribution with σ_0 set to values in $\{0.03, 0.02, 0.01, 0.005, 0.004\}$ on CUHK01.

We present the results of PAC-Bayes objective function trained on three datasets. Table 2 shows the results with trained PAC-Bayes priors with different hyperparameters. We give the test set results according to three predictors according to posterior distribution $\mathbb{Q}$: the test error of a randomized predictor (Rand. pred.), which is drawn from $\mathbb{Q}$, the test error w.r.t. predictor which is the mean (Mean pred.) of $\mathbb{Q}$ weights; the test error of an aggregated predictor (Agg. pred.), comprising 100 predictors drawn from $\mathbb{Q}$ and averaged. For these different predictors, the results are similar across all datasets. This may suggest that the obtained predictors are comparable.

Trade-Off Between Empirical Risk and KL Terms. According to observations in previous work [14], the KL term will dominate the training objective. Therefore, we consider the data-dependent priors and penalty coefficient [5,30] to trade off KL term and empirical risk,

$$\mathrm{KL} \leftarrow \lambda \cdot \mathrm{KL} \tag{12}$$

However, according to the previous results, it is expected that the smaller the penalty coefficient, the empirical risk term receives more emphasis during training, which may result in larger KL divergence between posterior and prior, and consequently looser risk bound [5,30]. We test several values $\lambda \in [1, 0.1, 0.01]$. We did observe that a too small $\lambda = 0.001$ gives vacuous bound indeed due to the KL divergence term getting too large.

In Table 2 and Fig. 2 we present results on three datasets. We observe that the best setting of λ and σ_0 is problem-dependent – for instance on the Market-1501 data set $\lambda = 1$ and $\sigma_0 = 0.005$ wins in nearly all quantities of interest, whereas CUHK03 $\lambda = 0.01$ and $\sigma_0 = 0.005$ works best for most of the quantities

Table 2. Comparison of quantities of interest, with different settings of σ_0 (0.005,0.01) and KL coefficient λ (1, 0.1, 0.01) on three datasets – CUHK01, CUHK03 and Market-1501 – after training the siamese convolutional neural network. The means of the PAC-Bayes priors are the weights of the siamese network trained with dropout-regularised ERM on 50% of the training set. As a reference for the comparisons, the test errors w.r.t. to the 0–1 loss of the siamese network with weights set to the trained prior's mean parameters on CUHK01, CUHK03 and Market-1501 are 0.1286, 0.1638 and 0.1298 respectively.
The column 'λ' means the penalty coefficient λ in Eq. (12). σ_0 is used to initialise the variance of posterior distribution. ℓ^{CE} and ℓ^{01} represent cross-entropy loss and 0–1 loss respectively. $\mathfrak{R}^{CE}$ and $\mathfrak{R}^{01}$ are computed as in Eq. (10). $\mathfrak{R}_S^{CE}$ and $\mathfrak{R}_S^{01}$ are computed as in Eq. (9). 'Rand. pred.' represents the test errors of a randomized predictor, drawn from the posterior distribution. 'Mean pred.' is the test error of the predictor having the mean of the posterior distribution as its weights. 'Agg. pred.' represents the test error of an aggregated predictor, which averages the outputs of 100 predictors drawn i.i.d. from the posterior distribution. The minimum results in each column are highlighted in bold font.

(a) CUHK01

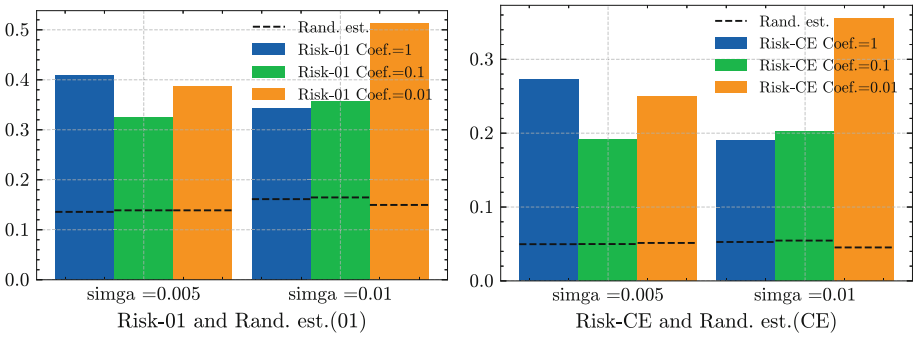
Setting		Risk & Empirical Risk		Rand. pred.	Mean pred.	Agg. pred.
λ	σ_0	$\mathfrak{R}^{CE}$ $\mathfrak{R}^{01}$	KL/n $\mathfrak{R}_S^{CE}$ $\mathfrak{R}_S^{01}$	ℓ^{CE} ℓ^{01}	ℓ^{CE} ℓ^{01}	ℓ^{CE} ℓ^{01}
1	0.005	.2725 .4083	.0279 .1214 **.2337**	.0497 **.1358**	.0466 .1309	.0480 .1314
0.1		.1916 **.3250**	.0027 .1226 .2405	.0499 .1389	.0486 .1348	.0503 .1358
0.01		.2491 .3865	.0182 .1244 .2404	.0514 .1389	.0500 .1353	.0512 .1343
1	0.01	**.1898** .3421	**.0006** .1295 .2673	.0527 .1613	.0475 .1348	.0518 .1379
0.1		.2021 .3571	.0031 .1297 .2679	.0546 .1647	.0469 .1312	.0527 .1376
0.01		.3553 .5127	.0688 **.1159** .2453	**.0453** .1497	**.0412 .1219**	**.0453 .1291**

(b) CUHK03

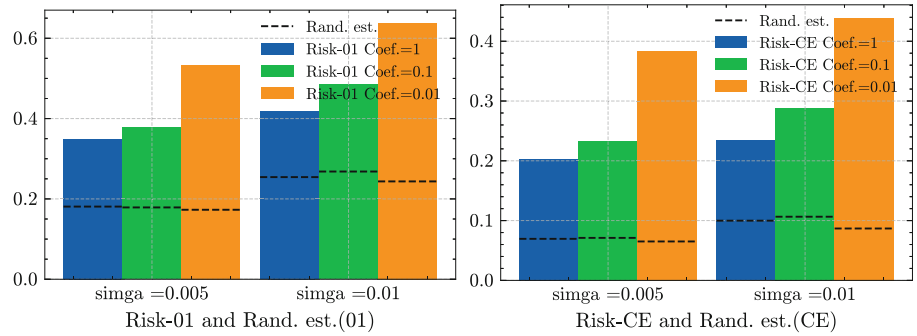
Setting		Risk & Empirical Risk		Rand. pred.	Mean pred.	Agg. pred.
λ	σ_0	$\mathfrak{R}^{CE}$ $\mathfrak{R}^{01}$	KL/n $\mathfrak{R}_S^{CE}$ $\mathfrak{R}_S^{01}$	ℓ^{CE} ℓ^{01}	ℓ^{CE} ℓ^{01}	ℓ^{CE} ℓ^{01}
1	0.005	**.2030 .3487**	.0026 .1522 .2873	.0695 .1809	.0654 .1684	.0693 .1759
0.1		.2326 .3778	.0063 .1598 .2923	.0711 .1788	.0672 .1674	.0719 .1761
0.01		.3835 .5333	.0654 **.1483 .2776**	**.0651 .1730**	**.0614 .1603**	**.0659 .1672**
1	0.01	.2340 .4191	**.0007** .1931 .3707	.0999 .2543	.0756 .1794	.0986 .2579
0.1		.2883 .4852	.0066 .2083 .3945	.1065 .2682	.0779 .1794	.1058 .2744
0.01		.4390 .6381	.0725 .1818 .3692	.0868 .2436	.0637 .1624	.0867 .2467

(c) Market-1501

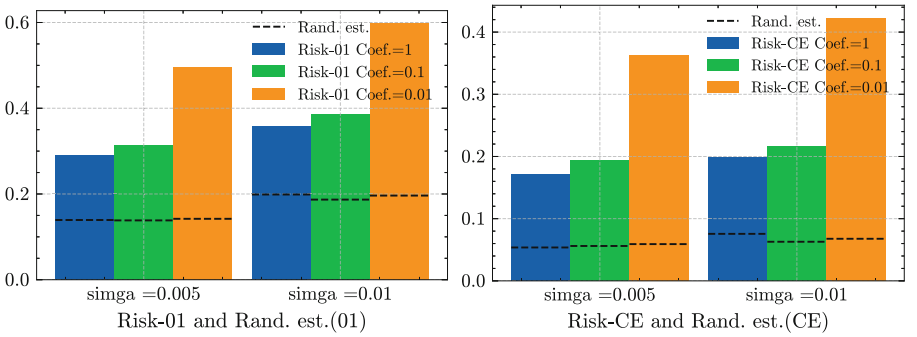
Setting		Risk & Empirical Risk		Rand. pred.	Mean pred.	Agg. pred.
λ	σ_0	$\mathfrak{R}^{CE}$ $\mathfrak{R}^{01}$	KL/n $\mathfrak{R}_S^{CE}$ $\mathfrak{R}_S^{01}$	ℓ^{CE} ℓ^{01}	ℓ^{CE} ℓ^{01}	ℓ^{CE} ℓ^{01}
1	0.005	**.1717 .2906**	.0016 **.1343 .2449**	**.0537** .1393	**.0509 .1267**	**.0533 .1300**
0.1		.1941 .3129	.0042 .1392 .2474	.0561 **.1384**	.0531 .1304	.0560 .1321
0.01		.3630 .4954	.0608 .1410 .2525	.0591 .1421	.0554 .1338	.0581 .1359
1	0.01	.1981 .3583	**.0007** .1660 .3192	.0756 .1988	.0568 .1393	.0770 .1721
0.1		.2170 .3871	.0066 .1482 .3036	.0629 .1869	.0525 .1295	.0631 .1559
0.01		.4220 .5978	.0822 .1544 .3120	.0677 .1963	.0586 .1388	.0681 .1755



(a) CUHK01



(b) CUHK03



(c) Market-1501

Fig. 2. Comparison of different KL penalty coefficients and initializations of σ_0. The figures plot results of $\Re^{CE}$, $\Re^{01}$ and the test errors of randomized predictor ('Rand. pred.') on three datasets. 'λ' means penalty coefficient in Eq. (12).

of interest. On the CUHK01 data set the results are mixed regarding the best value of λ and σ_0. However, the results are qualitatively similar with suboptimal λ and σ_0 values too, as there appears to be relatively little variation in the range of estimates in each quantity of interest (column).

5 Conclusions

We developed tuple-wise PAC-Bayes bounds, which can be applied in deep learning. Our general bound can be instantiated to any tuple size, and used as an objective function to optimize – yielding new learning algorithms that enjoy generalization guarantees. We exemplified experiments on an application of metric learning and evaluate the obtained objective functions empirically in the pairwise case. Specifically, we demonstrated learning the posterior distributions of siamese convolutional neural networks in Re-ID tasks. This complements our theoretical results by giving non-vacuous risk upper bounds based on the obtained objectives. For further flexibility, we explored balancing between the empirical risk and the KL term in the objectives. In future work, it would be interesting to test different prior and posterior distributions other than Gaussian distributions - e.g. Laplace distributions also have KL divergence in a closed form, and promote sparse connections in the neural net. We also plan to test other tuple-wise settings beyond the pairwise case, e.g. triplet case. Moreover, others frameworks with advanced training strategies, beyond the siamese convolutional network, would be worth investigating.

Acknowledgements. The work of Sijia Zhou is funded by CSC and UoB scholarship. The work of Yunwen Lei is partially supported by the Research Grants Council of Hong Kong [Project No. 22303723]. AK acknowledges recent past funding by EPSRC Fellowship grant EP/P004245/1. The experiments were conducted using the University of Birmingham's Baskerville and BlueBEAR HPC services.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

A Proofs

First, we will need to introduce two useful lemmas in the tuple-wise setting.

Lemma 3 (Lemma 3.1 in [31]). *Let T_i, $i = 1, \ldots, N$, be random variables, which are not necessarily independent and*

$$T = \sum_{i=1}^{N} p_i T_i, \quad \text{where } \sum_{i=1}^{N} p_i = 1 \quad \text{and} \quad p_i \geq 0, \quad \forall i \leq N.$$

Then for any $t > 0$, we have $\Pr\{T \geq t\} \leq \sum_{i=1}^{N} p_i \mathbb{E}\left[e^{s(T_i - t)}\right]$.

Te next lemma is a tuple-wise extension of Lemma 3.6 in [20].

Lemma 4. *For any $h_{\mathbf{w}} \in \mathcal{H}$, we denote by p the probability that $h_{\mathbf{w}}$ makes a wrong prediction. For a loss ℓ bounded by 1, any dataset S and any $\delta \in (0,1)$, given $k > 0$ we have*

$$\Pr_{S\sim D^n}\{C(n,m)R_S(h_{\mathbf{w}}) \le k\} \le e^{-\lfloor \frac{n}{m} \rfloor \mathrm{KL}_+(\frac{k}{C(n,m)}||p)}. \tag{13}$$

Proof. We bound the probability of making at most k mistakes, equivalently the probability of predicting at least $C(n,m) - k$ right. We have

$$\begin{aligned}\Pr_{S\sim\mathcal{D}^n}\{C(n,m)R_S \le k\} &= \Pr_{S\sim\mathcal{D}^n}\Big\{\sum_c \ell(h_{\mathbf{w}}, z_{i_1}, \ldots, z_{i_m}) \ge C(n,m) - k\Big\} \\ &= \Pr_{S\sim\mathcal{D}^n}\Big\{\frac{1}{C(n,m)}\ell(h_{\mathbf{w}}, z_{i_1}, \ldots, z_{i_m}) \ge 1 - \frac{k}{C(n,m)}\Big\},\end{aligned}$$

where $\sum_c$ denotes the summation over all $C(n,m)$ combinations $\{i_1, \ldots, i_m\}$ of $\{1, \ldots, n\}$. For simplicity, we denote by $q_{i_1,\ldots,i_m} = \ell(h_{\mathbf{w}}, z_{i_1}, \ldots, z_{i_m})$ and let $K = 1 - \frac{k}{C(n,m)}$.

Let $\sum_{\lfloor n/m \rfloor} q_{i_1,\ldots,i_m} = (q_{i_1,\ldots,i_m} + q_{i_{m+1},\ldots,i_{2m}} + \cdots + q_{i_{\lfloor n/m \rfloor m - m + 1},\ldots,i_{\lfloor n/m \rfloor m}})$. For any $\xi > 0$, and recall that π runs through all permutations of $\{1, \ldots, n\}$. According to Eq. (4), for π as all possible permutations of $\{1, \ldots, n\}$, we have

$$\begin{aligned}&\Pr_{S\sim\mathcal{D}^n}\Big\{\frac{1}{n!}\sum_\pi \frac{1}{\lfloor n/m \rfloor}\sum_{\lfloor n/m \rfloor} q_{i_1,\ldots,i_m} \ge K\Big\} \\ &\le \Pr_{S\sim\mathcal{D}^n}\Big\{\frac{\xi}{n!}\sum_\pi \frac{1}{\lfloor n/m \rfloor}\sum_{\lfloor n/m \rfloor} q_{i_1,\ldots,i_m} \ge \xi K\Big\} \le \sum_\pi \frac{1}{n!}\mathbb{E}_{S\sim\mathcal{D}^n}[e^{\xi(T_i - K)}],\end{aligned}$$

where we used Lemma 3 with $p_i = \frac{1}{n!}$, $N = n!$ and $T_i = \frac{1}{\lfloor n/m \rfloor}\sum_{\lfloor n/m \rfloor} q_{i_1,\ldots,i_m}$.

This decomposition gives a summation of $\lfloor n/m \rfloor$ i.i.d. random variables $q_{i_1,\ldots,i_m}$. Therefore, we can upper bound the expectation as

$$\begin{aligned}e^{-\xi K}\mathbb{E}_{S\sim\mathcal{D}^n}[e^{\xi T_i}] &= e^{-\xi K}\mathbb{E}_{S\sim\mathcal{D}^n}\Big[\exp\Big(\frac{\xi}{\lfloor n/m \rfloor}\sum_{\lfloor n/m \rfloor} q_{i_1,\ldots,i_m}\Big)\Big] \quad \text{(i.i.d.)} \\ &= e^{-\xi K}((1-p)e^{\frac{\xi}{\lfloor n/m \rfloor}} + p)^{\lfloor n/m \rfloor} = e^{\lfloor n/m \rfloor f(\xi)},\end{aligned}$$

where $f(\xi) = -\xi\frac{K}{\lfloor n/m \rfloor} + \ln((1-p)e^{\frac{\xi}{\lfloor n/m \rfloor}} + p)$. We get

$$\begin{aligned}\lfloor n/m \rfloor f(\xi) &= -\xi K + \lfloor n/m \rfloor \ln((1-p)e^{\frac{\xi}{\lfloor n/m \rfloor}} + p) \\ &= \lfloor n/m \rfloor\Big(\frac{-\xi}{\lfloor n/m \rfloor}K + \ln((1-p)e^{\frac{\xi}{\lfloor n/m \rfloor}} + p)\Big).\end{aligned}$$

Next, we minimize the bound, since the above holds with any $\xi > 0$. Let $\kappa = \frac{\xi}{\lfloor n/m \rfloor}$. Taking derivative, we have

$$f'(\kappa) = -K + \frac{(1-p)e^\kappa}{(1-p)e^\kappa + p}.$$

Solving $f'(\kappa^*) = 0$, we get that the optimal $\kappa = \kappa^*$ satisfies $K(1-p)e^{\kappa^*} + Kp = (1-p)e^{\kappa^*}$. Equivalently,

$$(1-p)(1-K)e^{\kappa^*} = Kp.$$

Hence,

$$e^{\kappa^*} = \frac{Kp}{(1-p)(1-K)}.$$

Plugging this back, we get

$$f(\kappa^*) = -K\ln(\frac{Kp}{(1-p)(1-K)}) + \ln(\frac{Kp}{1-K} + p) = -\mathrm{KL}_+(1-K||p).$$

Since $K = 1 - \frac{k}{C(n,m)}$, we got $f(\kappa^*) = -\mathrm{KL}_+(\frac{k}{C(n,m)}||p)$, which means predicting at most k times wrong. Therefore,

$$\Pr_{S\sim\mathcal{D}^n}\{C(n,m)R_S(h_{\mathbf{w}}) \leq k\} \leq e^{-\lfloor\frac{n}{m}\rfloor\mathrm{KL}_+(\frac{k}{C(n,m)}||p)}. \tag{14}$$

The proof is completed.

Proof (Proof of Lemmas 1). Recall that, for a hypothesis $h_{\mathbf{w}}$, we denoted $r_S = C(n,m)R_S(h_{\mathbf{w}})$. Given two independent training sets $S, S' \sim \mathcal{D}^n$, we consider the probability of the number of training errors to coincide for two hypotheses separately trained on them.

$$\mathbb{E}_{S\sim\mathcal{D}^n}\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{r_S = r_{S'}\}} \leq \sum_{k=1}^{C(n,m)+1}\frac{\Pr\limits_{S\sim\mathcal{D}^n}\{r_S = k\}}{\Pr\limits_{S'\sim\mathcal{D}^n}\{r_{S'} = k\}} = C(n,m)+1.$$

Take expectation w.r.t. $\mathbb{P}$ over all possible hypotheses and exchange $\mathbb{E}_{\mathbb{P}}$ and $\mathbb{E}_S$ by Fubini's theorem. This gives

$$\mathbb{E}_{S\sim\mathcal{D}^n}\mathbb{E}_{\mathbf{w}\sim\mathbb{P}}\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{r_S = r_{S'}\}} \leq C(n,m)+1.$$

By Markov inequality (for a random variable $X \geq 0$, we have $\Pr\{X \geq \delta\} \leq \frac{\mathbb{E}X}{\delta}$), for all $\mathbb{P}$, we have

$$\Pr_{S\sim\mathcal{D}^n}\left\{\mathbb{E}_{\mathbf{w}\sim\mathbb{P}}\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{r_S = r_{S'}\}} \geq \frac{C(n,m)+1}{\delta}\right\} \leq \delta.$$

The proof is completed.

Proof (Proof of Lemma 2). Based on the proof of Lemma 4, we have

$$\begin{aligned}
&\frac{\mathbb{E}_{\mathbf{w}\sim\mathbb{Q}}\ln\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{r_S = r_{S'}\}}}{\lfloor n/m\rfloor} \geq \frac{1}{\lfloor n/m\rfloor}\mathbb{E}_{\mathbf{w}\sim\mathbb{Q}}\ln\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{r_{S'} \leq r_S\}} \\
&\geq \mathbb{E}_{\mathbf{w}\sim\mathbb{Q}}\mathrm{KL}_+\Big(r_S/C(n,m)||R(h_{\mathbf{w}})\Big) \geq \mathrm{KL}_+\Big(\mathbb{E}_{\mathbf{w}\sim\mathbb{Q}}R_S(h_{\mathbf{w}})||\mathbb{E}_{\mathbf{w}\sim\mathbb{Q}}R(h_{\mathbf{w}})\Big),
\end{aligned}$$

where the first inequality follows from Eq. (14) and the last inequality is due to Jensen's inequality. The proof is completed.

Based on Lemma 1 and Lemma 2, we prove our main theorem.

Proof (Theorem 1). Given any $\mathbf{w}, \hat{\mathbf{w}} \sim \mathbb{P}$, $\hat{r}_S := C(n,m)R_S(\hat{h}_{\mathbf{w}})$, we define

$$\mathbb{P}_F(h_{\mathbf{w}}) = \frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{r_S = r_{S'}\}\mathbb{E}_{\hat{\mathbf{w}}\sim\mathbb{P}}\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{\hat{r}_S=\hat{r}_{S'}\}}}\mathbb{P}(h_{\mathbf{w}}).$$

We have

$$0 \le \mathrm{KL}(Q||\mathbb{P}_F) = \mathbb{E}_{\mathbf{w}\sim\mathbb{Q}}\ln\left[\frac{\mathbb{Q}(h_{\mathbf{w}})}{\mathbb{P}(h_{\mathbf{w}})}\Pr_{S'\sim\mathcal{D}^n}\{r_{S'} = r_S\}\mathbb{E}_{\hat{\mathbf{w}}\sim\mathbb{P}}\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{\hat{r}_S = \hat{r}_{S'}\}}\right]$$

$$= \mathrm{KL}(\mathbb{Q}||\mathbb{P}) - \mathbb{E}_{\mathbf{w}\sim\mathbb{Q}}\ln\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{r_S = r_{S'}\}} + \ln\left(\mathbb{E}_{\hat{\mathbf{w}}\sim\mathbb{P}}\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{\hat{r}_S = \hat{r}_{S'}\}}\right).$$

This gives

$$\mathbb{E}_{\mathbf{w}\sim\mathbb{Q}}\ln\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{r_S = r_{S'}\}} \le \mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\left(\mathbb{E}_{\hat{\mathbf{w}}\sim\mathbb{P}}\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{\hat{r}_S = \hat{r}_{S'}\}}\right).$$

Applying Lemma 2 on the left side, for all S, we have

$$\lfloor n/m \rfloor \mathrm{KL}_+\big(\Re(\mathbb{Q})||\Re_S(\mathbb{Q})\big) \le \mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\mathbb{E}_{\hat{\mathbf{w}}\sim\mathbb{P}}\frac{1}{\Pr\limits_{S'\sim\mathcal{D}^n}\{\hat{r} = \hat{r}_{S'}\}}.$$

By applying Lemma 1 to the above inequality, we prove the result.

References

1. Agarwal, S., Niyogi, P.: Generalization bounds for ranking algorithms via algorithmic stability. J. Mach. Learn. Res. **10**(Feb), 441–474 (2009)
2. Ambroladze, A., Parrado-Hernández, E., Shawe-Taylor, J.: Tighter pac-bayes bounds. Adv. Neural Inf. Process. Syst. **19** (2006)
3. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D., Ridgeway, G.: Learning a mahalanobis metric from equivalence constraints. J. Mach. Learn. Res. **6**(6), 1–29 (2005)
4. Bégin, L., Germain, P., Laviolette, F., Roy, J.F.: Pac-bayesian bounds based on the rényi divergence. In: Artificial Intelligence and Statistics, pp. 435–444. PMLR (2016)
5. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural network. In: International Conference on Machine Learning, pp. 1613–1622. PMLR (2015)
6. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a "siamese" time delay neural network. Adv. Neural Inf. Process. Syst. **6** (1993)
7. Cao, Q., Guo, Z.C., Ying, Y.: Generalization bounds for metric and similarity learning. Mach. Learn. **102**(1), 115–132 (2016)
8. Catoni, O.: A pac-bayesian approach to adaptive classification. preprint **840** (2003)

9. Clémençon, S., Lugosi, G., Vayatis, N.: Ranking and empirical minimization of U-statistics. Ann. Stat. 844–874 (2008)
10. Cortes, C., Mohri, M.: Auc optimization vs. error rate minimization. Adv. Neural Inf. Process. Syst. 313–320 (2004)
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009)
12. Dziugaite, G.K., Hsu, K., Gharbieh, W., Arpino, G., Roy, D.: On the role of data in pac-bayes bounds. In: International Conference on Artificial Intelligence and Statistics, pp. 604–612. PMLR (2021)
13. Dziugaite, G.K., Roy, D.M.: Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In: 33-rd Conference on Uncertainty in Artificial Intelligence. AUAI Press (2017)
14. Dziugaite, G.K., Roy, D.M.: Data-dependent pac-bayes priors via differential privacy. Adv. Neural Inf. Process. Syst. **31** (2018)
15. Guillaumin, M., Verbeek, J., Schmid, C.: Is that you? metric learning approaches for face identification. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 498–505. IEEE (2009)
16. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), vol. 2, pp. 1735–1742. IEEE (2006)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
18. Koestinger, M., Hirzer, M., Wohlhart, P., Roth, P.M., Bischof, H.: Large scale metric learning from equivalence constraints. In: Computer Vision and Pattern Recognition, pp. 2288–2295. IEEE (2012)
19. Langford, J., Caruana, R.: (not) bounding the true error. Adv. Neural Inf. Process. Syst. **14** (2001)
20. Langford, J., Schapire, R.: Tutorial on practical prediction theory for classification. J. Mach. Learn. Res. **6**(3) (2005)
21. Lei, Y., Liu, M., Ying, Y.: Generalization guarantee of SGD for pairwise learning. Adv. Neural. Inf. Process. Syst. **34**, 21216–21228 (2021)
22. Lei, Y., Yang, T., Ying, Y., Zhou, D.X.: Generalization analysis for contrastive representation learning. In: International Conference on Machine Learning, pp. 19200–19227. PMLR (2023)
23. Li, W., Wang, X.: Locally aligned feature transforms across views. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3594–3601 (2013)
24. Li, W., Zhao, R., Xiao, T., Wang, X.: Deepreid: deep filter pairing neural network for person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 152–159 (2014)
25. Lv, J., Chen, W., Li, Q., Yang, C.: Unsupervised cross-dataset person re-identification by transfer learning of spatial-temporal patterns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7948–7956 (2018)
26. McAllester, D.A.: Some pac-bayesian theorems. Mach. Learn. **37**(3), 355–363 (1999)
27. Parrado-Hernández, E., Ambroladze, A., Shawe-Taylor, J., Sun, S.: Pac-bayes bounds with data dependent priors. J. Mach. Learn. Res. **13**(1), 3507–3531 (2012)

28. Pérez-Ortiz, M., et al.: Learning pac-bayes priors for probabilistic neural networks. arXiv preprint arXiv:2109.10304 (2021)
29. Pérez-Ortiz, M., Rivasplata, O., Parrado-Hernandez, E., Guedj, B., Shawe-Taylor, J.: Progress in self-certified neural networks. In: NeurIPS 2021 workshop Bayesian Deep Learning [BDL] (2021)
30. Pérez-Ortiz, M., Rivasplata, O., Shawe-Taylor, J., Szepesvári, C.: Tighter risk certificates for neural networks. J. Mach. Learn. Res. **22** (2021)
31. Pitcan, Y.: A note on concentration inequalities for u-statistics. arXiv preprint arXiv:1712.06160 (2017)
32. Ralaivola, L., Szafranski, M., Stempfel, G.: Chromatic pac-bayes bounds for non-iid data: applications to ranking and stationary β-mixing processes. J. Mach. Learn. Res. **11**, 1927–1956 (2010)
33. Rivasplata, O., Parrado-Hernández, E., Shawe-Taylor, J.S., Sun, S., Szepesvári, C.: Pac-bayes bounds for stable algorithms with instance-dependent priors. Adv. Neural Inf. Process. Syst. 9214–9224 (2018)
34. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)
35. Serfling, R.J.: Approximation Theorems of Mathematical Statistics. John Wiley & Sons, Hoboken (2009)
36. Shawe-Taylor, J., Williamson, R.C.: A pac analysis of a bayesian estimator. In: Proceedings of the Tenth Annual Conference on Computational Learning Theory, pp. 2–9 (1997)
37. Xiong, F., Gou, M., Camps, O., Sznaier, M.: Person re-identification using kernel-based metric learning methods. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 1–16. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10584-0_1
38. Yi, D., Lei, Z., Liao, S., Li, S.Z.: Deep metric learning for person re-identification. In: 2014 22nd International Conference on Pattern Recognition, pp. 34–39. IEEE (2014)
39. Zhang, Z., Lan, C., Zeng, W., Chen, Z., Chang, S.F.: Beyond triplet loss: meta prototypical n-tuple loss for person re-identification. IEEE Trans. Multimedia **24**, 4158–4169 (2021)
40. Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., Tian, Q.: Scalable person re-identification: a benchmark. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1116–1124 (2015)
41. Zheng, Z., Zheng, L., Yang, Y.: A discriminatively learned CNN embedding for person reidentification. ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) **14**(1), 1–20 (2017)
42. Zhong, Z., Zheng, L., Cao, D., Li, S.: Re-ranking person re-identification with k-reciprocal encoding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1318–1327 (2017)