A Bayesian Neural Model for Documents' Relevance Estimation

Alberto Purpura[1], Gian Antonio Susto[1]

[1]Department of Information Engineering, University of Padova, Padova, Italy

Abstract

We propose QLFusion, an approach based on Quantification Learning (QL) to improve rank fusion performance in Information Retrieval. We first introduce a QL model based on a Bayesian Neural Network to estimate the proportion of relevant documents in a ranked list. The proposed model is trained using a probabilistic loss function formulated specifically for this QL task. Next, we describe a rank fusion algorithm which leverages on this information to merge multiple ranked lists. We compare our approach to various popular rank fusion baselines on multiple collections, showing how the proposed approach outperforms the baselines in several evaluation measures.

Keywords

bayesian neural models, quantification learning, information retrieval

1. Introduction

Quantification Learning (QL) is a Machine Learning task that can be defined as follows. Given a labeled training set, induce a quantifier that takes an unlabelled test set as input and returns its best estimate of the class distribution [1]. QL has been studied for prevalence estimation in the medical domain (i.e., to estimate the number of cases of a condition at a particular point in time) in [2], in-text classification in [1], and for sentiment analysis in [3] (more in Section 2).

However, the application of QL to traditional Information Retrieval (IR) tasks is still very limited [4]. This is likely due to the formulation of the ranking problem in IR, and to the lack of large training annotated datasets – which are starting to become available only recently with the increasing interest towards Neural IR. In this paper, we propose a QL Bayesian Neural Model (BNM) called QLFusion for the rank fusion task in IR. This task consists in combining items from multiple ranked lists into one, maximizing the relevance of the documents at the top. Over the years, other rank fusion strategies explored different probabilistic solutions to the problem; however, these solutions generally employ limited modeling strategies [5] or methods not flexible enough to adapt to the large variance introduced by the different query topics [6]. To overcome these limitations we propose to employ a shallow BNM trained with a probabilistic loss function which maximize the representation and generalization power of the model, limiting the number of parameters to be learnt and the need for large training datasets compared to an equivalent traditional neural model. BNMs are similar to traditional neural networks where the model weights are probability distributions instead of scalar values [7]. For this reason, a BNM can be compared to an ensemble of conventional neural networks in which weights are sampled from the weights distributions that our model learns [8]. We interpret the output of QLFusion as a probability value p indicating the likelihood of sampling a relevant document from a ranking list after n attempts. We then identify a Binomial distribution with parameters p and n, and train a QL model to minimize the Kullback–Leibler (KL)-divergence between the distribution identified by the QLFusion model output and the correct probability value computed using the relevance judgments – i.e., the ground truth. To minimize the need for training data, our model employs a Bayesian Neural Network (BNN) only as its output layer, i.e., a neural network with a prior distribution on its weights [7]. The use of these models in real-world scenarios has been recently made possible thanks to the theoretical advancements in the approximation of the sampling process from the learned weights distributions, which is necessary to train them [9, 10]. We then describe (Section 3) a rank fusion algorithm that leverages the information provided by the proposed model. In our evaluation (Section 4), we show how a BNM is superior to a deterministic one and to other more straightforward linear and non-linear machine learning models for the QL task at hand. Finally, we evaluate the quality of the proposed QL approach, training objective, and rank fusion algorithm in the rank fusion task, comparing it to other popular rank fusion algorithms such as CombSum, CombMNZ [11], META [5] and variants of our model on three different datasets – TREC-3 [12] and TREC-5 [13] – widely-used for testing rank-fusion methods, and CLEF-2018 [14], a recent dataset whose goal is

DESIRES 2021 – 2nd International Conference on Design of Experimental Search Information REtrieval Systems, September 15–18, 2021, Padua, Italy

purpuraa@dei.unipd.it (A. Purpura); sustogia@dei.unipd.it (G. A. Susto)

to maximize recall in the context of medical reviews.

The paper is organized as follows: Section 2 presents related works, Section 3 describes the proposed QL model and our rank fusion algorithm, Section 4 reports the experimental results and Section 5 concludes the paper.

2. Related Work

QL addresses the problem of estimating the number of positives in a set of data points, given a training set to learn to distinguish positives from negatives [4, 1]. However, this estimation is not computed by a classic classification-and-count approach, but relies on a model trained to predict the count skipping the classification step. This solution is particularly useful when: (i) the distribution of positives in the training set is different from the one in the test set, (ii) when acquiring training data is expensive, or (iii) when there are too few positive examples to learn an accurate classification model [1]. This is often the case in ad-hoc retrieval, where the problem of sparsity in the training data often limits the performance of machine learning approaches [15, 16]. For this reason, we investigate on how we can benefit from learning a model able to only *quantify* the number of relevant documents to a particular query in a collection without learning an accurate relevance model. The IR task that we choose is rank fusion, where the goal is to merge multiple ranked lists into one, maximizing the relevance of the documents at the top. We compare our approach to two classic and still competitive baselines: CombSum and CombMNZ [11]. Under CombSum, a document's score is calculated by adding the normalized scores returned by the individual input models. Regarding CombMNZ, a document's score is found by multiplying its CombSum score by the number of non-zero relevance scores that it received in the ranked lists to fuse. Our QLFusion approach is inspired from previous works in the rank fusion domain such as the linear combination model proposed by [17], where document scores are rescaled before being combined using different coefficients for each query. Our probabilistic approach to this problem is also shared by previous works in the rank fusion domain. For instance, [5] shows how documents' relevance scores associated to a query can be modeled with known distributions. Their main contribution is to show how Gaussian or Exponential distributions can fit the distribution of relevance scores of relevant and non-relevant documents, respectively. They then use this information to compute the relevance probability of each item in a ranked list and use it to combine the documents through a rank fusion algorithm (META). With a similar goal in mind but employing the quantification learning paradigm, the proposed QLFusion model directly computes these coefficients and then uses them to fuse multiple ranking lists into one.

3. Proposed Approach

In this section, we describe the BNM, the training objective and rank fusion algorithm that we propose. Afterwards, we will show how the proposed QL model can be employed to perform rank fusion and to this end we will present our QLFusion algorithm.

Quantification Learning Model. Given a user query and a document ranking returned by a retrieval model, the proposed QLFusion algorithm predicts the proportion of relevant documents in it considering their relevance scores. Indeed, the sequence of scores assigned to different documents by a retrieval model can be a proxy for the number/proportion of relevant documents retrieved. In Figure 1, we report the sequence of normalized relevance scores associated to the top 25 documents in three random topics by one of the runs submitted to TREC-3 achieving the best MAP. As observed by [6] and [5], the distribution of relevant documents in a ranked list can be estimated from the entire set of relevance scores. However, the number of retrieved relevant documents varies widely from topic to topic. For this reason, to estimate the proportion of relevant documents in different ranked lists we need a model able to adapt to each retrieval model that computed them and to the different characteristics of each topic. Following these observations, we developed a QL model to estimate the quality of a ranked list in terms of the proportion of relevant documents contained in its top-k positions. Our QL approach consists of a shallow neural model with one feed-forward hidden layer and a Bayesian neural output layer. Before feeding the data into the hidden layer of our model, we first rescale them within each batch to normalize the feature values independently [18]. Following [18], the transformation that we apply to each feature i is $\hat{x_i} = \frac{g_i}{\sigma_i}(x_i - \mu_i) + b_i$, where x_i are the features of the inputs to the normalization layer for each item in the input batch, $g_i,\ \sigma_i, \mu_i$ and b_i are respectively the gain term, standard deviation, mean and bias of the layer, out of which g_i and b_i are parameters learned by the model. After feeding the inputs to the hidden layer, we apply a sigmoid activation function to the layer outputs. The output layer that we employ is a Bayesian Neural Layer (BNL) [19]. A BNL is similar to a traditional neural network, where the weights of each layer are probability distributions instead of scalar values. We realized this layer employing the flipout technique, an efficient method for decorrelating the gradients within a mini-batch by implicitly sampling pseudo-independent weight perturbations for each example [20]. The employed weights priors are Gaussian. During inference and training, we *sample* a value for each of the network weights and we use it to compute the final output of the network. More precisely, if we represent the last layer of our model as the function $f(x, W)$ – which receives an

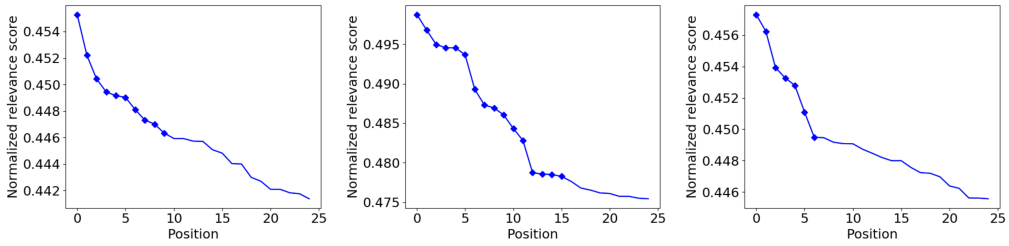

Figure 1: Normalized relevance scores for the top-ranked 25 documents of three random topics of three random runs submitted to TREC-3. Highlighted, the scores associated to relevant documents in the ranked lists.

input x and is parametrized by the weights in W – our model samples the values of W from a distribution q_θ parametrized by θ. The network is trained to minimize the expected loss $\mathbf{E}_{(x,y)\sim D,\ W\sim q_\theta}[\mathcal{L}(f(x,W),y]$, where D indicates the data distribution and $\mathcal{L}$ our loss function. The distribution of q_θ can be represented in terms of perturbations: $W = \overline{W} + \Delta W$, where $\overline{W}$ indicates the mean of the layer weights and ΔW is a stochastic perturbation. In our case, the stochastic perturbation follows a Gaussian distribution $W_{ij} \sim \mathcal{N}(\overline{W_{i,j}}, \sigma^2_{i,j})$ [20]. Thanks to these unique characteristics, a BNM can be considered equivalent to an ensemble of neural models where the weight of each model are sampled from our learned distributions. Hence, we leverage on the generalization power of ensemble approaches while at the same time improving training time and efficiency. We also apply a sigmoid activation function to the outputs of this layer to keep them in the [0-1] range. We train the proposed model in a supervised fashion by minimizing the KL-divergence between the Binomial distribution with probability equal to the model output $\hat{p}$ and the "true" proportion of relevant documents – interpreted as a Binomial distribution as well – in the input ranked list. The loss function formulation originates from the idea to interpret the output of a QL model as the proportion of successful attempts at sampling a relevant document from an input ranking list of length n. Hence, we train the model to minimize the KL-divergence between the Binomial distributions outputted by the model and the correct probability value. Formally, if we consider the true probability p to sample a relevant document from the input ranking list of length n and our model prediction $\hat{p}$, we can identify two Binomial distributions: $P \sim B(p,n)$ and $Q \sim B(\hat{p},n)$, respectively. The formulation of the loss function follows from the definition of KL-divergence between two discrete Binomial probability distributions:

$$\mathcal{L} = D_{KL}(P||Q) = \sum_{x\in\mathcal{D}} P(x)\log\left(\frac{P(x)}{Q(x)}\right) \tag{1}$$

$$= \sum_{k=0}^{n}\binom{n}{k}p^k(1-p)^{n-k}\log\left(\frac{p^k(1-p)^{n-k}}{\hat{p}^k(1-\hat{p})^{n-k}}\right), \tag{2}$$

$$= \log\left(\frac{p}{\hat{p}}\right)np + \log\left(\frac{1-p}{1-\hat{p}}\right)n(1-p). \tag{3}$$

where the last equation follows from the definition of expected value of Binomial distributions with parameters p and $(1-p)$. The loss function $\mathcal{L}$ is differentiable because it is the sum of two differentiable and continuous functions for p and $\hat{p}$ in the open $(0,1)$ interval. Note that this training strategy is different from the intuition of [5], who propose to fit a Gaussian distribution on the set of relevance scores of relevant documents. Our goal is not to learn how to fit the distribution of relevance scores but to estimate the number of relevant documents using it as an input. Our training objective allows the model to learn to better generalize the information from the training data. Indeed, the training labels are interpreted as a seed to compute a Binomial distribution which is in turn compared to the distribution obtained from the model outputs. Considering distributions instead of fixed ground truth labels gives our model some *slack* during the training process which should lead to an improvement of the performance and a decrease of the model's overfitting.

Quantification Learning-Based Rank Fusion The rank fusion approach that we propose is an extension of the popular CombSum algorithm [11] whose input is the set of document relevance scores assigned by different retrieval models. Then, given the relevance score s associated to each document and the predicted proportion of relevant documents in the ranked list p, we compute its new score w in the ranked list as: $w = s \cdot p$. We experimented with different ways of combining the information provided by the QL model in the rank fusion algorithm – such as convex combination of the QL coefficient and the normalized document score or adopting the same strategy as CombMNZ – but the differences were negligible and we preferred the simpler formulation. Finally, we combine the new document scores summing them.

	Model	P@5	Recall@5	nDCG@5	P@10	Recall@10	nDCG@10	P@20	Recall@20	nDCG@20
TREC-3	CombSum	0.8120	0.0429	0.8053	0.7680	0.0775	0.7774	0.7030	0.1331	0.7297
	CombMNZ	0.8120	0.0429	0.8053	0.7680	0.0775	0.7774	0.7030	0.1331	0.7297
	META	0.7600	0.0393	0.7652	0.7400	0.0718	0.7500	0.6810	0.1276	0.7056
	QLFusion (LR)	0.6440	0.0325	0.6540	0.6060	0.0611	0.6236	0.5800	0.1082	0.6011
	QLFusion (RF)	0.7960	0.0419	0.7917	**0.7720**	**0.0795**	0.7778	0.7040	0.1330	0.7287
	QLFusion (proposed)	**0.8160**	**0.0430**	**0.8090**	0.7680	0.0772	**0.7784**	**0.7050**	**0.1332**	**0.7316**
	QLFusion (det.)	0.8040	0.0426	0.8008	0.7660	0.0765	0.7764	0.7040	0.1331	0.7308
TREC-5	CombSum	0.6600	0.1251	0.6912	0.5720	0.1726	0.6377	0.4780	0.2240	0.5824
	CombMNZ	0.6640	0.1252	0.6938	0.5740	0.1727	0.6393	0.4780	0.2227	0.5826
	META	0.6360	0.1022	0.6546	0.5520	0.1406	0.6064	**0.4970**	0.2202	0.5837
	QLFusion (LR)	0.3440	0.0303	0.3542	0.3020	0.0486	0.3230	0.2780	0.0782	0.3085
	QLFusion (RF)	0.6520	0.1049	0.6942	**0.5840**	0.1535	0.6495	0.4830	0.2324	0.5928
	QLFusion (proposed)	**0.6720**	**0.1254**	**0.7143**	0.5780	**0.1730**	**0.6537**	0.4870	**0.2255**	**0.5969**
	QLFusion (det.)	0.6560	0.1250	0.6939	0.5740	0.1724	0.6426	0.4850	0.2254	0.5896
CLEF-2018	CombSum	0.6467	0.0673	0.6501	**0.6367**	0.1241	0.6418	0.6033	0.2305	0.6373
	CombMNZ	0.6467	0.0673	0.6501	**0.6367**	0.1241	0.6418	0.6033	0.2305	0.6373
	META	0.6467	0.0673	0.6501	**0.6367**	0.1241	0.6417	**0.6067**	**0.2310**	0.6394
	QLFusion (LR)	0.6333	0.0667	0.6335	0.6167	0.1224	0.6213	0.5933	0.2285	0.6240
	QLFusion (RF)	0.6333	0.0667	0.6456	0.6233	**0.1255**	0.6344	0.5967	0.2264	0.6328
	QLFusion (proposed)	**0.6667**	**0.0711**	**0.6628**	0.6333	0.1205	0.6403	0.6033	0.2284	0.6368
	QLFusion (det.)	0.6467	0.0673	0.6511	**0.6367**	0.1241	**0.642**6	**0.6067**	0.2307	**0.6400**

Table 1
Evaluation of QLFusion on different collections. We consider as baselines CombSum and CombMNZ and the proposed fusion algorithm relying on the predictions of a Linear Regression model (LR), a Random Forest (RF) or of a deterministic version of the proposed neural model (det.).

4. Experimental Results

We conduct our experiments on two widely-used ad-hoc TREC collections in the rank fusion domain TREC-3 [12], TREC-5 [13] and on the CLEF-2018 Technologically Assisted Reviews in Empirical Medicine (TAR) [14]. In particular, the CLEF-2018 TAR task focused on maximizing recall. We consider the 6 runs with the highest MAP submitted to each track and all of their topics. The evaluation measures that we report in our experiments are: P@{5, 10, 20}, Recall@{5, 10, 20} and nDCG@{5, 10, 20}. Also, since most of the relevant documents in a ranked list will likely be at the top of it – as we can also observe from the charts in Figure 1 – we only consider the top 100 documents in each ranked list. We employ 10-Fold cross validation to train and evaluate our model, reporting the averaged results over the 10 folds , i.e. we divide the set of topics into 10 groups, train our model on 8 of them leaving one set for validation, and evaluate our approach on the remaining one, we repeat this process 10 times. All the measures we report are averaged over the 10 folds after training a different model for each collection and retrieval model. The learning rate and hidden layer size used to train our model were 0.0005 and 32, for all collections. The batch size was optimized automatically according to the average performance of the model on the validation sets of each collection considering values in the range: [1, 2, 4, 8, 16]. We train our model for a maximum of 500 epochs with early stopping with patience 20, and evaluate it after each epoch on the validation set. For each fold, we keep the model with the minimum mean absolute error obtained on the validation set. Our implementation of the model is publicly available online. [1]

Rank Fusion. For this task, we merge the documents from different ranked lists employing QLFusion, which rescales the relevance scores of documents according to the proportion of relevant documents predicted for each ranking. The baselines that we consider are CombSum and CombMNZ – two popular and still highly competitive rank fusion approaches – META – a similar probabilistic approach for rank fusion that we reimplemented – and three variants of our QLFusion approach relying on different regression models such as a Linear Regressor (LR), a Random Forest (RF) a standard neural model with the same architecture and feature normalization strategy of the proposed Bayesian neural one but using only deterministic neural layers and the mean squared error as loss function (det.). From the results in Table 1, we observe that the proposed QLFusion algorithm combined with the probabilistic QL model outperforms all the other baselines in the majority of the evaluation measures on the TREC experimental collections, while the deterministic variant of our QLFusion approach is the best one overall on the CLEF-2018 dataset. Interestingly, the performance of all QLFusion variants on the CLEF-2018 collection is much higher than on TREC ones – especially considering the measures focusing on the top 5 and 10 items. We also observe that the META algorithm is more competitive on CLEF-2018 than on the other collections. This is likely due to the different characteristics of the runs to merge, which goal was to maximize recall instead of other ranking metrics and therefore makes them more similar to our

[1] Link to the online repository: https://github.com/albpurpura/QLFusion.

QLFusion model.

Collection	Model	Loss	P@5	Recall@5	nDCG@5
TREC-3	QLFusion prob.	proposed	**0.8160**	**0.0430**	**0.8090**
	QLFusion det.	proposed	0.8040	0.0426	0.8008
	QLFusion prob.	MSE	**0.8160**	**0.0430**	**0.8090**
	QLFusion det.	MSE	0.8040	0.0426	0.8008
TREC-5	QLFusion prob.	proposed	**0.6720**	**0.1254**	**0.7143**
	QLFusion det.	proposed	0.6520	0.1248	0.6951
	QLFusion prob.	MSE	0.6640	0.1251	0.7134
	QLFusion det.	MSE	0.6560	0.1250	0.6939
CLEF-2018	QLFusion prob.	proposed	**0.6667**	**0.0711**	**0.6628**
	QLFusion det.	proposed	0.6467	0.0673	0.6511
	QLFusion prob.	MSE	**0.6667**	**0.0711**	**0.6628**
	QLFusion det.	MSE	0.6467	0.0673	0.6511

Table 2
Evaluation of QLFusion with probabilistic and deterministic variants of the model and loss function.

Model Optimization. From the results reported in Table 2, we observe how the probabilistic neural model (QLFusion prob.) outperforms, in most cases, its deterministic variant (QLFusion det.) regardless of the chosen loss function. However, the proposed probabilistic loss gives the probabilistic model a sizeable advantage over its variant trained with the MSE loss, especially on the TREC collections. Also, the effect of the proposed loss function on the deterministic QLFusion model is not as large as on its probabilistic variant. Differently from the other experiments, on the CLEF-2018 dataset we observe that the effect of the different loss function used is negligible across the model variants. If we consider the variance of the models predictions – an indicator of how much our models overfit or underfit the data – of the probabilistic and deterministic QLFusion models with the two previous loss functions variants, we notice that the probabilistic loss favors a larger variance in the model's predictions – closer to the variance of the true proportions of relevant documents in different queries – decreasing the tendency to underfit the training data that we observed with the deterministic model and/or MSE loss functions.

5. Conclusions

In this work, we proposed a Quantification Learning-based approach for rank fusion (QLFusion). The model relies on a Bayesian Neural Network (BNN) to predict the number or relevant documents in a ranking list, an information that is then used to rescale the relevance scores of documents in different ranked lists and fuse them. We trained the proposed BNN using a custom probabilistic loss function which allowed us to improve the generalization power of the model. The proposed approach outperformed the considered baselines on three experimental collections. In the future, we plan to investigate on new features to add to our QLFusion approach to further improve its performance.

Acknowledgments

Work supported by the ExaMode project, as part of the European Union Horizon 2020 program under Grant Agreement no. 825292.

References

[1] G. Forman, Counting positives accurately despite inaccurate classification, in: Proc. of ECIR 2005, Springer, 2005, pp. 564–575.

[2] A. Moreo, F. Sebastiani, Tutorial: Supervised Learning for Prevalence Estimation, in: Proc. of the 13th International Conference on Flexible Query Answering Systems FQAS 2019, volume 11529 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 13–17.

[3] W. Gao, F. Sebastiani, Tweet Sentiment: From Classification to Quantification, in: Proc. of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM '15, ACM, New York, NY, USA, 2015, pp. 97–104.

[4] P. González, A. Castaño, N. Chawla, J. Coz, A review on quantification learning, ACM Computing Surveys (CSUR) 50 (2017) 74.

[5] R. Manmatha, T. Rath, F. Feng, Modeling score distributions for combining the outputs of search engines, in: In Proc. of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, 2001, pp. 267–275.

[6] D. Lillis, F. Toolan, R. Collier, J. Dunnion, Probfuse: a probabilistic approach to data fusion, in: Proc. of SIGIR 2006, 2006, pp. 139–146.

[7] R. Neal, Bayesian learning for neural networks, volume 118, Springer Science & Business Media, 2012.

[8] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural networks, arXiv preprint arXiv:1505.05424 (2015).

[9] A. Balan, V. Rathod, K. Murphy, M. Welling, Bayesian dark knowledge, in: Advances in Neural Information Processing Systems, 2015, pp. 3438–3446.

[10] A. Graves, Practical variational inference for neural networks, in: Advances in neural information processing systems, 2011, pp. 2348–2356.

[11] J. Shaw, E. Fox, Combination of multiple searches, in: Proc. TREC 1994, 1994, pp. 105–108.

[12] D. Harman, Overview of the third text retrieval conference (TREC-3), 500, DIANE Publishing, 1995.

[13] D. Harman, Overview of the fifth text retrieval conference (TREC-5), 1997.

[14] K. Evangelos, A. L. Li, D., R. Spijker, CLEF 2018 technologically assisted reviews in empirical

medicine overview, in: CEUR Workshop Proceedings, volume 2125, 2018.
[15] J. Guo, Y. Fan, L. Pang, L. Yang, Q. Ai, H. Zamani, C. Wu, W. Croft, X. Cheng, A deep look into neural ranking models for information retrieval, Information Processing & Management (2019) 102067.
[16] S. Marchesin, A. Purpura, G. Silvello, Focal elements of neural information retrieval models. an outlook through a reproducibility study, Information Processing & Management (2019) 102109.
[17] L. Si, J. Callan, Using sampled data and regression to merge search engine results, in: Proc. of SIGIR 2002, 2002, pp. 19–26.
[18] J. Lei Ba, J. Kiros, G. Hinton, Layer normalization, arXiv preprint arXiv:1607.06450 (2016).
[19] Y. Gal, Z. Ghahramani, Bayesian convolutional neural networks with bernoulli approximate variational inference, arXiv preprint arXiv:1506.02158 (2015).
[20] Y. Wen, P. Vicol, J. Ba, D. Tran, R. Grosse, Flipout: Efficient pseudo-independent weight perturbations on mini-batches, arXiv preprint arXiv:1803.04386 (2018).