Progressive Knowledge Distillation: Balancing Inference Latency and Accuracy at Runtime

Don Kurian Dennis [1] **Abhishek Shetty** [2] **Anish Sevekari** [1] **Kazuhito Koishida** [3] **Virginia Smith** [1]

Abstract

We study the problem of *progressive distillation*: Given a large, pretrained teacher model g, we seek to decompose the model into smaller, low-inference cost student models f_i, such that progressively evaluating additional models in this ensemble results in strict improvements over previous predictions. For user-facing inference applications, this allows us to flexibly trade accuracy for inference latency at runtime. We develop a boosting based algorithm, B-DISTIL, for progressive distillation, and demonstrate its effectiveness on standard datasets.

1. Introduction

Knowledge distillation aims to transfer the knowledge of a large model into a smaller one (Buciluǎ et al., 2006; Hinton et al., 2015). While this technique is commonly used for model compression, one downside is that the procedure is fairly rigid—resulting in a single compressed model of a fixed size. In this work, we instead consider the problem of *progressive distillation*: approximating a large model via an ensemble of smaller, low-latency models such that progressively evaluating additional models in this ensemble leads to improved predictions. The resulting decomposition is useful for many applications in on-device and low-latency inference. For example, components of the ensemble can be selectively combined to flexibly meet accuracy-latency constraints for user-facing interactive applications, can enable efficient parallel inference execution schemes, and can facilitate *early-exit* (Bolukbasi et al., 2017; Dennis et al., 2018) or *anytime inference* (Ruiz & Verbeek, 2021; Huang et al., 2017) applications, which are scenarios where inference may be interrupted due to variable resource availability. These are particularly relevant as we allow more media of interactions (ex. speech) with large language models.

[1]Carnegie Mellon University, Pittsburgh, PA, USA [2]University of California at Berkeley, CA, USA [3]Microsoft, Redmond, WA, USA. Correspondence to: Don Dennis <dondennis@cmu.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

In this work we propose an algorithm for progressive distillation, B-DISTIL and empirically evaluate its effectiveness in trading execution time (latency) and accuracy in sequential inference tasks. We supplement the experiments with theoretical guarantees in terms of in-sample convergence and generalization performance. While we only look at the benefits in terms of latency, we can also use B-DISTIL for other profiling based metrics. For instance, B-DISTIL can be applied to the cloud based batch-inference setting, to trade-off accuracy and throughput (in samples per second).

2. Background and Related Work

Knowledge distillation. On device machine learning inference is often resource-constrained in practice due to requirements around metrics such as memory, energy, cost, or latency. This has spurred the development of numerous techniques for model compression. A particularly popular approach is *knowledge distillation*, which aims to transfer the knowledge of a larger model (or model ensemble) to a smaller one (Buciluǎ et al., 2006; Hinton et al., 2015).

Despite its popularity, performing compression via distillation has several known pitfalls. Most notably, it is well-documented that distillation performs poorly when there is a *capacity gap*, i.e., the teacher is significantly larger than the student (Mirzadeh et al., 2020; Gao et al., 2021; Cho & Hariharan, 2019; Allen-Zhu & Li, 2020). When performing distillation onto a weighted combination of ensembles, it has been observed that adding additional models into the ensemble does not dramatically improve performance over that of a single distilled model (Allen-Zhu & Li, 2020). There is also a lack of theoretical work characterizing distillation and its effectiveness for compression (Gou et al., 2021).

Two-player games, online optimization and boosting. In this work we formulate progressive distillation as a two player zero-sum game. The framework of two player games have lead to a number of important results in machine learning over the years (von Neumann & Morgenstern, 1944; Freund, 1990; Schapire, 1990; Mason et al., 1999). While algorithms like AdaBoost (Freund & Schapire, 1997) have seen significant success in machine learning, it has only recently seen success in modern deep learning applications. In particular, Suggala et al. (2020) propose a generalized

boosting framework to *train* boosting based ensembles of deep networks. Their key insight is that allowing function compositions in feature space can help boost deep neural networks by bridging the capacity gap problem.

A more general application of boosting that is similar to our setup is by Trevisan et al. (2009). They prove that given a target bounded function g (e.g., the teacher model) and class of candidate approximating functions $f \in \mathcal{F}$, one can iteratively approximate g arbitrarily well with respect to $\mathcal{F}$ using ideas from online learning and boosting. However, this work depends on the ability to find a function f_t in iteration t that leads to at least a small constant improvement in a round-dependent approximation loss. A key contribution of our work is showing that such functions can be found for the practical application of progressive distillation by carefully selecting candidate models

3. Progressive Knowledge Distillation

Our goal is to approximate a large model via an ensemble of smaller, low-latency models so that we can easily trade-off accuracy and inference-time/latency at runtime. In this section we formalize the problem of progressive distillation as a two player game and discuss B-DISTIL

3.1. Problem Formulation and Algorithm

To cast the problem of progressive distillation as a two player game, we consider starting with a teacher (a model or an ensemble) $g : \mathcal{X} \to \mathbb{R}^L$ along with a non empty set of low inference cost[1] hypothesis classess $\{\mathcal{F}_m\}_{m=1}^M$. Here we assume that the set $\{\mathcal{F}_m\}$ is in increasing order of cost measured in inference time. Given a training dataset $\{x_i\}$, we aim to ensure that the ensemble is a good approximation of the teacher model by minimizing the total squared error between the teacher model and ensemble, i.e., the loss function of interest is: $\frac{1}{2}\sum_{i,j}\left(f(x_i) - g(x_i)\right)_j^2$. We achieve this by searching for 'weak learners' from $\{\mathcal{F}_m\}_{m=1}^M$.

Concretely, at each iteration t, our proposed algorithm, B-DISTIL, maintains matrices $K_t^+ \in R^{N\times L}$ and $K_t^- \in R^{N\times L}$ of probabilities (in our setting, it turns out to be easier to maintain the positive errors and the negative errors separately). The matrices K_t^+ and K_t^- are such that for all $j \in [L]$, $\sum_i K_t^+(i,j) + K_t^-(i,j) = 1$. Moreover, for all $(i,j) \in [N] \times [L]$, $0 \le K_t^+(i,j), K_t^-(i,j) \le 1$. At each round t, with the current probability matrices K_t^-, K_t^+, B-DISTIL performs two steps; first, it invokes a subroutine FIND-WL that attempts to find a classifier $f_t \in \mathcal{F}_r$ satisfying the weak learning condition (Definition 1). If such a predictor is found, we add it to our ensemble and proceed to the second step, updating the probability matrices

[1] 'Inference cost' can be in terms of memory requirements, compute requirements, or other metrics, based on the use-case.

Algorithm 1 B-DISTIL: Main algorithm

Require: Target g, rounds T, data $\{(x_i, y_i)\}_{i=1}^N$, learning rate η, model classes $\{\mathcal{F}_m\}_{m=1}^M$

$K_t^+(i,j), K_t^-(i,j) \leftarrow \frac{1}{2N}, \frac{1}{2N} \quad \forall (i,j)$
$F, r, t \leftarrow \emptyset, 1, 1$
while $r < R$ and $t < T$ **do**
  $f_t = \text{FIND-WL}(K_t^+, K_t^-, \mathcal{F}_r)$
  if f_t is NONE **then**
    $r \leftarrow r + 1$
    continue
  With $l := f_t - g$, update K_t^+, K_t^-. $\forall (i,j)$

$$K_{t+1}^+(i,j) \leftarrow K_t^+(i,j)\exp(-\eta \cdot l(x_i)_j) \quad (1)$$

$$K_{t+1}^-(i,j) \leftarrow K_t^-(i,j)\exp(\eta \cdot l(x_i)_j) \quad (2)$$

  Normalize K_t^+, K_t^-.
  $F, t \leftarrow F \cup \{f_t\}, t+1$
Return $\frac{1}{|F|}\sum_{i=1}^{|F|} f_i$

Algorithm 2 FIND-WL

Require: Probability matrices K^+, K^-, model class $\mathcal{F}$ parameterized by $\theta \in \Theta$, SGD hyperparameters

Obtain $\{\mathcal{F}_r\}_1^R$ by expanding $\mathcal{F}$ (Section 3.2).
for $\mathcal{F}' \in \{\mathcal{F}_r\}_{r=1}^R$ **do**
  Initialize initial parameter $\theta_0 \in \mathcal{F}'$.
  for $i \in \{1, \ldots, \text{max-search}\}$ **do**
    Randomly initialize f_{θ_i}.
    Run SGD to solve Equation (3).
    if f_{θ_i} is a weak learner **then**
      Return f_{θ_i}
Return NONE

K_t^-, K_t^+ based on errors made by f_t. If no such predictor can be found, we invoke the subroutine with the next class, $\mathcal{F}_{r+1}$, and repeat the search till a weak learner is found or we have no more classes to search in. This is similar in spirit to boosting algorithms such as AdaBoost (Schapire & Freund, 2013) for binary classification. We formalize our notion of weak learners in this setting using Definition 1, which can be seen as a natural extension of the standard weak learning assumption in the boosting literature. Note that the elements $K_t^+(i,j)$ and $K_t^-(i,j)$ can be thought of as the weight on the residual errors $f_{t-1}(x) - g(x)$ *and* $g(x) - f_{t-1}(x)$ respectively, up-weighting large deviations from the teacher model $g(x)$.

Definition 1 (Weak learning condition)**.** *Given a dataset $\{(x_i, y_i)\}_{i=1}^N$, a target function $g : \mathcal{X} \to \mathbb{R}^L$ and probability matrices K_t^+, K_t^-, a function $f_t : \mathcal{X} \to \mathbb{R}^L$ is said to satisfy the weak learning condition with respect to g, $\forall j$, if the following sum is strictly positive:*

$$\sum_i K_t^+(i,j)(f_t(x_i) - g(x_i))_j + K_t^-(i,j)(g(x_i) - f_t(x_i))_j.$$

3.2. Finding Weak Learners

As mentioned earlier, the main difficulty in provably approximating the teacher model in this setting is in finding a *single* learner f_t at round t that satisfies our weak learning condition simultaneously for all labels j. Thus, along with controlling temperature for distillation, we employ two additional strategies: 1) we use a log-barrier regularizer in the objective FIND-WL solves to promote weak learning and, 2) we efficiently reuse a limited number of stored activation outputs of previously evaluated models to increase the expressivity of the current base class.

Log-barrier regularizer. To find a weak learner, the FIND-WL method minimizes the sum of two loss terms using stochastic gradient descent. The first is standard binary/multi-class cross-entropy distillation loss (Hinton et al., 2015), with temperature smoothing. The second term is defined in Equation (3):

$$-\frac{1}{\gamma}\sum_{i,j} I_{ij}^{+}\log\left(1+\frac{l(x_i)_j}{2B}\right)+(1-I_{ij}^{+})\log\left(1-\frac{l(x_i)_j}{2B}\right) \tag{3}$$

Here $I_{ij}^{+} := I[K_t^{+}(i,j) > K_t^{-}(i,j)]$, B is an upper bound on the magnitude of the logits, and $l(x_i) := f(x_i) - g(x_i)$. To see the intuition behind Equation (3), assume the following holds; $\forall(i,j)$,

$$(K_t^{+}(i,j) - K_t^{-}(i,j))(f(x_i) - g(x_i))_j > 0. \tag{4}$$

Summing over all x_i, we can see that this is sufficient for f to be a weak learner with respect to g. Equation (3) is a soft log-barrier version of the weak learning condition, that penalizes those (i,j) for which Equation (4) does not hold. By tuning γ we can increase the relative importance of the regularization objective, encouraging f_t to be a weak learner potentially at the expense of classification performance.

Intermediate layer connections and profiling. As discussed in Section 2, distillation onto a linear combination of low capacity student models often offers no better performance than that of any single model in the ensemble trained independently. For boosting, empirically we see that once the first weak learner has been found in some class $\mathcal{F}_m$ of low-capacity deep networks, it is difficult to find a weak learner for the reweighed objective from the same class $\mathcal{F}_m$. To work around this we let our class of weak learners at round t depend on the restricted output of intermediate layers of previous weak learners (Suggala et al., 2020).

3.3. Theoretical Analysis

In our theoretical analysis we show that the ensemble produced by algorithm 1 converges to g at $\mathcal{O}(1/\sqrt{T})$ rate, provided that the procedure FIND-WL succeeds at every t (Theorem 1). We further show excess risk bounds in Theorem 2. The details of the proof are differed to Appendix A.

Theorem 1. *Suppose the class $\mathcal{F}$ satisfies that for all $f \in \mathcal{F}$, $\|f-g\|_\infty \le G_\infty$. Let $F = \{f_t\}$ be the ensemble after T rounds of Algorithm 1, with the final output $F_t = \frac{1}{T}\sum_{t=1}^{T} f_t$. If f_t satisfies eq. (4) for all $t \le T$ then for $T \ge \ln 2N$ and $\eta = \frac{1}{G_\infty}\sqrt{\frac{\ln 2N}{T}}$, we have for all j*

$$\|F_{t,j} - g_j\|_\infty \le G_\infty\sqrt{\frac{\ln 2N}{T}}\,, \tag{5}$$

where $F_{t,j}$ and g_j are the j^{th} coordinates of the functions F_t and g respectively.

Theorem 2 (Excess Risk)**.** *Suppose data D contains of N iid samples from distribution $\mathcal{D}$ and that the function g has ϵ margin on data D with probability μ, i.e., $\Pr_{x\sim D}[|g(x)| < \epsilon] < \mu$. Further, suppose that the class $\mathcal{C}_T$ has VC dimension d. Then, for $T \ge 4G_\infty^2 \ln 2N/\epsilon^2$, with probability $1-\delta$ over the samples, the output F_T of algorithm 1 satisfies:*

$$\mathrm{err}(F_T) \le \widehat{\mathrm{err}}(g) + O\left(\sqrt{\frac{d\ln(N/d)+\ln(1/\delta)}{N}}\right) + \mu\,.$$

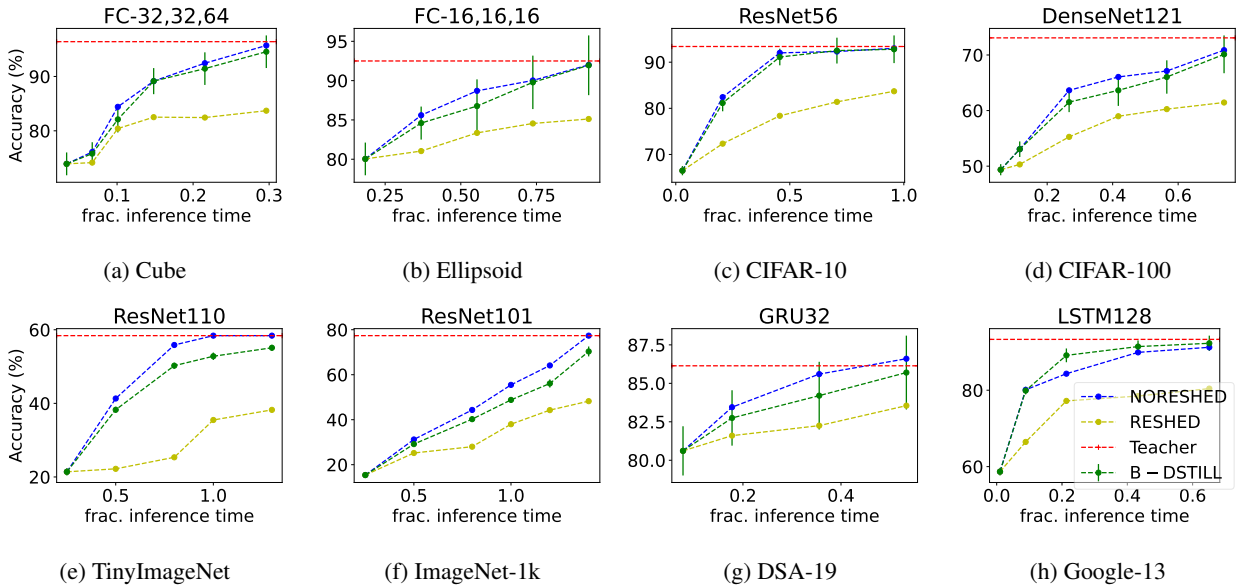
4. Empirical Evaluation

4.1. Dataset Information

We perform progressive distillation of publically available pretrained teacher models onto smaller student models using the *CIFAR-10*, *CIFAR-100*, *TinyImageNet*, *ImageNet*, *Google-13* speech commands, daily sports activities (*DSA*) dataset along with two synthetic datasets for our experiments. For all distillation tasks, for simplicity we design the student base model class from the same architecture type as the teacher model, but start with significantly fewer parameters and resource requirements. For detailed information of all datasets and model parameters used see Appendix B.

4.2. Experimental Evaluation and Results

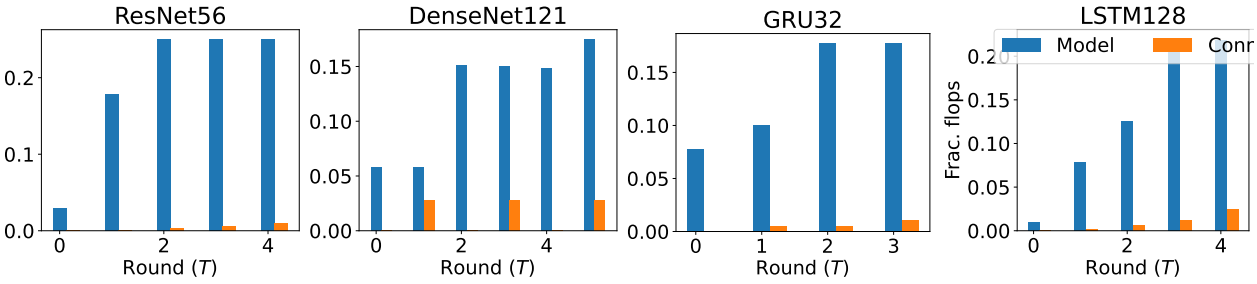
We apply B-DISTIL to the problem of *anytime-inference* where model is required to produce a prediction even when its execution is interrupted. Standard model architectures can only output a prediction once the execution is complete and thus are unsuitable for this setting. We instead compare against the idealized baseline where we assume *oracle* access to the inference budget which is usually only available *after* the execution is finished or is interrupted. Under this assumption, we can train a set of models suitable various inference time constraints, e.g., by training models at various depths, and then pick the one that fits the current inference budget obtained by querying the oracle. We refer to this baseline as NO-RESHED and compare B-DISTIL to it on both synthetic and real world datasets in Figure 1. This idealized baseline can be considered an upper bound on the accuracy of B-DISTIL for a fixed inference budget.



(a) Cube (b) Ellipsoid (c) CIFAR-10 (d) CIFAR-100

(e) TinyImageNet (f) ImageNet-1k (g) DSA-19 (h) Google-13

Figure 1. Accuracy vs. inference-time trade-offs. Inference time is reported as a fraction of teacher's inference time along with average ensemble accuracy and error bars. B-DISTIL performs this trade-off at runtime. The baseline NO-RESHED at inference time τ_w (x-axis) is the accuracy of a single model that is allowed $|\tau_w - 0|$ time for inference. Similarly the baseline RESHED at τ_w is the accuracy of an ensemble of models, where the model w is allowed $|\tau_w - \tau_{w-1}|$ time to perform its inference. This is also the latency between the successive predictions from B-DISTIL. We can see that B-DISTIL (green) remains competitive to the oracle baseline (NO-RESCHED, blue) and outperforms weighted averaging (RESCHED, yellow).



(a) CIFAR10 (b) CIFAR-100 (c) DSA-19 (d) Google-13

Figure 2. Overhead of connections. The floating point operations required to evaluate the model added in round T, compared to that required to evaluate just the connections used by this model. We present the results corresponding to datasets that have models with smaller required FLOPs overall. We see that even for these models the connections add relatively little overhead.

B-DISTIL can improve on its initial prediction whenever inference jobs are allowed to be rescheduled. To contextualize this possible improvement, we consider the case where the execution is interrupted and rescheduled (with zero-latency, for simplicity) at times $\{\tau_1, \tau_2, \ldots, \tau_W\}$. We are required to output a prediction at each τ_w. Assuming we know these interrupt points in advance, one solution can involve selecting models with inference budgets $|\tau_1|, |\tau_2 - \tau_1|, \ldots, |\tau_w - \tau_{w-1}|$ and sequentially evaluate them, and at at each interrupt τ_w, output the (possibly weighted) average prediction of the w models. We call this lower-bound baseline as RESCHED.

We see that at all interrupts points in Figure 1, the predictions provided by B-DISTIL are competitive to that of the idealized baseline RESHED which requires the inference budget ahead of time for model selection, while being able to improve on its initial predictions if rescheduled. The FLOPs required to evaluate the corresponding intermediate connections is shown in Figure 2. Here, we compare the FLOPs required to evaluate the model from round T to the FLOPs required evaluate the intermediate connections used by this model. Summing up all the FLOPs up to a round T, gives the total FLOPs required to for the ensemble with the first T models. For all our models, the overhead of connections is relatively negligible.

References

Allen-Zhu, Z. and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

Bolukbasi, T., Wang, J., Dekel, O., and Saligrama, V. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, 2017.

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *International Conference on Knowledge Discovery and Data-mining*, 2006.

Cho, J. H. and Hariharan, B. On the efficacy of knowledge distillation. In *Internationa Conference on Computer Vision*, 2019.

Dennis, D., Pabbaraju, C., Simhadri, H. V., and Jain, P. Multiple instance learning for efficient sequential data classification on resource-constrained devices. *Advances in Neural Information Processing Systems*, 2018.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Freund, Y. Boosting a weak learning algorithm by majority. In *Workshop on Computational Learning Theory*, 1990.

Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1997.

Gao, M., Wang, Y., and Wan, L. Residual error based knowledge distillation. *Neurocomputing*, 2021.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. Knowledge distillation: A survey. *International Journal of Computer Vision*, 2021.

Hinton, G., Vinyals, O., Dean, J., et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Huang, G., Chen, D., Li, T., Wu, F., Van Der Maaten, L., and Weinberger, K. Q. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Le, Y. and Yang, X. S. Tiny imagenet visual recognition challenge. 2015.

Mason, L., Baxter, J., Bartlett, P., and Frean, M. Boosting algorithms as gradient descent. *Advances in Neural Information Processing Systems*, 1999.

Mirzadeh, S. I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., and Ghasemzadeh, H. Improved knowledge distillation via teacher assistant. In *AAAI Conference on Artificial Intelligence*, 2020.

Ruiz, A. and Verbeek, J. Anytime inference with distilled hierarchical neural ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.

Schapire, R. E. The strength of weak learnability. *Machine learning*, 1990.

Schapire, R. E. and Freund, Y. Boosting: Foundations and algorithms. *Kybernetes*, 2013.

Suggala, A., Liu, B., and Ravikumar, P. Generalized boosting. *Advances in Neural Information Processing systems*, 2020.

Trevisan, L., Tulsiani, M., and Vadhan, S. Regularity, boosting, and efficiently simulating every high-entropy distribution. In *IEEE Conference on Computational Complexity*, 2009.

von Neumann, J. and Morgenstern, O. Theory of games and economic behavior, 1944.

Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition, 2018.

A. Proofs of Main Theorems

Here, we provide a proof of Theorem 1, which is restated below:

Theorem. *Suppose the class $\mathcal{F}$ satisfies that for all $f \in \mathcal{F}$, $\|f - g\|_\infty \leq G_\infty$. Let $F = \{f_t\}$ be the ensemble after T rounds of Algorithm 1, with the final output $F_t = \frac{1}{T}\sum_{t=1}^{T} f_t$. Then for $T \geq \ln 2N$ and*

$$\eta = \frac{1}{G_\infty}\sqrt{\frac{\ln 2N}{T}}$$

we have for all j

$$\|F_{t,j} - g_j\|_\infty \leq G_\infty\sqrt{\frac{\ln 2N}{T}} - \frac{1}{T}\sum_{t=1}^{T}\gamma_t(j)$$

where $F_{t,j}$ and g_j are the j^{th} coordinates of the functions F_t and g respectively.

Proof. For simplicity, we assume that f_t and g are scalar valued functions, since the proof goes through coordinate-wise. At each time, define the edge of the weak learning algorithm to be

$$\gamma_t = \sum_i K_t^+(i)(f_t(x_i) - g(x_i)) + \sum_i K_t^-(i)(g(x_i) - f_t(x_i))$$

Let Z_t denote the normalizing constant at time t, that is,

$$Z_t = \sum_i K_t^+(i)\exp\left(-\eta\left(f_t\left(x_i\right) - g(x_i)\right)\right) + K_t^-(i)\exp\left(\eta\left(f_t\left(x_i\right) - g(x_i)\right)\right)$$

From the update rule, we have

$$\begin{aligned}
K_{T+1}^+(i) &= \frac{K_T^+(i)e^{\eta(f_T(x_i) - g(x_i))}}{Z_T} \\
&= \frac{K_1^+(i)\exp\left(-\eta\sum_{t=1}^{T}\left(f_t\left(x_i\right) - g(x_i)\right)_j\right)}{\prod_{t=1}^{T} Z_t} \\
&= \frac{K_1^+(i)\exp\left(-\eta T(F_T(x_i) - g(x_i)\right)}{\prod_{t=1}^{T} Z_t}
\end{aligned}$$

and similarly

$$K_{T+1}^-(i) = \frac{K_1^-(i)\exp\left(\eta T(F_T(x_i) - g(x_i)\right)}{\prod_{t=1}^{T} Z_t}$$

First, we bound $\ln(Z_t)$:

$$\begin{aligned}
\ln(Z_t) &= \ln\left(\sum_i K_t^+(i)\exp(-\eta(f_t(x_i) - g(x_i))) + \sum_i K_t^-(i)\exp(\eta(f_t(x_i) - g(x_i)))\right) \\
&\leq \ln\left(\sum_i K_t^+(i)\left(1 - \eta(f_t(x_i) - g(x_i)) + \eta^2(f_t(x_i) - g(x_i))^2\right)\right. \\
&\qquad\qquad \left. + \sum_i K_t^-(i)\left(1 + \eta(f_t(x_i) - g(x_i)) + \eta^2(f_t(x_i) - g(x_i))^2\right)\right) \\
&\leq \ln\left(1 - \eta\sum_i K_t^+(i)(f_t(x_i) - g(x_i)) + \eta\sum_i K_t^-(i)(f_t(x_i) - g(x_i)) + \eta^2 G_\infty^2\right) \\
&\leq -\eta\gamma_t + \eta^2 G_\infty^2
\end{aligned}$$

where the second step follows from the identity $\exp(x) \leq 1 + x + x^2$ for $x \leq 1$, provided that $\eta \leq \frac{1}{G_\infty}$. This gives us a bound on regression error after T rounds:

$$\begin{aligned}
-\eta T \left(F_T(x_i) - g(x_i)\right) &= \ln(K^+_{T+1}(i)) - \ln(K^+_1(i)) + \sum_{t=1}^{T} \ln(Z_t) \\
&\leq \ln\left(\frac{K^+_{T+1}(i)}{K^+_1(i)}\right) + \sum_{t=1}^{T} -\eta\gamma_t + \eta^2 G^2_\infty \\
&= \ln\left(\frac{K^+_{T+1}(i)}{K^+_1(i)}\right) + \eta^2 T G^2_\infty - \eta \sum_{t=1}^{T} \gamma_t \\
&\leq \ln 2N + \eta^2 T G^2_\infty - \eta \sum_{t=1}^{T} \gamma_t \,,
\end{aligned}$$

where the last bound follows since $K^+_1 = \frac{1}{2N}$ and $K^+_{T+1} \leq 1$. Similarly, we have the bound

$$\eta T \left(F_T(x_i) - g(x_i)\right) \leq \ln 2N + \eta^2 T G^2_\infty - \eta \sum_{t=1}^{T} \gamma_t$$

Combining the two equations we get that

$$\sup_i |F_T(x_i) - g(x_i)| = \|F_T - g\|_\infty \leq \frac{\ln 2N}{\eta T} + \eta G^2_\infty - \frac{1}{T} \sum_{t=1}^{T} \gamma_t \,.$$

If we choose $\eta = \frac{1}{G_\infty}\sqrt{\frac{\ln 2N}{T}}$ to minimize this expression, then we get the following bound on regression error:

$$\|F_t - g\|_\infty \leq -\frac{1}{T} \sum_{t=1}^{T} \gamma_t + G_\infty \sqrt{\frac{\ln 2N}{T}} \,.$$

which is exactly Equation (5). Note that the value of η only satisfies the condition $\eta \leq \frac{1}{G_\infty}$ when $T \geq \ln 2N$, which is the time horizon after which the bound holds. This finishes the proof of Theorem 1. □

Now, we provide a proof of Theorem 2 which follows from the VC dimension bound and Theorem 1. Before we begin, we setup some notation. Given a function f, distribution $\mathcal{D}$ over space $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the label space, and data D consisting of N iid samples $(x, y) \sim \mathcal{D}$, we define

$$\widehat{\text{err}}(f) = \Pr_{(x,y)\sim D}\left[\text{sign}(F_T(x) \neq y)\right] \qquad \text{err}(f) = \Pr_{(x,y)\sim \mathcal{D}}\left[\text{sign}(F_T(x) \neq y)\right]$$

Theorem (Excess Risk)**.** *Suppose data D contains of N iid samples from distribution $\mathcal{D}$. Suppose that the function g has large margin on data D, that is*

$$\Pr_{x\sim D}\left[|g(x)| < \epsilon\right] < \mu$$

Further, suppose that the class $\mathcal{C}_T$ has VC dimension d, then for

$$T \geq \frac{4G^2_\infty \ln 2N}{\epsilon^2},$$

with probability $1 - \delta$ over the draws of data D, the generalization error of the ensemble F_T obtained after T round of Algorithm 1 is bounded by

$$\text{err}(F_T) \leq \widehat{\text{err}}(g) + O\left(\sqrt{\frac{d\ln(N/d) + \ln(1/\delta)}{N}}\right) + \mu$$

Proof. Recall the following probability bound Schapire & Freund (2013, theorem 2.5) which follows Sauer's Lemma:

$$\Pr\left[\exists f \in \mathcal{C}_T : \mathrm{err}(f) \geq \widehat{\mathrm{err}}(f) + \epsilon\right] \leq 8\left(\frac{me}{d}\right)^d e^{-m\epsilon^2/32}$$

which holds whenever $|D| = N \geq d$. It follows that with probability $1-\delta$ over the samples, we have for all $f \in \mathcal{C}_{\mathcal{T}}$

$$\mathrm{err}(f) \leq \widehat{\mathrm{err}}(f) + O\left(\sqrt{\frac{d\ln(N/d) + \ln(1/\delta)}{N}}\right) \tag{6}$$

Since we choose $T = \frac{4G_\infty^2 \ln 2N}{\epsilon^2}$, by Theorem 1, we have

$$\forall x \in D : \|F_t - g\|_1 \leq G_\infty\sqrt{\frac{\ln 2N}{T}} \leq \frac{\epsilon}{2}$$

Since g has ϵ margin on data with probability $1-\mu$, we have

$$\widehat{\mathrm{err}}(F_t) \leq \widehat{\mathrm{err}}(g) + \mu \tag{7}$$

Combining eqs. (6) and (7), we conclude the proof,

$$\mathrm{err}(F_T) \leq \widehat{\mathrm{err}}(g) + O\left(\sqrt{\frac{d\ln(N/d) + \ln(1/\delta)}{N}}\right) + \mu.$$

□

B. Dataset Information and Training Recipe

B.1. Dataset

Dataset	Train-samples	Test/Val-samples	Num.-labels	Source
CIFAR-10	50000	10000	10	(Krizhevsky et al.)
CIFAR-100	50000	10000	100	(Krizhevsky et al.)
DSA-19	6800	2280	19	(Dua & Graff, 2017)
Google-13	52886	6835	13	(Warden, 2018)
ImageNet-1k	1281167	50000	1000	(Russakovsky et al., 2015)
TinyImageNet-200	100000	10000	200	(Le & Yang, 2015)

We use two synthetic datasets in our experiments, *ellipsoid* and *cube*. To construct the ellipsoid dataset, we first sample a 32×32 matrix B, each entry sampled *iid*. We define $A := B^T B$ as our positive semi-definite matrix, and $I[x^T Ax \geq 0]$ determines the label of a data point x. We sample 10k points uniform randomly from $[-1, 1]^{32}$ and determine their labels to construct our data sample. We randomly construct a 80-20 train-test split for our experiments.

To construct *cube*, we first sample 16 vertices uniform randomly from $[-1, 1]^{32}$ and split them into 4 equal sets, say $\{S_1, \ldots, S_4\}$. As before, we sample 10k points uniformly from $[-1, 1]^{32}$ and determine the label $y(x)$ of each point x based on the closest vertex in $\{S_1, \ldots, S_4\}$.

$$y(x) = \arg\min_i \min_{x' \in S_i} \|x - x'\|.$$

B.2. Training Recepies

We use stochastic gradient descent (SGD) with momentum for all our experiments. For experiments on CIFAR100 and CIFAR10, we use a learning rate of 0.1, a momentum parameter of 0.9, and weight decay of 5×10^{-4}. We train for 200 epochs and reduce the learning rate by a factor of 0.2 in after 30%, 60% and 90% of the epoch execution. We perform a 4-GPU data-parallel training for ImageNet with a per-gpu batch size of 256, learning rate 0.1, momentum 0.9, regularization γ of 1.0, and a weight decay of $1e-4$. We train for 90 epochs with and discount the learning rate by a factor of 0.1 at 30% and 60% epochs. For experiments with time series data, Google-13 and DSA-19, we use a fixed learning rate of 0.05 and a momentum of 0.9. We do not use weight decay or learning rate scheduling for time-series data.