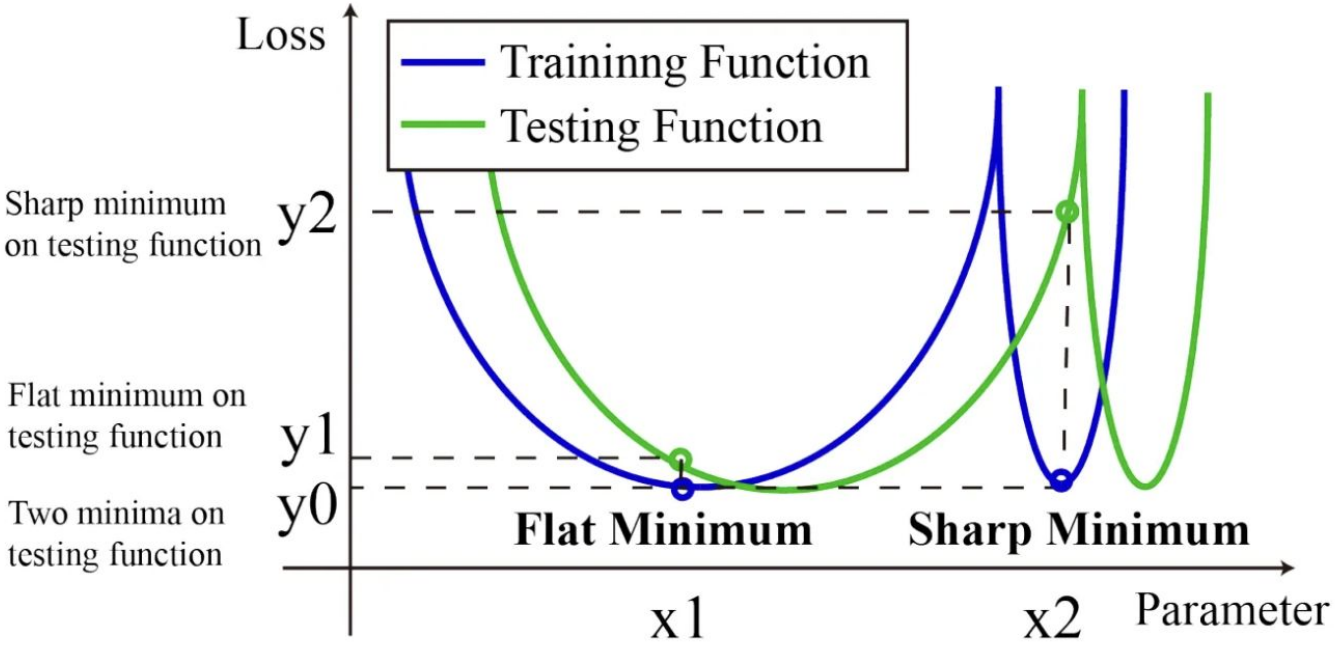
Broad optima and algorithms for finding them

Пискалов Дмитрий 212

Content

1. Broad Optima. Relevance.
2. Stochastic Weight Averaging (SWA)
    a. Algorithm
    b. Empirical evidence
    c. SWA connection to ensembling
    d. Connection to convex minimization
    e. Experiments
3. Sharpness-Aware Minimization (SAM)
    a. Objective
    b. Approximate minization
    c. Algorithm
    d. Experiments
    e. Doubts
4. ASAM
    a. Similar objective and similar minimization
    b. Experiments


Loss
Trainingg Function
Testing Function
Sharp minimum
on testing function
y2
Flat minimum on
testing function
y1
Two minima on
testing function
y0
Flat Minimum
Sharp Minimum
x1
x2
Parameter

Broad Optima relevance

1. Improves generalization
2. Robustness to label noise

Previously we discussed Fast Geometric Ensembling (FGE) approach that gives flatter solutions. But..

Ensembling cons

1. Longer inference
2. Longer fitting

Idea

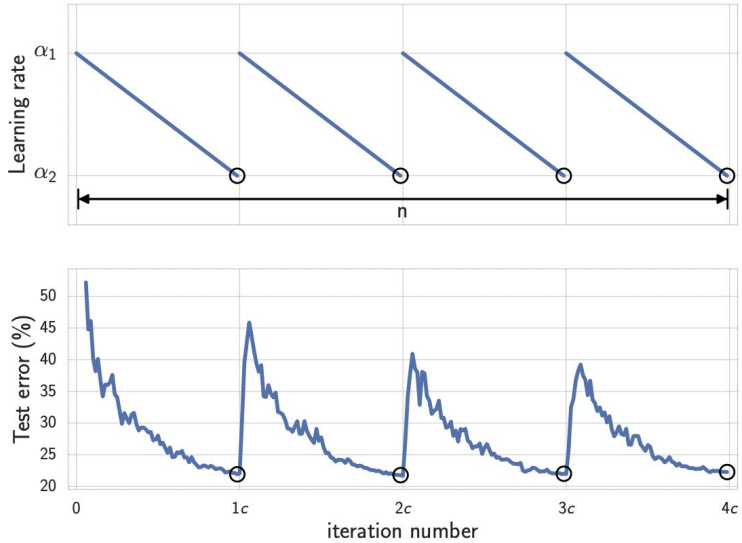
Instead of storing the parameter weights in separate models, the outputs of which we would average during inference, let's create one model with the parameter weight equal to the average of the weights.

Stochastic Weight Averaging (SWA)

Algorithm 1 Stochastic Weight Averaging

Require:
weights $\hat{w}$, LR bounds α_1, α_2,
cycle length c (for constant learning rate $c = 1$), number of iterations n

Ensure: w_{SWA}
$w \leftarrow \hat{w}$ {Initialize weights with $\hat{w}$}
$w_{\text{SWA}} \leftarrow w$
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $\alpha \leftarrow \alpha(i)$ {Calculate LR for the iteration}
  $w \leftarrow w - \alpha \nabla \mathcal{L}_i(w)$ {Stochastic gradient update}
  if $\text{mod}(i, c) = 0$ **then**
    $n_{\text{models}} \leftarrow i/c$ {Number of models}
    $w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1}$ {Update average}
  end if
end for
{Compute BatchNorm statistics for w_{SWA} weights}



1. SGD generally converges to a point near the boundary of the wide flat region of optimal points.
2. SWA on the other hand is able to find a point centered in this region, often with slightly worse train loss but with substantially better test error.

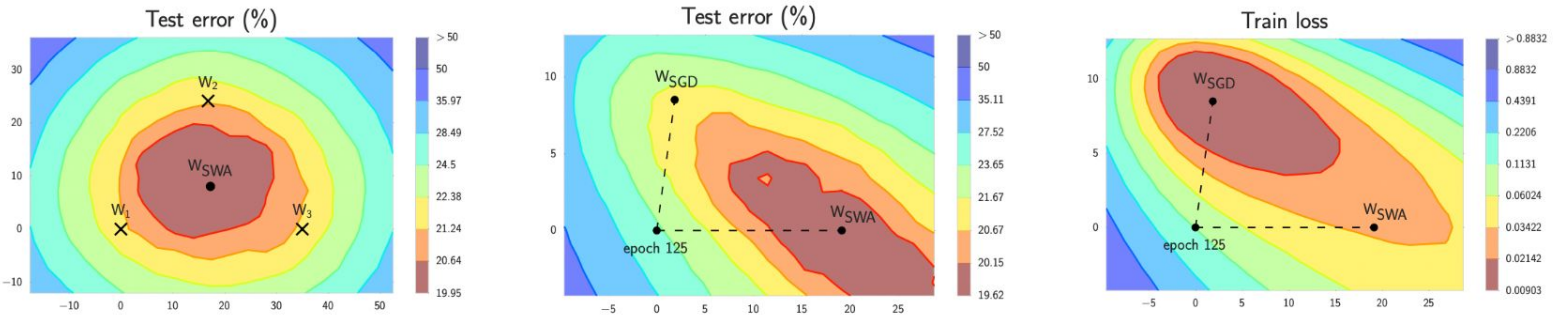


Figure 1: Illustrations of SWA and SGD with a Preactivation ResNet-164 on CIFAR-100[1]. **Left**: test error surface for three FGE samples and the corresponding SWA solution (averaging in weight space). **Middle** and **Right**: test error and train loss surfaces showing the weights proposed by SGD (at convergence) and SWA, starting from the same initialization of SGD after 125 training epochs.

We can empirically verify that the minimum obtained by the algorithm is actually wider.

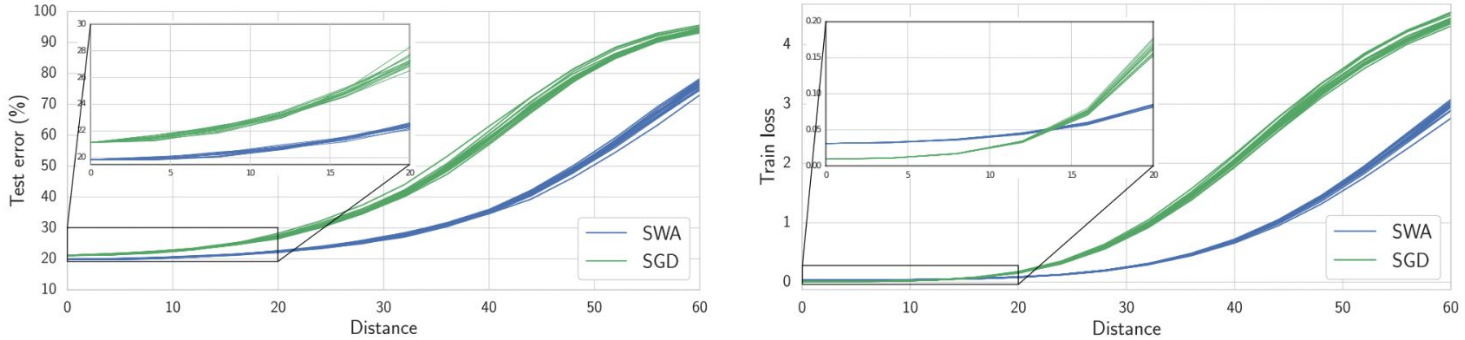


Figure 4: (**Left**) Test error and (**Right**) L_2-regularized cross-entropy train loss as a function of a point on a random ray starting at SWA (blue) and SGD (green) solutions for Preactivation ResNet-164 on CIFAR-100. Each line corresponds to a different random ray.

1. SWA is not finding a different minima than SGD, but rather a flatter region in the same basin of attraction

2. There is a significant asymmetry of the loss function in certain directions, such as the direction SWA to SGD

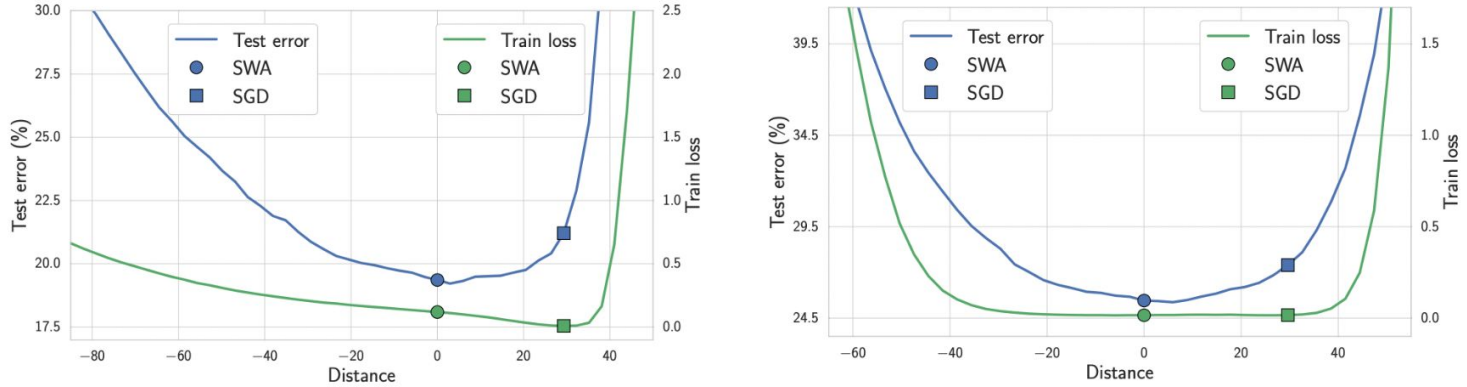


Figure 5: L_2-regularized cross-entropy train loss and test error as a function of a point on the line connecting SWA and SGD solutions on CIFAR-100. **Left**: Preactivation ResNet-164. **Right**: VGG-16.

Connection to ensembling

$$\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f(w_i).$$

$$\bar{f} - f(w_{\mathrm{SWA}}) = \frac{1}{n}\sum_{i=1}^{n}\left(\langle \nabla f(w_{\mathrm{SWA}}), \Delta_i\rangle + O(\|\Delta_i\|^2)\right)$$

$$= \left\langle \nabla f(w_{\mathrm{SWA}}), \frac{1}{n}\sum_{i=1}^{n}\Delta_i \right\rangle + O(\Delta^2) = O(\Delta^2),$$

where $\Delta = \max_{i=1}^{n} \|\Delta_i\|$. Note that the difference between the predictions of different perturbed networks is

$$f(w_i) - f(w_j) = \langle \nabla f(w_{\mathrm{SWA}}), \Delta_i - \Delta_j\rangle + O(\Delta^2),$$

The difference between the predictions of different perturbed network is of the first order of smallness.

The difference between SWA and FCE predictions is of the second order of smallness.

Mandt et al. [2017] showed that under strong simplifying assumptions SGD with a fixed learning rate approximately samples from a Gaussian distribution centered at the minimum of the loss. Suppose this is the case when we run SGD with a fixed learning rate for training a DNN.

Experiments

Table 1: Accuracies (%) of SWA, SGD and FGE methods on CIFAR-100 and CIFAR-10 datasets for different training budgets. Accuracies for the FGE ensemble are from Garipov et al. [2018].

			SWA		
DNN (Budget)	SGD	FGE (1 Budget)	1 Budget	1.25 Budgets	1.5 Budgets
		CIFAR-100			
VGG-16 (200)	72.55 ± 0.10	74.26	73.91 ± 0.12	74.17 ± 0.15	74.27 ± 0.25
ResNet-164 (150)	78.49 ± 0.36	79.84	79.77 ± 0.17	80.18 ± 0.23	80.35 ± 0.16
WRN-28-10 (200)	80.82 ± 0.23	82.27	81.46 ± 0.23	81.91 ± 0.27	82.15 ± 0.27
PyramidNet-272 (300)	83.41 ± 0.21	–	–	83.93 ± 0.18	84.16 ± 0.15
		CIFAR-10			
VGG-16 (200)	93.25 ± 0.16	93.52	93.59 ± 0.16	93.70 ± 0.22	93.64 ± 0.18
ResNet-164 (150)	95.28 ± 0.10	95.45	95.56 ± 0.11	95.77 ± 0.04	95.83 ± 0.03
WRN-28-10 (200)	96.18 ± 0.11	96.36	96.45 ± 0.11	96.64 ± 0.08	96.79 ± 0.05
ShakeShake-2x64d (1800)	96.93 ± 0.10	–	–	97.16 ± 0.10	97.12 ± 0.06

Sharpness-Aware Minimization (SAM)

Instead of the loss function itself, we will minimize the following functional, where ρ means a certain neighborhood.

$$L_S^{SAM}(\boldsymbol{w}) \triangleq \max_{||\boldsymbol{\epsilon}||_p \leq \rho} L_S(\boldsymbol{w} + \boldsymbol{\epsilon})$$

$$\boldsymbol{\epsilon}^*(\boldsymbol{w}) \triangleq \underset{\|\boldsymbol{\epsilon}\|_p \leq \rho}{\arg\max}\, L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\epsilon}) \approx \underset{\|\boldsymbol{\epsilon}\|_p \leq \rho}{\arg\max}\, L_{\mathcal{S}}(\boldsymbol{w}) + \boldsymbol{\epsilon}^T \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w}) = \underset{\|\boldsymbol{\epsilon}\|_p \leq \rho}{\arg\max}\, \boldsymbol{\epsilon}^T \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w}).$$

for approximation, a Taylor series expansion was performed

$$\hat{\boldsymbol{\epsilon}}(\boldsymbol{w}) = \rho\, \text{sign}\left(\nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})\right) \left|\nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})\right|^{q-1} / \left(\left\|\nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})\right\|_q^q\right)^{1/p}$$

where $1/p + 1/q = 1$. Substituting back into equation (1) and differentiating, we then have

$$\begin{aligned}\nabla_{\boldsymbol{w}} L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) &\approx \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w} + \hat{\boldsymbol{\epsilon}}(\boldsymbol{w})) = \frac{d(\boldsymbol{w} + \hat{\boldsymbol{\epsilon}}(\boldsymbol{w}))}{d\boldsymbol{w}} \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(w)} \\ &= \nabla_w L_{\mathcal{S}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})} + \frac{d\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}{d\boldsymbol{w}} \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}.\end{aligned}$$

In general, the calculation of the resulting formula could be easily implemented on any autograd framework by automatically calculating gradients, however, in order to simplify the calculations, the authors propose to discard the second term. So we get

$$\nabla_{\boldsymbol{w}} L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) \approx \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(w)|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}.$$

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}_+$, Batch size b, Step size $\eta > 0$, Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  Compute gradient $\nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})$ of the batch's training loss;
  Compute $\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $\boldsymbol{g} = \nabla_w L_{\mathcal{B}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$;
  Update weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

Algorithm 1: SAM algorithm

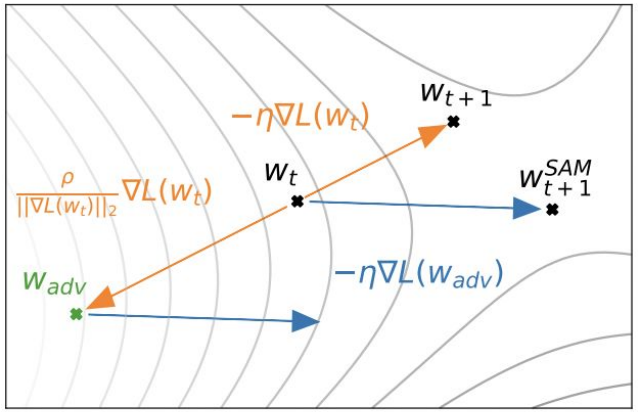
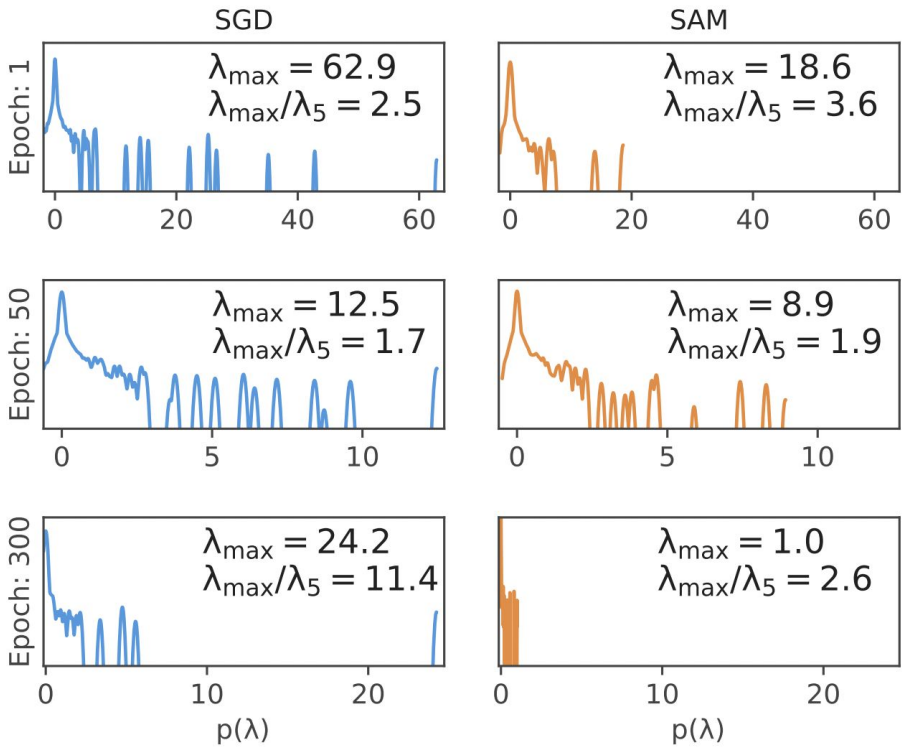


Figure 2: Schematic of the SAM parameter update.

Empirical evidence



The models trained with SAM converge to minima having lower curvature, as seen in the overall distribution of eigenvalues, the

Doubts

SAM calculates the maximum in the neighborhood by measuring the distance in Euclidean terms. We've discussed the problem with this approach in the past.

Therefore, the idea comes to mind to rewrite the distance in terms of filter-wise/element-wise normalization.

Adaptive Sharpness-Aware Minimization (ASAM)

$T_{\mathbf{w}}$ is normalization operator. Examples:

- element-wise

$$T_{\mathbf{w}} = \mathrm{diag}(|w_1|, \ldots, |w_k|)$$

where

$$\mathbf{w} = [w_1, w_2, \ldots, w_k],$$

- filter-wise

$$T_{\mathbf{w}} = \mathrm{diag}(\mathrm{concat}(\|\mathbf{f}_1\|_2 \mathbf{1}_{n(\mathbf{f}_1)}, \ldots, \|\mathbf{f}_m\|_2 \mathbf{1}_{n(\mathbf{f}_m)}, |w_1|, \ldots, |w_q|)) \quad (5)$$

where

$$\mathbf{w} = \mathrm{concat}(\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_m, w_1, w_2, \ldots, w_q).$$

$$\min_{\mathbf{w}} \max_{\|T_{\mathbf{w}}^{-1}\boldsymbol{\epsilon}\|_p \leq \rho} L_S(\mathbf{w}+\boldsymbol{\epsilon}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2.$$

Approximate minization

Minimization of this objective will give exact same result as for SAM.

Algorithm 1 ASAM algorithm ($p = 2$)

Input: Loss function l, training dataset $S := \cup_{i=1}^{n}\{(\mathbf{x}_i, \mathbf{y}_i)\}$, mini-batch size b, radius of maximization region ρ, weight decay coefficient λ, scheduled learning rate α, initial weight $\mathbf{w}_0$.

Output: Trained weight $\mathbf{w}$

Initialize weight $\mathbf{w} := \mathbf{w}_0$

while not converged **do**

Sample a mini-batch B of size b from S

$$\boldsymbol{\epsilon} := \rho \frac{T_{\mathbf{w}}^2 \nabla L_B(\mathbf{w})}{\|T_{\mathbf{w}} \nabla L_B(\mathbf{w})\|_2}$$

$$\mathbf{w} := \mathbf{w} - \alpha \left(\nabla L_B(\mathbf{w} + \boldsymbol{\epsilon}) + \lambda \mathbf{w}\right)^{\dagger}$$

end while

return $\mathbf{w}$

Experiments

Table 3. Maximum test accuracies for SGD, SAM and ASAM on CIFAR-100 dataset.

Model	SGD	SAM	ASAM
DenseNet-121	$68.70_{\pm 0.31}$	$69.84_{\pm 0.12}$	$\mathbf{70.60}_{\pm 0.20}$
ResNet-20	$69.76_{\pm 0.44}$	$71.06_{\pm 0.31}$	$\mathbf{71.40}_{\pm 0.30}$
ResNet-56	$73.12_{\pm 0.19}$	$75.16_{\pm 0.05}$	$\mathbf{75.86}_{\pm 0.22}$
VGG19-BN*	$71.80_{\pm 1.35}$	$73.52_{\pm 1.74}$	$\mathbf{75.80}_{\pm 0.27}$
ResNeXt29-32x4d	$79.76_{\pm 0.23}$	$81.48_{\pm 0.17}$	$\mathbf{82.30}_{\pm 0.11}$
WRN-28-2	$75.28_{\pm 0.17}$	$77.25_{\pm 0.35}$	$\mathbf{77.54}_{\pm 0.14}$
WRN-28-10	$81.56_{\pm 0.13}$	$83.42_{\pm 0.04}$	$\mathbf{83.68}_{\pm 0.12}$
PyramidNet-272†	$88.91_{\pm 0.12}$	$89.36_{\pm 0.20}$	$\mathbf{89.90}_{\pm 0.13}$

References

Averaging Weights Leads to Wider Optima and Better Generalization – https://arxiv.org/pdf/1803.05407.pdf

Sharpness-Aware Minimization for Efficiently Improving Generalization – https://arxiv.org/pdf/2010.01412.pdf

Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks – https://arxiv.org/pdf/2102.11600.pdf