Supplementary Material for MixupE

Yingtian Zou1, Vikas Verma2,3, Sarthak Mittal2, Wai Hoh Tang1, Hieu Pham4, Juho Kannala3
Yoshua Bengio2, Arno Solin3, Kenji Kawaguchi1

1National University of Singapore, Singapore
2Universite de Montreal, Mila, Canada
3Aalto University, Finland
4Google Brain, USA

A NOTATIONS

We denote by $z = (x, y)$ the input and output pair where $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y \in \mathcal{Y} \subseteq \mathbb{R}^C$. Let $f_\theta(x) \in \mathbb{R}^C$ be the output of the logits (i.e., the last layer before the softmax or sigmoid) of the model parameterized by $\theta$. We use $\ell(\theta, z) = h(f_\theta(x)) - y^T f_\theta(x)$ to denote the loss function. Let $g(\cdot)$ be the activation function. We use $x(i)$ to index $i$-th element of the vector $x$ and $x_j$ to represent $j$-th variable in a set. The notation list is:

- $S = \{x_i, y_i\}_{i \in [n]}$ is the fixed training set while $x'$ is the random test sample.
- $\ell$ is the loss function for any data point.
- $L_{\text{mix}}^n(\theta, S)$: empirical risk of Mixup of size $n$ with parameters $\theta$.
- $\mathcal{L}$: empirical risk of MixupE.
- $\Theta$: the constraint set of parameters $\theta$.
- $\mathcal{R}(\Theta, S)$: Empirical Rademacher complexity of set $\Theta$ over training set $S$.
- $J_a(b)$: Jacobian matrix of $a$ w.r.t $b$.

B PROOF OF THEOREM 1

Proof. For the cross-entropy loss, we have

$$\ell(\theta, (x, y)) = -\log \frac{\exp(y^T f_\theta(x))}{\sum_j \exp(f_\theta(x)_j)} = \log \left( \sum_j \exp(f_\theta(x)_j) \right) - y^T f_\theta(x)$$

(1)

where $y \in \mathbb{R}^C$ is a one-hot vector. For the logistic loss,

$$\ell(\theta, (x, y)) = -\log \frac{\exp(y f_\theta(x))}{1 + \exp(f_\theta(x))} = \log (1 + \exp(f_\theta(x)) - y f_\theta(x).$$

(2)

Thus, for both cases, we can write

$$\ell(\theta, (x, y)) = h(f_\theta(x)) - y^T f_\theta(x)$$

(3)

where $h(z) = \log \left( \sum_j \exp(z_j) \right)$ for the cross-entropy loss and $h(z) = \log(1 + \exp(z))$ for the logistic loss. Using this and equation (9) of [Zhang et al., 2021], we have that

$$L_{\text{mix}}^n(\theta, S) = \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{\lambda \sim \mathcal{D}_x} \mathbb{E}_{x' \sim \mathcal{D}_x} l(\theta, (r_i(x'), y_i)),$$

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).
where $\mathcal{D}_X$ is the empirical distribution induced by training samples, and

$$r_i(x) = \lambda x_i + (1 - \lambda) x.$$  \hfill (4)

Define $a_\lambda = 1 - \lambda$. Then,

$$r_i(x') = (1 - a_\lambda) x_i + a_\lambda x' = x_i + a_\lambda (x' - x_i).$$  \hfill (5)

Define

$$\phi_i(a_\lambda) := f_\theta(x_i + a_\lambda (x' - x_i))$$  \hfill (6)

Assume $f_\theta$ lies in the $C^K$ manifold ($K$-times differentiable), then there exists a function $\psi_i$ such that $\lim_{a_\lambda \to 0} \psi_i(a_\lambda) = 0$ and with Taylor expansion at $a_\lambda = 0$, we have

$$\phi_i(a_\lambda) = \phi_i(0) + \sum_{k=1}^{K} \frac{a_\lambda^k}{k!} \phi_i^{(k)}(0) + a_\lambda^k \psi_i(a_\lambda)$$

$$= f_\theta(x_i) + \sum_{k=1}^{K} \frac{a_\lambda^k}{k!} \phi_i^{(k)}(0) + a_\lambda^k \psi_i(a_\lambda)$$  \hfill (7)

where $\phi_i^{(k)}(0)$ is the $k$-th order derivative at $a_\lambda = 0$, $\psi_i(a_\lambda)$ is the remainder term:

$$\psi_i(a_\lambda) = \int_{\mathbb{R}} \phi_i^{(K)}(a_\lambda) da_\lambda - \frac{1}{k!} \phi_i^{(K)}(0)$$  \hfill (8)

Here, for any $k \in \mathbb{N}^+$, we have

$$\phi_i^{(k)}(0) = \phi_i^{(k)}(a_\lambda)|_{a_\lambda=0} = \frac{\partial^k f_\theta(x_i + a_\lambda(x' - x_i))}{\partial (x_i + a_\lambda(x' - x_i))^k} (x' - x_i) \otimes k$$

$$= \frac{\partial^k f_\theta(x_i)}{\partial (x_i)^k} (x' - x_i) \otimes k$$  \hfill (9)

where $\otimes$ denotes Kronecker product and thus $(x' - x_i) \otimes k \in \mathbb{R}^{d_k}$. We can then rewrite $\phi_i^{(k)}(0)$ as

$$\phi_i^{(k)}(0) = J_{f_\theta}^k(x_i)(x' - x_i) \otimes k$$  \hfill (10)

Plug back into the (7), we have

$$f_\theta(x_i + a_\lambda(x' - x_i)) = f_\theta(x_i) + \sum_{k=1}^{K} \frac{a_\lambda^k}{k!} J_{f_\theta}^k(x_i)(x' - x_i) \otimes k + a_\lambda^k \psi_i(a_\lambda)$$

$$= f_\theta(x_i) + a_\lambda \left( \sum_{k=1}^{K} \frac{a_\lambda^{k-1}}{k!} J_{f_\theta}^k(x_i)(x' - x_i) \otimes k + a_\lambda^{K-1} \psi_i(a_\lambda) \right) \Delta_i$$  \hfill (11)

Above equation will be

$$\ell(\theta, (r_i(x), y_i)) = \ell[\theta, (x_i + a_\lambda(x' - x_i), y_i)]$$

$$= h(f_\theta(x_i + a_\lambda(x' - x_i))) - y_i^T f_\theta(x_i + a_\lambda(x' - x_i))$$

$$= h(f_\theta(x_i) + a_\lambda \Delta_i) - y_i^T (f_\theta(x_i) + a_\lambda \Delta_i).$$  \hfill (12)

Analogously, we can define $\hat{\phi}_i^{(k)}(a_\lambda) := h(f_\theta(x_i) + a_\lambda \Delta_i)$ and the parallel notation $\hat{\psi}_i(a_\lambda)$, then

$$h(f_\theta(x_i) + a_\lambda \Delta_i) = h(f_\theta(x_i)) + \sum_{k=1}^{K} \frac{a_\lambda^k}{k!} J_{h \circ f_\theta}^k(x_i) \Delta_i \otimes k + a_\lambda^k \hat{\psi}_i(a_\lambda)$$  \hfill (13)
Combining these,

\[ \ell(\theta, (r_i(x), y_i)) = h(f_\theta(x_i)) - y_i^T f_\theta(x_i) - a_\lambda y_i \Delta_i + \sum_{k=1}^{K} \frac{a_k}{k!} J_{h_\theta f_\theta}(x_i) \Delta_i^{\otimes k} + a_\lambda^K \dot{\psi}_i(a_\lambda) \]

Thus, the implicit regularization of Mixup can be unfolded as

\[ L_n^{\text{mix}}(\theta, S) = \frac{1}{n} \sum_{i=1}^{n} E_{\lambda \sim D_x} E_{\epsilon_i \sim D_\epsilon} \ell(\theta, (r_i(x), y_i)) \]

\[ = L_n^{\text{std}}(\theta, S) + \frac{1}{n} \sum_{i=1}^{n} E_{\lambda \sim D_x} E_{\epsilon_i \sim D_\epsilon} \left( \sum_{k=1}^{K} \frac{a_k}{k!} J_{h_\theta f_\theta}(x_i) \Delta_i^{\otimes k} - a_\lambda y_i^T \Delta_i + a_\lambda^K \dot{\psi}_i(a_\lambda) \right) \]

where

\[ \Delta_i = \sum_{k=1}^{K} \frac{a_k^{k-1}}{k!} J_{f_\theta}(x_i) (x' - x_i)^{\otimes k} + a_\lambda^{K-1} \psi_i(a_\lambda). \]

Note that with probability 1, we have

\[ \lim_{a_\lambda \to 0} \dot{\psi}_i(a_\lambda) = 0, \quad \lim_{a_\lambda \to 0} \psi_i(a_\lambda) = 0 \]

C PROOF OF THEOREM 2

The Rademacher generalization bound is widely applied where the empirical Rademacher complexity of a function class \( \Theta \) is given by:

\[ R_n(\Theta, \{x_i\}_{i=1}^{n}) = \mathbb{E} \left[ \sup_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} f_\theta(x_i) \epsilon_i \right] \]

where, Rademacher r.v \( \epsilon_i \) independently takes values in \( \{-1, +1\} \) with equal probability.

Lemma 1. (Bartlett and Mendelson [2002]). For any \( B \)-uniformly bounded and \( L \) Lipchitz function \( \zeta \), for all \( \phi \in \Phi \), with probability at least \( 1 - \delta \),

\[ \mathbb{E} \zeta(\phi(x_i)) \leq \frac{1}{n} \sum_{i=1}^{n} \zeta(\phi(x_i)) + 2LR_n(\Phi, S) + B \sqrt{\frac{\log(1/\delta)}{2n}} \]

Proof. Consider GLM that \( h(f_\theta(x)) = A(\theta^T x) \) and training set \( S \), the constraint of \( \Theta = \{ x \to f_\theta(x) \sup_x \hat{g}(x) \leq \gamma \} \) implies that

\[ \sup_x |\hat{g_i}(x)| = \sup_x (y - A'(\theta^T x))^T (\theta^T x) \leq \gamma \]

Rearranging the terms, and by Cauchy–Schwarz inequality we have

\[ \gamma \geq \sup_x (y - A'(\theta^T x))^T (\theta^T x) \]

\[ = \sup_x (y, \theta^T x) - \sup_x (A'(\theta^T x), \theta^T x) \]

\[ \geq \sup_x (y, \theta^T x) - \sup_x \| A'(\theta^T x) \|_2 \| \theta^T x \|_2 \]

Due to the fact that \( A(\cdot) \) is a \( L_A \) Lipchitz function, then it’s trivial to prove

\[ \| A'(\theta^T x) \|_2 \leq L_A \]
Let \( y = (\theta^*)^\top x = (\Sigma\theta)^\top x \) where \( \Sigma \) is the diagonal matrix. Thus the above relation will be

\[
\gamma \geq \sup_x \langle y, \theta^\top x \rangle - \sup_x \|A'(\theta^\top x)\|_2 \|\theta^\top x\|_2
\]

\[
\geq \sup_x \langle (\Sigma\theta)^\top x, \theta^\top x \rangle - L_A \sup_x \|\theta^\top x\|_2
\]

(21)

Let \( v = \sup_x \theta^\top x \) and \( \sigma \) be the expected value that \( \sigma = E_j \in [d] \sup_x \Sigma_i(j) = \sup_x \frac{\text{tr}(\Sigma)}{d} \), then we have

\[
\gamma \geq \sigma \|v\|_2^2 - L_A \|v\|_2
\]

(22)

which implies

\[
\frac{L_A - \sqrt{L_A^2 + 4\gamma \sigma}}{2\sigma} \leq \|v\|_2 \leq \frac{L_A + \sqrt{L_A^2 + 4\gamma \sigma}}{2\sigma}
\]

(23)

Obviously,

\[
\left| \frac{L_A + \sqrt{L_A^2 + 4\gamma \sigma}}{2\sigma} \right| > \frac{L_A + \sqrt{L_A^2 - 4\gamma \sigma}}{2\sigma}
\]

(24)

Denote \( v_i = \theta^\top x_i \), we have the Rademacher complexity \( R(\Theta, S) \) that

\[
R(\Theta, S) = E_{\epsilon} \sup_{E_{x \hat{q}(x) \leq \gamma}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i \theta^\top x_i
\]

\[
\leq E_{\epsilon} \sup_{\|v_i\|_2^2 \leq \left( \frac{L_A + \sqrt{L_A^2 + 4\gamma \sigma}}{2\sigma} \right)} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i v_i
\]

\[
\leq \frac{1}{\sqrt{n}} \frac{L_A + \sqrt{L_A^2 + 4\gamma \sigma}}{2\sigma} \sqrt{E_{\epsilon} \left( \sum_{i=1}^{n} \epsilon_i \right)^2}
\]

\[
= \frac{1}{\sqrt{n}} \frac{L_A + \sqrt{L_A^2 + 4\gamma \sigma}}{2\sigma}
\]

\[
\leq \frac{1}{\sqrt{n}} \frac{2L_A + 2\sqrt{\gamma \sigma}}{2\sigma}
\]

\[
= \frac{L_A + \sqrt{\gamma \sigma}}{\sigma \sqrt{n}}
\]

(25)

Consequently, we have

\[
R(\Theta, S) \leq \frac{L_A + \sqrt{\gamma \sigma}}{\sigma \sqrt{n}}
\]

(26)

Recall the objective of MixupE.

\[
L(\theta, S) := \hat{\eta} \left( L_n^{\text{mix}}(\theta, S) + \eta R(\theta, S) \right)
\]

\[
\hat{\eta} = \frac{|L_n^{\text{mix}}(\theta, S)|}{|L_n^{\text{mix}}(\theta, S) + \eta R(\theta, S)|}
\]

(27)

(28)

With Lemma 1, we can get

\[
L(\theta, S) \leq \hat{\eta} L_n^{\text{mix}}(\theta, S) + 2\hat{\eta}LL_A(\Theta, S) + B \sqrt{\frac{1}{2n}} \log(1/\delta)
\]

\[
\leq \hat{\eta} L_n^{\text{mix}}(\theta, S) + 2\hat{\eta}LL_A \frac{L_A + \sqrt{\gamma \sigma}}{\sigma \sqrt{n}} + B \sqrt{\frac{1}{2n}} \log(1/\delta)
\]

(29)
C.1 COMPARISON TO VANILLA MIXUP

As a comparison, for vanilla Mixup with parameter space \( \hat{\Theta} = \{ \theta | \| \theta \|_2^2 \leq \gamma \} \) and assume \( \| x_i \|_2^2 \leq \mathcal{X}, \forall i \in [n] \) the Rademacher complexity will be

\[
R(\hat{\Theta}, S) = \mathbb{E}_\epsilon \sup_{\| \theta \|_2^2 \leq \gamma} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i \theta^\top x_i
\]

\[
= \frac{1}{n} \mathbb{E}_\epsilon \sup_{\| \theta \|_2^2 \leq \gamma} \sqrt{\sum_{i=1}^{n} \epsilon_i^2 \| \theta \|_2^2 \| x_i \|_2^2}
\]

\[
= \frac{\sqrt{\gamma}}{n} \mathbb{E}_\epsilon \sqrt{\sum_{i=1}^{n} \epsilon_i^2 \| x_i \|_2^2} \leq \frac{\sqrt{\gamma \mathcal{X}}}{\sqrt{n}}
\]

(30)

Compared to the Rademacher complexity of Mixup, we found that MixupE don’t need to bound the norm of input data by \( \mathcal{X} \) which may cause a large term. However, if considering normalized input space where \( \mathcal{X} \leq 1 \), the condition to have a shrink parameter space is

\[
\frac{L_A + \sqrt{\gamma \sigma}}{\sigma} \leq \sqrt{\gamma} \Rightarrow \frac{L_A}{\sigma - \sqrt{\gamma}} \leq \sqrt{\gamma} \text{ and } \sigma > 1
\]

(31)

Thus, when the above condition is satisfied, our regularization reduces the norm of parameter space for the case where input space is normalized \( \mathcal{X} \leq 1 \). In general, the \( \sigma \) is the average entry value of the maximum correction matrix to the ground truth which can be quite large. Scaling by \( \sigma \), it is probably satisfied in most cases.

D IMPLEMENTATION

The code implementation in PyTorch is shown as Listing 1.

```python
def beta_mean(alpha, beta):
    return alpha / (alpha + beta)

lam_mod_mean = beta_mean(alpha+1, alpha) # mean of beta distribution

# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1.0 - lam) * x2)
    y = Variable(lam * y1 + (1.0 - lam) * y2)
    loss = loss_function(net(x), y) # mixup loss
    loss_scale = torch.abs(loss.detach().data.clone())
    f = net(x1)
    b = y1 - torch.softmax(f, dim=1)
    loss_new = torch.sum(f == b, dim=1)
    loss_new = (1.0 - lam_mod_mean) * torch.sum(torch.abs(loss_new)) / batch_size # additional loss term
    loss = loss + (mixup_eta * loss_new) # total loss
    loss_new_scale = torch.abs(loss.detach().data.clone())
    loss = (loss_scale / loss_new_scale) * loss # loss after scaling
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

Listing 1: One epoch MixupE training in PyTorch
References
