FIXING WEIGHT DECAY REGULARIZATION IN ADAM

Ilya Loshchilov & Frank Hutter

University of Freiburg

Repoter: Yang Liu

作者

Ilya Loshchilov

Frank Hutter

SGD

- $g_t = \nabla_{\theta_{t-1}} f(\theta_{t-1})$
- $\Delta\theta_t = -\eta * g_t$

- 选择合适的learning rate比较困难（所有的参数更新使用同样的learning rate，稀疏数据或者特征更新慢）
- SGD容易收敛到局部最优

SGD with Momentum

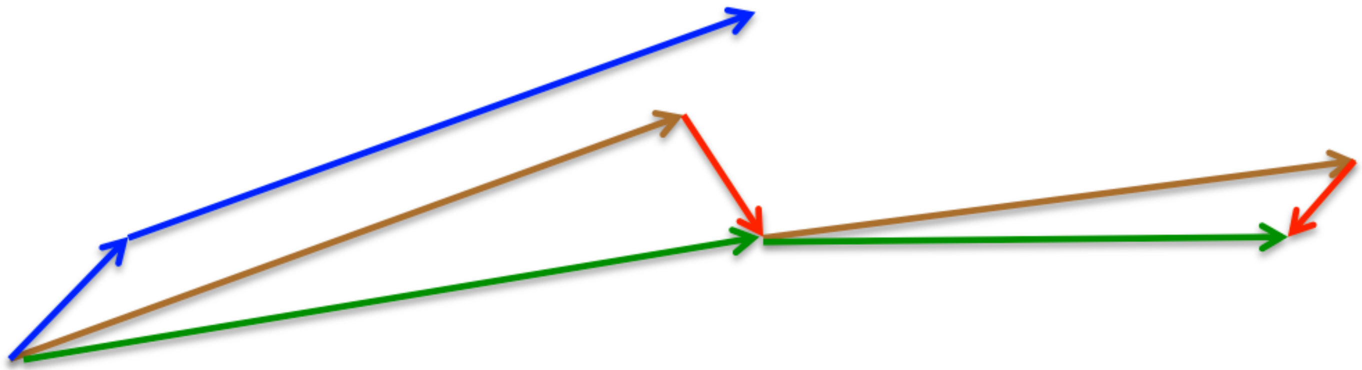
- $g_t = \nabla_{\theta_{t-1}} f(\theta_{t-1})$
- $m_t = \mu * m_{t-1} + g_t$
- $\Delta\theta_t = -\eta * m_t$

- 下降初期时，使用上一次参数更新，下降方向一致，乘上较大的μ能够进行很好的加速
- 下降中后期时，在局部最小值来回震荡的时候，使得更新幅度增大，跳出陷阱
- 能够在相关方向加速SGD，抑制振荡，从而加快收敛

Nesterov

- $g_t = \nabla_{\theta_{t-1}} f(\theta_{t-1} - \eta * \mu * m_{t-1})$
- $m_t = \mu * m_{t-1} + g_t$
- $\Delta\theta_t = -\eta * m_t$

- momentum首先计算一个梯度(短的蓝色向量)，然后在加速更新梯度的方向进行一个大的跳跃(长的蓝色向量)，nesterov项首先在之前加速的梯度方向进行一个大的跳跃(棕色向量)，计算梯度然后进行校正(绿色梯向量)



Adagrad

- $n_t = n_{t-1} + g_t^2$
- $\Delta\theta_t = -\frac{\eta}{\sqrt{n_t+\epsilon}} * g_t$

- 适合处理稀疏梯度，梯度更新次数少的稀疏特征，n_t小，更新更快一些。
- 同样可以做到前期放大梯度，后期缩小梯度。对于每个参数。这个前期后期的时刻可能不一样。
- 问题：
  - 仍依赖于人工设置全局学习率
  - 中后期，分母上梯度平方的累加将会越来越大，导致训练提前结束（收敛到任意位置）

Ada Δ

- $n_t = v * n_{t-1} + (1 - v) * g_t^2$
- $\Delta\theta_t = -\frac{\eta}{\sqrt{n_t + \epsilon}} * g_t$

- $E|g^2|_t = v * E|g^2|_{t-1} + (1 - v) * g_t^2$
- $\Delta\theta_t = -\frac{\sqrt{\sum_{r=1}^{t-1} \Delta\theta_r}}{\sqrt{E|g^2|_t + \epsilon}}$

- Adadelta已经不用依赖于全局学习率
- 训练初中期，加速效果不错，很快
- 训练后期，反复在局部最小值附近抖动

RMSprop

- $E|g^2|_t = v * E|g^2|_{t-1} + (1-v) * g_t^2$
- 当 $v = \frac{t-1}{\mathrm{t}}$ 变为求均方
- 均方根 $RMS|g|_t = \sqrt{E|g^2|_t + \epsilon}$
- $\Delta\theta_t = -\frac{\eta}{RMS|g|_t} * g_t$

- 依赖于全局学习率
- RMSprop算是Adagrad的一种发展，和Adadelta的变体，效果趋于二者之间
- 适合处理非平稳目标 - 对于RNN效果很好

Adam

- $m_t = \mu * m_{t-1} + (1 - \mu) * g_t$
- $n_t = v * n_{t-1} + (1 - v) * g_t^2$
- $\widehat{m_t} = \frac{m_t}{1-\mu^t}$
- $\widehat{n_t} = \frac{n_t}{1-v^t}$
- $\Delta\theta_t = -\frac{\widehat{m_t}}{\sqrt{\widehat{n_t}}+\epsilon} * \eta$
- $\mu = 0.9, v = 0.999, \eta = 10^{-8}$

- 善于处理稀疏梯度、非平稳目标的优点
- 对内存需求较小
- 为不同的参数计算不同的自适应学习率
- 也适用于大多非凸优化 - 适用于大数据集和高维空间

Adamax

- $m_t = \mu * m_{t-1} + (1-\mu) * g_t$
- $n_t = \max(v * n_{t-1}, |g_t|)$
- $\widehat{m_t} = \frac{m_t}{1-\mu^t}$
- $\Delta\theta_t = -\frac{\widehat{m_t}}{n_t+\epsilon} * \eta$
- $\mu = 0.9, v = 0.999, \eta = 10^{-8}$

- Adamax学习率的边界范围更简单

Nadam

$$\hat{g}_t = \frac{g_t}{1 - \Pi_{i=1}^{t}\mu_i}$$

$$m_t = \mu_t * m_{t-1} + (1 - \mu_t) * g_t$$

$$\hat{m}_t = \frac{m_t}{1 - \Pi_{i=1}^{t+1}\mu_i}$$

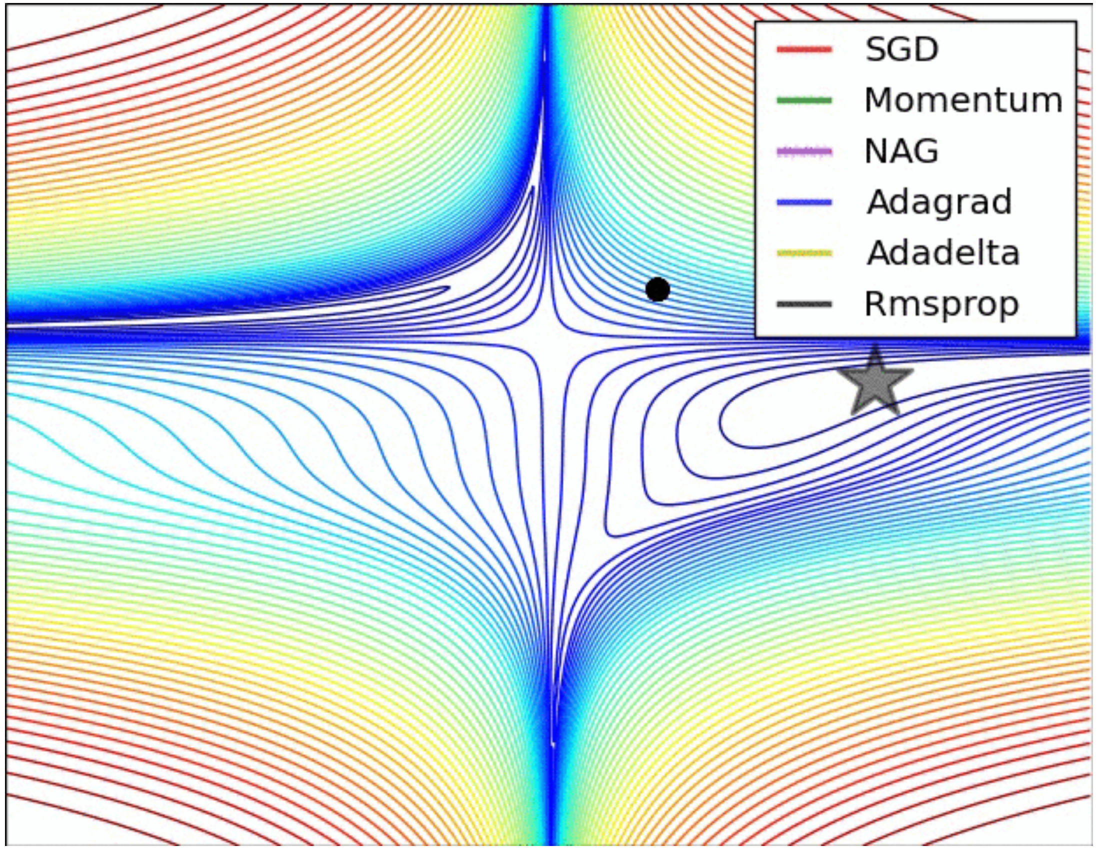
$$n_t = \nu * n_{t-1} + (1 - \nu) * g_t^2$$

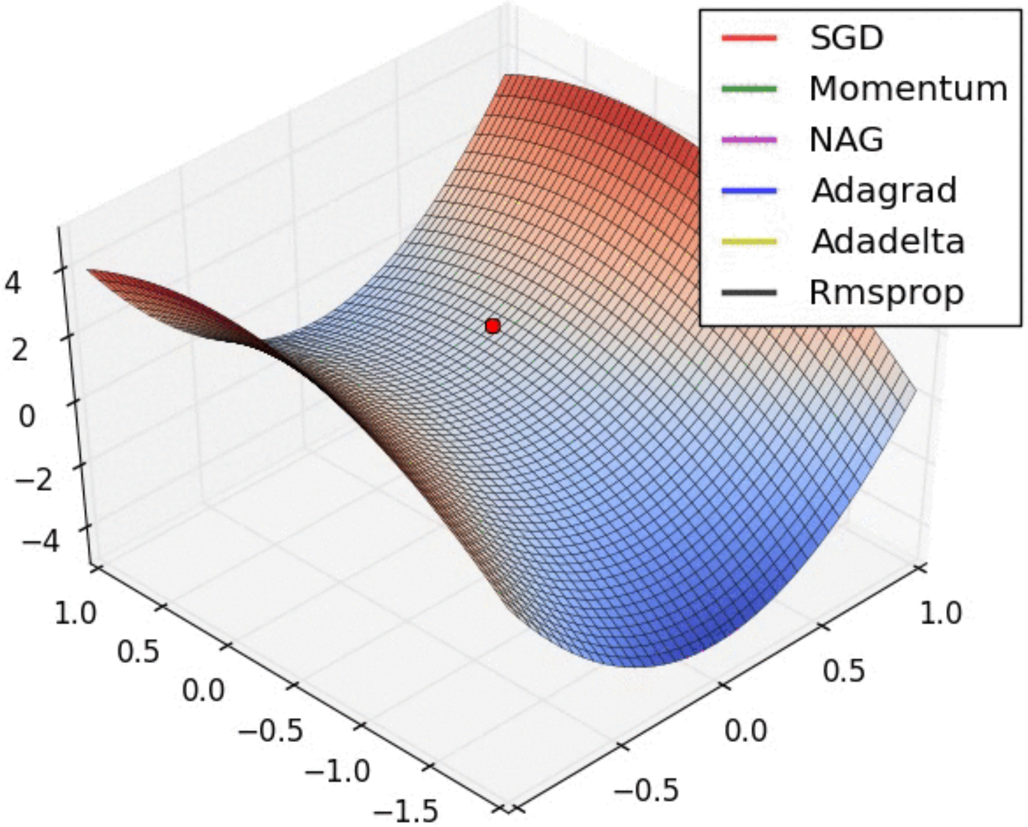
$$\hat{n}_t = \frac{n_t}{1 - \nu^t} \quad \bar{m}_t = (1 - \mu_t) * \hat{g}_t + \mu_{t+1} * \hat{m}_t$$

$$\Delta\theta_t = -\eta * \frac{\bar{m}_t}{\sqrt{\hat{n}_t} + \epsilon}$$

- 可以看出，Nadam对学习率有了更强的约束，同时对梯度的更新也有更直接的影响。一般而言，在想使用带动量的RMSprop，或者Adam的地方，大多可以使用Nadam取得更好的效果。

比较


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

Weight Decay

- $\mathrm{x}_{i+1} = (1 - w_i)x_t - \eta \frac{\partial E}{\partial w_i}$

L2 regularization

- $f_{reg}(\theta_t) = f(\theta_t) + \frac{w_t}{2} \|x_t\|_2^2$

Weight Decay

- Implementation

Original: $w \leftarrow w - \eta \frac{\partial L}{\partial w}$

$\lambda = 0.01$

Weight Decay: $w \leftarrow \underbrace{\boxed{0.99}}_{\text{Smaller and smaller}} w - \eta \frac{\partial L}{\partial w}$

Keras: http://keras.io/regularizers/

SGDW

- 梯度下降的是 $\alpha w_t x_{t-1}$ 而不是公式1中给出的 $w_t x_{t-1}$
- 耦合了学习率和 weight decay

Algorithm 1 SGD with momentum and SGDW with momentum

given learning rate $\alpha \in \mathbb{R}$, momentum factor $\beta_1 \in \mathbb{R}$, weight decay factor $w \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{x}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{x}_{t-1}) \leftarrow SelectBatch(\boldsymbol{x}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{x}_{t-1})$ $+w_t \boldsymbol{x}_{t-1}$
  $\eta_t \leftarrow SetScheduleMultiplier(t)$ ▷ can be fixed, decay, be used for warm restarts
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + \eta_t \alpha_t \boldsymbol{g}_t$
  $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} - \boldsymbol{m}_t$ $-\eta_t w_t \boldsymbol{x}_{t-1}$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{x}_t$

AdamW

- 当t很大时。有如下迭代公式。可以看出，weight decay项和梯度一起被归一化了。
- 导致大梯度的weight decay也变弱了。实际上这是不正确的。weight decay应该依赖其值，而不是梯度。
- 作为L2正则，这种实现是正确的。

Algorithm 2 Adam and AdamW

given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, w \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{x}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{x}_{t-1}) \leftarrow SelectBatch(\boldsymbol{x}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{x}_{t-1})$ $+w_t\boldsymbol{x}_{t-1}$
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$ ▷ here and below all operations are element-wise
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2$
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t/(1-\beta_1^t)$ ▷ here, β_1 is taken to the power of t
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t/(1-\beta_2^t)$ ▷ here, β_2 is taken to the power of t
  $\eta_t \leftarrow SetScheduleMultiplier(t)$ ▷ can be fixed, decay, be used for warm restarts
  $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} - \eta_t \left(\alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) +w_t\boldsymbol{x}_{t-1} \right)$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{x}_t$

$$\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} - \eta_t \alpha \frac{\beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t}{\sqrt{\beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2} + \epsilon}, \quad \text{with } \boldsymbol{g}_t = \nabla f_t(\boldsymbol{x}_{t-1}) + w_t \boldsymbol{x}_{t-1}.$$

NORMALIZED WEIGHT DECAY

- b_t : batch 大小
- B ：总样本数
- T_i : 总共经过的epoch数

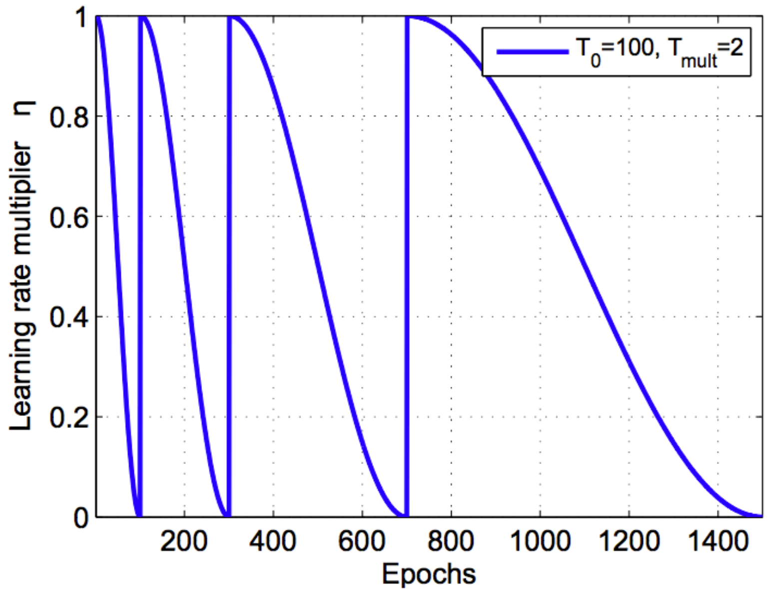
- $w_t = w_{norm}\sqrt{\frac{b_t}{BT_i}}$

Warm restart

- η_{max}, η_{min}：每次重启时可以调整的超参数，实验中设置为1和0
- T_i：i次重启中总共运行的epoch
- T_{cur}:本次重启的epoch数

$$\eta_t = \eta_{min}^{(i)} + 0.5(\eta_{max}^{(i)} - \eta_{min}^{(i)})(1 + \cos(\pi T_{cur}/T_i)),$$

$$\eta_t = 0.5 + 0.5\cos(\pi T_{cur}/T_i).$$

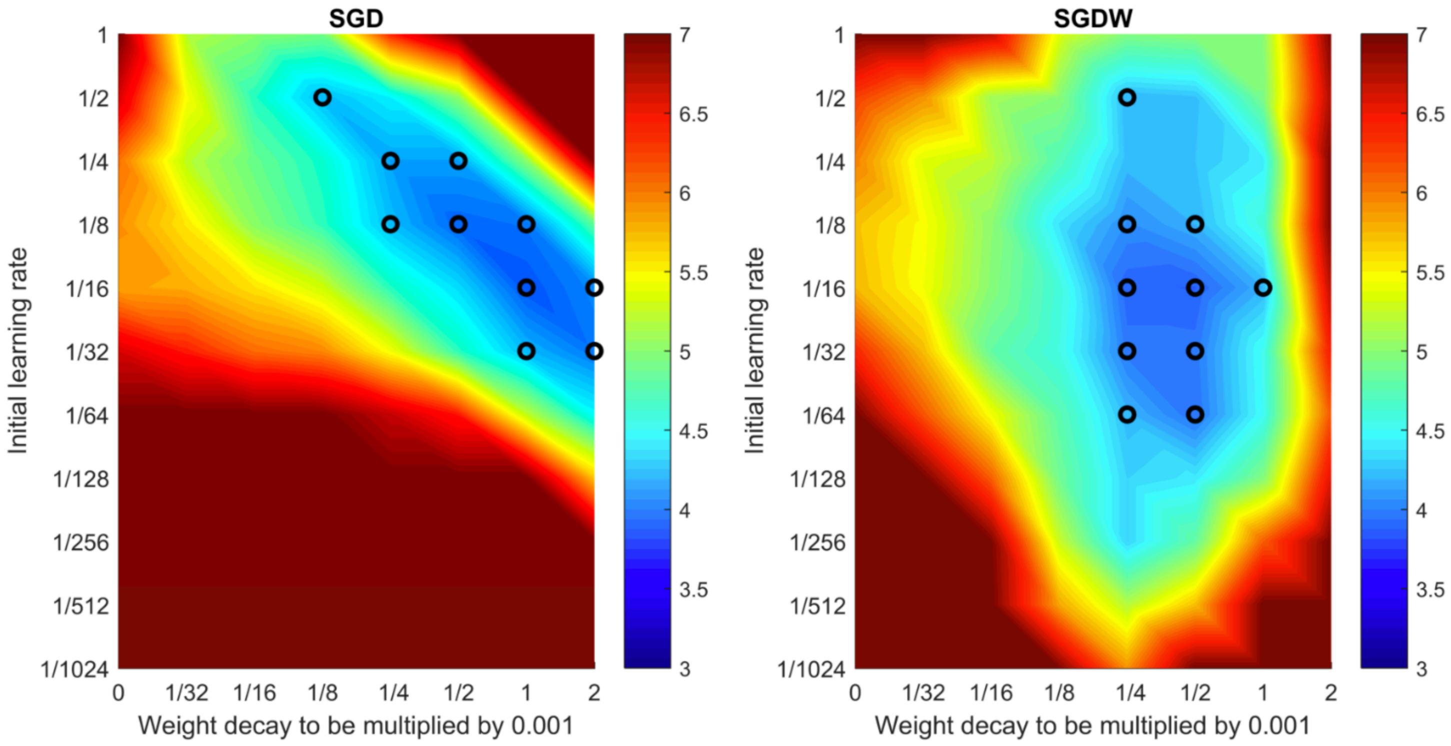


实验

- 数据集、任务：CIFAR-10、CIFAR-100
- 使用 Shake-Shake regularization
- SGDR with batch size 128 for 1800 epochs (T0 = 1800) 不重启动
- 26 2x64d ResNet 26层 2残差分支 第一个残差分支64宽
- 26 2x96d

实验

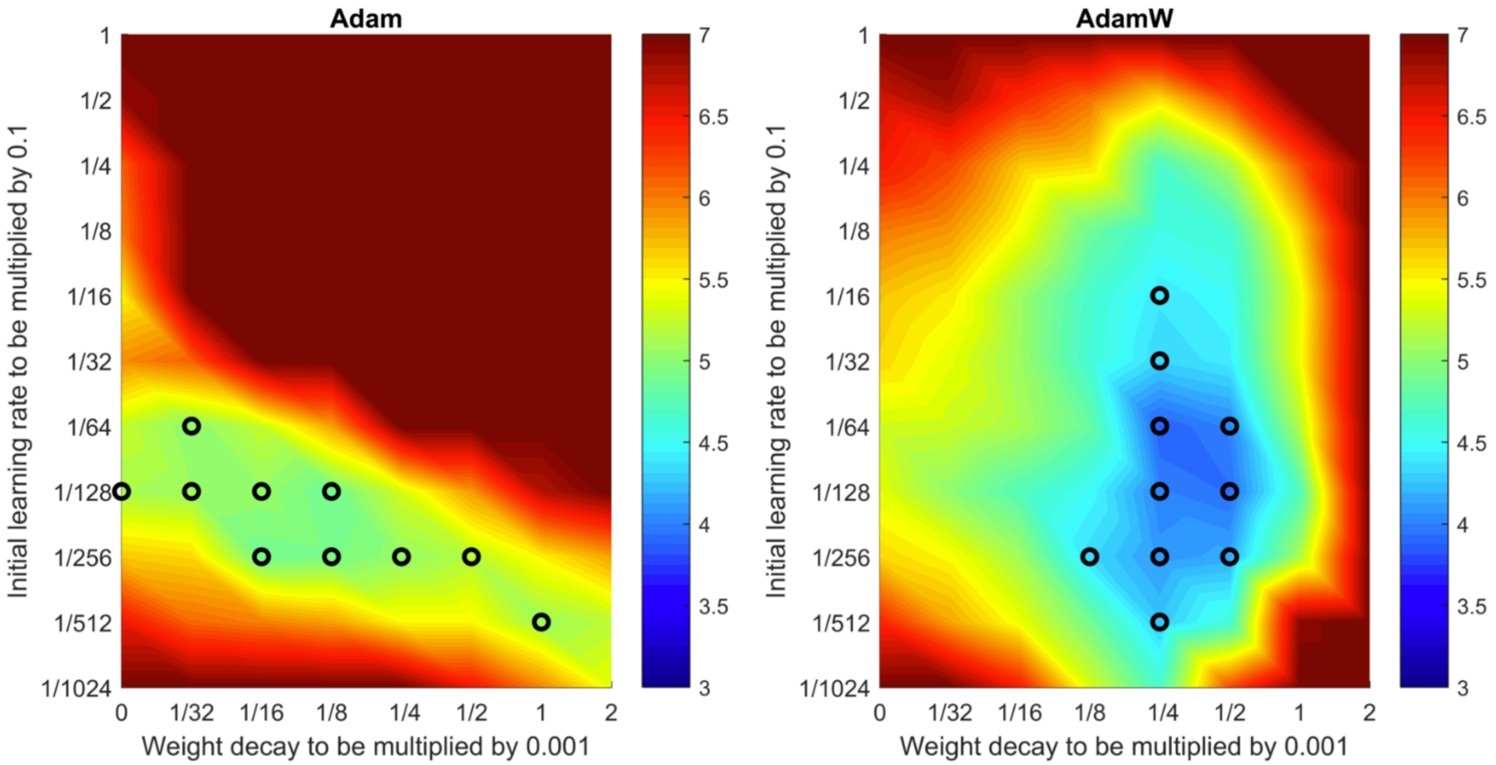
- SGDW weight decay和learning rate 解耦，更好调整两个超参数。
- 即使在不可能达到最优的学习率前提下1/1024也可以得到最好的weight decay



26 2x64d ResNet on CIFAR-10 measured after 100 epochs.

实验

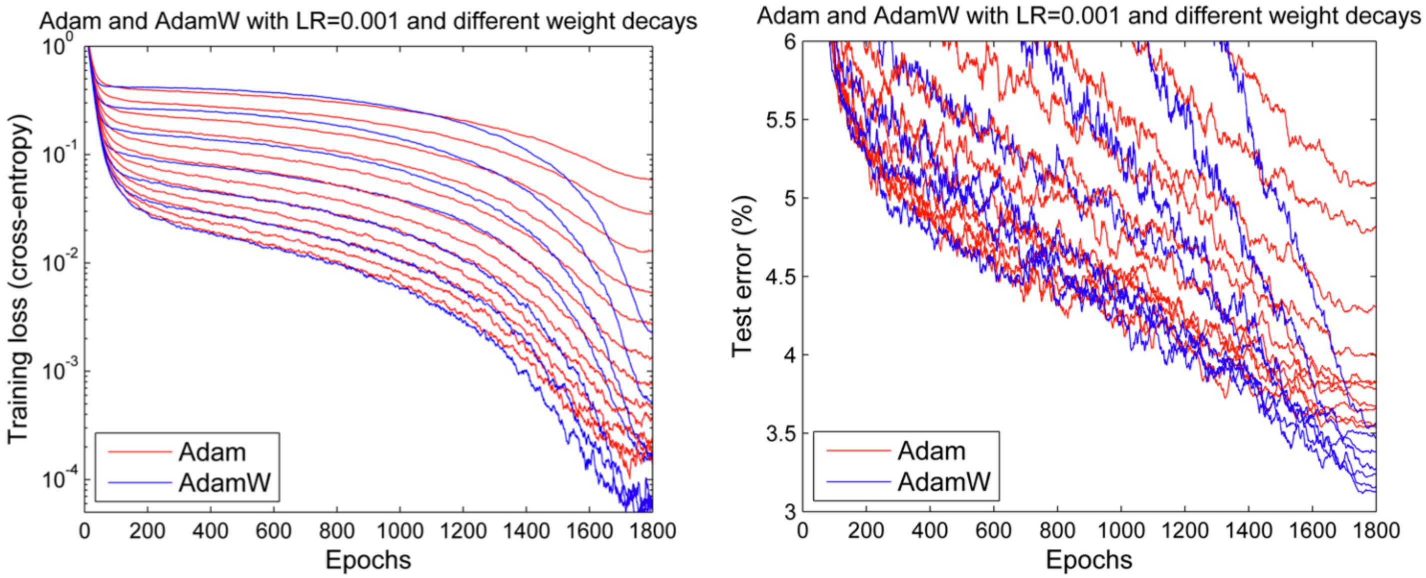
- Adam 比 动量SGD最好结果差
- Adam没有在Weight decay上收益
- Adam 两个超参数耦合
- AdamW 解耦、能与SGD、SGDW方法得到相同最好结果



26 2x64d ResNet on CIFAR-10 measured after 100 epochs. 2

实验

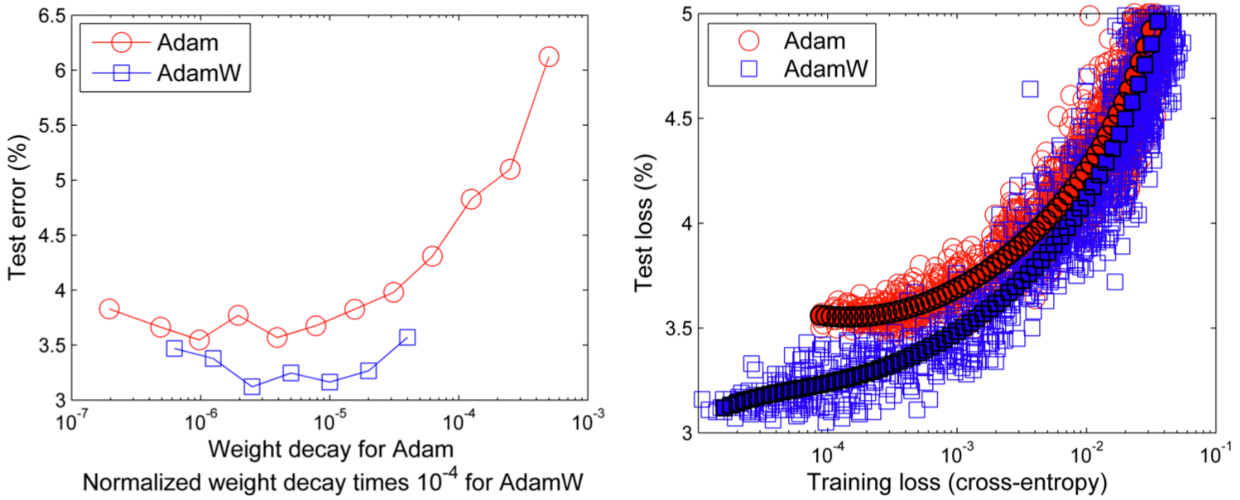
- AdamW 比 Adam，前期基本一致，后期收敛速度快，结果好。



训练更多轮次

实验

- AdamW 比 Adam，前期基本一致，后期收敛速度快，结果好。更好的收敛和范化能力。



训练更多轮次

实验

- 热重启达到更快，更好的性能。

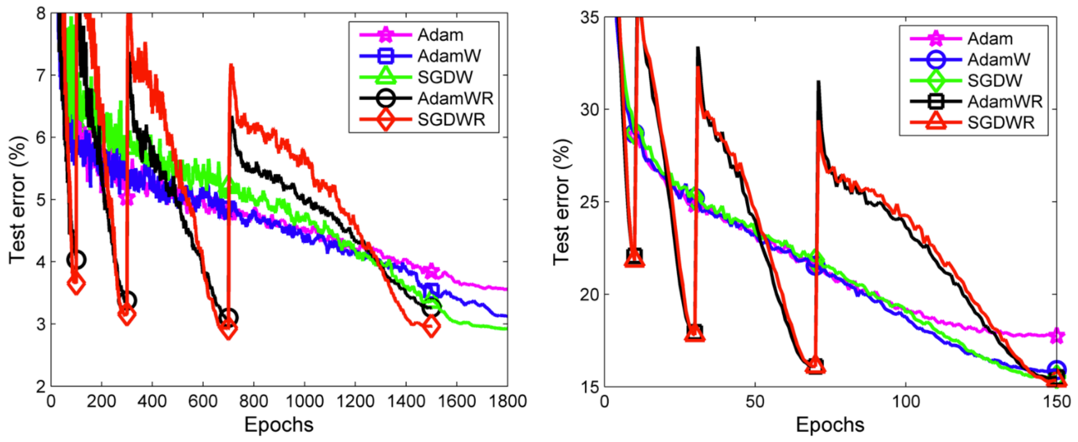


Figure 3: Top-1 test error on CIFAR-10 (left) and Top-5 test error on ImageNet32x32 (right).

总结

- 修改weight decay实现方式。使Adam能够在设置weight decay下受益。
- 解耦learning rate。使得超参数更容易调整。
- 热重启达到更快，更好的性能。