Adam AdamW

Adam??????

1. 初始化参数
   - 初始化一阶矩（动量）向量 $m0=0$ 和二阶矩（梯度平方的指数平均）向量 $v0=0$。
   - 设定超参数：学习率 η（默认 0.001）、动量衰减因子 $\beta1$（默认 0.9）、二阶矩衰减因子 $\beta2$（默认 0.999）、数值稳定常数 ϵ（默认 $1e-8$）
2. 计算当前梯度 在时间步 t，计算损失函数对参数 θ_{t} 的梯度 g_{t}
3. 更新一阶矩（动量项） 对梯度进行指数加权移动平均，模拟动量效果：
   $m_{t}=\beta1\cdot m_{t-1}+(1-\beta1)\cdot g_{t}$
4. 更新二阶矩（梯度平方的指数平均） 计算梯度平方的移动平均，用于自适应调整学习率：
   $v_{t}=\beta2\cdot v_{t-1}+(1-\beta2)\cdot g_{t}^{2}$
5. 偏差校正（**Bias Correction**）因初始时刻 mt 和 vt 偏向 0，需通过时间步 t 修正：
   $m_{t}=m_{t}/(1-\beta_{1}^{t})$ $v^{t}=v^{t}/(1-\beta_{2}t)$
6. 参数更新 结合修正后的矩估计调整学习率，更新参数

```
class AdamOptimizer:
    def __init__(self, learning_rate=0.01, beta1=0.9, beta2=0.999, epsilon=1e-8):
        self.lr = learning_rate
        self.beta1 = beta1
        self.beta2 = beta2
        self.epsilon = epsilon
        self.m = 0
        self.v = 0
        self.t = 0
    def update(self, params, grads):
        self.t += 1
        for param, grad in zip(params, grads):
            self.m[param] = self.beta1 * self.m[param] + (1 - self.beta1) * grad
            self.v[param] = self.beta2 * self.v[param] + (1 - self.beta2) * grad**2
            m_hat = self.m[param] / (1 - self.beta1**self.t)
            v_hat = self.v[param] / (1 - self.beta2**self.t)
            param = param - self.lr * m_hat / (np.sqrt(v_hat) + self.epsilon)
        return params
```

weight decay

权重衰减，本质上希望对权重进行惩罚，权重绝对值越大越不好

```
w = w - lr * w.grad - lr * weight_decay * w
```

L2正则化，表示对权重的衰减比例和权重绝对值的平方成正比

```
final_loss = loss + wd * all_weights.pow(2).sum() / 2
```

Adam在实现 L2正则化的时候是直接修改梯度，随后梯度会参与动量的计算，导致 L2 正则化的梯度就会被抵消掉，导致效果不佳

```
grad_w = grad_w + weight_decay * w
```

AdamW

AdamW 的实现方法与传统的 L2 正则化不同，它直接对权重进行衰减，而不是将其加到损失函数上（改变 **grad_w**）

```
grad_w = grad_w
# update step
w = w - learning_rate * grad_w - learning_rate * weight_decay * w
```

Revision #4
Created 2025-03-12 08:09:34 UTC by Colin
Updated 2026-04-29 07:34:10 UTC by Colin