



人工智能与深度学习

2-5 梯度下降法及其衍生算法

王中雷

经济学院、王亚南经济研究院、邹至庄经济研究院

厦门大学

(第一版)

内容摘要

1. 梯度下降法的变体

2. 衍生算法

3. 学习率衰减策略

回顾：批量梯度下降法

步骤1. 随机初始化 $\boldsymbol{\theta}^{(0)}$

步骤2. 基于当前模型参数 $\boldsymbol{\theta}^{(t)}$ ，计算

$$\nabla \mathcal{J}(\boldsymbol{\theta}^{(t)}) = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^{(t)}) = \textcolor{red}{n}^{-1} \sum_{i=1}^{\textcolor{red}{n}} \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^{(t)})$$

- $\mathcal{L}_i$ ：针对第 i 个训练样本的损失函数

步骤3. 参数更新

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \cdot \nabla \mathcal{J}(\boldsymbol{\theta}^{(t)})$$

步骤4. 回到步骤 2 直至收敛，或达到最大迭代次数

回顾：批量梯度下降法

1. 优点

- 我们用的是“真实梯度”

2. 缺点

- 计算效率低下，尤其当样本量 n 很大的时候
- 严重消耗内存

3. 为什么不能只用部分训练样本更新参数呢？

小批量梯度下降法

1. 基本思想：每次参数更新只使用一小批训练样本
2. 将训练样本的指标集划分成若干互斥的集合： $\{1, \dots, n\} = S_1 \cup \dots \cup S_k$
 - $S_i \cap S_j = \emptyset \quad (i \neq j)$
 - $|S_1| = \dots = |S_{k-1}| = m$
 - $|S_k| \leq m$
 - $k = \lceil n/m \rceil$
 - 为了使 GPU 有更好的计算性能， m 的取值通常是 2 的幂
 - ▷ 例如， $m = 32$

小批量梯度下降法

1. 对于第 t 步更新, 模型参数的更新规则为

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \cdot \nabla \mathcal{J}_t(\boldsymbol{\theta}^{(t)}),$$
$$\nabla \mathcal{J}_t(\boldsymbol{\theta}^{(t)}) = |S_{t \% k + 1}|^{-1} \sum_{i \in S_{t \% k + 1}} \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^{(t)})$$

- 只用 $S_{t \% k + 1}$ 中的样本点计算梯度

2. 两种特殊情况

- $m = 1$: 随机梯度下降法
- $m = n$: 批量梯度下降法

3. 当全部样本被遍历完, 我们称训练完成一个周期 (epoch)

4. 注: 在每个 epoch 开始前随机打乱训练样本, 然后重新划分小批量

关于随机梯度下降法的备注

1. 缺点:

- 仅仅基于一个样本点的梯度估计，是对真实梯度的一个包含很大噪声的估计

2. 优点:

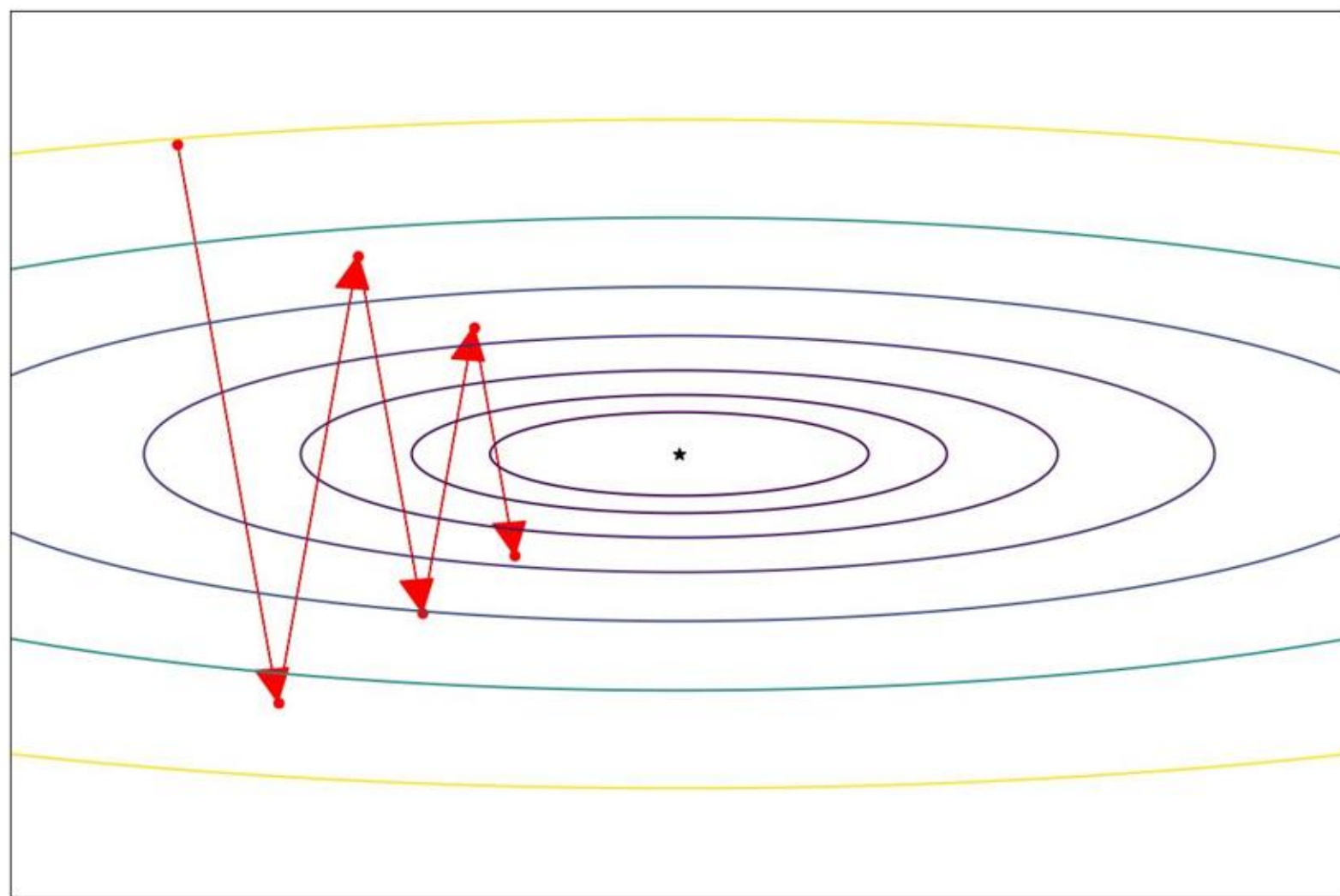
- 随机梯度中的噪声有时有助于算法离开鞍点或较差的局部区域，但并不保证一定能够逃离局部极小值
- 我们需要调整学习率，以达到该算法的收敛
- 单次更新的计算成本较低，适用于在线学习

讨论

1. 当 $m < n$ 时，小批量梯度是批量梯度的有噪声估计，其方差通常随 m 减小而增大
2. 然而，小批量梯度下降法，却能够节省内存，以及提高计算效率
3. Masters and Luschi (2018) 建议使用较小批量；实际批量大小应根据任务、显存、硬件利用率和泛化表现选择
4. 广泛用于深度学习模型的训练
5. 问题：为什么 m 不是越大越好

问题背景

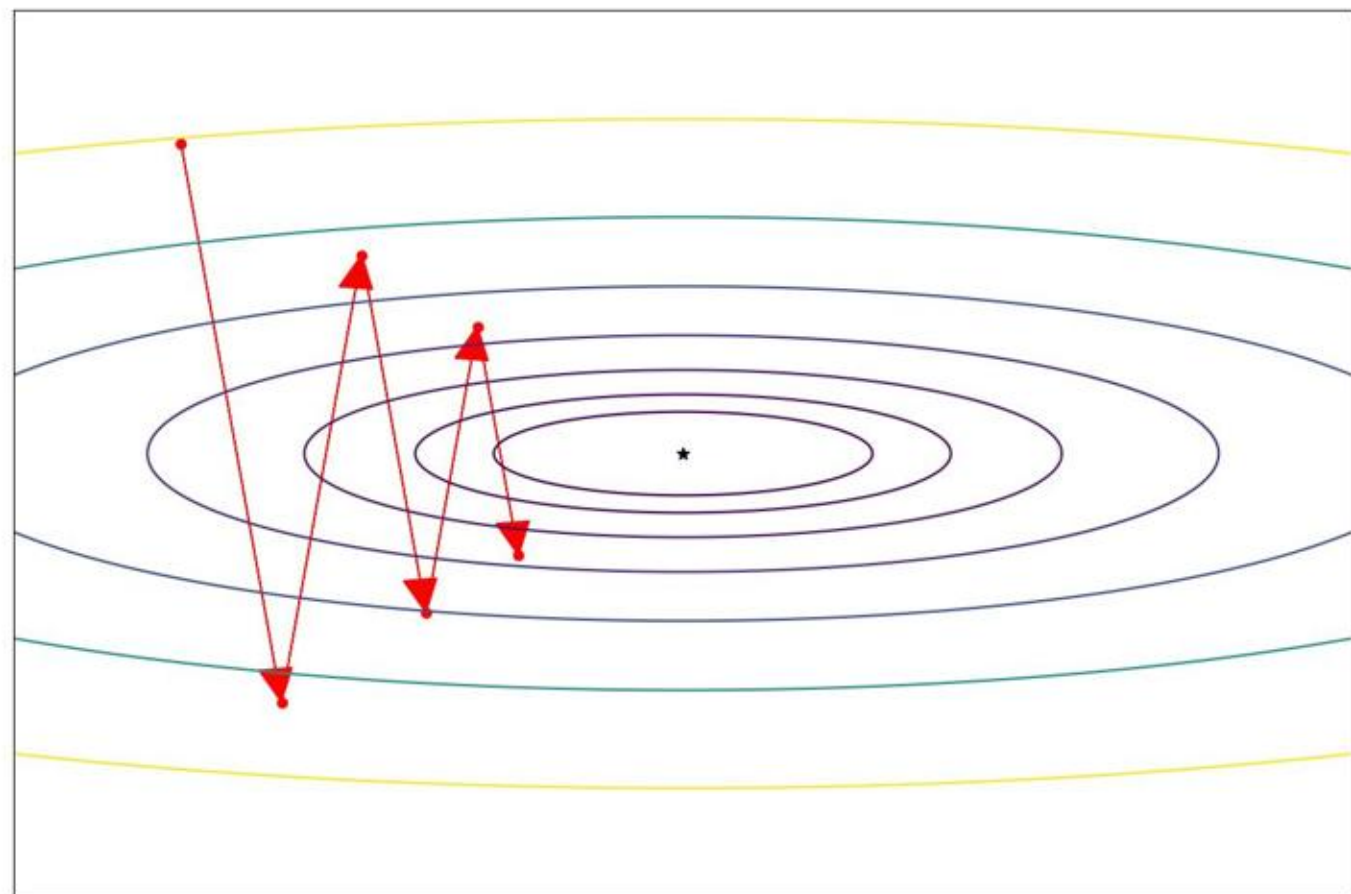
1. 梯度展示的是，代价函数值增加最快的方向
2. 当代价函数的梯度“不稳定”时，梯度下降法的效率将降低



问题背景

1. 我们希望用更加“稳定”的方向，替代负梯度
2. 为了达到这个目标，我们考虑下面几种算法
 - 动量法 (Momentum): 通过加权求和，以平滑参数更新方向
 - RMSprop: 自适应地调节每个模型参数的学习率
 - Adam: 结合动量法和 RMSprop 的优势

动量法 (Momentum)



1. 我们希望达到如下效果

- 减小上-下方向的效应
- 增大左-右方向的效应

动量法 (Momentum)

1. 考虑EWMA (Exponential Weighted Moving Average)

- 原始序列: $\{\mathbf{g}^{(t)} : t = 1, 2, \dots\}$
 - ▷ 可以是 $d\mathbf{b}^{[l](t)}$ 或者 $d\mathbf{W}^{[l](t)}$, 在当前模型参数 $\boldsymbol{\theta}^{(t)}$ 下的计算结果
- EWMA 序列: $\{\mathbf{v}^{(t)} : t = 1, 2, \dots\}$
$$\mathbf{v}^{(t)} = \beta_1 \cdot \mathbf{v}^{(t-1)} + (1 - \beta_1) \cdot \mathbf{g}^{(t)} \quad (t = 1, 2, \dots)$$
 - ▷ $\mathbf{v}^{(0)} = 0$
 - ▷ $\beta_1 \in (0, 1)$: 控制我们多么重视“动量” $\mathbf{v}^{(t-1)}$ 的值
 - ▷ $\beta_1 = 0.9$: 模型超参 (通常不调参)

备注

1. 一个事实：对于 $\beta_1 \in (0, 1)$ ，我们有

$$\frac{1}{(1 - \beta_1)^{-1}} \sum_{i=0}^{\infty} \beta_1^i = 1$$

2. 更多细节

$$\begin{aligned} \mathbf{v}^{(t)} &= \beta_1 \cdot \mathbf{v}^{(t-1)} + (1 - \beta_1) \cdot \mathbf{g}^{(t)} \\ &= (1 - \beta_1) \cdot \beta_1^{t-1} \cdot \mathbf{g}^{(1)} + (1 - \beta_1) \cdot \beta_1^{t-2} \cdot \mathbf{g}^{(2)} + \dots + (1 - \beta_1) \cdot \mathbf{g}^{(t)} \\ &= \frac{1}{(1 - \beta_1)^{-1}} \sum_{i=1}^t \beta_1^{t-i} \cdot \mathbf{g}^{(i)} \end{aligned}$$

- 当 t 较大时，以上结果可被视为求加权平均
- $(1 - \beta_1)^{-1}$ ：可被视作 EWMA 的“近似记忆长度”

基于动量的梯度下降法

步骤1. 随机初始化 $\theta^{(0)}$

步骤2. 基于当前模型参数 $\theta^{(t)}$, 计算 $d\mathbf{b}^{[1](t)}$

- 作为例子, 我们仅展示模型参数 $\mathbf{b}^{[1]}$ 的更新过程
- 相同的过程, 可用于更新其他模型参数

步骤3. 更新模型参数

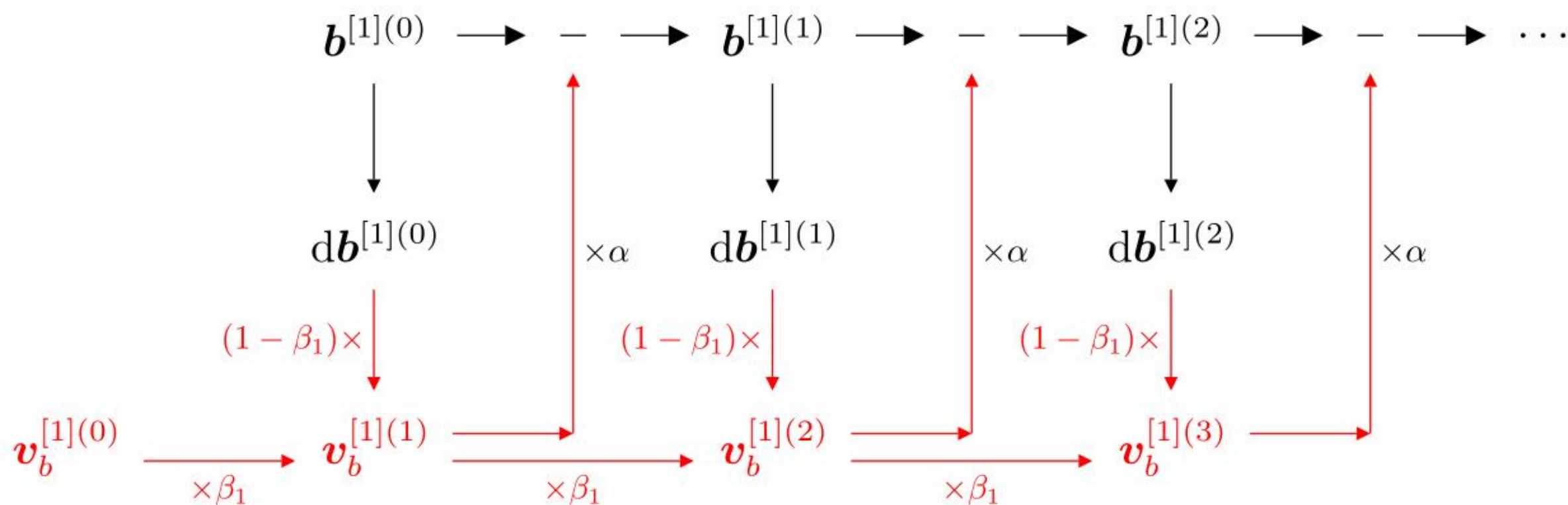
$$\mathbf{b}^{[1](t+1)} = \mathbf{b}^{[1](t)} - \alpha \cdot \mathbf{v}_b^{[1](t+1)}$$

- $\mathbf{v}_b^{[1](t+1)} = \beta_1 \cdot \mathbf{v}_b^{[1](t)} + (1 - \beta_1) \cdot d\mathbf{b}^{[1](t)}$
- $\mathbf{v}_b^{[1](0)} = 0$
- $\beta_1 = 0.9$ (默认值, 通常不调参)

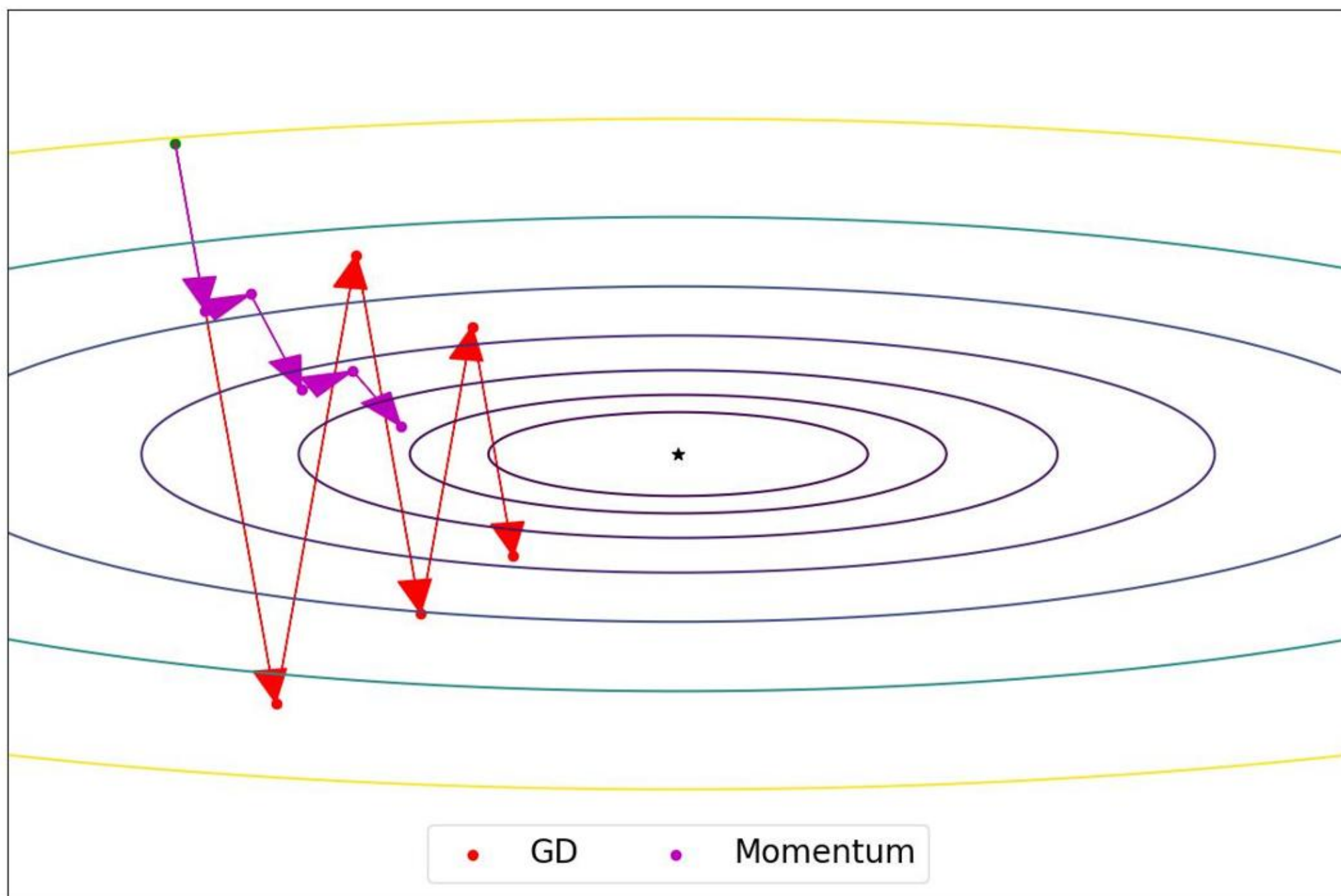
步骤4. 回到步骤 2 直至收敛, 或达到最大迭代次数

算法展示

1. 我们针对参数 $\mathbf{b}^{[1]}$ 展示动量法
2. 其他模型参数，以类似的方式，在同一步迭代中进行更新

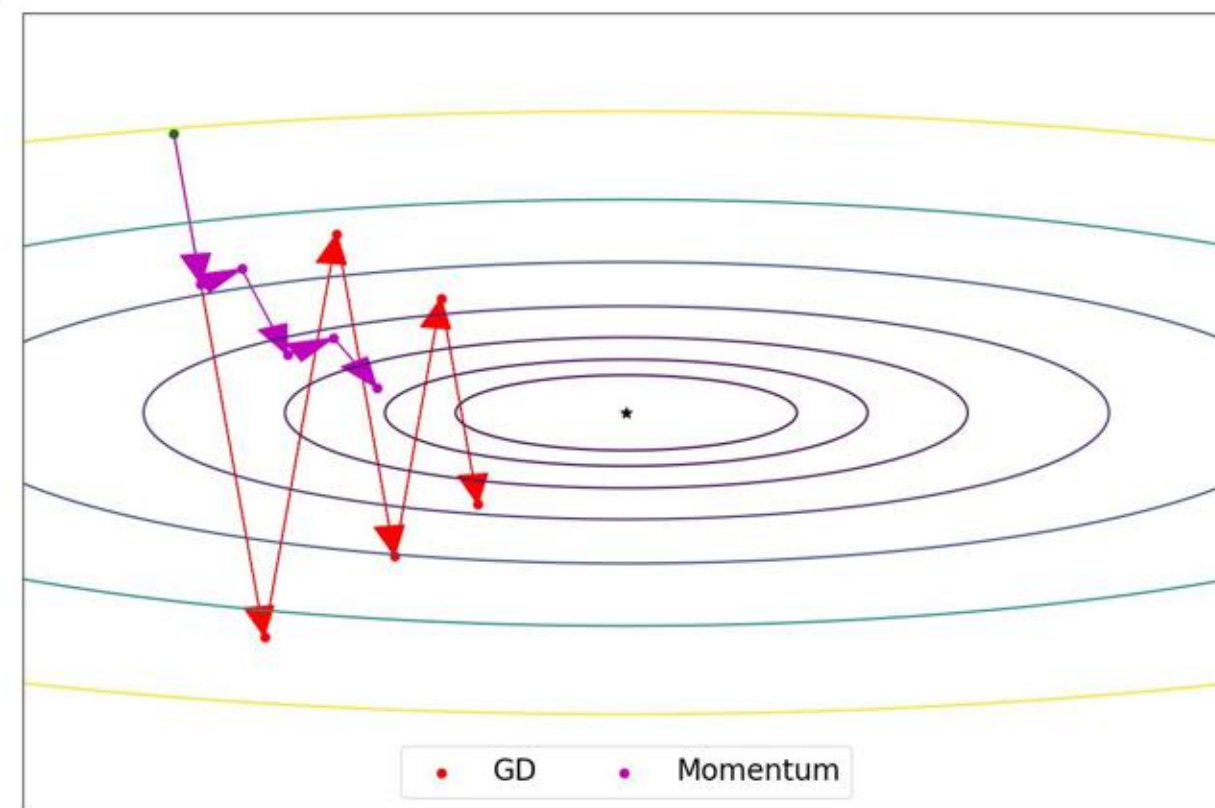


比较



比较

1. 动量法能够削弱梯度在振荡方向（上下方向）上的变化
2. 从而，相对突出长期保持一致的下降方向（左右方向）
3. 一般而言，基于相同的学习率，动量法会优于梯度下降法
4. 一种可能的解决方法：不同方向用不同的学习率
 - 较小的学习率用于上-下方向
 - 较大的学习率用于左-右方向



基于RMSprop的梯度下降法

步骤1. 随机初始化 $\theta^{(0)}$

步骤2. 基于当前模型参数 $\theta^{(t)}$, 计算 $d\mathbf{b}^{[1](t)}$

- 作为例子, 我们考虑模型参数 $\mathbf{b}^{[1]}$ 的更新过程

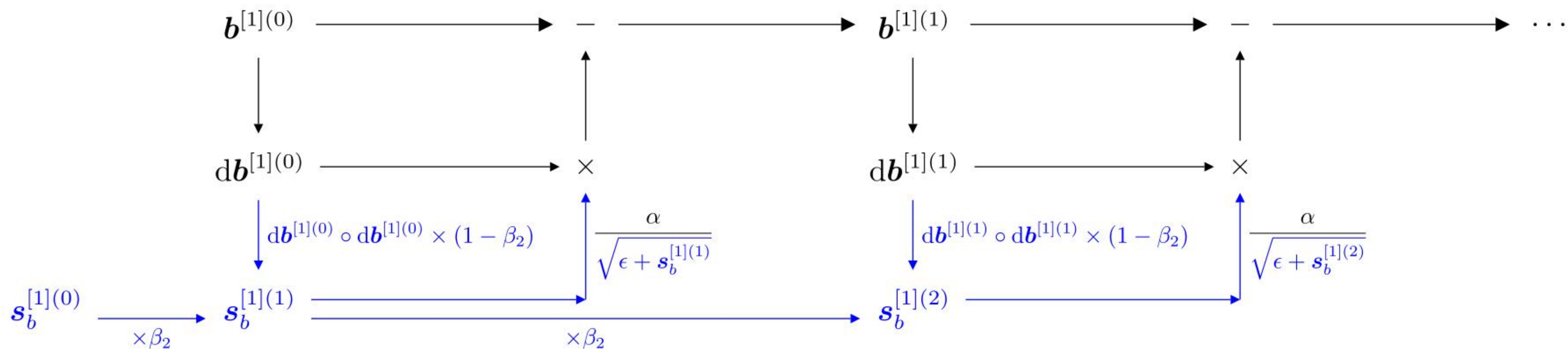
步骤3. 更新参数

$$\mathbf{b}^{[1](t+1)} = \mathbf{b}^{[1](t)} - \frac{\alpha}{\sqrt{\epsilon + \mathbf{s}_b^{[1](t+1)}}} \cdot d\mathbf{b}^{[1](t)}$$

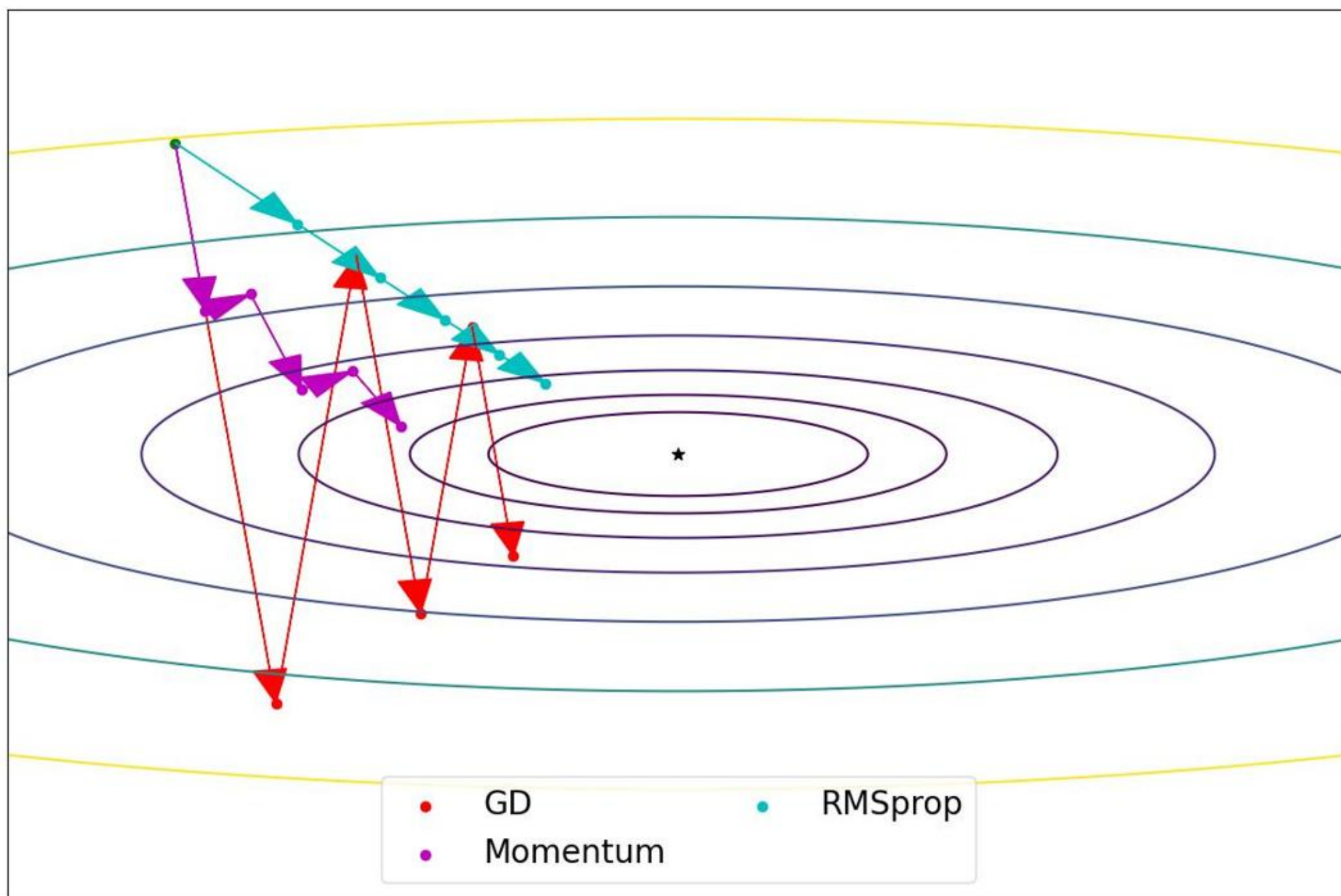
- $\epsilon = 10^{-8}$ (默认值, 通常不调参)
- $\mathbf{s}_b^{[1](t+1)} = \beta_2 \cdot \mathbf{s}_b^{[1](t)} + (1 - \beta_2) \cdot d\mathbf{b}^{[1](t)} \circ d\mathbf{b}^{[1](t)}$
- $\mathbf{s}_b^{[1](0)} = 0$
- $\beta_2 = 0.99$ (默认值, 通常不调参)

算法展示

1. 我们针对参数 $\mathbf{b}^{[1]}$ 展示 RMSprop 算法
2. 其他模型参数，以类似的方式，在同一步迭代中进行更新



比较



算法总结

1. 传统的梯度下降法

$$\mathbf{b}^{[1](t+1)} = \mathbf{b}^{[1](t)} - \alpha \cdot d\mathbf{b}^{[1](t)}$$

2. 动量法只更改梯度部分

$$\mathbf{b}^{[1](t+1)} = \mathbf{b}^{[1](t)} - \alpha \cdot \mathbf{v}_b^{[1](t+1)}$$

3. RMSprop 只更改学习率部分

$$\mathbf{b}^{[1](t+1)} = \mathbf{b}^{[1](t)} - \frac{\alpha}{\sqrt{\epsilon + \mathbf{s}_b^{[1](t+1)}}} \cdot d\mathbf{b}^{[1](t)}$$

4. 为什么不同结合两种方法，同时调整梯度和学习率？

Adam (Adaptive moment estimation)

步骤1. 随机初始化 $\theta^{(0)}$

步骤2. 基于当前模型参数 $\theta^{(t)}$, 计算 $d\mathbf{b}^{[1](t)}$

- 作为例子, 我们考虑模型参数 $\mathbf{b}^{[1]}$ 的更新过程

步骤3. 更新参数

$$\mathbf{b}^{[1](t+1)} = \mathbf{b}^{[1](t)} - \frac{\alpha}{\epsilon + \sqrt{\hat{\mathbf{s}}_b^{[1](t+1)}}} \cdot \hat{\mathbf{v}}_b^{[1](t+1)}$$

- 关于 $\hat{\mathbf{s}}_b^{[1](t+1)}$ 以及 $\hat{\mathbf{v}}_b^{[1](t+1)}$ 的解释请参见下一页
- $\epsilon = 10^{-8}$ (默认值, 通常不调参)

步骤4. 回到步骤 2 直至收敛, 或达到最大迭代次数

Adam (Adaptive moment estimation)

1. 关于 $\hat{\mathbf{s}}_b^{[1](t+1)}$ 和 $\hat{\mathbf{v}}_b^{[1](t+1)}$ 的计算细节

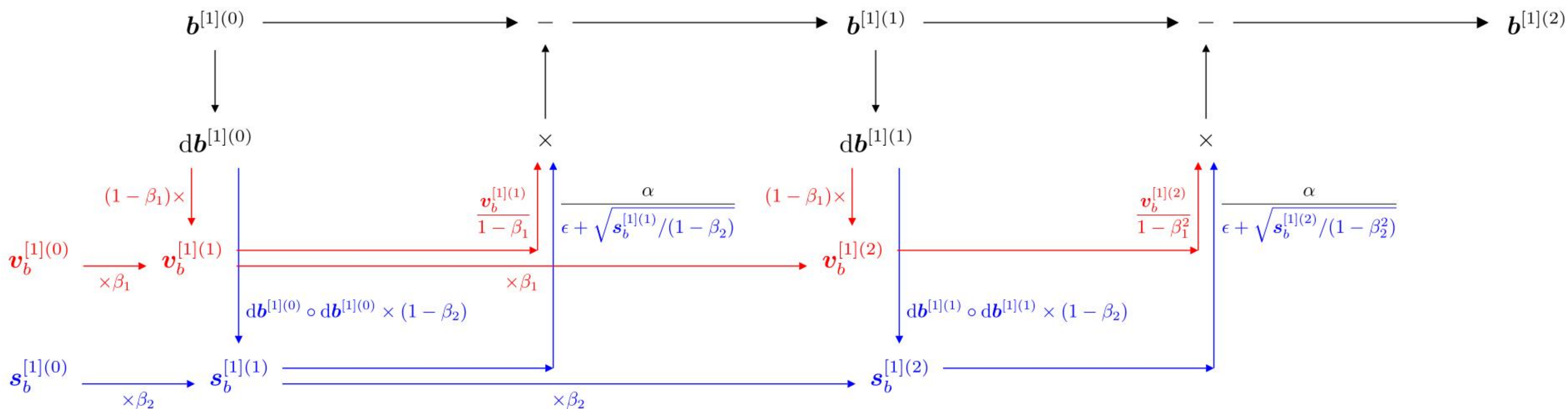
$$\mathbf{s}_b^{[1](t+1)} = \beta_2 \cdot \mathbf{s}_b^{[1](t)} + (1 - \beta_2) \cdot \mathrm{d}\mathbf{b}^{[1](t)} \circ \mathrm{d}\mathbf{b}^{[1](t)}, \quad \hat{\mathbf{s}}_b^{[1](t+1)} = \frac{\mathbf{s}_b^{[1](t+1)}}{1 - \beta_2^{t+1}},$$

$$\mathbf{v}_b^{[1](t+1)} = \beta_1 \cdot \mathbf{v}_b^{[1](t)} + (1 - \beta_1) \cdot \mathrm{d}\mathbf{b}^{[1](t)}, \quad \hat{\mathbf{v}}_b^{[1](t+1)} = \frac{\mathbf{v}_b^{[1](t+1)}}{1 - \beta_1^{t+1}}$$

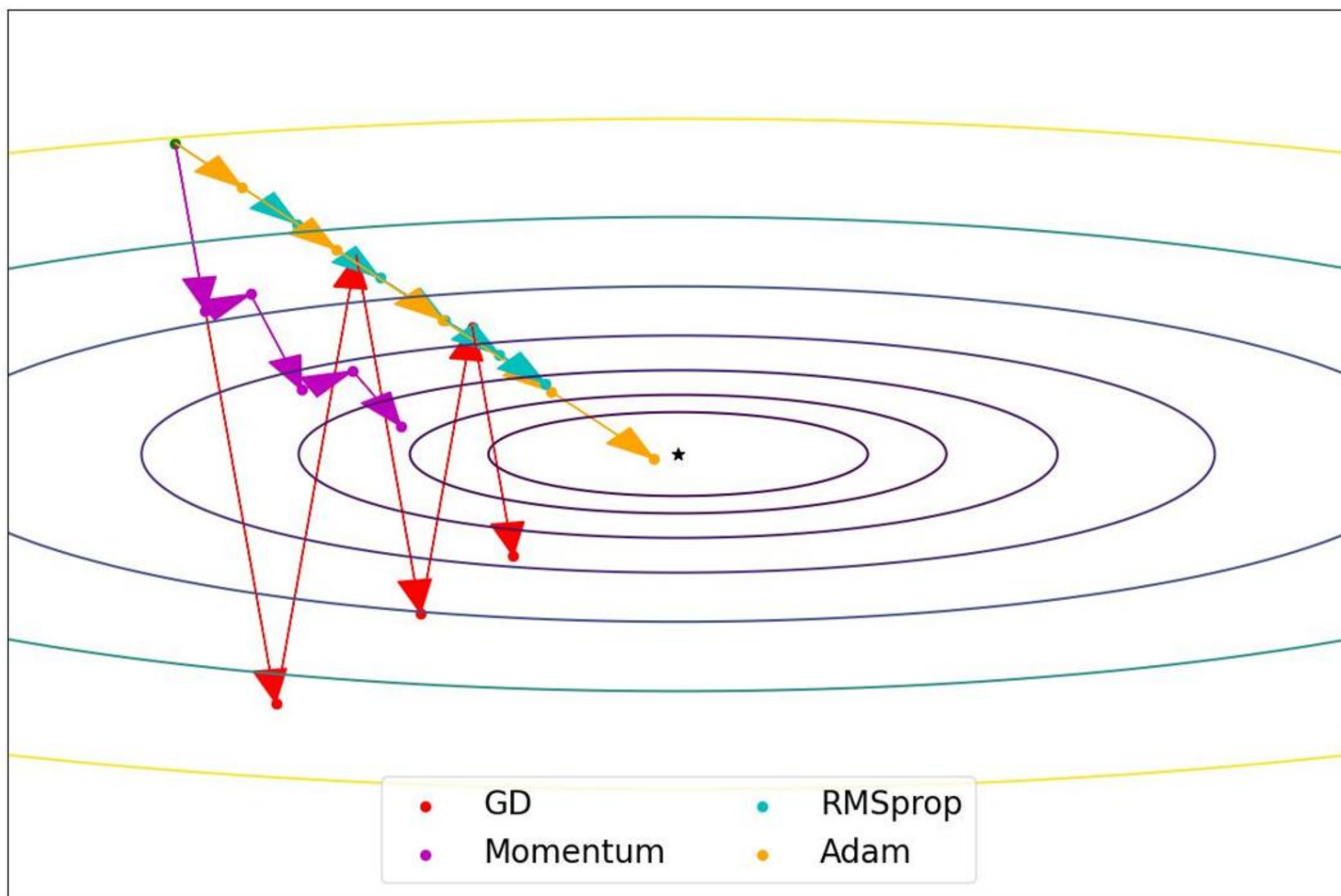
- $\mathbf{v}_b^{[1](0)} = \mathbf{s}_b^{[1](0)} = 0$
- $\beta_1 = 0.9$ （默认值，通常不调参）
- $\beta_2 = 0.999$ （默认值，通常不调参）

算法展示

1. 我们针对参数 $\mathbf{b}^{[1]}$ 展示 Adam 算法
2. 其他模型参数，以类似的方式，在同一步迭代中进行更新



算法对比



算法对比

1. Adam (Kingma and Ba, 2014) 或者其变体，通常用于深度模型参数估计

- Adam 在默认参数下，就能表现得很好
- 收敛速度较快
- 自动处理不同尺度的特征

2. Adam 的变体 AdamW 被 Loshchilov 和 Hutter (2019) 提出

- 引入权重衰减策略
- 很多深度神经网络训练的默认优化器
- 将在正则化一节中详细讨论

3. Adam 在一些情况下，也会失效 (Zaheer et al., 2018; Reddi et al., 2019)

学习率衰减策略

1. 直观

- 在训练初期，我们需要一个比较大的学习率
- 随着迭代的增加，估计值参数通常逐渐接近较优解或稳定区域
- 我们不应当在每步更新中，使用不变的学习率
 - ▷ [效率低] 较小的学习率通常具有较好的收敛性质，但我们通常需要多步迭代
 - ▷ [不稳定] 当迭代次数较大时，较大的学习率往往导致参数更新不太稳定
- 随着迭代次数的增加，我们应当以较为有效的方式降低学习率

学习率衰减策略

1. 记

- t : 迭代指标
- $epoch$: 周期指标

2. 有多种学习率衰减策略 ($t \geq 1, epoch \geq 1$)

$$\alpha_t = \frac{\alpha_0}{1 + \gamma \cdot epoch}, \quad \alpha_t = \frac{\alpha_0}{\sqrt{epoch}},$$

$$\alpha_t = 0.95^{epoch} \cdot \alpha_0, \quad \alpha_t = \frac{1}{\sqrt{t}} \cdot \alpha_0,$$

$$\alpha_t = 0.95^t \cdot \alpha_0$$

- 例如, $\alpha_0 = 5 \times 10^{-3}$ 以及 $\gamma = 1$

课程总结

1. 模型训练中的三个方面

- 每次更新用多少样本
- 更新向量是否需要平滑或缩放
- 学习率是否需要分阶段调整

2. 优化器的选择从 Adam 或其变体 AdamW 开始