

8. 深度学习中的优化

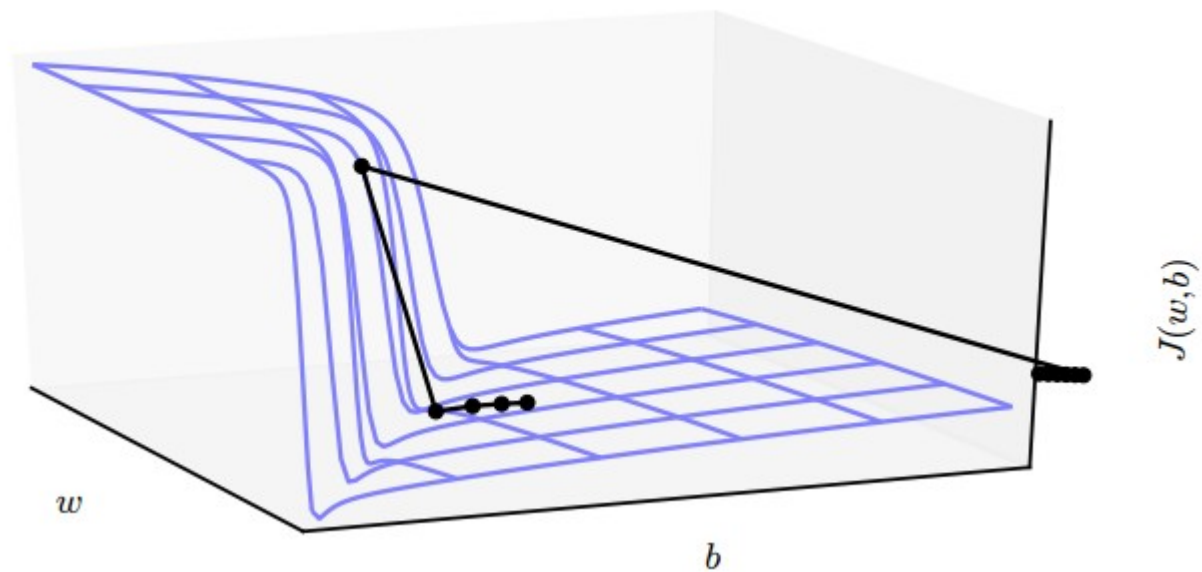
局部极小值

- 如果一个足够大的训练集可以唯一确定一组模型参数，那么该模型被称为可辨认的；而神经网络模型是不可辨认的，神经网络代价函数具有非常多的局部极小值。对于足够大的神经网络而言，大部分局部极小值都具有很小的代价函数，我们能不能找到真正的全局最小点并不重要，而是需要在参数空间中找到一个代价很小（但不是最小）的点。

鞍点

- 对于很多高维非凸函数而言，局部极小值事实上远少于另一类梯度为零的点
- 实验中梯度下降似乎可以在许多情况下逃离鞍点。
- 梯度下降旨在朝“下坡”移动，而非明确寻求临界点。而牛顿法的目标是寻求梯度为零的点。

悬崖和梯度爆炸



随机梯度下降

算法 8.1 随机梯度下降 (SGD) 在第 k 个训练迭代的更新

Require: 学习率 ϵ_k

Require: 初始参数 θ

while 停止准则未满足 do

从训练集中采包含 m 个样本 $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ 的小批量, 其中 $\mathbf{x}^{(i)}$ 对应目标为 $\mathbf{y}^{(i)}$ 。

计算梯度估计: $\hat{\mathbf{g}} \leftarrow +\frac{1}{m} \nabla_{\theta} \sum_i L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

应用更新: $\theta \leftarrow \theta - \epsilon \hat{\mathbf{g}}$

end while

动量

$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \epsilon \nabla_{\theta} \left(\frac{1}{m} \sum_{i=1}^m L(\mathbf{f}(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)}) \right),$$

$$\theta \leftarrow \theta + \mathbf{v}.$$

算法 8.2 使用动量的随机梯度下降 (SGD)

Require: 学习率 ϵ , 动量参数 α

Require: 初始参数 θ , 初始速度 \mathbf{v}

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ 的小批量, 对应目标为 $\mathbf{y}^{(i)}$ 。

计算梯度估计: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(\mathbf{f}(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

计算速度更新: $\mathbf{v} \leftarrow \alpha \mathbf{v} - \epsilon \mathbf{g}$

应用更新: $\theta \leftarrow \theta + \mathbf{v}$

end while

Nesterov 动量

$$\begin{aligned}v &\leftarrow \alpha v - \epsilon \nabla_{\theta} \left[\frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}^{(i)}; \theta + \alpha v), \mathbf{y}^{(i)}) \right], \\ \theta &\leftarrow \theta + v,\end{aligned}$$

算法 8.3 使用 Nesterov 动量的随机梯度下降 (SGD)

Require: 学习率 ϵ , 动量参数 α

Require: 初始参数 θ , 初始速度 v

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ 的小批量, 对应目标为 $\mathbf{y}^{(i)}$ 。

应用临时更新: $\tilde{\theta} \leftarrow \theta + \alpha v$

计算梯度 (在临时点): $g \leftarrow \frac{1}{m} \nabla_{\tilde{\theta}} \sum_i L(f(\mathbf{x}^{(i)}; \tilde{\theta}), \mathbf{y}^{(i)})$

计算速度更新: $v \leftarrow \alpha v - \epsilon g$

应用更新: $\theta \leftarrow \theta + v$

end while

自适应学习率算法

- AdaGrad

算法 8.4 AdaGrad 算法

Require: 全局学习率 ϵ

Require: 初始参数 θ

Require: 小常数 δ , 为了数值稳定大约设为 10^{-7}

初始化梯度累积变量 $\mathbf{r} = 0$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ 的小批量, 对应目标为 $\mathbf{y}^{(i)}$ 。

计算梯度: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

累积平方梯度: $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{g} \odot \mathbf{g}$

计算更新: $\Delta \theta \leftarrow -\frac{\epsilon}{\delta + \sqrt{\mathbf{r}}} \odot \mathbf{g}$ (逐元素地应用除和求平方根)

应用更新: $\theta \leftarrow \theta + \Delta \theta$

end while

RMSProp

算法 8.5 RMSProp 算法

Require: 全局学习率 ϵ , 衰减速率 ρ

Require: 初始参数 θ

Require: 小常数 δ , 通常设为 10^{-6} (用于被小数除时的数值稳定)

初始化累积变量 $\mathbf{r} = 0$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ 的小批量, 对应目标为 $\mathbf{y}^{(i)}$ 。

计算梯度: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

累积平方梯度: $\mathbf{r} \leftarrow \rho \mathbf{r} + (1 - \rho) \mathbf{g} \odot \mathbf{g}$

计算参数更新: $\Delta \theta = -\frac{\epsilon}{\sqrt{\delta + \mathbf{r}}} \odot \mathbf{g}$ ($\frac{1}{\sqrt{\delta + \mathbf{r}}}$ 逐元素应用)

应用更新: $\theta \leftarrow \theta + \Delta \theta$

end while

算法 8.6 使用 Nesterov 动量的 RMSProp 算法

Require: 全局学习率 ϵ , 衰减速率 ρ , 动量系数 α

Require: 初始参数 θ , 初始参数 v

初始化累积变量 $r = 0$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ 的小批量, 对应目标为 $\mathbf{y}^{(i)}$ 。

计算临时更新: $\tilde{\theta} \leftarrow \theta + \alpha v$

计算梯度: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\theta}} \sum_i L(f(\mathbf{x}^{(i)}; \tilde{\theta}), \mathbf{y}^{(i)})$

累积梯度: $\mathbf{r} \leftarrow \rho \mathbf{r} + (1 - \rho) \mathbf{g} \odot \mathbf{g}$

计算速度更新: $\mathbf{v} \leftarrow \alpha \mathbf{v} - \frac{\epsilon}{\sqrt{\mathbf{r}}} \odot \mathbf{g}$ ($\frac{1}{\sqrt{r}}$ 逐元素应用)

应用更新: $\theta \leftarrow \theta + \mathbf{v}$

end while

目前完成的改进：

梯度：

1) 随机梯度下降：SGD

2) 计算动量：Nesterov 动量

学习率

1) 为每个参数计算出不同学习率：adagrad

2) 防止学习率衰减或梯度消失等问题的出现：RMSProp

3) 也计算了动量项：Nesterov 动量 + RMSProp

还可以做什么改进？

- 在之前的方法中计算了每个参数的对应学习率，但是为什么不计算每个参数的对应动量变化并独立存储呢？这就是 Adam 算法提出的改良点。

Adam

算法 8.7 Adam 算法

Require: 步长 ϵ (建议默认为: 0.001)

Require: 矩估计的指数衰减速率, ρ_1 和 ρ_2 在区间 $[0, 1)$ 内。(建议默认为: 分别为 0.9 和 0.999)

Require: 用于数值稳定的小常数 δ (建议默认为: 10^{-8})

Require: 初始参数 θ

初始化一阶和二阶矩变量 $\mathbf{s} = 0, \mathbf{r} = 0$

初始化时间步 $t = 0$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ 的小批量, 对应目标为 $\mathbf{y}^{(i)}$ 。

计算梯度: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

$t \leftarrow t + 1$

更新有偏一阶矩估计: $\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1) \mathbf{g}$

更新有偏二阶矩估计: $\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1 - \rho_2) \mathbf{g} \odot \mathbf{g}$

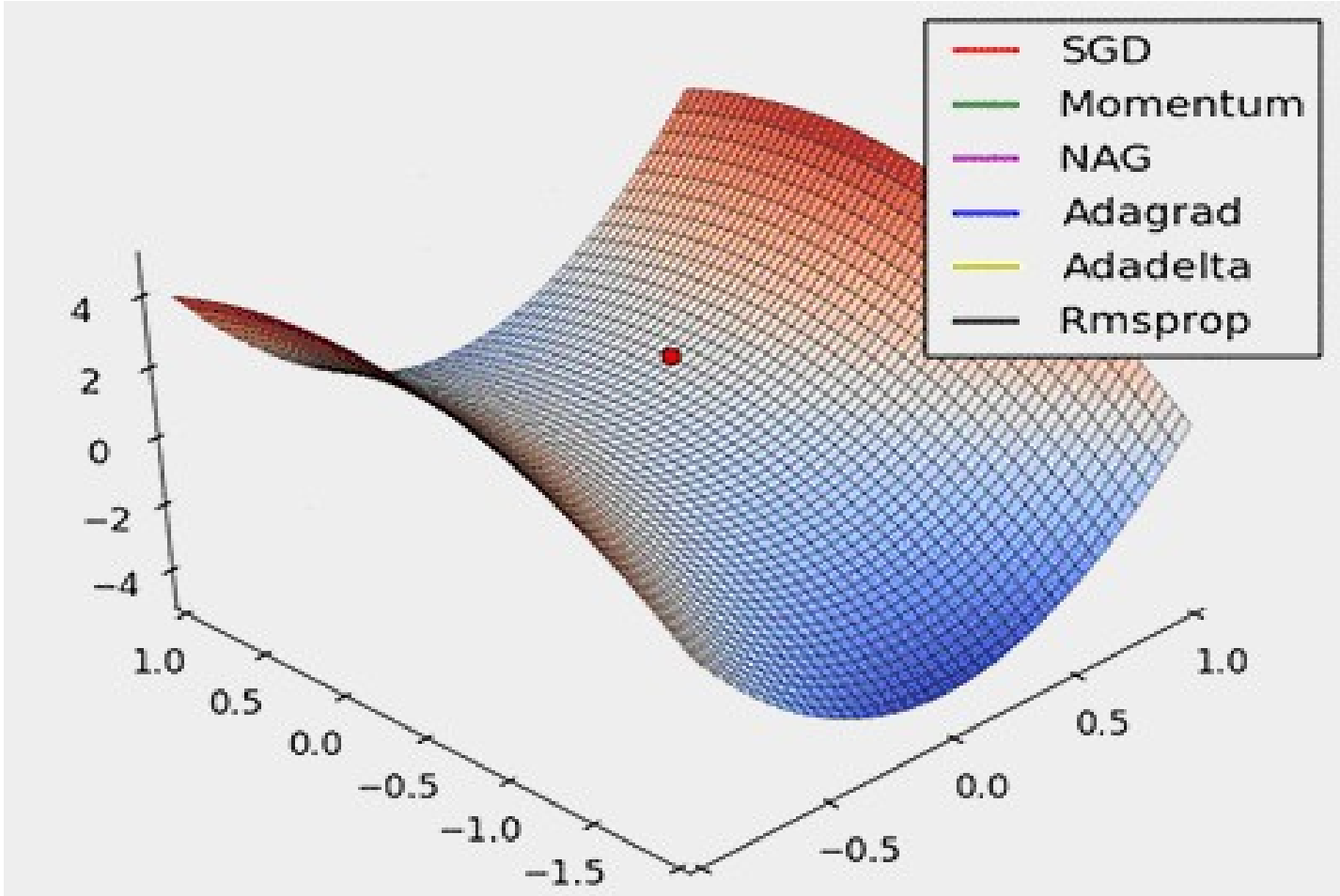
修正一阶矩的偏差: $\hat{\mathbf{s}} \leftarrow \frac{\mathbf{s}}{1 - \rho_1^t}$

修正二阶矩的偏差: $\hat{\mathbf{r}} \leftarrow \frac{\mathbf{r}}{1 - \rho_2^t}$

计算更新: $\Delta \theta = -\epsilon \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}} + \delta}}$ (逐元素应用操作)

应用更新: $\theta \leftarrow \theta + \Delta \theta$

end while



牛顿法

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}, y \sim \hat{p}_{\text{data}}(\mathbf{x}, y)} [L(f(\mathbf{x}; \boldsymbol{\theta}), y)] = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)}).$$

$$J(\boldsymbol{\theta}) \approx J(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \mathbf{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_0).$$

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0).$$

算法 8.8 目标为 $J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$ 的牛顿法

Require: 初始参数 $\boldsymbol{\theta}_0$

Require: 包含 m 个样本的训练集

while 没有达到停止准则 **do**

 计算梯度: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$

 计算 Hessian 矩阵: $\mathbf{H} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}}^2 \sum_i L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$

 计算 Hessian 逆: \mathbf{H}^{-1}

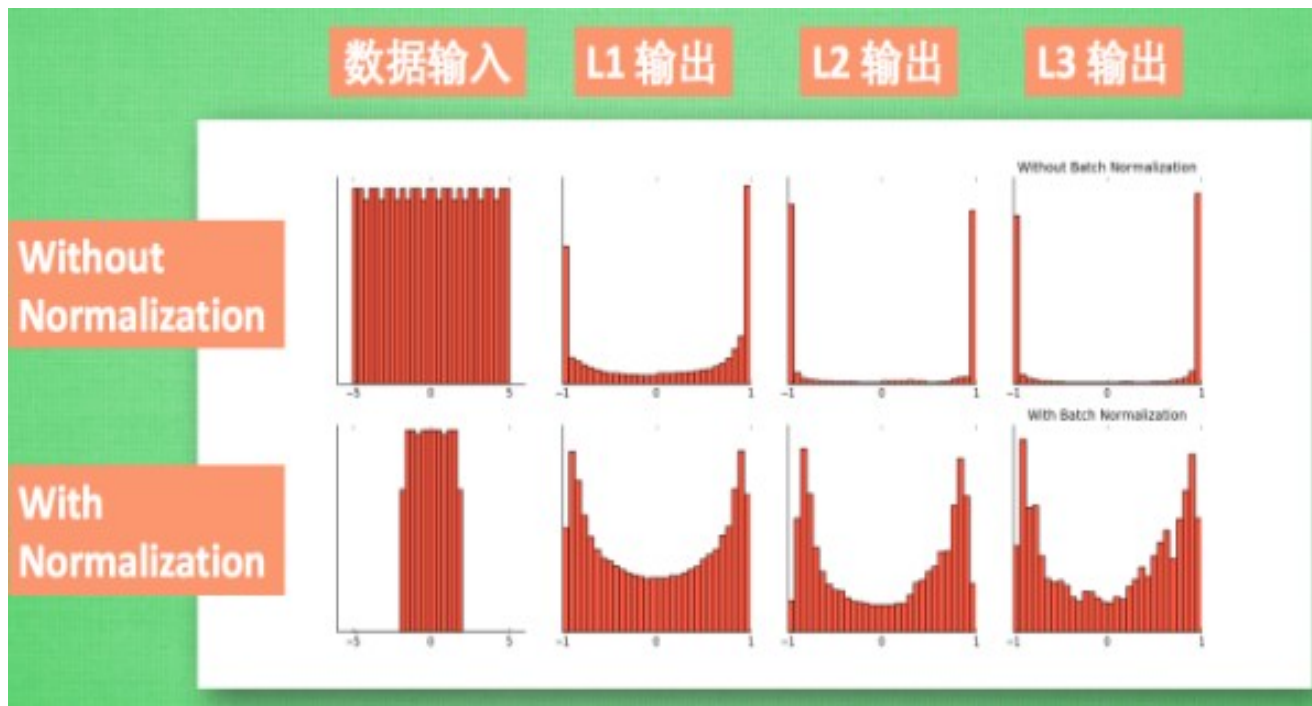
 计算更新: $\Delta \boldsymbol{\theta} = -\mathbf{H}^{-1} \mathbf{g}$

 应用更新: $\boldsymbol{\theta} = \boldsymbol{\theta} + \Delta \boldsymbol{\theta}$

end while

优化策略和元方法

- 批标准化



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \quad // \text{ scale and shift}$$

坐标下降法

- 不是求整个变量的梯度，然后下降，而是循环的沿着变量的每一维进行单变量优化。
- 当一个变量的值很大程度地影响另一个变量的最优值时，坐标下降不是一个很好的方法。

Polyak 平均

- 当应用 Polyak 平均于非凸问题时，通常会使用指数衰减计算平均值：

$$\theta(t) = \alpha\theta(t-1) + (1-\alpha)\theta(t)$$

监督预训练

如果模型太复杂难以优化，或是如果任务非常困难，直接训练模型来解决特定任务的挑战可能太大。因此可以训练一个较简单的模型来求解问题，然后使模型更复杂会更有效。训练模型来求解一个简化的问题，然后转移到最后的问题，有时也会更有效些。

•设计有助于优化的模型

改进优化的最好方法并不总是改进优化算法。
相反，深度模型中优化的许多改进来自于设计易于优化的模型。