An overview of gradient descent optimization algorithms

高仕源

2018 年 5 月 18 日，北京

中国科学院
计算机网络信息中心
Computer Network Information Center,
Chinese Academy of Sciences

目录

简介

- 虽然梯度下降优化算法越来越受欢迎，但通常作为黑盒优化器使用，因此很难对其优点和缺点的进行实际的解释。本文旨在让读者对不同的算法有直观的认识，以帮助读者使用这些算法。在本综述中，我们介绍梯度下降的不同变形形式，总结这些算法面临的挑战，介绍最常用的优化算法，回顾并行和分布式架构，以及调研用于优化梯度下降的其他的策略。
- 梯度下降算法是通过沿着目标函数 $J(\theta)$ 参数 $\theta \in \Re$ 的梯度 (一阶导数) 相反方向 $-\nabla \theta J(\theta)$ 来不断更新模型参数来到达目标函数的极小值点（收敛），更新步长为 η。

梯度下降法的变形形式

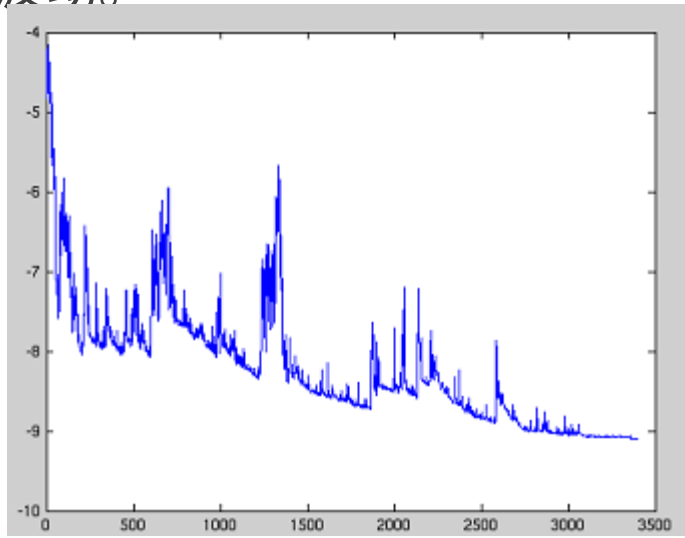
- 1. Vanilla 梯度下降法，又称为批梯度下降法 (batch gradient descent)
  - $\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$
- 因为在执行每次更新时，我们需要在整个数据集上计算所有的梯度，所以批梯度下降法的速度会很慢，同时，批梯度下降法无法处理超出内存容量限制的数据集。批梯度下降法同样也不能在线更新模型，即在运行的过程中，不能增加新的样本。

梯度下降法的变形形式

- 2. 随机梯度下降法（ stochastic gradient descent, SGD)

  - $\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x_i; y_i)$

- 对于大数据集，因为批梯度下降法在每一个参数更新之前，会对相似的样本计算梯度，所以在计算过程中会有冗余。而 SGD 在每一次更新中只执行一次，从而消除了冗余。因而，通常 SGD 的运行速度更快，同时，可以用于在线学习。 SGD 以高方差频繁地更新，导致目标函数出现如图所示的剧烈波动。



梯度下降法的变形形式

- 3. 小批量梯度下降 (Mini-batch gradient descent)
  - 小批量梯度下降法最终结合了上述两种方法的优点，在每次更新时使用个小批量训练样本

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x(i:i+n); y(i:i+n))$$

- 相对于随机梯度下降， Mini-batch 梯度下降降低了收敛波动性，即降低了参数更新的方差，使得更新更加稳定。相对于全量梯度下降，其提高了每次学习的速度。并且其不用担心内存瓶颈从而可以利用矩阵运算进行高效计算。一般而言每次更新随机选择 [50,256] 个样本进行学习，但是也要根据具体问题而选择，实践中可以进行多次试验，选择一个更新速度与更次次数都较适合的样本数。

挑战

- 选择一个合理的学习速率很难。如果学习速率过小，则会导致收敛速度很慢。如果学习速率过大，那么其会阻碍收敛，即在极值点附近会振荡。
- 学习速率调整 (又称学习速率调度， Learning rate schedules) 试图在每次更新过程中，改变学习速率，如退火。一般使用某种事先设定的策略或者在每次迭代中衰减一个较小的阈值。无论哪种调整方法，都需要事先进行固定设置，这边便无法自适应每次学习的数据集特点。
- 模型所有的参数每次更新都是使用相同的学习速率。如果数据特征是稀疏的或者每个特征有着不同的取值统计特征与空间，那么便不能在每次更新中每个参数使用相同的学习速率，那些很少出现的特征应该使用一个相对较大的学习速率。
- 对于非凸目标函数，容易陷入那些次优的局部极值点中，如在神经网路中。Dauphin 指出更严重的问题不是局部极值点，而是鞍点

梯度下降优化算法

- **Momentum**
- 如果在峡谷地区（某些方向较另一些方向上陡峭得多，常见于局部极值点），SGD 会在这些地方附近振荡

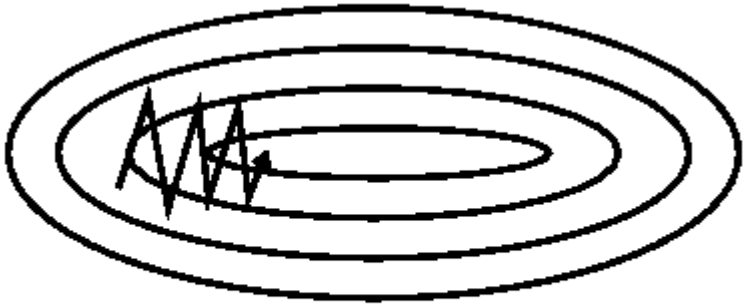


图2 没有动量

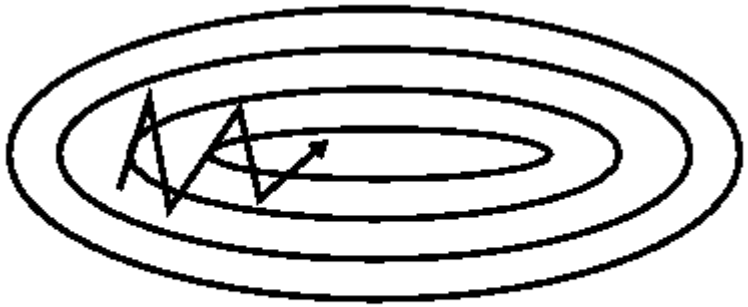


图3 加上动量

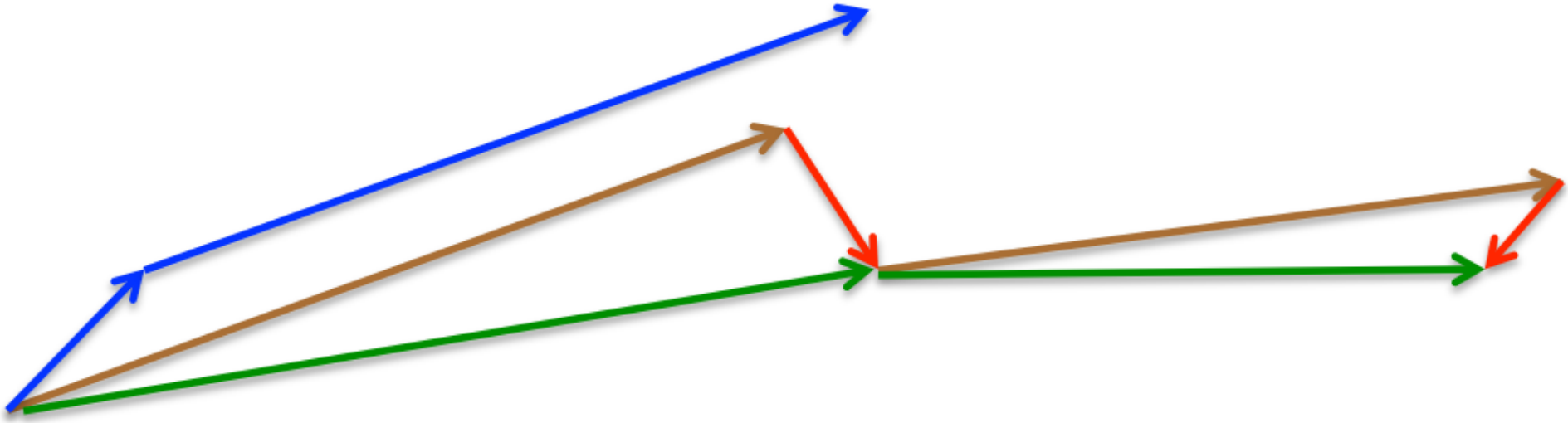
- 其中动量项超参数 $\gamma<1$ 一般是小于等于 0.9 。

梯度下降优化算法

- **Nesterov accelerated gradient(NAG, 涅斯捷罗夫梯度加速)**
  - 不仅增加了动量项，并且在计算参数的梯度时，在损失函数中减去了动量项

$$v_t = \gamma v_{t-1} + \eta \cdot \nabla_\theta J(\theta - \gamma v_{t-1})$$

$$\theta = \theta - v_t$$



梯度下降优化算法

- **Adagrad**
- 能够对每个参数自适应不同的学习速率，对稀疏特征，得到大的学习更新，对非稀疏特征，得到较小的学习更新，因此该优化算法适合处理稀疏特征数据。
- Adagrad 在每一个更新步骤中对于每一个模型参数 θi 使用不同的学习速率 ηi ，设第 t 次更新步骤中，目标函数的参数 θi 梯度为 gt,i ，即：

$$g_{t,i} = \nabla_\theta J(\theta_i)$$

那么SGD更新方程为：

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}$$

而Adagrad对每一个参数使用不同的学习速率，其更新方程为：

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

梯度下降优化算法

- **Adagrad （续）**

进一步，将所有$G_{t,ii}, g_{t,i}$ 的元素写成向量G_t, g_t ，这样便可以使用向量点乘操作：

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

Adadelta

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

梯度下降优化算法

- **RMSprop**
- 是 Adadelta 的中间形式，也是为了降低 Adagrad 中学习速率衰减过快问题，即：

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \odot g_t$$

- Hinton 建议 γ=0.9,η=0.001

梯度下降优化算法

- **Adaptive Moment Estimation(Adam)**
- 是一种不同参数自适应不同学习速率方法，与 Adadelta 与 RMSprop 区别在于，它计算历史梯度衰减方式不同，不使用历史平方衰减，其衰减方式类似动量

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - beta_2) g_t^2$$

梯度下降优化算法

- Adaptive Moment Estimation(Adam) 续

m_t 与 v_t 分别是梯度的带权平均和带权有偏方差，初始为0向量，Adam的作者发现他们倾向于0向量(接近于0向量)，特别是在衰减因子(衰减率)β_1 ,β_2 接近于1时。为了改进这个问题，对m_t 与 v_t 进行偏差修正(bias-corrected)：

$$\hat{m}_t = \frac{m_t}{1 - beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - beta_2^t}$$

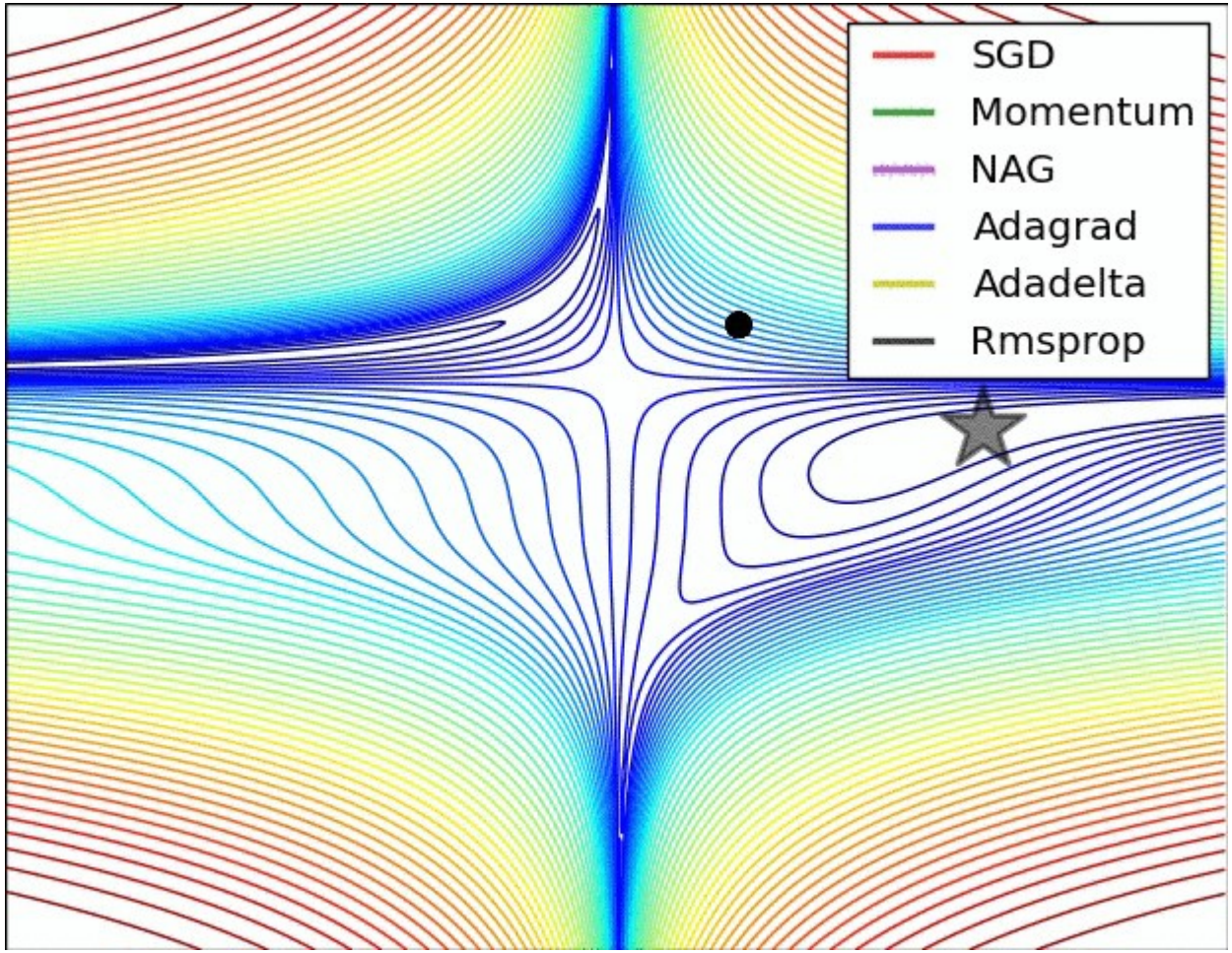
最终，Adam的更新方程为：

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

论文中建议默认值：$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\epsilon = 10^{-8}$。论文中将Adam与其它的几个自适应学习速率进行了比较，效果均要好。

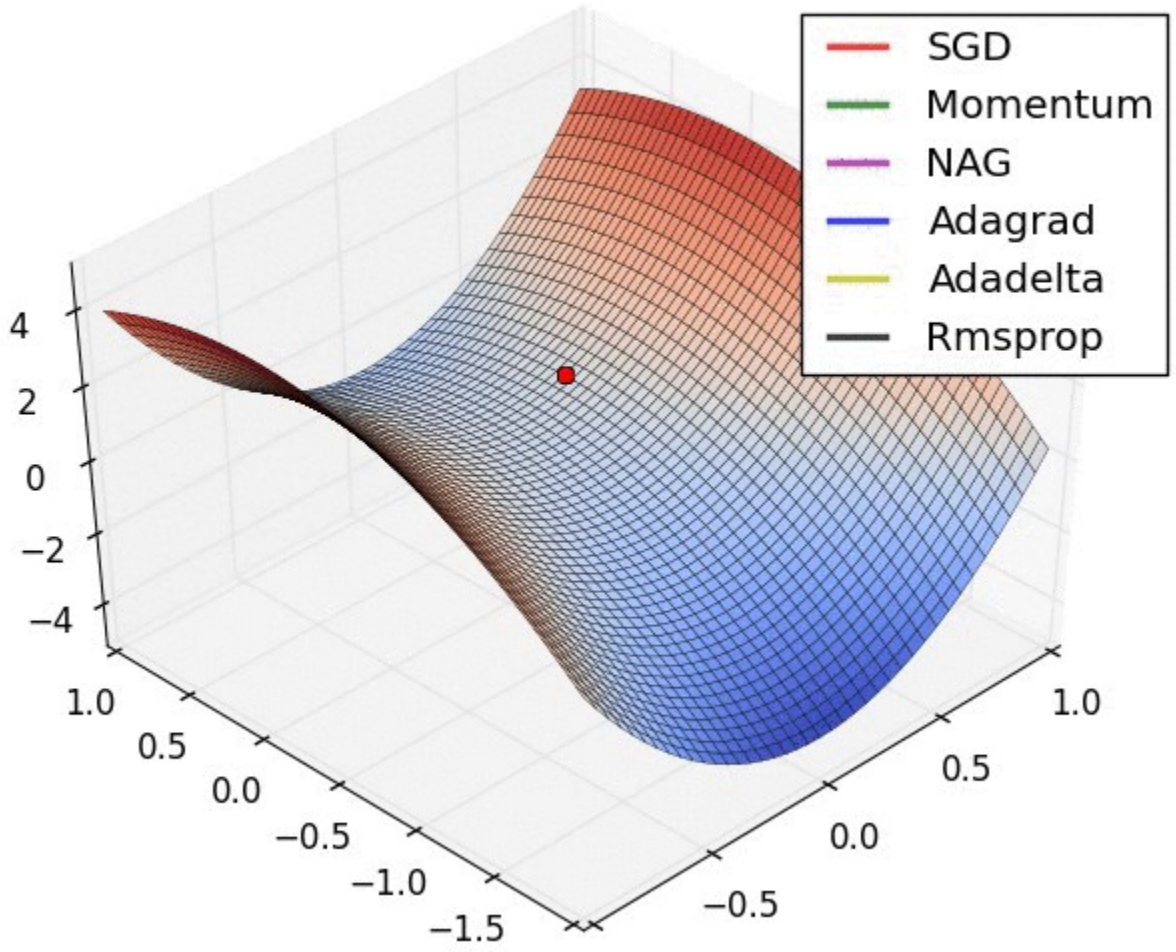
各种优化方法比较

- SGD 各优化方法在损失曲面上的表现



各种优化方法比较

- SGD 各优化方法在损失曲面鞍点处上的表现



如何选择 SGD 优化器

- 如果你的数据特征是稀疏的，那么你最好使用自适应学习速率 SGD 优化方法 (Adagrad 、 Adadelta 、 RMSprop 与 Adam) ，因为你不需要在迭代过程中对学习速率进行人工调整。
- RMSprop 是 Adagrad 的一种扩展，与 Adadelta 类似，但是改进版的 Adadelta 使用 RMS 去自动更新学习速率，并且不需要设置初始学习速率。而 Adam 是在 RMSprop 基础上使用动量与偏差修正。 RMSprop 、 Adadelta 与 Adam 在类似的情形下的表现差不多。 Kingma 指出收益于偏差修正， Adam 略优于 RMSprop ，因为其在接近收敛时梯度变得更加稀疏。因此， Adam 可能是目前最好的 SGD 优化方法。

并行和分布式

- Hogwild

  Niu[23]提出了被称为Hogwild的并行SGD方法。该方法在多个CPU时间进行并行。处理器通过共享内存来访问参数，并且这些参数不进行加锁。它为每一个cpu分配不重叠的一部分参数（分配互斥），每个cpu只更新其负责的参数。该方法只适合处理数据特征是稀疏的。该方法几乎可以达到一个最优的收敛速度，因为cpu之间不会进行相同信息重写。

- Downpour SGD

  Downpour SGD是Dean[4]提出的在DistBelief(Google TensorFlow的前身)使用的SGD的一个异步变种。它在训练子集上训练同时多个模型副本。这些副本将各自的更新发送到参数服务器(PS,parameter server)，每个参数服务器只更新互斥的一部分参数，副本之间不会进行通信。因此可能会导致参数发散而不利于收敛。

- Delay-tolerant Algorithms for SGD

  McMahan与Streeter[12]扩展AdaGrad，通过开发延迟容忍算法(delay-tolerant algorithms)，该算法不仅自适应过去梯度，并且会更新延迟。该方法已经在实践中表明是有效的。

- TensorFlow

  TensorFlow[13]是Google开源的一个大规模机器学习库，它的前身是DistBelief。它已经在大量移动设备上或者大规模分布式集群中使用了，已经经过了实践检验。其分布式实现是基于图计算，它将图分割成多个子图，每个计算实体作为图中的一个计算节点，他们通过Rend/Receive来进行通信。具体参见这里。

- Elastic Averaging SGD

  Zhang等[14]提出Elastic Averaging SGD(EASGD)，它通过一个elastic force(存储参数的参数服务器中心）来连接每个work来进行参数异步更新。

优化 SGD 的其他策略

- Shuffling and Curriculum Learning

  为了使得学习过程更加无偏，应该在每次迭代中随机打乱训练集中的样本。

  另一方面，在很多情况下，我们是逐步解决问题的，而将训练集按照某个有意义的顺序排列会提高模型的性能和 SGD 的收敛性，如何将训练集建立一个有意义的排列被称为 Curriculum Learning。

  Zaremba 与 Sutskever 在使用 Curriculum Learning 来训练 LSTMs 以解决一些简单的问题中，表明一个相结合的策略或者混合策略比对训练集按照按照训练难度进行递增排序要好。

- Batch normalization

  为了方便训练，我们通常会对参数按照 0 均值 1 方差进行初始化，随着不断训练，参数得到不同程度的更新，这样这些参数会失去 0 均值 1 方差的分布属性，这样会降低训练速度和放大参数变化随着网络结构的加深。

  Batch normalization 在每次 mini-batch 反向传播之后重新对参数进行 0 均值 1 方差标准化。这样可以使用更大的学习速率，以及花费更少的精力在参数初始化点上。 Batch normalization 充当着正则化、减少甚至消除掉 Dropout 的必要性。

- Early stopping

  在验证集上如果连续的多次迭代过程中损失函数不再显著地降低，那么应该提前结束训练，详细参见 NIPS 2015 Tutorial slides，或者参见防止过拟合的一些方法。

- Gradient noise

  Gradient noise 即在每次迭代计算梯度中加上一个高斯分布 N(0,σ2t)N(0,σt2) 的随机误差，即

优化 SGD 的其他策略（续）

- Early stopping

  在验证集上如果连续的多次迭代过程中损失函数不再显著地降低，那么应该提前结束训练，详细参见 NIPS 2015 Tutorial slides，或者参见防止过拟合的一些方法。

- Gradient noise

  Gradient noise 即在[illegible]个高斯分布 N(0,σ2t) 的随机误差，即

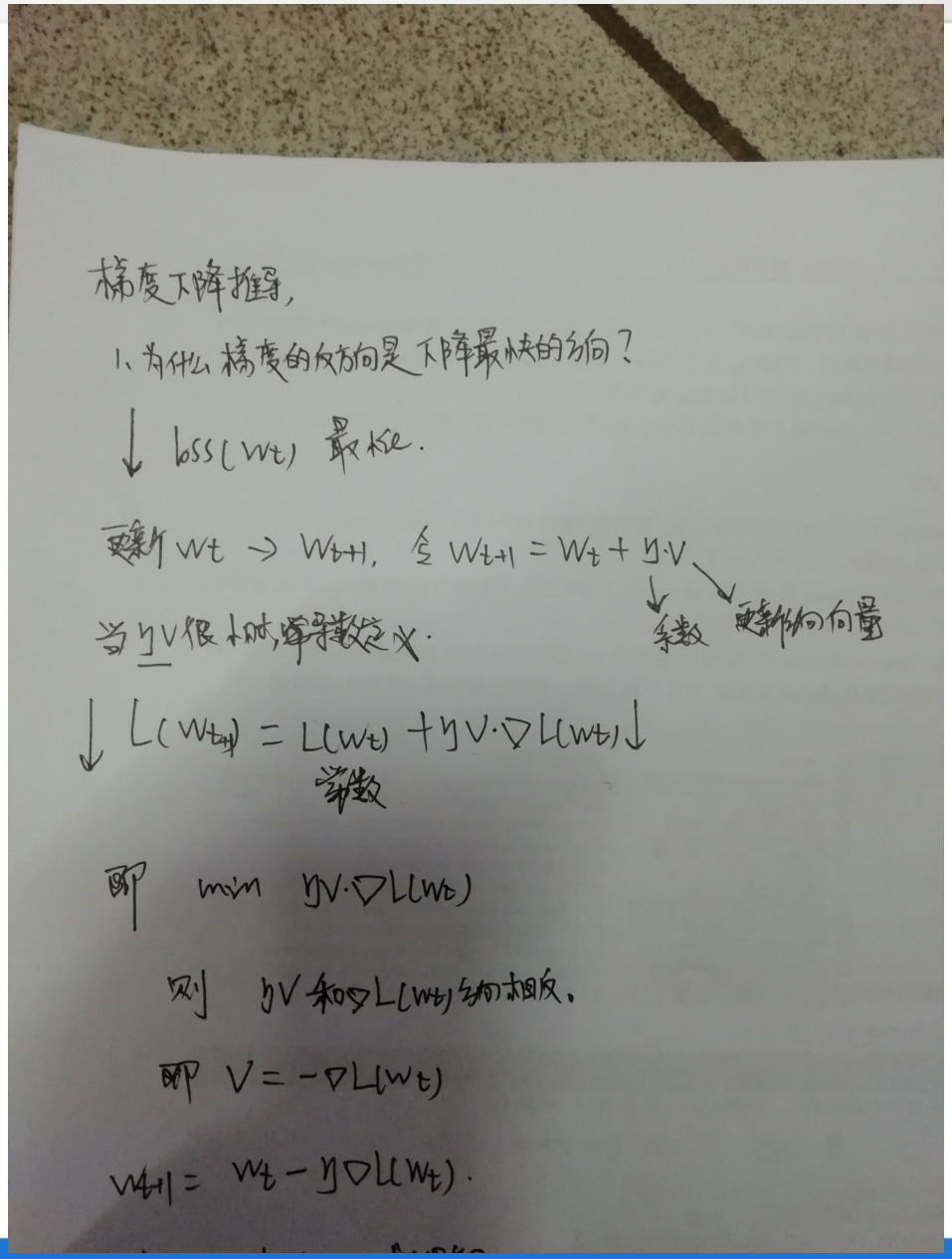
$$g_{t,i} = g_{t,i} + N(0, \sigma_t^2)$$

- 高斯误差的方差需要进行退[illegible]

$$\sigma_t^2 = \frac{\eta}{(1+t)^\gamma}$$

对梯度增加随机误差会增加模型的鲁棒性，即使初始参数值选择地不好，并适合对特别深层次的负责的网络进行训练。其原因在于增加随机噪声会有更多的可能性跳过局部极值点并去寻找一个更好的局部极值点，这种可能性在深层次的网络中更常见。

附录


梯度下降推导,
1、为什么梯度的反方向是下降最快的方向?
↓ loss(wt) 最小.
更新 wt → wt+1, 令 wt+1 = wt + ηV
η 系数
V 更新的向量
当ηV很小时,
↓ L(wt+1) = L(wt) + ηV·∇L(wt) ↓
常数
即 min ηV·∇L(wt)
则 ηV 和 ∇L(wt) 方向相反.
即 V = -∇L(wt)
wt+1 = wt - η∇L(wt).

附录

牛顿法求最优

① $L(w)$ 是二次可微的实函数，使用二次泰勒展开计算 w_t 的邻域

$$L(w) \approx \phi(w) = L(w_t) + \nabla L(w_t)(w - w_t) + \frac{1}{2}(w - w_t)\cdot\nabla^2 L(w_t)\cdot(w - w_t)$$

其中 $\nabla L(w_t)$ 是 L 的一阶导数矩阵（Jacobi矩阵），$\nabla^2 L(w_t)$ 是二阶导数矩阵，Hesse矩阵

为求 $\phi(x)$ 的驻点，令 $\nabla\phi(x) = 0$

$$\nabla\phi(x) = \nabla L(w_t) + \nabla^2 L(w_t)(w - w_t) = 0$$

即 $$w = w_t - \nabla^2 L(w_t)^{-1}\cdot\nabla L(w_t).$$

则可得到 $$w_{t+1} = w_t - \nabla^2 L(w_t)^{-1}\nabla L(w_t)$$

几个问题？为什么 $\phi(x)$ 的驻点是 $\nabla\phi(x) = 0$，（数学有时候）

这里如何感性理解：$t \to t+1$ 次更新参数

$$L(w_{t+1}) = L(w_t) + (\nabla L(w_t)\cdot(w_{t+1} - w_t) + \frac{1}{2}(w_{t+1} - w_t)\nabla^2 L(w_t)(w_{t+1} - w_t))$$

（$L(w_t)$ 常数）

$$\min f(w_{t+1})$$

$$\min f(w_{t+1}) = \nabla L(w_t)\cdot(w_{t+1} - w_t) + \frac{1}{2}(w_{t+1} - w_t)\nabla^2 L(w_t)\cdot(w_{t+1} - w_t))$$

可得：对取 $f(w_{t+1})$ 的极值点，得有

$$\nabla f(w_{t+1}) = 0 = \nabla L(w_t) + \nabla^2 L(w_t)(w_{t+1} - w_t) = 0$$

则 $$w_{t+1} = w_t - \nabla^2 L(w_t)^{-1}\nabla L(w_t)$$

Thank you !

中国科学院
计算机网络信息中心
Computer Network Information Center,
Chinese Academy of Sciences