如果训练数据很多，网络很大，一次训练所需的时间会很长。本周讲了一些技术，让我们的网络运行的更快！

1. Mini-batch梯度下降法

1.1 思想

对整个训练集进行一次Forward/Backward-Propagation，我们必须处理整个训练集，然后才能进行一步梯度下降，即每一次更新参数w, b需遍历整个训练集，如果训练集很大，如此做速度太慢！

如果我们每次处理训练数据的一部分就进行梯度下降法，则我们的算法执行速度会更快！而处理的一小部分训练集称为Mini-batch.

1.2 算法细节

符号规定如图所示：

$$\underset{(n_x,m)}{X} = [\underbrace{x^{(1)} \cdots x^{(1000)}}_{X^{\{1\}}\ (n_x,1000)}\ \underbrace{x^{(1001)} \cdots x^{(2000)}}_{X^{\{2\}}} \cdots \underbrace{\cdots x^{(m)}}_{X^{\{5000\}}}]$$

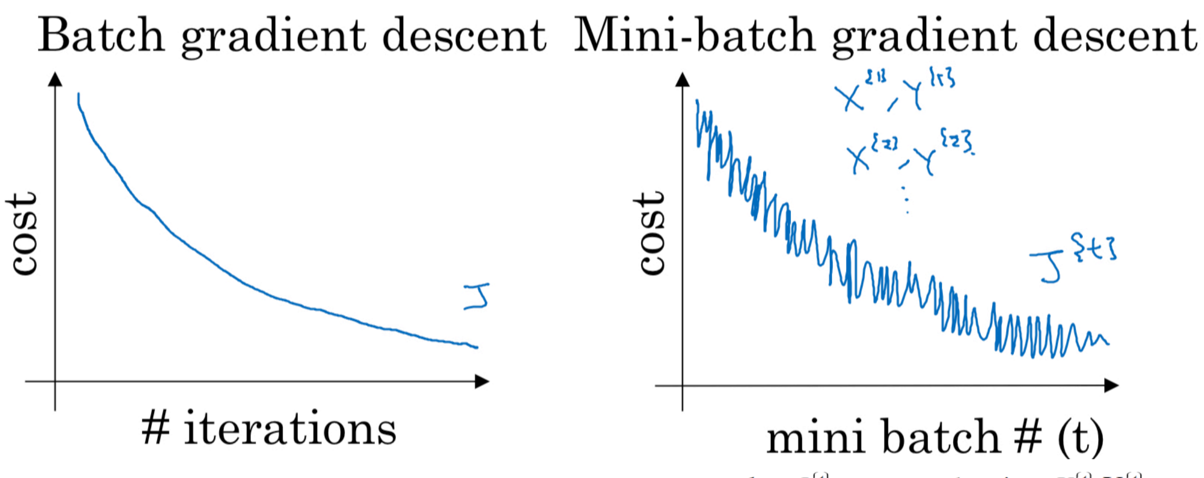
$$\underset{(1,m)}{Y} = [\underbrace{y^{(1)} \cdots y^{(1000)}}_{Y^{\{1\}}\ (1,1000)}\ \underbrace{y^{(1001)} \cdots y^{(2000)}}_{Y^{\{2\}}} \cdots \underbrace{\cdots y^{(m)}}_{Y^{\{5000\}}}]$$

Mini-batch的执行过程：

$$\text{for } i \text{ in range(epochs):}$$
$$\quad \text{for } t \text{ in range(\#mini\_batch)\{}$$
$$\quad\quad \text{forwardprop on } X^{\{t\}}$$
$$\quad\quad\quad Z^{[1]} = W^{[1]}X^{\{t\}} + b^{[1]}$$
$$\quad\quad\quad A^{[1]} = g^{[1]}(Z^{[1]})$$
$$\quad\quad\quad \ldots$$
$$\quad\quad\quad A^{[l]} = g^{[l]}(Z^{[l]})$$
$$\quad\quad J^{\{t\}} = \frac{1}{1000}\sum_{i=1}^{1000} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2 \cdot 1000}\sum_{l} \|w^{[l]}\|_F^2$$
$$\quad\quad \text{Backprop to compute } grads$$
$$\quad\quad W^{[l]} := W^{[l]} - \alpha \cdot dW^{[l]}, \quad b^{[l]} := b^{[l]} - \alpha \cdot db^{[l]}$$
$$\quad \}$$
$$\}$$

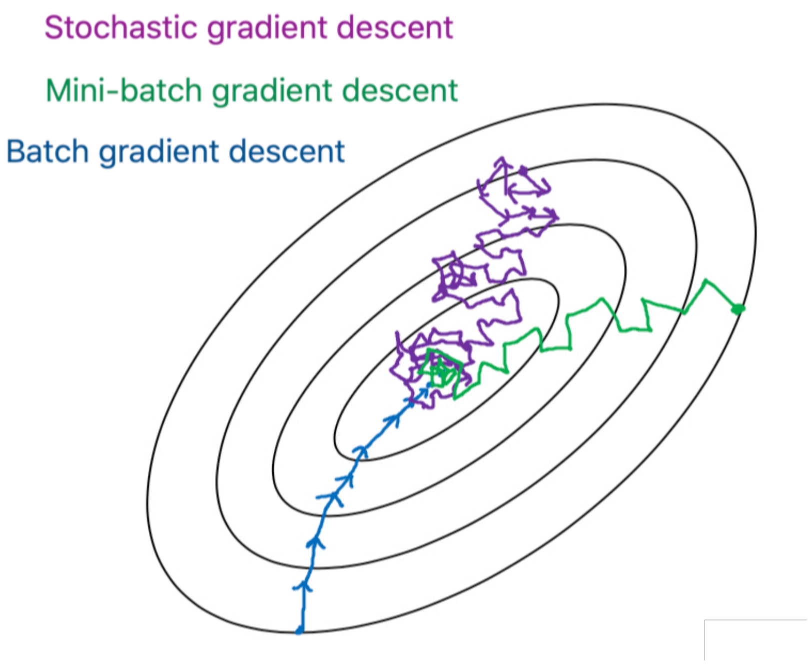
1.3 mini-batch size选择

普通的batch和mini-batch梯度下降法的变化趋势：



plot $J^{\{t\}}$ computed using $X^{\{t\}}, Y^{\{t\}}$

1. mini-batch size = m，即普通的Batch gradient descent
    - 对所有m个样本执行一次梯度下降，每一次迭代时间较长；
    - Cost总是随着迭代次数的增加而下降
2. Mini-batch size = 1，即Stochastic gradient descent
    - 对每一个样本执行一次梯度下降，这样就没法利用Vectorization带来的加速；
    - Cost总体的趋势是向最小值方向下降，但无法收敛到全局最优点，在全局最优点附近波动。
3. Mini-batch size in-between 1 and m
    - 既可以实现快速学习，又可利用Vectorization带来的加速；
    - Cost的下降曲线处于Bath gradient descent和SGD之间。



Mini-batch size选择：

1. 如果训练样本的大小比较小，如$m \leq 2000$时，直接选择Bath gradient descent法；
2. 如果训练样本比较大，典型的mini-batch size:

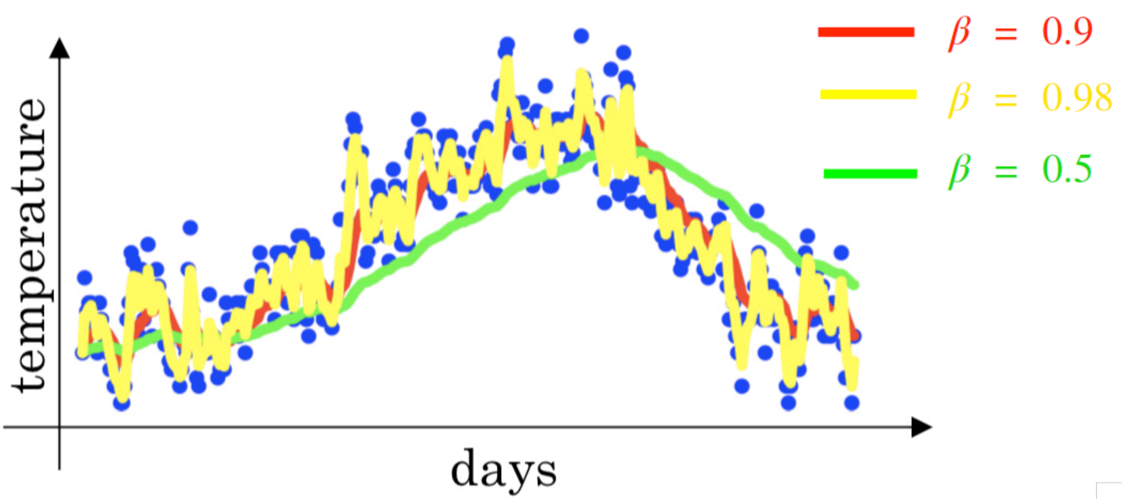
   64, 128, 256, 512
3. Mini-batch的大小要与CPU/GPU内存相匹配。

2. 指数加权平均

2.1 什么是指数加权平均

指数加权的通项公式：

$$
\begin{aligned}
v_0 &= 0 \\
v_t &= \beta \cdot v_0 + (1-\beta) \cdot \theta_t
\end{aligned}
$$



2.2 理解指数加权平均

例子，当$\beta=0.9$时：

$$
\begin{aligned}
v_{100} &= 0.9v_{99}+0.1\theta_{100} \\
v_{99} &= 0.9v_{98}+0.1\theta_{99} \\
v_{98} &= 0.9v_{97}+0.1\theta_{98} \\
\ldots
\end{aligned}
$$

展开后：

$$v_{100}=0.1\theta_{100}+0.1\times 0.9\theta_{99}+0.1\times(0.9)^2\theta_{98}+\cdots+0.1\times(0.9)^{99}\theta_1$$

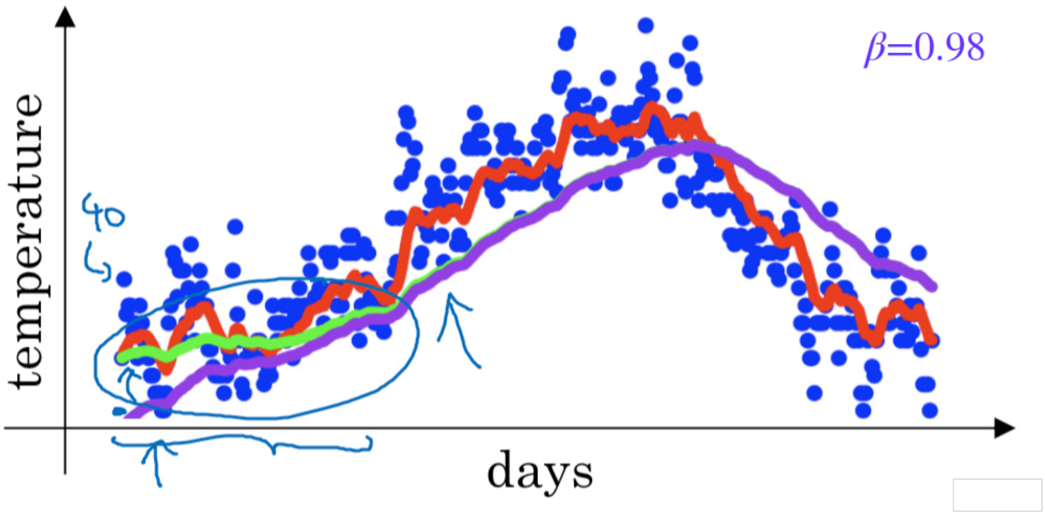
可以看到，离当前时间越近的的温度的权重越重，时间越远，对当前v的贡献越小。

当ε比较小时，有$(1-\varepsilon)^{1/\varepsilon}\approx 1/e$，在我们的例子中，$(1-0.1)^{10}\approx 0.35\approx 1/e$。也就是说大于10天后，系数的峰值下降到原来的$1/e$，约等于关注了过去10天的温度。

因此：

- $\beta=0.9$ 约等于过去10天的平均温度；
- $\beta=0.98$ 约等于过去50天的平均温度；
- $\beta=0.5$ 约等于过去2天的平均温度

2.3 指数加权平均的偏差修正



当$\beta=0.98$时，我们按照上面的公式计算得到的并不是绿色曲线，而是紫色曲线，why?

$$
\begin{aligned}
v_0 &= 0 \\
v_1 &= 0.98v_0+0.02\theta_1=0.02\theta_1 \\
v_2 &= 0.98v_1+0.02\theta_2=0.0196\theta_1+0.02\theta_2
\end{aligned}
$$

因为初始值$v_0=0$，所以前几项会明显小于实际值，为了解决这个问题，引入了偏差修正：

$$v_t=\frac{v_t}{1-\beta^t}$$

因此上面的计算修正后如下：

$$
\begin{aligned}
v_0 &= 0 \\
v_1 &= \frac{0.02\theta_1}{1-0.98^1} \\
v_2 &= \frac{0.0196\theta_1 + 0.02\theta_2}{1-0.98^2}
\end{aligned}
$$

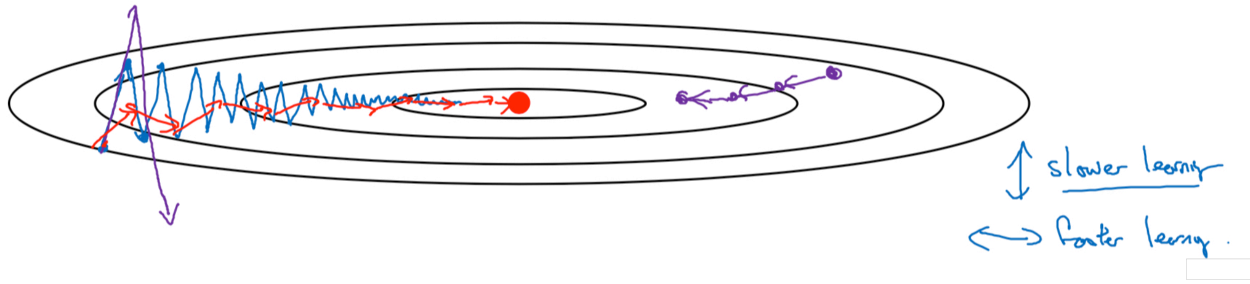
虽然原始的指数加权公式存在偏差，但在实际使用中，因为多次迭代后，前期的偏差可忽略不计，因此一般很少有人加偏差修正系数。

3. 动量(Momentum)梯度下降法

思想：计算梯度的指数加权平均数，并利用该梯度更新权重。

3.1 问题分析

在优化Cost Function的过程中，梯度下降的执行过程可能如下：



利用梯度下降法最小化Cost Function的过程中，每一次迭代更新的参数变化曲线如图中蓝色线所示呈上下波动趋势。这种幅度较大的波动，减缓的梯度下降的速度，而且我们只能用一个较小的学习率老进行迭代。如果用较大的学习率，则有可能如图紫色线一样，偏离函数范围。

我们希望的是每一次迭代，梯度在纵向上的幅度小些，而在横轴上下降快些，如此可能更快达到最小点。可以用动量梯度下降法实现，如图中红线所示。

3.2 算法实现

On iteration t:

Compute dW, db on the current mini-batch

$$\boxed{\begin{aligned} v_{dW} &= \beta v_{dW} + (1-\beta) dW \\ v_{db} &= \beta v_{db} + (1-\beta) db \end{aligned}} \quad \Bigg| \quad \begin{aligned} v_{dW} &= \beta v_{dW} + dW \\ v_{db} &= \beta v_{db} + db \end{aligned}$$

$$W = W - \alpha v_{dW}, \quad b = b - \alpha v_{db}$$

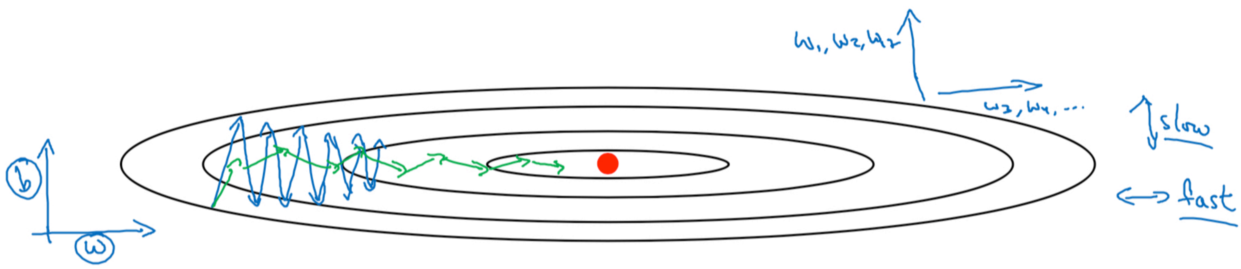
一般地，$\beta = 0.9$

有的人可能会写成左右边的形式，两种形式都是OK的。

这里并没有使用偏差修正，主要是经过几次迭代后，移动平均已经经过了初始阶段，不再是一个具有偏差的预测，所以在实践中，一般不做偏差修正。

4. RMSprop

RMSprop(root mean sqart prop)是另外一种加速梯度下降的算法。



On iteration t:

Compute dW, db on the current mini-batch

$$s_{dW} = \beta_2 v_{dW} + (1-\beta_2) dW^2$$

$$s_{db} = \beta_2 v_{db} + (1-\beta_2) db^2$$

$$W = W - \alpha \frac{dW}{\sqrt{s_{dW} + \varepsilon}}, \quad b = b - \alpha \frac{db}{\sqrt{s_{db} + \varepsilon}}$$

$$\varepsilon = 10^{-8}$$

原理：在上图的例子中，$dW < db$，在更新W, b时，W的变化幅度反而要大于b。

5. Adam

Adam(Adapt moment estimation)，可以看做是Momentum和RMSprop两种优化算法的结合。

$V_{dw}=0,\ S_{dw}=0,\ V_{db}=0,\ S_{db}=0,$

On iteration t:

Compute dW, db on the current mini-batch

$$V_{dw} = \beta_1 V_{dw} + (1-\beta_2) dW,\ V_{db} = \beta_1 V_{db} + (1-\beta_2) db$$

$$S_{dw} = \beta_2 v_{dW} + (1-\beta_2) dW^2,\ S_{db} = \beta_2 v_{db} + (1-\beta_2) db^2$$

$$V_{dw}^{corrected} = V_{dw} \ / \ (1-\beta_1^t),\quad V_{db}^{corrected} = V_{db} \ / \ (1-\beta_1^t)$$

$$S_{dw}^{corrected} = S_{dw} \ / \ (1-\beta_2^t),\quad S_{db}^{corrected} = S_{db} \ / \ (1-\beta_2^t)$$

$$W = W - \alpha \frac{V_{dw}^{corrected}}{\sqrt{S_{dw}^{corrected}}+\varepsilon},\ b = b - \alpha \frac{V_{db}^{corrected}}{\sqrt{S_{db}^{corrected}}+\varepsilon}$$

V_{dw}、S_{dw}可以看做是dW的一阶矩估计和二阶矩估计，可以看做是对期望$E[dW]$、$E[dW^2]$的近似；$V_{dw}^{corrected}$、$S_{dw}^{corrected}$是对V_{dw}、S_{dw}的校正，可以看做对期望的无偏估计。

超参数的选择：

- α : need to be tune
- β_1： 0.9
- β_2： 0.999
- ε ： 10^{-8}

6. Learning rate decy

学习率衰减的主要作用：

1. 在迭代初期，学习率相对较大，梯度能较快向最小值点下降；
2. 而在快下降到最小值点时，学习率变小，避免在最小值点附近出现较大波动。

学习率衰减的实现：

1. 常用：

$$\alpha = \frac{1}{1+\text{decay\_rate}\times\text{epoch\_num}}\alpha_0$$

2. 指数衰减：

$$\alpha = 0.95^{\text{epoch\_num}}\cdot\alpha_0$$

3. 其它

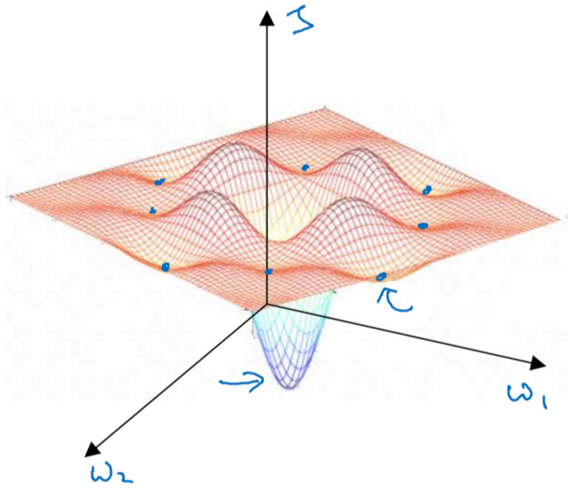
$$\alpha = \frac{k}{\text{epoch\_num}}\alpha_0$$

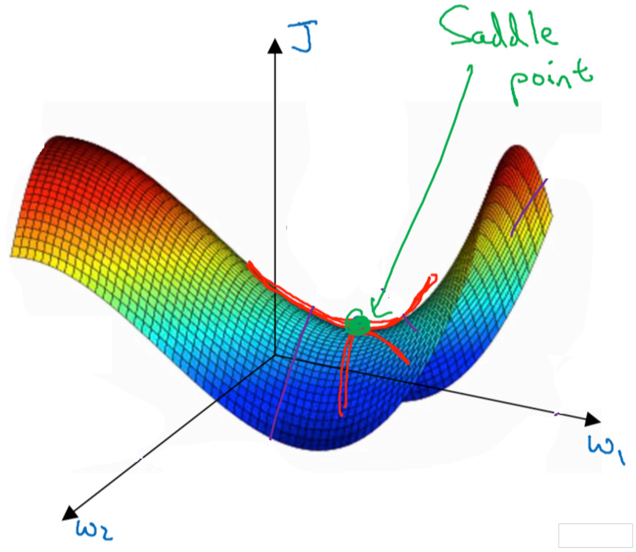
4. 离散下降

在不同跌点阶段使用不同的学习率

5. Manual decay

7. 局部最优问题





人们对局部最优认识的变化，如上图：

1. 在低纬度情况下，人们最初想象的Cost Function如作图所示，存在一些局部最小值点，如果参数初始化不当，可能陷入局部最优情况；
2. 而实际上，一个神经网络梯度为0的点而是右图所示的鞍点。

why?

作图所示的局部最优要求在所有维度上都同时为凹函数，在神经网络中，参数众多，要求所有参数在梯度为0的点都是凹函数的可能性太低太低。也就是说在高维情况下，局部最优更可能是鞍点！

因此，在高维度情况下：

1. 几乎不可能陷入局部最优点；
2. 在鞍点的停滞区，会减缓学习过程，可利用如Adam等算法改善。

8. 遗留问题

Q1: momentum和Adam公式的原理是什么？