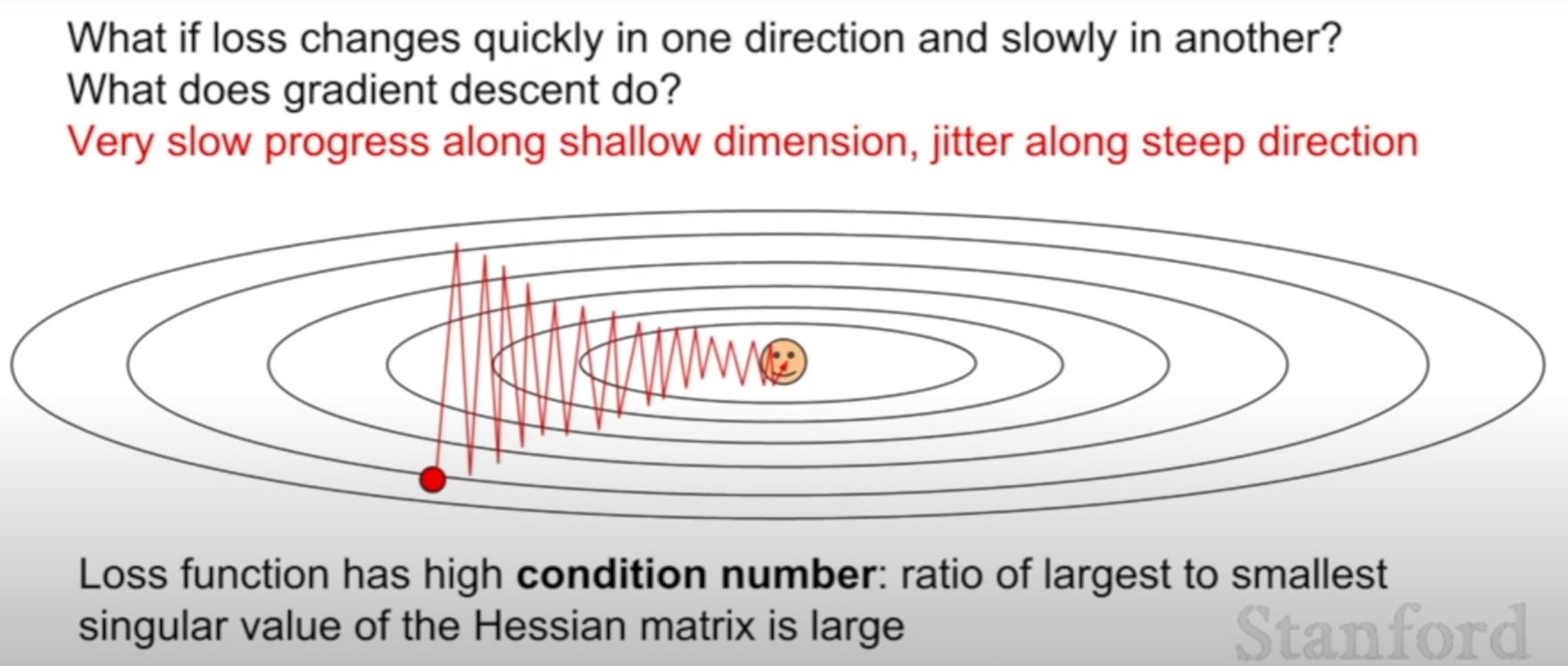
**Vanilla SGD问题：**

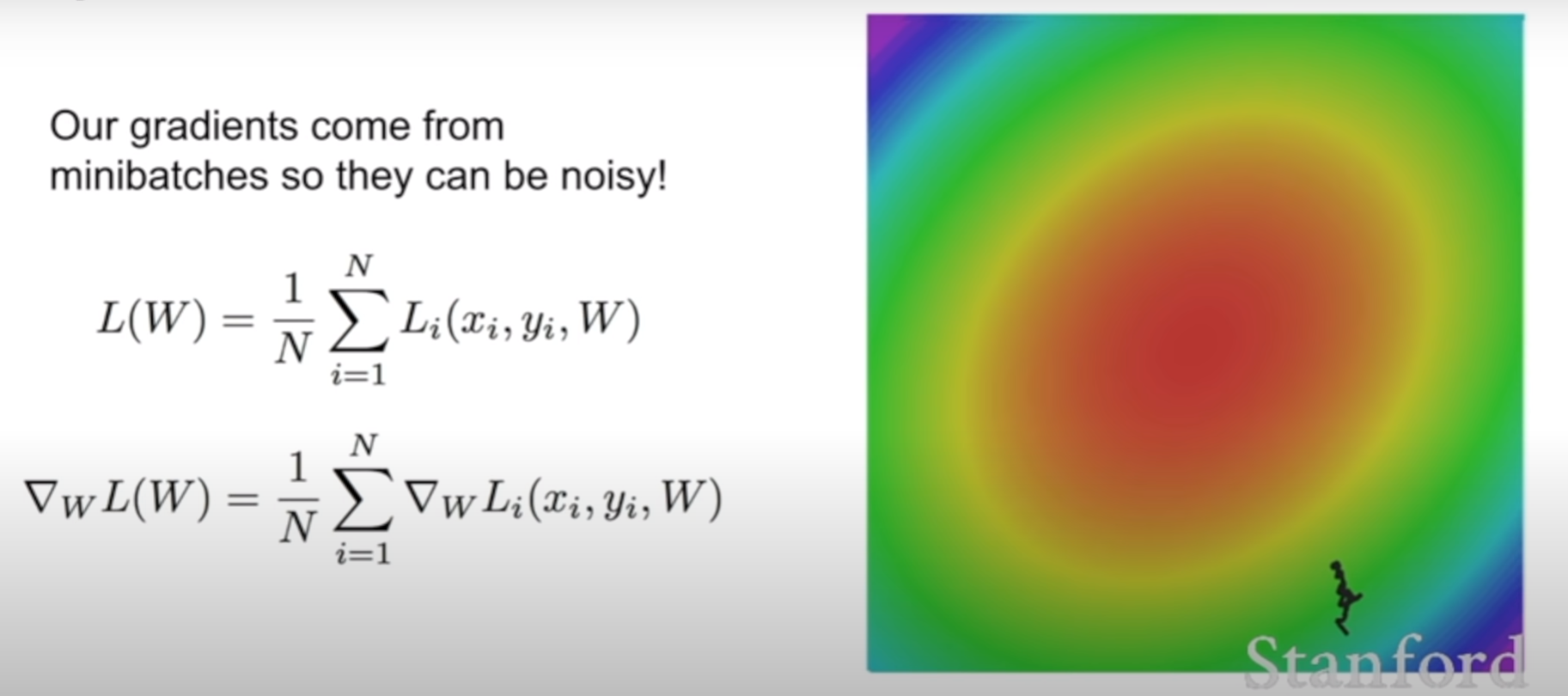
（1）一个方向loss变化很快，一个方向变化很慢（loss函数具有很高的condition number：Hessian矩阵中最大最小奇异值之和很大），会导致梯度Z字型变化，在高纬度的情况会更加明显。



(2) loss函数存在局部最小点或者鞍点：梯度为0，卡在这些位置。但是在实际应用中，这两个不算是大问题，局部最小点的结果也可能表现出不错的性能。在高维度中鞍点更常见。



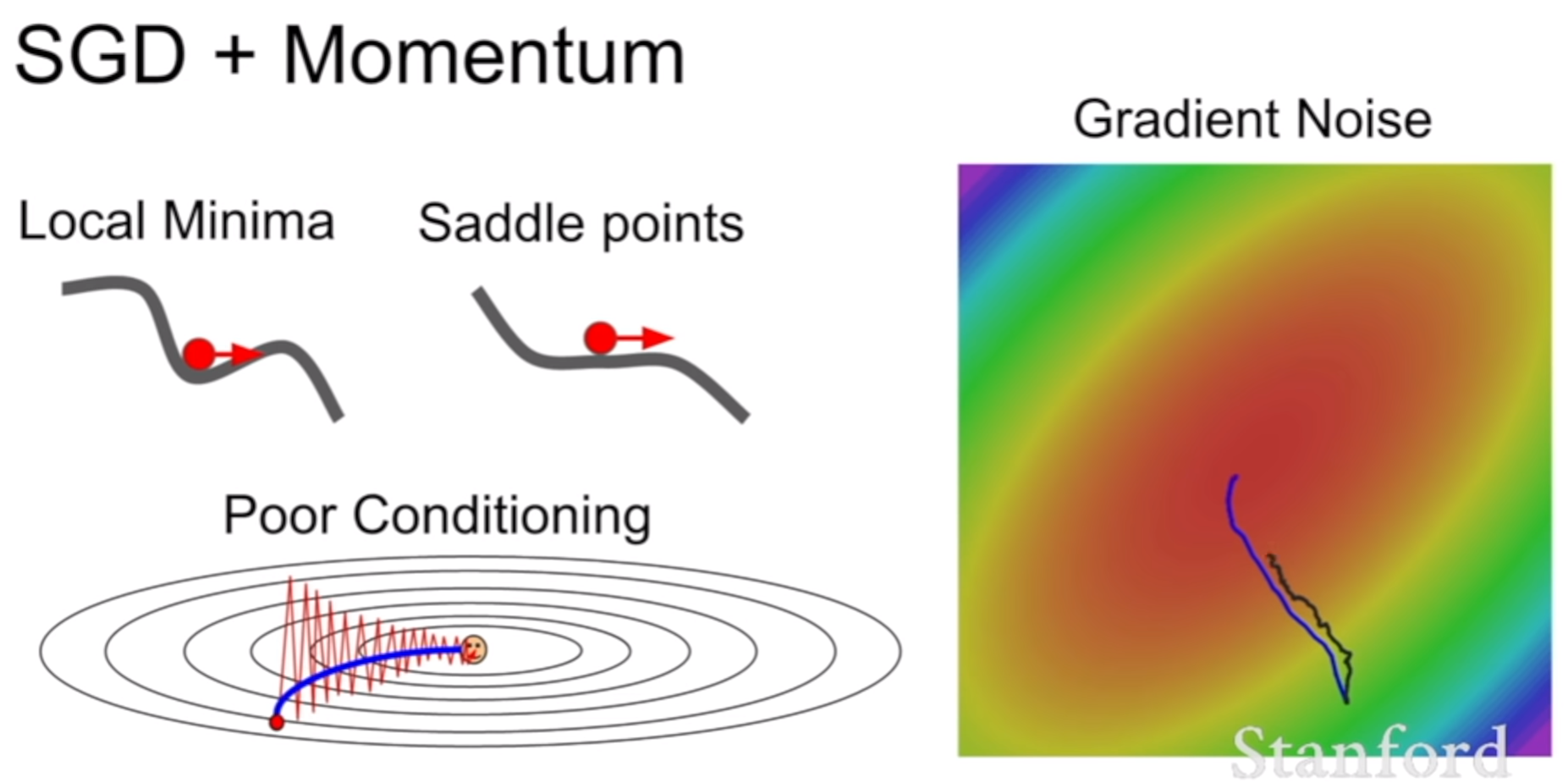
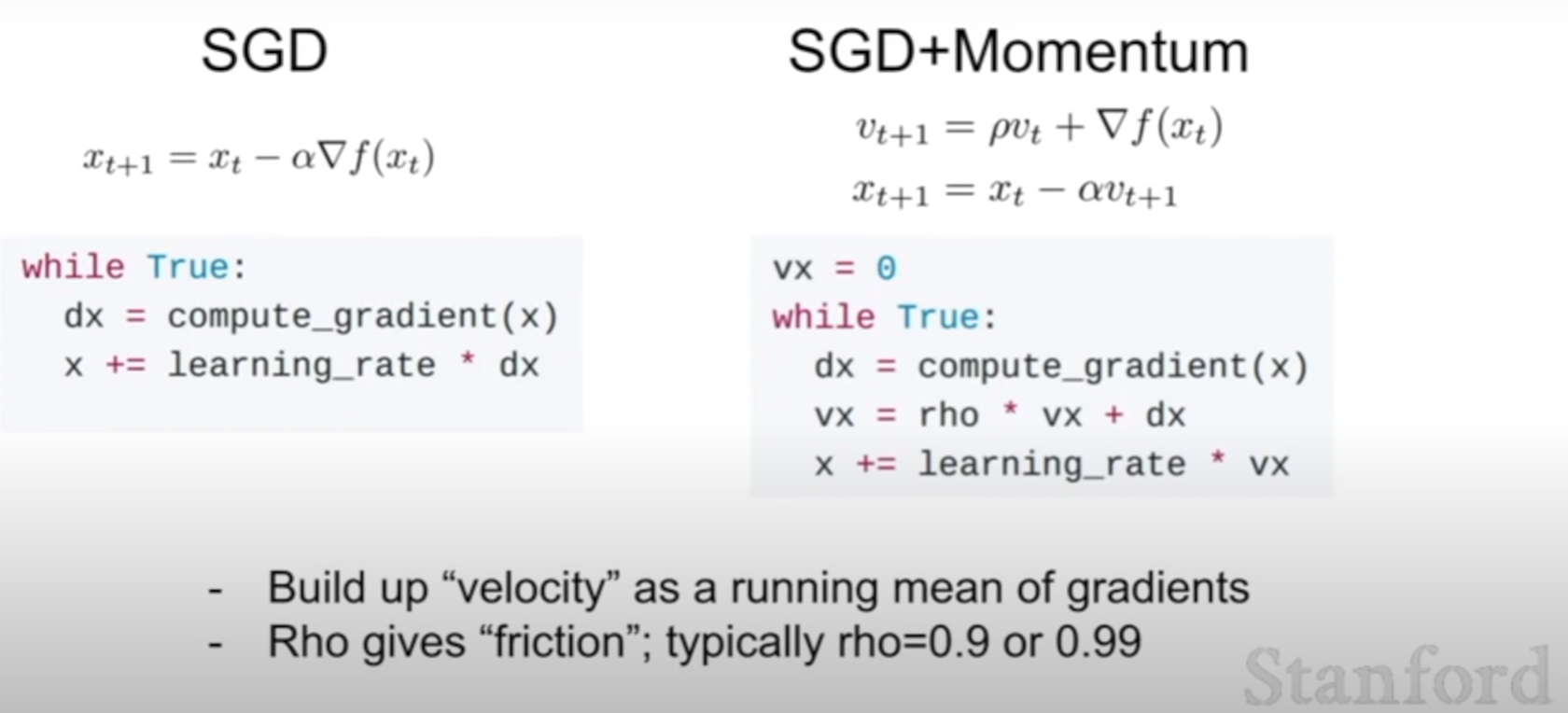
（3）来自于SGD中的S (stochastic)：因为训练的时候不是采用整个数据集，而是采用minibatch来评估loss或者梯度下降，也就是说我们并不是真实地在每一个时间步都获得真实的梯度信息，实际上我们获得的只是对梯度的noisy estaimation。

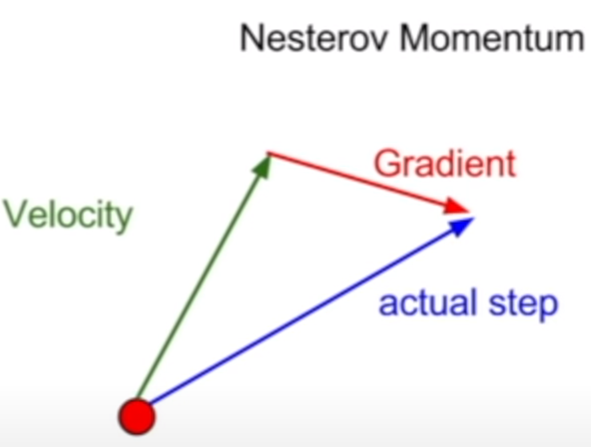
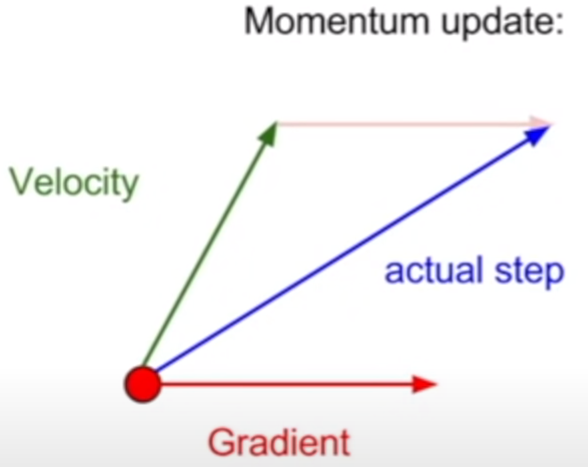


**SGD优化算法 ：**常见的优化操作就是对更新公式中的梯度项做出各种调整变化。

***SGD + Momentum***

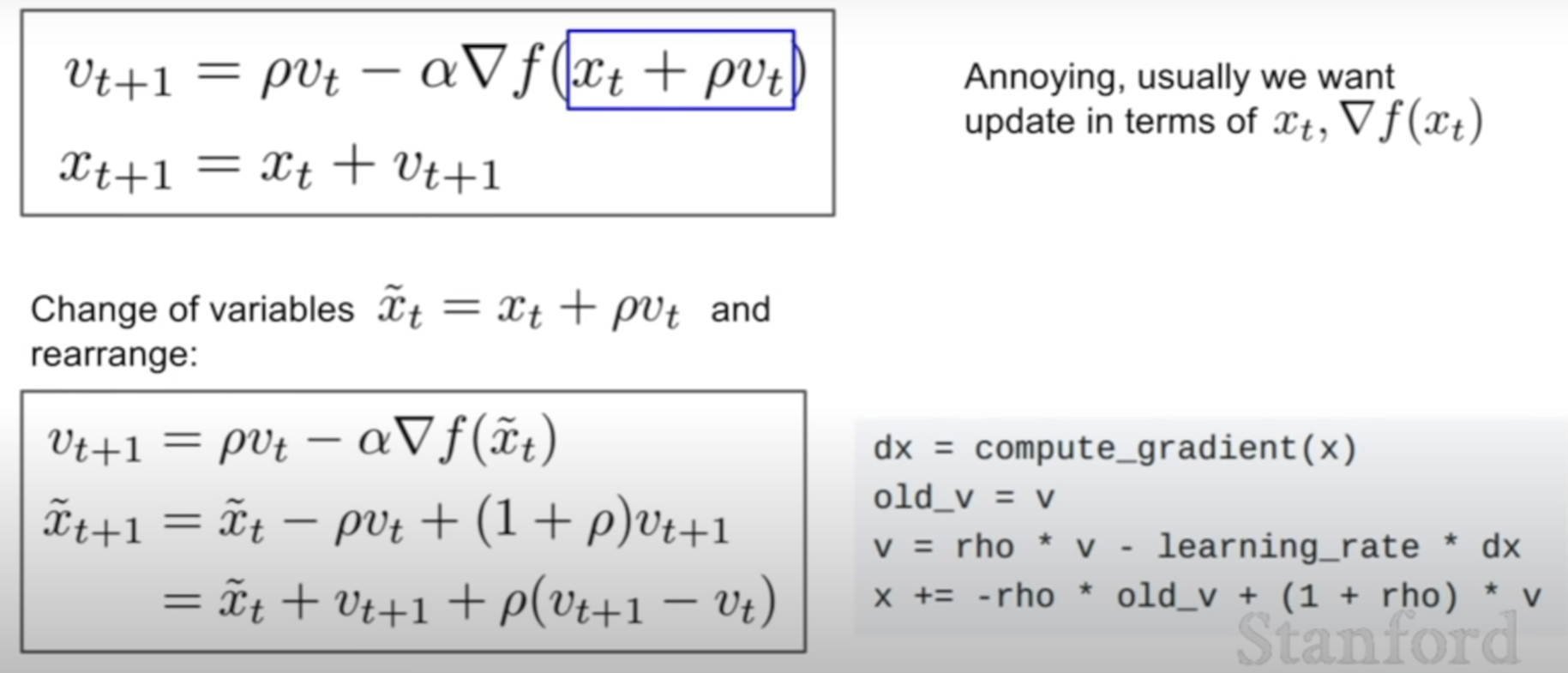
采用SGD时，如果表面曲线在一个方向比另一个方向更陡（常见于局部最优解中），SGD就会在这些峡谷斜率中徘徊犹豫不前。给梯度加个动量，动量就相当于对SGD在有关方向加速并减少波动。通过在梯度衰减项加上上一个动量，相当于权重衰减程度更大，因此加速梯度下降。其中ρ相当于摩擦，ρ一般为0.9~0.99。动量法类似滚雪球，越滚越快。它可以帮助跳出局部最小值的原因是，即使在该点的梯度为0，但是仍然具有速度或/动量/惯性，因此更容易跳出局部最小点。





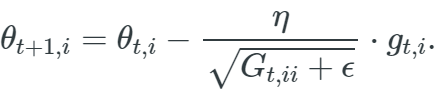
***Nesterov Momentum***

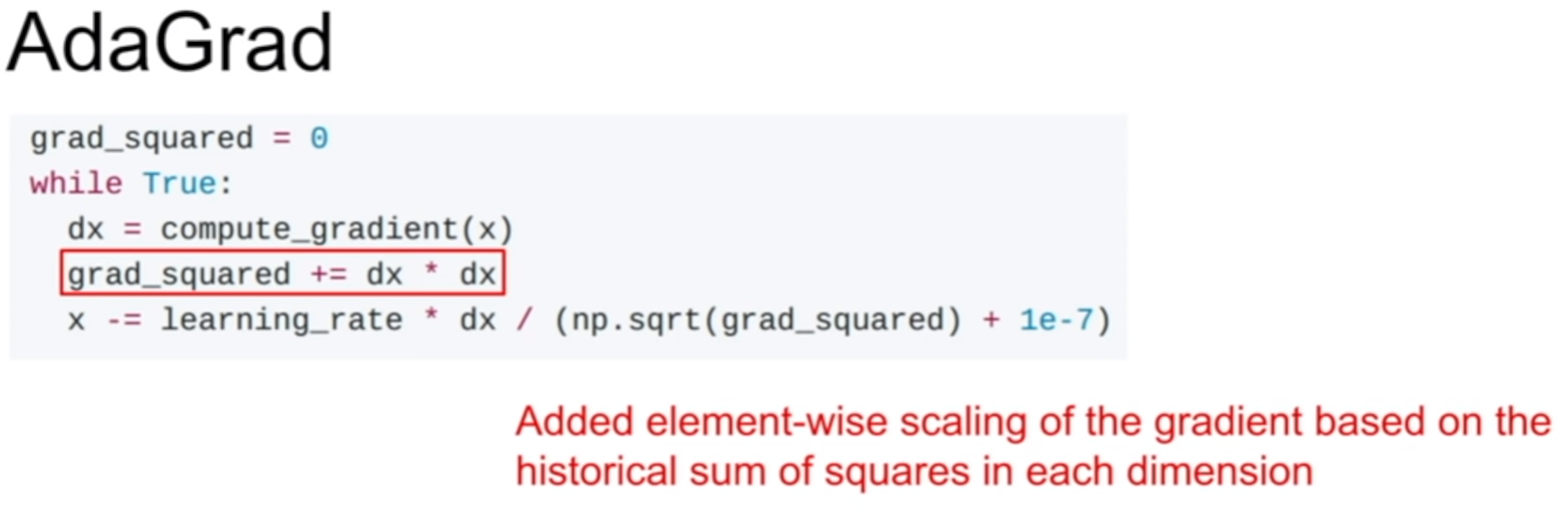
Nesterov动量法也是动量法的一种。跟普通动量法的区别是：普通动量法中的滚雪球方式是盲目跟着斜坡走的，但是不够智能。更智能的滚雪球应该是在知道斜坡要变向上时，提前进行减速。如何实现：先用速度矢量带到某一点的位置，然后在这个点再进行梯度下降。 Nesterov项在梯度更新时做一个校正，避免前进太快，同时提高灵敏度。从下面的速度之差可以看出，ρ起到调节速度的作用。



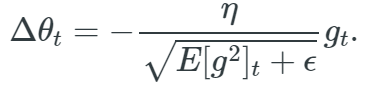
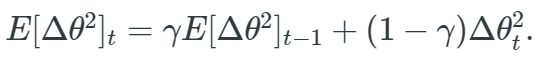
***Adagrad***

基于历史的每个方向的梯度平方和，对这个值开方后，取倒数作为梯度的elementwise scaling因子。假如两个梯度优化方向，一个下降很快，一个下降很慢，采用AdaGrad就会让下降很慢的加快，让下降很慢的减速。因为梯度更新中除以的是历史梯度平方和的开方，下降很慢表示梯度小，所以除了以后，梯度更新就变大，减速同理。对于凸模型，AdaGrad表现很好，因为当逐步靠近全局最小值时，梯度越来越小，那么除了scaling因子后就会更大，因此会导致学习率逐渐减小，然后达到最优值。但是对于非凸模型就不太好了，因为会逃不出局部最小值或者鞍点，因为学习步长逐渐减小。实际上在训练网络时不太用AdaGrad。1e-7可避免分母为0。

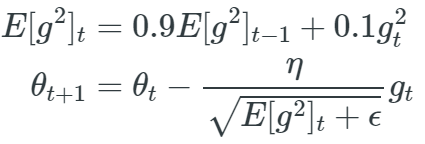
自适应修改learning rate，对于频繁出现的特征采用小α，对于非频繁出现的特征采用大α，因此很适合用于稀疏数据，对SGD很有效。Adagrad对每个参数θ在每个时间步都采用不同的α，表示在t时间步对θi的梯度。，，G是对角矩阵，对角线的元素为时间步t对θi的梯度的平方和。ε是辅助项，可避免除以0。Adagrad的一个好处是避免了手动调整learning rate，大部分情况把它设为0.01，然后就让它自己调整。缺点是分母是梯度平方的累积，且累积会一直增大。因此learning rate会一直减少，直到最终变得非常小，从而停留在某一节点就不继续了。

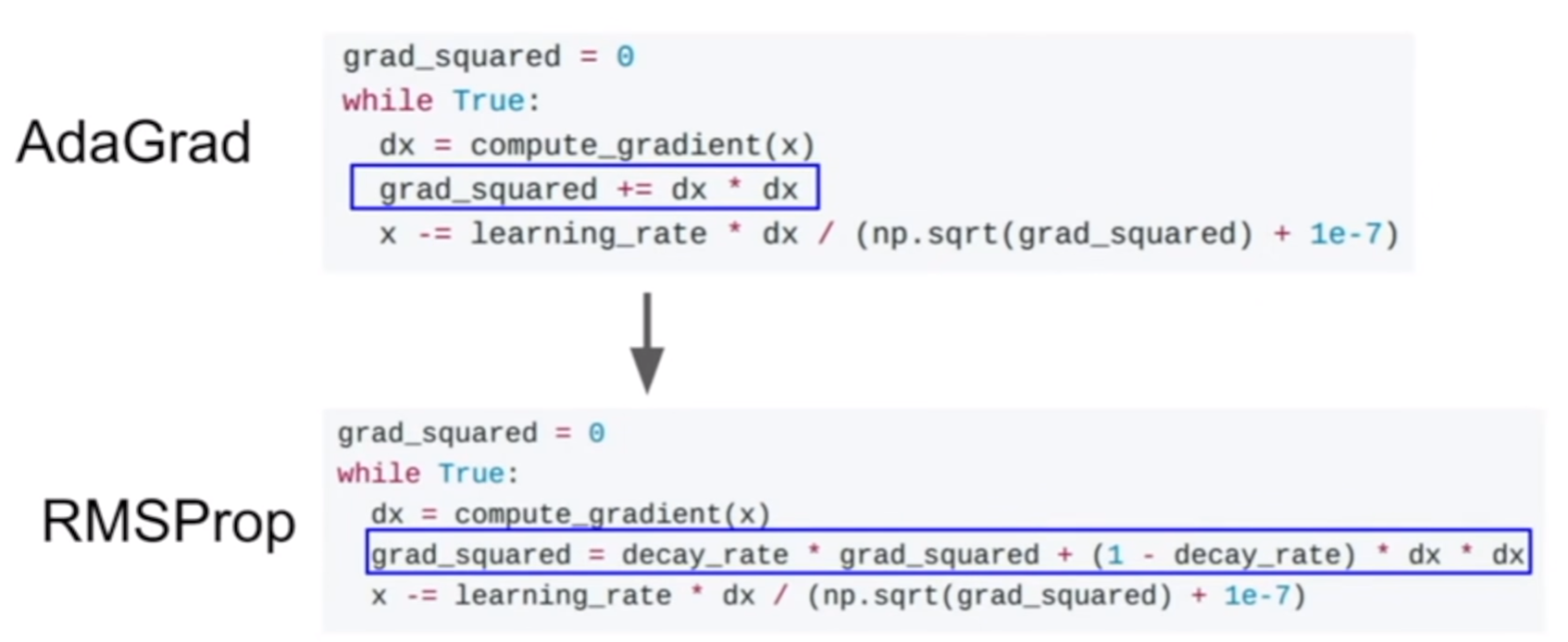


***Adadelta***

解决Adagrad中learning rate消失的问题，主要将过去累积梯度的窗口限制在了固定大小ω。将梯度平方和定义成所有过去梯度平方和的衰减平均值。，其中γ是小于1的小数，0.9左右。，主要修改了分母的形式。但是作者指出用E这个公式的话，单位不匹配，所以他们用训练参数重新定义，

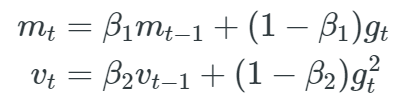
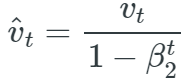
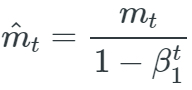
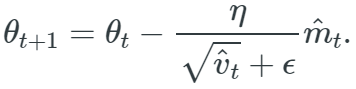
***RMSprop***

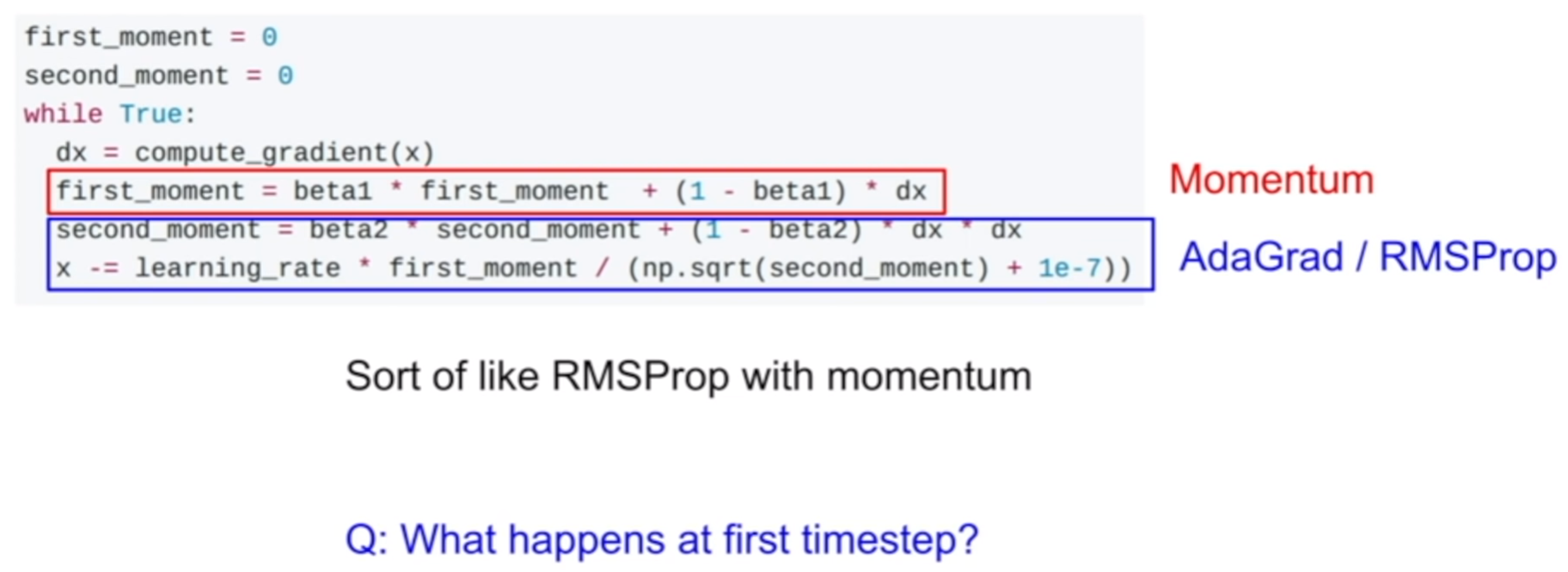
目的跟Adadelta一样，未发表算法。，本质上一样，直接建议把γ设为0.9



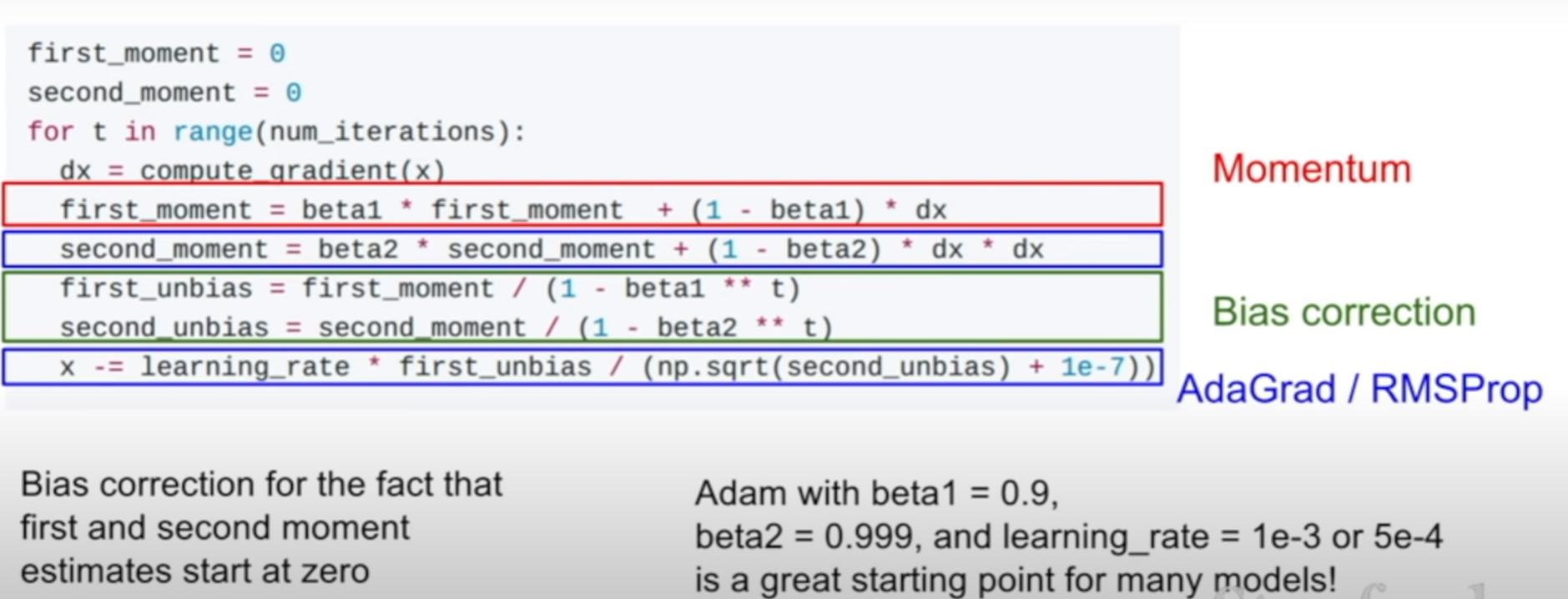
***Adam***

Adaptive Moment Estimation，把前面的动量和梯度平方累积的特点都综合考虑起来。对每个参数自适应计算learning rate。除了存储索过去梯度平方和的衰减平均值外，还保留了过去梯度值的指数衰减平均值，类似于动量。动量法像一个雪球一样往下滚，Adam像一个带有摩擦的重球，在误差平面中更倾向平坦的最小值。

，其中mt和vt指过去梯度和过去梯度平方。由于这两个参数被初始化为0向量，作者观察到他们的结果很容易倾向于0，尤其是在初始时间步和衰减率较小的时候。因此作者进行了修改：，用两个新参数作为更新公式。



在上面代码中，可以看到对于second\_moment初始化为0，在经过第一次更新后，这个变量还是很小，因为β2为0.9~0.99。然后在进行一部分权重更新时，要除以这个很小的数，结果就导致产生了很大的步长，在训练一开始就生成这么大的步长，显然是不利于训练的。然后first\_moment也是初始化很小，某些情况下两种可能抵消相互的影响，但是我们无法避免上述大步长的情况。下面是Adam最终的优化形式，也是上述提到的公式里面引入跟时间t有关的变量。

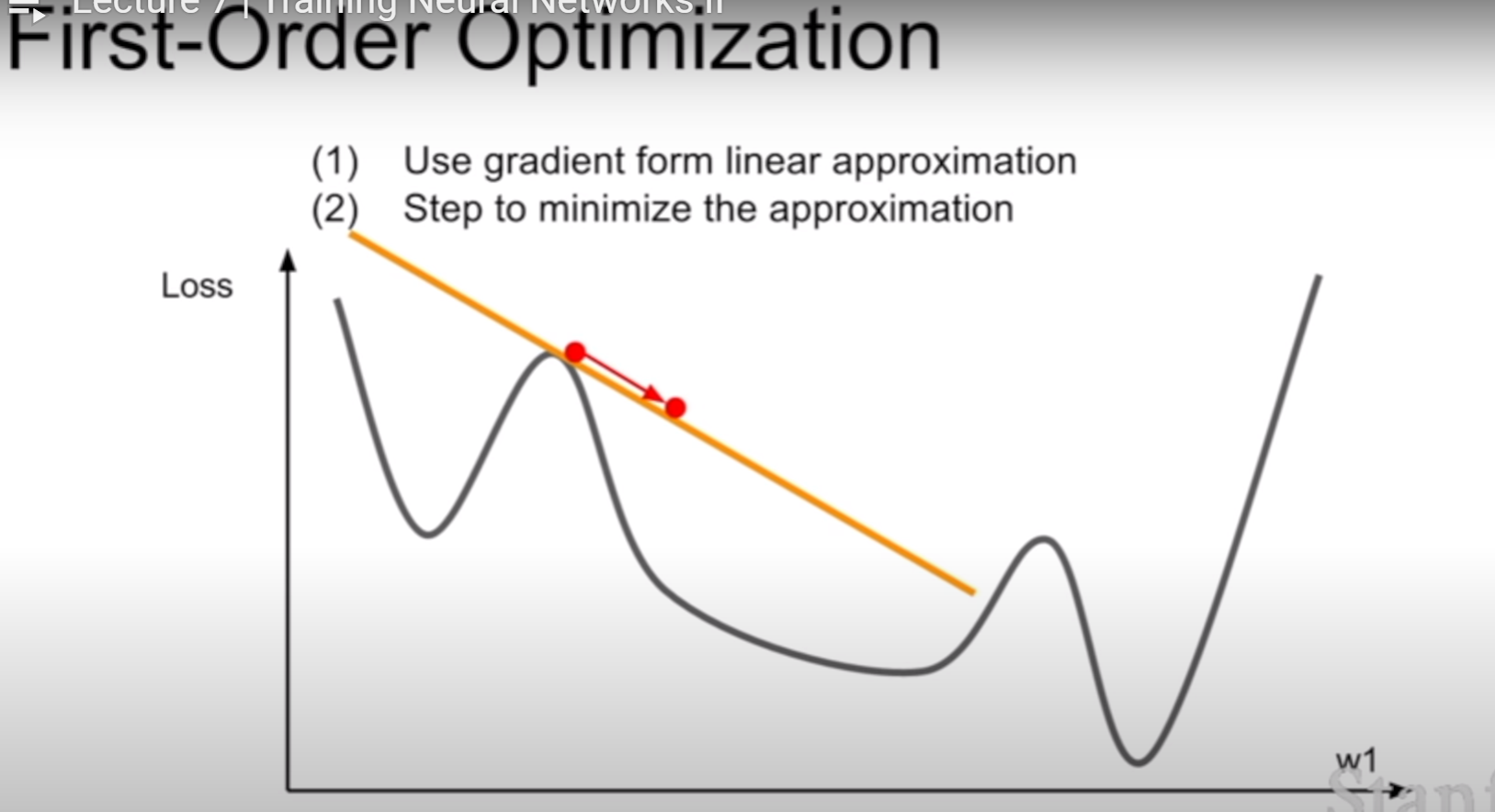
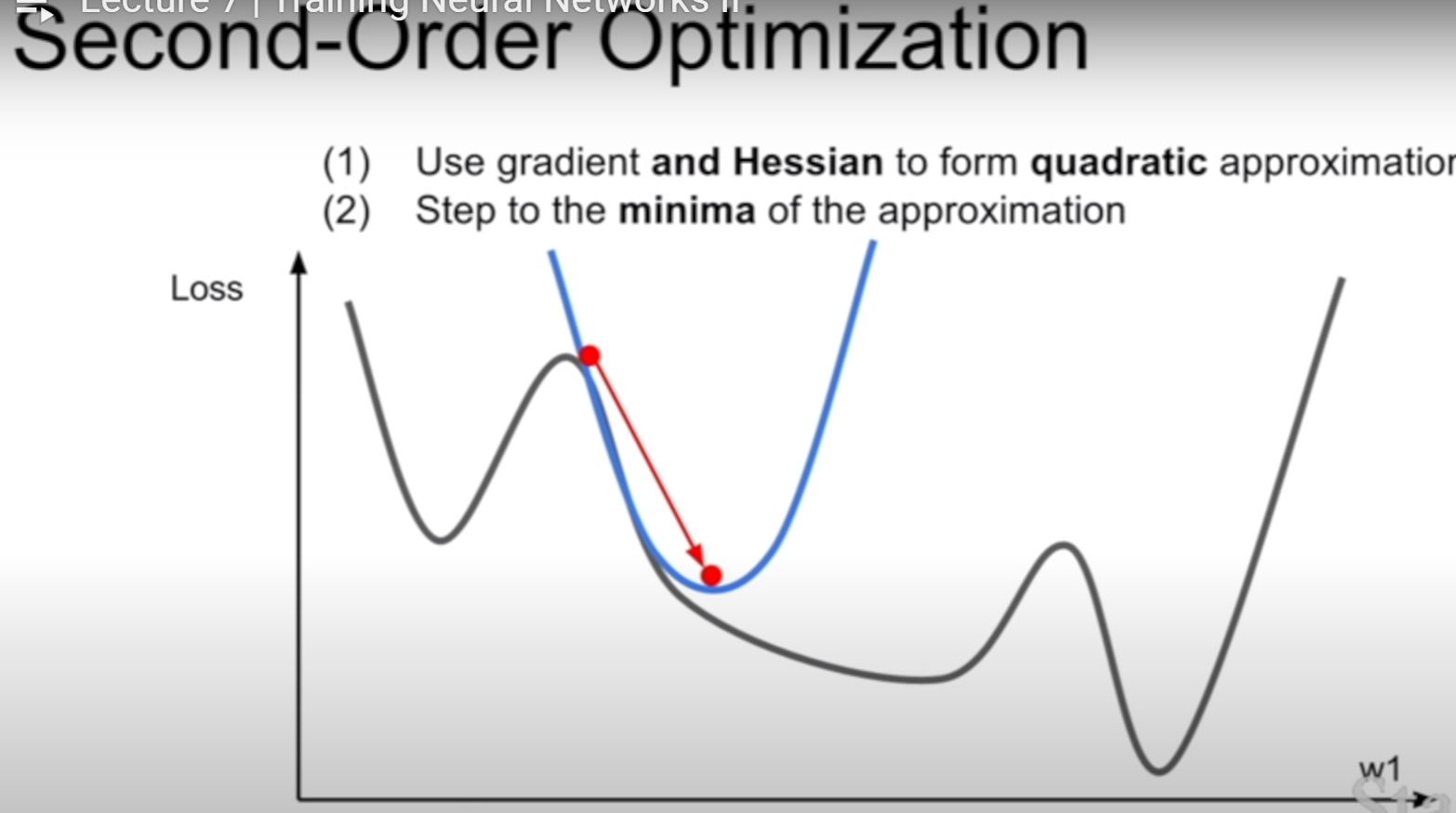


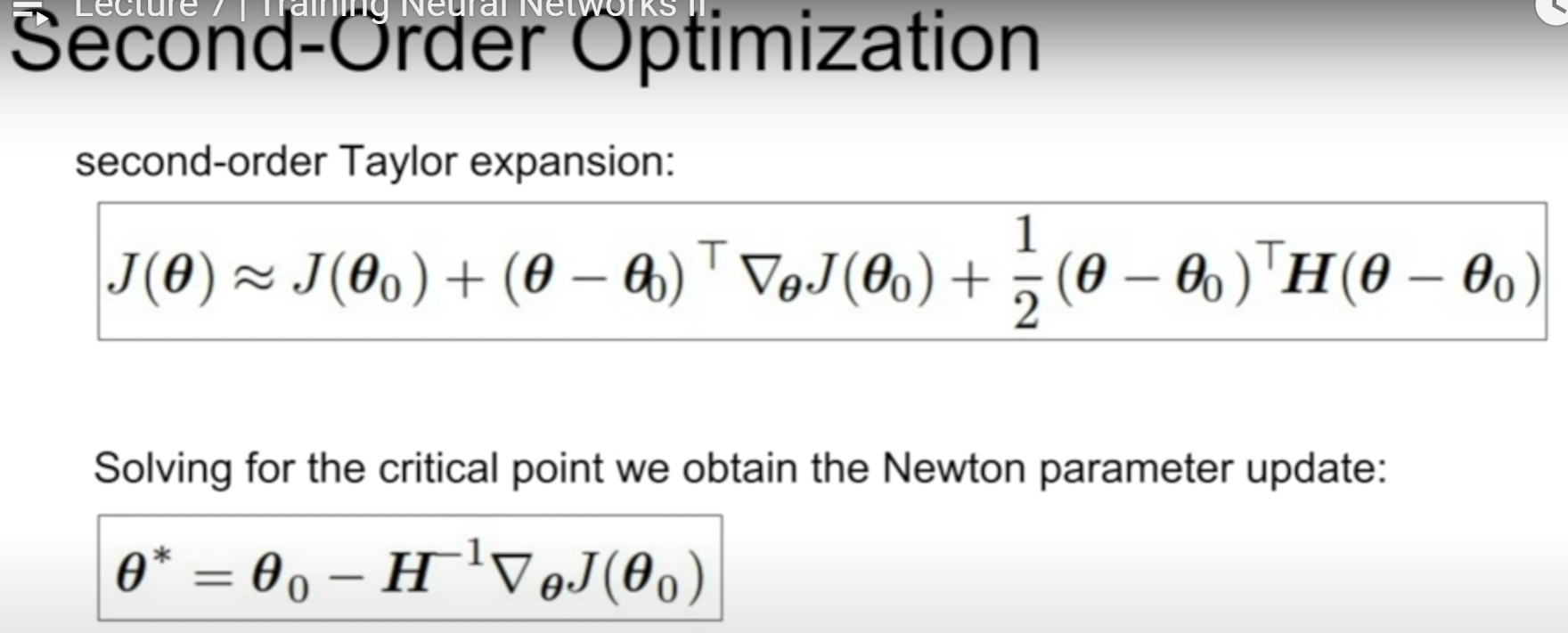
Adam（非完全形式）: momentum（第一项）和RMSProp（第二项）的结合。三个优点：第一个是帮助SGD跳出局部最小点和鞍点；第二个是momentum的功能，加速推进梯度下降过程；第三个是RMSProp功能，避免了momentum过度推进导致的overfit，可以调节控制不同方向的推进速度。缺点：在第一个时间步的时候，第二项非常接近0，即便是第一步更新后也是很接近0，这样就会导致在梯度更新公式中的更新项很大，这个问题不是模型本身带来了的问题，而是由于我们将其初始化为0导致的。

Adam（完全形式）：在非完全形式上添加了bias项，目的是为了消除上述的初始化问题。Adam对很多优化问题表现很好，可以当作是第一默认的优化选项，建议值：β1=0.9，β2=0.999，η=1e-3/5e-4

***二阶优化***

上述描述的优化都属于一阶优化。

没有学习率！！！但是这种二阶优化不适于深度学习，因为Hessian矩阵的维度为N^2，求逆时需要O(N^3)，而N可能为百万或千万。

为了解决这个问题，提出了一些近似的优化：

（1）Quasi-Newton法（BGFS最常用）：不直接求Hessian逆阵，而是用O(n^2)更新秩为1来近似Hessian逆阵。（2）L-BFGS(有限内存BFGS)：不形成或存储所有Hessian逆阵。

