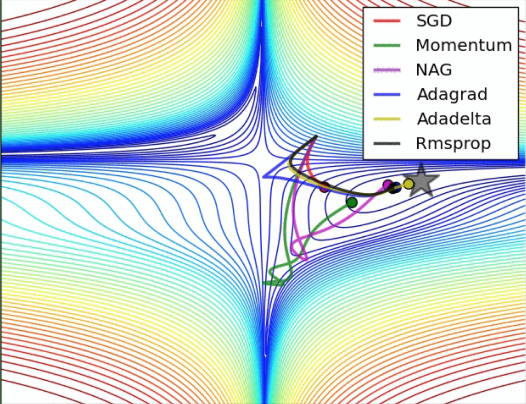
2017.11.27

http://www.bilibili.com/video/av12556478/

1. 什么是机器学习
   1. 监督学习：有数据和标签
   2. 无监督学习：有数据无标签
   3. 半监督学习：结合监督学习和无监督学习
   4. 强化学习：从经验中总结提升
   5. 遗产算法：适者生存

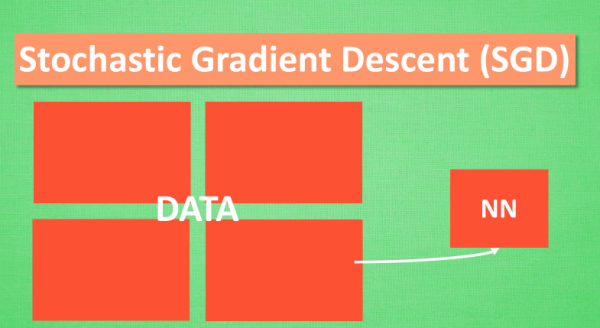
2017.11.28

1、加速网络训练：optimizer



(1)Stochastic Gradient Descent(SGD)

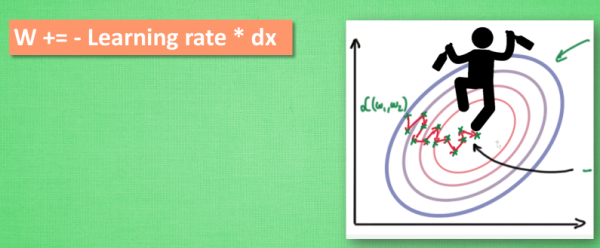




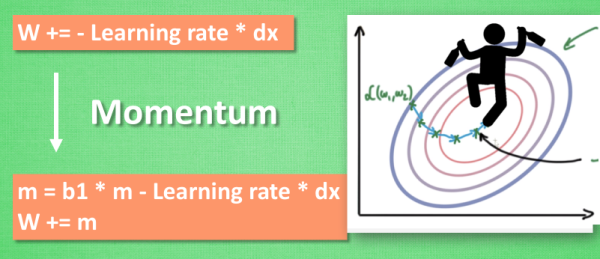
红色方块是我们要训练的 data, 如果用普通的训练方法, 就需要重复不断的把整套数据放入神经网络 NN训练, 这样消耗的计算资源会很大。

如果把这些数据拆分成小批小批的, 然后再分批不断放入 NN 中计算, 这就是我们常说的 SGD 的正确打开方式了。每次使用小批数据, 虽然不能反映整体数据的情况, 不过却很大程度上加速了 NN 的训练过程, 而且也不会丢失太多准确率。

(2)Momentum

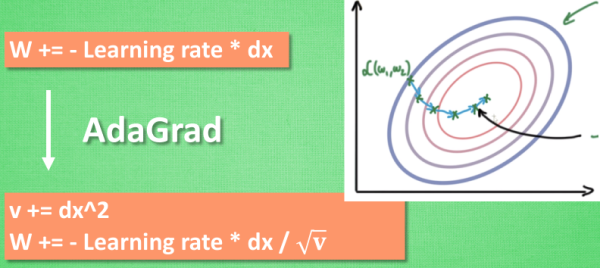


大多数其他途径是在更新神经网络参数那一步上动动手脚. 传统的参数 W 的更新是把原始的 W 累加上一个负的学习率(learning rate) 乘以校正值 (dx). 这种方法可能会让学习过程曲折无比, 看起来像 喝醉的人回家时, 摇摇晃晃走了很多弯路.



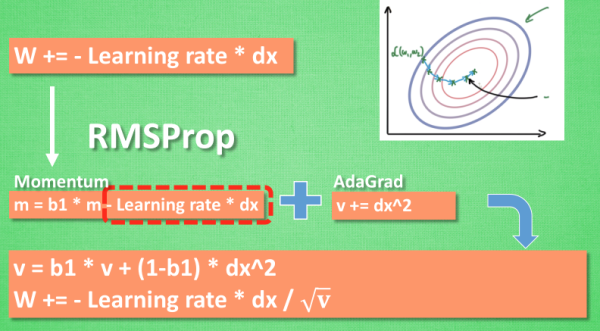
所以我们把这个人从平地上放到了一个斜坡上, 只要他往下坡的方向走一点点, 由于向下的惯性, 他不自觉地就一直往下走, 走的弯路也变少了. 这就是 Momentum 参数更新.

(3)AdaGrad



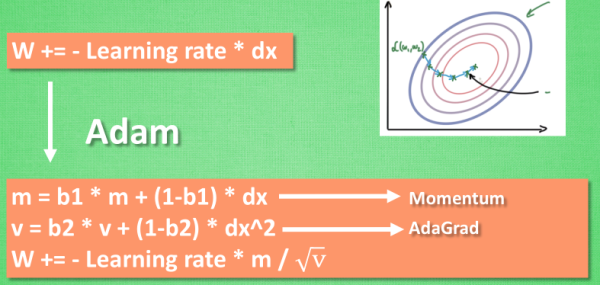
这种方法是在学习率上面动手脚, 使得每一个参数更新都会有自己与众不同的学习率, 他的作用和 momentum 类似, 不过不是给喝醉酒的人安排另一个下坡, 而是给他一双不好走路的鞋子, 使得他一摇晃着走路就脚疼, 鞋子成为了走弯路的阻力, 逼着他往前直着走.

(4)RMSProp



有了 momentum 的惯性原则 , 加上 adagrad 的对错误方向的阻力, 我们就能合并成这样. 让 RMSProp同时具备他们两种方法的优势. 不过细心的同学们肯定看出来了, 似乎在 RMSProp 中少了些什么. 原来是我们还没把 Momentum合并完全, RMSProp 还缺少了 momentum 中的 这一部分. 所以, 我们在 Adam 方法中补上了这种想法.

(5)Adam



计算m 时有 momentum 下坡的属性, 计算 v 时有 adagrad 阻力的属性, 然后再更新参数时 把 m 和 V 都考虑进去. 实验证明, 大多数时候, 使用 adam 都能又快又好的达到目标, 迅速收敛. 所以说, 在加速神经网络训练的时候, 一个下坡, 一双破鞋子, 功不可没.

2、优化器optimizer

莫烦：

<https://morvanzhou.github.io/tutorials/machine-learning/tensorflow/3-4-optimizer/>

tensorflow官方：

<https://www.tensorflow.org/versions/r0.9/api_docs/python/train.html>

tensorflow提供7种优化器：



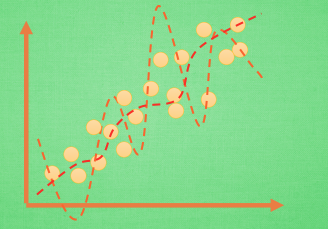
2017.12.11

过拟合

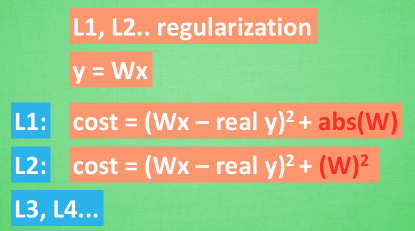
<https://morvanzhou.github.io/tutorials/machine-learning/tensorflow/5-02-A-overfitting/>

解决方法：

方法一: 增加数据量, 大部分过拟合产生的原因是因为数据量太少了. 如果我们有成千上万的数据, 红线也会慢慢被拉直, 变得没那么扭曲 .

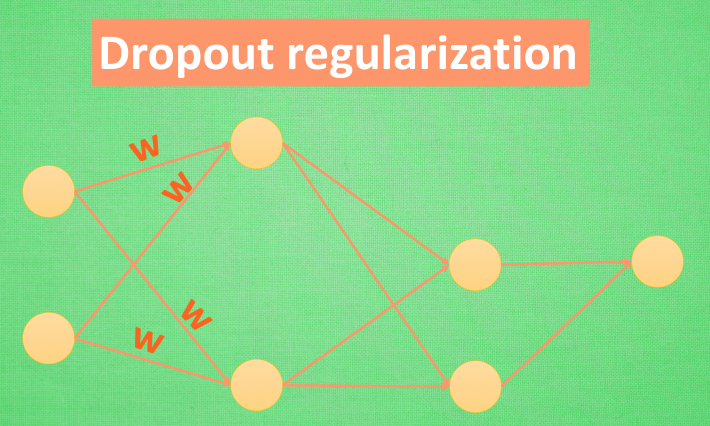


方法二：运用正规化. L1, l2 regularization等等, 这些方法适用于大多数的机器学习, 包括神经网络. 他们的做法大同小异, 我们简化机器学习的关键公式为 y=Wx . W为机器需要学习到的各种参数. 在过拟合中, W 的值往往变化得特别大或特别小. 为了不让W变化太大, 我们在计算误差上做些手脚. 原始的 cost 误差是这样计算, cost = 预测值-真实值的平方. 如果 W 变得太大, 我们就让 cost 也跟着变大, 变成一种惩罚机制. 所以我们把 W 自己考虑进来. 这里 abs 是绝对值. 这一种形式的 正规化, 叫做 l1 正规化. L2 正规化和 l1 类似, 只是绝对值换成了平方. 其他的l3, l4 也都是换成了立方和4次方等等. 形式类似. 用这些方法,我们就能保证让学出来的线条不会过于扭曲.



方法三：专门用在神经网络的正规化的方法, 叫作 dropout。在训练的时候, 我们随机忽略掉一些神经元和神经联结 , 是这个神经网络变得”不完整”. 用一个不完整的神经网络训练一次.

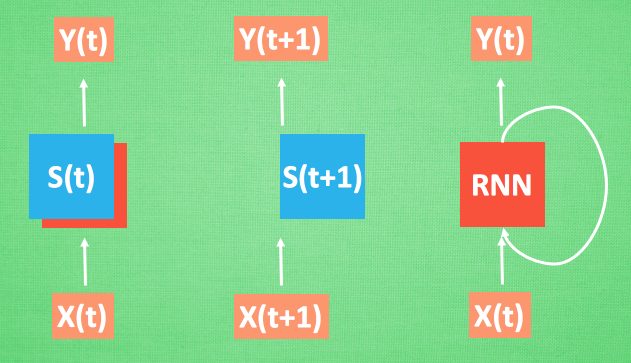
到第二次再随机忽略另一些, 变成另一个不完整的神经网络. 有了这些随机 drop 掉的规则, 我们可以想象其实每次训练的时候, 我们都让每一次预测结果都不会依赖于其中某部分特定的神经元. 像l1, l2正规化一样, 过度依赖的 W , 也就是训练参数的数值会很大, l1, l2会惩罚这些大的 参数. Dropout 的做法是从根本上让神经网络没机会过度依赖.



2017.12.13

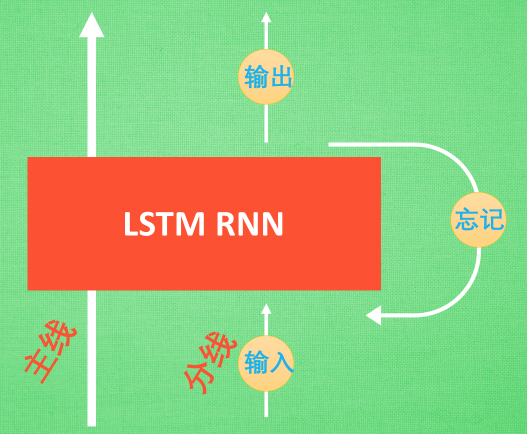
RNN(Recurrent Neural Network)循环神经网络

<https://morvanzhou.github.io/tutorials/machine-learning/tensorflow/5-07-A-RNN/>



LSTM RNN

<https://morvanzhou.github.io/tutorials/machine-learning/tensorflow/5-07-B-LSTM/>



循环神经网络(RNN, Recurrent Neural Networks)介绍

<http://blog.csdn.net/heyongluoyao8/article/details/48636251>

2017.12.18

Batch Normalization

<https://morvanzhou.github.io/tutorials/machine-learning/tensorflow/5-13-A-batch-normalization/>