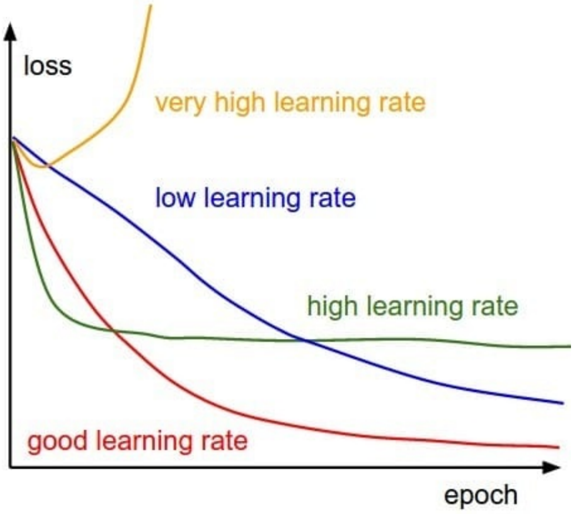
cs231n-1: How to Train a Neuron Network 如何训练神经网络

CS231N第六第七课时的一些笔记，如何训练神经网络是一个比较琐碎的事情，所以整理了一下，以后训练Neuron Network的时候可以看一下

1. 注意区分epoch和iteration，一个是对所有数据集过几遍，一个是迭代多少次
2. Activation Functions
    1. ReLu（good）
        1. ELU
        2. leaky ReLu
        3. no saturated on +region
        4. converges much faster 差不多6倍的速度，因为梯度不会被杀死
        5. easy computation
        6. but half of the data will die
        7. 符合生物神经网络的概念
    2. tanh
        1. saturated -> kill gradient
    3. sigmoid
        1. 不要用这个，because it’s not zero centered
        2. 同时还有饱和、exp难以计算的问题
        3. gradient on w will be all negative or positive
    4. maxout
3. Data Preprocessing
    1. Mean/Normalization
        1. 如果不这么做的话，w轻轻一动就会无法分类
    2. Batch Normalization
        1. 每个layer添加一层normalization
        2. 最后再用y = gamma * xi + beta还原，提高其expressive的能力
4. Weight Initialization
    1. W=0的话，会导致不更新
    2. W很小的话，比较高的层会越来越接近0
    3. W较大的时候，很多的激活函数会saturated，导致梯度为0
    4. 最后要用Xavier initialization，这个挺好用
    5. ReLu的时候会有Half Killed的情况，多个1/2
5. babysitting the network
    1. 开始
        1. 先看loss是不是合理的
        2. 然后加入reg，loss变大
        3. 然后训练小数据集，看一下效果
    2. Learning Rate
        1. 从比较小的reg开始，找到一个能让loss变小的learning rate
        2. 如果loss不怎么变的话，learning rate太小了
        3. 如果NaN或者inf的话，说明learning rate太大了
        4. learning rate一般在1e-3~1e-5之间

        5. 

   3. Hyper Parameters Learning
      1. Grid Search
         1. 随机选择一些，不要间隔很近，特别是对重要的参数，较好的分布很重要
      2. 先随机找到rough range，然后更细致地搜索
      3. Regularization
         1. 注意查看Training accuracy和Validation Accuracy，如果Validation Accuracy不怎么变化，Training Accuracy还在增加，说明过拟合了，可能需要增大
6. 算法
   1. SGD算法
      1. 注意一个decay的问题
         1. 使得接近最优点时速度减慢，for learning rate
         2. decay usually be 0.9 or 0.999
      2. 容易陷入鞍点(saddle)或者local optima
         1. 注意，对于high dimension问题，saddle会更加常见，因为local optima意味着所有方向上都是最优，saddle则意味着部分方向最优
         2. 即使没有被陷入，也会导致梯度很小，使得速度变慢
      3. SGD会在瞎跑
      4. 部分方向 really sensitive
   2. SGD+Momentum(动量)
      1. 增加一个velocity（initialized to be zero）
         1. vx = rho * vx + dx
            1. 相当于是之前的梯度的一个组合了
         2. gradient -= alpha * vx
      2. 相当于，小球从上往下走的时候，有一个动量，就不会在某个小小的local optima，也就是小坑上被停住，它会有一个惯性继续向前跑
   3. Nesters Momentum Gradient Descent
      1. Firstly compute the gradient of (v + grad)

4. AdaGrad (not so common)
    1. 注意遇到局部最优时会很惨
    2. for convex case, it's a good feature to slow it down when you approach the optima
    3. RMSprop (it runs well)
5. **Adam (Stick all above together)**
    1. **Problems come up at the initial steps, 'cause these steps might be really large**
    2. **so we should add a bias correction term**
    3. **We can set like this, and it's a good start point**
        1. **beta1 = 0.9**
        2. **beta2 = 0.99**
        3. **learning_rate = 1e-3 or 5e-4**
6. What's more
    1. 我们倾向于平滑的Minima
    2. 因为直觉上来说，sharp minima通常不是好的最优点，我们往往可以增大数据量消除这种sharp

7. Also we can decay our learning rate
    1. step decay
    2. exponential decay
    3. 1/t decay
    4. Adam貌似通常不用（步长自动减？）
    5. 最好先从no decay开始，再看要不要
8. Second-order optimization
    1. All above are first-order optimization
    2. Try second-order taylor expansion
        1. Newton's method to solve 'gradient = 0'
        2. =>theta_star = theta - Hessian-1(gradient)
        3. in vanilla version of Newton's method, H replace the learning rate, which is used to be a hyper parameter (but actually we still need to add learning rate because the second-order approximation maybe not perfect too)
        4. However, Hessian is time-consuming to compute, not to say invert
    3. alternatively, we can use BGFS / L-BGFS
9. Ensemble Model 聚合模型
    1. Less the gap of training error and test error (validation error)
        1. enjoy 2% extra performance to address the problem of overfitting
        2. hyper parameters usually are not the same
10. Overfitting
    1. 以下都属于Regularization方法
        1. Vanilla Regularization
        2. Dropout
            1. Every time we do a forward pass through the network, at each layer, we randomly set some neurons to zero.
            2. interpretation
                1. Not use too much features to prevent overfitting
                2. ensemble
            3. 注意除以概率P

3. Common pattern
    1. Add some randomness to improve the generalization in training, and then average out randomness in testing
    2. **Batch Normalization (most commonly use and tend to be enough)**
        1. 这里有点难理清它引入的随机性，因为训练时batch normalization是在每个mini batch上做的，所以其规范化更加有随机性，而在test阶段，我们用一个pre-compute的全局均值、方差来做，所以平均化了这种随机性
        2. 有了它，通常可以不用做dropout
    3. Data Augmentation
        1. **把输入做一个随机变换，比如图片，旋转镜像、Color jittering、图像信息处理的那一堆操作**
        2. **增加随机性！！！！**
    4. DropConnect
    5. Fractional Max Pooling
        1. Pooling（池化）随机选择某个区域内的最大值用来表征该区域的feature，主要是方便减小图片大小，更容易控制
        2. Fractional的话，貌似是个随机Pooling，region大小不固定
    6. Stochastic Depth

2. Transfer Learning => 当然，直接增大数据量也可缓解Overfitting
    1. 已学训练好的模型参数迁移到新的模型来帮助新模型训练
    2. 特别的，如果没有很大数据量，那么可以去类似ImageNet之类的地方找个类似的大数据集训练一下