# [对优化算法的一些感想](http://blog.csdn.net/silence1214/article/details/21520217)

2014-03-19 11:25 1003人阅读 [评论](http://blog.csdn.net/silence1214/article/details/21520217#comments)(2) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/silence1214/article/details/21520217#report)

1：今天早上阅读了on optimization methods for deep learning这篇paper，是andrew NG组的博士的一篇paper，这篇paper介绍了L-BFGS，CG（共轭梯度）和SGD算法。

2：个人感受最深的就是，我对CG没用过，用L-BFGS最多，SGD是最想用的，但是往往找不到好的模型参数。L-BFGS当样本过多的时候运行非常慢，调一次参数太慢了，SGD速度倒是很快，但是往往结果的差异性很大，没法找到那组参数。

3：在文中作者给了很多的experiment，发现L-BFGS和CG的效果是最好的，效果好指的有2个方面：达到一样的精度的时候用的时间最少；用的时间一样的多的时候，精度最高。

并且作者给出了实验中用L-BFGS的话，使用mini-batch，也就是类似SGD的训练方法，这样可以让L-BFGS速度也很快，精度也很高，同时这篇paper也给的有源代码，我好好阅读下再来说下怎么做。

4：最近在准备一组数据，看来还是可以用L-BFGS了，只需要参考一下训练的一些strategy就好了。哈哈。。

[对随机梯度下降的一些使用心得](http://blog.csdn.net/silence1214/article/details/22263969)

2014-03-27 09:19 3738人阅读 [评论](http://blog.csdn.net/silence1214/article/details/22263969#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/silence1214/article/details/22263969#report)

1：对于随机梯度下降SGD可能大家都比较了解，也很熟悉，说起来也很简单，在使用中我们一般用的是带mini batch的SGD。这个也描述起来很简单，但是在使用中还是有一些trick才可以的，最近在做一个实验，由于L-BFGS的速度太慢所以选择了带Mini-batch的SGD，我来说下我的心得。

2：其实梯度下降算法，在使用的时候无非是要考虑到2个方面，一个是方向，一个是步长，方向决定你是否走在了优化的道路上还是优化道路的负方向，步长是决定你要走多久才能到最优的地方。对于第一个问题很好解决，就是求梯度，梯度的负方向就是了。难的是求步长，如果步子太小，则需要很长的时间才能走到目的地，如果步子过大可能在目的地的周围来走震荡。所以重点在于如何选择步长。

3：对于随机梯度中，步长的选择方法有很多，最简单的莫过于设置一个比较小的步长，让算法慢慢去运行去就是了，也有别的方法就是可以计算步长的算法，这个我也试过了，反正不好弄，我就选择了最简单的小步长。但是何时算法自己知道差不多了可以停止了呢？我主要想说下这个问题：很多人都说设置迭代一定的次数或者比较两次梯度的变化，或者两次cost的变化，这个地方我不是特别同意，因为尤其是设置了一个小步长的时候，迭代一定次数当然可以，但是这个次数到底多少可以？没办法知道，所以如果设置了一定次数，次数过小的话肯定此时并没有达到最优的或者很接近最优点的地方，如果过大理论上是可以的，但是多少才算过大？你觉得10万次很多，但是不一定10万次算法可以达到，所以这个我觉得不太靠谱，对于比较两次梯度变化或者cost变化，同样存在这个问题，如果步长很小的话，那么同样连续两次之间的梯度和cost变化很小也是无法保证此时接近最优点的啊。

4：这里我介绍一个方法叫做early-stop，其实也是很成熟的方法了，大概思路是在训练的过程中，使用验证集周期性的来测试当前计算出来的参数，在验证集上测试当前参数对验证集的效果，如果效果可以，就保存起来，当很长一段时间都是此效果的话那么就迭代停止，该组参数就认为是不错的参数。这个方法叫做交叉验证，但是我看到有的地方写的是交叉验证用于超参的选择，而这个地方我不是选取的超参，所以不知道到底用对了没有。

5：我在下面贴出来我的整个sgd的matlab的代码，来大概说下这个early-stop。

**[plain]** [view plaincopyprint?](http://blog.csdn.net/silence1214/article/details/22263969)[在CODE上查看代码片](https://code.csdn.net/snippets/259272)

1. function [ optParams ] = SGD( funObj,theta,data,labels,options )
2. % Runs stochastic gradient descent with momentum to optimize the
3. % parameters for the given objective.
4. %
5. % Parameters:
6. %  funObj     -  function handle which accepts as input theta,
7. %                data, labels and returns cost and gradient w.r.t
8. %                to theta.
9. %  theta      -  unrolled parameter vector
10. %  data       -  stores data in m x n x numExamples tensor
11. %  labels     -  corresponding labels in numExamples x 1 vector
12. %  options    -  struct to store specific options for optimization
13. %
14. % Returns:
15. %  opttheta   -  optimized parameter vector
16. %
17. % Options (\* required)
18. %  epochs\*     - number of epochs through data
19. %  alpha\*      - initial learning rate
20. %  minibatch\*  - size of minibatch
21. %  momentum    - momentum constant, defualts to 0.9
22. %% Setup
23. assert(all(isfield(options,{'epochs','alpha','minibatch'})),...
24. 'Some options not defined');
25. if ~isfield(options,'momentum')
26. options.momentum = 0.9;
27. end;
28. epochs = options.epochs;
29. alpha = options.alpha;
30. minibatch = options.minibatch;
31. m = length(labels); % training set size
32. % Setup for momentum
33. mom = 0.5;
34. momIncrease = 20;
35. velocity = zeros(size(theta));
37. %%======================================================================
38. %% SGD loop
39. patience = options.patience;
40. patienceIncreasement = options.patienceIncreasement;
41. improvement = options.improvement;
42. validationHandler = options.validationHandler;
44. bestParams = [];
45. bestValidationLoss = inf;
46. validationFrequency = min(ceil(m/minibatch), patience/2);
47. doneLooping = false;

50. it = 0;
51. e = 0;
52. while (e < epochs) && (~doneLooping)
53. e = e + 1;
55. % randomly permute indices of data for quick minibatch sampling
56. rp = randperm(m);
58. for s=1:minibatch:(m-minibatch+1)
59. it = it + 1;
61. % increase momentum after momIncrease iterations
62. if it == momIncrease
63. mom = options.momentum;
64. end;
66. % get next randomly selected minibatch
67. mb\_data = data(:,rp(s:s+minibatch-1));
68. mb\_labels = labels(rp(s:s+minibatch-1));
70. % evaluate the objective function on the next minibatch
71. [cost grad] = funObj(theta,mb\_data,mb\_labels);
73. % early stop
74. if mod(it, validationFrequency) == 0
75. validationLoss = validationHandler(theta);
76. if validationLoss < bestValidationLoss
77. fprintf('validate=====================================current cost:%f, last cost:%f\n', validationLoss, bestValidationLoss);
78. if validationLoss < bestValidationLoss\*improvement
79. patience = max(patience, it\* patienceIncreasement);
80. bestParams.param = theta;
81. bestParams.loss = validationLoss;
82. end
83. bestValidationLoss = validationLoss;
84. end
85. end
87. if patience < it
88. doneLooping = true;
89. fprintf('stop due to patience[%d] greater than iterate[%d]\n', patience, it);
90. break;
91. end
93. % Instructions: Add in the weighted velocity vector to the
94. % gradient evaluated above scaled by the learning rate.
95. % Then update the current weights theta according to the
96. % sgd update rule
98. %%% YOUR CODE HERE %%%
99. velocity = mom\*velocity + alpha\*grad;
100. theta = theta - velocity;
101. % fprintf('Epoch %d: Cost on iteration %d is %f\n',e,it,cost);
102. end;
104. % aneal learning rate by factor of two after each epoch
105. alpha = alpha/2.0;
107. end;
109. optParams = theta;
111. end

这个代码中包含了对步长的一些侧率，比如momentum了，还有每个epoch之后除以2了，这些可以忽略的，直接看early-stop就好了。我想说明的是对patience的改变，因为当前得到一个更好的结果的话，说明迭代需要时间久点，那么就把patience变大让多迭代点次数，最外层的迭代次数我一般设置的非常大。。

**[plain]** [view plaincopyprint?](http://blog.csdn.net/silence1214/article/details/22263969)[在CODE上查看代码片](https://code.csdn.net/snippets/259272)

1. patience