EE6530 Neural Networks Computer Assignment #2, brought to you by Prof. YWLiu

Printed Report Due April 2 14:20 (in class). No late homework, please.

Task description:

In this homework, your goal is to implement the *stochastic gradient descent* method (SGD, also known as LMS) that is covered extensively in class, by Hinton, and also in Haykin's textbook. You will need to apply it to real data (GradeData_HW2.csv) as well as fake data generated by the function getFakeData().

The real data are basically the same data given to you in HW1, but the .csv file now also includes the total score. The fake data are randomly generated via *principal component analysis* (PCA) to have the same 1st and 2nd-order statistics as the real data.

To be specific, your mission is to find out a set of linear combination weights that best predict the total score out of individual activities (i.e., columns).

Before you proceed to implement SGD, note that the true weights can be directly inferred by solving a pseudo inverse problem in the common least-square sense. This least-square procedure is coded at line 16 of <Hw2_Starter.m>

```
w_ls = inv(inp'*inp)*(inp'*des);
```

So, with this in mind, your goal would be to compare the weights obtained by SGD to the "true" weight `w_ls`. Hopefully the weights given by SGD will converge to `w_ls` *in the stochastic sense*.

Report guideline:

Again, your report should be no longer than 5 pages, including figures and texts. The report should include the following components (中英文不拘):

- A brief statement of the objective of this homework
- Clear description of the methods, using pseudo-codes, equations, or graphic illustration if you will.
- Results with appropriate visualization.
- Analysis of the performance, and discussion on how the performance varies if problems are solved in slightly different ways, or if the problem is defined slightly differently.
- A list of references, if any.

Following are a few issues you might choose to investigate for this homework.

1. How much does your weight vector deviate from the “true” weight `w_ls`?
2. What is the maximal learning rate (`eta`) you can allow before the system becomes unstable? Can this only be found by “trial and error”? Or is there a guideline for the choice of an appropriate learning rate?
3. Is there a trade off between the learning rate and how small the approximation error (`err`) can get?
4. When sampling the real data, does it help to keep drawing the samples randomly?
5. (Continuing from above) If so, does it help to draw the samples repeatedly in the same order?
6. In the for-loop, how would the performance be changed if the observation vector `d` is subject to additive noise? E.g., let `d = d + randn(5.0)` under a predefined standard deviation $\sigma = 5.0$ --- Would SGD fail to converge? Would the performance degrade?