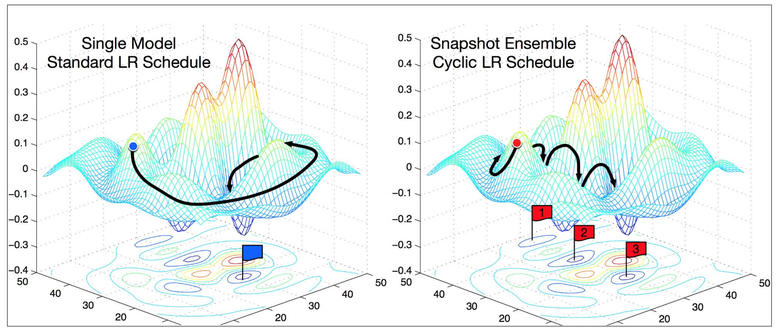
Reproduce and Explore Variations of *SNAPSHOT ENSEMBLES*

Jin Xi, Jiyang Li {jinxi,jiyangli}@stanford.edu

Motivation

SNAPSHOT ENSEMBLES: TRAIN 1, GET *M* FOR FREE is published as a conference paper at ICLR 2017. In this paper, authors proposed a method to assemble multiple neural networks at no additional training cost. They achieved this goal by training a single neural network, converging to several local minima along its optimization path and saving the model parameters.

It can be properly described using the following image([1] Gao & Yixuan et al., 2017):



We want to reproduce their work, to verify if Snapshot Ensemble can provide better prediction, and explore different variations of Snapshot Ensemble.

Datasets

CIFAR-10. (Krizhevsky & Hinton, 2009), Dataset of 50,000 32x32 color training images, labeled over 10 categories, and 10,000 test images.

CIFAR-100. Dataset of 50,000 32x32 color training images, labeled over 100 categories, and 10,000 test images.

Models

We tried both Wide-ResNet (Zagoruyko & Komodakis, 2016) and DenseNet (Huang et al., 2016a).

Our experiments are mostly done with two Wide-ResNet models:

- WRN-16: the original 16-layer Wide-ResNet, with 4 times as many convolutional features per layer as a standard ResNet
- WRN-22: a 22-layer Wide-ResNet model

Methods

We explored different ways to assemble the snapshots:

- Avg: Take average outputs of the M models, and output the category with largest average output. This is used in the original paper.
- Weighted Avg: There are M models, so there will be M outputs for each sample. Take these M outputs as feature, model the output ensemble as linear combination of these M features.

  $$y_{\text{ensemble}} := \sum_{i=1}^{M} w_i y_i$$

  The weights can be learnt from the training set. As avg is a special case of weighted average (all models have same weight), weighted avg is supposed to perform better than avg.
- Locally weighted: This method is inspired by the locally weighted linear regression taught on class. We first define similarity between two samples a and b to be:

  $$\text{sim}(a,b) := e^{-\sum_{i=1}^{M} ||\hat{y}_i(a)-\hat{y}_i(b)||_2^2}$$

  It first calculate the norm 2 distance between prediction outputs, then calculate the similarity from distance.
  Using that similarity function, for each test sample, we can find training samples that similar to that test sample, and take these samples average label:

  $$y_{\text{ensemble}}(x) = \sum_{i=1}^{m_{train}} \text{sim}(training\ sample\ i, x) y_i$$

- We have another idea that didn't show in the Results section: for each model, calculate the Gaussian distribution of all samples where it provides correct prediction; then for each test sample, find the model whose distribution best match the test sample, and use that model's output.

Results

Error rate	CIFAR-10 / WRN-16	CIFAR-10 / WRN-22	CIFAR-100 / WRN-16	CIFAR-100 / WRN-22
Single Model Best	8.53%	8.71%	31.58%	29.82%
Avg	8.39%	**8.64%**	**30.91%**	**29.0%**
Weighted Avg	**8.34%**	8.67%	31.17%	29.2%
Locally weighted	8.42%	8.7%	30.95%	29.03%

For each column, the models are only trained once. The only difference is the way that these models get assembled together.

Discussion

The experiment result shows that average performs better than more complex methods in 3 out of 4 cases.

Weighted average uses training data again to optimize the weights.It's having overfitting problem. The optimized weights tend to have large weights only on a few models that performs slightly better on training data, and give other models very small weights. As a result, the trained weights benefit less for the multiple models that we have.

The locally weighted method we proposed also failed to compete with average. One possible reason is that as these models are generated sequentially, they may correlate to each other. So the similarity function we defined doesn't provide a good estimate on the similarity of different samples.

Future Work

- Run more experiment with different NN architecture and datasets. To verify the observation in this experiment (naive average generally performs best).
- Try other ideas to assemble the snapshot models.
- Explore other strategies to take the snapshots.
- Apply the locally weighted method to assemble models trained by different methods, and evaluate the result.

Reference

[1] Gao Huang∗ , Yixuan Li∗ , Geoff Pleiss, Zhuang Liu, John E. Hopcroft, Kilian Q. Weinberger.

SNAPSHOT ENSEMBLES: TRAIN 1, GET M FOR FREE. In ICLR 2017.

[2] titu1994. Snapshot Ensembles in Keras. https://github.com/titu1994/Snapshot-Ensembles