tsneNet

Transferring neural network knowledge with t-sne regularilizer.

Check README.pdf if Math Equations are not rendered.

Overview

In order to achieve better performance of neural networks in resource limited environments, knowledge transfer from big network to small network is an effective approach. We propose a novel t-SNE regularizer, which can allow the student network to learn the training data structure in the feature space of the teacher network. With the minimal computational cost, it achieves a 50% drop of the difference of error rate between student and teacher networks.

Run the code

1. Install tensorflow on python3
2. python3 tsneNet.py

Introduction

Knowledge transfer from big network to small network

Neural networks have achieved astonishing results in many popular areas. However, behind the great benchmark results, there is still an important factor preventing it from further deployment: the computation cost. Although we can use clusters of GPUs and wait for days for it to finish in the training stage, both energy and time are limited when we deploy them. For example, mobile devices have tiny processors and many real-time jobs require being done within several milliseconds.

The general way to deal with that problem is to use a much smaller network. However, a small network can not fully decipher the complex structure of the training data, and hence usually results in a lower accuracy. We know that, in the human world, self-teaching is much worse compared with learning from others, since the bottleneck in learning is not the brain but the training procedure. Similarly, the low performance of smaller networks should not be blamed on their network structures, but the ineffective training procedure.

Since there are many bigger networks which are much stronger, small networks need not to learn everything by themselves. Similar to the student-teacher relationship in the human world, we can use a teacher (bigger) network to supervise a student (smaller) network, and transfer its knowledge to the student network.

Knowledge hidden in the training data structure in the feature space

The foundation of image classification is the ability to identify similar images. For a CNN based image classification network, this knowledge is hidden in the training data structure in the feature space.

There are two parts in such a network: a feature extractor based on convolution and pooling, and a linear classifier. The feature extractor embeds raw images into a feature space, and the classifier classifies them. Naturally, similar images tend to sit closer in that feature space. Therefore, we can create a similarity matrix based on the structure of the data in the feature space. Each element in that matrix represents the similarity of every pairs of images. The knowledge we want to transfer can be represented by that matrix.

However, the dimension of student network's feature space is usually much lower, so it is necessary to have a measurement of how well a lower-dimensional structure matches to a higher-dimensional structure. Furthermore, this measurement should concentrate on local structure, because it is the local structure that truly shows the similarity relationships.

Knowledge transfer with t-SNE regularizer

The t-SNE algorithm [1] is a widespread solution for data visualization, which can well preserve local structure in dimensionality reduction. That is similar to the dimension mismatch problem we face. The core of t-SNE algorithm includes two matrices and one loss function: matrix P and Q represent the similarities of every two points in the high-dimensional space and low-dimensional space respectively, and loss function C is the Kullback-Leibler divergence of P and Q. By minimizing C with gradient descent method, P and Q become closer, which means the two data structures become more similar.

Following the idea of t-SNE, we design a loss function representing the difference of the data structures in two feature spaces. By simply adding that new loss function to the original supervised learning loss function (typically a cross-entropy loss) and optimizing them together, we can get a much better result. The new loss function (we call it tsne-loss) acts as a regularizer, it can help the student network mimic the data structure of the teacher network, along with the regular supervised training process.

Related work

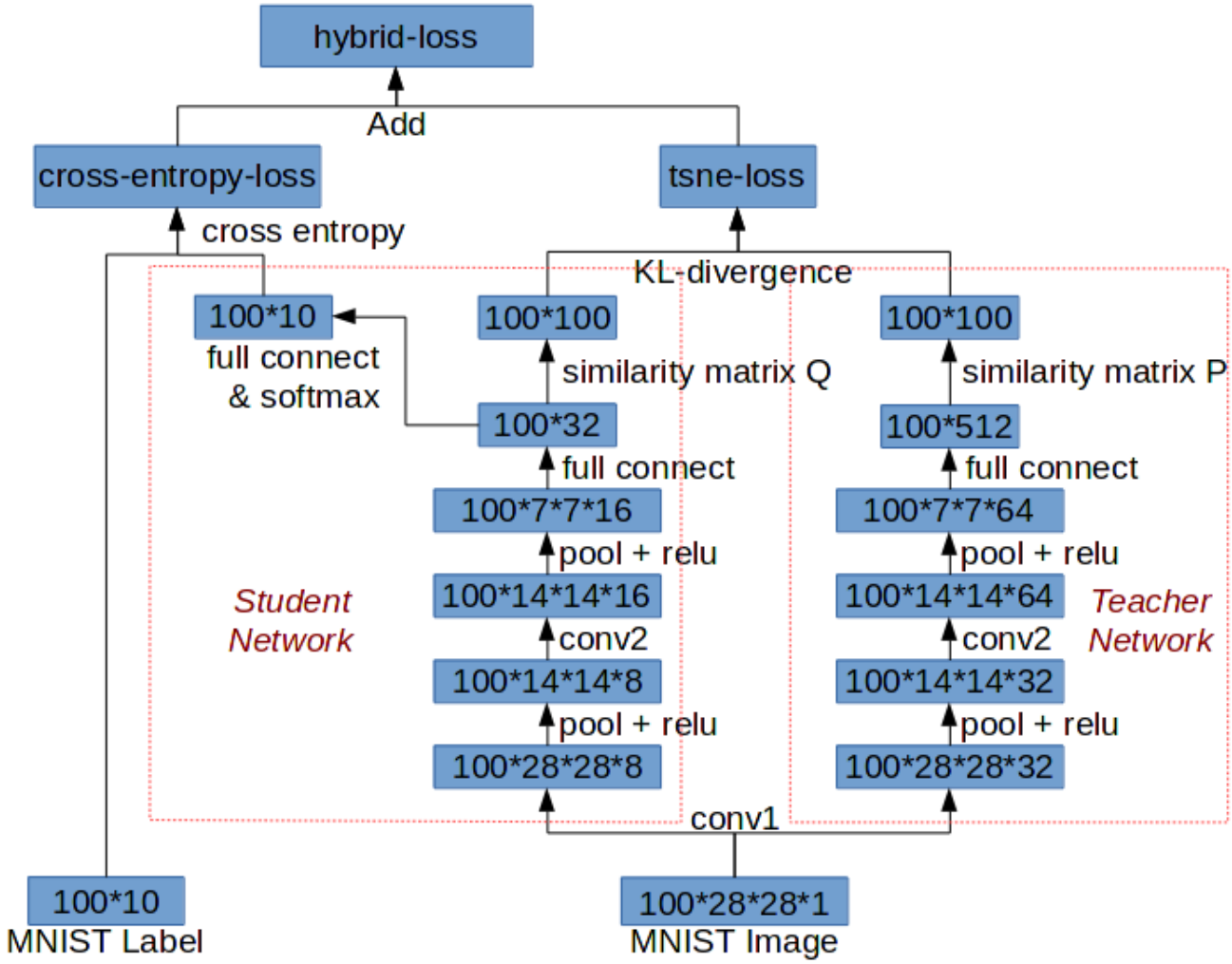
Hinton et al. (2015) [2] shows another way of knowledge transfer between neural networks. Normally, neural networks are trained by human annotated labels. The labels act as one-hot targets, which contain a single 1 and a whole lot of 0's. In that paper, apart from the normal one-hot targets, a new "soft targets" is declared. The soft targets is a softened version of probability distribution predicted by a neural network. It is produced by adding a *temperature* on the softmax layer. Before training a student network, a teacher network is used to produce the soft targets, and then the original one-hot targets are partially replaced by the soft targets (added together with weights). As a result, the student network learns from both the ground truth labels and the soft targets. Soft targets contain knowledge of the teachers network and can increase the performance of the student network greatly.

However, server limits are in that method. Because the knowledge transfer is done by utilizing the final output layer, it can not work when the labels do not match. For example, the teacher network is trained on imageNet, and the goal of the student network is to distinguish different flowers. On the other hand, when the number of labels are too small (for example in a yes-or-no problem), the effect of soft targets are minimized.

We circumvent these restrictions by learning directly from the feature space. The feature space contains a full representation of a image, which is not limited by output labels. Consequently, knowledge transfer between different domains can be much easier. Finally, our method also makes a inspiring expansion of the usage of the t-SNE algorithm.

Implementation

For demonstration, assuming there is a 5-hidden-layer teacher network and a 5-hidden-layer student network, both of them use MNIST dataset as input. Noticed that the size of the student network is much smaller (8-16-32 vs 32-64-512). Batch size is set to 100.



The teacher network is pretrained and its value won't change during the training of the student. Every new input batch goes through all hidden layers in teacher network and produces 100 points in the 512-dimension feature space. Then by calculating each p_{ij}, we get the 100*100 similarity matrix P of those 100 points. Following the same routine, we get the similarity matrix Q, and at last, the tsne-loss. Computation details are listed below.

- Similarity matrix P:

(Check README.pdf if Math Equations are not rendered.)

Let vector x_i denote the position of the ith datapoint in the 512-dimension feature space,

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n},$$

where n is the batch size and

$$p_{j|i} = \begin{cases} \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)} & j \neq i \\ 0 & j = i \end{cases}.$$

Parameter σ_i means the local datapoint density around point i. It is determined by a binary search procedure. Let $H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$ represent the entropy of P_i, we define *perplexity* as $Perp(P_i) = 2^{H(P_i)}$. The final value of σ_i should produce the P_i with a fixed *perplexity* that is specified by the user.

- Similarity matrix Q:

Let vector y_i denote the position of the ith datapoint in the 32-dimension feature space, and α be the dimension of student-t distribution.

$$q_{ij} = \begin{cases} \frac{(1+\|y_i - y_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{k \neq l}(1+\|y_k - y_l\|^2/\alpha)^{-\frac{\alpha+1}{2}}} & j \neq i \\ 0 & j = i \end{cases}.$$

- Tsne-loss:

The tsne-loss is the sum of Kullback-Leibler divergences of P_i and Q_i.

Hence, we have

$$loss_{tsne} = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

Meanwhile, following the normal training process of the student network, we get a normal cross-entropy loss. Here we introduce a new metric, hybrid loss, as follows.

$$loss_{hybrid} = loss_{cross-entropy} + \beta * loss_{tsne}.$$

Then the training step is simply to optimize the hybrid loss, and the student network will be properly trained.

We remark, although the procedure shown above is available, it might be slow. The reason is that running the big teacher network every training step costs a lot. Noticed that the teacher network won't change all the time, and the P matrices won't change either. As a result, simply by caching the P matrices, the only additional computation introduced by the new method is the calculation of matrix Q and KL divergence, which is very little.

Experiment

Training procedure

1. **Pretrain the teacher network**

   Specification of the teacher network is identical to the illustration above:

   - 5*5 conv, depth of 32
   - 2*2 max pooling + relu
   - 5*5 conv, depth of 64
   - 2*2 max pooling + relu
   - full connect, depth of 512, 50% dropout
   - softmax output, depth of 10

   Training data includes 55000 images from MNIST, batch size is set to 50. The network is trained by the Adam Optimizer with 1e-4 learning rate, and for 10000 epochs.

The final test accuracy is 99.0%

2. **Cache a list of matrix P for each training batch**

   Training data for the student network is limited to 6000 images, 10% of the original set. By forwarding the teacher network, we get 6000 512-dimensional points. Then we separate the 6000 into 60 batches, which size is 100, and get a list of 60 P matrices, which size is 100*100. The initial dimension and perplexity are set to 50 and 20 respectively.

3. **Train the student network**

   Specification of the teacher network is identical to the illustration above:

   - 5*5 conv, depth of 8
   - 2*2 max pooling + relu
   - 5*5 conv, depth of 16
   - 2*2 max pooling + relu
   - full connect, depth of 32, 50% dropout
   - softmax output, depth of 10

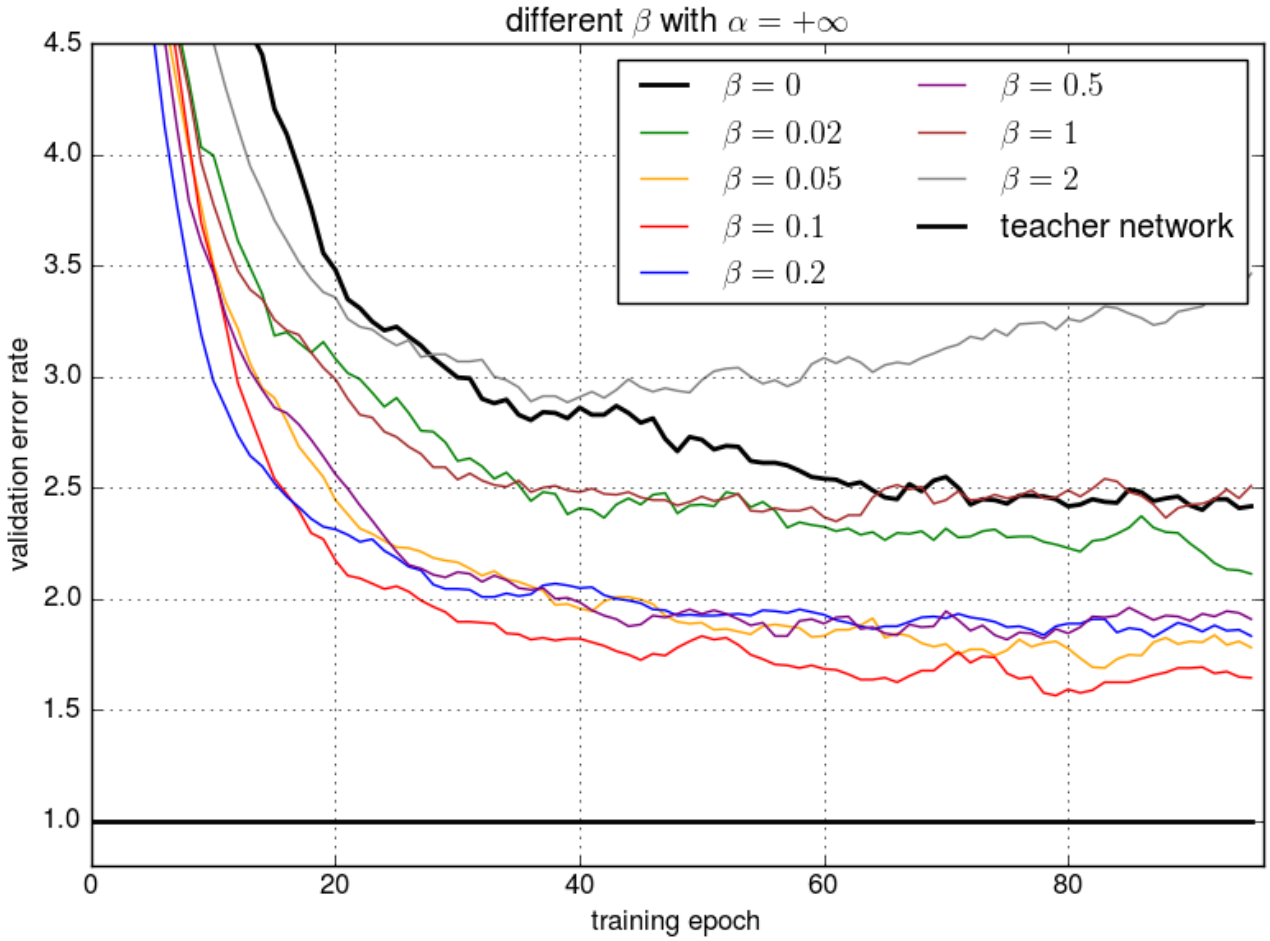
   Training data includes 6000 images from MNIST, batch size is set to 100. The network is trained by the Adam Optimizer with 5e-4 learning rate, and trained for 10000 epochs. Parameters α and β vary in different experiment.

   The only difference from normal training procedure is to replace the original loss function with the hybrid one.

Result and discussion

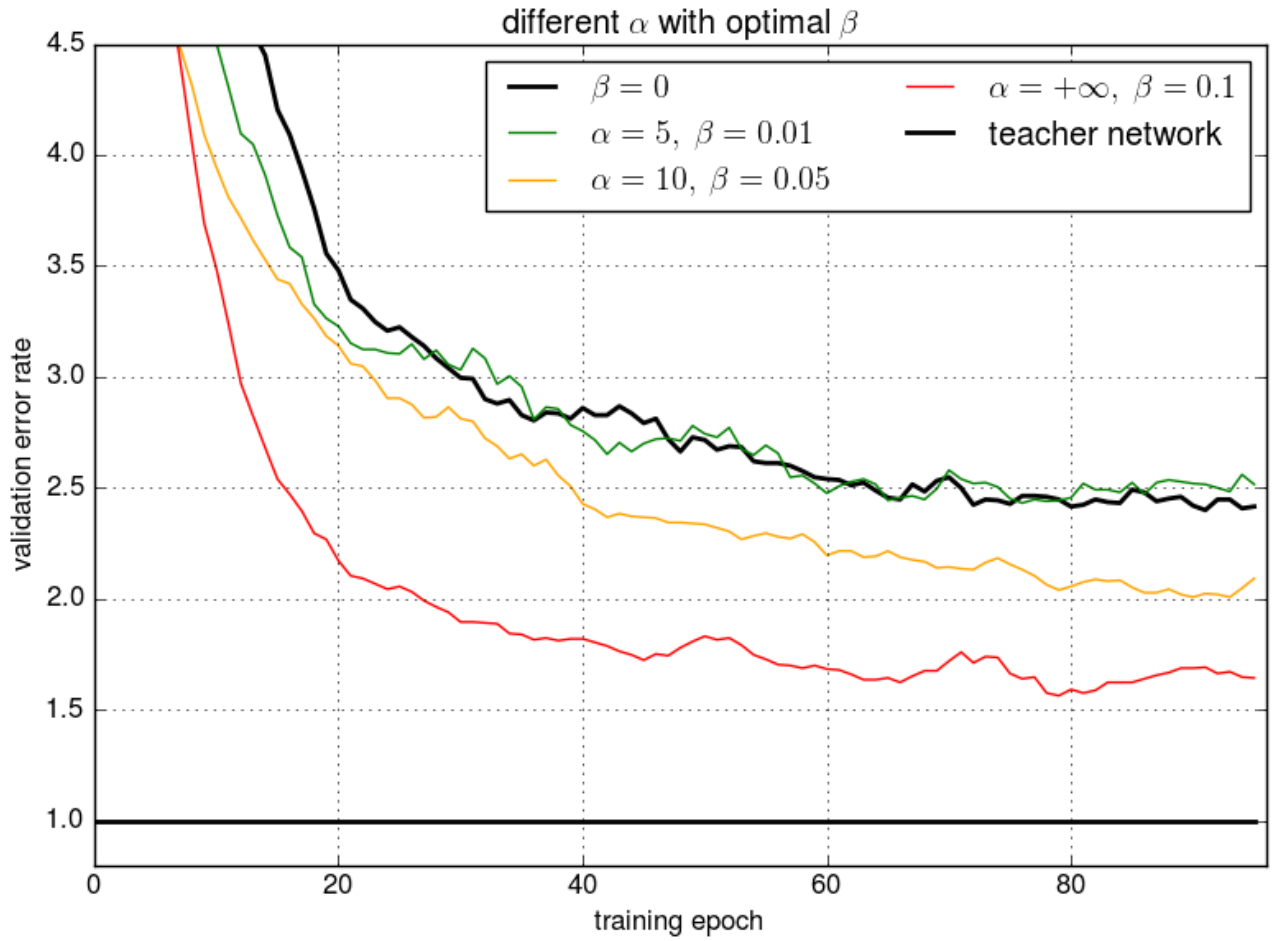
Parameters α and β can affect the training result greatly.

When α is fixed, different β leads to different final error rate.



In the above figure, α is set to $+\infty$, the bold straight line is the error rate of the teacher network, and the bold curve is the error rate of the baseline student network ($\beta = 0$). When β is very small ($\beta = 0.02$), the t-SNE regularization has some effect but not enough. When β is between 0.05 and 0.5, the result is good. Especially when $\beta = 0.1$, the difference of error rate between student and teacher networks drops from 1.4% to 0.7%, achieves a 50% reduce. When β becomes larger, the regularization may overwhelm the normal training. Given the fixed α, we can find the optimal β.

Because the cross-entropy loss and tsne-loss do not converge to the same point, β need to be carefully selected to get the best result. As shown in the experiment, the final error rate changes little when β stays in a relatively large area (0.05 to 0.5), so it is not hard to get the optimal β.



In the above figure, each curve represents the error rate with the optimal β. We further find the optimal error rate will increase when α is set to a lower value. $\alpha = +\infty$ is the optimal choice of α.

The reason of using 2-dimensional student-t distribution in normal t-SNE algorithm is that the long tail can produce a better separation between clusters. While in our scenario, the long tail will only interfere to normal training process. As a result, a higher α always produces a better result.

TODO

- [x] Add documentations
- [x] Update the figures
- [x] Add Related Work
- [ ] Compare different batch sizes
- [ ] Apply on bigger dataset (imageNet)
- [ ] Compare different sizes of student networks
- [ ] Track computation cost / memory reduction with different fixed accuracies
- [ ] Combine / compare with the Hinton's soft target method
- [ ] Try to allow the teacher network enhance itself

References

[1] Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE."*Journal of Machine Learning Research* 9.Nov (2008): 2579-2605.

[2] Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531* (2015).

Thanks

`tsne3.py` is ported from https://lvdmaaten.github.io/tsne/code/tsne_python.zip, adjusted `print()` style to fit python3.