Summaries for "A fast learning algorithm for deep belief nets"

Weituo Hao 109241801

Contribution:

Traditionally, the way to learn a densely connected belief network one layer a time is to assume the higher layers do not exist which is not compatible with the use of simple factorial approximations to replace the intractable posterior distribution. However, the author in this paper assumes the higher layers still exist but they share same weights (the author called it "tied weights"). Then the so-called "complementary priors" is proposed to eliminate the "explaining away" effects that influence the inference in densely connected belief nets so the true posterior distribution is exactly factorial. Then a general method based on this is presented and exceeds the best discriminative learning algorithms

Problem Introduction:

In densely connected networks, the posterior distribution over the hidden variables is hard to address because of the "explaining away" phenomenon. To put it in a simple way, "explaining away" phenomenon illustrates such a situation: when the visible event is on, one of the two hidden causes can explain the observation so sufficiently that the other one can be ignored. Many methods have been proposed to approximate the posterior like MCMC, Variational methods with their corresponding defects.

Method:

Inspired by the logistic belief net composed of binary unites, the author proposed to use extra hidden layers to create "complementary" prior to multiply the likelihood term, and get a posterior that is factorial. That means for a given likelihood function, P(x|y), the corresponding family of complementary priors are those distributions, P(y), for which the joint distribution, P(x,y)=P(x|y)P(y), leads to posterior P(y|x) can be expressed as $\prod_j y_j \,|x$.

Note that not all functional forms of likelihood admit a complementary prior, but it can be demonstrated that the following family constitutes all likelihood functions admitting a complementary prior:

$$P(x|y) = \exp\,(\sum_j \Phi_j(x, y_j) + \beta(x) - log\Omega(y))$$

where Ω is the normalization term. Suppose both P(y)>0 and P(x|y)>0 for every value of y and x. The corresponding family of complementary priors assumes the form:

$$P(y) = \frac{1}{C}\exp\,(log\Omega(y) + \sum_j \alpha_j(y_j))$$

This seemingly mere trick makes use of the same weights for the higher layer and turns the directed models into undirected ones, which results in an efficient learning algorithm.

Since the weights are replicated, the derivatives of the log probability of the data between all pairs of layers will be turned into the Boltzmann machine learning rule as $\frac{\partial logp(v^0)}{\partial w_{ij}} =< v_i^0 h_j^0 > -< v_i^\infty h_j^\infty >$. Then to maximize the log probability is the same problem as minimize the Kullback-Leibler divergence KL($P^0||P_\theta^\infty$).

So far, the whole algorithm can be written as:
1 Learn W_0 assuming all the weight matrices are tied.
2 Fix W_0 and make use of it to infer factorial posterior distributions over the states of the variables in the first hidden layer.
3 Learn RBM model using W_0^T to transform original data with tied weights for higher layer but not tied from W_0

Results:
The performance of a network with three hidden layers and 1.7 million weights on the MNIST set of handwritten digits is evaluated. When no knowledge of geometry is provided and there is no special preprocessing, the generalization performance of the network is 1.25% errors on the 10,000 digit official test set. This beats the 1.5% achieved by the best-back-propagation nets when they are not hand-crafted for this particular application. It is also slightly better than the 1.4% errors reported by Decoste and Schoelkopf for support vector machines one the same task.

Questions:
1 When it comes to learn one layer at a time, the author used contrastive divergence learning instead of Boltzmann machine learning. So the log probability of the data under the full generative model is not guaranteed non-decreasing. Is there any problem with this property?

2 When training the network for MNIST database, the weights were updated after each mini-batch. How on earth were the weights updated in detail?

3 Why the learning rules for picking up unit i are unaffected by the competition between units in a softmax group?

4 What's the point of table 1 in which different tasks are compared? And is this method always superior to SVM?

5 Also the author himself mentioned that it is still unknown why weight-sharing and sub-sampling cannot be used to reduce the error-rate of the generative model.

Summaries for “Reducing the Dimensionality of Data with Neural Networks”
Weituo Hao 109241801

Contribution:
The author describes an effective way of initializing weights of multi-layer neural networks for dimensionality reduction. And it works much better than principal components analysis.

Problem Introduction:
A symmetric neural network cannot only be used to reduce the dimensionality of the data but also reconstruct data from compressed code. But the problem is to properly initialize the weights. Large weights will make algorithm get stuck in local minima and small weights will make algorithm too time-consuming especially for many hidden layers.

Method:
To find a better way to adjust weights, the author made use of the restricted Boltzmann machine to implement a kind of layer-by-layer learning. Given a training image, the binary state h_j of each feature detector j is set to 1 with probability $\sigma(b_j + \sum_i v_i w_{ij})$ where $\sigma(x)$ is the logistic function. Once binary states have been chosen for the hidden units, v_i is set to 1 with probability $\sigma(b_j + \sum_j h_j w_{ij})$ where b_j is the bias of i. The hidden units are then updated once more so that they represent features of the confabulation. The weight is updated by

$$\Delta W_{ij} = \varepsilon(< v_i h_j >_{data} - < v_i h_j >_{recon})$$

where ε is a learning rate, $< v_i h_j >_{data}$ is the fraction of times that the pixel I and feature detector j are on together when the feature detectors are being driven by data, and $< v_i h_j >_{recon}$ is the corresponding fraction for confabulations. The same learning rule is also applied to adjust bias. And the layer-by-layer learning rule is used as many times as desired. So it solves the weights update for network with multiple hidden layers.

Result:
Performance of the method is first compared to PCA on the synthetic data set that contains 20,000 training images and 10,000 testing image. The proposed method worked much better than PCA. Subsequent experiments with different layer autoencoder on MNIST hand-written digits, Olivetti face data set, and documents retrieval turned out this method is superior to PCA. Also, the method exceeded the best previous results on MNIST with an error rate of 1.2%.