Finding a biologically plausible learning algorithm for Deep Belief Nets trained on Natural Images

Joseph Marrama
Department of Symbolic Systems
Stanford University
`jmarrama@stanford.edu`

1 Introduction

Much recent work in the field of Artifical Intelligence has gone into 'deep learning'. 'Deep' models, which are simply models that contian multiple layers of hidden units, offer much potential, because of their recent successes in common AI benchmarks (2), their potential to learn hierarchical representations, and their similairity to the structure of the brain. Deep Belief Nets, a recently introduced class of deep models, have drawn much attention, because of their relative simplicity, easy of training, and strong capacity to learn hierarchical representations. For my project, I investigate different classical PDP methods in their effictiveness at training Deep Belief Nets, both in their capacity to learn and how biologically plausibile they are.

2 Related Work

2.1 Deep Belief Nets

Deep Belief Nets are a fairly simple class of models which consist of multiple hidden layers. Each hidden layer is stacked on top of the other, and each unit in each hidden layer is strictly connected to every other unit in the layer above and below it, and nothing else. There is an input layer which sits below every hidden layer and is fully connected to the first hidden layer. Hinton's 2006 paper (2) proposed a novel learning algorithm to effectively learn the parameters of the network called 'Contrastive Divergence', which is essentailly a simplified version of Gibbs sampling. Using this algorithm, Deep Belief Nets are trained one layer at a time, starting from the first hidden layer going upwards. Deep Belief Nets trained with this algorithm generally have been show to exhibit great performance in many different unsupervised and supervised learning tasks, including image processing.

While Contrastive Divergence succeeds at extracting useful features from the input data, it is still limited to training one new hidden layer at a time. While this isn't a serious limitation of the algorithm, it does limit its biological plausibility, because the brain likely doesn't train one layer of neurons at a time.

2.2 Natural Image Statistics and Sparsity

Natural Image Statistics is another popular focus in AI which is concerned with effectively processing natural images. Much like Deep Belief Nets, algorithms commonly used in Natural Image Statistics, such as Independent Components Analysis (ICA), have been shown to be very effective at extracting meaningful features from visual input (1). ICA and other related algorithms also have another link with the brain: both the V1 area of the brain and ICA have been shown to learn 'sparse' feature detectors. 'Sparse' feature detectors are simply feature detectors that are rarely activated, but whey they are activated they are usually strongly activated. The 'ideal' set of feature detectors in early visual processing are likely sparse, because sparse features provide an optimal way of en-

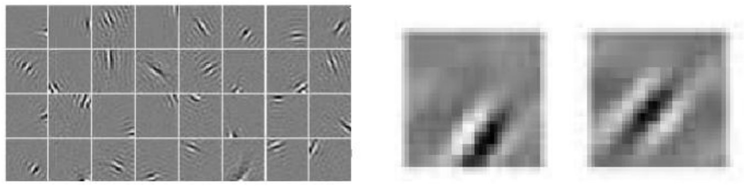
Figure 1: (Left) Maximally sparse feature detectors generated by training on Natural Images, (right) low-level feature detectors in monkey's V1 areas (Hyvarinen 2009)

Figure 2: 2 of the natural images that the training data was sampled from

coding the most amount of information in a small amount of non-zero feature activations. From a biological standpoint, this is also optimal because the neurons that compromise each feature don't have to fire more than necessary.

The weights for sparse feature detectors are characterized by a likeness to the Gabor function, which is a localized sinusoidal grating (see figure 1). The visualizations of sparse features provide a good benchmark for evaluating the performance of learning algorithms on DBNs, because the bottom layer weights can also be visualized as feature detectors. An ideal algorithm for learning a DBN would produce sparse feature detectors at the bottom layers.

3 Methods

3.1 Training Data

For all experiements, I trained a standard Deep Belief net on 10,000 20-by-20 image patches randomly sampled from 13 greyscale natural images (see figure 2). After randomly sampling the image patches, I subtracted the mean from each image patch so that each patch has a zero mean.

3.2 Training Method

For all experiments unless otherwise stated, I trained a standard DBN with 3 hidden layers. The DBN's input layer has 400 units, the first hidden layer has 500 units, the second has 300 units, and the third has 200. I typically trained for 60 epochs (that is, 60 passes through the whole training set). Many times the learning process converged before 60 epochs. To promote faster learning, I used simulated annealing on the learning rate, decreasing it from 0.05 to 0.005 gradually over multiple epochs. I also used a momentum of 0.5, and randomly initialized the weights by sampling each weight from a normal distribution with zero mean and a variance 0.01. I also used a batch size of 10 training examples to speed up learning. Setting the batch size too high typically worsened the quality of the solution and sometimes led to divergence.

4 Learning Algorithms and Results

4.1 Backpropagation

Backpropagation is a classical method for training neural networks. Put simply, it updates weights in a network according to the gradient of the error function with respect to each weight (4). Backpropagation is typically used in a supervised setting, because it requires both input values and target

Table 1: Reconstruction errors with different learning algorithms

Learning algorithm used	Reconstruction error
Backpropagation over all layers	6257
Greedy Layer-by-Layer Backprop	391
Greedy Backprop then Backprop over all layers	414
Greedy Contrastive Hebbian Learning	402
Generalized Recirculation over all layers	1327

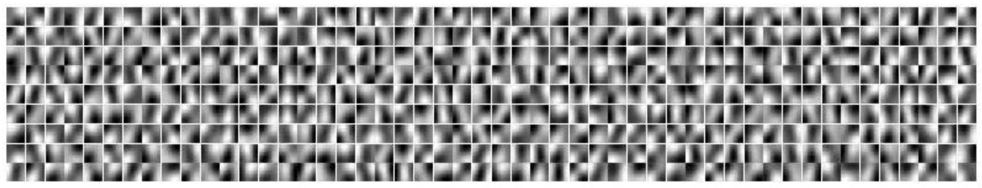

Figure 3: weights learned from doing Backpropagation over all hidden layers

activation values. However, DBNs are trained in an unsupervised fashion without target activations, and there is no well-defined way to set target activation values with natural image patches. To get around this problem, I simply used the input values as their corresponding target activation values. This essentially requires that backpropagation (and the subsequent learning algorithms I tried) will learn to accurately reconstruct the input by minimizing the reconstruction error. This is the same measure of progress that Contrastive Divergence uses. To train the DBN with Backprop, I simply ran the input up to the top hidden layer, then ran the computed top-layer hidden activations back down to the input layer to reconstruct the inputs, computed the error between the input and reconstructed input, and then finally backpropagated the error back up the network in order to do gradient descent on the weights.

The first strategy I tried was to use a standard DBN with the sigmoid activation function, as its common practice with DBNs. However, backpropagation quickly diverged from a solution and learned infinite weights, without fail. Stepping through training on each batch reveals that the first layer weights quickly increase in magnitude, much more than the top-layer weights. This is because the sigmoid function must dilute the gradient as error is backpropagated up the network, so that the top-layer weights cannot keep up with the learning of the bottom layer weights, and the bottom weights try and over-compensate for this, which leads to an even bigger error and weight change the next cycle, and so forth. Before the end of the first training epoch, the first layer weights diverge to either negative infinity or infinity, and the other higher-level weights quickly follow suit.

There is a simple solution to this problem, which is to use the linear activation function, where the activation of each unit a_j is simply $\sum_i w_{ij} a_i + b_j$. This simplifies the learning process and makes it much easier for error to backpropagate up the network, since it is no longer diluted. Using the linear activation function for backpropagation (and all other subsequent experiments), the learning process was able to reach a stable solution.

Backpropagation over the whole network reaches a solution that still has high reconstruction error, yet learns an interesting, spatially-coherent set of feature detectors (figure 3). As you can see in table 1, the reconstruction error remains relatively high ($\approx$ 6200). Even without ‘dampening’ of the back-propagated error due to the sinusoidal activation function, backpropagation tends to easily get stuck in local minima in the error function, which would explain the high reconstruction error. However, the bottom-level feature detectors seem to detect activity in specific, localized regions of each image patch, which could indicate that backprop still learns useful features from the data.

4.2 Greedy Layer-by-layer Backpropagation

The next strategy I tried was using backpropagation to train one layer at a time moving up from the input, just like Contrastive Divergence. Although it is less ‘biologically plausible’ that learning would proceed in this fashion, my rationale was that it would help minimize the reconstruction

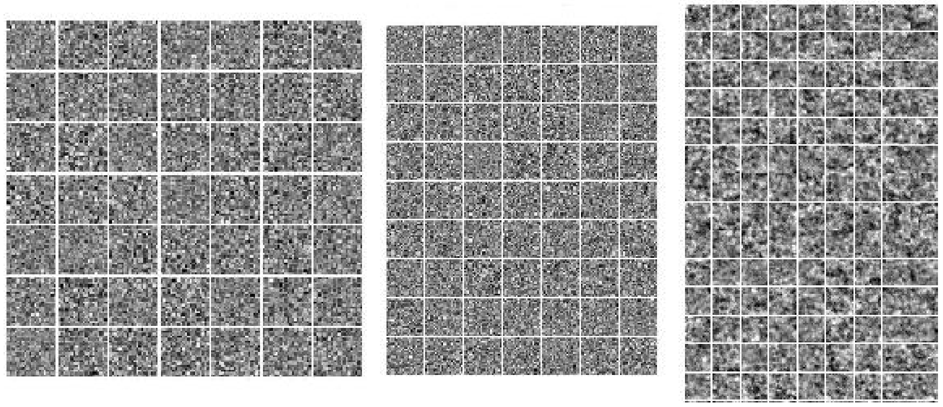

Figure 4: (From left to right) Weights learned by greedy Backpropagation, Greedy CHL, and Generalized Recirculation over all layers. Note that weights from Greedy Backprop then All-layer Backprop are identical, and that weights from GenRec are slightly more spatially coherent

error by learning better features at each error. As in table 1, this strategy actually achieves the best reconstruction error over the full DBN, which is a noteworthy feat. This is because in training each layer seperately, backprop actually succeeds in finding a good local minimum in the error function, which helps avoid the numerous other local minima you might run into trying to train the whole network at once. However, the feature detectors learned are very spatially incoherent (see figure 5). This likely isn't due to overfitting, either. Training the bottom layer on 50,000 20-by-20 patches still results in nearly identical feature detectors.

4.3 Greedy Backpropagation then Backpropagation

In an effort to combine the successes of backprop and greedy backprop, I next ran an experiement where I trained the DBN with greedy layer-by-layer backprop and then ran backprop over the whole network. However, this approach largely failed. After running greedy backprop, running backprop over every layer barely changed the connection weights. In fact, this stage often worsened the overall reconstruction error. As in figure 5, the first layer weights are almost identical to those learned by just running greedy backpropagation.

4.4 Contrastive Hebbian Learning

For my next experiment, I decided to train a DBN with Contrastive Hebbian Learning (CHL) instead of backprop. The weight update rule for CHL is:

$$\frac{1}{\epsilon}\Delta w_{ij} \quad = \quad (a_i^+ a_j^+) - (a_i^- a_j^-)$$

The '+' and '-' superscripts refer to the activation of each unit in the positive phase and the negative phase, which are simply the activations from running the input and reconstructed inputs through the network, respectively. This update rule is more biologically plausible in the fact that it only uses local activations to compute the gradient update for each weight.

This algorithm quickly diverges when using it to train a full DBN all at once. The weights quickly diverge to an infinite magnitude after less than one epoch. However, CHL works when training the network one layer at a time. Training in a greedy fashion, CHL typically gets the same results as greedy backpropagation, with similair spatially-incoherent feature detectors learned and an overall reconstruction error of about 400.

4.5 Generalized Recirculation

As given in O'Reilly's 1996 paper (3), the Generalized Recirculation (GenRec) learning rule is:

$$\frac{1}{\epsilon}\Delta w_{ij} \quad = \quad a_i^- (a_j^+ - a_j^-)$$

This algorithm is also more biologically plausible, since it doesn't use backpropagated error. Also, this algorithm worked well for training a full DBN all at once. It succeeded in minimizing the reconstructed error to be about 1300, which is over 4 times smaller than that obtained by using backpropagation over every layer (table 1). However, the weights obtained by this algorithm aren't particularly spatially coherent, although they do show somewhat of an improvement over CHL and greedy backprop.

5 Discussion

5.1 Evaluating Measures of Performance

After running several classical PDP algorithms to train Deep Belief Nets, one thing becomes immediately apparent - there is no one best objective measure of the quality of the features learned. The ideal learning algorithm would be 'biologically plausible', minimize error on the training set, produce sparse feature detectors, and exhibit good classification performance when used in conjunction with a supervised learning algorithm. But when there isn't a clear winner, how do we rank learning algorithms?

5.1.1 Minimizing reconstruction error

Minimizing the reconstructed error of the training set may be a good measure of error to start. Reconstruction error provides a crude measure of how well the features learned capture and compress the information in the training data. However, its usefulness may be limited. If a Deep Belief Net perfectly recreated the training data with zero error, then this may be a sign of overfitting the training data more than anything else. In training DBNs on natural images, we are looking for features that will generalize over all natural images well, and not just fit a small subset perfectly. Contrastive Divergence doesn't exhibit particularly good performance on reconstructing the training data (2), nor does ICA (1).

5.1.2 'Biological Plausability'

'Biological Plausability' is a fairly ill-defined term in the contexts of PDP, Natural Image Statistics, and this project, and I'd like to shed some light on the term and using it as a measure of learning algorithms. Essentially, the term has different meanings when used in a statistical context and a PDP context. Looking through the lense of Natural Image Statistics, saying an algorithm is biologically plausible generally means that it learns similar receptive fields as V1 in the brain - that is, sparse, gabor-filter-like receptive fields. As discussed above, this makes sense because it allows neurons to fire less often and captures the most information.

In PDP, the 'biological plausability' of a learning algorithm generally describes how realistically one could implement the weight-update rule in actual neurons. For example, GenRec is more biologically plausable than Backprop (in a PDP context), because GenRec only uses localized activation differences to compute the weight-update rule, whereas Backprop requires sending error signals all the way through the network. In general, statistical biological plausability is concerned with a higher level of analysis: the output and features learned in V1, whereas PDP biological plausability is concerned with a lower level: how realistically the update rule could be implemented in the brain.

These two definitions often find each other at odds, because rarely is something 'biologically plausible' in both contexts. As I proved with my experiments, the PDP biologically plausible algorithms GenRec and CHL don't learn sparse features. Statistically biological plausible algorithms, such as ICA, aren't very PDP-biologically plausible either, because they use complicated learning rules that involve many large complicated trigonometric and matrix methods. Its hard to say which measure that one should weight more. The approach taken by Natural Image Statistics is certainly more rigorous, although the ideal learning algoirithm would be plausible in both different levels of analysis.

5.1.3 Classification Performance

A common measure of performance of Deep Belief Nets has been classification performance. In a typical supervised learning setting, the training data and training labels are fed directly into a supervised learning algorithm. However, DBNs have had success on improving supervised learning by providing a better set of features for the supervised learning algorithm to use. Training a DBN on the input data and then using the resulting top-layer activations as input will typically perform better than not using a DBN. The classification performance provides another measure of how ‘good’ the features learned by the DBN are, because features that capture and condense information more effectively are likely to exhibit better classification performance.

5.2 Extensions

There are many, many different directions to extend this project in. A valuable first step would be to include classification performance as a metric. The MNIST handwritten digit dataset is a standard in AI, although there are plenty of other options for simpler classification tests. Unfortunately I ran into numerous complications using MNIST and couldn’t get results in time.

A harder but possibly more fruitful approach could be to adopt that of Natural Image Statistics, which would be to start off modeling visual-processing as a constrained optimization problem, and try and forumulate a learning algorithm which would be ‘biologically plausible’ in both statistical and PDP contexts. Formulating a learning algorithm which approximately optimizes sparsity and also uses fairly simple weight update rules would certainly not be trivial, but it would be worth exploring nonetheless.

6 Conclusion

In my project, I investigated the effectiveness of different classical PDP learning algorithms in training Deep Belief Nets. While no one algorithm did particularly well, Backpropagation had limited success. The successes and failures of my experiments offer a window into the different types of approaches towards modeling low-level vision and learning features from natural images in a DBN. I hope to continue this project so I can both figure out more effective methods of evaluating the quality of learning algorithms and formulate new learning algorithms.

7 References

[1] Hyvarinen, Aapo., Hurri, Jarmo & Hoyer, Patrik (2009). Natural Image Statistics. www.naturalimagestatistics.net

[2] Hinton, G. E., Osindero, S. & Teh, Y. (2006) A fast learning algorithm for deep belief nets. Neural Computation, 18, pp 1527-1554.

[3] O’Reilly, R.C. (1996). Biologically Plausible Error-driven Learning using Local Activation Differences: The Generalized Recirculation Algorithm. *Neural Computation*, 8, 895-938

[4] McClelland, James L. (2011) Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises