Layerwise Pre-training with Autoencoders

Vanessa Jurtz

June 17, 2014
Motivation

Exploring Strategies for Training Deep Neural Networks

Hugo Larochelle
Yoshua Bengio
Jérôme Louradour
Pascal Lamblin

Département d’informatique et de recherche opérationnelle
Université de Montréal
2920, chemin de la Tour
Montréal, Québec, Canada, H3T 1J8

Extracting and Composing Robust Features with Denoising Autoencoders

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, Pierre-Antoine Manzagol
Dept. IRO, Université de Montréal
C.P. 6128, Montreal, Qc, H3C 3J7, Canada
http://www.iro.umontreal.ca/~lisa

Contents

What is an autoencoder?
  ▶ Introduction
  ▶ Possible problems with autoencoders
  ▶ Tied Weights
  ▶ Denoising autoencoders

Some examples of deep learning
  ▶ Peptide-MHC training
  ▶ MNIST dataset training
What is an autoencoder?

Neural network that is trained to reproduce its input in the output layer.

Figure adapted from Larochelle et al. *Exploring strategies for training Deep Neural Networks* Journal of Machine Learning Research 2009
What is an autoencoder?

Greedy unsupervised, layer wise pretraining $\Rightarrow$ stack the autoencoders to initialize weights in deep net

Figure adapted from Larochelle et al. *Exploring strategies for training Deep Neural Networks* Journal of Machine Learning Research 2009
What is an autoencoder?

Idea: the layers of deep nets should separate the factors of variation, they should be high-level feature representations of the input

⇒ pre-training with autoencoders should initialize the weights closer to good solutions.

Figure adapted from Larochelle et al. *Exploring strategies for training Deep Neural Networks* Journal of Machine Learning Research 2009
Possible problems with autoencoders

They can learn an uninteresting identity function!

Figure adapted from Larochelle et al. *Exploring strategies for training Deep Neural Networks* Journal of Machine Learning Research 2009
Tied weights

Avoid learning the identity function for continuous input by setting: $W^T = W^*$

Motivation: $W^T$ and $W^*$ tend to be similar after training

Figure adapted from Larochelle et al. *Exploring strategies for training Deep Neural Networks* Journal of Machine Learning Research 2009
Denoising autoencoders

- corrupt part of the input of the autoencoder by setting values to 0
- this simulates removal of the neurons
- train the autoencoder to reproduce the original input from the corrupted version

Figure adapted from Vincent et al. *Extracting and Composing robust features with Denoising Autoencoders* Technical Report 1316 2008
Some examples of deep learning

Peptide MHC-ClassI binding:
- Dataset and protein sequence encoding
- Different training strategies
- Noise levels in DA
- Cost functions
- Pretraining rounds

MNIST dataset
- Introduction to the dataset
- Examples of training
Dataset

MHC Class I binding data

IC 50 binding values are log transformed using $1 - \frac{\log(\text{affinity})}{\log(50000)}$

Figure by Eric AJ Reits
Protein sequence encoding

Each amino acid is encoded by a sequence of 20 numbers. One of them is 0.9 and the position of 0.9 indicates which amino acid is encoded. The other numbers are all 0.05.

Example:

A ⇒ 0.9 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
Peptide and receptor sequence, sparse encoding, 4 hidden layers 20 neurons in each
Peptide training - different noise levels

Peptide + receptor sequence, sparse encoding, 4 hidden layers 20 neurons in each
Peptide - MHC training

MSE cost function:

\[ E = \sum_k (t_k - O_k)^2 \Rightarrow \delta_{kO} = (O_k - t_k) \cdot O_k (1 - O_k) \]

CE cost function:

\[ E = -\sum_k [t_k \ln(O_k) + (1 - t_k) \ln(1 - O_k)] \Rightarrow \delta_{kO} = (O_k - t_k) \]
Peptide - MHC training

Denoising Autoencoder MSE, sparse encoded peptide + receptor

Training PCC vs iterations

Training Error vs iterations

Test PCC vs iterations

Test Error vs iterations

Graphs showing the performance metrics (PCC and error) over iterations for different configurations of the training process.
Peptide - MHC training

Denoising Autoencoder CE, sparse encoded peptide + receptor
Peptide - MHC training

Denoising autoencoder CE, sparse encoded peptide + receptor 50,20,10,10
Peptide - MHC training

DA CE, sparse encoded peptide + receptor 50,20,10,10 e=0.0005

- Training PCC
- Test PCC
- Training Error
- Test Error

Graphs showing iterative training and testing data with 20 cycles and 40 cycles of pretraining.
Conclusions:

- Training deep networks with autoencoder pre-training is possible
- Denoising autoencoders work better than tied weights for sparse encoded peptid-receptor data
- Training parameters need to be further optimized
MNIST Dataset

Handwritten digits (written by different people)

- 10 000 training examples
- 5 000 validation examples (for early stopping)
- 50 000 test examples
MNIST Dataset training

We used 500 hidden neurons in each hidden layer, a learning rate of 0.005 and 20 rounds of pre-training.

<table>
<thead>
<tr>
<th>Network</th>
<th>Depth</th>
<th>Test performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural network (random... + fine-tuning)</td>
<td>1</td>
<td>4.332</td>
</tr>
<tr>
<td>2</td>
<td>4.03 % ± 0.17</td>
<td>10.63 % ± 0.27</td>
</tr>
<tr>
<td>3</td>
<td>4.24 % ± 0.18</td>
<td>11.98 % ± 0.28</td>
</tr>
<tr>
<td>4</td>
<td>4.47 % ± 0.18</td>
<td>11.73 % ± 0.29</td>
</tr>
<tr>
<td>SAA network (autoassociator learning + fine-tuning)</td>
<td>1</td>
<td>7.428</td>
</tr>
<tr>
<td>2</td>
<td>3.87 % ± 0.17</td>
<td>11.43 % ± 0.28</td>
</tr>
<tr>
<td>3</td>
<td>3.38 % ± 0.16</td>
<td>9.88 % ± 0.26</td>
</tr>
<tr>
<td>4</td>
<td>3.37 % ± 0.16</td>
<td>9.22 % ± 0.25</td>
</tr>
<tr>
<td>3.39 % ± 0.16</td>
<td>9.20 % ± 0.25</td>
<td></td>
</tr>
<tr>
<td>SRBM network (CD-1 learning + fine-tuning)</td>
<td>1</td>
<td>4.588</td>
</tr>
<tr>
<td>2</td>
<td>3.17 % ± 0.15</td>
<td>10.47 % ± 0.27</td>
</tr>
<tr>
<td>3</td>
<td>2.74 % ± 0.14</td>
<td>9.54 % ± 0.26</td>
</tr>
<tr>
<td>4</td>
<td>2.71 % ± 0.14</td>
<td>8.80 % ± 0.25</td>
</tr>
<tr>
<td>2.72 % ± 0.14</td>
<td>8.83 % ± 0.24</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Classification performance on MNIST-small and MNIST-rotation of different networks for different strategies to initialize parameters, and different depths (number of layers).
MNIST Dataset training

- 0s in MNIST data replaced by 0.1
- learning rate: 0.005
- pretraining: 100 rounds + early stopping
- fine-tuning: 300 rounds + early stopping

![Graphs showing prediction error (%) for 100 hidden neurons and 100,100,100 hidden neurons for different methods (BP, TW, DA) with MSE and CE.]
MNIST Dataset training

Weight images
all weights in 1st hidden layer leading to one hidden neuron

**top:** BP 1 hidden layer 100 hidden neurons
**bottom:** DA 1 hidden layer 100 hidden neurons
MNIST Dataset training

Backpropagation, 1 hidden layer with 100 hidden neurons, MSE cost function
MNIST Dataset training

Tied weights, 1 hidden layer with 100 hidden neurons, CE cost function
MNIST Dataset training

Denoising autoencoder, 1 hidden layer with 100 hidden neurons, CE cost function
Conclusions

Layerwise pretraining with autoencoders was implemented and makes training deep nets possible.

Training procedure must be refined to improve performance:

- decreasing learning rate
- automized detection of training success
- regularization and weight decay
Acknowledgements

Morten Nielsen
Henrike Zschach and Pascal Timshel
PISP group