Multi-layer auto-encoders

a CS221 artificial intelligence project

Charles Celerier
Computer Science
Stanford
cceleri@cs.stanford.edu

Bill Chickering
Computer Science
Stanford
chickering@cs.stanford.edu

August 18, 2013

This is a short introduction.

1 Introduction

A fundamental challenge to supervised machine learning is that of feature selection. There exist many strategies for determining a set of variables derived from an original representation to accurately classify domain data. Ideally, the selected variables minimize redundant and/or irrelevant information and thereby reduce the risk of overfitting. Historically, domain experts and computer scientists have laboriously handcrafted feature sets for a variety of machine learning problems. These manually engineered features suffer from the requirements of domain expertise and a great deal of work, and therefore, might not scale well. Moreover, such feature sets can inadvertently ignore important structure in the data. These deficiencies can be addressed via unsupervised feature learning using dimension reduction techniques such as principal component anaysis (PCA) or factor analysis. Recently it was found that K-means clustering offers another straightforward method for determining an efficacious feature space (e.g. the CS221 Visual Cortex programming assigment). In this report we document a project in which unsupervised feature learning is performed using autoencoders consisting of multilayer feedforward neural networks.

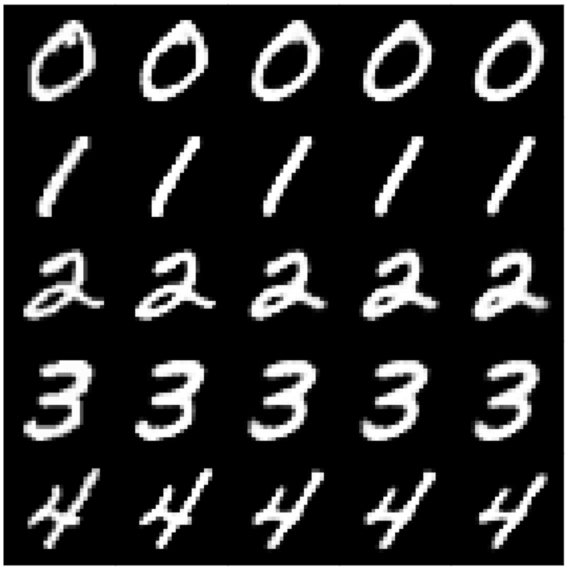
An autoencoder is a special type of neural network consisting of an input layer, an output layer, and one or more hidden layers. An autoencoder aims to approximate the identity function as closely as possible, reproducing the signals of its input layer at its output layer. By placing constraints on the network, input signals must be compressed and decompressed using optimized encodings in order to approximately reproduce the input. Such encodings can then serve as feature sets in a supervis ed learning problem. The challenge is training such autoencoders in order to discover the optimal e ncoding for a particular domain space.

2 Method

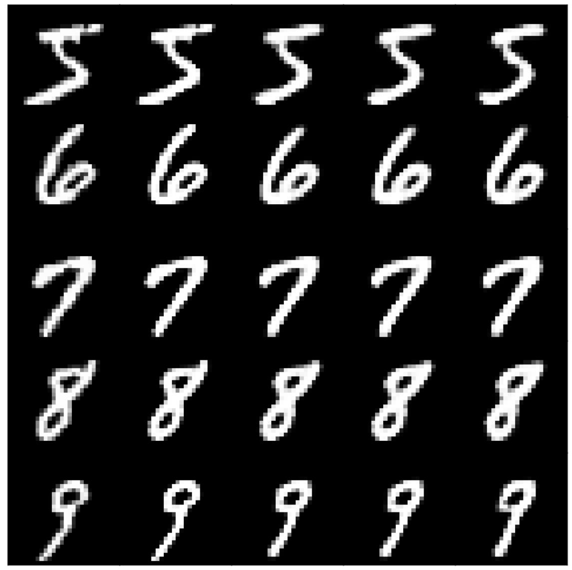
Until recently, the standard approach to training multilayer feedforward neural networks, such as autoencoders, is to use a variation of the backpropagation algorithm. We describe this algorithm later in this report. For now, we merely point out a key feature of the backpropagation algorithm: it attempts to optimize all weights and biases of a neural network simultaneously. In the case of large networks containing multiple layers and 1000+ nodes, backpropagation following random initialization of the parametes can fail due to local minima in the objective function **??**. A technique pioneered by Hinton et al. **??** largely overcomes this challenge by "pretraining" individual restricted Boltzmann machines (RBMs), which can then be "unrolled" to form a deep, multilayer autoencoder with weights and biases that are close to optimal prior to running a backpropagation algorithm.

An RBM may be represented as a complete bipartite graph where one set corresponds to "visible" units ν_i and the other to "hidden" units h_j. We work with binary RBMs in which each node is either on or off. A weight w_{ij} is associated with each edge that connects a visible unit ν_i to a hidden unit h_j. Given the state of all visible units, the probability that a hidden unit is in the "on" state is given by $\sigma\left(b_j + \sum_i \nu_i W_{ij}\right)$, where $\sigma(x)$ is the logistic function $1/(1+\exp(-x))$ and b_j is the bias associated with hidden unit j.

3 Results



(a) Encoding of digits $0-4$ in each layer



(b) Encoding of digits $5-9$ in each layer

Here's an equation:

$$\mathcal{C} = \frac{\sum_{q \in Q} \sum_{i=1}^{L_q} \mathbb{1}\left\{click \text{ @ } i \wedge i \leq 16\right\}}{\sum_{q \in Q} \sum_{i=1}^{L_q} \mathbb{1}\left\{i \leq 16\right\}}, \tag{1}$$

4 Conclusion

References

[1] *Restricted Boltzmann Machines (RBM)*, `http://deeplearning.net/tutorial/rbm.html`.

[2] G. E. Hinton and R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, Science **313** (2006), no. 5786, 504–507.

[3] N. Kambhatla and T. Leen., *Dimension reduction by local principal component analysis*, Neural Computation **9** (1997), no. 7, 1493–1516.

[4] Raul Rojas, *Neural networks - a systematic introduction*, `http://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf`, 1996.