Methods for Feature Learning and Extraction

Ross Freeman

May 4, 2017

1 Introduction

Feature learning is widely regarded as a fundamental basis for many modern machine learning algorithms, specifically those classified under unsupervised learning. Its applications span a wide variety of fields, ranging from determining the connectedness of airports to classifying an apartment's location based solely on a few properties. In particular, researchers have focused on image recognition; they are intrigued that the human brain can easily process and interpret images, yet even extremely powerful computers struggle to do the same. It is only logical then that researchers have turned to the human brain for inspiration and have attempted to mimic the visual cortex's operations in code. Several methods and algorithms have developed over the years with some remarkable success, despite their drawbacks

In general, these algorithms follow a similar structure in terms of their derivation and operation. The basic equation that they try to solve is:

$$X = WZ$$

where Z is the code matrix that represents a decomposed version of the input set and W is the weight matrix, which varies based on the output we are trying to derive. From here, the models diverge by placing different constraints on the input, the code matrix, or the weight matrix. In some cases, the constraint is enforced by transforming the input to satisfy it. A reconstruction error is then defined and minimized to determine the best code matrix for the model. Some methods also outline a method for preprocessing the input, which helps increase the accuracy of the trained model.

2 Sparse Coding

In their paper on sparse coding, Bruno Olshausen and David Field utilize the above steps to recreate the visual cortex's functionality to improve image recognition. From their perspective, the brain removes redundancy from the information it receives from the photoreceptors and represents the image as a collection of independent events. Previous research in sparse coding focused on 2-dimensional redundancy reduction, which

completely omits curves and other higher dimensional collections seen in natural images. To increase the accuracy of existing sparse coding algorithms, Olshausen and Field explore the creation of a linear strategy that can reduce higher order redundancy.

They use the following equation to create such a strategy, which is a slight modification of the feature extraction equation described earlier:

$$I(x) = \sum_i a_i \phi_i(x)$$

where $\phi_i(x)$ represents the basis vectors that determine the image code and a_i is the collection of amplitudes for each basis vector, which is computed for each image. The goal now is to find a set of basis vectors that can accurately represent the basic image structure of the input. An added constraint that Olshausen and Field make is that the basis functions must be overcomplete, which is when the number of basis vectors exceeds the dimensionality of the input. This will make the model more robust in the face of noise and help create greater flexibility in matching the model to the input since there can be multiple amplitudes that output the same image.

Finding the best set of basis vectors is analogous to creating a model with a probability distribution of each image ($P(I|\phi)$) that matches the probability distribution of images experienced in nature ($P(I^*)$). The Kullback-Leibler divergence is used as their reconstruction error, which is defined as:

$$KL = \int P^*(I) \log \frac{P^*(I)}{P(I|\phi)} dI$$

where it is equal to 0 if and only if the two distributions are equivalent. Since the probability distribution of nature is fixed, Olshausen and Field focus on the log likelihood instead. The optimization function thus becomes:

$$\phi^* = argmax_\phi [argmax_a \log P(I|a,\phi)P(a)]$$

and the learning rule for the model is:

$$\Delta\phi_i(x) = \eta(a_i r(x))$$

where $r(x)$ is defined as the residual image, or the difference between the output of the machine and the intended output. While their algorithm is quite accurate compared to other similar models, it is also slow and restricted. It assumes a linear image model, which is not always the case in nature. It also only consists of a single layer, preventing it from being as accurate as possible.

3 Independent Component Analysis

Aapo Hyvarinen and Erkki Oja describe another feature extraction algorithm known as ICA, or Independent Component Analysis. One of the many goals of ICA is to solve the cocktail party problem, which is to separate independent signals from a mixed signal input. They define their model as:

$$\mathbf{x} = \mathbf{As}$$

where **A** is the mixing matrix and **s** is the signal matrix. This is very similar to both the general model described earlier as well as the model derived for sparse coding. Once A is estimated, the signal matrix can be derived with:

$$\mathbf{s} = \mathbf{W}\mathbf{x}$$

where W is the inverse of A.

Hyvarinen and Oja developed several constraints on their model that serves to differentiate it from other feature extraction algorithms. First of all, each component in **s** is constrained to be statistically independent and have a non-Gaussian distribution. Non-Gaussianity is necessary for this model since a Gaussian distribution is symmetric, which does not give any indication of the direction of the columns in the mixing matrix, thus making it impossible to derive. Additionally, since both **s** and **A** are totally unknown, then the variance cannot be determined. Thus, the variance is arbitrarily fixed at 1 to simplify the model.

As seen earlier, the given derivation of the signal matrix requires that the mixing matrix be known. Since nothing can be assumed about the mixing matrix, Hyvarinen and Oja instead derive an estimator that can give a good estimation. They base their estimator off the equation:

$$\mathbf{w}^\top \mathbf{x} = \mathbf{z}^\top \mathbf{s}$$

where $z = \mathbf{A}\mathbf{w}^\top$. Thus, they determined that their problem is to maximize the non-Gaussianity of $\mathbf{w}^\top \mathbf{x}$. The researchers outline two main measures of non-Gaussianity, each of which has many advantages and disadvantages. They eventually determine that both estimators are impractical and instead focus on minimizing mutual information, which is defined as:

$$I(y_1, y_2, \ldots, y_m) = \sum_{i=1}^{m} H(y_i) - H(y)$$

where $H(y) = -\int f(y) \log f(y) dy$ is the differential entropy and $f(y)$ is the density of y. This equation is always nonnegative and equals 0 if and only if every variable is statistically independent. Hyvarinen and Oja continue to describe an additional measure of non-Gaussianity, but conclude that it is very much similar to minimizing the mutual information.

An area that particularly differentiates their approach from others is the process of preprocessing data. They describe two main ways in which this is done. The first is to center the input, which merely simplifies the computation of ICA. The second is to whiten the input, or to linearly transform it such that its components are uncorrelated and of equal variance. This makes the mixing matrix orthogonal and reduces the number of parameters that need to be estimated.

4 Non-negative Matrix Factorization

Non-negative Matrix Factorization (NMF) is an algorithm that has been around for several years and was the focus of a paper by Daniel Lee and H. Sebastian Seung.

Much like the models previously explored, Lee and Seung use the following equation as the basis for NMF:

$$V \approx WH$$

What makes this particular algorithm standout is its constraint. In fact, the constraint is in its very name - that the matrices in the equation are non-negative. The overall simplicity of this algorithm makes it very simple to implement, despite its increased computation time.

In determining a cost function, Lee and Seung discussed two distinct possibilities. The first is to minimize the Euclidean distance, which is defined as:

$$||V - WH||_2^2$$

The second is to minimize the divergence, which is defined as:

$$D(V||WH) = \sum_{ij}(V_{ij} \log \frac{V_{ij}}{WH_{ij}} - V_{ij} + WH_{ij})$$

Both functions converge, as proven by the researchers, making them excellent candidates for a cost function, though neither is definitively better than the other.

Where this algorithm diverges from most others is in its update rule. Rather than using an additive update rule, the authors instead chose to use a multiplicative one, defined as:

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^\top V)_{a\mu}}{(W^\top WH)_{a\mu}}$$
$$W_{ia} \leftarrow W_{ia} \frac{(VH^\top)_{ia}}{(WHH^\top)_{iat}}$$

for the Euclidean distance cost function and:

$$H_{a\mu} \leftarrow H_{a\mu} \frac{\sum_i W_{ia} V_{i\mu} / (WH)_{i\mu}}{\sum_k W_{ka}}$$
$$W_{ia} \leftarrow W_{ia} \frac{\sum_\mu H_{a\mu} V_{i\mu} / (WH)_{i\mu}}{\sum_v H_{av}}$$

for the divergence cost function. These multiplicative equations are simply a rescaled version of the additive update rule, potentially increasing the speed at which the model is updated.

5 Stacked Denoising Autoencoders

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol discuss stacked denoising autoencoders in their 2010 paper, which is a slight modification to the already established autoencoders. Their motivation, much like

sparse coding, is to mimic the functions of the visual cortex to enable feature recognition and classification.

The authors describe note one, but two base equations for this model. The first is the encoder, which is defined as:

$$\mathbf{y} = f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b})$$

where $\theta = \{\mathbf{W}, \mathbf{x}\}$ are the parameters to be learned. This appears very much like the general equation described at the beginning of this paper as well as to the other models described earlier. The purpose of the encoder is to transform the input into some hidden representation that can later be decoded. Similarly, the decoder is defined as:

$$\mathbf{z} = g_{\theta'}(\mathbf{y}) = s(\mathbf{W}^{'}\mathbf{y} + \mathbf{b}^{'})$$

where $\theta^{'} = \{\mathbf{W}^{'}, \mathbf{b}^{'}\}$, $\mathbf{y}$ is the hidden representation output from the encoder, and $\mathbf{z}$ is the output of the decoder.

The model's reconstruction error is defined as:

$$L(x,z) = C(\sigma^2)||x-z||^2$$

for real valued x, where C is a constant that can be ignored for optimization. For binary values, the reconstruction is defined as:

$$L(x,z) = -\sum_j [x_j \log z_j + (1-x_j)\log(1-z_j)]$$

which is very similar to the reconstruction error defined for a logistic regression model.

What differentiates the researcher's model from others is that it preprocesses the input, similar to ICA. An issue with the reconstruction error defined earlier is that the model will trivially solve the problem to simply be $X = Y$, which is a very uninteresting solution. The traditional solution to this problem is to simply reduce the dimensionality of Y to produce an under-complete representation. The obvious implication is that this results in a loss of data and essentially makes this algorithm perform PCA.

The researchers instead decided to "corrupt" the input by preprocessing it through a noising algorithm. The corrupted input is then used to train the model. However, the reconstruction error still compares the output of the decoder and the original, uncorrupted input. This process will help provide a more interesting model that retains the dimensionality of the initial input.

As for how to corrupt the input, the paper discusses focuses on a well established noise algorithm called Masking Noise. This algorithm takes a fraction v of the elements of $\mathbf{x}$ at random and forces them to 0. The autoencoder is thus trained to be able to "fill" in these 0's, providing for a useful and interesting model.

This modification to the traditional autoencoder model also warrants a change to the reconstruction error. Since we want to output the uncorrupted $\mathbf{x}$, the loss function should put an emphasis on the corrupted elements. This makes the squared loss function:

$$L_{2,\alpha}(\mathbf{x},\mathbf{z}) = \alpha(\sum_{j\in\tilde{\mathbf{x}}}(\mathbf{x}_j - \mathbf{z}_j)^2) + \beta((\sum_{j\notin\tilde{\mathbf{x}}}(\mathbf{x}_j - \mathbf{z}_j)^2)$$

and the cross-entropy loss:

$$L_{H,\alpha}(\mathbf{x},\mathbf{z}) = \alpha(-\sum_{j\in\tilde{\mathbf{x}}}(\mathbf{x}_j \log \mathbf{z}_j + (1-\mathbf{x}_j)\log(1-\mathbf{z}_j)) + \beta(-\sum_{j\notin\tilde{\mathbf{x}}}(\mathbf{x}_j \log \mathbf{z}_j + (1-\mathbf{x}_j)\log(1-\mathbf{z}_j))$$

where α and β define how much to weigh the corrupted elements versus the uncorrupted elements. α can be set to 1 and β set to 0 to make the model only account for the corrupted elements.

To further improve the accuracy of the model, the autoencoder can be stacked. In other words, once the denoising autoencoder is trained, the uncorrupted input is then passed through to the trained model and the output is used to train a second. Layer. This can be repeated multiple times and then topped off with a basic supervised learning algorithm, such as logistic regression, to create a deep neural network.

Experimentally, the researchers show that the SDAE with 3 layers outperforms a standard SAE as well as a number of other models in almost every test they perform. This makes the SDAE an excellent model for feature learning, especially given its relatively simple implementation. The main ambiguity that the researchers leave is which noise algorithm should be used, noting that the optimal one will likely vary depending on what the model is trying to accomplish.

6 Another Autoencoder Approach

G.E. Hinton and R. R. Salakhutdinov discuss another modification to that can be made to the classic stacked autoencoder model. While they do not explicitly define a stacked autoencoder, it appears that they refer to a standard model, similar to what is described early in the previous section. The difference with the previous model, however, is that Hinton and Salakhutdinov make an additional of preprocessing on the weight matrix. Normally, the weight matrix is initialized to either random values or to some set of fixed values. However, this can lead to decreased performance, depending on the input data used. If the initial weights are rather large, the autoencoders will struggle to find a local minima of the discrepancy between the original input and the output. With small initial weights, the gradients in earlier layers are tiny, making it practically infeasible to train later layers.

The researchers propose a pretraining algorithm that can somehow set good initial values for the weight matrix to produce an optimal model. They first define an energy model for an image recognition model as:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in pixels} b_i bv_i - \sum_{j\in features} b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$

where $\mathbf{v}$ is the set of visible pixels, $\mathbf{h}$ is the set of hidden units, corresponding to the feature detectors, and w is the weight between them. This energy function is used by the model to produce probabilities of every possible "learned" image.

They then propose creating a function which they refer to as a confabulation, which is meant to set each individual pixel with a certain probability. Using this, they describe a weight matrix update rule as:

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

where ε is the learning rate, $< v_i h_j >_{data}$ is the percentage of time each pixel corresponds to the hidden feature, and $< v_i h_j >_{recon}$ is the fraction for the confabulations. After the weight matrix is pretrained, the deep autoencoder can then be layered, with the hidden values of one layer becoming the visible units of the next.

The researchers performed multiple tests on their model, including on the MNSIT data set. Through their experimentation, they found that their pretrained deep autoencoder significantly outperformed both PCA and a standard deep autoencoder. In fact, it achieved a 1.2% error rate with backpropagation, beating commonly used models. They conclude that this is an excellent modification to the standard autoecncoder, leaving potential applications open for discovery.

7 Conclusion

Feature learning is still a relatively new field of study, even for computer science. The oldest paper discussed is only from 1997 and the newest is as recent as 2010. Considering the number of developments over such a short time period, it will be interesting to see what else researchers will discover in this field. The models outlined here can potentially be refined and utilized to solve various societal problems, It is now only a matter of time before new, more accurate, and more efficient models are discovered, enabling a new era of problem solving.

References

[1] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[2] A. Hyvarinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(45):411 – 430, 2000.

[3] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2001.

[4] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311 – 3325, 1997.

[5] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.