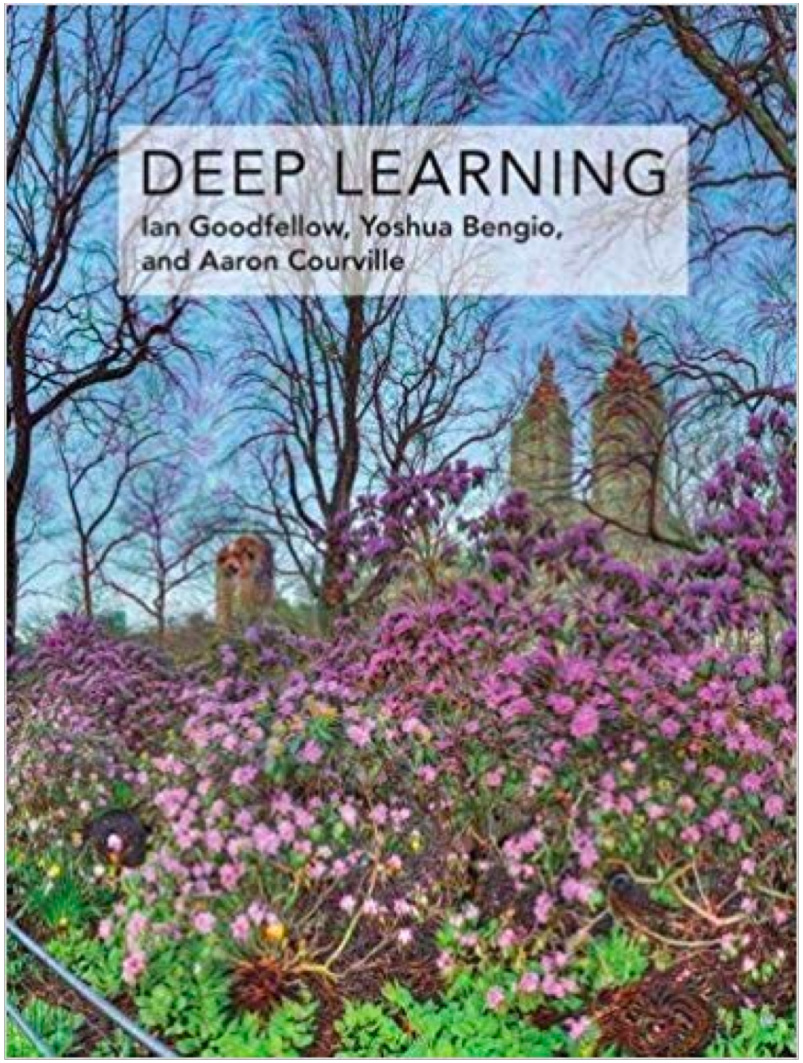
國立交通大學
National Chiao Tung University

Deep Learning
深度學習
Fall 2018

Autoencoders
(Chapter 14)

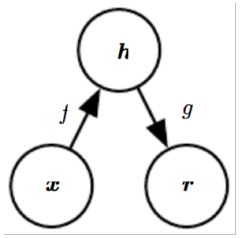


Prof. Chia-Han Lee
李佳翰 副教授

Autoencoders

- An autoencoder is a neural network that is trained to attempt to copy its input to its output.

- The network may be viewed as consisting of two parts: an encoder function $\boldsymbol{h} = f(\boldsymbol{x})$ and a decoder that produces a reconstruction $\boldsymbol{r} = g(\boldsymbol{h})$.



- If an autoencoder succeeds in simply learning to set $g(f(\boldsymbol{x})) = \boldsymbol{x}$ everywhere, then it is not especially useful. Autoencoders are designed to be unable to learn to copy perfectly. Usually they are restricted in ways that allow them to copy only approximately, and to copy only input that resembles the training data. Because the model is forced to prioritize which aspects of the input should be copied, it often learns useful properties of the data.

Autoencoders

- Modern autoencoders have generalized the idea of an encoder and a decoder beyond deterministic functions to stochastic mappings $p_{\text{encoder}}(\boldsymbol{h}|\boldsymbol{x})$ and $p_{\text{encoder}}(\boldsymbol{x}|\boldsymbol{h})$.

- Traditionally, autoencoders were used for dimensionality reduction or feature learning. Recently, theoretical connections between autoencoders and latent variable models have brought autoencoders to the forefront of generative modeling.

- Autoencoders may be thought of as being a special case of feedforward networks and may be trained with all the same techniques.

Undercomplete autoencoders

- Copying the input to the output may sound useless, but we are typically not interested in the output of the decoder. Instead, we hope that training the autoencoder to perform the input copying task will result in $\boldsymbol{h}$ taking on useful properties.

- One way to obtain useful features from the autoencoder is to constrain $\boldsymbol{h}$ to have a smaller dimension than $\boldsymbol{x}$. An autoencoder whose code dimension is less than the input dimension is called undercomplete. Learning an undercomplete representation forces the autoencoder to capture the most salient features of the training data.

Undercomplete autoencoders

- The learning process is described simply as minimizing a loss function

$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))), \tag{14.1}$$

where L is a loss function penalizing $g(f(\boldsymbol{x}))$ for being dissimilar from $\boldsymbol{x}$, such as the mean squared error.

- When the decoder is linear and L is the mean squared error, an undercomplete autoencoder learns to span the same subspace as PCA. In this case, an autoencoder trained to perform the copying task has learned the principal subspace of the training data as a side effect.

- Autoencoders with nonlinear encoder functions f and nonlinear decoder functions g can thus learn a more powerful nonlinear generalization of PCA.

Undercomplete autoencoders

- Unfortunately, if the encoder and decoder are allowed too much capacity, the autoencoder can learn to perform the copying task without extracting useful information about the distribution of the data.

- If the hidden code is allowed to have dimension equal to the input, and in the overcomplete case in which the hidden code has dimension greater than the input, even a linear encoder and a linear decoder can learn to copy the input to the output without learning anything useful about the data distribution.

Regularized autoencoders

- Rather than limiting the model capacity by keeping the encoder and decoder shallow and the code size small, regularized autoencoders use a loss function that encourages the model to have other properties besides the ability to copy its input to its output.

- These properties include sparsity of the representation, smallness of the derivative of the representation, and robustness to noise or to missing inputs.

- A regularized autoencoder can be nonlinear and overcomplete but still learn something useful about the data distribution, even if the model capacity is great enough to learn a trivial identity function.

Sparse autoencoders

- A sparse autoencoder is simply an autoencoder whose training criterion involves a sparsity penalty $\Omega(\boldsymbol{h})$ on the code layer $\boldsymbol{h}$, in addition to the reconstruction error:

$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{h}), \tag{14.2}$$

where $g(\boldsymbol{h})$ is the decoder output, and typically we have $\boldsymbol{h} = f(\boldsymbol{x})$, the encoder output.

- Sparse autoencoders are typically used to learn features for another task, such as classification.

- An autoencoder that has been regularized to be sparse must respond to unique statistical features of the dataset it has been trained on, rather than simply acting as an identity function.

Sparse autoencoders

- We can think of the entire sparse autoencoder framework as approximating likelihood training of a generative model that has latent variables.

- Suppose we have a model with visible variables $\boldsymbol{x}$ and latent variables $\boldsymbol{h}$, with an explicit joint distribution $p_{\text{model}}(\boldsymbol{x}, \boldsymbol{h}) = p_{\text{model}}(\boldsymbol{h})\, p_{\text{model}}(\boldsymbol{x}|\boldsymbol{h})$. We refer to $p_{\text{model}}(\boldsymbol{h})$ as the model's prior distribution over the latent variables, representing the model's beliefs prior to seeing $\boldsymbol{x}$.

- The log-likelihood can be decomposed as

$$\log p_{\text{model}}(\boldsymbol{x}) = \log \sum_{\boldsymbol{h}} p_{\text{model}}(\boldsymbol{h}, \boldsymbol{x}). \tag{14.3}$$

Sparse autoencoders

- We can think of the autoencoder as approximating this sum with a point estimate for just one highly likely value for $\boldsymbol{h}$. This is similar to the sparse coding generative model, but with $\boldsymbol{h}$ being the output of the parametric encoder rather than the result of an optimization that infers the most likely $\boldsymbol{h}$. From this point of view, with this chosen $\boldsymbol{h}$, we are maximizing

$$\log p_{\text{model}}(\boldsymbol{h}, \boldsymbol{x}) = \log p_{\text{model}}(\boldsymbol{h}) + \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{h}). \tag{14.4}$$

- The $\log p_{\text{model}}(\boldsymbol{h})$ term can be sparsity inducing. For example, the Laplace prior,

$$p_{\text{model}}(h_i) = \frac{\lambda}{2} e^{-\lambda |h_i|}, \tag{14.5}$$

corresponds to an absolute value sparsity penalty.

Sparse autoencoders

- Expressing the log-prior as an absolute value penalty, we obtain

$$\Omega(\boldsymbol{h}) = \lambda \sum_i |h_i|, \tag{14.6}$$

$$-\log p_{\text{model}}(\boldsymbol{h}) = \sum_i \left(\lambda |h_i| - \log \frac{\lambda}{2} \right) = \Omega(\boldsymbol{h}) + \text{const}, \tag{14.7}$$

where the constant term depends only on λ and not $\boldsymbol{h}$.

- We typically treat λ as a hyperparameter and discard the constant term since it does not affect the parameter learning.

Sparse autoencoders

- From this point of view of sparsity as resulting from the effect of $p_{\text{model}}(\boldsymbol{h})$ on approximate maximum likelihood learning, the sparsity penalty is not a regularization term at all. It is just a consequence of the model's distribution over its latent variables.

- This view provides a different motivation for training an autoencoder: it is a way of approximately training a generative model.

- It also provides a different reason for why the features learned by the autoencoder are useful: they describe the latent variables that explain the input.

Denoising autoencoders

- Rather than adding a penalty Ω to the cost function, we can obtain an autoencoder that learns something useful by changing the reconstruction error term of the cost function.

- Traditionally, autoencoders minimize some function

$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))), \tag{14.8}$$

where L is a loss function penalizing $g(f(\boldsymbol{x}))$ for being dissimilar from $\boldsymbol{x}$, such as the L^2 norm of their difference. This encourages $g \circ f$ to learn to be merely an identity function if they have the capacity to do so.

Denoising autoencoders

- A denoising autoencoder (DAE) instead minimizes

$$L(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}}))), \tag{14.9}$$

where $\tilde{\boldsymbol{x}}$ is a copy of $\boldsymbol{x}$ that has been corrupted by some form of noise. Denoising autoencoders must therefore undo this corruption rather than simply copying input.

- Denoising training forces f and g to implicitly learn the structure of $p_{\text{data}}(\boldsymbol{x})$. Denoising autoencoders provide another example of how useful properties can emerge as a byproduct of minimizing reconstruction error.

- They are an example of how overcomplete, high-capacity models may be used as autoencoders as long as care is taken to prevent them from learning the identity function.

Regularizing by penalizing derivatives

- Another strategy for regularizing an autoencoder is to use a penalty Ω, as in sparse autoencoders,

$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{h}, \boldsymbol{x}), \tag{14.10}$$

but with a different form of Ω:

$$\Omega(\boldsymbol{h}, \boldsymbol{x}) = \lambda \sum_i ||\nabla_{\boldsymbol{x}} h_i||^2. \tag{14.11}$$

- This forces the model to learn a function that does not change much when $\boldsymbol{x}$ changes slightly. Because this penalty is applied only at training examples, it forces the autoencoder to learn features that capture information about the training distribution. An autoencoder regularized in this way is called a contractive autoencoder (CAE).

Representational power, layer size and depth

- Autoencoders are often trained with only a single layer encoder and a single layer decoder. However, using deep encoders and decoders offers many advantages.

- An autoencoder with a single hidden layer is able to represent the identity function along the domain of the data arbitrarily well. However, we are not able to enforce arbitrary constraints, such as that the code should be sparse.

- Experimentally, deep autoencoders yield much better compression than corresponding shallow or linear autoencoders.

Training autoencoders

- All autoencoder training procedures involve a compromise between two forces:

- Learning a representation $\boldsymbol{h}$ of a training example $\boldsymbol{x}$ such that $\boldsymbol{x}$ can be approximately recovered from $\boldsymbol{h}$ through a decoder. The fact that $\boldsymbol{x}$ is drawn from the training data is crucial, because it means the autoencoder need not successfully reconstruct inputs that are not probable under the data-generating distribution.

- Satisfying the constraint or regularization penalty. This can be an architectural constraint that limits the capacity of the autoencoder, or a regularization term added to the reconstruction cost. These techniques generally prefer solutions that are less sensitive to the input.

Training autoencoders

- Clearly, neither force alone would be useful—copying the input to the output is not useful on its own, nor is ignoring the input. Instead, the two forces together are useful because they force the hidden representation to capture information about the structure of the data-generating distribution.

- The important principle is that the autoencoder can afford to represent only the variations that are needed to reconstruct training examples.

Contractive autoencoders

- The contractive autoencoder introduces an explicit regularizer on the code $\boldsymbol{h} = f(\boldsymbol{x})$, encouraging the derivatives of f to be as small as possible:

$$\Omega(\boldsymbol{h}) = \lambda \left\| \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_F^2. \tag{14.18}$$

- The penalty $\Omega(\boldsymbol{h})$ is the squared Frobenius norm (sum of squared elements) of the Jacobian matrix of partial derivatives associated with the encoder function.

Contractive autoencoders

- There is a connection between the denoising autoencoder and the contractive autoencoder: It was shown that in the limit of small Gaussian input noise, the denoising reconstruction error is equivalent to a contractive penalty on the reconstruction function that maps $\boldsymbol{x}$ to $\boldsymbol{r} = g(f(\boldsymbol{x}))$.

- Denoising autoencoders make the reconstruction function resist small but finite-sized perturbations of the input.

- Contractive autoencoders make the feature extraction function resist infinitesimal perturbations of the input.

Contractive autoencoders

- The name contractive arises from the way that the CAE warps space. Specifically, because the CAE is trained to resist perturbations of its input, it is encouraged to map a neighborhood of input points to a smaller neighborhood of output points. We can think of this as contracting the input neighborhood to a smaller output neighborhood.

- To clarify, the CAE is contractive only locally—all perturbations of a training point x are mapped near to $f(\boldsymbol{x})$. Globally, two different points $\boldsymbol{x}$ and $\boldsymbol{x}'$ may be mapped to $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ points that are farther apart than the original points.

Contractive autoencoders

- In the case of the CAE, these two forces in training are reconstruction error and the contractive penalty $\Omega(\boldsymbol{h})$.
- Reconstruction error alone would encourage the CAE to learn an identity function.
- The contractive penalty alone would encourage the CAE to learn features that are constant with respect to $\boldsymbol{x}$.
- The compromise between these two forces yields an autoencoder whose derivatives $\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}$ are mostly tiny. Only a small number of hidden units, corresponding to a small number of directions in the input, may have significant derivatives.

Applications of autoencoders

- Autoencoders have been successfully applied to dimensionality reduction and information retrieval tasks.
- Lower-dimensional representations can improve performance on many tasks, such as classification. Many forms of dimensionality reduction place semantically related examples near each other. The hints provided by the mapping to the lower-dimensional space aid generalization.
- Models of smaller spaces consume less memory and runtime.

Applications of autoencoders

- Information retrieval is the task of finding entries in a database that resemble a query entry. This task derives the usual benefits from dimensionality reduction that other tasks do, but also derives the additional benefit that search can become extremely efficient in certain kinds of low-dimensional spaces.

- Specifically, if we train the dimensionality reduction algorithm to produce a code that is low-dimensional and binary, then we can store all database entries in a hash table that maps binary code vectors to entries. This hash table allows us to perform information retrieval by returning all database entries that have the same binary code as the query.