WIKIPEDIA

Feature learning

In machine learning, **feature learning** or **representation learning**[1] is a set of techniques that allows a system to automatically discover the representations needed for feature detection or classification from raw data. This replaces manual feature engineering and allows a machine to both learn the features and use them to perform a specific task.

Feature learning is motivated by the fact that machine learning tasks such as classification often require input that is mathematically and computationally convenient to process. However, real-world data such as images, video, and sensor data has not yielded to attempts to algorithmically define specific features. An alternative is to discover such features or representations through examination, without relying on explicit algorithms.

Feature learning can be either supervised or unsupervised.

- In supervised feature learning, features are learned using labeled input data. Examples include supervised neural networks, multilayer perceptron and (supervised) dictionary learning.
- In unsupervised feature learning, features are learned with unlabeled input data. Examples include dictionary learning, independent component analysis, autoencoders, matrix factorization[2] and various forms of clustering.[3][4][5]

Contents

Supervised

Supervised feature learning is learning features from labeled data. The data label allows the system to compute an error term, the degree to which the system fails to produce the label, which can then be used as feedback to correct the learning process (reduce/minimize the error). Approaches include:

Supervised dictionary learning

Dictionary learning develops a set (dictionary) of representative elements from the input data such that each data point can be represented as a weighted sum of the representative elements. The dictionary elements and the weights may be found by minimizing the average representation error (over the input data), together with *L1* regularization on the weights to enable sparsity (i.e., the representation of each data point has only a few nonzero weights).

Supervised dictionary learning exploits both the structure underlying the input data and the labels for optimizing the dictionary elements. For example, this[6] supervised dictionary learning technique applies dictionary learning on classification problems by jointly optimizing the dictionary elements, weights for representing data points, and parameters of the classifier based on the input data. In particular, a minimization problem is formulated, where the objective function consists of the classification error, the representation error, an *L1* regularization on the representing weights for each data point (to enable sparse representation of data), and an *L2* regularization on the parameters of the classifier.

Neural networks

Neural networks are a family of learning algorithms that use a "network" consisting of multiple layers of inter-connected nodes. It is inspired by the animal nervous system, where the nodes are viewed as neurons and edges are viewed as synapses. Each edge has an associated weight, and the network defines computational rules for passing input data from the network's input layer to the output layer. A network function associated with a neural network characterizes the relationship between input and output layers, which is parameterized by the weights. With appropriately defined network functions, various learning tasks can be performed by minimizing a cost function over the network function (weights).

Multilayer neural networks can be used to perform feature learning, since they learn a representation of their input at the hidden layer(s) which is subsequently used for classification or regression at the output layer. The most popular network architecture of this type is Siamese networks.

Unsupervised

Unsupervised feature learning is learning features from unlabeled data. The goal of unsupervised feature learning is often to discover low-dimensional features that capture some structure underlying the high-dimensional input data. When the feature learning is performed in an unsupervised way, it enables a form of semisupervised learning where features learned from an unlabeled dataset are then employed to improve performance in a supervised setting with labeled data.[7][8] Several approaches are introduced in the following.

K-means clustering

K-means clustering is an approach for vector quantization. In particular, given a set of *n* vectors, *k*-means clustering groups them into k clusters (i.e., subsets) in such a way that each vector belongs to the cluster with the closest mean. The problem is computationally NP-hard, although suboptimal greedy algorithms have been developed.

K-means clustering can be used to group an unlabeled set of inputs into *k* clusters, and then use the centroids of these clusters to produce features. These features can be produced in several ways. The simplest is to add *k* binary features to each sample, where each feature *j* has value one iff the *j*th centroid learned by *k*-means is the closest to the sample under consideration.[3] It is also possible to use the distances to the clusters as features, perhaps after transforming them through a radial basis function (a technique that has been used to train RBF networks[9]). Coates and Ng note that certain variants of *k*-means behave similarly to sparse coding algorithms.[10]

In a comparative evaluation of unsupervised feature learning methods, Coates, Lee and Ng found that *k*-means clustering with an appropriate transformation outperforms the more recently invented auto-encoders and RBMs on an image classification task.[3] *K*-means also improves performance in the domain of NLP, specifically for named-entity recognition;[11] there, it competes with Brown clustering, as well as with distributed word representations (also known as neural word embeddings).[8]

Principal component analysis

Principal component analysis (PCA) is often used for dimension reduction. Given an unlabeled set of *n* input data vectors, PCA generates *p* (which is much smaller than the dimension of the input data) right singular vectors corresponding to the *p* largest singular values of the data matrix, where the *k*th row of the data matrix is the *k*th input data vector shifted by the sample mean of the input (i.e., subtracting the sample mean from the data vector). Equivalently, these singular vectors are the eigenvectors corresponding to the *p* largest eigenvalues of the sample covariance matrix of the input vectors. These *p* singular vectors are the feature vectors learned from the input data, and they represent directions along which the data has the largest variations.

PCA is a linear feature learning approach since the *p* singular vectors are linear functions of the data matrix. The singular vectors can be generated via a simple algorithm with *p* iterations. In the *i*th iteration, the projection of the data matrix on the *(i-1)*th eigenvector is subtracted, and the *i*th singular vector is found as the right singular vector corresponding to the largest singular of the residual data matrix.

PCA has several limitations. First, it assumes that the directions with large variance are of most interest, which may not be the case. PCA only relies on orthogonal transformations of the original data, and it exploits only the first- and second-order moments of the data, which may not well characterize the data distribution. Furthermore, PCA can effectively reduce dimension only when the input data vectors are correlated (which results in a few dominant eigenvalues).

Local linear embedding

Local linear embedding (LLE) is a nonlinear learning approach for generating low-dimensional neighbor-preserving representations from (unlabeled) high-dimension input. The approach was proposed by Roweis and Saul (2000).[12][13] The general idea of LLE is to reconstruct the original high-dimensional data using lower-dimensional points while maintaining some geometric properties of the neighborhoods in the original data set.

LLE consists of two major steps. The first step is for "neighbor-preserving", where each input data point *Xi* is reconstructed as a weighted sum of *K* nearest neighbor data points, and the optimal weights are found by minimizing the average squared reconstruction error (i.e., difference between an input point and its reconstruction) under the constraint that the weights associated with each point sum up to one. The second step is for "dimension reduction," by looking for vectors in a lower-dimensional space that minimizes the representation error using the optimized weights in the first step. Note that in the first step, the weights are optimized with fixed data, which can be solved as a least squares problem. In the second step, lower-dimensional points are optimized with fixed weights, which can be solved via sparse eigenvalue decomposition.

The reconstruction weights obtained in the first step capture the "intrinsic geometric properties" of a neighborhood in the input data.[13] It is assumed that original data lie on a smooth lower-dimensional manifold, and the "intrinsic geometric properties" captured by the weights of the original data are also expected to be on the manifold. This is why the same weights are used in the second step of LLE. Compared with PCA, LLE is more powerful in exploiting the underlying data structure.

Independent component analysis

Independent component analysis (ICA) is a technique for forming a data representation using a weighted sum of independent non-Gaussian components.[14] The assumption of non-Gaussian is imposed since the weights cannot be uniquely determined when all the components follow Gaussian distribution.

Unsupervised dictionary learning

Unsupervised dictionary learning does not utilize data labels and exploits the structure underlying the data for optimizing dictionary elements. An example of unsupervised dictionary learning is sparse coding, which aims to learn basis functions (dictionary elements) for data representation from unlabeled input data. Sparse coding can be applied to learn overcomplete dictionaries, where the number of dictionary elements is larger than the dimension of the input data.[15] Aharon et al. proposed algorithm K-SVD for learning a dictionary of elements that enables sparse representation.[16]

Multilayer/deep architectures

The hierarchical architecture of the biological neural system inspires deep learning architectures for feature learning by stacking multiple layers of learning nodes.[17] These architectures are often designed based on the assumption of distributed representation: observed data is generated by the interactions of many different factors on multiple levels. In a deep learning architecture, the output of each intermediate layer can be viewed as a representation of the original input data. Each level uses the representation produced by previous level as input, and produces new representations as output, which is then fed to higher levels. The input at the bottom layer is raw data, and the output of the final layer is the final low-dimensional feature or representation.

Restricted Boltzmann machine

Restricted Boltzmann machines (RBMs) are often used as a building block for multilayer learning architectures.[3][18] An RBM can be represented by an undirected bipartite graph consisting of a group of binary hidden variables, a group of visible variables, and edges connecting the hidden and visible nodes. It is a special case of the more general Boltzmann machines with the constraint of no intra-node connections. Each edge in an RBM is associated with a weight. The weights together with the connections define an energy function, based on which a joint distribution of visible and hidden nodes can be devised. Based on the topology of the RBM, the hidden (visible) variables are independent, conditioned on the visible (hidden) variables. Such conditional independence facilitates computations.

An RBM can be viewed as a single layer architecture for unsupervised feature learning. In particular, the visible variables correspond to input data, and the hidden variables correspond to feature detectors. The weights can be trained by maximizing the probability of visible variables using Hinton's contrastive divergence (CD) algorithm.[18]

In general training RBM by solving the maximization problem tends to result in non-sparse representations. Sparse RBM[19] was proposed to enable sparse representations. The idea is to add a regularization term in the objective function of data likelihood, which penalizes the deviation of the expected hidden variables from a small constant p.

Autoencoder

An autoencoder consisting of an encoder and a decoder is a paradigm for deep learning architectures. An example is provided by Hinton and Salakhutdinov[18] where the encoder uses raw data (e.g., image) as input and produces feature or representation as output and the decoder uses the extracted feature from the encoder as input and reconstructs the original input raw data as output. The encoder and decoder are constructed by stacking multiple layers of RBMs. The parameters involved in the architecture were originally trained in a greedy layer-by-layer manner: after one layer of feature detectors is learned, they are fed up as visible variables for training the corresponding RBM. Current approaches typically apply end-to-end training with stochastic gradient descent methods. Training can be repeated until some stopping criteria are satisfied.

See also

- Automated machine learning (AutoML)
- Basis function
- Deep learning
- Feature detection (computer vision)
- Feature extraction
- Kernel trick
- Vector quantization
- Variational autoencoder

References

1. Y. Bengio; A. Courville; P. Vincent (2013). "Representation Learning: A Review and New Perspectives". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **35** (8): 1798–1828. arXiv:1206.5538 (https://arxiv.org/abs/1206.5538). doi:10.1109/tpami.2013.50 (https://doi.org/10.1109%2Ftpami.2013.50). PMID 23787338 (https://pubmed.ncbi.nlm.nih.gov/23787338). S2CID 393948 (https://api.semanticscholar.org/CorpusID:393948).
2. Nathan Srebro; Jason D. M. Rennie; Tommi S. Jaakkola (2004). *Maximum-Margin Matrix Factorization*. NIPS.
3. Coates, Adam; Lee, Honglak; Ng, Andrew Y. (2011). *An analysis of single-layer networks in unsupervised feature learning* (https://web.archive.org/web/20170813153615/http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS2011_CoatesNL11.pdf) (PDF). Int'l Conf. on AI and Statistics (AISTATS). Archived from the original (http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS2011_CoatesNL11.pdf) (PDF) on 2017-08-13. Retrieved 2014-11-24.
4. Csurka, Gabriella; Dance, Christopher C.; Fan, Lixin; Willamowski, Jutta; Bray, Cédric (2004). *Visual categorization with bags of keypoints* (https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/csurka-eccv-04.pdf) (PDF). ECCV Workshop on Statistical Learning in Computer Vision.
5. Daniel Jurafsky; James H. Martin (2009). *Speech and Language Processing*. Pearson Education International. pp. 145–146.

6. Mairal, Julien; Bach, Francis; Ponce, Jean; Sapiro, Guillermo; Zisserman, Andrew (2009). "Supervised Dictionary Learning". *Advances in Neural Information Processing Systems*.
7. Percy Liang (2005). *Semi-Supervised Learning for Natural Language* (http://people.csail.mit.edu/pliang/papers/meng-thesis.pdf) (PDF) (M. Eng.). MIT. pp. 44–52.
8. Joseph Turian; Lev Ratinov; Yoshua Bengio (2010). *Word representations: a simple and general method for semi-supervised learning* (https://web.archive.org/web/20140226202823/http://www.newdesign.aclweb.org/anthology/P/P10/P10-1040.pdf) (PDF). Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Archived from the original (http://www.newdesign.aclweb.org/anthology/P/P10/P10-1040.pdf) (PDF) on 2014-02-26. Retrieved 2014-02-22.
9. Schwenker, Friedhelm; Kestler, Hans A.; Palm, Günther (2001). "Three learning phases for radial-basis-function networks". *Neural Networks*. **14** (4–5): 439–458. CiteSeerX 10.1.1.109.312 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.109.312). doi:10.1016/s0893-6080(01)00027-2 (https://doi.org/10.1016%2Fs0893-6080%2801%2900027-2). PMID 11411631 (https://pubmed.ncbi.nlm.nih.gov/11411631).
10. Coates, Adam; Ng, Andrew Y. (2012). "Learning feature representations with k-means". In G. Montavon, G. B. Orr and K.-R. Müller (ed.). *Neural Networks: Tricks of the Trade*. Springer.
11. Dekang Lin; Xiaoyun Wu (2009). *Phrase clustering for discriminative learning* (http://wmmks.csie.ncku.edu.tw/ACL-IJCNLP-2009/ACLIJCNLP/pdf/ACLIJCNLP116.pdf) (PDF). Proc. J. Conf. of the ACL and 4th Int'l J. Conf. on Natural Language Processing of the AFNLP. pp. 1030–1038.
12. Roweis, Sam T; Saul, Lawrence K (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding". *Science*. New Series. **290** (5500): 2323–2326. Bibcode:2000Sci...290.2323R (https://ui.adsabs.harvard.edu/abs/2000Sci...290.2323R). doi:10.1126/science.290.5500.2323 (https://doi.org/10.1126%2Fscience.290.5500.2323). JSTOR 3081722 (https://www.jstor.org/stable/3081722). PMID 11125150 (https://pubmed.ncbi.nlm.nih.gov/11125150).
13. Saul, Lawrence K; Roweis, Sam T (2000). "An Introduction to Locally Linear Embedding" (http://www.cs.toronto.edu/~roweis/lle/publications.html).
14. Hyvärinen, Aapo; Oja, Erkki (2000). "Independent Component Analysis: Algorithms and Applications". *Neural Networks*. **13** (4): 411–430. doi:10.1016/s0893-6080(00)00026-5 (https://doi.org/10.1016%2Fs0893-6080%2800%2900026-5). PMID 10946390 (https://pubmed.ncbi.nlm.nih.gov/10946390).
15. Lee, Honglak; Battle, Alexis; Raina, Rajat; Ng, Andrew Y (2007). "Efficient sparse coding algorithms". *Advances in Neural Information Processing Systems*.
16. Aharon, Michal; Elad, Michael; Bruckstein, Alfred (2006). "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation". *IEEE Trans. Signal Process*. **54** (11): 4311–4322. Bibcode:2006ITSP...54.4311A (https://ui.adsabs.harvard.edu/abs/2006ITSP...54.4311A). doi:10.1109/TSP.2006.881199 (https://doi.org/10.1109%2FTSP.2006.881199). S2CID 7477309 (https://api.semanticscholar.org/CorpusID:7477309).
17. Bengio, Yoshua (2009). "Learning Deep Architectures for AI". *Foundations and Trends in Machine Learning*. **2** (1): 1–127. doi:10.1561/2200000006 (https://doi.org/10.1561%2F2200000006).
18. Hinton, G. E.; Salakhutdinov, R. R. (2006). "Reducing the Dimensionality of Data with Neural Networks" (http://www.cs.toronto.edu/~hinton/science.pdf) (PDF). *Science*. **313** (5786): 504–507. Bibcode:2006Sci...313..504H (https://ui.adsabs.harvard.edu/abs/2006Sci...313..504H). doi:10.1126/science.1127647 (https://doi.org/10.1126%2Fscience.1127647). PMID 16873662 (https://pubmed.ncbi.nlm.nih.gov/16873662). S2CID 1658773 (https://api.semanticscholar.org/CorpusID:1658773).

19. Lee, Honglak; Ekanadham, Chaitanya; Andrew, Ng (2008). "Sparse deep belief net model for visual area V2". *Advances in Neural Information Processing Systems*.