Chapter 2
Fundamental Knowledge of Machine Learning

Abstract This chapter introduces the basic concepts and methods of machine learning that are related to this book. The classical machine learning methods, like neural network (CNN), support vector machine (SVM), clustering, Bayesian networks, sparse learning, Boosting, and deep learning, are presented in this chapter.

Machine learning (ML) is a subfield of artificial intelligence (AI) and computer science (CS). It concerns the construction and study of systems that learn from data, rather than following explicitly programmed instructions. ML is widely used in circumstances where the explicit program cannot be applied. For example, for navigating on Mars and routing on computer networks, there is no human expertise or the solutions change in time. A widely quoted and formal definition is "A computer program is said to learn from experience **E** with respect to some task **T** and some performance measure **P**, if its performance on **T**, as measured by **P**, improves with experience **E**" [1].

ML tasks can be classified as: supervised, unsupervised, semi-supervised, reinforcement and development learnings, etc. In supervised learning, the computer is presented with inputs and the corresponding desired outputs (labels). The target is to learn a general rule that connects inputs to outputs with the least error between given labels and predicted labels. In unsupervised learning [2], no labels are provided to the learning task, leaving it on its own to group similar inputs (clustering) [3]. Unsupervised learning is to seek hidden structure in unlabeled data. As to unsupervised learning, we cannot evaluate a potential solution since there is no error or reward signal which comes from the labeled data. Semi-supervised learning combines both labeled and unlabeled examples to generate an appropriate function or classifier. In reinforcement learning, a computer program interacts with a dynamic environment so that it has to execute a certain task without explicit instructions. Developmental learning is usually designed for robot learning. It generates its own sequences (also called as curriculum) of learning situations to acquire repertoires of novel

skills cumulatively through autonomous self-exploration and social interaction with human "teachers." We next give some representative approaches falling into these categories.

Many kinds of ML approaches were developed by the pioneer researchers, including artificial neural network (ANN), support vector machine (SVM), Bayesian networks (BN), clustering, representation learning, sparse dictionary learning, deep learning, AdaBoost, and so on. In this chapter, we give brief description of various ML methods (with emphasis in SVM and deep learning) that are relevant to both existing and future VQA, and their applications will be further investigated in the following chapters.

2.1 Artificial Neural Networks

An ANN (or "neural network" (NN)) [4] is a kind of learning algorithms with the inspiration from the structure and functional aspects of biological neural networks. Computation is structured based on an interconnected group of artificial neurons, and information is processed by using a connectionist approach. The NNs are designed to model complex relationships between inputs and outputs which are usually nonlinear. They try to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

2.2 Support Vector Machine

In ML family, SVM [5, 6] is a supervised learning algorithm, which is usually used for classification and regression analysis. Given a set of training samples with associated labels, an SVM classifier is learnt from this training set, and used to assign new examples into one category or the other. An SVM model is trained to find the hyperplanes that have the largest distance to the nearest training data point of any class, so that it can separate samples by a clear gap as wide as possible. Then, the new examples can be easily classified into one of the categories based on which side of the gap they fall on. The simplest form of SVM is a linear classifier. SVM can also perform a nonlinear classification by using the kernels which can convert their inputs into high-dimensional feature spaces so that a linear classifier can be used in the high-dimensional space.

Given training data $\mathscr{D}$ with n points as

$$\mathscr{D} = \left\{ (\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p, \; y_i \in \{-1, 1\} \right\}_{i=1}^n \tag{2.1}$$

where the y_i being either 1 or -1 indicates the class to which the point $\mathbf{x}_i$ belongs. Each $\mathbf{x}_i$ is a p-dimensional real vector. We intend to find the hyperplane that has the largest distance from the nearest points so that it can divide the points having $y_i = 1$

from those having $y_i = -1$ clearly. Any hyperplane can be represented by a set of points $\mathbf{x}$ that satisfy

$$\mathbf{w} \cdot \mathbf{x} - b = 0, \tag{2.2}$$

where $\cdot$ represents the dot product, the vector $\mathbf{w}$ is normal to the hyperplane, and the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$ is quantified by $b/\|\mathbf{w}\|$.

If the training data are linearly separable, there exist two hyperplanes which can separate the data points clearly without data points between them as shown in Fig. 2.1. The region bounded by these two hyperplanes is called "the margin." These two hyperplanes can be defined as

$$\mathbf{w} \cdot \mathbf{x} - b = 1, \tag{2.3}$$

and

$$\mathbf{w} \cdot \mathbf{x} - b = -1. \tag{2.4}$$

Observing Fig. 2.1, we know that the distance between these two hyperplanes is $2/\|\mathbf{w}\|$. The SVM training tries to maximize this distance, i.e., minimize $\|\mathbf{w}\|$. Since we also need to prevent data points from falling into the margin, the constraints
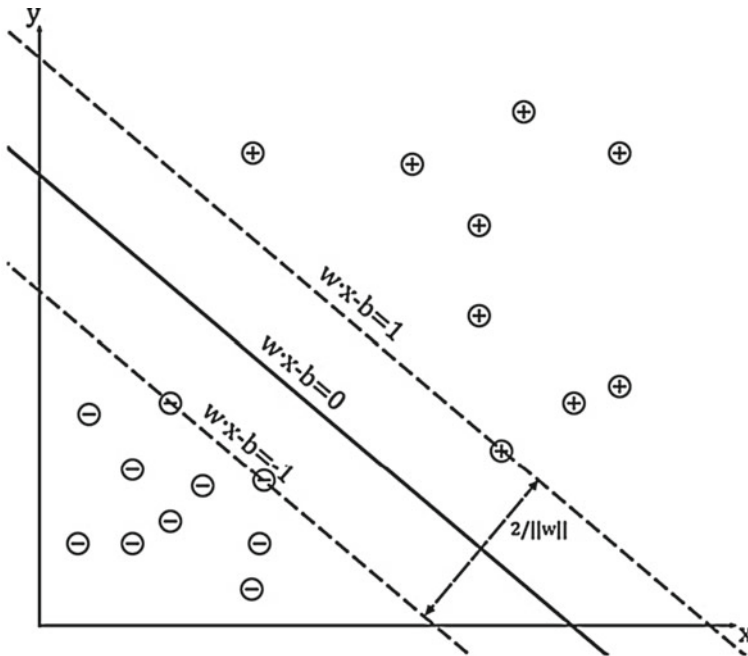

Fig. 2.1 Hyperplanes and margin for an SVM trained with samples from two classes

$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1$ and $\mathbf{w} \cdot \mathbf{x}_i - b \leq -1$ are imposed for $\{\mathbf{x}_i\}$ of the first class and the second class, respectively. These constraints can be rewritten as

$$y_i (\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1; \quad 1 \leq i \leq n. \tag{2.5}$$

Then, an optimization problem can be formulated as

$$\begin{aligned} &\min_{\mathbf{w}, b} \|\mathbf{w}\| \\ &s.t. \quad y_i (\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1; \quad 1 \leq i \leq n. \end{aligned} \tag{2.6}$$

2.2.1 Primal Form

By substituting $\|\mathbf{w}\|$ with $\frac{1}{2}\|\mathbf{w}\|^2$, a quadratic programming optimization problem is proposed as

$$\begin{aligned} &\min_{\mathbf{w}, b} \tfrac{1}{2}\|\mathbf{w}\|^2 \\ &s.t. \quad y_i (\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1; \quad 1 \leq i \leq n. \end{aligned} \tag{2.7}$$

Introducing Lagrange multipliers $\alpha_i, i = 1, \ldots, n$, (2.7) can be expressed as

$$\min_{\mathbf{w}, b} \max_{\alpha_i \geq 0} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i (\mathbf{w} \cdot \mathbf{x_i} - b) - 1] \right\}. \tag{2.8}$$

2.2.2 Dual Form

Let

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i (\mathbf{w} \cdot \mathbf{x_i} - b) - 1]. \tag{2.9}$$

By setting the derivations of $L(\mathbf{w}, b, \alpha)$ with respect to $\mathbf{w}$ and b to zero. We have

$$\begin{aligned} \nabla_{\mathbf{w}} L(\mathbf{w}, b, \alpha) &= \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \\ \frac{\partial}{\partial b} &= \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned} \tag{2.10}$$

which implies that

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}. \tag{2.11}$$

Plugging (2.11) back into (2.8), we get

$$\max_{\boldsymbol{\alpha}_i \geq 0} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j < \mathbf{x}_i, \mathbf{x}_j > \right\}. \tag{2.12}$$

This is the dual form of SVM, which indicates that the maximum-margin hyperplane and the associated classification task only concern the support vectors. In (2.12), only α_i are variables. By solving the dual form (2.12), α_i can be obtained. Then, $\mathbf{w}$ can be calculated from (2.11). Having found $\mathbf{w}$, since the hyperplane is given by $\mathbf{w} \cdot \mathbf{x_i} - b = 0$, it is also straightforward to find the optimal value for the intercept term b as

$$b = \frac{\max_{i:y_i = -1} \mathbf{w} \cdot \mathbf{x}_i + \min_{i:y_i = 1} \mathbf{w} \cdot \mathbf{x}_i}{2}. \tag{2.13}$$

SVM regression was proposed by Drucker et al. in [7]. It is also named as support vector regression (SVR). The SVR is modeled by the same way of support vector classification (as mentioned above). It only concerns a subset of the training data points, since the training data points that lie beyond the margin are excluded for establishing optimization function (cost function) in SVM/SVR. There are two general forms of SVR, i.e., ε-SVR, least square (LS)SVR proposed by Suykens and Vandewalle in [8].

In ε-SVR, the target is to find a function $f(x)$ that has at most ε deviation from the actually obtained targets y_i for all the training data.

Similar to SVM, the optimization function of ε-SVR is written as

$$
\begin{aligned}
&\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2 \\
&s.t. y_i - \mathbf{w} \cdot \mathbf{x}_i + b \leq \varepsilon; \\
&\qquad \mathbf{w} \cdot \mathbf{x}_i - b - y_i \leq \varepsilon; \qquad 1 \leq i \leq n.
\end{aligned} \tag{2.14}
$$

A tacit assumption in 2.14 is that such a function f exists that approximates all pairs (x_i, y_i) with ε precision, or in other words, that the convex optimization problem is feasible. However, this assumption is false, or we may want to allow for some errors. Analogously to the "soft margin" loss function in SVM [5], we introduce slack variables ξ_i, ξ_i^* to cope with otherwise infeasible constraints of the optimization problem 2.14. Then, a new optimization objective function with slack variables is derived as

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\begin{aligned} s.t. \, y_i - \mathbf{w}\cdot\mathbf{x}_i + b &\le \varepsilon + \xi_i; \\ \mathbf{w}\cdot\mathbf{x}_i - b - y_i &\le \varepsilon + \xi_i^*; \\ \xi_i, \xi_i^* &\ge 0; \\ 1 &\le i \le n. \end{aligned} \tag{2.15}$$

The constant $C > 0$ determines that trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated.

By introducing a dual set of variables, a Lagrange function from both the objective function (called primal objective function) and the corresponding constraints is constructed as

$$\begin{aligned} \min_{\mathbf{w},b} \max_{\boldsymbol{\alpha}\ge 0;\boldsymbol{\xi}\ge 0} &\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ &- \sum_{i=1}^{n}\alpha_i(\varepsilon + \xi_i - y_i + \mathbf{w}\cdot\mathbf{x}_i - b) \\ &- \sum_{i=1}^{n}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \mathbf{w}\cdot\mathbf{x}_i + b) - \sum_{i=1}^{n}(\eta_i\xi_i + \eta_i^*\xi_i^*) \end{aligned} \tag{2.16}$$

Following the procedure of SVM for dual form (Eqs. (2.9)–(2.12)), the dual form of ε-SVR is derived as

$$\max_{\alpha} \left\{ -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(\mathbf{x}_i \cdot \mathbf{x}_j) \right. $$
$$\left. -\varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n}y_i(\alpha_i - \alpha_i^*) \right\}$$

$$s.t. \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0; \, \alpha_i, \alpha_i^* \in [0, C] \tag{2.17}$$

2.3 Clustering

Cluster analysis is to organize observations into several subsets (called clusters) so that observations within the same cluster are similar, while those observations coming from different clusters are dissimilar with respect to some predefined criteria. Different clustering methods have the different assumptions upon the data structure; they

are often defined by some similarity metric and evaluated by internal compactness (similarity between members of the same cluster) and separation between different clusters. Other methods are established upon estimated density and graph connectivity. Clustering method belongs to unsupervised learning, which is used for statistical data analysis. K-means clustering is a classical representation of clustering methods, and serves as a prototype of the cluster. It partitions n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

2.4 Bayesian Networks

A Bayesian network (BN) [9], also known as belief network or directed acyclic graphical (DAG) model belongs to the family of probabilistic graphical models (GMs). These GMs are used to represent a set of random variables and their conditional independencies via a directed acyclic graph (DAG). Formally, a Bayesian network **B** is a DAG that represents a joint probability distribution over a set of random variable **V**. The network is defined by a pair $\mathbf{B} = <G, \Theta>$. G represents the DAG whose nodes $\{X_i, i = 1, \ldots, n\}$ represent random variables, and edges represent conditional dependencies between these variables.

2.5 Representation Learning

The success of ML algorithms generally depends on data representation. Although specific domain knowledge can be used to design representations, learning with generic priors can also be used. The learned representations are more powerful than the ones designing for the specific purposes with respect to their generalization abilities. Representation learning [10] is about learning representations of the data that make it easier to extract useful information when building classifiers or other predictors. It aims at discovering better representations of the data. The typical examples are principal components analysis [11, 12] and cluster analysis [13]. Representation learning algorithms can be either unsupervised or supervised, including autoencoders, dictionary learning, matrix factorization, restricted Boltzmann machines, and various forms of clustering.

2.6 Sparse Dictionary Learning

In sparse dictionary learning, the data are represented by a group of basis functions [14]. The coefficients are assumed to be sparse. Let x be a d-dimensional datum, D be a $d \times n$ matrix, where each column of D represents a basis function. r is the coefficient to represent x using D. Mathematically, sparse dictionary learning means

the following $x \approx D \times r$ where r is sparse. Generally speaking, n is assumed to be larger than d to allow the freedom for a sparse representation.

Sparse dictionary learning has been used in several circumstances. In classification, the problem is to group the unknown input data into the corresponding classes. Assume that we have already had a dictionary for each class. Then, we determine the class for a new input such that this new input is best sparsely represented by the corresponding dictionary. Sparse dictionary learning is also applicable to image denoising on the basis that a clean image path can be sparsely represented by an image dictionary, but the noise cannot be represented well [15].

2.7 AdaBoost

"Adaptive Boosting" (AdaBoost) [16, 17] is an ML meta-algorithm. It can be employed in conjunction with other types of learning algorithms to improve their performance. An AdaBoost learner is composed by several other learning algorithms which are called "weak learners." In ensemble of other learning algorithms, the outputs of these learning algorithms are combined into a weighted sum which represents the output of the boosted classifier. AdaBoost highlights the instances that are misclassified by these weak classifiers for ensemble. The shortcoming of AdaBoost lies in that it is sensitive to noisy data and outliers. However, it can be less susceptible to the overfitting problem than other learning algorithms. In AdaBoost, although the individual learner is weak, the final synthesized model can converge to a strong learner as long as the performance of each week learner is slightly better than random guessing.

AdaBoost refers to a particular method of training a boosted classifier. A boost classifier is a classifier in the form

$$F_T(x) = \sum_{t=1}^{T} f_t(x), \tag{2.18}$$

where f_t is a weak learner which takes an object x as input. f_t has a real-valued output which indicates the class of the object belonging to. The sign of the weak learner output identifies the predicted class of the object x. The absolute value of the week learner gives the confidence in that classification. In a similar way, the T-layer classifier would be positive if the sample is regarded to be in the positive class and negative otherwise.

Each weak learner produces an output, hypothesis $h(x_i)$, for each sample in the training set. At each iteration t, a weak learner is selected, and a coefficient α_t is assigned such that the sum training error E_t of the resulting t-stage boost classifier is minimized.

$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_t h(x_i)], \tag{2.19}$$

where $F_{t-1}(x)$ represents the boost classifier which has been built up to the previous stage of training, $E(F)$ is error function, and $f_t(x) = \alpha_t h(x)$ is the weak learner that is being included for addition to the final classifier.

2.8 Deep Learning

Deep learning is a kind of ML method, and it is used to learn the representations of data. An observation (e.g., an image) can be represented in many ways (e.g., a vector of pixels). Some representations are in favor of the tasks of learning from given samples. The associated research in this area attempts to figure out what are better representations and how to learn these representations.

Various deep learning architectures, such as convolutional neural networks (CNNs), deep neural networks (DNNs), and deep belief networks (DBNs) have been successfully applied to computer vision, natural language processing, automatic speech recognition, and music/audio signal recognition. They have shown the competitive performance on various tasks. "Deep learning" was initially raised in 1980 [18–20]. However, the time complexity to train the network was every expensive, preventing it from practical use [21]. Thus, it gave the way to another popular ML technique, SVM temporarily. Until mid of 2000, it once again drawn the interest of researchers after a publication by Hinton [22]. This publication investigated how a many-layered feedforward NN could be effectively pretrained one layer at a time, by treating each layer in turn as an unsupervised restricted Boltzmann machine, and then using supervised backpropagation for fine-tuning.

Advances in hardware are also an important reason for the renewed interest of deep learning. Particularly, the graphics processing units (GPUs) are highly suitable for the kind of number crunching, matrix/vector computations in neural networks. GPUs have been witnessed to speed up training processing dramatically [23, 24], making deep learning practical for general use.

For explaining the common architectures of deep learning, two basic models: DNN and DBN are described as follows.

2.8.1 Deep Neural Networks

A DNN is defined to be an ANN with at least one hidden layer of unit between the input and output layers. It can model complex nonlinear relationships well. The extra layers give a DNN-added levels of abstraction, thus increasing its modeling capability. A DNN is usually designed as a feedforward network. It also can be recurrent neural networks, e.g., the application of language modeling in [25]. A DNN can be

discriminatively trained by the standard backpropagation algorithm. The weights of neurons are updated via stochastic gradient descent as

$$\Delta\omega_{ij}(t+1) = \Delta\omega_{ij}(t) + \eta\frac{\partial C}{\partial\omega_{ij}} \tag{2.20}$$

where η represents the learning rate, C is the cost function. The cost function is chosen depending on the activation function and learning type (supervised, unsupervised, reinforcement, etc.). For example, when encountering a multiclass classification problem by using supervised learning, the activation function and cost function are usually the softmax function and cross-entropy function, respectively. The softmax function is defined as

$$p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)}, \tag{2.21}$$

where p_j represents the class probability, and x_j and x_k are the inputs to units j and k, respectively. The cross entropy is defined as

$$C = -\sum_j d_j \log(p_j), \tag{2.22}$$

where d_j is the target probability for output unit j, and p_j represents the probability output for unit j after applying the activation function [26].

The overfitting and computation time are two common issues in DNNs. DNNs are very prone to overfitting due to the added layers of abstraction, which allow them to model rare dependencies in the training data. Therefore, regularization methods such as weight decay (ℓ_2-regularization) or sparsity (ℓ_1-regularization) are used during the training process to prevent overfitting [27]. In [28], another regularization method, namely "dropout" regularization was applied to DNNs. In dropout, some units are randomly omitted from the hidden layers during training process, which could break the rare dependencies which may occur in the training data.

In comparison with other training methods, backpropagation and gradient descent are easier to be implemented, and tend to converge to better local optima. Thus, they have been the preferred for the training of DNNs. However, they are very computationally expensive, especially for training DNNs. There are many parameters in a DNN, such as the size (number of layers and number of units per layer), the learning rate, the initial weights, etc. Sweeping through all these parameters may not be feasible in many tasks due to the expensive time cost. Therefore, various "tricks" have been used to speed up computation, such as using mini-batching (computing the gradient on several training examples at once rather than individual examples) [29]. Since the matrix and vector computations can be well suited for GPUs, the GPU-facilitated processing contributes significantly for speeding up. However, it is hard to use large cluster machines for training DNNs, so better parallelizing training methods are undoubtedly desirable.

2.8.2 Deep Belief Network

A DBN is a probabilistic, generative model made up of multiple layers of hidden units [30]. It can be regarded as a composition of simple learning modules which make up each layer. A DBN can be used for generatively pretraining a DNN by using the learned weights from the DBN as the initial weights of the DNN. Backpropagation or other discriminative algorithms can then be employed to fine-tune these weights. This is particularly useful in situations with the limited number of training data since the initial weights are influential on the performance of the final model. Good initial weights benefit modeling capability and convergence of the fine tuning [31].

A DBN can be efficiently trained in an unsupervised, layer-by-layer manner. Each layer is typically made of restricted Boltzmann machines (RBM). An RBM is an undirected, generative energy-based model with an input layer and single hidden layer. The visible units of the input layer and the hidden units of the hidden layer are connected without the connections of visible–visible or hidden–hidden. The training method for RBMs was initially proposed by Hinton [32] for training Product of Expert (PoE) models. This method was named as contrastive divergence (CD) which provided an approximation to the maximum likelihood method [29, 33].

In training a single RBM, weight updates are performed with gradient ascent as

$$\Delta \omega_{ij}(t+1) = \omega_{ij}(t) + \eta \frac{\partial \log(p(v))}{\partial \omega_{ij}}, \tag{2.23}$$

where $p(v)$ is the probability of a visible vector, which is given by

$$p(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}, \tag{2.24}$$

where Z is used for normalizing, and $E(v, h)$ is the energy function assigned to the state of the network. A lower energy indicates that the network is in a more "desirable" configuration. The gradient $\partial \log(p(v))/\partial \omega_{ij}$ has the simple form

$$\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}, \tag{2.25}$$

where $\langle \cdot \rangle_p$ represent averages with respect to distribution p.

In the procedure of CD, we first initialize the visible units to a training vector. Then, we update the hidden units in parallel provided the visible units by

$$p(h_j = 1 \mid \mathbf{V}) = \sigma(b_j + \sum_i v_i \omega_{ij}), \tag{2.26}$$

where σ represents the sigmoid function and b_j is the bias of h_j. Next, the visible units are updated in parallel given the hidden units as

$$p(v_i = 1 \mid \mathbf{H}) = \sigma(a_i + \sum_j h_j \omega_{ij}), \tag{2.27}$$

where a_i is the bias of v_i. This step is called the "reconstruction" step. After that, we reupdate the hidden units in parallel given the reconstructed visible units as (2.26). Finally, the weights are updated as

$$\Delta \omega_{ij} \propto \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{reconstruction}}. \tag{2.28}$$

Once an RBM is trained, another RBM can be "stacked" atop of it to create a multilayer model. At each time, another RBM is stacked, the input visible layer is initialized to a training vector and values for the units in the already-trained RBM layers are assigned using the current weights and biases. The final layer of the already-trained layers are used as input to the new RBM. The new RBM is then trained with the procedure above such that this whole process can be repeated until some desired stopping criterion is satisfied. Although CD is a crude approximation of maximum likelihood that would be ideal for training RBM, it has been proved to be an effective method for training deep architectures [29].

References

1. Mitchell T (1997) Machine learning. McGraw Hill, New York. ISBN 0-07-042807-7
2. Duda RO, Hart PE, Stork DG (2009) Unsupervised learning and clustering, chapter 10 in pattern classification. Wiley, New York, p 571. ISBN 0-471-05669-3
3. Bishop CM (2006) Pattern recognition and machine learning. Springer, New York. ISBN 0-387-31073-8
4. Golovko V, Imada A (1990) Neural networks in artificial intelligence. Ellis Horwood Limited, Chichester. ISBN 0-13-612185-3
5. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273
6. Support Vector Machine (2015) In: Wikipedia, The free encyclopedia. http://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=654587935. Accessed 8 Apr 2015
7. Drucker H et al (1997) Support vector regression machines In: Advances in neural information processing systems 9, NIPS 1997. MIT Press, Cambridge, pp 155–161
8. Suykens JAK, Vandewalle J, Joos PL (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
9. Pearl J, Russel S (2001) Bayesian networks, report (R-277), November 2000, In: Arbib M (ed) Handbook of brain theory and neural networks. MIT Press, Cambridge, pp 157–160
10. Bengioy Y, Courville A, Vincent P (2014) Representation learning: a review and new perspectives. 1206.5538v3[cs.LG]. Accessed 23 April 2014
11. Jolliffe IT (2002) Principal component analysis. In: Series: springer series in statistics, 2nd ed. Springer, New York, XXIX, 487, p. 28 illus
12. Smith LI (2002) A tutorial on pricipal component analysis. http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf. Accessed Feb 2002
13. Coates A, YN Andrew (2012) Learning feature representations with k-means. In: Neural networks: tricks of the trade. Springer LNCS, Heidelberg (reloaded)
14. Kenneth K-D et al (2003) Dictionary learning algorithms for sparse representation. Neural Comput 15.2: 349–396

15. Aharon M, Elad M, Bruckstein A, Katz Y (2006) K-SVD: an algorithm for designing over-complete dictionaries for sparse representation. IEEE Trans Signal Process 54(11):4311–4322

16. Freund Y, Schapire RE (1999) A short introduction to boosting. In: Proceedings of the 16-th international joint conference on artificial intelligence, vol 2, pp 1401–1406

17. AdaBoost (2015) In: Wikipedia, The free encyclopedia. http://en.wikipedia.org/w/index.php?title=AdaBoost&oldid=647686369. Accessed 8 Apr 2015

18. Deep Learning (2015) In: Wikipedia, The free encyclopedia. http://en.wikipedia.org/w/index.php?title=Deep_learningoldid=655313266. Accessed 8 Apr 2015

19. Fukushima K (1980) Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 36:193–202

20. Werbos P (1974) Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University

21. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1:541–551

22. Hinton GE (2007) Learning multiple layers of representation. Trends Cogn Sci 11(10):428–434

23. Ciresan DC, Meier U, Gambardella LM, Schmidhuber J (2010) Deep big simple neural nets for handwritten digit recognition. Neural Comput 22:3207–3220

24. Raina R, Madhavan A, YN Andrew (2009) Large-scale deep unsupervised learning using graphics processors. Proceedings of 26th international conference on machine learning

25. Mikolov T, Karafiat M, Burget L, Cernnocky J, Khudanpur S (2010) Recurrent neural network based language model. In: Proceedings of NTERSPEECH 2010, 11th annual conference of the international speech communication association, Makuhari, Chiba, Japan, 26–30 Sept 2010

26. Hinton GE, Li D, Dong Y, Dahl GE et al (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process Mag 29(6):82–87

27. Bengio Y, Boulanger-Lewandowski N, Pascanu R (2013) Advances in optimizing recurrent networks. In: Proceedings of IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 8624–8628, May 2013

28. Dahl GE, Sainath TN, Hinton GE (2013) Improving deep neural networks for LVCSR using rectified linear units and dropout. In: Proceedings of IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 8609–8623, May 2013

29. Hinton GE (2010) A practical guide to training restricted Boltzmann machines. Technical report, UTML TR 2010–003. Universityof Toronto, Department of Computer Science

30. Hinton GE (2009) Deep belief networks. Scholarpedia 4(5):5947

31. Larochelle H, Erhan D, Courville A, Bergstra J, Bengio Y (2007) An empirical evaluation of deep architectures on problems with many factors of variation. In: Proceedings of 24th international conference machine learning, pp 473–480

32. Hinton GE (2002) Training product of experts by minimizing contrastive divergence. Neural Comput 14:1771–1800

33. Fischer A, Igel C (2014) Training restricted Boltzmann machines: an introduction. Pattern Recogn 47(1):25–39

Visual Quality Assessment by Machine Learning
Xu, L.; Lin, W.; Kuo, C.-C.J.
2015, XIV, 132 p. 19 illus., 16 illus. in color., Softcover
ISBN: 978-981-287-467-2