Pattern recognition

In machine learning, **pattern recognition** is the assignment of some sort of output value (or *label*) to a given input value (or *instance*), according to some specific algorithm. An example of pattern recognition is classification, which attempts to assign each input value to one of a given set of *classes* (for example, determine whether a given email is "spam" or "non-spam"). However, pattern recognition is a more general problem that encompasses other types of output as well. Other examples are regression, which assigns a real-valued output to each input; sequence labeling, which assigns a class to each member of a sequence of values (for example, part of speech tagging, which assigns a part of speech to each word in an input sentence); and parsing, which assigns a parse tree to an input sentence, describing the syntactic structure of the sentence.

Pattern recognition algorithms generally aim to provide a reasonable answer for all possible inputs and to do "fuzzy" matching of inputs. This is opposed to *pattern matching* algorithms, which look for exact matches in the input with pre-existing patterns. A common example of a pattern-matching algorithm is regular expression matching, which looks for patterns of a given sort in textual data and is included in the search capabilities of many text editors and word processors. In contrast to pattern recognition, pattern matching is generally not considered a type of machine learning, although pattern-matching algorithms (especially with fairly general, carefully tailored patterns) can sometimes succeed in providing similar-quality output to the sort provided by pattern-recognition algorithms.

Pattern recognition is studied in many fields, including psychology, ethology, cognitive science and computer science.

Overview

Pattern recognition is generally categorized according to the type of learning procedure used to generate the output value. *Supervised learning* assumes that a set of *training data* (the *training set*) has been provided, consisting of a set of instances that have been properly labeled by hand with the correct output. A learning procedure then generates a *model* that attempts to meet two sometimes conflicting objectives: Perform as well as possible on the training data, and generalize as well as possible to new data (usually, this means being as simple as possible, for some technical definition of "simple", in accordance with Occam's Razor). Unsupervised learning, on the other hand, assumes training data that has not been hand-labeled, and attempts to find inherent patterns in the data that can then be used to determine the correct output value for new data instances. A combination of the two that has recently been explored is semi-supervised learning, which uses a combination of labeled and unlabeled data (typically a small set of labeled data combined with a large amount of unlabeled data). Note that in cases of unsupervised learning, there may be no training data at all to speak of; in other words, the data to be labeled *is* the training data.

Note that sometimes different terms are used to describe the corresponding supervised and unsupervised learning procedures for the same type of output. For example, the unsupervised equivalent of classification is normally known as *clustering*, based on the common perception of the task as involving no training data to speak of, and of grouping the input data into *clusters* based on some inherent similarity measure (e.g. the distance between instances, considered as vectors in a multi-dimensional vector space), rather than assigning each input instance into one of a set of pre-defined classes. Note also that in some fields, the terminology is different: For example, in community ecology, the term "classification" is used to refer to what is commonly known as "clustering".

The piece of input data for which an output value is generated is formally termed an *instance*. The instance is formally described by a vector of *features*, which together constitute a description of all known characteristics of the instance. (These feature vectors can be seen as defining points in an appropriate multidimensional space, and methods for manipulating vectors in vector spaces can be correspondingly applied to them, such as computing the dot product or the angle between two vectors.) Typically, features are either categorical (also known as nominal, i.e. consisting of one of a set of unordered items, such as a gender of "male" or "female", or a blood type of "A", "B", "AB" or "O"), ordinal (consisting of one of a set of ordered items, e.g. "large", "medium" or "small"), integer-valued (e.g. a count of the number of occurrences of a particular word in an email) or real-valued (e.g. a measurement of blood pressure). Often, categorical and ordinal data are grouped together; likewise for integer-valued and real-valued data. Furthermore, many algorithms work only in terms of categorical data and require that real-valued or integer-valued data be *discretized* into groups (e.g. less than 5, between 5 and 10, or greater than 10).

Many common pattern recognition algorithms are *probabilistic* in nature, in that they use statistical inference to find the best label for a given instance. Unlike other algorithms, which simply output a "best" label, often times probabilistic algorithms also output a probability of the instance being described by the given label. In addition, many probabilistic algorithms output a list of the *N*-best labels with associated probabilities, for some value of *N*, instead of simply a single best label. When the number of possible labels is fairly small (e.g. in the case of classification), *N* may be set so that the probability of all possible labels is output. Probabilistic algorithms have many advantages over non-probabilistic algorithms:

- They output a confidence value associated with their choice. (Note that some other algorithms may also output confidence values, but in general, only for probabilistic algorithms is this value mathematically grounded in probability theory. Non-probabilistic confidence values can in general not be given any specific meaning, and only used to compare against other confidence values output by the same algorithm.)
- Correspondingly, they can *abstain* when the confidence of choosing any particular output is too low.
- Because of the probabilities output, probabilistic pattern-recognition algorithms can be more effectively incorporated into larger machine-learning tasks, in a way that partially or completely avoids the problem of *error propagation*.

Techniques to transform the raw feature vectors are sometimes used prior to application of the pattern-matching algorithm. For example, feature extraction algorithms attempt to reduce a large-dimensionality feature vector into a smaller-dimensionality vector that is easier to work with and encodes less redundancy, using mathematical techniques such as principal components analysis (PCA). Feature selection algorithms, attempt to directly prune out redundant or irrelevant features. The distinction between the two is that the resulting features after feature extraction has taken place are of a different sort than the original features and may not easily be interpretable, while the features left after feature selection are simply a subset of the original features.

Problem statement (supervised version)

Formally, the problem of supervised pattern recognition can be stated as follows: Given an unknown function $g:\mathcal{X}\rightarrow\mathcal{Y}$ (the *ground truth*) that maps input instances $\boldsymbol{x}\in\mathcal{X}$ to output labels $y\in\mathcal{Y}$, along with training data $\mathbf{D}=\{(\boldsymbol{x}_1,y_1),\dots,(\boldsymbol{x}_n,y_n)\}$ assumed to represent accurate examples of the mapping, produce a function $h:\mathcal{X}\rightarrow\mathcal{Y}$ that approximates as closely as possible the correct mapping g. (For example, if the problem is filtering spam, then $\boldsymbol{x}_i$ is some representation of an email and y is either "spam" or "non-spam"). In order for this to be a well-defined problem, "approximates as closely as possible" needs to be defined rigorously. In decision theory, this is defined by specifying a loss function that assigns a specific value to to "loss" resulting from producing an incorrect label. The goal then is to minimize the expected loss, with the expectation taken over the probability distribution of $\mathcal{X}$. In practice, neither the distribution of $\mathcal{X}$ nor the ground truth function $g:\mathcal{X}\rightarrow\mathcal{Y}$ are known exactly, but can be computed only empirically by collecting a large number of samples of $\mathcal{X}$ and hand-labeling them using the correct value of $\mathcal{Y}$ (a time-consuming process, which is typically the limiting factor in the amount of data of this sort that can be collected). The particular loss function depends on the type of label being predicted. For example, in the case of classification, the simple zero-one loss function is often sufficient. This corresponds simply to assigning a loss of 1 to any incorrect labeling and is equivalent to computing the accuracy of the classification procedure over the set of test data (i.e. counting up the fraction of instances that the learned function $h:\mathcal{X}\rightarrow\mathcal{Y}$ labels correctly. The goal of the learning procedure is to maximize this test accuracy on a "typical" test set.

For a probabilistic pattern recognizer, the problem is instead to estimate the probability of each possible output label given a particular input instance, i.e. to estimate a function of the form

$$p(\text{label}|\boldsymbol{x},\boldsymbol{\theta}) = f\left(\boldsymbol{x};\boldsymbol{\theta}\right)$$

where the feature vector input is $\boldsymbol{x}$, and the function *f* is typically parameterized by some parameters $\boldsymbol{\theta}$. In a discriminative approach to the problem, *f* is estimated directly. In a generative approach, however, the inverse probability $p(\boldsymbol{x}|\text{label})$ is instead estimated and combined with the prior probability $p(\text{label}|\boldsymbol{\theta})$ using Bayes' rule, as follows:

$$p(\text{label}|\boldsymbol{x},\boldsymbol{\theta}) = \frac{p(\boldsymbol{x}|\text{label}) p(\text{label}|\boldsymbol{\theta})}{\sum_{L \in \text{all labels}} p(\boldsymbol{x}|L) p(L|\boldsymbol{\theta})}$$

When the labels are continuously distributed (e.g. in regression analysis), the denominator involves integration rather than summation:

$$p(\text{label}|\boldsymbol{x},\boldsymbol{\theta}) = \frac{p(\boldsymbol{x}|\text{label}) p(\text{label}|\boldsymbol{\theta})}{\int_{L \in \text{all labels}} p(\boldsymbol{x}|L) p(L|\boldsymbol{\theta}) \operatorname{d}L}$$

The value of $\boldsymbol{\theta}$ is typically learned using maximum a posteriori (MAP) estimation. This finds the best value that simultaneously meets two conflicting objects: To perform as well as possible on the training data and to find the simplest possible model. Essentially, this combines maximum likelihood estimation with a regularization procedure that favors simpler models over more complex models. In a Bayesian context, the regularization procedure can be viewed as placing a prior probability $p(\boldsymbol{\theta})$ on different values of $\boldsymbol{\theta}$. Mathematically:

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathbf{D})$$

where $\boldsymbol{\theta}^*$ is the value used for $\boldsymbol{\theta}$ in the subsequent evaluation procedure, and $p(\boldsymbol{\theta}|\mathbf{D})$, the posterior probability of $\boldsymbol{\theta}$, is given by

$$p(\boldsymbol{\theta}|\mathbf{D}) = \left[\prod_{i=1}^n p(y_i|\boldsymbol{x}_i,\boldsymbol{\theta}) \right] p(\boldsymbol{\theta})$$

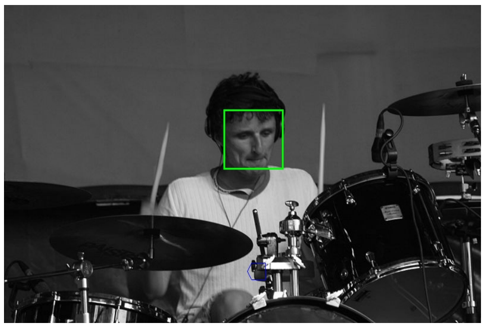
In the Bayesian approach to this problem, instead of choosing a single parameter vector $\boldsymbol{\theta}^*$, the probability of a given label for a new instance $\boldsymbol{x}$ is computed by integrating over all possible values of $\boldsymbol{\theta}$, weighted according to the posterior probability:

$$p(\text{label}|\boldsymbol{x}) = \int p(\text{label}|\boldsymbol{x},\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{D}) \operatorname{d}\boldsymbol{\theta}$$

Uses

Within medical science, pattern recognition is the basis for computer-aided diagnosis (CAD) systems. CAD describes a procedure that supports the doctor's interpretations and findings.

Typical applications are automatic speech recognition, classification of text into several categories (e.g. spam/non-spam email messages), the automatic recognition of handwritten postal codes on postal envelopes, or the automatic recognition of images of human faces. The last two examples form the subtopic image analysis of pattern recognition that deals with digital images as input to pattern recognition systems.[1] [2]


The face was automatically detected by special software.

Algorithms

Algorithms for pattern recognition depend on the type of label output, on whether learning is supervised or unsupervised, and on whether the algorithm is statistical or non-statistical in nature. Statistical algorithms can further be categorized as generative or discriminative.

Classification algorithms (supervised algorithms predicting categorical labels)

- Maximum entropy classifier (aka logistic regression, multinomial logistic regression): Note that logistic regression is an algorithm for classification, despite its name. (The name comes from the fact that logistic regression uses an extension of a linear regression model to model the probability of an input being in a particular class.)
- Naive Bayes classifier
- Decision trees, decision lists
- Support vector machines
- K-nearest-neighbor algorithms
- Perceptrons
- Neural networks (multi-level perceptrons)

Clustering algorithms (unsupervised algorithms predicting categorical labels)

- Categorical mixture models
- K-means clustering
- Hierarchical clustering (agglomerative or divisive)
- Kernel principal component analysis (Kernel PCA)

Regression algorithms (predicting real-valued labels)

Supervised:

- Linear regression and extensions
- Neural networks
- Gaussian process regression (kriging)

Unsupervised:

- Principal components analysis (PCA)
- Independent component analysis (ICA)

Categorical sequence labeling algorithms (predicting sequences of categorical labels)

Supervised:

- Hidden Markov models (HMMs)
- Maximum entropy Markov models (MEMMs)
- Conditional random fields (CRFs)

Unsupervised:

- Hidden Markov models (HMMs)

Real-valued sequence labeling algorithms (predicting sequences of real-valued labels)

Supervised (?):

- Kalman filters
- Particle filters

Unsupervised:

- ???

Parsing algorithms (predicting tree structured labels)

Supervised and unsupervised:

- Probabilistic context free grammars (PCFGs)

General algorithms for predicting arbitrarily-structured labels

- Bayesian networks
- Markov random fields

Ensemble learning algorithms (supervised meta-algorithms for combining multiple learning algorithms together)

- Bootstrap aggregating ("bagging")
- Boosting
- Ensemble averaging
- Mixture of experts, hierarchical mixture of experts

See also

- Compound term processing
- Computer-aided diagnosis
- Data mining
- List of numerical analysis software
- List of numerical libraries
- Machine learning
- Neocognitron
- Predictive analytics
- Prior knowledge for pattern recognition
- Template matching

References

This article was originally based on material from the Free On-line Dictionary of Computing, which is licensed under the GFDL.

[1] Richard O. Duda, Peter E. Hart, David G. Stork (2001) *Pattern classification* (2nd edition), Wiley, New York, ISBN 0-471-05669-3
[2] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*, Wiley, ISBN 978-0-470-51706-2, 2009 *(* (http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470517069.html) *TM book)*

Further reading

- Fukunaga, Keinosuke (1990). *Introduction to Statistical Pattern Recognition* (2nd ed.). Boston: Academic Press. ISBN 0-12-269851-7.
- Bishop, Christopher (2006). *Pattern Recognition and Machine Learning*. Berlin: Springer. ISBN 0-387-31073-8.
- Koutroumbas, Konstantinos; Theodoridis, Sergios (2008). *Pattern Recognition* (4th ed.). Boston: Academic Press. ISBN 1-59749-272-8.
- Bhagat, Phiroz (2005). *Pattern Recognition in Industry*. Amsterdam: Elsevier. ISBN 0-08-044538-1.
- Hornegger, Joachim; Paulus, Dietrich W. R. (1999). *Applied Pattern Recognition: A Practical Introduction to Image and Speech Processing in C++* (2nd ed.). San Francisco: Morgan Kaufmann Publishers. ISBN 3-528-15558-2.
- Schuermann, Juergen (1996). *Pattern Classification: A Unified View of Statistical and Neural Approaches*. New York: Wiley. ISBN 0-471-13534-8.
- Godfried T. Toussaint, ed (1988). *Computational Morphology*. Amsterdam: North-Holland Publishing Company.
- Kulikowski, Casimir A.; Weiss, Sholom M. (1991). *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Machine Learning. San Francisco: Morgan Kaufmann Publishers. ISBN 1-55860-065-5.

External links

- Introduction to Pattern Recognition using Mathematica (http://demonstrations.wolfram.com/PatternRecognitionPrimer)
- More on Pattern Recognition using Mathematica (http://demonstrations.wolfram.com/PatternRecognitionPrimerII)
- The International Association for Pattern Recognition (http://www.iapr.org)
- List of Pattern Recognition web sites (http://cgm.cs.mcgill.ca/~godfried/teaching/pr-web.html)
- Journal of Pattern Recognition Research (http://www.jprr.org)
- Pattern Recognition Info (http://www.docentes.unal.edu.co/morozcoa/docs/pr.php)
- Pattern Recognition (http://www.sciencedirect.com/science/journal/00313203) (Journal of the Pattern Recognition Society)
- International Journal of Pattern Recognition and Artificial Intelligence (http://www.worldscinet.com/ijprai/mkt/archive.shtml)

Article Sources and Contributors

Pattern recognition *Source*: http://en.wikipedia.org/w/index.php?oldid=391785728 *Contributors*: AnAj, Anaxial, Antaeus Feldspar, B4hand, Basilwhite, Benwing, Bluemoose, Cmbishop, CommodiCast, Compvision, Crimson30, Cybercobra, Dancter, Dappe, David Eppstein, DavidWBrooks, Denoir, Devantheryv, Dfletter, Dysprosia, Eco84, Electron9, ExDxV, Fastilysock, Feureau, Finlay McWalter, Fuzzform, Gene s, GerryWolff, Gioto, GodfriedToussaint, Gtfjbl, Guaka, Hike395, Hobbes, Hu12, Intgr, Ireny, Itss ieee, Izik, Jbmurray, JonHarder, Justin W Smith, Kamoshirenai, Karada, KellyCoinGuy, Kku, Krabeert, Kraftlos, Lauerfab, Lexor, LiDaobing, Lotje, M4gnum0n, Masroor, Matthias Röder, Maurice Carbonaro, Mav, Mcsee, Mebden, Melcombe, Memming, Michael Hardy, Mietchen, Movado73, MrOllie, Msm, Mycotoxin, Nandesuka, Novum, Offput, Ojigiri, Pagingmrherman, Patrick, Peak, Peptidefarmer, Pir, Pizza1512, Pmbhagat, Predictor, Quiddity, RJASE1, Radshashi, Ramir, Recury, Redgecko, Rethunk, Rsrikanth05, Ruud Koot, Rvencio, Sebastjanmm, Shirik, Shoejar, Sludge, Storm Rider, Tabor, Tanja-Else, The Anome, TheRingess, Trebor, Tuxide, Utcursch, Vdm, Who-is-me, Wikipedie, Wprlh, Yosri, Zeno Gantner, Іванко1, 122 anonymous edits

Image Sources, Licenses and Contributors

Image:800px-Cool Kids of Death Off Festival p 146-face selected.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:800px-Cool_Kids_of_Death_Off_Festival_p_146-face_selected.jpg *License*: GNU Free Documentation License *Contributors*: User:MaGIc2laNTern, User:Przykuta

License