Support vector machine

From Wikipedia, the free encyclopedia
Jump to: navigation, search

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Their common factor is the use of a technique known as the "kernel trick" to apply linear classification techniques to non-linear classification problems.

Contents

[hide]

[edit]

Linear classification

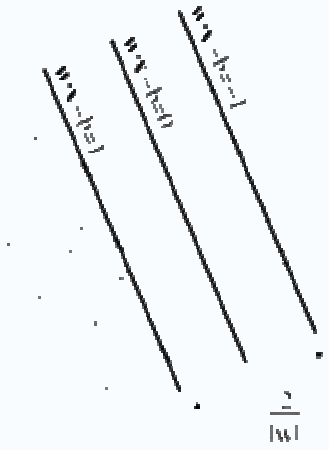
[edit]

Motivation

Suppose we want to classify some data points into two classes. Often we are interested in classifying data as part of a machine-learning process. These data points may not necessarily be points in $\mathbf{R}^2$ but may be multidimensional $\mathbf{R}^p$ (statistics notation) or $\mathbf{R}^n$ (computer science notation) points. We are interested in whether we can separate them by a hyperplane. As we examine a hyperplane, this form of classification is known as linear classification. We also want to choose a hyperplane that separates the data points "neatly", with maximum distance to the closest data point from both classes -- this distance is called the *margin*. We desire this property since if we add another data point to the points we already have, we can more accurately classify the new point since the separation between the two classes is greater. Now, if such a hyperplane exists, the hyperplane is clearly of interest and is known as the *maximum-margin hyperplane* or the *optimal hyperplane*, as are the vectors that are closest to this hyperplane, which are called the *support vectors*.

[edit]

Formalization



Maximum-margin hyperplanes for a SVM trained with samples from two classes. Samples along the hyperplanes are called the support vectors.

We consider data points of the form: $\{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \ldots, (\mathbf{x}_n, c_n)\}$ where the c_i is either 1 or −1 -- this constant denotes the class to which the point $\mathbf{x}_i$ belongs. Each $\mathbf{x}_i$ is a *p* − (statistics notation), or *n* − (computer science notation) dimensional vector of scaled [0,1] or [-1,1] values. The scaling is important to guard against variables (attributes) with larger variance that might otherwise dominate the classification. We can view this as *training data*, which denotes the correct classification which we would like the SVM to eventually distinguish, by means of the dividing hyperplane, which takes the form

$$\mathbf{w} \cdot \mathbf{x} - b = 0.$$

As we are interested in the maximum margin, we are interested in the support vectors and the parallel hyperplanes (to the optimal hyperplane) closest to these support vectors in either class. It can be shown that these parallel hyperplanes can be described by equations

$$\mathbf{w} \cdot \mathbf{x} - b = 1, \qquad (1)$$
$$\mathbf{w} \cdot \mathbf{x} - b = -1. \qquad (2)$$

We would like these hyperplanes to maximize the distance from the dividing hyperplane and to have no data points between them. By using geometry, we find the distance between the hyperplanes being 2/|***w***|, so we want to minimize |***w***|. To exclude data points, we need to ensure that for all i either

$$\mathbf{w} \cdot \mathbf{x_i} - b \geq 1 \quad \text{or}$$
$$\mathbf{w} \cdot \mathbf{x_i} - b \leq -1$$

This can be rewritten as:

$$c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 \quad 1 \leq i \leq n. \qquad (3)$$

The problem now is to minimize |***w***| subject to the constraint (3). This is a quadratic programming (QP) optimization problem.

After the SVM has been trained, it can be used to classify unseen 'test' data. This is achieved using the following decision rule;

$$\hat{c} = \begin{cases} 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ -1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \end{cases}$$

Writing the classification rule in its dual form reveals that classification is only a function of the Support vectors, i.e. the training data that lie on the margin.

[edit]

Further theory

The use of the maximum-margin hyperplane is motivated by Vapnik Chervonenkis theory, which provides a probabilistic test error bound that is minimized when the margin is maximized. However the utility of this theoretical analysis is sometimes questioned given the large slack associated with these bounds: the bounds often predict more than 100% error rates.

The parameters of the maximum-margin hyperplane are derived by solving the optimization. There exist several specialized algorithms for quickly solving the QP problem that arises from SVMs. The most common method for solving the QP problem is Platt's SMO algorithm.

[edit]

Non-linear classification

The original optimal hyperplane algorithm proposed by Vladimir Vapnik in 1963 was a linear classifier. However, in 1992, Bernhard Boser, Isabelle Guyon and Vapnik suggested a way to create non-linear classifiers by applying the kernel trick (originally proposed by Aizerman) to maximum-margin hyperplanes. The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in the transformed feature space. The transformation may be non-linear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space it may be non-linear in the original input space.

If the kernel used is a Gaussian radial basis function, the corresponding feature space is a Hilbert space of infinite dimension. Maximum margin classifiers are well regularized, so the infinite dimension does not spoil the results. Some common kernels include,

- Polynomial (homogeneous): $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^d$
- Polynomial (inhomogeneous): $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^d$
- Radial Basis Function: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$, for $\gamma > 0$
- Gaussian Radial_basis_function: $k(\mathbf{x}, \mathbf{x}') = \exp(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2})$

- Sigmoid: $k(\mathbf{x}, \mathbf{x}') = \tanh(\kappa \mathbf{x} \cdot \mathbf{x}' + c)$, for some (not every) $\kappa > 0$ and $c < 0$

[edit]

Soft margin

In 1995, Corinna Cortes and Vapnik suggested a modified maximum margin idea that allows for mislabeled examples. If there exists no hyperplane that can split the "yes" and "no" examples, the *Soft Margin* method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. This work popularized the expression *Support Vector Machine* or *SVM*. This method introduces slack variables and the equation (3) now transforms to

$$c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 - \xi_i \quad 1 \leq i \leq n \qquad (4)$$

and the optimization problem becomes

$$\min ||w||^2 + C \sum_i \xi_i \quad \textbf{such that } c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 - \xi_i \quad 1 \leq i \leq n$$

This constraint in (4) along with the objective of minimizing $|\boldsymbol{w}|$ can be solved using Lagrange multipliers or setting up a dual optimization problem to eliminate the slack variable.

[edit]

Regression

A version of a SVM for regression was proposed in 1996 by Vapnik, Harris Drucker, Chris Burges, Linda Kaufman and Alex Smola. This method is called support vector regression (SVR). The model produced by support vector classification (as described above) only depends on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin. Analogously, the model produced by SVR only depends on a subset of the training data, because the cost function for building the model ignores any training data that is close (within a threshold ε) to the model prediction.

[edit]

See also

- Predictive analytics

[edit]

References

- B. E. Boser, I. M. Guyon, and V. N. Vapnik. *A training algorithm for optimal margin classifiers*. In D. Haussler, editor, 5th Annual ACM Workshop on COLT, pages 144-152, Pittsburgh, PA, 1992. ACM Press.
- Corinna Cortes and V. Vapnik, "Support-Vector Networks, *Machine Learning, 20, 1995. [1]*
- Christopher J. C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition". Data Mining and Knowledge Discovery 2:121 - 167, 1998 *(Also available at CiteSeer: [2])*
- Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000. ISBN 0-521-78019-5 *([3] SVM Book)*
- Harris Drucker, Chris J.C. Burges, Linda Kaufman, Alex Smola and Vladimir Vapnik (1997). "Support Vector Regression Machines". *Advances in Neural Information Processing Systems 9, NIPS 1996*, 155-161, MIT Press.
- Huang T.-M., Kecman V., Kopriva I. (2006), Kernel Based Algorithms for Mining Huge Data Sets, Supervised, Semi-supervised, and Unsupervised Learning, Springer-Verlag, Berlin, Heidelberg, 260 pp. 96 illus., Hardcover, ISBN 3-540-31681-7[4]
- Vojislav Kecman: "Learning and Soft Computing - Support Vector Machines, Neural Networks, Fuzzy Logic Systems", The MIT Press, Cambridge, MA, 2001.[5]
- Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Aleksandr Mylläri and Tapio Salakoski. Improving the Performance of Bayesian and Support Vector Classifiers in Word Sense Disambiguation using Positional Information. In Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05), Jun 2005.
- Bernhard Schölkopf and A. J. Smola: *Learning with Kernels*. MIT Press, Cambridge, MA, 2002. *(Partly available on line: [6].)* ISBN 0-262-19475-9
- Bernhard Schölkopf, Christopher J.C. Burges, and Alexander J. Smola (editors). "Advances in Kernel Methods: Support Vector Learning". MIT Press, Cambridge, MA, 1999. ISBN 0-262-19416-3. [7]
- John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004. ISBN 0-521-81397-2 *([8] Kernel Methods Book)*
- P.J. Tan and D.L. Dowe (2004), MML Inference of Oblique Decision Trees, Lecture Notes in Artificial Intelligence (LNAI) 3339, Springer-Verlag, pp1082-1088. Links require password. (This paper uses minimum message length (MML) and actually incorporates probabilistic support vector machines in the leaves of decision trees.)
- Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1999. ISBN 0-387-98780-0

[edit]

External links

- www.pascal-network.org *(EU Funded Network on Pattern Analysis, Statistical Modelling and Computational Learning)*
- www.kernel-machines.org *(general information and collection of research papers)*
- www.kernel-methods.net *(News, Links, Code related to Kernel methods - Academic Site)*
- www.support-vector.net *(News, Links, Code related to Support Vector Machines - Academic Site)*

- www.support-vector-machines.org *(Literature, Review, Software, Links related to Support Vector Machines - Academic Site)*
- www.support-vector.ws *(Free educational MATLAB based software for SVMs, NN and FL , Links, Publications downloads, Semisupervised learning software SemiL, Links)*

[edit]

Software

- Lush -- an Lisp-like interpreted/compiled language with C/C++/Fortran interfaces that has packages to interface to a number of different SVM implementations. Interfaces to LASVM, LIBSVM, mySVM, SVQP, SVQP2 (SVQP3 in future) are available. Leverage these against Lush's other interfaces to machine learning, hidden markov models, numerical libraries (LAPACK, BLAS, GSL), and builtin vector/matrix/tensor engine.
- SVMlight -- a popular implementation of the SVM algorithm by Thorsten Joachims; it can be used to solve classification, regression and ranking problems.
- LIBSVM -- A Library for Support Vector Machines, Chih-Chung Chang and Chih-Jen Lin
- YALE -- a powerful machine learning toolbox containing wrappers for SVMLight, LibSVM, and MySVM in addition to many evaluation and preprocessing methods.
- LS-SVMLab - Matlab/C SVM toolbox - well-documented, many features
- Gist -- implementation of the SVM algorithm with feature selection.
- Weka -- a machine learning toolkit that includes an implementation of an SVM classifier; Weka can be used both interactively though a graphical interface or as a software library. (One of them is called "SMO". In the GUI Weka explorer, it is under the "classify" tab if you "Choose" an algorithm.)
- OSU SVM - Matlab implementation based on LIBSVM
- Torch - C++ machine learning library with SVM
- Shogun - Large Scale Machine Learning Toolbox with interfaces to Octave, Matlab, Python, R
- Spider - Machine learning library for Matlab
- e1071 - Machine learning library for R
- SimpleSVM - SimpleSVM toolbox for Matlab
- SVM and Kernel Methods Matlab Toolbox
- PCP -- C program for supervised pattern classification. Includes LIBSVM wrapper.
- TinySVM -- a small SVM implementation, written in C++

[edit]

Interactive SVM applications

- ECLAT classification of Expressed Sequence Tag (EST) from mixed EST pools using codon usage
- EST3 classification of Expressed Sequence Tag (EST) from mixed EST pools using nucleotide triples

Categories: Machine learning | Classification algorithms | Ensemble learning