Support-vector machine

In machine learning, **support-vector machines** (**SVMs**, also **support-vector networks**[1]) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The **support-vector clustering**[2] algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

Contents

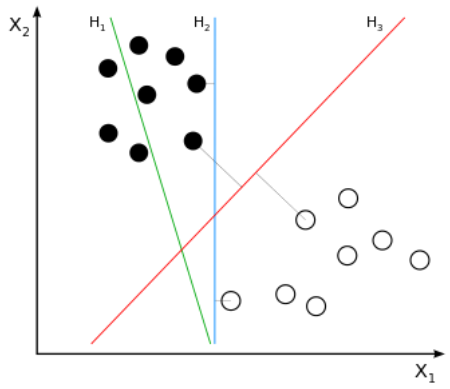
Motivation

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a *new* data point will be in. In the case of support-vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p-1)$-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the *maximum-margin hyperplane* and the linear classifier it defines is known as a *maximum-margin classifier*; or equivalently, the *perceptron of optimal stability*.

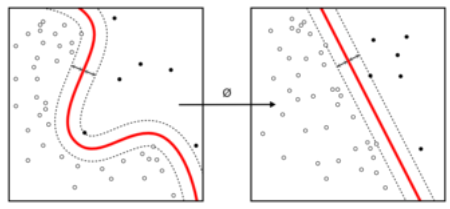
Definition

More formally, a support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection.[3] Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.[4]

Whereas the original problem may be stated in a finite-dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products of pairs of input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x, y)$ selected to suit the problem.[5] The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vectors is an orthogonal (and thus minimal) set of vectors that defines a hyperplane. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters α_i of images of feature vectors x_i that occur in the data base. With this choice of a hyperplane, the points x in the feature space that are mapped into the hyperplane are defined by the relation $\sum_i \alpha_i k(x_i, x) = \mathbf{constant.}$ Note that if $k(x, y)$ becomes small as y grows further away from x, each term in the sum measures the degree of closeness of the test point x to the corresponding data base point x_i. In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points x mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets that are not convex at all in the original space.



H_1 does not separate the classes. H_2 does, but only with a small margin. H_3 separates them with the maximal margin.


Kernel machine

Applications

SVMs can be used to solve various real-world problems:

- SVMs are helpful in text and hypertext categorization, as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings. Some methods for shallow semantic parsing are based on support vector machines.[6]
- Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true for image segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik.[7][8]
- Hand-written characters can be recognized using SVM.[9]
- The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models.[10][11] Support-vector machine weights have also been used to interpret SVM models in the past.[12] Posthoc interpretation of support-vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.

History

The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes.[13] The current standard incarnation (soft margin) was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995.[1]

Linear SVM

We are given a training dataset of n points of the form

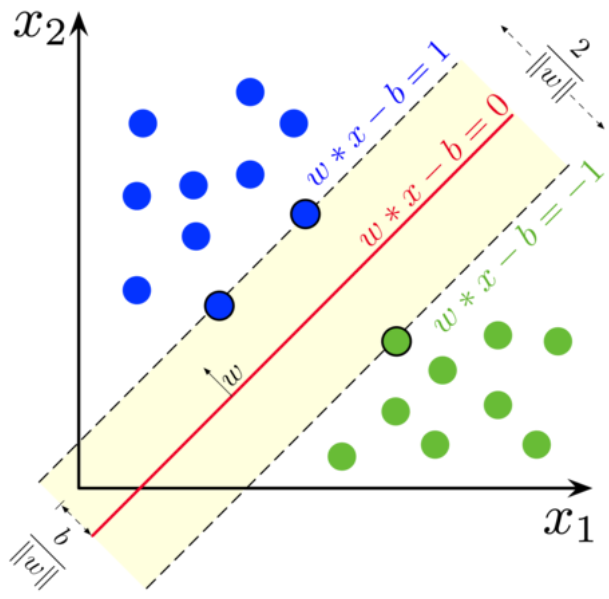
$$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n),$$

where the y_i are either 1 or −1, each indicating the class to which the point $\vec{x}_i$ belongs. Each $\vec{x}_i$ is a p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points $\vec{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\vec{x}_i$ from either group is maximized.

Any hyperplane can be written as the set of points $\vec{x}$ satisfying

$$\vec{w} \cdot \vec{x} - b = 0,$$

where $\vec{w}$ is the (not necessarily normalized) normal vector to the hyperplane. This is much like Hesse normal form, except that $\vec{w}$ is not necessarily a unit vector. The parameter $\frac{b}{\|\vec{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\vec{w}$.



Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

Hard-margin

If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible. The region bounded by these two hyperplanes is called the "margin", and the maximum-margin hyperplane is the hyperplane that lies halfway between them. With a normalized or standardized dataset, these hyperplanes can be described by the equations

$\vec{w} \cdot \vec{x} - b = 1$ (anything on or above this boundary is of one class, with label 1)

and

$\vec{w} \cdot \vec{x} - b = -1$ (anything on or below this boundary is of the other class, with label −1).

Geometrically, the distance between these two hyperplanes is $\frac{2}{\|\vec{w}\|}$[14], so to maximize the distance between the planes we want to minimize $\|\vec{w}\|$. The distance is computed using the distance from a point to a plane equation. We also have to prevent data points from falling into the margin, we add the following constraint: for each i either

$\vec{w} \cdot \vec{x}_i - b \geq 1$, if $y_i = 1$,

or

$$\vec{w} \cdot \vec{x}_i - b \leq -1, \text{ if } y_i = -1.$$

These constraints state that each data point must lie on the correct side of the margin.

This can be rewritten as

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n. \qquad (1)$$

We can put this together to get the optimization problem:

"Minimize $\|\vec{w}\|$ subject to $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$ for $i = 1, \ldots, n$."

The $\vec{w}$ and b that solve this problem determine our classifier, $\vec{x} \mapsto \operatorname{sgn}(\vec{w} \cdot \vec{x} - b)$.

An important consequence of this geometric description is that the max-margin hyperplane is completely determined by those $\vec{x}_i$ that lie nearest to it. These $\vec{x}_i$ are called *support vectors*.

Soft-margin

To extend SVM to cases in which the data are not linearly separable, we introduce the *hinge loss* function,

$$\max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right).$$

Note that y_i is the *i*-th target (i.e., in this case, 1 or −1), and $\vec{w} \cdot \vec{x}_i - b$ is the current output.

This function is zero if the constraint in (1) is satisfied, in other words, if $\vec{x}_i$ lies on the correct side of the margin. For data on the wrong side of the margin, the function's value is proportional to the distance from the margin.

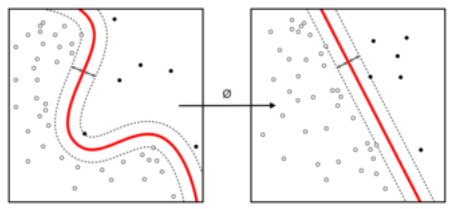
We then wish to minimize

$$\left[\frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right)\right] + \lambda\|\vec{w}\|^2,$$

where the parameter λ determines the trade-off between increasing the margin size and ensuring that the $\vec{x}_i$ lie on the correct side of the margin. Thus, for sufficiently small values of λ, the second term in the loss function will become negligible, hence, it will behave similar to the hard-margin SVM, if the input data are linearly classifiable, but will still learn if a classification rule is viable or not.

Nonlinear classification

The original maximum-margin hyperplane algorithm proposed by Vapnik in 1963 constructed a linear classifier. However, in 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick (originally proposed by Aizerman et al.[15]) to maximum-margin hyperplanes.[13] The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high-dimensional; although the classifier is a hyperplane in the transformed feature space, it may be nonlinear in the original input space.


Kernel machine

It is noteworthy that working in a higher-dimensional feature space increases the generalization error of support-vector machines, although given enough samples the algorithm still performs well.[16]

Some common kernels include:

- Polynomial (homogeneous): $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} \cdot \overrightarrow{x_j})^d$.
- Polynomial (inhomogeneous): $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} \cdot \overrightarrow{x_j} + 1)^d$.
- Gaussian radial basis function: $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \exp(-\gamma\|\overrightarrow{x_i} - \overrightarrow{x_j}\|^2)$ for $\gamma > 0$. Sometimes parametrized using $\gamma = 1/(2\sigma^2)$.
- Hyperbolic tangent: $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \tanh(\kappa\overrightarrow{x_i} \cdot \overrightarrow{x_j} + c)$ for some (not every) $\kappa > 0$ and $c < 0$.

The kernel is related to the transform $\varphi(\overrightarrow{x_i})$ by the equation $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \varphi(\overrightarrow{x_i}) \cdot \varphi(\overrightarrow{x_j})$. The value **w** is also in the transformed space, with $\vec{w} = \sum_i \alpha_i y_i \varphi(\vec{x}_i)$. Dot products with **w** for classification can again be computed by the kernel trick, i.e. $\vec{w} \cdot \varphi(\vec{x}) = \sum_i \alpha_i y_i k(\vec{x}_i, \vec{x})$.

Computing the SVM classifier

Computing the (soft-margin) SVM classifier amounts to minimizing an expression of the form

$$\left[\frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i(w \cdot x_i - b)\right)\right] + \lambda\|w\|^2. \qquad (2)$$

We focus on the soft-margin classifier since, as noted above, choosing a sufficiently small value for λ yields the hard-margin classifier for linearly classifiable input data. The classical approach, which involves reducing (2) to a quadratic programming problem, is detailed below. Then, more recent approaches such as sub-gradient descent and coordinate descent will be discussed.

Primal

Minimizing (2) can be rewritten as a constrained optimization problem with a differentiable objective function in the following way.

For each $i \in \{1, \ldots, n\}$ we introduce a variable $\zeta_i = \max\left(0, 1 - y_i(w \cdot x_i - b)\right)$. Note that ζ_i is the smallest nonnegative number satisfying $y_i(w \cdot x_i - b) \geq 1 - \zeta_i$.

Thus we can rewrite the optimization problem as follows

$$\begin{aligned}
&\text{minimize } \frac{1}{n}\sum_{i=1}^{n}\zeta_i + \lambda\|w\|^2 \\
&\text{subject to } y_i(w\cdot x_i - b) \geq 1-\zeta_i \text{ and } \zeta_i \geq 0, \text{ for all } i.
\end{aligned}$$

This is called the *primal* problem.

Dual

By solving for the Lagrangian dual of the above problem, one obtains the simplified problem

$$\begin{aligned}
&\text{maximize } f(c_1 \ldots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (x_i \cdot x_j) y_j c_j, \\
&\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.
\end{aligned}$$

This is called the *dual* problem. Since the dual maximization problem is a quadratic function of the c_i subject to linear constraints, it is efficiently solvable by quadratic programming algorithms.

Here, the variables c_i are defined such that

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \vec{x}_i.$$

Moreover, $c_i = 0$ exactly when $\vec{x}_i$ lies on the correct side of the margin, and $0 < c_i < (2n\lambda)^{-1}$ when $\vec{x}_i$ lies on the margin's boundary. It follows that $\vec{w}$ can be written as a linear combination of the support vectors.

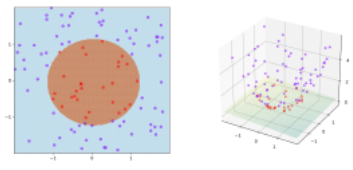
The offset, b, can be recovered by finding an $\vec{x}_i$ on the margin's boundary and solving

$$y_i(\vec{w}\cdot\vec{x}_i - b) = 1 \iff b = \vec{w}\cdot\vec{x}_i - y_i.$$

(Note that $y_i^{-1} = y_i$ since $y_i = \pm 1$.)

Kernel trick

Suppose now that we would like to learn a nonlinear classification rule which corresponds to a linear classification rule for the transformed data points $\varphi(\vec{x}_i)$. Moreover, we are given a kernel function k which satisfies $k(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i)\cdot\varphi(\vec{x}_j)$.



A training example of SVM with kernel given by φ((*a*, *b*)) = (*a*, *b*, a^2 + b^2).

We know the classification vector $\vec{w}$ in the transformed space satisfies

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \varphi(\vec{x}_i),$$

where, the c_i are obtained by solving the optimization problem

$$\begin{aligned}
&\text{maximize } f(c_1 \ldots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (\varphi(\vec{x}_i)\cdot\varphi(\vec{x}_j)) y_j c_j \\
&\qquad\qquad = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i k(\vec{x}_i, \vec{x}_j) y_j c_j \\
&\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.
\end{aligned}$$

The coefficients c_i can be solved for using quadratic programming, as before. Again, we can find some index i such that $0 < c_i < (2n\lambda)^{-1}$, so that $\varphi(\vec{x}_i)$ lies on the boundary of the margin in the transformed space, and then solve

$$\begin{aligned}
b = \vec{w}\cdot\varphi(\vec{x}_i) - y_i &= \left[\sum_{j=1}^{n} c_j y_j \varphi(\vec{x}_j)\cdot\varphi(\vec{x}_i)\right] - y_i \\
&= \left[\sum_{j=1}^{n} c_j y_j k(\vec{x}_j, \vec{x}_i)\right] - y_i.
\end{aligned}$$

Finally,

$$\vec{z} \mapsto \operatorname{sgn}(\vec{w}\cdot\varphi(\vec{z}) - b) = \operatorname{sgn}\left(\left[\sum_{i=1}^{n} c_i y_i k(\vec{x}_i, \vec{z})\right] - b\right).$$

Modern methods

Recent algorithms for finding the SVM classifier include sub-gradient descent and coordinate descent. Both techniques have proven to offer significant advantages over the traditional approach when dealing with large, sparse datasets—sub-gradient methods are especially efficient when there are many training examples, and coordinate descent when the dimension of the feature space is high.

Sub-gradient descent

Sub-gradient descent algorithms for the SVM work directly with the expression

$$f(\vec{w}, b) = \left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(w \cdot x_i - b)\right)\right] + \lambda\|w\|^2.$$

Note that f is a convex function of $\vec{w}$ and b. As such, traditional gradient descent (or SGD) methods can be adapted, where instead of taking a step in the direction of the functions gradient, a step is taken in the direction of a vector selected from the function's sub-gradient. This approach has the advantage that, for certain implementations, the number of iterations does not scale with n, the number of data points.[17]

Coordinate descent

Coordinate descent algorithms for the SVM work from the dual problem

$$\begin{aligned}
&\text{maximize } f(c_1 \ldots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (x_i \cdot x_j) y_j c_j, \\
&\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.
\end{aligned}$$

For each $i \in \{1, \ldots, n\}$, iteratively, the coefficient c_i is adjusted in the direction of $\partial f/\partial c_i$. Then, the resulting vector of coefficients $(c_1', \ldots, c_n')$ is projected onto the nearest vector of coefficients that satisfies the given constraints. (Typically Euclidean distances are used.) The process is then repeated until a near-optimal vector of coefficients is obtained. The resulting algorithm is extremely fast in practice, although few performance guarantees have been proven.[18]

Empirical risk minimization

The soft-margin support vector machine described above is an example of an empirical risk minimization (ERM) algorithm for the *hinge loss*. Seen this way, support vector machines belong to a natural class of algorithms for statistical inference, and many of its unique features are due to the behavior of the hinge loss. This perspective can provide further insight into how and why SVMs work, and allow us to better analyze their statistical properties.

Risk minimization

In supervised learning, one is given a set of training examples $X_1 \ldots X_n$ with labels $y_1 \ldots y_n$, and wishes to predict y_{n+1} given X_{n+1}. To do so one forms a hypothesis, f, such that $f(X_{n+1})$ is a "good" approximation of y_{n+1}. A "good" approximation is usually defined with the help of a *loss function*, $\ell(y, z)$, which characterizes how bad z is as a prediction of y. We would then like to choose a hypothesis that minimizes the *expected risk:*

$$\varepsilon(f) = \mathbb{E}\left[\ell(y_{n+1}, f(X_{n+1}))\right].$$

In most cases, we don't know the joint distribution of X_{n+1}, y_{n+1} outright. In these cases, a common strategy is to choose the hypothesis that minimizes the *empirical risk:*

$$\hat{\varepsilon}(f) = \frac{1}{n}\sum_{k=1}^{n}\ell(y_k, f(X_k)).$$

Under certain assumptions about the sequence of random variables X_k, y_k (for example, that they are generated by a finite Markov process), if the set of hypotheses being considered is small enough, the minimizer of the empirical risk will closely approximate the minimizer of the expected risk as n grows large. This approach is called *empirical risk minimization,* or ERM.

Regularization and stability

In order for the minimization problem to have a well-defined solution, we have to place constraints on the set $\mathcal{H}$ of hypotheses being considered. If $\mathcal{H}$ is a normed space (as is the case for SVM), a particularly effective technique is to consider only those hypotheses f for which $\|f\|_{\mathcal{H}} < k$. This is equivalent to imposing a *regularization penalty* $\mathcal{R}(f) = \lambda_k\|f\|_{\mathcal{H}}$, and solving the new optimization problem

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \hat{\varepsilon}(f) + \mathcal{R}(f).$$

This approach is called *Tikhonov regularization.*

More generally, $\mathcal{R}(f)$ can be some measure of the complexity of the hypothesis f, so that simpler hypotheses are preferred.

SVM and the hinge loss

Recall that the (soft-margin) SVM classifier $\hat{w}, b : x \mapsto \operatorname{sgn}(\hat{w} \cdot x - b)$ is chosen to minimize the following expression:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(w \cdot x_i - b)\right)\right] + \lambda\|w\|^2.$$

In light of the above discussion, we see that the SVM technique is equivalent to empirical risk minimization with Tikhonov regularization, where in this case the loss function is the hinge loss

$$\ell(y, z) = \max(0, 1 - yz)\,.$$

From this perspective, SVM is closely related to other fundamental classification algorithms such as regularized least-squares and logistic regression. The difference between the three lies in the choice of loss function: regularized least-squares amounts to empirical risk minimization with the square-loss, $\ell_{sq}(y, z) = (y - z)^2$; logistic regression employs the log-loss,

$$\ell_{\log}(y, z) = \ln(1 + e^{-yz}).$$

Target functions

The difference between the hinge loss and these other loss functions is best stated in terms of *target functions* - the function that minimizes expected risk for a given pair of random variables X, y.

In particular, let y_x denote y conditional on the event that $X = x$. In the classification setting, we have:

$$y_x = \begin{cases} 1 & \text{with probability } p_x \\ -1 & \text{with probability } 1 - p_x \end{cases}$$

The optimal classifier is therefore:

$$f^*(x) = \begin{cases} 1 & \text{if } p_x \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

For the square-loss, the target function is the conditional expectation function, $f_{sq}(x) = \mathbb{E}[y_x]$; For the logistic loss, it's the logit function, $f_{\log}(x) = \ln(p_x/(1 - p_x))$. While both of these target functions yield the correct classifier, as $\operatorname{sgn}(f_{sq}) = \operatorname{sgn}(f_{\log}) = f^*$, they give us more information than we need. In fact, they give us enough information to completely describe the distribution of y_x.

On the other hand, one can check that the target function for the hinge loss is *exactly* f^*. Thus, in a sufficiently rich hypothesis space—or equivalently, for an appropriately chosen kernel—the SVM classifier will converge to the simplest function (in terms of $\mathcal{R}$) that correctly classifies the data. This extends the geometric interpretation of SVM—for linear classification, the empirical risk is minimized by any function whose margins lie between the support vectors, and the simplest of these is the max-margin classifier.[19]

Properties

SVMs belong to a family of generalized linear classifiers and can be interpreted as an extension of the perceptron. They can also be considered a special case of Tikhonov regularization. A special property is that they simultaneously minimize the empirical *classification error* and maximize the *geometric margin*; hence they are also known as **maximum margin classifiers**.

A comparison of the SVM to other classifiers has been made by Meyer, Leisch and Hornik.[20]

Parameter selection

The effectiveness of SVM depends on the selection of kernel, the kernel's parameters, and soft margin parameter C. A common choice is a Gaussian kernel, which has a single parameter γ. The best combination of C and γ is often selected by a grid search with exponentially growing sequences of C and γ, for example, $C \in \{2^{-5}, 2^{-3}, \dots, 2^{13}, 2^{15}\}$; $\gamma \in \{2^{-15}, 2^{-13}, \dots, 2^{1}, 2^{3}\}$. Typically, each combination of parameter choices is checked using cross validation, and the parameters with best cross-validation accuracy are picked. Alternatively, recent work in Bayesian optimization can be used to select C and γ , often requiring the evaluation of far fewer parameter combinations than grid search. The final model, which is used for testing and for classifying new data, is then trained on the whole training set using the selected parameters.[21]

Issues

Potential drawbacks of the SVM include the following aspects:

- Requires full labeling of input data
- Uncalibrated class membership probabilities -- SVM stems from Vapnik's theory which avoids estimating probabilities on finite data
- The SVM is only directly applicable for two-class tasks. Therefore, algorithms that reduce the multi-class task to several binary problems have to be applied; see the multi-class SVM section.
- Parameters of a solved model are difficult to interpret.

Extensions

Support-vector clustering (SVC)

SVC is a similar method that also builds on kernel functions but is appropriate for unsupervised learning. It is considered a fundamental method in data science.

Multiclass SVM

Multiclass SVM aims to assign labels to instances by using support-vector machines, where the labels are drawn from a finite set of several elements.

The dominant approach for doing so is to reduce the single multiclass problem into multiple binary classification problems.[22] Common methods for such reduction include:[22][23]

- Building binary classifiers that distinguish between one of the labels and the rest (*one-versus-all*) or between every pair of classes (*one-versus-one*). Classification of new instances for the one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest-output function assigns the class (it is important that the output functions be calibrated to produce comparable scores). For the one-versus-one approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with the most votes determines the instance classification.

- Directed acyclic graph SVM (DAGSVM)[24]
- Error-correcting output codes[25]

Crammer and Singer proposed a multiclass SVM method which casts the multiclass classification problem into a single optimization problem, rather than decomposing it into multiple binary classification problems.[26] See also Lee, Lin and Wahba.[27][28]

Transductive support-vector machines

Transductive support-vector machines extend SVMs in that they could also treat partially labeled data in semi-supervised learning by following the principles of transduction. Here, in addition to the training set $\mathcal{D}$, the learner is also given a set

$$\mathcal{D}^\star = \{\vec{x}_i^\star \mid \vec{x}_i^\star \in \mathbb{R}^p\}_{i=1}^k$$

of test examples to be classified. Formally, a transductive support-vector machine is defined by the following primal optimization problem:[29]

Minimize (in $\vec{w}, b, \overrightarrow{y^\star}$)

$$\frac{1}{2}\|\vec{w}\|^2$$

subject to (for any $i = 1, \ldots, n$ and any $j = 1, \ldots, k$)

$$y_i(\vec{w} \cdot \overrightarrow{x_i} - b) \geq 1,$$

$$y_j^\star(\vec{w} \cdot \overrightarrow{x_j^\star} - b) \geq 1,$$

and

$$y_j^\star \in \{-1, 1\}.$$

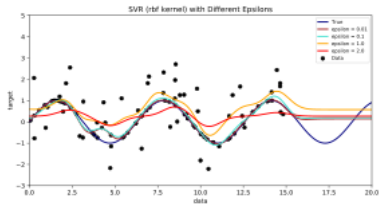
Transductive support-vector machines were introduced by Vladimir N. Vapnik in 1998.

Structured SVM

SVMs have been generalized to structured SVMs, where the label space is structured and of possibly infinite size.

Regression

A version of SVM for regression was proposed in 1996 by Vladimir N. Vapnik, Harris Drucker, Christopher J. C. Burges, Linda Kaufman and Alexander J. Smola.[30] This method is called support-vector regression (SVR). The model produced by support-vector classification (as described above) depends only 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 depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction. Another SVM version known as least-squares support-vector machine (LS-SVM) has been proposed by Suykens and Vandewalle.[31]



Support-vector regression (prediction) with different thresholds ε. As ε increases, the prediction becomes less sensitive to errors.

Training the original SVR means solving[32]

$$\text{minimize } \frac{1}{2}\|w\|^2$$

$$\text{subject to } \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon, \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon, \end{cases}$$

where x_i is a training sample with target value y_i. The inner product plus intercept $\langle w, x_i \rangle + b$ is the prediction for that sample, and ε is a free parameter that serves as a threshold: all predictions have to be within an ε range of the true predictions. Slack variables are usually added into the above to allow for errors and to allow approximation in the case the above problem is infeasible.

Bayesian SVM

In 2011 it was shown by Polson and Scott that the SVM admits a Bayesian interpretation through the technique of data augmentation.[33] In this approach the SVM is viewed as a graphical model (where the parameters are connected via probability distributions). This extended view allows the application of Bayesian techniques to SVMs, such as flexible feature modeling, automatic hyperparameter tuning, and predictive uncertainty quantification. Recently, a scalable version of the Bayesian SVM was developed by Wenzel et al. enabling the application of Bayesian SVMs to big data.[34]

Implementation

The parameters of the maximum-margin hyperplane are derived by solving the optimization. There exist several specialized algorithms for quickly solving the quadratic programming (QP) problem that arises from SVMs, mostly relying on heuristics for breaking the problem down into smaller, more manageable chunks.

Another approach is to use an interior-point method that uses Newton-like iterations to find a solution of the Karush–Kuhn–Tucker conditions of the primal and dual problems.[35] Instead of solving a sequence of broken-down problems, this approach directly solves the problem altogether. To avoid solving a linear system involving the large kernel matrix, a low-rank approximation to the matrix is often used in the kernel trick.

Another common method is Platt's sequential minimal optimization (SMO) algorithm, which breaks the problem down into 2-dimensional sub-problems that are solved analytically, eliminating the need for a numerical optimization algorithm and matrix storage. This algorithm is conceptually simple, easy to implement, generally faster, and has better scaling properties for difficult SVM problems.[36]

The special case of linear support-vector machines can be solved more efficiently by the same kind of algorithms used to optimize its close cousin, logistic regression; this class of algorithms includes sub-gradient descent (e.g., PEGASOS[37]) and coordinate descent (e.g., LIBLINEAR[38]). LIBLINEAR has some attractive training-time properties. Each convergence iteration takes time linear in the time taken to read the train data, and the iterations also have a Q-linear convergence property, making the algorithm extremely fast.

The general kernel SVMs can also be solved more efficiently using sub-gradient descent (e.g. P-packSVM[39]), especially when parallelization is allowed.

Kernel SVMs are available in many machine-learning toolkits, including LIBSVM, MATLAB, SAS (http://support.sas.com/documentation/cdl/en/whatsnew/64209/HTML/default/viewer.htm#emdocwhatsnew71.htm), SVMlight, kernlab (https://cran.r-project.org/package=kernlab), scikit-learn, Shogun, Weka, Shark (http://image.diku.dk/shark/), JKernelMachines (https://mloss.org/software/view/409/), OpenCV and others.

See also

- In situ adaptive tabulation
- Kernel machines
- Fisher kernel
- Platt scaling
- Polynomial kernel
- Predictive analytics
- Regularization perspectives on support-vector machines
- Relevance vector machine, a probabilistic sparse-kernel model identical in functional form to SVM
- Sequential minimal optimization
- Space mapping
- Winnow (algorithm)

References

1. Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks" (http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf) (PDF). *Machine Learning*. **20** (3): 273–297. CiteSeerX 10.1.1.15.9362 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.9362). doi:10.1007/BF00994018 (https://doi.org/10.1007%2FBF00994018).
2. Ben-Hur, Asa; Horn, David; Siegelmann, Hava; and Vapnik, Vladimir N.; "Support vector clustering"; (2001); *Journal of Machine Learning Research*, 2: 125–137.
3. "1.4. Support Vector Machines — scikit-learn 0.20.2 documentation" (http://scikit-learn.org/stable/modules/svm.html). Archived (https://web.archive.org/web/20171108151644/http://scikit-learn.org/stable/modules/svm.html) from the original on 2017-11-08. Retrieved 2017-11-08.
4. Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning : Data Mining, Inference, and Prediction* (https://web.stanford.edu/~hastie/Papers/ESLII.pdf#page=153) (PDF) (Second ed.). New York: Springer. p. 134.
5. Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, Brian P. (2007). "Section 16.5. Support Vector Machines" (http://apps.nrbook.com/empanel/index.html#pg=883). *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8. Archived (https://web.archive.org/web/20110811154417/http://apps.nrbook.com/empanel/index.html#pg=883) from the original on 2011-08-11.
6. Pradhan, Sameer S., et al. "Shallow semantic parsing using support vector machines (http://www.aclweb.org/anthology/N04-1030)." Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004. 2004.
7. Vapnik, Vladimir N.: Invited Speaker. IPMU Information Processing and Management 2014).
8. Barghout, Lauren. "Spatial-Taxon Information Granules as Used in Iterative Fuzzy-Decision-Making for Image Segmentation (https://pdfs.semanticscholar.org/917f/15d33d32062bffeb6401eee9fe71d16d6a84.pdf)". Granular Computing and Decision-Making. Springer International Publishing, 2015. 285–318.
9. DeCoste, Dennis (2002). "Training Invariant Support Vector Machines" (https://people.eecs.berkeley.edu/~malik/cs294/decoste-scholkopf.pdf) (PDF). *Machine Learning*. **46**: 161–190. doi:10.1023/A:1012454411458 (https://doi.org/10.1023%2FA%3A1012454411458).
10. Gaonkar, Bilwaj; Davatzikos, Christos; "Analytic estimation of statistical significance maps for support vector machine based multi-variate image analysis and classification" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3767485/).
11. Cuingnet, Rémi; Rosso, Charlotte; Chupin, Marie; Lehéricy, Stéphane; Dormont, Didier; Benali, Habib; Samson, Yves; and Colliot, Olivier; "Spatial regularization of SVM for the detection of diffusion alterations associated with stroke outcome" (http://www.aramislab.fr/perso/colliot/files/media2011_remi_published.pdf), *Medical Image Analysis*, 2011, 15 (5): 729–737.
12. Statnikov, Alexander; Hardin, Douglas; & Aliferis, Constantin; (2006); "Using SVM weight-based methods to identify causally relevant and non-causally relevant variables" (http://www.ccdlab.org/paper-pdfs/NIPS_2006.pdf), *Sign*, 1, 4.
13. Boser, Bernhard E.; Guyon, Isabelle M.; Vapnik, Vladimir N. (1992). "A training algorithm for optimal margin classifiers". *Proceedings of the fifth annual workshop on Computational learning theory – COLT '92* (http://www.clopinet.com/isabelle/Papers/colt92.ps.Z). p. 144. CiteSeerX 10.1.1.21.3818 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.3818). doi:10.1145/130385.130401 (https://doi.org/10.1145%2F130385.130401). ISBN 978-0897914970.
14. "Why does the SVM margin is $\frac{2}{\|\mathbf{w}\|}$" (https://math.stackexchange.com/q/1305925/168764). *Mathematics Stack Exchange*. 30 May 2015.
15. Aizerman, Mark A.; Braverman, Emmanuel M. & Rozonoer, Lev I. (1964). "Theoretical foundations of the potential function method in pattern recognition learning". *Automation and Remote Control*. **25**: 821–837.
16. Jin, Chi; Wang, Liwei (2012). *Dimensionality dependent PAC-Bayes margin bound* (http://papers.nips.cc/paper/4500-dimensionality-dependent-pac-bayes-margin-bound). Advances in Neural Information Processing Systems. CiteSeerX 10.1.1.420.3487 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.420.3487). Archived (https://web.archive.org/web/20150402185336/http://papers.nips.cc/paper/4500-dimensionality-dependent-pac-bayes-margin-bound) from the original on 2015-04-02.
17. Shalev-Shwartz, Shai; Singer, Yoram; Srebro, Nathan; Cotter, Andrew (2010-10-16). "Pegasos: primal estimated sub-gradient solver for SVM". *Mathematical Programming*. **127** (1): 3–30. CiteSeerX 10.1.1.161.9629 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.161.9629). doi:10.1007/s10107-010-0420-4 (https://doi.org/10.1007%2Fs10107-010-0420-4). ISSN 0025-5610 (https://www.worldcat.org/issn/0025-5610).
18. Hsieh, Cho-Jui; Chang, Kai-Wei; Lin, Chih-Jen; Keerthi, S. Sathiya; Sundararajan, S. (2008-01-01). *A Dual Coordinate Descent Method for Large-scale Linear SVM*. *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. New York, NY, USA: ACM. pp. 408–415. CiteSeerX 10.1.1.149.5594 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.149.5594). doi:10.1145/1390156.1390208 (https://doi.org/10.1145%2F1390156.1390208). ISBN 978-1-60558-205-4.
19. Rosasco, Lorenzo; De Vito, Ernesto; Caponnetto, Andrea; Piana, Michele; Verri, Alessandro (2004-05-01). "Are Loss Functions All the Same?" (http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6789841&abstractAccess=no&userType=inst). *Neural Computation*. **16** (5): 1063–1076. CiteSeerX 10.1.1.109.6786 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.109.6786). doi:10.1162/089976604773135104 (https://doi.org/10.1162%2F089976604773135104). ISSN 0899-7667 (https://www.worldcat.org/issn/0899-7667). PMID 15070510 (https://www.ncbi.nlm.nih.gov/pubmed/15070510).
20. Meyer, David; Leisch, Friedrich; Hornik, Kurt (September 2003). "The support vector machine under test". *Neurocomputing*. **55** (1–2): 169–186. doi:10.1016/S0925-2312(03)00431-4 (https://doi.org/10.1016%2FS0925-2312%2803%2900431-4).
21. Hsu, Chih-Wei; Chang, Chih-Chung & Lin, Chih-Jen (2003). *A Practical Guide to Support Vector Classification* (http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf) (PDF) (Technical report). Department of Computer Science and Information Engineering, National Taiwan University. Archived (https://web.archive.org/web/20130625201224/http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf) (PDF) from the original on 2013-06-25.
22. Duan, Kai-Bo; Keerthi, S. Sathiya (2005). "Which Is the Best Multiclass SVM Method? An Empirical Study". *Multiple Classifier Systems* (https://www.cs.iastate.edu/~honavar/multiclass-svm2.pdf) (PDF). LNCS. **3541**. pp. 278–285. CiteSeerX 10.1.1.110.6789 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.110.6789). doi:10.1007/11494683_28 (https://doi.org/10.1007%2F11494683_28). ISBN 978-3-540-26306-7.
23. Hsu, Chih-Wei & Lin, Chih-Jen (2002). "A Comparison of Methods for Multiclass Support Vector Machines" (http://www.cs.iastate.edu/~honavar/multiclass-svm.pdf) (PDF). *IEEE Transactions on Neural Networks*.

24. Platt, John; Cristianini, Nello; Shawe-Taylor, John (2000). "Large margin DAGs for multiclass classification" (http://www.wisdom.weizmann.ac.il/~bagon/CVspring07/files/DAGSVM.pdf) (PDF). In Solla, Sara A.; Leen, Todd K.; and Müller, Klaus-Robert; eds. (eds.). *Advances in Neural Information Processing Systems*. MIT Press. pp. 547–553. Archived (https://web.archive.org/web/20120616221540/http://www.wisdom.weizmann.ac.il/~bagon/CVspring07/files/DAGSVM.pdf) (PDF) from the original on 2012-06-16.
25. Dietterich, Thomas G.; Bakiri, Ghulum (1995). "Solving Multiclass Learning Problems via Error-Correcting Output Codes" (http://www.jair.org/media/105/live-105-1426-jair.pdf) (PDF). *Journal of Artificial Intelligence Research*. **2**: 263–286. arXiv:cs/9501101 (https://arxiv.org/abs/cs/9501101). Bibcode:1995cs.....1101D (https://ui.adsabs.harvard.edu/abs/1995cs.....1101D). doi:10.1613/jair.105 (https://doi.org/10.1613%2Fjair.105). Archived (https://web.archive.org/web/20130509061344/http://www.jair.org/media/105/live-105-1426-jair.pdf) (PDF) from the original on 2013-05-09.
26. Crammer, Koby & Singer, Yoram (2001). "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines" (http://jmlr.csail.mit.edu/papers/volume2/crammer01a/crammer01a.pdf) (PDF). *Journal of Machine Learning Research*. **2**: 265–292. Archived (https://web.archive.org/web/20150829102651/http://jmlr.csail.mit.edu/papers/volume2/crammer01a/crammer01a.pdf) (PDF) from the original on 2015-08-29.
27. Lee, Yoonkyung; Lin, Yi & Wahba, Grace (2001). "Multicategory Support Vector Machines" (http://www.interfacesymposia.org/I01/I2001Proceedings/YLee/YLee.pdf) (PDF). *Computing Science and Statistics*. **33**. Archived (https://web.archive.org/web/20130617093314/http://www.interfacesymposia.org/I01/I2001Proceedings/YLee/YLee.pdf) (PDF) from the original on 2013-06-17.
28. Lee, Yoonkyung; Lin, Yi; Wahba, Grace (2004). "Multicategory Support Vector Machines". *Journal of the American Statistical Association*. **99** (465): 67. CiteSeerX 10.1.1.22.1879 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.1879). doi:10.1198/016214504000000098 (https://doi.org/10.1198%2F016214504000000098).
29. Joachims, Thorsten; "Transductive Inference for Text Classification using Support Vector Machines (http://www1.cs.columbia.edu/~dplewis/candidacy/joachims99transductive.pdf)", *Proceedings of the 1999 International Conference on Machine Learning (ICML 1999)*, pp. 200–209.
30. Drucker, Harris; Burges, Christ. C.; Kaufman, Linda; Smola, Alexander J.; and Vapnik, Vladimir N. (1997); "Support Vector Regression Machines (http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf)", in *Advances in Neural Information Processing Systems 9, NIPS 1996*, 155–161, MIT Press.
31. Suykens, Johan A. K.; Vandewalle, Joos P. L.; "Least squares support vector machine classifiers (https://lirias.kuleuven.be/bitstream/123456789/218716/2/Suykens_NeurProcLett.pdf)", *Neural Processing Letters*, vol. 9, no. 3, Jun. 1999, pp. 293–300.
32. Smola, Alex J.; Schölkopf, Bernhard (2004). "A tutorial on support vector regression" (http://eprints.pascal-network.org/archive/00000856/01/fulltext.pdf) (PDF). *Statistics and Computing*. **14** (3): 199–222. CiteSeerX 10.1.1.41.1452 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.1452). doi:10.1023/B:STCO.0000035301.49549.88 (https://doi.org/10.1023%2FB%3ASTCO.0000035301.49549.88). Archived (https://web.archive.org/web/20120131193522/http://eprints.pascal-network.org/archive/00000856/01/fulltext.pdf) (PDF) from the original on 2012-01-31.
33. Polson, Nicholas G.; Scott, Steven L. (2011). "Data Augmentation for Support Vector Machines". *Bayesian Analysis*. **6** (1): 1–23. doi:10.1214/11-BA601 (https://doi.org/10.1214%2F11-BA601).
34. Wenzel, Florian; Galy-Fajou, Theo; Deutsch, Matthäus; Kloft, Marius (2017). "Bayesian Nonlinear Support Vector Machines for Big Data". *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. Lecture Notes in Computer Science. **10534**: 307–322. arXiv:1707.05532 (https://arxiv.org/abs/1707.05532). Bibcode:2017arXiv170705532W (https://ui.adsabs.harvard.edu/abs/2017arXiv170705532W). doi:10.1007/978-3-319-71249-9_19 (https://doi.org/10.1007%2F978-3-319-71249-9_19). ISBN 978-3-319-71248-2.
35. Ferris, Michael C.; Munson, Todd S. (2002). "Interior-Point Methods for Massive Support Vector Machines" (http://www.cs.wisc.edu/~ferris/papers/siopt-svm.pdf) (PDF). *SIAM Journal on Optimization*. **13** (3): 783. CiteSeerX 10.1.1.216.6893 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.216.6893). doi:10.1137/S1052623400374379 (https://doi.org/10.1137%2FS1052623400374379). Archived (https://web.archive.org/web/20081204224416/http://www.cs.wisc.edu/~ferris/papers/siopt-svm.pdf) (PDF) from the original on 2008-12-04.
36. Platt, John C. (1998). *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines* (http://research.microsoft.com/pubs/69644/tr-98-14.pdf) (PDF). NIPS. Archived (https://web.archive.org/web/20150702075055/http://research.microsoft.com/pubs/69644/tr-98-14.pdf) (PDF) from the original on 2015-07-02.
37. Shalev-Shwartz, Shai; Singer, Yoram; Srebro, Nathan (2007). *Pegasos: Primal Estimated sub-GrAdient SOlver for SVM* (http://ttic.uchicago.edu/~shai/papers/ShalevSiSr07.pdf) (PDF). ICML. Archived (https://web.archive.org/web/20131215051700/http://ttic.uchicago.edu/~shai/papers/ShalevSiSr07.pdf) (PDF) from the original on 2013-12-15.
38. Fan, Rong-En; Chang, Kai-Wei; Hsieh, Cho-Jui; Wang, Xiang-Rui; Lin, Chih-Jen (2008). "LIBLINEAR: A library for large linear classification" (https://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf) (PDF). *Journal of Machine Learning Research*. **9**: 1871–1874.
39. Allen Zhu, Zeyuan; Chen, Weizhu; Wang, Gang; Zhu, Chenguang; Chen, Zheng (2009). *P-packSVM: Parallel Primal grAdient desCent Kernel SVM* (http://people.csail.mit.edu/zeyuan/paper/2009-ICDM-Parallel.pdf) (PDF). ICDM. Archived (https://web.archive.org/web/20140407095709/http://people.csail.mit.edu/zeyuan/paper/2009-ICDM-Parallel.pdf) (PDF) from the original on 2014-04-07.

Further reading

- Bennett, Kristin P.; Campbell, Colin (2000). "Support Vector Machines: Hype or Hallelujah?" (http://kdd.org/exploration_files/bennett.pdf) (PDF). *SIGKDD Explorations*. **2** (2): 1–13.
- Cristianini, Nello; Shawe-Taylor, John (2000). *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press. ISBN 0-521-78019-5.
- Fradkin, Dmitriy; Muchnik, Ilya (2006). "Support Vector Machines for Classification" (http://paul.rutgers.edu/~dfradkin/papers/svm.pdf) (PDF). In Abello, J.; Carmode, G. (eds.). *Discrete Methods in Epidemiology*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. **70**. pp. 13–20.
- Ivanciuc, Ovidiu (2007). "Applications of Support Vector Machines in Chemistry" (http://www.ivanciuc.org/Files/Reprint/Ivanciuc_SVM_CCR_2007_23_291.pdf) (PDF). *Reviews in Computational Chemistry*. **23**: 291–400.
- James, Gareth; Witten, Daniela; Hastie, Trevor; Tibshirani, Robert (2013). "Support Vector Machines" (http://www-bcf.usc.edu/~gareth/ISL/ISLR%20Seventh%20Printing.pdf#page=345) (PDF). *An Introduction to Statistical Learning : with Applications in R*. New York: Springer. pp. 337–372. ISBN 978-1-4614-7137-0.
- Schölkopf, Bernhard; Smola, Alexander J. (2002). *Learning with Kernels*. Cambridge, MA: MIT Press. ISBN 0-262-19475-9.
- Steinwart, Ingo; Christmann, Andreas (2008). *Support Vector Machines*. New York: Springer. ISBN 978-0-387-77241-7.
- Theodoridis, Sergios; Koutroumbas, Konstantinos (2009). *Pattern Recognition* (4th ed.). Academic Press. ISBN 978-1-59749-272-0.

External links

- libsvm (http://www.csie.ntu.edu.tw/~cjlin/libsvm/), LIBSVM is a popular library of SVM learners
- liblinear (http://www.csie.ntu.edu.tw/~cjlin/liblinear/) is a library for large linear classification including some SVMs
- SVM light (http://svmlight.joachims.org) is a collection of software tools for learning and classification using SVM
- SVMJS live demo (http://cs.stanford.edu/people/karpathy/svmjs/demo/) is a GUI demo for JavaScript implementation of SVMs