svmtrain

Train support vector machine classifier

Syntax

```
SVMStruct = svmtrain(Training,Group)
SVMStruct = svmtrain(Training,Group,Name,Value)
```

Description

`SVMStruct` = `svmtrain(`**`Training,Group`**`)` returns a structure, `SVMStruct`, containing information about the trained support vector machine (SVM) classifier.

`SVMStruct` = `svmtrain(`**`Training,Group,Name,Value`**`)` returns a structure with additional options specified by one or more `Name,Value` pair arguments.

Input Arguments

`Training`	Matrix of training data, where each row corresponds to an observation or replicate, and each column corresponds to a feature or variable. `svmtrain` treats `NaN`s or empty strings in `Training` as missing values and ignores the corresponding rows of `Group`.
`Group`	Grouping variable, which can be a categorical, numeric, or logical vector, a cell vector of strings, or a character matrix with each row representing a class label. Each element of `Group` specifies the group of the corresponding row of `Training`. `Group` should divide `Training` into two groups. `Group` has the same number of elements as there are rows in `Training`. `svmtrain` treats each `NaN`, empty string, or `'undefined'` in `Group` as a missing value, and ignores the corresponding row of `Training`.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`' '`). You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

`'autoscale'`	Boolean specifying whether `svmtrain` automatically centers the data points at their mean, and scales them to have unit standard deviation, before training. **Default:** `true`
`'boxconstraint'`	Value of the box constraint `C` for the soft margin. `C` can be a scalar, or a vector of the same length as the training data. If `C` is a scalar, it is automatically rescaled by `N/(2*N1)` for the data points of group one and by `N/(2*N2)` for the data points of group two, where `N1` is the number of elements in group one, `N2` is the number of elements in group two, and `N = N1 + N2`. This rescaling is done to take into account unbalanced groups, that is cases where `N1` and `N2` have very different values. If `C` is an array, then each array element is taken as a box constraint for the

	data point with the same index. **Default:** `1`
`'kernelcachelimit'`	Value that specifies the size of the kernel matrix cache for the SMO training method. The algorithm keeps a matrix with up to `kernelcachelimit` × `kernelcachelimit` double-precision, floating-point numbers in memory. **Default:** `5000`

`'kernel_function'`	Kernel function `svmtrain` uses to map the training data into kernel space. The default kernel function is the dot product. The kernel function can be one of the following strings or a function handle:
	
	▪ `'linear'` — Linear kernel, meaning dot product.
	▪ `'quadratic'` — Quadratic kernel.
	▪ `'polynomial'` — Polynomial kernel (default order 3). Specify another order with the `polyorder` name-value pair.
	▪ `'rbf'` — Gaussian Radial Basis Function kernel with a default scaling factor, sigma, of 1. Specify another value for sigma with the `rbf_sigma` name-value pair.
	▪ `'mlp'` — Multilayer Perceptron kernel with default scale `[1 -1]`. Specify another scale with the `mlp_params` name-value pair.
	▪ `@`*`kfun`* — Function handle to a kernel function. A kernel function must be of the form
	
	`function K = kfun(U, V)`
	
	The returned value, `K`, is a matrix of size `M`-by-`N`, where `U` and `V` have `M` and `N` rows respectively.
	
	If `kfun` has extra parameters, include the extra parameters via an anonymous function. For example, suppose that your kernel function is:
	
	`function k = kfun(u,v,p1,p2)`
	`k = tanh(p1*(u*v')+p2);`
	
	Set values for `p1` and `p2`, and then use an anonymous function:
	
	`@(u,v) kfun(u,v,p1,p2)`
	
	Default: `'linear'`

`'kktviolationlevel'`	Value that specifies the fraction of variables allowed to violate the Karush-Kuhn-Tucker (KKT) conditions for the SMO training method. Set any value in `[0,1)`. For example, if you set `kktviolationlevel` to `0.05`, then 5% of the variables are allowed to violate the KKT conditions. **Tip** Set this option to a positive value to help the algorithm converge if it is fluctuating near a good solution. For more information on KKT conditions, see Cristianini and Shawe-Taylor **[4]**. **Default:** `0`
`'method'`	Method used to find the separating hyperplane. Options are: ▪ `'QP'` — Quadratic programming (requires an Optimization Toolbox™ license). The classifier is a 2-norm soft-margin support vector machine. Give quadratic programming options with the `options` name-value pair, and create `options` with **`optimset`**. ▪ `'SMO'` — Sequential Minimal Optimization. Give SMO options with the `options` name-value pair, and create `options` with **`statset`**. ▪ `'LS'` — Least squares. **Default:** `SMO`
`'mlp_params'`	Parameters of the Multilayer Perceptron (`mlp`) kernel. The `mlp` kernel requires two parameters, `[P1 P2]`. The kernel `K = tanh(P1*U*V' + P2)`, where `P1 > 0` and `P2 < 0`. **Default:** `[1 -1]`
`'options'`	Options structure for training.

- When you set `'method'` to `'SMO'` (default), create the `options` structure using **`statset`**. Options are:

`Display`	String that specifies the level of information about the optimization iterations that is displayed as the algorithm runs. Choices are: ▪ `off` (default) — Reports nothing. ▪ `iter` — Reports every 500 iterations. ▪ `final` — Reports only when the algorithm finishes.
`MaxIter`	Integer that specifies the maximum number of iterations of the main loop. If this limit is exceeded before the algorithm converges, then the algorithm stops and returns an error. Default is `15000`.

  The other name-value pairs that relate specifically to the `'SMO'` method are `kernelcachelimit`, `kktviolationlevel`, and `tolkkt`.

- When you set `method` to `'QP'`, create the options structure using **`optimset`**. For details of applicable option choices, see **`quadprog`** options. SVM uses a convex quadratic program, so you can choose the `'interior-point-convex'` `quadprog` algorithm. In limited testing, the `'interior-point-convex'` algorithm was the best `quadprog` option for `svmtrain`, in both speed and memory utilization.

`'polyorder'`	Order of the polynomial kernel. **Default:** `3`
`'rbf_sigma'`	Scaling factor (sigma) in the radial basis function kernel. **Default:** `1`
`'showplot'`	Boolean indicating whether to plot the grouped data and separating line. Creates a plot only when the data has two columns (features). **Default:** `false`
`'tolkkt'`	Value that specifies the tolerance with which the Karush-Kuhn-Tucker (KKT) conditions are checked for the SMO training method. For a definition of KKT conditions, see **Karush-Kuhn-Tucker (KKT) Conditions**. **Default:** `1e-3`

Output Arguments

`SVMStruct`	Structure containing information about the trained SVM classifier in the following fields:

- `SupportVectors` — Matrix of data points with each row corresponding to a support vector in the normalized data space. This matrix is a subset of the *`Training`* input data matrix, after normalization has been applied according to the `'AutoScale'` argument.
- `Alpha` — Vector of weights for the support vectors. The sign of the weight is positive for support vectors belonging to the first group, and negative for the second group.
- `Bias` — Intercept of the hyperplane that separates the two groups in the normalized data space (according to the `'AutoScale'` argument).
- `KernelFunction` — Handle to the function that maps the training data into kernel space.
- `KernelFunctionArgs` — Cell array of any additional arguments required by the kernel function.
- `GroupNames` — Categorical, numeric, or logical vector, a cell vector of strings, or a character matrix with each row representing a class label. Specifies the group identifiers for the support vectors. It has the same number of elements as there are rows in `SupportVectors`. Each element specifies the group to which the corresponding row in `SupportVectors` belongs.
- `SupportVectorIndices` — Vector of indices that specify the rows in `Training`, the training data, that were selected as support vectors after the data was normalized, according to the `AutoScale` argument.
- `ScaleData` — Field containing normalization factors. When `'AutoScale'` is set to `false`, it is empty. When `AutoScale` is set to `true`, it is a structure containing two fields:
  - `shift` — Row vector of values. Each value is the negative of the mean across an observation in *`Training`*, the training data.
  - `scaleFactor` — Row vector of values. Each value is `1` divided by the standard deviation of an observation in *`Training`*, the training data.

  Both `svmtrain` and **`svmclassify`** apply the scaling in `ScaleData`.
- `FigureHandles` — Vector of figure handles created by `svmtrain` when using the `'Showplot'` argument.

Examples

expand all

Train an SVM Classifier

More About

expand all

Karush-Kuhn-Tucker (KKT) Conditions

Tips

Algorithms

- **Support Vector Machines (SVM)**
- **Grouping Variables**