Two-Class Locally Deep Support Vector Machine

Published: August 14, 2015

Updated: August 31, 2015

Creates a binary classification model using the locally deep Support Vector Machine algorithm

Category: Machine Learning / Initialize Model / Classification (https://msdn.microsoft.com/en-us/library/azure/dn905808.aspx)

Module Overview

You can use the **Two-Class Locally Deep Support Vector Machine** module to create a two-class, non-linear support vector machines (SVM) classifier that is optimized for efficient prediction.

SVMs are an extremely popular and well-researched class of supervised learning models, which can be used in linear and non-linear classification tasks. Recent research has focused on ways to optimize these models to efficiently scale to larger training sets. In this implementation from Microsoft Research, the kernel function that is used for mapping data points to feature space is specifically designed to reduce the time needed for training while maintaining most of the classification accuracy.

This model is a supervised learning method, and therefore requires a *tagged dataset*, which includes a label column.

You can train the model by providing the model and the tagged dataset as an input to Train Model (https://msdn.microsoft.com/en-us/library/azure/dn906044.aspx) or Sweep Parameters (https://msdn.microsoft.com/en-us/library/azure/dn905810.aspx). The trained model can then be used to predict values for the new input examples.

How to Configure an LD-SVM Model

1. Specify how you want the model to be trained, by setting the **Create trainer mode** option.
    - **Single Parameter**

      If you know how you want to configure the SVM model, you can provide a specific set of values as arguments. You might have learned these values by experimentation or received them as guidance.

    - **Parameter Range**

If you are not sure of the best parameters, you can find the optimal parameters by specifying multiple values and using a parameter sweep to find the optimal configuration.

Sweep Parameters (https://msdn.microsoft.com/en-us/library/azure/dn905810.aspx) will iterate over all possible combinations of the settings you provided and determine the combination of settings that produces the optimal results.

2. Continue to set other parameters that affect the behavior of the SVM model, such as the depth of the tree and lambda values. For more information, see the Options section.

3. If you set the **Create trainer mode** option to **Single Parameter**, add a tagged dataset and train the model by using the Train Model (https://msdn.microsoft.com/en-us/library/azure/dn906044.aspx) module.

   If you set the **Create trainer mode** option to **Parameter Range**, add a tagged dataset and train the model using Sweep Parameters (https://msdn.microsoft.com/en-us/library/azure/dn905810.aspx). You can use the model trained using those parameters, or you can make a note of the parameter settings to use when configuring a learner.

Options

You can optimize the behavior of the algorithm by setting the following parameters:

Create trainer mode

Choose the method used for configuring and training the model:

- ***Single Parameter***

  Select this option to configure and train the model with a single set of parameter values that you supply.

  If you choose this option, you should train the model by using the Train Model (https://msdn.microsoft.com/en-us/library/azure/dn906044.aspx) module.

- ***Parameter Range***

  Select this option to use the Range Builder and specify a range of possible values. You then train the model using a parameter sweep, to find the optimum configuration.

Warning
• If you pass a parameter range to Train Model (https://msdn.microsoft.com/en-us/library/azure/dn906044.aspx), it will use only the first value in the parameter range list.

- If you pass a single set of parameter values to the Sweep Parameters (https://msdn.microsoft.com/en-us/library/azure/dn905810.aspx) module, when it expects a range of settings for each parameter, it ignores the values and using the default values for the learner.
- If you select the **Parameter Range** option and enter a single value for any parameter, that single value you specified will be used throughout the sweep, even if other parameters change across a range of values.

Depth of the tree

Type an integer that determines the maximum depth of the tree that can be created by the local deep kernel learning SVM (LD-SVM) model.

The cost of training increases linearly with tree depth; therefore, choose an appropriate depth, depending on how much time you can afford to spend when building the model.

- Training time should roughly double as the depth is increased by one.
- Prediction accuracy should increase, reach a peak, and then decrease with increasing depth.

Lambda W

Specify the weight given to the regularization term.

Regularization restricts large value components in the trained classifier. When the number of samples is insufficient given the number of features, you can use L2 regularization to avoid overfitting. Larger values for **Lambda W** mean that more emphasis will be placed on regularizing the classifier weights and less on the training-set classification error.

If the default value (0.1) doesn't work well, you should also try {0.0001, 0.001 and 0.01}.

Lambda Theta

Specify how much space should be left between a region boundary and the closest data point.

This model works by partitioning the data space and feature space into regions. When **Lambda Theta** is minimized in such a way that that region boundaries in the trained model are too close to the training data points, the model might yield a low training error, but a high test error, due to overfitting.

To reduce the number of parameters that need to be validated, a good rule of thumb is to set **Lambda Theta** to one-tenth of the value that is used for **Lambda W**. Larger values mean that more emphasis will be put on preventing overfitting than on minimizing classification errors in the training set.

If the default value (0.01) doesn't work well, you should also try {0.0001, 0.001 and 0.1}.

Lambda Theta Prime

Type a value to control the amount of curvature that is allowed in decision boundaries in the model.

Larger values will give the model the flexibility to learn curved decision boundaries, while smaller values might constrain the decision boundaries to more of a stepwise linear pattern.

This parameter works in conjunction with the **Sigma** parameter. To reduce the number of parameters that need to be validated, a good rule of thumb is to set **Lambda Theta Prime** to one-tenth the value of **Lambda W**.

If the default value (0.01) doesn't work well, you should also try {0.0001, 0.001 and 0.1,}.

Sigma sharpness

Type a value to use for the scaling parameter σ.

Larger values mean that the tanh in local kernel Θ (theta) is saturated, whereas a smaller value implies a more linear operating range for theta. You can find the full optimization formula in the Technical Notes section.

If the default value (1) does not work well, you can also try {0.1, 0.01, 0.001}.

Number of iterations

Specify the number of times that the algorithm updates the classifier parameters by using a random subset of examples.

Feature normalizer

Choose the method that is used for feature normalization, or use no normalization. The following feature normalization methods are supported:

- ***Binning normalizer***: The binning normalizer creates bins of equal size, and then normalizes every value in each bin to be divided by the total number of bins.

- ***Gaussian normalizer***: The Gaussian normalizer rescales the values of each feature to have mean 0 and variance 1. This is done by computing the mean and the variance of each feature, and then, for each instance, subtracting the mean value and dividing by the square root of the variance (the standard deviation).

- ***Min-max normalizer***: The min-max normalizer linearly rescales every feature to the [0,1] interval.

  Rescaling to the [0,1] interval is done by shifting the values of each feature so that the minimal value is 0, and then dividing by the new maximal value (which is the difference between the original maximal and minimal values).

- ***Do not normalize***: No normalization is performed.

Random number seed

Type a value to use as a seed if you want to ensure reproducibility across runs.

Allow unknown categorical levels

Select this option to create a group for unknown values in the testing or validation sets.

If you deselect it, the model can accept only the values that are contained in the training data. In the former case, the model might be less precise for known values, but it can provide better predictions for new (unknown) values.

Recommendations

This LD-SVM classifier is most useful under the following conditions:

- You have a binary classification issue, or you can reduce your issue to a binary classification task.
- You tried a linear classifier, but it did not perform well.
- You tried a non-linear SVM or other classifier, and got good classification accuracy, but it took too long to train the model.
- You can afford to sacrifice prediction accuracy to reduce training time.

LD-SVM models are a good choice when your data is complicated enough that linear models (such as logistic regression) perform poorly. LD-SVM models are also small enough to be used in mobile devices or other scenarios where complex models (such as neural networks) are too big to be consumed efficiently.

Conversely, this model should not be used if you don't care about model size or if a linear model is required for simplicity or prediction speed. There is also no point to changing to LD-SVM if linear classifiers are already giving good results or if you can get high classification accuracy by adding small amounts of non-linearity.

Technical Notes

The LD-SVM model was developed by Microsoft Research as part of ongoing efforts to speed up non-linear SVM prediction. The work of Gonen and Alpaydin (2008) on the localized multiple kernel learning method was particularly valuable. Use of a local kernel function enables the model to learn arbitrary local feature embeddings, including high-dimensional, sparse, and computationally deep features that introduce non-linearities into the model.

LD-SVM is faster than most other classifiers for several reasons:

- The model learns decision boundaries that are locally linear. Therefore, a test point can be efficiently classified by testing it against its local decision boundary, rather than testing against the entire set of decision boundaries all over feature space.
- The model uses efficient primal-based routines to optimize the space of tree-structured local feature embeddings that scale to large training sets with more than half a million training points.
- The cost of testing a point against its local decision boundary is logarithmic in the number of training points.

As a consequence of these optimizations, training the LD-SVM model is exponentially faster than training traditional SVM models.

Optimization formula

$$\min_{W,\theta,\theta'} P(W,\theta,\theta') = \frac{\lambda_W}{2}Tr(W^tW)+\frac{\lambda_\theta}{2}Tr(\theta^t\theta)+\frac{\lambda_{\theta'}}{2}Tr(\theta'^t\theta')+\sum_{i=1}^{N} L(y_i,\phi_L^t(x_i)W^tx_i)$$

Research

For more details, see Local Deep Kernel Learning for Efficient Non-linear SVM Prediction (http://go.microsoft.com/fwlink/?LinkId=511662).

Module Parameters

Name	Range	Type	Default	Description
Create trainer mode	List	Learner parameter option	Single Parameter	Advanced learner options: 1. Create learner using a single parameter 2. Create learner using a parameter range
Depth of the tree	>=1	Integer	3	The depth of the locally deep SVM tree.
Lambda W	>=1.401298E-45	Float	0.1	Regularization weight for the classifier parameter Lambda W.
Lambda Theta	>=1.401298E-45	Float	0.01	Regularization weight for the classifier parameter Lambda Theta.
Lambda Theta Prime	>=1.401298E-45	Float	0.01	Regularization weight for the classifier parameter Lambda Theta prime.
	>=1.401298E-45	Float	1.0	

Sigmoid sharpness				The sigmoid sharpness.
Depth of the tree	[1;int.MaxValue]	ParameterRangeSettings	1; 3; 5; 7	The range for the depth of the locally deep SVM tree.
Lambda W	[1.401298E-45;3.40282347E+38]	ParameterRangeSettings	0.1; 0.01; 0.001	Range for the regularization weight for the classifier parameter Lambda W.
Lambda Theta	[1.401298E-45;3.40282347E+38]	ParameterRangeSettings	0.1; 0.01; 0.001	Range for the regularization weight for the classifier parameter Lambda Theta.
Lambda Theta Prime	[1.401298E-45;3.40282347E+38]	ParameterRangeSettings	0.1; 0.01; 0.001	Range for the regularization weight for the classifier parameter Lambda Theta prime'.
Sigmoid sharpness	[1.401298E-45;3.40282347E+38]	ParameterRangeSettings	1.0; 0.1; 0.01	The range for the sigmoid sharpness.
Feature normalizer	List	Normalizer type (https://msdn.microsoft.com/en-us/library/azure/dn905916.aspx)	Min-Max normalizer	The type of normalization to apply to learning examples.
Number of iterations	>=1	Integer	15000	Number of learning iterations.
Number of iterations	[1;int.MaxValue]	ParameterRangeSettings	10000; 15000; 20000	The range for the number of learning iterations.
Random number seed	Any	Integer		The seed for the random number generator that is used by the

				model. Leave it blank for the default.
Allow unknown categorical levels	Any	Boolean	True	If True, creates an additional level for each categorical column. Any levels in the test dataset that are not available in the training dataset are mapped to this additional level.

Output

Name	Type	Description
Untrained model	ILearner interface (https://msdn.microsoft.com/en-us/library/azure/dn905938.aspx)	An untrained binary classification model.

See Also

Machine Learning / Initialize Model / Classification (https://msdn.microsoft.com/en-us/library/azure/dn905808.aspx)
A-Z List of Machine Learning Studio Modules (https://msdn.microsoft.com/en-us/library/azure/dn906033.aspx)