GRADUATE SCHOOL OF BUSINESS

Introduction to Machine Learning

Fall 2019

Dmitry Udler

Hyperparameter Optimization

- Optimization, in its most general form, is the process of locating a point that minimizes a real-valued function called the objective function. Bayesian optimization is the name of one such process. Bayesian optimization internally maintains a Gaussian process model of the objective function, and uses objective function evaluations to train the model. One innovation in Bayesian optimization is the use of an acquisition function, which the algorithm uses to determine the next point to evaluate. The acquisition function can balance sampling at points that have low modeled objective functions, and exploring areas that have not yet been modeled well.

Hyperparameter Optimization

- A hyperparameter is an internal parameter of a classifier or regression function, such as the box constraint of a support vector machine, or the learning rate of a robust classification ensemble. These parameters can strongly affect the performance of a classifier or regressor, and yet it is typically difficult or time-consuming to optimize them.
- Typically, optimizing the hyperparameters means that you try to minimize the cross-validation loss of a classifier or regression.

Hyperparameter Optimization

- A hyperparameter is an internal parameter of a classifier or regression function, such as the box constraint of a support vector machine, or the learning rate of a robust classification ensemble. These parameters can strongly affect the performance of a classifier or regressor, and yet it is typically difficult or time-consuming to optimize them.
- Typically, optimizing the hyperparameters means that you try to minimize the cross-validation loss of a classifier or regression.

Hyperparameter Optimization

- Two Ways to Perform Bayesian Optimization
- You can perform a Bayesian optimization in two distinct ways:
- Fit function — Include the OptimizeHyperparameters name-value pair in many fitting functions to have Bayesian optimization apply automatically. The optimization minimizes cross-validation loss. This way gives you fewer tuning options, but enables you to perform Bayesian optimization most easily.
- bayesopt — Exert the most control over your optimization by calling bayesopt directly. This way requires you to write an objective function, which does not have to represent cross-validation loss.

Hyperparameter Optimization

- Bayesian Optimization Using a Fit Function
- To minimize the error in a cross-validated response via Bayesian optimization, follow these steps.
- Choose your classification or regression solver among fitcdiscr, fitcecoc, fitcensemble, fitckernel, fitcknn, fitclinear, fitcnb, fitcsvm, fitctree, fitrensemble, fitrgp, fitrkernel, fitrlinear, fitrsvm, or fitrtree.
- Decide on the hyperparameters to optimize, and pass them in the OptimizeHyperparameters name-value pair. For each fit function, you can choose from a set of hyperparameters. See Eligible Hyperparameters for Fit Functions, or use the hyperparameters function, or consult the fit function reference page.
- You can pass a cell array of parameter names. You can also set 'auto' as the OptimizeHyperparameters value, which chooses a typical set of hyperparameters to optimize, or 'all' to optimize all available parameters.
- For ensemble fit functions fitcecoc, fitcensemble, and fitrensemble, also include parameters of the weak learners in the OptimizeHyperparameters cell array.
- Optionally, create an options structure for the HyperparameterOptimizationOptions name-value pair. See Hyperparameter Optimization Options for Fit Functions.
- Call the fit function with the appropriate name-value pairs.

Hyperparameter Optimization

- Bayesian Optimization Using bayesopt
- To perform a Bayesian optimization using bayesopt, follow these steps.
- Prepare your variables. See Variables for a Bayesian Optimization.
- Create your objective function. See Bayesian Optimization Objective Functions. If necessary, create constraints, too. See Constraints in Bayesian Optimization.
- Decide on options, meaning the bayseopt Name,Value pairs. You are not required to pass any options to bayesopt but you typically do, especially when trying to improve a solution.
- Call bayesopt.
- Examine the solution. You can decide to resume the optimization by using resume, or restart the optimization, usually with modified options.
- For an example, see Optimize a Cross-Validated SVM Classifier Using bayesopt.

Hyperparameter Optimization

- Bayesian Optimization Advantages and Limitations
- Bayesian optimization algorithms are best suited to these problem types.
- Low dimension.Bayesian optimization works best in a low number of dimensions, typically 10 or fewer. While Bayesian optimization can solve some problems with a few dozen variables, it is not recommended for dimensions higher than about 50.
- Expensive objective. Bayesian optimization is designed for objective functions that are slow to evaluate. It has considerable overhead, typically several seconds for each iteration.
- Low accuracy.y Bayesian optimization does not necessarily give very accurate results. If you have a deterministic objective function, you can sometimes improve the accuracy by starting a standard optimization algorithm from the bayesopt solution.
- Global solution Bayesian optimization is a global technique. Unlike many other algorithms, to search for a global solution you do not have to start the algorithm from various initial points.
- Hyperparameters. Bayesian optimization is well-suited to optimizing hyperparameters of another function. A hyperparameter is a parameter that controls the behavior of a function. For example, the fitcsvm function fits anSVM model to data. It has hyperparameters BoxConstraint and KernelScale for its 'rbf' KernelFunction. For an example of Bayesian optimization applied to hyperparameters, see Optimize a Cross-Validated SVM Classifier Using bayesopt.

Hyperparameter Optimization

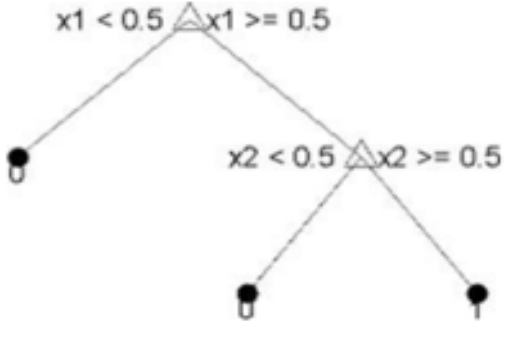
- fitcsvm
- BoxConstraint
  KernelScale
  KernelFunction
  PolynomialOrder
  Standardize

Decision Trees

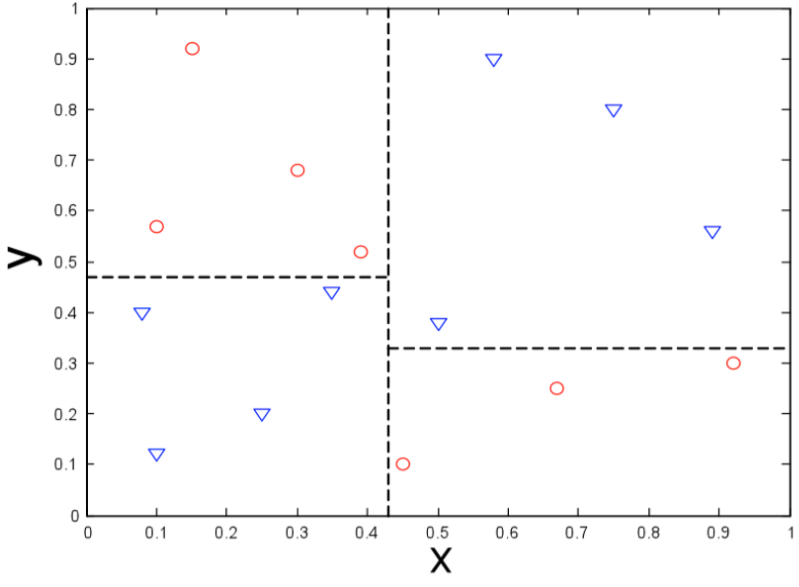
- Decision Trees or Classification Trees or Regression Trees predict responses to data.
- The response is predicted by going down from the root to the particular leaf node.
- Classification trees give responses that are nominal, while regression trees give numerical responses.
- In matlab trees are binary.

Decision Trees Example

- Example. This tree predicts classifications based on two predictors, x1 and x2. To predict, start at the top node, represented by a triangle(Δ). The first decision is whether x1 is smaller than 0.5. If so, follow the left branch, and see that the tree classifies the data as type0.
- If, however, x1 exceeds 0.5, then follow the right branch to the lower-right triangle node. Here the tree asks if x2 is smaller than 0.5. If so, then follow the left branch to see that the tree classifies the data as type 0. If not, then follow the right branch to see that the that the tree classifies the data as type 1.



Decision Trees Example



Training a Classification Tree

CART

- Start with input data and examine possible binary splits on every predictor.
- Select a split with the best optimization criterion
- A split may lead to a child node having too few observations (less than MinLeafSize parameter). To avoid this the algorithm selects a split that delivers the best optimization criterion subject to the MinLeafSize constraint).
- Impose the split.
- Repeat recursively for the two nodes.

To visualize the tree:

view(TreeModel,'Mode','Graph')

CART

The algorithm requires two conditions:

1. Optimization criterion
1. Stopping rule

You can control the depth of the trees using the parameters:

MaxNumSplits

MeanLeafSize

MeanParentSize

Fitctree grows deep decision trees by default.

CART

The default values of the tree depth controllers for growing classification trees are:

N-1 for MaxNumSplits, where n is the training sample size

1 for MeanLeafSize

10 for MeanParentSize

These default values tend to grow deep trees for large training sample sizes.

Decision Trees: Optimization criterion

Gini diversity index (gdi) –default

The Gini index of a node is

$$1 - \sum_i p^2(i),$$

where the sum i ıe node, and p(i) is the observed fraction of classes with class i that reach the node. A node with just one class (a pure node) has Gini index 0, otherwise the Gini index is positive . So the Gini index is a measure of node impurity.

Decision Trees: Optimization criterion

Twoing is not a purity measure of a node, but is a different measure for deciding how to split a node. Let L(i) denote the fraction of members of class i in the left child node after a split, and R(i) denote the fraction of members of class i in the right child node after the split.

$$P(L)P(R)\left(\sum_i |L(i) - R(i)|\right)^2,$$

where P(L) and P(R) are the fractions of observations that split to the left and right respectively. If the expression is large, the split made each child node purer. Similarly, if the expression is small, the split made each child node similar to each other, and hence similar to the parent node, and so the split did not increase node purity.

Decision Trees: split criterion

Deviance ('deviance')

The deviance of a node is

$$-\sum_i p(i)\log_2 p(i).$$

A pure node has dev ; positive.

Entropy.

Decision Trees: Optimization criterion

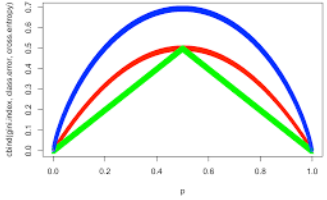
Misclassification error: the highest fraction of the class in the node.

$1 - p(j)$.

Red – Gini index

Green – Classification error

Blue – Deviance



Decision Trees: Optimization criterion

Node error — The node error is the fraction of misclassified classes at a node. If j is the class with the largest number of training samples at a node, the node error is

$$1 - p(j).$$

Decision Trees: Stopping rule

1. The node is pure, i.e. all its observations belong to the same class.
2. There are fewer than MinParentSize observations in the node.
3. Any split imposed on this node produces children with fewer than MinLeafSize observations.
4. The algorithm reaches MacNumSplits.

CART

In general the idea the following:

1. Grow an overly large tree using forward selection. At each step, find the best split. Grow until all terminal nodes either

(a) have < n (perhaps n = 1) data points,

(b) are “pure” (all points in a node have [almost] the same outcome).

2. Prune the tree back, creating a nested sequence of trees, decreasing in complexity.

A problem in tree construction is how to use the training data to determine the binary splits of X into smaller and smaller pieces. The fundamental idea is to select each split of a subset so that the data in each of the descendant subsets are “purer” than the data in the parent subset.

CART

- The standard CART algorithm tends to select continuous predictors that have many levels. Sometimes, such a selection can be spurious and can also mask more important predictors that have fewer levels, such as categorical predictors. That is, the predictor-selection process at each node is biased. Also, standard CART tends to miss the important interactions between pairs of predictors and the response.
- To mitigate selection bias and increase detection of important interactions, you can specify usage of the curvature or interaction tests using the 'PredictorSelection' name-value pair argument. Using the curvature or interaction test has the added advantage of producing better predictor importance estimates than standard CART.

CART

Technique	'PredictorSelection' Value	Description	Training speed	When to specify
Standard CART [1]	Default	Selects the split predictor that maximizes the split-criterion gain over all possible splits of all predictors.	Baseline for comparison	Specify if any of these conditions are true: • All predictors are continuous • Predictor importance is not the analysis goal • For boosting decision trees
Curvature test [2][3]	'curvature'	Selects the split predictor that minimizes the *p*-value of chi-square tests of independence between each predictor and the response.	Comparable to standard CART	Specify if any of these conditions are true: • The predictor variables are heterogeneous • Predictor importance is an analysis goal • Enhance tree interpretation
Interaction test [3]	'interaction-curvature'	Chooses the split predictor that minimizes the *p*-value of chi-square tests of independence between each predictor and the response (that is, conducts curvature tests), and that minimizes the *p*-value of a chi-square test of independence between each pair of predictors and response.	Slower than standard CART, particularly when data set contains many predictor variables.	Specify if any of these conditions are true: • The predictor variables are heterogeneous • You suspect associations between pairs of predictors and the response • Predictor importance is an analysis goal • Enhance tree interpretation

CART

Value	Description
'allsplits'	Standard CART — Selects the split predictor that maximizes the split-criterion gain over all possible splits of all predictors [1].
'curvature'	Curvature test — Selects the split predictor that minimizes the p-value of chi-square tests of independence between each predictor and the response [4]. Training speed is similar to standard CART.
'interaction-curvature'	Interaction test — Chooses the split predictor that minimizes the p-value of chi-square tests of independence between each predictor and the response, and that minimizes the p-value of a chi-square test of independence between each pair of predictors and response [3]. Training speed can be slower than standard CART.

Out-of-Sample Prediction

When a classification tree Mdl model is trained it can be used for prediction:

Ynew=predict(Mdl, Xnew)

Decision trees –what is nice?

- Decision trees are very “natural” constructs, in particular when the explanatory variables are categorical (and even better, when they are binary).
- Trees are very easy to explain to non-statisticians.
- The models are invariant under transformations in the predictor space.
- Multi-factor response is easily dealt with.
- The treatment of missing values is more satisfactory than for most other model classes.
- The models go after interactions immediately, rather than as an afterthought.
- The tree growth is actually more efficient than I have described it.

Decision trees –what is not so nice

- The tree-space is huge, so we may need a lot of data.
- We might not be able to find the “best” model at all.
- It can be hard to assess uncertainty in inference about trees.
- The results can be quite variable (the tree selection is not very stable).
- Simple trees often do not have a lot of predictive power (are weak learners).
- There is a selection bias for the splits - predictors with a higher number of distinct values are favored over more coarse-granular predictors .

Improving Classification Trees and Regression Trees

- Examining Resubstitution Error
- Resubstitution error is the difference between the response training data and the predictions the tree makes of the response based on the input training data. If the resubstitution error is high, you cannot expect the predictions of the tree to be good. However, having low resubstitution error does not guarantee good predictions for new data. Resubstitution error is often an overly optimistic estimate of the predictive error on new data.

Improving Classification Trees and Regression Trees

- Cross Validation
- To get a better sense of the predictive accuracy of your tree for new data, cross validate the tree. By default, cross validation splits the training data into 10 parts at random. It trains 10 new trees, each one on nine parts of the data. It then examines the predictive accuracy of each new tree on the data not included in training that tree. This method gives a good estimate of the predictive accuracy of the resulting tree, since it tests the new trees on new data.

Improving Classification Trees and Regression Trees

- Control Depth or “Leafiness”
- When you grow a decision tree, consider its simplicity and predictive power. A deep tree with many leaves is usually highly accurate on the training data. However, the tree is not guaranteed to show a comparable accuracy on an independent test set. A leafy tree tends to overtrain (or overfit), and its test accuracy is often far less than its training (resubstitution) accuracy. In contrast, a shallow tree does not attain high training accuracy. But a shallow tree can be more robust — its training accuracy could be close to that of a representative test set. Also, a shallow tree is easy to interpret. If you do not have enough data for training and test, estimate tree accuracy by cross validation.
- fitctree and fitrtree have three name-value pair arguments that control the depth of resulting decision trees:
- MaxNumSplits — The maximal number of branch node splits is MaxNumSplits per tree. Set a large value for MaxNumSplits to get a deep tree. The default is size(X,1) – 1.
- MinLeafSize — Each leaf has at least MinLeafSize observations. Set small values of MinLeafSize to get deep trees. The default is 1.
- MinParentSize — Each branch node in the tree has at least MinParentSize observations. Set small values of MinParentSize to get deep trees. The default is 10.
- If you specify MinParentSize and MinLeafSize, the learner uses the setting that yields trees with larger leaves (i.e., shallower trees):
- MinParent = max(MinParentSize,2*MinLeafSize)
- If you supply MaxNumSplits, the software splits a tree until one of the three splitting criteria is satisfied.

Improving Classification Trees and Regression Trees

Pruning

Pruning optimizes tree depth (leafiness) by merging leaves on the same tree branch. Control Depth or “Leafiness” describes one method for selecting the optimal depth for a tree. Unlike in that section, you do not need to grow a new tree for every node size. Instead, grow a deep tree, and prune it to the level you choose.

Prune a tree at the command line using the prune method (classification) or prune method (regression). Alternatively, prune a tree interactively with the tree viewer:

```
view(tree,'mode','graph')
```

To prune a tree, the tree must contain a pruning sequence. By default, both fitctree and fitrtree calculate a pruning sequence for a tree during construction. If you construct a tree with the 'Prune' name-value pair set to 'off', or if you prune a tree to a smaller level, the tree does not contain the full pruning sequence. Generate the full pruning sequence with the prune method (classification) or prune method (regression).

Hyperparameters for Classification Trees

MinLeafSize
MaxNumSplits
SplitCriterion
NumVariablesToSample

Number of predictors to select at random for each split, specified as the comma-separated pair consisting of 'NumVariablesToSample' and a positive integer value. Alternatively, you can specify 'all' to use all available predictors.
If a training data includes many predictors and an analysis of predictor importance is your goal, then specify 'NumVariablesToSample' as 'all'. Otherwise, the software may never choose some predictors and underestimate their importance.
To reproduce the random selections, you must set the seed of the random number generator by using rng and specify 'Reproducible',true.