AI 学习笔记--GRT--DBScan

GRT 是一个二维的学习库，内部集成了很多种基础的 AI 学习算法可应用于最多 6 维数据的分析。目前 兼容 2 维数据分析比较好。首先基础能力提供以下几个方法：

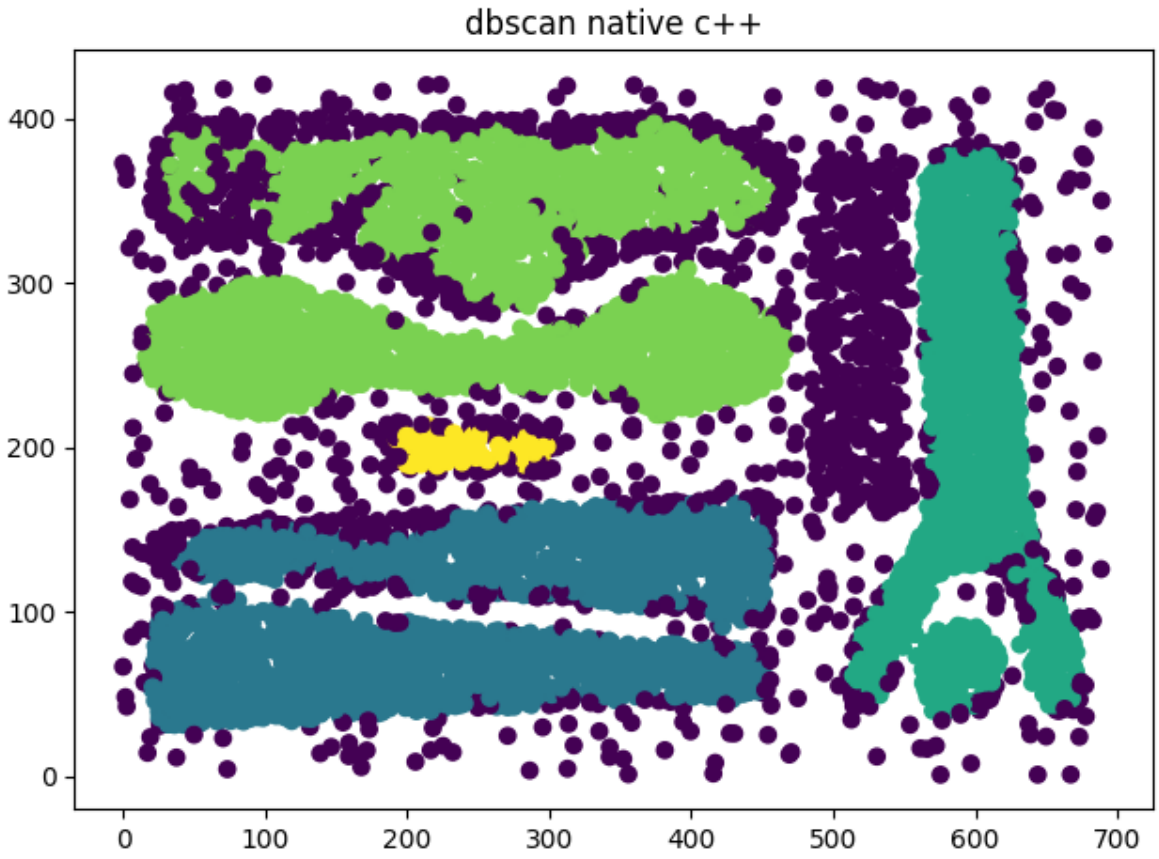
```
/**
    This is the main interface for training the clusterer model.

    @param trainingData: a reference to the training data that will be used
to train the ML model
    @return returns true if the model was successfully trained, false
otherwise
    */
   virtual bool train_(MatrixFloat &trainingData) override;

   /**
    Override the main ClassificationData train function to pass MatrixFloat
data to the Clusterer train function.

    @param trainingData: a reference to the training data that will be used
to train the ML model
    @return returns true if the model was successfully trained, false
otherwise
    */
   virtual bool train_(ClassificationData &trainingData) override;

   /**
    Override the main UnlabelledData train function to pass MatrixFloat
data to the Clusterer train function.

    @param trainingData: a reference to the training data that will be used
to train the ML model
    @return returns true if the model was successfully trained, false
otherwise
    */
   virtual bool train_(UnlabelledData &trainingData) override;

   /**
    This resets the Clusterer.
    This overrides the reset function in the MLBase base class.

    @return returns true if the Clusterer was reset, false otherwise
    */
   virtual bool reset() override;

   /**
    This function clears the Clusterer module, removing any trained model
and setting all the base variables to their default values.

    @return returns true if the derived class was cleared succesfully,
```

```
false otherwise
    */
   virtual bool clear() override;

   /**
    Returns the number of clusters in the model.

    @return returns the number of clusters
    */
   UINT getNumClusters() const;

   /**
    Returns the predicted cluster label.

    @return returns the predicted cluster label
    */
   UINT getPredictedClusterLabel() const;

   /**
    Returns the current maximumLikelihood value.
    The maximumLikelihood value is computed during the prediction phase and
is the likelihood of the most likely model.
    This value will return 0 if a prediction has not been made.

    @return returns the current maximumLikelihood value
    */
   Float getMaximumLikelihood() const;

   /**
    Returns the current bestDistance value.
    The bestDistance value is computed during the prediction phase and is
either the minimum or maximum distance, depending on the algorithm.
    This value will return 0 if a prediction has not been made.

    @return returns the current bestDistance value
    */
   Float getBestDistance() const;

   /**
    Gets a Vector of the cluster likelihoods from the last prediction, this
will be an N-dimensional Vector, where N is the number of clusters in the
model.
    The exact form of these likelihoods depends on the cluster algorithm.

    @return returns a Vector of the cluster likelihoods from the last
prediction, an empty Vector will be returned if the model has not been
trained
    */
   VectorFloat getClusterLikelihoods() const;

   /**
```

```
    Gets a Vector of the cluster distances from the last prediction, this
will be an N-dimensional Vector, where N is the number of clusters in the
model.
    The exact form of these distances depends on the cluster algorithm.

    @return returns a Vector of the cluster distances from the last
prediction, an empty Vector will be returned if the model has not been
trained
    */
   VectorFloat getClusterDistances() const;

   /**
    Gets a Vector of unsigned ints containing the label of each cluster,
this will be an K-dimensional Vector, where K is the number of clusters in
the model.

    @return returns a Vector of unsigned ints containing the label of each
cluster, an empty Vector will be returned if the model has not been trained
    */
   Vector< UINT > getClusterLabels() const;

   GRT_DEPRECATED_MSG( "getClustererType() is deprecated, use getId() or
getBaseId() instead", std::string getClustererType() const );

   /**
    Sets the number of clusters that will be used the next time a model is
trained.
    This will clear any previous model.
    The number of clusters must be greater than zero.

    @param numClusters: the number of clusters, must be greater than zero
    @return returns true if the value was updated successfully, false
otherwise
    */
   bool setNumClusters(const UINT numClusters);

   /**
    Defines a map between a string (which will contain the name of the
Clusterer, such as KMeans) and a function returns a new instance of that
Clusterer
    */
   typedef std::map< std::string, Clusterer*(*)() > StringClustererMap;

   /**
    Creates a new Clusterer instance based on the input string (which
should contain the name of a valid Clusterer such as KMeans).

    @param id: the id of the Clusterer
    @return a pointer to the new instance of the Clusterer
    */
   static Clusterer* create( std::string const &id );
```

```
    /**
     Creates a new Clusterer instance based on the current clustererType
string value.

     @return Clusterer*: a pointer to the new instance of the Clusterer
    */
    Clusterer* create() const;

    GRT_DEPRECATED_MSG( "createNewInstance is deprecated, use create
instead.", Clusterer* createNewInstance() const );
    GRT_DEPRECATED_MSG( "createInstanceFromString is deprecated, use create
instead.", static Clusterer* createInstanceFromString( const std::string
&id ) );

    /**
     This creates a new Clusterer instance and deep copies the variables and
models from this instance into the deep copy.
     The function will then return a pointer to the new instance. It is up
to the user who calls this function to delete the dynamic instance
     when they are finished using it.

     @return returns a pointer to a new Clusterer instance which is a deep
copy of this instance
     */
    Clusterer* deepCopy() const;

    /**
     Returns a pointer to this Clusterer. This is useful for a derived class
so it can get easy access to this base Clusterer.

     @return Clusterer&: a reference to this Clusterer
     */
    const Clusterer& getBaseClusterer() const;

    /**
     Returns a Vector of the names of all Clusterers that have been
registered with the base Clusterer.

     @return Vector< std::string >: a Vector containing the names of the
Clusterers that have been registered with the base Clusterer
    */
static Vector< std::string > getRegisteredClusterers();

//Tell the compiler we are explicitly using the following classes from the
base class (this stops hidden overloaded virtual function warnings)
    using MLBase::train;
```

主要的核心方法分为两类，一个是 Clustering，用于数据的聚类。此类算法包含了 DBScan、ClusterTree、Gaussian、Kmeans 等等。针对不同的数据模型选用合适的聚类方法，比如 dbscan算法：



立刻离开了87777776　　上层暴露的接口可能包括了

```
/**
This trains the GMM model, using the labelled classification data.
This overrides the train function in the GRT::Classifier base class.
The GMM is an unsupervised learning algorithm, it will therefore NOT use
any class labels provided

@param trainingData: a reference to the training data
@return returns true if the GMM model was trained, false otherwise
*/
virtual bool train_(ClassificationData &trainingData);

/**
This predicts the class of the inputVector.
This overrides the predict function in the GRT::Classifier base class.

@param inputVector: the input vector to classify
@return returns true if the prediction was performed, false otherwise
*/
virtual bool predict_(VectorFloat &inputVector);

/**
This overrides the clear function in the Classifier base class.
It will completely clear the ML module, removing any trained model and
```

```
setting all the base variables to their default values.

@return returns true if the module was cleared succesfully, false otherwise
*/
virtual bool clear();

/**
This saves the trained GMM model to a file.
This overrides the save function in the GRT::Classifier base class.

@param file: a reference to the file the GMM model will be saved to
@return returns true if the model was saved successfully, false otherwise
*/
virtual bool save( std::fstream &file ) const;

/**
This loads a trained GMM model from a file.
This overrides the load function in the GRT::Classifier base class.

@param file: a reference to the file the GMM model will be loaded from
@return returns true if the model was loaded successfully, false otherwise
*/
virtual bool load( std::fstream &file );
```

load 用于模型导入、数据导入，配置导入等等（传输格式待定）。train 用于训练数据，predict 用于预测，save 保存训练后的模型结果。

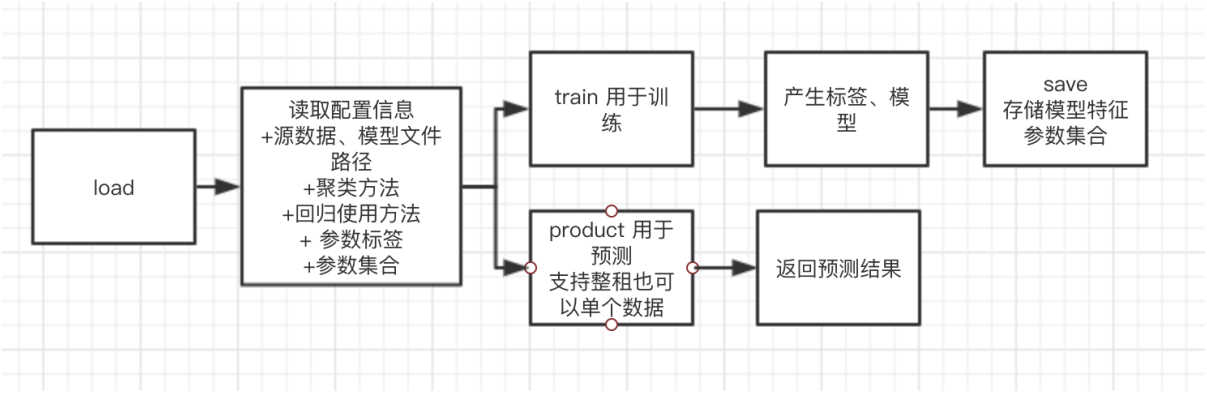
- 输入规范和输出规范

输入最好是以 二维数组，部分算法支持最高 6 维。以 csv 格式文件最佳。规定第一列为标签，第二列和第三列等作为数据维度，可以依据后续需求作为修改。例如 x,y,id,label的维度数据如下：

```
id,label,x,y
0,0,84.769,33.369
1,1,569.791,55.458
2,1,657.623,47.035
3,2,217.057,362.065
4,2,131.724,353.369
5,0,146.775,77.422
6,0,368.503,154.196
7,0,391.971,154.476
8,0,370.949,60.969
```

输入参数规范，每种算法都具有每种算法的特定，需要在 SK-learn 上模拟训练参数，然后倒入到执行层面。

- 大概执行流程


load
读取配置信息
+源数据、模型文件路径
+聚类方法
+回归使用方法
+ 参数标签
+参数集合
train 用于训练
产生标签、模型
save
存储模型特征
参数集合
product 用于预测
支持整租也可以单个数据
返回预测结果