Hilbert SDK 接口文档

概述

该 SDK 提供了与 Hilbert 向量数据库服务交互的 Python 客户端，支持索引管理、向量操作和查询功能。所有操作通过 gRPC 协议与服务端通信。

安装与初始化

```
# 创建客户端实例
client = HilbertClient(server_address="127.0.0.1:7000", debug=False)
```

参数说明

参数	类型	说明	默认值
`server_address`	str	服务器地址和端口	必填
`debug`	bool	启用详细响应日志, 可打印服务端详细返回	False

索引管理

1. 创建索引

```
client.create_index(name="my_index", dim=128, replica_num=2, index_type=1,
card_num=1)
```

参数说明：

参数	类型	说明	约束	默认值
`name`	str	索引名称	唯一标识，1<=长度<=50	-
`dim`	int	向量维度	0<dim<=8192	-
`replica_num`	int	副本数量	0<=replica_num<=2	1
`index_type`	int	索引类型，0：BF, 1: IVF	预留参数	1
`card_num`	int	卡片数量	预留参数	1

返回: `True` （成功时）

异常:

- `RuntimeError`：创建失败时抛出，包含错误详情

2. 删除索引

```
client.delete_index(name="my_index")
```

参数说明:

参数	类型	说明	约束
`name`	str	索引名称	必须存在

返回: `True` （成功时）

异常:

- `RuntimeError`：删除失败时抛出

3. 查询所有索引

```
indexes = client.query_all_index()
```

返回:

```
[
    {
        'name': 'index1',
        'nlist': 128,
        'dim': 128,
        'nb': 10000,
        'replica_num': 1,
        'index_type': 1
    },
    # ...其他索引
]
```

字段说明:

- `nlist`：聚类中心数
- `nb`：向量数量
- 其他字段含义同创建索引

向量操作

4. 训练索引

```
# 准备训练数据 (Numpy数组)
train_data = np.random.rand(10000, 128).astype(np.float32)
client.train(name="my_index", data=train_data, nlist=256)
```

参数说明：

参数	类型	说明	约束
`name`	str	索引名称	必须存在
`data`	np.ndarray	训练数据	shape=(N, dim), dtype=float32; data受nlist限制, 应满足nb >= nlist*39
`nlist`	int	聚类中心数	>0, 默认128

返回： `True` （成功时）

异常：

- `TypeError`：数据类型错误
- `ValueError`：数据维度错误
- `RuntimeError`：训练失败

5. 添加向量

```
vectors = np.random.rand(500, 128).astype(np.float32)
vector_ids = client.add(name="my_index", data=vectors, mode_flag=0)
```

参数说明：

参数	类型	说明	约束	默认值
`name`	str	索引名称	必须存在	-
`data`	np.ndarray	向量数据	shape=(N, dim), dtype=float32	-

返回：

```
[123, 124, 125, ...]  # 生成的向量ID列表
```

异常:

- `TypeError`: 数据类型错误
- `ValueError`: 数据维度错误
- `RuntimeError`: 添加失败

查询操作

8. 向量搜索

```
queries = np.random.rand(10, 128).astype(np.float32)
distances, labels = client.search(
    name="my_index",
    queries=queries,
    k=5,
    nprob=32,
    batch_size=100
)
```

参数说明:

参数	类型	说明	约束	默认值
`name`	str	索引名称	必须存在	-
`queries`	np.ndarray	查询向量	shape=(N, dim)	-
`k`	int	返回结果数	>0	10，整体需要满足 8 * k * batch_size < 4M
`nprob`	int	搜索探测数	0<nprob<=分簇数量	32
`batch_size`	int	查询分批大小	>0	300, 整体需要满足 8 * k * batch_size < 4M

返回:

```
(
    np.ndarray,  # 距离矩阵 shape=(nq, k)
    np.ndarray   # 标签矩阵 shape=(nq, k)
```

```
)
```

9. 查询单个向量

```
response = client.query_vector(
    name="my_index",
    vector_id=123
)
```

参数说明：

参数	类型	说明	约束	默认值
`name`	str	索引名称	必须存在	-
`vector_id`	int	向量ID	必须存在	-

响应对象字段：

```
{
    'data': [0.1, 0.2, ...],          # 原始向量数据
    'digital_data': [0.1, 0.2, ...],  # 数字域处理后的数据
    'reram_data': [123, 456, ...],    # ReRAM格式数据
    # ...其他服务端返回字段
}
```

10. 更新向量

```
client.update_vector(
    name="my_index",
    vector_id=123,
    data=[0.1, 0.2, ..., 0.128]  # 新向量数据
)
```

参数说明：

参数	类型	说明	约束
`name`	str	索引名称	必须存在
`vector_id`	int	向量ID	必须存在
`data`	list	新向量数据	长度=dim

返回： `True` （成功时）

11. 删除向量

```
client.delete_vector(name="my_index", vector_id=123)
```

参数说明：

参数	类型	说明	约束
`name`	str	索引名称	必须存在
`vector_id`	int	向量ID	必须存在

返回： `True`（成功时）

错误处理

所有方法可能抛出以下异常：

- `RuntimeError`：包含 gRPC 返回的错误代码和消息
- `TypeError`：参数类型错误
- `ValueError`：参数值错误
- `FileNotFoundError`：HDF5 文件不存在（仅限 `get_data_from_hdf5`）

性能建议

1. 确保向量数据为 `np.float32` 类型
2. 搜索操作合理设置 `batch_size`（通常 100-500）
3. 生产环境关闭 `debug` 模式减少日志开销

> 注意：实际使用时请根据服务端支持的参数范围调整具体数值