Lab2 实验报告

牛庆源 PB21111733

part_1 传统机器学习

1. 决策树

1. 根据训练数据构建决策树并在测试集上测试。构建决策树的过程主体是一个递归的过程:
   - 读入数据 → 生成节点 → 判断节点是否可以划分属性 → 选择最优划分 → 生成分支

   递归进行。

   1. 判断部分（出口）： 若为空则赋值 Null； 若样本都属于类别C则标记为C类叶节点。判断结束本次递归结束。
   2. 选择最优划分部分的依据为信息增益，依据实验文档给出的公式计算。
   3. 连续值处理采用二分法离散化。
2. 具体过程:
   1. 计算信息熵:

```
def entropy(y):
    classes, counts = np.unique(y, return_counts=True)
    probabilities = counts / len(y)
    return -np.sum(probabilities * np.log2(probabilities))
```

   2. 计算信息增益（二分法处理连续值）：

```
# 信息增益
    def info_gain(self, X, y, feature, threshold):
        left_idx = X[:, feature] < threshold
        right_idx = X[:, feature] >= threshold
        left_entropy = self.entropy(y[left_idx])
        right_entropy = self.entropy(y[right_idx])
        return self.entropy(y) - np.mean(left_idx) * left_entropy -
np.mean(right_idx) * right_entropy
```

   3. 遍历分割点选择最佳分割:

```python
# 选择最佳分割点
    def best_split(self, X, y):
        best_feature, best_threshold, best_info_gain = None, None, -
np.inf
        n_features = X.shape[1]
        for feature in range(n_features):
            thresholds = np.unique(X[:, feature])
            for threshold in thresholds:
                ig = self.info_gain(X, y, feature, threshold)
                if ig > best_info_gain:
                    best_feature, best_threshold, best_info_gain =
feature, threshold, ig
        return best_feature, best_threshold, best_info_gain
```

④ 构建决策树

```python
# 递归构建决策树
    def build_tree(self, X, y, max_depth, min_samples_leaf):
        if len(y) == 0:
            return {'threshold': None}
        if max_depth == 0 or len(y) < min_samples_leaf:
            return {'threshold': np.argmax(np.bincount(y))}
        feature, threshold, info_gain = self.best_split(X, y)
        # 信息增益为0
        if info_gain == 0:
            return {'threshold':np.argmax(np.bincount(y))}
        left_idx = X[:, feature] < threshold
        right_idx = X[:, feature] >= threshold
        left_tree = self.build_tree(X[left_idx], y[left_idx],
max_depth-1, min_samples_leaf)
        right_tree = self.build_tree(X[right_idx], y[right_idx],
max_depth-1, min_samples_leaf)
        return {'feature': feature,
                'threshold': threshold,
                'left_tree': left_tree,
                'right_tree': right_tree}
```

⑤ 做出预测

```python
def predict(self, X):
        # X: [n_samples_test, n_features],
        # return: y: [n_samples_test, ]
        X = np.array(X)
        y = np.zeros(X.shape[0])

        for i in range(X.shape[0]):
            node = self.tree
            while 'feature' in node:
                if X[i][node['feature']] < node['threshold']:
                    node = node['left_tree']
                else:
                    node = node['right_tree']
            y[i] = node['threshold']
        return y
```

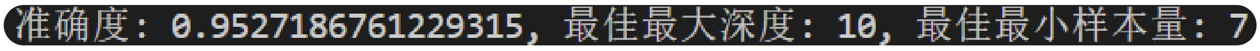
3 调试参数

```
accuracy_init = 0
    for i in range(5, 12):
        max_depth = i
        for j in range(2, 10):
            min_samples_leaf = j
            clf.fit(X_train, y_train, max_depth=max_depth,
min_samples_leaf=min_samples_leaf)
            y_pred = clf.predict(X_test)
            if accuracy(y_test, y_pred) > accuracy_init:
                accuracy_init = accuracy(y_test, y_pred)
                max_depth_best = max_depth
                min_samples_leaf_best = min_samples_leaf

print(f"准确度: {accuracy_init}, 最佳最大深度: {max_depth_best}, 最佳最小样本
量: {min_samples_leaf_best}")
```

最大深度从5到11，最小样本量从2到9。

最大深度为最佳准确率和最佳参数如下:



```
准确度: 0.9527186761229315, 最佳最大深度: 10, 最佳最小样本量: 7
```

2. PCAKMeans

1 PCA

1 核函数:

```python
def get_kernel_function(kernel):
    if kernel == "rbf":
        def rbf_kernel(x, y, gamma=1.0):
            return np.exp(-gamma * np.linalg.norm(x - y) ** 2)
        return rbf_kernel
    # 这里使用线性核
    elif kernel == "linear":
        def linear_kernel(x, y):
            return np.dot(x, y)
        return linear_kernel
    else:
        raise ValueError("Unsupported kernel")
```

2. 主成分分析：

```python
def fit(self, X: np.ndarray):
        # X: [n_samples, n_features]
        n_samples = X.shape[0]
        K = np.zeros((n_samples, n_samples))

        for i in range(n_samples):
            for j in range(n_samples):
                K[i, j] = self.kernel_f(X[i], X[j])

        # 中心化
        one_n = np.ones((n_samples, n_samples)) / n_samples
        K_centered = K - one_n @ K - K @ one_n + one_n @ K @ one_n
        self.K = K_centered

        # Eigenvalue decomposition
        eigenvalues, eigenvectors = np.linalg.eig(K_centered)
        idx = eigenvalues.argsort()[::-1]

        self.eigenvectors = eigenvectors[:, idx[:self.n_components]]
        self.eigenvalues = eigenvalues[idx[:self.n_components]]
```

3. 投影： $x'_j = \frac{1}{\sqrt{\lambda}}[K(x_1, x_j), \ldots, K(x_n, x_j)] \cdot u$

```python
def transform(self, X: np.ndarray):
        # X: [n_samples, n_features]
        self.fit(X)
        X_reduced = (self.K @ self.eigenvectors) /
np.sqrt(self.eigenvalues[:self.n_components])

        return X_reduced
```

2. KMeans
   1. 初始化。
   2. 计算每个点到各个中心的距离，并分配到最近的聚类。

```python
def assign_points(self, points):
        n_samples = points.shape
        self.labels = np.zeros(n_samples, dtpe = int)
        for i in range(n_samples):
            distances = np.linalg.norm(points[i] - self.centers,
axis=1)
            self.labels[i] = np.argmin(distances)
        return self.labels
```

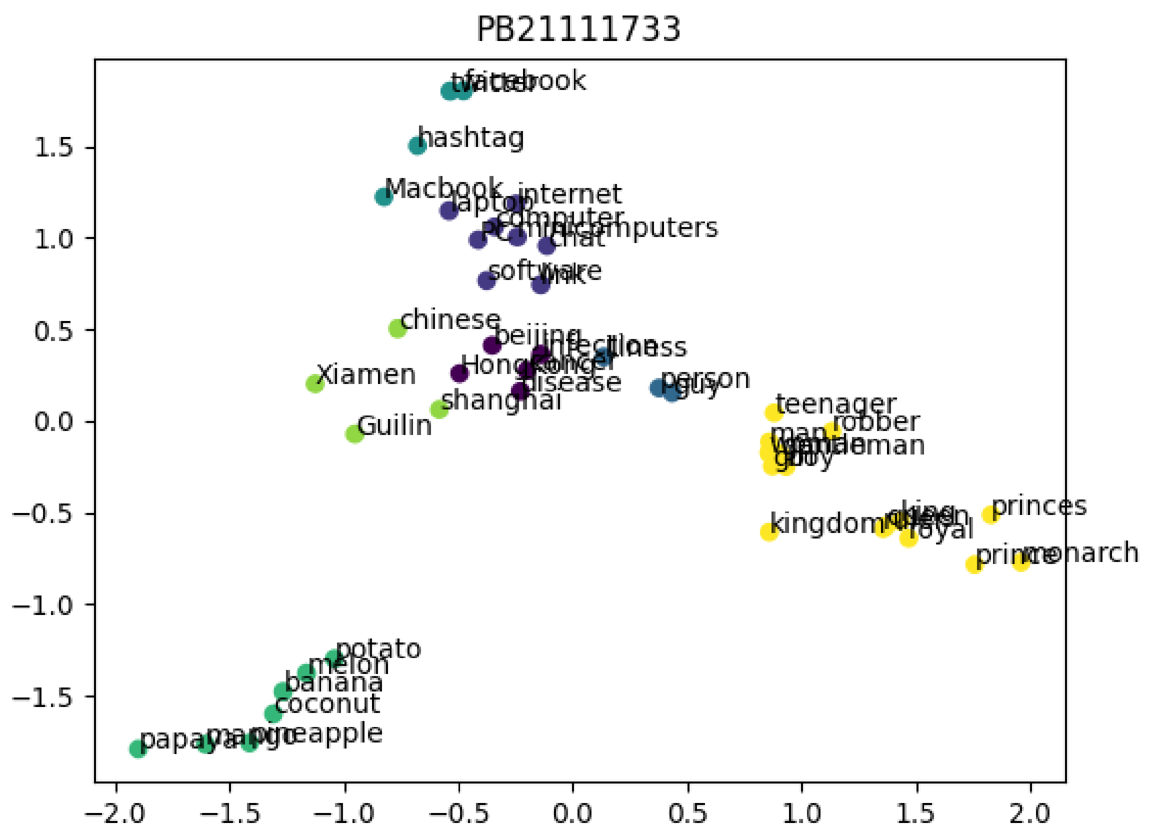
   3. 更新聚类中心。

```python
def update_centers(self, points):
        for i in range(self.k):
            cluster_points = points[self.labels == i]
            if len(cluster_points) > 0:
                self.centers[i] = cluster_points.mean(axis=0)
            else:
                self.centers[i] =
points[np.random.choice(len(points))]
```

   4. KMeans执行。（使用 `allclose` 判断两个数组的元素是否逐一接近）

```python
def fit(self, points):
        self.centers = self.initialize_centers(points)
        for _ in range(self.max_iter):
            previous_centers = self.centers.copy()
            self.labels = self.assign_points(points)
            self.update_centers(points)
            if np.allclose(previous_centers, self.centers):
                break
```

3 测试：分解到2维，使用`linear`核。7个聚类。


PB21111733
twitter
facebook
hashtag
Macbook
internet
laptop
computer
PC
computers
chat
software
link
chinese
beijing
infection
illness
Xiamen
HongKong
cancer
disease
person
guy
shanghai
Guilin
teenager
robber
man
woman
gentleman
girl
boy
princes
kingdom
king
queen
royal
prince
monarch
potato
melon
banana
coconut
papaya
mango
pineapple