P.4. Noise-Removal and Classification

Machine Learning Tasks

- Many algorithms are sensitive to **outliers** or **noise**:
  - An object with extreme values may substantially distort the distribution of the data.
- A **good dataset** is often better than a **good algorithm**.

Project Objectives

- **Develop** efficient **noise-removal algorithms**,
  - using e.g., the k-NN and the Clustering-PCA.
- **Merge** the noise-removal algorithms to **classification**.
- **Test and tune** the resulting algorithms for public-domain datasets.

For each of selected datasets, you will design the best model for noise-removal and classification.

Confidence Region

A **confidence score** indicates the **likelihood** that a machine learning model assigns the respective intent correctly.

Definition P.6. A **confidence region** is the region where a new point belongs to a specified class, given a confidence score/value.

Review: k-NN and PCA

k-Nearest Neighbors (k-NN)

Algorithm 5.37, p. 137. (**k-NN algorithm**). The algorithm itself is fairly straightforward and can be summarized by the following steps:

1. Choose the number k and a distance metric.
2. For the new sample, find the k-nearest neighbors.
3. Assign the class label by majority vote.

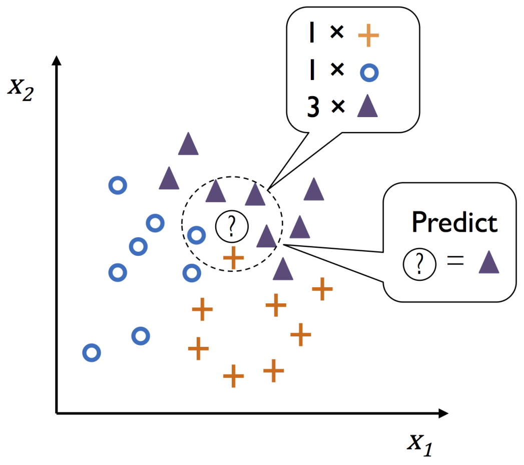

Figure 5.16: Illustration for how a new data point (?) is assigned the triangle class label, based on majority voting, when $k = 5$.

Example P.7. Along with the k-NN algorithm:

- Select k.
- Set a confidence value $\xi \le k$.

Then the **confidence region** for a class can be defined as the region where the k-NN of a point includes at least ξ points from the same class. **For example, $k = 5$ and $\xi = 4$.**

Remark P.8. Rather than counting (a constant weighting), an **IDW** may be incorporated; the goal is to keep **grouped data points**.

PCA

Recall: **(PCA), p. 197**. Consider a **data matrix** $X \in \mathbb{R}^{N\times d}$:
- ∘ each of the N rows represents a different data point,
- ∘ each of the d columns gives a particular kind of feature, and
- ∘ each column has zero empirical mean (e.g., after standardization).

- The goal of the standard PCA is to find an **orthogonal** weight matrix $W_k \in \mathbb{R}^{d\times k}$ such that
$$Z_k = X\, W_k, \quad k \le d, \tag{P.4.1}$$
where $Z_k \in \mathbb{R}^{N\times k}$ is call the **truncated score matrix** and $Z_d = Z$. Columns of Z represent the **principal components** of X.
- (Claim 7.3, p. 160). The transformation matrix W_k turns out to be the collection of normalized eigenvectors of X^TX:
$$W_k = [\mathbf{w}_1|\mathbf{w}_2|\cdots|\mathbf{w}_k], \quad (X^TX)\,\mathbf{w}_j = \lambda_j\,\mathbf{w}_j, \;\; \mathbf{w}_i^T\mathbf{w}_j = \delta_{ij}, \tag{P.4.2}$$
where $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_k \ge 0$.
- (Remark 7.4, p. 160). The matrix $Z_k \in \mathbb{R}^{N\times k}$ is scaled eigenvectors of XX^T:
$$Z_k = [\sqrt{\lambda_1}\,\mathbf{u}_1|\sqrt{\lambda_2}\,\mathbf{u}_2|\cdots|\sqrt{\lambda_k}\,\mathbf{u}_k], \quad (XX^T)\,\mathbf{u}_j = \lambda_j\,\mathbf{u}_j, \;\; \mathbf{u}_i^T\mathbf{u}_j = \delta_{ij}. \tag{P.4.3}$$
- A **data (row) vector** x **(new or old)** is transformed to a k-dimensional row vector of principal components
$$\mathbf{z} = \mathbf{x}W_k \in \mathbb{R}^{1\times k}. \tag{P.4.4}$$
- (Remark 7.5, p. 161). Let $\boldsymbol{X = U\,\Sigma V^T}$ be the **SVD** of X, where
$$\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \cdots, \sigma_d), \quad \sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_d \ge 0.$$
Then,
$$\begin{aligned} &V \cong W; \quad \sigma_j^2 = \lambda_j, \;\; j = 1, 2, \cdots, d, \\ &Z_k = [\sigma_1\,\mathbf{u}_1|\sigma_2\,\mathbf{u}_2|\cdots|\sigma_k\,\mathbf{u}_k]. \end{aligned} \tag{P.4.5}$$

Geometric Interpretation of PCA

Example P.9. Consider the following synthetic dataset.

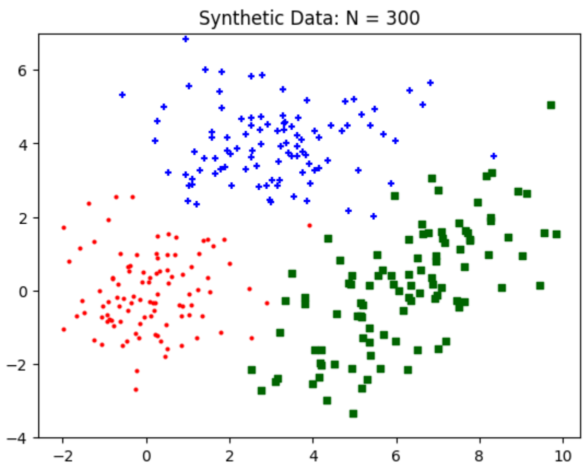

Figure P.5: A synthetic dataset of three classes.

- For each class, one may perform PCA along with the SVD.

```
for c in range(nclass):
    Xc = X[y==c]; CC = np.mean(Xc,axis=0)
    U, s, VT = svd(Xc-CC,full_matrices=False)
```

- Let $\boldsymbol{\mu}^{(c)} = \texttt{CC[c]}$, $V^{(c)} = [\mathbf{v}_1^{(c)}, \cdots, \mathbf{v}_d^{(c)}]$, and $\sigma_j^{(c)} = \texttt{s[j]}$, the jth singular value for Class c. Define an **anisotropic distance** as

$$\gamma^{(c)}(\mathbf{x}) = \sum_{j=1}^{d} \left(\frac{(\mathbf{x} - \boldsymbol{\mu}^{(c)}) \cdot \mathbf{v}_j^{(c)}}{\sigma_j^{(c)}} \right)^2. \tag{P.4.6}$$

  - It is implemented in the function `aniso_dist2`, in `util_PCA.py`.
  - For $r > 0$, $\gamma^{(c)}(\mathbf{x}) = r^2$ assigns an **ellipse**. □

Definition P.10. The **minimum-volume enclosing ellipsoid (MVEE)** is the ellipsoid of smallest volume that fully contains all the objects.

Remark P.11. Let $r^{(c)}_{\max}$ be

$$r^{(c)}_{\max}{}^{2} = \max_{\mathbf{x}\in X^{(c)}} \gamma^{(c)}(\mathbf{x}). \tag{P.4.7}$$

Then $\gamma^{(c)}(\mathbf{x}) = r^{(c)}_{\max}{}^{2}$ approximates the MVEE relatively well.
See Figure P.6 (a).

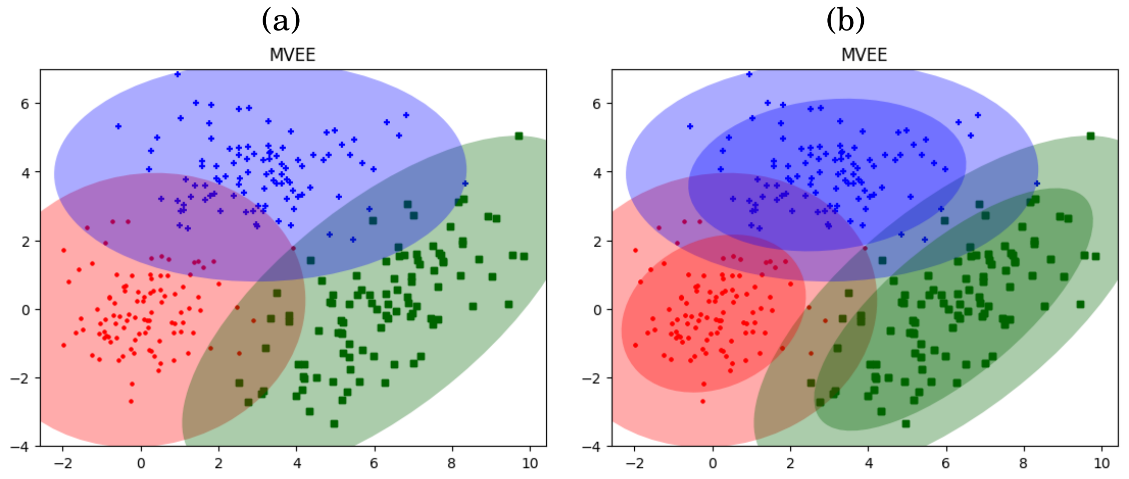

Figure P.6: Approximate MVEEs for: (a) the dataset and (b) the confidence regions.

Example P.12. Along with PCA:

- Either set a threshold $\theta > 0$
- or a portion $0 < p < 1$.

The **confidence region** for a class can be defined as the region where

(a) either the points x satisfy $\gamma^{(c)}(\mathbf{x}) \le \theta$
(as a result of a **histogram analysis**)

(b) or only the p-portion of the near-center points are picked from the dataset ordered by the anisotropic distances.

Figure P.6(b) shows the confidence regions, for $p = 0.9$.

Note: You must **first find confidence regions** for the training dataset, which can be viewed as **denoising**. You may **then begin the training step with the denoised dataset.**

What to do

1. Download `PCA-KNN-Denoising.PY.tar`:
   https://skim.math.msstate.edu/LectureNotes/data/PCA-KNN-Denoising.PY.tar
2. Compose a **denoising-and-classification** code, using appropriate functions from the downloaded package.
   - You must implement both denoising algorithms: the k-NN-based and the PCA-based.
3. Use similar datasets, utilized for **Project 1. mCLESS**, Section P.3:
   - Select a **synthetic dataset**, using Line 7 or 8 in `GLOBAL_VARIABLES.py`.
   - **Real datasets**. Use public datasets such as `iris` and `wine`.
     To get the public datasets, you may use:
     ```
     from sklearn import datasets
     data_read1 = datasets.load_iris()
     data_read2 = datasets.load_wine()
     ```
4. Compare performances of the classifiers **with and without denoising**
   - `LogisticRegression(max_iter = 1000)`
   - `KNeighborsClassifier(5)`
   - `SVC(gamma=2, C=1)`
   - `RandomForestClassifier(max_depth=5, n_estimators=50, max_features=1)`
5. (**Optional for Undergraduate Students**)
   Add modules for **clustering-and-PCA denoising**.
   - For example, the MVEE does not make sense for a half-moon dataset.
   - Add another dataset, such as
     ```
     from sklearn.datasets import make_moons
     X, y = make_moons(noise=0.2, n_samples=400, random_state=12)
     ```
   - Preform k-Means cluster analysis for each class, with $k = 4$.
   - For each cluster in each class, perform the PCA-based denoising.
   - Carry out Steps 2–4.
6. Report your experiments with the code and results.

Note: You did already the portion: "without denoising". Undergraduate students may consider that a class is a cluster. For graduate students, Step 5 will be worth 40% your score.