Linear Discriminant Analysis

Đồ án CS116 - Lập trình Python cho Máy học

Giảng viên:	TS. Nguyễn Vinh Tiệp
Thành viên:	
Trương Chí Diễn	19520464
Trần Hoàn Đức Duy	19521434
Nguyễn Anh Dũng	19521394

Khoa khoa học máy tính
Đại học Công nghệ thông tin
Đại học Quốc gia thành phố Hồ Chí Minh

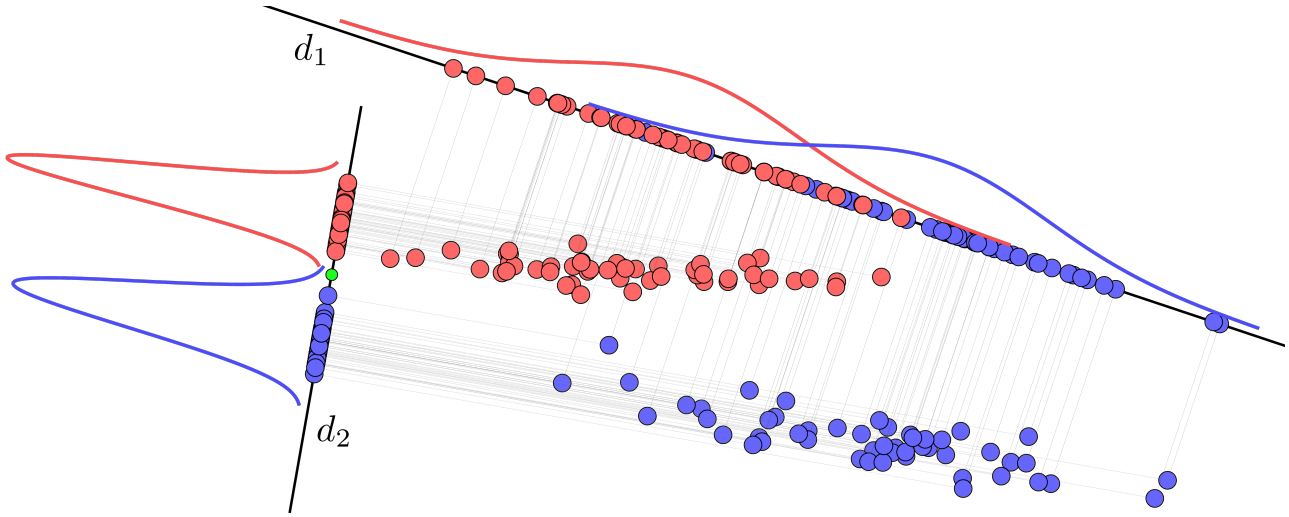
Mục lục

1 Giới thiệu bài toán

Phân tích phân biệt tuyến tính (LDA) là một phương pháp được sử dụng trong thống kê và các lĩnh vực khác, để tìm một tổ hợp tuyến tính của các đối tượng đặc trưng hoặc phân tách hai hoặc nhiều lớp của các đối tượnn. Tổ hợp thu được có thể được sử dụng như 1 bộ phân loại tuyến tính (linear classifier) hoặc phổ biến hơn là để giảm kích thước trước khi phân loại sau này.

LDA cũng liên quan chặt chẽ đến phân tích thành phần chính (PCA) và phân tích nhân tố (factor analysis) ở chỗ cả ba đều tìm kiếm các tổ hợp tuyến tính của các biến mô tả tốt nhất cho dữ liệu. LDA cố gắng mô hình hóa sự khác biệt giữa các lớp dữ liệu một cách rõ ràng. Ngược lại, PCA không tính đến bất kỳ sự khác biệt nào về lớp và phân tích nhân tố xây dựng các tổ hợp đặc trưng dựa trên sự khác biệt hơn là tương đồng. LDA cũng khác với phân tích nhân tố ở chỗ nó không phải là một kỹ thuật phụ thuộc lẫn nhau: phải thực hiện sự phân biệt giữa các biến độc lập và các biến phụ thuộc (còn gọi là biến tiêu chí).

Từ 'Discriminant' được hiểu là những thông tin đặc trưng cho mỗi class, khiến nó không bị lẫn với các classes khác. Từ 'Linear' được dùng vì cách giảm chiều dữ liệu được thực hiện bởi một ma trận chiếu (projection matrix), là một phép biến đổi tuyến tính (linear transform).



Hình 1: Chiếu dữ liệu lên các đường thẳng khác nhau .Phương của d_1 gần giống với phương của thành phần chính thứ nhất của dữ liệu, phương của d_2 gần với thành phần phụ của dữ liệu nếu dùng PCA. Nguồn Machine Learning cơ bản

2 LDA cho bài toán phân loại 2 lớp

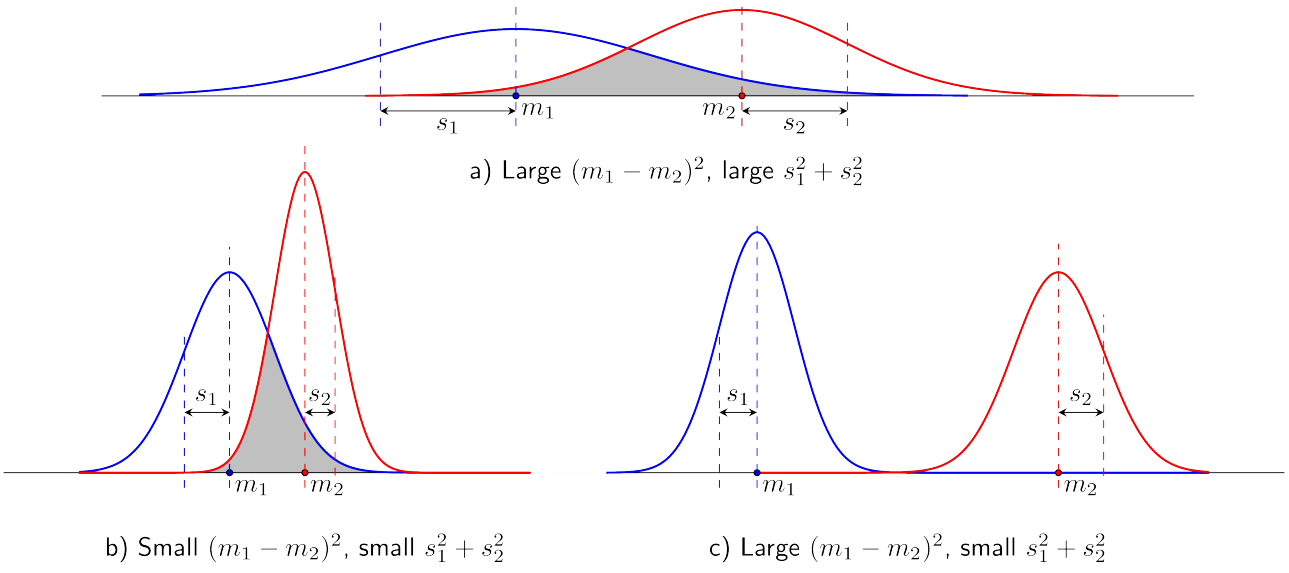
2.1 Ý tưởng cơ bản

Cùng phân tích ý tưởng cho bài toán classification cơ bản với 2 lớp.

Trong Hình 1, khi chiếu các điểm dữ liệu lên d_1 ta giữ được phân phối xác xuất của các điểm dữ liệu trong từng lớp. Tuy nhiên, do độ lệch chuẩn lớn nên 2 phân phối này trộn lẫn vào nhau rất nhiều. Việc bảo toàn phân phối xác xuất không thực sự quan trọng trong bài toán classification. Do đó, việc chiếu dữ liệu lên d_2 tuy khiến phân phối dữ liệu bị thay đổi nhưng giúp ích hơn rất nhiều khi có thể phân tách 2 lớp dữ liệu ra rõ rệt.

Hình 2a) Cả hai class đều quá phân tán khiến cho tỉ lệ chồng lấn là lớn, tức dữ liệu chưa thực sự discriminative.

Hình 2b) Là trường hợp khi độ lệch chuẩn của hai class đều nhỏ nhưng khoảng cách giữa hai class, được đo bằng khoảng cách giữa hai kỳ vọng m_1 và m_2, là quá nhỏ, khiến cho phần chồng lấn cũng chiếm môt tỉ lệ lớn,



Hình 2: Khoảng cách giữa các kỳ vọng và tổng các phương sai ảnh hưởng tới độ discriminant của dữ liệu. Nguồn: Machine Learning cơ bản

không tốt cho classification.

Hình 2c) Là trường hợp khi hai độ lệch chuẩn là nhỏ và khoảng cách giữa hai kỳ vọng là lớn, phần chồng lấn nhỏ không đáng kể.

Độ lệch chuẩn và khoảng cách giữa hai kỳ vọng đại diện cho các tiêu chí:

- Độ lệch chuẩn nhỏ thể hiện việc dữ liệu ít phân tán. Điều này có nghĩa là dữ liệu trong mỗi class có xu hướng giống nhau. Hai phương sai s_1^2, s_2^2 còn được gọi là các **within-class variances**.
- Khoảng cách giữa các kỳ vọng là lớn chứng tỏ rằng hai classes nằm xa nhau, tức dữ liệu giữa các classes là khác nhau nhiều. Bình phương khoảng cách giữa hai kỳ vọng $(m_1 - m_2)^2$ còn được gọi là **between-class variance**.

Hai classes được gọi là discriminative nếu hai class đó cách xa nhau (**between-class variance**) và dữ liệu trong mỗi class có xu hướng giống nhau (**within-class variances** nhỏ). Linear Discriminant Analysis là thuật toán đi tìm một phép chiếu sao cho tỉ lệ giữa between-class variance và within-class variance lớn nhất có thể.

2.2 Xây dựng hàm mục tiêu

Giả sử rằng có N điểm dữ liệu $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ trong đó $N_1 < N$ điểm đầu tiên thuộc class thứ nhất, $N_2 = N - N_1$ điểm cuối cùng thuộc class thứ hai. Ký hiệu $\mathcal{C}_1 = \{n | 1 \leq n \leq N_1\}$ là tập hợp các chỉ số của các điểm thuộc class 1 và $\mathcal{C}_2 = \{m | N_1 + 1 \leq m \leq N\}$) là tập hợp các chỉ số của các điểm thuộc class 2. Phép chiếu dữ liệu xuống 1 đường thẳng có thể được mô tả bằng một vector hệ số $\mathbf{w}$, giá trị tương ứng của mỗi điểm dữ liệu mới được cho bởi:

$$y_n = \mathbf{w}^T \mathbf{x}_n, 1 \leq n \leq N$$

Vector kỳ vọng của mỗi class:

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n, \quad k = 1, 2 \tag{1}$$

Khi đó:

$$m_1 - m_2 = \frac{1}{N_1}\sum_{i\in\mathcal{C}_1} y_i - \frac{1}{N_2}\sum_{j\in\mathcal{C}_2} y_j = \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2) \tag{2}$$

Các **within-class variances** được định nghĩa là:

$$s_k^2 = \sum_{n\in\mathcal{C}_k} (y_n - m_k)^2, \quad k = 1, 2 \tag{3}$$

LDA là thuật toán đi tìm giá trị lớn nhất của hàm mục tiêu:

$$J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} \tag{4}$$

Với tử số:

$$(m_1 - m_2)^2 = \mathbf{w}^T \underbrace{(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T}_{\mathbf{S}_B} \mathbf{w} = \mathbf{w}^T\mathbf{S}_B\mathbf{w} \tag{5}$$

$\mathbf{S}_B$ còn được gọi là **between-class covariance matrix**.

Với mẫu số:

$$s_1^2 + s_2^2 = \sum_{k=1}^{2}\sum_{n\in\mathcal{C}_k} \left(\mathbf{w}^T(\mathbf{x}_n - \mathbf{m}_k)\right)^2 = \mathbf{w}^T \underbrace{\sum_{k=1}^{2}\sum_{n\in\mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T}_{\mathbf{S}_W} \mathbf{w} = \mathbf{w}^T\mathbf{S}_W\mathbf{w} \tag{6}$$

$\mathbf{S}_W$ còn được gọi là **within-class covariance matrix**.

Như vậy, bài toán tối ưu cho LDA trở thành:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} \tag{7}$$

2.3 Nghiệm của bài toán tối ưu

Nghiệm $\mathbf{w}$ của (7) sẽ là nghiệm của phương trình đạo hàm hàm mục tiêu bằng 0. Sử dụng chain rule cho đạo hàm hàm nhiều biến ta có:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{(\mathbf{w}^T\mathbf{S}_W\mathbf{w})^2}\left(2\mathbf{S}_B\mathbf{w}(\mathbf{w}^T\mathbf{S}_W\mathbf{w}) - 2\mathbf{w}^T\mathbf{S}_B\mathbf{w}^T\mathbf{S}_W\mathbf{w}\right) = \mathbf{0} \tag{8}$$

$$\Leftrightarrow \mathbf{S}_B\mathbf{w} = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}\mathbf{S}_W\mathbf{w} \tag{9}$$

$$\Leftrightarrow \mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = J(\mathbf{w})\mathbf{w} \tag{10}$$

Vậy, để hàm mục tiêu là lớn nhất thì $J(\mathbf{w})$ chính là trị riêng lớn nhất của $\mathbf{S}_W^{-1}\mathbf{S}_B$. Dấu bằng xảy ra khi $\mathbf{w}$ là vector riêng ứng với trị riêng lớn nhất đó.

Nếu $\mathbf{w}$ là nghiệm của thì $k\mathbf{w}$ cũng là nghiệm với k là số thực khác không bất kỳ. Ta có thể chọn $\mathbf{w}$ sao cho $(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{w} = J(\mathbf{w}) = L =$ trị riêng lớn nhất của $\mathbf{S}_W^{-1}\mathbf{S}_B$. Khi đó, thay định nghĩa của $\mathbf{S}_B$ ở vào ta có:

$$L\mathbf{w} = \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)\underbrace{(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{w}}_{L} = L\mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

Điều này có nghĩa là ta có thể chọn:

$$\mathbf{w} = \alpha\mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2) \tag{11}$$

với $\alpha \neq 0$ bất kỳ.

3 LDA cho bài toán phân loại đa lớp

3.1 Xây dựng hàm mất mát

Chúng ta sẽ xem xét trường hợp tổng quát khi có nhiều hơn 2 lớp. Giả sử rằng chiều của dữ liệu D lớn hơn số lượng classes C. Giả sử rằng chiều mà chúng ta muốn giảm về là $D' < D$ và dữ liệu mới ứng với mỗi điểm dữ liệu x là:

$$\mathbf{y} = \mathbf{W}^T\mathbf{x}$$

với $\mathbf{W} \in R^{D\times D'}$

Một vài ký hiệu:

- $\mathbf{W} \in R^{D\times D'}$ lần lượt là ma trận dữ liệu của class **k** trong không gian ban đầu và không gian mới với số chiều nhỏ hơn.
- $\mathbf{m}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{x}_k \in R^D$ là vector kỳ vọng của class k trong không gian ban đầu.
- $\mathbf{e}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{y}_n = \mathbf{W}^T\mathbf{m}_k \in R^{D'}$ là vector kỳ vọng của class k trong không gian mới.
- $\mathbf{m}$ là vector kỳ vọng của toàn bộ dữ liệu trong không gian ban đầu và e là vector kỳ vọng trong không gian mới.

3.1.1 Within-class nhỏ

Within-class variance của class k có thể được tính như sau:

$$\sigma_k^2 = \sum_{n\in\mathcal{C}_k}||\mathbf{y}_n - \mathbf{e}_k||_F^2 = ||\mathbf{Y}_k - \mathbf{E}_k||_2^2 = trace\left(\mathbf{W}^T(\mathbf{X}_k - \mathbf{M}_k)(\mathbf{X}_k - \mathbf{M}_k)^T\mathbf{W}\right) \tag{12}$$

Với $\mathbf{E}_k$ một ma trận có các cột giống hệt nhau và bằng với vector kỳ vọng $\mathbf{e}_k$. Có thể nhận thấy $\mathbf{E}_k = \mathbf{W}^T\mathbf{M}_k$ với $\mathbf{M}_k$ là ma trận có các cột giống hệt nhau và bằng với vector kỳ vọng $\mathbf{m}_k$ trong không gian ban đầu.

Vậy đại lượng đo **within-class** trong multi-class LDA có thể được đo bằng:

$$s_W = \sum_{k=1}^{C}\sigma_k^2 = \sum_{k=1}^{C}trace\left(\mathbf{W}^T(\mathbf{X}_k - \mathbf{M}_k)(\mathbf{X}_k - \mathbf{M}_k)^T\mathbf{W}\right) = trace\left(\mathbf{W}^T\mathbf{S}_W\mathbf{W}\right) \tag{13}$$

Với

$$\mathbf{S}_W = \sum_{k=1}^{C} ||\mathbf{X}_k - \mathbf{M}_k||_F^2 = \sum_{k=1}^{C} \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T \tag{14}$$

và nó có thể được coi là **within-class covariance matrix của multi-class LDA**.

3.1.2 Between-class lớn

Between-class lớn có thể đạt được nếu tất cả các điểm trong không gian mới đều xa vector kỳ vọng chung $\mathbf{e}$. Vậy ta có thể định nghĩa đại lượng **between-class** như sau:

$$s_B = \sum_{k=1}^{C} N_k ||\mathbf{e}_k - \mathbf{e}||_F^2 = \sum_{k=1}^{C} ||\mathbf{E}_k - \mathbf{E}||_F^2 \tag{15}$$

Lập luận tương tự như (13) có thể chứng minh được:

$$s_B = trace\left(\mathbf{W}^T \mathbf{S}_B \mathbf{W}\right) \tag{16}$$

với:

$$\mathbf{S}_B = \sum_{k=1}^{C} (\mathbf{M}_k - \mathbf{M})(\mathbf{M}_k - \mathbf{M})^T = \sum_{k=1}^{C} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T \tag{17}$$

3.2 Hàm mất mát cho multi-class LDA

Với cách định nghĩa và ý tưởng về **within-class** nhỏ và **between-class** lớn như trên, ta có thể xây dựng bài toán tối ưu:

$$\mathbf{W} = \arg\max_{\mathbf{W}} J(\mathbf{W}) = \arg\max_{\mathbf{W}} \frac{trace(\mathbf{W}^T \mathbf{S}_B \mathbf{W})}{trace(\mathbf{W}^T \mathbf{S}_W \mathbf{W})}$$

Với cách tính tương tự như (8) - (10) ta có:

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = \frac{2\left(\mathbf{S}_B \mathbf{W} trace(\mathbf{W}^T \mathbf{S}_W \mathbf{W}) - trace(\mathbf{W}^T \mathbf{S}_B \mathbf{W}) \mathbf{S}_W \mathbf{W}\right)}{\left(trace(\mathbf{W}^T \mathbf{S}_W \mathbf{W})\right)^2} = \mathbf{0} \tag{18}$$

$$\Leftrightarrow \mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{W} = J\mathbf{W} \tag{19}$$

4 Hyperparameter

Có 3 phương pháp chính được sử dụng trong thư viện Sklearn để tính toán LDA, được chỉ định bằng tham số **solver** và các tham số khác có thể tùy chỉnh dựa vào **solver** được mô tả như sau:

- **solver: 'svd', 'lsqr', 'eigen', mặc định='svd'**:
  - 'svd': sử dụng Singular Value Decomposition. Dữ liệu cần có số lượng lớn đặc trưng.
  - 'lsqr': dùng thuật giải Least Squares. Có thể kết hợp với **shrinkage** hoặc **covariance estimator** tự chọn.

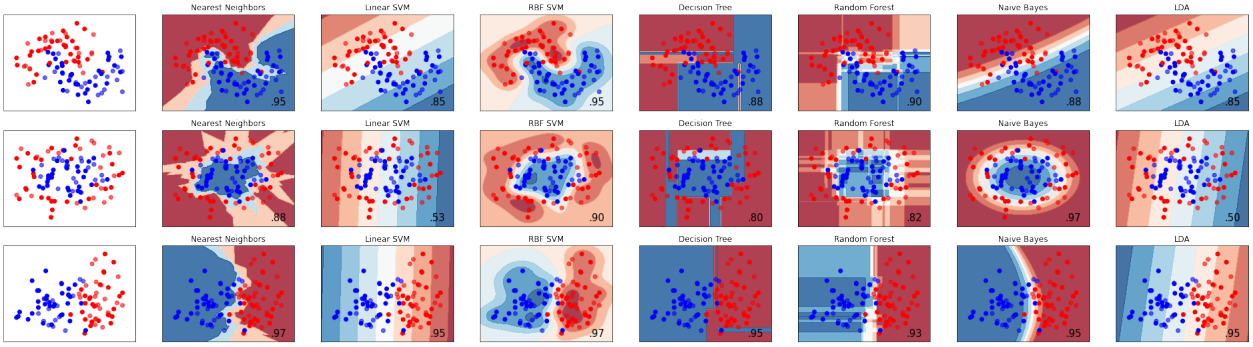
  - 'eigen': dùng phương pháp eigen decomposition (được trình bày phía trên). Có thể kết hợp với **shrinkage** hoặc **covariance estimator** tự chọn.

- **shrinkage: 'auto' hoặc số float, mặc định=None**: dùng khi dữ liệu có số feature quá ít, ước tính covariance matrix thực nghiệm có thể cho kết quả không tốt.
  - None: không dùng.
  - 'auto': được tự động lựa chọn dựa trên bổ đề Ledoit-Wolf.
  - số float từ 0 đến 1: 0 là không dùng.
- **priors: mảng có dạng (số lớp,), mặc định=None** dùng xác định phân phối lớp. Mặc định được suy ra từ dữ liệu huấn luyện.
- **n_components: int, mặc định=None** số chiều giữ lại (≤min(số lớp-1, số đặc trưng).
- **store_covariance: bool, mặc định=False** tính toán rõ covariance matrix và lưu lại.
- **tol: float, mặc định=1.0e-4** ngưỡng tuyệt đối cho giá trị singular X được xem là quan trọng, dùng để ước tính hạng của X. Chiều của giá trị không quan trọng bị loại bỏ. Chỉ sử dụng khi **solver** là 'svd'.
- **covariance_estimator: đối tượng covariance, default=None** xác định phương pháp tính covariance matrix tùy chỉnh được sử dụng thay cho empirical covariance. Chỉ sử dụng với 'lsqr', 'eigen'.

5 Thực nghiệm

5.1 Sử dụng cho bài toán classification

Sinh ra một tập dữ liệu ngẫu nhiên, sau đó dùng các thuật toán phân loại khác nhau để đánh giá độ hiệu quả.



Hình 3: So sánh LDA và các thuật toán classification khác. Số bên góc phải dưới là accuracy của mộ hình

Có thể thấy LDA thể hiện tốt trên bộ dữ liệu có dạng Linear separability nhưng không thể phân loại tốt trên các dữ liệu None-linear separability.

5.2 Sử dụng để giảm chiều dữ liệu

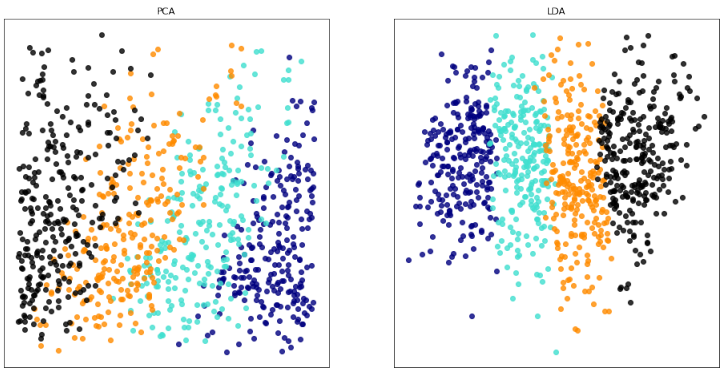
Sử dụng Principal component analysis(PCA), một thuật toán giảm chiều dữ liệu được sử dụng rộng rãi để so sánh với LDA trên bộ dữ liệu Mobile Price - Kaggle, bộ dữ liệu bao gồm 20 features và 4 classes.

Kết quả cho thấy LDA có thể biểu diễn dữ liệu hiệu quả hơn PCA,dù vẫn bị overlap giữa các vùng dữ liệu tuy nhiên các điểm dữ liệu được phân chia khá rõ ràng.

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
0	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	19	0	0	1	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7	1	1	0	2
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9	1	1	0	2
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11	1	0	0	2
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15	1	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	794	1	0.5	1	0	1	2	0.8	106	6	14	1222	1890	668	13	4	19	1	1	0	0
1996	1965	1	2.6	1	0	0	39	0.2	187	4	3	915	1965	2032	11	10	16	1	1	1	2
1997	1911	0	0.9	1	1	1	36	0.7	108	8	3	868	1632	3057	9	1	5	1	1	0	3
1998	1512	0	0.9	0	4	1	46	0.1	145	5	5	336	670	869	18	10	19	1	1	1	0
1999	510	1	2.0	1	5	1	45	0.9	168	6	16	483	754	3919	19	4	2	1	1	1	3

2000 rows × 21 columns

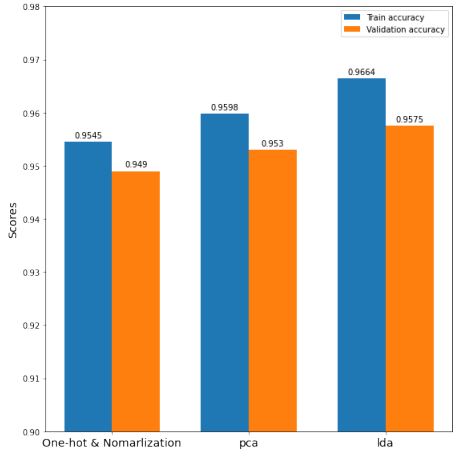
Hình 4: Bộ dữ liệu Mobile Price



Hình 5: Sử dụng thuật toán PCA và LDA để giảm chiều dữ liệu cho dataset

5.3 Sử dụng để giảm chiều dữ liệu sau đó huấn luyện một mô hình phân lớp

Bộ dữ liệu Mobile Price vẫn sẽ được tiếp tục sử dụng ở phần này. Để xem xét lợi ích và bất lợi của thuật toán giảm chiều dữ liệu LDA, dữ liệu sẽ được huấn luyện với mô hình Support vector machine(SVM) với 3 cách xử lý dữ liệu, đầu tiên là sử dụng One-hot encoder và normalization, thứ hai là PCA dùng để giảm chiều dữ liệu xuống còn 6, cuối cùng là LDA giảm chiều dữ liệu xuống còn 3.



Hình 6: Accuracy scores khi sử dụng 3 cách xử lý dữ liệu trên tập Mobile Price

Để đảm bảo tính công bằng của kết quả, thực nghiệm đã được thực hiện trên nhiều tham số và sử dụng K-fold cross validation. Cuối cùng, LDA cho kết quả tốt nhất so với hai phương pháp còn lại, qua đó có thể nhận xét rằng không phải việc giữ lại thông tin nhiều nhất sẽ luôn mang lại kết quả tốt nhất. Tuy nhiên LDA chỉ có thể thực hiện hiệu quả trên các bộ dữ liệu có số lượng class nhiều.

5.4 Điều chỉnh tham số

Các tham số ở phần 4 được điều chỉnh để tìm cấu hình tốt nhất cho mô hình. Bộ dữ liệu Mobile Price tiếp tục được xử lý bằng LDA sau đó cho qua một SVM để phân loại.

index	solver	n_components	shrinkage	train accuracy	test accuracy
0	svd	1	None	0.9551	0.9515
1	svd	2	None	0.9611	0.9565
2	svd	3	None	**0.9654**	**0.9580**
3	eigen	1	auto	0.9210	0.9195
4	eigen	1	None	0.9551	0.9515
5	eigen	2	auto	0.9410	0.9335
6	eigen	2	None	0.9591	0.9530
7	eigen	3	auto	0.9460	0.9415
8	eigen	3	None	0.9619	0.9550

Bảng 1: Kết quả quá trình điều chỉnh tham số sử dụng LDA để xử lý dữ liều và SVM làm mô hình phân loại. Phần in đậm thể hiện kết quả tốt nhất

6 Kết luận

- LDA là một phương pháp giảm chiều dữ liệu có sử dụng thông tin về label của dữ liệu. LDA là một thuật toán supervised.
- LDA phù hợp với nhưng bộ dữ liệu có thể phân biệt tuyến tính. Phương pháp này hiệu quả hơn PCA trong bài toán classification.
- Số chiều dữ liệu ở không gian mới khi sử dụng LDA luôn phải nhỏ hơn số lượng lớp, điều này khác biệt so với PCA.
- Các kết quả thực nghiệm cho thấy LDA tuy không hoạt động tốt như một mô hình phân lớp nhưng hoạt động tốt hơn PCA như một phương pháp giảm chiều dữ liệu nếu dữ liệu có gán nhãn.
- Ngoài ra còn còn thể nhận định không phải việc giữ lại thông tin nhiều nhất sẽ luôn mang lại kết quả tốt nhất.

Tài liệu

- Linear Discriminanat Analysis
- Bài 29: Linear Discriminant Analysis - Machine learning cơ bản
- Mobile price dataset