Mathematical techniques in data science

Lecture 7: Classification – Logistic regression and LDA

March 1st, 2019

- Logistic regression
- Linear Discriminant Analysis
- Nearest neighbours
- Support Vector Machines

- Next Friday (03/08): Homework 1 due
- Groups (for class projects) need to be formed by the end of next week

Class project: example

Dataset: 25,000 images of dogs and cats

Similar: Malaria cell images dataset, Sign language dataset

Class project: example

Dataset: 285,000 credit card transactions

Similar: heart disease dataset

Linear discriminant analysis

Linear discriminant analysis

Suppose

- $Y \in \{1, 2, \ldots, K\}$
- $P(Y = i) = \pi_i, \quad i = 1, 2, \ldots, K.$
- $P(X = x | Y = i) \sim f_i(x)$

Then

$$P(Y = i | X = x) = \frac{P(X = x | Y = i) P(Y = i)}{\sum_{j=1}^{K} P(X = x | Y = j) P(Y = j)}$$
$$= \frac{f_i(x)\pi_i}{\sum_{j=1}^{K} f_j(x)\pi_j}$$

Model $P(X = x | Y = i)$ using Gaussian distributions

The natural model for $f_i(x)$ is the multivariate Gaussian distribution

$$f_i(x) = \frac{1}{\sqrt{(2\pi)^p \det(\Sigma_i)}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i)}, \quad x \in \mathbb{R}^p$$

- μ: mean vector
- Σ: covariance matrix

$$\Sigma = E[(X-\mu)^T (X-\mu)]$$

LDA and QDA

The natural model for $f_i(x)$ is the multivariate Gaussian distribution

$$f_i(x) = \frac{1}{\sqrt{(2\pi)^p \det(\Sigma_i)}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)}, \quad x \in \mathbb{R}^p$$

- Linear discriminant analysis (LDA): We assume

$$\Sigma_1 = \Sigma_2 = \ldots = \Sigma_K$$

- Quadratic discriminant analysis (QDA): general cases

Parameter estimation: LDA

Suppose we have dataset $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where n_i observations have label i.

- An estimate of the class probabilities π_i

$$\hat{\pi}_i = \frac{n_i}{n}$$

- Estimate the mean vectors

$$\hat{\mu}_i = \frac{1}{n_i} \sum_{y_j = i} x_j$$

- Estimate the covariance matrix Σ

$$\hat{\Sigma} = \frac{1}{N - K} \sum_{i=1}^{K} \sum_{y_j = i} (x_j - \hat{\mu}_i)(x_j - \hat{\mu}_i)^T$$

Decision rule

Suppose we have dataset $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where n_i observations have label i.

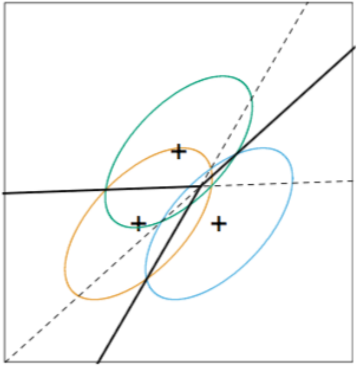
A new instance x arrives, how to predict label of x?

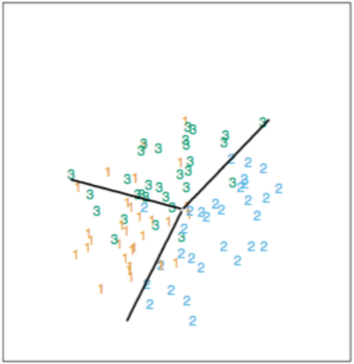
- Compute $\hat{\pi}_i$, $\hat{\mu}_i$ and $\hat{\Sigma}$
- Compute

$$P(Y = i | X = x) \approx p_k(x) = \frac{f_i(x, \hat{\mu}_i, \hat{\Sigma})\hat{\pi}_i}{\sum_{j=1}^{K} f_j(x, \hat{\mu}_j, \hat{\Sigma})\hat{\pi}_j}$$

- Bayes classifier: assign an observation to the class for which the posterior probability $p_k(x)$ is greatest.

LDA





LDA: linearity of the decision boundary

Note that

$$p_i(x) = \frac{f_i(x, \hat{\mu}_i, \hat{\Sigma})\hat{\pi}_i}{\sum_{j=1}^{K} f_j(x, \hat{\mu}_j, \hat{\Sigma})\hat{\pi}_j}$$

and

$$f_i(x) = \frac{1}{\sqrt{(2\pi)^p \det(\hat{\Sigma})}} e^{-\frac{1}{2}(x-\hat{\mu}_i)^T \Sigma^{-1}(x-\hat{\mu}_i)}, \quad x \in \mathbb{R}^p$$

Thus

$$\begin{aligned}\log \frac{p_i(x)}{p_k(x)} &= \log \frac{\hat{\pi}_i}{\hat{\pi}_k} - \frac{1}{2}(\hat{\mu}_i + \hat{\mu}_k)^T \hat{\Sigma}^{-1}(\hat{\mu}_i - \hat{\mu}_k) + x^T \hat{\Sigma}^{-1}(\hat{\mu}_i - \hat{\mu}_k) \\ &= \beta_0 + x^T \beta\end{aligned}$$

How is that different from logistic regression?

Recall that for logistic regression

$$\log \frac{P[Y=1|X=x]}{P[Y=0|X=x]} = \beta_0 + x^T\beta$$

- Both methods use linear decision boundary
- Both are simple, and often perform very well.

However

- The probability models are different
- The estimations are different

Practical problem when $n \ll p$

- Estimating covariance matrices when $n \ll p$ is challenging
- The sample covariance $\hat{\Sigma}$ is singular when $n \ll p$
- Need regularization

LDA

```
>>> import numpy as np
>>> from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> y = np.array([1, 1, 1, 2, 2, 2])
>>> clf = LinearDiscriminantAnalysis()
>>> clf.fit(X, y)
LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None,
              solver='svd', store_covariance=False, tol=0.0001)
>>> print(clf.predict([[-0.8, -1]]))
[1]
```

Nearest neigbours

Nearest neigbours

- Suppose $Y = \{0, 1\}$
- Parameter: k
- Use closest observations in the training set to make predictions

$$\hat{Y}(x) = \frac{1}{k} \sum_{N_k(x)} y_i$$

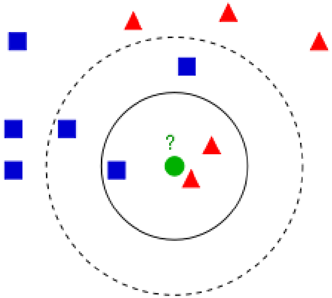
  Here $N_k(x)$ denotes the k-nearest neighbors of x (w.r.t. some metric, e.g. Euclidean distance)
- Decision:

$$label(x) = \begin{cases} 0 & \text{if } \hat{Y}(x) < 0.5 \\ 1 & \text{otherwise} \end{cases}$$

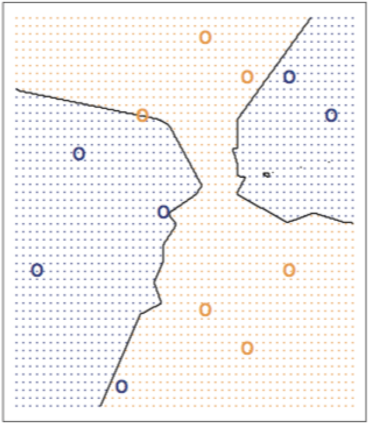
Nearest neighbors

Note: $Y = \{0, 1\}$,

$$\hat{Y}(x) = \frac{1}{k} \sum_{N_k(x)} y_i$$



Nearest neighbors



Decision boundary, $k = 3$

Nearest neighbors

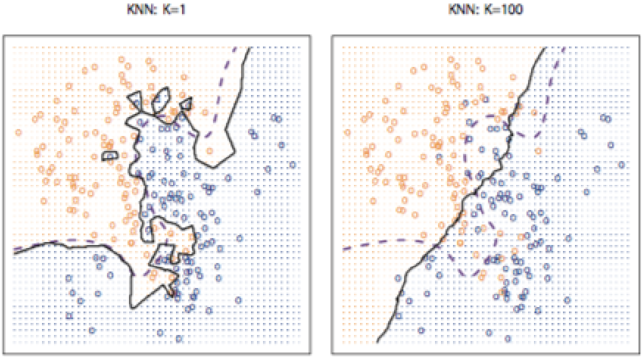


FIGURE 2.16. *A comparison of the KNN decision boundaries (solid black curves) obtained using $K = 1$ and $K = 100$ on the data from Figure 2.13. With $K = 1$, the decision boundary is overly flexible, while with $K = 100$ it is not sufficiently flexible. The Bayes decision boundary is shown as a purple dashed line.*