LEC 4: Discriminant Analysis for Classification

Dr. Guangliang Chen

February 25, 2016

Outline

- Last time: FDA (dimensionality reduction)
- Today: QDA/LDA (classification)
- Naive Bayes classifiers
- Matlab/Python commands

Probabilistic models

We introduce a mixture model to the training data:

- We model the distribution of each training class C_i by a pdf $f_i(\mathbf{x})$.
- We assume that for a fraction π_i of the time, $\mathbf{x}$ is sampled from C_i.

The *Law of Total Probability* implies that the mixture distribution has a pdf

$$f(\mathbf{x}) = \sum f(\mathbf{x} \mid \mathbf{x} \in C_i) P(\mathbf{x} \in C_i) = \sum f_i(\mathbf{x})\pi_i$$

that generates both training and test data (two independent samples from $f(\mathbf{x})$).

We call $\pi_i = P(\mathbf{x} \in C_i)$ the *prior probabilities*, i.e., probabilities that $\mathbf{x} \in C_i$ prior to we see the sample.

How to classify a new sample

A naive way would be to assign a sample to the class with largest prior probability

$$i^* = \mathrm{argmax}_i \ \pi_i$$

We don't know the true values of π_i, so we'll estimate them using the observed training classes (in fact, only their sizes):

$$\hat{\pi}_i = \frac{n_i}{n}, \quad \forall\, i$$

This method makes constant prediction, with error rate $1 - \frac{n_{i^*}}{n}$.

Is there a better way?

Maximum A Posterior (MAP) classification

A (much) better way is to assign the label based on the **posterior probabilities** (i.e., probabilities after we see the sample):

$$i^* = \text{argmax}_i \; P(\mathbf{x} \in C_i \mid \mathbf{x})$$

Bayes' Rule tells us that the posterior probabilities are given by

$$P(\mathbf{x} \in C_i \mid \mathbf{x}) = \frac{f(\mathbf{x} \mid \mathbf{x} \in C_i) P(\mathbf{x} \in C_i)}{f(\mathbf{x})} \propto f_i(\mathbf{x})\pi_i$$

Therefore, the MAP classification rule can be stated as

$$\boxed{i^* = \text{argmax}_i \; f_i(\mathbf{x})\pi_i}$$

This is also called Bayes classifier.

Estimating class-conditional probabilities $f_i(\mathbf{x})$

To estimate $f_i(\mathbf{x})$, we need to pick a model i.e., a distribution from certain family to represent each class.

Different choices of the distribution lead to different classifiers:

- **LDA/QDA**: by using multivariate Gaussian distributions

$$f_i(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1}(\mathbf{x}-\mu_i)}, \quad \forall \text{ Class } i$$

- **Naive Bayes**: by assuming independent features in $\mathbf{x} = (x_1, \ldots, x_d)$

$$f_i(\mathbf{x}) = \prod_{j=1}^{d} f_{ij}(x_j)$$

MAP classification with multivariate Gaussians

In this case, we estimate the distribution means μ_i and covariances Σ_i using their sample counterparts (based on training data):

$$\hat{\mu}_i = \frac{1}{n_i}\sum_{\mathbf{x}\in C_i}\mathbf{x}, \quad \text{and} \quad \hat{\Sigma}_i = \frac{1}{n_i-1}\sum_{\mathbf{x}\in C_i}(\mathbf{x}-\hat{\mu}_i)(\mathbf{x}-\hat{\mu}_i)^T$$

This leads to the following classifier:

$$\begin{aligned} i^* &= \text{argmax}_i \; \frac{n_i}{n(2\pi)^{d/2}|\hat{\Sigma}_i|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\hat{\mu}_i)^T\hat{\Sigma}_i^{-1}(\mathbf{x}-\hat{\mu}_i)} \\ &= \boxed{\text{argmax}_i \; \log n_i - \frac{1}{2}\log|\hat{\Sigma}_i| - \frac{1}{2}(\mathbf{x}-\hat{\mu}_i)^T\hat{\Sigma}_i^{-1}(\mathbf{x}-\hat{\mu}_i)} \end{aligned}$$

Decision boundary

The decision boundary of a classifier consists of points that have a tie.

For the MAP classification rule based on mixture of Gaussians modeling, the decision boundaries are given by

$$\begin{aligned} &\log n_i - \frac{1}{2}\log|\hat{\Sigma}_i| - \frac{1}{2}(\mathbf{x}-\hat{\mu}_i)^T\hat{\Sigma}_i^{-1}(\mathbf{x}-\hat{\mu}_i) \\ =&\log n_j - \frac{1}{2}\log|\hat{\Sigma}_j| - \frac{1}{2}(\mathbf{x}-\hat{\mu}_j)^T\hat{\Sigma}_j^{-1}(\mathbf{x}-\hat{\mu}_j) \end{aligned}$$

This shows that the MAP classifier has quadratic boundaries.

We call the above classifier *Quadratic Discriminant Analysis (QDA)*.

Equal covariance: A special case

QDA assumes that each class distribution is multivariate Gaussian (but with its own center μ_i and covariance Σ_i).

We examine the special case when $\Sigma_1 = \cdots = \Sigma_c = \Sigma$ so that the different classes are shifted versions of each other.

In this case, the MAP classification rule becomes

$$i^* = \mathrm{argmax}_i \ \log n_i - \frac{1}{2}(\mathbf{x} - \hat{\mu}_i)^T \hat{\Sigma}^{-1} (\mathbf{x} - \hat{\mu}_i)$$

where $\hat{\Sigma}$ represents the pooled estimate of Σ using all classes

$$\hat{\Sigma} = \frac{1}{n-c} \sum_{i=1}^{c} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \hat{\mu}_i)(\mathbf{x} - \hat{\mu}_i)^T$$

Decision boundary in the special case

The decision boundary of the equal-covariance classifier is:

$$\log n_i - \frac{1}{2}(\mathbf{x} - \hat{\mu}_i)^T \hat{\Sigma}^{-1}(\mathbf{x} - \hat{\mu}_i) = \log n_j - \frac{1}{2}(\mathbf{x} - \hat{\mu}_j)^T \hat{\Sigma}^{-1}(\mathbf{x} - \hat{\mu}_j)$$

which simplifies to

$$\mathbf{x}^T \hat{\Sigma}^{-1}(\hat{\mu}_i - \hat{\mu}_j) = \log \frac{n_i}{n_j} - \frac{1}{2}\left(\hat{\mu}_i^T \hat{\Sigma}^{-1} \hat{\mu}_i - \hat{\mu}_j^T \hat{\Sigma}^{-1} \hat{\mu}_j\right)$$

This is a hyperplane with normal vector $\hat{\Sigma}^{-1}(\hat{\mu}_i - \hat{\mu}_j)$, showing that the classifier has linear boundaries.

We call it *Linear Discriminant Analysis (LDA)*.

Relationship between LDA and FDA

The LDA boundaries are hyperplanes with normal vectors $\hat{\Sigma}^{-1}(\hat{\mu}_i - \hat{\mu}_j)$.

In 2-class FDA, the projection direction is

$$\mathbf{v} = \mathbf{S}_w^{-1}(\hat{\mu}_1 - \hat{\mu}_2)$$

where

$$\mathbf{S}_w = \mathbf{S}_1 + \mathbf{S}_2 = \sum_{i=1}^{2} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \hat{\mu}_i)(\mathbf{x} - \hat{\mu}_i)^T = (n-2)\hat{\Sigma}.$$

Therefore, LDA is essentially a union of 2-class FDAs (with cutoffs selected based on Bayes rule). However, they are derived from totally different perspectives (optimization versus probabilistic).

Naive Bayes

The naive Bayes classifier is also based on the MAP decision rule:

$$i^* = \mathrm{argmax}_i \; f_i(\mathbf{x})\pi_i$$

A simplifying assumption is made on the individual features of $\mathbf{x}$:

$$f_i(\mathbf{x}) = \prod_{j=1}^{d} f_{ij}(x_j) \qquad (x_1, \ldots, x_d \text{ are independent})$$

Accordingly, the decision rule becomes

$$i^* = \mathrm{argmax}_i \; \pi_i \prod_{j=1}^{d} f_{ij}(x_j) = \mathrm{argmax}_i \; \log \pi_i + \sum_{j=1}^{d} \log f_{ij}(x_j)$$

How to estimate f_{ij}

The independence assumption reduces the high dimensional density estimation problem ($f_i(\mathbf{x})$) to a union of simple 1D problems ($\{f_{ij}(x)\}_j$).

Again, we need to pick a model for the f_{ij}.

For continuous features (which is the case in this course) the standard choice is the 1D normal distribution

$$f_{ij}(x) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} e^{-(x-\mu_{ij})^2/2\sigma_{ij}^2}$$

where μ_{ij}, σ_{ij} can be estimated similarly using the training data.

MAP classification: A summary

- General decision rule

$$i^* = \text{argmax}_i \ f_i(\mathbf{x})\pi_i$$

- Examples of Bayes classifiers
  - **QDA**: multivariate Gaussians
  - **LDA**: multivariate Gaussians with equal covariance
  - **Naive Bayes**: independent features $x_1, \ldots, x_d$

We will show some experiments with MATLAB (maybe also Python) next class.

The Fisher Iris dataset

- Background (see Wikipedia)
  - A typical test case for many statistical classification techniques in machine learning
  - Originally used by Fisher for developing his linear discriminant model
- Data information
  - **150 observations**, with 50 samples from each of three species of Iris (*setosa*, *virginica* and *versicolor*)
  - **4 features** measured from each sample: the length and the width of the sepals and petals, in centimeters

MATLAB implementation of LDA/QDA

% fit a discriminant analysis classifier

mdl = fitcdiscr(trainData, trainLabels, 'DiscrimType', type)

% where **type** is one of the following:

- **'Linear'** (default): LDA
- **'Quadratic'**: QDA

% classify new data

pred = predict(mdl, testData)

Python scripts for LDA/QDA

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

#from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

lda = LinearDiscriminantAnalysis()

pred = lda.fit(trainData,trainLabels).predict(testData)

print("Number of mislabeled points: %d" %(testLabels != pred).sum())
```

The singularity issue in LDA/QDA

Both LDA and QDA require inverting covariance matrices, which may be singular in the case of high dimensional data.

Common techniques to fix this:

- Apply PCA to reduce dimensionality first, or
- Regularize the covariance matrices, or
- Use psuedoinverse: 'pseudoLinear', 'pseudoQuadratic'

MATLAB functions for Naive Bayes

% fit a naive Bayes classifier

mdl = fitcnb(trainData, trainLabels, 'Distribution', ‘normal')

% classify new data

pred = predict(mdl, testData)

Python scripts for Naive Bayes

```
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()

pred = gnb.fit(trainData, trainLabels).predict(testData)

print("Number of mislabeled points: %d" %(testLabels != pred).sum())
```

Improving Naive Bayes

- **Independence assumption**: apply PCA to get uncorrelated features (closer to being independent)

- **Choice of distribution**: change normal to kernel smoothing to be more flexible

  mdl = fitcnb(trainData, trainLabels, 'Distribution', ‘kernel')

  However, this will be at the expense of speed.

HW3a (due in 2 weeks)

First use PCA to project the MNIST dataset into s dimensions and then do the following.

1. For each values of s below perform LDA on the data set and compare the errors you get:
    - $s = 154$ (95% variance)
    - $s = 50$
    - $s =$ your own choice (preferably better than the above two)
2. Repeat Question 1 with QDA instead of LDA (everything else being the same).

3. For each values of s below apply the Naive Bayes classifier (by fitting pixelwise normal distributions) to the data set and compare the errors you get:

   - $s = 784$ (no projection)
   - $s = 154$ (95% variance)
   - $s = 50$
   - $s =$ your own choice (preferably better than the above three)

Next time: Two-dimensional LDA

Midterm project 2: MAP classification

Task: Concisely describe the classifiers we have learned in this part and summarize their corresponding results in a poster to be displayed in the classroom.

In the meantime, you are encouraged to try the following ideas.

- The kernel smoothing option in Naive Bayes
- The cost option in LDA/QDA:
  mdl=fitcdiscr(trainData,trainLabels,'Cost',COST) where COST is a square matrix, with COST(I,J) being the cost of classifying a point into class J if its true class is I. Default: COST(I,J)=1 if I =J, and COST(I,J)=0 if I=J.
- What else?

Who can participate: One to two students from this class, subject to instructor's approval.

When to finish: In 2 to 3 weeks.

How it will be graded: Based on clarity, completeness, correctness, originality.