2.1 Supervised and Unsupervised Prediction

肖　麗穎

Abstract

This chapter introduces some common classification and clustering algorithms and discusses how they can be extended to structure prediction problems in NLP.

The main focus will be classification.

Functions f: X-->Y

(X:multi-dimensional input space, Y:a set of discrete classes)

Such as spam filtering..

Particular words have particular probabilities of occurring in spam email or legitimate email.

A simple classification algorithm -->count the number of words in email-->classifying the email as spam or not.

y	zebra	viagra	venus
spam	0	1	0
non-spam	1	0	0
non-spam	1	0	1
?	0	1	1

As for bigger datasets..

- Gather the different classes of documents in separate folders, such as a folder of spam emails and a folder of non-spam emails.
- Represent documents as vectors
- Find good representations in machine learning

  >by using words or unigrams?

2.1 Standard assumptions in supervised learning

- Four of the standard assumptions in supervised learning.

  1. Smoothness Assumption

     A minimal assumption in order to be able to infer from a finite sample to a possibly infinite set of unseen data points.

     if(xi and xj are close) ==>(yi and yj are also close)

     Email that shares many features with known spam emails is likely to be spam.

2. Independently and Identically Distributed(i.i.d)

- Each data point is sampled from the same probability distribution and that all data points are mutually independent.

  ==>If the roulette ball in a casino lands on red 100 times in a row, the next spin is no less likely to be black.

  ==>In spam filtering the next email always has the same chance of being a spam email.

But in fact..

- The i.i.d. assumption simplifies the math in machine learning, but is invalid in most pratical learning scenarios.

  ==>In spam filtering, emails are typically neigther independently nor identically distributed. The way email corpora are collected, the occurrence of an email on hockey in an email corpus typically does increase the probability of seeing another email on hockey later.

3. Single Cluster Assumption

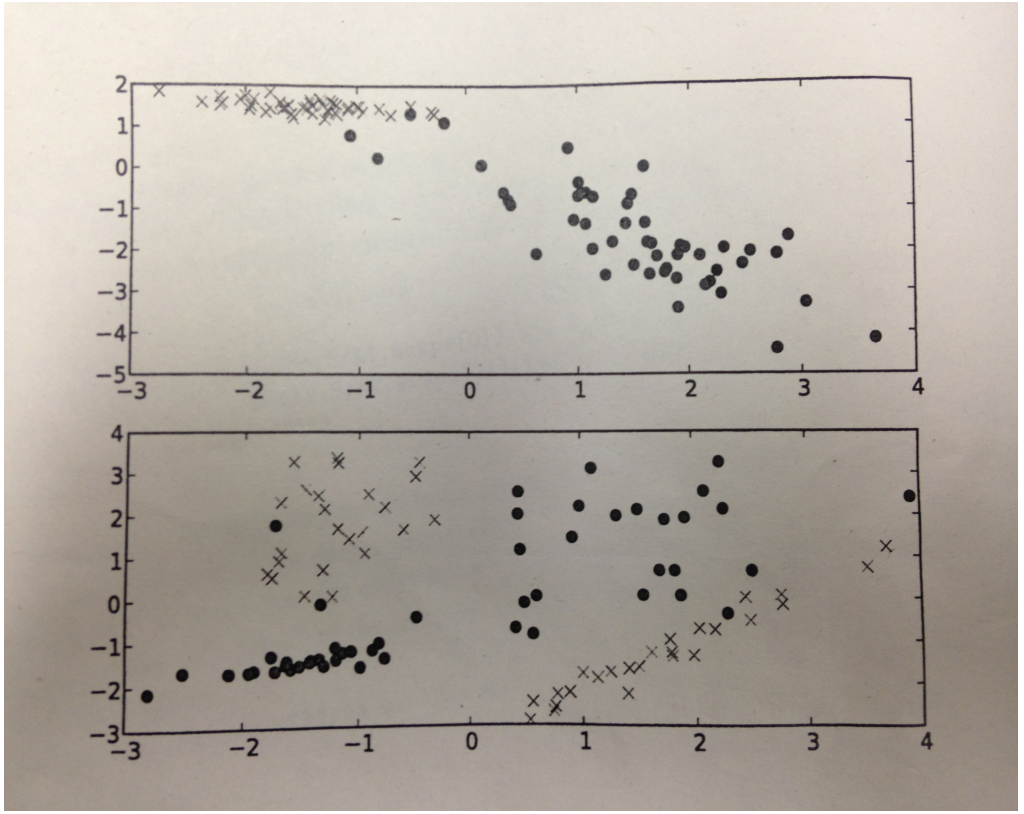

Figure 2.2: binary classification problems where each class consists of one or two clusters.

- Many supervised learning algorithms, but not all, also assume that classes form single,coherent clusters.

  ==>In spam filtering,this means that both spam and non-spam emails are assumed to form coherent clusters in our feature space.

- Figure 2.2 illstrates the single cluster assumption using artificial two-dimensional data.

4. Low-density separation

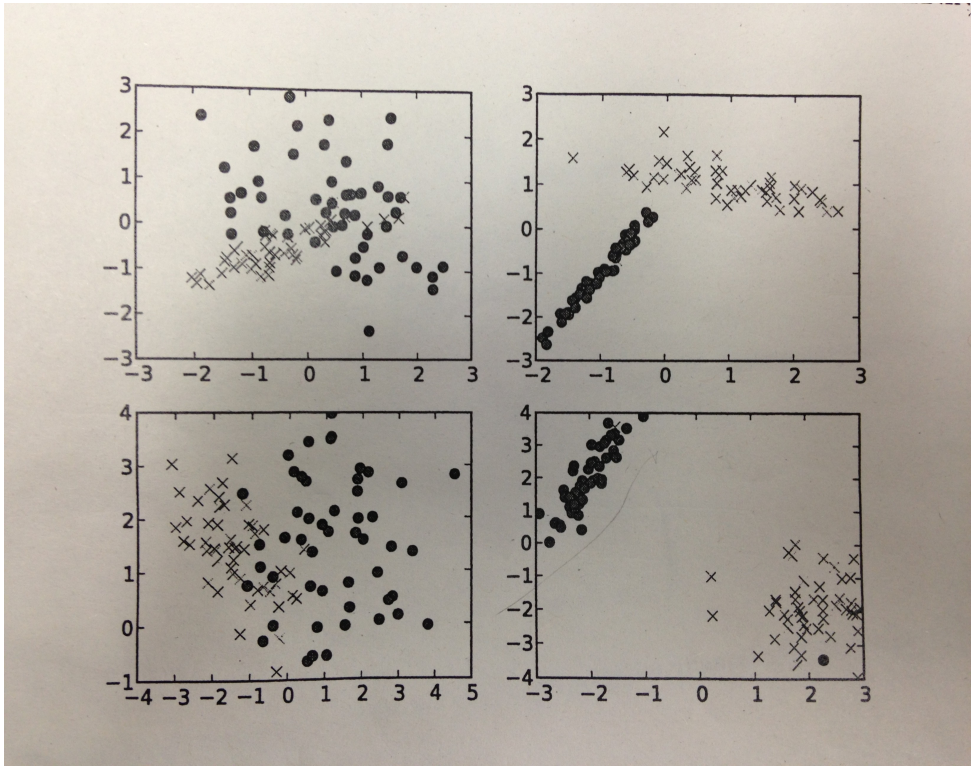

Figure 2.3: Binary classification problems with varying degrees of separability

- This assumption states that two data points in the same cluster are likely to belong to the same class.

2.1.1 How to check whether the assumptions hold

- This subsection introduces standard metrics that quantify to what extent the i.i.d assumption,the coherence assumption, and the separability assumption hold.

  1. To quantify to what extent data is identically distributed, we use two standard divergence measures: Kullback-Leibler(KL) divergence and Jensen-Shannon divergence.

  2. For the single cluster assumption and low-density separation, use with-in class and between-class scatter.

Divergence measures

- KL divergence is often used to test the representativety of samples or to measure the distance between two distributions P(x) and Q(x).

$$KL(P,Q) = \sum_x P(x) log \frac{P(x)}{Q(x)}$$

- Jensen-Shannon divergence is the symmetric version of KL divergence: (with M the average of P and Q.)

$$D_{JS}(P,Q) = \frac{1}{2} D_{KL}(P,M) + \frac{1}{2} D_{KL}(Q,M)$$

In practice, we typically have a finite sample of labeled data and some test instances and we cannot compute P(x) and Q(x) in general.

```
Def KLDisc(X1,X2):
P=np.array([(X1[:,i].mean()+.5/X1.shape[0] for i in range(X1.shape[1])])
Q=np.array([(X2[:,i].mean()+.5/X2.shape[0] for i in range(X2.shape[1])])
div=sum(multiply(P,log(P/Q)))
return div
```

- The shape attribute of numpy arrays is the number of rows and, if any, columns. In this case, we compute a smoothed mean for each column, that is each of our feature.

Within-class and between-class scatter

- Rely on the notion of co-variance.
- The average observed value of a variable(its mean) is often referred to as its expected value.
- If E(x1)=E(x2)=20 ≠>P(x1=40)=P(x2=40)

The variance of a variable is the expected value of the difference between the expected value of a variable and its actual value squared,μ:

$$\sum_{i=1}^{N}(x_j^i - u_j)^2$$

Co-variance is a measure of how dependent two variables are:

$$Cov(xj, xk) = \sum_{N}^{i=1} (x_j^i - u_j)^2 (x_k^i - u_k)^2$$

- The Python in-built function cov takes the transpose of a dataset and computes a co-variance matrix:

$$\begin{pmatrix} Cov(x1, x1) & ... & Cov(x1, xn) \\ ... & ... & ... \\ Cov(xn, x1) & ... & Cov(xn, xn) \end{pmatrix}$$

If the within-class scatter is small, the cluster assumption is likely to hold. Similarly, if the between-class scatter is lager, the low-density separation assumption is likely to hold.

- The **within-class scatter** is computed by weighting the covariances of the class variables by class probabilities:

$$\sum_{y \epsilon Y} P(y) \sum_{(y,xi) \epsilon T} (xi - uy)(xi - uy)^T$$

- The **between-class scatter** is simply:

$$\sum_{y \epsilon Y} (uy - \overline{x})(uy - \overline{x})^T$$