11. SUPPORT VECTOR MACHINES AND IMBALANCED DATASETS

LEARNING OBJECTIVES

- SVM
- Imbalanced datasets

REVIEW OF LESSON 10

LAST LESSON REVIEW

- K-
  Means
- PCA

TODAY

SUPPORT VECTOR MACHINES

HYPERPLANE

Classification separating an hyperplane

- 1D => point
- 2D => line
- 3D =>
  surface
- etc ...

A hyperplane in 2 dimension is defined by a line equation $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$

In **p dimensions** a hyperplane is defined by:

$$\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p = 0 \qquad p > 0$$

LINEARLY SEPARABLE

We have n samples in p dimensions/features $X = \{x_i\} \quad i \in [1, n]$.

These observations belong to two classes $y_i \in \{-1, +1\}$

The classes are **linearly separable**: there is an hyperplane that fully separates the points according to their classes.

All points x_i such that

- $\beta_0 + \sum_{j=1}^{n} \beta_j x_{i,j} < 0$ belong to class -1
- $\beta_0 + \sum_{j=1}^{n} \beta_j x_{i,j} > 0$ belong to class +1

LINEAR DECISION BOUNDARY

We predict the class of a new point x'

by calculating

$$f(x') = \beta_0 + \sum_{j=1}^{n} \beta_j x'_j$$

- $f(x') > 0 => +1$
- $f(x') < 0 => -1$

A classifier that is based on a separating hyperplane leads to a **linear decision boundary**.

ONE LINE

Note that since

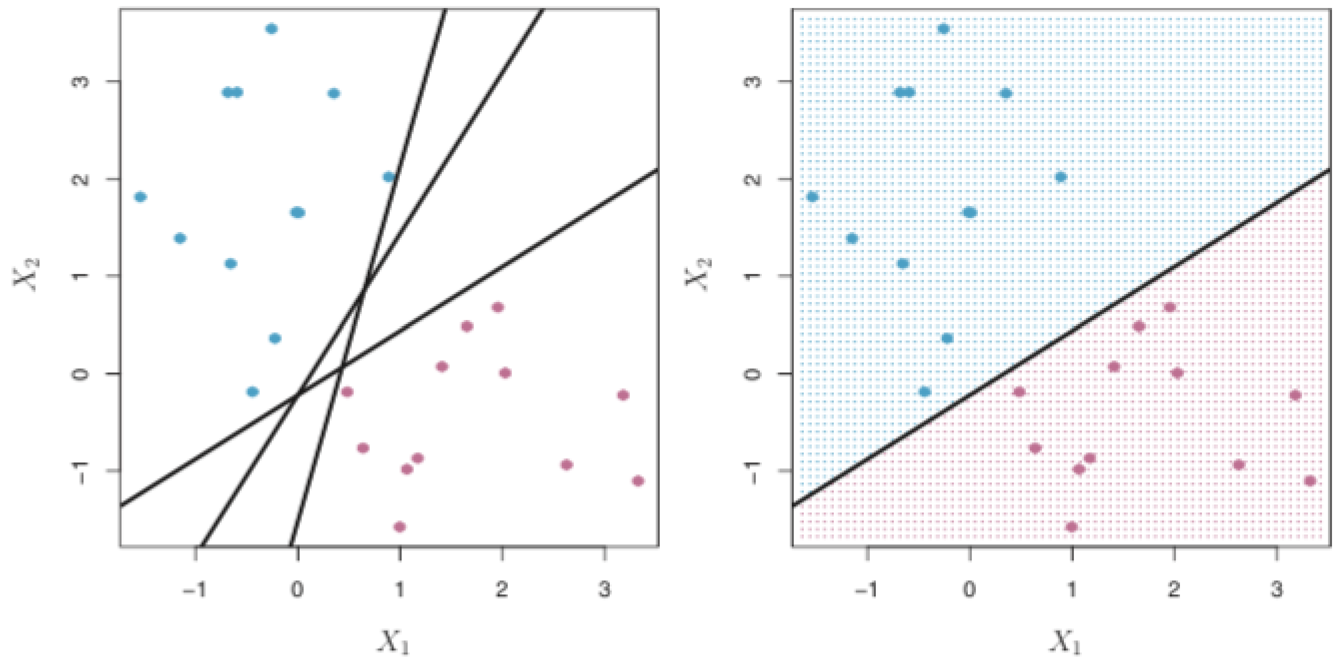
- y_i
- $\beta_0 + \sum_{j=1}^{n} \beta_j x_{i,j}$

have the same sign

$$y_i(\beta_0 + \sum_{j=1}^{n} \beta_j x_{i,j}) > 0$$

INFINITY OF HYPERPLANES

If our data is linearly separable then there exists an infinity of hyperplanes that can separate it



=> We need to introduce a margin to separate the classes

Fig from Introduction to Statistical Learning

MAXIMAL MARGIN CLASSIFIER

We want to find the Hyperplane that will maximise the distance to all the points. Best separation of the classes

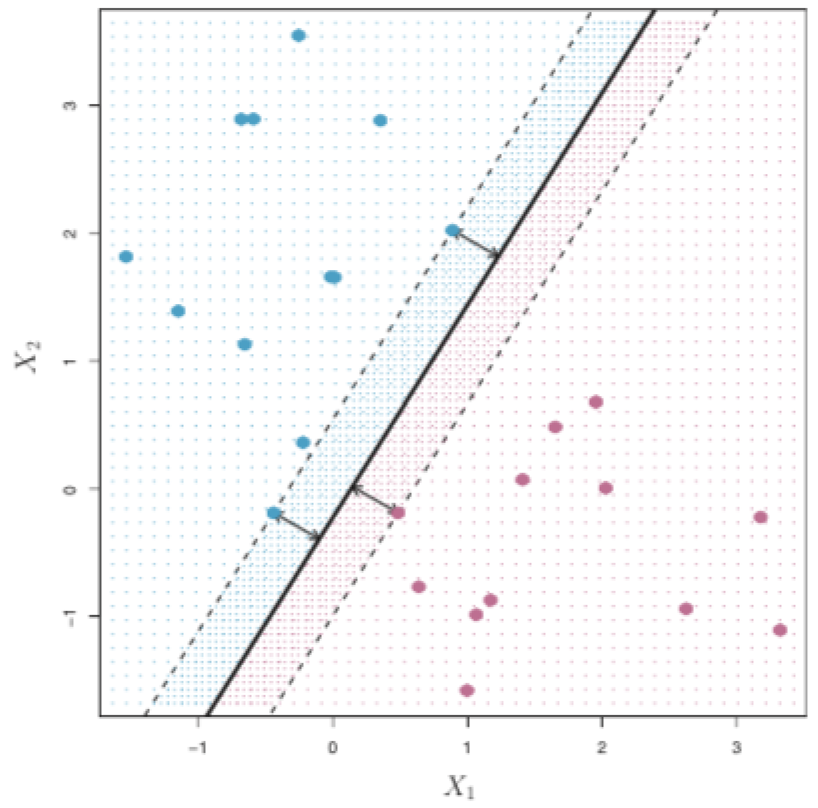


Fig from Introduction to Statistical Learning

MAXIMAL MARGIN CLASSIFIER

Finding the largest margin that separates the classes is equivalent to solving

$$\max_{\beta_i} M \quad \text{such that} \quad y_i(\beta_0 + \sum_{j=1}^{n} \beta_j x_{i,j}) > M \quad \forall i \in [1, \ldots, n]$$

with $\sum_{j=0}^{p} \beta_j^2 = 1$

SUPPORT VECTORS

The observations that are on the margin lines are called **support vectors**

They *support* the maximal margin hyperplane in the sense that if these points were moved slightly then the maximal margin hyper- plane would move as well.

Interestingly, the maximal margin hyperplane depends directly on the support vectors, but not on the other observations: a movement to any of the other observations would not affect the separating hyperplane, provided that the observation’s movement does not cause it to cross the boundary set by the margin.

A change in the support vector impacts the margin a lot => over fitting

SUPPORT VECTOR CLASSIFIER

What if we allow some points to be either wrongly classified or at least within the margin boundaries?

This is the case when the classes are not separable, we still want the best margin possible. But some points will be on the wrong side

Add a tuning parameter C that dictates the severity of the violation of the margin

The bigger C is the more points are

- within the margin
- or misclassified

C is some budget for violation of the margin which is determined during cross validation

Even in the case of linearly separable classes adding some flexibility to the margin will make the classfier more robust, more flexible (less overfitting)

SUPPORT VECTOR CLASSIFIER

C is a non negative tuning parameter

$$\max_{\beta_i, \epsilon_i} M \quad \text{such that} \quad y_i(\beta_0 + \sum_{j=1}^{n} \beta_j x_{i,j}) > M - \epsilon_i \quad \forall i \in [1, \ldots, n]$$

with

- $\sum_j \beta_j^2 = 1$
- $\epsilon_i > 0$
- $\sum_i \epsilon_i \leq C$

LAB SVC

L11 N1 SVC.py

SUPPORT VECTOR MACHINE

What if the data is not even close to be linearly separable?

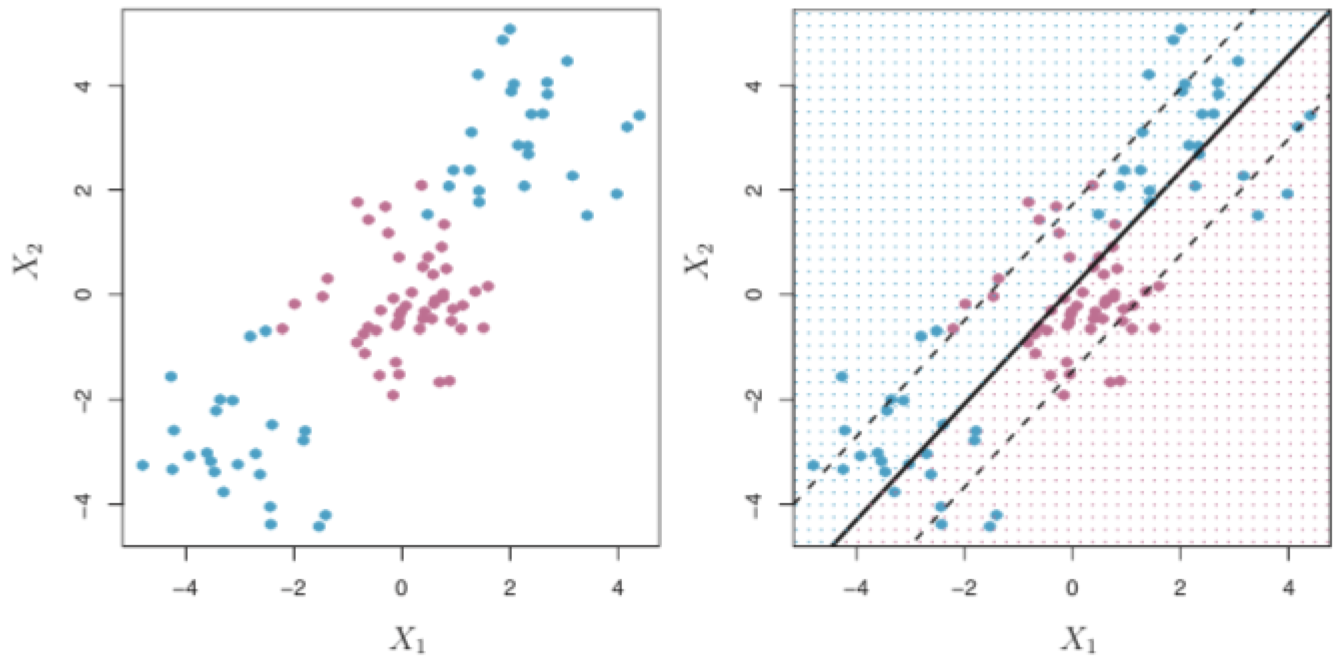


Fig from Introduction to Statistical Learning

INTRODUCING KERNELS

The optimization equation for the linear support vector classifier can be rewritten as such

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

with $K(x, x_i) = \langle x, x_i \rangle$ the vector dot product.

But we could use a different Kernel function

Note:This involves $p * \frac{n(n-1)}{2}$ multiplications! However only the support vectors are useful. So we can limit the sum to S support vectors.

KERNELS

Here are some classic Kernel functions

- Linear Kernel $K(x, x_i) = \langle x, x_i \rangle$
- Polynomial Kernel (d): $K(x, x_i) = (1 + \sum_{j=1}^{p} x_j x_{i,j})^d$
- Radial Kernel $K(x, x_i) = \exp(-\gamma \sum_{j=1}^{p} (x_j - x_{i,j})^2)$

SVM LAB

gads/11_svm_imbalanced/py/L11_N2_SVM.py

PART II IMBALANCED DATASETS

IMBALANCED DATASETS

Caravan dataset: a few hundreds YES vs thousands of NO

- Setting all predictions to No gives you a 94% prediction rate!

Strategies

- Collect more data
- Accuracy is not always the best metric (Cohen's Kappa, F1 score, ...)
- Oversample or Undersample
- SMOTE
- Penalized Models
- Decision trees often perform well on imbalanced datasets

OVER / UNDER SAMPLE

Yes is the minority class and No the majority class

The idea is to balance the ratio of Yes vs No samples $r = \frac{n_{Yes}}{n_{No}} \ll 1$

UNDER SAMPLE

Build a training dataset composed of n No samples and n Yes samples.

$$n_{Yes} = n_{No} \implies r = \frac{n_{Yes}}{n_{No}} = 1$$

OVER SAMPLE

Copy the Yes samples k times and keep $k * n_{Yes}$ samples of the No class.

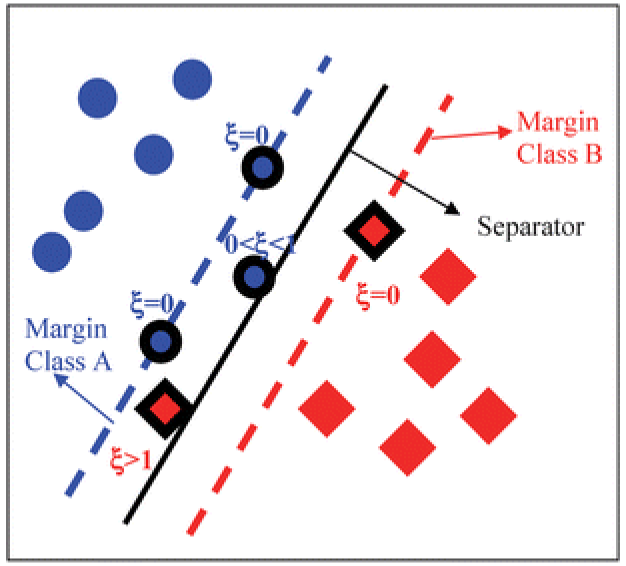
$$k * n_{Yes} = n_{No} \implies r = \frac{n_{Yes}}{n_{No}} = 1$$

SMOTE:

N. V. Chawla et al. (2002) "SMOTE: Synthetic Minority Over-sampling Technique", Volume 16, pages 321-357

SMOTE is an oversampling method that creates creates entries that are interpolations of the minority class, as well as undersamples the majority class.

SUPPORT VECTOR MACHINE



```
class sklearn.svm.SVC( C=1.0, kernel='rbf',  ...)¶
```

- C: Penalty on the error term. Level of samples within the margin
- Kernel: The internal representation that models the boundaries: linear, polynomial, Gaussian (rbf), ...

LAB ON CARAVAN

In the next exercise we will use the Caravan dataset. Highly unbalanced.

Compare the AUC score, the ROC Curve and the confusion matrix for each of the following strategies

1. Dumb classifier: Accuracy Paradox
2. Support Vector Machine
3. Under Sample
4. Over Sample
5. SMOTE with imbalanced-learn

L11 Imbalanced Notebook

1. ACCURACY PARADOX

Load the Caravan dataset

Consider a simple classifier that always gives 'No' as the result.

Accuracy?

2. SVM

- Split the dataset into train and test.

Grid search an SVM on the train set.

Accuracy? Confusion Matric, ROC Curve, AUC?

Conclusion?

Keep the initial X_test and y_test

3. UNDER SAMPLE

In the initial dataset we only have 348 samples in the Yes class.

Build a dataframe with these 348 samples and a random selection of 348 No samples

Grid Search an SVM (same params as last step)

Score and confusion matrix on the original test set?

4. OVER SAMPLE

Build a new dataframe by replicating the Yes set 4 times

Grid Search an SVM (same params)

Score and confusion matrix on the original test set?

5. SYNTHETIC MINORITY OVERSAMPLING TECHNIQUE

Install

pip install -U imbalanced-learn

from imblearn.over_sampling import SMOTE smo_X, smo_y = smote.fit_sample(X_train, y_train)

results?

LINKS

- Dealing with imbalanced data - R
- 8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset

NLP IN HEALTHCARE