Support Vector Machines (and not only)

Alipio Jorge (DCC-FCUP)

November 2020

Classification methods

- **Classification**: learning
  - **Given** examples with classe labels
  - **Obtain** a model that assigns each new example to one class
- **Methods** we have seen for classification
  - kNN
  - NaiveBayes
  - Decision Tree
  - Neural Networks

Linear methods

- Linear regession is **not** a classification method
  - it finds a **linear combination** of the input variables
  - in order to **predict a continuous** target variable
- Are there similar methods for classification?
  - Logistic regression
  - Linear Discriminant Analysis

A two class easily separable dataset

- We will now apply these methods on a slightly modified version of the iris dataset (call it sepiris)
    - two classes only
    - artificially improved separation of the classes

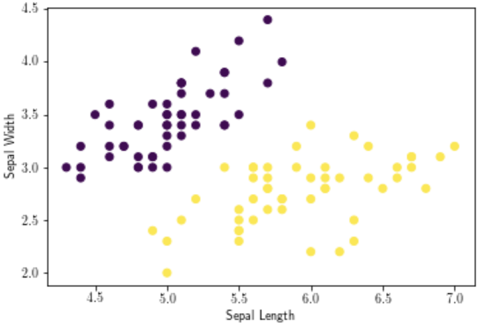


Figure 1: The sepiris data

Logistic Regression

- Logistic regression is a **linear model** for classification
  - finds a linear boundary between two classes
  - Let's see the result for **sepiris**

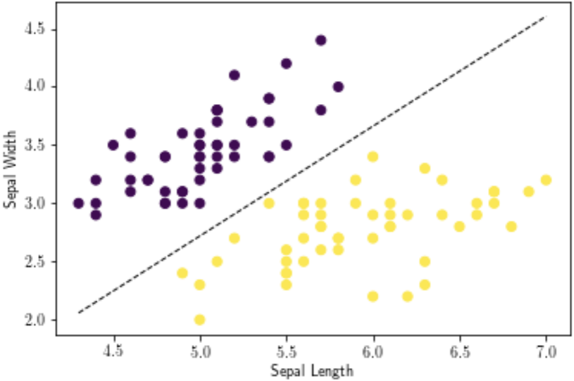


Figure 2: Logistic regression on the sepiris data

Logistic regression: brief explanation

- We want to classify example x in one of two classes
- Assume the probability of being one of the classes is modeled by

$$sigmoid(\beta_0 + \beta_1 x_1 + \ldots + \beta_m x_m)$$

- The **classification** boundary is determined by the *sigmoid*

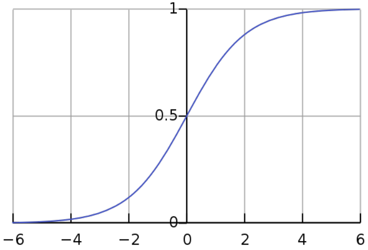


Figure 3: logistic function

Logistic regression: brief explanation

- The coefficients are estimated using **Maximum Likelihood Estimation**
- The approach is **iterative** (no closed form)
- In the sepiris example we find intercept and slope

Linear Discriminant Analysis: LDA

- Also finds a **linear boundary**
- Strong assumptions about the data
  - Conditional probabilities per class are **normally distributed**
  - Class covariances are identical
- Similar results as logistic regression
- A lot to be said about these two approaches
- Is generalised to **Quadratic Discriminant Analysis**: QDA
- Good baselines, but easily beaten

Support vector machines

- Support Vector Machines **start** with linear separation
  - but can be generalised to **non-linear** separation
  - and to **regression** as well

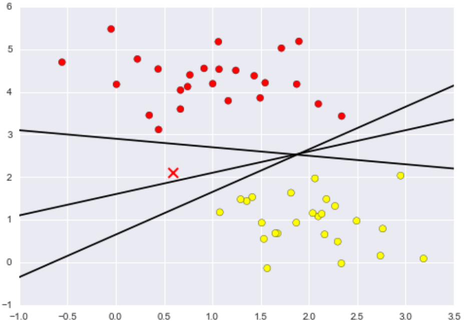
Support Vector Machines

- We are back to a **linearly separable** example
  - This is an **easy case**, but it will get complicated
  - What is the **best linear boundary**?
    - Linear regression and DLA give answers to that

Figure 4: Artificial data [VanDerPlas]

Support Vector Machines

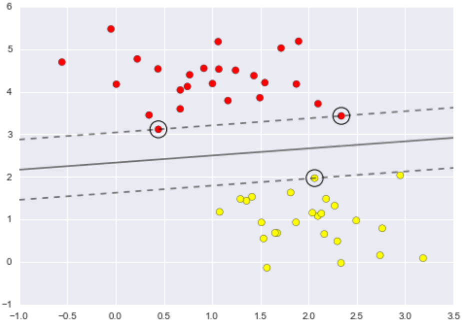
- There are **many possible** boundaries
  - Different boundaries may give different results for **boundary cases**



  - What is **the best** one?

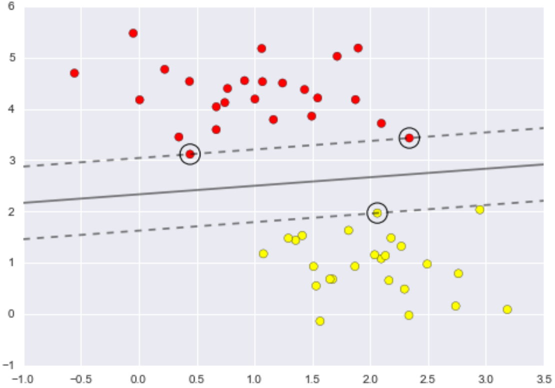
Support Vector Machines

- Consider the **distance of the boundary to the closest point**
  - that is called the **margin**
- Find the boundary that **maximizes** the margin
  - That is the **SVM approach**
  - The points on the limit of the margin are the **support vectors**
  - The solution can be found using **optimization**

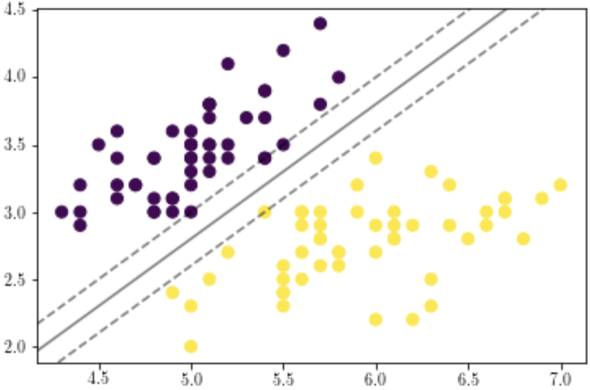


Support Vector Machines

- The support vectors **define** the boundary
- All the other points are **irrelevant** for the SVM approach
- That is not the case with **any** of the methods so far
    - So the answer is different



- Let's try SVM on the sepiris data
  - we see th boundary is now **centered**
  - there are 3 support vectors
    - (5.0, 3.0), (5.5, 3.5), (5.4, 3.0)



SVM: Mathematically

- A boundary hyperplane is defined by

$$f(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p = 0$$

- If $f(x) \geq 0$,
  - x is in the **positive class** $y = 1$
  - otherwise it is in the negative class $y = -1$
- **So** we want a boundary so that $y_i f(x_i) \geq 0$
  - the distance of a point x to the margin is given by $f(x)$ if $\|\beta\| = 1$

SVM: Mathematically

- So, we can define the learning problem as

$$\max_{\beta, \beta_0, \|\beta\|=1} M$$

$$\text{subject to} \quad y_i(x_i{}^T \beta + \beta_0) \geq M$$

- A solution β, β_0 to this problem gives us a **maximal margin classifier**

Support Vector Machines

- The shown examples are **easy**
- What if:
  - there is **no clear separation** of the classes?
  - the boundary is **not linear**

- **No clear separation** of the classes
  - We cannot find a boundary with a positive margin
  - we will have to allow **soft margins**

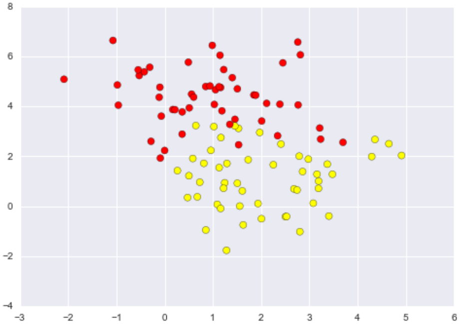


Figure 8: No clear separation [VanderPlas]

Support Vector Machines: soft margin

- The **soft margin**
  - we **allow some points** into the margin
  - these are also **support vectors**
  - we **minimize** the total intrusion of points $\sum_i \xi_i$
    - instead of maximizing the margin

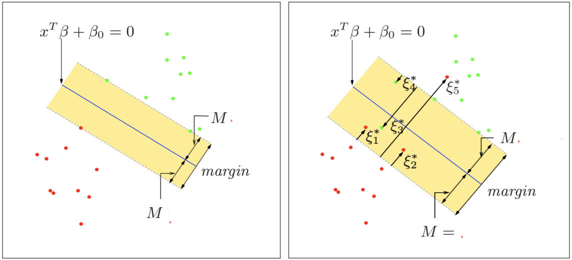


Figure 9: Soft margins [Elements of Statistical Learning]

Support Vector Machines: soft margin

The learning problem can be **softened** as

$$\max_{\beta,\beta_0,\|\beta\|=1} M$$

$$\text{subject to} \quad y_i(x_i{}^T\beta + \beta_0) \geq M(1-\xi_i)$$

$$\xi_i \geq 0 \;, \quad \sum_i \xi_i \leq C$$

This leads to the standard **Support Vector Classifier**

- the ξ_i are exceptions to the margin
- the constant C is like a budget for the misclassification cost

SVM: sklearn

- We can use the `SVC` function from `sklearn`
  - linear separation is given by parameter `kernel`
  - the cost is parameter `C`
    - `C > 0`

```
from sklearn.svm import SVC # "Support vector classifier"
model = SVC(kernel='linear', C=1.0)
model.fit(X, y)
```

SVM: non-linear boundaries

- What if the data is **not linearly separable**?

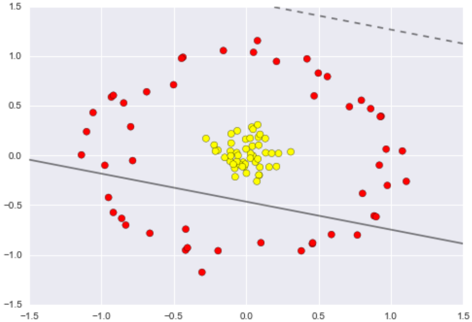


Figure 10: No linear separation [Van der Plas]

SVM: non-linear boundaries

- We can project the **original space**
  - into an **expanded space**
  - where the data is **linearly separable**

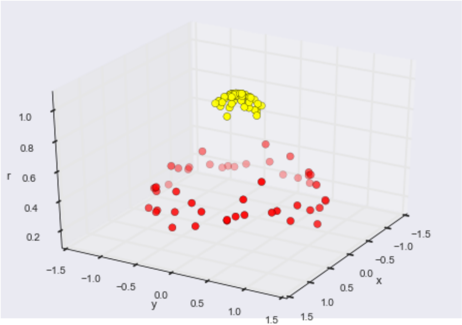


Figure 11: Expanded with linear separation [Van der Plas]

SVM: kernels

- **How** do we expand the original space in a **feasible way**?
- It can be shown that the β_i can be **defined** using the support vectors

$$\widehat{\beta} = \sum_{i=1}^{N} \alpha_i y_i x_i$$

  - α_i are non-negative and are **non zero** only for the support vectors
- So, $f(x)$, the classifier, is

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i x_i x + \beta_0$$

SVM: kernels

- Now suppose we expand the coordinates x into a set of coordinates $h(x)$
  - e.g.: original space is (a, b) and expanded space is $h(a, b) = (a, b, a^2, b^2, ab)$
  - a linear boundary in the expanded space is non-linear in the original
- So we can **generalize** $f(x)$ to the expanded coordinates

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

SVM: the kernel trick

- The inner product can be defined by a **kernel** function

$$K(x, x') = \langle h(x), h(x') \rangle$$

- The expansion is done **implicitly** by choosing a **kernel**
- Popular kernels
  - the **linear** kernel: $K(x, x') = \sum_{j=1}^{p} x_j x'_j + c$
  - the **radial basis**: $K(x, x') = \exp(-\gamma \|x - x'\|^2)$
  - the **polynomial**: $K(x, x') = (1 + \langle x, x' \rangle)^d = (1 + \sum_j x_j . x'_j)^d$

SVM: some thoughts

- **Advantages**
  - They depend only on support vectors means they are very compact
  - Prediction is very fast
  - Because they are affected only by points near the margin, they work well with high-dimensional data
  - Kernel methods are very versatile, able to adapt to many types of data.

SVM: some thoughts

- **Disadvantages**
  - Complexity is $O(N^3)$ at worst, or $O(N^2)$ for efficient implementations
    - For large data sets, this can be prohibitive
  - The results are strongly dependent on a suitable choice for the softening parameter C.
    - Must be carefully chosen via cross-validation
  - Results do not have a direct probabilistic interpretation
    - This can be estimated via an internal cross-validation (see the probability parameter of SVC), but this extra estimation is costly.

References

- Books
  - Han, Kamber & Pei, Data Mining Concepts and Techniques, Morgan
  - Van der Plas, Python Data Science Handbook
- Wikipedia
  - Logistic Regression
  - Linear Discriminant Analysis