Support Vector Machines

Pattern Recognition and Machine Learning - MuMeT 2017

Davide Abati

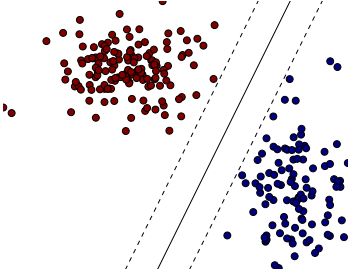
June 23th, 2017

University of Modena and Reggio Emilia

Support Vector Machines (SVM)

Very famous supervised learning algorithm:

- performs **binary** classification (can be adapted for multiclass problems)
- is **linear** (in its original formulation)
- main feature: chooses the decision boundary that maximizes the margin between positive and negative examples



How do you find such line?

SVM: problem setting

- we have a set of examples $\{\vec{x_i}\}_{i=1}^{N}$, with label $\{y_i\}_{i=1}^{N}$
  - $\vec{x_i}$ is a feature vector
  - $y_i \in \{-1, 1\} \quad \forall i = 1, \ldots, N$
- the line that maximizes the margin (hyperplane) is identified by
  - the vector $\vec{w}$ orthogonal to it
  - the intercept b

SVM: hard margin optimization

It can be shown that solving this constrained optimization problem maximizes the margin:

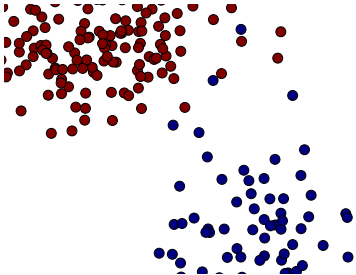
$$\begin{aligned} \min_{\vec{w},b} \quad & \frac{1}{2}||w||^2 \\ \text{s.t.} \quad & y_i(\vec{w}^T \cdot \vec{x_i} + b) > 1 \quad \forall i = 1, \ldots, N \end{aligned}$$

After estimating $\vec{w}$ and b, the decision rule for an unknown example $\vec{u}$ is simply:

$$f(\vec{u}) = \text{sign}(\vec{w}^T \cdot \vec{u} + b)$$

SVM: non separable data

What if you cannot find a linear decision boundary?



Two solutions:

- introduce slack variables ξ_i (patch solution)
- kernel trick

SVM: soft margin optimization

Introduce a slack variable ξ_i for every training example

$$\begin{aligned} \min_{\vec{w},b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i(\vec{w}^T \cdot \vec{x_i} + b) > 1 - \xi_i \quad \forall i = 1,\ldots,N \\ & \xi_i > 0 \qquad\qquad\qquad\qquad \forall i = 1,\ldots,N \end{aligned}$$

where C is a tradeoff between the margin maximization and the slack variable minimization.

The decision rule for an unknown example $\vec{u}$ does not change:

$$f(\vec{u}) = \text{sign}(\vec{w}^T \cdot \vec{u} + b)$$

Math guys find out that if you employ Lagrangian multipliers to solve the optimization the problem, you need to find the extrema of the following function ($\{\alpha\}_{i=1}^{N}$ are the lagrangian multipliers of each training example):

!!!!!!

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x_i} \cdot \vec{x_j} \tag{1}$$

SVM: kernels (2)

You can replace that dot product with a kernel ϕ

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi(\vec{x_i}, \vec{x_j}) \tag{2}$$

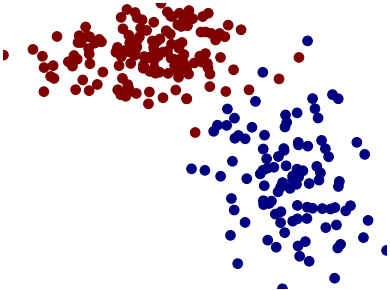
Choose a kernel:

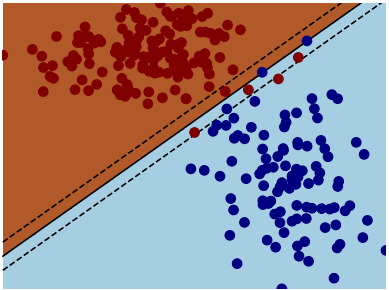
- linear: $\phi(\vec{x_i}, \vec{x_j}) = \vec{x_i} \cdot \vec{x_j}$
- polynomial: $\phi(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + c)^d$
- gaussian: $\phi(\vec{x_i}, \vec{x_j}) = \exp(-\frac{||\vec{x_i} - \vec{x_j}||^2}{2\sigma^2})$

This way, you draw the line in a transformed space, leading to non linear decision boundaries in the original space!

SVM example

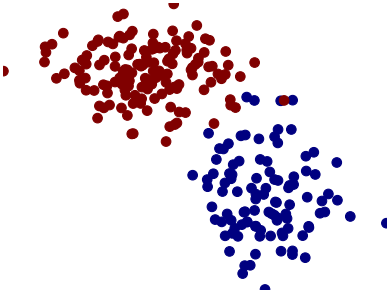
Using a linear kernel:

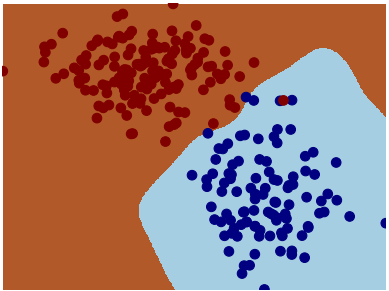




SVM example

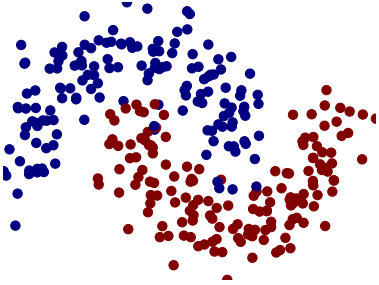
Using a rbf (gaussian) kernel:

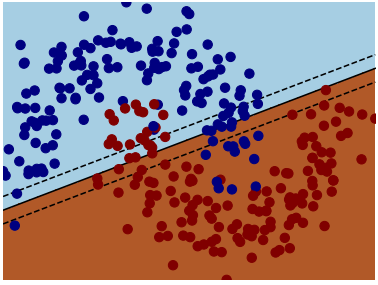




SVM example (2)

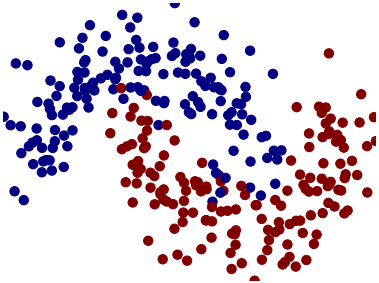
Using a linear kernel:

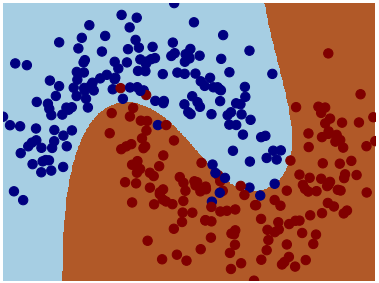




SVM example (2)

Using a rbf (gaussian) kernel:





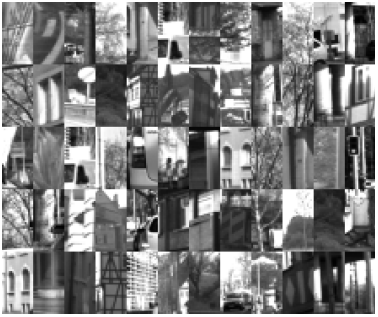
Application: people vs non people classification

SVM for people detection

We will discriminate between people and non people

people

non people



SVM for people detection (2)

Pipeline for people vs non people classification with **HOG features**:

- load training examples (images, HOG features, labels)
- load test examples (images, HOG features, labels)
- choose a kernel (linear or rbf) and a value for C and initialize a SVM model
  - use sklearn.svm.SVC
- fit a SVM model on training examples
  - use SVC.fit()
- use model for computing predictions on test examples
  - use SVC.predict()