Support Vector Machines

Machine Learning and Deep Learning 2018

Davide Abati, Angelo Porrello

October 25th, 2018

University of Modena and Reggio Emilia

Today's agenda

SVM basics

PEGASOS

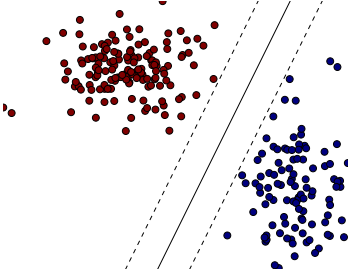
Application: people vs non people classification

SVM basics

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:

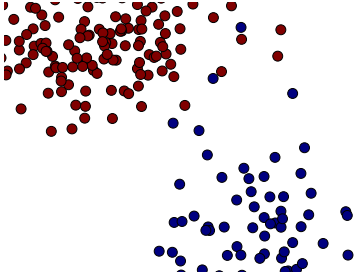
$$\begin{aligned} \min_{\vec{w},b} \quad & \frac{1}{2}||\vec{w}||^2 \\ \text{s.t.} \quad & y_i(\langle \vec{w}, \vec{x_i} \rangle + b) > 1 \quad \forall i = 1, \ldots, N \end{aligned}$$

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

$$f(\vec{u}) = \text{sign}(\langle \vec{w}, \vec{u} \rangle + b)$$

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

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

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:

$$f(\vec{u}) = \text{sign}(\langle \vec{w}, \vec{u}\rangle + b)$$

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):

!!!!!!

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

SVM: kernels (2)

You can replace that dot product with a kernel $K(\vec{x_i}, \vec{x_j}) = \langle \phi(\vec{x_i}), \phi(\vec{x_j}) \rangle$. Here, ϕ is a non linear mapping to a new space, but you don't really need to compute it!

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

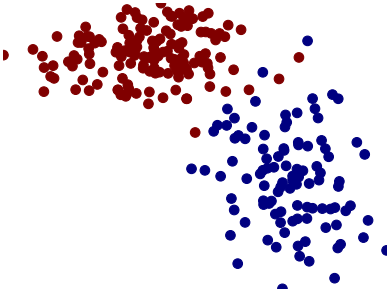
Choose a kernel:

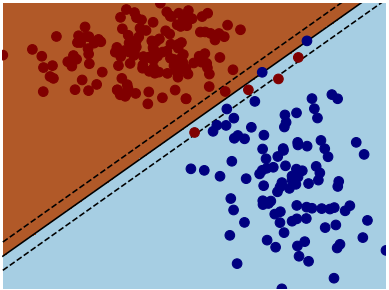
- linear: $K(\vec{x_i}, \vec{x_j}) = \langle \vec{x_i}, \vec{x_j} \rangle$
- polynomial: $K(\vec{x_i}, \vec{x_j}) = (\langle \vec{x_i}, \vec{x_j} \rangle + c)^d$
- gaussian: $K(\vec{x_i}, \vec{x_j}) = \exp(-\frac{||\vec{x_i} - \vec{x_j}||^2}{2\sigma^2})$

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

SVM example

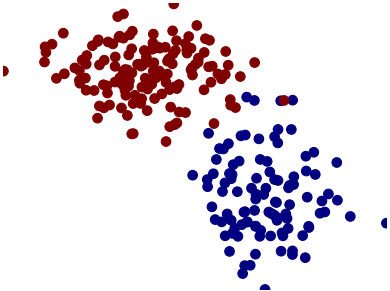
Using a linear kernel:

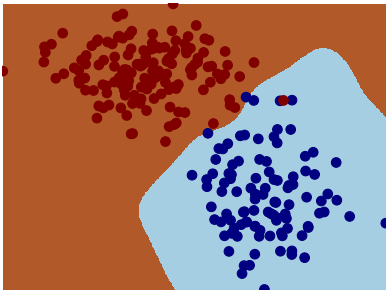




SVM example

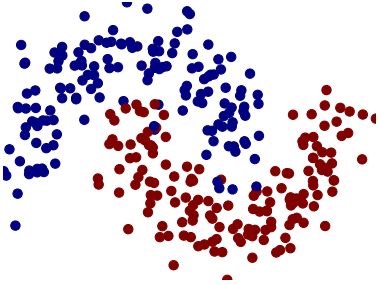
Using a rbf (gaussian) kernel:

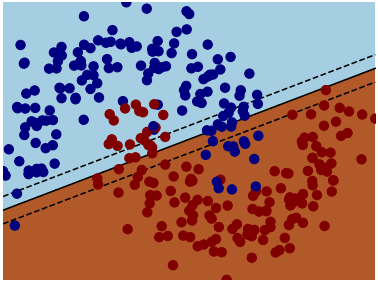




SVM example (2)

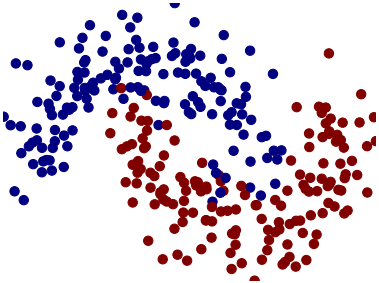
Using a linear kernel:

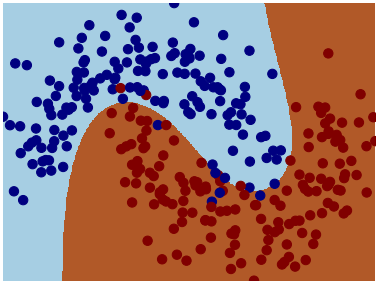




SVM example (2)

Using a rbf (gaussian) kernel:





PEGASOS

Pegasos is a training algorithm for SVMs allowing to solve the optimization problem in its **primal form** by gradient descent.

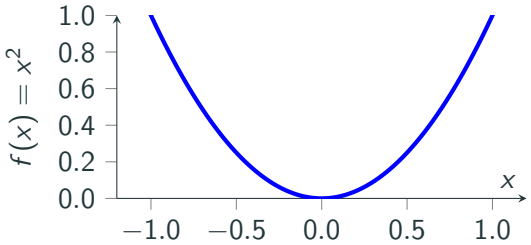
It starts from the soft margin formulation:

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

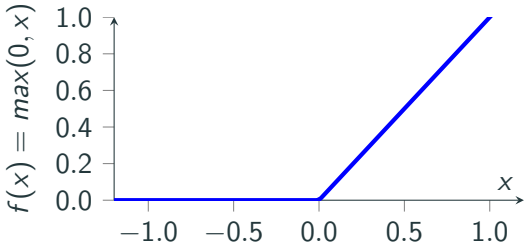
And rewrites it as:

$$\min_{\vec{w}} L(\vec{w}) = \underbrace{\frac{\lambda}{2}||w||^2}_{\text{regularization}} + \underbrace{\frac{1}{N}\sum_{i=1}^{N}\max(0, 1 - y_i\langle \vec{w}, \vec{x_i}\rangle)}_{\text{hinge loss}}$$

Subgradient for non differentiable functions



$$f(x) = x^2$$
$$\nabla f(x) = 2x$$



$$f(x) = \max(0, x)$$
$$\nabla f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if} x < 0 \end{cases}$$

We can now employ gradient descent to optimize

$$L(\vec{w}) = \frac{\lambda}{2}||\vec{w}||^2 + \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - y_i\langle \vec{w}, \vec{x_i}\rangle)$$

Algorithm 1 Gradient descent

1: $\vec{w}^0 = \vec{0}$
2: **for** $t = 0, \ldots,$ n_steps-1 **do**
3: $\quad \vec{w}^{t+1} = \vec{w}^t - \eta^t \nabla L(\vec{w}^t)$
4: **end for**

Where:

- $\eta^t = \frac{1}{t\lambda}$ (step size)
- $\nabla L(\vec{w}^t) = \lambda \vec{w}^t - \frac{1}{N}\sum_{y_i\langle \vec{w}^t, \vec{x_i}\rangle < 1} y_i \vec{x_i}$

Algorithm 2 PEGASOS

$\vec{w}^0 = \vec{0},\ t = 0$
for epoch in $1, \ldots,$ n_epochs **do**
  for $j = 1, \ldots, N$ **do**
    $t = t + 1$
    $\eta^t = \frac{1}{t\lambda}$
    if $y_j \langle w_t \vec{x_j} \rangle < 1$ **then**
      $\vec{w}^{t+1} = (1 - \eta^t \lambda)\vec{w}^t + \eta^t y_j x_j$
    else
      $\vec{w}^{t+1} = (1 - \eta^t \lambda)\vec{w}^t$
    end if
  end for
end for

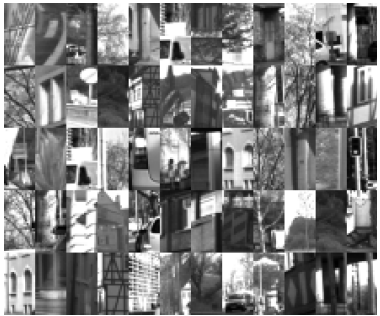
Application: people vs non people classification

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()