Introductory Notes on Support Vector Machines

Anca Ralescu
Machine Learning and Computational Intelligence Laboratory
Department of Computer Science
University of Cincinnati
Cincinnati, Ohio 45221, USA
ancaralescu@gmail.com

March 31, 2009

- Introduce the notion of **margin** and **support vectors** working out directly from the formulation of the classification problem.
- Objective: build classifiers for data, that is **Algorithms for partitioning the data**.
- Two main approaches:
  - **Statistical**(parametric/classical): Data to be classified comes from a population whose underlying distribution (its parameters) are known: this means that we know the **probability density/mass function** which can be used for classification of a new data point. However, in real world problems we do not know the distribution underlying the data. In fact in theoretical studies of probability/statistics this is a much researched problem. We can say that **it** is a (machine) learning problem;
  - **Non-parametric**: Given known classifications (training data) $\Longrightarrow$ derive a rule (**decision functions**) for classifying unknown/new data points: $g(x)$ In a two class problem we have:

$$\begin{cases} g(x) = 0 & \text{boundary} \\ g(x) > 0 & \text{class 1} \\ g(x) < 0 & \text{class 2} \end{cases} \tag{1}$$

- We will be concerned with the Non-parametric case.

1 Linear Classifiers

To begin with, we will be concerned only with linear classifiers. This means that we start by assuming that the data set **is linearly separable**.

Then, in the case when this is not true, the data set is NOT linearly separable, we will use a device which makes this data linearly separable. This device maps data implicitly into a higher dimension where it becomes linearly separable.

In the case of **linear classifiers** the decision function $g(\mathbf{x})$ is linear in $\mathbf{x}$, that is

$$g(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x} + b) \tag{2}$$

where *sign* (*signum*)is defined as

$$sign(a) = \begin{cases} -1 & \text{if } a < 0 \\ 0 & \text{if } a = 0 \\ +1 & \text{if } a > 0 \end{cases}$$

The job of the classifier is to find the weight vector $\mathbf{w}$ and the constant term b (bias).

2 Example

Consider the data from figure 1

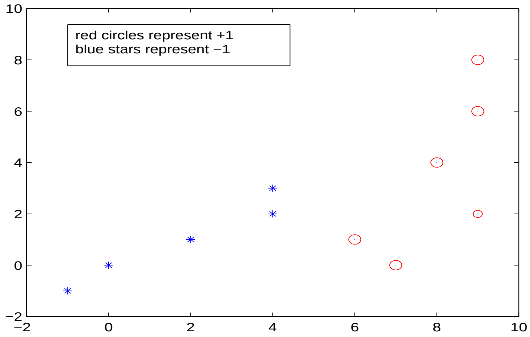

Figure 1: A two class example set

How do we separate the two classes represented there? Figure 2 shows three decision functions (lines) for separating these points.

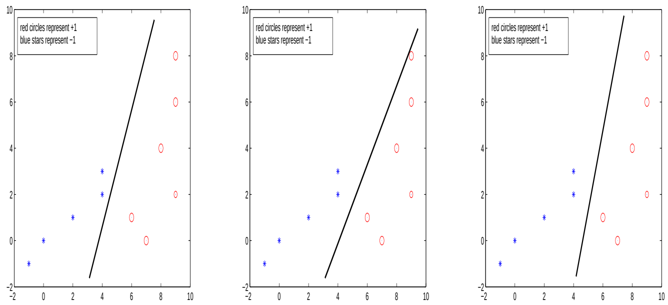

Figure 2: Three decision functions (lines) for the data in Figure 1.

Obviously there are **infinitely many such decision functions** for this data set!!!! So, the question that arises is **which is the <u>best</u> decision function and how to determine it**.

The use to the word best suggest that we must come up with some criterion with respect to which a solution (here a decision function) is judged to be best.

3 Margin

To do this we introduce the concept of **margin**. Informally we will define the margin as follows: let $\mathbf{w} \cdot \mathbf{x} + b$ be the boundary. Then the margin is the **width** by which the boundary can be increased before hitting a data point, illustrated in Figure 3.

The data points **hit** by the margin are called **support vectors**. Obviously for each decision function/boundary we have different margins and therefore possibly different support vectors. The **maximum margin linear classifier** is the classifier with the largest/widest margin. The resulting classifier is called **Linear Support Vector Machine** (LSVM), the simplest kind of SVM.

Next we need to actually find the classifier, which means finding $\mathbf{w}$ and b in equation (2).

Remark 1 *It is fair to ask why we look for the maximum margin. There are answers both from intuitive point of view and from theoretical point of view. From intuitive point of view:*

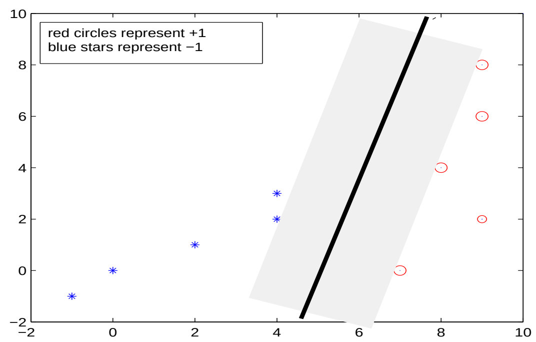

Figure 3: Margin for a two class example set

- *it looks like this will give the best behavior with respect to possible mistakes (best generalization error) for both classes;*
- *experimental results showed that this works VERY well;*
- *the model is immune to removal, addition to data points which are outside the margin;*

From theoretical point of view the arguments come mainly from the PAC/VC dimension learning theory.

The planes delimiting the margin are actually very similar to the boundary, as shown in figure 4. Let $D_c(\mathbf{x}) \equiv \mathbf{w} \cdot \mathbf{x} + b = c$.

$$\begin{cases} \text{the } +1 \text{ plane:} & D_{+1}(\mathbf{x}) \\ \text{boundary:} & D_0(\mathbf{x}) \\ \text{the -1 plane:} & D_{-1}(\mathbf{x}) \end{cases} \tag{3}$$

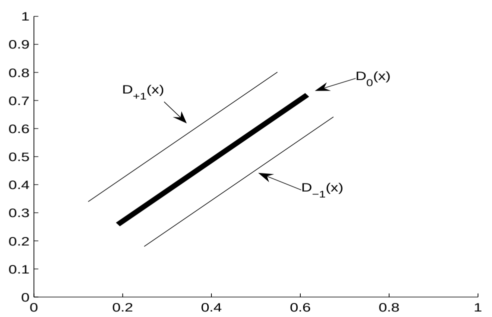

Figure 4: Sketch of $D_0(\mathbf{x}), D_{+1}(\mathbf{x}), D_{-1}(\mathbf{x})$

The classification is then done according to the following rules: For the unknown vector $\mathbf{x}_0$

$$Class(\mathbf{x}_0) = \begin{cases} +1 & \text{if} \quad \mathbf{w} \cdot \mathbf{x} + b \geq 1 \\ -1 & \text{if} \quad \mathbf{w} \cdot \mathbf{x} + b \leq -1 \\ \text{can't tell} & \text{if} \quad -1 < \mathbf{w} \cdot \mathbf{x} + b < +1 \end{cases} \tag{4}$$

4 Computing the margin

We want to compute the margin width, M, in terms of $\mathbf{w}$ and b.

Claim 1 *The vector* $\mathbf{w}$ *is perpendicular on both* D_{+1} *and* D_{-1}.

It is easy to see this from computing $\mathbf{w} \cdot (\mathbf{u} - \mathbf{v})$ for each case, $\mathbf{u}, \mathbf{v} \in D_{+1}$, and $\mathbf{u}, \mathbf{v} \in D_{-1}$.

Next let $\mathbf{u}^- \in D_{-1}$ an arbitrary vector (need not be support vector). Let $\mathbf{u}^+ \in D_{+1}$ nearest to $\mathbf{u}^-$: Obviously this is on the perpendicular from $\mathbf{u}^-$ on D_{+1}.

Claim 2 *There exists* λ *such that*

$$\mathbf{u}^+ = \mathbf{u}^- + \lambda \mathbf{w}$$

Since the line connecting $\mathbf{u}^+$ and $\mathbf{u}^-$ is perpendicular on all the planes, and $\mathbf{w}$ is also perpendicular, to get from $\mathbf{u}^-$ to $\mathbf{u}^+$ one has to travel some distance in the direction $\mathbf{w}$.

Thus we have

$$M = |\mathbf{u}^+ - \mathbf{u}^-| = \mathbf{u}^- = |\mathbf{u}^- + \lambda \mathbf{w} - \mathbf{u}^-| = |\lambda \mathbf{w}|$$

$$\begin{aligned}
&\mathbf{w} \cdot \mathbf{u}^+ + b = +1 \\
&\Longrightarrow \mathbf{w} \cdot (\mathbf{u}^- + \lambda \mathbf{w}) + b = +1 \\
&\Longrightarrow -1 + \lambda \mathbf{w} \cdot \mathbf{w} = 1 \\
&\Longrightarrow \lambda = \tfrac{2}{\mathbf{w} \cdot \mathbf{w}}
\end{aligned}$$

Thus

$$M = \lambda |\mathbf{w}| = \lambda \sqrt{\mathbf{w} \cdot \mathbf{w}} = \frac{2\sqrt{\mathbf{w} \cdot \mathbf{w}}}{\mathbf{w} \cdot \mathbf{w}} = \frac{2}{\sqrt{\mathbf{w} \cdot \mathbf{w}}}$$

The next issue is how to learn the maximum margin linear classifier. There are several approaches. We will pursue that using Quadratic Programming(QP).

5 Quadratic Programming

Constrained optimization problems:

- Objective function: the function to be optimized (maximize)
- Constraints: inequalities / equalities

When the objective function is quadratic (degree 2) we call this quadratic optimization problem.

A trivial (quadratic) optimization problem subject to constraints: find the maximum area that can be surrounded by a fence of fixed length(l): Let L, W be the unknown length and the width of the area we are looking for: $A = L \times W$. We know that $2(L + W) = l$. The optimization problem is

$$\begin{aligned}
&\text{Maximize } A = L \times W \\
&\text{subject to} \\
&2(L + W) = l
\end{aligned}$$

From the formula for the margin, $M = \frac{2}{\sqrt{\mathbf{w} \cdot \mathbf{w}}}$ it follows that to maximize M we need to minimize $\sqrt{\mathbf{w} \cdot \mathbf{w}}$, or equivalently

$$\textbf{Minimize } \mathbf{w} \cdot \mathbf{w}$$

which is a quadratic (degree 2) expression. Now this minimization is subject to the conditions that the training points must be correctly classified. That is, if $\mathbf{x}^+$ is in the positive class, it must satisfy

$$D_{+1}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \geq +1$$

and if it is in the negative class, then

$$D_{-1}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \leq -1$$

Now denoting by $y = \pm 1$ the label for $\mathbf{x}$ in each case, and multiplying the corresponding D with it we obtain the expression

$$y(\mathbf{w} \cdot \mathbf{x} + b) \geq 1$$

Thus, given n training points, we will have the constraints

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ for } i = 1, \ldots, n$$

To summarize then, our quadratic programming problem is

$$\begin{array}{l} \text{Minimize } ||\mathbf{w}|| \\ \text{subject to} \\ y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ for } i = 1, \ldots, n \end{array}$$