Support Vector Machines

From algebra: dot products

The dot product of two vectors $\mathbf{x}, \mathbf{y} \in \mathcal{R}^n$: $\mathbf{xy} = \sum_{i=1}^{n} x_i y_i$

The dot product $\mathbf{xx}$ is the square of the length of the $\mathbf{x}$ (i.e. $\mathbf{xx} = ||\mathbf{x}||^2$)

Geometric interpretation

If $\mathbf{y}$ is a unit vector (i.e.$||\mathbf{y}|| = 1$), then $\mathbf{xy} = ||\mathbf{x}|| \cos\alpha$ is a projection of $\mathbf{x}$ in direction of $\mathbf{y}$.

Section 1. Separating the training data with a hyperplane

A linear classifier: a separable case

The equation of a general hyperplane is

$$\mathbf{wx} + b = 0$$

with $\mathbf{x} \in X$ being the point of $\mathcal{R}^n$, $\mathbf{w} \in \mathcal{R}^n$ the weights and $b \in \mathcal{R}$.

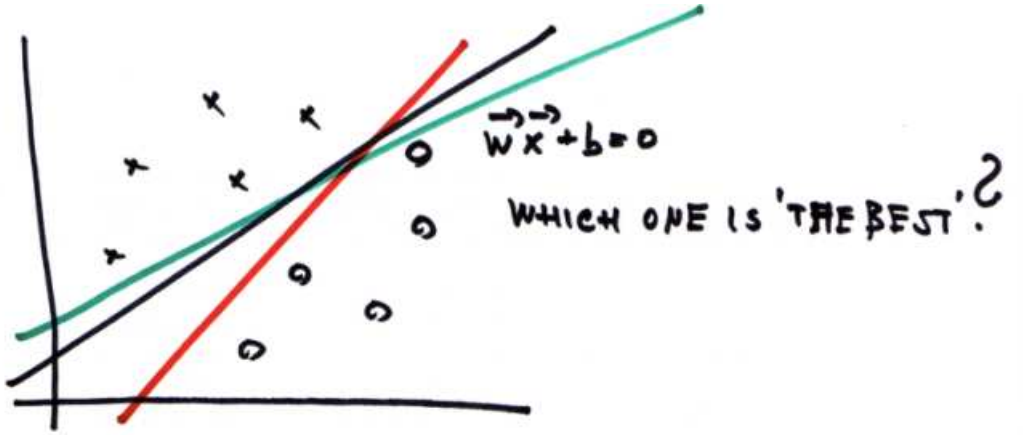
- Distance from the hyperplane to origin = $\frac{-b}{||\mathbf{w}||}$;
- Distance from an arbitrary point $\mathbf{x}'$ to the hyperplane = $\frac{\mathbf{wx}'+b}{||w||}$;

Classifier margin

- Find the closest points to the separating line. Draw two lines parallel to the separating line and passing through these points. A **margin** of a linear classifier is the distance between these two lines.
- A **maximum margin linear classifier** is the linear classifier with the maximum margin.

Let's look at a simple 2-dimensional example. Assume a classification into two classes and the features of the instances are of continuous values.

Figure 1



A linear separation of the data

Finding the hyperplane

Let the training set consists of $\langle \mathbf{x}_i, y_i \rangle$ pairs and $y_i \in \{+1, -1\}$.

Let the perpendicular distance from the hyperplane to the nearest $+1$ class point be denoted d_2 and similarly d_1 for other class.

The margin M is defined $M = d_2 + d_1$.

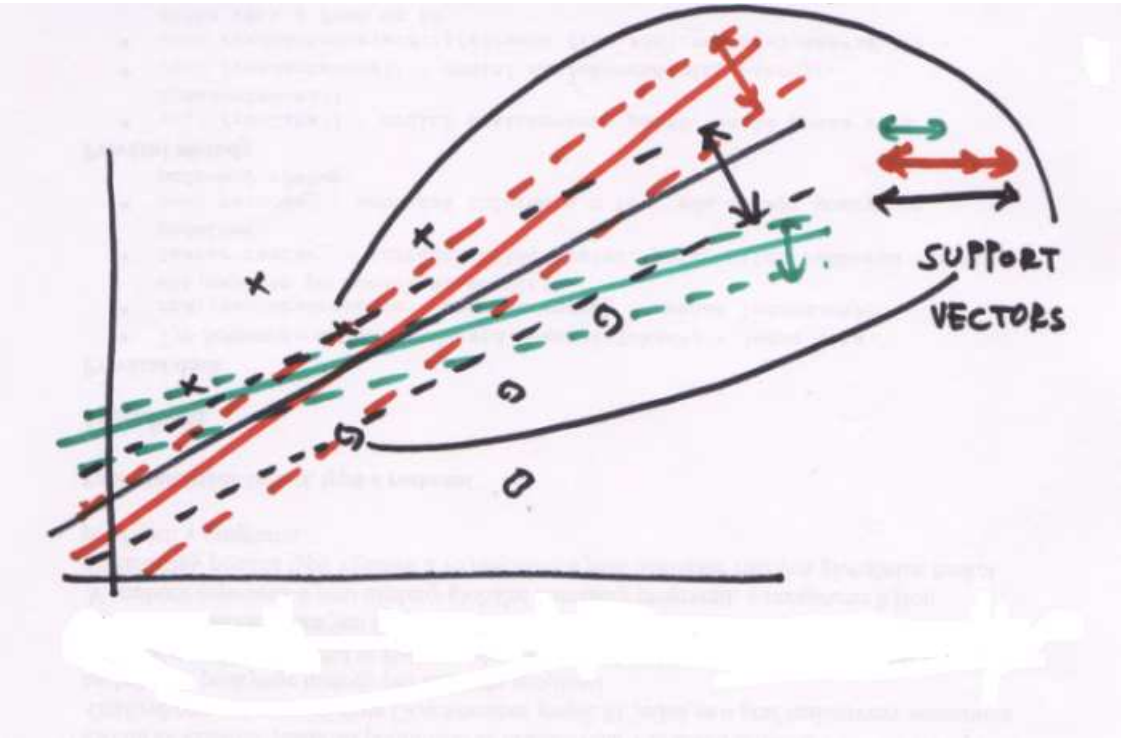
SVM solution looks for the weight vector $\mathbf{w}$ and paramater b that maximizes M, i.e.

The hyperplane should separate the data, so that $\mathbf{w}\mathbf{x}_k + b > 0$ for all $\mathbf{x}_k$ of one class, and $\mathbf{w}\mathbf{x}_k + b < 0$ for all $\mathbf{x}_j$ of other class.

If the data are separable in this way, there is probably more than one way to do it.

Among the possible hyperplanes, SVMs select the one where the margin of the hyperplane from the closest data points is large as possible.

Figure 2



A maximum margin classifier

Computing the margin

How do we compute the margin M in terms of $\mathbf{w}$ and b?

We want an expression for the distance between the hyperplane and the closest points: $\mathbf{w}$ and b will be chosen to **maximize** this expression. Let's assume "supporting hyperplanes" parallel to the separating hyperplane and passing through the closest points (**the support vectors**). These are

$$\mathbf{w}\mathbf{x}_1 + b = 1, \mathbf{w}\mathbf{x}_2 + b = -1$$

for some points $\mathbf{x}_1, \mathbf{x}_2$ (there may be more than one such point on each side). **How it is achieved?**

We work with so-called *canonical form of a hyperplane*.

Definition The hyperplane is in canonical form w.r.t. the points $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \}$ if

$$min_{\mathbf{x}_i \in X} |\mathbf{w}\mathbf{x}_i + b| = 1.$$

Redefine the hyperplane $\mathbf{u}\mathbf{x} - d = 0$, where $\mathbf{u}$ is a unit vector, and d is the distance of the hyperplane to origin; Note that the same hyperplane is also defined by $c\mathbf{u}\mathbf{x} - cd = 0$, where c is an arbitrary positive real number.

The criterion of optimal separating hyperplane suggests that we are looking for $\mathbf{u}$, and d such that $d = \frac{d_1+d_2}{2}$, and $d_2 - d_1$ is maximized, where $\mathbf{u}\mathbf{x}_1 = d_1$, $\mathbf{u}\mathbf{x}_2 = d_2$, and $\mathbf{x}_1$ and $\mathbf{x}_2$ are the closest point to the hyperplane for each of the two classes.

Let $f(\mathbf{x}) = c\mathbf{u}\mathbf{x} - cd$; then $f(\mathbf{x}_1) = c\mathbf{u}\mathbf{x}_1 - cd = cd_1 - cd = c\frac{d_1-d_2}{2}$, $f(\mathbf{x}_2) = c\mathbf{u}\mathbf{x}_2 - cd = cd_2 - cd = c\frac{d_2-d_1}{2}$. Let $c\frac{d_2-d_1}{2} = 1$, then $f(\mathbf{x}_1) = -1$ and $f(\mathbf{x}_2) = +1$.

Hence in canonical form $f(\mathbf{x})$, to maximize $d_2 - d_1$ is equivalent to minimize $\frac{c}{2}$.

If we define $f(\mathbf{x}) = \mathbf{wx} + b$, where $\mathbf{w} = c\mathbf{u}$, and $b = -cd$ note that $c = ||\mathbf{w}||^2$.

Then in canonical form, to maximize $d_2 - d_1$ is equivalent to minimize $\frac{||\mathbf{w}||^2}{2}$.

Separating Hyperplane $\rightarrow$ Optimization task - the primal problem

$$(\mathbf{w}, b) = argmin_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$

s.t.

$$y_i(\mathbf{wx}_i + b) \geq +1, i = 1, ..., n.$$

How?

Via **quadratic programming**. QP is a class of optimization algorithms to maximize a quadratic function of some real-valued variables subject to linear constraints.

Introduce Lagrange multipliers $\lambda_i \geq 0$ and a Lagrangian

$$L(\mathbf{w}, b, \lambda) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \lambda_i(y_i(\mathbf{wx}_i + b) - 1)$$

KKT theorem states a solution to the primal problem must satisfy the following

$$\frac{\partial}{\partial b}L(\mathbf{w}, b, \lambda) = 0, \frac{\partial}{\partial \mathbf{w}}L(\mathbf{w}, b, \lambda) = 0$$

i.e.

$$\mathbf{w} = \sum_{i=1}^{n} \lambda_i y_i \mathbf{x}_i, \sum_{i=1}^{n} \lambda_i y_i = 0$$

and

$$\sum_{i=1}^{n} \lambda_i(y_i(\mathbf{wx}_i + b) - 1) = 0.$$

Solve the dual problem

Find $\lambda_i, i = 1, ..., n$ such that

$$\lambda = argmax_\lambda(\sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i\lambda_j y_i y_j \mathbf{x_i}\mathbf{x_j}),$$

s.t.

$$\lambda_i \geq 0, i = 1, .., n$$

and

$$\sum_{i=1}^{n} \lambda_i y_i = 0.$$

.

Prons: We don't have to optimize a vector $\mathbf{w}$. Instead, we optimize real numbers λ_i s.t. simple constraints.

Go back to the primal problem

$$\lambda \rightarrow \mathbf{w}, b$$

,

Use the Karush-Kuhn-Tucker conditions for an optimum with inequality constraints and dual optimization:

$$min_{\mathbf{w},b} L = \mathbf{ww} - \sum_{k=1}^{n} \lambda_k (y_k(\mathbf{wx_k} + b) - 1)$$

Taking the derivate with respect to $\mathbf{w}$ gives

$$2\mathbf{w} - \sum_{k=1}^{n} \lambda_k y_k \mathbf{x}_k = 0$$

or

$$\mathbf{w} = \frac{1}{2} \sum_{k=1}^{n} \lambda_k y_k \mathbf{x}_k.$$

The key feature is that λ_i is zero for every $\mathbf{x}_i$ EXCEPT those which lie on the hyperplanes $\mathbf{wx} + b = +1$, $\mathbf{wx} + b = -1$; these points are called the SUPPORT VECTORS.

b: Let's define a set $I = \{i; \lambda_i > 0\}$. This set consists of indeces of training points that lie in a distance either $+1$ or -1 from the separating hyperplane, i.e. these are indeces of the support vectors. Then b is

$$b = \frac{1}{|I|} \sum_{i \in I} (y_i - \sum_{j=1}^{n} \lambda_j y_j (\mathbf{x}_i, \mathbf{x}_j))$$

Conclusion

With $\mathbf{w}, b$ known the separating hyperplane is defined.

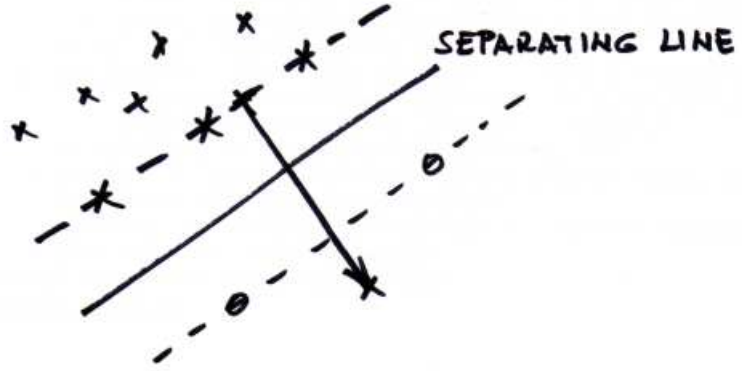
Section 2. Non-linear separation: soft margin classifier

In a real problem it is unlikely that a line will exactly separate the data -- even if a curved decision boundary is possible. So exactly separating the data is probably not desirable -- if the data has noise and outliers, a smooth decision boundary that ignores a few data points is better than one that loops around the outliers.

Figure 3

A curved decision boundary

Figure 4



Slacks

Thus

$$(\mathbf{w}, b, \xi) = argmin_{\mathbf{w},b,\xi}(\frac{1}{2}||\mathbf{w}^2|| + C\sum_{i=1}^{n}\xi_i)$$

we introduce "slack variables" $\xi_k \geq 0$ and allow $y_k(\mathbf{wx} + b) \geq 1 - \xi_k$.

s.t.

$$y_i(\mathbf{w}, \mathbf{x}_i + b) \geq +1 - \xi_i, i = 1, ..., n.$$

The idea is that we do allow the constraints $y_i(\mathbf{wx}_i + b) \geq +1$ to be violated, but only if pay a "price".

This allows that a point to be a small distance $\xi_k \geq 0$ on the wrong side of the hyperplane

without violating the stated constraint.

To avoid the trivial solution whereby huge slacks allow any line to separate the data, we add another constraints that penalize large slacks. We have a cost parameter, C, that controls the trade off between allowing training errors and forcing rigid margins. It creates a soft margin that permits some misclassifications. Increasing the value of C increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well.

Thus

$$min_{\mathbf{w},b}L = \mathbf{ww} - \sum_{k=1}^{n}\lambda_k(y_k(\mathbf{wx_k}+b)+\xi_k-1)+C\sum_{k=1}^{n}\xi_k$$

Reducing C allows more of the data to lie on the wrong side of the hyperplane and be treated as outliers, which gives a smoother decision boundary.

Section 3. Kernel trick: non-linear boundaries

If the points are separated by a nonlinear region?

Rather than fitting nonlinear curves to the data, SVM handles this by using a kernel function to map the data into a different space where a hyperplane can be used to do the separation.

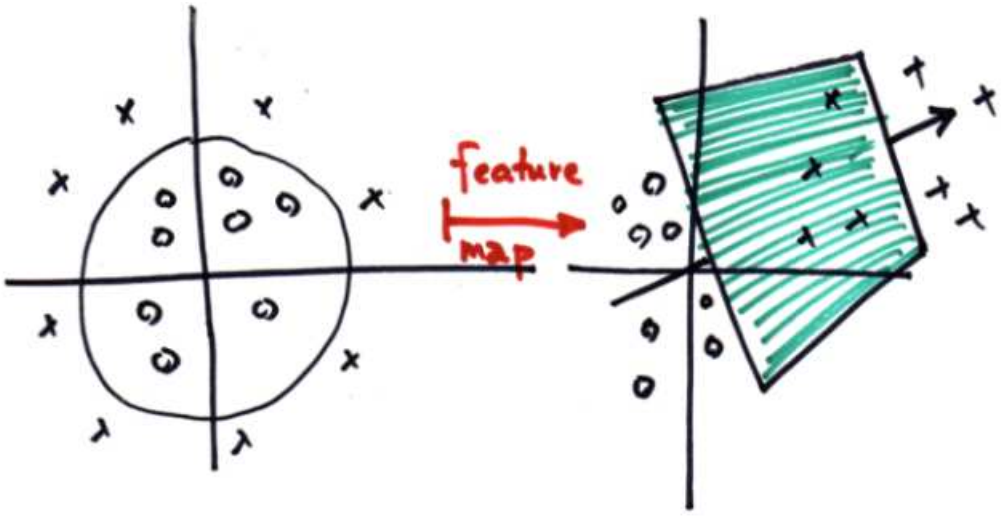
The kernel function may transform the data into a higher dimensional space to make it possible to perform the separation.

Kernel tricks

1. **Dual problem** The major point of the dual formulation is that the training data appear in the form of their dot product $\mathbf{x}_k\mathbf{x}_l$ (see above).
2. **Nonlinear map**

   The training data are passed through a nonlinear mapping $instance\_space \rightarrow feature\_space$ where the hyperplane can be used for separating the data.

Figure 5



Kernel tricks

The x, y can be mapped to three dimensions u, v, w: $u \leftarrow x, v \leftarrow y, w \leftarrow x^2 + y^2$. The dimension w (squared distance from origin) allows the data to be linearly separated by $u - v$ plane situated along w axis.

3. **"Kernel" summarizes the inner product**

   A trick is to make use of the fact that only the dot product of the data vectors are used. Every dot product is replaced by a non-linear kernel function $K(\mathbf{x}_j, \mathbf{x}_k)$:

   - polynomial $K(\mathbf{x}_j, \mathbf{x}_k) = (\mathbf{x}_j, \mathbf{x}_k)^d$
   - radial-basis function $K(\mathbf{x}_j, \mathbf{x}_k) = \frac{exp(-||\mathbf{x}_j - \mathbf{x}_k||)^2}{\sigma}$
   -
   - ...

Section 4. Multi-class classification

Binary classification is a very well developed technique.

A direct solution of multiclass problem - $Y = \{1, 2, ..., k\}$ - using a single SVM is usually avoided.

Mostly used approaches:

- **one-versus-all method using winner-takes-all strategy**

  Construct *k* classifiers. The *i*-th classifier trained taking the examples from class "*i*" as positive and the examples from all other classes as negative. For a new example $\mathbf{x}$, this strategy assigns it to the class with the largest value of the margin.

- **one-versus-one method implemented by max-wins voting**

  Construct one binary classifier for every pair of distinct classes: all together $\frac{k(k-1)}{2}$. The binary classifier $C_{i,j}$ is trained taking the examples from the class i as positive and the examples from the class j as negative. For a new example $\mathbf{x}$, if classifier $C_{i,j}$ say $\mathbf{x}$ is in class i, then the vote for class i is added by one. Otherwise, the vote for class j is inreased by one. After each of the $\frac{k(k-1)}{2}$ classifiers makes its vote, this strategy assigns $\mathbf{x}$ to the class with the largest number of votes.

- ...