MATH BEHIND SUPPORT VECTOR MACHINE

Yilin Wu

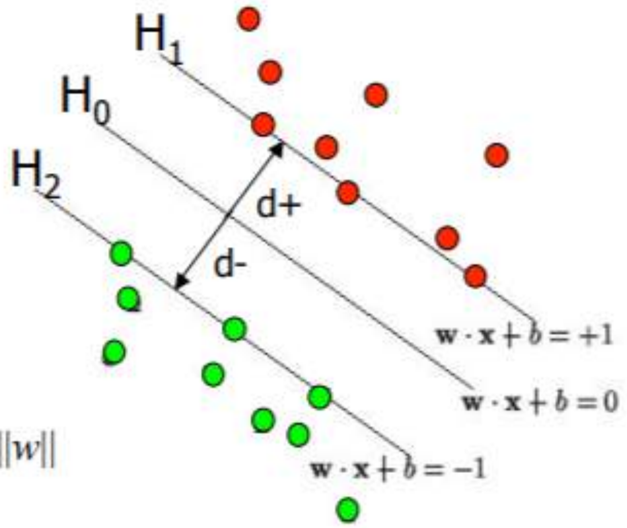
DigiPen Institute of Technology

OUTLINE

- Lagrange Multiplier
- Dual Problem
- Non-Linear SVM

Maximizing the margin (aka street width)

We want a classifier (linear separator) with as big a margin as possible.



H_1
H_0
H_2
d+
d-
$\mathbf{w}\cdot\mathbf{x}+b=+1$
$\mathbf{w}\cdot\mathbf{x}+b=0$
$\mathbf{w}\cdot\mathbf{x}+b=-1$

Recall the distance from a point(x_0,y_0) to a line: $Ax+By+c=0$ is: $|Ax_0+By_0+c|/\text{sqrt}(A^2+B^2)$, so,
The distance between H_0 and H_1 is then:
$|w\bullet x+b|/||w||=1/||w||$, so

The total distance between H_1 and H_2 is thus: $2/||w||$

In order to maximize the margin, we thus need to minimize $||w||$. With the condition that there are no datapoints between H_1 and H_2:

$$\left.\begin{array}{l}\mathbf{x}_i\bullet\mathbf{w}+b \geq +1 \text{ when } y_i = +1 \\ \mathbf{x}_i\bullet\mathbf{w}+b \leq -1 \text{ when } y_i = -1\end{array}\right\}$$ **Can be combined into: $y_i(x_i\bullet w) \geq 1$**

We now must solve a quadratic programming problem

- Problem is: minimize **$||\mathbf{w}||$, s.t.** discrimination boundary is obeyed, i.e., min $f(x)$ s.t. $g(x)=0$, which we can rewrite as:
  min f: $\frac{1}{2}\,||w||^2$ (Note this is a quadratic function)
  s.t. g: $y_i(\mathbf{w} \bullet \mathbf{x}_i) - \mathbf{b} = \mathbf{1}$ or $[y_i(\mathbf{w} \bullet \mathbf{x}_i) - \mathbf{b}] - \mathbf{1} = \mathbf{0}$

This is a **constrained optimization problem**

It can be solved by the Lagrangian multipler method

Because it is quadratic, the surface is a paraboloid, with just a single global minimum (thus avoiding a problem we had with neural nets!)

Definition

Lagrange method is used for maximizing or minimizing a general function $f(x,y,z)$ subject to a constraint (or side condition) of the form $g(x,y,z) = k$.

Assumptions made: the extreme values exist

$$\nabla g \neq 0$$

Then there is a number λ such that

$$\nabla f(x_0,y_0,z_0) = \lambda \nabla g(x_0,y_0,z_0)$$

and λ is called the Lagrange multiplier.

- Finding all values of x,y,z and λ such that

$$\nabla f(x,y,z) = \lambda \nabla g(x,y,z)$$

and $$g(x,y,z) = k$$

And then evaluating f at all the points, the values obtained are studied. The largest of these values is the maximum value of f; the smallest is the minimum value of f.

- Writing the vector equation $\nabla f = \lambda \nabla g$ in terms of its components, give

$\nabla f_x = \lambda \nabla g_x$ $\nabla f_y = \lambda \nabla g_y$ $\nabla f_z = \lambda \nabla g_z$ $g(x,y,z) = k$

- It is a system of four equations in the four unknowns, however it is not necessary to find explicit values for λ.

- A similar analysis is used for functions of two variables.

- Example 1:

A rectangular box without a lid is to be made from 12 m^2 of cardboard. Find the maximum volume of such a box.

- Solution:

let x,y and z are the length, width and height, respectively, of the box in meters.

and $V = xyz$

Constraint: $g(x, y, z) = 2xz + 2yz + xy = 12$

Using Lagrange multipliers,

$V_x = \lambda g_x \quad V_y = \lambda g_y \quad V_z = \lambda g_z \quad 2xz + 2yz + xy = 12$

which become

- $yz = \lambda(2z+y)$ (1)
- $xz = \lambda(2z+x)$ (2)
- $xy = \lambda(2x+2y)$ (3)
- $2xz + 2yz + xy = 12$ (4)
- Solving these equations;
- Let's multiply (2) by x, (3) by y and (4) by z, making the left hand sides identical.
- Therefore,
- $x\,yz = \lambda(2xz+xy)$ (6)
- $x\,yz = \lambda(2yz+xy)$ (7)
- $x\,yz = \lambda(2xz+2yz)$ (8)

- It is observed that $\lambda \neq 0$ therefore from (6) and (7)

$$2xz+xy=2yz+xy$$

which gives $xz = yz$. But $z \neq 0$, so $x = y$. From (7) and (8) we have

$$2yz+xy=2xz+2yz$$

which gives $2xz = xy$ and so (since $x \neq 0$) $y=2z$. If we now put $x=y=2z$ in (5), we get

$$4z^2+4z^2+4z^2=12$$

Since x, y, and z are all positive, we therefore have $z=1$ and so $x=2$ and $y = 2$.

- Example 2:

Find the extreme values of the function $f(x,y)=x^2+2y^2$ on the circle $x^2+y^2=1$.

- Solution:

Solve equations $\nabla f = \lambda \nabla g$ and $g(x,y)=1$ using Lagrange multipliers

Constraint: $g(x, y)= x^2+y^2=1$

Using Lagrange multipliers,

$f_x= \lambda g_x \qquad f_y= \lambda g_y \qquad g(x,y) = 1$

which become

- $2x = 2x\lambda$ (9)
- $4y = 2y\lambda$ (10)
- $x^2+y^2 = 1$ (11)
- From (9) we have $x=0$ or $\lambda=1$. If $x=0$, then (11) gives $y=\pm 1$. If $\lambda=1$, then $y=0$ from (10), so then (11) gives $x=\pm 1$. Therefore f has possible extreme values at the points (0,1), (0,-1), (1,0), (1,0). Evaluating f at these four points, we find that

- $f(0,1)=2$
- $f(0,-1)=2$
- $f(1,0)=1$
- $f(-1,0)=1$
- Therefore the maximum value of f on the circle

- $x^2+y^2=1$ is $f(0,\pm 1) =2$ and the minimum value is $f(\pm 1,0) =1$.

How Langrangian solves constrained optimization

$$L(x,a) = f(x) - ag(x) \text{ where}$$
$$\nabla(x,a) = 0$$

Partial derivatives wrt x recover the parallel normal constraint
Partial derivatives wrt λ recover the $g(x,y)=0$

In general, $$L(x,a) = f(x) + \sum_i a_i g_i(x)$$

In general

Gradient min of f

constraint condition g

$$\underset{\big\downarrow}{}L(x,a) = f(x) + \sum_i a_i g_i(x)$$ a function of $n+m$ variables

n for the $x's$, m for the a. Differentiating gives $n+m$ equations, each set to 0. The n eqns differentiated wrt each x_i give the gradient conditions; the m eqns differentiated wrt each a_i recover the constraints g_i

In our case, $f(x)$: $\frac{1}{2}\|\mathbf{w}\|^2$; $g(x)$: $y_i(\mathbf{w}\cdot x_i + b) - 1 = 0$ so Lagrangian is:

$$min\ L = \tfrac{1}{2}\|\mathbf{w}\|^2 - \Sigma a_i[y_i(\mathbf{w}\cdot x_i + \mathrm{b}) - 1] \text{ wrt } \mathbf{w}, b$$

We expand the last to get the following L form:

$$min\ L = \tfrac{1}{2}\|\mathbf{w}\|^2 - \Sigma a_i y_i(\mathbf{w}\cdot x_i + \mathrm{b}) + \Sigma a_i \text{ wrt } \mathbf{w}, b$$

Lagrangian Formulation

- So in the SVM problem the Lagrangian is

$$\min L_P = \tfrac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} a_i y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) + \sum_{i=1}^{l} a_i$$

s.t. $\forall i, a_i \geq 0$ where l is the # of training points

- From the property that the derivatives at min = 0 we get:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} a_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial L_P}{\partial b} = \sum_{i=1}^{l} a_i y_i = 0 \text{ so}$$

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{l} a_i y_i = 0$$

The Lagrangian <u>Dual</u> Problem: instead of <u>minimizing</u> over **w**, *b*, <u>subject to</u> constraints involving *a*'s, we can <u>maximize</u> over *a* (the dual variable) <u>subject to</u> the relations obtained previously for **w** and *b*

Our solution must satisfy these two relations:

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{l} a_i y_i = 0$$

By substituting for **w** and *b* back in the original eqn we can get rid of the dependence on **w** and *b*.

Note first that we already now have our answer for what the weights **w** must be: they are a linear combination of the training inputs and the training outputs, x_i and y_i and the values of *a*. We will now solve for the *a*'s by differentiating the dual problem wrt *a*, and setting it to zero. <u>Most</u> of the *a*'s will turn out to have the value <u>zero</u>. The non-zero *a*'s will correspond to the <u>support vectors</u>

Primal problem:

$$\min L_P = \tfrac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} a_i y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) + \sum_{i=1}^{l} a_i$$

s.t. $\forall i \ \mathrm{a}_i \geq 0$

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{l} a_i y_i = 0$$

Dual problem:

$$\max L_D(a_i) = \sum_{i=1}^{l} a_i - \frac{1}{2}\sum_{i=1}^{l} a_i a_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

s.t. $\sum_{i=1}^{l} a_i y_i = 0 \ \& \ a_i \geq 0$

(note that we have removed the dependence on **w** and *b*)

The Dual problem

- Kuhn-Tucker theorem: the solution we find here will be the same as the solution to the original problem
- Q: But why are we doing this???? (why not just solve the original problem????)
- Ans: Because this will let us solve the problem by computing the just the inner products of x_i, x_j (which will be very important later on when we want to solve non-linearly separable classification problems)

The Dual Problem

Dual problem:

$$\max L_D(a_i) = \sum_{i=1}^{l} a_i - \frac{1}{2}\sum_{i=1}^{l} a_i a_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

$$\text{s.t.} \ \sum_{i=1}^{l} a_i y_i = 0 \ \& \ a_i \geq 0$$

Notice that all we have are the dot products of x_i, x_j

If we take the derivative wrt a and set it equal to zero, we get the following solution, so we can solve for a_i:

$$\sum_{i=1}^{l} a_i y_i = 0$$

$$0 \leq a_i \leq C$$

Now knowing the a_i we can find the weights **w** for the maximal margin separating hyperplane:

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i$$

And now, after training and finding the **w** by this method, given an <u>unknown</u> point u measured on features x_i we can classify it by looking at the sign of:

$$f(x) = \mathbf{w} \bullet \mathbf{u} + b = (\sum_{i=1}^{l} a_i y_i \mathbf{x_i} \bullet \mathbf{u}) + b$$

Remember: <u>most</u> of the weights $\mathbf{w_i}$, i.e., the a's, will be <u>zero</u>
Only the support vectors (on the gutters or margin) will have nonzero weights or a's – this reduces the dimensionality of the solution

Inner products, similarity, and SVMs

Why should inner product kernels be involved in pattern recognition using SVMs, or at all?

– Intuition is that inner products provide some measure of ‘similarity’

– Inner product in 2D between 2 vectors of unit length returns the cosine of the angle between them = how ‘far apart’ they are

e.g. $\mathbf{x} = [1, 0]^T$, $\mathbf{y} = [0, 1]^T$

i.e. if they are parallel their inner product is 1 (completely similar)

$$\mathbf{x}^T \mathbf{y} = \mathbf{x} \bullet \mathbf{y} = 1$$

If they are perpendicular (completely unlike) their inner product is 0 (so should not contribute to the correct classifier)

$$\mathbf{x}^T \bullet \mathbf{y} = \mathbf{x} \bullet \mathbf{y} = 0$$

Insight into inner products

Consider that we are trying to maximize the form:

$$L_D(a_i) = \sum_{i=1}^{l} a_i - \frac{1}{2}\sum_{i=1}^{l} a_i a_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

$$\text{s.t. } \sum_{i=1}^{l} a_i y_i = 0 \;\&\; a_i \geq 0$$

The claim is that this function will be <u>maximized</u> if we give nonzero values to a's that correspond to the support vectors, ie, those that 'matter' in fixing the maximum width margin ('street'). Well, consider what this looks like. Note first from the constraint condition that all the a's are positive. Now let's think about a few cases.

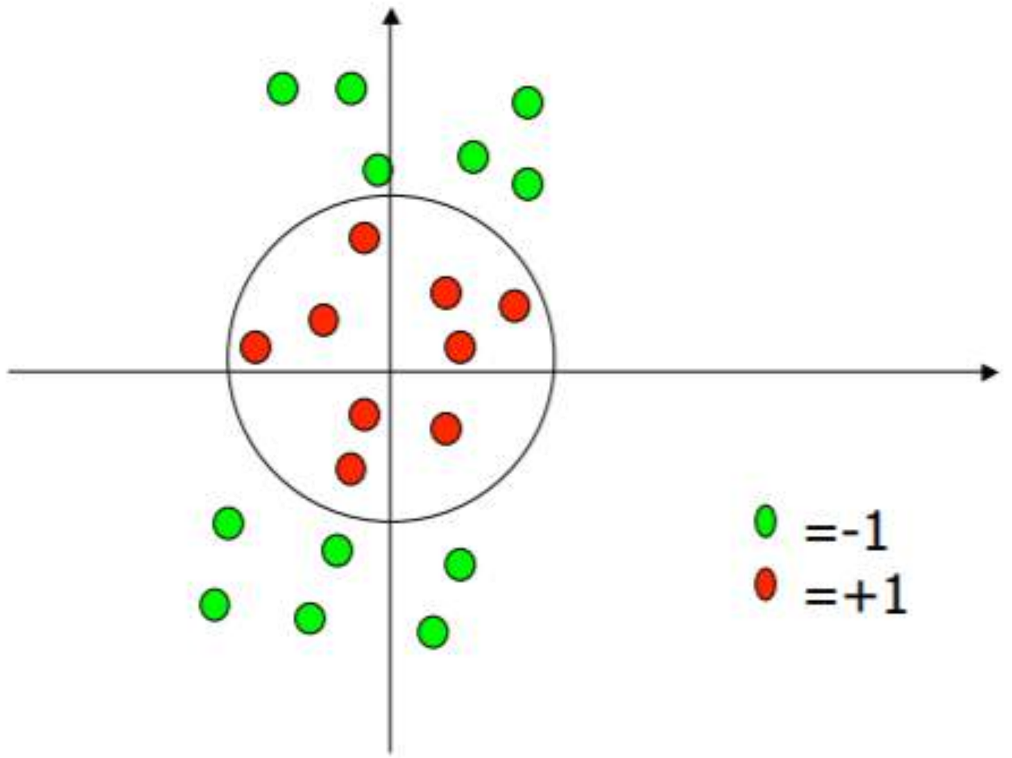
Case 1. If two features x_i , x_j are completely <u>dissimilar</u>, their dot product is 0, and they don't contribute to L.

Case 2. If two features x_i,x_j are completely <u>alike</u>, their dot product is 0. There are 2 subcases.

Subcase 1: both x_i,and x_j predict the <u>same</u> output value y_i (either +1 or –1). Then y_i x y_j is always 1, and the value of $a_ia_jy_iy_jx_ix_j$ will be positive. But this would <u>decrease</u> the value of L (since it would subtract from the first term sum). So, the algorithm downgrades similar feature vectors that make the <u>same</u> prediction.

Subcase 2: x_i,and x_j make <u>opposite</u> predictions about the output value y_i (ie, one is +1, the other –1), but are otherwise very closely similar: then the product $a_ia_jy_iy_jx_ix$ is <u>negative</u> and we are <u>subtracting</u> it, so this <u>adds</u> to the sum, maximizing it. This is precisely the examples we are looking for: the critical ones that tell the two classses apart.

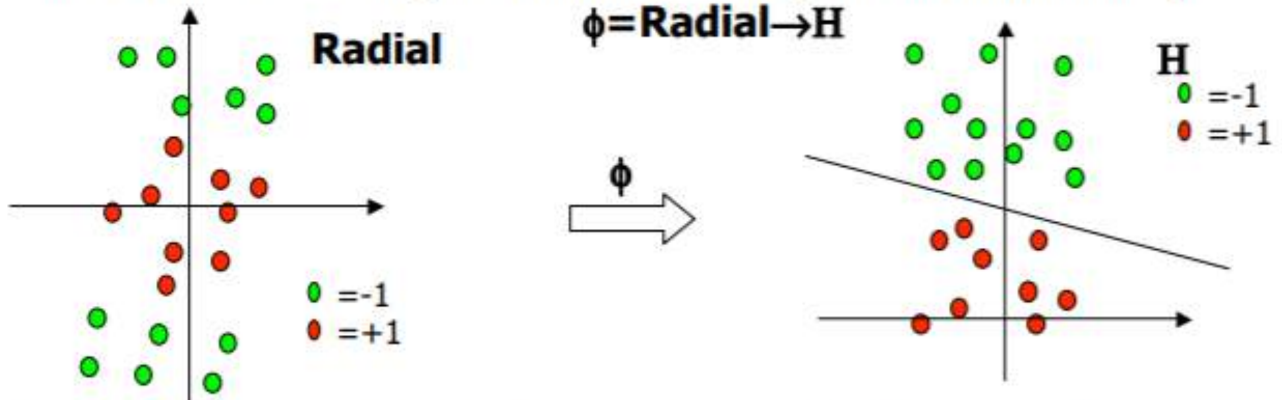
Problems with linear SVM



What if the decision function is not linear? What transform would separate these?

Ans: polar coordinates!
Non-linear SVM

The Kernel trick Imagine a function ϕ that maps the data into another space:



Remember the function we want to optimize: $L_d = \sum a_i - \frac{1}{2}\sum a_i a_j y_i y_j (x_i \cdot x_j)$ where $(x_i \cdot x_j)$ is the dot product of the two feature vectors. If we now transform to ϕ, instead of computing this dot product $(x_i \cdot x_j)$ we will have to compute $(\phi(x_i) \cdot \phi(x_j))$. But how can we do this? This is expensive and time consuming (suppose ϕ is a quartic polynomial… or worse, we don't know the function explicitly. Well, here is the neat thing:

If there is a "kernel function" K such that $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, then we do not need to know or compute ϕ at all!! That is, the kernel function defines inner products in the transformed space. Or, it defines similarity in the transformed space.

Non-linear SVMs

So, the function we end up optimizing is:

$L_d = \Sigma a_i - ½\Sigma a_i a_j y_i y_j K(x_i \cdot x_j)$,

Kernel example: The polynomial kernel

$K(xi,xj) = (x_i \cdot x_j + 1)^p$, where p is a tunable parameter

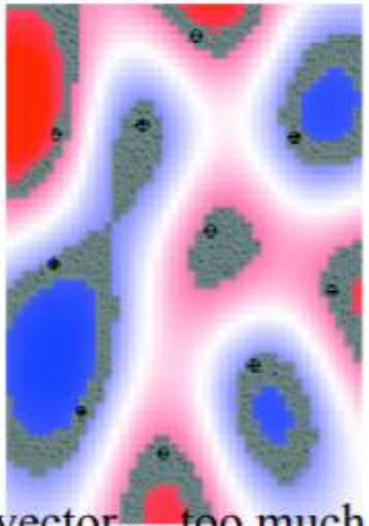
Note: Evaluating K only requires <u>one</u> addition and <u>one</u> exponentiation more than the original dot product

Kernels generalize the notion of ‘inner product similarity’

Note that one can define kernels over more than just vectors: strings, trees, structures, … in fact, just about anything

A very powerful idea: used in comparing DNA, protein structure, sentence structures, etc.

Overfitting by SVM



Every point is a support vector… too much freedom to bend to fit the training data – no generalization.
In fact, SVMs have an 'automatic' way to avoid such issues, but we won't cover it here… see the book by Vapnik, 1995. (We add a penalty function for mistakes made after training by over-fitting: recall that if one over-fits, then one will tend to make errors on new data. This penalty fn can be put into the quadratic programming problem directly. You don't need to know this for this course.)