

try to understand SVM

reference: An Idiot's guide to Support vector machines (SVMs)

R. Berwick, Village Idiot

abbreviations in notes:

- *wrt.*: with regard to.
- *s.t.*: subject to.
- MRLine: the line has max margin.

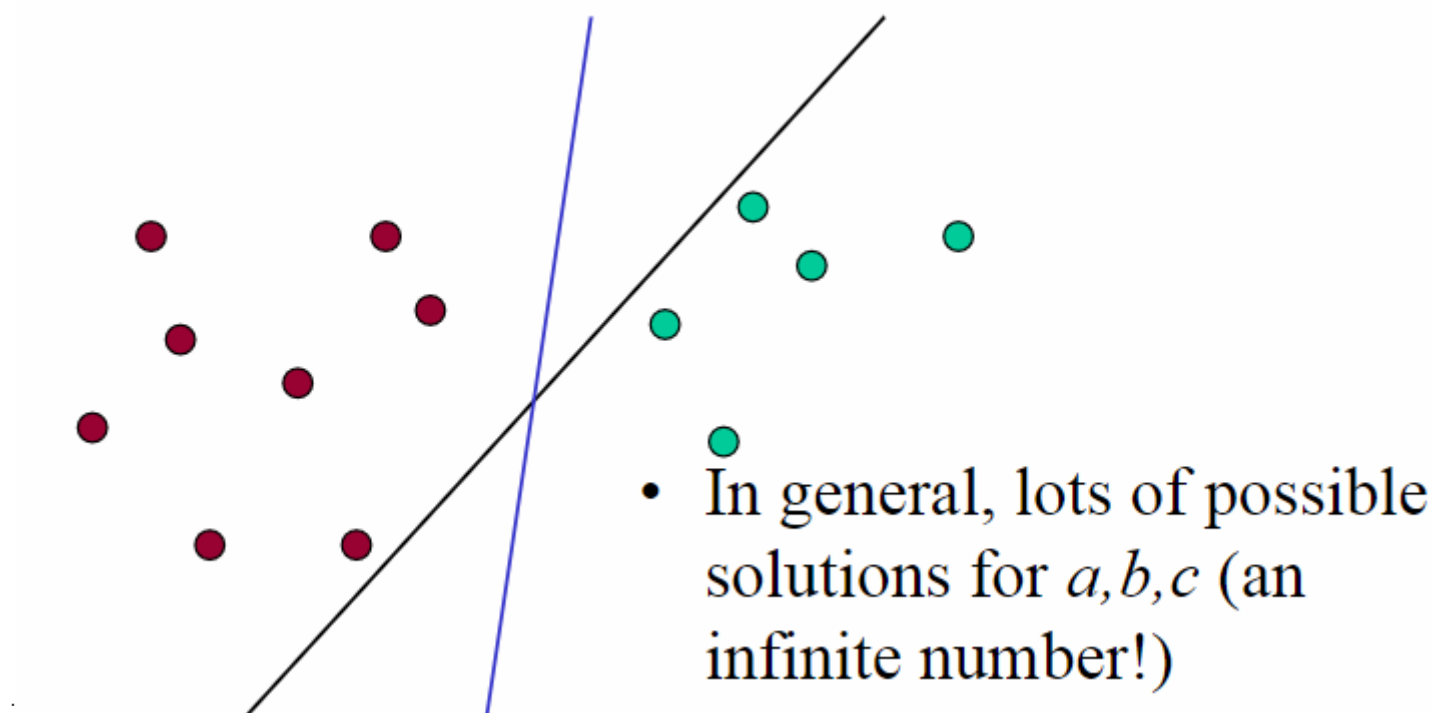
keywords:

max margin/hyperplane/Lagrangian/support vectors/mapping/kernels

here we go

1. origin: how to make a classification.

take the simplest classification problem: **linear classification** as an example.



there are infinite possible lines to separate data set, and we want the optimal one.

Which one?

the line has max margin

further, in high dimensions we still want to find the hyperplane, which has the max margin, to be our solution.

2. how to find the line

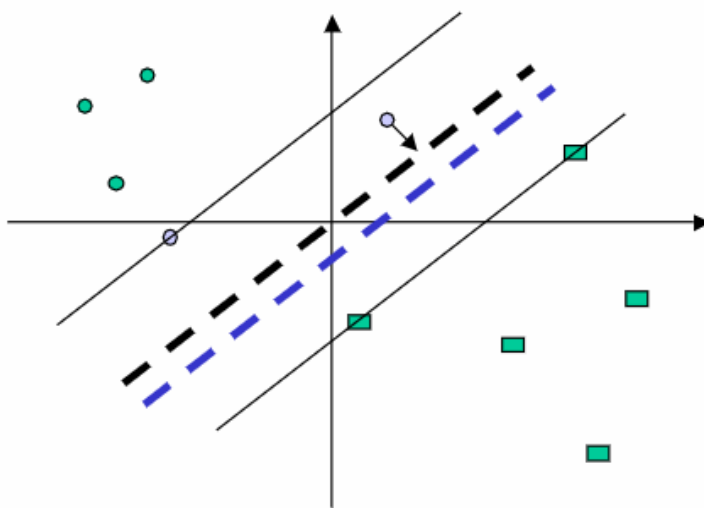
to answer this question, we need to figure out:

what factors determine the MRLine

Let's recall logistic regression and neural network first.

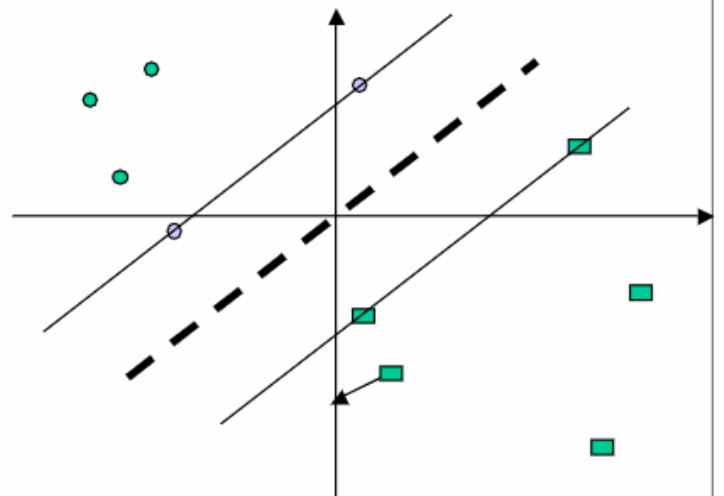
In those methods, we build up model by using **all** examples we have. In other words, we considered that **each examples** has influence on our optimal line.

Actually, MRLine only be influenced by some **fatal points**.



Moving the other vectors
has no effect

Moving a support vector
moves the decision
boundary



these points determine MRLine, which means **if one had been removed, MRLine changes.**

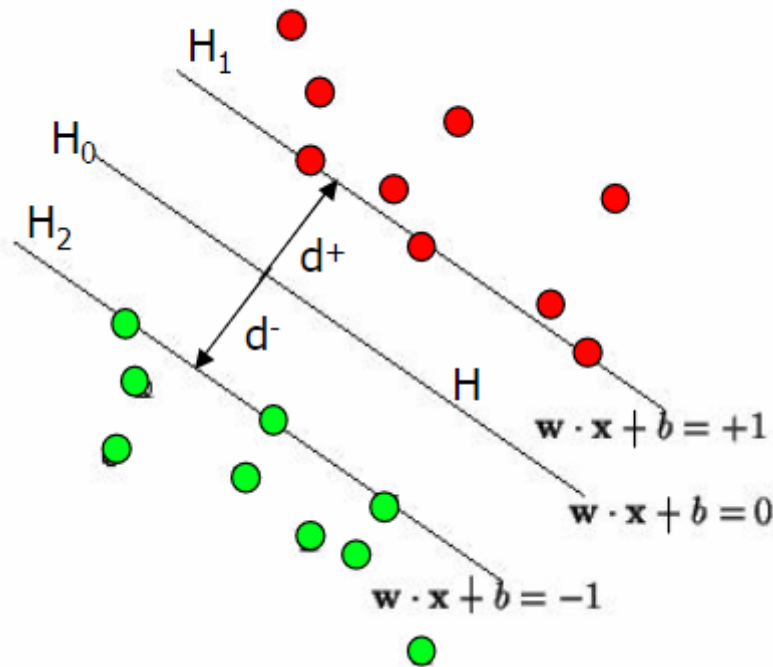
and these fatal points, are the **Support Vectors**.

While other examples have no influence on our MRLine.

3. Solve the Support Vectors.

this part need understanding of Calculus and is quite hard to understand, but keep patient, we can work out.

3.1. brief definition of MRLine**



page 8

$$MRLine : w * x + b = 0$$

- w is a weight vector
- x is input vector
- b is bias

notes:

1. when considering 2-D situation, the vector x just $x=[x,y]$.
(the y here just another feature given by input. To lead $w * x + b = 0$ more like the un-vectorized one (most people familiar with), we can consider it as y . our y is used to determine which side should the example locate).
2. notice that we separate positive and negative examples by $+1$ and -1 .
 d in picture is our margin.

3.2 describe the margin

the distance from a point to line is:

$$p(x_0, y_0), \text{Line} : AX + BY + C = 0$$
$$dist = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

and vectorize it, we have below universal one:

$$dist = \frac{|w \cdot x + b|}{\|w\|}$$
$$= \frac{1}{\|w\|}$$

note that because we have $|w \cdot x + b| = 1$, so we get that equation.

3.3 maximize the margin

In 3.2 we successfully gain the equation to measure margin. Now just recall our original task: **maximizing the margin**.

Recall the equation:

$$dist = \frac{1}{\|w\|}$$

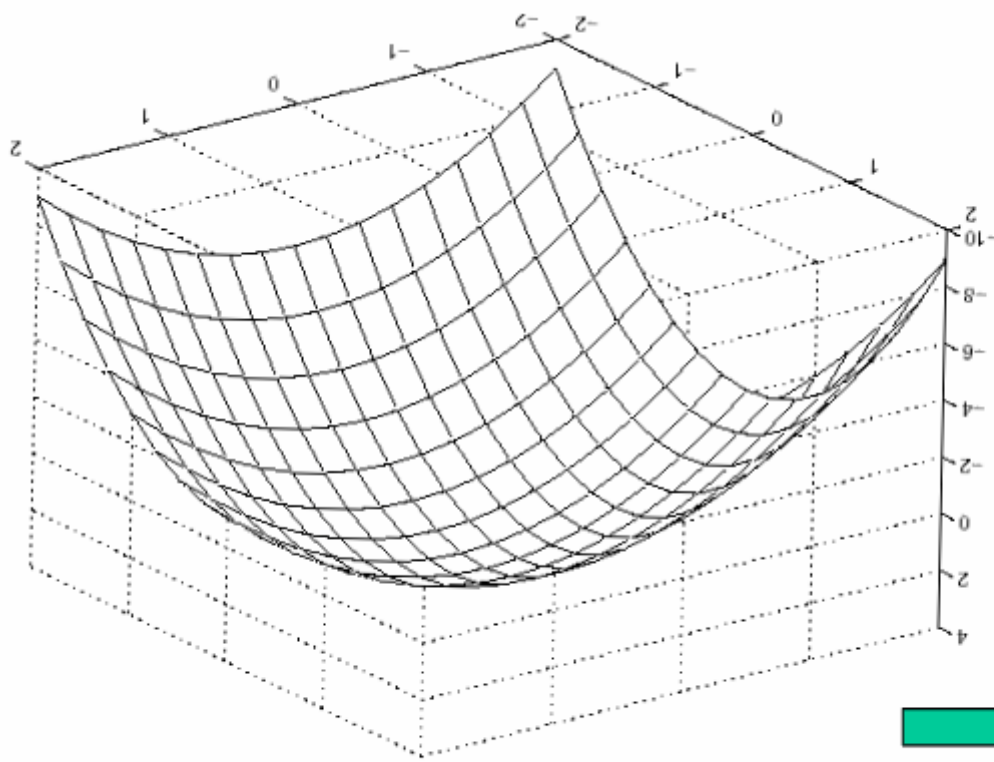
Here is a quadratic function. And our task has changed into a

constrained optimization problem.

which can be solved by the **Lagrangian multiplier method**.

What's more, because it is quadratic, the surface is a paraboloid, with just a **single global minimum**, which means we won't **stuck in local optimum**!

Really fantastic isn't it?



flatten

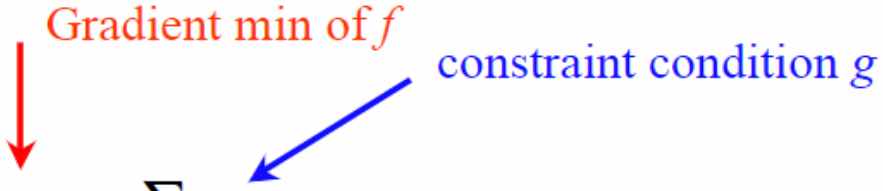
Example: paraboloid $2+x^2+2y^2$ s.t. $x+y=1$

page 11

3.4 Lagrangian

Let me just quote pdf (whose explanation is smart and brief), and give some notes.

In general


 $L(x, a) = f(x) + \sum_i a_i g_i(x)$ 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, $f(x): \frac{1}{2} \|\mathbf{w}\|^2$; $g(x): y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0$ so Lagrangian is:

$$\min L = \frac{1}{2} \|\mathbf{w}\|^2 - \sum a_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \text{ wrt } \mathbf{w}, b$$

We expand the last to get the following L form:

$$\min L = \frac{1}{2} \|\mathbf{w}\|^2 - \sum a_i y_i (\mathbf{w} \cdot \mathbf{x}_i + b) + \sum a_i \text{ wrt } \mathbf{w}, b$$

derivatives here.

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

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

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

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

page 15

And now come to the tricky moment: instead solving the primal form, we actually solving for **the dual of this original problem**.

Let's see dual problem first then explain why.

Primal problem:

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

$$\text{s.t. } \forall i \ a_i \geq 0$$

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

Dual problem:

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

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

inner product

(note that we have removed the dependence on \mathbf{w} and b)

page 16

Reasons below:

1. By substituting we can get rid of the dependence on \mathbf{w} and b .
2. the dual one solving the problem by computing just

the inner products of $\mathbf{x}_i \cdot \mathbf{x}_j$

(Notice that $\mathbf{x}_i, \mathbf{x}_j$ is vectors).

This is important when we expanding into **non-linearly** separable classification problems.

Up to now, we can finally find out **support vectors**!

by solving the weights w :

$$w = \sum a_i x_i y_i$$

Most of w would equal to **zero**,

Only the **support vectors** (on the gutters or margin) will have **non-zero**

weights.

4. prediction

And now, after training and finding the \mathbf{w} by this method, given an unknown point u measured on features x_i we can classify it by looking at the sign of:

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

page 18

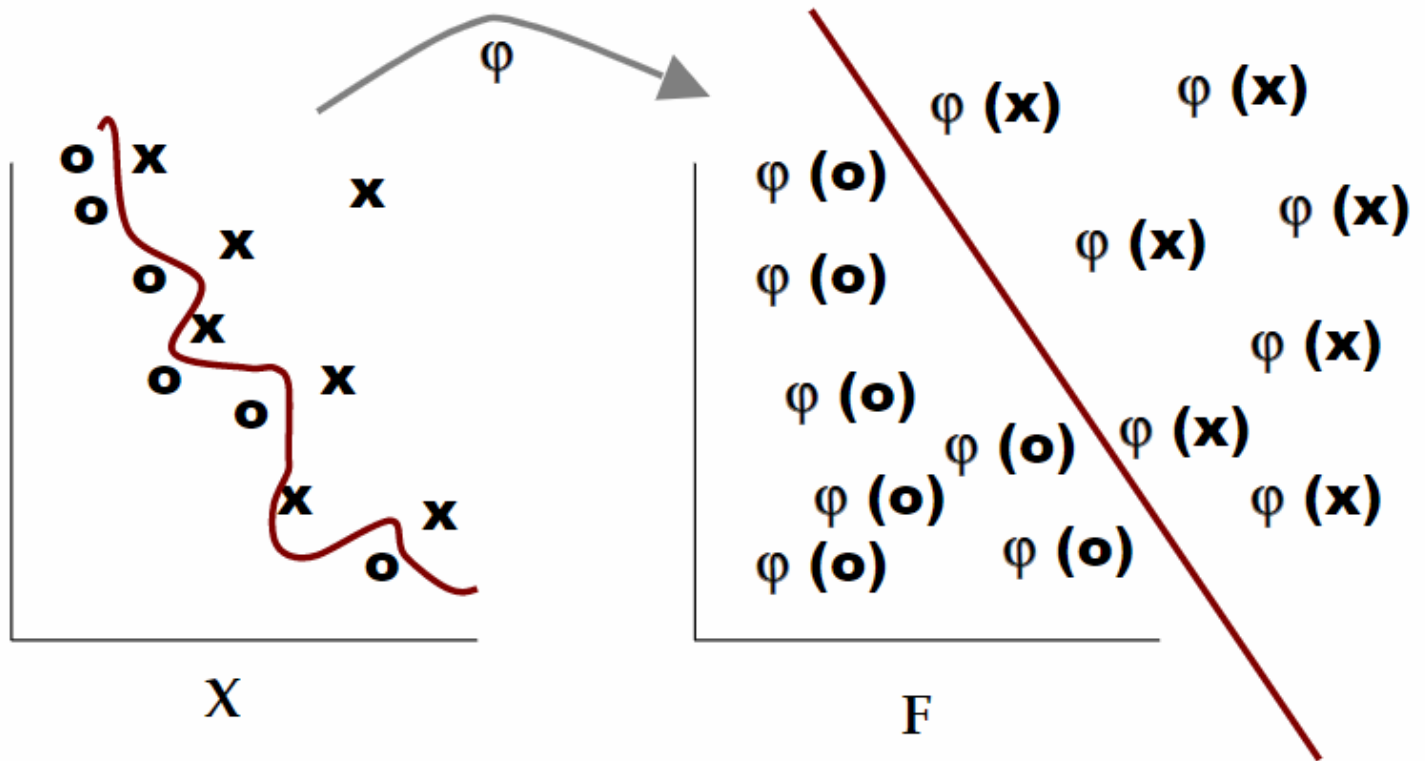
5. non-linear separable classification

5.1 mapping

Facing problem which can't be separate by line, we just want it can be seperated.

So we have to **mapping!!!**.

Mapping the data to a **higher dimensional space**



page 22

5.2 kernel function

However,

If we now transform to ϕ , instead of computing this dot product $(x_i \cdot x_j)$ we will have to compute $(\phi(x_i) \cdot \phi(x_j))$, which is quite expensive.

As a matter of fact, master come up with a extraordinary neat solution.

kernel function

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

the kernel function defines **inner products(similarity)** in the transformed space. Then we do not need to compute $(\phi(x_i) \cdot \phi(x_j))$ at all.

Here are some kernel functions:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$

$$K(\mathbf{x}, \mathbf{y}) = \exp \left\{ -\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2} \right\}$$

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x} \cdot \mathbf{y} - \delta)$$

1st is polynomial (includes $\mathbf{x} \cdot \mathbf{x}$ as special case)

2nd is radial basis function (gaussians)

3rd is sigmoid (neural net activation function)

page 24

6. supplement: inner product

inner product provide some measure of **similarity**.

In 2 dimension,

We all know that

- if two lines are **parallel** (**similar**) , there inner product equal to 1.
- if two lines are **perpendicular** (**un-similar**) , there inner product equal to 0.

Insight into inner products

Consider that we are trying to maximize the form:

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

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

The claim is that this function will be maximized 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 dissimilar, their dot product is 0, and they don't contribute to L .

Case 2. If two features x_i, x_j are completely alike, their dot product is 0. should be 1 here There are 2 subcases.

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

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

that's all.