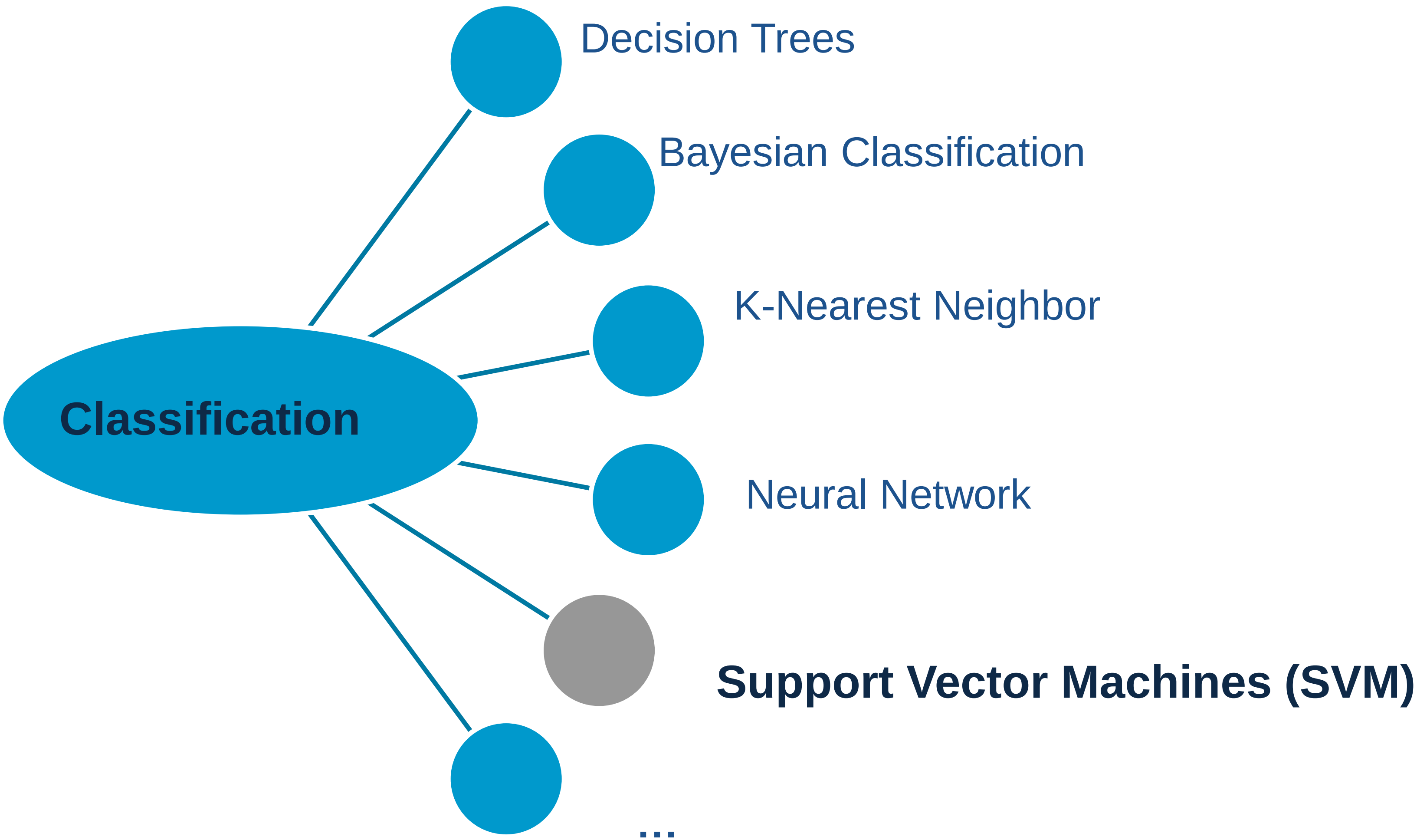
Classification
Part 4

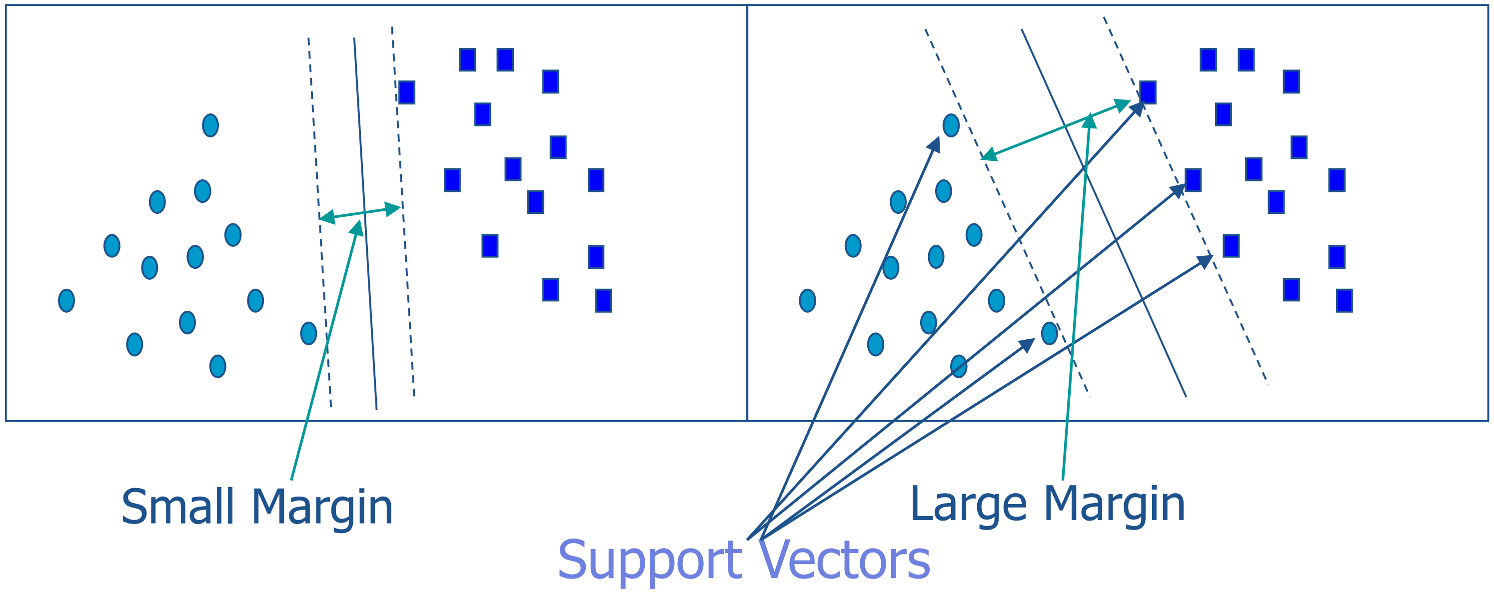
CME4416 – Introduction to Data Mining

Classification Techniques



Support Vector Machines (SVM)

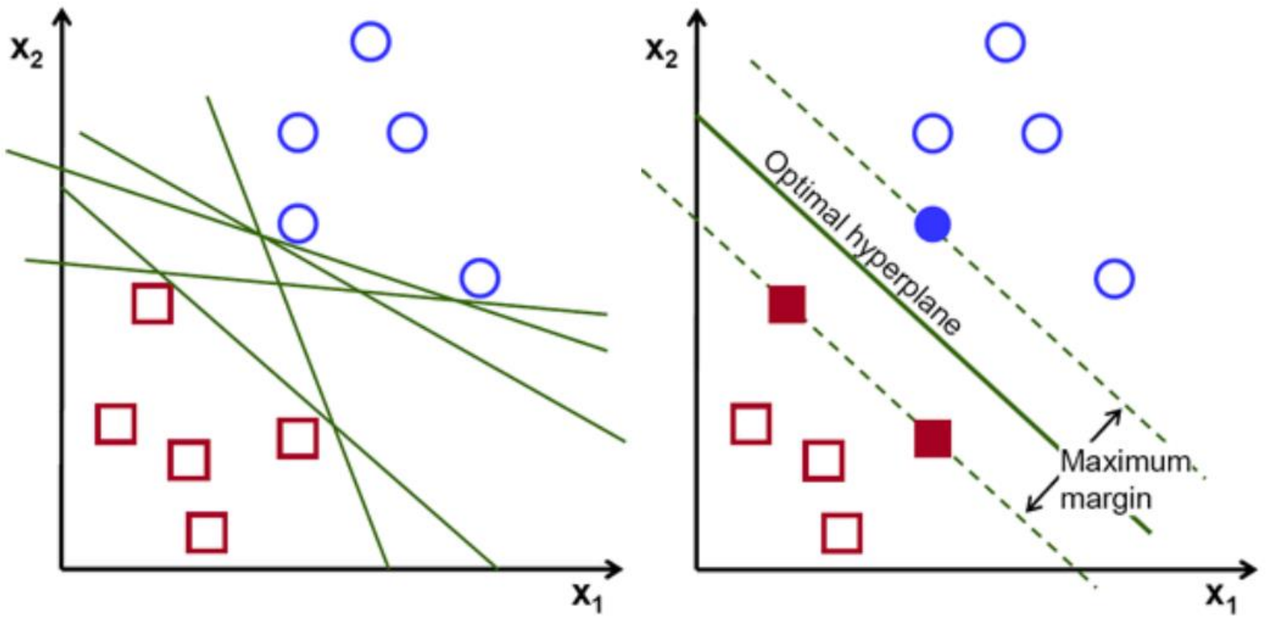
- It searches for the linear optimal separating hyperplane (i.e., “decision boundary”)
- SVM finds this hyperplane using support vectors (training tuples) and margins



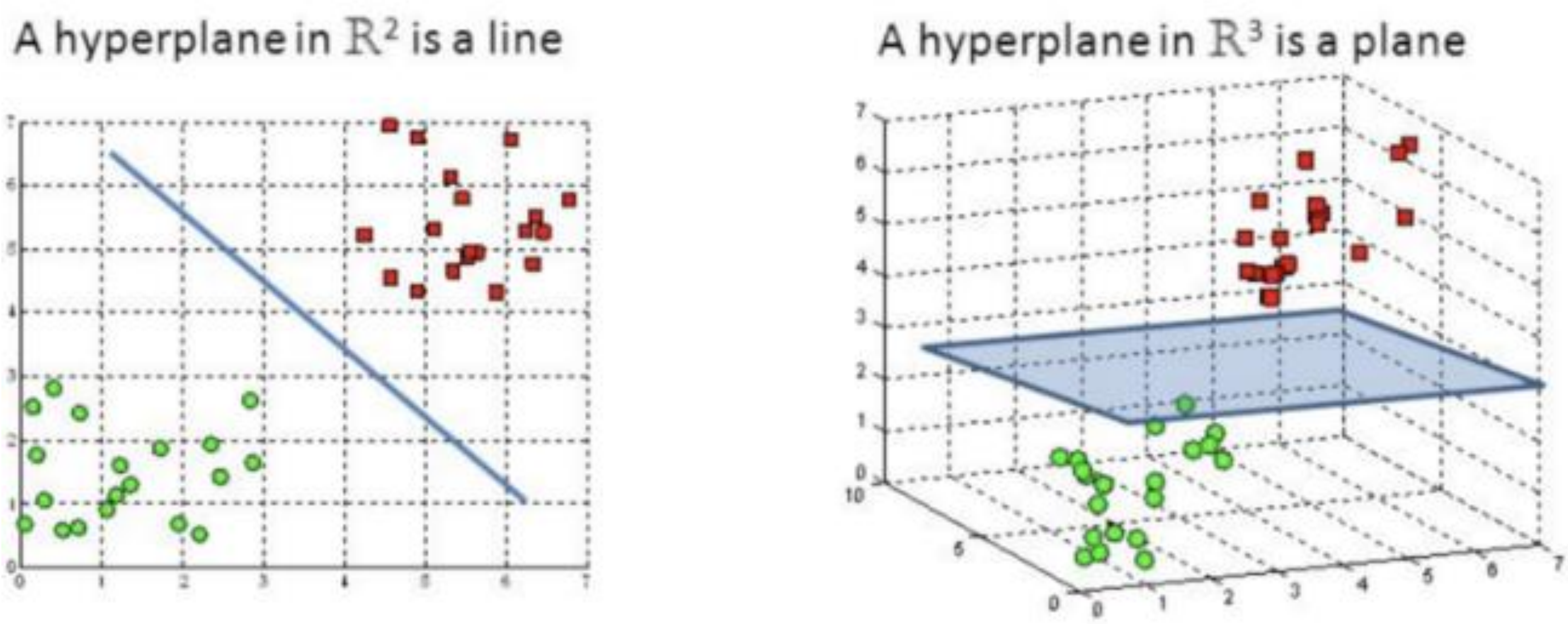
Support Vector Machines (SVM)

- There are infinite lines (hyperplanes) separating the two classes but we want to find the best one
  (the one that minimizes classification error on unseen data)

- SVM searches for the hyperplane with the **largest margin**



Possible hyperplanes



Hyperplanes in 2D and 3D feature space

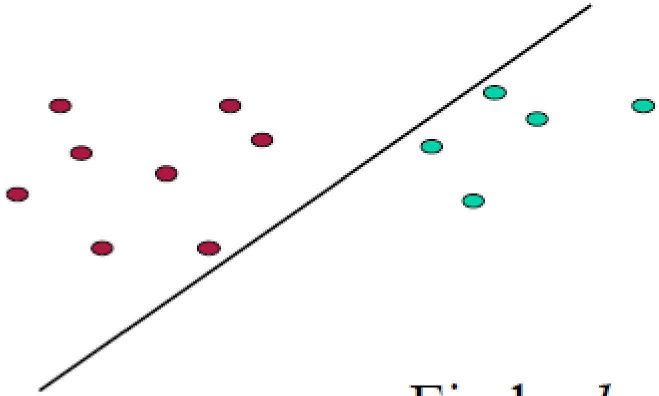
General input/output for SVMs just like for neural nets, but for one important addition…

Input: set of (input, output) training pair samples; call the input sample features $x_1, x_2 \ldots x_n$, and the output result y. Typically, there can be lots of input features x_i.

Output: set of weights $\mathbf{w}$ (or w_i), one for each feature, whose linear combination predicts the value of y. (So far, just like neural nets…)

Important difference: we use the optimization of maximizing the margin ('street width') to reduce the number of weights that are nonzero to just a few that correspond to the important features that 'matter' in deciding the separating line(hyperplane)…these nonzero weights correspond to the support vectors (because they 'support' the separating hyperplane)

2-D Case



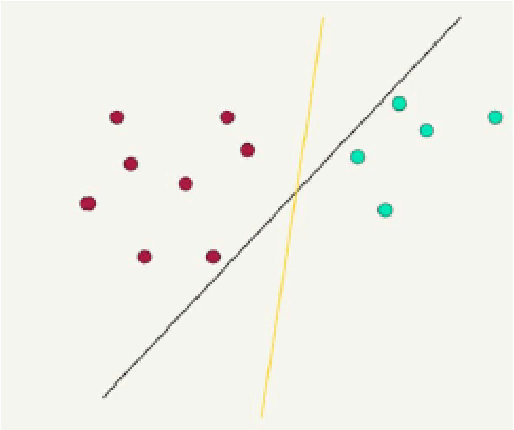
Find a, b, c, such that

$ax + by \geq c$ for red points

$ax + by \leq$ (or $<$) c for green points.

<u>Which</u> Hyperplane to pick?

- Lots of possible solutions for a,b,c.
- Some methods find a separating hyperplane, but not the optimal one (e.g., neural net)
- But: <u>Which</u> points should influence optimality?
  - All points?
    - Linear regression
    - Neural nets
  - Or only "difficult points" close to decision boundary
    - Support vector machines

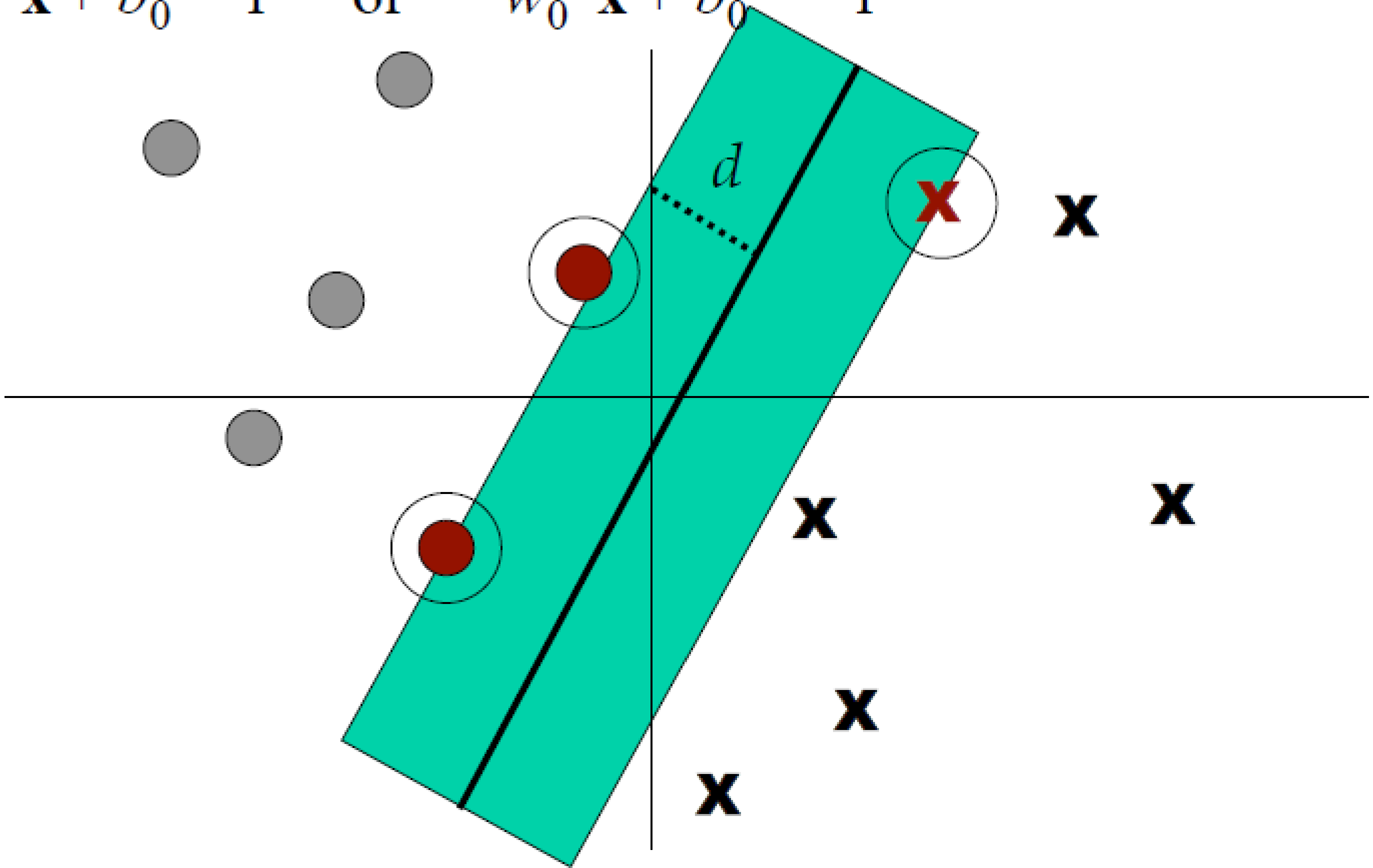


Support Vectors again for linearly separable case

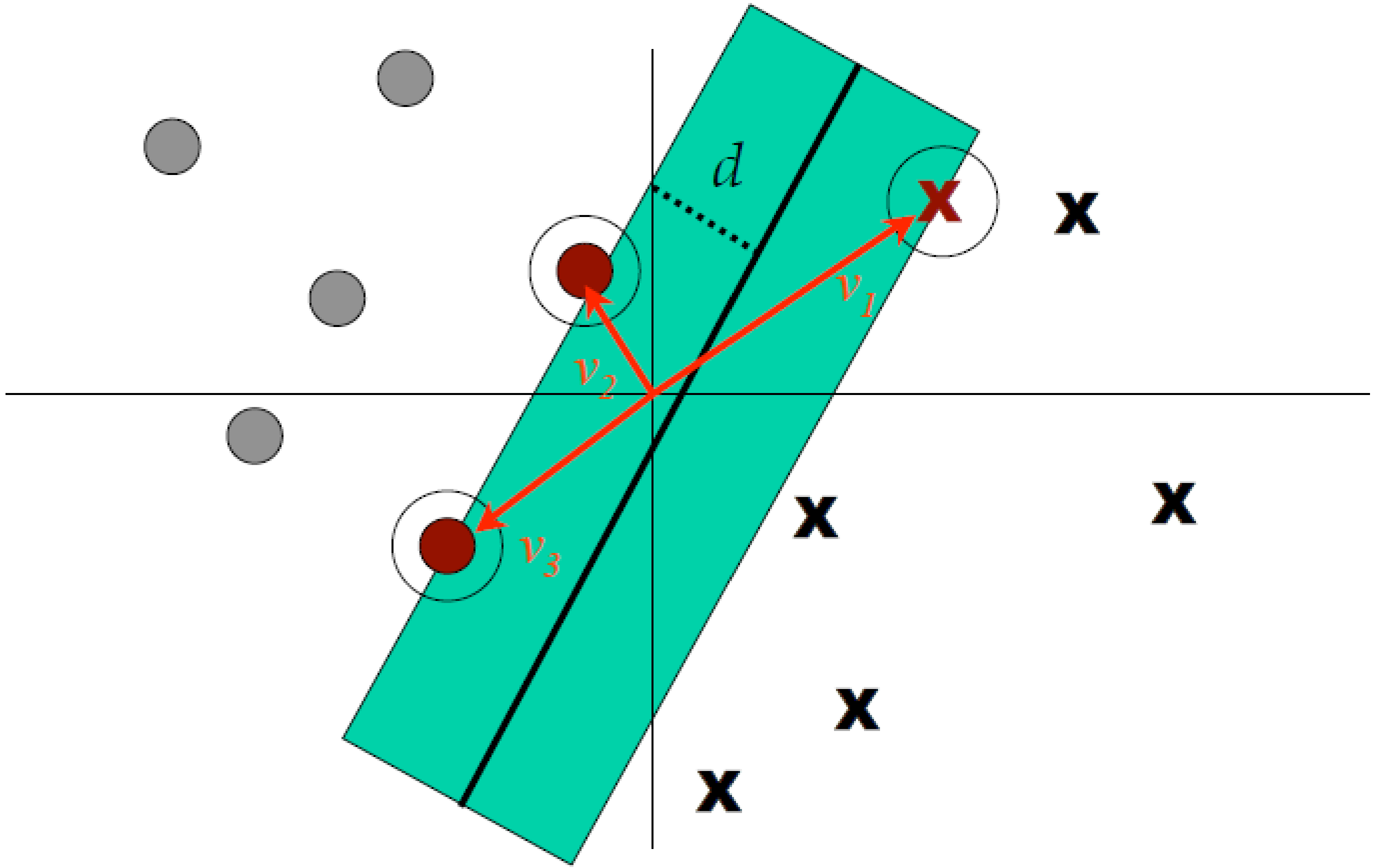
- Support vectors are the elements of the training set that would change the position of the dividing hyperplane if removed.
- Support vectors are the critical elements of the training set
- The problem of finding the optimal hyper plane is an optimization problem and can be solved by optimization techniques (we use Lagrange multipliers to get this problem into a form that can be solved analytically).

Support Vectors: Input vectors that just touch the boundary of the margin (street) – circled below, there are 3 of them (or, rather, the 'tips' of the vectors

$w_0^T\mathbf{x} + b_0 = 1$ or $w_0^T\mathbf{x} + b_0 = -1$



Here, we have shown the actual support vectors, v_1, v_2, v_3, instead of just the 3 circled points at the tail ends of the support vectors. d denotes 1/2 of the street 'width'

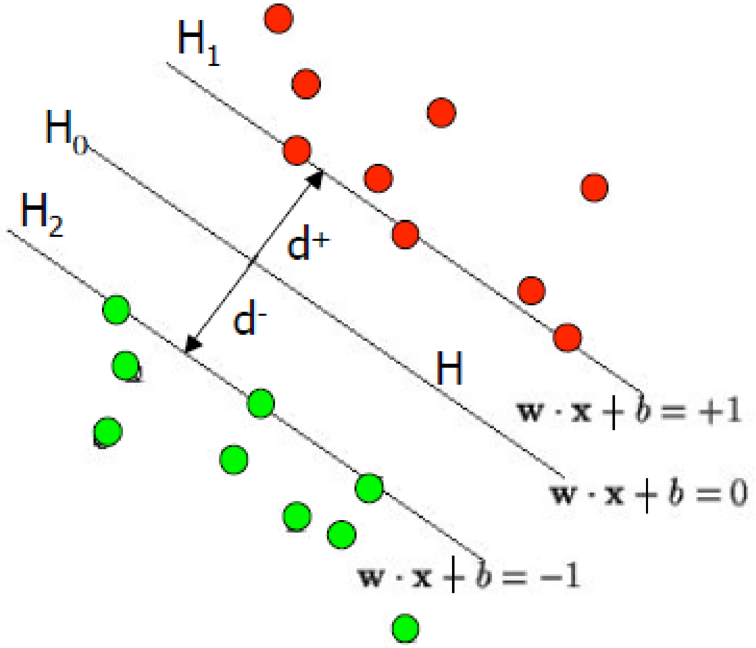


Definitions

Define the hyperplanes H such that:

$w \bullet x_i + b \geq +1$ when $y_i = +1$

$w \bullet x_i + b \leq -1$ when $y_i = -1$



H_1 and H_2 are the planes:

H_1: $w \bullet x_i + b = +1$

H_2: $w \bullet x_i + b = -1$

The points on the planes H_1 and H_2 are the tips of the Support Vectors

The plane H_0 is the median in between, where $w \bullet x_i + b = 0$

d+ = the shortest distance to the closest positive point

d- = the shortest distance to the closest negative point

The margin (gutter) of a separating hyperplane is d+ + d–.

Defining the separating Hyperplane

- Form of equation defining the decision surface separating the classes is a hyperplane of the form:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

  - **w is a weight vector**
  - **x is input vector**
  - **b is bias**
- Allows us to write

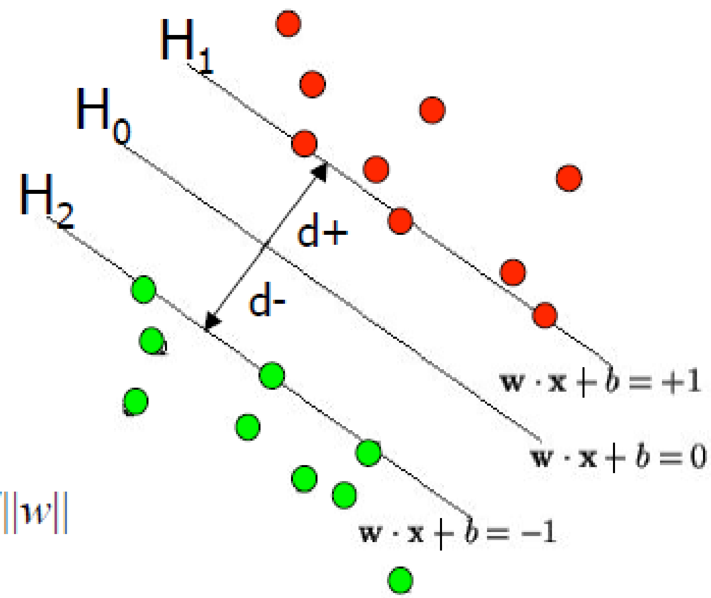
$$\mathbf{w}^T\mathbf{x} + b \geq 0 \text{ for } d_i = +1$$
$$\mathbf{w}^T\mathbf{x} + b < 0 \text{ for } d_i = -1$$

Some final definitions

- Margin of Separation (d): the separation between the hyperplane and the closest data point for a given weight vector **w** and bias b.
- Optimal Hyperplane (maximal margin): the particular hyperplane for which the margin of separation d is maximized.

Maximizing the margin (aka street width)

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



Recall the distance from a point(x_0,y_0) to a line:
$Ax+By+c = 0$ is: $|Ax_0 + By_0 + c|/\mathrm{sqrt}(A^2+B^2)$, 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 <u>maximize</u> the margin, we thus need to <u>minimize</u> $||w||$. <u>With the condition that there are no datapoints between H_1 and H_2:</u>

$\mathbf{x}_i \bullet \mathbf{w}+b \geq +1$ when $y_i = +1$
$\mathbf{x}_i \bullet \mathbf{w}+b \leq -1$ when $y_i = -1$

Can be combined into: $y_i(\mathbf{x}_i \bullet \mathbf{w}) \geq 1$

We now must solve a <u>quadratic</u> programming problem

- Problem is: <u>minimize</u> $||\mathbf{w}||$, **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 <u>quadratic</u> 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 <u>**constrained optimization problem**</u>

It can be solved by the Lagrangian multipler method

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

Two constraints

1. Parallel normal constraint (= gradient constraint on *f*, *g* s.t. solution is a max, or a min)
2. g(x)=0 (solution is on the constraint line as well)

We now recast these by combining *f*, *g* as the new Lagrangian function by introducing new 'slack variables' denoted *a* or (more usually, denoted α in the literature)

In general

Gradient min of f

constraint condition g

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

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

We expand the last to get the following L form:

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

Lagrangian Formulation

- So in the SVM problem the Lagrangian is

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

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:

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

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

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

The Lagrangian Dual Problem: instead of minimizing over **w**, b, subject to constraints involving a's, we can maximize over a (the dual variable) subject to 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. Most of the a's will turn out to have the value zero. The non-zero a's will correspond to the support vectors

Primal problem:

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

s.t. $\forall i \; a_i \geq 0$

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

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

Dual problem:

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

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

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:

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

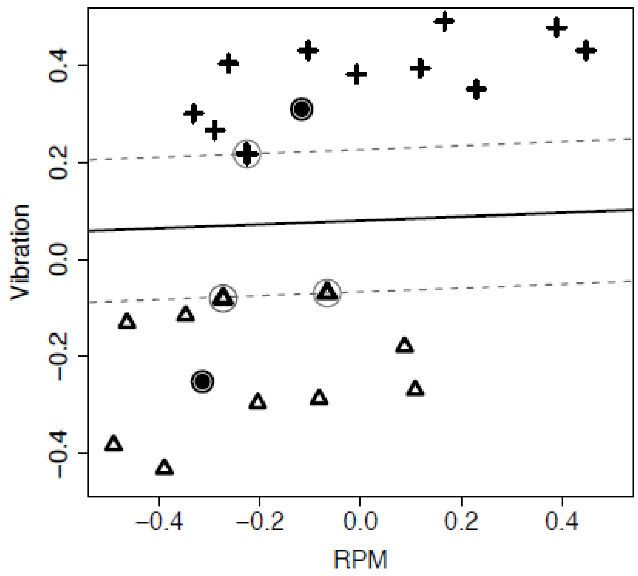
Remember: <u>most</u> of the weights $\mathbf{w_i}$, i.e., the a's, will be <u>zero</u>
Only the support vectors (on the gutters or margin) will have nonzero weights or a's – this reduces the dimensionality of the solution

Example

- The descriptive feature values and target feature values for the support vectors in these cases are
  - $(\langle -0.225, 0.217\rangle, +1)$,
  - $(\langle -0.066, -0.069\rangle, -1)$,
  - $(\langle -0.273, -0.080\rangle, -1)$.
- The value of b is -0.1838,
- The values of the α parameters are

$$\langle 23.056, 6.998, 16.058\rangle.$$

Example



- The plot shows the position of two new query instances for this problem.
- The descriptive feature values for these querys are
  1. $\mathbf{q}_1 = \langle -0.314, -0.251 \rangle$
  2. $\mathbf{q}_2 = \langle -0.117, 0.31 \rangle$.

Example

- For the first query instance, $\mathbf{q}_1 = \langle -0.314, -0.251 \rangle$, the output of the support vector machine model is:

$$
\begin{aligned}
&\mathbb{M}_{\boldsymbol{\alpha}, w_0}(\mathbf{q}_1) \\
&\quad = (1 \times 23.056 \times ((-0.225 \times -0.314) + (0.217 \times -0.251)) - 0.1838) \\
&\qquad + (-1 \times 6.998 \times ((-0.066 \times -0.314) + (-0.069 \times -0.251)) - 0.1838) \\
&\qquad + (-1 \times 16.058 \times ((-0.273 \times -0.314) + (-0.08 \times -0.251)) - 0.1838) \\
&\quad = -2.145
\end{aligned}
$$

- The model output is less than -1, so this query is predicted to be a *'faulty'* generator.
- For the second query instance, the model output is 1.592, so this instance is predicted to be a *'good'* generator.

Insight into inner products

Consider that we are trying to maximize the form:

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

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

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.

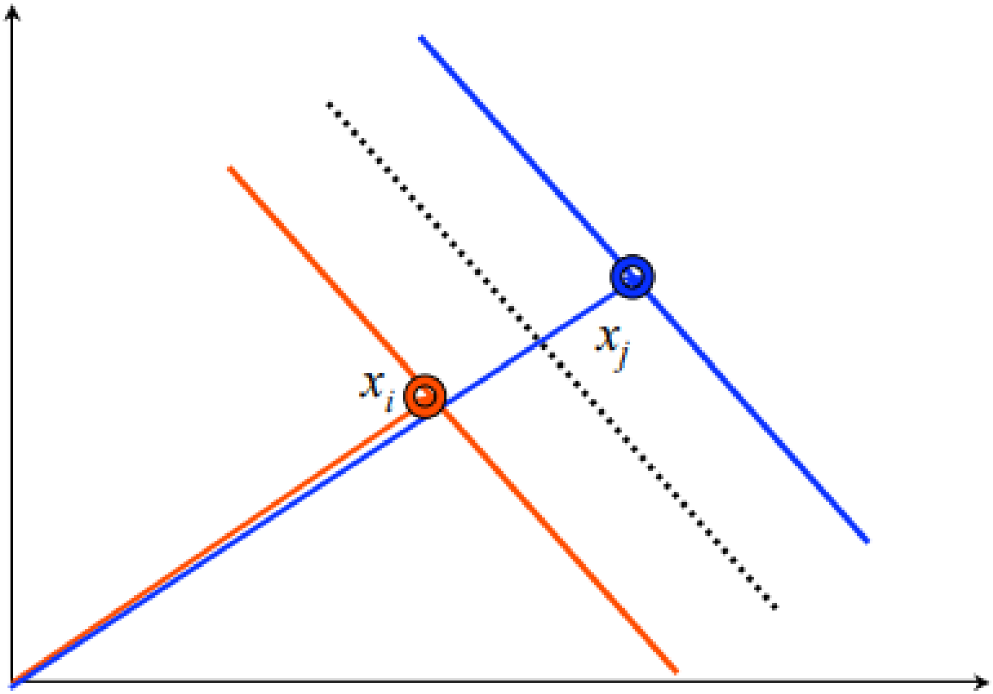
Case 1. If two features x_i, x_j are completely <u>dissimilar</u>, their dot product is 0, and they don't contribute to L.

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

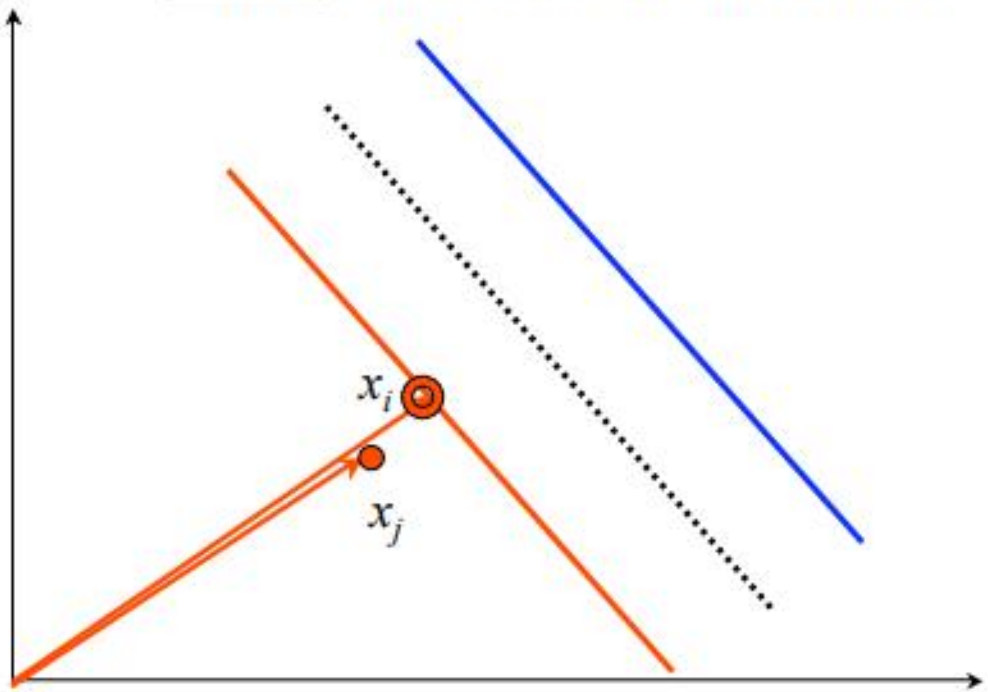
Subcase 1: both x_i and x_j predict the <u>same</u> output value y_i (either +1 or –1). Then y_i x y_j is always 1, and the value of $a_i a_j y_i y_j x_i x_j$ will be positive. But this would <u>decrease</u> the value of L (since it would subtract from the first term sum). So, the algorithm downgrades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

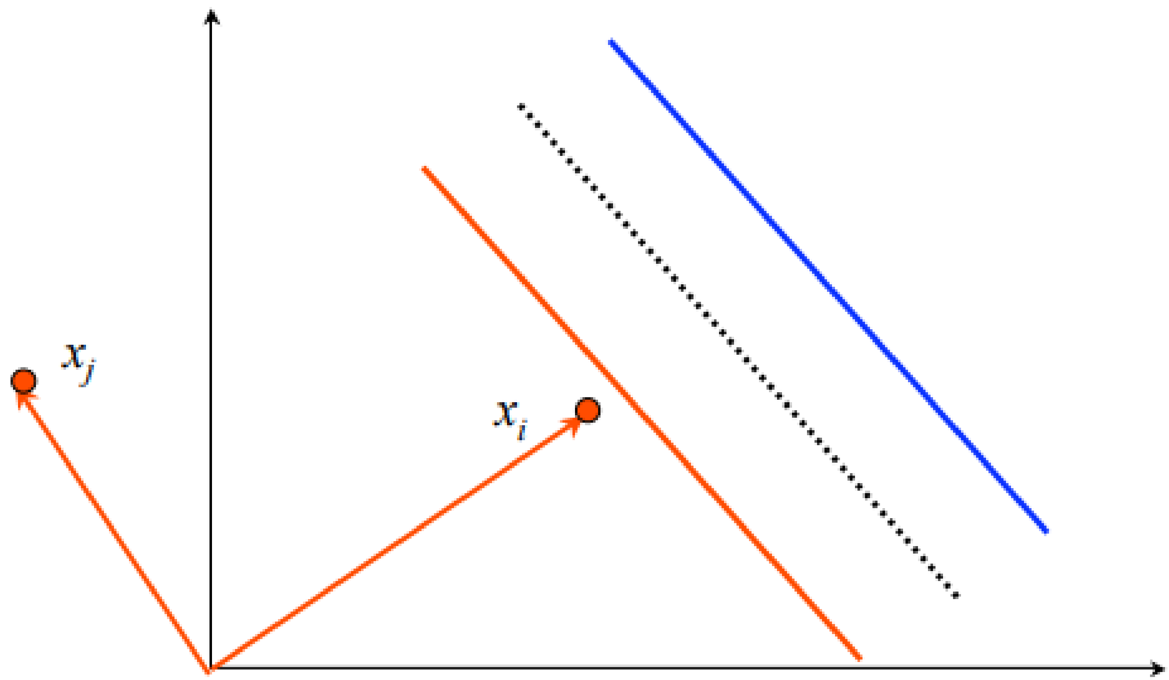
Insight into inner products, graphically: 2 very very similar x_i, x_j vectors that predict difft classes tend to <u>maximize</u> the margin width



2 vectors that are similar but predict the <u>same</u> class are redundant



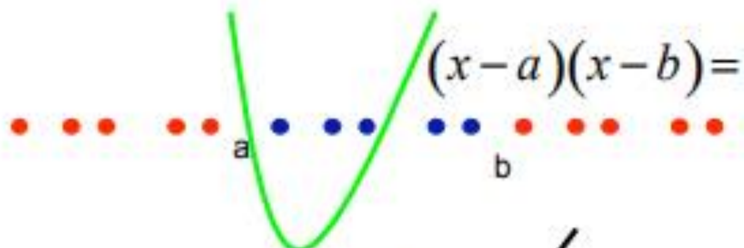
2 dissimilar (orthogonal) vectors don't count at all



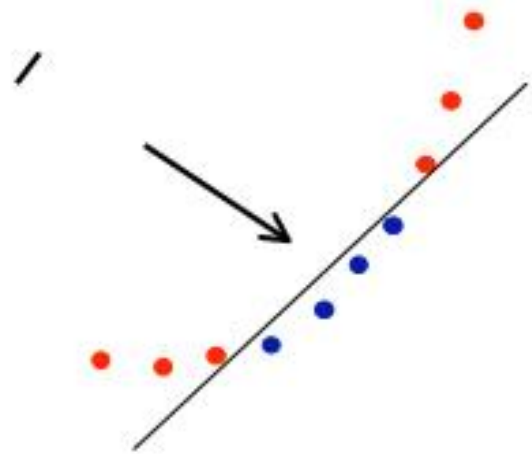
Non–Linear SVMs

- The idea is to gain linearly separation by mapping the data to a higher dimensional space
  - The following set can't be separated by a linear function, but can be separated by a quadratic one

$$(x-a)(x-b)=x^2-(a+b)x+ab$$

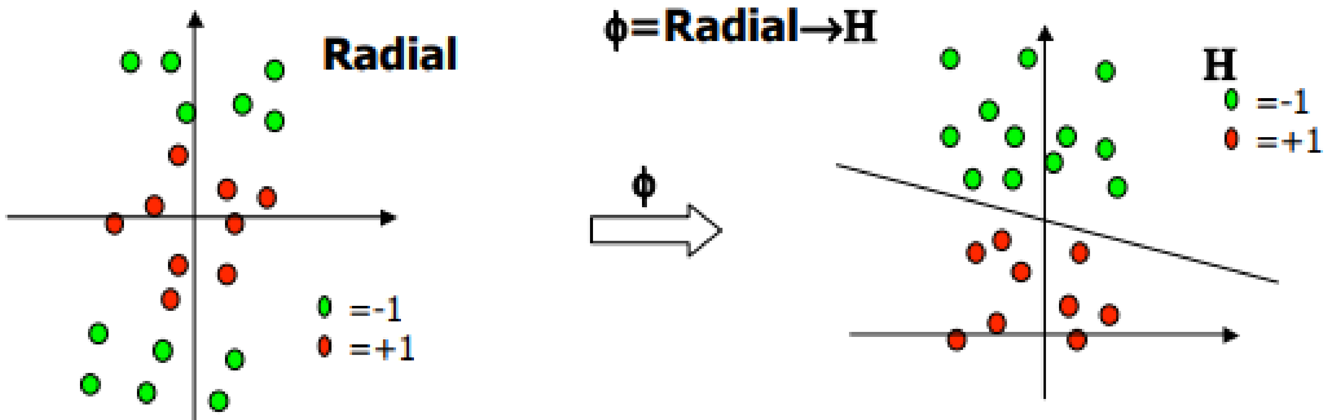


  - So if we map $x \mapsto \{x^2, x\}$ we gain linear separation



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: $L_d = \sum a_i - \frac{1}{2}\sum a_i a_j y_i y_j (x_i \bullet x_j)$ where $(x_i \bullet x_j)$ is the dot product of the two feature vectors. If we now transform to ϕ, instead of computing this dot product $(x_i \bullet x_j)$ we will have to compute $(\phi(x_i) \bullet \phi(x_j))$. 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 $K(x_i, x_j) = \phi(x_i) \bullet \phi(x_j)$, 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:

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

Kernel example: The polynomial kernel

$K(xi,xj) = (x_i \bullet x_j + 1)^p$, 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

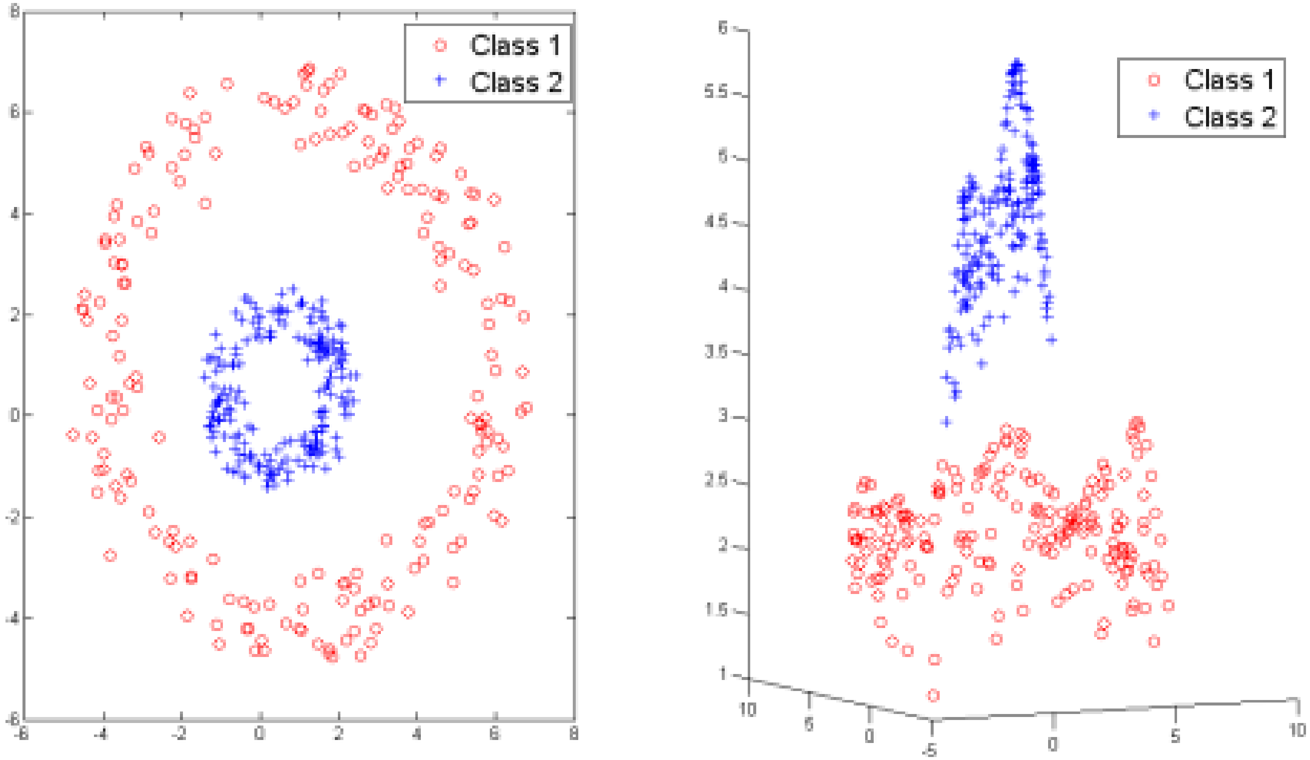
Examples for Non Linear SVMs

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

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

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

1st is polynomial (includes x•x as special case)
2nd is radial basis function (gaussians)
3rd is sigmoid (neural net activation function)



: *Dichotomous data re-mapped using Radial Basis Kernel*

Acknowledgements

- Prof. Bob Berwick – MIT
- http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf
- Lab:
- https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/