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

R. Berwick, Village Idiot

SVMs: A New Generation of Learning Algorithms

- Pre 1980:
  - Almost all learning methods learned linear decision surfaces.
  - Linear learning methods have nice theoretical properties
- 1980's
  - Decision trees and NNs allowed efficient learning of non-linear decision surfaces
  - Little theoretical basis and all suffer from local minima
- 1990's
  - Efficient learning algorithms for non-linear functions based on computational learning theory developed
  - Nice theoretical properties.

Key Ideas

- Two independent developments within last decade
  - Computational learning theory
  - New efficient separability of non-linear functions that use "kernel functions"
- The resulting learning algorithm is an optimization algorithm rather than a greedy search.

Statistical Learning Theory

- Systems can be mathematically described as a system that
  - Receives data (observations) as input and
  - Outputs a function that can be used to predict some features of future data.
- Statistical learning theory models this as a function estimation problem
- Generalization Performance (accuracy in labeling test data) is measured

Organization

- Basic idea of support vector machines
  - Optimal hyperplane for linearly separable patterns
  - Extend to patterns that are not linearly separable by transformations of original data to map into new space – Kernel function
- SVM algorithm for pattern recognition

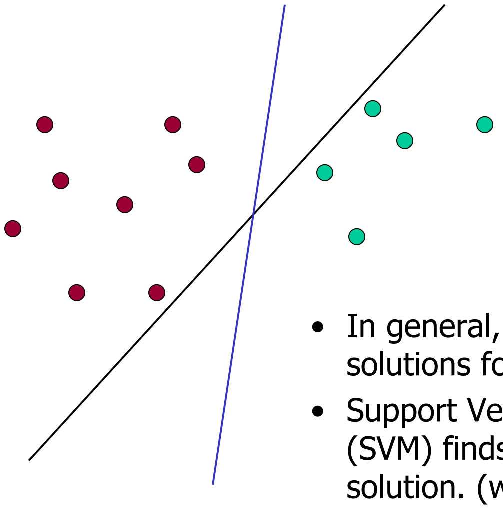
Unique Features of SVM's and Kernel Methods

- Are explicitly based on a theoretical model of learning
- Come with theoretical guarantees about their performance
- Have a modular design that allows one to separately implement and design their components
- Are not affected by local minima
- Do not suffer from the curse of dimensionality

Support Vectors

- Support vectors are the data points that lie closest to the decision surface
- They are the most difficult to classify
- They have direct bearing on the optimum location of the decision surface
- We can show that the optimal hyperplane stems from the function class with the lowest “capacity” (VC dimension).

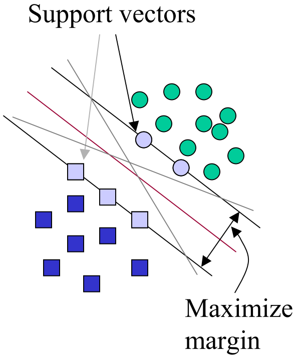
Recall: Which Hyperplane?



- In general, lots of possible solutions for *a,b,c*.
- Support Vector Machine (SVM) finds an <u>optimal</u> solution. (wrt what cost?)

Support Vector Machine (SVM)

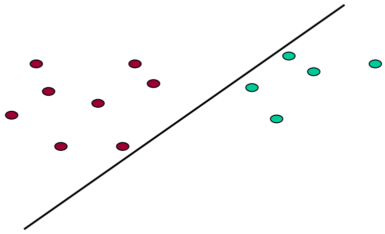
- SVMs maximize the *margin* around the separating hyperplane.
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- *Quadratic programming* problem
- Text classification method du jour



Separation by Hyperplanes

- Assume *linear separability* for now:
  - in 2 dimensions, can separate by a line
  - in higher dimensions, need hyperplanes
- Can find separating hyperplane by *linear programming* (e.g. perceptron):
  - separator can be expressed as $ax + by = c$

Linear Programming / Perceptron

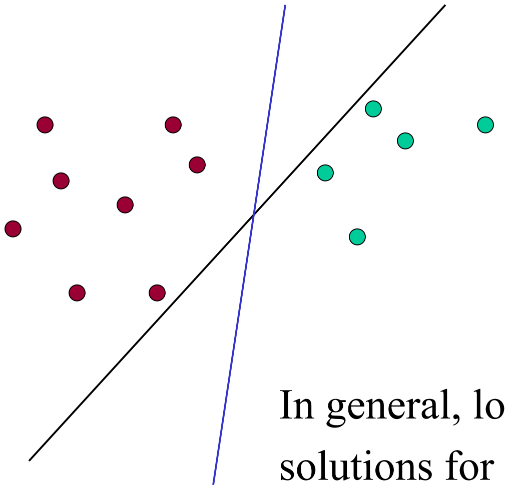


Find a,b,c, such that

$ax + by \geq c$ for red points

$ax + by \leq c$ for green points.

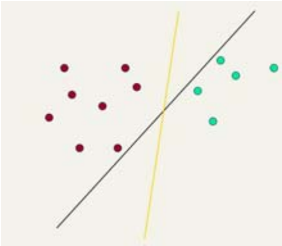
Which Hyperplane?



In general, lots of possible solutions for *a,b,c.*

Which Hyperplane?

- Lots of possible solutions for *a,b,c.*
- Some methods find a separating hyperplane, but not the optimal one (e.g., perceptron)
- Most methods find an optimal separating hyperplane
- Which points should influence optimality?
  - All points
    - Linear regression
    - Naïve Bayes
  - Only "difficult points" close to decision boundary
    - Support vector machines
    - Logistic regression (kind of)

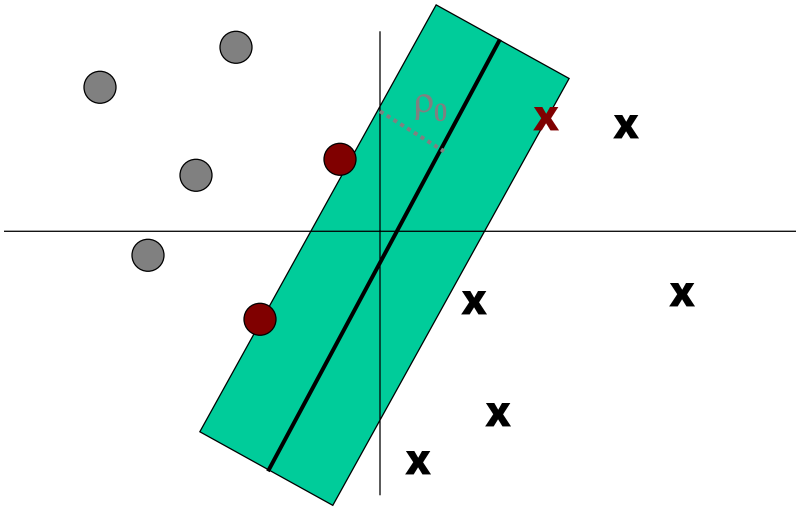


Support Vectors again for linearly separable case

- Support vectors are the elements of the training set that would change the position of the dividing hyper plane 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 (use Lagrange multipliers to get into a form that can be solved analytically).

Support Vectors: Input vectors for which

$w_0^T x + b_0 = 1$ or $w_0^T x + b_0 = -1$



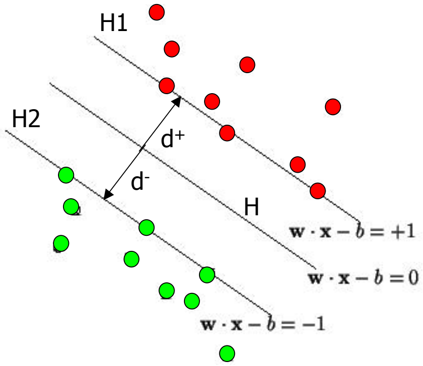
Definitions

Define the hyperplane H such that:

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

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

H1 and H2 are the planes:
H1: $\mathbf{x}_i \bullet \mathbf{w} + b = +1$
H2: $\mathbf{x}_i \bullet \mathbf{w} + b = -1$
The points on the planes
H1 and H2 are the
Support Vectors

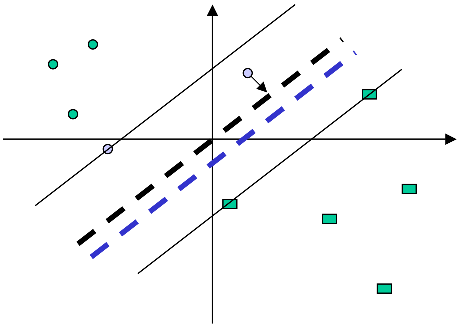


H1
H2
d^+
d^-
H
$\mathbf{w} \cdot \mathbf{x} - b = +1$
$\mathbf{w} \cdot \mathbf{x} - b = 0$
$\mathbf{w} \cdot \mathbf{x} - b = -1$

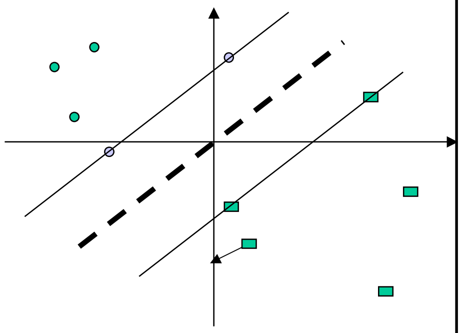
d+ = the shortest distance to the closest positive point

d- = the shortest distance to the closest negative point

The margin of a separating hyperplane is $d^+ + d^-$.



Moving a support vector moves the decision boundary

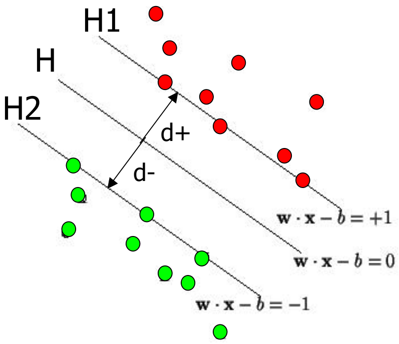


Moving the other vectors has no effect

The algorithm to generate the weights proceeds in such a way that only the support vectors determine the weights and thus the boundary

Maximizing the margin

We want a classifier with as big margin as possible.



Recall the distance from a point(x_0,y_0) to a line: Ax+By+c = 0 is|A x_0 +B y_0 +c|/sqrt(A^2+B^2)

The distance between H and H1 is: |**w•x**+b|/||w||=1/||w||

The distance between H1 and H2 is: 2/||w||

In order to maximize the margin, we need to minimize ||w||. With the condition that there are no datapoints between H1 and H2:

$\mathbf{x}_i \bullet \mathbf{w}$+b ≥ +1 when y_i =+1
$\mathbf{x}_i \bullet \mathbf{w}$+b ≤ -1 when y_i =-1 **Can be combined into $\mathbf{y_i(x_i \bullet w)} \geq \mathbf{1}$**

We now must solve a quadratic programming problem

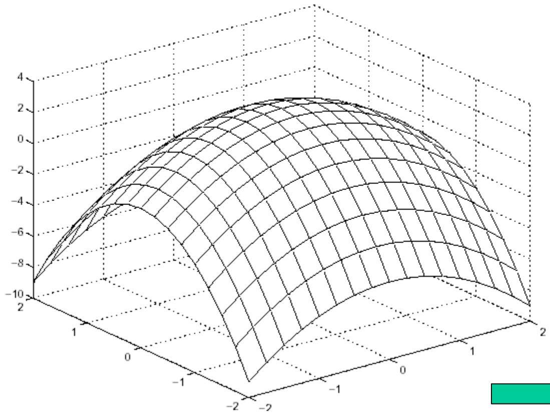
- Problem is: minimize **||w||, s.t.** discrimination boundary is obeyed, i.e., min $f(x)$ s.t. $g(x)=0$, where

  f: ½ $||w||^2$ and

  g: $\mathbf{y_i(x_i \bullet w)\text{-}b\ = 1}$ **or** $\mathbf{[y_i(x_i \bullet w)\text{-}b] - 1 = 0}$

This is a **constrained optimization problem**

Solved by Lagrangian multipler method

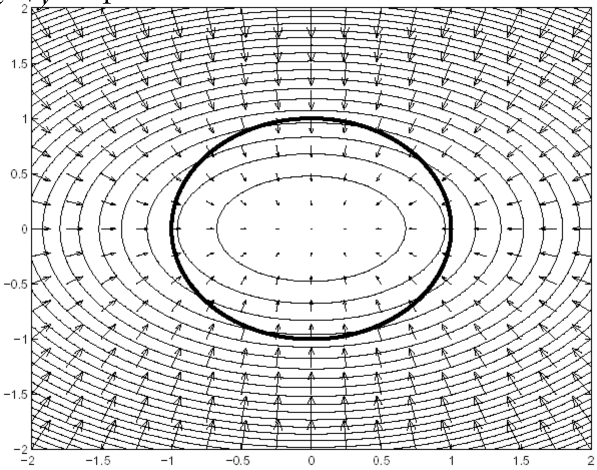


flatten

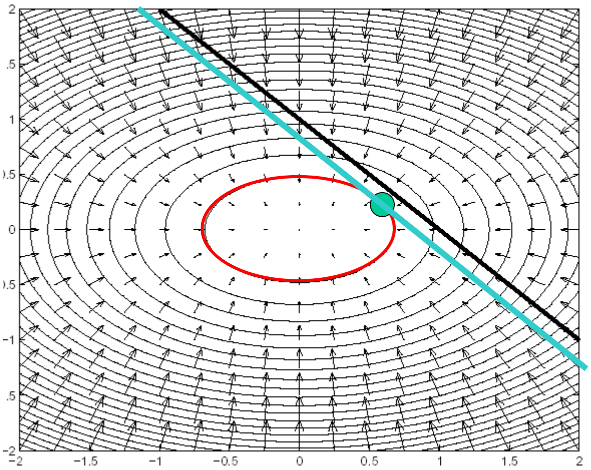
paraboloid 2-x^2-2y^2

Intuition: intersection of two functions at a tangent point.

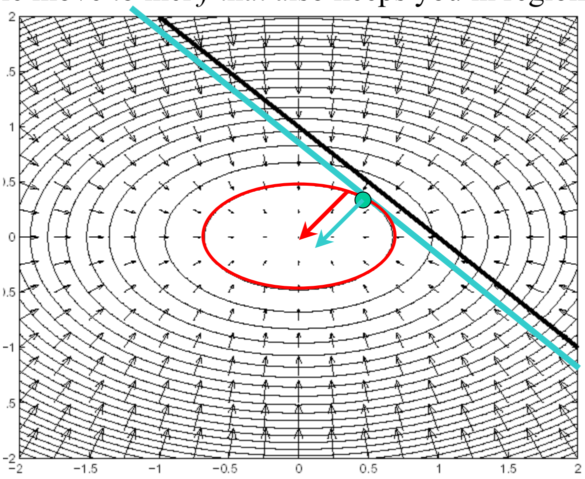
flattened paraboloid $2\text{-}x^2\text{-}2y^2$ *with* superimposed constraint $x^2 + y^2 = 1$



flattened paraboloid *f:* $2\text{-}x^2\text{-}2y^2 = 0$ *with* superimposed constraint *g:* $x + y = 1$



Maximize when the constraint line *g* is tangent to the inner ellipse contour line of *f* flattened paraboloid *f*: $2-x^2-2y^2=0$ *with* superimposed constraint *g*: $x+y = 1$; at tangent solution *p*, gradient vectors of *f,g* are parallel (no possible move to incr *f* that also keeps you in region *g*)



Maximize when the constraint line *g* is tangent to the inner ellipse contour line of *f*

Two constraints

1. Parallel normal constraint (= gradient constraint on *f, g* solution is a max)
2. G(x)=0 (solution is on the constraint line)

We now recast these by combining f, g as the Lagrangian

Redescribing these conditions

- Want to look for solution point *p* where

$$\nabla f(p) = \nabla \lambda g(p)$$
$$g(x) = 0$$

- Or, combining these two as the *Langrangian L* & requiring derivative of *L* be zero:

$$L(x,\lambda) = f(x) - \lambda g(x)$$
$$\nabla(x,\lambda) = 0$$

How Langrangian solves constrained optimization

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

Partial derivatives wrt *x* recover the parallel normal constraint

Partial derivatives wrt λ recover the *g*(*x,y*)=0

In general, $L(x,\lambda) = f(x) + \sum_i \lambda_i g_i(x)$

In general

Gradient max of *f*

constraint condition *g*

$L(x,\alpha) = f(x) + \sum_i \alpha_i g_i(x)$ a function of $n+m$ variables

n for the x's, m for the α. 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 α_i recover the constraints g_i

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

$$L = \tfrac{1}{2}\|\mathbf{w}\|^2 - \Sigma\alpha_i[y_i(\mathbf{w}.x_i + b)-1]$$

Lagrangian Formulation

- In the SVM problem the Lagrangian is

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

$$\alpha_i \geq 0, \forall i$$

- From the derivatives = 0 we get

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

The Lagrangian trick

Reformulate the optimization problem:
A ”trick” often used in optimization is to do an Lagrangian formulation of the problem.The constraints will be replaced by constraints on the Lagrangian multipliers and the training data will occur only as dot products.

Gives us the task:
Max $L = \sum\alpha_i - ½\sum\alpha_i\alpha_j x_i \bullet x_j$,
Subject to:
$\mathbf{w} = \sum\alpha_i y_i x_i$
$\sum\alpha_i y_i = 0$

What we need to see: $x_i$and x_j (input vectors) appear only in the form of dot product – we will soon see why that is important.

The Dual problem

- Original problem: fix value of *f* and find α
- New problem: Fix the values of α, and solve the (now unconstrained) problem max $L(\alpha, x)$
- Ie, get a solution for each α, $f^*(\alpha)$
- Now minimize this over the space of α
- Kuhn-Tucker theorem: this is equivalent to original problem

At a solution *p*

- The the constraint line *g* and the contour lines of *f* must be tangent
- If they are tangent, their gradient vectors (perpindiculars) are parallel
- Gradient of *g* must be 0 – I.e., steepest ascent & so perpendicular to *f*
- Gradient of *f* must also be in the same direction as *g*

Inner products

The task:
Max L = $\sum\alpha_i - \frac{1}{2}\sum\alpha_i\alpha_j x_i \bullet x_j$,
Subject to:

$\mathbf{w} = \sum\alpha_i y_i x_i$

$\sum\alpha_i y_i = 0$

Inner product

Inner products

Why should inner product kernels be involved in pattern recognition?

-- Intuition is that they provide some measure of similarity

-- cf Inner product in 2D between 2 vectors of unit length returns the cosine of the angle between them.

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

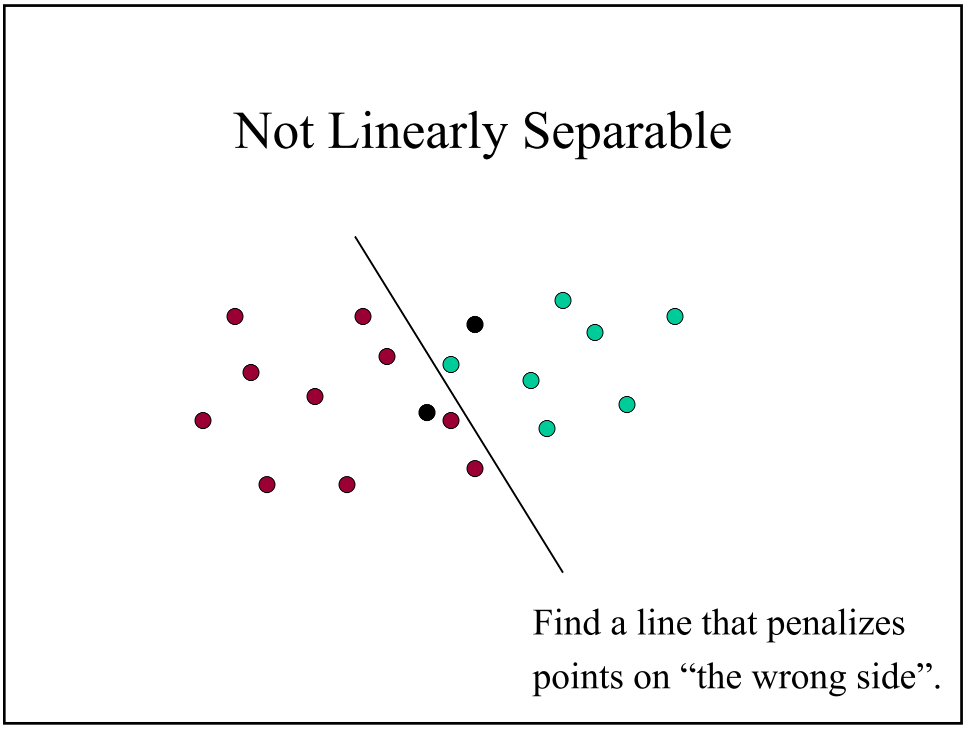
I.e. if they are parallel inner product is 1

$$\underline{x}^T \underline{x} = \underline{x}.\underline{x} = 1$$

If they are perpendicular inner product is 0

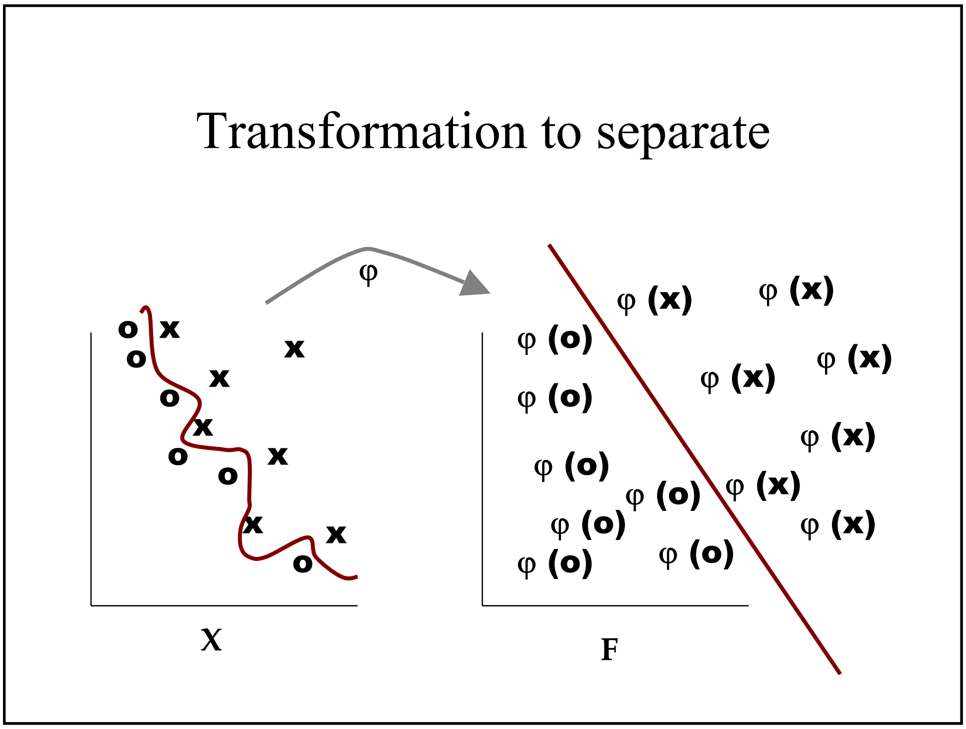
$$\underline{x}^T \underline{y} = \underline{x}.\underline{y} = 0$$

But…are we done???



Not Linearly Separable

Find a line that penalizes points on “the wrong side”.

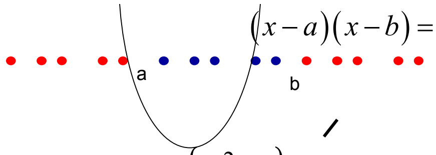


Transformation to separate

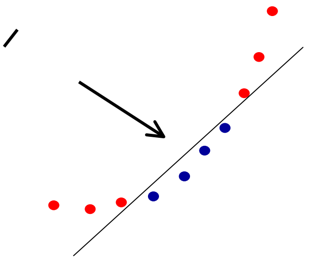
Non Linear SVMs

- 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

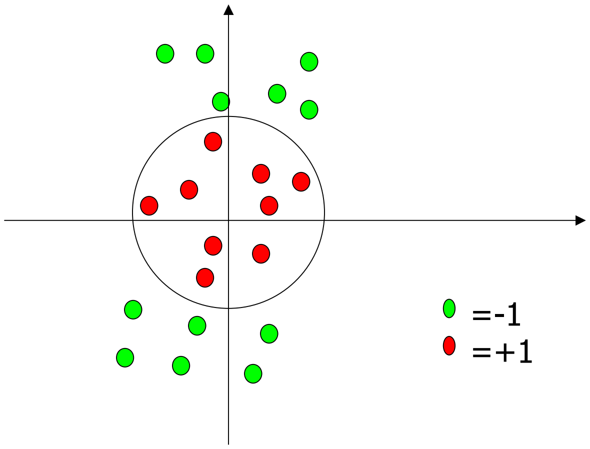
$$(x-a)(x-b)=x^2-(a+b)x+ab$$



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



Problems with linear SVM



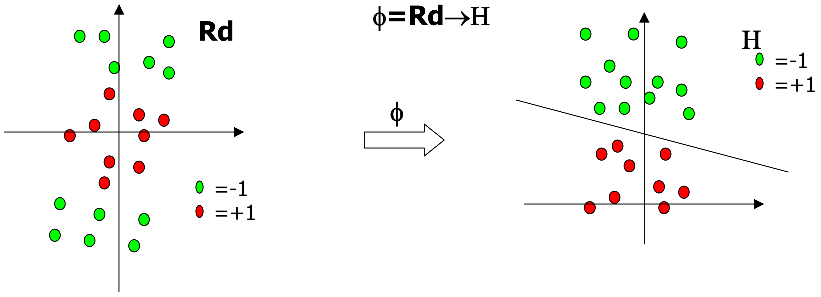
What if the decision function is not linear? What transform would separate these?

Ans: polar coordinates!
Non-linear SVM 1

The Kernel trick

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

ϕ=**Rd**$\rightarrow$H

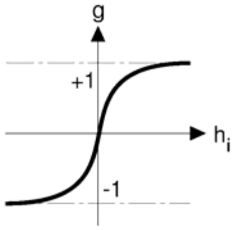


Remember the function we want to optimize: $L_{dual} = \sum\alpha_i - ½\sum\alpha_i\alpha_j x_i \bullet x_j$,
x_i and x_j as a dot product. We will have $\phi(x_i) \bullet \phi(x_j)$ in the non-linear case.
If there is a "kernel function" K such as K(xi,xj) = ϕ(xi) • ϕ(xj), we
do not need to know ϕ explicitly. One example:

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

We've already seen a nonlinear transform…

- What is it???
- $\tanh(\beta_0 x^T x_i + \beta_1)$



Examples for Non Linear SVMs

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

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

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

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

Inner Product Kernels

Type of Support Vector Machine	Inner Product Kernel $K(x,x_i)$, I = 1, 2, …, N	Comments
Polynomial learning machine	$(x^Tx_i + 1)^p$	Power p is specified apriori by the user
Radial-basis function network	$\exp(1/(2\sigma^2)\|\|x-x_i\|\|^2)$	The width σ^2 is specified apriori
Two layer perceptron	$\tanh(\beta_0x^Tx_i + \beta_1)$	Mercer's theorem is satisfied only for some values of β_0 and β_1

Non-linear svm2

The function we end up optimizing is:

Max Ld = $\sum\alpha_i - ½\sum\alpha_i\alpha_j K(xi \bullet x_j)$,

Subject to:

$\mathbf{w} = \sum\alpha_i y_i x_i$

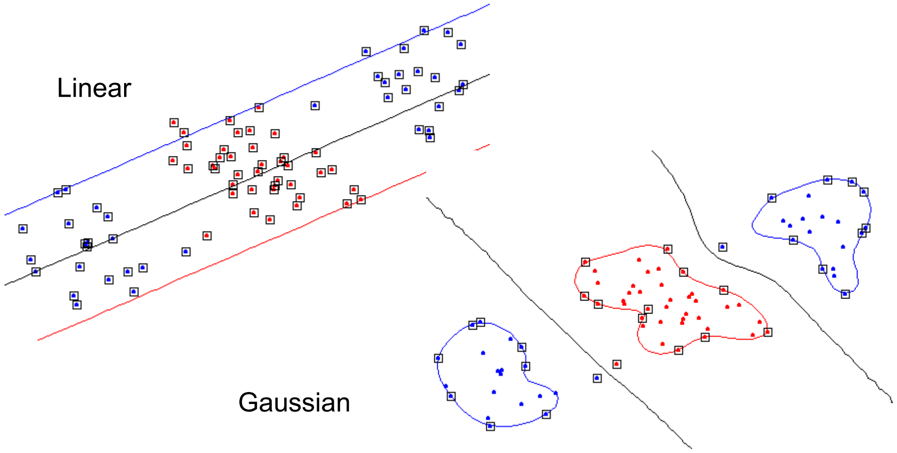
$\sum\alpha_i y_i = 0$

Another kernel example: The polynomial kernel

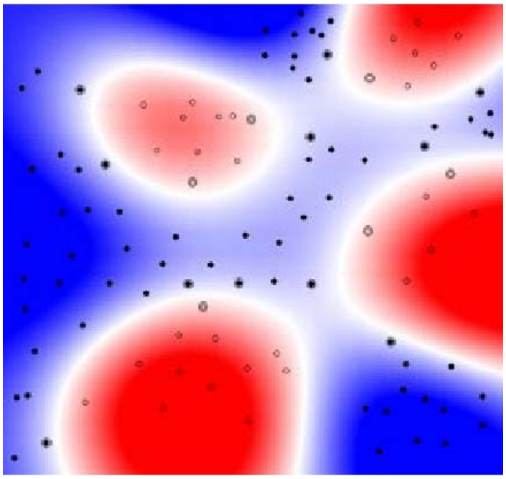
$K(xi,xj) = (xi \bullet xj + 1)^p$, where p is a tunable parameter.

Evaluating K only require one addition and one exponentiation more than the original dot product.

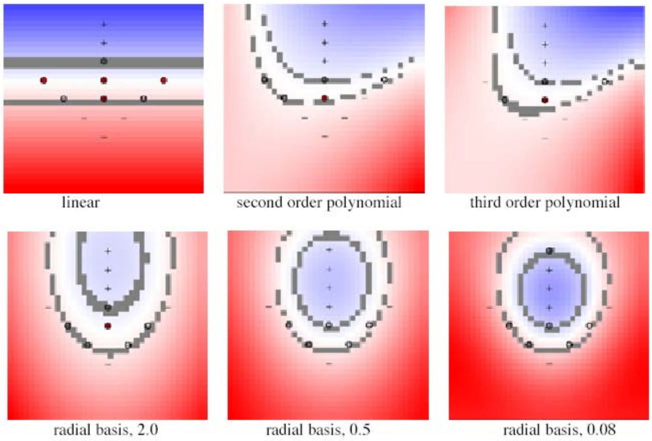
Examples for Non Linear SVMs 2 – Gaussian Kernel



Nonlinear rbf kernel



Admiral’s delight w/ diffft kernel functions



linear

second order polynomial

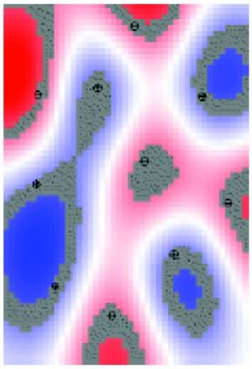
third order polynomial

radial basis, 2.0

radial basis, 0.5

radial basis, 0.08

Overfitting by SVM



Building an SVM Classifier

- Now we know how to build a separator for two linearly separable classes
- What about classes whose exemplary examples are not linearly separable?

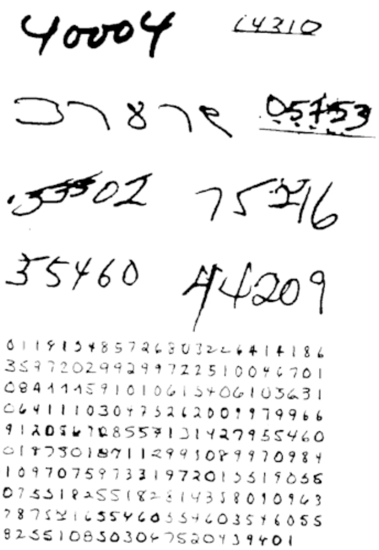


FIGURE 10.8
Examples of ZIP code image, and segmented and normalized numerals from the testing set. (*Source:* Reprinted with permission from Y. Le Cun, et al., "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, 1:541–551, 1989. ©1989 The MIT Press.)