Support Vector Machines

Xiaojin Zhu

jerryzhu@cs.wisc.edu

Computer Sciences Department

University of Wisconsin, Madison

[Based on slides from Andrew Moore http://www.cs.cmu.edu/~awm/tutorials]

The no-math verison

Support vector machines

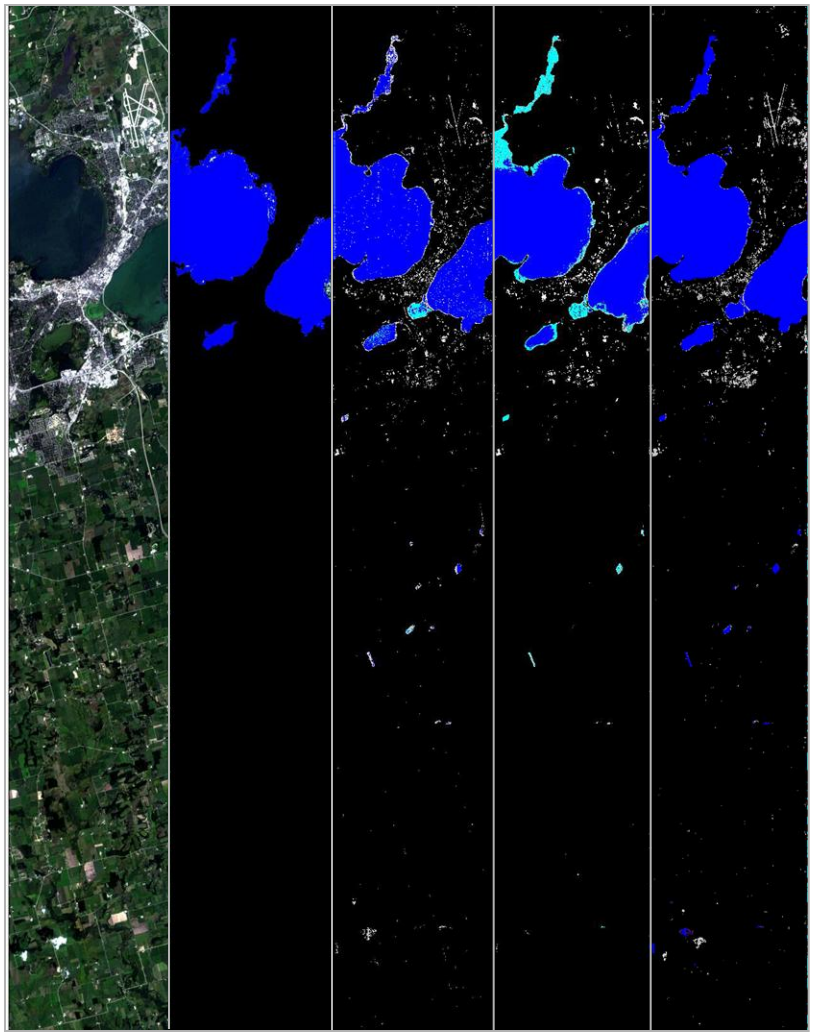
Courtesy of Steve Chien of NASA/JPL

Lake Mendota, Madison, WI

- Identify areas of land cover (land, ice, water, snow) in a scene
- Three algorithms:
  - Scientist manually derived
  - Automatic best ratio
  - Support Vector Machine (SVM)

Classifier	Expert Derived	Automated Ratio	SVM
cloud	45.7%	43.7%	58.5%
ice	60.1%	34.3%	80.4%
land	93.6%	94.7%	94.0%
snow	63.5%	90.4%	71.6%
water	84.2%	74.3%	89.1%
unclassified	45.7%		

Lake Mendota, Wisconsin



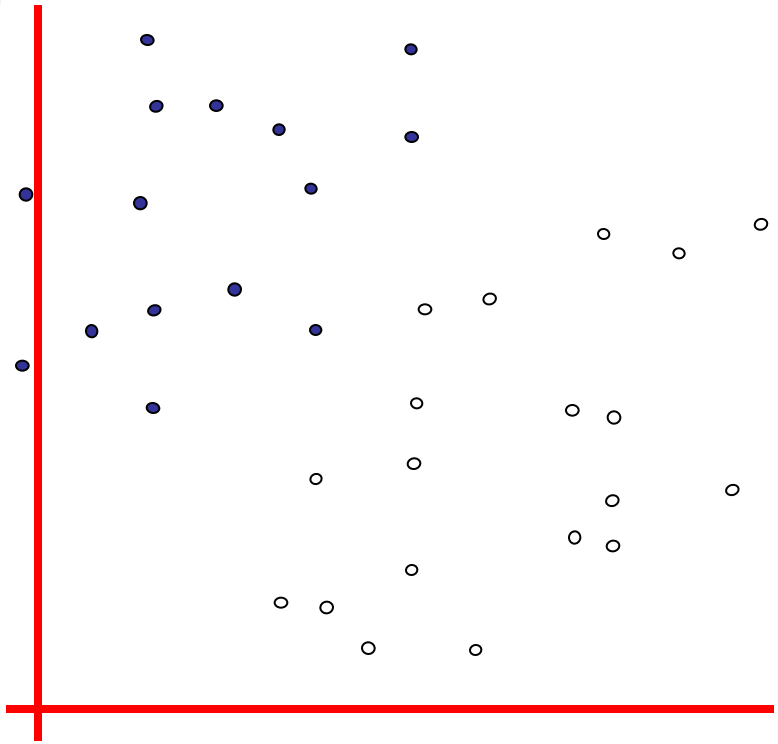
Visible Image | Expert Labeled | Expert Derived | Automated Ratio | SVM

Support vector machines

- The state-of-the-art classifier

Class labels

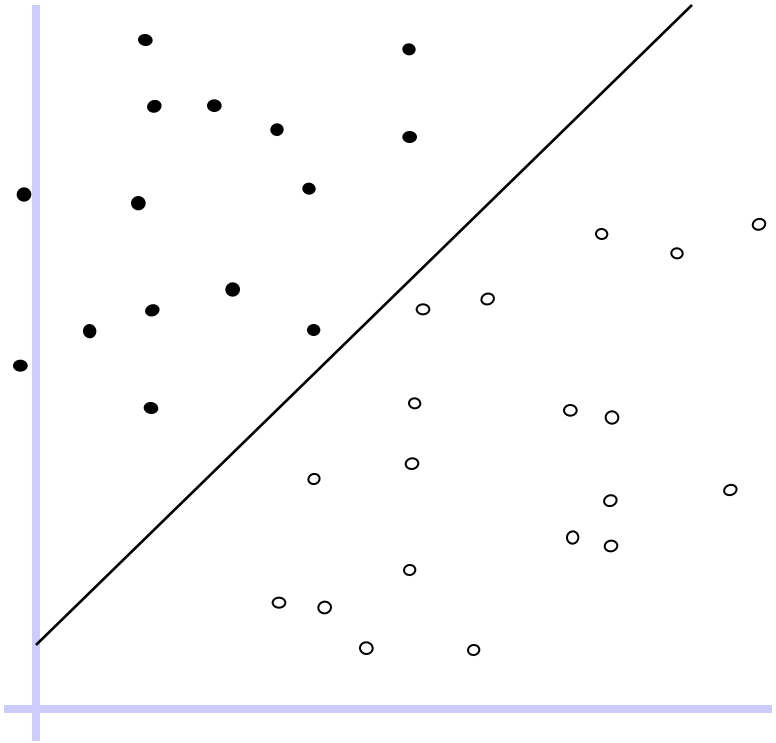
- denotes +1
- denotes -1



How would you classify this data?

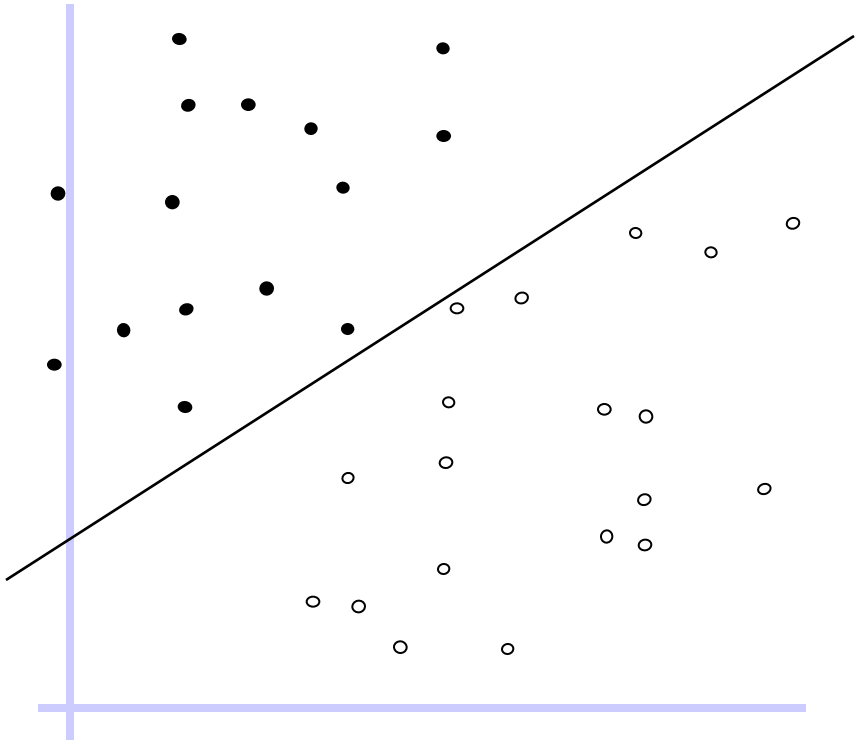
Linear classifier

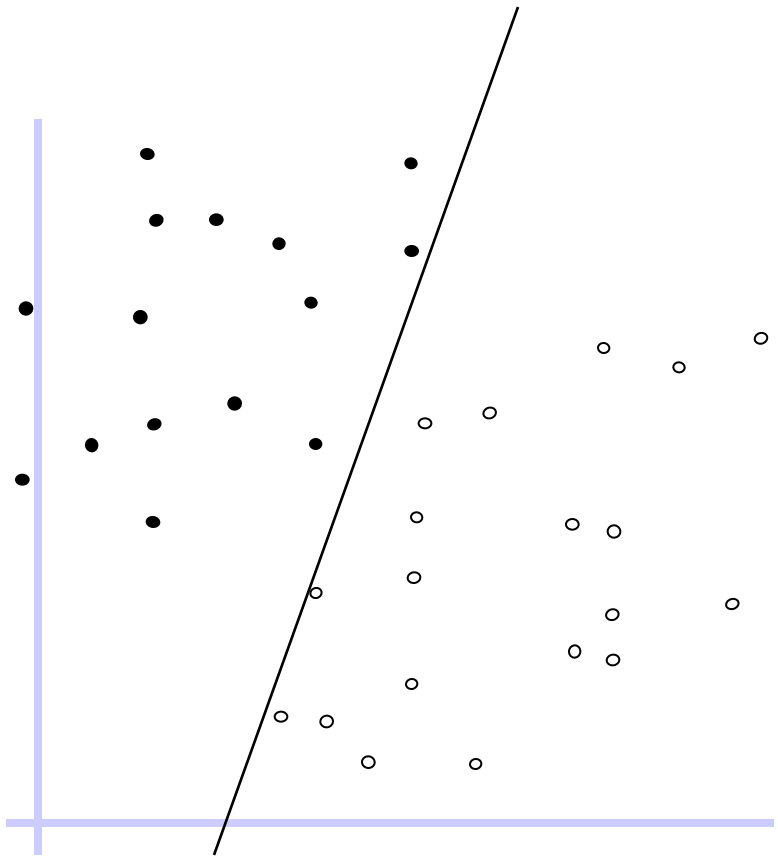
- If all you can do is to draw a straight line, the ‘linear decision boundary’…

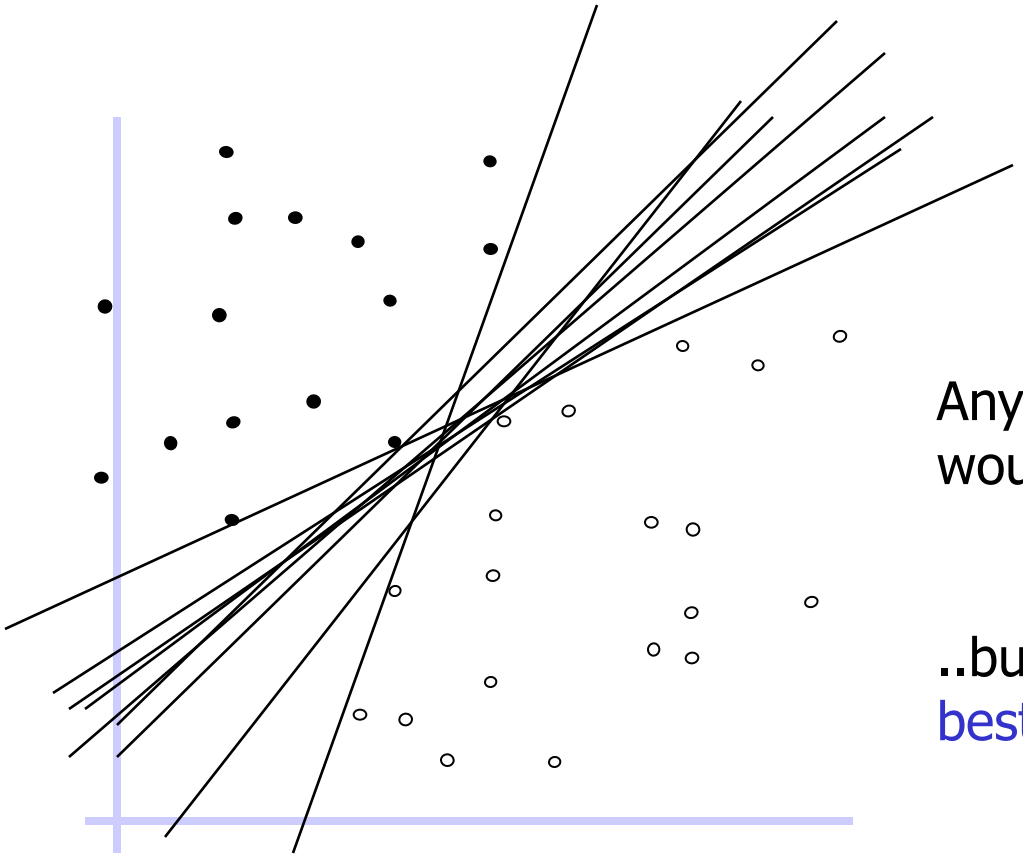


Linear classifier

- Another OK ‘decision boundary’





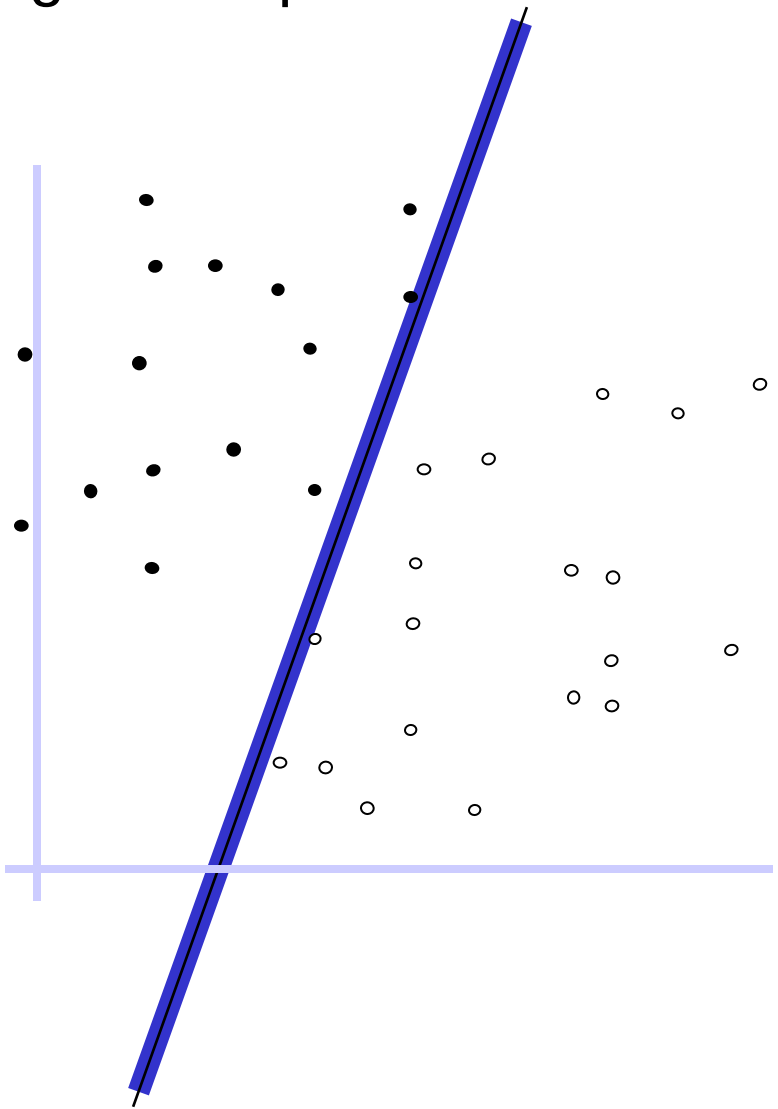


Any of these would be fine..

..but which is the best?

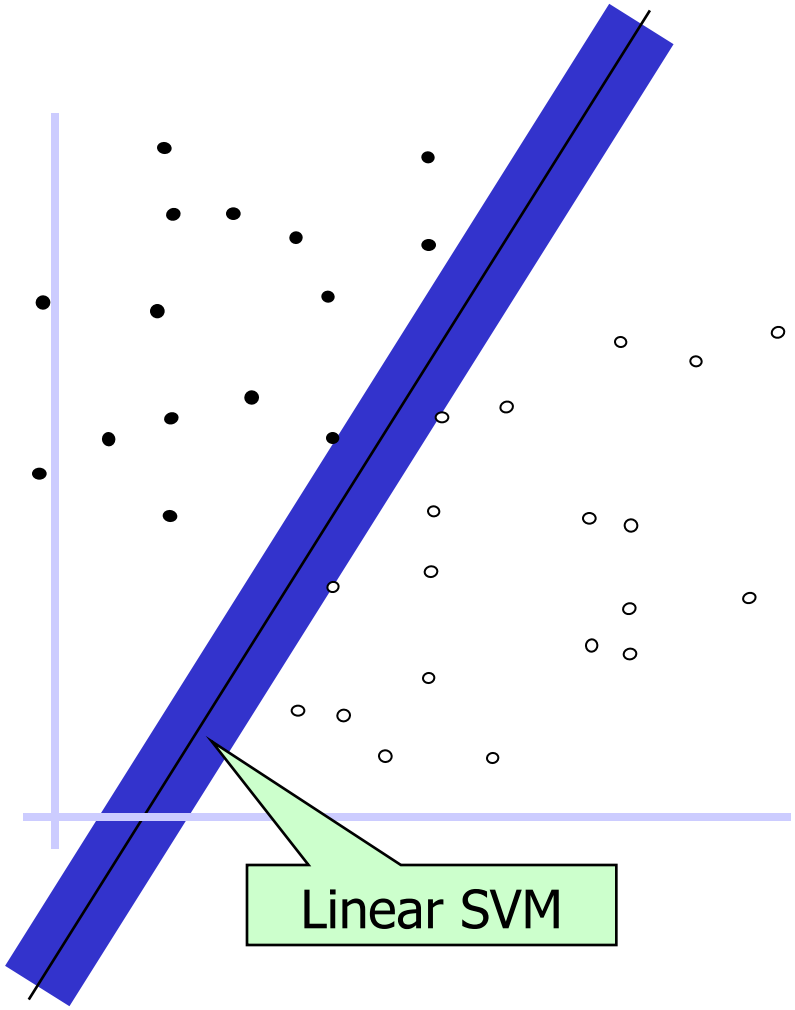
The margin

- Margin: the width that the boundary can be increased before hitting a data point



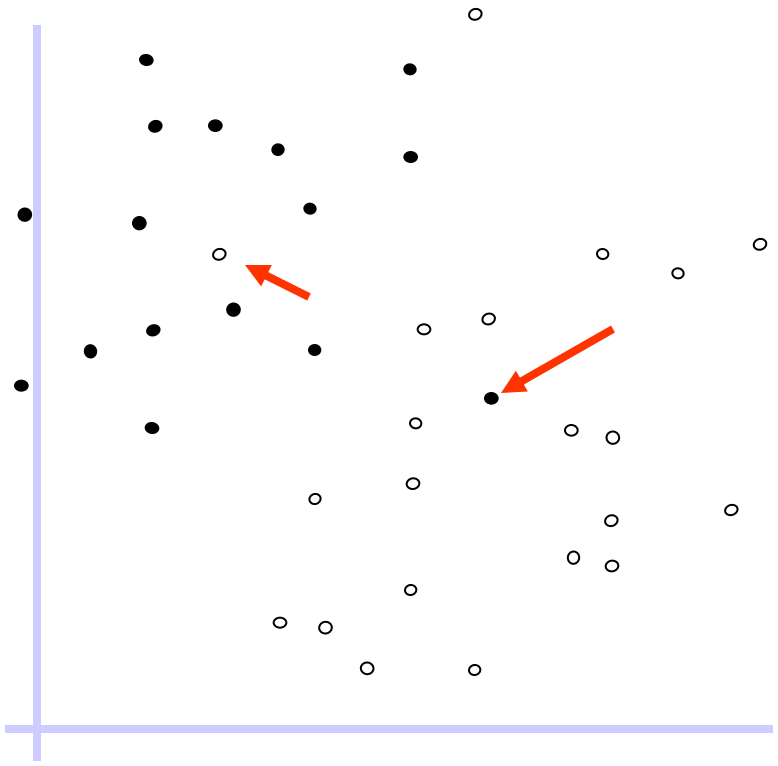
SVM: maximize margin

- The simplest SVM (linear SVM) is the linear classifier with the maximum margin



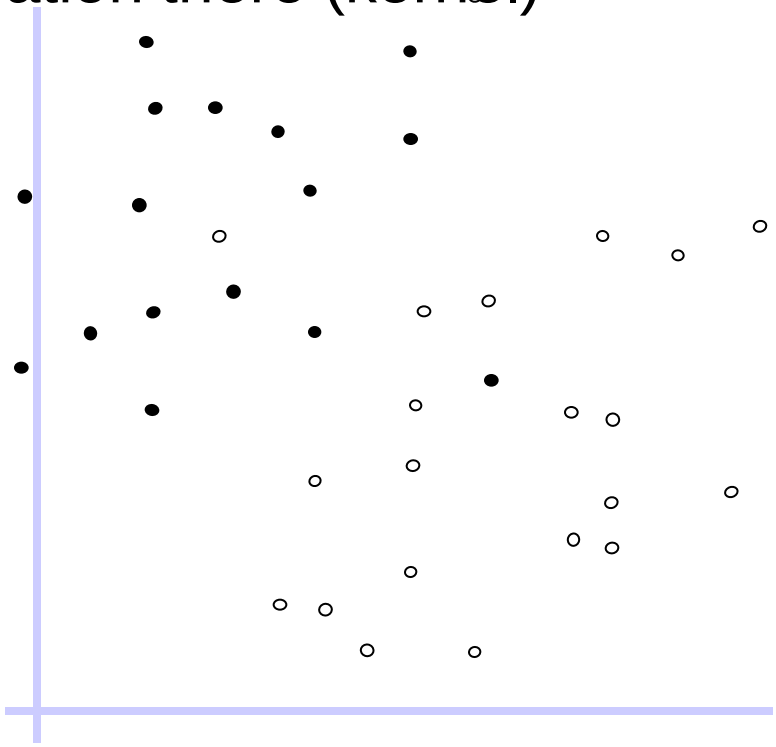
SVM: linearly non-separable data

- What if the data is not linearly separable?



SVM: linearly non-separable data

- Two solutions:
  - Allow a few points on the wrong side (slack variables), and/or
  - Map data to a higher dimensional space, do linear classification there (kernel)



SVM: more than two classes

- N class problem: Split the task into N binary tasks:
  - Class 1 vs. the rest (class 2—N)
  - Class 2 vs. the rest (class 1, 3—N)
  - …
  - Class N vs. the rest
- Finally, pick the class that put the point furthest into the positive region.

SVM: get your hands on it

- There are many implementations
- http://www.support-vector.net/software.html
- http://svmlight.joachims.org/

- You don't have to know the rest of the class in order to use SVM.

end of class

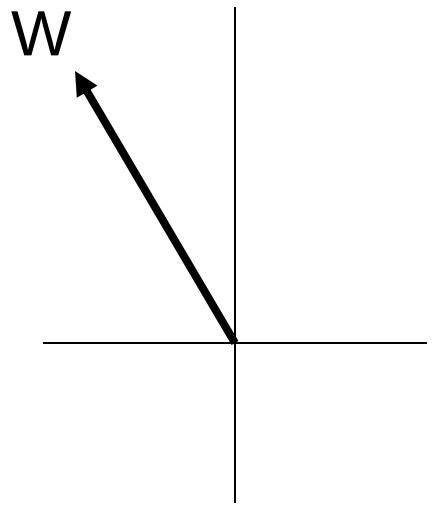
(But you want to know more, don't you?)

The math verison

Support vector machines

Vector

- A vector W in *d*-dimensional space is a list of *d* numbers, e.g. W=(-1,2)’
- A vector is a vertical column. ’ is matrix transpose
- Vector is a line segment, with arrow, in space
- The norm of a vector ||W|| is its length



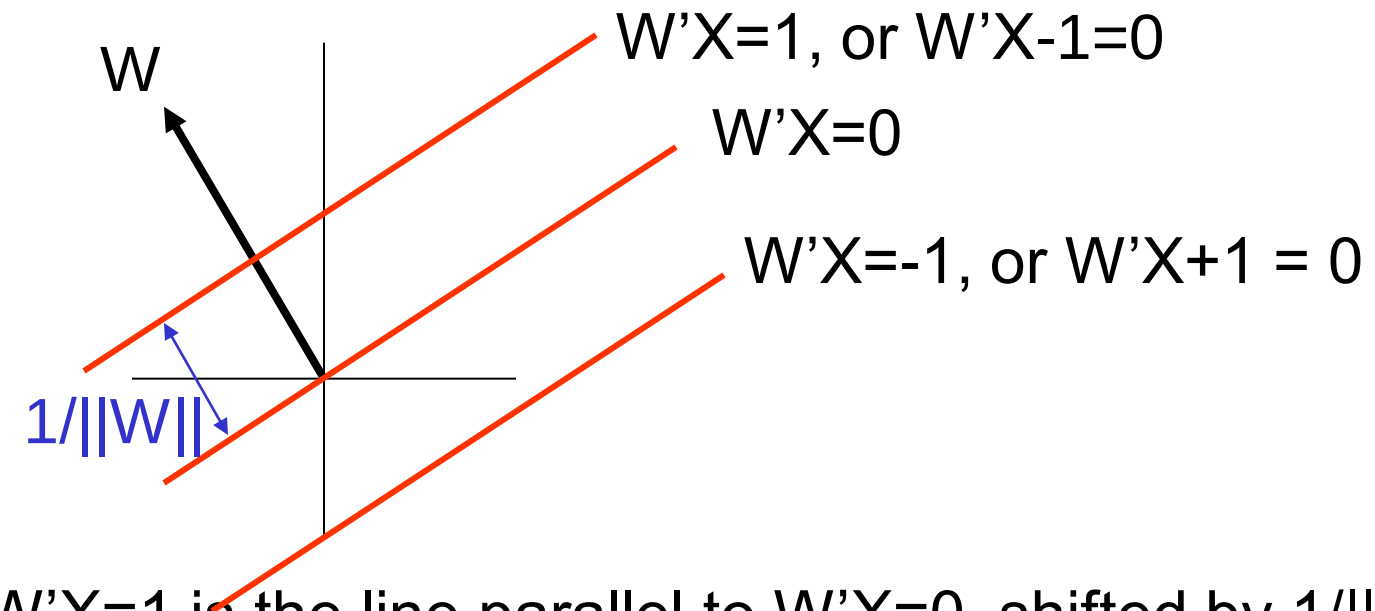
||W|| = sqrt(W’ W)

Inner product: $X'Y = \sum X_i Y_i$

What d-dimensional point X makes W’ X = 0?

Lines

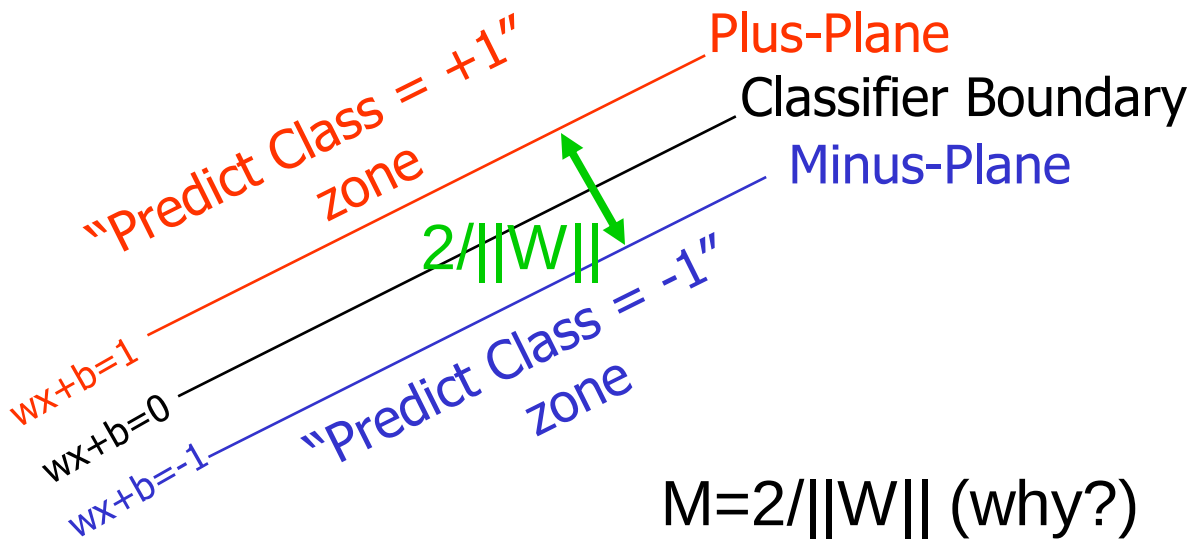
- W' X is a scalar (single number): X's projection onto W
- W'X=0 specifies the set of points X, which is the line perpendicular to W, and intersects at (0,0)



- W'X=1 is the line parallel to W'X=0, shifted by 1/||W||
- What if the boundary doesn't go through origin?

SVM boundary and margin

- Want: find W, b (offset) such that
  - all positive training points (X,Y=1) in the red zone,
  - all negative ones (X, Y=-1) in the blue zone,
  - the margin M is maximized

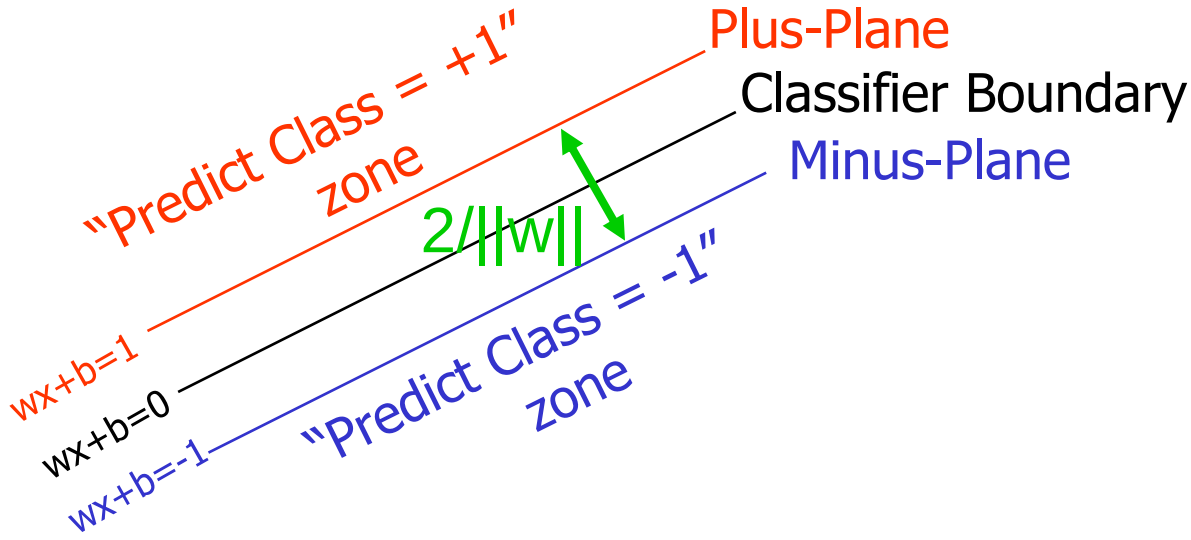


M=2/||W|| (why?)
How do we find such W,b?

SVM as constrained optimization

- Variables: W, b
- Objective function: maximize the margin M=2/||W||
- Equiv. to minimize ||W||, or $||W||^2$=W'W, or ½W'W

- Assume N training points (Xi, Yi), Yi = 1 or -1
- Subject to each training point on the correct side (the constraint). How many constraints do we have?



SVM as constrained optimization

- Variables: W, b
- Objective function: maximize the margin M=2/||W||
- Equiv. to minimize ||W||, or $||W||^2$=W'W, or ½W'W

- Assume N data points (Xi, Yi), Yi = 1 or -1
- Subject to each training point on the correct side (the constraint). How many constraints do we have? N
  - W'Xi + b >= 1, if Yi=1
  - W'Xi + b <= -1, if Yi=-1
  - we can unify them: Yi(W'Xi+b) >= 1

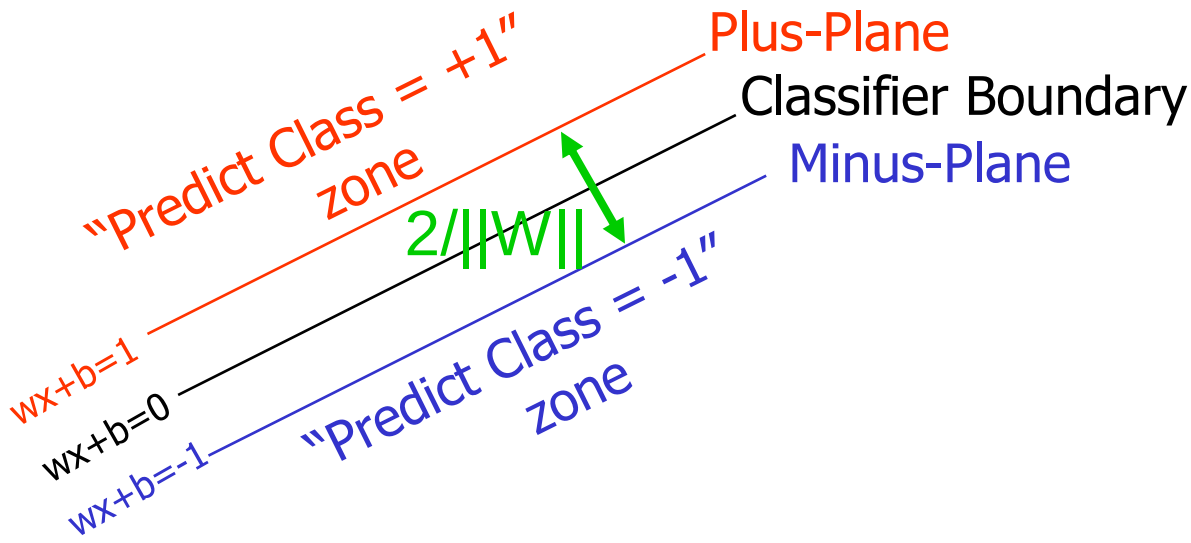
- We have a continuous constrained optimization problem! What do we do?

SVM as QP

$$\min_{W,b} \tfrac{1}{2}\, W'W$$

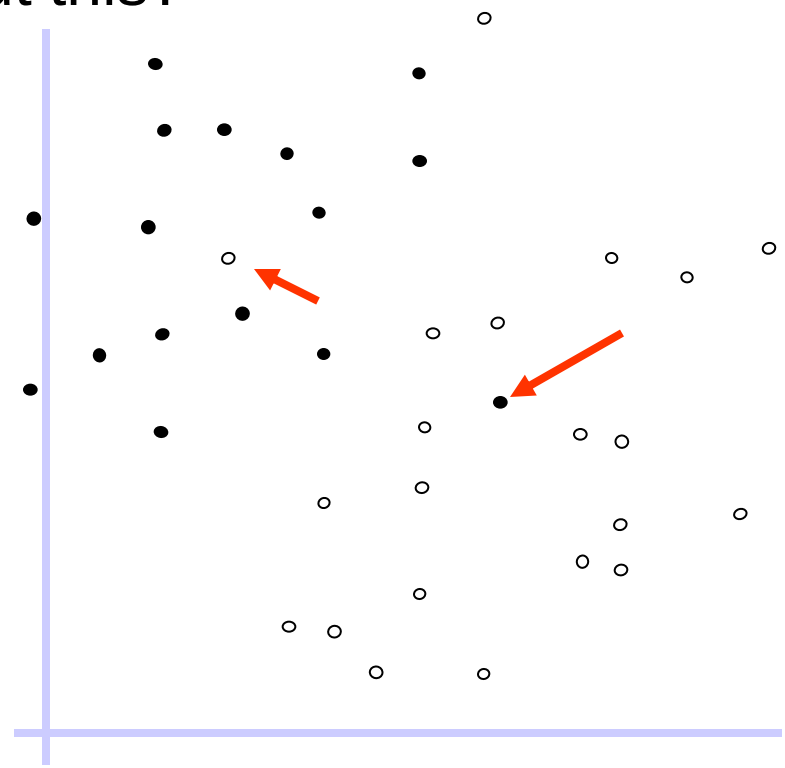
Subject to $Y_i (W'X_i + b) >= 1$, for all i

- Objective is convex, quadratic
- Linear constraints
- This problem is known as Quadratic Program (QP), for which efficient global solution algorithms exist.



Non-separable case

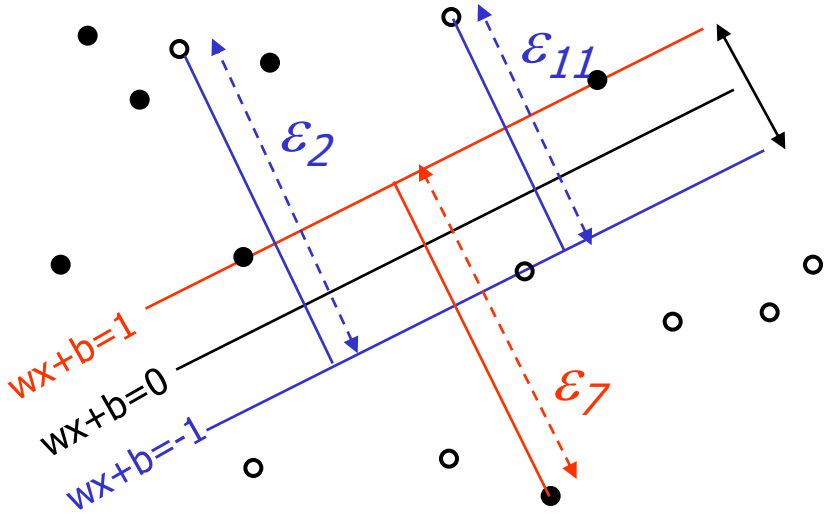
- What about this?



Can we insist on Yi (W'Xi + b) >= 1, for all i?

Trick #1: slack variables

- Relax the constraints – allow a few “bad apples”
- For a given linear boundary W, b, we can compute how far off into the wrong side a bad point is



- Here’s how we relax the constraints:

$$Yi\ (W'Xi + b) >= 1 - \varepsilon_i$$

Trick #1: SVM with slack variables

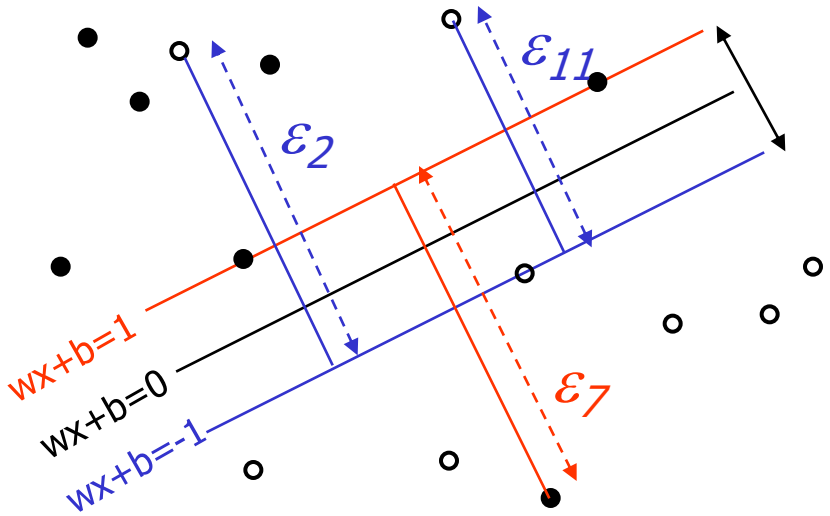
$$\min_{W,b,\varepsilon} \tfrac{1}{2}\, W'W + C \sum_i \varepsilon_i$$

Trade-off parameter

Subject to

$$Y_i\,(W'X_i + b) >= 1 - \varepsilon_i \text{ for all } i$$

$$\varepsilon_i >= 0 \text{ for all } i \text{ (why?)}$$



Trick #1: SVM with slack variables

$$\min_{W,b,\varepsilon} \tfrac{1}{2}\, W'W + C \sum_i \varepsilon_i$$

Trade-off parameter

Subject to

$$Yi\,(W'Xi + b) >= 1 - \varepsilon_i \text{ for all } i$$

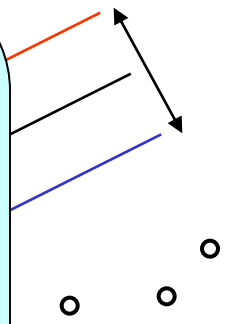
$$\varepsilon_i >= 0 \text{ for all } i$$

Originally we optimize variables W (d-dimensional vector), b

Now we optimize W, b, $\varepsilon_1 \ldots \varepsilon_N$

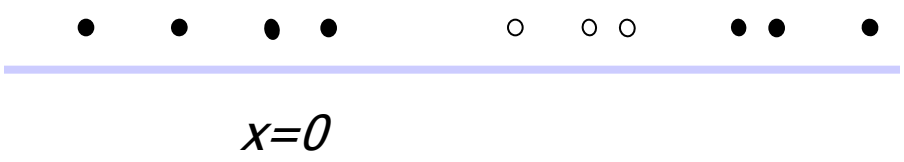
Now we have 2N constraints

Still a QP (**soft-margin SVM**)



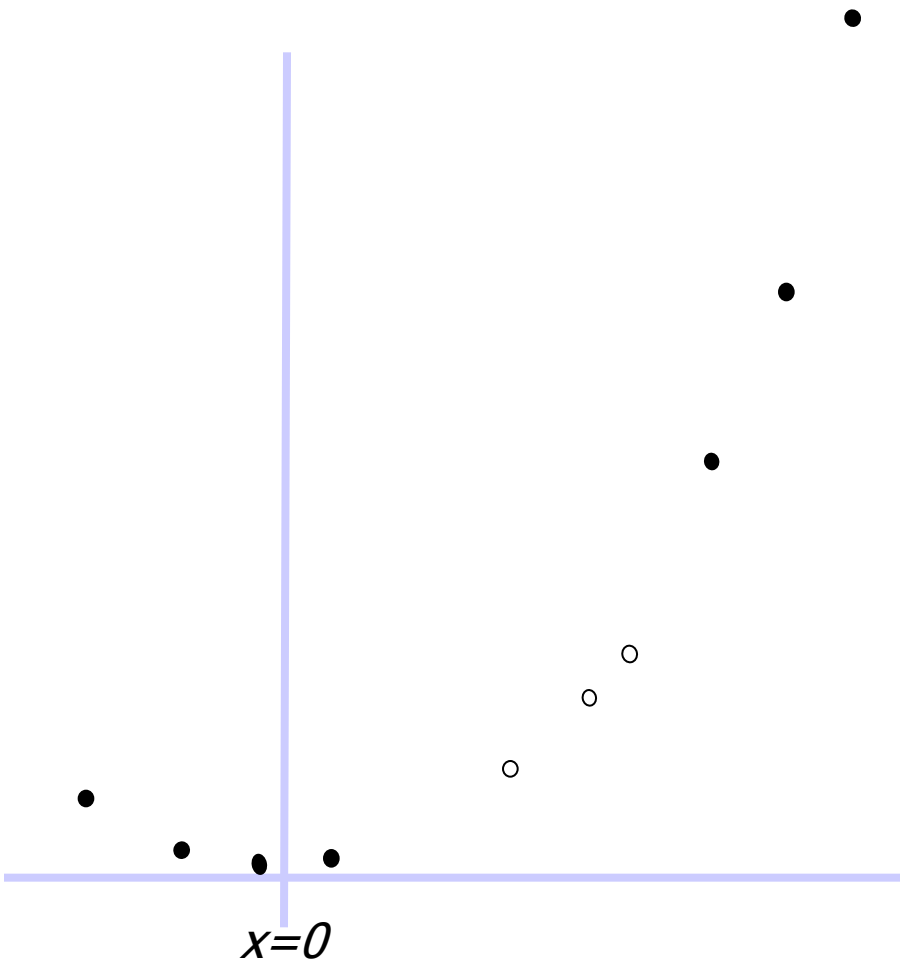
Another look at non-separable case

- Here's another non-separable dataset, in 1-dimensional space
- We can of course use slack variables… but here's another trick!



Trick #2:Map data to high dimensional space

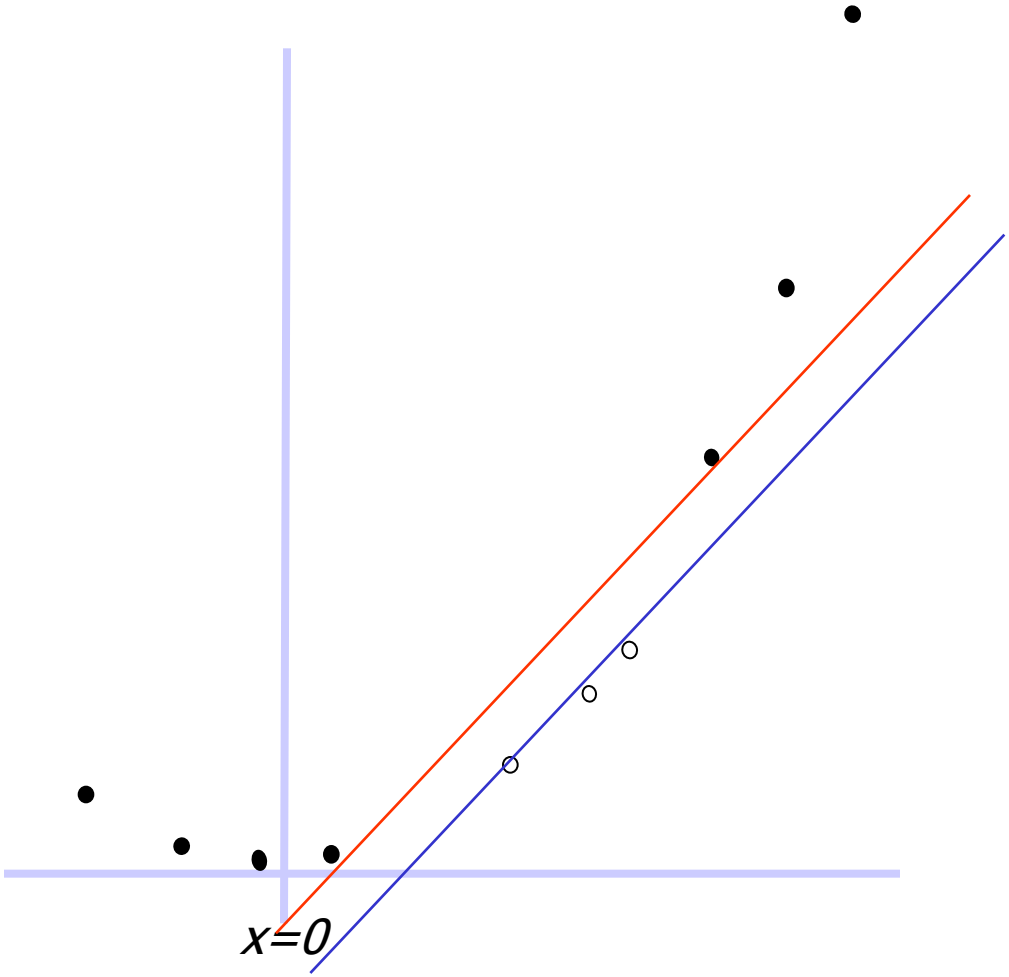
- We can map data x from 1-D to 2-D by x → (x, x^2)



Trick #2:Map data to high dimensional space

- We can map data x from 1-D to 2-D by

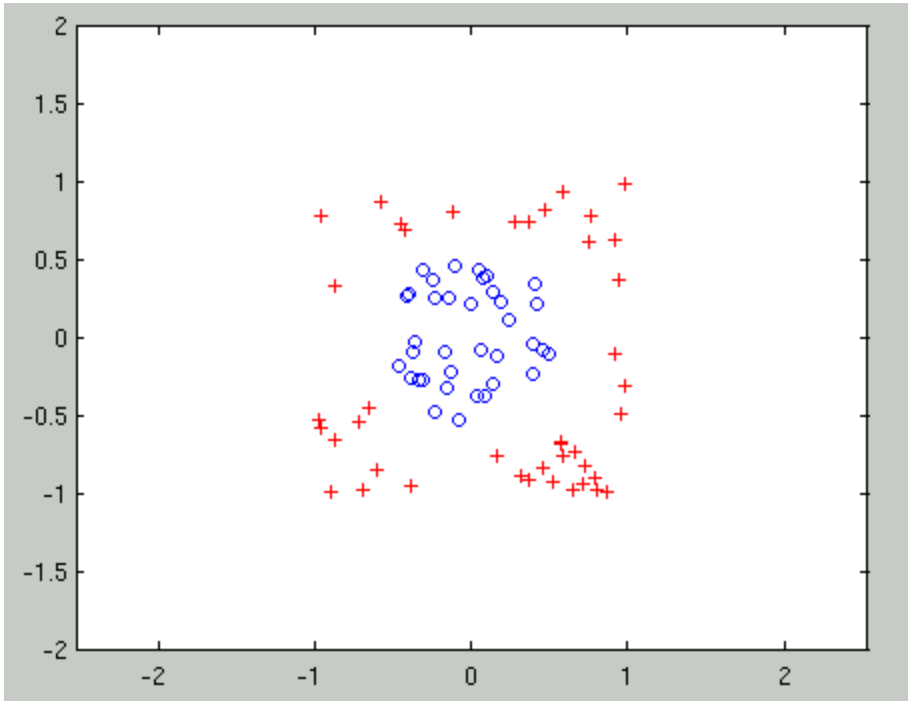
$$x \rightarrow \Phi(x)=(x, x^2)$$

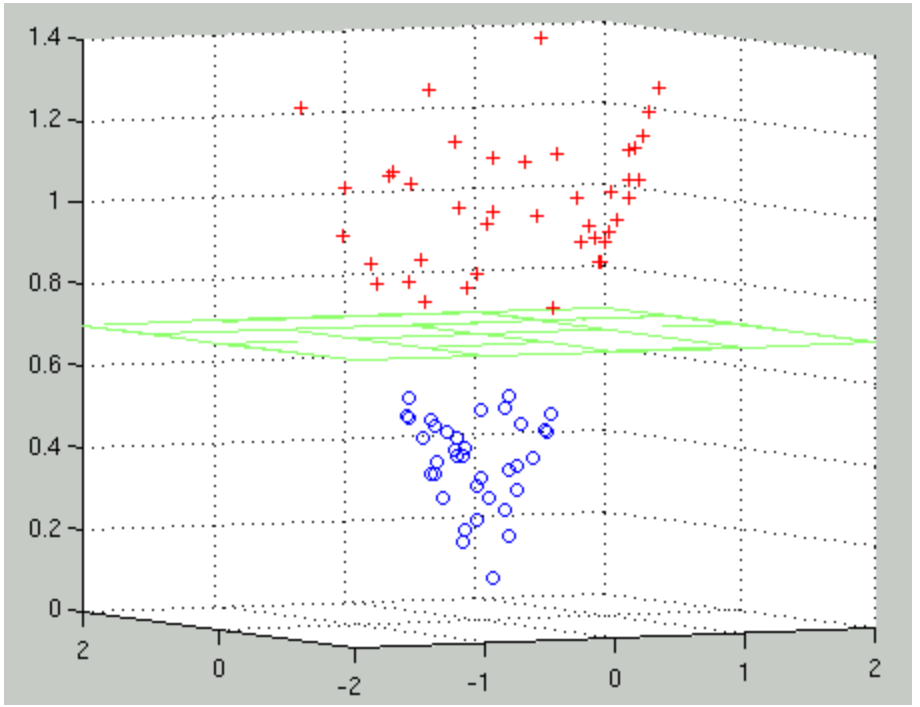


•Now the data is linearly separable in the new space!
•We can run SVM in the new space
•The linear boundary in the new space corresponds to a non-linear boundary in the old space

Another example

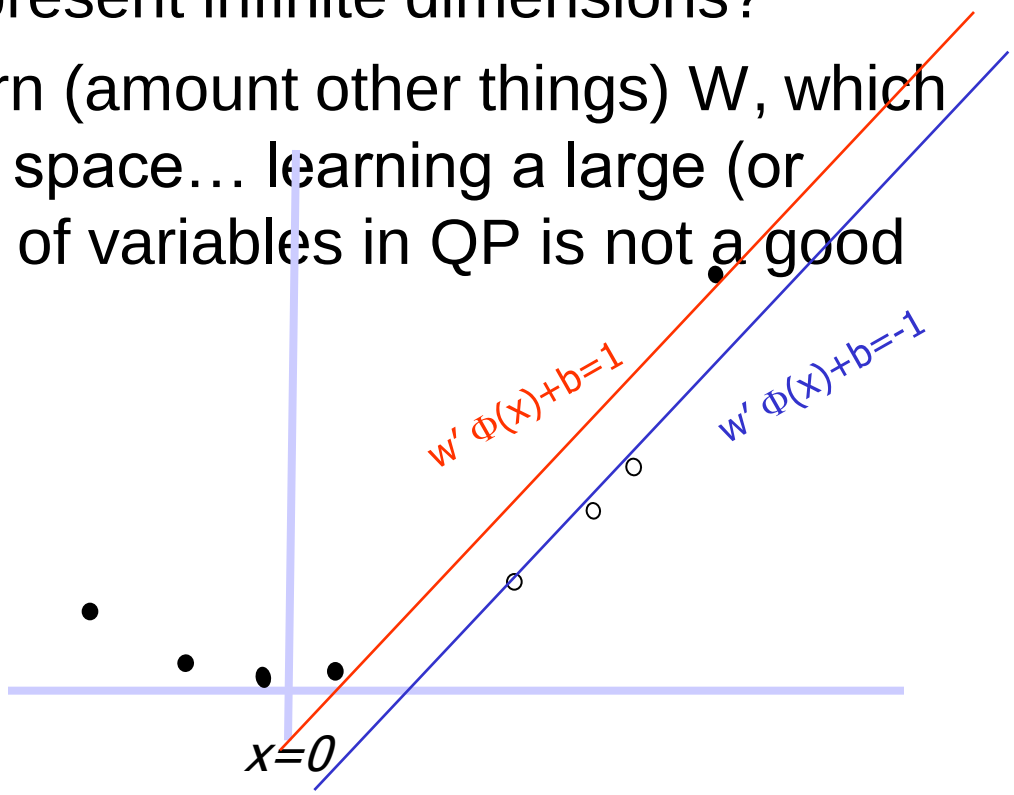
$$(x_1, x_2) \Rightarrow (x_1, x_2, \sqrt{x_1^2 + x_2^2})$$





Trick #2:Map data to high dimensional space

- In general we might want to map an already high-dimensional $X=(x_1, x_2, \ldots, x_d)$ into some much higher, even infinite dimensional space $\Phi(x)$
- Problems:
  - How do you represent infinite dimensions?
  - We need to learn (amount other things) W, which lives in the new space… learning a large (or infinite) number of variables in QP is not a good idea.



Trick #2: kernels

- We will do several things:
  - Convert it into a equivalent QP problem, which does not use W, or even $\Phi(X)$ alone!
    - It only uses the inner product $\Phi(X)' \Phi(Y)$, where X and Y are two training points. The solution also only uses such inner product
    - It still seems infeasible to compute for high (infinite) dimensions
  - But there are smart ways to compute such inner product, known as kernel (a function on two var.)
    - kernel(X,Y) $\Leftrightarrow$ inner product $\Phi(X)' \Phi(Y)$
- Why should you care:
  - One kernel, one new (higher dimensional) space
  - You will impress friends at cocktail parties

Prepare to bite the bullet…

- Here's the original QP formula

$$\min_{W,b} \tfrac{1}{2}\, W'W$$

Subject to $Y_i (W'X_i + b) >= 1$, for all i (N constraints)

- Remember Lagrange multipliers?

$$L = \tfrac{1}{2}\, W'W - \sum a_i [Y_i (W'X_i + b) - 1]$$

Subject to $a_i >= 0$, for all i

We have them here because those are inequality constraints

- We want the gradient of L to vanish with respect to W, b, a. Try this and you'll get

$$W = \sum a_i Y_i X_i$$

$$\sum a_i Y_i = 0$$

Put them back into the Lagrangian L

Biting the bullet

$$\max_{\{ai\}} \sum ai - \tfrac{1}{2} \sum_{i,j} ai\ aj\ Yi\ Yj\ \mathbf{Xi'\ Xj}$$

Subject to

$$ai >= 0, \text{ for all I}$$

$$\sum ai\ Yi = 0$$

- This is an equivalent QP problem (the dual)
- Before we optimize W (d variables), now we optimize a (N variables): which is better?
- X only appears in the inner product

Biting the bullet

$$\max_{\{ai\}} \sum ai - \tfrac{1}{2} \sum_{i,j} ai\ aj\ Yi\ Yj\ \Phi(Xi)' \Phi(Xj)$$

Subject to

$$ai >= 0, \text{ for all I}$$

$$\sum ai\ Yi = 0$$

If we map X to new space $\Phi(X)$

- This is an equivalent QP problem
- Before we optimize W (d variables), now we optimize a (N variables): which is better?
- X only appears in the inner product

Biting the bullet

$$\max_{\{ai\}} \sum ai - \tfrac{1}{2} \sum_{i,j} ai\ aj\ Yi\ Yj\ \mathbf{K(Xi, Xj)}$$

Subject to

$$ai >= 0, \text{ for all I}$$

$$\sum ai\ Yi = 0$$

If we map X to new space **Φ(X)**

- This is an equivalent QP problem
- Before we optimize W (d variables), now we optimize a (N variables): which is better?
- X only appears in the inner product

Kernel K(Xi,Xj) = **Φ(Xi)' Φ(Xj)**

What's special about kernel

- Say data is two dimensional: s=(s1,s2)
- We decide to use a particular mapping into 6 dimensional space

$$\Phi(s)=(s1^2, s2^2, \sqrt{2}\, s1s2, s1, s2, 1)$$

- Let another point be t=(t1,t2).

$$\Phi(s)' \Phi(t)=s1^2\ t1^2 + s2^2\, t2^2 + 2\, s1s2t1t2 + s1t1 + s2t2 + 1$$

What's special about kernel

- Say data is two dimensional: s=(s1,s2)
- We decide to use a particular mapping into 6 dimensional space

$$\Phi(s)=(s1^2, s2^2, \sqrt{2}\, s1s2, \sqrt{2}\, s1, \sqrt{2}\, s2, 1)$$

- Let another point be t=(t1,t2).

$$\Phi(s)' \Phi(t)=s1^2\, t1^2 + s2^2\, t2^2 + 2\, s1s2t1t2 + 2\, s1t1 + 2\, s2t2 + 1$$

- Let the kernel be $K(s,t) = (s't+1)^2$
- Verify that they are the same. We saved computation.

Kernels

- You ask: “Is there such a good K for any Φ I pick?”
- The inverse question: “Given some K, is there a Φ so that K(X,Y) = Φ(X)’ Φ(Y)?”
- Mercer’s condition: the inverse question is true…

$$\text{if for any } g(\mathbf{x}) \text{ such that } \int g(\mathbf{x})^2 d\mathbf{x} \text{ is finite}$$

$$\int K(\mathbf{x},\mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0.$$

- This is positive semi-definiteness, if you must know
- Φ may be infinite dimensional; we may not be able to explicitly write down Φ

Some frequently used kernels

- Linear kernel: $K(X,Y) = X'Y$
- Quadratic kernel: $K(X,Y) = (X'Y+1)^2$
- Polynomial kernel: $K(X,Y) = (X'Y+1)^n$
- Radial Basis Function kernel: $K(X,Y) = \exp(- ||X-Y||^2/\sigma)$
- Many, many other kernels
- Hacking with SVM: create various kernels, hope their space Φ is meaningful, plug them into SVM, pick the one with good classification accuracy (equivalent to feature engineering)
- Kernel summary: QP of size N, nonlinear SVM in the original space, new space possibly high/infinite dim, efficient if K is easy to compute
- Kernel can be combined with slack variables

Why the name "support vector machines"

$$\max_{\{ai\}} \sum ai - \tfrac{1}{2} \sum_{i,j} ai\ aj\ Yi\ Yj\ \mathbf{K(Xi, Xj)}$$

Subject to

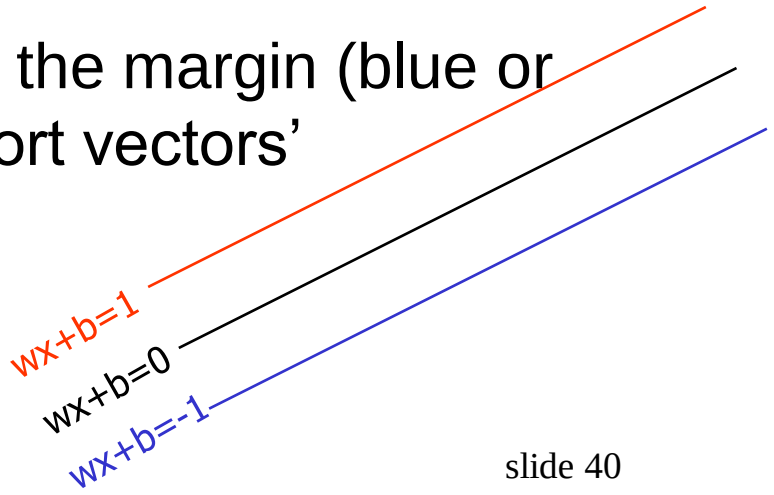
$$ai >= 0, \text{ for all I}$$

$$\sum ai\ Yi = 0$$

- The decision boundary is

$$f(Xnew)=W' Xnew + b= \sum ai\ Yi\ Xi' Xnew + b$$

- In practice, many *a*'s will be zero in the solution!
  - Those few X with a>0 lies on the margin (blue or red lines), they are the 'support vectors'



What you should know

- The intuition, where to find software
- Vector, line, length
- Margin
- QP with linear constraints
- How to handle non-separable data
  - Slack variables
  - Kernels ⇔ new feature space

- Ref: **A Tutorial on Support Vector Machines for Pattern Recognition (1998)** Christopher J. C. Burges