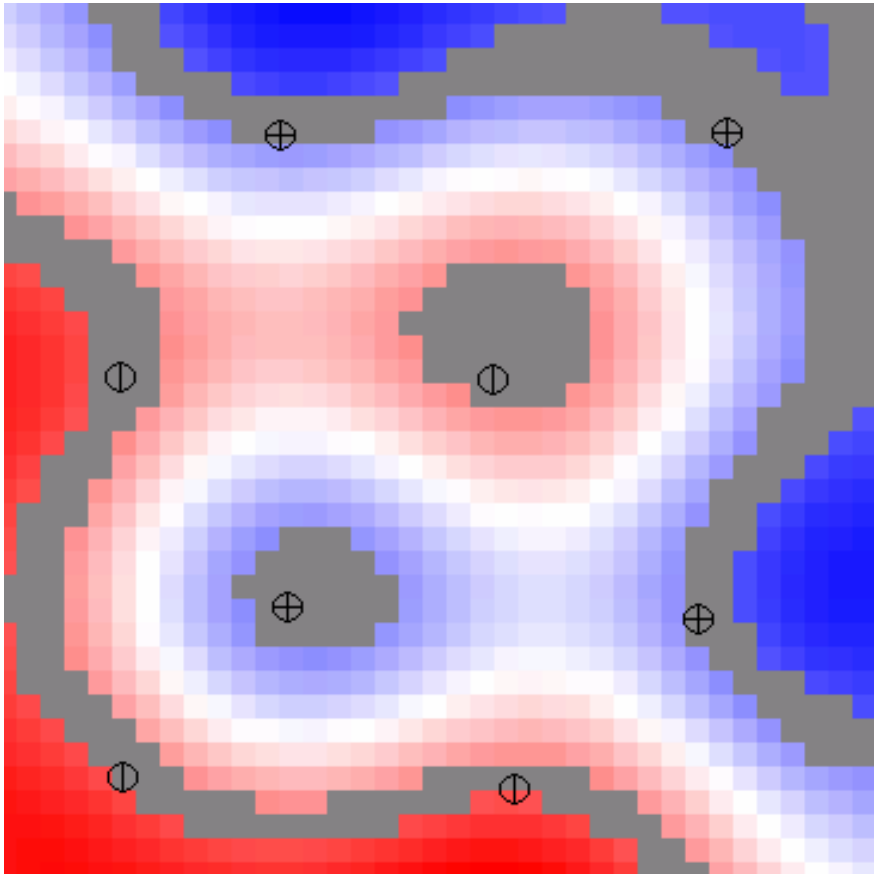
6.034 Artificial Intelligence: Lecture 19
Support Vector Machines

Robert C. Berwick

October 19, 2020



AI Methods

- **Problem solving**
  - **G+T, *search, optimal search, games, constraint satisfaction***
- **Inference**
  - ***rule-based systems, Bayesian inference***
- **Machine learning**
  - ***k-nearest neighbors, id trees, neural nets*, deep neural nets, *support vector machines*, genetic algorithms, near miss/one-shot**
- **Communication, perception, action**
  - **natural language processing, vision, robotics**

Vladimir Vapnik

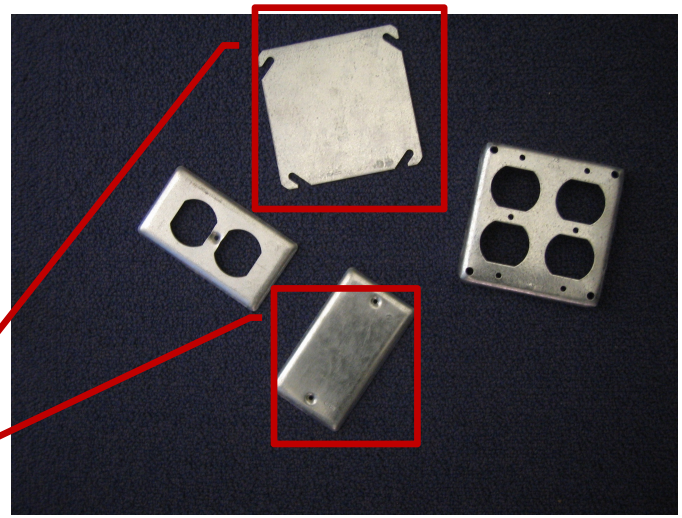
The evolution of an idea

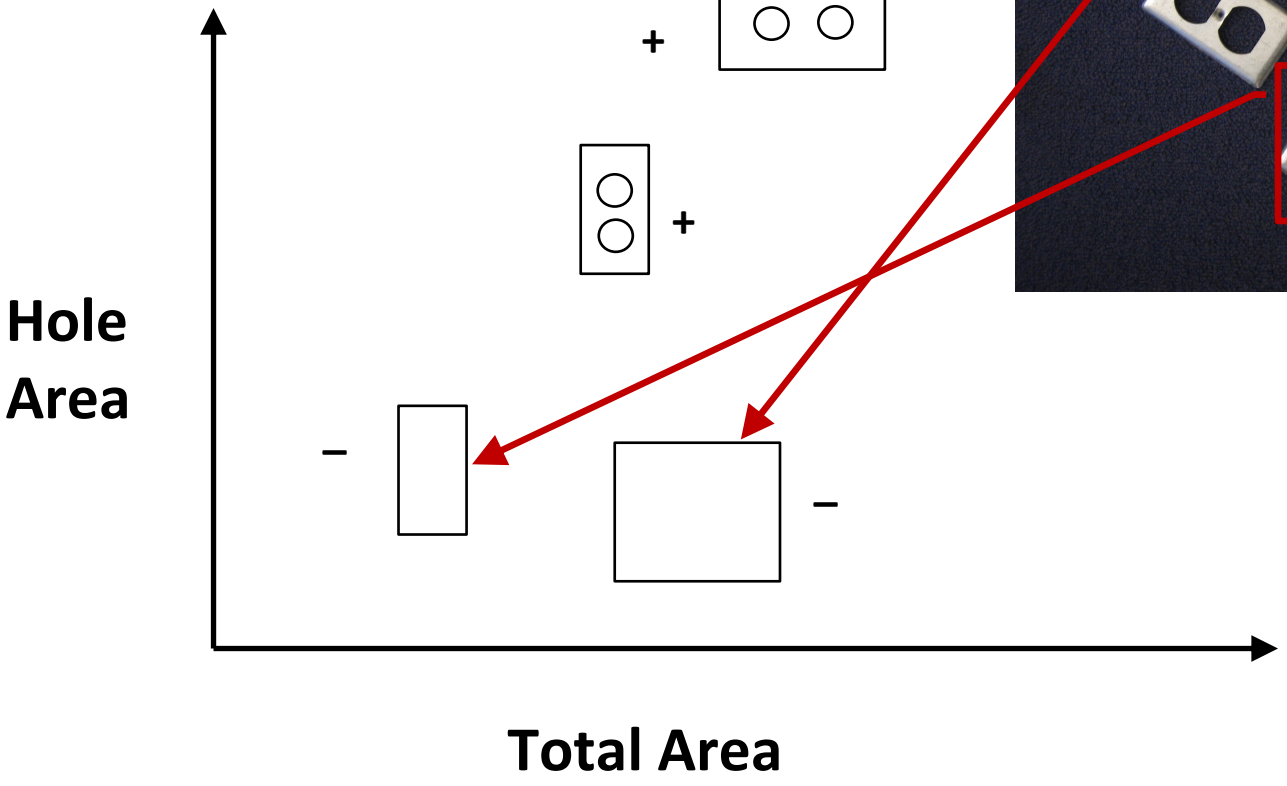
VAPNIK, V., 1979. *Estimation of Dependences Based on Empirical Data* [in Russian]. Moscow: Nauka.

SVM menu

- Decision boundaries: looking linear again
- Widest street approach
- Kernel functions: how changing representations can help
- History lesson

Representation choice





Many possible classifiers fit same data

But which classifier is “best”?
Which classifiers are better than others?
Is there some way to figure this out?
Is there some general method?

Yes!
Vladimir Vapnik
Support Vector Machines (SVMs)

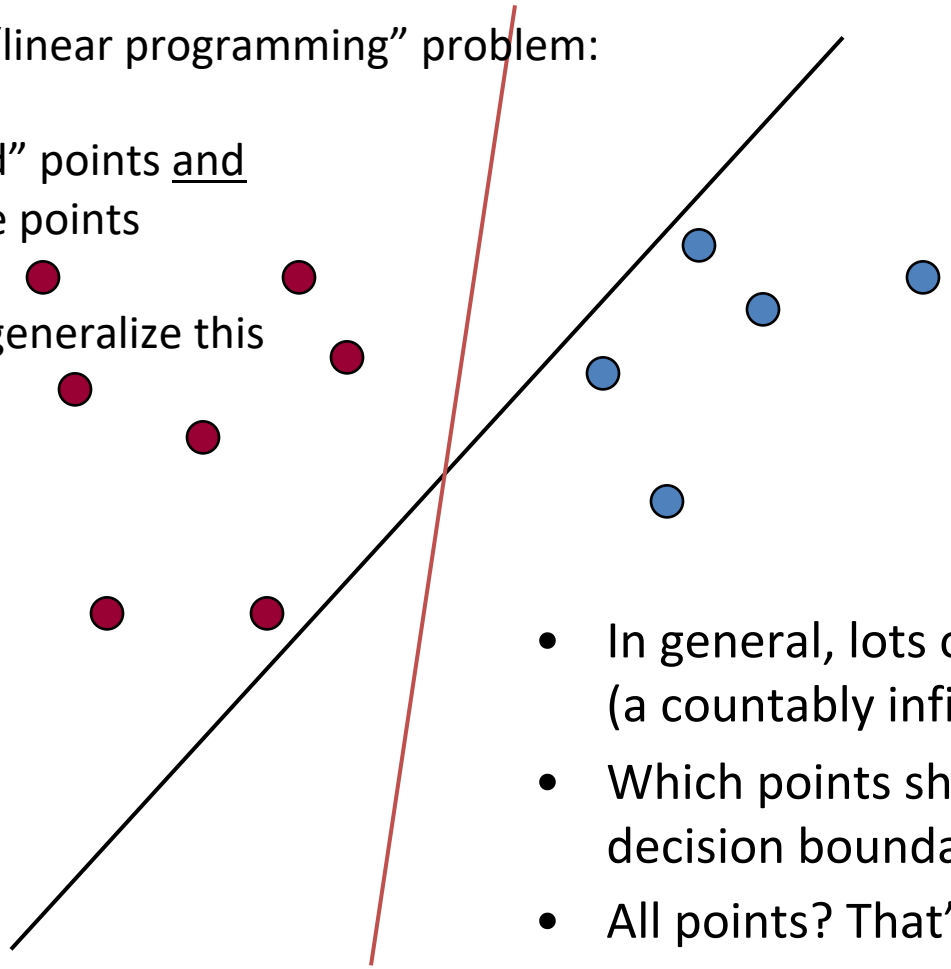
The evolution of an idea

VAPNIK, V., 1979. *Estimation of Dependences Based on Empirical Data* [in Russian]. Moscow: Nauka.

What's the basic idea? Recall 1-layer Perceptrons: decision boundary is a separating hyperplane

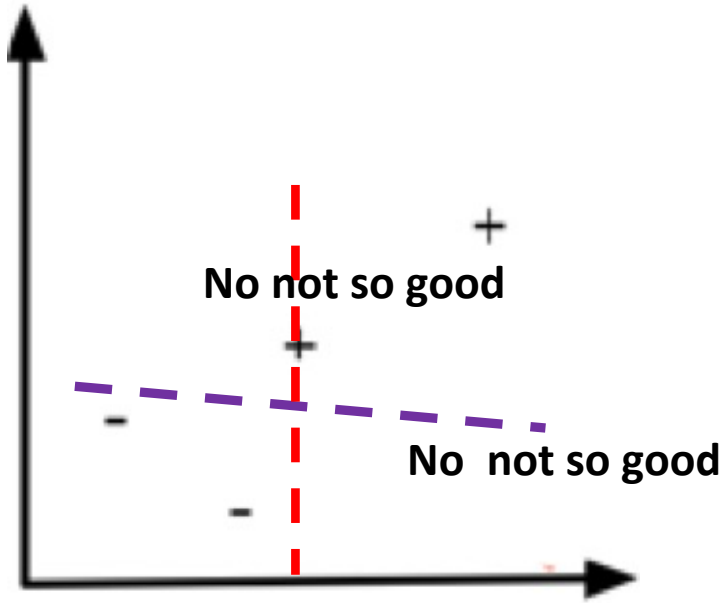
To do this we solve a "linear programming" problem:
Find *a, b, c* such that:
$ax + by > c$ for the "red" points and
$ax + by < c$ for the blue points

We are now going to generalize this

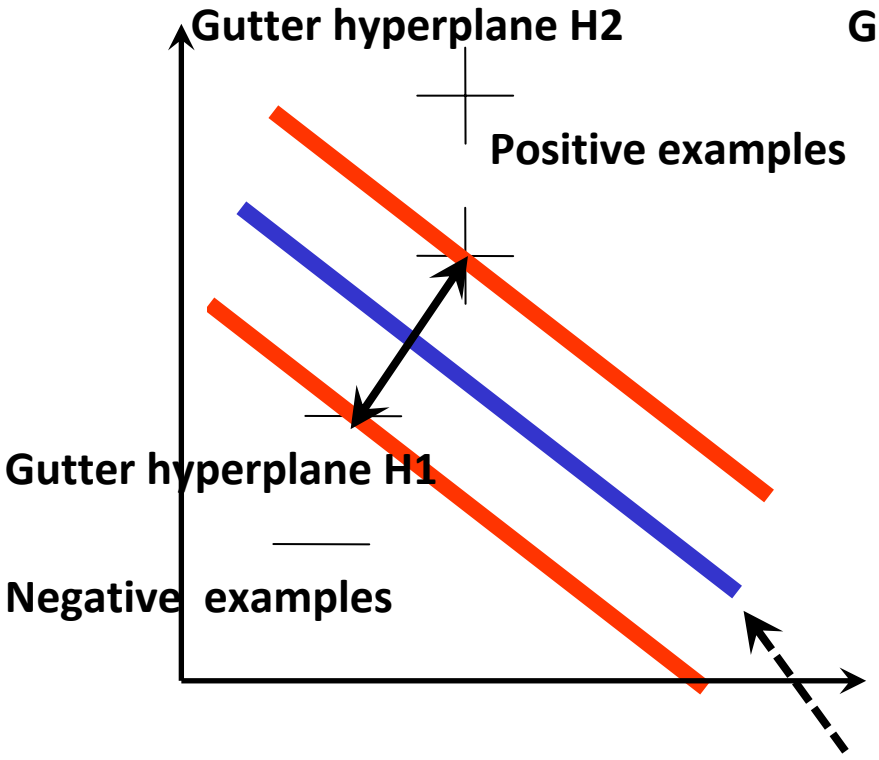


- In general, lots of possible solutions (a countably infinite number!)
- Which points should influence decision boundary?
- All points? That's linear regression
- "Difficult" points close to boundary? That's today's topic

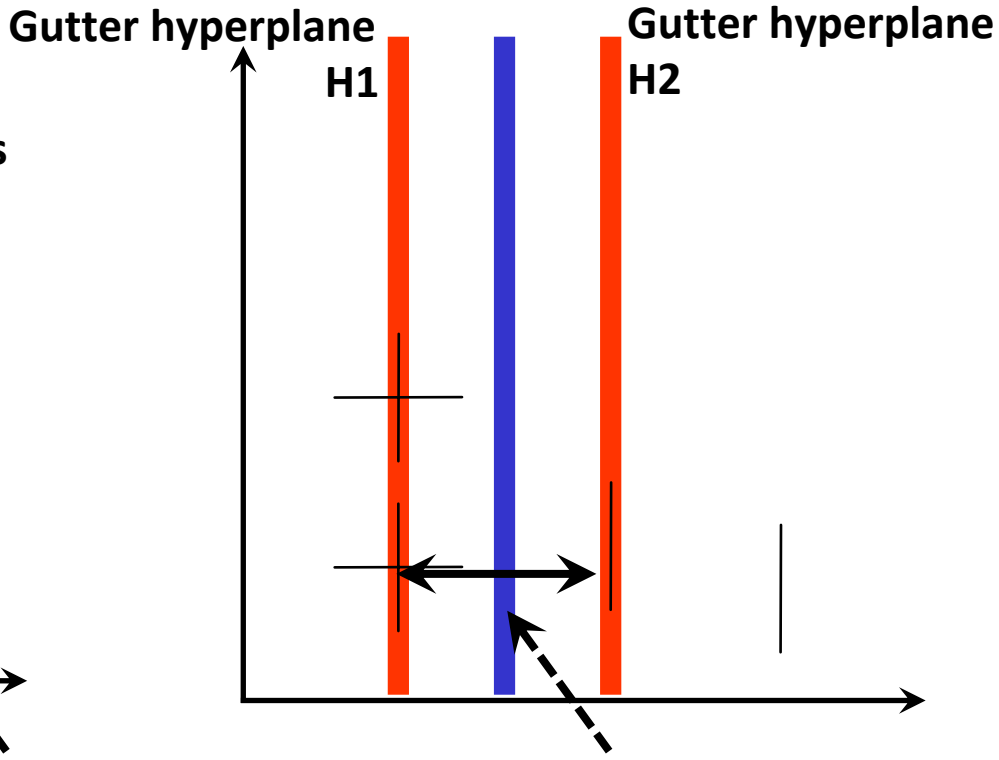
Which decision boundary is best?
Not all are as good as others – and some points are “more difficult” to classify than others



Idea: the "best" decision line boundary places the +, – examples as far apart from each other as possible, so that "street" separating them is as wide as possible



Decision boundary line
"middle of street" (median)

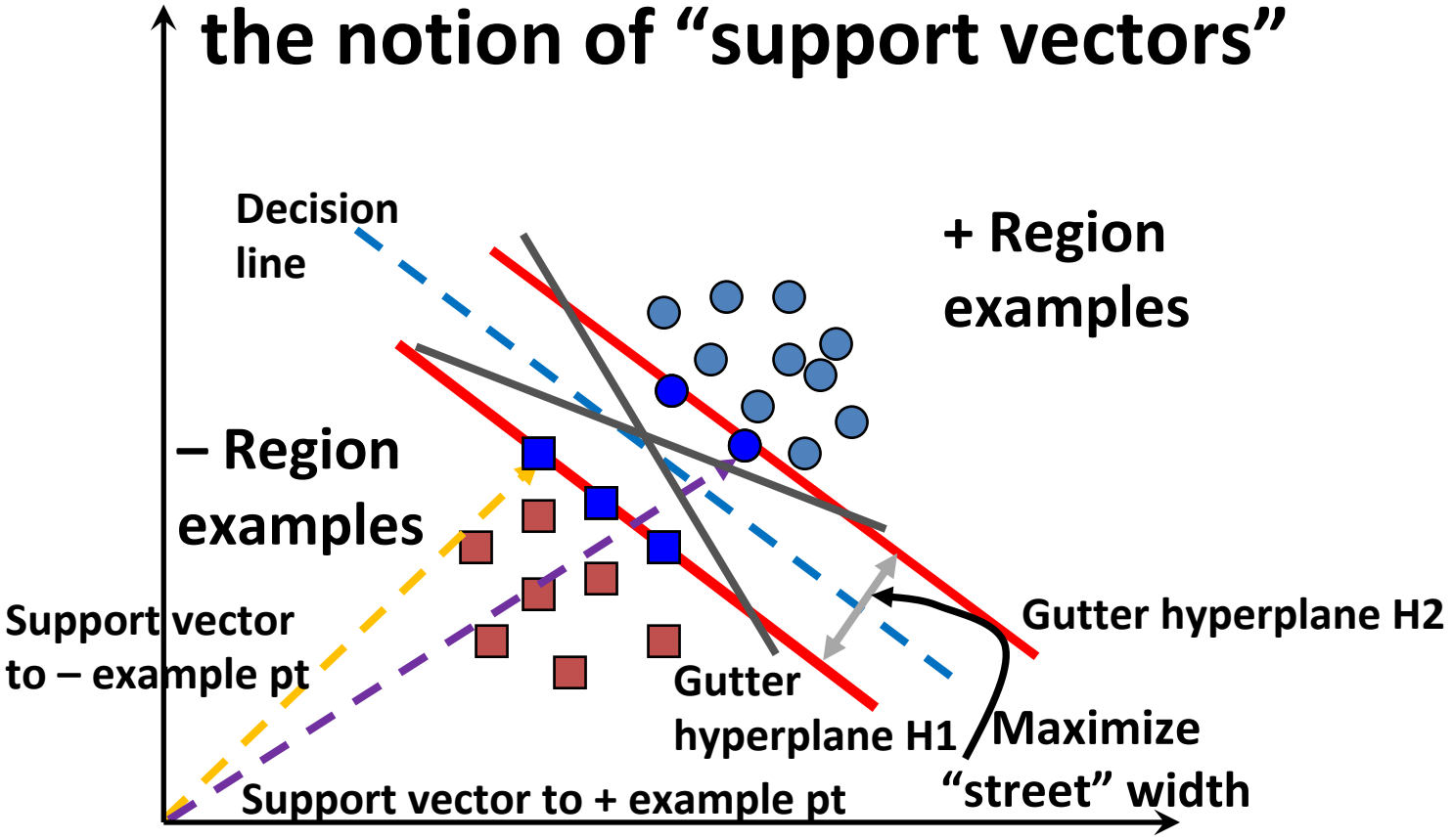


Decision boundary line

We want to <u>maximize</u> this "street width" (between gutter lines)

How to describe this "road"?

Formally, we add some geometric constraints to describe a particular separating hyperplane via the notion of "support vectors"

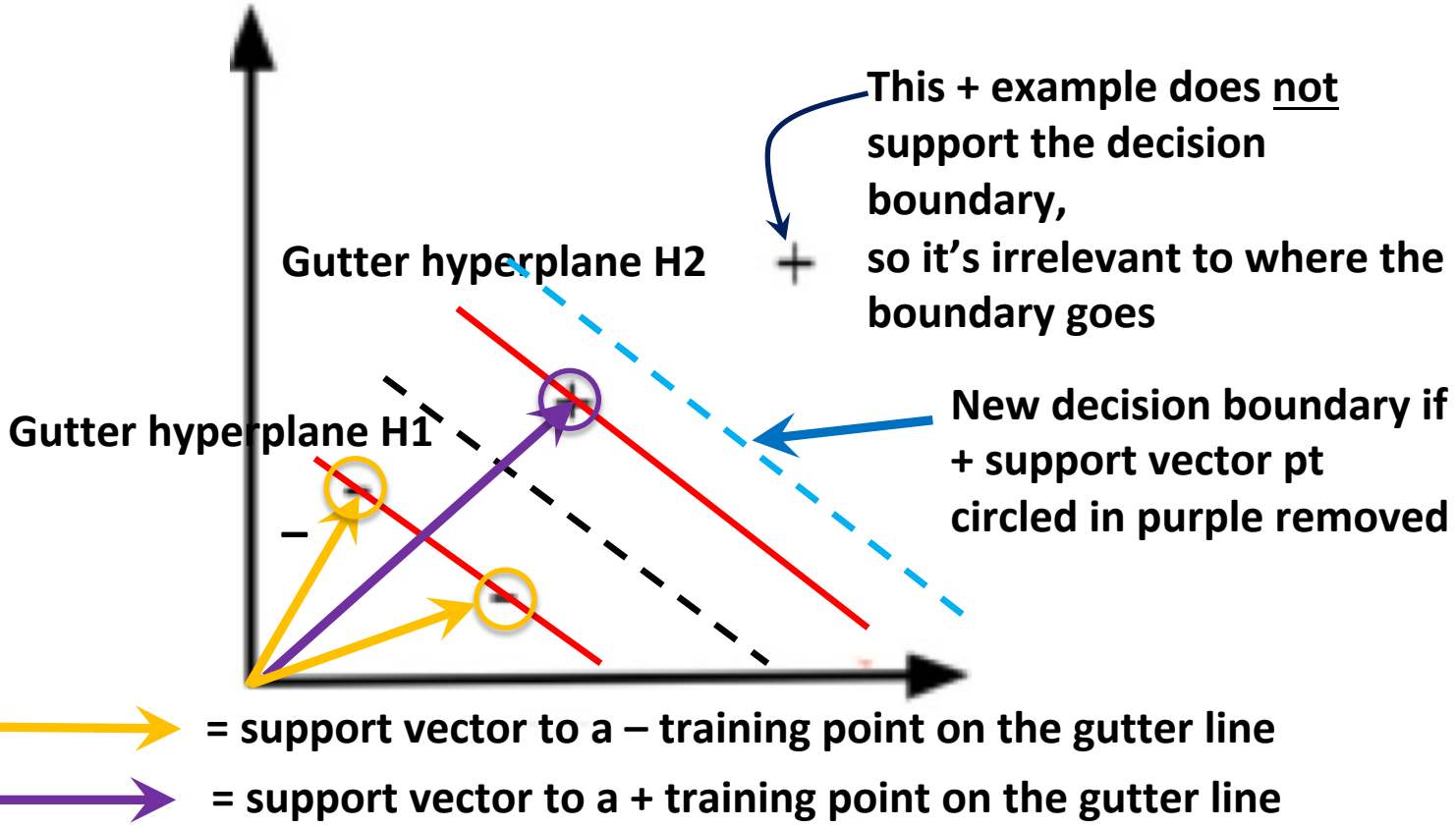


Support vectors touch positive, negative sample points that lie on the "gutters" that maximize the street width –

The optimal hyperplane is a linear combination of these support vectors

let's figure out what vector equations describing them

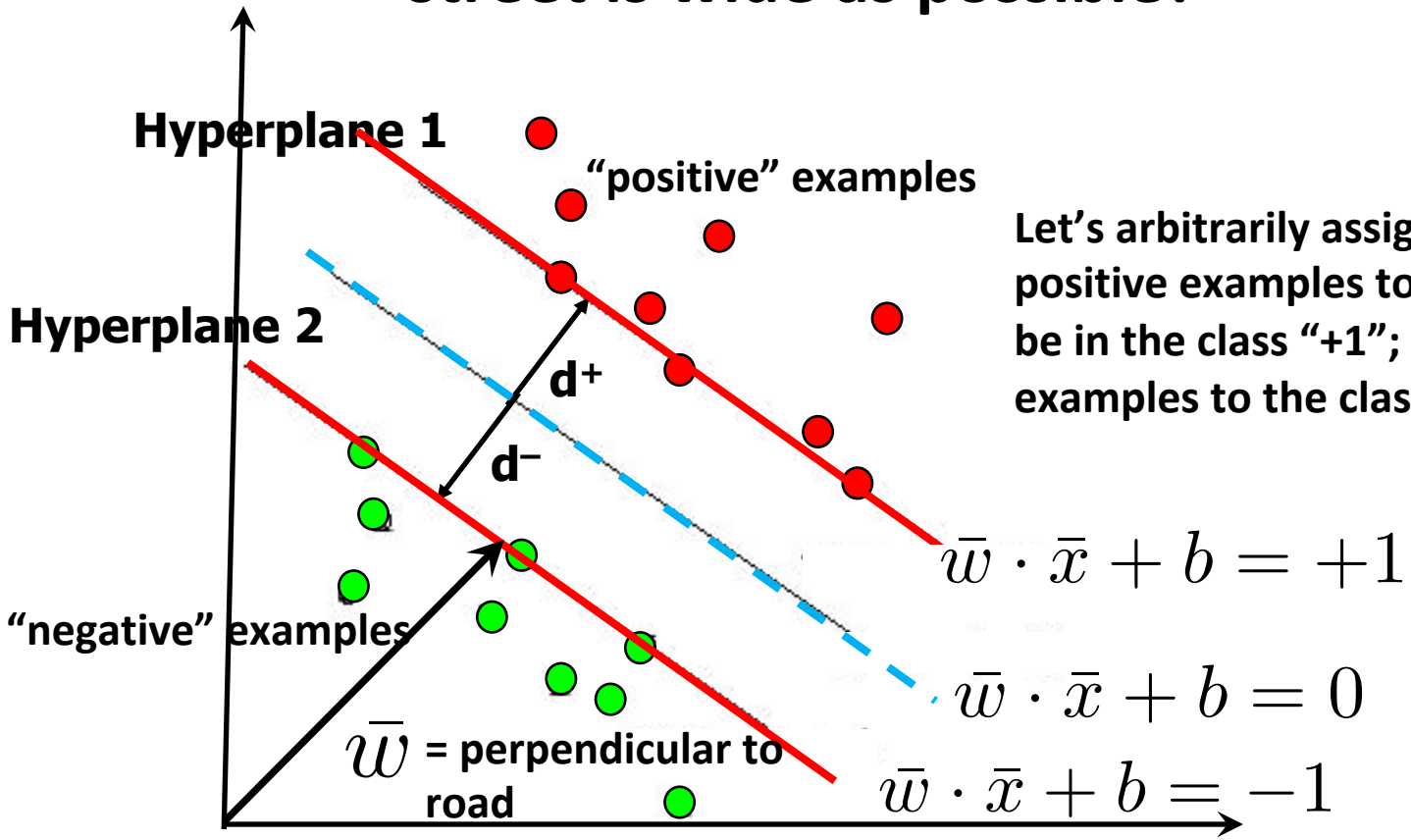
Goal: solve for maximal width street – the support vectors fix its width



A support vector is defined as a training point that, if removed, would change the decision boundary

Question: But given a set of {+, –} pts,
how do we compute the maximum street width and gutter lines (support vectors)?

Our goal is to compute this geometry of the 3 lines below: what are the values for these lines so that street is wide as possible?



Let's arbitrarily assign the positive examples to be in the class "+1"; the negative examples to the class "–1"

d+ = the shortest distance to the closest positive point

d- = the shortest distance to the closest negative point

The width of the street = the margins of the separating hyperplanes= $d^+ + d^-$

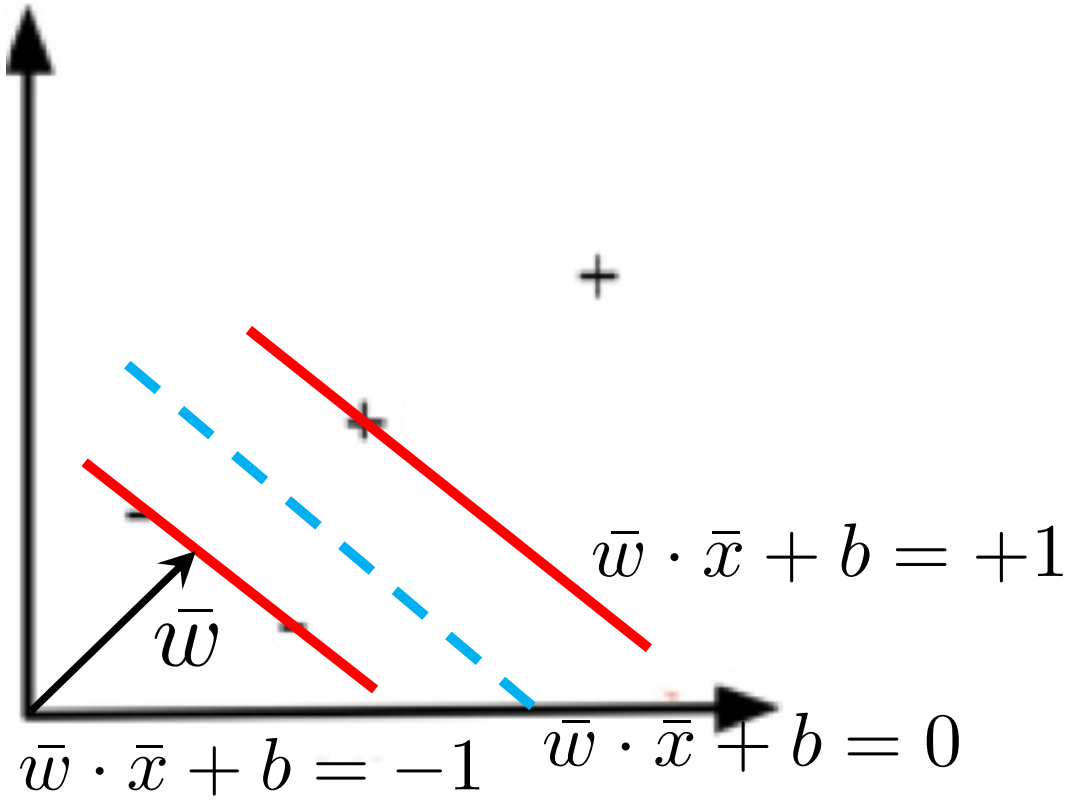
(We don't know what the value of d is yet, and we don't know what *b* is)

The rest of the game

- 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
- We convert the problem of finding the maximum width street hyperplanes while meeting the 3 line conditions constraints, into an optimization problem with side conditions
- We solve this via optimization techniques–we use Lagrange multipliers to derive a form that can be solved analytically by setting derivatives to zero

The view from the street and the decision boundary – again describing the 3 key line constraints that are imposed

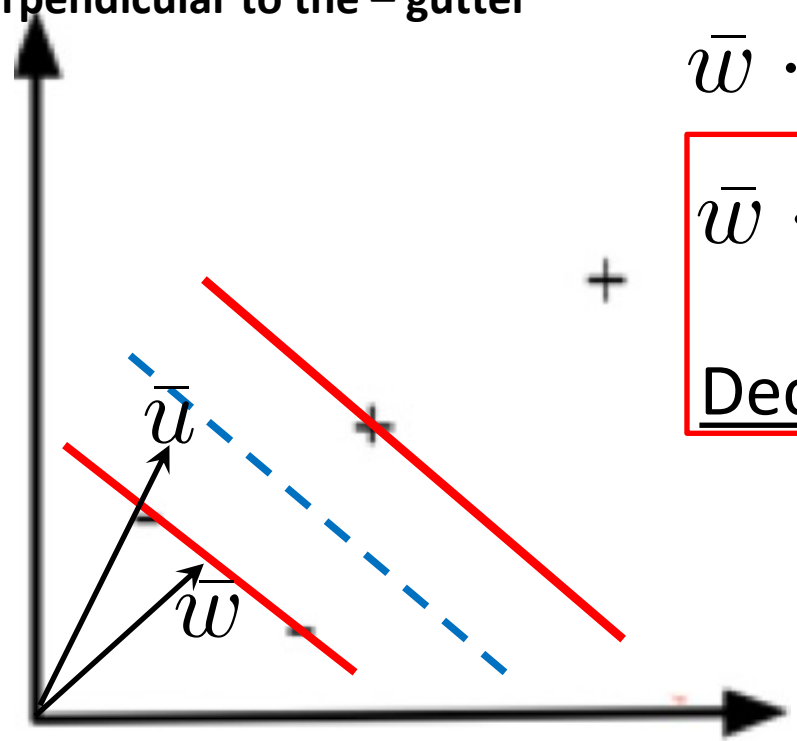
Note:



Step 1: Find a vector equation decision rule for classifying uknown pts & use it to construct constraints that must be satisfied on the gutter and middle of the road

$\bar{u}$ is unknown point to classify

$\bar{w}$ **Is a vector perpendicular to the – gutter**



$$\bar{w} \cdot \bar{u} \geq c \qquad c = -b$$

$$\bar{w} \cdot \bar{u} + b \geq 0 \text{ then } +$$

Decision Rule

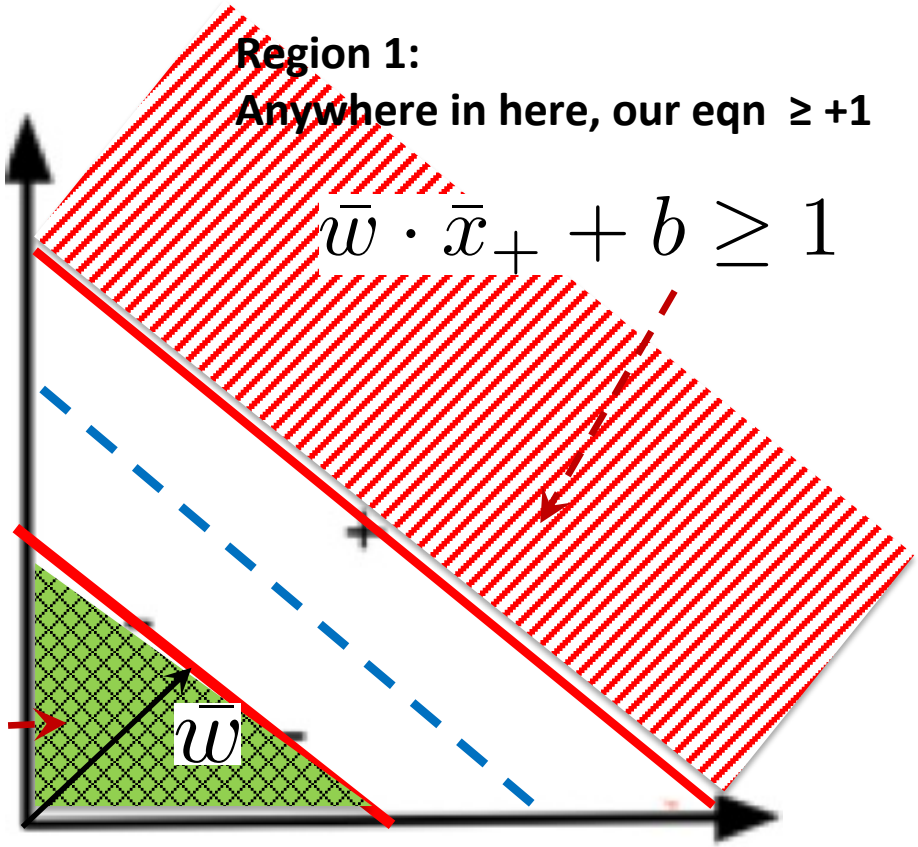
Step 1

Picture: if an unknown point *u* is in the street between the gutters, it is a + example
But, we don't know the value of *b* yet

Next, impose the constraints implied by the +, – sample points: force such points to be +1/–1 distance apart

$$\bar{w} \cdot \bar{x}_+ + b \geq 1$$

$$\bar{w} \cdot \bar{x}_- + b \leq -1$$



Recall: $\bar{w}$ **is perpendicular to decision line**

Region 3
Anywhere in here, our eqn ≤ –1

$$\bar{w} \cdot \bar{x}_- + b \leq -1$$

On the two red gutter lines, we have the equations:

$$\bar{w} \cdot \bar{x} + b = +1$$

$$\bar{w} \cdot \bar{x} + b = -1$$

For the blue median strip, we have the equation: $\bar{w} \cdot \bar{x} + b = 0$

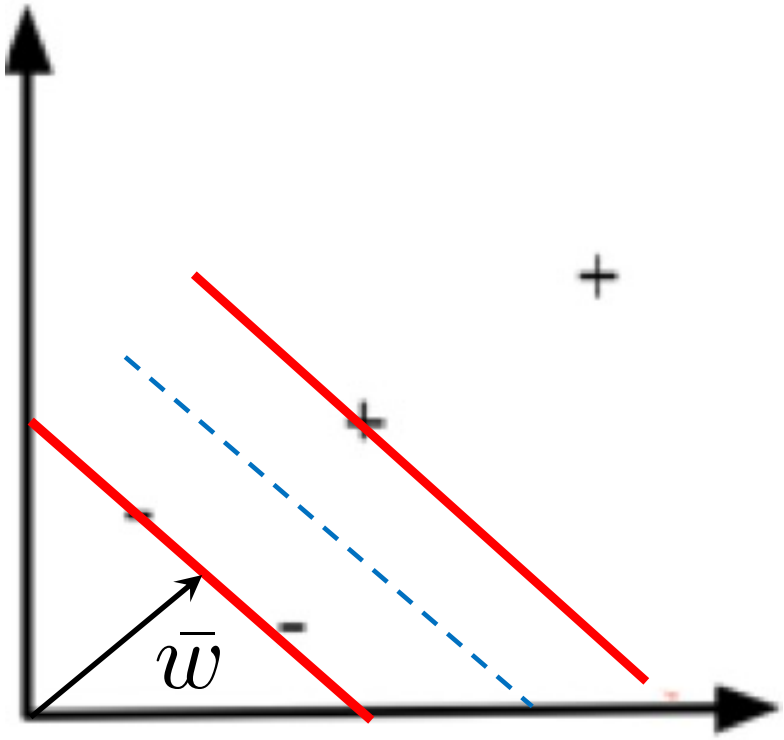
Introduce a new "outcome" variable y_i to combine these two inequality constraints into a single constraint

$$\bar{w} \cdot \bar{x}_+ + b \geq 1$$
$$\bar{w} \cdot \bar{x}_- + b \leq -1$$

y_i such that

$y_i = +1$ for $+$ samples

$y_i = -1$ for $-$ samples



Collapse these 2 constraints into one by using the outcome variable y_i

$$\bar{w} \cdot \bar{x}_+ + b \geq 1 \qquad y_i(\bar{w} \cdot \bar{x}_i + b) \geq 1$$

$$\bar{w} \cdot \bar{x}_- + b \leq -1 \qquad y_i(\bar{w} \cdot \bar{x}_i + b) \leq -1;$$

$$\text{for } y_i = -1 :$$

$$-1(\bar{w} \cdot \bar{x}_i + b) \leq -1 \text{ or}$$

$$y_i(\bar{w} \cdot \bar{x}_i + b) \geq 1$$

Put these together : $y_i(\bar{w} \cdot \bar{x}_i + b) - 1 \geq 0$

Outcome variable:

y_i such that $y_i(\bar{w} \cdot \bar{x}_i + b) - 1 \geq 0$

$y_i = +1$ for $+$ samples

$y_i = -1$ for $-$ samples

Collapsing the constraints on points inside & outside the gutter into one constraint

$$y_i(\bar{w} \cdot \bar{x}_i + b) \geq 1$$

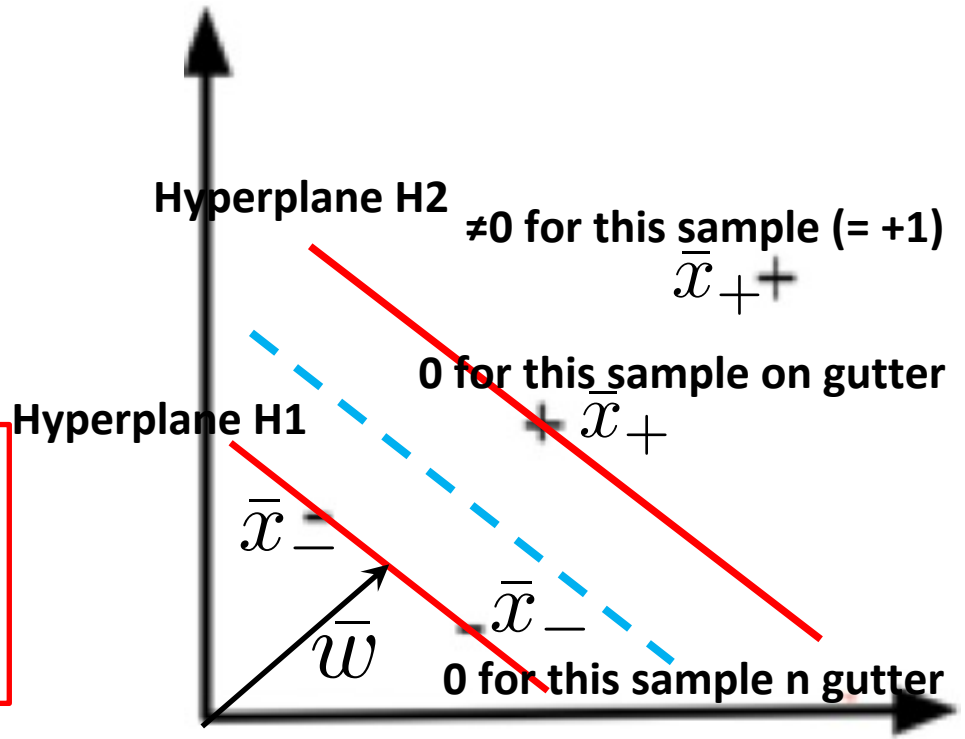
$$y_i(\bar{w} \cdot \bar{x}_i + b) \geq 1$$

$$y_i(\bar{w} \cdot \bar{x}_i + b) - 1 \geq 0$$

STEP 2

$$y_i(\bar{w} \cdot \bar{x}_i + b) - 1 = 0$$

for points x_i in gutter

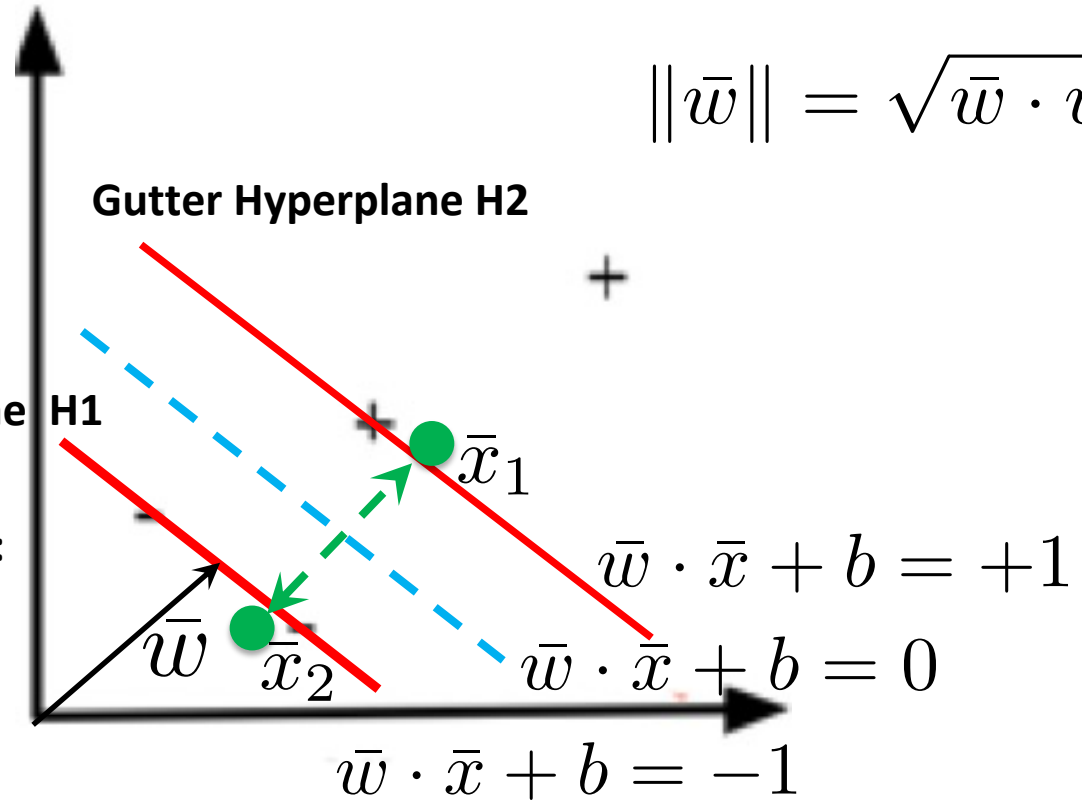


For the 2 red gutter lines, and the blue "median strip" of the road:

$$\bar{w} \cdot \bar{x} + b = +1$$
$$\bar{w} \cdot \bar{x} + b = -1$$
$$\bar{w} \cdot \bar{x} + b = 0$$

Figuring out the street width

$$\|\bar{w}\| = \sqrt{\bar{w} \cdot \bar{w}}$$



Red hyperplane (gutter) lines:

$$\bar{w} \cdot \bar{x}_1 + b = +1$$
$$\bar{w} \cdot \bar{x}_2 + b = -1$$

Subtract:

$$\bar{w} \cdot (\bar{x}_1 - \bar{x}_2) = 2$$

Divide by the length of $\bar{w}$ to get distance (street width) between the gutter lines:

$$\frac{\bar{w} \cdot (\bar{x}_1 - \bar{x}_2)}{\|\bar{w}\|} = \frac{2}{\|\bar{w}\|}$$

This is what we want to maximize (subject to conditions)

Next: How to maximize this street width – now a constrained optimization problem

Maximize Street width $= \dfrac{2}{\|\bar{w}\|}$

Maximize $= \dfrac{1}{\|\bar{w}\|}$

$\Rightarrow$ Minimize $\|\bar{w}\|$

$\Rightarrow$ Minimize $\dfrac{1}{2}\|\bar{w}\|^2$

Subject to constraints:

$y_i(\bar{w} \cdot \bar{x}_i + b) - 1 = 0$

for x_i in gutter

"Maximum street width (margin) classifier"

How to do this?

How to solve?

Optimization under constraints– use Lagrange multiplier method

$$L = \frac{1}{2}\|\bar{w}\|^2 - \sum \alpha_i \left[y_i(\bar{w}\cdot\bar{x}_i) + b) - 1\right]$$

Lagrange multipliers α_i 1 for each example

Must turn out to be nonzero if a support vector

Differentiate L wrt both *w* and *b*, and set to zero…

$$\frac{\partial L}{\partial \bar{w}} = \bar{w} - \sum_i \alpha_i y_i \bar{x}_i = 0 \Rightarrow \boxed{\bar{w} = \sum_i \alpha_i y_i \bar{x}_i}$$

Step #4

Optimization under constraints– use Lagrange multiplier method

Differentiate L wrt *b*, and set to zero…

$$\frac{\partial L}{\partial b} = -\sum_i \alpha_i y_i \Rightarrow \boxed{\sum_i \alpha_i y_i = 0}$$ **Step #5**

$$\sum_i \alpha_i y_i = 0$$

α_i <u>cannot</u> be 0 for a support vector; α_i=0 for an example point iff it's not a support vector. Recall that the outcome variable class y_i is either +1 or –1

SO: $$\sum_{p \in +\text{pts}} \alpha_p \bar{x}_p + \sum_{p \in -\text{pts}} \alpha_p \bar{x}_p = 0$$

Summarizing: the optimization to maximize the street width has found
2 general equations that fix the support vectors α

$$\bar{w} = \sum_i \alpha_i y_i \bar{x}_i$$

$$\sum_i \alpha_i y_i = 0$$

$(\bar{x}_i, y_i)$ are the $(\text{example}, \text{output})$ class pairs

Re-expressing the Lagrangian to gain insight into the system by substituting for $\bar{w}$

$$L = \frac{1}{2}\|\bar{w}\|^2 - \sum \alpha_i \left[y_i(\bar{w}\cdot\bar{x}_i) + b) - 1\right]$$

From Step #5 plug in

$$\bar{w} = \sum_i \alpha_i y_i \bar{x}_i$$

$$L = \frac{1}{2}\left(\sum_i \alpha_i y_i \bar{x}_i\right) \cdot \left(\sum_j \alpha_i y_j \bar{x}_j\right)$$

$$L = \frac{1}{2}\left(\sum_i \alpha_i y_i \bar{x}_i\right) \cdot \left(\sum_j \alpha_i y_j \bar{x}_j\right) - \sum_i \alpha_i y_i \bar{x}_i \cdot \left(\sum_j \alpha_j y_j \bar{x}_j\right)$$

Now we have two squared terms…

Finish plugging in substitution for $\bar{w} = \sum_i \alpha_i y_i \bar{x}_i$

$$L = \frac{1}{2}\|\bar{w}\|^2 - \sum \alpha_i \left[y_i(\bar{w} \cdot \bar{x}_i) + b) - 1\right]$$

constant

$$L = \frac{1}{2}\left(\sum_i \alpha_i y_i \bar{x}_i\right) \cdot \left(\sum_j \alpha_j y_j \bar{x}_j\right) - \sum_i \alpha_i y_i \bar{x}_i \cdot \left(\sum_j \alpha_j y_j \bar{x}_j\right) - \sum_i \alpha_i y_i b + \sum_i \alpha_i$$

$$b \sum_i \alpha_i y_i$$

$$L = \sum_i \alpha_i - \frac{1}{2}\left(\sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{x}_i \cdot \bar{x}_j\right)$$

0 $\sum_i \alpha_i y_i = 0$

THIS is what we maximize (or minimize) – so what??

KEY POINT:

Optimization depends only on dot product of sample pairs!

And, if we substitute for *w* in the decision rule for the classification of unknown points…

$$\bar{w} \cdot \bar{u} + b \geq 0 \text{ then } +$$

$$\bar{w} = \sum_i \alpha_i y_i \bar{x}_i$$

$$\sum_i \alpha_i y_i \bar{x}_i \cdot \bar{u} + b \geq 0 \text{ then } +$$

<u>Recognition</u> of unclassified example *u* also depends <u>only</u> on the dot product of sample vectors and the unknown

So BOTH <u>training</u> and <u>recognition</u> depend only on knowing the dot products output by the kernel function, not the original kernel function itself

Why is dot product so important?

- Learning depends only on dot products of sample pairs
- Recognition depends only on dot products of unknown with samples
- Intuitively, dot product is the intrinsic "distance measure" that says how far "apart" 2 pts are
- Exclusive reliance on dot products enables an approach to problems in which samples cannot be separated by a straight line – as we shall see

Dot products as similarity measures

- If two vectors are parallel (completely alike), their inner product is 1:

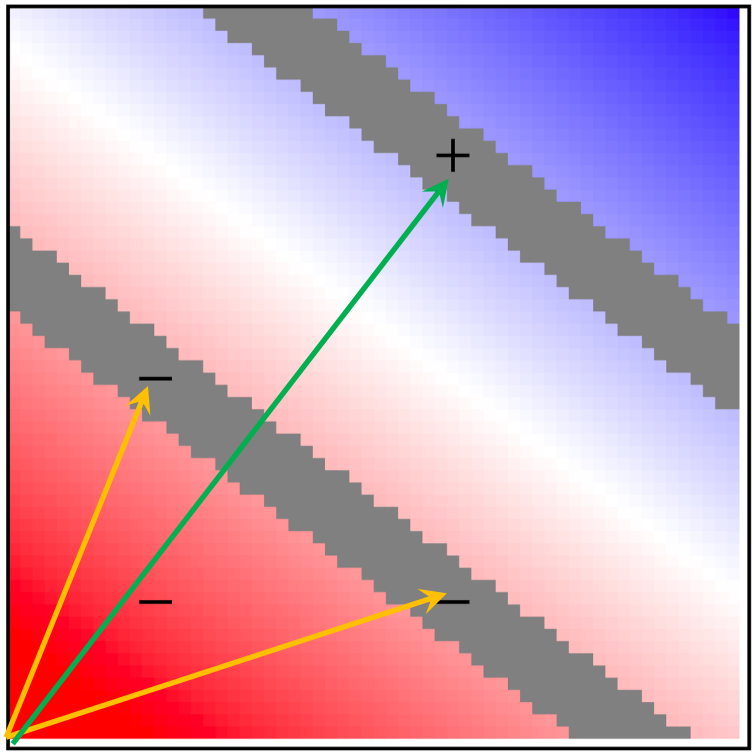
$$x_1 \cdot x_2 = 1$$

- If the 2 vectors are perpendicular (completely unlike), their inner product is 0:

$$x_1 \cdot x_2 = 0$$

12:54:59 EDT 09-Oct-2020

Start | Stop | Pause | Clear | Hide choices



Green:
Support vector
(to + pt on gutter)

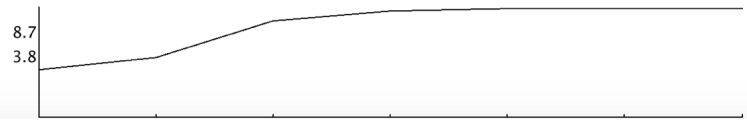
Orange:
Support vectors
(to – pts on gutter)

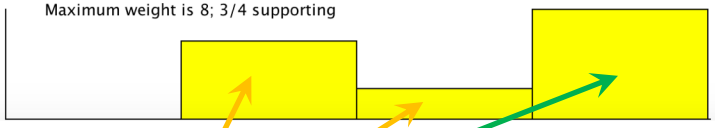
Example
- Electrical covers 1
- Electrical covers 2
- Electrical covers 3
- Electrical covers 4
- Basic test
- Two vertical points
- Two diagonal points
- Admiral's delight
- Rings of Saturn
- Rings of Saturn, LP
- Line
- Linearly inseparable
- Mirror 1
- Mirror 2

Kernal form
- Dot product
- Dot product augmented
- Radial basis function

Kernal: v1 . v2

Delay: 0.0 0.2 0.4 0.6 0.8 1.0

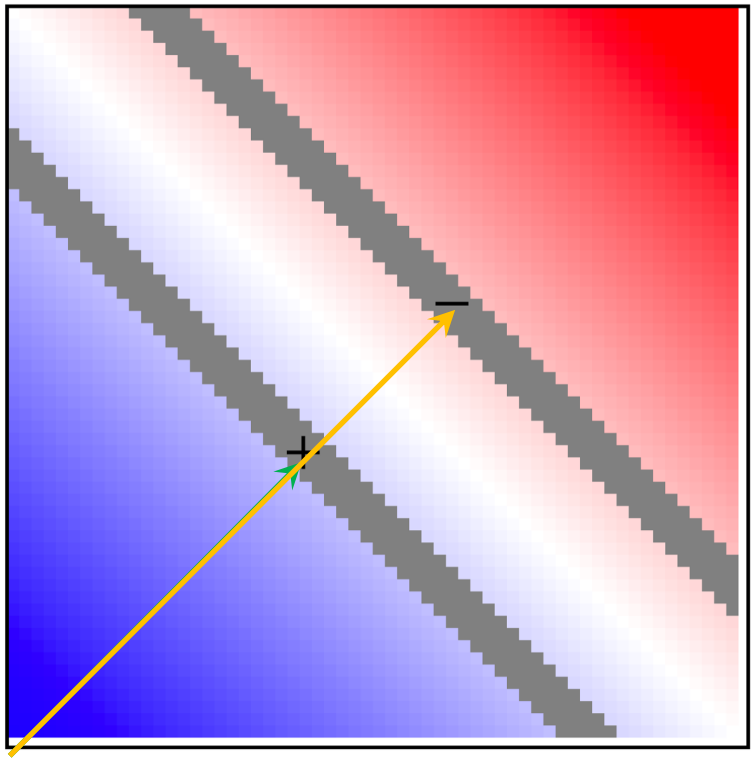




Note the 3 support vectors

13:14:00 EDT 09-Oct-2020

Start Stop Pause Clear Hide choices



Example
- Electrical covers 1
- Electrical covers 2
- Electrical covers 3
- Electrical covers 4
- Basic test
- Two vertical points
- Two diagonal points
- Admiral's delight
- Rings of Saturn
- Rings of Saturn, LP
- Line
- Linearly inseparable
- Mirror 1
- Mirror 2

Kernal form
- Dot product
- Dot product augmented
- Radial basis function

Kernal
v1 . v2

Delay
0.0 0.2 0.4 0.6 0.8 1.0

Maximum weight is 24; 2/2 supporting

Note just 2 support vectors

13:21:56 EDT 09-Oct-2020

Start Stop Pause Clear Hide choices

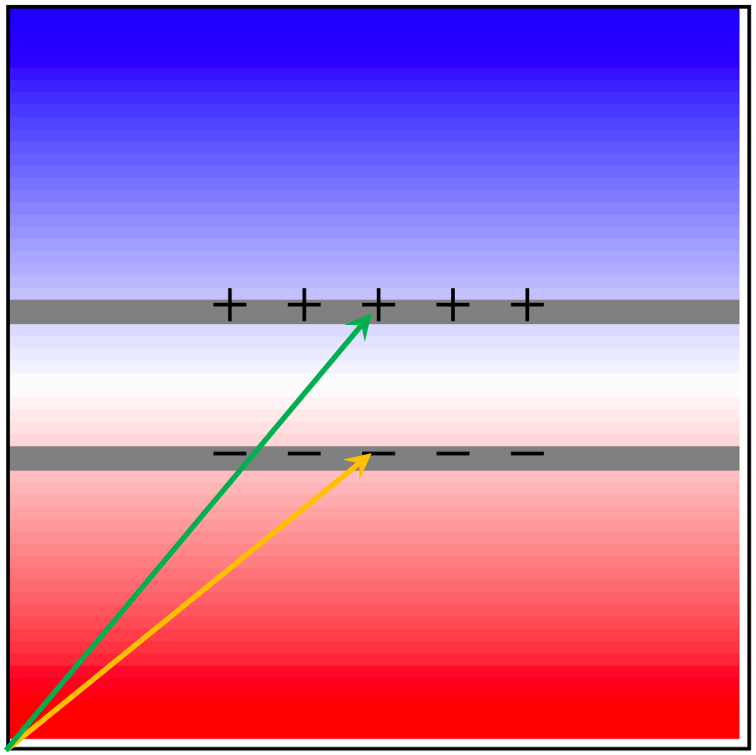
Example
- Electrical covers 1
- Electrical covers 2
- Electrical covers 3
- Electrical covers 4
- Basic test
- Two vertical points
- Two diagonal points
- Admiral's delight
- Rings of Saturn
- Rings of Saturn, LP
- Line
- Linearly inseparable
- Mirror 1
- Mirror 2

Kernal form
- Dot product
- Dot product augmented
- Radial basis function

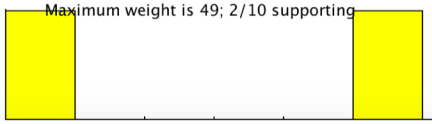
Kernal
v1 . v2

Delay
0.0 0.2 0.4 0.6 0.8 1.0

Here they are

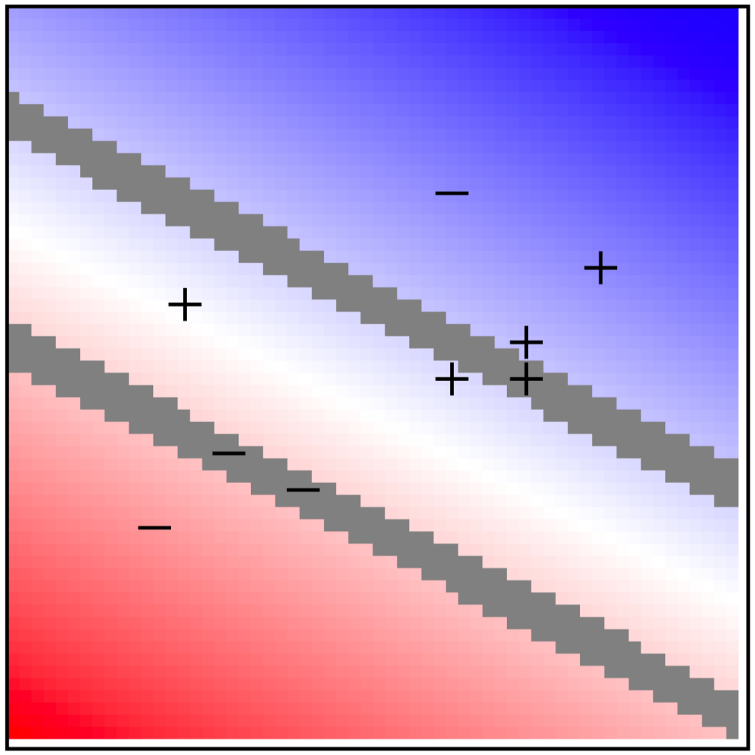


Note 2 support vectors…



13:27:14 EDT 09-Oct-2020

Start | Stop | Pause | Clear | Hide choices

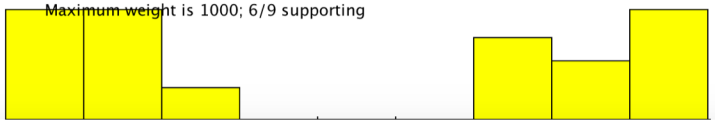


Example
- Electrical covers 1
- Electrical covers 2
- Electrical covers 3
- Electrical covers 4
- Basic test
- Two vertical points
- Two diagonal points
- Admiral's delight
- Rings of Saturn
- Rings of Saturn, LP
- Line
- (•) Linearly inseparable
- Mirror 1
- Mirror 2

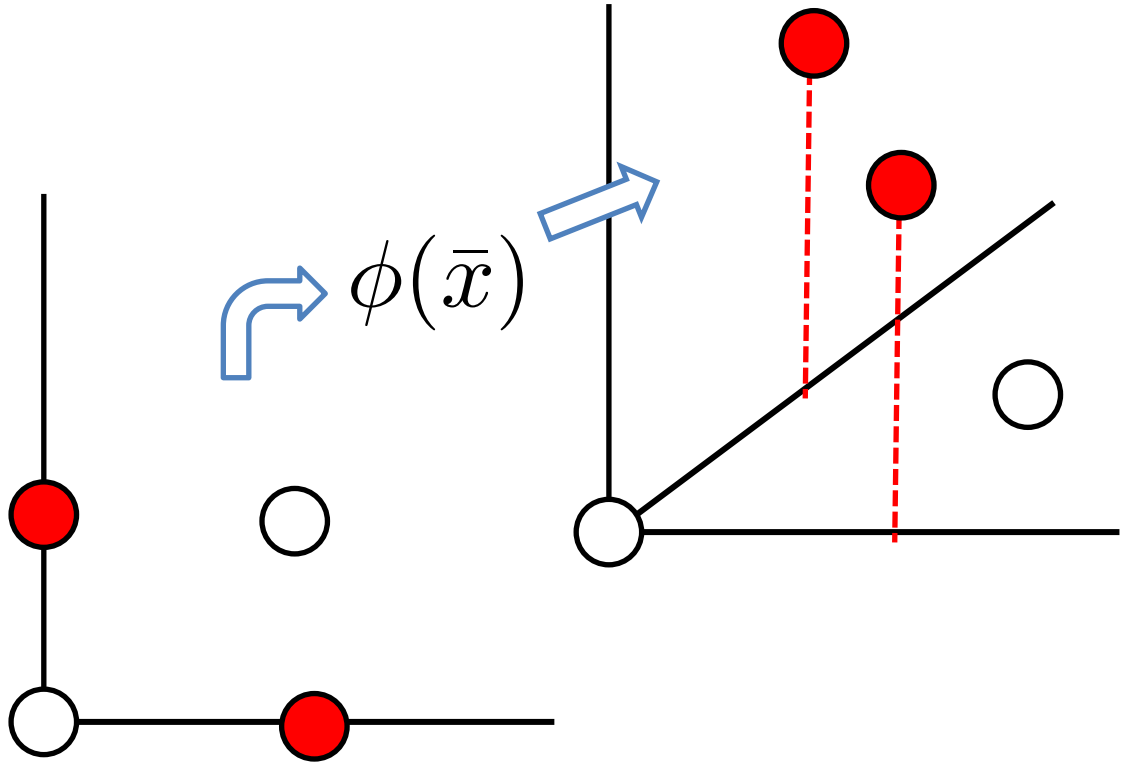
Kernal form
- (•) Dot product
- Dot product augmented
- Radial basis function

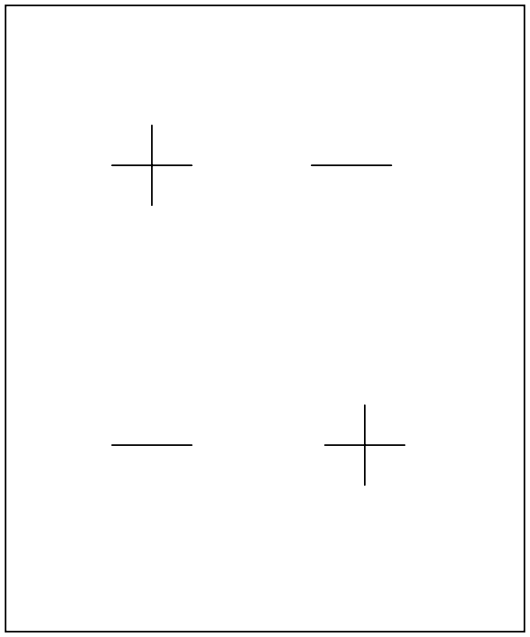
Kernal
v1 . v2

Delay
0.0 0.2 0.4 0.6 0.8 1.0

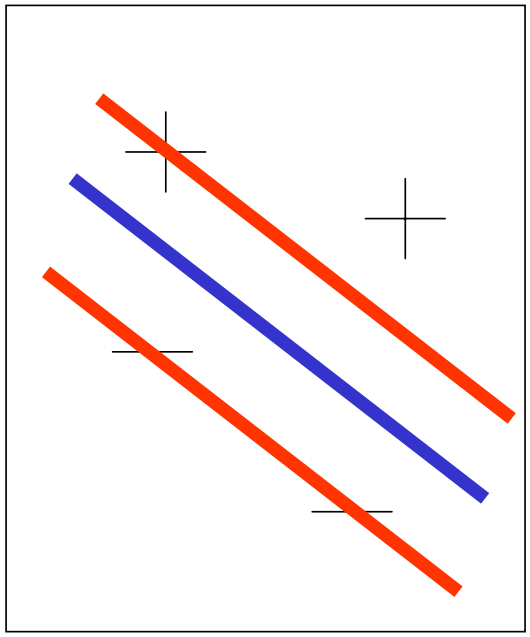


Kernel functions: project into another space where the points are linearly separable…





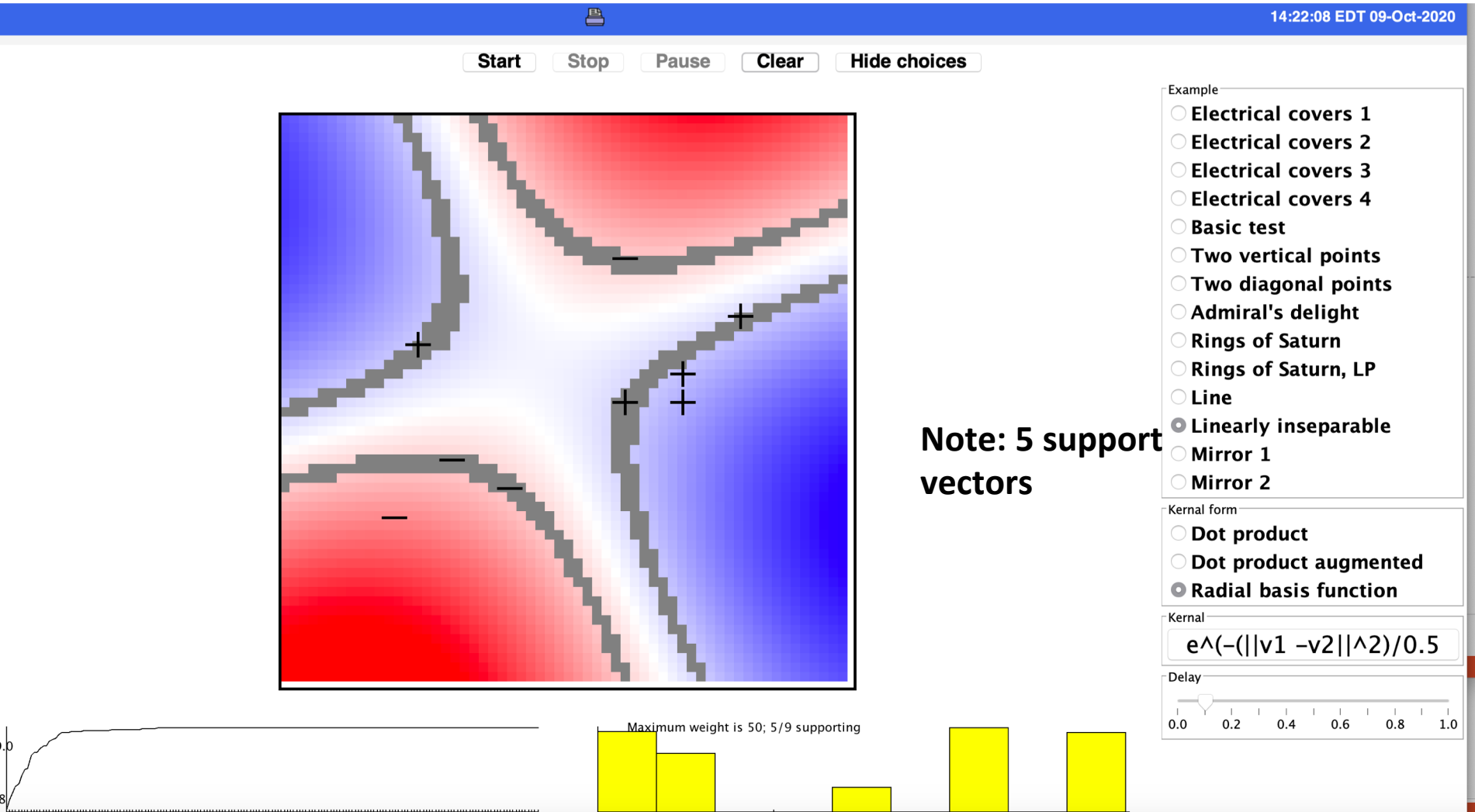
Problem starts here



Dot products computed here

Transforming a nonlinear space to linear by mapping to radial "circles"

$$K(\bar{x}_1, \bar{x}_2) = e^{\frac{-\|\bar{x}_1 - \bar{x}_2\|^2}{\sigma^2}}$$



Note: 5 support vectors

The general "kernel trick": forcing linear separability

#4

Optimization depends only on dot product of sample pairs!

$\phi(\bar{x})$ **Is the required transform… but we need only:**

$$\phi(\bar{x}_1) \cdot \phi(\bar{x}_2) \text{ to maximize}$$

$$\phi(\bar{x}_1) \cdot \phi(\bar{u}) \text{ to recognize}$$

So the only thing we <u>really</u> need to know to solve the optimization & recognition problems is, for any pair of values:

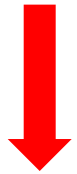
$K(x_1, x_2) = \phi(\bar{x}_1) \cdot \phi(\bar{x}_2)$ **NOT NEEDED:** $\phi(\bar{x})$

The function K is called a <u>kernel</u> <u>Function</u>

To use, we replace dot products in our equations with:

$$\phi(\bar{x}_1) \cdot \phi(\bar{x}_2)$$

$$L = \sum_i \alpha_i - \frac{1}{2}\left(\sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{x}_i \cdot \bar{x}_j\right)$$



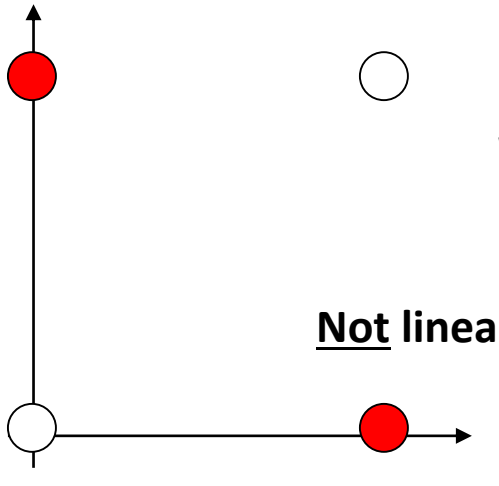
$$L = \sum_i \alpha_i - \frac{1}{2}\left(\sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi(\bar{x}_i) \cdot \phi(\bar{x}_j)\right)$$

What are some popular kernels?

- **No change:** $\phi(\bar{x}_1) \cdot \phi(\bar{x}_2) = K(\bar{x}_1, \bar{x}_2) = \bar{x}_1 \cdot \bar{x}_2$
- **Polynomial:** $K(\bar{x}_1, \bar{x}_2) = (\bar{x}_1 \cdot \bar{x}_2)^n$
- **Radial basis (Gaussian):** $K(\bar{x}_1, \bar{x}_2) = e^{\frac{-\|\bar{x}_1 - \bar{x}_2\|^2}{\sigma^2}}$
- **In general: we want to "separate" points linearly by pushing the original data into another space– can we always do this?**

Solving the exclusive-or (XOR) problem by projection to a higher dimension

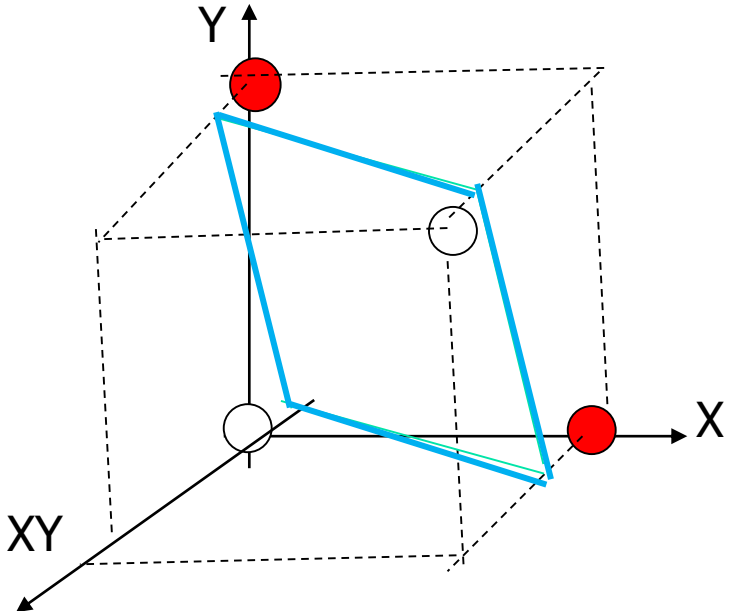
X	Y	Output	
0	0	0	(white)
0	1	1	(red)
1	0	1	(red)
1	1	0	(white)



Red: output 1
White: output 0

Not linearly separable

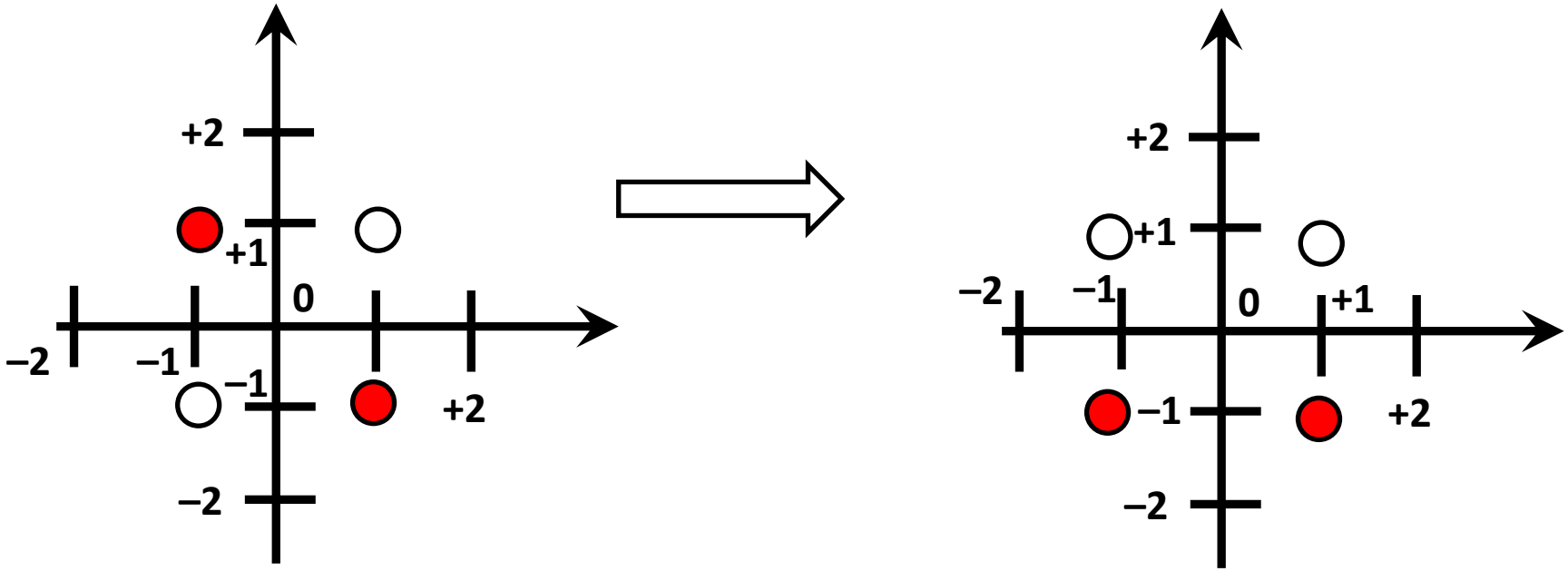
X	Y	XY	Output	
0	0	0	0	(white)
0	1	0	1	(red)
1	0	0	1	(red)
1	1	1	0	(white)



Suggests mapping (x, y, output) to (x, xy, output) (recoding 0 as –)

We can show this x-or separation clearly with a polynomial transformation

$$(x, y, o) \mapsto (x, xy, o)$$



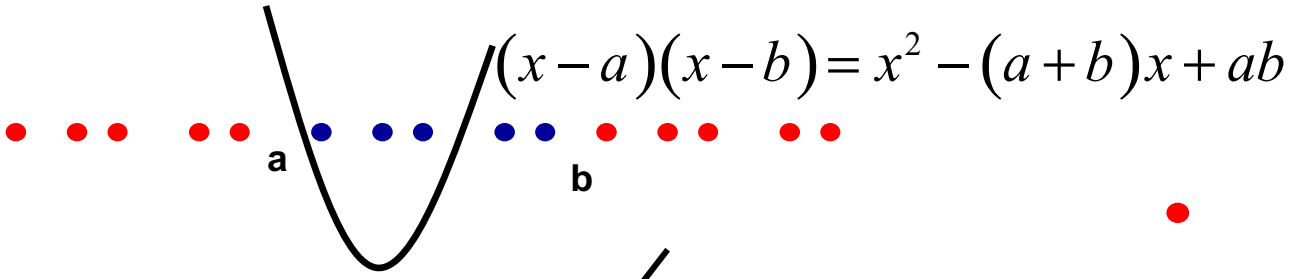
○ = output 0

● = output 1

(–1, –1, 0)↦(–1, +1, 0)
(–1, +1, +1)↦(–1, –1, –1)
(+1, –1, +1)↦(+1, –1, –1)
(+1, +1, 0) ↦ (+1, +1, 0)

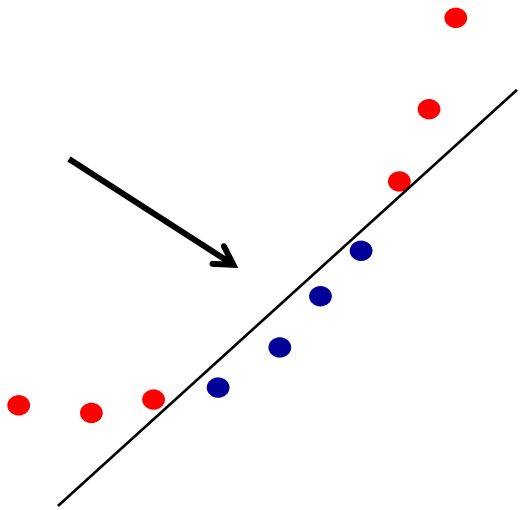
Non–Linear SVMs

- The idea is to gain linear 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

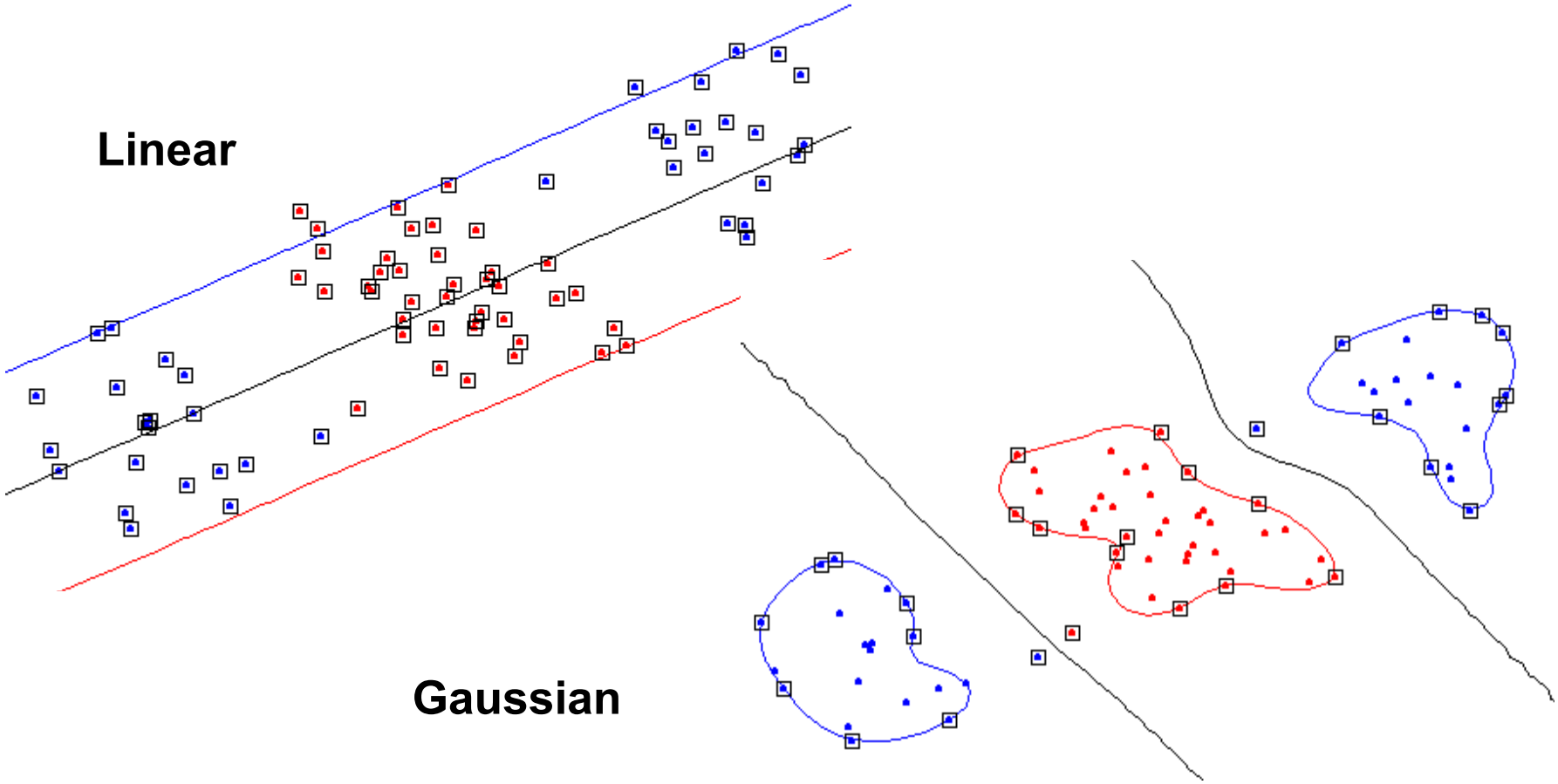


  - So if we map $x \mapsto \{x^2, x\}$

    we gain linear separation



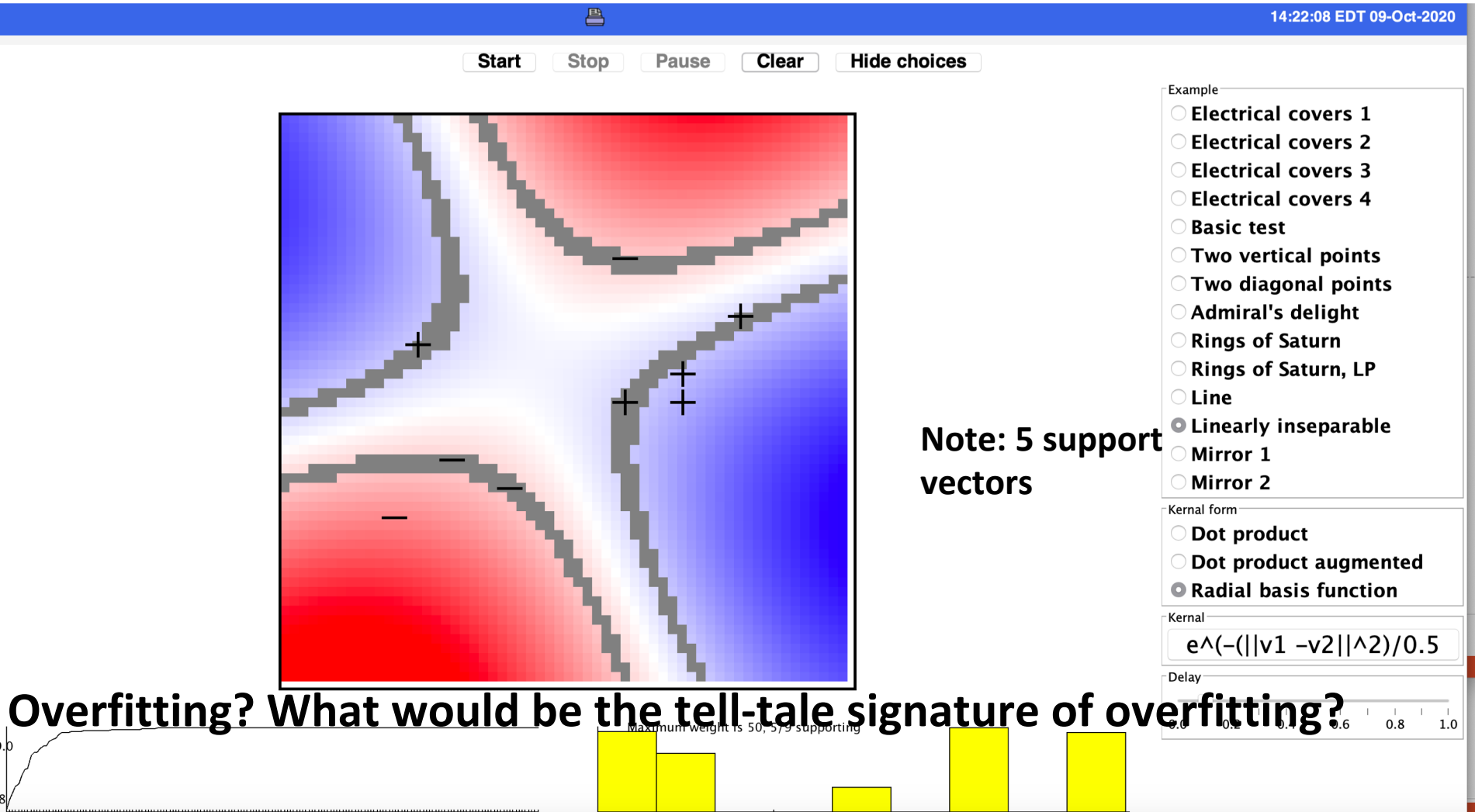
Another example for nonlinear Support Vector Machines – Gaussian or "radial basis function" kernel



What are some popular kernels?

$$K(\bar{x}_1, \bar{x}_2) = e^{\frac{-\|\bar{x}_1 - \bar{x}_2\|^2}{\sigma^2}}$$

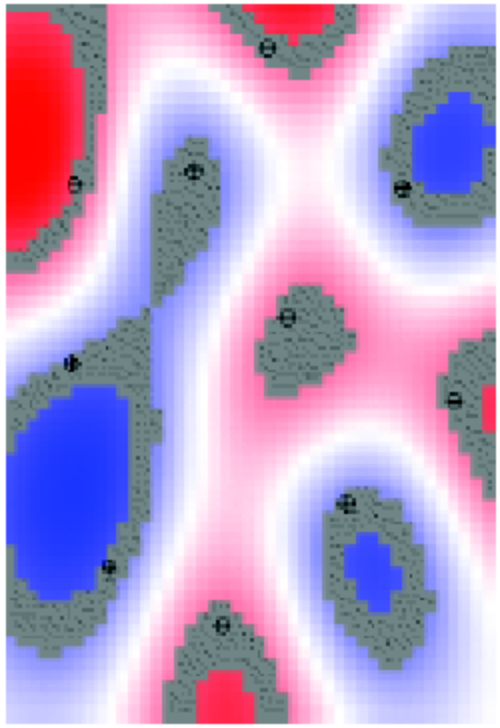
Note: additions of Gaussian kernels in general can map up into an infinite # of dimensions…



Note: 5 support vectors

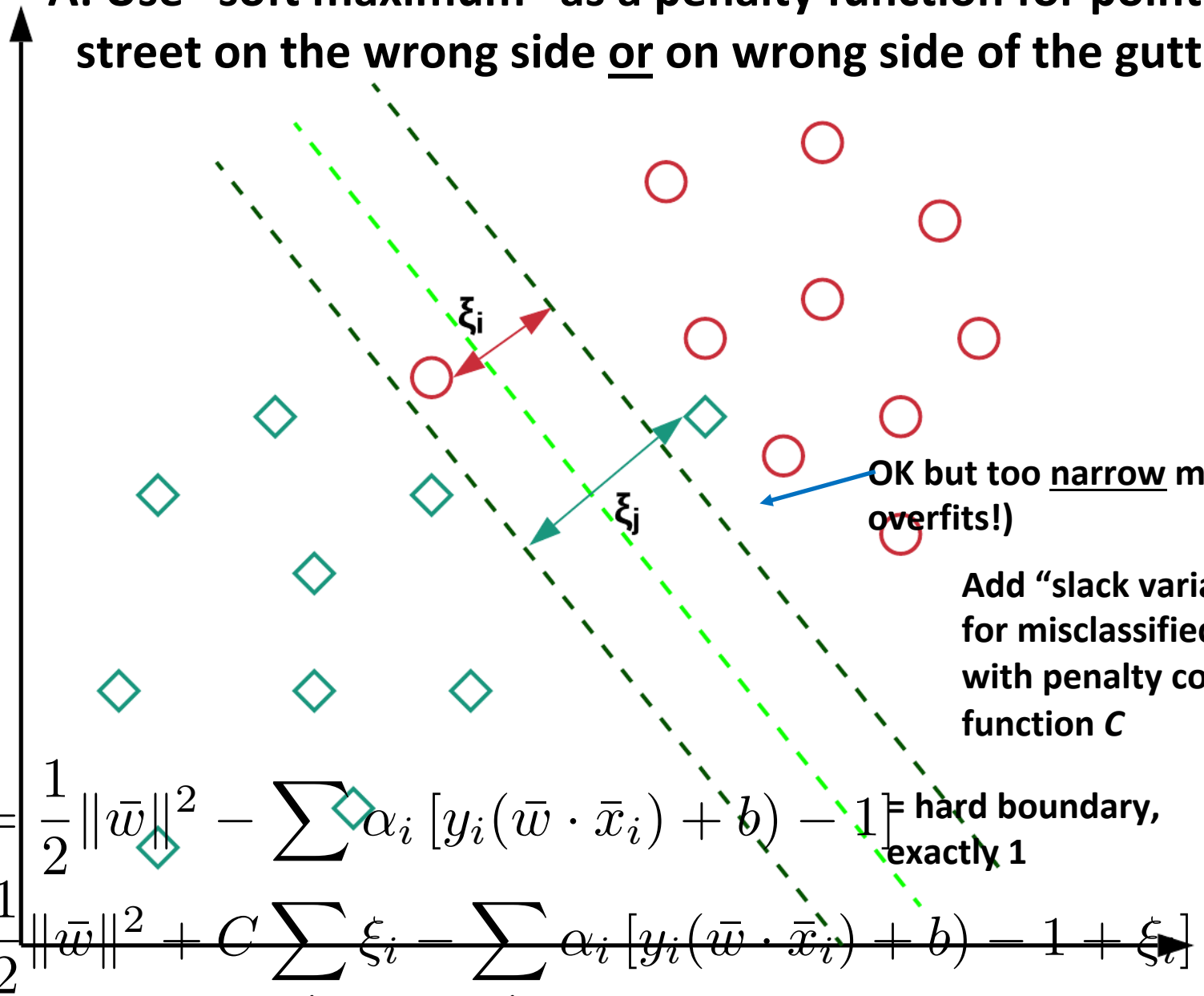
Overfitting? What would be the tell-tale signature of overfitting?

Overfitting by SVM



Q: What about misclassified points?

A: Use "soft maximum" as a penalty function for points in street on the wrong side or on wrong side of the gutter



OK but too narrow margin (it overfits!)

Add "slack variables" ξ for misclassified points with penalty cost function *C*

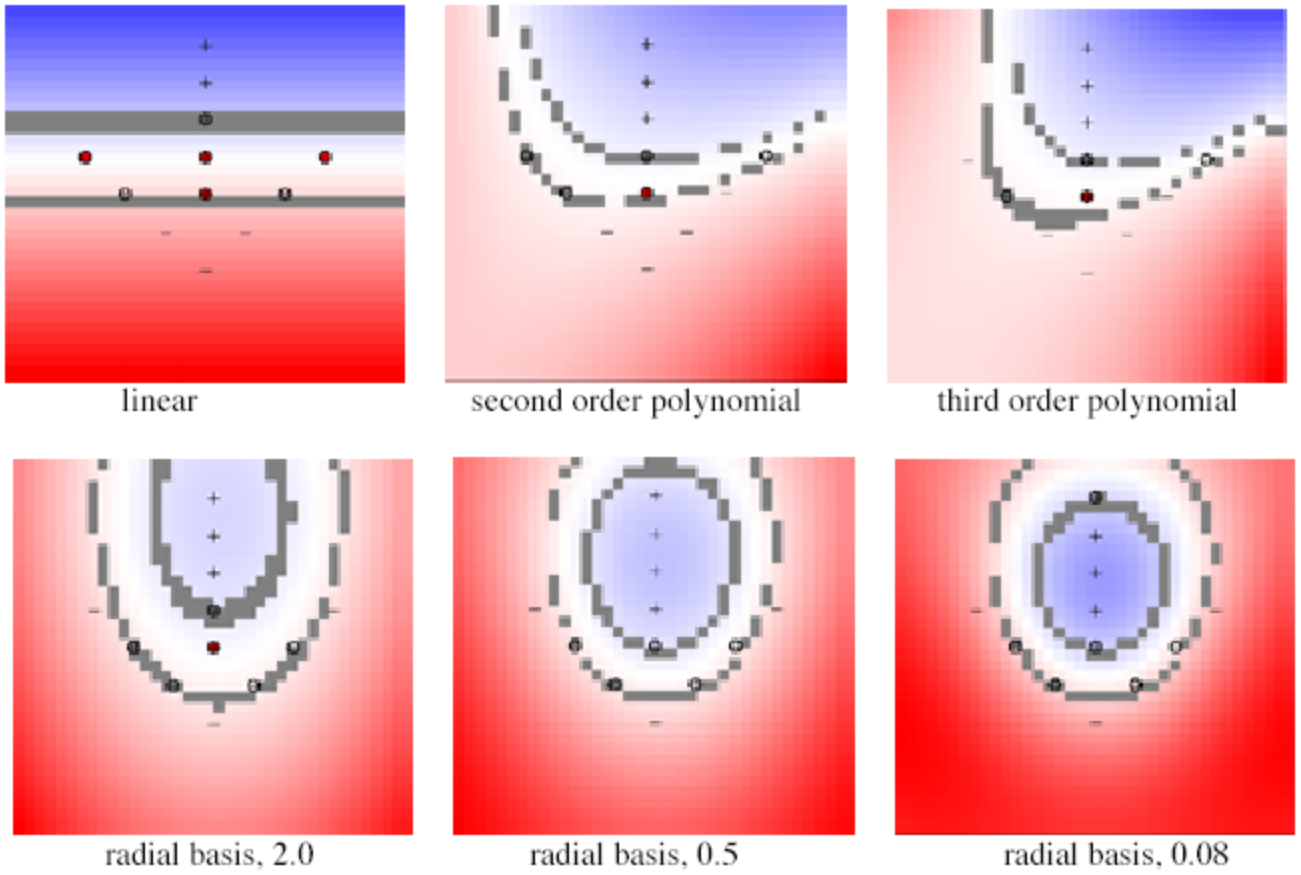
$$L = \frac{1}{2}\|\bar{w}\|^2 - \sum \alpha_i \left[y_i(\bar{w}\cdot\bar{x}_i) + b) - 1\right]$$

= hard boundary, exactly 1

$$L = \frac{1}{2}\|\bar{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i \left[y_i(\bar{w}\cdot\bar{x}_i) + b) - 1 + \xi_i\right]$$

= soft boundary

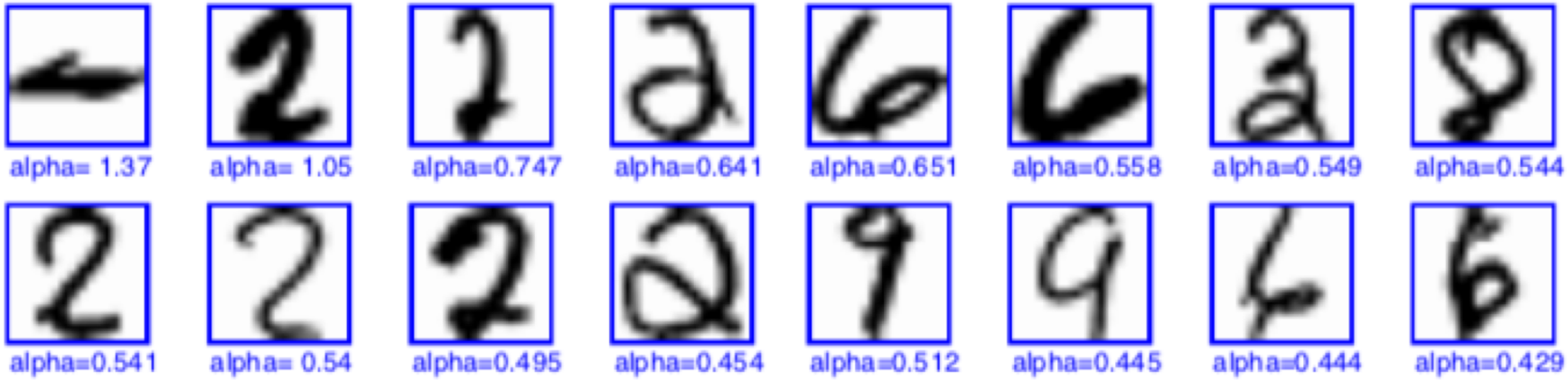
Admiral's delight w/ difft kernel functions



Applications

- Handwritten digits recognition
  - US Postal services
  - 4% error obtained
  - about 4% of the training data were SVs only
- Text categorisation
- Face detection
- DNA analysis

Early competition with neural nets



q	DB1 error	DB1 <m>	DB2 error	DB2 <m>	N
1 (linear)	3.2 %	36	10.5 %	97	256
2	1.5 %	44	5.8 %	89	$3 \cdot 10^4$
3	1.7 %	50	5.2 %	79	$8 \cdot 10^7$
4			4.9 %	72	$4 \cdot 10^9$
5			5.2 %	69	$1 \cdot 10^{12}$