Support Vector Machines (SVMs)

V.S. Subrahmanian
Dartmouth College
vs@dartmouth.edu

Basic Idea, I

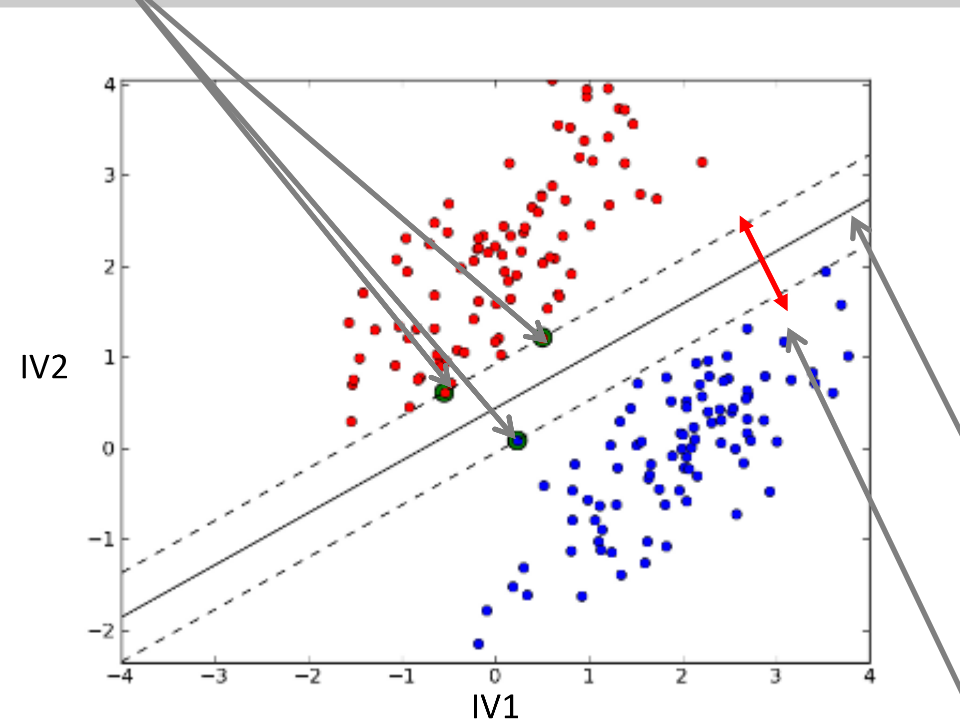
- Looking at 2-class SVMs.
- You have two classes – we will call them -1 and +1.
- You have a set of independent variables.
- You have a table
  - Rows correspond to entities (e.g. people, movies, events, loans)
  - Columns correspond to the IVs and the DV (-1,+1).
- SVM tries to find a "separator" line/formula that separates the rows with DV=-1 from those that have DV=+1.

Basic Idea, II: Geometry

- Suppose we have *k* IVs in all.
- Each row [vector of independent variables] can be thought of as a point in a *k*-dimensional vector space.
- Thus, the values of the IVs determine the location of the point in the *k*-dimensional vector space.
- The *color* of the point denotes whether it is a +1 or a -1, e.g. color=red means +1, color=blue means -1.

Basic Idea,III: Picture

Support vectors



- Consider *k=2.*
- SVM tries to identify a *separator* line that separates the +1's (reds) from the -1's (blues).

Separator line

Picture from:
http://www.mblondel.org/journal/2010/09/19/support-vector-machines-in-python/

Margin: bigger the better.
We usually look for classifiers with the biggest possible margins.
No error so far.

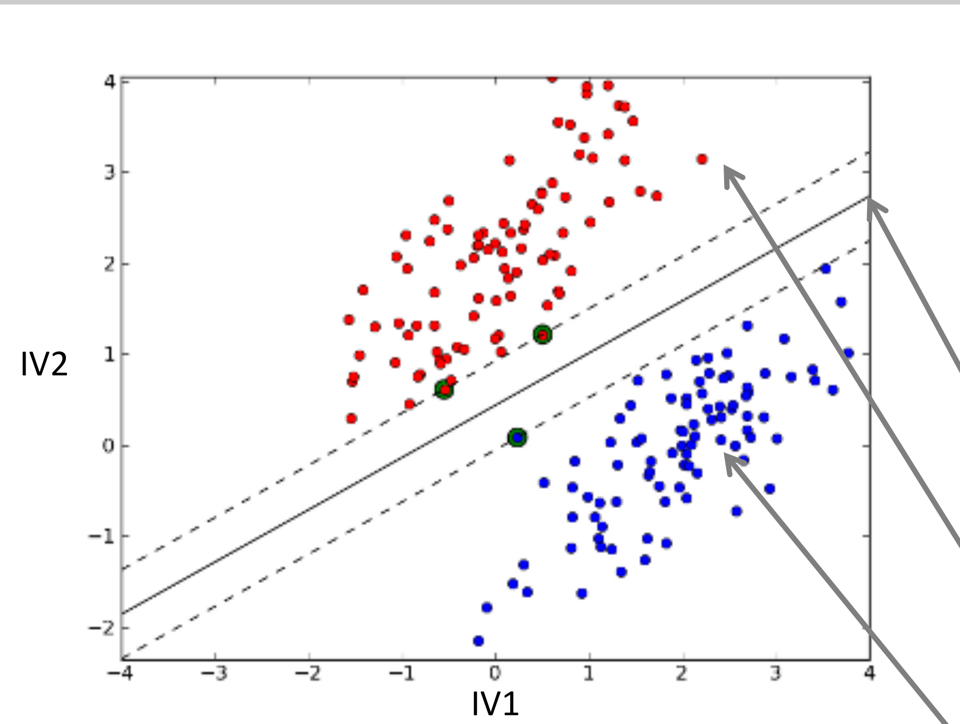
Example: Loans

Name	Education	Education	Salary	Debt	OK
Joe	1	3	80000	200000	1
Mary	1	2	250000	160000	1
Jim	1	1	40000	872000	-1
Tina	0	2	86000	40000	1
Ed	0	3	400000	20000	-1
Lisa	0	1	69000	76000	-1

IVs map onto a 4-dimensional space.

DV

Back to 2-d



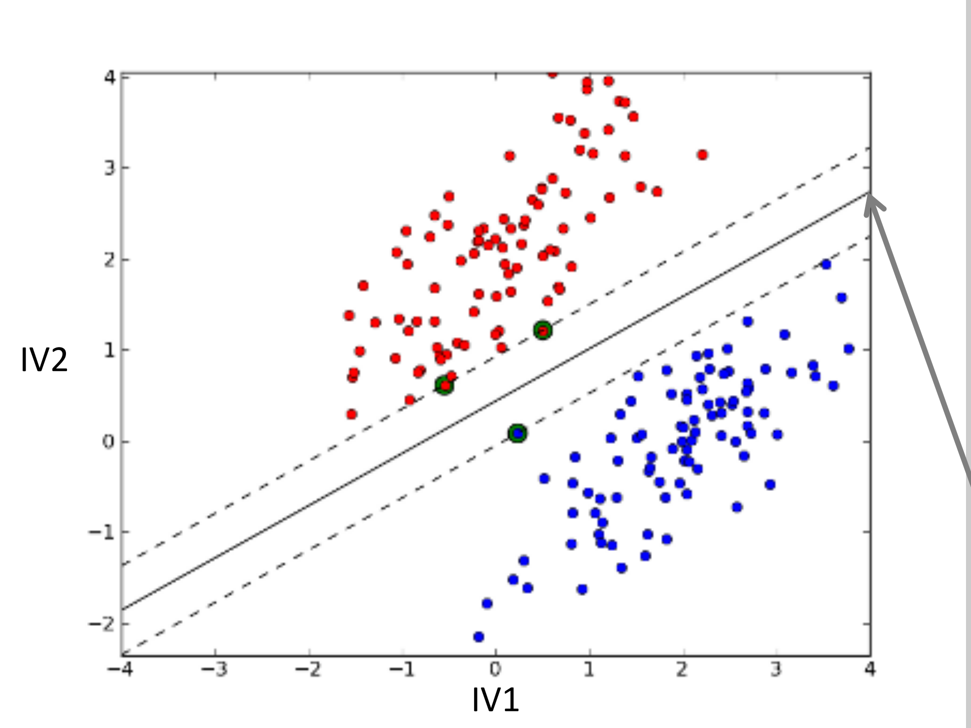
- Consider *k=2.*
- SVM tries to identify a *separator* line that separates the +1's (reds) from the -1's (blues).

Separator line equation in 2-d is *y=mx + c*

Everything in this region satisfies the constraint y > *mx + c*

Everything in this region satisfies the constraint y < *mx + c*

Back to 2-d



- Consider *k=2.*
- SVM tries to identify a *separator* line that separates the +1's (reds) from the -1's (blues).

- In n-dimensions, the equation *mx + c = 0* is replaced by the equation $\mathbf{w}.\boldsymbol{x} + c = 0$ where $\mathbf{w}, \boldsymbol{x}$ are vectors and the "." is dot product.
- This is a "hyperplane" and it also divides the *n*-dimensional space into 2.

Next few slides deal with the "Separable" case where the assumption is that a separator hyperplane exists.

Generalization to *k*-dimensions

- Now, you have a vector $\vec{x}$ of dimensionality *k*.
- Separator line equation is $\mathbf{w}.\boldsymbol{x} + c = 0$. [generalizes 2-d case]
- Here $\mathbf{w}$ is an unknown weight vector that assigns a weight to each of the dimensions in $\boldsymbol{x}$.
- Upper boundary is the equation $\mathbf{w}.\boldsymbol{x} + c = +1$.
- Lower boundary is the equation $\mathbf{w}.\boldsymbol{x} + c = -1$.
- We need to "learn" the weight vector $\mathbf{w}$ and *c* from the data.
- Plan: for a non-training point **z**, predict
  - +1 if $\mathbf{w}.\boldsymbol{z} + c \geq +1$
  - -1 if $\mathbf{w}.\boldsymbol{x} + c \leq -1$

Generalization to *k*-dimensions

- Suppose a new vector $\boldsymbol{z}$ of dimensionality *k* comes in and needs to be classified.
- In reality, $\boldsymbol{z}$ may lie between the two boundaries.
- In this case, compute

$$\boldsymbol{w}.\boldsymbol{z} + c$$

- If $\boldsymbol{w}.\boldsymbol{z} + c \geq 0$ [+1], then classify as "+1".
- If$\boldsymbol{w}.\boldsymbol{z} + c < 0$ $[-1]$, then classify as a "-1".
- The red +1's are used in training, the 0's are used when the classifier is deployed.

How to find the separator line

- Suppose you have *n* training feature vectors $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$.
- Suppose $t_i \in \{-1, +1\}$ denotes the classification of $\overrightarrow{x_i}$.
- Write down the constraint:

  $t_i(\boldsymbol{w}.\boldsymbol{x_i} + c) \geq 1$ for each i=1,..,n.
- When $t_i = +1$, this constraint is satisfied iff $\boldsymbol{w}.\boldsymbol{x_i} + c \geq +1$ which allows us to classify $\overrightarrow{x_i}$ as +1.
- When $t_i = -1$, this constraint is satisfied iff $\boldsymbol{w}.\boldsymbol{x_i} + c \leq -1$ which allows us to classify $\overrightarrow{x_i}$ as -1.

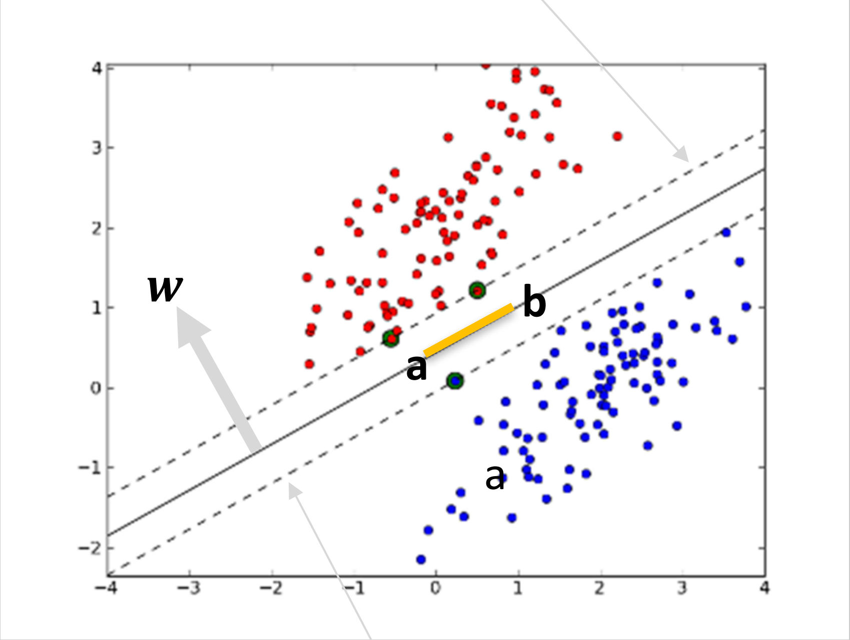
How to find the separator line, II

- Choose w, b so that the distance between the separator and the nearest point is maximized.
- The margin is $\frac{1}{\|\vec{w}\|}$ on each side of the separator, so total margin is $\frac{2}{\|\vec{w}\|}$
- $\|\vec{w}\|^2 = w_1^2 + \cdots + w_k^2$ where $w = (w_1, .., w_k)$
- Maximizing $\frac{2}{\|\vec{w}\|}$ subject to a set C of constraints is same as minimizing either $\|\boldsymbol{w}\|$ or $\|\boldsymbol{w}\|^{\mathbf{2}}$ subject to the same set of constraints.

Why is the margin $\frac{2}{\|\vec{w}\|}$?

- Any point in the separator line satisfies the equation $\boldsymbol{w}.\boldsymbol{x} + c = 0$.
- So if we have two points $\boldsymbol{a}, \boldsymbol{b}$ on the separator line:
- $\boldsymbol{w}.\boldsymbol{a} + c = 0$
- $\boldsymbol{w}.\boldsymbol{b} + c = 0$.
- Subtracting: $\boldsymbol{w}.(\boldsymbol{a} - \boldsymbol{b})$= 0.
- $(\vec{a} - \vec{b})$ is parallel to the separator, so $\boldsymbol{w}$ must be perpendicular as shown.

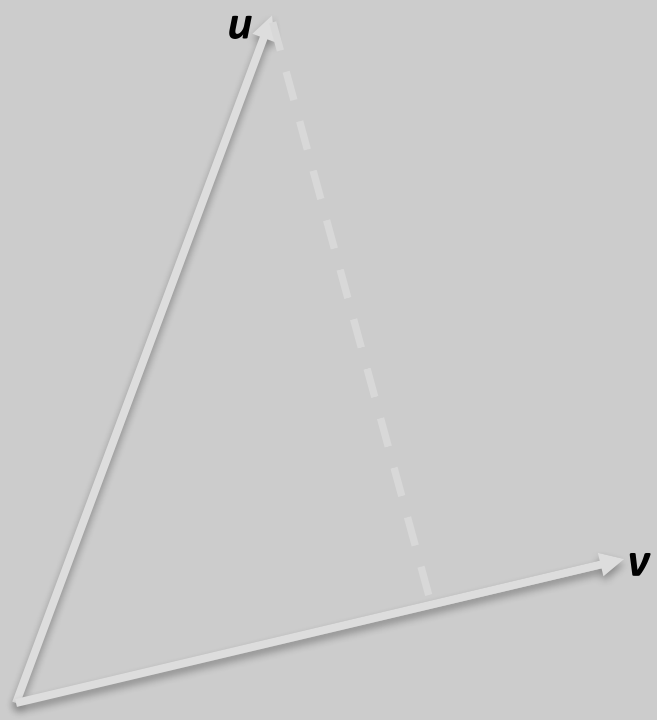
$w^T(\vec{x_i}) + c = 1$



$w^T(\vec{x_i}) + c = -1$

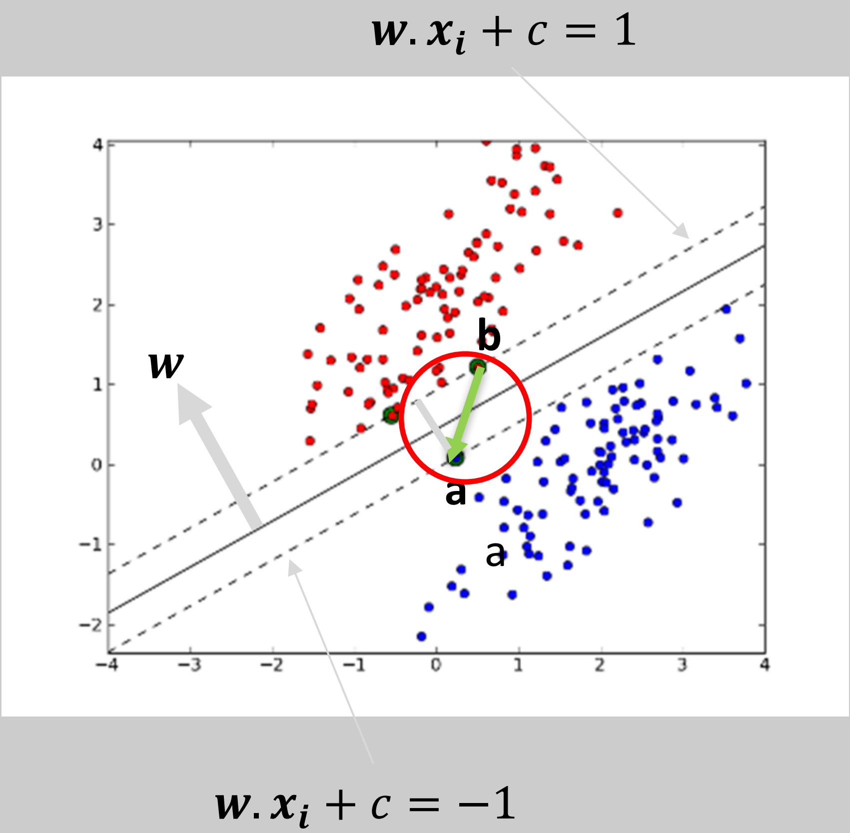
Why is the margin $\frac{2}{\|\vec{w}\|}$?

- In general, if we have two vectors ***u*** **,** ***v*** then
- The length of the perpendicular on ***v*** is $\boldsymbol{u} * \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}$



Why is the margin $\frac{2}{\|\vec{w}\|}$?

- Now suppose $\boldsymbol{a}, \boldsymbol{b}$ are support vectors as shown.
- $\boldsymbol{w}.\boldsymbol{a} + c = -1$
- $\boldsymbol{w}.\boldsymbol{b} + c = +1.$
- Subtracting 1st from the second: $\boldsymbol{w}.(\boldsymbol{b} - \boldsymbol{a})$= +2.
- By standard linear algebra, we now have two vectors:
  - $(\boldsymbol{b} - \boldsymbol{a})$ shown in green
  - $\boldsymbol{w}$ shown in grey.
- Distance is $\frac{\boldsymbol{w}.(\boldsymbol{b}-\boldsymbol{a})}{\|\boldsymbol{w}\|}$
- As $\boldsymbol{w}(\boldsymbol{b} - \boldsymbol{a})$= 2, distance is $\frac{2}{\|\boldsymbol{w}\|}$



Maximizing margins

- We want to maximize the margins.
- That is, we want to solve the optimization problem

Maximize $\frac{2}{||\boldsymbol{w}||}$

Subject to $t_i(\boldsymbol{w}.\boldsymbol{x} + c) \geq 1$ for each training point $(\overrightarrow{x_i}, t_i)$ for i=1,..,n.

- This maximization is the same as minimizing either $||\boldsymbol{w}||$ or $||\boldsymbol{w}||^2$ subject to the same constraints.

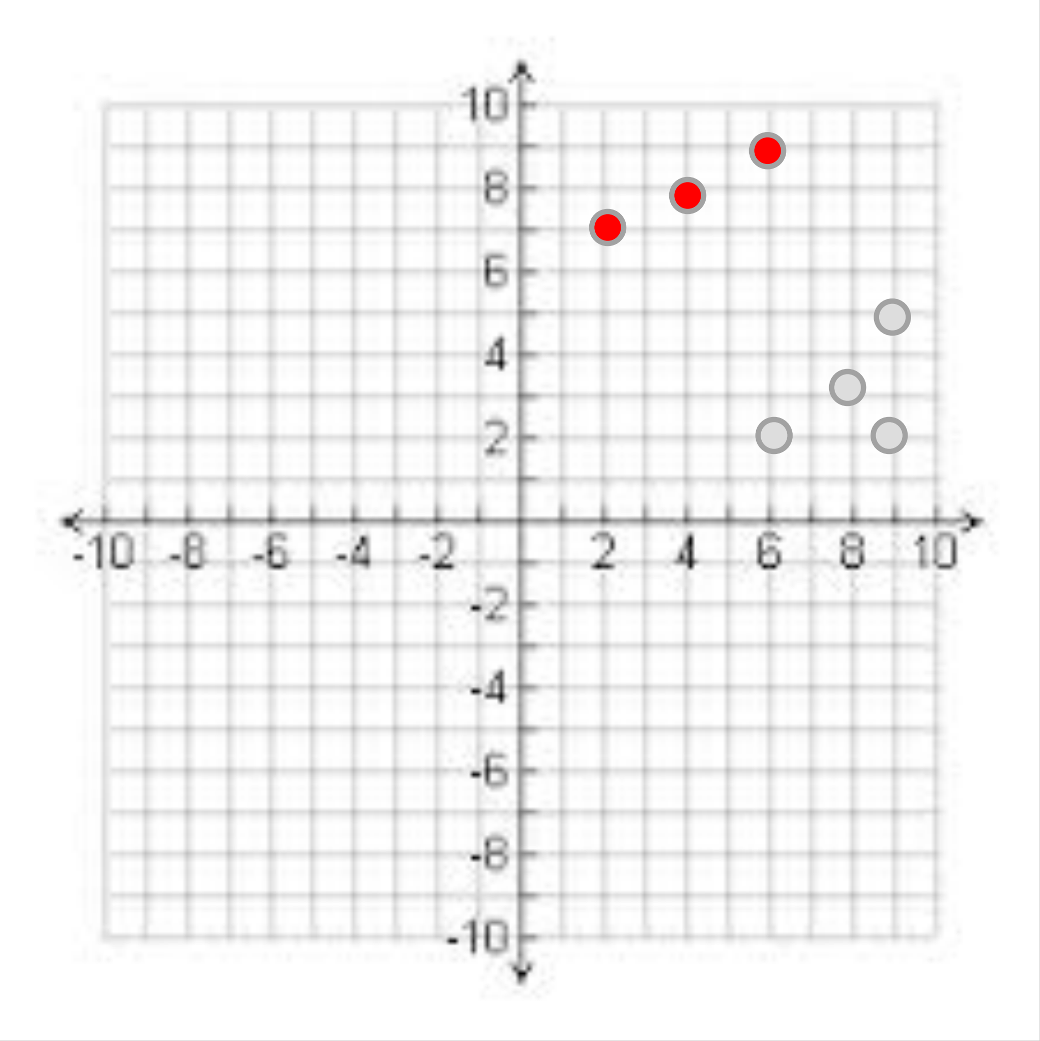
How to find a good separator line

Minimize $\|\boldsymbol{w}\|^2$

Subject to $\mathrm{t_i} * (\boldsymbol{w}.\boldsymbol{x_i} + c) \geq 1$ for each i=1,..,n.

- Here the set of constraints is linear
- But the objective function is *quadratic.*
- ***Can be solved by several classical algorithms in mathematics called***
  - ***Quadratic Programming Algorithms***
  - ***Gradient Descent***

Example in 2-d

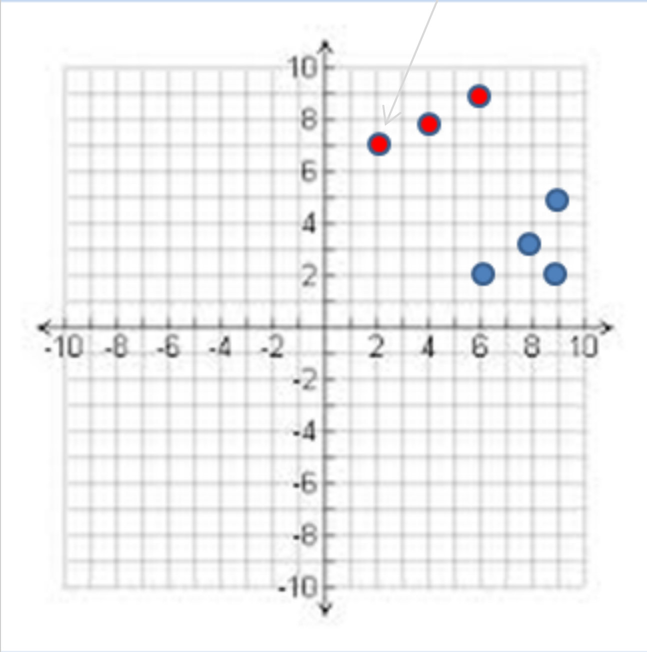


Example

Let $w = (w_1, w_2)$

Constraints: Point (2,7)

$1 * [(w_1, w_2) * (2,7) \ + c] \geq 1$

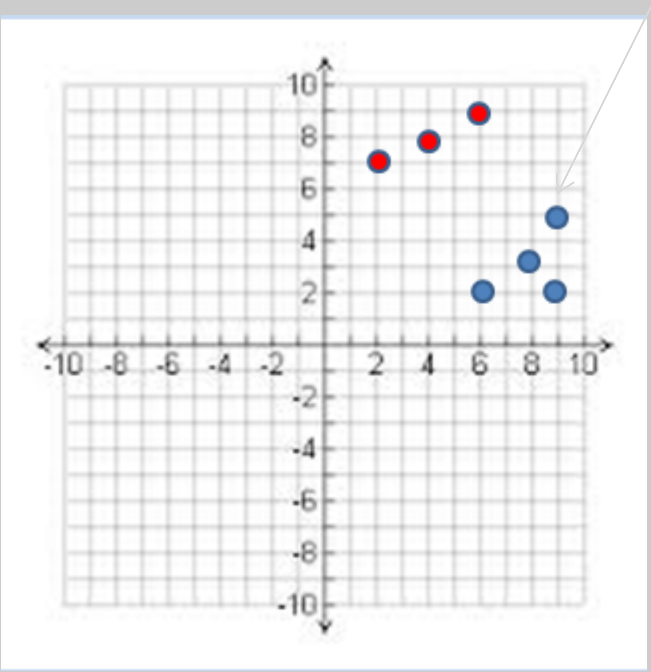


Example

Let $w = (w_1, w_2)$

Constraints: Add (9,5)

$$1 * [(w_1, w_2) * (2,7) \ + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (95) \ + c] \ \geq 1$$

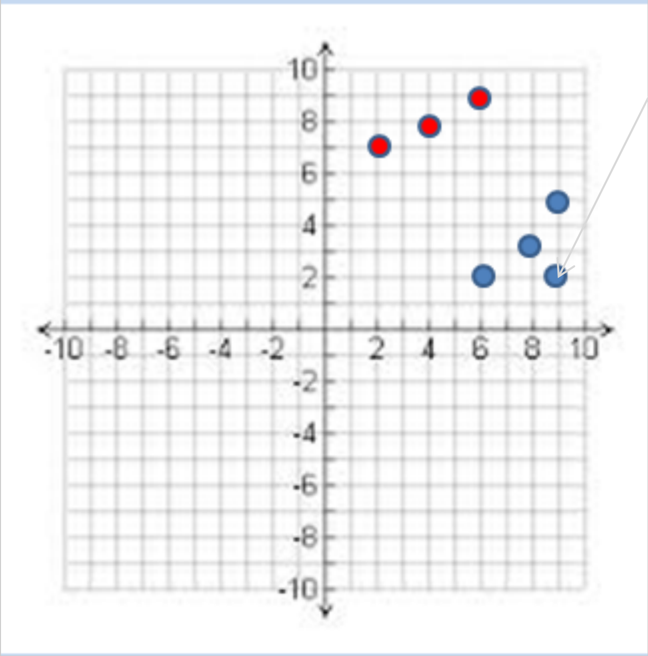


Example

Let $w = (w_1, w_2)$

Constraints: Add (9,2)

$$1 * [(w_1, w_2) * (2{,}7) \ + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (95) \ + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (9{,}2) + c] \ \geq 1$$

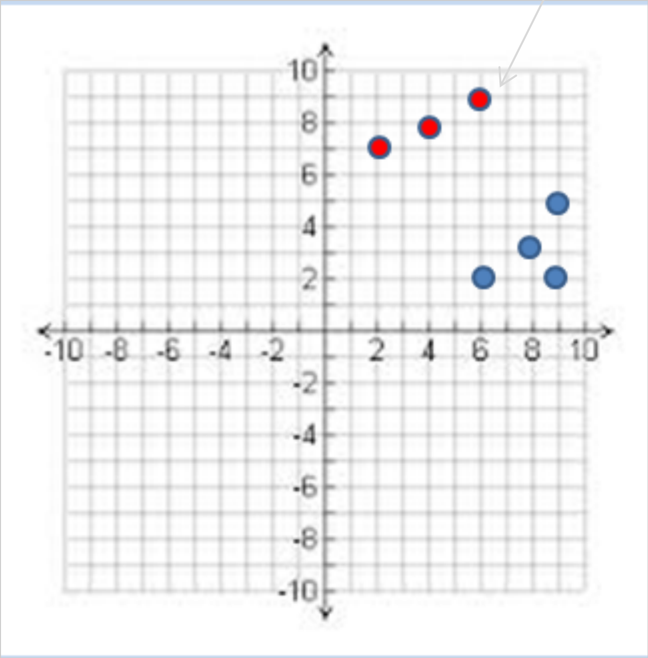


Example

Let $w = (w_1, w_2)$

Constraints: Add (6,9)

$$1 * [(w_1, w_2) * (2,7) \ + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (95) \ + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (9,2) + c] \ \geq 1$$
$$1 * [(w_1, w_2) * (6,9) + c] \ \geq 1$$



Example

Let $w = (w_1, w_2)$
Constraints: Add (4,8),(8,3),(6,2):

$$1 * [(w_1, w_2) * (2,7) \ + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (95) \ + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (9,2) + c] \ \geq 1$$
$$1 * [(w_1, w_2) * (6,9) + c] \ \geq 1$$
$$1 * [(w_1, w_2) * (4,8) + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (8,3) + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (6,2) + c] \ \geq 1$$

To find the best solution

Minimize $\|\boldsymbol{w}\|^2$

Subject to:

$$1 * [(w_1, w_2) * (2,7) \ + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (95) \ + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (9,2) + c] \ \geq 1$$
$$1 * [(w_1, w_2) * (6,9) + c] \ \geq 1$$
$$1 * [(w_1, w_2) * (4,8) + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (8,3) + c] \ \geq 1$$
$$-1 * [(w_1, w_2) * (6,2) + c] \ \geq 1$$

- Recall that the norm of a vector, i.e. $\|\boldsymbol{w}\|^2$ where $w = (w_1, w_2)$ is given by $\|\boldsymbol{w}\|^2 = \boldsymbol{w_1^2} + \boldsymbol{w_2^2}$.
- So the optimization problem on the left has 3 unknowns and 7 constraints and hence (hopefully) can be solved.

In the next few slides, we remove the assumption that a separator hyperplane exists.

Variant

- **Preceding slides assume that:**
  - **a separator of the kind shown exists which may not true**
  - **allow for no error, i.e. no misclassifications.**
- SVM's quadratic programming formulation can be redone so as to avoid these two problems.
- How?

How to find a good separator line when misclassification is allowed

Minimize $\|w\|^2 + \boldsymbol{\tau} * \sum_i \boldsymbol{e_i}$

Subject to $\mathrm{t_i} * (\boldsymbol{w}.\boldsymbol{x_i} + c) \geq 1 - e_i$ for each i=1,..,n.

- $\boldsymbol{\tau}$ is a factor that captures error tolerance. Big value of $\boldsymbol{\tau}$ suggests that we have less tolerance for error.
- e_i is the distance between a misclassified point and the separator line.
- ***Can be solved by a classical set of algorithms in mathematics called Quadratic Programming Algorithms => We will not go into this in the class.***

Hyper-parameter Optimization:Effects of τ

- When τ is small:
  - SVM pays less attention to points near the decision boundary.
  - Margin goes up.
- When is large:
  - SVM pays a high penalty when there are points near the decision boundary.
  - So margin goes down.

Minimize $\|\boldsymbol{w}\|^2 + \boldsymbol{\tau} * \sum_i \boldsymbol{e_i}$
Subject to $\mathrm{t_i} * (\boldsymbol{w}.\boldsymbol{x_i} + c) \geq 1 - e_i$
for each i=1,..,n.

With Misclassification Error

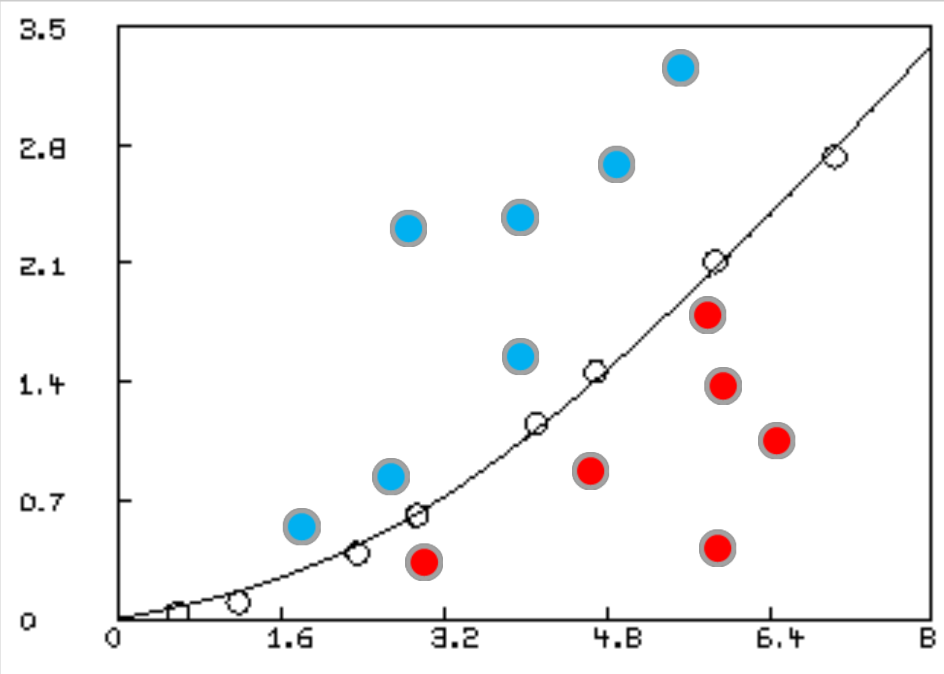
Minimize $\|\boldsymbol{w}\|^2 + \boldsymbol{\tau} * \sum_i \boldsymbol{e_i}$

Subject to $\mathrm{t_i} * (\boldsymbol{w}.\boldsymbol{x_i} + c) \geq 1 - e_i$ for each i=1,..,n.

- $\boldsymbol{\tau}$ is a factor that captures error tolerance. Big value of $\boldsymbol{\tau}$ suggests that we have less tolerance for error.
- e_i is an error term.
- ***Can be solved by a classical set of algorithms in mathematics called Quadratic Programming Algorithms => We will not go into this in the class.***

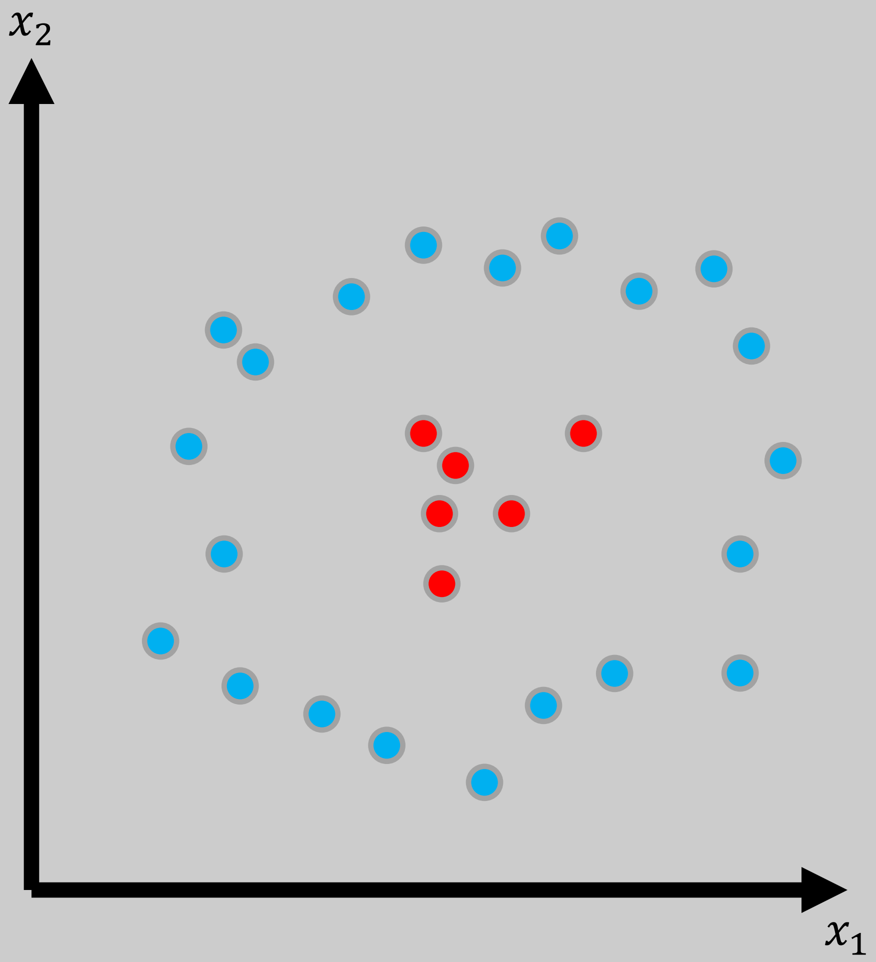
Non-Linear Separators

- Thus far, we have looked at SVM with *linear separators*, i.e. the separating hyper-plane is linear.
- Can do the same thing with *non-linear* (e.g. quadratic) separators.
  - Optimization problem can be written similarly but the resulting optimization problem may no longer be quadratic
  - Distance computations may get more complex



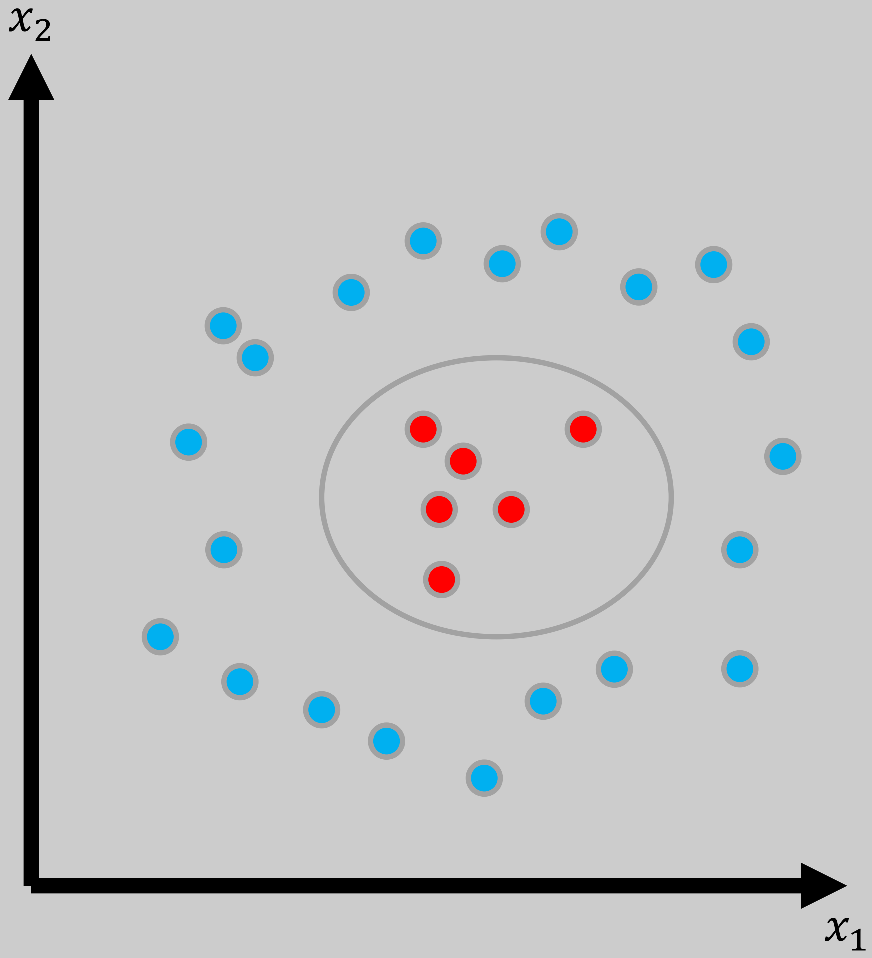
SAMPLE QUADRATIC CURVE FROM:
http://www.civilized.com/mlabexamples/multsitebind.htmld/
Point placement by me!

What if we have the following situation?



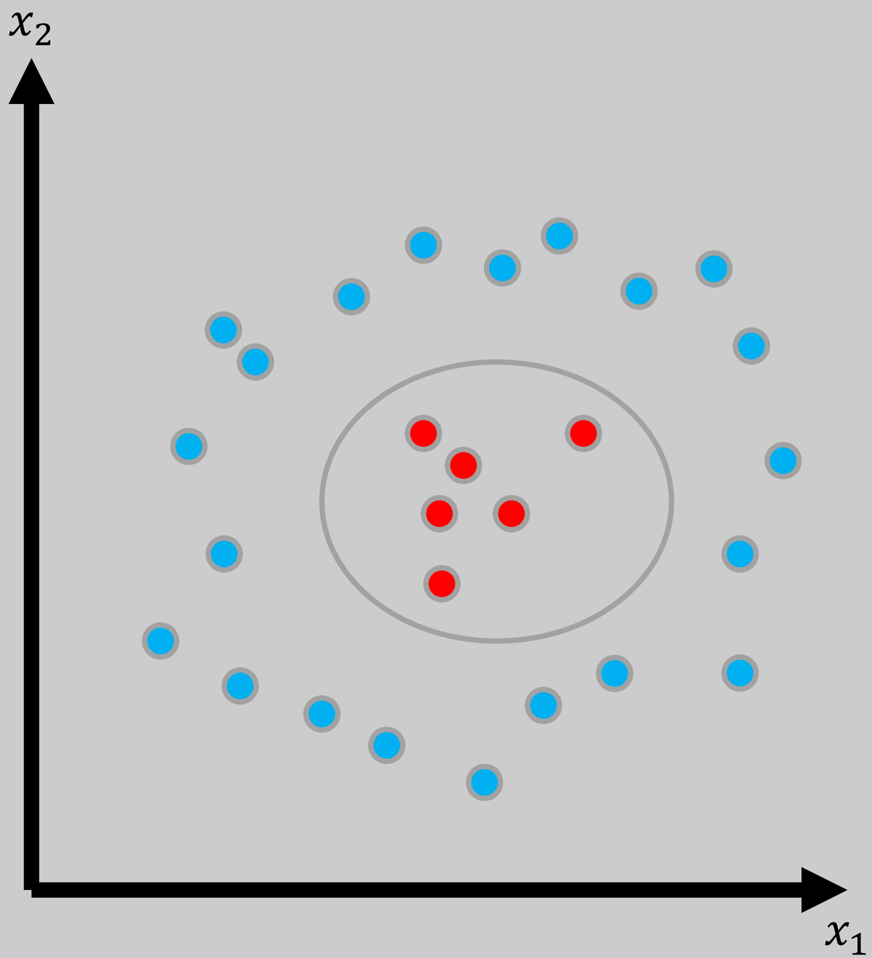
Will a linear separator work? What about a polynomial separator?

What if we have the following situation?



But this is a good separator!

Kernels Help

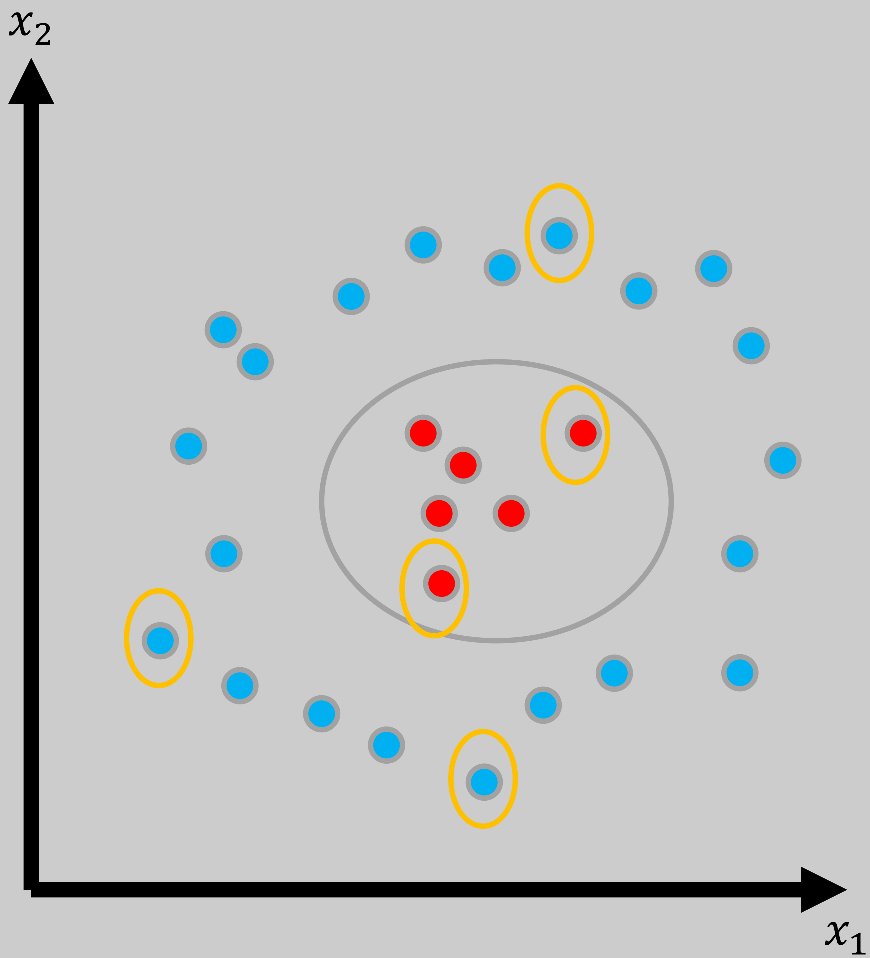


How do Kernels Work?

1. Map training points into a new "feature" space.

2. How?
   a. Associate a set of "landmark" points in the space.
   b. Compute <u>similarity</u> between training points and landmarks
   c. Use these similarities to derive features.

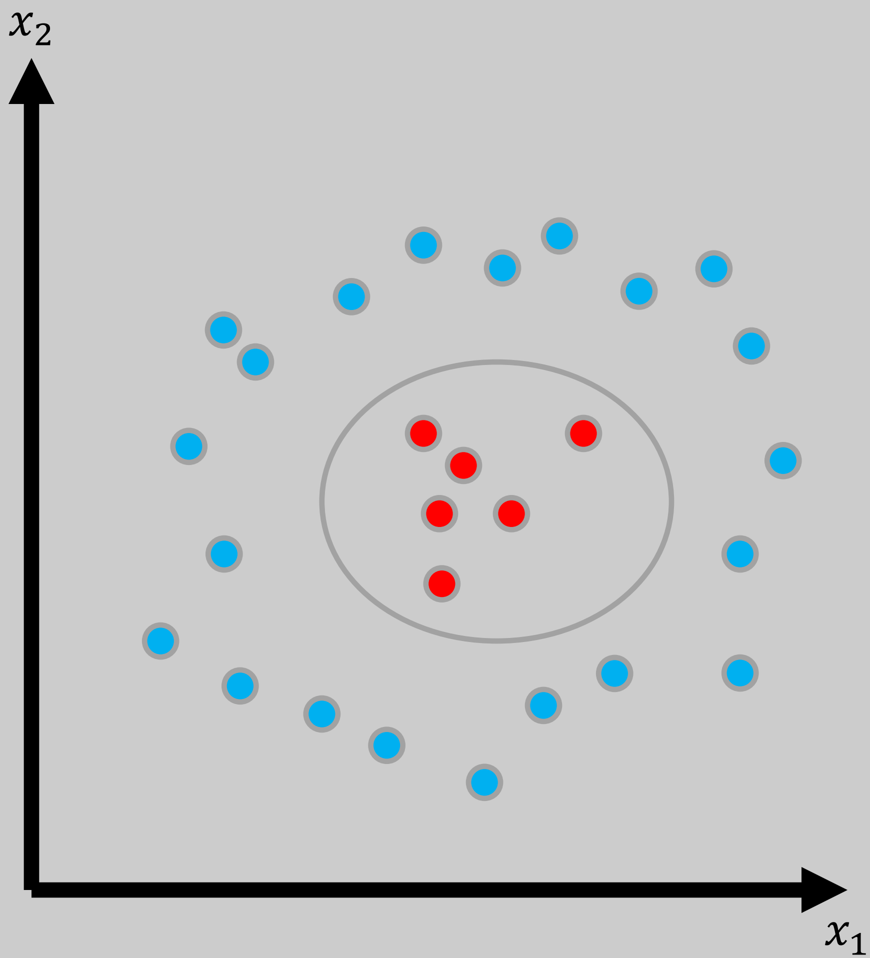
Kernel Example



How do Kernels Work?

1. Suppose we use the 5 points shown in orange as landmarks.
2. Compute similarity between each training point and these 5 landmarks according to some similarity measure.
3. Thus, each training point now has an associated vector of length 5.

Gaussian Radial Basis Kernel



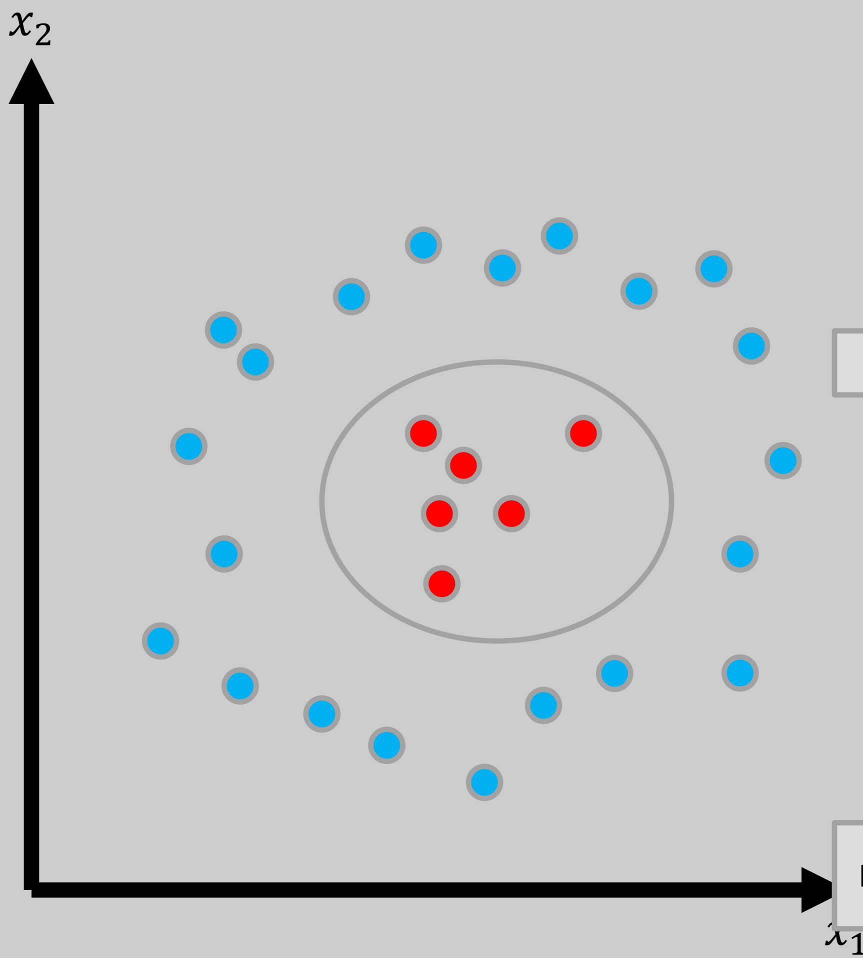
GAUSSIAN RADIAL BASIS KERNEL

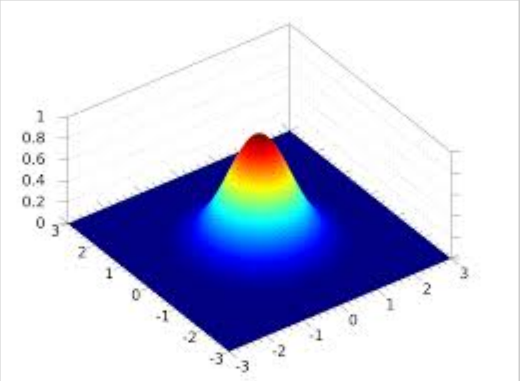
- Pick all *n* training points $x_1, \ldots, x_n$ as landmarks.
- Associate a feature vector for each training point *x*.
- The *i*'th feature (for *x*) is the similarity between *x* and the *i*'th training point.
- $f_i(x) = e^{-\left(\frac{\|x-x_i\|^2}{2\sigma^2}\right)}$ where σ is some constant.
- $\|x - x_i\|^2 = \sum_{j=1}^{m} \left(x^j - x_i^j\right)^2$

- When x, x_i are near each other, this feature value is close to 1.
- When they are far apart, the feature value is close to 0.

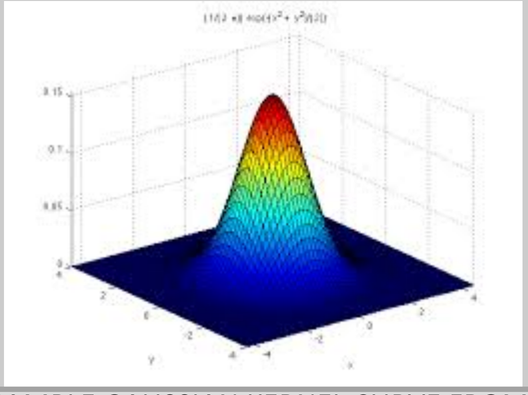
Other kernels differ in the formula used for $\boldsymbol{f_i(x)}$, e.g. use a non-Gaussian formula.

Gaussian Radial Basis Kernels



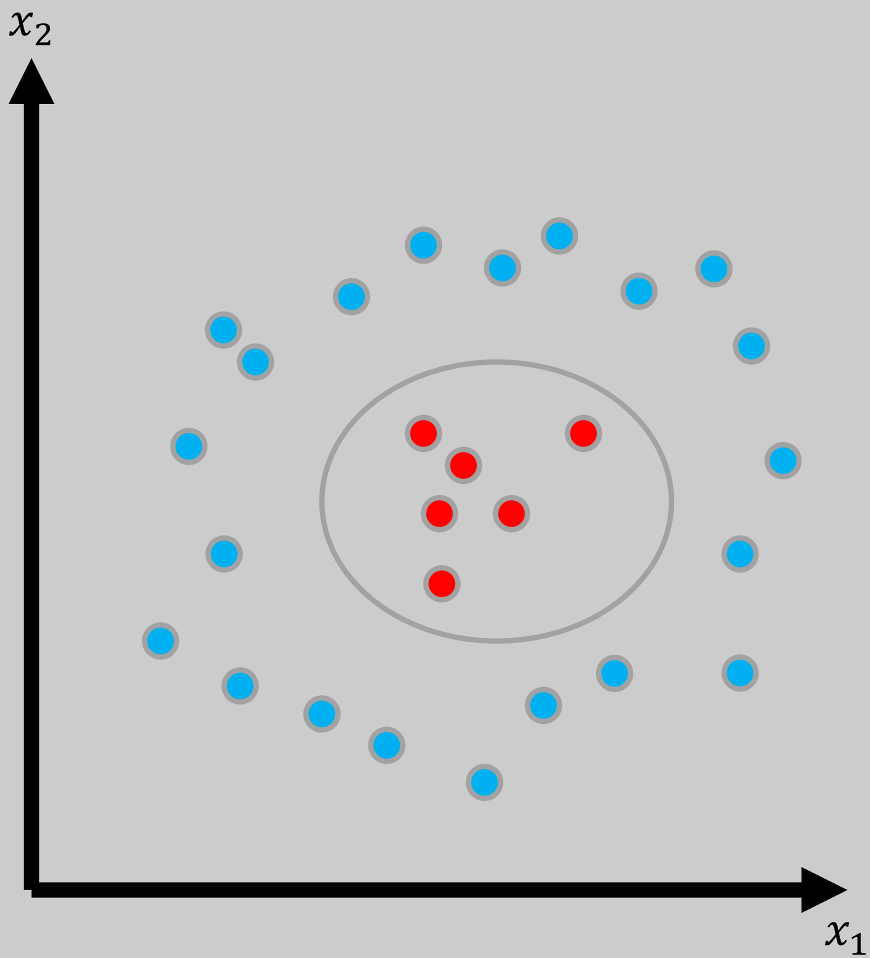


SAMPLE GAUSSIAN KERNEL CURVE FROM:
https://en.wikipedia.org/wiki/Gaussian_function/



SAMPLE GAUSSIAN KERNEL CURVE FROM:
https://imagineatness.wordpress.com/2011/12/01/3d-convolution-and-the-gaussian-kernel/

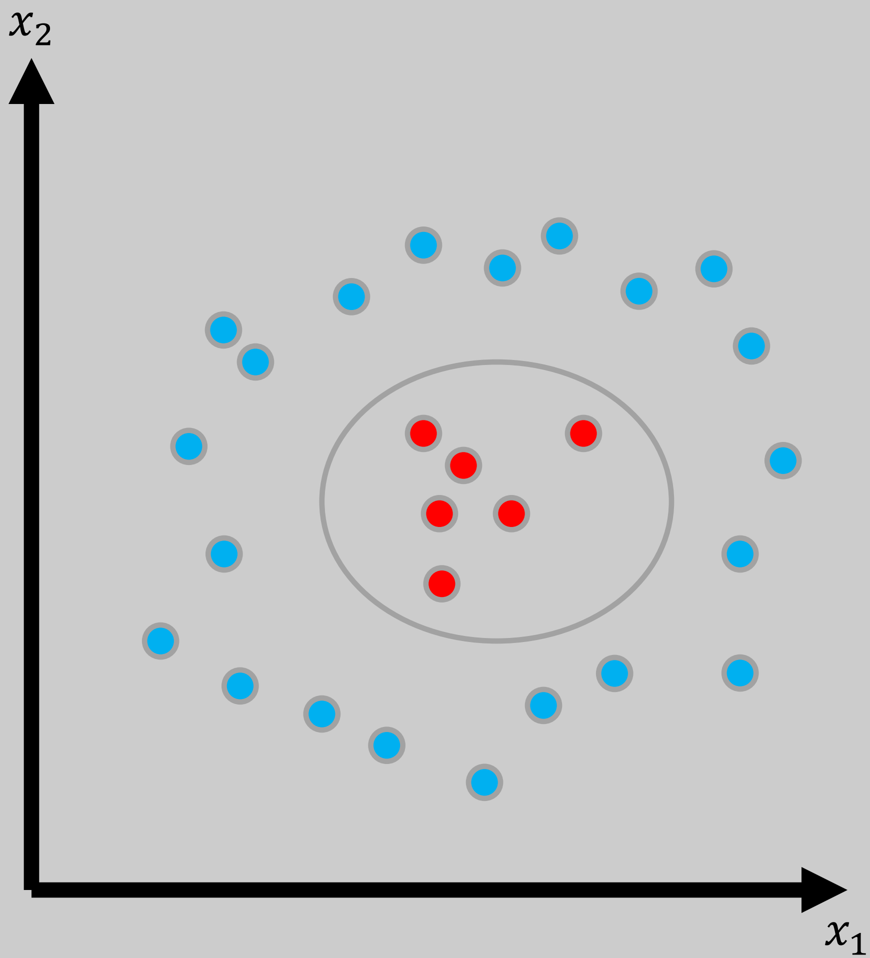
Gaussian Radial Basis Kernels



GAUSSIAN RADIAL BASIS KERNEL

- So for each training point *x*, we have an *n*-dimensional feature vector $(f_1(x), \ldots, f_{n(x)})$ where $f_i(x)$is the similarity of *x* to the i'th training point.

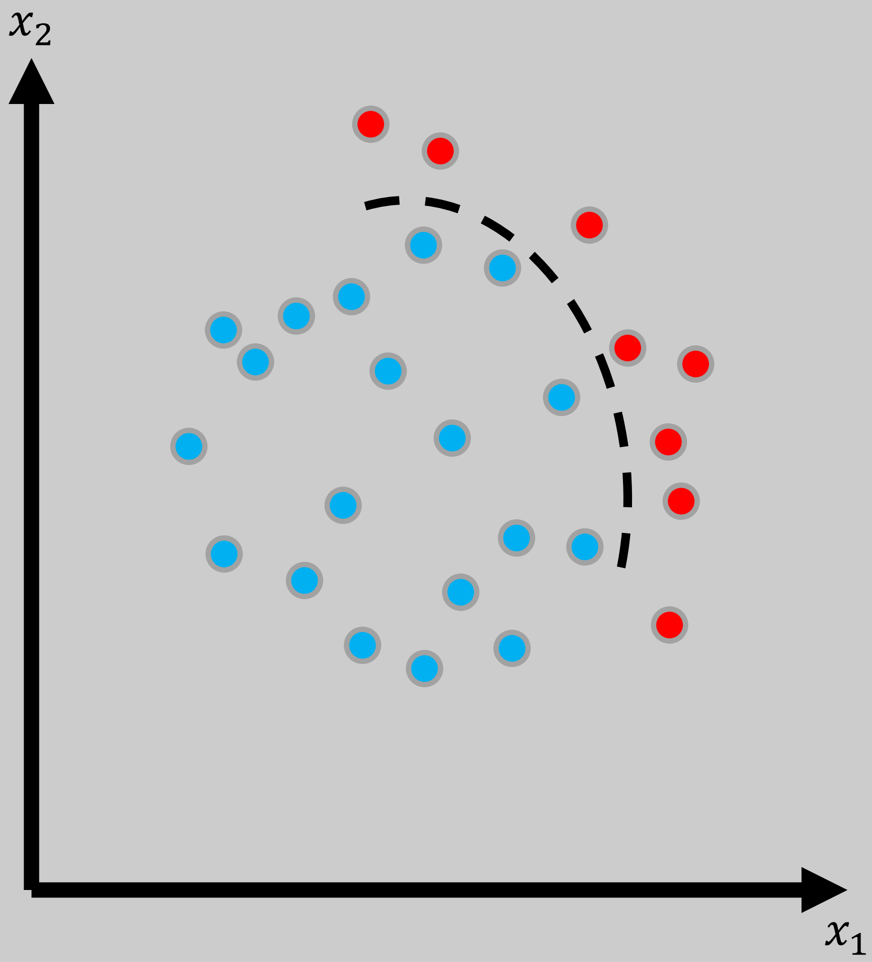
Gaussian Radial Basis Kernels



GAUSSIAN RADIAL BASIS KERNELS

- Predict class "1" for test example x when $\alpha_0 + \alpha_1 f_1(x) + \cdots + \alpha_n f_n(x) \geq 0$. Otherwise predict class -1.
- How do we learn $\alpha_0, \ldots, \alpha_n$ from the training data?
- We solve an optimization problem as before.

Other Types of Kernels



POLYNOMIAL KERNELS:

$$f(x, x_i) = (x.x_i)^d$$

or

$$f_i(x) = (x.x_i)^d$$

Uses dot product

Hyper-parameter Optimization

- Determining which kernel to use is an input *parameter* to the SVM.
- Need to identify the right parameter settings when using SVM.
  - Linear?
    - What should τ be?
  - Kernel SVM?
    - Which type of kernel?
    - What should τ be?

Minimize $\|\boldsymbol{w}\|^2 + \boldsymbol{\tau} * \sum_i \boldsymbol{e_i}$
Subject to $\mathrm{t_i} * (\boldsymbol{w}.\boldsymbol{x_i} + c) \geq 1 - e_i$
for each i=1,..,n.

Hyper-parameter Optimization:Effects of τ

- When τ is small:
  - SVM pays less attention to points near the decision boundary.
  - Margin goes up.
- When is large:
  - SVM pays a high penalty when there are points near the decision boundary.
  - So margin goes down.

Minimize $\|\boldsymbol{w}\|^2 + \boldsymbol{\tau} * \sum_i \boldsymbol{e_i}$
Subject to $t_i * (\boldsymbol{w}.\boldsymbol{x_i} + c) \geq 1 - e_i$
for each i=1,..,n.

THE END