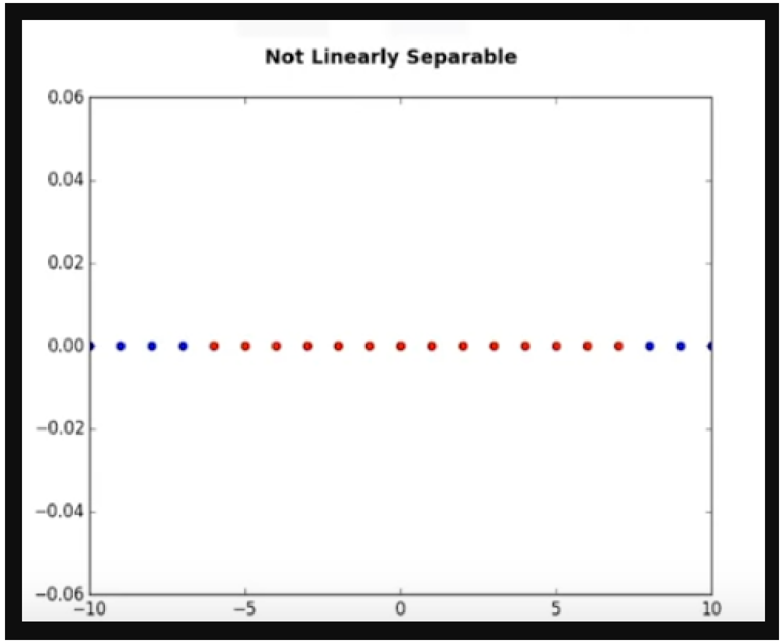
SUPPORT VECTOR MACHINES

Dr. Brian Mc Ginley

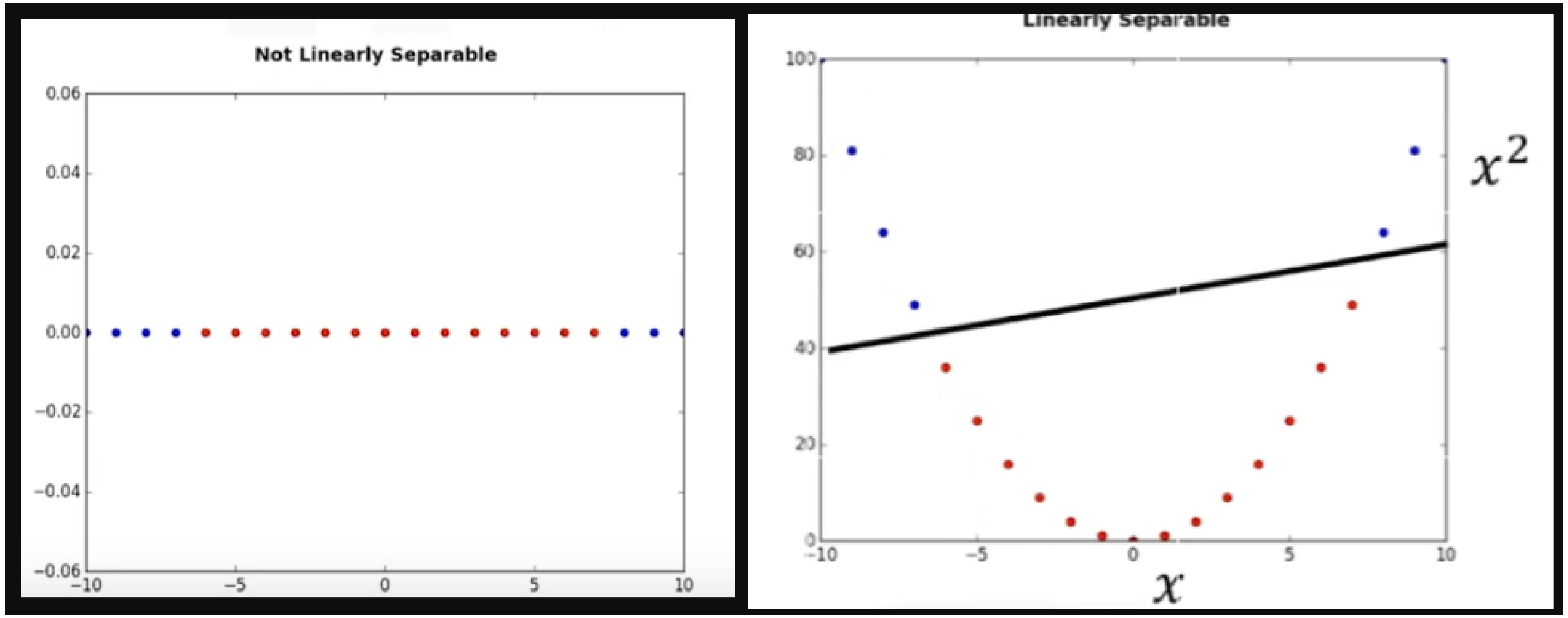
SVM - TWO KEY IDEAS

- Assuming linearly separable classes, learn separating hyperplane with maximum margin (SVC)
- But what if data is not linearly separable



SVM - TWO KEY IDEAS

- Assuming linearly separable classes, learn separating hyperplane with maximum margin (SVC)
- Expand input into high-dimensional space to deal with linearly non-separable cases



SUPPORT VECTOR CLASSIFIER: HARD MARGIN

- Finally, formulate our optimization problem: Find a decision boundary that maximises the distance to both classes – i.e. maximises the margin, M, while maintaining zero misclassifications

$$\begin{cases} max_w \dfrac{2}{\|w\|} \\ such\ that\ y^{(i)}\left(w^T x^{(i)} + w_0\right) \geq 1, \qquad \forall i \end{cases}$$

- Maximising $\frac{2}{\|w\|}$ is the same as minimizing $\|w\|$. However L2 optimisations are more stable. Therefore:

$$\begin{cases} min_w \|w\|^2 \\ such\ that\ y^{(i)}\left(w^T x^{(i)} + w_0\right) \geq 1, \qquad \forall i \end{cases}$$

- This is a quadratic optimisation problem, has linear constraints and there is a unique solution.
- Calculus again! (Lagrange multipliers if you want to look up the maths)

SOFT MARGIN SOLUTION

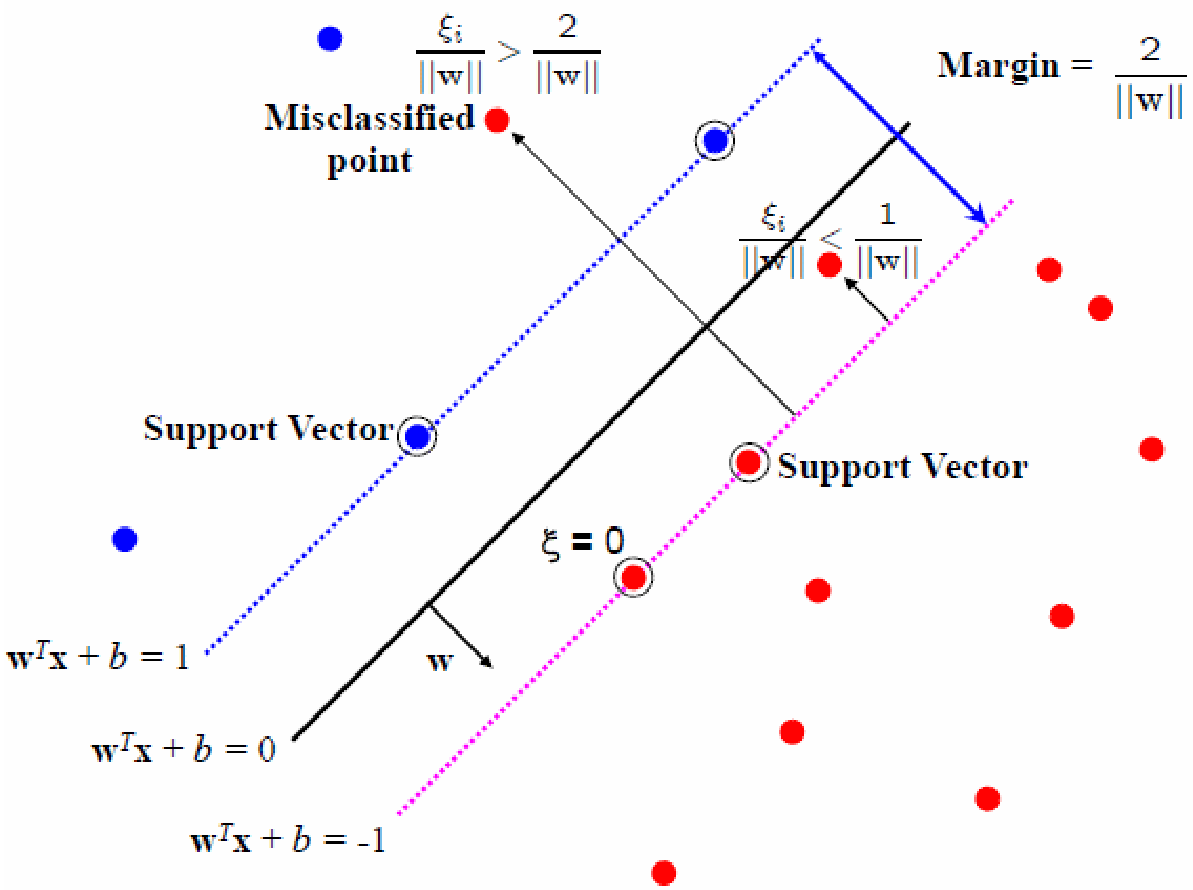
- To relax the constraints, our problem is re framed as

$$\begin{cases} min_w \|w\|^2 + C \sum_{i=1}^{N} \xi_i \\ such\ that\ y^{(i)}\left(w^T x^{(i)} + w_0\right) \geq 1 - \xi_i\ and\ \ \xi_i \geq 0, \qquad \forall i \end{cases}$$

- C is a regularisation parameter: (some notes will use λ instead of C , sklearn uses C)
  - Small C allows constraints to be easily ignored → large margin
  - Large C makes constraints hard to ignore → narrow margin
  - C → ∞ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note: there is only one parameter, C (that you choose/cross-validation).
- In general, the best C parameter depends on the situation. Experiment (Cross-Validation). One note: larger C takes more computation to train.

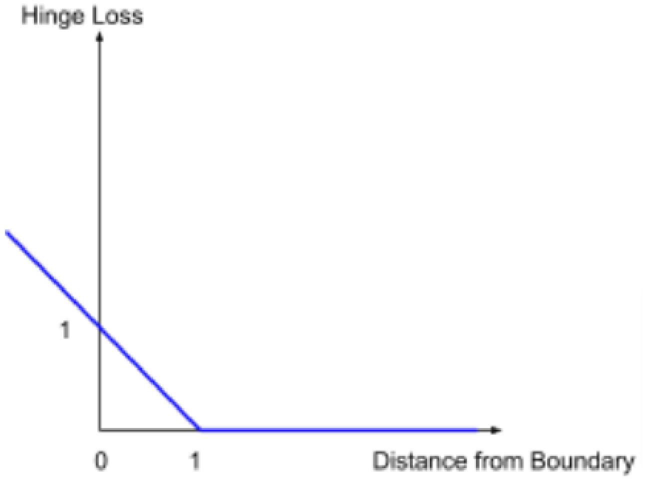
SLACK VARIABLES

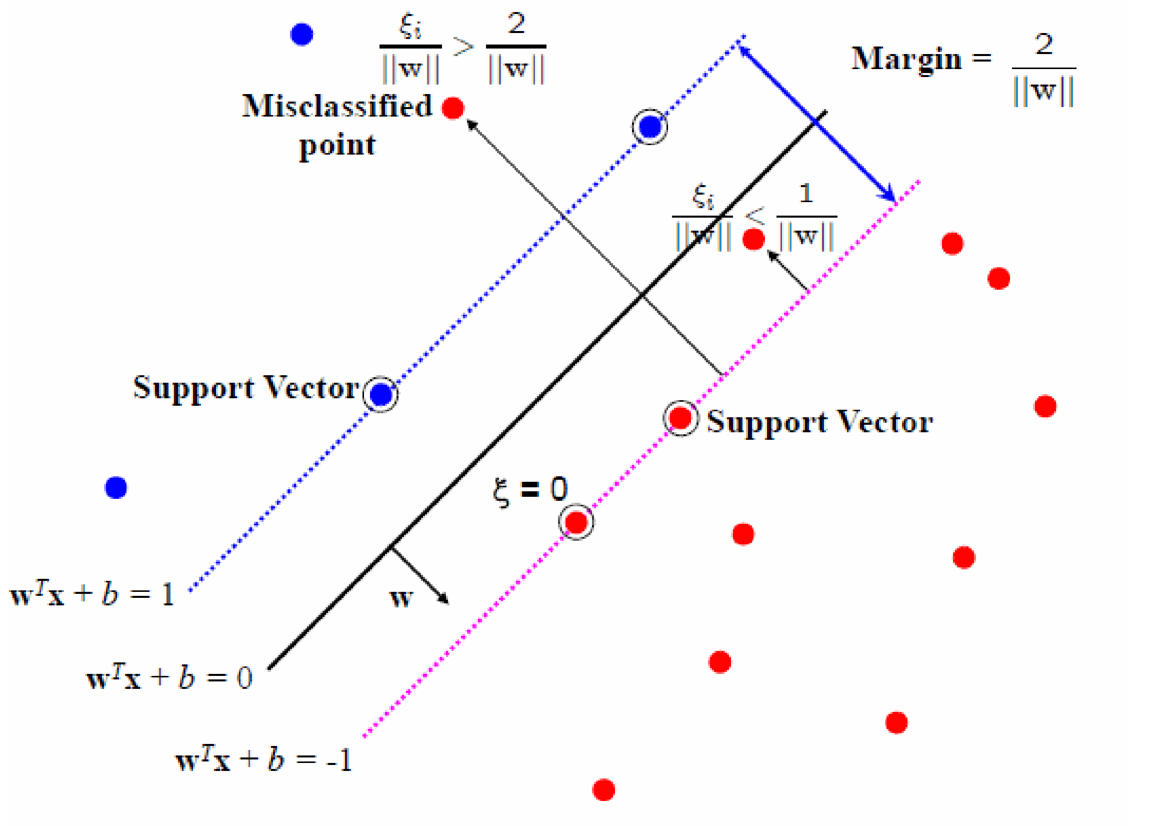
- We can add a variable $\xi_i \geq 0$ (must be bigger than 0) for each point/sample.
  - For $0 < \xi \leq 1$ point is between margin and correct side of hyperplane. This is called a margin violation
  - For $\xi \geq 1$ point is misclassified
  - For $\xi = 0$ point is the correct side of the margin.



SLACK VARIABLES

- We can add a variable $\xi_i \geq 0$ (must be bigger than 0) for each point/sample.
  - For $0 < \xi \leq 1$ point is between margin and correct side of hyperplane. This is called a margin violation
  - For $\xi \geq 1$ point is misclassified
  - For $\xi = 0$ point is the correct side of the margin.





HINGE LOSS

- The hinge loss function for a single data point is defined as:

$$L\left(y^{(i)}, f(x^{(i)})\right) = \max(0, 1 - y^{(i)} f\left(x^{(i)}\right))$$

- And in the case of SVM as:

$$L\left(y^{(i)}, f(x^{(i)})\right) = \max(0, 1 - y^{(i)}\left(w^T x^{(i)} + b\right))$$

HINGE LOSS

- The hinge loss function for a single data point is defined as:

$$L\left(y^{(i)}, f(x^{(i)})\right) = \max(0, 1 - y^{(i)} f\left(x^{(i)}\right))$$

- And in the case of SVM as:

$$L\left(y^{(i)}, f(x^{(i)})\right) = \max(0, 1 - y^{(i)}\left(w^T x^{(i)} + b\right))$$

$y^{(i)}$	$\left(w^T x^{(i)} + b\right)$	$1 - y^{(i)}\left(w^T x^{(i)} + b\right)$	$\max(0, 1 - y^{(i)}\left(w^T x^{(i)} + b\right))$
1	1	0	0
1	2	-1	0
-1	-0.5	0.5	0.5
-1	4	5	5

HINGE LOSS

- The hinge loss function for a single data point is defined as:

$$L\left(y^{(i)}, f\left(x^{(i)}\right)\right) = \max\left(0, 1 - y^{(i)} f\left(x^{(i)}\right)\right)$$

- And in the case of SVM as:

$$L\left(y^{(i)}, f\left(x^{(i)}\right)\right) = \max\left(0, 1 - y^{(i)}\left(w^T x^{(i)} + b\right)\right)$$

- Rewriting the above:

$$\xi_i \geq 1 - y^{(i)}\left(w^T x^{(i)} + b\right) \; and \; \xi_i \geq 0$$

$$y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \xi_i \; and \; \xi_i \geq 0$$

PRIMAL FORM

- Here's the classifier:

$$f(x) = w^T x + b$$

- Here's the optimisation problem:

$$\begin{cases} min_w \|w\|^2 + C \sum_{i=1}^{N} \xi_i \\ such\ that\ y^{(i)}\left(w^T x^{(i)} + w_0\right) \geq 1 - \xi_i\ and\ \ \xi_i \geq 0, \qquad \forall i \end{cases}$$

- However, it can be written another way (it has loads of ways of being written). This next way should be the "most" useful way.

DUAL FORM

- If you formulate the linear classifier instead as (instead of $^{(i)}$, I am going to write it as a subscript α_i for a particular sample):

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \left(x_i^T x\right) + b$$

Remember (Primal)
$f(x) = w^T x + b$

- we now solve an optimisation problem over α_i . i.e.

$$max_w W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j \alpha_k y_j y_k \left(x_j^T x_k\right)$$

Subject to $0 \leq \alpha_i \leq C$ for all i and $\sum_i \alpha_i y_i = 0$

DUAL FORM

- To classify a new point/sample:
  - Primal version of classifier:

$$f(x) = w^T x + b$$

  - Dual version of classifier

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \left(x_i^T x\right) + b$$

  - and take the sign as before as the classification output.

DUAL FORM

- To classify a new point/sample:
  - Primal version of classifier:

$$f(x) = w^T x + b$$

  - Dual version of classifier

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \left(x_i^T x\right) + b$$

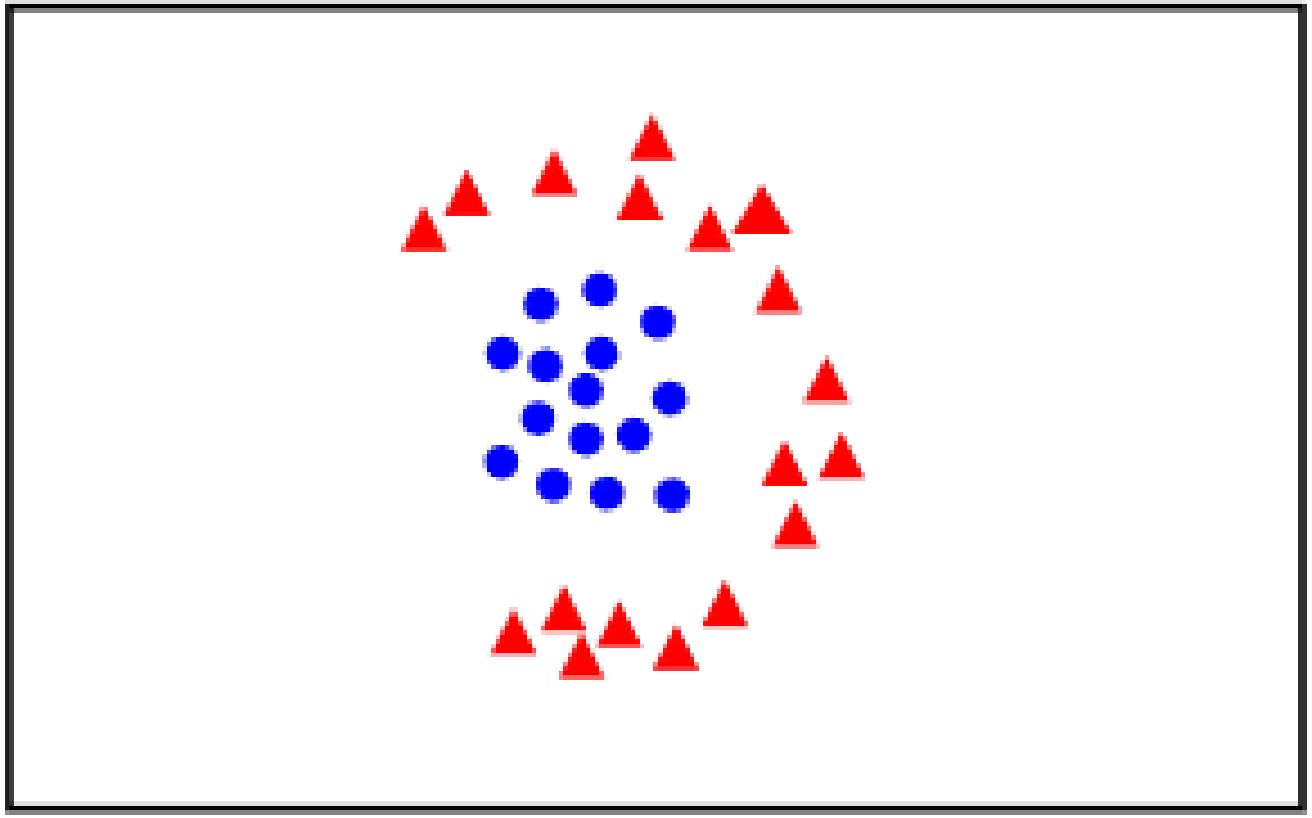
  - and take the sign as before as the classification output.
- The dual form appears to have the disadvantage of a kNN classifier — it requires the training data points x_i. However, many of the α_i's are zero. The ones that are non-zero define the support vectors x_i. You could write it as

$$f(x) = \sum_{x_i \text{ is a support vector}} \alpha_i y_i \left(x_i^T x\right) + b$$

  - then we classify a point by finding the sign of the above.

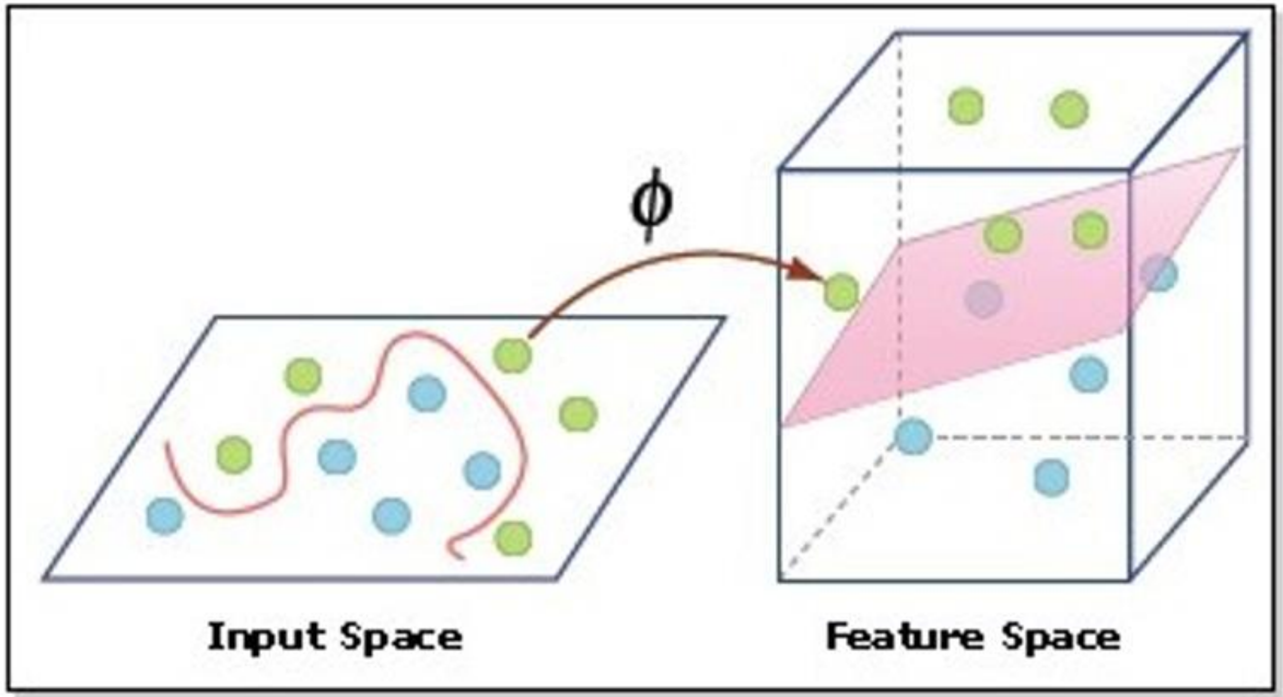
NON LINEARLY SEPARABLE DATA

- How do we deal with this?



NON LINEARLY SEPARABLE DATA

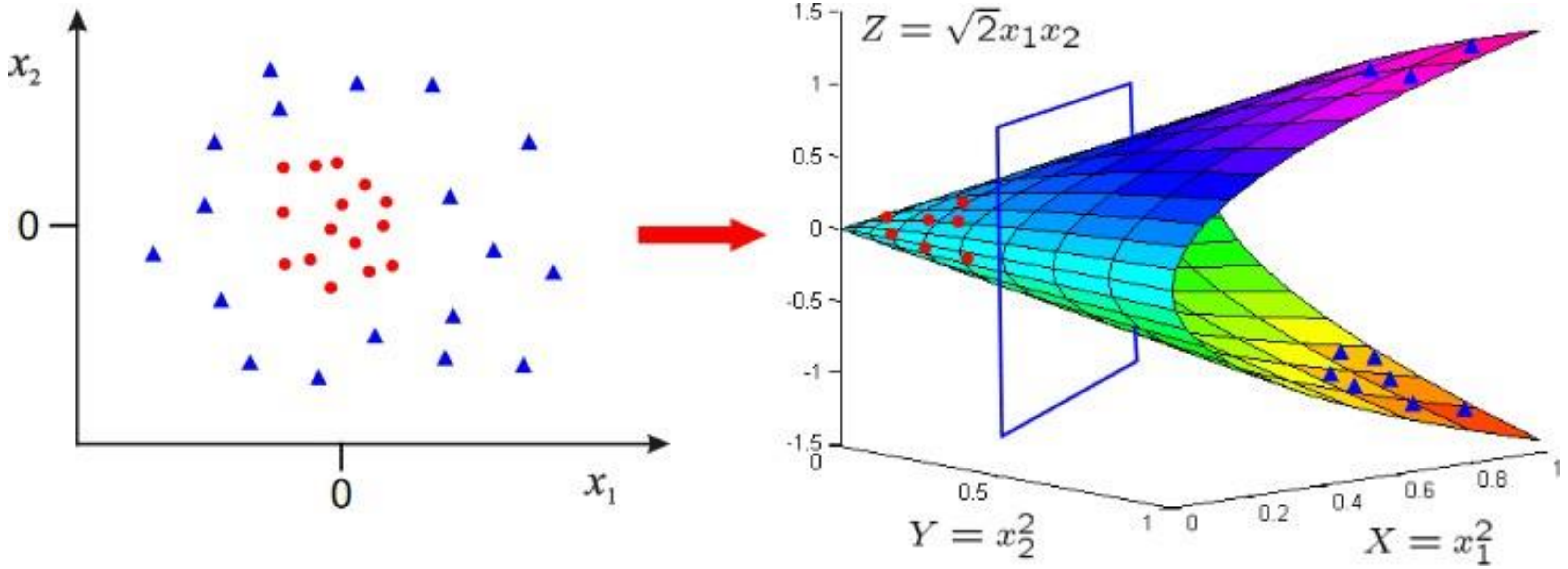
- It is not possible to find a hyperplane or a linear decision boundary for some classification problems. If we project the data into a higher dimension from the original space, we may get a hyperplane in the projected dimension that helps to classify the data.



A SOLUTION - MAP TO HIGHER DIMENSION

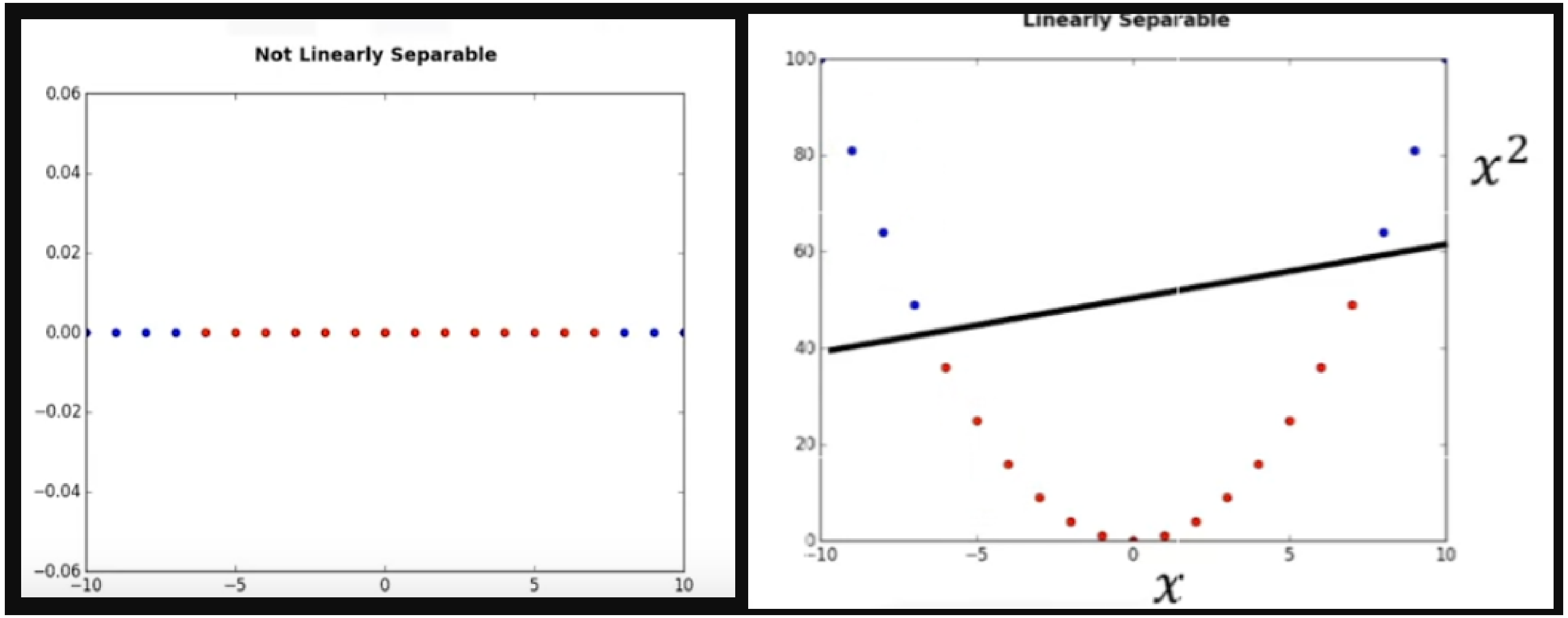
- Data is linearly separable in 3D
- The problem can still be solved by a linear classifier

$$\Phi(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{pmatrix} \quad R^2 \rightarrow R^3$$



A SOLUTION - MAP TO HIGHER DIMENSION

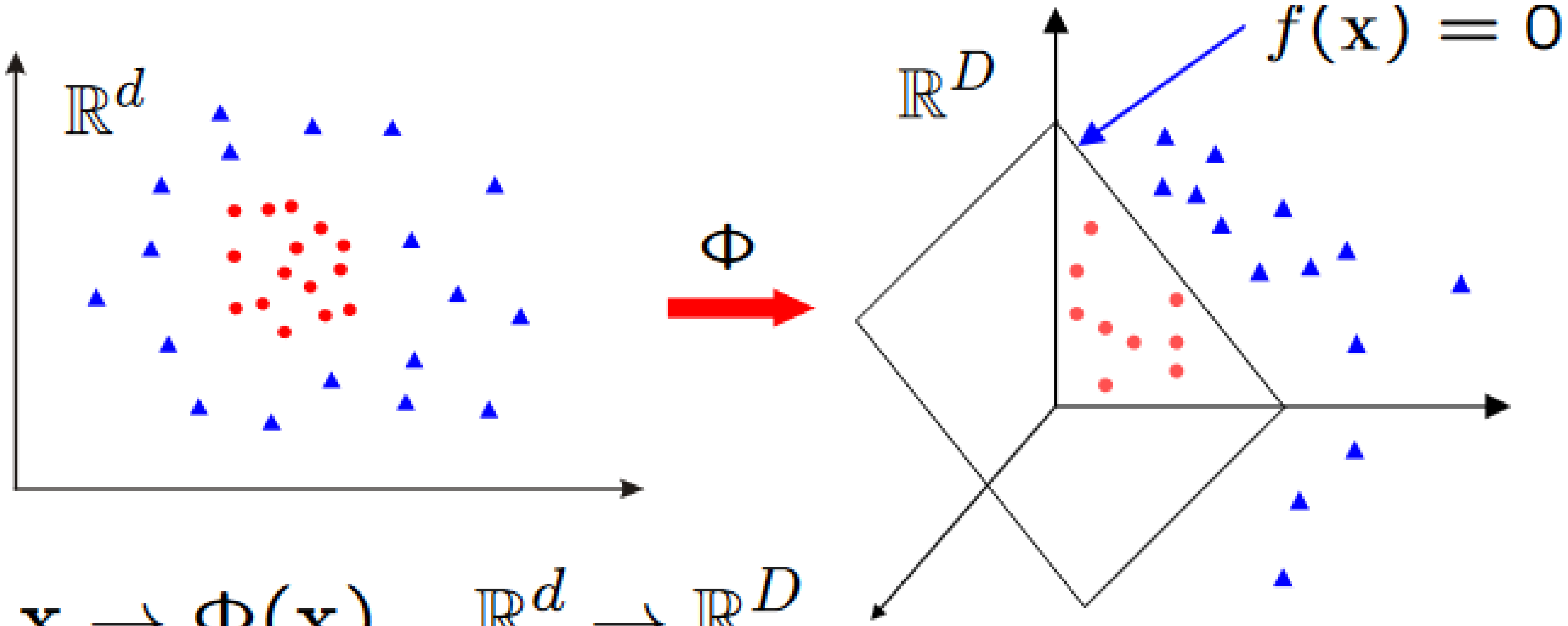
- Data is linearly separable in 2D
- The problem can still be solved by a linear classifier



$$\Phi(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \quad R^2 \to R^3$$

SVM IN A TRANSFORMED FEATURE SPACE

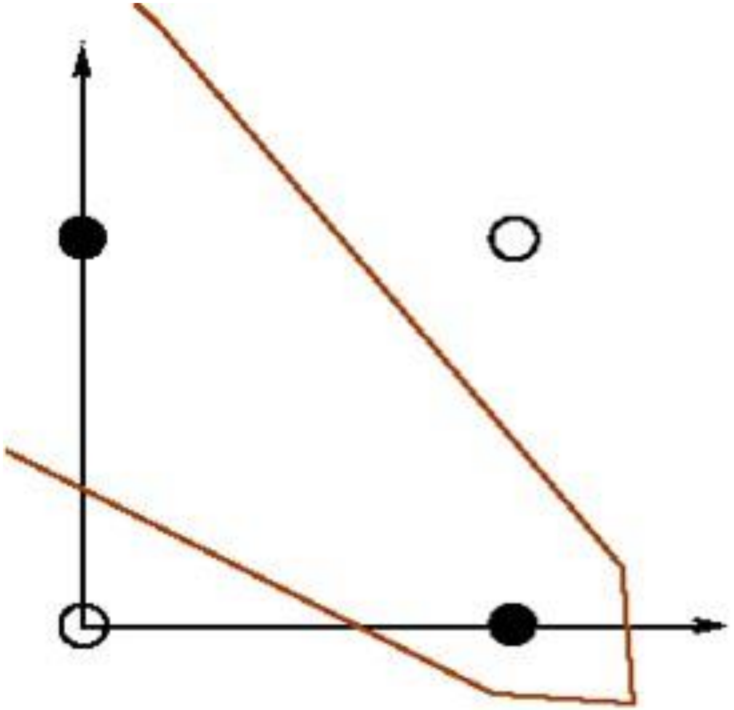
- Task: Learn linear classifier **w** for R^D: $f(x) = w^T\Phi(x) + b$
  - Where $\Phi(x)$ is a feature map

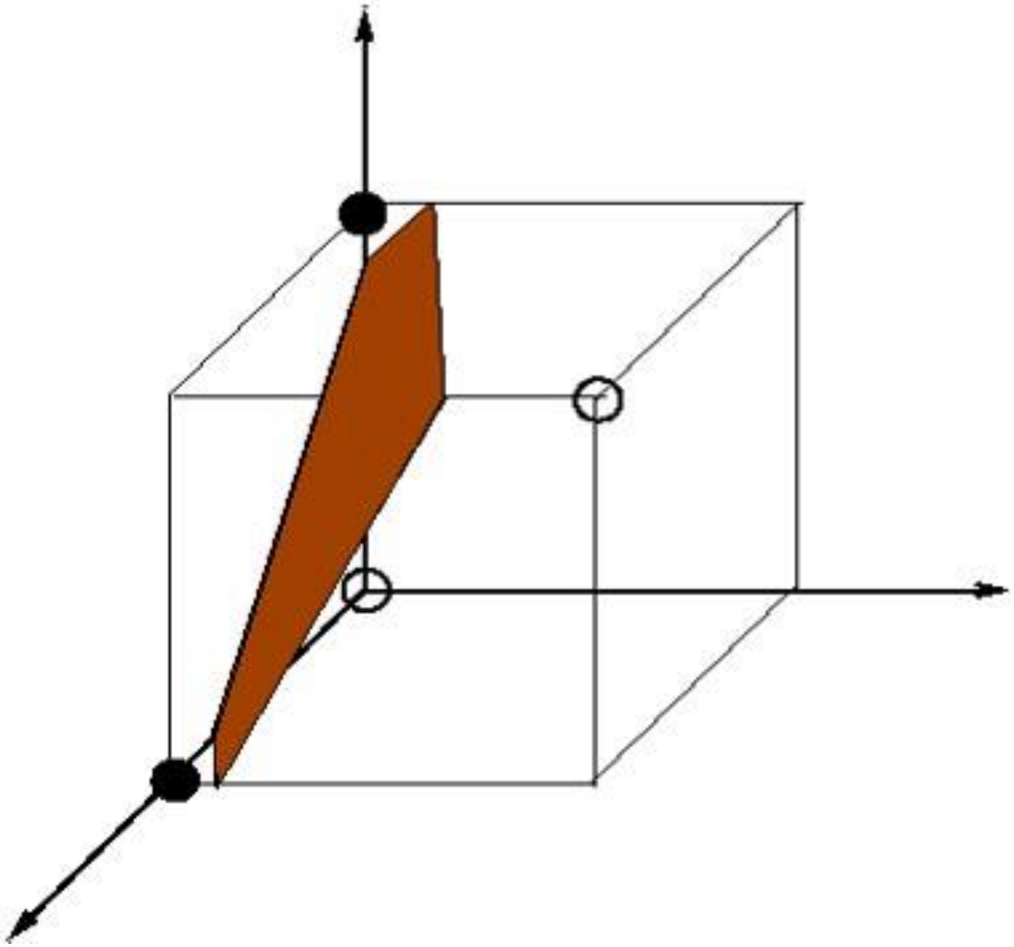


$\Phi : \mathbf{x} \to \Phi(\mathbf{x}) \quad \mathbb{R}^d \to \mathbb{R}^D$

EXAMPLE: THE XOR PROBLEM

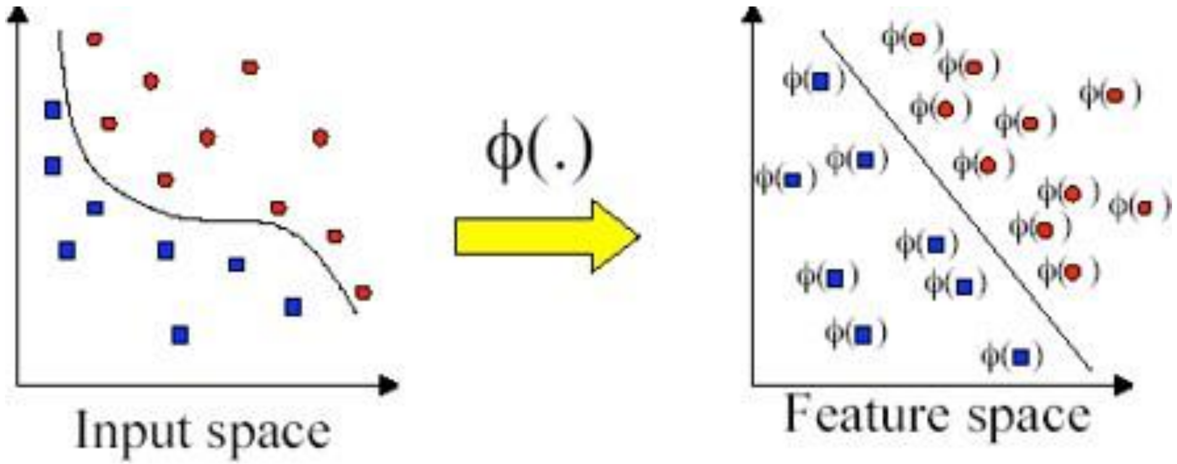
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0





SVM NON-LINEAR DECISION BOUNDARIES

- The idea: instead of tweaking the definition of SVC to accommodate non-linear decision boundaries, we map the data into a feature space in which the classes are linearly separable (or nearly separable):
  - Transform $x \rightarrow \phi(x)$
  - The linear algorithm depends only on $x^T x_i$, hence transformed algorithm depends only on $\phi(x)^T \phi(x_i)$
  - Use kernel function $K(x, x_i)$ such that $K(x, x_i) = \phi(x)^T \phi(x_i)$



COMPLEXITY

- After projecting the data into a higher dimension, we could find the hyperplane which classifies the data.
- Usually, the computational cost will increase, if the dimension of the data increases.
- Kernel helps to find a hyperplane in the higher dimensional space without increasing the computational cost much.

KERNEL TRICK

$$\Phi(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \quad R^2 \rightarrow R^3$$

- Φ is a function that transforms x from 2D to 3D.
- We can now have a decision boundary in this 3D space:

$$w_0 + w_1x_1^2 + w_2x_2^2 + w_3\sqrt{2}x_1x_2 = 0$$

- Project x into high dim space and calculate dot product

$$\phi(x_i).\phi(x_j) = \left(x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2}\right)\left(x_{j1}^2, x_{j2}^2, \sqrt{2}x_{j1}x_{j2}\right)$$

$$\boxed{\phi(x_i).\phi(x_j) = x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2}}$$

- The Kernel Trick:
  - The kernel is defined as: $\mathrm{K}(x_i, x_j) = (x_i^T.x_j)^2$

$$(x_i^T.x_j)^2 = \{(x_{i1}, x_{i2})^T.(x_{j1}, x_{j2})\} = \left(x_{i1}x_{j1} + x_{i2}x_{j2}\right)^2$$

$$\boxed{(x_i^T.x_j)^2 = x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2}}$$

KERNEL TRICK EXAMPLE

- Let's say we have two features $x_i = (x_{i1}, x_{i2})$, i.e. $\in$ R^2.
- Assume we have some transformation to convert to a four-dimensional feature space $\left(x_{i1}^2, x_{i1}x_{i2}, x_{i2}x_{i1}, x_{i2}^2\right)$. It requires $O(n^2)$ time to calculate n data points in the 4-dimensional space.
- If we want to calculate the dot product in the 4-dim space the standard way it's:
  1. Convert each point from $R^2 \rightarrow R^4$ by applying the transformation

$$\phi(x_i) = (x_{i1}^2, x_{i1}x_{i2}, x_{i2}x_{i1}, x_{i2}^2) \qquad \phi(x_j) = (x_{j1}^2, x_{j1}x_{j2}, x_{j2}x_{j1}, x_{j2}^2)$$

  2. Dot product the two vectors $\phi(x_i).\phi(x_j)$

KERNEL TRICK EXAMPLE

- Example, say $x_i = (1, 2)$ and $x_j = (3, 5)$
- Transform them through ϕ to get:

$$\phi(x_i) = (1, 2, 2, 4) \qquad \phi(x_j) = (9, 15, 15, 25)$$

- Get the dot product:

$$\phi(x_i).\,\phi(x_j) = (1, 2, 2, 4).\,(9, 15, 15, 25) = 169$$

- But, the kernel of our function is actually:

$$\mathrm{k}(x_i, x_j) = (x_i^T x_j)^2$$

- So, we can calculate with our example directly as:

$$\mathrm{k}(x_i, x_j) = ((1, 2).\,(3, 5))^2 = (3 + 10)^2 = 169$$

- The standard method of calculating this requires $O(n^2)$ but kernel requires just $O(n)$

KERNEL TRICK

- Kernel helps to find a hyperplane in the higher dimensional space without increasing the computational cost much.
  - https://www.quora.com/What-is-the-kernel-trick/answer/Chitta-Ranjan-2 I think is a really good detailed answer about this "Kernel Trick".
- Since the feature space R^D is extremely high dimensional, computing Φ explicitly can be costly. This is because if D >> d then there are many more parameters to learn for w.
  - Instead, we note that computing Φ is unnecessary.
- Classifier:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \left(x_i^T x \right) + b$$

$$\rightarrow f(x) = \sum_{i=1}^{N} \alpha_i y_i \left(\phi(x_i)^T \phi(x) \right) + b$$

KERNEL TRICK

- Optimisation:

$$max_w W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} \alpha_j \alpha_k y_j y_k \left(x_j^T x_k\right)$$

$$\rightarrow max_w\ W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} \alpha_j \alpha_k y_j y_k \left(\phi\left(x_j\right)^T \phi(x_k)\right)$$

Subject to $0 \leq \alpha_i \leq C$ for all i and $\sum_i \alpha_i y_i = 0$

- Note that $\Phi(\mathbf{x})$ only appears in pairs $\Phi(\mathbf{x}_j)^T\Phi(\mathbf{x}_i)$. So we only need to compute these.
- Once these products are computed (dot product, scalar product), only the N dimensional vector α needs to be learnt.
- Write $\Phi(\mathbf{x}_j)^T\Phi(\mathbf{x}_i) = k(\mathbf{x}_j, \mathbf{x}_i)$. This is known as the Kernel

KERNEL TRICK

- Classifier:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \, \mathrm{k}(x_i, x) + b$$

- Optimisation:

$$max_w W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j \alpha_k y_j y_k \, \mathrm{k}\left(x_j, x_k\right)$$

Subject to $0 \leq \alpha_i \leq C$ for all i and $\sum_i \alpha_i y_i = 0$

KERNEL TRICK SUMMARY

- Now: $\Phi(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$ $R^2 \rightarrow R^3$
- Can calculate

$$\phi(x)^T . \phi(z) = \cdots = (x^t z)^2$$

- Classifier can be learnt and applied without explicitly computing $\Phi(x)$.
- All that is required is the kernel $k(x, z) = (x^T z)^2$
- Complexity of learning depends on N (the size of the training set) not on D (the higher dimension). This is a lot faster.

SVM KERNELS

- Linear kernels (kernel='linear') $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T\mathbf{x}_2$
- Sigmoid Kernel (kernel='sigmoid') $k(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\kappa\mathbf{x}_1^T\mathbf{x}_2 + \theta)$
  - κ and θ are hyperparameters.

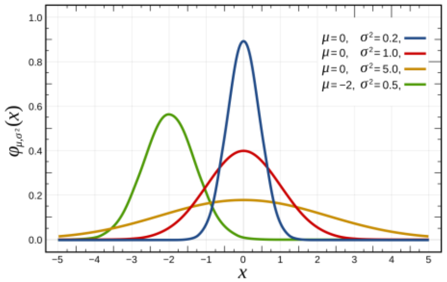
SVM POLYNOMIAL KERNEL

- Polynomial kernels (kernel='poly') $k(x_1, x_2) = (x_1^T x_2 + c)^d$ for any $d > 0$
  - Contains all polynomials terms up to degree d .
  - d is a hyperparameter.
  - c is also a hyperparameter that varies the importance of the non-linear terms
- c (called coef0 in sklearn) – default is 0

c Value	Expanded Polynomial Terms
$c = 0$	$(x_1y_1)^2 + (x_2y_2)^2 + 2(x_1y_1)(x_2y_2)$
$c = 1$	$(x_1y_1)^2 + (x_2y_2)^2 + 1 + 2(x_1y_1)(x_2y_2) + 2(x_1y_1) + 2(x_2y_2)$
$c = 10$	$(x_1y_1)^2 + (x_2y_2)^2 + 100 + 2(x_1y_1)(x_2y_2) + 20(x_1y_1) + 20(x_2y_2)$

  - As c increases, the constant term dominates more, thereby smoothing the decision boundary.
  - The linear terms (x_1y_1 and x_2y_2) grow proportionally to c, overshadowing higher-order interactions for large c.

$$f(x) = a \exp\left(-\frac{(x-b)^2}{2c^2}\right)$$



SVM RBF KERNEL

- Gaussian kernels (kernel='rbf') $\mathbf{k}(\mathbf{x}_1, \mathbf{x}_2) = exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)$ for $\sigma > 0$.
  - Infinite dimensional feature space for Radial Basis Function kernel
  - σ is a hyperparameter.
- In sklearn , the RBF kernel is expressed in terms of γ instead of σ:
  - $\mathbf{k}(\mathbf{x}_1, \mathbf{x}_2) = exp(-\gamma\|x_1 - x_2\|^2)$ where $\gamma = \frac{1}{2\sigma^2}$
  - This can be expanded using Taylor series into an infinite series:

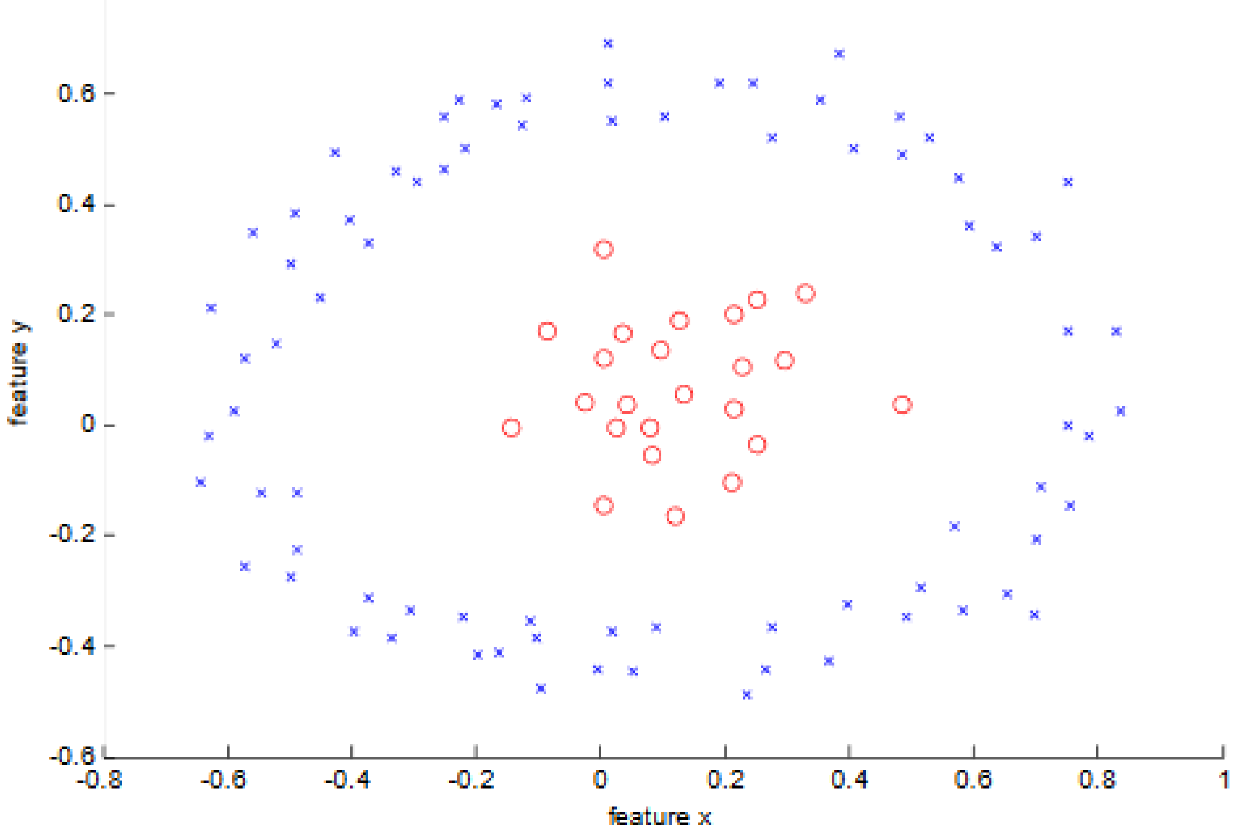
$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \ldots.$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

$$\exp\left(-\gamma\|\mathbf{x}-\mathbf{y}\|^2\right) = 1 - \gamma\|\mathbf{x}-\mathbf{y}\|^2 + \frac{\left(\gamma\|\mathbf{x}-\mathbf{y}\|^2\right)^2}{2!} - \frac{\left(\gamma\|\mathbf{x}-\mathbf{y}\|^2\right)^3}{3!} + \cdots$$

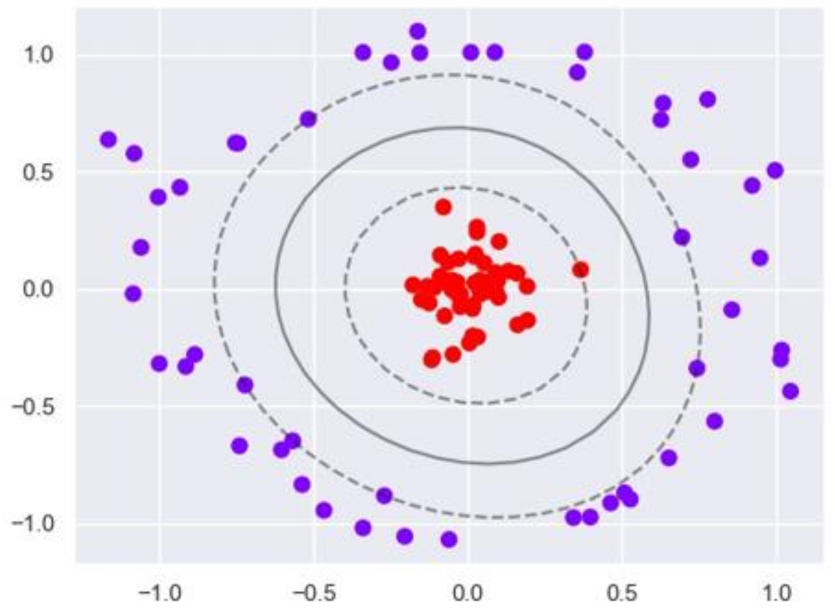
RADIAL BASIS FUNCTION (RBF) SVM

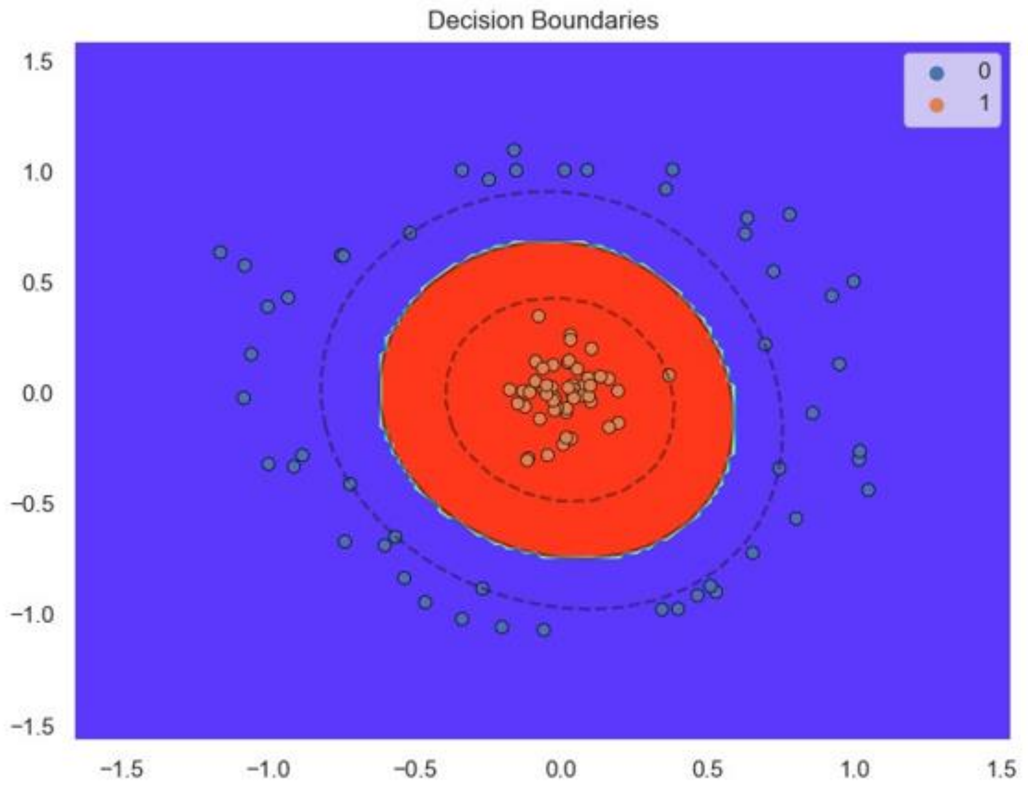
- Classifier: $$f(x) = \sum_{i=1}^{N} \alpha_i y_i \, exp(-\gamma \|x_1 - x_2\|^2) + b$$
- In sklearn we choose our γ (related to σ) and C (margin softness) when building our model. There are default values.
- Data is not linearly separable in the original feature space.



EXAMPLE PARAMS

- σ = 1, large C , γ = 0.5



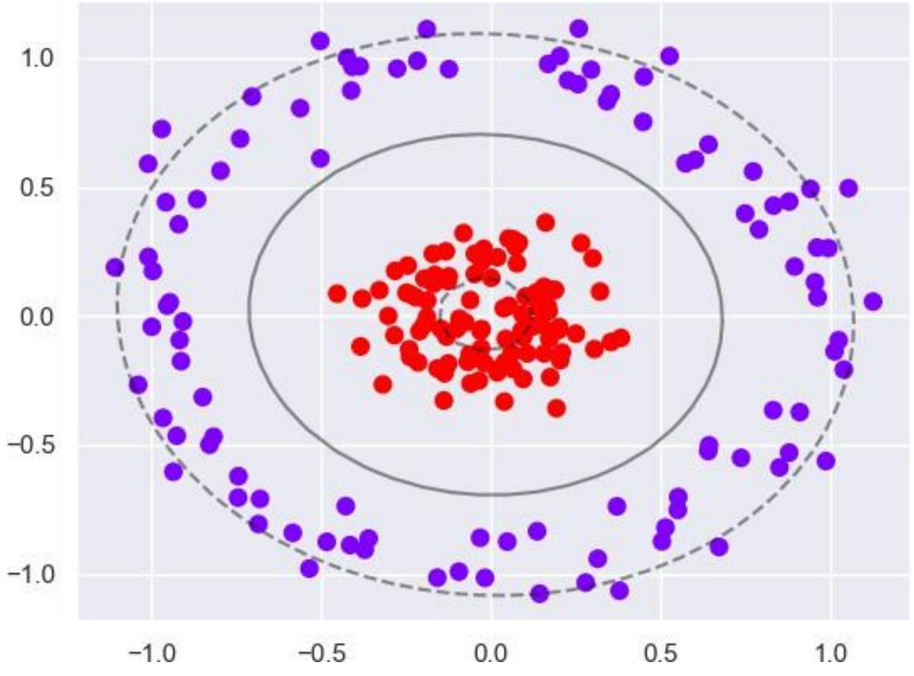


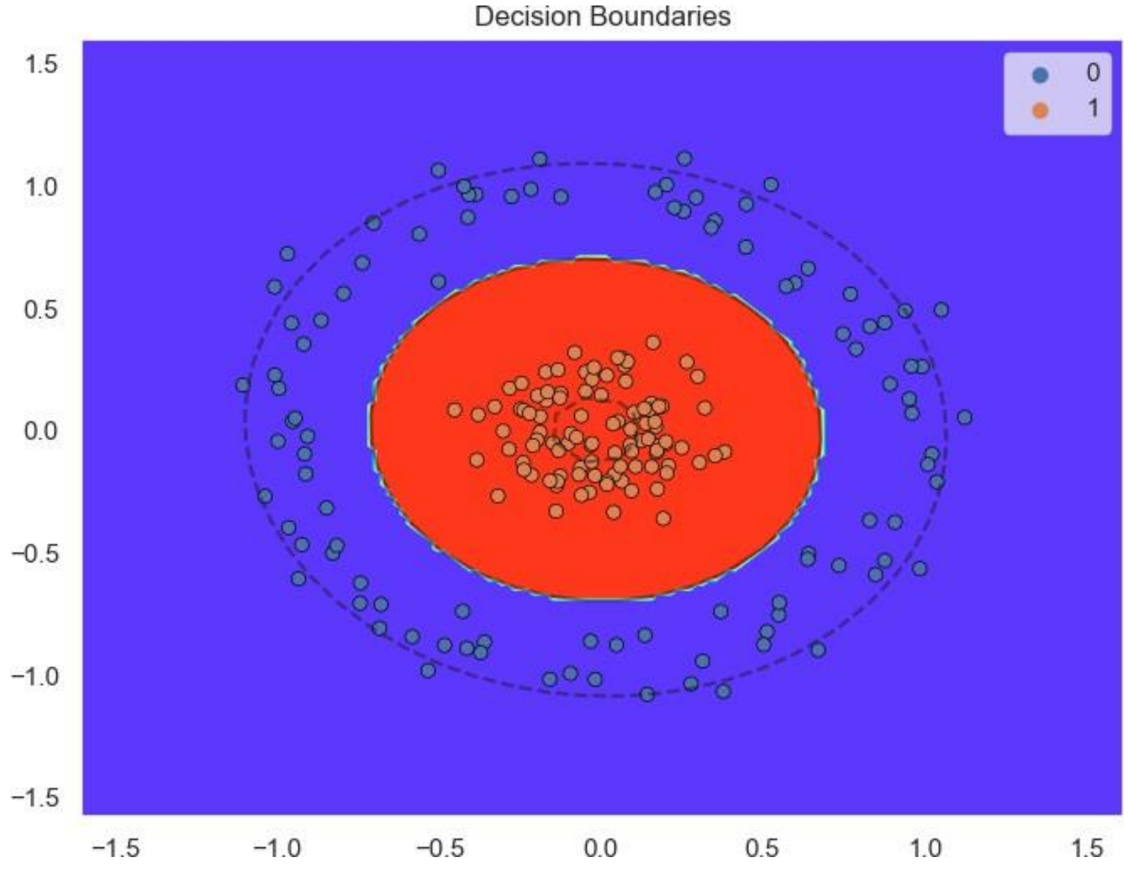

```
len(clf.support_vectors_)
6

clf.score(X, y)
1.0
```

EXAMPLE PARAMS

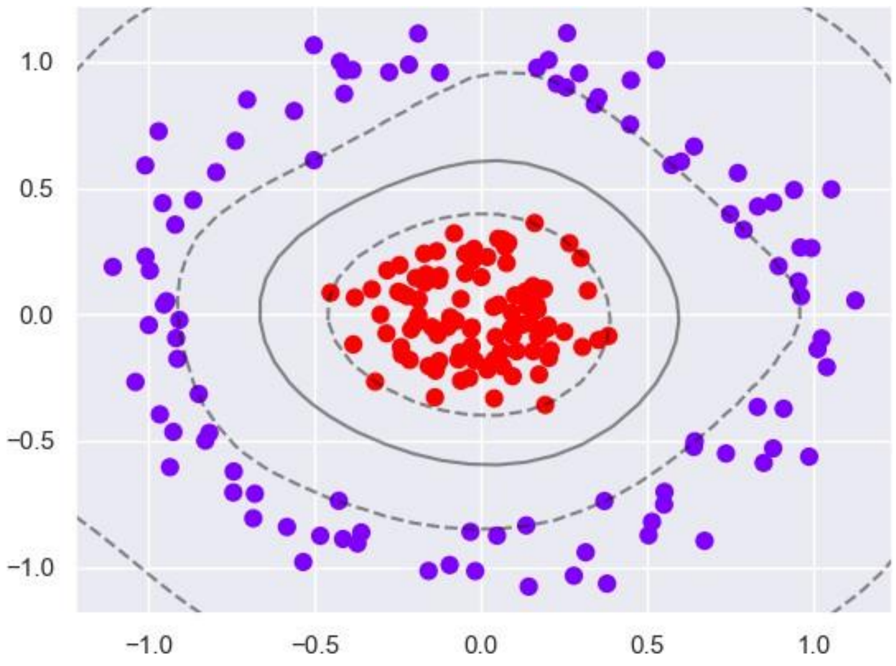
- $\sigma = 1$, $C = 0.1$, $\gamma = 0.5$
- Number of support vectors 171

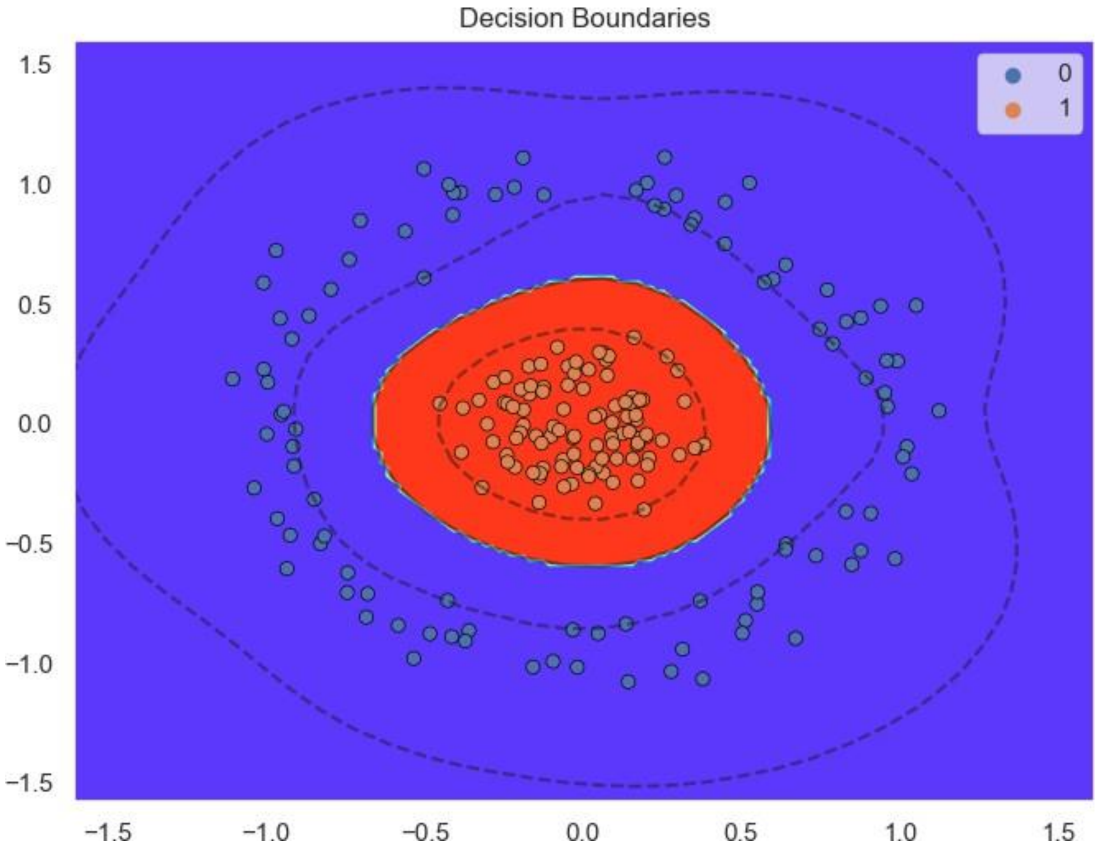




EXAMPLE PARAMS

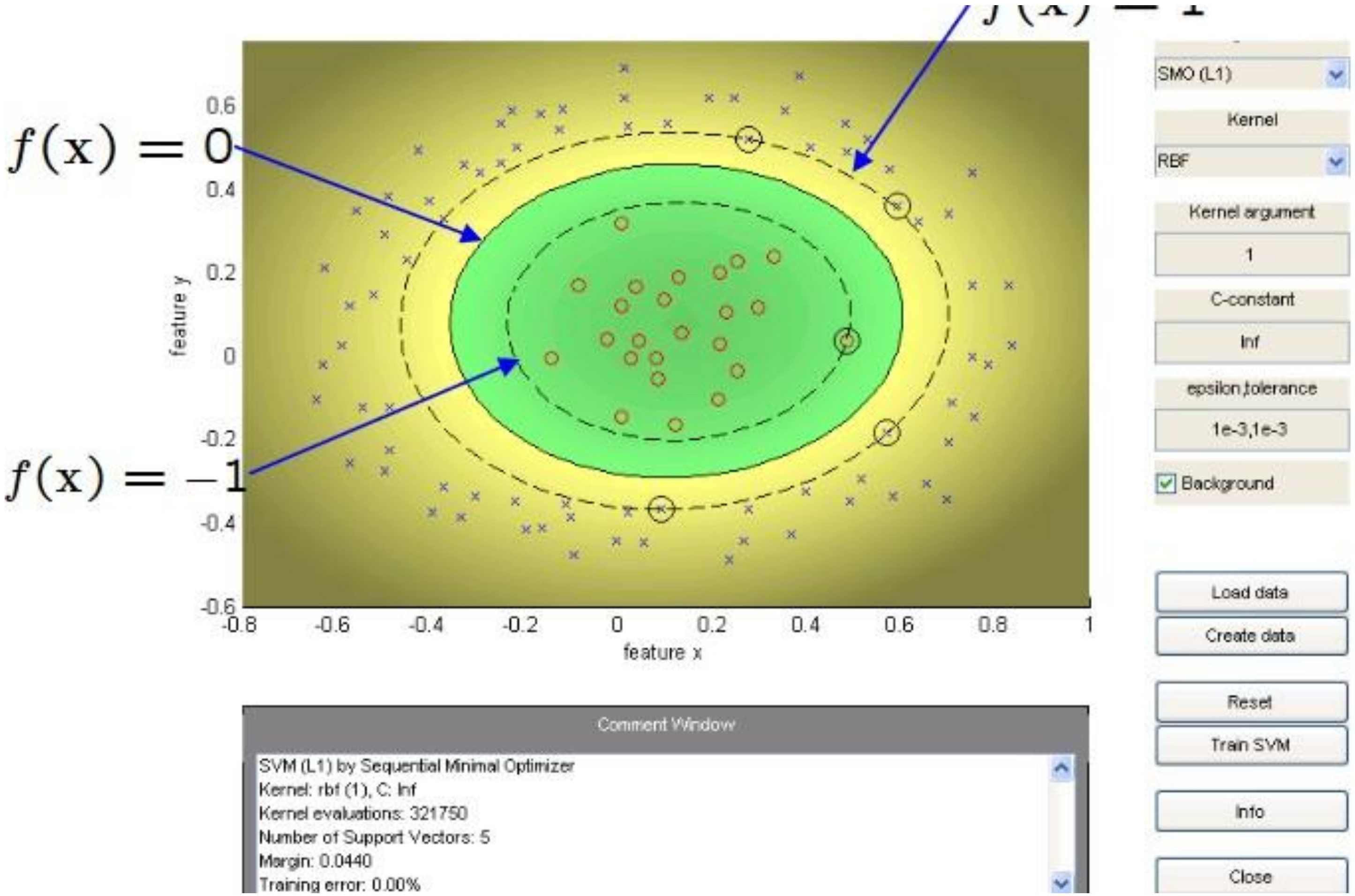
- $C = 0.1$, $\gamma = 3$
- The bigger γ gets, the closer towards nearest neighbour classifier the rbf kernel looks.





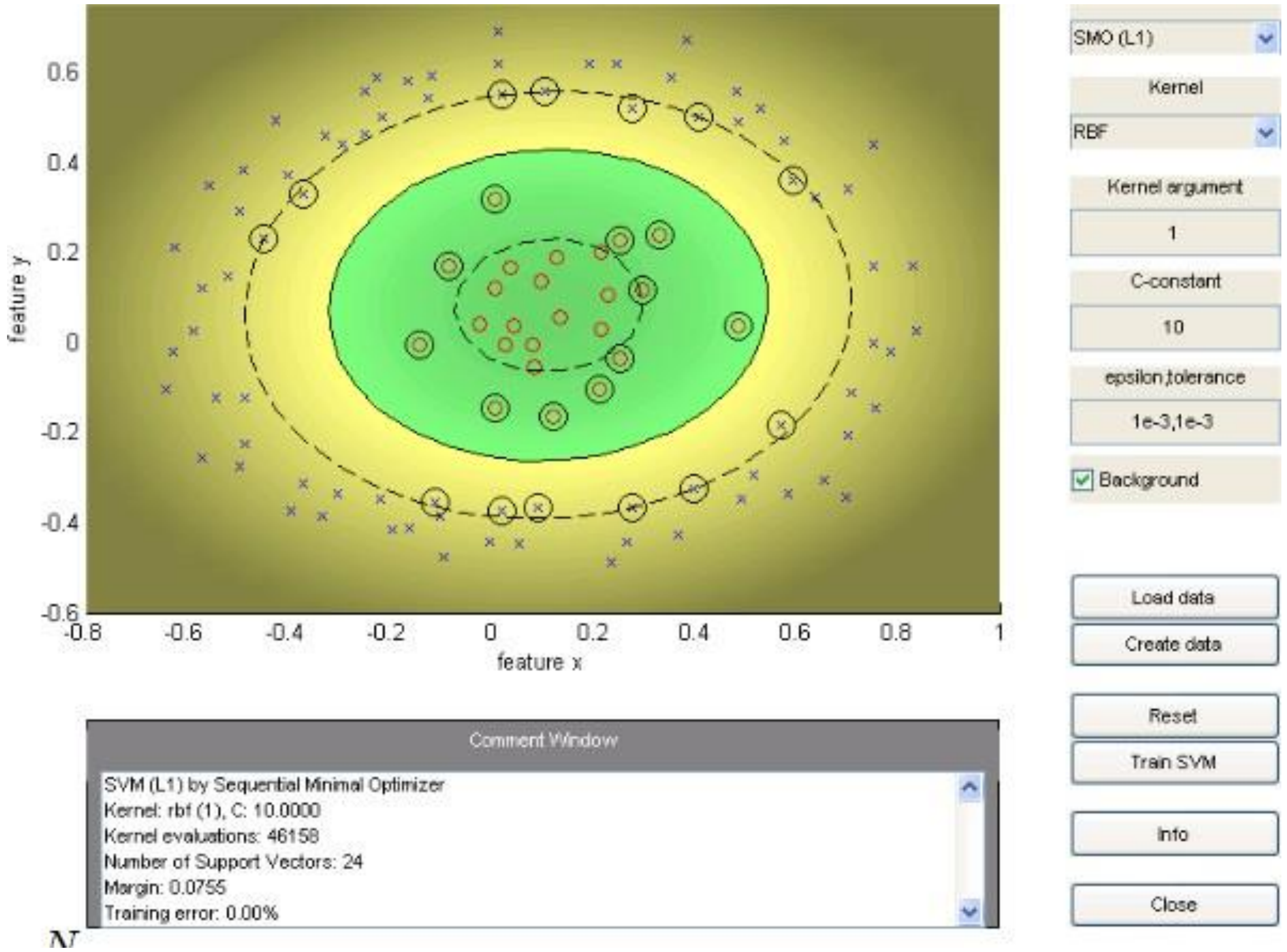
EXAMPLE PARAMS

- C = infinity, γ = 0.5



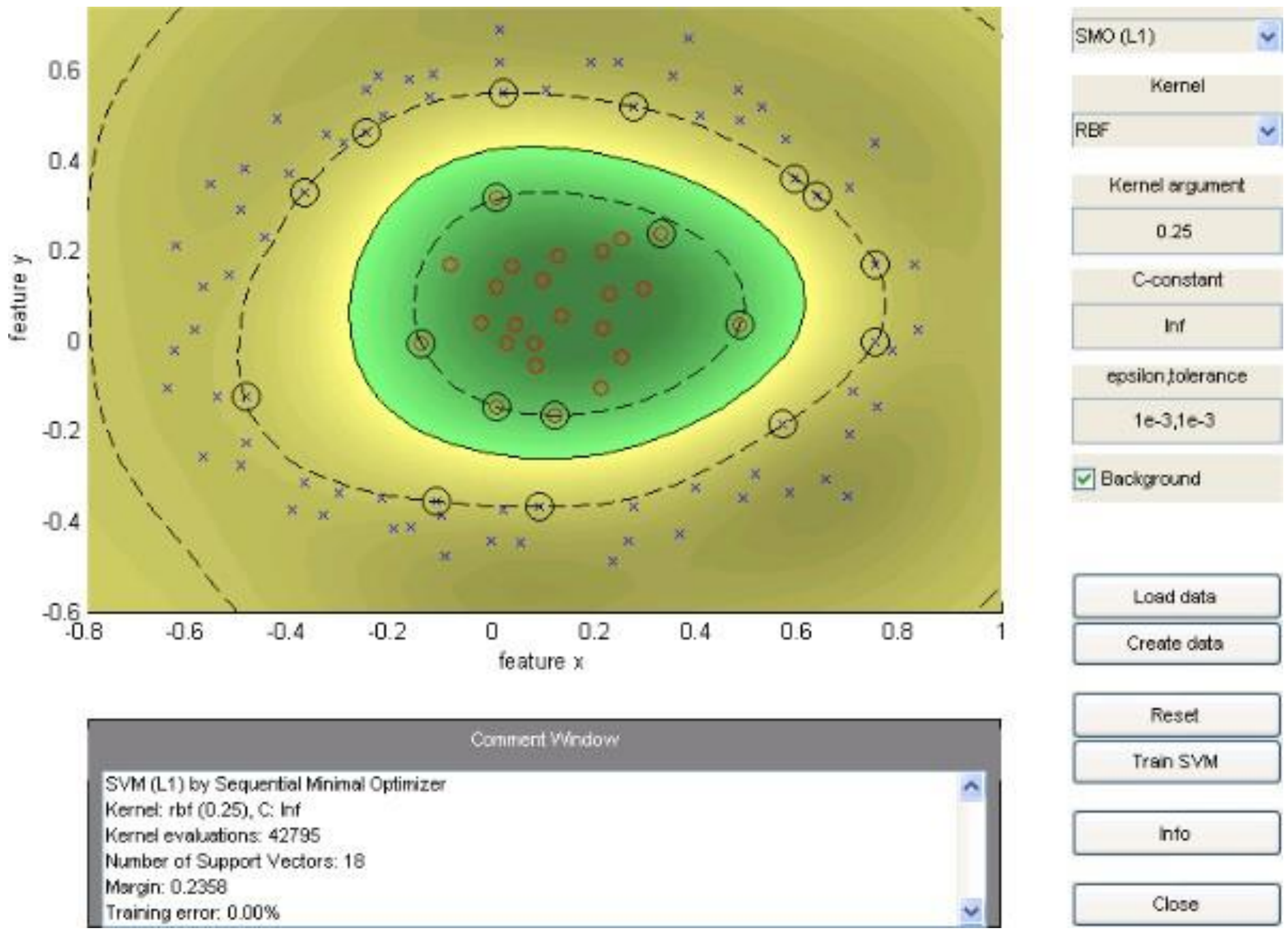
EXAMPLE PARAMS

- C = 10, γ = 0.5

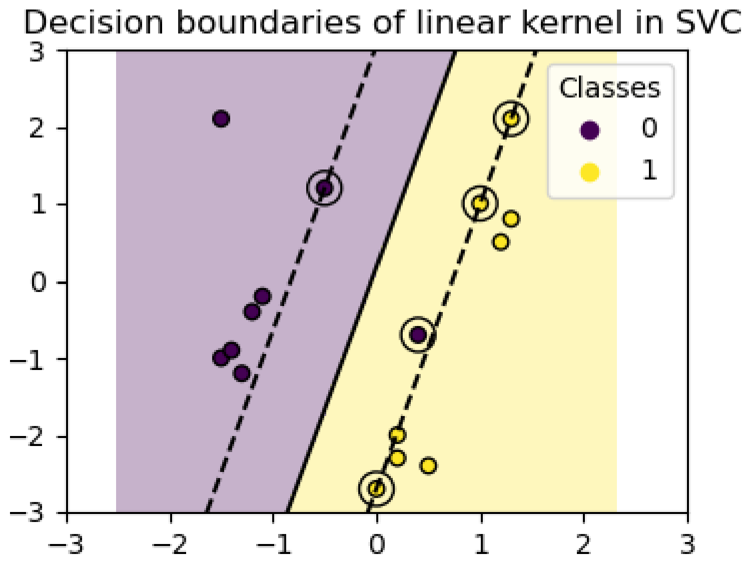


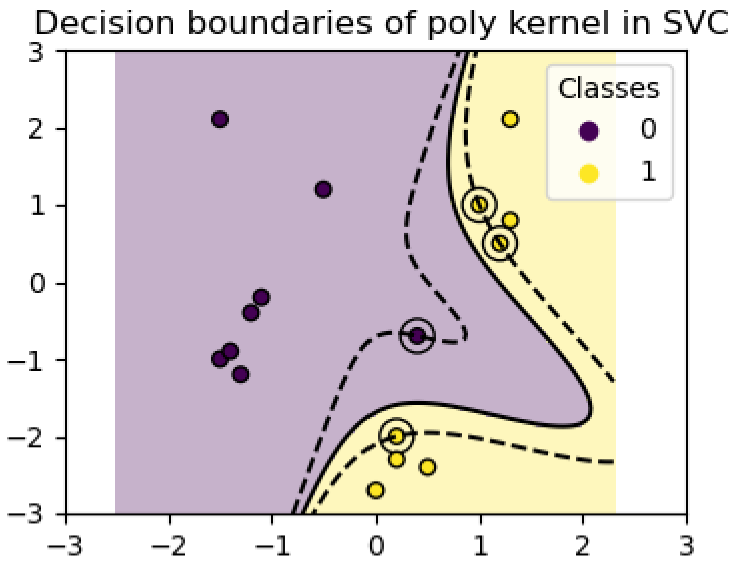
EXAMPLE PARAMS

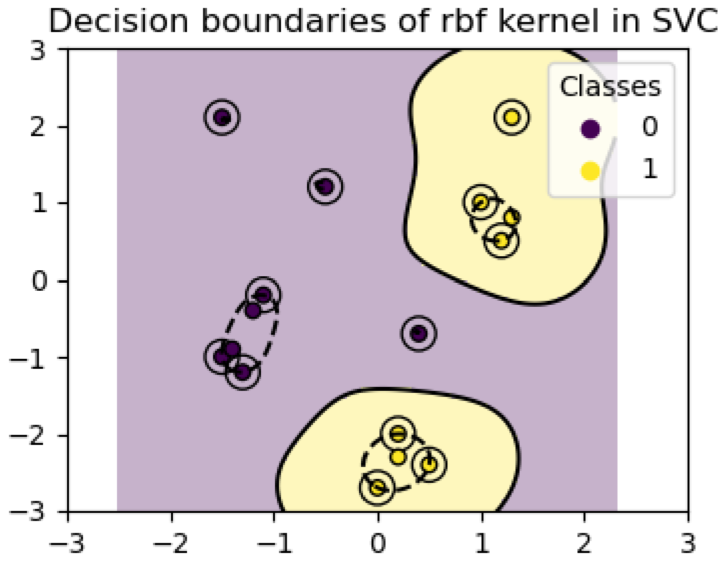
- C = 10, γ = 8
- The bigger γ gets, the closer towards nearest neighbour classifier the rbf kernel looks.

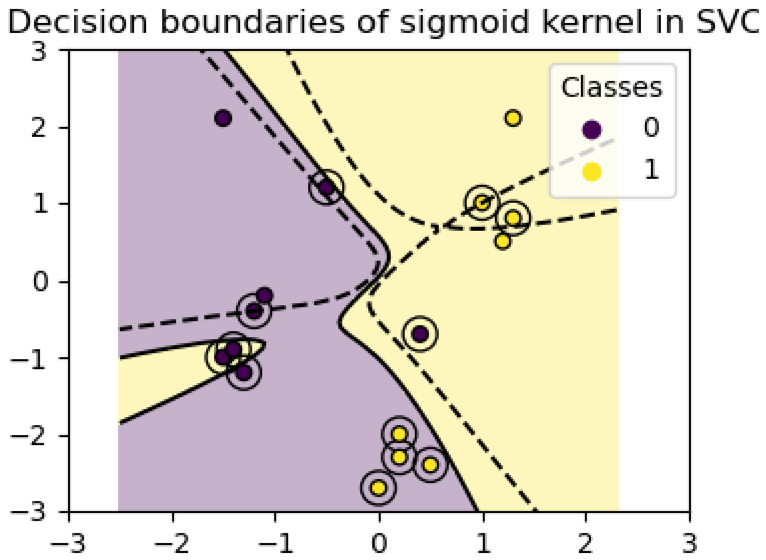


KERNEL SUMMARY



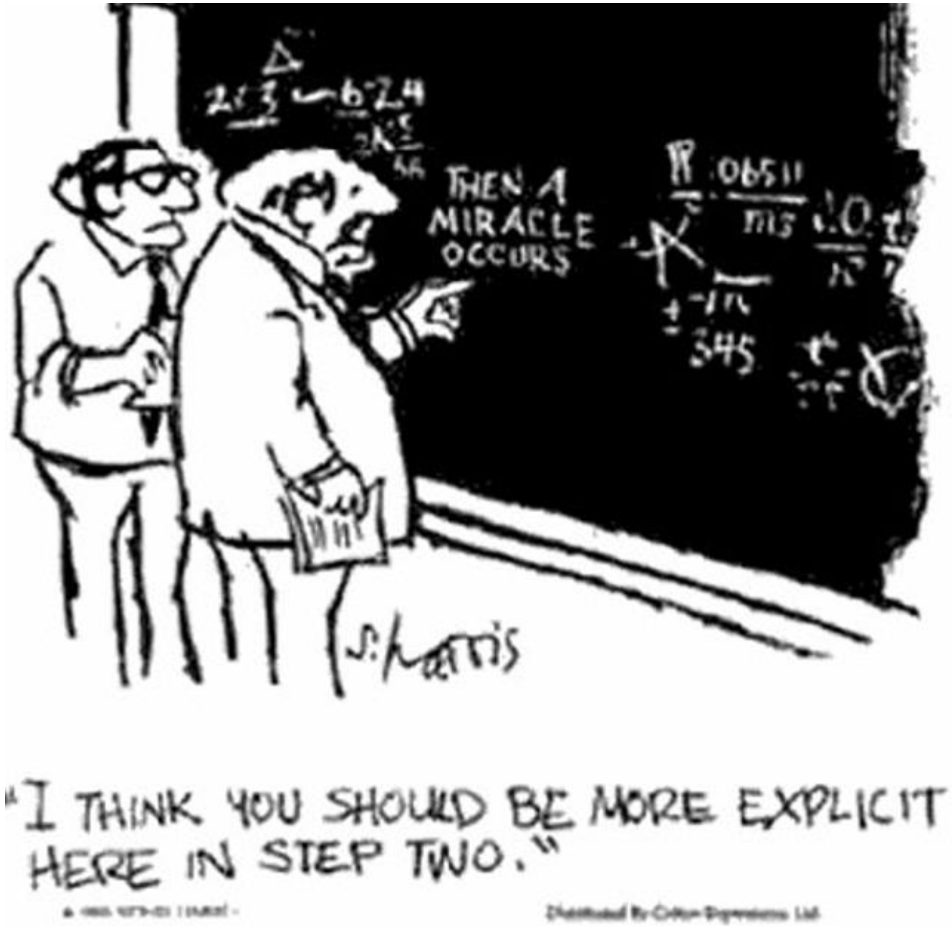






KERNEL TRICK - SUMMARY

- Classifiers can be learnt for high dimensional features spaces, without actually having to map the points into the high dimensional space
- Data may be linearly separable in the high dimensional space, but not linearly separable in the original feature space



SVM TIPS AND TRICKS

- SVMs are not scale invariant
- Check if your library normalizes by default
- Normalize your data
  - mean: 0 , stddev: 1
  - map to [0,1] or [-1,1]
  - StandardScaler, MinMaxScaler
- Normalize test set in same way

PARAMETER TUNING

- Given a classification task
  - Which kernel?
  - Which kernel parameter values (γ)?
  - Which value for C ?
- *GridSearchCV* can be used for trying the different combinations
- By default, GridSearchCV does a 5-Fold Cross Validation test