Ex2

1. Theoretical Part

1.1 Hard & Soft SVM

1. let's define: $V = \begin{pmatrix} b \\ w_1 \\ \vdots \\ w_d \end{pmatrix} \in \mathbb{R}^{d+1}$

from the hint we know: $\|w\|^2 = w^T I_d w$ and now we can use V instead of W, lets define $Q = \begin{bmatrix} I_d & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{(d+1)\times(d+1)}$

$$[w_1 \cdots w_d] \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_d \end{pmatrix} = [w_1 \cdots w_d, b] \begin{bmatrix} I_d & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_d \\ b \end{pmatrix} = V^T Q V$$

for the conditin: $y_i(\langle w, x_i \rangle + b) \geq 1 \Leftrightarrow y_i(V^T[1, x_i]) \geq 1 \Leftrightarrow$

$$-y_i(V^T[1, x_i]) \leq -1$$

so lets define $d = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix} \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times d}$ s.t $A_i = -y_i \cdot (V^T[1, x_i])$

and $a = 0$ because that isn't non-linear term in $\|w\|^2$

now we get that $\arg\min_{(w,b)} \|w\|^2 = \arg\min_{(w,b)} \frac{1}{2} V^T Q V$

we can multiply by $\frac{1}{2}$ because this will give the same solution for the optimzation problem

1.2 PAC Learnability

1. (a) $\left(1 + \frac{x}{n}\right)^n \leq e^x$

$x \in \mathbb{R}, x > -1 \Rightarrow \ln(1+x) \leq x \Rightarrow 1 + nx \leq e^{nx}$ $\quad x_1, \ldots, x_m \in S_m$

$$P_D[S_m \cap X' = \emptyset] = P_D\left[\bigcap_{i=1}^m x_i \notin X'\right] \overset{i.i.d}{=} \prod_{i=1}^m P_D[x_i \notin X'] = \left(1 - P_D[x \in X']\right)^m$$

*

We know that $P_D[x_i \in X'] > \varepsilon \Leftrightarrow P_D[x_i \notin X'] \leq \varepsilon$

$$1 - P_D[x \in X'] \leq 1 - \varepsilon \Rightarrow \left(1 - P_D[x \in X']\right)^m \leq (1-\varepsilon)^m$$

know from Bernoulli's inequality $(1-\varepsilon)^m \leq e^{-m\varepsilon}$

∎

(b) let's define $f :=$ the real function

$$Z_{\hat{f}_S} = \{x \in X \text{ s.t } \hat{f}_S(x) \neq f(x)\}$$

$$H_{Bad} := \{\hat{f} \in H \text{ s.t } L_{D,f}(\hat{f}) > \varepsilon\}$$

we can see that by definition that mistake can

happen if $S_m \cap Z_{\hat{f}_S} = \emptyset$. Then we can get $L_{S,f}(\hat{f}_{Sm}) = 0$ and $f \neq \hat{f}_S$

The probability to get a set like that

is the one from (a) meaning: $\mathbb{P}[S_m \cap Z_{f_S} = \emptyset] \leq e^{-\varepsilon m}$

so $\{L_{D,f}(\hat{f}_S) > \varepsilon\} \subseteq [\bigcup_{h \in H_{bad}} L_{sm}(\hat{f}_S) = 0\}$

because by definition this is all the functions that

$L_{D,f}(\hat{f}) > \varepsilon$ mistake will be made $L_{sm}(\hat{f}_S) = 0$

$$\Rightarrow \mathbb{P}_D(\{L_{D,f}(\hat{f}_S) > \varepsilon\}) \leq \mathbb{P}_D([\bigcup_{h \in H_{bad}} L_{sm}(\hat{f}_S) = 0\})$$

$$\leq \sum_{\hat{f}_S \in H_{Bad}} \mathbb{P}_D[S_m \cap Z_{\hat{f}_S} = \emptyset] \leq |H| \cdot e^{-m\varepsilon}$$

(c) From (b) we know that $\mathbb{P}_D[L_{DF}(A(S_m)) > \varepsilon] \leq |H| \cdot e^{-m\varepsilon}$

we need m that $|H| \cdot e^{-m\varepsilon} \leq \delta$

$$\Uparrow$$

$$-m\varepsilon = \log(e^{-m\varepsilon}) \leq \log\left(\frac{\delta}{|H|}\right)$$

$$\Uparrow$$

$$m\varepsilon \geq -\log\left(\frac{\delta}{|H|}\right) = \log|H| - \log\delta =$$

$$\Updownarrow \qquad \Downarrow$$

$$\log 1 - \log\delta = \log\left(\frac{1}{\delta}\right)$$

Holds for every $m_0 > m$ $\quad m \geq \dfrac{\log|H| + \log\frac{1}{\delta}}{\varepsilon}$

We still hold the conditions

$$\mathbb{P}_D[L_{DF}(A(S_m)) > \varepsilon] \leq |H| \cdot e^{-m\varepsilon} \leq \delta$$

$$\Updownarrow$$

$$\mathbb{P}_D[L_{DF}(A(S_m)) > \varepsilon] > 1 - \delta$$

$\blacksquare$

2. $H := \{h_r : r \in \mathbb{R}^+\}$ $\quad h(x) = \mathbb{1}_{[\|x\|_2 \le r]}$

let A be an algorithm that for every S_m Sample over X choose $x_0 \in S_m$ s.t $(\forall x \in S_m \ \|x_0\| \ge \|x\| \cap y_0 = 1)$
we want the max_r that we know that his label 1

the algorithm will label sample $h(x) = 1$ if $X \le max\_r$
0. otherwise how we know that for ever sampl labeld 1 the algorithm is correct but we still have The margin between $max\_r \le \|x\| \le real\_r$ that mistakenly labeld 0
lets define $Z_n = \{r_x : max\_r \le \|x\| \le real\_x\}$ the "bad samples" the algorithm hold to all th conditions in Q.1
From 1.(a) we know

$$\mathbb{P}(L_{D,r\_real}(h_x) > \varepsilon) \overset{i.i.d}{=} 1 - \mathbb{P}(L_{D,real}(h_x) \le \varepsilon) \le (1-\varepsilon)^m \le e^{-m\varepsilon}$$

and all we need now is δ that holds for this condition $e^{-m\varepsilon} \le \delta \iff m(\delta, \varepsilon) \ge \dfrac{\log(\frac{1}{\delta})}{\varepsilon}$

∎

3. Given that $\mathcal{H}$ has UCP that is $m_{\mathcal{H}}^{UC}: (0,1)^2 \to \mathbb{N}$
s.t $\forall \varepsilon, \delta \in (0,1)$ $\forall D$ over $X \times Y$ $P_D^m([S \subseteq (X \times Y)^m : S \to \varepsilon\text{-rep}) \geq 1-\delta$

lets define A with learning principle ERM
thus $A(S_m) = h_S = \underset{h \in \mathcal{H}}{\operatorname{argmin}} L_S(h)$

S is ε-representative, From the assumption $\mathcal{H}$-UCP
thus we can define $m_{\mathcal{H}}^{UC}(\varepsilon, \delta)$

S is ε-rep $\Rightarrow \forall h \in \mathcal{H}$ $|L_S(h) - L_D(h)| < \varepsilon$ with $P(\,) \geq 1-\delta$

let $h^n = \underset{h \in \mathcal{H}}{\operatorname{argmin}}(L_D(h))$ from UCP $\forall h \in \mathcal{H}$ $|L_S(h) - L_D(h)| < \varepsilon$

thus for h^n and for h_S
$|L_S(h^n) - L_D(h^n)| < \varepsilon \Leftrightarrow -\varepsilon < L_D(h^n) - L_S(h^n) < \varepsilon \Rightarrow L_S(h^n) < L_D(h^n) + \varepsilon$

for $\hat{h}$ also $L_D(\hat{h}) < L_S(\hat{h}) + \varepsilon$

From ERM h_S has the min loss over S $L_S(h_S) \leq L_S(h^n)$ (*)

Thus $L_D(h_S) \leq L_S(h_S) + \varepsilon \overset{(*)}{\leq} L_S(h^n) + \varepsilon \leq L_D(h^n) + \varepsilon + \varepsilon$

So for $\varepsilon_0 = \varepsilon/2$ we get $L_D(h_S) \leq \min_{h \in \mathcal{H}} L_D(h) + \varepsilon$

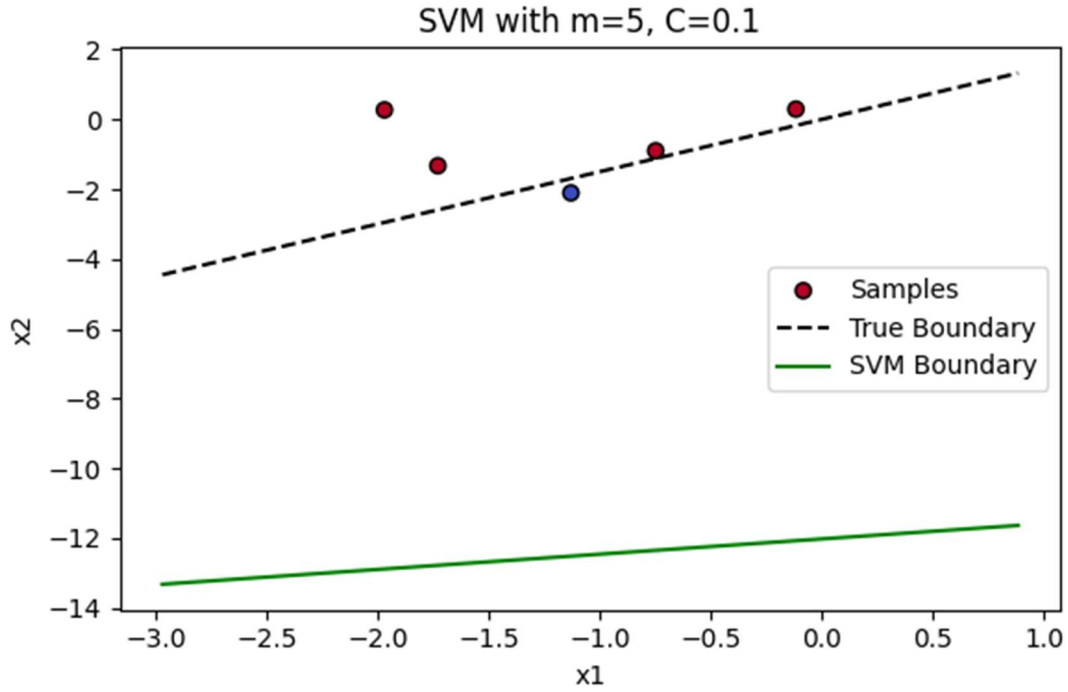
and thats happening in probability $> 1-\delta$
which satisfies the requirment of APAC ∎

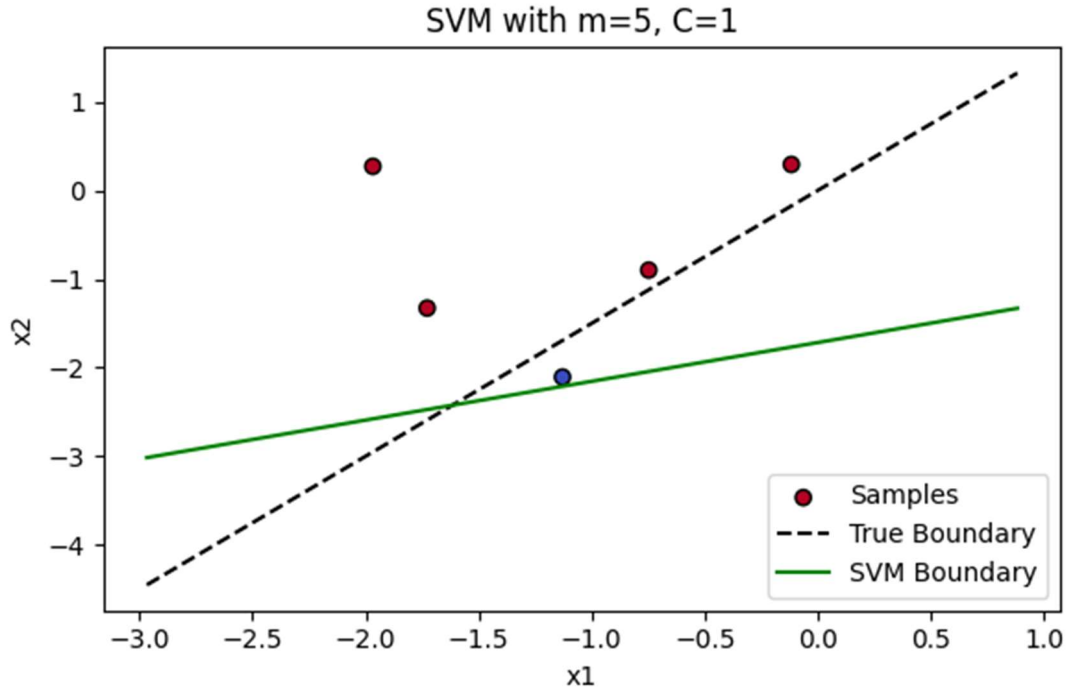
1.3 VC Dimension

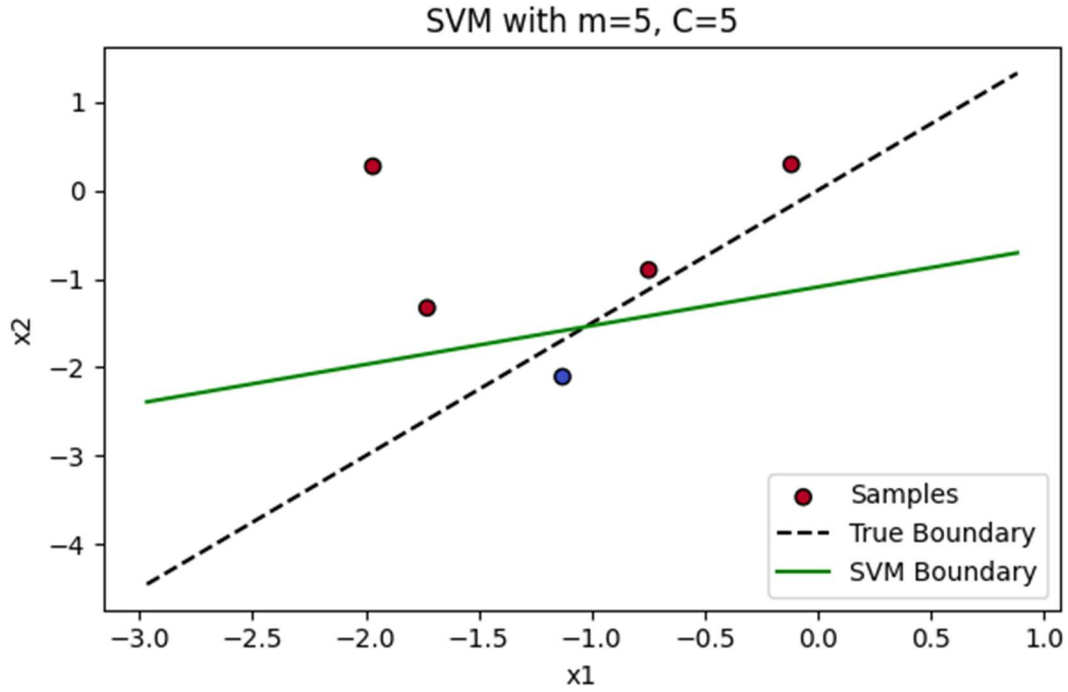
1. For sample of size n we need to show that some set can be shattered so we choose $[c_1, c_2 \cdots c_n]$ and defin $I := [i | y_i = 1]$ then if $y_i = 1$ $i \in I$ which means $h_I(b_i) = y_i = 1$ so it always odd The size of $\mathcal{H}$ is all the vector $[0,1]^n \in \mathbb{R}^n$ we know that the size of subsets $I \subseteq [n]$ is 2^n and in the reaction we saw that VC_dim cant be biger than is $\log(|H|) \Rightarrow VCdim(H) \leq \log(2^n) = n$

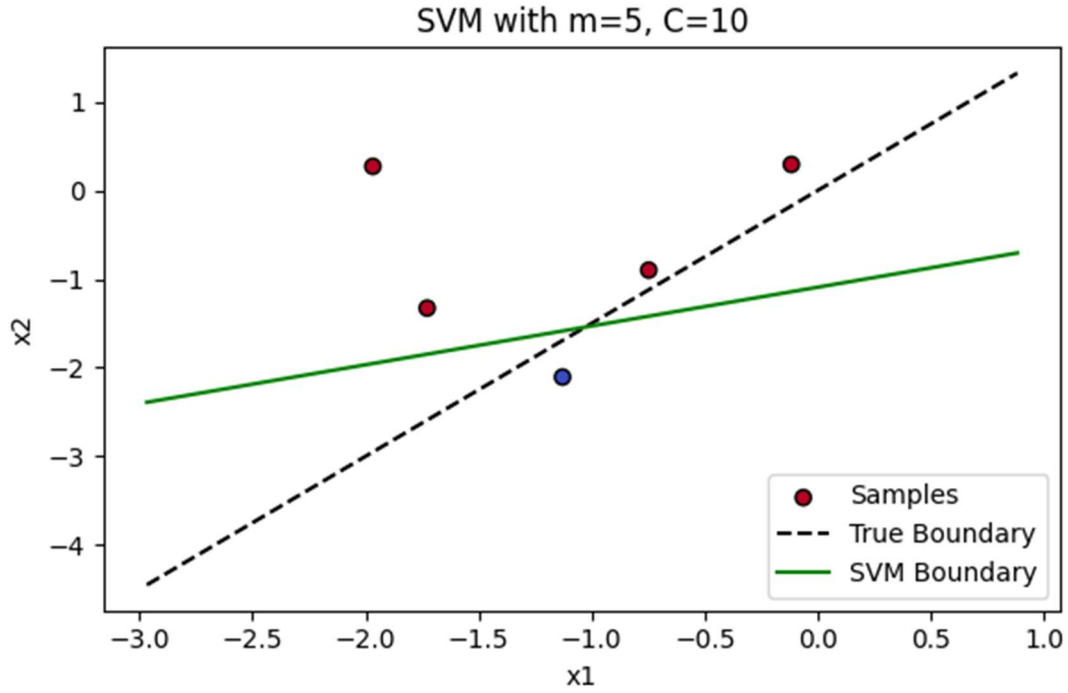
* $VCdim(H) \geq n \Rightarrow VCdim(H) = n$
* $VCdim(H) \leq n$

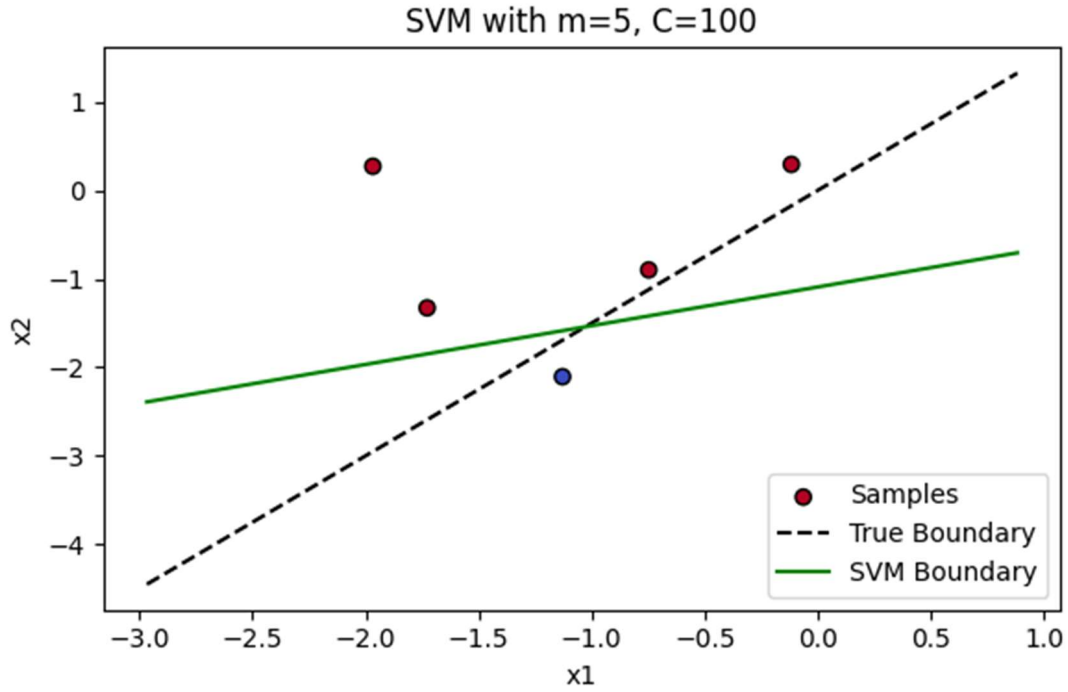
2. $H_1 \subseteq H_2$ then every $x \subseteq X$ s.t $x \in H_1$ if x shattered by H_1, $H_1 \subseteq H_2$ thus x shattered by $H_2 \Rightarrow VCdim(H_1) \leq VCdim(H_2)$

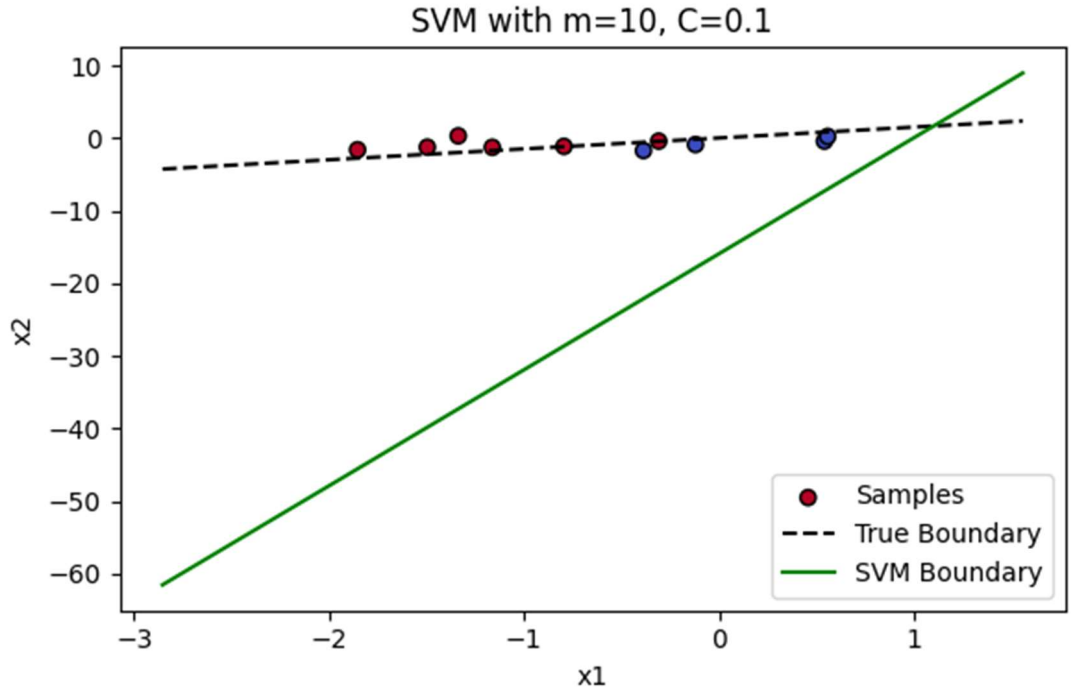

SVM with m=5, C=0.1
x1
x2
Samples
True Boundary
SVM Boundary

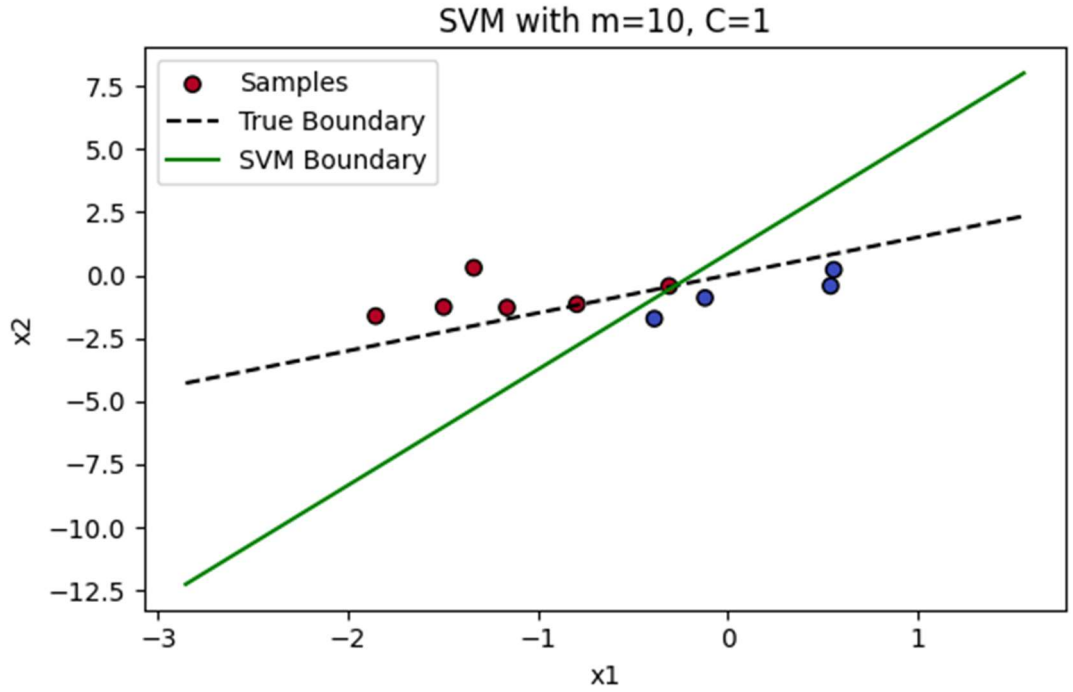

SVM with m=5, C=1
x1
x2
Samples
True Boundary
SVM Boundary

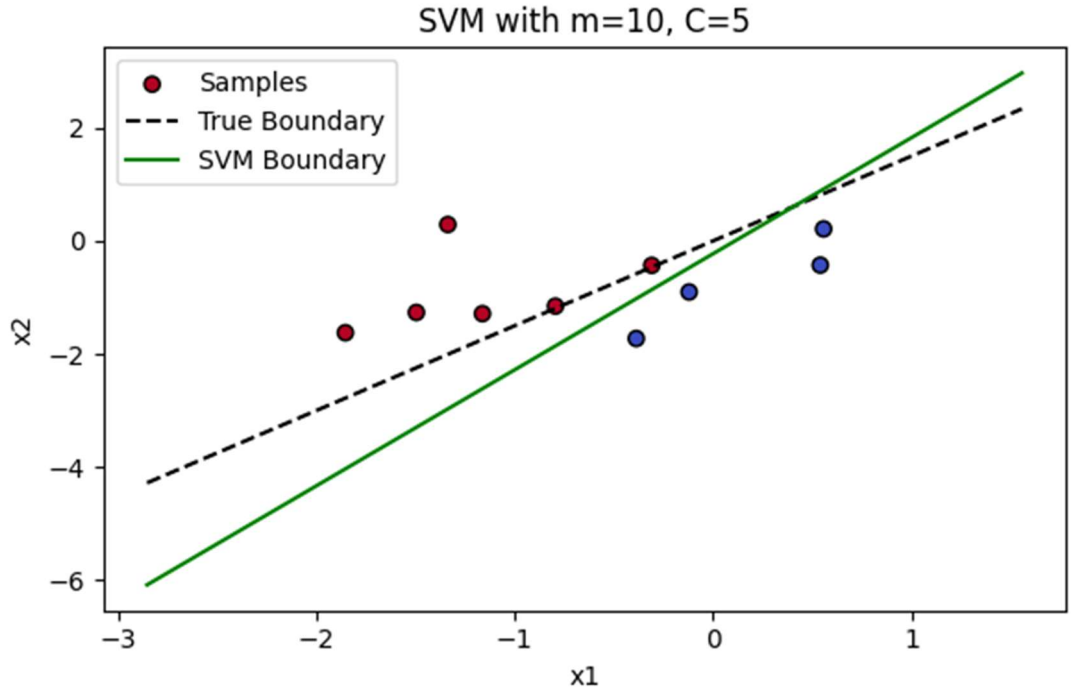

SVM with m=5, C=5
x2
x1
Samples
True Boundary
SVM Boundary

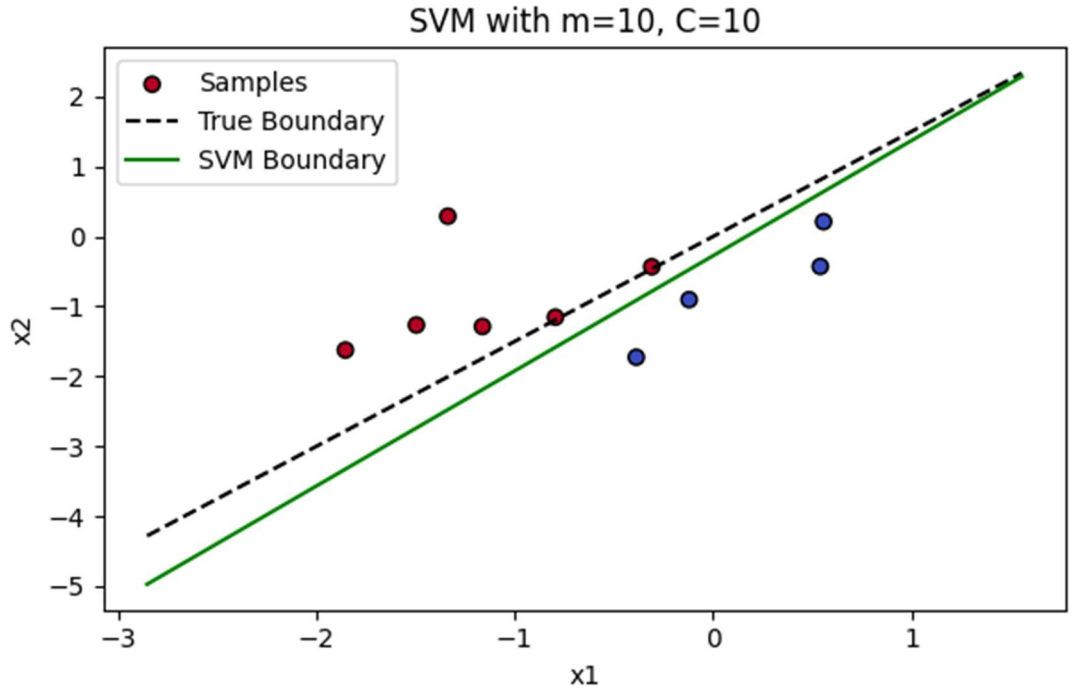

SVM with m=5, C=10
x2
x1
Samples
True Boundary
SVM Boundary

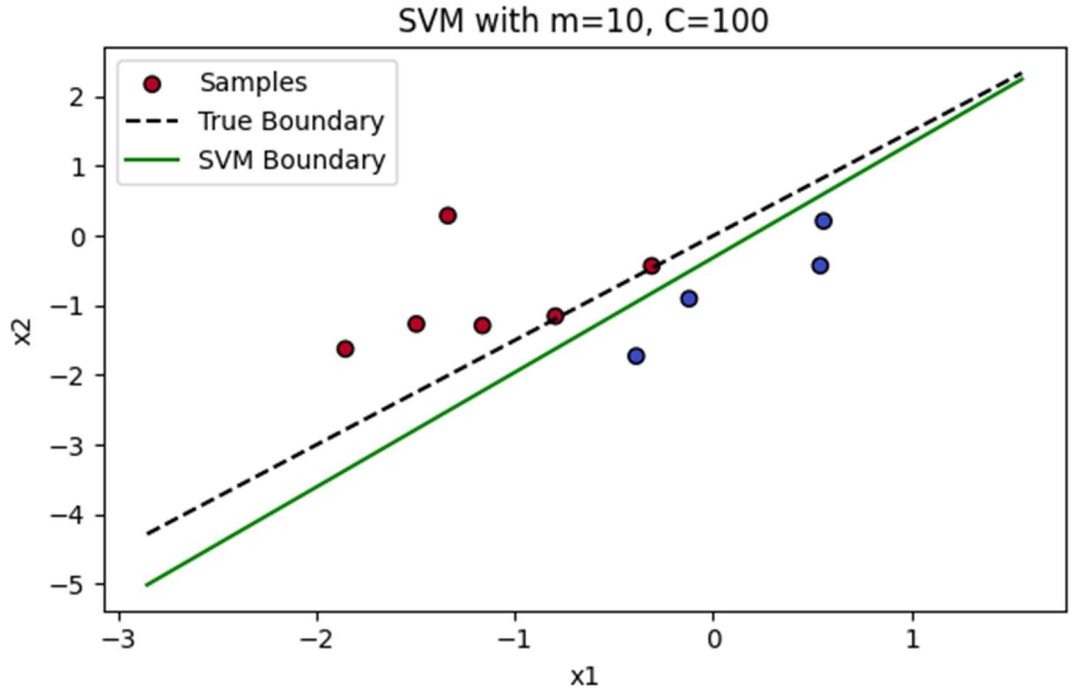

SVM with m=5, C=100
x1
x2
Samples
True Boundary
SVM Boundary

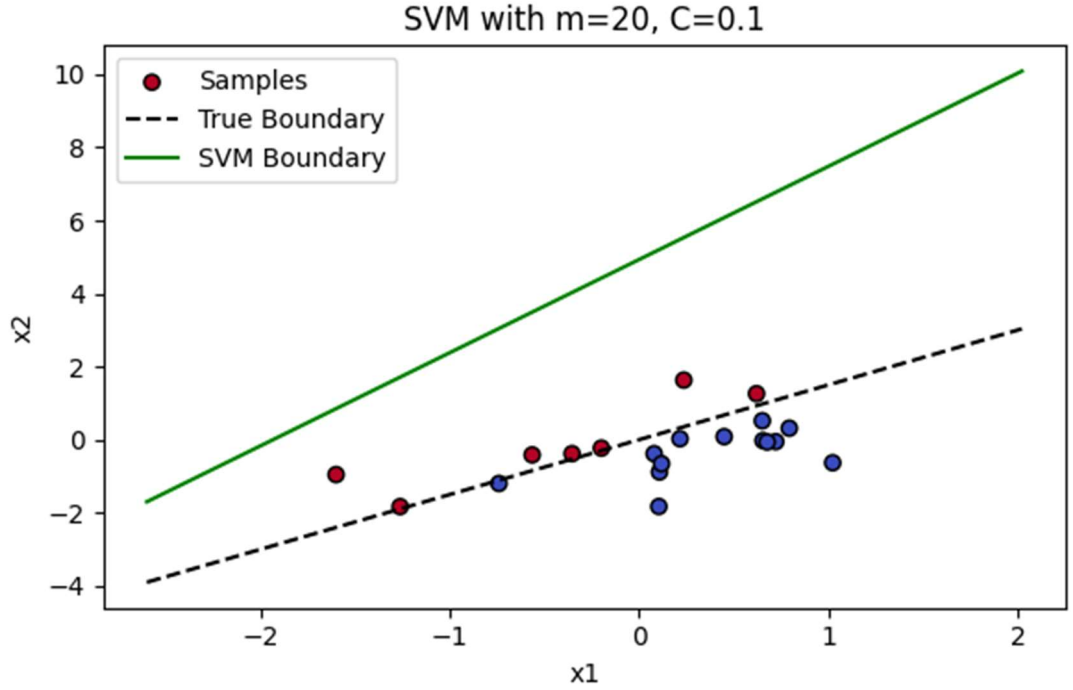

SVM with m=10, C=0.1
x1
x2
Samples
True Boundary
SVM Boundary

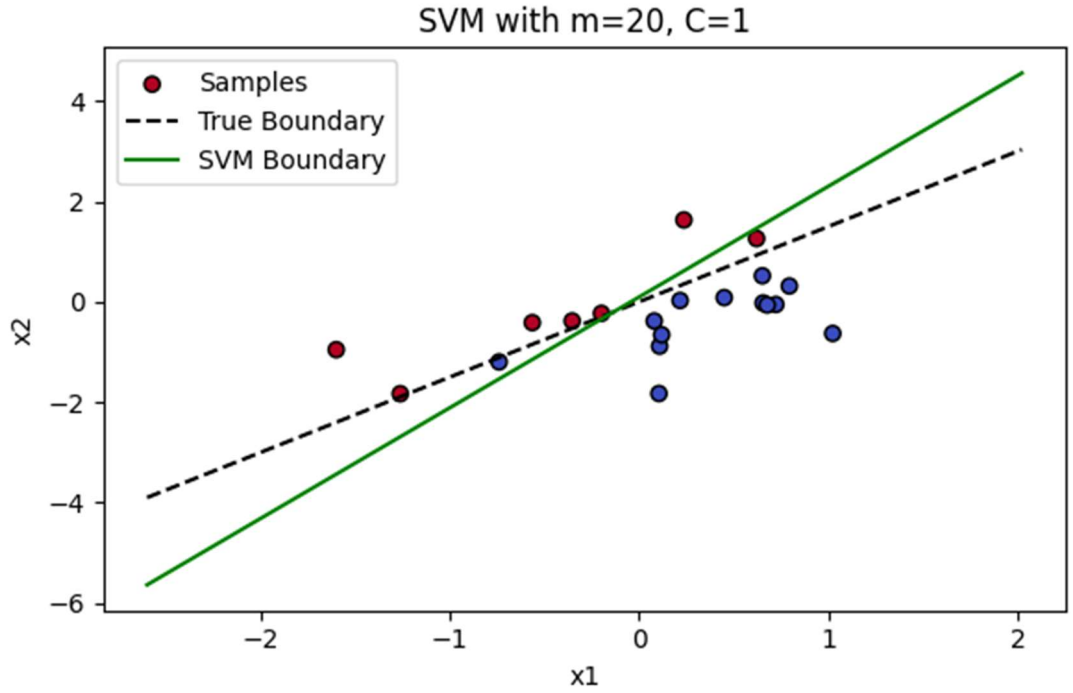

SVM with m=10, C=1
Samples
True Boundary
SVM Boundary
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
x2
−3
−2
−1
0
1
x1

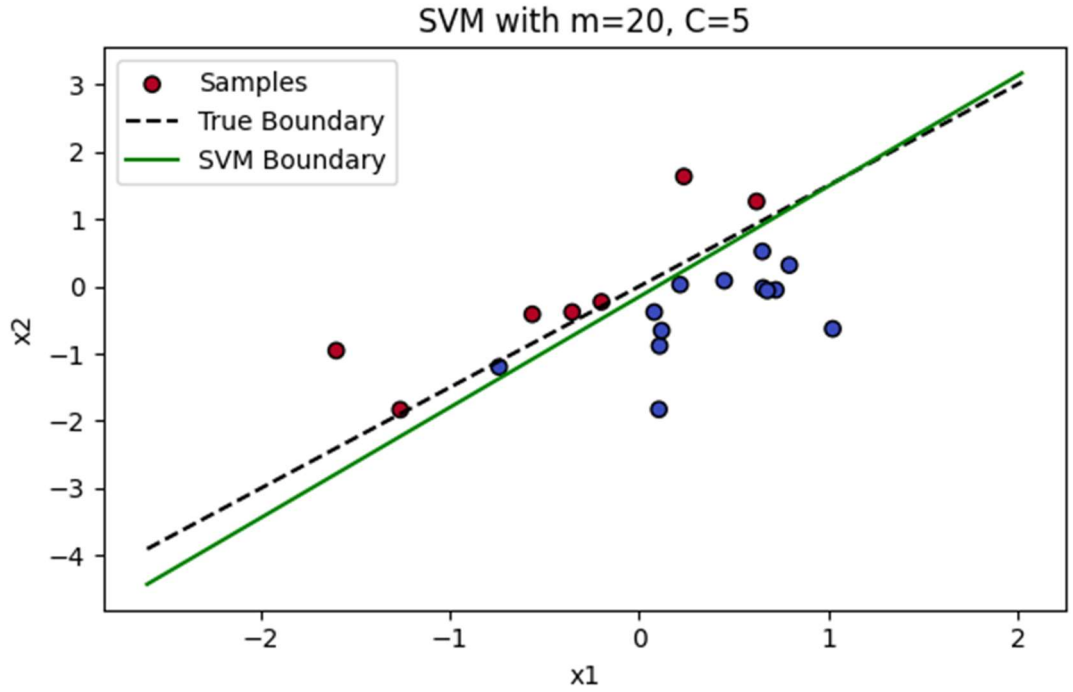

SVM with m=10, C=5
Samples
True Boundary
SVM Boundary
2
0
−2
−4
−6
x2
−3
−2
−1
0
1
x1

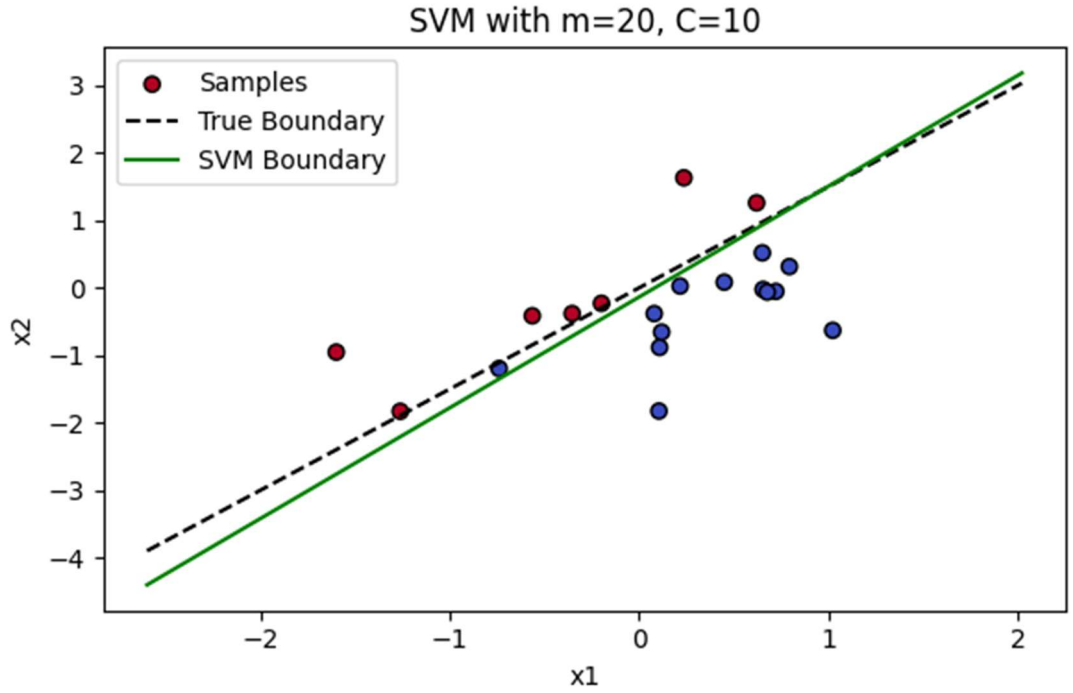

SVM with m=10, C=10
Samples
True Boundary
SVM Boundary
x2
x1
2
1
0
−1
−2
−3
−4
−5
−3
−2
−1
0
1

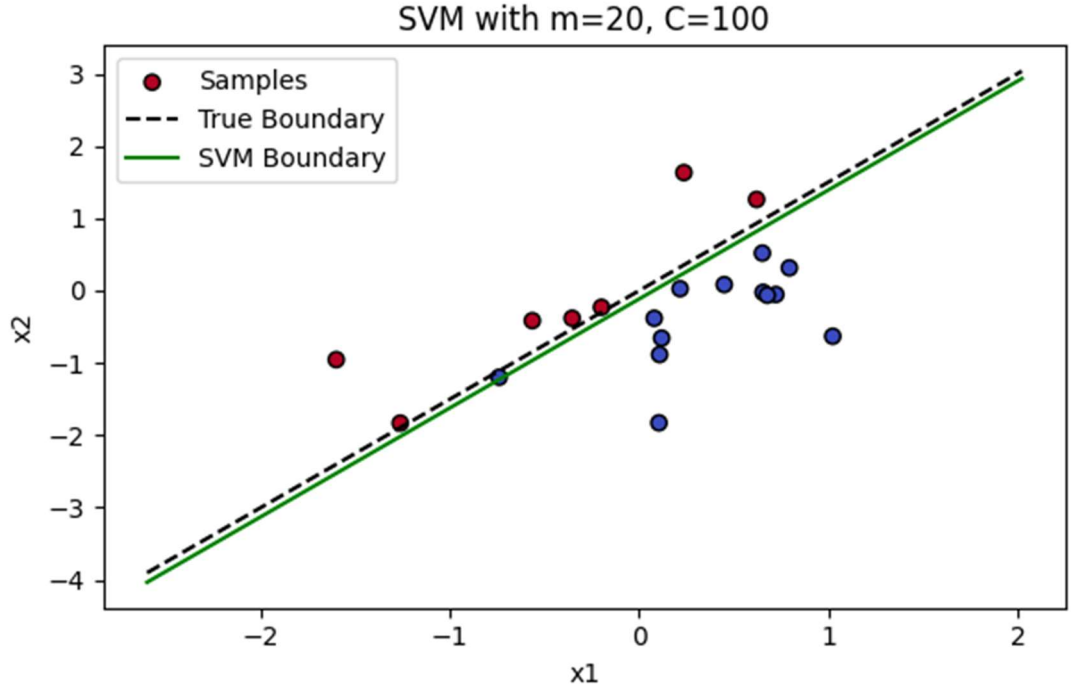

SVM with m=10, C=100
Samples
True Boundary
SVM Boundary
x2
x1
2
1
0
−1
−2
−3
−4
−5
−3
−2
−1
0
1

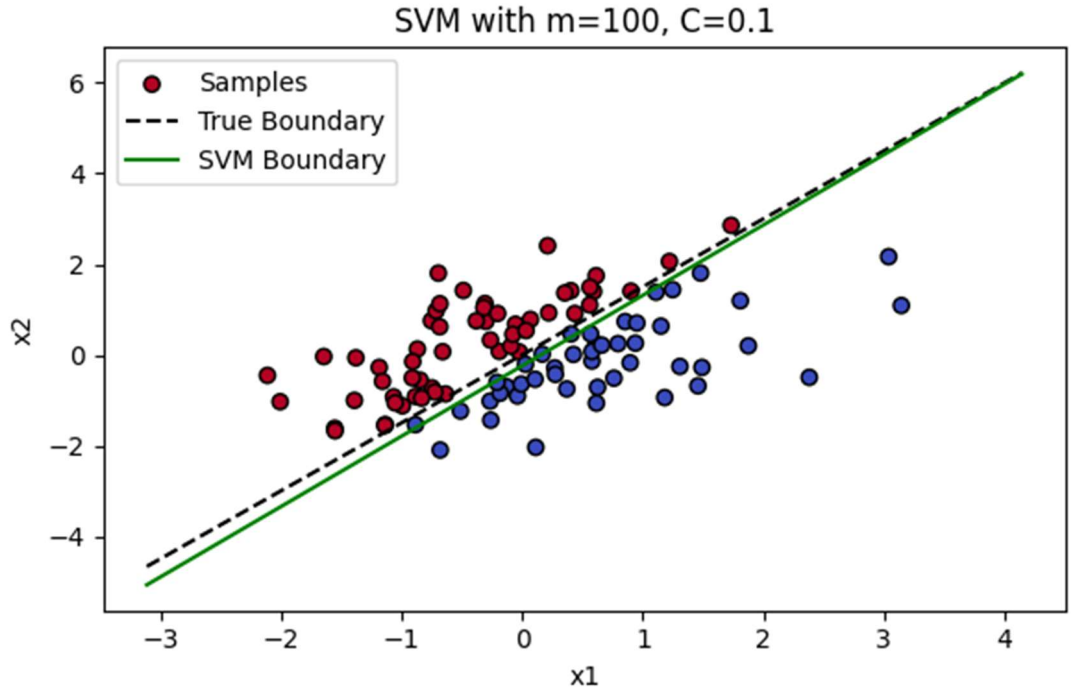

SVM with m=20, C=0.1
Samples
True Boundary
SVM Boundary
x1
x2

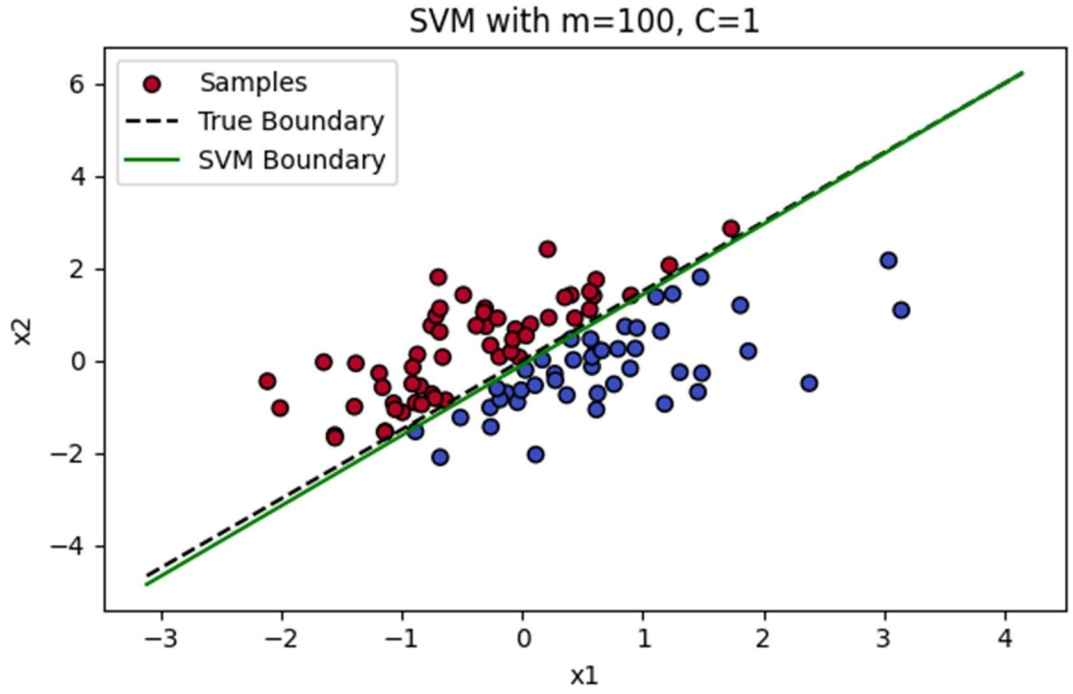

SVM with m=20, C=1
Samples
True Boundary
SVM Boundary
x1
x2

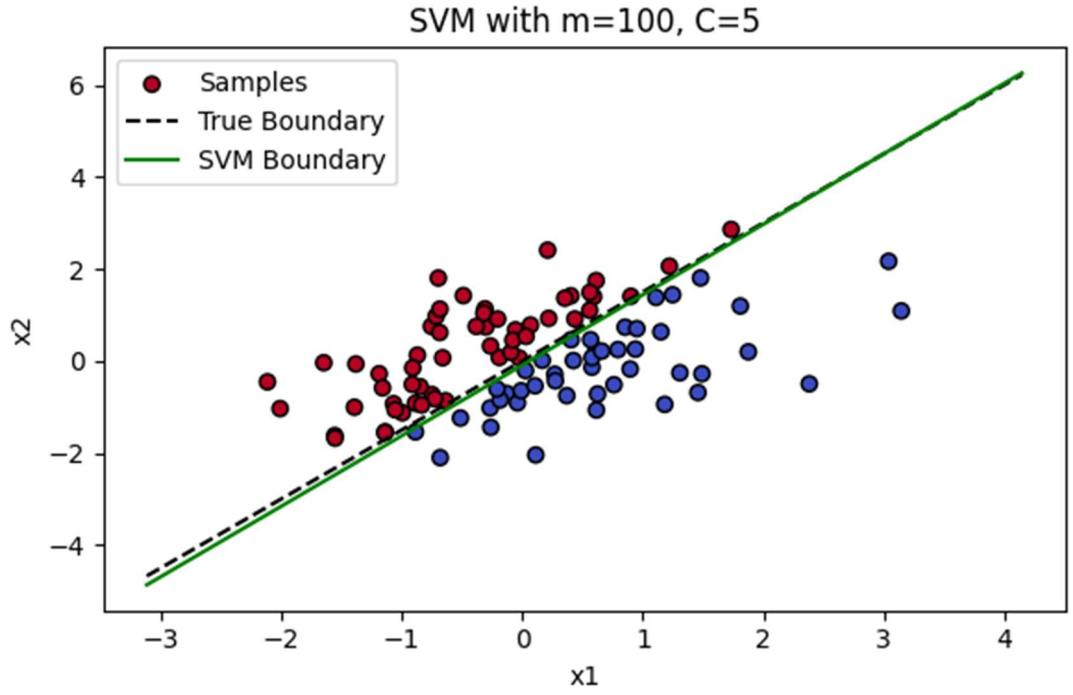

SVM with m=20, C=5
Samples
True Boundary
SVM Boundary
x1
x2

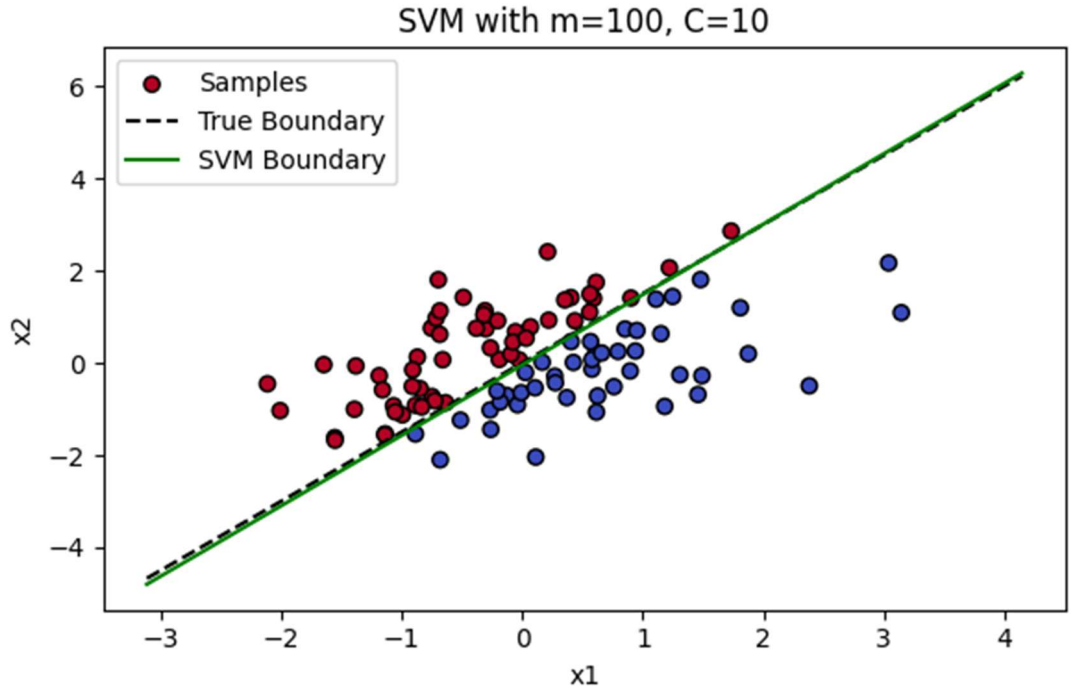

SVM with m=20, C=10
Samples
True Boundary
SVM Boundary
x1
x2

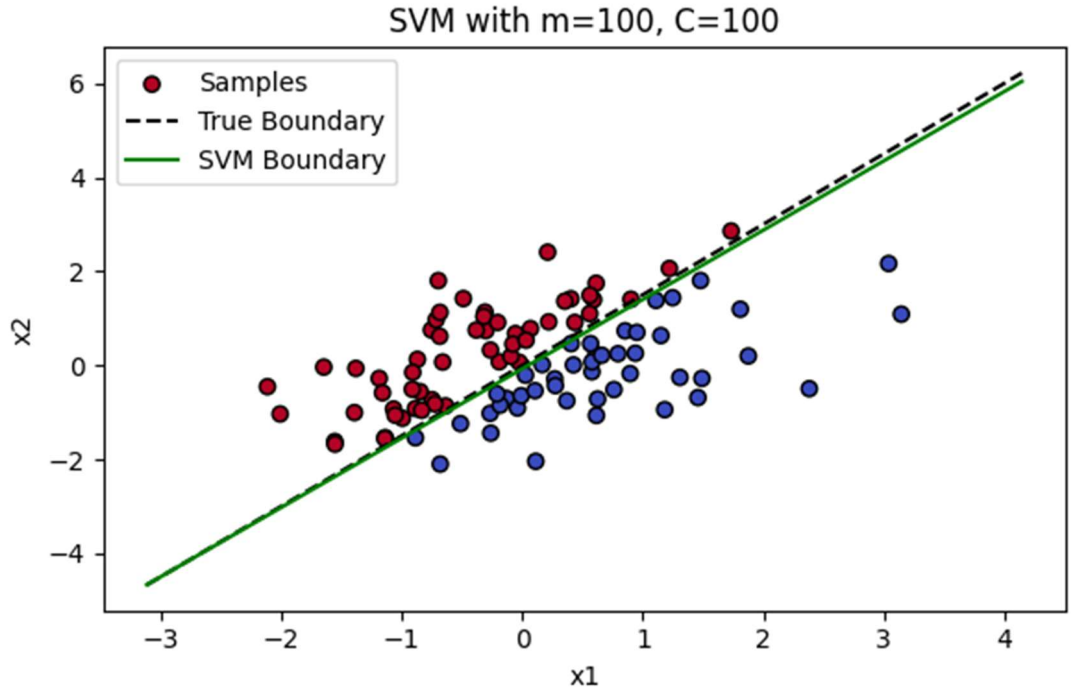

SVM with m=20, C=100
Samples
True Boundary
SVM Boundary
x1
x2


SVM with m=100, C=0.1
Samples
True Boundary
SVM Boundary
x1
x2


SVM with m=100, C=1
Samples
True Boundary
SVM Boundary
x1
x2


SVM with m=100, C=5
Samples
True Boundary
SVM Boundary
x1
x2


SVM with m=100, C=10
Samples
True Boundary
SVM Boundary
x1
x2


SVM with m=100, C=100
Samples
True Boundary
SVM Boundary
x1
x2

2

2.1 SVM

1.above

2.

We can see that as **C** gets bigger, the SVM line is more accurate.

For **C = 0.1**, we almost don't care about the mistakes — we just want the **w** with minimum norm,

so the learner will choose **ψ** (slack variables) that are very big without getting punished.

On the other hand, if **C = 100**,

then for every **ψ** we pay more, so the learner will search for the minimum **ψ** possible,

which could lead to overfitting.

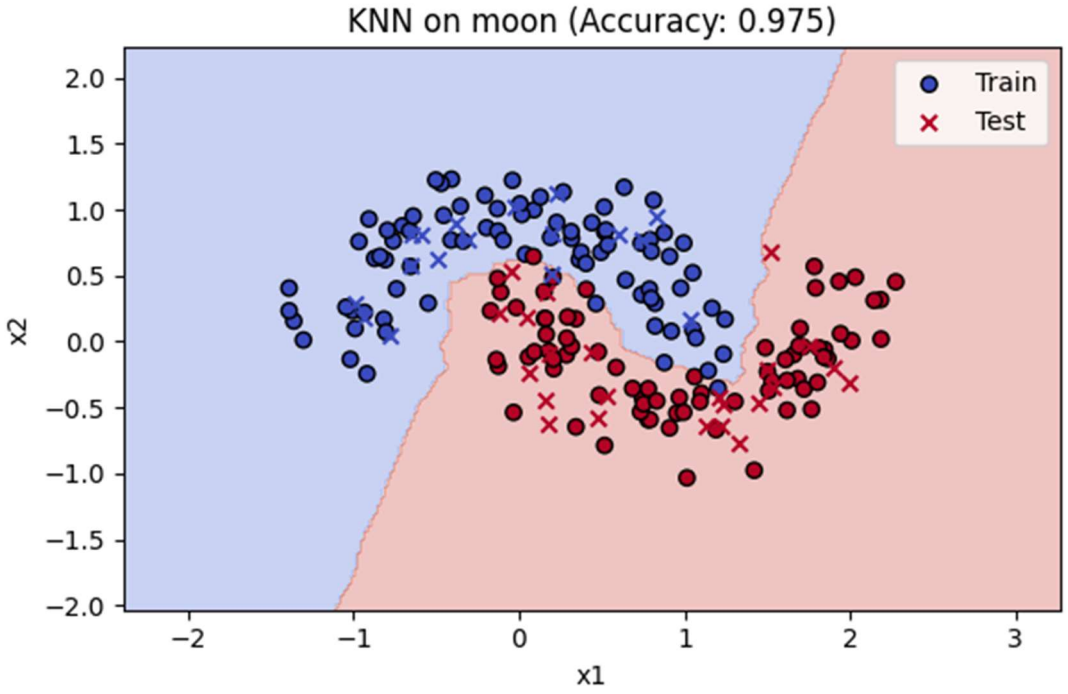
3.

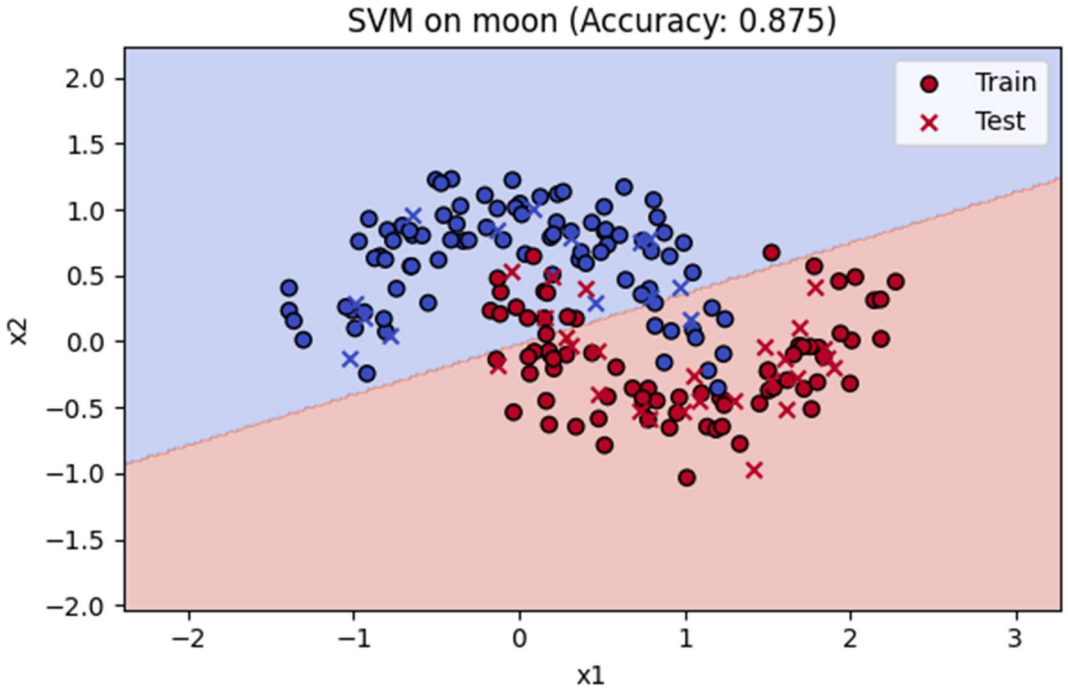
For **m**, as the sample size gets bigger, the SVM becomes more accurate for the same **C**.

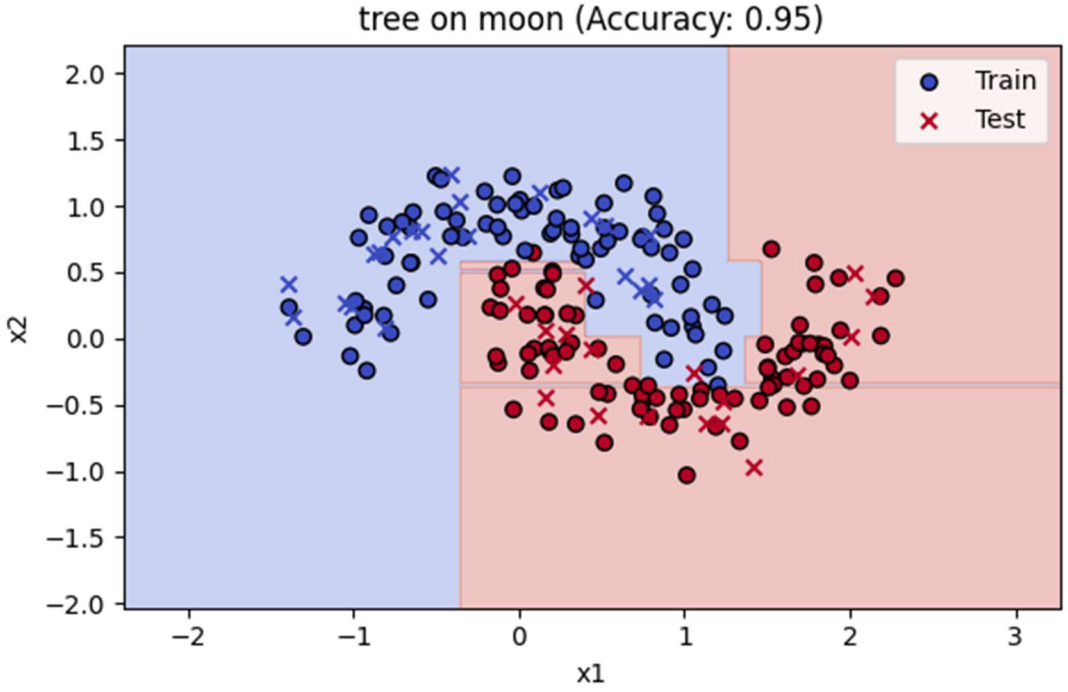
That's because in PAC learning there is a minimum **m** required for **H** to be PAC-learnable.

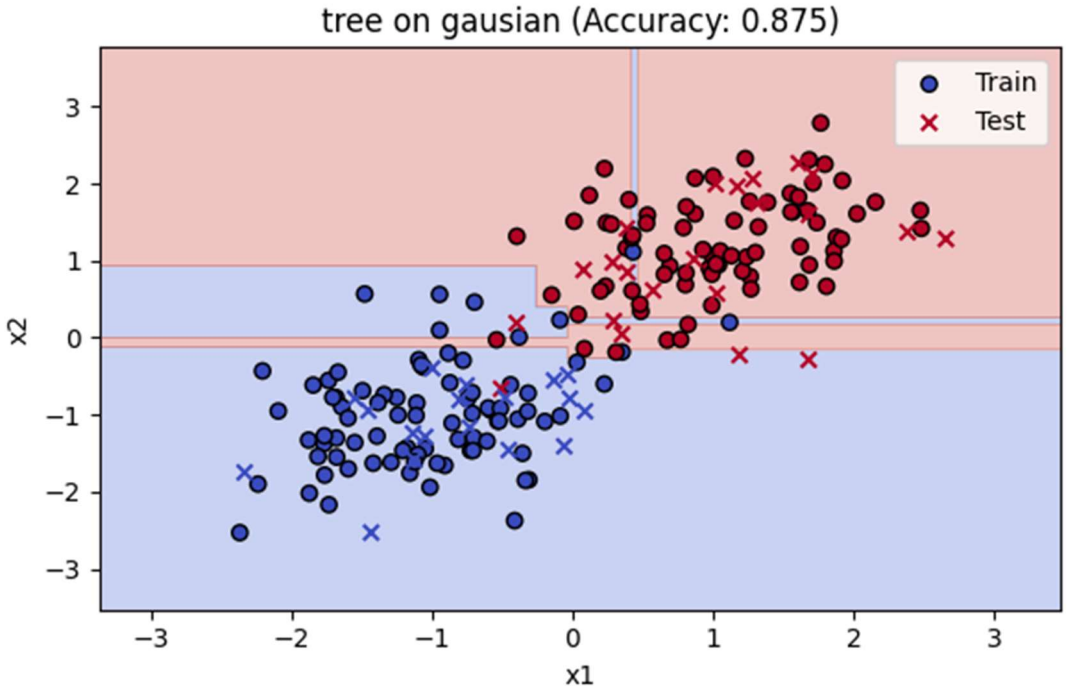
So as **m** increases, we approach that boundary such that:

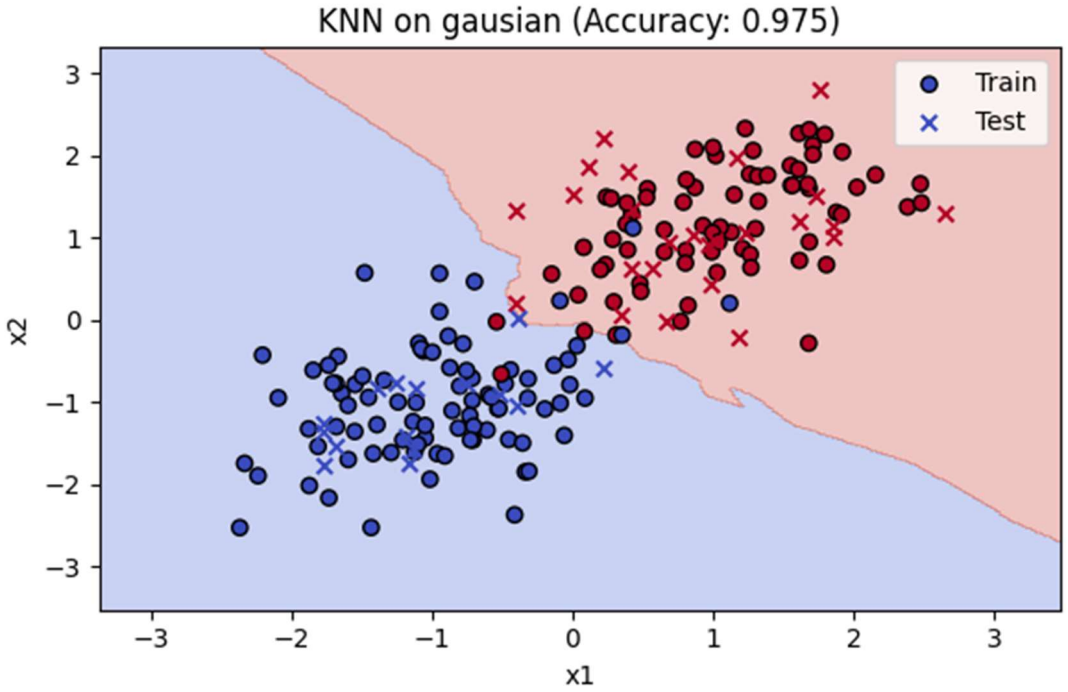
m > = to (log(|H|)*log(1/delta)/epsilon

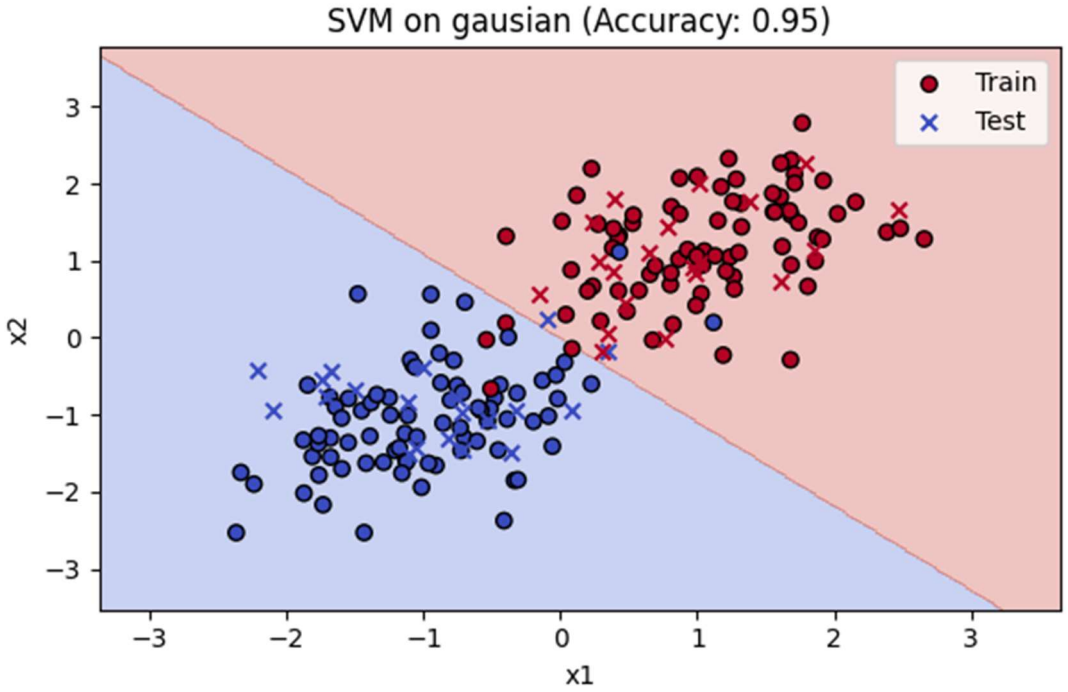

KNN on moon (Accuracy: 0.975)
Train
Test
x1
x2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2
−1
0
1
2
3

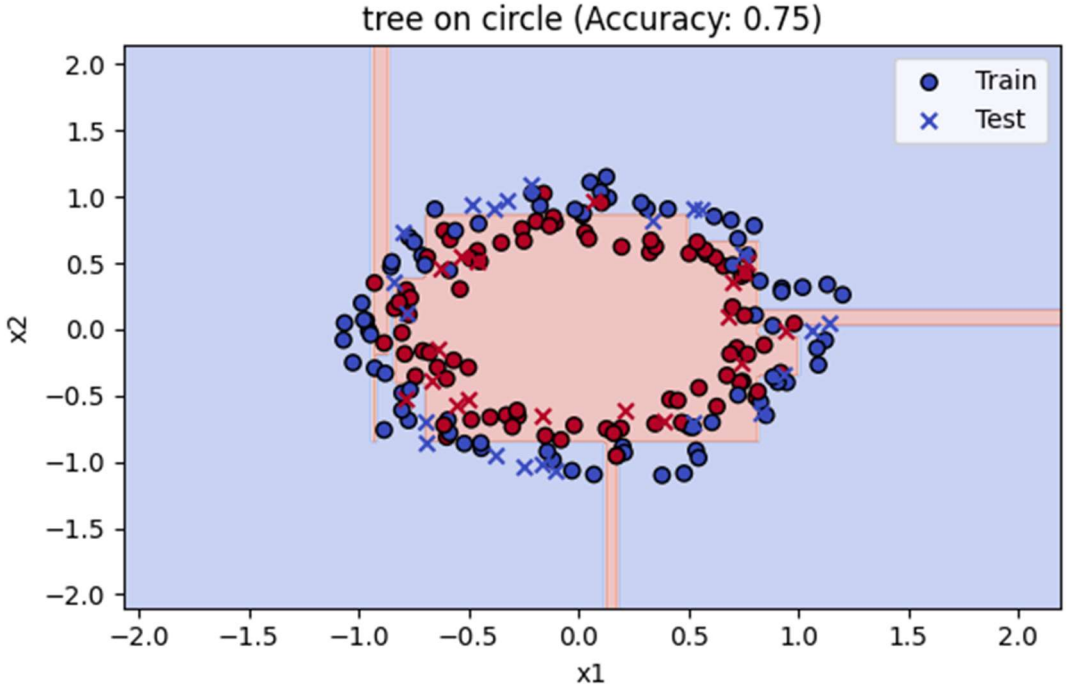

SVM on moon (Accuracy: 0.875)
Train
Test
x1
x2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2
−1
0
1
2
3

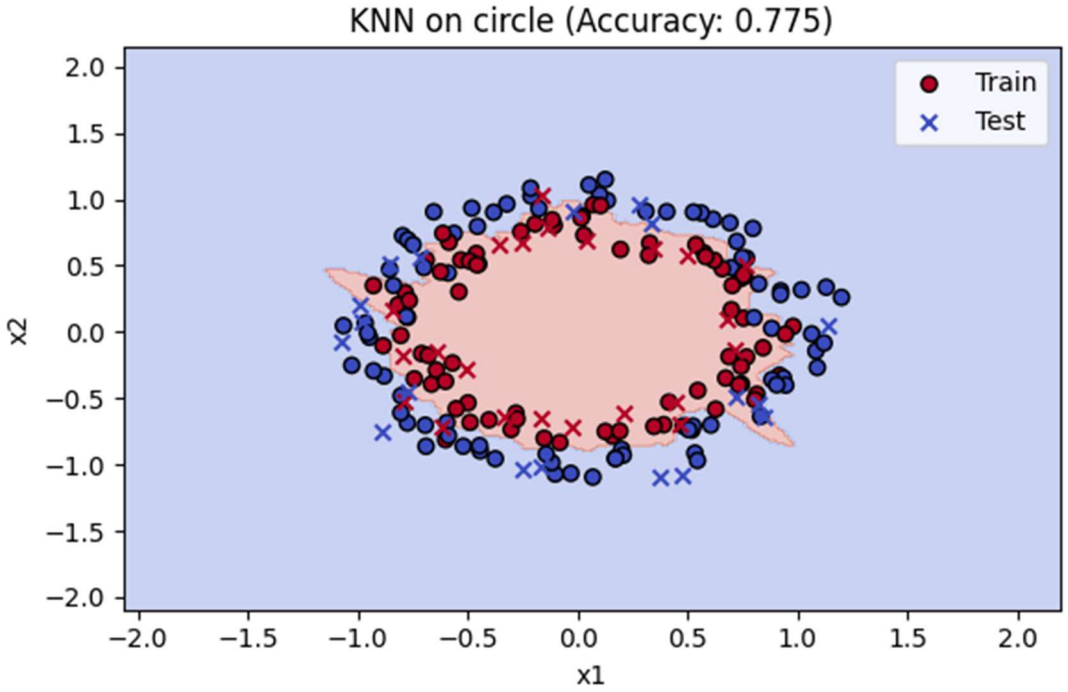

tree on moon (Accuracy: 0.95)
Train
Test
x1
x2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2
−1
0
1
2
3

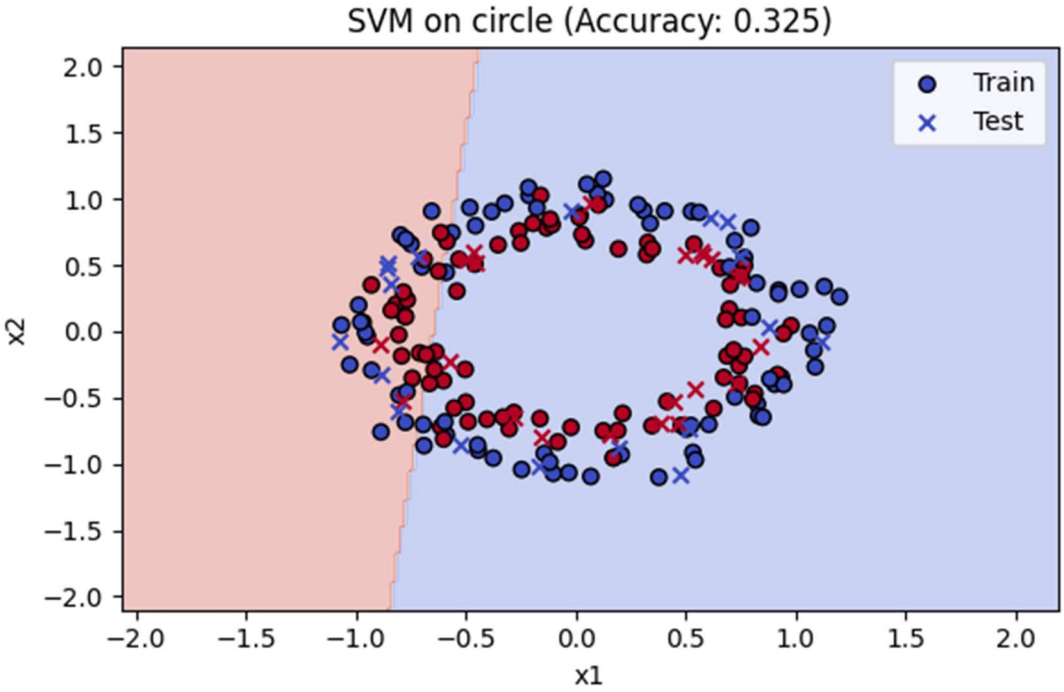

tree on gausian (Accuracy: 0.875)
Train
Test
x1
x2
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3


KNN on gausian (Accuracy: 0.975)
Train
Test
x1
x2


SVM on gausian (Accuracy: 0.95)
Train
Test
x1
x2


tree on circle (Accuracy: 0.75)
Train
Test
x1
x2


KNN on circle (Accuracy: 0.775)
Train
Test
x1
x2


SVM on circle (Accuracy: 0.325)
Train
Test
x1
x2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

2.2 Classification Boundaries and Model Comparison

1.abov

2.
Moon – KNN (acc = 0.975), Tree (acc = 0.95), SVM (acc = 0.875)
Because the data is not linear, SVM can't completely separate them.
Tree performs better because we can separate better by dividing into groups,
and KNN gives the best result because the data isn't mixed, which means the closest points to a point are the right choice.

Circle – KNN (acc = 0.775), Tree (acc = 0.75), SVM (acc = 0.325)
Here KNN performs best but still less accurately than KNN on moon because it's much less clear where one class starts and the other ends.
Tree also performs okay for the same reason.
But SVM performs poorly because for every **w** we choose, the distance from the data is the same.

Gaussian – KNN (acc = 0.975), Tree (acc = 0.875), SVM (acc = 0.95)
Tree performs worst because it only creates rectangles, which causes it to misclassify whole regions
unlike KNN that works best because the different classes are grouped in the same region.
That also allows SVM to stretch a line between them.

3.
SVM – It maximizes the margin between the two classes and is sensitive to noise and outliers.
We can use **delta** or **C** to decide how much we care about noise.

Tree – Stretches lines parallel to the axes with a pre-decided depth and splits them recursively. High k will cause overfitting while small one will mis classify

KNN – Based on the *k* nearest neighbors, we decide how to classify a point.
If we choose small *k*, it can cause overfitting,
but for large *k*, we can misclassify a lot.

I used chat gpt for typos here part of the ploting such as how to draw back ground and some docomentation

k