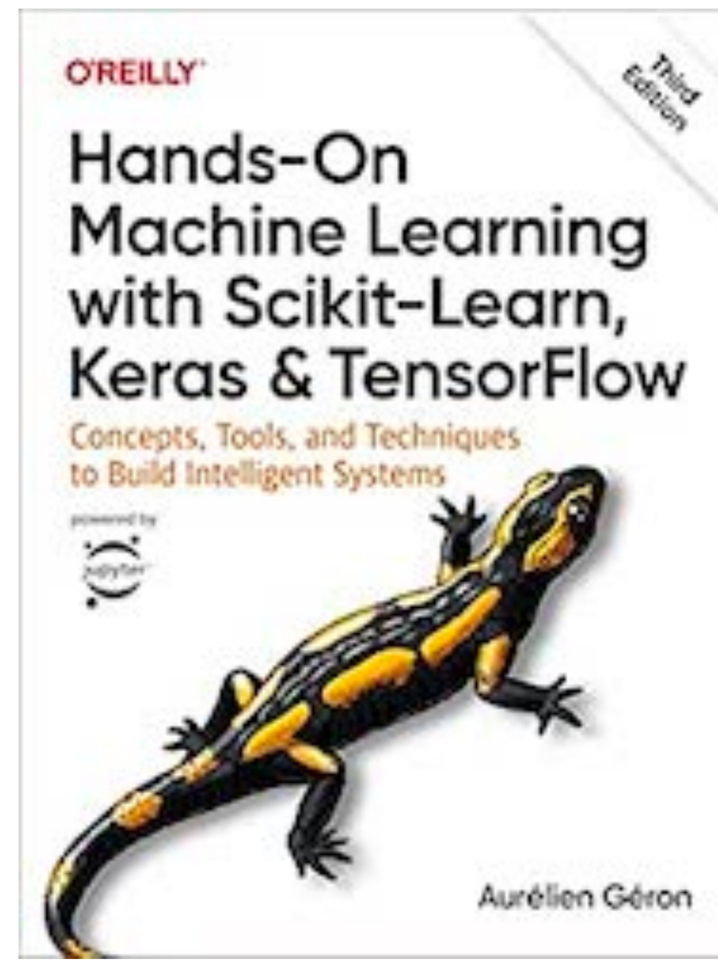
Machine Learning Security

5 Support Vector Machines



Updated Sep 23, 2023

Topics

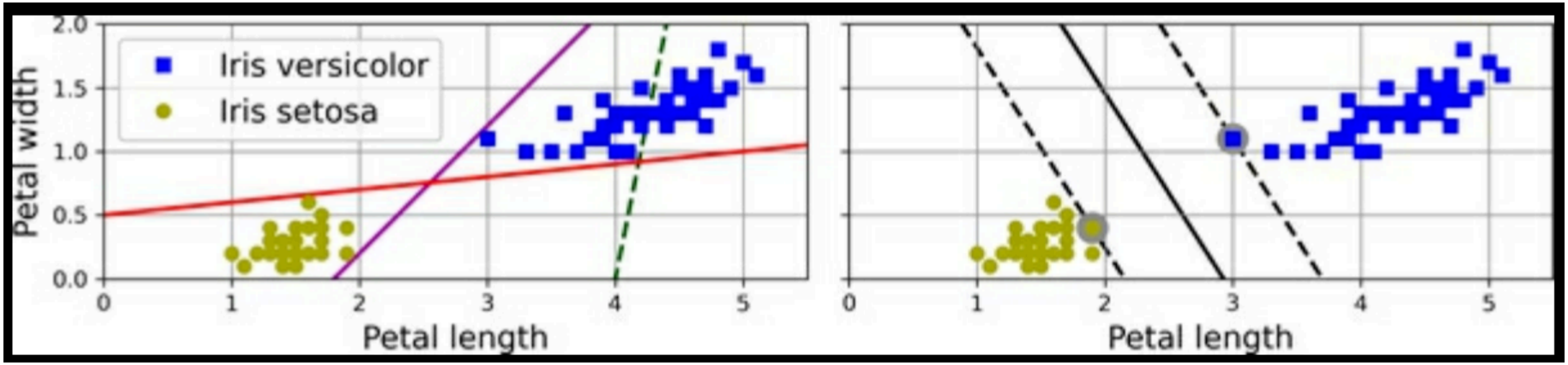
- **Linear SVM Classification**
- **Nonlinear SVM Classification**
- **SVM Regression**
- **Under the Hood of Linear SVM Classifiers**
- **The Dual Problem**

Support Vector Machines (SVMs)

- Powerful and versatile machine learning models
- Can perform
  - Linear and nonlinear classification
  - Regression
  - Novelty detection
- BUT they don't scale well to very large datasets

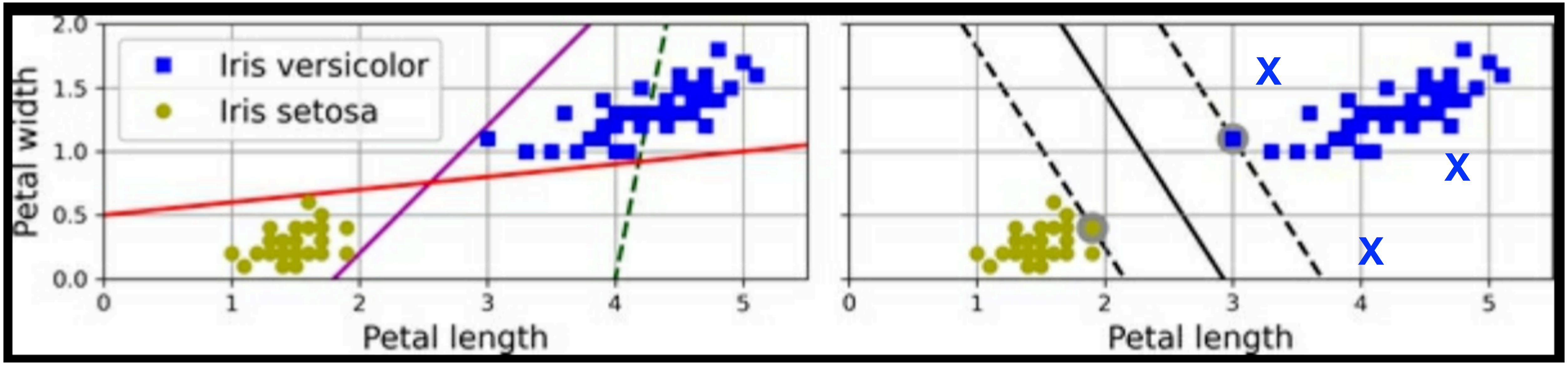
Linear SVM Classification

Large Margin Classification



- These classes are ***linearly separable***
- Either of the solid lines on the left side work
  - But won't generalize well because they are close to instances
- The line on the right side is the decision boundary of an SVM classifier
  - Widest possible margin

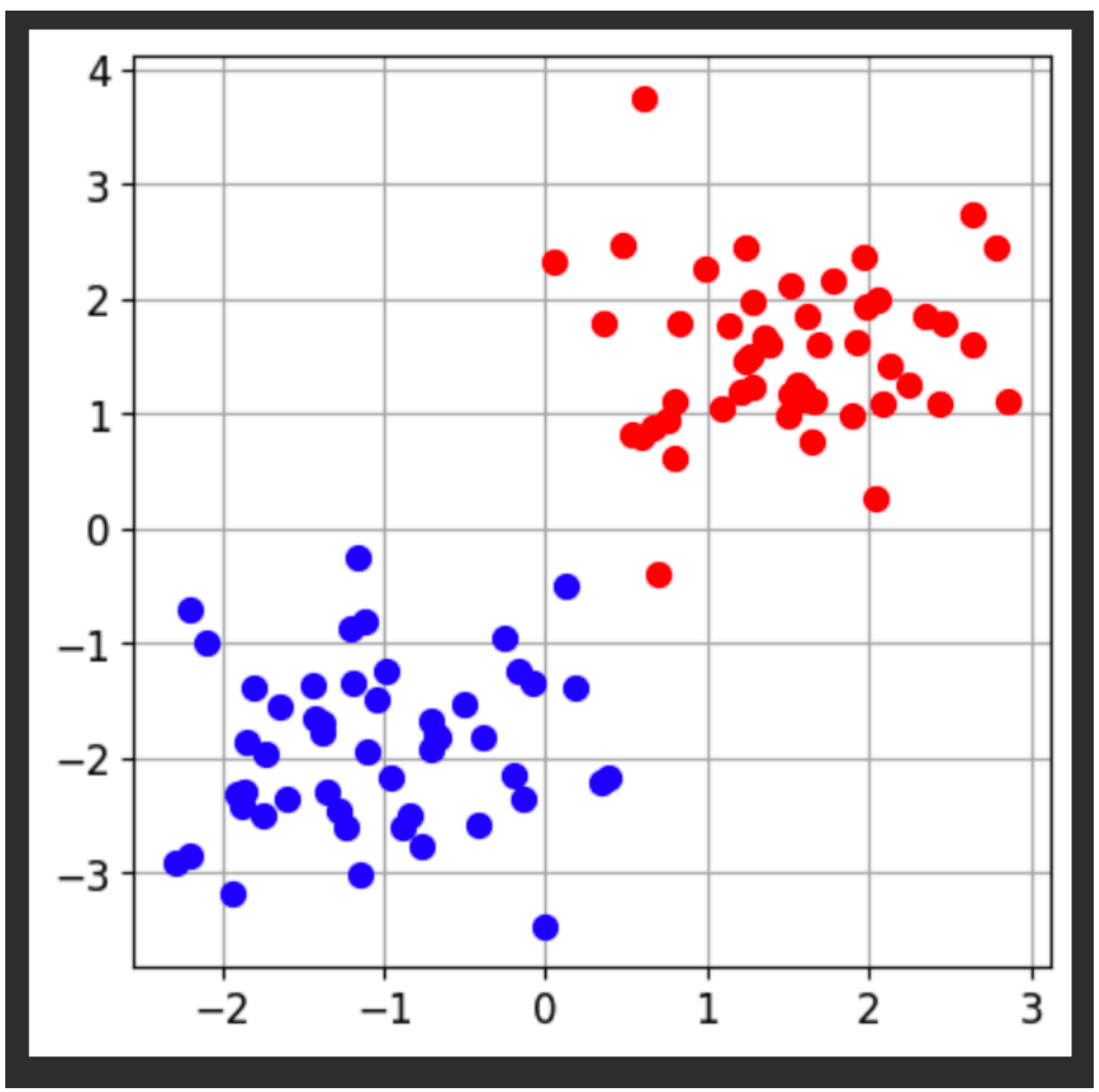
Large Margin Classification

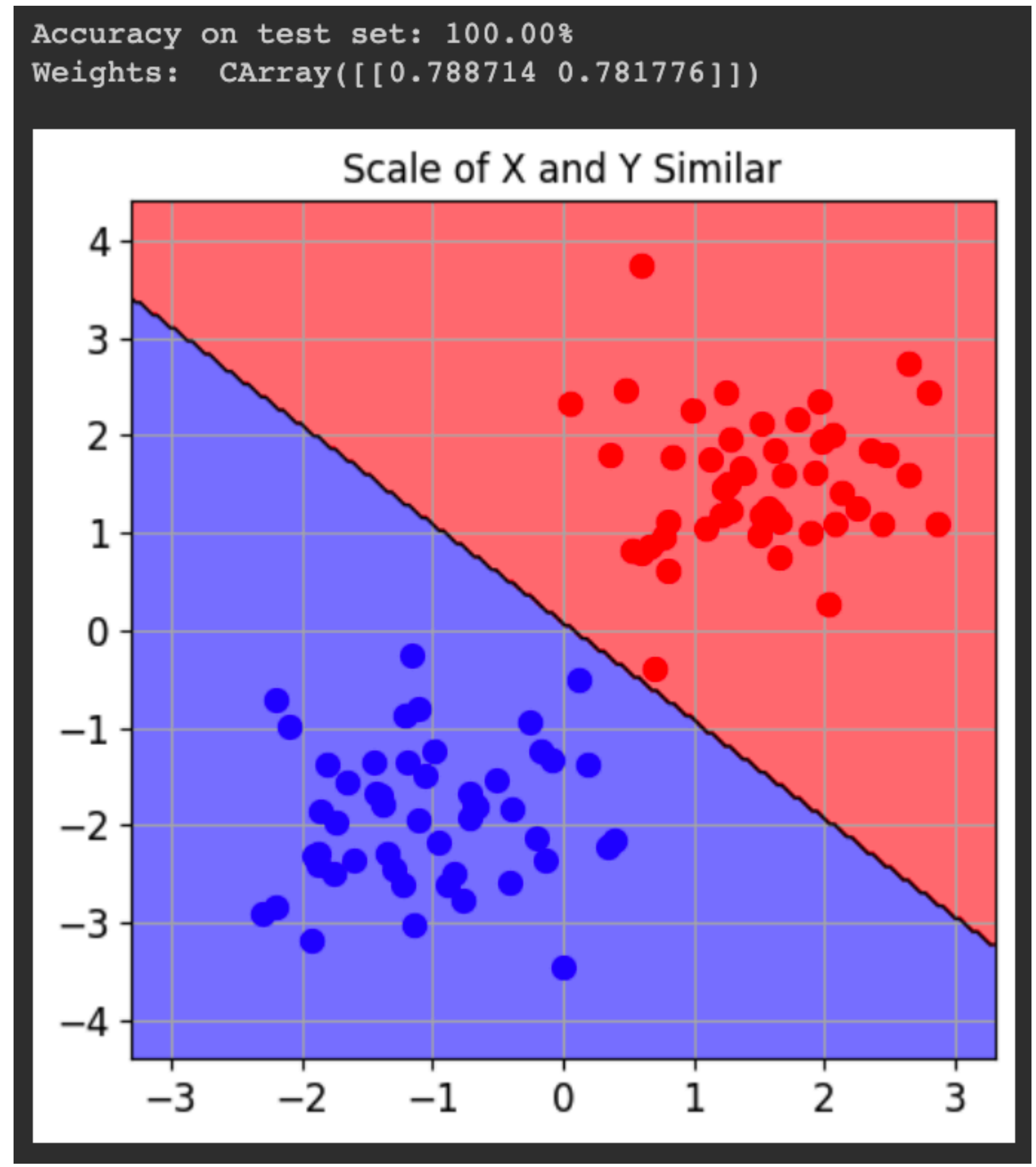


- Adding more instances won't affect the decision boundary
  - Unless they are inside the existing "street"
- The boundary is fully determined by the instances on the edge of the street
  - Those instances are the ***support vectors***

ML 112: Support Vector Machines

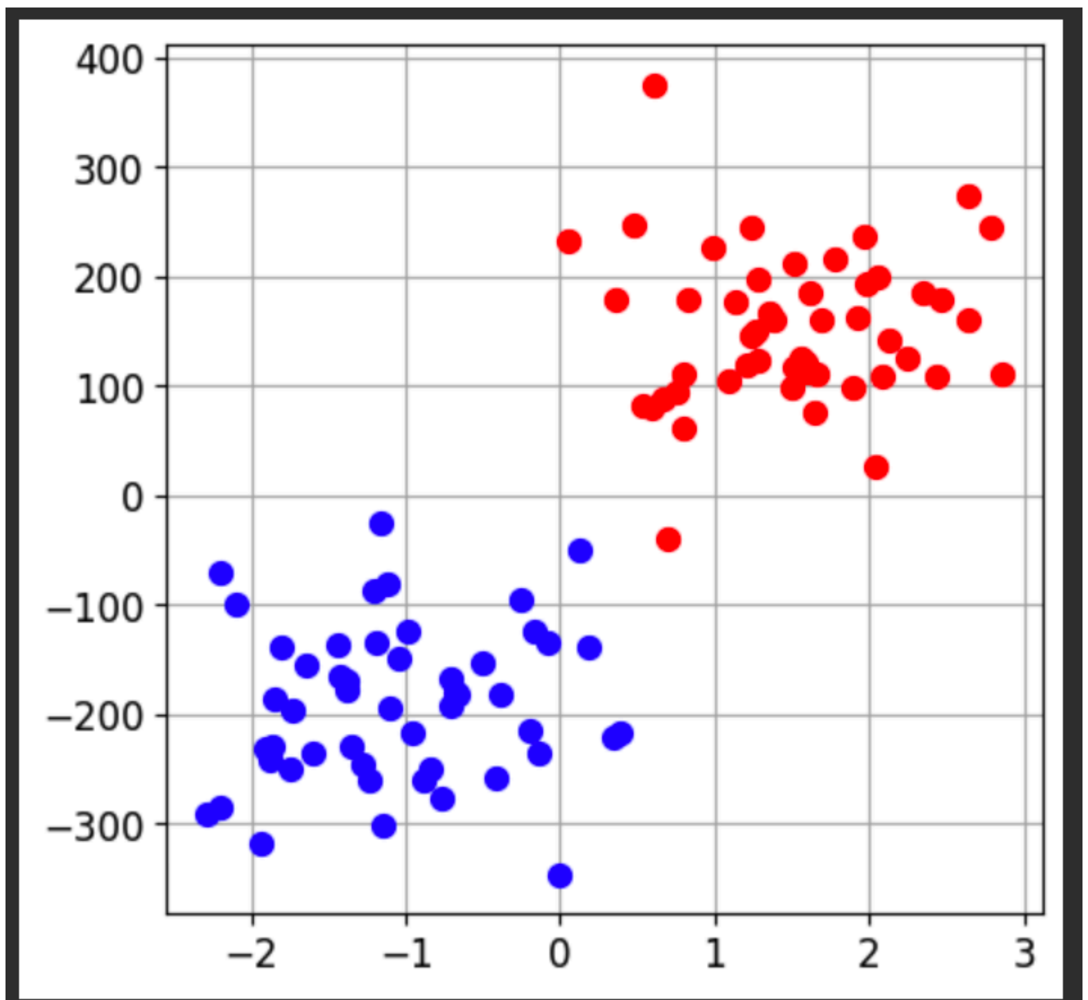
- Task: Classify these dots
- Linear SVM works well

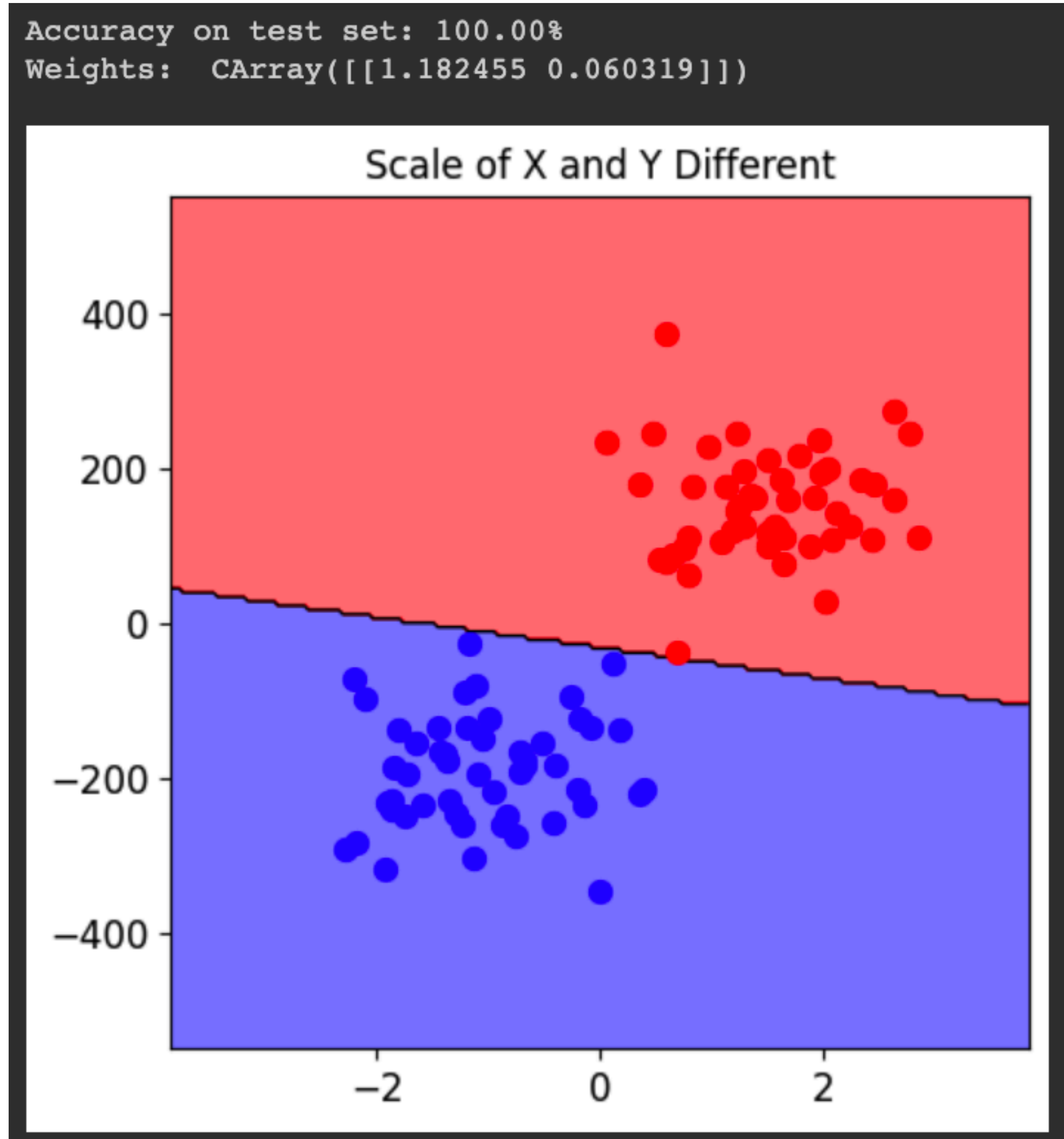




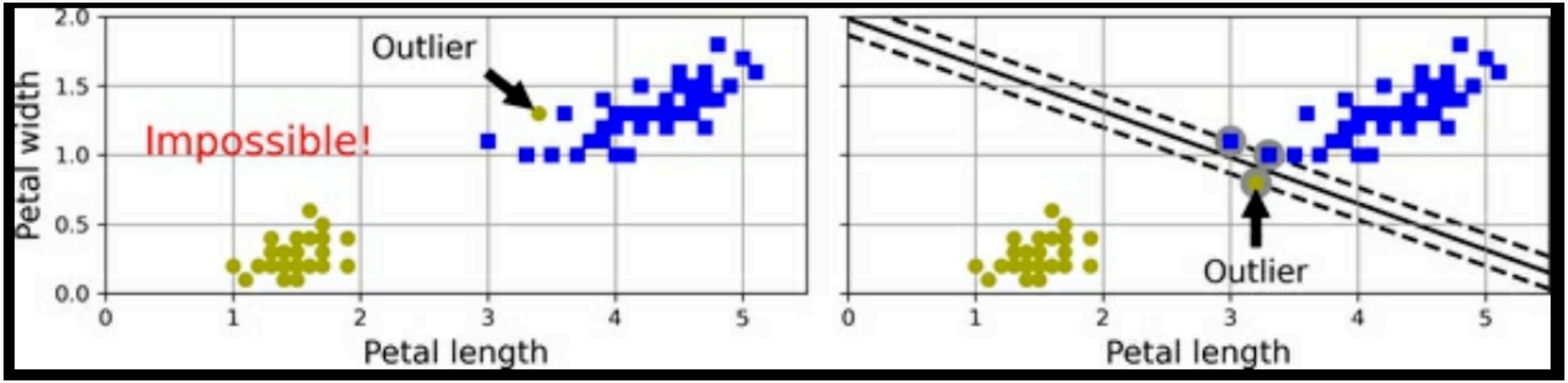
Scaling Changes the Fit

- Multiplying Y by 100
- Makes vertical distance count more



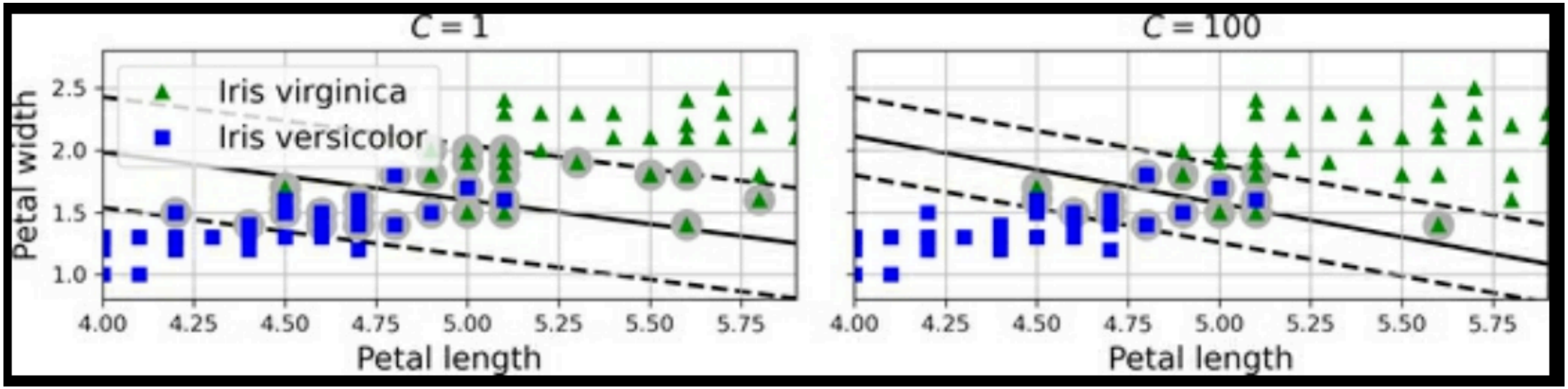


Hard Margin Classification



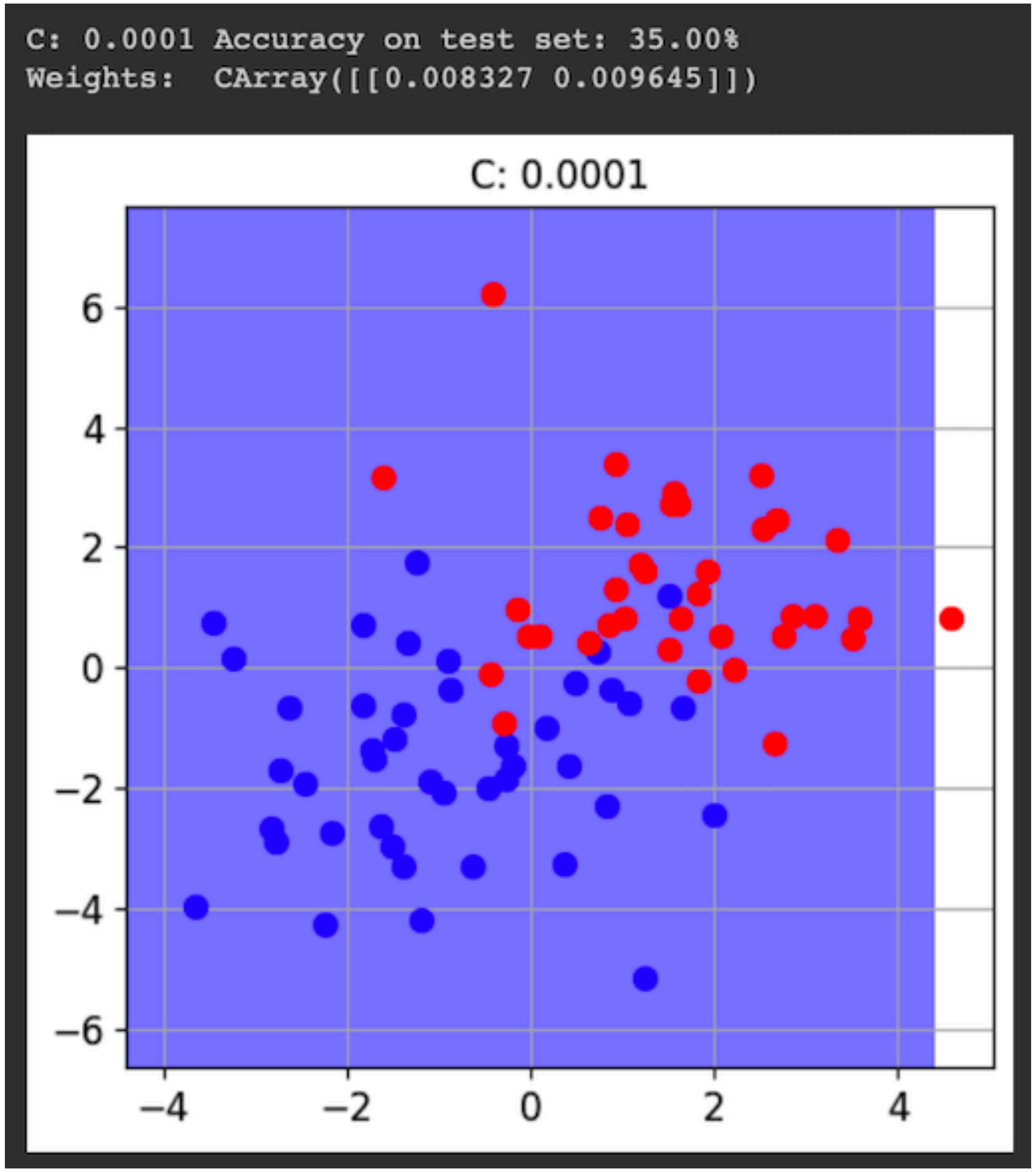
- Sensitive to outliers
- On the left, one outlier makes the solution impossible
- On the right, one outlier changes the decision line a lot

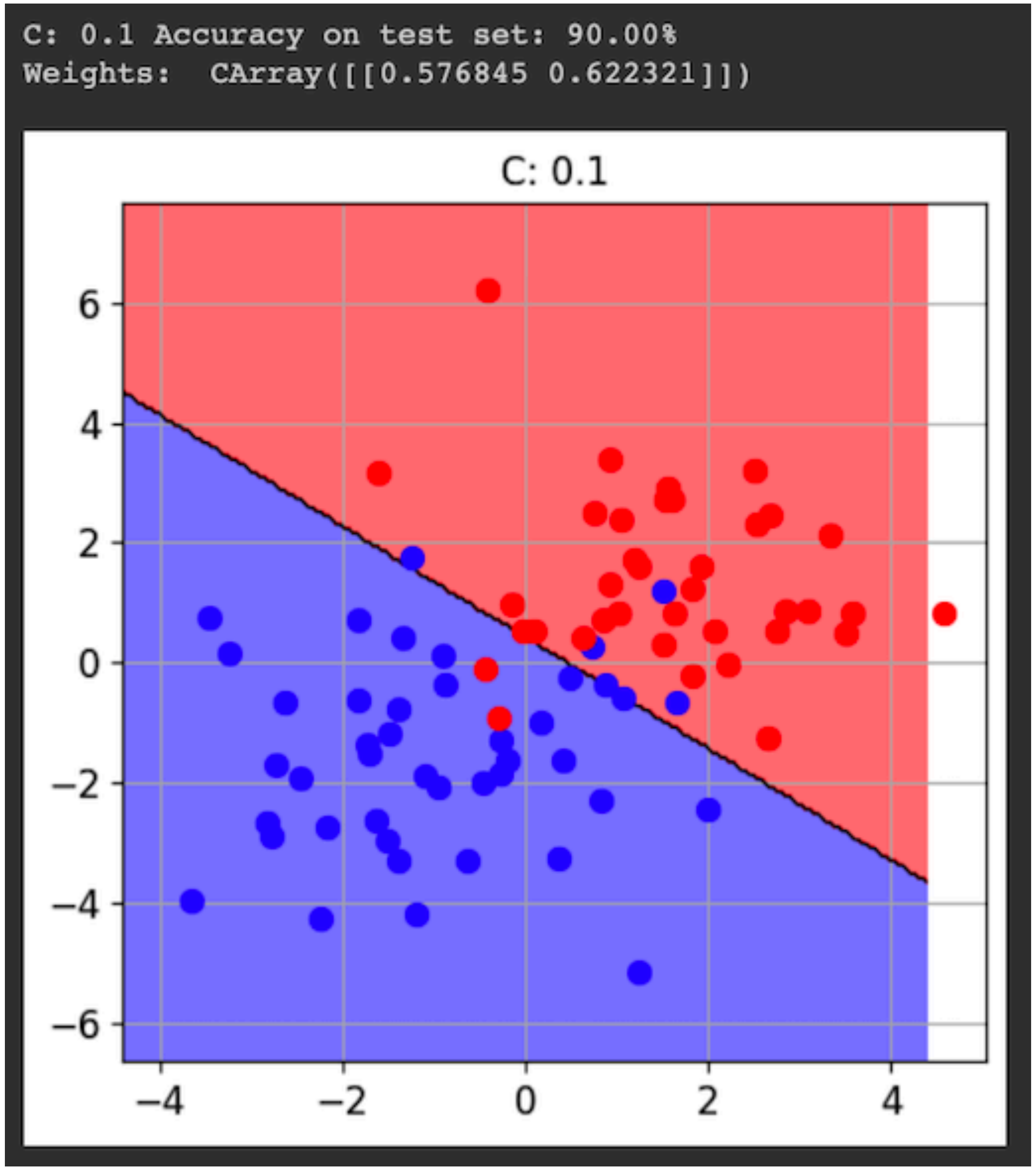
Soft Margin Classification



- Keep the street as wide as possible, while limiting the number of ***margin violations***
- Regularization hyperparameter **C**
  - Low **C** makes the street larger, with more margin violations
    - More instances supporting the street
    - Less chance of overfitting
    - But model may underfit

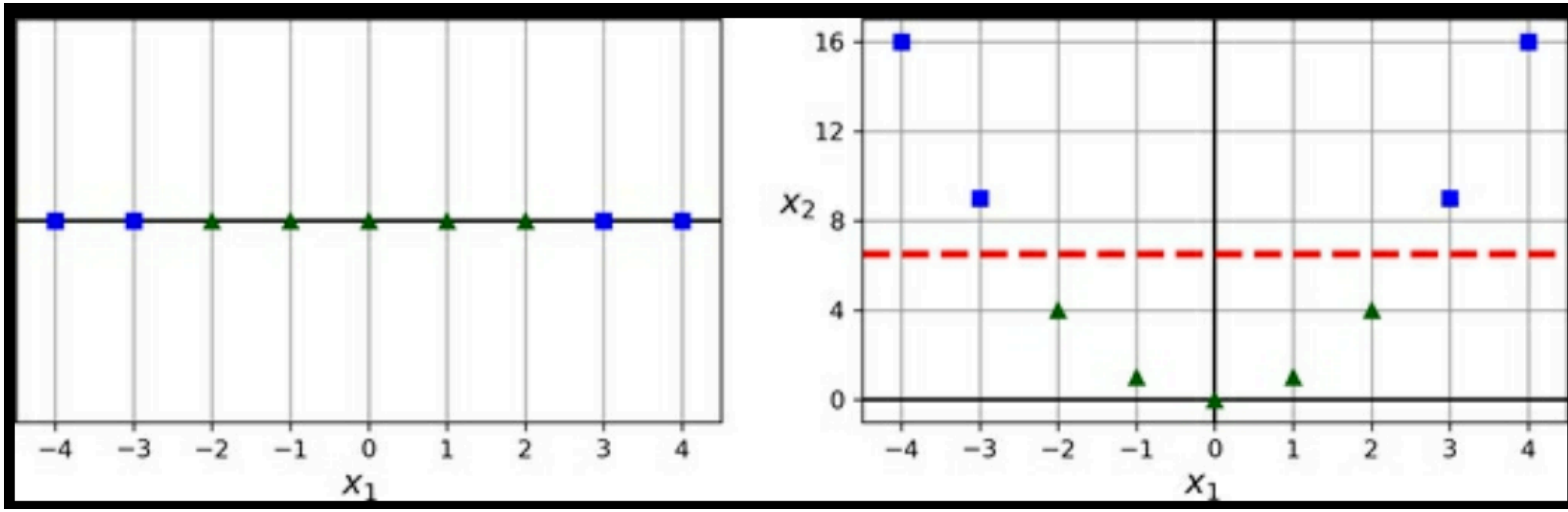
ML 112: Support Vector Machines





Nonlinear SVM Classification

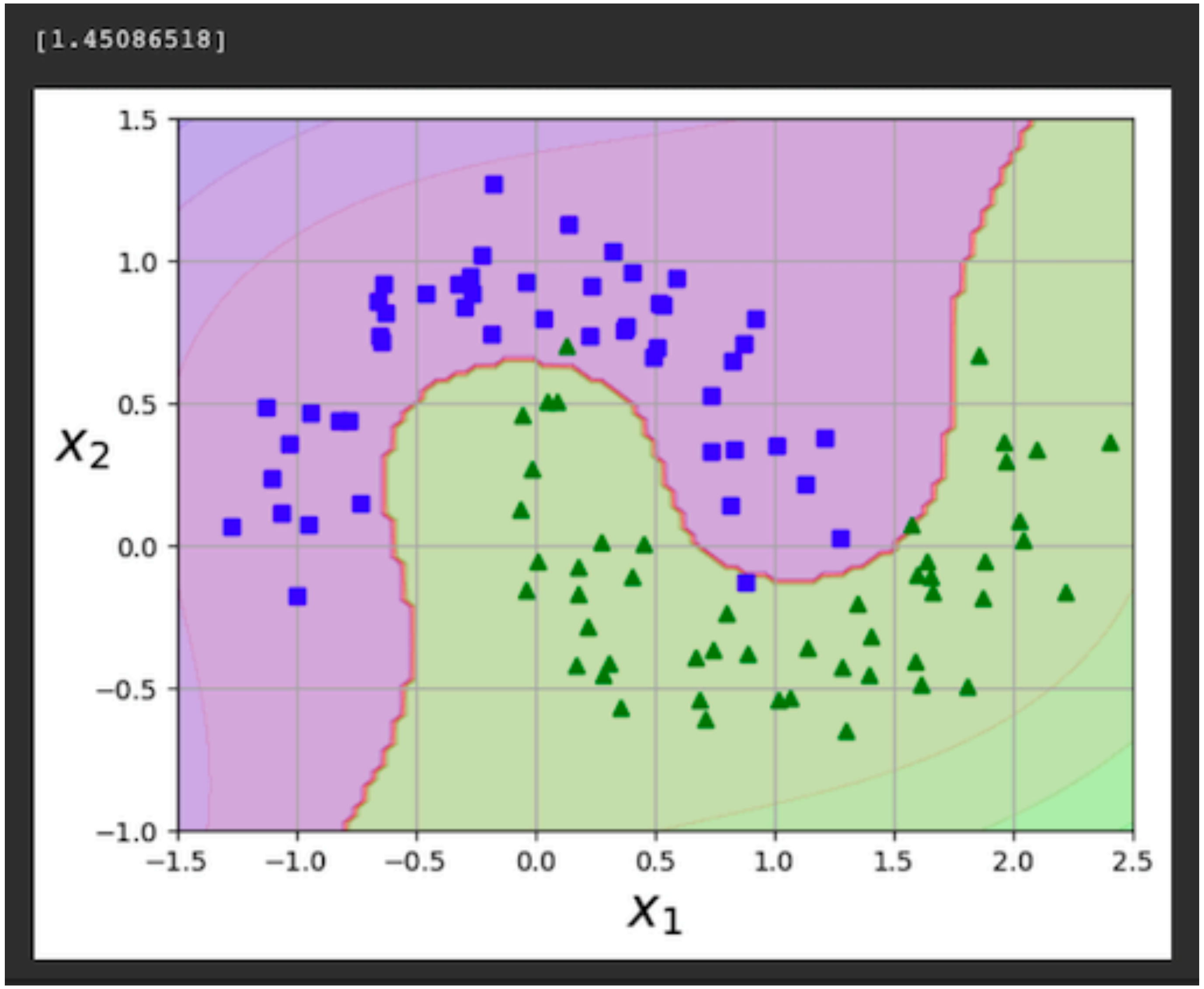
Adding Features



- With only x_1, a line can't separate the green and blue dots
- Adding x_1^2 makes an SVM possible

Adding Features

- Adding polynomial features up to degree 3 works for the "moons" data



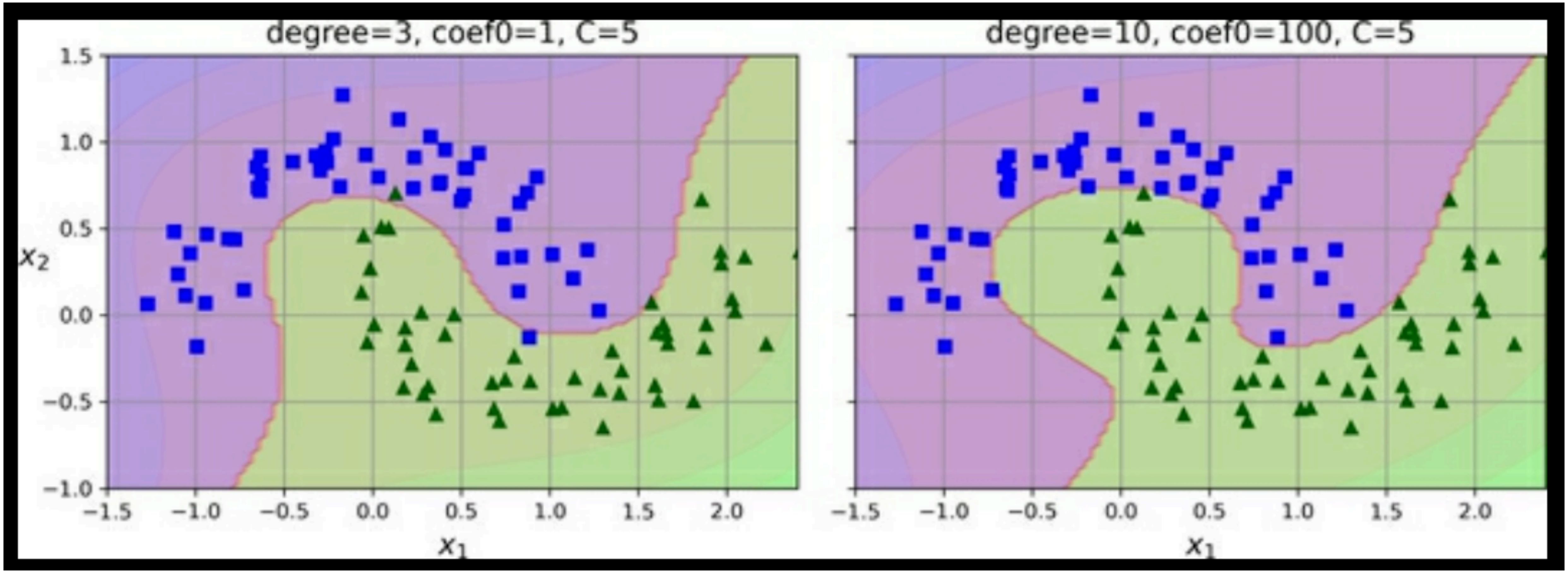
The Kernel Trick

- Gets same result as adding many polynomial features
  - Without actually increasing the number of features
- Implemented by the SVC class

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

poly_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
    ])
poly_kernel_svm_clf.fit(X, y)
```

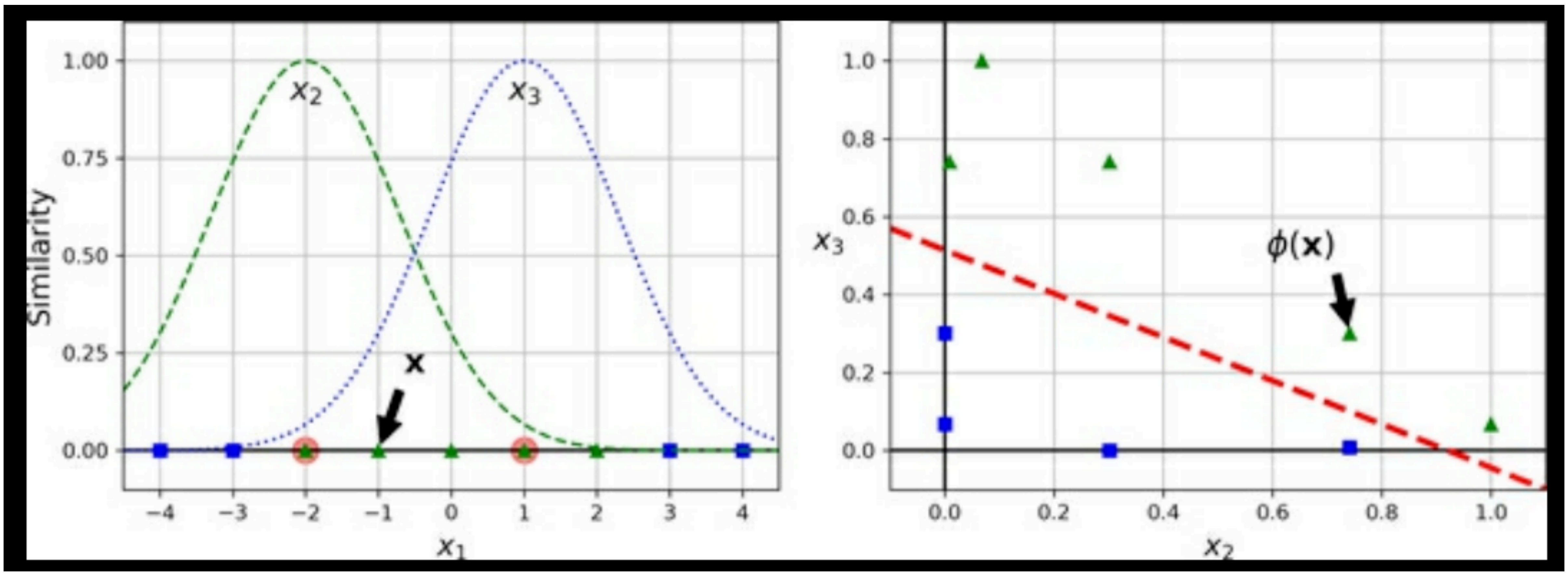
Polynomial Kernel



- Hyperparameter **coef0** controls how much the model is influenced by high-degree terms versus low-degree terms
- As before, low **C** makes the street wider, with more margin violations and a chance of underfitting

Similarity Features

- Similarity function measures how much each instance resembles a ***landmark***
- Here we use the ***Gaussian RBF*** (Radial Basis Function)

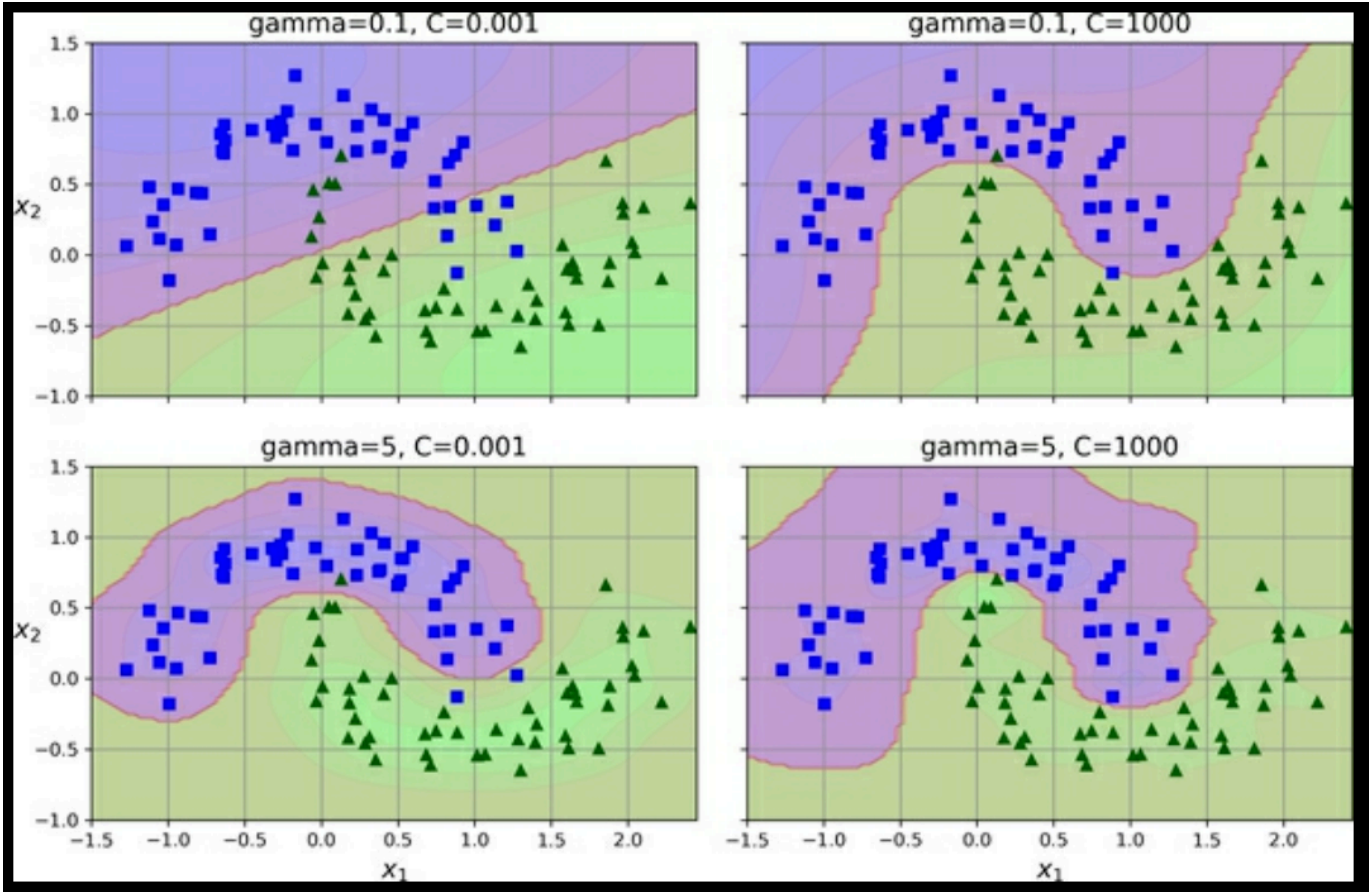


Gaussian RBF Kernel

```
rbf_kernel_svm_clf = make_pipeline(StandardScaler(),
                                   SVC(kernel="rbf", gamma=5, C=0.001))
rbf_kernel_svm_clf.fit(X, y)
```

- Hyperparameter **gamma** controls the width of the bell function
    - Lower gamma means narrower
    - If your model is overfitting, reduce **gamma**
- As before, reducing **C** also reduces overfitting

Gaussian RBF Kernel



String Kernel

- Can be used when classifying text documents or DNA sequences
- Using **Levenshtein distance**
  - The minimum number of single-character edits required to change one word into the other

Choosing a Kernel

- Always try linear kernel first
- Then try Gaussian RBF kernel
- Then others, such as polynomial kernel

Computational Complexity

- For ***m*** training instances and ***n*** features
- LinearSVC is fast, unless you require high precision (low ε or ***tol***)

Class	Time complexity	Out-of-core support	Scaling required	Kernel trick
`LinearSVC`	$O(m \times n)$	No	Yes	No
`SVC`	$O(m^2 \times n)$ to $O(m^3 \times n)$	No	Yes	Yes
`SGDClassifier`	$O(m \times n)$	Yes	Yes	No

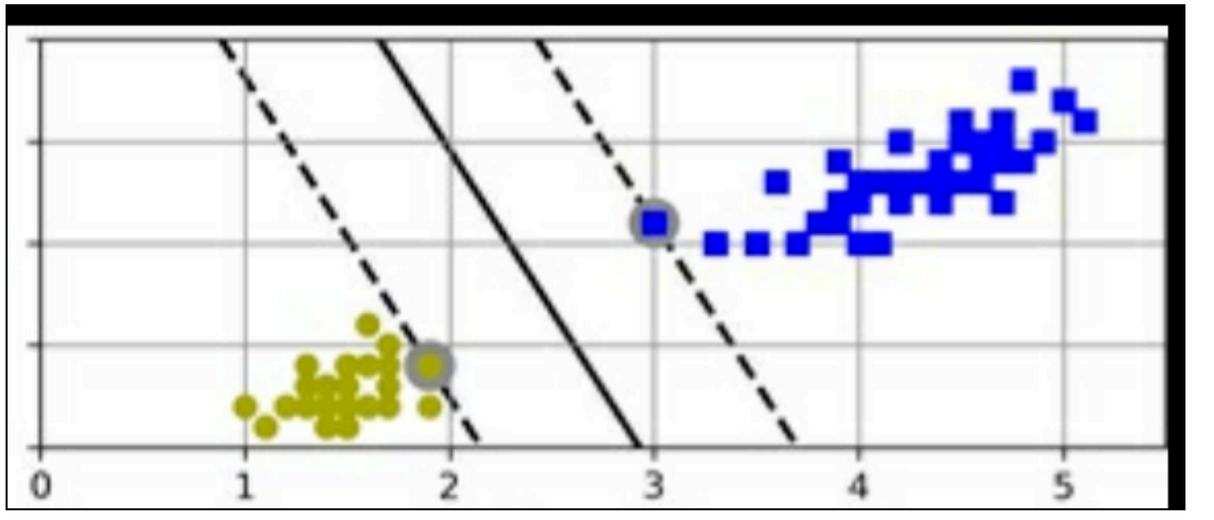
Kahoot!

Ch 5a

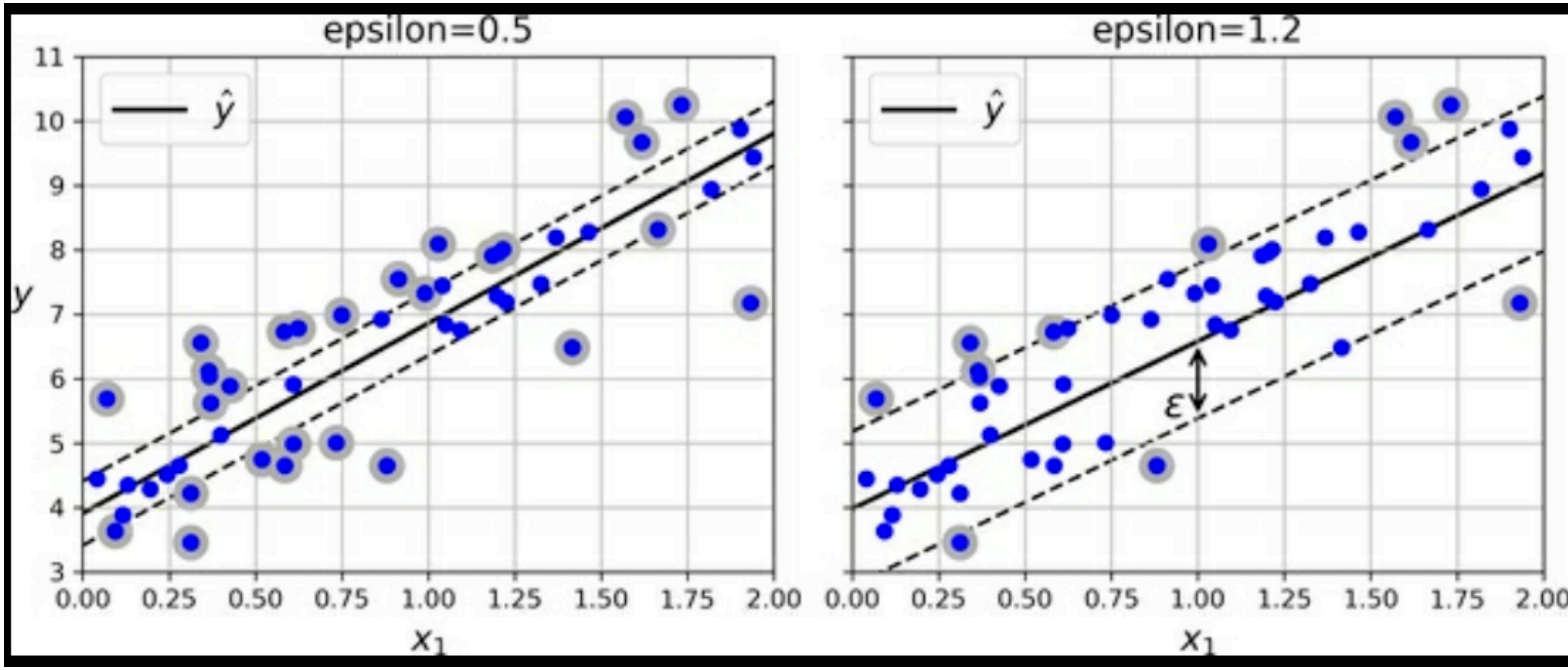
SVM Regression

SVM Regression

- **SVM classification**
  - Fit widest possible street between the classes
  - Minimizing instances on the street (margin violations)
- **SVM regression**
  - Fit as many instances as possible on the street
  - Minimizing instances off the street (margin violations)

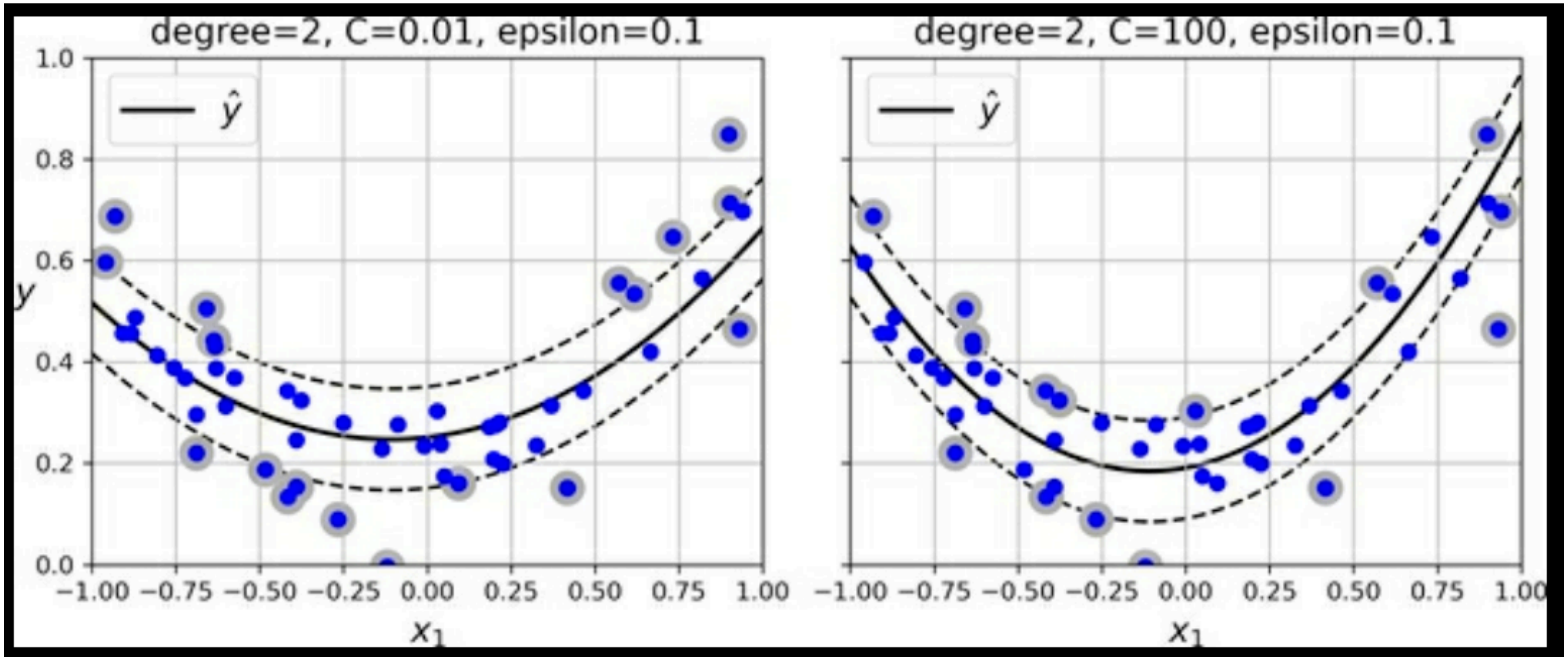


SVM Regression



- ***epsilon*** controls width of street

Kernelized SVM Regression



Under the Hood of Linear SVM Classifiers

Weights and Bias

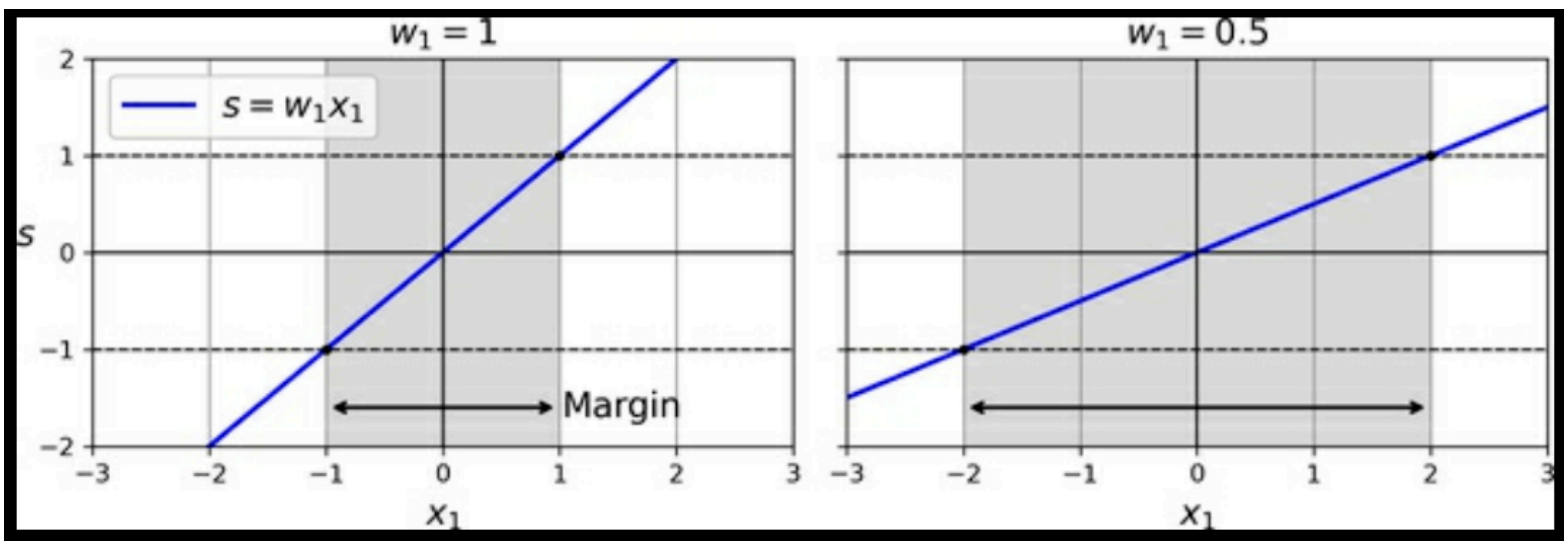
- Decision function is

$$\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x} + \boldsymbol{b} = \boldsymbol{w}_1 \boldsymbol{x}_1 + \boldsymbol{w}_2 \boldsymbol{x}_2 + \ldots + \boldsymbol{w}_n \boldsymbol{x}_n + \boldsymbol{b}$$

- $\boldsymbol{x}$ contains the instance values
- $\boldsymbol{w}$ contains the weights
- $\boldsymbol{b}$ is the bias

Margin Size

- Define borders of the street at decision function -1 and 1
- Smaller ***w*** means wider margin
- For an SVM classifier, we want the smallest possible ***w***



Hard Margin Linear SVM Classifier

- We want to minimize ***w***
- To keep instances off the street,
  - Decision function must be > 1 for all positive instances
  - And < -1 for all negative instances
- *t* is the instance's correct class

$$\underset{\mathbf{w},b}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^\intercal \mathbf{w}$$

$$\text{subject to} \quad t^{(i)}\left(\mathbf{w}^\intercal \mathbf{x}^{(i)} + b\right) \geq 1 \quad \text{for } i = 1, 2, \cdots, m$$

Soft Margin Linear SVM Classifier

- Add ***slack variable*** zeta $\boldsymbol{\zeta}$
  - Measures how much an instance is allowed to violate the margin

$$\underset{\mathbf{w},b,\zeta}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^\intercal\mathbf{w} + C\sum_{i=1}^{m}\zeta^{(i)}$$

$$\text{subject to} \quad t^{(i)}\left(\mathbf{w}^\intercal\mathbf{x}^{(i)} + b\right) \geq 1 - \zeta^{(i)} \quad \text{and} \quad \zeta^{(i)} \geq 0 \quad \text{for } i = 1, 2, \cdots, m$$

The Dual Problem

The Dual Problem

- Gives the same result for the SVM problem
- Faster to solve
  - When the number of instances ***m*** is smaller than the number of features ***n***
  - Makes the kernel trick possible

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \ \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha^{(i)}\alpha^{(j)}t^{(i)}t^{(j)}\mathbf{x}^{(i)\top}\mathbf{x}^{(j)} \ - \ \sum_{i=1}^{m}\alpha^{(i)}$$

$$\text{subject to } \alpha^{(i)} \geq 0 \text{ for all } i = 1, 2, \ldots, m \text{ and } \sum_{i=1}^{m}\alpha^{(i)}t^{(i)} = 0$$

Common Kernels

$$\begin{aligned}
\text{Linear:} \quad & K(\mathbf{a}, \mathbf{b}) = \mathbf{a}^\top \mathbf{b} \\
\text{Polynomial:} \quad & K(\mathbf{a}, \mathbf{b}) = (\gamma \mathbf{a}^\top \mathbf{b} + r)^d \\
\text{Gaussian RBF:} \quad & K(\mathbf{a}, \mathbf{b}) = \exp\left(-\gamma \|\mathbf{a} - \mathbf{b}\|^2\right) \\
\text{Sigmoid:} \quad & K(\mathbf{a}, \mathbf{b}) = \tanh\left(\gamma \mathbf{a}^\top \mathbf{b} + r\right)
\end{aligned}$$

Kahoot!

Ch 4b