# Support Vector Machines

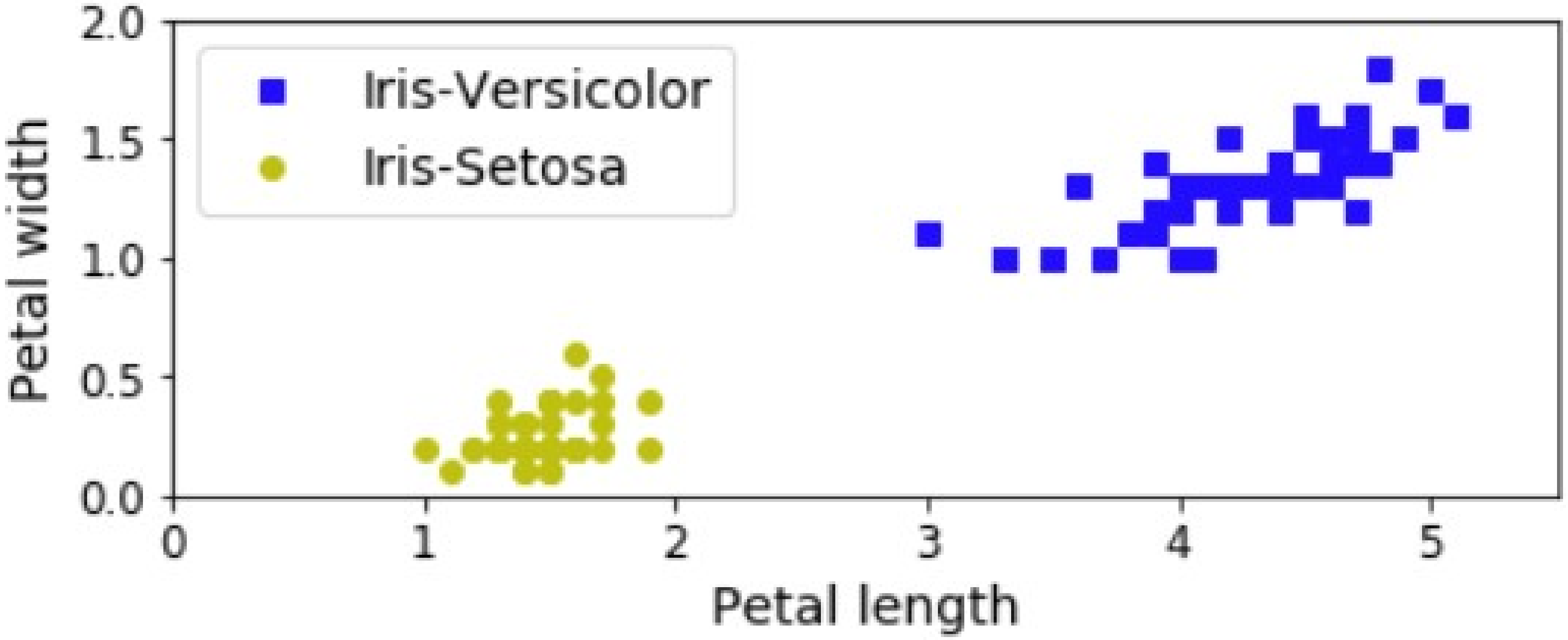
Tim Smith, PhD

# Support Vector Machines

- Very popular
- Powerful for both classification and regression
- Can uncover both linear and nonlinear relationships!
- Resource intensive!

# Linear SVM Classification

- Iris data: separate one class from the other

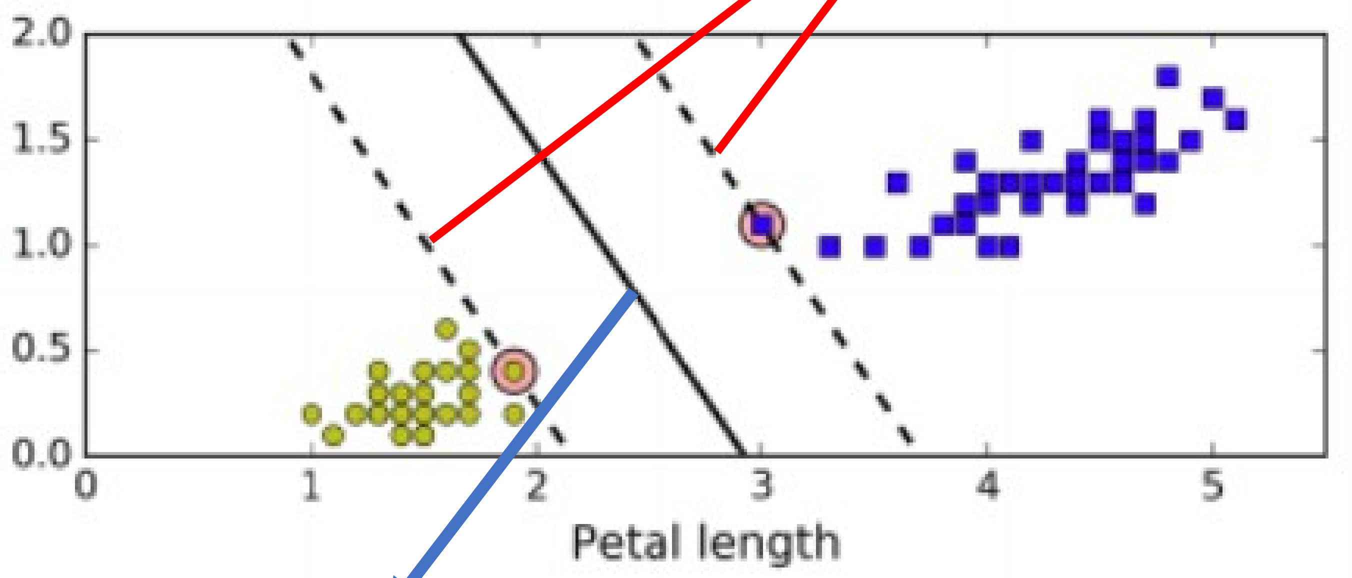


# Linear SVM Classification

- Iris data: separate one class from the other

Support vectors

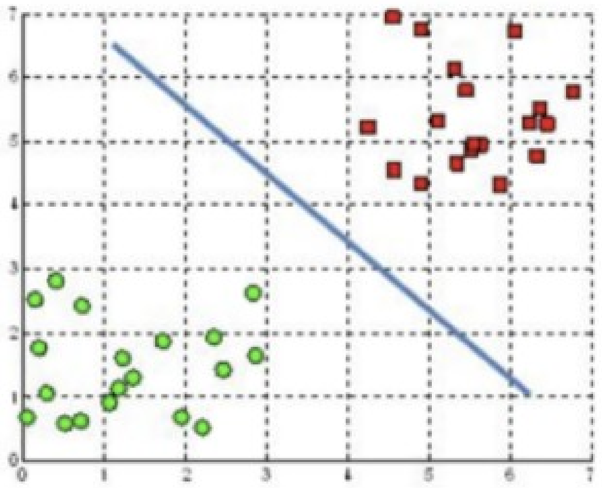
Goal: fit the widest possible "street" between classes



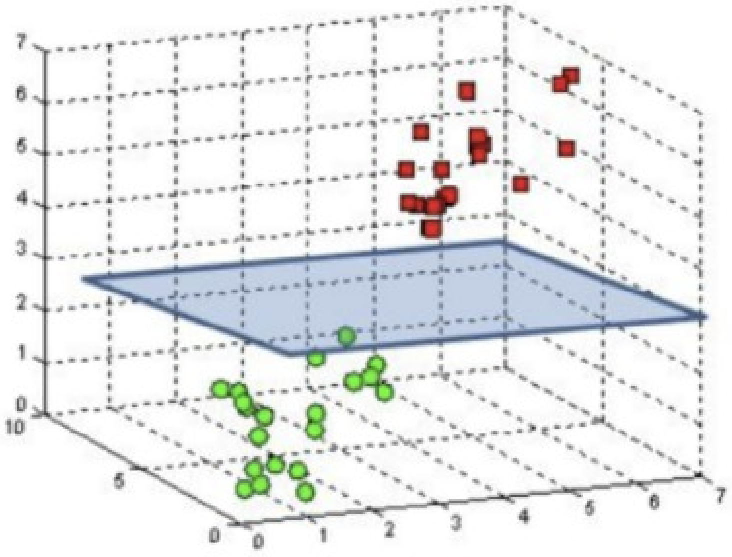

Decision boundary (stays as far away as possible from instances)

# Linear SVM Classification

- Hyperplanes: separate one class from another



In a 2-D world, the hyperplane is a "line"



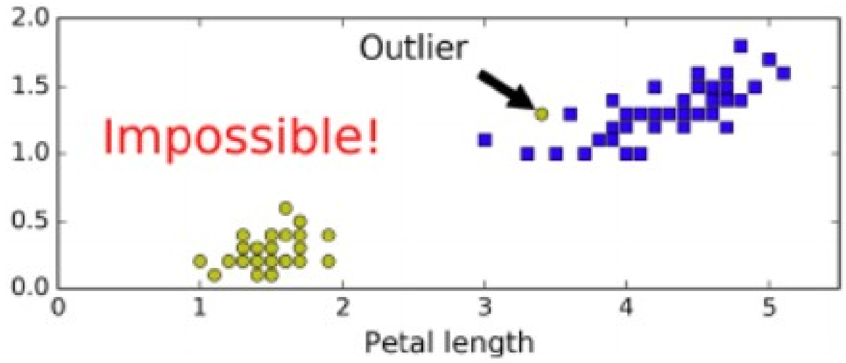
In a 3-D world, the hyperplane is a "plane"

Image source: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

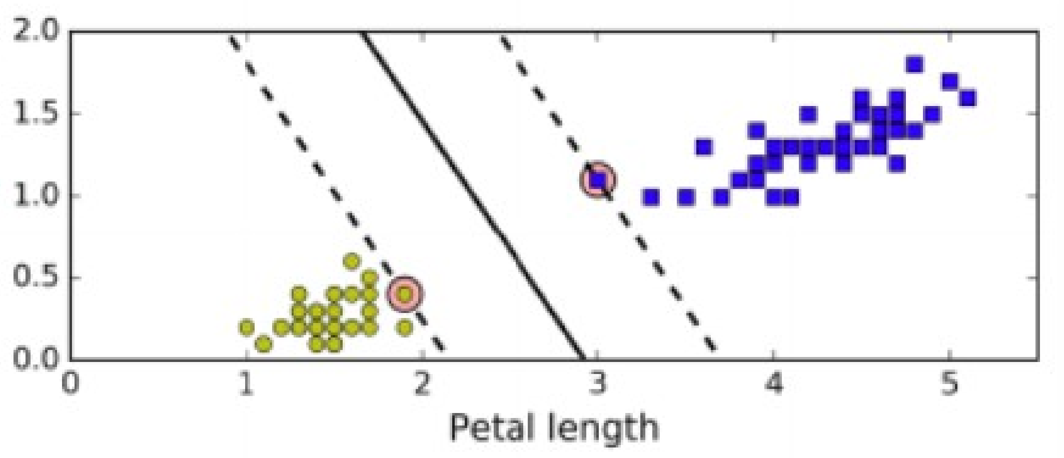
# Hard Margin Classification

- Data must be linearly separable
- There must be NO outliers
- (i.e., the perfect world!)

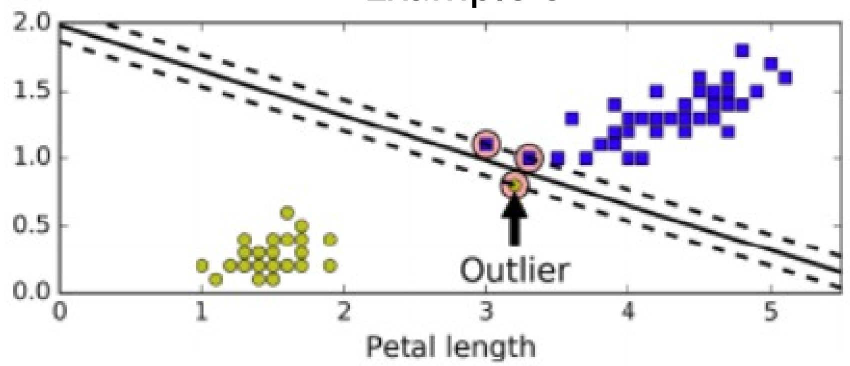
Example 2




Example 1

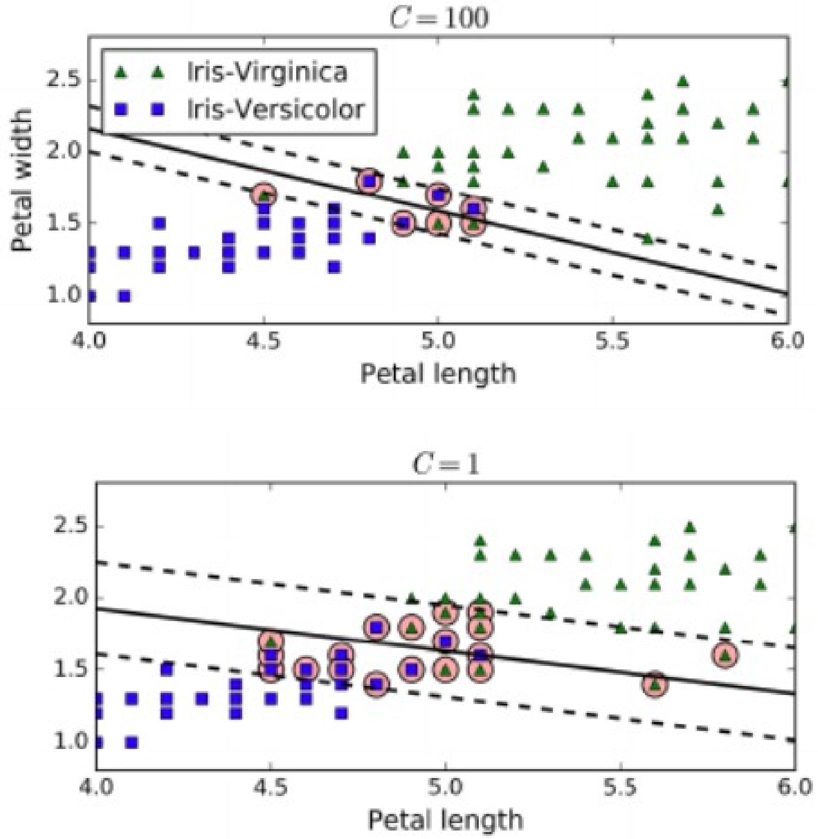



Example 3



# Soft Margin Classification

- Allows margin violations
  - Instances that are "in the street" or "misclassified"
- C is referred to as “regularization parameter”
  - <u>Controls the width of the margin</u>
  - **Determines how much violation is allowed!**
- Higher C values perform LESS regularization
  - Leads to a SMALLER margin
  - **Aims for less violations**
  - **There is a risk of overfitting**
- Lower C values perform MORE regularization
  - Leads to a WIDER margin
  - **Aims for more violations**
  - **Favors generalizability**



# Nonlinear SVM Classification

- Not all data are linearly separable
- Solution 1:
  - Create polynomial features manually and fit a linear SVM!
  - Problem: Generates too much data (so needs more resources!)

- (Example on next slide)

# Nonlinear SVM Classification

- Example

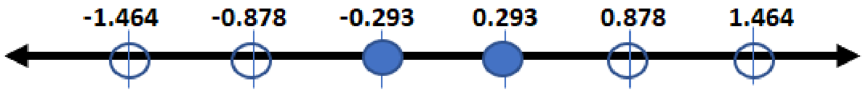
| Sqft | Sold |
|---|---|
| 1000 | 0 |
| 1250 | 0 |
| 1500 | 1 |
| 1750 | 1 |
| 2000 | 0 |
| 2250 | 0 |

**Standardize**

| Sqft | Sold |
|---|---|
| -1.464 | 0 |
| -0.878 | 0 |
| -0.293 | 1 |
| 0.293 | 1 |
| 0.878 | 0 |
| 1.464 | 0 |

# Nonlinear SVM Classification

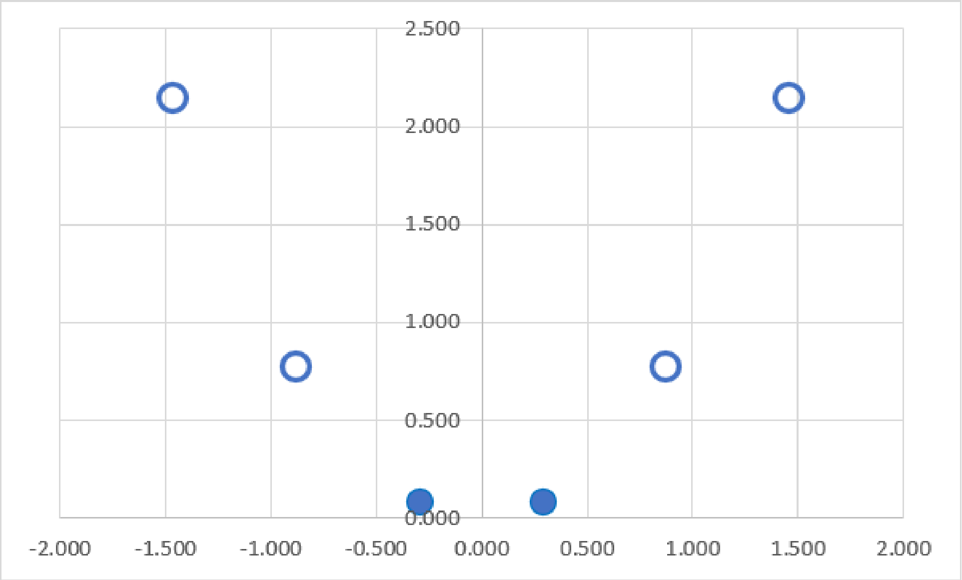
- Example




**Standardize**

| Sqft | Sold |
|---|---|
| -1.464 | 0 |
| -0.878 | 0 |
| -0.293 | 1 |
| 0.293 | 1 |
| 0.878 | 0 |
| 1.464 | 0 |

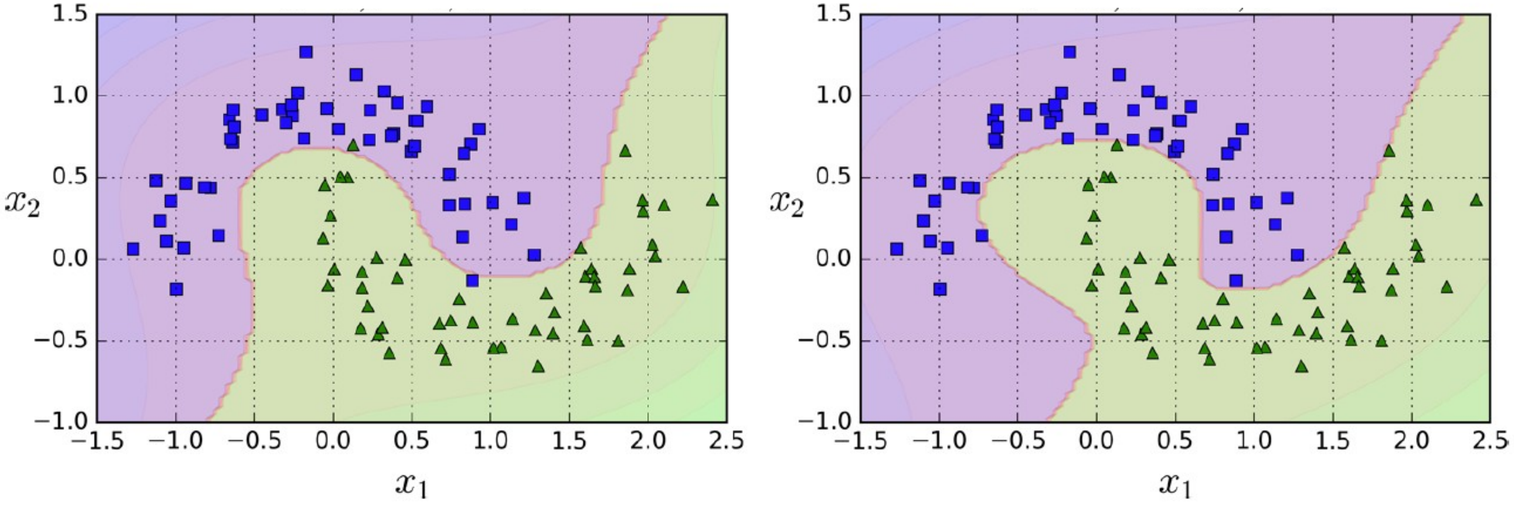
| Sqft | $Sqft^2$ | Sold |
|---|---|---|
| -1.464 | 2.143 | 0 |
| -0.878 | 0.771 | 0 |
| -0.293 | 0.086 | 1 |
| 0.293 | 0.086 | 1 |
| 0.878 | 0.771 | 0 |
| 1.464 | 2.143 | 0 |



# Nonlinear SVM Classification

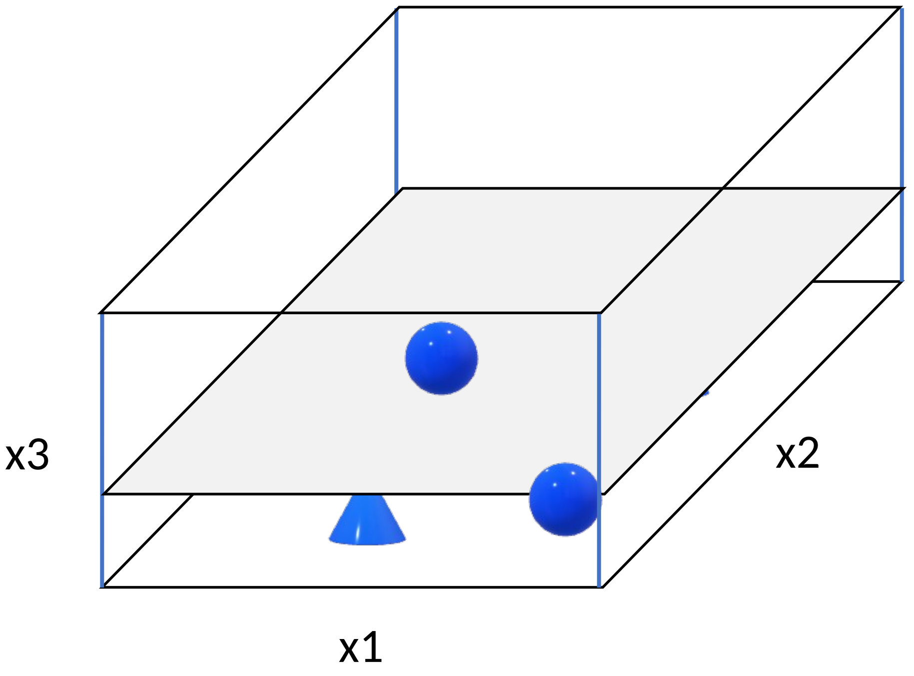
- Solution 2: use a "kernel trick"
  - Similar to adding many polynomial degrees; but it’s only calculated and the data is not actually altered by adding new polynomial features.
  - SVC(kernel="poly", degree=3, coef0=1, C=5)
    - coef0 controls how much the model should be influenced by higher degree polynomials

d=3, coef0=1, C=5 d=10, coef0=100, C=5



# Adding Similarity Features

- Another technique to tackle nonlinear classification
- Adds "features" (i.e., new variables/dimensions) for separation of instances
- Computationally expensive



# Gaussian RBF Kernel

- See https://www.youtube.com/watch?v=NYwVM6_EuxQ

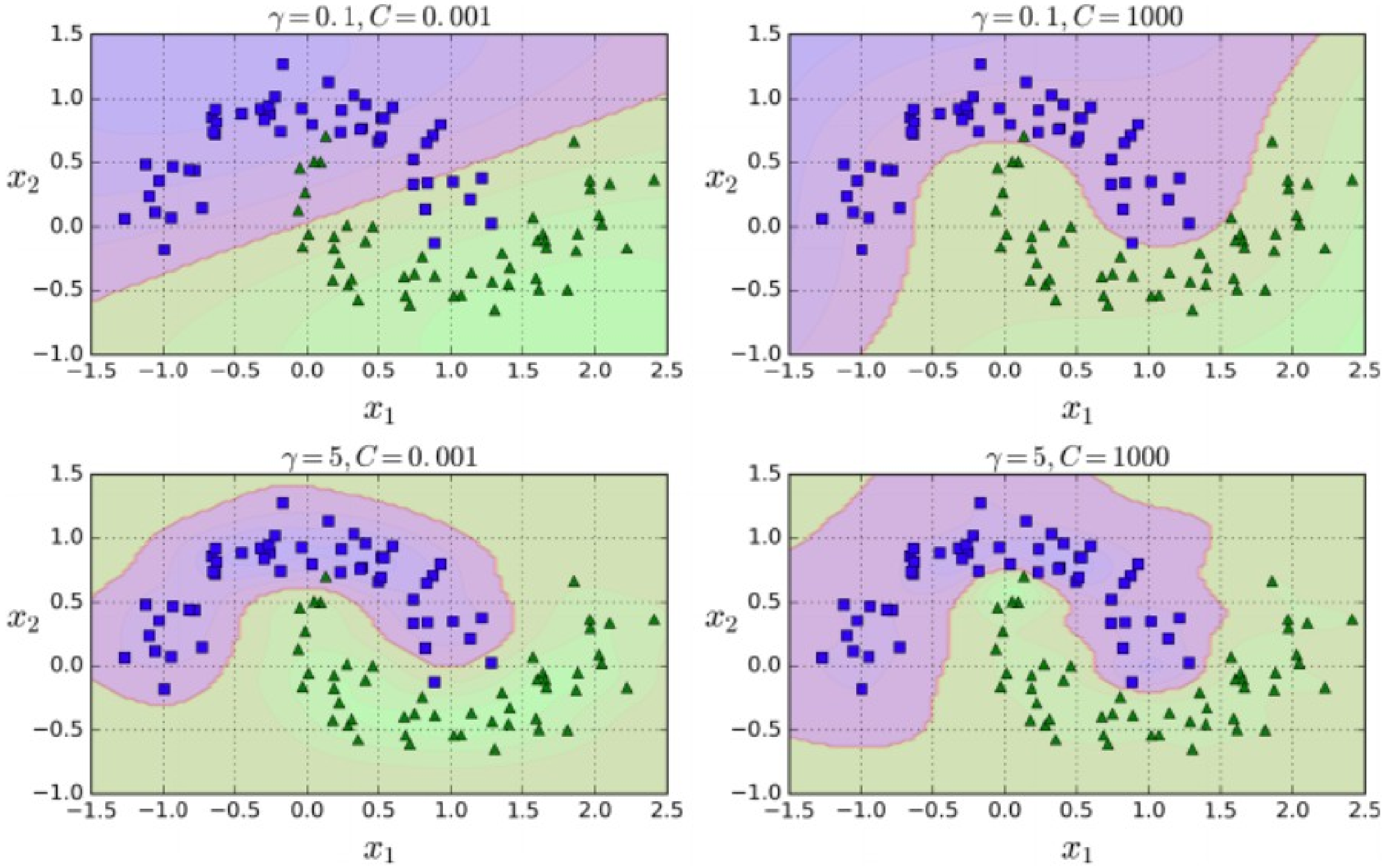
# Gaussian RBF Kernel

- SVC(kernel="rbf", gamma=5, C=0.001)
  - Higher gamma makes the bell shape narrower
  - Smaller gamma makes the bell shape wider
- Increase gamma if the model is underfitting!
- Decrease gamma if overfitting!

# Gaussian RBF Kernel

Increase C to minimize violations

Increase gamma to address underfitting



# Kernel Tricks

- SVC(kernel="linear")
  - Fits a straight line/plane to separate the two classes
- SVC(kernel="poly")
  - Tricks the SVC to think that there are polynomial features (**WITHOUT** creating polynomial features)
  - Fits a curved line/plane to separate the two classes
- SVC(kernel="rbf")
  - Tricks the SVC to think there are new features (i.e., similarity features)
  - Fits a straight line/plane in a new n-dimensional space

# Multi-Class Classification

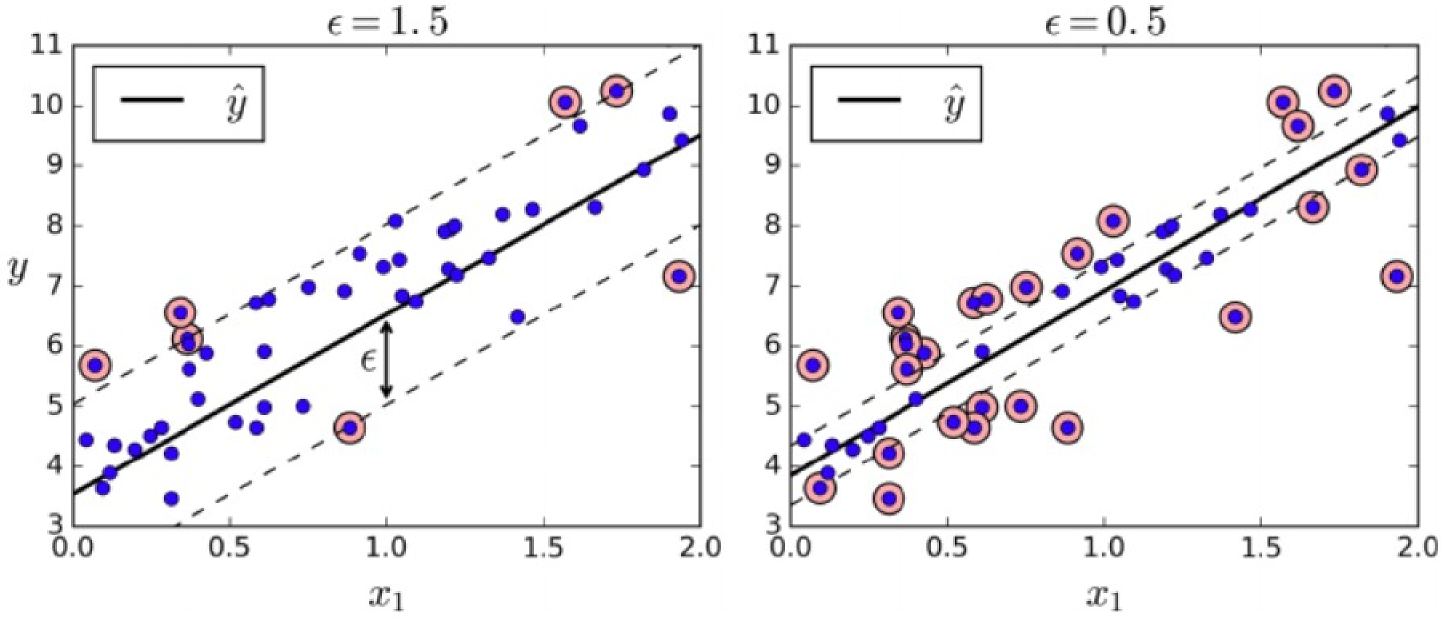
- SVM cannot perform multi-class classification (in its true sense)
- Instead, it performs: one-versus-rest ("ovr")
  - Create multiple binary class classification models
  - Run the observation on these models
  - Make a final determination based on the combined results

# OvR

- Logistic and SVM use this encoding in the background.
- One-vs-rest is a method for using binary classification algorithms for multi-class classification.
- OvR approach splits the multi-class dataset into multiple binary classification problems.
    - Example: Predicting red, blue, green or yellow:
        - Binary classification problem 1: red vs [blue, green, yellow]
        - Binary classification problem 2: blue vs [red, green, yellow]
        - Binary classification problem 3: green vs [red, blue, yellow]
        - Binary classification problem 4: yellow vs [red, blue, green]
- A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.

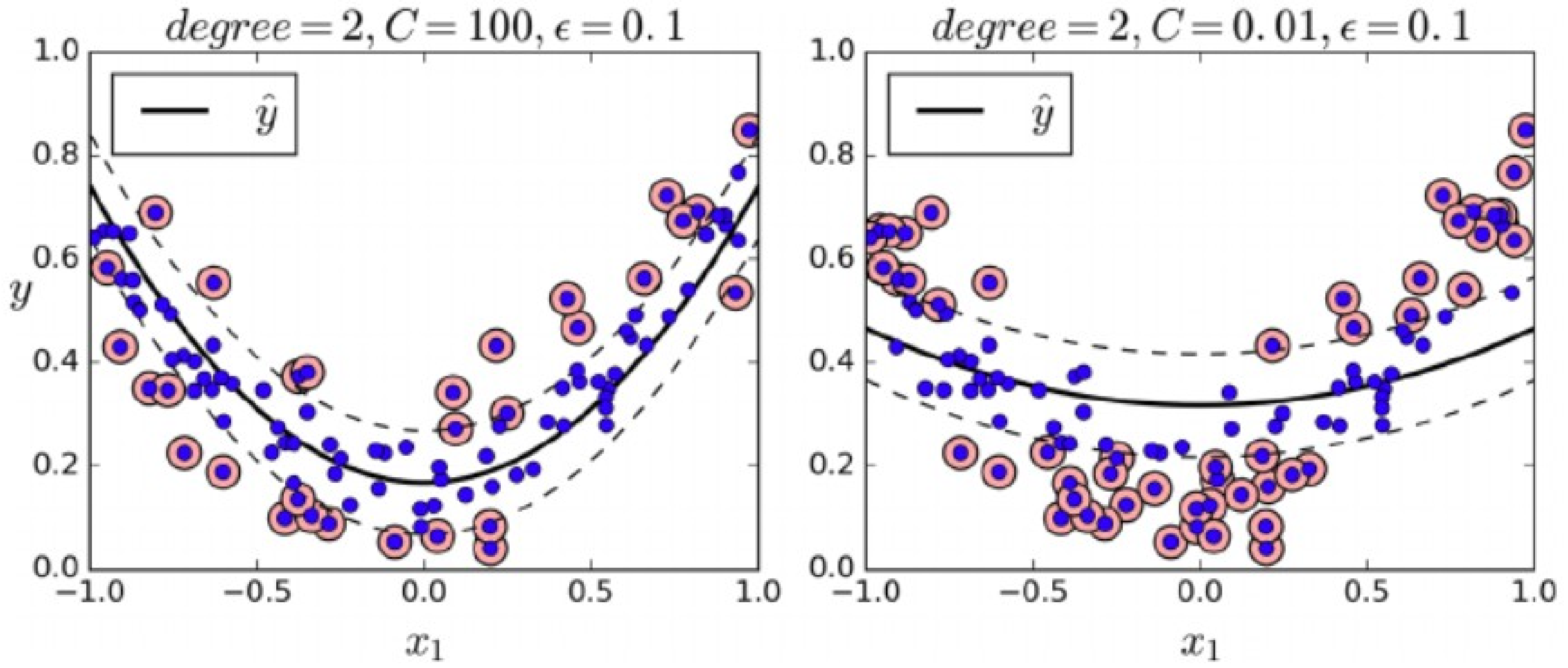
# SVM Regression

- SVM can be used to predict numerical values too
- Instead of creating the widest street, fits most instances on the street
- The width of the street is controlled by `epsilon`



# SVM Regression

- Use kernelized SVM for polynomial models



# Python Cheatsheet

- `C`: regularization
  - Small C: wide margin, **allows more violations (i.e., generalizable)**
  - High C: small margin, **allows less violations (i.e., overfitting)**
- `coef0`: used for poly kernel
  - Controls how much the model is influenced by higher degree polynomials
- `gamma`: the shape of the bell for Gaussian RBF
  - Higher values make it narrower
  - Smaller values make it wider
- `tol`: precision parameter (if to large, may not converge, too low, and computation will take time)
- `epsilon`: width of the margin in regression (smaller value, more tightly fitting)

- See:
  - https://scikit-learn.org/stable/modules/svm.html