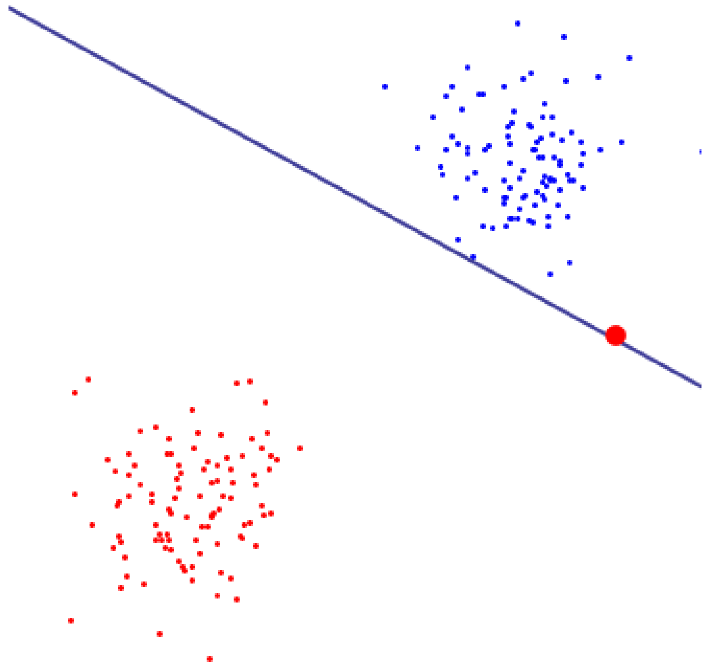
SUPERVISED MACHINE LEARNING: SUPPORT VECTOR MACHINES

Group members: Yuanqing, Nanxi, Lei, Juan, Aaron

SVM CONCEPT

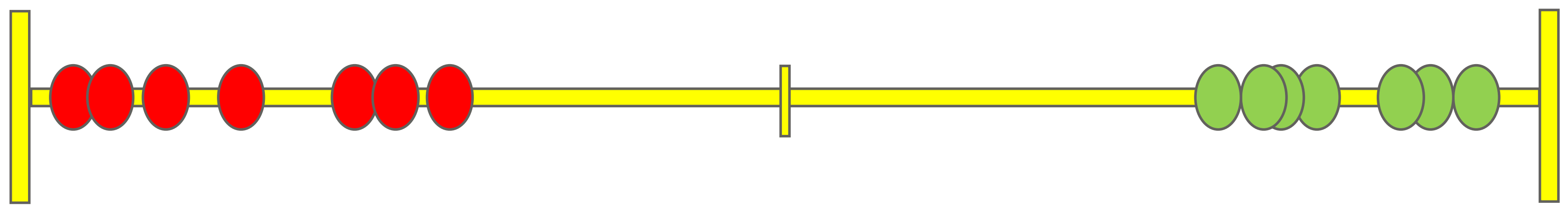
- **Hard Margins**: The core idea behind SVM is to find a decision boundary (or hyperplane) that best separates the classes of data while maximizing the distance between the closest data points of each class and the boundary.

- The term "hard margin" implies a strict separation of classes without any violations, meaning all data points from one class lie on one side of the hyperplane and those from the other class lie on the opposite side.

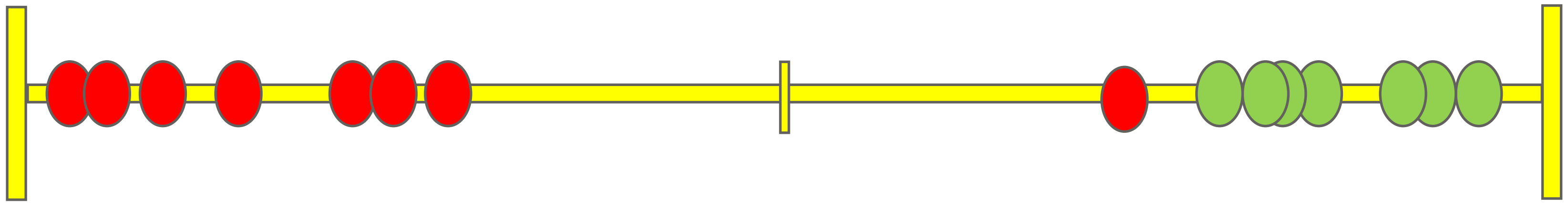


1 DIMENSION

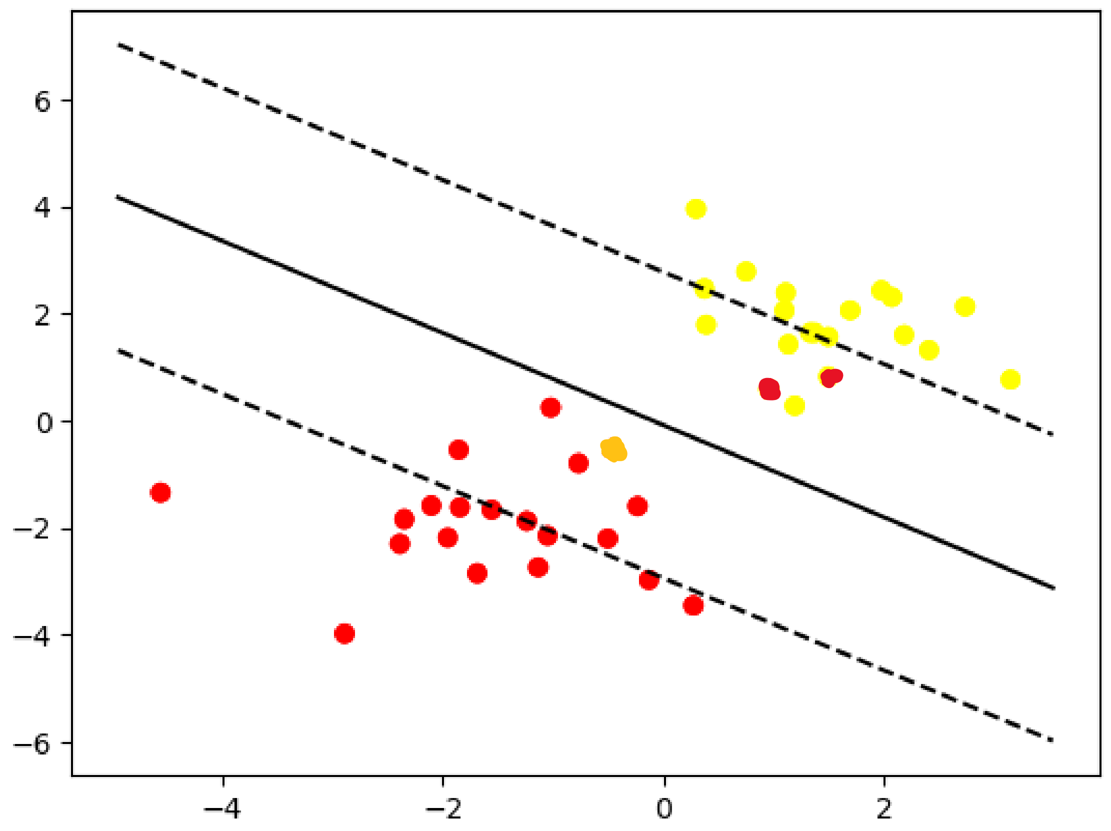
Weight (lbs)



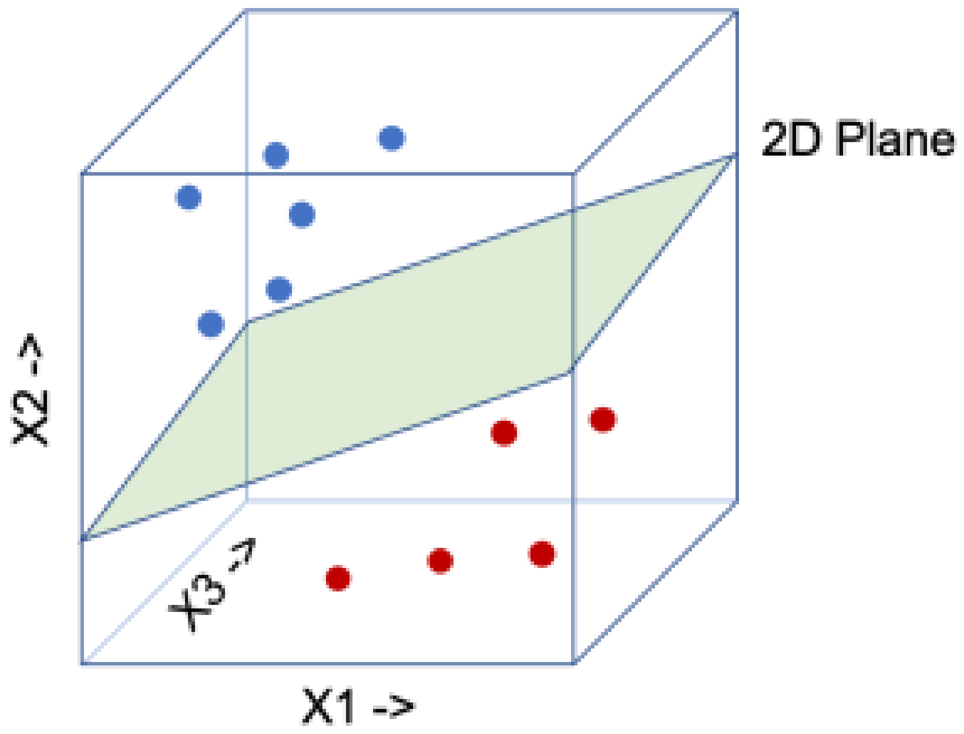
Weight (lbs)



2 DIMENSION

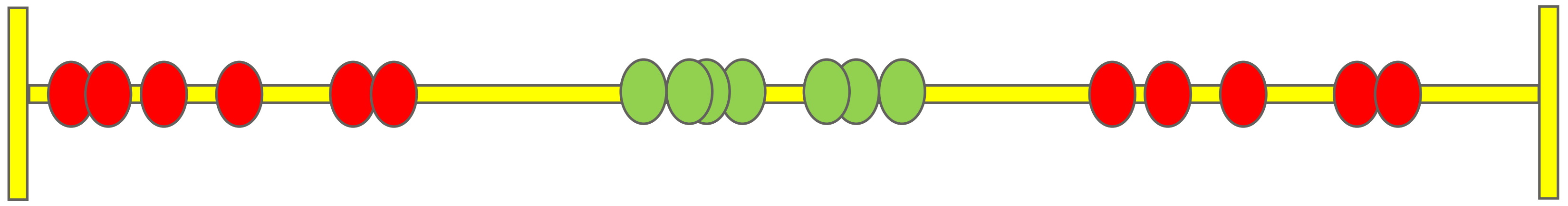


3 DIMENSION

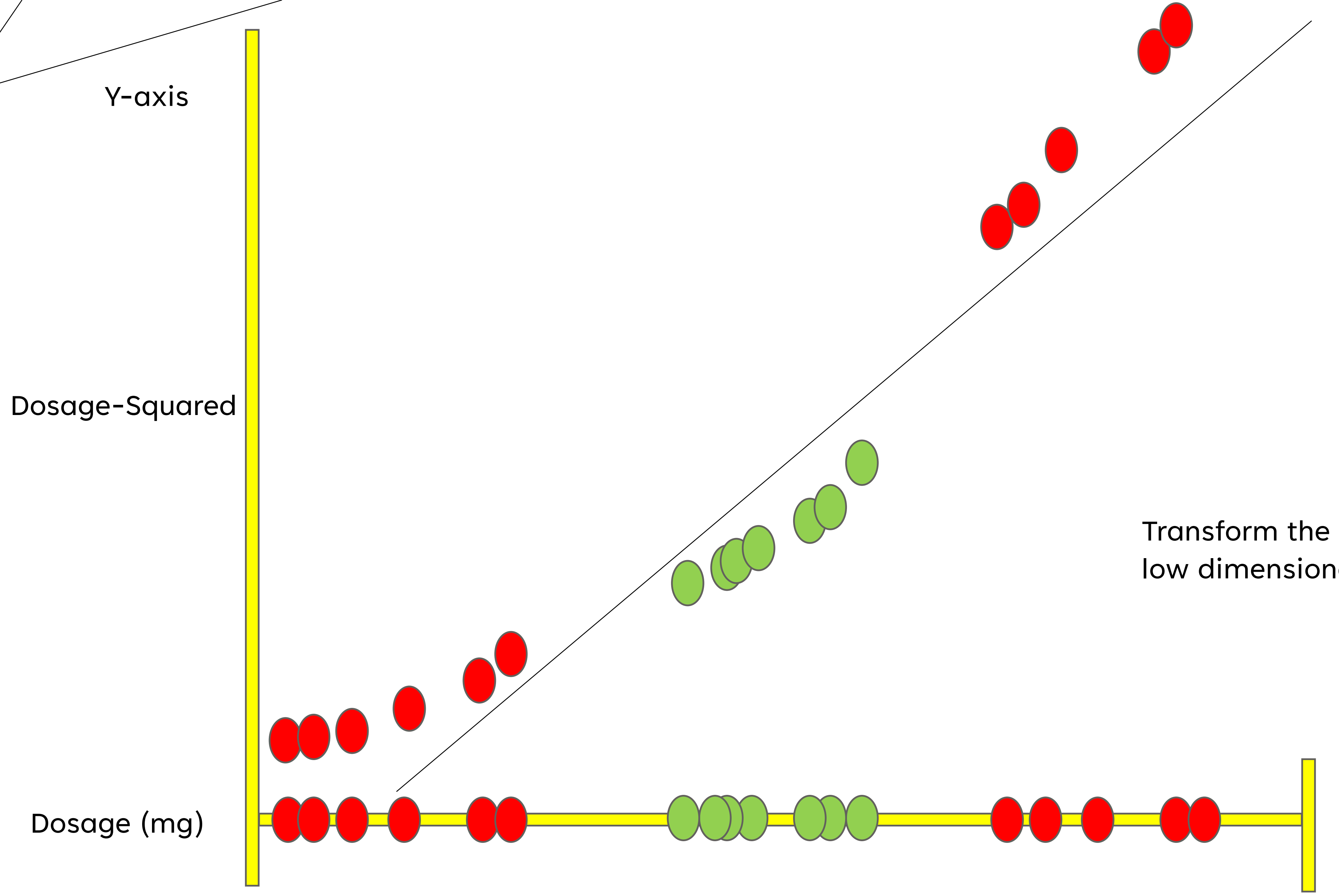


WHAT IF?

Dosage (mg)



No matter where you put the threshold, we will get bad classifications



Transform the values from low dimensionality

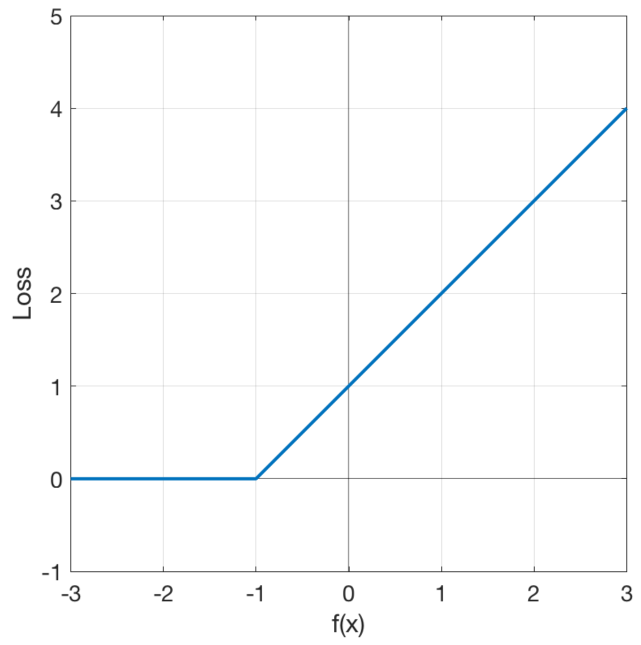
HINGE LOSS

- **Soft Margin & Hinge Loss**: Soft margin SVM allows for some misclassifications by introducing a "slack" variable, making the model more generalizable. The hinge loss is the cost function used to penalize misclassifications in the soft margin setting.

- For an intended output t = ±1 and a classifier score y, the hinge loss of the prediction y is defined as:

$$\ell(y) = \max(0, 1 - t \cdot y)$$

y is the output of the classifier; $y = w \cdot x + b$



MATHEMATICAL FOUNDATIONS OF LINEAR SUPPORT VECTOR MACHINES

- **Hyperplane Equation**: In an N-dimensional space, the equation of a hyperplane can be defined as $w \cdot x + b = 0$, where w is the weight vector perpendicular to the hyperplane, x is any point on the hyperplane, and b is the bias.

- **Incorporation of Hard Margins**: For a linearly separable dataset, we can define two hyperplanes for the two classes as: $w \cdot x + b = 1$ (for positive class) $w \cdot x + b = -1$ (for negative class).

- **Maximization of Margin**: The objective of SVM is to maximize the margin, or equivalently, minimize ||w||, the magnitude of the weight vector. The distance between the two hyperplanes is a half by ||w||, and by minimizing ||w||, we're maximizing this distance.

LINEAR SUPPORT VECTOR MACHINES

- **Support Vectors and Decision Making**: The SVM decision function is essentially an inner product between the input vector and the support vectors.

Given a dataset (x1 ,y1),(x2 ,y2),...,(xn ,yn) where X_i are data points and y_i are labels (+1 or -1 for binary classification), the primal optimization problem is:

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot x_i + b))$$

W is the weight vector
B is the bias
C is the regularization parameter

A larger C values means we are giving more importance to the classification, and a smaller C values means we are giving more importance to maximizing the margin.

LAGRANGE MULTIPLIERS

- **Lagrange Multipliers for Constrained Optimization**: To solve the SVM optimization problem, we use Lagrange multipliers. This technique transforms the constrained optimization problem into an unconstrained one, making it easier to solve. The Lagrangian for SVM incorporates the objective function and the constraints, leading to a dual problem that can be solved efficiently.

$$\mathcal{L}(x, \lambda) = f(x) - \lambda g(x)$$

LAGRANGE MULTIPLIER IN SVM

$$L(w,b,\xi,\alpha,r) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i(w\cdot x_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{N} r_i\xi_i$$

w is the weight vector.
b is the bias term.
ξi are the slack variables.
C is a regularization parameter.
xi and yi are the data points and their corresponding labels.
αi and ri are the Lagrange multipliers.

To find the dual problem, we need to minimize L with respect to the primal variables w, b, and ξ.

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \alpha_i y_i x_i \qquad (1)$$

$\Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i (2)$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i \qquad (3)$$

$\Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 (4)$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - r_i \qquad (5)$$

$\Rightarrow \alpha_i + r_i = C (6)$

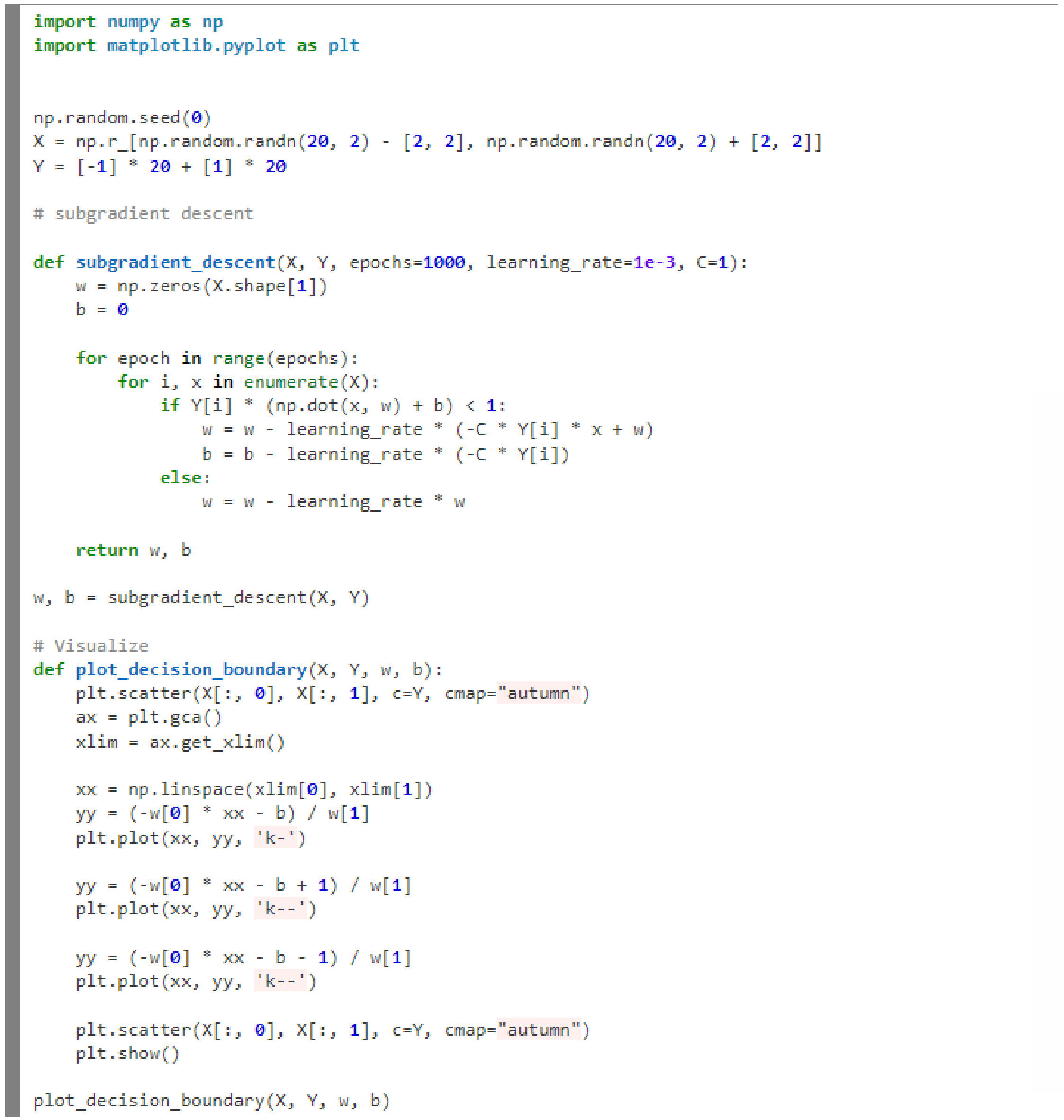
PLUG ISOLATED VARIABLES

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

Now, the problem is to maximize L(α) with respect to α under the constraints:

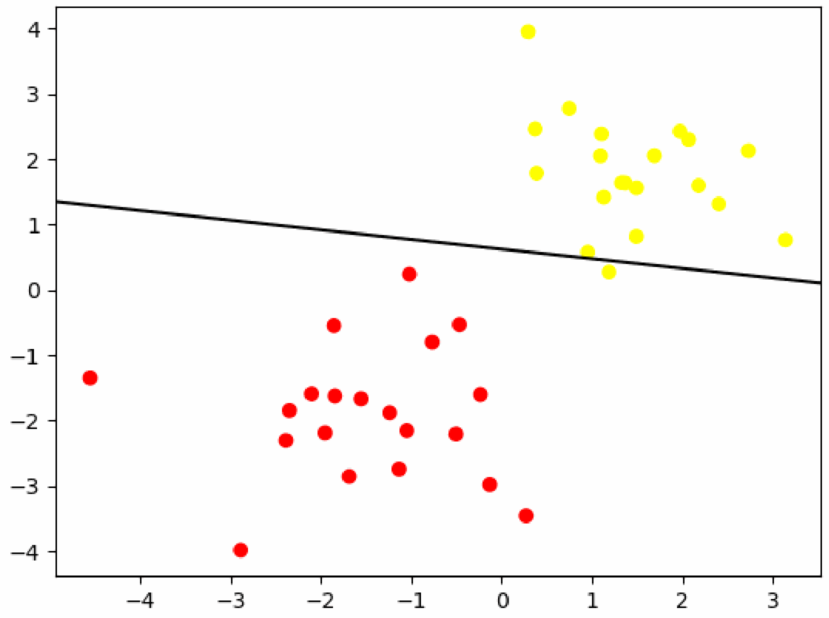
$$\alpha_i \geq 0$$
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

EXAMPLE BY HAND



```
import numpy as np
import matplotlib.pyplot as plt


np.random.seed(0)
X = np.r_[np.random.randn(20, 2) - [2, 2], np.random.randn(20, 2) + [2, 2]]
Y = [-1] * 20 + [1] * 20

# subgradient descent

def subgradient_descent(X, Y, epochs=1000, learning_rate=1e-3, C=1):
    w = np.zeros(X.shape[1])
    b = 0

    for epoch in range(epochs):
        for i, x in enumerate(X):
            if Y[i] * (np.dot(x, w) + b) < 1:
                w = w - learning_rate * (-C * Y[i] * x + w)
                b = b - learning_rate * (-C * Y[i])
            else:
                w = w - learning_rate * w

    return w, b

w, b = subgradient_descent(X, Y)

# Visualize
def plot_decision_boundary(X, Y, w, b):
    plt.scatter(X[:, 0], X[:, 1], c=Y, cmap="autumn")
    ax = plt.gca()
    xlim = ax.get_xlim()

    xx = np.linspace(xlim[0], xlim[1])
    yy = (-w[0] * xx - b) / w[1]
    plt.plot(xx, yy, 'k-')

    yy = (-w[0] * xx - b + 1) / w[1]
    plt.plot(xx, yy, 'k--')

    yy = (-w[0] * xx - b - 1) / w[1]
    plt.plot(xx, yy, 'k--')

    plt.scatter(X[:, 0], X[:, 1], c=Y, cmap="autumn")
    plt.show()

plot_decision_boundary(X, Y, w, b)
```

Scikit-learn uses more complex optimization method than this example. Those others being Sequential Minimal Optimization or Quadratic Programming

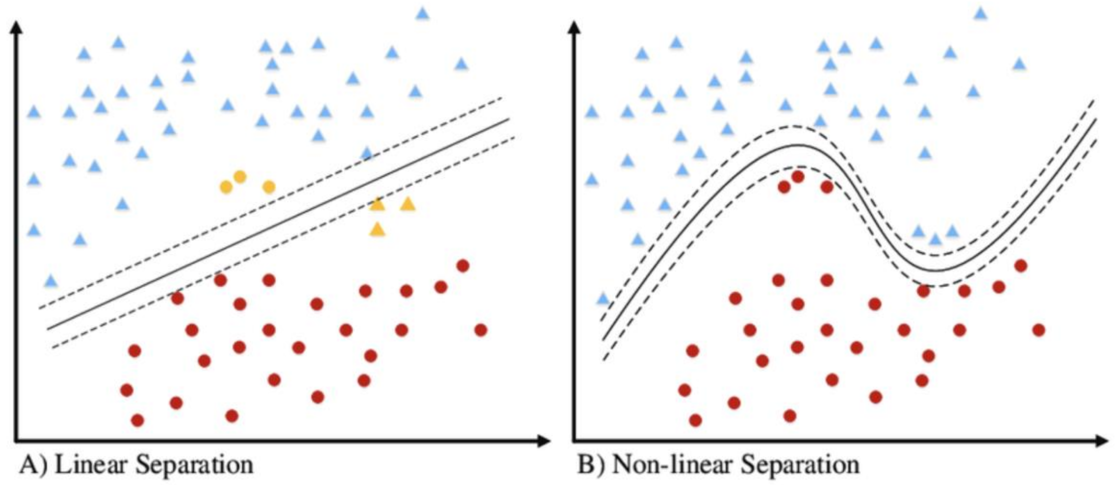


NON-LINEAR SUPPORT VECTOR MACHINES

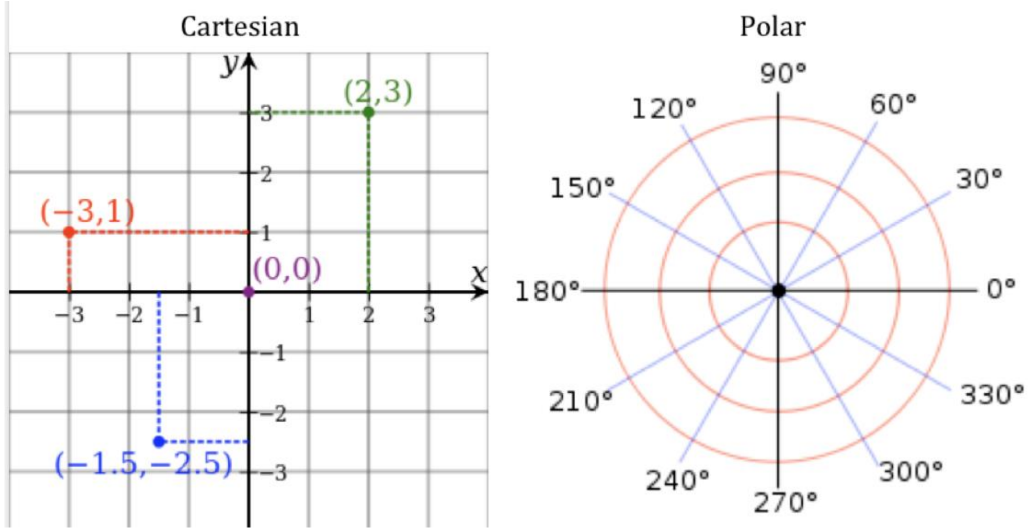
Lei Wang

NON-LINEAR SUPPORT VECTOR MACHINES

Motivation (Limitation of Linear Separation):

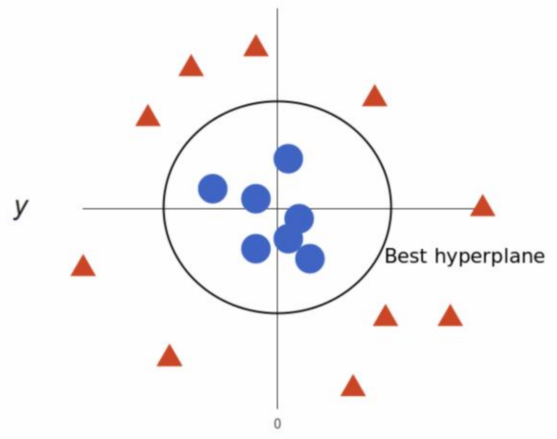


Cartesian vs. Polar:



Be care for: Potential overfitting!

- **Nonlinear Decision Boundaries:** Nonlinear SVM allows for the creation of complex decision boundaries that can accurately separate data points of different classes.
- By transforming the data using kernel functions, SVM can capture nonlinear relationships between features.

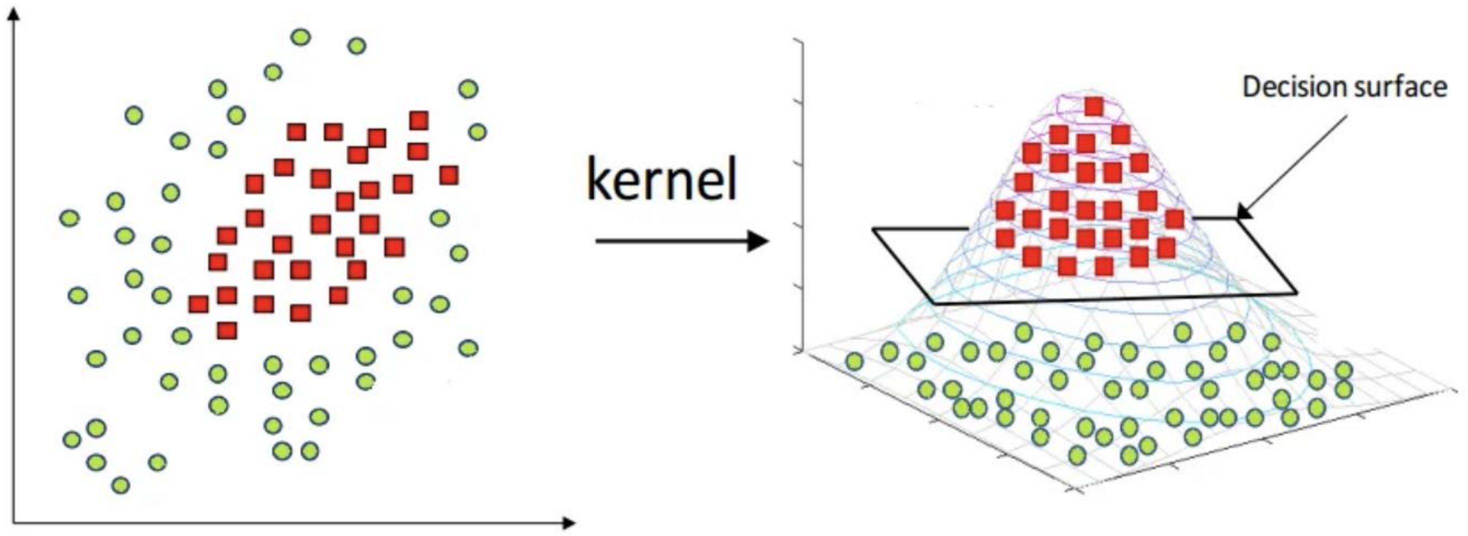


NON-LINEAR SUPPORT VECTOR MACHINES

Kernel: Kernel is a mathematical function used in SVM to map the original input data points into high-dimensional feature spaces

$$K: X \times X \rightarrow R$$
$$K(\vec{x}, \vec{z}) = < \phi(\vec{x}), \phi(\vec{z}) >$$

Hyperplane: Hyperplane is the decision boundary that is used to separate the data points of different classes in a feature space.



NON-LINEAR SUPPORT VECTOR MACHINES

Kernel Trick:

However, when there are more and more dimensions, computations within that space become more and more expensive. This is when the kernel trick comes in.

It allows us to operate in the original feature space without computing the coordinates of the data in a higher dimensional space.

Example:

$$\mathbf{x} = (x_1, x_2, x_3)^T$$
$$\mathbf{y} = (y_1, y_2, y_3)^T$$

Feature space

$$\phi(\mathbf{x}) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)^T$$
$$\phi(\mathbf{y}) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2)^T$$

$$\phi(\mathbf{x})^T\phi(\mathbf{y}) = \sum_{i,j=1}^{3} x_ix_jy_iy_j$$

Kernel trick

$$\begin{aligned} k(\mathbf{x},\mathbf{y}) &= (\mathbf{x}^T\mathbf{y})^2 \\ &= (x_1y_1 + x_2y_2 + x_3y_3)^2 \\ &= \sum_{i,j=1}^{3} x_ix_jy_iy_j \end{aligned}$$

NON-LINEAR SUPPORT VECTOR MACHINES

Kernel Example:

Let $\phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$

Let $\vec{x}$=(1,2) $\vec{z}$=(-2,3)

$\phi(\vec{x}) = (1, 4, 2\sqrt{2})$ $\phi(\vec{z}) = (4, 9, -6\sqrt{2})$

Feature space

$$\boxed{\begin{aligned} K(\vec{x},\vec{z}) &=< \phi(\vec{x}), \phi(\vec{z}) > \\ &=< (1,4,2\sqrt{2}), (4,9,-6\sqrt{2}) > \\ &= 1*4 + 4*9 - 2*6*2 = 16 \end{aligned}}$$

$< \vec{x}, \vec{z} >= -2 + 2*3 = 4$

Let $\phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$

$$\boxed{K(\vec{x},\vec{z})}$$

$$=< \phi(\vec{x}), \phi(\vec{z}) >$$

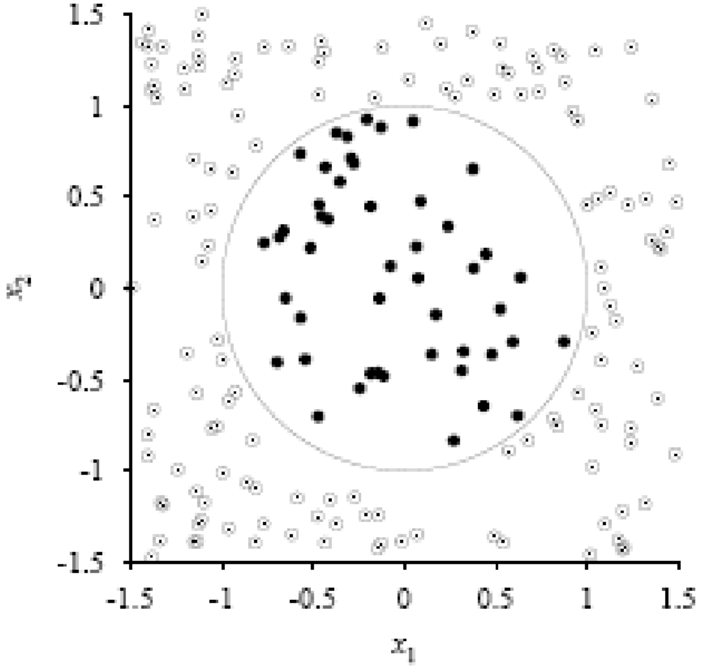
$$=< (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) >$$

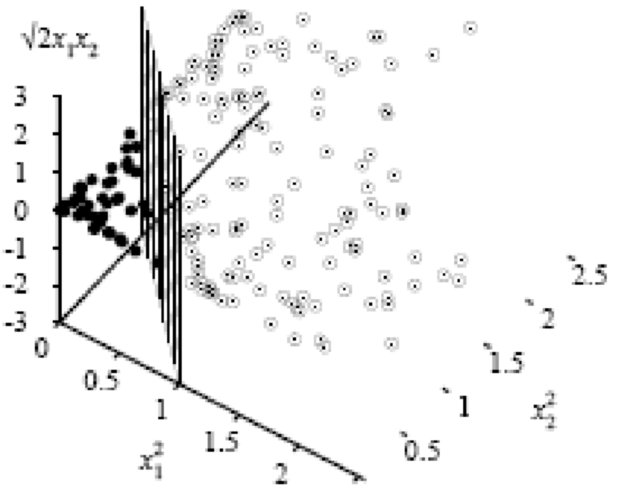
$$= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1z_1x_2z_2$$

$$= (x_1z_1 + x_2z_2)^2$$

$$= \boxed{< \vec{x}, \vec{z} >^2}$$

Kernel Trick





COMMON INNER PRODUCT KERNELS USED WITH SVM:

Polynomial:

Definition:
For degree-d polynomials, the polynomial kernel is defined as

$$K(x, y) = (x^\mathsf{T} y + c)^d$$

$c \geq 0$ is a free parameter trading off the influence of higher-order versus lower-order terms in the polynomial.
Note: c here is only a constant parameter for polynomial kernel, not the Upper C which is the regularization parameter.

Consider the previous linear model (c=0, d=1):

(linear) $K(x, y) = x^\mathsf{T} y \equiv K(x, y) = (x^\mathsf{T} y + c)^d$ (polynomial)

For example, sklearn's polynomial kernel assumes by default c=1 if you do not specify the parameter.

Limits:
One problem with the polynomial kernel is that it may suffer from numerical instability: when $x^\mathsf{T}y + c < 1$, $K(x, y) = (x^\mathsf{T}y + c)^d$ tends to zero with increasing d, whereas when $x^\mathsf{T}y + c > 1$, $K(x, y)$ tends to infinity.
The most common degree is $d = 2$ (quadratic), since larger degrees tend to overfit.

NON-LINEAR SUPPORT VECTOR MACHINES

Common inner product kernels used with SVM:

Radial basis function (RBF) (Gaussian):

The RBF kernel on two samples $\mathbf{x} \in \mathbb{R}^k$ and **x'**, represented as feature vectors in some *input space*,

$$K(\mathbf{x}, \mathbf{x'}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x'}\|^2}{2\sigma^2}\right)$$

$\|\mathbf{x} - \mathbf{x'}\|^2$ may be recognized as the squared Euclidean distance between the two feature vectors.

Sigma is a free parameter. An equivalent definition involves a parameter $\gamma = \frac{1}{2\sigma^2}$

$$K(\mathbf{x}, \mathbf{x'}) = \exp(-\gamma\|\mathbf{x} - \mathbf{x'}\|^2)$$

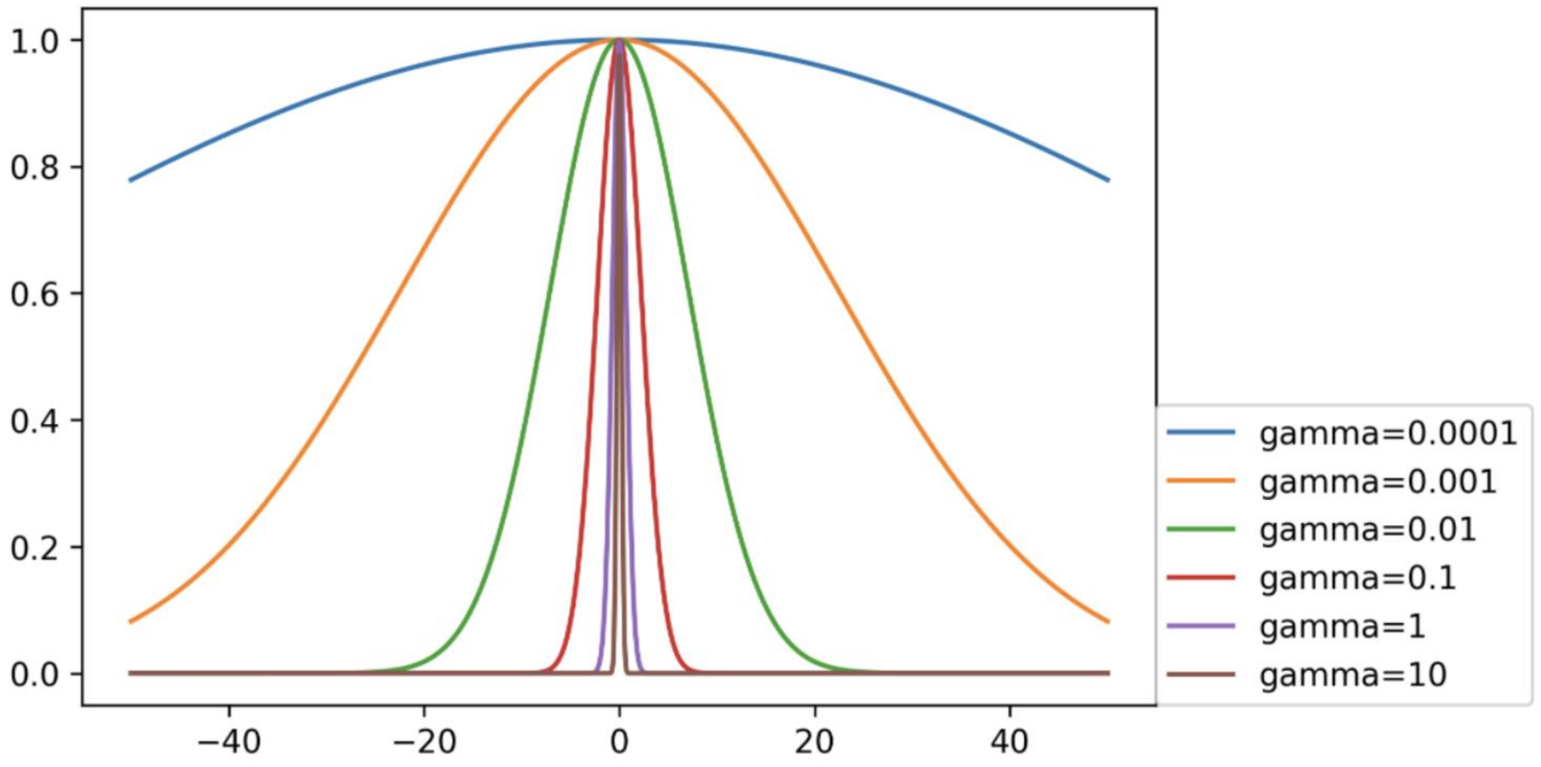
Gamma is bandwidth

NON-LINEAR SUPPORT VECTOR MACHINES

Common inner product kernels used with SVM:

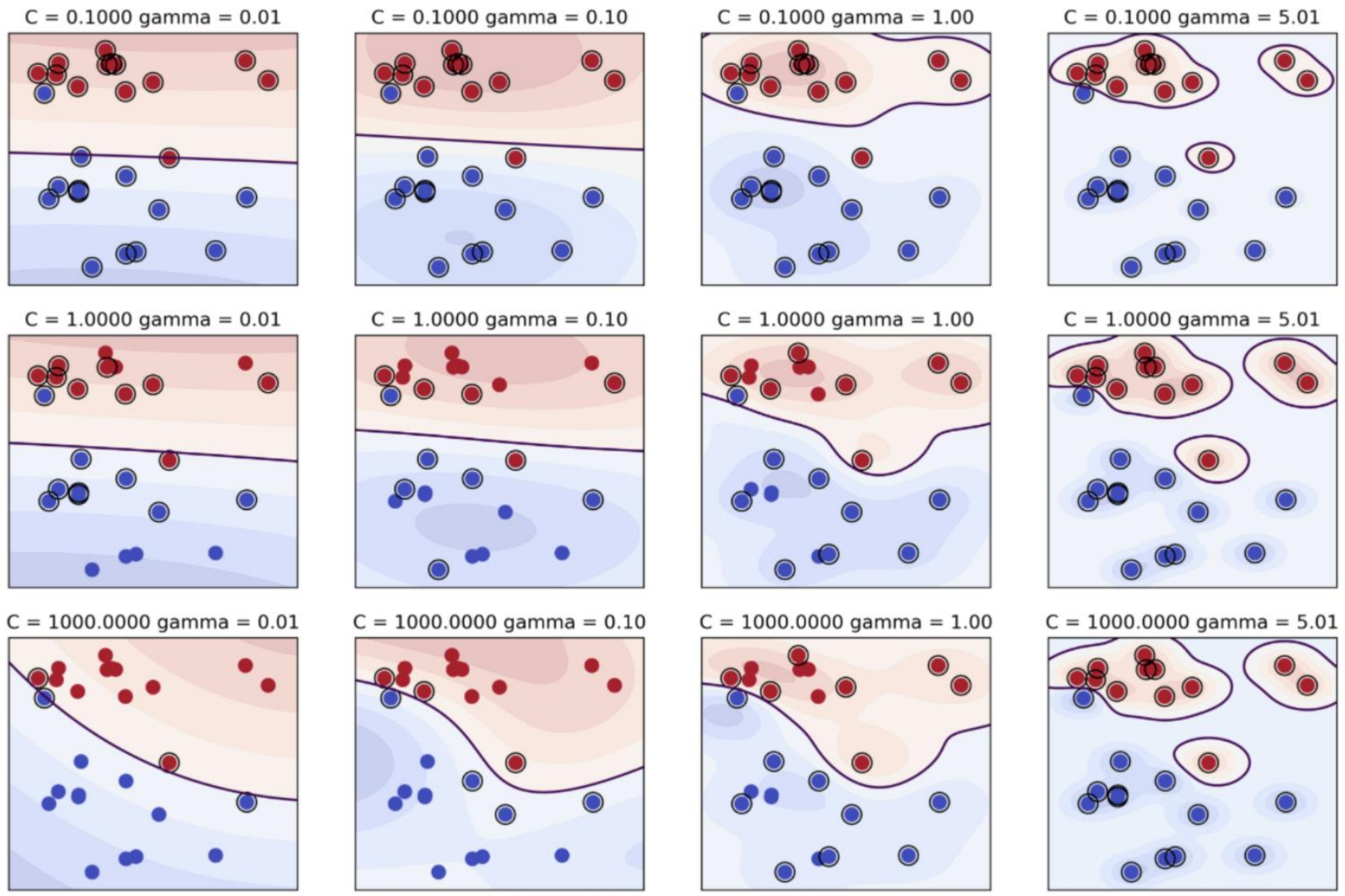
Radial basis function (RBF) (Gaussian):
Gamma is bandwidth

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2) \qquad \gamma = \frac{1}{2\sigma^2}$$



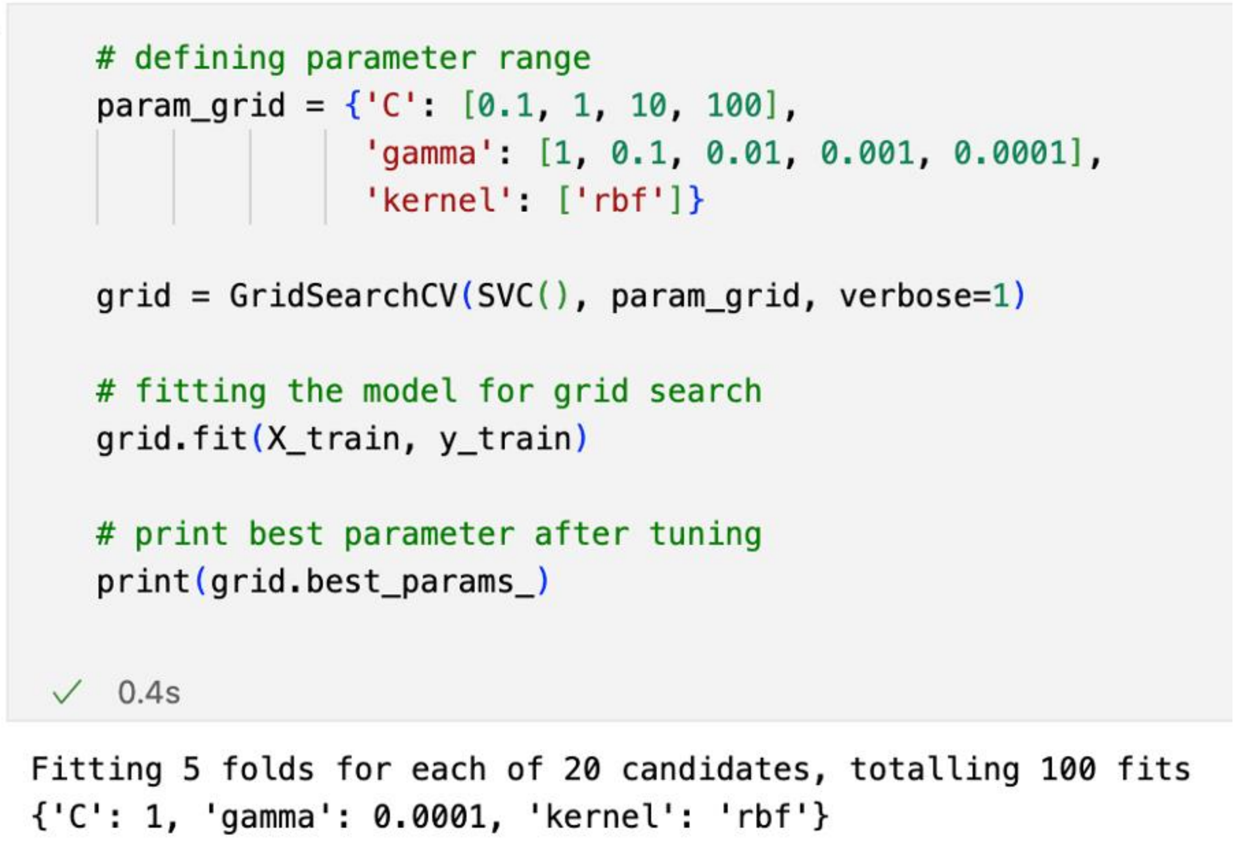
HYPER-PARAMETER EXAMPLE (GAUSSIAN)

- **Regularization parameter C (for any kernel)**
- **Gamma is bandwidth:** $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$



HYPER-PARAMETER

- **Grid search:**
- The usual way to search for the hyperparameter values is by combining each of the proposed values through a **grid search** along with a procedure that applies those hyperparameter values and obtains metrics for different parts of the data called **cross validation**.
- In Scikit-Learn, this is already implemented as the GridSearchCV (CV from cross validation) method.



```
# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, verbose=1)

# fitting the model for grid search
grid.fit(X_train, y_train)

# print best parameter after tuning
print(grid.best_params_)
```

✓ 0.4s

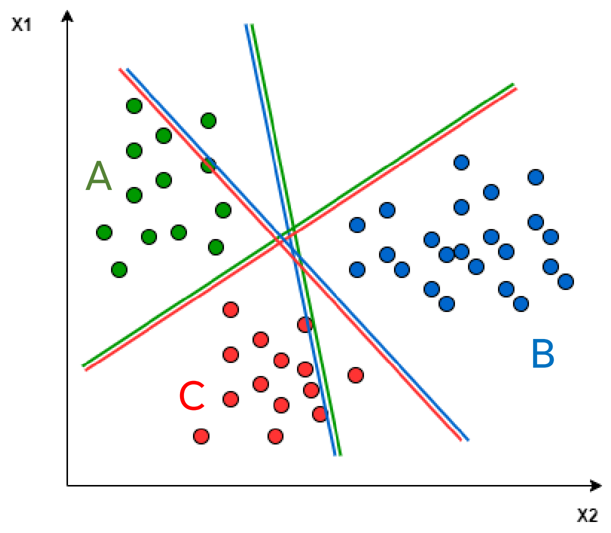
```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
{'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
```

CONSIDERATIONS WITH SUPPORT VECTOR MACHINES

Nanxi Guo

1 Extension of SVM to handle multiclass classification

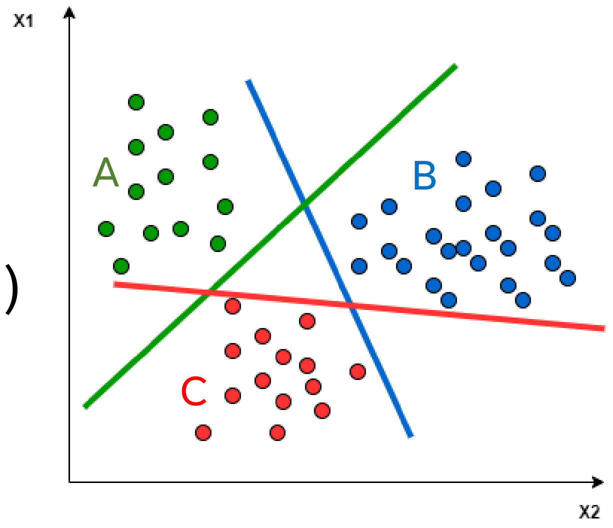
SVM does **NOT** support multiclass classification natively!



- Assume three (k) classes: A, B and C
- **One-to-One approach**

  A vs B, B vs C and A vs C - k(k-1)/2 classifiers

  select class with the most votes as output



- **One-to-rest approach**

  A vs (B+C), B vs (A+C), C vs (A+B) - k classifiers

  Select class with the highest probability (transformation)

https://www.baeldung.com/cs/svm-multiclass-classification

2 Necessary of feature scaling

- **Distance computation** between hyperplane and support vectors

  Larger scales features dominate the distance computation

Biased decision boundary!

- **Kernel functions**

  e.g., Gaussian kernel $K\left(X_i, X_j\right) = \exp(-\gamma\left\|X_i - X_j\right\|^2)$

Biased similarity measure!

- **Standardization (Z-score normalization):** $X_{standardized} = \frac{X - mean}{standard\ deviation}$
- **Min-Max Scaling:** $X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$
- **Robust Scaling:** $X_{robust-scaled} = \frac{X - median}{IQR}$

3 Affect of outliers

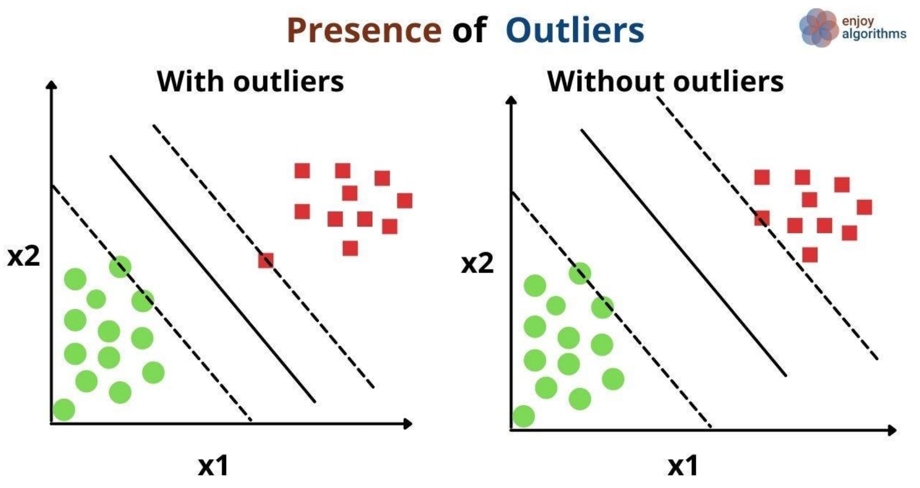
- **Hard Margin SVM** simply cannot tolerate outliers

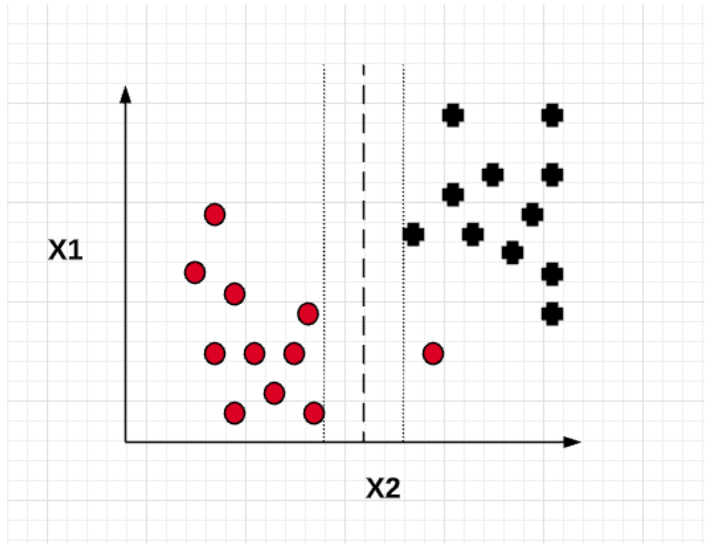
  Pull the **decision boundary** away from the optimal location – outliers serve as support vectors!

- **Soft Margin SVM is more robust to outliers**

  Controls the trade-off between maximizing margin and minimizing misclassification

  More flexible and robust to noisy





https://www.enjoyalgorithms.com/blog/support-vector-machine-in-ml
https://vitalflux.com/svm-soft-margin-classifier-c-value-importance/

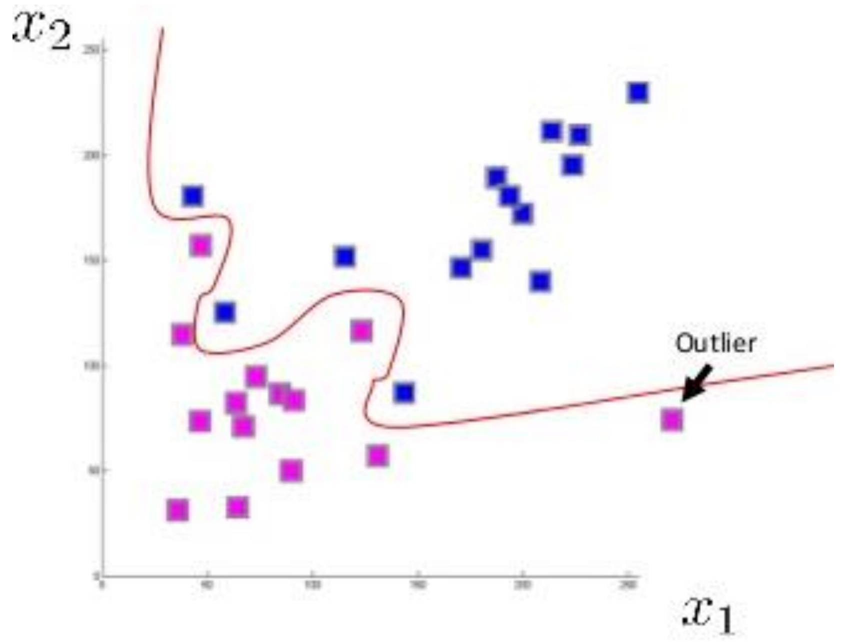
3 Affect of outliers

- **Kernel selection**

  e.g., Gaussian kernel can be highly sensitive to outliers

- **Overfitting**

  Decision boundary is overly influenced by the outliers – overfitting model



4 Computational implications - large number of samples or features

- SVM tends to overfitting with a large number of features if the regularization parameter **isn't** chosen properly.

$$\min_{w,b} \frac{1}{2}||w||^2 + \boxed{C} \sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot x_i + b))$$

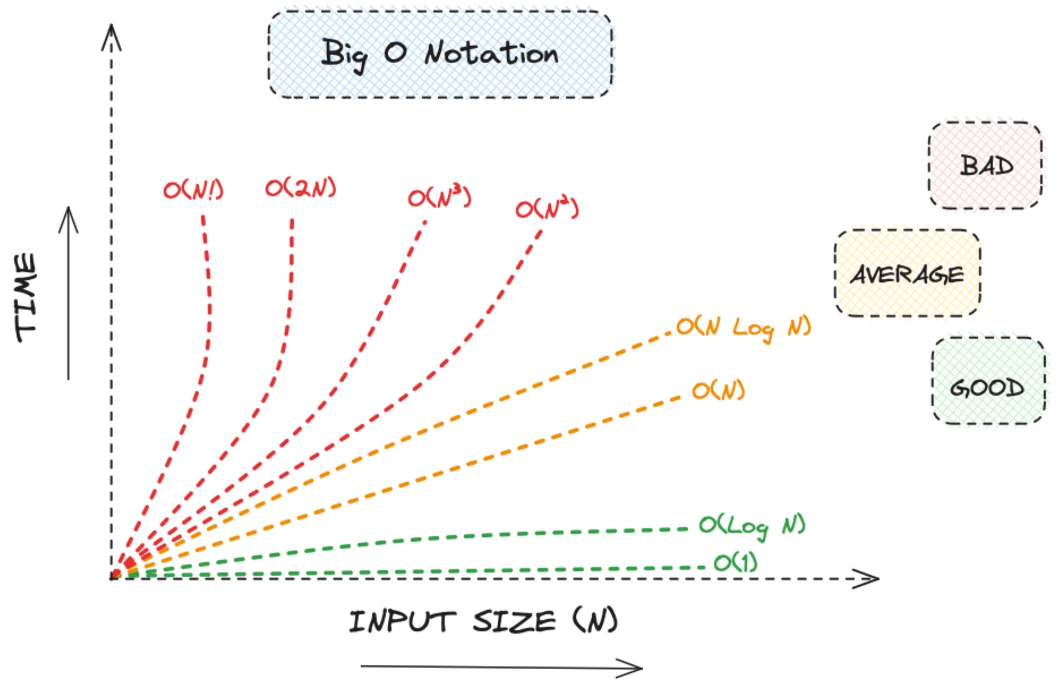
C: to relax the hard margin
Trade-off training error vs model generalizability

P.S. Subset features or employ dimensionality reduction (e.g. PCA) to reduce feature space without losing too much information...

https://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/15_svm.pdf

4 Computational implications - large number of samples or features

- **Training complexity** of *non-linear* SVM is generally between $O(n^2)$ to $O(n^3)$ with the number of input samples *n* – computational time grows quadratically or cubically.



https://medium.com/@hlfdev/algorithms-discover-the-power-of-big-o-notation-17a367bd62a

5 Overfitting issue – makes model less generalizable

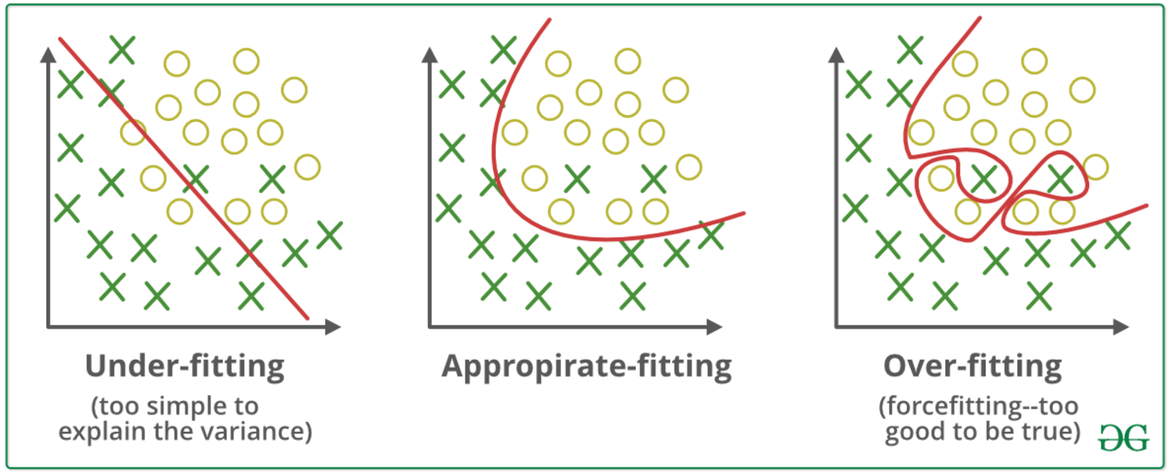
- **Non-linear SVM:**

  Complex decision boundaries - fitting the training data exceptionally well

  Parameter selection can significantly impact the risk of overfitting (e.g., regularization parameter *c*)

- A **linear SVM** is generally less prone to overfitting compared to a non-linear SVM

  Still can overfit if the regularization parameter (*C*) is not chosen correctly, or if the feature space is high-dimensional with many irrelevant features


Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

INTERPRETING PREDICTIONS

Yuanqing Liu

INTERPRETING PREDICTIONS
MEANING OF DISTANCE TO MODEL BOUNDARY

- Prediction based on signed distance of data point from the decision boundary
- Decision rule: $w \cdot x + b \geq 0$ then class A, otherwise class B
- Sign of distance determines predicted class label
- Magnitude of distance represents how far data point is from the decision boundary, larger magnitude indicates greater confidence in classification

$$\frac{|w \cdot x + b|}{||w||}$$

INTERPRETING PREDICTIONS
PLATT'S SCALING ALGORITHM

- Platt (1999) proposed a calibration method for SVM
- Main idea: Transform output of SVM to probability distribution through logistic regression

$$P(y=1|f(x)) = \frac{1}{1+\exp(Af(x)+B)}$$

where f(x)=w·x+b, decision function of SVM

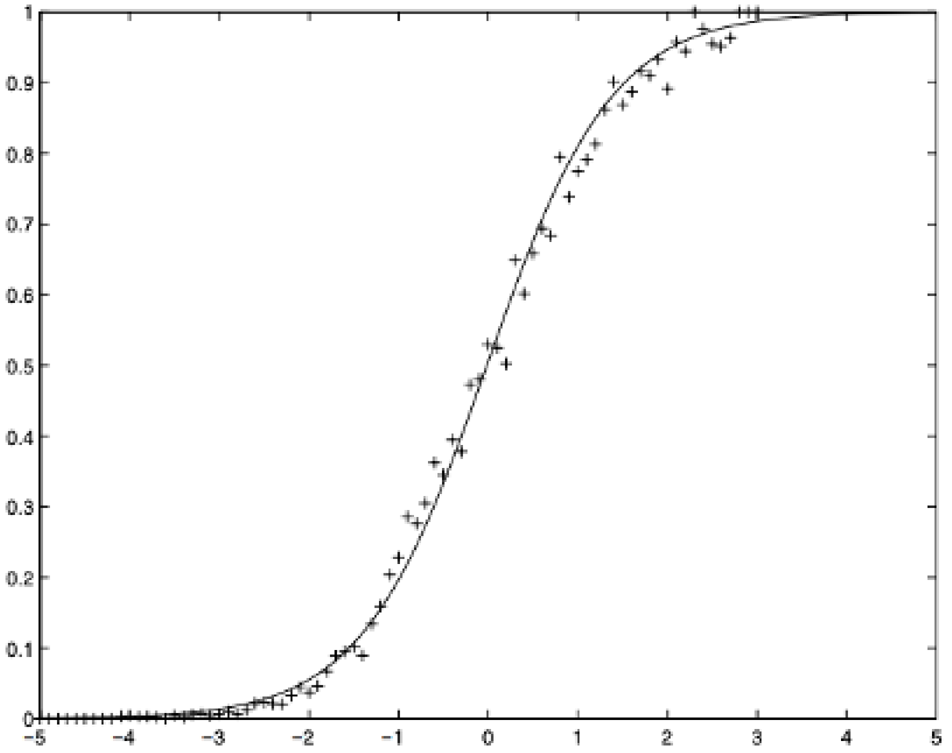
- A and B can be solved using maximum likelihood estimation from a training set (f_i, y_i)

$$\underset{A,B}{\operatorname{argmin}}\{-\sum_i y_i \log(p_i) + (1-y_i)\log(1-p_i)\}$$

where

$$p_i = \frac{1}{1+\exp(Af_i+B)}$$

INTERPRETING PREDICTIONS
VISUAL REPRESENTATION OF PLATT'S SCALING



Source: Platt, John. "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods." Advances in large margin classifiers 10.3 (1999): 61-74.

INTERPRETING PREDICTIONS
CONSIDERATIONS OF PLATT'S SCALING

- Can we use the same dataset to train SVM and calibrate the model?

No, bias can be introduced especially for non-linear SVM

Solution: Use a hold-out set or cross-validation

- How to avoid overfitting?

Solution: Use out-of-sample model

Out-of-sample data modeled with same empirical density as training data, but with a finite probability of opposite label

INTERPRETING PREDICTIONS
DISCUSSION ON PLATT'S SCALING

- Assumption: sigmoidal relationship between the decision function and the class probabilities, logistic function can well approximate this relationship
- More general approach: Isotonic regression (only assume monotonically increasing mapping)
- Advantage: Robust to overfitting and require less training data than isotonic regression
- Designed for binary classification, other methods (one-vs-rest or softmax scaling) for multi-class classification

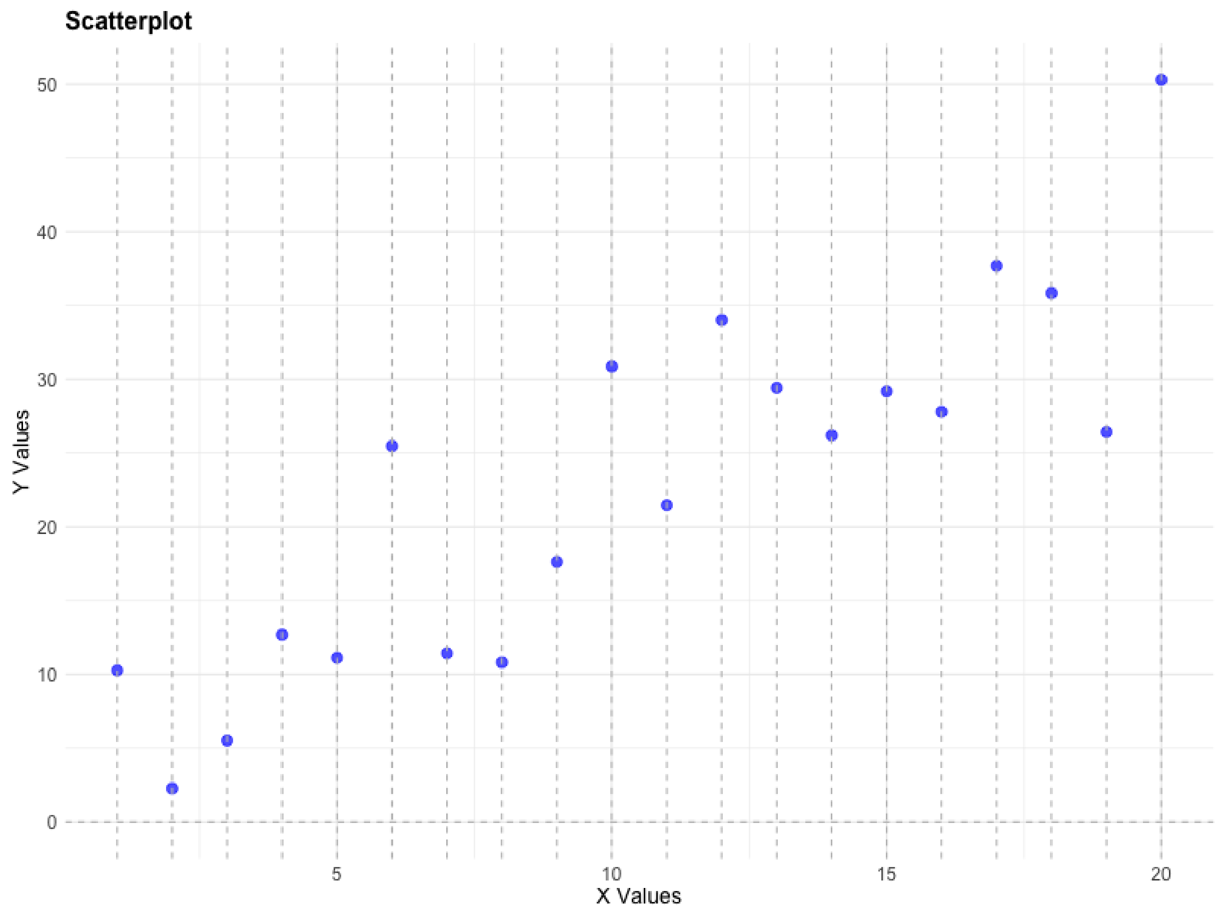
SUPPORT VECTOR REGRESSION

Aaron Beckwith

Support vector regression

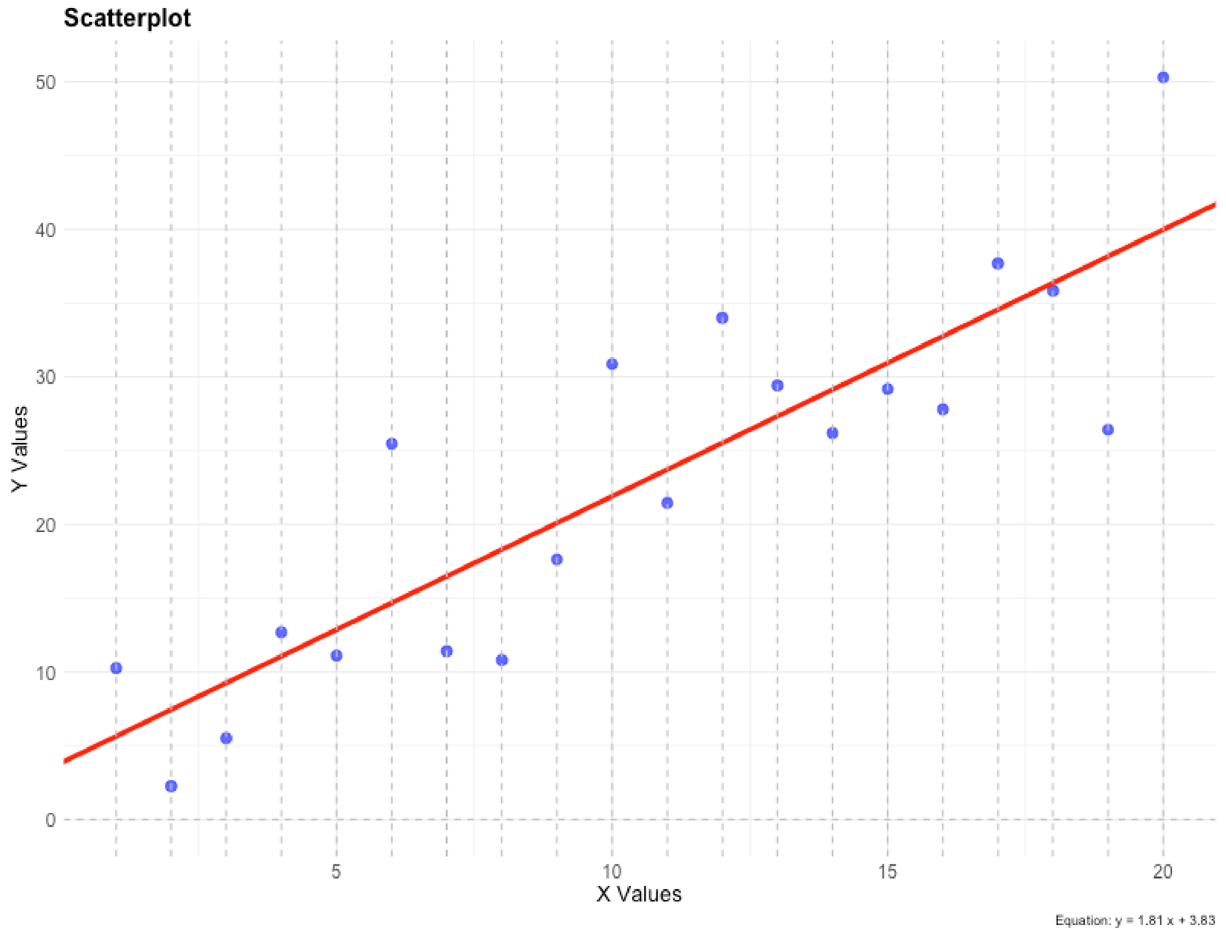
- A similar concept to support vector machine classifiers, but used for predicting continuous variables (Regression)

- Slightly different optimization problem than ordinary least-squares regression



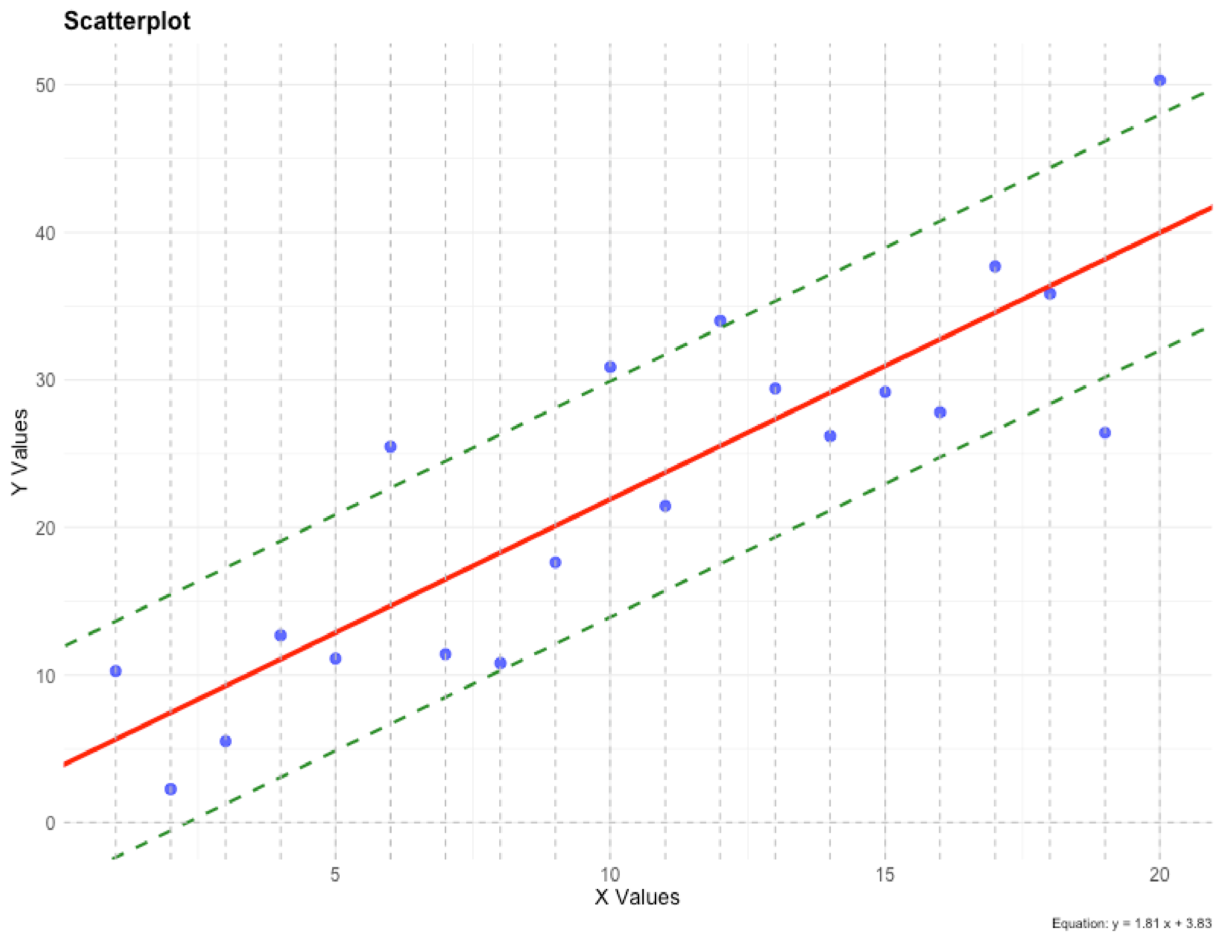
Support vector regression

- Ordinary Least-Squares
  - Minimize the residual distance from the fitted line
  - $\sum_{i=1}^{n}(y_i - wx_i)^2$



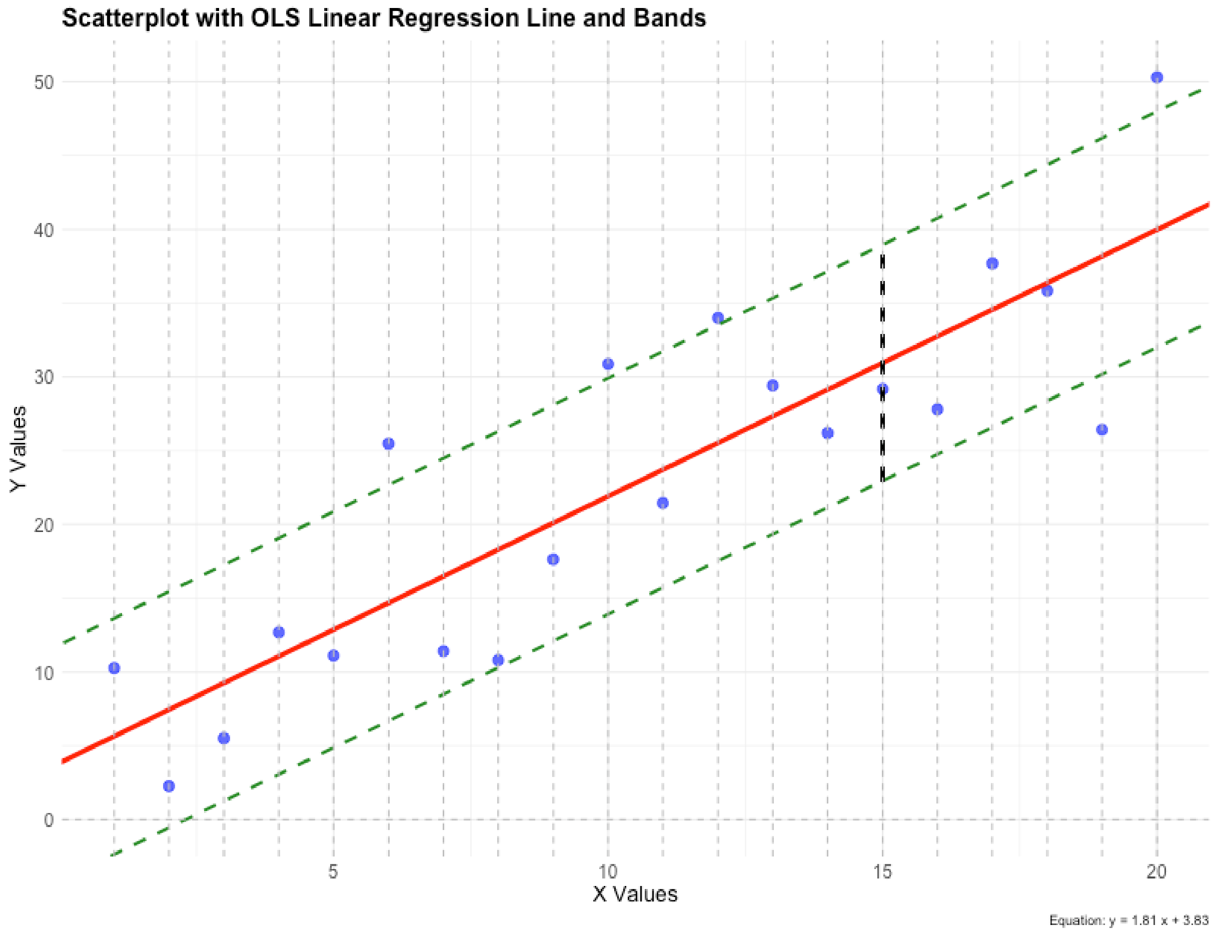
Support vector regression

- Ordinary Least-Squares
  - Minimize the residual distance from the fitted line
  - $\sum_{i=1}^{n} (y_i - wx_i)^2$
- Support Vector Regression
  - Minimize the slope of the fitted line, while maintaining as many observations as possible inside the margin



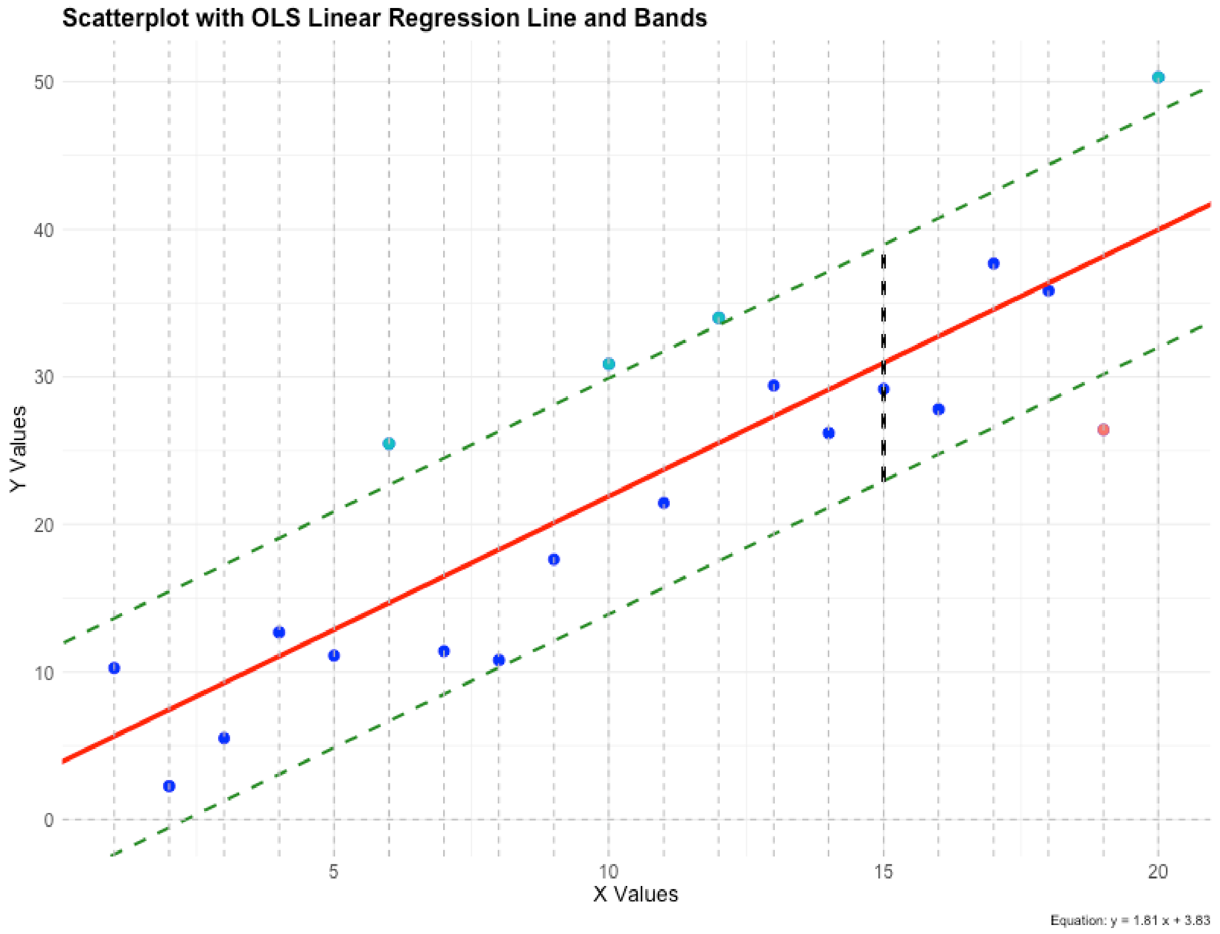
Support vector regression

- Ordinary Least-Squares
  - Minimize the residual distance from the fitted line
  - $\sum_{i=1}^{n} (y_i - wx_i)^2$
- Support Vector Regression
  - Minimize the slope of the fitted line, while maintaining as many observations as possible inside the margin



Support vector regression

- Ordinary Least-Squares
  - Minimize the residual distance from the fitted line
  - $\sum_{i=1}^{n}(y_i - wx_i)^2$
- Support Vector Regression
  - Minimize the slope of the fitted line, while maintaining as many observations as possible inside the margin
  - Observations outside of the margin are often called "slack" variables.



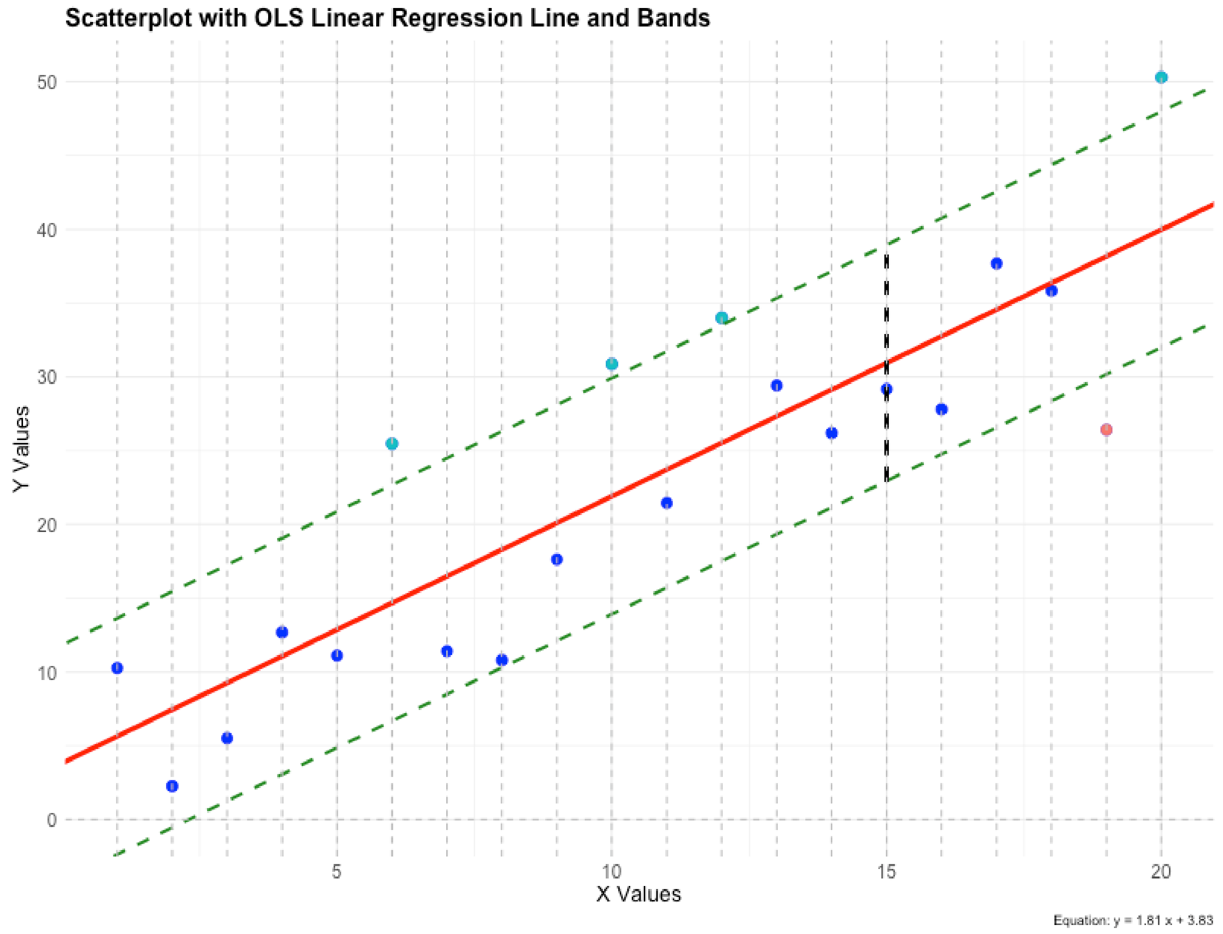
Support vector regression

Minimize:

$$\frac{1}{2}||w||^2$$

Constrained on the assumption:

$$|y_i - < w, x_i > \ -b| < \epsilon$$



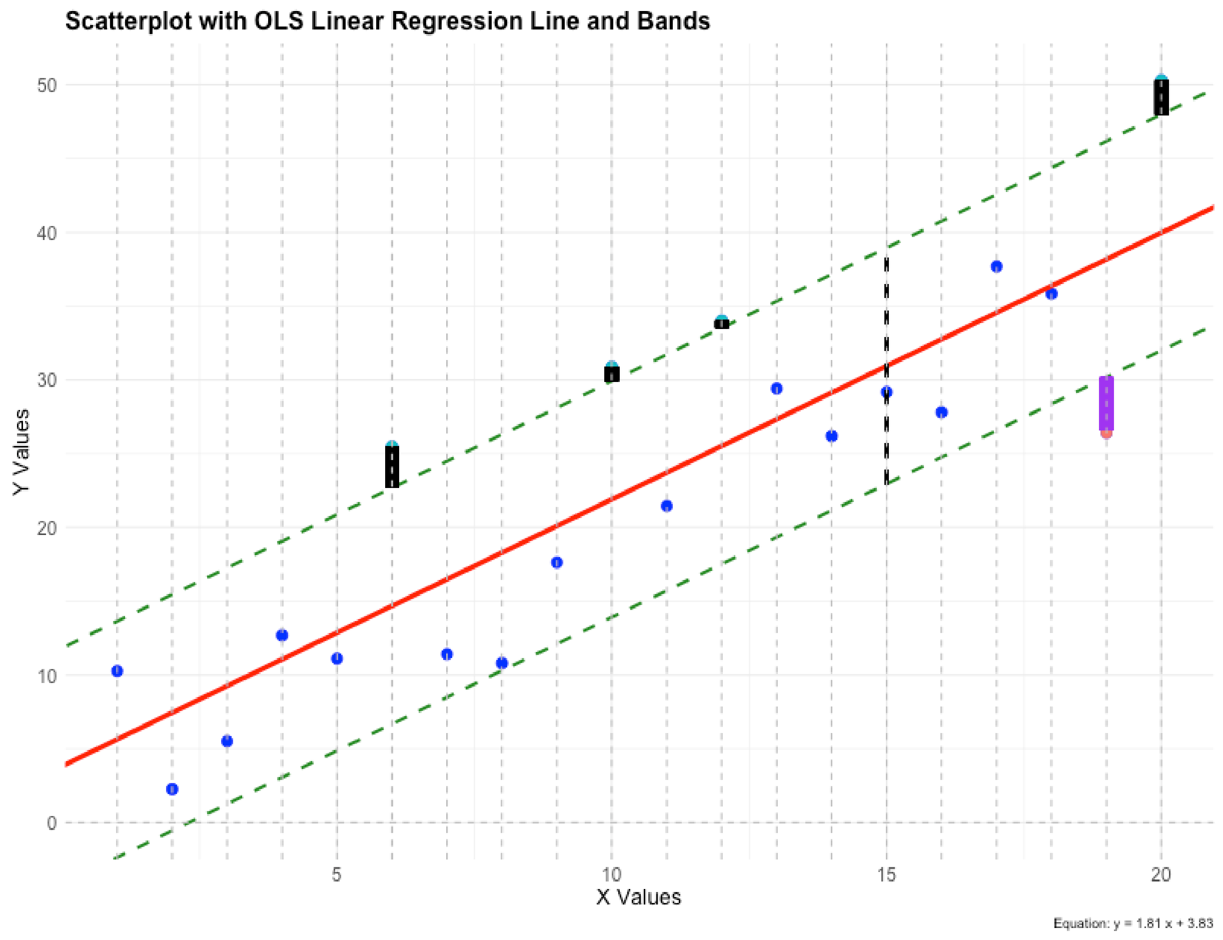
Support vector regression

Minimize:

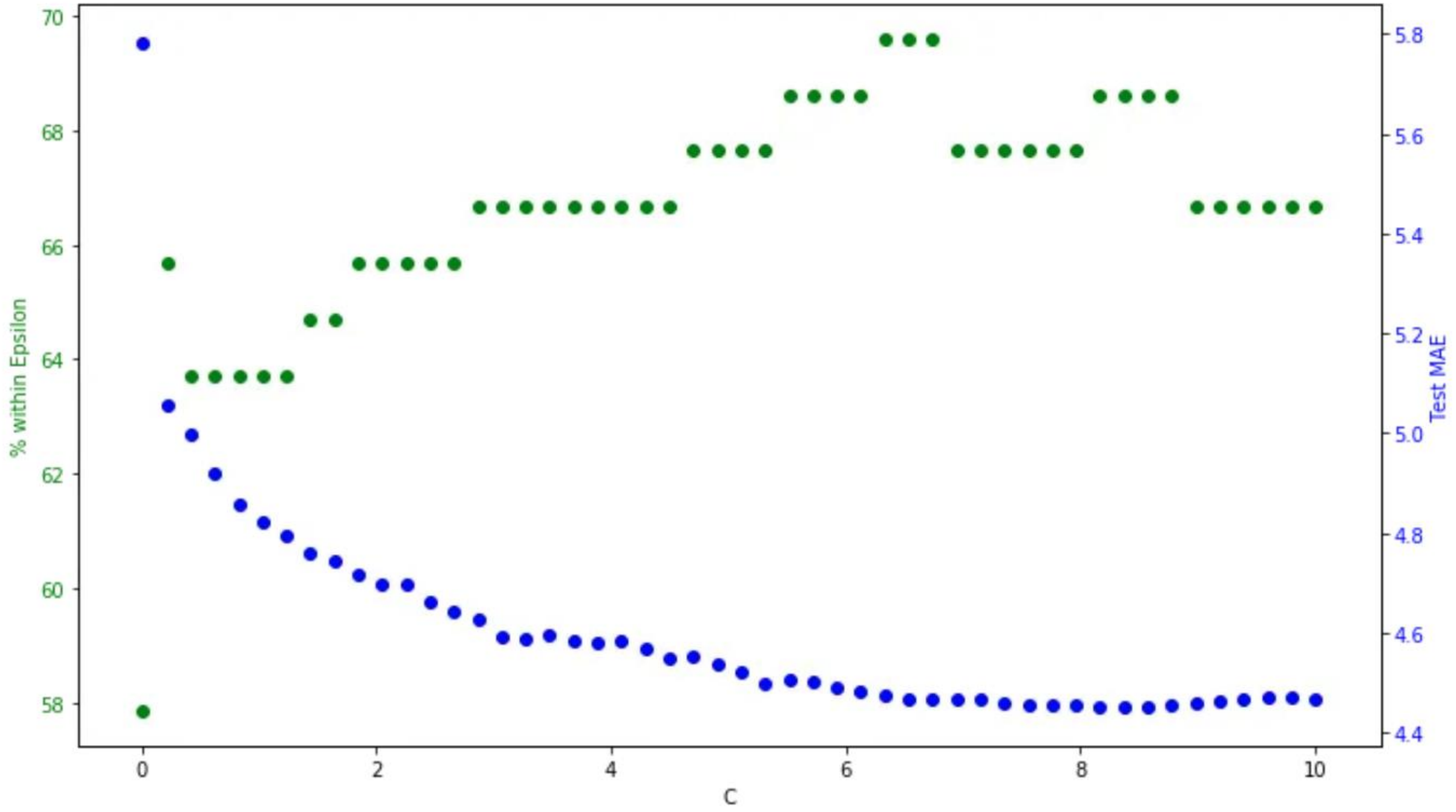
$$\frac{1}{2}||w||^2 + c\sum_{i=0}^{\ell}|\xi_i|$$

Constrained on the assumption:

$$|y_i - < w, x_i > \; - b| < \epsilon + \xi_i$$



Support vector regression



GridSearch for C

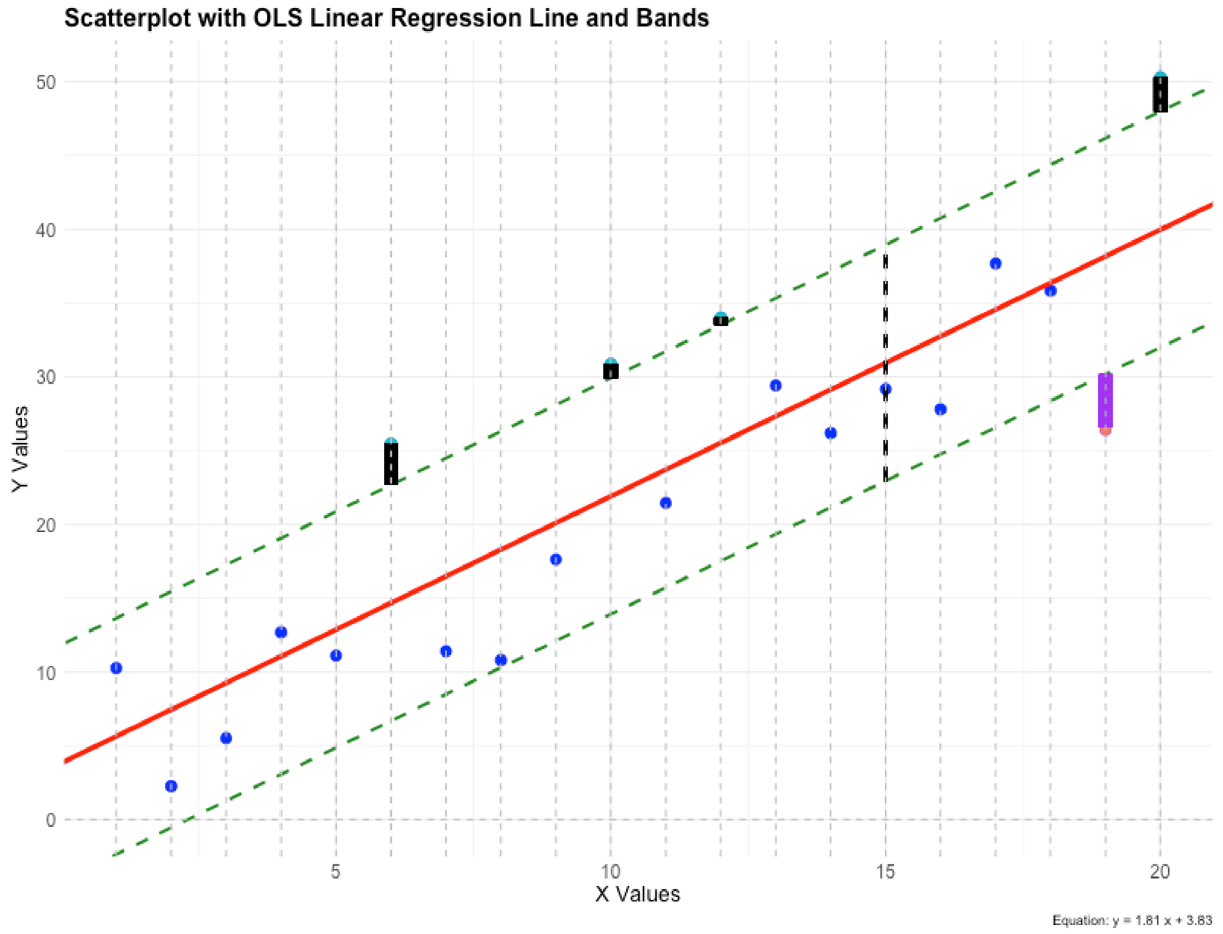
Support vector regression

Minimize:

$$\frac{1}{2}||w||^2 + c\sum_{i=0}^{\ell}|\xi_i|$$

Constrained on the assumption:

$$|y_i - < w, x_i > \; -b| < \epsilon + \xi_i$$



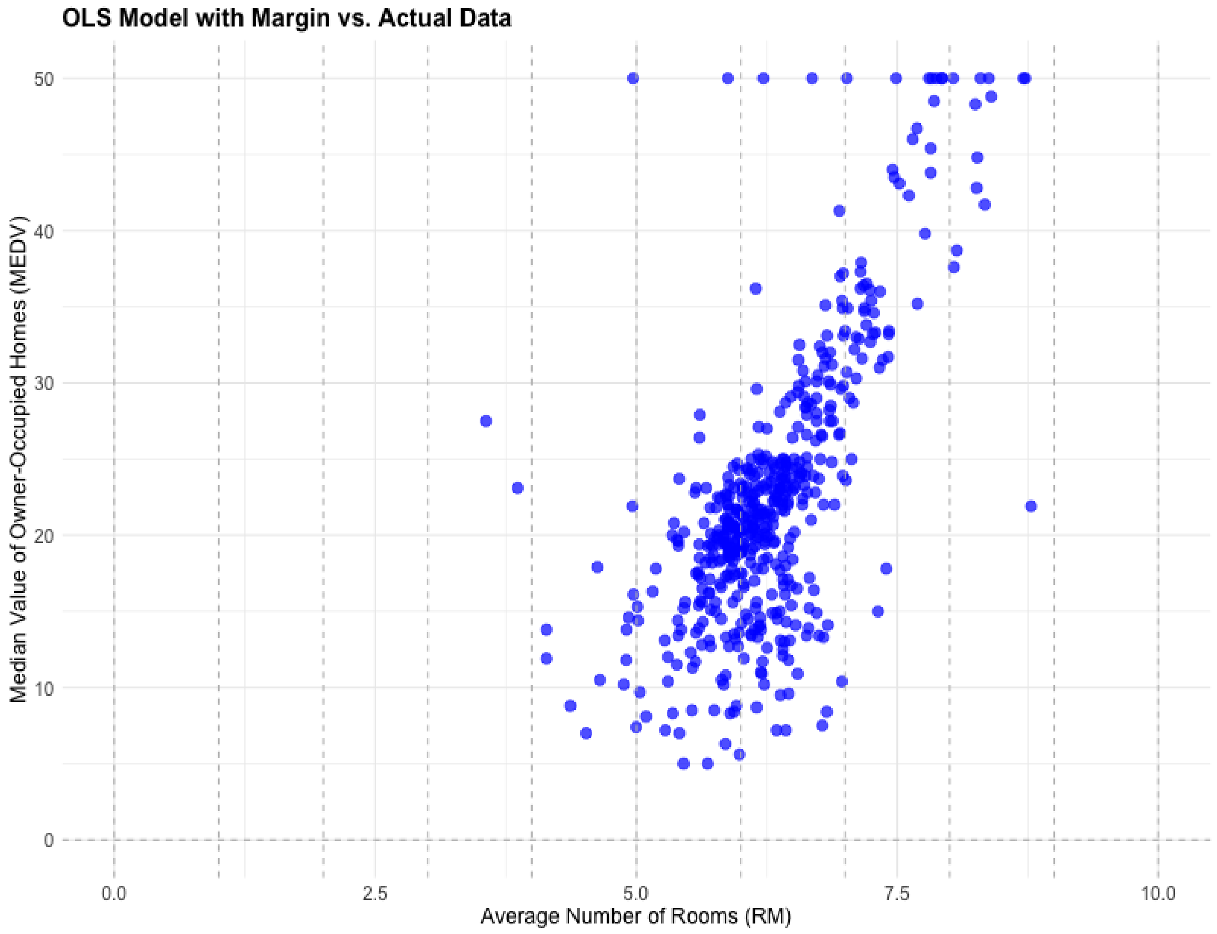
Support vector regression

Minimize:

$$\frac{1}{2}||w||^2 + c\sum_{i=0}^{\ell}|\xi_i|$$

Constrained on the assumption:

$$|y_i - < w, x_i > \; -b| < \epsilon + \xi_i$$



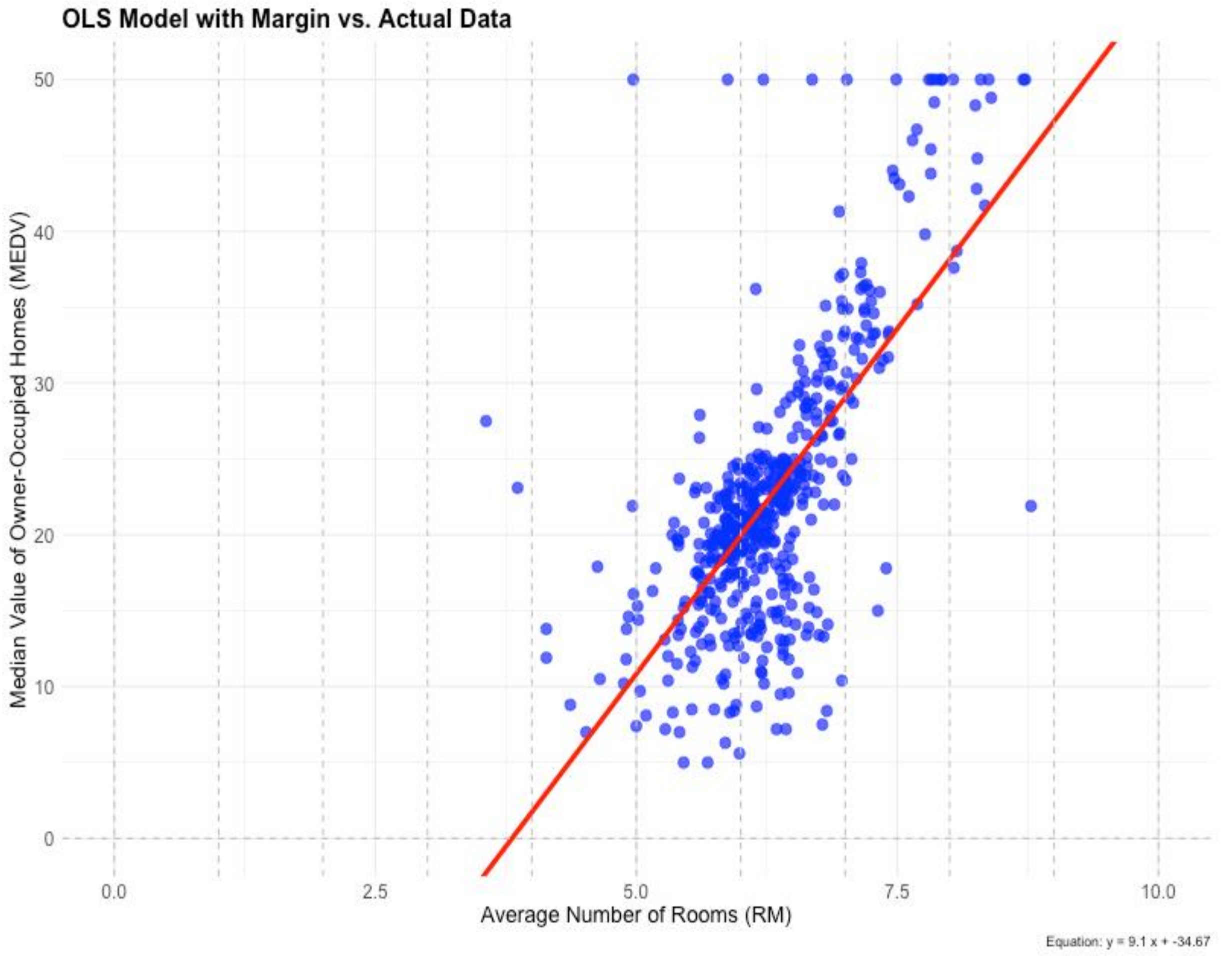
Support vector regression

Minimize:

$$\frac{1}{2}||w||^2 + c\sum_{i=0}^{\ell}|\xi_i|$$

Constrained on the assumption:

$$|y_i - < w, x_i > \; -b| < \epsilon + \xi_i$$



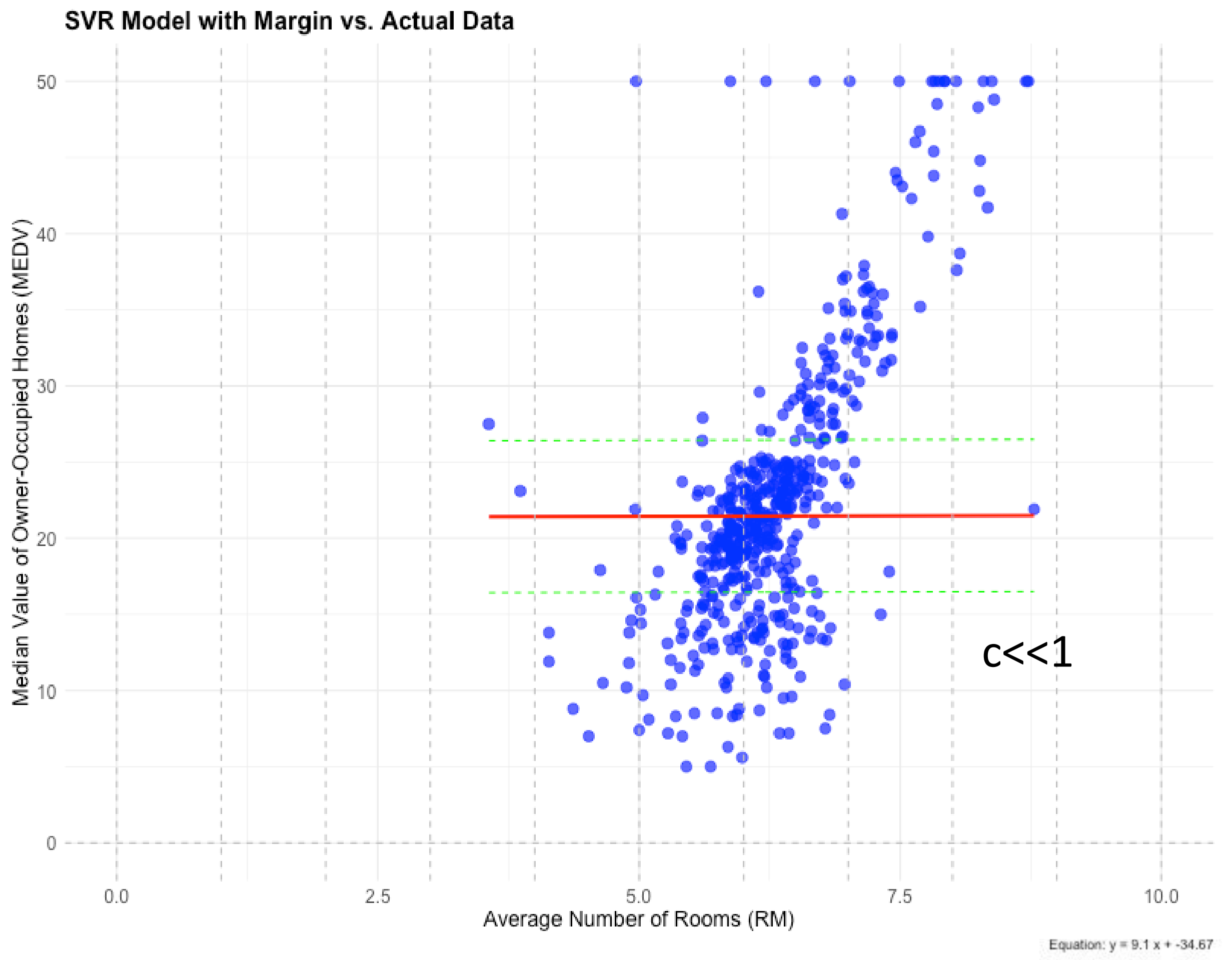
Support vector regression

Minimize:

$$\frac{1}{2}||w||^2 + c\sum_{i=0}^{\ell}|\xi_i|$$

Constrained on the assumption:

$$|y_i - < w, x_i > \ -b| < \epsilon + \xi_i$$



c<<1

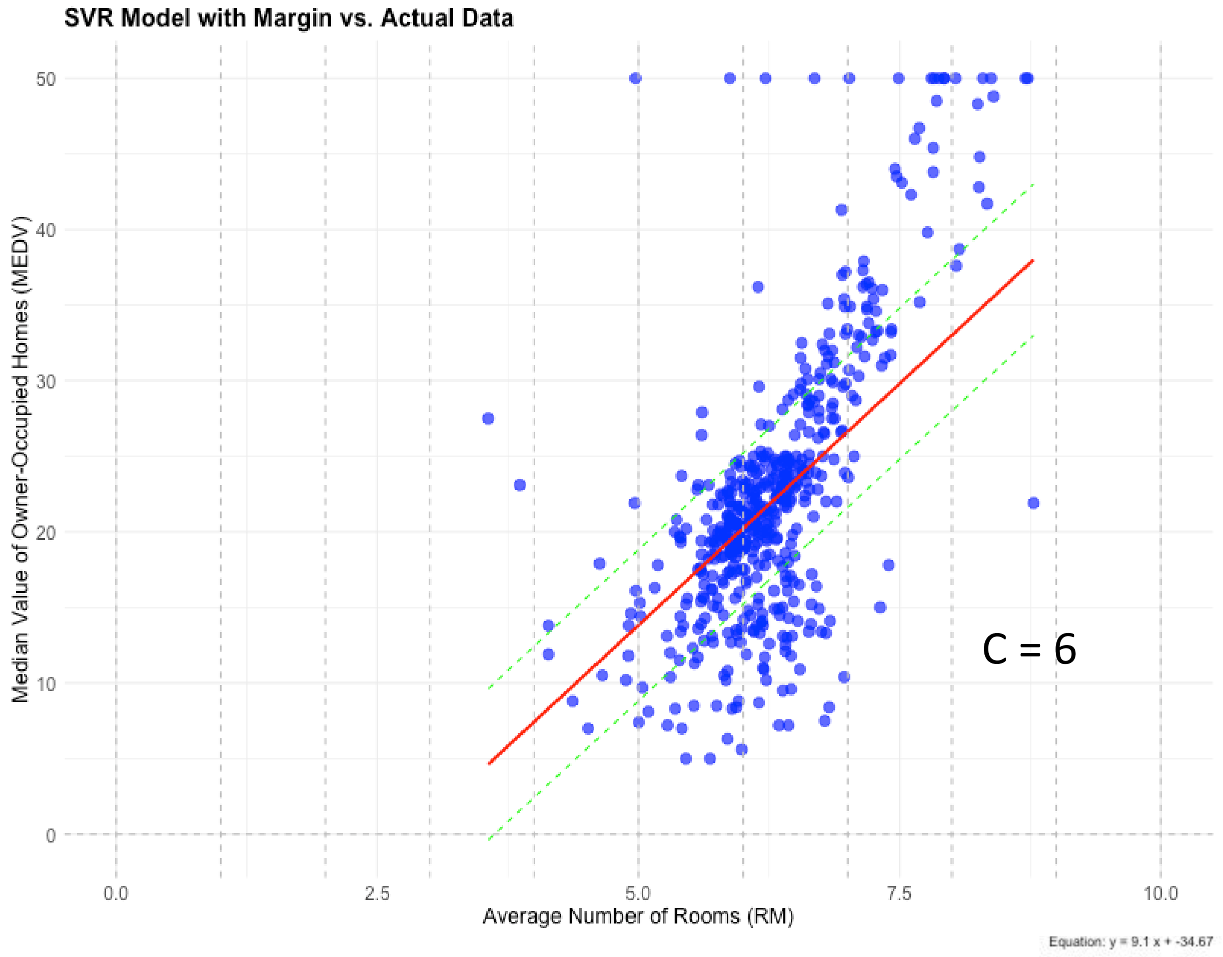
Support vector regression

Minimize:

$$\frac{1}{2}||w||^2 + c\sum_{i=0}^{\ell}|\xi_i|$$

Constrained on the assumption:

$$|y_i - < w, x_i > \ -b| < \epsilon + \xi_i$$



C = 6

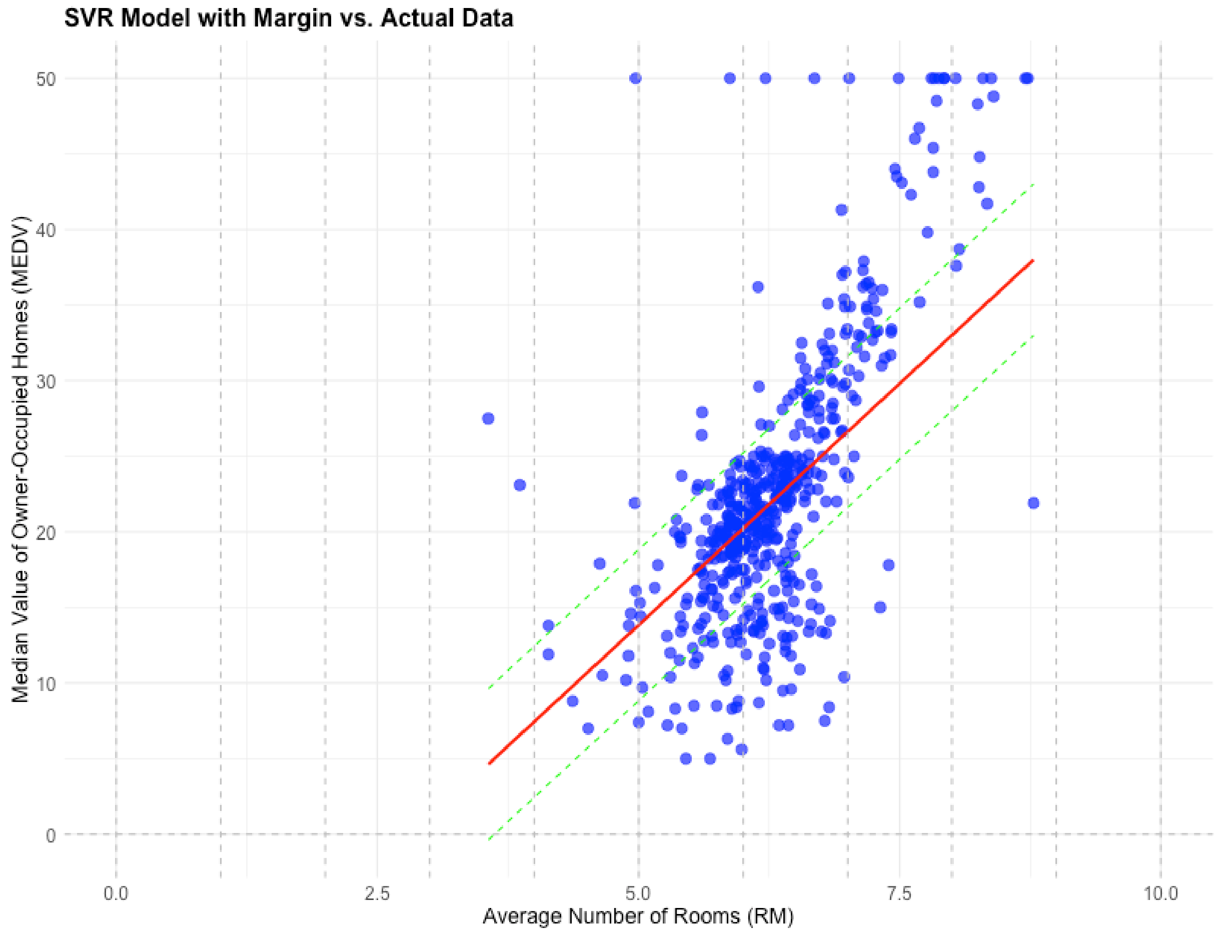
Support vector regression

Minimize:

$$\frac{1}{2}||w||^2 + c\sum_{i=0}^{\ell}|\xi_i|$$

Constrained on the assumption:

$$\left|y_i - < w_k, \phi(x_{i,k}) > \ -b\right| < \epsilon + \xi_i$$



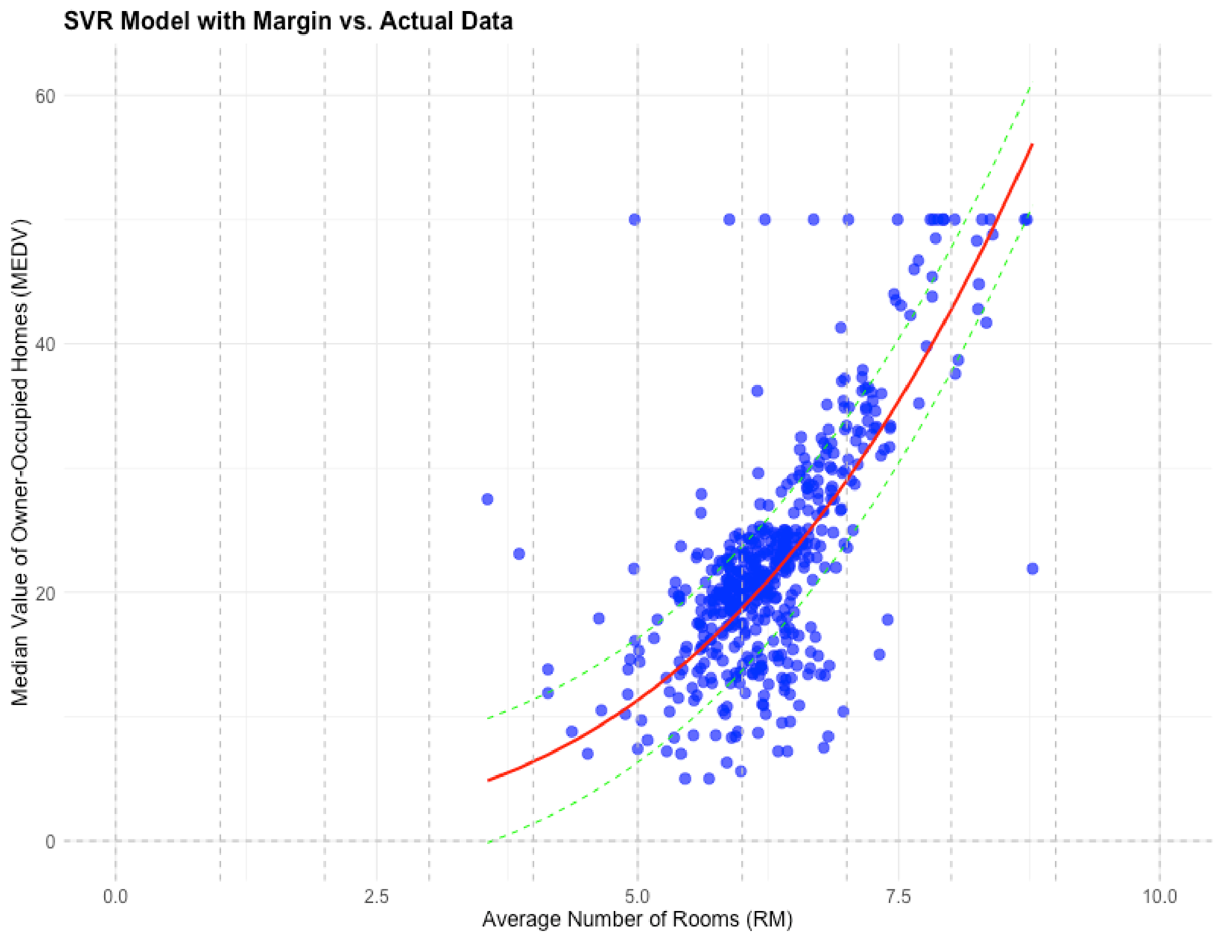
Support vector regression

Minimize:

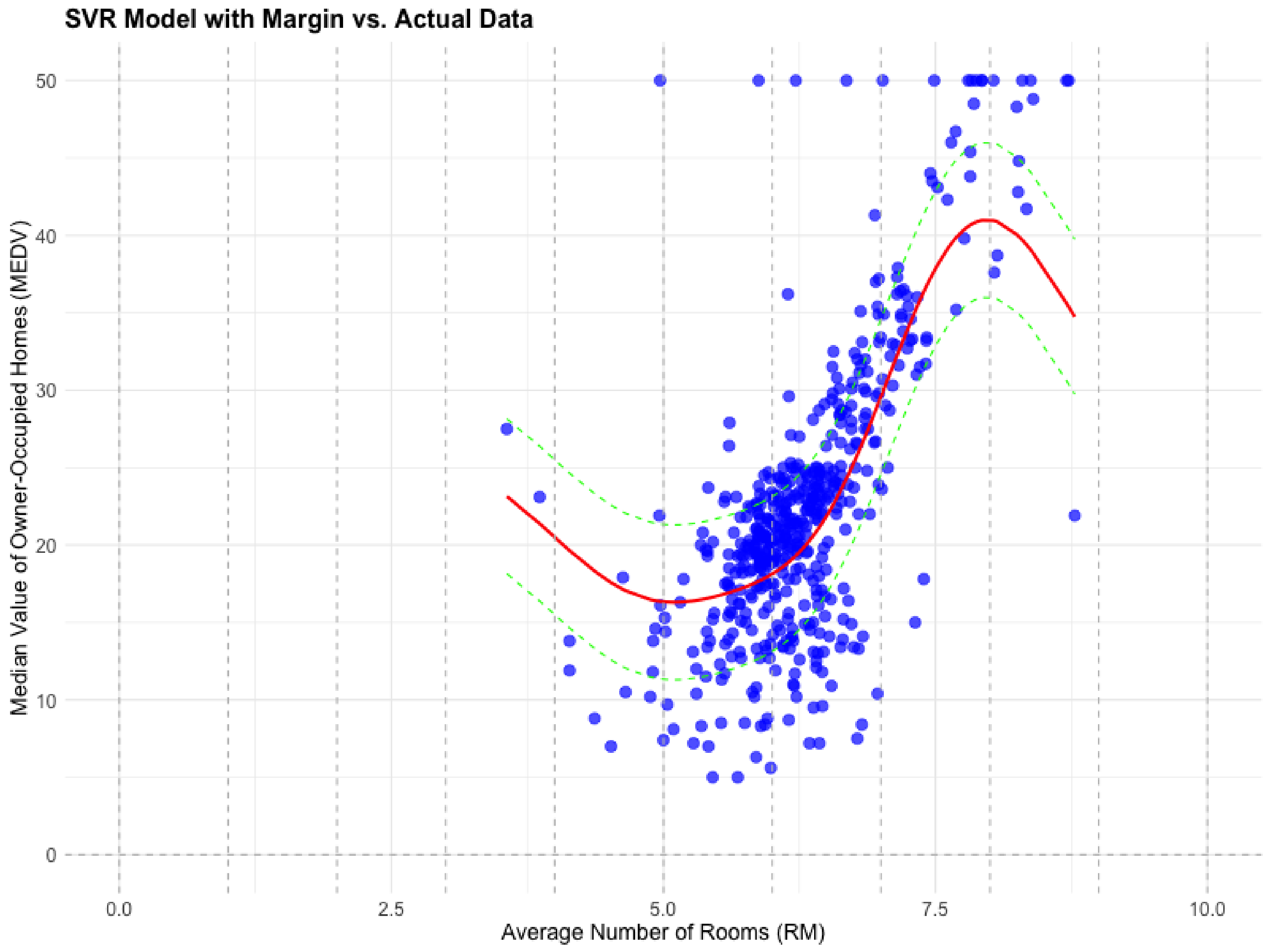
$$\frac{1}{2}||w||^2 + c\sum_{i=0}^{\ell}|\xi_i|$$

Constrained on the assumption:

$$\left|y_i - < w_k, \phi(x_{i,k}) > \; -b\right| < \epsilon + \xi_i$$



Support vector regression



THANK YOU