# DATA MINING 2
# Support Vector Machine

Riccardo Guidotti

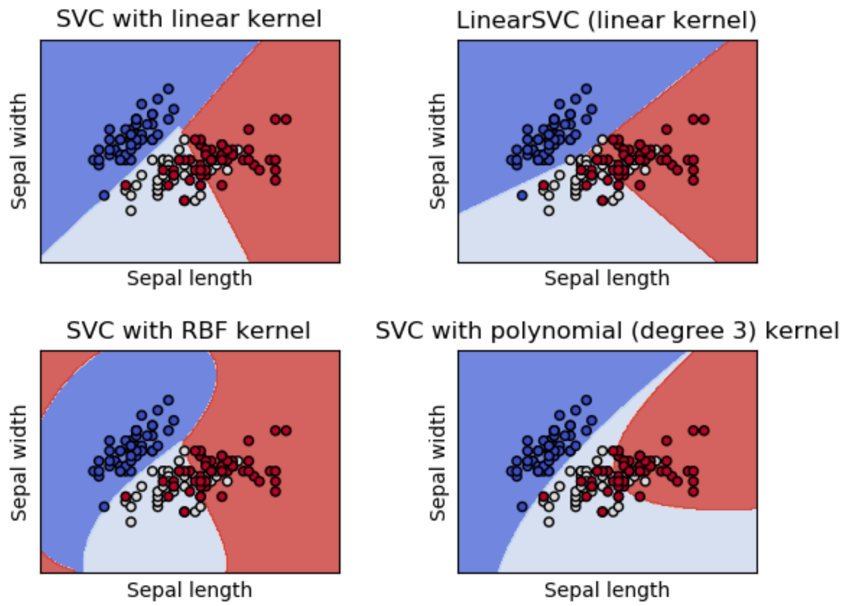
a.a. 2019/2020

UNIVERSITÀ DI PISA

# Support Vector Machine (SVM)

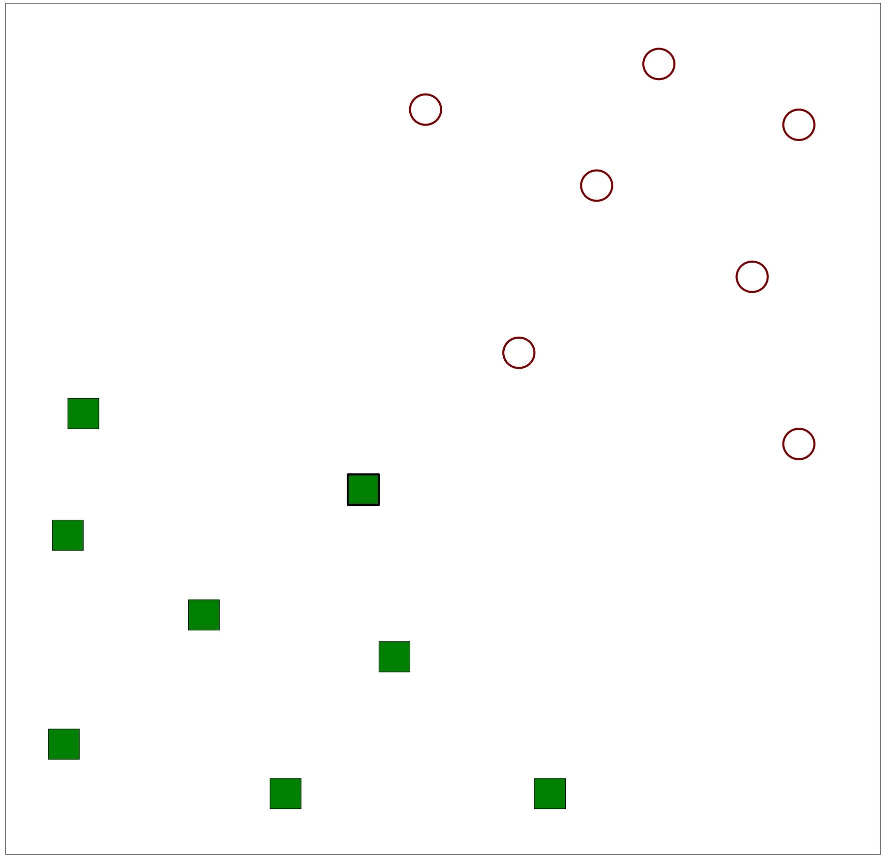
- SVM represents the decision boundary using a subset of the training examples, known as the **support vectors**.

- We illustrate the basic idea behind SVM by introducing the concept of **maximal margin hyperplane** and explain the rationale of choosing such a hyperplane.



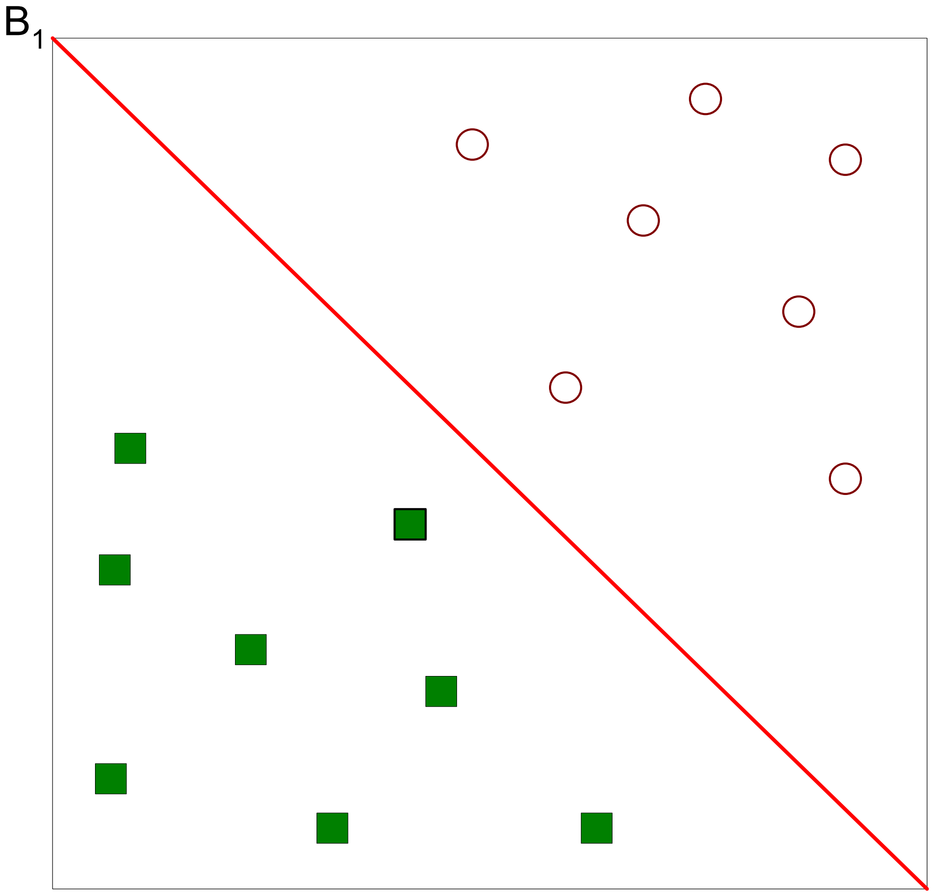
# Maximum Margin Hyperplanes

- Find a linear hyperplane (decision boundary) that separates the data.



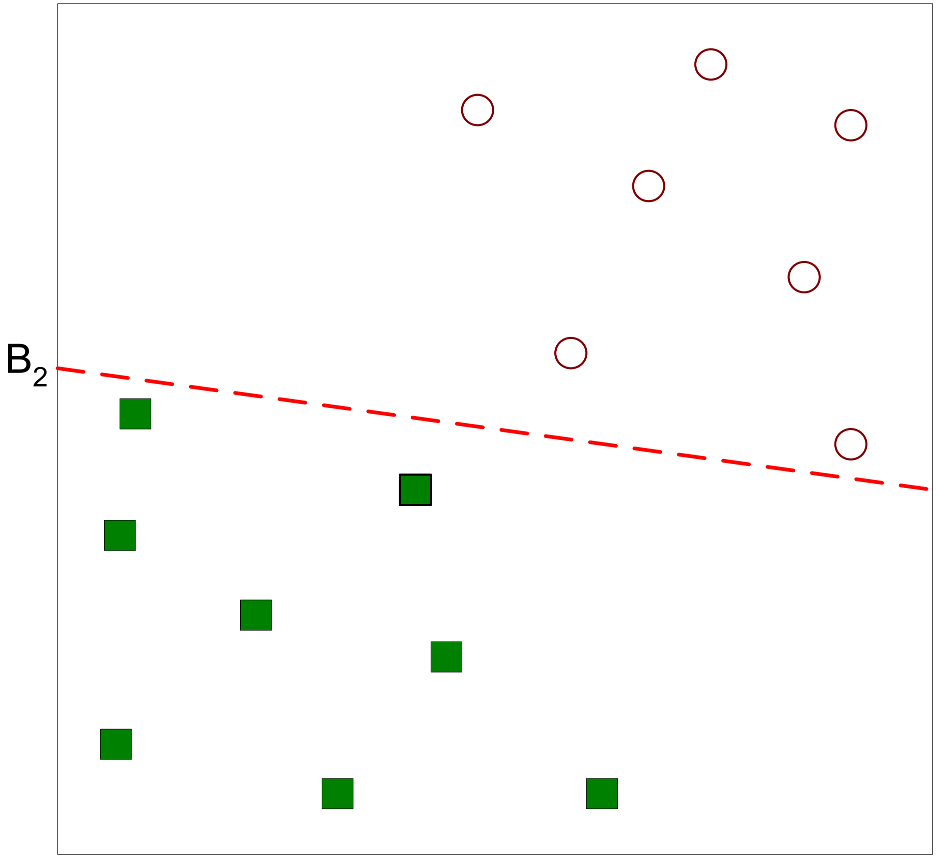
# Maximum Margin Hyperplanes

- One possible solution.



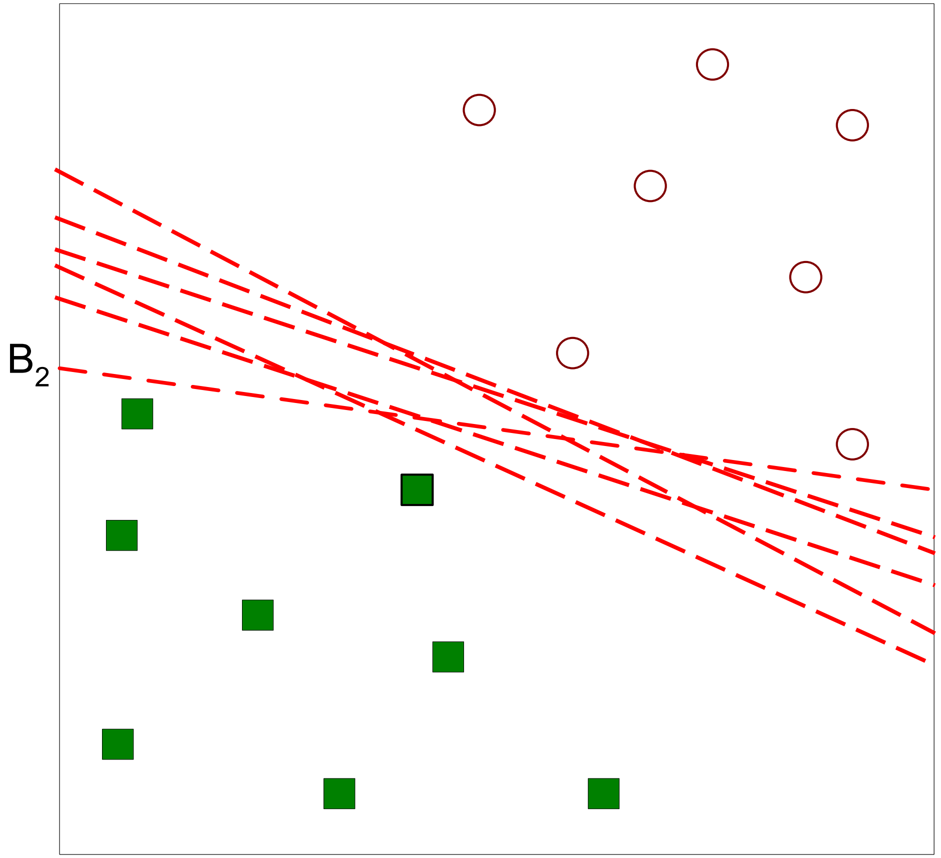
# Maximum Margin Hyperplanes

- Another possible solution.



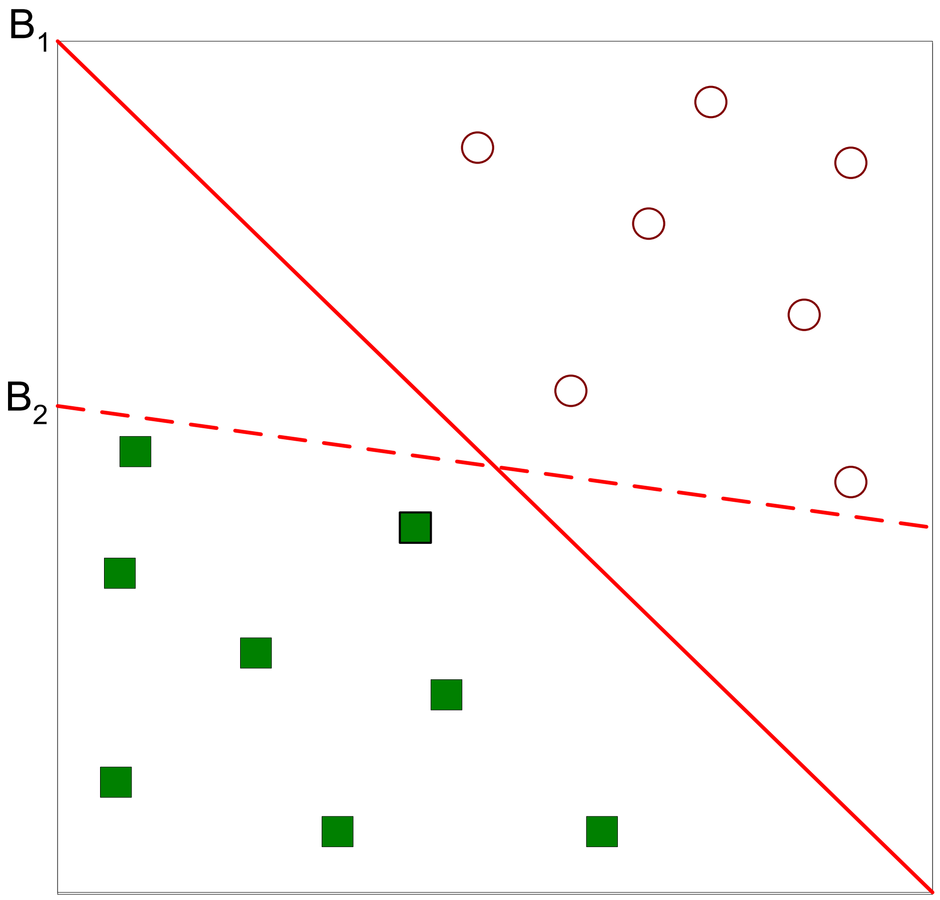
# Maximum Margin Hyperplanes

- Other possible solutions.



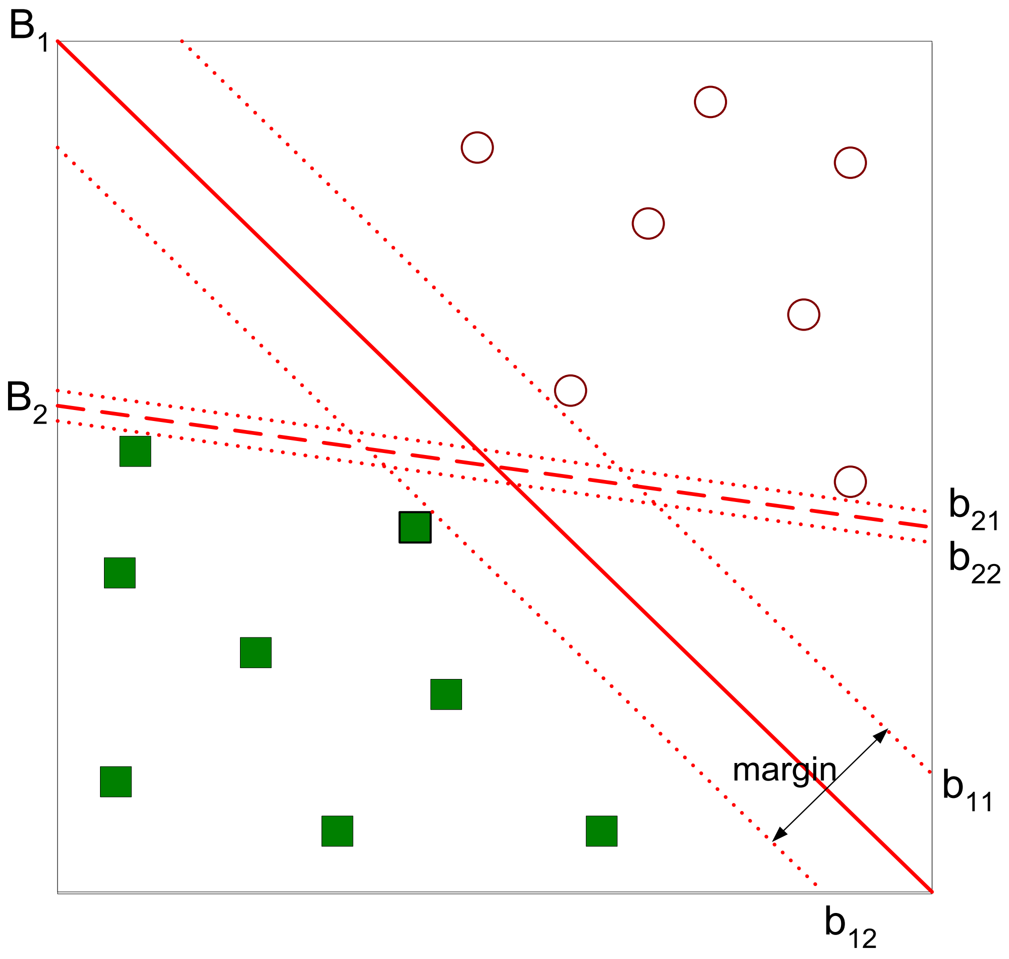
# Maximum Margin Hyperplanes

- Let's focus on $B_1$ and $B_2$.
- Which one is better?
- How do you define better?



# Maximum Margin Hyperplanes

- The best solution is the hyperplane that **maximizes** the **margin**.
- Thus, $B_1$ is better than $B_2$.



# Linear SVM: Separable Case

$$\vec{w} \bullet \vec{x} + b = +1$$

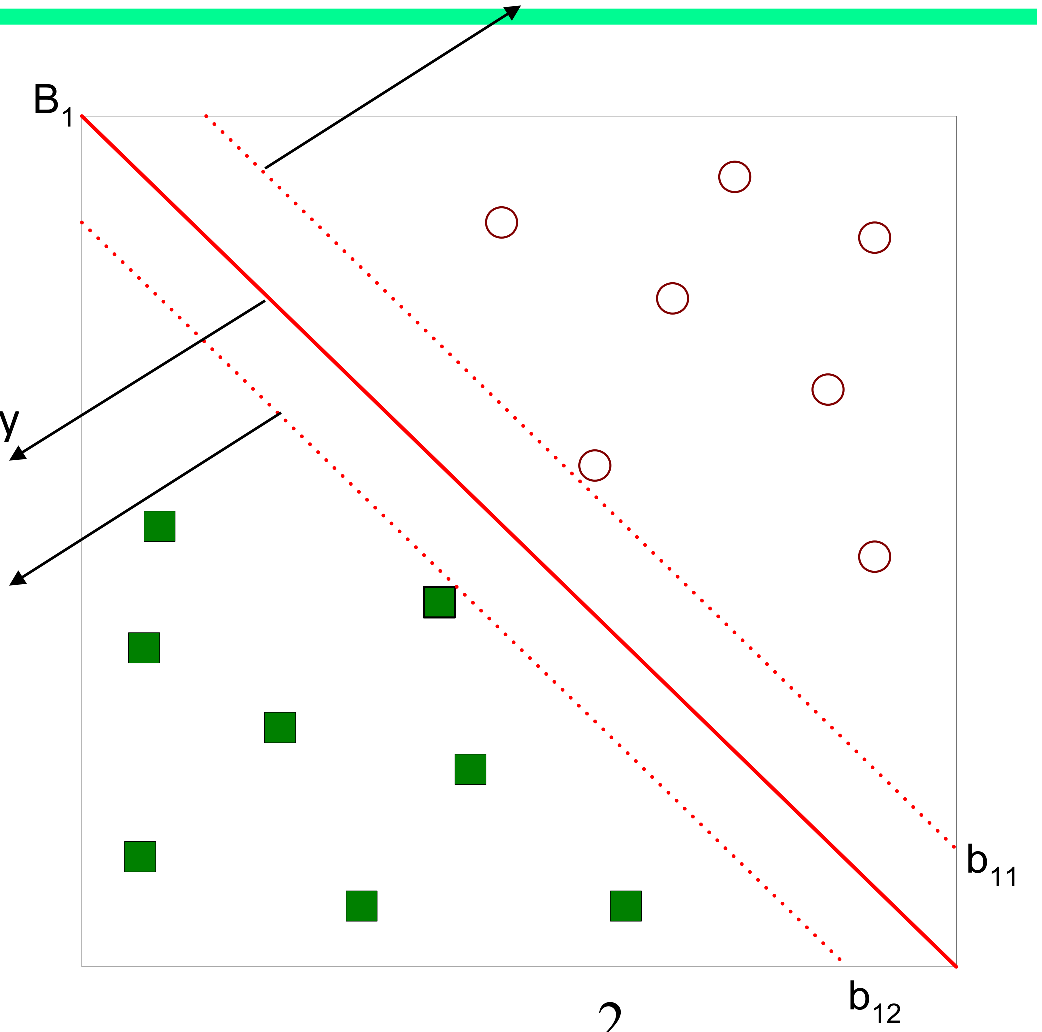
- A linear SVM is a classifier that searches for a hyperplane with the largest margin (a.k.a. maximal margin classifier).

decision boundary

$$\vec{w} \bullet \vec{x} + b = 0$$

- *w* and *b* are parameters.

$$\vec{w} \bullet \vec{x} + b = -1$$




- Given *w* and *b* the classifiers work as

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|}$$

Example calculus dot product

w = [.3  .2]   x = [1 2]   b = -2

w · x + b = .3*1 + .2*2 +(-2) = -1.3

# Learning a Linear SVM

- Learning the model is equivalent to determining *w* and *b.*
- How to find *w* and *b?*
- Objective is to maximize the margin.
- Which is equivalent to minimize
- Subject to to the following constraints
- This is a constrained optimization problem that can be solved using the *Lagrange* multiplier method.
- Introduce Lagrange multiplier $\lambda$

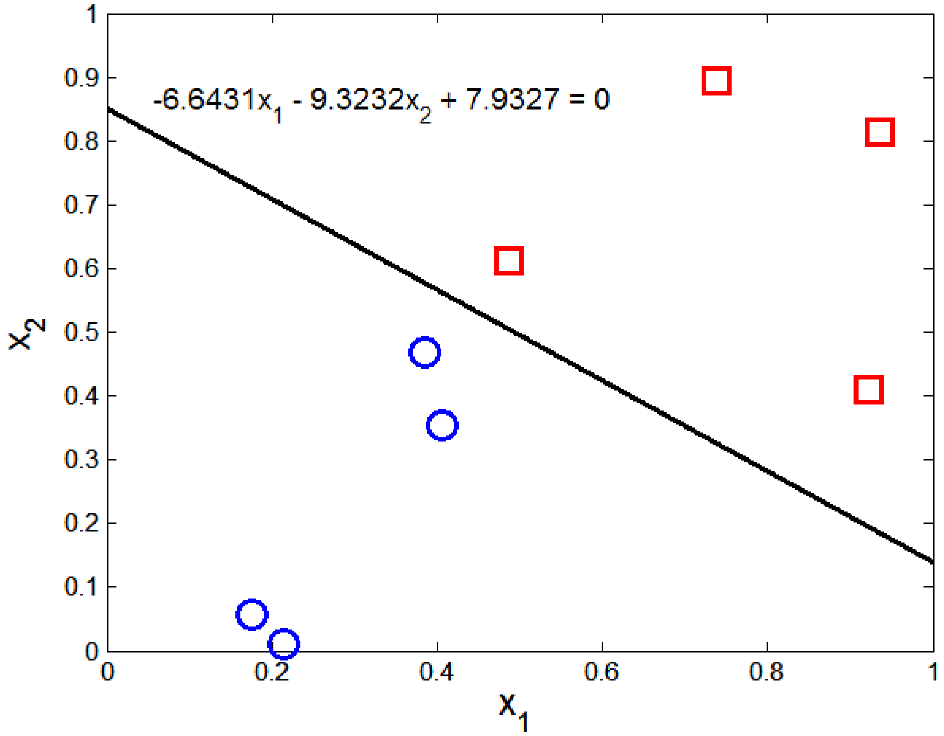
$$L(\vec{w}) = \frac{\| \vec{w} \|^2}{2}$$

$$y_i = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \leq -1 \end{cases}$$

=

$$y_i(\mathbf{w} \bullet \mathbf{x}_i + b) \geq 1, \quad i = 1,2,\ldots,N$$

# Example of Linear SVM

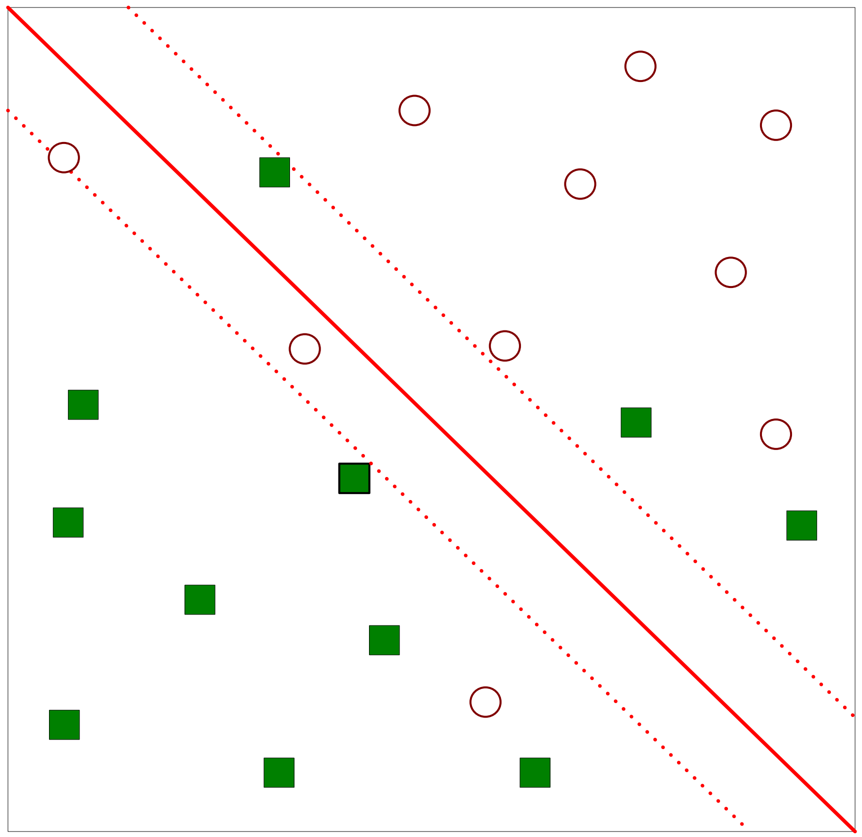



Support vectors

| x1 | x2 | y | λ |
|---|---|---|---|
| 0.3858 | 0.4687 | 1 | 65.5261 |
| 0.4871 | 0.611 | -1 | 65.5261 |
| 0.9218 | 0.4103 | -1 | 0 |
| 0.7382 | 0.8936 | -1 | 0 |
| 0.1763 | 0.0579 | 1 | 0 |
| 0.4057 | 0.3529 | 1 | 0 |
| 0.9355 | 0.8132 | -1 | 0 |
| 0.2146 | 0.0099 | 1 | 0 |

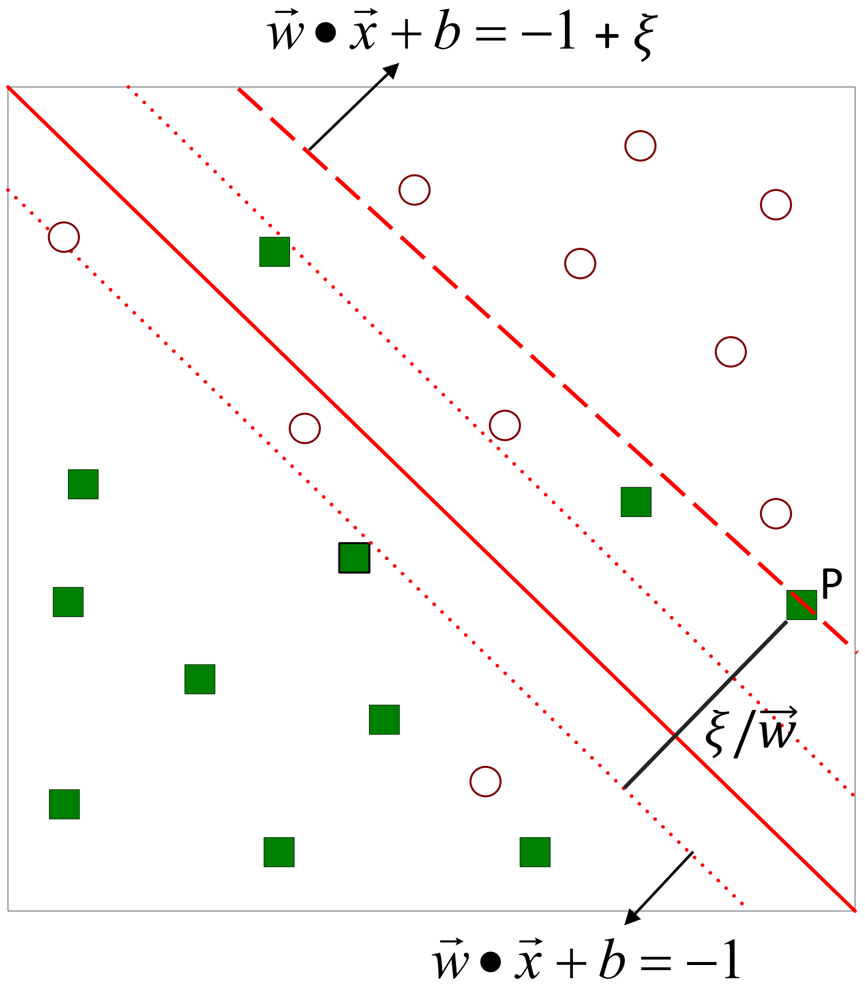
# Linear SVM: Nonseparable Case

- What if the problem is not linearly separable?



# Slack Variables

- The inequality constraints must be relaxed to accommodate the nonlinearly separable data.
- This is done introducing slack variables $\xi$ into the constrains of the optimization problem.
- $\xi$ provides an estimate of the error of the decision boundary on the misclassified training examples.



# Learning a Nonseparable Linear SVM

- Objective is to minimize

$$L(w) = \frac{\| \vec{w} \|^2}{2} + C\left( \sum_{i=1}^{N} \xi_i^k \right)$$

- Subject to to the constraints

$$y_i = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \geq 1 - \xi_{\mathrm{i}} \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \leq -1 + \xi_{\mathrm{i}} \end{cases}$$

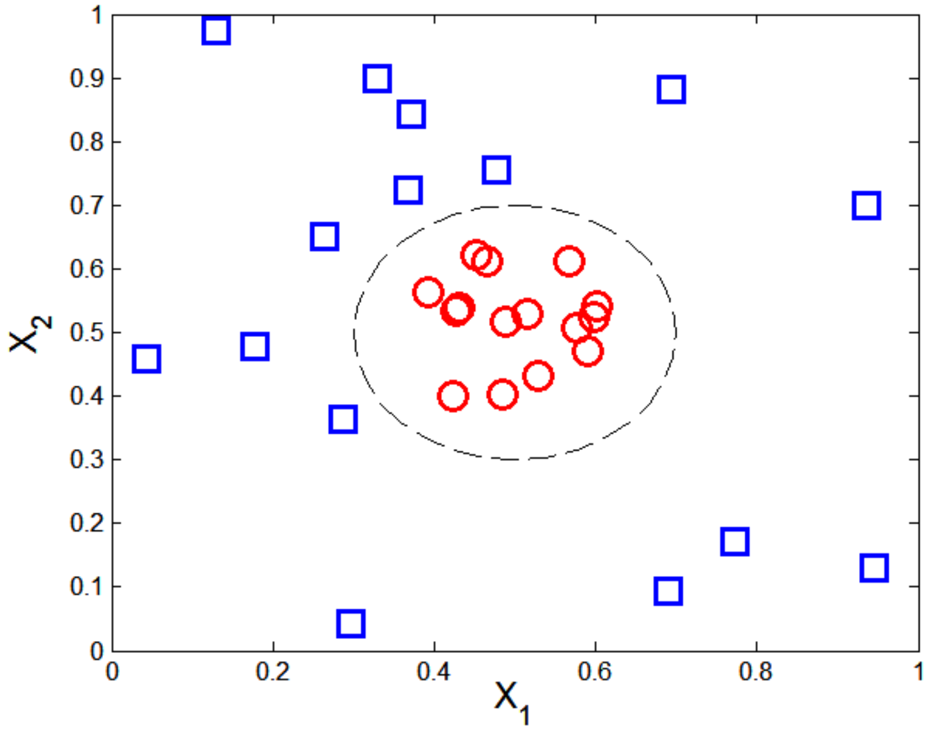
- where *C* and *k* are user-specified parameters representing the penalty of misclassifying the training instances
- Lagrangian multipliers are constrained to $0 \leq \lambda \leq C$.

https://scikit-learn.org/stable/auto_examples/svm/plot_linearsvc_support_vectors.html#sphx-glr-auto-examples-svm-plot-linearsvc-support-vectors-py

# Nonlinear SVM

- What if the decision boundary is not linear?

$$y(x_1, x_2) = \begin{cases} 1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\ -1 & \text{otherwise} \end{cases}$$



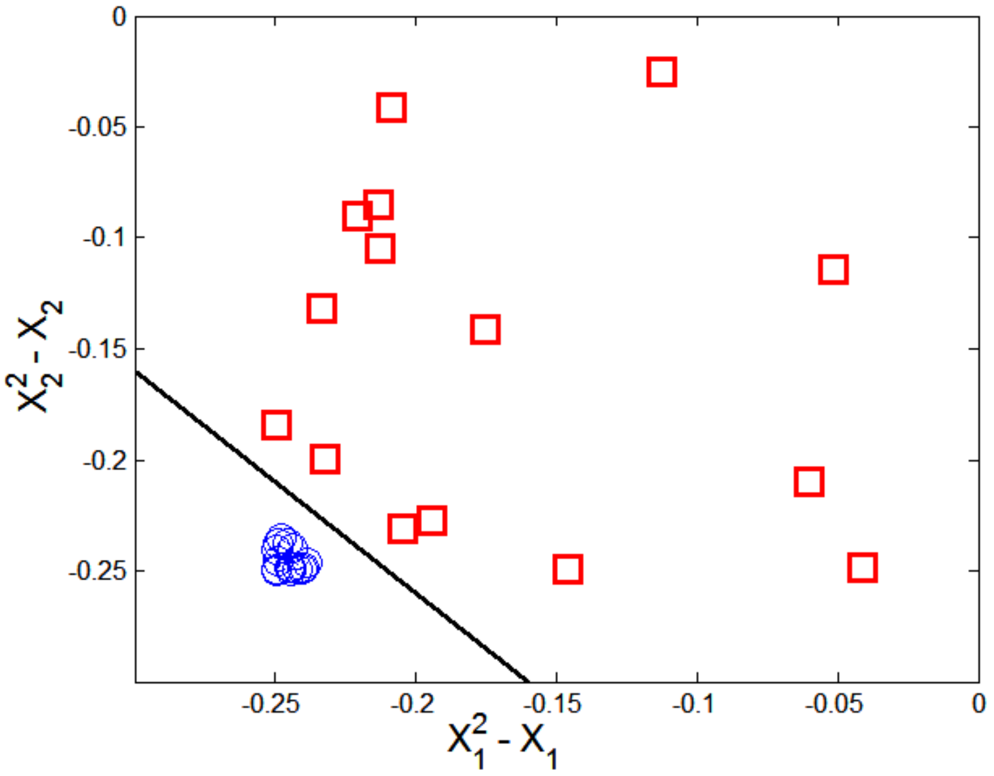
# Nonlinear SVM

- The trick is to transform the data from its original space $x$ into a new space $\Phi(x)$ so that a linear decision boundary can be used.

$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46.$$

$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

$$w_4 x_1^2 + w_3 x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$




- Decision boundary $\vec{w} \bullet \Phi(\vec{x}) + b = 0$

# Learning a Nonlinear SVM

- Optimization problem

$$\min_{w} \frac{\|\mathbf{w}\|^2}{2}$$

$$\text{subject to} \quad y_i(\boldsymbol{w} \cdot \Phi(\boldsymbol{x}_i) + b) \geq 1, \ \forall\{(\boldsymbol{x}_i, y_i)\}$$

- Which leads to the same set of equations but involve $\Phi(x)$ instead of $x$.

$$f(\mathbf{z}) = sign(\mathbf{w} \cdot \Phi(\mathbf{z}) + b) = sign(\sum_{i=1}^{n} \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z}) + b).$$

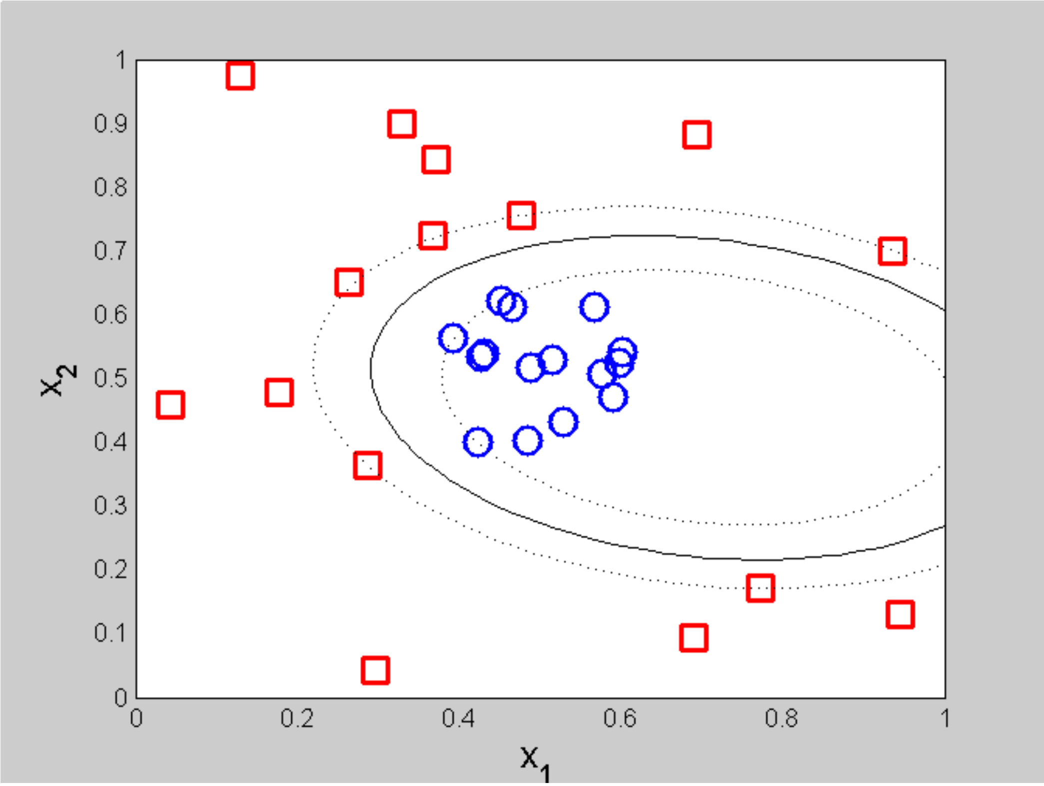
Issues:

- What type of mapping function $\Phi$ should be used?
- How to do the computation in high dimensional space?
  - Most computations involve dot product $\Phi(x) \cdot \Phi(x)$
  - Curse of dimensionality?

# Kernel Trick

- $\Phi(x) \cdot \Phi(x) = K(x_i, x_j)$
- $K(x_i, x_j)$ is a kernel function (expressed in terms of the coordinates in the original space)
- Examples:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$
$$K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/(2\sigma^2)}$$
$$K(\mathbf{x}, \mathbf{y}) = \tanh(k\mathbf{x} \cdot \mathbf{y} - \delta)$$




https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py
https://scikit-learn.org/stable/auto_examples/exercises/plot_iris_exercise.html#sphx-glr-auto-examples-exercises-plot-iris-exercise-py

# Kernel Trick

Advantages of using kernel:

- Don't have to know the mapping function $\Phi$.
- Computing dot product $\Phi(x) \cdot \Phi(y)$ in the original space avoids curse of dimensionality.

Not all functions can be kernels

- Must make sure there is a corresponding $\Phi$ in some high-dimensional space.
- *Mercer's theorem* (see textbook) that ensures that the kernel functions can always be expressed as the dot product in some high dimensional space.

# Characteristics of SVM

- Since the learning problem is formulated as a convex optimization problem, efficient algorithms are available to find the **global** minima of the objective function (many of the other methods use greedy approaches and find **locally** optimal solutions).
- Overfitting is addressed by maximizing the margin of the decision boundary, but the user still needs to provide the type of kernel function and cost function.
- Difficult to handle missing values.
- Robust to noise.
- High computational complexity for building the model.

# References

- Support Vector Machine (SVM). Chapter 5.5. Introduction to Data Mining.