Lecture 5

Supervised Algorithms: KNN and SVM

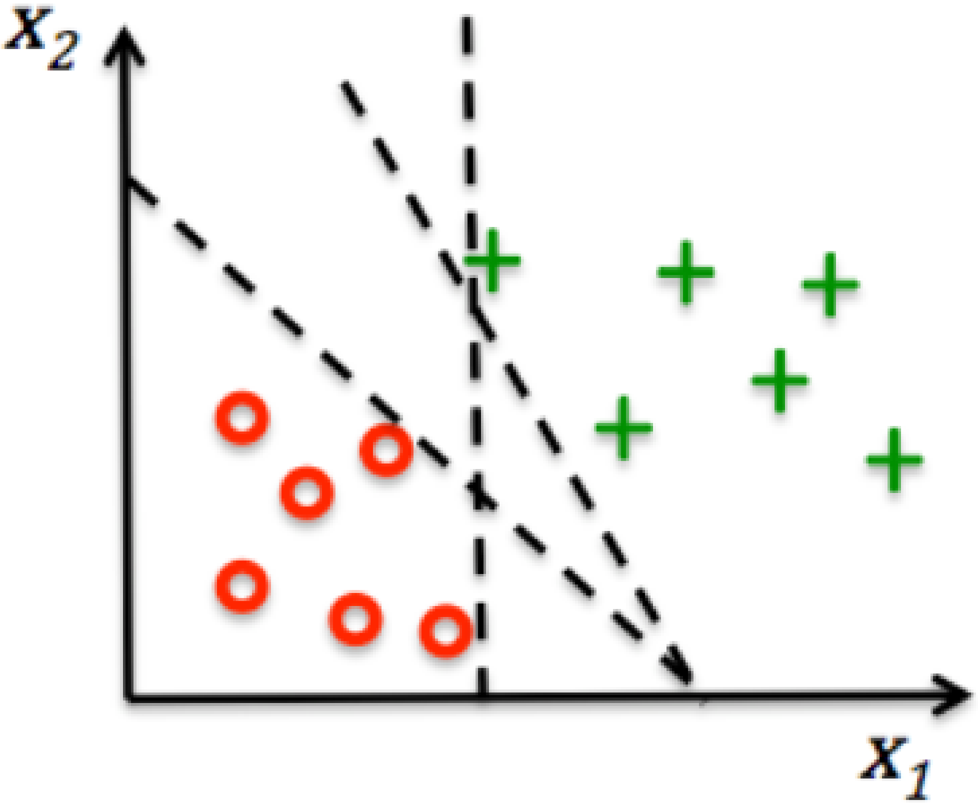
By: Nazerke Sultanova

Support vector machines

- In SVMs, the optimization objective is to maximize the margin
- The margin is defined as the distance between the separating

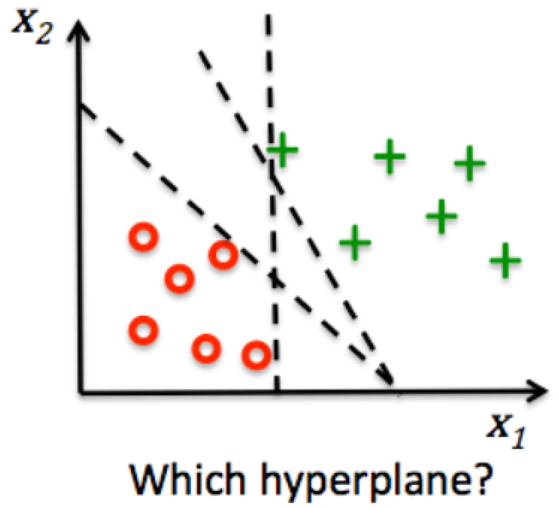
hyperlane and the training samples that are closest to this hyperplane (support vectors)

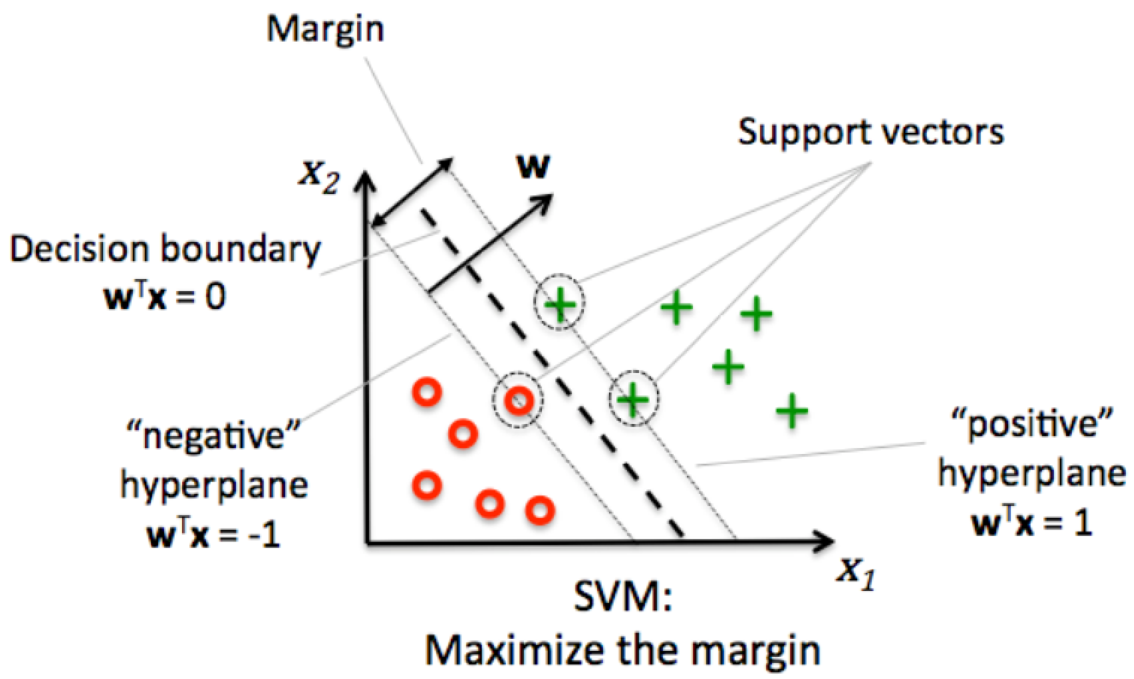
- Intuitively, the larger the margin, the lower generalization error
- Models with small margin prone to overfitting



Which hyperplane?

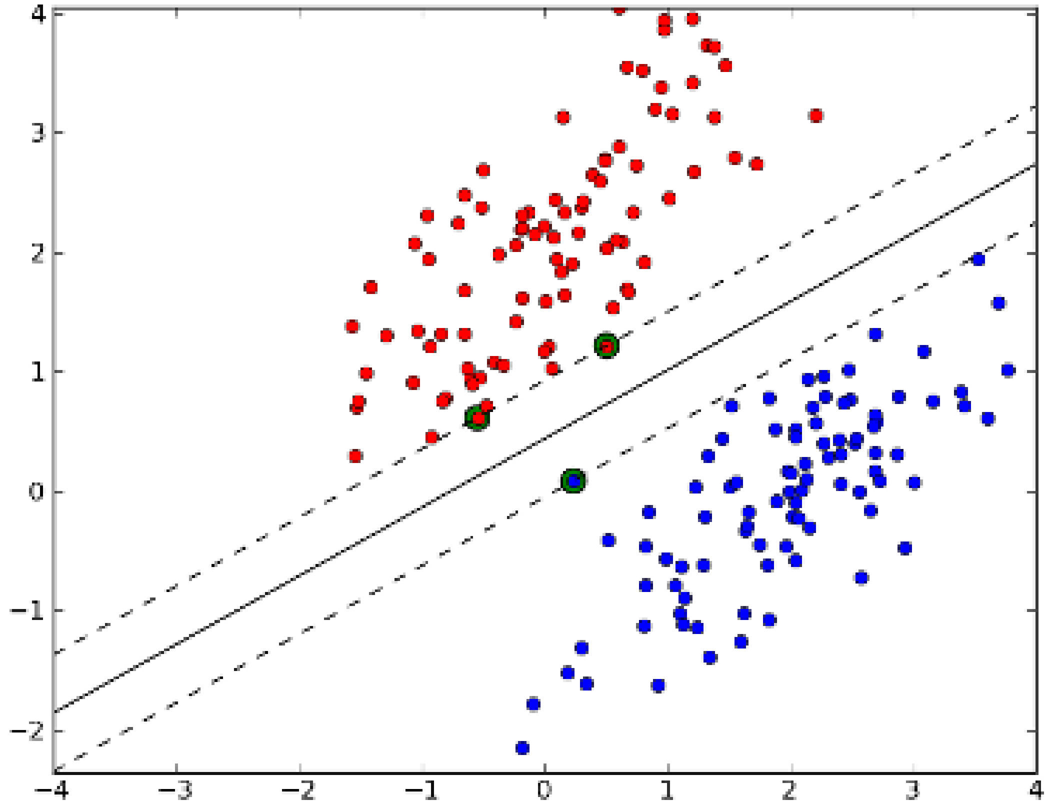
Maximum margin classification





Support Vectors:

The points with the smallest margins are exactly the ones closest to the decision boundary;



Mathematical intuition

Positive and negative hyperplanes that are parallel to the decision boundary, which can be expressed as follows:

$$w_0 + \boldsymbol{w}^T \boldsymbol{x_{pos}} = 1$$
$$w_0 + \boldsymbol{w}^T \boldsymbol{x_{neg}} = -1$$

Distance between these two planes, i.e. the margin:

$$\frac{2}{||\boldsymbol{w}||}$$

where the length of the **w** is defined as follows:

$$||w|| = \sqrt{\sum_{j=1}^{m} w_j^2}$$

Constraint optimization problem

Minimize:

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

Subject to constraints that the samples are classified correctly:

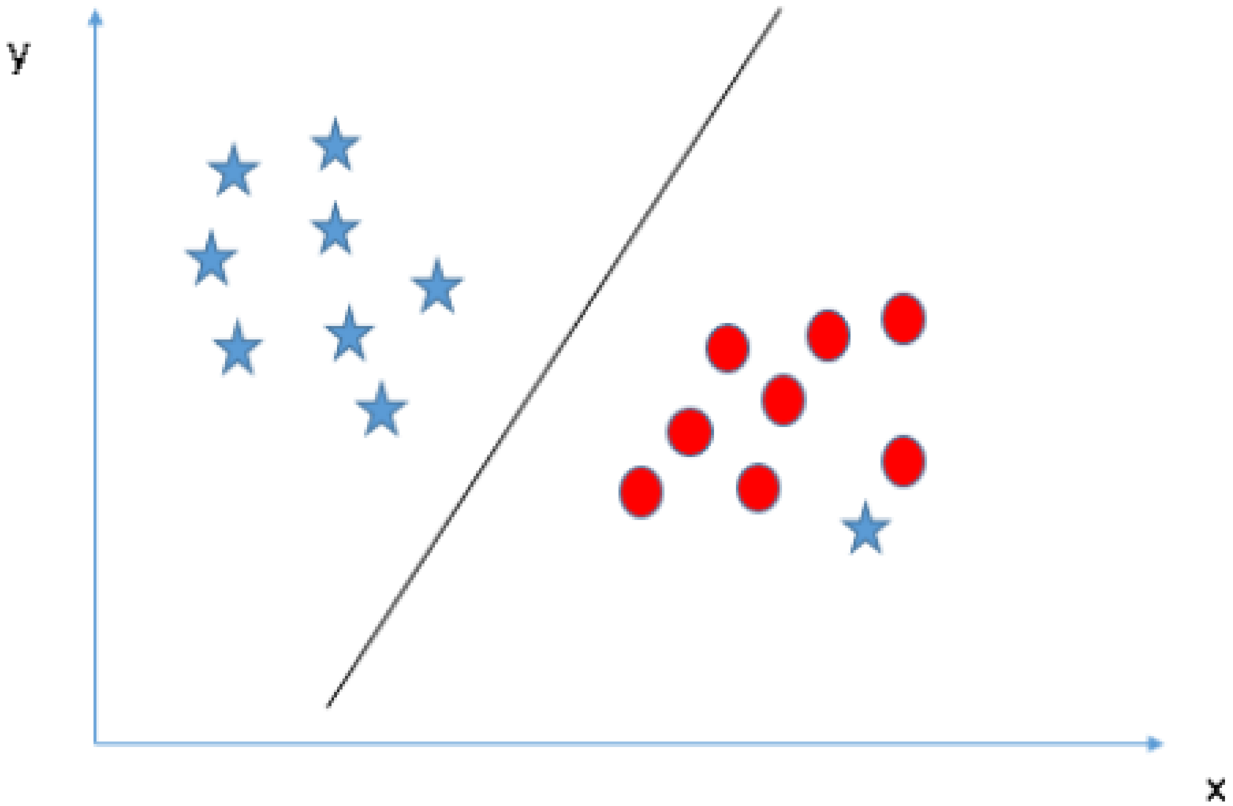
$$w_0 + \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{\boldsymbol{i}} \geq 1 \ \ if \ \ y^i = 1$$

$$w_0 + \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{\boldsymbol{i}} < -1 \ \ if \ \ y^i = -1$$

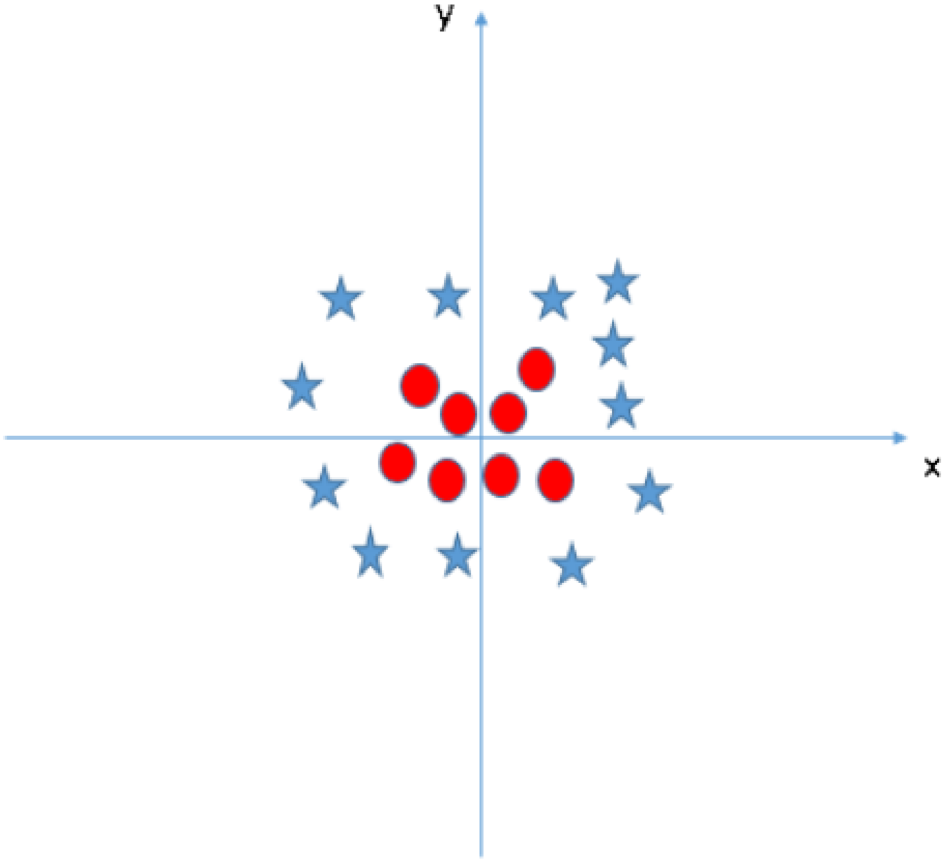
These equations say that all negative and positive samples should fall respectively on one side of the negative and positive hyperplanes. This can be written more compactly:

$$y^i\left(w_0 + \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^i\right) \geq 1 \ \forall_i$$

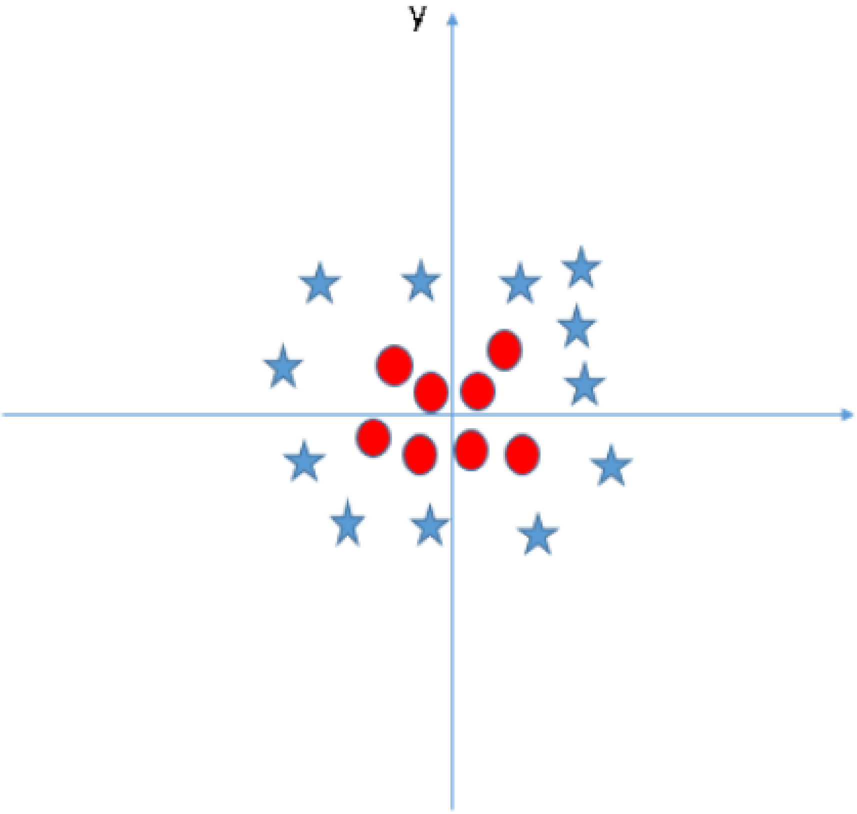
SVM - tolerant to outliers

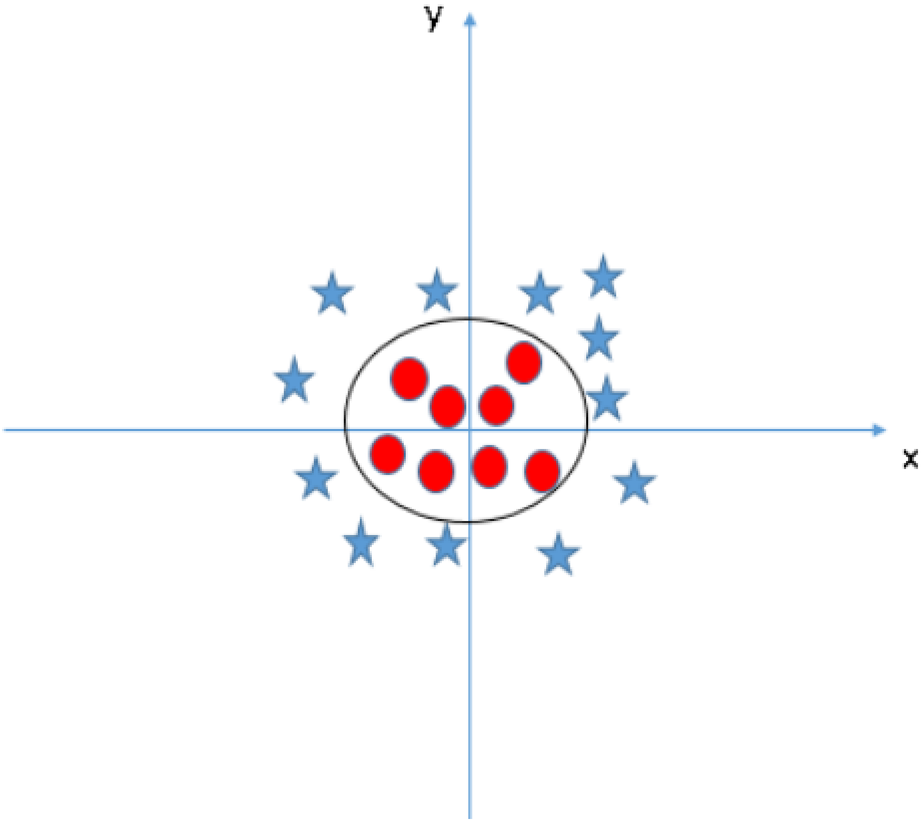


Is it linearly separable?



Separable



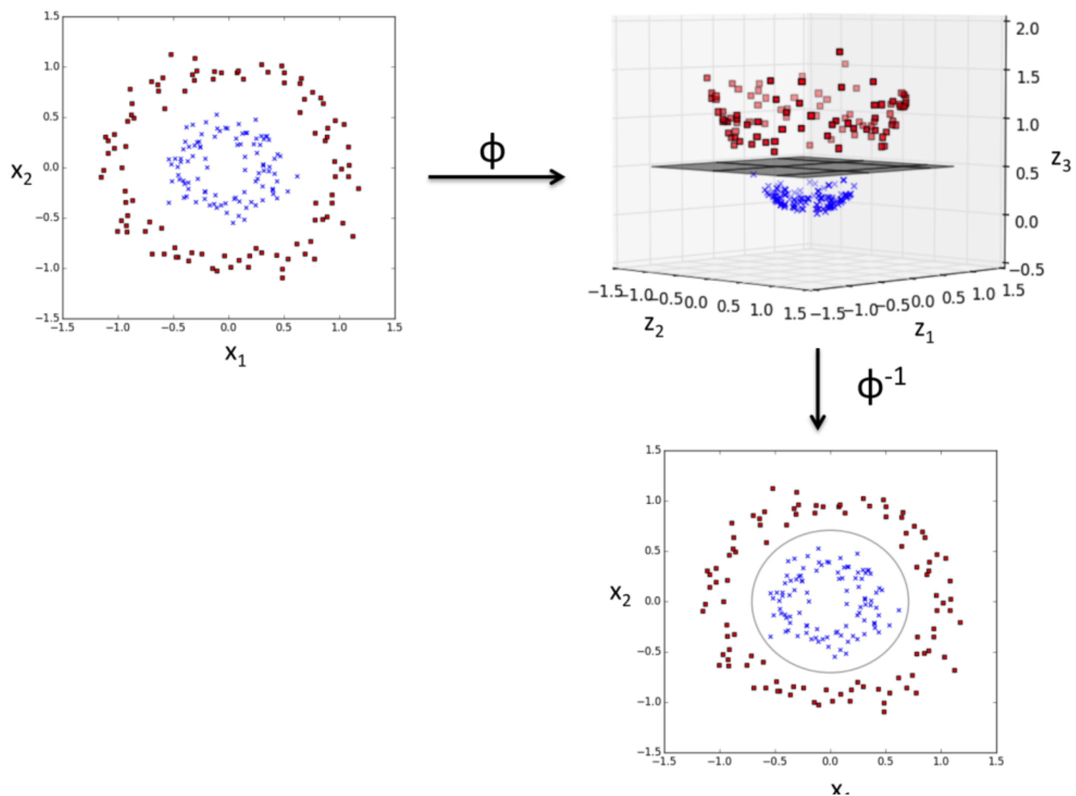


SVM can solve it!!!

- Easily! It solves this problem by introducing additional feature. Here, we will add a new feature $z=x^2+y^2$.

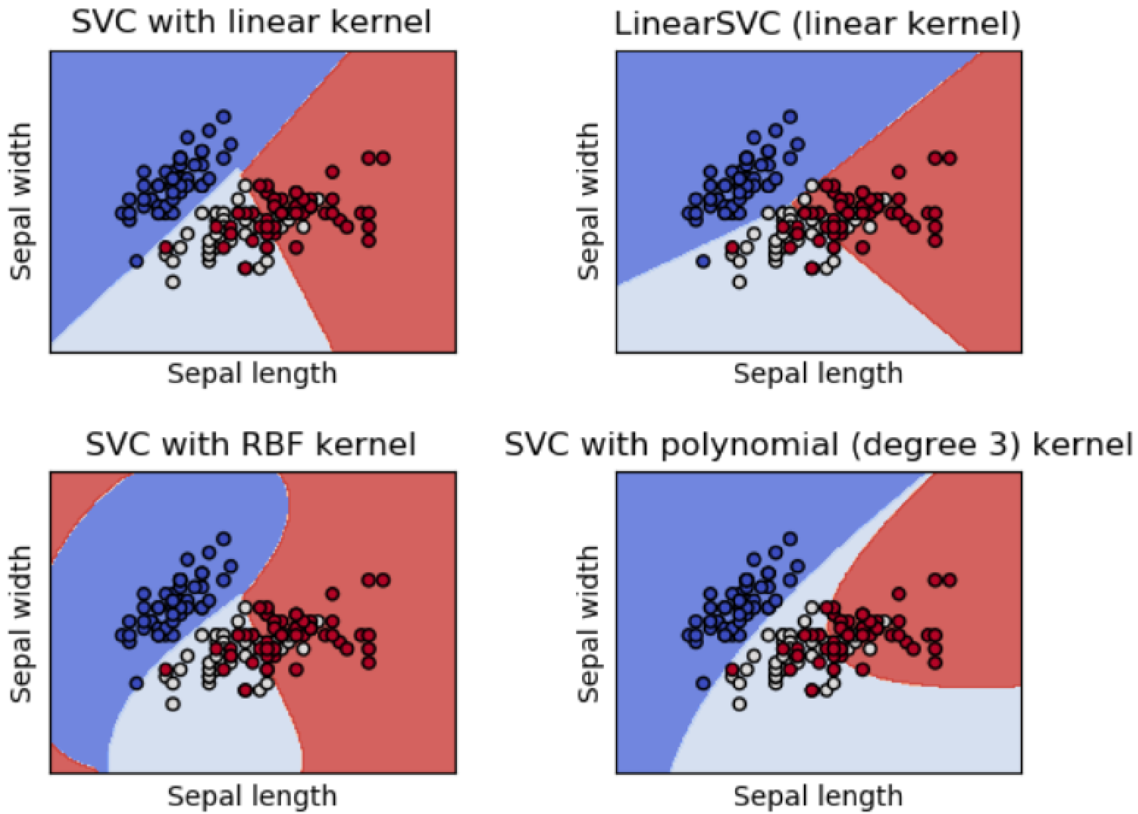
SVM - Kernel Trick

Turn no-separable classes to separable



SVM Kernels:

`SVC`, `NuSVC` and `LinearSVC` are classes capable of performing multi-class classification on a dataset.



Extend SVM to non-linearly separable cases

- Need to relax the linear constraints
- To ensure convergence in presence of misclassifications
- Introduce slack variables ξ

$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{\boldsymbol{i}} \geq 1 - \xi^i \quad if \;\; y^i = 1$$

$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{\boldsymbol{i}} < -1 + \xi^i \;\; if \;\; y^i = -1$$

New objective to be minimized:

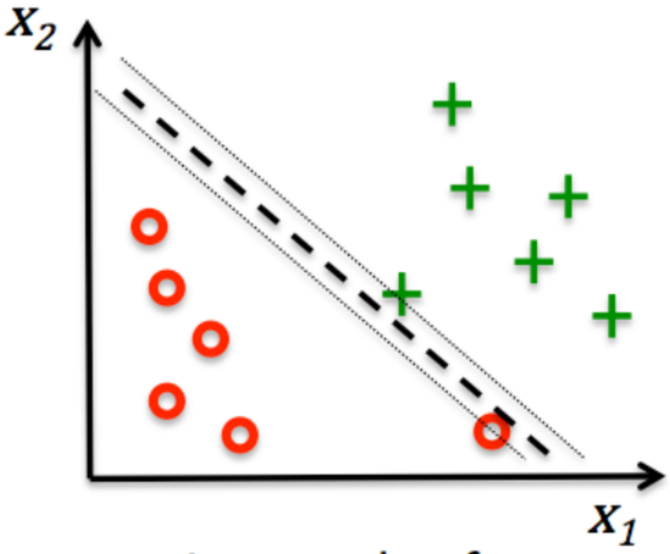
$$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\left(\sum_i \xi^i\right)$$

Regularization of SVM

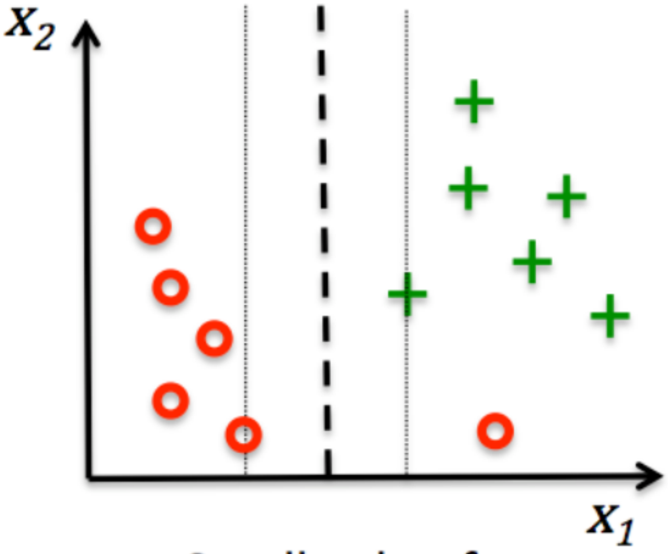
$$\frac{1}{2}\|w\|^2 + C\left(\sum_i \xi^i\right)$$

- Large values of C - large error penalties
- Small values of C - less strict about misclassifications
- Parameter C controls width of the margin
- I.e. C is a way to do regularization in SVMs

Regularization of SVM



Large value for parameter *C*



Small value for parameter *C*

RBF Kernel – Gamma

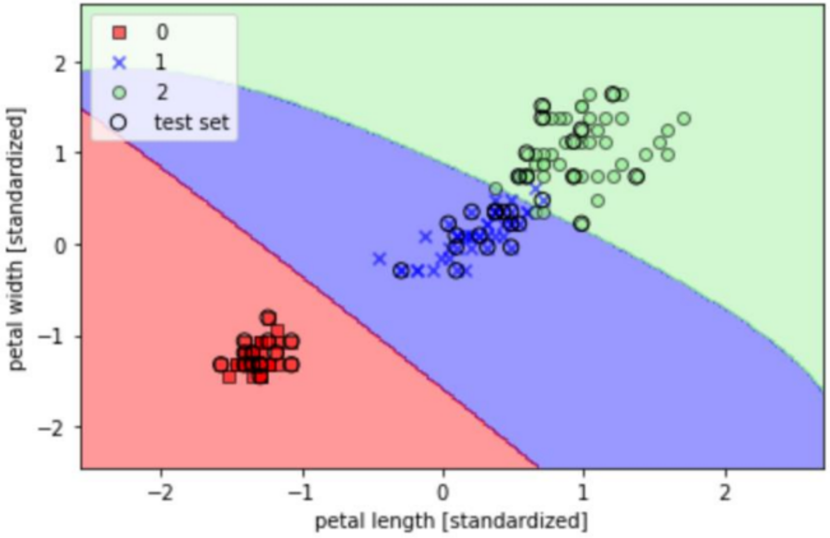
- Influence of gamma

  Gamma affects decision boundary

```python
from sklearn.svm import SVC

svm = SVC(kernel='rbf', random_state=0, gamma=0.2, C=1.0)
svm.fit(X_train_std, y_train)

plot_decision_regions(X_combined_std, y_combined,
                      classifier=svm, test_idx=range(105, 150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.tight_layout()
# plt.savefig('./figures/support_vector_machine_rbf_iris_1.png', dpi=300)
plt.show()
```

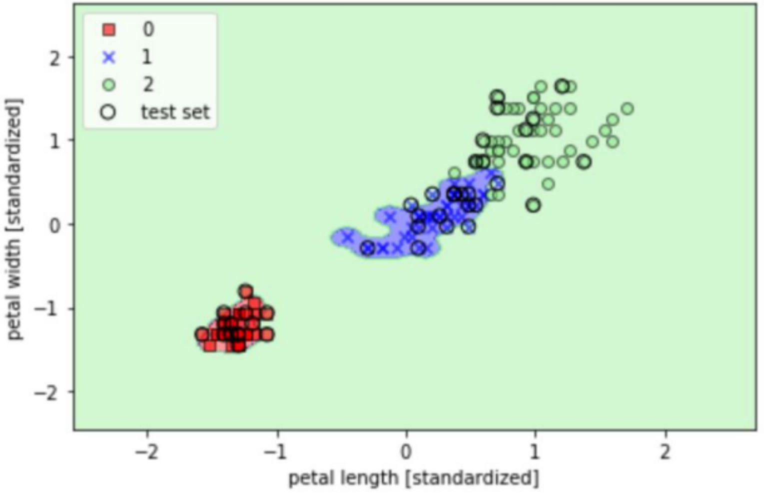


RBF Kernel – Gamma

Influence of gamma

```python
svm = SVC(kernel='rbf', random_state=0, gamma=100.0, C=1.0)
svm.fit(X_train_std, y_train)

plot_decision_regions(X_combined_std, y_combined,
                      classifier=svm, test_idx=range(105, 150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.tight_layout()
# plt.savefig('./figures/support_vector_machine_rbf_iris_2.png', dpi=300)
plt.show()
```



Assignment for week 5:

- Read and visualize the dataset
- Write your own kNN algorithm
- Visualize results
- Compute accuracy (you can use sklearn or your own)
- Apply Linear SVM algorithm to this data (use sklearn)
- Compute accuracy (you can use sklearn or your own)
- Visualize decision boundary and margins
- Compare accuracies of kNN and SVM