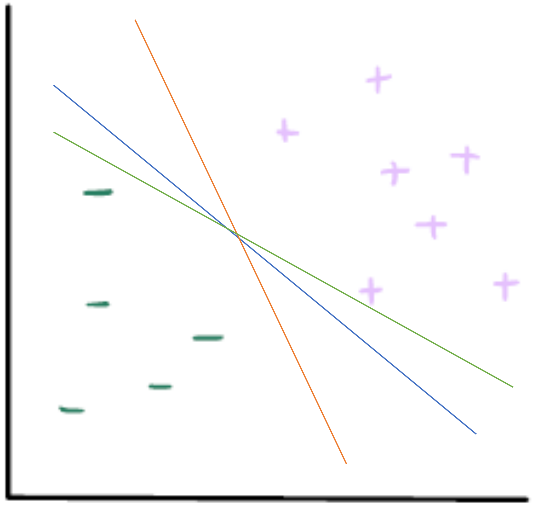
Support Vector Machines

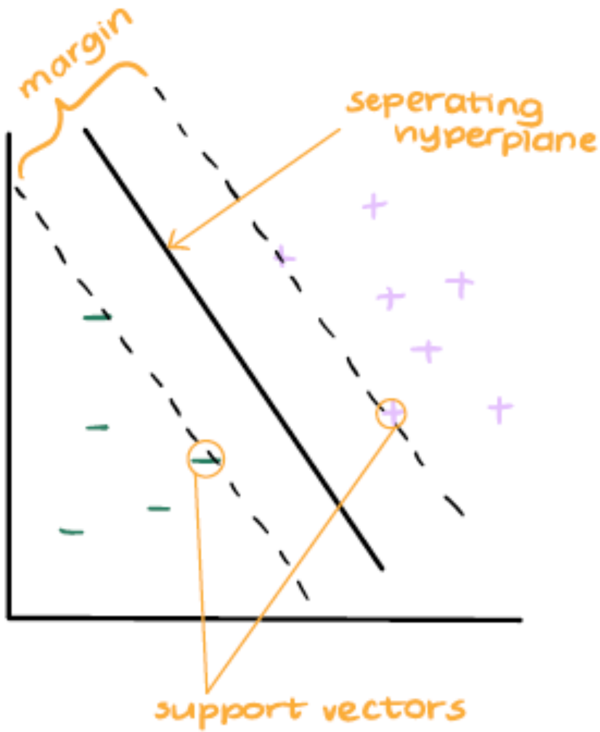
PS2705 – Final Presentation

Isil Idrisoglu

What are SVMs

- A classification approach





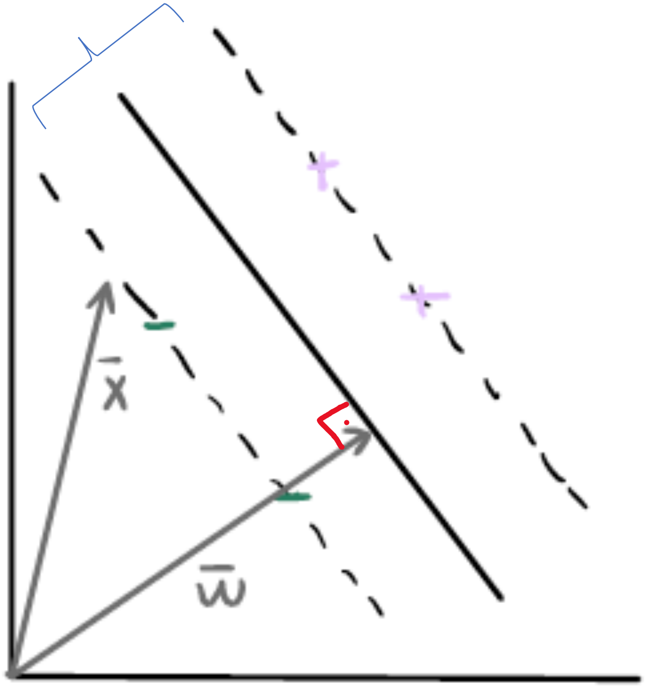
How does it work ?

- We need to make a decision rule.

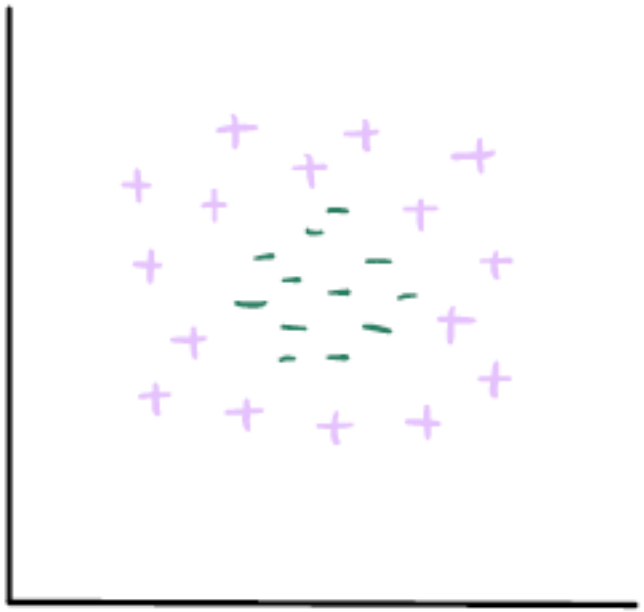
- $\boxed{\text{If } \hat{w} \cdot \hat{x} + b \geq 0 \text{, then } \hat{x} \text{ belong to } +}$

$$\hat{w} \cdot \hat{x}_{+} + b \geq 1$$
$$\hat{w} \cdot \hat{x}_{-} + b \leq -1$$

Maximize the margin

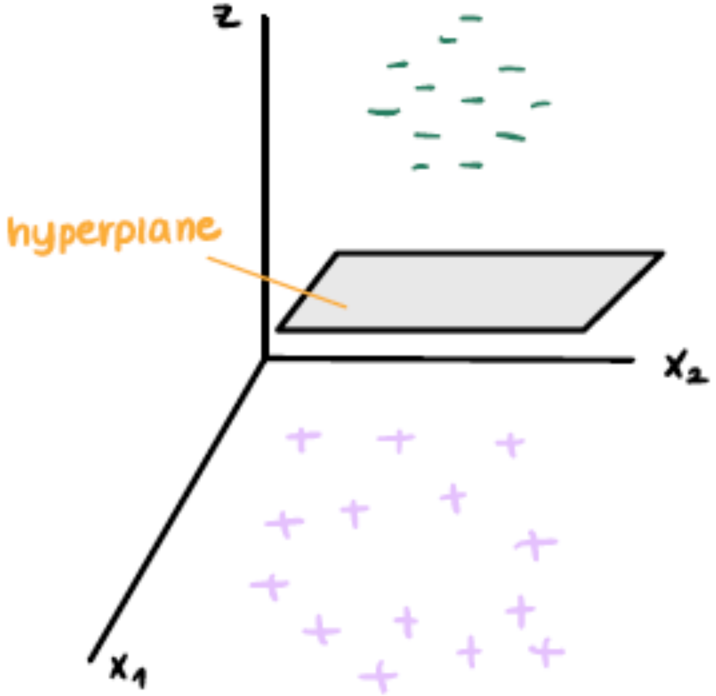


What if samples are not linearly separable?



What if samples are not linearly separable?

- Kernel trick
- $K(X_i, X_j) = \phi(X_i) \cdot \phi(X_j)$



A small historical background

Titanic Survival Classification

In [1]:
```
# imports
import  numpy as np
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline
```

In [2]:
```
# check if dataset exist
!ls
```

```
README.md          titanic-dataset.csv titanic-svm.pdf
requirements.txt   titanic-svm.ipynb
```

In [3]:
```
df = pd.read_csv("./titanic-dataset.csv")

print("Dataset shape: ", df.shape)
df.head()
```

```
Dataset shape:  (891, 12)
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Data Preparation

```
Check the distribution of Survived
```

In [4]:
```
df['Survived'].value_counts()
```

Out[4]:
```
Survived
0    549
1    342
Name: count, dtype: int64
```

```
Drop irrelevant variables, make sex a dummy variable
```

In [5]:
```
df.drop(['PassengerId','Name','Ticket','Fare','Embarked'],axis=1, inplace=True)
df.loc[df['Sex']=='male','Sex']=1
df.loc[df['Sex']=='female','Sex']=0
```

```
Check the percentage of null data
```

In [6]:
```
((df.isnull().sum())/len(df))*100
```

Out[6]:
```
Survived     0.000000
Pclass       0.000000
Sex          0.000000
Age         19.865320
SibSp        0.000000
Parch        0.000000
Cabin       77.104377
dtype: float64
```

In [7]:
```
df.drop('Cabin', axis=1,inplace=True)
df['Age'].fillna(df['Age'].mean(), inplace = True)
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Sex       891 non-null    object 
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64  
 5   Parch     891 non-null    int64  
dtypes: float64(1), int64(4), object(1)
memory usage: 41.9+ KB
```

Model

```
In [9]: X = df.drop('Survived', axis=1)  # features
        y = df['Survived']  # labels

        X.shape, y.shape

Out[9]: ((891, 5), (891,))
```

Train, test split

“Supervised Learning”:
means we need training data

```
In [10]: from sklearn.model_selection import train_test_split

         # random_state = random seed
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=101)

         X_train.shape, X_test.shape

Out[10]: ((801, 5), (90, 5))
```

Replication!

90 – 10 split

Data Standardization

```
In [11]: from sklearn.preprocessing import StandardScaler

         scaler = StandardScaler()
         scaler.fit(X_train)

         scaled_X_train= scaler.transform(X_train)
         scaled_X_test= scaler.transform(X_test)
```

The standard score of sample x is calculated as:

- $z = (x - u) / s$

SVM Model

We are applying soft SVM here, with default C value and linear kernel

In [12]:
```
from sklearn.svm import SVC

model = SVC(kernel='linear')
model.fit(scaled_X_train, y_train)
```

Out[12]:
```
▾ SVC
SVC(kernel='linear')
```

In [13]:
```
y_pred = model.predict(scaled_X_test)
```

In [14]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.74	0.90	0.81	51
1	0.82	0.59	0.69	39
accuracy			0.77	90
macro avg	0.78	0.75	0.75	90
weighted avg	0.78	0.77	0.76	90

Hyperparameter Tuning

So far we used default values for hyperparameters, which is C.

In [15]:
```python
# This gives us the implementation details and the parameters.

help(SVC)
```

```
|  ----------
|  C : float, default=1.0
|      Regularization parameter. The strength of the regularization is
|      inversely proportional to C. Must be strictly positive. The penalty
|      is a squared l2 penalty.
|
|  kernel : {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable,          default='rbf'
|      Specifies the kernel type to be used in the algorithm.
|      If none is given, 'rbf' will be used. If a callable is given it is
|      used to pre-compute the kernel matrix from data matrices; that matrix
|      should be an array of shape ``(n_samples, n_samples)``.
|
|  degree : int, default=3
|      Degree of the polynomial kernel function ('poly').
|      Must be non-negative. Ignored by all other kernels.
|
|  gamma : {'scale', 'auto'} or float, default='scale'
|      Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.
|
|      - if ``gamma='scale'`` (default) is passed then it uses
```

```
In [16]: from sklearn.model_selection import GridSearchCV

         # in order to filter some sklearn warnings
         import warnings
         warnings.filterwarnings('ignore')

         svm = SVC(max_iter=500)
         param_grid = {'C':[0.01, 0.1, 1, 10], 'gamma': [1, 0.1, 0.01, 0.001],
                       'kernel': ['linear', 'rbf']}
         grid = GridSearchCV(svm, param_grid)
         grid.fit(scaled_X_train, y_train)

         grid.best_params_

Out[16]: {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
```

```
In [17]: y_pred_grid = grid.predict(scaled_X_test)
         print(classification_report(y_test, y_pred_grid))
```

True positives / (True Positives + False Positives)

	precision	recall	f1-score	support
0	0.79	0.94	0.86	51
1	0.90	0.67	0.76	39
accuracy			$\boxed{0.82}$	90
macro avg	0.84	0.80	0.81	90
weighted avg	0.83	0.82	0.82	90

True positives / (True positives + False negatives)

Previous one was 0.77

SVMs vs. Logistic Regression

- Data type
  - SVM: unstructured and semi-structured data like text and images
  - LR: already identified independent variables.

- SVM is based on the geometrical properties of the data, while logistic regression is based on statistical approaches.

Advantages

- Handles non-linearly separable data
- Effective in high dimensions - can work with a large number of features
- Robustness to outliers – soft SVM
- Allows improving the model by hyperparameter tuning
- Less risk of overfitting since we maximize margin
- Memory efficiency

Disadvantages

- Hyperparameter tuning
- It may become computationally expensive
- Commonly used for *binary* classification problems, multiclass may be not as efficient
  - Overlapping classes may be a problem as well
- Dataset problems

Social Science Applications

- Using ‘unconventional’ data:
  - Text as data: sentiment analysis, topic modeling, or content analysis
  - Image classification: facial and object recognition, or emotion and motion detection
- Useful predictions
  - Predicting individuals’ participation in surveys (Kirchner and Signorino, 2018)
  - Predicting outcome of militarized conflict (Beck et al. 2000; Marvala and Lagazio, 2011; Colaresi and Mahmood, 2017)
  - Election forecasts (Zolghadr, Niaki, and Niaki, 2018)

Additional Resources

- Aggarwal, C.C. (2018). Linear Classification and Regression for Text. In: Machine Learning for Text. Springer, Cham. https://doi.org/10.1007/978-3-319-73531-3_6
- Winston, P. (Fall, 2010). Learning: Support Vector Machines [Video]. MIT OpenCourseWare, Youtube. https://www.youtube.com/watch?v=_PwhiWxHK8o
- Marwala, T., & Lagazio, M. (2011). Militarized Conflict Modeling Using Computational Intelligence. *Advanced Information and Knowledge Processing*.
- Scikit-Learn website for understanding sci-kit library: https://scikit-learn.org/stable/index.html
- Check the Kaggle website for additional datasets and tutorials on SVMs.

References

- Beck, N., King, G., Zeng, L.: Improving quantitative studies of international conflict: a conjecture. Am. Politic Sci. Rev. 94, 21–33 (2000)
- Colaresi, M., & Mahmood, Z. (2017). Do the robot: Lessons from machine learning to improve conflict forecasting. *Journal of Peace Research*, 54(2), 193–214. http://www.jstor.org/stable/44511206
- Kirchner, Antje, and Curtis S. Signorino. 2018. "Using Support Vector Machines for Survey Research." Survey Practice 11 (1). https://doi.org/10.29115/SP-2018-0001.
- Marwala, T., & Lagazio, M. (2011). Militarized Conflict Modeling Using Computational Intelligence. Advanced Information and Knowledge Processing.
- Precision-recall. Scikit-Learn. (n.d.). Retrieved April 17, 2023, from https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html
- Winston, P. (Fall, 2010). Learning: Support Vector Machines [Video]. MIT OpenCourseWare, Youtube. https://www.youtube.com/watch?v=_PwhiWxHK8o
- Zolghadr, M., Niaki, S.A.A. & Niaki, S.T.A. Modeling and forecasting US presidential election using learning algorithms. *J Ind Eng* Int 14, 491–500 (2018). https://doi.org/10.1007/s40092-017-0238-2