Supervised Machine Learning:
Naïve Bayes Classifier
Support Vector Machine

Sachin Pawar

schnpwr@gmail.com

Outline

- Naïve Bayes Classifier
- Naïve Bayes using sklearn

- Support Vector Machine (SVM) Classifier
- SVM using sklearn

Naïve Bayes Classifier

Naïve Bayes Classifier: Introduction

- Naïve Bayes is a generative probabilistic classifier trained in a supervised way
- Goal: To predict the most probable class label y for a given instance $\mathbf{x}$ which is represented using *n* difference features $\mathbf{x} = [x_1, x_2, \cdots, x_n]$
- It is based on the Bayes' Theorem which is formally stated as follows:

$$p(A|B) = \frac{p(B|A) \cdot p(A)}{p(B)}$$

  - where A and B are events such that $P(B) \neq 0$
  - $p(A|B)$ is the conditional probability of the event A occurring given the event B has already occurred.
  - $p(B|A)$ is the conditional probability of the event B occurring given the event A has already occurred.
  - $p(A)$ and $p(B)$ are the marginal probabilities of observing the events A and B, resp.

Naïve Bayes Classifier: Probabilistic Model

- The conditional probability of the class label y given the instance $\mathbf{x}$ is computed as follows:

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y) \cdot p(y)}{p(\mathbf{x})}$$

- The most probable class label is chosen as:

$$y^* = \underset{y}{\operatorname{argmax}} \frac{p(\mathbf{x}|y) \cdot p(y)}{p(\mathbf{x})} = \underset{y}{\operatorname{argmax}}\, p(\mathbf{x}|y) \cdot p(y)$$

- An instance is represented by *n* features $\mathbf{x} = [x_1, x_2, \cdots, x_n]$

$$y^* = \underset{y}{\operatorname{argmax}}\, p(x_1, x_2, \cdots, x_n|y) \cdot p(y)$$

Why the Naïve Bayes Classifier is called "Naïve" ?

- Conditional Independence Assumption: Any feature of an instance is conditionally independent of all other features given the class label.

$$p(x_1, x_2, \cdots, x_n | y) = \prod_{i=1}^{n} p(x_i | y)$$

- Hence, the Naïve Bayes classifier chooses the most probable class label for any given instance as follows:

$$y^* = \underset{y}{\operatorname{argmax}}\ p(y) \cdot \prod_{i=1}^{n} p(x_i | y)$$

- Simplifies the probability estimation a lot leading to a smaller number of parameters to learn and fast training
- Conditional Independence assumptions may not hold for some features in practice

Naïve Bayes Classifier: Training

- Training instances: $D = \{\langle \mathbf{x}^1, y^1 \rangle, \langle \mathbf{x}^2, y^2 \rangle, \cdots \langle \mathbf{x}^N, y^N \rangle\}$
- Each training instance is characterized by *n* features $\mathbf{x}^i = [x_1^i, x_2^i, \cdots, x_n^i]$ and the corresponding class is $y^i \in \mathbf{Y}$ which is a set of all possible class labels

- Training a Naïve Bayes classification model is equivalent to learning the parameters needed to estimate the following probabilities
  - Prior probability of each class

    $$p(y) \,, \forall_{y \in \mathbf{Y}}$$

  - Conditional probability of each feature value given each class

    $$p(x_i | y), \forall_{y \in \mathbf{Y}, 1 \leq i \leq n}$$

  - The necessary parameters are estimated using Maximum Likelihood Estimation (MLE)
  - The parameters to be estimated are dependent on the nature of each feature, e.g., categorical, real-valued

Naïve Bayes Classifier: Example

Outlook	**Temperature**	**Humidity**	**Wind**	**Play Tennis? (*y*)**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

$$p(y = Yes) = \frac{9}{14}$$

$$p(y = No) = \frac{5}{14}$$

Naïve Bayes Classifier: Example

	x_1 **Outlook**		
Play Tennis	Sunny	Overcast	Rain
Yes	(2+1)/(9+3)	(4+1)/(9+3)	(3+1)/(9+3)
No	(3+1)/(5+3)	(0+1)/(5+3)	(2+1)/(5+3)

	x_3 **Humidity**	
Play Tennis	High	Normal
Yes	(3+1)/(9+2)	(6+1)/(9+2)
No	(4+1)/(5+2)	(1+1)/(5+2)

	x_2 **Temperature**		
Play Tennis	Hot	Mild	Cool
Yes	(2+1)/(9+3)	(4+1)/(9+3)	(3+1)/(9+3)
No	(2+1)/(5+3)	(2+1)/(5+3)	(1+1)/(5+3)

	x_4 **Wind**	
Play Tennis	Strong	Weak
Yes	(3+1)/(9+2)	(6+1)/(9+2)
No	(3+1)/(5+2)	(2+1)/(5+2)

Naïve Bayes Classifier: Example

- Instance to be classified:

$$\mathbf{x} = [x_1 = Rain, x_2 = Mild, x_3 = High, x_4 = Strong]$$

- Class label for this instances in to be predicted as follows:

$$y^* = \underset{y}{\operatorname{argmax}}\ p(y) \cdot \prod_{i=1}^{n} p(x_i|y)$$

$$\begin{aligned} Score(y = Yes|\mathbf{x}) \\ &= p(y = Yes) \cdot p(x_1 = Rain|y = Yes) \\ &\cdot p(x_2 = Mild|y = Yes) \cdot p(x_3 = High|y = Yes) \\ &\cdot p(x_4 = Strong|y = Yes) \end{aligned}$$

$$Score(y = Yes|\mathbf{x}) = \frac{9}{14} \cdot \frac{4}{12} \cdot \frac{5}{12} \cdot \frac{4}{11} \cdot \frac{4}{11} = 0.01181$$

Naïve Bayes Classifier: Example

- Instance to be classified:

$$\mathbf{x} = [x_1 = Rain, x_2 = Mild, x_3 = High, x_4 = Strong]$$

- Class label for this instances in to be predicted as follows:

$$y^* = \operatorname*{argmax}_y \, p(y) \cdot \prod_{i=1}^{n} p(x_i|y)$$

$$\begin{aligned} Score(y = No|\mathbf{x}) \\ &= p(y = No) \cdot p(x_1 = Rain|y = No) \\ &\cdot p(x_2 = Mild|y = No) \cdot p(x_3 = High|y = No) \\ &\cdot p(x_4 = Strong|y = No) \end{aligned}$$

$$Score(y = No|\mathbf{x}) = \frac{5}{14} \cdot \frac{3}{8} \cdot \frac{3}{8} \cdot \frac{5}{7} \cdot \frac{4}{7} = 0.0205$$

$$\boxed{Predicted\ class = No}$$

Gaussian Naïve Bayes Classifier

- The "Play Tennis" dataset consisted of only categorical features
- In case, any feature is real-valued (continuous values), then a different approach is needed for estimating class conditional probabilities
- Gaussian Naïve Bayes assumes that the continuous feature values associated with each class follow Normal (Gaussian) distribution
- For any i^{th} feature, for the class label *y*, the necessary parameters (mean and standard deviation) are computed as follows:

$$\mu_{(i,y)} = \frac{1}{N_y}\sum_{j=1}^{N}[y = y^j] \cdot x_i^j \qquad \sigma^2_{(i,y)} = \frac{1}{N_y}\sum_{j=1}^{N}[y = y^j] \cdot \left(x_i^j - \mu_{(i,y)}\right)^2$$

  - Where, N_y is the number of training instances with true class label as *y*

Multinomial Naïve Bayes Classifier

- A variant of Naïve Bayes Classifier which is suitable for text classification tasks in Natural Language Processing (NLP)
  - Email classification (SPAN / NOT SPAM); News articles classification (SPORTS / POLITICS / ENTERTAINMENT / BUSINESS / SCIENCE)
- An instance to be classified is generally a piece of text – sentence or paragraph or complete document
  - Represented using a Bag-of-words strategy – word order is ignored; word frequency is important
- Inference rule for such an instance $\mathbf{x} = [w_1, w_2, \cdots, w_{len}]$ using Multinomial Naïve Bayes uses the Multinomial Distribution to estimate the probability of any word being generated for a particular class
  - E.g., **x** = [`Serum, Institute, to, provide,` **`vaccine`**`, at, Rs, 225, under, new, agreement, with, Gates, Foundation, SII, will, produce, up, to, 100, million, Covid-19,` **`vaccine`**`, doses, for, India, and, low, and, middle-income, countries`]

$$y^* = \underset{y}{\operatorname{argmax}}\ p(y) \cdot \prod_{i=1}^{len} p(w_i|y) = \underset{y}{\operatorname{argmax}} \log p(y) + \sum_{i=1}^{len} \log p(w_i|y)$$

Naïve Bayes Classifier using sklearn

Car Evaluation Dataset

- Dataset from the UCI Machine Learning Repository
  - https://archive.ics.uci.edu/ml/datasets.php
- Number of instances = 1728
- Number of attributes = 6
- **Attributes / Features**:
  - buying: vhigh, high, med, low
  - maint: vhigh, high, med, low
  - doors: 2, 3, 4, 5more
  - persons: 2, 4, more
  - lug_boot: small, med, big
  - safety: low, med, high
- **Class labels**: unacc, acc, good, vgood

```python
import pandas as pd
data_frame = pd.read_csv('../input/car-evaluation-data-set/car_evaluation.csv', header=None)
print(data_frame.columns)
X = data_frame.iloc[:,0:6].to_numpy()
y = data_frame.iloc[:,6].to_numpy()
print(X)
print(y)
```

```
Int64Index([0, 1, 2, 3, 4, 5, 6], dtype='int64')
[['vhigh' 'vhigh' '2' '2' 'small' 'low']
 ['vhigh' 'vhigh' '2' '2' 'small' 'med']
 ['vhigh' 'vhigh' '2' '2' 'small' 'high']
 ...
 ['low' 'low' '5more' 'more' 'big' 'low']
 ['low' 'low' '5more' 'more' 'big' 'med']
 ['low' 'low' '5more' 'more' 'big' 'high']]
['unacc' 'unacc' 'unacc' ... 'unacc' 'good' 'vgood']
```

```python
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
labelEncoderModel = LabelEncoder().fit(y)
y = labelEncoderModel.transform(y)
print(labelEncoderModel.classes_)
print(y)
```

```
['acc' 'good' 'unacc' 'vgood']
[2 2 2 ... 2 1 3]
```

```python
ordinalEncoderModel = OrdinalEncoder()
ordinalEncoderModel.fit(X)
X = ordinalEncoderModel.transform(X)
for category in ordinalEncoderModel.categories_:
    print(category)
print(X.shape)
print(X[0:2])
```

```
['high' 'low' 'med' 'vhigh']
['high' 'low' 'med' 'vhigh']
['2' '3' '4' '5more']
['2' '4' 'more']
['big' 'med' 'small']
['high' 'low' 'med']
(1728, 6)
[[3. 3. 0. 0. 2. 1.]
 [3. 3. 0. 0. 2. 2.]]
```

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=728, random_state=10)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(1000, 6)
(1000,)
(728, 6)
(728,)
```

```python
from sklearn.naive_bayes import CategoricalNB
NBC_model = CategoricalNB()
NBC_model.fit(X_train, y_train)
```

```
CategoricalNB()
```

```python
print(len(NBC_model.feature_log_prob_))
print(NBC_model.feature_log_prob_[0].shape)
print(NBC_model.feature_log_prob_[0])
```

```
6
(4, 4)
[[-1.39432653 -1.47840965 -1.2039728  -1.49610923]
 [-3.8501476  -0.59205106 -0.90570862 -3.8501476 ]
 [-1.26606693 -1.61090742 -1.58908837 -1.15758629]
 [-3.63758616 -0.59306372 -0.92953596 -3.63758616]]
```

```python
y_predicted = NBC_model.predict(X_test)
print(y_predicted)
```

```
[2 2 2 2 0 2 2 0 2 2 2 2 2 2 2 0 2 2 2 2 2 2 0 2 2 2 2 0 2 2 2 2 2 2 2 2 2
 2 2 0 2 2 2 0 2 0 2 2 2 2 2 0 0 0 0 2 0 2 0 2 2 2 2 0 2 2 0 2 2 2 3 2 2 2
 0 2 0 0 2 2 0 2 2 2 2 2 2 0 2 0 2 2 2 0 2 2 2 2 2 2 2 2 2 2 0 2 2 2 0 2 2
 2 2 2 0 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 0 2 2 0 2 2 3 2 2 0 2 0 3 2 2 2 2
 2 2 2 2 0 0 2 2 2 2 0 0 2 2 2 2 2 0 2 2 2 2 2 1 2 3 2 2 2 2 0 0 2 2 2 2 0
 2 0 2 0 0 0 2 2 2 0 3 1 0 2 2 2 2 0 2 2 2 2 2 2 0 0 0 0 2 2 0 2 2 0 2 0 2
 2 2 2 2 0 2 2 2 2 2 0 0 2 2 2 2 3 2 2 1 2 0 2 0 2 2 2 2 2 0 2 2 2 0 2 2 2
 2 2 2 0 2 2 2 0 2 2 0 2 2 2 1 0 2 2 2 2 2 2 0 0 0 0 2 1 2 2 0 0 2 2 0 2 2
 2 0 2 0 2 2 2 0 2 0 2 0 2 2 2 2 0 2 2 2 1 2 2 2 0 2 2 0 0 2 0 0 2 2 0 0 2
 2 2 2 2 0 2 2 2 0 2 2 2 2 2 2 2 2 1 0 0 2 2 2 2 0 2 2 2 3 2 2 0 2 1 2 0 2
 0 0 2 2 2 2 2 2 2 2 2 0 0 2 2 2 2 2 2 2 2 0 2 0 0 0 2 1 2 2 2 2 0 2 0 2 0
 2 2 2 2 2 2 2 0 0 2 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2 0 2 2 2 2 1 2 2
 2 2 2 2 2 0 2 2 2 0 2 2 0 0 2 2 2 0 0 2 2 2 2 2 2 0 2 2 2 0 2 2 2 2 0 2 2
 2 2 2 2 2 2 2 0 2 2 2 2 2 0 0 2 2 2 0 2 2 2 2 2 0 2 0 2 2 0 2 0 2 2 2 0 2
 0 0 2 2 2 0 2 2 2 2 0 2 2 2 2 2 2 2 2 0 2 2 2 2 0 1 2 0 2 2 2 2 0 2 2 2 2
 2 2 2 2 0 0 2 2 2 0 2 2 0 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 0 0 2 2 2 2 0 2
 2 2 2 2 2 2 2 2 2 2 2 2 0 2 0 2 0 2 0 2 2 0 2 0 2 2 2 0 2 2 2 2 2 0 2 2 0
 2 2 0 0 2 2 2 2 0 2 2 2 2 2 0 2 2 0 2 0 2 0 2 2 0 2 1 2 2 2 2 2 0 2 2 0 0
 2 2 2 2 2 1 2 2 2 2 2 2 0 0 2 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 0 0 1 2 2 2 2
 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2]
```

```python
from sklearn.metrics import classification_report
report = classification_report(y_test, y_predicted)
print(report)
```

	precision	recall	f1-score	support
0	0.61	0.75	0.67	138
1	0.50	0.31	0.38	26
2	0.95	0.95	0.95	533
3	1.00	0.23	0.37	31
accuracy			0.86	728
macro avg	0.76	0.56	0.59	728
weighted avg	0.87	0.86	0.85	728

AG's News Classification Dataset

- Dataset from the Kaggle
  - https://www.kaggle.com/amananandrai/ag-news-classification-dataset
  - Number of instances = 120000 (training), 7600 (test)
- Number of attributes = 2
- **Attributes / Features**:
  - title: Title of a news article in text form
  - description: Description in text form
- **Class labels**:
  - 1: World news
  - 2: Sports news
  - 3: Business news
  - 4: Science-Technology news

```python
data_frame = pd.read_csv('../input/ag-news-classification-dataset/train.csv', header=0)
print(data_frame.columns)
X_train = data_frame['Description'].to_numpy()
y_train = data_frame['Class Index'].to_numpy()
print(X_train.shape)
print(y_train.shape)
print(X_train[0:2])
print(y_train[0:2])
```

```
Index(['Class Index', 'Title', 'Description'], dtype='object')
(120000,)
(120000,)
["Reuters - Short-sellers, Wall Street's dwindling\\band of ultra-cynics, are seeing green again."
 'Reuters - Private investment firm Carlyle Group,\\which has a reputation for making well-timed and occasionall
y\\controversial plays in the defense industry, has quietly placed\\its bets on another part of the market.']
[3 3]
```

```python
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
vectorizer.fit(X_train)
print(X_train.shape)
X_train = vectorizer.transform(X_train)
print(X_train.shape)

from sklearn.naive_bayes import MultinomialNB
NBC_model = MultinomialNB()
NBC_model.fit(X_train, y_train)
```

```
(120000,)
(120000, 60734)
```

```
MultinomialNB()
```

```
data_frame = pd.read_csv('../input/ag-news-classification-dataset/test.csv', header=0)
print(data_frame.columns)
X_test = data_frame['Description'].to_numpy()
y_test = data_frame['Class Index'].to_numpy()
print(y_test.shape)
print(X_test[0:2])
print(y_test[0:2])
print(X_test.shape)
X_test = vectorizer.transform(X_test)
print(X_test.shape)
```

```
Index(['Class Index', 'Title', 'Description'], dtype='object')
(7600,)
["Unions representing workers at Turner   Newall say they are 'disappointed' after talks with stricken parent fi
rm Federal Mogul."
 'SPACE.com - TORONTO, Canada -- A second\\team of rocketeers competing for the  #36;10 million Ansari X Prize,
a contest for\\privately funded suborbital space flight, has officially announced the first\\launch date for its
manned rocket.']
[3 4]
(7600,)
(7600, 60734)
```

```python
y_predicted = NBC_model.predict(X_test)
print(y_predicted)

from sklearn.metrics import classification_report
report = classification_report(y_test, y_predicted)
print(report)
```

```
[3 4 4 ... 2 3 4]
              precision    recall  f1-score   support

           1       0.90      0.89      0.90      1900
           2       0.94      0.97      0.96      1900
           3       0.87      0.82      0.84      1900
           4       0.85      0.87      0.86      1900

    accuracy                           0.89      7600
   macro avg       0.89      0.89      0.89      7600
weighted avg       0.89      0.89      0.89      7600
```

```python
class_labels={1:"World news", 2:"Sports news", 3:"Business news", 4:"Science-Tech news"}
for i in range(20):
    print("\n"+data_frame.iloc[i,2])
    print('Predicted='+str(class_labels[y_predicted[i]])+'\nActual='+str(class_labels[y_test[i]]))
```

```
Unions representing workers at Turner   Newall say they are 'disappointed' after talks with stricken parent fir
m Federal Mogul.
Predicted=Business news
Actual=Business news

SPACE.com - TORONTO, Canada -- A second\team of rocketeers competing for the  #36;10 million Ansari X Prize, a
contest for\privately funded suborbital space flight, has officially announced the first\launch date for its ma
nned rocket.
Predicted=Science-Tech news
Actual=Science-Tech news
```

```
European Space Agency -- ESAs Mars Express has relayed pictures from one of NASA's Mars rovers for
the first time, as part of a set of interplanetary networking demonstrations.     The demonstration
s pave the way for future Mars missions to draw on joint interplanetary networking capabilities...
Predicted=Science-Tech news
Actual=Science-Tech news

When did life begin? One evidential clue stems from the fossil records in Western Australia, althou
gh whether these layered sediments are biological or chemical has spawned a spirited debate. Oxford
researcher, Nicola McLoughlin, describes some of the issues in contention.
Predicted=Science-Tech news
Actual=Science-Tech news

update Earnings per share rise compared with a year ago, but company misses analysts' expectations
by a long shot.
Predicted=Business news
Actual=Science-Tech news
```

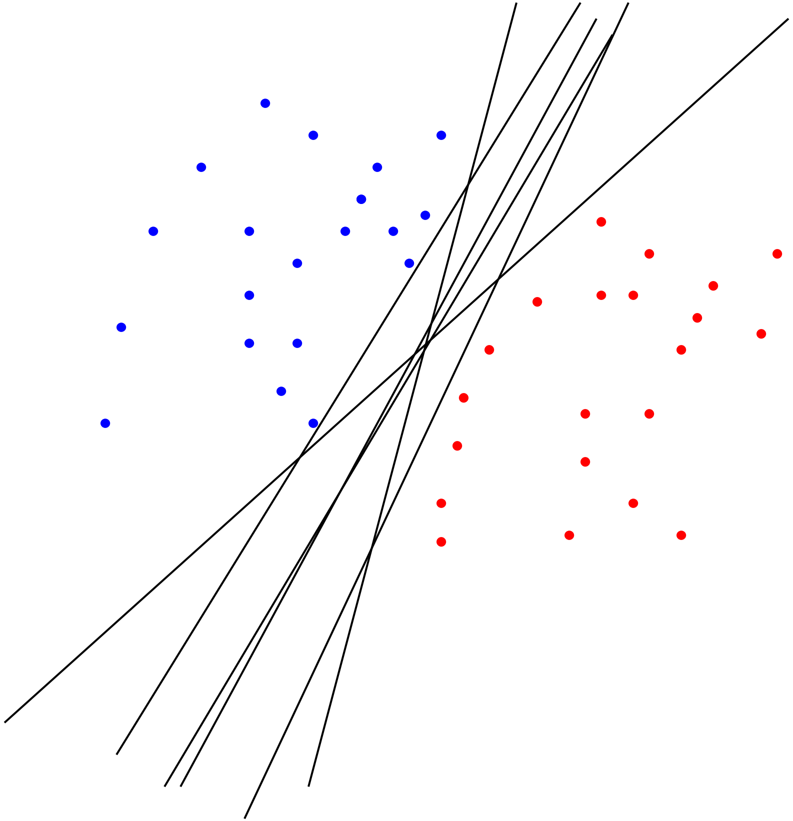
Support Vector Machine (SVM)

Introduction

- Support Vector Machine (SVM): Learns a linear separator for separating instances belonging to two different classes
  - In case of 1 dimensional instances, the separator is a point
  - In case of 2 dimensional instances, the separator is a line
  - In case of 3 dimensional instances, the separator is a plane
  - In case of instances in more than 3 dimensional space, the separator is a hyperplane

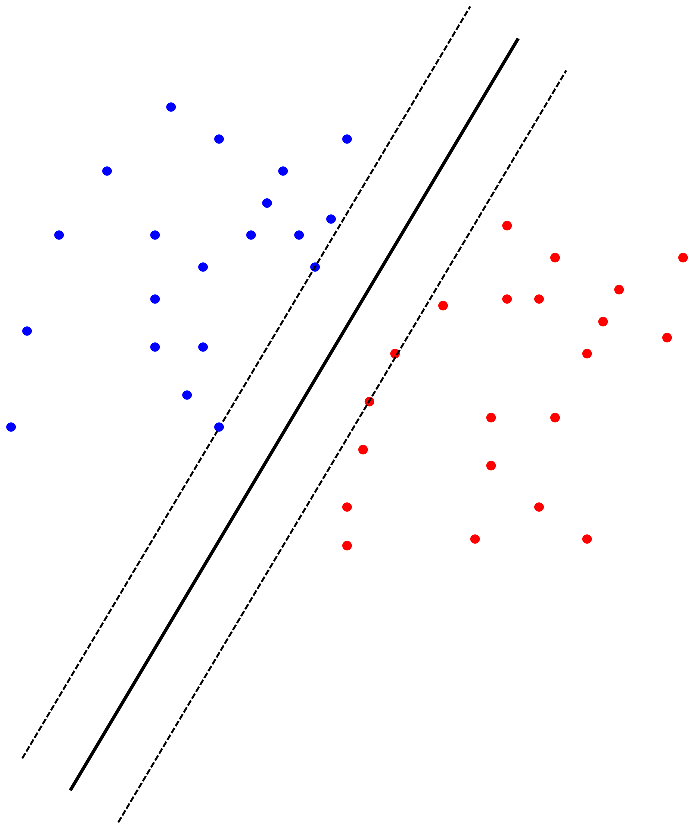
- Given a set of linearly separable instances, there exist infinite number of linear separators which can separate the instances into 2 classes
  - SVM chooses that linear separator which has the maximum "margin"
  - Intuition: A linear separator with the maximum margin will generalize better for new unseen instances in test data

SVM: Linear Separator



- An example of two dimensional instances
- Two classes:
  - Positive and Negative
- Linearly separable data
- Infinite number of linear separators are possible

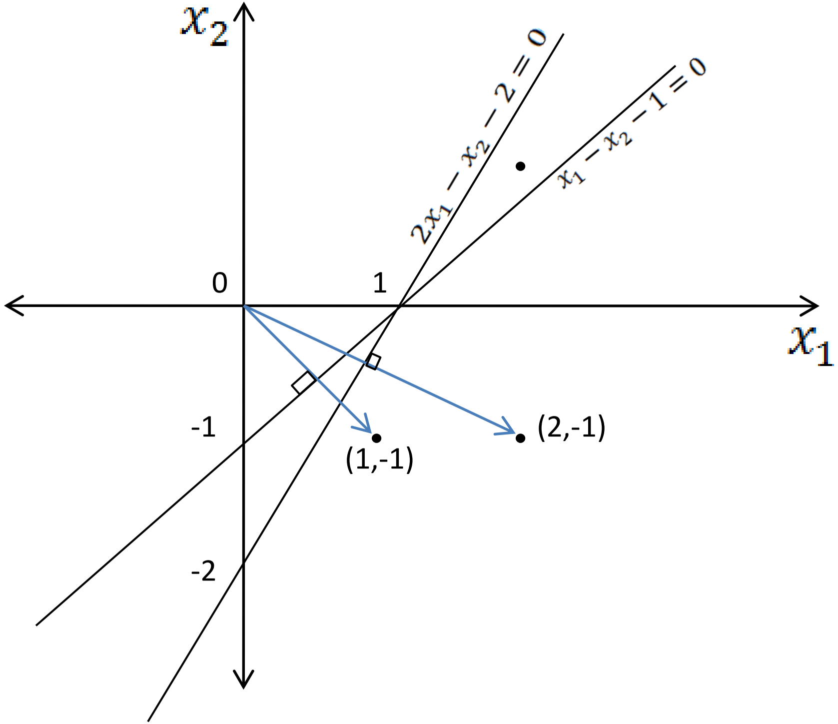
SVM: Linear Separator



- **Goal**: To find the linear separator which provides the maximum margin of separation between two classes
- Support vectors: Instances which lie on the margin boundaries

Representation of a linear separator

- A linear separator in n-dimensional space is represented using an equation of the form: $w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + w_0 = 0$
- Can be expressed in a vector form as: $\mathbf{w}^T \cdot \mathbf{x} + w_0 = 0$

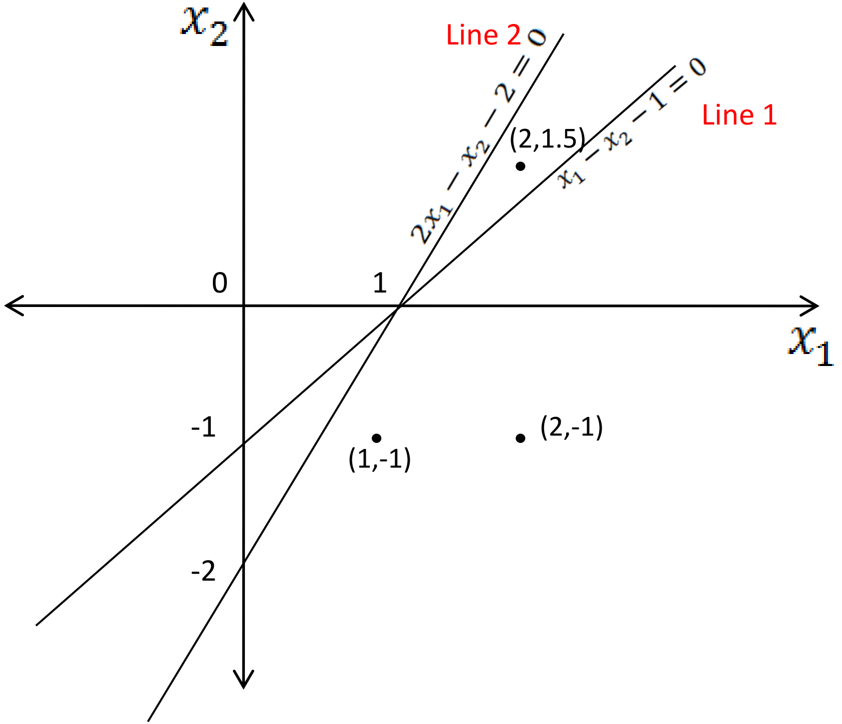


- The vector $\mathbf{w}$ is perpendicular to the lines of the form $\mathbf{w}^T \cdot \mathbf{x} + w_0 = 0$

- The learning task in SVM is to determine the optimal values of the parameters representing the linear separator with the maximum margin, i.e., $\mathbf{w}$ and w_0

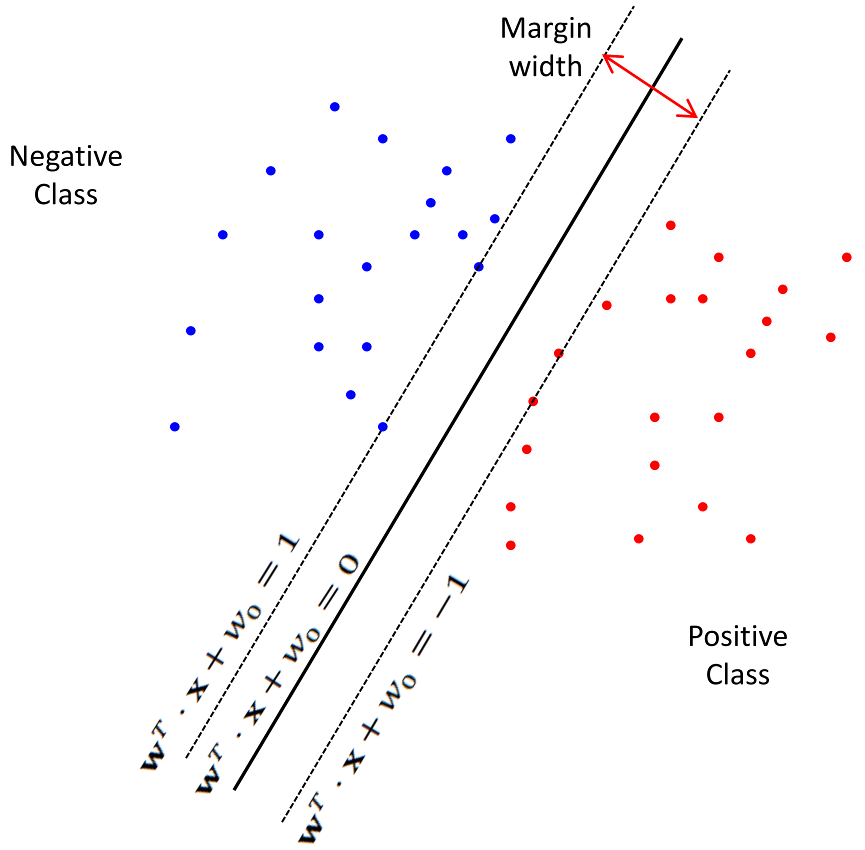
Representation of a linear separator

- A linear separator in n-dimensional space is represented using an equation of the form: $w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + w_0 = 0$
- Can be expressed in a vector form as: $\mathbf{w}^T \cdot \mathbf{x} + w_0 = 0$



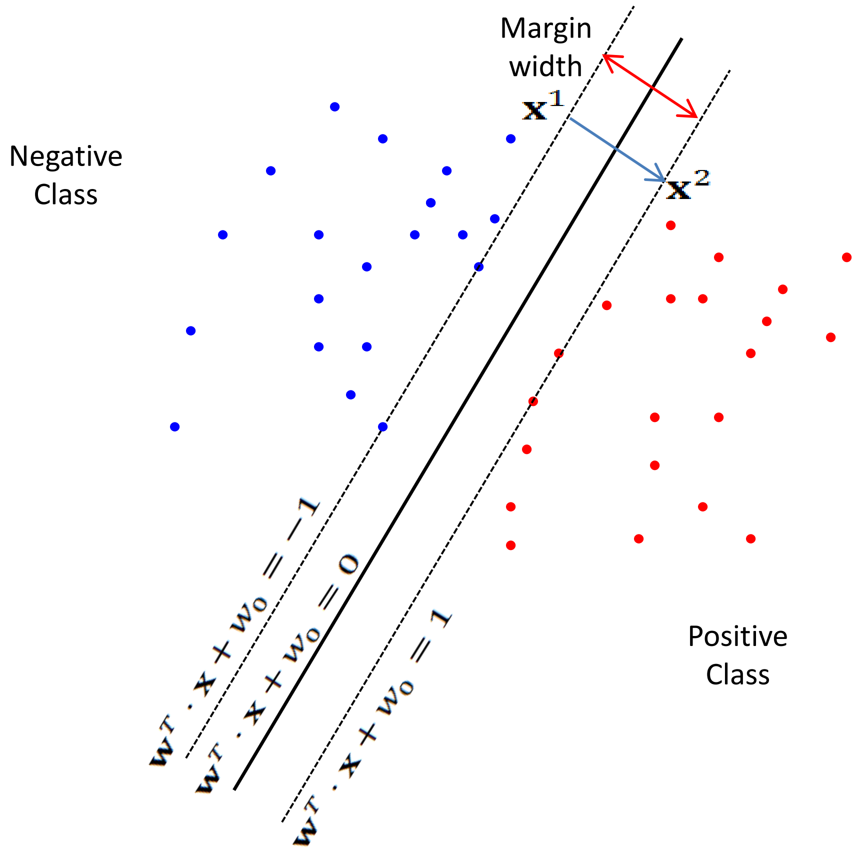
- For a particular linear separator, any point $\mathbf{x}^i$ lying on the "positive" side will have positive value of $\mathbf{w}^T \cdot \mathbf{x}^i + w_0$
- Also, any point $\mathbf{x}^i$ lying on the "negative" side will have negative value of $\mathbf{w}^T \cdot \mathbf{x}^i + w_0$
- E.g., the point (2,1.5) is on positive side of Line 2 (0.5) and on negative side of Line 1 (-0.5)

SVM: Training



- **Training instances**: $\{\langle \mathbf{x}^1, y^1 \rangle, \langle \mathbf{x}^2, y^2 \rangle, \cdots \langle \mathbf{x}^N, y^N \rangle\}$
- $\mathbf{x}^i$ is a point in n-dimensional space
- $y^i \in \{+1, -1\}$ is its corresponding true class label
- **Goal**: To find optimal linear separator which maximizes the margin

Computing Margin Width



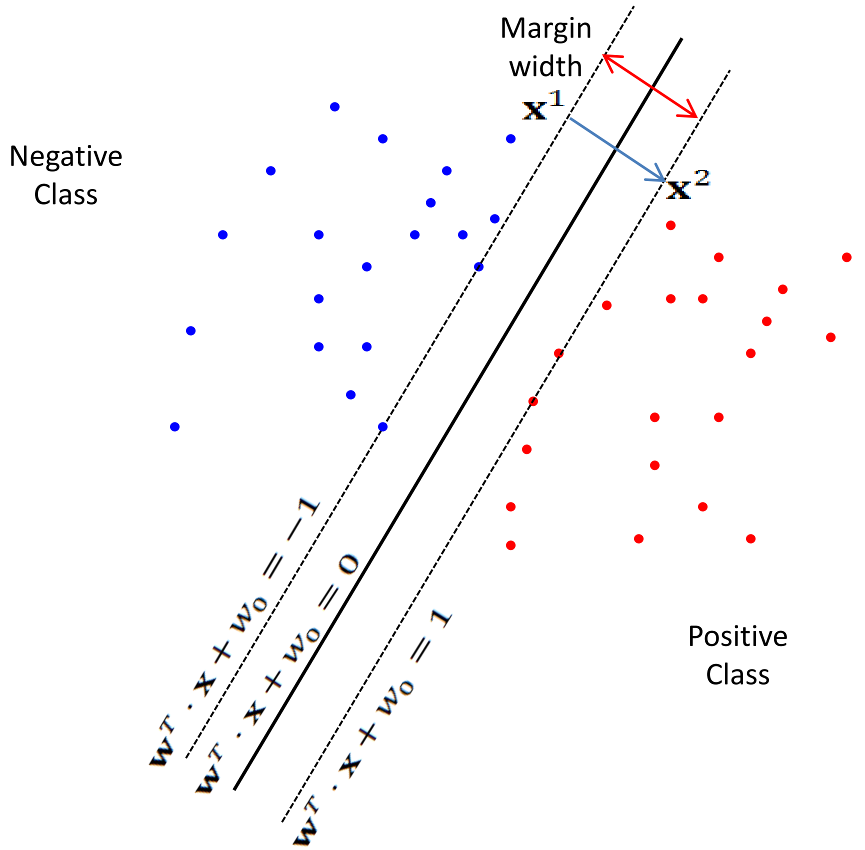
- Consider two points $\mathbf{x}^1$ and $\mathbf{x}^2$ such that they lie on the opposite margins and the vector $\mathbf{x}^2 - \mathbf{x}^1$ is perpendicular to the linear separator
- The vector $\mathbf{w}$ is also perpendicular to the linear separator
- Therefore, $(\mathbf{x}^2 - \mathbf{x}^1) = \lambda \cdot \mathbf{w}$
- By definition,

$$\mathbf{w}^T \cdot \mathbf{x}^1 + w_0 = -1$$

$$\mathbf{w}^T \cdot \mathbf{x}^2 + w_0 = 1$$

$$\mathbf{w}^T \cdot (\mathbf{x}^2 - \mathbf{x}^1) = 2$$

Computing Margin Width



- Substituting $(\mathbf{x}^2 - \mathbf{x}^1) = \lambda \cdot \mathbf{w}$

$$\mathbf{w}^T \cdot (\mathbf{x}^2 - \mathbf{x}^1) = 2$$

$$\lambda \mathbf{w}^T \cdot \mathbf{w} = 2 \Rightarrow \lambda = \frac{2}{\mathbf{w}^T \cdot \mathbf{w}}$$

- Margin width:

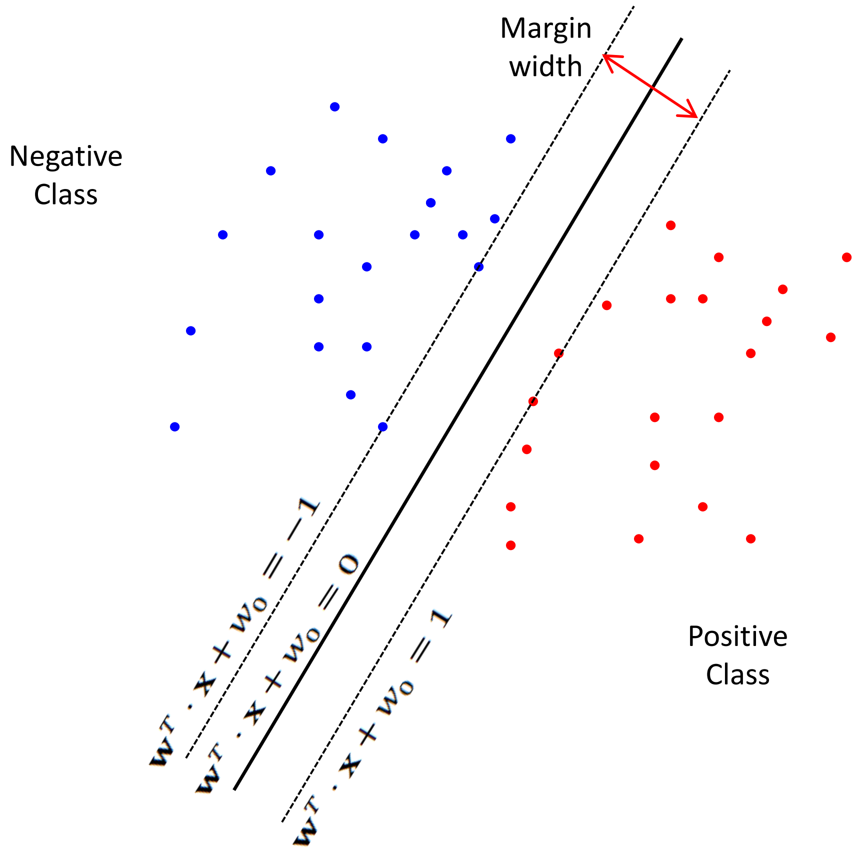
$$\|\mathbf{x}^2 - \mathbf{x}^1\| = \sqrt{(\mathbf{x}^2 - \mathbf{x}^1)^T \cdot (\mathbf{x}^2 - \mathbf{x}^1)}$$

$$\|\mathbf{x}^2 - \mathbf{x}^1\|^2 = \lambda^2 (\mathbf{w}^T \cdot \mathbf{w})$$

$$\|\mathbf{x}^2 - \mathbf{x}^1\|^2 = \frac{4}{(\mathbf{w}^T \cdot \mathbf{w})^2} (\mathbf{w}^T \cdot \mathbf{w})$$

$$\|\mathbf{x}^2 - \mathbf{x}^1\|^2 = \frac{4}{\mathbf{w}^T \cdot \mathbf{w}} \propto \frac{1}{\mathbf{w}^T \cdot \mathbf{w}}$$

SVM: Optimization Problem



- Objective:
  - Maximize the margin

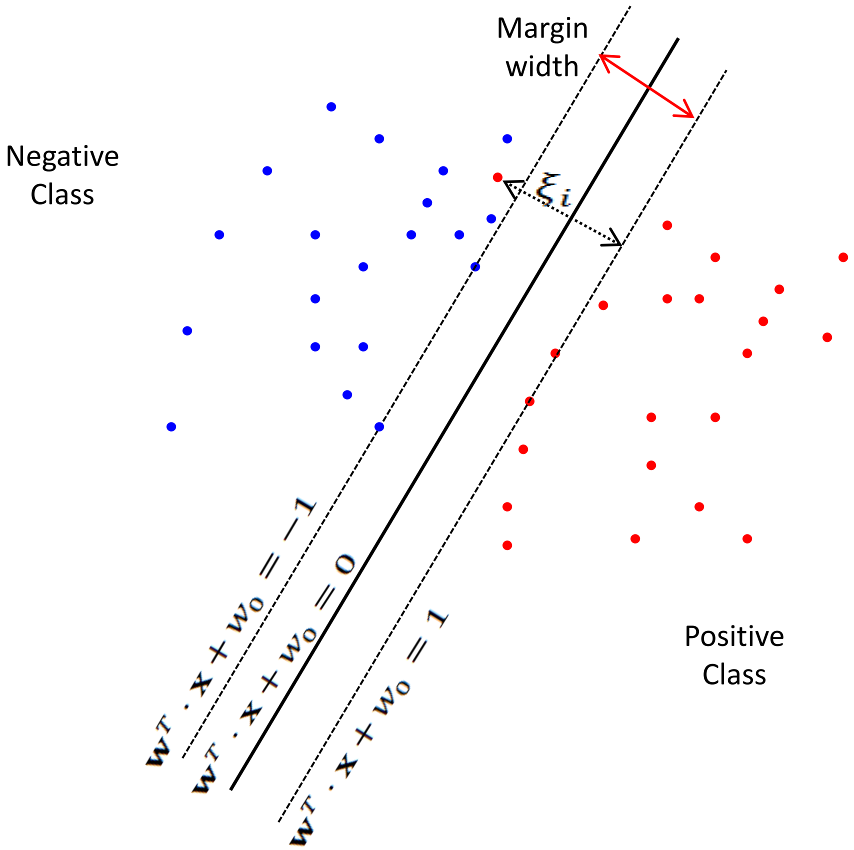
$$\min_{\mathbf{w}, w_0} \left(\frac{1}{2} \mathbf{w}^T \cdot \mathbf{w} \right)$$

- Subject to the following constraints:
  - Every training instance should lie on the appropriate (positive / negative) side of the linear separator

$$y^i \left(\mathbf{w}^T \cdot \mathbf{x}^i + w_0 \right) \geq 1, \forall_{1 \leq i \leq N}$$

$$-y^i \left(\mathbf{w}^T \cdot \mathbf{x}^i + w_0 \right) + 1 \leq 0, \forall_{1 \leq i \leq N}$$

SVM: Soft-margin Formulation



- **Objective**:
  - Maximize the margin and minimize the training error

$$\min_{\mathbf{w}, w_0, \xi} \left(\frac{1}{2}\mathbf{w}^T \cdot \mathbf{w}\right) + C \cdot \sum_{i=1}^{N} \xi_i$$

- Subject to the following constraints:
  - Introducing slack variables so that the constraint is satisfied for training instances lying on incorrect side

$$-y^i\left(\mathbf{w}^T \cdot \mathbf{x}^i + w_0\right) + 1 - \xi_i \leq 0, \forall_{1 \leq i \leq N}$$

$$-\xi_i \leq 0, \forall_{1 \leq i \leq N}$$

Optimization using Lagrange Multipliers

- One Lagrange multiplier is associated with each distinct constraint

$$L(\alpha_1, \cdots, \alpha_N, \mu_1, \cdots, \mu_N)$$

$$= \min_{\mathbf{w}, w_0, \xi} \left(\frac{1}{2} \mathbf{w}^T \cdot \mathbf{w} \right) + C \cdot \sum_{i=1}^{N} \xi_i$$

$$+ \sum_{i=1}^{N} \alpha_i \cdot \left(-y^i \left(\mathbf{w}^T \cdot \mathbf{x}^i + w_0 \right) + 1 - \xi_i \right)$$

$$+ \sum_{i=1}^{N} \mu_i \cdot (-\xi_i)$$

$$s.t. \quad \alpha_i \geq 0, \mu_i \geq 0, \forall_{1 \leq i \leq N}$$

Optimization using Lagrange Multipliers

- Differentiating w.r.t. $\mathbf{w}$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y^i \mathbf{x}^i = 0 \qquad \Rightarrow \boxed{\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i y^i \mathbf{x}^i}$$

- Differentiating w.r.t. w_0

$$\frac{\partial L}{\partial w_0} = \sum_{i=1}^{N} -\alpha_i y^i = 0 \;\Rightarrow \boxed{\sum_{i=1}^{N} \alpha_i y^i = 0}$$

- Differentiating w.r.t. ξ_i

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \;\Rightarrow \boxed{\mu_i + \alpha_i = C}$$

- Substituting optimal values:

$$L(\alpha_1, \cdots, \alpha_N, \mu_1, \cdots, \mu_N)$$

$$= \left(\frac{1}{2}\mathbf{w}^{*T} \cdot \mathbf{w}^*\right) + C \cdot \sum_{i=1}^{N} \xi_i^*$$

$$+ \sum_{i=1}^{N} \alpha_i \cdot \left(-y^i\left(\mathbf{w}^{*T} \cdot \mathbf{x}^i + w_0^*\right) + 1 - \xi_i^*\right) + \sum_{i=1}^{N} \mu_i \cdot (-\xi_i^*)$$

$$= \frac{1}{2}\left(\sum_{i=1}^{N} \alpha_i y^i \mathbf{x}^i\right)^T \cdot \left(\sum_{j=1}^{N} \alpha_j y^j \mathbf{x}^j\right) + C \cdot \sum_{i=1}^{N} \xi_i^*$$

$$+ \sum_{i=1}^{N} -\alpha_i y^i \left(\sum_{j=1}^{N} \alpha_j y^j \mathbf{x}^j\right)^T \mathbf{x}^i - w_0^* \sum_{i=1}^{N} \alpha_i y^i$$

$$+ \sum_{i=1}^{N} \alpha_i (1 - \xi_i) + \sum_{i=1}^{N} \mu_i \cdot (-\xi_i^*)$$

Terms involving ξ_i cancel each other out because $\mu_i + \alpha_i = C$

Sum in the red circle is zero

- Finally, we get:

$$L(\alpha_1, \cdots, \alpha_N) = \frac{-1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y^i y^j \left(\mathbf{x}^{i^T}\mathbf{x}^j\right) + \sum_{i=1}^{N}\alpha_i$$

- Dual optimization problem:
  - Objective function:

$$\max_{\alpha_1,\cdots,\alpha_N} \frac{-1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y^i y^j \left(\mathbf{x}^{i^T}\mathbf{x}^j\right) + \sum_{i=1}^{N}\alpha_i$$

  - Subject to the following constraints:

$$\alpha_i \geq 0, \mu_i \geq 0, \mu_i + \alpha_i = C, \forall_i \ \ and \ \sum_{i=1}^{N}\alpha_i y^i = 0$$

$$\Rightarrow \ \alpha_i \geq 0, \alpha_i \leq C, \forall_i \ \ and \ \sum_{i=1}^{N}\alpha_i y^i = 0$$

Any Quadratic Programming Solver can be used for solving this

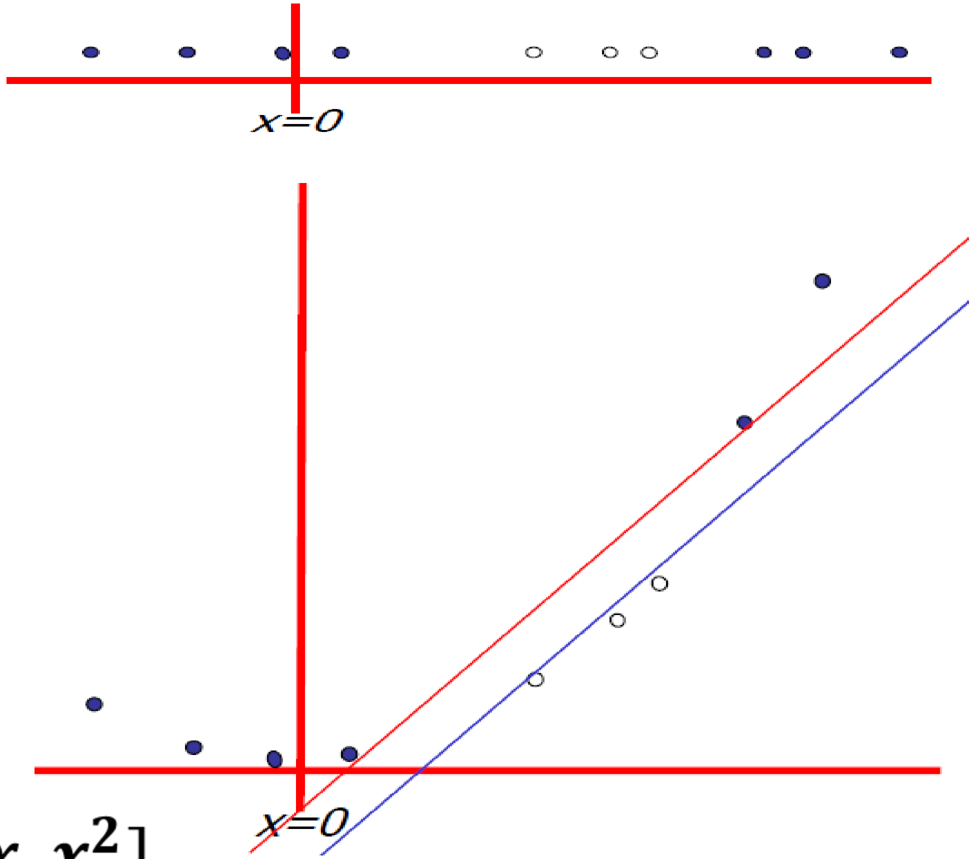
Using SVM for Predictions

- How to predict the class label for a new instance $\mathbf{x}$ given a trained SVM
- Primal Form:
  - Compute $\mathbf{w}^{*T} \cdot \mathbf{x} + w_0^*$; Positive value indicates the positive class and vice versa
  - E.g., $\mathbf{w}^* = [2, -1]$ and $w_0^* = -2$ be the learned parameters
  - For the new instance $\mathbf{x} = [2, 4]$, $\mathbf{w}^{*T} \cdot \mathbf{x} + w_0^* = -2$ and hence Negative class is predicted
- Dual Form:
  - Compute $\mathbf{w}^{*T} \cdot \mathbf{x} + w_0^* = \left(\sum_{i=1}^{N} \alpha_i y^i \mathbf{x}^i \right)^T \mathbf{x} + w_0^* = \sum_{i=1}^{N} \alpha_i y^i \mathbf{x}^{i^T} \mathbf{x} + w_0^*$
  - Positive value indicates the positive class and vice versa
  - Practically, most of the α_i values are zeros; non-zero only for support vectors

Linear Separability in Higher Dimensions

1 D :

2 D :



$$\phi([x]) = [x, x^2]$$

Courtesy : Prof. Andrew Moore's slides

Kernel Functions

- Instances which are not linearly separable in *n* dimensions can be separable in higher dimensional space
- Each instance in *n* dimensional space can be transformed to a higher dimensional space using a mapping ϕ
  - The following mapping transforms instances from a 2-dimensional space to 3-dimensional space

$$\phi(\mathbf{x} = [x_1, x_2]) = [x_1^2, x_2^2, \sqrt{2}x_1 x_2]$$

- Kernel function returns the value of the dot product in the transformed space

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

- A Kernel function can be defined in such a way that the dot product in the transformed space can be computed **without explicitly transforming the original instances**

$$K(\mathbf{x}, \mathbf{z}) = (x_1 z_1 + x_2 z_2)^2$$

SVM with Kernels (1/3)

- Dual formulation of SVM (training as well as prediction) involves only dot product of instances and NOT the actual space representations of the instances
- Objective function during training:

$$\max_{\alpha_1,\cdots,\alpha_N} \frac{-1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y^i y^j \left(\mathbf{x}^{i^T}\mathbf{x}^j\right) + \sum_{i=1}^{N}\alpha_i$$

- Computation during prediction:

$$\mathbf{w}^{*T}\cdot\mathbf{x} + w_0^* = \left(\sum_{i=1}^{N}\alpha_i y^i \mathbf{x}^i\right)^T \mathbf{x} + w_0^* = \sum_{i=1}^{N}\alpha_i y^i \mathbf{x}^{i^T}\mathbf{x} + w_0^*$$

SVM with Kernels (2/3)

- A Kernel function can replace the dot product as it also represents a dot product of instances in a transformed space.
- Objective function during training:

$$\max_{\alpha_1,\cdots,\alpha_N} \frac{-1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y^i y^j (K(\mathbf{x}^i,\mathbf{x}^j)) + \sum_{i=1}^{N}\alpha_i$$

- Computation during prediction:

$$\mathbf{w}^T \cdot \mathbf{x} + w_0 = \left(\sum_{i=1}^{N}\alpha_i y^i \mathbf{x}^i\right)^T \mathbf{x} = \sum_{i=1}^{N}\alpha_i y^i K(\mathbf{x}^i,\mathbf{x}^j)$$

SVM with Kernels (3/3)

- Using such Kernel functions, SVM can be used to learn a separator in a transformed space, which is usually a higher dimensional space.
- Kernel function can be designed in such a way that, we need not explicitly construct the mapped instances in the higher dimensional transformed space.
- Dot product in the transformed space can be computed efficiently using original instances only.

- Kernel functions are not limited to only SVM. Any learning algorithm which can be written only in terms of dot products of the instances, can use Kernels
  - k-Nearest Neighbours (kNN) classifier
  - Kernel perceptron algorithm

Examples of Kernel Functions

- Polynomial Kernel:

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z} + c)^p$$

- Gaussian Kernel / Radial Basis Function (RBF) Kernel:

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

- In Natural Language Processing (NLP), there are complex structures like strings, sequences, trees, etc.
  - Kernel functions are designed to compute number of common sub-structures such as substrings, subsequences, subtrees
  - Input structures are projected in a space where each sub-structure is a dimension

Examples of the String Kernel

- String Kernel: Computes the number of common subsequences shared by two strings
- Each dimension in the transformed space represents a possible subsequence of a certain length (say 2 characters)
- For any string, there will be a non-zero value for a dimension if it contains the subsequence represented by that dimension
- Sparse subsequences are penalized by the length of their spread in the string

	c-a	c-t	a-t	b-a	b-t	c-r	a-r	b-r
$\phi(\text{cat})$	λ^2	λ^3	λ^2	0	0	0	0	0
$\phi(\text{car})$	λ^2	0	0	0	0	λ^3	λ^2	0
$\phi(\text{bat})$	0	0	λ^2	λ^2	λ^3	0	0	0
$\phi(\text{bar})$	0	0	0	λ^2	0	0	λ^2	λ^3

$0 < \lambda < 1$

Lodhi, Huma, et al. "Text classification using string kernels." Journal of Machine Learning Research 2.Feb (2002): 419-444.

SVM Classifier using sklearn

Blood Transfusion Service Center Dataset

- Dataset from the UCI Machine Learning Repository
  - https://archive.ics.uci.edu/ml/datasets.php
- Number of instances = 748
- Number of attributes = 4
- **Attributes / Features**:
  - Recency - months since last donation
  - Frequency - total number of donation
  - Monetary - total blood donated in c.c.
  - Time - months since first donation
- **Class label**:
  - A binary variable representing whether he/she donated blood in March 2007
  - 1 stands for donating blood; 0 stands for not donating blood

```python
import pandas as pd
data_frame = pd.read_csv('../input/blood-transfusion-dataset/transfusion.csv')
print(data_frame.columns)
X = data_frame[['Recency (months)','Frequency (times)','Monetary (c.c. blood)','Time (months)']].to_numpy()
y = data_frame['whether he/she donated blood in March 2007']
print(X.shape)
print(y.shape)
```

```
Index(['Recency (months)', 'Frequency (times)', 'Monetary (c.c. blood)',
       'Time (months)', 'whether he/she donated blood in March 2007'],
      dtype='object')
(748, 4)
(748,)
```

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=248, random_state=10)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(500, 4)
(500,)
(248, 4)
(248,)
```

```python
from sklearn.preprocessing import StandardScaler
scaling_model = StandardScaler()
scaling_model.fit(X_train)
print(X_train[0:2])
X_train = scaling_model.transform(X_train)
print(X_train[0:2])
X_test = scaling_model.transform(X_test)
```

```
[[   2    2  500   10]
 [   2    6 1500   28]]
[[-0.91873929 -0.57755664 -0.57755664 -0.97689596]
 [-0.91873929  0.12081898  0.12081898 -0.23843941]]
```

```python
from sklearn.svm import SVC
SVM_classifier = SVC(C=1.0,kernel='linear',class_weight='balanced')
SVM_classifier.fit(X_train, y_train)
print(len(SVM_classifier.support_vectors_))
```

```
359
```

```
y_predicted = SVM_classifier.predict(X_test)
print(y_predicted)

from sklearn.metrics import classification_report
report = classification_report(y_test, y_predicted)
print(report)
```

```
[0 1 1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 1
 1 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 1 1 1 0 0 1
 1 0 0 0 1 1 1 1 0 1 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1
 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 0 1
 0 1 1 0 1 0 0 1 0 0 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1 0 1 0 0 1 1 1 1 1 0 1
 0 1 1 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1 0 0 0 1 1 0
 0 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 0]
              precision    recall  f1-score   support

           0       0.90      0.57      0.70       190
           1       0.36      0.79      0.49        58

    accuracy                           0.62       248
   macro avg       0.63      0.68      0.60       248
weighted avg       0.77      0.62      0.65       248
```

```python
SVM_classifier = SVC(C=1.0,kernel='rbf',class_weight='balanced')
SVM_classifier.fit(X_train, y_train)
print(len(SVM_classifier.support_vectors_))
```

```
361
```

```python
y_predicted = SVM_classifier.predict(X_test)
print(y_predicted)

from sklearn.metrics import classification_report
report = classification_report(y_test, y_predicted)
print(report)
```

```
[0 1 1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1
 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 1 1 1 0 0 1
 0 0 0 0 0 0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 1 1 1 0 1
 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1 1 1 0 0
 0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 1 1 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 1 1 0
 0 1 1 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1 0 0 0 1 1 0
 0 1 1 1 0 0 1 1 0 1 1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 0]
              precision    recall  f1-score   support

           0       0.90      0.64      0.74       190
           1       0.39      0.76      0.51        58

    accuracy                           0.67       248
   macro avg       0.64      0.70      0.63       248
weighted avg       0.78      0.67      0.69       248
```

```python
import pandas as pd
data_frame = pd.read_csv('../input/car-evaluation-data-set/car_evaluation.csv', header=None)
print(data_frame.columns)
X = data_frame.iloc[:,0:6].to_numpy()
y = data_frame.iloc[:,6].to_numpy()
print(X)
print(y)
```

```
Int64Index([0, 1, 2, 3, 4, 5, 6], dtype='int64')
[['vhigh' 'vhigh' '2' '2' 'small' 'low']
 ['vhigh' 'vhigh' '2' '2' 'small' 'med']
 ['vhigh' 'vhigh' '2' '2' 'small' 'high']
 ...
 ['low' 'low' '5more' 'more' 'big' 'low']
 ['low' 'low' '5more' 'more' 'big' 'med']
 ['low' 'low' '5more' 'more' 'big' 'high']]
['unacc' 'unacc' 'unacc' ... 'unacc' 'good' 'vgood']
```

```python
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelEncoderModel = LabelEncoder().fit(y)
y = labelEncoderModel.transform(y)
print(y)
oneHotEncoderModel = OneHotEncoder(sparse=False)
oneHotEncoderModel.fit(X)
X = oneHotEncoderModel.transform(X)
print(X.shape)
print(X[0:2])
```

```
[2 2 2 ... 2 1 3]
(1728, 21)
[[0. 0. 0. 1. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 1. 0.]
 [0. 0. 0. 1. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1.]]
```

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=728, random_state=10)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(1000, 21)
(1000,)
(728, 21)
(728,)
```

```python
from sklearn.svm import SVC
SVM_classifier = SVC(C=1.0,kernel='linear',class_weight='balanced')
SVM_classifier.fit(X_train, y_train)
print(len(SVM_classifier.support_vectors_))
```

```
335
```

```python
y_predicted = SVM_classifier.predict(X_test)
print(y_predicted)
```

```
[2 2 2 2 1 2 0 0 2 0 2 2 2 2 0 0 2 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 0 0 2 2
 2 2 0 2 2 2 3 2 0 2 0 2 2 2 0 1 1 1 2 0 2 0 2 2 2 2 3 2 2 0 0 2 2 3 2 2 2
 1 2 0 3 2 2 2 2 2 2 2 2 2 0 2 0 2 2 2 0 2 2 2 2 2 2 2 2 2 2 0 2 2 2 0 2 2
 2 2 2 1 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 0 2 2 3 2 2 0 2 0 3 2 2 2 2
 2 2 2 2 3 0 0 2 2 2 1 1 2 0 2 2 2 0 2 2 2 2 2 1 0 3 2 2 2 2 0 0 2 2 2 2 3
 2 1 2 1 0 0 2 2 2 0 3 1 0 2 2 2 2 0 2 0 2 2 0 2 0 3 0 1 2 2 3 2 0 1 2 0 2
 2 2 0 2 0 2 2 2 2 2 0 0 2 2 2 2 1 2 2 1 2 0 2 0 2 2 2 2 2 0 2 2 2 0 2 2 0
 2 2 2 0 2 2 2 0 2 2 0 2 2 2 1 1 2 2 2 0 0 2 0 3 3 0 0 1 2 2 0 1 2 2 1 2 2
 2 0 2 0 0 2 2 1 2 2 2 0 2 2 0 2 1 0 2 2 1 2 2 2 0 2 2 0 0 2 0 0 2 2 0 0 2
 2 2 0 2 0 2 2 2 0 2 2 2 2 2 0 2 2 1 1 3 2 2 2 2 0 2 2 2 3 2 2 1 2 1 2 0 2
 0 0 2 2 2 2 2 0 2 0 2 0 1 2 2 2 2 2 0 2 2 3 0 0 0 0 2 1 2 2 2 0 0 2 0 2 1
 0 2 2 0 2 2 2 0 3 0 0 0 2 0 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2 1 2 2 2 2 1 2 2
 2 0 2 2 2 0 2 0 2 0 2 2 1 3 2 2 2 1 3 2 2 2 2 2 2 0 2 0 2 0 2 2 2 2 0 2 2
 2 2 2 2 2 2 2 1 2 2 2 2 2 0 0 2 2 2 0 2 2 2 2 2 0 2 0 2 2 0 2 0 2 0 2 3 2
 0 0 2 2 0 0 2 2 2 2 0 2 2 0 2 2 2 2 2 1 2 2 2 2 0 1 2 0 2 2 2 0 0 2 2 2 2
 2 2 2 2 0 3 2 2 2 0 2 2 0 2 2 2 2 0 2 0 2 0 2 2 2 2 2 2 2 3 0 2 2 2 0 3 2
 2 2 2 2 2 2 2 2 2 2 2 2 0 2 1 2 0 2 0 2 2 0 2 0 2 2 2 0 1 2 2 2 2 0 2 2 0
 2 2 0 0 2 2 2 2 0 2 2 2 2 2 0 2 2 1 2 0 2 0 2 0 0 2 1 2 2 2 2 2 0 2 2 3 0
 2 2 2 2 2 1 2 2 2 2 0 2 0 0 2 2 0 2 2 0 3 2 2 0 2 2 3 2 2 0 0 0 1 2 2 2 2
 2 2 0 2 0 2 2 2 2 2 0 2 2 2 0 2 2 2 0 2 2 2 1 0 2]
```

```python
from sklearn.metrics import classification_report
report = classification_report(y_test, y_predicted)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.74      0.91      0.82       138
           1       0.49      0.88      0.63        26
           2       1.00      0.91      0.95       533
           3       0.93      0.81      0.86        31

    accuracy                           0.91       728
   macro avg       0.79      0.88      0.82       728
weighted avg       0.93      0.91      0.91       728
```

```python
SVM_classifier = SVC(C=1.0,kernel='rbf',class_weight='balanced')
SVM_classifier.fit(X_train, y_train)
print(len(SVM_classifier.support_vectors_))
y_predicted = SVM_classifier.predict(X_test)
report = classification_report(y_test, y_predicted)
print(report)
```

```
605
              precision    recall  f1-score   support

           0       0.76      0.98      0.86       138
           1       0.81      0.96      0.88        26
           2       1.00      0.92      0.96       533
           3       0.97      0.94      0.95        31

    accuracy                           0.93       728
   macro avg       0.88      0.95      0.91       728
weighted avg       0.95      0.93      0.94       728
```

Thank You!

Questions?