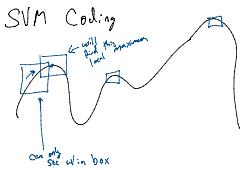
20211013 SVM example

Wednesday, October 13, 2021 2:38 PM



- Midterm is Friday - Monday (10 days)
- Look at data, try to get some intuition, create some new and interesting features
- Singular value decomposition
- Will do well on the midterm if you find features that other people don't
- And the tuning parameters there of
- These are super powerful tools

- There is a way of generating multiple classifiers and then aggregating it into one mega classifier

- Midterm is same for both classes
- 80% of grade on report
- 20% proportional to the score you get
  - Not necessarily your rank on the leader board
- Want to reward high level understanding of the tools and methods
- Have a good workflow and process
- Want every step to be well understood
- Each step should have some insight, and based on that you do the next step

- Is there every a reason not to use SVD?
  - If you don't have a lot of features to begin with

- Deliverable 0 tonight
- Midterm Friday, start ASAP

Questions I have:

- Are midterm instructions posted?
- Will deliverable 1 be due during the midterm?
- Do we need to have a particular python version installed to use the provided start code?

File - C:\Users\Wolfs\PycharmProjects\cs506lec20211013\facial_recog.py

```
import numpy as np
from PIL import Image
import seaborn as sns
from sklearn.svm import SVC
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix
from sklearn.datasets import fetch_lfw_people
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import GridSearchCV,
train_test_split

# There are lots of datasets in sklearn that you would
 need to download
# Preprocess data to improve performance
# General data science flow for midterm

# THis requires an older version of sklearn??
# It looks like he is using Python 3.7.4

sns.set()

# Get face data
faces = fetch_lfw_people(min_faces_per_person=60)
print(faces.target_names)

# plot face data
fig, ax = plt.subplots(3, 5)
for i, axi in enumerate(ax.flat):
    axi.imshow(faces.images[i], cmap='bone')
    axi.set(xticks=[], yticks=[],
            xlabel=faces.target_names[faces.target[i
]])
plt.show()

# split train test set
Xtrain, Xtest, ytrain, ytest = train_test_split(faces.
data, faces.target, random_state=42)

# blindly fit svm
```


File - C:\Users\Wolfs\PycharmProjects\cs506lec20211013\facial_recog.py

```
svc = SVC(kernel='rbf', class_weight='balanced', C=10
, gamma=0.001)

"""
#fit model
model = svc.fit(Xtrain, ytrain)
yfit = model.predict(Xtest)

fig, ax = plt.subplots(6, 6)
for i, axi in enumerate(ax.flat):
    axi.imshow(Xtest[i].reshape(62, 47), cmap='bone')
    axi.set(xticks=[], yticks=[])
    axi.set_ylabel(faces.target_names[yfit[i]].split
()[-1],
                   color='black' if yfit[i] == ytest[i
] else 'red')
fig.suptitle('Predicted Names; Incorrect Labels in Red
', size=14)
plt.show()

mat = confusion_matrix(ytest, yfit)
print(mat)
sns.heatmap(mat.T, square=True, annot=True, fmt='d',
cbar=False,
            xticklabels=faces.target_names,
            yticklabels=faces.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label')
plt.show()
"""

# Model just learned to predict bush for facial
recognition
# look at confusion matrix to see what was predicted
vs what it wanted

############################
############################
############################

# look at singular values
```

```
_, s, _ = np.linalg.svd(Xtrain, full_matrices=False)
plt.plot(range(1,len(s)+1),s)
plt.title("Singular Values")
plt.show()

# extract principal components
pca = PCA(n_components=150, whiten=True)
svc = SVC(kernel='rbf', class_weight='balanced')
# Make pipeline is worth looking up and see how it ties toge
svcpca = make_pipeline(pca, svc)

param_grid = {'svc__C': [1, 5, 10, 50],
              'svc__gamma': [0.0001, 0.0005, 0.001, 0.005]}
grid = GridSearchCV(svcpca, param_grid)

# fit model
grid.fit(Xtrain, ytrain)
print(grid.best_params_)

model = grid.best_estimator_
yfit = model.predict(Xtest)

fig, ax = plt.subplots(6, 6)
for i, axi in enumerate(ax.flat):
    axi.imshow(Xtest[i].reshape(62, 47), cmap='bone')
    axi.set(xticks=[], yticks=[])
    axi.set_ylabel(faces.target_names[yfit[i]].split()[-1],
                   color='black' if yfit[i] == ytest[i] else 'red')
fig.suptitle('Predicted Names; Incorrect Labels in Red', size=14)
plt.show()

mat = confusion_matrix(ytest, yfit)
print(mat)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=faces.target_names,
```



```
            yticklabels=faces.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label')
plt.show()


# # optimal weights from before
# pca = PCA(n_components=20, whiten=True)
# svc = SVC(kernel='rbf', class_weight='balanced', C=5, gamma=0.001)
# svcpca = make_pipeline(pca, svc)
# bagging = BaggingClassifier(svcpca, n_estimators=100)
# model = bagging.fit(Xtrain, ytrain)
# yfit = model.predict(Xtest)

# fig, ax = plt.subplots(6, 6)
# for i, axi in enumerate(ax.flat):
#     axi.imshow(Xtest[i].reshape(62, 47), cmap='bone')
#     axi.set(xticks=[], yticks=[])
#     axi.set_ylabel(faces.target_names[yfit[i]].split()[-1],
#                    color='black' if yfit[i] == ytest[i] else 'red')
# fig.suptitle('Predicted Names; Incorrect Labels in Red', size=10)
# plt.show()

# mat = confusion_matrix(ytest, yfit)
# print(mat)
# sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
#             xticklabels=faces.target_names,
#             yticklabels=faces.target_names)
# plt.xlabel('true label')
# plt.ylabel('predicted label')
# plt.show()

```

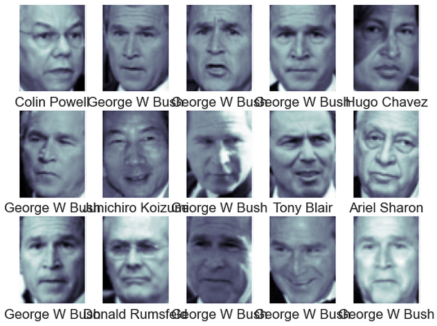
File - C:\Users\Wolfs\PycharmProjects\cs506lec20211013\requirements.txt

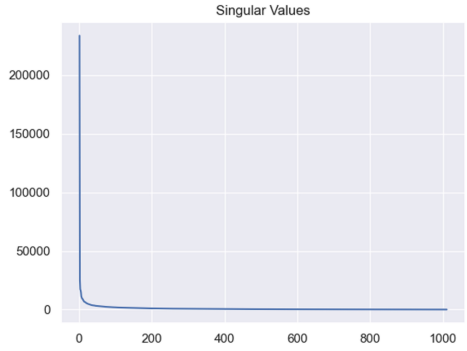
```
cycler==0.10.0
joblib==0.14.0
kiwisolver==1.1.0
matplotlib==3.1.1
numpy==1.17.3
pandas==0.25.3
Pillow==6.2.1
pyparsing==2.4.4
python-dateutil==2.8.1
pytz==2019.3
scikit-learn==0.21.3
scipy==1.3.1
seaborn==0.9.0
six==1.13.0
sklearn==0.0
```

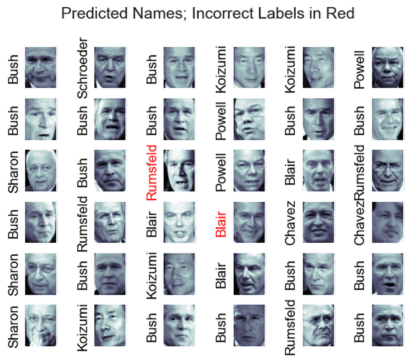


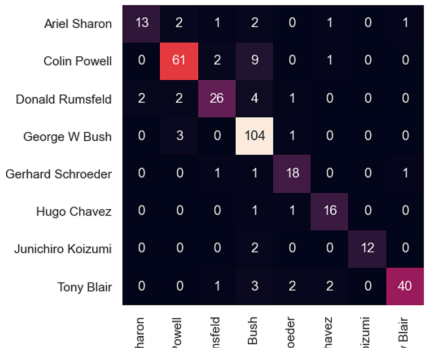
```
C:\Users\Wolfs\anaconda3\envs\cs506lec20211013\python.exe C:/Users/Wolfs/PycharmProjects/cs506lec20211013/facial_recog.py
['Ariel Sharon' 'Colin Powell' 'Donald Rumsfeld' 'George W Bush'
 'Gerhard Schroeder' 'Hugo Chavez' 'Junichiro Koizumi' 'Tony Blair']
{'svc__C': 5, 'svc__gamma': 0.001}
[[ 13   0   2   0   0   0   0   0]
 [  2  61   2   3   0   0   0   0]
 [  1   2  26   0   1   0   0   1]
 [  2   9   4 104   1   1   2   3]
 [  0   0   1   1  18   1   0   2]
 [  1   1   0   0   0  16   0   2]
 [  0   0   0   0   0   0  12   0]
 [  1   0   0   0   1   0   0  40]]

Process finished with exit code 0
```




Singular Values
200000
150000
100000
50000
0
0
200
400
600
800
1000


Predicted Names; Incorrect Labels in Red
Bush
Schroeder
Bush
Koizumi
Koizumi
Powell
Bush
Bush
Bush
Powell
Bush
Bush
Sharon
Bush
Rumsfeld
Powell
Blair
Rumsfeld
Bush
Rumsfeld
Blair
Blair
Chavez
Chavez
Sharon
Bush
Koizumi
Blair
Bush
Bush
Sharon
Koizumi
Bush
Bush
Rumsfeld
Bush


Ariel Sharon 13 2 1 2 0 1 0 1
Colin Powell 0 61 2 9 0 1 0 0
Donald Rumsfeld 2 2 26 4 1 0 0 0
George W Bush 0 3 0 104 1 0 0 0
Gerhard Schroeder 0 0 1 1 18 0 0 1
Hugo Chavez 0 0 0 1 1 16 0 0
Junichiro Koizumi 0 0 0 2 0 0 12 0
Tony Blair 0 0 1 3 2 2 0 40
haron
owell
nsfeld
Bush
oeder
havez
izumi
Blair

20211013 Naïve Bayes and SVM

Wednesday, October 13, 2021 4:57 PM

Naïve Bayes and SVM

Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:

$$P(C \mid A) = \frac{P(A,C)}{P(A)}$$

$$P(A \mid C) = \frac{P(A,C)}{P(C)}$$

- Bayes theorem:

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20

- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

- Probability of stiff neck with meningitis is high, but if you have a stiff neck that doesn't really mean high probability of having meningitis

Bayesian Classifiers

- Consider each attribute and class label as random variables

- Given a record with attributes (A_1, A_2,…,A_n)
  - Goal is to predict class C
  - Specifically, we want to find the value of C that maximizes P(C| A_1, A_2,…,A_n)

- Can we estimate P(C| A_1, A_2,…,A_n) directly from data?

- Have to know the probability of A1, A2,… and so on
- If they are independent that is a little easier
- Need a prior probability of being in C as well

Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximizes
    $P(C \mid A_1, A_2, \ldots, A_n)$

  - Equivalent to choosing value of C that maximizes
    $P(A_1, A_2, \ldots, A_n \mid C)\, P(C)$

- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?

Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n \mid C) = P(A_1 \mid C_j)\, P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$
  - Can estimate $P(A_i \mid C_j)$ for all A_i and C_j.
  - New point is classified to C_j if $P(C_j) \prod P(A_i \mid C_j)$ is maximal.

- e.g. height, weight, bmi
  - BMI is a function of height and weight
  - These are NOT independent
- Assuming independence is a BIG assumption
- If a single one of these is 0 then you loose a lot of information (multiplying all the probabilities together)

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(C) = N_c/N$
  - e.g., P(No) = 7/10,
    P(Yes) = 3/10

- For discrete attributes:

$$P(A_i \mid C_k) = |A_{ik}| / N_c$$

  - where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
  - Examples:

    P(Status=Married|No) = 4/7
    P(Refund=Yes|Yes)=0

How to Estimate Probabilities from Data?

- For continuous attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
    - violates independence assumption
  - Two-way split: $(A < v)$ or $(A > v)$
    - choose only one of the two splits as new attribute
  - Probability density estimation:
    - Assume attribute follows a normal distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

k

-

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i, c_i) pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

-

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7

- P(X|Class=No) = P(Refund=No|Class=No)
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)

P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 0 × 1.2 × 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)

=> Class = No

- =0 is an issue

Naïve Bayes Classifier

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m - estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of classes

p: prior probability

m: parameter

- Look at 1+ that so that it is never 0
- add c based on how many features you have so that you have a proper probability
- A probability of 0 is you don't have a particular

example in your dataset
    - But this could be because you don't have enough data
- If you have A LOT of data and you still have a 0, then the 0 might be real valid estimate
- Can use these different transforms in this case, normalizing so they add up to proper probabilities

Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A \mid M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A \mid N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A \mid M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A \mid N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

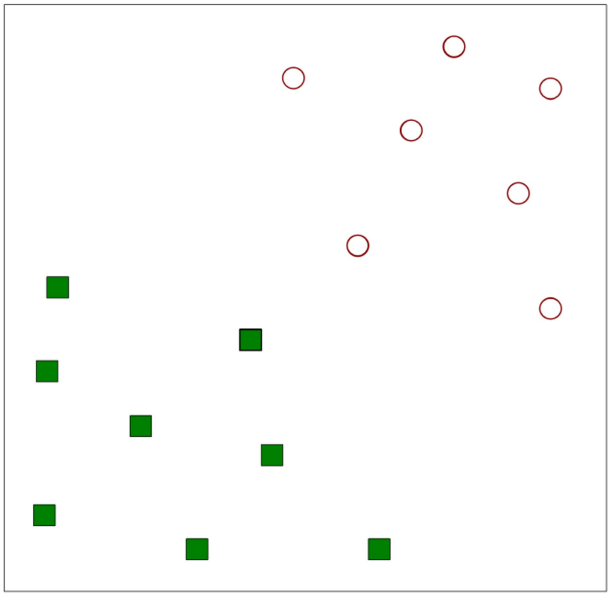
- What is the probability it's mammal given these attributes
- What is the probability it's NOT mammal given these attributes
- Assign to whatever has the higher probability
- Assume independence to make the problem easier

Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)

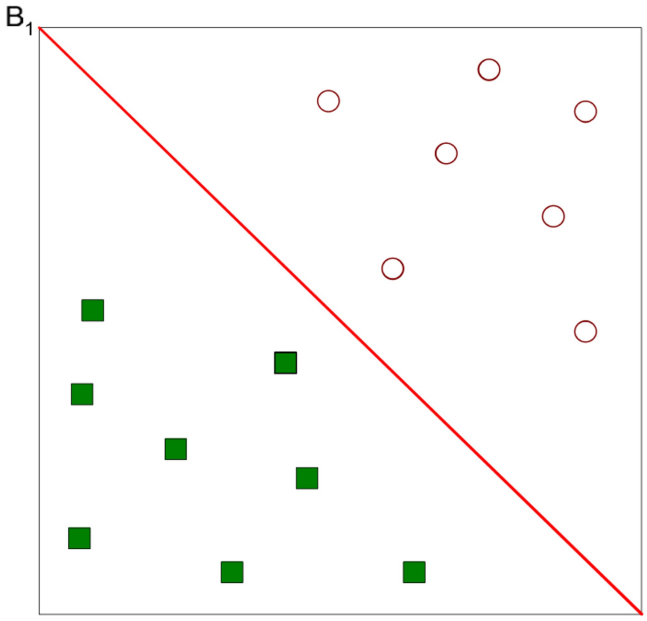
- Called Naïve because it's assuming independence

Support Vector Machines



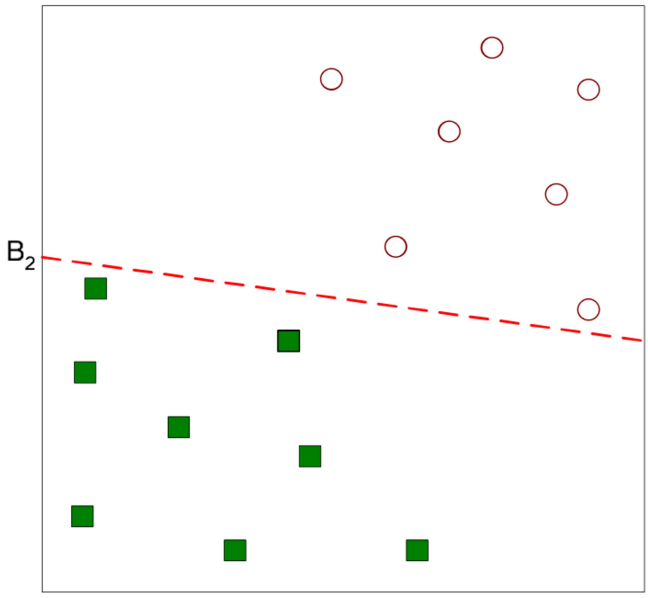
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



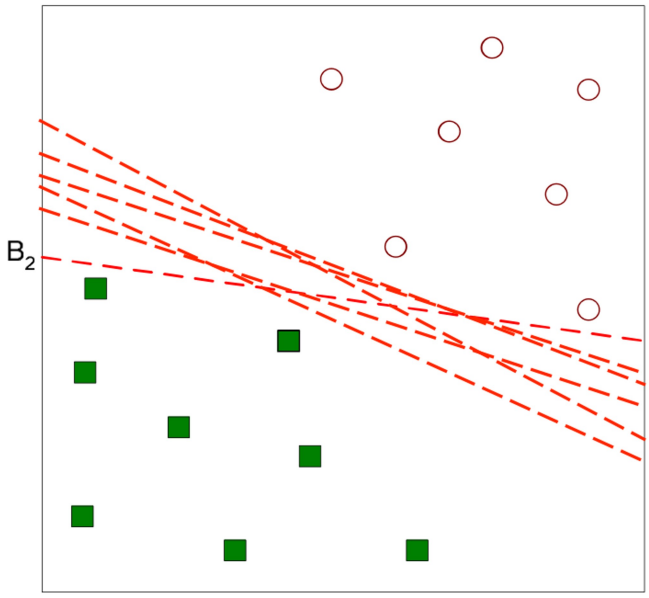
- One Possible Solution

Support Vector Machines



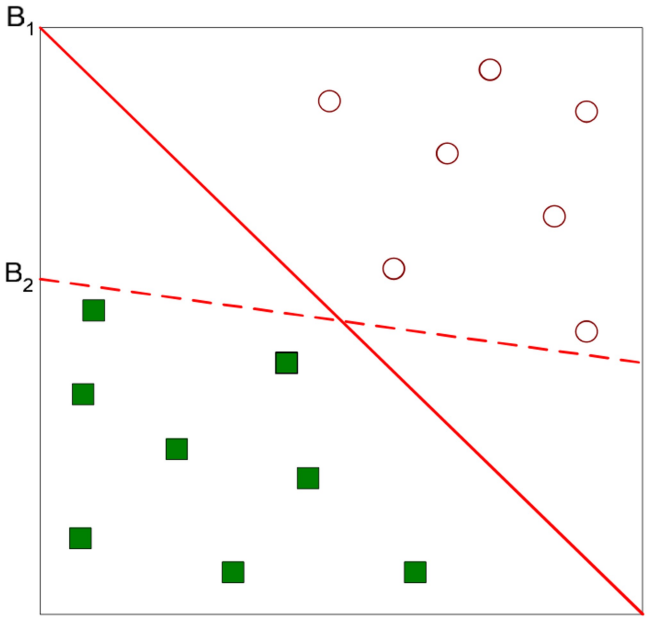
- Another possible solution

Support Vector Machines



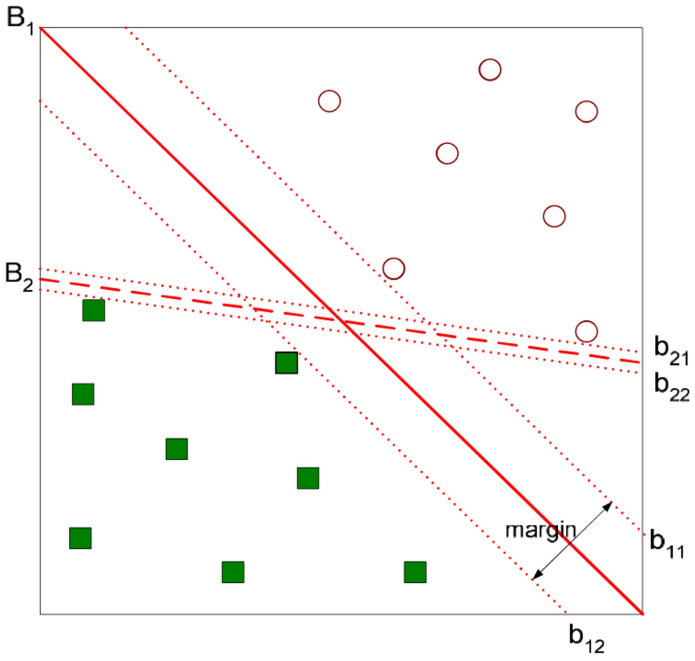
- Other possible solutions

Support Vector Machines



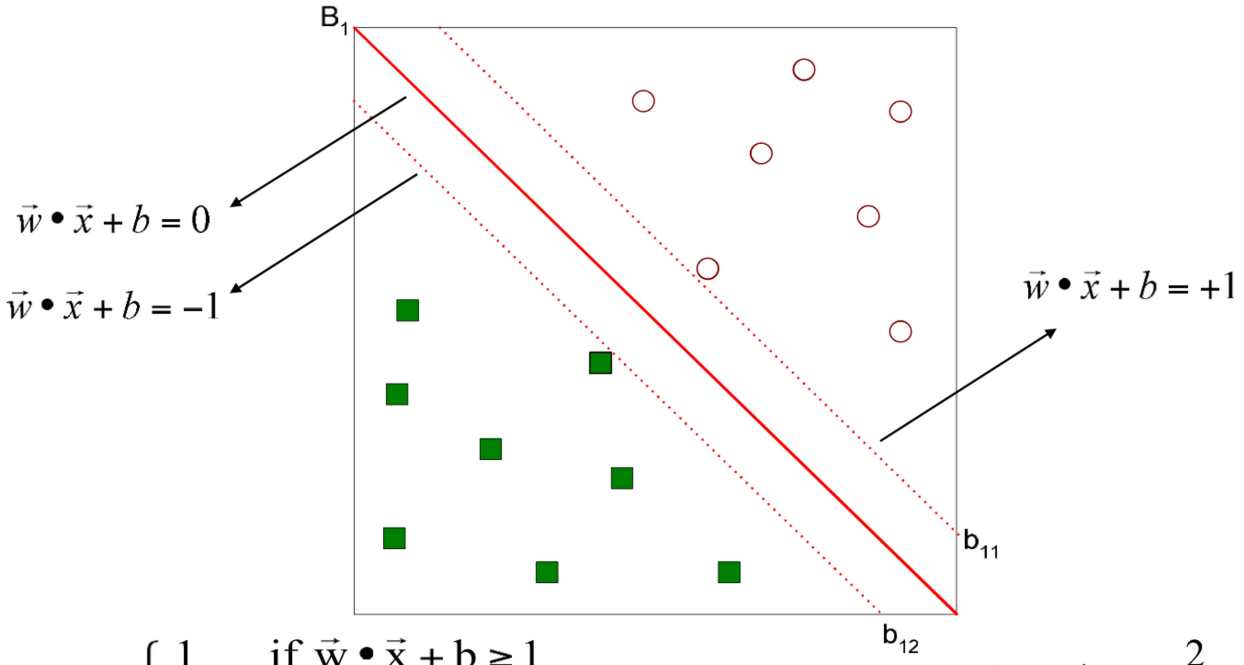
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines



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

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

Support Vector Machines

- We want to maximize:

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

  - Which is equivalent to minimizing:

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

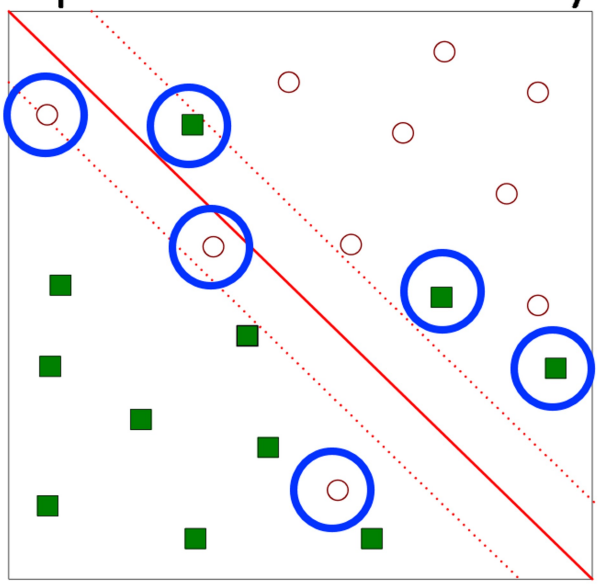
  - But subjected to the following constraints:

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

    - This is a constrained optimization problem
      - Numerical approaches to solve it (e.g., quadratic programming)

Support Vector Machines

- What if the problem is not linearly separable?



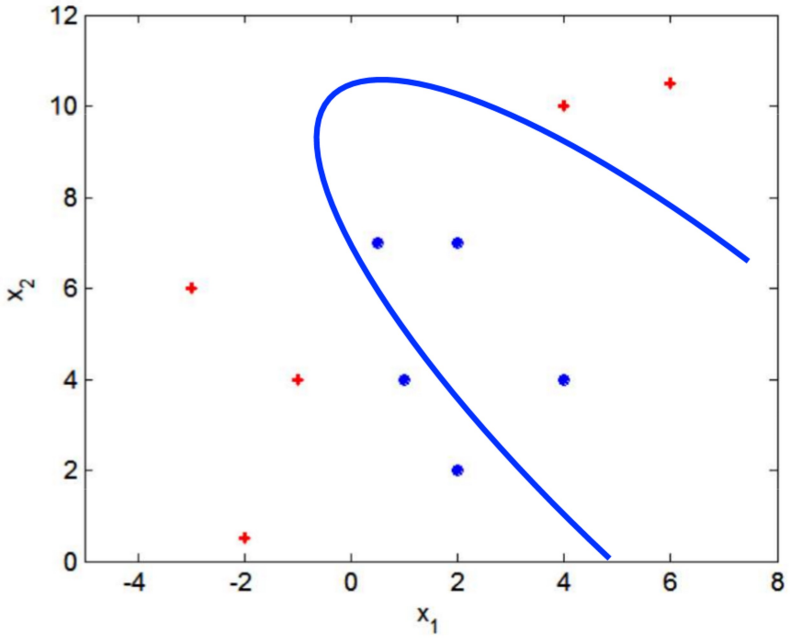
Support Vector Machines

- What if the problem is not linearly separable?
  - Introduce slack variables
    - Need to minimize: $L(w) = \frac{\| \vec{w} \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k \right)$
    - Subject to:

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

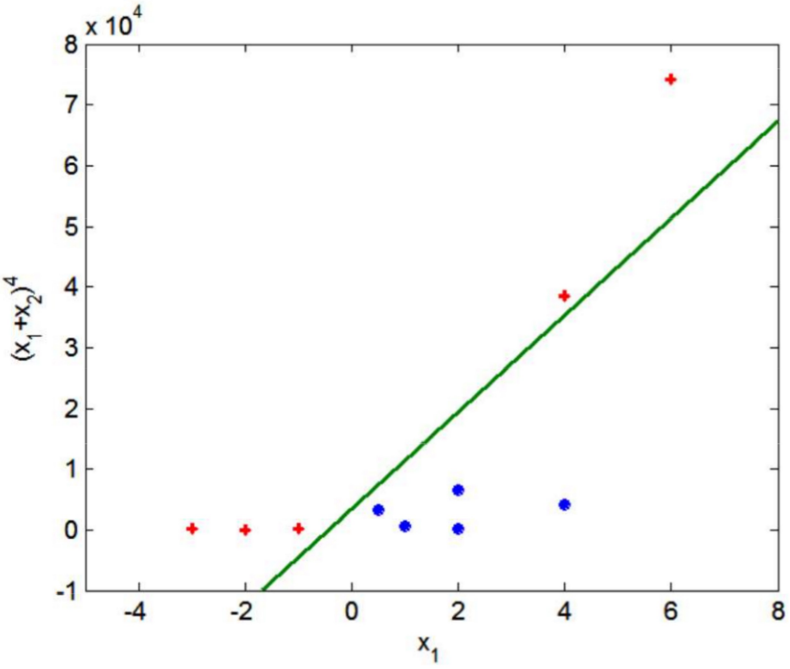
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Nonlinear Support Vector Machines

- Transform data into higher dimensional space



Kernel Trick!

- We do not need to actually map explicitly each point to a high dimensional space!!
- We just need to have a function that computes the similarity (distance) in the mapped space given the points in the input space (without the need to do the mapping!!!)
- Kernel function!!! (unfortunate term☹)

Kernel function

- Given x and y two vectors in R^d assume that we want to map them in a higher dimensional space R^n using a function ϕ
- We can define a function K(x,y): R^d x $R^d \rightarrow R$ that computes directly the distance of $\phi(x)$ and $\phi(y)$! No need to compute the actual ϕ!!
- Actually, the Kernel function K should give the same results as the inner (dot) product between of $\phi(x)$ and $\phi(y)$ (simulates a dot product in the "mapped" space)

$$K(x,y) = \langle \phi(x), \phi(y) \rangle_n$$

SVM with Kernels

- Using the kernel trick, we can compute the linear separator in the mapped space without the need to do the actual mapping!! just use the Kernel function when you need to compute the inner product between two “mapped” vectors.

Kernel options

- Many different ways to define kernels
- Most popular kernels:
  - Linear: $K(x,y) = (x \bullet y)$
  - Polynomial: $K(x,y) = (x \bullet y)^d$ or $K(x,y) = (x \bullet y+1)^d$
  - RBF: $K(x,y) = \exp(-\gamma ||x-y||^2)$

In practice, RBF works well in most cases (adds small bumps around the "mapped" points)

20211013 Support Vector Machines

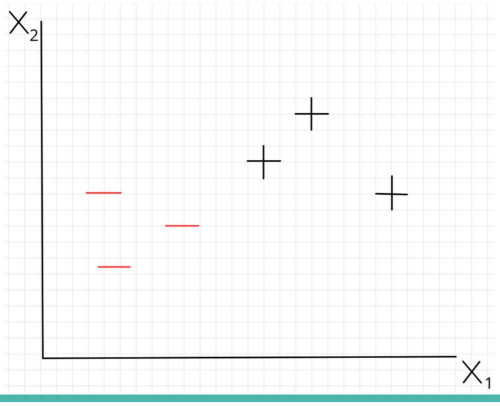
Wednesday, October 13, 2021 4:30 PM

- If you don't have access to GIT
  - Keep harassing everyone with cc'ing to address the issue

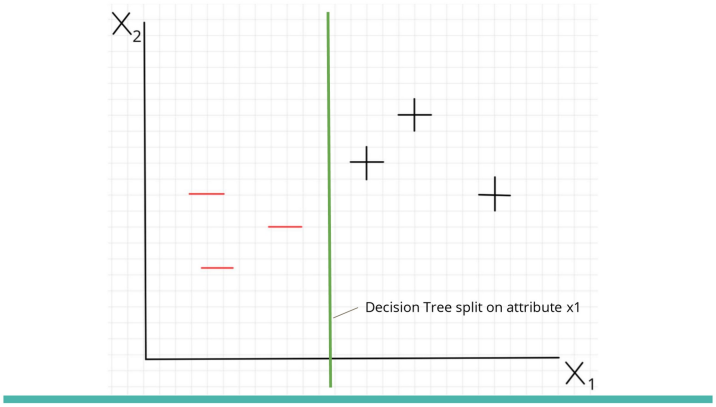
Support Vector Machines

Boston University CS 506 - Lance Galletti

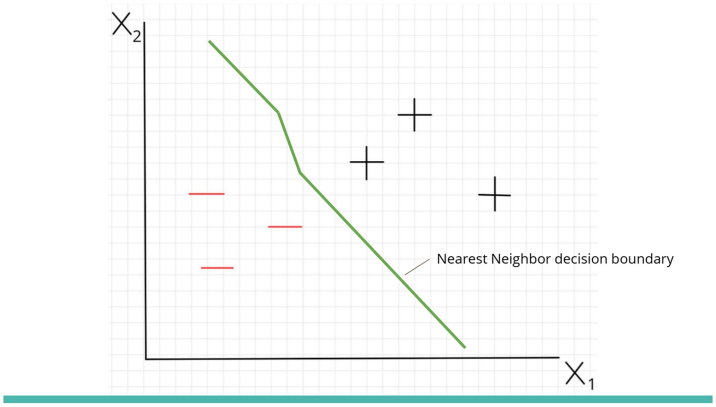
- Coming back to the example we had at the beginning of class

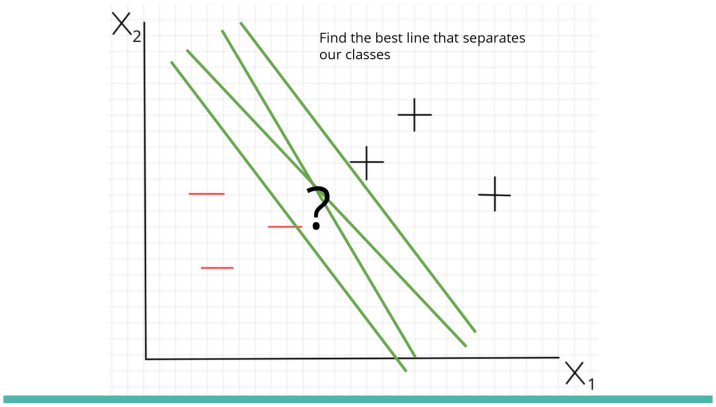


- We've seen a lot of classification techniques in the last day



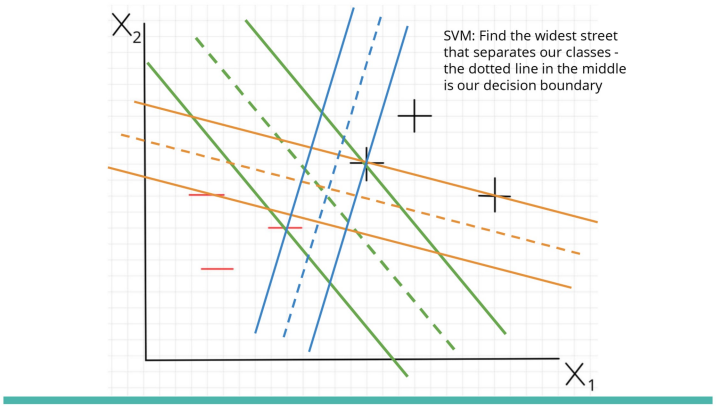
-



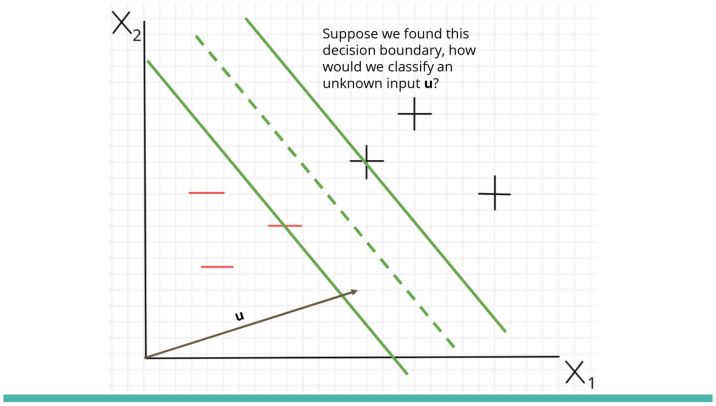


- There are a lot of possible lines, so what do we

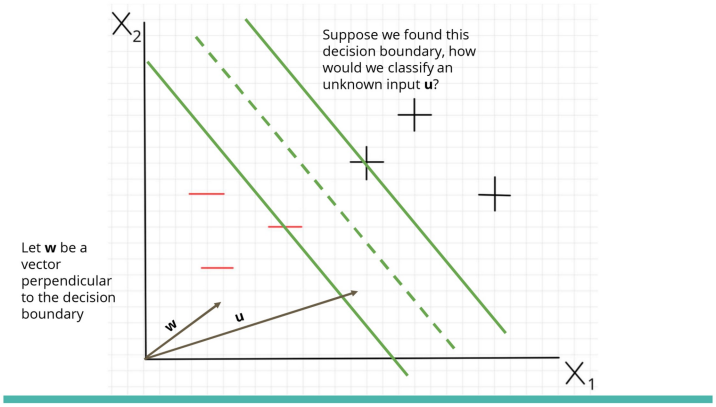
mean by the best line that separates our data?



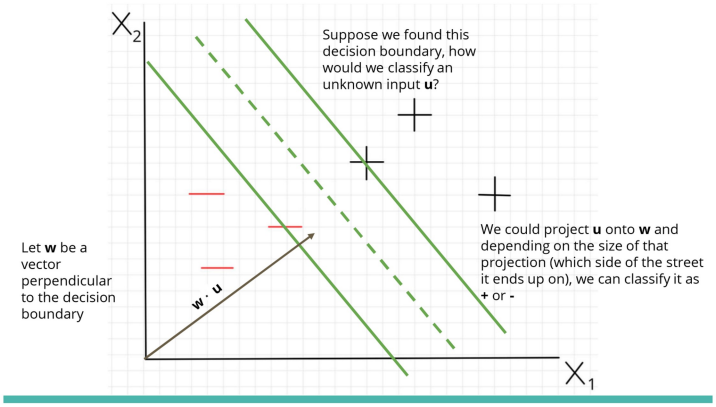
- SVM: tries to maximize the separation between the classes
- In this case, it will be the green one
- How do we find this street?

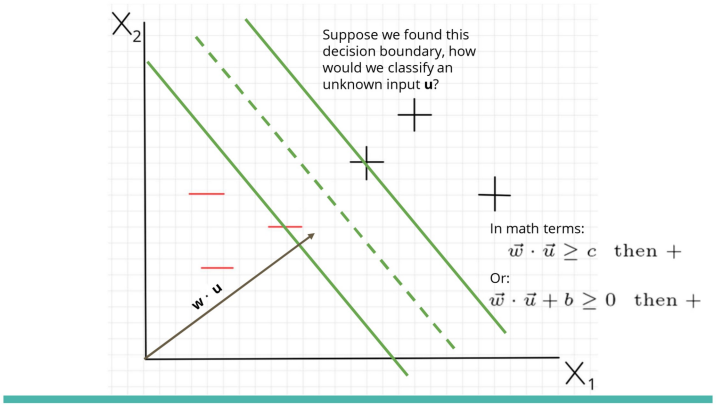


- Once we have the widest street, how do you classify unseen record
- What side of the street is u on?
  - Find the vector that is perpendicular of decision boundary
  -



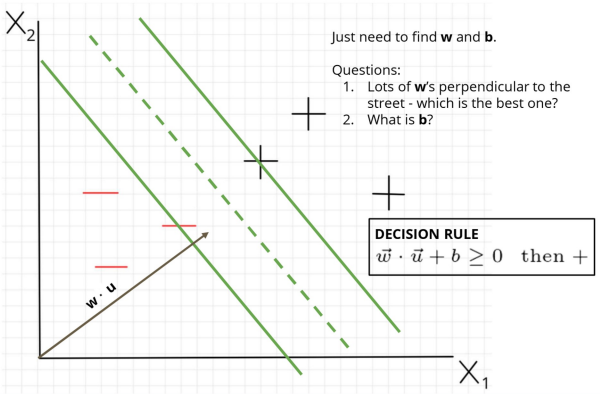
- w * u (dot)





Why does it have to perpendicular to the street?

- bias term "b"



Just need to find **w** and **b**.

Questions:

1. Lots of **w**'s perpendicular to the street - which is the best one?
2. What is **b**?

DECISION RULE

$\vec{w} \cdot \vec{u} + b \geq 0 \quad \text{then} +$

- We can fully characterize this street if we have the right w and b

How to find the widest street

We want our samples to lie beyond the street. That is:

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

Note: for an unknown **u**, we can have

$$-1 < \vec{w} \cdot \vec{u} + b < 1$$

- Whatever w we find must satisfy this condition
- We don't WANT any examples to be inside the street
- Want some space between positive and negative classes
- -1 & 1 is just arbitrary
- Want to find w and b that maximizes the width of that street
- Unknowns can fall inside the street later
- This is pretty complicated

How to find the widest street

Let's introduce a variable

$$y_i = \begin{cases} +1 & \text{if } x_i \text{ is a } + \text{sample} \\ -1 & \text{if } x_i \text{ is a } - \text{sample} \end{cases}$$

How to find the widest street

Let's introduce a variable

$$y_i = \begin{cases} +1 & \text{if } x_i \text{ is a } + \text{sample} \\ -1 & \text{if } x_i \text{ is a } - \text{sample} \end{cases}$$

Note: this is effectively the class label of x_i

How to find the widest street

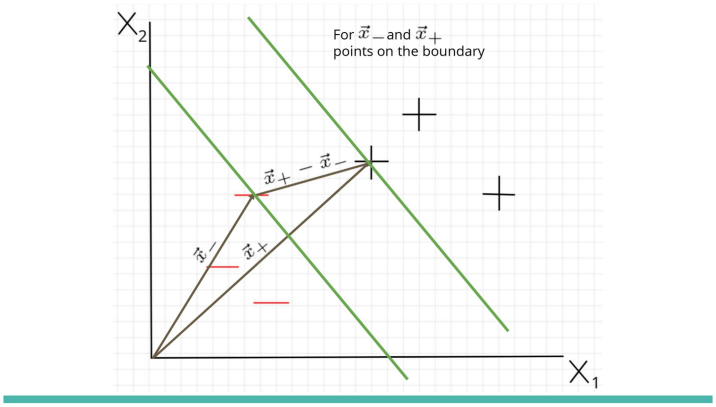
If we multiply our sample decision rules by this new variable:

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$$

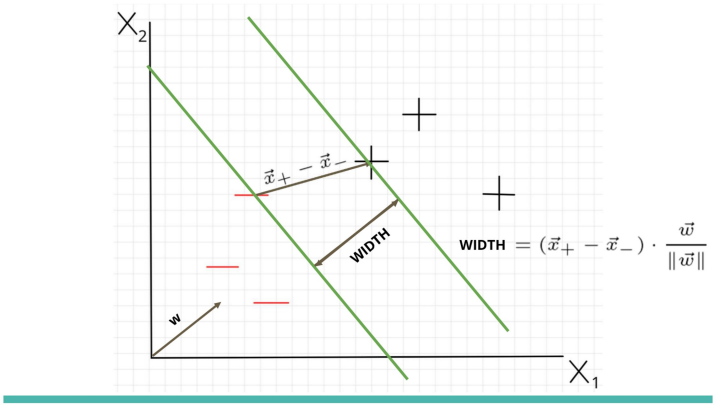
Meaning, for x_i on the decision boundary, we want:

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

- If we multiply both by yi, then we get these equations
- Now we have both constraints inside a single constraint
- Now we have equation of what to do with points EXACTLY on the side of the street



- Now we need to find the width of the street
- X+ minus X-



$$\text{WIDTH} = (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

- Now have the width of the street
- Two points can fully characterize this?

Are there multiple possible solutions? (especially with a small number of points)

How to find the widest street

We know that **WIDTH** $= (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$ for $\vec{x}_-$ and $\vec{x}_+$ points on the boundary

And, since they are on the boundary, we know that

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

Hence, **WIDTH** $= \frac{2}{\|\vec{w}\|}$

(as an exercise, try to show this)

How to find the widest street

Goal is to maximize the width

$$\max(\frac{2}{\|\vec{w}\|}) = \min(\|\vec{w}\|)$$
$$= \min(\frac{1}{2}\|\vec{w}\|^2)$$

Subject to:

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

How to find the widest street

Can use Lagrange multipliers to form a single expression to find the extremum of

$$L = \frac{1}{2}\|\vec{w}\|^2 - \sum_i \alpha_i [y_i(\vec{x}_i \cdot \vec{w} + b) - 1]$$

where α_i is 0 for x_i not on the boundary.

Now we can take derivatives to find the extremum of **L**.

- Lagrange multiplier to min-max

- Take derivates of L and partial derivative with respect to w and b

How to find the widest street

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_i \alpha_i y_i \vec{x}_i = 0$$

$$\implies \vec{w} = \sum_i \alpha_i y_i \vec{x}_i$$

Means **w** is a linear sum of vectors in our sample/training set!

$$\frac{\partial L}{\partial b} = -\sum_i \alpha_i y_i = 0$$

$$\implies \sum_i \alpha_i y_i = 0$$

- w just a linear combination of things in the dataset, points on the boundary between the two classes

How to find the widest street

Let's plug these values back into L to see what happens to L at its extremum

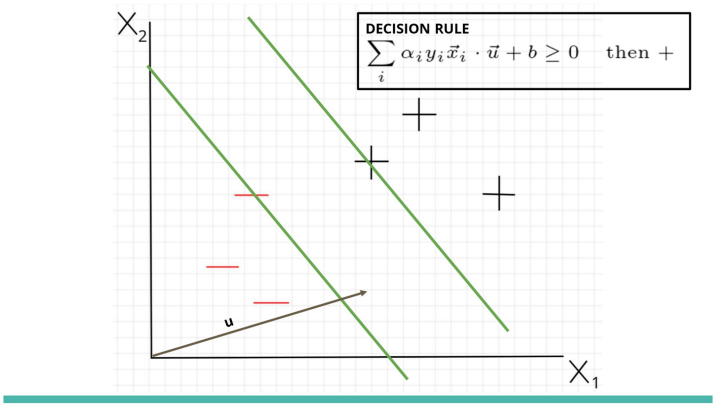
$$L = \frac{1}{2}\left(\sum_i \alpha_i y_i \vec{x}_i\right) \cdot \left(\sum_i \alpha_i y_i \vec{x}_i\right) - \left(\sum_i \alpha_i y_i \vec{x}_i\right) \cdot \left(\sum_i \alpha_i y_i \vec{x}_i\right) - \sum_i \alpha_i y_i b + \sum_i \alpha_i$$

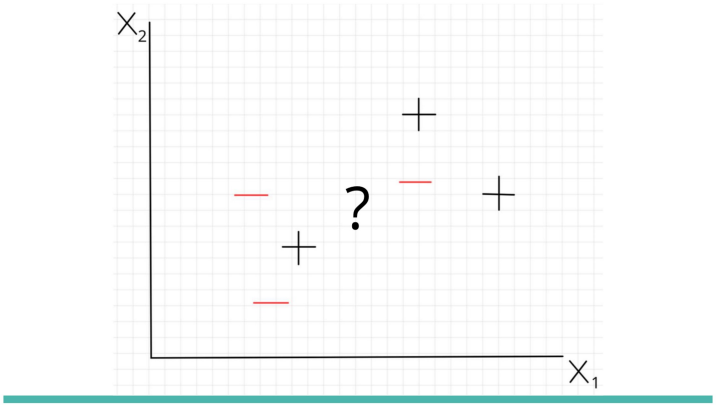
Simplifying, we get:

$$L = \sum_i \alpha_i - \frac{1}{2}\left(\sum_i \alpha_i y_i \vec{x}_i\right) \cdot \left(\sum_i \alpha_i y_i \vec{x}_i\right)$$

$$= \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$$

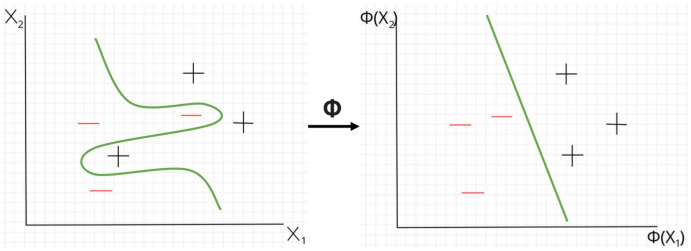
- What happens to this optimization function as we get to the extreme?
- Linear combination of points in this dataset
- Fully characterized by inner products of points in the dataset





- What if our data is not linearly separable?
- DO TRANSFORMATION PHI and then do this
- There are a bunch of transformations that could achieve linearly separable
  - infinite many, so lets define a function

When stuck - change perspective



But how to find Φ?

Turns out we don't need to find or define a transformation Φ!

Looking back at L, since it depends only on the dot product of our input, we only need to define the dot product in our transformed space.

i.e. we only need to define

$$K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$$

Called a Kernel function. This is often referred to as the "kernel trick".

- You don't necessarily need to find the transform, you just need to know what the inner product looks like in that space
- Just need to define the inner product for a potential candidate for the space

Example Kernel Functions

$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^n$$

$$K(\vec{x}_i, \vec{x}_j) = e^{\frac{\|\vec{x}_i - \vec{x}_j\|}{\sigma}}$$

DEMO