Support Vector Machine

S M T W T F S DATE:_____

- Support Vector Machine are one of the more complex algorithms we will learn, but it all begins with a simple premise:
  - Does a hyperplane exist that can effectively seperate classes?
- To finrlt understand this question, we first need to go through the history and development of support ~~support~~ vector machine!

Section Overview

- History
- Intuition and Theory for SVM
- SVM classification Examples
- SVM Regression Example
- SVM project Exercise and Solution

2. History of Support Vector Machines

- SVM will be one of the most recently developed machine learning algorithms we will learn about!
- Lets ~~bsei~~ briefly review the recent history behind the development of support vector Machines.

MIDTERMS DATE:

Quick Notes:-

- 1960s: Vladmir Vapnik obtained his PhD in statistic at the institute of Control sciences in Moscow.
  - worked at the Institute from 1961-1990.

- 1960s: Alex chervonenkis and Vladimir Vapnik begin the development of support Vector Machines

- 1963: published "Generalized portrait Algorithm" for Image analysis by computers.

(SVM in image analysis -

- 1964: Aizerman, Braverman, and Rozonoer, publish "Theoritical foundation of the potential Function Method" in pattern Recognition learning.

- Introduces geometrical Interpretation of the Kernels as inner products in a feature spaces.

- 1974 - Vapnik and Chervonenkis continue work publishing "Theory of pattern Recognition"

M T W T F S DATE:

- 1960s-1990s - Vapnik continues to work further develop SVM
- 1992 - Bernhard Boser, Isabelle Guyon and Vladimir Vapnik suggested a way to create non-linear classifiers by applying the kernal trick to maximize-margin hyperplane.
- 1995: Corinna cortes and Vladimir publish SVM incarnation utilizing a soft margin.
- 1996: Vladmir Vapnik, Harris Drucker, Christopher Burges, Linda Kaufman and Alexander Smola publish "Support vector Regression Machines", expanding SVM beyound classification tasks.

- 30 years of work to get Support Vector machine we will be using today

as we move
We will see advance clearly from maximum margin classifiers, to support vector classifiers, and finally support Vector Machines.

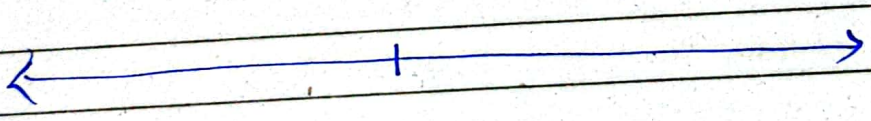
3. SVM - Hyperplane and Margin

MINDS DATE:

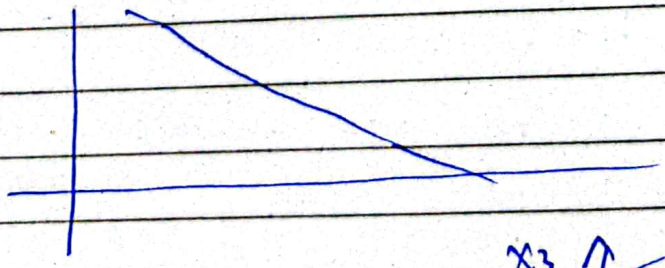
- We will slowly build up to SVMs:
  - Maximize Margin classifier
  - Support vector classifier
  - Support Vector Machine
- Let begins by understanding what a hyperplane is.

- In a N-dimensional space, a hyperplane is flat affine subspace of hyperplane dimension N-1
- for example:
  - 1-D Hyperplane is a single point
  - 2-D Hyperplane is a line
  - 3-D Hyperplane is a flat line.

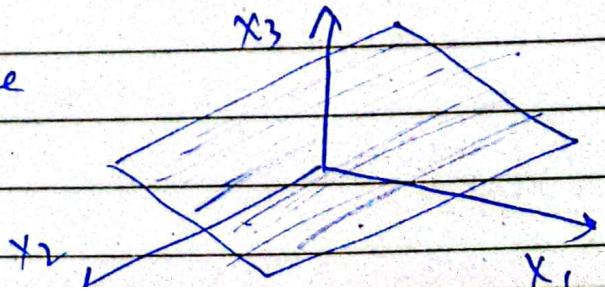
- 1-D Hyperplane



- 2-D Hyperplane

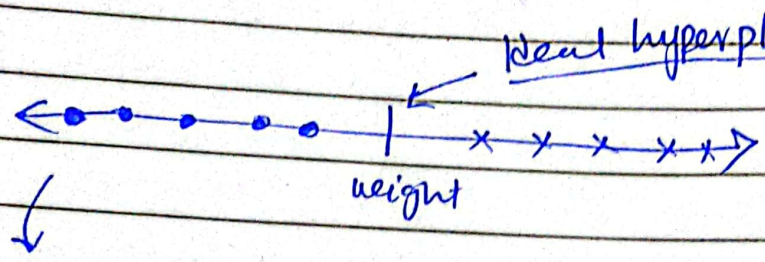


- 3-D Hyperplane



M T W T F S DATE:

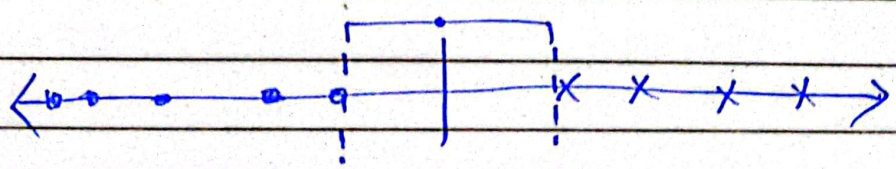
- The main idea behind SVM is that we can use hyperplane to create a seperation between classes.
- Then new points will fall on one side of this seperating hyperplane, which we can then use to assign a class.



× Male
○ Female

Classes are perfectly seperably

- But for above case there could be many seperation how to choose best fit line?
- We could choose seperator that maximizes the margin between the classes.

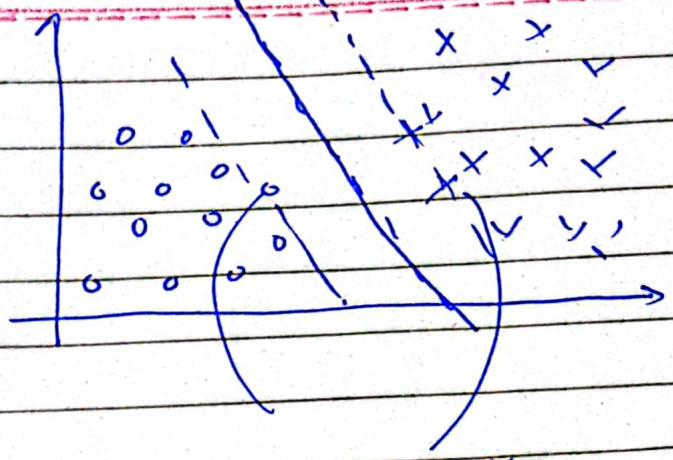


This is know as maximum marginal classifier.

- Same ideal is implemented for n-dimension.

• Multiple possible margins
Chose the maximum margin

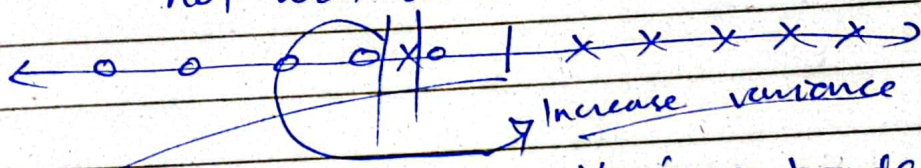
M T W T F S DATE:



Delta points at the margin "support" separator.

• What happen if the classes are not perfectly seperable?

not will able to classify without margin of error

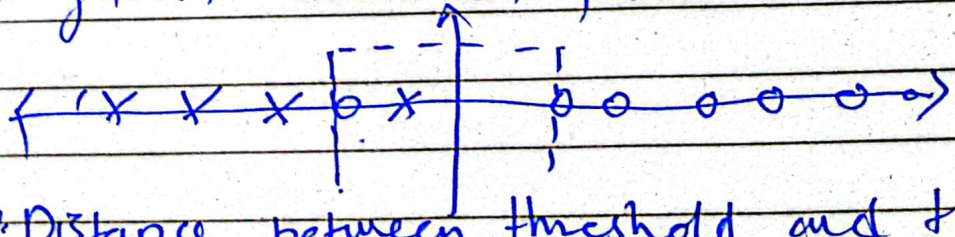


Increase variance

We will have a bias-Variance tradeoff depending where we place this seperator.

mean increase biase to lead to better long term result on future

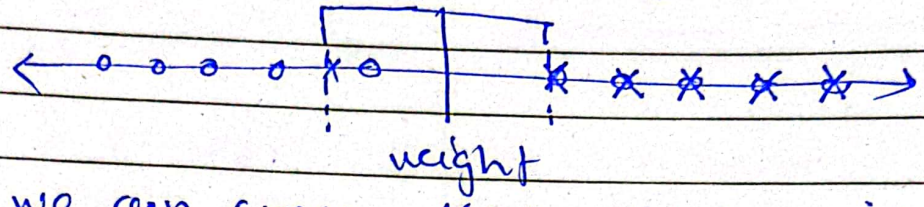
• Here we allow more bias to lead to better long term result on future data.



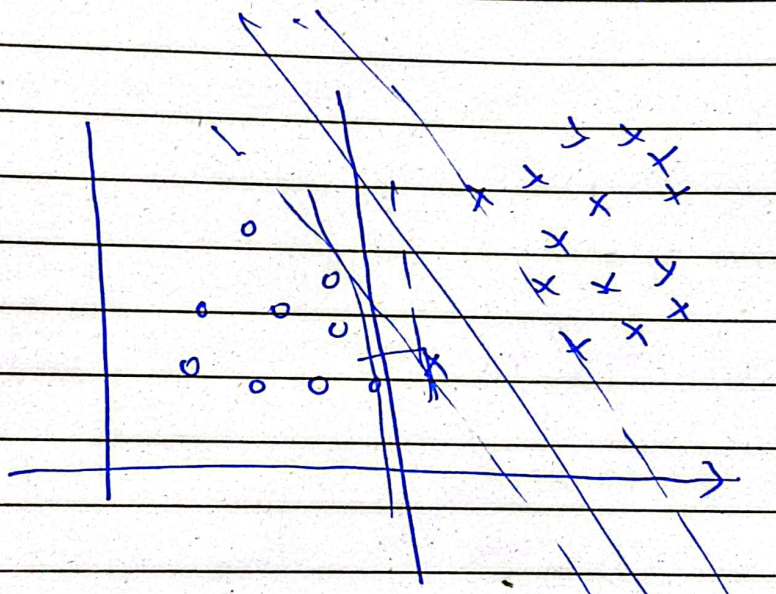
• Distance between threshold and the observation is a soft margin:

M T W T F S DATE:_____

- Soft margin allows for misclassification inside the margin.
- There are many possible threshold splits if we allow for soft margins.



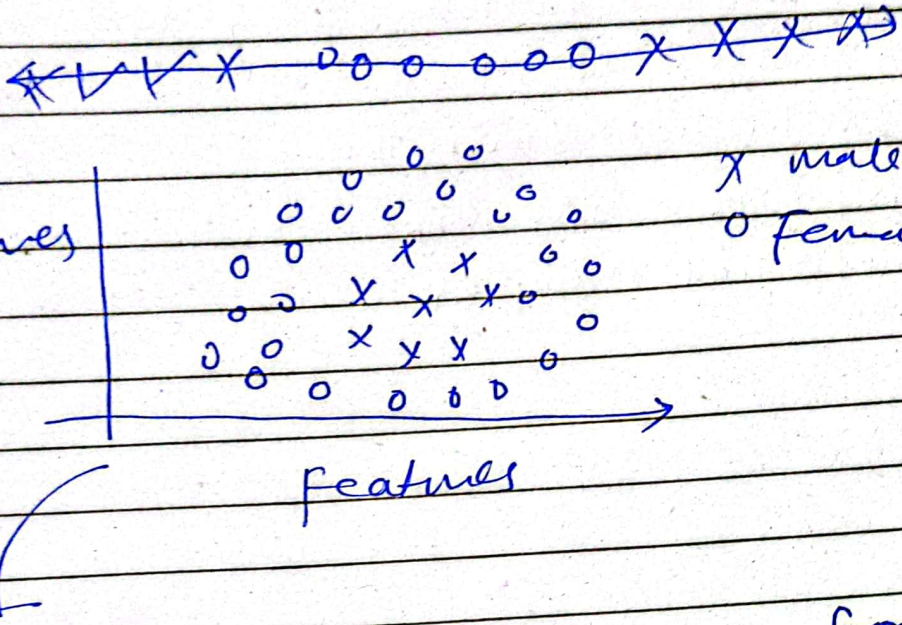
We can cross validation to determine the optimal size of the margins.



- We have only visualized cases where the classes are easily separated by the hyperplane in the original feature spaces.
- Allowing for some misclassifications still resulted in a reasonable results.

M T W T F S DATE:

- What would happen in case where hyperplane performs poorly, even when allowing for misclassification?



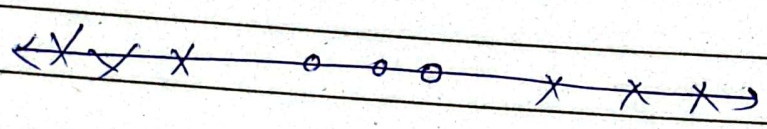
To resolve these cases, we move on from support vector classifiers, to support vector machines

- SVMs use kernels to project the data to a higher dimension, in order to use a hyperplane in this higher dimension to seperate the data.

009 SVM - Kernels

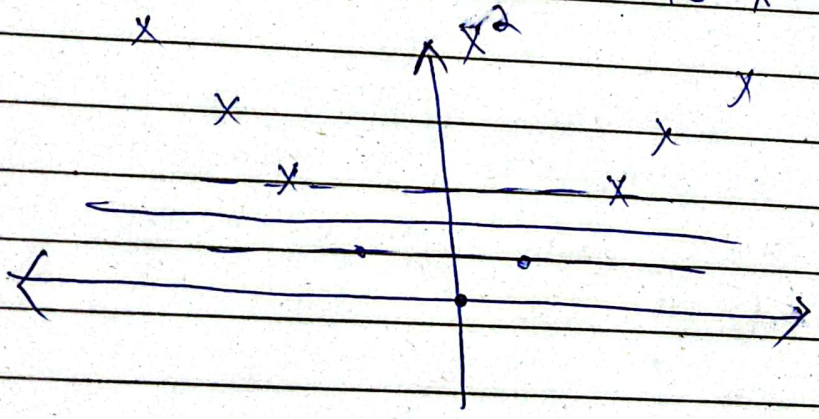
M T W T F S DATE:

- Kernels allows us to move beyond a support vector classifier and use support vector vector Machines.
- There are a variety of kernels we can use to "project" the features to a higher dimension -
- Lets explore how this works through some visual examples

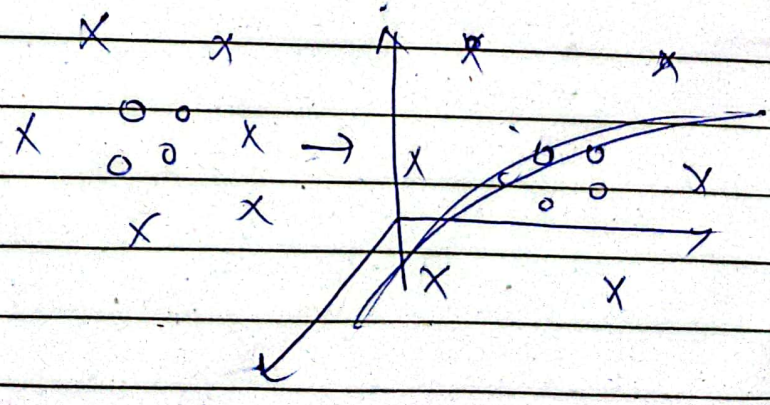


o Default
X No Default

Using a polynomial Kernel will expand onto X^2 dimension



Imagine



NOTES

DATE:

- You may have heard of the use of kernels in SVM as the "kernel trick".
- We previously visualized transforming data points from one dimensional into higher dimension.
- Mathematically, the kernel trick actually avoids recomputing the points in a higher dimensional space!

- How does the kernel trick acheive this?
- It takes advantages of dot products ~~to acheive~~ of the transposition of the data to acheive.
- In the next lecture we will go through the basic Mathematics idea behind the 'kernel trick'!

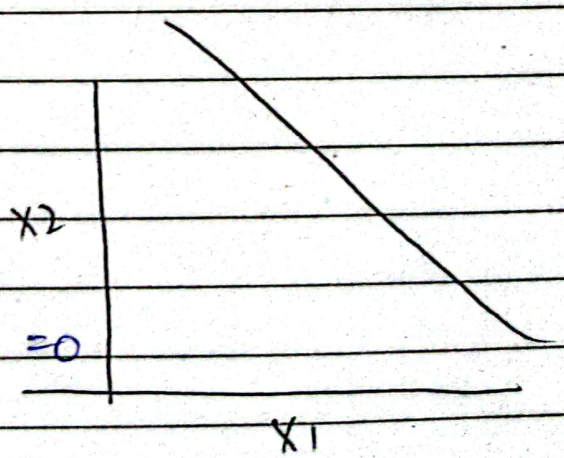
5 SVM - Kernel Trick and Mathematics

Hyper defined

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$

$\Downarrow$

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$

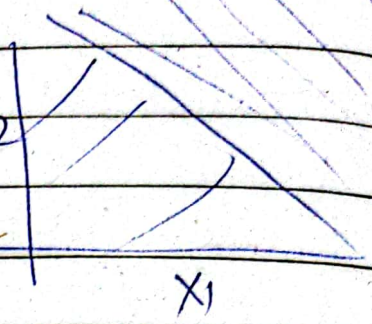


M T W T F S DATE:

- Seperating hyperplane

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0$

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0$



- Data points

$$X_1 = \begin{bmatrix} X_{11} \\ \vdots \\ X_{1p} \end{bmatrix}, \cdots X_n = \begin{bmatrix} X_{n1} \\ \vdots \\ X_{np} \end{bmatrix}$$

- Max Margin Classifier

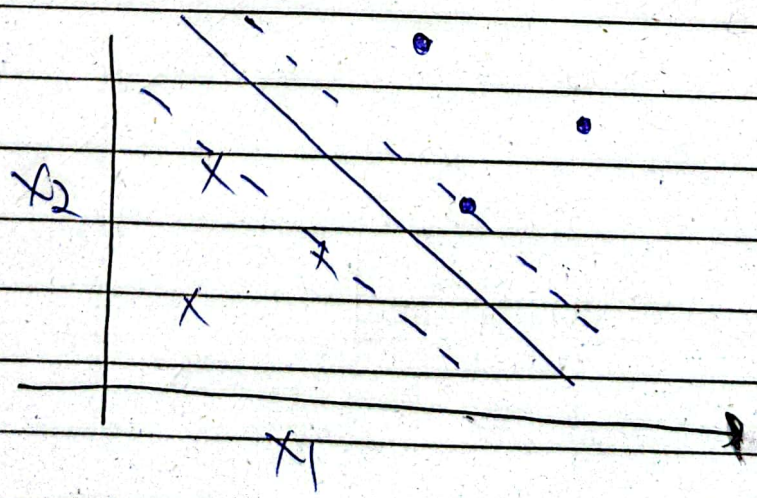
$$X_1 = \begin{bmatrix} X_{11} \\ \vdots \\ X_{1p} \end{bmatrix} \cdots, X_n = \begin{bmatrix} X_{n1} \\ \vdots \\ X_{np} \end{bmatrix}$$

maximize M

$\beta_0, \beta_1, \beta_p \cdots M$

subject to $\sum_{j=1}^{p} \beta_j^2 = 1$

$y_i(\beta_0 + \beta_1 x_{i1} + \beta x_{i2} + \cdots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \cdots n.$



M I D T E R M S DATE:

Support Vector Classifier

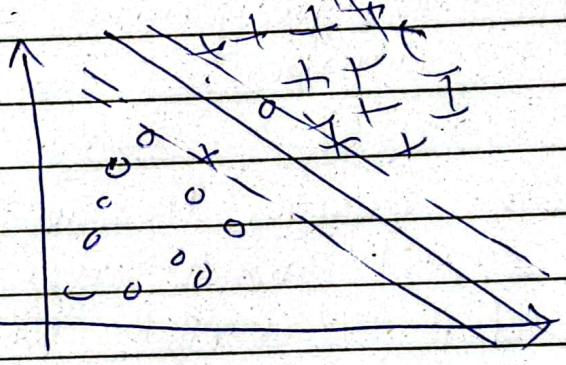
maximize M

$\beta_0, \beta_1, \beta \ldots, \beta_p, \epsilon_1, \ldots \epsilon_n, M$

subject to $\sum_{j=1}^{p} \beta_j^2 = 1$

$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C$

$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$



- Note in Scikit-learn SVC!

C : float default = 1.0

Regularization parameter. The strength of the regularization is inversly proportional to C. Must be strictly positive. The penalty is squared l2 penalty.

Support Vector Machine

$X_1, X_2, X_3 \ldots X_p,$

$\Downarrow$

$X_1, X_1^2, X_2, X_2^2, \ldots X_p, X_p^2$

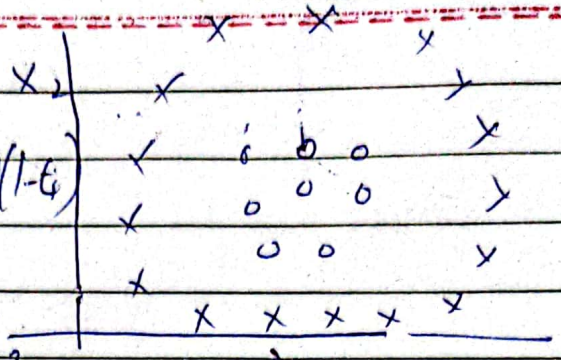
maximize M

$\beta_0, \beta_{11}, \beta_{12} \ldots \beta_{p1}, \beta_{p2}, \epsilon_1, \ldots \epsilon_n, M$

Subject to y_i

$$\left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2\right) \geq M(1-\epsilon_i)$$

$$\sum_{i=1}^{N} \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1.$$



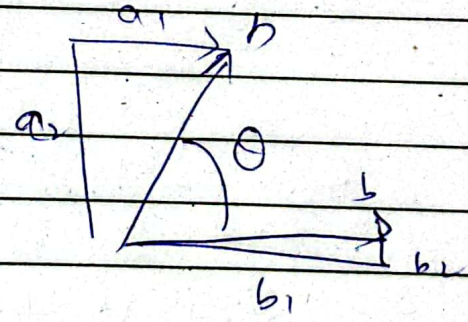
- How to deal with very large features space?
- As a polynomial order grows larger, the number of computations necessary to solve for margin also grows!
- The answer lies in the kernel trick which make use of the inner product of vectors, also known as the dot product.

- Dot product

$$\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$$

$$a \cdot b = a_1 b_1 + a_2 b_2$$

$$a \cdot b = |a||b| \cos(\theta)$$



Notice that dot product can be thought of as a similarity between the vectors.

$$a \cdot b = |a||b| \cos(\theta)$$

$\cos(0) = 1$
$\cos(90) = 0$
$\cos(180) = -1$

MINDFUL DATE:

* Linear Support Vector Classifier rewritten:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

- Only non zero for support vectors
- Calculate the inner product of all pairs of training observations

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$$

S collection of indices of these support points

* Kernel function :-

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

A kernel is a function that quantifies the similarity of two observations.

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i)$$

$$\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$$

M T W T F S DATE:

- polynomial kernel

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d \qquad f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$$

$$\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$$

Radial basis kernel

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right) \quad //$$

//

- The use of kernels as a replacement is know as the kernel trick
- kernel allow us to avoids computations in the enlarged features spaces, by only needing to perform computation for each distinct pair of training points.

SVM - Classification with Scikit-learn part-one

M T W T F S DATE:

```
# code
df = df pd.read_csv("mouse_viral_study.csv")
df.head()

sns.scatterplot(x="Med_1_mL", y='Med_2_mL",
                hue="Virus present', data=df)

x = np.linspace(0,10,100)
m = -1
b = 11
y = m*x + b
plt.plot(x,y,'black')

from sklearn.svm import SVC
help(SVC)

y = df['Virus present']
X = df.drop('Virus present',axis=1)

model = SVC(kernel="linear", C=1000)
model.fit(X,y)
from svm_margin_plot import plot_svm_boundary

plot_svm_boundary(model,X,y)
```

SVM - Classification with Scikit Learn part-2

M T W T F S DATE:_____

```
model = SVC(kernel='rbf', C=1)
model.fit(X, y)
plot_svm_boundary(model, X, y)
```

check for
C = 0,

gamma : {'scale', 'auto'} or float
for rbf, poly, sigmoid

if "gamma = 'scale'

1 / n_features * X.var() as the value of gama

if:
auto

$\frac{1}{n\_features}$

or we can give float value.

```
model = SVC(kernel='sigmoid')
model.fit(X, y)
plot_svm_boundary(model, X, y)

model = SVC(kernel='poly', C=1, degree=1)
model.fit(X, y)
plot_svm(model, X, y)
```

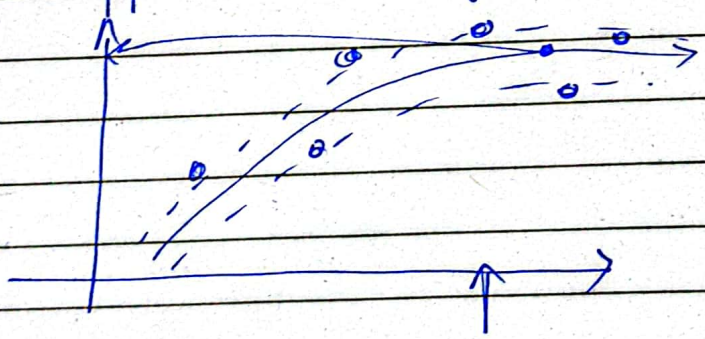
M T W T F S DATE:

```
from sklearn.model_selection import GridSearchCV
svm = SVC()
param_grids = { 'C': [0.01, 0.1, 1, 10],
                'kernel': ['linear', 'rbf'] }
grid = GridSearchCV(svm, paramgrid)
grid.fit(X, y)

grid.best_params
```

8 SVM with scikit-learn Regression Task

- Support Vector Regression

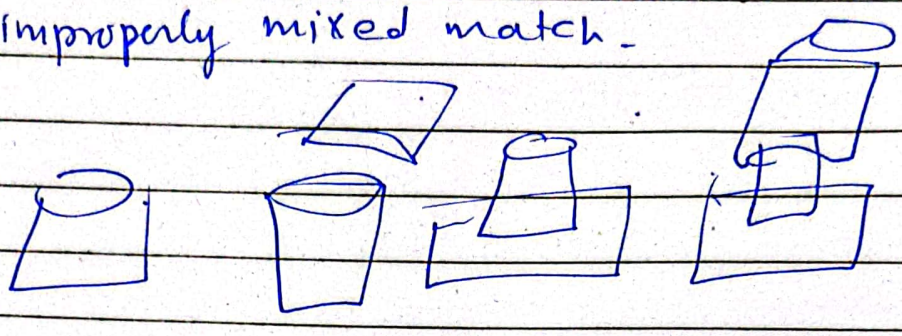


- Concrete slump test :-

It measure the consistency ~~before~~ it ~~actually sets~~ of fresh concrete before it actually sets and its to ~~check~~ perform to check the workability of ~~fresh~~ freshly made concrete and therefore the ease

M T W T F S DATE:_____

with which concrete flows.
also b can be used as an indicater of improperly mixed match.



code:

```
df = pd.read_csv("cement_slump.csv")
df.head()
plt.figsize(figsize=(10,8))
sns.heatmap(df.corr(), annot=True)
df.columns
X = df[" "]
X = df(" ", axis=1)
y = df(" ")

X_train, X_test, y_train, test = train_test_sp(X, y, test=0.3)

from sklearn.preprocessing
scaler = StandardScaler()
scaled_X_train = " "
scaled_X_test = " "

from sklearn.svm import SVR, LinearSVR
```

faster computation but only linear kernel

NOTES

DATE:

if zero it can be considered as overfit

help(SVR)

SVR(*, kernel, epsilon = 0.1
gamma =
C = }

base_model = SVR()
base_model.fit(scaled_X_train, y_train)

based_pred = based_model.predict(scaled_X_test)

mean_absolute_error(y_test, ypreds)
5.23690

mean = np.sqrt(mean_squared_error( ))
6.2542

param_grid = {'C': [0.001, 0.01, 0.1, 0.5, 0.1],
'kernel': ['linear', 'bbf', 'poly'],
'gamma': ['scale', 'auto'],
'degree': [2, 3, 4],
'epsilon': [0, 0.01, 0.5, 1, 2]}

will try to allow error

from sklearn.model_selection import GridSearchCV
svr = SVR()
grid = GridSearchCV(svr, param_grid)
grid.fit(scaled_X_train, y_train)

M T W T F S DATE:_____

grid.best_params=

grid_preds=grid.predic(scaled_X_test)

mean_absolute_error( )

2.5128

np.sqrt( )

3.178