DEFENCE INSTITUTE OF ADVANCED TECHNOLOGY

DEEMED UNIVERSITY

शास्त्रेण शस्त्रं प्रकरोति रक्षाम्

Machine Learning for Cyber Security (CS-602)

L#09

K Nearest Neighbor

By

Dr Sunita Dhavale

Syllabus

- Data Analytics Foundations: R programming, Python Basics -Expressions and Variables, String Operations, Lists and Tuples, Sets, Dictionaries Conditions and Branching, Loops, Functions, Objects and Classes, Reading/Writing files, Hand ling data with Pandas, Scikit Library, Numpy Library, Matplotlib, scikit programming for data analysis, setting up lab environment, study of standard datasets. Introduction to Machine Learning- Applications of Machine Learning, Supervised, unsupervised classification and regression analysis
- Python libraries suitable for Machine Learning Feature Extraction. Data pre-processing, feature analysis etc., Dimensionality Reduction & Feature Selection Methods, Linear Discriminant Analysis and Principal Component Analysis, tackle data class imbalance problem

Syllabus

- Supervised and regression analysis, Regression, Linear Regression, Non-linear Regression, Model evaluation methods, Classification, K-Nearest Neighbor, Naïve Bayes, Decision Trees, Logistic Regression, Support Vector Machines, Artificial Neural Networks, Model Evaluation. Ensemble Learning, Convolutional Neural Networks, Spectral Embedding, Manifold detection and Anomaly Detection
- Unsupervised classification K-Means Clustering, Hierarchical Clustering, Density-Based Clustering, Recommender Systems-Content-based recommender systems, Collaborative Filtering, machine learning techniques for standard dataset, ML applications, Case studies on Cyber Security problems that can be solved using Machine learning like Malware Analysis, Intrusion Detection, Spam detection, Phishing detection, Financial Fraud detection, Denial of Service Detection.

Text/Reference Books

1. Building Machine Learning Systems with Python – Willi Richert, Luis Pedro Coelho
2. Alessandro Parisi, Hands-On Artificial Intelligence for Cybersecurity: Implement smart AI systems for preventing cyber attacks and detecting threats and network anomalies Publication date :Aug 2, 2019, Packt, ISBN-13, 9781789804027
3. Machine Learning: An Algorithmic Perspective – Stephen Marsland
4. Sunita Vikrant Dhavale, "Advanced Image-based Spam Detection and Filtering Techniques", IGI Global, 2017
5. Soma Halder , Sinan Ozdemir, Hands-On Machine Learning for Cybersecurity: Safeguard your system by making your machines intelligent using the Python ecosystem, By Publication date : Dec 31, 2018, Packt, ISBN-13 :9781788992282

1. Stuart Russell, Peter Norvig (2009), "Artificial Intelligence – A Modern Approach", Pearson Elaine Rich & Kevin Knight (1999), "Artificial Intelligence", TMH, 2nd Edition
2. NP Padhy (2010), "Artificial Intelligence & Intelligent System", Oxford
3. ZM Zurada (1992), "Introduction to Artificial Neural Systems", West Publishing Company
4. Research paper for study (if any) – White papers on multimedia from IEEE/ACM/Elsevier/Spinger/ Nvidia sources.

Lab assignments

1	**Python Programming part-1**
2	Python Programming part-2
3	Study and Implement Linear Regression Algorithm for any standard dataset like in cyber security domain
4	Study and Implement KMeans Algorithm for any standard dataset in cyber security domain
5	Study and Implement KNN for any standard dataset in cyber security domain
6	Study and Implement ANN for any standard dataset in cyber security domain
7	Study and Implement PCA for any standard dataset in cyber security domain
8	Case Study: Use of ML along with Fuzzy Logic/GA to solve real world Problem in cyber security domain
9	Mini assignment: Apply ML along with PSO/ACO to solve any real world problem in cyber security domain
10	ML Practice Test – 1 Quiz

Defence Institute of Advanced Technology

School of Computer Engineering & Mathematical Sciences

SEMESTER-I TIME TABLE (AUTUMN 2024)$

PROGRAMMES: (I) CS [M.TECH IN CYBER SECURITY] (II) AI [M.TECH CSE (ARTIFICIAL INTELLIGENCE)] BATCH: 2024-2026

Lecture / Day	L1 0900-1000	L2 1000-1100	L3 1100-1200	L4 1200-1300		L4 1400-1500	L4 1500-1600	L4 1600-1700	L4 1700-1800
Monday	CE-602 (AI) CS-602 (CS)	CE-604 (AI) CS-603 (CS)	CE-601 (AI) CS-604 (CS)	CE-601 (AI) LAB CS-603 (CS)	Lunch Break 1300-1400	LAB CE-601 (AI) LAB CS-602 (CS)		AM607	
Tuesday	CE-603 (AI) LAB CS-603 (CS)	CE-602 (AI) CS-602 (CS)	CE-601 (AI) CS-605 (CS)	CE-604 (AI) CS-604 (CS)		PGC 601		AM607	
Wednesday	CS-605 (CS)	CE-603 (AI) CS-602 (CS)	CE-602 (AI) CS-603 (CS)	CE-604 (AI) CS-604 (CS)		CE-605(AI) LAB CS-605 (CS)	LAB CS-605 (CS)	AM607	
Thursday	LAB CE-604 (AI) CS-603 (CS)	LAB CE-604 (AI) CS-605 (CS)	LAB CE-602 (AI) CS-601 (CS)	CE-603 (AI) CS-601 (CS)		PGC 601		AM607	
Friday	LAB CE-603 (AI) LAB CS-601 (CS)		LAB CE-602 (AI) CS-601 (CS)	LAB CS-604 (CS)		CE-605(AI) LAB CS-604 (CS)	CE-605(AI)	LAB CE-605(AI)	

COURSE CODE & COURSE NAME		FACULTY
Programme: CS [M.Tech in Cyber Security] Classroom: Arjun	Programme: AI [M.Tech CSE (Artificial Intelligence)] Classroom: Kaveri	
CS-601 Data Security & Privacy	CE-601 Responsible Artificial Intelligence;	MJN: Dr. Manisha J. Nene
CS-602 ML for Cyber Security	CE-604 Practical Machine Learning;	SVD: Dr. Sunita V. Dhavale
CS-605 Network and Cloud Security	CE-602 Intelligent Algorithms	CRS: Prof. CRS Kumar
CS-604 Advanced System Security	---------	DVV: Dr. Deepti V. Vidyarthi
CS-603 Applied Cryptography	---------	AM: Dr. Arun Mishra
---------	CE-603 Deep Neural Network;	US: Dr. Upasna Singh
---------	CE-605 Mathematics for ML;	Unit-2: Dr Upasna, Unit 4: Dr Sunita, Unit3:MJN, Unit 1: Faculty To be Nominated
AM-607 Mathematics for Engineers	AM-607 Mathematics for Engineers	OO/DS/DP: Dr Odellu O., Dr Dasari S., Dr. Debasis P.
PGC-601 Research Methodology	PGC-601 Research Methodology	Common Subject for All

$ TENTATIVE T.T. SUBJECT TO CHANGE

Program Coordinator,
M.Tech (CS & AI), Batch 2024-26

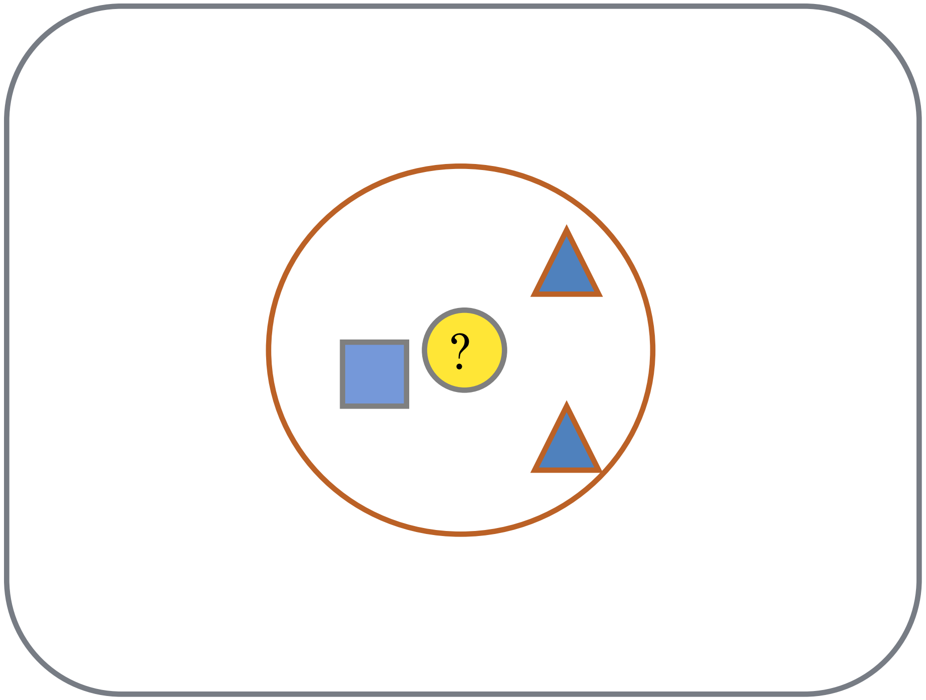
Director, SoCE&MS

K Nearest Neighbor (KNN)

Introduction

- k nearest neighbor (kNN) algorithm is a classification algorithm.
- data points that are "near" each other are classified as belonging to the same class.
- When a new point is introduced, it's added to the class of the majority of its nearest neighbor.
- For example, suppose that k equals 3, and a new data point is introduced.
- Look at the class of its 3 nearest neighbors: let's say they are A, A, and B. Then by majority vote, the new data point is labeled as a data point of class A.
- Measure similarity by creating a vector representation of the items, and then compare the vectors using an appropriate distance metric (such as Euclidean distance).

KNN



- Requires 3 things:
  - Feature Space(Training Data)
  - Distance metric
    - to compute distance between records
  - The value of *k*
    - the number of nearest neighbors to retrieve from which to get majority class
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record

Steps

- Decide k
- Calculate distance of unknown point x from each data point
- Sort distances
- Select k nearest neighbors
- Determine label for x based on majority vote

The simplest version of the k-NN classifier

Suppose we have a mechanism to evaluate the similarly between attribute vectors. Let $\mathbf{x}$ denote the object whose class we want to determine.

1. Among the training examples, identify the k nearest neighbors of $\mathbf{x}$ (examples most similar to $\mathbf{x}$).
2. Let c_i be the class most frequently found among these k nearest neighbors.
3. Label $\mathbf{x}$ with c_i.

Example

- Predict class label for unknown point Dx=(3,4)

Point	x	y	label	D=euclidian(Dx-Di)	Rank
D1	3	3	L1		
D2	-1	4	L2		
D3	2	3	L1		
D4	0	-5	L2		

Common Distance Metrics

Euclidean distance – used mostly

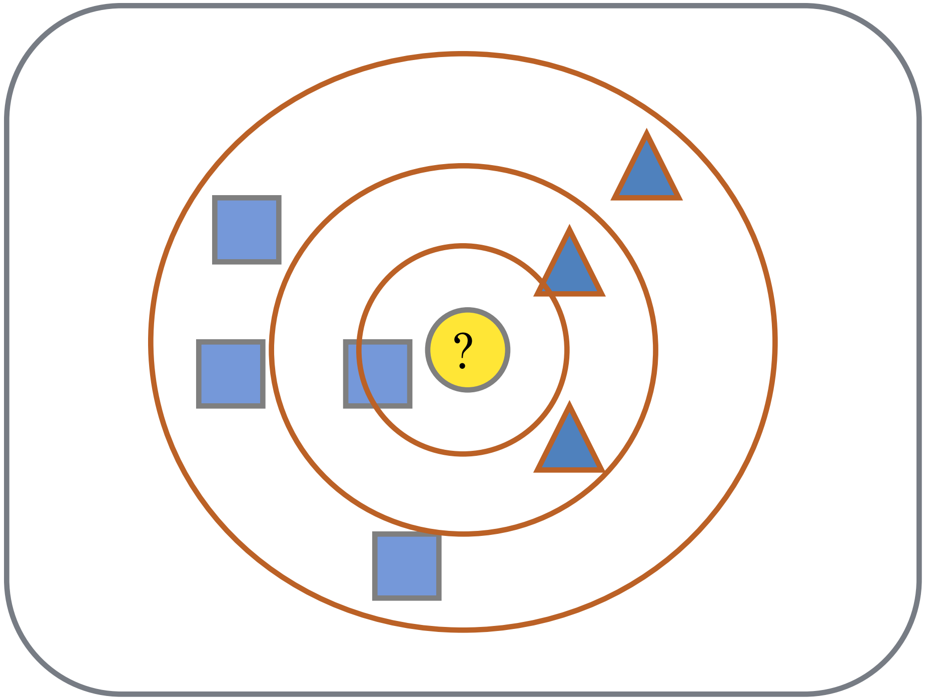
Hamming distance

if $x = y$ then $d(x,y) = 0$. Otherwise, $d(x,y) = 1$

Any distance metric has to satisfy the following requirements:

1. the distance must never be negative;
2. the distance between two identical vectors, $\mathbf{x}$ and $\mathbf{y}$, is zero;
3. the distance from $\mathbf{x}$ to $\mathbf{y}$ is the same as the distance from $\mathbf{y}$ to $\mathbf{x}$;
4. the metric must satisfy the triangular inequality: $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$.

KNN



- $k = 1$:
  - Belongs to square class
- $k = 3$:
  - Belongs to triangle class
- $k = 7$:
  - Belongs to square class

- Choosing the value of *k*:
  - If *k* is too small, sensitive to noise points
  - If *k* is too large, neighborhood may include points from other classes
  - Choose an odd value for *k*, to eliminate ties

KNN

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes.
  - Examples
    - Height of a person may vary from 4' to 6'
    - Weight of a person may vary from 100lbs to 300lbs
    - Income of a person may vary from $10k to $500k
  - **min-max normalization**
  - **z-score standardization**
- z-scores fall in an unbound range of negative and positive numbers. Unlike the normalized values, they have no predefined minimum and maximum.

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

$$X_{new} = \frac{X - \mu}{\sigma} = \frac{X - \mathrm{Mean}(X)}{\mathrm{StdDev}(X)}$$

Nearest Neighbor classifiers are lazy learners. No pre-constructed models for classification.

How to Handle a Tie in kNN

- An odd value for k is less likely to result in a tie vote, but it's not impossible. For example, suppose that k equals 7, and when a new data point is introduced, its 7 nearest neighbors belong to the set {A,B,A,B,A,B,C}. As you can see, there is no majority vote, because there are 3 points in class A, 3 points in class B, and 1 point in class C.
- To handle a tie in kNN:
  - Assign higher weights to closer points
  - Increase the value of k until a winner is determined
  - Decrease the value of k until a winner is determined
  - Randomly select one class

How to determine the good value for k?

- Determined experimentally
- Start with k=1 and use a test set to validate the error rate of the classifier
- Repeat with k=k+2
- Choose the value of k for which the error rate is minimum
- Note: k should be odd number to avoid ties
- The decision of how many neighbors to use for k-NN determines how well the model will generalize to future data.
- The balance between overfitting and underfitting the training data is a problem known as **bias-variance tradeoff**.
- Choosing a large *k* reduces the impact or variance caused by noisy data, but can bias the learner so that it runs the risk of ignoring small, but important patterns.

Curse of Dimensionality

- Imagine instances described by 20 features (attributes) but only 3 are relevant to target function
- Curse of dimensionality: nearest neighbor is easily misled when instance space is high-dimensional
- Dominated by large number of irrelevant features

Possible solutions

- Use cross-validation
- feature subset selection
- PCA

Pros/Cons

- Simple technique that is easily implemented
- Building model is inexpensive
- Extremely flexible classification scheme
  - does not involve preprocessing
- Classifying unknown records are relatively expensive
  - Requires distance computation of k-nearest neighbors
  - Computationally intensive, especially when the size of the training set grows
- Lazy learner; must compute distance over k neighbors
- Accuracy can be severely degraded by the presence of noisy or irrelevant features

Rcode for solved example

- x=c(3,-1,2,0)
- y=c(3,4,3,-5)
- z=c(1,2,1,2)
- t=data.frame(x,y)
- library(class)
- pred <- knn(train = t, test = c(3,4), cl= z, k = 3, prob=TRUE)
- #no prior model is generated here
- pred

RCode

- summary(iris)
- table(iris$Species)
- round(prop.table(table(iris$Species)) * 100,digits = 1)
- ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
- trainData <- iris[ind==1,]
- testData <- iris[ind==2,]
- *#removing factor variable from training and test datasets*
- trainData1 <- trainData[-5]
- testData1 <- testData[-5]

RCode

- *#checking the dimensions of train and test datasets*
- dim(trainData)
- dim(trainData1)
- dim(testData)
- dim(testData1)
- iris_train_labels <- trainData$Species
- dim(iris_train_labels)
- Class(iris_train_labels)
- iris_test_labels <- testData$Species

RCode

- *#install.packages("class")*
- **library**(class)
- iris_test_pred1 <- knn(train = trainData1, test = testData1, cl= iris_train_labels,k = 3,prob=TRUE)
- *#KNN returns the predicted lables for test data set*
- *#install.packages("gmodels")*
- table(iris_test_labels, iris_test_pred1)
- **library**(gmodels)
- CrossTable(x = iris_test_labels, y = iris_test_pred1,prop.chisq=FALSE)

R Code

- str(iris)
- head(iris)
- plot(iris$Sepal.Length, iris$Sepal.Width)
- #install.packages('ggvis')
- library(ggvis)
- iris %>% ggvis(~Sepal.Length, ~Sepal.Width, fill = ~Species) %>% layer_points()
- iris %>% ggvis(~Petal.Length, ~Petal.Width, fill = ~Species) %>% layer_points()
- # Overall correlation `Petal.Length` and `Petal.Width`
- cor(iris$Petal.Length, iris$Petal.Width)
- # Return values of `iris` levels
- x=levels(iris$Species)

R code

- # Print Setosa correlation matrix
- print(x[1])
- cor(iris[iris$Species==x[1],1:4])
- # Print Versicolor correlation matrix
- print(x[2])
- cor(iris[iris$Species==x[2],1:4])
- # Print Virginica correlation matrix
- print(x[3])
- cor(iris[iris$Species==x[3],1:4])
- #the overall correlation is 0.96, while for Versicolor this is 0.79. Setosa and Virginica, on the other hand, have correlations of petal length and width at 0.31 and 0.32 when you round up the numbers.

R Code

- data(iris)
- table(iris$Species)
- round(prop.table(table(iris$Species)) * 100, digits = 1)
- summary(iris)
- normalize <- function(x) {
-   num <- x - min(x)
-   denom <- max(x) - min(x)
-   return (num/denom)
- }
- # Normalize the `iris` data
- iris_norm <- as.data.frame(lapply(iris[1:4], normalize))
- # Summarize `iris_norm`
- summary(iris_norm)
- set.seed(1234)
- #setting a seed is that you can get the same sequence of random numbers whenever you supply the same seed in the random number generator
- ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.67, 0.33))

R Code

- # Compose training set
- iris.training <- iris[ind==1, 1:4]
- head(iris.training)
- # Compose test set
- iris.test <- iris[ind==2, 1:4]
- head(iris.test)
- # Compose `iris` training labels
- iris.trainLabels <- iris[ind==1,5]
- print(iris.trainLabels)
- # Compose `iris` test labels
- iris.testLabels <- iris[ind==2, 5]
- print(iris.testLabels)
- library(class)
- # Build the model
- iris_pred <- kNN(train = iris.training, test = iris.test, cl = iris.trainLabels, k=3)
- # Inspect `iris_pred`
- print(iris_pred)
- table(iris.testLabels,iris_pred)

R code

- # Put `iris.testLabels` in a data frame
- irisTestLabels <- data.frame(iris.testLabels)
- # Merge `iris_pred` and `iris.testLabels`
- merge <- data.frame(iris_pred, irisTestLabels)
- # Specify column names for `merge`
- names(merge) <- c("Predicted Species", "Observed Species")
- # Inspect `merge`
- merge
- #install.packages("gmodels")
- library(gmodels)
- CrossTable(x = iris.testLabels, y = iris_pred, prop.chisq=FALSE)
- #the last argument prop.chisq indicates whether or not the chi-square contribution of each cell is included. The chi-square statistic is the sum of the contributions from each of the individual cells and is used to decide whether the difference between the observed and the expected values is significant

Python code

- # Import necessary modules
- from sklearn.neighbors import KNeighborsClassifier
- from sklearn.model_selection import train_test_split
- from sklearn.datasets import load_iris
- # Loading data
- irisData = load_iris()
- # Create feature and target arrays
- X = irisData.data
- y = irisData.target
- # Split into training and test set
- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)
- knn = KNeighborsClassifier(n_neighbors=7)
- knn.fit(X_train, y_train)
- # Predict on dataset which model has not seen before
- print(knn.predict(X_test))
- # Calculate the accuracy of the model
- print(knn.score(X_test, y_test))

Thank You

- **Any Questions???**

References

- https://courses.washington.edu/css490/2012.Winter/lecture_slides/02_math_essentials.pdf
- Christopher Bishop: ”Pattern Recognition and Machine Learning” , 2006
- Kevin Murphy: ”Machine Learning: a Probabilistic Perspective”
- David Mackay: ”Information Theory, Inference, and Learning Algorithms”
- Ethem Alpaydin: ”Introduction to Machine Learning” , 2nd edition, 2010.
- R. Duda, P. Hart & D. Stork, ***Pattern Classification*** (2nd ed.), Wiley T. Mitchell, ***Machine Learning,*** McGraw-Hill