Final Exam

CIS 635 Final Exam Study Guide

1. data
   a) type(nominal, ordinal, numeric)

   ordinal same as nominal except for some order
   b) selecting attributes(feature reduction)

   This is done when too many attributes

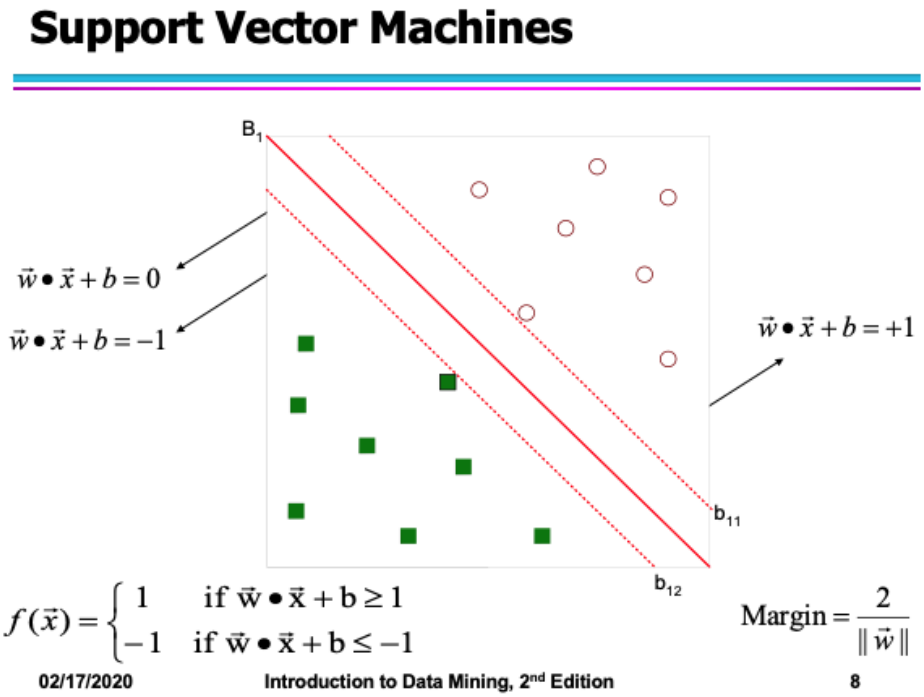
   Methods: **brute force**(one attribute at time, and do classification for each attribute and choose by accuracy. then add another attribute and so on), **PCA: Pricinple Component Analysis** (mathematically create a smaller set of smaller attributes. But only mathematical composition, may not be meaning but faster)
   c) R (instance selection, summary stats, sub-matrices, plots)

   Not expecting of detail knowledge R, select row, column, plots, summary, etc
   d) missing values

   In some instance, remove entire data like if bad data. If only one piece, estimate it like taking mean.
   e) discretization

   turn numeric attribute into bins(ordinal, nominal)

2. classification
   a) calculate models for
   • **decision tree(gini(nominal), y or n from tree)**
   • **naive Bayes (use calculation for posterior) prior, likelihoods**
   b) classification
   • decision tree
   • naive Bayes
   • KNN(k =1, distance already, yor n)
   • SVM (Visual representations of data, no calculation)



• ANN(no question)

Kmeans:

c) confusion matrix

• calculating the matrix from results

• calculating precision, recall and accuracy from confusion matrix

d) ability to describe the concepts of overfitting and curse of dimensionality

e) match data sets to models

1. describe overfitting,: naiveBayes cannot do overfitting, KNN no model no need. SVM: **has to do with kerne**l. **Polynomical or X is more complex than linear.**
    a. ANN: **more hidden nodes or layers**

- **overfitting**
  - how does each model overfit
  - decision trees – tree grows too large
  - svm – kernel is too complex (polynomial vs linear)
  - ann – too many hidden layers or nodes
- **curse of dimensionality**
  - as the number of attributes grows larger at some point the accuracy of the test data goes down

handover: compAlgs go through it, what model fit the data. (One question)

look at the shape of data and what model.

3. clustering

- **what are the general characteristics**
  - kmeans
    - fast, not deterministic
    - finds globular clusters
  - agglomerative
    - finds a hierarchy of clusters from 2 to n
    - single-link merges based on closest instances
    - complete-link merges based on farthest instances

- **SSE (sum squared error)**
  - for each cluster j
    - for each instance i in cluster j
      - sum = sum + $d(x_i, c_j)^2$
- **sse can be calculated for entire clustering or just one cluster**

a) general understanding of kmeans, agglomerative and DBSCAN
b) proximity measures (euclidean → **numeric**, cosine → **factors, nominal**) **for clustering do not mix nominal and numeric**. **either use all numeric or nominal**. → change the data.
c) characteristics of the different algorithms

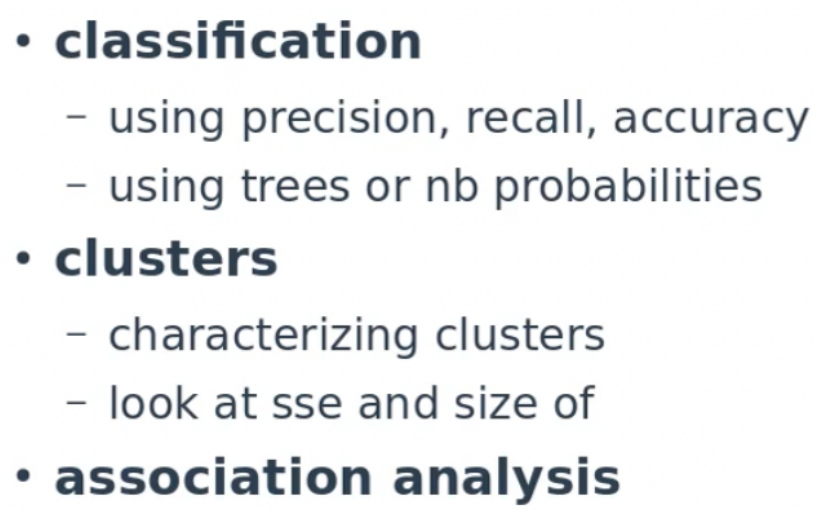
for kmeans: find equal size cluser,
d) kmeans
• calculate centroids
• assign instances to clusters
e) agglomerative
• understand the general process
• know the difference between single(**some cluster big and small**) and complete link(**same size**)
f) describe cluster characteristics(**go straight what makes it unique**)

eucilidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

4. association analysis
   a) calculate support for itemsets and rules

b) calculate confidence for rules
c) interpret results



1. miscellany
   a) anomaly detection(**general understanding**): **statistical approac: means, sd**
   b) ensemble methods(**general idea what they do**): **take several model, run data, different model for different part of data, → increase chance of precision like random forest**
   c) network mining
   • models (random, small world, scale free)
   • metrics (centrality, diameter...)
   • techniques:

   - link-based classification
   - link prediction → **predict if link exist or not**
   - ranking → **used by google, rank node based on authority**
   - influence matrix → **find influencer**
   - communities finding → **same as clustering just use the graph, put very few links between communities and lot of links in the community**

   d) text mining
   • preprocessing (stemming[**trying to find words with same root(node) and replace it. like run running ran replace with run**], pos tagging(**determine which word is noun, verb etc in the speech., how often people use those**

thing), tfidf(term freq. inverse doc freq: **how often ⇒ tf/df**))

• building blocks (document similarity, sentiment analysis(**is it +ve word or -ve word, take word and turn into vector.**), word2vec)

• applications (text categorization, document clustering, ascribing authorship)

Review for final

Data classification

sales orders type churn predic

56.07 24 p n n

sales	orders	type	churn	predict
56.08	24	p	n	n
34.08	29	d	y	n

type → nominal

1. College applicant data (data from the applications of high school students applying to college)

   a) data matrix along with predictions. Data matrix has 9 columns including the class "accept". Note that the column "pred" does not belong to the data matrix – it is beside the data matrix for your convenience.

id	age	act	sat	gpa	IB	sports	music	accept		pred
101	24	28	1260	3.1	y	n	y	0		0
102	25	28	1260	3.22	n	y	y	0		0
103	18	28	1260	3.29	y	y	y	0		1
104	23	27	1170	2.9	n	n	n	0		0
105	20	22	990	NA	y	n	n	1		1
106	24	24	1080	3.53	n	n	n	1		1
107	25	34	1530	2.79	y	y	n	0		1
108	17	20	900	3.21	y	y	y	1		0
109	18	26	1170	3.09	y	n	y	0		0
110	22	26	1170	3.28	n	n	y	1		1
111	19	23	1035	3.64	n	n	n	1		1
112	18	28	1260	3.18	n	y	y	0		0
113	17	26	1170	3.46	n	y	n	1		1
114	24	30	1350	3.28	y	n	y	0		0
115	20	27	1215	3.35	y	y	y	1		1
116	22	28	1260	3.77	n	y	n	1		1

Select

x[1:10, 2:3]

x[1:10, c(2,3,4)]

ind = x[,8]== “n”

mean(x[ind, 4])

mean(na.omit(x[ind, 4]))

x$gpa[is.na(x$gpa)] = mean(x$gpa, na.rm=TRUE)

x$gpa[is.na(x$gpa)] <- mean(x[x$gpa == “class”,], na.rm=TRUE)

plot: scatter, bar, boxplot

first row: x[1,] or head(x, 1)

column for gpa x[, “gpa”] or x[, c(”gpa”)]

all rows where accept is 0: x[x$accept == 0,]

column headings: colnames(x)

mean of sat = mean(x$sat) or mean(x[, “sat”])

min of age: min(x$age)

count of n in IB = nrow(x[x$IB == “n”,])

replace NA iwith global gpa average = x$gpa[is.na(x$gpa)] = mean(x$gpa, na.rm=TRUE)

replace NA wigth class gpa average = x$gpa[is.na(x$gpa)] = mean(x[x$gpa == “class”, “gpa”])

ind of logical: ind <- data$column == “accepted”

submatrics: x[x$accept == 1, c(”age”, “act”, “sat”)]

scatterplot: ggplot(data=x, mapping = aes(x=act, y=gpa, color=accept) + geom_point()

barchart: ggplot(data=x, aes(x=act, y=gpa, color=accept) + geom_bar(stat=”identity”)

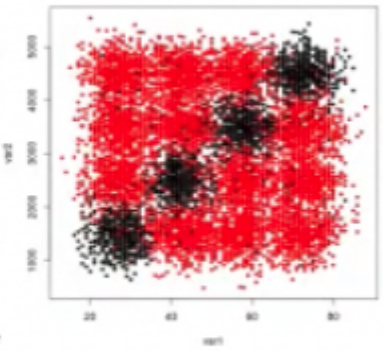
boxplot: ggplot(data=x, aes(x=act, y=gpa, color=accept) + geom_boxplot()

Model comparisons:

Decision Tree (Rectangular: slice and dice data horizontal and vertical)

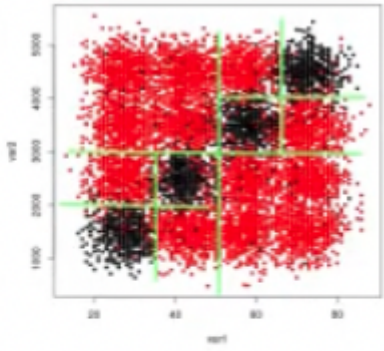
1 Rectangular

In data set 1, the two classes are to be separated in a way that favors decision trees. Thus, instances can be separated by slicing the data horizontally and vertically using var1 and var2, forming rectangles. The data is plotted to the right with black dots for the healthy people and red dots for those who are not healthy.



Story: Imagine 4 groups of healthy people whose fitness times are proportional to the calories they consume. Those are the four black groups in the main diagonal. So in the lower left there is a group that works out for an average of about 25 minutes and consumes an average of about 400 calories. The other red groups represent unhealthy people who either eat too little for their workout sets or workout too little for their calorie consumption.

Results: You can see from the table below that the tree classifier has the highest accuracy which is due to the rectangular nature of var1 and var2. Below the plot of the data is the same plot with lines drawn where a hypothetical tree model would place its branches. One can imagine the first branch at var2>3000. The actual tree model that is found by *cart* is different but the difference is a result of the greedy nature of cart. It does not impede our understanding of the algorithm to use this simple example model.



		time (seconds)	
	accuracy	learn	predict
tree	**0.882**	1.432	0.027
nb	0.745	0.104	1.138
knn	0.887	0.000	1.359
svm	0.745	22.022	0.633
ann	0.807	8.424	0.038

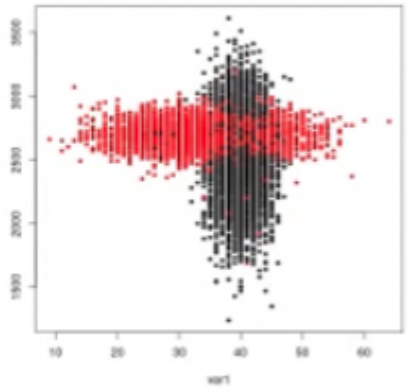
The other methods have difficulty with this data for different reasons. For nb the model consists of priors and likelihoods. Since we have the same number of instances of both classes the priors will be identical. The likelihoods also will be identical – means of 50 for var1 and 3000 for var2. An effective nb model should have some different likelihoods for different class values. svm performs poorly because the healthy people are in the middle of the unhealthy ones so it is difficult to find a boundary that cleanly separates them. ann (with one hidden layer) is essentially a linear method with multiple lines. This data set is too complex for this simple case of ann.

Naive Bayes (Looking for normal or gaussian distribution- > good for these data)

helps which attribute are important in makding decison

2 Gaussian

The Gaussian set has two groups of people that are distributed normally but in different ways. The healthy group (black dots) is spread in a narrow band horizontally but spread wide vertically. The unhealthy group is spread narrowly vertically but widely horizontally (see the plot to the right).



Story: This story is more implausible than the previous but here goes. The healthy people in this population have workouts right around 40 minutes. That length of time for these people seems to be just the right amount. It doesn't seem to matter how much they eat – 1500 to 3500 calories, they still remain healthy. On the other hand, the unhealthy people eat a narrow diet of around 2700 calories but their workouts can be 10 to 60 minutes per day.

Results: nb has the best accuracy for this data, barely doing better than decision tree. The nb model finds slightly different means for the two classes. For var1, healthy is 40 and unhealthy is 35. For var2, healthy is about 2500 and unhealthy is about 2700. The std. dev values are what is helpful. The difficulty area to predict is the intersection of the two groups.

		time (seconds)	
	accuracy	learn	predict
tree	0.827	1.081	0.029
nb	**0.837**	0.107	1.135
knn	0.800	0.000	1.159
svm	0.728	34.905	0.782
ann	0.723	5.140	0.034

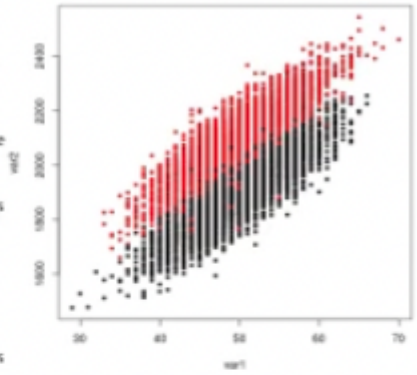
The tree algorithm has difficulty with that same overlap area which is easy to see if you imagine the horizontal and vertical boundaries. This data again presents problems for svm because it is not possible to divide the two classes with a single line. When ann is tried, it appears to find boundary lines that are not close to the data (so that all of the instances are on only one side of the boundary. I'm not sure why this is, maybe because the large area of overlap confounds the classifier.

For the nb classifier to work well, it is not necessary for the data to be distributed normally but it is necessary for the two classes to have data that is centered apart from each other and spread in such a way to make it possible to separate instances of different classes.

SVM (Always binary, linear, curve or circle → two difference classes)

3 Linear

In this set, the instances for the unhealthy people are distributed normally such that the mean for var1 is about 50 and for var2 it is about 2200. There is a positive correlation between the two variables though. The distribution for the healthy people is very similar to the distribution for the unhealthy people except that in general, healthy people eat fewer calories. As you can tell from the plot to the left, one can easily draw a boundary line between the two groups.



Story: So the more you work out, the more you are likely to eat. This is true for both groups. However, since the healthy people eat fewer calories than the unhealthy ones, they are leaner and fitter.

Results: The results show that svm does better than all of the other algorithms but it is only marginally better than ann. This is because both svm and ann, as configured are finding linear boundaries. The tree algorithm had a fairly high level of accuracy. It accomplishes this by making many branches of small increments to essentially draw a jagged boundary between the two classes.

		time (seconds)	
	accuracy	learn	predict
tree	0.904	0.893	0.111
nb	0.796	0.097	1.094
knn	0.921	0.000	1.102
svm	**0.942**	9.135	0.217
ann	0.934	3.284	0.035

For svm with a linear kernel to perform well, the data should have a linear separation between the two classes. For more than 2 variables, the boundary will be a hyperplane but still linear. If the groups are separable by a curved hyperplane then the algorithm will still work fine but a more complex kernel (polynomial or RBF) would need to be used.

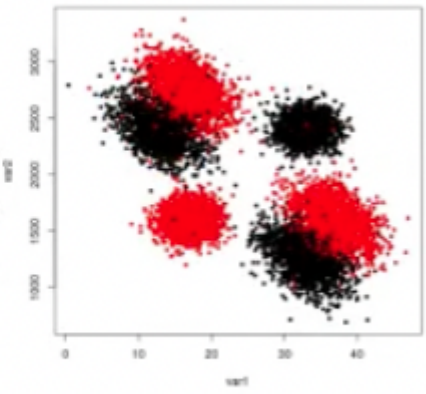
ANN(Number of different linear seperators to break down data into areas of mostly pure data from just one class)

4 Clumps

This data set is simply a set of 6 groups. The variables for the instances in each group are distributed normally but the groups form clumps that are easily separated by visual inspection. Each group is also mostly composed on instances from one or the other class.

Story: Cheez, I don't know. I was really trying to find a set where ann (simple 1 hidden layer) would out-perform the others but it is not as easy as one would think. Lets just say that this represents six groups of people who know each other and have similar habits.

Results: As the table shows, ann has the best accuracy. The plot below, right shows the same data only with the boundary lines that separate the classes. svm cannot do as well because it is using only a single linear separator. nb has poor accuracy because the mean and std. dev. values for the two classes are very similar. Only the tree classifier got close to the same accuracy as ann because it again, created a complex tree with many branches of small differences.



		time (seconds)	
	accuracy	learn	predict
tree	0.938	1.022	0.029
nb	0.652	0.106	1.158
knn	0.945	0.000	1.135
svm	0.694	45.481	1.067
ann	**0.953**	9.582	0.033

For ann to perform well, the data should be clumped some way so that a relatively small number of linear boundaries can separate the classes.