Knowledge Discovery and Data Mining

Unit # 9

Eager Learners

- The classification methods discussed so far in this course—decision tree induction, Bayesian classification and classification by back-propagation—are all examples of *eager learners.*
- *Eager learners,* when given a set of training tuples, will construct a generalization (i.e., classification) model before receiving new (e.g., test) tuples to classify.
- We can think of the learned model as being ready and eager to classify previously unseen tuples.

Lazy Learners

- Imagine a contrasting lazy approach, in which the learner instead waits until the last minute before doing any model construction in order to classify a given test tuple.
- That is, when given a training tuple, a lazy learner simply stores it (or does only a little minor processing) and waits until it is given a test tuple.
- Only when it sees the test tuple does it perform generalization in order to classify the tuple based on its similarity to the stored training tuples.

Instance-based Learning

- In instance-based learning the training examples are stored verbatim, and a distance function is used to determine which member of the training set is closest to an unknown test instance.
- Once the nearest training instance has been located, its class is predicted for the test instance.
- The only remaining problem is defining the distance function, and that is not very difficult to do, particularly if the attributes are numeric.

Distance Function

- Although there are other possible choices, most instance-based learners use Euclidean distance.
- When comparing distances it is not necessary to perform the square root operation; the sums of squares can be compared directly.
- One alternative to the Euclidean distance is the Manhattan or city-block metric, where the difference between attribute values is not squared but just added up (after taking the absolute value).

Normalization

- Different attributes are measured on different scales, so if the Euclidean distance formula were used directly, the effects of some attributes might be completely dwarfed by others that had larger scales of measurement.
- Consequently, it is usual to normalize all attribute values to lie between 0 and 1,

Distance for Categorical Attributes

- For categorical attributes, a simple method is to compare the corresponding value of the attribute in tuple *X1 with that in tuple X2.*
- *If the two are identical (e.g., tuples X1 and X2 both have the color blue), then the difference between the two is taken as 0.*
- If the two are different (e.g., tuple *X1 is blue but tuple X2 is red), then the difference* is considered to be 1.

Right Value for K?

- A good value for K can be determined experimentally.
- Starting with k = 1, we use a test set to estimate the error rate of the classifier.
- This process can be repeated each time by incrementing k to allow for one more neighbor.
- The k value that gives the minimum error rate may be selected.
- In general, the larger the number of training tuples is, the larger the value of k will be (so that classification and prediction decisions can be based on a larger portion of the stored tuples)

Low Training Time

- Unlike eager learning methods, lazy learners do less work when a training tuple is presented and more work when making a classification or prediction.
- Because lazy learners store the training tuples or "instances," they are also referred to as instance based learners, even though all learning is essentially based on instances.

High Prediction Time

- Although instance-based learning is simple and effective, it is often slow.
- The obvious way to find which member of the training set is closest to an unknown test instance is to calculate the distance from every member of the training set and select the smallest.
- This procedure is linear in the number of training instances: in other words, the time it takes to make a single prediction is proportional to the number of training instances.

High Prediction Time (Cont'd)

- When making a classification or prediction, lazy learners can be computationally expensive.
- They require efficient storage techniques and are well-suited to implementation on parallel hardware.
- They offer little explanation or insight into the structure of the data.
- Lazy learners, however, naturally support incremental learning.
- They are able to model complex decision spaces having hyper polygonal shapes that may not be as easily describable by other learning algorithms (such as hyper-rectangular shapes modeled by decision trees).

K-Nearest Neighbor

- The *k-nearest-neighbor* method was first described in the early 1950s.
- The method is labor intensive when given large training sets, and did not gain popularity until the 1960s when increased computing power became available.
- It has since been widely used in the area of pattern recognition.

K-Nearest Neighbor (Cont'd)

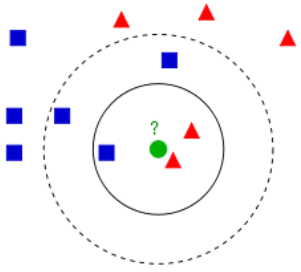
- Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuple with training tuples that are similar to it.
- The training tuples are described by *n attributes.*
- *Each tuple represents a point in an n-dimensional space. In this way,* all of the training tuples are stored in an *n-dimensional pattern space.*
- *When given an* unknown tuple, a ***k-nearest-neighbor classifier searches the pattern space for the k training*** tuples that are closest to the unknown tuple.
- These *k training tuples are the k "nearest* neighbors" of the unknown tuple.
- "Closeness" is defined in terms of a distance metric, such as Euclidean distance.

Prediction using Nearest Neighbor

- Nearest neighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown tuple.
- In this case, the classifier returns the average value of the real-valued labels associated with the *k nearest neighbors of the unknown* tuple.

Example

- The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles.
- If *k = 3* it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle.
- If *k = 5* it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).



Example: Continuous Attributes

X	Y	C
0.3	0.7	A
0.2	0.9	B
0.6	0.6	A
0.5	0.5	A
0.7	0.7	B
0.4	0.9	B

0.2	0.6	?

Example: Mammals vs. Non-mammals
K = 3, K = 5

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

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

Example: Play Tennis
K = 3, K = 5

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Outlook	Temperature	Humidity	Windy	Class
rain	hot	high	false	?

Limiations

- It tends to be slow for large training sets, because the entire set must be searched for each test instance
- It performs badly with noisy data, because the class of a test instance is determined by its single nearest neighbor without any "averaging" to help to eliminate noise.
- It performs badly when different attributes affect the outcome to different extents—in the extreme case, when some attributes are completely irrelevant—because all attributes contribute equally to the distance formula.
- It does not perform explicit generalization

Reducing the Number of Exemplars

- The plain nearest-neighbor rule stores a lot of redundant exemplars: it is almost always completely unnecessary to save all the examples seen so far.
- A simple variant is to classify each example with respect to the examples already seen and to save only ones that are misclassified.
- Discarding correctly classified instances reduces the number of exemplars and proves to be an effective way to prune the exemplar database.

Reducing the Number of Exemplars (Cont'd)

- Unfortunately, the strategy of only storing misclassified instances does not work well in the face of noise.
- Noisy examples are very likely to be misclassified, and so the set of stored exemplars tends to accumulate those that are least useful.
- This effect is easily observed experimentally. Thus this strategy is only a stepping-stone on the way toward more effective instance-based learners.

Pruning Noisy Exemplars

- Noisy exemplars inevitably lower the performance of any nearest-neighbor scheme that does not suppress them because they have the effect of repeatedly misclassifying new instances.
- A solution is to monitor the performance of each exemplar that is stored and discard ones that do not perform well.
- This can be done by keeping a record of the number of correct and incorrect classification decisions that each exemplar makes.

Weighting Attributes

- The Euclidean distance function, modified to scale all attribute values to between 0 and 1, works well in domains in which the attributes are equally relevant to the outcome.
- In most domains, however, some attributes are irrelevant, and some relevant ones are less important than others.
- The next improvement in instance-based learning is to learn the relevance of each attribute incrementally by dynamically updating feature weights.

Weighting Attributes (Cont'd)

- Call the training instance x and the most similar exemplar y.
- For each attribute i, the difference |xi - yi| is a measure of the contribution of that attribute to the decision.
- If this difference is small then the attribute contributes positively, whereas if it is large it may contribute negatively.
- The basic idea is to update the ith weight on the basis of the size of this difference and whether the classification was indeed correct.
- If the classification is correct the associated weight is increased and if it is incorrect it is decreased.