Artificial Intelligence and Machine Learning

Similarity Based Learning

Big Idea

- If it looks like a duck, sounds like a duck, walks like a duck, then it IS a duck.

Applications

- Document classification
- Book/Movie recommendation
- Customer propensity prediction

Fundamentals

- The fundamentals of similarity-based learning are:
  - Feature space
    - An abstract n-dimensional space that is created by taking each of the descriptive features in an ABT to be the axes of a reference space and each instance in the dataset is mapped to a point in the feature space based on the values of its descriptive features.
  - Similarity metrics
    - Measures the similarity between two instances according to a feature space.

Metric

- Mathematically, a metric must conform to the following four criteria:
  - Non-negativity: metric(a, b) >= 0
  - Identity: metric(a, b) = 0 ⇔ a = b
  - Symmetry: metric(a, b) = metric(b, a)
  - Triangular Inequality:
    metric(a, b) <= (metric(a, c) + metric(c, b)

  Where metric(a, b) is a function that returns the distance (or dissimilarity) between two instances a and b.

metric for Simple Attributes

p and *q* are the attribute values for two data objects.

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = -d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

Table 5.1. Similarity and dissimilarity for simple attributes

Common Metric

- Hamming (Manhattan) distance (p = 1)
- Euclidean distance (p = 2)
- Minkowski distance in a feature space with m descriptive features:

$$Minkowski(a,b) = \left(\sum_{i=1}^{m} abs(a[i] - b[i])^p\right)^{\frac{1}{p}}$$

- The larger the value of p, the more emphasis is placed on the features with large differences in values because their differences are raised to the power of p.

Similarity Between Binary Vectors

- Common situation is that objects, *p* and *q*, have only binary attributes
- Compute similarities using the following quantities

M_{01} = the number of attributes where p was 0 and q was 1
M_{10} = the number of attributes where p was 1 and q was 0
M_{00} = the number of attributes where p was 0 and q was 0
M_{11} = the number of attributes where p was 1 and q was 1

- Simple Matching and Jaccard Coefficients

SMC = number of matches / number of attributes
= $(M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00})$

JC = number of 11 matches / number of not-both-zero attributes values
= $(M_{11}) / (M_{01} + M_{10} + M_{11})$

Cosine Similarity

- If d_1 and d_2 are two document vectors, then

$$\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \|d_2\| ,$$

where $\bullet$ indicates vector dot product and $\| d \|$ is the length of vector d.

- Example:

d_1 = **3 2 0 5 0 0 0 2 0 0**
d_2 = **1 0 0 0 0 0 0 1 0 2**

$d_1 \bullet d_2$= 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5

$\|d_1\|$ = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)$^{0.5}$ = (42) $^{0.5}$ = 6.481

$\|d_2\|$ = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2) $^{0.5}$ = (6) $^{0.5}$ = 2.245

$$\cos(d_1, d_2) = .3150$$

Extended Jaccard Coefficient

- Variation of Jaccard Coefficient for continuous or count attributes

$$T(p,q) = \frac{p \bullet q}{\|p\|^2 + \|q\|^2 - p \bullet q}$$

Correlation

- Correlation measures the linear relationship between objects
- To compute correlation, we standardize data objects, p and q, and then take their dot product

$$p'_k = (p_k - mean(p)) / std(p)$$

$$q'_k = (q_k - mean(q)) / std(q)$$

$$correlation(p, q) = p' \bullet q'$$

General Approach for Combining Similarities

- Sometimes attributes are of many different types, but an overall similarity is needed.

1. For the k^{th} attribute, compute a similarity, s_k, in the range $[0, 1]$.
2. Define an indicator variable, δ_k, for the k_{th} attribute as follows:

$$\delta_k = \begin{cases} 0 & \text{if the } k^{th} \text{ attribute is a binary asymmetric attribute and both objects have a value of 0, or if one of the objects has a missing values for the } k^{th} \text{ attribute} \\ 1 & \text{otherwise} \end{cases}$$

3. Compute the overall similarity between the two objects using the following formula:

$$similarity(p, q) = \frac{\sum_{k=1}^{n} \delta_k s_k}{\sum_{k=1}^{n} \delta_k}$$

Using Weights to Combine Similarities

- May not want to treat all attributes the same.
  - Use weights w_k which are between 0 and 1 and sum to 1.

$$similarity(p,q) = \frac{\sum_{k=1}^{n} w_k \delta_k s_k}{\sum_{k=1}^{n} \delta_k}$$

$$distance(p,q) = \left(\sum_{k=1}^{n} w_k |p_k - q_k|^r \right)^{1/r}$$

The Nearest Neighbor Algorithm

Require: set of training instances

Require: a query to be classified

Algorithm:

1. Iterate across the instances and find the instance that is shortest distance from the query position in the feature space.
2. Make a prediction for the query equal to the value of the target feature of the nearest neighbor.

Advantage of Nearest Neighbor Algorithm

- It is a instance-based learning algorithm
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified.
- It is easy to add new data items into the training dataset to update the model.

Flaws in supervised learning

- Supervised machine learning is based on the stationarity assumption which states that the data doesn't change – remains stationary – over time.
- In the context of classification, supervised machine learning creates models that distinguish between the classes that are present in the dataset they are induced from.
- So if a classification model is trained to distinguish between lions, frogs and ducks, the model will classify a query as being either a lion, a frog or a duck; even if the query is actually an elephant.

Handling Noisy Data

- The KNN (K nearest neighbors) model predicts the target level with the majority vote from the set of k nearest neighbors to the query q.
- Distance-weighted KNN:
  - Weight the contribution of each of the k neighbors according to their distance to the query point, and give the greater weight to the closer neighbors, e.g. $w = \frac{1}{(d(x_q,x_i)^2}$

Data Normalization

- Data features take different ranges of values.
- This is equivalent to features having different variances.

ID	Salary	Age
1	55370	21
2	55000	55
Diff	370	34

- This can be adjusted using normalization. The equation commonly used for range normalization is:

$$a_i' = \frac{a_i - \min(a)}{\max(a) - \min(a)} \times (high - low) + low$$

Predicting Continuous Targets

- Using KNN algorithm, and return the average value in the neighborhood.
- Using weighted KNN algorithm, the model prediction equation can be changed to:

$$Model(x_q) = \frac{\sum_{i=1}^{k}(\frac{1}{dist(x_q, x_i)^2} \times t_i)}{\sum_{i=1}^{k} \frac{1}{dist(x_q, x_i)^2}}$$

Dimension Reduction

- Curse of Dimensionality: distance between neighbors could be dominated by irrelevant or redundant attributes.
- Solutions:
  - feature selection – a search problem
    - Find an evaluation metric to measure the score of a subset of the features
    - Hypothesis: Good feature subsets contain features highly correlated with the classification, yet uncorrelated to each other.
    - Approaches:
      - Score all combinations of features and find the best one (not feasible)
      - Iteratively select the best feature stepwise (forward)
      - Iteratively eliminate the worst feature stepwise (backward)

Dimension Reduction (II)

- Feature Extraction/Creation
  - It is sometimes possible to create, from the original attributes, a new set of attributes that captures the important information in a data set much more effectively.
  - The number of new attributes can be smaller than the number of original attributes.
  - One of the approaches (for all numerical attributes): Principal Component Analysis
    - Given N data vectors from k-dimensions, find c <= k orthogonal vectors that can be best used to represent data
    - The original data set is reduced to one consisting of N data vectors on c principal components (reduced dimensions)
    - Each data vector is a linear combination of the c principal component vectors

Efficient Memory Search

- Assuming that the training set will remain relatively stable, it is possible to speed up the prediction speed of a nearest neighbor model by investing in some one-off computation to create an index of the instances that enables efficient retrieval of the nearest neighbors.
- The best know indices:
  - k-d tree (short for k-dimensional tree)
  - k-d B-tree
  - R tree

K-D Tree

- A binary search tree
- A space-partitioning data structure for organizing points in a k-dimensional space.
- Each node is associated with a sub space in the k-dimensional hyper space and all the data points reside in this sub space.
- Each internal node is associated with one dimension and generates a splitting hyperplane that divides the space into two parts, associated with its two child nodes.
- Important issue: keep it balanced

K-D Tree Example