SIGNIFICANCE OF SELECTION ALGORITHMS IN DATA PROCESSING

Ma'mura Gayratovna Ergasheva,

Pedagogue, Department of general technical sciences

Asia International University

Annotation:

This work focuses on the application of selection algorithms for reducing training datasets in classification tasks. The method analyzes the structure of data using distance-based measures and identifies less informative or redundant objects for removal. As a result, a more compact and efficient dataset is obtained, which improves computational performance and supports accurate model learning.

Keywords: Selection algorithms, training set reduction, data preprocessing, feature selection, classification, nearest neighbor, distance metric, data mining, machine learning, data optimization

Selection algorithms play an important role in data analysis and machine learning by helping to identify the most relevant and informative elements from a dataset. Instead of using all available data, which may contain noise, redundancy, or irrelevant information, these algorithms focus on selecting a smaller, high-quality subset.

This process improves the efficiency of models, reduces computational cost, and often increases accuracy. In particular, selection algorithms are essential in tasks such as feature selection and training set reduction, where the goal is to simplify the data while preserving its meaningful structure. As a result, they contribute to building faster, more robust, and more reliable intelligent systems.

The selection algorithm is used to reduce the size of a dataset by removing unnecessary or redundant objects while preserving the most informative ones. This is especially useful before applying methods like K-nearest neighbors (KNN), where too many objects can slow down computation and reduce accuracy.

We start with a dataset $E_0=\{S_1,S_2,...,m\}$, where each object is described by a set of features and belongs to one of two classes. The goal is to construct a smaller subset by eliminating objects that do not contribute significantly to classification.

For each object S_i, two important values are calculated. The first is the radius $r(S_i)$, which is the distance from S_i to the nearest object belonging to the opposite class. This value defines a kind of "safety zone" around the object: inside this radius, there are only objects of the same class.

The second value is $g(S_i)$, which represents the number of objects of the same class that lie within this radius. In other words, it counts how many "friendly neighbors" surround the object.

After computing these values for all objects, the algorithm processes them in order of increasing radius. This means that objects closest to the boundary between classes are considered first, since they are more likely to be ambiguous or less reliable.

For each selected object, the algorithm examines its neighborhood. If the object has no significant neighbors of its own class (meaning it is isolated within its radius), it is removed from the dataset. Such objects are considered uninformative or noisy.

If the object does have neighbors, it is not immediately removed. Instead, its presence affects those neighbors. Specifically, for each neighboring object, the value g is decreased by one, because one of its nearby points is being reconsidered. If this reduction causes any neighbor to have only one remaining neighbor within its radius, that neighbor may also become a candidate for removal.

To prevent the same object from being repeatedly processed, it is marked as "blocked" after being considered.

This process continues iteratively until all objects have been evaluated. At the end, the algorithm produces two sets: one containing the removed objects, and another containing the remaining, selected objects. The remaining subset is smaller, cleaner, and better suited for classification tasks.

In intuitive terms, the algorithm removes points near the class boundaries and keeps those well inside their class regions. As a result, it reduces noise, simplifies the dataset, and improves the efficiency and accuracy of learning algorithms like KNN. There are many types belong to selection in machine learning.

Filter Methods - These methods evaluate each feature independently and select the most useful ones. They use statistical measures to score features and then keep the best ones. Algorithms of the method are Chi-Square Test, Information Gain, Correlation Coefficient[1].

Wrapper Methods - These methods test different combinations of features using a model and choose the best set. They train a model multiple times and evaluate performance. Algorithms are Forward Selection, Backward Elimination, Recursive Feature Elimination (RFE)

Embedded Methods - Feature selection happens inside the model itself. The model automatically decides which features are important during training. Algorithms are Lasso Regression, Decision Tree, Random Forest.

Instance Selection (Data Point Selection) - Instead of selecting features, these methods select important data points (objects). They remove: noisy points, redundant points, boundary (uncertain) points. Algoritmhs are k-Nearest Neighbors based methods, Condensed Nearest Neighbor (CNN), Edited Nearest Neighbor (ENN).

Clustering-Based Selection - Group similar data and select representatives from each group. They divide data into clusters and pick central points. Algoritmhs are K-Means Clustering, DBSCAN[2].

We are given a dataset of objects:

- Each object has features (vector)
- There are two classes: K_1 and K_2
- We want to reduce the dataset by removing unnecessary objects

1. Distance function - $p(S_iS_j)$

This measures how far two objects are (usually Euclidean distance).

2. $r(S_i)$ — nearest enemy distance

- For each object S_i , find the closest object from the opposite class
- That distance is: $r(S_i)$

3. $g(S_i)$— number of same-class neighbors

Count how many objects of the same class are inside radius $r(S_i)$:

$$g(S_i)=|\{S_j \in \text{same class} \mid p(S_iS_j)<r(S_i)\}|$$

Algorithm idea

✧ Compute $r(S_i)$ and $g(S_i)$ for all objects

✧ Sort objects by **smallest** $r(S_i)$ (closest to boundary first)

✧ Take the most "dangerous" object

✧ Check its neighbors: If it is not important → **remove it,** Otherwise → update neighbors[3]

✧ Repeat until done

In results here is a simple working version of this algorithm: import numpy as np

```
def distance(a, b):
    return np.linalg.norm(a - b)
def compute_r(X, y):
    r = []
    for i in range(len(X)):
        distances = []
        for j in range(len(X)):
            if y[i] != y[j]:
                distances.append(distance(X[i], X[j]))
        r.append(min(distances))
    return np.array(r)
def compute_g(X, y, r):
    g = []
```

```
    for i in range(len(X)):
        count = 0
        for j in range(len(X)):
            if y[i] == y[j] and distance(X[i], X[j]) < r[i]:
                count += 1
        g.append(count)
    return np.array(g)
def selection_algorithm(X, y):
    X = list(X)
    y = list(y)

    removed = []
    while True:
        if len(X) == 0:
            break
        r = compute_r(np.array(X), np.array(y))
        g = compute_g(np.array(X), np.array(y), r)
        idx = np.argmin(r)
        Si = X[idx]
        yi = y[idx]
        L = []
        for j in range(len(X)):
            if y[j] == yi and distance(X[j], Si) < r[j] and g[j] > 1:
                L.append(j)
        if len(L) <= 1:
            removed.append((Si, yi))
            X.pop(idx)
            y.pop(idx)
```

```
        else:
            # update neighbors
            for j in L:
                g[j] -= 1
                if g[j] == 1:
                    removed.append((X[j], y[j]))
        if len(X) == 0:
            break
    return removed, X
X = np.array([
    [1, 2], [2, 2], [2, 3],
    [8, 8], [9, 8], [8, 9]])
y = np.array([0, 0, 0, 1, 1, 1])
removed, remaining = selection_algorithm(X, y)
print("Removed objects:", removed)
print("Remaining objects:", remaining)
```

In conclusion, $r(S_i)$ → distance to nearest enemy, $g(S_i)$ → number of same-class neighbors. Remove objects: close to boundary, or redundant. In short, The algorithm selects the most informative and reliable objects by removing those that are redundant or too close to class boundaries, resulting in a cleaner and more efficient training dataset.

References:

1. Trevor Hastie, Robert Tibshirani, Jerome Friedman. “The Elements of Statistical Learning: Data Mining, Inference, and Prediction.” Springer, 2009.
2. Christopher M. Bishop. “Pattern Recognition and Machine Learning.” Springer, 2006.
3. Wes McKinney. “Data Structures for Statistical Computing in Python.” Proceedings of the 9th Python in Science Conference, 2010.
4. Richard O. Duda, Peter E. Hart, David G. Stork. “Pattern Classification.” Wiley-Interscience, 2001.
5. Tom M. Mitchell. “Machine Learning.” McGraw-Hill, 1997.
6. Ian Goodfellow, Yoshua Bengio, Aaron Courville. “Deep Learning.” MIT Press, 2016.
7. Kevin P. Murphy. “Machine Learning: A Probabilistic Perspective.” MIT Press, 2012.
8. Ethem Alpaydin. “Introduction to Machine Learning.” MIT Press, 2020.