Machine Learning

Non-parametric Classification: k-NN Method and its variants. Common Metrics. Classification Mean Error.

Aleksandr Petiushko

ML Research

January 29th, 2024

Content

1. Classification Mean Error

Content

1. Classification Mean Error
2. Euclidean and Manhattan Distance

Content

1. Classification Mean Error
2. Euclidean and Manhattan Distance
3. 1-NN

Content

Content

1. Classification Mean Error
2. Euclidean and Manhattan Distance
3. 1-NN
4. k-NN
5. Weighted k-NN

Content

Mean error

- X – set of objects descriptions, Y – set of objects labels
- Unknown target dependency: mapping $y : X \to Y$
- Finite training set: $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$, so as $y_i = y(x_i)$
- Finite test set: $X_t^q = \{(x_1, y_1), \dots, (x_q, y_q)\}$, so as $y_i = y(x_i)$

Mean error

- X – set of objects descriptions, Y – set of objects labels
- Unknown target dependency: mapping $y : X \to Y$
- Finite training set: $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$, so as $y_i = y(x_i)$
- Finite test set: $X_t^q = \{(x_1, y_1), \dots, (x_q, y_q)\}$, so as $y_i = y(x_i)$

Mean error for classification

The main goal is to train an algorithm $a : X \to Y$ on the train set X^m so as the **mean error** on the test set is minimal: $R(a, X_t^q) = \frac{1}{q} \sum_{i=1}^{q} [a(x_i) \neq y(x_i)] \to min_a$

Common metrics

- Object $x \in X$ is represented in the R^n space: $x = (x^1, \dots, x^n)$ — n-dimensional vector
  - E.g., points on the XY-plane are from R^2: $x = (x_1, x_2)$

Common metrics

- Object $x \in X$ is represented in the R^n space: $x = (x^1, \ldots, x^n)$ — n-dimensional vector
  - E.g., points on the XY-plane are from R^2: $x = (x_1, x_2)$

Euclidean metric

Euclidean, or L_2-distance, between 2 points x and y from R^n is:
$d_2(x, y) = ||x - y||_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

Common metrics

- Object $x \in X$ is represented in the R^n space: $x = (x^1, \ldots, x^n)$ — n-dimensional vector
  - E.g., points on the XY-plane are from R^2: $x = (x_1, x_2)$

Euclidean metric

Euclidean, or L_2-distance, between 2 points x and y from R^n is:
$d_2(x, y) = ||x - y||_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

Manhattan metric

Manhattan, or L_1-distance, between 2 points x and y from R^n is:
$d_1(x, y) = ||x - y||_1 = \sum_{i=1}^{n} |x_i - y_i|$

Parametric and non-parametric machine learning methods

Parametric methods

- Assumption: the dependency being sought does depends on some parameters
- Parameters are found as a solution of the optimization problem

Parametric and non-parametric machine learning methods

Parametric methods

- Assumption: the dependency being sought does depends on some parameters
- Parameters are found as a solution of the optimization problem

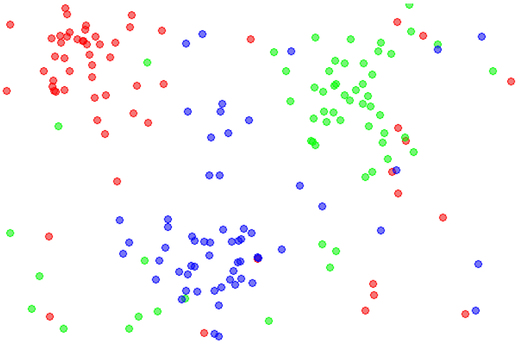
Non-parametric methods

Nonparametric methods are methods that are not parametric methods.

- Examples: Metric algorithms, kernel methods

Basic Assumption

- "Close" objects *usually* lie in the same class
- Proximity is specified by the metric
- Typical example [1]



[1] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

Nearest neighbor (1-NN, NN) method

- Method parameter: metric
- Algorithm: using a given metric, we look for the nearest object in the training set and classify the object with the same class as the found nearest neighbor

Nearest neighbor (1-NN, NN) method

- Method parameter: metric
- Algorithm: using a given metric, we look for the nearest object in the training set and classify the object with the same class as the found nearest neighbor

Advantages

- Ease of implementation (there is no training procedure as such in the naive implementation)
- Good interpretability

Nearest neighbor (1-NN, NN) method

- Method parameter: metric
- Algorithm: using a given metric, we look for the nearest object in the training set and classify the object with the same class as the found nearest neighbor

Advantages

- Ease of implementation (there is no training procedure as such in the naive implementation)
- Good interpretability

Disadvantages

- Instability to outliers
- Ambiguity of classification at equal distances to two objects
- The need to store the entire training set
- The search algorithm is computationally complex (if the training sample is quite large)
- Distance value is not taken into account

k-nearest neighbors (k-NN, kNN) method

- Method parameter: metric, **k**
- Algorithm: using a given metric, we search for the **k** closest objects in the training set and classify the object as the **majority class** of the **k** objects

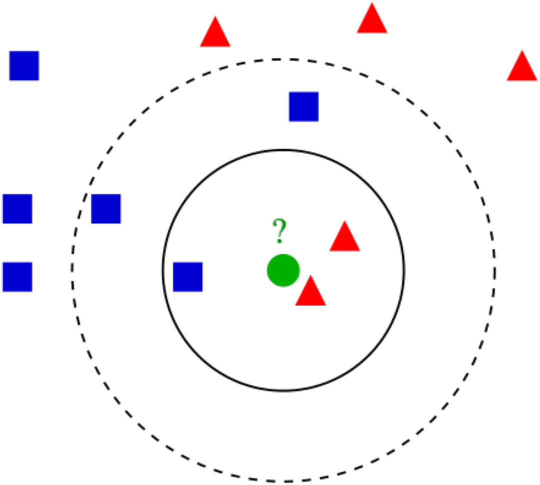
Advantages

- Ease of implementation
- Good interpretability
- Parameter **k** can be optimized using cross-validation

Disadvantages

- ~~Instability to outliers~~
- ~~Classification ambiguity at equal distances to two objects~~
- The need to store the entire training set
- The search algorithm is computationally complex (if the training sample is quite large)
- Distance value is not taken into account

k-NN method



Weighted k-NN method

- Method parameters: metric, k, **weights**
- Algorithm: using a given metric, we look for the k closest objects in the training set and classify the object by weighted voting

Advantages

- Ease of implementation
- Good interpretability
- Parameter k can be optimized using cross-validation

Disadvantages

- ~~Instability to outliers~~
- ~~Classification ambiguity at equal distances to two objects~~
- The need to store the entire training set
- The search algorithm is computationally complex (if the training sample is quite large)
- ~~Distance value is not taken into account~~

Weighted k-NN: choosing weights

- Weights are dependent on the index in the ordered list

Weighted k-NN: choosing weights

- Weights are dependent on the index in the ordered list
  - Linearly decreasing weights

Weighted k-NN: choosing weights

- Weights are dependent on the index in the ordered list
  - Linearly decreasing weights
  - Exponentially decreasing weights

Weighted k-NN: choosing weights

- Weights are dependent on the index in the ordered list
  - Linearly decreasing weights
  - Exponentially decreasing weights
  - Any non-increasing function of the ordinal number

Weighted k-NN: choosing weights

- Weights are dependent on the index in the ordered list
  - Linearly decreasing weights
  - Exponentially decreasing weights
  - Any non-increasing function of the ordinal number
- Weights are dependent on distance

Weighted k-NN: choosing weights

- Weights are dependent on the index in the ordered list
  - Linearly decreasing weights
  - Exponentially decreasing weights
  - Any non-increasing function of the ordinal number
- Weights are dependent on distance
  - Any non-increasing function of distance

Weighted k-NN: choosing weights

- Weights are dependent on the index in the ordered list
  - Linearly decreasing weights
  - Exponentially decreasing weights
  - Any non-increasing function of the ordinal number
- Weights are dependent on distance
  - Any non-increasing function of distance
- Fixed object weights

Weighted k-NN among a set of templates

- Method parameters: metric, k, weights, **template selection method**
- Algorithm: using a given metric, we look for the k closest objects among the templates selected from the training set and classify the object by weighted voting

Advantages

- Ease of implementation
- Good interpretability
- Parameter k can be optimized using cross-validation

Disadvantages

- ~~Instability to outliers~~
- ~~Classification ambiguity at equal distances to two objects~~
- ~~Need to store the entire training set~~
- ~~The search algorithm is computationally complex~~
- ~~Distance value is not taken into account~~

Template selection

Task

Get approximately the same quality of the algorithm with less stored data.
It is even possible to obtain an improvement in quality because outliers will be removed during the template selection process.

Ideas

- Object clustering
- Greedy algorithm

Template selection by k-means clustering method

Task

$$V = \sum_{i=1}^{k} \sum_{x \in S_i} (x - \mu_i)^2 \to \min_{S_i},$$

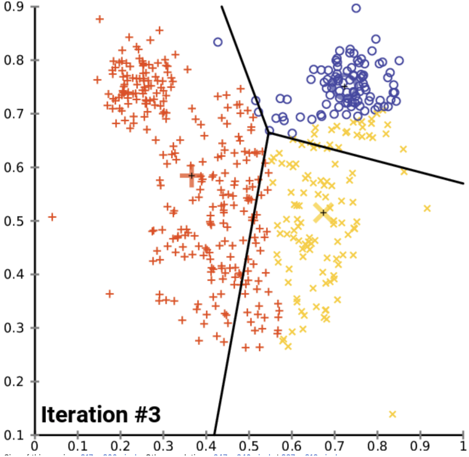
where k is the number of clusters, S_i is the resulting clusters, μ_i is the center of mass of the S_i cluster.

Algorithm

1. k elements are randomly selected from the sample and declared as centroids
2. For fixed **centroids**, each sample element belongs to one of the clusters
3. For fixed **clusters**, centroids are calculated
4. Steps 2 and 3 are repeated until convergence (or exhausting the computation/time budget)

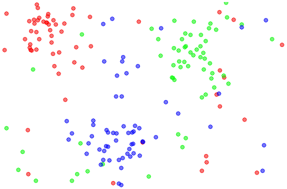
Visualization of k-means

Click to see the gif:



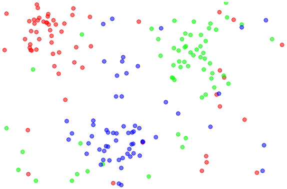
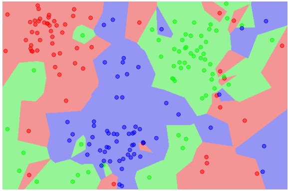
Examples of 1-NN w/o and w/ template selection

- 1st image: dataset
- 2nd image: 1-NN w/o template selection
- 3rd image: templates
- 4th image: 1-NN w/ template selection



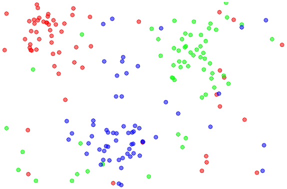
Examples of 1-NN w/o and w/ template selection

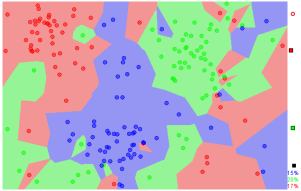
- 1st image: dataset
- 2nd image: 1-NN w/o template selection
- 3rd image: templates
- 4th image: 1-NN w/ template selection

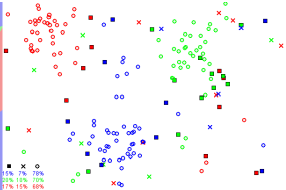



Examples of 1-NN w/o and w/ template selection

- 1st image: dataset
- 2nd image: 1-NN w/o template selection
- 3rd image: templates
- 4th image: 1-NN w/ template selection

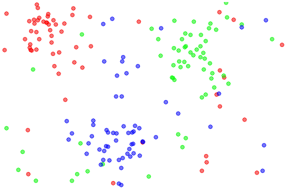


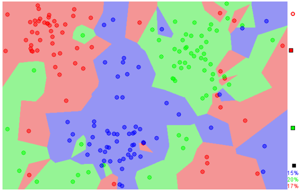


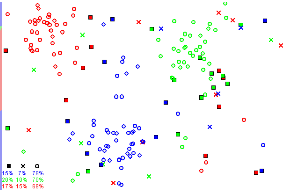


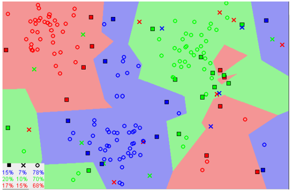
Examples of 1-NN w/o and w/ template selection

- 1st image: dataset
- 2nd image: 1-NN w/o template selection
- 3rd image: templates
- 4th image: 1-NN w/ template selection









Additional modification: RadiusNN

Idea

Sometimes it makes sense to look for neighbors at a distance no greater than some radius r

Parameter r

Instead of the input parameter for the number of neighbors, the radius is used

Conclusion

- Nearest neighbors method is a simple and well-interpreted classification method

Conclusion

- Nearest neighbors method is a simple and well-interpreted classification method
- The method has a large number of variations for customization

Conclusion

- Nearest neighbors method is a simple and well-interpreted classification method
- The method has a large number of variations for customization
  - Selection of metrics (metric learning)

Conclusion

- Nearest neighbors method is a simple and well-interpreted classification method
- The method has a large number of variations for customization
  - Selection of metrics (metric learning)
  - Number of nearest neighbors

Conclusion

- Nearest neighbors method is a simple and well-interpreted classification method
- The method has a large number of variations for customization
  - Selection of metrics (metric learning)
  - Number of nearest neighbors
  - Weights in the weighted version of the method

Conclusion

- Nearest neighbors method is a simple and well-interpreted classification method
- The method has a large number of variations for customization
  - Selection of metrics (metric learning)
  - Number of nearest neighbors
  - Weights in the weighted version of the method
  - Algorithm for selecting templates

Thank you!