



K Nearest Neighbor

📖 K Nearest Neighbor algorithm is a simple supervised learning algorithm that uses the K items from the training set that is the closest to a new item to predict the label of the new item: [Link](#), [Wikipedia](#).

⇒ 1 nearest neighbor copies the label of the closest item.

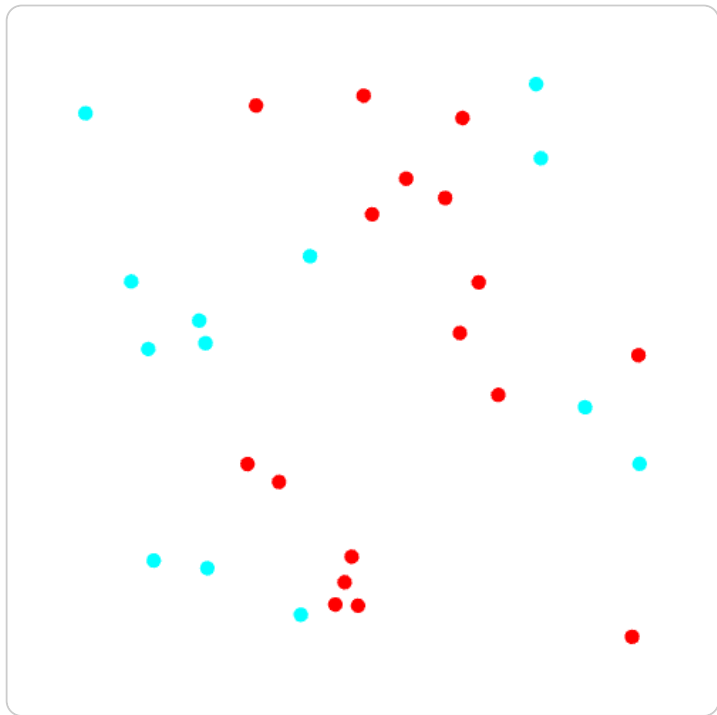
⇒ 3 nearest neighbor finds the majority label of the three closest items.

⇒ N nearest neighbor uses the majority label of the training set (of size N) to predict the label of every new item.

▼ TopHat Discussion

ID: Confirm

📖 [1 points] Find the value of K for K nearest neighbor that is the most appropriate for the dataset. Click on an existing point to perform leave-one-out cross validation, and click on a new point to find the nearest neighbor.



K:



Distance Measures

📖 The distance measure used in K nearest neighbor can be any of the L_p distances.

⇒ L_1 Manhattan distance.

⇒ L_2 Euclidean distance.

⇒ L_∞ Maximum distance from all features.



Training Set Accuracy

- For 1NN, the accuracy of the prediction on the training set is **always 100 percent**.
- When comparing the accuracy of KNN for different values of K (called hyperparameter tuning), training set accuracy is not a great measure.
- K fold cross validation is often used instead to measure the performance of a supervised learning algorithm on the training set.
 - ⇒ The training set is divided into K groups (K can be different from the K in KNN).
 - ⇒ Train the model on K - 1 groups and compute the accuracy on the remaining 1 group.
 - ⇒ Repeat this process K times.
- K fold cross validation with $K = 1$ is called Leave One Out Cross Validation (LOOCV).

▼ TopHat Quiz

ID: Confirm

■ [4 points] Given the following training data, what is the 6 fold cross validation accuracy if 1NN (Nearest Neighbor) classifier with Manhattan distance is used. The first fold is the first 1 instances, the second fold is the next 1 instances, etc. Break the tie (in distance) by using the instance with the smaller index. Enter a number between 0 and 1.

x_i	-5	-4	-3	-2	8	10
y_i	0	0	1	0	0	1

■ Answer: Calculate

$K=N$ is called LOOCV, here $K=N=6$ (# of datapoint)

Leave -5 out: train on -4, -3, -2, 8, 10, test on -5, nearest neighbor of -5 is -4, predict 0.

Leave -4 out: nn is -5, predict 0.

Leave -3 out: nn is -4, predict 0 (wrong)

Leave -2 out: nn is -3, predict 1 (wrong)

Leave 8 out: nn is 10, predict 1 (wrong)

Leave 10 out: nn is 8, predict 0 (wrong)

LOOCV accuracy: $2/6 = 33.3\%$

LOOCV for 3NN:

Leave -5 out: nn is {-4, -3, -2}, predict 0.

Leave -4 out: nn is {-5, -3, -2}, predict 0

Leave -3 out: nn is {-5, -4, -2}, predict 0 (wrong)

Leave -2 out: nn is {-3, -4, -3}, predict 0


Leave 8 out: nn is {-3, -2, 10}, predict 1 (wrong)

Leave 10 out: nn is {-3, -2, 8}, predict 0 (wrong)

LOOCV accuracy: $3/6 = 50\%$




Decision Tree

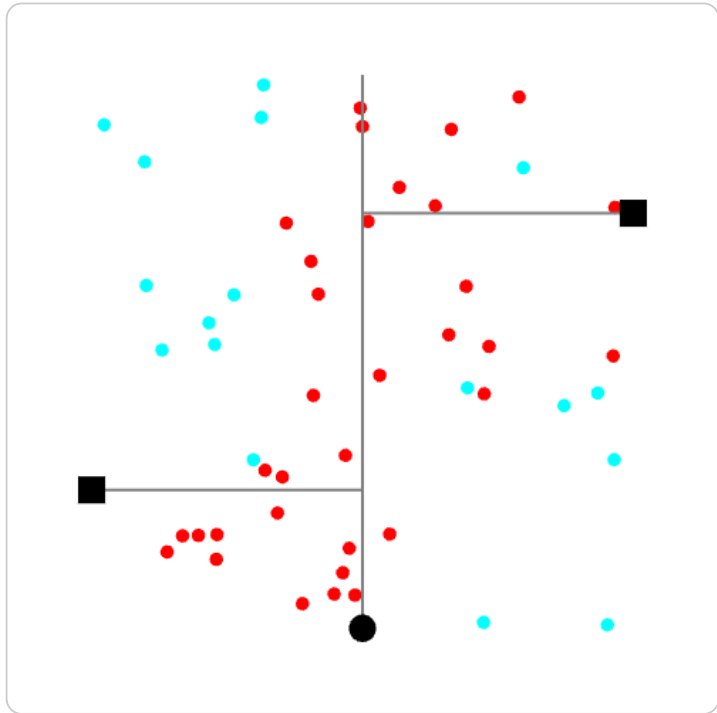
 Another simple supervised learning algorithm is decision trees.

- ⇒ Find the feature that is the most informative.
- ⇒ Split the training set into subsets based on this feature.
- ⇒ Repeat on each of the subset recursively until all features or labels in the subset are the same.

▼ TopHat Discussion

ID: Confirm

 [1 points] Find the thresholds for the decision tree that optimize (i) the total number of mistakes and (ii) the information gain (iterative). Move the black circle to the threshold for the first feature, and move the two black squares to the thresholds for the second feature.



Total number of mistakes: ???

Information gain for the first feature: ???

Information gain for the second feature (left): ???

Information gain for the first feature (right): ???



Measure of Uncertainty

Let p_0 be the fraction of items in a training with label 0 and p_1 be the fraction of items with label 1.

⇒ If $p_0 = 0, p_1 = 1$, the outcome is certain, so there is no uncertainty, the measure of uncertainty should be 0.

⇒ If $p_0 = 1, p_1 = 0$, the outcome is certain, so there is no uncertainty, the measure of uncertainty should be 0.

⇒ If $p_0 = \frac{1}{2}, p_1 = \frac{1}{2}$, the outcome is the most uncertain, so the measure of uncertainty should be at its maximum value, for example 1.

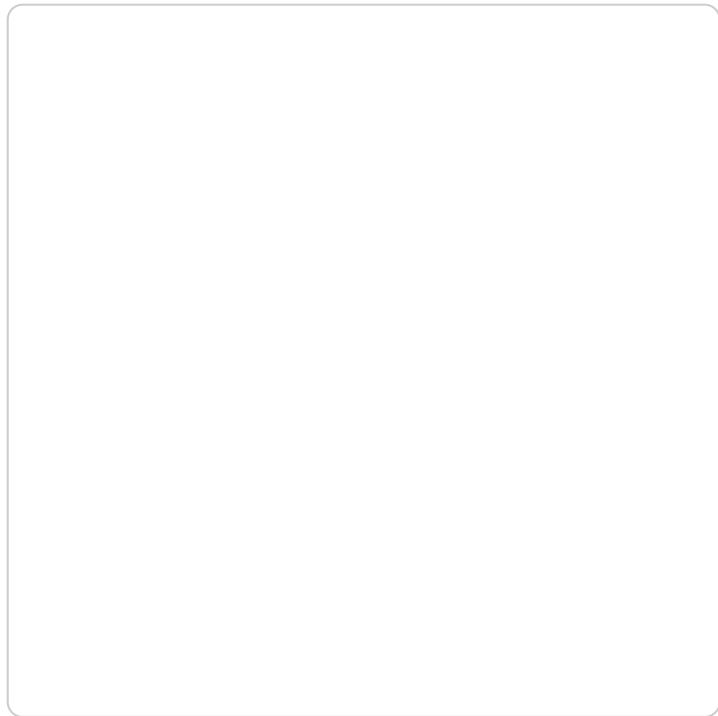
One measure of uncertainty that satisfies the above condition is **entropy**:

$$H = p_0 \log_2 \left(\frac{1}{p_0} \right) + p_1 \log_2 \left(\frac{1}{p_1} \right) \text{ or } H = -p_0 \log_2(p_0) - p_1 \log_2(p_1).$$

The realized value of something uncertain is more informative than the value of something certain.

▼ Example

Plot the function: $-p * \frac{\log(p)}{\log(2)} - (1-p) * \frac{\log(1-p)}{\log(2)}$ as a function of p from 0 to 1 Plot .





Entropy

■ In general, if there are K classes ($y = \{1, 2, \dots, K\}$), and p_y is the fraction of the training set with label y , then the entropy of y is $H(y) = -p_1 \log_2(p_1) - p_2 \log_2(p_2) - \dots - p_K \log_2(p_K)$.

■ Conditional entropy is the entropy of the conditional distribution:

$H(y|x) = q_1 H(y|x=1) + q_2 H(y|x=2) + \dots + q_{K_x} H(y|x=K_x)$, where K_x is the number of possible values of the feature x and q_x is the fraction of training data with feature x .

$H(y|x=k) = -p_{y|1} \log_2(p_{y|1}) - p_{y|2} \log_2(p_{y|2}) - \dots - p_{y|K_x} \log_2(p_{y|K_x})$, where $p_{y|x}$ is the fraction of training data with label y among the items with feature x .



Information Gain

📖 The information gain from a feature x is defined as the difference between the entropy of the label and the conditional entropy of the label given that feature: $I(y|x) = H(y) - H(y|x)$.

📖 The larger the information gain, the larger the reduction in uncertainty, and the better predictor the feature is.

📖 A decision tree iteratively splits the training set based on the feature with the largest information gain. This algorithm is called ID3 (Iterative Dichotomiser 3): [Wikipedia](#).

⇒ Find feature j so that $I(y|x_{ij})$ is the largest.

⇒ Split the training set into the set with $x_{ij} = 1, x_{ij} = 2, \dots, x_{ij} = K_j$.

⇒ Repeat the process on each of the subsets to create a tree.

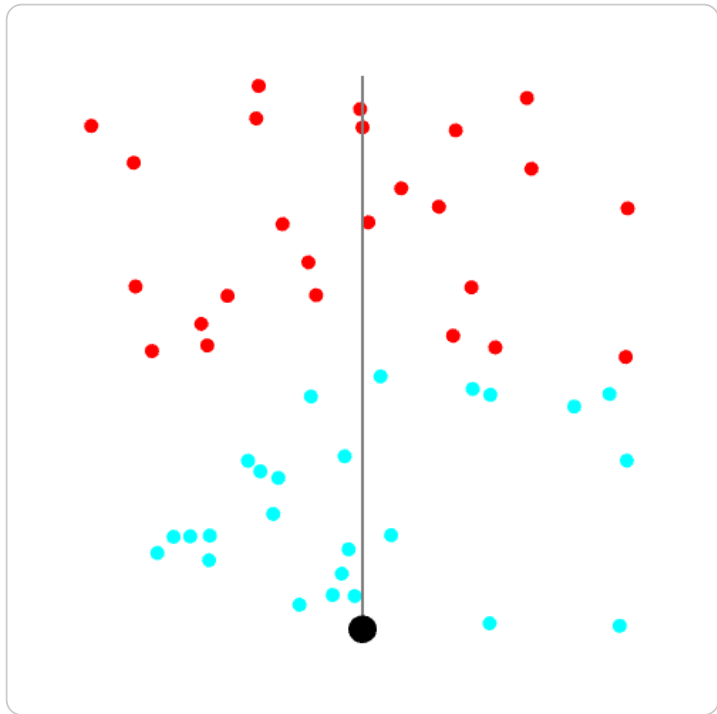
📖 For continuous features, construct all possible splits and find the one that yields the largest information gain: this is the same as creating new variables $z_{ij} = 1$ if $x_{ij} \leq t_j$ and $z_{ij} = 0$ if $x_{ij} > t_j$.

⇒ In practice, the efficient way to create the binary splits uses the midpoint between items with different labels.

▼ TopHat Discussion

ID: Confirm

📖 [1 points] Find the threshold for the decision stump that maximize the information gain. Move the black circle to the threshold.



Information gain: ???

▼ TopHat Quiz

(Past Exam Question) ID: Confirm



answer your question "Will door 1 be opened?" What's the information gain (also called mutual information) between Pennywise's answer and your encounter with a monster?



Answer:



Calculate

[4 points] It has a house with many doors. A random door is about to be opened with equal probability. Doors 1 to 4 have monsters that eat people. Doors 5 to 7 are safe. With sufficient bribe, Pennywise will answer your question "Will door 1 be opened?" What's the information gain (also called mutual information) between Pennywise's answer and your encounter with a monster?

$IG(y | x) = IG(x | y)$, mutual information.

$y = 1$ if monster, $y = 0$ if no monster.

$x = 1$ if door 1 is opened, $x = 0$ if door 1 is not opened.

$H(y) = -4/7 * \log(4/7)/\log(2) - 3/7 * \log(3/7)/\log(2) = 0.9852$,

given $x = 0$, doors 2 to 7 will be opened.

$H(y|x=0) = -1/2 * \log(1/2)/\log(2) - 1/2 * \log(1/2)/\log(2) = 1$

given $x = 1$, door 1 will be opened \Rightarrow I am 100% certain there will be a monster.

$H(y|x=1) = -1 * \log(1)/\log(2) - 0 * \log(0)/\log(2) = 0$, (assume $0 \log(0) = \lim_{x \rightarrow 0} x \log x = 0$)

$H(y|x) = 6/7 * 1 + 1/7 * 0 = 6/7$.

$IG(y|x) = 0.9852 - 6/7 = 0.1280$.

Answer:

Calculate





Pruning

📖 Decision trees can be pruned by replacing a subtree by a leaf when the accuracy on a validation set with the leaf is equal or higher than the accuracy with the subtree. This method is called Reduced Error Pruning: [Wikipedia](#).

⇒ A validation set is a subset of the training set that is set aside when training the decision tree and only used for pruning the tree.

⇒ The items use to train the decision tree cannot be used to prune the tree.



Random Forest

📖 Smaller training sets can be created by sampling from the complete training set, and different decision trees can be trained on these smaller training sets. This is called bagging (or Bootstrap AGGregatING):

[Link, Wikipedia.](#)

⇒ Training items are sampled with replacement.

⇒ Features are sampled without replacement.

📖 The label of a new item can be predicted based on the majority vote from the decision trees training on these smaller training sets. These trees form a random forest: [Wikipedia.](#)



Adaptive Boosting

📖 Decision trees can also be trained sequentially. The items that are classified incorrectly by the previous trees are made more important when training the next decision tree.

📖 Each training item has a weight representing how important they are when training each decision tree, and the weights can be updated based on the error made by the previous decision trees. This is called AdaBoost (ADAPtive BOOSTing): [Wikipedia](#).

⇒ Viola-Jones algorithm uses adaptively boosted decision stumps (decision trees with depth 1). More on this in the computer vision lecture: [Wikipedia](#).

📖 Notes and code adapted from the course taught by Professors Jerry Zhu, Yingyu Liang, and Charles Dyer.

📖 Please use Ctrl+F5 or Shift+F5 or Shift+Command+R or Incognito mode or Private Browsing to refresh the cached JavaScript.

📖 Anonymous feedback can be submitted to: [Form](#).

Prev: [L3](#), Next: [L5](#)

Last Updated: June 26, 2024 at 2:55 AM



UNIVERSITY OF WISCONSIN-MADISON



Powered by w3.css