

Lecture 8: Classification: kNN, Naïve Bayes, and Bayesian Networks

Goals:

- Describe Nearest Neighbor Classifier
- Implement kNN algorithm
- Describe Bayesian learning
- Implement Naïve Bayes algorithm
- Describe Bayesian network models

Part I: Nearest Neighbor Classifier and kNN algorithm

Find which training data is closest to the test instance and classify the test instance as that class.

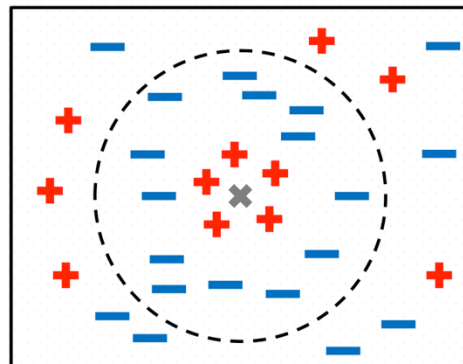
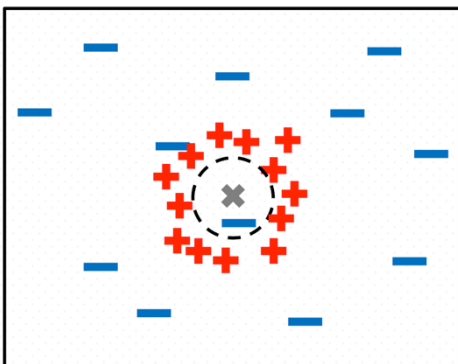
***k*-Nearest Neighbor (*k*NN) Classifier:** Find the k training instances that are closest to the test instance, and classify the test instance as the majority class of the k nearest training instances.

The k -nearest neighbor classification algorithm.

- 1: Let k be the number of nearest neighbors and D be the set of training examples.
 - 2: **for** each test example $z = (\mathbf{x}', y')$ **do**
 - 3: Compute $d(\mathbf{x}', \mathbf{x})$, the distance between z and every example, $(\mathbf{x}, y) \in D$.
 - 4: Select $D_z \subseteq D$, the set of k closest training examples to z .
 - 5: $y' = \operatorname{argmax}_v \sum_{(x_i, y_i) \in D_z} I(v = y_i)$
 - 6: **end for**
-

Choosing k :

If k is too small, then the nearest-neighbor classifier may be susceptible to overfitting because of noise in the training data. If k is too large, the nearest-neighbor classifier may misclassify the test instance because its list of nearest neighbors may include data points that are located far away from its neighborhood.



kNN Classification:

Take the majority vote of class labels among the k-nearest neighbors:

$$\text{Majority voting: } y = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_Z} I(v = y_i)$$

where v is a class label, y_i is the class label for one of the nearest neighbors, and $I(\cdot)$ is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

kNN Regression:

Predict the average value of the class value of the k-nearest neighbors.

$$\text{Average value: } y = \frac{1}{|\{(\mathbf{x}_i, y_i) \in D_Z\}|} \sum_{(\mathbf{x}_i, y_i) \in D_Z} y_i$$

where v is a class label and y_i is the class label for one of the nearest neighbors.

Advantages:

- Generate their predictions based on local information, so it can produce **arbitrarily shaped decision boundaries**.
- **Very efficient in model induction** (training)
- Only store the training data.
- Note that 1NN and kNN are equally efficient
- Retrieving the k nearest neighbors is (almost) no more expensive than retrieving a single nearest neighbor
- k nearest neighbors can be maintained in a queue.

Disadvantages:

- Can easily produce **wrong predictions** without appropriate data preprocessing.
- **Not particularly efficient** in testing
- Computation of distance measure to every training instance

Part II: Bayesian learning

Bayes' Theorem:

- $P(H)$ (prior probability): the initial probability
- $P(X)$ (evidence): probability that sample data is observed
- $P(X|H)$ (likelihood): the probability of observing the sample X , given that the hypothesis holds

Given training data \mathbf{X} , *posteriori probability of a hypothesis* H , $P(H|\mathbf{X})$, follows the Bayes' theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

Informally, this can be viewed as **posteriori = likelihood * prior / evidence**

Predicts \mathbf{X} belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|\mathbf{X})$ for all the k classes.

Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n -D attribute vector $\mathbf{X} = (x_1, x_2, \dots, x_n)$.

Suppose there are m classes C_1, C_2, \dots, C_m .

Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$.

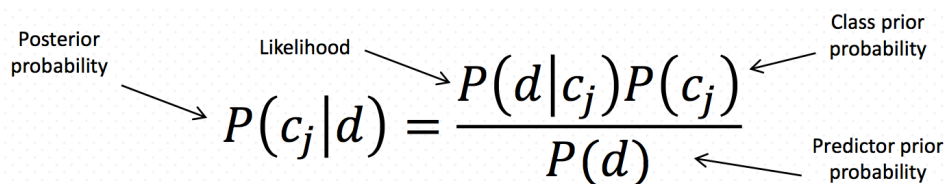
This can be derived from Bayes' theorem:

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

Since $P(\mathbf{X})$ is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized.



$$P(c_j|d) = \frac{P(d|c_j)P(c_j)}{P(d)}$$

$P(c_j|d)$ = probability of instance d being in class c_j

$P(d|c_j)$ = probability of generating instance d given class c_j

$P(c_j)$ = probability of occurrence of class c_j

$P(d)$ = probability of instance d occurring

A simplified (Naïve) assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^n P(x_k | C_i) = P(x_1 | C_i) \times P(x_2 | C_i) \times \dots \times P(x_n | C_i)$$

Exercise:

(Playing Tennis) Can you build a Tennis Weather-to-Play Naïve Bayes classifier to predict if the testing instance leads to “Yes” or “No”?

		Outlook	Temperature	Humidity	Windy	Label: Play?
1	9/1/17	Sunny	Hot	High	"False"	No
2	9/8/17	Sunny	Hot	High	"True"	No
3	9/15/17	Overcast	Hot	High	"False"	Yes
4	9/22/17	Rainy	Mild	High	"False"	Yes
5	9/29/17	Rainy	Cool	Normal	"False"	Yes
6	10/1/17	Rainy	Cool	Normal	"True"	No
7	10/8/17	Overcast	Cool	Normal	"True"	Yes
8	10/15/17	Sunny	Mild	High	"False"	No
9	10/22/17	Sunny	Cool	Normal	"False"	Yes
10	10/29/17	Rainy	Mild	Normal	"False"	Yes
11	11/1/17	Sunny	Mild	Normal	"True"	Yes
12	11/8/17	Overcast	Mild	High	"True"	Yes
13	11/15/17	Overcast	Hot	Normal	"False"	Yes
14	11/22/17	Rainy	Mild	High	"True"	No
15	11/29/17	Rainy	Hot	High	"False"	?

We have

14 training instances and **1 testing instance**

4 attributes:

- (a) 3-value attribute (Sunny/Overcast/Rainy),
- (b) 3-value attribute (Hot/Mild/Cool),
- (c) 2-value attribute (High/Normal),
- (d) 2-value attribute (True/False)

Use Naïve Bayes to predict the label of the testing instance ☺

Name:

NetID:

Please write down whatever question you have about this course:

Solution

H: Hypothesis on class label; **X**: Data samples

1. $P(H)$: Prior Probability

$P(\text{Play?} = \text{"yes"}) =$

$P(\text{Play?} = \text{"no"}) =$

2. $P(\mathbf{X}|H)$: Likelihood

$P(\text{Outlook} = \text{Rainy} \mid \text{Play?} = \text{"yes"}) =$

$P(\text{Outlook} = \text{Rainy} \mid \text{Play?} = \text{"no"}) =$

$P(\text{Temperature} = \text{Hot} \mid \text{Play?} = \text{"yes"}) =$

$P(\text{Temperature} = \text{Hot} \mid \text{Play?} = \text{"no"}) =$

$P(\text{Humidity} = \text{High} \mid \text{Play?} = \text{"yes"}) =$

$P(\text{Humidity} = \text{High} \mid \text{Play?} = \text{"no"}) =$

$P(\text{Windy} = \text{"False"} \mid \text{Play?} = \text{"yes"}) =$

$P(\text{Windy} = \text{"False"} \mid \text{Play?} = \text{"no"}) =$

$P(\mathbf{X} \mid \text{Play?} = \text{"yes"}) =$

$P(\mathbf{X} \mid \text{Play?} = \text{"no"}) =$

3. $P(\mathbf{X})$: Evidence

4. $P(H|\mathbf{X})$: Posteriori Probability

$P(\text{Play?} = \text{"yes"} \mid \mathbf{X}) =$

$P(\text{Play?} = \text{"no"} \mid \mathbf{X}) =$

Conclusion: