Quiz: Answer

1 2 3
4 5 6

$$\frac{1 + 2\times4}{3} = 3.$$

$$\frac{2 + 2.5}{3} = 4.$$

$$\frac{1 + 2\times2 + 3}{4}.$$

Introduction to Machine Learning and Data Mining Lecture-5: Naïve Bayes and Nearest Neighbors

Prof. Eugene Chang

Today

- Naïve Bayes classifier
- Decision Tree algorithm revisit
- Nearest Neighbor classifier
- Some slides are based on materials from Prof. Raymond J. Mooney, University of Texas at Austin, and Prof Jiawei Han, University of Illinois at Urbana-Champaign

About Final Projects

similar to the homeworks

- I'm considering adding back the final exam on week-15
- Projects will be optional. Only students with good grasp of Python are encouraged to take on projects.
  - If you want to get grade A and above, you need to do a project and your project need to have meaningful results
  - Basically I want to avoid wasting your and my time on meaningless projects
- Project logistics
  - Teams will be single or 2-person based
  - Duration will be probably 4-5 weeks, probably week-10 (after mid-term) to week-14 (project presentation)
  - I will prepare and announce the topics in week-7

Top 10 Algorithms in KDD (Knowledge Discovery and Data Mining)

- Identified in ICDM'06
- A series of criteria
- Select from 18 nominations and vote from various researchers
- In 10 topics
  - association analysis, classification, clustering, statistical learning, bagging and boosting, sequential patterns, integrated mining, rough sets, link mining, and graph mining

Top 10 Algorithms in KDD

read the Ref document pdf provided by the Prof.

- * C4.5 (decision tree)
- * *k*-means (clustering)
- * Support vector machines
- The Apriori algorithms (Association rule analysis)
- The EM algorithm
- PageRank ← google search.
- * AdaBoost (tree based)
- * *k*NN: *k*-nearest neighbors classification
- * Naive Bayes
- CART: Classification and Regression Tree

Bayes' Theorem: Basics

- Total probability Theorem: $$P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i)$$
- Bayes' Theorem: $$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$
  - Let **X** be a data sample ("*evidence*"): class label is unknown ← feature
  - Let H be a *hypothesis* that X belongs to class C
  - Classification is to determine P(H|**X**), (i.e., *posteriori probability):* the probability that the hypothesis holds given the observed data sample **X**
  - P(H) (*prior probability*): the initial probability
    - E.g., **X** will buy computer, regardless of age, income, …
  - P(**X**): probability that sample data is observed
  - P(**X**|H) (likelihood): the probability of observing the sample **X**, given that the hypothesis holds
    - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Prediction Based on Bayes' Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**)*,* follows the Bayes' theorem

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

- Informally, this can be viewed as

  posteriori = likelihood x prior/evidence

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

- Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

Classification Is to Derive the Maximum Posteriori

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = $(x_1, x_2, \ldots, x_n)$
- Suppose there are *m* classes $C_1, C_2, \ldots, C_m$.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

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

- Since P(X) is constant for all classes, only $P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$ needs to be maximized

Naïve Bayes Classifier

example : credit card application approval

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes): eg: income× age× other prob

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

given each class C_i, compute the prob of X & x belong to C_i iff $P(C_i|x)$ is max.

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

gaussian

and $P(x_k|C_i)$ is $$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayesian Categorization

- If we assume features of an instance are independent **given the category** (***conditionally independent***).

$$P(X \mid \mathrm{Ci}) = P(X_1, X_2, \cdots X_n \mid \mathrm{Ci}) = \prod_{i=1}^{n} P(X_i \mid \mathrm{Ci})$$

- Therefore, we then only need to know $P(X_i \mid C_i)$ for each possible pair of a feature-value and a category.
- If Y and all X_i and binary, this requires specifying only $2n$ parameters:
  - $P(X_i\text{=true} \mid Y\text{=true})$ and $P(X_i\text{=true} \mid Y\text{=false})$ for each X_i
  - $P(X_i\text{=false} \mid Y) = 1 - P(X_i\text{=true} \mid Y)$

calculate X_i = true in both 2 classes of Y = true/false 4 find out max.

- Compared to specifying 2^n parameters without any independence assumptions.

Naïve Bayes Example

- Y (binary class): positive or negative
- X (features)
  - Size: large, medium, small
  - Color: red, green, blue
  - Shape: square, triangle, circle
- Independence among features

Naïve Bayes Example

Probability	positive	negative
P(*Y*)	0.5	0.5
P(small | *Y*)	0.4	0.4
P(medium | *Y*)	0.1	0.2
P(large | *Y*)	0.5	0.4
P(red | *Y*)	0.9	0.3
P(blue | *Y*)	0.05	0.3
P(green | *Y*)	0.05	0.4
P(square | *Y*)	0.05	0.4
P(triangle | *Y*)	0.05	0.3
P(circle | *Y*)	0.9	0.3

Sum = 1

1

1

Test Instance:
<medium, red, circle>

Let $X = (m, r, c)$.
y = positive $\Rightarrow$ $\sim y$ = negative.

$P(y|x) + P(\sim y|x) = 1$

$\Rightarrow \dfrac{P(x|y)P(y)}{P(x)} + \dfrac{P(x|\sim y)P(\sim y)}{P(x)} = 1$

$\Rightarrow P(x|y)P(y) + P(x|\sim y)P(\sim y) = P(x)$.

$\Rightarrow P(m,r,c) = P(m,r,c|p)P(p) + P(m,r,c|n)P(n)$

$= P(m|p)P(r|p)P(c|p)P(p) + P(m|n)P(r|n)P(c|n)P(n)$

$= 0.1 \times 0.9 \times 0.9 \times 0.5 + 0.2 \times 0.3 \times 0.3 \times 0.5$

$= 0.0495$

Method2:

$P(x) = P(x,y) + P(x,\sim y)$

$= P(x,y)P(y) + P(x,\sim y)P(\sim y)$

Naïve Bayes Example

Test Instance: X = <medium, red, circle>

Probability	positive	negative
P(Y)	0.5	0.5
P(medium | Y)	0.1	0.2
P(red | Y)	0.9	0.3
P(circle | Y)	0.9	0.3

Naïve Bayes Example

$P(X): X = (med, red, cir)$

$P(x) = P(x|+) + P(x|-)$

P(positive | *X*) = P(positive)*P(medium | positive)*P(red | positive)*P(circle | positive) / P(*X*)

0.5 * 0.1 * 0.9 * 0.9

= 0.0405 / P(*X*) = 0.0405 / 0.0495 = 0.8181

P(negative | *X*) = P(negative)*P(medium | negative)*P(red | negative)*P(circle | negative) / P(*X*)

0.5 * 0.2 * 0.3 * 0.3

= 0.009 / P(*X*) = 0.009 / 0.0495 = 0.1818

P(positive | *X*) + P(negative | *X*) = 0.0405 / P(*X*) + 0.009 / P(*X*) = 1

P(*X*) = (0.0405 + 0.009) = 0.0495

Naïve Bayes Classifier: Training Dataset

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data to be classified:

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

Question will be: provided X, buy or not buy?

= find max of $P(C_i|X)$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Classifier Example

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643

  P(buys_computer = "no") = 5/14= 0.357

- Compute $P(X|C_i)$ for each class

  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222

  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6

  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444

  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4

  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667

  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2

  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667

  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

goal: find out & compare.

$P(\text{buy}|X)$ v/s $P(\text{not buy}|X)$.

$* \; P(\text{buy}|X) = \frac{P(X|\text{buy}) \, P(\text{buy})}{P(X)}$.

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028

P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Buy

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Estimating Probabilities

- Probabilities are estimated based on observed frequencies in the training data.
- If D contains n_k examples in category y_k, and n_{ijk} of these n_k examples have the jth value for feature X_i, x_{ij}, then:

$$P(X_i = x_{ij} \mid Y = y_k) = \frac{n_{ijk}}{n_k}$$

- However, estimating such probabilities from small training sets is error-prone.
- If due only to chance, a rare feature, X_i, is always false in the training data, $\forall y_k$:P(X_i=true | Y=y_k) = 0.
- If X_i=true then occurs in a test example, X, the result is that $\forall$y$_k$: P(X | Y=y$_k$) = 0 and $\forall$y$_k$: P(Y=y$_k$ | X) = 0

can be a Quiz

Probability Estimation Example

if the training data is limited.

Ex	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

Test Instance *X*: <medium, red, circle>

this prob impacts the result

P(positive | *X*) = 0.5 * 0.0 * 1.0 * 1.0 / P(X) = 0

P(negative | *X*) = 0.5 * 0.0 * 0.5 * 0.5 / P(X) = 0

Use technique call "Smoothing"

Probability	positive	negative
P(*Y*)	0.5	0.5
P(small | *Y*)	0.5	0.5
P(medium | *Y*)	0.0	0.0
P(large | *Y*)	0.5	0.5
P(red | *Y*)	1.0	0.5
P(blue | *Y*)	0.0	0.5
P(green | *Y*)	0.0	0.0
P(square | *Y*)	0.0	0.0
P(triangle | *Y*)	0.0	0.5
P(circle | *Y*)	1.0	0.5

Smoothing

- To account for estimation from small samples, probability estimates are adjusted or *smoothed*.
- Laplace smoothing using an *m*-estimate assumes that each feature is given a prior probability, *p*, that is assumed to have been previously observed in a "virtual" sample of size *m*.

$$\boxed{P(X_i = x_{ij} \mid Y = y_k) = \frac{n_{ijk} + mp}{n_k + m}}$$

- For binary features, *p* is simply assumed to be 0.5.

Laplace Smoothing Example

- Assume training set contains 10 positive examples:
  - 4: small
  - 0: medium
  - 6: large

Assume there is 1 medium, accounted as 1/3 unit in the set.

- Estimate parameters as follows (if m=1, p=1/3)
  - P(small | positive) = (4 + 1/3) / (10 + 1) = 0.394
  - P(medium | positive) = (0 + 1/3) / (10 + 1) = 0.03
  - P(large | positive) = (6 + 1/3) / (10 + 1) = 0.576
  - P(small or medium or large | positive) = 1.0

Comments on Naïve Bayes

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
    - Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier
- How to deal with these dependencies? Bayesian Belief Networks

Decision Tree Induction Algorithm

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in top-down recursive divide-and-conquer
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)

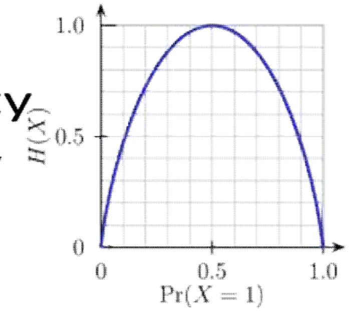
features

  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Brief Review of Entropy

Expected information

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable Y taking m distinct values $\{y_1, \ldots, y_m\}$,
    - $\boxed{H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)}$, where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy => higher uncertainty
    - Lower entropy => lower uncertainty
- Conditional Entropy
  - $H(Y|X) = \sum_x p(x) H(Y|X = x)$



m = 2

Attribute Selection Measure: Information Gain

- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:
- Information needed (after using A to split D into v partitions) to classify D:
- Information gained by branching on attribute A

Entropy of R
= Expected Information of R
= Sum (over x) Prob (R = x) × Information (x).
= Sum (over x) Prob (R = x) × $\log_2$ (1 / Prob (R = x)).
= Sum (over x) − Prob (R = x) × $\log_2$ (Prob (R = x))
= − Sum (over x) Prob (R = x) × $\log_2$ (Prob (R = x)).
$= -\sum_{i=1}^{m} p_i \log_2(p_i)$

$$\boxed{Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)}$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

after split D by A

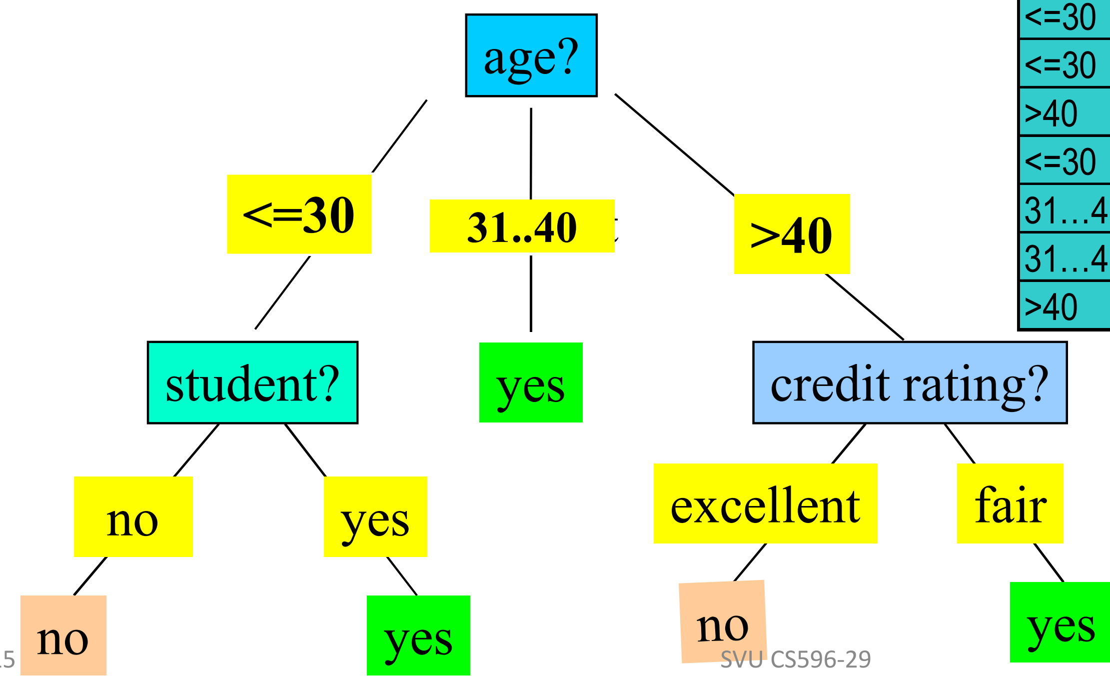
$$\boxed{Gain(A) = Info(D) - Info_A(D)}$$

- Select the attribute with the highest information gain

Decision Tree Results

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3 (Playing Tennis)
- Resulting tree:

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



$I(2,3) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0.9709$

$I(4,0) = -\frac{4}{4}\log_2\left(\frac{4}{4}\right) - 0 = 0$

$I(3,2) = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0.9709$

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

yes no

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

split by buy-computer → yes / no

info needed when we do nothing

Let's split by age:

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

pure

info needed after split by age

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

the 1st split should use age

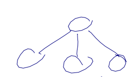
Similarly, (the goal is to define which feature to use to split data)

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

(And continue to calculate for next level with income, student, credit_rating)



We want to reach the bottom faster so the bigger the value of D the better

0.694

0.94

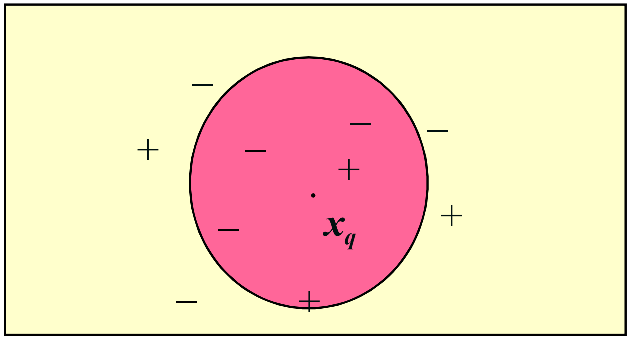
→ HW 2 Prob 2.

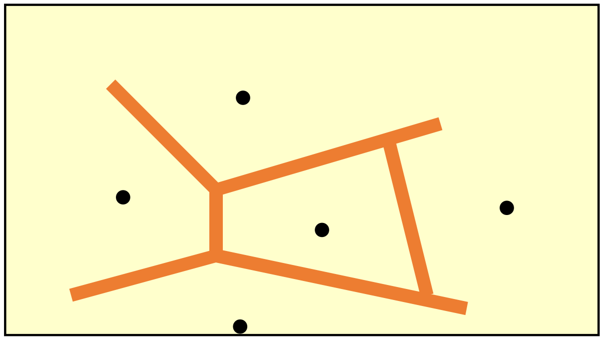
(can use python tool to verify the result inf-gain.py (only binary, need modification))

k-Nearest Neighbor (kNN) Classification

or manhattan_dist

- All instances correspond to points in the n-D space
- The nearest neighbor are defined in terms of Euclidean distance, dist($\mathbf{X_1}$, $\mathbf{X_2}$)
- Target function could be discrete- or real- valued
- For discrete-valued, *k*-NN returns the most common value among the *k* training examples nearest to x_q
- Vonoroi diagram: the decision surface induced by 1-NN for a typical set of training examples





k-Nearest Neighbor (kNN) Classification

- Unlike all the previous learning methods, kNN does not build model from the training data.
- To classify a test instance *d*, define *k*-neighborhood *P* as *k* nearest neighbors of *d*
- Count number *n* of training instances in *P* that belong to class c_j
- Estimate P(c_j|*d*) as *n*/*k*
- No training is needed. Classification time is linear in training set size for each test case.

kNN Algorithm

Algorithm kNN(D, d, k)

1 Compute the distance between d and every example in D;

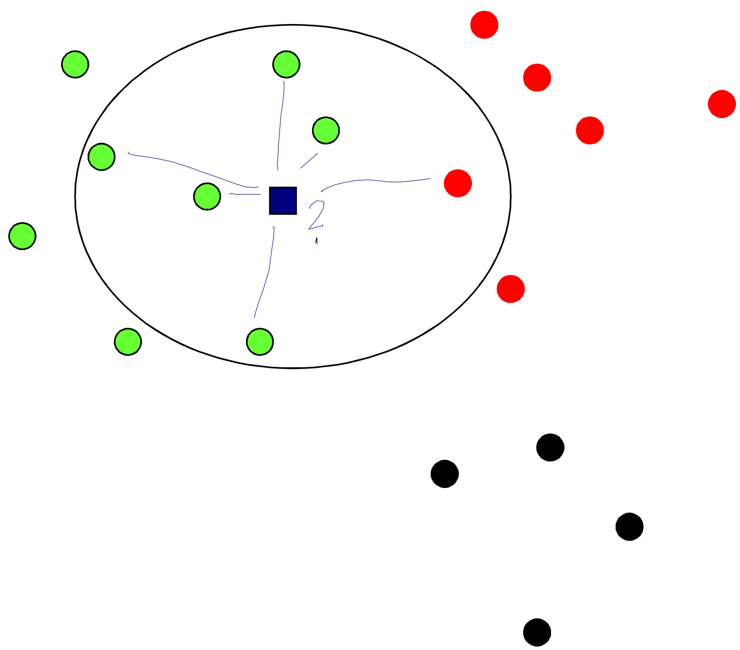
2 Choose the k examples in D that are nearest to d, denote the set by P ($\subseteq D$);

3 Assign d the class that is the most frequent class in P (or the majority class);

- *k* is usually chosen empirically via a validation set or cross-validation by trying a range of *k* values.
- Distance function is crucial, but depends on applications.

6 nearest pnts to the unknon

Example: k=6 (6NN)



Government

Science

Arts

A new point ■

P(science| ■)?

Discussion on the *k*-NN Algorithm

- *k*-NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the *k* neighbors according to their distance to the query x_q
    - Give greater weight to closer neighbors

weight

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

inverse = more weight for closer neighbors

- Robust to noisy data by averaging *k*-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes

Discussions

- kNN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of kNN can be quite strong and in many cases as accurate as those elaborated methods.
- kNN is slow at the classification time
- kNN does not produce an understandable model