Machine Learning

Classification Methods

Bayesian Classification, Nearest Neighbor, Ensemble Methods

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data

Bayes' Rule

$$\boxed{p(h \mid d) = \frac{P(d \mid h)P(h)}{P(d)}}$$

Understanding Bayes' rule

d = data

h = hypothesis (model)

- rearranging

$p(h \mid d)P(d) = P(d \mid h)P(h)$

$P(d,h) = P(d,h)$

the same joint probability

on both sides

<u>Who is who in Bayes' rule</u>

$P(h)$:	prior belief (probability of hypothesis *h* before seeing any data)
$P(d \mid h)$:	likelihood (probability of the data if the hypothesis *h* is true)
$P(d) = \sum_h P(d \mid h)P(h)$:	data evidence (marginal probability of the data)
$P(h \mid d)$:	posterior (probability of hypothesis *h* after having seen the data *d*)

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20
- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Choosing Hypotheses

- *Maximum Likelihood* hypothesis:

$$h_{ML} = \arg\max_{h \in H} P(d \mid h)$$

- Generally we want the most probable hypothesis given training data.This is the *maximum a posteriori* hypothesis:
  - Useful observation: it does not depend on the denominator P(d)

$$h_{MAP} = \arg\max_{h \in H} P(h \mid d)$$

Bayesian Classifiers

- Consider each attribute and class label as random variables

- Given a record with attributes ($A_1, A_2,\ldots,A_n$)
  - Goal is to predict class C
  - Specifically, we want to find the value of C that maximizes $P(C| A_1, A_2,\ldots,A_n)$

- Can we estimate $P(C| A_1, A_2,\ldots,A_n)$ directly from data?

Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$\boxed{P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}}$$

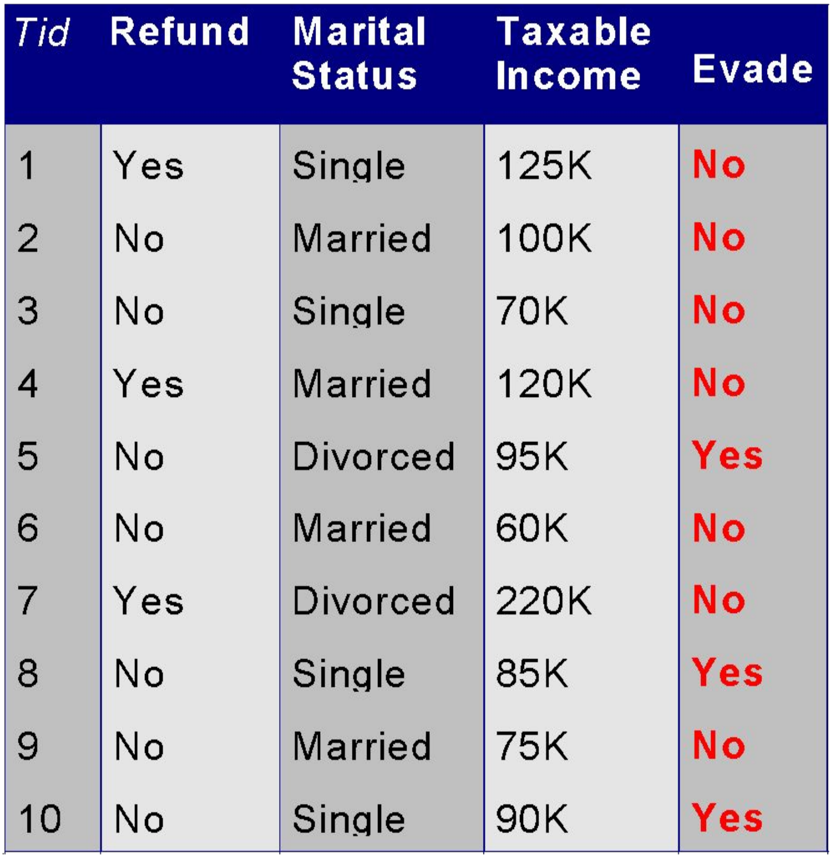
  - Choose value of C that maximizes
    $P(C \mid A_1, A_2, \ldots, A_n)$
  - Equivalent to choosing value of C that maximizes
    $P(A_1, A_2, \ldots, A_n \mid C)\, P(C)$
- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?

Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n | C) = P(A_1 | C_j) P(A_2 | C_j) \ldots P(A_n | C_j)$
  - Can estimate $P(A_i | C_j)$ for all A_i and C_j.
  - This is a simplifying assumption which may be violated in reality
- The Bayesian classifier that uses the Naïve Bayes assumption and computes the MAP hypothesis is called Naïve Bayes classifier

$$\boxed{c_{Naive\ Bayes} = \arg\max_c P(c) P(\mathbf{x} | c) = \arg\max_c P(c) \prod_i P(a_i | c)}$$

How to Estimate Probabilities from Data?



Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(C) = N_c/N$
  - e.g., $P(No) = 7/10$, $P(Yes) = 3/10$
- For discrete attributes:

$$P(A_i \mid C_k) = |A_{ik}| / N_{c_k}$$

  - where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
  - Examples:

    P(Status=Married|No) = 4/7
    P(Refund=Yes|Yes)=0

How to Estimate Probabilities from Data?

- For continuous attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
    - violates independence assumption
  - Two-way split: $(A < v)$ or $(A > v)$
    - choose only one of the two splits as new attribute
  - Probability density estimation:
    - Assume attribute follows a normal distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i, c_i) pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Naïve Bayesian Classifier: Training Dataset

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

New Data:
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

age	income	student	credit_rating	com
<=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 Bayesian Classifier: An Example

Given X (age=youth, income=medium, student=yes, credit=fair)

Maximize P(X|Ci)P(Ci), for i=1,2

First step: Compute P(C) The prior probability of each class can be computed based on the training tuples:

P(buys_computer=yes)=9/14=0.643

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

Naïve Bayesian Classifier: An Example

Given X (age=youth, income=medium, student=yes, credit=fair)

Maximize P(X|Ci)P(Ci), for i=1,2

Second step: compute P(X|Ci)

P(**X|buys_computer=yes**)= P(age=youth|buys_computer=yes)x
P(income=medium|buys_computer=yes) x
P(student=yes|buys_computer=yes)x
P(credit_rating=fair|buys_computer=yes)
= 0.044

P(age=youth|buys_computer=yes)=0.222

P(income=medium|buys_computer=yes)=0.444

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

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

Naïve Bayesian Classifier: An Example

Given X (age=youth, income=medium, student=yes, credit=fair)

Maximize P(X|Ci)P(Ci), for i=1,2

Second step: compute P(X|Ci)

P(**X|buys_computer=no**)= P(age=youth|buys_computer=no)x
P(income=medium|buys_computer=no) x
P(student=yes|buys_computer=no) x
P(credit_rating=fair|buys_computer=no)
= 0.019

P(age=youth|buys_computer=no)=3/5=0.666
P(income=medium|buys_computer=no)=2/5=0.400
P(student=yes|buys_computer=no)=1/5=0.200
P(credit_rating=fair|buys_computer=no)=2/5=0.400

Naïve Bayesian Classifier: An Example

Given X (age=youth, income=medium, student=yes, credit=fair)

Maximize P(X|Ci)P(Ci), for i=1,2

We have computed in the first and second steps:

P(buys_computer=yes)=9/14=0.643

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

P(X|buys_computer=yes)= 0.044

P(X|buys_computer=no)= 0.019

Third step: compute **P(X|Ci)P(Ci)** for each class

P(X|buys_computer=yes)P(buys_computer=yes)=0.044 x 0.643=0.028

P(X|buys_computer=no)P(buys_computer=no)=0.019 x 0.357=0.007

The naïve Bayesian Classifier predicts **X belongs to class (“buys_computer = yes”)**

Example

Training set :
(Öğrenme Kümesi)

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

k

Given a Test Record:

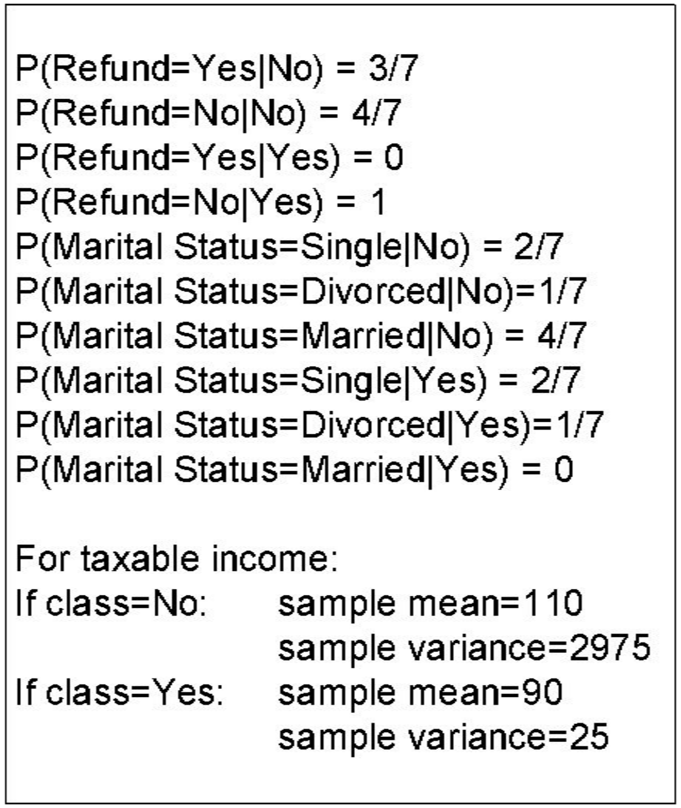
$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:



P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:

If class=No:	sample mean=110
	sample variance=2975
If class=Yes:	sample mean=90
	sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No) × P(Married| Class=No) × P(Income=120K| Class=No) = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes) × P(Married| Class=Yes) × P(Income=120K| Class=Yes) = $1 \times 0 \times 1.2 \times 10^{-9} = 0$

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)

=> Class = No

Avoiding the 0-Probability Problem

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m-estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of classes

p: prior probability

m: parameter

Naïve Bayes (Summary)

- ***Advantage***
  - Robust to isolated noise points
  - Handle missing values by ignoring the instance during probability estimate calculations
  - Robust to irrelevant attributes
- ***Disadvantage***
  - Assumption: class conditional independence, which may cause loss of accuracy
  - Independence assumption may not hold for some attribute. Practically, dependencies exist among variables
    - Use other techniques such as Bayesian Belief Networks (BBN)

Remember

- Bayes’ rule can be turned into a classifier
- Maximum A Posteriori (MAP) hypothesis estimation incorporates prior knowledge; Max Likelihood (ML) doesn’t
- Naive Bayes Classifier is a simple but effective Bayesian classifier for vector data (i.e. data with several attributes) that assumes that attributes are independent given the class.
- Bayesian classification is a generative approach to classification

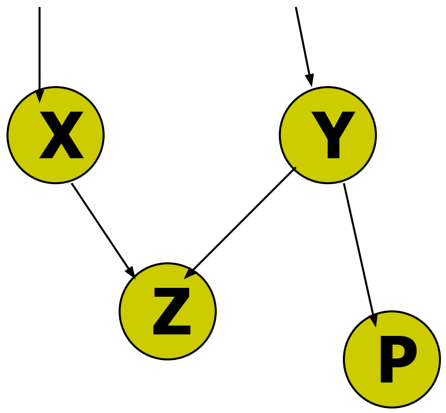
Classification Paradigms

- In fact, we can categorize three fundamental approaches to classification:
- Generative models: Model $p(x|C_k)$ and $P(C_k)$ separately and use the Bayes theorem to find the posterior probabilities $P(C_k|x)$
  - E.g. Naive Bayes, Gaussian Mixture Models, Hidden Markov Models,…
- Discriminative models:
  - Determine $P(C_k|x)$ directly and use in decision
  - E.g. Linear discriminant analysis, SVMs, NNs,…
- Find a discriminant function f that maps x onto a class label directly without calculating probabilities

Slide from B.Yanik

Bayesian Belief Networks

- Bayesian belief network allows a *subset* of the variables to be conditionally independent
- A graphical model of causal relationships *(neden sonuç ilişkilerini simgeleyen bir çizge tabanlı model)*
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution



❑ Nodes: random variables

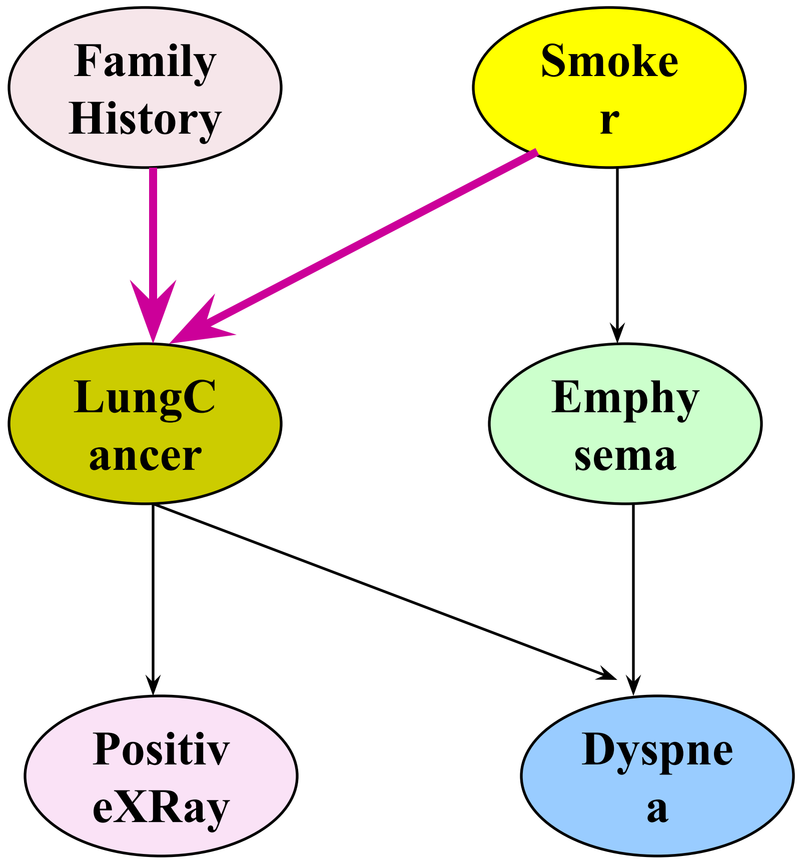
❑ Links: dependency

❑ X and Y are the parents of Z, and Y is the parent of P

❑ No dependency between Z and P

❑ Has no loops or cycles

Bayesian Belief Network: An Example



The **conditional probability table** (**CPT**) for variable LungCancer:

	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

CPT shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of **X**, from CPT:

Bayesian Belief Networks

$$P(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(Y_i))$$

Training Bayesian Networks

- Several scenarios:
  - Given both the network structure and all variables observable: *learn only the CPTs*
  - Network structure known, some hidden variables: *gradient descent* (greedy hill-climbing) method, analogous to neural network learning
  - Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
  - Unknown structure, all hidden variables: No good algorithms known for this purpose
- Ref. D. Heckerman: Bayesian networks for data mining

Lazy Learners

- The classification algorithms presented before are **eager learners**
  - Construct a model before receiving new tuples to classify
  - Learned models are ready and eager to classify previously unseen tuples
- **Lazy learners**
  - The learner waits till the last minute before doing any model construction
  - In order to classify a given test tuple
    - Store training tuples
    - Wait for test tuples
    - Perform generalization based on similarity between test and the stored training tuples

Lazy vs Eager

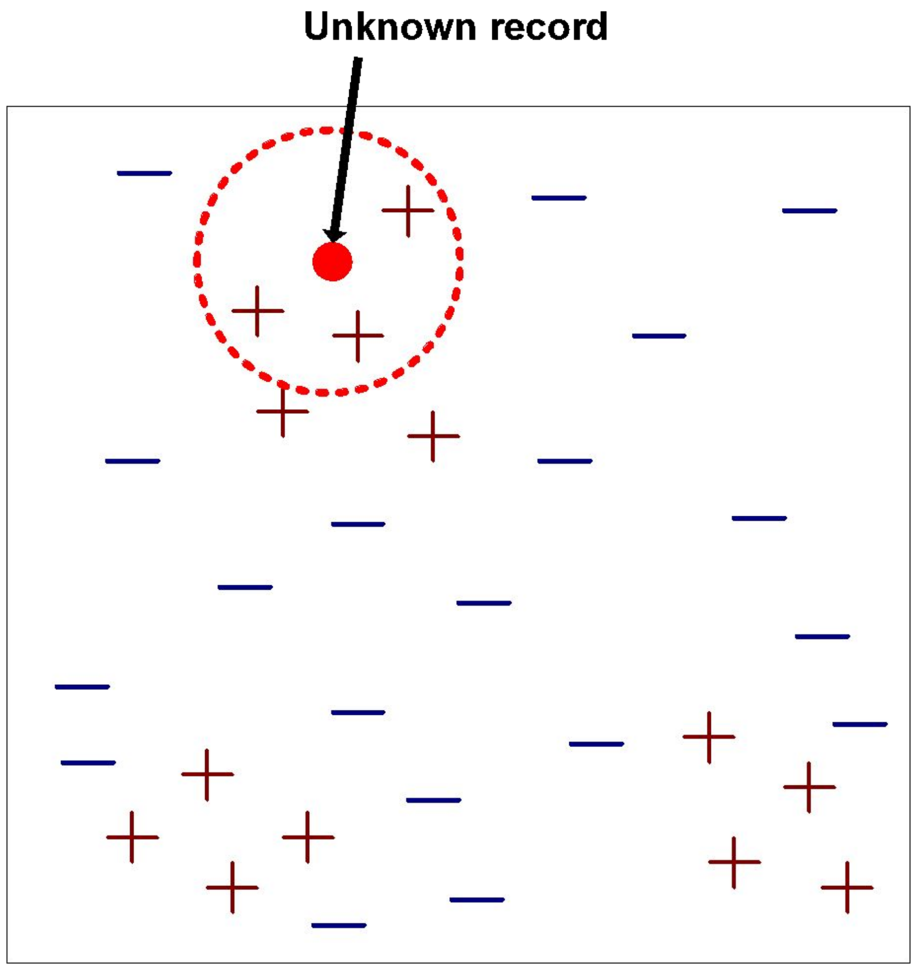
Eager Learners	**Lazy Learners**
• Do lot of work on training data	• Do less work on training data
• Do less work when test tuples are presented	• Do more work when test tuples are presented

Basic k-Nearest Neighbor Classification

- Given training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- Define a distance metric between points in input space ***D(x$_1$,x$_i$)***
  - E.g., Eucledian distance, Weighted Eucledian, Mahalanobis distance, TFIDF, etc.

- Training method:
  - Save the training examples
- At prediction time:
  - <u>Find</u> the ***k*** training examples *(x$_1$,y$_1$),…(x$_k$,y$_k$)* that are <u>**closest**</u> to the test example *x* given the distance *D(x$_1$,x$_i$)*
  - Predict the most frequent class among those *y$_i$*'s.

Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

K-Nearest Neighbor Model

- **Classification:**

$$\hat{y} = \text{ most common class in set } \{y_1, \ldots, y_K\}$$

- **Regression:**

$$\hat{y} = \frac{1}{K}\sum_{k=1}^{K} y_k$$

K-Nearest Neighbor Model: Weighted by Distance

- **Classification**:

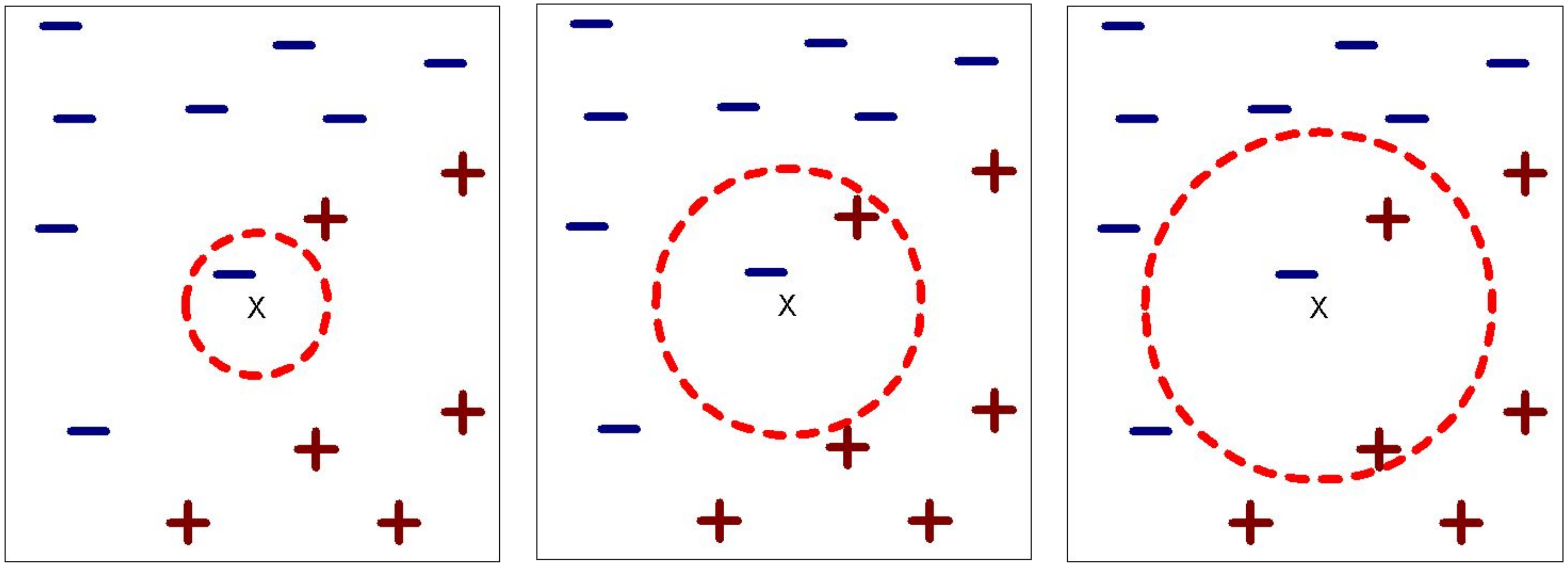
$\hat{y}$ = most common class in wieghted set

$$\left\{ D\left(\mathbf{x}, \mathbf{x}_1\right) y_1, \ldots, D\left(\mathbf{x}, \mathbf{x}_K\right) y_K \right\}$$

- **Regression**:

$$\hat{y} = \frac{\sum_{k=1}^{K} D(x, x_k) y_k}{\sum_{k=1}^{K} D(x, x_k)}$$

Definition of Nearest Neighbor

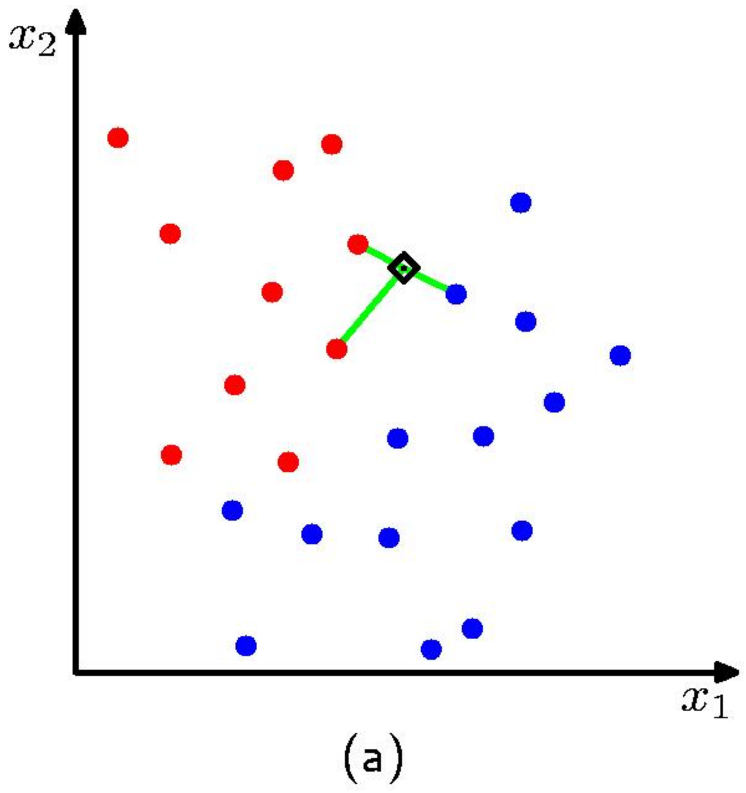


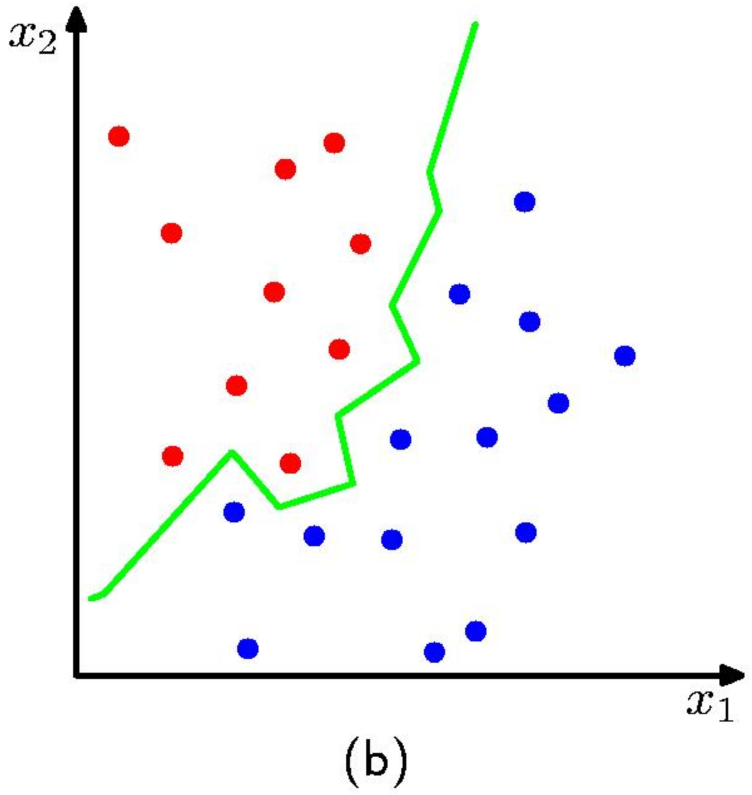
(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

The decision boundary implemented by 3NN

The boundary is always the perpendicular bisector of the line between two points (Voronoi tessellation)

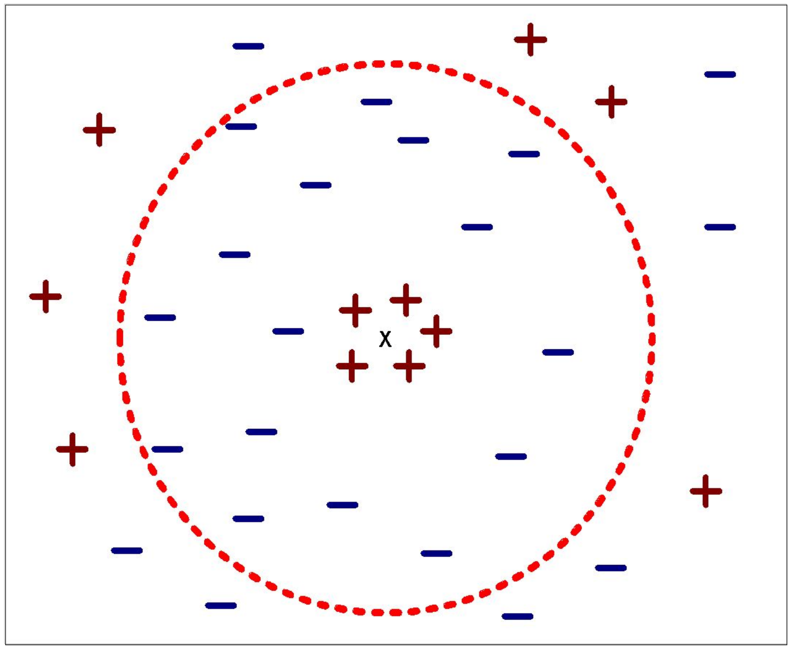




Slide by Hinton

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



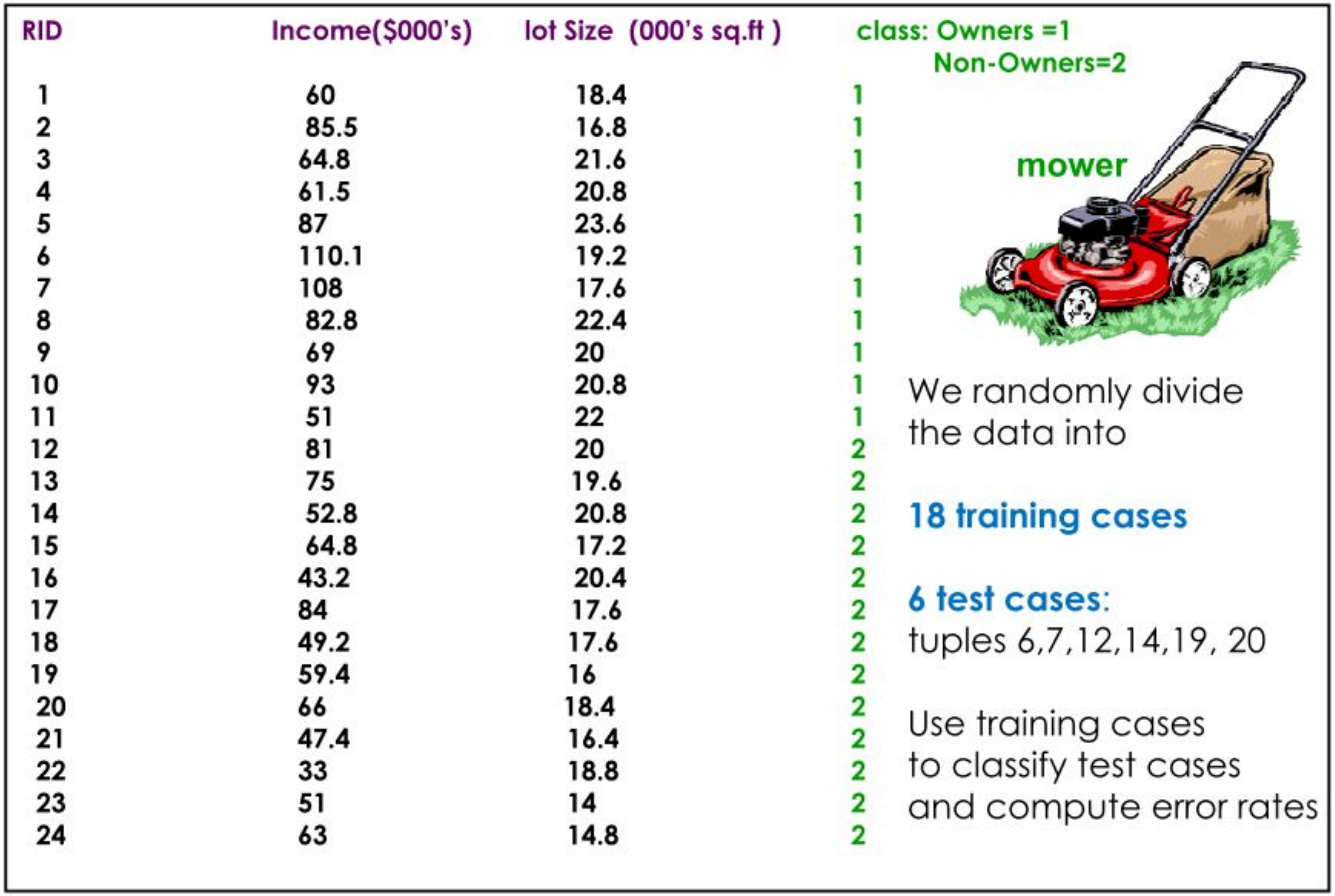
Determining the value of k

- In typical applications k is in units or tens rather than in hundreds or thousands
- Higher values of k provide smoothing that reduces the risk of overfitting due to noise in the training data
- Value of k can be chosen based on error rate measures
- We should also avoid over-smoothing by choosing k=n, where n is the total number of tuples in the training data set

Determining the value of k

- Given training examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- Use N fold cross validation
  - Search over K = (1,2,3,…,*Kmax*). Choose search size *Kmax* based on compute constraints
  - Calculated the average error for each K:
    - Calculate predicted class $\hat{y}_i$ for each training point $(\mathbf{x}_i, y_i),\ i = 1, \ldots, N$ (using all other points to build the model)
    - Average over all training examples
- Pick K to minimize the cross validation error

Example



RID	Income($000's)	lot Size (000's sq.ft)	class: Owners =1 Non-Owners=2
1	60	18.4	1
2	85.5	16.8	1
3	64.8	21.6	1
4	61.5	20.8	1
5	87	23.6	1
6	110.1	19.2	1
7	108	17.6	1
8	82.8	22.4	1
9	69	20	1
10	93	20.8	1
11	51	22	1
12	81	20	2
13	75	19.6	2
14	52.8	20.8	2
15	64.8	17.2	2
16	43.2	20.4	2
17	84	17.6	2
18	49.2	17.6	2
19	59.4	16	2
20	66	18.4	2
21	47.4	16.4	2
22	33	18.8	2
23	51	14	2
24	63	14.8	2

We randomly divide the data into

18 training cases

6 test cases:
tuples 6,7,12,14,19, 20

Use training cases to classify test cases and compute error rates

Example from J. Gamper

Choosing k

- If we choose **k=1** we will classify in a way that is very sensitive to the local characteristics of our data

- If we choose a **large value of k** we average over a large number of data points and average out the variability due to the noise associated with data points

- If we choose **k=18** we would simply predict the most frequent class in the data set in all cases
  - → Very stable but completely ignores the information in the independent variables

k	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**18**
Misclassification error %	33	33	33	33	33	17	17	50

  - → We would choose k=11 (or possibly 13) in this case

Slide from J. Gamper

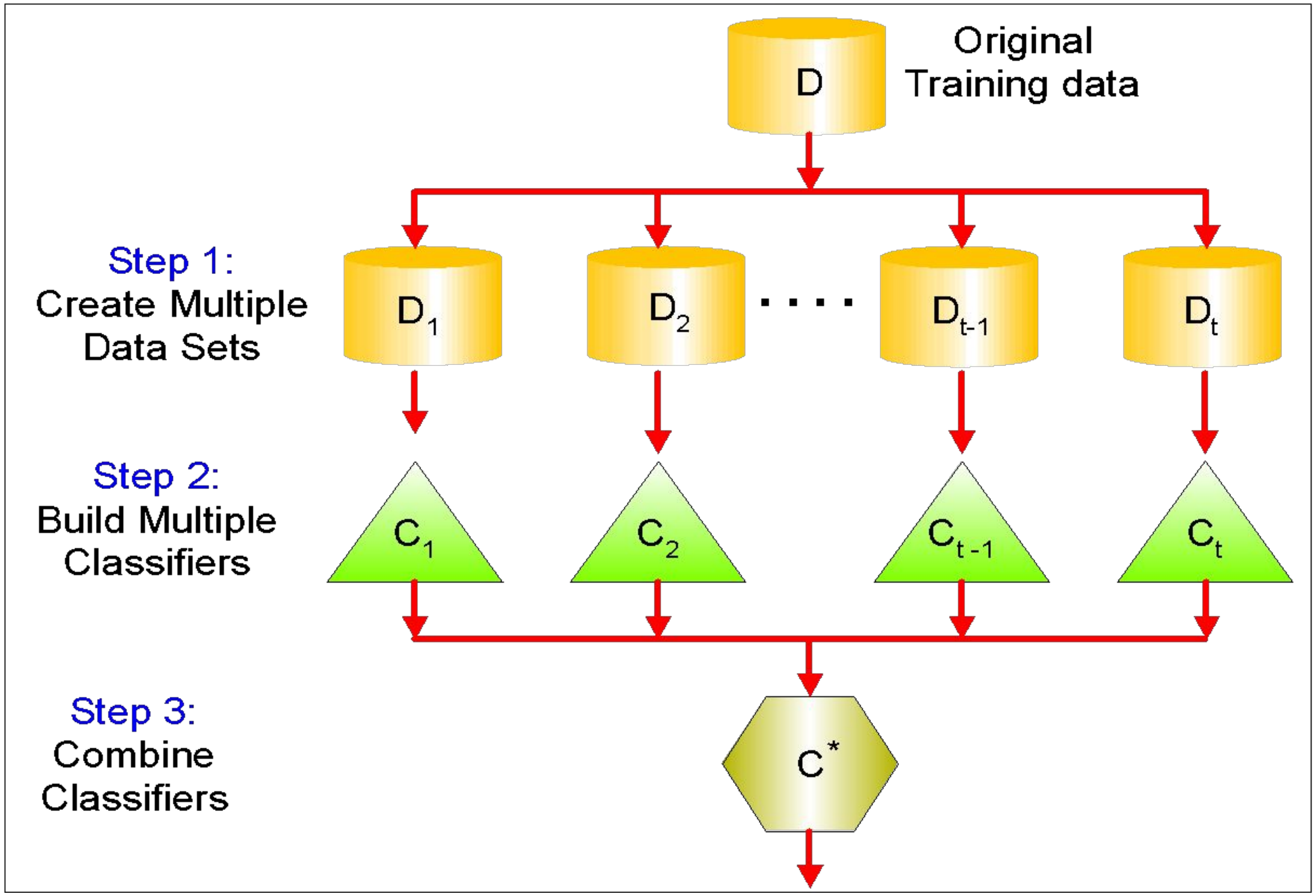
Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
- Adv: No training time
- Disadv:
  - Testing time can be long, classifying unknown records are relatively expensive
  - Curse of Dimensionality : Can be easily fooled in high dimensional spaces
    - Dimensionality reduction techniques are often used

Ensemble Methods

- One of the eager methods => builds model over the training set
- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

General Idea


D
Original Training data
Step 1: Create Multiple Data Sets
D_1
D_2
....
D_{t-1}
D_t
Step 2: Build Multiple Classifiers
C_1
C_2
C_{t-1}
C_t
Step 3: Combine Classifiers
C^*

Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=1}^{25} \binom{25}{i} \varepsilon^{i} (1-\varepsilon)^{25-i} = 0.06$$

Examples of Ensemble Methods

- How to generate an ensemble of classifiers?
  - Bagging
  - Boosting
  - Random Forests

Bagging: Bootstrap AGGregatING

- Bootstrap: data resampling
  - Generate multiple training sets
    - Resample the original training data
    - With replacement
  - Data sets have different "specious" patterns
- Sampling with replacement
  - Each sample has probability $(1 - 1/n)^n$ of being selected

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
  - Specious patterns will not correlate
- Underlying true pattern will be common to many
- Combine the classifiers: Label new test examples by a majority vote among classifiers

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round
- The final classifier is the weighted combination of the weak classifiers.

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds