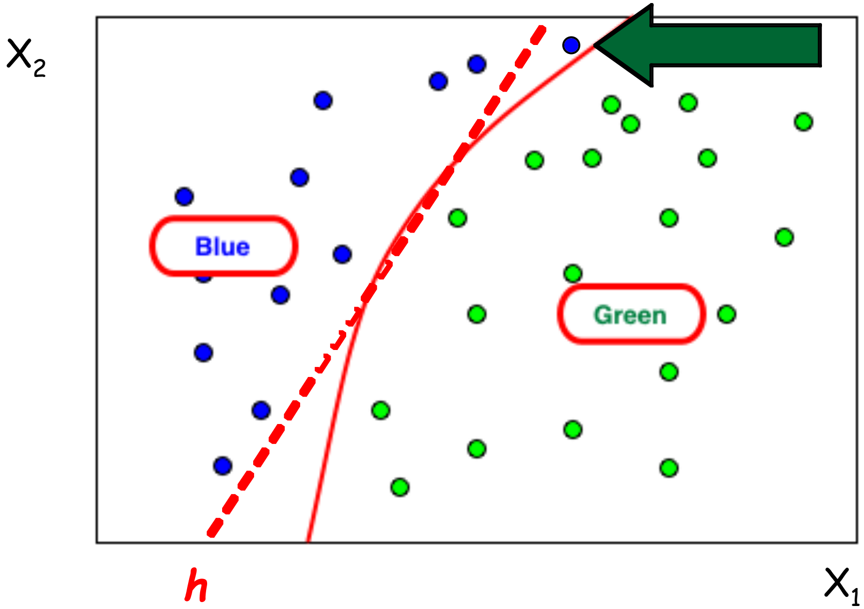
Data Mining & Machine Learning

CS57300
Purdue University

February 13, 2017

Classification

- In its simplest form, a classification model defines a decision boundary (h) and labels for each side of the boundary
- Input: **x**={x_1,x_2,...,x_n} is a set of attributes, function f assigns a label y to input **x**, where y is a discrete variable with a finite number of values



Parametric vs. non-parametric models

- Parametric
  - Particular functional form is assumed (e.g., Binomial)
  - **Number of parameters is fixed in advance**
  - Examples: Naive Bayes, perceptron
- Non-parametric
  - Few assumptions are made about the functional form
  - **Model structure is determined from data**
  - Examples: classification tree, nearest neighbor

Example model:
Naive Bayes classifiers

Classification as probability estimation

- Instead of learning a function f that assigns labels
- Learn a conditional probability distribution over the output of function f
- $P(f(x) \mid x) = P(f(x) = y \mid x_1,x_2,\dots, x_p)$
- Can use probabilities for the other two tasks
  - Classification
  - Ranking

Knowledge representation and model space

Bayes rule for probabilistic classifier

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

Bayes rule

$$= \frac{P(\mathbf{X}|Y)P(Y)}{[P(\mathbf{X}|Y=+)P(Y=+)] + [P(\mathbf{X}|Y=-)P(Y=-)]}$$

$$\propto P(\mathbf{X}|Y)P(Y)$$

Denominator: normalizing factor to make probabilities sum to 1 (can be computed from numerators)

Naive Bayes classifier

$$P(Y|\mathbf{X}) \propto P(\mathbf{X}|Y)P(Y)$$

Bayes rule

$$\propto \prod_{i=1}^{m} P(X_i|Y)P(Y)$$

Naive assumption

Assumption: Attributes are *conditionally independent* given the class

Naive Bayes classifier

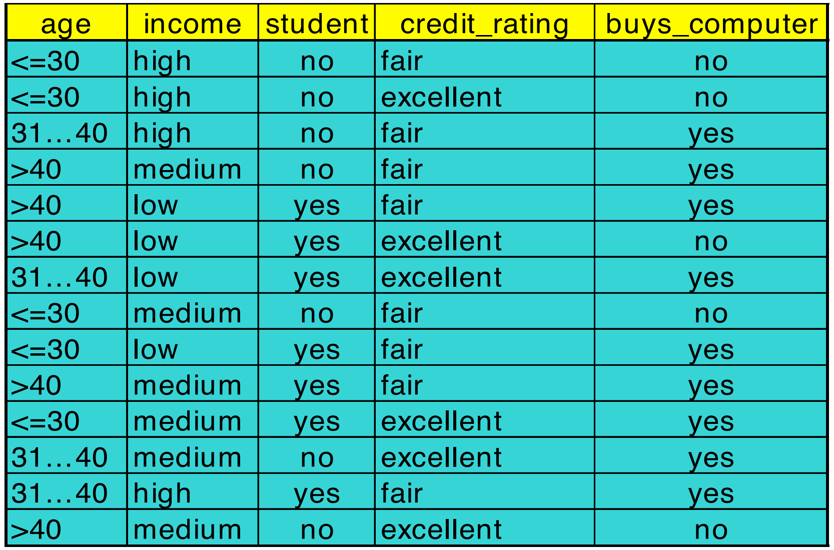
$$
\begin{aligned}
p(y|\mathbf{x}) &= \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \\
&= \frac{\prod_i p(x_i|y)\, p(y)}{\sum_j p(\mathbf{x}|y_j)p(y_j)}
\end{aligned}
$$

Model space:
parameters in conditional distributions $p(x_i|y)$
parameters in prior distribution $p(y)$

NBC learning

$$\begin{aligned} P(BC|A,I,S,CR) &= \frac{P(A,I,S,CR|BC)P(BC)}{P(A,I,S,CR)} \\ &= \frac{P(A|BC)P(I|BC)P(S|BC)P(CR|BC)P(BC)}{P(A,I,S,CR)} \\ &\propto \underline{P(A|BC)P(I|BC)P(S|BC)P(CR|BC)P(BC)} \end{aligned}$$

NBC parameters = Conditional Prob. Dist. (CPD), prior



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

CPDs: $P(A|BC)$
$P(I|BC)$
$P(S|BC)$
$P(CR|BC)$
Prior: $P(BC)$

Score function

Likelihood

- Let $D = \{x(1), ..., x(n)\}$
- Assume the data *D* are independently sampled from the same distribution:

$$p(X|\theta)$$

- The likelihood function represents the probability of the data as a function of the model parameters:

$$\begin{aligned} L(\theta|D) &= L(\theta|x(1), ..., x(n)) \\ &= p(x(1), ..., x(n)|\theta) \\ &= \prod_{i=1}^{n} p(x(i)|\theta) \end{aligned}$$

If instances are independent, likelihood is product of probs

Likelihood (cont’)

- Likelihood is not a probability distribution
  - Gives relative probability of data given a parameter
  - Numerical value of *L* is not relevant, only the ratio of two scores is relevant, e.g.,:

$$\frac{L(\theta_1|D)}{L(\theta_2|D)}$$

- **Likelihood function**: allows us to determine unknown parameters based on known outcomes
- **Probability distribution**: allows us to predict unknown outcomes based on known parameters

NBCs: Likelihood

- NBC likelihood uses the NBC probabilities for each data instance (i.e., probability of the class given the attributes)

$$L(\theta|D) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i;\theta)$$

General likelihood

$$\propto \prod_{i=1}^{n} p(\mathbf{x}_i|y_i;\theta)p(y_i|\theta)$$

Bayes rule

$$\propto \prod_{i=1}^{n}\prod_{j=1}^{p} p(x_{ij}|y_i;\theta)p(y_i|\theta)$$

Naive assumption

Search

Maximum likelihood estimation

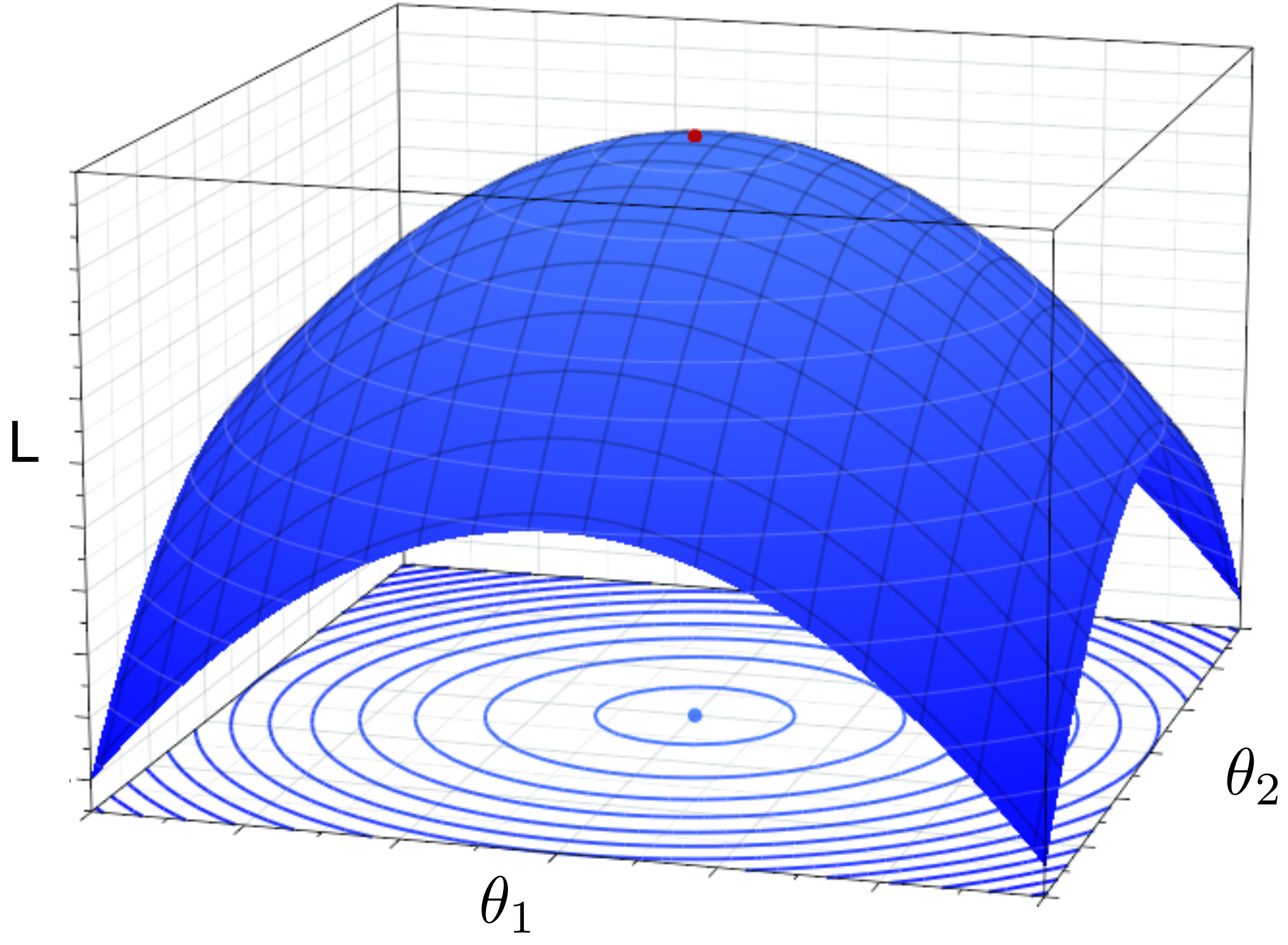
- Most widely used method of parameter estimation
- "Learn" the best parameters by finding the values of θ that maximizes likelihood:

$$\hat{\theta}_{MLE} = \arg\max_{\theta} L(\theta)$$

- Often easier to work with loglikelihood:

$$\begin{aligned} l(\theta|D) &= log\, L(\theta|D) \\ &= log \prod_{i=1}^{n} p(x(i)|\theta) \\ &= \sum_{i=1}^{n} logp(x(i)|\theta) \end{aligned}$$

Likelihood surface



If the likelihood surface is convex we can often determine the parameters that maximize the function analytically

MLE for multinomials

- Let $X \in \{1, ..., k\}$ be a discrete random variable with k values, where $P(X=j)=\theta_j$
- Then P(X) is a multinomial distribution:

$$P(X|\theta) = \prod_{j=1}^{k} \theta_j^{I(X=j)}$$

where I(X=j) is an indicator function

- The likelihood for a data set D=[x_1, ..., x_N] is:

$$P(D|\theta) = \prod_{n=1}^{N} \prod_{j=1}^{k} \theta_j^{I(x_n=j)} = \prod_{j} \theta_j^{N_j}$$

- The maximum likelihood estimates for each parameter are: $\boxed{\hat{\theta}_j = \frac{N_j}{N}}$
(using Lagrange multipliers)

In this case, MLE can be determined analytically by counting

Learning CPDs from examples

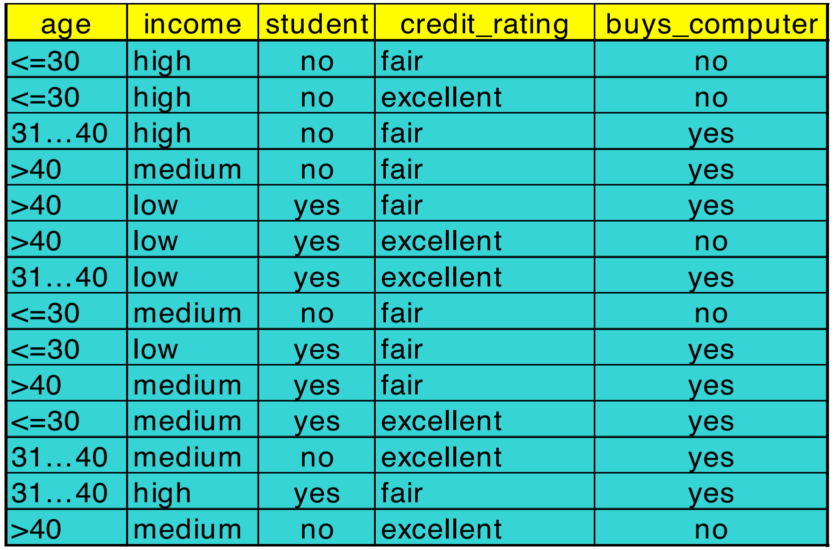
X_1

		Low	Medium	High
Y	Yes	10	13	17
	No	2	13	0

$$P[\ X_1 = \text{Low} \mid Y = \text{Yes}] = \frac{10}{(10+13+17)}$$

$$P[\ Y = \text{No}] = \frac{(2+13)}{(2+13+10+13+17)}$$

NBC learning



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

- Estimate prior P(BC) and conditional probability distributions P(A | BC), P(I | BC), P(S | BC), P(CR | BC) independently with maximum likelihood estimation

P(BC)

BC	θ
yes	$9/14$
no	$5/14$

P(A I BC)

BC	A	θ
yes	$<= 30$	$2/9$
	$31..40$	$4/9$
	> 40	$3/9$
no	$<= 30$	$3/5$
	$31..40$	$0/5$
	> 40	$2/5$

P(I I BC)

BC	I	θ
yes	high	$2/9$
	med	$4/9$
	low	$3/9$
no	high	$2/5$
	med	$2/5$
	low	$1/5$

P(S I BC)

BC	S	θ
yes	yes	$6/9$
	no	$3/9$
no	yes	$1/5$
	no	$4/5$

P(CR I BC)

BC	CR	θ
yes	exc	$3/9$
	fair	$6/9$
no	exc	$4/5$
	fair	$1/5$

NBC prediction

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
31..40	**high**	**no**	**excellent**	**?**

- What is the probability that a new person will buy a computer?

$$P(BC = yes|A = 31..40, I = high, S = no, CR = exc)$$
$$\propto P(A = 31..40|BC = yes)P(I = high|BC = yes)$$
$$P(S = no|BC = yes)P(CR = exc|BC = yes)P(BC = yes)$$

P(BC)

BC	θ
yes	$9/14$
no	$5/14$

P(A I BC)

BC	A	θ
yes	$<= 30$	$2/9$
	$31..40$	$4/9$
	> 40	$3/9$
no	$<= 30$	$3/5$
	$31..40$	$0/5$
	> 40	$2/5$

P(I I BC)

BC	I	θ
yes	high	$2/9$
	med	$4/9$
	low	$3/9$
no	high	$2/5$
	med	$2/5$
	low	$1/5$

P(S I BC)

BC	S	θ
yes	yes	$6/9$
	no	$3/9$
no	yes	$1/5$
	no	$4/5$

P(CR I BC)

BC	CR	θ
yes	exc	$3/9$
	fair	$6/9$
no	exc	$4/5$
	fair	$1/5$

Zero counts are a problem

- If an attribute value does not occur in training example, we assign **zero** probability to that value
- How does that affect the conditional probability P[f(x) | x] ?
- It equals 0!!!
- Why is this a problem?
- Adjust for zero counts by “smoothing” probability estimates

Smoothing: Laplace correction

X_1

		Low	Medium	High
Y	Yes	10	13	17
	No	2	13	0

Laplace correction
Numerator: ***add 1***
Denominator: ***add k***,
where k=number of possible values of X

$$P[\ X_1 = \text{High}\ |\ Y = \text{No}] = \frac{0 + 1}{(2+13+0) + 3}$$

Adds uniform prior

Is assuming independence a problem?

- What is the effect on probability estimates?
  - Over-counting evidence, leads to overly confident probability estimate

- What is the effect on classification?
  - Less clear…
  - For a given input x, suppose f(x) = True
  - Naïve Bayes will correctly classify if P[f(x) = True | x] > 0.5
    ...thus it may not matter if probabilities are overestimated

Naive Bayes classifier

- Simplifying (naive) assumption:
  attributes are conditionally independent given the class
- Strengths:
  - Easy to implement
  - Often performs well even when assumption is violated
  - Can be learned incrementally
- Weaknesses:
  - Class conditional assumption produces skewed probability estimates
  - Dependencies among variables cannot be modeled

NBC learning

- Model space
  - Parametric model with specific form
    (i.e., based on Bayes rule and assumption of conditional independence),
  - Models vary based on parameter estimates in CPDs
- Search algorithm
  - MLE optimization of parameters
    (convex optimization results in exact solution)
- Scoring function
  - Likelihood of data given NBC model form

Other predictive models

Nearest neighbor

- Instance-based method
- Learning
    - Stores training data and delays processing until a new instance must be classified
    - Assumes that all points are represented in p-dimensional space
- Prediction
    - **Nearest neighbors** are calculated using Euclidean distance
    - Classification is made based on class labels of neighbors

Nearest neighbor

Rule: find k closest (training) points to the test instance and assign the most frequently occurring class

1NN

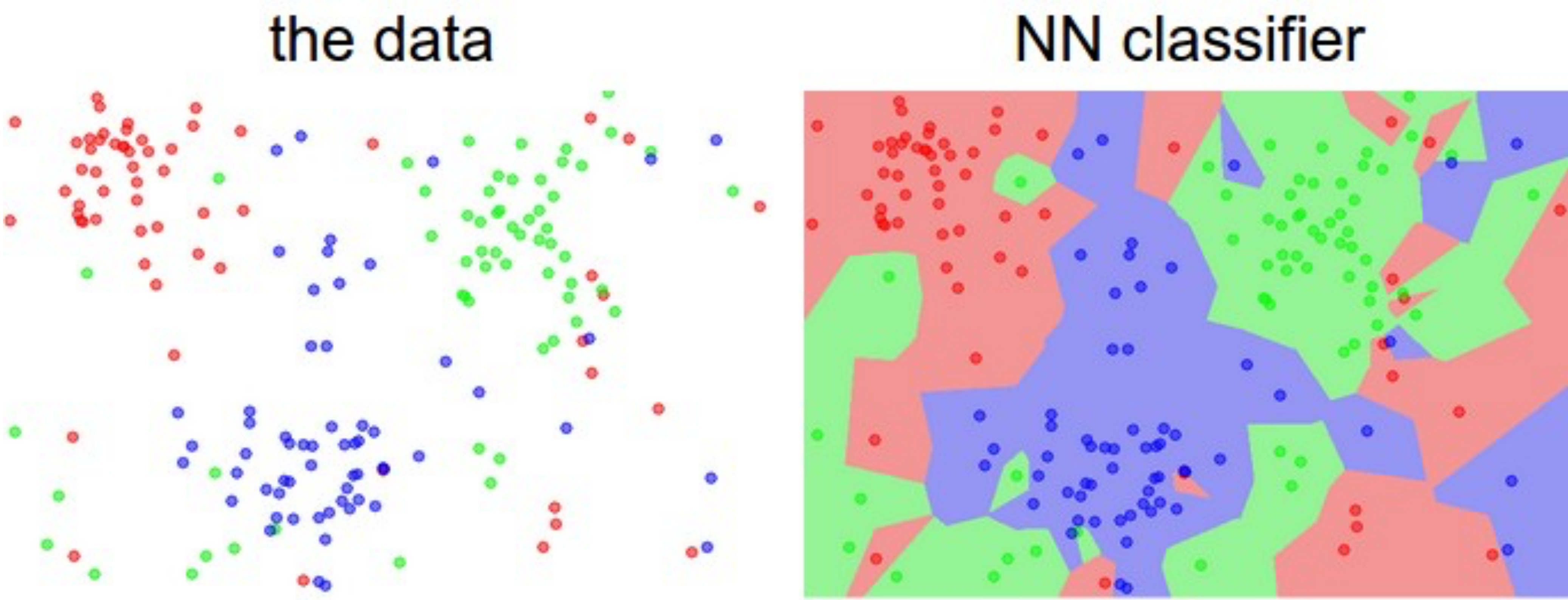
- Training set: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$
  where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{ip}]$ is a feature vector of p continuous attributes
  and y_i is a discrete class label

- **1NN algorithm**

  To predict a class label for new instance j:
  Find the training instance point $\mathbf{x}_i$ such that $d(\mathbf{x}_i, \mathbf{x}_j)$ is minimized
  Let $f(\mathbf{x}_j) = y_i$

- Key idea: Find instances that are "similar" to the new instance and use their class labels to make prediction for the new instance
  - 1NN generalizes to kNN when more neighbors are considered

kNN model: decision boundaries



Source: http://cs231n.github.io/classification/

kNN

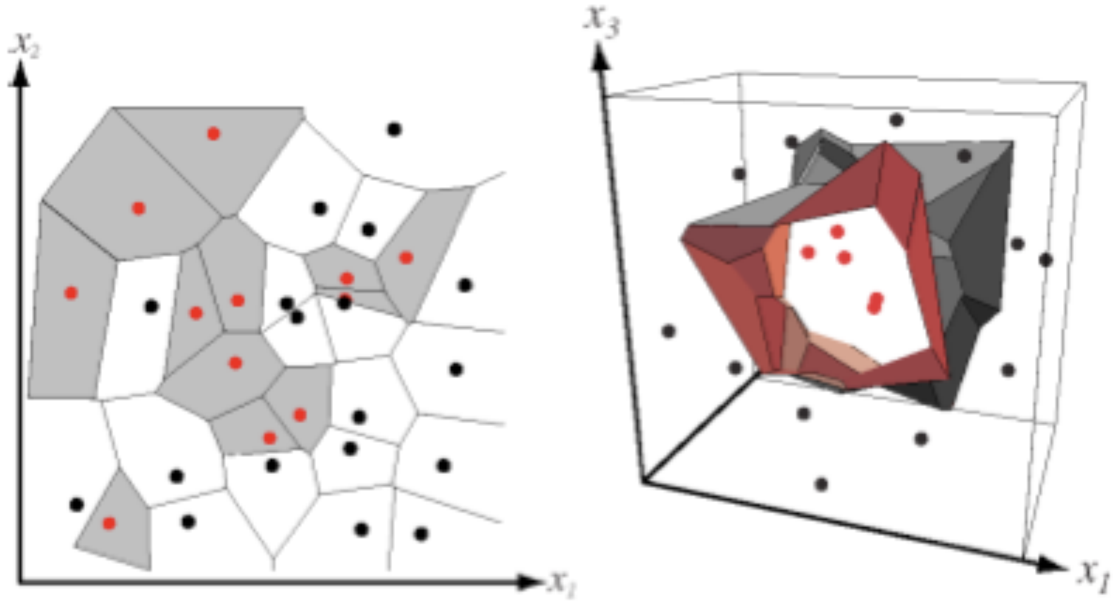
- **kNN algorithm**

  To predict a class label for new instance j:
  Find the k nearest neighbors of j, i.e., those that minimize d($\mathbf{x}_k$, $\mathbf{x}_j$)
  Let f($\mathbf{x}_j$)= g($\mathbf{y}_k$), e.g., majority label in $\mathbf{y}_k$

- *Algorithm choices*
  - How many neighbors to consider (i.e., choice of k)?
    ... Usually a small value is used, e.g. k<10
  - What distance measure d( ) to use?
    ... Euclidean L2 distance is often used
  - What function g( ) to combine the neighbors' labels into a prediction?
    ... Majority vote is often used

1NN decision boundary

- For each training example i, we can calculate its **Voronoi cell**, which corresponds to the space of points for which i is their nearest neighbor
- All points in such a Voronoi cell are labeled by the class of the training point, forming a Voronoi tessellation of the feature space



Source: http://www.cs.bilkent.edu.tr/~saksoy/courses/cs551-Spring2008/slides/cs551_nonbayesian1.pdf

Nearest neighbor

- Strengths:
  - Simple model, easy to implement
  - Very efficient learning: O(1)
- Weaknesses:
  - Inefficient inference: time and space O(n)
  - Curse of dimensionality:
    - As number of features increase, you need an exponential increase in the size of the data to ensure that you have nearby examples for any given data point

k-NN learning

- Parameters of the model:
    - k (number of neighbors)
    - any parameters of distance measure (e.g., weights on features)

- **Model space**
    - Possible tessellations of the feature space
- **Search algorithm**
    - Implicit search: choice of k, d, and g uniquely define a tessellation
- **Score function**
    - Majority vote is minimizing misclassification rate

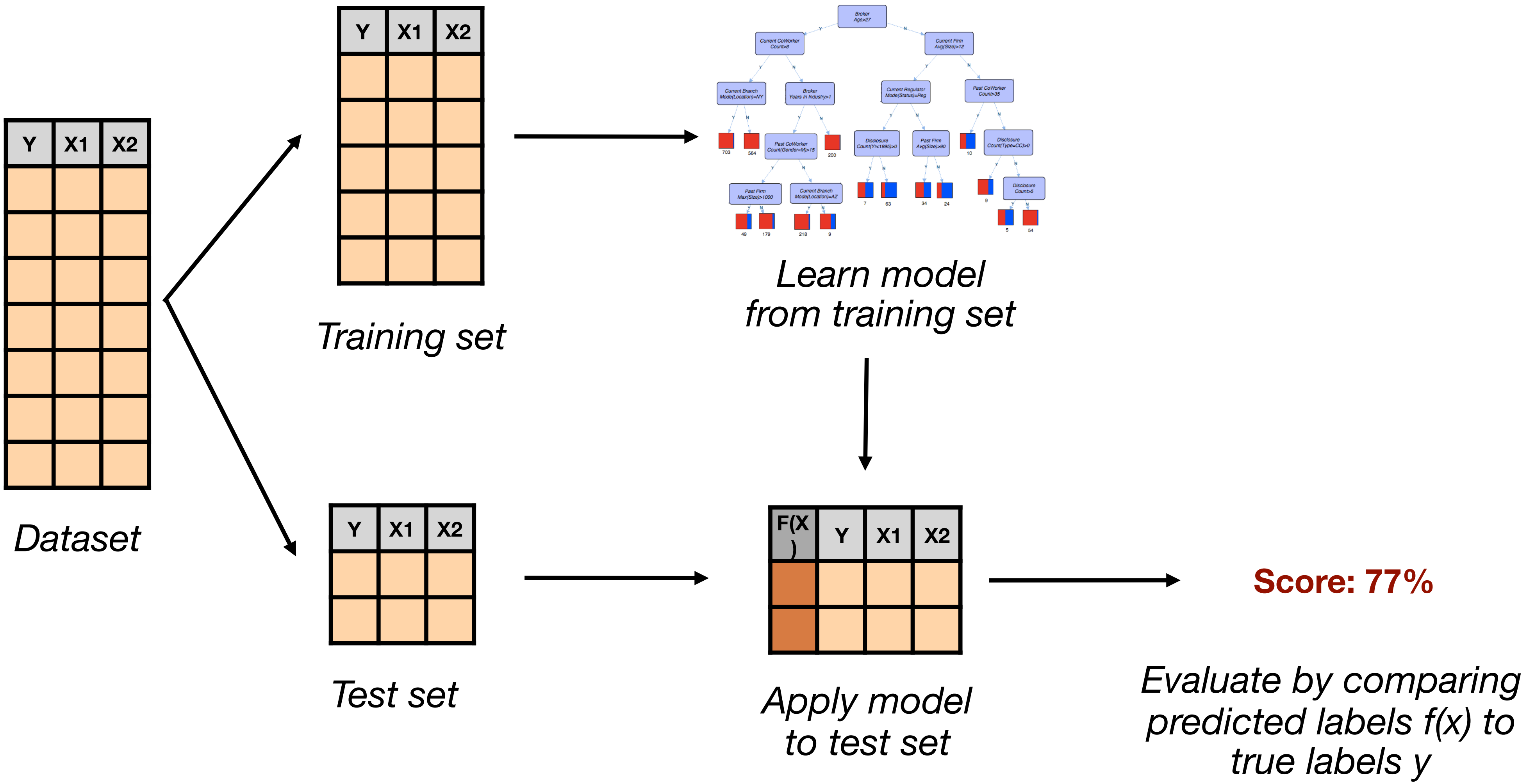
Putting it all together: Classification

Inputs and choices

- Input:
    - Dataset
    - Task
- Choices
    - Knowledge representation
    - Scoring function
    - Evaluation

- *Example:*
    - *Yelp data*
    - *Classification:*
      predict goodForGroups (Y)
      *using discrete attributes (**X**)*
    - *Naive Bayes*
    - *MLE w/smoothing*
    - *Zero-one loss,*
      square-loss

Illustration



Dataset

Training set

Learn model from training set

Test set

Apply model to test set

Score: 77%

Evaluate by comparing predicted labels f(x) to true labels y

Step 1

- Read in data
- Choose a data representation, e.g.,
    - In python the data can be represented as a list of lists (of strings): [['3', '?', 'alfa-romero', 'gas', 'std', 'two', 'convertible', 'rwd', 'front', '88.60', '168.80', '64.10', '48.80', '2548', 'dohc', 'four', '130', 'mpfi', '3.47', '2.68', '9.00', '111', '5000', '21', '27', '13495'], ['3', '?', 'alfa-romero', 'gas', 'std', 'two', 'convertible', 'rwd', 'front', '88.60', '168.80', '64.10', '48.80', '2548', 'dohc', 'four', '130', 'mpfi', '3.47', '2.68', '9.00', '111', '5000', '21', '27', '16500'], ...]
    - Or you can use separate structures to store the attributes and class labels (e.g., by assigning a unique id to each instance and using maps with the id as key)

Step 2

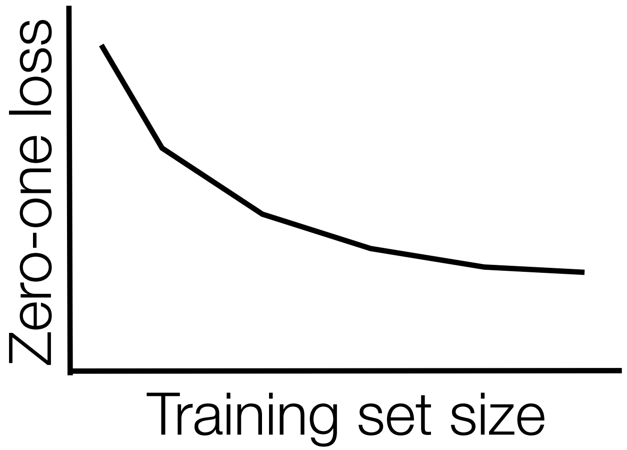
- Split into training and test sets
- There are many ways to split the data into training and test sets...
- The primary goal is to ensure that training and test examples are **disjoint**. This prevents the evaluation from being biased.
- Simple example:

```
partition1 = []
partition2 = []
partition3 = []
i = 0
for item in trainDS.getItems():
    if i<65: partition1.append(item)
    elif i<130: partition2.append(item)
    elif i<195: partition3.append(item)
    i += 1
partitions = [partition1,partition2,partition3]
```

Step 2b

- Consider repeated subsamples of the datato plot learning curves
- For each < $train_i$, $test_i$ >:
  - Learn model with $train_i$
  - Apply model to $test_i$

- You will average results over the 10 trials for each TSS to plot learning curve



Step 3

- From training data, create features (**X'**)
  - *Note: for your assignment you do not need to create features, just drop the continuous attributes, and use the discrete features as is*
- Example:
  - Let **X** be the set of 10 nominal attributes
  - For each attribute X_i with *k* possible values, construct k binary features to to add to **X'**, e.g.,
    - for X_i={red, green, blue}
      let F_1={red, ¬red}, F_2={green, ¬green}, F_3={blue, ¬blue}
      then **X'** = **X'** + {F_1, F_2, F_3}

Step 4

- Given training data, learn a model to predict Y given **X**

- Learn NBC model
  - Estimate class prior P(Y)
  - For each attribute estimate CPD P(X_i | Y)
  - Use smoothing for probability estimates

Step 5

- Given test data, apply model M to predict Y given **X**
- For each example, calculate:

$$P'(Y=1|\mathbf{X}) = \prod_i P(X_i = x_i|Y=1)P(Y=1)$$

$$P'(Y=0|\mathbf{X}) = \prod_i P(X_i = x_i|Y=0)P(Y=0)$$

$$P(Y=1|\mathbf{X}) = \frac{P'(Y=1|\mathbf{X})}{P'(Y=1|\mathbf{X}) + P'(Y=0|\mathbf{X})}$$

$$P(Y=0|\mathbf{X}) = 1 - P(Y=1|\mathbf{X})$$

- Predict class with max probability, i.e., if P(Y=1|X) > P(Y=0|X) then predict Y=1

Step 6

- Given a set of predictions for test data, evaluate the model by comparing the predicted values to the true values
  - Zero-one loss measures the mismatches between predicted and true class label:

$$Loss_{0/1}(T) = \frac{1}{n} \sum_{i \in n} \left\{ \begin{array}{ll} 0 & \text{if } y(i) = \hat{y}(i) \\ 1 & \text{otherwise} \end{array} \right\}$$

  - Squared loss measures the quality of the probability estimates. Let p_i refer to the probability that the NBC assigns to example *i*'s true class value, then:

$$Loss_{sq}(T) = \frac{1}{n} \sum_{i \in n} (1 - p_i)^2$$