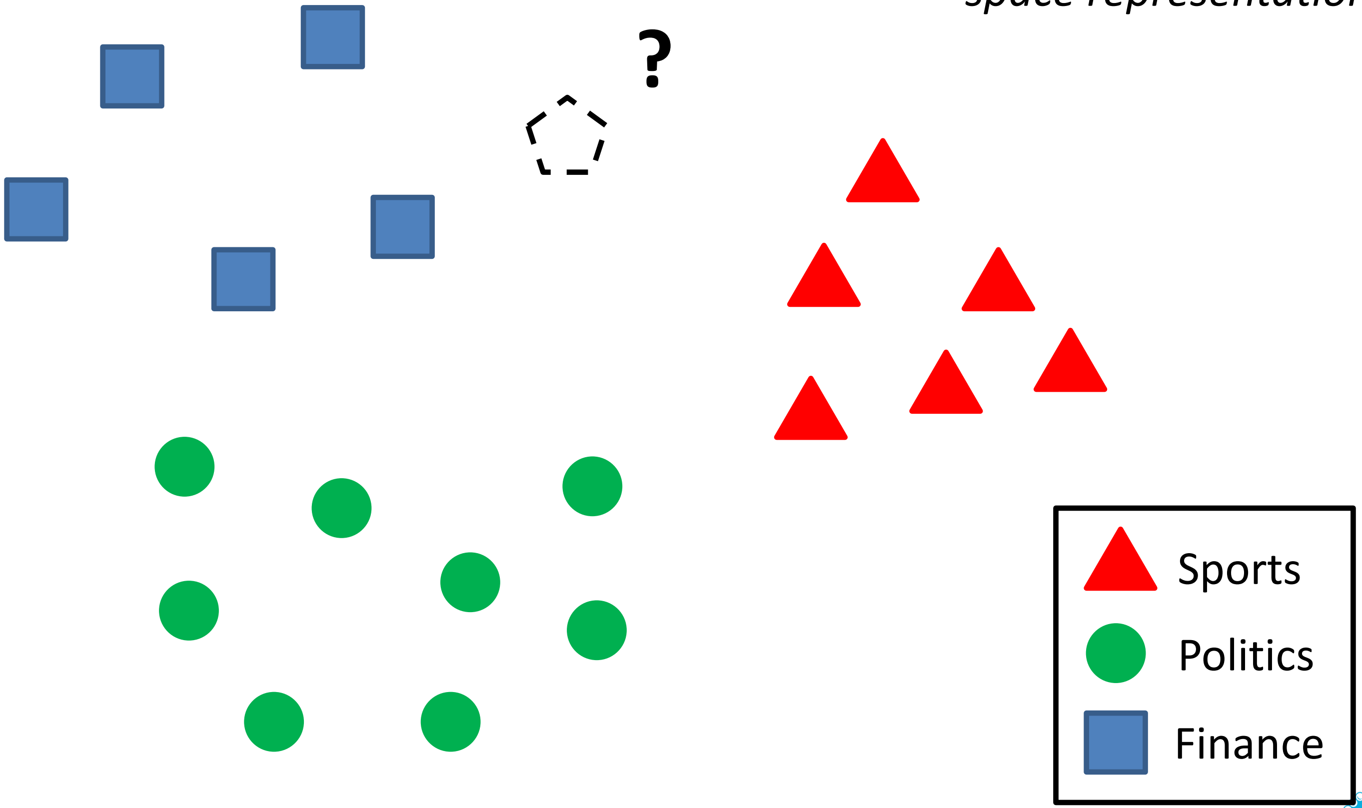
KNN and Naïve Bayes

School of Electrical and Computer Engineering
University of Tehran
Erfan Darzi
erfandarzi@ut.ac.ir

Today’s lecture

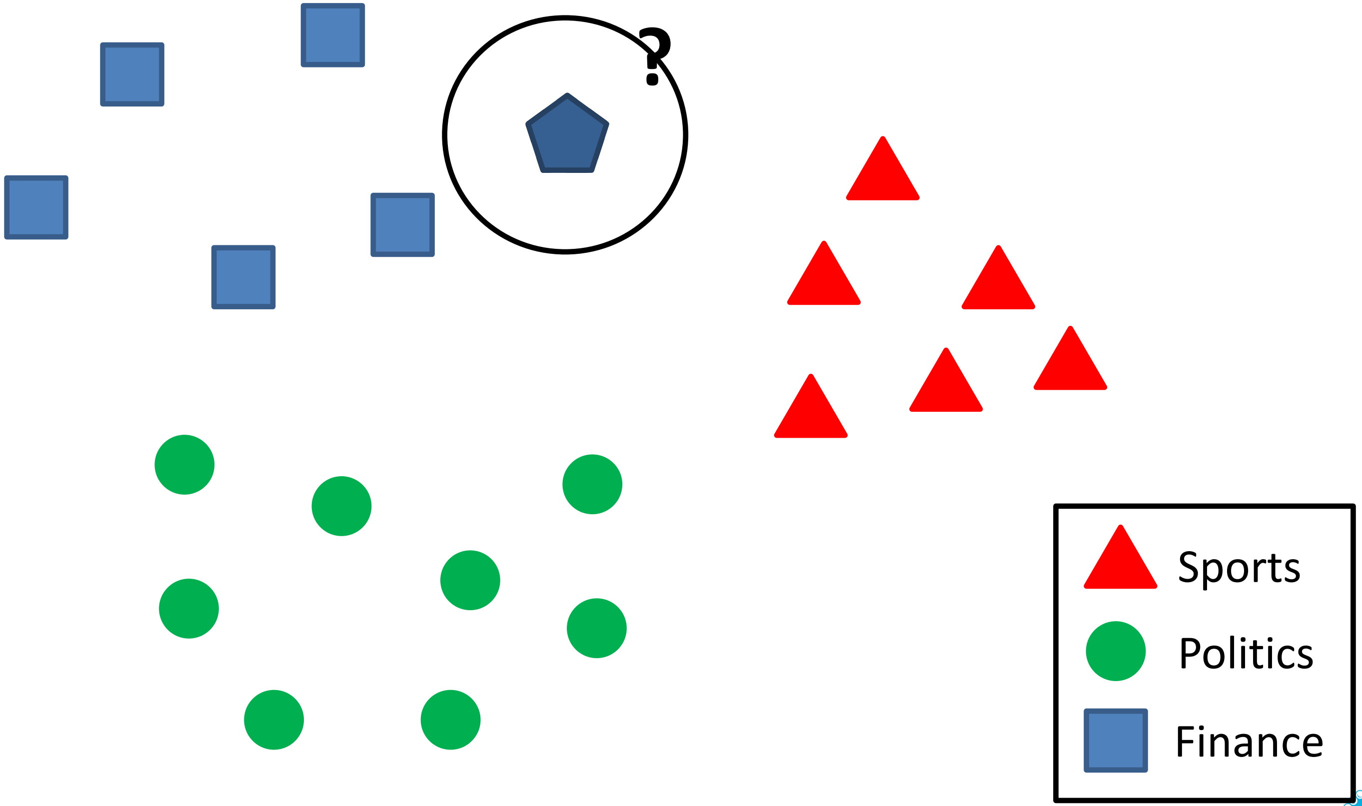
- Instance-based classifiers
  - k nearest neighbors
  - Non-parametric learning algorithm
- Model-based classifiers
  - Naïve Bayes classifier
    - A generative model
  - Parametric learning algorithm

How to classify this document?



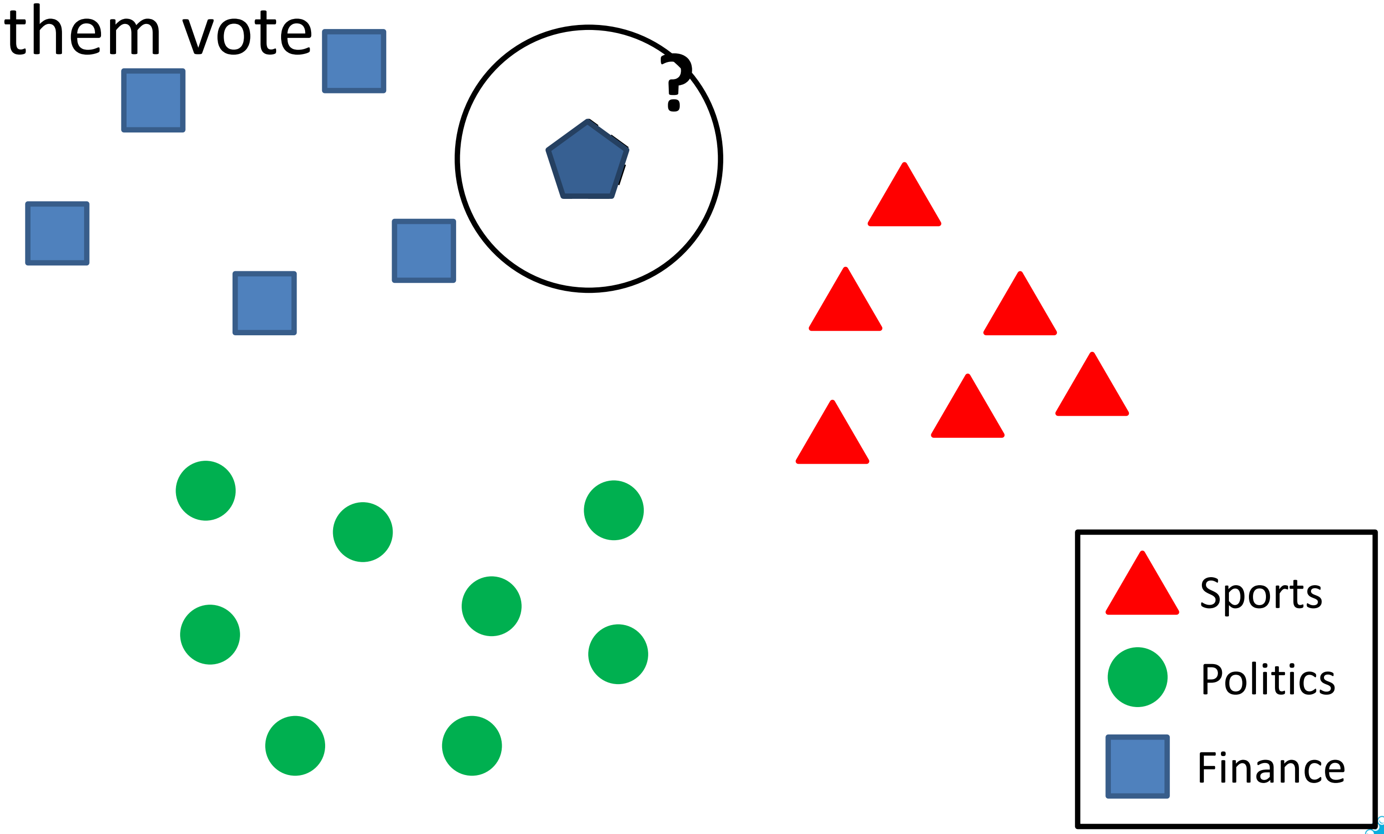
Let’s check the nearest neighbor

Are you confident about this?



Let’s check more nearest neighbors

- Ask k nearest neighbors
  - Let them vote



Probabilistic interpretation of kNN

- Approximate Bayes decision rule in a subset of data around the testing point
- Let V be the volume of the m dimensional ball around x containing the k nearest neighbors for x, we have

Nearest neighbors from class 1

$$p(x)V = \frac{k}{N} \Rightarrow p(x) = \frac{k}{NV} \qquad p(x|y=1) = \frac{k_1}{N_1 V} \qquad p(y=1) = \frac{N_1}{N}$$

Total number of instances

Total number of instances in class 1

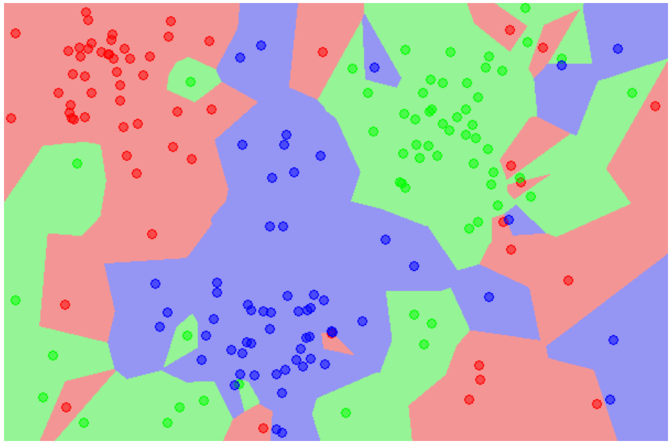
With Bayes rule:

$$p(y=1|x) = \frac{\frac{N_1}{N} \times \frac{k_1}{N_1 V}}{\frac{k}{NV}} = \frac{k_1}{k}$$

Counting the nearest neighbors from class 1

kNN is close to optimal

- Asymptotically, the error rate of 1-nearest-neighbor classification is less than twice of the Bayes error rate
- Decision boundary
  - 1NN - Voronoi tessellation



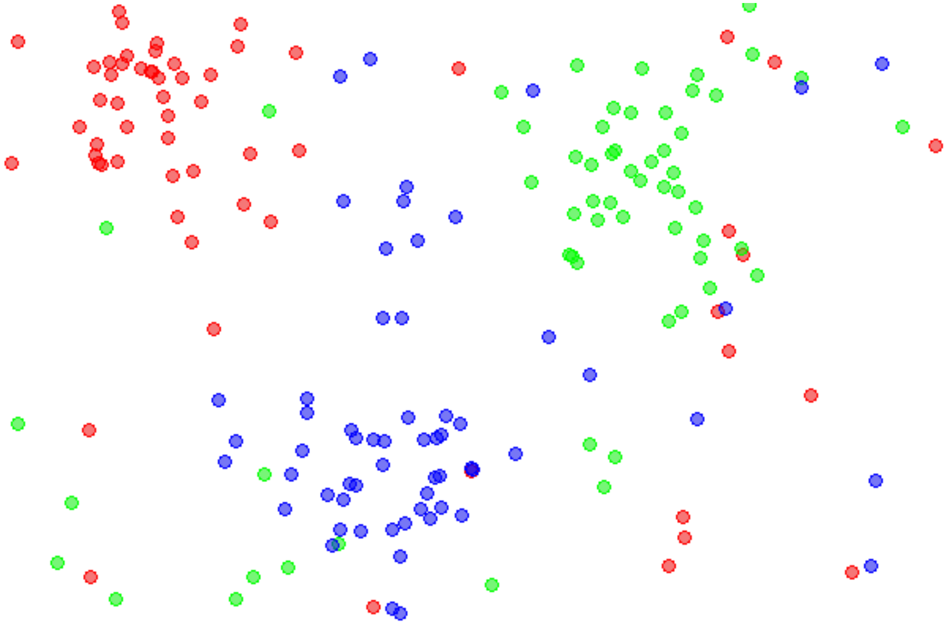
A non-parametric estimation of posterior distribution

Components in kNN

- A distance metric
  - Euclidean distance/cosine similarity
- How many nearby neighbors to look at
  - k
- Instance look up
  - Efficiently search nearby points

Effect of k

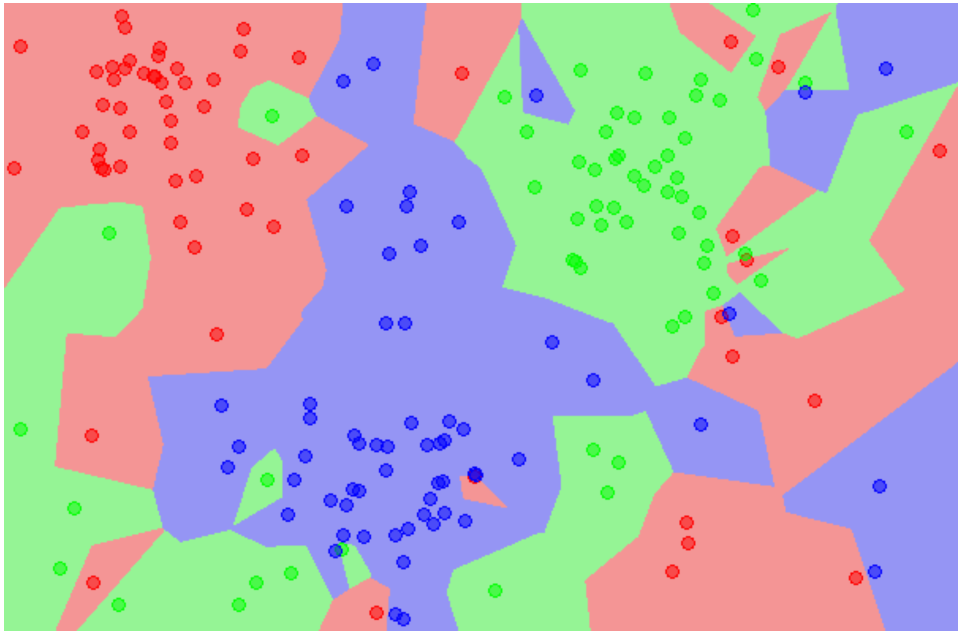
- Choice of k influences the “smoothness” of the resulting classifier



Effect of k

- Choice of k influences the “smoothness” of the resulting classifier

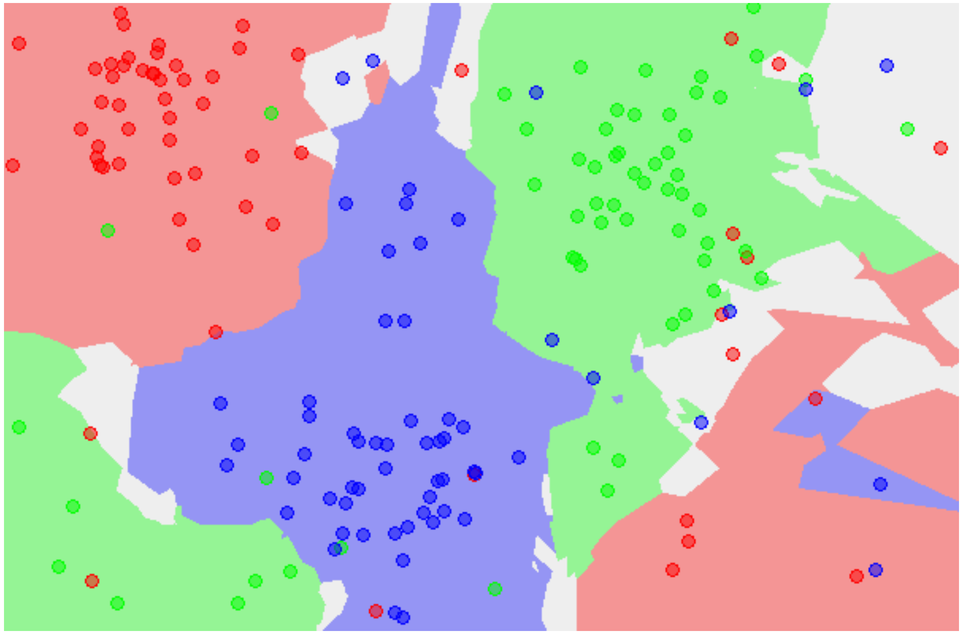
k=1



Effect of k

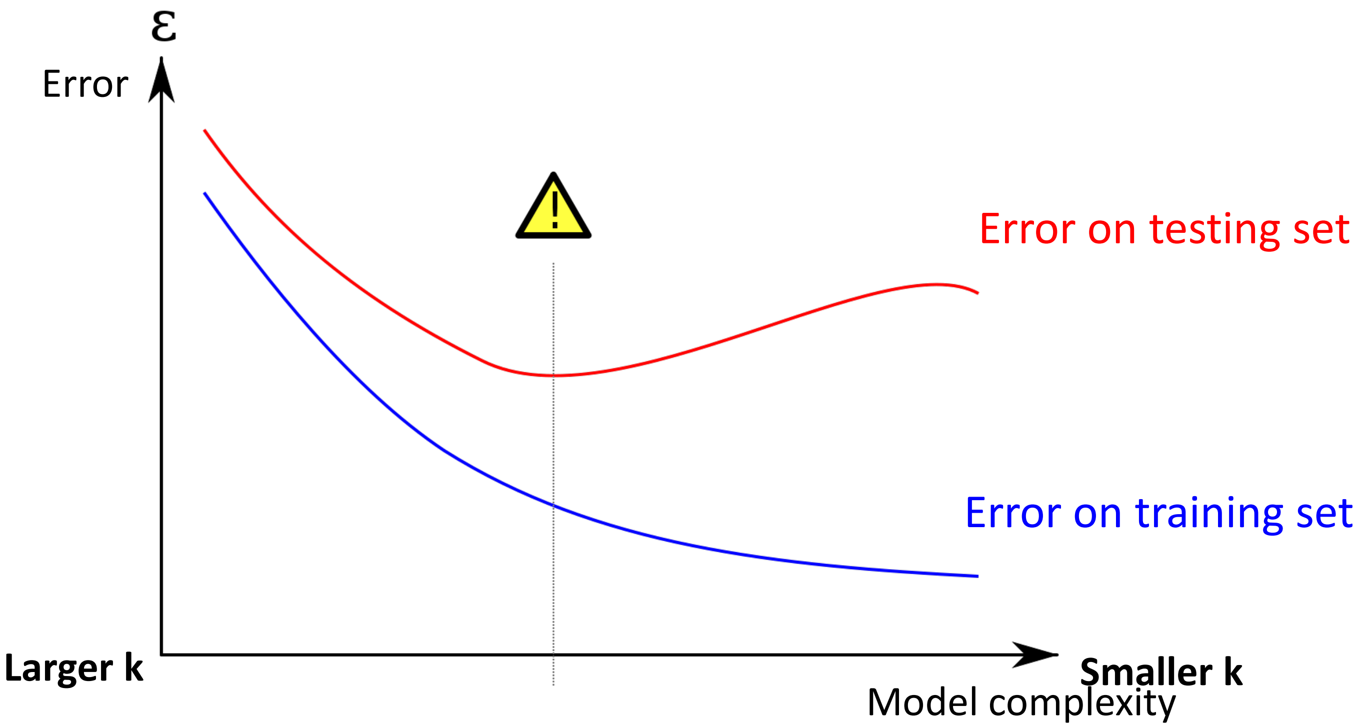
- Choice of k influences the “smoothness” of the resulting classifier

k=5



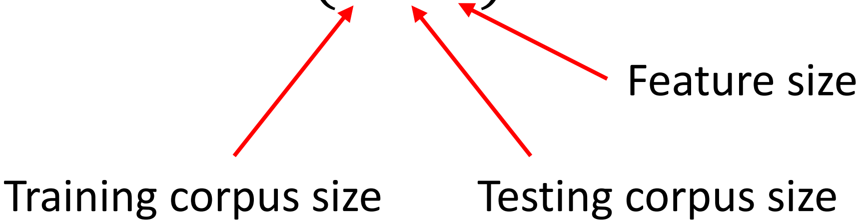
Effect of k

- Large k -> smooth shape for decision boundary
- Small k -> complicated decision boundary



Efficient instance look-up

- Recall MP1
  - In Yelp_small data set, there are 629K reviews for training and 174K reviews for testing
  - Assume we have a vocabulary of 15K
  - Complexity of kNN
    - $O(NMV)$



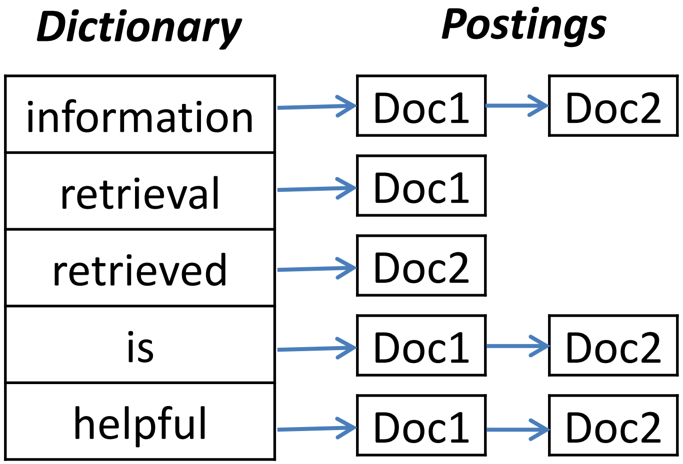
Feature size

Training corpus size

Testing corpus size

Efficient instance look-up

- Exact solutions
  - Build inverted index for text documents
    - Special mapping: word -> document list
    - Speed-up is limited when average document length is large



Dictionary ***Postings***

information → Doc1 → Doc2

retrieval → Doc1

retrieved → Doc2

is → Doc1 → Doc2

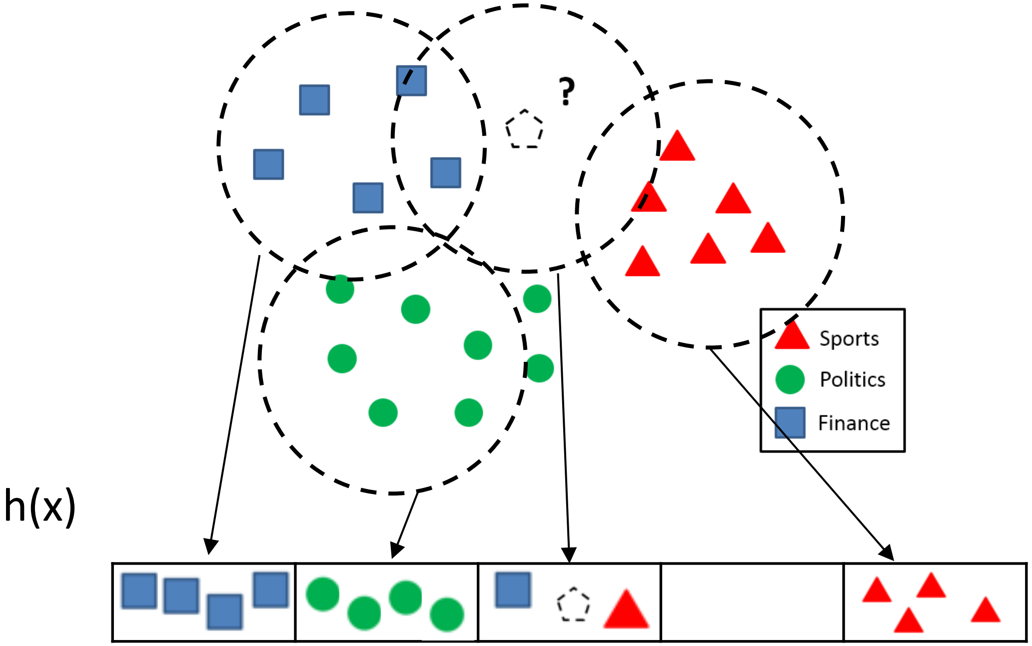
helpful → Doc1 → Doc2

Efficient instance look-up

- Exact solutions
  - Build inverted index for text documents
    - Special mapping: word -> document list
    - Speed-up is limited when average document length is large
  - Parallelize the computation
    - Map-Reduce
      - Map training/testing data onto different reducers
      - Merge the nearest k neighbors from the reducers

Efficient instance look-up

- Approximate solution
  - Locality sensitive hashing
    - Similar documents -> (likely) same hash values



Efficient instance look-up

- Approximate solution
  - Locality sensitive hashing
    - Similar documents -> (likely) same hash values
    - Construct the hash function such that similar items map to the same “buckets” with a high probability
      - Learning-based: learn the hash function with annotated examples, e.g., must-link, cannot-link
      - Random projection

Recap: probabilistic interpretation of kNN

- Approximate Bayes decision rule in a subset of data around the testing point
- Let V be the volume of the m dimensional ball around x containing the k nearest neighbors for x, we have

Nearest neighbors from class 1

$$p(x)V = \frac{k}{N} \quad => \quad p(x) = \frac{k}{NV} \qquad p(x|y=1) = \frac{k_1}{N_1 V} \qquad p(y=1) = \frac{N_1}{N}$$

Total number of instances

Total number of instances in class 1

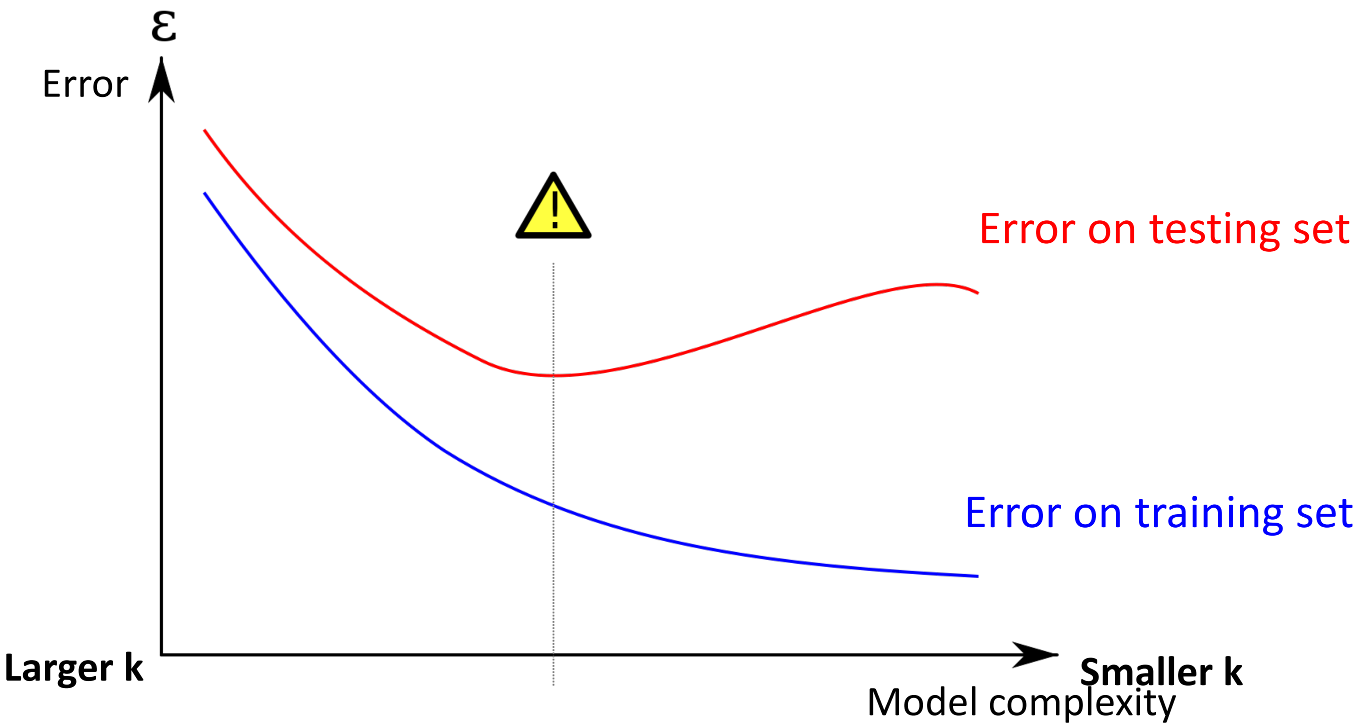
With Bayes rule:

$$p(y=1|x) = \frac{\frac{N_1}{N} \times \frac{k_1}{N_1 V}}{\frac{k}{NV}} = \frac{k_1}{k}$$

Counting the nearest neighbors from class 1

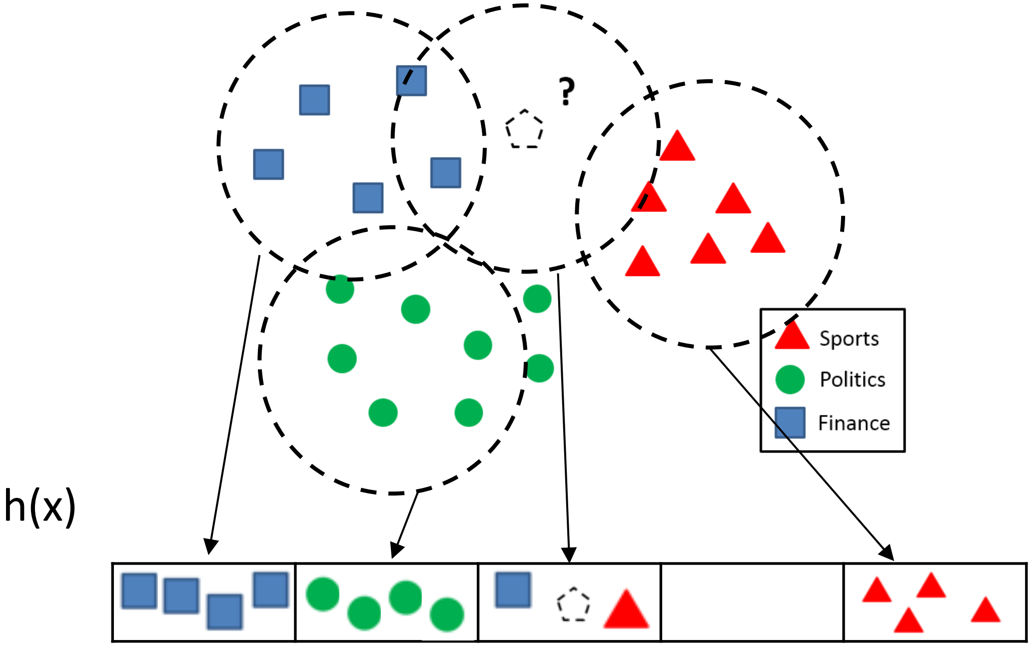
Recap: effect of k

- Large k -> smooth shape for decision boundary
- Small k -> complicated decision boundary



Recap: efficient instance look-up

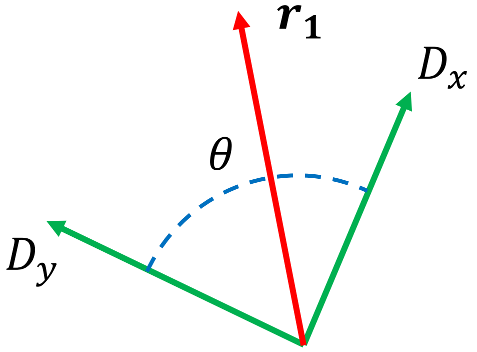
- Approximate solution
  - Locality sensitive hashing
    - Similar documents -> (likely) same hash values

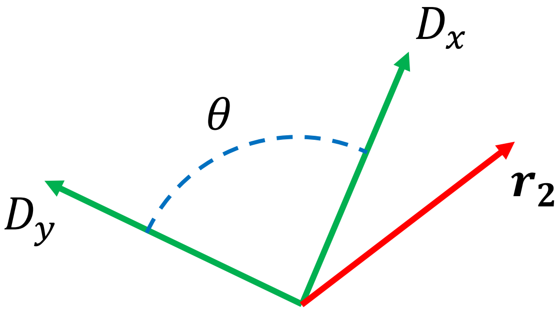


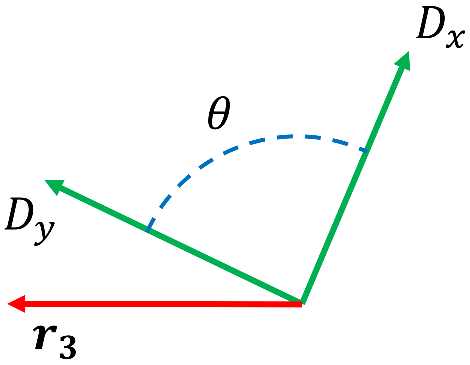
Random projection

- Approximate the cosine similarity between vectors
  - $h^r(x) = sgn(x \cdot r)$, r is a **random** unit vector
  - Each r defines one hash function, i.e., one bit in the hash value

	$\boldsymbol{r_1}$	$\boldsymbol{r_2}$	$\boldsymbol{r_3}$
D_x	1	1	0
D_y	1	0	1



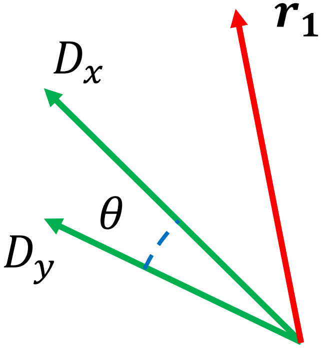


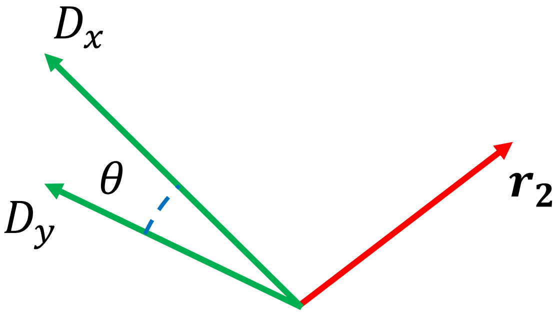


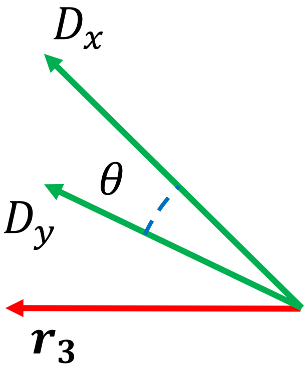
Random projection

- Approximate the cosine similarity between vectors
  - $h^r(x) = sgn(x \cdot r)$, r is a random unit vector
  - Each r defines one hash function, i.e., one bit in the hash value

	$\boldsymbol{r_1}$	$\boldsymbol{r_2}$	$\boldsymbol{r_3}$
D_x	1	0	1
D_y	1	0	1





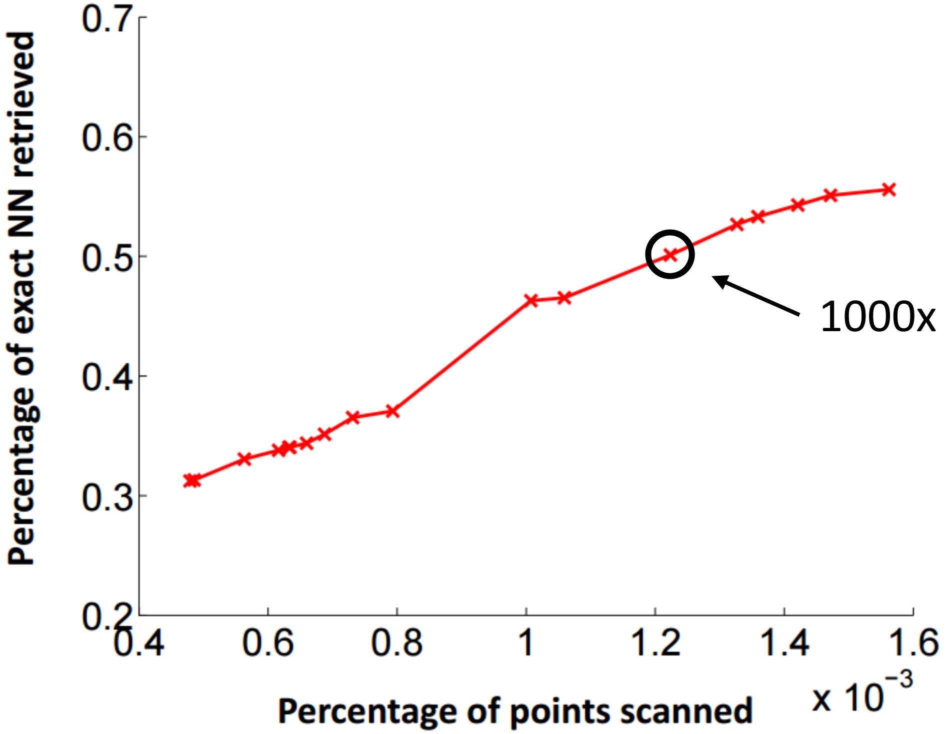


Random projection

- Approximate the cosine similarity between vectors
  - $h^r(x) = sgn(x \cdot r)$, r is a random unit vector
  - Each r defines one hash function, i.e., one bit in the hash value
  - Provable approximation error
    - $P\big(h(x) = h(y)\big) = 1 - \frac{\theta(x,y)}{\pi}$

Efficient instance look-up

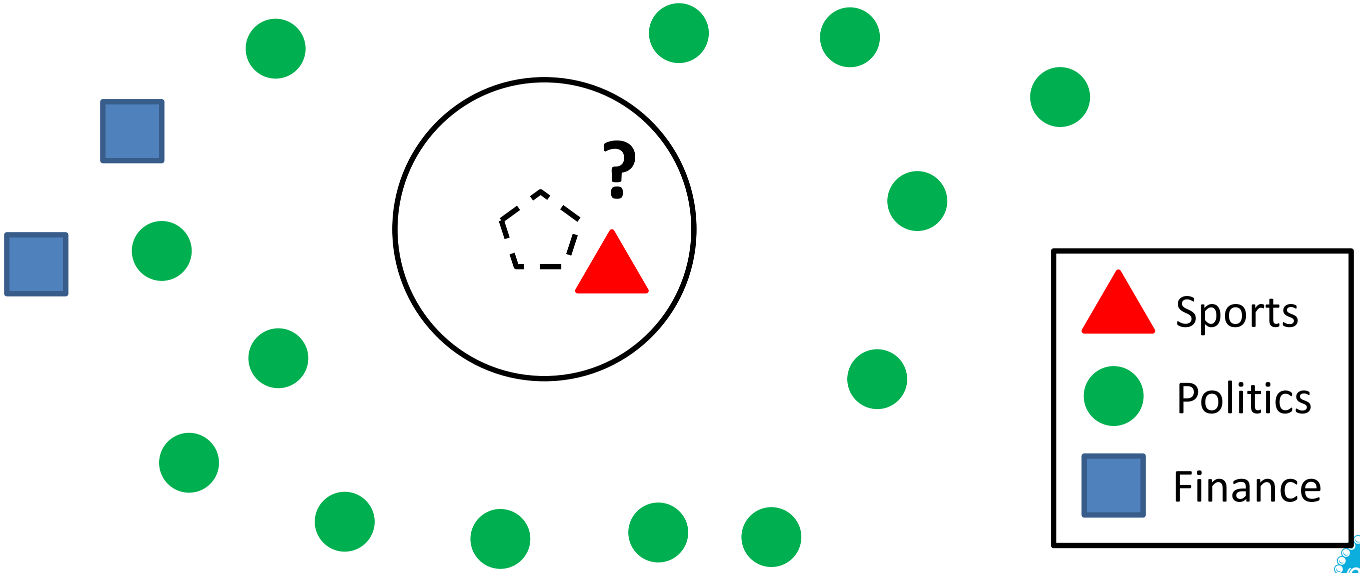
- Effectiveness of random projection
  - 1.2M images + 1000 dimensions



1000x speed-up

Weight the nearby instances

- When the data distribution is highly skewed, frequent classes might dominate majority vote
  - They occur more often in the k nearest neighbors just because they have large volume



Weight the nearby instances

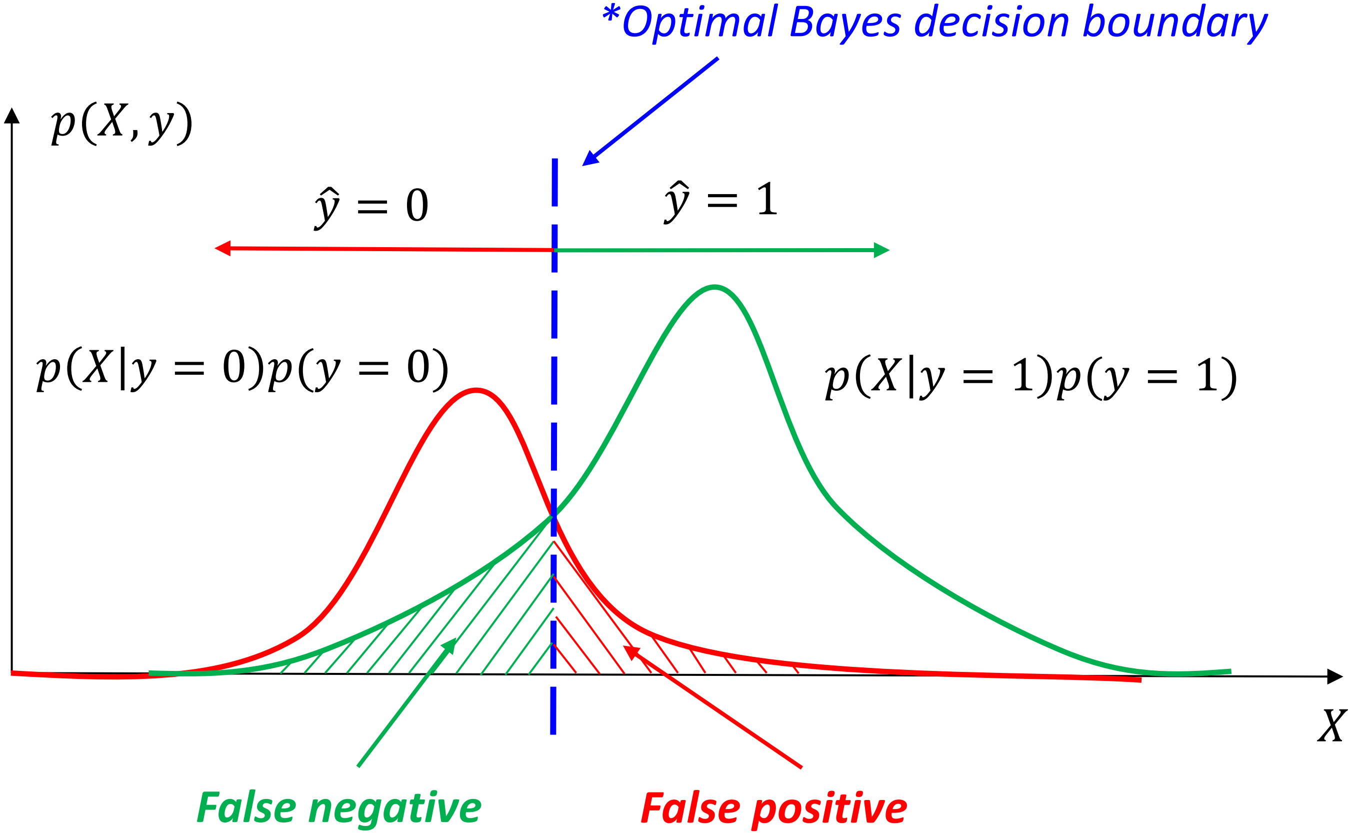
- When the data distribution is highly skewed, frequent classes might dominate majority vote
  - They occur more often in the k nearest neighbors just because they have large volume
- Solution
  - Weight the neighbors in voting
    - $w(x, x_i) = \frac{1}{|x - x_i|}$ or $w(x, x_i) = \cos(x, x_i)$

Summary of kNN

- Instance-based learning
  - No training phase
  - Assign label to a testing case by its nearest neighbors
  - Non-parametric
  - Approximate Bayes decision boundary in a local region
- Efficient computation
  - Locality sensitive hashing
    - Random projection

Recall optimal Bayes decision boundary

- $f(X) = argmax_y P(y|X)$



Estimating the optimal classifier

- $f(X) = argmax_y P(y|X)$
  $= argmax_y P(X|y)P(y)$

Requirement:

$|D| >> |Y| \times (2^V - 1)$

Class conditional density

Class prior

#parameters: $|Y| \times (2^V - 1)$ $|Y| - 1$

	text	information	identify	mining	mined	is	useful	to	from	apple	delicious	**Y**
D1	1	1	1	1	0	1	1	1	0	0	0	**1**
D2	1	1	0	0	1	1	1	0	1	0	0	**1**
D3	0	0	0	0	0	1	0	0	0	1	1	**0**

V binary features

We need to simplify this

- Features are conditionally independent given class labels
  - $p(x_1, x_2 | y) = p(x_2 | x_1, y) p(x_1 | y)$
    $= p(x_2 | y) p(x_1 | y)$
  - E.g.,
    $p(\text{‘}white\ house\text{’}, \text{‘}obama\text{’} | political\ news) = p(\text{‘}white\ house\text{’} | political\ news) \times p(\text{‘}obama\text{’} | political\ news)$

This does not mean ‘white house’ is independent of ‘obama’!

Conditional v.s. marginal independence

- Features are not necessarily marginally independent from each other
  - $p(\text{‘}white\ house\text{’}|\text{‘}obama\text{’}) > p(\text{‘}white\ house\text{’})$
- However, once we know the class label, features become independent from each other
  - Knowing it is already political news, observing ‘obama’ contributes little about occurrence of ‘while house’

Naïve Bayes classifier

- $$\begin{aligned} f(X) &= argmax_y P(y|X) \\ &= argmax_y P(X|y)P(y) \\ &= argmax_y \prod_{i=1}^{V} P(x_i|y)\, P(y) \end{aligned}$$

Class conditional density

Class prior

#parameters: $|Y|\times V$ $|Y|-1$

v.s.

$|Y|\times(2^V-1)$

Computationally feasible

Naïve Bayes classifier

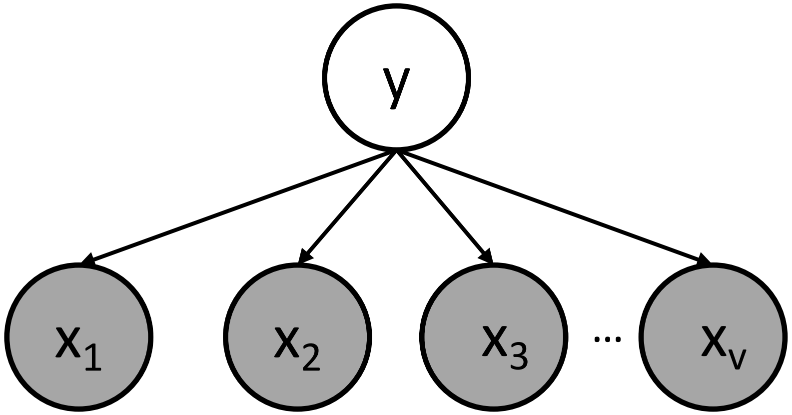
- $f(X) = argmax_y P(y|X)$

$$= argmax_y P(X|y)P(y)$$

By Bayes rule

$$= argmax_y \prod_{i=1}^{V} P(x_i|y)\, P(y)$$

By conditional independence assumption



Estimating parameters

- Maximial likelihood estimator
  - $P(x_i|y)$
  - $P(y)$

	text	information	identify	mining	mined	is	useful	to	from	apple	delicious	**Y**
D1	1	1	1	1	0	1	1	1	0	0	0	**1**
D2	1	1	0	0	1	1	1	0	1	0	0	**1**
D3	0	0	0	0	0	1	0	0	0	1	1	**0**

Enhancing Naïve Bayes for text classification I

- The frequency of words in a document matters
  - $P(X|y) = \prod_{i=1}^{|d|} P(x_i|y)^{c(x_i,d)}$
  - In log space

    Essentially, estimating $|Y|$ different language models!

    - $$f(y,X) = argmax_y \log P(y|X)$$
      $$= argmax_y \underbrace{\log P(y)}_{\text{Class bias}} + \sum_{i=1}^{|d|} \underbrace{c(x_i,d)}_{\text{Feature vector}} \underbrace{\log P(x_i|y)}_{\text{Model parameter}}$$

Enhancing Naïve Bayes for text classification I

- For binary case
  - $f(X) = sgn\left(\log \frac{P(y=1|X)}{P(y=0|X)}\right)$

$$= sgn\left(\log \frac{P(y=1)}{P(y=0)} + \sum_{i=1}^{|d|} c(x_i, d) \log \frac{P(x_i|y=1)}{P(x_i|y=0)}\right)$$

$$= sgn(w^T \bar{x})$$

a linear model with vector space representation?

where

$$w = \left(\log \frac{P(y=1)}{P(y=0)}, \log \frac{P(x_1|y=1)}{P(x_1|y=0)}, \ldots, \log \frac{P(x_v|y=1)}{P(x_v|y=0)}\right)$$

$$\bar{x} = (1, c(x_1, d), \ldots, c(x_v, d))$$

We will come back to this topic later.

Enhancing Naïve Bayes for text classification II

- Usually, features are not conditionally independent
  - $p(X|y) \neq \prod_{i=1}^{|d|} P(x_i|y)$
- Enhance the conditional independence assumptions by N-gram language models
  - $p(X|y) = \prod_{i=1}^{|d|} P(x_i|x_{i-1}, \dots, x_{i-N+1}, y)$

Enhancing Naïve Bayes for text classification III

- Sparse observation
  - $\delta\left(x_d^j = w_i, y_d = y\right) = 0 \Rightarrow p(x_i|y) = 0$
  - Then, no matter what values the other features take, $p(x_1, \dots, x_i, \dots, x_V|y) = 0$
- Smoothing class conditional density
  - All smoothing techniques we have discussed in language models are applicable here