

# kNN & Naïve Bayes

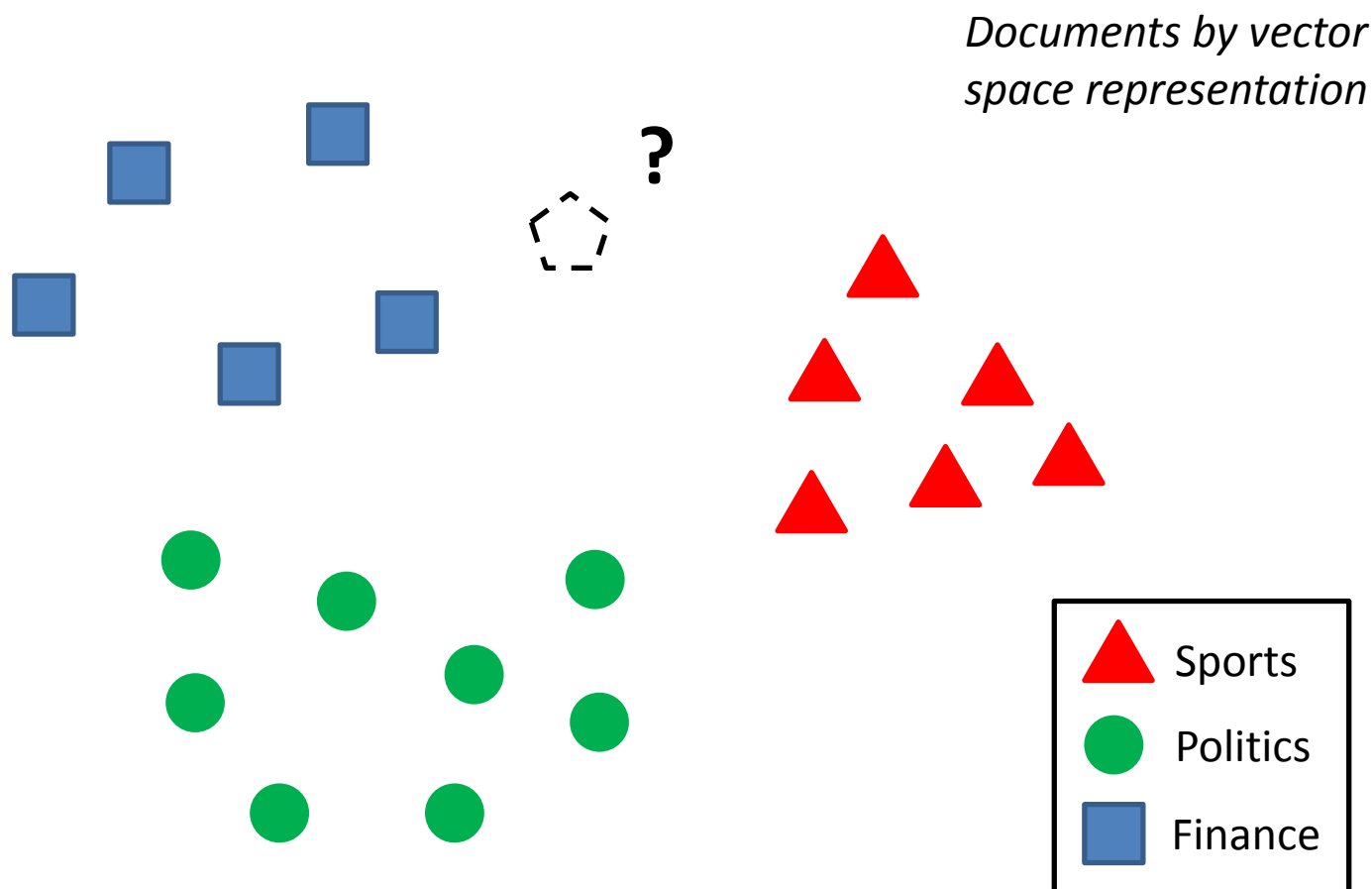
Hongning Wang

CS@UVa

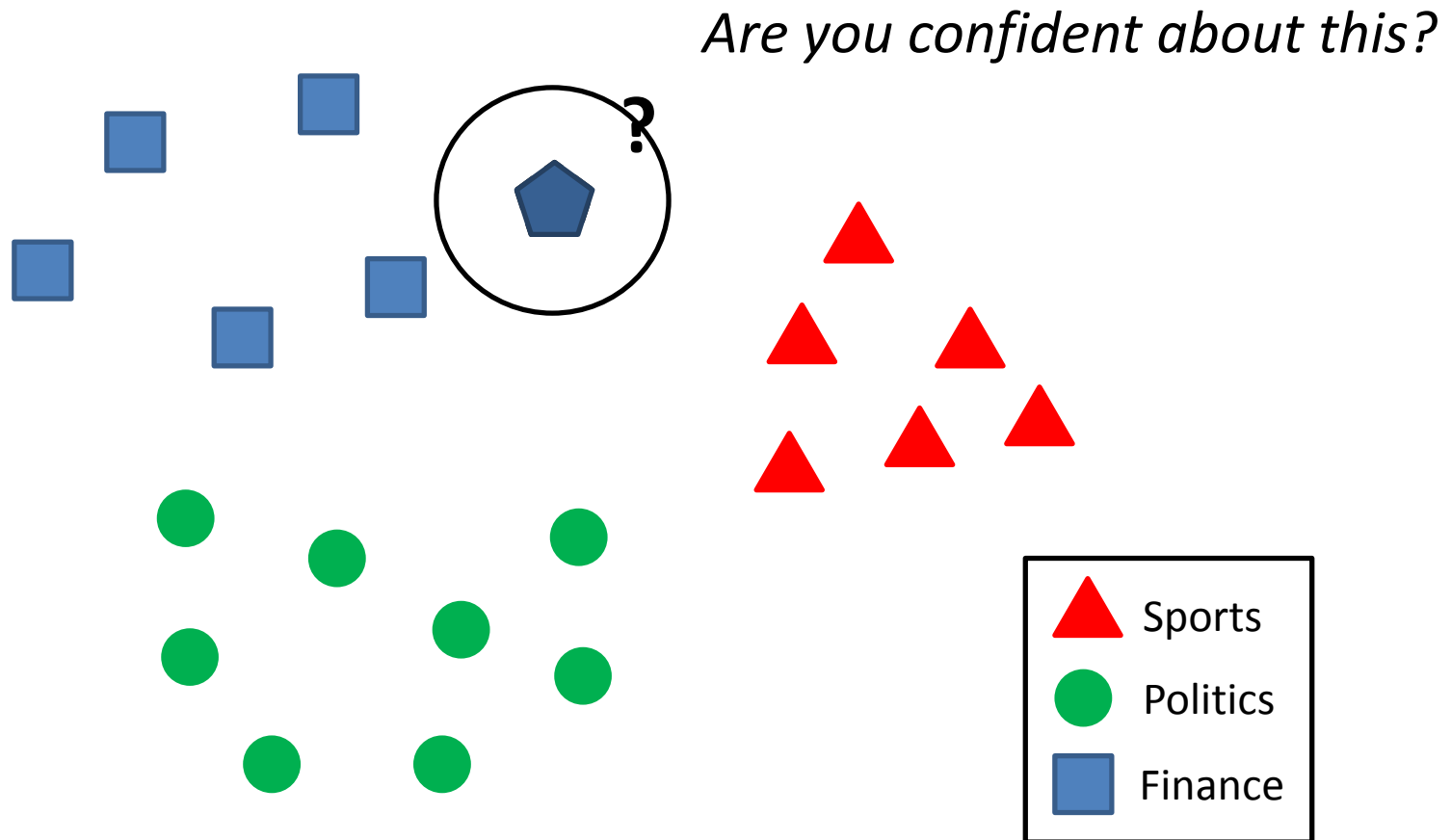
# Today's lecture

- 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?



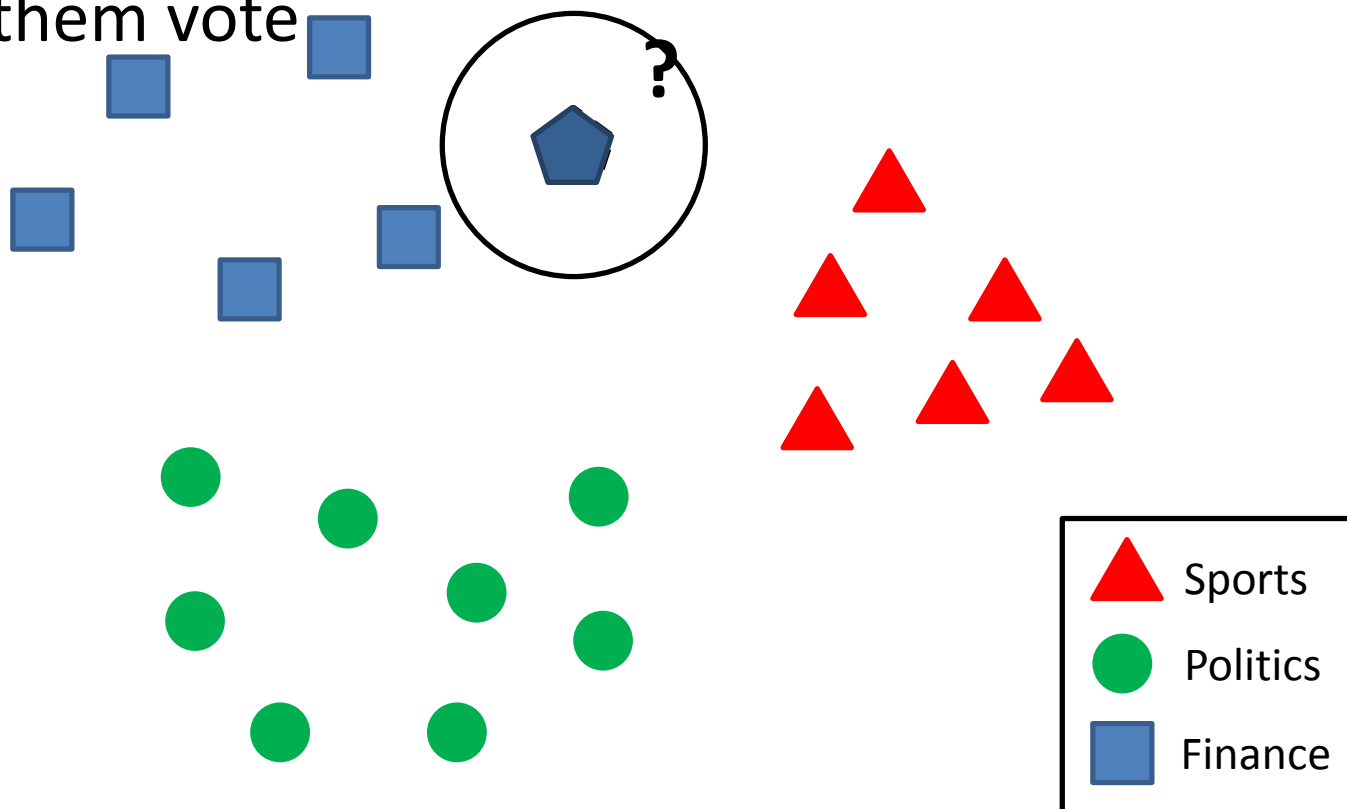
# Let's check the nearest neighbor



# 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

$$p(x)V = \frac{k}{N} \Rightarrow p(x) = \frac{k}{NV}$$

*Total number of instances*

$$p(x|y=1) = \frac{k_1}{N_1V}$$

*Nearest neighbors from class 1*

$$p(y=1) = \frac{N_1}{N}$$

*Total number of instances in class 1*

With Bayes rule:

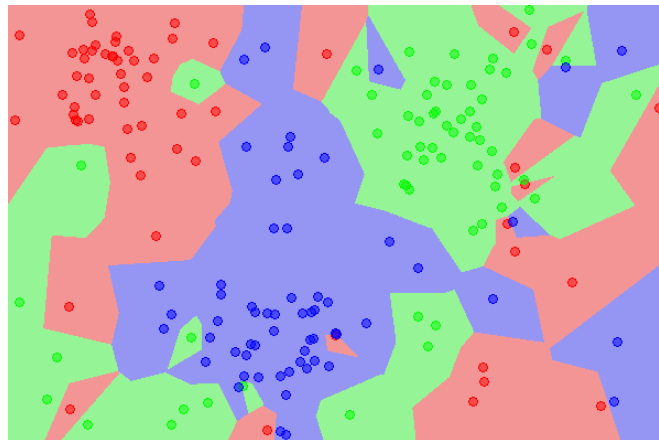
$$p(y=1|x) = \frac{\frac{N_1}{N} \times \frac{k_1}{N_1V}}{\frac{k}{NV}} = \frac{k_1}{k}$$

*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*



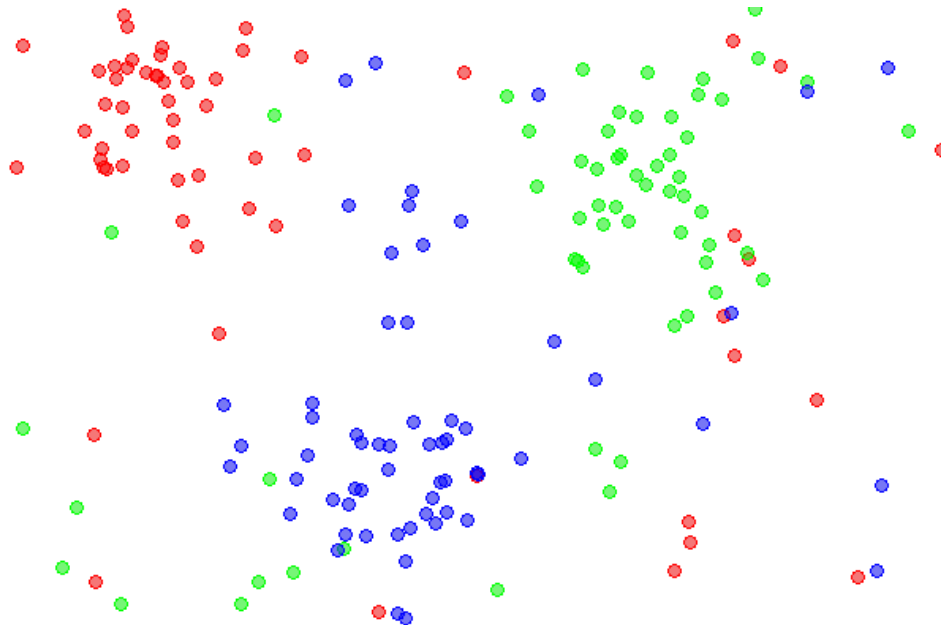
# Components in kNN

- 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

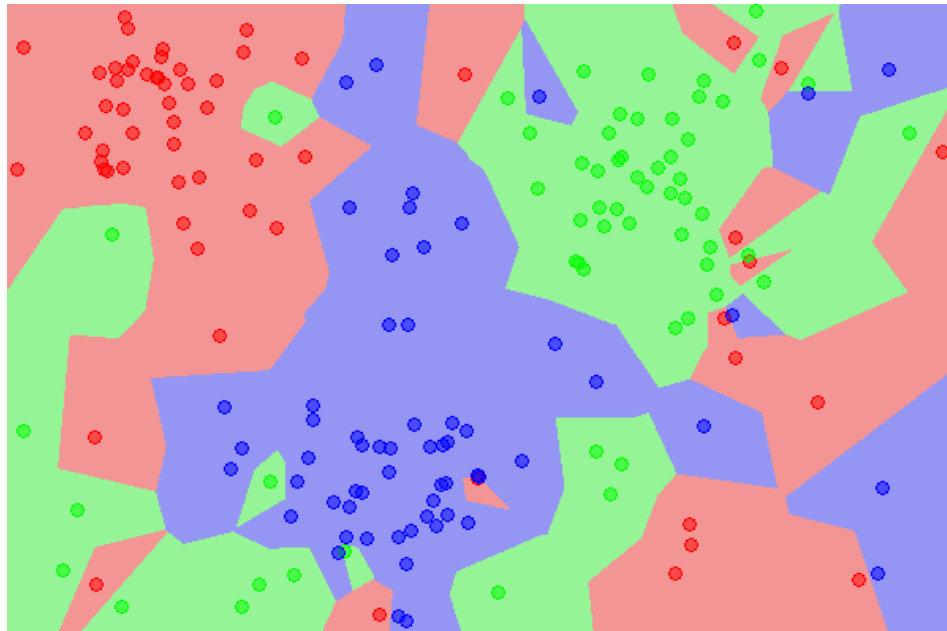
- Choice of k influences the “smoothness” of the resulting classifier



# Effect of k

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

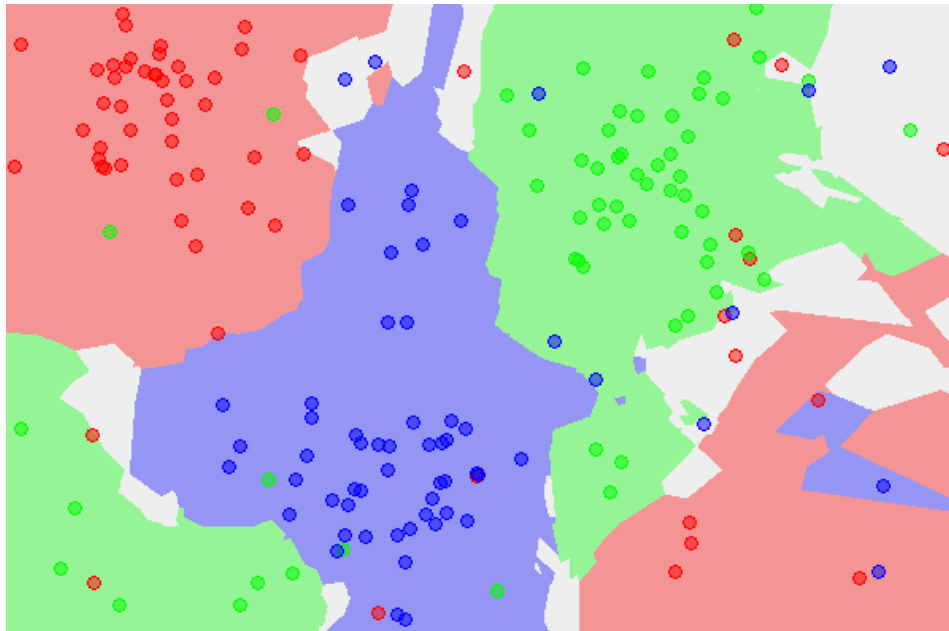
k=1



# Effect of k

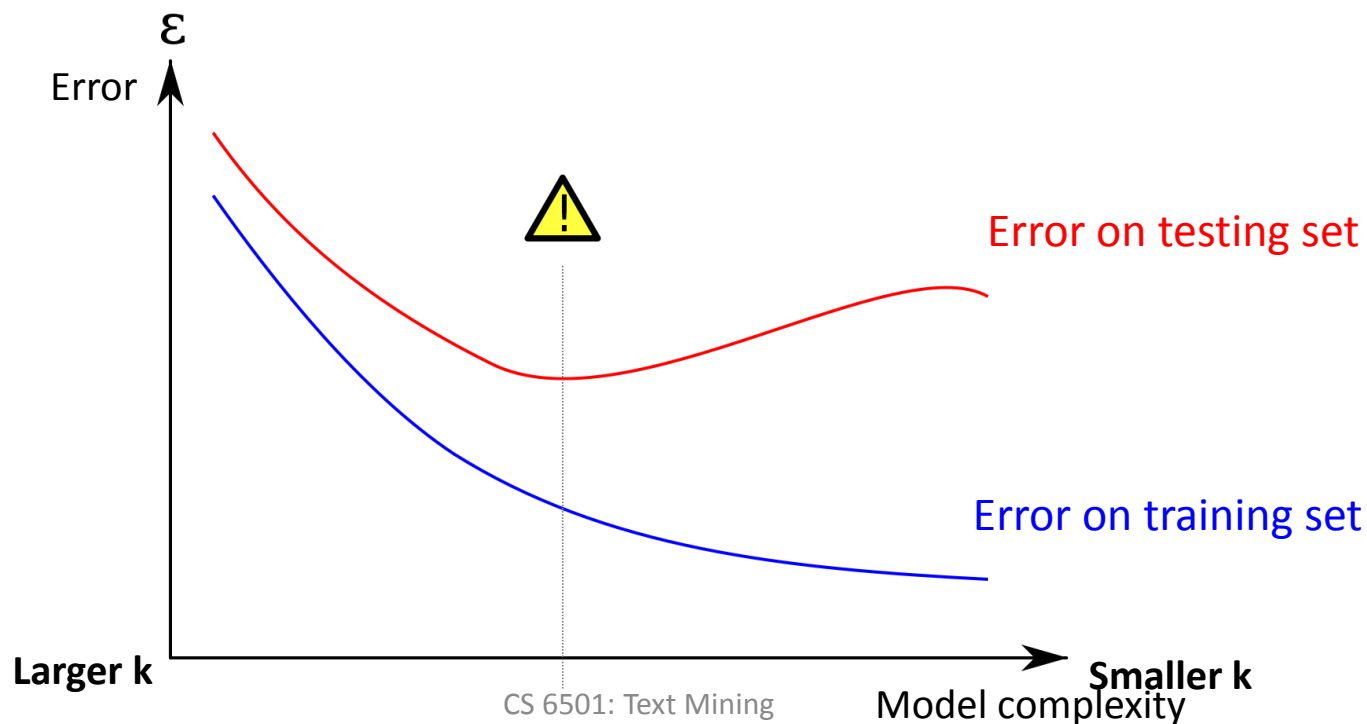
- Choice of k influences the “smoothness” of the resulting classifier

k=5

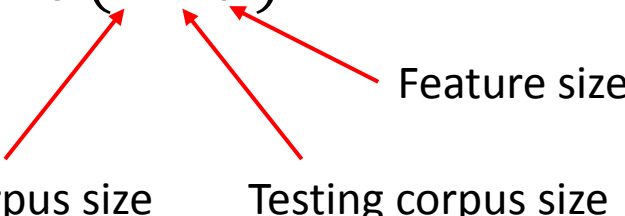


# Effect of k

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

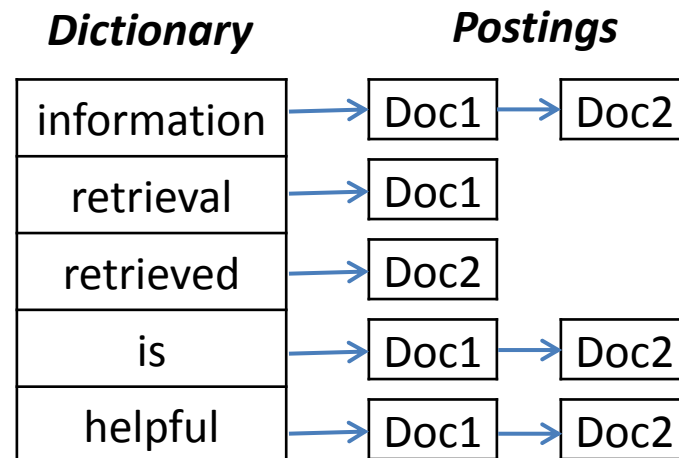


# Efficient instance look-up

- 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
    - $O(NMV)$ 
      - Training corpus size
      - Testing corpus size
      - Feature size

# Efficient instance look-up

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

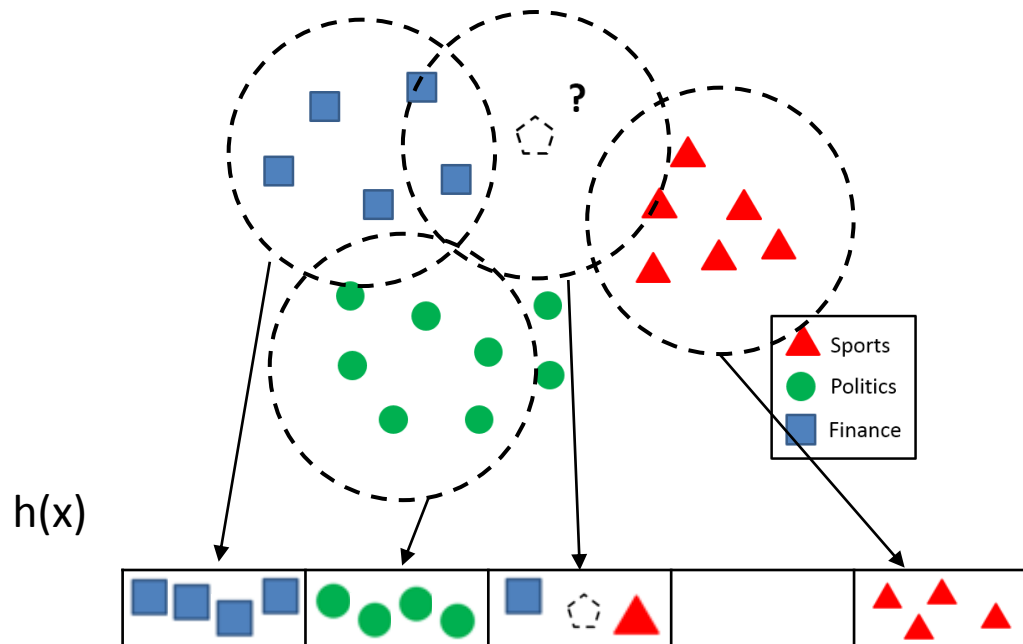


# Efficient instance look-up

- Exact solutions
  - Build inverted index for 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 solutions
  - Locality sensitive hashing
    - Similar documents -> (likely) same hash values





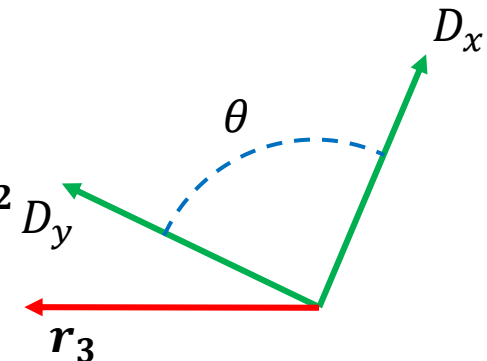
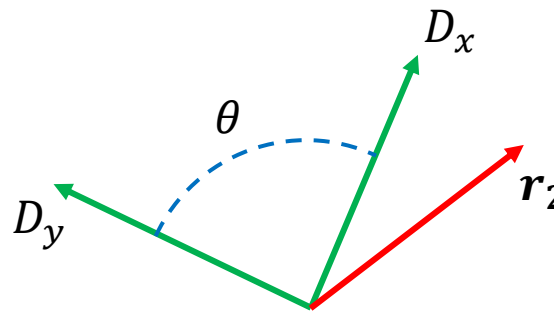
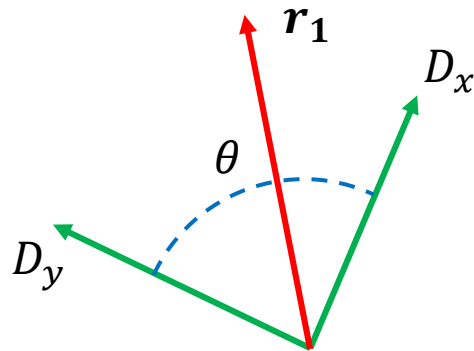
# Efficient instance look-up

- 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 high probability
      - Learning-based: learn the hash function with annotated examples, e.g., must-link, cannot-link
      - Random projection

# Random projection

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

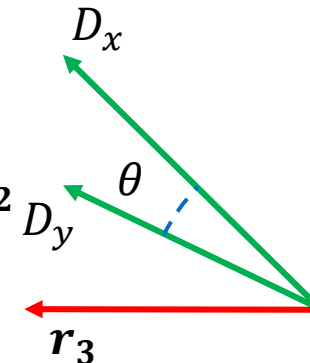
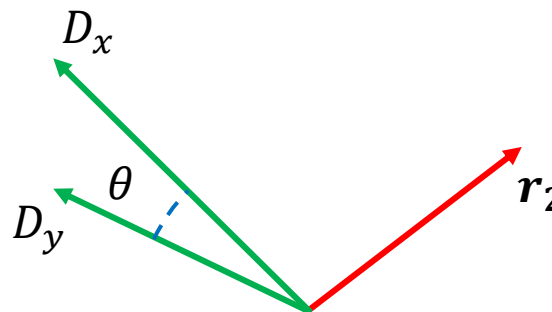
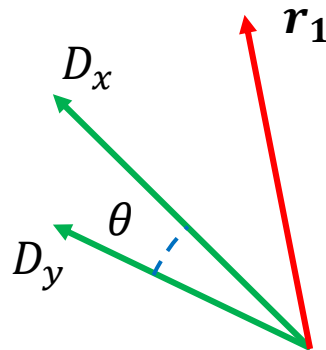
	$r_1$	$r_2$	$r_3$
$D_x$	1	1	0
$D_y$	1	0	1



# Random projection

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

	$r_1$	$r_2$	$r_3$
$D_x$	1	0	1
$D_y$	1	0	1

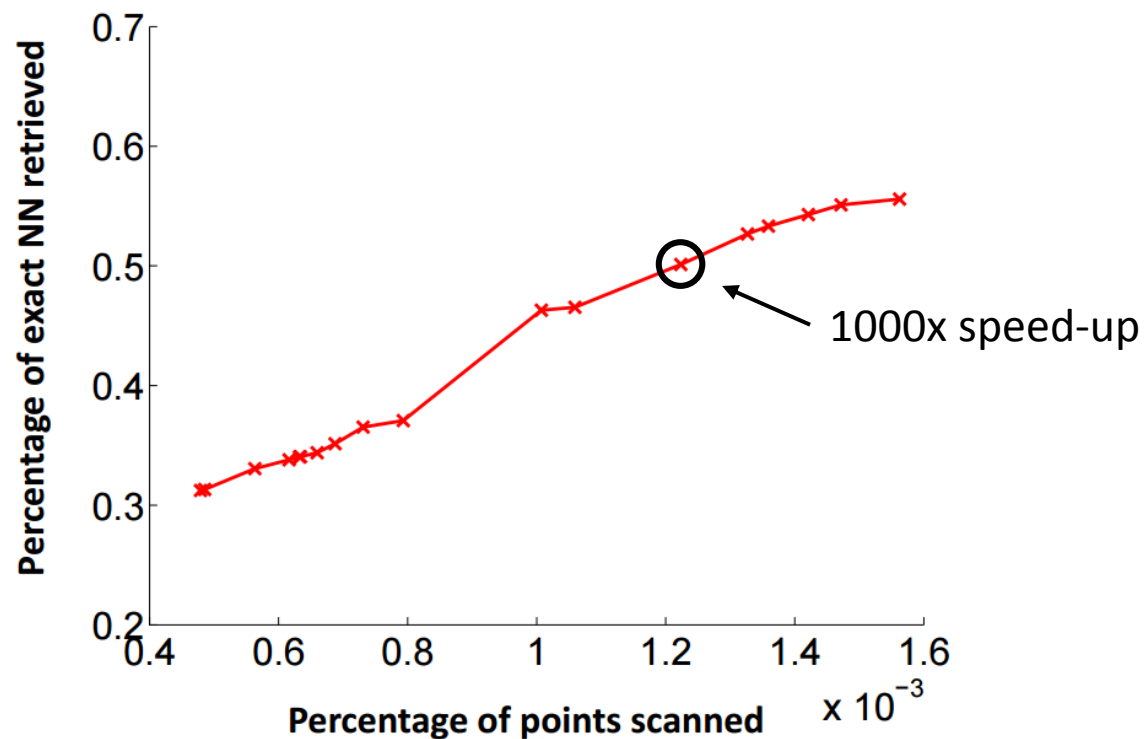


# Random projection

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

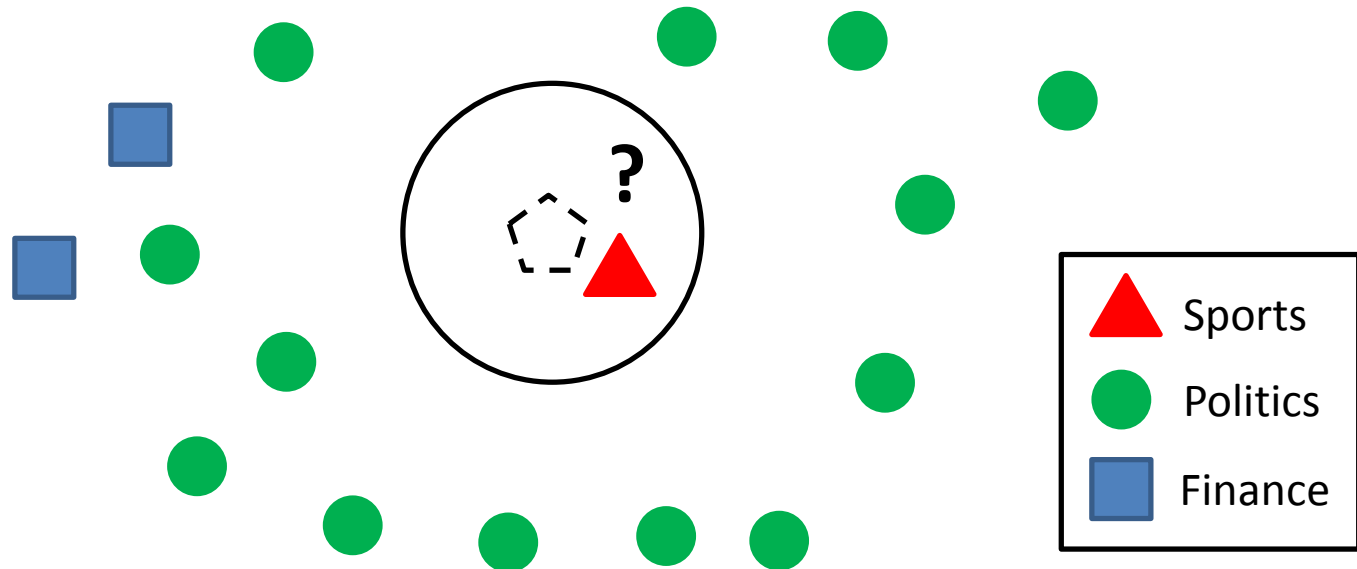
# Efficient instance look-up

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



# 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  $w(x, x_i) = \cos(x, x_i)$

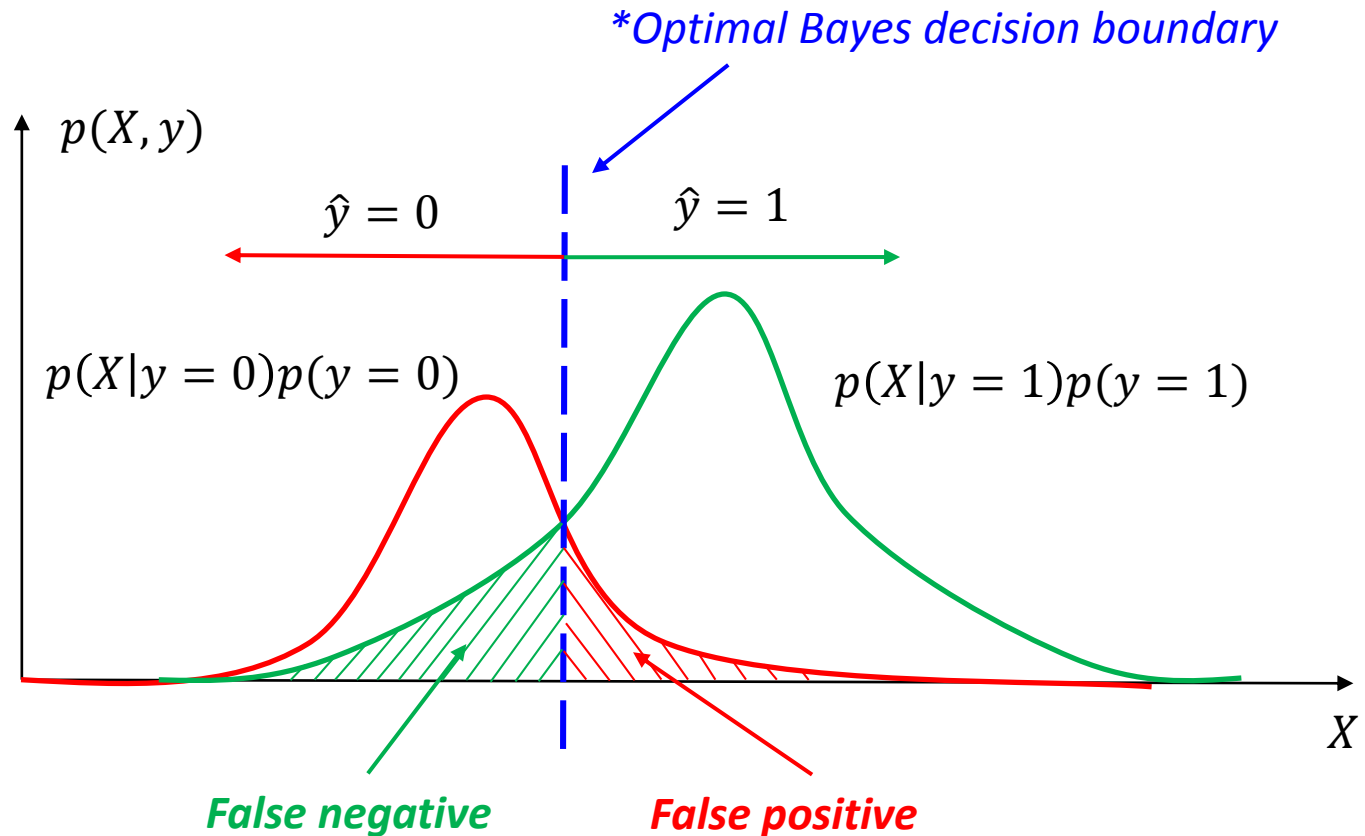
# Summary of kNN

- 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

- $f(X) = \operatorname{argmax}_y P(y|X)$



# Estimating the optimal classifier

- $$f(X) = \operatorname{argmax}_y P(y|X)$$

$$= \operatorname{argmax}_y P(X|y)P(y)$$

*Requirement:*

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

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 label
  - $p(x_1, x_2|y) = p(x_2|x_1, y)p(x_1|y)$   
 $= p(x_2|y)p(x_1|y)$
  - E.g.,  
 $p(\text{'white house'}, \text{'obama'} | \text{political news}) =$   
 $p(\text{'white house'} | \text{political news}) \times$   
 $p(\text{'obama'} | \text{political news})$

*This does not mean 'white house' is independent of 'obama'!*

# Conditional v.s. marginal independence

- Features are not necessarily marginally independent from each other
  - $p(\text{'white house'} | \text{'obama'}) > p(\text{'white house'})$
- 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

- $f(X) = \operatorname{argmax}_y P(y|X)$   
 $= \operatorname{argmax}_y P(X|y)P(y)$

$$= \operatorname{argmax}_y \prod_{i=1}^{|d|} P(x_i|y) P(y)$$

Class conditional density

Class prior

#parameters:

$$|Y| \times V$$

$$|Y| - 1$$

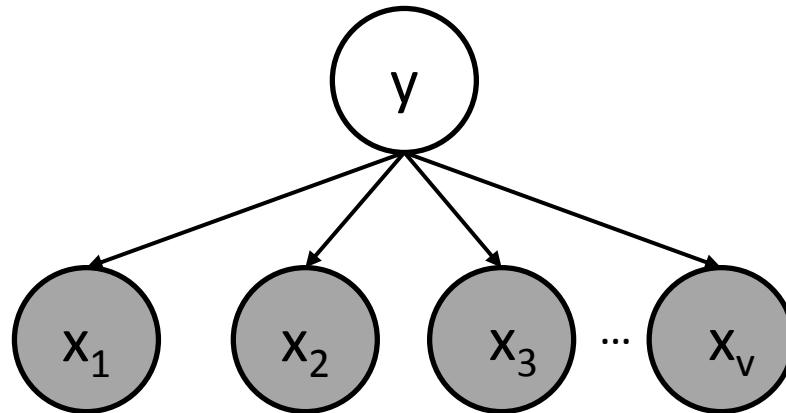
v.s.

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

**Computationally feasible**

# Naïve Bayes classifier

- $f(X) = \operatorname{argmax}_y P(y|X)$   
 $= \operatorname{argmax}_y P(X|y)P(y)$  *By Bayes rule*  
 $= \operatorname{argmax}_y \prod_{i=1}^{|d|} P(x_i|y) P(y)$  *By independence assumption*



# Estimating parameters

- Maximal likelihood estimator

$$- P(x_i|y) = \frac{\sum_d \sum_j \delta(x_d^j = x_i, y_d = y)}{\sum_d \delta(y_d = y)}$$

$$- P(y) = \frac{\sum_d \delta(y_d = y)}{\sum_d 1}$$

# Enhancing Naïve Bayes for text classification

- 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!*

- $f(X) = \operatorname{argmax}_y \log P(y|X)$   
 $= \operatorname{argmax}_y \log P(y) + \sum_{i=1}^{|d|} c(x_i, d) \log P(x_i|y)$

Class bias

Feature vector

Model parameter



# Enhancing Naïve Bayes for text classification

- For binary case

$$\begin{aligned} -f(X) &= \text{sgn} \left( \log \frac{P(y = 1|X)}{P(y = 0|X)} \right) \\ &= \text{sgn} \left( \log \frac{P(y = 1)}{P(y = 0)} + \sum_{i=1}^{|d|} c(x_i, d) \log \frac{P(x_i|y = 1)}{P(x_i|y = 0)} \right) \\ &= \text{sgn}(w^T \bar{x}) \end{aligned}$$

a linear model with vector space representation?

where

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

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

We will come back to this topic later.

# Enhancing Naïve Bayes for text classification

- 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

- Sparse observation
  - $\delta(x_d^j = x_i, y_d = y) = 0 \Rightarrow p(x_i|y) = 0$
  - 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

# Maximum a Posterior estimator

- Adding pseudo instances

- Priors:  $q(y)$  and  $q(x, y)$

*Can be estimated from a related corpus or manually tuned*

- MAP estimator for Naïve Bayes

- $$P(x_i|y) = \frac{\sum_d \sum_j \delta(x_d^j = x_i, y_d = y) + Mq(x_i, y)}{\sum_d \delta(y_d = y) + Mq(y)}$$

#pseudo instances

# Summary of Naïve Bayes

- Optimal Bayes classifier
  - Naïve Bayes with independence assumptions
- Parameter estimation in Naïve Bayes
  - Maximum likelihood estimator
  - Smoothing is necessary

# Today's reading

- Introduction to Information Retrieval
  - Chapter 13: Text classification and Naive Bayes
    - 13.2 – Naive Bayes text classification
    - 13.4 – Properties of Naive Bayes
  - Chapter 14: Vector space classification
    - 14.3 k nearest neighbor
    - 14.4 Linear versus nonlinear classifiers