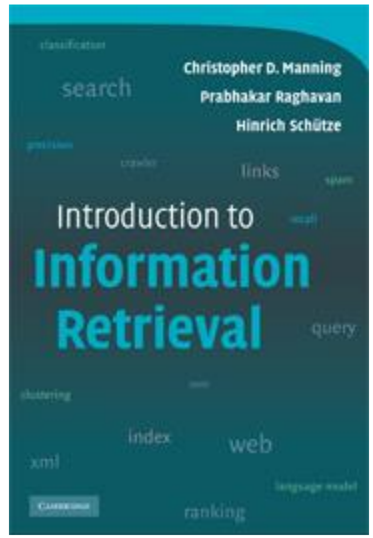
Information Retrieval and Organisation



Chapter 14

Vector Space Classification

Dell Zhang
Birkbeck, University of London

Recall: Vector Space Model

- Docs → Vectors (Points)
  - Each doc can now be viewed as a vector with one component for each term (TFxIDF weights).
  - Usually normalized to unit length.
- So we have a high-dimensional **vector space**
  - Terms are axes
  - May have 10,000+ dimensions, or even 100,000+
    - even with stemming
  - Docs live in this space
- How can we do classification in this space?

Classification based on VSM

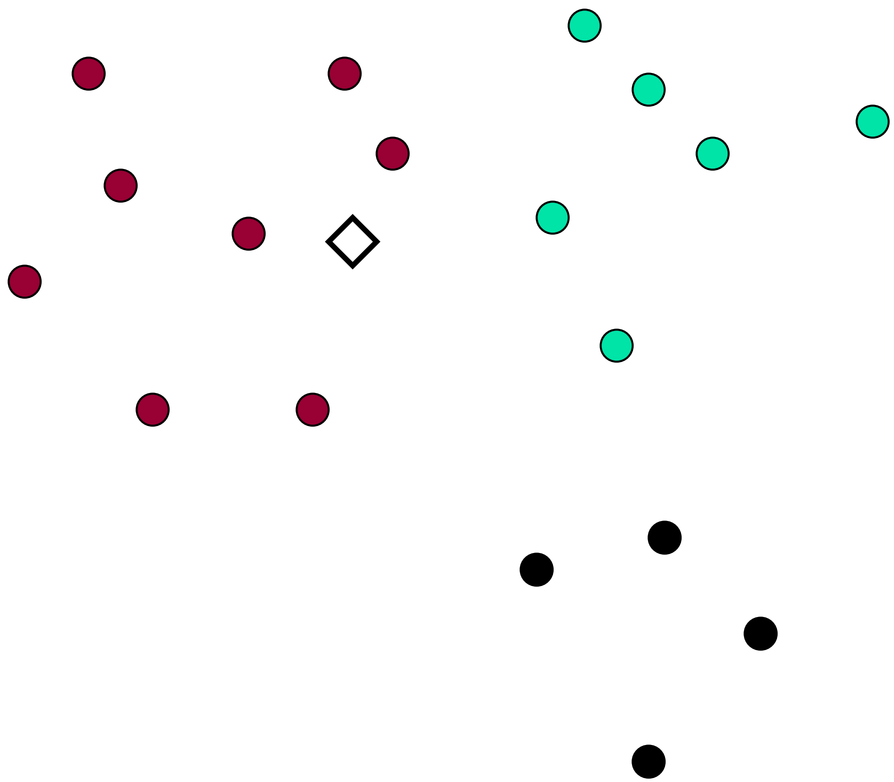
- As before, the training set is a set of documents, each labeled with its class (e.g., topic)
- In vector space classification, this set corresponds to a labeled set of points (or, equivalently, vectors) in the vector space
  - *Premise 1*: Documents in the same class form a contiguous region of space
  - *Premise 2*: Documents from different classes don't overlap (much)
- We define surfaces to delineate classes in the space

k Nearest Neighbors (kNN)

- Given a test doc d and the training data
  - identify the set S_k of the k nearest neighbors of d, i.e., the k training docs most similar to d.
  - for each class $c_j \in C$
    - compute $N(S_k, c_j)$ the number of S_k members that belong to class c_j
    - estimate $\Pr[c_j|d]$ as $N(S_k, c_j)/k$
  - classify d to the *majority* class of S_k memebers.

$$c(d) = \underset{c_j \in C}{\operatorname{argmax}} \Pr[c_j \mid d] = \underset{c_j \in C}{\operatorname{argmax}} N(S_k, c_j)$$

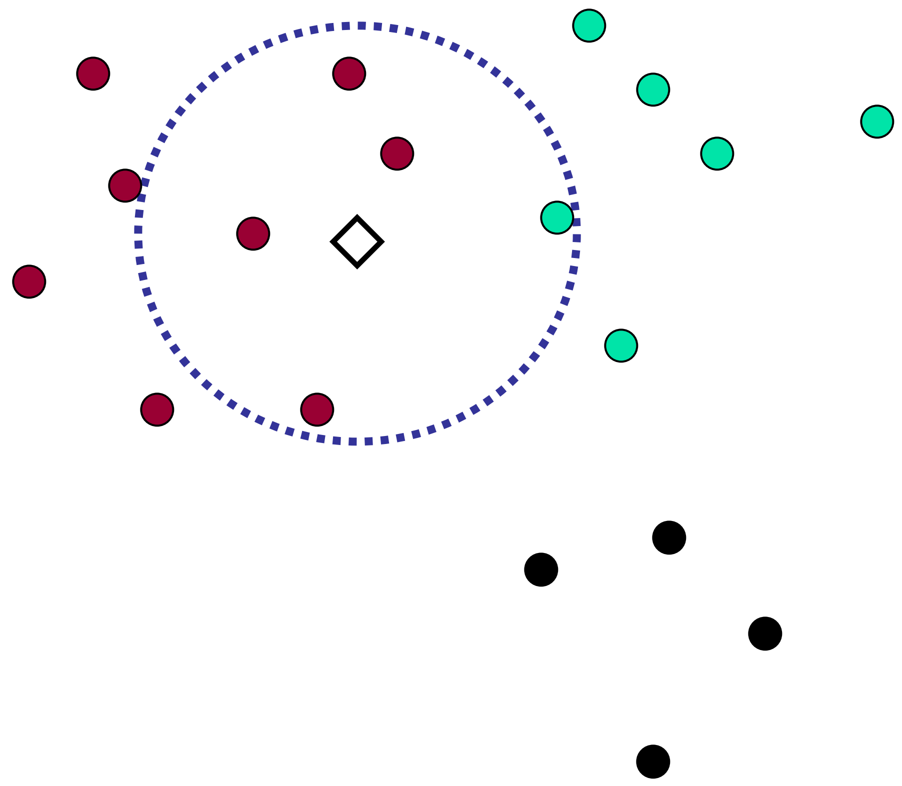
*k*NN – Example



c(◇) = ?

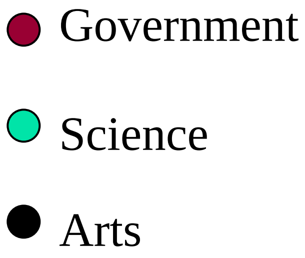
- Government
- Science
- Arts

*k*NN – Example

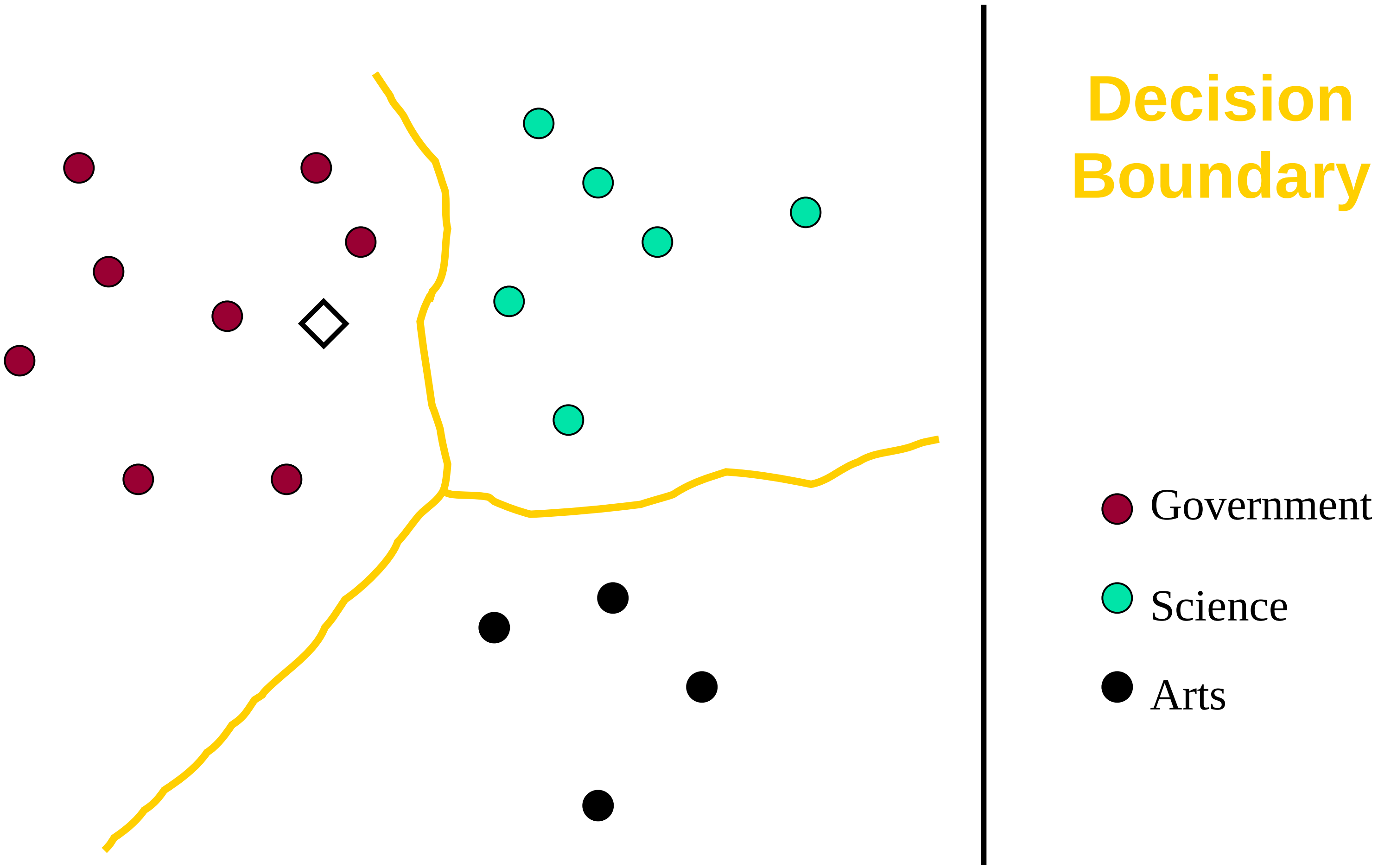


5NN (*k*=5)

c(◇) = ?



*k*NN – Example



kNN Algorithm

TRAIN-KNN($\mathbb{C}$, $\mathbb{D}$)

$\mathbb{D}' \leftarrow$ PREPROCESS($\mathbb{D}$)

$k \leftarrow$ SELECT-K($\mathbb{C}$, $\mathbb{D}'$)

return $\mathbb{D}', k$

APPLY-KNN($\mathbb{C}$, $\mathbb{D}'$, k, d)

$S_k \leftarrow$ COMPUTENEARESTNEIGHBORS($\mathbb{D}'$, k, d)

for each $c_j \in \mathbb{C}$

do $p_j \leftarrow |S_k \cap c_j|/k$

return $\arg\max_j p_j$

Parameter k

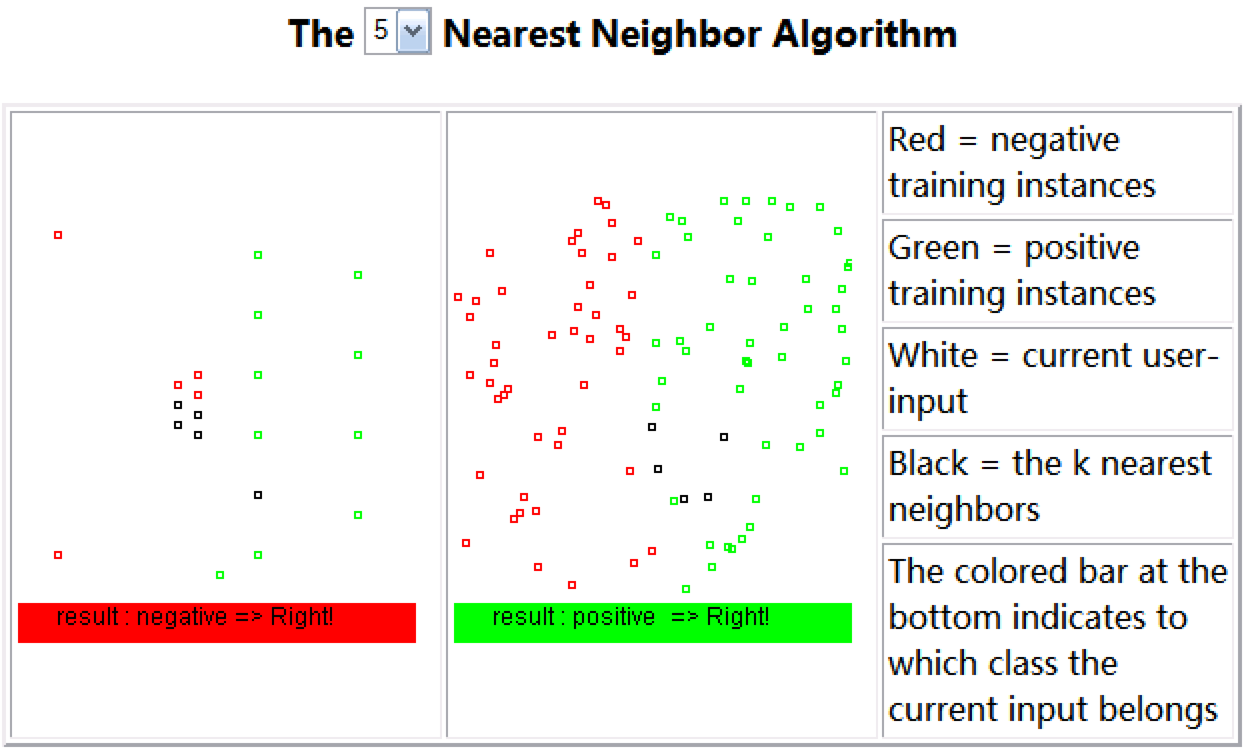
- $k = 1$
  - Using only the nearest neighbor to determine classification is often error-prone due to:
    - Atypical training documents
    - Noise (i.e. error) in the class labels
- $k = N$
  - Every test doc would be classified into the largest class in spite of its content.
    - Degenerate to classification using *priori* probabilities $P(c_j)$

Parameter k

- $1 < k < N$
  - More robust with a moderate value of k
  - The value of k is typically odd to avoid ties
    - 3 and 5 are most common

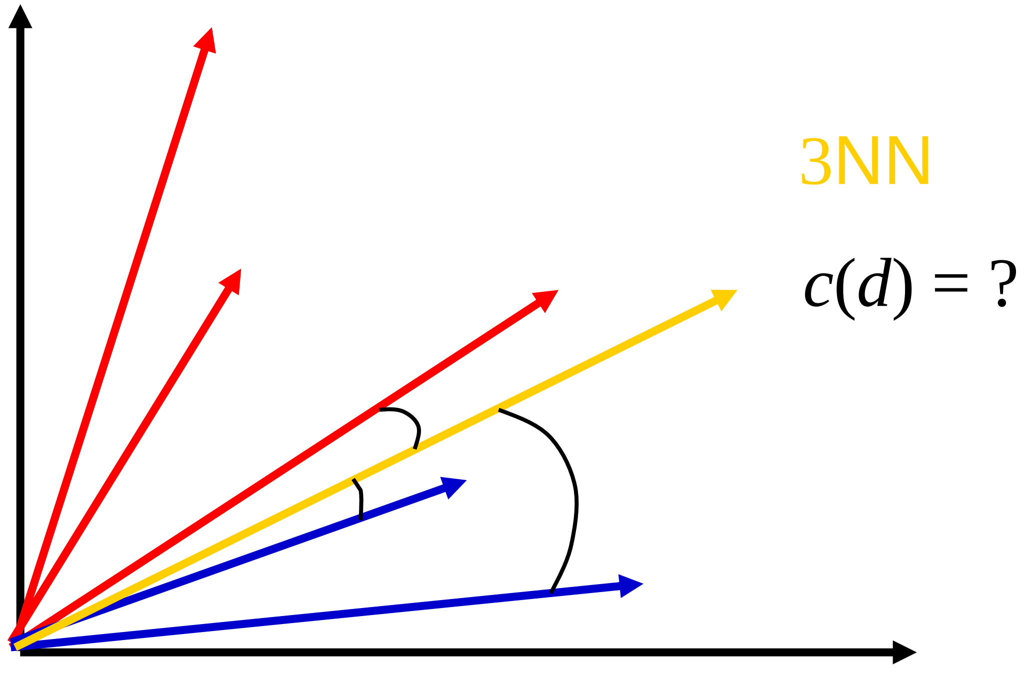
*k*NN – Online Demo

- http://www.comp.lancs.ac.uk/~kristof/research/notes/nearb/cluster.html



Similarity Metric

- *k*NN depends on a similarity/distance metric
  - For text, *cosine* similarity of TFxIDF weighted vectors is usually most effective.



kNN Works

- Effectiveness
  - More training documents lead to better accuracy, though lower speed
  - kNN is close to optimal
    - Asymptotically, the error rate of 1NN classification is less than twice the error rate of the Bayes optimal classifier.

*k*NN Works

- Efficiency
  - Lazy Learning or Memory-based Learning or Case-based Learning
    - No training (except for data preprocessing etc.)
    - More expensive testing
  - *S*cales well with the number of classes
    - Don’t need to train n classifiers for n classes

kNN Works

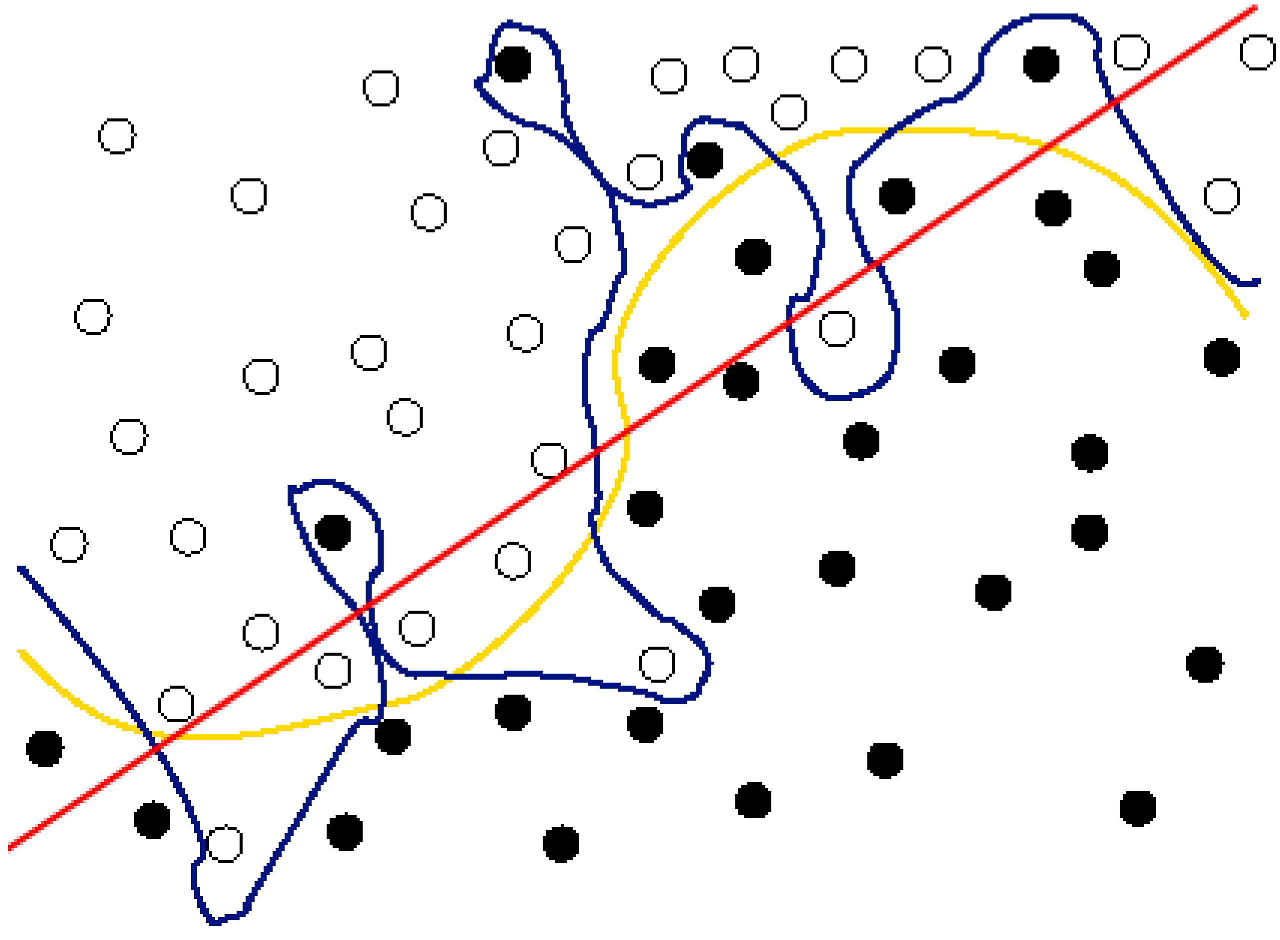
- Efficiency
  - kNN with Inverted Index
    - Naively finding the kNN of a test doc d requires a scan through all training docs.
    - But this is actually same as finding the top k retrieval results using d as a (long) query to the collection of training docs.
    - Therefore the standard inverted index method for VSM retrieval could be used to accelerate this process.

kNN vs. NB

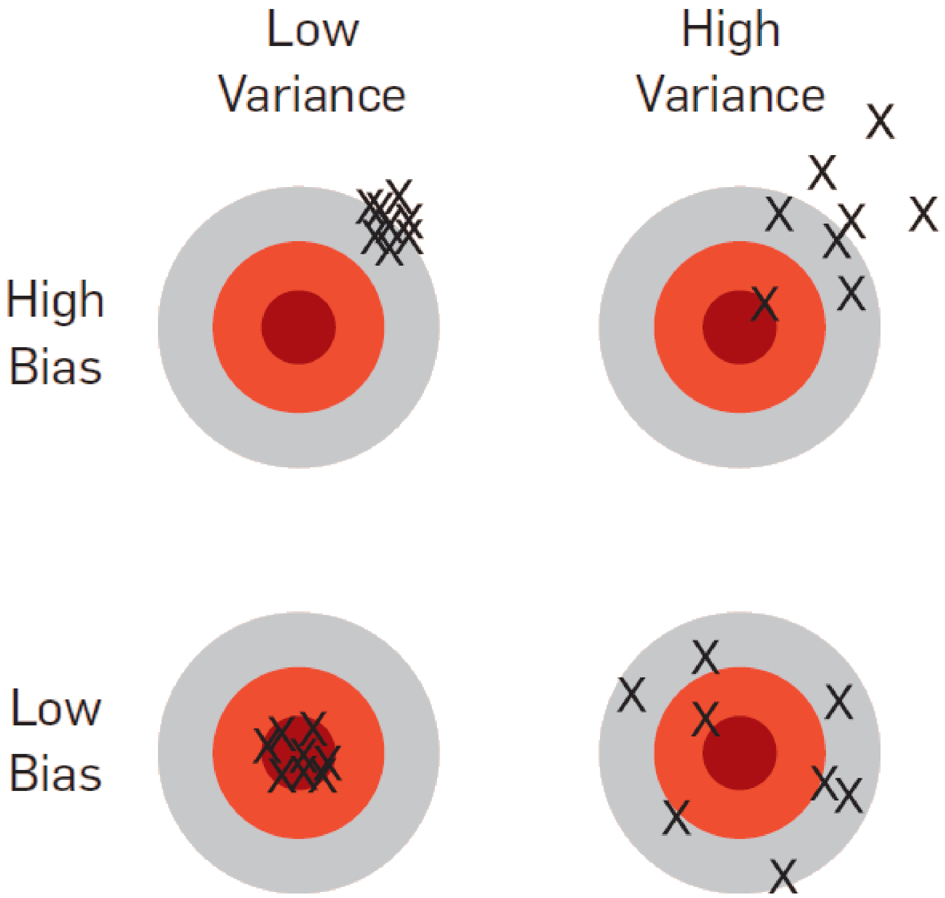
- kNN has *high variance* and *low bias*
  - Decision boundary can be arbitrary
- NB has *low variance* and *high bias*
  - Decision boundary has to be linear (hyperplane)

Variance ≈ Capacity

Bias/Variance Tradeoff



Bias/Variance in Dart-Throwing



Bias/Variance Tradeoff

- Consider asking a botanist: *Is an object a tree?*
  - Too much variance, low bias
    - A botanist who just memorizes
    - Says “no” to any new object (e.g., different # of leaves)
  - Not enough variance, high bias
    - A botanist who is very lazy
    - Says “yes” as long as the object is green
  - You want the middle ground
    - Choose the correct model capacity!

(Example due to C. Burges)

Which Classifier Shall I Use?

- Is there a learning method that is optimal for all text classification problems?
  - No, because there is a tradeoff between bias and variance.
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem?
    - e.g., linear vs. nonlinear decision boundary
  - How noisy is the problem? How stable is the problem over time?
    - It would better to use a simple and robust classifier for a noisy or unstable problem.

More Than Two Classes

- Any-of classification
  - Classes are independent of each other
  - A document can belong to 0, 1, or >1 classes
    - Decomposes into *n* binary problems
    - Quite common for documents
- One-of classification
  - Classes are mutually exclusive
  - A document belongs to exactly one class
    - For example, hand-written digit recognition

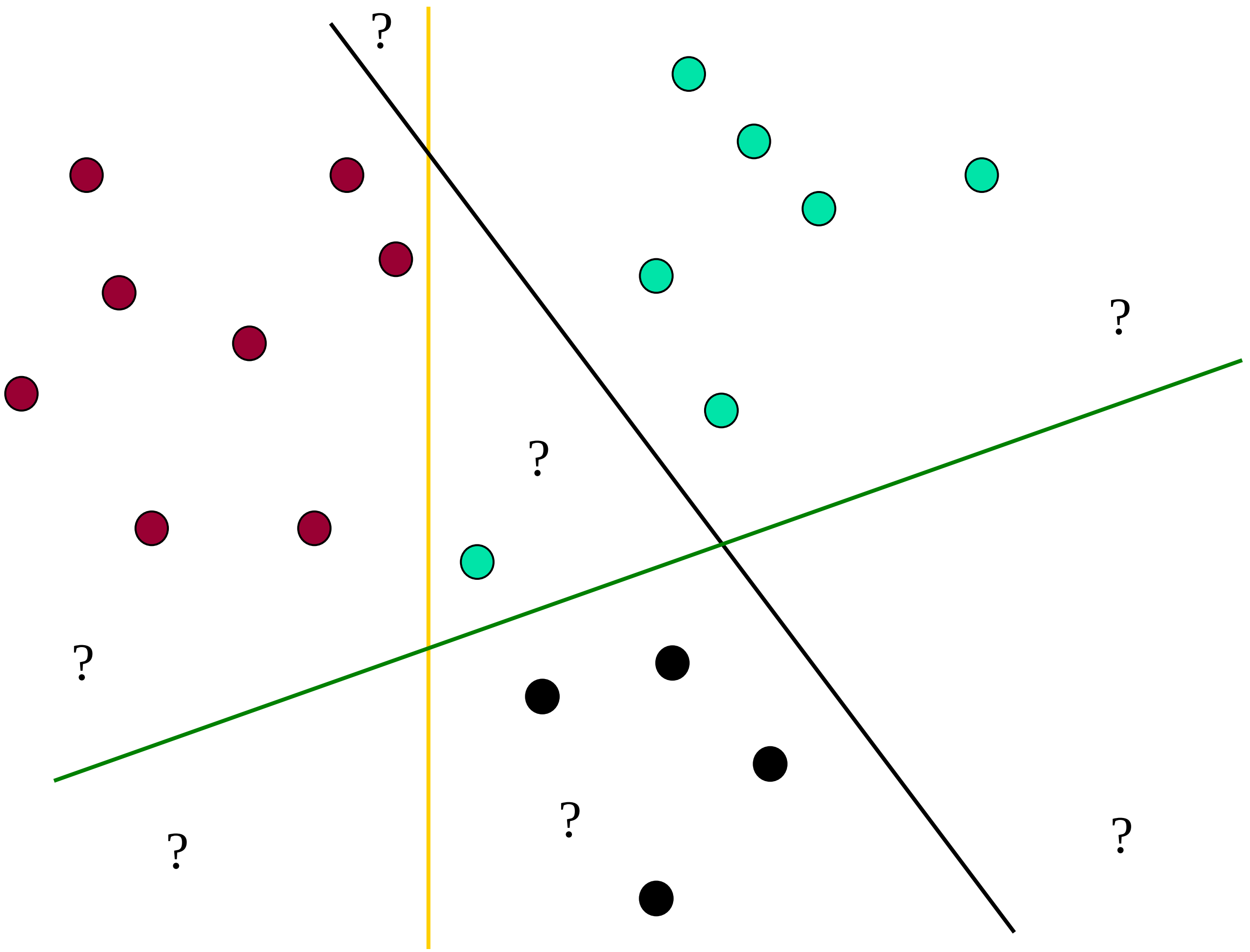
Any-of Classification

- One-vs-Rest Ensemble
  - Build a binary classifier between each class and its complementary set (docs from all other classes).
- Given test doc, evaluate it for membership in each class
- Apply decision criterion of classifiers independently

One-of Classification

- One-vs-Rest Ensemble
  - Build a binary classifier between each class and its complementary set (docs from all other classes).
- Given test doc, evaluate it for membership in each class.
- Assign document to the class with:
  - maximum score
  - maximum confidence
  - maximum probability

Any-of vs. One-of



Tools

WEKA
The University
of Waikato