

K nearest neighbor

LING 572 Advanced Statistical Methods for NLP

Shane Steinert-Threlkeld

January 16, 2020

The term “weight” in ML

- Weights of features
- Weights of instances
- Weights of classifiers

The term “binary” in ML

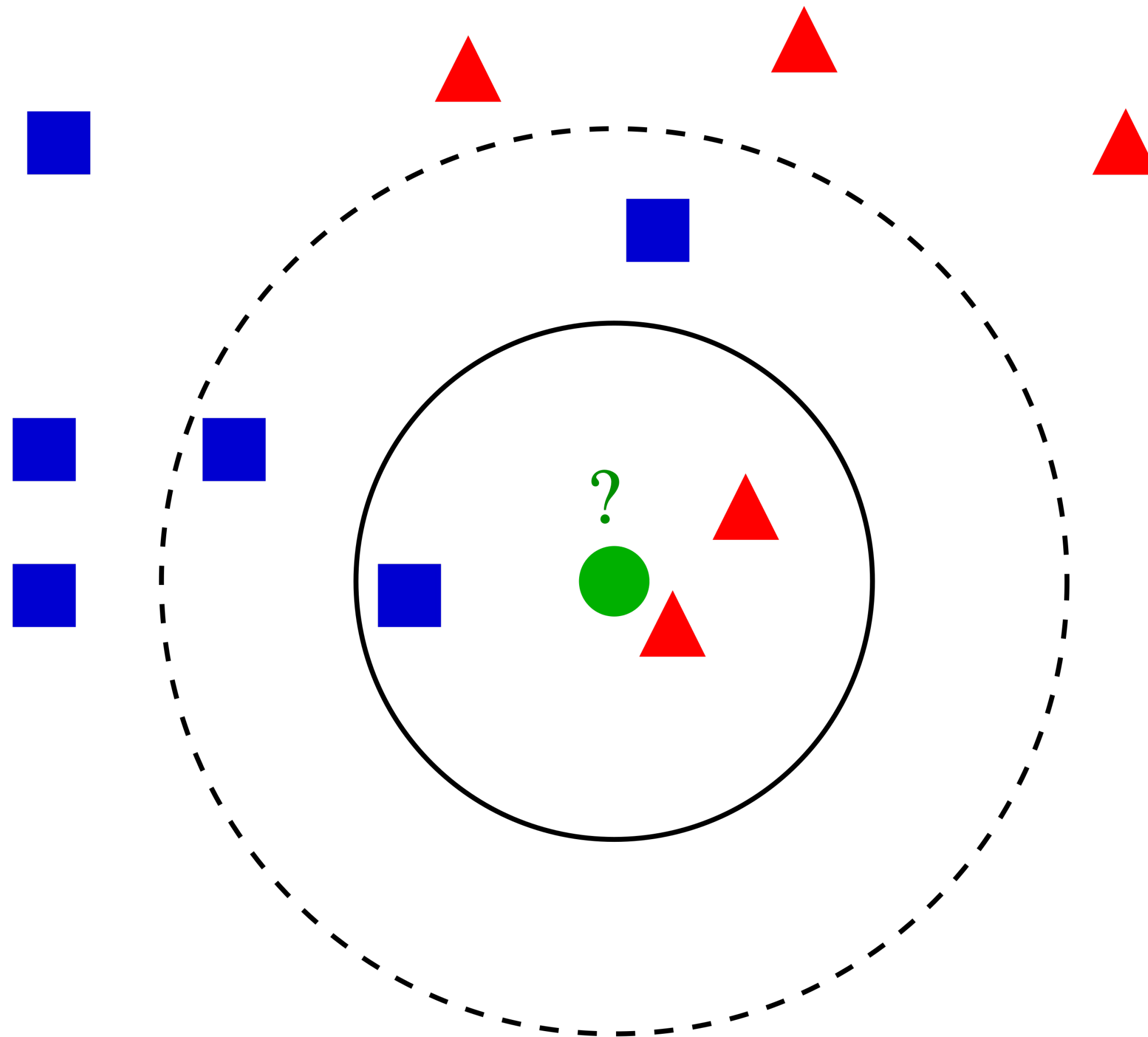
- Classification problem:
 - Binary: the number of classes is 2
 - Multi-class: the number of classes is > 2
- Features:
 - Binary: the number of possible feature values is 2.
 - Categorical / discrete: > 2 values
 - Real-valued / scalar / continuous: the feature values are real numbers
- File format:
 - Binary: human un-readable
 - Text: human readable

kNN

Instance-based (IB) learning

- No training: store all training instances.
 - “Lazy learning”
- Examples:
 - kNN
 - Locally weighted regression
 - Case-based reasoning
 - ...
- The most well-known IB method: kNN

kNN



kNN

- Training: record labeled instances as feature vectors
- Test: for a new instance d ,
 - find k training instances that are **closest** to d .
 - perform majority voting or weighted voting.
- Properties:
 - A “lazy” classifier. No learning in the training stage.
 - Feature selection and distance measure are crucial.

The algorithm

- Determine parameter K
- Calculate the distance between the test instance and all the training instances
- Sort the distances and determine K nearest neighbors
- Gather the labels of the K nearest neighbors
- Use simple majority voting or weighted voting.

Issues

- What's K ?
- How do we weight/scale/select features?
- How do we combine instances by voting?

Picking K

- Split the data into
 - Training data
 - Dev/val data
 - Test data
- Pick k with the lowest error rate on the validation set
 - use N-fold cross validation if the training data is small

Normalizing attribute values

- Distance could be dominated by some attributes with large numbers:
 - Example: features: age, income
 - Original data: $x_1=(35, 76K)$, $x_2=(36, 80K)$, $x_3=(70, 79K)$
- Rescale: i.e., normalize to $[0,1]$
 - Assume: age $\in [0,100]$, income $\in [0, 200K]$
 - After normalization: $x_1=(0.35, 0.38)$,
 $x_2=(0.36, 0.40)$, $x_3 = (0.70, 0.395)$.

The Choice of Features

- Imagine there are 100 features, and only 2 of them are relevant to the target label.
- Differences in irrelevant features likely to dominate:
 - kNN is easily misled in high-dimensional space.
 - Feature weighting or feature selection is key (It will be covered next time)

Feature weighting

- Reweighting a dimension j by weight w_j
 - Can increase or decrease weight of feature on that dimension
 - Setting w_j to zero eliminates this dimension altogether.
- Use (cross-)validation to automatically choose weights $w_1, \dots, w_{|F|}$

Some distance measures

- Euclidean distance:

$$d(d_i, d_j) = \|d_i - d_j\|_2^2 = \sqrt{\sum_k (d_{i,k} - d_{j,k})^2}$$

- Weighted Euclidean distance:

$$d(d_i, d_j) = \sqrt{\sum_k w_k (a_{i,k} - a_{j,k})^2}$$

- Cosine:

$$\cos(d_i, d_j) = \frac{d_i \cdot d_j}{\|d_i\|_2 \|d_j\|_2}$$

Voting by k-nearest neighbors

- Suppose we have found the k-nearest neighbors.
- Let $f_i(x)$ be the class label for the i -th neighbor of x .

$$\delta(c, f_i(x)) = \begin{cases} 1 & f_i(x) = c \\ 0 & \text{otherwise} \end{cases}$$
$$g(c) = \sum_i \delta(c, f_i(x))$$

that is, $g(c)$ is the number of neighbors with label c .

Voting

- Majority voting: $c^* = \arg \max_c g(c)$
- Weighted voting: weighting is on each neighbor

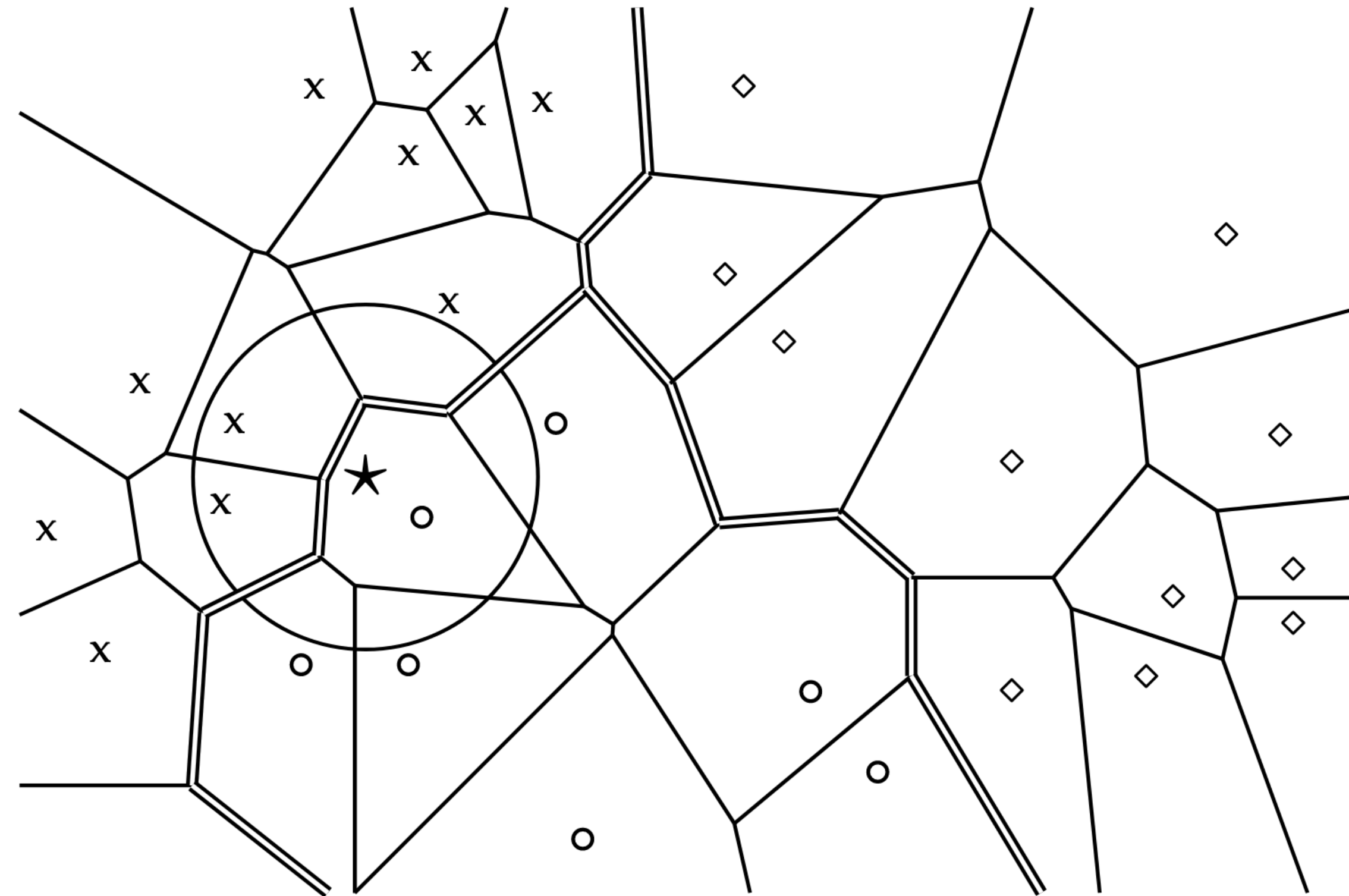
$$c^* = \arg \max_c \sum_i w_i \delta(c, f_i(x))$$

- Weighted voting allows us to use more training examples, e.g.:

$$w_i = \frac{1}{d(x, x_i)}$$

→ We can use all the training examples.

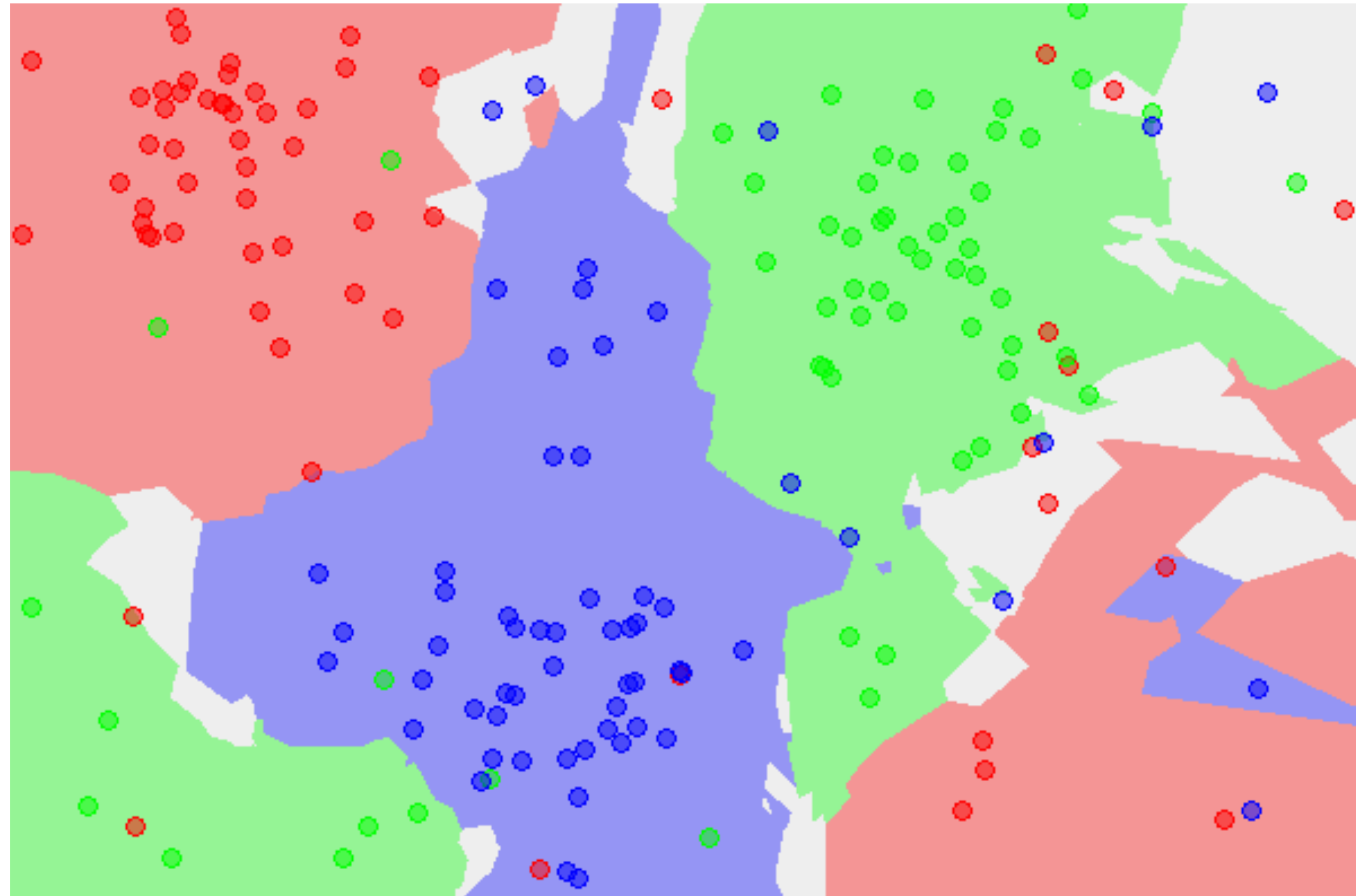
kNN Decision Boundary



[IR, fig 14.6](#)

1-NN: unions of cells of *Voronoi tessellation*

kNN Decision Boundary



[link](#)

5-NN example

Summary of kNN algorithm

- Decide k , feature weights, and similarity measure
- Given a test instance x
 - Calculate the distances between x and all the training data
 - Choose the k nearest neighbors
 - Let the neighbors vote

Pros/Cons of kNN algorithm

- Strengths:
 - Simplicity (conceptual)
 - Efficiency at training: no training
 - Handling multi-class
 - Stability and robustness: averaging k neighbors
 - Predication accuracy: when the training data is large
 - Complex decision boundaries
- Weakness:
 - Efficiency at testing time: need to calculate all distances
 - Better search algorithms: e.g., use k-d trees
 - Reduce the amount of training data used at the test time: e.g., Rocchio algorithm
 - Sensitivity to irrelevant or redundant features
 - Distance metrics unclear on non-numerical/binary values