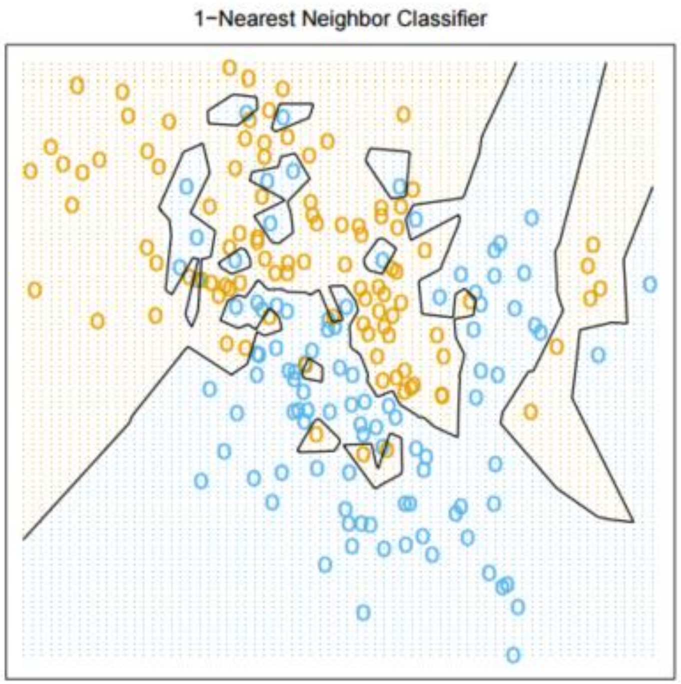
Nearest Neighbour Classifiers



Slides from:

Derek Hoiem
Friedman, Hastie and Tibshirani

CSC321: Intro to Machine Learning and Neural Networks, Winter 2016

Michael Guerzhoy

The Task

- Given a set of labelled examples (the *training set*), determine/predict the labels of a set of unlabelled examples (the *test set*)
  - Training set:

    Example 1: $(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots, x_m^{(1)})$ Label 1: $y^{(1)}$

    Example 2: $(x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, \dots, x_m^{(2)})$ Label 2: $y^{(2)}$

    ….

    Example N: $(x_1^{(N)}, x_2^{(N)}, x_3^{(N)}, \dots, x_m^{(N)})$ Label N: $y^{(N)}$
  - Test set:

    $(x_1^{(N+1)}, x_2^{(N+1)}, x_3^{(N+1)}, \dots, x_m^{(N+1)})$ Label: $y^{(N+1)} = ?$

    $(x_1^{(N+2)}, x_2^{(N+2)}, x_3^{(N+2)}, \dots, x_m^{(N+2)})$ Label: $y^{(N+2)} = ?$

Face Recognition Example

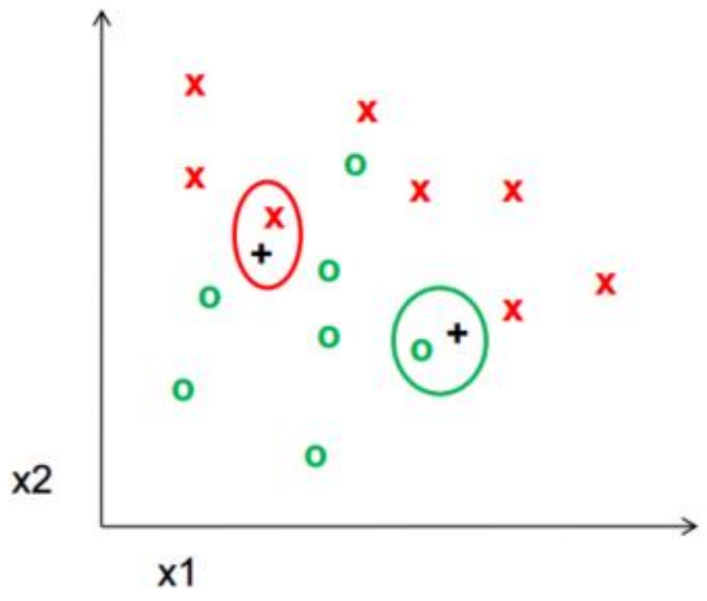
- Training set: photos of musicians with names ("labels")
- Test set: photos of musicians whose name we want to figure out
    - Note: generally, we *will* know the labels for the test set, but we pretend we don't. We can then predict the labels using our algorithm and compare the answers the algorithm gives to the correct answers to figure out the performance of our algorithm.
- An estimate for the performance of the algorithm on *new data*: the proportion of the examples in the test set that were correctly classified

Nearest Neighbour Classification

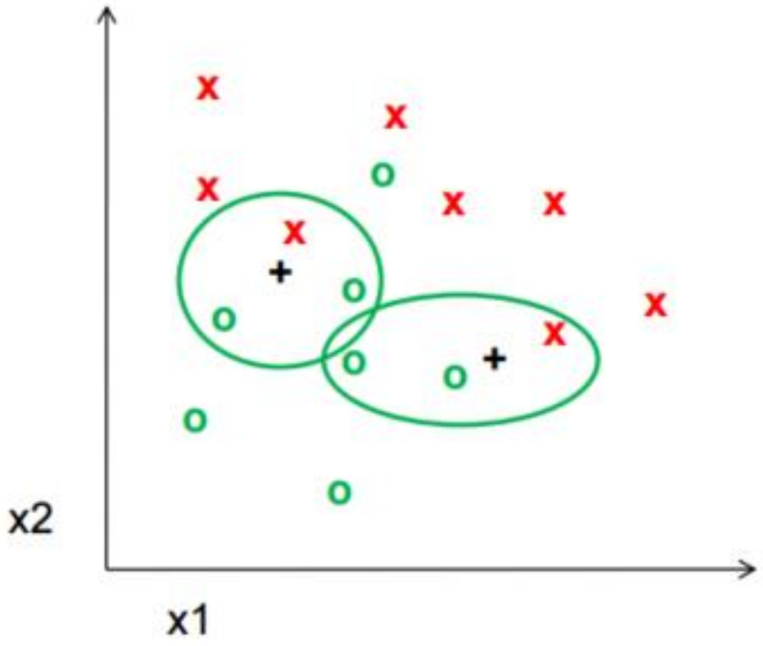
- Task: classify the test example
  x = $(x_1\ , x_2\ , x_3\ , \ldots , x_m)$
- Idea: find the *k* nearest neighbours of x in the training set, and output the majority label
- The distance between $x^{(1)}$ and $x^{(2)}$ could be
  - $|x^{(1)} - x^{(2)}| = \sqrt{\sum_i \left(x_i^{(1)} - x_i^{(2)}\right)^2}$ ("Euclidean/L2 distance")
  - $|x^{(1)} - x^{(2)}| = \max_{\mathrm{i}} \left|x_i^{(1)} - x_i^{(2)}\right|$ ("L-infinity distance")
  - …
  - By default, we use the Euclidean distance

1-nearest neighbour

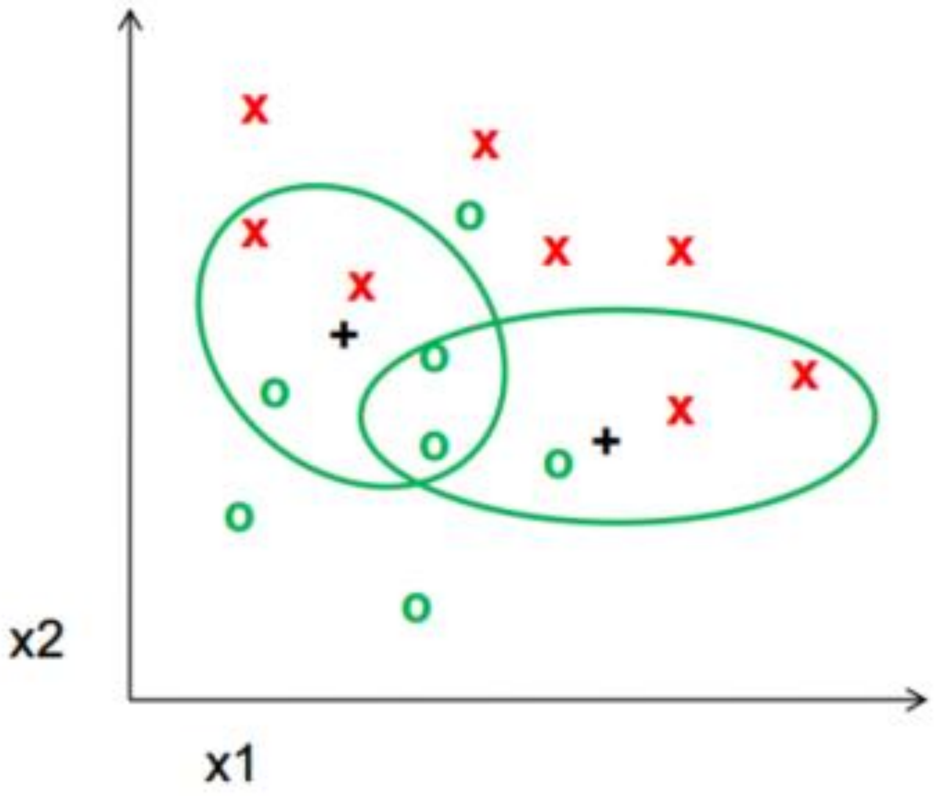
Task: classify the "+"
The labels for the training set are GREEN and RED
The examples are 2-dimensional
User L2/Euclidean distance



3-nearest neighbour



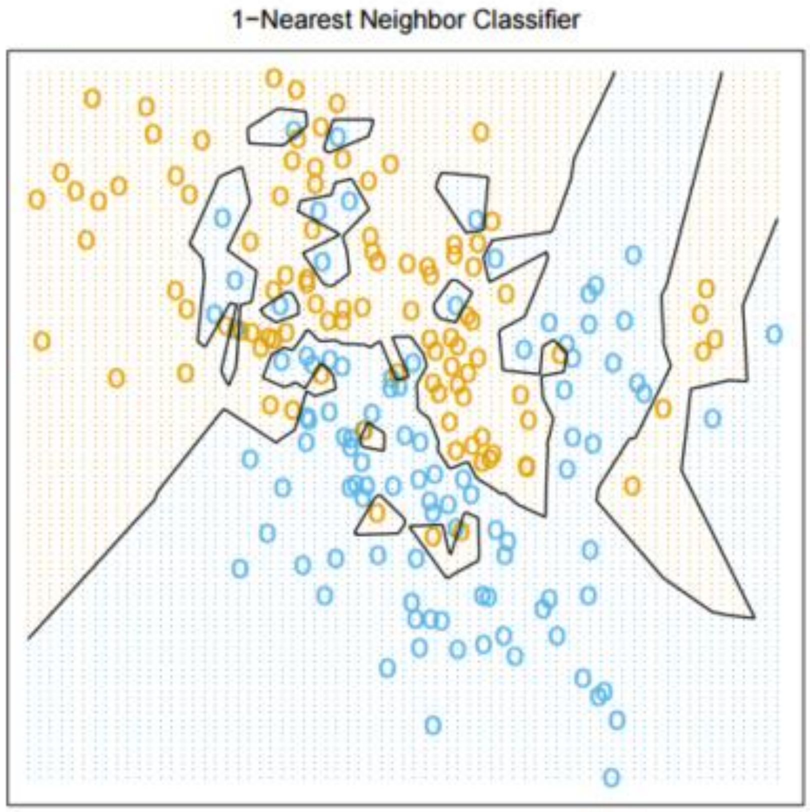
5-nearest neighbour

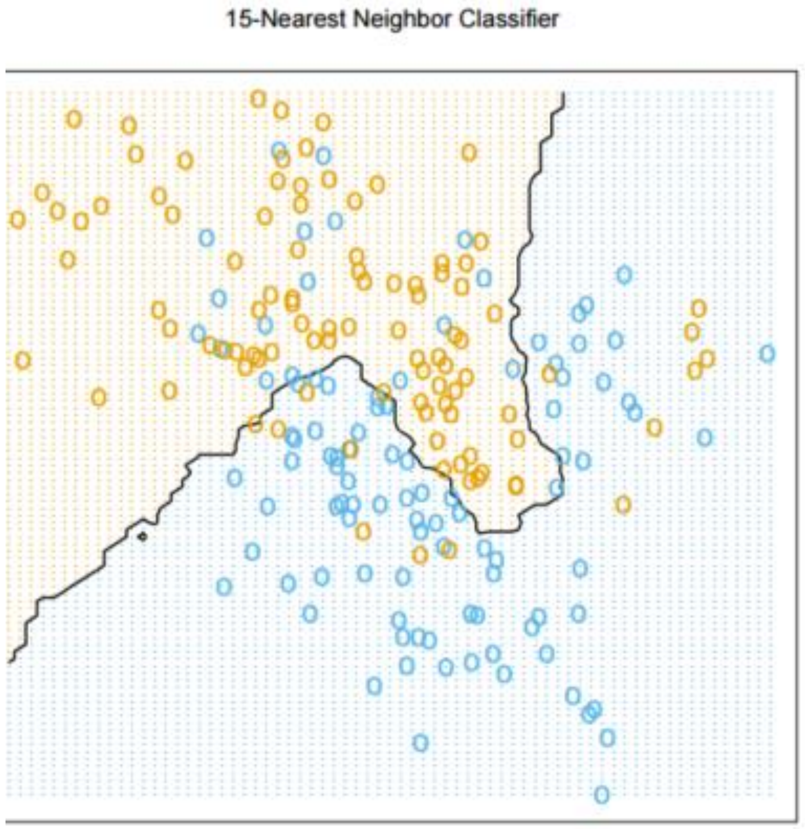


How do we determine K?

- Try different values, and see which works best on the test set?
  - Could do that, but then we are selecting the best K for our particular test set. This means that the performance on our test set is now an overestimate of how well we'd do no *new data*
- Solution: set aside a *validation set* (which is separate from both the training and the test set), and select the K for the best performance on the validation set, but report the results on the test set
  - Generally, the performance on the validation set will be better than on the test set
  - What about the performance on the *training set*?

What does the best K say about the data?





Large k: relatively simple boundary, no small "islands" in the data. Small changes in x do no generally change the label
Small k: a complex boundary between the labels. Small changes in x often change the labsl

Why not let K be very small?

- Great for the performance on the training set!
  - Perfect performance guaranteed for k = 1
- If the test data does not look exactly like the training data, the performance on the test data will be worse for k that is too small
  - The training data could be noisy (e.g., in the orange region, data points are sometimes blue with probability 5%, randomly)
  - This is an example of *overfitting* – building a classifier that works well on the training set, but does not generalize well to the test set

Why not let K be very large?