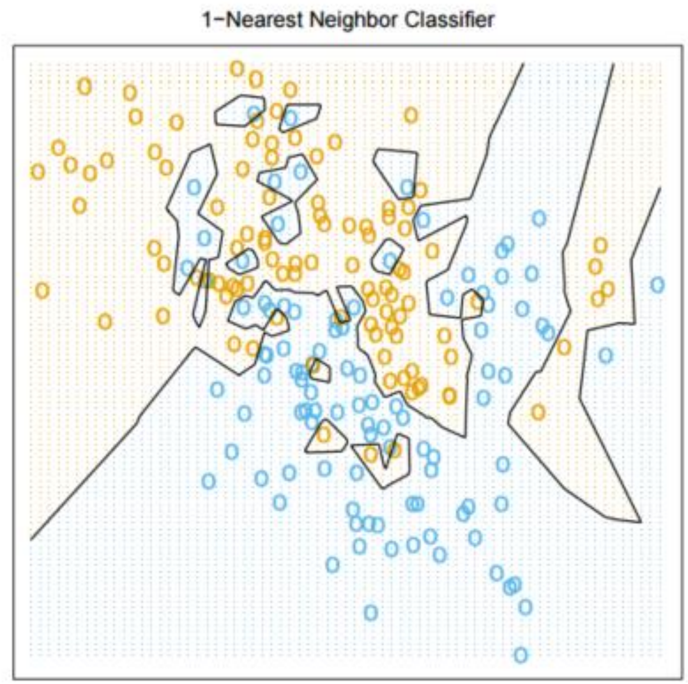
Nearest Neighbour Classifiers



Slides from:
Derek Hoiem
Friedman, Hastie and Tibshirani

CSC411: Machine Learning and Data Mining, Winter 2018

Michael Guerzhoy and Lisa Zhang

The Task: Supervised Learning

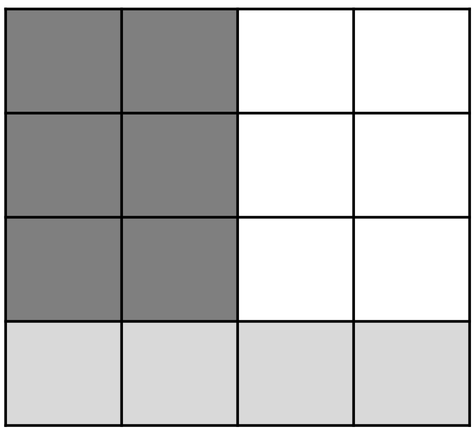
- Given a set of labelled examples (the *training set*), determine/predict the labels of a set of unlabelled examples (the *test set*)
  - Training set:
    - Train Example 1: $(x_1^{(1)}, x_2^{(1)}, \dots., x_m^{(1)})$ Label: $y^{(1)}$
    - Train Example 2: $(x_1^{(2)}, x_2^{(2)}, \dots., x_m^{(2)})$ Label: $y^{(2)}$
    - ...
    - Train Example N: $(x_1^{(N)}, x_2^{(N)}, \dots., x_m^{(N)})$ Label: $y^{(N)}$
  - Test set:
    - Test Example 1: $(x_1^{(N+1)}, x_2^{(N+1)}, \dots., x_m^{(N+1)})$ Label: $y^{(N+1)}$
    - Test Example 2: $(x_1^{(N+2)}, x_2^{(N+2)}, \dots., x_m^{(N+2)})$ Label: $y^{(N+2)}$
    - ...
    - Test Example K: $(x_1^{(N+K)}, x_2^{(N+K)}, \dots., x_m^{(N+K)})$ Label: $y^{(N+K)}$

Task: Face Recognition

- 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

What Justin Bieber Looks like to a Computer

Images ↔ Vectors



60	60	255	255
60	60	255	255
60	60	255	255
128	128	128	128

60
60
255
255
60
60
255
255
60
60
255
255
128
128
128
128

The Face Recognition Task

- Training set:
  - $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\}$
    - $x^{(i)}$ is a k-dimensional vector consisting of the intensities of all the pixels in in the i-th photo (20×20 photo $\rightarrow x^{(i)}$ is 400-dimensional)
    - $y^{(i)}$ is the *label* (i.e., name)
- Test phase:
  - We have an input vector x, and want to assign a label y to it
    - Whose photo is it?

Face Recognition using 1-Nearest Neighbors (1NN)

- Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\}$
- Input: x
- 1-Nearest Neighbor algorithm:
    - Find the training photo/vector $x^{(i)}$ that's as "close" as possible to x, and output the label $y^{(i)}$

Input x

Closest training image to the input x

Output: Paul

Are the two images a and b close?

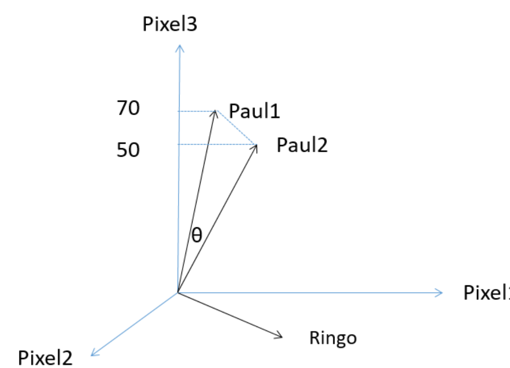
- Key idea: think of the images as *vectors*
  - Reminder: to turn an image into a vector, simply "flatten" all the pixels into a 1D vector
- Is the distance between the endpoints of vectors a and b small?

$$|a - b| = \sqrt{\sum_i \ (a_i - b_i)^2} \text{ small}$$

- Is the cosine of the angle between the vectors a and b large?

$$\cos \theta_{ab} = \frac{a \cdot b}{|a||b|} = \frac{\sum_i \ a_i b_i}{\sqrt{\sum_i \ a_i^2}\sqrt{\sum_i \ b_i^2}} \text{ large}$$

By the law of cosines

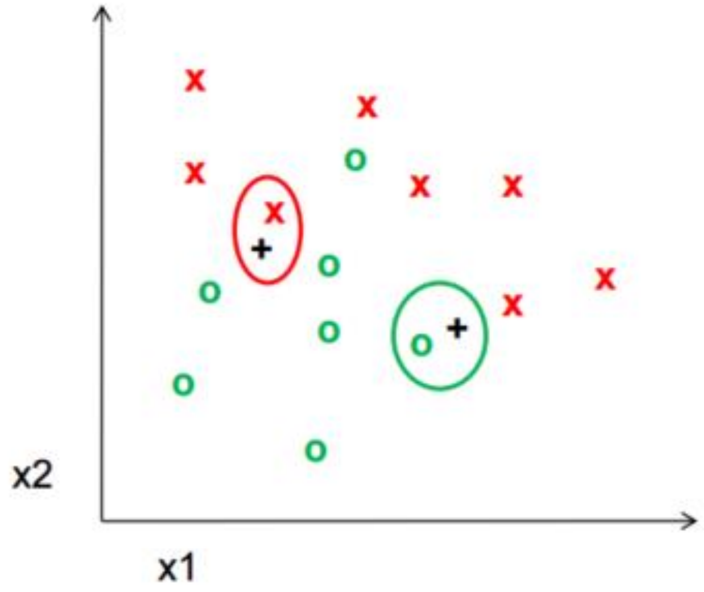


k-Nearest Neighbour Classification

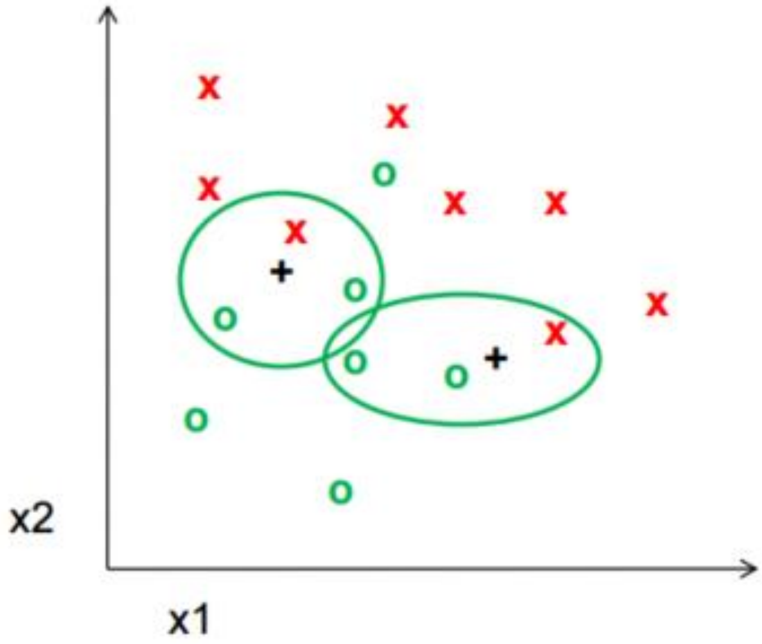
- For an example x
  - Find the k closest examples (neighbours) to x in the training set
  - Output the plurality label for the k closest examples
- Can use various distance functions:
  - Euclidian (L2): $\text{dist}(a, b) = \sqrt{\sum_i (a_i - b_i)^2}$ (default)
  - L-infinity: $\text{dist}(\text{a}, \text{b}) = \max_i |a_i - b_i|$
  - L-zero: $\text{dist}(\text{a}, \text{b}) = \#\{a_i \neq b_i\}$
  - Negative cosine: $\text{dist}(\text{a}, \text{b}) = -\frac{a \cdot b}{|a||b|}$

1-nearest neighbour

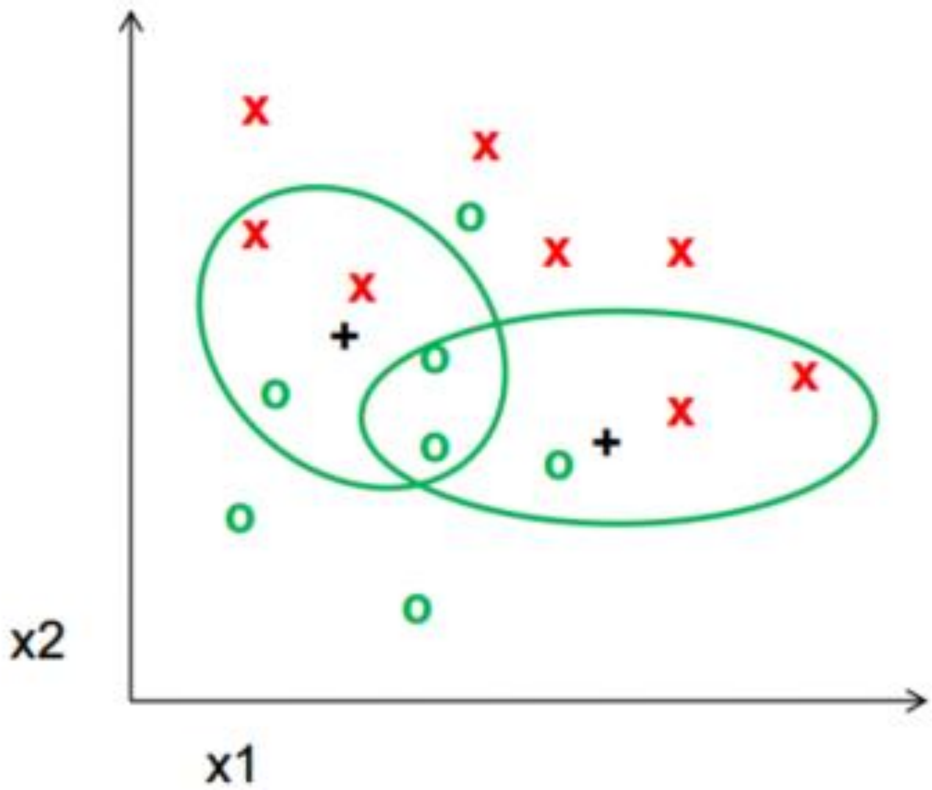
Task: classify the test set of "+"
The labels for the training set are GREEN and RED
The examples are 2-dimensional
Use L2/Euclidean distance



3-nearest neighbour



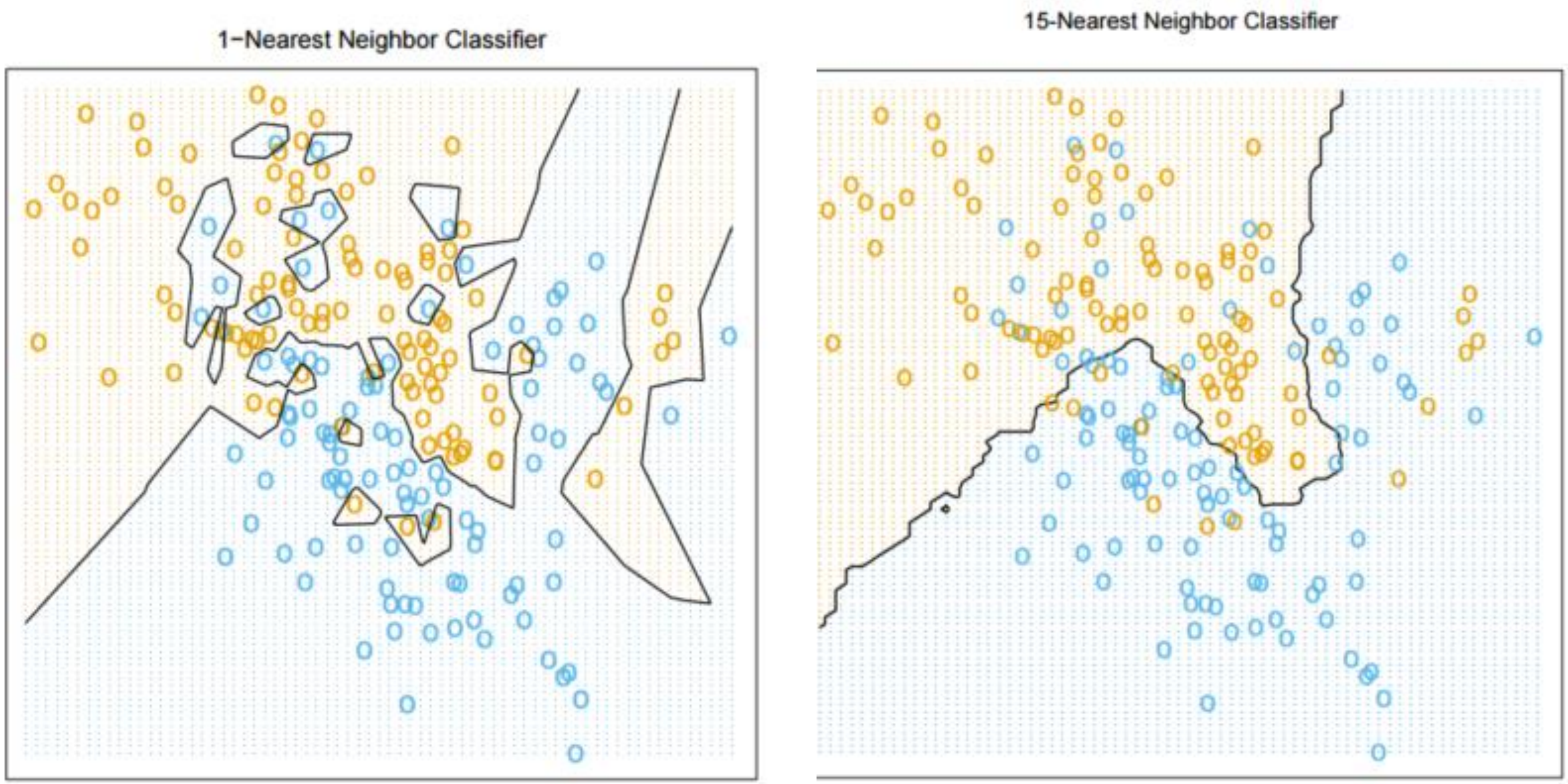
5-nearest neighbour



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 on *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 labels

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?

Distance Functions

- For images, why might the cosine distance make sense?
- For images, why might the Euclidean distance make sense?