Machine Learning for NLP
Lecture 5 part 1: Example-based learning

UNIVERSITY OF
GOTHENBURG

Richard Johansson

October 1, 2015

learning by observing examples

- humans often classify new objects by comparing them to examples they have seen before
- when we use this intuition in machine learning algorithms, we have something called **example-based learning** (or **memory-based**...)

nearest neighbor classifier

- "learning" algorithm: remember all examples in the training set
- prediction algorithm for a new instance $\boldsymbol{x}$:
  1. find the instance $\boldsymbol{x}_t$ that is "most similar" to $\boldsymbol{x}$
  2. return the corresponding training set label y_t
- this is called the **nearest neighbor classifier**
- unlike e.g. linear classifiers and decision trees, there isn't obviously any abstraction

similarity

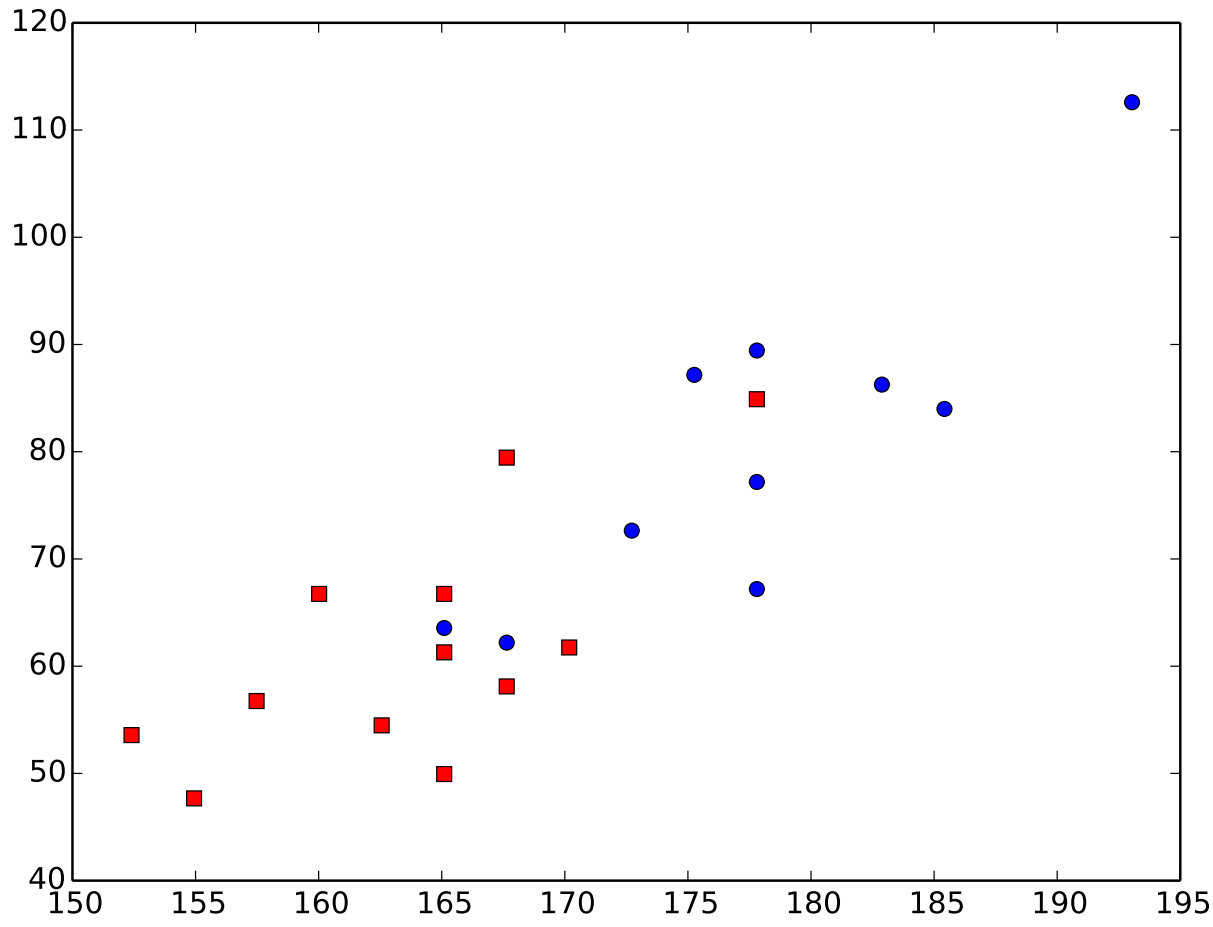
- in the nearest neighbor classification algorithm, we used a notion of "most similar"
- how should that be interpreted?
- we need to define a **similarity function**
  - or conversely, a **distance function**

geometric distance

- convert the instances to numerical vectors
  - in scikit-learn, we'd use a `DictVectorizer`, `CountVectorizer`, or `TfidfVectorizer`
- then use some distance measure defined for the vectors
- most common and intuitive choice: the Euclidean distance
  - i.e. corresponding to our intuitive notion of distance in 2 or 3 dimensions

$$d(V, W)^2 = \|V - W\|^2 = (v_1 - w_1)^2 + \ldots + (v_n - w_n)^2$$

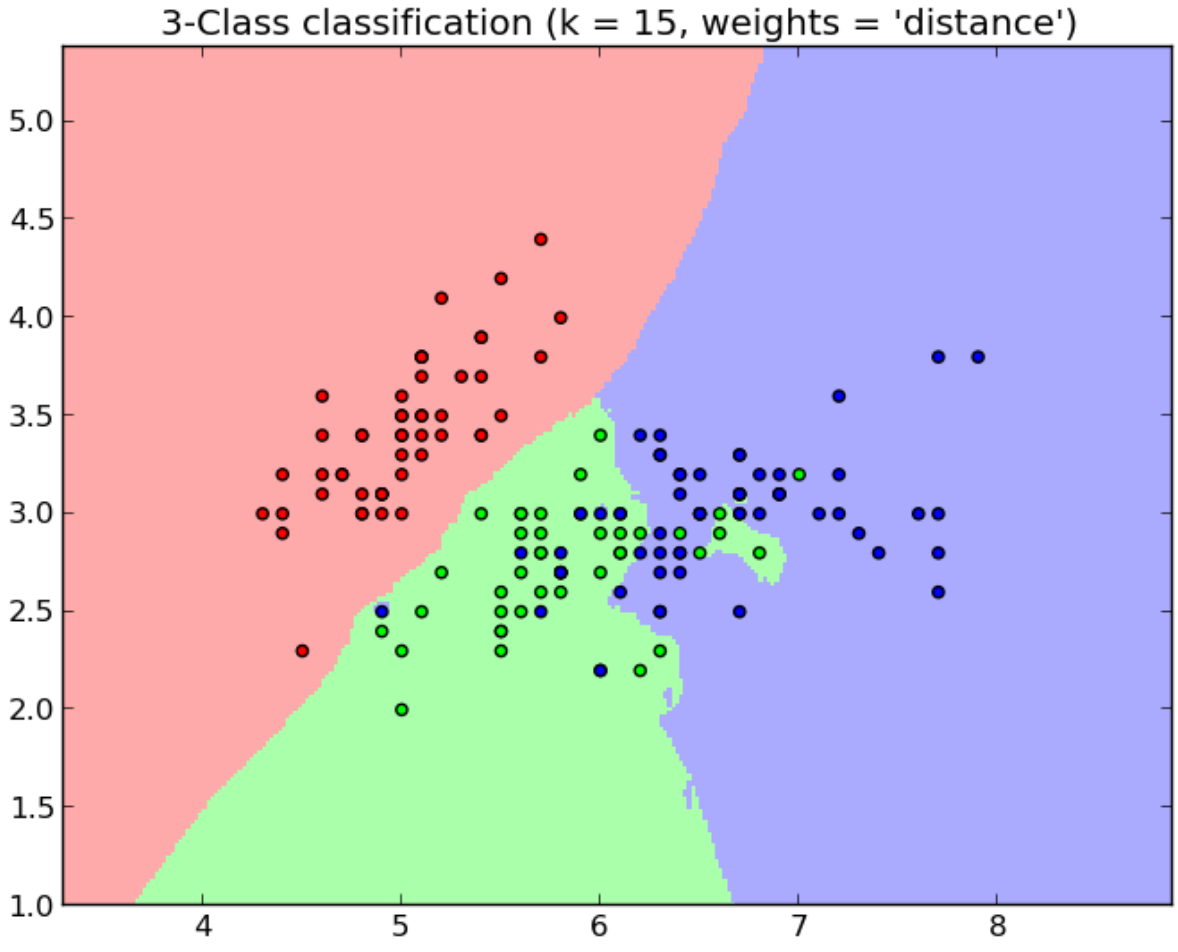
example: guessing the gender



using more neighbors

- too unstable to base our decision on just one example?
- the *k*-**nearest neigbor classifier**: select the k nearest neighbors, then
  - use a voting scheme, where we output the label favored by most of the k neighbors,
  - or a distance-weighted voting scheme

example decision boundary



implementation

- how do we find the nearest neighbor if the training set has a million instances?
- a careful design of algorithms and data structures is needed
- examples:
  - KD-trees and ball trees
  - locality sensitive hashing (LSH)
- `http://scikit-learn.org/stable/modules/neighbors.html`
- training a nearest neigbor classifier takes almost no time, but it can be slow at classification time

experiment

- I tried out scikit-learn's `KNeighborsClassifier` using the sentiment polarity dataset we've previously used
  - `http://scikit-learn.org/stable/modules/neighbors.html`
  - example code:

```
from sklearn import neighbors
cl = neighbors.KNeighborsClassifier(256, 'distance')
```

- best results with $k = 256$, distance weighting of the neighbors, and Euclidean distance
- initially, very poor results – this could be fixed by using a `TfidfVectorizer` instead of a `CountVectorizer`
  - this shows how sensitive these algorithms are to the selection of a similarity function

the effect of k