Our first learning algorithm

- How would you classify the next example?

kNN Algorithm (simple form)

- At "training time" do nothing.
- Store examples.
- When given new example:
  - find k nearest neighbors
  - Predict L= majority vote of their labels

kNN Algorithm

- Theoretical basis + intuition:
  - "in the limit", when the dataset is dense, this should pick up "all important regions"
  - Very flexible classifier: no prior commitment to the shape, density, or distribution of regions

kNN: problems and extensions

- In some cases we have "noisy" labels in training data, or otherwise the label map is not smooth.
  - How can we address this?

kNN: problems and extensions

- Expensive test time/application: because for every new example we must scan the entire dataset to find the neighbors.
  - In many cases a Linear Time Scan is too expensive.
  - How can we address this?
kNN: problems and extensions

- $k$ is a free parameter of the algorithm
- And different values of $k$ are suitable to difference datasets

- How can we choose $k$ automatically?

- (for large $k$ and/or non uniformly sampled regions) Some neighbors are significantly closer than others.

- Weighted kNN weights the prediction of each neighbor by some function of its distance (e.g., $1/distance$).

- Algorithm completely dependent on the distance metric and representation

  - E.g., Euclidean distance:
    - Different features may have different scale
    - Treats all dimensions equally
    - Sensitive to high dimension/ irrelevant features (why?)

- How can we address these issues?

  - Linear scaling
  - Z-normalization
  - The Relief Algorithm

- So far we know how to predict one of a small number of categories - i.e. we are solving the classification problem.

  Can we adjust the algorithm to predict numerical real-valued labels? (i.e., solve the regression problem)

- Simple basic algorithm
- Has theoretical guarantees

- Adjustment of the basic scheme can make it robust and widely applicable

- Performs surprisingly well in many cases