K-Nearest-Neighbor (KNN) Classifier

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

Math behind ...

- Estimate predictive probability by counting labels in neighboring area
  \[ p(y|x) = \frac{1}{K} \sum_{x_i \in N(x)} y_i \]
- What is the effect of K
  - Large K: more accurate average, but need to cover larger area
  - Small K: the opposite

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: problem 1

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

KNN: problem 2

- Expensive test time/application
  - for every new example we must scan the entire dataset to find the neighbors.
  - How can we address this?
    - Some form of pruning in the search process.
    - kd-trees
KNN: problem 3

- 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?

- Normalization / standardization
- Metric learning (cover later)

KNN Recap

- Simple basic algorithm
- Has theoretical guarantees

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

- Performs surprisingly well in many cases