Instance Based Learning
Nearest Neighbors

Our first learning algorithm

- How would you classify the next example?

\[ \text{comp135} \]

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

\[ \text{comp135} \]

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

\[ \text{comp135} \]

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?

\[ \text{comp135} \]
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?

kNN: problems and extensions

• (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).

• How can we address these issues?

Normalizing Feature Range

• Linear scaling into [0,1]

\[ x \leftarrow \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \]

• Z-normalization: scale to have mean 0 and std 1

\[ x \leftarrow \frac{x - \mu_x}{\sigma_x} \]

Relief Algorithm

• Pick instance + nearest hit (same label) + nearest miss (different label)

• Update weights based on distances and whether hit is closer/farther than miss

• This is done on each feature separately

\[ w_i \leftarrow w_i - |x_i^{\text{hit}} - x_i| + |x_i^{\text{miss}} - x_i| \]

• Distance is modified to

\[ \|x - y\|_c^2 = \sum w_i (x_i - y_i)^2 \]
### kNN: problems and extensions

- 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)*

### 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