STA 314: Statistical Methods for Machine Learning I

Lecture - k-nearest neighbour (classification)

Xin Bing

Department of Statistical Sciences
University of Toronto

A Classical Local Approach: Nearest Neighbors

- Suppose we're given a new feature vector $\mathbf{x} \in \mathbb{R}^p$ we consider classification.
- The idea: find the nearest feature vector to $\mathbf{x}$ in the training set and use its label.
- Can formalize "nearest" in terms of the Euclidean distance

$$||\mathbf{x}_i - \mathbf{x}_{i'}||_2 = \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2}$$

Algorithm (1-NN):

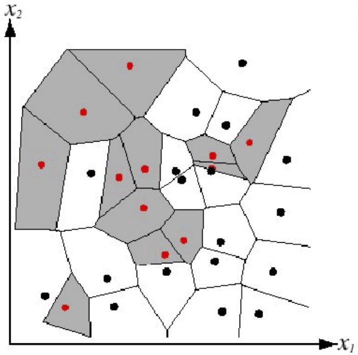
1. Find example $(\mathbf{x}_*, y_*)$ (from the stored training data) closest to $\mathbf{x}$. That is:

$$\mathbf{x}_* = \underset{\mathbf{x}_i \in \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}}{\operatorname{argmin}} \ \mathrm{distance}(\mathbf{x}_i, \mathbf{x})$$

2. Output y_* as the label

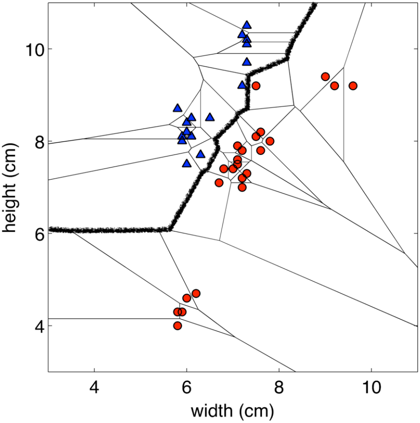
Nearest Neighbors: Decision Boundaries

We can visualize the behavior in the classification setting using a Voronoi diagram.

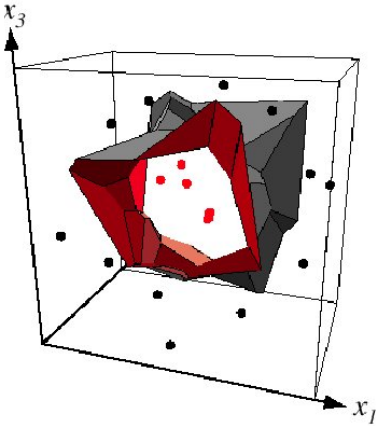


Nearest Neighbors: Decision Boundaries

Decision boundary: the boundary between regions of the feature space assigned to different categories.



Nearest Neighbors: Decision Boundaries



Example: 2D decision boundary

k-Nearest Neighbors

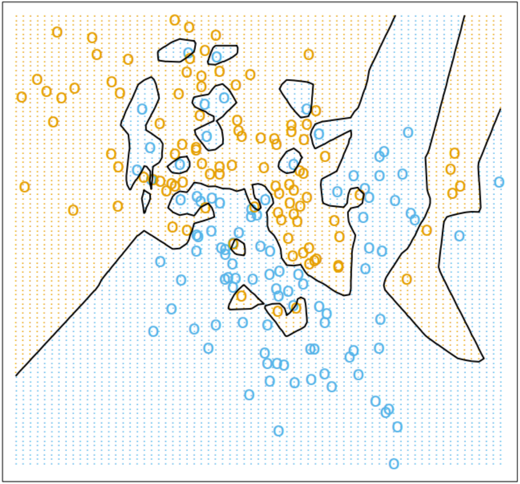
Algorithm (k-NN):

1. Find k data points $(\mathbf{x}_{(1)}, y_{(1)}), \ldots, (\mathbf{x}_{(k)}, y_{(k)})$ closest to the test instance $\mathbf{x}$
2. Classification output is majority class

$$\arg\max_{y \in C} \sum_{i=1}^{k} 1\left\{y = y_{(i)}\right\}.$$

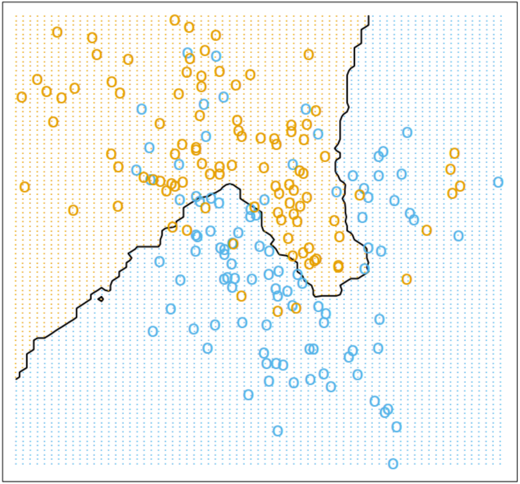
k-NN

k=1



k-NN

k=15



k-NN

Tradeoffs in choosing k

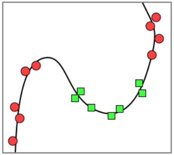
- Small k
  - More flexible decision boundary but high variance
  - Overfitting: sensitive to random noise in the training data.
- Large k
  - Less flexible decision boundary and smaller variance
  - Underfitting: fail to capture the true decision boundary.
- Balancing k
  - Optimal choice of k depends on number of data points n.
  - Nice theoretical properties if

$$k \to \infty, \qquad \text{and} \qquad \frac{k}{n} \to 0 \quad \text{(ESL 2.4).}$$

  - Rule of thumb: choose k from $[1, \sqrt{n}]$ via cross-validation!

Pitfalls: The Curse of Dimensionality

- k-NN suffers the curse of dimensionality!
  - In high dimensions, "most" points are approximately the same distance because they are far away from each other.
- Saving grace: some datasets (e.g. images) may have low intrinsic dimension, i.e. lie on or near a low-dimensional manifold.



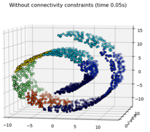
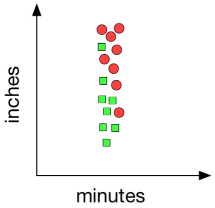


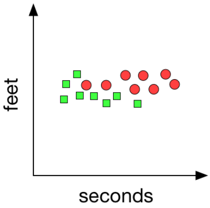
Image credit: `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_swiss_roll.html`

The neighborhood structure depends on the intrinsic dimension.

Pitfalls: Normalization

- Nearest neighbors can be sensitive to the ranges of different features.
- Often, the units are arbitrary:





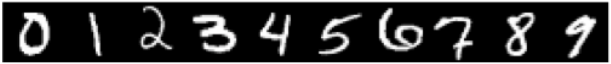
- Simple fix: standardize each dimension to be zero mean and unit variance.
- Caution: depending on the problem, the scale might be important!

Pitfalls: Computational Cost

- Computational cost for training: 0
- Computational cost for classifying test data, per data point (un-modified algorithm)
  - Calculuate p-dimensional Euclidean distances with n data points: $\mathcal{O}(np)$
  - Sort the distances: $\mathcal{O}(n \log n)$
- This must be done for *each* test data point, which is very expensive by the standards of a learning algorithm!
- Need to store the entire dataset in memory!
- Tons of work has gone into algorithms and data structures for efficient nearest neighbors with high dimensions and/or large datasets.

Example: Digit Classification

- Decent performance when lots of data

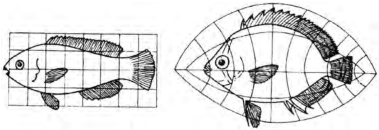


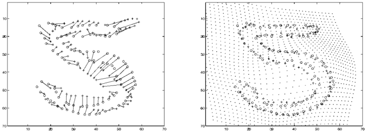
- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

Example: Digit Classification

- Changing the distance measure can really improve k-NN.
- Example: shape contexts for object recognition. In order to achieve invariance to image transformations, they tried to warp one image to match the other image.
  - Distance measure: average distance between corresponding points on *warped* images
- Achieved 0.63% error on MNIST, compared with 3% for Euclidean KNN.
- Competitive with the state of the art at the time, but required careful engineering.





[Belongie, Malik, and Puzicha, 2002. Shape matching and object recognition using shape contexts.]