Introduction to Machine Learning

Reading

- Chapters 2.1-2.3
  - Hastie, Trevor, et al. The elements of statistical learning: data mining, inference, and prediction. Vol. 2. New York: springer, 2009.
  - Available online: https://hastie.su.domains/Papers/ESLII.pdf
- Chapters 2.1
  - James, Gareth, et al. An introduction to statistical learning. Vol. 112. New York: springer, 2013.
  - Available online: https://www.statlearning.com/
- Supervised learning from a statistical point of view

Supervised Learning Overview

Rensselaer

- For the first few weeks of the course, we will assume we are given a training set of N labeled examples: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$
- The $\boldsymbol{x}_i$ variables are called features or inputs
  - Also called predictors or independent variables in the statistical community
- The y_i variables are called labels or outputs
  - Also called responses or dependent variables in the statistical community
  - Can be discrete (e.g., $y_i \in \{cat, dog\}$) as in classification
    - Typically represented numerically, e.g., $cat = 0, dog = 1$
  - Can be continuous (e.g., $y_i \in [0,1]$) as in regression

High-level Supervised Learning Task

Rensselaer

- We are given a labeled training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$
  - E.g., the features $\boldsymbol{x}_i$ are images (i.e., pixel values) and the labels are digits (from 0 to 9)
- We are assuming that the features contain sufficient information in order to determine the label
  - One could also consider hidden state problems, where some features are not observed, but out of scope for this course
- A (naïve) goal is to find a function f such that

$$y_i = f(\boldsymbol{x}_i)$$

- What is an example function f that satisfies the above?
  - A table of rules: if you see $\boldsymbol{x}_i$, then output y_i
  - What is wrong with that function?

Statistical Learning

Rensselaer

- We don't want to learn a function that only works on the training data
  - The point of learning is to *generalize* to new data
- How do we think about new data?
- Each data point is a realization of random variables $\boldsymbol{X}, Y$
- The variables $\boldsymbol{X}, Y$ follow an unknown distribution $\mathcal{D}$
  - written $(\boldsymbol{X}, Y) \sim \mathcal{D}$
- Each example $(\boldsymbol{x}_i, y_i)$ has some unknown probability under $\mathcal{D}$
- One way to approach the problem is to try to learn $\mathcal{D}$
  - Then for new $\boldsymbol{x}_i$, we output the y_i that has highest probability according to $\mathcal{D}$
  - However, learning an arbitrary unknown $\mathcal{D}$ is hard

Statistical Learning, cont'd

Rensselaer

- The standard approach to the learning problem is to pick a class of functions $\mathcal{F}$ and choose the *best* $f \in \mathcal{F}$
  - We'll talk later about ways of defining *best*
- Examples of $\mathcal{F}$ are all linear functions, all polynomials, all neural networks, etc.
- Ideally, $\mathcal{F}$ captures underlying relationship between $\boldsymbol{X}$ and Y
  - If that is the case, the *best* f will satisfy

$$Y = f(\boldsymbol{X})$$

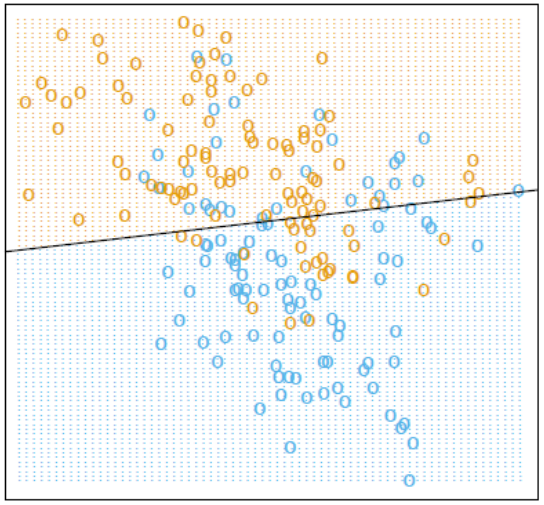
    - for all (or most) pairs $(\boldsymbol{X}, Y) \sim \mathcal{D}$
  - The *best* f is selected based on the training set

Statistical Learning, cont'd

Rensselaer

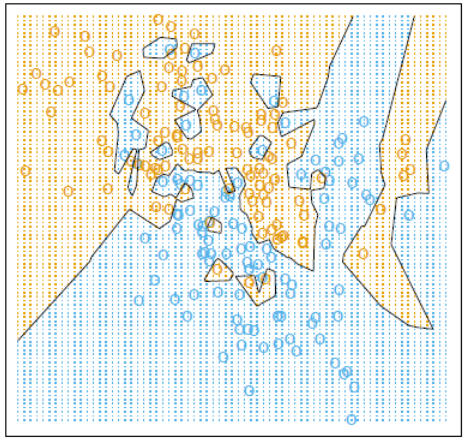
- This statistical formulation is nice, but why is it any good?
  - There is a lot of great theory for the above setting
- If data points are IID (independent and identically distributed)
  - For many classes $\mathcal{F}$, one can show that performing well on the training set implies good performance on unseen data
  - One can show that performing well on a test set (unseen during training) implies good performance on new data
- We will explore some of these notions in this class

A Simple, but Effective, Classifier: Nearest Neighbor

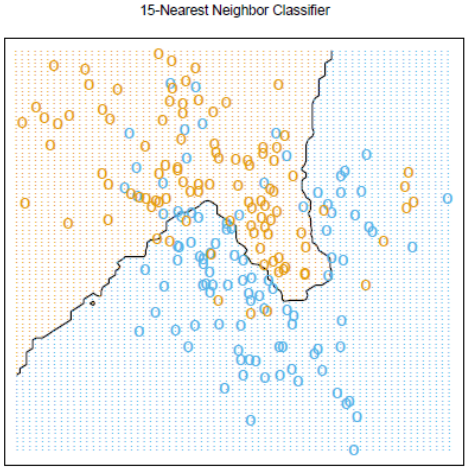


- Suppose we are given a 2D training dataset with two classes as shown here
  - Orange and blue
  - (Ignore the line for now)
- If we see a new point, what's an easy way to choose a label?
  - Nearest neighbor!
  - Choose the label of the nearest point from the training set
- What is a problem with this strategy?

An Improved Version: K-Nearest Neighbors



- 1-Nearest neighbor similar to a table classifier
  - Decision space looks very fragmented
  - Overadapting to training data is overfitting
  - Unlikely to generalize well in some parts
- What's a simple way to improve?
- K-nearest neighbors!
- Instead of choosing the label of nearest neighbor, pick the majority label of the K-nearest neighbors
  - Much smoother decision boundary
  - More robust to noisy data



Formalizing KNN

- What is the "closest neighbor"?
  - Need a distance metric!
- Many distance metrics have been used in ML
- Most standard is Euclidean distance, or L_2 distance, or 2-norm
- The Euclidean distance between vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$ is

$$\left|\left|\boldsymbol{x}-\boldsymbol{y}\right|\right|_2 = \sqrt{(x_1-y_1)^2+\cdots+\left(x_p-y_p\right)^2}$$
$$= \sqrt{(\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y})}$$

Formalizing KNN, cont'd

Rensselaer

- Let $\mathcal{N}_K(\boldsymbol{x})$ be the indices of K closest neighbors for a point $\boldsymbol{x}$
- Assume the labels are $y_i = 0$ for BLUE and $y_i = 1$ for ORANGE
- Then the following is the number of ORANGE votes by $\mathcal{N}_K$:

$$Or(\boldsymbol{x}) = \sum_{i \in \mathcal{N}_K(\boldsymbol{x})} y_i$$

- Finally, classify $\boldsymbol{x}$ as Orange if $Or(\boldsymbol{x}) > K/2$:

$$f(\boldsymbol{x}) = 1 \quad if \;\; Or(\boldsymbol{x}) > K/2$$
$$f(\boldsymbol{x}) = 0 \quad if \;\; Or(\boldsymbol{x}) < K/2$$

  - Pick K to be odd to avoid ties