Nearest neighbor classification

CSE 250B

Nearest neighbor classification

Given a labeled training set $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$.

Example: the MNIST data set of handwritten digits.

To classify a new instance x:

- Find its nearest neighbor amongst the $x^{(i)}$
- Return $y^{(i)}$

The data space

We need to choose a distance function.

Each image is 28×28 grayscale.
One option: Treat images as 784-dimensional vectors, and use Euclidean (ℓ_2) distance:

$$\|x - x'\| = \sqrt{\sum_{i=1}^{784} (x_i - x_i')^2}.$$

Summary:

- Data space $\mathcal{X} = \mathbb{R}^{784}$ with ℓ_2 distance
- Label space $\mathcal{Y} = \{0, 1, \ldots, 9\}$

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%.**
- Test error of nearest neighbor: **3.09%.**

Examples of errors:

Ideas for improvement: (1) k-NN (2) better distance function.

K-nearest neighbor classification

Classify a point using the labels of its k nearest neighbors among the training points.

MNIST:

k	1	3	5	7	9	11
Test error (%)	3.09	2.94	3.13	3.10	3.43	3.34

How to choose k in general?
Let $S \in \mathcal{Z}^n$ be the training set, where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ is the space of *labeled* points. Let $\Gamma_k(S, x)$ be the prediction made on x using its k-NN in S.

1. Hold-out set. Choose a subset $V \subset S$ as a *validation set*.

$$\arg\min_k \sum_{(x,y)\in V} \mathbf{1}(\Gamma_k(S \setminus V, x) \neq y)$$

2. Leave-one-out cross-validation.

$$\arg\min_k \sum_{(x,y)\in S} \mathbf{1}(\Gamma_k(S \setminus \{(x,y)\}, x) \neq y)$$

Better distance functions

Let x be an image. Consider an image x' that is just like x, but is either:

- shifted one pixel to the right, or
- rotated slightly.

Then $\|x - x'\|$ could easily be quite large.

It makes sense to choose distance measures that are invariant under:

- Small translations and rotations. e.g. *tangent distance*.
- A broader family of natural deformations. e.g. *shape context*.

Test error rates:

ℓ_2	tangent distance	shape context
3.09	1.10	0.63

Are there families of distance functions that are often useful?

ℓ_p norms

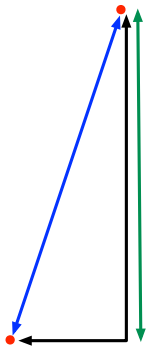
How can we measure the length of a vector in $\mathbb{R}^m$?
Usual choice is the *Euclidean norm*:

$$\|x\|_2 = \sqrt{\sum_{i=1}^m x_i^2}.$$

Generalization: For $p \geq 1$, the ℓ_p norm is

$$\|x\|_p = \left(\sum_{i=1}^m |x_i|^p\right)^{1/p}$$

- $p = 2$: Euclidean norm
- ℓ_1 norm: $\|x\|_1 = \sum_{i=1}^m |x_i|$
- ℓ_∞ norm: $\|x\|_\infty = \max_i |x_i|$



Metric spaces

A more general notion is a *metric space*.

Let $\mathcal{X}$ be the space in which data lie. A distance function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a *metric* if it satisfies these properties:

- $d(x, y) \geq 0$ (nonnegativity)
- $d(x, y) = 0$ if and only if $x = y$
- $d(x, y) = d(y, x)$ (symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

For instance:

- $\mathcal{X} = \mathbb{R}^m$ and $d(x, y) = \|x - y\|_p$
- $\mathcal{X} = \{\text{strings over some alphabet}\}$ and $d = \text{edit distance}$.

Later in the course: methods for *learning* suitable distance measures.

Statistical learning theory

Model of reality: there is an (unknown) underlying distribution, call it P, from which pairs (x, y) are generated.

- Training points (x, y) come from this distribution.
- Future test points will also come from this distribution.

We want a classifier

$$f : \mathcal{X} \to \mathcal{Y}$$

which will do well on future data: in other words, do well on P. But we don't know P, so we treat the training data as a proxy for it.

The underlying distribution

Reality $\equiv$ the underlying distribution on pairs (x, y)

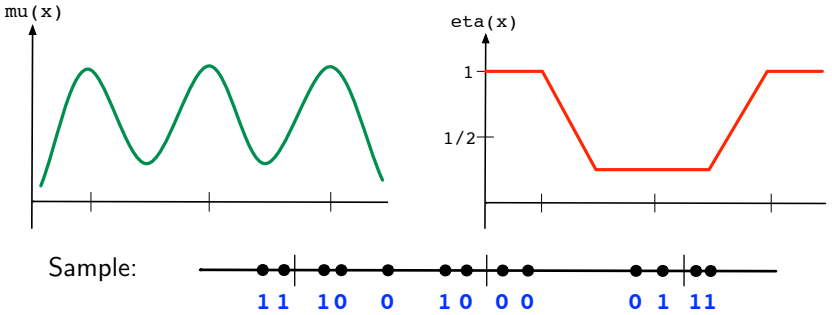
Can factor this distribution into two parts:

- A distribution on x. Call this μ.
- A distribution over labels y given x. In the binary case ($\mathcal{Y} = \{0, 1\}$) this can be specified as $\eta(x) = \Pr(Y = 1|x)$.

For instance, distribution of patients in a community:

x = age

$y = 1$ if visited doctor in the last year



Statistical learning theory, cont'd

Let's look at the binary case, $\mathcal{Y} = \{0, 1\}$.

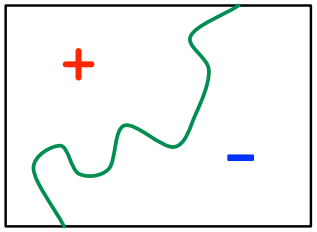
- There is an (unknown) underlying probability distribution on $\mathcal{X}$ from which *all* points are generated. Call this distribution μ.
- The label of any point x can, in general, be *stochastic*. It is a coin flip with bias $\eta(x) = \Pr(Y = 1|X = x)$.
- A classifier is a rule $h : \mathcal{X} \to \{0, 1\}$. Its misclassification rate, or *risk*, is $R(h) = \Pr(h(X) \neq Y)$.

The Bayes-optimal classifier

$$h^*(x) = \begin{cases} 1 & \text{if } \eta(x) > 1/2 \\ 0 & \text{otherwise} \end{cases}$$

has minimum risk,

$$R^* = R(h^*) = \mathbb{E}_X \min(\eta(X), 1 - \eta(X)).$$



Statistical theory of nearest neighbor

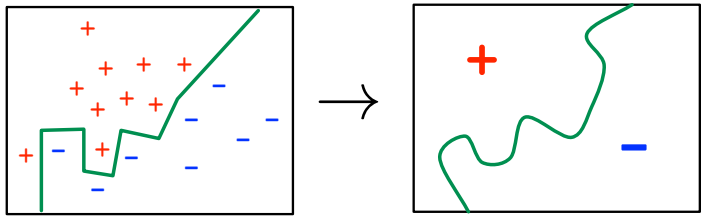
Let h_n be a classifier based on n training points from the underlying distribution.

- Its risk is $R(h_n) = \Pr(h_n(X) \neq Y)$.
- We say it is **consistent** if, as n grows to infinity, $R(h_n) \to R^*$.

Consistency of NN (Fix and Hodges, 1951; Cover and Hart, 1967):

- Suppose $\eta(x) = \Pr(Y = 1|X = x)$ is continuous in x.
- Set k to a growing function of n, with (1) $k \to \infty$ and (2) $k/n \to 0$, as $n \to \infty$.

Then k-NN is consistent.



Statistical theory of 1-NN

1-NN is not consistent.

E.g. $\mathcal{X} = \mathbb{R}$ and $\eta(x) \equiv 1/4$. Every label is a coin flip with bias $1/4$.

- Bayes optimal classifier: always predict 0. Risk $R^* = 1/4$.
- 1-NN risk: what is the probability that two coins of bias $1/4$ disagree?

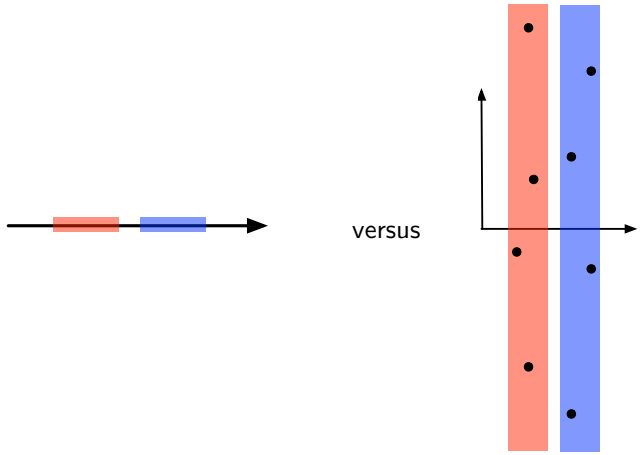
$$R(h_n) \;=\; 2 \cdot \frac{1}{4} \cdot \frac{3}{4} \;=\; \frac{3}{8} \;>\; \frac{1}{4}.$$

But (Cover-Hart 1967) it is always within a factor two of optimal:

$$R(h_n) \to 2R^*(1 - R^*).$$

Nearest neighbor: sensitivity to noise

Adding a single sufficiently noisy feature can wreak havoc with the classifier.



Solutions: feature selection/reweighting; distance function learning.

Fast NN search

Naive search is $O(n)$ for training set of size n: very slow.

Two popular approaches to fast nearest neighbor search, for data set $S \subset \mathcal{X}$ and query q.

1. Locality sensitive hashing
   Collection of special hash functions $h_1, \ldots, h_m : \mathcal{X} \to \mathbb{Z}$.
   Search for nearest neighbor in

$$\bigcup_{i=1}^{m} \{x \in S : h_i(x) = h_i(q)\}$$

   This set is smaller than S, and is likely to contain the nearest neighbor of q.
2. Tree-based search
   Build tree structure on S, and use it to discard subsets of S that are far from a query q.
   Common options: k-d tree, PCA tree, cover tree.

K-d trees for NN search

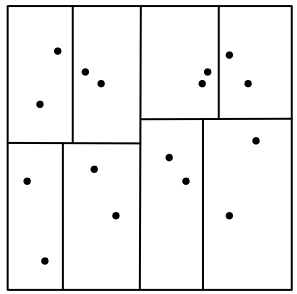
A hierarchical, rectilinear spatial partition.

For data set $S \subset \mathbb{R}^p$:

- Pick a coordinate $1 \leq i \leq p$.
- Compute $v = \text{median}(\{x_i : x \in S\})$.
- Split S into two halves:

$$S_L = \{x \in S : x_i < v\}$$
$$S_R = \{x \in S : x_i \geq v\}$$

- Recurse on S_L, S_R



Two types of search, given a query $q \in \mathbb{R}^p$:

- *Defeatist search*: Route q to a leaf cell and return the NN in that cell. This might not be the true NN.
- *Comprehensive search*: Grow the search region to other cells that cannot be ruled out using the triangle inequality.