The problem we’ll solve today

Given an image of a handwritten digit, say which digit it is.

Nearest neighbor classification

The machine learning approach

Assemble a data set:

The MNIST data set of handwritten digits:

- Training set of 60,000 images and their labels.
- Test set of 10,000 images and their labels.

And let the machine figure out the underlying patterns.

How to classify a new image $x$?

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

How to measure the distance between images?

**MNIST images:**
- Size 28 × 28 (total: 784 pixels)
- Each pixel is grayscale: 0-255

Stretch each image into a vector with 784 coordinates:

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

**Euclidean distance in higher dimension**

Euclidean distance between 784-dimensional vectors $x, z$ is

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

Here $x_i$ is the $i$th coordinate of $x$.

**The distance function**

Remember Euclidean distance in two dimensions?

$$
z = (3, 5)
$$

$$
x = (1, 2)
$$

**Accuracy of nearest neighbor 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**?
  - (One that picks a label 0 – 9 at random?) **90%**.

- Test error of nearest neighbor: **3.09%**.
Examples of errors

Test set of 10,000 points:
- 309 are misclassified
- Error rate 3.09%

Examples of errors:

Query

NN

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.

<table>
<thead>
<tr>
<th>$k$</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test error (%)</td>
<td>3.09</td>
<td>2.94</td>
<td>3.13</td>
<td>3.10</td>
<td>3.43</td>
<td>3.34</td>
</tr>
</tbody>
</table>

In real life, there’s no test set. How to decide which $k$ is best?

1. Hold-out set.
   - Let $S$ be the training set.
   - Choose a subset $V \subset S$ as a validation set.
   - What fraction of $V$ is misclassified by finding the $k$-nearest neighbors in $S \setminus V$?

2. Leave-one-out cross-validation.
   - For each point $x \in S$, find the $k$-nearest neighbors in $S \setminus \{x\}$.
   - What fraction are misclassified?

Cross-validation

How to estimate the error of $k$-NN for a particular $k$?

10-fold cross-validation

- Divide the training set into 10 equal pieces.
  - Training set (call it $S$): 60,000 points
    - Call the pieces $S_1, S_2, \ldots, S_{10}$: 6,000 points each.
  - For each piece $S_i$:
    - Classify each point in $S_i$ using $k$-NN with training set $S - S_i$
    - Let $\epsilon_i$ = fraction of $S_i$ that is incorrectly classified
  - Take the average of these 10 numbers:

    estimated error with $k$-NN = $\frac{\epsilon_1 + \cdots + \epsilon_{10}}{10}$

Another improvement: better distance functions

The Euclidean ($\ell_2$) distance between these two images is very high!

Much better idea: 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:

<table>
<thead>
<tr>
<th>$\ell_2$</th>
<th>tangent distance</th>
<th>shape context</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.09</td>
<td>1.10</td>
<td>0.63</td>
</tr>
</tbody>
</table>
Related problem: feature selection

Feature selection/reweighting is part of picking a distance function. And, one noisy feature can wreak havoc with nearest neighbor!

Algorithmic issue: speeding up NN search

Naive search takes time $O(n)$ for training set of size $n$: slow!

There are data structures for speeding up nearest neighbor search, like:

1. Locality sensitive hashing
2. Ball trees
3. K-d trees

These are part of standard Python libraries for NN, and help a lot.

Example: $k$-d trees for NN search

A hierarchical, rectilinear spatial partition.

For data set $S \subset \mathbb{R}^d$:
- Pick a coordinate $1 \leq i \leq d$.
- 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 $q \in \mathbb{R}^d$:
- 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.

The curse of dimension in NN search

Situation: $n$ data points in $\mathbb{R}^d$.

1. Storage is $O(nd)$
2. Time to compute distance is $O(d)$ for $\ell_p$ norms
3. Geometry
   It is possible to have $2^{O(d)}$ points that are roughly equidistant from each other.

Current methods for fast exact NN search have time complexity proportional to $2^d$ and $\log n$. 
Postscript: useful families of distance functions

1 $\ell_p$ norms
2 Metric spaces

Measuring distance in $\mathbb{R}^m$

Usual choice: Euclidean distance:

$$\|x - z\|_2 = \sqrt{\sum_{i=1}^{m} (x_i - z_i)^2}.$$ 

For $p \geq 1$, here is $\ell_p$ distance:

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

- $p = 2$: Euclidean distance
- $\ell_1$ distance: $\|x - z\|_1 = \sum_{i=1}^{m} |x_i - z_i|
- $\ell_\infty$ distance: $\|x - z\|_\infty = \max_i |x_i - z_i|

Example 1

Consider the all-ones vector $(1, 1, \ldots, 1)$ in $\mathbb{R}^d$.
What are its $\ell_2$, $\ell_1$, and $\ell_\infty$ length?

Example 2

In $\mathbb{R}^2$, draw all points with:
1 $\ell_2$ length 1
2 $\ell_1$ length 1
3 $\ell_\infty$ length 1
Metric spaces

Let $\mathcal{X}$ be the space in which data lie.

A distance function $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ 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)

Example 1

$\mathcal{X} = \mathbb{R}^m$ and $d(x, y) = \|x - y\|_p$

Check:

- $d(x, y) \geq 0$ (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)

Example 2

$\mathcal{X} = \{\text{strings over some alphabet}\}$ and $d = \text{edit distance}$

Check:

- $d(x, y) \geq 0$ (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)

A non-metric distance function

Let $p, q$ be probability distributions on some set $\mathcal{X}$.

The **Kullback-Leibler divergence** or **relative entropy** between $p, q$ is:

$$d(p, q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}.$$