Nearest neighbors

Overview

▶ The NN classifier
▶ Evaluation, hyperparameter tuning
▶ Ways to improve the NN classifier

Example: OCR for digits

▶ Goal: Automatically label images of handwritten digits
▶ Possible labels are \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}
▶ Start with a large collection of already-labeled images

Figure 1: Example OCR digits from MNIST data set

Nearest neighbor (NN) classifier

▶ Nearest neighbor (NN) classifier \( \hat{f}_D \) represented using collection of labeled examples \( D := ((x_1, y_1), \ldots, (x_n, y_n)) \), plus a snippet of code
▶ Input: \( x \)
  ▶ Find \( x_i \) in \( D \) that is “closest” to \( x \) (the nearest neighbor)
  ▶ (Break ties in some arbitrary fixed way)
  ▶ Return \( y_i \)

Figure 2: Schematic of NN classifier
Distances
- Treat (grayscale) images as vectors in Euclidean space $\mathbb{R}^d$
  - $d = 28^2 = 784$
  - Generalizes physical 3-dimensional space
- Each point $x = (x_1, \ldots, x_d)$ is a vector of $d$ real numbers
  - $\|x - z\|_2 = \sqrt{\sum_{i=1}^d (x_i - z_i)^2}$
  - Also called $\ell_2$ distance
- Why use this for images? Simplicity
- Why not use this for images? Spatial information is lost, ...

Figure 3: Pixels of OCR image

OCR via NN
- Images are represented as vectors of real numbers
- Labels are $\{0, 1, \ldots, 9\}$
- Given: 60000 labeled examples
- Construct NN classifier using these examples
  - Distance comes from treating "pixel space" as "Euclidean space"
- How good is this classifier?

Error rate
- Error rate (on a collection of labeled examples $S$)
  - Fraction of labeled examples in $S$ that have incorrect label prediction from $\hat{f}$
  - Written $\text{err}(\hat{f}, S)$
  - (Often, the word "rate" is omitted)
- Error rate of NN classifier?

Test error rate
- Better evaluation: test error rate
  - Train/test split, $S \cap T = \emptyset$
  - Classifier $\hat{f}$ only based on $S$
    - Training error rate: $\text{err}(\hat{f}, S)$
    - Test error rate: $\text{err}(\hat{f}, T)$
  - On OCR data: test error rate is 3.09%
**Why does NN work?**

- Assumption: Nearby points have same label.
- As number of training examples increases, nearest neighbor of a test point becomes closer.
- Corollary: NN will have test error rate zero, given enough training examples.

**Diagnostics**

- Error analysis: look at the data and try to understand what is going on.
- Some mistakes made by NN could have been fixed by plurality vote over three nearest neighbors.

**k-nearest neighbor classifier**

- **k-nearest neighbor (k-NN) classifier**
  - Input: $x$
    - Find the $k$ nearest neighbors of $x$ in $D$
    - Return the plurality of the corresponding labels
  - As before, break ties in some arbitrary fixed way

**Typical effect of $k$**

- Smaller $k$: smaller training error rate
- Larger $k$: higher training error rate, but predictions more “stable” due to voting.

<table>
<thead>
<tr>
<th>$k$</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>test error rate</td>
<td>0.0309</td>
<td>0.0295</td>
<td>0.0312</td>
<td>0.0306</td>
<td>0.0341</td>
</tr>
</tbody>
</table>
Hyperparameter tuning

- \( k \) is a **hyperparameter** of \( k \)-NN
- How to choose hyperparameters?
  - Bad idea: Choosing \( k \) that yields lowest training error rate
    (degenerate choice: \( k = 1 \))
  - Better idea: Simulate train/test split on the training data
- Hold-out approach
  - **Hold-out set** (aka **validation set**)

Distance functions I

- Specialize to input types
  - Edit distance for strings
  - Shape distance for images
  - Time warping distance for audio waveforms

Distance functions II

- Generic distances for vectors of real numbers
  - \( \ell_p \) distances
  \[
  \| \mathbf{x} - \mathbf{z} \|_p = \left( \sum_{i=1}^{d} |x_i - z_i|^p \right)^{1/p}.
  \]
  - What are the unit balls for these distances (in \( \mathbb{R}^2 \))? 

Distance functions III

- On OCR data:
<table>
<thead>
<tr>
<th>distance</th>
<th>( \ell_2 )</th>
<th>( \ell_3 )</th>
<th>tangent</th>
<th>shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>test error rate</td>
<td>0.0309</td>
<td>0.0283</td>
<td>0.0110</td>
<td>0.0063</td>
</tr>
</tbody>
</table>
Features

- When using numerical features (arranged in a vector, like in $\mathbb{R}^d$):
  - Scale of features matters
  - Noisy features can ruin NN
- “Curse of dimension”
  - Weird effects in $\mathbb{R}^d$ for large $d$
  - E.g., can find $2^{\Omega(d)}$ points that are approximately equidistant

Computation for NN

- Brute force search: $\Theta(dn)$ time for each prediction
- Data structures: “improve” to $2^d \log(n)$ time
- Approximate nearest neighbors: sub-linear time to get “approximate” answers