Nearest neighbor classifiers

COMS 4771
1. Nearest neighbor rule
1. Classify images of handwritten digits by the actual digits they represent.

2. Classification problem: $\mathcal{Y} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ (a discrete set).
Nearest neighbor (NN) classifier

\textbf{Given:} labeled examples \( D := \{(x_i, y_i)\}_{i=1}^n \)

\textbf{Predictor:} \( \hat{f}_D : \mathcal{X} \to \mathcal{Y} \)

On input \( x \),

1. Find the point \( x_i \) among \( \{x_i\}_{i=1}^n \) that is “closest” to \( x \) (the \textit{nearest neighbor}).
2. Return \( y_i \).

\[ \begin{array}{c}
4 \\
\hat{f}_D \\
\text{“4”}
\end{array} \]
How to measure distance?

A default choice for distance between points in $\mathbb{R}^d$ is the *Euclidean distance* (also called $\ell_2$ distance):

$$\|u - v\|_2 := \sqrt{\sum_{i=1}^{d} (u_i - v_i)^2}$$

(where $u = (u_1, u_2, \ldots, u_d)$ and $v = (v_1, v_2, \ldots, v_d)$).

Grayscale 28x28 pixel images.

Treat as vectors (of 784 real-valued features) that live in $\mathbb{R}^{784}$.
2. Evaluation
Example: OCR for digits with NN classifier

- Classify images of handwritten digits by the digits they depict.
Example: OCR for digits with NN classifier

- Classify images of handwritten digits by the digits they depict.

\[ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \]

- \( \mathcal{X} = \mathbb{R}^{784} \), \( \mathcal{Y} = \{0, 1, \ldots, 9\} \).
Example: OCR for digits with NN classifier

- Classify images of handwritten digits by the digits they depict.

$0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9$

- $\mathcal{X} = \mathbb{R}^{784}$, $\mathcal{Y} = \{0, 1, \ldots, 9\}$.

- **Given**: labeled examples $D := \{(x_i, y_i)\}_{i=1}^{n} \subset \mathcal{X} \times \mathcal{Y}$. 
Example: OCR for digits with NN classifier

- Classify images of handwritten digits by the digits they depict.

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}
\]

- \(\mathcal{X} = \mathbb{R}^{784}, \mathcal{Y} = \{0, 1, \ldots, 9\}\).

- **Given**: labeled examples \(D := \{(x_i, y_i)\}_{i=1}^{n} \subset \mathcal{X} \times \mathcal{Y}\).

- Construct NN classifier \(\hat{f}_D\) using \(D\).
Example: OCR for digits with NN classifier

- Classify images of handwritten digits by the digits they depict.

- \( \mathcal{X} = \mathbb{R}^{784}, \mathcal{Y} = \{0, 1, \ldots, 9\} \).

- **Given**: labeled examples \( D := \{(x_i, y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y} \).

- Construct NN classifier \( \hat{f}_D \) using \( D \).

- **Question**: Is this classifier any good?
Error rate

- **Error rate** of classifier $f$ on a set of labeled examples $D$:

  \[
  \text{err}(f; D) := \frac{\# \text{ of } (x, y) \in D \text{ such that } f(x) \neq y}{|D|}
  \]

  (i.e., the fraction of $D$ on which $f$ disagrees with paired label).
Error rate

- Error rate of classifier $f$ on a set of labeled examples $D$:

$$\text{err}(f; D) := \frac{\# \text{ of } (x, y) \in D \text{ such that } f(x) \neq y}{|D|}$$

(i.e., the fraction of $D$ on which $f$ disagrees with paired label).

- Question: What is $\text{err}(\hat{f}_D; D)$?
A better way to evaluate the classifier

- Split the labeled examples $\{(x_i, y_i)\}_{i=1}^{n}$ into two sets (randomly).
  - Training data $S$.
  - Test data $T$.
A better way to evaluate the classifier

- Split the labeled examples \( \{(x_i, y_i)\}_{i=1}^{n} \) into two sets (randomly).
  - Training data \( S \).
  - Test data \( T \).

- Only use training data \( S \) to construct NN classifier \( \hat{f}_{S} \).
A better way to evaluate the classifier

- Split the labeled examples \( \{(x_i, y_i)\}_{i=1}^{n} \) into two sets (randomly).
  - Training data \( S \).
  - Test data \( T \).

- Only use training data \( S \) to construct NN classifier \( \hat{f}_S \).
  - Training error rate of \( \hat{f}_S \): \( \text{err}(\hat{f}_S; S) = 0\% \).
A better way to evaluate the classifier

- Split the labeled examples \( \{(x_i, y_i)\}_{i=1}^{n} \) into two sets (randomly).
  - *Training data* \( S \).
  - *Test data* \( T \).

- Only use *training data* \( S \) to construct NN classifier \( \hat{f}_S \).
  - Training error rate of \( \hat{f}_S \): \( \text{err}(\hat{f}_S; S) = 0\% \).

- Use *test data* \( T \) to evaluate accuracy of \( \hat{f}_S \).
A better way to evaluate the classifier

- Split the labeled examples \( \{(x_i, y_i)\}_i=1^n \) into two sets (randomly).
  - Training data \( S \).
  - Test data \( T \).

- Only use training data \( S \) to construct NN classifier \( \hat{f}_S \).
  - Training error rate of \( \hat{f}_S \): \( \text{err}(\hat{f}_S; S) = 0\% \).

- Use test data \( T \) to evaluate accuracy of \( \hat{f}_S \).
  - Test error rate of \( \hat{f}_S \): \( \text{err}(\hat{f}_S; T) = 3.09\% \).
A better way to evaluate the classifier

- Split the labeled examples \( \{(x_i, y_i)\}_{i=1}^{n} \) into two sets (randomly).
  - Training data \( S \).
  - Test data \( T \).

- Only use training data \( S \) to construct NN classifier \( \hat{f}_S \).
  - Training error rate of \( \hat{f}_S \): \( \text{err}(\hat{f}_S; S) = 0\% \).

- Use test data \( T \) to evaluate accuracy of \( \hat{f}_S \).
  - Test error rate of \( \hat{f}_S \): \( \text{err}(\hat{f}_S; T) = 3.09\% \).

Is this good?
3. Upgrading the nearest neighbor rule
Diagnostics

- Some mistakes made by the NN classifier (test point in $T$, nearest neighbor in $S$):

```
28  3 5  54
```
Some mistakes made by the NN classifier (test point in $T$, nearest neighbor in $S$):

- First mistake (correct label is “2”) could’ve been avoided by looking at the three nearest neighbors (whose labels are “8”, “2”, and “2”).

```
[3 8 8 2 2]
```

- Test point   three nearest neighbors
$k$-nearest neighbors classifier

**Given:** labeled examples $D := \{(x_i, y_i)\}_{i=1}^n$

**Predictor:** $\hat{f}_{D,k} : \mathcal{X} \rightarrow \mathcal{Y}$:

On input $x$,

1. Find the $k$ points $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ among $\{x_i\}_{i=1}^n$ “closest” to $x$ (the $k$ nearest neighbors).
2. Return the plurality of $y_{i_1}, y_{i_2}, \ldots, y_{i_k}$.

(Break ties in both steps arbitrarily.)
Effect of $k$

- Smaller $k$: smaller training error rate.
- Larger $k$: higher training error rate, but predictions are 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>
The hold-out set approach

1. Pick a subset \( V \subset S \) (\textit{hold-out set}, a.k.a. \textit{validation set}).
2. For each \( k \in \{1, 3, 5, \ldots \} \):
   - Construct \( k \)-NN classifier \( \hat{f}_{S \setminus V, k} \) using \( S \setminus V \).
   - Compute error rate of \( \hat{f}_{S \setminus V, k} \) on \( V \) (\textit{“hold-out error rate”}).
3. Pick the \( k \) that gives the smallest hold-out error rate.
Choosing $k$

The hold-out set approach

1. Pick a subset $V \subset S$ (hold-out set, a.k.a. validation set).
2. For each $k \in \{1, 3, 5, \ldots \}$:
   - Construct $k$-NN classifier $\hat{f}_{S \setminus V, k}$ using $S \setminus V$.
   - Compute error rate of $\hat{f}_{S \setminus V, k}$ on $V$ ("hold-out error rate").
3. Pick the $k$ that gives the smallest hold-out error rate.

(There are many other approaches.)
4. Other issues with nearest neighbor prediction
Better distance functions

- **Strings**: edit distance

\[ \text{dist}(u, v) = \# \text{ insertions/deletions/mutations needed to change } u \text{ to } v. \]
Better distance functions

- **Strings**: edit distance
  \[ \text{dist}(u, v) = \# \text{ insertions/deletions/mutations needed to change } u \text{ to } v. \]

- **Images**: shape context distance
  \[ \text{dist}(u, v) = \text{how much “warping” is required to change } u \text{ to } v. \]
Better distance functions

- **Strings**: edit distance
  
  \[ \text{dist}(u, v) = \# \text{ insertions/deletions/mutations needed to change } u \text{ to } v. \]

- **Images**: shape context distance
  
  \[ \text{dist}(u, v) = \text{how much “warping” is required to change } u \text{ to } v. \]

- **Audio waveforms**: dynamic time warping

- Etc.
Better distance functions

- **Strings**: edit distance
  \[ \text{dist}(u, v) = \# \text{ insertions/deletions/mutations needed to change } u \text{ to } v. \]

- **Images**: shape context distance
  \[ \text{dist}(u, v) = \text{how much “warping” is required to change } u \text{ to } v. \]

- **Audio waveforms**: dynamic time warping

- Etc.

<table>
<thead>
<tr>
<th>OCR digits classification</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>3.09%</td>
<td>2.83%</td>
<td>1.10%</td>
<td>0.63%</td>
</tr>
</tbody>
</table>
**Caution:** nearest neighbor classifier can be broken by bad/noisy features!
Naïve method for computing NN predictions: \( O(n) \) distance computations.
Computation

- Naïve method for computing NN predictions: $O(n)$ distance computations.
- Better: organize training data in a data structure to improve look-up time.
Naïve method for computing NN predictions: \( O(n) \) distance computations.

Better: organize training data in a data structure to improve look-up time.

- Space: \( O(nd) \) for \( n \) points in \( \mathbb{R}^d \).
- Query time: \( O(2^d \log n) \) time in worst-case.
Naïve method for computing NN predictions:
$O(n)$ distance computations.

Better: organize training data in a data structure to improve look-up time.

- Space: $O(nd)$ for $n$ points in $\mathbb{R}^d$.
- Query time: $O(2^d \log n)$ time in worst-case.

Finding an “approximate” NN can be more efficient.
Naïve method for computing NN predictions: 
\( O(n) \) distance computations.

Better: organize training data in a data structure to improve look-up time.

- Space: \( O(nd) \) for \( n \) points in \( \mathbb{R}^d \).
- Query time: \( O(2^d \log n) \) time in worst-case.

Finding an “approximate” NN can be more efficient.

E.g., how to quickly find a point among the top-1% closest points?
Naïve method for computing NN predictions: \(O(n)\) distance computations.

Better: organize training data in a data structure to improve look-up time.

- Space: \(O(nd)\) for \(n\) points in \(\mathbb{R}^d\).
- Query time: \(O(2^d \log n)\) time in worst-case.

Finding an “approximate” NN can be more efficient.

E.g., how to quickly find a point among the top-1% closest points?

- Popular technique: Locality sensitive hashing
Questions

- In what sense is $k$-NN a good learning method?
- Why is test error rate a better way to evaluate the classifier?
Questions

- In what sense is $k$-NN a good learning method?
- Why is test error rate a better way to evaluate the classifier?

We will answer these questions in the context of a statistical model. (Next lecture!)
1. $k$-NN learning procedure; role of $k$, distance functions, features.
2. Training and test error rates.