Programming Assignment 1

Assigned: Sept. 8
Due: Sept. 27.

The nearest neighbors technique is a powerful method used in machine learning. Here’s an example. Suppose that you have an image, and you want to identify what is in the picture: Is it a camel, a building, an eggbeater or what? What you do is

- You download 80,000,000 labelled images from Google Images.
- You find the 20 or so images that most nearly resemble the picture you have.
- You take the most common label among those 20, and guess that that’s what your picture is. E.g. if among these 20 images, 9 are labelled “camel” 5 are labelled “giraffe”, 3 are labelled “elk”, and the rest are labelled “eggbeater”, “Eiffel Tower”, and “July 3” then guess that your picture is a camel.

Prof. Rob Fergus of this department wrote a famous paper “80 Million Tiny Images” that worked this way. (There’s a link on the course web site.) It worked amazingly well.

You can use this same technique for all kinds of classification problems. For example, you want to know whether an email message is spam or not. What you do is get a collection of message, correctly labelled “spam” or “legit”, and find the 20 that most closely resemble your message. If these are mostly are spam, then guess that your message is spam; if they’re mostly legit, guess that your message is legit.

Abstracting, we can describe this technique as follows: You have a collection of pairs \( \langle D_i, L_i \rangle \) of a data point \( D_i \) with a label \( L_i \). You have an unlabelled point \( P \) which you wish to label. You have a distance function \( \text{Dist}(P, Q) \) which gives a numeric value for the “distance” between \( P \) and \( Q \) — that is, a measure of how dissimilar they are. You fix a value of \( k \) points that you’re going to use. You find the \( k \) labelled points in the collection whose distance to \( P \) is smallest. Then you choose the most common label among those \( k \); and that’s your answer.

Of course, what kind of “distance function” you use depends strongly on the type of entities that are the data points, and to a lesser extent on the categories you’re trying to match. You certainly need a different distance function if you’re comparing images than if you’re comparing email messages. You probably need a different distance function if you’re trying to distinguish camels from giraffes, than if you’re trying to tell whether a newly discovered painting is by Rembrandt or Caravaggio.

So in this programming assignment you will implement the nearest neighbors function in a way that will work for any distance function, over a particular type of data point, namely arrays of 3 reals (sorry about the anti-climax). In fact, you’ll use two different techniques to do this: abstract methods and interfaces. Skeletons of the code to do this is given in the two files NNAbs.java and NNInt.java on the course web site. You should fill in the missing parts of the code so that they work properly. You should not change any of the code that is currently there, you should just fill in the gaps.

In both of these files, there are three distance functions defined:
- \( \text{SquaredEuclideanDist}(P, Q) \) is the sum of the squares of the difference of the values in each coordinate.
- \( \text{MaxDiff}(P, Q) \) is the maximum of the absolute value of the difference of the values in each coordinate.
- \( \text{SumAbs}(P, Q) \) is the sum of the absolute value of the difference of the values in each coordinate.
For example, if \( P = (3, 3, 3) \) and \( Q = (3, 5, 9) \) then
\[
\text{SquaredEuclideanDist}(P, Q) = (3 - 3)^2 + (5 - 3)^2 + (9 - 3)^2 = 0 + 4 + 36 = 40.
\]
\[
\text{MaxDiff}(P, Q) = \max(|3 - 3|, |5 - 3|, |9 - 3|) = 6.
\]
\[
\text{SumAbs}(P, Q) = |3 - 3| + |5 - 3| + |9 - 3| = 8.
\]

The \textit{main} method in both files defines the new point \( P \) to be labelled as \( (3, 3, 3) \). The table below shows the five points in the data set and their three distances from \( P \).

<table>
<thead>
<tr>
<th>Point</th>
<th>Label</th>
<th>SED</th>
<th>MaxDiff</th>
<th>SumAbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ( (0, 0, 0) )</td>
<td>true</td>
<td>27</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>2 ( (7, 3, 3) )</td>
<td>true</td>
<td>16</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3 ( (10, 3, 3) )</td>
<td>false</td>
<td>49</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>4 ( (4, 8, 8) )</td>
<td>true</td>
<td>51</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>5 ( (3, 5, 9) )</td>
<td>false</td>
<td>40</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

If we choose \( k = 3 \), then with distance SED, the closest points are points 2, 1, and 5, for an overall vote of \textit{true}. With distance MaxDiff, the closest points are 1,2,4 for an overall vote of \textit{true}. With distance SumAbs, the closest points are 2,3,5, for an overall vote of \textit{false}.

For this assignment, you do not have to worry about the efficiency of your code; you can use any method you want to look for the \( k \) closest data points.

**Honors assignment**

Do the following, more complex, version of the assignment described above:

A more sophisticated way of combining votes is to give the different labelled points \( D[i] \) different weights, depending on how close they are to \( P \). That is, there is a weight function \( w(d) \) which is a decreasing function of the distance \( d \). You find the \( k \) nearest points; add up \( w(\text{Distance}(D[i], P)) \) for all the point that vote \textit{true}; add up \( w(\text{Distance}(D[i], P)) \) for all the point that vote \textit{false}; and go with the label with the largest total label.

In the above example, if we choose the distance function to be \textit{SumAbs} and choose \( w(d) = 1/d^2 \), then point 2 votes \textit{true} for a total weight of \( 1/4^2 = 0.625 \); points 3 and 5 vote \textit{false} for a total weight of \( 1/7^2 + 1/8^2 = 0.036 \), so the decision would be \textit{true}.

The simple majority vote corresponds to a weighting function which is 1.0 for all distances.

Using both the technique of abstract methods and the technique of interfaces, write a version of the code so that \textit{TallyVotes} can use different weighting functions.