Data Mining

Lecture 03: Nearest Neighbor Learning

Theses slides are based on the slides by

- Tan, Steinbach and Kumar (textbook authors)
- Prof. R. Mooney (UT Austin)
- Prof E. Keogh (UCR),
- Prof. F. Provost (Stern, NYU)
Nearest Neighbor Classifier

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it’s probably a duck
Nearest Neighbor Classifier

If the nearest instance to the previously unseen instance is a Katydid, class is Katydid. Else, class is Grasshopper.

Evelyn Fix 1904-1965
Joe Hodges 1922-2000

Katydids
Grasshoppers
Different Learning Methods

- Eager Learning
  - Explicit description of target function on the whole training set
  - Pretty much all methods we will discuss except this one

- Lazy Learner
  - Instance-based Learning
  - Learning=storing all training instances
  - Classification=assigning target function to a new instance
Instance-Based Classifiers

- **Examples:**
  - **Rote-learner**
    - Memorizes entire training data and performs classification only if attributes of record exactly match one of the training examples
      - No generalization
  - **Nearest neighbor**
    - Uses $k$ “closest” points (nearest neighbors) for performing classification
      - Generalizes
Nearest-Neighbor Classifiers

- Requires three things
  - The set of stored records
  - Distance metric to compute distance between records
  - The value of $k$, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify $k$ nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)
Similarity and Distance

- One of the fundamental concepts of data mining is the notion of similarity between data points, often formulated as a numeric distance between data points.

- Similarity is the basis for many data mining procedures.

- We learned a bit about similarity when we discussed data.

- We will consider the most direct use of similarity today.

- Later we will see it again (for clustering)
kNN learning

- Assumes $X = n$-dim. space
  - discrete or continuous $f(x)$

- Let
  - $x = \langle a_1(x), ..., a_n(x) \rangle$
  - $d(x_i, x_j) =$ Euclidean distance

- Algorithm (given new $x$)
  - find $k$ nearest stored $x_i$ (use $d(x, x_i)$)
  - take the most common value of $f$
Similarity/Distance Between Data Items

- Each data item is represented with a set of attributes (consider them to be numeric for the time being)

\[ D(X_i, X_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \]

“Closeness” is defined in terms of the distance (Euclidean or some other distance) between two data items.

- Euclidean distance between \( X_i = \langle a_1(X_i), \ldots, a_n(X_i) \rangle \) and \( X_j = \langle a_1(X_j), \ldots, a_n(X_j) \rangle \) is defined as:
- \( \text{Distance}(\text{John, Rachel}) = \sqrt{(35-22)^2 + (35K-50K)^2 + (3-2)^2} \)

John:
Age = 35
Income = 35K
No. of credit cards = 3

Rachel:
Age = 22
Income = 50K
No. of credit cards = 2
Some Other Distance Metrics

- **Cosine Similarity**
  - Cosine of the angle between the two vectors.
  - Used in text and other high-dimensional data.

- **Pearson correlation**
  - Standard statistical correlation coefficient.
  - Used for bioinformatics data.

- **Edit distance**
  - Used to measure distance between unbounded length strings.
  - Used in text and bioinformatics.
Nearest Neighbors for Classification

- To determine the class of a new example E:
  - Calculate the distance between E and all examples in the training set
  - Select $k$ examples closest to E in the training set
  - Assign E to the most common class (or some other combining function) among its $k$ nearest neighbors
k-Nearest Neighbor Algorithms

- Also called Instance-based learning (IBL) or Memory-based learning
- No model is built: Store all training examples
- Any processing is delayed until a new instance must be classified → lazy classification technique
### k-Nearest Neighbor Classifier

**Example (k=3)**

<table>
<thead>
<tr>
<th>Customer</th>
<th>Age</th>
<th>Income (K)</th>
<th>No. of cards</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>35</td>
<td>35</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>Rachel</td>
<td>22</td>
<td>50</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>Ruth</td>
<td>63</td>
<td>200</td>
<td>1</td>
<td>No</td>
</tr>
<tr>
<td>Tom</td>
<td>59</td>
<td>170</td>
<td>1</td>
<td>No</td>
</tr>
<tr>
<td>Neil</td>
<td>25</td>
<td>40</td>
<td>4</td>
<td>Yes</td>
</tr>
<tr>
<td>David</td>
<td>37</td>
<td>50</td>
<td>2</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Distance from David**

- \( \sqrt{(35-37)^2 + (35-50)^2 + (3-2)^2} = 15.16 \)
- \( \sqrt{(22-37)^2 + (50-50)^2 + (2-2)^2} = 15 \)
- \( \sqrt{(25-37)^2 + (40-50)^2 + (4-2)^2} = 15.74 \)
Strengths and Weaknesses

**Strengths:**
- Simple to implement and use
- Comprehensible – easy to explain prediction
- Robust to noisy data by averaging $k$-nearest neighbors
- Distance function can be tailored using domain knowledge
- Can learn complex decision boundaries
  - Much more expressive than linear classifiers & decision trees
  - More on this later

**Weaknesses:**
- Need a lot of space to store all examples
- Takes much more time to classify a new example than with a parsimonious model (need to compare distance to all other examples)
- Distance function must be designed carefully with domain knowledge
Are These Similar at All?

Humans would say “yes” although not perfectly so (both Homer)

Nearest Neighbor method without carefully crafted features would say “no” since the colors and other superficial aspects are completely different. Need to focus on the shapes.

Notice how humans find the image to the right to be a bad representation of Homer even though it is a nearly perfect match of the one above.
Distance between neighbors can be dominated by an attribute with relatively large values (e.g., income in our example).

Important to normalize (e.g., map numbers to numbers between 0-1)

- Example: Income
- Highest income = 500K
- John’s income is normalized to 95/500, Rachel’s income is normalized to 215/500, etc.

(there are more sophisticated ways to normalize)
The nearest neighbor algorithm is sensitive to the units of measurement.

X axis measured in **centimeters**
Y axis measure in dollars
The nearest neighbor to the **pink** unknown instance is **red**.

X axis measured in **millimeters**
Y axis measure in dollars
The nearest neighbor to the **pink** unknown instance is **blue**.

One solution is to normalize the units to pure numbers. Typically the features are Z-normalized to have a mean of zero and a standard deviation of one. $X = (X - \text{mean}(X))/\text{std}(x)$
Feature Relevance and Weighting

- Standard distance metrics weight each feature equally when determining similarity.
  - Problematic if many features are irrelevant, since similarity along many irrelevant examples could mislead the classification.

- Features can be weighted by some measure that indicates their ability to discriminate the category of an example, such as information gain.

- Overall, instance-based methods favor global similarity over concept simplicity.

<table>
<thead>
<tr>
<th>Training Data</th>
<th>Test Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+ + - + ??
Strengths and Weaknesses II

- Distance works naturally with numerical attributes

Distance(John, Rachel) = \sqrt{(35-22)^2+(35,000-50,000)^2+(3-2)^2} = 15.16

- What if we have nominal/categorical attributes?
  - Example: married

<table>
<thead>
<tr>
<th>Customer</th>
<th>Married</th>
<th>Income (K)</th>
<th>No. of cards</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Yes</td>
<td>35</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>Rachel</td>
<td>No</td>
<td>50</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>Ruth</td>
<td>No</td>
<td>200</td>
<td>1</td>
<td>No</td>
</tr>
<tr>
<td>Tom</td>
<td>Yes</td>
<td>170</td>
<td>1</td>
<td>No</td>
</tr>
<tr>
<td>Neil</td>
<td>No</td>
<td>40</td>
<td>4</td>
<td>Yes</td>
</tr>
<tr>
<td>David</td>
<td>Yes</td>
<td>50</td>
<td>2</td>
<td>?</td>
</tr>
</tbody>
</table>
Recall: Geometric interpretation of classification models

- Bad risk (Default) – 16 cases
- Good risk (Not default) – 14 cases
How does 1-nearest-neighbor partition the space?

- Very different boundary (in comparison to what we have seen)
- Very tailored to the data that we have
- Nothing is ever wrong on the training data
- Maximal overfitting

- Bad risk (Default) – 16 cases
- Good risk (Not default) – 14 cases
We can visualize the nearest neighbor algorithm in terms of a decision surface...

Note the we don’t actually have to construct these surfaces, they are simply the implicit boundaries that divide the space into regions “belonging” to each instance.

Although it is not necessary to explicitly calculate these boundaries, the learned classification rule is based on regions of the feature space closest to each training example.

This division of space is called Dirichlet Tessellation (or Voronoi diagram, or Theissen regions).
The nearest neighbor algorithm is sensitive to outliers...

The solution is to...
We can generalize the nearest neighbor algorithm to the \( k \)-nearest neighbor (kNN) algorithm.

We measure the distance to the nearest \( k \) instances, and let them vote. \( k \) is typically chosen to be an odd number.

\( k \) – complexity control for the model

![K = 1](image1)

![K = 3](image2)
Curse of Dimensionality

- Distance usually relates to all the attributes and assumes all of them have the same effects on distance.

- The similarity metrics do not consider the relation of attributes which result in inaccurate distance and then impact on classification precision. Wrong classification due to presence of many irrelevant attributes is often termed as the curse of dimensionality.

- For example:
  - If each instance is described by 20 attributes out of which only 2 are relevant in determining the classification of the target function.
  - Then instances that have identical values for the 2 relevant attributes may nevertheless be distant from one another in the 20-dimensional instance space.
The nearest neighbor algorithm is sensitive to irrelevant features... 

Suppose the following is true, if an insect's antenna is longer than 5.5 it is a **Katydid**, otherwise it is a **Grasshopper**.

Using just the antenna length we get perfect classification!

Suppose however, we add in an **irrelevant** feature, for example the insects mass.

Using both the antenna length and the insects mass with the 1-NN algorithm we get the wrong classification!
How to Mitigate Irrelevant Features?

- Use more training instances
  - Makes it harder to obscure patterns
- Ask an expert which features are relevant and irrelevant and possibly weight them
- Use statistical tests (prune irrelevant features)
- Search over feature subsets
Why Searching over Feature Subsets is Hard

Suppose you have the following classification problem, with 100 features. Features 1 and 2 (the X and Y below) give perfect classification, but all 98 of the other features are irrelevant...

Using all 100 features will give poor results, but so will using only Feature 1, and so will using Feature 2! Of the \(2^{100} - 1\) possible subsets of the features, only one really works.
k-Nearest Neighbor (kNN)

- Unlike all the previous learning methods, kNN does not build model from the training data.
- To classify a test instance $d$, define $k$-neighborhood $P$ as $k$ nearest neighbors of $d$
- Count number $n$ of training instances in $P$ belonging to class $c_j$
- Estimate $P_r(c_j|d)$ as $n/k$
- Note: classification time is linear in training set size for each test case.
Nearest Neighbor Variations

- Can be used to estimate the value of a real-valued function (regression)
  - Take average value of the $k$ nearest neighbors
- Weight each nearest neighbor’s “vote” by the inverse square of its distance from the test instance
  - More similar examples count more
Example: k=6 (6NN)

What is probability of science for the new point?
**kNN - Important Details**

- **kNN** estimates the value of the target variable by a function of the $k$-nearest neighbors. The simplest techniques are:
  - for classification: majority vote
  - for regression: mean/median/mode
  - for class probability estimation: fraction of positive neighbors

- **For all of these cases**, it may make sense to create a weighted average based on the distance to the neighbors, so that closer neighbors have more influence in the estimation.

- **In Weka**: classifiers → lazy → IBk (for “instance based”)
  - parameters for distance weighting and automatic normalization
  - can set $k$ automatically based on (nested) cross-validation
Case-based Reasoning (CBR)

- CBR is similar to instance-based learning
  - But may reason about the differences between the new example and match example
- CBR Systems are used a lot
  - help-desk systems
  - legal advice
  - planning & scheduling problems
  - Next time you are on the help with tech support
    - The person maybe asking you questions based on prompts from a computer program that is trying to most efficiently match you to an existing case!
Summary of Pros and Cons

- **Pros:**
  - No learning time (lazy learner)
  - Highly expressive since can learn complex decision boundaries
  - Via use of k can avoid noise
  - Easy to explain/justify decision

- **Cons:**
  - Relatively long evaluation time
  - No model to provide insight
  - Very sensitive to irrelevant and redundant features
  - Normalization required