



Artificial Intelligence Programming

Instance Based Learning

Chris Brooks

Department of Computer Science

University of San Francisco



Instance-Based Learning

- So far, all of the learning algorithms we've studied construct an explicit hypothesis about the data set.
- This is nice because it lets us do a lot of the training ahead of time.
- It has the weakness that we must then use the same hypothesis for each element in the test set.
- One way to get around this is to construct different hypotheses for each test example.
 - Potentially better results, but more computation needed at evaluation time.
- We can use this in either a supervised or unsupervised setting.

k-nearest neighbor

- The most basic instance-based method is k-nearest neighbor.
- Assume:
 - Each individual can be represented as an N-dimensional vector: $\langle v_1, v_2, \dots, v_n \rangle$.
 - We have a distance metric that tells us how far apart two individuals are.
 - Euclidean distance is common:

$$d(x_1, x_2) = \sqrt{\sum (x_1[i] - x_2[i])^2}$$

Supervised kNN

- Training is trivial.
 - Store training set. Assume each individual is a n -dimensional vector, plus a classification.
- Testing is more computationally complex:
 - Find the k closest points and collect their classifications.
 - Use majority rule to classify the unseen point.

kNN Example

- Suppose we have the following data points and are

using 3-NN:

X1	X2	Class
4	3	+
1	2	-
2	2	+
5	0	-

- We see the following data point: $x_1=3$, $x_2 = 1$. How should we classify it?

kNN Example

- Begin by computing distances:

X1	X2	Class	Distance
4	3	+	$\sqrt{5} = 2.23$
1	1	-	2
2	2	+	$\sqrt{2} = 1.41$
5	1	-	2

- The three closest points are 2,3,4. There are 2 '-', and 1 '+'.
 - Therefore the new example is negative.

Discussion

- K-NN can be a very effective algorithm when you have lots of data.
 - Easy to compute
 - Resistant to noise.
- Bias: points that are “close” to each other share classification.

Discussion

- Issues:
- How to choose the best k ?
 - Search using cross-validation
- Distance is computed globally.
 - Recall the data we used for decision tree training.
 - Part of the goal was eliminate irrelevant attributes.
- All neighbors get an equal vote.

Distance-weighted voting

- One extension is to weight a neighbor's vote by its distance to the example to be classified.
- Each 'vote' is weighted by the inverse square of the distance.
- Once we add this, we can actually drop the 'k', and just use all instances to classify new data.

Attribute Weighting

- A more serious problem with kNN is the presence of irrelevant attributes.
- In many data sets, there are a large number of attributes that are completely unrelated to classification.
- More data actually lowers classification performance.
 - This is sometimes called the *curse of dimensionality*.

Attribute Weighting

- We can address this problem by assigning a weight to each component of the distance calculation.
- $d(p_1, p_2) = \sqrt{(\sum w[i](p_1[i] - p_2[i]))^2}$ where w is a vector of weights.
- This has the effect of transforming or stretching the instance space.
- More useful features have larger weights

Learning Attribute Weights

- We can learn attribute weights through a hillclimbing search.

```
let w = random weights
```

```
let val(w) be the error rate for w under n-fold cross-validation
```

```
while not done :
```

```
  for i in range(len(w)) :
```

```
    w[i] = w[i] +  $\delta$ 
```

```
    if val(w + w[i]) > val(w) :
```

```
      keep new weights
```

- We could also use a GA or simulated annealing to do this.

Unsupervised Learning

- What if we want to group instances, but we don't know their classes?
- We just want “similar” instances to be in the same group.
- Examples:
 - Clustering documents based on text
 - Grouping users with similar preferences
 - Identifying demographic groups

K-means Clustering

- Let's suppose we want to group our items into K clusters.
 - For the moment, assume K given.
- Approach 1:
- Choose K items at random. We will call these the *centers*.
- Each center gets its own cluster.
- For each other item, assign it to the cluster that minimizes distance between it and the center.
- This is called K -means clustering.

K-means Clustering

- To evaluate this, we measure the sum of all distances between instances and the center of their cluster.
- But how do we know that we picked good centers?

K-means Clustering

- To evaluate this, we measure the sum of all distances between instances and the center of their cluster.
- But how do we know that we picked good centers?
- We don't. We need to adjust them.

Tuning the centers

- For each cluster, find its mean.
 - This is the point c that minimizes the total distance to all points in the cluster.
- But what if some points are now in the wrong cluster?

Iterate

- Check all points to see if they are in the correct cluster.
- If not, reassign them.
- Then recompute centers.
- Continue until no points change clusters.

K-means pseudocode

```
centers = random items
```

```
while not done :
```

```
    foreach item :
```

```
        assign to closest center
```

```
    foreach center :
```

```
        find mean of its cluster.
```

Hierarchical Clustering

- K-means produces a flat set of clusters.
- Each document is in exactly one cluster.
- What if we want a tree of clusters?
 - Topics and subtopics.
 - Relationships between clusters.
- We can do this using *hierarchical clustering*

Hierarchical Clustering

- One application is in document processing.
- Given a collection of documents, organize them into clusters based on topic.
- No preset list of potential categories, or labeled documents.
- Algorithm:
 - $D = \{d_1, d_2, \dots, d_n\}$
 - While $|D| > k$:
 - Find the documents d_i and d_j that are closest according to some similarity measure.
 - Remove them from D
 - Construct a new d' that is the “union” of d_i and d_j and add it to D

Recommender Systems

- One application of these sorts of approaches is in *recommender systems*
 - Netflix, Amazon
- Goal: Suggest items to users that they're likely to be interested in.
- Real goal: For a given user, find other users she is similar to.

Basic Approach

- A user is modeled as a vector of items she has rated.
- For every other user, compute the distance to that user.
 - (We might also use K-means here ahead of time)
- Find the closest user(s), and suggest items that similar users liked.

Advantages

- Computation is simple and scalable
- No need to model the items themselves
 - Don't need an ontology, or even any idea of what items are.
- Performs better as more data as added.

Algorithmic Challenges

- Curse of dimensionality
- Not all items are independent
 - We might want to learn weights for items, or combine items into larger groups.
- This approach tends to recommend popular items.
 - They're likely to have been rated by lots of people.

Practical Challenges

- How to get users to rate items?
- How to get users to rate truthfully?
- What about new and unrated items?
- What if a user is not similar to anyone?

Summary

- Instance-based learning is a very effective approach to dealing with large numeric data sets.
- k-NN can be used in supervised settings.
- In unsupervised settings, k-means is a simple and effective choice.
- Most recommender systems use a form of this approach.