K-Nearest Neighbors

Data Mining

Prof. Sujee Lee

Department of Systems Management Engineering

Sungkyunkwan University

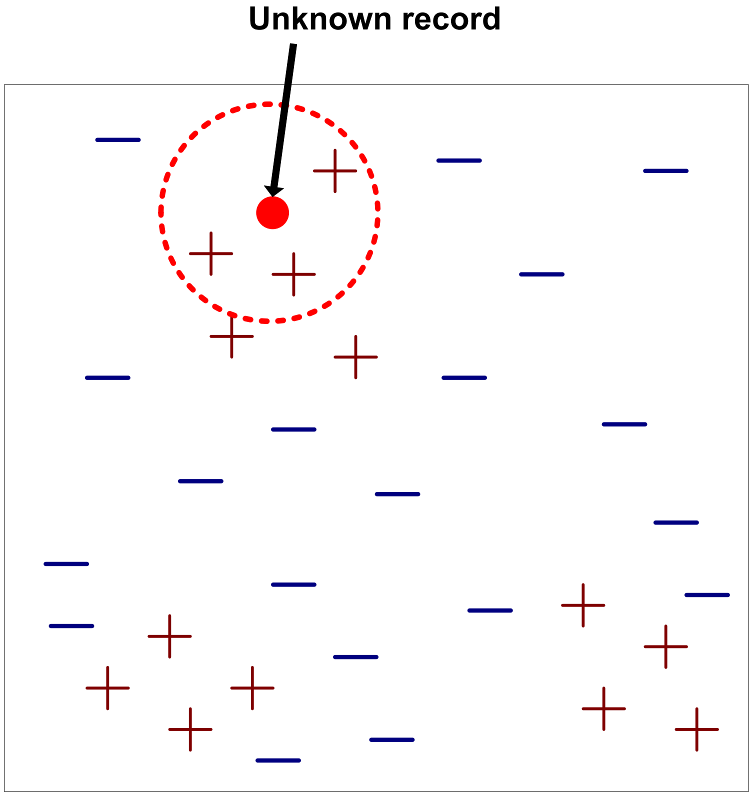
Non-parametric Models

- **Parametric Models**
  - Parametric models (model-based approaches) use a fixed number of parameters
  - These models assume a predefined structure and estimate parameters from the data to make predictions, regardless of the data size

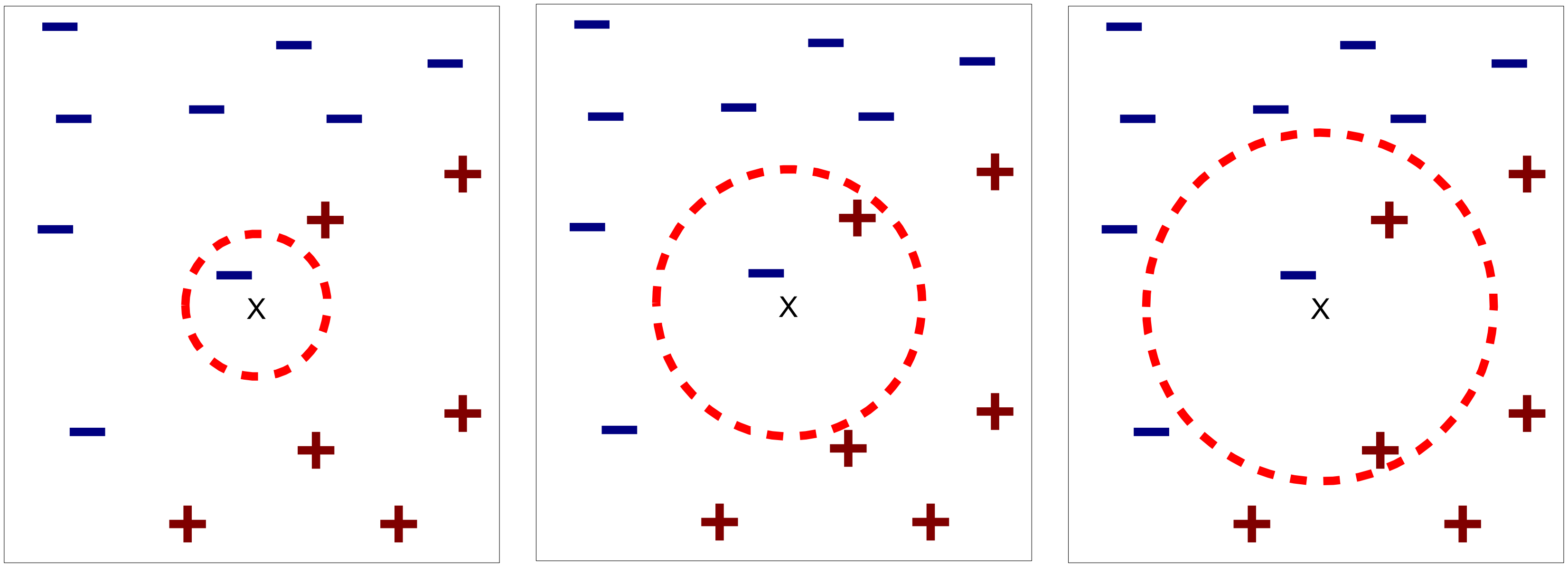
- **Non-Parametric Model**
  - No fixed number of parameters
    - The complexity of the model is determined by the size of the data
    - Example: As the data size increases, the model becomes more complex
  - Fewer assumptions about data
    - Makes minimal assumptions about the shape or distribution of data
    - More flexible predictions with more data

k-Nearest Neighbors

- **k-Nearest Neighbors**
  - Requires three things
    - The set of **stored observations**
    - **Distance metric** to compute distance between observations
    - The value of k, **the number of nearest neighbors** to retrieve
  - To classify an unknown record:
    - Compute distance to other training observations
    - Identify k nearest neighbors
    - Use class labels of nearest neighbors to determine the class label of unknown observation (e.g., by taking majority vote)



Definition of Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

k-nearest neighbors of a record x are data points that have the k smallest distance to x

Nearest Neighbor Classification

- **Compute distance between two objects (observations)**
  - E.g. Euclidean distance

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{j=1}^{p} (x_{1j} - x_{2j})^2}$$

- **Algorithm**

1: Let k be the number of nearest neighbors and D be the set of training examples.

2: **for** each test example $z = (\mathbf{x}')$ **do**

3: Compute $d(\mathbf{x}', \mathbf{x})$, the distance between z and every example, $(\mathbf{x}, y) \in D$.

4: Select $D_z \subseteq D$, the set of k closest training examples to z.

5: $y' = \arg\max_{v} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$

6: **end for**

Nearest Neighbor Classification

- **Distance-weighted voting**

$$y' = \arg\max_{v} \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i \times I(v = y_i)\,, \quad w_i = \frac{1}{d(\mathbf{x}', \mathbf{x}_i)}$$

  - Training examples that are located far away from z have a weaker impact on classification compared to those that are located close to z .

Toy Example

Object	x1	x2	Class
1	5	7	1
2	4	3	2
3	7	8	2
4	8	6	2
5	3	6	1
6	2	5	1
7	9	6	2

	1	2	3	4	5	6	7
1	0	4.123	2.236	3.162	2.236	3.606	4.123
2	4.123	0	5.831	5.000	3.162	2.828	6.481
3	2.236	5.831	0	2.236	4.472	5.831	2.828
4	3.162	5.000	2.236	0	5.000	6.083	1.000
5	2.236	3.162	4.472	5.000	0	1.414	6.000
6	3.606	2.828	5.831	6.083	1.414	0	7.071
7	4.123	6.481	2.828	1.000	6.000	7.071	0

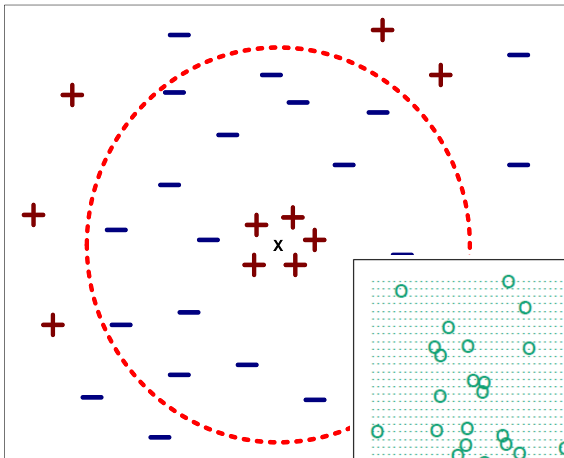
Object	3-nearest neighbors	True class	Predicted class
1	3, 4, 5	1	2
2	1, 5, 6	2	1
3	1, 4, 7	2	2
4	1, 3, 7	2	2
5	1, 2, 6	1	1
6	1, 2, 5	1	1
7	1, 3, 4	2	2

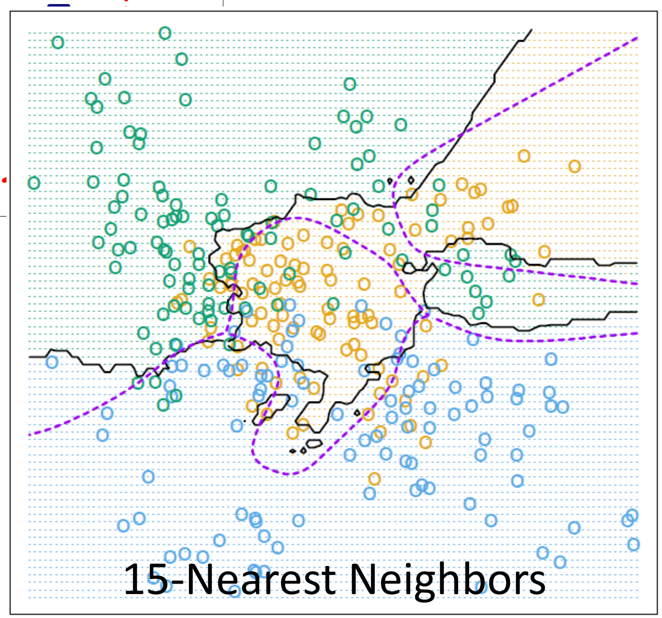
$\mathbf{x}_1 = (6, 7)^\top \quad \mathbf{x}_2 = (4, 2)^\top$

Predict the classes of $\mathbf{x}_1$ and $\mathbf{x}_2$.

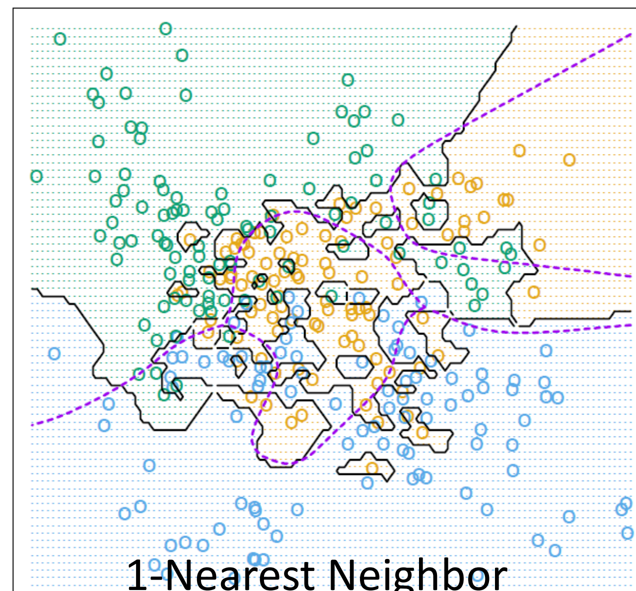
Number of Nearest Neighbors (k)

- **Choosing the value of k:**
  - If k is too small, sensitive to noise points.
  - If k is too large, neighborhood may include points from other classes.





15-Nearest Neighbors



1-Nearest Neighbor

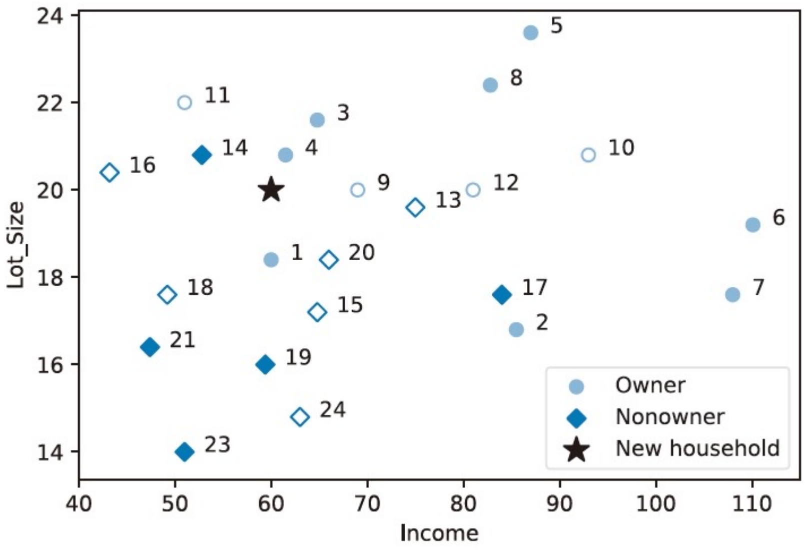
Other issues

- **Scaling issues**
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M
- **k-NN classifiers are lazy learners**
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction
- **Finding nearest neighbors is computationally expensive.**
  - Classifying unknown records are relatively expensive
- **Curse of Dimensionality**

Programming: k-NN Classifier

```
mower_df = pd.read_csv('RidingMowers.csv')
mower_df['Number'] = mower_df.index + 1

trainData, validData = train_test_split(mower_df, test_size=0.4, random_state=26)

## new household
newHousehold = pd.DataFrame([{'Income': 60, 'Lot_Size': 20}])

## scatter plot
def plotDataset(ax, data, showLabel=True, **kwargs):
    subset = data.loc[data['Ownership']=='Owner']
    ax.scatter(subset.Income, subset.Lot_Size, marker='o',
        label='Owner' if showLabel else None, color='C1', **kwargs)
    subset = data.loc[data['Ownership']=='Nonowner']
    ax.scatter(subset.Income, subset.Lot_Size, marker='D',
        label='Nonowner' if showLabel else None, color='C0', **kwargs)
    plt.xlabel('Income') # set x-axis label
    plt.ylabel('Lot_Size') # set y-axis label
    for _, row in data.iterrows():
        ax.annotate(row.Number, (row.Income + 2, row.Lot_Size))

fig, ax = plt.subplots()
plotDataset(ax, trainData)
plotDataset(ax, validData, showLabel=False, facecolors='none')
ax.scatter(newHousehold.Income, newHousehold.Lot_Size, marker='*',
    label='New household', color='black', s=150)

plt.xlabel('Income'); plt.ylabel('Lot_Size')
ax.set_xlim(40, 115)
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, labels, loc=4)
plt.show()
```

- Use “Riding Mowers: dataset
- Draw a scatter plot



Programming: k-NN Classifier

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html

- Scale Data
- Find NNs

```
predictors = ['Income', 'Number]
outcome =

# initialize normalized training, validation, and complete data frames
# use the training data to learn the transformation.
scaler = preprocessing.StandardScaler()
scaler.fit(trainData[['Income', 'Lot_Size']]) # Note use of array of column names

# Transform the full dataset
mowerNorm = pd.concat([pd.DataFrame(scaler.transform(mower_df[['Income', 'Lot_Size']]),
                                    columns=['zIncome', 'zLot_Size']),
                       mower_df[['Ownership', 'Number']]], axis=1)
trainNorm = mowerNorm.iloc[trainData.index]
validNorm = mowerNorm.iloc[validData.index]
newHouseholdNorm = pd.DataFrame(scaler.transform(newHousehold),
    columns=['zIncome', 'zLot_Size'])

# use NearestNeighbors from scikit-learn to compute knn
from sklearn.neighbors import NearestNeighbors
knn = NearestNeighbors(n_neighbors=3)
knn.fit(trainNorm.iloc[:, 0:2])
distances, indices = knn.kneighbors(newHouseholdNorm)

# indices is a list of lists, we are only interested in the first element
trainNorm.iloc[indices[0], :]
```

	zIncome	zLot_Size	Ownership	Number
3	-0.409776	0.743358	Owner	4
13	-0.804953	0.743358	Nonowner	14
0	-0.477910	-0.174908	Owner	1

Programming: k-NN Classifier

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

- Fit k-NN classifiers for different k's

```
train_X = trainNorm[['zIncome', 'zLot_Size']]
train_y = trainNorm['Ownership']
valid_X = validNorm[['zIncome', 'zLot_Size']]
valid_y = validNorm['Ownership']

# Train a classifier for different values of k
results = []
for k in range(1, 15):
    knn = KNeighborsClassifier(n_neighbors=k).fit(train_X, train_y)
    results.append({
        'k': k,
        'accuracy': accuracy_score(valid_y, knn.predict(valid_X))
    })

# Convert results to a pandas data frame
results = pd.DataFrame(results)
print(results)
```

	k	accuracy	
0	1	0.6	
1	2	0.7	
2	3	0.8	
3	4	0.9	<==
4	5	0.7	
5	6	0.9	
6	7	0.9	
7	8	0.9	
8	9	0.9	
9	10	0.8	
10	11	0.8	
11	12	0.9	
12	13	0.4	
13	14	0.4	