Machine Learning and Data Mining

Nearest neighbor methods

Prof. Alexander Ihler
Supervised learning

- Notation
  - Features $x$
  - Targets $y$
  - Predictions $\hat{y}$
  - Parameters $\theta$

Program ("Learner")
Characterized by some "parameters" $\theta$
Procedure (using $\theta$) that outputs a prediction

Learning algorithm
Change $\theta$
Improve performance

Training data (examples)
Features
Feedback / Target values

Score performance ("cost function")
Regression; Scatter plots

- Suggests a relationship between x and y
- Regression: given new observed $x^{(new)}$, estimate $y^{(new)}$
Nearest neighbor regression

Find training datum $x^{(i)}$ closest to $x^{(\text{new})}$; predict $y^{(i)}$

“Predictor”:
Given new features: Find nearest example Return its value
Nearest neighbor regression

- Find training datum $x^{(i)}$ closest to $x^{(new)}$; predict $y^{(i)}$
- Defines an (implicit) function $f(x)$
- "Form" is piecewise constant

“Predictor”:
Given new features:
Find nearest example
Return its value
Nearest neighbor classifier

“Predictor”:
Given new features:
Find nearest example
Return its value
Nearest neighbor classifier

“Predictor”:
Given new features:
Find nearest example
Return its value

“Closest” training $x$?
Typically Euclidean distance:

$$d(x, x') = \sqrt{\sum_i (x_i - x'_i)^2}$$
Nearest neighbor classifier

All points where we decide 1

Decision Boundary

All points where we decide 0

X₁ →

X₂ →
Nearest neighbor classifier

Voronoi tessellation: Each datum is assigned to a region, in which all points are closer to it than any other datum

Decision boundary: Those edges across which the decision (class of nearest training datum) changes
Nearest neighbor classifier

Nearest Nbr:
Piecewise linear boundary
More Data Points
In general: Nearest-neighbor classifier produces piecewise linear decision boundaries.
Machine Learning and Data Mining

Nearest neighbor methods:
K-Nearest Neighbors

Prof. Alexander Ihler
K-Nearest Neighbor (kNN) Classifier

- Find the k-nearest neighbors to $x$ in the data
  - i.e., rank the feature vectors according to Euclidean distance
  - select the $k$ vectors which are have smallest distance to $x$

- Regression
  - Usually just average the $y$-values of the $k$ closest training examples

- Classification
  - ranking yields $k$ feature vectors and a set of $k$ class labels
  - pick the class label which is most common in this set (“vote”)
  - classify $x$ as belonging to this class
  - Note: for two-class problems, if $k$ is odd ($k=1, 3, 5, \ldots$) there will never be any “ties”

- “Training” is trivial: just use training data as a lookup table, and search to classify a new datum
kNN Decision Boundary

- Piecewise linear decision boundary
- Increasing $k$ “simplifies” decision boundary
  - Majority voting means less emphasis on individual points

$K = 1$  \hspace{1cm}  $K = 3$
kNN Decision Boundary

- Recall: piecewise linear decision boundary
- Increasing $k$ “simplifies” decision boundary
  - Majority voting means less emphasis on individual points

$K = 5$

$K = 7$
kNN Decision Boundary

- Recall: piecewise linear decision boundary
- Increasing $k$ “simplifies” decision boundary
  - Majority voting means less emphasis on individual points
Error rates and $K$

<table>
<thead>
<tr>
<th>$K$</th>
<th>Error on Training Data</th>
<th>Error on Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Zero error!</td>
<td>Training data have been memorized...</td>
</tr>
</tbody>
</table>

Best value of $K$
Complexity & Overfitting

- Complex model predicts all training points well
- Doesn’t generalize to new data points
- $K=1$ : perfect memorization of examples (complex)
- $K=M$ : always predict majority class in dataset (simple)
- Can select $K$ using validation data, etc.

Too complex

simpler

$K$ (# neighbors)

K (# neighbors)
K-Nearest Neighbor (kNN) Classifier

• Theoretical Considerations
  – as k increases
    • we are averaging over more neighbors
    • the effective decision boundary is more “smooth”
  – as N increases, the optimal k value tends to increase
  – k=1, m increasing to infinity: error < 2x optimal

• Extensions of the Nearest Neighbor classifier
  – weighted distances
    • e.g., if some of the features are more important
    • e.g., if features are irrelevant
    \[ d(x, x') = \sqrt{\sum_i w_i(x_i - x'_i)^2} \]
  – fast search techniques (indexing) to find k-nearest neighbors in d-space
Summary

• K-nearest neighbor models
  – Classification (vote)
  – Regression (average or weighted average)

• Piecewise linear decision boundary
  – How to calculate

• Test data and overfitting
  – Model “complexity” for knn
  – Use validation data to estimate test error rates & select k