Classification: K-Nearest Neighbors

3/27/17
Recall: Machine Learning Taxonomy

Supervised Learning
- For each input, we know the right output.
- Regression
  - Outputs are continuous.
- Classification
  - Outputs come from a (relatively small) discrete set.

Unsupervised Learning
- We just have a bunch of inputs.

Semi-Supervised Learning
- We have inputs, and occasional feedback.
Labeling the city an apartment is in.  

Labeling hand-written digits.
Hypothesis Space for Classification

• The hypothesis space is the types of functions we can learn.
  • This is partly defined by the problem, and partly by the learning algorithm.

• In classification we have:
  • Continuous inputs
  • Discrete output labels

• The algorithm will constrain the possible functions from input to output.
  • Perceptrons learn linear decision boundaries.
K-nearest neighbors algorithm

Training:
• Store all of the test points and their labels.
• Can use a data structure like a kd-tree that speeds up localized lookup.

Prediction:
• Find the k training inputs closest to the test input.
• Output the most common label among them.
KNN implementation decisions

(And possible answers)

• How should we measure distance?
  • (Euclidean distance between input vectors.)

• What if there’s a tie for the nearest points?
  • (Include all points that are tied.)

• What if there’s a tie for the most-common label?
  • (Remove the most-distant point until a plurality is achieved.)

• What if there’s a tie for both?
  • (We need some arbitrary tie-breaking rule.)
Weighted nearest neighbors

• Idea: closer points should matter more.

• Solution: weight the vote by \( \frac{1}{distance + c} \)

• Instead of contributing one vote for its label, each neighbor contributes \( \frac{1}{distance + c} \) votes for its label.
Why do we even need k neighbors?

Idea: if we’re weighting by distance, we can give all training points a vote.
• Points that are far away will just have really small weight.

Why might this be a bad idea?
• Slow: we have to sum over every point in the training set.
• If we’re using a kd-tree, we can get the neighbors quickly and sum over a small set.
The same ideas can apply to regression.

• **K-nearest neighbors setting:**
  • Supervised learning (we know the correct output for each test point).
  • Classification (small number of discrete labels).

vs.

• **Locally-weighted regression setting:**
  • Supervised learning (we know the correct output for each test point).
  • Regression (outputs are continuous).
Locally-Weighted Average

• Instead of taking a majority vote, average the y-values.

• We could average over the k nearest neighbors.

• We could weight the average by distance.

• Better yet, do both.

\[
\begin{align*}
\hat{f}(x_q) & \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i} \\
 w_i & \equiv \frac{1}{d(x_q, x_i)^2}
\end{align*}
\]
Locally-weighted (linear) regression

Least squares linear regression solves the following problem:

- Select weights weights $w_0, \ldots, w_D$ for each dimension to minimize squared error:

$$\hat{f}(x) = w_0 + w_1 x_1 + \ldots + w_D x_D \quad E = \sum_{x \in \text{training set}} \left( \hat{f}(x) - f(x) \right)^2$$

Instead, we can minimize the distance-weighted squared error:

$$E = \sum_{x \in \text{training set}} \frac{\left( \hat{f}(x) - f(x) \right)^2}{\text{distance}(x) + c}$$
Decision Trees

- Solve classification problems by repeatedly splitting the space of possible inputs; store splits in a tree.
- To classify a new input, compare it to successive splits until a leaf (with a label) is reached.

Who plays tennis when it’s raining but not when it’s humid?
Building a Decision Tree

Greedy algorithm:

1. Within a region, pick the best:
   • feature to split on
   • value at which to split it

2. Sort the training data into the sub-regions.

3. Recursively build decision trees for the sub-regions.

Does this give us an optimal decision tree?
Compare the Hypothesis Spaces

• K-nearest neighbors

• Decision trees

• Locally-weighted regression

Considerations:
• Inputs
• Outputs
• Possible mappings