Decision Trees and Naïve Bayes

3/29/17
Hypothesis Spaces

• Decision Trees and K-Nearest Neighbors
  • Continuous inputs
  • Discrete outputs

• Naïve Bayes
  • Discrete inputs
  • Discrete outputs
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.
Picking the Best Split

Key idea: minimize entropy

- $S$ is a collection of positive and negative examples
- $\text{Pos}$: proportion of positive examples in $S$
- $\text{Neg}$: proportion of negative examples in $S$

Entropy$(S) = -\text{Pos} \times \log_2(\text{Pos}) - \text{Neg} \times \log_2(\text{Neg})$

- Entropy is 0 when all members of $S$ belong to the same class, for example: $\text{Pos} = 1$ and $\text{Neg} = 0$
- Entropy is 1 when $S$ contains equal numbers of positive and negative examples: $\text{Pos} = \frac{1}{2}$ and $\text{Neg} = \frac{1}{2}$
Searching for the Best Split

Try all features. Try all possible splits of that feature. If feature F is _______, there are _________ possible splits to consider.

- binary ... one
- discrete and ordered ... $|F| - 1 \quad \bullet|\bullet|\bullet|\bullet|\bullet$
- discrete and unordered ... $2|F| - 1 - 1$
  - (two options for where to put each value)
- continuous ... $| \text{training set} | - 1$
  - (any split between two points is the same)
Can we do better?

Try all features. Try all possible splits of that feature. If feature F is ________, there are __________ possible splits to consider.

- binary ... one
- discrete and ordered ... $|F| - 1$ \[ \bullet | \bullet | \bullet | \bullet \]
- discrete and unordered ... $2|F| - 1 - 1$
  - (two options for where to put each value)
- continuous ... $|\text{training set}| - 1$
  - (any split between two points is the same)

How can we avoid trying all possible splits?

Binary Search

Local Search
When do we stop splitting?

Bad idea:
• When every training point is classified correctly.
• Why is this a bad idea?
  • Overfitting

Better idea:
• Stop at some limit on depth, #points, or entropy
• How should we choose the limit?
  • Training/test split
  • Cross validation (more on Friday)
Bayesian Approach to Classification

Key idea: use training data to estimate a probability for each label given an input.

\[ P(label \mid input) \]

Classify a point as its highest-probability label.
Estimating Probabilities from Data

Suppose we flip a coin 10 times and observe:

7 3

What do we believe to be the true $P(\text{H})$?

Now suppose we flip it 1000 times and observe:

700 300
Prior Probability

We need to combine our initial beliefs with data.

Empirical frequency:

$$\frac{\text{observations of outcome}}{\text{total observations}} = \frac{n_H}{n}$$

Prior for a coin toss:

$$P(H) = \frac{1}{2}$$

Add $m$ “observations” of the prior to the data:

$$P(H) = \frac{n_H + \frac{1}{2}m}{n + m}$$
Estimating Label Probabilities

• We want to compute \( P(label \mid input) \)
  • Conditional on a particular input point what is the probability of each label?

• Estimating this empirically requires many observations of every possible input.

• In such a case, we aren’t really learning: there’s no generalization to new data.

• We want to generalize from many training points to get estimate a probability at an unobserved test point.
The Naïve Part of Naïve Bayes

Assume that all features are independent.

This lets us estimate probabilities for each feature separately, then multiply them together:

\[ P(l \mid x) = P(l \mid x_1)P(l \mid x_2) \ldots P(l \mid x_n) \]

This assumption is almost never literally true, but makes the estimation feasible and often gives a good enough classifier.
Empirical Probabilities for Classification

Given a data set consisting of

- Inputs \( x = (x_1, x_2, x_3) \)
- Labels \( l \)

For each possible value of \( x_i \) and \( l \), we can compute an empirical frequency:

\[
P(x_1 = 2 \mid l = +1) = \frac{\text{count}(x_1 = 2 \land l = +1)}{\text{count}(l = +1)}
\]
Bayes Rule

- We can empirically estimate $P(x \mid l)$
- But we actually want $P(l \mid x)$
- We can get it using Bayes rule:

\[ P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} \]
Bayes Rule Applied

To compute \( P(l | x_1) = \frac{P(x_1 | l) \cdot P(l)}{P(x_1)} \) from our data, we need to estimate two more quantities from data:

- \( P(x_1) \)
- \( P(l) \)

- This means doing additional empirical estimates across our data set for each possible value of each input dimension and the label.
Naïve Bayes Training

• We need to estimate the probability of each value for each dimension
  • For example: \( P(x_1 = 5) \)

• We need to estimate the probability of each label
  • For example: \( P(l = +1) \)

• We need to estimate the probability of each value for each dimension conditional on each label
  • For example: \( P(x_1 = 5 \mid l = -1) \)

All of these are estimated empirically, with some prior (usually uniform).
Naïve Bayes Prediction

Given a new input: \( x = (x_1, x_2, x_3) \)

Compute for each possible label: \( P(label = l \mid x) \)

Using the naïve assumption this is estimated as:

\[
P(l \mid x) = \frac{P(l)P(x_1 \mid l)P(x_2 \mid l)P(x_3 \mid l)}{P(x)}
\]

Return the highest-probability label.