Artificial Intelligence Programming

Bayesian Learning

Chris Brooks

Department of Computer Science
University of San Francisco

19-2: Learning and Classification

- An important sort of learning problem is the *classification* problem.
- This involves placing examples into one of two or more classes.
  - Should/shouldn't get credit
  - Categories of documents.
  - Golf-playing/non-golf-playing days
- This requires access to a set of *labeled* training examples, which allow us to induce a hypothesis that describes how to decide what class an example should be in.

19-3: Bayes' Theorem

- Recall the definition of Bayes' Theorem
- $P(b|a) = \frac{P(a|b)P(b)}{P(a)}$
- Let's rewrite this a bit.
- Let D be the data we've seen so far.
- Let h be a possible hypothesis
- $P(h|D) = \frac{P(D|h)P(h)}{P(D)}$

19-4: MAP Hypothesis

- Usually, we won't be so interested in the particular probabilities for each hypothesis.
- Instead, we want to know: Which hypothesis is most likely, given the data?
  - Which classification is the most probable?
  - Is $PlayTennis$ or $\neg PlayTennis$ more likely?
- We call this the *maximum a posteriori hypothesis* (MAP hypothesis).
- In this case, we can ignore the denominator in Bayes' Theorem, since it will be the same for all h.
- $h_{MAP} = argmax_{h \in H} P(D|h)P(h)$
- Advantages:
  - Simpler calculation
  - No need to have a prior for $P(D)$

19-5: ML Hypothesis

- In some cases, we can simplify things even further.
- What are the priors $P(h)$ for each hypothesis?
- Without any other information, we'll often assume that they're equally possible.
  - Each has probability $\frac{1}{H}$
- In this case, we can just consider the conditional probability $P(D|h)$.
- We call the hypothesis that maximizes this conditional probability the *maximum likelihood* hypothesis.
- $h_{ML} = argmax_{h \in H} P(D|h)$

19-6: Example

- Imagine that we have a large bag of candy. We want to know the ratio of cherry to lime in the bag.
- We start with 5 hypotheses:
  1. h_1: 100% cherry
  2. h_2 75% cherry, 25% lime.
  3. h_3 50% cherry, 50% lime
  4. h_4 25% cherry, 75% lime
  5. h_5 100% lime
- Our agent repeatedly draws pieces of candy.
- We want it to correctly pick the type of the next piece of candy.

19-7: Example

- Let's assume our priors for the different hypotheses are:
- $(0.1, 0.2, 0.4, 0.2, 0.1)$
- Also, we assume that the observations are i.i.d.
  - This means that each choice is independent of the others. (order doesn't matter)
- In that case, we can multiply probabilities.
- $P(D|h_i) = \Pi_j P(d_j|h_i)$
- Suppose we draw 10 limes in a row. $P(D|h_3)$ is $(\frac{1}{2})^{10}$, since the probability of drawing a lime under h_3 is $\frac{1}{2}$.

19-8: Example

- How do the hypotheses change as data is observed?
- Initially, we start with the priors: $(0.1, 0.2, 0.4, 0.2, 0.1)$
- Then we draw a lime.
  - $P(h_1|lime) = \alpha P(lime|h_1)P(h_1) = 0.$
  - $P(h_2|lime) = \alpha P(lime|h_2)P(h_2) = \alpha\frac{1}{4} * 0.2 = \alpha 0.05.$
  - $P(h_3|lime) = \alpha P(lime|h_3)P(h_3) = \alpha\frac{1}{2} * 0.4 = \alpha 0.2$
  - $P(h_4|lime) = \alpha P(lime|h_4)P(h_4) = \alpha\frac{3}{4} * 0.2 = \alpha 0.15.$
  - $P(h_5|lime) = \alpha P(lime|h_5)P(h_5) = \alpha 1 * 0.1 = \alpha 0.1.$
  - $\alpha = 2.$

19-9: Example

- Then we draw a second lime.
  - $P(h_1|lime, lime) = \alpha P(lime, lime|h_1)P(h_1) = 0.$
  - $P(h_2|lime, lime) = \alpha P(lime, lime|h_2)P(h_2) = \alpha\frac{1}{4}\frac{1}{4} * 0.2 = \alpha 0.0125.$
  - $P(h_3|lime, lime) = \alpha P(lime, lime|h_3)P(h_3) = \alpha\frac{1}{2}\frac{1}{2} * 0.4 = \alpha 0.1$
  - $P(h_4|lime, lime) = \alpha P(lime, lime|h_4)P(h_4) = \alpha\frac{3}{4}\frac{3}{4} * 0.2 = \alpha 0.1125.$
  - $P(h_5|lime) = \alpha P(lime|h_5)P(h_5) = \alpha 1 * 0.1 = \alpha 0.1.$
  - $\alpha = 3.07.$
- Strictly speaking, we don't really care what α is.
- We can just select the MAP hypothesis, since we just want to know the most likely hypothesis.

19-10: Bayesian Learning

- Eventually, the true hypothesis will dominate all others.
  - Caveat: assuming the data is noise-free, or noise is uniformly distributed.
- Notice that we can use Bayesian learning (in this case) either as a batch algorithm or as an incremental algorithm.
- We can always easily update our hypotheses to incorporate new evidence.
  - This depends on the assumption that our observations are independent.

19-11: Learning bias

- What sort of bias does Bayesian Learning use?
- Typically, simpler hypotheses will have larger priors.
- More complex hypotheses will fit data more exactly (but there's many more of them).
  - Under these assumptions, h_{MAP} will be the simplest hypothesis that fits the data.
  - This is Occam's razor, again.
  - Think about the deterministic case, where $P(h_i|D)$ is either 1 or 0.

19-12: Bayesian Concept Learning

- Bayesian Learning involves estimating the likelihood of each hypothesis.
- In a more complex world where observations are not independent, this could be difficult.
- Our first cut at doing this might be a brute force approach:
  1. For each h in H, calculate $P(h|D) = \frac{P(D|h)P(h)}{P(D)}$
  2. From this, output the hypothesis h_{MAP} with the highest posterior probability.
- This is what we did in the example.
  - Challenge - Bayes' Theorem can be computationally expensive to use in the case where observations are not i.i.d.
  - $P(h|o_1, o_2) = \frac{P(o_1|h, o_2)P(h|o_2)}{P(o_1|o_2)}$

19-13: Bayesian Optimal Classifiers

- There's one other problem with the formulation as we have it.
- Usually, we're not so interested in the hypothesis that fits the data.
- Instead, we want to classify some unseen data, given the data we've seen so far.
- One approach would be to just return the MAP hypothesis.
- We can do better, though.

19-14: Bayesian Optimal Classifiers

- Suppose we have three hypotheses and posteriors: $h_1 = 0.4, h_2 = 0.3, h_3 = 0.3.$
- We get a new piece of data - h_1 says it's positive, h_2 and h_3 negative.
- h_1 is the MAP hypothesis, yet there's a 0.6 chance that the data is negative.
- By combining weighted hypotheses, we improve our performance.

19-15: Bayesian Optimal Classifiers

- By combining the predictions of each hypothesis, we get a Bayesian optimal classifier.
- More formally, let's say our unseen data belongs to one of v classes.
- The probability $P(v_j|D)$ that our new instance belongs to class v_j is:
- $\sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$
- Intuitively, each hypothesis gives its prediction, weighted by the likelihood that that hypothesis is the correct one.
- This classification method is provably optimal - on average, no other algorithm can perform better.

19-16: Problems with the Bayes Optimal classifier

- However, the Bayes optimal classifier is mostly interesting as a theoretical benchmark.
- In practice, computing the posterior probabilities is exponentially hard.
- This problem arises when instances or data are conditionally dependent upon each other.
- Can we get around this?

19-17: Naive Bayes classifier

- The Naive Bayes classifer makes a strong assumption that makes the algorithm practical:
  - Each attribute of an example is independent of the others.
  - $P(a \wedge b) = P(a)P(b)$ for all a and b.
- This makes it straightforward to compute posteriors.

19-18: The Bayesian Learning Problem

- Given: a set of labeled, multivalued examples.
- Find a function $F(x)$ that correctly classifies an unseen example with attributes $(a_1, a_2, ..., a_n)$.
- Call the most probable category v_{map}.
- $v_{map} = argmax_{v_i \in V} P(v_i|a_1, a_2, ..., a_n)$
- We rewrite this with Bayes' Theorem as: $v_{map} = argmax_{v_i \in V} P(a_1, a_2, ..., a_n|v_i)P(v_i)$
- Estimating $P(v_i)$ is straightforward with a large training set; count the fraction of the set that are of class v_i.
- However, estimating $P(a_1, a_2, ..., a_n|v_i)$ is difficult unless our training set is *very* large. We need to see every possible attribute combination many times.

19-19: Naive Bayes assumption

- Naive Bayes assumes that all attributes are conditionally independent of each other.
- In this case, $P(a_1, a_2, ..., a_n|v_i) = \Pi_i P(a_i|v_i)$.
- This can be estimated from the training data.
- The classifier then picks the class with the highest probability according to this equation.
- Interestingly, Naive Bayes performs well even in cases where the conditional independence assumption fails.

19-20: Example

- Recall your tennis-playing problem from the decision tree homework.
- We want to use the training data and a Naive Bayes classifier to classify the following instance:
- Outlook = Sunny, Temperature = Cool, Humidity = high, Wind = Strong.

19-21: Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

19-22: Example

- Our priors are:
  - $P(PlayTennis = yes) = 9/14 = 0.64$
  - $P(PlayTennis = no) = 5/14 = 0.36$
- We can estimate:
  - $P(wind = strong|PlayTennis = yes) = 3/9 = 0.33$
  - $P(wind = strong|PlayTennis = no) = 3/5 = 0.6$
  - $P(humidity = high|PlayTennis = yes) = 3/9 = 0.33$
  - $P(humidity = high|PlayTennis = no) = 4/5 = 0.8$
  - $P(outlook = sunny|PlayTennis = yes) = 2/9 = 0.22$
  - $P(outlook = sunny|PlayTennis = no) = 3/5 = 0.6$
  - $P(temp = cool|PlayTennis = yes) = 3/9 = 0.33$
  - $P(temp = cool|PlayTennis = no) = 1/5 = 0.2$

19-23: Example

- $v_{yes} = P(yes)P(sunny|yes)P(cool|yes)P(high|yes)P(strong|yes) = 0.005$
- $v_{no} = P(no)P(sunny|no)P(cool|no)P(high|no)P(strong|no) = 0.0206$
- So we see that not playing tennis is the maximum likelihood hypothesis.
- Further, by normalizing, we see that the classifier predicts a $\frac{0.0206}{0.005+0.0206} = 0.80$ probability of not playing tennis.

19-24: Estimating Probabilities

- As we can see from this example, estimating probabilities through frequency is risky when our data set is small.
- We only have 5 negative examples, so we may not have an accurate estimate.
- A better approach is to use the following formula, called an m-estimate:
- $\frac{n_c+mp}{n+m}$
- Where n_c is the number of individuals with the characteristic of interest (say Wind = strong), n is the total number of positive/negative examples, p is our prior estimate, and m is a constant called the *equivalent sample size*.

19-25: Estimating Probabilities

- m determines how heavily to weight p.
- p is assumed to be uniform.
- So, in the Tennis example, $P(wind = strong|playTennis = no) = \frac{3+0.2m}{5+m}$
- We'll determine an m based on sample size.
  - If m is zero, we just use observed data.
  - If $m >> n$, we use the prior.
  - Otherwise m lets us weight these parameters' relative influence.

19-26: Using Naive Bayes to classify text

- One are where Naive Bayes has been very successful is in text classification.
  - Despite the violation of independence assumptions.
- Problem - given some text documents labeled as to category, determine the category of new and unseen documents.
- Our features will be the words that appear in a document.
- Based on this, we'll predict a category.

19-27: Using Naive Bayes to classify text

- The first thing we need is some prior probabilities for words.
- We get this by scanning the entire training set and counting the frequency of each word.
  - Common, content-free words such as a, an, the, he, she, etc. are often left out. (these are called *stop words*.)
- This word frequency count gives us an estimate of the priors for each word.

19-28: Using Naive Bayes to classify text

- We then need priors for each category.
  - We get this by counting the fraction of the training set that belong to a given category.
- Finally, we need to determine the conditional probability $P(w_k|v_j)$.
  - We do this by concatenating all documents in class v_j. Call this the Text.
  - Then, for each word in the vocabulary, we count how often it occurs in the Text. Call this n_k
  - Let n be the total number of words in Text.
  - $P(w_k|v_j) = \frac{n_k+1}{n+|Vocabulary|}$
  - m is the size of the vocabulary.

19-29: Classifying text

- Once we've computed the conditional probabilities, we can classify text.
- For a new document, find all distinct words in the document.
- For each class of document, compute the posterior probability of that class, given the words present in the document.
- Seems too simple, but it actually works. (Up to 90% accurate in some cases.)
- There are *a lot* of details that can drastically affect performance. You will get to see these firsthand in project 3.