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.

Bayes’ Theorem

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

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 ¬PlayTennis more likely?
- We call this the maximum a posterior hypothesis (MAP hypothesis).
- In this case, we can ignore the denominator in Bayes’ Theorem, since it will be the same for all \( h \).
- \( h_{MAP} = \arg\max_{h \in H} P(D|h)P(h) \)

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.
- \( h_{ML} = \arg\max_{h \in H} P(D|h) \)
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.

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.

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, \( h_{\text{MAP}} \) 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.
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 when observations are not i.i.d.

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.

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.

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:
  $$P(v_j|D) = \frac{\sum_{h \in H} P(v_j|h) P(h|D)}{P(D)}$$
- 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.

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?

Naive Bayes classifier

- The Naive Bayes classifier makes a strong assumption that makes the algorithm practical:
  1. Each attribute of an example is independent of the others.
  2. $P(a \land b) = P(a)P(b)$ for all $a$ and $b$.
- This makes it straightforward to compute posteriors.
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 $v_{map}$:
  \[ v_{map} = \text{argmax}_{v \in \mathcal{V}} P(v | a_1, a_2, ..., a_n) \]
- We rewrite this with Bayes' Theorem as:
  \[ v_{map} = \text{argmax}_{v \in \mathcal{V}} \frac{P(v) P(a_1, a_2, ..., a_n | v)}{P(a_1, a_2, ..., a_n)} \]

Naive Bayes assumption

- Naive Bayes assumes that all attributes are conditionally independent of each other.
- In this case, $P(a_1, a_2, ..., a_n | v_i) = \Pi P(a_i | v_i)$.
- 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.

Example

- Recall your tennis-playing problem from the decision tree homework.
- Find a function $F(x)$ that correctly classifies an unseen instance $(a_1, a_2, ..., a_n)$.
- In this case, $(a_2, a_3, ..., a_n | v_i) = \Pi P(a_i | v_i)$.
- This can be estimated from the training data.
- Call the most probable category $v_{map}$.
- We want to use the training data and a Naive Bayes classifier to classify the following instance: $(a_1, a_2, ..., a_n)$.
- 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.

### Example

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
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:
  \[ \frac{n_c + m}{n + m} \]
  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.

Using Naive Bayes to classify spam

- For a given email, we'll want to compute the MAP hypothesis - that is, is:
  \[ P(\text{spam}|t_1, t_2, \ldots, t_n) \text{ greater than } P(\text{ham}|t_1, t_2, \ldots, t_n) \]
- We can use Bayes' rule to rewrite these as:
  \[ \alpha P(t_1, t_2, \ldots, t_n|\text{spam}) P(\text{spam}) \]
  \[ \alpha P(t_1, t_2, \ldots, t_n|\text{ham}) P(\text{ham}) \]

Using Naive Bayes to classify spam

- We can then use the Naive Bayes assumption to rewrite these as:
  \[ \alpha P(t_1|\text{spam}) P(t_2|\text{spam}) \cdots P(t_n|\text{spam}) P(\text{spam}) \]
  \[ \alpha P(t_1|\text{ham}) P(t_2|\text{ham}) \cdots P(t_n|\text{ham}) P(\text{ham}) \]
- And this we know how to compute.

Using Naive Bayes to classify spam

- One area where Naive Bayes has been very successful is in text classification.
- Despite the violation of independence assumptions.
- Classifying spam is just a special case of text classification.
- Problem - given some emails labeled ham or spam, determine the category of new and unseen documents.
- Our features will be the tokens that appear in a document.
- Based on this, we'll predict a category.
Using Naive Bayes to classify spam

There are a lot of wrinkles to consider:

- What should be treated as a token? All words? All strings? Only some words?
- Should headers be given different treatment? Greater or less emphasis?
- What about HTML?
- When classifying an email, should you consider all tokens, or just the most significant?
- When computing conditional probabilities, should you could the fraction of documents a token appear in, or the fraction of words represented by a particular token?

These are for you to decide ...