Machine Learning is... 

Machine learning is about predicting the future based on the past.  
-- Hal Daume III

Probabilistic Modeling

Model the data with a probabilistic model.  
specifically, learn $p(\text{features}, \text{label})$  
$p(\text{features}, \text{label})$ tells us how likely these features and this example are
An example: classifying fruit

Training data

<table>
<thead>
<tr>
<th>examples</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>red, round, leaf, 3oz, ...</td>
<td>apple</td>
</tr>
<tr>
<td>green, round, no leaf, 4oz, ...</td>
<td>apple</td>
</tr>
<tr>
<td>yellow, curved, no leaf, 4oz, ...</td>
<td>banana</td>
</tr>
<tr>
<td>green, curved, no leaf, 5oz, ...</td>
<td>banana</td>
</tr>
</tbody>
</table>

Probabilistic models

Probabilistic models define a probability distribution over features and labels:

- yellow, curved, no leaf, 6oz, banana: 0.004

Probabilistic model vs. classifier

Probabilistic model:

- yellow, curved, no leaf, 6oz: 0.004

Classifier:

- yellow, curved, no leaf, 6oz: banana

Probabilistic models: classification

Probabilistic models define a probability distribution over features and labels:

Given an unlabeled example: yellow, curved, no leaf, 6oz, predict the label.

How do we use a probabilistic model for classification/prediction?
Probabilistic models define a probability distribution over features and labels:

- yellow, curved, no leaf, 6oz, banana
- yellow, curved, no leaf, 6oz, apple

For each label, ask for the probability under the model. Pick the label with the highest probability.

Probabilistic model vs. classifier

- Probabilistic model:
  - yellow, curved, no leaf, 6oz, banana: 0.004
  - yellow, curved, no leaf, 6oz, apple: 0.00002

- Classifier:
  - yellow, curved, no leaf, 6oz, banana

Why probabilistic models?

Probabilistic models: big questions

1. Which model do we use, i.e., how do we calculate $p(\text{feature, label})$?
2. How do we train the model, i.e., how do we estimate the probabilities for the model?
3. How do we deal with overfitting (i.e., smoothing)?

Probabilistic models

Probabilities are nice to work with:

- range between 0 and 1
- can combine them in a well understood way
- lots of mathematical background/theory

Provide a strong, well-founded groundwork:

- Allow us to make clear decisions about things like smoothing
- Tend to be much less “heuristic”
- Models have very clear meanings
Basic steps for probabilistic modeling

Step 1: pick a model

Probabilistic models
Which model do we use, i.e. how do we calculate $p(\text{feature, label})$?

How do train the model, i.e. how do we estimate the probabilities for the model?

How do we deal with overfitting?

Step 2: figure out how to estimate the probabilities for the model

Step 3 (optional): deal with overfitting

What was the data generating distribution?

Step 1: picking a model

What we’re really trying to do is model the data generating distribution, that is how likely the feature/label combinations are

Some math

$p(\text{features, label}) = p(x_1, x_2, ..., x_n, y)$

$= p(y)p(x_1, x_2, ..., x_n \mid y)$

What rule?
Some math

\[p(\text{features}, \text{label}) = p(x_1, x_2, \ldots, x_m, y)\]

\[= p(y)p(x_1, x_2, \ldots, x_m | y)\]

\[= p(y)p(x_1 | y)p(x_2, \ldots, x_m | y, x_1)\]

\[= p(y)p(x_1 | y)p(x_2 | y, x_1)p(x_3, \ldots, x_m | y, x_1, x_2)\]

\[= p(y) \prod_{j=1}^m p(x_j | y, x_1, \ldots, x_{m-1})\]

Step 1: pick a model

\[p(\text{features}, \text{label}) = p(y) \prod_{j=1}^m p(x_j | y, x_1, \ldots, x_{m-1})\]

So, far we have made NO assumptions about the data

\[p(x_m | y, x_1, x_2, \ldots, x_{m-1})\]

How many entries would the probability distribution table have if we tried to represent all possible values and we had 7000 binary features?

Full distribution tables

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(\ldots)</th>
<th>(y)</th>
<th>(p())</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(\ldots)</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(\ldots)</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>(\ldots)</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>(\ldots)</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>(\ldots)</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>(\ldots)</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
</tr>
</tbody>
</table>

All possible combination of features!

Table size: \(2^{7000}\) = ?

Any problems with this?
Storing a table of that size is impossible!
How are we supposed to learn/estimate each entry in the table?

### Step 1: pick a model

\[
p(\text{features}, \text{label}) = p(y) \prod_{j=1}^{m} p(x_j | y, x, \ldots, x_{i-1})
\]

So far we have made NO assumptions about the data.

Model selection involves making assumptions about the data.

We've done this before, n-gram language model, parsing, etc.

These assumptions allow us to represent the data more compactly and to estimate the parameters of the model.

---

### Naive Bayes assumption

\[
p(\text{features}, \text{label}) = p(y) \prod_{j=1}^{m} p(x_j | y, x, \ldots, x_{i-1})
\]

\[
p(x_i | y, x, x_1, \ldots, x_{i-1}) = p(x_i | y)
\]

What does this assume?

Assumes feature i is independent of the other features given the label.

Is this true for text, say, with unigram features?
Naïve Bayes assumption

\[ p(x_i \mid y, x_1, x_2, ..., x_{i-1}) = p(x_i \mid y) \]

For most applications, this is not true!

For example, the fact that “San” occurs will probably make it more likely that “Francisco” occurs.

However, this is often a reasonable approximation:

\[ p(x_i \mid y, x_1, x_2, ..., x_{i-1}) \approx p(x_i \mid y) \]

Naïve Bayes model

\[ p(\text{features, label}) = p(y) \prod_{j=1}^{m} p(x_j \mid y, x_1, x_2, ..., x_{j-1}) \]

\[ = p(y) \prod_{j=1}^{m} p(x_j \mid y) \quad \text{naïve Bayes assumption} \]

\( p(x_i \mid y) \) is the probability of a particular feature value given the label.

How do we model this?
- for binary features (e.g., “banana” occurs in the text)
- for discrete features (e.g., “banana” occurs \( x_i \) times)
- for real valued features (e.g., the text contains \( x_i \) proportion of verbs)

Basic steps for probabilistic modeling

Step 1: Pick a model

Which model do we use, i.e., how do we calculate \( p(\text{feature, label}) \)?

Step 2: Figure out how to estimate the probabilities for the model

How do we train the model, i.e., how to we estimate the probabilities for the model?

Step 3 (optional): Deal with overfitting

How do we deal with overfitting?

Binary features (aka, Bernoulli Naïve Bayes):

\[ p(x_j \mid y) = \begin{cases} \theta_j & \text{if } x_j = 1 \\ 1 - \theta_j & \text{otherwise} \end{cases} \]

biased coin toss!
Obtaining probabilities

Training data → Probabilistic model → \( p(y) \)

\( p(y) \) \( p(x_1 | y) \)

\( p(x_2 | y) \)

\( \vdots \)

\( p(x_m | y) \)

(m = number of features)

MLE estimation for Bernoulli NB

Training data → Probabilistic model → \( p(y) \)

\( p(y) \) \( \prod_{j=1}^m p(x_j | y) \)

What are the MLE estimates for these?

Maximum likelihood estimates

\( p(y) = \frac{\text{count}(y)}{n} \)

Number of examples with label

Total number of examples

\( p(x_j | y) = \frac{\text{count}(x_j, y)}{\text{count}(y)} \)

Number of examples with the label with feature

Number of examples with label

What does training a NB model then involve?

How difficult is this to calculate?

Text classification

Unigram features:

\( w_j \) whether or not word \( w_j \) occurs in the text

\( p(w_j | y) = \frac{\text{count}(w_j, y)}{\text{count}(y)} \)

What are these counts for text classification with unigram features?
Text classification

\[ p(y) = \frac{\text{count}(y)}{n} \]
\[ p(w_j | y) = \frac{\text{count}(w_j, y)}{\text{count}(y)} \]

Naïve Bayes classification

\[ p(y) \prod_{j=1}^{m} p(x_j | y) \]

Given an unlabeled example: yellow, curved, no leaf, 6oz, predict the label

How do we use a probabilistic model for classification/prediction?

\[ \text{label} = \arg\max_{y \in \text{labels}} p(y) \prod_{j=1}^{m} p(x_j | y) \]

Notice that each label has its own separate set of parameters, i.e. \( p(x_j | y) \)
Bernoulli NB for text classification

How good is this model for text classification?

Each word that occurs, contributes $p(w_i | y)$
Each word that does NOT occur, contributes $1 - p(w_i | y)$

Generative Story

To classify with a model, we’re given an example and we obtain the probability.

We can also ask how a given model would generate an example.

This is the “generative story” for a model.

Looking at the generative story can help understand the model.

We also can use generative stories to help develop a model.
Bernoulli NB generative story

\[ p(y) \prod_{j=1}^{m} p(x_j | y) \]

What is the generative story for the NB model?

1. Pick a label according to \( p(y) \)
   - roll a biased, \( \text{num\_labels} \)-sided die
2. For each feature:
   - Flip a biased coin:
     - if heads, include the feature
     - if tails, don't include the feature

What does this mean for text classification, assuming unigram features?

Bernoulli NB generative story

\[ p(y) \prod_{j=1}^{m} p(x_j | y) \]

1. Pick a label according to \( p(y) \)
   - roll a biased, \( \text{num\_labels} \)-sided die
2. For each word in your vocabulary:
   - Flip a biased coin:
     - if heads, include the word in the text
     - if tails, don't include the word

Pros/cons?
Bernoulli NB

Pros
- Easy to implement
- Fast!
- Can be done on large data sets

Cons
- Naive Bayes assumption is generally not true
- Performance isn’t as good as other models
- For text classification (and other sparse feature domains) the \( p(x_i=0 \mid y) \) can be problematic

Another generative story

Randomly draw words from a “bag of words” until document length is reached

Draw words from a fixed distribution

Selected: \( w_1 \)

Selected: \( w_1 \)

Put a copy of \( w_1 \) back
Draw words from a fixed distribution

Selected: $w_1$, $w_1$

Put a copy of $w_1$ back

Draw words from a fixed distribution

Selected: $w_1$, $w_1$

Put a copy of $w_1$ back

Draw words from a fixed distribution

Selected: $w_1$, $w_1$, $w_2$

Put a copy of $w_2$ back

Draw words from a fixed distribution

Selected: $w_1$, $w_1$, $w_2$

Put a copy of $w_2$ back
Draw words from a fixed distribution

Selected: $w_1, w_2, w_3, \ldots$

Is this a NB model, i.e. does it assume each individual word occurrence is independent?

Yes! Doesn't matter what words were drawn previously, still the same probability of getting any particular word

Does this model handle multiple word occurrences?
Draw words from a fixed distribution

Selected: $w_1, w_1, w_2, \ldots$