Why (when?) does learning work?

- What can we say about generalization performance for particular learning methods or problems?
- Guarantees for classification of specific examples not possible, so what is?
  - Relative guarantees (e.g., Adaboost)
  - Bounds on \( \text{expected} \) performance
Hypothesis Accuracy

- Assume examples generated according to probability distribution $D$.
- Error of hypothesis $h$ (our model):
  \[ \text{error}(h) \equiv \Pr_D(h(x) \neq f(x)) \]
- Hypothesis $h$ is approximately correct iff:
  \[ \text{error}(h) \leq \epsilon \]

PAC Learning

- Idea: As we see more examples, set of consistent hypotheses shrinks.
- Suppose $h_{bad}$ is not approximately correct.
  - $\Pr(h_{bad} \text{ consistent with random example}) \leq 1 - \epsilon$.
  - $\Pr(h_{bad} \text{ consistent with } m \text{ examples}) \leq (1 - \epsilon)^m$.
- Let be $H_{bad}$ all bad hypotheses.
  \[ \Pr(H_{bad} \text{ contains a consistent hypothesis}) \leq |H_{bad}|(1 - \epsilon)^m \leq |H|(1 - \epsilon)^m \]
- If this probability $< \delta$, the hypothesis is probably approximately correct (PAC).
Sample Complexity

Pr(H_{bad} contains a consistent hypothesis) \leq |H_{bad}|(1-\varepsilon)^m \leq |H|(1-\varepsilon)^m

- Can ensure PAC learning by observing enough examples:

  \[ m \geq \frac{1}{\varepsilon} \left( \ln \frac{1}{\delta} + \ln |H| \right) \]

  \[ \text{epsilon: error rate of hypothesis} \]

  \[ \text{delta: probability that a bad hypothesis is consistent with training examples} \]

- This sample complexity depends strongly on size of hypothesis space.

Mushroom Example

- How many hypotheses are consistent with the 12 examples we’ve seen?
- How many of those are exactly right?
- How many are approximately right?
  - Let’s say 75% correct
- If we had seen only 8 examples, how many consistent hypotheses?
- How many of those exactly right?
- How many are approximately right?
  - Again 75% correct...

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Size</th>
<th>Color</th>
<th>OnPizza</th>
<th>Edible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>Yellow</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>Yellow</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>NY</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>NY</td>
<td>No</td>
<td>YES</td>
</tr>
<tr>
<td>Spotted</td>
<td>Small</td>
<td>Yellow</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Spotted</td>
<td>Small</td>
<td>Yellow</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>Spotted</td>
<td>Small</td>
<td>NY</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Spotted</td>
<td>Small</td>
<td>NY</td>
<td>No</td>
<td>YES</td>
</tr>
<tr>
<td>No Spots</td>
<td>Large</td>
<td>Yellow</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Large</td>
<td>NY</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>Yellow</td>
<td>No</td>
<td>YES</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>Yellow</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>NY</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>NY</td>
<td>No</td>
<td>NO</td>
</tr>
</tbody>
</table>
Validation

- How do we numerically evaluate the performance of our learning algorithm?

Overfitting

- We don’t initially know which attributes are going to be relevant, and which irrelevant.
  - Chances are, some might be irrelevant.

- Overfitting arises when patterns are seen that turn out to be irrelevant.
  - Example in Mushroom data: whether the mushroom is picked on a Tuesday.

- Examples in everyday life?

- Solutions
  - Significance test: more complex examples must be “sufficiently” compelling.
  - Cross validation
Performance Measurement

How do we know that $h \approx f$?

- Use theorems of computational/statistical learning theory
- Try $h$ on a new test set of examples
  - (use same distribution over example space as training set)

Learning curve:
% correct on test set as a function of training set size

Performance Measurement

- What would constitute a good set of training examples?
  - Can we just use the first $n$ training examples from the table?
  - Stationarity is critical
- Can performance ever get worse with more examples?
Validation

- Data is precious
  - Can’t always obtain new test datasets
  - Want largest possible training and test datasets
  - How can we make maximum use of our datasets?

- Simple method: Collect N samples
  - Use M for training
  - Use (N-M) for testing.

Cross Validation

- Suppose we have N samples
  - Train on first N/2 samples
  - Test on last N/2 samples

- Can we also:
  - Train on last N/2 samples
  - Test on first N/2 samples
  - Average test error from both runs!
  - 2-fold cross validation
K-fold Cross-validation

- Divide dataset into K sets
- for n=1 to K
  - Train on all datasets except n
  - Test on dataset n
  - Compute average test error over all K runs

- If K=number of samples
  - Leave-one-out cross-validation