Overfitting, Cross-Validation

Recommended reading:

• Neural nets: Mitchell Chapter 4
• Decision trees: Mitchell Chapter 3

Machine Learning 10-701

Tom M. Mitchell
Carnegie Mellon University
Overview

• Followup on neural networks
  – Example: Face classification

• Cross validation
  – Training error
  – Test error
  – True error

• Decision trees
  – ID3, C4.5
  – Trees and rules
Multilayer Networks of Sigmoid Units
Learning Hidden Layer Representations

A target function:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000</td>
<td>10000000</td>
</tr>
<tr>
<td>01000000</td>
<td>01000000</td>
</tr>
<tr>
<td>00100000</td>
<td>00100000</td>
</tr>
<tr>
<td>00010000</td>
<td>00010000</td>
</tr>
<tr>
<td>00001000</td>
<td>00001000</td>
</tr>
<tr>
<td>00000100</td>
<td>00000100</td>
</tr>
<tr>
<td>00000010</td>
<td>00000010</td>
</tr>
<tr>
<td>00000001</td>
<td>00000001</td>
</tr>
</tbody>
</table>

Can this be learned??
Learning Hidden Layer Representations

A network:

Learned hidden layer representation:

<table>
<thead>
<tr>
<th>Input</th>
<th>Hidden Values</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000</td>
<td>.89 .04 .08</td>
<td>10000000</td>
</tr>
<tr>
<td>01000000</td>
<td>.01 .11 .88</td>
<td>01000000</td>
</tr>
<tr>
<td>00100000</td>
<td>.01 .97 .27</td>
<td>00100000</td>
</tr>
<tr>
<td>00010000</td>
<td>.99 .97 .71</td>
<td>00010000</td>
</tr>
<tr>
<td>00001000</td>
<td>.03 .05 .02</td>
<td>00001000</td>
</tr>
<tr>
<td>00000100</td>
<td>.22 .99 .99</td>
<td>00000100</td>
</tr>
<tr>
<td>00000010</td>
<td>.80 .01 .98</td>
<td>00000010</td>
</tr>
<tr>
<td>00000001</td>
<td>.60 .94 .01</td>
<td>00000001</td>
</tr>
</tbody>
</table>
Training

Sum of squared errors for each output unit
Training

Weights from inputs to one hidden unit

# of gradient descent steps →
Training

Hidden unit encoding for input 01000000

# of gradient descent steps →
Neural Nets for Face Recognition

90% accurate learning head pose, and recognizing 1-of-20 faces
Learned Hidden Unit Weights

left  str  rght  up

Learned Weights

30x32 inputs

Typical input images

http://www.cs.cmu.edu/~tom/faces.html
Cognitive Neuroscience Models Based on ANN’s

[McClelland & Rogers, Nature 2003]
Figure 4 | The process of differentiation of conceptual representations. The representations are those seen in the feedforward network model shown in Fig. 3. a | Acquired patterns of activation that represent the eight objects in the training set at three points in the learning process (epochs 250, 750 and 2,500). Early in learning, the patterns are undifferentiated; the first difference to appear is between plants and animals. Later, the patterns show clear differentiation at both the superordinate (plant–animal) and intermediate (bird–fish/tree–flower) levels. Finally, the individual concepts are differentiated, but the overall hierarchical organization of the similarity structure remains. b | A standard hierarchical clustering analysis program has been used to visualize the similarity structure in the
Overfitting in ANNs
How should we choose the number of weight updates?
Overfitting in ANNs
Overfitting in ANNs

How should we choose the number of weight updates?
How should we allocate N examples to training, validation sets?
How will curves change if we double training set size?
How will curves change if we double validation set size?
What is our best unbiased estimate of true network error?
Overfitting and Cross Validation

**Overfitting**: a learning algorithm overfits the training data if it outputs a hypothesis, $h \not\in H$, when there exists $h' \not\in H$ such that:

$$[err_{train}(h) < err_{train}(h')] \land [err_{true}(h') < err_{true}(h)]$$

where

$$err_{true}(h) = \sum_{x \in X} P(x) \delta((h(x) \neq f(x)))$$
Three types of error

True error:

$$\text{err}_{true}(h) = \sum_{x \in X} P(x) \delta((h(x) \neq f(x))$$

Train set error:

$$\text{err}_{train}(h) = \frac{1}{|S_{train}|} \sum_{x \in S_{train}} \delta((h(x) \neq f(x))$$

Test set error:

$$\text{err}_{test}(h) = \frac{1}{|S_{test}|} \sum_{x \in S_{test}} \delta((h(x) \neq f(x))$$
Bias in estimates

$err_{train}(h)$  \hspace{1cm} \text{Gives a biased (optimistically) estimate for} \hspace{1cm} err_{true}(h)$

$err_{test}(h)$ \hspace{1cm} \text{Gives an unbiased estimate for} \hspace{1cm} err_{true}(h)$
Leave one out cross validation

Method for estimating true error of h’

• $e = 0$

• For each training example $z$
  – Training on $\{\text{data} - z\}$
  – Test on single example $z$; if error, then $e \leftarrow e + 1$

Final error estimate (for training on sample of size $|\text{data}| - 1$) is: $e / |\text{data}|$
Leave one out cross validation

The leave-one-out error, $e / |\text{data}|$, gives an almost unbiased estimate for

$$E_{P(X)}[err_{true}(L(D_{m-1}))]$$

where $L$ denotes the learning algorithm, $L(D_{m-1})$ denotes the hypothesis output by learner $L$ given training set $D_{m-1}$, $D_{m-1}$ denotes a sample containing $m-1$ training examples drawn independently from $P(X)$, and $m$ is the number of examples available to the leave-one-out procedure. In other words, leave-one-out error estimates the expected error of the hypothesis learned by $L$, given $m-1$ training examples drawn at random from $P(X)$. 
Leave one out cross validation

In fact, the $e / |data|$ estimate of leave-one-out cross validation is a slightly pessimistic estimate of

$$E_P(X)[err_{true}(L(D_{m-1}))]$$

To see why, imagine learning the probability of heads with a coin with true probability 0.5. Given a sample $\{H, T, H, T\}$ it is easy to see that when we leave out the first example, $H$, the learner will estimate $\hat{P}(H) = 0.33$, which will then make it incorrectly predict tails for this held out example. Similarly, it will misclassify each of the four left out examples in turn.
How should we choose the number of weight updates?
How should we allocate N examples to training, validation sets?
How will curves change if we double training set size?
How will curves change if we double validation set size?
What is our best unbiased estimate of true network error?
What you should know:

• Neural networks
  – Hidden layer representations

• Cross validation
  – Training error, test error, true error
  – Cross validation as low-bias estimator