Outline for Today

- Return of the Midterm
- VC dimension of feed-forward networks
- Multiplicative weight updates and mistake bounds

VC dimension of Networks

Can we determine the VC dimension of neural networks?

Sometimes. Consider *directed, acyclic, layered* networks

- Directed: network units have input/output
- Acyclic: no loops
- Layered: can partition network units into layers such that outputs of layer k units go only to inputs of layer $k+1$

Can describe VC dimension of such networks in terms of

1. Graph structure of network
2. VC dimension of individual units

VC dimension of G-Composition

Suppose individual units can represent function class C

- e.g., perceptrons represent Lin. Threshold fns

  Definition: G-**composition of** C is the class of all functions that can be implemented by network G using individual units that take on functions from the class C.

  VC-dimension of compositions. Let G be a layered directed acyclic graph with n input nodes and $s \geq 2$ internal nodes, each of indegree r. Let C be a concept class over $\Re^r$ of VC dimension d, and let GC be the G-composition of C. Then $VC(GC) \leq 2ds \log(es)$

Example: networks of perceptrons with r inputs each

- $VC(C_{perceptron}) = r + 1$
- $VC(GC_{perceptron}) \leq 2(r + 1)s \log(es)$

Sample complexity of neural nets

number m of training examples sufficient to learn (with probability at least $(1-\delta)$) any target concept from $GC_{perceptrons}$ to within error ϵ:

$$
\begin{aligned}
m &\geq \frac{1}{\epsilon}(4\log(2/\delta) + 8VC(H)\log(13/\epsilon)) \\
&\geq \frac{1}{\epsilon}(4\log(2/\delta) + 16(r+1)s\log(es)\log(13/\epsilon))
\end{aligned}
$$

- depends on r, but not on number of inputs to G
- m examples sufficient to assure low future error for consistent hypothesis from $GC_{perceptrons}$
- but we don't have effective algorithms to learn nets of perceptrons
- however, $GC_{sigmoids} \subseteq GC_{perceptrons}$, so provides some insight into this

For more, see [Kearns and Vazirani, 1994]

Winnow and Perceptrons

- Both make a recommendation by some weighted vote among inputs
- Winnow uses muliplicative weight updates, Perceptron additive
- Winnow useful when many irrelevant features, mistakes expensive

Winnow and Version Spaces

Consider each feature a hypothesis (e.g., outputs a 1 or 0), and Winnow is taking a weighted vote

We discussed earlier that we could allow vote ammong $h \in VS$

- What's a mistake bound for voting using VS? (halving algorithm)
- VS algorithm zero's wt if h inconsistent, Winnow just reduces it
- Winnow more robust, but explicitly listing VS can get expensive