# CS 2416 - Machine Learning Theory

## Lecture #7: 27 October 2005

**Lecturer: Toniann Pitassi**

**Scribe Notes by: Toniann Pitassi**

## 1 VC dimension and Associated Theorems

Last week, VC dimension was introduced and four theorems pertaining to the VC dimensions of concept classes (hereafter denoted $VCD(\mathcal{C})$) were presented. (KV refers to the course text by Kearns and Vazirani) The $VCD(\mathcal{C})$ is the size of the largest set of points $S$ in the instance space that can be split by $\mathcal{C}$ in all $2^{|S|}$ ways. In other words, for any labelling of the points in $S$, there is a concept in $\mathcal{C}$ that is consistent with this labelling. Also, $\mathcal{C}[m]$ is the maximum number of ways to split $m$ points in the instance space using concepts of $\mathcal{C}$.

**Theorem 1** *(Sauer's Lemma; lemmata 3.1, 3.2 in KV)*
*for* $VCD(\mathcal{C}) = d$ *then* $\mathcal{C}[m] \leq \sum_{i=0}^{d} \binom{m}{i} = O(m^d)$

**Theorem 2** *(section 3.5.2 in KV)*
*for any class* $\mathcal{C}$*, distribution* $\mathcal{D}$*, if the number of samples* $m > \frac{2}{\varepsilon}\left[\log_2(\mathcal{C}[2m]) + \log_2\left(\frac{1}{\delta}\right)\right]$*, then with probability* $(1-\delta)$*, all bad hypothesis (error* $> \varepsilon$*) in* $C$ *are inconsistent with the data.*

**Theorem 3** *(theorem 3.3 in KV)*
*For* $VCD(\mathcal{C}) = d$*, same as Theorem 2 except with the bound* $m > \frac{8}{\varepsilon}\left[d \log\left(\frac{1}{\varepsilon}\right) + \log\left(\frac{1}{\delta}\right)\right]$

**Theorem 4** *(theorem 3.5 in KV is a tighter bound)*
*Let* $VCD(\mathcal{C}) = d$*. For any learning algorithm* $A$*, there exists a concept class* $\mathcal{C}$ *and distribution* $\mathcal{D}$ *such that if* $m \leq \frac{d}{2}$ *examples are selected from* $\mathcal{D}$*, then* $\Pr_{\mathcal{D}}(error(h) > \frac{1}{8}) \geq \frac{1}{8}$.

We already proved Theorems 1, 2 and 3. We will now sketch the proof of Theorem 4. A full proof can be found in the text. This theorem demonstrates why VC dimension is important, because it states that one requires roughly $VCD(\mathcal{C})$ examples to get a reasonable hypothesis most of the time.

Let $C$ be the concept class. Since the VC dimension is $d$, consider some set $X'$ of $d$ points that can be shattered. Let $C' \subset C$ be a set of $2^d$ concepts, one for each dichotomy of $X'$. Choose a random target $c'$ from $C'$. Define the distribution $D$ as follows. One point $x'$ in $X'$ has probability $1 - 4(\epsilon)$, and the other $d-1$ points in $X'$ have probability $4(\epsilon)/(d-1)$. (So one point is highly likely and the other ones are rare.) The intuition is as follows. In $m$ examples, we expect to see only half of the rare ones. And on the unseen examples, we err half of the time, since $c'$ is chosen randomly. A full proof can be found in the text.

## Application of $\epsilon$-Nets and VC dimension: The Set Cover Problem

The notion of VC dimension is important for other applications as well as learning theorem. We will give one prominent example–a polynomial-time approximation algorithm for the set cover problem. More precisely, we will present an approximation algorithm for set cover that finds a set cover of size roughly $d \log dc$ where $c$ is the size of the optimal set cover. The algorithm uses the concept of $\epsilon$-nets plus the idea behind the multiplicative weights update method.

Above we defined the concept of an $\epsilon$-net for a set system $S$ over a universe $U$. Intuitively, $Q$ is an $\epsilon$-net for $S$ if it hits all sufficiently large sets in $S$. Below is the formal definition.

**Definition** Let $S$ be a set of subsets over a universe $U$. An $\epsilon$-net for $S$ is a set $Q \subseteq U$ such that for all sets $s \in S$ such that $|s| \geq \epsilon |U|$, $s \cap Q \neq \emptyset$.

Our main theorem on VC dimension stated that if we choose a set $Q$ of samples randomly, and of sufficiently large size, then with probability at least $(1 - \delta)$, $Q$ will form an $\epsilon$-net.

**Theorem 5** *(Theorem 3 reformulated in terms of $\epsilon$-nets) Let $d$ be the VC-dimension of a set system $(S, U)$ and let $\delta > 0$. If we draw $O((d/\epsilon)\log(d/\epsilon) + (1/\epsilon)\log(1/\delta))$ points at random from $U$, the resulting set $Q$ is an $\epsilon$-net with probability at least $(1 - \delta)$.*

**Definition** The set cover problem is defined as follows. The input is a collection $C$ of subsets over a ground set of $X$ elements, $|X| = n$. The output is the smallest subcollection $C' \subset C$ whose union is $X$. (That is, the union of the sets in $C'$ covers all of $X$.)

Set cover is a classic NP-complete problem. The well-known greedy algorithm gives an $O(\log n)$ approximation for the problem, and furthermore, it is known that no better approximation exists in general unless $P = NP$. However we can still hope for better approximation algorithms if we restrict ourselves to "sufficiently simple" set systems. In particular, what happens when the VC dimension of the underlying set system $C$ is bounded by $d$? Even in such cases, computing the optimal set cover is still NP-complete, as vertex cover is a special case of set cover with VC dimension 2. However, when $d$ is constant, we can achieve a better polynomial-time approximation algorithm than $\log n$. Below we will show how to efficiently compute a set cover that is within a factor $d'$ of optimal, where $d'$ depends only on the VC dimension, $d$. (In fact, $d'$ will turn out to be exponential in $d$.) The algorithm is based on our basic theorem about $\epsilon$-nets combined with the weighted majority technique that we've seen earlier.

First, what do $\epsilon$ nets have to do with the set cover problem? Let us view the set cover input as a bipartite graph. There are $|C|$ many nodes on the left, $L$, with one node corresponding to each set in $C$. There are $|X|$ many nodes on the right, $R$, with one node corresponding to each $x \in X$. There is an edge between $c \in L$ and $x \in R$ if and only if $x \in c$. Now the set cover problem can be viewed in terms of its dual hitting set problem, where now the underlying universe, $C'$, is $L$, and the sets, $X'$, are $set(x)$, $x \in R$, where $set(x)$ is the set containing all $c \in L$ such that $x \in c$.

**Lemma 6** *If the set system $(X, C)$ has VC-dimension $d$, then the dual set system $(X', C')$ has VC-dimension at most $2^{d+1} - 1$.*

**Proof** Suppose for contradiction that a subset $X'$ of size $2^{d+1}$ is shattered by $C'$; we will then show that a subset of $X$ of size $d+1$ is shattered by $C$. It is not hard to verify that a set $A \subseteq X'$ is shattered by $C'$ if and only if for each subset $A' \subset A$, there is an element $x \in X$ such that $x \in S$ if $S \in A'$ and $x \notin S$ if $S \in A\ A'$. Let $A \subseteq X'$ be a shattered subset of size $k = 2^{d+1}$ and write $A = \{S_0, S_1, \ldots, S_{k-1}\}$ For $i = 1, 2, \ldots, d+1$, let $x_i \in X$ be an element that belongs to $S_j$ if and only if the $i$th significant bit of $j$ (written in binary) is 1. Then the set $B = \{x_1, \ldots, x_{d+1}\}$ is shattered by $C$: for $B' \subseteq B$ we have $B \cap S_j = B'$, where the $i$th bit of $j$ is 1 if and only if $x_i \in B'$.

Using the $\epsilon$ net theorem, it follows that with probability at least $(1-\delta)$, a random set $Q$ of sufficiently large size will be an $\epsilon$-net. This means that $Q$ is a hitting set for all "heavy" sets $set(x)$ (all sets $set(x)$ of size at least $\epsilon|L|$). It is left to show how to modify the algorithm soas to cover not just the "heavy" sets, but all sets.

To accomplish this, we first notice that the $\epsilon$ net theorem also applies to the weighted case.

**Definition** Suppose that $(X, C)$ is a set system and $w$ is a weight function that assigns a weight to each $x \in X$. Define $w(S) = \sum_{w \in S} w(x)$ for $S \subseteq X$. A set $N \subseteq X$ is a weighted $\epsilon$-net for $(X, C, w)$ if $N \cap S \neq \emptyset$ whenever $S \in C$ and $w(S) \geq \epsilon w(X)$.

It is implicit in our argument that our main theorem continues to hold in the case of weighted set systems. (That is, a set $Q$ chosen randomly according to the weight function will still be an $\epsilon$-net with high probability as long as the size of $Q$ is sufficiently large.)

We are now ready to present the algorithm for approximating set cover. The main idea is to learn a weight function that approximates the true weight function. Set $\epsilon = 1/opt$.

1. Initialize $w(x) = 1$ for all $x \in X$
2. Find a weighted $\epsilon$-net $Q$ for $(X, C, w)$
3. If $Q$ is a hitting set for $(X, C)$ then halt and return $Q$, and otherwise find a set $S \in C$ that is not hit by $Q$. Double the weights of all elements in $S$ and repeat.

**Theorem 7** *When $\epsilon = 1/2opt$, the algorithm BG terminates in $O(n)$ iterations where $n = |X|$.*

**Proof** As in previous arguments, we show two things. First, we prove that the total weight of the system, $w(X)$ does not grow too quickly. That is, after $k$ iterations, $w(X) \leq n(1+\epsilon)^k$. To see this, initially we have $w(X) = |X| = n$. At each iteration, the set $S$ selected satisfies $w(S) < \epsilon w(X)$; since doubling the weight of elements in $S$ adds a total of $w(S)$ new weight to the system, we find that $w(X)$ grows by at most a $(1+\epsilon)$ factor in each iteration. Thus after $k$ iterations, $w(X) \leq n(1+\epsilon)^k$. Secondly, we prove that the total weight concentrated on the optimal hitting set, $H^*$, grows pretty fast. That is after $k$ iterations, $w(H^*) \geq opt \cdot 2^{k/opt}$. To see this, initially we have $w(H^*) = |H^*| = opt$. Since $H^*$ is a hitting set, there is at least one element of $H^*$ in each set of $C$; thus no matter which set $S$ is chosen, some element of $H^*$ will have its weight doubled. By convexity of the function $2^x$, $w(H^*)$ will be minimized if these doublings are spread out over the elements of $H^*$ as evenly as possible. Thus after $k$ iterations, $w(H^*) \geq opt 2^{k/opt}$ as desired.

Setting $w(H^*) \leq w(X)$ and solving for $k$, it follows that $k \leq 4 \cdot opt \cdot \log(n/opt)$, and thus $k = O(n)$.

Note that the above theorem assumes that $\epsilon = 1/2opt$. In reality we don't know $opt$, but nonetheless, we can run the algorithm with successive guesses, $opt = 1, 2, 4, 8, \ldots$

# Notes

## The Multiplicative Weights Update Method

So far, we've seen this method applied to several different applications, including: the perceptron algorithm for learning linear threshold functions, the weighted majority algorithm for online learning, and now an approximation algorithm for set cover. In fact, we'll use the method once again next lecture to prove Freund and Schapire's famous boosting theorem for PAC learning. A very recent survey article by Arora, Kazan and Kale presents the multiplicative weights update method as a general algorithmic paradigm, and then gives many additional applications of the paradigm.

## The Complexity of Computing the VC Dimension

The computational complexity of determining the VC dimension is an interesting open problem. More precisely, the input to the problem is a set of $m$ vectors, each of length $n$, and a number $d$. The input should be accepted if and only if the VC dimension of the vectors is at least $d$. By Sauer's Lemma, the VC dimension is at most ?? and therefore we can assume that $d$ is at most $O(\log m)$. The brute-force algorithm is to try all $d$-sized subsets of $[n]$ to see if any of these sets are shattered. The runtime of the brute-force algorithm is therefore $n^{O(\log m)}$, which is quasipolynomial time. Thus it is highly unlikely that determining the VC dimension is NP-hard since if this were the case, then any problem in NP would be solvable in quasipolynomial time. On the other hand, there is no known polynomial-time algorithm, and there is some evidence that such an efficient algorithm is not possible.

## The VC Dimension of Neural Nets

We have seen how to calculate VC dimension for some simple function classes in previous lectures. Now we will examine this problem for some non-trivial classes like neural networks.

To estimate the VC dimension, we first abstract the structure of neural networks, then we figure out the function set the structure represents with different setting of all possible parameters. By estimating the maximal number of samples that the function set can shatter, we will upper bound the VC dimension.

We can abstract the structure of neural network as a layered directed acyclic graph. Figure 1 shows a layered directed acyclic graph where the number of input nodes, $n$, is 3, the number of internal nodes, $s$, is 3, and the indegree of each node, $r$ is at most 2.

Let $\mathcal{C}$ be a concept class over $\mathbb{R}^r$. For such a graph $G$, we now label each internal node with a concept $c \in \mathcal{C}$. Since the number of input nodes is $n$, $G$ gives us a concept over $\mathbb{R}^n$. It is clear that the computation can be performed in a bottom-up way from input nodes to root node just as the neural network does. Let's see how this works by an example.

**Example**

In figure 1, internal nodes $h_1, h_2, h_3$ represent binary operators AND, OR, OR respectively. This gives us a concept over $\{0,1\}^3$. For a set of inputs $S = \{000, 001, 010, 110\}$, output of $h_2$ is $\{0,0,1,1\}$ for each input respectively, output of $h_3$ is $\{0,1,1,1\}$, and $h_1$ takes the output of $h_2$ and $h_3$ which gives us $h_2 \wedge h_3 = \{0,0,1,1\}$. We can change $h_1, h_2, h_3$ to any other binary operators in the class, say OR, AND, AND respectively, then for the same inputs $S$, the output of $h_1$ will be $\{1,0,0,0\}$.

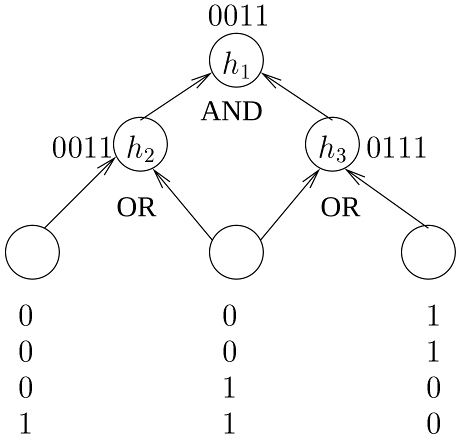


Figure 1: A layered directed acyclic graph with number of leaf nodes $n = 3$, number of internal nodes $s = 3$ and internal in-degree $r = 2$.

Formally, we have the following definition:

**Definition**

Let $\mathcal{C}$ be a concept class. The *G-decomposition* of $\mathcal{C}$, which we denote $\mathcal{C}_G$, is the set of all concepts obtained by labeling internal nodes of $G$ with concepts from $\mathcal{C}$.

**Theorem 8** *(theorem 3.6 in KV)*

*Let $G$ be a layered directed acyclic graph with $n$ inputs, $s \geq 2$ internal nodes, each of in-degree $r$. Let $\mathcal{C}$ be a concept class (over $R^r$ or $\{0,1\}^r$) of VC dimension $d$. Let $\mathcal{C}_G$ be the G-decomposition of $\mathcal{C}$. Then $VCdim(\mathcal{C}_G) \leq 2ds\log(es)$.*

**Proof**

Fix any set $S$ of $m$ input vectors, $\overline{x}_1, \ldots, \overline{x}_m$ to the graph $G$. For this fixed input set $S$, we want to compute an upper bound on the total number of dichotomies of $S$ that is realized by $\mathcal{C}_G$. Fix a particular assignment of concepts from $\mathcal{C}$ to each of the internal nodes of $G$. This induces a computation of $G$ on $S$. A computation can be thought of as a labelling of each internal node of $G$ by a vector in $\{0,1\}^m$, where bit $i$ in the vector $v_j$ corresponding to the node $j$ represents the value of the $j^{th}$ gate on the $i^{th}$ input, $s_i$. The set of all possible computations of $G$ on $S$ is obtained by ranging over all possible choices of labels from $\mathcal{C}$ for the nodes of $G$. Clearly the number of dichotomies on $S$, $|\Pi_{\mathcal{C}_G}(S)|$ is bounded by the total number of possible computations of $G$ on $S$, which we shall denote $T_{\mathcal{C}_G}(S)$.

To bound $T_{\mathcal{C}_G}(S)$, let $G'$ be the subgraph obtained by removing the output node from $G$. Let $T_{\mathcal{C}_{G'}(S)}$ denote the total number of computations of $G'$ on $S$. Each fixed computation of $G'$ can be extended to at most $\Pi_{\mathcal{C}m}$ computations of $G$. Thus we obtain

$$T_{\mathcal{C}_G}(S) \leq (\Pi_{\mathcal{C}}(m))^s \leq ((em)/d)^{ds}.$$

The last inequality follows from the lemma that we proved earlier, giving an upper bound on number of dichotomies of a set of any size $m$, as a function of the VC dimension. (This is Lemma 3.2 in the book).

**Corollary 9** *(theorem 3.7 in KV)*

*Let $\mathcal{C}_G$ be the class of neural networks on $G$ with in-degree $r$, $s$ internal nodes. Each node is a linear threshold function. The sample complexity of learning $\mathcal{C}_G$ is bounded by* $O(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{(rs+s)\log s}{\epsilon}\log\frac{1}{\epsilon})$.

**Proof**

Recall theorem 3 in the previous lecture states that the sample complexity of learning $\mathcal{C}_G$ is bounded by $O(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{1}{\epsilon}\text{VCdim}(\mathcal{C}_G)\log\frac{1}{\epsilon})$. We just showed that $\text{VCdim}(\mathcal{C}_G) \leq 2ds\log(es)$. If each node in $G$ is a linear threshold with in-degree $r$, then $d = r+1$, thus $\text{VCdim}(\mathcal{C}_G) \leq 2(r+1)s\log(es)$. Substitute into sample complexity bound, we complete the proof. $\square$