# MLT Lecture 3 — representation: box fit

Matus Telgarsky

## 1 Administrivia

- Comments from Homework 0 (abbreviated from lecture).
  1. SVD is very useful; here are some refs.
     - My favorite are notes from Hsu (2017). Only 11 pages! Note that in these notes he invokes other notes on eigendecomposition to give the existence of the SVD.
     - The SVD chapter in the upcoming data science book is good (Blum et al., 2017, Chapter 3).
     - Tons of information can be found in a numerical linear algebra book I like by Trefethen and Bau (1997).
  2. Chebyshev breaks down with the union bound; Hoeffding gives concentration; LIL additionally gives *anti-concentration*.
  3. Oscillation counting is our main tool in separating function classes, and indeed at the heart of the proof from Minsky-Papert that linear functions can be inadequate (in the end it was a univariate argument. . . ).
- Any questions?

## 2 Representation

**Recap.** We're in the representation part of the course: showing how well various classes of predictors approximate others.

- Last time we showed that linear predictors can badly fail at approximating quadratic predictors in $\mathbb{R}^2$ (via a classical argument; on the line, there are simpler arguments. . . ).
- Today we'll show that various standard function classes in machine learning approximate boxes, and this in turn suffices to approximate continuous functions.
- Next time we'll show that various classes approximate polynomials, and this also suffices.

## 3 Box approximations

Approach in this lecture (and part of the next lecture):

1. First we give a lemma that says: if we can approximate piecewise constant functions on the grid, we can approximate continuous functions.
2. Then we use this to prove decision trees, boosted decision trees, and 3-layer ReLU networks can approximate continuous functions.

Recall also how we measure distance:

$$\|f - g\|_1 = \int_{[0,1]^d} |f(x) - g(x)| \,\mathrm{d}x \qquad \text{and} \qquad \|f - g\|_{\mathrm{u}} = \sup_{[0,1]^d} |f(x) - g(x)|.$$

Our hiding of $[0, 1]^d$ within the notation is nonstandard but convenient.

**Remark 3.1** (Curse of dimension)**.** When we decomposed statistical problems into approximation/representation, optimization, and generalization, we mentinoed how we can trade off between these by changing the size / complexity of the class of predictors/functions. Here, we will be fitting continuous function, but we'll pay a hefty price: most representations will be of the size $\mathcal{O}(1/\epsilon)^d$. This is one version of the *curse of dimension*: the description size rapidly (exponentially) blows up with the dimension. There are other manifestations of the "curse of dimension" (e.g., points in high dimension are nearly equidistant). $\diamond$

**Lemma 3.2.** *Let any continuous $g : \mathbb{R}^d \to \mathbb{R}$ and any $\epsilon > 0$ be given. Then there exists a $\delta > 0$ so that for any partition $\mathcal{P}$ of $[0,1]^d$ into rectangles (products of intervals) $\mathcal{P} = (R_1, \ldots, R_N)$ with all side lengths not exceeding $\delta$, there exists scalars $(\alpha_1, \ldots, \alpha_N)$ so that*

$$\left\|h - g\right\| \leq \epsilon, \qquad \text{where} \qquad h = \sum_{i=1}^{N} \alpha_i \mathbb{1}_{R_i}.$$

**Remark 3.3.** Lemma 3.2 requires $N \geq 1/\delta^d$, matching the exponential blowup we just mentioned. $\diamond$

*Proof.* This proof will use the following standard fact (Rudin, 1976, Theorem 4.19). *[ pictures omitted. ]*

Since $g$ is continuous, it is *uniformly continuous* over $[0,1]^d$, meaning: for any $\epsilon > 0$, there exists $\delta > 0$ so that every $x, x' \in [0,1]^d$ with $\|x - x'\|_\infty$ satisfies $|g(x) - g(x')| \leq \epsilon$.

To prove the desired statement, let $\epsilon > 0$ be given, and choose $\delta > 0$ according to the preceding analysis fact. Let a partition $(R_1, \ldots, R_N)$ be given, and suppose all side lengths are no greater than $\delta$. For each $i \in [N]$, choose any $x_i \in R_i$ and set $\alpha_i := g(x_i)$, whereby any other $x \in R_i$ satisfies $\|x - x_i\|_\infty \leq \delta$ and thus $|g(x) - \alpha| \leq \epsilon$, and thus the function $h := \sum_i \alpha_i \mathbb{1}_{R_i}$ satisfies

$$\sup_{[0,1]^d} |h(x) - g(x)| = \sup_{i \in [N]} \sup_{x \in R_i} |h(x) - g(x)| dx = \sup_{i \in [N]} \sup_{x \in R_i} |\alpha_i - g(x)| dx \leq \epsilon.$$

$\square$

## 3.1 Decision trees

Let $\mathrm{DT}(\mathcal{H})$ denote **decision trees** with splitting functions $\mathcal{H}$. Given an $x \in \mathbb{R}^d$, a decision tree $f \in \mathrm{DT}(\mathcal{H})$ is a function represented by a binary tree which evaluates its input starting from the root node as follows.

- If the current node is an internal node, then it is associated with a function $h \in \mathcal{H}$ which outputs $h(x) \in \{\text{left}, \text{right}\}$; in particular, if $h(x) = \text{left}$, then processing recurses on the left subtree, otherwise it recurses with the right.
- If the current node is a leaf, then it is associated with some scalar $\alpha \in \mathbb{R}$, and this scalar is output as the function value (and the process terminates).

*[ Tree drawn in class. ]*

**Remark 3.4.** The leaves of the tree form a partition of the input space, and the path through the tree to the leaf determines what resides in each partition cell. In particular, each leaf corresponds to the logical and of predicates satisfied by the edges in the path leading to it. $\diamond$

When $\mathcal{H}$ is not specified, let DT denote **standard decision trees**, which have the choice

$$\mathcal{H} := \left\{x \mapsto \mathbb{1}[x_i \geq r] : i \in [d], r \in \mathbb{R}\right\}.$$

Often these are called "decision trees with axis-aligned splits".

**Theorem 3.5.** *Given any continuous $g : \mathbb{R}^d \to \mathbb{R}$ and any $\epsilon > 0$, there exists $f \in DT$ with $\|f - g\|_{\mathrm{u}} \leq \epsilon$.*

*Proof.* *[ In class we essentially gave a picture proof. ]*

First let $g$ and $\epsilon > 0$ be given, and take $\delta$ as provided by Lemma 3.2. This proof will construct a tree so that the leaves form a uniform grid partition of $[0,1]^d$ into $k^d$ pieces where $k := \lceil 1/\delta \rceil$. By assigning values to leaves according to $(\alpha_i)_{i=1}^N$ from Lemma 3.2, the proof is complete, so it suffices to show that internal nodes can be chosen to form this partition.

To this end, consider the first dimension. By forming a chain of the predicates $\mathbb{1}[x \geq 1/k]$, $\mathbb{1}[x \geq 2/k]$, and so on up through $\mathbb{1}[x \geq (k-1)/k]$, a subtree is formed whose leaves are a partition of points along dimension one. Recursing in this same way on dimension $j+1$ from dimension $j$, a new set of leaves is obtained which partitions the first $j+1$ dimensions of $[0,1]^d$. The process terminates once all dimensions have been partitioned. $\square$

**Remark 3.6.** There are other ways to do the construction; for instance, rather than forming each dimension partition as a chain, one could construct a balanced tree by first splitting at 1/2, then at 1/4 and 3/4, and so on. But generalization bounds penalizing the construction of trees often care about the number of nodes and the depth is less relevant, so this doesn't buy much. In either case, the tree has $\mathcal{O}(k^d)$ leaves. $\diamond$

## 3.2 Boosted decision trees

Given some functions $\mathcal{F}$, let $\text{span}(\mathcal{F})$ denote their linear span:

$$\text{span}(\mathcal{F}) := \left\{ x \mapsto \sum_{i=1}^{N} \alpha_i f_i(x) \ : \ N \in \mathbb{Z}_{\geq 0},\ f_i, \dots f_N \in \mathcal{F} \right\}.$$

Boosting algorithms output predictors from $\text{span}(\mathcal{F})$, where $\mathcal{F}$ are called "weak learners", "base predictors", or something like that.

**Theorem 3.7.** *Let any continuous $g : \mathbb{R}^d \to \mathbb{R}$ and any $\epsilon > 0$ be given. Let $\mathcal{B}$ denote standard decision trees with at most $2d$ nodes. Then there exists $f \in \text{span}(\mathcal{B})$ such that $\|f - g\|_{\text{u}} \leq \epsilon$.*

**Remark 3.8.**
- This is the class of boosted decision trees. Popular implementations like `xgboost` are still in use. Many old implementations used decision "stumps": depth 1 standard decision trees.

$\diamond$

*Proof.* First note that a decision tree of depth $2d$ can represent an indicator an arbitrary product of intervals, the tree having a chain of nodes for each constituent halfspace, anything failing any halfspace test resulting in a leaf with output 0, and the intersection of all halfspace having leaf value 1, with some care taken for open halfspaces. *[ Picture drawn in class. ]*

By the lemma from the previous lecture, there exists an integer $k$ so that if we divide $[0,1]^d$ into $N = k^d$ cubes $(R_i)_{i=1}^N$, each formed by $d$ intervals $\otimes_{j=1}^d I_j$, where $I_j = [i/k, (i+1)/k)$ for some $i < k-1$ and $I_j := [(k-1)/k, 1]$ otherwise. By the same lemma, there exist scalars $(\alpha_i)_{i=1}^N$ so that $h := \sum_i \alpha_i \mathbb{1}_{R_i}$ satisfies $\|h - g\|_{\text{u}} \leq \epsilon$. The proof is complete since $1_{R_i} \in \mathcal{B}$ by the preceding, thus $h \in \text{span}(\mathcal{B})$. $\square$

**Remark 3.9.**
1. The size of the tree is relevant in two ways. First, shallow trees take less time to construct. Second, they have better generalization properties (this will make more sense in the third part of the course).
2. It is a disaster to learn even these trees! In fact, note that these trees are a single branch, and thus called "decision lists", which are hard to learn agnosticially (Feldman et al., 2009).

   **Open problem:** find some way to circumvent the hardness barrier for instance via "improper learning".

$\diamond$

## 3.3 3 layer ReLU networks

*[ We stared but did not complete this topic. Next time... ]*

## References

Avrim Blum, John Hopcroft, and Ravindran Kannan. Foundations of data science, 2017. URL `https://www.cs.cornell.edu/jeh/book.pdf`.

Vitali Feldman, Venkatesan Guruswami, Prasad Raghavendra, and Yi Wu. Agnostic learning of monomials by halfspaces is hard. 2009.

Daniel Hsu. Topic 5: Principal component analysis, 2017. URL `http://www.cs.columbia.edu/~djhsu/AML/lectures/notes-pca.pdf`.

Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill Book Company, 3 edition, 1976.

Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.