Law and KNN

1 Setting

Let $X \subset R^n$ be a closed connected set of all possible cases, and $p(x) : X \to [0, 1]$ the fraction of people who believe the outcome of case x should be 1 (vs 0); we assume $p(\cdot)$ is Lipschitz.

Cases $x_t \sim uniform[X]$ arrive at each time t together with an opinion of a randomly sampled person $o_t \in \{0, 1\}$, and a decision d_t has to be made on a case at the time of arrival. The quality of history of decisions $h = \{x_t, d_t\}$ is evaluated by the number of people who disagree with them:

$$L(h) = \sum_{t=1}^{\infty} e^{-\delta t} |p(x_t) - d_t|,$$

where δ is the discount factor. We let $L^*(h)$ be the optimal value of $L(h)$ given sequence $\{x_t\}$, that is value of L when $d_t = [p(x_t)]$, where [] denotes rounding to closest integer.

(Note that this isn't the loss function we're measuring in simulations now)

2 Algorithms

2.1 Local precedents

Inputs: integer k, real D_{max}.

At every step the algorithm maintains the set of precedents S, elements of which are case-decision tuplets (x, d); S is initialized with empty set. For a case x_t, if there are at least k precedents within distance D of x_t, the decision d_t is made by the majority rule over k nearest precedents to x_t. If there are not k precedents within D, the decision is set to $d_t = o_t$ and a precedent (x_t, d_t) is added to S.

Theorem 2.1. *As $\delta \to 0,\ k \to \infty, D \to 0$, the loss converges to optimal $\frac{L(h)}{L^*(h)} \to 1$.*

Conjecture 2.2. *There exists an algorithm that uses finite memory while achieving the optimality condition above for $\delta \to 1$.*

2.2 Single juror

Set $d_t = o_t$.

Generalizes to multiple jurors when there are multiple samples o_t available for a single case.

2.3 Rulebook

Eg: As cases arrive, a decision tree is constructed and cases are settled according to that tree

3 Questions

1. When do local precedents out-compete single juror?
2. Is conjecture true?
3. What's a good algorithm when $p(x)$ changes over time?
4. (vague) how do costs of all algorithms compare as the dimension of X is increased?
5. Is there a tractable game-theoretic angle here?

4 Spencer

4.1 Introduction

4.2 Simulations

The idea here is to explore how precedents do in different settings.

4.3 Objectives

Low regret / rapid convergence to good decider.

4.4 Asymptotics

4.4.1 Precedents vs Decision Trees

Problem 1: Make a small data structure that efficiently implements an arbitrary map from the d-dimensional, resolution $k = 2^r$ lattice $[2^r]^d$ to $\{0, 1\}$.

Obvious optimal solution: lookup table.

However, lookup tables have no way to benefit from geometric information to compress their data. So:

Problem 2: Problem 1, but if the map is just an upsampled map on a resolution $2^r/2^u$ lattice, then we see gains in either lookup time or data structure size.

Is this what "locality-sensitive hashing" is all about? Locality-sensitive hashing: map similar points to the same hashtable bucket. Technically, if two points are within a distance R, they must be hashed to the same bucket with $> p_1$ probability, and if they are farther than cR, then they must be hashed to different buckets with $< p_2$ probability. Can be used for e.g., nearest neighbor search. But not obviously what we want here.

Obvious solution: smaller lookup table. But that's cheating! Not sure why yet.

Problem 3: Problem 2, but the map can have *any* kind of geometric structure (whatever that means).

More realistically, given a prior over cases, we want the map to have a low probability of error. So even if a certain region of case space is complicated and resolves fine details of different cases, if cases rarely come up in that region, our data structure can be low-resolution in that region. Real legal systems almost certainly have this property, that law/precedent does not have much to say about types of cases that almost never occur.

Problem 4: problem 3, but now include a prior over cases.

Idea: binary search trees. The separating hyperplanes (or whatever) can divide regions of about equal prior mass.

If you want to get to our k^d resolution with binary search trees, we need a tree with k^d leaves. This gives us just $2 * k^d$ nodes and lookup that is linear in d. Intuitively, this is already a lot more promising than the lookup table. With a uniform prior, the separating hyperplanes will just look like the $[k]^d$ lattice, and we can downsample easily by aggregating nodes whose children have (mostly) the same value. This is most effective if our b-tree structure matches the structure of the judge distribution, but still pretty good (I think) if some initial mistakes are made. If the prior is not uniform, we'll save resolution in regions of low prior probability.

This is the decision tree solution. We want to contrast this with the nearest neighbor solution, which is much simpler. Ignoring decisions about how to learn / maintain the data structure in an online fashion as we see new cases (we've been ignoring this so far), the nearest neighbor solution just looks at a fixed set of points we've already seen and judged (ignoring noise in the judges for now).

Note that all the solutions have exponential complexity (curse of dimensionality) if we really are mapping $[k]^d -> \{0, 1\}$. It's all about the structure and how different solutions exploit it.

Could this be related to gradient boosting? At each step, we learn a correction to the legal system of the previous step.

The nearest neighbor strategy automatically gives higher resolution in regions of high prior probability, without having to know the prior. Downsampling can be achieved via condensed nearest neighbor: find "prototype" points and toss the rest.

In its original form, nearest neighbor (C-NN) is actually quite simple. The algorithm just maintains a subset of training points called the prototype points, and at each step picks a training point misclassified by the prototypes and adds it to the prototypes.

This sounds a lot like what precedents do. If it gets a point that cannot be classified by the prototypes, it adds it to the prototypes. What if we did a quasi-precedents thing, where we judge all cases, and only if a case disagrees with its precedents, we add it to the set of precedents?

There's not too much theory about C-NN. This appears to be the state of the art: `https://icml.cc/Conferences/2005/proceedings/papers/004_Fast_Angiulli.pdf` It's sub-quadratic running time and does not depend on some arbitrary parameter like the order in which points are considered. The secret sauce appears to be centroids and Voronoi cells.