Lesson 4: Instance Based Learning

Instance based learning

Instance based learning distinguishes machine learning models which perform **"lazy learning" at scoring-time directly from the training data**. This is different from **other** machine learning models which lossily compress the training data into a **simpler hypothesis function**.

Some benefits

- Remembers all training data points.
- Fast for known points (lookup – depending on properties of the data structure used to store).
- "Simple"

Possible problems

- Overfits and generalizes badly
- Must handle conflicting training data

Since instance based learning often requires **traversing the training data** itself, it may be slow to score unknown inputs, but is easily adaptable to new training data points.

k-Nearest Neighbors

The number of neighbors k is a free parameter which must be learned.

Given: * training data $D = \{x_i, y_i\}$, * distance metric $d(q, x)$ * the required number of neighbors k * the query point q:

$$NN = \{i : \ d(q, x_i) \ in \ k \ smallest\}$$

For **classification**: vote of y_i or weighted vote or some other strategy.

For **regression**: take mean or weighted mean (weights may be 1/distance)

If there are distance conflicts (exactly same distances), **"take all of them"** the way college rankings do (this fudges the k).

Comparing learning and query times

Given a list of n data points (x_i, y_i) sorted by x_i what is the running time and space consumption of "learning" (assume no effort for ETL – which make no sense) and "query" (do not consider the learned data structure's space consumption) of 1-NN, k-NN, and linear regression:

model	Running Time	Space
1-NN Learning	1	n
1-NN Querying	$log\ n$ (binary search)	1
k-NN Learning	1	n
k-NN Querying	$log\ n + k$	1
Linear Regression Learning	n	1
Linear Regression Querying	1	1

- Simple linear regression learning runs in O(n) (due to bound on the dimension of x_i, even normal equations solution is "linear" – though this is a bit misleading) and takes O(1) space (For parameter).

Querying in instance based algorithms may be slower (depending on the data structure), though assuming no ETL costs, learning in instance based algorithms is faster. Moreover, **incremental learning** of new data points are possible and probably very fast.

IBL is more lazy about learning (pushes it till querying). So k-NN is a **lazy learner** (or "just in time" learner) whereas linear regression is an **eager learner**.

K-NN Bias

Preference bias: when searching the hypothesis space, the search may "prefer" a particular subset of the hypothesis space over another. Perhaps the hypothesis space is not completely searched and rather only a subset is searched. E.g. simpler trees, simpler functions, Occam's razor. Compare to restriction bias which is the total representational power of the hypothesis space itself.

What about k-NN?

- **Locality** – near points are similar and how is nearness defined. Moreover, define distance on input not label/output. This comes from domain knowledge.
- **Smoothness** – averaging produces smoothness as opposed to discontinuities.
- All **features seem to matter "equally"** – this comes from the distance function too.

Curse of dimensionality

As the number of features or dimensions grows, the **amount of data we need** to generalize accurately **grows exponentially**. Comes from Richard Bellman, the dynamic programming guy.

Covering the "same space" in higher dimensions requires exponentially more points. ("Same space" is not well-defined.)

Applies in all of ML and not just k-NN.

Distance function weighted on the dimensionality may be a solution.

Some other points

Distance metrics $d(x, q)$ – e.g. Euclіean, Manhattan, weighted-components, mismatches (for classification), etc. But the choise really matters.

Implications of k.

- When $n = k$, we get a constant function (assuming vote or regular average).
- Let $k = n$ with weighted average, the computation may not be so easy, but “smooth”-ish hypothesis. Points closer have greater say.
- Perhaps instead of weighted average, do “local” regression on the k-closest-points. Known as “**locally weighted regression**”. We can do decision trees, neural nets, etc. there. We can achieve more sophisticated curves and more complex hypotheses (removes some restriction bias, but may overfit).

What have we learned?

- Instance based learning
- Eager and lazy learning – lazy puts off work until needed, eager generally learns from training data ahead of time.
- k-NN
- nearest-neighbor; similarity (distance)
- Domain knowledge matters (distance)
- Classification versus regression
- “Averaging”/combining results.
- Composing learning algorithms together (locally weighted regression $x regression).
- Curse of dimensionality. Required data is $O(2^d)$.
- No free lunch theorem(?) – averaging over all possible data makes any learning algorithm no better than random. Domain knowledge helps.