Hien Duong

CPTS 315

Homework 4

**1.**

Using Euclidean distance and the data structure KD-trees we can efficiently run nearest neighbor on the point to determine its nearest neighbors. With the KD-tree we can eliminate a large portion of the space and will cut down on the computational expense needed for high dimension spaces.

**2.**

As long as the rules are consistent than it is possible to convert the rule set R into an equivalent decision tree. By consistent meaning that the rules don’t contradict each other, for example if I have a rule that says if a = yes then true and I have another rule saying if a = yes then false. Those two rules contradict each other so tree is not possible. Using the previous example if we have these rules if a = yes then true and if a = no then false. We can build a tree from this set where the root node is a, and the two branches are yes and no with the target label as true and false.

**3.**

A = (1, 1); B = (1.5, 2.0); C = (3.0, 4.0); D = (5.0, 7.0); E = (3.5, 5.0); F = (4.5, 5.0); and G = (3.5, 4.5)

**Iteration 1:**

C1 = (1.5, 2.0) B

C2 = (3.0, 4.0) C

Distances from points to C1

A to C1 =

B to C1 =

C to C1 =

D to C1 =

E to C1 =

F to C1 =

G to C1 =

Distances from points to C2

A to C2 =

B to C2 =

C to C2 =

D to C2 =

E to C2 =

F to C2 =

G to C2 =

Cluster Assignment

C1 : A, B

C2 : C, D, E, F, G

New Cluster Centers

C1 : <(1+1.5)/2,(1+2)/2> = (1.25, 1.5)

C2 : <(3+5+3.5+4.5+3.5)/5, (4+7+5+5+4.5)/5> = (3.9, 5.1)

**Iteration 2:**

Distance from points to C1

A to C1 =

B to C1 =

C to C1 =

D to C1 =

E to C1 =

F to C1 =

G to C1 =

Distances from points to C2

A to C2 =

B to C2 =

C to C2 =

D to C2 =

E to C2 =

F to C2 =

G to C2 =

Cluster Assignment

C1 : A, B

C2 : C, D, E, F, G

Same clusters so convergence, C1 and C2 will be the same

**4.**

Big data research has grown tremendously over the past five years in both academia and industry. As the size and complexity of the available dataset is growing, ethical questions are also being raised by big data research. These questions are getting more urgent as data and research is going beyond simple computational and natural sciences and addressing more sensitive aspects of human behavior, interaction, and health. Data such as digital medical records, economic insights, relationship mappings, an individual’s speech and actions, movement across space, and much more are used for big data research. There are ten simples rules for addressing the ethical issue that will rise as big data research gets even more bigger. The first rule is to acknowledge that data are people and can do harm. This means that as a big data researcher you must acknowledge the fact that the data that you work with represent and can impact people. The data can contain private information on an individual and even data that seemingly have nothing to do with people might impact individual’s lives in unexpected ways. The second rule is to recognize that privacy is more than a binary value. Big data researchers must realize that breaches of privacy are main reason how the data can be used to do harm. Researchers must recognize that privacy is contextual and situational, not reducible to simple public/private binary. The third rule is guard against the reidentification of your data. It is problematic to assume that data cannot be reidentified. Researchers should always make sure to anonymize data sufficiently to prevent the later identification of a specific individual. The fourth rule is to practice ethical data sharing. Sharing data is an expectation in projects and a key part in ethical research. However, researchers should not freely give away data. The fifth rule is to consider the strengths and limitations of your data; big does not automatically mean better. In order to do both accurate and responsible big data resource, it is important to ground datasets in their proper context including conflicts of interests. Context can affect all the stages in research, from data acquisition, to cleaning to interpretation of findings and dissemination of the results. The sixth rule is debate the rough, ethical choices. Sometimes researchers can encounter situations that are foreign or outside of the mandate of IRBs, in these situations it is important to debate the issue within a group of peers. The seventh rule is to develop a code of conduct for you organization, research community, or industry. The process of debating tough choices inserts ethics directly into the workflow of research making “fake ethics” as unacceptable as faking data or results. The key to a successful research is to not treat issues as an afterthought or a problem to outsource but rather these debates should be internalized. Especially when using trace data produced by people. This rule is relevant for all forms of research including those within industry who have access to data streams of the information of individuals. The eighth rule is to design your data and systems for auditability. This is important because a well-designed system for auditability can keep track of factors that might contribute to problematic outcomes. This can help strengthen research. The ninth rule is engage with the broader consequences of data and analysis practices. This means that big data researchers should think beyond traditional metrics of success in business and academia. The last rule is to know when to break these rules. This rule might be counterintuitive to the previous 9 rules, but it is important to know when to break away from the rules if it is going to benefit everyone in the end.

**5.**

As machine learning is growing larger and the lives of millions are being impacted by machine learning everyday we need to step back and look at the trouble with fairness and bias involved with machine learning. Bias in systems is commonly caused by bias in training data and we can only gather data about the world that we are currently living in. Which has a history of discrimination. This in turn will apply to our systems and showcase our darkest biases. In order to solve this problem with bias, we are going to have to consider both the social and the technical halves of this problems and see it as a sociotechnical problem. Allocative harm is when a system allocates or withholds certain groups an opportunity or resource. Representational harms occur when systems reinforce the subordination of some groups along the lines of identity like race, class, gender, etc. This harm can take place regardless of whether the resources are being withheld to members of a protected class. Such as representing someone as something they are not like representing a person as a gorilla. When clearly they are not. This harm can cause representation of black criminality which can lead to racial stereotypes which can lead to prospects in the labor market. So, know we have to think of how machine learning that mis represent others. Allocative harms are more immediate while representational harm is more long term. Allocative harm is also more easily quantifiable and discrete while representational harm is more difficult to formalize and diffuse. One is transactional and the other is cultural. There are 5 different types of representational harm. The first is stereotyping such as assuming that a doctor is always male, and a nurse is always female like with google translate translating gender-neutral languages. The second is recognition, being able to recognize someone as an actual human. Such as being able to recognize someone with a darker skin tone. The third is denigration, like how Google’s auto complete returns a shocking suggestion criticizing someone or something. Classification is also a social issue not just a technical issue. We need to start working on fairness forensics. There are a lot of things we can do to test our systems, from building pre-release trials where we can see how the system is working across different populations. We need to start taking inter-disciplinary seriously, working with people not in our fields but with deep expertise in other areas. When building a machine learning system, we have to ask ourselves who is going to benefit from our system and who is going to be harmed.

**6.**

As machine learning continues to grow a wide-spread and uncomfortable trend has started emerging. Developing and deploying machine learning systems is fast and cheap but maintaining them over time is difficult and expensive. This issue can is known as technical debt, this term is coined by Ward Cunningham to reason about the long-term costs incurred by moving quickly in software engineering. Machine learning systems have a special capacity for incurring technical debt, because they have all the maintenance problems of traditional code plus an additional set of ML-specific issues. This debt may be difficult to detect because it exists at a system level rather than the code level. Complex machine learning models can erode boundaries. There are a few ways that erosion of boundaries can significantly increase technical debt in machine learning systems. The first way is with entanglement, this is when machine learning systems mix signals together, entangling them and making isolation of improvements impossible. A mitigation to this problem is to isolate models and serve ensembles. Another possible mitigation is to focus on detecting changes in prediction behavior as they occur. The second way is with correction cascades. When we try to correct a change the correction model has created a new system dependency on the original mode. Making it more expensive to analyze improvements to that model in the future. The third way is undeclared consumers. This is when a prediction from a machine learning model is made widely accessible and consumers maybe be silently using the output of a model ad an input to another system is this issue is known as visibility debt. This can create a hidden tight coupling of the original model. Dependency debt is a key contributor to code complexity and technical debit. Data dependencies in ML systems carry a similar capacity for building debt, more are more difficult to detect. Code dependencies can be identified with static analysis by compilers and linkers. It is convenient to consume signals as input features that are produced by other systems. However, these signals are unstable, this means that they can qualitatively or quantitatively change behavior over time. A mitigation to this is to create a versioned copy of a given signal. Underutilized dependencies are packages that are not needed, these are input signals that provide little incremental modeling benefit. These dependencies can make a machine learning system unnecessarily vulnerable to change. They can be detected with an exhaustive leave-one-feature-out evaluation. Machine learning systems can end up influencing their own behavior if they update overtime, this can lead to a form of analysis debt. This means that it will be difficult to predict the behavior of a model before it is released. The direct feedback loop can directly influence the selection of its own future training data. The hidden feedback loop is when two systems influence each other indirectly through the world. Machine leaning methods can end up with high-debt design patterns/ There several system design anti-patterns. The first is Glue Code, developers tend to develop general purpose solutions as self-contained packages. Using generic packages can result in a glue code design pattern, meaning there is a massive amount of support code to get data into and out of general-purpose packages. This is costly long term and can result in the seizure of a system to peculiarities of a specific package. The second is pipeline jungles, these appear in data preparation and can evolve organically, as new signals are identified, and new information sources are added incrementally. The prepared date can end up being a jungle of scrapes, joins,. And sampling steps. All of this can lead to technical system debt. Another is the dead experimental code paths; this is when implementing experimental code paths such as conditional branches within the main production code can lead to a growing debt overtime due to increasing difficulties of minding backward compatibility and increase in cyclomatic complexity. Another is common smells; a design smell can indicate that there is an underlying problem to the system or component. There are three smells, plain-old-data type smell, multiple-language smell, and prototype smell. Another area where debt can accumulate is in the configuration of machine learning systems. A large system can contain a wide range of configurable options this can lead to debt. Since machine learning systems interact directly with the external world, and since the external world is rarely stable this rate of change can create ongoing maintenance costs. Such as fixed thresholds in dynamic systems, and monitoring and testing. There are also other areas of machine learning relate debt such as data testing debt, reproducibility debt, process management debt, and cultural debt. As machine learning systems are playing a more central role in real-world production settings, the issue of ML reliability has become increasingly critical. Machine learning systems are different from traditional software-based systems in that the behavior of ML systems is not specified directly into the code but is learned from the data. So, in order to test our ml systems, we want to have a sufficient set of tests of data. Data 1 tests feature expectations captured in a schema. Data 2 all features a beneficial. Data 3 no feature’s cost is too much. Data 4 Features adhere to meta-level requirements. Data 5 the data pipeline has appropriate privacy. Data 6 new features can be added quickly. Data 7 all input feature code is tested. The best practices for ML model development are still emerging. Model 1 every model specification undergoes a code review and is checked in to a repository, model 2 offline proxy metrics correlate with actual online impact metrics, model 3 all hyperparameters have been tuned, model 4 the impact of model staleness is known, model 5 a simpler model is not better, model 6 a model quality is sufficient on all important data slices, model 7 the model has been tested for consideration of inclusion. ML systems often rely on complex pipeline rather than a single running binary. This is how you would test infrastructure. Infra 1 training is reproducible, infra 2 model specification code is unit tested, infra 3 the full ML pipeline is integration tested, infra 4 model quality is validated before attempting to serve it, infra 5 The model allows debugging by observing the step-by-step computation of training or inference on a single example, infra 6 models are tested via a canary process before they enter production serving environments, infra 7 models can be quickly and safely rolled back to a previous serving version. It is important to know that your ML system is working correctly at launch this is the bests ways to monitor tests for ML. Monitor 1 dependency changes result in notification, monitor 2 data invariants hold in training and serving inputs, monitor 3 training and serving features compute the same values, monitor 4 models are not too stale, monitor 5 the model is numerically stable, monitor 6 the model has not experienced a dramatic or slow-leak regressions in training speed, serving latency, throughput, or RAM usage, monitor 7 the model has not experienced a regression in prediction quality on served data. Technical debt is hard to quantify so we can use ML Test score to help identify our test quality. The final score is computed in the following way, for each test half a point is awarded for manual execution with results documented and distributed. A full point is awarded if the system is in place to run the tests automatically on a repeated bias. A rubric is used identify qualities of the real systems.