1 Introduction

Today we have a different format, and we are going to do the chosen topic, which is computational aspects of online learning, and how to relate it to online statistical learning. The project is due the 17th of May. Tomorrow in the morning 10 – 12 am, any questions, I’ll be around in my office. It’s better to ask me before the end of the week (Friday), because then I will be traveling and I will be less accessible. Of course I am still accessible by email.

2 Computational Relationship Between Optimization and Learning

When can we learn efficiently versus when can I optimize efficiently? The question is, what is the relationship between the ability to solve an offline problem, and the ability to learn the equivalent of the offline problem?

2.1 Efficient Statistical Learning

Example 2.1. Learning a route in a network.

The offline version of this problem would be given a weighted graph $G = (V, E, W : E \to \mathbb{R})$, find the shortest path. All of us know that this can be solved very efficiently, for instance Dijkstra’s algorithm if the weights are non-negative. Bellman-Ford if not. Then there is a stochastic version, which corresponds to statistical learning. We have a distribution $\mathcal{D}$ over the weights $W : E \to \mathbb{R}$. Our hypothesis class $\mathcal{H}$ is the set of all paths in the graph, and we are looking for the path whose expected weight is the minimum.

What is the fundamental theorem of statistical learning? We are looking to find a path whose expected length is shortest. If we sample enough weights, and compute the best path for these weights, we will find this best path - a hypothesis is generalizable with small error. How many examples do we need? $O(\frac{\text{VC-dim}(\mathcal{H})}{\epsilon^2} \log(\frac{1}{\delta}))$. We compute $h^* = \arg\min \{\text{err}_S(h)\}$ as the best hypothesis, implies err$(h^*) \leq \epsilon$ with probability $1 - \delta$. What is the VC-dimension? This is a discrete set, so it is just the size of the number of paths, which is approximately $2^{|V|}$. Since the VC-dim is the log of the size, the number of samples is
~ |V|. Then, we can just use Dijkstra’s algorithm to find \( h^* \) efficiently (we are just computing the minimum path size), and voila, the offline case solves the online case.

Suppose I have an algorithm OPT that solves the problem \((\mathcal{H}, \mathcal{X}, \mathcal{D})\). Then given examples \( x_1, \ldots, x_T \), I define \( \text{OPT}(x_1, \ldots, x_T) = \arg\min_{h \in \mathcal{H}} \{\text{err}_{x_1, \ldots, x_T}(h)\} \).

So this leads to question 1: Given efficient OPT for problem \((\mathcal{H}, \mathcal{X}, \mathcal{D})\), can we statistically learn the problem efficiently?

Suppose you can solve the offline version efficiently, can we learn the statistical setting? Yes. It follows from the fundamental theorem. Why? The fundamental theorem of statistical learning by Vapnik says that you can take \( \mathcal{O}\left(\frac{\text{VC-dim}(\mathcal{H})}{\epsilon^2} \log\left(\frac{1}{\delta}\right)\right) \) examples and return \( h_{\text{ERM}} \) which is essentially OPT over these examples, and with probability \( 1 - \delta \), the error of \( h_{\text{ERM}} < \epsilon \). If \( \mathcal{H} \) is finite, then \( \text{VC-dim}(\mathcal{H}) = \log(|\mathcal{H}|) \). And now we are efficient.

Therefore, what we can show is that an efficient OPT implies efficient statistical learning.

The next question is: is this true for efficient online learning? Does an efficient OPT imply efficient online learning?

### 2.2 Efficient Online Learning

Let us take a hypothesis class \( \mathcal{H} = \{h_1, \ldots, h_n\} \). We iterate \( t = 1, \ldots \), where the player chooses some \( h_t \in \mathcal{H} \), and experiences a loss \( f_t(h_t) \). We say that online learning is possible if the regret \( t \left( \sum_{t=1}^{T} f_t(h_t) - \min_{h} \sum_{t=1}^{T} f_t(h) \right) \) 0 as \( T \to \infty \). Let us assume that \( \forall h, f_t(h) \in [0, 1] \). Now is online learning even possible in this setting? Online learning is possible using gradient descent and so on over a convex combination of the hypotheses.

We know, for instance, that the regret of randomized weighted majority goes to \( 0 \) with running time \( Tn \). This is possible, but not efficient. Here I am running in time according to the size of the hypothesis class, not in terms of its representation.

**Example 2.2.** Shortest path in graph.

We have \( \mathcal{H} = s-t \) paths in \( G \). Let us say \( f_t(h) = \text{length of } h \text{ in } G \) according to some weights. Can we minimize regret in this setting efficiently? Yes, because we take over the flow polytope, and so on. The reason is we can solve shortest path in graphs, optimize over the online version of the problem.

We saw in class that if you can minimize regret, you can solve the offline version. However, there are some problems which are NP-hard and so on - we don’t expect to be able to solve these efficiently in the online setting (since the offline version cannot be solved!) So now the correct question is: suppose you can optimize efficiently in the offline setting, can we do the online learning version efficiently. This is the main question we are asking.

So to restate: Suppose we have OPT with \( h_{\text{ERM}}(x_1, \ldots, x_T) = \arg\min_{h \in \mathcal{H}} \{\sum_{t=1}^{T} h(x_t)\} \). Is there an efficient algorithm that gives average regret \( < \frac{1}{10} \) in time \( \mathcal{O}\left(\text{poly}(\log(n))\right) \) - this is not number of iterations, actual time. Why did I ask average regret \( < \frac{1}{10} \) - any regret going to 0 will be going to 0, so this will be fine. If I run multiplicative updates, I will get this, but every iteration I have to spend a lot of time. Does optimization oracle make this efficient? Yes, if \( \mathcal{X} \) is sampled from a distribution \( \mathcal{D} \) - this is the statistical setting we saw before. You can call the OPT whenever you want.

So what do you think for the online case?

The theorem is as follows:

**Theorem 2.3.** (Hazan, Koren)

Any algorithm in OPT-model (meaning we get an offline learning oracle) that guarantees an average regret \( \leq \frac{1}{10} \) with probability \( \geq \frac{2}{3} \) runs in time \( \Omega(\sqrt{N}) \). Thus the online setting is much harder than statistical learning.
We were able to do this for the flow polytope problem - the flow polytope is one specific case. When we prove a lower bound, we have to consider a general algorithm that applies to any hypothesis class in any sample set. There is no general algorithm. For special cases, we can see frameworks for which you can do it well! I.e., convexity, strong-convexity, and so on. But these restrictions are necessary to make any claims.

How do you go about proving such a theorem - you construct a problem for which you cannot run an efficient algorithm. How do you prove lower bounds for running times? In general no one knows how to do it. In this setting we have information theory.

**Proof.** The way we are going to prove this is we are going to construct a setting where \( \text{o}(\sqrt{N}) \) algorithm for regret \( \leq \frac{1}{10} \) with probability \( \geq \frac{2}{3} \) contradicts a known lower bound. Consider the array problem: There is one 1, and the rest are \( n - 1 \) 0s. Any algorithm finding 1 with probability \( \geq \frac{2}{3} \) runs in time \( \Omega(\sqrt{n}) \) - we won’t prove this, but it is true. We will use it to show our lower bound. Basically it means that finding a randomly selected coordinate is hard.

We define a matrix with the hypotheses \( h_1, \ldots, h_N \) for the rows and \( x_1, \ldots, x_m \) for the columns. Let \( N = n^2 \). We will first divide the matrix into blocks of size \( n \). We define \( \mathcal{H}^* \subseteq \mathcal{H} = \{ h_1^*, \ldots, h_N^* \} \). Then \( l(h_i, x_j) = 1 \) if \( h \in \mathcal{H}^* \), and 0 if \( h \notin \mathcal{H}^* \) and \( i < j \), and 1 otherwise. We have \( |\mathcal{H}^*| = n = \sqrt{N} \). Each \( h_i^* \in [i * n, (i + 1) * n] \).

Now we need to show the following things. How can an algorithm that minimizes regret access this game. Player cannot go over an entire column: can query points - what is the value for \( \geq T \)?

Formally, you can define the optimization oracle as \( \text{argmax}_S (\text{supp}(S)) \). So I’m just giving you the largest guy in the support of \( S \). And this will be useless information, since from now on, I am going to have a loss of 1.

\[ \text{OPT}(S) = \mathcal{H} \text{ if } S \cap \mathcal{H}^* \neq \emptyset, \text{ return } h_i^* \text{. Otherwise, } i_t = \max_i i \in \text{supp}(S), \text{ return } h_t \text{. So take the index of the largest example we have seen so far. Once you get to } h_i, x_i, \text{ you start having a loss of 1 from then on, so it is useless information.} \]

Now suppose regret \( \leq \frac{1}{10} \) with probability \( \geq 110 \). At some iteration \( t \), the loss is 0. That means we found a member of \( \mathcal{H}^* \) which has not been optimal thus far. This effectively means that we found \( h^* \) ahead of time, since we got a loss of 0 - this is in contradiction to the array theorem. If regret is \( \leq \frac{1}{10} \), we solved the array problem. In our matrix, we had \( \sqrt{N} \) problems of size \( \sqrt{N} \). So to solve one of them, you need at least \( \sqrt{N} \) time. Therefore, the time spent is at least \( \Omega(\sqrt{N}) \).

The conclusion for this is that online learning is computationally hard even if you can solve the offline problem efficiently, in contrast to the statistical setting. Even if you have an oracle that solves the offline problem, you cannot necessarily solve the online problem efficiently.

Now, let us consider how much time does it take for randomized weighted majority to get regret \( \leq \frac{1}{4} \). Well, we have \( \frac{1}{4} \text{ Regret(RMW)} \leq \sqrt{T \log(N)} \leq \frac{1}{10} \), which implies \( T \sim 100 \log(N) \). Here we need \( \Omega(N) \) time per iteration, so overall we get \( N \log(N) \).

Now let us consider the EXP3 algorithm. We have \( \text{Regret(EXP3)} \leq \sqrt{TN \log(N)} \leq \frac{1}{10} \), and we get \( T \geq 100N \log(N) \), with \( O(1) \) time per iteration, so we are still \( N \log(N) \).

Our third question is then as follows: In OPT-oracle model, what is the best running time to achieve \( \frac{1}{T} \text{Regret} \leq \frac{1}{10} \).
Theorem 2.5. (Hazan, Koren)
There exists an algorithm with regret $\sqrt{\sqrt{NT}}$ with $O(1)$ time per iteration. This has regret better than EXP3. Then $\sqrt{\sqrt{NT}} \leq \frac{1}{10}$, leads to $T \geq 100\sqrt{N}$. If you know how to solve the optimization problem offline, then you get something better in terms of running time. It doesn’t help you with regret, but you can go faster.

Proving this will take too long, but I will define the algorithm that works.

Algorithm 1 Sliding-Window EXP3

1: Let $h^*_t = \arg\min_{h \in \mathcal{H}} \{h(x_1), \cdots, h(x_{t-1})\}$.
2: Let $Q \subseteq [N]$ chosen uniformly at random, $|Q| = \sqrt{N}$.
3: Let $A_1, A_2, A_4, A_8, \cdots, A_{\sqrt{N}} \rightarrow$ Sliding-window EXP3 algorithms with $S = \{1, 2, 4, \cdots, \sqrt{N}\}$ (these are the sizes of windows), and there are $\log(N)$.
4: Play according to EXP3 on $\sqrt{N} + \log(N)$ experts which are $Q \cup \{A_1, A_2, A_4, \cdots, A_{\sqrt{N}}\}$. $A_k = \{h^*_t, h^*_{t-1}, \cdots, h^*_{t-k}\}$.

This is just to give you an idea about what is going on in research in online learning. When you have a smaller window size, the regret of EXP3 is reduced. This is intuitively why you want to vary window-size when you don’t need as many experts.

2.3 Conclusion

These are all computational bounds and not error bounds. In terms of computational efficiency, if we can do optimization efficiently, then we can do statistical learning efficiently by Vapnik. Efficient online learning implies efficient statistical learning. We also saw that having an efficient online learning algorithm allows you to optimize the offline problem. However, there is a $\sqrt{N}$ lower bound when you are given an offline optimization oracle to solve online learning. It does not give you nothing however, you can speed things up given offline optimization oracle.

3 Course Summary

In this course, we studied

1. Statistical learning theory. This is the basics of learning theory, some people still do research in this area though it is not that much. There are a lot of concentration bounds, VC-dimension, infinite hypothesis classes, the fundamental theorem, and so on. These are the classics.

2. Online model and regret minimization. We talked about how this is a more general model in terms of problems that do not have a statistical structure. We are living in an adversarial world. We talked about portfolio optimization, which is a special case of an adversarial world, and how to use convex optimization techniques. Other problems: routing and so on, that have efficient structure, and solved them efficiently.

3. Converting Regret $\Rightarrow$ Online model.

4. Boosting. In the statistical setting, how to take weak learners in the statistical setting to strong learners in the statistical setting. We also mentioned online boosting.

5. Games and Duality, and the relationship between online learning and these topics.

6. Computational Issues. (What we covered in this lecture.)
This is pretty much the picture that we looked at this semester. Now, what did we not cover? These are a good idea to look at for your project. Statistical learning theory is vast, and we did not have time to cover the following topics.

1. Spectral methods*. Notoriously, they do not have theory behind this. Recently there has been a lot of advancement: Tensor learning. A lot of theoretical analysis that helps you analyze these things efficiently. This is an emerging body of theoretical work which we did not go into.

2. Probabilistic modeling. We mostly took the discriminative view for learning. There is also a generative approach. We can model the universe as a Gaussian distribution over matrices, and try to infer. There is some advancement here (Professor Arora), but it is still an open field.

3. Reinforcement Learning*. You have studied dynamic programming before, modeling the Bellman equation and so on. A lot of theoretic work is devoted to studying Markov processes and so on. How to learn with state, solve the game of chess, and so on. The state of the art in terms of machine learning is not there yet though.

4. Deep Learning/Unsupervised Learning. This is very hype right now, along with optimization. You don’t get feedback in unsupervised learning. However it is very successful in practice.

That is pretty much the landscape in terms of machine learning. There are specialized courses for these topics however which you can take if you are interested.

4 Question and Answer Session

I guess it is time for some Q&A.

**Why is convexity such a powerful condition? How did people decide that convexity was the important assumption to make? Can you give some intuition for why a condition like convexity should be important for getting all these nice bounds?**

**Answer:** Nature behaves in a way such that we can solve optimization problems that are convex (these are efficient). We are just talking about offline algorithms. Under very weak conditions, you can solve these in polynomial time. In some sense, these are the weakest conditions you can solve in polynomial time. In some sense (this is debatable - I tend to think it is true), almost everything you can think of that you can solve inefficiently, you can solve as a convex optimization problem. The reason it pops up in learning is because it pops up in optimization. It is even more difficult than solving offline problems. Even if you remove that part, what we know how to do in polynomial time is convex optimization. There are very easy problems that are not convex that are very hard to optimize - for instance, Max-Cut. Max-Cut is NP-hard. You can model it as follows: \( i \in V, x_i \in [-1, 1] \). Let us say you want to maximize over the edges \( x_i, x_j \). If you minimize, you can solve this as a convex program. If you maximize, it is already NP-hard! Just two variables. The importance of convexity was realized by Tucker and some other people at Princeton. von Neumann proved the duality theorem in games. And then the Ellipsoid method from the Russians (this was in the 50s). In the 30s they talked about convex optimization in Fine hall, right here.

**What do you think are the most important open problems in machine learning?**

**Answer:** What is even the goal of theoretical machine learning? What is learnable that is independent from the brain? We want to do the same thing in terms of functionality, but only based on mathematical reasoning. The dream of AI has been to build a thinking machine, ML is more modest - only model functionality. My personal favorite is the problem of classification, which in my opinion is unresolved. It is still not nearly as good as we want it to be. We can classify documents with linear...
classification. This is closer to the dream of machine learning: we can classify images. Deep learning and classification, this is getting closer to the dream of machine learning. The main open problem would be to classify large-scale videos, huge-scale data. Taking a video of a classroom requires good representation. Linear classification fails at high amounts of data. We need to represent videos well. Give a theoretical framework for doing it in large scale dimensionality reduction. It seems to be making progress. But we don’t have theoretical justification yet, but I am interested in building a theory for that.

Why is the theory of deep learning so hard?
Answer: First of all, the objective of unsupervised learning is not well-defined. What does it mean? No one defined it. They took a model - took the brain, let us copy it. That is pretty much how things went by. There is some theory there, but it is more of an empirical theory. There is no well-defined objective of what it means, aside from intuitive. What is the mathematical definition of what we want to find. These are also non-convex optimization problems - by complexity theory we have no reason to say what will be efficient.

Info theory revolutionized communication. How do you feel about the location of ML in practice and in general with respect to info theory? (Shannon etc)
Answer: I think we are as far as can be. We are not even close. In other areas, at least we know what we are looking for. Even the basic definitions are still not clear. In my opinion, this is the most exciting field to study - the questions are so profound, answers are so far away, but we are making a lot of progress.

Do you think there will be new mathematics generated in the study of machine learning?
Answer: There are two headways that are happening in machine learning. Currently the experimentalists are dominating the field. We can build a self-driving car, but don’t know why it happens. Rob Schapire invented boosting, that led to huge advancement in the ways we can analyze and so on. Deep learning is an example of practice. Here is a silly guess: if I were to guess, I would say no. It is just a matter of combining different tools. For P = NP, we need a whole lot more of mathematics field. I think this is more elementary - everything can be explained to an undergrad. I don’t see any indication that we need a new type of mathematics, this is more a matter of algorithmics.

Where do you see the results coming out of applying optimization oracles?
Answer: There is a framework called Contextual Bandits. This is very popular in the web literature - how do you take search results and rank them, give search results, ads, etc. Web is an essentially online platform. It is an online model. There are many heuristics for things. Relaxing guarantees, not as strong definitions of regret, and so on.

In this class, in statistical learning or online learning, we also consider worst case. Is there an approach to consider average case?
Answer: Well, statistical learning is average case. Distribution could be arbitrary, you could relax distribution from a specific kind. You already have tight bounds though, so why would you do it? Optimization oracle this also holds for. In some sense, we already have a strong result. The rest of the course, online models were more adversarial, but you don’t lose much. Is there a need even to consider distributions? Sometimes there is. I mentioned probabilistic modeling and unsupervised learning and so on. These kind of models assume a model about the world. They have very hard computational problems. The only way people know how to move forward are by making assumptions about the distribution. There is literature on that, but even assuming Gaussian models, and so on, you typically don’t get efficient algorithms. One case in which theoretical guarantees are known - spectral methods. There, many times people make assumptions about the moments of a distribution - if they are separated
You mentioned weaker notions of regret. What else is there?
Answer: People have actually typically looked at stronger notions. There is for weaker notions, you can look at random permutations of examples. The adversary chooses any worst case sequence, but the objective is still regret.

What is the relationship between the competitive ratio and regret?
Competitive ratio is the ratio between online algorithm performance and offline algorithm performance - assuming we get all the data in advance. In particular what I want to know is why do both of these metrics exist, how are they related, and when is one used over another?
Answer: Competitive ratio came out of theory literature. They analyze the performance of the algorithm compared to best dynamic algorithm. It’s only useful in specific cases. The foundational work was done by Tarjan - it is very useful for data structures. Since then it has been extended, and has not really found any applications. In the real world, you cannot really compete with any changing comparator. Best you can do is with respect to unchanging. In the problems we discussed in this class, regret seems to be the right metric.

How do you think we should handle cases where the problem is obviously non-convex? Is it better to try to reduce it to a convex problem, or do you find other approaches interesting/useful as well (for instance, how do we tackle theorertizing deep learning)?
Answer: I can think of two ways: make probablistic assumptions on the world. Data comes from a distribution which makes it on average convex.

Approximation theory - then try to solve a different problem which is more general, and bypass the hardness of the problem. If we had done matrix completion, we would have seen this. Reconstructive and non-reconstructive.

For example, what happens in a video - how would you begin to formalize a problem like that theoretically?
Answer: There are two important points - what is the benchmark of performance? People don’t see that much from theory. We want to do better than a human - we don’t want to solve the problem, just do better than human. For classifying images, we have ImageNet. You can classify what an image contains. The description that the automated methods give are very close to how well a human does. How would you represent it? We described text classification - convert text to a vector, what appears and what doesn’t, find a linear classification (spam). For images, you can do dimension reduction with a deep net. Video maybe you need a sequence of vectors. But yeah, that is part of the problem too. There are ways to represent videos succinctly.