Algorithmic Statistics
Lecture 2: Course Overview & Linear Predictors

Samuel B. Hopkins

Last time we saw that even very basic high-dimensional statistical problems can be intractable because they require too many samples in high dimensions. Today we will see a different phenomenon: another basic statistical problem where the roadblock is not samples but computation. But first we will do a course overview.

1 Course Overview

1.1 Modules

The course will be divided into three main modules and one mini-module.

Mini-Module: Introduction and Sparse Recovery (3 lectures) Introduces some of the key themes for the course – learning in high dimensions requires assumptions, and under assumptions we can obtain interesting and useful algorithms with strong provable guarantees.

Module I: Graphical Models (8 lectures) Graphical models provide a powerful language to express assumptions under which learning might be tractable. We will introduce the basic language and explore algorithms for a variety of statistical tasks: learning, sampling, marginalization.

Module II: Spectral Methods and Beyond (9 lectures) We will explore algorithms for a wide range of unsupervised learning tasks – clustering, matrix completion, robust statistics – unified by the theme of eigenvectors and eigenvalues of matrices and tensors associated to data. We will also see several ways to go beyond such algorithms.

Module III: Computational Complexity of Statistics (4 lectures) We will switch gears to discuss computational hardness in the context of high dimensional statistics.

1.2 Techniques

Here is a not-comprehensive list of some of the algorithmic techniques you can expect to learn about in this course. The goal is to teach you a wide range of "first lines of defense" you should try when tackling a fresh problem – general algorithmic techniques with wide applicability.

- Linear programming
- Spectral clustering

- Semidefinite programming
- Method of moments/tensor decomposition
- Sparse recovery

1.3 What this Course Is Not

This is unabashedly a theory course. We are focused on what learning/statistics problems can be solved in principle, which are not, and why. Therefore, we will not focus on implementation aspects of algorithms, numerical stability, parallelization, GPUification, etc. This is not a course on deep learning (even though there is more and more interesting theory for deep learning), LLMs, or transformers.

Many of the algorithms we will analyze in this course are not the ones used in practice to solve the corresponding problems. Often, analyzing simple but somewhat slower polynomial-time algorithms is much mathematically cleaner, and offers better conceptual insight, than analyzing the faster algorithms used in practice. That said, there is a whole industry of trying to analyze the iterative methods used in practice to provide provable guarantees for alternating minimizing, expectation-maximization, non-convex gradient descent, etc. We will not discuss this very much, only because there is only so much time in the semester. It could be fodder for course projects.

2 Linear Classifiers

Returning now to the plot, let's discuss prediction with linear functions, one of the core paradigms in machine learning. In machine learning and statistical learning theory courses, it's common to paint the picture that prediction via linear functions is largely a tractable and well-understood task – there is a standard progression of least-squares linear regression, logistic regression, SVMs, kernel methods, and finally deep learning presented as linear classification with learned features. We will briefly review some of the basics and then add some color to the picture, showing that whether prediction using linear functions is tractable depends very delicately on how the problem is phrased, highlighting the role that *a priori assumptions about the world* play in how we interpret what our statistical learning algorithms produce.

2.1 Linear Classifiers

Let's do a bit of basic statistical learning theory. Suppose that there's a population of d-dimensional individuals each with a "true" label in $\{\pm 1\}$. We can represent a random draw from this population by a jointly-distributed pair of random variables (X, Y) where X has values in $\mathbb{R}^d$ and Y has values in $\{\pm 1\}$. Suppose we are interested in using samples to learn the best linear classification rule for the population. The most natural measure of "best" is the number of mistakes made by the rule. For a function $f : \mathbb{R}^d \to \{\pm 1\}$ define

$$L(f) = \mathbb{E}_{(X,Y)} \mathbf{1}(f(X) \neq Y)\,.$$

A linear classifier, or halfspace, is a function f defined by some $w \in \mathbb{R}^d$, with $f(x) = \operatorname{sign}(\langle w, x \rangle)$. Suppose we want to find $\arg\min \mathbb{E}_{(X,Y)} L(f)$, where the minimization is taken over all linear classifiers? Two questions arise:

1. How many samples are needed to get close to the minimizer? Is this problem like uniformity testing, requiring $\exp(\Omega(d))$ samples? Or something better?

2. Is there a computationally efficient algorithm which processes the samples "optimally", whatever that means, to extract such a linear predictor f?

The first question is answered by classic statistical learning theory, in particular the theory of VC dimensions.

Lemma 2.1. *Suppose $(X_1, Y_1), \ldots, (X_n, Y_n)$ are iid copies of (X, Y). Let $\hat{f} = \arg\min \frac{1}{n} \sum_{i \le n} \mathbf{1}(f(X_i) \neq Y_i)$ be the empirical loss minimizer. Then*

$$\mathbb{E}_{(X_1,Y_1),\ldots,(X_n,Y_n)} \mathbb{E}_{(X,Y)} L(\hat{f}) \le L(f^*) + O\left(\sqrt{\frac{d}{n}}\right),$$

where $f^* = \arg\min L(f)$ is the best linear classifier for the overall population.

So unlike uniformity testing, this problem is easy from a sample-complexity standpoint – with $O(d/\varepsilon^2)$ samples it's possible to compete with the best linear classifier for the whole population, up to ε extra mis-classification probability.

However, once computation enters the picture the story changes. To actually obtain $\hat{f}$ we would need to minimize the non-convex function $\frac{1}{n} \sum_{i \le n} \mathbf{1}(f(X_i) \neq Y_i)$. This on its own should not make us throw up our hands – we will see plenty of nonconvex optimization problems which can still be solved in polynomial time. But in this case it's bad news – non-convexity informally suggests some computational hardness, but an NP-hardness result really seals the deal.

Theorem 2.2 ([GR09])**.** *Unless $NP = RP$, there is no polynomial-time algorithm which takes as input n independent samples from a distribution on $\{\pm 1\}^{d+1}$ and decides with success probability $2/3$ between the case that there is a halfspace which correctly classifies a 0.99 fraction of the distribution versus the case that the best halfspace correctly classifies at most a 0.51 fraction of the distribution.*

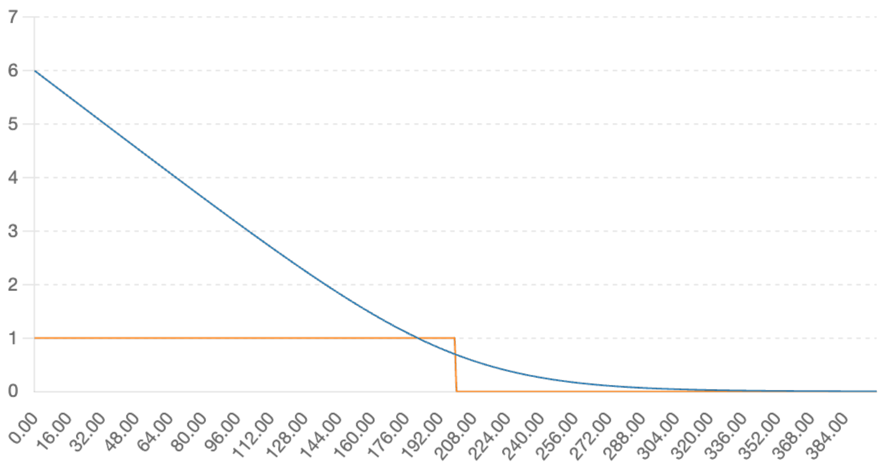
You can take a look at [AK98] for a simpler reduction which proves a weaker result in the same vein.

Of course, it would be a mistake to infer from this result that linear classification is hopeless. A better conclusion is that we need to think carefully about what the algorithms we actually use for linear classification – logistic regression, SVMs, kernels – are actually doing, and when we can/cannot trust the outputs they produce.

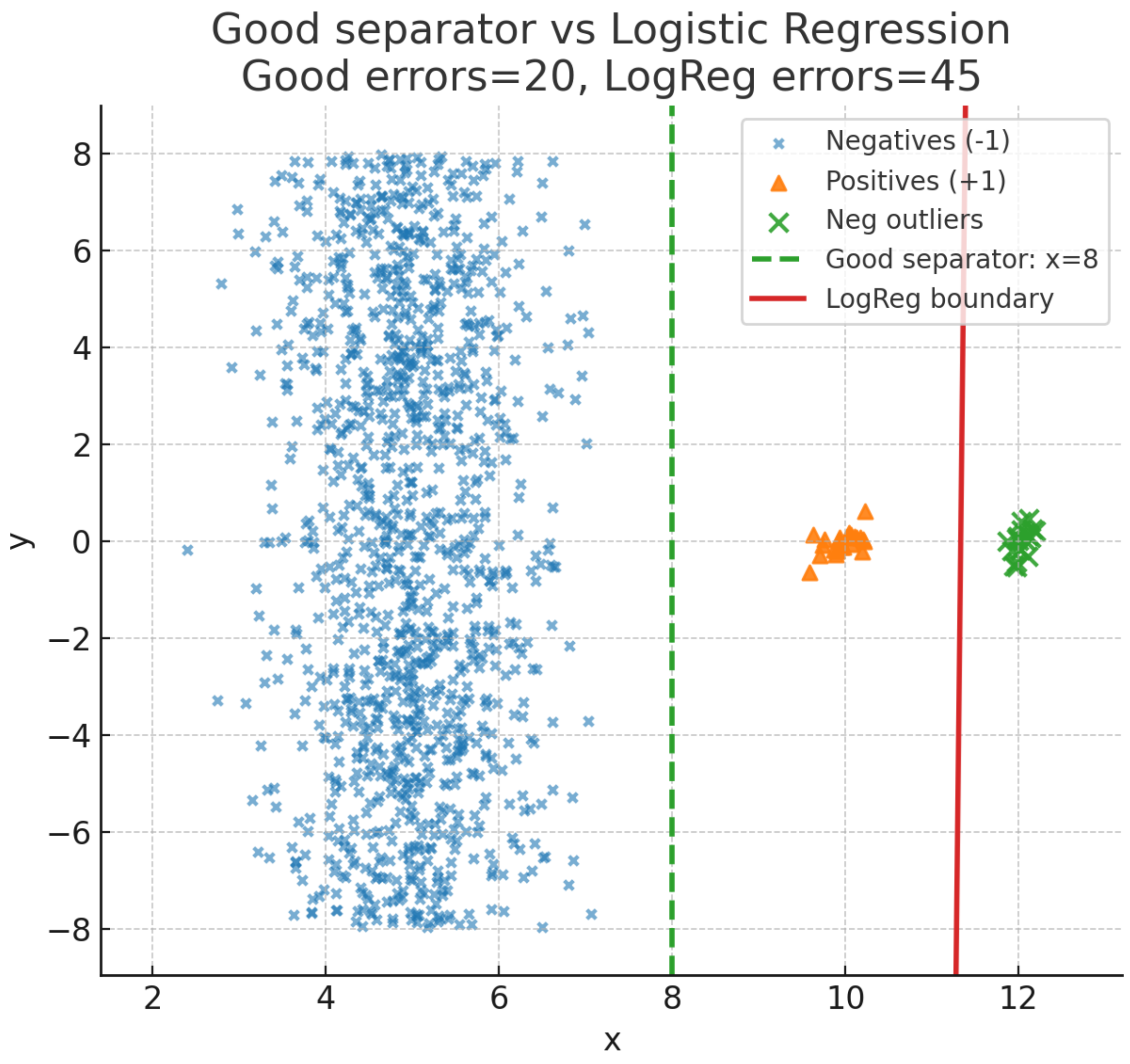
Let's discuss now one of the common approaches actually used to find a linear classifier – "convex surrogates". The idea is to replace the $0/1$ loss with a function which is convex on $\mathbb{R}^d$ so we can actually minimize it in polynomial time. There is a whole industry of studying properties of various such convex surrogates; I'll talk only about one very popular one, *logistic loss*. We define

$$\ell(x, y, w) = \log(1 + \exp(-y \cdot \langle w, x \rangle)).$$

Here is a plot of logistic loss compared to $0/1$ loss:



However, switching to a convex loss comes with some costs. One cost is that of course even if we had infinitely-many samples, the population loss minimizer for logistic loss might be very different from the optimal halfspace for $0/1$ loss. Here's a simple example to illustrate:



The logistic loss really hates having the decision boundary far from the small cluster of negative samples on the right. It is willing to mis-classify all the positive examples just to avoid having the decision boundary far on the wrong side of those negative outliers.

A second cost of this switch is that to get an analogue of the finite-sample generalization result Lemma 2.1, we need to impose some additional assumptions on the underlying distribution, and restrict the set of halfspaces under consideration. One set of assumptions which will work is to (a) assume that the weight vector w lies in a ball of radius R, and (b) that the second moment of the distribution of X exists. Then we can obtain a slightly more advanced result which would be proved in a statistical learning theory course using tools like uniform convergence of Lipschitz losses and Rademacher complexity:

Lemma 2.3. *Let $B_R \subseteq \mathbb{R}^d$ be the ball of radius R about the origin. Let (X, Y) be jointly distributed with $X \in \mathbb{R}^d$ and $Y \in \{\pm 1\}$, such that the second moment of X exists. Let $(X_1, Y_1), \ldots, (X_n, Y_n)$ be iid copies of (X, Y). It is possible to obtain in polynomial time from these iid copies a vector $\hat{w} \in B_R$ such that*

$$\mathbb{E}_{(X_1,Y_1),\ldots,(X_n,Y_n)} \mathbb{E}_{(X,Y)} \ell(X, Y, \hat{w}) \le \min_{w \in B_R} \mathbb{E}_{(X,Y)} \ell(X, Y, w) + O\left(R \cdot \sqrt{\frac{\mathbb{E}\|X\|^2}{n}}\right).$$

How to interpret the quantity inside the big-O? One reasonable interpretation is to think of $R = O(1)$ with respect to d and the entries of X having magnitudes also $O(1)$. In this case, for a random X not too correlated with a weight vector w, we expect $|\langle w, X \rangle| \approx O(1)$, and hence that the logistic loss lies in the region where it reasonably well approximates $0/1$ loss. In this case, the expression in the big-O is simply $\sqrt{d/n}$, same as in the $0/1$ loss case.

The message here is not to hate on logistic regression – it's a great tool. I just want to point out that it was the need to be able to compute our classifier efficiently which leads to some concessions on interpretability of the resulting $\hat{w}$. When later in the course we use logistic regression as a tool, we will need to be careful to interpret what it means for the $\hat{w}$ we find to compete with the best w according to the logistic loss, making sure that the data we are using logistic regression on are such that this is a sensible loss function.

References

[AK98] Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1-2):237–260, 1998.

[GR09] Venkatesan Guruswami and Prasad Raghavendra. Hardness of learning halfspaces with noise. *SIAM Journal on Computing*, 39(2):742–765, 2009.