LECTURER: JACOB STEINHARDT SCRIBE: KIRAN VODRAHALLI

Contents

1 Introduction

Say you have a central system that says it can locate vehicles with 99% accuracy that they are trying to distribute to lots of users; different users have different test distributions. We will mostly be focusing on transfer learning. How do you know you can match the training distribution to general distribution? We would like to give an abstract guarantee and general contract that works. The issue is that the training and test distributions could be quite different.

Google Flu Trends was this thing a few years back, where Google would try to predict how much flu there was in different areas of the U.S. based on search queries. This was a supervised task with the CDC data. They built a statistical model which worked pretty well for a while, but at some point, the CDC called up Google and said that Google is overestimating by a factor of 2 (and CDC is basically ground truth). They started with a ton of text features in search queries, and did feature selection, and then did some l_1 regularized logistic regression (we think).

This is a time series, but the goal is eventually to not know ground truth. The problems which are maybe hard to pick up with time series analysis: Google was pushing changes to Autocomplete software as this happened. So there are things like seasonal changes, but there are also cases where autocomplete assumptions are completely changed. You could probably model that, but you'd want to be able to detect features which don't work anymore. "Novel event" more than rare event.

I would argue that no matter how much good statistics you do, you'll still have weird stuff. You want to have a robust contract and detect when something bad is happening, and when nothing bad is happening. You can't hope to always do well, but you can hope to know how well we're doing. Unlabeled data could help us.

1.1 Unsupervised Risk Estimation

Definition 1.1. Main Problem: Given unlabeled data $x_1, \cdots, x_m \sim p^*(x)$, estimate the **labeled risk**

$$R(\theta) = \mathbf{E}_{x,y\sim p^*}[L(\theta; x, y)]$$

where p^* is nature (joint distribution over data and labels, but you only observe the marginal distribution over the data) and θ is your model.

We're basically not going to have any labels. If training and test are exactly the same, but two labels flip places, we won't be able to detect that. But in other cases, with generative assumptions about the distribution.

In general, the distribution could be arbitrary. But if the training and testing distributions are similar, maybe you could do density estimation. This might not work so well if there are large deviations. Blitzer and Kakade have some nice work on this.

It's possible to estimate the risk in unsupervised models if the model is well-specified, but mis-specified models are very common. CCA and multi-view learning were some work dealing with this.

We would like to look at the pieces of data which affect our loss function. We don't care about the full distribution of all our inputs; we just care about the scores assigned to each class. In the case of the $0-1$ loss, if you assume you have multiple classifiers (train two or three different models for the same problem) making independent errors/almost independent errors, you can figure out how well you're doing.

The classifier score could be distributed as a mixture of Gaussians: How do you get the classifier score; as long as scores are not too dependent, you may be able to get Gaussinity. So it could be Gaussian, but it could also not be (neural network experiment on MNIST: sometimes histogram of data points looks Gaussian, sometimes not).

1.2 Structural Assumptions

In this work we're trying to get away from distributional assumptions. It's hard to come up with strong a priori reasons why they would hold. A more robust thing than distributional assumptions are **structural assumptions** about the conditional independence. Predictive models don't hold up over time, so what stays constant?

If you have some model that's supposed to predict things well, and I want it to tell me when things change, that's one possible situation. Let's say I want a good part of speech tagger, and there's ten I could download. If I don't want to hand label myself, I'd like the part-of-speech models to tell me themselves how well they think they'll do.

We will show how to estimate $R(\theta)$ from unlabeled data using a 3-view assumption using tensor decomposition and the method of moments. Once you know the risk of a model, we can optimize. From the perspective of learning, the hard part about unsupervised learning is figuring out what the risk is in the first place. You can do domain adaptation, or you could boostrap a model from a few labeled examples. You need a little bit of signal to break ties. If you did full unsupervised learning, you could maybe permute classes around, so we also provide a Bayesian hypothesis test to see if you recovered the right permutation of classes.

2 Framework and Approach

We're assuming we're doing multiclass classification with k labels and 3 independent views. We need $L(\theta; x, y) = A(\theta; (x_1, x_2, x_3)) - f_1(\theta; x_1, y) - f_2(\theta; x_2, y) - f_3(\theta; x_3, y)$ where you have three sets of blocks x_1, x_2, x_3 as inputs (observations). As long as the loss decomposes in this way (a constraint on the model and the loss function), we're good. I would class this a starting point; this is maybe the simplest set of structural assumptions where you can make headway. The three-views is a way to get a provable algorithm. We're just going to require that this structure remains. We want a provable algorithm even in the case where things change a lot. In many settings of interest, you won't have this decomposition. The agenda should be have more general decomposition. It is sort of surprising that we can do something.

We're assuming three views, which is a stronger assumption than something like CCA which has two views. So for instance, consider audio, text, and video representation of data, and where you believe there

is some hidden low-rank state before. We do not model $p^*(y|x)$ or $p^*(x)$. Each x_i could be complex and high-dimensional.

Tensor methods take this three-view structure and make some distributional assumptions. What we're doing is taking a 3-view structure and instead of using that to estimate some generative model, we're trying to estimate the risk. The approach still is method of moments. We write down equations for the risk $R(\theta)$ in terms of observable moments of x. By solving these equations, we can recover $R(\theta)$ even without a model for $L(\theta; x, y)$. Draws on recent developments in leraning theory (Anandkumar). This is also reminisicent of classical approaches in econometrics (Hansen, Newey and McFadden).

Empirically, there's a general sense that method of moments is more robust than maximum likelihood. It's also the case that you can prove something along these lines (linear regression; MLE and method of moments agree, but if your noise distribution is uncorrelated with x, in that setting you can show that you can correctly estimate your parameters). The correct way to generalize it is to think of it as a method of moments estimator. There's a broader field called semiparametric estimation; method of moments is one way to do this. At least one of these economists got a Nobel prize for generalized method of moments, they showed it's asymptotically efficient. All the stuff they apply it to simple models. They don't pay attention to dimensional scale though.

3 The Main Theorem

Theorem 3.1. *Suppose the* 3*-view assumption holds. For tasks with* k *labels, we can estimate the risk* $R(\theta)$ *to error* ϵ *using* $\text{poly}(k)/\epsilon^2$ *samples. This holds even if* θ *has dimension* $d >> k$*. The number of samples is independent of model complexity.*

You need bounded second moments of your risk, but independent of hypothesis class. The classes also can't be arbitrarily similar to each other. There's no VC-dimension; the reason you should expect to be able to get something like this, since you only have a single model, you're only trying to get a single point-wise estimate. We're given a model, and now we want to know how well it does on a new distribution. You just want to know if your model is doing well here. If you want to use it for optimization, then you need uniform convergence, and you'll get a $d \log d$ in the bound.

We do this with esimtation via method of moments:

$$\mathbf{E}[L(x, y)] = \mathbf{E}[A(x)] - \sum_{v=1}^{3} \mathbf{E}[f_v(x_v, y)]$$

So we make these variables f_1, f_2, f_3. Now we have four unknown variables including y. Let's just look at all the model scores for each label $1 \to k$. The key is that there's a low rank structure:

$$\mathbf{E}[f_1 f_2^T] = \sum_{y=1}^{k} p^*(y) \mathbf{E}[f_1|y] \mathbf{E}[f_2|y]^T$$

So after tensor decomposition, you get $\mathbf{E}[f_1|y]$ and so on. From this decomposition, using standard tensor decomposition methods, we can recover conditional score distributions efficiently using tensor decompositions. Since this is all we need, then we're done. The technical trick is that we can decompose our risk in terms of this low dimensional linear function that pops out of the tensor decomposition.

You can just look at $f_1 \otimes f_2 \otimes f_3$ etc. So you need $k \geq 2$. The rank will be k in general. For each y you get a rank 1 term.

If you have two views, it's only unique up to rotations.

3.1 Identifying the Correct Permutation of Classes

If at training time you have a bunch of dogs and cats, and at testing time the definition of dogs and cats flips. We can only recover $R(\theta)$ up to permutation of classes. We can compute the optimistic risk: $\tilde{R}(\theta) = \min_{\sigma \in Sym(k)} \mathbf{E}[L(x, \sigma(y); \theta)]$. Is $\tilde{R}(\theta) = R(\theta)$? We will come up with a Bayesian test. Assume that either p^* induces low risk $R(\theta)$ or p^* is drawn at random from μ, a prior. We want a test with a low false positive rate in both cases. The rough theorem that we have is that for some small $\delta > 0$, if optimistic risk $\tilde{R}(\theta)$ is better than chance by at least δ, then $\tilde{R}(\theta) = R(\theta)$ with high probability. Note that δ depends on μ but is $<< 1$ in many cases.

If we're doing worse than chance, then our optimistic risk might tell us we're doing better than we thought, but we'll know if we're doing worse than chance. If we're doing better than chance, then we're almost certainly correct.

We can also do learning: Let's minimize over θ:

Theorem 3.2. *Givena seed model θ_0 such that $R(\theta_0) = \tilde{R}(\theta_0)$ we can learn the optimal θ to error ϵ given $d \log(d) polylog(k)/\epsilon^2$. The optimistic risk is the minimum of a bunch of functions, we can basically learn.*

4 Conclusion

There are two key ideas here. If you want contracts for a machine learning system, you need to find some invariant structure. Sometimes it doesn't need to be a strong distributional assumption, and once you have these contracts, you can do learning. Maybe we need to take a middle ground between generative and discriminative approaches. What we really care about is predictive performance.

Maybe what you should do is change what you're doing: Optimize how well your neural net can predict your hidden variable, and if you can estimate the hidden variable, then you're set.

An open question: Can we identify a broader class of assumptions that enable risk estimation? A possible principle is that to learn, it suffices to understand the structure of a domain.