Bias Sampling Correction for Supervised Learning

A Probabilistic, Bayesian Approach

Max Sklar Ely Spears

August 13, 2019

Abstract

abstract

1 Introduction

Introduction

1.1 Motivation

motivation
mention the Foursquare Attribution system
mention Maalouf paper

1.2 Previous Work

previous work
mention most bibliography items here
Yan Lecun' Energy Paper

2 Definitions and Conventions

This section outlines a typical setup for a probabilistic supervised learning problem. The terminology and variables will be reused in subsequent sections of this paper.

Let X represent the input space.
Let Y represent the output space.

The dataset D consists of N examples of pairs $(x, y) \in (X \times Y)$ from which we want to learn to predict an output $y \in Y$ from the corresponding input $x \in X$. This dataset may be generated by an oracle from which we can ask for an arbitrary number of examples, or it might be a small and limited collection. In either case, we assume that at this point in the problem our analysis will be based off of just these N examples.

For all $n \in N$, which is shorthand for $n \in (0, 1, 2, \ldots, N-1)$, label the specific instance (x_n, y_n).

Now to select a prediction function, we create a hypothesis space H which contains all the predictors under consideration. In some setups, each hypothesis $h \in H$ is a direct function from X to Y, but here we assume that each hypothesis is a function of $x \in X$ and returns a *probability measure* over Y.

While the equations in this paper generalize to all probability measures over Y, we keep several cases in mind.

1. Finite output spaces, or classification. This probability measure is the most familiar as it is simply a finite sum that adds to one.
2. Discrete but infinite output spaces, where the probabilities that add to one form an infinite sum.
3. Infinite and Continuous output spaces, where each hypothesis corresponds to a *probability distribution function* (PDF) over Y

Note that in all these cases, the hypothesis assigns a number to each $y \in Y$ which can be used to compute the relative likelihood of each potential output. We can even run this analysis for an *improper probability distribution function*, for example one that assigns an unnormalized PDF score of 1 to each real number ($Y = \Re$) even though such a function cannot be normalizes as the integral is infinite.

Once the hypothesis space has been defined, the task is either to *select* a model or *sample* a model. In the former case of selection, we want to select a single model as the result of our analysis and in the latter case of sampling, we want to pull a series of models according to their relative probabilities in order to take into account model uncertainty.

In the simplest cases, for example if the hypothesis space H is finite and small, we will be able to calculate the probability of each $h \in H$ and either select the model with the highest probability, or sample from it using a random number generator. In general, the hypothesis space H could be very complex and a variety of methods for selection and sampling have been developed.

In all of these cases, we simply want to know the relative probability between two hypotheses, say h_1 and h_2. Therefore, we will use Bayes rule without the normalizing constant in order to find that relative probability of each hypothesis given the dataset D.

Bayes rule in full with the normalizing constant is given as follows:

$$\mathbf{P}(h_1|D) = \frac{\mathbf{P}(D|h_1)\mathbf{P}(h_1)}{\sum_{h \in H} \mathbf{P}(D|h)}$$

Bayes rule for relative probability is:

$$\mathbf{P}(h|D) \propto \mathbf{P}(D|h)\mathbf{P}(h)$$

The forward probability of the dataset under the model $\mathbf{P}(D|h)$ is the product of the probability of each instance label y_n being produced.

$$\mathbf{P}(D|h) = \prod_{n \in N} \mathbf{P}(y_n|x_n, h)$$

In terms of the probability function generated by this hypothesis, we get the following:

$$\mathbf{P}(D|h) \propto \prod_{n \in N} f_h(x_n, y_n)$$

Finally, the formula for relative probability is given by:

$$\mathbf{P}(h|D) \propto \prod_{n \in N} f_h(x_n, y_n)\mathbf{P}(h)$$

In most cases, it is more convenient to turn this into a *negative log-likilihood loss* function by taking $-\ln(\ldots)$ of the right hand side of this equation and getting:

$$L(h) = -\sum_{n \in N} \ln(f_h(x_n, y_n)) - \ln(\mathbf{P}(h))$$

We can simplify further by assuming each hypothesis to comes with its own log-likelihood loss function l_h where $f_h(x_n, y_n) \propto e^{-l_h(x_n, y_n)}$ and for the prior $\mathbf{P}(h)$ to be reduced to a regularization function $\mathbf{r}(h)$ where $\mathbf{P}(h) \propto e^{-\mathbf{r}(h)}$. With that in place, the final form of the loss function is as follows:

$$L(h) = \sum_{n \in N} l_h(x_n, y_n) + \mathbf{r}(h)$$

This form is usually most convenient when it comes to model selection or sampling, through techniques such as hill climbing, gradient descent, or Markov Chain Monte Carlo.

3 Stating the Problem

In the previous section, we derived a probability distribution and loss function for a probabilistic supervised learning problem. A key assumption is that the dataset D is not sampled in any bias way. In an extreme example, suppose that the output set Y consists of 3 classifications $\{yes, no, maybe\}$ and whenever the result of one of the classes, say *no*, appears in the dataset, the results were thrown away. The model would learn that $\{yes, maybe\}$ were the possible responses, and would never put any significant amount of probability towards the *no* option!

Fortunately, if the sampling function is known, then a probability distribution and loss function can be computed for the remaining data. The sampling will affect what that final loss function knows about the data, but the results in this paper will represent the best-guess from the data available.

In the example above where all the *no* options were sampled out, the learner would have no way of knowing when to put high probability on *no* but it will also know - because of the sampling function - that many results will be no and will rely on the prior to assign probabilities.

The formal statement of the problem is as follows:

Suppose that the dataset D was generated from an even larger dataset $D+$of unknown size with a sampling function $\mathbf{s} : X \times Y \to [0, 1]$. For each element $(x, y) \in D+$, the sampling function $\mathbf{s}(x, y)$ is calculated and returns the probability that this example will be in the dataset D. Now from just the dataset D and knowledge of $\mathbf{s}$ we can calculate the posterior probability and loss function.

4 The General Solution

The updated version of Bayes rule is as follows:

$$\mathbf{P}(h|D, \mathbf{s}) \propto \mathbf{P}(D|h, \mathbf{s})\mathbf{P}(h)$$

As before, we must calculate the forward probabilities.

$$\mathbf{P}(D|h) = \prod_{n \in N} \mathbf{P}(y_n|x_n, h, \mathbf{s})$$

Note that in the expression $\mathbf{P}(y_n|x_n, h, \mathbf{s})$, the input x_n, model h and sampler $\mathbf{s}$ are all taken as given, and the question is the probability of producing the label y_n under these circumstances.

Also note that if there is no sampling function, or $\mathbf{s}(x, y) = 1$, then

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) = \mathbf{P}(y_n|x_n, h) = f_h(x_n, y_n)$$

and this reduces the original problem.

To understand how the value of this expression changed under a sampling function, the proper mental model is as follows:

1. Consider a specific value of x_n as given for which we want to assign a label.
2. Under the given model h, we have a probability measure over Y. This probability measure is determined by the function $f_h(x_n, y_n)$ or $l_h(x_n, y_n)$
3. Sample from that probability distribution, and make this a candidate for y_n, called y_n^*
4. Compute the sampling rate $\mathbf{s}(x_n, y_n^*)$ and use that rate to probabilistically determine whether or not y_n^* is accepted.
    (a) If it is accepted, set $y_n = y_n^*$, and we have successfully determined the label for x_n

(b) If it is not accepted, return to step 3 to generated another candidate.

The assumption here is that the sampling function will eventually halt which means that it cannot be equal to 0 for every potential candidate of the given x_n. This is a fair assumption because if it were false, x_n would not appear in the dataset D.

The generative description above can be used to product the following recursive equation. Note that the first term in the sum $\mathbf{s}(s_n, y)\,[y_n = y]$ represents the outcome of accepting the draw of y whether it is equal to y_n or not, and the second term represents a rejection of the draw of y in which case your probability measure reverts to the original problem. The term $[y_n = y]$ is the indicator function which is equal to 1 if true and 0 if false.

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) = \sum_{y \in Y} f_h(x_n, y)\left(\mathbf{s}(s_n, y)\,[y_n = y] + (1 - \mathbf{s}(s_n, y))P(y_n|x_n, h, \mathbf{s})\right)$$

One assumption of this equation is that the output space Y can be summed over which implies that its either finite or at least discrete. In the case of a continous Y, these equations will work just as well with integrals. But as before, because we only care about the relative probabilities of any finite subset of Y this form will suffice.

Now with some algebraic manipuation, we can solve for $\mathbf{P}(y_n|x_n, h, \mathbf{s})$ to get a proper formula:

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) = \sum_{y \in Y} f_h(x_n, y)\mathbf{s}(x_n, y)\,[y_n = y] + \sum_{y \in Y} f_h(x_n, y)(1 - \mathbf{s}(s_n, y))P(y_n|x_n, h, \mathbf{s})$$

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) = f_h(x_n, y_n)\mathbf{s}(x_n, y_n) + P(y_n|x_n, h, \mathbf{s}) \sum_{y \in Y} f_h(x_n, y)(1 - \mathbf{s}(s_n, y))$$

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) \left[1 - \sum_{y \in Y} f_h(x_n, y)(1 - \mathbf{s}(x_n, y))\right] = f_h(x_n, y_n)\mathbf{s}(s_n, y_n)$$

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) \left[1 - \sum_{y \in Y} f_h(x_n, y) + \sum_{y \in Y} f_h(x_n, y)\mathbf{s}(x_n, y)\right] = f_h(x_n, y_n)\mathbf{s}(s_n, y_n)$$

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) \left[1 - 1 + \sum_{y \in Y} f_h(x_n, y)\mathbf{s}(s_n, y)\right] = f_h(x_n, y_n)\mathbf{s}(x_n, y_n)$$

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) \left[\sum_{y \in Y} f_h(x_n, y)\mathbf{s}(s_n, y)\right] = f_h(x_n, y_n)\mathbf{s}(x_n, y_n)$$

$$\mathbf{P}(y_n|x_n,h,\mathbf{s}) = f_h(x_n,y_n)\mathbf{s}(s_n,y_n)\left[\sum_{y\in Y} f_h(x_n,y)\mathbf{s}(x_n,y)\right]^{-1} \tag{1}$$

For a continuous output space Y, that would be an integral rather than a sum. The easy computation on the sum/integral term depends on the structure Y, but is at least easy in the finite and small case.

Here we can check our assumption that if the sampling function were constant $\mathbf{s}(x,y) = p$ then we return to our old equation:

$$\mathbf{P}(y_n|x_n,h,\mathbf{s}) = f_h(x_n,y_n)\mathbf{s}(s_n,y_n)\left[\sum_{y\in Y} f_h(x_n,y)\mathbf{s}(x_n,y)\right]^{-1} = f_h(x_n,y_n)p(p^{-1})\left[\sum_{y\in Y} f_h(x_n,y)\right]^{-1} = f_h(x_n,y_n)$$

4.1 Formula for Negative Log-Likelihood

With equation (1), we now have all the tools needed to assign relative probabilities to the hypothesis space H. To get it in the form most helpful to form for machine learning algorithms, we derive a negative log likelihood loss function. Start with Bayes:

$$\mathbf{P}(h|D,\mathbf{s}) \propto \prod_{n\in N}\left[\mathbf{P}(y_n|x_n,h,\mathbf{s})\right]\mathbf{P}(h)$$

Then, use equation (1) to get it in terms of f_h:

$$\mathbf{P}(h|D,\mathbf{s}) \propto \prod_{n\in N}\left[f_h(x_n,y_n)\mathbf{s}(s_n,y_n)\left[\sum_{y\in Y} f_h(x_n,y)\mathbf{s}(x_n,y)\right]^{-1}\right]\mathbf{P}(h)$$

Finally, derive a negative likelihood loss function on h:

$$L(h) = \sum_{n\in N} -\ln\left[f_h(x_n,y_n)\mathbf{s}(s_n,y_n)\left[\sum_{y\in Y} f_h(x_n,y)\mathbf{s}(x_n,y)\right]^{-1}\right] + \mathbf{r}(h)$$

$$L(h) = \sum_{n\in N}\left[l_h(x_n,y_n) - \ln\mathbf{s}(s_n,y_n) + \ln\sum_{y\in Y} f_h(x_n,y)\mathbf{s}(x_n,y)\right] + \mathbf{r}(h)$$

5 Solution for Binary Logistic Regression

Binary logistic regression is a special case where the following is true:

The output space is binary: $Y = 0, 1$. The input space is a list of $|F|$ real valued features: $X = \Re^{|F|}$.

The hypothesis space consists of the weights and intercept of a logistic regression. $H = \Re^{|F|}$ where $(c, \mathbf{w}) \in H$ indicates that $c \in \Re$ is the constance intercept and $\mathbf{w} \in \Re^{|F|}$ is an —F—-dimentional vector of weights corresponding to each input feature.

Each hypothesis $(c, \mathbf{w}) = h \in H$ corresponds to the following probability distribution function:

$$f_h(x_n, y_n) = f_{c,\mathbf{w}}(x_n, y_n) = \frac{e^{y_n \cdot (c + \mathbf{w} \cdot x_n)}}{1 + e^{c + \mathbf{w} \cdot x_n}}$$

Therefore, using equation (1):

$$\mathbf{P}(y_n | x_n, h, \mathbf{s}) = \frac{f_h(x_n, y_n)\mathbf{s}(s_n, y_n)}{\sum_{y \in Y} f_h(x_n, y)\mathbf{s}(x_n, y)}$$

$$\mathbf{P}(y_n | x_n, h, \mathbf{s}) = \frac{\frac{e^{y_n \cdot (c + \mathbf{w} \cdot x_n)}}{1 + e^{c + \mathbf{w} \cdot x_n}}\mathbf{s}(s_n, y_n)}{\sum_{y \in Y} \frac{e^{y \cdot (c + \mathbf{w} \cdot x_n)}}{1 + e^{c + \mathbf{w} \cdot x_n}}\mathbf{s}(x_n, y)} = \frac{e^{y_n \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(s_n, y_n)}{\sum_{y \in Y} e^{y \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(x_n, y)}$$

$$\mathbf{P}(y_n | x_n, h, \mathbf{s}) = \frac{e^{y_n \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(s_n, y_n)}{e^{0 \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(x_n, 0) + e^{1 \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(x_n, 1)} = \frac{e^{y_n \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(s_n, y_n)}{\mathbf{s}(x_n, 0) + e^{c + \mathbf{w} \cdot x_n}\mathbf{s}(x_n, 1)}$$

Typically in these cases, we want to focus on the probability that $y_n = 1$, or the probability that the target condition is met. This target condition is also usually the rare condition that incentivized the use of adaptive sampling to begin with.

$$\mathbf{P}(y_n = 1 | x_n, h, \mathbf{s}) = \frac{e^{c + \mathbf{w} \cdot x_n}\mathbf{s}(s_n, 1)}{\mathbf{s}(x_n, 0) + e^{c + \mathbf{w} \cdot x_n}\mathbf{s}(x_n, 1)}$$

$$\mathbf{P}(y_n = 1 | x_n, h, \mathbf{s}) = \frac{e^{c + \mathbf{w} \cdot x_n}}{r_n + e^{c + \mathbf{w} \cdot x_n}}$$

Here, $r_n = \mathbf{s}(s_n, 0)(\mathbf{s}(s_n, 1))^{-1}$ is the true-to-false sample ratio for each instance n. This ratio is all that is needed to compute results for the binary logistic regression case. Note that for $r_n = 1$, we are left with the usual binary logistic regression formula.

For the prior, we can use a Gaussian distribution (aka L2, or ridge regression) with weight λ so:

$$\mathbf{r}(c, \mathbf{w}) = \frac{1}{2}\lambda(\mathbf{w} \cdot \mathbf{w})$$

We can put it together to compute a negative log-likelihood loss function. Some constant factors were dropped in this case.

$$L(h) = \sum_{n \in N} \left(\ln \left(r_n + e^{c + \mathbf{w} \cdot x_n} \right) - y_n \cdot (c + \mathbf{w} \cdot x_n) \right) + \frac{1}{2} \lambda (\mathbf{w} \cdot \mathbf{w})$$

The derivative on a single weight $\mathbf{w}_f$ can be computed to the following simple form, which is required for gradient descent and some versions of Markov Chain Monte Carlo.

$$\frac{\partial}{\partial \mathbf{w}_f} L(h) = \sum_{n \in N} \left(x_{n,f} \frac{e^{c + \mathbf{w} \cdot x_n}}{r_n + e^{c + \mathbf{w} \cdot x_n}} - y_n \cdot x_{n,f} \right) + \lambda \cdot w_f$$

$$\frac{\partial}{\partial \mathbf{w}_f} L(h) = \sum_{n \in N} x_{n,f} \left(\mathbf{P}(y_n = 1 | x_n, h, \mathbf{s}) - y_n \right) + \lambda \cdot w_f$$

6 Conclusions

Not sure there are any other than you can use this?

6.1 Future Work

Needed?

7 Notes: Potential references

Yan Lecuns Energy Paper?

The paper Ely found that has the equation in it for Binary Logistic: ¡ https://onlinelibrary.wiley.com/doi/full/10.1111/coin.121
¿. This has other citations that may be worthwhile for us to read, especially if they involve anything about the history of this modified logistic regression algorithm.

It could also be interesting to extend this to multinomial regression for multi-category prediction, in a case where each category has its own downsampling ratio.

Reference the most recently Foursquare patent. Find out when it is available.

The following references come from my 2014 paper on MLE for Dirichlet Prior. They should be changed to what we want for this paper

References

[1] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. the Journal of machine Learning research, 3, 993-1022.

[2] Dishon, M., & Weiss, G. H. (1980). Small sample comparison of estimation methods for the beta distribution. Journal of Statistical Computation and Simulation, 11(1), 1-11.

[3] Hariharan, H. S., & Velu, R. P. (1993). On estimating dirichlet parametersa comparison of initial values. Journal of statistical computation and simulation, 48(1-2), 47-58.

[4] Heinrich, G. (2005). Parameter estimation for text analysis. Web: http://www. arbylon. net/publications/text-est. pdf.

[5] Hijazi, R. H., & Jernigan, R. W. (2009). Modelling compositional data using Dirichlet regression models. Journal of Applied Probability & Statistics, 4, 77-91.

[6] Minka, T. (2000). Estimating a Dirichlet distribution. Technical report, M.I.T., 2000.

[7] Narayanan, A. (1991). Algorithm AS 266: maximum likelihood estimation of the parameters of the Dirichlet distribution. Journal of the Royal Statistical Society. Series C (Applied Statistics), 40(2), 365-374.

[8] Ng, K. W., Tian, G. L., & Tang, M. L. (2011). Dirichlet and Related Distributions: Theory, Methods and Applications (Vol. 888). John Wiley & Sons.

[9] Mosimann, J. E. (1962). On the compound multinomial distribution, the multivariate -distribution, and correlations among proportions. Biometrika, 49(1-2), 65-82.

[10] Robert, C. (2007). The Bayesian choice: from decision-theoretic foundations to computational implementation. Springer.

[11] Ronning, G. (1989). Maximum likelihood estimation of Dirichlet distributions. Journal of statistical computation and simulation, 32(4), 215-221.

[12] Sklar, M., Shaw, B., & Hogue, A. (2012, September). Recommending interesting events in real-time with foursquare check-ins. In Proceedings of the sixth ACM conference on Recommender systems (pp. 311-312). ACM.

[13] Wallach, H. M. (2008). Structured topic models for language. Unpublished doctoral dissertation, Univ. of Cambridge.

[14] Wicker, N., Muller, J., Kalathur, R. K. R., & Poch, O. (2008). A maximum likelihood approximation method for Dirichlet's parameter estimation. Computational Statistics & Data Analysis, 52(3), 1315-1322.

[15] Woodbury, M. A. (1950). Inverting modified matrices. Memorandum report, 42, 106.