What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features

Pasha Khosravi , **Yitao Liang** , **YooJung Choi** and **Guy Van den Broeck**
Computer Science Department
University of California, Los Angeles
{pashak, yliang, yjchoi, guyvdb}@cs.ucla.edu

Abstract

While discriminative classifiers often yield strong predictive performance, missing feature values at prediction time can still be a challenge. Classifiers may not behave as expected under certain ways of substituting the missing values, since they inherently make assumptions about the data distribution they were trained on. In this paper, we propose a novel framework that classifies examples with missing features by computing the expected prediction on a given feature distribution. We then use geometric programming to learn a naive Bayes distribution that embeds a given logistic regression classifier and can efficiently take its expected predictions. Empirical evaluations show that our model achieves the same performance as the logistic regression with all features observed, and outperforms standard imputation techniques when features go missing during prediction time. Furthermore, we demonstrate that our method can be used to generate "sufficient explanations" of logistic regression classifications, by removing features that do not affect the classification.

1 Introduction

Missing values pervasively exist in real-world machine learning (ML) applications, due to the noisy and uncertain nature of our environment. Consider autonomous driving as an example, where some sensors may be blocked at some point, leaving observations incomplete. Yet, even a small portion of missing data can severely affect the performance of well-trained models, leading to biased predictions [Dekel *et al.*, 2010; Graham, 2012]. In this paper, we focus on the classification task where features are missing during prediction time. To alleviate the impact of missing data, it is a common practice to substitute the missing values with plausible ones [Schafer, 1999; Little and Rubin, 2014]. As we will argue later, a drawback for such approach is that it sometimes makes overly strong assumptions about the feature distribution. Furthermore, to be compatible with multiple types of classifiers, they tend to overlook how the multitude of possible imputed values would in turn affect the trained classifier, which can result in biased predictions.

To better address this issue, we propose a principled framework of handling missing features by reasoning about the classifier's expected output given the feature distribution. One obvious advantage is that it can be tailored to the given family of classifiers and feature distributions. In contrast, the popular mean imputation approach only coincides with the expected prediction for simple models (e.g. linear functions) under very strong independence assumptions about the feature distribution. We later show that calculating the expected predictions with respect to arbitrary feature distributions is computationally highly intractable. In order to make our framework more feasible in practice, we leverage generative-discriminative counterpart relationships to learn a joint distribution that can take expectations of its corresponding discriminative classifier. We call this problem *conformant learning*. Then we develop an algorithm, based on geometric programming, for a well-known example of such relationship: naive Bayes and logistic regression. We call this specific algorithm naive conformant learning (NaCL).

Through an extensive empirical evaluation over five characteristically distinct datasets, we show that NaCL consistently achieves better estimates of the conditional probability as measured by average cross entropy and classification accuracy under different percentages of missing features than the commonly used existing imputation methods. Lastly, we conduct a short case study on how our framework can be applied to generating classifier explanations.

2 The Expected Prediction Task

In this section, we describe our intuitive approach to making a prediction when features are missing and discuss how it relates to existing imputation methods. Then we study the computational hardness of our expected prediction task.

We use uppercase letters to denote features/variables and lowercase letters for their assignment. Sets of variables $\mathbf{X}$ and their joint assignments $\mathbf{x}$ are written in bold. Concatenation (e.g., $\mathbf{XY}$) denotes the union of disjoint sets.

Suppose we have a model trained with features $\mathbf{X}$ but are now faced with the challenge of making a prediction without knowing all values $\mathbf{x}$. In this situation, a common solution is to impute certain substitute values for the missing data (for example their mean) [Little and Rubin, 2014]. However, the features that were observed provide information not only about the class but also about the missing features, yet

this information is typically not taken into account by popular methods such as mean imputation.

We propose a very natural alternative solution: to utilize the feature distribution to reason about what a predictor is expected to return if it could observe the missing features.

Definition 1. Let $\mathcal{F} : \mathbf{X} \to \mathbb{R}$ be a predictor and P be a distribution over features $\mathbf{X}$. Given a partitioning of features $\mathbf{X} = \mathbf{YM}$ and an assignment $\mathbf{y}$ to some of the features $\mathbf{Y}$, the *expected prediction task* is to compute

$$E_{\mathcal{F},P}(\mathbf{y}) = \mathop{\mathbb{E}}_{\mathbf{m}\sim P(\mathbf{M}|\mathbf{y})} [\mathcal{F}(\mathbf{ym})] .$$

Nevertheless, (mean) imputation and the expected prediction task are related, but only under restrictive assumptions.

Example 1. Let $\mathcal{F} : \mathbf{X} \to \mathbb{R}$ be a linear function. That is, $\mathcal{F}(\mathbf{x}) = \sum_{x\in\mathbf{x}} w_X x$ for some weights $\mathbf{w}$. Suppose P is a distribution over $\mathbf{X}$ that assumes independence between features: $P(\mathbf{X}) = \prod_{X\in\mathbf{X}} P(X)$. Then, using linearity of expectation, the following holds for any partial observation $\mathbf{y}$:

$$\begin{aligned} E_{\mathcal{F},P}(\mathbf{y}) &= \mathop{\mathbb{E}}_{\mathbf{m}\sim P(\mathbf{M}|\mathbf{y})} \left[\sum_{y\in\mathbf{y}} w_Y y + \sum_{m\in\mathbf{m}} w_M m\right] \\ &= \sum_{y\in\mathbf{y}} w_Y y + \sum_{M\in\mathbf{M}} w_M \mathop{\mathbb{E}}_{m\sim P(M)}[m]. \end{aligned}$$

Hence, substituting the missing features with their means effectively computes the expected predictions of linear models if the independence assumption holds. Furthermore, if $\mathcal{F}$ is the true conditional probability of the labels and features are generated by a fully-factorized $P(\mathbf{X})$, then classifying by comparing the expected prediction $E_{\mathcal{F},P}$ to 0.5 is Bayes optimal on the observed features.

Example 2. Consider a logistic regression model $\mathcal{G}(\mathbf{x}) = \text{sigmoid}(\mathcal{F}(\mathbf{x}))$ where $\mathcal{F}$ is a linear function. Now, mean imputation no longer computes the expected prediction, even when the independence assumption in the previous example holds. In particular, if $\mathbf{y}$ is a partial observation such that $\mathcal{G}(\mathbf{ym})$ is positive for all $\mathbf{m}$, then the mean-imputed prediction is an overapproximation of the expected prediction:

$$\begin{aligned} \mathcal{G}(\mathbf{y}\,\mathbb{E}[\mathbf{m}]) &= \text{sigmoid}\left(\mathbb{E}[\mathcal{F}(\mathbf{ym})]\right) \\ &> \mathbb{E}[\text{sigmoid}(\mathcal{F}(\mathbf{ym}))] = E_{\mathcal{G},P}(\mathbf{y}). \end{aligned}$$

This is due to Jensen's inequality and concavity of the sigmoid function in the positive portion; conversely, it is an under-approximation in the negative cases.

Example 1 showed how to efficiently take the expectation of a linear function w.r.t. a fully factorized distribution. Unfortunately, the expected prediction task is in general computationally hard, even on simple classifiers and distributions.

Proposition 1. *Taking expectation of a sufficiently complex but efficient classifier w.r.t. a uniform distribution is* #P*-hard.*

Suppose our classifier tests whether a logical constraint holds between the input features. Then asking whether there exists a positive example is NP-hard. The expected classification on a uniform distribution is solving an even harder task, of counting solutions to the constraint, which is #P-hard.

Obviously, computing expectations, even of trivial classifiers, is as hard as probabilistic reasoning in the feature distribution, which is #P-hard for graphical models [Roth, 1996].

Proposition 2. *The expectation of a classifier that returns the value of a single feature w.r.t. a distribution represented by a probabilistic graphical model is* #P*-hard.*

Previous propositions showed that the expected prediction task stays intractable, even when we allow either the distribution or the classifier to be trivial.

Our next theorem states that the task is hard even for a simple linear classifier and a tractable distribution.[1]

Theorem 1. *Computing the expectation of a logistic regression classifier over a naive Bayes distribution is* NP*-hard.*

That is, the expected prediction task is hard even though logistic regression classification and probabilistic reasoning on naive Bayes models can both be done in linear time.

In summary, while the expected prediction task appears natural for dealing with missing data, its vast intractability provides a serious challenge, in particular compared to efficient alternatives such as imputation. Next, we investigate specific ways of practically overcoming this challenge.

3 Joint Distributions as Classifiers

While the expected prediction task is designed to address missing features in arbitrary predictors, there exists a family of classifiers that inherently support missing features. Given a joint distribution $P(\mathbf{X}, C)$ over the features $\mathbf{X}$ and class label C, we can classify a partial observation $\mathbf{y}$ simply by computing the conditional probability $P(C \,|\, \mathbf{y})$. In some sense, a joint distribution embeds a classifier $P(C \,|\, \mathbf{Y})$ for each subset of observed features $\mathbf{Y}$. In fact, computing $P(C \,|\, \mathbf{y})$ is *equivalent to computing the expected prediction* where $\mathcal{F}(\mathbf{X}) = P(C \,|\, \mathbf{X})$ and $P(\mathbf{X})$ is given by the joint:

$$\begin{aligned} P(C \,|\, \mathbf{y}) &= \sum_{\mathbf{m}} P(C, \mathbf{m} \,|\, \mathbf{y}) = \sum_{\mathbf{m}} P(C \,|\, \mathbf{my}) P(\mathbf{m} \,|\, \mathbf{y}) \\ &= \mathop{\mathbb{E}}_{\mathbf{m}\sim P(\mathbf{M}|\mathbf{y})} [P(C \,|\, \mathbf{ym})] = E_{\mathcal{F},P}(\mathbf{y}). \end{aligned} \tag{1}$$

Nevertheless, the prevailing consensus is that in practice discriminatively training a classifier $P(C \,|\, \mathbf{X})$ should be preferred to generatively learning $P(\mathbf{X}, C)$, because it tends to achieve higher classification accuracy [Bouchard and Triggs, 2004].

There are many generative-discriminative pairs obtained from fitting the same family of probabilistic models to optimize either the joint or the conditional likelihood [Ng and Jordan, 2002]. A well-known example is naive Bayes and logistic regression. Naive Bayes models assume that features are all mutually independent given the class. Its joint distribution is $P(\mathbf{X}, C) = P(C) \prod_{X\in\mathbf{X}} P(X \,|\, C)$. Under such assumption, marginal inference is efficient, and so is taking expectations in case of missing features as in Equation 1.

Logistic regression is the discriminative counterpart to naive Bayes. It has parameters $\mathbf{w}$ and posits that[2]

$$P(c \mid \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\cdot\mathbf{x}}}.$$

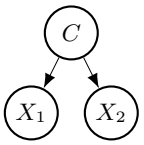
[1] All proofs can be found in Appendix A.

[2] Here, $\mathbf{x}$ also includes a dummy feature that is always 1 to correspond with the bias parameter w_0.

$$\mathbf{w} = \begin{bmatrix} -1.16 \\ 2.23 \\ -0.20 \end{bmatrix}$$

X_1	X_2	$\mathcal{F}(x_1, x_2)$
1	1	0.70
1	0	0.74
0	1	0.20
0	0	0.24

(a) Logistic regression weights and resulting predictions.



$P_1(c)$
0.5

C	$P_1(x_1 \mid C)$
1	0.8
0	0.3

C	$P_1(x_2 \mid C)$
1	0.45
0	0.5

$P_2(c)$
0.36

C	$P_2(x_1 \mid C)$
1	0.6
0	0.14

C	$P_2(x_2 \mid C)$
1	0.9
0	0.92

(b) Two naive Bayes distributions with same structure.

Figure 1: Logistic regression $\mathcal{F}(\mathbf{x}) = \text{sigmoid}(\mathbf{w}^T\mathbf{x})$ and two conformant naive Bayes models.

Any naive Bayes classifier can be translated to an equivalent logistic regression classifier on fully observed features.

Definition 2. We say $P(\mathbf{X}, C)$ *conforms* with $\mathcal{F} : \mathbf{X} \to [0, 1]$ if their classifications agree: $P(c \mid \mathbf{x}) = \mathcal{F}(\mathbf{x})$ for all $\mathbf{x}$.

Lemma 1. *Given a naive Bayes distribution P_{NB}, there is a unique logistic regression model P_{LR} that it conforms with. Additionally, such logistic regression model has the following weights:*

$$w_0 = \log \frac{P_{NB}(c)}{P_{NB}(\bar{c})} + \sum_{i=1}^{n} \log \frac{P_{NB}(\bar{x}_i \mid c)}{P_{NB}(\bar{x}_i \mid \bar{c})}$$

$$w_i = \log \frac{P_{NB}(x_i \mid c)}{P_{NB}(x_i \mid \bar{c})} \cdot \frac{P_{NB}(\bar{x}_i \mid \bar{c})}{P_{NB}(\bar{x}_i \mid c)}, \quad i = 1, \ldots, n$$

Here, x, $\bar{x}$ denote $X = 1$, $X = 0$ respectively.

Consider for example the naive Bayes distribution P_1 in Figure 1b. For all possible feature observations, the NB classification $P_1(c \mid \mathbf{x})$ is equal to that given by the logistic regression $\mathcal{F}$ in Figure 1a, whose weights are as given by above lemma (i.e., P_1 conforms with $\mathcal{F}$). Furthermore, distribution P_2 also translates into the same logistic regression. In fact, there can be infinitely many such naive Bayes distributions.

Lemma 2. *Given a logistic regression P_{LR} and $\theta \in (0, 1)^n$, there exists a unique naive Bayes model P_{NB} such that*

$$P_{NB}(c \mid \mathbf{x}) = P_{LR}(c \mid \mathbf{x}), \quad \forall\, c, \mathbf{x}$$
$$P_{NB}(x_i \mid c) = \theta_i, \quad i = 1, \ldots, n.$$

That is, given a logistic regression there are infinitely many naive Bayes models that conform with it. Moreover, after fixing values for n parameters of the NB model, there is a uniquely corresponding naive Bayes model.

We can expect this phenomenon to generalize to other generative-discriminative pairs; given a conditional distribution $P(C \mid \mathbf{X})$ there are many possible feature distributions $P(\mathbf{X})$ to define a joint distribution $P(\mathbf{X}, C)$. For instance, distributions P_1 and P_2 in Figure 1b assign different probabilities to feature observations; $P_1(\bar{x}_1, \bar{x}_2) = 0.23$ whereas $P_2(\bar{x}_1, \bar{x}_2) = 0.06$. Hence, we wish to define which one of these models is the "best". Naturally, we choose the one that best explains a given dataset of feature observations.[3]

Definition 3. Let $\mathcal{F} : \mathbf{X} \to [0, 1]$ be a discriminative classifier and D be a dataset where each example d is a joint assignment to $\mathbf{X}$. Given a family of distributions over C and $\mathbf{X}$, let $\mathcal{P}$ denote the subset of them that conforms with $\mathcal{F}$. Then *conformant learning* on D is to solve the following optimization:

$$\arg\max_{P \in \mathcal{P}} \prod_{d \in D} P(d) = \arg\max_{P \in \mathcal{P}} \prod_{d=(\mathbf{x}) \in D} \sum_{C} P(\mathbf{x}, C). \quad (2)$$

The learned model thus conforms with $\mathcal{F}$ and defines a feature distribution; therefore, we can take the expectation of $\mathcal{F}$ via probabilistic inference. In other words, it attains the high classification performance of the given discriminative model while also returning sophisticated predictions under missing features. Specifically, conformant naive Bayes models can be used to efficiently take expectations of logistic regression classifiers. Note that this does not contradict Theorem 1 which considers arbitrary pairs of LR and NB models.

4 Naive Conformant Learning

In this section, we study a special case of conformant learning–naive conformant learning, and show how it can be solved as a geometric program.

A naive Bayes distribution is defined by a parameter set θ that consists of $\theta_c, \theta_{\bar{c}}$, and $\theta_{x|c}, \theta_{x|\bar{c}}$ for all x. *Naive conformant learning* outputs the naive Bayes distribution P_θ that maximizes the (marginal) likelihood of given dataset and conforms with a given logistic regression model $\mathcal{F}$.

We will next show that above problem can be formulated as a *geometric program*, an optimization problem of the form:

$$\begin{aligned} &\min f_0(x) \\ &\text{s.t} \quad f_i(x) \leq 1, \quad i = 1 \ldots m \\ &\qquad g_i(x) = 1, \quad i = 1 \ldots p \end{aligned}$$

where each f_i is a posynomial and g_i monomial. A *monomial* is a function of the form $b x_1^{a_1} \cdots x_n^{a_n}$ defined over positive real variables $x_1, \ldots, x_n$ where $b > 0$ and $a_i \in \mathbb{R}$. A *posynomial* is a sum of monomials. Every geometric program can be transformed into an equivalent convex program through change of variables, and thus its global optimum can be found efficiently [Boyd *et al.*, 2007].

As we want to maximize the likelihood, we instead minimize its inverse. Let $n(\mathbf{x})$ denote how many times the joint assignment $\mathbf{x}$ appears in dataset D. Then we can express the objective function as

$$\prod_{d \in D} P_\theta(d)^{-1} = \prod_{\mathbf{x}} P_\theta(\mathbf{x})^{-n(\mathbf{x})} = \prod_{\mathbf{x}} \left(\sum_{c} \prod_{x \in \mathbf{x}} \theta_{x|c} \theta_c \right)^{-n(\mathbf{x})}.$$

Above formula, directly expanded, is not a posynomial. In order to express it as a posynomial we consider an auxiliary

[3] We assume we have i.i.d. sampled data. If a true distribution is known, we can equivalently minimize the KL-divergence to it.

dataset D' constructed from D as follows: for each data point $d_j = (\mathbf{x}) \in D$, there are $d'_{j,c} = (\mathbf{x}, c) \in D'$ with weight $\alpha_{j,c}$ for all values of c. If the weights are such that $\alpha_{j,c} \geq 0$ and $\sum_c \alpha_{j,c} = 1$ for all j, then the inverse of the expected joint likelihood given the new dataset D' is

$$\begin{aligned}
&\prod_{d'_{j,c}=(\mathbf{x},c)\in D'} P_\theta(\mathbf{x}, c)^{-\alpha_{j,c}} \\
&= \prod_{d_j=(\mathbf{x})\in D} \prod_c P_\theta(\mathbf{x})^{-\alpha_{j,c}} P_\theta(c \mid \mathbf{x})^{-\alpha_{j,c}} \\
&= \prod_{d_j=(\mathbf{x})\in D} P_\theta(\mathbf{x})^{-1} \cdot \prod_{d_j=(\mathbf{x})\in D, c} P_\theta(c \mid \mathbf{x})^{-\alpha_{j,c}}. \qquad (3)
\end{aligned}$$

For any $P_\theta \in \mathcal{P}$, the conditional distribution $P_\theta(C \mid \mathbf{X})$ is fixed by the logistic regression model; in other words, the last product term in Equation 3 is a constant. Therefore, maximizing the expected (joint) likelihood on a completed dataset must also maximize the marginal likelihood, which is our original objective. Intuitively, maximizing the joint likelihood on any dataset is equivalent to maximizing the marginal likelihood $P(\mathbf{X})$ if the conditional distribution $P(C \mid \mathbf{X})$ is fixed. Now our objective function can be written as a monomial in terms of the parameters:

$$\prod_{d_{j,c}\in D'} P_\theta(d_{j,c})^{-\alpha_{j,c}} = \prod_{d'_{j,c}=(\mathbf{x},c)\in D'} \left(\theta_c \prod_{x\in\mathbf{x}} \theta_{x|c}\right)^{-\alpha_{j,c}} \qquad (4)$$

Next, we express the set of conformant naive Bayes distributions $\mathcal{P}$ as monomial equality constraints in terms of θ. An NB model P_θ conforms with an LR $\mathcal{F}$ if and only if its corresponding logistic regression weights, according to Lemma 1, match those of $\mathcal{F}$. Hence, $P_\theta \in \mathcal{P}$ precisely when

$$e^{w_0}\ \theta_c^{-1}\ \theta_{\bar{c}} \prod_{i=1}^{n} \theta_{\bar{x}_i|c}^{-1}\ \theta_{\bar{x}_i|\bar{c}} = 1 \qquad (5)$$

$$e^{w_i}\ \theta_{x_i|c}^{-1}\ \theta_{x_i|\bar{c}}\ \theta_{\bar{x}_i|c}\ \theta_{\bar{x}_i|\bar{c}}^{-1} = 1,\ \forall\, i \qquad (6)$$

We also need to ensure that the parameters define a valid probability distribution (e.g., $\theta_c + \theta_{\bar{c}} = 1$). Because such posynomial equalities are not valid geometric program constraints, we instead relax these to posynomial inequalities:[4]

$$\theta_c + \theta_{\bar{c}} \leq 1,\ \theta_{x_i|c} + \theta_{\bar{x}_i|c} \leq 1,\ \theta_{x_i|\bar{c}} + \theta_{\bar{x}_i|\bar{c}} \leq 1, \quad \forall\, i \qquad (7)$$

Putting everything together, naive conformant learning can be solved as a geometric program whose objective function is given by Equation 4 and constraints by Equations 5 – 7.[5]

5 Empirical Evaluation

In this section, we empirically evaluate the performance of our proposed naive conformant learning (NaCL) and provide a detailed discussion of our method's advantages over existing imputation approaches in practice. More specifically, we want to answer the following questions:

[4]The learned parameters may not sum to 1. They can still be interpreted as a multi-valued NB with same likelihood that conforms with $\mathcal{F}$. These constraints were always active in our experiments.

[5]We have assumed binary class for conciseness, but the approach easily generalizes to multiclass. Details can be found in Appendix B.

Datasets	Size	# Classes: Dist.	# Features	Feature Types
MNIST	60K	10: Balanced	784	Integer: pixel value
Fashion	60K	10: Balanced	784	Integer: pixel value
CovType	581K	7: Unbalanced	54	Continuous & Categorical
Adult	49K	2: Unbalanced	14	Integer & Categorical
Splice	3K	3: Unbalanced	61	Categorical

Table 2: Summary of our testbed.

Q1 Does NaCL reliably estimate the probabilities of the original logistic regression with full data? How do these estimates compare to those from imputation techniques, including ones that also use the feature distribution?

Q2 Do higher-quality expectations of a logistic regression classifier result in higher accuracy on test data?

Q3 Does NaCL retain logistic regression's higher predictive accuracy over unconstrained naive Bayes?

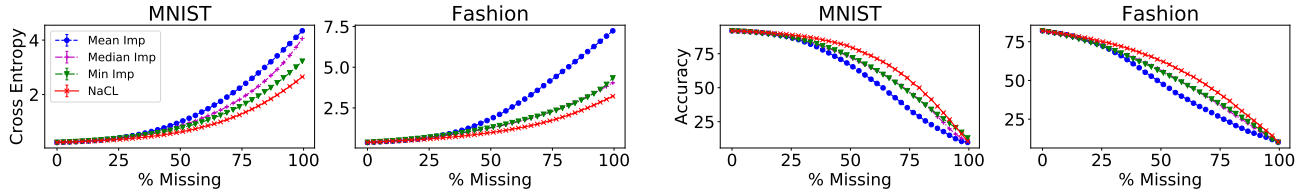
Experimental Setup To demonstrate the generality of our method, we construct a 5-dataset testbed suite that covers assorted configurations [Yann *et al.*, 2009; Xiao *et al.*, 2017; Blackard and Dean, 1999; Dua and Karra Taniskidou, 2017; Noordewier *et al.*, 1991]; see Table 2. The suite ranges from image classification to DNA sequence recognition; from fully balanced labels to $> 75\%$ of samples belonging to a single class; from continuous to categorical features with up to 40 different values. For datasets with no directly provided test set, we construct one by a $80:20$ split. As our method assumes binary inputs, we transform categorical features through one-hot encoding and binarize continuous ones based on whether they are larger than their respective mean plus 0.05 standard deviation.

5.1 Conformity with the Original Predictions

The optimal method to deal with missing values would be one that enables the original classifier to act as if no features were missing. In other words, we want the predictions to be affected as little as possible by the missingness. As such, we evaluate the similarity between predictions made with and without missingness, measured by the average cross entropy. The results are reported in Figure 3a[6] and Table 4. Our method outperforms all the baselines by a significant margin, demonstrating the superiority of the expected predictions produced by our method.

We also compare NaCL with two imputation methods that consider the feature distribution, namely EM [Dempster *et al.*, 1977] and MICE [Buuren and Groothuis-Oudshoorn, 2010]. EM imputation reports the second to worst average cross entropies and MICE's results are very similar to mean imputation's when 1% of features are missing. Due to the fact that both EM and MICE are excessively time-consuming to run and their imputed values are no better quality than more lightweight alternatives, we do not compare with them in the rest of the experiments. We would like to especially emphasize this comparison; it demonstrates that directly leveraging feature distributions without also considering how the

[6]The results associated with max imputation are dismissed as they are orders of magnitude worse than the rest.



(a) Cross Entropy (fidelity to logistic regression predictions)

(b) Accuracy (impact on test-set prediction quality)

Figure 3: Standard image classification datasets: comparison of average cross entropy to the original predictions and classification accuracies between naive conformant learning (NaCL) and commonly used imputation methods. The conditional probabilities from NaCL are consistently closest to the full-data predictions, and NaCL consistently outperforms other methods with different percentages missing features.

Cross Entropy	CovType				Adult				Splice			
Under % Missing	20%	40%	60%	80%	20%	40%	60%	80%	20%	40%	60%	80%
Min Imputation	12.8	15.5	20.7	29.4	41.0	49.2	55.4	59.6	71.3	81.8	97.2	117.2
Max Imputation	52.6	89.1	133.5	187.7	84.2	114.6	125.3	114.5	70.5	78.3	89.3	103.2
Mean Imputation	12.8	15.6	21.2	30.5	34.1	38.7	44.8	52.6	69.2	**74.7**	**82.4**	92.3
Median Imputation	12.8	15.7	21.4	30.8	35.3	41.2	48.6	57.8	70.0	75.7	83.0	**92.0**
Naive Conformant Learning	**12.6**	**14.8**	**18.9**	**25.8**	**33.6**	**37.0**	**41.2**	**46.6**	**69.1**	**74.7**	82.8	94.0

Table 4: Three unbalanced UCI datasets with categorical features: comparison of average cross entropy to the original predictions between naive conformant learning (NaCL) and commonly used imputation methods. The closest are denoted in bold.

Weighted F1	CovType				Adult				Splice			
under % Missing	20%	40%	60%	80%	20%	40%	60%	80%	20%	40%	60%	80%
Min Imputation	64.0	58.1	52.2	46.1	81.7	79.3	77.5	**76.0**	86.9	69.8	49.2	38.8
Max Imputation	49.8	44.4	41.6	37.3	81.7	79.3	77.4	**76.0**	86.9	69.8	49.1	38.8
Mean Imputation	64.0	58.0	52.2	46.3	82.9	79.8	75.3	70.7	91.8	82.3	66.2	45.7
Median Imputation	64.0	58.1	52.2	46.1	82.7	79.2	74.8	70.5	89.4	77.6	59.5	42.5
Naive Conformant Learning	**66.1**	**61.7**	**56.9**	**51.7**	**83.4**	**81.2**	**77.9**	73.5	**93.3**	**87.2**	**76.6**	**59.1**

Table 5: Three unbalanced UCI datasets with categorical features: comparison of weighted F1 scores between naive conformant learning (NaCL) and commonly used imputation. The highest are denoted in bold.

imputed values impact the classifier may lead to unsatisfactory predictions, further justifying the need for solving the expected prediction task and doing conformant learning. This also concludes our answer to Q1.

5.2 Classification Accuracy

Encouraged by the fact that NaCL produces more reliable estimates of the conditional probability of the original logistic regression, we further investigate how much it helps achieve better classification accuracy with different percentages of missing features (i.e., Q2). As suggested by Figure 3b and Table 5[7], NaCL consistently outperforms all other methods except on the Adult dataset with 80% of the features missing.

Lastly, we compare NaCL to naive Bayes to answer Q3.[8] In all datasets except Splice, with fully observed features logistic regression classifies better than naive Bayes. NaCL maintains this advantage until 40% of the features go missing, further demonstrating the effectiveness of our method. Note that these four datasets have a large number of samples, which is consistent with the prevailing consensus that discriminative learners are better classifiers given a sufficiently large number of samples [Ng and Jordan, 2002].

6 Case Study: Sufficient Explanations

In this section we briefly discuss utilizing conformant learning to explain classifications and show some empirical examples as a proof of concept.

On a high level, the task of explaining a particular classification can be thought of as quantifying the "importance" of each feature and choosing a small subset of the most important features as the explanation. Linear models are widely considered easy to interpret, and thus many explanation methods learn a linear model that is closely faithful to the original one, and then use the learned model to assign importance to features [Ribeiro *et al.*, 2016; Lundberg and Lee, 2017; Shrikumar *et al.*, 2017]. These methods often assume a black-box setting, and to generate explanations they internally evaluate the predictor on multiple perturbations of the given instance. A caveat is that the perturbed values may have a very low probability on the distribution the classifier was trained

[7] We report weighted F1 scores as the datasets are unbalanced.

[8] Since naive Bayes optimizes for a different loss and effectively solves a different task than NaCL and the reported imputation methods, we do not include its results in the table.

on. This can lead to unexpected results as machine learning models typically only guarantee generalization if both train and test data are drawn from the same distribution.

Instead we propose to leverage the feature distribution in producing explanations. To explain a given binary classifier, we consider a small subset of feature observations that is sufficient to get the same classification, in expectation w.r.t. a feature distribution. Next, we formally define our method:

Definition 4. (Support and Opposing Features) Given $\mathcal{F}$, P, and $\mathbf{x}$, we partition the given feature observations into two sets. The first set consists of the *support* features that contribute towards the classification of $\mathcal{F}(\mathbf{x})$:

$$\mathbf{x}_+ = \begin{cases} \{x \in \mathbf{x} : E_{\mathcal{F},P}(\mathbf{x} \setminus x) \leq \mathcal{F}(\mathbf{x})\} & \text{if } \mathcal{F}(\mathbf{x}) \geq 0.5, \\ \{x \in \mathbf{x} : E_{\mathcal{F},P}(\mathbf{x} \setminus x) > \mathcal{F}(\mathbf{x})\} & \text{otherwise.} \end{cases}$$

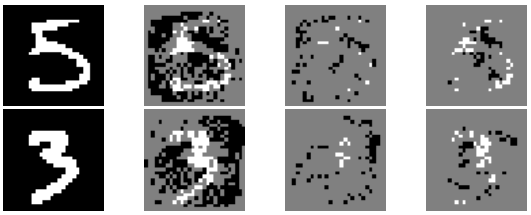
The rest are the *opposing* features that provide evidence against the current classification: $\mathbf{x}_- = \mathbf{x} \setminus \mathbf{x}_+$.

Definition 5. *Sufficient explanation* of $\mathcal{F}(\mathbf{x})$ with respect to P is defined as the following:

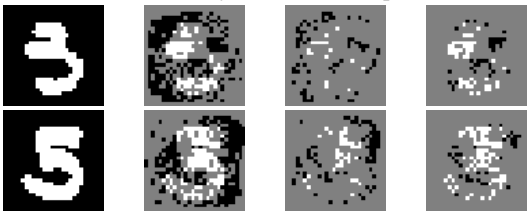
$$\arg\min_{\mathbf{e} \subseteq \mathbf{x}_+} |\mathbf{e}|$$
$$\text{s.t. } \operatorname{sgn}(E_{\mathcal{F},P}(\mathbf{e}\mathbf{x}_-) - 0.5) = \operatorname{sgn}(\mathcal{F}(\mathbf{x}) - 0.5)$$

Intuitively, this is the smallest set of support features that, in expectation, result in the same classification despite all the evidence to the contrary. In other words, we explain a classification using the strongest evidence towards it.

For a qualitative evaluation, we generate sufficient explanations on instances of a binary logistic regression task for MNIST digits 5 and 3; see the last column of Figure 6. Take the first example in Figure 6a: the white pixels selected as sufficient explanation show that the digit should be a 5. Also notice the black pixels in the explanation: they express how the absence of white pixels significantly contributes to the classification, especially in parts of the image where they would be expected for the opposing class. Similarly, the black pixels in the first example in Figure 6b look like a 3, and the white pixels in the explanation look like a 5, explaining why this 3 was misclassified as a 5. We further compare our approach to an alternative one that selects a subset of support features based on their logistic regression weights; see the third column of Figure 6. It selects features that will cause a large difference in prediction if the value was flipped, as opposed to missing, which is what sufficient explanation considers.



(a) Correctly classified examples



(b) Misclassified examples

Figure 6: Explanations for MNIST classifications. Grey features were not chosen as explanations; white/black are the true color of chosen features. From left to right: 1) all features; 2) support features; 3) top-k support features; 4) sufficient explanation of size k.

7 Related Work

There have been many approaches developed to classify with missing values, which can broadly be grouped into two different types. The first one focuses on increasing classifiers' inherent robustness to feature corruption, which includes missingness. A common way to achieve such robustness is to spread the importance weights more evenly among features [Globerson and Roweis, 2006; Dekel *et al.*, 2010; Xia *et al.*, 2017]. One downside of this approach is that the trained classifier may not achieve its best possible performance if no features go missing.

The second one investigates how to impute the missing values. In essence, imputation is a form of reasoning about missing values from observed ones [Sharpe and Solly, 1995; Batista *et al.*, 2002; McKnight *et al.*, 2007]. An iterative process is commonly used during this reasoning process [Buuren and Groothuis-Oudshoorn, 2010]. Some recent works also adapt auto-encoders and GANs for the task [Gondara and Wang, 2018; Zhang *et al.*, 2018]. Some of these works can be incorporated into a framework called multiple imputations to reflect and better bound one's uncertainty [Schafer, 1999; Azur *et al.*, 2011]. These existing methods focus on substituting missing values with those closer to the ground truth, but do not model how the imputed values interact with the trained classifier. On the other hand, our proposed method explicitly reasons about what the classifier is expected to return.

We are among the first to incorporate feature distribution to generate explanations. A notable recent work along this line is [Chen *et al.*, 2018], which proposes to maximize the mutual information between selected features and the class. To more explicitly leverage the feature distribution, one can marginalize over the universe of plausible alternative values for the select features by conditioning a generative model of the input distribution given the remaining features [Chang *et al.*, 2019].

8 Conclusion & Future Work

In this paper we introduced the expected prediction task, a principled approach to predicting with missing features. It leverages a feature distribution to reason about what a classifier is expected to return if it could observe all features. We then proposed conformant learning to learn joint distributions that conform with and can take expectations of discriminative classifiers. A special instance of it–naive conformant learning–was shown empirically to outperform many existing imputation methods. For future work, we would like to explore conformant learning for other generative-discriminative pairs of models, and extend NaCL to real-valued features.

Acknowledgements

This work is partially supported by NSF grants #IIS-1657613, #IIS-1633857, #CCF-1837129, DARPA XAI grant #N66001-17-2-4032, NEC Research and a gift from Intel.

References

[Azur *et al.*, 2011] Melissa J Azur, Elizabeth A Stuart, Constantine Frangakis, and Philip J Leaf. Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research*, 2011.

[Batista *et al.*, 2002] Gustavo EAPA Batista, Maria Carolina Monard, et al. A study of k-nearest neighbour as an imputation method. *HIS*, 2002.

[Blackard and Dean, 1999] Jock A Blackard and Denis J Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 1999.

[Bouchard and Triggs, 2004] Guillaume Bouchard and Bill Triggs. The tradeoff between generative and discriminative classifiers. In *COMPSTAT*, 2004.

[Boyd *et al.*, 2007] Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming. *Optimization and engineering*, 8(1):67, 2007.

[Buuren and Groothuis-Oudshoorn, 2010] S van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 2010.

[Chang *et al.*, 2019] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. Explaining image classifiers by counterfactual generation. 2019.

[Chen *et al.*, 2013] Suming Jeremiah Chen, Arthur Choi, and Adnan Darwiche. An exact algorithm for computing the same-decision probability. In *IJCAI*, 2013.

[Chen *et al.*, 2018] Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *ICML*, 2018.

[Dekel *et al.*, 2010] Ofer Dekel, Ohad Shamir, and Lin Xiao. Learning to classify with missing and corrupted features. *Machine learning*, 2010.

[Dempster *et al.*, 1977] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1977.

[Dua and Karra Taniskidou, 2017] Dheeru Dua and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[Globerson and Roweis, 2006] Amir Globerson and Sam Roweis. Nightmare at test time: Robust learning by feature deletion. In *ICML*, 2006.

[Gondara and Wang, 2018] Lovedeep Gondara and Ke Wang. Mida: Multiple imputation using denoising autoencoders. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2018.

[Graham, 2012] John W Graham. *Missing data: Analysis and design*. Springer Science & Business Media, 2012.

[Little and Rubin, 2014] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 333. John Wiley & Sons, 2014.

[Lundberg and Lee, 2017] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NeurIPS*. 2017.

[McKnight *et al.*, 2007] Patrick E McKnight, Katherine M McKnight, Souraya Sidani, and Aurelio Jose Figueredo. *Missing data: A gentle introduction*. Guilford Press, 2007.

[Ng and Jordan, 2002] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[Noordewier *et al.*, 1991] Michiel O Noordewier, Geoffrey G Towell, and Jude W Shavlik. Training knowledge-based neural networks to recognize genes in dna sequences. In *NeurIPS*, 1991.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016.

[Roth, 1996] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 1996.

[Schafer, 1999] Joseph L Schafer. Multiple imputation: a primer. *Statistical methods in medical research*, 1999.

[Sharpe and Solly, 1995] Peter K. Sharpe and RJ Solly. Dealing with missing values in neural network-based diagnostic systems. *Neural Computing & Applications*, 1995.

[Shrikumar *et al.*, 2017] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *ICML*, 2017.

[Xia *et al.*, 2017] Jing Xia, Shengyu Zhang, Guolong Cai, Li Li, Qing Pan, Jing Yan, and Gangmin Ning. Adjusted weight voting algorithm for random forests in handling missing values. *Pattern Recognition*, 2017.

[Xiao *et al.*, 2017] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[Yann *et al.*, 2009] LeCun Yann, Cortes Corinna, and Christopher JC Burges. The mnist database of handwritten digits, 2009.

[Zhang *et al.*, 2018] Hongbao Zhang, Pengtao Xie, and Eric Xing. Missing value imputation based on deep generative models. *arXiv preprint arXiv:1808.01684*, 2018.

A Proofs

A.1 Proof of Theorem 1

The proof is by reduction from computing the same-decision probability, whose decision problem **D-SDP** was shown to be NP-hard. [Chen *et al.*, 2013]

D-SDP: Given a naive Bayes distribution $P(.)$ over variables C and $\mathbf{X}$, a threshold T, and a probability p, is the following same-decision probability

$$\sum_{\mathbf{x}} \mathbb{I}(P(c \,|\, \mathbf{x}) > T) P(\mathbf{x}) \tag{1}$$

greater than p?

Here, $\mathbb{I}(.)$ denotes an indicator function which returns 1 if the enclosed expression is true, and 0 otherwise.

Using Lemma 1 we can efficiently translate a naive Bayes model to a logistic regression with a weight function $w(.)$ such that

$$P(c \,|\, \mathbf{x}) = \frac{1}{1 + e^{-w(\mathbf{x})}}.$$

Note that $P(c \,|\, \mathbf{x}) > T$ iff $w(\mathbf{x}) > -\log(\frac{1}{T} - 1)$. Then we construct another logistic regression whose weight function is:

$$w'(\mathbf{x}) = n \left(w(\mathbf{x}) + \log\left(\frac{1}{T} - 1\right)\right),$$

for some positive constant n. As w is a linear model, w' is also linear, and $w'(\mathbf{x}) > 0$ iff $P(c \,|\, \mathbf{x}) > T$. As n grows, $w'(.)$ approaches ∞ and $-\infty$ for positive and negative examples, respectively. Hence, this logistic regression model outputs 1 if $P(c \,|\, \mathbf{x}) > T$ and 0 otherwise, effectively being an indicator function. Therefore, the expectation of such classifier over $P(\mathbf{X})$ is equal to the same-decision probability of $\mathbf{X}$. □

A.2 Proof of Lemma 1

We want to prove there is a unique solution for

$$P_{\text{NB}}(c \mid \mathbf{x}) = P_{\text{LR}}(c \mid \mathbf{x}), \quad P_{\text{NB}}(\bar{c} \mid \mathbf{x}) = P_{\text{LR}}(\bar{c} \mid \mathbf{x}), \quad \forall\, \mathbf{x}.$$

Since we have binary classification, it only suffices to solve for the first equation.

$$\begin{aligned} P_{\text{NB}}(c \mid \mathbf{x}) &= \frac{P_{\text{NB}}(\mathbf{x} \mid c)\; P_{\text{NB}}(c)}{P_{\text{NB}}(\mathbf{x})} \\ &= \frac{P_{\text{NB}}(\mathbf{x} \mid c)\; P_{\text{NB}}(c)}{P_{\text{NB}}(\mathbf{x} \mid c)\; P_{\text{NB}}(c) + P_{\text{NB}}(\mathbf{x} \mid \bar{c})\; P_{\text{NB}}(\bar{c})} \\ &= \frac{1}{1 + \dfrac{P_{\text{NB}}(\mathbf{x} \mid \bar{c})\; P_{\text{NB}}(\bar{c})}{P_{\text{NB}}(\mathbf{x} \mid c)\; P_{\text{NB}}(c)}} \\ &= \frac{1}{1 + \exp\left[\ln \dfrac{P_{\text{NB}}(\mathbf{x} \mid \bar{c})\; P_{\text{NB}}(\bar{c})}{P_{\text{NB}}(\mathbf{x} \mid c)\; P_{\text{NB}}(c)}\right]} \\ &= \frac{1}{1 + \exp\left[-\ln \dfrac{P_{\text{NB}}(\mathbf{x} \mid c)\; P_{\text{NB}}(c)}{P_{\text{NB}}(\mathbf{x} \mid \bar{c})\; P_{\text{NB}}(\bar{c})}\right]} \end{aligned} \tag{2}$$

We want

$$P_{\text{NB}}(c \mid \mathbf{x}) = P_{\text{LR}}(c \mid \mathbf{x}) = \frac{1}{1 + \exp\left[-\sum_i w_i\, x_i\right]} \tag{3}$$

By combining Equations 2 and 3 we get:

$$-\log \frac{P_{\text{NB}}(\mathbf{x} \mid c)\; P_{\text{NB}}(c)}{P_{\text{NB}}(\mathbf{x} \mid \bar{c})\; P_{\text{NB}}(\bar{c})} = -\sum_i w_i\, x_i$$

Using naive Bayes assumption and some simplifications we have:

$$\sum_{i=0}^{n} w_i x_i = \log \frac{P_{\text{NB}}(c)}{P_{\text{NB}}(\bar{c})} + \sum_{i=1}^{n} \log \frac{P_{\text{NB}}(x_i \mid c)}{P_{\text{NB}}(x_i \mid \bar{c})} \tag{4}$$

Now we want the RHS of Equation 4 to be a linear function of x_i's, so we do the following substitutions ($i > 0$):

$$\begin{aligned} \log \frac{P_{\text{NB}}(x_i \mid c)}{P_{\text{NB}}(x_i \mid \bar{c})} =& x_i \log \frac{P_{\text{NB}}(x_i = 1 \mid c)}{P_{\text{NB}}(x_i = 1 \mid \bar{c})} \\ &+ (1 - x_i) \log \frac{P_{\text{NB}}(x_i = 0 \mid c)}{P_{\text{NB}}(x_i = 0 \mid \bar{c})} \end{aligned} \tag{5}$$

By combining Equations 4 and 5 we can get the final result of the lemma by simple algebraic manipulations. To solve for w_0 we plug-in $x_i = 0$ (for $i > 0$ because x_0 is a dummy feature for the bias term so its always 1). To compute w_i we take the coefficient of x_i in Equation 5.

The last remaining task is to prove given the above weights we have $P_{\text{NB}}(c \mid \mathbf{x}) = P_{\text{LR}}(c \mid \mathbf{x})$ for all possible instantiations of $\mathbf{x}$. It suffices to show Equation 4 always holds. Lets say k of the x_i's are set to 1 and others are set to 0. Without loss of generality assume they are the first k variables. We can show Equation 4 holds as follows:

$$\begin{aligned} \sum_{i=0}^{n} w_i x_i =& w_0 + \sum_{i=1}^{k} w_i x_i + \sum_{i=k+1}^{n} w_i x_i \\ =& \log \frac{P_{\text{NB}}(c)}{P_{\text{NB}}(\bar{c})} + \sum_{i=1}^{n} \log \frac{P_{\text{NB}}(x_i = 0 \mid c)}{P_{\text{NB}}(x_i = 0 \mid \bar{c})} \\ &+ \sum_{i=1}^{k} \log \frac{P_{\text{NB}}(x_i = 1 \mid c)}{P_{\text{NB}}(x_i = 1 \mid \bar{c})} \cdot \frac{P_{\text{NB}}(x_i = 0 \mid \bar{c})}{P_{\text{NB}}(x_i = 0 \mid c)} \\ &+ \sum_{i=k+1}^{n} w_i \cdot 0 \\ =& \log \frac{P_{\text{NB}}(c)}{P_{\text{NB}}(\bar{c})} + \sum_{i=1}^{n} \log \frac{P_{\text{NB}}(x_i \mid c)}{P_{\text{NB}}(x_i \mid \bar{c})} \end{aligned}$$

□

A.3 Proof of Lemma 2

Through the same algebraic manipulation as before, we get the same equations as in Lemma 1 with the only difference being that our unknown variables are now the parameters of the naive Bayes model rather than weights of the logistic regression model. Intuitively, because NB model has $2n + 1$ free parameters but the LR model only has $n + 1$ parameters,

we would have some degree of freedom. To get rid of the freedom we fix the values for $P_{\text{NB}}(x_i = 1 \mid c)$:

$$P_{\text{NB}}(x_i = 1 \mid c) = \theta_i \tag{6}$$

We can fix the parameters in other ways as long as we fix one parameter per feature. We show this one for notational simplicity.

Now there is a unique naive Bayes model that matches the logistic regression classifier and also agrees with Equation 6. The parameters for that naive Bayes model are as follows:

$$\begin{aligned}
P_{\text{NB}}(x_i \mid c) &= \theta_i \\
P_{\text{NB}}(\bar{x}_i \mid c) &= 1 - P_{\text{NB}}(x_i \mid c) \\
P_{\text{NB}}(x_i \mid \bar{c}) &= \frac{1}{1 + e^{w_i} \dfrac{1-\theta_i}{\theta_i}} \\
P_{\text{NB}}(\bar{x}_i \mid \bar{c}) &= 1 - P_{\text{NB}}(x_i \mid \bar{c})
\end{aligned}$$

After this we can uniquely determine $P_{\text{NB}}(c)$ based on previous parameters.

$$P_{\text{NB}}(c) = \text{sigmoid}\left(w_0 - \sum_{i=1}^{n} \log \frac{P_{\text{NB}}(\bar{x}_i \mid c)}{P_{\text{NB}}(\bar{x}_i \mid \bar{c})}\right)$$

□

B Beyond Binary Classification: Multi-Class

In the paper we gave the constraints and the objective for the case of binary classification, in this section we show that our method can be easily extended to multiclass classification. The flow of the methods is similar to the binary classification but with the following modifications:

Definition 6. (Multi-class Classifiers) Let's say we have a classifier with K classes, each denoted by c_k ($k \in [0, K-1]$). Conditional probability for logistic regression naive Bayes classifier are defined as:

$$P_{\text{LR}}(c_k \mid \mathbf{x}) = \frac{e^{W_k \cdot \mathbf{x}}}{\sum_j e^{W_j \cdot \mathbf{x}}}$$

$$P_{\text{NB}}(c_k \mid \mathbf{x}) = \frac{P_{\text{NB}}(c_k) \prod_{i=1}^{n} P_{\text{NB}}(x_i \mid c_k)}{\sum_j \left[P_{\text{NB}}(c_j) \prod_{i=1}^{n} P_{\text{NB}}(x_i \mid c_j)\right]}$$

Definition 7. (Multi-class Substitutions) We define the following notation to make the equations simpler to read.

$$\begin{aligned}
\theta_{c_k} &= P_{\text{NB}}(c_k) \\
\theta_{x_i|c_k} &= P_{\text{NB}}(x_i = 1 \mid c_k) \\
\theta_{\bar{x}_i|c_k} &= P_{\text{NB}}(x_i = 0 \mid c_k)
\end{aligned}$$

Now, we want to get the same classifiers out of P_{LR} and P_{NB}, one way to make solving the equations simpler is to set the ratio of the probabilities to be equal to each other rather than directly (as we did in binary case). Moreover, to get same classifiers it suffices to divide by the probability of only one class, so we want the following be true for all $k \in [1, K-1]$.

$$\frac{P_{\text{LR}}(c_k \mid \mathbf{x})}{P_{\text{LR}}(c_0 \mid \mathbf{x})} = \frac{P_{\text{NB}}(c_k \mid \mathbf{x})}{P_{\text{NB}}(c_0 \mid \mathbf{x})}$$

This leads to:

$$\frac{e^{-W_k \cdot \mathbf{x}}}{e^{-W_0 \cdot \mathbf{x}}} = \frac{P_{\text{NB}}(c_k) \prod_{i=1}^{n} P_{\text{NB}}(x_i \mid c_k)}{P_{\text{NB}}(c_0) \prod_{i=1}^{n} P_{\text{NB}}(x_i \mid c_0)}$$

Finally, after doing some algebra similar to binary case we get the following constraints:

$$\sum_k \theta_{c_k} = 1 \tag{7}$$

$$\theta_{x_i|c_k} + \theta_{\bar{x}_i|c_k} = 1, \; \forall\, i, k > 0 \tag{8}$$

$$e^{w_{k,i} - w_{0,i}} \;\; \theta^{-1}_{x_i|c_k} \;\; \theta_{\bar{x}_i|c_k} \;\; \theta_{x_i|c_0} \;\; \theta^{-1}_{\bar{x}_i|c_0} = 1, \; \forall\, i, k > 0$$

$$e^{w_{k,0} - w_{0,0}} \;\; \theta^{-1}_{c_k} \;\; \theta_{c_0} \prod_{i=1}^{n} \theta^{-1}_{\bar{x}_i|c_k} \;\; \theta_{\bar{x}_i|c_0} = 1, \; \forall\, k > 0$$

Similar to the binary case, we relax Equations 7 and 8 to inequalities, so that we can use geometric programming.