**Changelog:**

* Plot loss graph
* Reworded hypothesis
* Added Fβ-score as a specific metric of evaluation, how it will be determined, and the motivation behind using it
* Updated methods section:
  + Gave Gradient Estimator
  + Removed VAE from methods (only using BNN)
  + Added BNN prior
  + Rewrote likelihood function for binary input

**Motivation:**

Churn rate, or the annual percentage rate at which customers stop subscribing to a service, is an important piece of information for companies to gather year after year as a measure of company success, and as a predictor of future performance. Similarly, with the ability to understand which customers are likely to cancel a subscription, companies can leverage targeted incentives to convince these customers to stay. With churn analysis, an understanding of the likelihood of a prediction is critical: labeling a customer as likely to churn when they actually are not will result in wasted resources on incentives, while overlooking a customer who is likely to churn will result in lost business.

For evaluation of our models’ performances, we will focus mainly on maximizing the Fβ-score, which is a weighted average of the recall and the precision. The recall represents the proportion of customers who churn that our model is able to correctly identify. Maximizing recall minimizes the number of customers lost; as previously mentioned, customers that leave a service usually leave for good. One customer lost costs more than one distribution of an incentive to a customer who is not at risk of leaving. However, because recall can be maximized by predicting that every customer will churn, we need to balance recall with precision. So, β will be set to be greater than 1 (weighting recall more heavily than precision); the specific value of β is yet to be set, but it will be determined it after doing further research into the corresponding value of a lost customer versus the cost of providing incentives for customers believed to be leaving a service in the context of telecom companies.

Our project’s main objective will be to, given a dataset containing features of customers and labels of whether they have churned or not, predict as a classification task: if a current customer is at significant risk of changing providers or not. Due to the nature of our dataset, which does not give any temporal information about the period in which a given customer churned, we will not be able to use the dataset to estimate when in the future a given customer will churn. Towards this goal, we will train a Bayesian neural network classifier. Specifically, we will explore the effect on performance of using a model that models the aleatoric uncertainty across our dataset. We will compare the resulting predictions and confidence levels against models that assume a homoscedastic variance across the input feature space. For clarification, aleatoric uncertainty is uncertainty whose source cannot be explained by the data; we will model it by predicting the likelihood in addition to the classification of customers as “at risk for churning”/“not at risk” as a function of the input features. The motivation behind exploring this heteroscedastic variance in the input data’s feature space is the nature of real-world scenarios: the variance across any classification problem’s input feature space should not remain constant. For example, although we do not know what features are included in our dataset in order to protect the company’s proprietary information, noise could have been introduced if the dataset includes any surveys on customer satisfaction--in these cases, different customers may have different perceptions of the rating scale. Alternatively, there may be cases where information may be outdated.

Because of the uneven noise we expect across the input features in the dataset, we predict that if we include the modeling of heteroskedastic variance as a function of the input, the model will have a higher Fβ-score than a Bayesian neural net that assumes constant variance. We believe that it will also provide a more nuanced confidence of predictions that the model makes, which will be evaluated by using calibration plots. If our model accurately models change in variance across the input feature space, the hope is that the model would be able to ask for more information, or perhaps make more educational predictions about what the level of risk associated with allocating resources towards retaining each customer is and what should be allocated given that risk.

**Methods:**

Our probabilistic model will be a bayesian neural network using variational inference and an altered log likelihood in order to model heteroscedastic variance. The BNN will accomplish this by having two outputs: the predicted mean (x, W, b), and the predicted variance(x, W, b) (Kendall, 2017). The actual predicted y will be calculated by passing through a sigmoid which will result in a probability p̂. p̂ will be thresholded at p = 0.5 to produce a binary prediction of 1 (“churn”) or 0 (“no churn”).

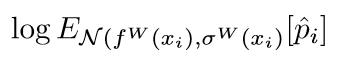
Both outputs of the BNN will be required to calculate the loss, which will be the objective function to take the gradient of to minimize in order to train the model through back propagation in pytorch. We will use the traditional ELBO function composed of the prior, likelihood, and estimated posterior as follows, and calculate its value through Monte Carlo sampling:



*Figure 1. Elbo function*

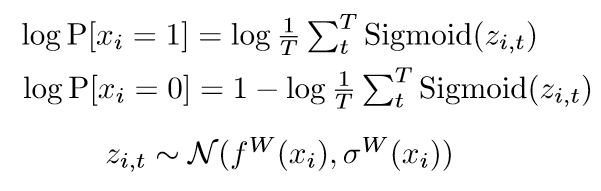
The prior of the weights and biases will be a standard distribution with mean = 0 and variance = 1. To incorporate a heteroskedastic variance, a new equation can be formulated that uses a variance dependent on the BNN weights *w*, and input *x*.

p̂i = SigmoidW(xi))



*Figure 2. The log likelihood for a given value x\_i*

The log likelihood for any input x\_i is given in figure 2. Unfortunately, this value is not analytically tractable. For this reason, we approximate it using monte carlo integration using T samples.



*Figure 3. Log likelihood for a given input x\_i and weights W by utilizes T samples*

To summarize, we can obtain the likelihood by using a sample of T values from the distribution specified by the outputs of our BNN. After sampling, apply the sigmoid to each sample and take the mean. This formulation of the likelihood allows the usage of the BNN output variance to aid in decreasing the loss function.

By building a heteroscedastic variance into the ELBO function, we hypothesize the BNN will optimize for a heteroscedastic variance that better captures the aleatoric uncertainty in the data than if a uniform variance was used. In order to test if the BNN does this successfully, our main baseline will be another BNN with identical architecture. The only difference between the two will be that the baseline BNN will instead a uniform variance when calculating the likelihood as opposed to the output of the BNN like previously suggested.

**Experiments:**

The dataset we are using was originally made public for a data science competition. As part of the competition, the performance of multiple baseline models on the dataset, including the company’s existing solution, was made public. Both BNN models will also be compared to these models in order to validate variational inference against other machine learning techniques.

We will use the [Orange Telecom dataset](https://www.kdd.org/kdd-cup/view/kdd-cup-2009/Data) from the [2009 Knowledge Discovery in Data Competition](http://www.mtome.com/Publications/CiML/CiML-v3-book.pdf). For this competition, a French telecom company, Orange, provided customer data with 230 features and a label for whether or not each customer will churn. The dataset has 50,000 entries with imbalanced classes. 7.5% of the instances belong to the churn class, while 92.5% will not churn. The company inflated the proportion of people who churn in the sample, so it is actually higher than under natural circumstances. In order to protect the privacy of its customers and its proprietary information, the company removed all variable names, encoded all categorical features, and multiplied all continuous features by a scalar value. Although we do not know what any of the features in our dataset represent, this dataset is ideal to explore the effect of modeling aleatoric uncertainty, as it is large enough that issues arising from epistemic uncertainty should be minimal.

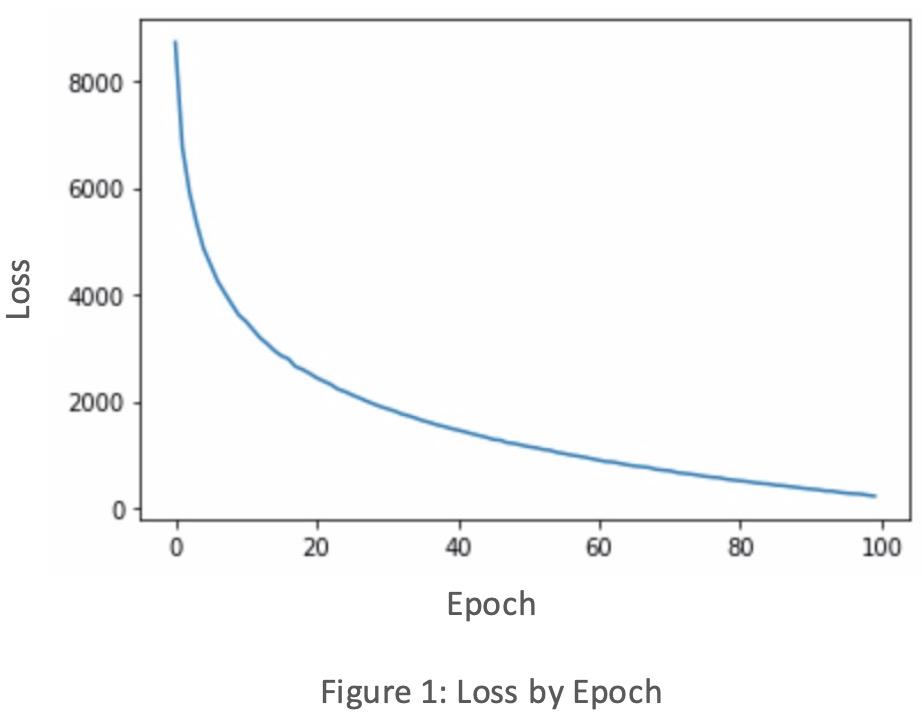
For many of our models’ hyperparameters, we will use the same tuning procedures across models. In both our focus and baseline models, we will gridsearch our learning rates optimizing for Fβ, and, for simplicity, we will fix the mean and variance of our prior at mean=0, variance=0.1 for the weights and biases of our model. For both models, we will test pytorch optimizers including SGD and Adam. We will vary the structure of our neural networks, specifically by adding a hidden layer after we split network to ensure each head has at least one hidden layer that is not shared. This seems like it will be one of the most crucial ways that we can alter the architecture of our MLP. We will leave the number of Monte Carlo samples fixed for both models for simplicity.

We will stratify our data when we split into training and test sets, and because our dataset is large, we will use a 90 : 10 train test split. We will run our variational inference until the difference in loss between epochs falls below 0.05%.

The baseline models mentioned in the report on the Knowledge Discovery in Data Competition are evaluated using AUC. We will use AUC as a secondary metric so that we can compare our results with results previously published on this dataset. We will create precision/recall (PRC) plots to show the precision and recall of our models because these curves are shown to be most effective on imbalanced datasets (Saito, 2015). As described in the motivation, so we will seek to maximize the Fβ-score, with a β value that balances the cost of a churn with the cost of providing an unnecessary incentive to continue services. This will allow us to compare the performance of our models, and understand whether accounting or heteroscedastic variance in our dataset results in more desirable predictions.

In order to get an approximate evaluation for how accurate the class probabilities generated from our model are, we will use calibration plots. To create these plots, we will group our predictions by ranges of probabilities assigned by our model, and then count the number of instances from each group for which our model made the correct classification.

We will be using pytorch to implement our program. The variational inference will be written by ourselves, the variational autoencoder will be borrowed from previous work.



**Timeline:**

~~10/21 Additional literature review, confirm dataset~~

~~10/28 Noise modeling research~~

~~11/04 Noise modeling implementation w/ simple model~~

11/18 DNN prototyping

11/25 DNN prototyping p2

12/02 Final Presentations

12/18 Paper

Looking Ahead / Potential Roadblocks:

We have several concerns that we want to address within the coming weeks. One concern is while our model currently works to reduce its loss, the accuracy does not necessarily decrease. Because of this, we believe there may be something wrong with our implementation, and more specifically our current method of evaluating the likelihood (we believe it to be a valid unbiased estimator of the likelihood, but are not certain). Another concern is that our implementation might be run too slowly to be effective on a larger model. Lastly, we will need to narrow down which advanced models we would like to train. These are all relevant to having a successful checkpoint three.

Kendall, A., & Gal, Y. (2017). *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?* 11.

Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, *10*(3), e0118432.<https://doi.org/10.1371/journal.pone.0118432>