Abstract:

Baysian neural networks (BNNs) are powerful tools that augment traditional neural networks' powerful inference abilities with the power to describe uncertainties. BNNs have led to advances in various areas including computer vision and health.. In most BNNs, only epistemic uncertainty, uncertainty from the model parameters, is accounted for, but recent methods have shown that aleatoric uncertainty, uncertainty deriving from the data, can also be modeled and lead to model improvements. In this paper, we argue for modeling aleatoric uncertainty in the churn prediction problem. I.e determining whether subscribers of a service will quit in the near future. We demonstrate results from modeling aleatoric uncertainty with neural networks on churn data for a telecom company.

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. 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, understanding the likelihood that a prediction will be correct 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 F_β-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; 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 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. We will show that any significant differences in the F_β-score between our baseline model and our experimental model are due to the modeling of heteroscedasticity by tuning the hyperparameters of each model to find the combination that results in the best performance, before comparing the baseline to the experimental, averaging the results across 10 training sessions each.

Our project's main objective will be to predict as a classification task if a current customer is at significant risk of changing providers or not. Towards this goal, we will train a Bayesian neural network classifier. Specifically, we will explore the effect on performance of modeling 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. Specifically, we expect to observe more accurate probabilities associated with our predictions when our model incorporates heteroscedastic uncertainty (Wang et. al 2012). We will measure this with calibration plots, along with a corresponding histogram that will show the density of predictions within each range of probability values. The calibration plots and histogram can be evaluated visually: the closer the calibration plot values follow a 'y=x' line, the better the model is shown to be accurate in its understanding of its own predicted probabilities; the histogram will show how often the model predicts with each level of confidence, or show how often the model is extremely confident vs. otherwise. We expect our model which incorporates heteroscedastic uncertainty to yield better F_β scores due to its ability to more fully represent the relationship in our data.

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.

Because of the uneven noise we expect across the input features in the dataset, we predict that if we include the modeling of heteroscedastic variance as a function of the input, the model will have a higher F_β-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 educated 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:

We build a BNN which can model heteroscedastic variance in the dataset using variational inference to learn the probability distribution over our neural network's weights. The BNN will model heteroscedastic variance by having two outputs for every input: the predicted mean $\hat{f}$(X, W, b), and the predicted variance $\hat{\sigma}$(X, W, b) (Kendall, 2017). This predicted mean and predicted variance will be used in our model's likelihood function. We estimate the likelihood of a particular class given feature values (X), and the weights (W) and biases (b) of our neural network as follows. First we get a real valued s:

$$s \sim N(\hat{f}(X, W, b), \hat{\sigma}(X, W, b))$$

Then we get a $\hat{p}$ where $\hat{p}$ is a probability between 0 and 1, by:

$$\hat{p} = \text{Sigmoid}(s)$$

$\hat{p}$ represents the probability, based on one sample from the above distribution, that the input features, X, belong to a customer who churned. To get an accurate estimate, we will use Monte Carlo sampling, with T samples. Thus, we will estimate our likelihood as:

$$p(y|X, W, b) = \begin{cases} \frac{1}{T}\Sigma_{l=0}^{T}\hat{p}(s_l) & y = 1 \\ 1 - \frac{1}{T}\Sigma_{l=0}^{T}\hat{p}(s_l) & y = 0 \end{cases}$$

We will pick a threshold, *t**, such that we predict the following where 1 corresponds to churn and 0 corresponds to no churn:

$$\hat{y} = \begin{cases} 1 & \hat{p} > t^* \\ 0 & \hat{p} \leq t^* \end{cases}$$

We will choose *t** by examining precision and recall at different *t** values; similarly to how we will be comparing results from and selecting other hyperparameter values, we will choose the value of t* that maximizes the F_β-score.

We define the prior distribution over the weights and biases of our neural network as follows:

$$p(W, b) \sim N(0, 1)$$

We estimate our posterior distribution with variational inference. We define our variational distribution $q \sim N(m, s)$ where *m* is a vector the same size as the number of trainable parameters of our network representing the means of the trainable parameters, and *s* diagonal matrix, whose diagonal is the same size and represents standard deviations of our trainable parameters (Kingma & Welling, 2014). We will use as our loss function the negative of the ELBO function, which is a lower bound on the evidence (log p(y|x)). Maximizing the ELBO minimizes the KL divergence between our variational distribution, *q*, and the actual posterior distribution of our model, which is a measure of dissimilarity between the estimated posterior and the true posterior. As discussed in (Roeder, 2017) this minimizes the dissimilarity between our model's estimated posterior and the true posterior. We will use gradient descent on our loss function, defined below, to find the *m*, s** which minimize loss. We use the reparameterization trick in our estimation of the ELBO function, which, as discussed by (Xu et. al, 2019), gives a lower marginal variance estimate of the gradient of the ELBO than the score function estimator.

$$\mathcal{L}(m, s) \quad = \quad -\mathbb{E}_{q(w,b|m,s)}[\log p(y|x, w, b) \; + \; \log p(w, b) \; - \; \log q(w, b|m, s)]$$

By using a likelihood which incorporates heteroscedastic variance into our ELBO function, we hypothesize the BNN will better capture aleatoric uncertainty in the data than if our

likelihood assumed constant variance; thus, we believe this model will lead to improvements in accuracy. In order to test this hypothesis, we will compare our results against a baseline BNN with identical architecture, but which assumes a constant variance in its likelihood.

For our baseline, our neural network will only predict one value, the predicted mean, $\hat{f}$(X, W, b). All calculations will be the same, except, our likelihood will assume a constant variance, so we calculate *s* as follows:

$$s \sim \mathcal{N}(\hat{f}(X, W, b), \sigma^2)$$

We planned to do a logspace search of 8 values for σ: 0.01, 0.05, 0.1, 0.5, 1, 5, 10, and 50; however in the end we were only able to test for σ with a constant value of 0.1.

Experimental Setup:

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. We compare both BNN models to these models in order to validate Bayesian neural networks against other machine learning techniques.

We use the Orange Telecom dataset from the 2009 Knowledge Discovery in Data Competition. 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% do 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 use the same tuning procedures across models. In both our focus and baseline models, we gridsearch our learning rates (testing 1e-3, 5e-3, 7e-3, 9e-3) optimizing for F_β, and, for simplicity, we will fix the mean and variance of our prior at mean=0, variance=1 for the weights and biases of our model. For both models, we tried optimization with SGD and Adam. We leave the number of Monte Carlo samples fixed at 100 for both models for simplicity. We leave architecture fixed for both focus and baseline models. Our focus model shares its first 2 hidden layers between both output heads, and each output head has 3 hidden layers. To ensure the two networks are of comparable complexity, our baseline model will be a fully connected BNN with 5 hidden layers. For simplicity, we keep hidden sizes

uniform across the networks for both baseline and focus models, and we gridsearch hidden sizes of 128 and 512.

We stratify our data when we split into training and test sets, and we use a 80 : 20 train test split. To help our model learn instances of churn despite the dataset's class imbalance, we duplicate instances of churn until the dataset is composed of 40% churn instances and 60% no churn instances. We run variational inference for 200 epochs or until loss increases for 20 consecutive epochs.

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 F_5-score. 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.

We evaluate the accuracy of our model's predicted class probabilities with calibration plots. To create these plots, we group our model's predictions by ranges of 10% 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 use be using pytorch to implement our program. We implement variational inference ourselves.

Experiments

To validate our approach. We tested a simplified version of our model on a toy dataset. This dataset was comprised of X values between -10 and 10 with Y = X - 5 + noise. The noise was equal to samples from a normal distribution ~N(0, x + 10.0000001). The hyperparameters used to train the model are given below.

Hyperparameters	
Batchsize	100
Number of Epochs	100
Learning Rate	1e-3
NN Hidden Layers	0

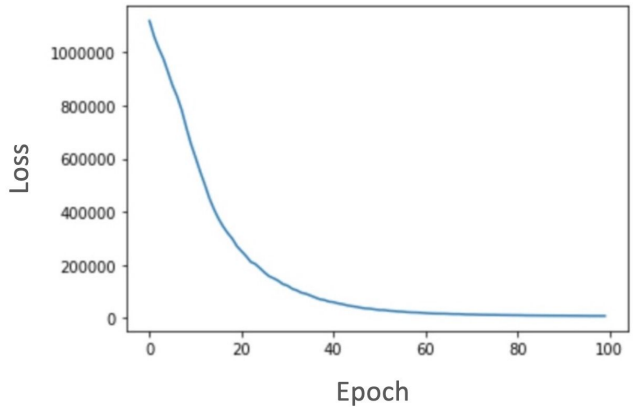

Figure 1: Loss for a linear model on our toy dataset. Hyperparameters used are batchsize of 100, 100 epochs, and learning rate of 1e-3

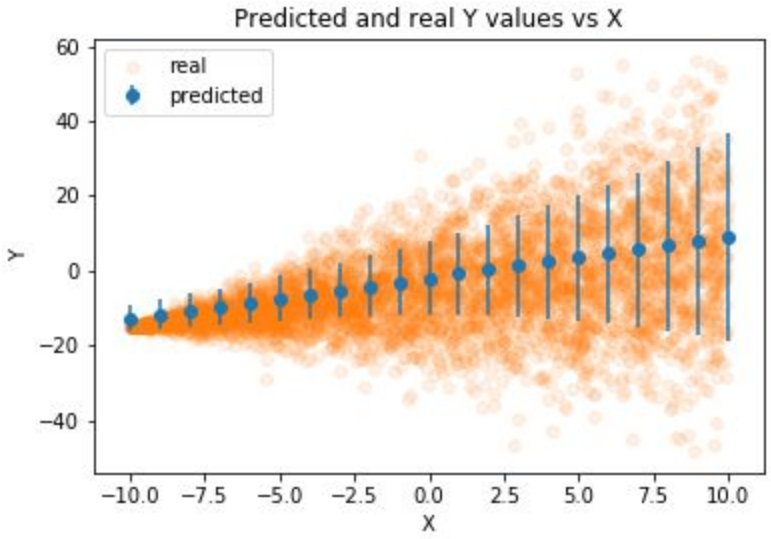

Figure 2: Predicted means and standard deviations plotted against the real Y values in a toy dataset.

After training the model, we obtained predicted means and standard deviations for various X values. In the above figure we can observe these predicted means and standard deviations against the actual Y values. As X increases in value, the standard deviation increases.

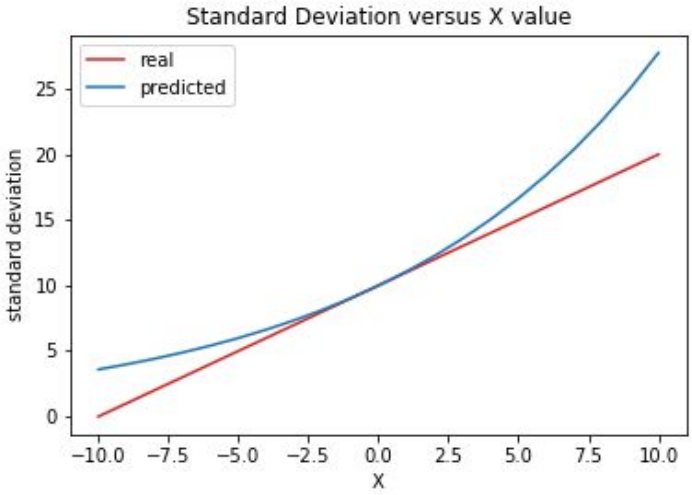

Figure 3: Predicted standard deviations over X values against the standard deviation used to generate the dataset

In figure 3 we explicitly graph the standard deviation predicted from the network at various X values and compare it against the real standard deviation used to generate the data the model trained against. In this graph we can see our model outputs an exponential standard deviation. We believe this occurs because the simplified model is linear and outputs the log standard deviation. Therefore, the predicted log standard deviation is linear and the predicted standard deviation is exponential. After incorporating more layers into our network the output standard deviation will not be linear.

On the same toy dataset parameters, we trained a baseline model that does not model heteroscedasticity. The hyperparameters were kept the same as the set used for the experimental model. The graphed results are as follows:

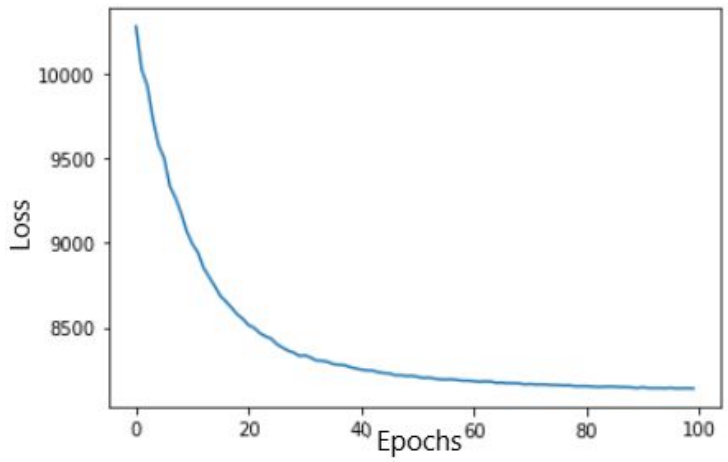

Figure 4: Loss for a homoscedastic linear model on our toy dataset. Hyperparameters used are batchsize of 100, 100 epochs, and learning rate of 1e-3

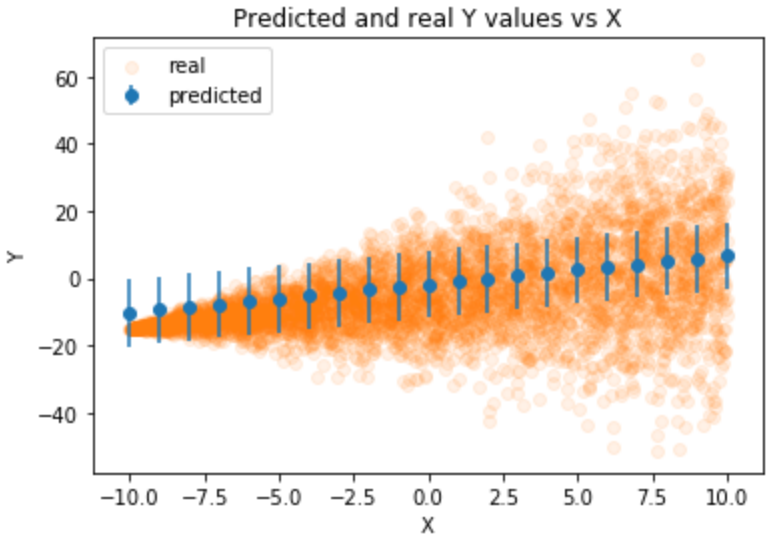

Figure 5: Performance of model with constant variance.

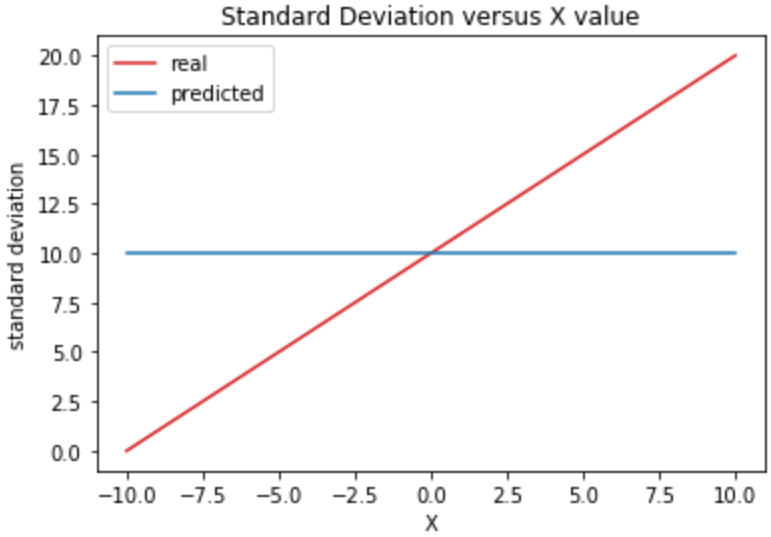

Figure 6: Constant variance against the variance used to generate the dataset

As can be seen by the graphs, the baseline model and the experimental model predicted relatively similar mean values; however, it's clear that the experimental model's learned variance values describe the data more completely. In terms of basic metrics, the baseline model's final batch resulted in a loss value of 8139.12 and a MAE value of 8.81, compared to the experimental

model's slightly higher loss value of 8230.07 and a slightly lower MAE value of 8.60. We have yet to confirm what the best value of β to use for the F_β, but once that is confirmed we will be using that primarily as the metric for evaluating and comparing the two models.

After performing our grid search, we found three layers of hidden size 128 and a learning rate of 9e-3 to be optimal for the focus and baseline methods. From here, we evaluated the models and generated calibration plots and precision-recall curves

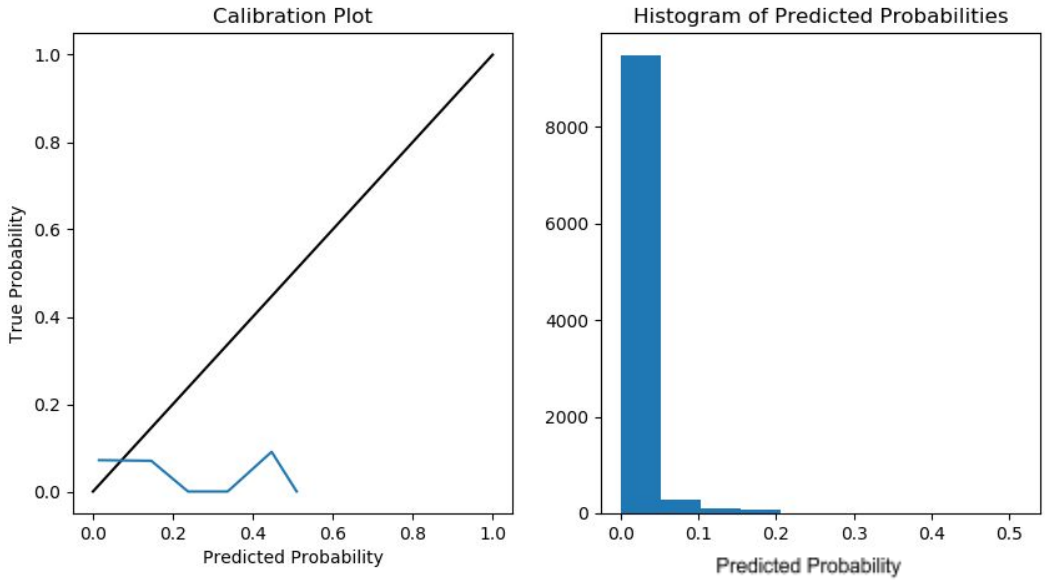

Figure 7: Calibration plot and histogram of predicted probabilities for focus model trained for 30 epochs with hidden size 128 and learning rate 9e-3

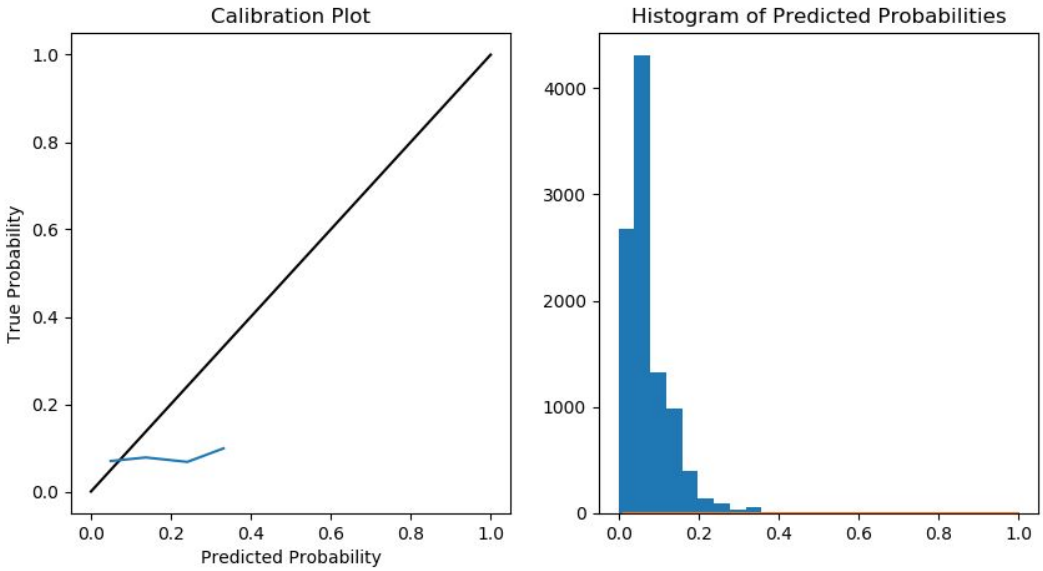

Figure 8: Calibration plot and histogram of predicted probabilities for baseline model trained for 30 epochs with hidden size 128 and learning rate 9e-3 and homoscedastic variance of 0.1

Unfortunately, we were not able to generate significantly different results between our focus method and the baseline method. One concern that came up is that the differences to model aleatoric uncertainty in binary classification are relatively small in comparison to multi-class classification or even regression, which we did when testing our model against toy data. Other things that may have caused issues were insufficient training epochs. In the future, it would be good to explore both of these issues further.

Blundell, C., Cornebise, J., Kavukcuoglu, K. & Wierstra, D.. (2015). Weight Uncertainty in Neural Network. *Proceedings of the 32nd International Conference on Machine Learning, in PMLR* 37:1613-1622

Kendall, A., & Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *Advances in Neural Information Processing Systems 30* (pp. 5574–5584).

Kingma, Diederik P, and Max Welling. (2014) "Proceedings of the 2nd International Conference on Learning Representations." ArXiv, *Auto-Encoding Variational Bayes*

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

Vafeiadis, T., Diamantaras, K. I., Sarigiannidis, G., & Chatzisavvas, K. Ch. (2015). A comparison of machine learning techniques for customer churn prediction. *Simulation Modelling Practice and Theory*, *55*, 1–9. https://doi.org/10.1016/j.simpat.2015.03.003

Wang, C., & Neal, R. M. (2012). *Gaussian Process Regression with Heteroscedastic or Non-Gaussian Residuals*. Technical report, University of Toronto. http://arxiv.org/abs/1212.6246

Zhang, R., Li, W., Tan, W., & Mo, T. (2017) "Deep and Shallow Model for Insurance Churn Prediction Service," *2017 IEEE International Conference on Services Computing (SCC)*, Honolulu, HI, pp. 346-353. https://doi.org/10.1109/SCC.2017.51

Roeder, G., Wu, Y., & Duvenaud, D. K. (2017). Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In Advances in Neural Information Processing Systems (pp. 6925-6934). http://papers.nips.cc/paper/7268-sticking-the-landing-simple-lower-variance-gradient-estimators-for-variational-inference.pdf

Xu, M., Quiroz, M., Kohn, R., & Sisson, S. A. (2019, April). Variance reduction properties of the reparameterization trick. In The 22nd International Conference on Artificial Intelligence and Statistics (pp. 2711-2720).