Bayesian Estimate of Minimum Loss

Jellyfish

JFR - Research Day

Mobolaji Williams, August 31, 2023

Summary

Setup of Problem: Say that we have a data set consisting of N points in feature space denoted $\mathbf{x}_i \in \mathbb{R}^M$ and their corresponding one-hot-encoded classes $y_{i,\alpha} \in \{0, 1\}$ where $\alpha = 1, \ldots, C$ for C classes.

The principal task of machine learning is to generate a function $\hat{p}(\alpha|\mathbf{x})$ (i.e., the probability that a data point with feature representation $\mathbf{x}$ has class α) that maximizes the probability that we get the labels from the features i.e., maximizes

$$\text{Probability of getting labels } \{y_{i,\alpha}\} \text{ given features } \{\mathbf{x}_i\} = \prod_{i=1}^{N} \prod_{\alpha=1}^{C} \hat{p}(\alpha|\mathbf{x}_i)^{y_{i,\alpha}} \tag{1}$$

Typically, in ML language we frame this "probability maximization" as a "loss minimization" with the loss function defined as follows minimize the function

$$\mathcal{L} = -\frac{1}{N} \sum_{\alpha=1}^{C} \sum_{i=1}^{N} y_{i,\alpha} \ln \hat{p}(\alpha|\mathbf{x}_i). \qquad \text{[Categorical Cross Entropy]} \tag{2}$$

The negative of the log of Eq.(1) is proportional to Eq.(3), and thus "minimizing the categorical cross entropy loss" is equivalent to "maximizing the probability that we get the labels from the features."

Setup of Problem: Say that we have a data set consisting of N points in feature space denoted $\mathbf{x}_i \in \mathbb{R}^M$ and their corresponding one-hot-encoded classes $y_{i,\alpha} \in \{0, 1\}$ where $\alpha = 1, \ldots, C$ for C classes.

The principal task of machine learning: Find $\hat{p}(\alpha|\mathbf{x})$ (i.e., the probability that a data point with feature representation $\mathbf{x}$ has class α) that minimizes

$$\mathcal{L} = -\frac{1}{N} \sum_{\alpha=1}^{C} \sum_{i=1}^{N} y_{i,\alpha} \ln \hat{p}(\alpha|\mathbf{x}_i). \qquad \text{[Categorical Cross Entropy]} \tag{3}$$

Question

Can we use a Bayesian argument to estimate what this loss should be? Can we use this Bayesian estimate to determine whether it is possible to "learn" (in an ML sense) from a data set?

We want to estimate the loss we expect to obtain from training a model on a data set $\{\{\mathbf{x}_i, y_{i,\alpha}\}; i = 1, \ldots, N; \alpha = 1, \ldots, C\}$. First we need to estimate the class probability given our feature.

- We denote as $\hat{\rho}(\mathbf{x}|\alpha)$ the estimate of the *probability density* in feature space for class α.
- The quantity $\hat{p}(\alpha|\mathbf{x})$ is the estimate of the probability (i.e., *not* density) of being in class α given feature point $\mathbf{x}$. This latter quantity is what ML problems aim to find and what we will use Bayes rule to compute.
- By Bayes rule, we have

$$\hat{p}(\alpha|\mathbf{x}) = \frac{\hat{\rho}(\mathbf{x}|\alpha)\hat{p}(\alpha)}{\hat{\rho}(\mathbf{x})} = \frac{\hat{\rho}(\mathbf{x}|\alpha)\hat{p}(\alpha)}{\sum_{\alpha'=1}^{C} \hat{\rho}(\mathbf{x}|\alpha')\hat{p}(\alpha')}. \tag{4}$$

We want to estimate the loss we expect to obtain from training a model on a data set $\{\{\mathbf{x}_i, y_{i,\alpha}\}; i = 1, \ldots, N; \alpha = 1, \ldots, C\}$. First we need to estimate the class probability given our feature.

- By Bayes rule, we have

$$\hat{p}(\alpha|\mathbf{x}) = \frac{\hat{\rho}(\mathbf{x}|\alpha)\hat{p}(\alpha)}{\hat{\rho}(\mathbf{x})} = \frac{\hat{\rho}(\mathbf{x}|\alpha)\hat{p}(\alpha)}{\sum_{\alpha'=1}^{C} \hat{\rho}(\mathbf{x}|\alpha')\hat{p}(\alpha')}. \tag{5}$$

- The quantity $\hat{p}(\alpha)$ is the data-based estimate for being in class α:

$$\hat{p}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} y_{i,\alpha}. \tag{6}$$

- The quantity $\hat{\rho}(\mathbf{x}|\alpha)$ is the data-based estimate for the probability density at point $\mathbf{x}$ given that we are in class α. Using the explicit definition of the probability density of samples, we have

$$\hat{\rho}(\mathbf{x}|\alpha) = \frac{1}{\sum_{i=1}^{N} y_{i,\alpha}} \sum_{j=1}^{N} \delta(\mathbf{x} - \mathbf{x}_j)\, \delta_{1,y_{j,\alpha}} \tag{7}$$

We want to estimate the loss we expect to obtain from training a model on a data set $\{\{\mathbf{x}_i, y_{i,\alpha}\}; i = 1, \ldots, N; \alpha = 1, \ldots, C\}$. First we need to estimate the class probability given our feature.

- By Bayes rule, we have

$$\hat{p}(\alpha|\mathbf{x}) = \frac{\hat{\rho}(\mathbf{x}|\alpha)\hat{p}(\alpha)}{\hat{\rho}(\mathbf{x})} = \frac{\hat{\rho}(\mathbf{x}|\alpha)\hat{p}(\alpha)}{\sum_{\alpha'=1}^{C} \hat{\rho}(\mathbf{x}|\alpha')\hat{p}(\alpha')}. \tag{8}$$

- Probability definitions

$$\hat{p}(\alpha) = \frac{1}{N}\sum_{i=1}^{N} y_{i,\alpha}, \quad \hat{\rho}(\mathbf{x}|\alpha) = \frac{1}{\sum_{i=1}^{N} y_{i,\alpha}} \sum_{j=1}^{N} \delta(\mathbf{x} - \mathbf{x}_j)\, \delta_{1,y_{j,\alpha}} \tag{9}$$

- Or defining $\mathcal{S}_\alpha$ as the set of data points i in class α

$$\hat{\rho}(\mathbf{x}|\alpha) = \frac{1}{|\mathcal{S}_\alpha|} \sum_{j \in \mathcal{S}_\alpha} \delta(\mathbf{x} - \mathbf{x}_j), \tag{10}$$

where $\delta(X)$ is the Dirac delta function and $\delta_{i,j}$ is the Kronecker delta function.

We want to estimate the loss we expect to obtain from training a model on a data set $\{\{\mathbf{x}_i, y_{i,\alpha}\}; i = 1, \ldots, N; \alpha = 1, \ldots, C\}$. First we need to estimate the class probability given our feature.

- Probability of feature given class

$$\hat{\rho}(\mathbf{x}|\alpha) = \frac{1}{|\mathcal{S}_\alpha|} \sum_{j \in \mathcal{S}_\alpha} \delta(\mathbf{x} - \mathbf{x}_j), \tag{11}$$

  where $\delta(X)$ is the Dirac delta function and $\delta_{i,j}$ is the Kronecker delta function.

- In practical circumstances to compute Eq.(11) we use what is known as *kernel density estimation*. This involves replacing the Dirac delta function with another function $K(x)$ with a given width h. We then have

$$\hat{\rho}(\mathbf{x}|\alpha) = \frac{1}{|\mathcal{S}_\alpha|} \sum_{j \in \mathcal{S}_\alpha} \delta(\mathbf{x} - \mathbf{x}_j), \quad \rightarrow \quad \frac{1}{|\mathcal{S}_\alpha| h^M} \sum_{j \in \mathcal{S}_\alpha} K\left(\frac{\mathbf{x} - \mathbf{x}_j}{h}\right) \equiv \hat{\rho}_{\mathsf{KDE}}(\mathbf{x}|\alpha). \tag{12}$$

We want to estimate the loss we expect to obtain from training a model on a data set $\{\{\mathbf{x}_i, y_{i,\alpha}\}; i = 1, \ldots, N; \alpha = 1, \ldots, C\}$.

- In effect, we will replace the theoretical distribution $\hat{\rho}(\mathbf{x}|\alpha)$ with KDE distribution $\hat{\rho}_{\mathsf{KDE}}(\mathbf{x}|\alpha)$ (thus in effect approximating the estimate of a theoretical quantity) and use the kernel density estimate in our Bayes formulas.
- In particular, the previous expression for $\hat{p}(\alpha|\mathbf{x})$ becomes

$$\hat{p}(\alpha|\mathbf{x}) \rightarrow \frac{\hat{\rho}_{\mathsf{KDE}}(\mathbf{x}|\alpha)\hat{p}(\alpha)}{\sum_{\alpha'=1}^{C} \hat{\rho}_{\mathsf{KDE}}(\mathbf{x}|\alpha')\hat{p}(\alpha')}, \tag{13}$$

  where the "$\rightarrow$" is meant to signify that the right-hand-side is a numerical approximation of the left-hand-side.

We want to estimate the loss we expect to obtain from training a model on a data set $\{\{\mathbf{x}_i, y_{i,\alpha}\}; i = 1, \ldots, N; \alpha = 1, \ldots, C\}$.

- Inserting this KDE probability into the original loss function, we ultimately find

$$\begin{aligned}\mathcal{L} &= -\frac{1}{N}\sum_{\alpha=1}^{C}\sum_{i=1}^{N} y_{i,\alpha}\ln\hat{p}(\alpha|\mathbf{x}_i) \\ &= -\sum_{\alpha=1}^{C}\hat{p}(\alpha)\ln\hat{p}(\alpha) - \frac{1}{N}\sum_{\alpha=1}^{C}\sum_{i=1}^{N} y_{i,\alpha}\ln\left[\frac{\hat{\rho}_{\mathsf{KDE}}(\mathbf{x}_i|\alpha)}{\sum_{\alpha'=1}^{C}\hat{\rho}_{\mathsf{KDE}}(\mathbf{x}_i|\alpha')\hat{p}(\alpha')}\right]. \quad (14)\end{aligned}$$

- The first term in Eq.(14) represents the loss we would expect from just predicting class probabilities from a frequency count of the classes in the data; we denote this term $\mathcal{L}_0$.
- The second term represents actual learning. If our model has managed to learn anything about class-assignment from the data, then we would expect this term to be negative and the total loss to be lower than $\mathcal{L}_0$.

With Eq.(14), we can define how much we expect to learn from from a given data set. We define the *learning quotient* $\mathcal{Q}$ as

$$\mathcal{Q} \equiv \frac{\mathcal{L}_0 - \mathcal{L}}{\mathcal{L}_0} = \frac{1}{N\mathcal{L}_0} \sum_{\alpha=1}^{C} \sum_{i=1}^{N} y_{i,\alpha} \ln \left[\frac{\hat{\rho}_{\mathsf{KDE}}(\mathbf{x}_i|\alpha)}{\sum_{\alpha'=1}^{C} \hat{\rho}_{\mathsf{KDE}}(\mathbf{x}_i|\alpha')\hat{p}(\alpha')} \right] \tag{15}$$

representing the fractional decrease in loss we expect from the naive class-frequency-based estimate of loss. Given that $\mathcal{L} \geq 0$ and $\mathcal{L} \leq \mathcal{L}_0$, we find that $\mathcal{Q} \in [0, 1]$.

Definition of Learning Quotient $\mathcal{Q}$:

The *learning quotient* is the fraction of total information we can extract from the data set $\mathcal{D} = \{\{\mathbf{x}_i, y_i\}; i = 1, \ldots, N\}$ by representing the classes $\{\alpha\}$ in terms of features $\mathbf{x}$.

Definition of Learning Quotient $\mathcal{Q}$:

The *learning quotient* is the fraction of total information we can extract from the data set $\mathcal{D} = \{\{\mathbf{x}_i, y_i\}; i = 1, \ldots, N\}$ by representing the classes $\{\alpha\}$ in terms of features $\mathbf{x}$.

Qualitatively, the learning potential tells us how predictive feature representation $\mathbf{x}$ is of class α. The value 0 corresponds to not at all predictive and 1 corresponds to maximally predictive.

Next, we want to apply the learning quotient on sample data sets by exploring three questions:

Definition of Learning Quotient $\mathcal{Q}$:

The *learning quotient* is the fraction of total information we can extract from the data set $\mathcal{D} = \{\{\mathbf{x}_i, y_i\}; i = 1, \ldots, N\}$ by representing the classes $\{\alpha\}$ in terms of features $\mathbf{x}$.

Next, we want to apply the learning quotient on sample data sets by exploring three questions:

- **Intuition:** Does the learning quotient Eq.(15) match our intuitive sense of how much we can learn given a feature set and labels?
- **Minimum Loss:** Does the Bayesian estimate of the loss Eq.(14) match the minimum value we expect from model training?
- **Feature Importance:** Can the learning quotient tell us how well various features (or their combinations) separate the classes of data?

- **Intuition:** Does the learning quotient Eq.(15) match our intuitive sense of how much we can learn given a feature set and labels?

Let's say we are trying to predict whether companies churn or renew their contract with our business. The predictor variable we want to use to determine the likelihood of renewal is "engagement score."

We have collected data on the engagement scores and renewal status of past companies.

	engagement_score	status
0	40.0	Churned
1	325.0	Renewed
2	320.0	Renewed
3	100.0	Renewed
4	235.0	Renewed
5	305.0	Renewed
6	260.0	Renewed

Figure 1: Data table for engagement score and renewal status

How can we know whether these engagement scores will be predictive of renewal status?

Let's say we are trying to predict whether companies churn or renew their contract with our business. The predictor variable we want to use to determine the likelihood of renewal is "engagement score."

How can we know whether these engagement scores will be predictive of renewal status?

- **Answer:** For a qualitative estimate, we can plot distributions of the the engagement scores for "Churned" and "Renewed" companies.

Let's consider two example datasets

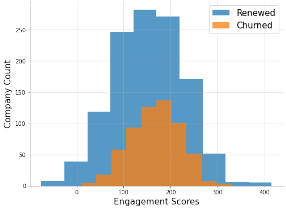

Figure 2: (Example) No clear separation between classes

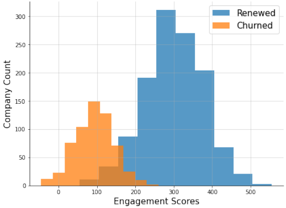

Figure 3: (Example) Fairly clear separation between classes

Let's consider two example datasets

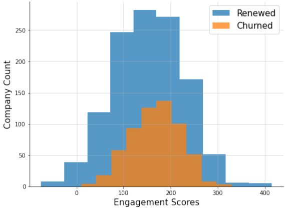

Figure 4: (Example) No clear separation between classes

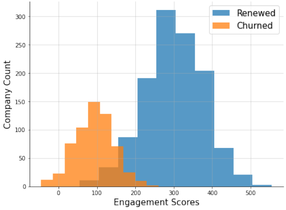

Figure 5: (Example) Fairly clear separation between classes

- Fig.4 shows no clear separation between the churned or renewed classes, so for this dataset we don't expect the feature to tell us much about renewal status.
- Conversely, Fig.5 does show fairly clear separation between the classes, so for this data set we expect the feature to tell us a lot about renewal status.

Can we reify these intuitions by calculating the learning potential?

Can we reify these intuitions by calculating the learning potential?

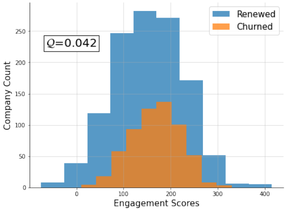

Figure 6: (Example) No clear separation between classes; Low learning potential

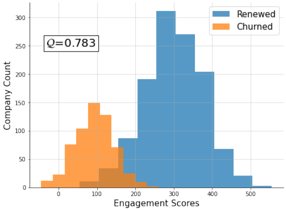

Figure 7: (Example) Fairly clear separation between classes; High learning potential

Computing the Learning Potential $\mathcal{Q}$ for both of these datasets, we find...

- Fig.6 shows that the learning potential for the nearly overlapping dataset is **very low** (i.e., ~ 0.0), indicating the feature has little ability to separate the classes
- Conversely, Fig.7 shows that the learning potential for the more separated dataset is **fairly high** (i.e., ~ 0.8), indicating that there is a lot of potential learning from this feature set

- **Minimum Loss:** Does the Bayesian estimate of the loss Eq.(14) match the minimum value we expect from model training?

We will consider the same single-feature data set from before (in particular the one with cleanly separated labels). We will train a neural network on this data set and track the loss for the training and validation set.

```
In [65]: from keras.models import Sequential
         from keras import layers
         from keras import backend as K

         input_dim = X.shape[1]  # Number of features

         model = Sequential()
         model.add(layers.Dense(2, input_dim=input_dim, activation='relu'))
         model.add(layers.Dense(1, activation='sigmoid'))

In [66]: model.compile(loss='binary_crossentropy',
                       optimizer='adam',
                       metrics=['accuracy'])

         K.set_value(model.optimizer.learning_rate, 0.0005)
         model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 2)                 4

 dense_1 (Dense)             (None, 1)                 3

=================================================================
Total params: 7 (28.00 Byte)
Trainable params: 7 (28.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 8: Two-layer neural-network Architecture

Does the neural network loss level off at a value that matches the predicted loss $\mathcal{L}$ in Eq.14?

Does the neural network loss level off at a value that matches the predicted loss $\mathcal{L}$ in Eq.14?

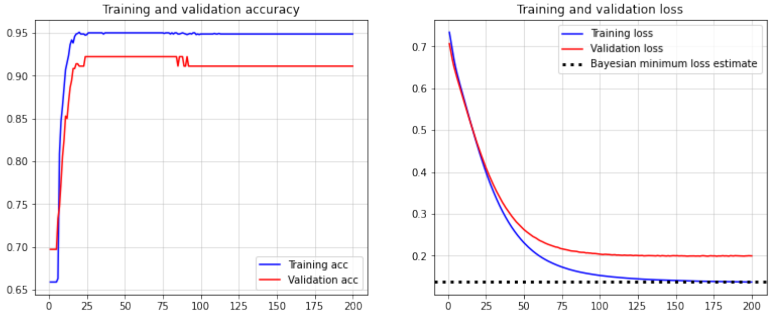

Figure 9: Accuracy and Loss for Network

- The neural network loss levels off at $\mathcal{L}$ meaning that by this point in training (i.e., around epoch f) we have exhausted the learning potential for this feature set.
- $\mathcal{L}$ does predict final loss value for training

- **Feature Importance:** Can the learning quotient tell us how well various features (or their combinations) separate the classes of data?

We again consider our example of predicting renewal status for a collection of companies.

But now we assume we have two features that we can use in the prediction.

	feature1	feature2	status
0	31.672545	38.098898	Renewed
1	34.362789	44.012386	Renewed
2	27.303062	38.053530	Renewed
3	32.832428	43.339298	Renewed
4	28.831346	49.473523	Churned
5	28.994351	46.263649	Churned
6	30.650861	35.684540	Renewed

Figure 10: Data table for two features and renewal status

We can plot distributions of these features for Churned and Renewed companies to get a sense of the ranges over which they vary

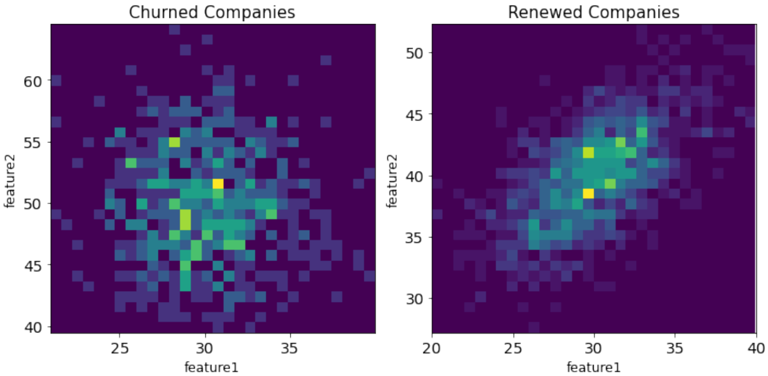

Figure 11: Distribution plots of the two features for churned and renewed companies

Can we use the learning potential to know which features are most essential for distinguishing the two classes?

Can we use the learning potential to know which features are most essential for distinguishing the two classes?

To answer this, we compute the learning potential in three cases:

- **feature1** is the only predictor variable for status
- **feature2** is the only predictor variable for status
- both **feature1** and **feature2** are predictor variables for statuses

The case with the highest learning potential corresponds to the feature set which can best predict renewal status.

Computing the learning potential for these three cases, we find

Feature Selection	**Learning Potential, $\mathcal{Q}$**
feature1	0.003
feature2	0.619
feature1 & feature2	0.683

Table 1: Learning Potential by Feature

Thus we see

- **feature1** has very low learning potential so it is not an essential feature
- **feature2** has a fairly high learning potential so it is an essential feature
- **feature1** and **feature2** together have only a slightly higher learning potential than **feature2** alone

Conclusion

$\Longrightarrow$ We can likely use **feature2** alone to predict renewal status

Conclusion and Summary

We found two main results:

$$\mathcal{L} = -\sum_{\alpha=1}^{C} \hat{p}(\alpha) \ln \hat{p}(\alpha) - \frac{1}{N} \sum_{\alpha=1}^{C} \sum_{i=1}^{N} y_{i,\alpha} \ln \left[\frac{\hat{\rho}_{\text{KDE}}(\mathbf{x}_i|\alpha)}{\sum_{\alpha'=1}^{C} \hat{\rho}_{\text{KDE}}(\mathbf{x}_i|\alpha') \hat{p}(\alpha')} \right] \tag{16}$$

[Bayesian Estimate of Loss]

$$\mathcal{Q} \equiv \frac{\mathcal{L}_0 - \mathcal{L}}{\mathcal{L}_0}; \quad \mathcal{L}_0 = -\sum_{\alpha=1}^{C} \hat{p}(\alpha) \ln \hat{p}(\alpha) \qquad \text{[Learning Quotient]} \tag{17}$$

And we found three applications of these results

- **Intuition:** Learning quotient can tell us how much a model can learn from a feature data set
- **Minimum Loss:** Computing the Bayesian expected loss and the loss of a trained model can provide a signal for when a model has been overtrained
- **Feature Importance:** Comparing learning quotients can tell us which combinations of features are most important

Fin