Data 37711: Machine Learning & AI Part I **Fall 2025**

Homework 2

Student: Chenxi Peng Due date: November 1^{th}

1 Q1

1.1 a

Let the neural network output be

$$g(x) = (u(x), v(x)) \in \mathbb{R}^2.$$

We define a single link function

$$h : \mathbb{R}^2 \to \mathbb{R}^2,$$

which maps the network output to the mean and variance functions:

$$(f_\mu(x), f_\sigma(x)) = h(g(x)).$$

A suitable choice for h is:

$$h(u, v) = \big(u,\ \text{softplus}(v)\big),$$

where

$$\text{softplus}(v) = \log(1 + e^v)$$

ensures $f_\sigma(x) > 0$.

Thus, the link function h converts the unconstrained neural outputs into

$$f_\mu(x) = u(x), \quad f_\sigma(x) > 0,$$

which are appropriate for modeling the conditional mean and variance, respectively.

1.2 b

Assume the model outputs $f_\mu(x)$ and $f_\sigma(x)$ (standard deviation)

The conditional distribution is $Y|X \sim \mathcal{N}\big(f_\mu(x),\ f_\sigma(x)^2\big)$

So define the loss function as the negative log-likelihood:

$$\ell(x, y) = -\log p(y \mid x) = \frac{1}{2}\log\big(2\pi\, f_\sigma(x)^2\big) + \frac{(y - f_\mu(x))^2}{2\, f_\sigma(x)^2}.$$

Empirical risk over n samples:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i) = \frac{1}{2n}\sum_{i=1}^{n}\left[\log\big(2\pi\, f_\sigma(x_i)^2\big) + \frac{(y_i - f_\mu(x_i))^2}{f_\sigma(x_i)^2}\right].$$

1.3 c

Metrics here are robust mean squared error (RMSE), mean absolute error (MAE), and negative log likelihood (NLL). From the Table 1, we can see that compared to the baseline linear regression model, MLP outperforms the baseline on the regression problem.

Table 1: Comparison of the performance of the multi-layer perception (MLP) and the linear regression

Metrics	Baseline	MLP
Conditional mean		
RMSE	7.490	4.210
MAE	5.552	2.734
Conditional variance and mean		
NLL	2.542	1.835

1.4 d

First of all, we should prepare the new feature vector. Then, use the fitted MLP model to estimate the conditional mean and variance for each trip. Assume the distribution of trip duration is normal for each trip. Based on the cumulative distribution function for normal distribution, we can get the probability that a trip will take less than 45 minutes for each trip.

Finally, for estimated probability greater than 0.5, we can assign the value of 1 to the binary variable $y\_pred\_binary_i$ corresponding to this trip. For the true trip duration, if the value is smaller than 45, we assign the value of 1 to another binary variable $y\_true\_binary_i$ corresponding to this trip.

In the end, by comparing the two binary variables, we can get the accuracy rate approximately 0.996.

For question 1, the complete code can be found at here.

2 Q2

2.1 a

The number of unique stock codes and customer IDs are 3665 and 4339, respectively. These two categories have high cardinality, so one-hot encoding is not feasible because we will end up with 8004 columns just for categorical variables, which would be extremely sparse, memory-heavy, and inefficient for neural networks.

2.2 b-e

Here we tried three encoding methods for the categorical variables. Target ecoded variables is inputted in a XGBoost model. The next two model is neural network models with an embedding layer. One with input categorial variables, the other with additional target encoded variables.

The performance in regression task is shown on the Table 2. We can see that the embedding-only model performs best overall. The embedding layer transfrom each categorical ID (like a stock code or customer ID) into a dense numeric vector. For example,

$$\text{CustomerID } 10011 \rightarrow [0.11, -0.21, ...]$$

These learned vectors can capture latent characteristics. For the stock codes, how items tend to sell together or product similarities may be captured. For the customer IDs, purchase behavior similarities and preference

patterns may be captured. When the neural network learns, it adjusts these embeddings so that products and customers with similar effects on quantity end up closer together in embedding space.

In this context, categorical IDs like `Stockcode` and `CustomerID` have high cardinality, with respect to 3665 and 4339 unique IDs. Target encoding gives only one number per category, which assumes each category's effect on `Quantity` is independent and linear. But embeddings allow the model to learn nonlinear interactions between products and customers and let similar customers or products share information through nearby vectors. So if a new product is similar to another in the embedding space, the embedding model can infer likely demand even without exact training data. It indicates that the embedding model can generalize better.

The reason why the hybrid model don't perform better than embedding-only model may be: 1. the target encoding may introduce a more "linear" signal that can slightly constrain the nonlinear flecibility; 2. the target encoding may duplicate information that embeddings are alrealy learning.

The complete code can be found here.

Table 2: Comparison of the performance of different methods for encoding categorial random variables

Metrics	**Target encoding**	**Embedding**	**Hybrid**
MSE	1089.393	892.821	951.241
MAE	8.927	7.735	7.929
R^2	0.347	0.465	0.429

3 Q3

Based on the training data, the visualization of true y over the `(x1,x2)` is shown on Figure 1.

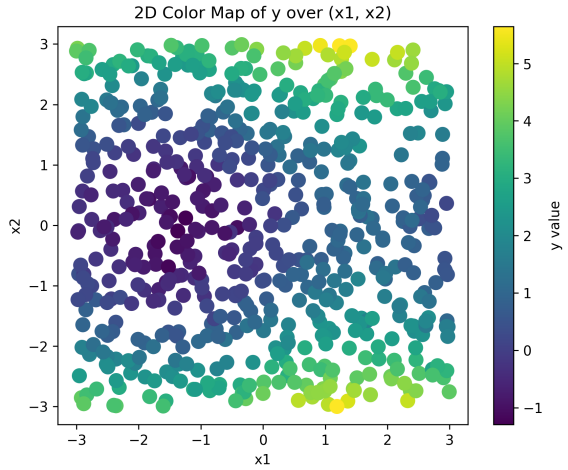

Figure 1: 2D grid-plot in training data

For constructing a large set of features and then fitting relatively simple models, Random Kitchen Sink (RKS) model and the Multi-Layer Perceptron (MLP) with one-hidden layer are implemented. By conducting two hyperparameter sweeps over the number of random features and hidden layer dimensions on the two models respectively, we get the relationship between MSE and the hyperparameter for each model. Figure 2 shows that given fixed other hyperparameters, the optimal number of random feature pairs is 50. Figure 3 shows that given fixed other hyperparameters, the optimal number of hidden layer dimensions is 256.

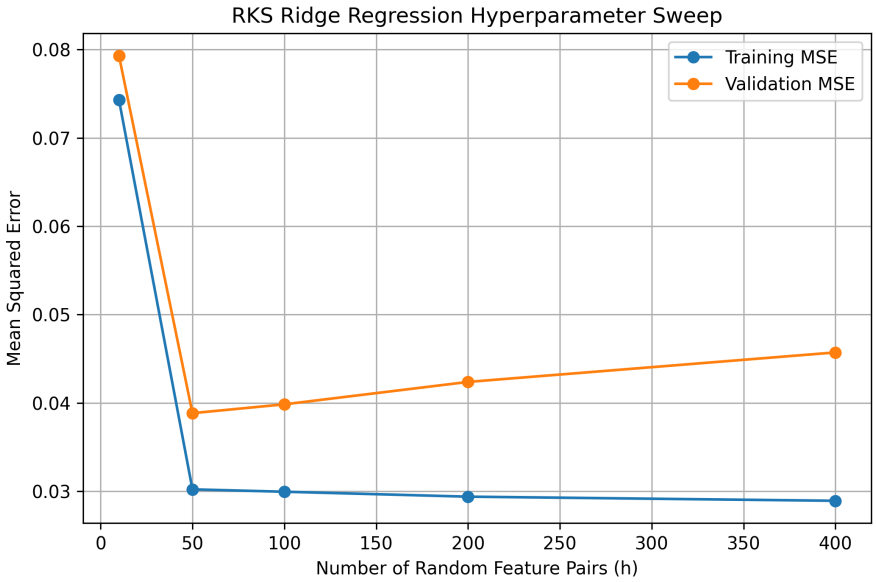

Figure 2: The relationship between MSE and the number of random feature pairs for RKS with ridge regression

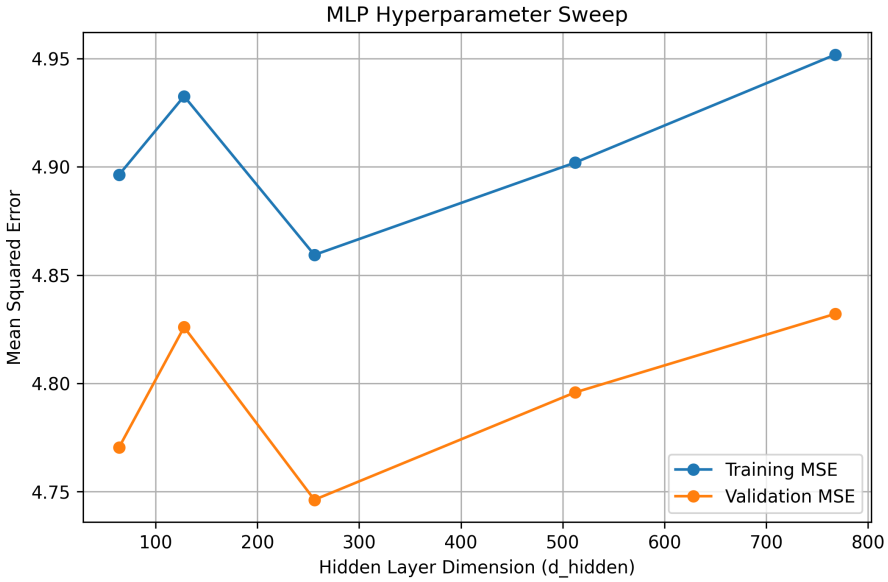

Figure 3: The relationship between MSE and the number of hidden layer dimensions for one-hidden layer MLP

After finding the best-performing hyperparameters, I evaluate both the RKS and MLP models on the test dataset. Table 3 shows that the RKS feature mapping + ridge regression is more effective at capturing the functional relationship between `x1, x2, and y` than the one-hidden layer MLP.

Table 3: Comparison of the performance of the two models on the test dataset

Metrics	**RKS**	**MLP**
MSE	0.0416	4.6414

Then, I visualize the fitted functions from the two models over the input space. Comparing Figure 4 and Figure 1, we can see that these two models both capture the relationship between input and target. However, the fitted surface from the RKS model closely aligns with the ground truth function, exhibiting smooth transitions and accurate capture of nonlinearity.

RKS approximates a Radial Basis Function (RBF) kernel by projecting data into a high-dimensional feature space using random sine and cosine bases. Then it fits a simple linear model in that feature space. So it's naturally good at modeling smooth, nonlinear relationships. Even with limited data, it generalizes well due to the strong inductive bias of the RBF kernel.

The MLP tries to learn the nonlinear mapping from data through a single hidden layer with ReLU activations. It's flexible but relies on gradient-based learning and hyperparameter tuning. Because it depends on initialization, optimization, and regularization, it may struggle to converge to a global minimum — especially with small data and without extensive tuning. So, the MLP captures some nonlinearity but fails to reproduce the overall smooth structure, leading the fitted surface to show irregular patterns.

The complete code can be found here.

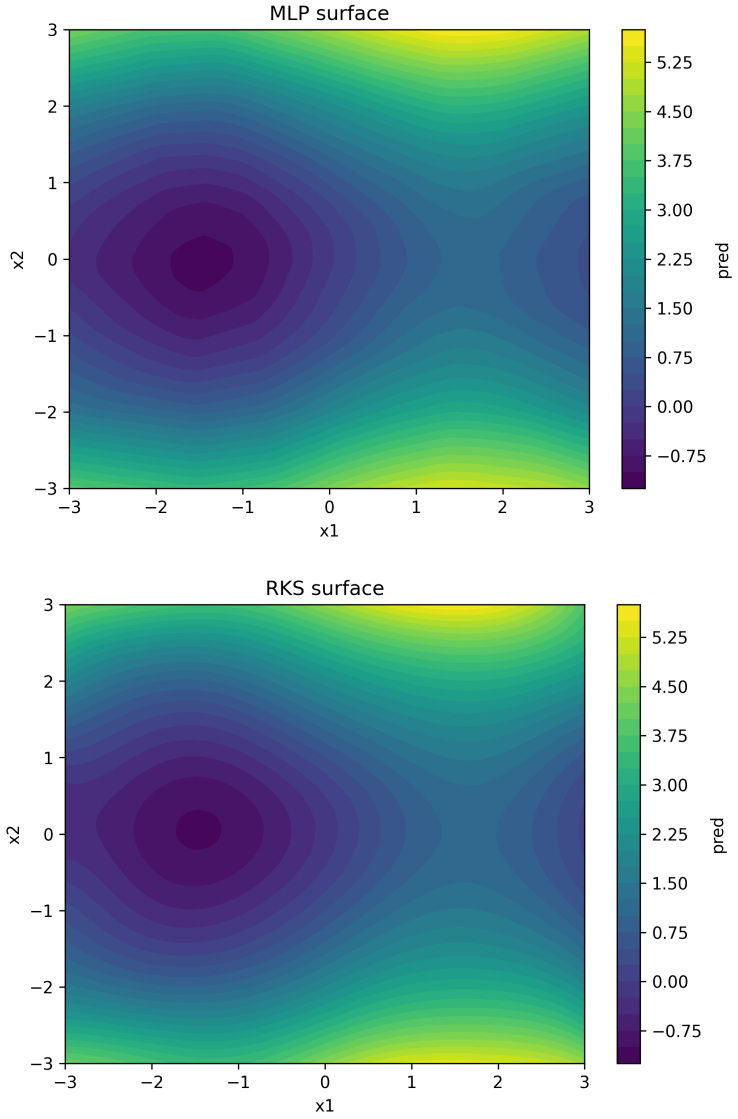

Figure 4: The visualization of the fitted functions from the two models over the input space