Take Home Exam

Machine Learning (PhD 430) – Fall 2023

Instructions

Follow the instructions below when completing the exam:

1. Work on the exam individually.
2. Submit your answers in PDF format.
3. On tasks labelled [Code], include screenshots of the modifications you made to the original code. Put the screenshots in context by describing what you changed and why.
4. On tasks labelled [Code], it is not necessary to achieve improvements on the machine learning task. It is sufficient to demonstrate that you understand the code and are able to make the modifications described in the task.
5. On tasks labelled [No Code], most answers should only require a couple of sentences or paragraphs; however, there is no minimum or maximum length.

Task 1: Autodifferentiation [No Code].

Autodifferentiation (`AutoDiff`) is used in libraries such as `TensorFlow` and `PyTorch` to simplify the computation of derivatives and improve their accuracy relative to numerical methods. `AutoDiff` is typically used to train deep learning models, but can be applied to the computation of derivatives in general.

Assume you are given the following loss function, L, where x is a feature, y is the target, and b and w are model parameters:

$$L = g(f(x)) = (y - wx - b)^2, \tag{1}$$

where

$$g(z) = z^2 \tag{2}$$

and

$$f(x) = y - wx - b \tag{3}$$

Following the approach described on slide 109 of Lecture 1, write down the steps for calculating $\frac{\partial L}{\partial w}$ using the three methods below. In each case, evaluate the derivative at $x = 2$ and $y = 3$. Assume that $b = 0$ for simplicity, and denote the step size for the forward difference method as h.

(a) Symbolic differentiation.

(b) The forward difference method.

(c) Autodifferentiation.

Task 2: Optimization in TensorFlow [Code].

The following notebook provides you with code to train an autoregressive model of the USD-GBP exchange rate in `TensorFlow`: Notebook: Task 2. The model is defined as follows:

$$y_t = \alpha + \rho y_{t-1} + \epsilon_t \tag{4}$$

We can obtain estimates of the model parameters by minimizing the mean squared prediction error:

$$L = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \alpha - \rho y_{t-1}\right)^2 \tag{5}$$

Make the changes to the code described below. In each case, provide a screenshot and a brief explanation of the changes you made to the code.

(a) Modify the code to minimize the mean of the absolute value of the model prediction errors, rather than the mean squared error. (Hint: You can find the list of available loss functions here.)

(b) Train the model using a different optimizer. (Hint: You can find the list of available optimizers here.)

(c) Use a different initialization for the parameters. Comment on whether this changes your parameter estimates.

Task 3: Gradient Descent [No Code].

It is common to minimize the loss function for economic and financial models by performing gradient descent. The parameter update rule is given below:

$$\theta := \theta - \alpha \nabla_\theta J(\theta), \tag{6}$$

where θ is the parameter vector, α is the step size, and $\Delta_\theta J(\theta)$ is the gradient of the parameter vector.

Answer the following questions:

(a) Why is the gradient used to update θ?

(b) What are the advantages and disadvantages of increasing α?

(c) How does stochastic gradient descent (SGD) differ from gradient descent?

(d) What is mini-batch stochastic gradient descent?

(e) Why is SGD and mini-bach SGD (rather than gradient descent) commonly used to train deep learning models?

Task 4: Neural Networks [Code].

You are given code in the following notebook file that trains a small neural network using `TensorFlow` to predict U.S. CPI inflation using lags of inflation and unemployment: Notebook: Task 4. You are also given U.S. data on the growth rate in hours worked, earnings, and a measure of the M1 money supply.

In each part of this task, you will attempt to improve the predictive performance of the model. Include screenshots documenting your changes to the code, along with brief explanations of your decisions.

(a) Train the model using the code provided without modification and discuss its performance in the training and test sets.

(b) Add more features to the model and evaluate its performance.

(c) Add another hidden layer to the model and evaluate its performance.

(d) Change the model's hyperparameters, such as the number of epochs, and evaluate its performance.

(e) The model uses dense layers. What other layers might be used to process sequential data, such as the time series used? You do not need to add them to the code.

Task 5: Tree-Based Models [Code].

You are given the following notebook file, which uses `sklearn` to train gradient boosted trees to predict financial crises using data from the Macro History database: Notebook: Task 5.

Execute the code in the notebook. To interpret the results, you may need to read the documentation for the accuracy metric and confusion matrix. After you have done that, answer the following questions. Where changes in the code are needed, provide screenshots of your modifications, along with a brief explanation of what you did and why.

(a) What could explain the high classification accuracy in the training set?

(b) In the test set, does the model tend to mis-classify non-crises as crises? Or does it tend to mis-classify crises as non-crises? Explain how you determined this.

(c) Based on the model's performance in the training and test sets, does there appear to be an issue with overfitting? Explain your reasoning.

(d) Adjust the model's hyperparameters, change the set of features used as inputs, and then compare your results to the original model's performance.

Task 6: Dimensionality Reduction and Clustering [No Code].

Answer the following questions:

(a) When performing k-means clustering, what approach can we use to avoid getting stuck in local minima?

(b) Explain the difference between partial least squares (PLS) and principal component analysis (PCA).

(c) Explain how an autoencoder can be used to perform dimensionality reduction.

Task 7: Regularized Regression [No Code].

Assume you are estimating a linear model with p features and n observations using a regularized regression. You minimize the following objective function to recover the model's parameters.

$$\mathrm{L} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{7}$$

Answer the following questions:

(a) Is it possible to use this approach to recover the model's parameters if $p > n$? Explain your answer.

(b) How does the loss function differ from OLS? How does it differ from ridge regression? What implications does this have for the parameter estimates?

(c) What procedure can we use to select λ?

Task 8: Natural Language Processing [No Code].

You are given the following example paragraph from a Riksbank statement:

> *Economic activity in Sweden remains strong and inflation is close to the target of 2 per cent. Uncertainty abroad has increased but new information since the monetary policy decision in April has not led to any major revisions of the forecasts overall. With continued support from monetary policy, the conditions for inflation to remain close to the target in the period ahead are considered good. The Executive Board has decided to hold the repo rate unchanged at –0.25 per cent. The forecast for the repo rate is also unchanged and indicates that it will be increased again towards the end of the year or at the beginning of next year. However, the risks surrounding developments abroad can have a bearing on the prospects for Sweden, which emphasises the importance of proceeding cautiously with monetary policy.*

Assume you have all such paragraphs from statements made by the Riksbank over the period between 2001 - 2023. You also have all of its policy rate decisions for the same period of time.

Answer the following questions:

(a) What are three different types of textual features you could measure in central bank statements, such as the one given above, that might be useful for predicting interest rates?

(b) What methods in natural language processing would you use to measure those features?

(c) How would those textual features be represented numerically?

(d) What types of models could you use to predict the policy rate using your textual features?