ML Kit: A Machine Learning Library for Rust

Owen Wetherbee (ocw6), Ethan Ma (em834), Sylvan Martin (slm338)

Keywords: Machine learning, Gradient Descent, PCA, Rust

Application Setting

Rust is a relatively new programming language with a dearth of machine learning infrastructure. We aimed to create a library for Rust programmers to make machine learning convenient, much like NumPy or TensorFlow in Python.

Project Description

Initially, we wanted to create a comprehensive machine learning library for Rust, which would use Rust's safety and speed to implement many algorithms used in data science. We had the original (lofty) goal of being able to run a diffusion model by the end of the semester, or be able to do anything that NumPy/SciKitLearn could do. As we began implementation we recognized that we lacked the time and resources to implement the sheer amount of algorithms we set out to. So, we shifted our focus to what we believe are some of the core algorithms in the field of Machine Learning.

In the end, we created the pure-Rust library `ml_kit`, which implements, from scratch, the following:

- Neural Network based learning, consisting of
  - the basic neural network model with user-defined network shape and activation functions for each layer
  - functionality for handling large datasets for training and testing
  - a stochastic gradient descent trainer, with whatever batch size and epochs a user may want
  - various "gradient update schedules," such as fixed learning rates, time-decay learning rates, AdaGrad, etc.
  - Convolutional Neural Networks (Owen/Ethan talk more on this?)

- Principle Component Analysis, consisting of
  - an implementation of Singular Value Decomposition (SVD)[1],
  - using SVD to obtain a k-dimensional plane of best fit for a set of points in $\mathbb{R}^n$,
  - compressing images (or any data) using SVD by truncating low-variance dimensions

Relationship to Other Work

Linear algebra is the foundatin to machine learning, so we needed to use a good linear algebra library. Sylvan had previously spent winter break working on `matrix_kit`, which is a pure-Rust linear algebra package that implemented incredibly basic matrix-vector operations. We continued developing this library in parallel with `ml_kit` over the semester as we recognized more features that were needed from the linear algebra library. So, the sum of our work for the course can be thought of as the entirety of `ml_kit`, as well as significant improvement to the functionality of `matrix_kit`.

The `matrix_kit` library can be found on GitHub at https://github.com/SylvanM/matrix_kit.

Evaluation

Gradient Descent

Principle Component Analysis

we first plan to implement the standard minimization techniques such as Newton's method and gradient descent, including stochastic gradient descent and AdaGrad (i.e. with AdaBoosting). In addition, we will implement the Perceptron (for linearly separable data), hard and soft support vector machines (SVMs) with kernels, and various k-clustering algorithms (k-nearest neighbors and k-means)." We further emphasized a possibility of extending upon these algorithms, especially in relation to convolutional and recurrent neural networks, and even transformers if time allowed. The project aimed to enable Rust users to get started with using machine learning for various

1. SVD is the process of taking a matrix A and factoring it as

$$A = U\Sigma V^\top$$

where U and V are orthogonal and Σ is diagonal.

applications. We also hope it can serve as a starting point from which more sophisticated machine learning algorithms can be added and implemented in the future. - **Developments:** - **Results:**