ML Frameworks

Projects are done individually, to make sure that you all have experience enough to use machine learning frameworks on your own.

Motivation. This is a somewhat open-ended project. It is designed to be an introduction to ML frameworks and tools for those who need it, while also providing the opportunity for those who may be an expect in ML frameworks an opportunity to learn new things. In particular, there are the following learning goals:

- Practice writing code that reads an external dataset and makes it available for use in an ML framework.
- Implement SGD in a general numerical framework, such as numpy.
- Test and evaluate SGD written in a general numerical framework.
- Implement SGD in a machine learning framework that supports backpropagation, such as PyTorch.
- Test and evaluate SGD in a machine learning framework that supports backpropagation.
- Explore the use of other learning algorithms that we covered in class, and see how this is facilitated by the ML framework.

At several points in this project description, you will be presented an alternative choice to one or more of the project instructions. The goal of these choices is to give you more options for exploring aspects of ML frameworks that you might not be familiar with. When making one of these choices, consider your own learning goals, and select the option that will best advance them. If you do make an alternate choice, be sure to indicate that in your project report.

Overview. In this project, you will be implementing and evaluating stochastic gradient descent for empirical risk minimization on a logistic regression task on the **Adult** dataset. This dataset is available publically from the UCI data repository; here for simplicity we will be using the **a9a** featurization of the dataset available from the LibSVM datasets page here (these are both great sources for simple ML datasets to use for testing, by the way). The goal of the Adult dataset is to predict based on census features of an individual American adult whether that person's income exceeds $50k/yr.

The datasets we will use for this assignment are available here: a9a.train and a9a.test. These are just local copies of the data available on the LibSVM datasets website. The format for the datasets is as follows. Each line represents a training example. The first number on each line is the training label y_i. The remaining entries on each line are pairs of the form `[feature index]:[feature value]`. The features are 1-indexed. There are 123 total features in this dataset. The dataset is presented in this file in a sparse manner: features that are not present have value 0 implicitly.

Part 1: Deriving SGD for logistic regression. Two-class logistic regression in this setting has loss function

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-w^T x_i y_i)),$$

where $w \in R^d$ are the weights, $x_i \in R^d$ and $y_i \in \{-1,1\}$ are the i^{th} training example and training label, respectively, and n is the number of training examples. For this model, the prediction made is the simple linear classifies

$$h_w(x) = \mathbf{sign}(w^T x).$$

In this assignment, we are going to train such a two-class classifier.

Derive an expression for the gradient of the component loss function f_i with respect to the weights w. Also write out an expression for a single update step of SGD (with mini-batch size 1). Include these expressions in your report.

Part 2: Loading the data. Write code to load the training and validation data for the Adult dataset above into numpy, in such a format as you can use it later. Do this by loading the data into a dense numpy matrix. Importantly, when loading data for a linear model like this, we want to append an "extra" feature that is always 1: this feature is used to represent the bias term in the linear model representation. Thus, the matrix of examples you load should be of size $X \in R^{124 \times 32561}$ for the training set and $X_{val} \in R^{124 \times 16281}$ for the validation set.

Alternatively, you may choose to represent your data as a sparse matrix. Evaluate how the performance of this approach compares to the dense matrix approach.

Optional: If you wish, you may choose to use Julia as your numerical framework instead of numpy. If you do, use it for the rest of this assignment, whereever you are instructed to run something in numpy. While Python and numpy are more commonly used for ML tasks, if you feel that you already have mastery over them and would like to learn another tool, Julia is a good choice.

Part 3: Implementing stochastic gradient descent. Implement the following three functions in numpy.

- A function named `logreg_loss` that, given a dataset of examples $\boldsymbol{X}$ and example labels $\boldsymbol{y}$, and a vector of model weights ww, computes the loss $\boldsymbol{F}$ of that model on that dataset.
- A function named `logreg_error` that, given a dataset of examples $\boldsymbol{X}$ and example labels $\boldsymbol{y}$, and a vector of model weights ww, computes the error of that model on that dataset. The error is the fraction of examples that the model predicts incorrectly, and should be a number between 0 and 1.
- A function named `logreg_sgd` that, given a dataset of examples $\boldsymbol{X}$ and example labels $\boldsymbol{y}$, a vector of initial model weights ww, a learning rate $\boldsymbol{\alpha}$, an ℓ_2 regularization constant $\boldsymbol{\sigma}$, and a number of steps $\boldsymbol{T}$, runs SGD on that dataset for that number of iterations, returning the resulting model. This function should use a minibatch size of 1, and should sample a new training example uniformly at random from the training set at each iteration.

Part 4: Evaluating stochastic gradient descent. Run your implementation of SGD with the following parameters:

- Use step size $\boldsymbol{\alpha}$=0.001.
- Use regularization $\boldsymbol{\sigma}$=0.0001.
- Use initial model $\boldsymbol{w_0}$=0.
- Run for a number of steps equal to 10 epochs, where each epoch involves a number of iterations equal to the number of examples in the training set, $\boldsymbol{T}$=32561 × 10.

Before training and after each epoch (i.e. a total of 11 times after iteration numbers t=0,32561,…), measure the following

- The training loss.
- The training error.
- The validation error.

This setting should achieve validation error and training error of around 15%. Plot these observed values (on a figure with iterations on the x-axis) and include the resulting figures in your report.

Part 5: Simple hyperparameter exploration. Now explore how the performance of the training algorithm is affected by different hyperparameter choices. Can you identify a better setting of hyperparameters that improves the accuracy or lets us train in less iterations?

- Based on your intuition and/or what we learned in class, form a hypothesis about how the observed performance of the training algorithm will change if you change one or more hyperparameters. Include this hypothesis in your report.
- Describe what experiment or experiments you will run to evaluate this hypothesis, and explain briefly why this is appropriate. Include this experiment description in your report.
- Run the experiments.
- Report the results of your experiments. Do the results support or refute your hypothesis? Explain.

Alternatively, you may choose to explore other hyperparameters, which we have not included in this assignment description. For example, you may choose to add momentum or evaluate minibatching.

Part 6: Logistic regression in PyTorch. Now set up the logistic regression objective in PyTorch. Hint: you can use PyTorch to specify the loss function as a shallow neural network. Then, run stochastic gradient descent on the problem in PyTorch using the same parameters you used above. (Do not worry about what method PyTorch uses to pick what sample is chosen in each step; just go with whatever you think is appropriate.)

Alternatively, you may choose to use a ML framework other than PyTorch for this part and Part 7. If you do, be sure to use a framework that supports automatic differentiation. Choices here include TensorFlow, MXNet, and Flux. If you are using TensorFlow, you may find the `tf.keras.optimizers.SGD` and `tf.keras.losses.BinaryCrossentropy` classes to be useful here, if you are taking the Keras route.

Measure and plot the same values (training loss, training error, validation error) that you did in the numpy case, and include the resulting figures in your report.

How does the performance of PyTorch compare with your numpy implementation? Was one faster? Was one more accurate? Was one easier or harder for you to implement? Explain in a paragraph in your report.

Part 7: Other Optimization Algorithms in PyTorch. PyTorch makes it easy to "switch out" the optimization algorithm for something else. For example, here is a list of the optimizers that PyTorch supports out-of-the box. (Note that the SGD optimizer also supports momentum natively, so there is no need for a separate optimizer class for momentum.) Can we find an optimizer that works better for this problem than SGD?

- Based on your intuition and/or what we learned in class, form a hypothesis about how the observed performance of the training algorithm will change if a different optimizer (e.g. Momentum, Adam) is used. Include this hypothesis in your report.
- Describe what experiment or experiments you will run to evaluate this hypothesis, and explain briefly why this is appropriate. Include this experiment description in your report.
- Run the experiments.
- Report the results of your experiments, including at least one figure. Do the results support or refute your hypothesis? Explain.

Project Deliverables. Submit two files: ***pdf*** version of your report and a single ***ipynb*** file of your code. The lab report should contain at least the following.

1. A 1-2 paragraph abstract summarizing what you did in the project.
2. Expressions for the stochastic gradient and SGD update step for logistic regression.
3. Figures displaying the training loss, training error, and validation error across training from Part 4.
4. Your hypothesis, experiment description, and results from Part 5.
5. Figures displaying the training loss, training error, and validation error across training from Part 6, and a paragraph comparing the performance.
6. Your hypothesis, experiment description, and results from Part 7.