David D'Attile
Prof. Kauchak
CS158
1 March 2022

Assignment 06: Gradient Descent

1 and 2. *One of the challenges with implementing machine learning algorithms, particularly iterative approaches like this, is determining if they are operating correctly. Include a short (~1 page), concrete justification of why your implementation of the hinge loss with L2 regularization is correct. Your argument must include two arguments:*

- Create a small example dataset (e.g., two examples and two features) and work through an iteration or two by hand (or using excel or something like that). Add print statements to your code to show that your code gets the same calculations. Include this comparison in your writeup in a way that helps argue for the correctness of your approach.

- Include another type of argument for the correctness of your code, e.g., could include snippets of code, along with an explanation.

For the first part of this questions, I have created a demo data set with the following values:

consumesCaffeine	consumesAlcohol	passesML
1	0	1
0	1	-1

To help convince ourselves that my gradient descent with hinge loss and L2 regularization is implemented correctly, I will hand-calculate the weight changes for each example through the first training iteration then compare these values to those which are produced by the implemented algorithm. Given that these examples only have two features, denoted f_1 (consumesCaffeine) and f_2 (consumesAlcohol), we know our model will contain two respective weights and a bias, denoted w_1, w_2, and b that will be initialized at [0,0] and 0. These weights will be updated at every observed feature by the equation $w_j = w_j + \eta((y_i * f_{ij} * max(0, 1 - y_i y_i')) - (\lambda * w_j))$ and the bias will be updated at every observed example by the equation $b = b + \eta((y_i * max(0, 1 - y_i y_i')) - (\lambda * b))$ where "*i*" denotes the *ith* example.

The four weight hand calculations take place in the following table:

Weights/ bias at start	Example, feature indices	Label, prediction	Old, new weight	Old, new bias	Weights at end
[w_1,w_2, b] = [0,0,0]	0,0	1,0	0, 0.01	0,0	[w_1,w_2, b] = [0.01,0,0]
[w_1,w_2, b] = [0.01,0,0]	0,1	1,0	0,0	0,0.1	[w_1,w_2, b] = [0.01,0,0.1]
[w_1,w_2, b] = [0.01,0,0.1]	1,0	-1,0.01	0.01,0.009999	0.01,0.01	[w_1,w_2, b] = [0.009999,0,0.1]
[w_1,w_2, b] = [0.009999,0,0.1]	1,1	-1,0.01	0,-0.0101	0.1, $-1.01xE^{-4}$	[w_1,w_2, b] = [0.009999,-0.0101,$-1.01xE^{-4}$]

And sure enough, the following is printed by our code at each example of one iteration given identical test data:

```
GD classifier weights for example 0 at iteration 0
0:0.01 1:0.0 b:0.01

GD classifier weights for example 1 at iteration 0
0:0.009999000000000001 1:-0.0101 b:-1.0100000000000004E-4
```

So, we have at least seen that for our basic dataset and only one iteration, we are getting the expected weight values for our gradient descent with hinge loss and L2 regularization based on the training data. However, we can look at the snippets of code that define our loss and regularization functions to further convince ourselves.

```java
private static double calculateLoss(double label, double prediction, int loss) {

    // init loss calculation
    double lossCalc;

    // calc based on loss fxn
    switch (loss) {
        case HINGE_LOSS -> lossCalc = Math.max(0, 1 - (label * prediction));
        case EXPONENTIAL_LOSS -> lossCalc = Math.exp(-(label * prediction));
        default -> {
            String msg = String.format("expected valid loss specification for calculation, received %d", loss);
            throw new IllegalArgumentException(msg);
        }
    }

    return lossCalc;
}
```

Loss Calculation

For our loss calculation, we can see that we rely on the inbuilt Java “Math” package, which is leveraged consistently for operations such as .max() and .exp(). Moreover, we can see that our calculations for hinge and exponential loss are aligned with the material from lecture, i.e.:

- HingeLoss(y,y’) = max(0, 1-(y*y’))
- ExponentialLoss(y,y’) = exp(-(y*y’))

```java
private static double calculateRegularization(double weight, int regularization) {

    // init regularization calculation
    double regCalc;

    // calc based on regularization fxn
    switch (regularization) {
        case NO_REGULARIZATION -> regCalc = 0;
        case L1_REGULARIZATION -> regCalc = weight >= 0 ? 1.0 : -1.0;
        case L2_REGULARIZATION -> regCalc = weight;
        default -> {
            String msg = String.format("expected valid regularization specification for calculation, received %d", regularization);
            throw new IllegalArgumentException(msg);
        }
    }

    return regCalc;
}
```

Regularization Calculation

Our regularization calculation is even more simple in code. For no regularization, we yield zero. For L1 regularization, we yield the sign of the passed weight. For L2 regularization, we simply yield the weight itself. These calculations are aligned with our lecture material and textbook.

3 and 4. *For hinge loss with no regularization and the default parameters, plot the loss function summed over all of the training examples with respect to the iterations, i.e., one value for each iteration. Repeat this experiment and plot the loss using hinge loss with L2 regularization. Write 2-3 sentences discussing these two experiments*:

In the above graph, we can see that the loss function values (calculated as $L = hingeLoss(y, y') + \lambda/2 * ||w||^2$ for L2 regularization and $L = hingeLoss(y, y')$ for no regularization) at each iteration for gradient descent trend downwards as the classifier is trained which indicates the desired minimizing of the loss function. With L2 regularization, we see an overall higher loss since we are trying to generalize the model from the given training data rather than train exactly to it.

These experiments were conducted on an 80/20 train/test data split of the binary titanic data.

5. *Run one additional experiment that highlights something interesting about the algorithm, include the resulting data, and include a few sentences of explanation/analysis. For example, you could investigate the optimal λ/η for one of the variants on one of the data sets, or you could investigate how the different loss/regularization approaches work (though pick some reasonable lambda). Plan on spending around an hour playing around with this*:

For my experiment, I attempted to isolate the "best" λ/η values between 0.001 and 1.000 for the gradient descent classifier with default hyperparameters of exponential loss and no regularization classifying the binary Titanic data set. To do this, I split the data into an 80/20 train/test ratio. Then, in increments of 0.001, I trained/tested the classifier to get testing accuracies averaged over 100 runs on the training data with the following tests for each increment:

- lambda = incremented value and eta = 0.01
- lambda = 0.01 and eta = incremented value
- both lambda and eta = incremented value

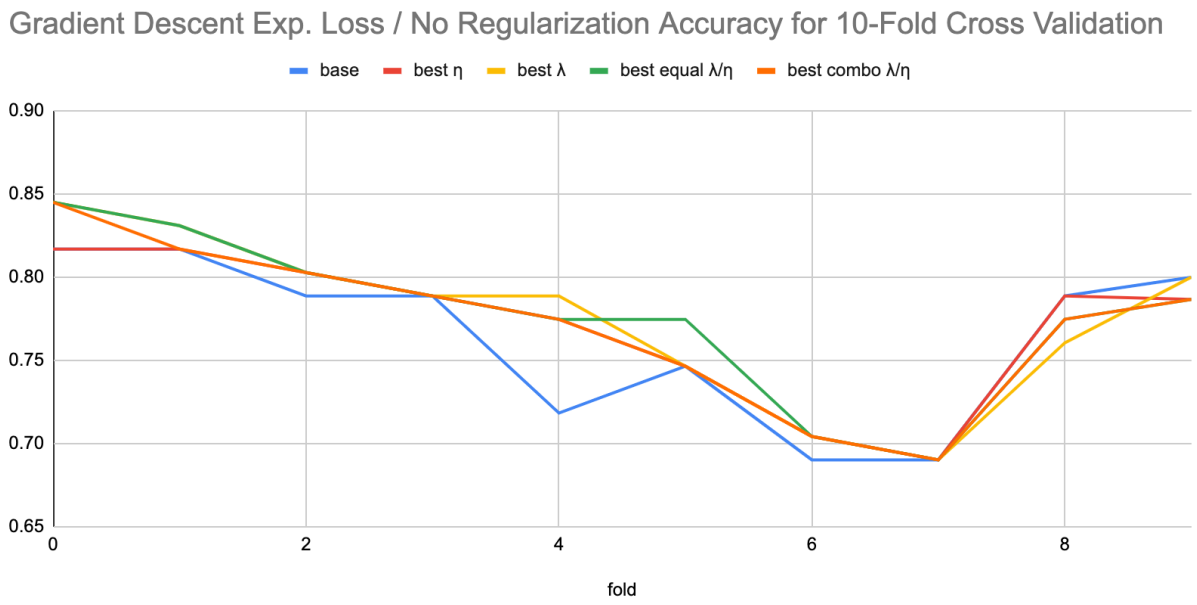
After conducting these 1,000 tests (raw data available at https://docs.google.com/spreadsheets/d/1hILLjl923dlUbTqA9tzESAyNrpHAXKr9WG3OXeI8Qjg/edit?usp=sharing), I determined the best λ/η values to be:

- λ = 0.437 when η = 0.01: 76.2238% accurate
- η = 0.002 when λ = 0.01: 76.1678% accurate
- λ = η = 0.001: 76.1818% accurate

Once these values were isolated, I hoped to prove via a paired t-test that fine-tuning these hyperparameters would lead to developing a provably more accurate model. So, I split the binary titanic data into a 10-fold cross validation set and gauged the classifier accuracy for the following models with the results stated in the graph below:

- "base": λ = 0.01 and η = 0.01
- "best η": λ = 0.01 and η = 0.002
- "best λ": λ = 0.437 and η = 0.01

- “best equal λ/η”: λ = 0.001 and η = 0.001
- “best combo λ/η”: λ = 0.437 and η = 0.002



As we can see from the graph, regardless of the tuning of λ/η, our models all performed quite similarly over the cross validation set. The base model tended to do the worst when compared to the “tuned” models, but not in a statistically significant manner as demonstrated by the t-test table below. Thus, even though we can tune the values of λ/η between 0.001 and 1.000 and get somewhat better results on the binary Titanic data set, we cannot claim that these results are statistically significant.

Best η vs. Base:	0.2660316064
Best λ vs. Base:	0.1964884233
Best Equal λ/η vs. Base:	0.09165998534
Best Combo λ/η vs. Base:	0.2336490702