Project A - Problem 2

Camelia D. Brumar

October 27, 2020

1 Methods for Sneaker-Sandal

1.1 Experimental Design

How did you divide the provided labeled set to develop models?

Answer: I used a variation of the nested cross validation approach for this project. First, I split the training data into A) training (90%) and B) testing (10%). Then, I used A for both 5-fold CV and for the final selected hyper-parameters. Next, I used B as my own test set, for examining the final models one last time.

I used this approach since half of the training data are sandals, and the other half are sneakers (thus the data set is balanced), so I wasn't worried that the folds wouldn't have enough data of each type of shoe. According to the textbook, this approach reduces the variability in the estimated test MSE, and at the same time it does not overestimate the test error rate for the model fit on the entire data set.

1.2 Method 0: Baseline

1. Describe the feature transformation you tried and why you thought it would work.

Answer: For the baseline method, I used the raw pixel values in order to get, as the name of the method suggests, a baseline for the errors of the upcoming methods.

2. Describe your model fitting process: what were parameters, and how were they fit? are there concerns about overfitting? concerns about convergence of the optimization?

Answer: For the logistic regression, the parameters are the weights that are multiplied by each raw pixel in the image.

There are no concerns about overfitting because I specifically chose the model that minimizes the validation BCE. In this case, it is C = 0.316.

I have a slight concern about the convergence since I was receiving warnings that the maximum number of iteration has been reached, but increasing the value of max_iter would have meant more time to wait. Due to working alone and the time constraint, I decided to ignore those warning and fix the maximum number of iterations to considerable value (1000).

3. Describe your hyperparameter selection process: what were hyperparameters, and how were they selected? What candidate values were considered? What performance metric did you try to optimize?

Answer: For the logistic regression I used the following hyperparameters:

- C, the inverse of regularization strength. This one was chosen using grid search and 5-fold CV. The grid is defined as follows: $C\_grid = np.logspace(-9, 6, 31)$. The metric I tried to minimize by doing this grid search is the average CV BCE (see table 1).
- solver, or the algorithm used in the optimization problem, fixed to 'lbfgs'.
- max_iter, or maximum number of iterations taken for the solvers to converge, fixed to 1000. I could have done CV to find this parameter as well but the grid search for the C hyper-parameter already took a minute or so to complete. I specifically set it to this value because it seemed to work just fine in problem 1, where the images were of the same size.
- Threshold fixed to 0.5. Again, I could have used CV to choose it, but due to time constraints, I decided to fix it.

1.3 Method 1: Logistic Regression + Average number of "on" Pixels per Image Feature

1. ...

Answer: In this case, I kept the raw pixels for each image, but I also added an extra feature which encodes the average number of "on" pixels in each image. That is, for each image, I counted how many pixels had a value greater than 0.0, added 1 to a counter variable, and after doing so for all pixels in an image, I divided that number by the total number of pixels in the image, i.e. 28^2, which is 784. I thought that this is an appropriate feature transformation since the sneakers seem to have more pixels that are "on" than the sandals in average.

2. ...

Answer: For this model, the parameters are the weights that are multiplied by the raw pixels, plus the weight that goes with the extra feature I've created for the average number of "on" pixels in each image.

There are no concerns about overfitting because I specifically chose the model that minimizes the validation BCE.

I have a slight concern about the convergence since I was receiving warnings that the maximum number of iteration has been reached, but increasing the value of max_iter would have meant more time to wait. Due to working alone and the time constraint, I decided to ignore those warning and fix the maximum number of iterations to considerable value (1000).

3. ...

Answer: The same ones as for Method 0, but now the hyperparameter C that minimized the avg validation BCE is $C = 0.484$. All the other answers coincide with the answers for model 0.

1.4 Method 2: Logistic Regression + Edge Detection Convolution

1. ...

Answer: For this model I used convolutions as a feature transformation. For details about what a convolution is, please see https://en.wikipedia.org/wiki/Kernel_(image_processing). First, I created a 3x3 filter that looks like this:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

This filter in particular makes the horizontal edges to stand out more. Then, I convoluted each image with the above filter. The code used to do these operations is written on top of the numpy library and inspired from the following source: https://towardsdatascience.com/building-convolutional-neural-network-using-numpy-from-scratch-b30aac50e50a.

I thought it would work after looking at some examples of how the images look like after applying this particular filter. For example, I thought that for sandals, there would be more defined horizontal lines than for sneakers.

2. ...

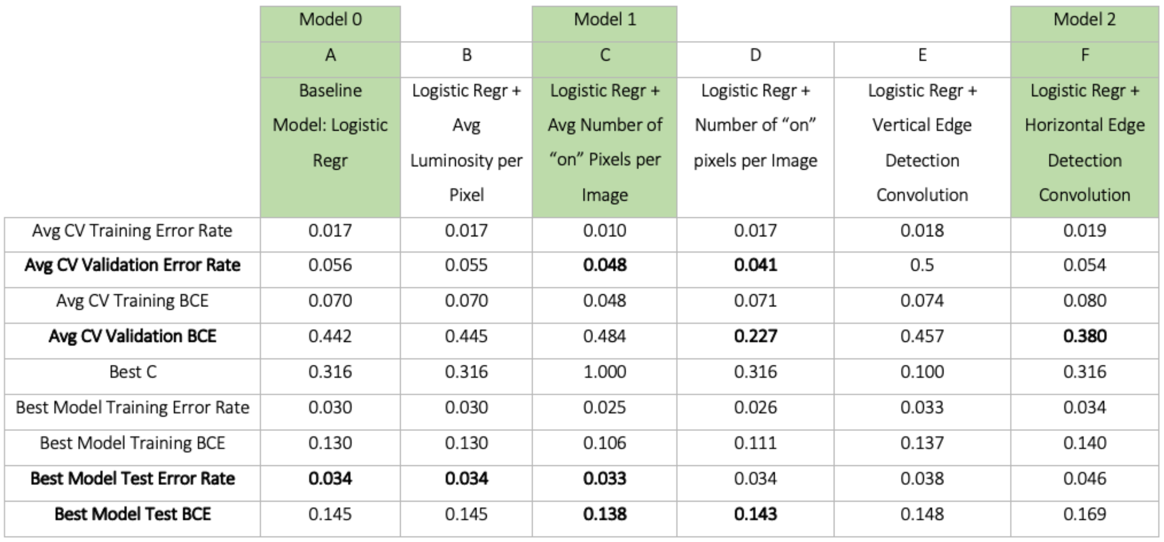
Answer: The parameters are the weights that are being multiplied by each pixel, after applying the convolution. In this case, the convolution reduces the size of the image from 28^2 to 26^2, which is 676. Thus there are 676 weights for this model. There are no concerns for overfitting, and the concerns for convergence are the same as explained in the previous models.

3. ...

Answer: The same ones as for Method 0 and 1, but now the hyperparameter C that minimized the avg validation BCE is $C = 0.316$. All the other answers coincide with the answers for model 0 and 1.

2 Results for Sneaker-Sandal

Please see table 1 on next page.



	Model 0		Model 1			Model 2
	A	B	C	D	E	F
	Baseline Model: Logistic Regr	Logistic Regr + Avg Luminosity per Pixel	Logistic Regr + Avg Number of "on" Pixels per Image	Logistic Regr + Number of "on" pixels per Image	Logistic Regr + Vertical Edge Detection Convolution	Logistic Regr + Horizontal Edge Detection Convolution
Avg CV Training Error Rate	0.017	0.017	0.010	0.017	0.018	0.019
Avg CV Validation Error Rate	0.056	0.055	**0.048**	**0.041**	0.5	0.054
Avg CV Training BCE	0.070	0.070	0.048	0.071	0.074	0.080
Avg CV Validation BCE	0.442	0.445	0.484	**0.227**	0.457	**0.380**
Best C	0.316	0.316	1.000	0.316	0.100	0.316
Best Model Training Error Rate	0.030	0.030	0.025	0.026	0.033	0.034
Best Model Training BCE	0.130	0.130	0.106	0.111	0.137	0.140
Best Model Test Error Rate	**0.034**	**0.034**	**0.033**	0.034	0.038	0.046
Best Model Test BCE	0.145	0.145	**0.138**	**0.143**	0.148	0.169

Table 1: Accuracy metrics for multiple models I tried for this project. The ones highlighted in green are the ones I chose to include in the project as method 0, 1 and 2.

Before entering the details, I will briefly explain what the rows from table 1 represent.

- Avg CV Training Error Rate: The average training error rate across the 5 folds.
- Avg CV Validation Error Rate: The average validation error rate across the 5 folds.
- Avg CV Training BCE: The average binary cross entropy across the 5 folds.

- Avg CV Validation BCE: The average validation binary cross entropy across the 5 folds. I used this measure to select the best C hyper-parameter in logistic regression.
- Best C: the hyper-parameter C that minimizes the Avg CV Validation BCE in logistic regression.
- Best Model Training Error Rate: The training error rate of the model with the best C retrained on A (the 90% of the original training data).
- Best Model Training BCE: The training binary cross entropy of the model with the best C retrained on A (the 90% of the original training data).
- Best Model Test Error Rate: The test error rate of the model with the best C retrained on A (the 90% of the training data) and tested on B (the 10% of the original training data).
- Best Model Test BCE: The test binary cross entropy of the model with the best C retrained on A (the 90% of the original training data) and tested on B (the 10% of the original training data).

In table 1, the numbers in **bold** are the best two results for a particular row. I did this only for the rows associated to the validation and testing process since those are the ones that determine what model is best.

I tried many options for this project and the reasoning behind all these trials is the following. After training and testing a plain logistic regression model (model 0), I followed the instructions and I found it intuitive to add a feature to my training data that encodes the number of pixels that are "on". This corresponds to model D. The former model, improved the average CV error rate (average taken across the 5 folds) with respect to the baseline model, as well as the average CV validation BCE and the test BCE. Then, instead of just counting the pixels that are "on", I averaged their number by dividing the number of pixels "on" by the total number of pixels in the image. This is model C. This one improved the Test error rate and also the test BCE. I also tried other feature transforms which are averaging the luminosity of an image (model B), i.e. summing up the values of all the pixels and dividing by the total number of the pixels of an image. This model didn't show improvement with respect to the baseline. Finally, I played around with a couple of convolutions + logistic regression (models E and F). E was not improving any of the metrics in the table, and the model F will be explained in more detail in the results.

2.1 Hyperparameter Selection

For each model, I performed 5-fold CV. In this process, I calculated the training and validation error, and the training and validation BCE for each fold. In figure 1, these errors are averaged across all 5 folds and displayed with a single curve for each kind of error. For each model, I plotted the error (training error rate, validation error rate, training BCE and validation BCE) against $log_{10}(C)$. In this way, the plots are more pretty than if I would have plotted against C directly.

In terms of fitting the model, I can observe that when C increases the training and validation errors decrease until a certain value of C, where the training and testing error split ways since the validation error starts to increase. That's when the model starts to overfit. That value of C where this phenomena happens, is the C that minimizes the error, both in the case of the validation error and the validation BCE. In my case, since it is a binary classification, I chose the C that minimizes the validation BCE for each model 0, 1, and 2. Those optimal values of C are listed in table 1 on the "Best C" row.

2.2 ROC Curves

Since it is hard to determine visually which one of the curves is higher, I zoomed in differently for the ROC on Training than the ROC on Heldout. In both cases, the red line, which corresponds to Model 1, is most of the time above the other curves. This is also proven when I compute the AUC

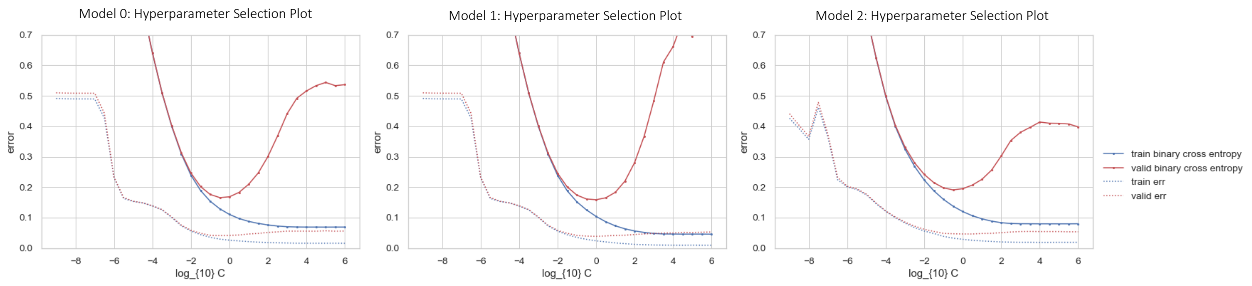


Figure 1: Hyperparameter selection plots for all three chosen methods, displaying training and validation error rates and BCEs.

	AUC on Training Set	AUC on Heldout Set
Model 0	0.995	0.993
Model 1	0.997	0.993
Model 2	0.996	0.991

Figure 3: AUC values for all three models.

(area under the roc curve) because model 1 has the higher AUC for both the training and heldout set among the three selected models.

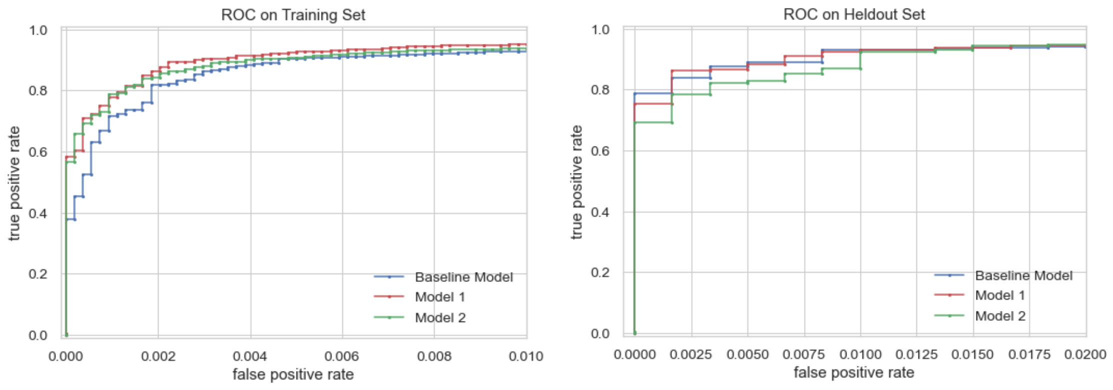


Figure 2: ROC Curves on Trianing and Heldout sets for all three chosen methods.

2.3 FN and FP Examples

Figure 4 shows examples of sandals and sneakers that were misclassified by model 1 and 2. The false negatives are misclassified sandals. I assumed that if a picture had more pixels that are on, the picture would represent sneakers. However, the false negatives I found, seem to be as dense with on pixels as an image with sneakers would be. Also, the false positives, which are the misclassified sneakers, may as well have less pixels that are ”on” that some sandals.

In the case of model 2’s false negatives and false positives, it seems that the horizontal edge detection is not as useful. It might seem from these examples that some of the sandals can even be confused with sneakers even for the human eye. Thus, this transformation does not seem to be very appropriate since there are sandals that have an associated image that resemble a sneaker, and the same happens vice-versa.

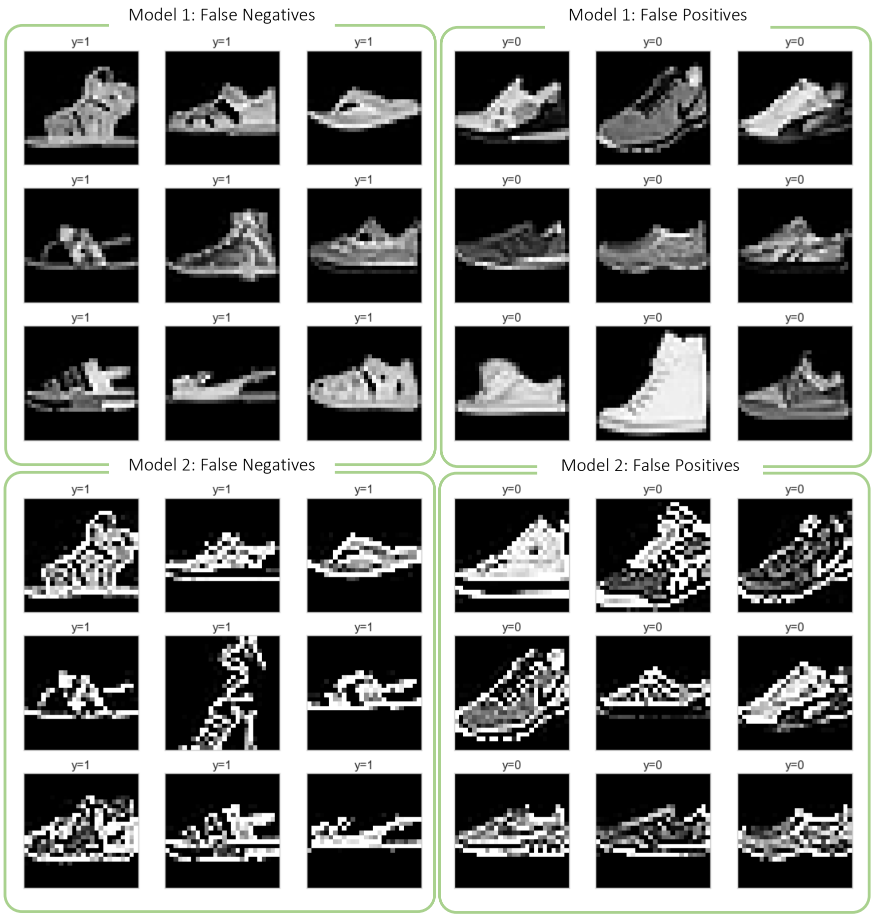


Figure 4: Examples of false negatives and false positives for models 1 and 2.