ISYE 6740 Homework 5

Nick DiNapoli, ndinapoli6@gatech.edu

Due: November 5, 2021

1 Conceptual question

1.1 Mutual information

Consider the information regarding two keywords, "prize" and "hello", in spam versus non-spam emails in the following two tables:

	"prize" = 1	"prize" = 0
"spam" = 1	150	10
"spam" = 0	1000	15000

	"hello" = 1	"hello" = 0
"spam" = 1	155	5
"spam" = 0	14000	2000

First, I calculate the mutual information for each the two keywords using the dicretized formulation,

$$I(U;C) = \sum_{t\in\{0,1\}} \sum_{c\in\{0,1\}} P(U=t, C=c) \quad \log_2 \frac{P(U=t, C=c)}{P(U=t)P(C=c)} \tag{1}$$

where I is the mutual information, U is the keyword and t is the boolean indication of the keyword being present ($t = 1$) or not present ($t = 0$) in the email, and C is the class and c is the boolean indication of the email class being spam ($c = 1$) or not spam ($c = 0$). Equation 1 simplifies to,

$$I(U;C) = \frac{N_{11}}{N}\log_2\frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N}\log_2\frac{NN_{01}}{N_{0.}N_{.1}} + \frac{N_{10}}{N}\log_2\frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N}\log_2\frac{NN_{00}}{N_{0.}N_{.0}} \tag{2}$$

after using the fact that the operand of the log function has a $1/N$ term in the numerator and two $1/N$ terms being multiplied together in the denominator, cancelling and resulting in a N term in the numerator. In Equation 2, the probabilities can be calculated by simple fractions using the information in the two tables above. Here, $N_{t,c}$ represents the number of number of emails with (or without) keyword t that are of class c. Recall from the previous paragraph the meaning of t,c = {0, 1}. Also note that "." notation refers to "all" in the sense that $N_{.c}$ is all emails of class c and $N_{t.}$ is all emails with keyword t.

Now, using Equation 2 and the above table, I calculate the mutual information of the keyword "prize",

$$\begin{aligned} I(U;C) &= \frac{150}{16,160}\log_2\frac{(16,160)(150)}{(150+1,000)(150+10)} + \frac{10}{16,160}\log_2\frac{(16,160)(10)}{(10+15,000)(150+10)} \\ &+ \frac{1,000}{16,160}\log_2\frac{(16,160)(1,000)}{(150+1,000)(1,000+15,000)} + \frac{15,000}{16,160}\log_2\frac{(16,160)(15,000)}{(10+15,000)(1,000+15,000)} \\ &\approx 0.032960 \end{aligned} \tag{3}$$

and the mutual information of the keyword “hello”,

$$
\begin{aligned}
I(U;C) &= \frac{155}{16,160}\log_2\frac{(16,160)(155)}{(155+14,000)(155+5)} + \frac{5}{16,160}\log_2\frac{(16,160)(5)}{(5+2,000)(155+5)} \\
&+ \frac{14,000}{16,160}\log_2\frac{(16,160)(14,000)}{(155+14,000)(14,000+2,000)} + \frac{2,000}{16,160}\log_2\frac{(16,160)(2,000)}{(5+2,000)(14,000+2,000)} \\
&\approx 0.000784
\end{aligned} \tag{4}
$$

It is clear from the above results in Equations 4 & 5 that the mutual information for the keyword, “prize” is significantly higher and therefore is more informative for deciding if an email is spam or not.

1.2 Change detection

I consider two distributions, $f_0 = \mathcal{N}(0,1)$ and $f_1 = \mathcal{N}(3,1)$, and aim to detect a mean shift of minimum size 3. The CUSUM (detection) statistic is,

$$
W_t = \max\left(0, W_{t-1} + \log\frac{f_1(x_t)}{f_0(x_t)}\right), \quad W_0 = 0 \tag{5}
$$

and for this problem can be simplified,

$$
\begin{aligned}
W_t &= \max\left(0, W_{t-1} + \log\frac{1/\sqrt{2\pi}e^{-(x_t-3)^2/2}}{1/\sqrt{2\pi}e^{-(x_t)^2/2}}\right) \\
&= \max\left(0, W_{t-1} + \log(e^{-(x_t-3)^2/2}) - \log(e^{-(x_t)^2/2})\right) \\
&= \max\left(0, W_{t-1} - \frac{(x_t-3)^2}{2} + \frac{x_t^2}{2}\right) \\
&= \max\left(0, W_{t-1} + \frac{1}{2}(6x_t - 9)\right) \\
&= \max(0, W_{t-1} + 3x_t - 4.5).
\end{aligned} \tag{6}
$$

Next, I plot the CUSUM statistic from Equation 6 for a sequence of randomly generated samples where the first 100 samples are i.i.d. from distribution f_0 and the second 100 samples are i.i.d. from distribution f_1. Figure 1 shows the CUSUM statistic for this sequence. It is clear that the change is detected essentially immediately after the 101st sample.

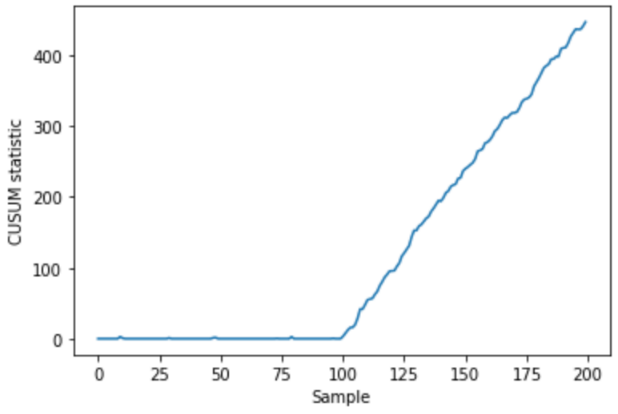


Figure 1: Change detection using CUSUM.

2 House price dataset

I now change directions and analyze the HOUSES dataset, which contains 781 real estate listings in San Luis Obispo county and around it. The dataset contains many attributes but for this regression analysis I regress Price (price of the house in dollars) on: the number of bedrooms, the number of bathrooms, the size (in sq. ft.), the price per square foot, and the status. The status is a categorical variable with options: "Short Sale", "Foreclosure", and "Regular". I use one-hot encoding to represent this categorical feature. As an additional pre-processing step, the independent variables are all scaled before regression.

2.1 Ridge regression

First, I fit a Ridge regression model to predict the Price from the other variables considered above. Completing ridge regression with regularizer parameter α, regularizes the mean-squared-error and effectively shrinks the coefficients of the linear model by penalizing the squared L2 norm of the coefficients. I sweep over values of α from 1-80 in the ridge regressions and use 5-fold cross validation to find the regularizer optimal parameter. As a note, I use negative mean squared error as the scoring for cross validation. Similarly, Figure 2 shows the sum-of-squares residuals (SSR) for each value of α. I aim to choose the optimal model (α) that minimizes the SSR.

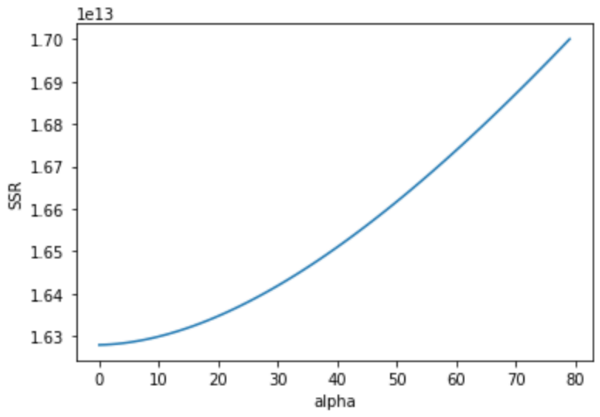


Figure 2: The sum-of-squares residuals for each model by varying α from 1-80.

Table 1 includes a summary of the parameters for the optimal model (α). When completing this computationally, I used sklearn's RidgeCV() and GridSearchCV() + Ridge() to verify this task and I shuffle the data and seed the models using seed = 2. Also, when sweeping α from 1-80, I now used a step size of .1 to obtain an even more accurate optimal α. The optimal α is 2 when using integer sweeping and 2.2 when using a step size of .1. Lastly, the SSR for the optimal α is 1.62797 x 10^13.

α	Bedrooms	Bathrooms	Size	Price/SQ.Ft	Short Sale	Foreclosure	Regular
2.2	-9130.40	13992.16	204224.98	210717.12	-5939.69	-780.51	9246.76

Table 1: Optimal α value and variable coefficients for ridge regression.

2.2 Lasso regression

Now, I complete a similar task but using Lasso regression as a feature selection method where the mean squared error is regularized by penalizing the L1 norm of the coefficients. This control model complexity and encourages sparsity as an increasing α means less parameters will be selected. As in the previous section, 5-fold cross validation is employed when varying α now from 1-3,000 but a seeding of 3 is now used for data shuffling and model fitting. Because of the limitation of sklearn functions used in this problem, Lasso() and GridSearchCV(), I now visualize the CV curve by plotting the mean negative mean squared error during 5-fold cross validation for each value of α. Now I can visually see the optimal α. Figure 3 shows a visual inspection of the α that minimizes the mean-squared-error (MSE). Table 2 is a summary of the coefficients for the optimal α and model. The optimal α is 564 for this lasso regression problem.

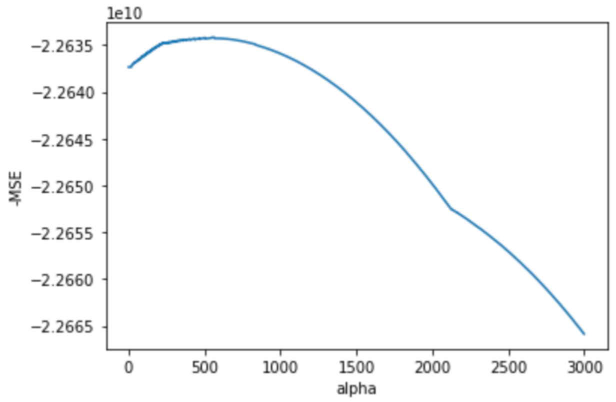


Figure 3: The mean negative MSE for each model by varying α from 1-3,000.

Table 2 is a summary of the coefficients for the optimal α and model. As it can be seen, over this search for an optimal α, the choice for best estimator is able to rid of the Foreclosure variable and hence its impact entirely on the prediction of house price. I now have a simplified model compared to considering all variables.

α	Bedrooms	Bathrooms	Size	Price/SQ.Ft	Short Sale	Foreclosure	Regular
564	-7854.71	12231.61	204824.53	211237.94	-4556.27	0.	9522.35

Table 2: Optimal α value and variable coefficients for lasso regression.

As another feature selection tool, it can be informative to visualize the solution paths for each of the variables in the lasso regression model. The solution path shows at what values of α do different predictors' coefficients reach 0 and hence not selected for the purpose of model simplification. Figure 4 & 5 show the solution paths for the predictors in this problem for α ranging from 1-10,000 and 1-100,000 respectively. Note that the x-axis is log scale. The reason I include Figure 5 is to show that one may need to expand their search over α to see when the variable coefficients reach 0. From these figures, it is clear that the quantitative analysis above is verified as it can be seen that the variable Foreclosure is certainly the first to reach a coefficient of 0. If the search over α in this analysis were expanded, I would see that the variables Bedrooms and Short Sale would be the next variable or features to be selected out of the model. After this, Bathrooms and Regular and finally the two variables that involve square-footage. This makes intuitive sense because the variables to be selected, or last to be left out in this case, are the ones that influence the model predictions (specifically the loss function) the most. Clearly, and intuitively, the size of a house and price per square foot, two extremely common and influential factors in home buying, are deemed most important.

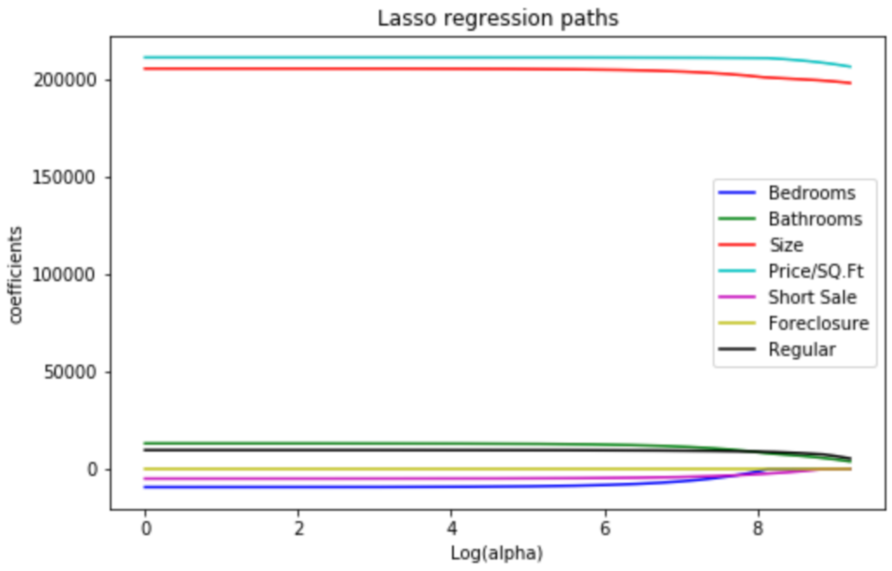


Figure 4: The solution paths of each variable when varying α from 1-10,000.

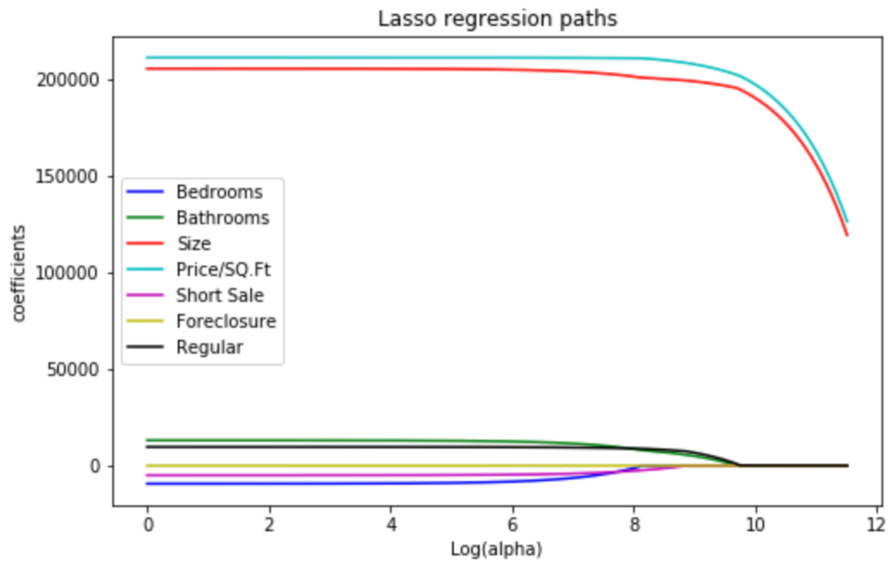
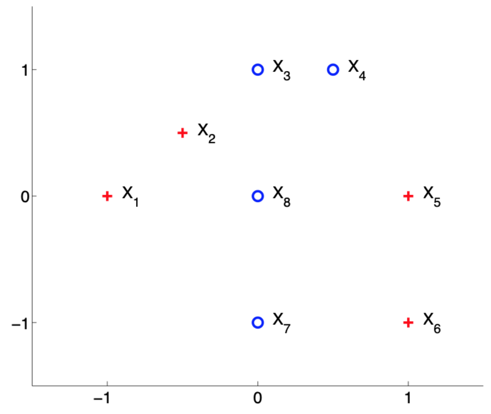


Figure 5: The solution paths of each variable when varying α from 1-100,000.

3 AdaBoost

In this problem, I consider the following dataset (see plot as well), where the first two coordinates are the values of two features of a point and the third is its class/label. Using this data, I run three iterations of AdaBoost with decision stumps by hand and document all of the weights and constants along the way (decision stump/weak learner weights, performance weights, data point weights and normalizing constants).

$$X_1 = (-1, 0, +1), X_2 = (-0.5, 0.5, +1), X_3 = (0, 1, -1), X_4 = (0.5, 1, -1),$$
$$X_5 = (1, 0, +1), X_6 = (1, -1, +1), X_7 = (0, -1, -1), X_8 = (0, 0, -1).$$



3.1 Calculations for three iterations

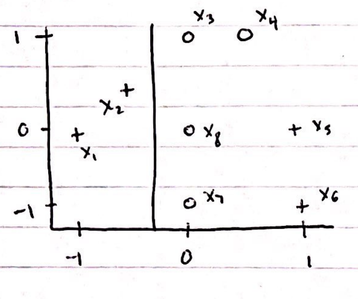
The following images show all work and calculations. Table 3 is a full summary of all calculated weights.

Table 3: Values of AdaBoost parameters at each timestep.

t	ϵ_t	α_t	Z_t	$D_t(1)$	$D_t(2)$	$D_t(3)$	$D_t(4)$	$D_t(5)$	$D_t(6)$	$D_t(7)$	$D_t(8)$
1	0.2500	0.5493	0.8660	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250
2	0.1766	0.7698	0.7456	0.0833	0.0833	0.0833	0.0833	0.2500	0.2500	0.0833	0.0833
3	0.1034	1.0800	0.6092	0.2412	0.2412	0.0517	0.0517	0.1553	0.1553	0.0517	0.0517

I chose the first decision stump b/c it missclassifies the least (or equal to) amount of points leaving the missclassified points the closest (or equal to) from the stump. All weights equal at first, first stump at $x_1 = -.25$

a) $t=1$ $D_1(1) = D_1(2) = D_1(3) = D_1(4) = D_1(5) = D_1(6) = D_1(7) = D_1(8) = 1/8$
$= .125$



$$\varepsilon_1 = \sum_{i=1}^{m} D_1(i)\, \mathbb{I}\{y^i \neq h_1(x^i)\}$$

h_1 missclassified x_5 + x_6

$$\varepsilon_1 = D_1(5) + D_1(6)$$
$$= 1/8 + 1/8$$
$$= 1/4$$
$$= .25$$

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right)$$
$$= 1/2 \ln(3)$$
$$= 0.5493$$

Recall $D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{-\alpha_t y^i h_t(x^i)} = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{if } y^i = h_t(x^i) \\ e^{\alpha_t}, & \text{else} \end{cases}$

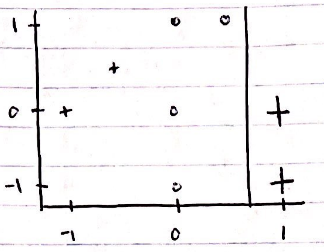
$$Z_t = \sum_{i=1}^{m} D_t(i) e^{-\alpha_t y^i h_t(x^i)}$$

$$Z_1 = 1/8\, e^{-1/2\ln(3)}\,\underset{\uparrow\text{ correct points of equal weight}}{(6)} + 1/8\, e^{1/2\ln(3)}\,\underset{\uparrow\ \#\text{ incorrect } x_5 + x_6}{(2)}$$

$= 0.8660$

$$D_2(1) = D_2(2) = D_2(3) = D_2(4) = D_2(7) = D_2(8) = \frac{.125}{0.8660} e^{-1/2\ln(3)}$$
$= 0.0833$
(b/c they have equal weights in 1st iter)

$$D_2(5) = D_2(6) = \frac{.125}{0.8660} e^{1/2\ln(3)}$$
$= 0.2500$



- some points (weights) shrink & some grow
- new stump at $X_1 = 0.75$ (left neg, righ pos)

$\varepsilon_2 = D_2(1) + D_2(2)$ just X_1 & X_2 missclassified

$= 2(.0883)$

$= 0.1766$

$\alpha_2 = 1/2 \ln\left(\frac{1-0.1766}{0.1766}\right)$

$= 0.7698$

$Z_2 = .0833e^{.7698} + .0833e^{.7698} + .0833e^{-.7698} + .0833e^{-.7698} + .25e^{-.7698} + .25e^{-.7698} + .0833e^{-.7698} + .0833e^{-.7698}$

$= 0.7456$ Using Python

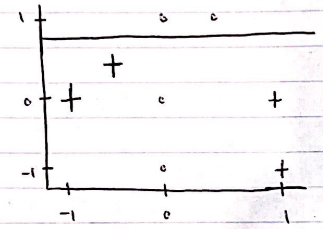
$D_3(i) = \frac{D_2(i)}{Z_2}\begin{cases} e^{-\alpha_2}, & \text{if } y^i = h_2(x^i) \\ e^{\alpha_2}, & \text{else} \end{cases}$

Using Python arrays to compute
- have $D_2(i)$, Z_2, α_2 & y^i's & $h_2(x^i)$'s

$D_3(1) = D_3(2) = 0.2412$

$D_3(3) = D_3(4) = D_3(7) = D_3(8) = .0517$

$D_3(5) = D_3(6) = .1553$



- size to show weights, $D_3(i)$
- choose stump now that classifies high-weight points right ($x_5 \circ x_6$ then $x_1 \circ x_2$)
- stump at $x_2 = 0.75$ (bottom positive)

$$\varepsilon_3 = D_3(7) + D_3(8)$$
$$= 2(0.0517)$$
$$= .1034$$

$$\alpha_3 = \tfrac{1}{2} \ln \left(\frac{1 - .1034}{.1034} \right)$$
$$= 1.0800$$

$$Z_3 = .2412 e^{-1.08} + .2412 e^{-1.08} + .0517 e^{-1.08} + .0517 e^{-1.08} + .1553 e^{-1.08} + .1553 e^{-1.08} + .0517 e^{1.08} + .0517 e^{1.08}$$
$$= 0.6092$$

3.2 AdaBoost performance

The AdaBoost algorithm yields a final classifier,

$$H(x) = \text{sign}\Big(\sum_{t=1}^{T} \alpha_t h_t(x) \Big). \tag{7}$$

Using the weights calculated in the previous section, I use a quick Python script to calculate the training error of AdaBoost. Turns out, there is no training error whatsoever, the algorithm in just three iterations yields a classifier that can classify all of the points correctly. It is clear to see that this AdaBoost classifier will outperform any decision stump on this particular dataset because the data is not linearly separable by any stump on either of the two feature dimensions. Therefore, a single decision stump will always get at least one, in fact two, points incorrectly in the training set. This shows the significant advantage of AdaBoost after just a few iterations of simple calculations.

References

[1] https://scikit-learn.org/stable/auto_examples/linear_model/plot_lasso_coordinate_descent_path.html#sphx-glr-auto-examples-linear-model-plot-lasso-coordinate-descent-path-py