CS6140: Machine Learning

Homework Assignment # 4

Assigned: 03/21/2020 *Due: 03/31/2020, 11:59pm, through Blackboard*

Piyush Goel
Email: goel.pi@northeastern.edu

Problem 1. (20 points) Consider a logistic regression problem with its initial solution obtained through the OLS regression; i.e., $\mathbf{w}^{(0)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, in the context of the code provided in class (Week 6). Recall that $\mathbf{x}$ was drawn from a mixture of two Gaussian distributions with $\dim\{\mathbf{x}\} = 2$ (before adding a column of ones) and that $y \in \{0, 1\}$. You probably noticed that the initial separation line is consistently closer to the data points of class 0.

a) (10 points) Why was this the case? Draw a picture (if possible) to support your argument.

b) (5 points) Devise a better initial solution by modifying the standard formula $\mathbf{w}^{(0)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.

c) (5 points) Now again consider the case where $y \in \{-1, +1\}$. What is the form of the modified solution from part (b) in this case?

Solution

a) The formula we use to initialize the weights is the same as the solution of OLS Linear Regression. If we consider this as a linear regression problem instead of a logistic regression one then we have two features (which is the same as in the case of logistic regression), and one output (but the output here is continuous variable, albeit the correct values are binary). So linear regression would look for a 3-D plane with the 3 axes as X_1, X_2, Y.
For all the points of class 0 the intersection of the planes $Y = 0$ and the plane which is the solution of linear regression would be a line closer to the data points of class 0, since that is how linear regression is supposed to work (i.e. predict a value as close to zero as possible for all the points with $y = 0$). Similarly there would be a line closer to the points of class 1 in the plane $Y = 1$ as well.
Now, for logistic regression we're only working in two dimensions with axes X_1, X_2 or simply the plane $Y = 0$, if we try to relate to the linear regression scenario. Hence we only see the intersection of the plane $Y = 0$ and the plane which is the solution of OLS regression, which is line close to the points of class 0.

b) Firstly, for simplicity lets assume the number of points of both the classes is roughly the same. For a better initial solution we somehow need to find the line which is the intersection of the OLS regression solution plane and the plane $Y = 0.5$ (instead of $Y = 0$), since that line should equally be closer to the points of both classes 0 and 1. Or instead of finding the line that we can just shift the whole solution plane down along the Y-axis by 0.5, which can simply be done by adding -0.5 to w_0. Therefore a better starting solution would be $\mathbf{w}^{(0)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} + \begin{bmatrix} -0.5 \\ 0 \\ 0 \end{bmatrix}$.
This was assuming that both the classes have roughly the same number of points. Otherwise instead

of the plane $Y = 0.5$ a better choice would be $Y = \frac{n_1}{n_0+n_1}$, where n_0 and n_1 are the number of points belonging to the class 0 and 1 respectively. Therefore a better starting solution would be $\mathbf{w}^{(0)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} + \begin{bmatrix} -\frac{n_1}{n_0+n_1} \\ 0 \\ 0 \end{bmatrix}$. To help imagine this let $n_0 = 1000$ and $n_1 = 10$, in this case the solution plane would relatively have a small angle of intersection with the plane $Y = 0$ as compared to the case when $n_0 \approx n_1$, hence we only need to shift the plane by a small amount to get a better initialization and shifting by 0.5 might make the line move too far. This might not still be the best initialization, but is certainly better than the simple OLS regression one.

c) In this case we don't need to modify our solution since the line which is at the intersection of $Y = 0$ and the solution plane would be already be equally close to points of both classes +1 and -1, since 0 is in the middle of +1 and -1. Hence the solution remains $\mathbf{w}^{(0)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.
Again, similar to previous part for the case when n_0 and n_1 have large difference a better solution would be $\mathbf{w}^{(0)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} + \begin{bmatrix} 0.5 - \frac{n_1}{n_0+n_1} \\ 0 \\ 0 \end{bmatrix}$.

Problem 2. (15 points) Let A, B, C, and D be binary input variables (features). Give decision trees to represent the following Boolean functions:

a) (3 points) $A \wedge \bar{B}$

b) (3 points) $A \vee (B \wedge \bar{C})$

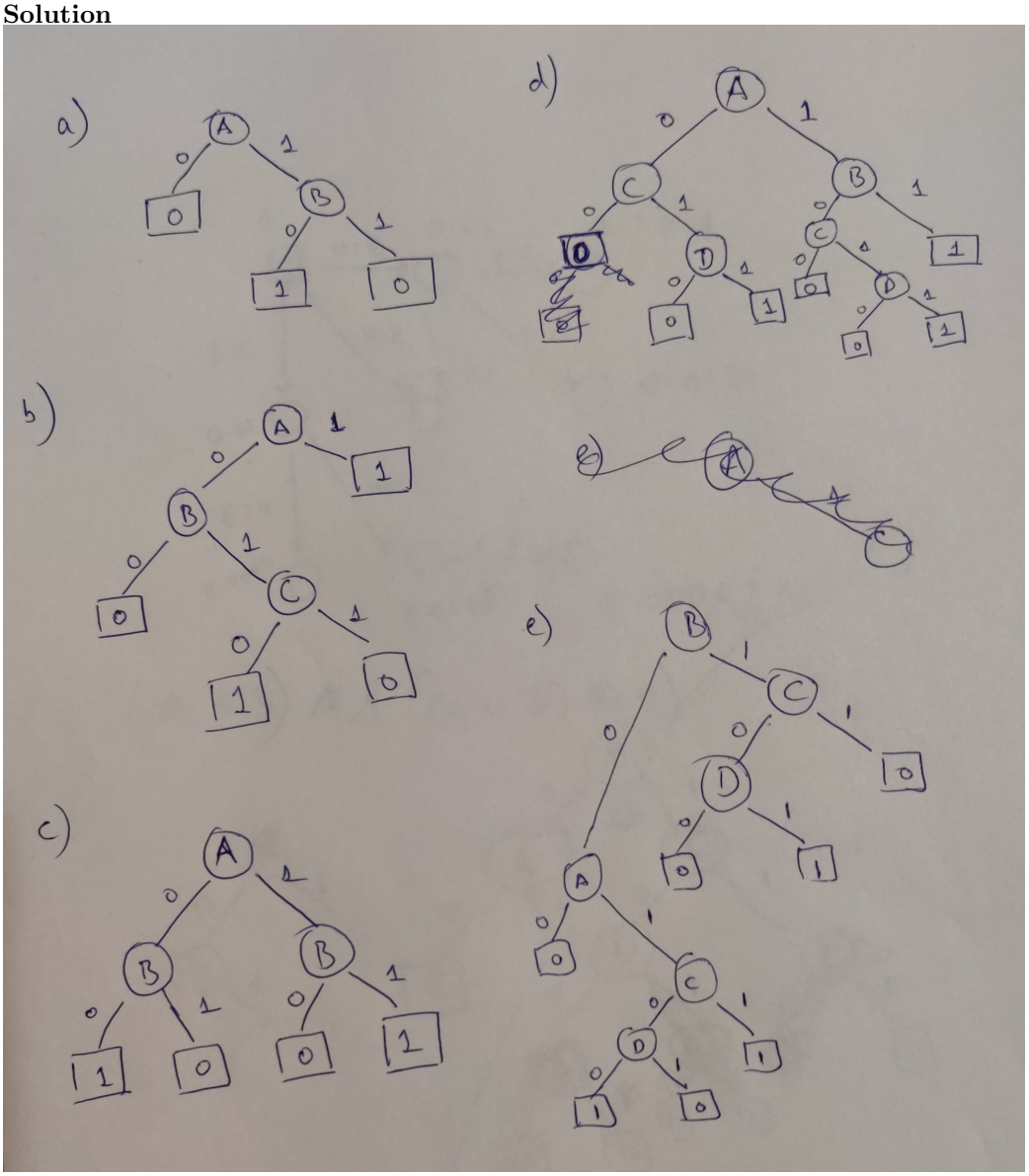
c) (3 points) $\bar{A} \oplus B$

d) (3 points) $(A \wedge B) \vee (C \wedge D)$

e) (3 points) $(A \vee B) \wedge ((C \vee \bar{D}) \oplus B)$

where $\bar{A}$ is the negation of A and $\oplus$ is an exclusive OR operation.

Solution



Problem 3. (10 points) Consider the data set from the following table:

	Sky	Temperature	Humidity	Wind	Water	Forecast	Enjoy Sport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	Hign	Strong	Cool	Change	Yes

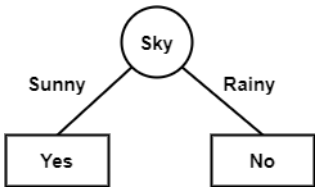
a) (5 points) Using Enjoy Sport as the target, show the decision tree that would be learned if the splitting criterion was information gain.

b) (5 points) Add the training example from the table below and compute the new decision tree. This time, show the value of the information gain for each candidate attribute at each step in growing the tree.

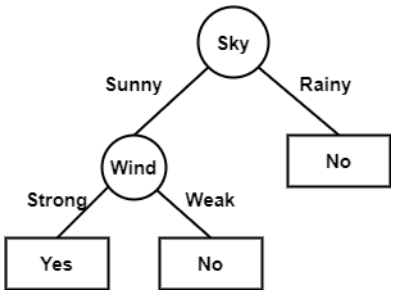
	Sky	Temperature	Humidity	Wind	Water	Forecast	Enjoy Sport
1	Sunny	Warm	Normal	Weak	Warm	Same	No

Solution

a)



b)



Information Gain calculation at step 1:
Entropy of the "Sky" node $= -p(No)\log p(No) - p(Yes)\log p(Yes) = -0.4\log 0.4 - 0.6\log 0.6 = 0.0.970$
Entropy of the "Sunny" side if we split using "Sky" $= -p(No)\log p(No) - p(Yes)\log p(Yes) = -0.25\log 0.25 - 0.75\log 0.75 = 0.811$
Entropy of the "Rainy" side if we split using "Sky" $= -p(No)\log p(No) - p(Yes)\log p(Yes) = -1\log 1 - 0\log 0 = 0$
Information Gain $= 0.970 - \frac{4}{5}0.811 - \frac{1}{5}0 = 0.322$

Information Gain calculation at step 2:
Entropy of the "Wind" Node $= -p(No)\log p(No) - p(Yes)\log p(Yes) = -0.25\log 0.25 - 0.75\log 0.75 = 0.811$
Entropy of the "Strong" side if we split using "Wind" $= -p(No)\log p(No) - p(Yes)\log p(Yes) = -0\log 0 - 1\log 1 = 0$
Entropy of the "Weak" side if we split using "Wind" $= -p(No)\log p(No) - p(Yes)\log p(Yes) = -1\log 1 - 0\log 0 = 0$
Information Gain $= 0.811$

Problem 4. (40 points) Principal Component Analysis (PCA) as part of dimensionality reduction. Implement PCA yourself with the only allowed library function being the eigenvalue/eigenvector determination. Download at least 3 data sets from the UCI Machine Learning repository (or elsewhere) and learn a logistic regression model in the following three modes:

a) (5 points) Use the data set as is; i.e., no normalization or any form of preprocessing.

b) (10 points) Normalize the data set using z-score normalization and evaluate the model.

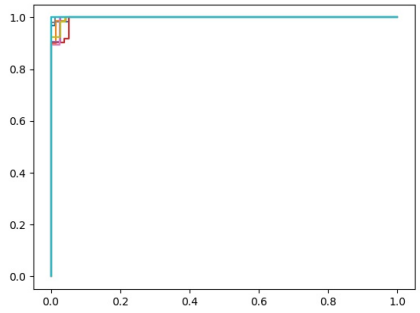
c) (20 points) Normalize the data set to zero mean, apply PCA with 99% of retained variance, and then train and evaluate the model.

d) (5 points) Normalize the data set using z-score normalization, apply PCA with 99% of retained variance, and then train and evaluate the model.

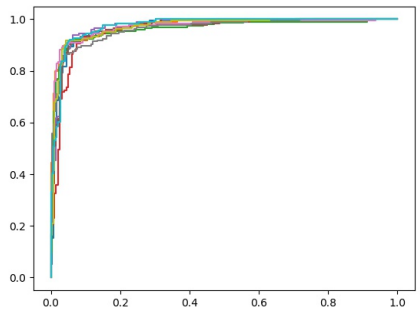
In each case, evaluate the model using 10-fold cross-validation. Make sure that all preprocessing steps are performed on training partition only and then applied to the test partition. Provide area under the ROC curve for each model on each data set. To do that, you can average ROC curves obtained on each partition. Discuss your results and provide appropriate support for your conclusions.

Solution

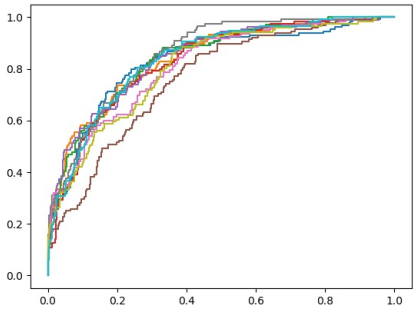
No normalization and pre-processing
Dataset: banknote authentication: Avg AUC = 0.99843



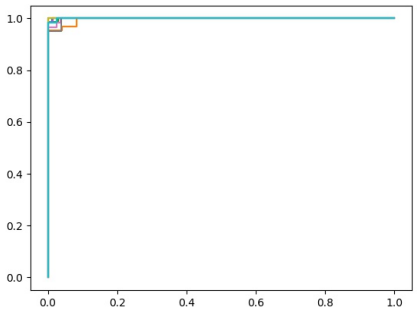
Dataset: spambase: Avg AUC = 0.96855



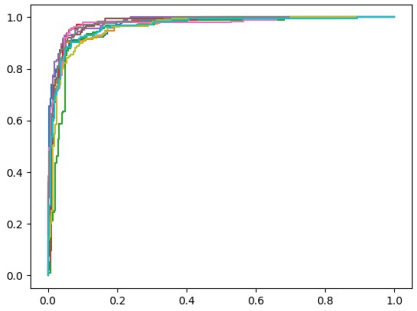
Dataset: adults with over 50k salary: Avg AUC = 0.82571



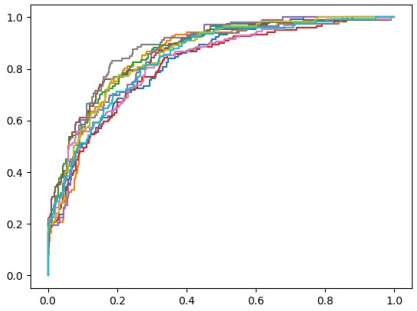
z-score normalization
Dataset: banknote authentication: Avg AUC = 0.99925



Dataset: spambase: Avg AUC = 0.96874

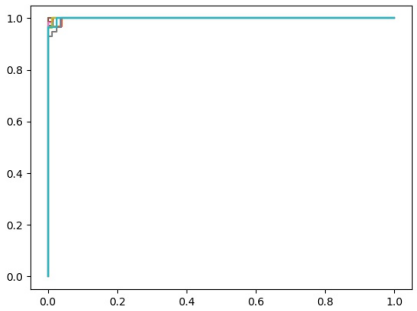


Dataset: adults with over 50k salary: Avg AUC = 0.84754

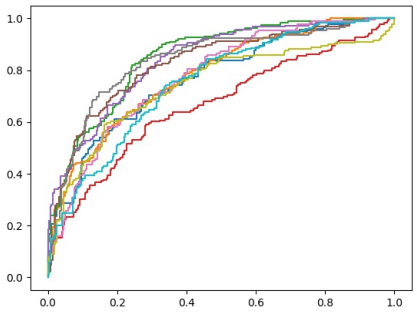


PCA with zero mean normalization

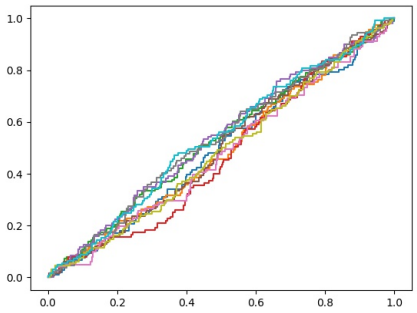
Dataset: banknote authentication: Avg AUC = 0.99933, Number of chosen Columns = 4, Total number of Columns = 4



Dataset: spambase: Avg AUC = 0.78305, Number of chosen Columns = 2, Total number of Columns = 57

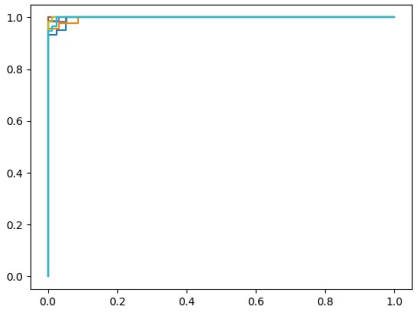


Dataset: adults with over 50k salary: Avg AUC = 0.50349, Number of chosen Columns = 1, Total number of Columns = 14

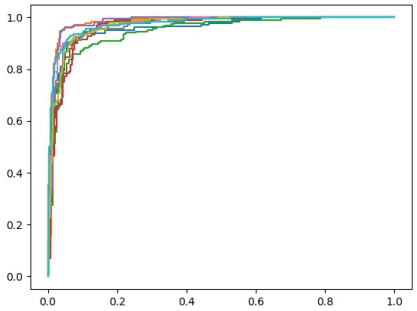


PCA with z-score normalization

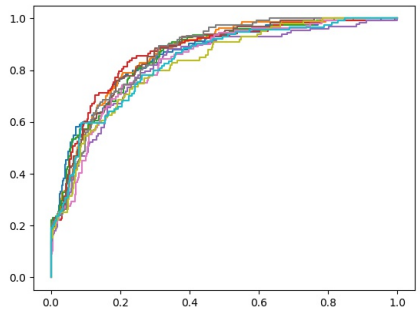
Dataset: banknote authentication: Avg AUC = 0.99913, Number of chosen Columns = 4, Total number of Columns = 4



Dataset: spambase: Avg AUC = 0.96829, Number of chosen Columns = 54, Total number of Columns = 57



Dataset: adults with over 50k salary: Avg AUC = 0.84726, Number of chosen Columns = 14, Total number of Columns = 14



Now, as we can see the average AUC-ROC (which is indicative of accuracy) for almost all of the datasets increase from part a to b then decreases for part c and then again increases for d. Therefore the final order of AUCs is (very) roughly $b > a > d > c$. This can be explained as follows:
$b > a$: because we just normalize the data in part b over part a and this may help the learning process, but not too much, hence the increase is almost insignificant.
$c < a, c < b, d < b, d < c$: because we're doing PCA which removes some amount of information by removing the features with low variance contribution to the data.
$d > c$: Since, during z-score normalization we divide each feature value by the feature's standard-deviation as well and not just subtract the mean, this helps scale down the feature values, effectively lowering the variance of the feature as well. This results in a higher (much higher in some cases) number of features (of better quality) being selected in when PCA is done after z-score normalization instead of zero mean normalization. Hence the better results for part d.

Extra Problem. (25 points) Consider a data set $\mathbf{X} \in \mathbb{R}^{n \times d}$ such that $d \gg n$. Propose how to perform principal component analysis if one must avoid the construction of a large $d \times d$ covariance matrix. You can assume that a $d \times n$ matrix can fit in the memory. This is an all or nothing question. If your result is correct you get 25 points; if not, you get 0 points.

Solution

Throughout this solution it is assumed that the means of the columns of matrix X have been reduced to 0. Therefore, the covariance matrix Σ can be expressed simply as $X^T X$. From the class we know that PCA reduces simply to the eigen problem, that is $V^T \Sigma V = \Lambda$. Here, since V is an orthogonal matrix we can rewrite the expression as $\Sigma = V \Lambda V^T$, where V is a matrix containing eigen vectors as columns and Λ is a diagonal matrix containing eigen values as the diagonal elements.

Using the compact form of SVD the matrix X can be expressed as USV^T, where U is a n x n orthogonal matrix, S is a n x n diagonal matrix, V (this V and the V in the previous paragraph should be considered different, even though they later turn out to be the same) is a d x n orthogonal matrix.

Therefore, Σ can be expressed as follows

$$\Sigma = X^T X = (USV^T)^T (USV^T) = VS^T U^T USV^T = VS^T SV^T$$

Comparing this with the result which we got in the first paragraph ($\Sigma = V \Lambda V^T$), we see that both the V are the same and that $S^T S$ would give us Λ.

This shows that if we do compact SVD of the matrix X the V^T which we get would give us the eigen vectors and the S would give us the eigen values and since none of U, V or S are d x d and none of the calculation during SVD would involve using a d x d matrix, we can easily evade having to store a large d x d matrix.