HOMEWORK ASSIGNMENT #3
B555- MACHINE LEARNING

FNU ANIRUDH
(aanirudh@umail.iu.edu)

Solution 1

a)

Split 100000 rows into train=500 and test=300 rows
Accuracy of naive base: 74.6666666667%

ZeroDivisionError: float division by zero

Is observed when run with columns of Ones.

b) I have applied stochastic gradient descent in my code.

$W^{(t+1)} = w(t) + (X^T P^{(t)} (1 - P^{(t)}) X)^{-1} XT (y - p^{(t)})$

Running learner = Logistic Regression
Accuracy for Logistic Regression: 76.6666666667
Running learner = Linear Regression
Accuracy for Linear Regression: 76.0

Accuracy for Logistic Regression: 74.0
Running learner = Linear Regression
Accuracy for Linear Regression: 71.6666666667

As seen from output above, Logistic regression is able to outperform linear regression most of the times (more than 90% of time)

Please refer code for details.

c) Neural Network not Implemented completely, have written learn function.

d)

<u>Naïve Bayes</u>

Mechanism:-

For the given features (x) and the label y, it estimates a joint probability from the training data. Hence this is a Generative model.

Assumption:-

Model assumes all the features are conditionally independent .so, if some of the features are dependent on each other (in case of a large feature space), the prediction might be poor.

Limitations:-

Works well even with less training data, as the estimates are based on the joint density function.

Logistic Regression

Mechanism:-

Estimates the probability(y/x) directly from the training data by minimizing error. Hence this is a Discriminative model.

Assumptions:-

It the splits feature space linearly, it works OK even if some of the variables are correlated.

Limitations:-

With the small training data, model estimates may over fit the data.

Neural Network

Mechanism:-

Neural network are a form of supervised representation learning. All the first hidden layers constitute representation layer; the output learning is the supervised prediction part.

Assumptions:-

We will often learn with stochastic gradient descent assuming one sample.

Limitations:-

Neural network are slow to converge and hard to set parameter.

Sample Outputs: - (Neural Network not implemented completely)

i)
Accuracy for Classifier Logistic Regression: 71.3333333333
Running learner = Neural Network
Accuracy for Neural Network: 46.3333333333
Running learner = Logistic Regression

Accuracy for Logistic Regression: 73.6666666667
Running learner = Linear Regression
Accuracy for Linear Regression: 71.0
Accuracy of naive base: 77.6666666667%

ii)

Accuracy for Classifier Logistic Regression: 72.0
Running learner = Neural Network
Accuracy for Neural Network: 52.0
Running learner = Logistic Regression
Accuracy for Logistic Regression: 75.6666666667
Running learner = Linear Regression
Accuracy for Linear Regression: 75.6666666667
Accuracy of naive base: 73.6666666667%

iii)

Accuracy for Classifier Logistic Regression: 53.0
Running learner = Neural Network
Accuracy for Neural Network: 50.6666666667
Running learner = Logistic Regression
Accuracy for Logistic Regression: 73.0
Running learner = Linear Regression
Accuracy for Linear Regression: 72.3333333333
Accuracy of naive base: 78.0%

iv)

Accuracy for Classifier Logistic Regression: 76.0
Running learner = Neural Network
Accuracy for Neural Network: 56.3333333333
Running learner = Logistic Regression
Accuracy for Logistic Regression: 75.0
Running learner = Linear Regression
Accuracy for Linear Regression: 75.6666666667
Accuracy of naive base: 78.3333333333%

After running the program several times, I can say that accuracy for Naïve bayes is consistent and is always more than 70%. Logistic regression accuracy is more than linear regression most of the times (90% probability) of being more, though I couldn't implement neural network I believe it should outperform all the other algorithms most of the time.

2. Accuracy for Classifier Logistic Regression: 73.6666666667
Running learner = Neural Network
Accuracy for Neural Network: 51.6666666667
Running learner = Logistic Regression

Accuracy for Logistic Regression: 74.0
Running learner = Linear Regression
Accuracy for Linear Regression: 71.6666666667

Accuracy for Classifier Logistic Regression: 70.0
Running learner = Neural Network
Accuracy for Neural Network: 55.0
Running learner = Logistic Regression
Accuracy for Logistic Regression: 70.3333333333
Running learner = Linear Regression
Accuracy for Linear Regression: 72.3333333333

Accuracy for Classifier Logistic Regression: 71.0
Running learner = Neural Network
Accuracy for Neural Network: 49.6666666667
Running learner = Logistic Regression
Accuracy for Logistic Regression: 73.3333333333
Running learner = Linear Regression
Accuracy for Linear Regression: 72.3333333333

Accuracy is either same or less compared to Logistic regression hence we can say that logistic regression and classifier are same for datasets which are highly clustered and there is less chance of linear separator. Accuracy varies highly when positive and negative values are far apart.

(Please refer derivation at the last for this question)

3.

a)

The most common variants in machine learning are L_1 and L_2 regularization, which can be added to learning algorithms that minimize a loss function E(X, Y) by instead minimizing E(X, Y) + α‖w‖, where w is the model's weight vector, ‖·‖ is either the L_1 norm or the squared L_2 norm, and α is a free parameter that needs to be tuned empirically (typically by cross-validation).

L_1 regularization is often preferred because it produces sparse models and thus performs feature selection within the learning algorithm, but since the L_1 norm is not differentiable, it may require changes to learning algorithms, in particular gradient-based learners.

Regularization can be used to fine-tune model complexity using an augmented error function with cross-validation. The data sets used in complex models can produce a levelling-off of validation as complexity of the models increases. Training data sets errors decrease while the validation data set error remains constant. Regularization introduces a second factor which weights the penalty against more complex models with an increasing variance in the data errors. This gives an increasing penalty as model complexity increases.

L1 Regularizer

$W^{(t+1)} = w(t) + (X^T P^{(t)} (1 - P^{(t)}) X)^{-1} XT (y - p^{(t)}) \pm \lambda_1 w$

L2 Regularizer

$W^{(t+1)} = w(t) + (X^T P^{(t)} (1 - P^{(t)}) X)^{-1} XT (y - p^{(t)}) \pm \lambda_2 w$

Running on train=2000 and test=600 samples
Running learner = Logistic Regression2
Accuracy for Logistic Regression2: 59.5
Running learner = Logistic Regression1
Accuracy for Logistic Regression1: 57.8333333333
Running learner = Random
Accuracy for Random: 53.8333333333

L1 Regularizer

On choosing λ= 100 I am able to achieve accuracy of 57.833 %, for all lambda values from 0 to 30, Accuracy remained 50%. On increasing λ value more than 50 we start to see major shift in accuracy.

L2 Regularizer

On choosing λ = 0.05 I am able to achieve accuracy of 59.5 %.

b) $W^{(t+1)} = w(t) + (X^T P^{(t)} (1 - P^{(t)}) X)^{-1} XT (y - p^{(t)}) \pm (\lambda_3 / 2^* m) * w$

By using above regularizer and setting λ= 100, I was able to achieve 59% for third regularizer.

Running on train=2000 and test=600 samples
Running learner = Logistic Regression3
Accuracy for Logistic Regression3: 59.0
Running learner = Logistic Regression2
Accuracy for Logistic Regression2: 59.5
Running learner = Logistic Regression1
Accuracy for Logistic Regression1: 57.8333333333
Running learner = Random
Accuracy for Random: 46.1666666667

c)

- L1 Regularized Logistic Regression requires a sample size that grows logarithmically in the number of irrelevant features
- Rotationally invariant algorithms (i.e. L2 Regularized Logistic Regression) requires a sample size that grows linearly in the number of irrelevant features

We consider a supervised learning task where we are given M training instances $\{(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \ldots, M\}$. Here, each $\mathbf{x}^{(i)} \in \mathbb{R}^N$ is an N-dimensional feature vector, and $y^{(i)} \in \{0, 1\}$ is a class label. Logistic regression models the probability distribution of the class label y given a feature vector $\mathbf{x}$ as follows:

$$p(y = 1|\mathbf{x}; \theta) = \sigma(\theta^\top \mathbf{x}) = \frac{1}{1 + \exp(-\theta^\top \mathbf{x})}. \quad (1)$$

Here $\theta \in \mathbb{R}^N$ are the parameters of the logistic regression model; and $\sigma(\cdot)$ is the sigmoid function, defined by the second equality.

Under the Laplacian prior $p(\theta) = (\beta/2)^N \exp(-\beta\|\theta\|_1)$ (with $\beta > 0$), the *maximum a posteriori* (MAP) estimate of the parameters θ is given by:

$$\min_\theta \sum_{i=1}^{M} -\log p(y^{(i)}|\mathbf{x}^{(i)}; \theta) + \beta\|\theta\|_1. \quad (2)$$

This optimization problem is referred to as L_1 regularized logistic regression. Often, it will be convenient to consider

3. L_1 regularized logistic regression

We are interested in supervised learning problems where there is a very large number n of input features, but where there may be a small subset—say $r \ll n$ of them—that is sufficient to learn the target concept well. We will consider the following implementation of L_1 regularized logistic regression:

1. Split the data S into a training set S_1 consisting of the first $(1 - \gamma)m$ examples, and a hold-out cross

Space constraints preclude a full discussion, but we also note that C can be chosen automatically (as a function of m) so that the same bound as stated above holds, but with the dependence on C removed. Further, by modifying the definition of $p(y|x;\theta)$, it is straightforward to generalize this result to L_1 regularized versions of other models from the generalized linear model family (McCullagh & Nelder, 1989), such as linear least squares regression.

4. Rotational invariance and L_2 regularization

Let $\mathcal{M} = \{M \in \mathbb{R}^{n\times n} | MM^T = M^TM = I, |M| = 1\}$ be the class of rotational matrices.[1] Thus, if $x \in \mathbb{R}^n$ and $M \in \mathcal{M}$, then Mx is x rotated through some angle around the origin.[2]

Given a training set $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^m$, we let $MS = \{(Mx^{(i)}, y^{(i)})\}_{i=1}^m$ denote the training set with all the inputs rotated according to M. Given a learning algorithm L, we let $L[S](x)$ denote the predicted label resulting from using the learning algorithm to train on a dataset S, and using the resulting hypothesis/classifier to make a prediction on x.

Using L1 regularization, logistic regression is extremely insensitive to the presence of irrelevant features.

List of References

1. Pattern Recognition and Machine Learning :- Christopher M. Bishop
2. Machine Learning Notes:- Predrag Radivojac and Martha White
3. Logistic Matlab code by Predrag Radivojac
4. http://machinelearningmastery.com/naive-bayes-classifier-scratch-python/
5. https://en.wikipedia.org/wiki/Artificial_neural_network
6. https://en.wikipedia.org/wiki/Regularization_(mathematics)
7. http://ai.stanford.edu/~ang/papers/icml04-l1l2.pdf