Logistic Classifier

MLSALT 1 - Christopher Tegho

November 1, 2016

1 Data Visualization

A logistic classifier is implemented, a applied to a simple dataset. Performance is then improved using a non linear feature expansion. In the first part, the classifier is linear. The expected performance is not high in this case, because the two classes are not linearly separable. One single line (a linear classifier) cannot separate the data accurately.

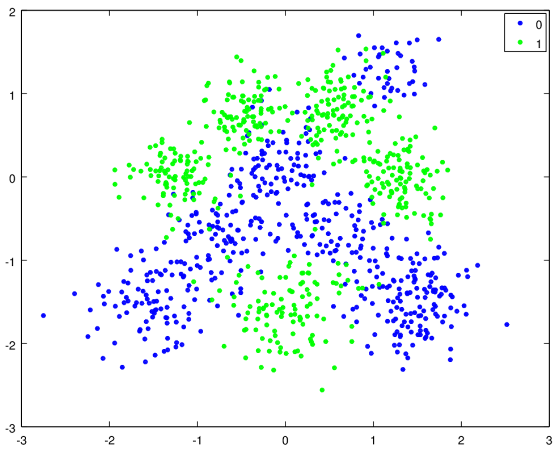


Figure 1: Visualization of the data set

2 Training and Test Data

The data is split into 80% training data and 20% testing data. The indices for the 20% test data are chosen randomly after which we separate the dataset into training and testing. We use the same datasets throughout all sections.

3 Gradient for the Logistic Classifier

The probability of the dataset is a product of Bernouilli distributions,

$$P(y|X,\beta) = \prod_{n=1}^{N} P(y^{(n)}|x^{(n)}) = \prod_{n=1}^{N} \sigma(\beta^T x^{(n)})^{y^{(n)}}(1-\sigma(\beta^T x^{(n)})^{1-y^{(n)}}$$

$$log(P(y|X,\beta)) = L(\beta) = \sum_{n=1}^{N} log(\sigma(\beta^T x^{(n)})^{y^{(n)}}) + \sum_{n=1}^{N} log((1-\sigma(\beta^T x^{(n)})^{1-y^{(n)}})$$

$$L(\beta) = \sum_{n=1}^{N}(y^{(n)}log(\sigma(\beta^T x^{(n)}))) + (1-y^{(n)})log(1-\sigma(\beta^T x^{(n)}))))$$

$$\frac{\delta L(\beta)}{\delta\beta} = \sum_{n=1}^{N} \frac{\delta L(\beta)}{\delta\sigma(x^{(n)})}\frac{\delta\sigma(x^{(n)})}{\delta\beta}$$

$$\frac{\delta L(\beta)}{\delta\sigma(x^{(n)})} = \frac{y(n)}{\sigma(\beta^T x^{(n)})} - \frac{(1-y(n))}{(1-\sigma(\beta^T x^{(n)})} \tag{1}$$

$$\frac{\delta\sigma(x^{(n)})}{\delta\beta} = \frac{x^{(n)}exp(-\beta^T x^{(n)})}{(1+exp(-\beta^T x^{(n)}))^2} = x^{(n)}\frac{\frac{1-\sigma(\beta^T x^{(n)})}{\sigma(\beta^T x^{(n)})}}{\frac{1}{(\sigma(\beta^T x^{(n)}))^2}} = x^{(n)}(1-\sigma(\beta^T x^{(n)}))\sigma(\beta^T x^{(n)}) \tag{2}$$

$$Combining (1) and (2), \frac{\delta L(\beta)}{\delta\beta} = \sum_{n=1}^{N}(y(n) - \sigma(\beta^T x^{(n)})x(n)) \tag{3}$$

4 Implementation of gradient

We use the gradient ascent of the log likelihood to estimate the parameters β with $\beta^{(new)} = \beta^{(old)} + \eta\frac{\delta}{\delta\beta}L(\beta^{(old)})$. In order to determine the learning rate, we estimate the parameters with different rates and plot the loglikelihood for each iteration. The convergence rate for the likelihood with respect to the number of iterations is slower with a small η. The likelihood oscillates with a large learning rate as a result of an overshoot of the ascent. Looking at figure (2), a learning rate of 2 is a good choice resulting in a fast convergence without overshooting.

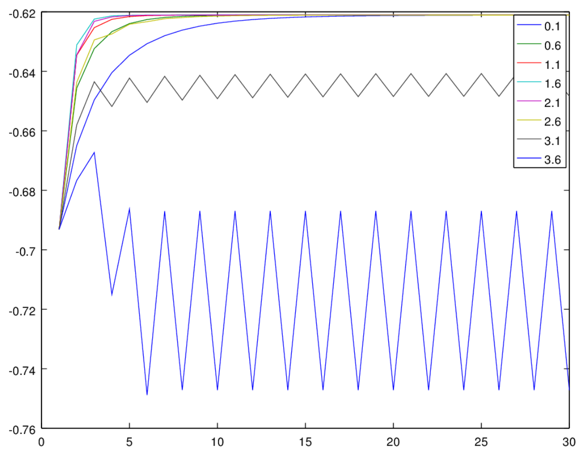


Figure 2: Convergence of the log likelihood for different learning rate. Note the oscillation for a learning rate larger than 2.6.

When implementing the gradient and the computation of the log likelihood, we normalize by dividing by m (number of data points at the input).

Code implementation for estimating β with gradient ascent. There is a regularization term that is added and which is provided in section 10. We use lambda to determine the weight of the regularization. In sections 1 to 9, we use lambda = 0 (no regularization). In section 10, lambda is 1.

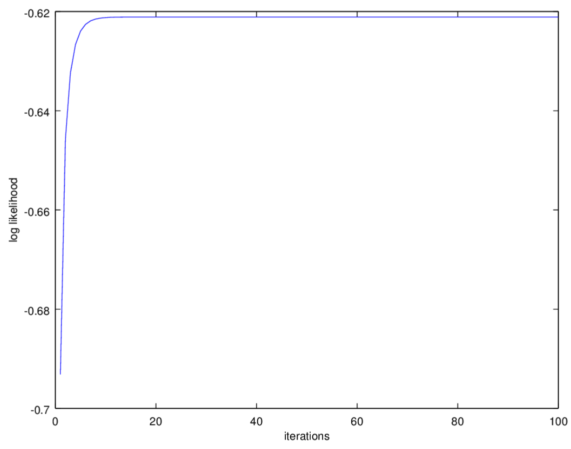
```
function [theta, logLik] = trainLogClass(X, y, numIter, alpha, lambda)
m = size(X, 1); n = size(X, 2);
% Add ones to the X data matrix
X = [ones(m, 1) X];
% Set Initial theta
theta = zeros(n + 1, 1); logLik = zeros(numIter,1); grad = zeros(size(theta));

for k = 1:numIter
  z = X*theta;  h=sigmoid(z);
  logLik(k)= computeLogLik(h, y, z);

  %compute gradient
  grad = (X'*(y-h))/m - lambda*theta/m;
  theta += alpha*grad;
end
```

Code implementation for computing the log likelihood using the fact that $log(\sigma(\beta^T x^{(n)})) \rightarrow \beta^T x^{(n)}$ as $\beta^T x^{(n)} \rightarrow \infty$.

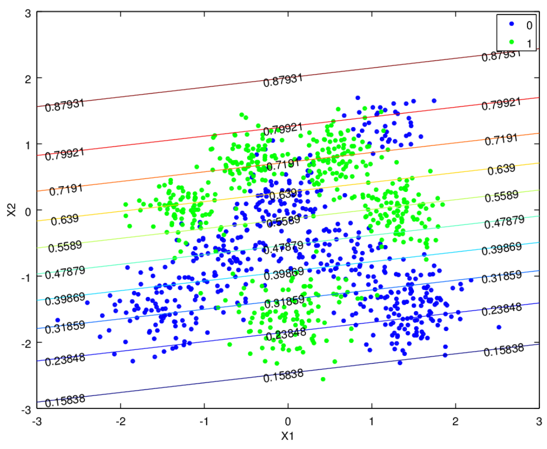
```
function logLik = computeLogLik(h, y, z)
        m = size(y, 1);
          %compute log likelihood
          u = p = log(h); v = q = log(ones(size(y))-h);
        u(p==-inf) = 0; v(q==-inf) = 0;
        u+=z.*(p==inf); v+=-z.*(q==-inf);

    logLik = sum(y.*u+(ones(size(y))-y).*v)/m;
```

5 Training Logistic Classifier

We train the classifier for 100 iterations and plot the likelihood with respect to the iterations. We notice convergence after 17 iterations. The training log likelihood is -0.62 and the testing log likelihood is -0.65. Both log likelihoods are low and are a result of implementing a linear classifier.



(a) Log Likelihood Convergence



(b) Predictions Visualization

Figure 3: (a) The log likelihood converges after 16 iterations for a learning rate of $\eta = 2$. It was not necessary to iterate past 20 iteration in this case. (b) Predictions are visualized with probability contours for the training and test datasets.

6 Confusion Matrix

The final training log likelihood per datapoint is -0.62 and the test log likelihood per datapoint is -0.65. In order to classify the data, a threshold $\tau = .5$ is applied. This threshold would correspond to a straight line class boundary between the 0.48 and the 0.55 lines in Figure 3 (b).

		predicted label	
		0	1
true label	0	fraction: .71, tn = 77	fraction: .29, fp = 31
	1	fraction: .41, fn = 38	fraction: .59, tp = 54

Table 1: Confusion Matrix with a threshold $\tau = .5$ Note that the test dataset length is 200.

7 ROC Curve

Figure 4 displays the ROC curve for the predictions with a threshold that is swept from $\tau = 0$ to $\tau = 1$.The area under this ROC curve is lower than .5, an area lower than the one for a ROC curve of a chance classifier. This is because for thresholds between about .4 and .6, there is a large number of false positives (blue dots above mid threshold lines in figure 3 (b)) and a large number of false negatives (green dots below mid threshold lines in figure 3 (b)).

8 RBF Augmented Input

We train the logistic classification model on the feature expanded input for three choices of RBF width l = 0.01, 0.1, 1). Figures 5 displays the convergence of the log likelihood during training for the three choices of the length scale. We notice that the smaller the length scale, the more iterations it takes for the likelihood to converge. The training likelihood is higher with a smaller length scale, suggesting a better fit. However, when looking at the contour of the predictions for l=0.01, we notice the most probable predictions are centered around the training data. The likelihood for a test data point that falls far

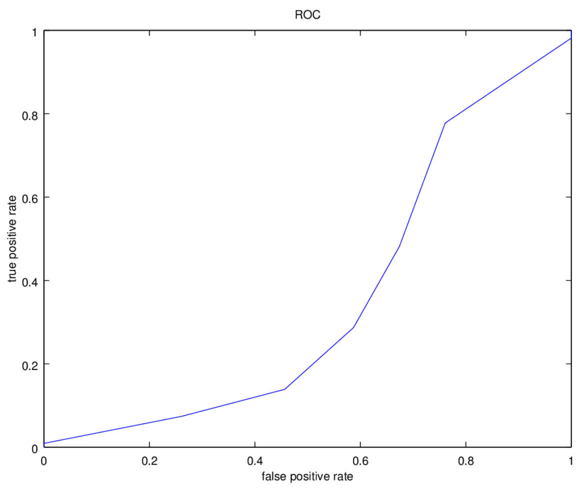
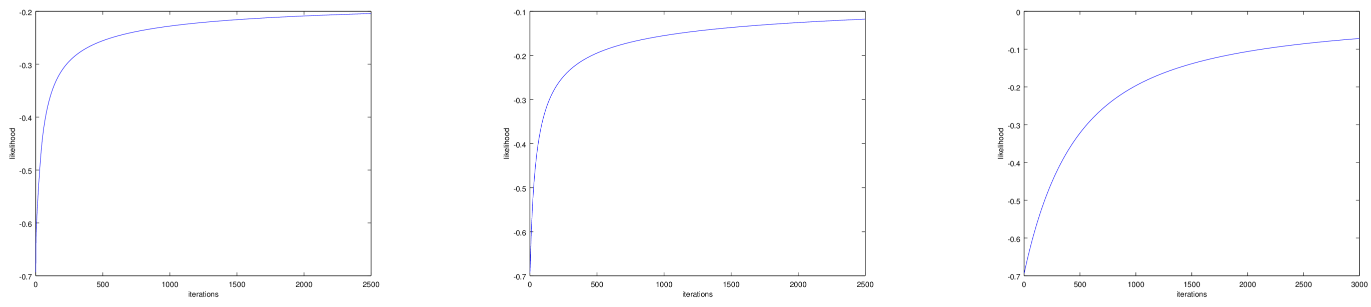


Figure 4: ROC Curve for the linear classifier in parts c to g.

from a training data point vanishes to 0 which can result in misclassification. As reported by the findings in the next section, augmented features with l = 0.01 have led to an overfitting of the training data resulting in a high rate of false negatives and false positives (see section 9) and a low test log likelihood (-0.65) which is similar to the case in the first part (without the augment features input).

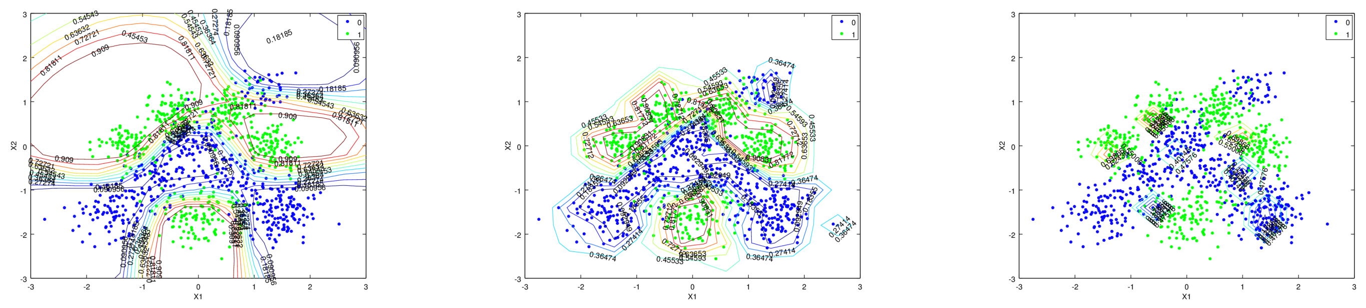


(a) l = 1

(b) l = 0.1

(c) l = 0.01

Figure 5: Convergence of the log likelihood with respect to the number of iterations. (a) l = 1, the log likelihood converges after 2000 iterations with a learning rate of $\eta = .1$. (b) l = 0.1, the log likelihood converges after 2500 iterations with a learning rate of $\eta = 1.2$. (c) For l = 0.01, there is still possibility of achieving higher log likelihood with further iterations, which is consistent with overfitting in this situation. Note that the training log likelihood achieved in this case is the highest (-0.07) with lowest test loglikelihood. Learning rate used is $\eta = 3$



(a) l = 1

(b) l = 0.1

(c) l = 0.01

Figure 6: Contour of the likelihood for the training and testing datasets. Note the overfitting in the case of l = 0.01 (c).

9 Results Obtained With Augmented Features Input

As shown in Table 2, the train likelihood is highest for l=0.01. However, the test likelihood is the lowest for this length scale and this is a result of overfitting. For l=0.01, the classifier is classifying most datapoints to 0, resulting in a high tn and a high fn. The likelihoods obtained are consistent with the results reported in Figure 6 and the ROC curves and confusion matrices below. The area under the ROC curve for l = 1 and l = 0.01 are highest. The ROC curve for l=0.001 is lowest and is very close to the are of the ROC curve of a chance classifier.

l	1	0.1	0.01
Train loglikelihood	-0.20	-0.12	-0.07
Test loglikelihood	-0.29	-0.30	-0.65

Table 2: Train and Test loglikelihood **per datapoint**.

		predicted label	
		0	1
true label	0	fraction: .86, tn = 93	fraction: .14, fp = 15
	1	fraction: .12, fn = 11	fraction: .88, tp = 81

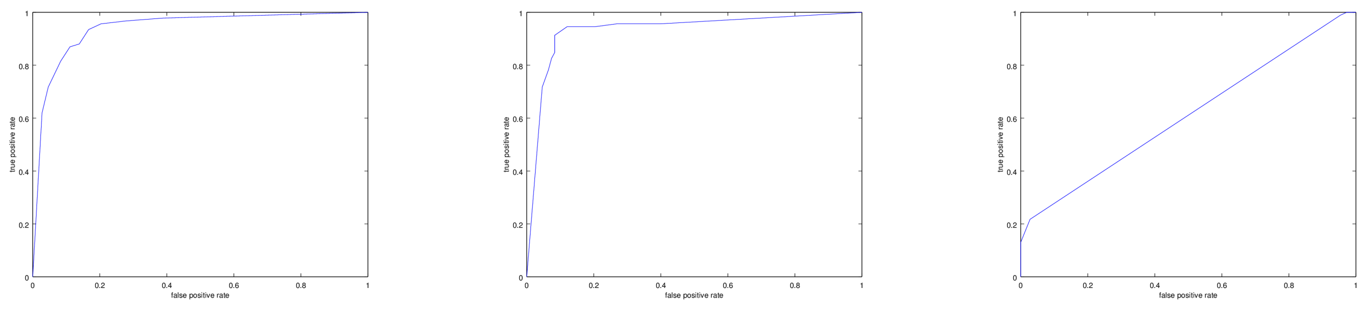
Table 3: Confusion Matrix for l=1, $\lambda = 0$, with a threshold $\tau = .5$. Length of test dataset is 200.

		predicted label	
		0	1
true label	0	fraction: .92, tn = 99	fraction: .08, fp = 9
	1	fraction: .09, fn = 8	fraction: .91, tp = 84

Table 4: Confusion Matrix for l=0.1, $\lambda = 0$, with a threshold $\tau = .5$. Length of test dataset is 200.

		predicted label	
		0	1
true label	0	fraction: .97, tn = 105	fraction: .03, fp = 3
	1	fraction: .78, fn = 72	fraction: .22, tp = 20

Table 5: Confusion Matrix for l=0.01, $\lambda = 0$, with a threshold $\tau = .5$. Length of test dataset is 200.



(a) l = 1 (b) l = 0.1 (c) l = 0.01

Figure 7: ROC Curves for the model, $\lambda = 0$ (no regularization).

10 Regularization

In this section, we add a Gaussian prior over the paramters with unit variance, $P(\beta_m) = \frac{1}{Z}exp(-\frac{\beta_m^2}{2})$ for m = 0 ... N. Equation (4) is the resulting gradient ascent.

$$\frac{\delta L(\beta)}{\delta\beta} = eqn(3) + \frac{\delta log(P(\beta_m))}{\delta\beta} = \sum_{n=1}^{N}(y(n) - \sigma(\beta^T x^{(n)})x(n)) + \frac{\delta}{\delta\beta}(log(\frac{1}{Z}) + log(exp(-\frac{\beta_m^2}{2}))).$$

$$\frac{\delta L(\beta)}{\delta\beta} = \sum_{n=1}^{N}(y(n) - \sigma(\beta^T x^{(n)})x(n)) - \beta \tag{4}$$

The train and test log likelihood obtained with regularization are very similar to the results obtained without adding a prior to the likelihood. The number of true positives and true negatives is very similar to the case without regularization. The expected improvement did not happen because the Variance in the prior is too high.

l	1	0.1	0.01
Train loglikelihood	-0.21	-0.22	-0.50
Test loglikelihood	-0.29	-0.31	-0.69

Table 6: Train and Test loglikelihood **per datapoint** with regularization $\lambda = 1$.

		predicted label	
		0	1
true label	0	fraction: .85, tn = 92	fraction: .15, fp = 16
	1	fraction: .12, fn = 11	fraction: .88, tp = 81

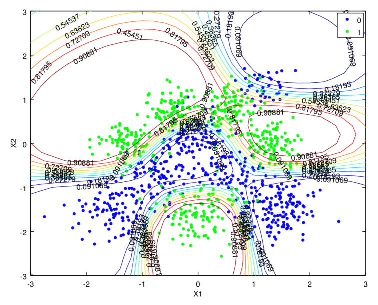
Table 7: Confusion Matrix for l=1, $\lambda = 1$

		predicted label	
		0	1
true label	0	fraction: .92, tn = 99	fraction: .08, fp = 9
	1	fraction: .11, fn = 10	fraction: .89, tp = 82

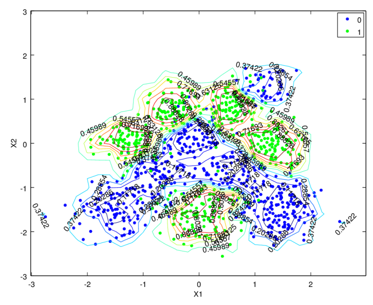
Table 8: Confusion Matrix for l=0.1, $\lambda = 1$

		predicted label	
		0	1
true label	0	fraction: .07, tn = 8	fraction: .93, fp = 100
	1	fraction: .01, fn = 1	fraction: .99, tp = 91

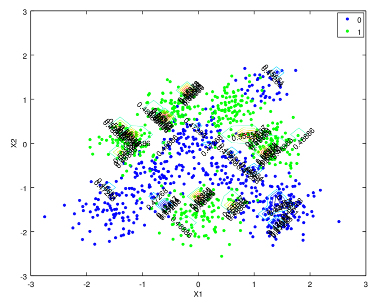
Table 9: Confusion Matrix for l=0.01, $\lambda = 1$



(a) l = 1

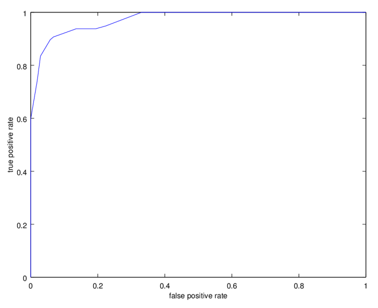


(b) l = 0.1

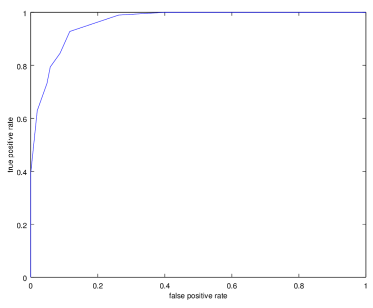


(c) l = 0.01

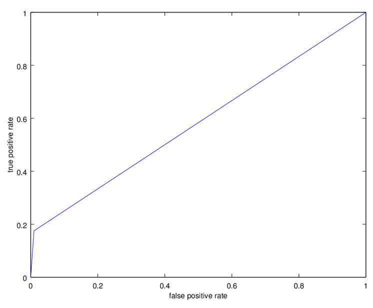
Figure 8: Contour of the likelihood for the training and testing datasets.



(a) l = 1



(b) l = 0.1



(c) l = 0.01

Figure 9: ROC Curves for the model. $\lambda = 1$ (with regularization).