Foundations of Data Analysis

Lab Assignment: Supervised Learning

Christian Wiskott, 01505394.

November 10, 2020

1 Data-Visualization

Fig. 1 shows the 3-d plot of the three features of all 100 samples from the training-set which are split into two classes. Class 0 has the label *Iris-versicolor* and class 1 the label *Iris-virginia*. These class-labels were converted into (0,1).

As it can be seen in the plot, the two classes seem to be well-suited for linear separation via a hyperplane.

2 Regularized logistic regression model using stochastic gradient descent

The first task was to extend the feature matrix X by a column of *1's* to incorporate the offset-term into the calculation.

Next, in order to compute the optimal weight vector θ_* that minimizes the loss of the logistic regression model, the regularized stochastic gradient descent (SGD) method was chosen. The SGD uses a randomly chosen subset of size k of the training-data to update the θ value, instead of taking the entire set each iteration. This seeks to approximate the true gradient and leads to increased computational efficiency. The random numbers were implemented such that, no sample was chosen twice in a subset i.e. random selection without replacement.

In order to make the computation of the gradient more efficient, the problem was reformulated using matrix-vector multiplication instead of summing up using a simple for-loop. This simplification lead to a 4x-speedup compared to the naive gradient-formulation. The proof that these two formulations are equivalent is presented in fig. 2.

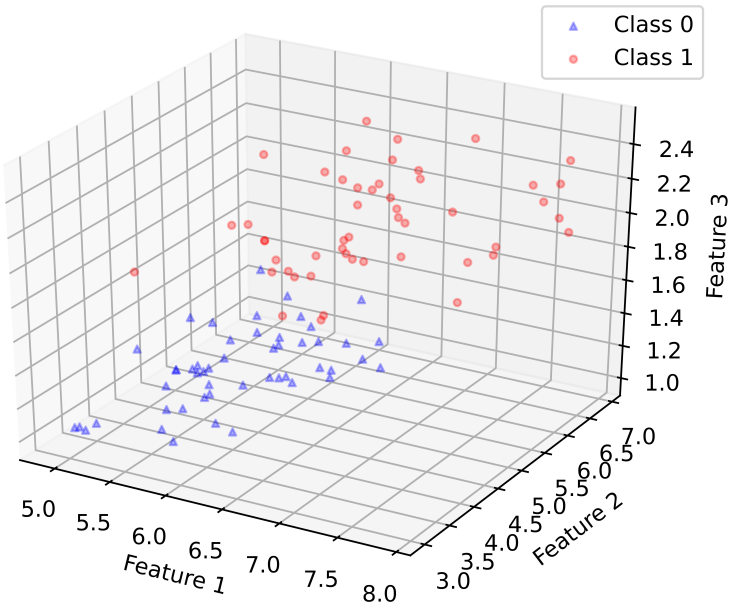

Figure 1: Plot of the features for the members of the two classes.

The gradient of the loss is defined as:

$$\begin{aligned} \frac{\partial}{\partial\theta} L_{l.r.}(\theta, X, y) &= \sum_{i=1}^{m} [(h(\boldsymbol{x_i}) - y_i)\boldsymbol{x_i}^T] + \lambda \; sign(\theta) \\ &\iff X^T \; [h_\theta(X) - y] + \lambda \; sign(\theta) \end{aligned} \tag{1}$$

where $h_\theta(X)$ is a column-vector containing the value of the hypothesis $h_\theta(x_i) = \frac{1}{1+\exp(-x_i\theta)}$ for each sample x_i.

The first SGD parameters that were tested, were: number of iterations T = 100, batch size k = 20, initial learning rate $\eta_{init} = 0.1$ and regularization parameter $\lambda = 1$. In order to asses the quality of the model, the logistic regression loss and the number of correctly labeled samples were computed. This was done by comparing the true labels with the predicted labels using the optimal hypothesis that was determined via the SGD.

The results with these parameters were rather mixed since, with an approx. loss of 0.66 the models classification accuracy fluctuates quite significantly between 50% and 85% correct classifications (see table 1).

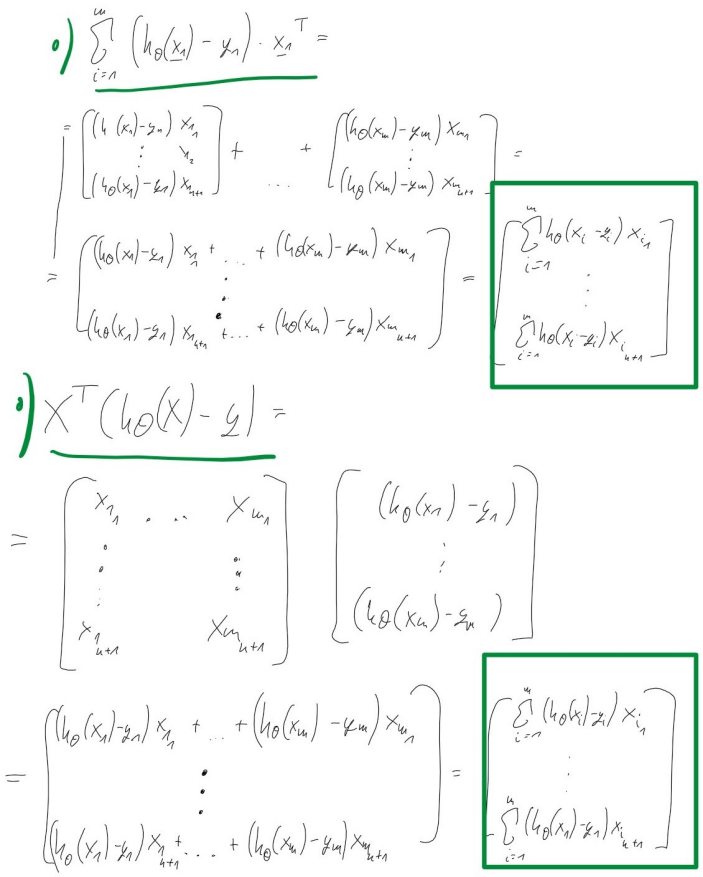

Figure 2: Shows that both formulations lead to the same result. $n+1$ refers to the extension of the matrix X by the offset terms. m describes the number of samples.

T	k	η_{init}	λ	loss	corr. class.
100	20	0.1	1	0.659 ± 0.003	$\sim 70\%$

Table 1: Results for recommended parameters using the loss-function and the number of correctly predicted labels by the hypothesis. Values were obtained by averaging over 1000 SGD-computations.

T	k	η_{init}	λ	loss	corr. class.
500	20	2	-0.7	-0.004 ± 0.004	$\sim 95\%$

Table 2: Optimal parameter-set. Although the loss is quite small, the relatively high standard deviation reveals a relatively large fluctuation of the loss for these parameters.

Experimenting with the parameters lead to the fig. 3 which showcases the individual effects of the parameters on the loss and the classification accuracy. The orange vertical line (loss = 0) helped to identify the set of parameters that lead to the best overall performance i.e. minimizing the loss while maximizing the classification accuracy. These were empirically determined and led to a loss of ~ 0.004 and 95% correct predictions (see. table 2).

Fig. 3 also shows that, when choosing the parameters shown in table 2, the greatest influence on the performance stems from the regularizer and the initial learning rate. As the regularizer increases beyond 0, the number of correct predictions sinks precipitously and the loss diverges away from 0. The optimal value was chosen as -0.7.

Similar but opposite behaviour is visible when studying the effect of the initial learning rate η_{init}. With a value of 2, the loss is close to 0 and the number of correct classification is optimal.

When looking at the effect of the batch size k and the number of iterations T, it's apparent, that in this case, the influence on the quality plays a minor role compared to the first two parameters but in principle reduce the loss and increase the accuracy of the model as they both increase. The change in loss and accuracy changes comparatively little as T and k are altered (see fig. 3). $T = 500$ and $k = 20$ have been chosen as optimal with the help of the orange horizontal line.

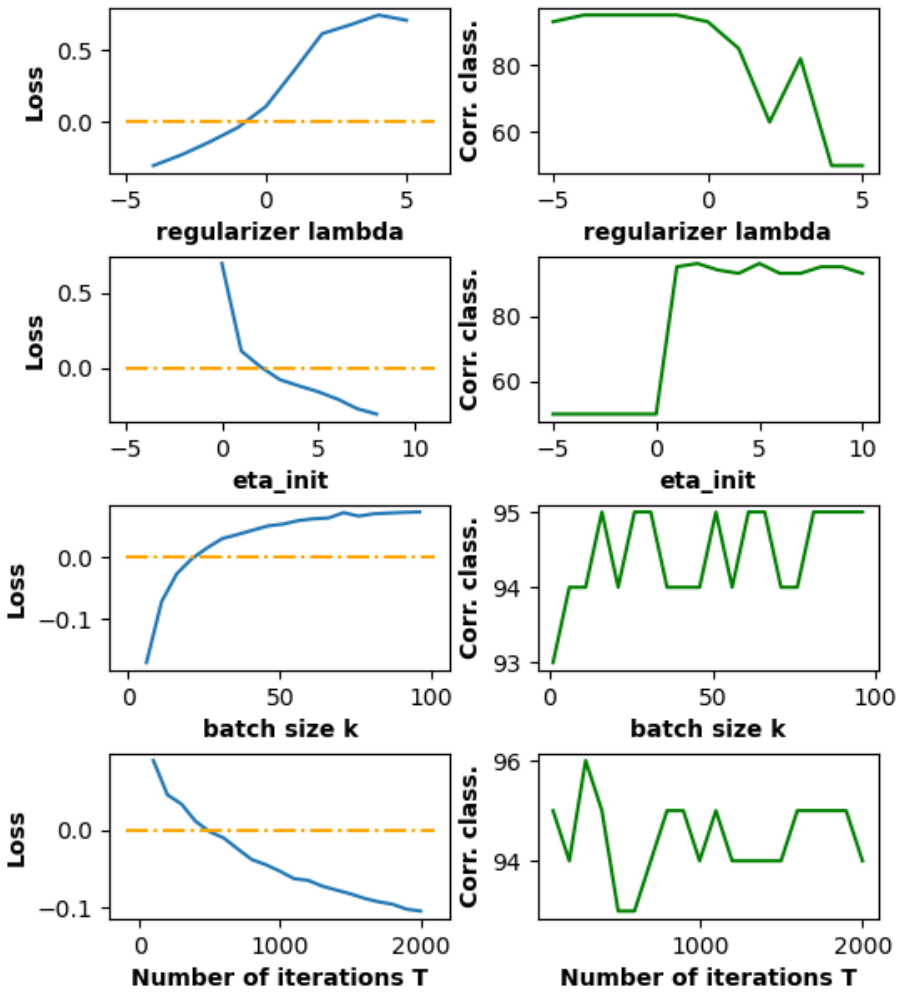

Figure 3: Study of the effects of the parameters on the loss and the accuracy of the logistic regression model. The following optimal parameters were empirically derived, such that the quality measurements were optimized: $\lambda = -0.7$, $\eta_{init} = 2$, $k = 20$, $T = 500$. Every row corresponds to the study of the respective parameter. The other parameters are kept constant to isolate the effect.

3 Separating hyperplane

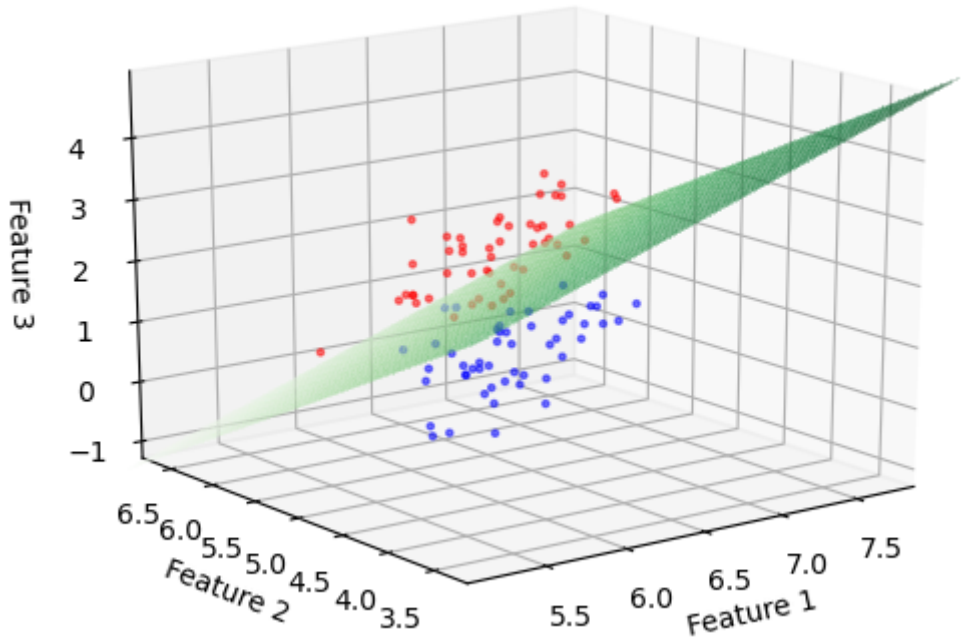

Figure 4: Plot of the hyperplane separating the two classes.

Fig. 4 shows the plot of the data and the obtained separating hyperplane in green. It separates the two classes very nicely with $\sim 95\%$ accuracy.

4 Further questions

1. The different versions of GD exist for different uses and always pose a trade-off between accuracy and convergence-speed. As the naive GD takes the whole data set for the update each iteration, it's slow and has a high demand for memory space as the size of the data-set increases. SGD combats this by only using a subset of the training-data for the update and hence, converges faster, while showing larger fluctuations during the descent. Hence both algorithms have their place. (1)

2. The SGD produces the same trained model with a batch size of $k = m$, with

m being the number of samples in the training-set, which makes it the naive GD. But since this is exactly what SGD seeks to avoid, it's not helpful.

3. In order to use a different loss function, it is necessary to substitute the old gradient for the new one in line *5* of the pseudocode. The other steps of the computation remain the same.

4. Since the regularization term controls the complexity of the model i.e. penalizes high θ_i-values in order to avoid over-fitting, the expected result without the regularizer would be a better training model but would perform worse for the test-set, especially as the number of features increases and a more complex model is necessary. (2)

References

[1] Sebastian Ruder. *An overview of gradient descent optimization algorithms.*, `https://ruder.io/optimizing-gradient-descent/index.html#minibatchgradientdescent`.

[2] Shams S. *Regularized-logistic-regression.*, `https://machinelearningmedium.com/2017/09/15/regularized-logistic-regression/`.