Hot Topics in Machine Learning (HWS17)
Assignment 1: Logistic Regression

Steffen Schmitz
University of Mannheim
stefschm@mail.uni-mannheim.de

1. DATASET STATISTICS

Explore and preprocess the dataset.

1.a Kernel Density Plot

Task. Look at the kernel density plot (code provided) of all features and discuss what you see (or don't see).

We estimate the kernel density estimation with the gaussian_kde method from the scipy.stats package[1]. It returns the estimated probability density function of each feature. As described in the documentation the plot may be "oversmoothed" what leads to non-zero results for $x < 0$, although the statistics (Output of Figure 2) show that all features have minimal values that are greater or equal to zero.

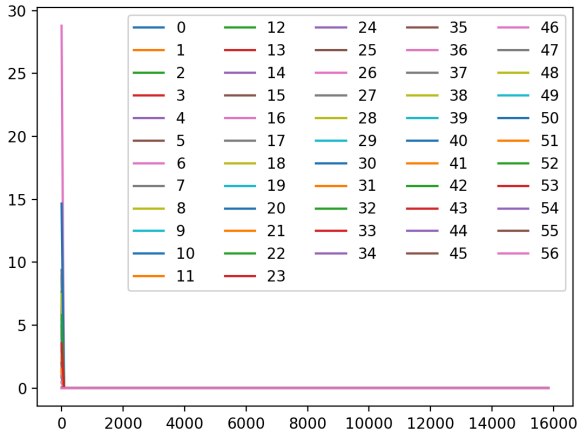

Figure 1: Unregularized Kernel Density Plot

Figure 1 displays the result of the gaussian_kde for the training dataset. The x-axis has a scale from 0 to 16000 and the y-axis ranges from 0 to 30. We can see that almost all data is close to zero and all features approximate zero asymptotically for values $x >> 0$.

There are four visible spikes at $x \approx 0$ that range to about 28, 15, 10 and 4, respectively. All other features are either smaller or overwritten by the latest features.

[1]https://docs.scipy.org/gaussian_kde

The behaviour that is observed in the graph is confirmed by the dataset's statistics that we get by executing the code in Figure 2. The features 0 to 54 are percentages and, therefore, have a range from 0 to 1, while the other features are counts of uppercase letters in the e-mail. Their minimum is 1 and the maximum can range into the thousands.

```
# look at some dataset statistics
import scipy.stats
scipy.stats.describe(X)
```

Figure 2: Show statistics about the data.

From this we can subsume that the current visualization is insufficient, because a few features that can take large numbers overshadow the features that represent percentages. It is hard to draw any conclusions about the data under the current circumstances.

1.b Normalization using z-scores

Task. Normalize the data using z-scores, i.e., normalize each feature to mean 0 and variance 1. Normalize both training and test data. In particular, think about how test data should be normalized.

For machine learning algorithms it is often helpful to normalize or scale the features to avoid huge coefficients or the dominance of a few features in algorithms, which use the euclidean distance as their error function. It also makes gradient descent converge faster compared to the convergence on unnormalized data [2].

The z-score or standard score transforms the features to a normal distribution with mean 0 and variance 1 and is computed with:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where μ is the mean and σ the standard deviation [4].

The sklearn.preprocessing package contains a StandardScaler that normalizes features with the z-score [2].

Figure 3 shows how to apply the normalization to input array X to get output array Xz. At first, is fits the mean

[2]http://scikit-learn.org/StandardScaler

```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(X)

# save mean and variance for later use
mean = scaler.mean_
var = scaler.var_

Xz = scaler.transform(X)
Xtestz = scaler.transform(Xtest)
```

Figure 3: Normalization using sklearn.

and standard deviation to X and, secondly, transforms it into a new array. This enables the transformation with the same values to the test set to guarantee that the scaled value for every input is computed alike in the training and the test set.

The input arrays X and $Xtest$ are transformed into Xz and $Xtestz$.

1.c Normalized Kernel Density Plot

Task. Redo the kernel density plot on the normalized data. What changed? Is there anything that "sticks out"?

Figure 4 shows the estimated probability density functions after normalization. The x-axis' range is $[-6\sigma, 6\sigma]$ and it includes negative values that are now valid due to the changes made on the dataset. We use the six sigma range, because it contains almost all datapoints (99,99966%). After the transformations all features take values in the range $[-1, 46]$ and we can see in Figure 4 that some features have a positive skew. This means that most data points of those features are concentrated on the left side of the graph or around zero, respectively. This is supported by the fact that the features have a mean of zero and variation one, but show huge positive values, compared to the smallest negative value. Those values could be some outliers that may be interesting for the machine learning algorithm.

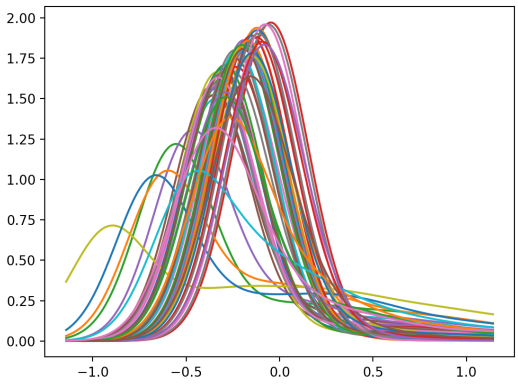

Figure 4: Normalized Kernel Density Plot

The new plot improved the readability and allows to make inferences about the data. The data also points to features or values that may become relevant for the algorithm and the training phase.

2. MAXIMUM LIKELIHOOD ESTIMATION

2.a Effects of Rescaling

Task. Show analytically that rescaling (multiply by constant) and shifting (add a constant) features leads to ML estimates with the same likelihood if there is a bias term. Why do you think we computed z-scores then?

In Maximum Likelihood Estimation (MLE) we try to find a value $\hat{\theta}$ that maximizes the likelihood $\mathcal{L}$ that we see the values observed in a given sample $\mathbf{X}$[1, p.316].

Supposing that $\hat{\theta}$ is the MLE of θ, then we want to prove that transforming the input by rescaling and shifting leads to an MLE of $f(\hat{\theta})$ for $f(\theta)$, where f is, in our example, the function in equation 1.

Casella and Berger show in that we get the same answer for the likelihood if the function f is a one-to-one function. This is called the "invariance property of maximum likelihood estimators". [1, p.319]

To show that equation 1 is one-to-one it suffices to show that $f(x) = f(y) \Rightarrow x = y$ [3].

$$f(x) = \frac{x-\mu}{\sigma} = f(y) = \frac{y-\mu}{\sigma} \; ; \; \cdot\sigma$$

$$\Leftrightarrow x - \mu = y - \mu \; ; \; +\mu$$

$$\Leftrightarrow x = y$$

The z-score, therefore, is a one-to-one function and yields the same maximum likelihood estimate.

Feature scaling or normalization is desirable, because it improves the convergence of gradient descent, without changing the accuracy of the predictions [2].

2.e Compare Gradient Descent with Stochastic Gradient Descent

Task. Explore the behavior of both methods for the parameters provided to you. Discuss!

The gradient descent converges to a fixed error in about 13 epochs and reduces epsilon to zero after about 25 epochs. The reduction of epsilon needs the same number of epochs in the stochastic gradient descent, but it converges faster to the approximate minimum. It only needs about 7 epochs. The convergence of both algorithms is shown in Figure 5.

Differences of the algorithms include that the gradient descent is deterministic as it will find the same optimum in the same number of steps, while the stochastic gradient descent has some randomness included due to the random selection

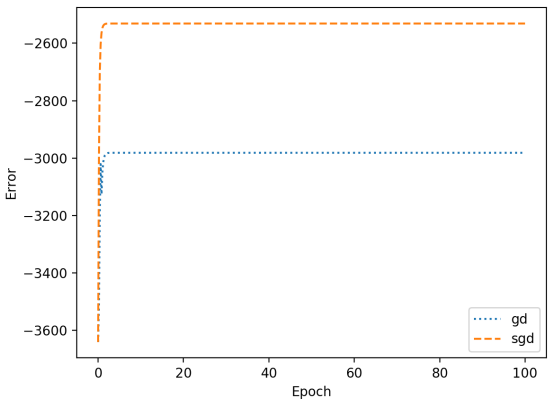

Figure 5: Gradient Descent vs Stochastic Gradient Descent

of the samples. The number of steps to converge and the exact optimum may, therefore, change between different runs of stochastic gradient descent.

Another feature of stochastic gradient descent is that it reduces the memory footprint. While gradient descent computes the updated weights and updates them all at once, the stochastic gradient descent does this incrementally.

As we can see in Figure 5 the error computed by stochastic gradient descent is also lower than the gradient descent error. We would expect them to converge to the same optimum. The error might be caused by a numerical trick in the definition of the log-likelihood function, which we use to avoid the division by zero.

3. PREDICTION

Task. Complete the predict and classify methods for the predicted spam probability and predicted class label, respectively. Explore the models that you fit in the previous task and discuss. Study the composition of the weight vector: which features are important, which are not? Is this intuitive?

For gradient descent and stochastic gradient descent we get a parameter vector wz_gd and wz_sgd, respectively, that is fitted to our normalized training set. To make predictions we multiply the test set $Xtestz$ with the fitted parameter vectors and compute the sigmoid function of the result. If the result for a feature is bigger than 0.5 we set it to 1 and 0 for all other results. The implementation is shown in Figure 6. Using the scikit-learn classification-report[3] we get the Precision, Recall and F1-Score for both, stochastic and normal gradient descent. The results for gradient descent and stochastic gradient descent are similar with only a 1% deviation between them. See Figure 7 for more details.

More interesting is that the fitted parameter vectors of gradient descent and stochastic gradient descent differ. If we compute the statistics for each parameter vector as described in Figure 2 we can see that the gradient descent has a bigger variance (4.45) in its parameters and has positive skew, while the stochastic gradient descent has a variance of 1.52 and a small negative skew (-0.16).

[3]http://scikit-learn.org/classification_report

```
# Returns vector of pred. confidence [0,1]
def predict(Xtest, w):
    return sigma(Xtest @ w)

# Returns vector of predictions {0,1}
def classify(X, w):
    def pred(x):
        return 1 if x > 0.5 else 0
    return np.vectorize(pred)(predict(X, w))
```

Figure 6: Methods to predict and classify the test set.

```
Gradient descent prediction accuracy

              precision    recall  f1-score   support

           0       0.94      0.88      0.91       941
           1       0.83      0.91      0.87       595

avg / total       0.90      0.89      0.89      1536


Stochastic gradient descent prediction accuracy

              precision    recall  f1-score   support

           0       0.94      0.89      0.91       941
           1       0.84      0.91      0.87       595

avg / total       0.90      0.90      0.90      1536
```

Figure 7: Precision, Recall and F1-Score

In the parameter vector of gradient descent there are some features that have a high impact. Features 6, 7, 15, 19, 22, 23, 52, 56 have a high positive impact and indicate spam, while features 24, 25 and 41 have a big negative impact and, therefore, indicate not-spam. They correspond to the frequency of remove, internet, free, credit, 000, money, $ and the total length of consecutive uppercase letters in the case of a positive impact and hp, hpl and meeting in the case of a negative impact.

As the dataset is provided by the Hewlett-Packard Labs which may be abbreviated as hp or hpl the negative impact of those acronyms is not surprising. A high frequency of the word meeting is also expected in non-spam e-mails as spammers usually have a low interest in meeting their recipients.

On the other hand, the occurrence of internet, free, credit and money in combination with dollar-signs and many consecutive uppercase letters is what we would expect in spam.

We can draw the conclusion that the weights that are assigned in gradient descent do not come as a surprise and fit the common estimate of spam e-mails.

The parameter vector of stochastic gradient descent shows approximately the same results, but overall, has a smaller variance. The outliers are the same, but the overall weights are smaller.

4. MAXIMUM APOSTERIORI ESTIMATION

4.b Effect of Prior

Task. Study the effect of the prior on the result by varying the value of λ. Consider at least the training data log-likelihood, the test data log-likelihood, and the prediction accuracy. Are these results surprising to you?

For an initial parameter configuration of w, we can see that the log-likelihood takes on huge values for both the training and the test data. The bigger λ is, the bigger is the penalty on large coefficients in the weight vector and the lower the chance of overfitting the data. An issue may be that the data is underfit for very large values of λ.

We can see in the results in Figure 8 that the F1-Score increases for bigger lambdas, but also decreases after a certain threshold.

```
Gradient descent prediction accuracy for lambda=0

             precision    recall  f1-score   support

          0       0.94      0.88      0.91       941
          1       0.83      0.91      0.87       595

avg / total       0.90      0.89      0.89      1536


Gradient descent prediction accuracy for lambda=50

             precision    recall  f1-score   support

          0       0.93      0.94      0.94       941
          1       0.91      0.89      0.90       595

avg / total       0.92      0.92      0.92      1536


Gradient descent prediction accuracy for lambda=100

             precision    recall  f1-score   support

          0       0.94      0.92      0.93       941
          1       0.88      0.90      0.89       595

avg / total       0.91      0.91      0.91      1536
```

Figure 8: Precision, Recall and F1-Score for different λ

A way to compute a good value for λ would be a cross-validation set that is used to compute different results and optimize the F1-Score, by trial-and-error. The expected accuracy on unknown data can still be calculated through the test data afterwards.

4.c Composition of Weight Vector

Task. Study the composition of the weight vector for varying choices of λ (try very large values). Try to explain what you saw in the task above.

For ever increasing values of λ, the mean and the variance of the values of the parameter vector approach zero. This is expected as a high value of λ translates into a high penalty for high values of the parameter vector. The model, therefore, is prone to underfit the training data and make biased decisions. This explains the worse F1-Score that we can observe in section 4.b.

5. EXPLORATION

Task. Explore variants of preprocessing and logistic regression further.

See the Jupyter Notebook for further exploration.

6. REFERENCES

[1] G. Casella and R. Berger. *Statistical Inference*. Duxbury Resource Center, June 2001.

[2] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[3] D. Mitra. How to determine if a function is one-to-one. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/101978 (version: 2017-09-26).

[4] I. Mohamad and D. Usman. Standardization and its effects on k-means clustering algorithm. 6:3299–3303, 09 2013.