Machine learning for computational linguistics: Assignment 2

Zarah Weiß

8 Jun 2016

Task 1

Task 1.1: Write down the fitted model equation (estimated probability given the predictor)

$$P(is\ German) = \frac{1}{1 + e^{0.4125701 - (-0.0053695 * sentence\ length)}}$$

Task 1.2: Write a brief (with no more than three sentences) interpretation of the coefficients

```
##
## Call:
## glm(formula = isGerman ~ sent_length, family = binomial(link = "logit"),
##     data = dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0058  -0.9785  -0.9454   1.3879   1.6342
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.4125701  0.0182046 -22.663  < 2e-16 ***
## sent_length -0.0053695  0.0007967  -6.739 1.59e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 56199  on 42510  degrees of freedom
## Residual deviance: 56153  on 42509  degrees of freedom
## AIC: 56157
##
## Number of Fisher Scoring iterations: 4
```

The intercept has a value of -0.413 with a small standard error. Albeit its comparativeley low value, it is highly significantly different from 0 with $p < 0.01$. The predictor sentence length is -0.005 and also highly significant with $p < 0.01$. The negative polarity indicates that the probability of a sentence being classified as German decreases with increasing sentence length.

Task 1.3: Report accuracy, precision, recall and F1-score of the fitted model on the training data

```
# calculate accuracy
accuracy <- function(tp, fp, tn, fn) {
  return((tp + tn) / (tp + tn + fp + fn))
}
accuracy.glm1 <- accuracy(tp.glm1, fp.glm1, tn.glm1, fn.glm1)
accuracy.glm1
```

```
## [1] 0.6261203
```

```
# calculate precision
precision <- function(tp, fp) {
  return(ifelse((tp + fp) > 0, tp / (tp + fp), 0))
}
precision.glm1 <- precision(tp.glm1, fp.glm1)
precision.glm1
```

```
## [1] 0
```

```
# calculate recall
recall <- function(tp, fn) {
  return(ifelse((tp + fn) > 0, tp / (tp + fn), 0))
}
recall.glm1 <- recall(tp.glm1, fn.glm1)
recall.glm1
```

```
## [1] 0
```

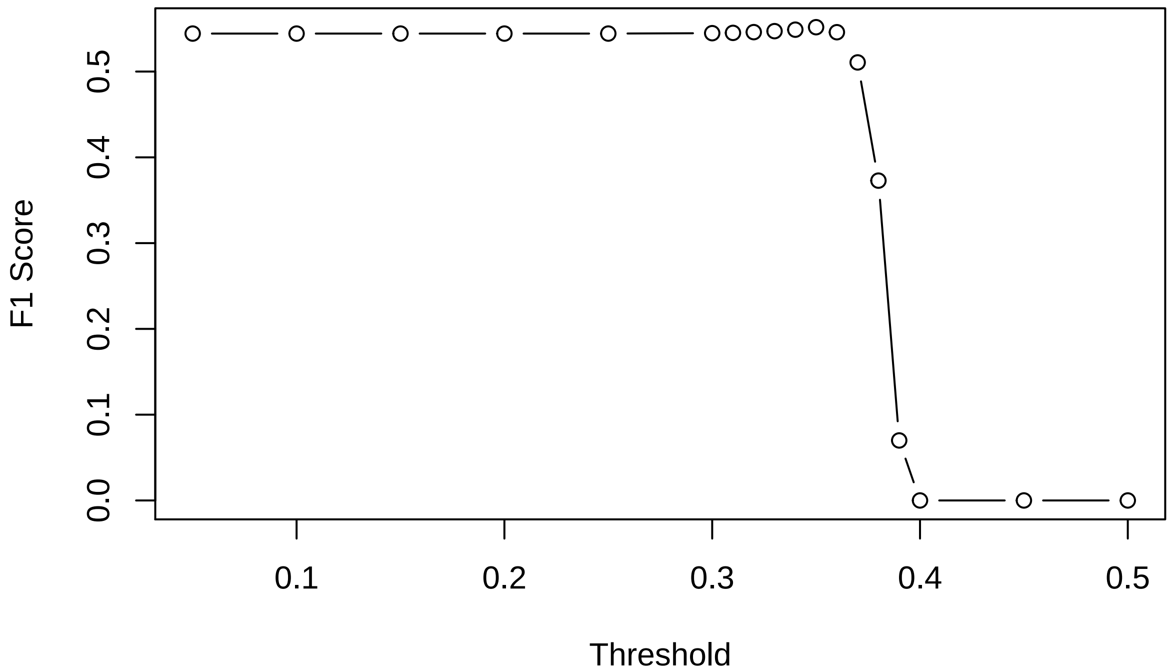
```
# calculate f score
f1Score <- function(tp, fp, fn) {
  return(
    ifelse(precision(tp, fp) + recall(tp, fn) > 0,
           2 * (precision(tp, fp) * recall(tp, fn)) / (precision(tp, fp) + recall(tp, fn)),
           0))
}
f1.glm1 <- f1Score(tp.glm1, fp.glm1, fn.glm1)
f1.glm1
```

```
## [1] 0
```

Task 2

Task 2.1: Find and report the best threshold value that maximizes the F1-score

The best threshold is 0.35 leading to an f score of 0.552. This is also illustrated in the following plot.



Task 2.2: Write down the discriminant function (the function f(X) whose value is positive for the positive instances (sentences in German) and negative for the negative instances)

```
f(x) = 0, if x < 0.35, else f(x) = 1
```

Or in R:

```
dat$IsGermanPred.glm1.bestThreshold <- ifelse(dat.pred.glm1 > tBest, TRUE, FALSE)
```

Task 2.3: Report the accuracy, precision, recall and F1-score at the best threshold value

```
# calculate new measures
accuracy.glm1.35 <- accuracy(tp.glm1.35, fp.glm1.35, tn.glm1.35, fn.glm1.35)
accuracy.glm1.35
```

```
## [1] 0.4191621
```

```
precision.glm1.35 <- precision(tp.glm1.35, fp.glm1.35)
precision.glm1.35
```

```
## [1] 0.3877405
```

```
recall.glm1.35 <- recall(tp.glm1.35, fn.glm1.35)
recall.glm1.35
```

```
## [1] 0.9559582
```

```r
f1.glm1.35 <- f1Score(tp.glm1.35, fp.glm1.35, fn.glm1.35)
f1.glm1.35
```

```
## [1] 0.5517066
```

Task 3

```
## 'data.frame':    42511 obs. of  17 variables:
##  $ ADJ        : num  0 0.0526 0.0345 0 0.0333 ...
##  $ ADP        : num  0.143 0.158 0.172 0 0.167 ...
##  $ ADV        : num  0 0 0 0 0.0333 ...
##  $ AUX        : num  0 0 0 0 0 ...
##  $ CONJ       : num  0 0 0 0 0 ...
##  $ DET        : num  0.2857 0.0526 0.1034 0 0.0667 ...
##  $ NOUN       : num  0.143 0.211 0.103 0 0.1 ...
##  $ NUM        : num  0 0.0526 0.0345 0 0.0333 ...
##  $ PART       : num  0 0.0526 0 0 0 ...
##  $ PRON       : num  0 0 0 0 0 ...
##  $ PROPN      : num  0.143 0.211 0.379 0 0.4 ...
##  $ PUNCT      : num  0.1429 0.0526 0.1034 1 0.1 ...
##  $ SCONJ      : num  0 0 0 0 0 ...
##  $ VERB       : num  0.1429 0.1579 0.069 0 0.0667 ...
##  $ X          : num  0 0 0 0 0 ...
##  $ sent_length: int  7 19 29 1 30 18 31 16 18 9 ...
##  $ isGerman   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

Task 3.1: This time, besides the sentence length, use 15 additional predictors that indicate the relative frequencies of POS unigrams within the sentence. Evaluate your model with probability threshold of 0.5,

```r
summary(glm.2)
```

```
##
## Call:
## glm(formula = isGerman ~ ., family = binomial(link = "logit"),
##     data = dat.reduced)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8720  -0.6880  -0.2825   0.7560   5.8400
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -17.061351   0.837509 -20.372  < 2e-16 ***
## ADJ          17.716542   0.847298  20.909  < 2e-16 ***
## ADP          17.852731   0.855065  20.879  < 2e-16 ***
## ADV          21.209763   0.858209  24.714  < 2e-16 ***
```

```
## AUX          11.635816   0.872059  13.343  < 2e-16 ***
## CONJ         20.280064   0.899407  22.548  < 2e-16 ***
## DET          31.363951   0.864858  36.265  < 2e-16 ***
## NOUN         12.418114   0.845375  14.689  < 2e-16 ***
## NUM          17.360289   0.862973  20.117  < 2e-16 ***
## PART          6.950780   0.964719   7.205 5.81e-13 ***
## PRON         19.295260   0.863053  22.357  < 2e-16 ***
## PROPN        16.059863   0.838902  19.144  < 2e-16 ***
## PUNCT        17.367860   0.853688  20.345  < 2e-16 ***
## SCONJ        -0.769437   1.081114  -0.712    0.477
## VERB          9.260456   0.874992  10.583  < 2e-16 ***
## X                   NA         NA      NA       NA
## sent_length   0.008460   0.001185   7.140 9.32e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 56199  on 42510  degrees of freedom
## Residual deviance: 38197  on 42495  degrees of freedom
## AIC: 38229
##
## Number of Fisher Scoring iterations: 7
```

The intercept is negative again (as with the previous model), but has a more extreme value, namely -17.061. It significantly differs from zero, with $p < 0.01$. As for sentence öength, the feature is still significant, however, it is not negatively correlated with a sentence being German anymore, i.e. the longer a sentence, the more likely it is to be German. This might be due to the fact that the sentences from the German data set are longer (mean = 18.76) than the English ones (mean = 15.33), but shorter than the Japanese ones (mean = 26.78), see below.

The POS features are mostly highly significant, too, i.e. make good predictors for language, and positively correlated with a sentence being German. Interestingly, the only negative coefficient, namely the relative frequency of the SCONJ POS tag, is also not predictive with $p = 0.477$. The other not significant predictor is the relative frequency of X, for which NA is returned. This indicates a high correlation of X with some other predictor.

Also, the AIC droped from 56,157 to 38,229 indicating that the second model is much better suited for the identification of German in this data set.

```r
# Sanity check: sentence length across languages
summary(dat$sent_length[dat$language=='English'])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    6.00   13.00   15.33   21.00  159.00
```

```r
summary(dat$sent_length[dat$language=='German'])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   12.00   17.00   18.76   24.00  115.00
```

```r
summary(dat$sent_length[dat$language=='Japanese'])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   17.00   24.00   26.78   34.00  127.00
```

Task 3.2 … and report accuracy, precision, recall and F1-score values.

```
# calculate accuracy, precision, recall, and fscore
accuracy.glm2 <- accuracy(tp.glm2, fp.glm2, tn.glm2, fn.glm2)
accuracy.glm2
```

```
## [1] 0.792195
```

```
precision.glm2 <- precision(tp.glm2, fp.glm2)
precision.glm2
```

```
## [1] 0.7288344
```

```
recall.glm2 <-recall(tp.glm2, fn.glm2)
recall.glm2
```

```
## [1] 0.7073739
```

```
f1.glm2 <- f1Score(tp.glm2, fp.glm2, fn.glm2)
f1.glm2
```

```
## [1] 0.7179438
```

Task 3.3: Does this model suffer from the class imbalance problem equally?

No it doesn't. This can be seen at the results for accuracy, precision, recall and F1 being quite similar. This might be due to the fact, that we have a number of good predictors now, that can identify the classes not just by a majority vote.

Task 4

Task 4.1: Fit two separate models to the complete data, one with L1, the other one with L2 regularization. For both models, use the regularization parameter Lambda = 50.

see R code

Task 4.2: Briefly explain the differences between the coefficient values.

Regularisation is used to avoid overfitting. This is an issue we are quite exposed to, as we use our training data also as testing data. Model linM.l1 uses a L1 regularisation to constrain the parameter space, that is, it adds the L1 norm multiplied with Lambda = 50 to the cost function. Model linM.l2 uses a L2 regularisation, that is, it adds the Euclidean and not the Manhattan distance to the cost function.

```r
linM.l1
```

```
## $TypeDetail
## [1] "L1-regularized logistic regression (L1R_LR)"
##
## $Type
## [1] 6
##
## $W
##                  ADJ        ADP        ADV       AUX        CONJ
## English    1.0762314 -9.2082185  -2.417411 -6.033646   0.6221297
## German     0.8540112  0.7674345   4.286729 -5.295643   3.2613366
## Japanese -15.3488712 18.1763843 -11.504682 17.976239 -11.1796611
##                DET       NOUN         NUM        PART       PRON
## English   -3.305723  0.3832428  0.03138756  10.9010497   5.478051
## German    14.366471 -4.2269801  0.42427694 -10.0034786   2.410240
## Japanese -46.540107 -0.5114110 -2.03932067   0.6151892 -36.367251
##                PROPN      PUNCT      SCONJ       VERB          X
## English    1.5917124 -0.4029408   1.434296   5.866365  13.557877
## German    -0.7680987  0.3870908 -17.846450  -7.457666 -16.791608
## Japanese -12.6458059 -2.6026855  22.879924 -11.554714  -8.737827
##           sent_length        Bias
## English  -0.018475217  0.07676909
## German    0.008523455 -0.19022828
## Japanese  0.067141719 -0.39995776
##
## $Bias
## [1] TRUE
##
## $ClassNames
## [1] English  German   Japanese
## Levels: English German Japanese
##
## $NbClass
## [1] 3
##
## attr(,"class")
## [1] "LiblineaR"
```

```r
linM.l2
```

```
## $TypeDetail
## [1] "L2-regularized logistic regression primal (L2R_LR)"
##
## $Type
## [1] 0
##
## $W
##                ADJ        ADP        ADV       AUX       CONJ        DET
## English   0.6857224 -7.5130140 -0.7832083 -3.028476  0.7378444  -1.924565
## German    1.9886923 -0.7236038  1.8059409 -3.144421  0.7341410   7.122533
## Japanese -5.7071440 14.4031352 -4.0984844 13.035045 -2.9635377 -15.065589
##                NOUN        NUM         PART       PRON      PROPN
```

```
## English  -1.156510 -0.1383791  2.254829954  4.4464282 -0.0087899
## German   -2.184627 -0.1861638 -1.089416445  0.1642955  0.3484264
## Japanese  4.220319  2.0201765  0.003708113 -9.2919527 -4.9457591
##               PUNCT      SCONJ      VERB          X  sent_length
## English  -0.01843138  0.1497023  3.801662  2.9900801 -0.019900851
## German   -0.03923988 -1.5100842 -1.958400 -1.8183289  0.004098783
## Japanese  1.07009851  4.9089908 -1.149571 -0.8029581  0.042046788
##               Bias
## English   0.4948953
## German   -0.4902554
## Japanese -4.3635235
##
## $Bias
## [1] TRUE
##
## $ClassNames
## [1] English  German   Japanese
## Levels: English German Japanese
##
## $NbClass
## [1] 3
##
## attr(,"class")
## [1] "LiblineaR"
```

In principle, the coefficients of both models resemble each other. However, the L2 regularization returns less extreme values. So when the L1 regularised model returns a coefficient of -10.688 for ADJ in Japanese sentences, the L2 regularised model returns a coefficient of -5.707. Something similar can be observed for DET, where for German the L1 regularised model returns a coefficient of 14.547 and the L2 regularised model a coefficient of 7.122. As the L2 regularisation may be viewed as equivalent to a normal prior centered around the mean, if we view it in a Bayesian setting, this is not surprising: the constraint on the parameter space is simply centered around zero leading to coefficients more closely to zero compared to the L1 regularisation. However, even if the coefficients take less extreme values centered more closely around zero in the L2 regularised model, within a parameter both models keep the same relation of the language specific coefficients, i.e. for PROPN, Japanese has in both models the most extreme negative slope compared to the slope for German and English, even though the absolute values differ across the models.

Task 4.3: Calculate and compare accuracy of both L1 and L2 regularized models.

```
# calculate accuracy for model using L1 regularisation
accuracy.l1.overall <- accuracy(tp.l1.english + tp.l1.german + tp.l1.japanese,
                                fp.l1.english + fp.l1.german + tp.l1.japanese,
                                tn.l1.english + tn.l1.german + tn.l1.german,
                                fn.l1.english + fn.l2.german + fn.l1.japanese)
accuracy.l1.overall
```

```
## [1] 0.7772778
```

```
# calculate accuracy for model using L2 regularisation
accuracy.l2.overall <- accuracy(tp.l2.english + tp.l2.german + tp.l2.japanese,
                                fp.l2.english + fp.l2.german + tp.l2.japanese,
```

```
                                 tn.l2.english + tn.l2.german + tn.l2.german,
                                 fn.l2.english + fn.l2.german + fn.l2.japanese)
accuracy.l2.overall
```

```
## [1] 0.771957
```

The accuarcy for both models is rather high and very similar: for the L2 regularised model it is 0.77, for the L1 regularised model it is slightly higher with 0.78.

Task 4.4: Tabulate the confusion matrix of the L2-regularized model you fit in the previous step.

```
##              pred.english pred.german pred.japanese
## exp.english         11577        4591           454
## exp.german           4366       11290           238
## exp.japanese          233          57          9705
```

Task 5: Using the same model in exercise 4 with L2 regularization, evaluate the model accuracy using 10-fold cross validation, and report the average accuracy and its standard error.

The mean accuracy:

```
avgAcc
```

```
## [1] 0.8385645
```

The standard error:

```
stdErr
```

```
## [1] 0.01101142
```