University of Iceland

Machine Learning (REI602M)

Preterm Birth Prediction

Authors:

Einar Jóhannesson

Hallur Kristinn Hallsson

Zakia Shafaee Ahmad

Introduction

In this project we try to predict what pregnancies are term- and preterm pregnancies from EHG signals recorded in pregnant women. A preterm birth is when a baby is born before 37 completed weeks of gestation.

We extract various features from the EHG signals and make datasets that suit our needs and define Suport Vector and Convolutional Neural Network classifiers that predict if a pregnancy is a term or preterm pregnancy, with various results.

Methods

Data

The following datasets were used

- TPEHGT DS: 5 recordings of non-pregnant women, 13 preterm recordings and 13 term recordings.
- TPEHG DB: 262 term recordings and 38 preterm recordings

The records in both datasets contain information from three EHG signals (labeled S1, S2 and S3). The approximation of the forth signal, S4 was calculated as:

$$S4 = S1 - S2 + S3$$

The datasets were run through a function of our making that extracted features from them and made a new file with our features and if the data was term or preterm. These files were used when training and testing the classifiers.

Feature extraction

Before feature extraction each signal was filtered using 4th order band-pass Butterworth filter (scipy.signal.butter) furthermore a filter was used in both directions to cancel out phase shift (scipy.signal.filtfilt). Each feature was calculated in the following specific frequency bands.

- 0.08 - 1.0 Hz (beating frequencies)
- 1.0 - 2.2 Hz (heart beat frequencies)
- 2.2 - 3.5 Hz (higher harmonics)
- 3.5 - 5.0 Hz (higher harmonics)

After signals were filtered the features were extracted using the following methods.

- **Sample entropy**

The sample entropy was calculated using the Antropy[1] package. It was decided to use the Antropy package after having experimented using other packages as well as implementing it ourselves and seeing that using the Antropy package was the fastest and easiest to use.
The Antropy package only has the embedding dimension as a parameter. That parameter was then tuned so that the difference between preterm and term pregnancy readings were as high as possible.
The embedding dimension was tuned by trying three different values three times based on the previous result. After testing for the values 2,3,4,6,8,16 we found that 3 is the best value for our needs.

- **Median frequency**
  The power spectrum was found using the welch[2] function from the scipy package.
  The power spectrum was then sorted and the median found from the sorted power spectrum.
- **Peak amplitude of normalized power spectrum**
  Power spectra was calculated using Fourier transform (scipy.fft), in order to make the Fourier transform faster only half of the frequencies were computed using (scipy.fft.rfft), then the peak amplitude was calculated for each frequency band as following:

$$PA = max_{k=k_{low}}^{k=k_{high}} \left(\frac{P(k)}{P_{max}}\right)^3$$

  Where k_{high} and k_{low} are indexes of the power spectrum components at the start and end of each frequency band, and P_{max} is the maximum component of the power spectrum in frequency interval 0 to 10.

Classification

Classification was performed using two different methods:

- Support Vector Machine using an rbf kernel and k-fold hyper parameter tuning on the datasets with extracted features.
- 1-dimensional Convolutional Neural Network on a 15000 unit segment of the time-series of each filtered signal. (Performed only on the TPEHG dataset)

Support Vector Machine

The Support Vector Machine was initialized with a Radial Basis Function (rbf) kernel and balanced class weight to account for imbalance between term and preterm records. The C and gamma

[1] https://raphaelvallat.com/antropy/build/html/index.html
[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.welch.html
[3] F. Jager, S. Libenšek and K. Geršak, *Characterization and automatic classification of preterm and term uterine records.* PLOS ONE, 13, 8, e0202125

hyper-parameters were tuned using Grid Search over the K-Fold method with C values in the range $[2^{-7}, 2^{17}]$ and gamma values in the range $[2^{-15}, 2^{9}]$ [4]. Standard scaling was performed separately at every pass. Due to the inherent imbalance in the dataset (A prediction consisting of only term births could yield an accuracy score beyond 85%!), the balanced_accuracy_score metric was used to evaluate accuracy.

The SVM we used to evaluate the extracted features of each dataset.

```
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import balanced_accuracy_score

X = X_large
y = y_large

cs = []
gammas = []
sp = 12

for i in range(24):
    cs.append(2**(-7 + i))
    gammas.append(2**(-15 + i))

b_score = 0
b_preds = []
b_c = 0
b_g = 0
for c in cs:
    for g in gammas:
        kf = KFold(n_splits=sp, shuffle=True)
        kf.get_n_splits(X)
        score = 0
        preds = []
        for train_index, test_index in kf.split(X):
            X_train, X_test = X[train_index], X[test_index]
            y_train, y_test = y[train_index], y[test_index]
            clf = SVC(class_weight='balanced', C=c, gamma=g)
            scaler = StandardScaler().fit(X_train)
            X_train = scaler.transform(X_train)
            clf.fit(X_train, y_train)
            y_pred = clf.predict(X_test)
            preds.append(y_pred)
            X_test = scaler.transform(X_test)
            score += balanced_accuracy_score(y_test, y_pred)
        score = score/sp
        if(score > b_score):
            b_score = score
            b_c = c
            b_g = g
            b_preds = preds

print("Best average scores: ", b_score)
print("Best C value: ", b_c)
print("Best gamma value: ", b_g)
```

1-Dimensional CNN

An attempt was made to construct a 1-Dimensional Convolutional Neural Network to fit the time-series of filtered signals. This was only attempted on the larger TPEHG dataset. The network was an adaptation of one found on the internet [5]. The final network consisted of the following:

---

[4] Hyperparameter ranges: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
[5] https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/

- 2x 1-dimensional convolutional layers with 32 filters and kernel size of 3. ReLu used as activation. 50% Dropout performed on the second.
- 1-dimensional Max-Pooling layer with pool size of 2
- A 100 node dense layer, with ReLu activation and 50% Dropout
- A 2-node output layer with softmax activation and a 20% l2 kernel regularization.

Binary Crossentropy was used as the loss function, with the Adam optimizer (learning rate 10^{-5}).

To overcome class imbalance, we implemented a simple upsampling method by re-introducing preterm data until the proportions became more or less equivalent. It was important that this was done after the set had been split into training and testing sets, as otherwise we ran a risk of having the same pre-term data in both the training and testing sets. A careless early run where this had not been done resulted in a ludicrous near 100% test accuracy!

Results

Data

After extraction we got datasets with 49 columns.

The first 16 columns are the sample entropy, the next 16 are the median frequencies and then 16 peak amplitudes, four signals per frequency band each. And the last column was if it was a preterm or term pregnancy, where 0 is term, 1 is preterm and -1 is non-pregnant.

Classifiers

SVM

We ran the Support Vector Machine code five times on each dataset: The dummy period dataset, the contraction period dataset and the TPEHG dataset. Attempts were made to run an SVM on a different version of the TPEHGT dataset, where each datapoint represented an entire recording, but this was abandoned as we ended up only with 26 datapoints.

All data from non-pregnant women was excised from the dummy and contraction datasets, resulting in both cases in 100 datapoints, each representing a single period. The TPEHG dataset meanwhile contained 300 datapoints. All datapoints contained 48 extracted features as described above.

For each run, we recorded the best accuracy (using the sklearn ‘balanced accuracy’ metric) along with the accompanying values for the C and gamma hyperparameters.

Dummy Periods

	Run 1	Run 2	Run 3	Run 4	Run 5
Best Accuracy	74.3512%	74.4167%	72.6786%	71.1393%	69.8690%
Best value of C	2^{7}	2^{4}	2^{5}	2^{5}	2^{11}
Best value of Gamma	2^{-12}	2^{-9}	2^{-10}	2^{-11}	2^{-15}

Contraction Periods

	Run 1	Run 2	Run 3	Run 4	Run 5
Best Accuracy	65.7937%	65.1845%	65.1905%	65.5536%	66.3512%
Best value of C	2^{8}	2^{9}	2^{7}	2^{8}	2^{5}
Best value of Gamma	2^{-15}	2^{-15}	2^{-15}	2^{-15}	2^{-13}

TPEHG Recordings

	Run 1	Run 2	Run 3	Run 4	Run 5
Best Accuracy	62.9277%	62.7424%	63.2199%	62.5%	61.5763%
Best value of C	2^{8}	2^{11}	2^{5}	2^{13}	2^{7}
Best value of Gamma	2^{-13}	2^{-15}	2^{-9}	2^{-2}	2^{-12}

1-Dimensional CNN

The 1-Dimensional Convolutional Neural Network was run using the actual time series of filtered signals on the TPEHG dataset. To ensure a uniform number of timesteps, we looked up the smallest recording in the dataset and used it to determine a set length to extract from each recording (15000 timesteps). The data was fit into the network described above with a batch size of 500 over 20 epochs.

With some effort, we were able to design the 1-dimensional CNN in such a way that it would no longer overfit up to 100% training accuracy. However, the actual testing accuracy of the

network left much to be desired, never going above 50% in any of our runs (when using balanced accuracy metric. When using regular accuracy it would log a score as high as 80%). While there is certainly a possibility that a competent 1-dimensional Convolutional Neural Network can be designed and trained to adequately process this data, designing a functioning CNN is a complicated task and we believe that greater experience is needed to be able to properly do so.

Summary

We found that the SVM classifier that used features extracted from dummy periods had the best result, ~70-75% accuracy. Given that the dataset included many periods over a few individuals, it can be assumed that, despite ca. 70% not being exemplary, a large enough sample of dummy periods on the same individual might give a good insight into whether the pregnancy would end prematurely or not. We can therefore say that our SVM classifier could possibly be used in reality though results should be taken with a pinch of salt.

References

F. Jager, S. Libenšek and K. Geršak, *Characterization and automatic classification of preterm and term uterine records.* PLOS ONE, 13, 8, e0202125 https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0202125#pone.0202125.ref010

P. Seguro, *A deep neural network for insurance classification written in tensorflow,* [online] Kaggle.com, from https://www.kaggle.com/camnugent/class-imbalance-a-lesson-learned-tensorflow-nn

Wikipedia contributors. (2021, January 2). Sample entropy. In *Wikipedia, The Free Encyclopedia*. Retrieved 15:28, April 23, 2021, from https://en.wikipedia.org/w/index.php?title=Sample_entropy&oldid=997794642

Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin. *A Practical Guide to Support Vector Classification.* Department of Computer Science, National Taiwan University. From https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Brownlee, Jason. *1D Convolutional Neural Network Models for Human Activity Recognition.* Machinelearningmastery.com, from https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/

Collaboration

This project's collaboration went well. Each team member took initiative and contributed to the project. We held remote meetings on a regular basis.
Group members main contributions:

- Einar - bandwidth filtering and classifiers.
- Hallur - sample entropy and median.
- Zakia - peak amplitude and power spectrum.