Data Science: Principles and Practice
Final Assignment B

Kamilė Stankevičiūtė (`ks830`)
Gonville & Caius College

7th February 2020

1493 words

1 Data preparation

I use the raw data (*diabetic_ data_ original.csv*) of the same medical care record dataset [1] as in Assignment A. The following section describes the changes in preprocessing steps with respect to the previous Assignment.

1.1 Selection of training instances

The preprocessing steps below resulted in 16515 training instances.

Filtering Only the patients with at least two recorded encounters were considered, since the later encounter is required to generate the label for the earlier one. To ensure independence of all examples in the dataset (corresponding to the assumptions of the probabilistic model described below), only one randomly sampled example per patient was used.

Multiple readmission Taking a closer look at the dataset has shown that some patients have recorded follow-on visits even in cases when the readmission outcome was 'NO' (for example, at least 44 patients have been *not readmitted more than once*). Further investigation would be necessary to determine the reason for this (e.g. human error, admission to another hospital,...) in order to handle it correctly. I decided to not sample from any no-readmission encounters (of which after initial filtering there were 490).

1.2 Feature preprocessing

The main advantage of using the deep learning tools is their capacity to select the most useful features without manual feature engineering by the analyst. For this reason, the models will be trained in an end-to-end manner on all features, but with the following considerations.

Anonymisation As before, the encounter and patient numbers were removed at training stage.

Missing values In the full dataset, the *weight* feature was missing in 98.6% of instances, so I excluded it from training. For other features, the missing values were imputed using median or constant strategy for numerical and categorical features respectively.

Numerical and categorical features As before, the features were converted to numerical or categorical values based on their meaning, and further normalised/one-hot encoded for performance reasons. To avoid a large number of sparse features, the diagnoses were compressed to broader categories (e.g. circulatory, respiratory, etc.)

2 Machine learning set-up

2.1 Train and test set split

With a view to get a more generalisable model, I split the data into qualitatively different sets using a single-component t-SNE projection (perplexity=30), holding out the top 10% of values for testing, and another 10% for validation.

2.2 Probabilistic model

I model length of next stay using *zero-truncated Poisson distribution* (ZTP). I truncated the zero because, if the follow-on visit happens, the length of stay must be at least 1, violating the assumptions of the regular Poisson distribution where observations of 0 are possible (on the other hand, I still allow predictions of more than 14 days that were not observed in the dataset but in theory should be possible). The observations $y_i \in \mathbb{N}^+$ therefore follow

$$Y_i \sim \mathrm{ZTP}(f_\theta(x_i))$$

with probability mass function

$$\mathbb{P}[Y_i = y_i | \theta, x_i] = \frac{f_\theta(x_i)^{y_i}}{(e^{f_\theta(x_i)} - 1) y_i!}. \tag{1}$$

Loss function For M examples, parameters θ maximising the probability of the dataset (expressed as the product of probabilities in (1) for each observation, assuming independence) minimise the loss function

$$\mathcal{L} = \sum_{i=1}^{M} -y_i \log(f_\theta(x_i)) + \log(e^{f_\theta(x_i)} - 1) + \log(y_i!) \quad (2)$$

which I will use for training, but additionally taking the average (multiplying $\mathcal{L}$ by constant $1/M$).

3 Feedforward neural networks

Ideally, I wish to be able to predict the length of the follow-on visit for a patient from as little as a single encounter rather than sequences of multiple encounters: the original dataset suggests that in majority (54%) of cases this is indeed the most information that a hospital would have for a patient. (It is also unclear how to handle the multiple readmission anomalies, mentioned above, for the encounter sequences to be correct.)

For this task I implement a set of feedforward neural networks.

3.1 Choice of architecture

It is tempting to assume that the deeper the neural network, the better, since with more parameters the model has a higher capacity to learn more complex feature representations (but taking longer to train).

To test this assumption and as part of the hyperparameter selection process, I implemented three types of networks, which I will call *deep*, *moderate* and *shallow*, containing around 6×10^6, 3.5×10^6 and 2.80×10^5 trainable parameters respectively. Full specification (and the training results) can be found in supplementary code.

Selection process To compare their typical performance, I trained each of the networks 5 times with early stopping, measuring the average time it takes for the models to converge, and evaluating the training and development set predictions using Pearson's correlation coefficient r. I chose this metric as being arguably easier to interpret compared to the custom loss function.

Vanishing gradients During training, the networks (especially deeper ones) tended to suffer from the vanishing gradients problem, getting stuck with unlucky initialisations about once in every five runs. Adding batch normalisation between layers and using ReLU activation function resolved this issue, but sacrificed the training speed.

	deep	**moderate**	**shallow**
time, s	276	84	35
epochs	13.8	7.4	9.4
train r	0.208	0.376	0.458
dev. r	0.127	0.178	0.207

Table 1: Mean training time, number of epochs to early stopping convergence, and training and development set performance of feedforward neural networks.

3.2 Results

The results, presented in Table 1, seem to suggest that deeper networks are, in fact, not always better. With an almost 8-fold increase in training time and lower correlations on both training and development datasets, this particular experiment suggests that deeper networks might, contrary to the assumption, give even worse predictions than the simpler models. Moreover, since the models are trained with early stopping, the number of epochs themselves (not just the time they take) could also be used to compare the convergence speed. In this case the deep network required the most steps on average, indicating its slow convergence.

However, it is curious that even 6 million trainable parameters are not enough to actually fit the training data better and give correlations closer to 1, despite the antagonistic effect of early stopping. Determining whether this is a fundamental limitation of the dataset (with no useful signal for length of stay prediction), or the lack of hyperparameter optimisation would need further investigation that is out of this report's scope.

3.3 Using PCA features as input

Motivation One possible way to mitigate the slow training times (especially for the *deep* and *moderate* models) could be to use PCA components as input features, hopefully saving some time on learning a useful feature representation. At the same time, if PCA is capable to extract the most useful features with its first few components, it could also be used to effectively reduce the input dimensions, in turn reducing the model size and training time.

Experiments I test the impact of PCA on neural network performance using an experimental set-up similar to the previous section, comparing the training time and development set predictions on the full feature representation, the PCA features of the same dimensionality and the *reduced* PCA (rPCA) feature

	full	PCA	rPCA
deep	276	237	269
moderate	84	85	88
shallow	35	20	20

Table 2: Mean training time (in seconds) on raw, PCA and reduced PCA feature sets.

	full	PCA	rPCA
deep	0.127	0.168	0.116
moderate	0.178	0.210	0.191
shallow	0.207	0.193	0.149

Table 3: Mean development set correlations when training on full, PCA and reduced PCA feature sets.

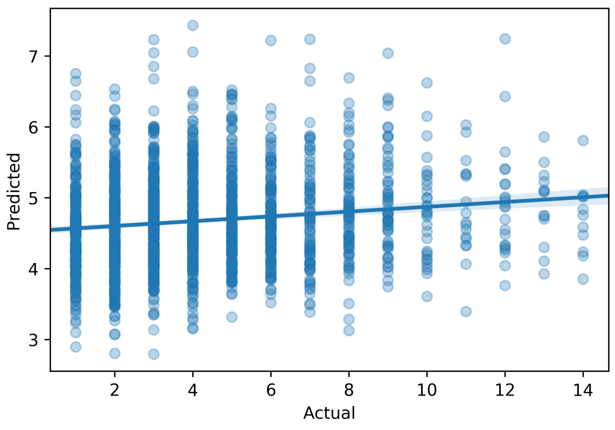

Figure 1: Prediction versus actual length of next stay observations with their linear regression fit, as evaluated on the holdout test set.

set which only considers the first half of the PCA components (with the assumption that they are the most useful).

Results The training times using those techniques are shown in Table 2. While it is possible that PCA could help with speeding up training, the improvement is not guaranteed – for the moderate network, PCA and rPCA only increased the training time. Even more importantly, PCA could not compensate for the overhead of having a bigger network: the shallow network running on the original feature set is still much faster than the deeper counterparts running on PCA features. The takeaway from this is that, for the purpose of decreasing the training time, we should worry about simplifying the network architecture first, and only then consider any fancy dimensionality reduction techniques.

On the other hand, we might be interested in using PCA techniques to improve the predictive power rather than the training time, assuming the usefulness of PCA representation. However, while PCA-transformed features gave slightly better average development set correlations for deeper networks (Table 3), it was not the case for the shallow network, showing once again that PCA is not a magic formula we can use for an instant performance or predictive power boost.

rPCA seems to be particularly ineffective. Not only did it generally *increase* the training time, but with the information loss from the excluded low-variance components it also tended to learn less, as demonstrated in Table 3. Moreover, unless even more layers are decreased in size and number (i.e. the overall model is reduced in addition to using fewer PCA components), the improvement in memory usage will be marginal compared to the memory requirement for the rest of the network.

4 Evaluation

From the above model selection and hyperparameter analysis I have decided that the simplest model – the shallow network without PCA preprocessing – is actually the most promising one, both with respect to the training time and the generalisation. I tested it on the holdout test set to get the final Pearson's r correlation of **0.138** (Figure 1). Given that this number is quite similar to the development set figures in Table 3 and the overall predictive power being weak even on the training set, the final model was able to sustain its performance in the novel context, potentially indicating its generalisability.

References

[1] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore. Impact of HbA1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records. *BioMed Research International*, 2014.