methods, frailty classifier

Structured data

Structured data extracted from the EHR, across our corpus, is summarized in table XX. Infrequently-observed values of categorical variables were set to “other.” Much of the data is missing. For the purpose of predictive modeling, missing values were imputed using chained random forests (Stekhoven and Bühlmann 2012).

term	N Missing	mean of observed	mean including imputed
n_encs	22290	4.7	4.1
n_ed_visits	22290	0.32	0.29
n_admissions	22290	0.39	0.27
days_hospitalized	22290	2	1.4
mean_sys_bp	447	130	130
mean_dia_bp	471	74	74
sd_sys_bp	12703	12	11
sd_dia_bp	12740	7.3	7.1
bmi_mean	6261	29	29
bmi_slope	32633	-0.00086	-0.00069
tsh	49912	2.8	2.8
sd_tsh	64426	1.9	1.2
n_tsh	49912	1.4	1.2
n_unique_meds	2560	16	16
elixhauser	0	2.6	2.6
n_comorb	0	10	10
age	4123	68	68
sexfemale	4105	0.57	0.58
sexmale	4105	0.43	0.42
marital_statusmarried	4105	0.5	0.51
marital_statusother	4105	0.0084	0.0079
marital_statussingle	4105	0.23	0.23
marital_statuswidowed	4105	0.14	0.13
empy_statfull_time	7452	0.18	0.18
empy_statnot_employed	7452	0.054	0.049
empy_statother	7452	0.037	0.033

term	N Missing	mean of observed	mean including imputed
empy_statpart_time	7452	0.023	0.02
empy_statretired	7452	0.56	0.56
raceother	5313	0.057	0.053
racewhite	5313	0.63	0.64
languageother	4105	0.012	0.011
languagespanish	4105	0.0071	0.0066
countyburlington	4105	0.033	0.031
countycamden	4105	0.041	0.039
countychester	4105	0.12	0.11
countydelaware	4105	0.088	0.085
countygloucester	4105	0.036	0.034
countymercer	4105	0.043	0.04
countymontgomery	4105	0.062	0.059
countyother	4105	0.17	0.18
countyphiladelphia	4105	0.37	0.38

Model architecture

The model's target is the quadruple $[Y_{1ic}, Y_{4ic}, Y_{4ic}, Y_{4ic}]$, where Y_{1-4} are the four phenotypes for which the clinical notes are annotated, and c represents the center of the context window w, such that $c = w//2$.

The inputs to our model were pairs S_{it}, E_{iw}, where S are structure data from the EHR for patient i at time t, and E are $w \times 300$ matrices of text representing w total words embedded in a 300-dimensional space. The word embeddings are described in (cite CWE paper).

Inputs are first chunked in batches of size 256, and then passed to a bi-directional LSTM (Gers, Schmidhuber, and Cummins 1999) with u units in each direction. The output of the LSTM is of shape $256 \times w \times 2u$, which are each passed to d blocks consisting of (1) a dense layer with with u units, (2) a leaky ReLU activation (Maas, Hannun, and Ng 2013), and (3) a dropout layer, with a dropout rate of r. The result is then flattened, and fed to each of four dense layers corresponding to the phenotypes, each with 3 units (for positive, negative, and neutral) and having a softmax (i.e.: multinomial logistic) activation. Each dense layer is regularized with both L1 and L2 penalties, at rate λ, and has the shape $256 \times uw$.

Furthermore, we test two variants of this model. In the fully non-parametric version, we repeat observations of S for each w in i to enable S to be concatenated to E, despite its invariance to the sequential progression of text. This augments the input matrix to have dimension $w \times 300 + rank(S)$. In the semiparametric version, the structured data is concatenated to the penultimate dense layer (before the outcome layers), resulting in a dimension of $256 \times uw + rank(S)$. The semiparametric version of the model is equivalent to the multinomial logistic regressions

$$\mathbf{Y}_p = (\alpha + \mathbf{S}\beta + \mathbf{V}\Gamma + \epsilon > 0)$$

for the pth phenotype, where α is the standard linear model intercept, $\mathbf{V}$ is the representation learned by the neural network, and ϵ has the standard type-1 extreme value distribution that corresponds to the multinomial logit link function. In the fully-nonparametric version, $\mathbf{S}$ is subsumed as part of $\mathbf{V}$.

The values of w, u, d, r, λ, and the choice of whether to use the semiparametric or nonparametric versions of the model, are all hyperparameters which are allowed to evolve over the course of active learning.

Hyperparameter selection

To determine optimal hyperparameters, we randomly sample many times from from the distribution of plausible values indicated in table XX.

```
@@@ this is a placeholder for a real table.  it shows the ranges @@@
def draw_hps(seed):
    np.random.seed(seed)
    hps = (int(np.random.choice(list(range(5, 40)))),  # window size
           int(np.random.choice(list(range(1, 10)))),  # n dense
           int(np.random.choice(list(range(10, 100)))),  # n units
           float(np.random.uniform(low = 0.01, high = .5)),  # dropout
           float(np.random.uniform(low = -6, high = -1)), # log10 of l1/l2 penalty
           bool(np.random.choice(list(range(2)))))  # semipar
    model = makemodel(*hps)
    return model, hps
```

The set of hyperparameters chosen is that which minimizes the average of the categorical cross-entropy losses for the four phenotypes, in the holdout set. The holdout is chosen by random selecting 1/3rd of concatenated notes from 2018.

Training

Models are implemented in Tensorflow 2.1.0 (Abadi et al. 2015). Parameters are optimized using the Adaptive Moments (ADAM) optimizer of Kingma and Ba (2014) with a learning rate set to .00001. At model initialization, the bias terms for each phenotype are set to $log(p(1-p))$, where p is the proportion of each tag in the training set. The model is then trained using early stopping with a patience of 5 epochs, and the loss corresponding with the best holdout set performance is saved.

Active learning

Given a final model, subsequent notes to annotate are chosen to maximize a function of average entropy over the four phenotypes. For each token in each note, entropy is defined as

$$h = pr \times log(pr)$$

where pr is a 1×3 vector of probabilities (corresponding to positive, negative, and neutral) summing to one. h is maximized where $pr = [.33, .33, .33]$ – corresponding to maximal uncertainty, and minimized where any value is equal to 1, indicating complete certainty. By the definition of entropy, notes containing regions of high entropy contain text that the model is unable to assign to a phenotype. Active learning seeks these out for subsequent annotation.

The function of entropy that we use to combined values across four phenotypes within a note is

$$\mathbb{E}_{itp}[h_{itcp} \times \mathbf{1}\left(h_{itcp} > \mathbb{E}[h_{itp}]\right)]$$

which is the average of most entropic half of tokens per patient-note, over phenotypes.

TBD

As we move through active learning, plots showing the evolution of losses, entropies, and optimal hyperparameters over time will be made.

References

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. 2015. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." https://www.tensorflow.org/.

Gers, Felix A, Jürgen Schmidhuber, and Fred Cummins. 1999. "Learning to Forget: Continual Prediction with Lstm." IET.

Kingma, Diederik P, and Jimmy Ba. 2014. "Adam: A Method for Stochastic Optimization." *arXiv Preprint arXiv:1412.6980*.

Maas, Andrew L, Awni Y Hannun, and Andrew Y Ng. 2013. "Rectifier Nonlinearities Improve Neural Network Acoustic Models." In *Proc. Icml*, 30:3. 1.

Stekhoven, Daniel J, and Peter Bühlmann. 2012. "MissForest—Non-Parametric Missing Value Imputation for Mixed-Type Data." *Bioinformatics* 28 (1). Oxford University Press: 112–18.