Survival analysis of tcga

Sharon yedidia

Feature Selection and cleaning

(Independently per cancer type and omic)
The methods below were implemented in order to normalize and select the features.

- Log transformation to mirna and exp files
- Remove samples with negative 'last_contact_days_to'
- Removing duplicates
- Removing features with std == 0
- Normalization (scaling) of features
- Calculate mutual information and remove insignificant features relatively to the event the duration
- Correlation between all remaining features and remove highly correlated features (>0.9)

The results of the feature selection in the appendix.

Task 1

In order to have a wide infrastructure and arsenal for comparison, I've implemented multiple methods on each omic and on the merged omics, on each cancer type.
All applied methods mentioned below are pending hyperparameter tuning.

COX model [here]

I implemented the following regularizations using scikit survival library:

- COX model Without regularization
- Ridge regularization
  I checked the alphas [0.01,0.1,0.3,0.5,0.7].
- Lasso regularization
  I checked 50 random α values that give up to 5% of the estimated maximum.
- Elastic-Net regularization
  I chose r = 0.5 and checked 50 random α values that give up to 5% of the estimated maximum.

Gradient Boosting Survival Tree [here]
I checked a range of estimators (weak learners), between 25 to 160 with jumps of 5 estimators in each iteration.

I tried each of the number of estimators on each of the regularizations below.
I implemented the following regularizations using scikit survival library:

- The Gradient boosting model with dropout
  This regularization forces the base learners to account for some of the previously fitted base learners to be missing.
  I tested a range of dropout rates, between 0.1 to 0.3 with jumps of 0.05 in each iteration.
- The Gradient boosting model using subsample
  The subsample regularization uses a subsample out of the training data on each iteration.
  I tested a subsample of the portion between 0.1 to 0.5 with jumps of 0.05 in each iteration.
- The Gradient boosting model using a learning rate
  This regularization makes the gradient boosting slow down the learning and by that avoid overfitting.
  Without the regularization the learning rate is 1,I tested a range between 0.9 to 0.01 with jumps of 0.05 to restrict the influence of individual base learners.

Random Survival Forest [here]

I chose 1000 estimators as the number of estimators.
I implemented the model using the scikit survival library.

Concatenated omics NN [here]

The NN was implemented on the concatenated omics for each cancer type using pytorch and was wrapped by the LogisticHazard model of the library Pycox.
The Logistic-Hazard method parametrizes the discrete hazards and optimizes the survival likelihood (more info here).
I tried a few variations of layers and eventually I used a NN using pytorch containing 9 layers and checked a range of each of the hyper parameters: 32<= x,y,s,k<=40 in jumps of 4 nodes and 0.1<w,z<0.3 in jumps of 0.1.

```
# net = torch.nn.Sequential(
#     torch.nn.Linear(in_features, x),
#     torch.nn.ReLU(),
#     torch.nn.BatchNorm1d(y),
#     torch.nn.Dropout(z),

#     torch.nn.Linear(s, s),
```

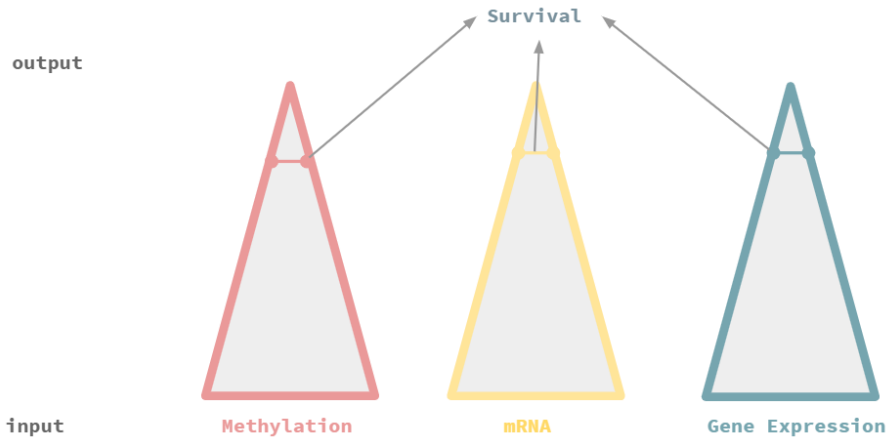
```
#     torch.nn.ReLU(),
#     torch.nn.BatchNorm1d(r),
#     torch.nn.Dropout(w),

#     torch.nn.Linear(k, out_features)
# )
```

I chose to split the duration to 80 intervals -the output layer has 80 nodes..
The best hyperparameters were chosen by freezing the values of all the hyperparameters to be their “median value”, except from the tested one, and choosing the best result during cross validation.
There wasn’t much change between the hyper parameters values so I chose to stay with the default values of the Vanilla NN : x=y=s=k=32, z=w=0.1 .

Multi View NN

I Built a baseline survival NN predictor based on each of the 3 omics with the same structure and hyperparameters of the Concatenated omics NN.
I combined the 3 models by removing the models last layer and adding a fully connected last layer.
All the parameters of the layers accept the new added layer were freezed.
Afterwards I trained the multi view model with the additional fully connected layer that took the separate representations of each model to generate the final predictions.
I chose to split the duration to 80 intervals.



Task 1 - choosing the best model

After using the folds with averaged cross validation concordance index, averaged over all the cancer types:
- Cox model : 0.676
- Gradient Boosting: 0.788
- Random Survival Forest: 0.666
- Concatenated omics NN: 0.88
- Multi View NN: 0.952

The Multi View model got good results - too good to be true?
Indeed.
- The feature preparation included all the data, not data by folds. But the concordance index was still high.
- I found that I put the label as a feature (!). This caused the results to be too good to be true, after fixing it, the result was 0.492 - the lowest score of all the models.
I tested again the hyperparameters tuning using cross validation and got no change in the results.

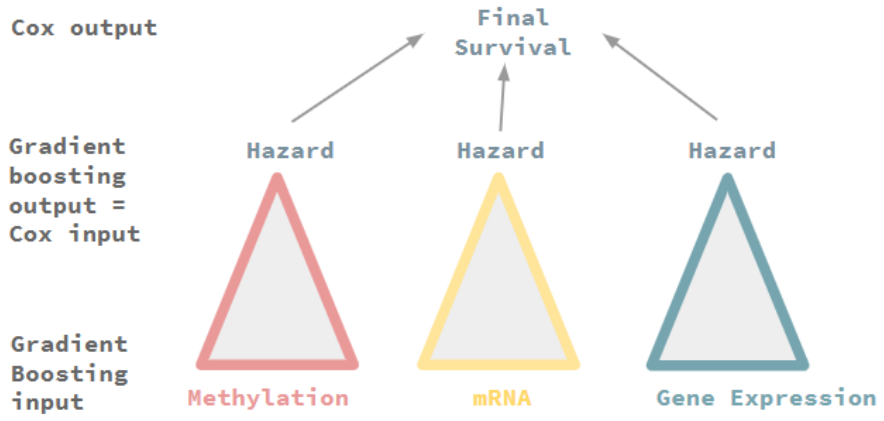
After that I checked different kinds of models other than the logisticHazard:
- **PMP [implementation here]**: The PMF method parametrizes the probability mass function (PMF) and optimizes the survival likelihood. (more info here)
- **DeepHit [implementation here]**: DeepHit is a deep neural network that learns the distribution of survival times directly. (more info here)
- **MLTR [implementation here]**: The (Neural) Multi-Task Logistic Regression is a PMF methods (more info here)

MLTR gave the best results - using the folds with cross validation the averaged concordance index over all the cancer types was 0.529.
So eventually I decided to change direction, assuming that NN should have more data as input and the chosen model is the Gradient Boosting Survival Tree.

Combining regularizations on Gradient Boosting Survival Tree
With input of the concatenated omics, I checked the combinations of parameters described above per number of estimators using cross validation.
The chosen model combined few regularizations and got an optimal averaged concordance index of 0.816 using 80 estimators, learning rate of 0.8, subsample of 0.5 in each iteration and dropout rate of 0.1.

Multi view Gradient Boosting Survival Tree

I built a Gradient Boosting Survival Tree for each omic with the structure described above, each Gradient Boosting Survival Tree will serve as a baseline model.
The output of each baseline model is the hazard function.
In order to combine these models, I built a Cox model that gets as an input the predictions out of each baseline model and predicts the hazard out of it.



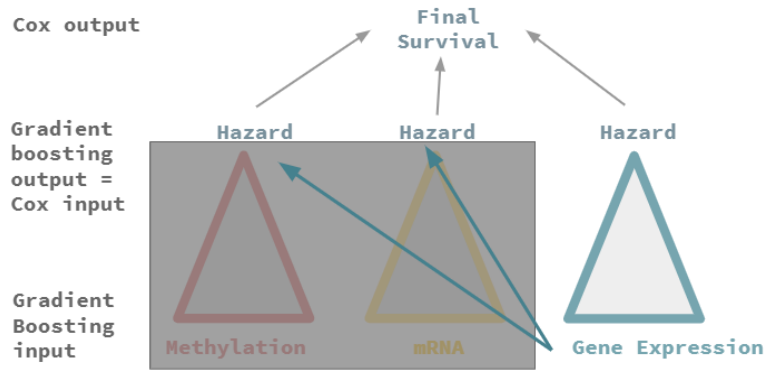
The concatenated omics Gradient Boosting Survival Tree VS. Multi view Gradient Boosting Survival Tree

I compared the 2 models using the folds with cross validation, the averaged concordance index over all the cancer types of the Multi view model was 0.843> 0.816.
So the chosen model is the Multi view Gradient Boosting Survival Tree.
The baseline model of Gene expression got an averaged concordance index of 0.65,
The baseline model of mRna got an averaged concordance index of 0.672,
The baseline model of Methyl got an averaged concordance index of 0.641.

Task 2

The original idea was using the Multi View NN (described above):
- Build a baseline survival NN predictor based on each of the 3 omics
- Combine the 3 models by removing the models last layer and add a fully connected last layer
- Build a predictor that gets as input the omic gene expression and predict the representation layer of the baseline models of the omics mRna and methyl .
- Complete the Multi View NN using the representation layers predictions.

After changing direction to Gradient Boosting Survival Trees I'm using the multi view Gradient Boosting Survival Tree (described above):
- Build a baseline Gradient Boosting Survival Tree predictor based on each of the 3 omics, with the same hyper parameters of task 1.
- Combine the 3 models by using a Cox model that gets as an input the predictions out of each baseline model and predicts the hazard out of it.
- Build a predictor that gets as input the gene expression and predicts the predictions of methyl and mRna of the baseline models.(*described below)
-The predictions complete the multi view Gradient Boosting Survival Tree using gene expression only in the testing.



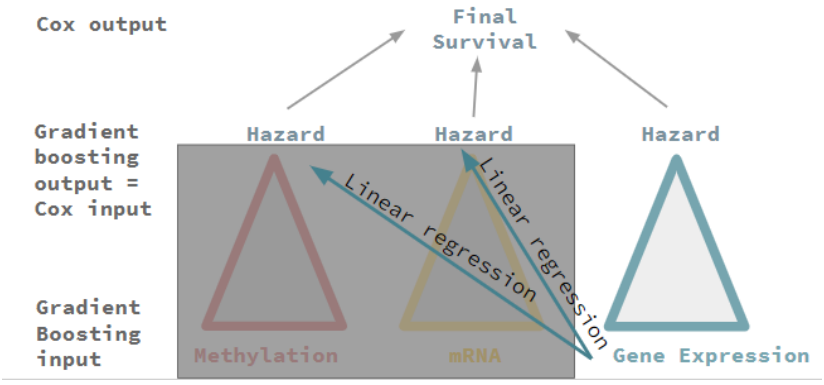
The Gene expression Gradient Boosting Survival Tree VS. Multi view Gradient Boosting Survival Tree

I compared the 2 models using the folds with cross validation, the averaged concordance index over all the cancer types of the Gene expression Gradient Boosting Survival Tree was 0.65 << 0.843 of the multi view model.

Choosing the predictor for mRna and methyl predictions (*)

- NN - 5 layer NN using pytorch of python
  # net = torch.nn.Sequential(
- *# torch.nn.Linear(in_features, x),*
- *# torch.nn.ReLU(),*
- *# torch.nn.BatchNorm1d(y),*
- *# torch.nn.Dropout(z),*
- *# torch.nn.Linear(k, out_features)*
- *#)*
- Random forest - I used the Random Forest model of scikit library with the default values of hyperparameters.
- Linear regression - I used the linear regression model of scikit library- 2 predictors, one for mRna predictions and one for Methyl predictions.

I compared between the multi view model that gets all the omics (concordance index 0.843) to these models. Using the folds with cross validation, the averaged concordance index over all the cancer types the of the model using the NN predictor was 0.531, the model using the Random forest predictor was 0.677, and the model using linear regression model was 0.829 (even better than the concatenated omics gradient boosting model).
The final model was built by the drawing:

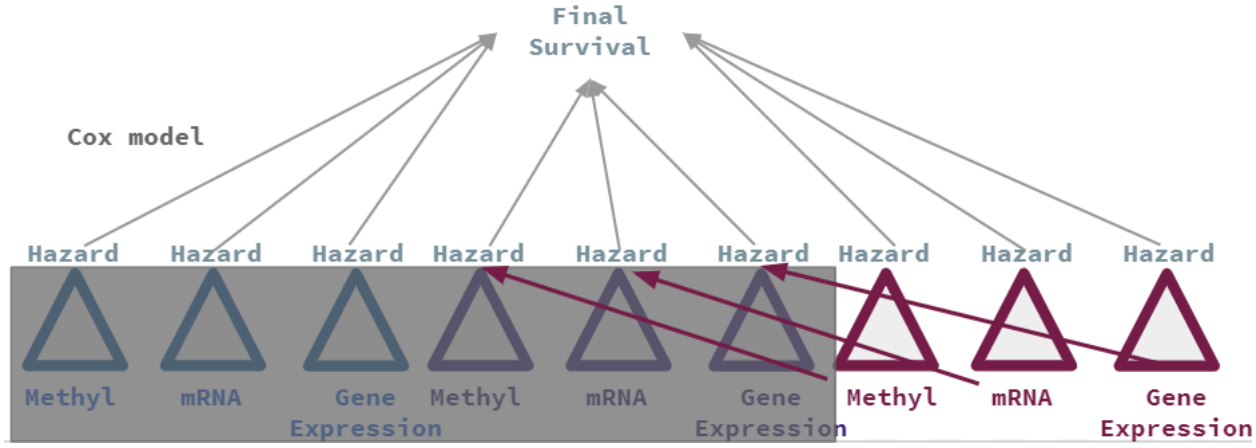


Task 3

The assumption behind this task idea is that the information of the omics of other cancer types can add to the information the tested cancer types.

The original Idea:
- Build a predictor for each of the cancer types in the training based on the best model of task 1.
- Cut each model in some optimal layer and combine the nets to one model.
- Build a predictor of the chosen layer of each of the models of the cancer types that are available only on the training with input of the omics of the cancer type that is available on testing.

After changing direction to Gradient Boosting Survival Trees the idea:
-Build a new predictor for each of the omics for missing cancer types such that:
    inputs: omics of the available cancer type
    Outputs: the predictions of task1 single omic models for the missing cancer types
    We should note that feature preprocessing and selection for the data provided is the same as for the missing cancer types - not as the tested type.
- Combine the models by using a per-missing-cancer type Cox model that gets as an input all the predictions made by each model and predicts the hazard out of it.
In the drawing each color represents a different cancer type, the redish is the one that will be available during the testing.
- Build a linear regression predictor for each omic for each of the cancer types in the training using the omic of the cancer type that is available in the testing.



Choosing the predictor for the cancer types that available only on the training

I chose to use linear regression of scikit because the results of task 2 were very satisfying, The concordance index for each cancer type

The model of task 1 VS. The results of the combined model VS. the combined model using only the cancer type of the testing

	BLCA	BRCA	HNSC	LAML	LGG	LUAD	avrage
Task 1 Multy View Gradient Boosting Survival	0.8	0.86	0.83	0.85	0.86	0.86	0.843333
Combined cancer types model using all cancer types as input	0.83	0.88	0.88		0.81	0.89	0.858
Combined cancer types model using the tested cancer type only as input	0.81	0.87	0.87		0.81	0.88	0.848

Conclusion

For all tasks, infrastructure is set in the form of stand alone per-cancer type models. Such models are found to be best performing when built as multi-view (per omic). After experimenting with many architectures, Gradient Boosting Survival is selected for each view (omic), and the results of these views are aggregated with a Cox model that yields the final prediction.

With this infrastructure in place, task 1 is complete.

In Task 2, some data is missing for their ideal setup: only one omic is present.
In Task 3, we have full data for the test cancer, but want to see if training other cancers' models and their prediction can improve this stand alone model.

For both, additional extra models are built that take existing data and predict the results needed for the top-level relevant cox model.

For task 2, such models get exp data, and predict the predictions of other omic models (to be input to the top level cox).

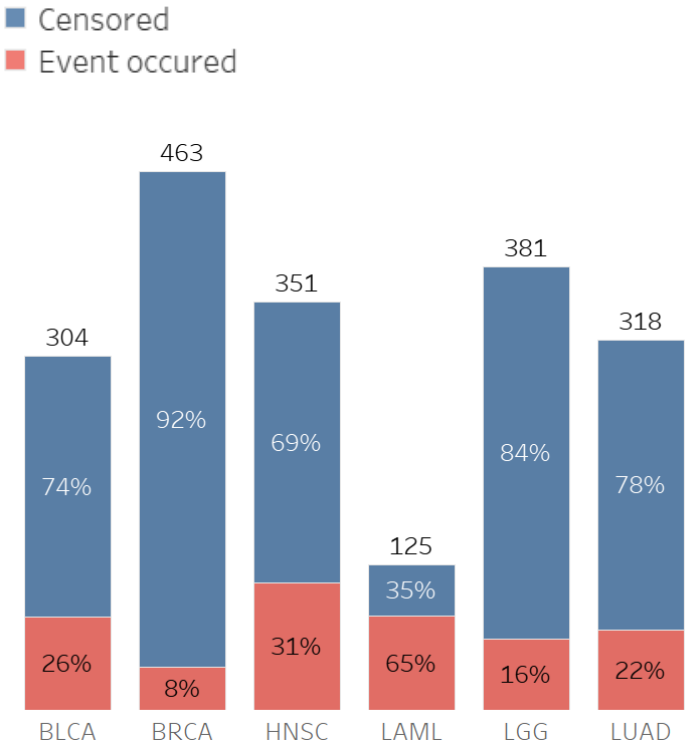
For task 3, such models get "present" cancer omics after being processed for the "missing" cancer, and their output is the predictions of the stand alone models of task 1.

In both tasks 2 and 3, we compare the results with two baselines: each stand-alone cancer model, and a model built with all data (all predictions of all per-omic-per-cancer models). We show that there is some improvement over the stand alone model (but not over the full data scenario).

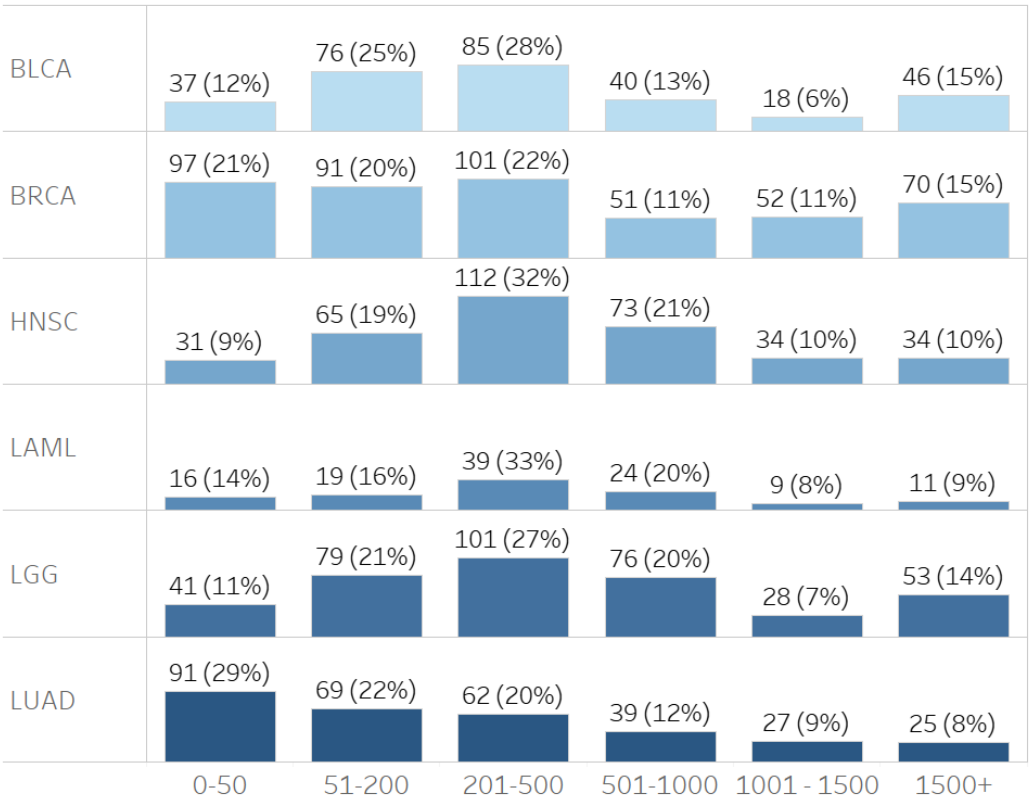
Appendix

Dataset

The dataset contains 1942 patients of 6 different cancer types.
In the graph below you can see the distribution of the patients' event and the amount of patients from each cancer type.



The median duration (time to event) over all the cancer types is 304 days and the average is 640 days. In the graph below you can see the distribution of the duration by days of the different cancer types.



We can see that the data of each cancer type has its own distribution of duration and ratio between censored and uncensored patients.

Results of feature selection

	# Gene exp after	% features left	# mRna after	% features left	# Methyl after	% features left
BLCA	4579	22%	338	18%	4198	21%
BRCA	4890	24%	330	18%	3881	19%
HNSC	4370	21%	343	18%	4122	21%
LAML	5206	25%	243	13%	3577	18%
LGG	4235	21%	342	18%	4596	23%
LUAD	3760	18%	285	15%	2839	14%

Training Times by seconds

Task 1	BLCA	BRCA	HNSC	LAML	LGG	LUAD
exp	18.9	34.8	20.9	7.1	28.5	16.3
methy	18.8	29.2	21.6	5.5	26.9	13.2
mirna	1.2	1.6	1.3	0.4	1.59	1.05
combined multi view	39.3	67.7	45.7	12.7	59.3	31.6

Task2	BLCA	BRCA	HNSC	LAML	LGG	LUAD
methy	1.6	1.8	1.3	0.5	1.5	1.2
mirna	0.17	0.27	0.2	0.06	0.24	0.15

Task3	12.3	19.1	14.3	NA	15.44	12.9

References

Scikit survival user guid
-https://scikit-survival.readthedocs.io/en/latest/user_guide/index.html

Introduction to the pycox package-
https://nbviewer.jupyter.org/github/havakv/pycox/blob/master/examples/01_introduction.ipynb

Understanding PyTorch with an example: a step-by-step tutorial, Daniel Godoy,

https://towardsdatascience.com/understanding-pytorch-with-an-example-a-step-by-step-tutorial-81fc5f8c4e8e#5017

Deep Learning for Survival Analysis, Authors: Laura Löschmann, Daria Smorodina,

FEBRUARY 6, 2020 -
https://humboldt-wi.github.io/blog/research/information_systems_1920/group2_survivalanalysis/#evaluation

Towards Data Science Survival Analysis — Part A
https://towardsdatascience.com/survival-analysis-part-a-70213df21c2e

How to ensemble two model in pytorch?
https://discuss.pytorch.org/t/how-to-ensemble-two-model-in-pytorch/65348

Custom Ensemble approach
https://discuss.pytorch.org/t/custom-ensemble-approach/52024

Building Neural Network Using PyTorch
https://towardsdatascience.com/building-neural-network-using-pytorch-84f6e75f9a