pyanno4rt: a toolkit for machine learning (N)TCP model-based inverse radiotherapy treatment plan optimization

Tim Ortkamp[1,2,3], Patrick Salome[2,4], Oliver Jäkel[2,3,5,6], Martin Frank[1,3], and Niklas Wahl[2,5]

[1]Karlsruhe Institute of Technology (KIT), Scientific Computing Center, Karlsruhe, Germany
[2]Department of Medical Physics in Radiation Oncology, German Cancer Research Center (DKFZ), Heidelberg, Germany
[3]Helmholtz Information and Data Science School for Health (HIDSS4Health), Karlsruhe/Heidelberg, Germany
[4]Clinical Cooperation Unit Radiation Oncology, German Cancer Research Center (DKFZ), Heidelberg, Germany
[5]National Center for Radiation Research in Oncology (NCRO), Heidelberg Institute for Radiation Oncology (HIRO), Heidelberg, Germany
[6]Heidelberg Ion Beam Therapy Center (HIT), Heidelberg, Germany

Abstract Machine learning (ML) models on tumor control probability (TCP) or normal tissue complication probability (NTCP) are on the verge of replacing classical radiotherapy treatment outcome models. In spite of this, solving the inverse problem in intensity-modulated radiotherapy (IMRT) treatment planning still relies on mathematically simple, non-model-based surrogate functions with empirical dose prescription and tolerance parameters, rather than directly optimizing (N)TCP based on such outcome models. In this paper, we present the open-source Python package *pyanno4rt*, a toolkit which implements a technical framework for ML outcome model-based IMRT optimization. *pyanno4rt* leverages transformations of the probability prediction functions as (N)TCP objectives and backward propagation of the gradients from these objectives to the fluence vector, both of which could then be embedded in any gradient-based solver. To demonstrate *pyanno4rt*, we first fitted ML outcome models (logistic regression, neural network, support vector machine) internally, using two outcome data sets from a head-and-neck patient cohort. Second, we optimized photon treatment plans on a head-and-neck patient by adding the corresponding (N)TCP objectives to the conventional plan. Our results show that *pyanno4rt* is able to generate ML model-based treatment plans with appealing dose distributions, while allowing for improved outcome predictions.

1 Introduction

Machine learning models are increasingly applied in radiotherapy treatment outcome prediction, and have already been shown to outperform classical outcome models [1]. One of the main advantages is the ability to handle high-dimensional and mixed inputs, which allows them to exploit large amounts of treatment and patient data and therefore provide a highly personalized treatment outcome prediction. This makes machine learning outcome models interesting for application in intensity-modulated radiotherapy (IMRT), where the inverse problem is commonly solved by using surrogate functions with empirical dose prescription and tolerance parameters, e.g. squared deviation from a reference dose, rather than directly generating outcome-oriented treatment plans.
One possible approach for the integration of inverse treatment plan optimization and machine learning outcome models is at the level of the optimization components (objectives and constraints). However, this poses some challenges, e.g. in terms of model stability with changing input over the optimization iterations or the dependence on the efficiency of the forward objective and backward gradient calculations.
In this paper, we present *pyanno4rt* [2], a toolkit which implements a technical framework for addressing these issues. We first introduce the software framework and the underlying mathematical concepts to derive feasible (N)TCP objectives and gradients from the machine learning model prediction functions, then outline the datasets used for model fitting and provide details on treatment planning. To showcase *pyanno4rt*, we integrate the best-fit models into photon treatment plan optimization on some independent head-and-neck patient. An analysis of the results concludes this paper.

2 Materials and Methods

In this chapter, we present the software framework and the underlying mathematical concepts, as well as the datasets used for model fitting and some treatment planning details.

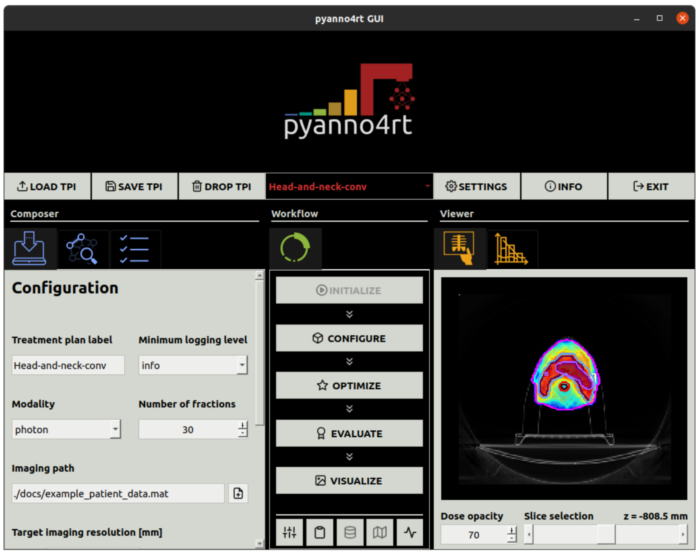
2.1 Software Framework

pyanno4rt is an open-source Python package for conventional and outcome prediction model-based inverse photon and proton treatment plan optimization. It encompasses an extensive and scalable number of optimization components, feature definitions, different solution methods and algorithms, plan evaluation and visualization tools, as well as a graphical user interface in addition to the code-based interface (see Figure 1). The package is under continuous development and currently being actively maintained. All results in this paper were generated from *pyanno4rt*.

2.2 Inverse Treatment Plan Optimization with Machine Learning Model-based Components

We now explain how *pyanno4rt* combines inverse planning and machine learning outcome prediction models.
First, we define the weighted-sum optimization problem for the conventional intensity-modulated treatment plan as

$$\begin{aligned} \min_{\mathbf{w}} \; & f(\mathbf{w}) = \sum_r \omega_r f_r(\mathbf{D}_r\mathbf{w}) \\ \text{s.t.} \; & c_m^l \leq c_m(\mathbf{w}) \leq c_m^u \,. \end{aligned} \tag{1}$$



Configuration

Treatment plan label: Head-and-neck-conv
Minimum logging level: info
Modality: photon
Number of fractions: 30
Imaging path: ./docs/example_patient_data.mat
Target imaging resolution [mm]
Dose opacity: 70
Slice selection
z = -808.5 mm

Figure 1: pyanno4rt GUI interface

with the fluence vector $\mathbf{w} \in \mathbb{R}_+^J$, the dose-influence matrix $\mathbf{D}_r \in \mathbb{R}_+^{I_r \times J}$, the objective function f_r and the weight ω_r for the segment r, as well as the (optional) constraints c_m.
The novelty of our approach lies in the interpretation of the machine learning outcome prediction functions as objectives or constraints (where we only show the objective case below). Throughout, we denote by $\mathbf{x} \in \mathbb{R}^p$ the p-dimensional feature vector, and by y the corresponding (binary) class label. Additionally, let n with index $i = 1, \dots, n$ be the number of samples. Under the assumption that $\mathbf{x}$ can be grouped into dosiomic ($\mathbf{x}^{dose}$), radiomic ($\mathbf{x}^{rad}$) and demographic features ($\mathbf{x}^{dem}$), all possibly with a transformation t, the generalized outcome prediction function becomes

$$(N)TCP_{ML} = p_\theta\left(t(\mathbf{x}^{dose}) | t(\mathbf{x}^{rad}), t(\mathbf{x}^{dem})\right). \tag{2}$$

In principle, $(N)TCP_{ML}$ could be used directly for optimization. However, it may show unfavorable mathematical properties, e.g. non-convexity or vanishing gradients. We therefore propose an equivalent formulation $f_{ML}(p_\theta)$ as objective and the backward gradient of f_{ML} with respect to $\mathbf{w}$ as

$$\nabla_{\mathbf{w}} f_{ML}(p_\theta) = \nabla_{\mathbf{w}} \mathbf{d}^\top \nabla_{\mathbf{d}} \mathbf{x}^{dose} \nabla_{\mathbf{x}^{dose}} t(\mathbf{x}^{dose}) \nabla_{t(\mathbf{x}^{dose})} f_{ML}(p_\theta). \tag{3}$$

pyanno4rt handles both reformulation of $(N)TCP_{ML}$ and differentiation of $f_{ML}(p_\theta)$ as well as the iterative recalculation of $\mathbf{x}^{dose}$ efficiently through JIT compilation and automatic differentiation using *JAX* [3], with the following machine learning model-based objective and gradient functions:

Logistic regression (LR)

Fitting the logistic regression model yields the coefficient vector $\beta \in \mathbb{R}^p$ and the outcome prediction function reads

$$(N)TCP_{LR} = p_\beta(y = 1 | t(\mathbf{x})) = \sigma\left(\beta^\top t(\mathbf{x})\right). \tag{4}$$

Inversion of the enclosing sigmoid function $\sigma(z) = (1 - \exp(-z))^{-1} \in (0,1)$ yields the objective

$$f_{LR}(p_\beta) = \beta^\top t(\mathbf{x}). \tag{5}$$

This function is convex in $t(\mathbf{x})$ with the constant gradient

$$\nabla_{t(\mathbf{x})} f_{LR}(p_\beta) = \beta. \tag{6}$$

Neural networks (NN)

Let $\mathbf{W}_l$ be the weight matrix, b_l the bias, and g_l the activation function in layer $l = 1, \dots, L$, the outcome prediction function can be described by the recursive rule

$$(N)TCP_{NN} = p_{(\mathbf{W},\mathbf{b})}(y = 1 | t(\mathbf{x})) = \mathbf{h}_L, \tag{7}$$

with the input $\mathbf{h}_0 = t(\mathbf{x})$ in the first layer and

$$\mathbf{h}_l = g_l\left(\mathbf{W}_l \mathbf{h}_{l-1} + b_l \cdot \mathbf{1}\right), \quad l = 1, \dots, L. \tag{8}$$

In the output layer L, we use the sigmoid activation function $g_L(z) = \sigma(z)$ for projecting onto the $(0,1)$ interval.
By the same reasoning as before, the objective after inverting the sigmoid activation function in the output layer reads

$$f_{NN}(p_{(\mathbf{W},\mathbf{b})}) = \mathbf{W}_L \mathbf{h}_{L-1} + b_L \cdot \mathbf{1}. \tag{9}$$

The gradient function can therefore be expressed as

$$\nabla_{t(\mathbf{x})} f_{NN}(p_{(\mathbf{W},\mathbf{b})}) = \nabla_{t(\mathbf{x})} \mathbf{h}_0 \prod_{l=1}^{L-1} \nabla_{\mathbf{h}_{l-1}} \mathbf{h}_l \nabla_{\mathbf{h}_{L-1}} f_{NN}(p_{(\mathbf{W},\mathbf{b})}). \tag{10}$$

Tensorflow's GradientTape API provides the gradient function via automatic input differentiation. We implemented both standard feedforward and input-convex neural networks [4, 5] in *pyanno4rt* to enable the comparison between the less restrictive but non-convex feedforward architecture and the more restrictive but input-convex architecture.

Support vector machines (SVM)

With the learnable dual parameters $\alpha \in \mathbb{R}^n$ and $b \in \mathbb{R}$, the outcome prediction function can be formulated as

$$(N)TCP_{SVM} = p_\alpha(y = 1 | t(\mathbf{x})) = \sigma_P\left(\sum \alpha_i K(t(\mathbf{x}_i), t(\mathbf{x})) + b\right), \tag{11}$$

where $\sigma_P(z) = (1 - \exp(-Az - B))^{-1}$ denotes the Platt scaling function with extra parameters $A, B \in \mathbb{R}$ [6] and K is a kernel. Table 1 shows the kernels implemented in *pyanno4rt*. By inverting the Platt scaling function, we end up with

$$f_{SVM}(p_\alpha) = \sum \alpha_i K\left(t(\mathbf{x}_i), t(\mathbf{x})\right) + b \tag{12}$$

as the objective, so that the gradient function is simply obtained by differentiation of the kernel, i.e.,

$$\nabla_{t(\mathbf{x})} f_{SVM}(p_\alpha) = \sum \alpha_i \nabla_{t(\mathbf{x})} K\left(t(\mathbf{x}_i), t(\mathbf{x})\right). \tag{13}$$

Kernel	Definition
linear	$K_{lin}(\mathbf{x}_i,\mathbf{x}) = \mathbf{x}^\top \mathbf{x}_i$
rbf	$K_{rbf}(\mathbf{x}_i,\mathbf{x}) = \exp(-\gamma \|\mathbf{x}-\mathbf{x}_i\|^2)$
poly	$K_{poly}(\mathbf{x}_i,\mathbf{x}) = (\gamma \mathbf{x}^\top \mathbf{x}_i + c_0)^p$
sigmoid	$K_{sig}(\mathbf{x}_i,\mathbf{x}) = \tanh(\gamma \mathbf{x}^\top \mathbf{x}_i + c_0)$

Table 1: Kernel functions for the SVM [7].

Kernel	Definition
linear	$\nabla_{\mathbf{x}} K_{lin}(\mathbf{x}_i,\mathbf{x}) = \mathbf{x}_i$
rbf	$\nabla_{\mathbf{x}} K_{rbf}(\mathbf{x}_i,\mathbf{x}) = -2\gamma(\mathbf{x}-\mathbf{x}_i) K_{rbf}(\mathbf{x}_i,\mathbf{x})$
poly	$\nabla_{\mathbf{x}} K_{poly}(\mathbf{x}_i,\mathbf{x}) = \gamma p \mathbf{x}_i (\gamma \mathbf{x}^\top \mathbf{x}_i + c_0)^{p-1}$
sigmoid	$\nabla_{\mathbf{x}} K_{sig}(\mathbf{x}_i,\mathbf{x}) = \gamma \mathbf{x}_i \cosh(\gamma \mathbf{x}^\top \mathbf{x}_i + c_0)^{-2}$

Table 2: Kernel gradients for the SVM [7].

Table 2 lists the kernel gradients implemented in *pyanno4rt*.

Putting everything together, we can expand the weighted-sum objective function from Equation 1 to yield

$$f(\mathbf{w}, p_\theta) = f(\mathbf{w}) + \sum_s \omega_s f_s(p_{\theta_s}), \tag{14}$$

where we add the machine learning model-based objectives f_s. Note that this approach also allows to (partially) substitute $f(\mathbf{w})$, or integrate outcome model-based constraints c_{ML}.

2.3 Datasets, Modeling and Treatment Planning

Machine learning modeling in this paper is based on two outcome data sets from a retrospective cohort of 153 head-and-neck patients treated with IMRT at Heidelberg University Hospital between the years 2010 and 2015.
The first data set includes features of the parotids to predict the probability of long-term grade 2+ xerostomia, and we refer to [1] for details on data structure and handling. To prevent multicollinearity, we focus on 6 dosiomic (mean, standard deviation, skewness, gradients) and 3 radiomic features (volume, sphericity, eccentricity) for each parotid.
The second data set has 14 dosiomic and radiomic features of the PTV to predict the probability of progression-free survival. From these, we removed 3 dosiomic features (maximum, minimum, quantile at 10% volume), which we found to cause over- or underdosing of the PTV after optimization.
Internal to *pyanno4rt*, the outcome models from Section 2.2 are fitted with *scikit-learn* [8] and *tensorflow* [9], and equipped with individual preprocessing, hyperparameter tuning, evaluation and inspection units, where we currently restrict the preprocessing to feature standardization and the hyperparameter tuning to sequential model-based global optimization using tree-structured Parzen estimators from *hyperopt* [10], modified by a robust "worst-case" variant of stratified k-fold cross-validation to prevent overfitting.

Segment	$\mathbf{d_{mean}}$			$\mathbf{d_{std}}$		
	S1	S2	S3	S1	S2	S3
Par. (L)	0.72	0.66	0.55	0.28	0.23	0.31
Par. (R)	0.70	0.61	0.22	0.28	0.22	0.21
PTV63	2.09	2.09	1.56	0.20	0.25	0.75
PTV70	2.30	2.31	2.30	0.10	0.12	0.10

Table 3: Dose mean and deviation for the scenarios S1–S3.

Segment	$\mathbf{d_{max}}$			$\mathbf{d_{min}}$		
	S1	S2	S3	S1	S2	S3
Par. (L)	1.66	1.52	1.87	0.24	0.33	0.16
Par. (R)	1.73	1.36	1.35	0.15	0.14	0.01
PTV63	2.48	2.78	3.07	0.25	0.24	0.11
PTV70	2.48	2.66	2.51	0.46	0.46	0.47

Table 4: Dose maximum and minimum for the scenarios S1–S3.

Treatment planning relies on an independent head-and-neck patient from the CORT dataset [11]. The segments of interest are left/right parotid, skin, PTV63 and PTV70. To evaluate the impact of machine learning outcome models for optimization, we investigate three optimization scenarios:

(**S1**) Dose objectives only (squared overdosing on left/right parotid and skin, squared deviation on PTV63/PTV70)

(**S2**) S1 with additional machine learning model-based objectives on left/right parotid and PTV63

(**S3**) S2 with only machine learning model-based objectives on left/right parotid and PTV63

Moreover, we opt for photon irradiation with 30 fractions and the L-BFGS-B algorithm from *scipy* [12] for optimization.

3 Results

In this chapter, we report the most important findings on the resulting dose distributions and outcome predictions.

3.1 Dose Distributions

The optimal 2D dose distributions are shown in Figure 2, with the corresponding dosimetrics in Table 3 and 4. S1 achieves good coverage of the PTV63/PTV70 and sparing on the left/right parotid. Adding the best-fit models, the LR models, in S2 (center) implies a minor shift of dose away from the parotids and more pronounced hotspots in the PTVs. The impact of the LR models is most evident for S3, where the heterogeneity of the dose distribution in the PTV63 increases, while the dose distribution in the PTV70 compares well with S1. Due to the overall lower dose in the PTV63, there is also a larger dose reduction in the parotids. We attribute the changes in dose to the most contributing features, i.e., the mean dose in the parotids or the median dose in the PTV63.

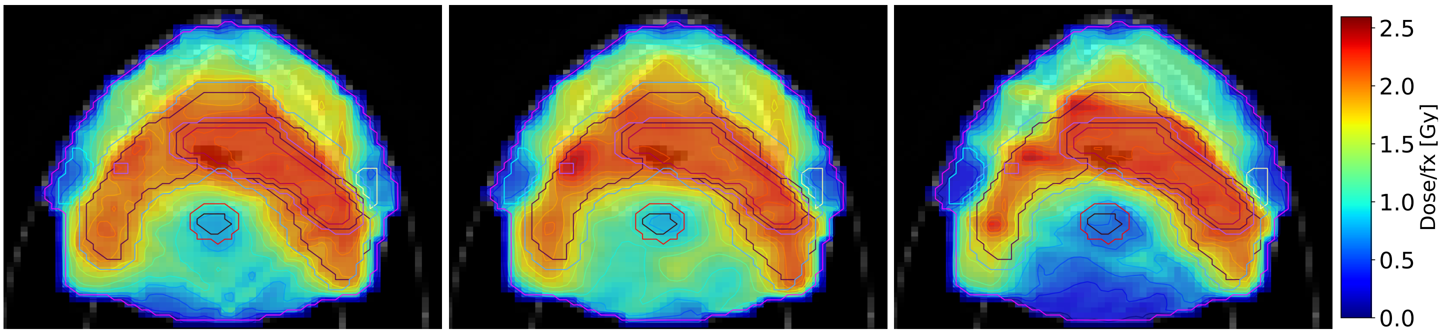


Figure 2: Optimal 2D dose distributions for the scenarios S1–S3 (from left to right).

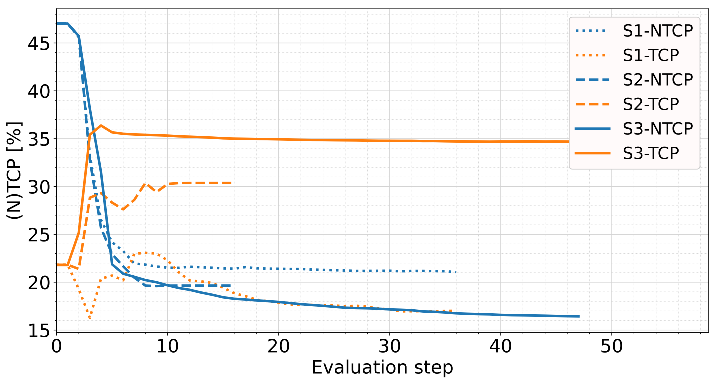


Figure 3: (N)TCP curves for the scenarios S1–S3. Our optimization approach allows for improved outcome predictions, even though the range of the outcome values seems fallacious.

3.2 Outcome Predictions

Figure 3 displays the (N)TCP curves for the three planning scenarios S1–S3. We observe both an increase in TCP (S1: 17%, S2: 30.4%, S3: 34.7%) and a decrease in NTCP (S1: 21%, S2: 19.7%, S3: 16.3%) as the LR models become more influential. This meets our expectation that, on the one hand, the mere inclusion of the (N)TCP objectives in S2 should already improve the outcome predictions, and on the other hand, that the removal of the parallel dose objectives in S3 resolves some trade-offs, allowing the LR outcome models to shape the dose distribution in a less restricted way.

4 Discussion

pyanno4rt shows the ability to optimize machine learning outcome model-based treatment plans with low additional overhead and generate appealing dose distributions with improved outcome predictions. We are also able to (partially) interpret some effects of the fitted outcome models on the optimal dose distribution. However, there is still a lack of sufficient data and high-quality prediction models, which can be observed from the fact that the (N)TCP values from Figure 3 are in a clinically unacceptable range. In addition to that, convexity of the derived (N)TCP objectives only holds under conditions, with no absolute guarantees for a gradient-based algorithm to find a reasonably good optimal point.

5 Conclusion

pyanno4rt is an open-source toolkit for conventional and outcome prediction model-based inverse planning, written purely in Python. It efficiently implements machine learning model-based treatment plan optimization by utilizing the outcome prediction functions. This has a great potential to support the future use of machine learning models in IMRT.

References

[1] H. S. Gabryś, F. Buettner, F. Sterzing, et al. “Design and Selection of Machine Learning Methods Using Radiomics and Dosiomics for Normal Tissue Complication Probability Modeling of Xerostomia”. *Frontiers in Oncology* 8 (2018), p. 35.

[2] T. Ortkamp, O. Jäkel, M. Frank, et al. *pyanno4rt: python-based advanced numerical nonlinear optimization for radiotherapy*. http://github.com/pyanno4rt/pyanno4rt. Version 0.0.1. 2024.

[3] J. Bradbury, R. Frostig, P. Hawkins, et al. *JAX: composable transformations of Python+NumPy programs*. http://github.com/google/jax. Version 0.3.13. 2018.

[4] B. Amos, L. Xu, and J. Z. Kolter. *Input Convex Neural Networks*. 2017.

[5] S. Sivaprasad, A. Singh, N. Manwani, et al. *The Curious Case of Convex Neural Networks*. 2021.

[6] J. Platt. “Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods”. *Adv. Large Margin Classif.* 10 (June 2000).

[7] J. E. Johnson, V. Laparra, A. Pérez-Suay, et al. “Kernel methods and their derivatives: Concept and perspectives for the earth system sciences”. *PLoS ONE* 15.10 (2020).

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. “Scikit-learn: Machine Learning in Python”. *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] Martín Abadi, Ashish Agarwal, Paul Barham, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. https://www.tensorflow.org/. Software available from tensorflow.org. 2015.

[10] J. Bergstra, R. Bardenet, Y. Bengio, et al. “Algorithms for hyper-parameter optimization”. *Advances in neural information processing systems* 24 (2011).

[11] D. Craft, M. Bangert, T. Long, et al. “Shared Data for Intensity Modulated Radiation Therapy (IMRT) Optimization Research: The CORT Dataset”. *GigaScience* 3.1 (2014).

[12] P. Virtanen, R. Gommers, T. E. Oliphant, et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. *Nature Methods* 17 (2020), pp. 261–272.