Supervision on W versus H

Classical nonnegative matrix factorization (NMF) is non-identifiable: for any invertible matrix R with nonnegative entries, the factorization (W, H) can be transformed to $(WR, R^{-1}H)$ without changing the reconstruction error, yielding multiple valid solutions [@donoho2004nmf; @laurberg2008uniqueness; @gillis2014nmf]. Although normalizing columns of W or rows of H resolves the trivial scaling ambiguity, nonnegative mixing transformations persist, and empirical studies consistently show that the sample-loading matrix H is more sensitive to initialization and local minima than the program matrix W [@brunet2004metagenes; @kim2007sparse; @gillis2012accelerated].

In DeSurv, survival supervision enters through the projection

$$Z = X^\top W, \qquad \ell_{\text{Cox}} = \ell_{\text{Cox}}(Z),$$

so that the supervised gradient with respect to the programs is

$$\frac{\partial \ell_{\text{Cox}}}{\partial W} = X \, \frac{\partial \ell_{\text{Cox}}}{\partial Z},$$

which directly reshapes the gene-program definitions. This mechanism amplifies genes aligned with the hazard gradient and suppresses those that dilute prognostic structure, ensuring that the learned programs encode survival-relevant biological variation.

By contrast, if supervision were applied to the sample loadings H, the model could reduce the Cox loss by redistributing patient-specific coefficients while leaving the gene-level programs W nearly unchanged, a behavior documented in multiple supervised variants of NMF and supervised topic models, where supervision applied only to the coefficient matrix modifies H but not the basis W [@cai2011graph; @wang2014supervised; @blei2007slda].

Supervising W also provides a practical advantage for **portability**: because W defines gene-level programs, new samples can be embedded by the closed-form projection $Z_{\text{new}} = X_{\text{new}}^\top W$. In contrast, supervising H would not yield such a mapping; instead, obtaining sample loadings for external datasets would require solving a nonnegative least-squares (NNLS) problem for each new sample, introducing optimization noise and reducing reproducibility.

Together, these considerations motivate supervising through W, which shapes the gene programs toward prognostic directions, stabilizes solutions across initializations, and yields biologically interpretable signatures that generalize across cohorts.

Optimization details

Algorithm~S1 describes the block coordinate descent (BCD) algorithm for optimizing the DeSurv model in Equation~**??**.

Algorithm 1 DeSurv block coordinate descent

Input: $X \in \mathbb{R}_{\geq 0}^{p \times n}$, survival times $y \in \mathbb{R}_{\geq 0}^{n}$, event indicators $\delta \in \{0,1\}^n$, tolerance tol, maximum iterations $maxit$, supervision parameter α, rank k, penalties λ_H, λ, and ξ

Output: Fitted DeSurv parameters (W, H, β)

Initialize $W^{(0)}, H^{(0)}$ with positive entries (e.g., $W^{(0)}, H^{(0)} \sim \text{Uniform}(0, \max X)$)
Initialize $\beta^{(0)}$ (e.g., $\beta^{(0)} = \mathbf{1}$)
$loss = \mathcal{L}(W^{(0)}, H^{(0)}, \beta^{(0)})$
$eps = \infty$, $t = 0$
while $eps \geq tol$ **and** $t < maxit$ **do**
    (H-update)
        Update $H^{(t+1)}$ from $H^{(t)}$ using the multiplicative rule in Eq. **??**,
        holding $W^{(t)}$ and $\beta^{(t)}$ fixed.
    (W-update)
        Update $W^{(t+1)}$ from $W^{(t)}$ using the hybrid multiplicative rule in Eq. **??**,
        holding $H^{(t+1)}$ and $\beta^{(t)}$ fixed, and perform Armijo backtracking to ensure
        $\mathcal{L}$ does not increase.
    (Column normalization)
        Compute $D = \text{diag}\big(\|W_{:,1}^{(t+1)}\|_2, \ldots, \|W_{:,k}^{(t+1)}\|_2\big)$
        and set $W^{(t+1)} \leftarrow W^{(t+1)} D^{-1}$, $H^{(t+1)} \leftarrow D H^{(t+1)}$, and $\beta^{(t)} \leftarrow D\beta^{(t)}$
    (β-update)
        Update $\beta^{(t+1)}$ from $\beta^{(t)}$ using a Newton-like step for the Cox loss
        (Eq. **??**), holding $W^{(t+1)}$ and $H^{(t+1)}$ fixed.
    $lossNew = \mathcal{L}(W^{(t+1)}, H^{(t+1)}, \beta^{(t+1)})$
    $eps = |lossNew - loss|/|loss|$
    $loss = lossNew$
    $t = t + 1$
end while
return $\hat{W} = W^{(t+1)}, \hat{H} = H^{(t+1)}, \hat{\beta} = \beta^{(t+1)}$

Update for H

We adopt the standard multiplicative update for NMF @Lee1999,

$$H \leftarrow H \odot \frac{W^\top X}{W^\top W H}, \tag{1}$$

which guarantees nonnegativity and monotonic decrease in the reconstruction error conditional on W.

Update for W

The projected-gradient update derived above is evaluated with an ϵ-floor in the denominator to avoid division by zero. After each step we rescale W to keep the columns on comparable magnitude, which improves numerical stability.

$$W^{(t+1)} = W^{(t)} \odot \frac{\frac{(1-\alpha)}{np} X H^T + \delta^{(t)} \nabla_W \ell(W^{(t)}, \beta)}{\frac{(1-\alpha)}{np} W^{(t)} H H^T} \tag{2}$$

where

$$\delta^{(t)} = \frac{||\frac{(1-\alpha)}{np} \nabla_W L_{NMF}(W^{(t)}, H)||_F^2}{||\alpha \nabla_W L_{cox}(W^{(t)}, \beta)||_F^2} = \frac{\frac{(1-\alpha)}{np} ||\nabla_W L_{NMF}(W^{(t)}, H)||_F^2}{\alpha ||\nabla_W \ell(W^{(t)}, \beta)||_F^2} \tag{3}$$

balances the contribution of the NMF and Cox terms to the update. As in the H update, a small ϵ is added to the denominator and an ϵ-floor is applied.

The quantity $\nabla_W \ell(W^{(t)}, \beta)$ denotes the gradient of the Cox partial log-likelihood with respect to W evaluated at $W^{(t)}$

$$\nabla_W \ell(W^{(t)}, \beta) = \frac{2}{n_{event}} \sum_{i=1}^{n} \delta_i \left(x_i - \frac{\sum_{y_j \geq y_i} x_j e^{x_j^T W^{(t)} \beta}}{\sum_{y_j \geq y_i} e^{x_j^T W^{(t)} \beta}} \right) \beta^T, \tag{4}$$

and the quantity $\nabla \mathcal{L}_{NMF}(W^{(t)}, H)$ is the gradient of the reconstruction error in Equation~**??** with respect to W evaluated at $W^{(t)}$

$$\nabla \mathcal{L}_{NMF}(W^{(t)}, H) = 2(W^{(t)} H - X) H^T. \tag{5}$$

Update for β

Conditional on (W, H) we solve the elastic-net penalized Cox model using coordinate descent. The soft-thresholding operator $S(\cdot, \lambda\xi)$ yields closed-form updates for each coefficient, and convergence is declared when the relative change in the penalized likelihood falls below 10^{-6}.

Proof of Convergence to a Stationary Point

To prove convergence of Algorithm~S1 to a stationary point of the DeSurv model defined in Equation~**??**, we rely on the theory of @tseng2001convergence, which states:

Derivation of W update

The W update can be derived directly from the projected coordinate descent update with a specific step size.

Recall that the overall loss function is

$$\mathcal{L}(W, H, \beta) = \frac{(1-\alpha)}{2np} ||X - WH||_F^2 - \frac{2\alpha}{n_{event}} \ell(W, \beta) + \lambda(\xi ||\beta||_1 + \frac{(1-\xi)}{2}, \tag{6}$$

where $\ell(W, \beta)$ is the log-partial likelihood for the Cox model. Let $\nabla_W \ell$ represent the derivative of $\ell(W, \beta)$ with respect to W. Then the derivative of the overall loss with respect to W is

$$\frac{\partial L}{\partial W} = \frac{(1-\alpha)}{np} \left(WHH^T - XH^T \right) - \frac{2\alpha}{n_{event}} \nabla_W \ell \tag{7}$$

Then the gradient descent update rule at iteration t is

$$\begin{aligned} W^{(t)} &= W^{(t-1)} - \gamma \left(\frac{\partial L}{\partial W} \right) \\ &= W^{(t-1)} - \gamma \left(\frac{(1-\alpha)}{np} \left(WHH^T - XH^T \right) - \frac{2\alpha}{n_{event}} \nabla_W \ell \right) \end{aligned} \tag{8}$$

$$\tag{9}$$

Let the step size γ be defined as in CITE:

$$\gamma = \frac{np}{(1-\alpha)} \frac{W}{WHH^T} \tag{10}$$

Then the update becomes

$$
\begin{aligned}
W^{(t)} &= W^{(t-1)} - \frac{np}{(1-\alpha)} \frac{W}{WHH^T} \left(\frac{(1-\alpha)}{np} \left(WHH^T - XH^T \right) - \frac{2\alpha}{n_{event}} \nabla_W \ell \right) \\
&= \frac{W}{WHH^T} XH^T + \frac{2\alpha np}{n_{event}(1-\alpha)} \nabla_W \ell \\
&= W \odot \frac{XH^T + \frac{2\alpha np}{n_{event}(1-\alpha)} \nabla_W \ell}{WHH^T} \\
&= W \odot \frac{\frac{(1-\alpha)}{np} XH^T + \frac{2\alpha}{n_{event}} \nabla_W \ell}{\frac{(1-\alpha)}{np} WHH^T}
\end{aligned} \tag{11}
$$

Finally, projected coordinate descent projects the W update in to the positive space

$$
W^{(t)} = \max \left(W \odot \frac{\frac{(1-\alpha)}{np} XH^T + \frac{2\alpha}{n_{event}} \nabla_W \ell}{\frac{(1-\alpha)}{np} WHH^T}, 0 \right) \tag{12}
$$

Since $W \geq 0$ this is equivalent to

$$
W^{(t)} = W \odot \max \left(\frac{\frac{(1-\alpha)}{np} XH^T + \frac{2\alpha}{n_{event}} \nabla_W \ell}{\frac{(1-\alpha)}{np} WHH^T}, 0 \right) \tag{13}
$$

which matches the multiplicative form used in the software implementation.

Cross-validation procedure

Algorithm~S2 describes the cross validation procedure with F folds and R initializations.

Algorithm 2 Cross-validation for DeSurv pipeline

Input:
Number of folds F
Number of initializations R
Observed data (X, y, δ)
Hyperparameters k, α, λ, ξ, and λ_H
Output: The cross-validated c-index

Divide subjects into F folds.
for $f = 1$ to F **do**
  Split data into training and validation $(X_{(-f)}, y_{(-f)}, \delta_{(-f)})$, and $(X_{(f)}, y_{(f)}, \delta_{(f)})$ sets
  for $r = 1$ to R **do**
    set.seed(r)
    Apply Algorithm 1 with inputs $(X_{(-f)}, y_{(-f)}, \delta_{(-f)})$, and $(k, \alpha, \lambda, \xi, \lambda_H)$
    Obtain $\hat{W}$ and $\hat{\beta}$ as output from Algorithm 1
    Compute the estimated linear predictor: $\hat{\eta} = X_{(f)}^T \hat{W} \hat{\beta}$
    Compute c-index, denoted $\hat{c}_{(f)r}$, according to Equation **??**
  end for
  end loop over r
  Compute average c-index across initializations: $\hat{c}_{(f)} = \frac{1}{r} \sum_{r=1}^{R} \hat{c}_{(f)r}$
end for
end loop over f
Compute final cross-validated c-index: $\hat{c} = \frac{1}{F} \sum_{f=1}^{F} \hat{c}_{(f)}$
return Final estimate $\hat{c}$

Bayesian optimization roadmap

While the current study relies on exhaustive grid search, the calibration problem is well suited to Bayesian Optimization (BO). In future work we plan to model the mapping from hyperparameters (k,
$alpha$,
$lambda$,
xi) to the cross-validated c-index using a Gaussian Process surrogate, \[f(\lambda) \sim \mathcal{GP}(\mu(\lambda), k(\lambda, \lambda')), \] and select new evaluations via the Expected Improvement acquisition function. Each BO iteration would:

\begin{enumerate} \item Fit/refresh the surrogate using all previously evaluated configurations and their observed c-indices. \item Maximize the acquisition function to identify the next configuration
$lambda_{t+1}$. \item Launch the cross-validation target for
$lambda_{t+1}$ within the `targets` pipeline. \end{enumerate}

This strategy reduces the required number of expensive cross-validation runs while retaining the ability to handle mixed discrete and continuous hyperparameters.