Prevalidated ridge regression is a highly-efficient drop-in replacement for logistic regression for high-dimensional data

Daniel F. Schmidt
Work done with Angus Dempster and Geoffrey I Webb

Department of Data Science and AI
Monash University

National Technical University of Athens
March 30, 2024

Outline

Outline

Motivation

- I am working in the field of scalable time series classication
- One of the key methods is based on random features
    - Many times faster than competing methods of similar performance
- Features combined using “ridge regression classifier”
    - Very fast to fit
    - Does not produce probabilistic predictions
- Aim: produce probabilistic predictions from ridge classifier without affecting time complexity

Linear Classifiers

- Classification problem:
  - Targets $\mathbf{y} \in \{1, \ldots, L\}^n$
  - Feature matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$
- The aim is to predict which class an individual belongs to based on their features
- Linear classifiers are classification systems based around linear predictors of the form
$$\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$$
which are weighted linear combinations of the features
- Linear classifiers are popular due to several reasons
  - They are (relatively) simple to understand and fit
  - They suffer less from curse of dimensionality
  - With suitable features, they can perform very well
- Final layer of most deep neural networks is a linear classifier

Multinomial Regression (1)

- The standard linear classifier is the multinomial logistic regression
- Let $\beta^{(i)}$ denote the coefficients associated with class i; let

$$\boldsymbol{\eta}^{(i)} = \mathbf{X}\boldsymbol{\beta}^{(i)}$$

  be the linear predictor associated with class i, and form

$$\mathbf{H}(\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(L)}) = (\boldsymbol{\eta}^{(1)}, \ldots, \boldsymbol{\eta}^{(L)})$$

- Under the multinomial logistic regression model (i.e., "soft-max")

$$\mathbb{P}(Y_i = j) = \frac{e^{\eta_i^{(j)}}}{\sum_k e^{\eta_i^{(k)}}}$$

- The negative log-likelihood of $\mathbf{y}$, given $\mathbf{H}$ is then

$$l(\mathbf{H}) = \sum_{i=1}^{n} \left[-\eta_i^{(y_i)} + \log \sum_{j=1}^{L} \exp \eta_i^{(j)} \right]$$

Multinomial Regression (2)

- The negative log-likelihood of $\mathbf{y}$, given $\mathbf{H}$ is then

$$l(\mathbf{H}) = \sum_{i=1}^{n} \left[-\eta_i^{(y_i)} + \log \sum_{j=1}^{L} \exp \eta_i^{(j)} \right]$$

- Fitting can be done by minimising the negative log-likelihood

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}^{(1)},\ldots,\boldsymbol{\beta}^{(L)}}{\arg\max} \left\{ l(\mathbf{H}(\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(L)})) \right\}$$

$\Longrightarrow$ the maximum likelihood estimate

- This has several advantages
  - Convex optimisation problem $\Rightarrow$ one minima
  - Asymptotically (statistically) efficient
- However, for $n < p$, no minimiser exists
  - If $n \approx p$, the estimates can have very high variance

Multinomial Regression (3)

- These problems can be solved/moderated via regularisation
- We will consider ℓ_2 penalised maximum likelihood

$$\hat{\boldsymbol{\beta}}_\lambda = \underset{\boldsymbol{\beta}^{(1)},\ldots,\boldsymbol{\beta}^{(L)}}{\arg\max} \left\{ l(\mathbf{H}(\boldsymbol{\beta}^{(1)},\ldots,\boldsymbol{\beta}^{(L)})) + \lambda \sum_{j=1}^{L} ||\boldsymbol{\beta}^{(j)}||^2 \right\}$$

- Works well if most features are weakly associated with the targets
- Problem remains convex, but now we must choose λ
  - Frequently done via K-fold cross validation

Multinomial Regression (4)

- There is no closed-form solution to the penalized ML problem $\Longrightarrow$ we must find the estimates numerically
- Standard algorithms are
  - Iteratively re-weighted least squares/Fisher scoring (scikit-learn)
  - (Stochastic) gradient descent
- This is potentially difficult/slow for large p
  - Large numbers of parameters to optimise
  - Correlation between weight vectors affects convergence speed for $L > 2$
- Additionally, if we use K-fold CV to choose λ we must run the algorithm $K \times R$ times, where $R > 1$ is the number of candidate λ values, i.e., $\lambda \in \{\lambda_1, \ldots, \lambda_R\}$
- The final model can be sensitive to the particular K-split
  - Can be moderated by multiple CV repetitions, but makes it even slower

Outline

Ridge Regression Classifier (1)

- If we only want class predictions, we can speed the whole process up by using approximate solutions based on penalised least-squares
- To use squared error to fit a classifier we need recode the targets
- We make L new column vectors $\tilde{\mathbf{y}}^{(j)}$, one for each class with entries

$$\tilde{y}_i^{(j)} = \begin{cases} 1 & \text{if } y_i = j, \\ -1 & \text{otherwise} \end{cases}$$

- Example:

$$\underbrace{\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ \vdots \end{pmatrix}}_{\mathbf{y}} \Longrightarrow \underbrace{\begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \\ & & \vdots & \end{pmatrix}}_{\tilde{\mathbf{Y}}}$$

Ridge Regression Classifier (2)

- We can then fit L seperate ridge regressions to these new targets

$$\hat{\boldsymbol{\beta}}^{(j)} = \arg\min_{\boldsymbol{\beta}} \left\{ ||\tilde{\mathbf{y}}^{(j)} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda_j ||\boldsymbol{\beta}||^2 \right\}$$

  which have closed form solutions

$$\hat{\boldsymbol{\beta}}^{(j)} = \left(\mathbf{X}^T \mathbf{X} + \lambda_j \mathbf{I}_p \right)^{-1} \mathbf{X}^T \tilde{\mathbf{y}}^{(j)}$$

- The regularisation parameter λ_j can be found efficiently via LOOCV
  - This can be done using the clever “shortcut trick”
- Given a feature vector $\mathbf{x} = (x_1, \ldots, x_p)$, we can classify it using

$$c = \arg\max_{j} \left\{ \mathbf{x}^T \hat{\boldsymbol{\beta}}^{(j)} \right\}$$

Ridge Regression Classifier (3)

- This unfortunately only provides predictions of classes, and not estimates of probabilities
- Why are (accurate) probabilistic predictions useful?
- Consider two feature vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ and a probabilistic classifier
  1. $\mathbb{P}(Y = 1 \,|\, \mathbf{x}_1) = 0.45$ and $\mathbb{P}(Y = 2 \,|\, \mathbf{x}_1) = 0.55$
  2. $\mathbb{P}(Y = 1 \,|\, \mathbf{x}_2) = 0.05$ and $\mathbb{P}(Y = 2 \,|\, \mathbf{x}_2) = 0.95$
- Both would classify the individuals as class 2
  - But we are much more confident of that classification in case 2

  $\Longrightarrow$ non-probabilistic classifiers lack this information
- If we had information on the losses (monetary, of otherwise) incurred by incorrect classification, we can use this to choose classification with minimum expected loss
  - E.g., self-driving car classifying objects as people or not

Ridge Regression Classifier (4)

- We could try and post-hoc create probabilities for ridge regression
- We might try to calibrate this model directly; form

$$\hat{\mathbf{H}} = \left(\mathbf{X}\hat{\boldsymbol{\beta}}^{(1)}, \ldots, \mathbf{X}\hat{\boldsymbol{\beta}}^{(L)}\right)$$

  and pass it through the logistic transformation

$$\hat{\kappa} = \arg\min_{\kappa} l(\kappa \cdot \hat{\mathbf{H}})$$

  where κ is a scaling factor to "calibrate" the linear predictor
  - This is a form of restricted maximum likelihood estimation
- If $n \gg p$, this can work reasonably well
- In this case, the data will not be perfectly linearly separable $\Longrightarrow$ the likelihood provides information for estimating κ

Ridge Regression Classifier (5)

- Unfortunately, if p is large this does not work well
  - This is a setting in which linear classifiers are highly useful
  - Therefore it is important to make sure it works in these settings
- In this case, the ridge predictors $\hat{\mathbf{H}}$ tend to fit the training data very closely (potentially even perfectly) in the sense that

$$\arg\max_{j}\{\hat{\boldsymbol{\eta}}_{i,j}\} = y_i$$

- Due to the regularisation they may still predict the class of future data well (though usually not perfectly)
  $\Longrightarrow$ an example of "benign overfitting"?
- In this case $l(\kappa \cdot \hat{\mathbf{H}})$ is maximised by taking $\kappa = \infty$
- The model then assigns (near) zero or one probabilities to all instances
  $\Longrightarrow$ log-loss on future data is extremely high (near infinite)

Outline

Prevalidated Ridge Regression Classifier (1)

- The problem is that the regularisation is done with respect to the squared error fits, not the log-loss
- The subsequent calibration/scaling step acts as if the plug-in predictions are known exactly
  - but they are not, they are just estimates
  - and as such they have a degree of variability/uncertainty about them
  - this uncertainty/variability can be very large if $p \gg n$
  - we are using the same data to calibrate that we used to fit the classifier
- To fix this, we can somehow account for this uncertainty
  - Ideally then we can choose λ based on log-loss, not squared-error
- There are multiple possible ways of accounting for uncertainty
  - Bootstrapping
  - Some sort of Bayesian sampling/averaging
- We chose to use LOOCV
  - Related to the jacknife (linearised bootstrap)
  - Has computational advantages as well

Prevalidated Ridge Regression Classifier (1)

- The problem is that the regularisation is done with respect to the squared error fits, not the log-loss
- The subsequent calibration/scaling step acts as if the plug-in predictions are known exactly
  - but they are not, they are just estimates
  - and as such they have a degree of variability/uncertainty about them
  - this uncertainty/variability can be very large if $p \gg n$
  - we are using the same data to calibrate that we used to fit the classifier
- To fix this, we can somehow account for this uncertainty
  - Ideally then we can choose λ based on log-loss, not squared-error
- There are multiple possible ways of accounting for uncertainty
  - Bootstrapping
  - Some sort of Bayesian sampling/averaging
- We chose to use LOOCV
  - Related to the jacknife (linearised bootstrap)
  - Has computational advantages as well

Prevalidated Ridge Regression Classifier (2)

- Solution: we replace $\hat{\mathbf{H}}$ with the LOO cross validated predictions
- Let

$$\hat{\boldsymbol{\beta}}_{-i}^{(j)}(\lambda)$$

  denote the ridge regression solutions for class j and regularisation parameter λ when we drop observation i from our data
- Then, for a given λ, we can form the LOO linear predictions

$$\tilde{\eta}_i^{(j)} = \mathbf{x}_i^T \hat{\boldsymbol{\beta}}_{-i}^{(j)}(\lambda)$$

  into a matrix $\tilde{\mathbf{H}}_\lambda$
  $\Longrightarrow$ we call these pre-validated predictions
- To connect back to usual ridge-regression, the quantity

$$\sum_{i=1}^{n} \left(\tilde{\eta}_i^{(j)} - \tilde{y}_i^{(j)} \right)^2$$

  would be the LOOCV score that we would optimise w.r.t. λ

Prevalidated Ridge Regression Classifier (3)

- We then calibrate these predictions by solving

$$\hat{\kappa}, \hat{\lambda} = \arg\min_{\kappa,\lambda} l(\kappa \cdot \tilde{\mathbf{H}}_\lambda)$$

- What does this achieve?
  - The matrix $\tilde{\mathbf{H}}_\lambda$ is a perturbed version of the predictions $\hat{\mathbf{H}}_\lambda$, i.e., it no longer provide perfect fits to the training data
    $\Rightarrow$ there is now information for estimating κ
  - The regularisation parameter λ is now chosen with respect to overall log-loss, rather than the squared-error of each of the regressions

- A type of restricted maximum likelihood estimation
  - "Parameter space" is restricted to scalar multiples of the squared-error ridge solutions, i.e., to scalar multiples of

$$\left(\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I}_p\right)^{-1} \mathbf{X}^T \mathbf{y}^{(j)}$$

- We are adjusting a single parameter κ numerically, not $L \times p$ parameters as in the case of full maximum likelihood

Implementation Overview

- A key aspect of this process is that it is fast
  - Requires a number of linear algebra tricks
- Let $m = \max(n, p)$ and $r = \min(n, p)$
  - The number of classes $L \ll r < m$
- Key ideas underlying implementation
  - We note that each linear predictor uses the same feature matrix $\mathbf{X}$
  - If we perform an SVD on this matrix (time complexity $O(mr^2)$, we perform dimensionality reduction and orthogonalisation
  - The ridge solutions for a λ can be computed in $O(Lr)$ time given an orthogonal design
  - Using the LOOCV "shortcut", all the LOOCV predictions can be computed in $O(Lr^2)$ time for an orthogonal design

$\Longrightarrow$ Overall time complexity no greater than a single ridge fit

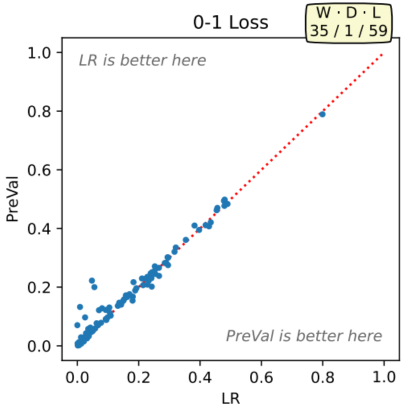
Tests: PreVal vs Maximum Likelihood

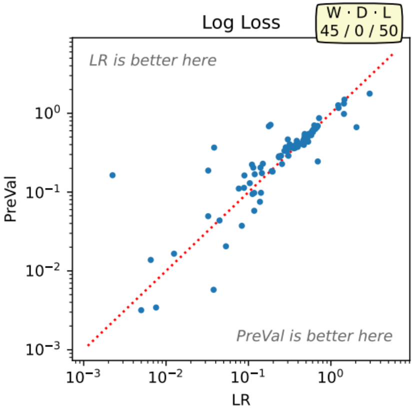
- Tabular data: 95 datasets from OpenML benchmarks
  - Large n (500 to 940,000), small p (4 to 500)
  - Including 2nd-order interactions (up to 4GB)
- Microarray data: 72 datasets from CuMiDa benchmark
  - Small n (12 to 357), large p (12,000 to 55,000)
- Time series classification: 106 datasets from UCR
  - Restricted to datasets with ≥ 5 examples per class
  - Used $10,000$ MiniRocket features
- Then fitted linear classifiers using
  - calibrated pre-validated ridge regression predictions, i.e. PreVal
  - ℓ_2-regularised maximum likelihood with 5-fold CV
- For the latter we used the `scikit-learn` implementation
- For the former, we used our `preval` package
- Evaluated on 0-1 loss, log-loss and wall-clock time

Tests: PreVal vs Maximum Likelihood

- Tabular data: 95 datasets from OpenML benchmarks
  - Large n (500 to 940,000), small p (4 to 500)
  - Including 2nd-order interactions (up to 4GB)
- Microarray data: 72 datasets from CuMiDa benchmark
  - Small n (12 to 357), large p (12,000 to 55,000)
- Time series classification: 106 datasets from UCR
  - Restricted to datasets with ≥ 5 examples per class
  - Used $10,000$ MiniRocket features
- Then fitted linear classifiers using
  - calibrated pre-validated ridge regression predictions, i.e. PreVal
  - ℓ_2-regularised maximum likelihood with 5-fold CV
- For the latter we used the `scikit-learn` implementation
- For the former, we used our `preval` package
- Evaluated on 0-1 loss, log-loss and wall-clock time

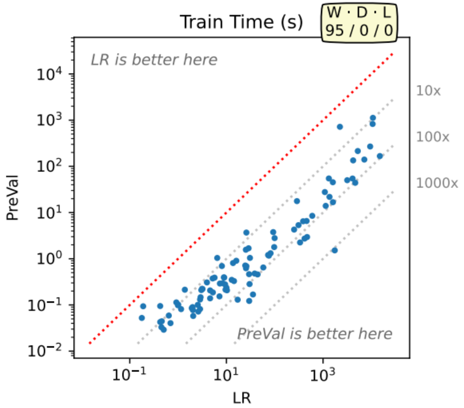
Results: Tabular Data (1)





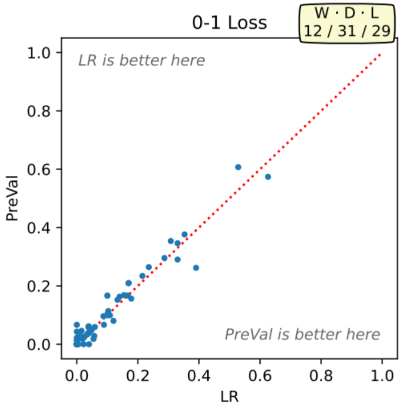
Pairwise 0-1 and log-loss for PreVal vs logistic regression (tabular data)
Logistic regression was trained using scikit-learn (5-fold CV)

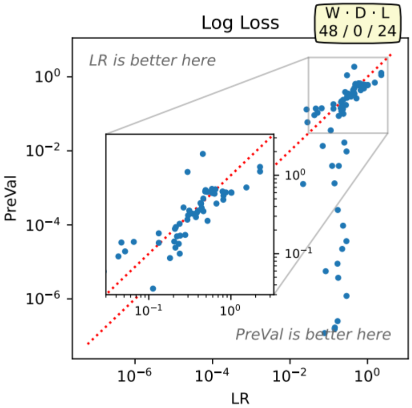
Results: Tabular Data (2)



Timing for PreVal vs logistic regression (tabular data)
Logistic regression was trained using scikit-learn (5-fold CV)

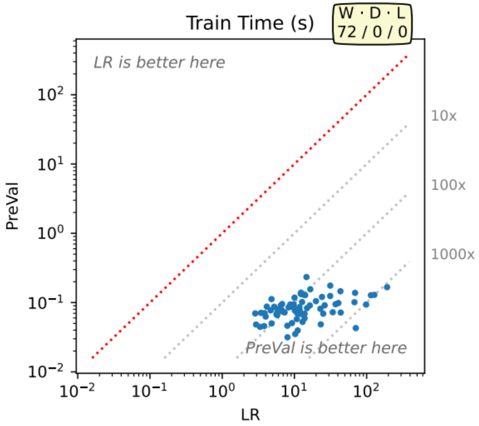
Results: Microarray Data (1)





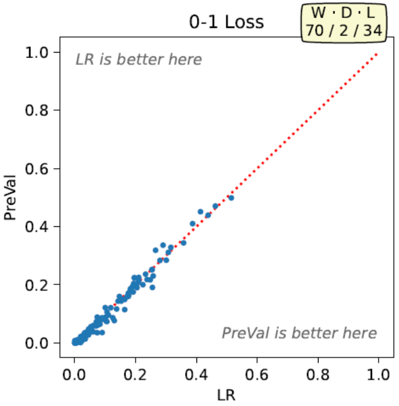
Pairwise 0-1 and log-loss for PreVal vs logistic regression (microarray data)
Logistic regression was trained using scikit-learn (5-fold CV)

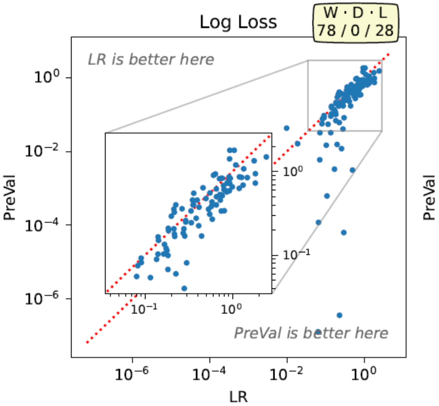
Results: Microarray Data (2)



Timing for PreVal vs logistic regression (microarray data)
Logistic regression was trained using scikit-learn (5-fold CV)

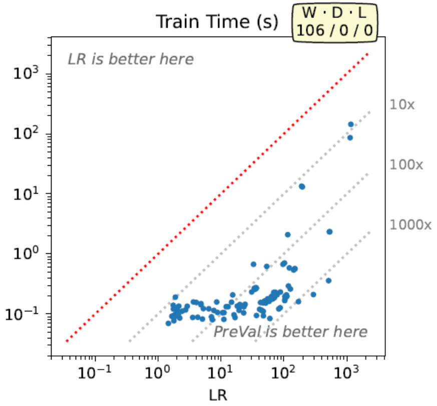
Results: Time Series Data (1)





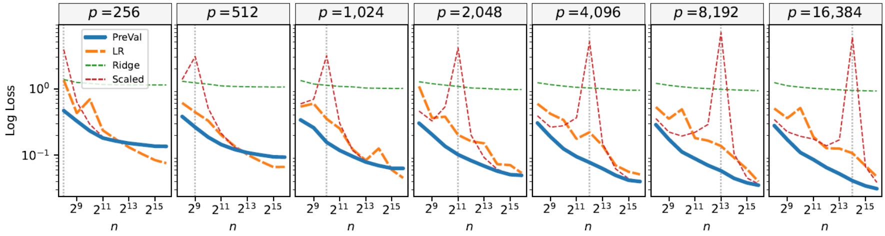
Pairwise 0-1 and log-loss for PreVal vs logistic regression (UCR data)
Logistic regression was trained using scikit-learn (5-fold CV)

Results: Time Series Data (2)



Timing for PreVal vs logistic regression (UCR data)
Logistic regression was trained using scikit-learn (5-fold CV)

Results: MNIST Learning Curves



Learning curves for PreVal, logistic regression, ridge regression (unscaled) and ridge regression (scaled) on random projections of the MNIST data

Note that direct scaling of ridge regression exhibits double-descent around $n \approx p$, while scaling pre-validated predictions removes this phenomena

For large p pre-validated is highly competitive; for large n one would expect logistic regression via maximum likelihood will outperform PreVal

Outline

Ridge Regression Facts (1)

- Consider the ridge regression solution

$$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- Let $\mathbf{A}$ be an orthonormal matrix, i.e., $\mathbf{A}^T\mathbf{A} = \mathbf{I}$
- Fact: if $\mathbf{Z} = \mathbf{XA}$, then

$$\hat{\boldsymbol{\alpha}} = \left(\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I}\right)^{-1}\mathbf{Z}^T\mathbf{y}$$

  is related to $\hat{\beta}$ by

$$\hat{\beta} = \mathbf{A}\hat{\boldsymbol{\alpha}}$$

- Proof: substitute $\mathbf{XA}$ into $\hat{\alpha}$ and simplify

Ridge Regression Facts (2)

- Let $\mathbf{Z}$ be an orthogonal matrix, i.e.,

$$\mathbf{Z}^T\mathbf{Z} = \mathbf{D} = \mathrm{diag}(d_1, \ldots, d_p)$$

$\Longrightarrow$ $\mathbf{D}$ is a diagonal matrix

- Fact: the ridge regression solution simplifies

$$\begin{aligned} \hat{\boldsymbol{\alpha}} &= \left(\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I}\right)^{-1}\mathbf{Z}^T\mathbf{y} \\ &= (\mathbf{D} + \lambda\mathbf{I})^{-1}\mathbf{Z}^T\mathbf{y} \\ &= \begin{pmatrix} \frac{c_1}{d_1+\lambda} \\ \vdots \\ \frac{c_p}{d_p+\lambda} \end{pmatrix} \end{aligned}$$

where $\mathbf{c} = \mathbf{Z}^T\mathbf{y}$

$\Longrightarrow$ so we can compute the estimates for a given λ in $O(p)$ time

SVD

- So if we can find a rotation $\mathbf{A}$ that renders $\mathbf{X}$ orthogonal we can use this to try lots of values of λ "for free"
- We can use the singular value decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

  where
  - $\mathbf{U}$ is an $n \times n$ orthonormal matrix;
  - $\mathbf{S}$ is a $n \times p$ "diagonal" matrix of singular values;
  - $\mathbf{V}$ is a $p \times p$ orthonormal matrix
- We can form a new feature matrix $\mathbf{R} = \mathbf{X}\mathbf{V} = \mathbf{U}\mathbf{S}$ which satisfies

$$\mathbf{R}^T\mathbf{R} = \mathbf{S}^2$$

  i.e., it is orthogonal
- If $p > n$, then an "economy SVD" means that $\mathbf{R} \in \mathbb{R}^{n \times n}$
- Crucially, as each linear prediction in a L-class classification problem uses the same feature matrix, this SVD only needs to be done once

SVD

- So if we can find a rotation $\mathbf{A}$ that renders $\mathbf{X}$ orthogonal we can use this to try lots of values of λ "for free"
- We can use the singular value decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

  where
  - $\mathbf{U}$ is an $n \times n$ orthonormal matrix;
  - $\mathbf{S}$ is a $n \times p$ "diagonal" matrix of singular values;
  - $\mathbf{V}$ is a $p \times p$ orthonormal matrix
- We can form a new feature matrix $\mathbf{R} = \mathbf{X}\mathbf{V} = \mathbf{U}\mathbf{S}$ which satisfies

$$\mathbf{R}^T\mathbf{R} = \mathbf{S}^2$$

  i.e., it is orthogonal
- If $p > n$, then an "economy SVD" means that $\mathbf{R} \in \mathbb{R}^{n \times n}$
- Crucially, as each linear prediction in a L-class classification problem uses the same feature matrix, this SVD only needs to be done once

LOOCV Predictions (1)

- The final piece we need is to compute $\tilde{\mathbf{H}}_\lambda$ efficiently
  - By using LOOCV we can exploit the "shortcut formula"
- Define the "hat matrix"

$$\mathbf{G}(\lambda) = \mathbf{X}\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_p\right)^{-1}\mathbf{X}^T$$

- Let $\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$ be the ridge predictions
- Let $\mathbf{e}$ denote the residuals from the full ridge fit, i.e.,

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$$

- Then the LOOCV squared error is given by

$$\sum_{i=1}^{n}\left(\frac{e_i}{1 - G_{i,i}(\lambda)}\right)^2$$

LOOCV Predictions (2)

- With some re-arrangement, we can use this to compute the cross-validation predictions

$$\tilde{\eta}_i = \hat{y}_i + e_i \left(1 - \frac{1}{1 - G_{i,i}(\lambda)} \right)$$

- Importantly, we only need the diagonal entries of $\mathbf{G}(\lambda)$
- If $\mathbf{X}$ is orthogonal, these can be computed in $O(r^2)$ time, where $r = \min(n, p)$
- Therefore the entire matrix can be computed in $O(Lr^2)$ time given an SVD decomposition of $\mathbf{X}$

PreVal Algorithm

1. Perform SVD on $\mathbf{X}$ and form $\mathbf{R} = \mathbf{UV}$, $O(mr^2)$
2. For each $\lambda \in \{\lambda_1, \ldots, \lambda_Q\}$
    1. Compute pre-validated predictors $\tilde{\mathbf{H}}_\lambda$, $O(Lr^2)$
    2. Solve
       $$\hat{\kappa}_\lambda = \arg\min_{\kappa > 0} \left\{ l \left(\kappa \cdot \tilde{\mathbf{H}}_\lambda \right) \right\}$$
       using a line search, $O(r^2)$
3. Use $(\lambda, \hat{\kappa}_\lambda)$ pair with smallest log loss to compute estimates
   $$\hat{\boldsymbol{\beta}}_\lambda^{(j)} = \mathbf{V}\hat{\boldsymbol{\alpha}}^{(j)}$$
   in the original feature space, $O(np)$
   $\Longrightarrow$ whole process same complexity as a single ridge fit, $O(mr^2)$

Thank you!

- Thank you for your attention!
- Code is available at
  - `https://github.com/angus924/preval`
- Relevant publications with more details
  - A. Dempster, G.I.Webb, D.F.Schmidt, "*Prevalidated ridge regression is a highly-efficient drop-in replacement for logistic regression for high-dimensional data*", arXiv:2401.15610v1
  - S. Tew, M. Boley and D.F.Schmidt, "*Bayes beats Cross Validation: Fast and Accurate Ridge Regression via Expectation Maximization*", NeuRIPS, 2023