JOURNAL OF COMPUTATIONAL BIOLOGY
Volume XX, Number XX, 2021
Mary Ann Liebert, Inc.
Pp. 1–10
DOI: 10.1089/cmb.2020.0284

Computing Leapfrog Regularization Paths with Applications to Large-Scale K-mer Logistic Regression

PHILIPP BENNER

ABSTRACT

High-dimensional statistics deals with statistical inference when the number of parameters or features p exceeds the number of observations n (i.e., $p \gg n$). In this case, the parameter space must be constrained either by regularization or by selecting a small subset of $m \leq n$ features. Feature selection through ℓ_1-regularization combines the benefits of both approaches and has proven to yield good results in practice. However, the functional relation between the regularization strength λ and the number of selected features m is difficult to determine. Hence, parameters are typically estimated for all possible regularization strengths λ. These so-called regularization paths can be expensive to compute and most solutions may not even be of interest to the problem at hand. As an alternative, an algorithm is proposed that determines the ℓ_1-regularization strength λ iteratively for a fixed m. The algorithm can be used to compute leapfrog regularization paths by subsequently increasing m.

Keywords: feature selection, ℓ_1-regularization, LARS, orthogonal matching pursuit.

1. INTRODUCTION

In statistics and machine learning, a common problem is to learn a model of the form

$$\mu_\theta(z) = \sigma(\theta_1 f_1(z) + \ldots + \theta_m f_m(z)),$$

with coefficients $\theta = (\theta_1, \ldots, \theta_m) \in \mathbb{R}^m$ and where f_j denotes the j th feature computed from the input data z. In statistics, σ is called *mean function*, whereas in machine learning it is commonly referred to as *activation function*. The model μ_θ is typically estimated by minimizing a given loss function on a set of n observations.

For instance, in linear regression, the mean function is $\sigma(x) = x$ and the parameters θ would be estimated by minimizing the sum of squared residuals, also called ordinary least squares (OLS). In the case of artificial neural networks, μ_θ represents a single output neuron and f_j denotes the jth output of the preceding layers of the network. The activation function σ is typically either a sigmoid function or some variant. Neural networks are estimated by minimizing, for instance, the mean squared or cross-entropy error.

Two fundamentally different ways can be found in the literature of how features f_j are estimated. Some machine learning methods, such as neural networks, treat each f_j as a parametric function, which is optimized jointly with the model parameters θ. In contrast, in statistics a fixed set of $p > m$ features

Department of Computational Molecular Biology, Max Planck Institute for Molecular Genetics, Berlin, Germany.

$\{x_j = f_j(z)\}$ is considered, from which a subset of size m is selected. Although the former is computationally more tractable, it often leads to nonconvex optimization problems. The focus of this study lies in the latter. In particular, it is assumed that the feature space is very large and that only a small subset of size m must be selected ($p \gg m$). Ideally, the parameters of a statistical model are estimated for a given loss function $\omega : \mathbb{R}^p \to \mathbb{R}$ by minimizing $\omega(\theta)$ subject to the constraint $\|\theta\|_0 = m$. However, this approach is generally not feasible since it requires to test all subsets of size m.

Several strategies exist to compute approximate solutions. For instance, matching pursuit (MP) (Tropp et al., 2007) is a greedy strategy that selects one feature at a time until the desired number of m features is reached. This algorithm was found to be overly greedy in practice (Efron et al., 2004) and may provide poor subset selections. A more powerful strategy is to consider the ℓ_1 constraint $\|\theta\|_1 = \Lambda$ (Tibshirani, 1996). In practice, this leads to the equivalent unconstrained optimization problem

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, \omega(\theta) + \lambda \|\theta\|_1, \tag{P0}$$

where the penalty strength $\lambda \geq 0$ is a parameter that controls the sparseness of θ and is often chosen so that $\|\theta\|_0 = m$. Although this approach often selects nearly optimal subsets, it comes with the difficulty of determining the penalty or regularization strength λ.

Unfortunately, there exists no known functional relation between m and λ. However, grid search can be used to compute solutions of the optimization problem P0 for a manually specified set of regularization strengths. More efficient algorithms, such as least angle regression (LARS) (Efron et al., 2004) and the homotopy (Osborne et al., 2000) algorithm, initially start with $\lambda = \lambda_{\max}$ for which all coefficients are zero. These methods decrease λ and determine *breakpoints* $\lambda_1 > \lambda_2 > \ldots \lambda_q$ at which features either become active (nonzero coefficient) or inactive (zero coefficient). By interpolating between breakpoints a *regularization path* $\{\hat{\theta}(\lambda) | 0 \leq \lambda \leq \lambda_{\max}\}$ is obtained, which contains solutions for all feasible regularization strengths.

Computing full regularization paths is costly for large feature spaces and complex models. In such cases, the regularization path is only computed up to a maximal $m_q \ll p$. Furthermore, a solution of the optimization problem P0 with m features will often behave very similarly to a solution with $m+1$ features. Therefore, the focus of this study is to present and test an algorithm that computes *leapfrog regularization paths* containing solutions $\hat{\theta}(\lambda_k)$ such that $\left\|\hat{\theta}(\lambda_k)\right\|_0 = m_k$, where $m_1 < m_2 < \ldots < m_q$ is a given sequence of cardinalities. The main advantage of this algorithm is that m_q can be chosen much larger than with existing regularization path algorithms, because it does not compute all solutions of the regularization path. Furthermore, the sequence of cardinalities $m_1 < m_2 < \ldots < m_q$ can be selected so that a clear difference in performance for m_k and m_{k+1} is observed.

2. ALGORITHMIC BACKGROUND

Many feature selection methods were first developed for linear regression and later extended to more complex models, such as generalized linear models (GLMs). Hence, linear regression may serve as a role model for introducing feature selection methods. Let $X \in \mathbb{R}^{n \times p}$ denote the covariates or data matrix with n observations $\{x_i\}$(rows) and p features $\{f_j\}$(columns). For linear regression, σ is the identity function so that $\mu_\theta(x_i) = x_i\theta$ and the objective function $\omega = \omega_L$ is the sum of squared residuals defined as

$$\omega_L(\theta) = \sum_{i=1}^{n} (y_i - \mu_\theta(x_i))^2 = \|y - X\theta\|_2^2.$$

Furthermore, $y \in \mathbb{R}^n$ is a vector of n dependent variables, also called response, and $\varepsilon = y - X\theta$ is referred to as residuals. If $p \leq n$ and X has full rank p then the coefficients θ can be estimated by OLS, which minimizes the sum of squared residuals $\omega_L(\theta) = \|\varepsilon\|_2^2 = \|y - X\theta\|_2^2$. When $p > n$, parameters can be estimated by minimizing the ℓ_1-penalized loss $L_\lambda(\theta) = \omega_L(\theta) + \lambda\|\theta\|_1$, that is

$$\hat{\theta} = \underset{\theta}{\arg\min}\, L_\lambda(\theta), \tag{P1}$$

which is known as the Lasso (Tibshirani, 1996).

The following interpretation of OLS (Hastie et al., 2009) is essential for understanding many feature selection methods. In $\mathbb{R}^n$ the dot product $X\theta$ is a point in the hyperplane H spanned by the columns

$[f_1, \ldots, f_p]$ of X and y is a vector typically not contained in H. The point $X\hat{\theta}$ in H that minimizes the length of the residual vector ε is located directly below y, that is, it is the projection of y onto H. It follows that

$$X\hat{\theta} = X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}y,$$

where $X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}$ is a projection matrix. Therefore, the OLS solution $\hat{\theta} = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}y$ can be interpreted as the scalar projection of y onto H. If the covariates are appropriately normalized, that is, $X^{\mathrm{T}}X = I$, then $\hat{\theta} = X^{\mathrm{T}}y$. Most importantly, $X^{\mathrm{T}}y$ can be interpreted as the correlation of y with X. The OLS solution satisfies $X^{\mathrm{T}}(y - X\hat{\theta}) = 0$, which shows that the residuals are uncorrelated with X. Many feature selection methods decrease the correlation $c(\theta) = X^{\mathrm{T}}(y - X\theta)$ iteratively starting with $c(0)$ until the residuals are orthogonal to X.

2.1. Matching pursuit

MP is a greedy feature selection algorithm (Tropp et al., 2007) that selects one feature at a time. Let the columns of X have unit length, that is, $f_j^{\mathrm{T}}f_j = 1$ for all $j = 1, \ldots, p$. The first feature $j_1 \in \{1, \ldots p\}$ is computed as $j_1 = \arg\max \left| f_j^{\mathrm{T}} y \right|$. Hence, the algorithm selects the feature j with the largest scalar projection of y onto f_j, or the feature that is most correlated with y.

All remaining features are selected accordingly. Let $\Omega = \{j_1, \ldots, j_t\}$ denote the set of active features and $X_\Omega = (f_j)_{j \in \Omega}$ the covariate matrix restricted to Ω. Furthermore, let

$$\varepsilon_t = y - \sum_{s=1}^{t} f_{j_s}\hat{\theta}_{j_s} = y - X_\Omega \hat{\theta}_\Omega,$$

denote the residuals at iteration t with $\hat{\theta}_\Omega = \underset{\theta}{argmin} \|y - X_\Omega\theta\|_2^2$. At iteration $t+1$ the algorithm computes

$$j_{t+1} = \arg\min_j \left\| \varepsilon_t - f_j\hat{\theta}_j \right\|_2^2 = \arg\max_{j \notin \Omega} \left| f_j^{\mathrm{T}}\varepsilon_t \right|,$$

that is, it selects that feature j for which the scalar projection of ε_t onto f_j is largest. Alternatively, $f_j^{\mathrm{T}}\varepsilon_t$ can be interpreted as the correlation of the jth feature with the residuals ε_t.

The MP algorithm has been applied to several other models, including logistic regression (Lozano et al., 2011), although the correlation $c(\hat{\theta}_\Omega)_k$ might be less informative.

2.2. LARS and homotopy algorithm

LARS is an iterative feature selection method for linear regression (Efron et al., 2004). The algorithm is related to MP, but it is less greedy and gives only as much weight to each feature as it deserves (Hastie et al., 2009). Another method is the homotopy algorithm (Osborne et al., 2000), which is identical to LARS except that it also allows features to be removed within an iteration. It is referred to as LARS with Lasso modification in the literature (Efron et al., 2004; Donoho and Tsaig, 2008). Both methods are introduced hereunder. X is typically assumed to be standardized so that the correlations $c(\theta)$ are not tainted by heterogeneous scaling.

The LARS and homotopy algorithms maintain a set of active features $\Omega \subset \{1, \ldots, p\}$ all equally correlated with the residuals $y - X\hat{\theta}$ for the current estimate $\hat{\theta}$. In contrast, features not in Ω are less correlated with the residuals. Both algorithms initially set $\theta = 0$. In each subsequent iteration, the coefficients θ are updated so that the correlations of all features in Ω uniformly shrink toward zero until some other feature $j' \in \Omega^c$ is equally correlated with the residuals. For the next iteration, j' is added to the active set Ω. The homotopy algorithm also removes a feature j from the active set Ω when the coefficient θ_j becomes zero. In this way, λ is incrementally reduced from its maximum value, where all features are inactive ($\Omega = \oslash$), to zero, where all coefficients are nonzero ($\Omega = \{1, 2, \ldots, p\}$). The values of λ at which a feature enters or leaves the active set Ω are referred to as breakpoints.

More specifically, the decrement in correlation at which a feature enters the active set is given by

$$\gamma^+ = \min_{j \in \Omega^c}{}^+ \left\{ \frac{\lambda - c_j(\theta)}{1 - f_j^{\mathrm{T}}Xv}, \frac{\lambda + c_j(\theta)}{1 + f_j^{\mathrm{T}}Xv} \right\}, \tag{1}$$

where $\min^+$ is the minimum over positive elements and $c_j(\theta)$ denotes the jth element of $c(\theta)$. For LARS without Lasso modification, the step size γ^* equals γ^+; however, the homotopy algorithm also removes a feature j from the active set when for some γ

$$\theta_j + \gamma v_j = 0,$$

so that $\gamma^- = \min_{j\in\Omega}\{-\theta_j/v_j\}$. The subsequent breakpoint is given by $\gamma^* = \min\{\gamma^+, \gamma^-\}$.

LARS with Lasso modification is equivalent to solving the Lasso problem (P1) for all regularization strengths $\lambda \geq 0$. A direct application of LARS or the homotopy algorithm to GLMs is not possible, because the path along which the correlations $c_\Omega(\theta) = \lambda \mathbf{1}$ uniformly decline is nonlinear. As a consequence, there is no analytical solution to the position of breakpoints. Nevertheless, a linear approximation for GLMs exists (Park and Hastie, 2007). The algorithm, however, requires several iterations until a breakpoint is reached.

3. COMPUTING LEAPFROG REGULARIZATION PATHS

LARS and the homotopy algorithm compute full regularization paths $\{\hat{\theta}(\lambda) | 0 \leq \lambda \leq \lambda_{\max}\}$ for all feasible regularization strengths λ. These algorithms start with $\lambda = \lambda_{\max}$ and decrease the regularization strength until the unconstrained model is obtained. The full and exact regularization path is computed by linear interpolation between breakpoints. Extensions of LARS to GLMs (*LARS-GLM*) instead compute approximate regularization paths, because $\hat{\theta}(\lambda)$ is nonlinear between breakpoints and there exists no analytical expression to compute at what values of λ breakpoints occur.

For certain applications the full regularization path can be very costly to compute, especially when the number of features p is large. In such cases, the algorithms can be stopped when a maximum number of features m_q is reached. Nevertheless, because LARS-GLM successively decreases the regularization strength λ, it might take very long until a desired number of features m is reached. Furthermore, estimated models with m and $m+1$ selected features might behave very similarly, for instance, in terms of classification performance. For such situations, an algorithm is introduced that computes for an a priori defined sequence $m_1 < m_2 < \ldots < m_q$ the estimates $\hat{\theta}(\lambda_k)$ such that $\left\|\hat{\theta}(\lambda_k)\right\|_0 = m_k$. The set of estimates $\{\hat{\theta}(\lambda_k) | \left\|\hat{\theta}(\lambda_k)\right\|_0 = m_k\}_{k=1}^q$ is called a leapfrog regularization path. The algorithm can be derived from proximal gradient descent (Boyd et al., 2004) or from a modification of LARS-GLM.

Let ω denote the objective function and $\omega_j'(\hat{\theta}) = \frac{\partial}{\partial\theta_j}\omega(\theta)|_{\theta=\hat{\theta}}$ the jth partial derivative at the current estimate $\hat{\theta}$. Given a fixed $m \in \{m_1, m_2, \ldots, m_q\}$ the following update rules are iterated until convergence:

$$\begin{aligned} \Omega &\leftarrow \left\{\mathrm{j} : \hat{\theta}_\mathrm{j} \neq 0 \text{ or } |\omega_j'(\hat{\theta})| > \lambda\right\} \text{ for } \lambda \geq 0 \text{ such that } |\Omega| = \mathrm{m} \\ \hat{\theta}_\Omega &\leftarrow \arg\min_{\theta\in\mathbb{R}^{|\Omega|}} \omega_\Omega(\theta) + \lambda\|\theta\|_1 \end{aligned}.$$

The algorithm has converged when Ω is identical between two successive iterations. The first update step computes a new active set Ω. All features with nonzero coefficients remain in the active set. Additional features are added based on the absolute value of the partial derivatives. An estimate of the optimal λ parameter is obtained during this step. In the second step, the coefficients of the active set are re-estimated, all other coefficients remain zero. The re-estimation is computationally cheap if $m \ll p$, because the gradient descent method operates only on m selected features. Once the algorithm has converged for a fixed $m = m_k$, the procedure is repeated for $m = m_{k+1}$ (see Algorithm 1).

Algorithm 1 Calculate leapfrog regularization path $\{\hat{\theta}(\lambda_k) | \left\|\hat{\theta}(\lambda_k)\right\|_0 = m_k\}_{k=1}^q$

procedure LEAPFROGPATH($m_1 < m_2 < \ldots < m_q$)
  $r \leftarrow \oslash$
  $\hat{\theta} \leftarrow 0 \in \mathbb{R}^p$
  for $k = 1, \ldots, q$ **do**
    repeat
      $\Omega \leftarrow \left\{\mathrm{j} : \hat{\theta}_\mathrm{j} \neq 0 \text{ or } |\omega_j'(\hat{\theta})| > \lambda\right\}$ for $\lambda \geq 0$ such that $|\Omega| = m_k$
      $\hat{\theta}_\Omega \leftarrow \arg\min_{\theta\in\mathbb{R}^{|\Omega|}} \omega_\Omega(\theta) + \lambda\|\theta\|_1$
    until Ω remains unchanged
    $r \leftarrow r \cup \{\hat{\theta}_\Omega\}$
  end for
  return r
end procedure

Algorithm 1 relies on ℓ_1-regularization for feature selection. Compared with MP it is much less greedy, because it does not simply select one feature per iteration (Hastie et al., 2009). Instead, the active set is refined until a solution to optimization problem (P0) is found. The algorithm is more similar to LARS-GLM, but it uses a different method for determining the regularization strength λ. Whereas LARS estimates the position of subsequent breakpoints, Algorithm 1 iteratively updates λ until a given number of m features is selected.

Despite extremely high-dimensional feature space, the estimation of parameters is relatively cheap when $m \ll p$, because the gradient descent method operates only on a small subset of m features and not the full feature space of size p. In other words, estimating the coefficients of the logistic regression with a gradient method takes $\mathcal{O}(nmT)$ steps, where T is the number of iterations of the gradient descent method. This is much faster than estimating the parameters on the full feature space, which would take $\mathcal{O}(npT)$ steps.

3.1. Derivation from proximal gradient descent

The optimization problem P0 is commonly solved using proximal gradient descent (Boyd et al., 2004). In general, proximal gradient descent can be used to compute minimizers of functions $f(x) = g(x) + h(x)$, where g is convex differentiable and h is convex but not differentiable. The update equation is given by

$$x \leftarrow \text{prox}_h(x - \gamma \nabla g(x)),$$
$$\text{where} \quad \text{prox}_h(x) = \arg\min_{x'} \frac{1}{2\gamma} \|x' - x\|_2^2 + h(x')$$

and γ is the step size. The proximal operator prox_h can be solved analytically for $h(x) = \|x\|_1$. Note that the equation inside the proximal operator is a simple gradient descent step applied to g..

For the optimization problem $P0$ the following update equations are obtained:

$$\vartheta \leftarrow \theta - \gamma \nabla \omega(\theta)$$
$$\theta \leftarrow \arg\min_{\theta'} \frac{1}{2\gamma} \|\theta' - \vartheta\|_2^2 + \|\theta'\|_1 = \text{sign}(\vartheta) \max\{|\vartheta| - \gamma\lambda, 0\},$$

where ϑ is the result after the gradient descent step and before applying the proximal operator. Assume that for a feature j the coefficient θ_j is zero. The coefficient remains zero during subsequent update steps unless the gradient update is large enough, because of the application of the proximal operator. More specifically, for any step size $\gamma > 0$, a coefficient θ_j remains zero unless $|\omega_j'(\hat{\theta})| > \lambda$.

3.2. Derivation from LARS

In each iteration, LARS either removes a feature from the active set or selects a new one. The focus here is on the selection of new features, that is, Equation (1). By assuming that $f_j^{\mathrm{T}} X v = 0$, a much simplified step size

$$\gamma^+ = \min_{j \in \Omega^c}{}^+ \{\lambda - c_j(\theta), \lambda + c_j(\theta)\},$$

is obtained, which shows that $\lambda - \gamma^+ = \max_{j \in \Omega^c} |c_j(\theta)|$ and remember that for linear and logistic regression $c_j(\theta) = \omega'_j(\theta)$. Hence, m features can be selected by setting λ equal to the mth largest absolute correlation or partial derivative.

4. APPLICATIONS TO LARGE-SCALE LOGISTIC REGRESSION

Algorithm 1 can be easily applied to GLMs, such as logistic regression, as demonstrated in the following. The closest method in the literature is LARS-GLM and the *GNU-R* library *glmpath* is used for comparing it with Algorithm 1. Decisive for the computational cost is the number of *steps* required for computing a regularization path. A step here refers to the estimation of parameters for a fixed regularization strength λ, that is, solving optimization problem $P0$. For Algorithm 1 this corresponds to one evaluation of the two update equations. In addition, the number of passes of the gradient descent method through the entire data set (*epochs*) is informative. However, because different gradient descent methods are used for Algorithm 1 and LARS-GLM, a direct comparison in terms of epochs is not possible.

Enhancers are regulatory elements in the genome that control the cell type-specific expression of genes. The DNA sequence of an enhancer is predictive of the cell types in which it actively controls gene expression (Lee et al., 2011; Hashimoto et al., 2016; Kelley et al., 2016; Yang et al., 2017). Training and test data sets were compiled from DNA sequences of 56, 880 enhancers active in embryonic mice at day 15.5 in brain (positive set) and nonbrain tissues (negative set). Enhancers were detected using ModHMM (Benner and Vingron, 2020) and the data are available online (Benner, 2020). For each enhancer sequence, the *K*-mer content is determined. *K*-mers are short DNA subsequences of length *K*. In this example, *K*-mers of variable length are considered. The resulting data matrix $X = (x_{ij})$ has n rows (observations) and p columns (features), where x_{ij} is the number of occurrences of the jth *K*-mer in the ith enhancer sequence. A logistic regression model is used to discriminate between brain and nonbrain enhancers. The average negative log-likelihood (*average loss*) is given by

$$\omega_G(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\{y_i \log \sigma(x_i\theta) + (1-y_i)\log(1-\sigma(x_i\theta))\},$$

where σ is the sigmoid function and $y \in \{0,1\}^n$ is a vector of n class labels. Analogously to linear regression, when $p > n$, parameters are estimated by minimizing the ℓ_1-penalized average logistic loss $G_\lambda(\theta) = \omega_G(\theta) + \lambda\|\theta\|$ (*penalized average loss*), that is

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, G_\lambda(\theta).$$

For logistic regression the objective function G_λ is convex and the optimum $\hat{\theta}$ must satisfy $X^{\mathrm{T}}(y - \sigma(X\hat{\theta})) \in \lambda\partial\|\theta\|_1$. SAGA (Defazio et al., 2014) is used as gradient descent method to compute solutions of this optimization problem. To guarantee stable convergence of the gradient descent method, the data are normalized to unit variance.

In a first example, *K*-mers of length $2-7$ are used, that is, $K \in \{2, 3, \ldots, 7\}$. This leads to a feature space of size $p = 21,840$, where only *K*-mers that actually occur in the data set are considered. Figure 1 shows the leapfrog regularization path computed with Algorithm 1 for $m \in \{1,\ 10,\ 500\}$. In addition, a regularization path computed with LARS-GLM is shown and the algorithm was stopped when it reached 500 features. The example shows that the computation of the leapfrog path requires much fewer steps, however, at the expense of not evaluating the full path up to 500 features. Nevertheless, for larger feature spaces and larger m the LARS-GLM algorithm quickly becomes computationally too expensive.

In a second example, the set of *K*-mers is extended to lengths of $2-12$, that is, $K \in \{2,\ 3,\ \ldots,\ 12\}$, which results in a feature space of size $p = 16,876,344$. Figure 2 shows the estimation of parameters for a leapfrog regularization path with Algorithm 1 and targets $m \in \{10,\ 100,\ 1000,\ 5000\}$. The algorithm

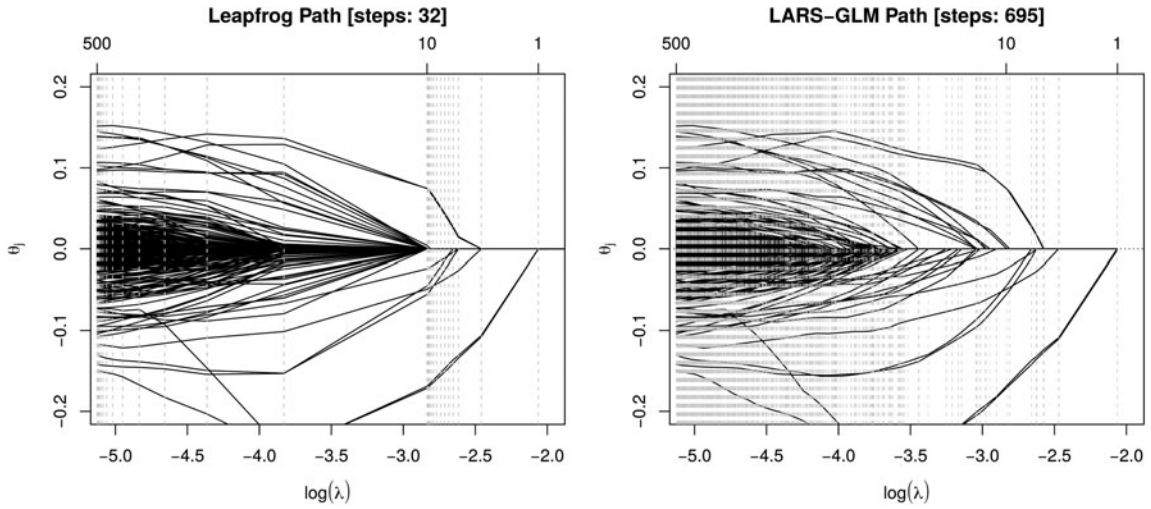

FIG. 1. Comparison of regularization paths. Leapfrog regularization path (left) with $m \in \{1, 10, 500\}$ and LARS-GLM regularization path (right). Vertical dashed gray lines show steps of the path algorithms, that is, λ values for which parameters $\hat{\theta}$ were estimated

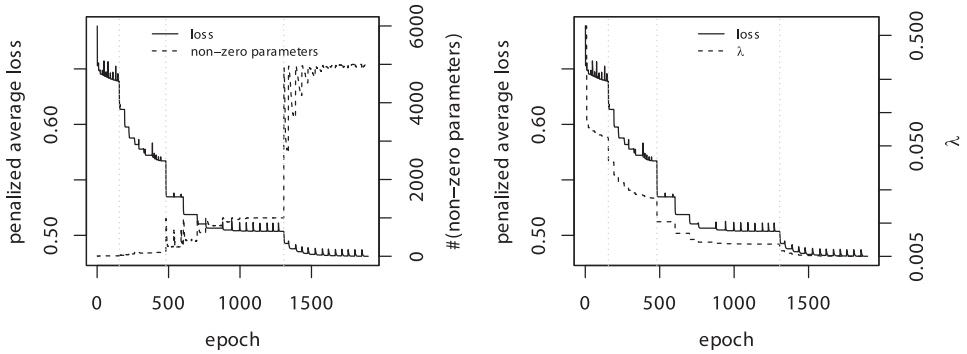

FIG. 2. Estimation of a leapfrog regularization path with $m \in \{10, 100, 1000, 5000\}$. The vertical lines show the epoch at which m is increased. An epoch is one pass of the gradient descent method through the entire data. For a fixed penalty strength λ, the gradient descent method is iterated until convergence. If the number of nonzero parameters does not match the target m, λ is updated and the parameters are re-estimated.

successively decreases the penalty strength λ until a target is reached. After only 1500 epochs, corresponding to ~ 50 steps of Algorithm 1, the leapfrog regularization path is computed. In contrast, extracting the 5000 most important K-mers with the LARS-GLM algorithm is computationally not feasible.

4.1. Bias-variance trade-off

In statistics and machine learning one is often confronted with the task of choosing a model family $\mathbb{H}$ that captures the underlying structure of the observed data, but also generalizes well to new observations. The bias-variance trade-off suggests that both cannot be accomplished with arbitrary precision. When the capacity of $\mathbb{H}$ is too small, the estimated models has poor performance on the observed data as well as on new observations. The underfitting of the estimated models is caused by a strong (inductive) *bias* of $\mathbb{H}$. As the capacity of $\mathbb{H}$ increases, the model is capable of extracting more complex patterns from the training data, but the *variance* of parameter estimates across samples increases. If the capacity of $\mathbb{H}$ is too large it causes overfitting, that is, the estimated models will perform very well on the training data but generalize poorly. A model family that generalizes well must minimize both sources of error, the bias and the variance.

For instance, in the case of linear or logistic regression, the capacity of $\mathbb{H}$ can be increased by using more features (i.e., increasing p). Regularization, in contrast, constrains the parameter space and, therefore, reduces the capacity of $\mathbb{H}$ (Hastie et al., 2009). In practice, it is often observed that models generalize best if regularization is used in combination with very large p (Bühlmann and Van De Geer, 2011).

In this section, the effect of regularization and the size of the feature space p on the capacity of $\mathbb{H}$ is studied. A balanced set of $n = 10{,}000$ brain and nonbrain enhancers is used. The size of the feature space p is controlled by the maximal length of K-mers. Figure 3 shows the training and test error for a fixed penalty

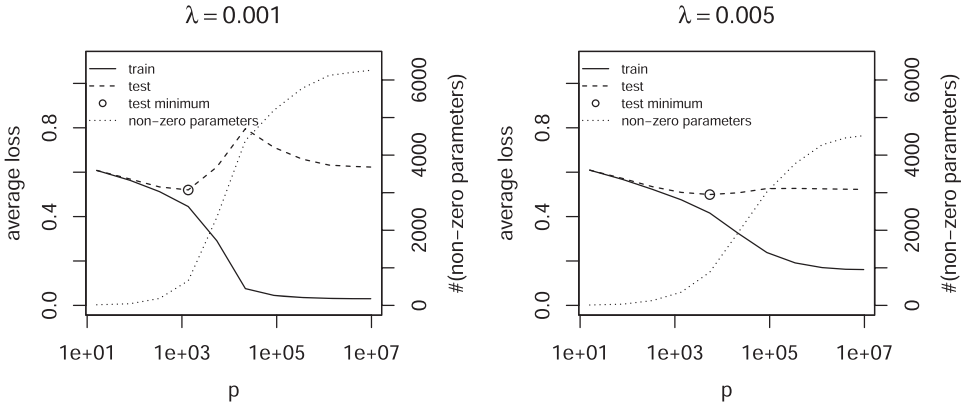

FIG. 3. Average loss as a function of p and fixed penalty λ.

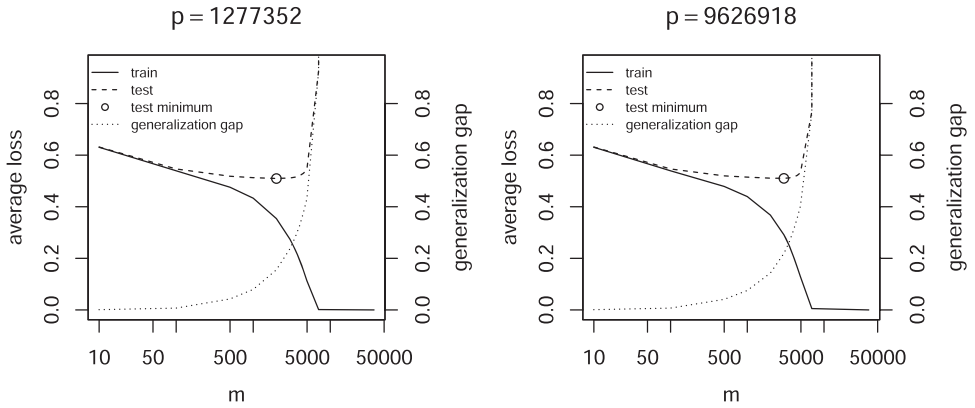

FIG. 4. Average loss as a function of m and fixed p.

λ and increasing p. Whereas the training error decreases monotonically, the test error starts to decrease after it reaches a maximum. This double descent has been observed before (Vallet et al., 1989; Opper et al., 1990; Duin, 2000; Belkin et al., 2019) and is theoretically expected (Cheema and Sugiyama, 2020). It shows that the capacity of $\mathbb{H}$ is the result of a complex interplay between the size of the feature space p and the regularization strength λ.

The dependency between λ and p makes it difficult in practice to find optimal values for both parameters, especially when double descents are possible, because an extensive search is required to eliminate the possibility of being in a local optimum. Instead of adding more parameters by increasing p, Algorithm 1 is utilized to control the number of active features m for a fixed p. Figure 4 shows that in this case no double descent is observed, suggesting a monotonic relation between m and the capacity of $\mathbb{H}$, as expected from the bias-variance trade-off. Furthermore, the test error increases rapidly as the interpolation threshold is crossed, that is, the point where the model is able to perfectly discriminate between the positive and negative class.

Figure 5 shows the converse case, the average loss for fixed m but variable p. Also in this case no double descent of the test error is observed. However, the training error increases with p after reaching a minimum, because the regularization strength must increase to keep m fixed. The generalization gap, that is, the difference between test and training loss, steadily decreases, showing that the capacity of $\mathbb{H}$ seems to decline.

5. CONCLUSION

Several iterative feature selection methods were discussed in this study. MP is among the simplest methods, but can lead to poor subset selection because of its overly greedy strategy (Hastie et al., 2009). More advanced methods, such as LARS and the homotopy algorithm, often yield better results in practice.

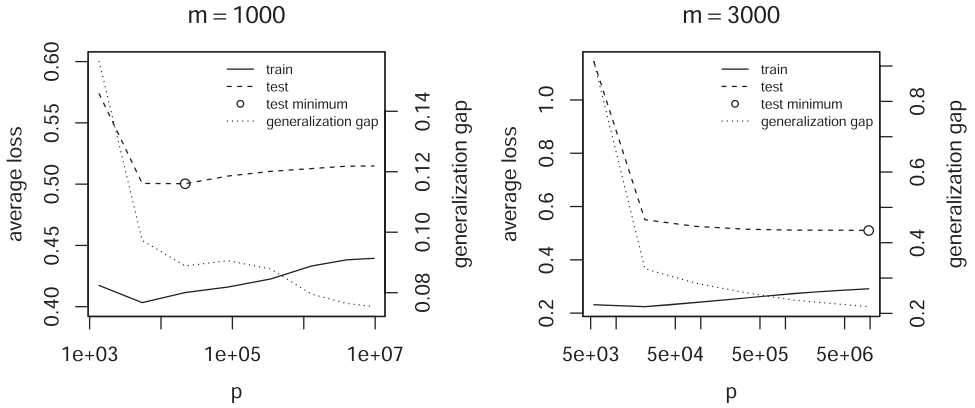

FIG. 5. Average loss as a function of p and fixed m.

The estimates of LARS with Lasso modification are equivalent to solving the Lasso problem (P1) for all regularization strengths λ. Extensions to GLMs are more expensive to compute, because there exists no analytical solution to the λ values at which the active set of features changes. As a result, many more iterations are required to compute a full regularization path. In practice, one is often not interested in the full solution, because estimates with m and $m+1$ features provide very similar predictions.

Algorithm 1 instead computes for an a priori defined sequence $m_1 < m_2 < \ldots < m_q$ a leapfrog regularization path $\{\hat{\theta}(\lambda_k) | \left\|\hat{\theta}(\lambda_k)\right\|_0 = m_k\}_{k=1}^{q}$. This algorithm can be derived from proximal gradient descent or a simple modification of LARS for GLMs. It is very effective for high-dimensional problems where only a small subset of features is required for making accurate predictions. It is less greedy than MP, because it does not simply select one feature per iteration. As opposed to LARS, which computes subsequent breakpoints, Algorithm 1 iteratively refines the regularization strength λ until a given number of m features is selected.

On a data set of DNA sequences the algorithm very efficiently computes the leapfrog regularization path despite the high-dimensional feature space, where LARS-GLM is computationally too expensive. In addition, the number of selected features m allows a good control of model complexity. An implementation of the algorithm is available at: https://github.com/pbenner/kmerLr

ACKNOWLEDGMENTS

I thank Martin Vingron and Kirsten Kelleher for their comments on the article and many inspiring discussions.

AUTHOR DISCLOSURE STATEMENT

No competing financial interests exist.

FUNDING INFORMATION

P.B. was supported by the German Ministry of Education and Research (BMBF, Grant No. 01IS18037G).

REFERENCES

Belkin, M., Hsu, D., Ma, S., and Mandal, S. 2019. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proc. Natl. Acad. Sci. U.S.A.* 116, 15849–15854.

Benner, P. 2020. ModHMM Genome Segmentations. Available at: https://github.com/pbenner/modhmm-segmentations. Accessed January 9, 2020.

Benner, P., and Vingron, M. 2020. ModHMM: A modular supra-Bayesian genome segmentation method. *J. Comput. Biol.* 27, 442–457.

Boyd, S., Boyd, S.P., and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press, Cambridge, United Kingdom.

Bühlmann, P., and Van De Geer, S. 2011. *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer Science & Business Media, Berlin, Germany.

Cheema, P., and Sugiyama, M. 2020. *A Geometric Look at Double Descent Risk: Volumes, Singularities, and Distinguishabilities*. arXiv preprint arXiv:2006.04366.

Defazio, A., Bach, F., and Lacoste-Julien, S. 2014. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In Ghahramani, Z., ed. *Advances in Neural Information Processing Systems*, 1646–1654. Curran Associates, Inc., New York, NY.

Donoho, D.L., and Tsaig, Y. 2008. Fast solution of l_1-norm minimization problems when the solution may be sparse. *IEEE Trans. Inf. Theory.* 54, 4789–4812.

Duin, R.P. 2000. Classifiers in almost empty spaces. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2, 1–7. IEEE, Barcelona, Spain.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R. et al. 2004. Least angle regression. *Ann. Stat.*. 32, 407–499.

Hashimoto, T., Sherwood, R.I., Kang, D.D., et al. 2016. A synergistic DNA logic predicts genome-wide chromatin accessibility. *Genome Res.* 26, 1430–1440.

Hastie, T., Tibshirani, R., and Friedman, J. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, New York, NY.

Kelley, D.R., Snoek, J., and Rinn, J.L. 2016. Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res.* 26, 990–999.

Lee, D., Karchin, R., and Beer, M.A. 2011. Discriminative prediction of mammalian enhancers from dna sequence. *Genome Res.* 21, 2167–2180.

Lozano, A., Swirszcz, G., and Abe, N. 2011. Group orthogonal matching pursuit for logistic regression. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 452–460. JMLR Workshop and Conference Proceedings, Fort Lauderdale, FL.

Opper, M., Kinzel, W., Kleinz, J., and Nehl, R. 1990. On the ability of the optimal perceptron to generalise. *J. Phys. A Math. Gen.* 23, L581.

Osborne, M.R., Presnell, B., and Turlach, B.A. 2000. A new approach to variable selection in least squares problems. *IMA J. Numer. Anal.* 20, 389–403.

Park, M.Y., and Hastie, T. 2007. L1-regularization path algorithm for generalized linear models. *J. R. Stat. Soc. Series B Stat. Methodol.* 69, 659–677.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Series B Stat. Methodol.* 58, 267–288.

Tropp, J., and Gilbert, A.C. 2007. Signal recovery from partial information via orthogonal matching pursuit. *IEEE Trans. Inform. Theory* 53, 4655–4666.

Vallet, F., Cailton, J.-G., and Refregier, P. 1989. Linear and nonlinear extension of the pseudo-inverse solution for learning Boolean functions. *Europhys. Lett.* 9, 315.

Yang, B., Liu, F., Ren, C., et al. 2017. Biren: Predicting enhancers with a deep-learning-based model using the DNA sequence alone. *Bioinformatics* 33, 1930–1936.

Address correspondence to:
Dr. Philipp Benner
Department of Computational Molecular Biology
Max Planck Institute for Molecular Genetics
Ihnestraße 73
Berlin 14195
Germany

E-mail: benner@molgen.mpg.de