
Early Stopping without a Validation Set

Maren Mahsereci

mmahsereci@tue.mpg.de
Max Planck Institute for Intelligent Systems,
Spemannstraße, Tübingen, Germany

Lukas Balles

lballes@tue.mpg.de
Max Planck Institute for Intelligent Systems,
Spemannstraße, Tübingen, Germany

Christoph Lassner*

classner@tue.mpg.de
Max Planck Institute for Intelligent Systems,
Spemannstraße, Tübingen, Germany

Philipp Hennig

ph@tue.mpg.de
Max Planck Institute for Intelligent Systems,
Spemannstraße, Tübingen, Germany

Abstract

Early stopping is a widely used technique to prevent poor generalization performance when training an over-expressive model by means of gradient-based optimization. To find a good point to halt the optimizer, a common practice is to split the dataset into a training and a smaller validation set to obtain an ongoing estimate of the generalization performance. We propose a novel early stopping criterion based on fast-to-compute local statistics of the computed gradients and entirely removes the need for a held-out validation set. Our experiments show that this is a viable approach in the setting of least-squares and logistic regression, as well as neural networks.

1 Introduction

The training of parametric machine learning models often involves the formal task of minimizing the expectation of a loss (risk) over a population $p(x)$ of data, of the form

$$\mathcal{L}(w) = \mathbf{E}_{x \sim p(x)} [\ell(w, x)], \quad (1)$$

where the loss function $\ell(w, x)$ quantifies the performance of parameter vector $w \in \mathbb{R}^D$ on data point x . In practice though, the data distribution $p(x)$ is usually unknown, and Eq. 1 is approximated by the *empirical risk*:

$$L_{\mathcal{D}}(w) = \frac{1}{M} \sum_{x \in \mathcal{D}} \ell(w, x). \quad (2)$$

Here \mathcal{D} denotes a dataset of size $M = |\mathcal{D}|$ with instances drawn independently from $p(x)$. Often there is easy access to the gradient of ℓ and gradient-based optimizers can be used to minimize the empirical risk. The gradient descent (GD) algorithm, for example, updates an estimate w_t for the minimizer of $L_{\mathcal{D}}$ according to $w_{t+1} = w_t - \alpha_t \nabla L_{\mathcal{D}}(w_t)$ with $\nabla L_{\mathcal{D}}(w) = 1/M \sum_{x \in \mathcal{D}} \nabla \ell(w, x)$, and some hand-tuned or adaptive step sizes α_t . In practice, however, evaluating $\nabla L_{\mathcal{D}}$ can become expensive for very large M thus making it impossible to make progress in a reasonable time. Instead, stochastic optimization methods are used, which use coarser but much cheaper gradient estimates by randomly choosing a mini-batch $\mathcal{B} \subset \mathcal{D}$ of size $|\mathcal{B}| = m \ll M$ from the training set and computing $\nabla L_{\mathcal{B}}(w) = 1/m \sum_{x \in \mathcal{B}} \nabla \ell(w, x)$. The gradient descent update then becomes $w_{t+1} = w_t - \alpha_t \nabla L_{\mathcal{B}}(w_t)$ and the corresponding iterative algorithm is commonly known as stochastic gradient descent (SGD) [17].

*equally affiliated with: Bernstein Center for Computational Neuroscience, Otfried-Müller-Str. 25, Tübingen, Germany

1.1 Overfitting, Regularization and Early-Stopping

Since the risk \mathcal{L} is virtually always unknown, a key question arising when minimizing the empirical risk $L_{\mathcal{D}}$, is how the performance of a model trained on a finite dataset \mathcal{D} generalizes to unseen data. Performance can be measured by the loss itself or other quantities, e.g., the mean accuracy in classification problems. Typically, to measure the generalization performance a finite *test set* is entirely withheld from the training procedure and the performance of the final model is evaluated on it. This test loss, however, is also only an estimator for \mathcal{L} (in the same sense as the train loss) with a finite stochastic error whose variance drops linearly with the test set size. If the used model is overly expressive, minimizing the empirical risk (Eq. 2) exactly—or close to exactly—will usually result in poor test performance, since the model overfits to the training data. There is a range of measures that can be taken to mitigate this effect; textbooks like Bishop [3] give an overview over general concepts, chapter 7 of Goodfellow et al. [6] gives a comprehensive summary targeted at deep learning. Some widely used concepts are briefly discussed in the following paragraphs.

Model selection techniques choose a model among a hypothesis class which, under some measure, has the closest level of complexity to the given dataset. They alter the form of the loss function ℓ in Eq. 2 over an outer optimization loop (first find a good ℓ , then optimize $L_{\mathcal{D}}$), such that the final optimization on $L_{\mathcal{D}}$ is conducted on an adequately expressive model. This can—but does not need to—constrain the number of variables of the model. In the case of deep neural networks the number of variables can even significantly exceed the number of training examples [8, 19, 20, 7].

If the dataset is not sufficiently representative of the data distribution, an opposite (although not incompatible) approach is to artificially enrich it to match a complex model. *Data augmentation* artificially enlarges the training set by adding transformations/perturbations of the training data. This can range from injecting noise [18, 23] to carefully tuned contrast and colorspace augmentation [8].

Finally, a widely-used provision against overfitting is to add regularization terms to the objective function that penalize the parameter vector w , typically measured by the l_1 or l_2 norm [9]. These terms constrain the magnitude of w . They tend to drive individual parameters toward zero or, in the l^1 case, enforce sparsity [3, 6]. In linear regression, these concepts are known as least-squares and LASSO regularization [21], respectively.

Despite these countermeasures, high-capacity models will often overfit in the course of the optimization process. While the loss on the training set decreases throughout the optimization procedure, the test loss saturates at some point and starts to increase again. This undesirable effect is usually countered by *early stopping* the optimization process, meaning, that for a *given* model, the optimizer is halted if a user-designed early stopping criterion is met. This is complementary to the model and data design techniques mentioned above and does not undo eventual poor design choices of ℓ . It merely ensures that we do not minimize the empirical risk $L_{\mathcal{D}}$ of a given model beyond the point of best generalization. In practice, however, it is often more accessible to ‘early-stop’ a high-capacity model for algorithmic purposes or because of restrictions to a specific model class, and thus preferred or even enforced by the model designer.

Arguably the gold-standard of early stopping is to monitor the loss on a *validation set* [14, 16, 15]. For this, a (usually small) portion of the training data is split off and its loss is used as an estimate of the generalization loss \mathcal{L} (again in the same sense as Eq. 2), leaving less effective training data to define the training loss $L_{\mathcal{D}}$. An ongoing estimate of this generalization performance is then tracked and the optimizer is halted when the generalization performance drops again. This procedure has many advantages, especially for very large datasets where splitting off a part has minor or no effect on the generalization performance of the learned model. Nevertheless, there are a few obvious drawbacks. Evaluating the model on the validation set in regular intervals can be computationally expensive. More importantly, the choice of the *size* of the validation set poses a trade-off: A small validation set has a large stochastic error, which can lead to a misguided stopping decision. Enlarging the validation set yields a more reliable estimate of generalization, but reduces the remaining amount of training data, depriving the model of potentially valuable information. This trade-off is not easily resolved, since it is influenced by properties of the data distribution (the variance Λ introduced in Eq. 3 below) and subject to practical considerations, e.g., redundancy in the dataset.

Recently Maclaurin et al. [11] introduced an interpretation of (stochastic) gradient descent in the framework of variational inference. As a side effect, this motivated an early-stopping criterion based on the estimation of the marginal likelihood, which is done by tracking the change in entropy of the

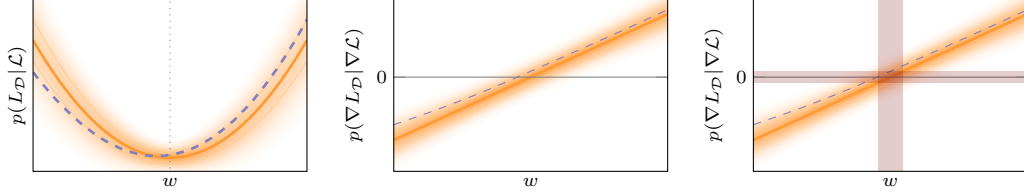


Figure 1: *Sketch of early stopping criterion.* **Left:** marginal distribution of function values defined by left expression in Eq. 3. Mean \mathcal{L} in thick solid orange, ± 1 standard deviations in light orange; pdf as shaded orange. The *full* dataset defines *one* realization of this distribution which is shown in dashed blue (same as $L_{\mathcal{D}}$ of Eq. 2). **Middle:** same as left plot but for corresponding gradients. The pdf is defined by the right expression in Eq. 3 and the corresponding $\nabla L_{\mathcal{D}}$ is shown in dashed blue. **Right:** orange and blue same as middle plot; red shaded area define desired stopping regions (details in text). The vertical red shaded area shows the region of ± 1 standard deviation of possible minima (where $\nabla L_{\mathcal{D}}$ is likely to be zero). If gradients are within this area, the optimization process is halted. This can be translated into a simple stopping criterion (horizontal shaded area, text for details); if gradients are within this area, the optimizer stops.

posterior distribution of w , induced by each optimization step. Since the method requires estimation of the Hessian diagonals, it comes with considerable computational overhead.

The following section motivates and derives a cheap and scalable early stopping criterion which is solely based on local statistics of the computed gradients. In particular, it does not require a held-out validation set, thus enabling the optimizer to use *all* available training data.

2 Model

This section derives a novel criterion for early stopping in stochastic gradient descent. We first introduce notation and model assumptions (§2.1), and motivate the idea of evidence-based stopping (§2.2). Section 2.3 covers the more intuitive case of gradient descent; Section 2.4 extends to stochastic settings.

2.1 Distribution of Gradient Estimators

Let \mathcal{S} be some set of instances sampled independently from $p(x)$. The following holds for any \mathcal{S} , but specifically for the training set \mathcal{D} or a subsampled mini-batch \mathcal{B} and any validation or test set. Using the same notation as in Eq. 2, $L_{\mathcal{S}}(w)$ and $\nabla L_{\mathcal{S}}(w)$ are unbiased estimators of $\mathcal{L}(w)$ and $\nabla \mathcal{L}(w)$ respectively. Since the elements in \mathcal{S} are independent draws from $p(x)$, by the Central Limit Theorem $L_{\mathcal{S}}(w)$ and $\nabla L_{\mathcal{S}}(w)$ are approximately normal distributed according to

$$L_{\mathcal{S}}(w) \sim \mathcal{N}\left(\mathcal{L}(w), \frac{\Lambda(w)}{|\mathcal{S}|}\right) \quad \text{and} \quad \nabla L_{\mathcal{S}}(w) \sim \mathcal{N}\left(\nabla \mathcal{L}(w), \frac{\Sigma(w)}{|\mathcal{S}|}\right) \quad (3)$$

with population (co-)variances $\Lambda(w) = \text{var}_{x \sim p(x)}[\ell(w, x)] \in \mathbb{R}$ and $\Sigma(w) = \text{cov}_{x \sim p(x)}[\nabla \ell(w, x)] \in \mathbb{R}^{D \times D}$, respectively. The (co-)variances of $L_{\mathcal{S}}(w)$ and $\nabla L_{\mathcal{S}}(w)$ both scale inversely proportional to the dataset size $|\mathcal{S}|$. In the population limit $|\mathcal{S}| \rightarrow \infty$, Eq. 3 concentrates on $\mathcal{L}(w)$ and $\nabla \mathcal{L}(w)$. To simplify notation, the indicator (w) will occasionally be dropped: e.g. $L_{\mathcal{S}}(w) =: L_{\mathcal{S}}$.

2.2 When to stop? An Evidence-Based Criterion

The perhaps obvious but crucial observation at the heart of the criterion proposed below is that even the full, but finite, data-set is just a finite-variance sample from a population: By Eq. (3), the *estimators* $L_{\mathcal{D}}$ and $\nabla L_{\mathcal{D}}$ are approximately Gaussian samples around their expectations \mathcal{L} and $\nabla \mathcal{L}$, respectively. Figure 1 provides an illustrative, one-dimensional sketch. The left subplot shows the marginal distribution of function values (Eq. 3, left). The true, but usually unknown, optimization objective \mathcal{L} (Eq. 1), is the mean of this distribution and is shown in solid orange. The objective $L_{\mathcal{D}}$ (Eq. 2), which is optimized in practice and is fixed by the training set \mathcal{D} , defines *one* realization out of this distribution and is shown in dashed blue.

In general, the minimizers of \mathcal{L} and $L_{\mathcal{D}}$ need not be the same. Often, for a finite but large number of parameters $w \in \mathbb{R}^D$, the loss $L_{\mathcal{D}}$ can be optimized to be very small. When this is the case the model tends to overfit to the training data and thus performs poorly on newly generated (test) data $\mathcal{T} \sim p(x)$ with $\mathcal{T} \cap \mathcal{D} = \emptyset$. A widely used technique to prevent overfitting is to stop the optimization process early. The idea is, that variations of training examples which do not contain information for generalization, are mostly learned at the very end of the optimization process where the weights w are fine-tuned. In practice the true minimum of \mathcal{L} is unknown, however the approximate errors of the estimators $L_{\mathcal{D}}$ and $\nabla L_{\mathcal{D}}$ are accessible at every position w . Local estimators for the diagonal of $\Sigma(w)$ have been successfully used before [12, 2] and can be computed efficiently even for very high dimensional optimization problems. Here the variance estimator of the gradient distribution is denoted as $\hat{\Sigma}(w) \approx \text{var}_{x \sim p(x)} [\nabla \ell(w, x)]$ with $\hat{\Sigma}(w) = 1/(|\mathcal{S}|-1) \sum_{x \in \mathcal{S}} (\nabla \ell(w, x) - \nabla L_{\mathcal{S}}(w))^{\odot 2}$, where $\odot 2$ denotes the elementwise square and \mathcal{S} is either the full dataset \mathcal{D} or a mini-batch \mathcal{B} .

Since the minimizers of \mathcal{L} and $L_{\mathcal{D}}$ are not generally identical, also their gradients will cross zero at different locations w . The middle plot of Figure 1 illustrates this behavior. Similar to the left plot, it shows a marginal distribution, but this time over gradients (right expression in Eq. 3). The true gradient $\nabla \mathcal{L}$ is the mean of this distribution and is shown in solid orange. The *one* realization defined by the dataset \mathcal{D} is shown as dashed blue and corresponds to the dashed blue function values $L_{\mathcal{D}}$ of the left plot. Ideally the optimizer should stop in an area in w -space where possible minima are likely to occur, if different datasets of same size were samples from p . In the sketch, this is encoded as the red *vertical* shaded area in the right plot. It is the area around the minimizer of \mathcal{L} where $\nabla \mathcal{L} \pm 1$ standard deviation still encloses zero.

Since $\nabla \mathcal{L}$ is unknown however, this criterion is hard to use in practice, and must be turned into a statement about $\nabla L_{\mathcal{D}}$. Denote the minimizer of \mathcal{L} by $w^* = \arg \min_w \mathcal{L}(w)$ and the population variance of gradients at w^* as $\Sigma^* := \Sigma(w^*)$. A similar criterion that captures this desiderata in essence is to stop when the collected gradients $\nabla L_{\mathcal{D}}$ are becoming consistently very small in comparison to the error Σ^*/M (red *horizontal* shaded area). Close enough to the minima of $L_{\mathcal{D}}$ and \mathcal{L} , the two criteria roughly coincide (intersection of red vertical and horizontal shaded areas). A measure for this is the probability

$$p(\nabla L_{\mathcal{D}} | \nabla \mathcal{L} = 0) = \mathcal{N} \left(\nabla L_{\mathcal{D}}; 0, \frac{\Sigma^*}{M} \right), \quad (4)$$

of observing $\nabla L_{\mathcal{D}}$, were it generated by a true zero gradient $\nabla \mathcal{L} = 0$. This can be seen as the evidence of the trivial model class $p(\nabla \mathcal{L}) = \delta(\nabla \mathcal{L})$, with $p(\nabla L_{\mathcal{D}}) = \int p(\nabla L_{\mathcal{D}} | \nabla \mathcal{L}) p(\nabla \mathcal{L}) d\nabla \mathcal{L}$ (in principal more general models can be formulated, which lead to a richer class of stopping criteria). If gradients $\nabla L_{\mathcal{D}}$ are becoming too small or, ‘too probable’ (stepping into the horizontal shaded area) the gradients are less likely to still carry information about $\nabla \mathcal{L}$ but rather represent noise due to the finiteness of the dataset, then the optimizer should stop. Using these assumptions, the next section derives a stopping criterion for the gradient decent algorithm which then can be extended to stochastic gradient descent as well.

2.3 Early Stopping Criterion for Gradient Descent

When using gradient descent, the whole dataset is used to compute the gradient $\nabla L_{\mathcal{D}}$ in each iteration. Still this gradient estimator has an error in comparison to the true gradient $\nabla \mathcal{L}$, which is encoded in the covariance matrix Σ . In practice Σ is unknown, the variance estimator $\hat{\Sigma}$ described in Section 2.2 however is always accessible. In addition Eq. 4 requires the gradient variance Σ^* at the true minimum which is unknown in practice. Again it can be approximated by $\Sigma(w_t)$ which is the gradient variance at the current position of the optimizer w_t . This is a sensible choice if the optimizer is in convergence and already close to a minimum. Thus, at every position w an approximation to $p(\nabla L_{\mathcal{D}})$ of Eq. 4 is

$$p(\nabla L_{\mathcal{D}}(w)) \approx \prod_{k=1}^D \mathcal{N} \left(\nabla L_{\mathcal{D}}^k(w); 0, \frac{\hat{\Sigma}_k(w)}{M} \right). \quad (5)$$

Though being a simplification, this allows for fast and scalable computations since dimensions are treated independent of each other. To derive an early stopping criterion based only on $\nabla L_{\mathcal{D}}$ we borrow the idea of the previous section that the optimizer should halt when gradients become so small that they are unlikely to still carry information about $\nabla \mathcal{L}$, and combine this with well-known

techniques from statistical hypothesis testing. Specifically: stop when

$$\log p(\nabla L_{\mathcal{D}}) - \mathbf{E}_{\nabla L_{\mathcal{D}} \sim p} [\log p(\nabla L_{\mathcal{D}})] > 0. \quad (6)$$

Here $\mathbf{E}[\cdot]$ is the expectation operator. According to Eq. 6, the optimizer stops when the logarithmic evidence of the gradients is larger than its expected value, roughly meaning that more gradient samples $\nabla L_{\mathcal{D}}$ lie inside of some expected range. In particular, combining Eq. 5 with Eq. 6 and scaling with the dimension D of the objective, gives

$$\frac{2}{D} [\log p(\nabla L_{\mathcal{D}}) - \mathbf{E}_{\nabla L_{\mathcal{D}} \sim p} [\log p(\nabla L_{\mathcal{D}})]] = 1 - \frac{M}{D} \sum_{k=1}^D \left[\frac{(\nabla L_{\mathcal{D}}^k)^2}{\hat{\Sigma}_k} \right] > 0. \quad (7)$$

This criterion (hereafter called EB-criterion, for ‘evidence-based’) is very intuitive; if all gradient elements lay at exactly one standard deviation distance to zero, then $\sum_k (\nabla L_{\mathcal{D}}^k)^2 / \hat{\Sigma}_k = \sum_k \hat{\Sigma}_k / M \cdot \hat{\Sigma}_k = D/M$; thus the left-hand side of Eq. 7 would become zero and the optimizer would stop.

We note on the side that Eq. 7 defines a mean criterion over all elements of the parameter vector w . This implicitly assumes that all dimensions converge in roughly the same time scale such that weighing the fractions $f_k := M \cdot (\nabla L_{\mathcal{D}}^k)^2 / \hat{\Sigma}_k$ equally is justified. If optimization problems deal with parameters that converge at different speeds, like for example different layers of neural networks (or biases and weights inside one layer) it might be appropriate to compute one stopping criterion per subset of parameters which are roughly having similar timescales. In Section 3.4 we will use this slight variation of Eq. 7 for experiments on a multi layer perceptron.

2.4 Stochastic Gradients and Mini-batching

It is straightforward to extend the stopping criterion of Eq. 7 to stochastic gradient descent (SGD); the estimator for $\nabla L_{\mathcal{D}}$ is replaced with an even more uncertain $\nabla L_{\mathcal{B}}$ by sub-sampling the training dataset at each iteration. The local gradient generation is

$$\nabla L_{\mathcal{B}} = \nabla L_{\mathcal{D}} + \eta = \nabla \mathcal{L} + \nu \quad \text{with} \quad \eta \sim \mathcal{N}(0, \Sigma_{\text{obs}}), \nu \sim \mathcal{N}(0, \Sigma/M + \Sigma_{\text{obs}}). \quad (8)$$

Combining this with Eq. 3 yields $\Sigma/M + \Sigma_{\text{obs}} = \Sigma/m$. Thus $\Sigma_{\text{obs}} = \frac{M-m}{mM} \Sigma$. Equivalently to Eq. 4, 5 and 7, this results in an early stopping criterion for stochastic gradient descent:

$$\frac{2}{D} [\log p(\nabla L_{\mathcal{B}}) - \mathbf{E}_{\nabla L_{\mathcal{B}} \sim p} [\log p(\nabla L_{\mathcal{B}})]] = 1 - \frac{m}{D} \sum_{k=1}^D \left[\frac{(\nabla L_{\mathcal{B}}^k)^2}{\hat{\Sigma}_k} \right] > 0. \quad (9)$$

Remark on implementation: Computing the stopping criterion is straight-forward, given that the variance estimate $\hat{\Sigma}$ is available. In this case, it amounts to an element-wise division of the squared gradient by the variance, followed by an aggregation over all dimensions. Balles et al. [2, §4.2] comment on this issue and present a solution for computing $\hat{\Sigma}$ in contemporary software frameworks, that computes the variance estimate implicitly, increasing e.g. the computational cost of a backward pass of a neural network by a factor of about 1.25.

3 Experiments

For proof of concept experiments, we evaluate the EB-criterion on a number of standard classification and regression problems. For illustration and analysis, Sections 3.1 and 3.2 show a least-squares toy problem and large synthetic quadratic problems; Sections 3.3 and 3.4 deal with the more realistic setting of logistic regression on the well-known Wisconsin Breast Cancer Dataset (WDBC) [24] and a multi layer perceptron on the handwritten digits dataset MNIST [10]. Section 3.5 contains experiments for logistic regression, as well as for a shallow neural network on the SECTOR dataset [4]; the SECTOR dataset complements MNIST and WDBC, in the sense, that it has a much less favorable feature-to-datapoint ratio (~ 9); increasing the gains on the generalization performance, when all available training data can be used.

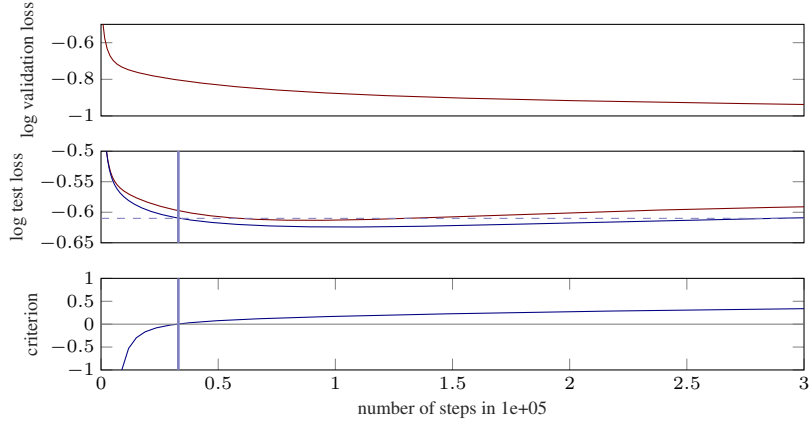


Figure 2: Results for *logistic regression* on the Wisconsin Breast Cancer dataset. Results for the two variants are color-coded; red for validation set-based early stopping, blue for the evidence-based criterion of Eq. 7. The **middle** plot shows test loss versus the number of optimization steps for both methods. The **top** row shows validation loss; since the validation loss decreases over the whole optimization process it does *not* induce a stopping point. The **bottom** row shows the evolution of the stopping criterion, inducing a stopping decision indicated by the blue vertical bar.

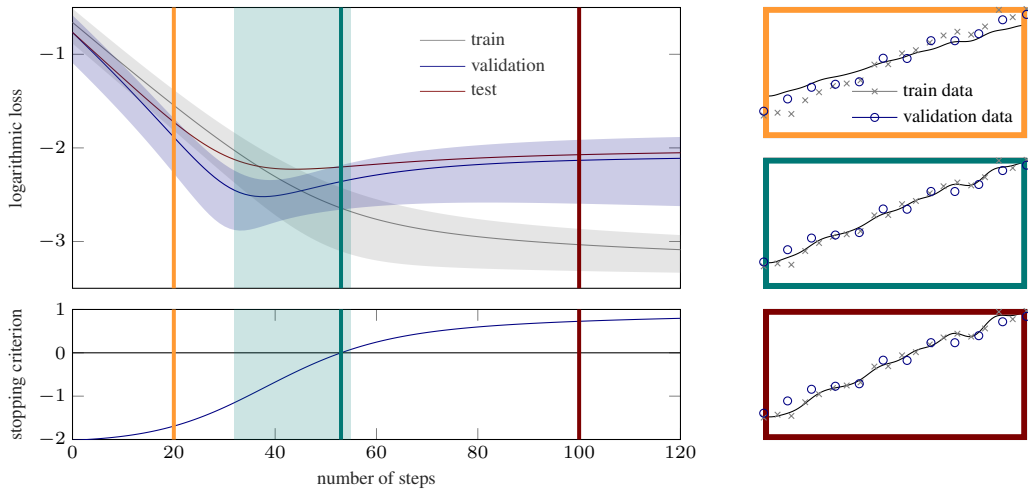


Figure 3: *Least-squares toy problem*. **Top left** logarithmic losses vs. number of optimization steps (colors in legend); shaded areas indicate two standard deviations $\pm 2\sqrt{\Lambda/|S|}$ of the noise loss estimates computed during the optimization (Eq. 3). **Bottom left**: evolution of the EB-criterion (Eq. 7); green vertical bar indicates the induced stopping point. For the steps marked with color-coded vertical bars, the model fit is illustrated on the **right column**; orange iteration: sub-optimal fit ($\hat{y}(w)$ in solid dark blue) to the training data (gray crosses); green iteration: fit, when the EB-criterion of Eq. 7 indicates stopping; red iteration: the model \hat{y} has already overfitted to the training data.

3.1 Linear Least-Squares as Toy Problem

We begin with a toy regression problem on artificial data generated from a one-dimensional linear function y with additive uniform Gaussian noise. This simple setup allows us to illustrate the model fit at various stages of the optimization process and provides us with the true generalization performance, since we can generate large amounts of test data. We use a largely over-parametrized 50-dimensional linear regression model $\hat{y}(w, x) = w^\top \phi(x)$ which contain the ground truth features (bias and linear) and additional periodic features with varying frequency. The features $\phi(x) = [1, x, \sin(a_1 x), \cos(a_1 x), \dots, \sin(a_p x), \cos(a_p x)]^\top$ with $p = 24$ obviously define a massively over-parametrized model for the true function and is thus prone to overfitting. We fit the model by

minimizing the squared error, i.e. the loss function is $\ell(w, (x, y)) = \frac{1}{2}(y - \hat{y}(w, x))^2$. We use 20 samples for training and about 10 for validation, and then train the model using gradient descent. The results are shown in Figure 3; both, validation loss, and the EB-criterion find an acceptable point to stop the optimization procedure, thus preventing overfitting.

3.2 Synthetic Large-Scale Quadratic Problem

We construct synthetic quadratic optimization problems of the form $\mathcal{L}(w) = \frac{1}{2}(w - w^*)^\top B(w - w^*)$, where $B \in \mathbb{R}^{D \times D}$ is a positive definite matrix and $w^* \in \mathbb{R}^D$ is the global minimizer of $\mathcal{L}(w)$; the gradient is $\nabla \mathcal{L} = B(w - w^*)$. In this controlled environment we can test the EB-criterion on different configurations of eigen-spectra, for example uniform, exponential, or structured (a few large, many small eigenvalues); the matrix B is constructed by defining a diagonal matrix $\Gamma \in \mathbb{R}^{D \times D}$ which contains the eigenvalues on its diagonal, and a random rotation $R \in \mathbb{R}^{D \times D}$ which is drawn from the Haar-measure on the D -dimensional uni-sphere [5]; then $B := R\Gamma R^\top$. We artificially define the ‘empirical’ loss $L_{\mathcal{D}}(w)$ by moving the true minimizer w^* by a Gaussian random variable $\zeta_{\mathcal{D}}$, such that $L_{\mathcal{D}}(w) = \frac{1}{2}(w - w^* + \zeta_{\mathcal{D}})^\top B(w - w^* + \zeta_{\mathcal{D}})$ with $\zeta_{\mathcal{D}} \sim \mathcal{N}(0, \Lambda)$. Thus $\nabla L_{\mathcal{D}} = \nabla \mathcal{L} + B\zeta_{\mathcal{D}}$ is distributed according to $\zeta_{\mathcal{D}} \sim \mathcal{N}(0, B\Lambda B^\top)$, and we define $\hat{\Sigma}_{|D|} := \text{diag}(B\Lambda B^\top)$. For experiments we chose $D = 10^3$ as input dimension and zero ($w^* = 0$) as the true minimizer of \mathcal{L} . Figure 4 shows results for three different types of eigen-spectra. The EB-criterion performs well across the different

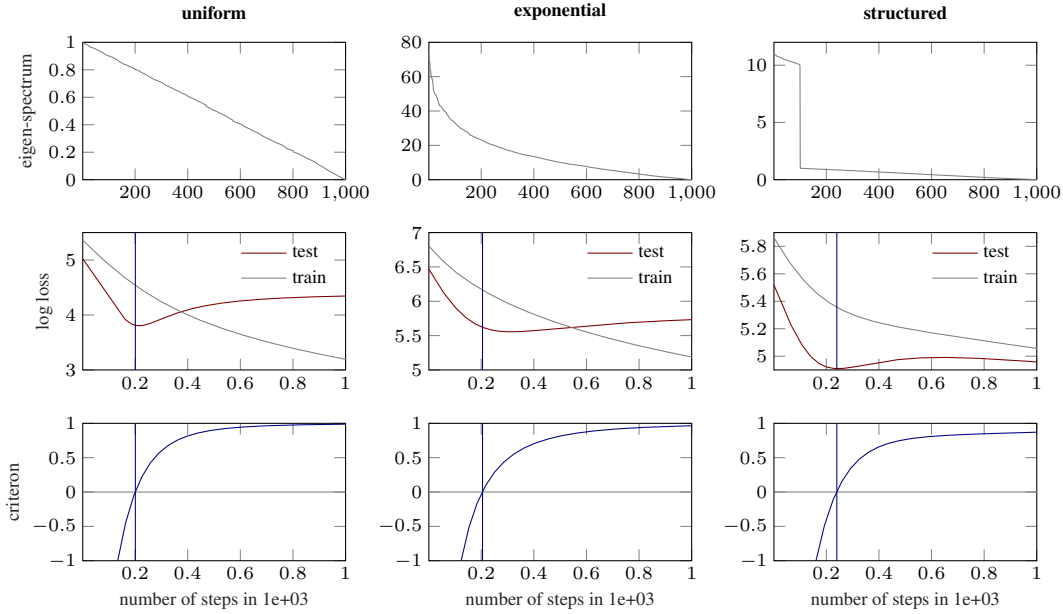


Figure 4: *Synthetic quadratic problem* for three different structures of eigen-spectra: uniform, exponential, structured. **middle row:** logarithmic (exact) test loss in red and train loss in gray; **bottom row:** evolution of the EB-criterion, inducing a stopping decision indicated by the blue vertical bar.

type of partially ill-conditioned problems and induced meaningful stopping decisions; this worked well for different noise levels Λ (Figure 4 shows $\Lambda = 10 \cdot \mathbf{I}$; note that the covariance matrix $B\Lambda B^\top$ of the gradient is dense).

We noticed, however, that another assumption is crucial for the EB-criterion, which might also explain the slightly early stopping decision for the logistic regressor on WBCD (Figure 2 in subsequent section) and full batch GD on MNIST (Figure 7, column 1). Eq. (6) implicitly assumes that (on its path to the minimum of the empirical loss $L_{\mathcal{D}}$) the optimizer passes by a better minimizer with higher generalization performance; this allows to use variances only (in the form of $\hat{\Sigma}$) in the stopping criterion; there is no information about bias (direction of shift $w^* - w_{\mathcal{D}}^*$) because this is fundamentally hard to know.

The assumption is usually well justified, primarily because otherwise early stopping would not be a viable concept in the first place; and second because over-fitting is usually associated with ‘too large’ weights (weights are initialized small; and regularizers that pull weights to zero are often a good idea); on the way from small weights (under-fitting) to too large weights (over-fitting), optimizers usually pass a better point with weights of intermediate size. If the assumption is fundamentally violated the EB-criterion will stop too early. We can artificially construct this setup by initializing the optimizer with weights that lead to an optimization path that does not lead to *any* over-fitting; this is depicted in Figure 5. The setup is identical to the one in Figure 4 (B, w^* as well as $\zeta_{\mathcal{D}}$ and $w_{\mathcal{D}}^*$ are identical); the only difference is the initialization of the weights w_0 for the optimization process. Since—with this initialization—the lowest point of \mathcal{L} that can be reached by minimizing $L_{\mathcal{D}}$ is $w_{\mathcal{D}}^*$, any early stopping decision will lead to under-fitting. In Figure 5 the (exact) test loss flattens out and does not increase again for all three configurations; the assumptions of the EB-criterion are violated and it induces a sub-optimal stopping decision. Figure 6 illustrates these two scenarios in a 2D-sketch.

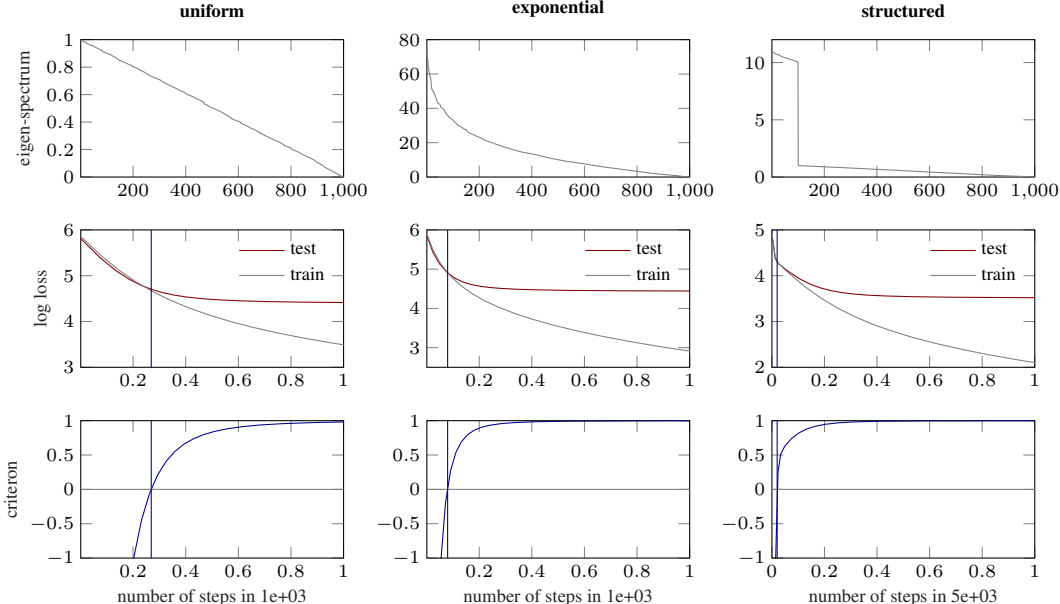


Figure 5: *Synthetic quadratic problem* for three different structures of eigen-spectra; subplots and colors as in Figure 4. Weights are initialized such, that the model can *not* overfit, as can be seen from the exact test loss (red) that flattens out, but does not increase again; the assumptions of the EB-criterion are violated and it induces a sub-optimal stopping decision.

3.3 Logistic Regression on WDBC

Next, we apply the EB-criterion to logistic regression on the Wisconsin Breast Cancer dataset. The task is to classify cell nuclei (described by features such as radius, area, symmetry, et cetera) as either malignant or benign. We conduct a second-order polynomial expansion of the original 30 features (i.e., features of the form $x_i x_j$) resulting in 496 effective features. Of the 569 instances in the dataset, we withhold 369, a relatively large share, for testing purposes in order to get a reliable estimate of the generalization performance. The remaining 200 instances are available for training the classifier. We perform two training runs: one with early stopping based on a validation set of 60 instances (reducing the training set to 140 instances) and one using the full training set and early stopping with the EB-criterion derived in Section 2.3.

If parameters converge at different speeds during the optimization, as indicated in Section 2.3, it is sensible to compute the criterion separately for different subgroups of parameters. Generally, if we split the parameters into N disjoint subgroups $S_i \subset \{1, \dots, D\}$, and denote $D_i = |S_i|$, the criterion reads $\frac{1}{N} \sum_{i=1}^N \left(1 - \frac{M}{D_i} \sum_{k \in S_i} \left[\frac{(\nabla L_{\mathcal{D}}^k)^2}{\hat{\Sigma}_k} \right] \right) > 0$. Since bias and weight gradients usually

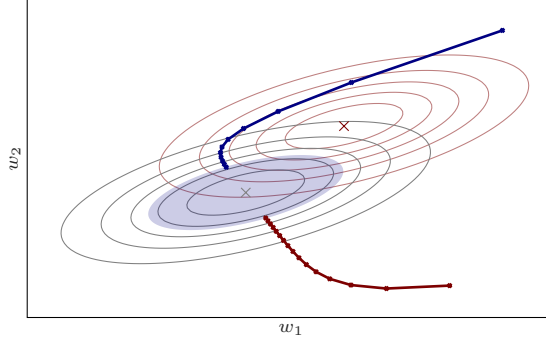


Figure 6: Illustration of implicit *early-stopping assumptions*: Contours of the true loss $\mathcal{L}(w)$ in red; contours of the optimizer’s objective $L_{\mathcal{D}}(w)$ in gray; their minimizers w^* and $w_{\mathcal{D}}^*$ are marked as crosses. The EB-criterion induces a stopping decision, which is roughly described by the blue shaded area. Blue solid line: path of an optimizer that passes by weights of better generalization performance than $w_{\mathcal{D}}^*$; it is stopped by the EB-criterion when it enters the blue shaded area, resulting in better generalization performance. Red solid line: path of an optimizer that can not overfit, since weights were initialized such that $w_{\mathcal{D}}^*$ yields best generalization performance. The assumptions of the EB-criterion are violated, and it thus induces a sub-optimal stopping decision that might lead to under-fitting.

have different magnitudes they converge at different speeds when trained with the same learning rate. For logistic regression, we thus treat the weight vector and the bias parameter of the logistic regressor as separate subgroups. Since the criterion above is noisy we also smooth it with an exponential running average. The results are depicted in the left-most column of Figure 7. The effect of the additional training data is clearly visible, resulting in lower test losses throughout the optimization process. In this scarce data setting the validation loss, computed on a small set of only 60 instances, is clearly misleading (left-most column, top plot). It decreases throughout the optimization process and, thus, fails to find a suitable stopping point. The bottom left plot of Fig. 7 shows the evolution of the EB-criterion. The induced stopping point is not optimal (in that it does not coincide with the point of minimal test loss) but falls into an acceptable region. Thanks to the additional training data, the test loss at the stopping point is lower than any test loss attainable when withholding a validation set.

3.4 Multi-Layer Perceptron on MNIST

For a non-convex optimization problem, we train a multi-layer perceptron (MLP) on the well-studied problem of hand-written digit classification on the MNIST dataset (28×28 gray-scale images). We use a MLP with five hidden layers with 2500, 2000, 1500, 1000 and 500 units, respectively, ReLU activation, and a standard cross-entropy loss for the 10 outputs with soft-max activation (~ 12 million trainable parameters). We treat each weight matrix and each bias vector of the network as a separate subgroup as described in Section 3.3. The MNIST dataset contains 60k training images, which we split into 40k-10k-10k for train, test and validation sets. Again, the criterion is smoothed by an exponential running average.

The results for full-batch gradient descent are shown on Column 1 of Figure 7, and SGD runs with minibatch size 128 and three different learning rates Column 2-4 of the same Figure. The relatively large validation set (10k images) yields accurate estimates of the generalization performance. Consequently, the stopping points more or less coincide with the points of minimal test loss. The reduced training set size leads to only slightly higher test losses. Since the strength of the EB-criterion is to utilize the additional training data and the fact, that also validation losses are only inexact guesses of the generalization error, both of these points thus favor the early stopping criterion based on the validation loss. Still, for all three SGD-runs (columns 2-4 in Figure 7) the EB-criterion performs as good as or better than the validation set induced method. An additional observation is that the quality of the stopping points induced by the EB-criterion varies between the different training configurations. It is thus arguably not as stable in comparison to setups where the validation loss is *very* reliable. For gradient descent (full training set in each iteration, Column 1 of Figure 7), the EB-criterion performs reasonably well, however (an very similarly to the gradient descent runs on the logistic regression on

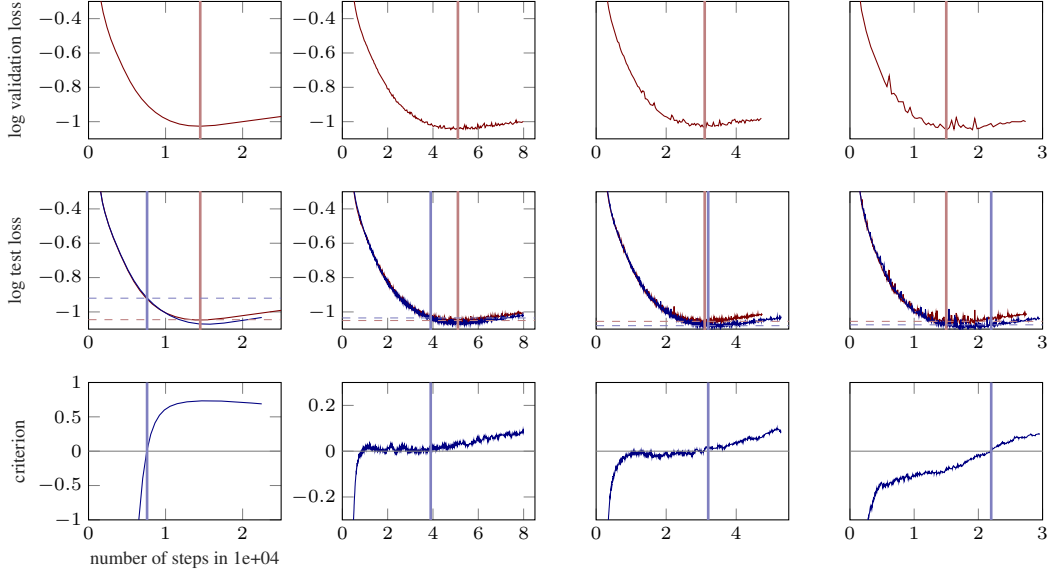


Figure 7: *Multi-layer perceptron* on MNIST: **Column 1**: full batch gradient descent with learning rate 0.01; **columns 2-4** SGD with a mini-batch size of 128 and learning rates 0.003, 0.005 and 0.01, respectively. Results are color-coded: red for validation set-based early stopping, blue for the EB-criterion. **Middle row**: logarithmic test loss versus the number of optimization steps for both methods; **top row** logarithmic validation loss; minimal point induces a stopping decision (red vertical bar); **bottom row**: evolution of the EB-criterion, stopping decision as blue vertical bar; details in text.

WDBC in Figure 2) chooses to stop a bit too early, and thus does result in a slightly worse test set performance. The difference is not very much (test loss red: $10^{-1.04}$, blue $10^{-0.92}$) but it also clearly does not outperform the nearly exactly positioned stopping point induced by this well calibrated validation loss.

3.5 Logistic Regression and Shallow-Net on SECTOR

Finally, we trained a logistic regressor and a shallow fully-connected neural network on the SECTOR dataset[4]. It contains 6412 training and 3207 test datapoints with 55 197 features each, thus having a less favorable feature-to-datapoint ratio than for example MNIST (784 features vs. 60 000 datapoints). The features are extracted from web-pages of companies and the classes describe 105 different industry sectors. The shallow network has one hidden layer with 200 hidden units; the logistic regressor, thus contains ~ 5.8 million, and the shallow net ~ 11.1 million trainable parameters. Experiments are set up in the same style as the ones in Section 3.3 and 3.4. We use 20% of the training data for the validation set; this yields 1282 validation examples and a reduced number of 5130 training examples. Figure 8 shows results; columns 1-2 for the logistic regressor and columns 3-4 for the shallow net. Since the size of the dataset is quite small, the gap between test losses is quite large (middle row, full training set (blue), reduced train set, due to validation split (red)). Both architectures do not overfit properly, the test loss rather flattens out, although we trained both architectures for very long ($2.5 \cdot 10^5$ steps) and initialized weights close to zero. The EB-criterion is again a bit too cautious, and induces stopping when the test loss starts to flatten out; but since it allows utilization of *all* training data, it beats the validation set on both architectures.

3.6 Greedy Element-wise Stopping

For the EB-criterion, we compute $f_k = m(\nabla L_B^k)^2 / \hat{\Sigma}_k$ for each gradient element k . This quantity can be understood as a ‘signal-to-noise ratio’ and the EB-criterion takes the mean over the individual f_k . As a side experiment, we employ the same idea in an element-wise fashion: we stop the training for an individual parameter $w_k \in \mathbb{R}$ (not to be confused with the full parameter vector $w_t \in \mathbb{R}^D$

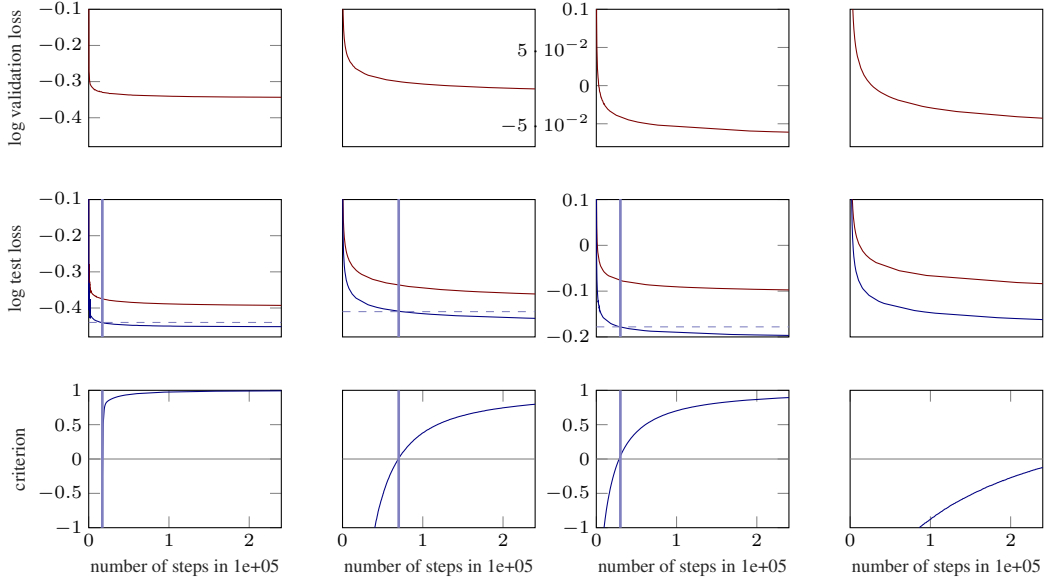


Figure 8: **Columns 1-2:** *Logistic regression* on SECTOR; SGD with batch size 128 and learning rates 0.03 and 0.003 respectively; **Columns 3-4:** *Shallow net* on SECTOR; SGD with batch size 128 and learning rates 0.03 and 0.003 respectively. Plots and colors as in Figure 7; text for details.

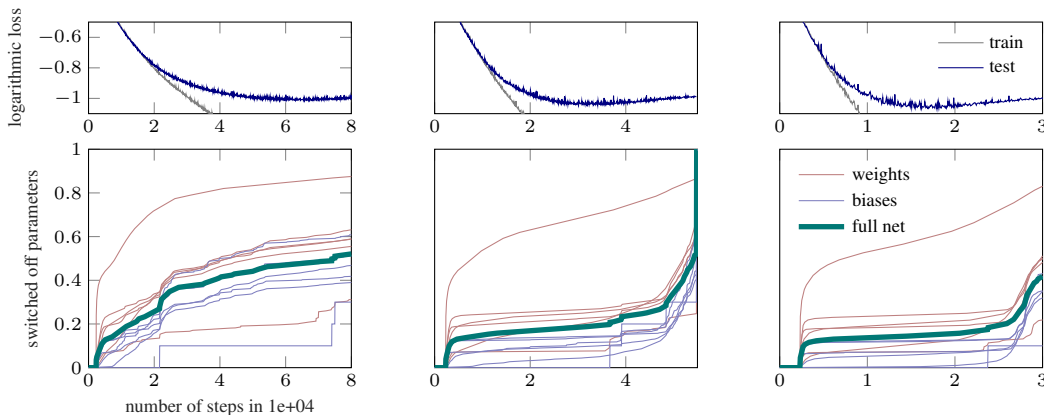


Figure 9: *Greedy element-wise stopping* for a multi-layer perceptron on MNIST. **Columns:** SGD with batch size 128 and learning rates 0.003, 0.005 and 0.01, respectively. **Top row** logarithmic training (gray) and test loss (blue). **Bottom row** fraction of weights where learning has been shut off by the greedy element-wise stopping; each weight matrix (red), each bias vector (blue), full net (green).

at iteration t) as soon as f_k falls below the threshold. Importantly, this is *not* a sparsification of the parameter vector, since w_k is not set to zero when being switched off but merely fixed at its current value. We smooth successive f_k over multiple steps using an exponential moving average; these averages are initialized at high values, resulting in a warm-up phase where all weights are ‘active’. Figure 9 presents results; intriguingly, immediately after the warm-up phase the training of a considerable fraction of all weights (10 percent or more, depending on the training configuration) is being stopped. This fraction increases further as training progresses. Especially towards the end where overfitting sets in, a clear signal can be seen; the fraction of weights where learning has been stopped suddenly increases at a higher rate. Despite this reduction in effective model complexity, the network reaches test losses comparable to our training runs without greedy element-wise stopping (test losses in Figure 7). The fraction of switched-off parameters towards the end of the optimization process reaches up to 80 percent in a single layer and around 50 percent for the whole net.

4 Conclusion

We presented the EB-criterion, a novel approach to the problem of determining a good point for early-stopping in gradient-based optimization. In contrast to existing methods it does not rely on a held-out validation set and enables the optimizer to utilize all available training data. We exploit fast-to-compute statistics of the observed gradient to assess when it represents noise originating from the finiteness of the training set, instead of an informative gradient direction. The presented method so far is applicable in gradient descent as well as stochastic gradient descent settings and adds little overhead in computation, time, and memory consumption. In our experiments, we presented results for linear least-squares fitting, logistic regression and a multi-layer perceptron, proving the general concept to be viable. Furthermore, preliminary findings on element-wise early stopping open up the possibility to monitor and control model fitting with a higher level of detail.

References

- [1] L. Balles and P. Hennig. Follow the Signs for Robust Stochastic Optimization. *ArXiv e-prints*, May 2017.
- [2] L. Balles, J. Romero, and P. Hennig. Coupling Adaptive Batch Sizes with Learning Rates. *ArXiv e-prints*, Dec. 2016.
- [3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [4] C.-C. Chang and C.-J. Lin. *LIBSVM: A library for support vector machines*, 2011. URL <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html>.
- [5] P. Diaconis and M. Shahshahani. The subgroup algorithm for generating uniform random variables. *Probability in Engineering and Informational Sciences*, 1(15-32):40, 1987.
- [6] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- [7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 25, pages 1097–1105, 2012.
- [9] A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 4, pages 950–957, 1991.
- [10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [11] D. Maclaurin, D. Duvenaud, and R. P. Adams. Early stopping is nonparametric variational inference. Technical Report arXiv:1504.01344 [stat.ML], 2015.
- [12] M. Mahsereci and P. Hennig. Probabilistic line searches for stochastic optimization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 28, pages 181–189, 2015.
- [13] J. Martens. New perspectives on the natural gradient method. *CoRR*, abs/1412.1193, 2014. URL <http://arxiv.org/abs/1412.1193>.
- [14] N. Morgan and H. Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, pages 630–637. MIT Press, 1989.
- [15] L. Prechelt. *Early Stopping — But When?*, pages 53–67. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35289-8. doi: 10.1007/978-3-642-35289-8_5.
- [16] R. Reed. Pruning algorithms—a survey. *IEEE transactions on Neural Networks*, 4(5):740–747, 1993.
- [17] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, Sep. 1951.
- [18] J. Sietsma and R. J. Dow. Creating artificial neural networks that generalize. *Neural networks*, 4(1):67–79, 1991.

- [19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition". *CoRR*, abs/1409.1556, 2014.
- [20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [21] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, pages 267–288, 1996.
- [22] T. Tieleman and G. Hinton. *RMSprop Gradient Optimization*, 2015. URL http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
- [23] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096–1103. ACM, 2008.
- [24] W. H. Wolberg, W. N. Street, and O. L. Mangasarian. *UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set*, Jan. 2011. URL [http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)).

—Supplements—

5 Comparison to RMSPROP

This Section explores the differences and similarities of SGD+EB-criterion and RMSPROP. This is rather meant as a means for gaining a better intuition, and not for comparing them as competitors; both methods were derived for different purposes and could be combined in principle.

5.1 Non-Greedy Elementwise EB-Criterion

The *non-greedy* elementwise EB-criterion can be formulated as

$$\begin{aligned} c_t &= \beta c_{t-1} + (1 - \beta) (1 - f_t^{\text{EB-crit}}) \\ w_{t+1} &= w_t - \alpha \cdot \mathbb{I}[c_t \leq 0] \odot \nabla L_{\mathcal{B}}(w_t) \end{aligned} \quad (10)$$

for some conservative smoothing constant $\beta \in (0, 1)$, usually $\beta \approx 0.999$, or 0.99, learning rate α , and the fraction $f_t^{\text{EB-crit}} := |\mathcal{B}|[\nabla L_{\mathcal{B}}(w_t) \odot \hat{\Sigma}(w_t)]$ as defined in Section 3.6. The symbol ‘ \odot ’ denotes elementwise division and $\mathbb{I}[\cdot]$ is the indicator function. In contrast to the greedy implementation of Section 3.6, where switched-off learning rates stayed switches off, Eq. 10 allows learning to be switched on again.

5.2 Learning Rate Damping in RMSPROP

RMSPROP [22] is a well known optimization algorithm that scales learning rates elementwise by an exponential running average of gradient magnitudes; specifically:

$$\begin{aligned} v_t &= \gamma v_{t-1} + (1 - \gamma) \nabla L_{\mathcal{B}}(w_t) \odot^2 \\ w_{t+1} &= w_t - \alpha \nabla L_{\mathcal{B}}(w_t) \oslash \sqrt{v_t}, \end{aligned} \quad (11)$$

again for some smoothing constant $\gamma \in (0, 1)$, usually $\gamma \approx 0.95$, and learning rate α . Let z_t^{\max} be the largest element of the factor $z_t := 1 \oslash \sqrt{v_t}$, then the second line of Eq. 11 can be rewritten as

$$w_{t+1} = w_t - \alpha z_t^{\max} \left(\frac{z_t}{z_t^{\max}} \right) \odot \nabla L_{\mathcal{B}}(w_t). \quad (12)$$

The fraction $f_t^{\text{RMSPROP}} := (z_t / z_t^{\max}) \in (0, 1]$ describes the scaling of learning rates relative to the largest one: if the i^{th} element of f_t^{RMSPROP} is very small, the learning of the corresponding parameter is damped heavily relative to a full step of size αz_t^{\max} . This can be interpreted as ‘switching-off’ the learning of these parameters, similarly to the elementwise EB-criterion.

5.3 Connections and Differences

The following table gives a rough overview over the possible set of learning rates for each method.

method	step size domain	maximal step size	minimal step size
SGD	$\{\alpha\}$	α	α
SGD+EB-crit	$\{0, \alpha\}$	α	0 (only when converged)
RMSPROP	$(0, \alpha z_t^{\max}]$	αz_t^{\max}	> 0

The table shows, that SGD+EB-criterion is a very minor variation of SGD, in the sense that it can also set the learning rate to zero, but only for *converged* parameters to prevent overfitting. It does not improve the convergence properties of SGD while it is still training, since the sizes of the ‘active’ learning rates remain unchanged. Specifically, it does not explicitly encode curvature, or other geometric properties of the loss.

In contrast to this, RMSPROP also adapts the *absolute value* of the largest possible step at every iteration by a varying factor z_t^{\max} , and scales the other steps relative to it. It is based on the steepest descent direction in w -space, measured by a weighted norm, where the weight matrix is the inverse Fisher information matrix F_t at ever position w_t .² If the learned conditional distribution approximates the true conditional data-distribution well, F_t also approximates the expected Hessian of the loss [13]. RMSPROP thus encodes geometric information, which allows for faster convergence compared to SGD.

Another interpretation of RMSPROP, which in spirit is much closer to the EB-criterion, has recently been formulated by Balles and Hennig [1]. It is possible to associate the RMSPROP-update of Eq. 11 with local gradient and variance estimators, according to

$$-\alpha \nabla L_{\mathcal{B}}(w_t) \oslash \sqrt{v_t} \approx -\alpha \frac{\text{sign}[\nabla \mathcal{L}(w_t)]}{\sqrt{1 + \text{diag}[\Sigma(w_t)] \oslash |\mathcal{B}| \nabla \mathcal{L}(w_t)^{\odot 2}}} \quad (13)$$

since

$$\begin{aligned} \nabla L_{\mathcal{B}}(w_t) &\approx \mathbf{E}_{x \sim p(x)} [\nabla \mathcal{L}_{\mathcal{B}}(w_t)] = \nabla \mathcal{L}(w_t), \quad \text{and} \\ v_t &\approx \mathbf{E}_{x \sim p(x)} [\nabla L_{\mathcal{B}}(w_t)^{\odot 2}] = \nabla \mathcal{L}(w_t)^{\odot 2} + \frac{\text{diag}[\Sigma(w_t)]}{|\mathcal{B}|}. \end{aligned} \quad (14)$$

The fraction on the right hand side of Eq. 13 contains the term $1/\text{snr}_t := \text{diag}[\Sigma(w_t)] \oslash |\mathcal{B}| \nabla \mathcal{L}(w_t)^{\odot 2}$, which closely resembles the inverse of $f_t^{\text{EB-crit}}$. Thus gradients with a small signal-to-noise ratio snr_t get shortened; noise free gradients induce steps of equal(!) size $-\alpha \cdot \text{sign}[\nabla \mathcal{L}(w_t)]$ in every direction (note, that they are independent of the magnitude of $\nabla L_{\mathcal{B}}$); RMSPROP thus can be seen as elementwise stochastic gradient-sign estimators, which are mildly damped if noisy.

We have now explored algebraic, as well as behavioral connections between SGD+EB-criterion and RMSPROP; the following paragraph summarizes the above points and lists some noteworthy distinctions:

Geometry encoding: RMSPROP encodes geometric information about the objective and can be loosely associated with second order methods that perform an approximate diagonal preconditioning at every iteration. Alternatively it can be interpreted as stochastic sign estimator, scaling each step with the inverse gradient magnitude, and damping due to noise. In contrast to this, the EB-criterion is just a mild add-on to SGD; it does not alter learning rates due to curvature or other geometric effects.

Mild damping vs. stopping: The EB-criterion defines a strict threshold, justified by a statistical test, when learning should be terminated. RMSPROP defines a vaguer version, in the sense, that the optimizer should move somewhat ‘less’ into directions of uncertain gradients. Even if the signal-to-noise ratio snr_t falls well below the threshold of the stopping decision induces by the EB-criterion (roughly $\text{snr}_t < 1$), RMSPROP just reduces the step proportional to the inverse if the square root $\sim (1 + 1/\text{snr}_t)^{-1/2}$ (e.g. for $\text{snr}_t = 0.5$ (EB-crit stops), the RMSPROP-step gets reduced by a factor of only $1/\sqrt{3} \approx 0.6$).

²If the loss ℓ can be interpreted as negative log likelihood, this is an approximation to the steepest descent direction in the distribution space, where an approximation to the KL-divergence defines a measure.

Smoothing and bias: The derivation of Eq. 13 omits the geometric smoothing contribution of γ which is present in the RMSPROP-update in Eq. 11. In contrast to this, the EB-criterion relies on local (non-smoothed) computations of $\hat{\Sigma}(w_t)$; this is essential to a stopping decision, since large gradient-samples are usually associated with large variances as well. Smoothing the latter would thus bias learning towards following large gradients; in case of RMSPROP it does bias towards larger steps for high variance samples.

The views presented above, give insight on the internal workings of RMSPROP as well as the EB-criterion. It is apparent, that, even though RMSPROP shortens high variance directions, they do not get damped enough to prevent overfitting the objective to the data.

5.4 Empirical Comparison

For an empirical comparison, we run RMSPROP, SGD with elementwise EB-criterion (as in Eq. 10), and an instance of vanilla SGD on a multi-layer-perception on MNIST, similar to the setup in Section 3.4. For the SGD instance that uses the EB-criterion, the fraction of switched-off parameters is defined as

$$P_t^{\text{EB-crit}} := \frac{1}{D} \sum_{i=1}^D \mathbb{I}[c_{i,t} \leq 0]. \quad (15)$$

The percentage of ‘switched-off’ parameters for RMSPROP can be roughly described as the fraction P_t^{RMSPROP} of parameters, whose f_t^{RMSPROP} (defined in Section 5.2) lie below a threshold $T \in (0, 1)$

$$P_t^{\text{RMSPROP}} := \frac{1}{D} \sum_{i=1}^D \mathbb{I}[f_{i,t}^{\text{RMSPROP}} < T]. \quad (16)$$

The same smoothing factor $\gamma = \beta = 0.99$ was used for both methods, for a meaningful comparison. Figure 10 depicts results; the first row shows training losses (light colors) and test losses (corresponding dark colors) of all three methods. Rows 3-7 show the evolution of P_t^{RMSPROP} for five choices of $T = [10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]$; the second row shows $P_t^{\text{EB-crit}}$. As mentioned above, in contrast to the ‘greedy’ implementation of Section 3.6 (switched-off learning rates, stayed switched-off), and for a more natural comparison to RMSPROP, we allowed learning rates to be switched *on* again as well. The results for P_t^{RMSPROP} and $P_t^{\text{EB-crit}}$ are color coded as in Figure 9 of the main paper: green for the full net, and additionally red for weight matrices and orange for biases per layer.

The test losses of vanilla SGD and SGD+EB-criterion are almost identical, while the training loss of SGD+EB-criterion is a bit more conservative than the one of vanilla SGD; this is expected, since the EB-criterion ideally should not impair generalization performance, but might lead to larger training losses at convergence, due to the overfitting prevention. Already at the beginning of the training SGD+EB-criterion switches off about 10-20% of all learning rates; after that, the fraction increases to about 50% (green line, second row); since the EB-criterion only detects convergence, the curve is quite monotonic, exhibiting not significant jumps.

RMSPROP converges a bit faster, as it is expected. Also the plots for P_t^{RMSPROP} are richer in structure. Especially one layer seems to have significantly smaller learning rates for both, biases and weights, than the other layers. Overall the difference between the largest learning rate and all others tends to roughly increase over the optimization process (especially for $T = 10^{-1}$, green line, last row). There are also significant jumps in all the curves, in contrast to the rather monotonic increasing line of SGD+EB-criterion. This indicates nontrivial scaling of the absolute, as well as relative sizes of learning rates throughout the optimization process; also, no learning rate is smaller than 10^{-5} times the largest one at each iteration (third row, green line at exactly zero).

In the future a combination of both—learning rate scaling and overfitting prevention—i.e. combining the EB-criterion with advanced search direction like RMSPROP, is desirable.

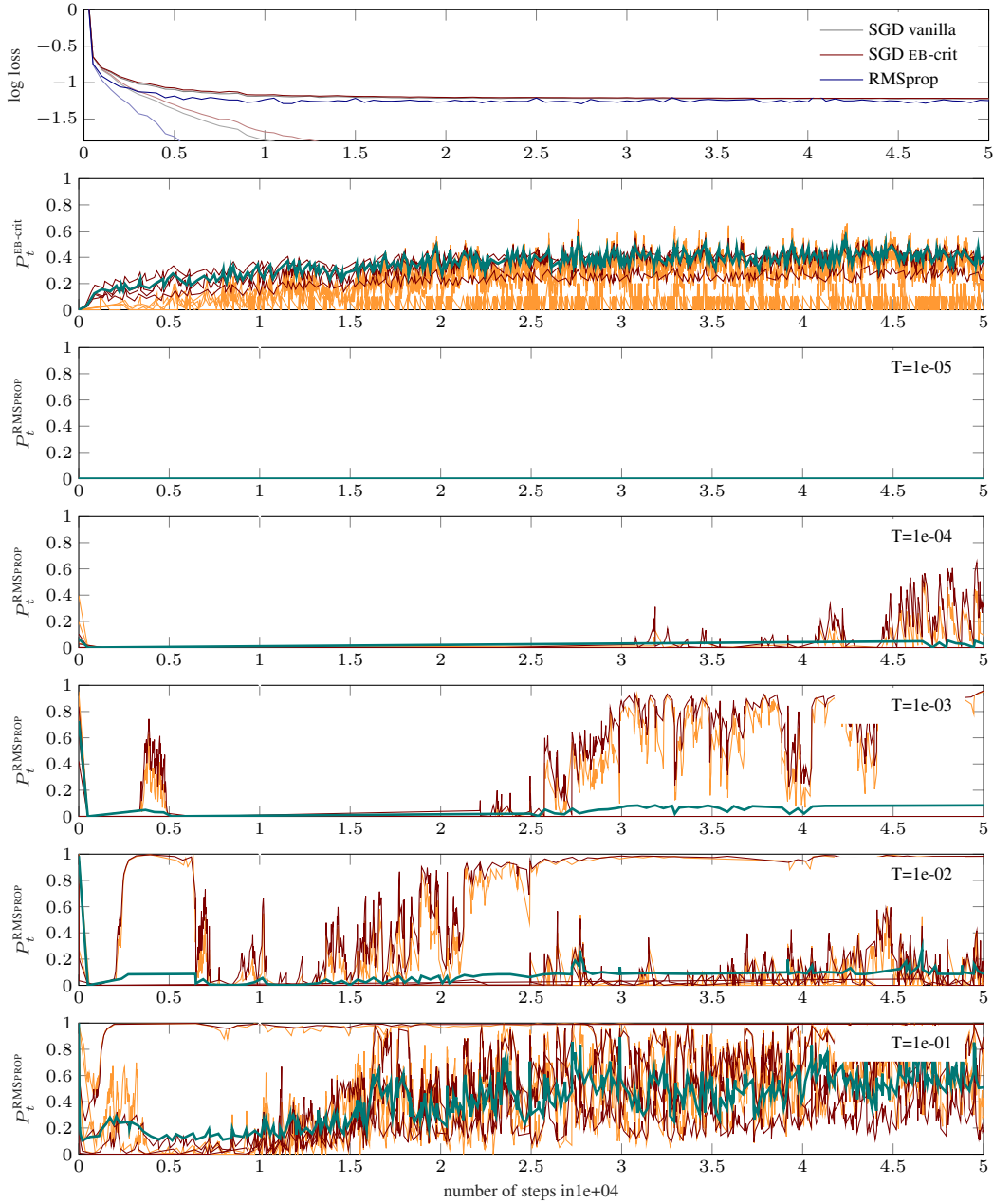


Figure 10: *Comparison of RMSPROP and SGD+EB-criterion on a multi-layer perceptron on MNIST; batch size is 120. **Top row:** logarithmic training loss (light colors) and test loss (corresponding dark colors) for vanilla SGD (gray), SGD+EB-criterion (red) and RMSPROP (blue). **Row 2:** fraction of weights $P_t^{\text{EB-crit}}$ where learning has been shut off by the elementwise stopping; each weight matrix (red), each bias vector (blue), full net (green). **Row 3-7:** same as row 2, but for P_t^{RMSPROP} for different choices of T (see legend).*