# Lab 1B: Gradient Descent

Juan Elenter, Ignacio Hounie and Alejandro Ribeiro[*]

January 22, 2023

## 1 Empirical Risk Minimization

In Section 1A.2.4.2 we considered the least squares prediction of Penn GPAs based on high school GPAs and SAT scores. This is an example of a more general class of problems that are known as empirical risk minimization (ERM).

Consider a generic collection of $N$ data pairs $(x_i, y_i)$ along with predictions $\hat{y} = \Phi(x; w)$. In the function $\Phi(x; w)$ the variable $x$ is an input and $w$ is a parameter to be determined. We say that $\Phi(x; w)$ is a learning parameterization. The goal is to compare outputs $y_i$ with predictions $\hat{y}_i = \Phi(x_i; w)$ so as to find a suitable value of $w$. With this parameter value on hand we can then make predictions $\hat{y} = \Phi(x; w)$ for input variables $x$ for which the output $y$ has not been observed yet.

In order to find suitable values for $w$ we introduce a loss function $\ell(y, \hat{y}) \geq 0$ which we use to evaluate the cost of predicting $\hat{y}$ when the true value realized by the world is $y$. Given that we have $N$ data points $x_i$ for which the true output of the system is known to be $y_i$, we define the empirical risk associated with parameter $w$ as

$$r(w) \; = \; \frac{1}{N} \sum_{i=1}^{N} \ell\big(y_i, \hat{y}_i\big) \; = \; \frac{1}{N} \sum_{i=1}^{N} \ell\Big(y_i, \Phi(x_i; w)\Big) \, . \tag{1}$$

[*]In alphabetical order.

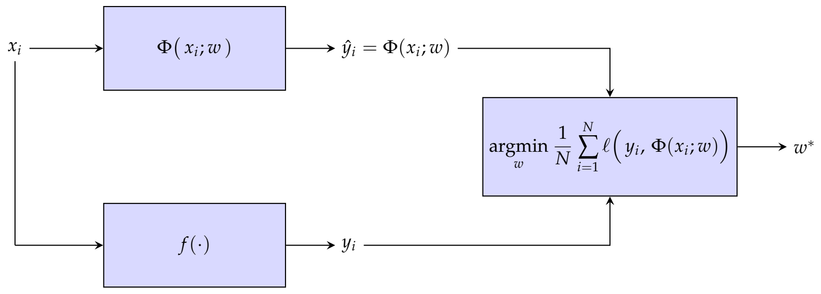
$x_i$
$\Phi(x_i; w)$
$\hat{y}_i = \Phi(x_i; w)$
$\operatorname{argmin}_w \frac{1}{N}\sum_{i=1}^{N} \ell\Big(y_i, \Phi(x_i; w)\Big)$
$w^*$
$f(\cdot)$
$y_i$

**Figure 1.** The system and the AI. The AI attempts to mimic the behavior of the actual system.

The empirical risk $r(w)$ measures the predictive power of coefficient $w$. The quantity $\ell(y_i, \hat{y}_i)$ is always nonnegative and indicates how close the predicted output $\hat{y}_i$ is to the true output $y_i$. The empirical risk averages this metric over all available datapoints. It follows that a natural choice for $w$ is the value that makes the empirical risk smallest. We therefore define the optimal coefficient as the one that solves the following empirical risk minimization (ERM) problem

$$w^* \;=\; \operatorname*{argmin}_w r(w) \;=\; \operatorname*{argmin}_w \frac{1}{N}\sum_{i=1}^{N} \ell\Big(y_i, \Phi(x_i; w)\Big)\,. \tag{2}$$

It is ready to show that linear MMSE is a particular case of (2). We do so in the following task.

**Task 1** Show that the linear MMSE problem is a particular case of the ERM problem in (2). ■

**Solution:** To recover the linear MMSE problem from the ERM problem in (2) define the loss function as $\ell(y, \hat{y}) = (y - \hat{y})^2$ and the learning parameterization as $\Phi(x; w) = w^T x$. With these definitions (2) reduces to,

$$w^* \;=\; \operatorname*{argmin}_w r(w) \;=\; \operatorname*{argmin}_w \frac{1}{N}\sum_{i=1}^{N} \Big(y_i - w^T x_i\Big)^2\,. \tag{3}$$

This the same problem that appears in Section 1A.2.4.2. ■

The ERM problem in (2) is more generic than linear MMSE because we can consider parametrizations that are not linear and losses that are not quadratic.

It is instructive to understand the motivation for (2) that we illustrate in Figure 1. In this figure we represent the real world with the bottom left block. The world is a function $f(\cdot)$ that for a given input $x_i$ produces an output $y_i$. The decision system we represent in the top left takes the same input $x_i$ and produces the output $\hat{y}_i = \Phi(x_i; w)$. This is the block that we can call an artificial intelligence (AI). This is a block that is trying to replicate the behavior of a real system and it is not unwarranted to define intelligence in this way.

The ERM problem is represented by the block on the right. This block compares outputs of the actual system $f(\cdot)$ with outputs of the artificial intelligence $\Phi(\cdot; w)$ and attempts to match them. This process of matching is what we call learning or training. It's goal is to use past experience to find a parameter $w^*$ that we can use to understand future inputs.

## 2 Gradient Descent

To solve (2) we use a gradient descent algorithm. Begin by taking the gradient of the empirical risk $r(w)$ with respect to the parameter $w$. If this parameter has $p$ components, i.e., if $w = [w_1; w_2; \ldots; w_p]$, the gradient of $r(w)$ is defined as

$$g(w) \ := \ \left[ \frac{\partial r(w)}{w_1}; \frac{\partial r(w)}{w_2}; \ldots; \frac{\partial r(w)}{w_p} \right]. \tag{4}$$

It is instructive to consider the first order Taylor's expansion of the risk $r(w)$ around a given point $w_0$,

$$r(w) \ := \ r(w_0) + g^T(w)(w - w_0) + e(w) \tag{5}$$

where $e(w)$ denotes the expansion's residual which is is of order $(w - w_0)^2$ if we assume the risk function $r(w)$ is sufficiently regular.

The expression in (5) is interesting because it shows that moving along the negative gradient can reduce the value of the risk. Indeed, let $\epsilon > 0$ be a sufficiently small scalar and suppose that we start at $w_0$ and move to the point

$$w = w_0 - \epsilon g(w). \tag{6}$$

Particularizing (5) to the variable $w$ in (6) results in the epression

$$\begin{aligned} r(w) \;&:=\; r(w_0) \;+\; g^T(w)(-\epsilon g(w)) \;+\; e(w) \\ &:=\; r(w_0) \;-\; \epsilon\|g(w)\|^2 \qquad\quad +\; e(w). \end{aligned} \tag{7}$$

In (7) the residual is of order $\epsilon^2$ and the squared gradient norm $\|\nabla^T r(w)\|^2$ is nonnegative. Thus, as long as $\epsilon$ is sufficiently small we have that

$$r(w) \leq r(w_0) \tag{8}$$

The observation in (8) motivates the proposals of a *recursive* algorithm to find the minimum of the empirical risk function. In this algorithm we consider an iteration index $k$ at which point the parameter takes the value $w(k)$. The parameter is then updated by following the negative gradient so that the next iterate is

$$w(k+1) = w(k) - \epsilon g(w(k)). \tag{9}$$

Since (8) holds for all iterations $k$, subsequent iterates result in subsequent reductions of the empirical risk $r(w(k))$. This argument can be formalized to prove that under certain regularity assumptions on the empirical risk $w(k)$ converges to $w^*$. In practice, we set a total number of iterations $K$ and use $x(K)$ as an approximation for $w^*$

## 2.1 Gradient Descent for Linear MMSE

For the particular case of linear MMSE the objective to be minimized is the quadratic cost in (3). The gradient $g(w_k)$ is then given by the expression

$$g(w_k) \;=\; \frac{1}{N}\sum_{i=1}^{N} -x_i\Big(\, y_i - w^T(k)x_i \Big)\;. \tag{10}$$

The gradient descent algorithm for this particular case is obtained by substituting (10) into (9) to obtain

$$w(k+1) \;=\; w(k) - \frac{\epsilon}{N}\sum_{i=1}^{N} -x_i\Big(\, y_i - w^T(k)x_i \Big)\;. \tag{11}$$

We address implementation of the recursion in (11) in the following task.

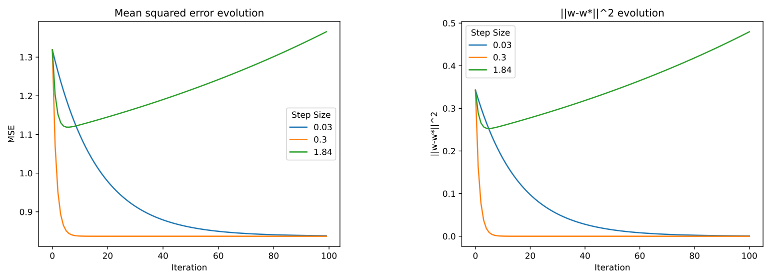

**Figure 2.** Gradient descent. Empirical risk $r(w(k))$ and squared distance to optimality $\|w(k) - w^*\|^2$ are shown as a function of the iteration index $k$ for the gradient descent algorithm implemented in Task 2. The choice of step size $\epsilon$ has a significant effect on the convergence rate towards the optimum.

**Task 2** Implement gradient descent for the data loaded in Task 1A.1. This implementation is complicated by the fact that the range of GPA and SAT scores have vast differences. To reduce this complication we perform data normalization. This requires that we compute mean and variances. For the particular case of SAT scores these are given by

$$\mathrm{S\bar{A}T} \;=\; \frac{1}{N}\sum_{i=1}^{N}\mathrm{SAT}_i, \qquad \mathrm{var(SAT)} \;=\; \frac{1}{N}\sum_{i=1}^{N}\Big(\mathrm{SAT}_i - \mathrm{S\bar{A}T}\Big)^2. \tag{12}$$

Normalized SAT scores for each individual student $i$ are then defined as

$$\mathrm{S\tilde{\bar{A}}T} \;=\; \Big(\mathrm{SAT}_i - \mathrm{S\bar{A}T}\Big)\Big/\Big(\mathrm{var(SAT)}\Big)^{1/2}. \tag{13}$$

The same normalization can be undertaken for high school and Penn GPAs. Implement this normalization and use this normalized data to implement gradient descent. Plot the value of the empirical risk $r(w(k))$ as a function of the iteration index $k$.

Use the expression in Task 1.4.3 to compute $w^*$ for this normalized data. Plot the value $\|w(k) - w^*\|^2$ of the distance between iterates $w(k)$ and the optimal parameter $w^*$. Notice that these plot is possible because we have access to the optimal argument $w^*$. This is not always the case.

This task requires that you try different values for the step sizes $\epsilon$ and the total number of iterations $K$. The plots in Figure 2 show the empirical risk $r(w(k))$ and the squared distance to optimality $\|w(k) - w^*\|^2$ as a function of the iteration index $k$ for different values of $\epsilon$ and $K = 100$. ■

Observe that in Figure 2 the choice of step size $\epsilon$ has a significant effect on the convergence rate towards the optimum. The risk reduction in (8) holds only when the step size $\epsilon$ is sufficiently small. This fact is highlighted in Figure 2 by the curve with $\epsilon = 1.84$. While convergence is guaranteed for all sufficiently small $\epsilon$, a very small stepsize results in slow convergence. This is highlighted in Figure 2 by the curve with $\epsilon = 0.03$.

## 3 Stochastic Gradient Descent

The gradient of the linear MMSE cost which we show in (10) is expressed as an average over dataset entries $(x_i, y_i)$. Indeed, to compute the gradient we evaluate $x_i(y_i - w^T(k)x_i)$ for each data pair $(x_i, y_i)$ and take the average. This is a feature that is common to all empirical risks.

This is true because the empirical risk itself is an average. If we consider the generic expression for $r(w)$ in (1) the gradients of the empirical risk are given by

$$g(w) \;=\; \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial w}\ell\Big(y_i, \Phi(x_i; w)\Big) \,. \tag{14}$$

The gradient $g(w)$ is then an average of the gradients $\partial(\ell(y_i, \Phi(x_i; w))/\partial w$ that correspond to the evaluation of the loss $\ell(y_i, \Phi(x_i; w))$ at each individual data pair $(x_i, y_i)$.

That the gradient is expressed as an average implies that computing gradients of the empirical risk is expensive and unnecessary. It is expensive because we need to compute gradients of the loss associated with each individual data pair $(x_i, y_i)$. When the dataset is large, this is a considerable number of gradients that need to be computed. It is unnecessary because an average can be accurately approximated by considering a smaller number of samples. This average of gradients over a smaller number of samples is known as a stochastic gradient.

For a formal definition consider a random choice of $B$ entries of the dataset and define the stochastic gradient of the empirical risk as

$$\hat{g}(w) \;=\; \frac{1}{B}\sum_{i=1}^{B}\frac{\partial}{\partial w}\ell\Big(y_i, \Phi(x_i; w)\Big) \,. \tag{15}$$

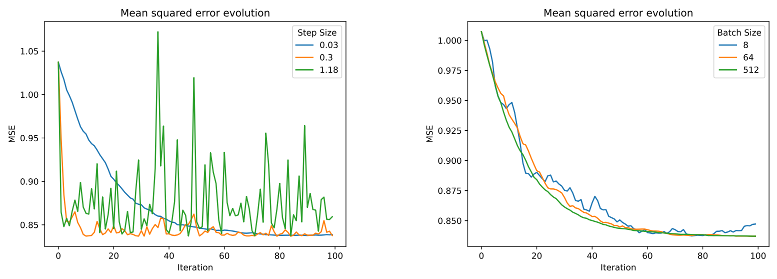

**Figure 3.** Stochastic Gradient Descent. Empirical risk $r(w(k))$ as a function of the iteration index $k$ for the stochastic gradient descent algorithm implemented in Task 3 for different batch sizes and step sizes. Larger batch sizes $B$ and smaller step sizes yield more regular convergence towards the optimum but increases the overall computational cost.

The expression is the same as in (14) except that instead of having a sum over the $N$ entries of the dataset we have a sum over $B$ randomly chosen entries. Although not required by its definition, the presumption is that $B \ll N$ and that, as a consequence, the computational cost of evaluating stochastic gradients is much smaller than the computational cost of evaluating full gradients.

As we did in Section 2 with gradients, we can use stochastic gradients in an iterative algorithm. Consider then an iteration index $k$ and a step size $\epsilon$. At iteration $k$ the parameter value $x(k)$ is updated according to the recursion

$$w(k+1) = w(k) - \epsilon \hat{g}(w(k)). \tag{16}$$

Since we expect that *stochastic* gradients $\hat{g}(w(k))$ are close to the actual gradients $g(w(k))$ we expect *stochastic* gradient descent iteration in (16) to be similar to the gradient descent iteration in (9). Since (9) approaches the optimal parameter $w$, we expect that the same is true of (16). This can be proven to be true.

**Task 3** Implement the stochastic gradient descent algorithm or the normalized scholastic performance data of Task 2. Plot the value of the empirical risk $r(w(k))$ as a function of the iteration index $k$.

This task requires that you try different values step sizes $\epsilon$ and batch sizes. The plots in Figure 2 show the empirical risk $r(w(k))$ as a function of the iteration index $k$ for different values of $\epsilon$ and $B$. ■

In Figure 3 the choice of step size $\epsilon$ and batch size $B$ determine the randomness of the empirical risk curves $r(w(k))$. For smaller batch sizes $B$ and larger step sizes $\epsilon$ the reduction of the empirical risk $r(w(k))$ has more spurious ups and downs. The more regular convergence attained by the use of larger batch sizes $B$ or smaller step sizes $\epsilon$ comes at the cost of increases in the computational cost. Either because each stochastic gradient is more costly to compute – when the batch size increases – or because we need more iterations to converge – when the step size decreases.

## 3.1 The Expected Value of the Stochastic Gradient

In the definition of the stochastic gradient in (15), the entries in the batch are chosen randomly. Thus, we can consider the expected value of the stochastic gradient with respect to the choice of batch. This expectation can be written as

$$\mathbb{E}\left[\hat{g}(w)\right] = \mathbb{E}\left[\frac{1}{B}\sum_{i=1}^{B}\frac{\partial}{\partial w}\ell\Big(y_i, \Phi(x_i; w)\Big)\right] = \frac{1}{B}\sum_{i=1}^{B}\mathbb{E}\left[\frac{\partial}{\partial w}\ell\Big(y_i, \Phi(x_i; w)\Big)\right], \tag{17}$$

where in the second equality we exchanged the expectation and summation operation. Observe now that, by definition, the expectation of the derivative of the loss is the average of the derivatives of the loss over *all* data pairs,

$$\mathbb{E}\left[\frac{\partial}{\partial w}\ell\Big(y_i, \Phi(x_i; w)\Big)\right] \;=\; \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial w}\ell\Big(y_i, \Phi(x_i; w)\Big). \tag{18}$$

If we compare (19) with (14) we see that this is precisely the gradient of the empirical risk,

$$\mathbb{E}\left[\frac{\partial}{\partial w}\ell\Big(y_i, \Phi(x_i; w)\Big)\right] \;=\; g(w). \tag{19}$$

Substituting (19) into (20) and simplifying terms we conclude that the expected value of the stochastic gradient is the gradient itself,

$$\mathbb{E}\left[\hat{g}(w)\right] \;=\; \sum_{i=1}^{B} g(w) \;=\; g(w)\;. \tag{20}$$

This fact gives an alternative explanation of the stochastic gradient algorithm. Having $\mathbb{E}\left[\hat{g}(w)\right] = g(w)$ means that, *on average,* stochastic gradients point in the direction of the gradient. Thus, the stochastic gradient descent iteration in (16) is, *on average,* moving iterates in a direction that reduces the value of the empirical risk. It may be that at some iteration the stochastic gradient takes us in the wrong direction. This does happen. As we see in Figure 3, the empirical risk grows in several iterations. However, in expectation, the stochastic gradient is pointing in the right direction. Therefore, more often than not, we see reductions in the value of the empirical risk. This is why the general trend in Figure 3 is a reduction of the empirical risk. Even though the empirical risk may randomly increase at several iteration indexes.

It is important to be aware (20), because its validity is the actual explanation of why stochastic gradient descent works. In particular, observe that (20) is true for any value of $B$, including $B = 1$. In this case the stochastic gradient is simply

$$\hat{g}(w) \;=\; \frac{\partial}{\partial w}\ell\Big(y_i, \Phi(x_i; w)\Big)\;. \tag{21}$$

In this case we can't argue that $\hat{g}(w)$ is close to $g(w)$ but we can still argue that $\mathbb{E}\left[\hat{g}(w)\right] = g(w)$. It is possible to prove that an implementation of stochastic gradient descent with individual samples as in (21) converges to the optimal argument $w^*$. This fact can also be verified numerically.

## A Appendix: Training Models with PyTorch

In this lab we have introduced ERM problems and we have studied gradient descent and stochastic gradient descent algorithms. The latter is the workhorse algorithm for solving learning problems. It bears emphasizing that we have implemented stochastic gradient descent for a very simple problem in which the loss is quadratic, the parametrization is linear, the data is low dimensional, and the number of data samples available is not too large. As we consider more complex problems with more involved losses, more complex parameterizations, higher dimensional data, and larger datasets, we will encounter three challenges:

(C1) Algorithms for solving ERM are finicky. Stochastic gradient descent can be quite sensitive to the choice of parameters such as step and batch sizes. Improvements upon stochastic gradient descent exist. They can be less finicky but they are more difficult to implement.

(C2) Computing derivatives is time consuming and cumbersome. Linear MMSE is simple and we can easily compute the derivative of the loss with respect to $w$ [cf. (10)]. But when we start looking at neural networks, with several layers and several filters per layers – whatever layers and filters are, which we will learn in due course –, we don't want to go through the trouble of computing derivatives by hand. We want them to be computed automatically.

(C3) Taking advantage of computational resources requires understanding computational hardware. At some point we will encounter problems where use of specialized computation units is necessary. To take advantage of these resources we need to understand the hardware and the operating system. This is knowledge we will not have except for those of you that intend to specialize in hardware and operating systems.

For these reasons it is convenient to use packages that implement the minimization of (2). In this class we will use Pytorch. For that reason, we ask that you complete the following task.

**Task 4** Implement stochastic gradient using PyTorch. ■

## B Report

Do not take much time to prepare a lab report. We do not want you to report your code and we don't want you to report your work. Just give us answers to questions we ask. Specifically give us the following:

| **Question** | **Report deliverable** |
|---|---|
| Task 1 | One paragraph with task solution |
| Task 2 | Plot of $r(w(k))$ versus $k$. Report $\epsilon$ |
| Task 2 | Plot of $\|w(k) - w^*\|^2$ versus $k$. Report $\epsilon$ |
| Task 3 | Plot of $r(w(k))$ versus $k$. Report $\epsilon$ and $B$ |
| Task 4 | Implement stochastic gradient using PyTorch. Plot $r(w(k))$ versus $k$. Report $\epsilon$ and $B$ |

We will check that your answers are correct. If they are not, we will get back to you and ask you to correct them. As long as you submit responses, you get an A for the assignment. It counts for 10% of your lab grade.