Empirical Risk Minimization

Alejandro Ribeiro and Juan Cerviño

September 11, 2020

We formulate Artificial Intelligence (AI) as the extraction of information from observations. Mathematically, we have an observation vector $\mathbf{x} \in \mathbb{R}^n$ out of which we want to extract information in the form of a vector $\mathbf{y} \in \mathbb{R}^m$. Observations and vectors are related by the probability distribution $p(\mathbf{x}, \mathbf{y})$. The AI is a function $\Phi(\mathbf{x})$ that when given an input $\mathbf{x}$ makes a prediction $\hat{\mathbf{y}} = \Phi(\mathbf{x})$ about the value $\mathbf{y}$ that is likely to be the one that was generated by nature according to the distribution $p(\mathbf{x}, \mathbf{y})$. In Figure 1 we are given a picture of a stern-looking man as an input. The AI processes this observation to extract some information out of it. In this case, that this is Prof. Alejandro.

1 Statistical Risk Minimization

We can think of AI as a process of imitating nature. In Figure 1 there is a natural association between the image and the name. The AI is trying to imitate this natural association. To measure the accuracy of the predictions of the AI we consider a loss function $\ell(\mathbf{y}, \hat{\mathbf{y}})$ to measure the cost of making prediction $\hat{\mathbf{y}}$ when the actual value produced by nature is $\mathbf{y}$. This leads to the definition of the optimal estimator as the solution of the statistical risk minimization problem (SRM),

$$\Phi_{\text{S}}^* = \underset{\Phi}{\text{argmin}}\, \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[\ell(\mathbf{y}, \Phi(\mathbf{x}))]. \tag{1}$$

The process of finding the optimal estimator Φ_{S}^* is the process of learning – or training. After the AI is trained we evaluate the optimal classifier to

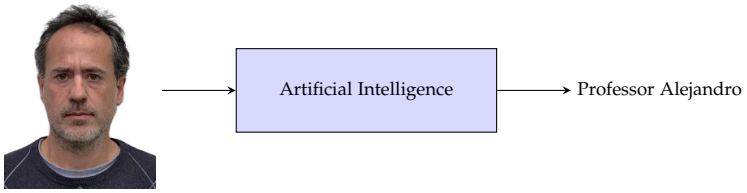

Figure 1. An artificial intelligence (AI) extracts information from observations. In this example, the observation is a picture, from which the AI extracts as information the name of the author.

predict output $\hat{\mathbf{y}} = \Phi_S^*(\mathbf{x})$ whenever we encounter observations $\mathbf{x}$. If the AI has been well trained, these predictions should be close the natural value $\mathbf{y}$.

The training process is illustrated in Figure 2. Given inputs $\mathbf{x}$ nature produces outputs $\mathbf{y}$. For the same input, the AI produces outputs $\hat{\mathbf{y}} = \Phi(\mathbf{x})$. These two outputs are compared using the loss function $\ell(\mathbf{y}, \Phi(\mathbf{x}))$. We search for the function Φ_S^* with minimum cost averaged over the probability distribution $p(\mathbf{x}, \mathbf{y})$ as dictated by (1).

Linear model. For an example model, consider observations $\mathbf{x} \in \mathbb{R}^n$ that are related to information $\mathbf{y} \in \mathbb{R}^m$ according to a linear transformation. To that end, let $p(\mathbf{x})$ be the probability distribution of the input vector $\mathbf{x}$ and let $\mathbf{w} \in \mathbb{R}^m$ be a random noise vector with mean $\mathbb{E}(\mathbf{w}) = \mathbf{0}$ drawn independently from $\mathbf{x}$. Further introduce the matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and assume the relationship between $\mathbf{x}$ and $\mathbf{y}$ follows the model

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w}. \tag{2}$$

The relationship in (2) implicitly defines a joint probability distribution $p(\mathbf{x}, \mathbf{y})$ which we can write down explicitly if we know the probability distributions of the noise vector $\mathbf{w}$ and the input vector $\mathbf{x}$.

Sign model. Another example model is a step version of (2) in which we pass the output through a sign function resulting in an input-output relationship of the form

$$\mathbf{y} = \operatorname{sign}\big(\mathbf{A}\mathbf{x} + \mathbf{w}\big). \tag{3}$$

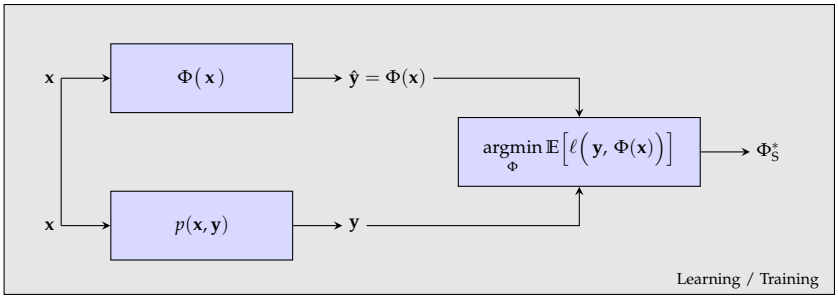

Figure 2. Mathematically, an artificial intelligence (AI) tries to imitate nature. Nature associates observation $\mathbf{x}$ to information $\mathbf{y}$ according to $p(\mathbf{x}, \mathbf{y})$. The AI produces an estimate $\hat{\mathbf{y}} = \Phi(\mathbf{x})$. During training, we search for the optimal estimator function $\Phi^*(\mathbf{x})$ that minimizes the statistical loss defined in (1). After training, the AI predicts information $\hat{\mathbf{y}} = \Phi^*(\mathbf{x})$, whenever it encounters observation $\mathbf{x}$.

The outputs in (3) are either plus or minus one depending on whether $\mathbf{A}\mathbf{x} + \mathbf{w}$ is positive or negative.

Squared norm loss. For an example loss, consider the squared 2-norm $\ell(\mathbf{y}, \hat{\mathbf{y}}) = (1/2)\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$. With this loss function the optimal SRM problem in (1) reduces to

$$\Phi_{\text{S}}^* = \underset{\Phi}{\operatorname{argmin}}\, \mathbb{E}_{p(\mathbf{x},\mathbf{y})} \left[\frac{1}{2} \|\mathbf{y} - \Phi(\mathbf{x})\|_2^2 \right]. \tag{4}$$

In the following questions we consider the linear model in (2) along with the squared norm cost in (4).

1.1 Optimal statistical estimator. In general, the SRM formulation in (4) is something that we just solve numerically. But in this particular case it has a well-known closed form solution. Give an expression for the function Φ_{S}^* of (4). Observe that in (4), $\Phi(\mathbf{x})$ is a variable. This is weird but inconsequential. If you think the answer to this question is ridiculously simple, you are right. We are just getting started.

1.2 Model Generator Function. Write down a function that takes as input the input and output dimensions n and m and returns a matrix $\mathbf{A}$ according to the following model:

(M1) Inputs $\mathbf{x}$ are drawn from a normal distribution with mean $\mathbb{E}(\mathbf{x}) = 0$ and energy $\mathbb{E}(\|\mathbf{x}\|^2|) = 1/2$. Components of $\mathbf{x}$ are independent and identically distributed.

(M2) Noise vectors $\mathbf{w}$ are drawn from a normal distribution with mean $\mathbb{E}(\mathbf{w}) = 0$ and energy $\mathbb{E}(\|\mathbf{w}\|^2|) = 1/2$. Components of $\mathbf{w}$ are independent and identically distributed.

(M3) Entries of $\mathbf{A}$ are binary, i.e. $(\mathbf{A})_{ij} \in \{0, 1\}$. They are drawn independently and identically distributed.

(M4) The energy of the output is $\mathbb{E}(\|\mathbf{y}\|^2|) = 1$.

This is busy work. We will use this function later.

1.3 Linear Model Sample Generator Function. Write a function that takes as input the matrix $\mathbf{A}$ of Question 1.2 and a number of samples Q. The function returns Q pairs of samples $(\mathbf{x}_q, \mathbf{y}_q)$ generated according to (2). Inputs $\mathbf{x}$ and noise $\mathbf{w}$ satisfy model conditions (M1) and (M2) in Question 1.2. More busy work. We will use this function later.

1.4 Sign Model Sample Generator Function. Write a function that takes as input the matrix $\mathbf{A}$ of Question 1.2 and a number of samples Q. The functions returns Q pairs of samples $(\mathbf{x}_q, \mathbf{y}_q)$ generated according to (3). Inputs $\mathbf{x}$ and noise $\mathbf{w}$ satisfy model conditions (M1) and (M2) in Question 1.2. More busy work. We will use this function later.

2 Empirical Risk Minimization

Solving (1) has varying degrees of challenge depending on the complexity of the loss function $\ell(\mathbf{y}, \hat{\mathbf{y}})$ and the distribution $p(\mathbf{x}, \mathbf{y})$. But before we face that challenge, we need to consider a more basic problems, which is the availability of the probability distribution itself. Indeed, to solve (1) we need to know the probability distribution $p(\mathbf{x}, \mathbf{y})$ relating inputs and outputs. This may be known, as we assumed in Question 1.1, but it very often is not. In the example in Figure 1, for instance, there is no known

law governing the relationship between the space of face images and the names of different people.

Approximating distributions with data. When a model is not available, we resort to the acquisition of data. Formally, we consider availability of Q pairs $(\mathbf{x}_q, \mathbf{y}_q)$ which are drawn independently from the distribution $p(\mathbf{x}, \mathbf{y})$. These are grouped in what we call a training set $\mathcal{T}$ which we write here for future reference,

$$\mathcal{T} = \big\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_Q, \mathbf{y}_Q)\big\}, \quad \text{s. t. } (\mathbf{x}_q, \mathbf{y}_q) \sim p(\mathbf{x}, \mathbf{y}). \tag{5}$$

With samples available, it is possible to approximate the statistical loss that appears in (1) using the law of large numbers to write

$$\mathbb{E}_{p(\mathbf{x},\mathbf{y})}[\ell(\mathbf{y}, \Phi(\mathbf{x}))] \approx \frac{1}{Q} \sum_{q=1}^{Q} \ell(\mathbf{y}_q, \Phi(\mathbf{x}_q)). \tag{6}$$

The sum in the right hand side of (6) is called the empirical risk. Under mild conditions, (6) is a good approximation for the statistical risk, provided the number of sampes Q is sufficiently large.

Empirical Risk Minimization. Given the approximation in (6) we can now think of replacing the SRM problem in (1) with the empirical risk minimization (ERM) problem

$$\Phi_{\text{E}}^{*} = \operatorname*{argmin}_{\Phi} \frac{1}{Q} \sum_{q=1}^{Q} \ell(\mathbf{y}_q, \Phi(\mathbf{x}_q)). \tag{7}$$

We illustrate ERM in Figure 3. The block diagram is similar to the SRM problem in Figure 2 except that we replace the probability distribution block with a a data sample block. We don't have a model of nature, but we can sample nature. Changing the model for samples, also requires that we change the expected risk by the empirical risk in the minimization block.

The ERM problem in (7) is easier to solve than it may seem at first look. We ask that you do that in the following questions.

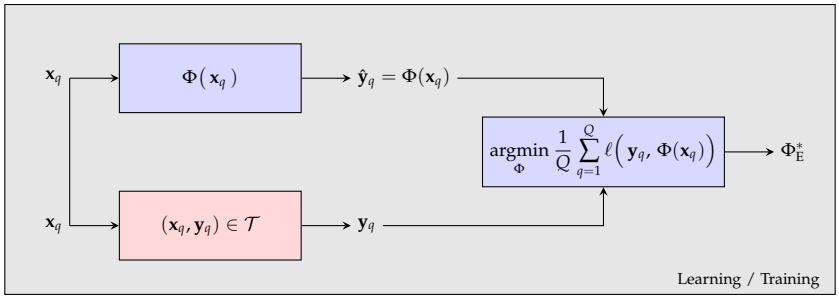

Figure 3. Solving the problem in Figure 2 requires access to the probability distribution $p(\mathbf{x}, \mathbf{y})$. When this model is not available, we resort to data. Instead of operating with the probability distribution $p(\mathbf{x}, \mathbf{y})$ we operate with samples $(\mathbf{x}_q, \mathbf{y}_q)$ that we acquire from nature. As good as this idea is, it does not work unless we restrict the function $\Phi(\mathbf{x})$ to a particular class, as we do in Figure 4.

2.1 Optimal empirical estimator. Consider an arbitrary loss function $\ell(\mathbf{y}, \hat{\mathbf{y}})$ with the condition that $\ell(\mathbf{y}, \hat{\mathbf{y}}) \geq 0$ and $\ell(\mathbf{y}, \hat{\mathbf{y}}) = 0$ when $\mathbf{y} = \hat{\mathbf{y}}$. The problem in (7) admits multiple solutions. Give one such solution. Or, alternatively, give a condition that all such optimal solutions must satisfy.

2.2 The problem defined in (7) is nonsensical. Your answer to Question 2.1 implies that solving (7) gives you no idea on how to predict outputs $\Phi(\mathbf{x})$ for any $\mathbf{x}$ that is not part of the training set $\mathcal{T}$. In turn, this implies that (7) is a nonsensical formulation of AI. Explain.

3 Learning Parametrizations

We have seen that the ERM problem in (7) in not a workable solution to learning with data. This is surprising in light of (6), which we know is true. The reason why we end up with a nonsensical problem when we go from (6) to (7) is that in the latter we are given excessive freedom in the choice of Φ. The solution is to take that freedom away. We therefore introduce a function class $\mathcal{C}$ and require that the function Φ belongs to this

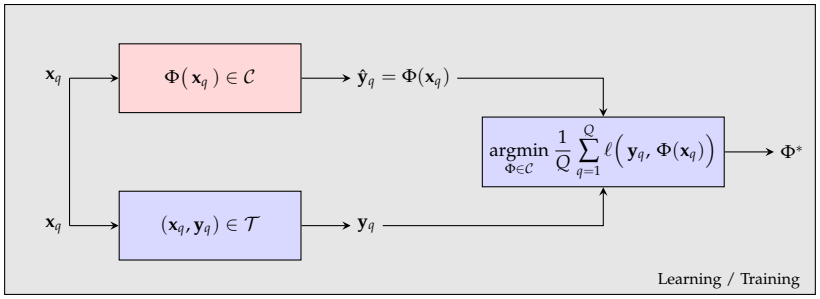

Figure 4. The empirical risk minimization problem in Figure 3 is nonsensical unless we introduce a function class $\mathcal{C}$ to restrict the variability of the function Φ. This block diagram is the true formulation of AI. The diagrams in Figures 2 and 3 are for motivation and contrast.

class. The ERM problem in (7) is therefore replaced by the optimization

$$\Phi^* = \underset{\Phi\in\mathcal{C}}{\operatorname{argmin}} \frac{1}{Q}\sum_{q=1}^{Q} \ell(\mathbf{y}_q, \Phi(\mathbf{x}_q)). \tag{8}$$

In introducing a function class, the idea is to force inputs $\mathbf{x}$ that are similar to inputs $\mathbf{x}_q$ of the training set to yield outputs $\Phi(\mathbf{x})$ that are similar to the output $\Phi(\mathbf{x}_q)$. This is what we mean by "taking freedom away" from the set of possible functions Φ. The crucial word in this paragraph is "similar." an obvious way to define "similar" is to evaluate the distances $\|\mathbf{x}_q - \mathbf{y}_q\|$. But in this course we will see that much richer definitions of what similar means are necessary.

Parametrized empirical risk minimization From a practical perspective we often prefer to make the parametrization more concrete. To that end, introduce a parameter $\mathbf{H} \in \mathbb{R}^p$ so that different choices of $\mathbf{H}$ produce different functions $\Phi(\mathbf{x};\mathbf{H})$ within the class $\mathcal{C}$. We can then rewrite the ERM problem in (8) as

$$\mathbf{H}^* = \underset{\mathbf{H}\in\mathbb{R}^p}{\operatorname{argmin}} \frac{1}{Q}\sum_{q=1}^{Q} \ell\Big(\mathbf{y}_q, \Phi(\mathbf{x}_q;\mathbf{H})\Big). \tag{9}$$

For future reference, observe that in (9) the training set $\mathcal{T}$ is given and the optimization is over parameters $\mathbf{H}$. This motivates definition of the loss

function

$$L(\mathbf{H}) := \frac{1}{Q}\sum_{q=1}^{Q} \ell\Big(\mathbf{y}_q, \Phi(\mathbf{x}_q;\mathbf{H})\Big), \tag{10}$$

which is the objective we are trying to minimize. We say that $L(\mathbf{H})$ is the average loss function. When we want to emphasize the difference with $\ell(\mathbf{y}, \hat{\mathbf{y}})$ we will call the latter, the pointwise loss function.

Linear learning parametrization. As an example, consider the class of linear functions so that $\Phi(\mathbf{x}) = \mathbf{H}\mathbf{x}$ for matrices $\mathbf{H} \in \mathbb{R}^{m\times n} = \mathbb{R}^p$ with $p = m \times n$. Using this particular class in (9) we end up with the program

$$\mathbf{H}^* = \underset{\mathbf{H}\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \frac{1}{Q}\sum_{q=1}^{Q} \ell(\mathbf{y}_q, \mathbf{H}\mathbf{x}_q). \tag{11}$$

This parametrization is not a particularly good and it works well only in simple problems. We are using it to illustrate ideas.

Artificial Intelligence. The similarity between the ERM formulation in (7) and the parametrized ERM formulation in (8) is misleading. For once, (8) is a sensical solution of AI because we can use the solution of (8) to predict outputs associated with inputs $\mathbf{x}$ that are not necessarily part of the training set. Once we have solved (9) and are in possession of the optimal parameter $\mathbf{H}^*$ we can predict $\Phi(\mathbf{x}) = \Phi(\mathbf{x};\mathbf{H}^*)$ whether $\mathbf{x}$ is part of the training set of not. For instance, if we use class of linear functions as in (12), we can predict $\Phi(\mathbf{x}) = \mathbf{H}^*\mathbf{x}$ for any $\mathbf{x}$.

The formulation in (8) is illustrated in Figure 4. It differs from Figure 3 in that Φ is restricted to belong to a certain function class. It differs from Figure 2 in that we use data sampled from nature – as we did in Figure 3 – instead of the distribution model $p(\mathbf{x}, \mathbf{y})$. This is the true formulation of AI. The block diagrams in Figures 2 and 3 were introduced for motivation and contrast. Henceforth, when we speak of AI we refer to the parametrized ERM problem in (8). We will may also call (8) the machine learning (ML) problem.

Model are Necessary in Machine Learning It is important to remark that adding a learning parametrization as in (8) is not something that we

do for convenience. It is a necessity. ML does not make sense without the introduction of a function parametrization. It is also important to remark that as per (8), the design of an ML system is tantamount to the design of the parametrization. For once, the choice of $\mathcal{C}$ is the only degree of freedom that is available to the system's designer. More importantly, the choice of $\mathcal{C}$ determines how we generalize from inputs $\mathbf{x}_q$ we have seen in the training set to inputs $\mathbf{x}$ we have not seen in training set. It defines, as we said before, what it means for inputs $\mathbf{x}$ and $\mathbf{x}_q$ to be similar and for outputs $\mathbf{y}$ and $\mathbf{y}_q$ to be similar.

In going from (1) to (8) we do away with the data model $p(\mathbf{x}, \mathbf{y})$. This leads to an interpretation of ML as being model free. This perspective is incomplete. ML needs a model. Except that instead of making an explicit model of the data, we make a model of the family of functions to which the input-output relationship belongs. This is very important for us, because it is the reason why ML on graphs is different from ML on, say, time signals or images. These signals have different underlying models and they therefore require that we utilize different parametrizations. The meaning of inputs $\mathbf{x}$ and $\mathbf{x}_q$ or outputs $\mathbf{y}$ and $\mathbf{y}_q$ being similar, is different.

We illustrate these point in the following questions.

3.1 A good model. Suppose we use a linear parametrization as in (12) and the loss function is $\ell(\mathbf{y}, \hat{\mathbf{y}}) = (1/2)\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$. We are thus left with the ERM problem

$$\mathbf{H}^* = \underset{\mathbf{H} \in \mathbb{R}^{m \times n}}{\operatorname{argmin}} \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{2} \|\mathbf{y}_q - \mathbf{H}\mathbf{x}_q\|_2^2 \tag{12}$$

Further restrict attention to the linear model in (2). We typically solve these problems numerically. But in this case there is a well known closed form solution. Find it.

Set $n = m = 10^2$ and $Q = 10^3$. Use the function in Question 1.2 to generate a matrix $\mathbf{A}$ and the function in Question 1.3 to generate a training set $\mathcal{T}$ with Q samples that follow the model in (2) for this particular $\mathbf{A}$. Use the samples to find the function $\mathbf{H}^*$ that solves (12) for this particular choice of $\mathbf{A}$. Evaluate the loss $\ell(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{H}^*\mathbf{x}\|_2^2$ averaged over this training set. This should be good. Why?

Use the function of Question 1.3 to generate a testing set $\mathcal{T}'$ of $Q = 10^3$ samples. Evaluate the loss $\ell(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{H}^*\mathbf{x}\|_2^2$ averaged over this training set. This should be good as well. Why?

3.2 A bad model. Consider the ERM formulation in (12). Further restrict attention to the sign model in (3). Find the solution of the resulting ERM problem. There's no trick here. The solution is what you think it is.

Set $n = m = 10^2$ and $Q = 10^3$. Use the function in Question 1.2 to generate a matrix $\mathbf{A}$ and the function in Question 1.4 to generate Q samples that follow the model in (3) for this particular $\mathbf{A}$. Use the samples to find the function $\mathbf{H}^*$ that solves (12) for this particular choice of $\mathbf{A}$. Evaluate the loss $\ell(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{H}^*\mathbf{x}\|_2^2$ averaged over this training set. This should be bad. Why?

Use the function of Question 1.4 to generate a testing set $\mathcal{T}'$ of $Q = 10^3$ samples. Evaluate the loss $\ell(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{H}^*\mathbf{x}\|_2^2$ averaged over this training set. This is still bad. Why?

3.3 An insufficient model. Consider the ERM formulation in (12). Further restrict attention to the linear model in (2). You have already found the solution to this problem in Question 3.1.

Set $n = m = 10^4$ and $Q = 10^3$. Use the function in Question 1.2 to generate a matrix $\mathbf{A}$ and the function in Question 1.3 to generate a training set $\mathcal{T}$ with Q samples that follow the model in (2) for this particular $\mathbf{A}$. Use the samples to find the function $\mathbf{H}^*$ that solves (12) for this particular choice of $\mathbf{A}$. Evaluate the loss $\ell(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{H}^*\mathbf{x}\|_2^2$ averaged over this training set. This should be good. Why?

Use the function of Question 1.3 to generate a testing set $\mathcal{T}'$ of $Q = 10^3$ samples. Evaluate the loss $\ell(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{H}^*\mathbf{x}\|_2^2$ averaged over this training set. This should be bad. Why?

3.4 A reflection. The following statements are true

- In Question 3.1 the loss is small over the training set $\mathcal{T}$ and the test

set $\mathcal{T}'$. This is because the model is accurate and we have sufficient data.

- In Question 3.2 the loss is large over the training set $\mathcal{T}$ and the test set $\mathcal{T}'$. This is because the model is inaccurate.
- In Question 3.3 the loss is small over the training set $\mathcal{T}$ but it is large oved the test set $\mathcal{T}'$. This is because the model is accurate but we don't have sufficient data.

Explain. In large scale problems where either n or m or both are large, we never have sufficient data. This is the problem that in the case of ML on graphs we will solve with graph filters and GNNs.

4 Stochastic Gradient Descent

In Section 3 we used linear parametrizations and quadratic costs. This is one of the few examples in which it is possible to find the optimal estimator $\mathbf{H}^*$ in closed form. Throughout this course we will use numerical optimization to find optimal classifiers. To that end, consider the average loss function $L(\mathbf{H})$ defined in (10) in which $\mathbf{H}$ is the parameter we want to learn in (9). We can compute gradients of this loss function as

$$\nabla L(\mathbf{H}) = \frac{1}{Q} \sum_{q=1}^{Q} \nabla_{\mathbf{H}} \ell\Big(\mathbf{y}_q, \Phi(\mathbf{x}_q; \mathbf{H})\Big). \tag{13}$$

Equipped with gradients it is possible to define a gradient descent method. Introduce then an iteration index t, a step size ϵ, and an initial iterate $\mathbf{H}_0$. Gradient descent produces a sequence of iterates $\mathbf{H}_t$ that follow the recursion

$$\mathbf{H}_{t+1} \;=\; \mathbf{H}_t - \epsilon \nabla L(\mathbf{H}) \;=\; \mathbf{H}_t - \frac{\epsilon}{Q} \sum_{q=1}^{Q} \nabla_{\mathbf{H}} \ell\Big(\mathbf{y}_q, \Phi(\mathbf{x}_q; \mathbf{H})\Big). \tag{14}$$

As t grows, iterates $\mathbf{H}_t$ approach a local minimum of the loss function $L(\mathbf{H})$ if the step size ϵ is small enough.

Stochastic Gradients. The challenge with implementing (14) is that the gradients in (13) are an average over the training set $\mathcal{T}$. When the number of samples Q is large, the cost of computing a gradient can become prohibitive. More important, the cost of computing gradients is also unnecessary because gradients need not be computed with accuracy. It is therefore better to compute stochastic gradients and implement stochastic gradient descent (SGD). In SGD we consider batches of samples that we draw from the training set. At iteration t we consider a batch set $\mathcal{T}_t$ made of $Q_t \ll Q$ elements of the training set. With this subset of samples we compute the stochastic gradients

$$\hat{\nabla} L(\mathbf{H}) = \frac{1}{Q_t} \sum_{(\mathbf{x}_q, \mathbf{y}_q) \in \mathcal{T}_t} \hat{\nabla}_{\mathbf{H}} \ell\Big(\mathbf{y}_q, \Phi(\mathbf{x}_q; \mathbf{H})\Big), \tag{15}$$

which differ from (13) in that the sum is over a subset of the training set. SGD is obtained by replacing the gradients in (14) with stochastic gradients,

$$\mathbf{H}_{t+1} \;=\; \mathbf{H}_t - \epsilon \hat{\nabla} L(\mathbf{H}) \;=\; \mathbf{H}_t - \frac{\epsilon}{Q_t} \sum_{(\mathbf{x}_q, \mathbf{y}_q) \in \mathcal{T}_t} \hat{\nabla}_{\mathbf{H}} \ell\Big(\mathbf{y}_q, \Phi(\mathbf{x}_q; \mathbf{H})\Big). \tag{16}$$

In SGD the sample set $\mathcal{T}_t$ chosen at each iteration is randomly chosen at each iteration. If the samples are independently chosen at each iteration, SGD converges to a local minimum of the loss function $L(\mathbf{H})$; same as gradient descent. An alternative implementation of SGD is to draw batches in a cyclic pattern. This also converges to a local minimum.

Some remarks on stochastic gradient descent We will not study SGD in this course. We will just use it. But I want you to be aware of some facts. We have said that convergence of SGD requires that the step size be sufficiently small, but this is not entirely accurate. There are caveats to what convergence means and there are important technical details.

Further notice that although we motivated SGD by saying that stochastic gradients $\hat{\nabla} L(\mathbf{H})$ are approximations of (deterministic) gradients $\nabla L(\mathbf{H})$, this is not the reason why SGD converges. The SGD method in (16) converges because the expected value of stochastic gradients is a gradient.

Another important point is that if the cost is convex there are no local minima. Thus, if SGD, or gradient descent for that matter, converges to

a local minimum we can claim that this is also a global minimum. The linear model with quadratic loss in (12) is convex, but this is not true in general. Notably, the cost is not convex for neural networks or their convolutional versions. We will nevertheless use SGD and assume that whatever we obtain as an output is the loss minimizer.

Finally, let us be aware that SGD is finicky. You need to try different step sizes and different batch sizes before you encounter ones that start reducing your average cost. It is not a great algorithm. But it is the one we have.

4.1 Stochastic gradient descent for linear model. Consider the ERM problem with linear parametrization and quadratic cost in (12). Particularize (15) and (16) to this example.

4.2 Stochastic gradient descent implementation. Write a function that takes a training set $\mathcal{T}$ as input and implements the SGD algorithm you developed in Question 4.1.

Use the same parameters of Question 3.1 to test that your SGD method is returning a matrix $\mathbf{H}^*$ that is close to the one you know is optimal. Plot the average loss $L(\mathbf{H}_t)$ for the trajectory of your SGD implementation.

4.3 Stochastic gradient descent implementation in pytorch. Pytorch is a software that uses automatic differentiation to compute (stochastic) gradients. Reimplement your SGD method using the Pythorch function `backward` to compute gradients.

5 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**
Question 1.1	Classifier expression.
Question 1.1	Classifier expression.
Question 1.2	Do not report.
Question 1.3	Do not report.
Question 1.4	Do not report.
Question 2.1	Classifier expression.
Question 2.2	Written paragraph.
Question 3.1	Classifier expression. Loss over training set. Loss over test set.
Question 3.2	Classifier expression. Loss over training set. Loss over test set.
Question 3.3	Classifier expression. Loss over training set. Loss over test set.
Question 3.4	Written paragraph.
Question 4.1	Stochastic gradient expression. SGD recursion expression.
Question 4.2	Average loss plot.
Question 4.3	Average loss plot.

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 16% of your final grade.