lions@epfl

EE-556: MATHEMATICS OF DATA: FROM THEORY TO COMPUTATION
LABORATORY FOR INFORMATION AND INFERENCE SYSTEMS
FALL 2024

EPFL

INSTRUCTOR
PROF. VOLKAN CEVHER

HEAD TAS
YONGTAO WU, PEDRO ABRANCHES

HANDOUT 1

We review probability theory and statistics in this handout. We will also have a basic introduction to Stochastic Gradient Descent, an extremely important algorithm for machine learning.

1 Basic Probability

PROBLEM 1: THE MOST USEFUL BOUND IN PROBABILITY THEORY

Recall the definition of a probability measure. In this problem, we will rigorously prove **the** most useful bound in probability theory, the Union Bound, stating that given any n events $E_1, E_2, ..., E_n$, we have

$$P(\cup_{i=1}^n E_i) \leq \sum_{i=1}^n P(E_i).$$

Prove the following statements.

(a) For any two events A and B such that $A \subseteq B$, prove that $P(A) \leq P(B)$.

Solution: Let A, B be any two events. Then $B = (A \cap B) \cup (A^c \cap B)$, where A^c is the complement of the set A. Notice that this is a union of two *disjoint* sets, so we can apply the additivity axiom of probability measures and find that $P(B) = P(A \cap B) + P(A^c \cap B)$. Since $A \subseteq B$, we have $A \cap B = A$. The last step is to apply the positivity axiom stating that $P(C) \geq 0$ for any event C, and therefore $P(B) = P(A) + P(A^c \cap B) \geq P(A)$.

(b) Prove the union bound $P(\cup_{i=1}^n E_i) \leq \sum_{i=1}^n P(E_i)$.

(Hint: For any two events E_1 and E_2, prove that $P(E_1 \cup E_2) \leq P(E_1) + P(E_2)$. For this, you can use a useful decomposition rule of sets: $E_1 \cup E_2 = (E_1 \cap E_2^c) \cup E_2$. Applying this bound recursively will finish the proof.

To use the recursive arguments, you need to write the union of n events as a union of 2 events.)

Solution: Decomposing $E_1 \cup E_2 = E_1 \cup (E_1^c \cap E_2)$ and applying part (a), we get

$$\begin{aligned} P(E_1 \cup E_2) &= P(E_1) + P(E_1^c \cap E_2) \\ &\leq P(E_1) + P(E_2). \end{aligned}$$

For $n > 2$ events, notice that $\cup_{i=1}^n E_i = \left(\cup_{i=1}^{n-1} E_i\right) \cup E_n$. Conclude by induction.

2 "Modeling" in the Absence of Models; Neural Networks as Universal Approximators

Recall that the least-square estimator naturally arises from the maximum likelihood estimation of

A1. a true signal $\mathbf{x}^\natural$,

A2. with a linear model,

A3. of added Gaussian noise.

That is, if we assume that the data $(\mathbf{a}_i, b_i) \in (\mathbb{R}^p \times \mathbb{R})$ is generated by the following relation:

$$b_i = \langle \mathbf{a}_i, \mathbf{x}^\natural \rangle + \mathbf{W}_i, \quad \mathbf{W}_i \sim \mathcal{N}(0, \sigma^2),$$

then

$$\mathbf{x}_{ML}^\star = \arg\min_{\mathbf{x} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$$

where $\mathbf{A}^\top = [\mathbf{a_1}\ \mathbf{a_2}\ \cdots\ \mathbf{a_n}]$.

Although the importance of LS estimators can hardly be overemphasized, it is still widely criticized by its restrictive setting; i.e., assuming A1-3. A natural question is: What if we drop these assumptions? For instance, what if the model is nonlinear, or there is no such thing as a true signal, or the noise is far away from Gaussian?

Neural networks [2] present a powerful framework to address these concerns with the following philosophy: First, denote the (possibly very complicated) relation between input and output by $b = h(\mathbf{a})$, where h is a function from $\mathbb{R}^p \to \mathbb{R}$ and $(\mathbf{a}, b)$ follows some **unknown** distribution $\mathbb{P}$. Then the function h should satisfy

$$h = \arg\min_g \frac{1}{2}\mathbb{E}_{\mathbb{P}}\left(g(\mathbf{a}) - b\right)^2, \quad g \text{ any function.} \tag{1}$$

(**Remark:** We focus on the regression example here, while the very same idea works for classification just as well.)
Now, there are two reasons why solving (1) is not possible:

1. We do not know the distribution $\mathbb{P}$; instead, we only have samples $(\mathbf{a}_i, b_i)$ from $\mathbb{P}$.
2. We do not know how to optimize over the set of all functions.

For the first issue, we already know that we can replace the true average by the empirical average, leading to an empirical risk minimization problem. For the second, the key idea of deep learning relies on the following theorem [3]: Informally speaking,

Any function f can be approximated arbitrarily well by a neural network, as long as the network size is big enough.

Combining these, we are lead to the empirical risk minimization of neural networks:

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \left(b_i - h_{\mathbf{x}}(\mathbf{a}_i)\right)^2 \right\}, \quad h_{\mathbf{x}}(\mathbf{a}) = \sigma\left(\mathbf{W}_k\sigma\left(\cdots\sigma\left(\mathbf{W}_2\sigma\left(\mathbf{W}_1\mathbf{a} + \boldsymbol{\mu}_1\right) + \boldsymbol{\mu}_2\right)\cdots\right) + \boldsymbol{\mu}_k\right); \tag{2}$$

where σ is a "proper" activation function that requires some condition on continuity, $\mathbf{x} = (\mathbf{W}_1, \boldsymbol{\mu}_1, \mathbf{W}_2, \boldsymbol{\mu}_2, \ldots, \mathbf{W}_k, \boldsymbol{\mu}_k)$, $\mathbf{W}_i \in \mathbb{R}^{d_i \times d_{i-1}}$, $\boldsymbol{\mu}_i \in \mathbb{R}^{d_i}$ and $\mathbf{a} \in \mathbb{R}^p$ (more on the notation in the lectures). The hope is that, as long as we have enough parameters and data, the learned neural network is a good approximator for the function h:

$$h_{\mathbf{x}^\star} \simeq h,$$

which is true by the theorem of [3]. There are however trade-offs involved in scaling up over-parametrized networks: width helps robustness; depth helps robustness under a certain initialization but hurts it under another [5].

In conclusion,

Neural networks can be viewed as a "universal modeling" scheme where no assumption about the data distribution is made.

Finally, all the above reasoning relies on the big "if" that we can optimize (2). It is an important fact that one can efficiently compute the (stochastic) **gradients** of (2) via the so-called *backpropagation* [4], and therefore one can run first-order algorithms. The details will be covered in the coming lectures and exercises.

3 Randomness in Statistical Learning Problems, and Stochastic Gradient Descent

PROBLEM 3: RECOGNIZING DIFFERENT RANDOMNESS

There are many different randomness in modern data science or machine learning problems. The purpose of this exercise is to help you get a deeper understanding of them.

This course is all about inferring from data, and the data from real world is often random. Besides this intrinsic randomness, another common source of randomness in modern applications is the *randomized algorithms*. It is extremely important that you have a clear picture of what randomness is truly relevant for statistical inference, and what is only for computational purposes.

Consider the Gaussian linear model from Lecture 1: Let $\mathbf{x}^\natural \in \mathbb{R}^p$ and $\mathbf{A} \in \mathbb{R}^{n\times p}$. We have observations of the form

$$\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}, \tag{3}$$

where $\mathbf{w} \sim \mathcal{N}(0, \sigma^2 I)$ is the Gaussian noise vector. We aim to solve the maximum likelihood estimator

$$\mathbf{x}^\star_{ML} = \arg\min_{x\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{2n}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 \right\}, \tag{4}$$

where we have normalized the the loss function by the number of measurements n (the number $\frac{1}{2}$ is just for convenience later).

(a) So far the only random component is the noise $\mathbf{w}$. Let $\mathbb{E}_{\mathbf{w}}$ denote the expectation with respect to the randomness of $\mathbf{w}$. Compute $\mathbb{E}_{\mathbf{w}}\|\mathbf{b} - \mathbf{A}\mathbf{x}^\natural\|_2^2$.

Solution: $\mathbb{E}_{\mathbf{w}}\|\mathbf{b} - A\mathbf{x}^\natural\|_2^2 = \mathbb{E}_{\mathbf{w}}\|\mathbf{w}\|_2^2 = n\sigma^2$.

(b) In practice, the measurement matrix $\mathbf{A}$ is often random. Assume that the entries of $\mathbf{A}$ are independent random variables with mean 0 and variance 1, and are independent of the noise $\mathbf{w}$. Let $\mathbb{E}_{\mathbf{A}}$ denote the expectation with respect to the randomness of $\mathbf{A}$. Show that

$$\frac{1}{n}\mathbb{E}_A\|\mathbf{A}\mathbf{x}\|_2^2 = \|\mathbf{x}\|_2^2 \tag{5}$$

for all $\mathbf{x} \in \mathbb{R}^p$.
(Hint: Let $\mathbf{a_i}^\top$ be a row of $\mathbf{A}$. What is $\mathbb{E}_{\mathbf{A}}|\langle \mathbf{a_i}, \mathbf{x}\rangle|^2$? Can you compute $\mathbb{E}_{\mathbf{A}}\|\mathbf{A}\mathbf{x}\|_2^2$ through $\mathbb{E}_{\mathbf{A}}|\langle \mathbf{a_i}, \mathbf{x}\rangle|^2$?)

Solution: Since $\|A\mathbf{x}\|_2^2 = \sum_{i=1}^n \langle \mathbf{a_i}, \mathbf{x}\rangle^2$, it suffices to prove that $\mathbb{E}_A|\langle \mathbf{a_i}, \mathbf{x}\rangle|^2 = \|\mathbf{x}\|_2^2$. To show this, we compute

$$\begin{aligned}\mathbb{E}_A\langle \mathbf{a_i}, \mathbf{x}\rangle^2 &= \mathbb{E}_A \mathbf{x}^\top \mathbf{a_i} \cdot \mathbf{a_i}^\top \mathbf{x} \\ &= \mathbf{x}^\top \cdot \mathbb{E}_A \mathbf{a_i}\mathbf{a_i}^\top \cdot \mathbf{x} \\ &= \mathbf{x}^\top \mathbf{x} = \|x\|_2^2\end{aligned}$$

where we have used the fact that $\mathbb{E}_A a_{ij} = 0$, $\mathbb{E}_A a_{ij}^2 = 1$, and each a_{ij} is independent.

(c) Show that the following useful basic inequality holds:

$$\|\mathbf{A}(\mathbf{x}^\star_{ML} - \mathbf{x}^\natural)\|_2^2 \le 2\langle \mathbf{w}, \mathbf{A}(\mathbf{x}^\star_{ML} - \mathbf{x}^\natural)\rangle. \tag{6}$$

(Hint: The maximum likelihood estimator minimizes the loss function, so you can compare the values of the loss function when substituting in $\mathbf{x}^\star_{ML}$ and any other $\mathbf{x}$.)

Solution: This is a simple rearrangement of the inequality

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}^\star_{ML}\|_2^2 \le \|\mathbf{b} - \mathbf{A}\mathbf{x}^\natural\|_2^2,$$

which follows by the definition of the maximum likelihood estimator. Indeed by expanding both the left and right hand side, we find that

$$\begin{aligned}
\|\mathbf{b}\|_2^2 - 2\langle \mathbf{b}, \mathbf{A}\mathbf{x}^\star_{ML}\rangle + \|\mathbf{A}\mathbf{x}^\star_{ML}\|_2^2 &\le \|\mathbf{b}\|_2^2 - 2\langle \mathbf{b}, \mathbf{A}\mathbf{x}^\natural\rangle + \|\mathbf{A}\mathbf{x}^\natural\|_2^2 && \\
\|\mathbf{A}\mathbf{x}^\star_{ML}\|_2^2 - \|\mathbf{A}\mathbf{x}^\natural\|_2^2 &\le 2\langle \mathbf{b}, \mathbf{A}(\mathbf{x}^\star_{ML} - \mathbf{x}^\natural)\rangle && \text{(substracting } \|\mathbf{b}\|_2^2 \text{ and } \|\mathbf{A}\mathbf{x}^\natural\|_2^2\text{)} \\
\|\mathbf{A}\mathbf{x}^\star_{ML}\|_2^2 - \|\mathbf{A}\mathbf{x}^\natural\|_2^2 &\le 2\|\mathbf{A}\mathbf{x}^\natural\|_2^2 + 2\langle \mathbf{A}\mathbf{x}^\natural, \mathbf{A}\mathbf{x}^\star_{ML}\rangle + 2\langle \mathbf{w}, \mathbf{A}(\mathbf{x}^\star_{ML} - \mathbf{x}^\natural)\rangle && \text{(Plugging in } \mathbf{b} = \mathbf{A}\mathbf{x}^\natural + w\text{)} \\
\|\mathbf{A}\mathbf{x}^\star_{ML}\|_2^2 - 2\langle \mathbf{A}\mathbf{x}^\natural, \mathbf{A}\mathbf{x}^\star_{ML}\rangle + \|\mathbf{A}\mathbf{x}^\natural\|_2^2 &\le 2\langle \mathbf{w}, \mathbf{A}(\mathbf{x}^\star_{ML} - \mathbf{x}^\natural)\rangle && \text{(rearranging)} \\
\|\mathbf{A}(\mathbf{x}^\star_{ML} - \mathbf{x}^\natural)\|_2^2 &\le 2\langle \mathbf{w}, \mathbf{A}(\mathbf{x}^\star_{ML} - \mathbf{x}^\natural)\rangle &&
\end{aligned}$$

(d) What we ultimately care about is the estimation error: $\mathbb{E}_{\mathbf{A},\mathbf{w}}\|\mathbf{x}^\star_{ML} - \mathbf{x}^\natural\|_2^2$. In view of (b) and (c), one might be tempted to conclude that

$$\mathbb{E}_{\mathbf{A},\mathbf{w}}\|\mathbf{x}^\star_{ML} - \mathbf{x}^\natural\|_2^2 \le 2\mathbb{E}_{\mathbf{A},\mathbf{w}}\langle \mathbf{w}, \frac{1}{n}\mathbf{A}(\mathbf{x}^\star_{ML} - \mathbf{x}^\natural)\rangle. \tag{7}$$

Please argue carefully why this argument is NOT true.
(Hint: In part (b) we considered a fixed, deterministic $\mathbf{x}$. Is $\mathbf{x}^\star_{ML} - \mathbf{x}^\natural$ deterministic? If not, what randomness does it depend on?)

Solution: The maximum likelihood estimator $\hat{\mathbf{x}}$ depends on the realisation of A and $\mathbf{w}$, and hence cannot be treated as a fixed/deterministic $\mathbf{x}$, as in (b).

(e) We introduce the important Stochastic Gradient Descent (SGD) in the exercise.

Recall Gradient Descent (GD) for minimizing (4):

- Choose $\mathbf{x}_0$ arbitrarily.
- Do $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)$ for some predetermined step-sizes $\alpha_k > 0$.

Recall also that for (4), the gradient at a point $\mathbf{x}$ is $\nabla f(\mathbf{x}) = \frac{1}{n}\mathbf{A}^\top(\mathbf{A}\mathbf{x} - \mathbf{b})$.

Consider a randomized algorithm as follows. At the current iterate $\mathbf{x}_k$, an index $i \in \{1, 2, ..., n\}$ is selected uniformly at random. We then replace the gradient at $\mathbf{x}_k$ by the vector $(\langle \mathbf{a_i}, \mathbf{x}_k\rangle - b_i)\mathbf{a_i}$, where $\mathbf{a_i}^\top$ is the i-th row of A and b_i is the i-th element of $\mathbf{b}$. All other steps are the same as GD. Let $\mathbb{E}_{SGD}$ denote the expectation with respect to the randomness of this algorithm. Show that

$$\mathbb{E}_{SGD}\left(\langle \mathbf{a_i}, \mathbf{x}_k\rangle - b_i\right)\mathbf{a_i} = \frac{1}{n}\mathbf{A}^\top(\mathbf{A}\mathbf{x}_k - \mathbf{b}). \tag{8}$$

That is, the algorithm is a randomized version of Gradient Descent, hence the name.
(Hint: Recall that $\mathbf{a_i}^\top$ is a row of $\mathbf{A}$, and therefore it is a column of $\mathbf{A}^\top$.)

Remark: There are many reasons for using SGD instead of GD; we refer the interested students to [1] for a gentle introduction of SGD. Please do bear in mind that SGD is extremely important in practice. Ever heard of deep learning? AlphaGo? Your interest might be piqued if you know that SGD is the go-to algorithm for these state-of-the-art learning machines.

Solution: Write $\nabla f(\mathbf{x}) = \frac{1}{n}A^\top(A\mathbf{x} - \mathbf{b}) = \frac{1}{n}\sum_{i=1}^n \left(\langle \mathbf{a_i}, \mathbf{x}\rangle - b_i\right)\mathbf{a_i}$. Then

$$\begin{aligned}\mathbb{E}_{SGD}\left(\langle \mathbf{a_i}, \mathbf{x}\rangle - b_i\right)\mathbf{a_i} &= \sum_{i=1}^n P(i)\left(\langle \mathbf{a_i}, \mathbf{x}\rangle - b_i\right)\mathbf{a_i} \\ &= \frac{1}{n}\sum_{i=1}^n \left(\langle \mathbf{a_i}, \mathbf{x}\rangle - b_i\right)\mathbf{a_i}\end{aligned}$$

which is $\nabla f(\mathbf{x})$.

(f) Under the assumptions in (b), show that

$$\mathbb{E}_{\mathbf{A},\mathbf{w}}\left(\langle \mathbf{a_i}, \mathbf{x}\rangle - b_i\right)\mathbf{a_i} = \mathbf{x} - \mathbf{x}^\natural \tag{9}$$

for any $\mathbf{x}$.

Solution: Since $b_i = \langle \mathbf{a_i}, \mathbf{x}^\natural\rangle + \mathbf{w}_i$, we have

$$\begin{aligned}\mathbb{E}_{\mathbf{A},\mathbf{w}}\mathbf{a_i}\left(\langle \mathbf{a_i}, \mathbf{x}\rangle - b_i\right) &= \mathbb{E}_{\mathbf{A},\mathbf{w}}\mathbf{a_i}\mathbf{a_i}^\top(\mathbf{x} - \mathbf{x}^\natural) - \mathbb{E}_{A,\mathbf{w}}\mathbf{w}_i\mathbf{a_i} \\ &= \mathbf{x} - \mathbf{x}^\natural.\end{aligned}$$

(g) Under the assumptions in (b), show that, for any $\mathbf{x}$,

$$\frac{1}{n}\mathbb{E}_{\mathbf{A},\mathbf{w}}\mathbf{A}^\top(\mathbf{A}\mathbf{x} - \mathbf{b}) = \mathbf{x} - \mathbf{x}^\natural. \tag{10}$$

(Hint: There are many ways of proving this. Combining (e) and (f) gives a very simple proof.)

Solution: Using (e) and (f), we may compute

$$\begin{aligned}\frac{1}{n}\mathbb{E}_{\mathbf{A},\mathbf{w}}\mathbf{A}^\top(\mathbf{A}\mathbf{x} - \mathbf{b}) &= \mathbb{E}_{\mathbf{A},\mathbf{w}}\mathbb{E}_{SGD}\left(\langle \mathbf{a_i}, \mathbf{x}\rangle - b_i\right)\mathbf{a_i} \\ &= \mathbb{E}_{SGD}\mathbb{E}_{\mathbf{A},\mathbf{w}}\left(\langle \mathbf{a_i}, \mathbf{x}\rangle - b_i\right)\mathbf{a_i} \\ &= \mathbb{E}_{SGD}(\mathbf{x} - \mathbf{x}^\natural) \\ &= \mathbf{x} - \mathbf{x}^\natural.\end{aligned}$$

4 Multinomial logistic regression and language model training

PROBLEM 4: TOWARDS TRAINING LANGUAGE MODELS

Recall that in the lecture we talk about logistic regression for binary classification problems. Let $\mathbf{x}^\natural \in \mathbb{R}^p$. Let $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^p$ be given. The sample is given by $\boldsymbol{b} := (b_1, \ldots, b_n) \in \{0, 1\}^n$. The classifier $h_\mathbf{x}$ estimates the probability $P(b = 1)$ by outputing a scalar $h_\mathbf{x}$ such that $h_\mathbf{x} := \left[1 + \exp\left(-\langle \mathbf{a}_i, \mathbf{x}\rangle\right)\right]^{-1} = P(b = 1)$.

(a) Show that maximizing the likelihood is equivalent to minimizing the cross-entropy between the real distribution and estimated distribution.

Solution: The objective of maximizing the likelihood is equivalent to minimizing the negative log likelihood:

$$\min_\mathbf{x} -\log \mathrm{p}_\mathbf{x}(\boldsymbol{b}) = -\log\prod_{i=1}^n h_\mathbf{x}^{b_i}(1 - h_\mathbf{x})^{1-b_i} = \left[-\sum_{i=1}^n b_i \log h_\mathbf{x} + (1 - b_i)\log(1 - h_\mathbf{x})\right]$$

Therefore, one can see that for each data sample, we are minimizing the cross-entropy between the real distribution Bern(1) and the estimated distribution Bern($h_{\mathbf{x}}$), when $b_i = 1$, or the cross-entropy between the real distribution Bern(0) and the estimated distribution Bern($h_{\mathbf{x}}$), when $b_i = 0$.

(b) In logistic regression, we only deal with two classes. But in reality, there are many situations where multiple classes are involved. In this case, we need to generalize the logistic regression to multinomial logistic regression (sometimes called the softmax regression).

Suppose there are K classes ($K \geq 2$). The classifier aims to output a vector $\mathbf{h}_{\mathbf{X}} \in \mathbb{R}^K$ to estimate the probability of each class such that its k element is $P(b = k|\mathbf{a})$. Note that in this case, the classifier is parameterized by a learnable matrix $\mathbf{X} \in \mathbb{R}^{K\times p}$ to estimate the probability as follows: $P(b = k|\mathbf{a}) = (\text{Softmax}(\mathbf{Xa}))^{[k]}$, for $k \in [K]$, where Softmax is defined by:

$$\text{Softmax}(\mathbf{Xa}) = \begin{bmatrix} \frac{\exp((\mathbf{Xa})^{[1]})}{\sum_{i=1}^K \exp((\mathbf{Xa})^{[i]})} \\ \vdots \\ \frac{\exp((\mathbf{Xa})^{[K]})}{\sum_{i=1}^K \exp((\mathbf{Xa})^{[i]})} \end{bmatrix} \in \mathbf{R^K}.$$

Again, show that maximizing the likelihood is equivalent to minimizing the cross entropy between the real distribution and the estimated distribution.

Solution: Similar to the previous case of binary classification, we have

$$\min_{\mathbf{X}} -\log \mathrm{p}_{\mathbf{X}}(\boldsymbol{b}) = -\log \prod_{i=1}^n \mathbf{h}_{\mathbf{X}}^{[b_i]} = \sum_{i=1}^n -\log \mathbf{h}_{\mathbf{X}}^{[b_i]} = \sum_{i=1}^n \sum_{j=1}^K -\log \mathbf{h}_{\mathbf{X}}^{[j]} \mathbb{1}_{j=b_i},$$

where the superscript "b_i" indicates the corresponding element for the b_i class in the vector $\mathbf{h}_{\mathbf{X}}$. Hence, we can see that $\sum_{j=1}^K -\log \mathbf{h}_{\mathbf{X}}^{[j]} \mathbb{1}_{j=b_i}$ is exactly the cross-entropy between the real categorical distribution and the estimated categorical distribution.

(c) To some extent, training a language model is essentially tackling a multinomial logistic regression problem where the model needs to predict next word given previous words. A neural network $\mathbf{h}_{\mathbf{x}}$ parameterized by $\mathbf{x}$ is used as an ML estimator, as mentioned in the lecture. Specifically, $\mathbf{h}_{\mathbf{x}}$ aims to predict the t token given $t-1$ tokens.

$$\min_{\mathbf{x}} -\log \mathrm{p}_{\mathbf{x}}(\mathbf{b}_{1:T}) = -\log\left(\prod_{t=1}^T \mathrm{p}_{\mathbf{x}}(\mathbf{b}_t|\mathbf{b}_{1:t-1})\right) = \sum_{t=1}^T \left(-\log \mathrm{p}_{\mathbf{x}}(\mathbf{b}_t|\mathbf{b}_{1:t-1})\right)$$
$$= \sum_{t=1}^T \left(-\log \mathbf{h}_{\mathbf{x}}(\mathbf{b}_{1:t-1})^{[\text{“}\mathbf{b}_t\text{”}]}\right) = \text{cross entropy loss.}$$

Write down the SGD algorithm for training a language model given a corpus that consists of n sentences.

Solution:

SGD for training a language model

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ and step-size $(\alpha_k)_{k\in\mathbb{N}} \in]0, +\infty[^{\mathbb{N}}$.
2. For $k = 0, 1, \ldots$ perform:

- Randomly sample a sentence S_i with embedding $\mathbf{b}_{i,1:T}$ from the corpus, $i \in [n]$.
- Set initial loss $L = 0$.
- Forward pass: **For** $t = 1, \ldots, T$:

$$L+ = (-\log \mathbf{h}_{\mathbf{x}_t}(\mathbf{b}_{i,1:t-1})^{[\text{“}\mathbf{b}_{i,t}\text{”}]})$$

- Backward pass:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \frac{\partial L}{\partial \mathbf{x}}.$$

3. Output the language model with weights $\mathbf{x}$.

References

[1] LÉON BOTTOU *Stochastic Gradient Descent Tricks*, *Neural Networks: Tricks of the Trade*, *page 421-436*, Springer 2012.

[2] SIMON HAYKIN *Neural networks: a comprehensive foundation*, Prentice Hall PTR 1994.

[3] ANDREW BARRON *Universal approximation bounds for superpositions of a sigmoidal function*, *IEEE Transactions on Information theory*, 1993.

[4] DAVID RUMELHART, GEOFFREY HINTON, AND RONALD WILLIAMS *Learning representations by back-propagating errors*, *Cognitive modeling*, 1988.

[5] Zhu, Z., Liu, F., Chrysos, G. & Cevher, V. Robustness in deep learning: The good (width), the bad (depth), and the ugly (initialization). *ArXiv Preprint ArXiv:2209.07263*. (2022)