# EE2211: Spring 2023
# Tutorial 8

1. Suppose we are minimizing $f(x) = x^4$ with respect to $x$. We initialize $x$ to be 2. We perform gradient descent with learning rate 0.1. What is the value of $x$ after the first iteration?

2. Please consider the csv file (government-expenditure-on-education2.csv), which depicts the government's educational expenditure over the years. We would like to predict expenditure as a function of year. To do this, fit an exponential model $f(\mathbf{x}, \mathbf{w}) = \exp(-\mathbf{x}^\top \mathbf{w})$ with squared error loss to estimate $\mathbf{w}$ based on the csv file and gradient descent. In other words, $C(\mathbf{w}) = \sum_{i=1}^{m} (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$.

   Note that even though year is one dimensional, we should add the bias term, so $\mathbf{x} = [1 \text{ year}]^\top$. Furthermore, optimizing the exponential function is tricky (because a small change in $\mathbf{w}$ can lead to large change in $f$). Therefore for the purpose of optimization, divide the "year" variable by the largest year (2018) and divide the "expenditure" by the largest expenditure, so that the resulting normalized year and normalized expenditure variables have maximum values of 1. Use a learning rate of 0.03 and run gradient descent for $2,000,000$ iterations.

   (a) Plot the cost function $C(\mathbf{w})$ as a function of the number of iterations.

   (b) Use the fitted parameters to plot the predicted educational expenditure from year 1981 to year 2023.

   (c) Repeat (a) using a learning rate of 0.1 and learning rate of 0.001. What do you observe relative to (a)?

   The goal of this question is for you to code up gradient descent, so I will provide you with the gradient derivation. First, please note that in general, $\nabla_{\mathbf{w}}(\mathbf{x}^\top \mathbf{w}) = \mathbf{x}$. To see this:

$$\nabla_{\mathbf{w}}(\mathbf{x}^\top \mathbf{w}) = \begin{bmatrix} \frac{\partial(\mathbf{x}^\top \mathbf{w})}{\partial w_1} \\ \frac{\partial(\mathbf{x}^\top \mathbf{w})}{\partial w_2} \\ \vdots \\ \frac{\partial(\mathbf{x}^\top \mathbf{w})}{\partial w_d} \end{bmatrix} = \begin{bmatrix} \frac{\partial(w_1 x_1 + w_2 x_2 + \cdots + w_d x_d)}{\partial w_1} \\ \frac{\partial(w_1 x_1 + w_2 x_2 + \cdots + w_d x_d)}{\partial w_2} \\ \vdots \\ \frac{\partial(w_1 x_1 + w_2 x_2 + \cdots + w_d x_d)}{\partial w_d} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} = \mathbf{x}$$

   The above equality will be very useful for the other questions as well. Now, going back to our question,

$$\begin{aligned} \nabla_{\mathbf{w}} C(\mathbf{w}) &= \nabla_{\mathbf{w}} \sum_{i=1}^{m} (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2 \\ &= \sum_{i=1}^{m} \nabla_{\mathbf{w}} (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2 \\ &= \sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) \nabla_{\mathbf{w}} f(\mathbf{x}_i, \mathbf{w}) \qquad \text{chain rule} \\ &= \sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) \nabla_{\mathbf{w}} \exp(-\mathbf{x}_i^\top \mathbf{w}) \end{aligned}$$

$$= -\sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) \exp(-\mathbf{x}_i^\top \mathbf{w}) \nabla_{\mathbf{w}} (\mathbf{x}_i^\top \mathbf{w}) \quad \text{chain rule}$$
$$= -\sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) \exp(-\mathbf{x}_i^\top \mathbf{w}) \mathbf{x}_i$$
$$= -\sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) f(\mathbf{x}_i, \mathbf{w}) \mathbf{x}_i$$

3. Given the linear learning model $f(\mathbf{x}, \mathbf{w}) = \mathbf{x}^\top \mathbf{w}$, where $\mathbf{x} \in \mathbb{R}^d$. Consider the loss function
$$L(f(\mathbf{x}_i, \mathbf{w}), y_i) = (f(\mathbf{x}_i, \mathbf{w}) - y_i)^4,$$
where $i$ indexes the $i$-th training sample. The final cost function is
$$C(\mathbf{w}) = \sum_{i=1}^{m} L(f(\mathbf{x}_i, \mathbf{w}), y_i),$$
where $m$ is the total number of training samples. Derive the gradient of the cost function with respect to $\mathbf{w}$.

4. Repeat Question 3 using $f(\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{x}^\top \mathbf{w})$, where $\sigma(a) = \frac{1}{1+\exp(-\beta a)}$

5. Repeat Question 3 using $f(\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{x}^\top \mathbf{w})$, where $\sigma(a) = \max(0, a)$

   **Remark 1.** *Strictly speaking, the function $\sigma$ as defined in this question is not differentiable at zero. This function is, however, used widely used as the so-called ReLU (Rectified Linear Unit) activation function in deep neural networks. Hence, it is of great importance in machine learning. Cavalierly, you may take the "derivative" of $\sigma$ here to be*
$$\sigma'(a) = \frac{\mathrm{d}\sigma(a)}{\mathrm{d}a} = \begin{cases} 1 & a \geq 0 \\ 0 & a < 0 \end{cases}.$$

6. Consider the univariate (one-input, one-output) function $C(w) = 5w^2$ and the initial point $w_0 = 2$. Find the learning rate $\eta$ so that gradient descent converges in one step. Find the set of learning rates such that gradient descent diverges. Finally, find the set of learning rates such that gradient descent converges.

7. **(Optional)** The *logistic regression model* (a generalized linear model) posits that the target $y_i \in \{-1, +1\}$ is related to the feature vector $\mathbf{x}_i$ as follows
$$\Pr(y_i \mid \mathbf{x}_i, \mathbf{w}) = g\big(y_i \mathbf{x}_i^\top \mathbf{w}\big) \quad \text{where} \quad g(z) = \frac{1}{1 + \exp(-z)}.$$
Suppose we have a set of training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$. The loss we want to minimize is
$$\min_{\mathbf{w}} \sum_{i=1}^{m} -\log \Pr(y_i \mid \mathbf{x}_i, \mathbf{w})$$
This is the same as minimizing the *logistic loss*
$$\min_{\mathbf{w}} \sum_{i=1}^{m} \mathrm{Logistic}\big(y_i \mathbf{x}_i^\top \mathbf{w}\big) \quad \text{where} \quad \mathrm{Logistic}(z) = \log[1 + \exp(-z)].$$
In *stochastic gradient descent*, which is used to train large-scale networks, we randomly choose one sample, say $(\mathbf{x}_i, y_i)$ and take a step in the negative gradient direction associated to this sample. Show that if we use a learning rate of $\eta$, the update can be expressed as follows:
$$\mathbf{w}_{k+1} = \mathbf{w}_k + \eta y_i \mathbf{x}_i \big[1 - \Pr(y_i \mid \mathbf{x}_i, \mathbf{w}_k)\big].$$

**Remark 2.** *Note that* $\Pr(y_i \mid \mathbf{x}_i, \mathbf{w})$ *is the probability that we predict the training label* $y_i$ *using the features* $\mathbf{x}_i$ *correctly under parameters* $\mathbf{w}$ *and* $1 - \Pr(y_i \mid \mathbf{x}_i, \mathbf{w})$ *is the probability of making a mistake. Hence, we make a larger correction to the current weight vector when the mistake made on the* $i$*-th sample is "larger" in the sense that* $1 - \Pr(y_i \mid \mathbf{x}_i, \mathbf{w})$ *is larger. Hence, this update makes sense. If you are interested in this, a good course to take is MA4270.*

8. **(Optional)** In this problem, we want to show some properties of gradient descent on "nice" functions. First, we have to make a few definitions.

   - We say that $f : \mathbb{R}^d \to \mathbb{R}$ is *convex* if
   $$f(t\mathbf{x} + (1-t)\mathbf{y}) \leq tf(\mathbf{x}) + (1-t)f(\mathbf{y}) \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d, t \in [0,1].$$
   - We say that $f$ is *L-smooth* if
   $$\left|f(\mathbf{y}) - (f(\mathbf{x}) + (\nabla f(\mathbf{x}))^\top (\mathbf{y} - \mathbf{x}))\right| \leq \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2 \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d,$$
   where $\|\cdot\|$ denotes the usual $\ell_2$ norm (i.e., $\|\mathbf{z}\|^2 = \sum_i z_i^2$).
   - We say that $f$ is $\mu$*-strongly convex* if
   $$f(\mathbf{y}) \geq f(\mathbf{x}) + (\nabla f(\mathbf{x}))^\top (\mathbf{y} - \mathbf{x}) + \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|^2 \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

   It can be shown that if $f$ is $\mu$-strongly convex for some $\mu \geq 0$, it is convex.

   (a) Show that if $f$ is $L$-smooth, then for all $\mathbf{x} \in \mathbb{R}^d$,
   $$f\Big(\mathbf{x} - \frac{1}{L}\nabla f(\mathbf{x})\Big) - f(\mathbf{x}) \leq -\frac{1}{2L}\|\nabla f(\mathbf{x})\|^2$$
   and
   $$f(\mathbf{x}^*) - f(\mathbf{x}) \leq -\frac{1}{2L}\|\nabla f(\mathbf{x})\|^2$$
   where $\mathbf{x}^*$ is a global minimizer of $f$.

   (b) Recall that in gradient descent, we implement
   $$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla f(\mathbf{x}_k).$$
   Show that if $f$ is $L$-smooth and $\mu$-strongly convex, the learning rate $\eta \in (0, 1/L]$, and we start the iterative procedure at $\mathbf{x}_0$, then
   $$\|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2 \leq (1 - \eta\mu)^{k+1}\|\mathbf{x}_0 - \mathbf{x}^*\|^2$$
   In particular, if we choose the learning rate $\eta = 1/L$, the iterates converge geometrically fast at a rate $\mu/L$.

**Remark 3.** *This result says that if we want to get* $\varepsilon$*-close to the optimal solution* $\mathbf{x}^*$ *(i.e.,* $\|\mathbf{x}_k - \mathbf{x}_0\| < \varepsilon$*), we need to use roughly*
$$\frac{2\log(\frac{\|\mathbf{x}_0 - \mathbf{x}^*\|}{\varepsilon})}{-\log(1 - \frac{\mu}{L})}$$
*iterations. The numerator can be interpreted as the log of the ratio of the initial suboptimality (i.e., gap between* $\mathbf{x}_0$ *and* $\mathbf{x}^*$*), to the final suboptimality (i.e., less than* $\varepsilon$*). The denominator is* $-\log(1 - \mu/L) \approx \mu/L$*. When* $\mu/L$ *is small, it takes many iterations to get to an* $\varepsilon$*-close solution. Thus, we favor situations in which* $\mu/L$ *is large. In fact,* $L/\mu$ *is known as the condition number; the smaller it is, the better! If you are interested in this, a good course to take is EE5138.*

**Remark 4.** *In fact, among the class of all L-smooth,* $\mu$*-strongly convex functions, the "contraction factor"* $(1 - \mu/L)$ *is the best possible. See the following:*

*Drori, Y., Teboulle, M.: Performance of first-order methods for smooth convex minimization: A novel approach. Math. Program. 145(1), 451–482 (2014)*