Homework 2 – Deep Learning (CS/DS 541, Whitehill, Spring 2020)

You may complete this homework assignment either individually or in teams up to 2 people.

1. **XOR problem** [10 points, on paper]: Show (by deriving the gradient, setting to 0, and solving mathematically, not in Python) that the values for $\mathbf{w} = (w_1, w_2)$ and b that minimize the function $J(\mathbf{w}, b)$ in Equation 6.1 (in the *Deep Learning* textbook) are: $w_1 = 0$, $w_2 = 0$, and $b = 0.5$.

We have,

$$J(\theta) = \frac{1}{4} \sum_{x \in X} \left(f^*(x) - f(x, \theta)\right)^2$$

$$\theta = \{w, b\}$$

$$\therefore J(w, b) = \frac{1}{4} \sum_{x \in X} \left(f^*(x) - f(x, w, b)\right)^2$$

$$\frac{\partial J}{\partial b} = \frac{-1}{2} \sum_{x \in X} \left[f^*(x) - (x^T w + b)\right] = 0$$

$$\therefore \sum_{x \in X} f^*(x) = \sum_{x \in X} (x^T w + b) = \sum_{x \in X} x^T w + 4b$$

$$\therefore b = \frac{1}{4} \left(\sum_{x \in X} f^*(x) - \sum_{x \in X} x^T w \right)$$

for XOR gate we have,

$$X = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \qquad w = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \qquad f^*(x) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\therefore x^T w = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ w_2 \\ w_1 \\ w_1 + w_2 \end{bmatrix}$$

$$\therefore \sum x^T w = 2w_1 + 2w_2$$

$$\sum f^*(x) = 2$$

$$\therefore \quad b = \frac{1}{4}(2 - 2w_1 - 2w_2)$$

$$b = \frac{1 - w_1 - w_2}{2} \qquad \text{——— (1)}$$

$$\therefore \frac{\partial J}{\partial w} = \frac{-1}{2} \sum_{x \in X} x \left[f^*(x) - (x^T w + b) \right] = 0$$

$$\sum_{x \in X} x \left[f^*(x) - (x^T w + b) \right] = 0$$

$$\sum_{x \in X} x f^*(x) = \sum_{x \in X} x (x^T w + b)$$

$$\therefore \sum_{x \in X} x f^*(x) = \sum_{x \in X} x x^T w + \sum_{x \in X} x b$$

$$\sum_{x \in X} x f^*(x) = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\sum_{x \in X} x x^T w = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 2w_1 + w_2 \\ w_1 + 2w_2 \end{bmatrix}$$

$$x_b = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \frac{(1 - w_1 - w_2)}{2}$$

$$\therefore \sum x_b = \begin{bmatrix} 1-w_1-w_2 \\ 1-w_1-w_2 \end{bmatrix}$$

$$\therefore \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2w_1+w_2 \\ w_1+2w_2 \end{bmatrix} + \begin{bmatrix} 1-w_1-w_2 \\ 1-w_1-w_2 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1+w_1 \\ 1+w_2 \end{bmatrix}$$

$\therefore w_1 = 0 \quad w_2 = 0$

Substituting w_1 and w_2 in (1),

$$b = \frac{1 - 0 - 0}{2}$$

$\therefore b = 0.5$

Alternatively, $y = w^T x + b$ can be reduced to $y = w^T x$ with,

$$x = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad w = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Using normal equation,

$$w = (x^T x)^{-1} x y$$

$$x^T x = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 4 \end{bmatrix}$$

$$(X^TX)^{-1} = \begin{bmatrix} 1 & 0 & -1/2 \\ 0 & 1 & -1/2 \\ -1/2 & -1/2 & 3/4 \end{bmatrix}$$

$$Xy = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

$$\therefore \quad w = (X^TX)\,Xy = \begin{bmatrix} 1 & 0 & -1/2 \\ 0 & 1 & -1/2 \\ -1/2 & -1/2 & 3/4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1/2 \end{bmatrix}$$

$\therefore w_1 = 0 \quad w_2 = 0 \quad b = 0.5$

3. **Regularization to encourage symmetry** [10 points, on paper]: Faces (and some other kinds of data) tend to be left-right symmetric. How can you use L_2 regularization to discourage the weights from becoming too *a*symmetric? For simplicity, consider the case of a tiny 1×2 "image". Hint: instead of using $\frac{\alpha}{2}\mathbf{w}^\top\mathbf{w} = \frac{\alpha}{2}\mathbf{w}^\top\mathbf{I}\mathbf{w}$ as the L_2 penalty term (where α is the regularization strength), consider a different matrix in the middle. Your answer should consist of a 2×2 matrix $\mathbf{S}$ as well as an explanation of why it works.

regularization cost $= \frac{\alpha}{2} w^T S w$ $\quad X = [x \;\; x]$

let $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ $\quad S = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}$

$\therefore \quad cost_r = \frac{\alpha}{2} [w_1 \; w_2] \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

$= \frac{\alpha}{2} \left([w_1 s_{11} + w_2 s_{21} \quad w_1 s_{12} + w_2 s_{22}] \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \right)$

$= \frac{\alpha}{2} \left(s_{11} w_1^2 + w_1 w_2 (s_{12} + s_{21}) + s_{22} w_2^2 \right)$

we want to maximize the cost for difference in the w_1 and w_2.

$\therefore cost_r = \frac{\alpha}{2}(w_1 - w_2)^2 = \frac{\alpha}{2} \left(s_{11} w_1^2 + w_1 w_2 (s_{12} + s_{21}) + s_{22} w_2^2 \right)$

$\therefore \; w_1^2 - 2 w_1 w_2 + w_2^2 = s_{11} w_1^2 + w_1 w_2 (s_{12} + s_{21}) + s_{22} w_2^2$

$\therefore$ comparing, $s_{11} = 1 \quad s_{12} = -1 \quad s_{21} = -1 \quad s_{22} = 1$

$\therefore S = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$

4. **Recursive state estimation in Hidden Markov Models** [10 points, on paper]: Teachers try to monitor their student's knowledge of the subject-matter, but teachers cannot directly peer inside students' brains. Hence, they must make *inferences* about what the student knows based on students' *observable behavior*, i.e., how they perform on tests, their facial expressions during class, etc. Let random variable (RV) X_t represent the student's *state*, and let RV Y_t represent the student's observable behavior, at time t. We can model the student as a Hidden Markov Model (HMM):

   (a) X_t depends *only* on the previous state X_{t-1}, *not* on any states prior to that (*Markov* property), i.e.

   $$P(x_t \mid x_1, \ldots, x_{t-1}) = P(x_t \mid x_{t-1})$$

   (b) The student's behavior Y_t depends only on his/her current state X_t, i.e.:

   $$P(y_t \mid x_t, y_1, \ldots, y_{t-1}) = P(y_t \mid x_t)$$

   (c) X_t cannot be observed directly (it is *hidden*).

Note that, in general, σ^2 can also be a function of $\mathbf{x}$ (heteroscedastic case). Moreover, *non*-linear Gaussian models are also completely possible, e.g., the mean (and possibly the variance) of the Gaussian distribution is output by a deep neural network. However, in this problem, we will assume that μ is linear in $\mathbf{x}$, and that σ^2 is the same for all $\mathbf{x}$ (homoscedastic case).

MLE: The parameters of probabilistic models are commonly optimized by *maximum likelihood estimation* (MLE). (Another common approach is *maximum a posteriori* estimation, which allows the practitioner to incorporate a "prior belief" about the parameters' values.) Suppose the training dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$. Let the parameters/weights of the linear-Gaussian model be $\mathbf{w}$, such that the mean $\mu = \mathbf{x}^\top \mathbf{w}$. Prove that the MLE of $\mathbf{w}$ and σ^2 given $\mathcal{D}$ is:

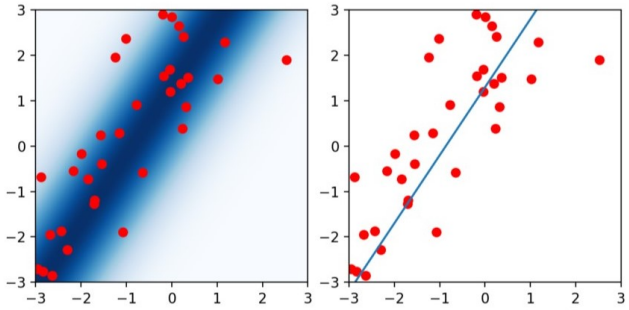
$$\mathbf{w} = \left(\sum_{i=1}^{n} \mathbf{x}^{(i)} {\mathbf{x}^{(i)}}^\top \right)^{-1} \left(\sum_{i=1}^{n} \mathbf{x}^{(i)} y^{(i)} \right)$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} ({\mathbf{x}^{(i)}}^\top \mathbf{w} - y^{(i)})^2$$

Note that this solution – derived based on *maximizing* probability – is exactly the same as the optimal weights of a 2-layer neural network optimized to *minimize* MSE.

Hint: Follow the same strategy as the MLE derivation for a biased coin in `Class2.pdf`. For a linear-Gaussian model, the argmax of the likelihood equals the argmax of the log-likelihood. The log of the Gaussian likelihood simplifies beautifully.

5. **Linear-Gaussian prediction model** [15 points, on paper]:



Probabilistic prediction models enable us to estimate not just the "most likely" or "expected" value of the target y (see figure above, right), but rather an entire *probability distribution* about which target values are more likely than others, given input $\mathbf{x}$ (see figure above, left). In particular, a linear-Gaussian model is a Gaussian distribution whose expected value (mean μ) is a linear function of the input features $\mathbf{x}$, and whose variance is σ^2:

$$P(y \mid \mathbf{x}) = \mathcal{N}(\mu = \mathbf{x}^\top \mathbf{w}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \mathbf{x}^\top \mathbf{w})^2}{2\sigma^2}\right)$$

Note that, in general, σ^2 can also be a function of $\mathbf{x}$ (heteroscedastic case). Moreover, *non*-linear Gaussian models are also completely possible, e.g., the mean (and possibly the variance) of the Gaussian distribution is output by a deep neural network. However, in this problem, we will assume that μ is linear in $\mathbf{x}$, and that σ^2 is the same for all $\mathbf{x}$ (homoscedastic case).

MLE: The parameters of probabilistic models are commonly optimized by *maximum likelihood estimation* (MLE). (Another common approach is *maximum a posteriori* estimation, which allows the practitioner to incorporate a "prior belief" about the parameters' values.) Suppose the training dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$. Let the parameters/weights of the linear-Gaussian model be $\mathbf{w}$, such that the mean $\mu = \mathbf{x}^\top \mathbf{w}$. Prove that the MLE of $\mathbf{w}$ and σ^2 given $\mathcal{D}$ is:

$$\mathbf{w} = \left(\sum_{i=1}^n \mathbf{x}^{(i)} {\mathbf{x}^{(i)}}^\top\right)^{-1} \left(\sum_{i=1}^n \mathbf{x}^{(i)} y^{(i)}\right)$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n ({\mathbf{x}^{(i)}}^\top \mathbf{w} - y^{(i)})^2$$

Note that this solution – derived based on *maximizing* probability – is exactly the same as the optimal weights of a 2-layer neural network optimized to *minimize* MSE.

Hint: Follow the same strategy as the MLE derivation for a biased coin in `Class2.pdf`. For a linear-Gaussian model, the argmax of the likelihood equals the argmax of the log-likelihood. The log of the Gaussian likelihood simplifies beautifully.