作业 2

1. 正则化的贝叶斯解释

背景: 在贝叶斯统计中，几乎每个量都是随机变量，可以是已观测的或未观测的。例如，参数 θ 通常是未观测的随机变量，而数据 x 和 y 则是观测到的随机变量。所有随机变量的联合分布也被称为模型（例如，$p(x, y, \theta)$）。每个未知量都可以通过将模型对所有观测量进行条件化来估计。这样的对未观测随机变量进行条件化的分布称为*后验分布*。例如在机器学习上下文中，$p(\theta \mid x, y)$ 就是后验分布。这种方法的一个结果是我们需要为模型参数赋予先验分布，即 $p(\theta)$。先验概率是在看到数据之前赋予的——它们捕捉了我们在观测到任何证据之前对模型参数的先验信念。

在最纯粹的贝叶斯解释中，我们需要一直保留参数的后验分布，直到预测为止，得到*后验预测分布*，最终的预测将是后验预测分布的期望值。然而在大多数情况下，这在计算上非常昂贵，我们会妥协，选择一种在贝叶斯意义上不那么纯粹的方法。

这种妥协是估计参数的点值（而不是完整的分布），即后验分布的众数。估计后验分布的众数也称为*最大后验估计*（MAP）。即：

$$\theta_{\mathrm{MAP}} = \arg\max_{\theta} p(\theta \mid x, y)\Theta$$

将其与之前我们见过的*最大似然估计*（MLE）进行比较：

$$\theta_{\mathrm{MLE}} = \arg\max_{\theta} p(y \mid x, \theta)\Theta$$

在这个问题中，我们探讨 MAP 估计与常见的正则化技术（这些技术应用于 MLE 估计）之间的联系。特别地，你将证明如何选择 θ 的先验分布（例如，高斯先验或拉普拉斯先验）与不同种类的正则化（例如，L_2 或 L_1 正则化）是等价的。你还将在第（d）部分探讨正则化强度如何影响泛化。

(a)

如果我们假设 $p(\theta) = p(\theta \mid x)$，请证明 $\theta_{\mathrm{MAP}} = \arg\max_{\theta} p(y \mid x, \theta)p(\theta)$。对于线性回归等不显式建模输入 x 的模型，假设 $p(\theta) = p(\theta \mid x)$ 是合理的。（请注意，这意味着 x 与 θ 在边际上独立，但在给定 y 的情况下并不条件独立。）

(b)

回顾一下，L_2 正则化在最小化损失时对参数的 L_2 范数进行惩罚（即，在概率模型中为负对数似然）。现在我们将证明，带零均值高斯先验的 MAP 估计，特别是 $\theta \sim \mathcal{N}(0, \eta^2 I)$，等价于

MLE 估计中的 L_2 正则化。具体地，证明对于某个标量 λ，

$$\theta_{\text{MAP}} = \arg\min_{\theta} \; -\log p(y \mid x, \theta) + \lambda \|\theta\|_2^2 \Theta$$

另外，λ 的值是什么？

(c)

现在考虑一个具体的例子，一个线性回归模型由 $y = \theta^T x + \epsilon$ 给出，其中 $\epsilon \sim \mathcal{N}(0, \sigma^2)$。假设随机噪声 $\epsilon^{(i)}$ 对于每个训练样本 $x^{(i)}$ 是独立的。和之前一样，假设这个模型有一个高斯先验分布，即 $\theta \sim \mathcal{N}(0, \eta^2 I)$。为了方便表示，设 X 是所有训练样本输入的设计矩阵，每一行向量是一个样本输入，$\vec{y}$ 是所有样本输出的列向量。请给出 θ_{MAP} 的闭式表达式。

(d)

接下来，考虑拉普拉斯分布，其密度函数为

$$f_L(z \mid \mu, b) = \frac{1}{2b} \exp\left(-\frac{|z - \mu|}{b}\right) \Theta$$

和之前一样，考虑线性回归模型 $y = x^T \theta + \epsilon$，其中 $\epsilon \sim \mathcal{N}(0, \sigma^2)$。假设该模型的每个参数 θ_i 都是边际独立的，且 $\theta_i \sim \mathcal{L}(0, b)$。证明此时的 θ_{MAP} 等价于带 L_1 正则化的线性回归的解，其损失函数定义为

$$J(\theta) = \|X\theta - \vec{y}\|_2^2 + \gamma \|\theta\|_1 \Theta$$

另外，γ 的值是什么？

注意： 带 L_1 正则化的线性回归问题并不存在闭式解。为了优化它，我们使用随机初始化的梯度下降法，进行数值求解。

备注： 带 L_2 正则化的线性回归也被称为*岭回归*（Ridge regression），而带 L_1 正则化的情况则通常被称为*套索回归*（Lasso regression）。这些正则化方法同样可以应用于任意广义线性模型，正如上面所示（通过将 $\log p(y \mid x, \theta)$ 替换为相应的族似然）。上述类型的正则化技术也称为*权重衰减*和*收缩*。高斯和拉普拉斯先验鼓励参数值更接近于它们的均值（即零），这导致了收缩效应。

备注： 套索回归（即 L_1 正则化）通常会导致稀疏参数，其中大多数参数值为零，只有一部分参数值非零。

2. 逻辑回归：训练稳定性

在这个问题中，我们将深入研究逻辑回归的工作机制。考虑一个二分类问题，其中数据点为 $(x^{(i)}, y^{(i)})$，$x^{(i)} \in \mathbb{R}^d$，$y^{(i)} \in \{0, 1\}$。我们使用逻辑回归模型正类的概率：

$$P(y = 1 | x^{(i)}) = \frac{1}{1 + e^{-\theta^T x^{(i)}}} \text{fi}$$

其中 $\theta \in \mathbb{R}^d$ 是参数向量。

逻辑回归的**损失函数**（也称为交叉熵损失）对于 n 个数据点的定义为：

$$L(\theta) = -\sum_{i=1}^{n} \left(y^{(i)} \log(P(y = 1 | x^{(i)})) + (1 - y^{(i)}) \log(1 - P(y = 1 | x^{(i)}))\right) \Theta$$

假设 John 实现了逻辑回归，并在两个标注数据集上进行训练：

- **数据集 A:** 一个不可分的数据集，其中不存在一个超平面能够完全分开这两个类。换句话说，对于任何参数向量 θ 的选择，总有一些来自每个类的点位于由 $\theta^T x = 0$ 定义的超平面的两侧。
- **数据集 B:** 一个可分的数据集，其中存在一个超平面可以完全分开这两个类，也就是说，存在一个参数向量 θ，使得所有正例 $(y^{(i)} = 1)$ 位于超平面的同一侧，而所有负例 $(y^{(i)} = 0)$ 位于另一侧。

训练后，John 观察到他的模型在数据集 A 上收敛，但在数据集 B 上未能收敛。他想了解原因。你的任务是帮助他分析这一现象。

(a)

假设我们用$c \cdot \theta$ 来替换 θ，其中 $c > 0$ 是一个常数。证明当 $c \to \infty$ 时，对于正确分类的样本，正例的预测概率趋于 1，负例的预测概率趋于 0。

(b)

利用第 (a) 部分的结果，证明在线性可分的数据中，损失函数 $L(\theta)$ 随着 $c \to \infty$ 而无限下降。解释为什么这意味着梯度下降在这种情况下会失败而无法收敛。

(c)

将此与不可分数据的情况进行对比。解释为什么在不可分的情况下损失函数有一个有限的下界，并且梯度下降能够收敛。

(d)

对于以下每个修改，说明它是否会导致训练算法在像 B 这样的数据集上收敛。

- 使用不同的常数学习率。
- 随时间减少学习率（例如，随着梯度下降迭代次数 t 增加，初始学习率按 $1/t^2$ 缩放）。
- 对输入特征应用线性缩放。
- 在损失函数中添加正则项 $\|\theta\|^2$。
- 在训练数据或标签上添加零均值高斯噪声。

在接下来的问题中，我们考虑具有多层的神经网络。每一层有多个输入和输出，并可以分为两个部分：

- 一个**线性**模块，它实现线性变换：$z_j = (\sum_{i=1}^{m} x_i w_{i,j}) + w_{0,j}$，由权重矩阵 W 和偏置向量 W_0 指定。输出为 $[z_1, \ldots, z_n]^T$。
- 一个**激活**模块，它对线性模块的输出应用激活函数，例如在隐藏层中使用 Tanh 或 ReLU 激活函数，或在输出层使用 Softmax（见下文）。我们将输出表示为 $[f(z_1), \ldots, f(z_m)]^T$，尽管技术上来说，对于某些激活函数（如 softmax），每个输出将依赖于所有 z_i，而不仅仅是一个。

我们将使用以下符号来表示网络中的量：

- 网络的输入为 $x_1, \ldots, x_d$。
- 层数为 L。
- 第 l 层有 m^l 个输入。
- 第 l 层有 $n^l = m^l + 1$ 个输出。
- 第 l 层的权重矩阵为 W^l，是一个 $m^l \times n^l$ 的矩阵，偏置向量（偏移）为 W_0^l，是一个 $n^l \times 1$ 的向量。
- 第 l 层线性模块的输出称为**预激活**值，记作 z^l。
- 第 l 层的激活函数为 $f^l(\cdot)$。
- 第 l 层的激活值为 $a^l = \left[f^l(z_1^l), \ldots, f^l(z_{n_l}^l)\right]^T$。
- 网络的输出是 $a^L = \left[f^L(z_1^L), \ldots, f^L(z_{n_L}^L)\right]^T$。
- 损失函数 $Loss(a, y)$ 用于度量输出值 a 在目标为 y 时的损失。

3. 多分类问题

如果我们需要将作业题目分类为三类：有趣的、无聊的和不可能完成的，我们可以在输出上使用*独热编码*，并使用三个输出单元以及称为*softmax*（SM）的激活模块。它不是一个典型的激活模块，因为它接受所有 n_L 个预激活值 $z_j^L \in \mathbb{R}$，并返回 n_L 个输出值 $a_j^L \in [0,1]$，使得 $\sum_j a_j^L = 1$。这可以被解释为对可能类别的概率分布。

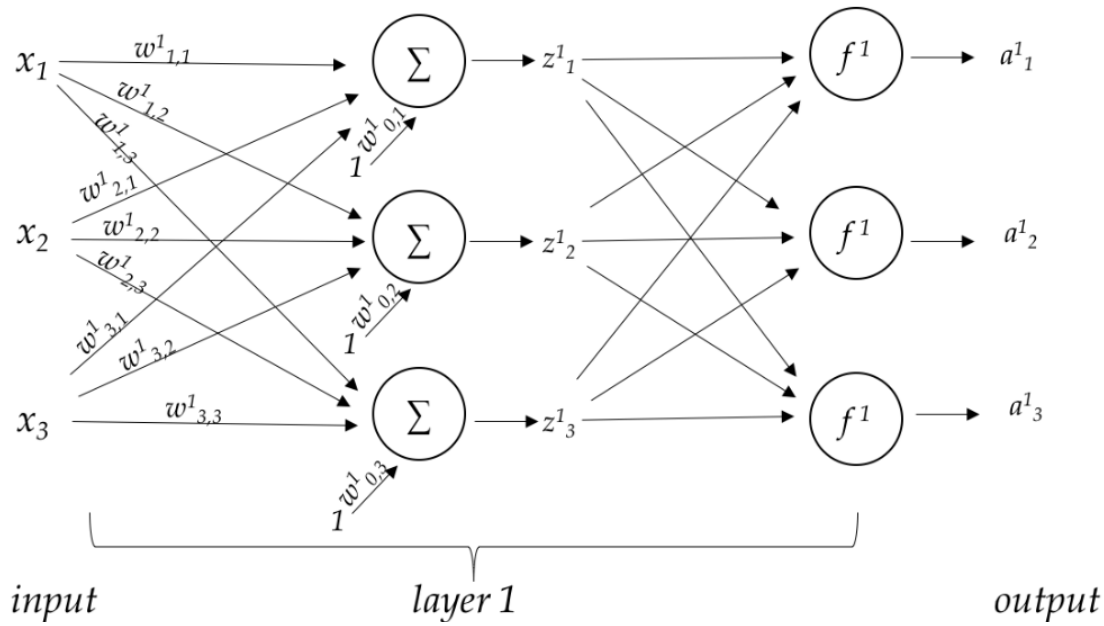
个别条目计算如下：

$$a_j = \frac{e^{z_j}}{\sum_{k=1}^{n_L} e^{z_k}}$$

我们将向量 $\mathbf{a}$ 与向量 $\mathbf{z}$ 的关系描述为

$$\mathbf{a} = \mathrm{SM}(\mathbf{z})$$

下图显示了一个由线性模块和 softmax 激活模块组成的单层网络。



(a)

由 $z_L = [-1, 0, 1]^T$ 表示的类别概率分布是什么？

(b)

现在，我们需要一个损失函数 $Loss(\mathbf{a}, \mathbf{y})$，其中 $\mathbf{a}$ 是离散概率分布，$\mathbf{y}$ 是一个单输出值的独热向量编码。由于与之前相同的原因，使用负对数似然作为损失函数是有意义的。因此，我们将之前的 NLL 定义扩展为：

$$NLL(\mathbf{a}, \mathbf{y}) = -\sum_{j=1}^{n_L} y_j \ln a_j^L \Theta$$

注意，上述表达式适用于多类（类别数 > 2）的情况。

如果 $\mathbf{a} = [0.3, 0.5, 0.2]^T$ 且 $\mathbf{y} = [0, 0, 1]^T$，那么 $NLL(\mathbf{a}, \mathbf{y})$ 是多少？

(c)

现在，我们可以考虑一个带有 softmax 激活模块的单层，训练目标是最小化 NLL。预激活值（线性模块的输出）为：

$$z_j^L = \sum_k w_{k,j}^L x_k + w_{0,j}^L$$

且 $a^L = SM(z^L)$。

为了进行梯度下降，我们需要知道 $\frac{\partial}{\partial w_{k,j}^L} NLL(a^L, \mathbf{y})$。尝试证明

$$\frac{\partial}{\partial w_{k,j}^L} NLL(a^L, \mathbf{y}) = x_k(a_j^L - y_j)$$

当然，可以通过一次快速矩阵计算来计算这些导数的整个矩阵，即 $\nabla_{W^L} NLL(a^L, \mathbf{y})$。

假设我们有两个输入单元和三个可能的输出值，权重矩阵 W^L 为

$$W^L = \begin{bmatrix} 1 & -1 & -2 \\ -1 & 2 & 1 \end{bmatrix}$$

假设偏置为零，输入 $\mathbf{x} = [1,1]^T$，目标输出 $\mathbf{y} = [0,1,0]^T$。那么矩阵 $\nabla_{W^L} NLL(a^L, \mathbf{y})$ 是多少？

(d)

在进行任何梯度更新之前，x 属于类别 1 的预测概率是多少？（假设我们有类别 0、1 和 2。）

(e)

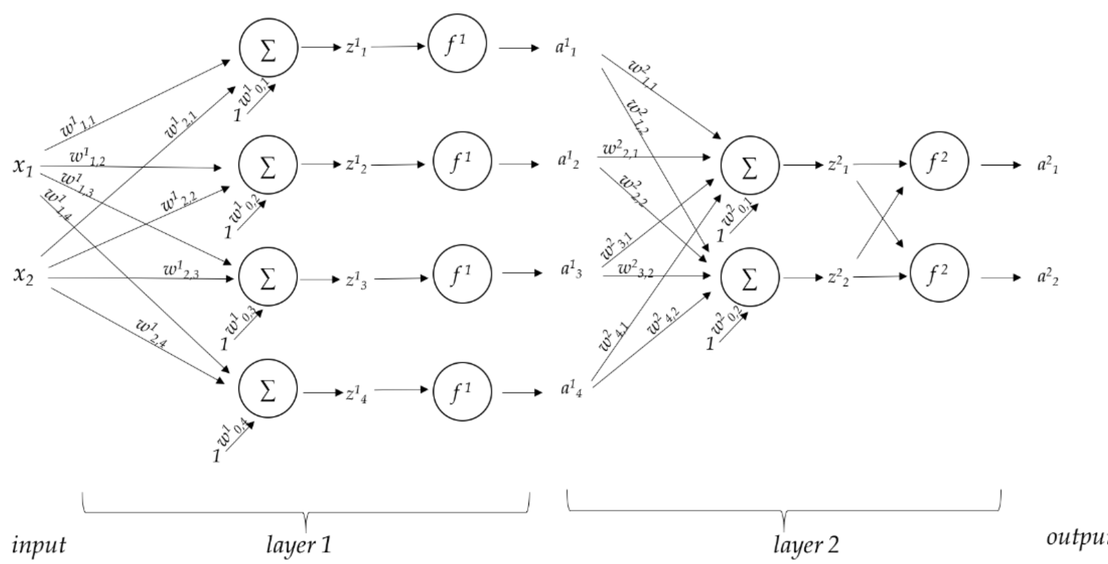
使用步长 0.5，经过一次梯度更新后，W^L 是多少？

(f)

给定新的权重矩阵，x 属于类别 1 的预测概率是多少？

4. 神经网络

考虑下图所示的神经网络，其中所有隐藏神经元都使用 ReLU 激活函数（图中的 f^1），输出层使用 softmax 激活函数（图中的 f^2），输出为 softmax 输出（图中的 a_1^2 和 a_2^2）。



给定输入 $\mathbf{x} = [x_1, x_2]^T$，网络的隐藏单元按以下方程分阶段激活：

$$z_1^1 = x_1 w_{1,1}^1 + x_2 w_{2,1}^1 + w_{0,1}^1 \qquad a_1^1 = \max\{z_1^1, 0\}$$
$$z_2^1 = x_1 w_{1,2}^1 + x_2 w_{2,2}^1 + w_{0,2}^1 \qquad a_2^1 = \max\{z_2^1, 0\}$$
$$z_3^1 = x_1 w_{1,3}^1 + x_2 w_{2,3}^1 + w_{0,3}^1 \qquad a_3^1 = \max\{z_3^1, 0\}$$
$$z_4^1 = x_1 w_{1,4}^1 + x_2 w_{2,4}^1 + w_{0,4}^1 \qquad a_4^1 = \max\{z_4^1, 0\}$$

$$z_1^2 = a_1^1 w_{1,1}^2 + a_2^1 w_{2,1}^2 + a_3^1 w_{3,1}^2 + a_4^1 w_{4,1}^2 + w_{0,1}^2$$
$$z_2^2 = a_1^1 w_{1,2}^2 + a_2^1 w_{2,2}^2 + a_3^1 w_{3,2}^2 + a_4^1 w_{4,2}^2 + w_{0,2}^2$$

网络的最终输出通过对最后一层应用 softmax 函数得到：

$$a_1^2 = \frac{e^{z_1^2}}{e^{z_1^2} + e^{z_2^2}}$$
$$a_2^2 = \frac{e^{z_2^2}}{e^{z_1^2} + e^{z_2^2}}$$

在这个问题中，我们将考虑以下参数设置：

$$\begin{bmatrix} w_{1,1}^1 & w_{1,2}^1 & w_{1,3}^1 & w_{1,4}^1 \\ w_{2,1}^1 & w_{2,2}^1 & w_{2,3}^1 & w_{2,4}^1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}, \qquad \begin{bmatrix} w_{0,1}^1 \\ w_{0,2}^1 \\ w_{0,3}^1 \\ w_{0,4}^1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix},$$

$$\begin{bmatrix} w_{1,1}^2 & w_{1,2}^2 \\ w_{2,1}^2 & w_{2,2}^2 \\ w_{3,1}^2 & w_{3,2}^2 \\ w_{4,1}^2 & w_{4,2}^2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}, \qquad \begin{bmatrix} w_{0,1}^2 \\ w_{0,2}^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

(a)

考虑输入 $x_1 = 3, x_2 = 14$。网络隐藏单元的输出 $(f^1(z_1^1), f^1(z_2^1), f^1(z_3^1), f^1(z_4^1))$ 和最终输出 (a_1^2, a_2^2) 是什么？

(b)

考虑以下输入向量：$x^{(1)} = [0.5, 0.5]^T$，$x^{(2)} = [0, 2]^T$，$x^{(3)} = [-3, 0.5]^T$。输入一个矩阵，其中每一列表示每个输入向量的隐藏单元的输出 $(f(z_1^1), \ldots, f(z_4^1))$。

在上面的网络中，输出层有两个 softmax 单元，用于分类为两个类之一。对于第一个类，第一个单元的输出应大于另一个单元的输出；对于第二个类，第二个单元的输出应大于第一个单元的输出。这可以通过使用 k 个输出单元自然地扩展到 k 个类。

下面描述网络输出表示第一个类（即 a_1^2 较大）或第二个类（即 a_2^2 较大）在 x 空间中的区域。

神经网络在以下情况下的输出值是什么？

情况 1) $f(z_1^1) + f(z_2^1) + f(z_3^1) + f(z_4^1) = 0$

情况 2) $f(z_1^1) + f(z_2^1) + f(z_3^1) + f(z_4^1) = 1$

情况 3) $f(z_1^1) + f(z_2^1) + f(z_3^1) + f(z_4^1) = 3$

(c)

使用交叉熵损失函数（Cross-Entropy Loss），给定输入样本 $(x_1 = 3, x_2 = 14)$ 和目标向量 $(y_1, y_2) = (0, 1)$，执行一次反向传播步骤，以学习率 $\eta = 0.1$ 更新网络中的每一个权重。