作业 1：线性模型和支持向量机

清华大学软件学院
机器学习，2021 年秋季学期

1 介绍

本次作业需要提交说明文档（PDF 形式）和 Python 的源代码。注意事项如下：

- 本次作业线性模型和支持向量机各占 50 分，但实际题量大于 50 分，若得分超过 50 分，则按照 50 分截断。
- 作业按点给分，因此请在说明文档中按点回答，方便助教批改。示例如下：
  2.2.2 $\nabla J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\nabla\left(h_\theta(x_i)-y_i\right)^2=\cdots$。
- 除非特别说明，本次作业不能直接使用机器学习的开源库，例如 sklearn、pytorch 等。
- 你需要熟练掌握 numpy 及其广播（broadcast）机制，使用 for 循环实现的矩阵运算不会得到编程部分的分数。
- 请提供易懂的注释，若代码的可读性过差，会酌情扣除对应题的分数。
- 在 PDF 中记录你在写程序的哪些模块时与他人有过交流，与他人交流需具体到姓名（网络论坛上的交流可填写用户名），参考网络资料需具体到链接。
- 不要使用他人的作业（含代码和文档），也不要向他人公开自己的作业，否则处罚很严厉，会扣至-100（倒扣本次作业的全部分值）。

2 线性模型 (53pt)

在本题中，你将使用梯度下降法（Gradient Descent）实现岭回归（Ridge Regression）算法。

2.1 特征归一化

在实际应用中，当数据的不同维特征差异很大时，梯度下降的收敛速度会变得很慢。此外，使用正则化时，具有较大绝对值的特征对正则化有更大的影响。因此，我们需要进行特征归一化。一种常见方法是执行仿射变换，将**训练集**中的所有特征值映射至 $[0,1]$，对验证集上的每个特征也需

要使用相同的仿射变换。你需要修改函数 `feature_normalization` 实现特征的归一化（2pt）。

2.2 梯度下降

在线性回归中，我们考虑线性函数的假设空间 $h_\theta : \mathbf{R}^d \to \mathbf{R}$,

$$h_{\theta,b}(x) = \theta^T x + b,$$

其中 $\theta, b, x \in \mathbf{R}^d$, 为了书写和编程的方便，我们通常为 x 添加一个额外的维度，该维度始终是一个固定值，例如 1，用于消除 h_θ 的偏移量参数 b。此时，我们可以将待优化的函数写成：

$$h_\theta(x) = \theta^T x,$$

其中 $\theta, x \in \mathbf{R}^{d+1}$。我们需要找到合适的 θ 最小化:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i\right)^2,$$

其中 $(x_1, y_1), \ldots, (x_m, y_m) \in \mathbf{R}^{d+1} \times \mathbf{R}$ 是训练数据.

1. 将训练数据的特征记成 $X \in \mathbf{R}^{m\times(d+1)}$，$X$ 的第 i 行是 x_i，训练数据的输出记作 $y = (y_1, \ldots, y_m)^T \in \mathbf{R}^{m\times 1}$，请写出 $J(\theta)$ 的矩阵形式。(3pt)
2. 请写出 $J(\theta)$ 对 θ 梯度的矩阵形式。(3pt)
3. 在最小化 $J(\theta)$ 时，假设取 θ 到 $\theta + \eta h$ 的一步，其中 $h \in \mathbf{R}^{d+1}$ 是前进方向（不一定是单位向量），$\eta \in (0, \infty)$ 是步长。请用梯度写出目标函数值变化的近似表达式 $J(\theta + \eta h) - J(\theta)$。(3pt)
4. 令 η 为步长，写出梯度下降中，更新 θ 的表达式。(3pt)
5. 修改函数 `compute_square_loss`，给定 θ，计算的 $J(\theta)$。(3pt)
6. 修改函数 `compute_square_loss_gradient`，给定 θ，计算 $J(\theta)$ 的梯度。(3pt)
7. 为了检查梯度的实现是否正确，可以用数值法来计算梯度。可导函数的梯度可以通过下述公式计算，
$$\lim_{\varepsilon\to 0} \frac{J(\theta + \varepsilon h) - J(\theta - \varepsilon h)}{2\epsilon}$$
实际中，我们可以选择一个较小的 $\varepsilon > 0$，通过逼近方向导数来逼近梯度。具体的，在每个坐标方向上，例如，首先取 $h = (1, 0, 0, \ldots, 0)$ 计算第一维的梯度，然后取 $h = (0, 1, 0, \ldots, 0)$ 得到第二维的梯度，以此类推。将每一维的梯度合起来，就得到 $J(\theta)$ 在 θ 处的梯度。根据提示，你可以实现函数 `grad_checker`。(3pt)
8. main 函数已经加载数据，将其拆分成一个训练集和验证集，并归一化。你现在需要修改 `batch_gradient_descent`，使得模型可以在训练集上进行训练。(3pt)
9. 选择合适的步长。从 0.1 的步长开始，尝试各种不同的固定步长（至少包括 0.5、0.1、0.05 和 0.01），记录哪个步长收敛最快或哪个发散。并绘制在不同步长下，目标函数随着训练时间变化的曲线。(3pt)

2.3 岭回归

岭回归是使用 ℓ_2 正则化的线性回归，其目标函数是

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i)-y_i\right)^2+\lambda\theta^T\theta,$$

其中 λ 是正则化参数，它控制正则化程度。

1. 计算 $J(\theta)$ 的梯度，以及梯度下降算法中的 θ 的更新公式（矩阵形式）。(3pt)
2. 实现 `compute_regularized_square_loss_gradient`。(3pt)
3. 实现 `regularized_grad_descent`。(3pt)
4. 选择合适的 λ 与步长，例如，$\lambda\in\left\{10^{-7},10^{-5},10^{-3},10^{-1},1,10,100\right\}$。并用表格记录超参数的调整过程。(3pt)

2.4 随机梯度下降

当训练数据集非常大时，梯度下降算法需要遍历整个数据集，因此效率较低。实际中用的往往是随机梯度下降（SGD）算法。令 $f_i(\theta)$ 是第 i 个数据点上的损失，则

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}f_i(\theta)$$

在 SGD 中，每一步会随机采样 $-\nabla f_i(\theta), i\in\{1,\ldots,m\}$ 来计算 θ 的优化方向。

1. 对于目标函数
$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i)-y_i\right)^2+\lambda\theta^T\theta$$
请给出对应的 $f_i(\theta)$ 的表达式。(3pt)
2. 证明随机梯度 $\nabla f_i(\theta)$ 是 $\nabla J(\theta)$ 无偏估计，即证明 $\mathbb{E}\left[\nabla f_i(\theta)\right]=\nabla J(\theta)$[1]。(3pt)
3. 实现 `stochastic_grad_descent`。(3pt)
4. 随机梯度下降算法的噪音较大，因此模型较难收敛。修改 `stochastic_grad_descent` 使得其支持不同的批大小，并记录随着批大小逐渐增大时，目标函数训练曲线发生的变化。(3pt)

3 支持向量机 (57pt)

在本题，我们将使用支持向量机对文本数据进行情绪检测。

3.1 次梯度 Subgradients

对于 $f:\mathbf{R}^d\to\mathbf{R}$，在 x 如果对于所有 z，

$$f(z)\geq f(x)+g^T(z-x),$$

则 $g\in\mathbf{R}^d$ 为 x 处的次梯度。次梯度并不唯一，f 在 x 处的次梯度集合记为 $\partial f(x)$。通过使用次梯度的定义，可以完成下面两题[2]。

[1]提示：对于一般的 $J(\theta)=\frac{1}{m}\sum_{i=1}^{m}f_i(\theta)$，而不是具体的岭回归目标函数，证明更加简洁。

[2]关于次梯度的一个很好的参考是 Boyd 等人关于次梯度的课程笔记。

1. 假设 $f_1, \ldots, f_m : \mathbf{R}^d \to \mathbf{R}$ 是凸函数，并且

$$f(x) = \max_{i=1,\ldots,m} f_i(x)$$

   令 k 是满足 $f_k(x) = f(x)$ 的索引，并保证 $g \in \partial f_k(x)$。求证 $g \in \partial f(x)$。[3](3pt)

2. 给出合页损失（Hinge Loss）的次梯度。合页损失如下。(3pt)

$$J(w) = \max\left\{0, 1 - yw^T x\right\}$$

3.2 感知机

给定数据集 $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbf{R}^d \times \{-1, 1\}$，感知器算法希望能找到一个超平面将两个类别完全区分开，即找到 $w \in \mathbf{R}^d$ 使得

$$y_i w^T x_i > 0 \; \forall i \in \{1, \ldots, n\}$$

这意味着所有标签 $y = 1$ 的 x 都在超平面 $\left\{x \mid w^T x = 0\right\}$ 的一侧，并且所有标签 $y = -1$ 的 x 都在另一侧。若这样的超平面存在，则称数据是**线性可分**的。具体算法如下，

Algorithm 1: 感知机算法

```
输入: 训练集 (x_1, y_1), ..., (x_n, y_n) ∈ R^d × {-1, 1}
w^(0) = (0, ..., 0) ∈ R^d
k = 0 # 迭代次数
repeat
  all_correct = TRUE
  for i = 1, 2, ..., n # 遍历数据集
    if (y_i x_i^T w^(k) ≤ 0)
      w^(k+1) = w^(k) + y_i x_i
      all_correct = FALSE
    else
      w^(k+1) = w^(k)
    end if
    k = k + 1
  end for
until (all_correct == TRUE)
return w^(k)
```

感知机中还定义了一个感知损失，

$$\ell(\hat{y}, y) = \max\{0, -\hat{y}y\}$$

1. 令 $\mathcal{H}$ 是由函数组成的线性假设空间 $x \mapsto w^T x$。考虑使用随机次梯度下降 (SSGD) 最小化感知器损失。证明当选择合适的次梯度，并使用固定步长 1 的时候，这个过程等价于感知机算法。(3pt)
2. 假设感知器算法返回 w。证明 w 是输入数据的线性组合。即 $w = \sum_{i=1}^n \alpha_i x_i$ 对于某些 $\alpha_1, \ldots, \alpha_n \in \mathbf{R}$。当 $\alpha_i \neq 0$ 时，x_i 称为支持向量（support vector）。(3pt)

[3]已知 $\mathbf{R}^d$ 上的凸函数具有在所有点都非空的次梯度。

3.3 软间隔支持向量机

线性可分是一种理想的情况，现实数据集经常会有噪声，无法约束所有的数据点都能被正确分类，因此实际中的支持向量机都会引入软间隔，目标是让不满足约束的数据点尽可能少，即

$$\min_{w\in\mathbf{R}^d}\frac{\lambda}{2}\|w\|^2+\frac{1}{m}\sum_{i=1}^{m}\max\left\{0,1-y_i(w^Tx_i+b)\right\}.$$

1. 求该问题的拉格朗日（Lagrange）方程。(5pt)
2. 求该问题的对偶形式（Dual Form）。(5pt)
3. 已知非线性映射 Φ 和其对应的核函数 $k(\cdot,\cdot)$，若在上述支持向量机中使用该核函数，请写出对应的原问题和对偶问题。(5pt)
4. 在求出对偶问题后，使用 Sequential Minimal Optimization (SMO) 算法，可以实现软间隔支持向量机的实际求解。SVM 求解还有另外一种基于次梯度下降的算法，

```
输入: λ > 0. 令 w_1 = 0, t = 0
当还没有达到终止条件时
  For j = 1, ..., m (假设数据是随机打乱的)
    t = t + 1
    η_t = 1/(tλ);
    If y_j(w_t^T x_j + b) < 1
      w_{t+1} = (1 − η_t λ)w_t + η_t y_j x_j
    Else
      w_{t+1} = (1 − η_t λ)w_t
```

   证明软间隔支持向量机中 $J_i(w)$ 的次梯度由下式给出（提示：参考合页损失函数的次梯度）。(3pt)

$$g=\begin{cases}\lambda w-y_ix_i & \text{for } y_i(w^Tx_i+b)<1\\ \lambda w & \text{for } y_i(w^Tx_i+b)\geq 1.\end{cases}$$

5. 证明如果你的步长规则是 $\eta_t=1/(\lambda t)$，则用上一个问题的次梯度方向做 SGD 的算法就是伪代码中给出的算法。(3pt)

3.4 情绪检测

我们将使用软间隔支持向量机进行情绪检测，类别包括：开心（joy）和伤心（sadness）。我们在 start_code.py 中已经实现了数据集的加载与预处理、文本的分词，你也可以自己实现。你需要完成

1. 在函数 `vectorize` 中，用词袋法或者词嵌入的方法，将文本数据转成向量表示。此处可以调用 sklearn 等开源工具（提示：词袋法可以使用 sklearn 的 TfidfVectorizer。使用词嵌入时，最简单的方法是将一个句子中所有词的词嵌入求平均，作为句子的特征表示。）。(4pt)
2. 按照问题 3.3.4 中提供的伪代码，在函数 `linear_svm_subgrad_descent` 中实现 SVM 的次梯度下降算法，并在情绪检测数据集上进行训练（提示：参考 2.4 节随机梯度下降部分的代码）。(8pt)

3. 调整超参数，例如正则化参数 λ、步长、步长衰减策略等, 观察训练准确率与验证准确率随着超参数发生的变化，并绘制曲线或者表格。(3pt)
4. 在函数 `kernel_svm_subgrad_descent` 中实现基于核函数的非线性 SVM（例如基于线性核或者高斯核），调整相关的超参数，记录它在测试集上的准确率。核函数的引入能提高当前模型的准确吗？试解释原因。(6pt)
5. 计算并汇报最终的 SVM 模型在验证集上的准确率，F1-Score 以及混淆矩阵（Confusion Matrix）。(3pt)