

一、问题一

1. 问题描述

考虑线性测量 $\mathbf{b} = A\mathbf{x} + \mathbf{e}$ ，其中 \mathbf{b} 为50维的测量值， A 为 50×100 维的测量矩阵， \mathbf{x} 为100维的未知稀疏向量且稀疏度为5， \mathbf{e} 为50维的测量噪声。从 \mathbf{b} 与 A 中恢复 \mathbf{x} 的一范数规范化，最小二乘模型如下：

$$\min \left(\frac{1}{2} \|A\mathbf{x} - \mathbf{b}\|_2^2 + p \|\mathbf{x}\|_1 \right)$$

其中 p 为非负的正则化参数。

请设计下述算法求解该问题：

1. 邻近点梯度下降法
2. 交替方向乘子法
3. 次梯度法

在实验中，设 \mathbf{x} 的真值中的非零元素服从均值为0方差为1的高斯分布， A 中的元素服从均值为0方差为1的高斯分布， \mathbf{e} 中的元素服从均值为0方差为0.1的高斯分布。对于每种算法，请给出每步计算结果与真值的距离以及每步计算结果与最优解的距离。此外，请讨论正则化参数 p 对计算结果的影响。

2. 算法设计

(i) 邻近点梯度下降法

设

$$\begin{cases} s(\mathbf{x}) := \frac{1}{2} \|A\mathbf{x} - \mathbf{b}\|_2^2 \\ r(\mathbf{x}) := p \|\mathbf{x}\|_1 \end{cases}$$

其中， $A \in \mathbb{R}^{m \times n}$ ， $\mathbf{x} \in \mathbb{R}^n$ ， $\mathbf{b}, \mathbf{e} \in \mathbb{R}^m$ （在本题中 $m = 50, n = 100$ ），则原式

$$f(\mathbf{x}) := \left(\frac{1}{2} \|A\mathbf{x} - \mathbf{b}\|_2^2 + p \|\mathbf{x}\|_1 \right) = s(\mathbf{x}) + r(\mathbf{x})$$

先求 $r(\mathbf{x})$ 的邻近点投影

$$\text{prox } \hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \left(p \|\mathbf{x}\|_1 + \frac{1}{2\alpha} \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \right) \quad (1)$$

对1式右侧展开有

$$\arg \min_{\mathbf{x}} \left(p \sum_{i=1}^n |x_i| + \frac{1}{2\alpha} \sum_{i=1}^n (x_i - \hat{x}_i)^2 \right) \quad (2)$$

注意到(2)式对于下标 i 相互独立，故要求2的最小值，等价于对每一个下标 i 求最小值后求和，即

$$\arg \min_{x_i} \left(p|x_i| + \frac{1}{2\alpha} (x_i - \hat{x}_i)^2 \right), \forall i \quad (3)$$

由不可微函数的极值判断条件有

$$0 \in \partial_{x_i} p|x_i| + \frac{1}{\alpha} (x_i - \hat{x}_i) \quad (4)$$

对 x_i 进行分类讨论

- 若 $x_i > 0$ ，则 $|x_i|$ 对于 $|x_i|$ 可微，即 $\partial_{x_i} |x_i| = 1$ ，有

$$p + \frac{1}{\alpha} (x_i - \hat{x}_i) = 0$$

整理得

$$x_i = \hat{x}_i - \alpha p$$

由于 $x_i > 0$ ，故 $\hat{x}_i - \alpha p > 0$ ，即 $\hat{x}_i > \alpha p$

- 若 $x_i < 0$ ，则 $|x_i|$ 对于 $|x_i|$ 可微，即 $\partial_{x_i} |x_i| = -1$ ，有

$$-p + \frac{1}{\alpha} (x_i - \hat{x}_i) = 0$$

整理得

$$x_i = \hat{x}_i + \alpha p$$

由于 $x_i < 0$ ，故 $\hat{x}_i + \alpha p < 0$ ，即 $\hat{x}_i < -\alpha p$

- 若 $x_i = 0$ ，则 $|x_i|$ 对于 $|x_i|$ 不可微，需要求次梯度， $\partial_{x_i} |x_i| = [-1, 1]$ ，即

$$0 \in \left[-p - \frac{\hat{x}_i}{\alpha}, p - \frac{\hat{x}_i}{\alpha} \right]$$

那么，需要满足

$$\begin{cases} p - \frac{\hat{x}_i}{\alpha} \geq 0 \\ -p - \frac{\hat{x}_i}{\alpha} \leq 0 \end{cases}$$

推得

$$\hat{x}_i \in [-\alpha p, \alpha p]$$

综上, 有

$$x_i = \begin{cases} \hat{x}_i + \alpha p & \hat{x}_i < -\alpha p \\ 0 & \hat{x}_i \in [-\alpha p, \alpha p] \\ \hat{x}_i - \alpha p & \hat{x}_i > \alpha p \end{cases} \quad (5)$$

可以得到图1的软门限图

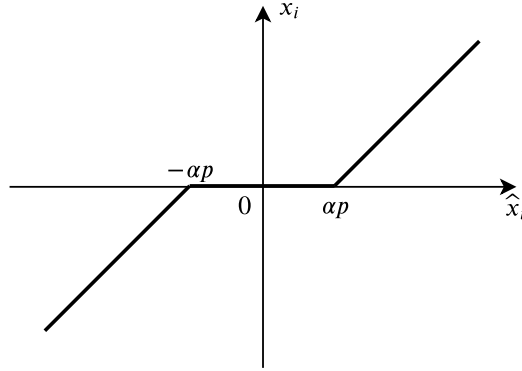


图 1: 关于 x_i 与 \hat{x}_i 的软门限图

进一步, 得到邻近点梯度下降法的迭代式如下

$$\begin{cases} \mathbf{x}^{(k+\frac{1}{2})} = \mathbf{x}^{(k)} - \alpha \nabla s(\mathbf{x}^{(k)}) = \mathbf{x}^{(k)} - \alpha A^T (A\mathbf{x}^{(k)} - \mathbf{b}) \\ \mathbf{x}^{(k+1)} = \text{prox } \mathbf{x}^{(k+\frac{1}{2})} = \arg \min_{\mathbf{x}} \left(p \|\mathbf{x}\|_1 + \frac{1}{2\alpha} \left\| \mathbf{x} - \mathbf{x}^{(k+\frac{1}{2})} \right\|_2^2 \right) \end{cases} \quad (6)$$

其中, $x^{(k+\frac{1}{2})}$ 可直接计算, $x^{(k+1)}$ 可由(5)式求得。

(ii) 交替方向乘子法

对原问题进行变形, 等价于下述约束问题

$$\begin{aligned} \min \quad & \frac{1}{2} \|A\mathbf{x} - \mathbf{b}\|_2^2 + p \|\mathbf{y}\|_1 \\ \text{s.t.} \quad & \mathbf{x} - \mathbf{y} = 0 \end{aligned}$$

构造增广拉格朗日函数

$$L_c(\mathbf{x}, \mathbf{y}, \mathbf{v}) = \frac{1}{2} \|A\mathbf{x} - \mathbf{b}\|_2^2 + p \|\mathbf{y}\|_1 + \mathbf{v}^T (\mathbf{x} - \mathbf{y}) + \frac{c}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 \quad (7)$$

可以得到交替方向乘子法的迭代格式

$$\begin{cases} \mathbf{x}^{(k+1)} = \arg \min_{\mathbf{x}} L_c(\mathbf{x}, \mathbf{y}^{(k)}, \mathbf{v}^{(k)}) \\ \mathbf{y}^{(k+1)} = \arg \min_{\mathbf{y}} L_c(\mathbf{x}^{(k+1)}, \mathbf{y}, \mathbf{v}^{(k)}) \\ \mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + c(\mathbf{x}^{(k+1)} - \mathbf{y}^{(k+1)}) \end{cases} \quad (8)$$

对 $\mathbf{x}^{(k+1)}$ 展开并配方，并将非主元项忽略，可求得(8)与下面的式子等价

$$\mathbf{x}^{(k+1)} = \arg \min_{\mathbf{x}} \left(\frac{1}{2} \|\mathbf{Ax} - \mathbf{b}\|_2^2 + \frac{c}{2} \left\| \mathbf{x} - \mathbf{y}^{(k)} + \frac{\mathbf{v}^{(k)}}{c} \right\|_2^2 \right) \quad (9a)$$

$$\mathbf{y}^{(k+1)} = \arg \min_{\mathbf{y}} \left(p \|\mathbf{y}\|_1 + \frac{c}{2} \left\| \mathbf{x}^{(k+1)} - \mathbf{y} + \frac{\mathbf{v}^{(k)}}{c} \right\|_2^2 \right) \quad (9b)$$

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + c(\mathbf{x}^{(k+1)} - \mathbf{y}^{(k+1)}) \quad (9c)$$

对于(9a)式，可直接通过求梯度的方法得到显式解，即

$$\mathbf{x}^{(k+1)} = (A^T A + cI)^{-1} (A^T \mathbf{b} + c\mathbf{y}^{(k)} - \mathbf{v}^{(k)})$$

对于(9b)式，由于涉及一范数，故需要求次微分，类似(5)式的方法，设 $z_i = x_i^{(k+1)} + \frac{v_i^{(k)}}{c}$ ，可得到

$$y_i = \begin{cases} z_i - \frac{p}{c} & z_i > \frac{p}{c} \\ 0 & z_i \in \left[-\frac{p}{c}, \frac{p}{c}\right] \\ z_i + \frac{p}{c} & z_i < -\frac{p}{c} \end{cases}$$

进而可以通过(9a)(9b)(9c)式迭代求解。

(iii) 次梯度法

对原式直接求次梯度有

$$\partial f(\mathbf{x}) = A^T (\mathbf{Ax} - \mathbf{b}) + p \partial \|\mathbf{x}\|_1 \quad (10)$$

其中

$$(\partial \|\mathbf{x}\|_1)_i = \partial |x_i| = \begin{cases} 1 & x_i > 0 \\ [-1, 1] & x_i = 0 \\ -1 & x_i < 0 \end{cases}$$

进而可以直接得到次梯度法的迭代格式

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \partial f(\mathbf{x}^{(k)}) \quad (11)$$

3. 数值实验

采用Python进行编程¹，对上述三种方法进行迭代计算，直到精度达到 10^{-8} 。

完整代码请见p1.py文件，这里只截取核心代码部分。

邻近点梯度法如下，设超参数 $\alpha = 10^{-3}$ 。

```

1 def soft_thresholding(x,offset):
2     if x < (-1)*offset:
3         return x + offset
4     elif x > offset:
5         return x - offset
6     else:
7         return 0
8
9 def prox(xk_old,offset):
10    xk_new = np.zeros(xk_old.size)
11    for i in range(xk_old.size):
12        xk_new[i] = soft_thresholding(xk_old[i],offset)
13    return xk_new
14
15 def proxgrad(res):
16    t = 0
17    xk = np.zeros(n)
18    while True:
19        xhat = xk - alpha * np.dot(A.T, np.dot(A, xk) - b)
20        xk_new = prox(xhat, alpha * p)
21        if np.linalg.norm(xk_new - xk, ord=2) < acc:
22            break
23        res.append(xk_new)
24        xk = xk_new.copy()
25        t += 1
26    print(t)
27    return xk

```

交替方向乘子法如下，设超参数 $c = 10^{-3}$ 。

```

1 def admm(res):
2     t = 0
3     xk = np.zeros(n)
4     yk = np.zeros(n)

```

¹使用numpy进行数值计算，matplotlib进行画图

```

5 vk = np.zeros(n)
6 while True:
7     xk_new = np.dot(
8         np.linalg.inv(np.dot(A.T, A) + c * np.eye(n,n)),
9         np.dot(A.T, b) + c * yk - vk)
10    yk_new = prox(xk_new + vk / c, p / c)
11    vk_new = vk + c * (xk_new - yk_new)
12    if np.linalg.norm(xk_new - xk, ord=2) < acc:
13        break
14    res.append(xk_new)
15    xk = xk_new.copy()
16    yk = yk_new.copy()
17    vk = vk_new.copy()
18    t += 1
19 print(t)
20 return xk

```

次梯度法如下，采用递减步长 $\alpha^{(k+1)} = \frac{\alpha^{(k)}}{k+1}$ 。

```

1 def subgrad(res):
2     xk = np.zeros(n)
3     t = 0
4     while True:
5         pdx = np.zeros(xk.size)
6         alphak = alpha / (t + 1) # remember to decay the step
7         for i in range(xk.size):
8             if xk[i] != 0:
9                 pdx[i] = 1 if xk[i] > 0 else -1
10            else: # pick a random float from [-1,1]
11                pdx[i] = 2 * np.random.random() - 1
12        pdf = np.dot(A.T, np.dot(A, xk) - b) + pdx
13        xk_new = xk - alphak * pdf
14        if np.linalg.norm(xk_new - xk, ord=2) < acc:
15            break
16        res.append(xk_new)
17        xk = xk_new.copy()
18        t += 1
19    print(t)
20    return xk

```

4. 结果分析

记最优解 \mathbf{x}^* 为最后一次迭代获得的 \mathbf{x} 值，真值为 \mathbf{x}_{true} 。

运行结果如图2、图3和图4所示。

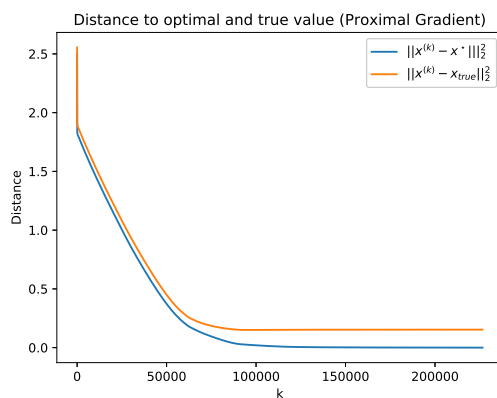


图 2: 邻近点梯度下降法

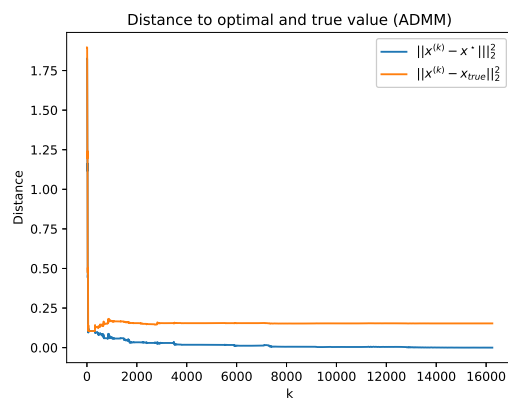


图 3: 交替方向乘法

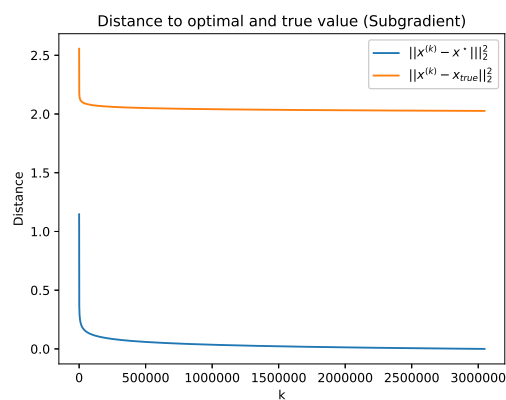


图 4: 次梯度法

由实际的运行时间可以得出，交替方向乘法达到目标精度需要的时间最短，邻近点梯度下降法次之，次梯度法所需的时间最长。从迭代的次数也可以看出，邻近点梯度下降法需要226545轮迭代，交替方向乘法需要16249轮迭代，次梯度法需要3047223轮迭代。（此实验仅仅展示了一种情况，实际上迭代次数与超参数的选择有关，见下面的实验。）

不仅如此，从上述三幅图中还可以看出，交替方向乘子法和邻近点梯度下降法都能达到很高的精度（即与真值相差不多），但次梯度法迭代次数多且精度非常低，是一种非常糟糕的算法。

关于正则化参数 p 对最终结果的影响可见图5、图6和图7。

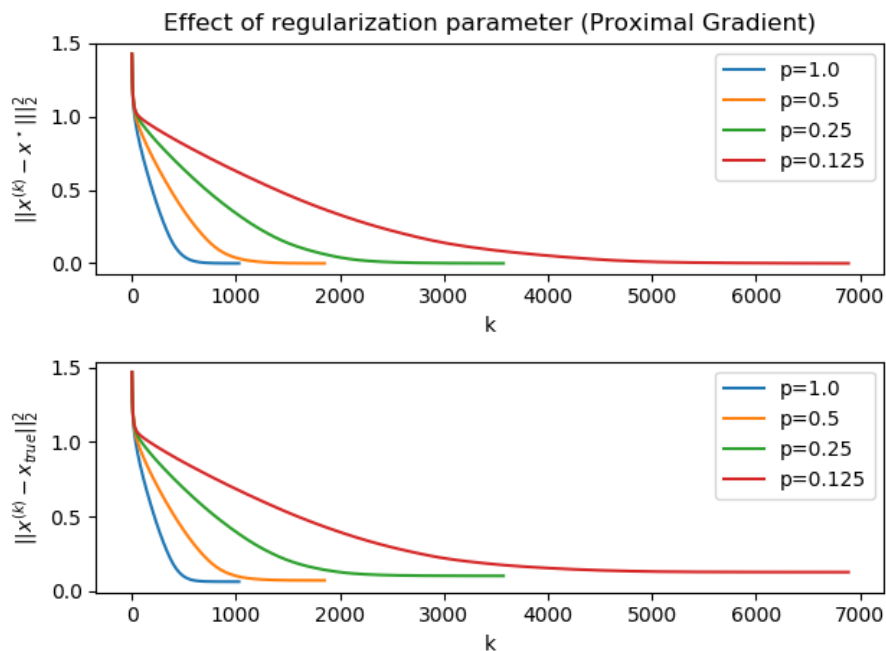


图 5: 邻近点梯度下降法

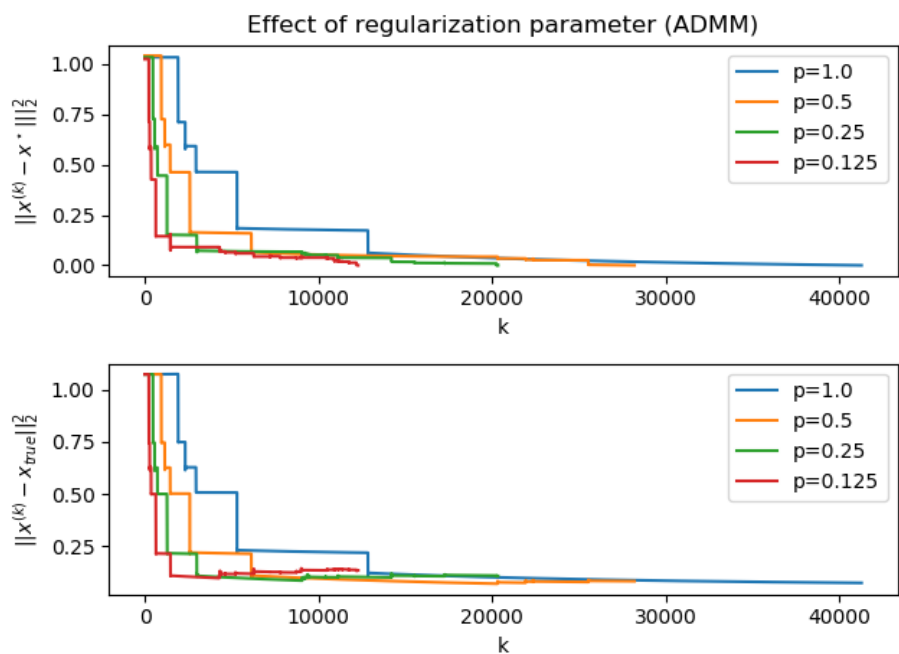


图 6: 交替方向乘法

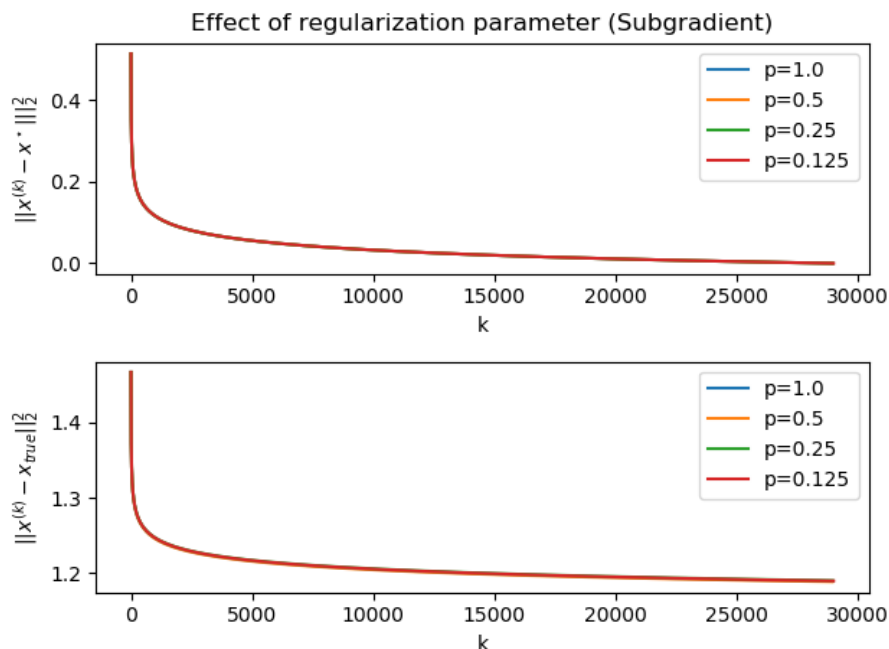


图 7: 次梯度法

从图中可以看出，当正则化参数 p 对迭代次数和收敛精度都有一定的影响。对于邻近点梯度法， p 取较大的值时，收敛速度比较快，且精度也较高。对于交替方向乘子法， p 取较小的值时，收敛速度较快，但精度差异没有邻近点梯度法那么明显。而对于次梯度法， p 对收敛速度和收敛精度几乎没有影响。

二、问题二

1. 问题描述

请设计下述算法，求解MNIST数据集上的Logistic Regression问题：

1. 梯度下降法
2. 随机梯度法

对于每种算法，请给出每步计算结果与最优解的距离以及每步计算结果在测试集上所对应的分类精度。此外，请讨论随机梯度法中Mini Batch大小对计算结果的影响。

可参考：<http://deeplearning.net/tutorial/logreg.html>

2. 算法设计

由于MNIST手写数据集为多分类问题，需要将输入的图片映射到对应的10个数字上，故采用多类别的logistic回归，即softmax回归进行求解。

设训练集 $\{(\mathbf{x}_i, y_i)\}_{i=1}^M$ ，其中 M 为训练集数目， $|C|$ 为类别数目， $\mathbf{x}_i \in \mathbb{R}^{n+1}$ 为输入样本， $y_i \in \{0, 1, \dots, |C| - 1\}$ 为对应的标签。在本问题中 $|C| = 10$ ， $M = 60000$ ， $n = 28 \times 28 = 784$ ，这里

已经将二维的图片数组展平为一维的向量。

输入 \mathbf{x}_i 属于第 y_i 类的条件概率可用softmax函数进行计算

$$p(y_i | \mathbf{x}_i; W) = \frac{e^{\mathbf{w}_{y_i}^T \mathbf{x}_i}}{\sum_{c=0}^{|C|-1} e^{\mathbf{w}_c^T \mathbf{x}_i}} \quad (12)$$

其中,

$$W = \begin{bmatrix} \mathbf{w}_0^T \\ \mathbf{w}_1^T \\ \vdots \\ \mathbf{w}_{|C|-1}^T \end{bmatrix} \in \mathbb{R}^{|C| \times (n+1)}$$

为需要训练的参数, $\mathbf{w}_i \in \mathbb{R}^{n+1}$ 为每一个类别对应的权重向量²。

从而得到数据集的似然函数为

$$\mathcal{L}(Y | X; W) = \prod_{i=1}^m p(y_i | \mathbf{x}_i; W)$$

对应的对数似然函数为

$$\ln \mathcal{L}(Y | X; W) = \sum_{i=1}^m \ln p(y_i | \mathbf{x}_i; W)$$

极大化对数似然函数, 相当于极小化负对数似然函数(negative log likelihood, NLL), 故得到最优化问题

$$\min_W \ell(W) := -\ln \mathcal{L}(Y | X; W) \quad (13)$$

$\ell(W)$ 其实也是深度学习领域常说的损失函数(loss)。

对 $\ln p(y_i | \mathbf{x}_i; W)$ 求梯度, 有

$$\frac{\partial \ln p(y_i | \mathbf{x}_i; W)}{\partial \mathbf{w}_j} = \begin{cases} \mathbf{x}_i(1 - p(y_i | \mathbf{x}_i; W)) & y_i = j \\ \mathbf{x}_i p(y_i | \mathbf{x}_i; W) & y_i \neq j \end{cases}, j \in \{0, 1, \dots, |C| - 1\}$$

进而

$$\nabla_{\mathbf{w}_j} \ell(W) = - \sum_{i=1}^m [\mathbf{x}_i (\mathbb{1}(y_i = j) - p(y_i | \mathbf{x}_i; W))] \quad (14)$$

其中, $\mathbb{1}(\cdot)$ 为示性函数, 当输入为真时返回1, 输入为假时返回0。

²注意到 $\mathbf{w}^T \mathbf{x} + b = [\mathbf{w}^T \quad b] \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$, 故将最后的偏置项归入 \mathbf{w} 中, 并在 \mathbf{x} 中添加为1的一个维度, 进而偏置量和权重可以一起运算, \mathbf{w}_c^T 和 \mathbf{x}_i 都有 $n+1$ 个维度。

最终得到梯度下降的表达式

$$\mathbf{w}_j^{(k+1)} = \mathbf{w}_j^{(k)} - \frac{\alpha^{(k)}}{m} \nabla_{\mathbf{w}_j} \ell(W) \quad (15)$$

其中， m 为小批量大小。当 m 选取不同值时，(15)式为不同的优化方法：

- 当 $m = M$ 时，梯度下降法，每次选取**所有样本**用于更新权重
- 当 $m = 1$ 时，随机梯度下降法，每次选取**一个样本**用于更新权重
- 当 $m \in [2, M - 1]$ 时，小批量梯度下降法，每次只选取**部分样本**用于更新权重

3. 数值实验

MNIST输入数据以numpy压缩格式存储，从<https://s3.amazonaws.com/img-datasets/mnist.npz>获取。可从中直接读出`x_train`、`y_train`、`x_test`和`y_test`。

核心代码如下，完整代码请见p2.py。

```

1 class SoftmaxRegression():
2
3     def train(self, X, y_true, n_classes, n_iters=10, learning_rate=0.1):
4         self.n_samples, n_features = X.shape
5         self.n_classes = n_classes
6
7         self.weights = np.random.rand(self.n_classes, n_features)
8         self.bias = np.zeros((1, self.n_classes))
9         all_losses = []
10        all_accuracy = []
11
12        for i in range(n_iters):
13            scores = self.compute_scores(X)
14            probs = self.softmax(scores)
15            y_predict = np.argmax(probs, axis=1)[: , np.newaxis]
16            y_one_hot = self.one_hot(y_true)
17
18            loss = self.nll_loss(y_one_hot, probs)
19            all_losses.append(loss)
20
21            dw = (1 / self.n_samples) * np.dot(X.T, (probs - y_one_hot))
22            db = (1 / self.n_samples) * np.sum(probs - y_one_hot, axis=0)
23
24            self.weights = self.weights - learning_rate * dw.T
25            self.bias = self.bias - learning_rate * db
26
27            if i % 100 == 0 or i == n_iters - 1:
28                y_predict = self.predict(X_test)

```

```
29         all_accuracy.append((np.sum(y_predict == y_test)/X_test.shape[0]) *  
30             ↪ 100)  
31         print(f'Iteration number: {i}, loss: {np.round(loss, 4)}, accuracy:  
32             ↪ {all_accuracy[-1]}%')  
33  
34     return self.weights, self.bias, all_losses, all_accuracy  
35  
36 def predict(self, X):  
37     scores = self.compute_scores(X)  
38     probs = self.softmax(scores)  
39     return np.argmax(probs, axis=1)[: , np.newaxis]  
40  
41 def softmax(self, scores):  
42     exp = np.exp(scores) # (n_samples, n_classes)  
43     sum_exp = np.sum(np.exp(scores), axis=1, keepdims=True)  
44     softmax = exp / sum_exp  
45     return softmax # (n_samples, n_classes)  
46  
47 def compute_scores(self, X):  
48     """  
49     X: (n_samples, n_features)  
50     scores: (n_samples, n_classes)  
51     """  
52     return np.dot(X, self.weights.T) + self.bias  
53  
54 def nll_loss(self, y_true, probs):  
55     loss = - (1 / self.n_samples) * np.sum(y_true * np.log(probs))  
56     return loss  
57  
58 def one_hot(self, y):  
59     one_hot = np.zeros((self.n_samples, self.n_classes))  
60     one_hot[np.arange(self.n_samples), y.T] = 1  
61     return one_hot
```

4. 结果分析

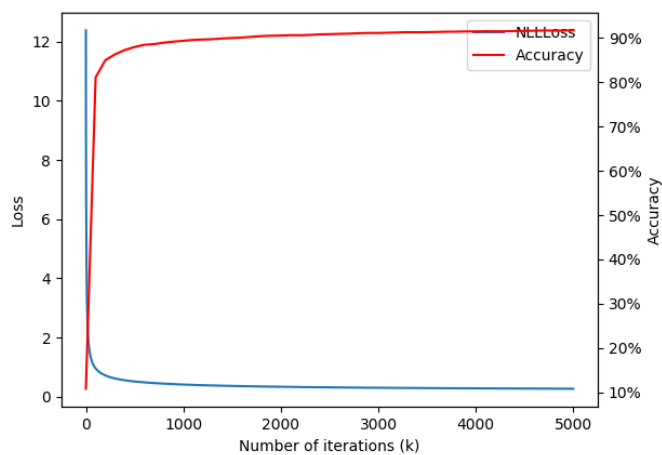


图 8: 与最优解距离及在测试集上的分类精度

从图8中可以看出, 运用梯度下降法求解, 损失函数不断下降, 在测试集上的分类精度不断上升。

批量大小、精度和训练时间的比较如表1所示。

表 1: 批量大小、精度和训练时间的比较

批量大小	精度	训练时间
60000	91.7%	933.93s

三、参考资料

1. Stanford CS231n, <http://cs231n.github.io/>
2. Stanford UFLDL Tutorial - Softmax Regression, <http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/>