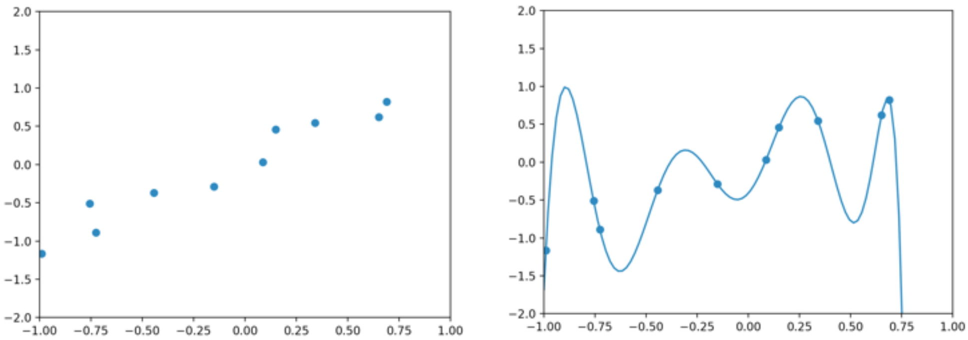
无约束经验损失最小化

给定样本空间 X ，标签空间 Y ，模型空间 Φ ，损失函数 l ： $Y \times Y \to \mathbb{R}^+$

输入： m 个训练数据 $S = \left\{\left(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}\right), \left(\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}\right), \ldots, \left(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}\right)\right\}$

输出模型： $h_S = \mathrm{argmin}_{h \in \Phi} L_S(h)$

图 2.1　经验损失最小化架构



(a) (b)

图 2.2　完美拟合训练数据的多项式 h_S

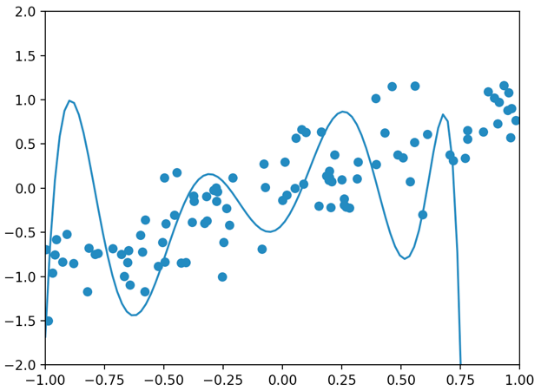
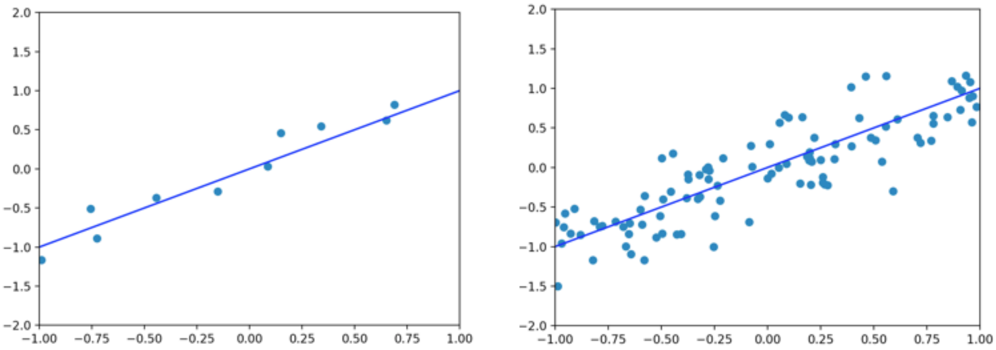


图 2.3　h_S 对 100 个测试数据的拟合效果



(a)　　　　　　　　　　(b)

图 2.4　线性模型对训练数据与测试数据的拟合效果

经验损失最小化

给定样本空间 X，标签空间 Y 以及损失函数 l： $Y \times Y \to \mathbb{R}^{+}$

取定模型假设 H

输入： m 个训练数据 $S = \left\{\left(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}\right), \left(\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}\right), \dots, \left(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}\right)\right\}$

输出模型：$h_S = \operatorname{argmin}_{h \in H} L_S(h)$

图 2.5　带模型假设的经验损失最小化架构

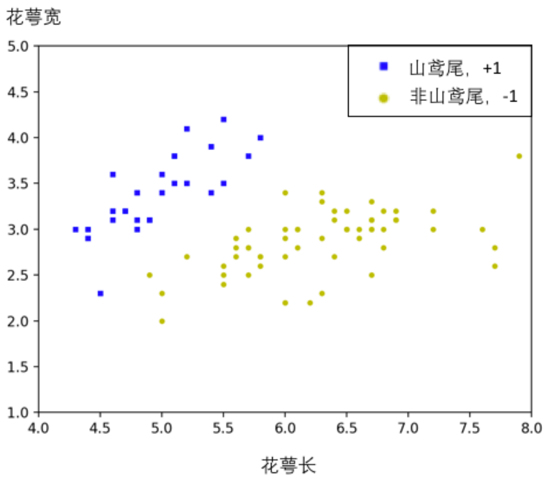


图 2.6　90 条训练数据的散点图

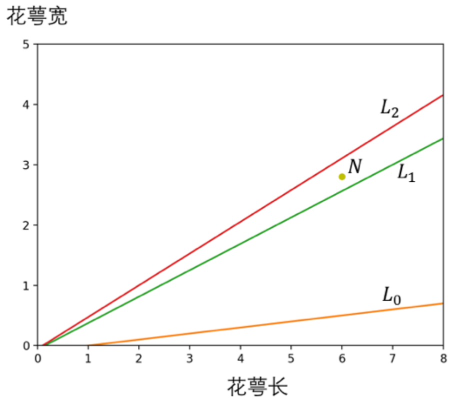


图 2.7　感知器算法运行过程

感知器算法

$\boldsymbol{w} = (0,0)$，$b = 0$,　done = False

While not done:

　done = True

　For $i = 1,2 \ldots, m$:

　　If $y^{(i)}\,\mathrm{Sign}\big(\langle \boldsymbol{w}, \boldsymbol{x}^{(i)} \rangle + b\big) \le 0$:

　　　$\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(i)} \boldsymbol{x}^{(i)}$

　　　$b \leftarrow b + y^{(i)}$

　　　done = False

Return $\boldsymbol{w}, b$

图 2.8　感知器算法描述

machine_learning.lib.perceptron

```python
import numpy as np

class Perceptron:
    def fit(self, X, y):
        m, n = X.shape
        w = np.zeros((n,1))
        b = 0
        done = False
        while not done:
            done = True
            for i in range(m):
                x = X[i].reshape(1,-1)
                if y[i] * (x.dot(w) + b) <= 0:
                    w = w + y[i] * x.T
                    b = b + y[i]
                    done = False
        self.w = w
        self.b = b

    def predict(self, X):
        return np.sign(X.dot(self.w) + self.b)
```

图 2.9　感知器算法实现

```python
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from machine_learning.lib.perceptron import Perceptron

iris = datasets.load_iris()
X= iris["data"][:,(0,1)]
y = 2 * (iris["target"]==0).astype(np.int) - 1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=5)

model = Perceptron()
model.fit(X_train, y_train)
model.predict(X_test)
```

图 2.10 感知器算法预测鸢尾花属性

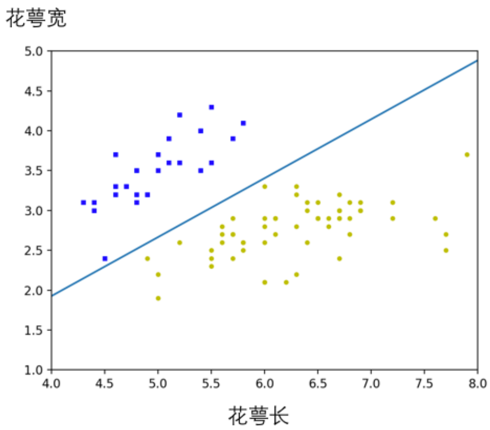


图 2.11　感知器算法对训练数据的区分效果

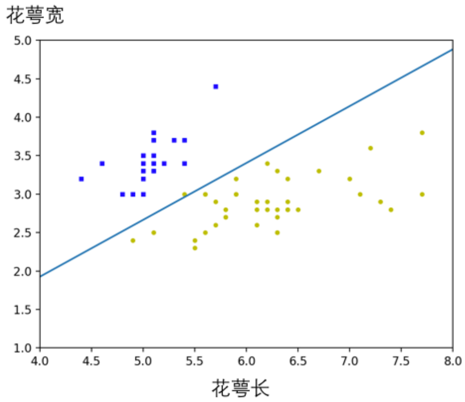


图 2.12　感知器算法对测试数据的区分效果

L_1正则化的经验损失最小化

参数化的模型假设 $H = \{h_{\boldsymbol{w}} : \boldsymbol{w} \in \mathbb{R}^n\}$

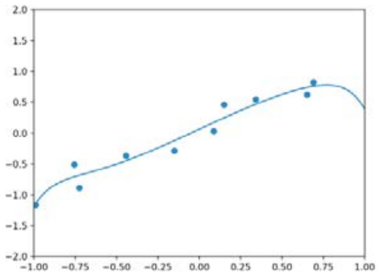
输入： m 个训练数据 $S = \{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), (\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}), \dots, (\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)})\}$

计算优化问题的最优解 $\boldsymbol{w}^*$：

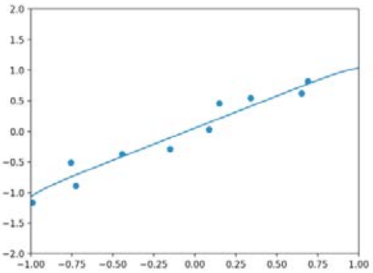
$$\min_{\boldsymbol{w}\in\mathbb{R}^n} L_S(h_{\boldsymbol{w}}) + \lambda|\boldsymbol{w}|$$

输出模型：$h_{\boldsymbol{w}^*}$

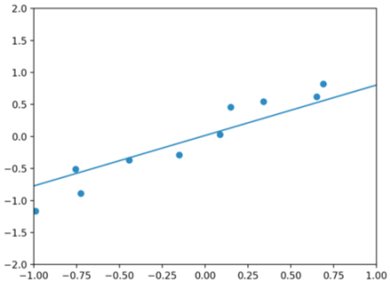
图 2.13 L_1 正则化的经验损失最小化算法



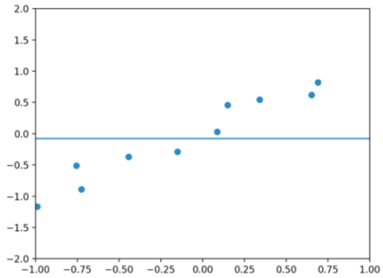
(a) $\lambda = 0.001$



(b) $\lambda = 0.01$



(c) $\lambda = 0.1$



(d) $\lambda = 0.5$

图 2.14　不同 L_1 正则化系数的模型

L_2 正则化的经验损失最小化

参数化的模型假设 $H=\{h_{\boldsymbol{w}}:\boldsymbol{w}\in\mathbb{R}^n\}$

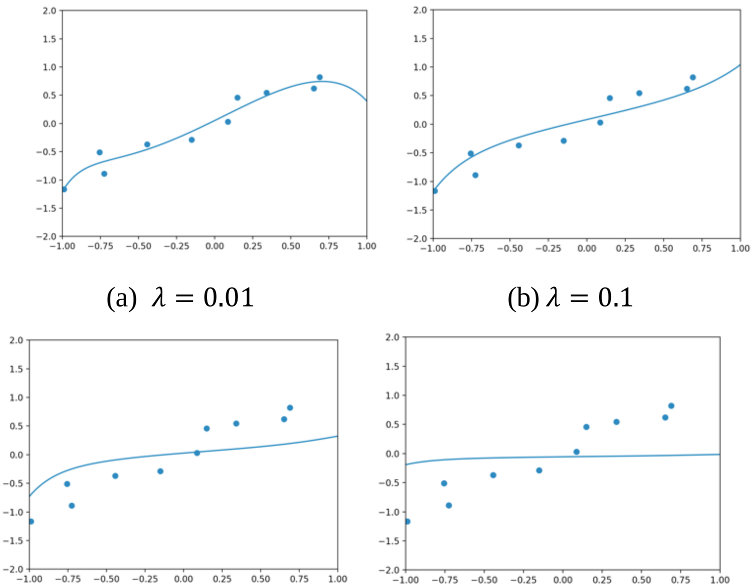
输入： m 个训练数据 $S=\{(\boldsymbol{x}^{(1)},\boldsymbol{y}^{(1)}),\ (\boldsymbol{x}^{(2)},\boldsymbol{y}^{(2)}),\dots,\ (\boldsymbol{x}^{(m)},\boldsymbol{y}^{(m)})\}$

计算优化问题的最优解 $\boldsymbol{w}^*$：

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} L_S(h_{\boldsymbol{w}})+\lambda\|\boldsymbol{w}\|^2$$

输出模型：$h_{\boldsymbol{w}^*}$

图 2.15　L_2 正则化的经验损失最小化算法



(a) $\lambda = 0.01$　(b) $\lambda = 0.1$

(c) $\lambda = 10$　(d) $\lambda = 100$

图 2.16 不同 L_2 正则化系数的模型

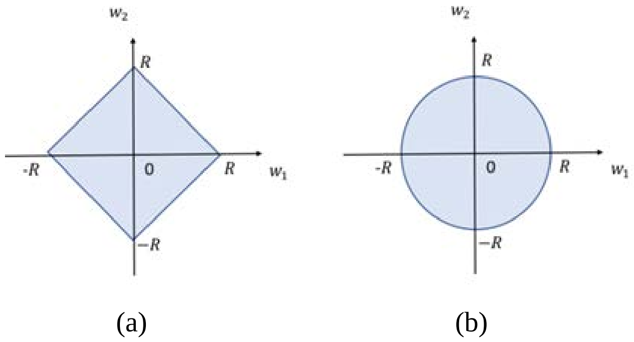


图 2.17　L_1 与 L_2 正则化算法的可行解区域