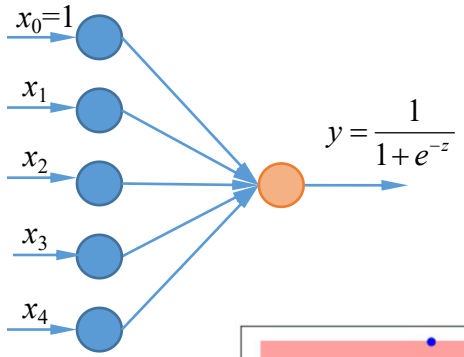
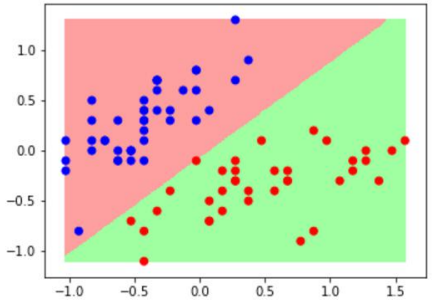
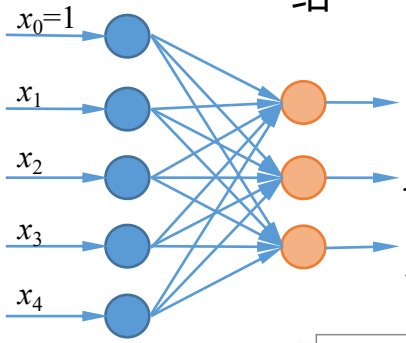
12.2 实例：实现单层神经网络

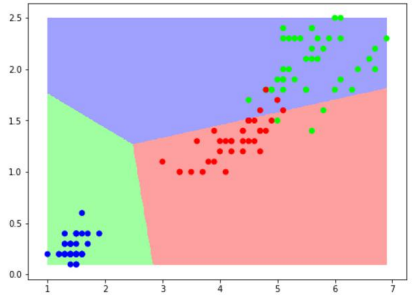
逻辑回归 —— 感知机





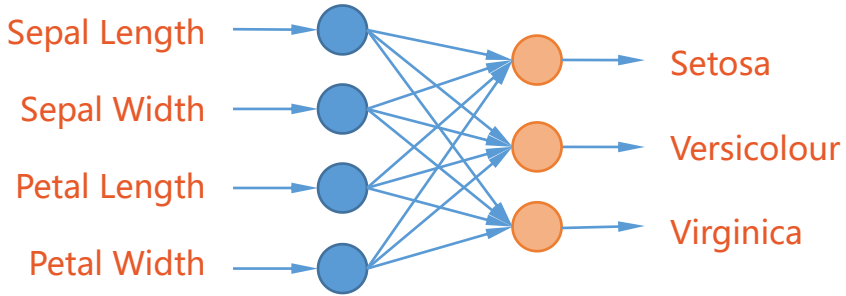
sofmax回归 —— 单层神经网络





口 神经网络的设计

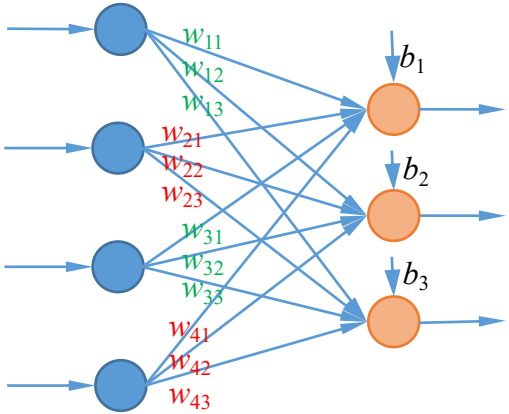
神经网络的结构	单层前馈型神经网络
激活函数	softmax函数
损失函数	交叉熵损失函数



独热编码

神经网络的实现

1	6.4	2.8	5.6	2.2
2	5	2.3	3.3	1
3	4.9	2.5	4.5	1.7
4	4.9	3.1	1.5	0.1
5	5.7	3.8	1.7	0.3
6	4.4	3.2	1.3	0.2
7	5.4	3.4	1.5	0.4
119	4.8	3	1.4	0.1
120	5.5	2.4	3.7	1



1	0	0	1	[3.04389629e-04, 1.29587591e-01, 8.70108008e-01]
2	0	1	0	[1.75134137e-01, 6.48399115e-01, 1.76466733e-01]
3	0	0	1	[7.53107527e-03, 3.44870061e-01, 6.47598863e-01]
4	1	0	0	[9.13596153e-01, 8.42635259e-02, 2.14024656e-03]
5	1	0	0	[8.66889775e-01, 1.27143607e-01, 5.96662890e-03]
6	1	0	0	[9.33885932e-01, 6.36122525e-02, 2.50180368e-03]
7	1	0	0	[8.95907521e-01, 9.70410407e-02, 7.05144275e-03]
119	1	0	0	[9.26600277e-01, 7.15029836e-02, 1.89674716e-03]
120	0	1	0	[9.72348675e-02, 7.34697282e-01, 1.68067887e-01]

(120, 4)

$$W=\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{31} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix}$$

(4, 3)

$B = [\, b_1, b_2, b_3 \,]$

(120, 3)

$Y=XW+B$

softmax函数

$Y=XW+B$

tf. nn. softmax()

tf.nn.softmax(tf.matmul(X_train,W)+b)

独热编码

tf. one_hot(indices, depth)

tf.one_hot(tf.constant(y_test, dtype=tf.int32), 3)

交叉熵损失函数

表示为独热编码的标签值

softmax函数的输出

tf.keras.losses.categorical_crossentropy(y_true, y_pred)

tf.reduce_mean(tf.keras.losses.categorical_crossentropy(y_true=Y_train,y_pred=PRED_train))

- 导入库

In [1]:
```
import tensorflow as tf
print("TensorFlow version:", tf.__version__)
```

```
TensorFlow version: 2.0.0
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
gpus = tf.config.experimental.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)
```

```
InternalError: Blas GEMM launch failed :
```

for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)

加载数据

```
In [4]: TRAIN_URL = "http://download.tensorflow.org/data/iris_training.csv"
        train_path = tf.keras.utils.get_file(TRAIN_URL.split('/')[-1], TRAIN_URL)

        TEST_URL = "http://download.tensorflow.org/data/iris_test.csv"
        test_path = tf.keras.utils.get_file(TEST_URL.split('/')[-1], TEST_URL)

In [5]: df_iris_train = pd.read_csv(train_path, header=0)
        df_iris_test = pd.read_csv(test_path, header=0)

In [6]: iris_train=np.array(df_iris_train)
        iris_test=np.array(df_iris_test)

In [7]: iris_train.shape,iris_test.shape

Out[7]: ((120, 5), (30, 5))
```

- 数据预处理

```
In [8]: x_train=iris_train[:,0:4]
        y_train=iris_train[:,4]

        x_test=iris_test[:,0:4]
        y_test=iris_test[:,4]

In [9]: x_train.shape,y_train.shape
Out[9]: ((120, 4), (120,))

In [10]: x_test.shape,y_test.shape
Out[10]: ((30, 4), (30,))

In [11]: x_train=x_train-np.mean(x_train,axis=0)
         x_test=x_test-np.mean(x_test,axis=0)
```

```
In [12]: x_train.dtype, y_train.dtype

Out[12]: (dtype('float64'), dtype('float64'))

In [13]: X_train=tf.cast(x_train, tf.float32)
         Y_train=tf.one_hot(tf.constant(y_train, dtype=tf.int32), 3)

         X_test=tf.cast(x_test, tf.float32)
         Y_test=tf.one_hot(tf.constant(y_test, dtype=tf.int32), 3)

In [14]: X_train.shape, Y_train.shape

Out[14]: (TensorShape([120,  4]), TensorShape([120,  3]))

In [15]: X_test.shape, Y_test.shape

Out[15]: (TensorShape([30,  4]), TensorShape([30,  3]))
```

- 设置超参数和显示间隔

In [16]:
```
learn_rate = 0.5
iter = 50

display_step =10
```

- 设置模型参数初始值

In [17]:
```
np.random.seed(612)
W = tf.Variable(np.random.randn(4,3),dtype=tf.float32)
B = tf.Variable(np.zeros([3]),dtype=tf.float32)
```

- 训练模型

```
In [18]: acc_train=[]
         acc_test=[]
         cce_train=[]
         cce_test=[]
```

```
for i in range(0,iter+1):
    with tf.GradientTape() as tape:
        PRED_train=tf.nn.softmax(tf.matmul(X_train,W)+B)
        Loss_train=tf.reduce_mean(tf.keras.losses.categorical_crossentropy(y_true=Y_train,y_pred=PRED_train))

    PRED_test=tf.nn.softmax(tf.matmul(X_test,W)+B)
    Loss_test=tf.reduce_mean(tf.keras.losses.categorical_crossentropy(y_true=Y_test,y_pred=PRED_test))

    accuracy_train=tf.reduce_mean(tf.cast(tf.equal(tf.argmax(PRED_train.numpy(),axis=1),y_train),tf.float32))
    accuracy_test=tf.reduce_mean(tf.cast(tf.equal(tf.argmax(PRED_test.numpy(),axis=1),y_test),tf.float32))

    acc_train.append(accuracy_train)
    acc_test.append(accuracy_test)
    cce_train.append(Loss_train)
    cce_test.append(Loss_test)

    grads = tape.gradient(Loss_train,[W,B])
    W.assign_sub(learn_rate*grads[0])
    B.assign_sub(learn_rate*grads[1])

    if i % display_step == 0:
        print("i: %i, TrainAcc:%f, TrainLoss: %f ,TestAcc:%f, TestLoss: %f" % (i,accuracy_train,Loss_train,accuracy_test,Loss_test))
```

dL_dw (4,3)

dL_db (3,)

训练结果

```
i: 0, TrainAcc:0.333333, TrainLoss: 2.066978 ,TestAcc:0.266667, TestLoss: 1.880856
i: 10, TrainAcc:0.875000, TrainLoss: 0.339410 ,TestAcc:0.866667, TestLoss: 0.461705
i: 20, TrainAcc:0.875000, TrainLoss: 0.279647 ,TestAcc:0.866667, TestLoss: 0.368414
i: 30, TrainAcc:0.916667, TrainLoss: 0.245924 ,TestAcc:0.933333, TestLoss: 0.314814
i: 40, TrainAcc:0.933333, TrainLoss: 0.222922 ,TestAcc:0.933333, TestLoss: 0.278643
i: 50, TrainAcc:0.933333, TrainLoss: 0.205636 ,TestAcc:0.966667, TestLoss: 0.251937
```

结果可视化

```
In [20]: plt.figure(figsize=(10,3))

         plt.subplot(121)
         plt.plot(cce_train, color="blue", label="train")
         plt.plot(cce_test, color="red", label="test")
         plt.xlabel("Iteration")
         plt.ylabel("Loss")
         plt.legend()

         plt.subplot(122)
         plt.plot(acc_train, color="blue", label="train")
         plt.plot(acc_test, color="red", label="test")
         plt.xlabel("Iteration")
         plt.ylabel("Accuracy")
         plt.legend()

         plt.show()
```