TP03-LogisticRegression

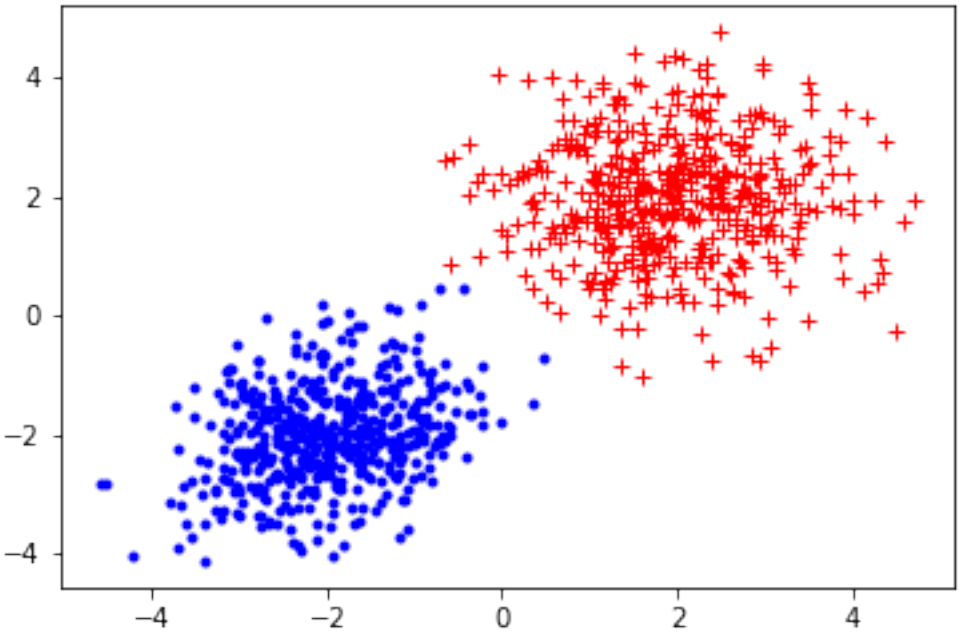
September 18, 2019

1 Logistic Regression

The following script generate a fake classification dataset.

[1]:
```
import numpy as np
import tensorflow as tf

import matplotlib.pyplot as plt
%matplotlib inline
import datetime as dt

def generate_all_dataset():
    # --- Fake dataset ---

    np.random.seed(0)

    ntrain = 1000
    nvalid = 100
    ntest = 100

    mupos = np.array([2., 2.])
    sigmapos = np.array([[1., 0.], [0., 1.]])
    muneg = np.array([-2., -2.])
    sigmaneg = np.array([[.7, .2], [.2, .7]])

    def generate_a_dataset(mupos, sigmapos, muneg, sigmaneg, n):
        npos = int(n/2)
        nneg = n - npos

        Xpos = np.random.multivariate_normal(mupos, sigmapos, npos)
        Ypos = np.stack((np.ones((npos,)), np.zeros((npos,))), axis=1)

        Xneg = np.random.multivariate_normal(muneg, sigmaneg, nneg)
        Yneg = np.stack((np.zeros((nneg,)), np.ones((nneg,))), axis=1)

        X, Y = np.concatenate((Xpos, Xneg)), np.concatenate((Ypos, Yneg))
```

```
        idx = np.arange(n)
        np.random.shuffle(idx)
        X, Y = X[idx], Y[idx]

        return np.array(X, dtype='float32'), np.array(Y, dtype='float32')

    Xtrain, Ytrain = generate_a_dataset(
        mupos, sigmapos, muneg, sigmaneg, ntrain)
    Xvalid, Yvalid = generate_a_dataset(
        mupos, sigmapos, muneg, sigmaneg, nvalid)
    Xtest, Ytest = generate_a_dataset(mupos, sigmapos, muneg, sigmaneg, ntest)

    return (Xtrain, Ytrain), (Xvalid, Yvalid), (Xtest, Ytest)


def plot_dataset(X, Y):
    plt.figure()
    idpos, = np.nonzero(Y[:, 0] == 1.)
    idneg, = np.nonzero(Y[:, 1] == 1.)
    plt.plot(X[idpos, 0], X[idpos, 1], 'r+')
    plt.plot(X[idneg, 0], X[idneg, 1], 'b.')
    plt.show()


def demo(trainset, validset, testset):
    plot_dataset(*trainset)


demo(*generate_all_dataset())
```



1.1 Exercise

Inspired by the precedent object oriented verision of the linear regression, write a logistic regression model:

$$\hat{y} = softmax(< x, A > +b)$$

It can be learnt by the cross-entropy loss:

$$\mathcal{L}(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

In fact, that's a **one layer perceptron** !

You will have to use the following operations: - `tf.matmul` - `tf.nn.softmax` - `tf.losses.softmax_cross_entropy`

Please note that the `softmax_cross_entropy` loss takes as input the logits, i.e. the output of the linear model before the softmax, and that both `softmax` and `softmax_cross_entropy` need a two column input (one per class).

Guidelines:

1) Copy the code of `LinearRegressionV3` class and rename the class to `LogisticRegressionV1`
2) Modify `_build_network` method to have a logistic regression model and a cross-entropy loss
3) Add a `_compute_pred_loss(self, set_iterator_init, nbatches)` method that return the prediction and the loss
4) Modify `train` method by using `_compute_pred_loss` on the test set, and make `train` return the prediction of the test set

[2]:
```
# object oriented verision of the linear regression
class LinearRegression:

    def __init__(self, dtype, ninputs, noutputs,
                 learning_rate=5e-3, training_epochs=100, batchsize=50):

        self.nc = ninputs
        self.no = noutputs

        self.dt_shapes = (tf.TensorShape((None, ninputs)),
                          tf.TensorShape((None, noutputs)))
        self.dt_types = (dtype, dtype)

        self.learning_rate = learning_rate
        self.training_epochs = training_epochs
        self.batchsize = batchsize

        self._build_network()

        # Global variable initializer
        self.init_global = tf.initializers.global_variables()

    def _build_network(self):
        # Create ONLY ONE iterator base on types and shapes of one of the␣
↪dataset
        # both dataset should have the same types and shapes...
        self.dataset_iterator = tf.data.Iterator.from_structure(
            self.dt_types, self.dt_shapes)

        # Placeholders from the dataset iterator
        x, y = self.dataset_iterator.get_next()

        # Linear regression parameters
        self.A = tf.Variable(tf.zeros([self.nc, self.no]))
        self.b = tf.Variable(tf.zeros([self.no]))
        # All parameters are gathered into var_list
        var_list = [self.A, self.b]

        # Actual linear regression
        self.pred = tf.matmul(x, self.A) + self.b

        # Model loss
        self.loss = tf.losses.mean_squared_error(y, self.pred)

        self.optimizer = tf.train.GradientDescentOptimizer(
            self.learning_rate).minimize(self.loss, var_list=var_list)
```

```
    def _prepareset(self, dataset, shuffle=True):
        if shuffle:
            dataset = dataset.shuffle(buffer_size=1000)
        dataset = dataset.batch(self.batchsize)
        return dataset

    def _compute_loss(self, set_iterator_init, nbatches):
        self.session.run(set_iterator_init)
        lossval = 0.
        for b in range(nbatches):
            batch_lossval, = self.session.run([self.loss])
            lossval += batch_lossval
        lossval /= nbatches
        return lossval

    def _compute_gradient_step(self, set_iterator_init, nbatches):
        self.session.run(set_iterator_init)
        lossval = 0.
        for b in range(nbatches):
            _, batch_lossval, = self.session.run([self.optimizer, self.loss])
            lossval += batch_lossval
        lossval /= nbatches
        return lossval

    def train(self, trainset, validset, testset):

        (Xtrain, Ytrain) = trainset
        (Xvalid, Yvalid) = validset
        (Xtest, Ytest) = testset

        # --- Linear Regression ---

        ntrain, _ = Xtrain.shape
        ntest, _ = Xtest.shape
        nvalid, _ = Xvalid.shape

        trainset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtrain, Ytrain)))
        testset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtest, Ytest)), shuffle=False)
        validset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xvalid, Yvalid)), shuffle=False)

        ntrainbatches = int(np.ceil(ntrain/self.batchsize))
        ntestbatches = int(np.ceil(ntest/self.batchsize))
        nvalidbatches = int(np.ceil(nvalid/self.batchsize))
```

```
        # Create one initializer per dataset
        training_init_op = self.dataset_iterator.make_initializer(trainset)
        test_init_op = self.dataset_iterator.make_initializer(testset)
        validation_init_op = self.dataset_iterator .make_initializer(validset)


        with tf.Session() as self.session:

            # We call the initialization of A and b
            self.session.run(self.init_global)

            # We compute the train and validation loss
            # Note that you just have to change the feed_dict to change the set

            trainloss = self._compute_loss(training_init_op, ntrainbatches)
            validationloss = self._compute_loss(
                validation_init_op, nvalidbatches)
            print("Init\t\t train loss %f\t valid loss %f" %
                  (trainloss, validationloss))

            # We cycle on epochs
            for epoch in range(self.training_epochs):

                trainloss = self._compute_gradient_step(
                    training_init_op, ntrainbatches)
                validationloss = self._compute_loss(
                    validation_init_op,  nvalidbatches)
                print("Epoch %03d\t train loss %f\t valid loss %f" %
                      (epoch+1, trainloss, validationloss))

            # We compute the test loss
            testloss = self._compute_loss(test_init_op, ntestbatches)
            print("Test loss %f" % (testloss,))

            # Found parameters
            Aval, bval = self.session.run([self.A, self.b])
            print("Estimated A\n", Aval)
            print('Estimated b\n', bval)
        # Here session is closed automatically
```

[3]:
```
class LogisticRegressionV1:

    def __init__(self, dtype, ninputs, noutputs,
                 learning_rate=1e-1, training_epochs=100, batchsize=50):

        self.nc = ninputs
        self.no = noutputs
```

```
        self.dt_shapes = (tf.TensorShape((None, ninputs)),
                          tf.TensorShape((None, noutputs)))
        self.dt_types = (dtype, dtype)

        self.learning_rate = learning_rate
        self.training_epochs = training_epochs
        self.batchsize = batchsize

        self._build_network()

        # Global variable initializer
        self.init_global = tf.initializers.global_variables()

    def _build_network(self):
        # Create ONLY ONE iterator base on types and shapes of one of the␣
↪dataset
        # both dataset should have the same types and shapes...
        self.dataset_iterator = tf.data.Iterator.from_structure(
            self.dt_types, self.dt_shapes)

        # Placeholders from the dataset iterator
        self.x, self.y = self.dataset_iterator.get_next()

        # Logistic regression parameters
        self.A = tf.Variable(tf.zeros([self.nc, self.no]))
        self.b = tf.Variable(tf.zeros([self.no]))
        # All parameters are gathered into var_list
        var_list = [self.A, self.b]

        # Actual logistic regression
        self.logit = tf.matmul(self.x, self.A) + self.b
        self.output = tf.nn.softmax(self.logit)
        self.pred = self.output > .5

        # Model loss
        self.loss = tf.losses.softmax_cross_entropy(self.y, self.logit)

        self.optimizer = tf.train.GradientDescentOptimizer(
            self.learning_rate).minimize(self.loss, var_list=var_list)

    def _prepareset(self, dataset, shuffle=True):
        if shuffle:
            dataset = dataset.shuffle(buffer_size=1000)
        dataset = dataset.batch(self.batchsize)
        return dataset
```

```
    def _compute_loss(self, set_iterator_init, nbatches):
        self.session.run(set_iterator_init)
        lossval = 0.
        for b in range(nbatches):
            batch_lossval, = self.session.run([self.loss])
            lossval += batch_lossval
        lossval /= nbatches
        return lossval

    def _compute_gradient_step(self, set_iterator_init, nbatches):
        self.session.run(set_iterator_init)
        lossval = 0.
        for b in range(nbatches):
            _, batch_lossval, = self.session.run([self.optimizer, self.loss])
            lossval += batch_lossval
        lossval /= nbatches
        return lossval

    def _compute_pred_loss(self, set_iterator_init, nbatches):
        self.session.run(set_iterator_init)
        lossval = 0.
        predval = None
        for b in range(nbatches):
            batch_lossval, batch_pred = self.session.run(
                [self.loss, self.pred])
            lossval += batch_lossval
            if predval is None:
                predval = batch_pred
            else:
                predval = np.concatenate((predval, batch_pred))
        lossval /= nbatches
        return predval, lossval

    def train(self, trainset, validset, testset):

        (Xtrain, Ytrain) = trainset
        (Xvalid, Yvalid) = validset
        (Xtest, Ytest) = testset

        # --- Linear Regression ---

        ntrain, _ = Xtrain.shape
        ntest, _ = Xtest.shape
        nvalid, _ = Xvalid.shape

        trainset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtrain, Ytrain)))
```

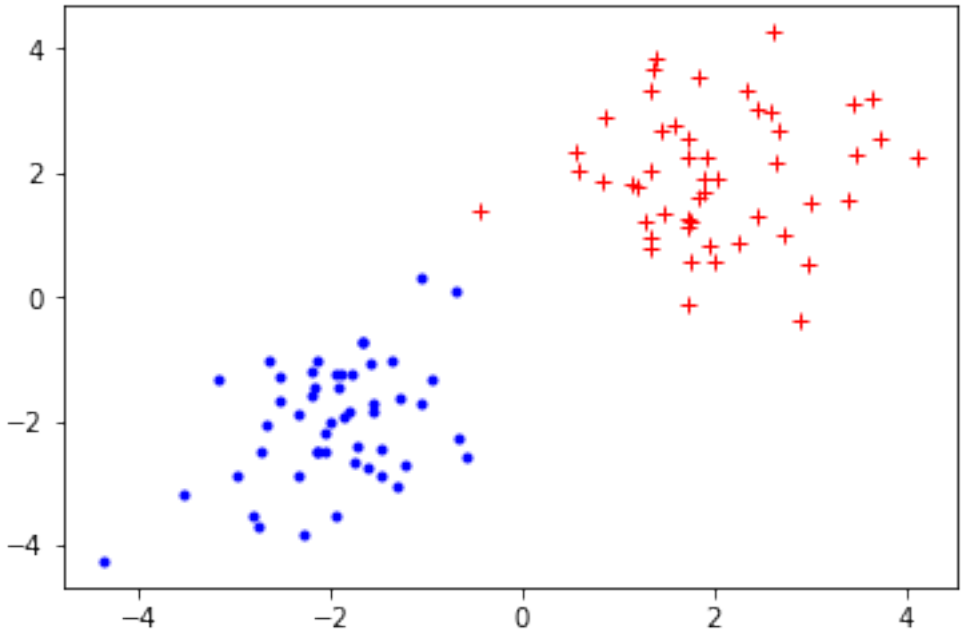
```
        testset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtest, Ytest)), shuffle=False)
        validset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xvalid, Yvalid)), shuffle=False)

        ntrainbatches = int(np.ceil(ntrain/self.batchsize))
        ntestbatches = int(np.ceil(ntest/self.batchsize))
        nvalidbatches = int(np.ceil(nvalid/self.batchsize))

        # Create one initializer per dataset
        training_init_op = self.dataset_iterator.make_initializer(trainset)
        test_init_op = self.dataset_iterator.make_initializer(testset)
        validation_init_op = self.dataset_iterator .make_initializer(validset)

        with tf.Session() as self.session:

            # We call the initialization of A and b
            self.session.run(self.init_global)

            # We compute the train and validation loss
            # Note that you just have to change the feed_dict to change the set

            trainloss = self._compute_loss(training_init_op, ntrainbatches)
            validationloss = self._compute_loss(
                validation_init_op, nvalidbatches)
            print("Init\t\t train loss %f\t valid loss %f" %
                  (trainloss, validationloss))

            # We cycle on epochs
            for epoch in range(self.training_epochs):

                trainloss = self._compute_gradient_step(
                    training_init_op, ntrainbatches)
                validationloss = self._compute_loss(
                    validation_init_op,  nvalidbatches)
                print("Epoch %03d\t train loss %f\t valid loss %f" %
                      (epoch+1, trainloss, validationloss))

            # We compute the prediction and the test loss
            ytestpred, testloss = self._compute_pred_loss(
                test_init_op, ntestbatches)
            print("Test loss %f" % (testloss,))

        # Here session is closed automatically
        return ytestpred
```

```python
def logistic_regression_v1(trainset, validset, testset):
    # reset tensorflow
    tf.reset_default_graph()

    Xtrain, Ytrain = trainset
    _, ninputs = Xtrain.shape
    _, noutputs = Ytrain.shape

    lr = LogisticRegressionV1(tf.float32, ninputs, noutputs)
    Ytestpred = lr.train(trainset, validset, testset)

    Xtest, Ytest = testset

    plot_dataset(Xtest, Ytestpred)


logistic_regression_v1(*generate_all_dataset())
```

```
WARNING:tensorflow:From /home/rherault/.local/venvs/spyder/lib/python3.7/site-
packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /home/rherault/.local/venvs/spyder/lib/python3.7/site-
packages/tensorflow/python/ops/losses/losses_impl.py:209: to_float (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
Init             train loss 0.693147     valid loss 0.693147
Epoch 001        train loss 0.152880     valid loss 0.063275
Epoch 002        train loss 0.044726     valid loss 0.041310
Epoch 003        train loss 0.031092     valid loss 0.032665
Epoch 004        train loss 0.024807     valid loss 0.027859
Epoch 005        train loss 0.021069     valid loss 0.024741
Epoch 006        train loss 0.018553     valid loss 0.022521
Epoch 007        train loss 0.016729     valid loss 0.020846
Epoch 008        train loss 0.015334     valid loss 0.019527
Epoch 009        train loss 0.014223     valid loss 0.018454
Epoch 010        train loss 0.013322     valid loss 0.017561
Epoch 011        train loss 0.012568     valid loss 0.016803
Epoch 012        train loss 0.011927     valid loss 0.016150
Epoch 013        train loss 0.011376     valid loss 0.015579
Epoch 014        train loss 0.010894     valid loss 0.015075
Epoch 015        train loss 0.010469     valid loss 0.014625
Epoch 016        train loss 0.010092     valid loss 0.014220
```

```
Epoch 017       train loss 0.009754     valid loss 0.013854
Epoch 018       train loss 0.009447     valid loss 0.013521
Epoch 019       train loss 0.009171     valid loss 0.013215
Epoch 020       train loss 0.008917     valid loss 0.012934
Epoch 021       train loss 0.008685     valid loss 0.012674
Epoch 022       train loss 0.008469     valid loss 0.012432
Epoch 023       train loss 0.008272     valid loss 0.012207
Epoch 024       train loss 0.008088     valid loss 0.011997
Epoch 025       train loss 0.007917     valid loss 0.011800
Epoch 026       train loss 0.007757     valid loss 0.011615
Epoch 027       train loss 0.007607     valid loss 0.011440
Epoch 028       train loss 0.007466     valid loss 0.011275
Epoch 029       train loss 0.007335     valid loss 0.011119
Epoch 030       train loss 0.007208     valid loss 0.010971
Epoch 031       train loss 0.007093     valid loss 0.010830
Epoch 032       train loss 0.006981     valid loss 0.010696
Epoch 033       train loss 0.006874     valid loss 0.010568
Epoch 034       train loss 0.006773     valid loss 0.010446
Epoch 035       train loss 0.006677     valid loss 0.010329
Epoch 036       train loss 0.006585     valid loss 0.010217
Epoch 037       train loss 0.006499     valid loss 0.010110
Epoch 038       train loss 0.006416     valid loss 0.010007
Epoch 039       train loss 0.006335     valid loss 0.009907
Epoch 040       train loss 0.006259     valid loss 0.009812
Epoch 041       train loss 0.006185     valid loss 0.009720
Epoch 042       train loss 0.006116     valid loss 0.009632
Epoch 043       train loss 0.006047     valid loss 0.009546
Epoch 044       train loss 0.005983     valid loss 0.009463
Epoch 045       train loss 0.005920     valid loss 0.009383
Epoch 046       train loss 0.005860     valid loss 0.009306
Epoch 047       train loss 0.005801     valid loss 0.009231
Epoch 048       train loss 0.005744     valid loss 0.009158
Epoch 049       train loss 0.005689     valid loss 0.009088
Epoch 050       train loss 0.005636     valid loss 0.009020
Epoch 051       train loss 0.005585     valid loss 0.008953
Epoch 052       train loss 0.005539     valid loss 0.008889
Epoch 053       train loss 0.005491     valid loss 0.008826
Epoch 054       train loss 0.005443     valid loss 0.008765
Epoch 055       train loss 0.005399     valid loss 0.008706
Epoch 056       train loss 0.005355     valid loss 0.008648
Epoch 057       train loss 0.005314     valid loss 0.008591
Epoch 058       train loss 0.005270     valid loss 0.008536
Epoch 059       train loss 0.005233     valid loss 0.008483
Epoch 060       train loss 0.005193     valid loss 0.008431
Epoch 061       train loss 0.005155     valid loss 0.008379
Epoch 062       train loss 0.005118     valid loss 0.008330
Epoch 063       train loss 0.005083     valid loss 0.008281
Epoch 064       train loss 0.005047     valid loss 0.008233
```

```
Epoch 065        train loss 0.005013     valid loss 0.008187
Epoch 066        train loss 0.004980     valid loss 0.008141
Epoch 067        train loss 0.004948     valid loss 0.008096
Epoch 068        train loss 0.004916     valid loss 0.008053
Epoch 069        train loss 0.004885     valid loss 0.008010
Epoch 070        train loss 0.004855     valid loss 0.007968
Epoch 071        train loss 0.004825     valid loss 0.007927
Epoch 072        train loss 0.004797     valid loss 0.007887
Epoch 073        train loss 0.004769     valid loss 0.007848
Epoch 074        train loss 0.004739     valid loss 0.007809
Epoch 075        train loss 0.004713     valid loss 0.007771
Epoch 076        train loss 0.004687     valid loss 0.007734
Epoch 077        train loss 0.004662     valid loss 0.007697
Epoch 078        train loss 0.004636     valid loss 0.007661
Epoch 079        train loss 0.004612     valid loss 0.007626
Epoch 080        train loss 0.004587     valid loss 0.007591
Epoch 081        train loss 0.004563     valid loss 0.007558
Epoch 082        train loss 0.004540     valid loss 0.007524
Epoch 083        train loss 0.004517     valid loss 0.007491
Epoch 084        train loss 0.004494     valid loss 0.007459
Epoch 085        train loss 0.004472     valid loss 0.007427
Epoch 086        train loss 0.004451     valid loss 0.007396
Epoch 087        train loss 0.004430     valid loss 0.007365
Epoch 088        train loss 0.004409     valid loss 0.007335
Epoch 089        train loss 0.004387     valid loss 0.007305
Epoch 090        train loss 0.004368     valid loss 0.007276
Epoch 091        train loss 0.004348     valid loss 0.007247
Epoch 092        train loss 0.004329     valid loss 0.007219
Epoch 093        train loss 0.004310     valid loss 0.007191
Epoch 094        train loss 0.004290     valid loss 0.007163
Epoch 095        train loss 0.004272     valid loss 0.007136
Epoch 096        train loss 0.004255     valid loss 0.007109
Epoch 097        train loss 0.004236     valid loss 0.007083
Epoch 098        train loss 0.004218     valid loss 0.007057
Epoch 099        train loss 0.004201     valid loss 0.007031
Epoch 100        train loss 0.004184     valid loss 0.007006
Test loss 0.004051
```



1.2 Metrics

In the precedent script, each time you compute the loss of batch, the result is copied from the memory of the computation engine back to the computer memory. Moreover, it is actually the computer which is doing the averaging.

You can faster the loss loop/computation by using `metrics`: that's a trick to do all the work inside the memory of the computation engine.

A metric contain an internal state that will be update for each batch. When all the examples have been seen , the last internal state is used to compute the actual metrics.

For example, a mean metrics consists in a `total` and `count` internal states. At each update, the targeted tensor (for example the loss) is added to `total` and `count` is incremented by one. At the end, `total` is divided by `count` to give the actual mean metric.

1.2.1 How to use a metric ?

1) Create a metric on targeted tensor(s) (like the loss), you will be provide by two tensors in return `actualmetric`, `metricupdate`
2) Initialize the internal state of the metric
3) Update the internal state of the metric running `metricupdate` for each batch inside a loop.
4) Get the metric value by running `actualmetric`.

1.2.2 Example

Here is how you can use `tf.metrics.mean` for averaging the loss over all the batches to compute a loss

```
class LogisticRegressionV2(LogisticRegressionV1):

    def __init__(self,*args,**kwargs):
        super().__init__(*args,**kwargs)

        # Initializers of metrics,
        # should be instantiate after the network is built
        self.init_local = tf.initializers.local_variables()

    def _build_network(self):
        super()._build_network()

        # Metrics
        self.lossmetric, self.lossmetric_update = tf.metrics.mean(self.loss)

    def _compute_loss(self, set_iterator_init, nbatches):
        # Initialize all the metrics
        self.session.run(self.init_local)
        # Initialize the iterator
        self.session.run(set_iterator_init)
        # Loop
        for b in range(nbatches):
            self.session.run(self.lossmetric_update)
        lossval = self.session.run(self.lossmetric)
        return lossval
```

1.2.3 Exercise

Complete the class `LogisticRegressionV2` by methods `_compute_gradient_step` and `_compute_pred_loss` that compute the loss through a metric.

[4]:
```
class LogisticRegressionV2(LogisticRegressionV1):

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        # Initializers of metrics,
        # should be instantiate after the network is built
        self.init_local = tf.initializers.local_variables()

    def _build_network(self):
        super()._build_network()

        # Metrics
```

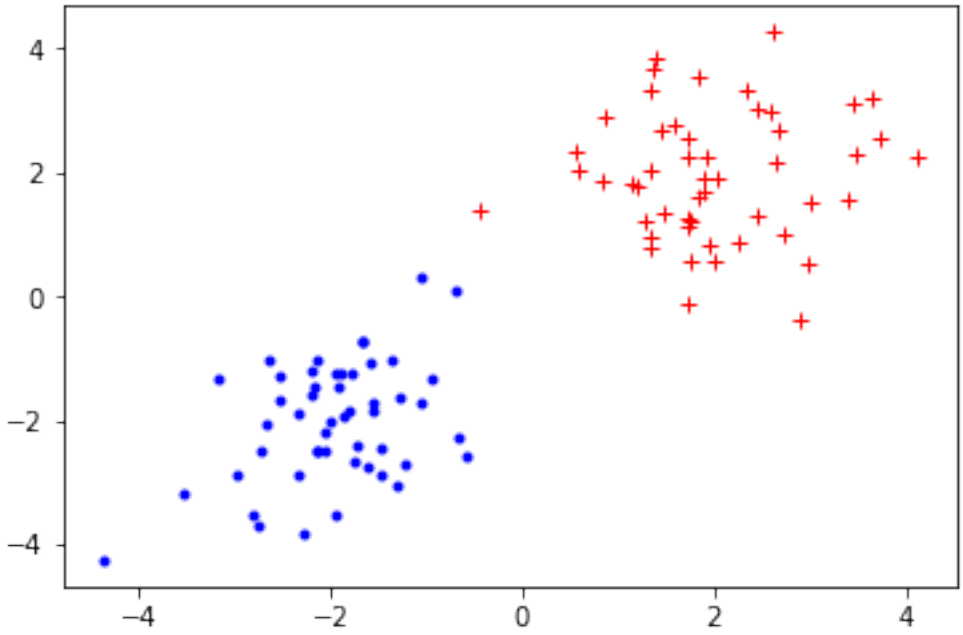
```
        self.lossmetric, self.lossmetric_update = tf.metrics.mean(self.loss)

    def _compute_loss(self, set_iterator_init, nbatches):
        # Initialize all the metrics
        self.session.run(self.init_local)
        # Initialize the iterator
        self.session.run(set_iterator_init)
        # Loop
        for b in range(nbatches):
            self.session.run(self.lossmetric_update)
        lossval = self.session.run(self.lossmetric)
        return lossval

    def _compute_gradient_step(self, set_iterator_init, nbatches):
        self.session.run(self.init_local)
        self.session.run(set_iterator_init)
        for b in range(nbatches):
            self.session.run([self.optimizer, self.lossmetric_update])
        lossval = self.session.run(self.lossmetric)
        return lossval

    def _compute_pred_loss(self, set_iterator_init, nbatches):
        self.session.run(self.init_local)
        self.session.run(set_iterator_init)
        predval = None
        for b in range(nbatches):
            batch_pred, _ = self.session.run(
                [self.pred, self.lossmetric_update])
            if predval is None:
                predval = batch_pred
            else:
                predval = np.concatenate((predval, batch_pred))
        lossval = self.session.run(self.lossmetric)
        return predval, lossval


def logistic_regression_v2(trainset, validset, testset):
    # reset tensorflow
    tf.reset_default_graph()

    Xtrain, Ytrain = trainset
    _, ninputs = Xtrain.shape
    _, noutputs = Ytrain.shape

    lr = LogisticRegressionV2(tf.float32, ninputs, noutputs)
    Ytestpred = lr.train(trainset, validset, testset)
```

```
    Xtest, Ytest = testset

    plot_dataset(Xtest, Ytestpred)


logistic_regression_v2(*generate_all_dataset())
```

```
Init            train loss 0.693147     valid loss 0.693147
Epoch 001       train loss 0.151678     valid loss 0.062934
Epoch 002       train loss 0.044502     valid loss 0.041140
Epoch 003       train loss 0.030996     valid loss 0.032590
Epoch 004       train loss 0.024763     valid loss 0.027824
Epoch 005       train loss 0.021045     valid loss 0.024717
Epoch 006       train loss 0.018544     valid loss 0.022507
Epoch 007       train loss 0.016725     valid loss 0.020837
Epoch 008       train loss 0.015331     valid loss 0.019521
Epoch 009       train loss 0.014226     valid loss 0.018450
Epoch 010       train loss 0.013324     valid loss 0.017558
Epoch 011       train loss 0.012568     valid loss 0.016801
Epoch 012       train loss 0.011929     valid loss 0.016148
Epoch 013       train loss 0.011383     valid loss 0.015577
Epoch 014       train loss 0.010897     valid loss 0.015073
Epoch 015       train loss 0.010470     valid loss 0.014623
Epoch 016       train loss 0.010094     valid loss 0.014219
Epoch 017       train loss 0.009756     valid loss 0.013853
Epoch 018       train loss 0.009449     valid loss 0.013520
Epoch 019       train loss 0.009170     valid loss 0.013214
Epoch 020       train loss 0.008920     valid loss 0.012933
Epoch 021       train loss 0.008686     valid loss 0.012673
Epoch 022       train loss 0.008470     valid loss 0.012431
Epoch 023       train loss 0.008273     valid loss 0.012207
Epoch 024       train loss 0.008090     valid loss 0.011996
Epoch 025       train loss 0.007917     valid loss 0.011799
Epoch 026       train loss 0.007757     valid loss 0.011614
Epoch 027       train loss 0.007607     valid loss 0.011440
Epoch 028       train loss 0.007466     valid loss 0.011275
Epoch 029       train loss 0.007335     valid loss 0.011118
Epoch 030       train loss 0.007210     valid loss 0.010970
Epoch 031       train loss 0.007092     valid loss 0.010830
Epoch 032       train loss 0.006981     valid loss 0.010695
Epoch 033       train loss 0.006874     valid loss 0.010568
Epoch 034       train loss 0.006775     valid loss 0.010445
Epoch 035       train loss 0.006679     valid loss 0.010329
Epoch 036       train loss 0.006587     valid loss 0.010217
Epoch 037       train loss 0.006498     valid loss 0.010109
Epoch 038       train loss 0.006416     valid loss 0.010006
Epoch 039       train loss 0.006336     valid loss 0.009907
```

```
Epoch 040       train loss 0.006260     valid loss 0.009812
Epoch 041       train loss 0.006186     valid loss 0.009720
Epoch 042       train loss 0.006115     valid loss 0.009631
Epoch 043       train loss 0.006048     valid loss 0.009546
Epoch 044       train loss 0.005983     valid loss 0.009463
Epoch 045       train loss 0.005920     valid loss 0.009383
Epoch 046       train loss 0.005859     valid loss 0.009306
Epoch 047       train loss 0.005801     valid loss 0.009231
Epoch 048       train loss 0.005746     valid loss 0.009158
Epoch 049       train loss 0.005691     valid loss 0.009088
Epoch 050       train loss 0.005638     valid loss 0.009019
Epoch 051       train loss 0.005587     valid loss 0.008953
Epoch 052       train loss 0.005538     valid loss 0.008889
Epoch 053       train loss 0.005490     valid loss 0.008826
Epoch 054       train loss 0.005443     valid loss 0.008765
Epoch 055       train loss 0.005399     valid loss 0.008705
Epoch 056       train loss 0.005355     valid loss 0.008648
Epoch 057       train loss 0.005314     valid loss 0.008591
Epoch 058       train loss 0.005271     valid loss 0.008536
Epoch 059       train loss 0.005231     valid loss 0.008483
Epoch 060       train loss 0.005193     valid loss 0.008430
Epoch 061       train loss 0.005155     valid loss 0.008379
Epoch 062       train loss 0.005118     valid loss 0.008329
Epoch 063       train loss 0.005083     valid loss 0.008281
Epoch 064       train loss 0.005047     valid loss 0.008233
Epoch 065       train loss 0.005013     valid loss 0.008186
Epoch 066       train loss 0.004981     valid loss 0.008141
Epoch 067       train loss 0.004948     valid loss 0.008096
Epoch 068       train loss 0.004916     valid loss 0.008053
Epoch 069       train loss 0.004885     valid loss 0.008010
Epoch 070       train loss 0.004855     valid loss 0.007968
Epoch 071       train loss 0.004825     valid loss 0.007927
Epoch 072       train loss 0.004797     valid loss 0.007887
Epoch 073       train loss 0.004768     valid loss 0.007847
Epoch 074       train loss 0.004741     valid loss 0.007809
Epoch 075       train loss 0.004714     valid loss 0.007771
Epoch 076       train loss 0.004688     valid loss 0.007734
Epoch 077       train loss 0.004661     valid loss 0.007697
Epoch 078       train loss 0.004636     valid loss 0.007661
Epoch 079       train loss 0.004610     valid loss 0.007626
Epoch 080       train loss 0.004588     valid loss 0.007592
Epoch 081       train loss 0.004562     valid loss 0.007558
Epoch 082       train loss 0.004540     valid loss 0.007524
Epoch 083       train loss 0.004518     valid loss 0.007491
Epoch 084       train loss 0.004495     valid loss 0.007459
Epoch 085       train loss 0.004472     valid loss 0.007427
Epoch 086       train loss 0.004450     valid loss 0.007396
Epoch 087       train loss 0.004429     valid loss 0.007365
```

```
Epoch 088        train loss 0.004408     valid loss 0.007335
Epoch 089        train loss 0.004388     valid loss 0.007305
Epoch 090        train loss 0.004368     valid loss 0.007276
Epoch 091        train loss 0.004348     valid loss 0.007247
Epoch 092        train loss 0.004329     valid loss 0.007219
Epoch 093        train loss 0.004310     valid loss 0.007191
Epoch 094        train loss 0.004291     valid loss 0.007163
Epoch 095        train loss 0.004272     valid loss 0.007136
Epoch 096        train loss 0.004255     valid loss 0.007109
Epoch 097        train loss 0.004237     valid loss 0.007083
Epoch 098        train loss 0.004219     valid loss 0.007057
Epoch 099        train loss 0.004202     valid loss 0.007031
Epoch 100        train loss 0.004185     valid loss 0.007006
Test loss 0.004050
```



1.2.4 Tracking the accuracy

Metrics are also useful to track indicators other than the loss. For example in our classification task we can look at the accuracy of the model, i.e. the matches between prediction and labels.

Here is a code for computing the loss and the accuracy at the same time:

```
class LogisticRegressionV3(LogisticRegressionV2):

    def _build_network(self):
        super()._build_network()
```

```
        self.accuracymetric,  self.accuracymetric_update = tf.metrics.accuracy(
            self.y, self.pred)

    def _compute_loss(self, set_iterator_init, nbatches):
        # Initialize all the metrics
        self.session.run(self.init_local)
        # Initialize the iterator
        self.session.run(set_iterator_init)
        # Loop
        for b in range(nbatches):
            self.session.run(
                [self.lossmetric_update, self.accuracymetric_update])
        return self.session.run([self.lossmetric, self.accuracymetric])
```

Please refer to this page to see other possible metrics: https://www.tensorflow.org/api_docs/python/tf/metrics

1.2.5 Excercice

Complete the class `LogisticRegressionV3` in order to have a method `_compute_pred_loss` that compute prediction, loss and accuracy, and a `train` method that displays the accuracy. Use a low learning rate (1e-10) if you want to see the evolution of accuracy as it is an easy dataset.

[5]:
```
class LogisticRegressionV3(LogisticRegressionV2):

    def _build_network(self):
        super()._build_network()

        self.accuracymetric,  self.accuracymetric_update = tf.metrics.accuracy(
            self.y, self.pred)

    def _compute_loss(self, set_iterator_init, nbatches):
        # Initialize all the metrics
        self.session.run(self.init_local)
        # Initialize the iterator
        self.session.run(set_iterator_init)
        # Loop
        for b in range(nbatches):
            self.session.run(
                [self.lossmetric_update, self.accuracymetric_update])
        return self.session.run([self.lossmetric, self.accuracymetric])

    def _compute_pred_loss(self, set_iterator_init, nbatches):
        self.session.run(self.init_local)
        self.session.run(set_iterator_init)
        predval = None
        for b in range(nbatches):
            batch_pred, _, _ = self.session.run(
                [self.pred, self.lossmetric_update, self.accuracymetric_update])
```

```
            if predval is None:
                predval = batch_pred
            else:
                predval = np.concatenate((predval, batch_pred))
        lossval, accval = self.session.run(
            [self.lossmetric, self.accuracymetric])
        return predval, lossval, accval

    def train(self, trainset, validset, testset):

        (Xtrain, Ytrain) = trainset
        (Xvalid, Yvalid) = validset
        (Xtest, Ytest) = testset

        # --- Linear Regression ---

        ntrain, _ = Xtrain.shape
        ntest, _ = Xtest.shape
        nvalid, _ = Xvalid.shape

        trainset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtrain, Ytrain)))
        testset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtest, Ytest)), shuffle=False)
        validset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xvalid, Yvalid)), shuffle=False)

        ntrainbatches = int(np.ceil(ntrain/self.batchsize))
        ntestbatches = int(np.ceil(ntest/self.batchsize))
        nvalidbatches = int(np.ceil(nvalid/self.batchsize))

        # Create one initializer per dataset
        training_init_op = self.dataset_iterator.make_initializer(trainset)
        test_init_op = self.dataset_iterator.make_initializer(testset)
        validation_init_op = self.dataset_iterator .make_initializer(validset)

        with tf.Session() as self.session:

            # We call the global initialization
            self.session.run(self.init_global)

            # We compute the train and validation loss
            # Note that you just have to change the feed_dict to change the set

            trainloss, trainacc = self._compute_loss(
                training_init_op, ntrainbatches)
            validationloss, validacc = self._compute_loss(
```

```
                validation_init_op, nvalidbatches)
            print("Init\t\t train loss %f\t valid loss %f\t valid acc %f" %
                  (trainloss, validationloss, validacc))

            # We cycle on epochs
            for epoch in range(self.training_epochs):

                trainloss = self._compute_gradient_step(
                    training_init_op, ntrainbatches)
                validationloss, validacc = self._compute_loss(
                    validation_init_op,  nvalidbatches)
                print("Epoch %03d\t train loss %f\t valid loss %f\t valid acc␣
↪%f" %
                      (epoch+1, trainloss, validationloss, validacc))

            # We compute the prediction and the test loss
            ytestpred, testloss, testacc = self._compute_pred_loss(
                test_init_op, ntestbatches)
            print("Test loss %f\t test acc %f" % (testloss, testacc))

        # Here session is closed automatically
        return ytestpred


def logistic_regression_v3(trainset, validset, testset):
    # reset tensorflow
    tf.reset_default_graph()

    Xtrain, Ytrain = trainset
    _, ninputs = Xtrain.shape
    _, noutputs = Ytrain.shape

    lr = LogisticRegressionV3(
        tf.float32, ninputs, noutputs, learning_rate=1e-10)
    Ytestpred = lr.train(trainset, validset, testset)

    Xtest, Ytest = testset

    plot_dataset(Xtest, Ytestpred)


logistic_regression_v3(*generate_all_dataset())
```

```
Init             train loss 0.693147     valid loss 0.693147     valid acc
0.500000
Epoch 001        train loss 0.693147     valid loss 0.693147     valid acc
0.500000
```

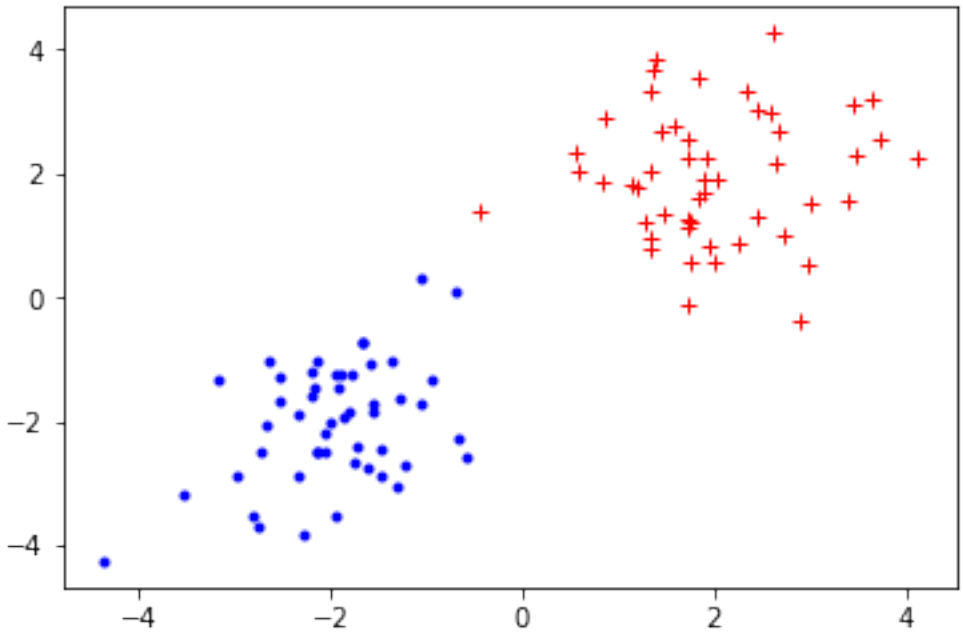
```
Epoch 002       train loss 0.693147     valid loss 0.693147     valid acc
0.500000
Epoch 003       train loss 0.693147     valid loss 0.693147     valid acc
0.500000
Epoch 004       train loss 0.693147     valid loss 0.693147     valid acc
0.540000
Epoch 005       train loss 0.693147     valid loss 0.693147     valid acc
0.670000
Epoch 006       train loss 0.693147     valid loss 0.693147     valid acc
0.775000
Epoch 007       train loss 0.693147     valid loss 0.693147     valid acc
0.840000
Epoch 008       train loss 0.693147     valid loss 0.693147     valid acc
0.895000
Epoch 009       train loss 0.693147     valid loss 0.693147     valid acc
0.930000
Epoch 010       train loss 0.693147     valid loss 0.693147     valid acc
0.940000
Epoch 011       train loss 0.693147     valid loss 0.693147     valid acc
0.950000
Epoch 012       train loss 0.693147     valid loss 0.693147     valid acc
0.960000
Epoch 013       train loss 0.693147     valid loss 0.693147     valid acc
0.975000
Epoch 014       train loss 0.693147     valid loss 0.693147     valid acc
0.980000
Epoch 015       train loss 0.693147     valid loss 0.693147     valid acc
0.980000
Epoch 016       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 017       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 018       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 019       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 020       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 021       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 022       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 023       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 024       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 025       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
```

```
Epoch 026       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 027       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 028       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 029       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 030       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 031       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 032       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 033       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 034       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 035       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 036       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 037       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 038       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 039       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 040       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 041       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 042       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 043       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 044       train loss 0.693147     valid loss 0.693147     valid acc
0.995000
Epoch 045       train loss 0.693147     valid loss 0.693147     valid acc
0.995000
Epoch 046       train loss 0.693147     valid loss 0.693147     valid acc
0.995000
Epoch 047       train loss 0.693147     valid loss 0.693147     valid acc
0.995000
Epoch 048       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 049       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
```

```
Epoch 050       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 051       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 052       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 053       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 054       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 055       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 056       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 057       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 058       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 059       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 060       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 061       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 062       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 063       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 064       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 065       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 066       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 067       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 068       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 069       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 070       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 071       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 072       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 073       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
```

```
Epoch 074       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 075       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 076       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 077       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 078       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 079       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 080       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 081       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 082       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 083       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 084       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 085       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 086       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 087       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 088       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 089       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 090       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 091       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 092       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 093       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 094       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 095       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 096       train loss 0.693146     valid loss 0.693146     valid acc
1.000000
Epoch 097       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
```

```
Epoch 098        train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 099        train loss 0.693146     valid loss 0.693146     valid acc
1.000000
Epoch 100        train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Test loss 0.693146       test acc 1.000000
```



1.3 Tensorboard

Tensorboard is an easy way to monitor the training of your model. It is a dashboard in a web browser displaying metrics and possibly parameters of your model.

In order to do so you first need to export your metrics to log files. Tensorboard will then explore the log files and show you the different graphs in your browser.

1.3.1 Summaries

Summaries are object that can be evaluated and written to tensorboard log files. You need to wrap your metrics around summaries.

After defining a metric just do:

```
trainlosssummary = tf.summary.scalar("train_loss",lossmetric)
```

You can merge multiple summaries into one :

```
trainlosssummary = tf.summary.scalar("train_loss",lossmetric)
trainaccuracysummary = tf.summary.scalar("train_accuracy",accuracymetric)
trainsummaries = tf.summary.merge([trainlosssummary,trainaccuracysummary ])
```

Now `trainsummaries` contains all the summaries of the trainset.

If you have multiple set (e.g. train and valid), you should create one summary per set, even if they are based on the same metric:

```
trainlosssummary = tf.summary.scalar("train_loss",lossmetric)
trainaccuracysummary = tf.summary.scalar("train_accuracy",accuracymetric)
trainsummaries = tf.summary.merge([trainlosssummary,trainaccuracysummary ])

validationlosssummary = tf.summary.scalar("validation_loss",lossmetric)
validationaccuracysummary = tf.summary.scalar("validation_accuracy",accuracymetric)
validationsummaries = tf.summary.merge([validationlosssummary,validationaccuracysummary ])
```

1.3.2 Writer

A Writer is an object that represent a recorder to a folder.

To instanciate a recorder do inside a session:

```
with tf.Session() as session:
    #[...]
    logdir = "logs/somename/" + dt.datetime.now().strftime("%Y%m%d-%H%M%S")
    writer = tf.summary.FileWriter(logdir)
    #[...]
```

Now at each epoch, you can evaluate the summaries and add them to the recorder:

```
with tf.Session() as session:

    # We call the global initialization
    session.run(init_global)
    # Create the log writer
    logdir="logs/logisticregression/" + dt.datetime.now().strftime("%Y%m%d-%H%M%S")
    writer = tf.summary.FileWriter(logdir)
    #[...]
    for epoch in range(training_epochs):
        # Work on the training set
        #[...]
        # And then
        trainsummariesval = session.run(trainsummaries) # evaluate the summary
        writer.add_summary(trainsummariesval, epoch+1) # record the summary
        # Work on the validation set
        #[...]
        # And then
        validationsummariesval = session.run(validationsummaries) # evaluate the summary
        writer.add_summary(validationsummariesval,epoch+1) # record the summary
```

1.3.3 Log analyzing

A `logisticregressionlogs` folder will be create where the script is run. It contains log file describing the evolution of the training.

To analyze them, we can actually launch tensorboard with the command inside a bash terminal:

```
tensorboard --logdir=logisticregressionlogs
```

It will automatically launch a web browser where you can monitor the metrics of your model.

Caution ! as summaries and log files cumulate it is highly recommended to restart the python kernel and to erase the log folder before each execution of the script.

1.3.4 Your turn !

1) Create a `LogisticRegressionV4` class derivated from `LogisticRegressionV3`
2) Modify `_build_networkand` `train` methods to support file logging.
3) Analyze the log in Tensorboard
4) Create a `LogisticRegression` class (no subclassing) merging all the logistic regression versions.

[6]:
```
class LogisticRegressionV4(LogisticRegressionV3):

    def _build_network(self):
        super()._build_network()

        trainlosssummary = tf.summary.scalar("train_loss", self.lossmetric)
        trainaccuracysummary = tf.summary.scalar(
            "train_accuracy", self.accuracymetric)
        self.trainsummaries = tf.summary.merge(
            [trainlosssummary, trainaccuracysummary])

        validationlosssummary = tf.summary.scalar(
            "validation_loss", self.lossmetric)
        validationaccuracysummary = tf.summary.scalar(
            "validation_accuracy", self.accuracymetric)
        self.validationsummaries = tf.summary.merge(
            [validationlosssummary, validationaccuracysummary])

    def train(self, trainset, validset, testset):

        (Xtrain, Ytrain) = trainset
        (Xvalid, Yvalid) = validset
        (Xtest, Ytest) = testset

        # --- Linear Regression ---
```

```
        ntrain, _ = Xtrain.shape
        ntest, _ = Xtest.shape
        nvalid, _ = Xvalid.shape

        trainset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtrain, Ytrain)))
        testset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtest, Ytest)), shuffle=False)
        validset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xvalid, Yvalid)), shuffle=False)

        ntrainbatches = int(np.ceil(ntrain/self.batchsize))
        ntestbatches = int(np.ceil(ntest/self.batchsize))
        nvalidbatches = int(np.ceil(nvalid/self.batchsize))

        # Create one initializer per dataset
        training_init_op = self.dataset_iterator.make_initializer(trainset)
        test_init_op = self.dataset_iterator.make_initializer(testset)
        validation_init_op = self.dataset_iterator .make_initializer(validset)

        with tf.Session() as self.session:

            # We call the global initialization
            self.session.run(self.init_global)

            # Create the log writer
            logdir = "logs/logisticregression/" + dt.datetime.now().
↪strftime("%Y%m%d-%H%M%S")
            writer = tf.summary.FileWriter(logdir)

            # We compute the train and validation loss
            # Note that you just have to change the feed_dict to change the set

            trainloss, trainacc = self._compute_loss(
                training_init_op, ntrainbatches)
            validationloss, validacc = self._compute_loss(
                validation_init_op, nvalidbatches)

            trainsummariesval = self.session.run(self.trainsummaries)
            writer.add_summary(trainsummariesval, 0)
            validationsummariesval = self.session.run(self.validationsummaries)
            writer.add_summary(validationsummariesval, 0)

            print("Init\t\t train loss %f\t valid loss %f\t valid acc %f" %
                  (trainloss, validationloss, validacc))

            # We cycle on epochs
```

```
            for epoch in range(self.training_epochs):

                trainloss = self._compute_gradient_step(
                    training_init_op, ntrainbatches)
                validationloss, validacc = self._compute_loss(
                    validation_init_op,  nvalidbatches)

                trainsummariesval = self.session.run(self.trainsummaries)
                writer.add_summary(trainsummariesval, epoch+1)
                validationsummariesval = self.session.run(
                    self.validationsummaries)
                writer.add_summary(validationsummariesval, epoch+1)

                print("Epoch %03d\t train loss %f\t valid loss %f\t valid acc␣
 ↪%f" %
                      (epoch+1, trainloss, validationloss, validacc))

            # We compute the prediction and the test loss
            ytestpred, testloss, testacc = self._compute_pred_loss(
                test_init_op, ntestbatches)
            print("Test loss %f\t test acc %f" % (testloss, testacc))

        # Here session is closed automatically
        return ytestpred


def logistic_regression_v4(trainset, validset, testset):
    # reset tensorflow
    tf.reset_default_graph()

    Xtrain, Ytrain = trainset
    _, ninputs = Xtrain.shape
    _, noutputs = Ytrain.shape

    lr = LogisticRegressionV4(
        tf.float32, ninputs, noutputs, learning_rate=1e-10)
    Ytestpred = lr.train(trainset, validset, testset)

    Xtest, Ytest = testset

    plot_dataset(Xtest, Ytestpred)


logistic_regression_v4(*generate_all_dataset())
```

```
Init             train loss 0.693147     valid loss 0.693147     valid acc
0.500000
```

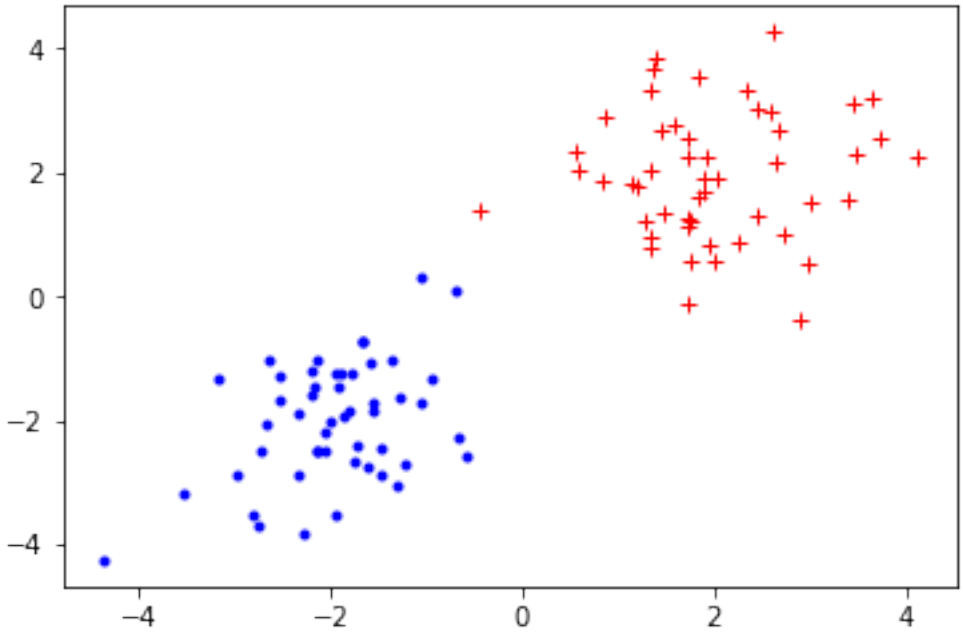
```
Epoch 001       train loss 0.693147     valid loss 0.693147     valid acc
0.500000
Epoch 002       train loss 0.693147     valid loss 0.693147     valid acc
0.500000
Epoch 003       train loss 0.693147     valid loss 0.693147     valid acc
0.500000
Epoch 004       train loss 0.693147     valid loss 0.693147     valid acc
0.540000
Epoch 005       train loss 0.693147     valid loss 0.693147     valid acc
0.670000
Epoch 006       train loss 0.693147     valid loss 0.693147     valid acc
0.775000
Epoch 007       train loss 0.693147     valid loss 0.693147     valid acc
0.840000
Epoch 008       train loss 0.693147     valid loss 0.693147     valid acc
0.895000
Epoch 009       train loss 0.693147     valid loss 0.693147     valid acc
0.930000
Epoch 010       train loss 0.693147     valid loss 0.693147     valid acc
0.940000
Epoch 011       train loss 0.693147     valid loss 0.693147     valid acc
0.950000
Epoch 012       train loss 0.693147     valid loss 0.693147     valid acc
0.960000
Epoch 013       train loss 0.693147     valid loss 0.693147     valid acc
0.975000
Epoch 014       train loss 0.693147     valid loss 0.693147     valid acc
0.980000
Epoch 015       train loss 0.693147     valid loss 0.693147     valid acc
0.980000
Epoch 016       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 017       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 018       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 019       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 020       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 021       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 022       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 023       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 024       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
```

```
Epoch 025       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 026       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 027       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 028       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 029       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 030       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 031       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 032       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 033       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 034       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 035       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 036       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 037       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 038       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 039       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 040       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 041       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 042       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 043       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 044       train loss 0.693147     valid loss 0.693147     valid acc
0.995000
Epoch 045       train loss 0.693147     valid loss 0.693147     valid acc
0.995000
Epoch 046       train loss 0.693147     valid loss 0.693147     valid acc
0.995000
Epoch 047       train loss 0.693147     valid loss 0.693147     valid acc
0.995000
Epoch 048       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
```

```
Epoch 049       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 050       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 051       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 052       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 053       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 054       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 055       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 056       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 057       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 058       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 059       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 060       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 061       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 062       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 063       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 064       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 065       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 066       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 067       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 068       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 069       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 070       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 071       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 072       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
```

```
Epoch 073       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 074       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 075       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 076       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 077       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 078       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 079       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 080       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 081       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 082       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 083       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 084       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 085       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 086       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 087       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 088       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 089       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 090       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 091       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 092       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 093       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 094       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 095       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 096       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
```

```
Epoch 097        train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 098        train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 099        train loss 0.693146     valid loss 0.693146     valid acc
1.000000
Epoch 100        train loss 0.693146     valid loss 0.693146     valid acc
1.000000
Test loss 0.693146       test acc 1.000000
```



[7]:
```python
## All in One
class LogisticRegression:

    def __init__(self, dtype, ninputs, noutputs,
                 learning_rate=1e-1, training_epochs=100, batchsize=50):

        self.nc = ninputs
        self.no = noutputs

        self.dt_shapes = (tf.TensorShape((None, ninputs)),
                          tf.TensorShape((None, noutputs)))
        self.dt_types = (dtype, dtype)

        self.learning_rate = learning_rate
        self.training_epochs = training_epochs
        self.batchsize = batchsize
```

```
        self._build_network()

        # local and global variable initializers
        self.init_global = tf.initializers.global_variables()
        self.init_local = tf.initializers.local_variables()

    def _build_network(self):
        # Create ONLY ONE iterator base on types and shapes of one of the␣
↪dataset
        # both dataset should have the same types and shapes...
        self.dataset_iterator = tf.data.Iterator.from_structure(
            self.dt_types, self.dt_shapes)

        # Placeholders from the dataset iterator
        self.x, self.y = self.dataset_iterator.get_next()

        # Logistic regression parameters
        self.A = tf.Variable(tf.zeros([self.nc, self.no]))
        self.b = tf.Variable(tf.zeros([self.no]))
        # All parameters are gathered into var_list
        var_list = [self.A, self.b]

        # Actual logistic regression
        self.logit = tf.matmul(self.x, self.A) + self.b
        self.output = tf.nn.softmax(self.logit)
        self.pred = self.output > .5

        # Model loss
        self.loss = tf.losses.softmax_cross_entropy(self.y, self.logit)

        self.optimizer = tf.train.GradientDescentOptimizer(
            self.learning_rate).minimize(self.loss, var_list=var_list)

        # Metrics
        self.lossmetric, self.lossmetric_update = tf.metrics.mean(self.loss)
        self.accuracymetric,  self.accuracymetric_update = tf.metrics.accuracy(
            self.y, self.pred)

        # Summaries
        trainlosssummary = tf.summary.scalar("train_loss", self.lossmetric)
        trainaccuracysummary = tf.summary.scalar(
            "train_accuracy", self.accuracymetric)
        self.trainsummaries = tf.summary.merge(
            [trainlosssummary, trainaccuracysummary])

        validationlosssummary = tf.summary.scalar(
```

```
            "validation_loss", self.lossmetric)
        validationaccuracysummary = tf.summary.scalar(
            "validation_accuracy", self.accuracymetric)
        self.validationsummaries = tf.summary.merge(
            [validationlosssummary, validationaccuracysummary])

    def _prepareset(self, dataset, shuffle=True):
        if shuffle:
            dataset = dataset.shuffle(buffer_size=1000)
        dataset = dataset.batch(self.batchsize)
        return dataset

    def _compute_loss(self, set_iterator_init, nbatches):
        # Initialize all the metrics
        self.session.run(self.init_local)
        # Initialize the iterator
        self.session.run(set_iterator_init)
        # Loop
        for b in range(nbatches):
            self.session.run(
                [self.lossmetric_update, self.accuracymetric_update])
        return self.session.run([self.lossmetric, self.accuracymetric])

    def _compute_gradient_step(self, set_iterator_init, nbatches):
        self.session.run(self.init_local)
        self.session.run(set_iterator_init)
        for b in range(nbatches):
            self.session.run([self.optimizer, self.lossmetric_update])
        lossval = self.session.run(self.lossmetric)
        return lossval

    def _compute_pred_loss(self, set_iterator_init, nbatches):
        self.session.run(self.init_local)
        self.session.run(set_iterator_init)
        predval = None
        for b in range(nbatches):
            batch_pred, _, _ = self.session.run(
                [self.pred, self.lossmetric_update, self.accuracymetric_update])
            if predval is None:
                predval = batch_pred
            else:
                predval = np.concatenate((predval, batch_pred))
        lossval, accval = self.session.run(
            [self.lossmetric, self.accuracymetric])
        return predval, lossval, accval

    def train(self, trainset, validset, testset):
```

```
        (Xtrain, Ytrain) = trainset
        (Xvalid, Yvalid) = validset
        (Xtest, Ytest) = testset

        # --- Linear Regression ---

        ntrain, _ = Xtrain.shape
        ntest, _ = Xtest.shape
        nvalid, _ = Xvalid.shape

        trainset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtrain, Ytrain)))
        testset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtest, Ytest)), shuffle=False)
        validset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xvalid, Yvalid)), shuffle=False)

        ntrainbatches = int(np.ceil(ntrain/self.batchsize))
        ntestbatches = int(np.ceil(ntest/self.batchsize))
        nvalidbatches = int(np.ceil(nvalid/self.batchsize))

        # Create one initializer per dataset
        training_init_op = self.dataset_iterator.make_initializer(trainset)
        test_init_op = self.dataset_iterator.make_initializer(testset)
        validation_init_op = self.dataset_iterator .make_initializer(validset)

        with tf.Session() as self.session:

            # We call the global initialization
            self.session.run(self.init_global)

            # Create the log writer
            logdir = "logs/logisticregression/" + dt.datetime.now().
↪strftime("%Y%m%d-%H%M%S")
            writer = tf.summary.FileWriter(logdir)

            # We compute the train and validation loss
            # Note that you just have to change the feed_dict to change the set

            trainloss, trainacc = self._compute_loss(
                training_init_op, ntrainbatches)
            validationloss, validacc = self._compute_loss(
                validation_init_op, nvalidbatches)

            trainsummariesval = self.session.run(self.trainsummaries)
            writer.add_summary(trainsummariesval, 0)
```

```
            validationsummariesval = self.session.run(self.validationsummaries)
            writer.add_summary(validationsummariesval, 0)

            print("Init\t\t train loss %f\t valid loss %f\t valid acc %f" %
                  (trainloss, validationloss, validacc))

            # We cycle on epochs
            for epoch in range(self.training_epochs):

                trainloss = self._compute_gradient_step(
                    training_init_op, ntrainbatches)
                validationloss, validacc = self._compute_loss(
                    validation_init_op,  nvalidbatches)

                trainsummariesval = self.session.run(self.trainsummaries)
                writer.add_summary(trainsummariesval, epoch+1)
                validationsummariesval = self.session.run(
                    self.validationsummaries)
                writer.add_summary(validationsummariesval, epoch+1)

                print("Epoch %03d\t train loss %f\t valid loss %f\t valid acc␣
↪%f" %
                      (epoch+1, trainloss, validationloss, validacc))

            # We compute the prediction and the test loss
            ytestpred, testloss, testacc = self._compute_pred_loss(
                test_init_op, ntestbatches)
            print("Test loss %f\t test acc %f" % (testloss, testacc))

        # Here session is closed automatically
        return ytestpred


def logistic_regression(trainset, validset, testset):
    # reset tensorflow
    tf.reset_default_graph()

    Xtrain, Ytrain = trainset
    _, ninputs = Xtrain.shape
    _, noutputs = Ytrain.shape

    lr = LogisticRegression(tf.float32, ninputs, noutputs, learning_rate=1e-10)
    Ytestpred = lr.train(trainset, validset, testset)

    Xtest, Ytest = testset

    plot_dataset(Xtest, Ytestpred)
```

```
logistic_regression(*generate_all_dataset())
```

```
Init            train loss 0.693147     valid loss 0.693147     valid acc
0.500000
Epoch 001       train loss 0.693147     valid loss 0.693147     valid acc
0.500000
Epoch 002       train loss 0.693147     valid loss 0.693147     valid acc
0.500000
Epoch 003       train loss 0.693147     valid loss 0.693147     valid acc
0.500000
Epoch 004       train loss 0.693147     valid loss 0.693147     valid acc
0.540000
Epoch 005       train loss 0.693147     valid loss 0.693147     valid acc
0.670000
Epoch 006       train loss 0.693147     valid loss 0.693147     valid acc
0.775000
Epoch 007       train loss 0.693147     valid loss 0.693147     valid acc
0.840000
Epoch 008       train loss 0.693147     valid loss 0.693147     valid acc
0.895000
Epoch 009       train loss 0.693147     valid loss 0.693147     valid acc
0.930000
Epoch 010       train loss 0.693147     valid loss 0.693147     valid acc
0.940000
Epoch 011       train loss 0.693147     valid loss 0.693147     valid acc
0.950000
Epoch 012       train loss 0.693147     valid loss 0.693147     valid acc
0.960000
Epoch 013       train loss 0.693147     valid loss 0.693147     valid acc
0.975000
Epoch 014       train loss 0.693147     valid loss 0.693147     valid acc
0.980000
Epoch 015       train loss 0.693147     valid loss 0.693147     valid acc
0.980000
Epoch 016       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 017       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 018       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 019       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 020       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 021       train loss 0.693147     valid loss 0.693147     valid acc
```

```
0.985000
Epoch 022       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 023       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 024       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 025       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 026       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 027       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 028       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 029       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 030       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 031       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 032       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 033       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 034       train loss 0.693147     valid loss 0.693147     valid acc
0.985000
Epoch 035       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 036       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 037       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 038       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 039       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 040       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 041       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 042       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 043       train loss 0.693147     valid loss 0.693147     valid acc
0.990000
Epoch 044       train loss 0.693147     valid loss 0.693147     valid acc
0.995000
Epoch 045       train loss 0.693147     valid loss 0.693147     valid acc
```

```
0.995000
Epoch 046       train loss 0.693147     valid loss 0.693147     valid acc
0.995000
Epoch 047       train loss 0.693147     valid loss 0.693147     valid acc
0.995000
Epoch 048       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 049       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 050       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 051       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 052       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 053       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 054       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 055       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 056       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 057       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 058       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 059       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 060       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 061       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 062       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 063       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 064       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 065       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 066       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 067       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 068       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 069       train loss 0.693147     valid loss 0.693147     valid acc
```

```
1.000000
Epoch 070       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 071       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 072       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 073       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 074       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 075       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 076       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 077       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 078       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 079       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 080       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 081       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 082       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 083       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 084       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 085       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 086       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 087       train loss 0.693147     valid loss 0.693147     valid acc
1.000000
Epoch 088       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 089       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 090       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 091       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 092       train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 093       train loss 0.693147     valid loss 0.693146     valid acc
```

```
1.000000
Epoch 094        train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 095        train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 096        train loss 0.693147     valid loss 0.693146     valid acc
1.000000
Epoch 097        train loss 0.693146     valid loss 0.693146     valid acc
1.000000
Epoch 098        train loss 0.693146     valid loss 0.693146     valid acc
1.000000
Epoch 099        train loss 0.693146     valid loss 0.693146     valid acc
1.000000
Epoch 100        train loss 0.693146     valid loss 0.693146     valid acc
1.000000
Test loss 0.693146       test acc 1.000000
```