p2_test

November 3, 2023

1 Multiclass Classification [30% impl, 20% writeup]

[1]:
```
from sklearn.tree import DecisionTreeClassifier
import multiclass
import util
from datasets import *
```

1.1 Getting Started

[2]:
```
h = multiclass.OAA(20, lambda: DecisionTreeClassifier(max_depth=1))
```

[3]:
```
h.train(WineData.X, WineData.Y)
```

```
training classifier for 0 versus rest
training classifier for 1 versus rest
training classifier for 2 versus rest
training classifier for 3 versus rest
training classifier for 4 versus rest
training classifier for 5 versus rest
training classifier for 6 versus rest
training classifier for 7 versus rest
training classifier for 8 versus rest
training classifier for 9 versus rest
training classifier for 10 versus rest
training classifier for 11 versus rest
training classifier for 12 versus rest
training classifier for 13 versus rest
training classifier for 14 versus rest
training classifier for 15 versus rest
training classifier for 16 versus rest
training classifier for 17 versus rest
training classifier for 18 versus rest
training classifier for 19 versus rest
```

[4]:
```
P = h.predictAll(WineData.Xte)
```

[5]:
```
mean(P == WineData.Yte)
```

[5]:
```
0.2949907235621521
```

[6]:
```
mode(WineData.Y)
```

[6]:
```
1
```

[7]:
```
WineData.labels[1]
```

[7]:
```
'Cabernet-Sauvignon'
```

[8]:
```
mean(WineData.Yte == 1)
```

[8]:
```
0.1725417439703154
```

[9]:
```
P = h.predictAll(WineData.Xte, useZeroOne=True)
```

[10]:
```
mean(P == WineData.Yte)
```

[10]:
```
0.19109461966604824
```

1.2 WU1 Code

1.2.1 WU1 a OAA code

[11]:
```
h = multiclass.OAA(5, lambda: DecisionTreeClassifier(max_depth=3))
```

[12]:
```
h.train(WineDataSmall.X, WineDataSmall.Y)
```

```
training classifier for 0 versus rest
training classifier for 1 versus rest
training classifier for 2 versus rest
training classifier for 3 versus rest
training classifier for 4 versus rest
```

[13]:
```
P = h.predictAll(WineDataSmall.Xte)
```

[14]:
```
mean(P == WineDataSmall.Yte)
```

[14]:
```
0.6039387308533917
```

[15]:
```
mean(WineDataSmall.Yte == 1)
```

[15]:
```
0.40700218818380746
```

[16]:
```
WineDataSmall.labels[0]
```

[16]:
```
'Sauvignon-Blanc'
```

[17]:
```
util.showTree(h.f[0], WineDataSmall.words)
```

```
citrus?
-N-> lime?
|    -N-> gooseberry?
|    |    -N-> class 0  (356 for class 0, 10 for class 1)
|    |    -Y-> class 1  (0 for class 0, 4 for class 1)
|    -Y-> variety?
|    |    -N-> class 1  (1 for class 0, 15 for class 1)
|    |    -Y-> class 0  (2 for class 0, 0 for class 1)
-Y-> grapefruit?
|    -N-> flavors?
|    |    -N-> class 1  (4 for class 0, 12 for class 1)
|    |    -Y-> class 0  (11 for class 0, 5 for class 1)
|    -Y-> honey?
|    |    -N-> class 1  (0 for class 0, 14 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
```

[18]:
```
WineDataSmall.labels[2]
```

[18]:
```
'Pinot-Noir'
```

[19]:
```
util.showTree(h.f[2], WineDataSmall.words)
```

```
cherry?
-N-> raspberries?
|    -N-> strawberry?
|    |    -N-> class 0  (225 for class 0, 58 for class 1)
|    |    -Y-> class 1  (0 for class 0, 4 for class 1)
|    -Y-> forward?
|    |    -N-> class 1  (0 for class 0, 12 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
-Y-> cassis?
|    -N-> petit?
|    |    -N-> class 1  (36 for class 0, 68 for class 1)
|    |    -Y-> class 0  (8 for class 0, 0 for class 1)
|    -Y-> allspice?
|    |    -N-> class 0  (21 for class 0, 0 for class 1)
|    |    -Y-> class 1  (0 for class 0, 2 for class 1)
```

1.2.2 WU1 b OAA code

[20]:
```
h = multiclass.OAA(20, lambda: DecisionTreeClassifier(max_depth=3))
```

[21]:
```
%%time
h.train(WineData.X, WineData.Y)
```

```
training classifier for 0 versus rest
training classifier for 1 versus rest
training classifier for 2 versus rest
training classifier for 3 versus rest
```

```
training classifier for 4 versus rest
training classifier for 5 versus rest
training classifier for 6 versus rest
training classifier for 7 versus rest
training classifier for 8 versus rest
training classifier for 9 versus rest
training classifier for 10 versus rest
training classifier for 11 versus rest
training classifier for 12 versus rest
training classifier for 13 versus rest
training classifier for 14 versus rest
training classifier for 15 versus rest
training classifier for 16 versus rest
training classifier for 17 versus rest
training classifier for 18 versus rest
training classifier for 19 versus rest
CPU times: total: 281 ms
Wall time: 289 ms
```

[22]:
```
%%time
P = h.predictAll(WineData.Xte)
```

```
CPU times: total: 1.91 s
Wall time: 1.92 s
```

[23]:
```
mean(P == WineData.Yte)
```

[23]:
```
0.3738404452690167
```

[24]:
```
WineData.labels[17]
```

[24]:
```
'Viognier'
```

[25]:
```
util.showTree(h.f[17], WineData.words)
```

```
peaches?
-N-> nectarine?
|    -N-> chilled?
|    |    -N-> class 0  (1036 for class 0, 1 for class 1)
|    |    -Y-> class 0  (6 for class 0, 1 for class 1)
|    -Y-> tannins?
|    |    -N-> class 0  (13 for class 0, 1 for class 1)
|    |    -Y-> class 1  (0 for class 0, 1 for class 1)
-Y-> milk?
|    -N-> harmonious?
|    |    -N-> class 0  (14 for class 0, 0 for class 1)
|    |    -Y-> class 1  (0 for class 0, 1 for class 1)
|    -Y-> class 1       (0 for class 0, 3 for class 1)
```

1.2.3 WU1 c OAA code

[26]:
```
P = h.predictAll(WineData.Xte, True)
```

[27]:
```
mean(P == WineData.Yte)
```

[27]:
```
0.24489795918367346
```

1.2.4 WU1 a AVA code

[28]:
```
h = multiclass.AVA(5, lambda: DecisionTreeClassifier(max_depth=3))
h.train(WineDataSmall.X, WineDataSmall.Y)
```

```
training classifier for 1 versus 0
training classifier for 2 versus 0
training classifier for 2 versus 1
training classifier for 3 versus 0
training classifier for 3 versus 1
training classifier for 3 versus 2
training classifier for 4 versus 0
training classifier for 4 versus 1
training classifier for 4 versus 2
training classifier for 4 versus 3
```

[29]:
```
WineDataSmall.labels[0]
```

[29]:
```
'Sauvignon-Blanc'
```

[30]:
```
label_j = 0
for label_i,trees in enumerate(h.f):
    if label_i == label_j:
        for label_k,tree in enumerate(trees):
            print(f"Tree for {WineDataSmall.labels[label_i]} (0) vs␣
 ↪{WineDataSmall.labels[label_k]} (+1)")
            util.showTree(tree, WineDataSmall.words)
            print()
    elif label_i > label_j:
        print(f"Tree for {WineDataSmall.labels[label_i]} (0) vs {WineDataSmall.
 ↪labels[label_j]} (+1)")
        util.showTree(trees[label_j], WineDataSmall.words)
        print()
```

```
Tree for Cabernet-Sauvignon (0) vs Sauvignon-Blanc (+1)
citrus?
-N-> lime?
|    -N-> melon?
|    |    -N-> class 0  (187 for class 0, 9 for class 1)
|    |    -Y-> class 1  (0 for class 0, 5 for class 1)
|    -Y-> class 1       (0 for class 0, 15 for class 1)
```

```
-Y-> class 1    (0 for class 0, 31 for class 1)

Tree for Pinot-Noir (0) vs Sauvignon-Blanc (+1)
crisp?
-N-> lime?
|    -N-> lemon?
|    |    -N-> class 0  (141 for class 0, 9 for class 1)
|    |    -Y-> class 1  (0 for class 0, 8 for class 1)
|    -Y-> mild?
|    |    -N-> class 1  (0 for class 0, 13 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
-Y-> red?
|    -N-> class 1       (0 for class 0, 30 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)

Tree for Pinot-Gris (0) vs Sauvignon-Blanc (+1)
thai?
-N-> very?
|    -N-> american?
|    |    -N-> class 1  (4 for class 0, 56 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> ripe?
|    |    -N-> class 1  (1 for class 0, 4 for class 1)
|    |    -Y-> class 0  (4 for class 0, 0 for class 1)
-Y-> class 0    (5 for class 0, 0 for class 1)

Tree for Pinot-Grigio (0) vs Sauvignon-Blanc (+1)
apple?
-N-> pasta?
|    -N-> quite?
|    |    -N-> class 1  (11 for class 0, 56 for class 1)
|    |    -Y-> class 0  (3 for class 0, 0 for class 1)
|    -Y-> class 0       (4 for class 0, 0 for class 1)
-Y-> bright?
|    -N-> class 0       (10 for class 0, 0 for class 1)
|    -Y-> reminiscent?
|    |    -N-> class 1  (0 for class 0, 4 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
```

[31]:
```
WineData.labels[2]
```

[31]:
```
'Pinot-Noir'
```

[32]:
```
label_j = 2
for label_i,trees in enumerate(h.f):
    if label_i == label_j:
        for label_k,tree in enumerate(trees):
```

```
            print(f"Tree for {WineDataSmall.labels[label_i]} (0) vs␣
↪{WineDataSmall.labels[label_k]} (+1)")
            util.showTree(tree, WineDataSmall.words)
            print()
    elif label_i > label_j:
        print(f"Tree for {WineDataSmall.labels[label_i]} (0) vs {WineDataSmall.
↪labels[label_j]} (+1)")
        util.showTree(trees[label_j], WineDataSmall.words)
        print()
```

```
Tree for Pinot-Noir (0) vs Sauvignon-Blanc (+1)
crisp?
-N-> lime?
|    -N-> lemon?
|    |    -N-> class 0  (141 for class 0, 9 for class 1)
|    |    -Y-> class 1  (0 for class 0, 8 for class 1)
|    -Y-> mild?
|    |    -N-> class 1  (0 for class 0, 13 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
-Y-> red?
|    -N-> class 1       (0 for class 0, 30 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)

Tree for Pinot-Noir (0) vs Cabernet-Sauvignon (+1)
cassis?
-N-> acidity?
|    -N-> duck?
|    |    -N-> class 1  (92 for class 0, 129 for class 1)
|    |    -Y-> class 0  (11 for class 0, 0 for class 1)
|    -Y-> tannins?
|    |    -N-> class 0  (22 for class 0, 0 for class 1)
|    |    -Y-> class 0  (15 for class 0, 11 for class 1)
-Y-> tea?
|    -N-> 100?
|    |    -N-> class 1  (1 for class 0, 47 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)

Tree for Pinot-Gris (0) vs Pinot-Noir (+1)
crisp?
-N-> peach?
|    -N-> pear?
|    |    -N-> class 1  (3 for class 0, 142 for class 1)
|    |    -Y-> class 0  (2 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> red?
|    -N-> class 0       (7 for class 0, 0 for class 1)
```

```
|    -Y-> class 1       (0 for class 0, 2 for class 1)

Tree for Pinot-Grigio (0) vs Pinot-Noir (+1)
straw?
-N-> crisp?
|    -N-> example?
|    |    -N-> class 1  (8 for class 0, 142 for class 1)
|    |    -Y-> class 0  (2 for class 0, 0 for class 1)
|    -Y-> red?
|    |    -N-> class 0  (7 for class 0, 0 for class 1)
|    |    -Y-> class 1  (0 for class 0, 2 for class 1)
-Y-> class 0    (12 for class 0, 0 for class 1)
```

1.2.5 WU1 b AVA code

[33]:
```
h = multiclass.AVA(20, lambda: DecisionTreeClassifier(max_depth=3))
```

[34]:
```
%%time
h.train(WineData.X, WineData.Y)
```

```
training classifier for 1 versus 0
training classifier for 2 versus 0
training classifier for 2 versus 1
training classifier for 3 versus 0
training classifier for 3 versus 1
training classifier for 3 versus 2
training classifier for 4 versus 0
training classifier for 4 versus 1
training classifier for 4 versus 2
training classifier for 4 versus 3
training classifier for 5 versus 0
training classifier for 5 versus 1
training classifier for 5 versus 2
training classifier for 5 versus 3
training classifier for 5 versus 4
training classifier for 6 versus 0
training classifier for 6 versus 1
training classifier for 6 versus 2
training classifier for 6 versus 3
training classifier for 6 versus 4
training classifier for 6 versus 5
training classifier for 7 versus 0
training classifier for 7 versus 1
training classifier for 7 versus 2
training classifier for 7 versus 3
training classifier for 7 versus 4
training classifier for 7 versus 5
```

```
training classifier for 7 versus 6
training classifier for 8 versus 0
training classifier for 8 versus 1
training classifier for 8 versus 2
training classifier for 8 versus 3
training classifier for 8 versus 4
training classifier for 8 versus 5
training classifier for 8 versus 6
training classifier for 8 versus 7
training classifier for 9 versus 0
training classifier for 9 versus 1
training classifier for 9 versus 2
training classifier for 9 versus 3
training classifier for 9 versus 4
training classifier for 9 versus 5
training classifier for 9 versus 6
training classifier for 9 versus 7
training classifier for 9 versus 8
training classifier for 10 versus 0
training classifier for 10 versus 1
training classifier for 10 versus 2
training classifier for 10 versus 3
training classifier for 10 versus 4
training classifier for 10 versus 5
training classifier for 10 versus 6
training classifier for 10 versus 7
training classifier for 10 versus 8
training classifier for 10 versus 9
training classifier for 11 versus 0
training classifier for 11 versus 1
training classifier for 11 versus 2
training classifier for 11 versus 3
training classifier for 11 versus 4
training classifier for 11 versus 5
training classifier for 11 versus 6
training classifier for 11 versus 7
training classifier for 11 versus 8
training classifier for 11 versus 9
training classifier for 11 versus 10
training classifier for 12 versus 0
training classifier for 12 versus 1
training classifier for 12 versus 2
training classifier for 12 versus 3
training classifier for 12 versus 4
training classifier for 12 versus 5
training classifier for 12 versus 6
training classifier for 12 versus 7
training classifier for 12 versus 8
```

```
training classifier for 12 versus 9
training classifier for 12 versus 10
training classifier for 12 versus 11
training classifier for 13 versus 0
training classifier for 13 versus 1
training classifier for 13 versus 2
training classifier for 13 versus 3
training classifier for 13 versus 4
training classifier for 13 versus 5
training classifier for 13 versus 6
training classifier for 13 versus 7
training classifier for 13 versus 8
training classifier for 13 versus 9
training classifier for 13 versus 10
training classifier for 13 versus 11
training classifier for 13 versus 12
training classifier for 14 versus 0
training classifier for 14 versus 1
training classifier for 14 versus 2
training classifier for 14 versus 3
training classifier for 14 versus 4
training classifier for 14 versus 5
training classifier for 14 versus 6
training classifier for 14 versus 7
training classifier for 14 versus 8
training classifier for 14 versus 9
training classifier for 14 versus 10
training classifier for 14 versus 11
training classifier for 14 versus 12
training classifier for 14 versus 13
training classifier for 15 versus 0
training classifier for 15 versus 1
training classifier for 15 versus 2
training classifier for 15 versus 3
training classifier for 15 versus 4
training classifier for 15 versus 5
training classifier for 15 versus 6
training classifier for 15 versus 7
training classifier for 15 versus 8
training classifier for 15 versus 9
training classifier for 15 versus 10
training classifier for 15 versus 11
training classifier for 15 versus 12
training classifier for 15 versus 13
training classifier for 15 versus 14
training classifier for 16 versus 0
training classifier for 16 versus 1
training classifier for 16 versus 2
```

```
training classifier for 16 versus 3
training classifier for 16 versus 4
training classifier for 16 versus 5
training classifier for 16 versus 6
training classifier for 16 versus 7
training classifier for 16 versus 8
training classifier for 16 versus 9
training classifier for 16 versus 10
training classifier for 16 versus 11
training classifier for 16 versus 12
training classifier for 16 versus 13
training classifier for 16 versus 14
training classifier for 16 versus 15
training classifier for 17 versus 0
training classifier for 17 versus 1
training classifier for 17 versus 2
training classifier for 17 versus 3
training classifier for 17 versus 4
training classifier for 17 versus 5
training classifier for 17 versus 6
training classifier for 17 versus 7
training classifier for 17 versus 8
training classifier for 17 versus 9
training classifier for 17 versus 10
training classifier for 17 versus 11
training classifier for 17 versus 12
training classifier for 17 versus 13
training classifier for 17 versus 14
training classifier for 17 versus 15
training classifier for 17 versus 16
training classifier for 18 versus 0
training classifier for 18 versus 1
training classifier for 18 versus 2
training classifier for 18 versus 3
training classifier for 18 versus 4
training classifier for 18 versus 5
training classifier for 18 versus 6
training classifier for 18 versus 7
training classifier for 18 versus 8
training classifier for 18 versus 9
training classifier for 18 versus 10
training classifier for 18 versus 11
training classifier for 18 versus 12
training classifier for 18 versus 13
training classifier for 18 versus 14
training classifier for 18 versus 15
training classifier for 18 versus 16
training classifier for 18 versus 17
```

```
training classifier for 19 versus 0
training classifier for 19 versus 1
training classifier for 19 versus 2
training classifier for 19 versus 3
training classifier for 19 versus 4
training classifier for 19 versus 5
training classifier for 19 versus 6
training classifier for 19 versus 7
training classifier for 19 versus 8
training classifier for 19 versus 9
training classifier for 19 versus 10
training classifier for 19 versus 11
training classifier for 19 versus 12
training classifier for 19 versus 13
training classifier for 19 versus 14
training classifier for 19 versus 15
training classifier for 19 versus 16
training classifier for 19 versus 17
training classifier for 19 versus 18
CPU times: total: 469 ms
Wall time: 474 ms
```

[35]:
```
%%time
P = h.predictAll(WineData.Xte)
```

```
CPU times: total: 18.4 s
Wall time: 18.4 s
```

[36]:
```
mean(P == WineData.Yte)
```

[36]:
```
0.2699443413729128
```

[37]:
```
WineData.labels[17]
```

[37]:
```
'Viognier'
```

[38]:
```
label_j = 17
for label_i,trees in enumerate(h.f):
    if label_i == label_j:
        for label_k,tree in enumerate(trees):
            print(f"Tree for {WineData.labels[label_i]} (0) vs {WineData.
 ↪labels[label_k]} (+1)")
            util.showTree(tree, WineData.words)
            print()
    elif label_i > label_j:
        print(f"Tree for {WineData.labels[label_i]} (0) vs {WineData.
 ↪labels[label_j]} (+1)")
        util.showTree(trees[label_j], WineData.words)
```

```
        print()
```

```
Tree for Viognier (0) vs Sauvignon-Blanc (+1)
floral?
-N-> tannin?
|    -N-> grenache?
|    |    -N-> class 1  (2 for class 0, 59 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (1 for class 0, 0 for class 1)
-Y-> finish?
|    -N-> class 0       (4 for class 0, 0 for class 1)
|    -Y-> class 1       (0 for class 0, 1 for class 1)

Tree for Viognier (0) vs Cabernet-Sauvignon (+1)
peach?
-N-> peaches?
|    -N-> pear?
|    |    -N-> class 1  (0 for class 0, 187 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)

Tree for Viognier (0) vs Pinot-Noir (+1)
peaches?
-N-> peach?
|    -N-> pear?
|    |    -N-> class 1  (0 for class 0, 144 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)

Tree for Viognier (0) vs Pinot-Gris (+1)
peaches?
-N-> nectarine?
|    -N-> pairs?
|    |    -N-> class 1  (1 for class 0, 15 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)

Tree for Viognier (0) vs Pinot-Grigio (+1)
peaches?
-N-> lovely?
|    -N-> combined?
|    |    -N-> class 1  (1 for class 0, 29 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
```

```
Tree for Viognier (0) vs Chardonnay (+1)
milk?
-N-> also?
|    -N-> tannin?
|    |    -N-> class 1  (3 for class 0, 147 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (1 for class 0, 0 for class 1)
-Y-> class 0    (3 for class 0, 0 for class 1)

Tree for Viognier (0) vs Brut (+1)
peaches?
-N-> apricot?
|    -N-> jasmine?
|    |    -N-> class 1  (1 for class 0, 61 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)

Tree for Viognier (0) vs Merlot (+1)
peaches?
-N-> peach?
|    -N-> apple?
|    |    -N-> class 1  (0 for class 0, 56 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> sugar?
|    |    -N-> class 0  (3 for class 0, 0 for class 1)
|    |    -Y-> class 1  (0 for class 0, 1 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)

Tree for Viognier (0) vs Shiraz (+1)
peaches?
-N-> peach?
|    -N-> pear?
|    |    -N-> class 1  (0 for class 0, 47 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)

Tree for Viognier (0) vs Malbec (+1)
peach?
-N-> peaches?
|    -N-> tannin?
|    |    -N-> class 1  (0 for class 0, 49 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
```

```
Tree for Viognier (0) vs Zinfandel (+1)
peaches?
-N-> peach?
|    -N-> pear?
|    |    -N-> class 1  (0 for class 0, 48 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)

Tree for Viognier (0) vs Cuvee (+1)
peaches?
-N-> peach?
|    -N-> lovely?
|    |    -N-> class 1  (0 for class 0, 31 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)

Tree for Viognier (0) vs Riesling (+1)
milk?
-N-> straw?
|    -N-> red?
|    |    -N-> class 1  (2 for class 0, 34 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> provides?
|    |    -N-> class 0  (2 for class 0, 0 for class 1)
|    |    -Y-> class 1  (0 for class 0, 1 for class 1)
-Y-> class 0    (3 for class 0, 0 for class 1)

Tree for Viognier (0) vs Chianti (+1)
peach?
-N-> peaches?
|    -N-> pear?
|    |    -N-> class 1  (0 for class 0, 45 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)

Tree for Viognier (0) vs Syrah (+1)
peaches?
-N-> peach?
|    -N-> apple?
|    |    -N-> class 1  (0 for class 0, 35 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)

Tree for Viognier (0) vs Blend (+1)
```

```
freshness?
-N-> apricot?
|    -N-> light?
|    |    -N-> class 1  (1 for class 0, 31 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> accompaniment?
|    |    -N-> class 0  (3 for class 0, 0 for class 1)
|    |    -Y-> class 1  (0 for class 0, 1 for class 1)
-Y-> class 0    (3 for class 0, 0 for class 1)

Tree for Viognier (0) vs Rhone (+1)
floral?
-N-> very?
|    -N-> jasmine?
|    |    -N-> class 1  (1 for class 0, 16 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)

Tree for Carmenere (0) vs Viognier (+1)
floral?
-N-> peaches?
|    -N-> white?
|    |    -N-> class 0  (14 for class 0, 0 for class 1)
|    |    -Y-> class 1  (1 for class 0, 2 for class 1)
|    -Y-> class 1       (0 for class 0, 2 for class 1)
-Y-> class 1    (0 for class 0, 4 for class 1)

Tree for Moscato (0) vs Viognier (+1)
fruits?
-N-> white?
|    -N-> sauce?
|    |    -N-> class 0  (15 for class 0, 0 for class 1)
|    |    -Y-> class 1  (0 for class 0, 1 for class 1)
|    -Y-> intensely?
|    |    -N-> class 1  (0 for class 0, 3 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
-Y-> class 1    (0 for class 0, 4 for class 1)
```

1.2.6 WU1 c AVA code

[39]:
```
P = h.predictAll(WineData.Xte, True)
```

[40]:
```
mean(P == WineData.Yte)
```

[40]: 0.26437847866419295

1.3 WU1 (10%): Answer questions A, B and C for both OAA and AVA. (Questions about "indicative" are open-ended. Any reasonable answers with analysis will be credited.)

Note my definition of "words that are indicative of label x" is basically what combination of words result in the most label x observations in the leaves of the corresponding tree(s), given that the majority of observations in that leaf is of label x. This is because you have to look at the words in context of the splits (hence the combination criteria) and you want the words that influence the most observations (hence the greedy criteria), and lastly you still want to make sure you predict x (hence the majority criteria).

(A) Train depth 3 decision trees on the WineDataSmall task. What words are most indicative of being Sauvignon-Blanc? Which words are most indicative of not being Sauvignon-Blanc? What about Pinot-Noir (label==2)? (OAA) The words most indicative of being Sauvignon-Blanc are the combination of words (considering both their appearance (Y) or lack thereof (N)) that result in the most class 1 observations in the Sauvignon-Blanc vs All tree's leaves, granted that the number of class 1 observations are greater than the number of class 0 observations (i.e. prediction is class 1). The top 2 are:

no citrus, has lime, no apple (15/16 class 1)

has citrus, has grapefruit, no extremely (14/14 class 1)

The words most indicative of NOT being Sauvignon-Blanc are the words that result in the most class 0 observations in the Sauvignon-Blanc vs All tree's leaves, granted that the number of class 0 observations are greater than the number of class 1 observations (i.e. prediction is class 0). The top 2 are:

no citrus, no lime, no gooseberry (356/366 class 0)

has citrus, no grapefruit, has flavors (11/16 class 0)

The words most indicative of being Pinot-Noir are the combination of words (considering both their appearance (Y) or lack thereof (N)) that result in the most class 1 observations in the Pinot-Noir vs All tree's leaves, granted that the number of class 1 observations are greater than the number of class 0 observations (i.e. prediction is class 1). The top 2 are:

has cherry, no cassis, no verdot (68/104 class 1)

no cherry, has raspberries, no integrated (12/12 class 1)

The words most indicative of NOT being Pinot-Noir are the combination of words that result in the most class 0 observations in the Pinot-Noir vs All tree's leaves, granted that the number of class 0 observations are greater than the number of class 1 observations (i.e. prediction is class 0). The top 2 are:

no cherry, no raspberries, and no strawberry (225/283 class 0)

has cherry, has cassis, no allspice (21/21 class 0)

(B) Train depth 3 decision trees on the full WineData task (with 20 labels). What accuracy do you get? How long does this take (in seconds)? One of my least favorite

wines is Viognier – what words are indicative of this? (OAA) I got a 36.7% test accuracy. It takes 0.3 seconds to train the model and 2 seconds to predict using the test set.

The words most indicative of being Viognier are the combination of words (considering both their appearance (Y) or lack thereof (N)) that result in the most class 1 observations in the Viognier vs All tree's leaves, granted that the number of class 1 observations are greater than the number of class 0 observations (i.e. prediction is class 1). The top 2 are:

has peaches, has milk (3/3 class 1)

has peaches, no milk, has straw (1/1 class 1)

(C) Compare the accuracy using zero-one predictions versus using confidence. How much difference does it make? (OAA) The test accuracy I got using zero-one is 24.6%. This is about 12% less accuracy compared to when using confidence which is a big difference (for the worse). Hence, we can say using confidence is better than using zero-one when using OAA.

(A) Train depth 3 decision trees on the WineDataSmall task. What words are most indicative of being Sauvignon-Blanc? Which words are most indicative of not being Sauvignon-Blanc? What about Pinot-Noir (label==2)? (AVA) The words most indicative of being Sauvignon-Blanc are the combination of words (considering both their appearance (Y) or lack thereof (N)) that result in the most Sauvignon-Blanc labeled observations in all the trees involving Sauvignon-Blanc, granted that the final prediction is Sauvignon-Blanc. The top 4 are:

no thai, no very, no planted (56/60 Sauvignon-Blanc)

no apple, no pasta, no quite (56/67 Sauvignon-Blanc)

has citrus (31/31 Sauvignon-Blanc)

has crisp, no red (30/30 Sauvignon-Blanc)

The words most indicative of NOT being Sauvignon-Blanc are the combination of words that result in the most non-Sauvignon-Blanc labeled observations in all the trees involving Sauvignon-Blanc, granted that the final prediction is NOT Sauvignon-Blanc. The top 4 are:

no citrus, no lime, no melon (187/196 not Sauvignon-Blanc)

no crisp, no lime, no lemon (141/150 not Sauvignon-Blanc)

has apple, no bright (10/10 not Sauvignon-Blanc)

has thai (5/5 not Sauvignon-Blanc)

The words most indicative of being Pinot-Noir are the combination of words (considering both their appearance (Y) or lack thereof (N)) that result in the most Pinot-Noir labeled observations in all the trees involving Pinot-Noir, granted that the final prediction is Pinot-Noir. The top 4 are:

no crisp, no peach, no pear (142/145 Pinot-Noir)

no straw, no crisp, no example (142/150 Pinot-Noir)

no crisp, no lime, no lemon (141/150 Pinot-Noir)

no cassis, has acidity, no tannins (22/22 Pinot-Noir)

The words most indicative of NOT being Pinot-Noir are the combination of words that result in the most non-Pinot-Noir labeled observations in all the trees involving Pinot-Noir, granted that the final prediction is NOT Pinot-Noir. The top 4 are:

no cassis, no acidity, no duck (129/221 not Pinot-Noir)

has cassis, no tea, no 100 (47/48 not Pinot-Noir)

has crisp, no red (30/30 not Pinot-Noir)

no crisp, has lime, no game (13/13 not Pinot-Noir)

(B) Train depth 3 decision trees on the full WineData task (with 20 labels). What accuracy do you get? How long does this take (in seconds)? One of my least favorite wines is Viognier – what words are indicative of this? (AVA) I got a 26.2% test accuracy. It takes 0.6 seconds to train the model and 22 seconds to predict using the test set.

The words most indicative of being Viognier are the combination of words (considering both their appearance (Y) or lack thereof (N)) that result in the most Viognier labeled observations in all the trees involving Viognier, granted that the final prediction is Viognier. The top 4 are:

has fruits (4/4 Viognier)

has preaches (4/4 Viognier)

has floral (4/4 Viognier)

has peach (4/4 Viognier)

(C) Compare the accuracy using zero-one predictions versus using confidence. How much difference does it make? (AVA) The test accuracy I got using zero-one is 26.1%. This is about 0.1% less accuracy compared to when using confidence which is a small difference (for the worse). This difference is so small, that we can say using zero-one or confidence when using AVA doesn't make a significant difference.

1.4 WU2 Code

[41]:
```
t = multiclass.makeBalancedTree(range(6))
```

[42]:
```
t
```

[42]:
```
[[0 [1 2]] [3 [4 5]]]
```

[43]:
```
t.isLeaf
```

[43]:
```
False
```

[44]:
```
t.getLeft()
```

[44]:
```
[0 [1 2]]
```

[45]:
```
t.getLeft().getLeft()
```

[45]: 0

[46]:
```
t.getLeft().getLeft().isLeaf
```

[46]: True

[47]:
```
t = multiclass.makeBalancedTree(range(5))
```

[48]:
```
h = multiclass.MCTree(t, lambda: DecisionTreeClassifier(max_depth=3))
```

[49]:
```
h.train(WineDataSmall.X, WineDataSmall.Y)
```

```
training classifier for [0, 1] versus [2, 3, 4]
training classifier for [0] versus [1]
training classifier for [2] versus [3, 4]
training classifier for [3] versus [4]
```

[50]:
```
P = h.predictAll(WineDataSmall.Xte)
```

[51]:
```
mean(P == WineDataSmall.Yte)
```

[51]: 0.562363238512035

[52]:
```
t = multiclass.makeBalancedTree(range(20))
```

[53]:
```
h = multiclass.MCTree(t, lambda: DecisionTreeClassifier(max_depth=3))
```

[54]:
```
h.train(WineData.X, WineData.Y)
```

```
training classifier for [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] versus [10, 11, 12, 13,
14, 15, 16, 17, 18, 19]
training classifier for [0, 1, 2, 3, 4] versus [5, 6, 7, 8, 9]
training classifier for [0, 1] versus [2, 3, 4]
training classifier for [0] versus [1]
training classifier for [2] versus [3, 4]
training classifier for [3] versus [4]
training classifier for [5, 6] versus [7, 8, 9]
training classifier for [5] versus [6]
training classifier for [7] versus [8, 9]
training classifier for [8] versus [9]
training classifier for [10, 11, 12, 13, 14] versus [15, 16, 17, 18, 19]
training classifier for [10, 11] versus [12, 13, 14]
training classifier for [10] versus [11]
training classifier for [12] versus [13, 14]
training classifier for [13] versus [14]
training classifier for [15, 16] versus [17, 18, 19]
training classifier for [15] versus [16]
training classifier for [17] versus [18, 19]
training classifier for [18] versus [19]
```

[55]:
```
P = h.predictAll(WineData.Xte)
```

[56]:
```
mean(P == WineData.Yte)
```

[56]:
```
0.30890538033395176
```

1.5 WU2 Show the test accuracy you get with a balanced tree on the WineData using a DecisionTreeClassifier with max depth 3.

The test accuracy I got using a balanced tree on the WineData using a DecisionTreeClassifier with max depth 3 is 30.8%.

2 Gradient Descent and Linear Classification [30% impl, 20% writeup]

[57]:
```
import gd
from numpy import *
from pylab import *
from util import *
import datasets,binary,dt,gd,knn,linear,mlGraphics,multiclass,runClassifier
```

2.1 Gradient Descent

[58]:
```
gd.gd(lambda x: x**2, lambda x: 2*x, 10, 10, 0.2)
```

[58]:
```
(1.0034641051795872,
 array([100.        ,  36.        ,  18.5153247 ,  10.95094653,
         7.00860578,   4.72540613,   3.30810578,   2.38344246,
         1.75697198,   1.31968118,   1.00694021]))
```

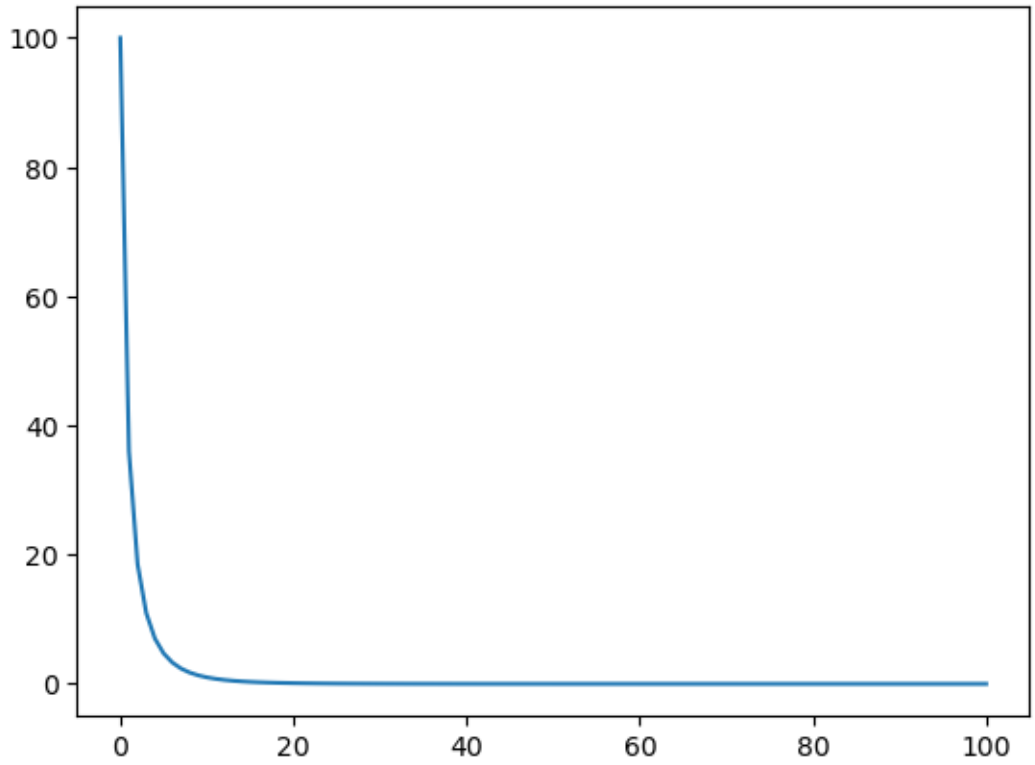
[59]:
```
x, trajectory = gd.gd(lambda x: x**2, lambda x: 2*x, 10, 100, 0.2)
```

[60]:
```
x
```

[60]:
```
0.003645900464603937
```

[61]:
```
plot(trajectory)
```

[61]:
```
[<matplotlib.lines.Line2D at 0x24d951c03d0>]
```



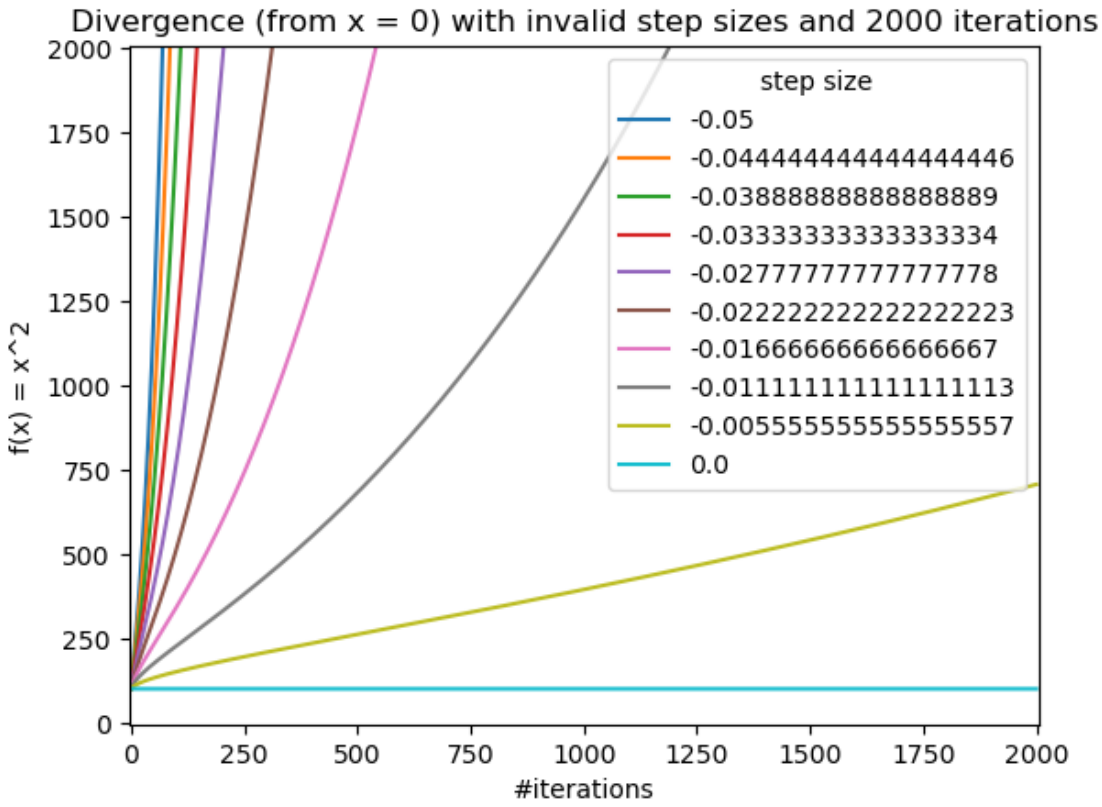
2.1.1 WU3 Code

[62]:
```
sizes = linspace(-0.05, 0,10)
for s in sizes:
    x, trajectory = gd.gd(lambda x: x**2, lambda x: 2*x, 10, 2000, s)
    print(f'x={x}, s={s}')
    plot(trajectory, label=s)
axis([-5,2005,-5,2005])
legend(title="step size")
ylabel('f(x) = x^2')
xlabel('#iterations')
title('Divergence (from x = 0) with invalid step sizes and 2000 iterations')
```

```
x=63695.230996291924, s=-0.05
x=24161.13755748736, s=-0.044444444444444446
x=9156.129085168799, s=-0.03888888888888889
x=3466.4734221414337, s=-0.03333333333333334
x=1311.1204811261139, s=-0.02777777777777778
x=495.4197937600729, s=-0.022222222222222223
x=187.01538884030032, s=-0.01666666666666667
x=70.52640196658601, s=-0.011111111111111113
x=26.570125228248305, s=-0.005555555555555557
```

```
x=10.0, s=0.0
```

[62]:
```
Text(0.5, 1.0, 'Divergence (from x = 0) with invalid step sizes and 2000
iterations')
```

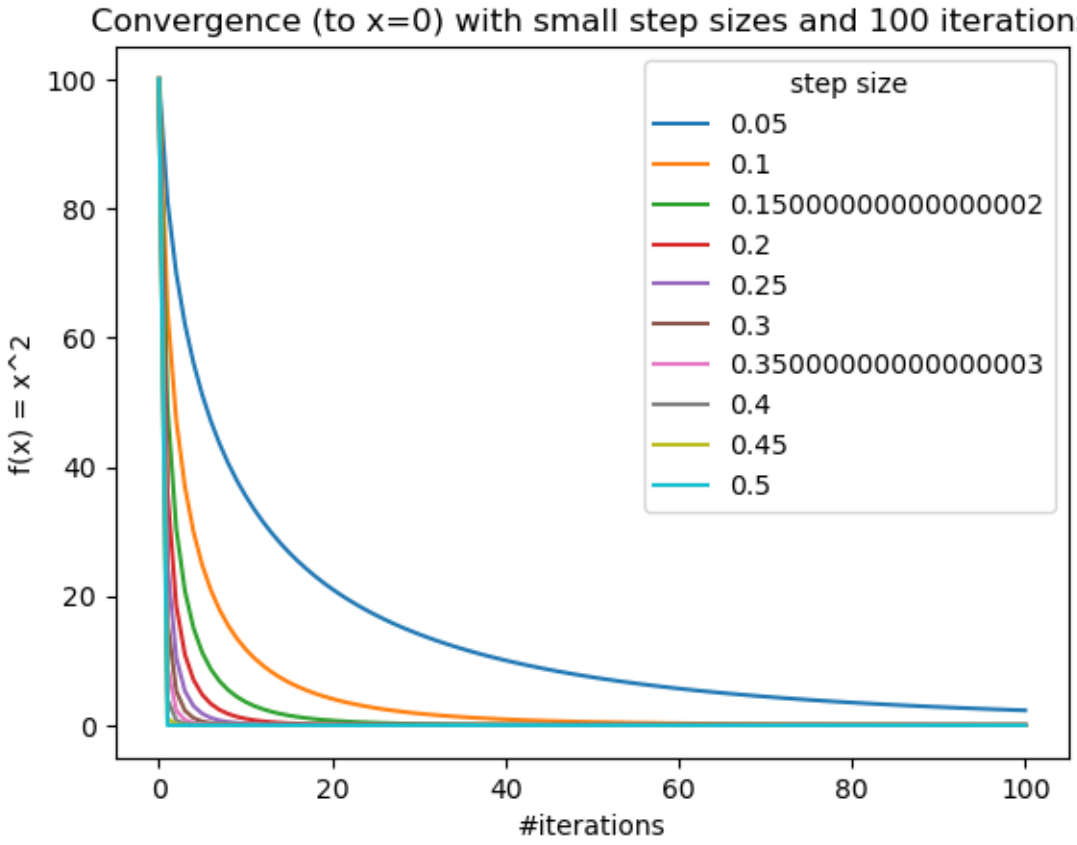


[63]:
```
sizes = linspace(0.05, 0.5,10)
for s in sizes:
    x, trajectory = gd.gd(lambda x: x**2, lambda x: 2*x, 10, 100, s)
    print(f'x={x}, s={s}')
    plot(trajectory, label=s)
axis([-5,105,-5,105])
legend(title="step size")
ylabel('f(x) = x^2')
xlabel('#iterations')
title('Convergence (to x=0) with small step sizes and 100 iterations')
```

```
x=1.517159379892363, s=0.05
x=0.21734427526984595, s=0.1
x=0.029199617704596413, s=0.15000000000000002
x=0.003645900464603937, s=0.2
x=0.0004178314061777894, s=0.25
x=4.313636016915168e-05, s=0.3
```

```
x=3.888043757509135e-06, s=0.35000000000000003
x=2.872395305714746e-07, s=0.4
x=1.4478753279228713e-08, s=0.45
x=0.0, s=0.5
```

[63]: `Text(0.5, 1.0, 'Convergence (to x=0) with small step sizes and 100 iterations')`

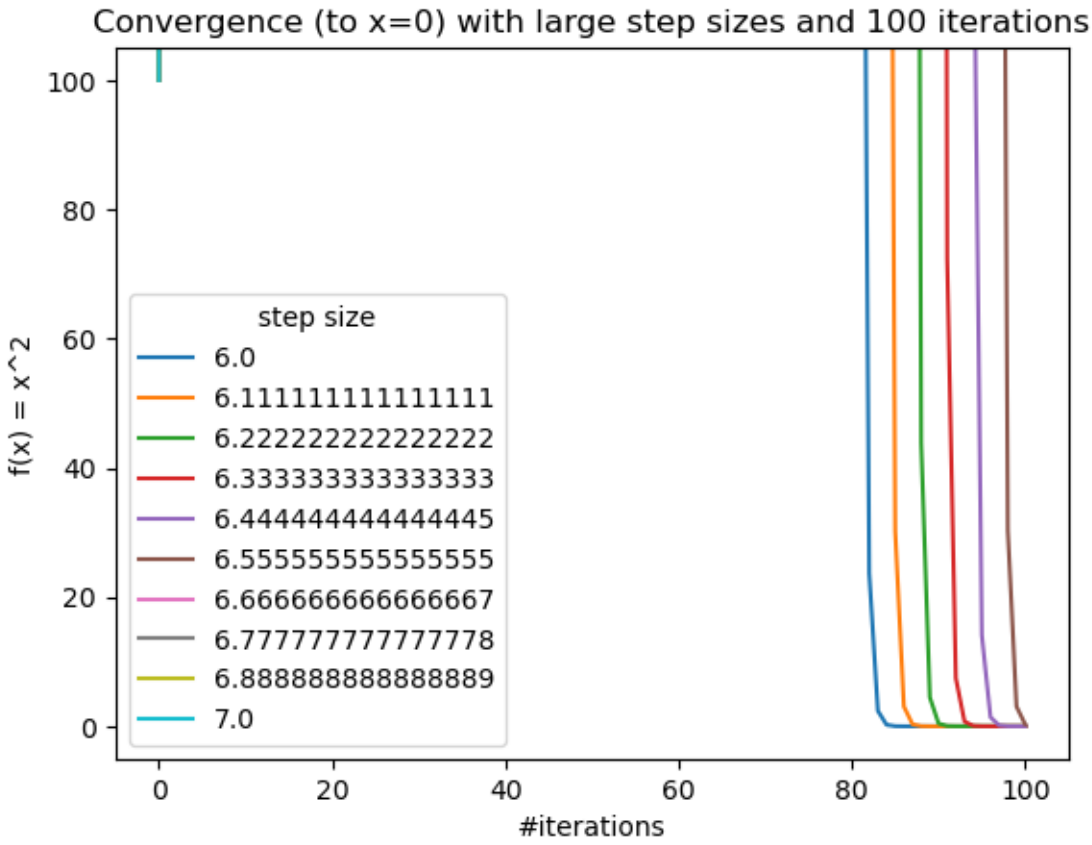


[64]:
```
sizes = linspace(6,7,10)
for s in sizes:
    x, trajectory = gd.gd(lambda x: x**2, lambda x: 2*x, 10, 100, s)
    print(f'x={x}, s={s}')
    plot(trajectory, label=s)
axis([-5,105,-5,105])
legend(title="step size")
ylabel('f(x) = x^2')
xlabel('#iterations')
title('Convergence (to x=0) with large step sizes and 100 iterations')
```

```
x=9.107243088758273e-11, s=6.0
x=1.3567187209896665e-08, s=6.111111111111111
x=1.52505286013393e-06, s=6.222222222222222
```

```
x=0.0001338012190076688, s=6.333333333333333
x=0.009419008151584689, s=6.444444444444445
x=0.5443349969987707, s=6.555555555555555
x=26.327329058204526, s=6.666666666666667
x=1083.2826860585665, s=6.777777777777778
x=38457.89097396072, s=6.888888888888889
x=1192445.5000873771, s=7.0
```

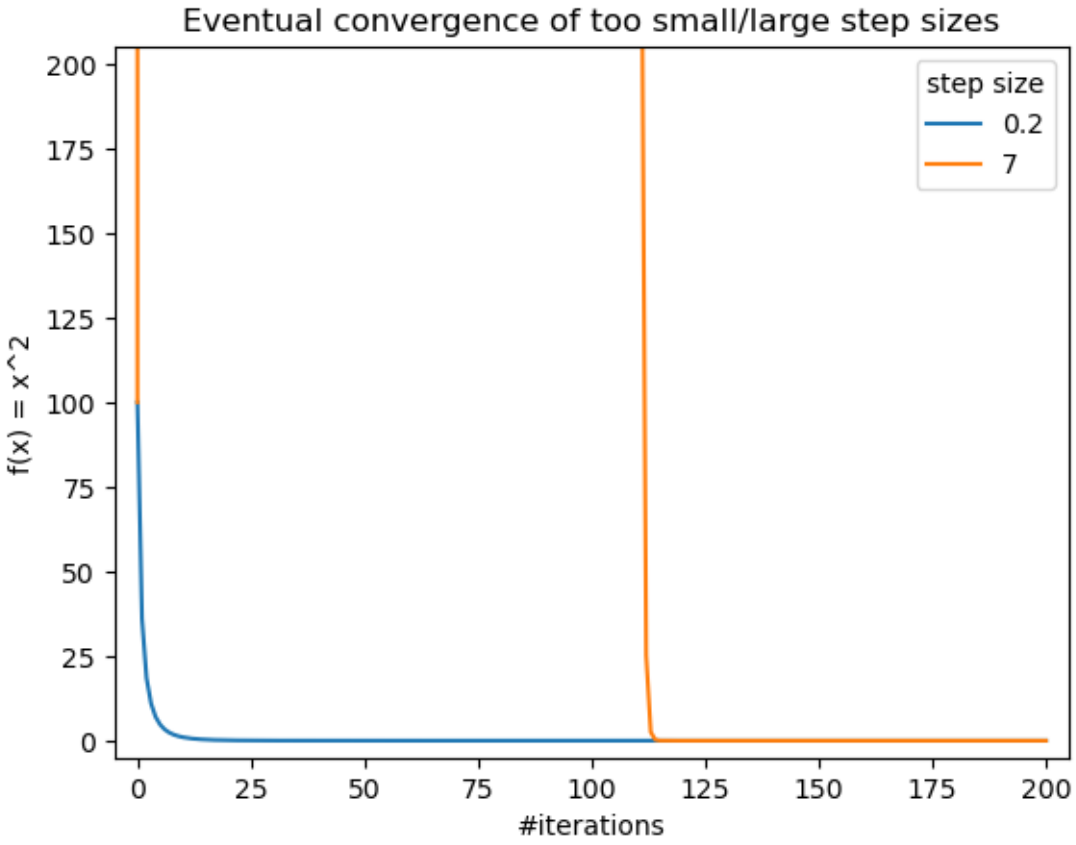
[64]: `Text(0.5, 1.0, 'Convergence (to x=0) with large step sizes and 100 iterations')`



[65]:
```
sizes = [0.2,7]
for s in sizes:
    x, trajectory = gd.gd(lambda x: x**2, lambda x: 2*x, 10, 200, s)
    print(f'x={x}, s={s}')
    plot(trajectory, label=s)
axis([-5,205,-5,205])
legend(title="step size")
ylabel('f(x) = x^2')
xlabel('#iterations')
title('Eventual convergence of too small/large step sizes')
```

```
x=0.000126086667325106, s=0.2
x=0.0, s=7
```

[65]: `Text(0.5, 1.0, 'Eventual convergence of too small/large step sizes')`



2.1.2 WU3 (5%): What is the impact of the step size on convergence? Find values of the step size where the algorithm diverges and converges.

If we suppose values of x $<$ 0.0005 are "good enough" to call it convergence (to the min of x = 0), then 0.25 <= step size <= 6.33 results in convergence when #iterations = 100. On the other hand, step size $<$ 0.25 or step size $>$ 6.33 results in divergence. Step size impacts convergence by influencing the rate of convergence. When the step size is too small or too big it takes longer for gradient descent to converge, for instance if we increase #iterations to 200, step sizes 0.2 and 7 end up converging but only after 100 iterations. Graphically you can see that values too small just converge really slowly, whereas values too big initially diverge, but due to our adaptive step size (which makes step size smaller), they do eventually converge (if you increase #iterations to be large enough i.e. expand the x-axis to the right).

However, clearly step size <= 0 will never converge no matter how many iterations you use. This is because a 0 step size remains a 0 step size even when using adaptive step size (0/k = 0 for all k), hence the x0 never moves. Also, step sizes $<$ 0 will just move the x0 in the wrong direction along

the gradient (step size remains negative even with adaptive step size because we only divide step size by a positive number, and neg/pos = neg), hence why it never converges. This can be seen in the first graph, where none of the lines converge even after 2000 iterations.

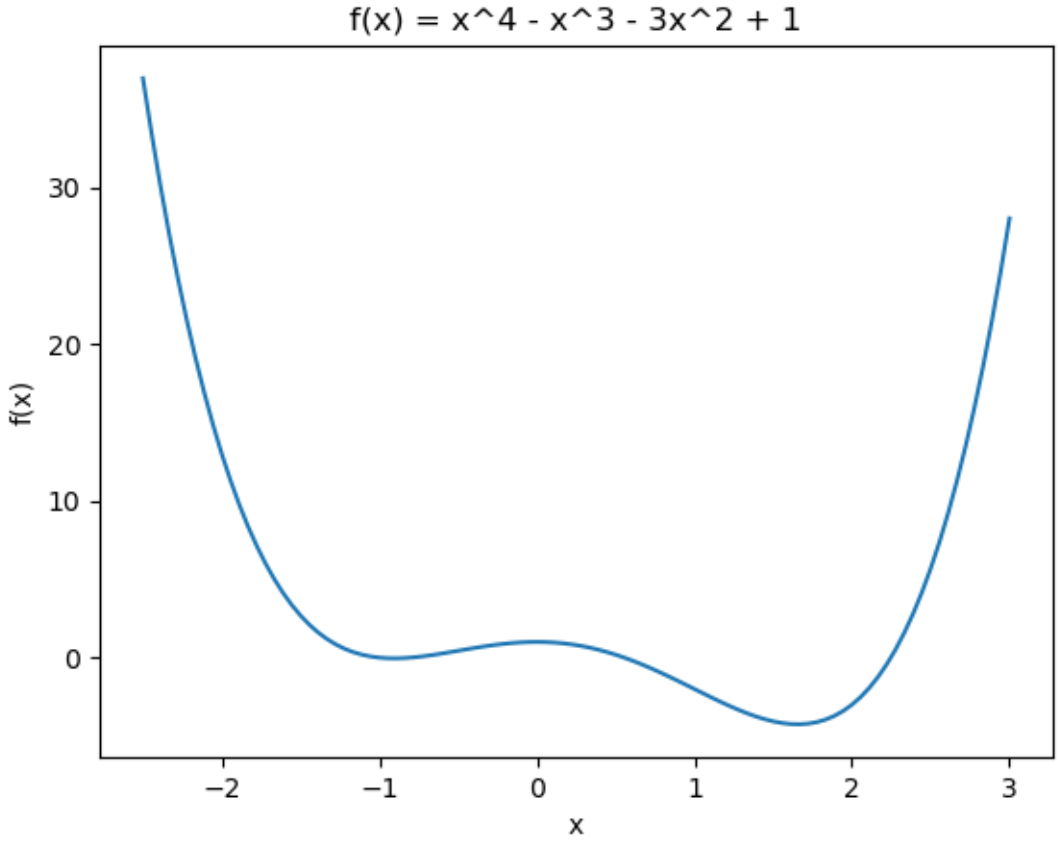
2.1.3 WU4 Code

[66]:
```
fun = lambda x: x**4 - x**3 - 3*x**2 + 1
g_fun = lambda x: 4*x**3 - 3*x**2- 6*x
```

[67]:
```
x_ = np.linspace(-2.5, 3, 100)
```

[68]:
```
plot(x_, fun(x_))
ylabel('f(x)')
xlabel('x')
title('f(x) = x^4 - x^3 - 3x^2 + 1')
```

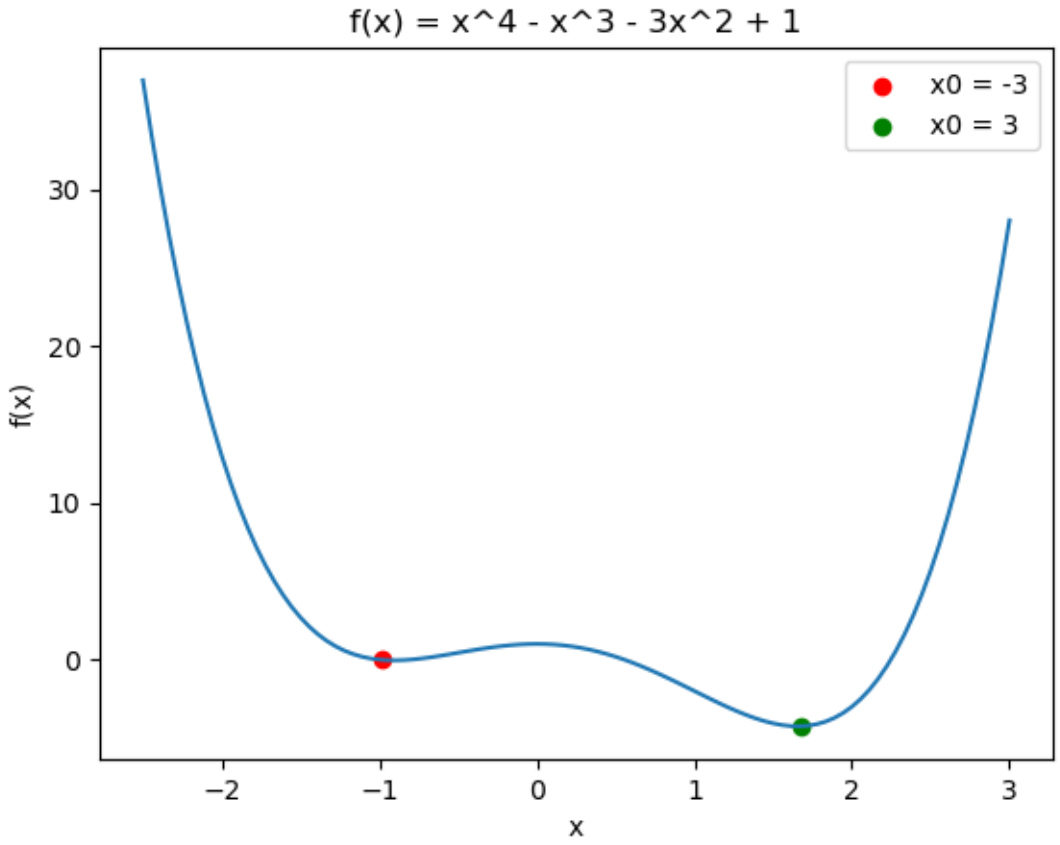
[68]:
```
Text(0.5, 1.0, 'f(x) = x^4 - x^3 - 3x^2 + 1')
```



[69]:
```
x0_1, x0_2 = -3, 3
x1, trajectory1 = gd.gd(fun, g_fun, x0_1, 100, 0.01)
```

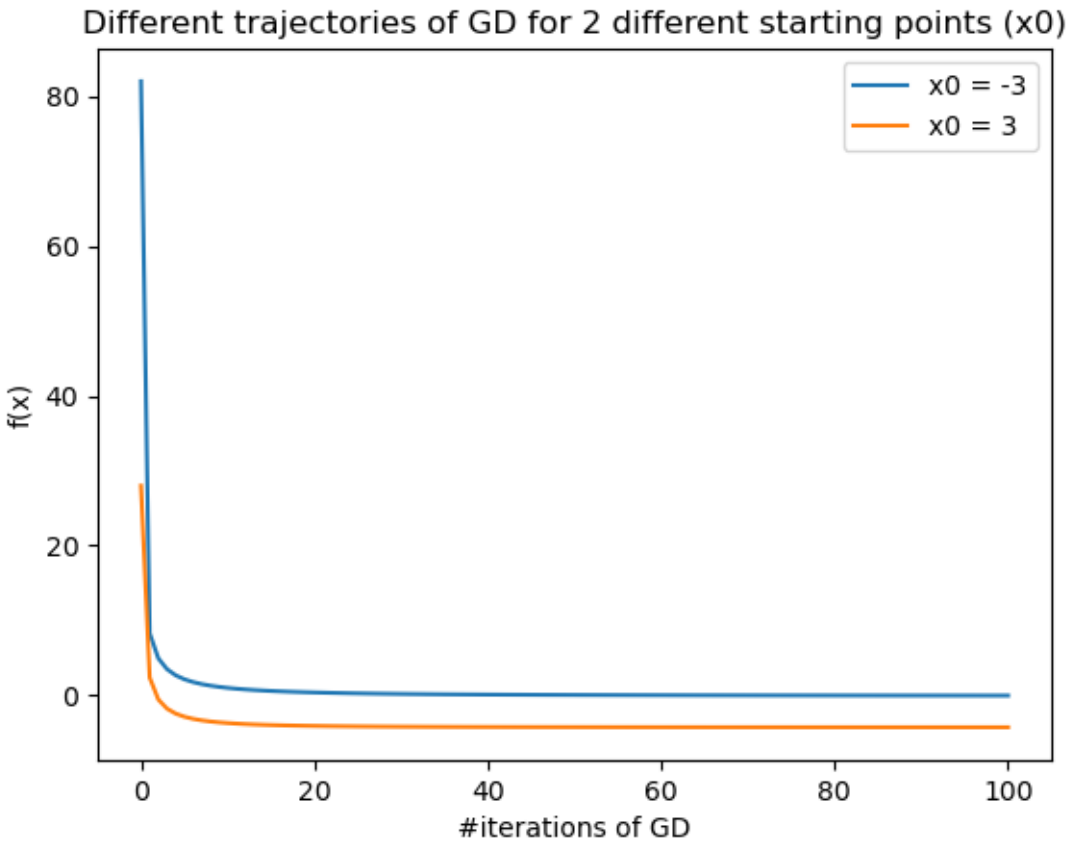
```
x2, trajectory2 = gd.gd(fun, g_fun, x0_2, 100, 0.01)
scatter(x1, fun(x1), label=f"x0 = {x0_1}", color="red")
scatter(x2, fun(x2), label=f"x0 = {x0_2}", color="green")
plot(x_, fun(x_))
ylabel('f(x)')
xlabel('x')
title('f(x) = x^4 - x^3 - 3x^2 + 1')
legend()
```

[69]: <matplotlib.legend.Legend at 0x24d962ebf50>



[70]:
```
plot(trajectory1, label=f"x0 = {x0_1}")
plot(trajectory2, label=f"x0 = {x0_2}")
ylabel('f(x)')
xlabel('#iterations of GD')
title('Different trajectories of GD for 2 different starting points (x0)')
legend()
```

[70]: <matplotlib.legend.Legend at 0x24d962aafd0>



2.1.4 WU4 (10%): Come up with a non-convex univariate optimization problem. Plot the function you're trying to minimize and show two runs of gd, one where it gets caught in a local minimum and one where it manages to make it to a global minimum. (Use different starting points to accomplish this.)

Here you can see for the non-convex univariate function f(x) = x^4 - x^3 - 3x^2 + 1, gradient descent starting at x0 = -3 gets caught in a local minima which is x = -0.905, f(x) = -0.04, whereas x0 = 3 finds the global minima which is x = 1.65, f(x) = -4.24. Graphically you can see this in both the trajectory plot and the f(x) plot. This happens because x0 = 3 is the to the right of the global minima so gradient descent descends to that global valley, whereas x0 = -3 is to the left of the local minima so gradient descent descends to that local valley.

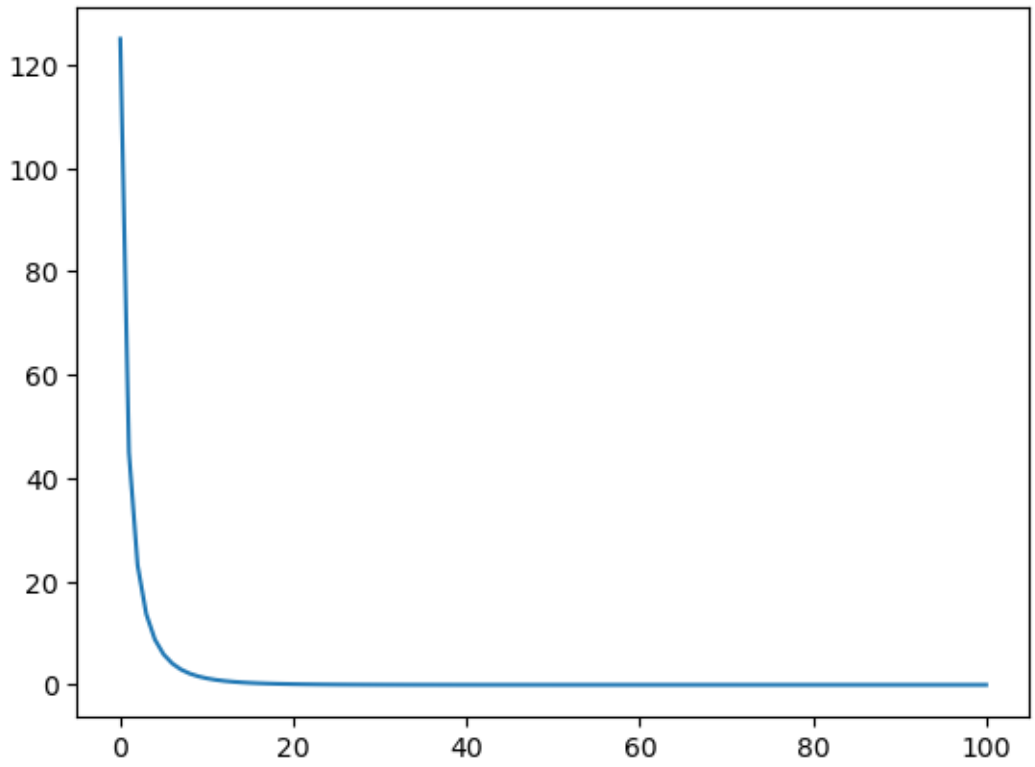
[71]:
```
x, trajectory = gd.gd(lambda x: linalg.norm(x)**2, lambda x: 2*x,␣
↪array([10,5]), 100, 0.2)
```

[72]:
```
x
```

[72]:
```
array([0.0036459 , 0.00182295])
```

[73]:
```
plot(trajectory)
```

[73]:
```
[<matplotlib.lines.Line2D at 0x24d96374ed0>]
```



2.2 Linear Classifiers

[74]:
```
f = linear.LinearClassifier({'lossFunction': linear.SquaredLoss(), 'lambda': 0,␣
↪'numIter': 100, 'stepSize': 0.5})
```

[75]:
```
runClassifier.trainTestSet(f, datasets.TwoDAxisAligned)
```

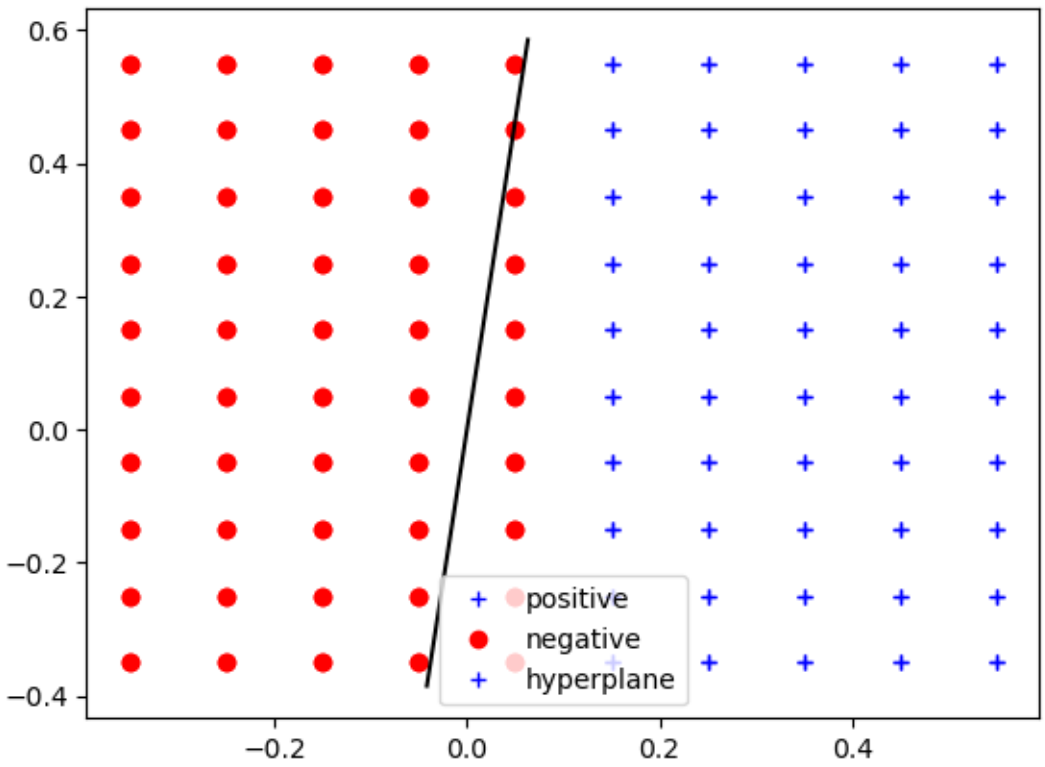
```
Training accuracy 0.91, test accuracy 0.86
```

[76]:
```
f
```

[76]:
```
w=array([ 2.73466371, -0.29563932])
```

[77]:
```
mlGraphics.plotLinearClassifier(f, datasets.TwoDAxisAligned.X, datasets.
↪TwoDAxisAligned.Y)
```

```
Axes(0.125,0.11;0.775x0.77)
```



[78]:
```
f = linear.LinearClassifier({'lossFunction': linear.SquaredLoss(), 'lambda':␣
↪10, 'numIter': 100, 'stepSize': 0.5})
```

[79]:
```
runClassifier.trainTestSet(f, datasets.TwoDAxisAligned)
```

```
Training accuracy 0.9, test accuracy 0.86
```

[80]:
```
f
```

[80]: w=array([1.30221546, -0.06764756])

[81]:
```
f = linear.LinearClassifier({'lossFunction': linear.LogisticLoss(), 'lambda':␣
↪10, 'numIter': 100, 'stepSize': 0.5})
```

[82]:
```
runClassifier.trainTestSet(f, datasets.TwoDDiagonal)
```

```
Training accuracy 0.99, test accuracy 0.86
```

[83]:
```
f
```

[83]: w=array([0.29809083, 1.01287561])

[84]:
```
f = linear.LinearClassifier({'lossFunction': linear.HingeLoss(), 'lambda': 1,␣
↪'numIter': 100, 'stepSize': 0.5})
```

[85]:
```
runClassifier.trainTestSet(f, datasets.TwoDDiagonal)
```

```
Training accuracy 0.98, test accuracy 0.86
```

[86]:
```
f
```

[86]:
```
w=array([1.17110065, 4.67288657])
```

2.2.1 WU5 Code

[87]:
```
fsq = linear.LinearClassifier({'lossFunction': linear.SquaredLoss(), 'lambda':␣
↪1, 'numIter': 100, 'stepSize': 0.5})
runClassifier.trainTestSet(fsq, datasets.WineDataBinary)
```

```
Training accuracy 0.242915, test accuracy 0.313653
```

[88]:
```
flog = linear.LinearClassifier({'lossFunction': linear.LogisticLoss(), 'lambda':
↪ 1, 'numIter': 100, 'stepSize': 0.5})
runClassifier.trainTestSet(flog, datasets.WineDataBinary)
```

```
Training accuracy 0.995951, test accuracy 0.97417
```

[89]:
```
fhinge = linear.LinearClassifier({'lossFunction': linear.HingeLoss(), 'lambda':␣
↪1, 'numIter': 100, 'stepSize': 0.5})
runClassifier.trainTestSet(fhinge, datasets.WineDataBinary)
```

```
Training accuracy 0.753036, test accuracy 0.686347
```

[90]:
```
lowest = argsort(flog.weights)[:5]
```

[91]:
```
highest = argsort(flog.weights)[-5:]
```

[92]:
```
for i in lowest:
    print(f'w={flog.weights[i]}, word={datasets.WineDataBinary.words[i]}')
```

```
w=-1.169521216404042, word=tannins
w=-0.7653093906427062, word=black
w=-0.6835931677893776, word=dark
w=-0.6295907281434338, word=cherry
w=-0.5321916724675303, word=blackberry
```

[93]:
```
for i in highest[::-1]:
    print(f'w={flog.weights[i]}, word={datasets.WineDataBinary.words[i]}')
```

```
w=0.8832897531176558, word=citrus
w=0.7701247691557765, word=crisp
w=0.7108905521536929, word=lime
```

```
w=0.6891990079029969, word=acidity
w=0.6064232619016878, word=tropical
```

[94]:
```
datasets.WineData.labels[0:2]
```

[94]:
```
['Sauvignon-Blanc', 'Cabernet-Sauvignon']
```

class WineDataBinary: > labels = WineData.labels[0:2]
> X = WineData.X[WineData.Y < 2, :]
> Y = 2 * (WineData.Y[WineData.Y < 2] == 0) - 1
> Xte = WineData.Xte[WineData.Yte < 2, :]
> Yte = 2 * (WineData.Yte[WineData.Yte < 2] == 0) - 1
> words = WineData.words

[95]:
```
datasets.WineData.X[:5,:5]
```

[95]:
```
array([[1., 1., 1., 1., 1.],
       [0., 1., 0., 0., 0.],
       [0., 1., 1., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.]])
```

2.2.2 WU5 (5%): For each of the loss functions, train a model on the binary version of the wine data (called WineDataBinary) and evaluate it on the test data. You should use lambda=1 in all cases. Which works best? For that best model, look at the learned weights. Find the words corresponding to the weights with the greatest positive value and those with the greatest negative value. Hint: look at WineDataBinary.words to get the id-to-word mapping. List the top 5 positive and top 5 negative and explain.

Clearly the logistic loss worked the best because it had the highest test accuracy out of the three 97.4% (whereas squared loss had 24% and hinge loss had 75%). The reason I think logistic loss did the best is because it is smoother (compared to hinge loss) and it doesn't cater towards prediction outliers as much (compared to squared loss).

Here you can see that WineDataBinary is a dataset that only looks at the observations from label 0 or label 1, which corresponds to Sauvignon-Blanc and Cabernet-Sauvignon respectively. According to the code, we relabel Sauvignon-Blanc (label 0) as 1 and Cabernet-Sauvignon (label 1) as -1.

Since we know that the X data is either 0 or 1 (indicating if a word is present or not), we know that greater weights produce a greater prediction which drives you towards predicting Sauvignon-Blanc (+1). On the other hand, lower weights produce a lower prediction which drives you towards predicting Cabernet-Sauvignon (-1).

As a result, we can say that according to the logistic loss linear classifier, the words most indicative of Sauvignon-Blanc (i.e. having the highest weights) as opposed to Cabernet-Sauvignon are:

citrus

crisp

lime

acidity

tropical

*Note “indicative” meaning: if the word appears it helps the most with predicting Sauvignon-Blanc by increasing the prediction the most (the word must appear i.e. =1 in order to make use of the weight)

Also, we can say that according to the logistic loss linear classifier the words most indicative of Cabernet-Sauvignon (i.e. having the lowest weights) as opposed to Sauvignon-Blanc are:

tannins

black

dark

cherry

blackberry

*Note that these words are the most indicative because if they appear they help the most with predicting Cabernet-Sauvignon by decreasing the prediction the most

[ ]: