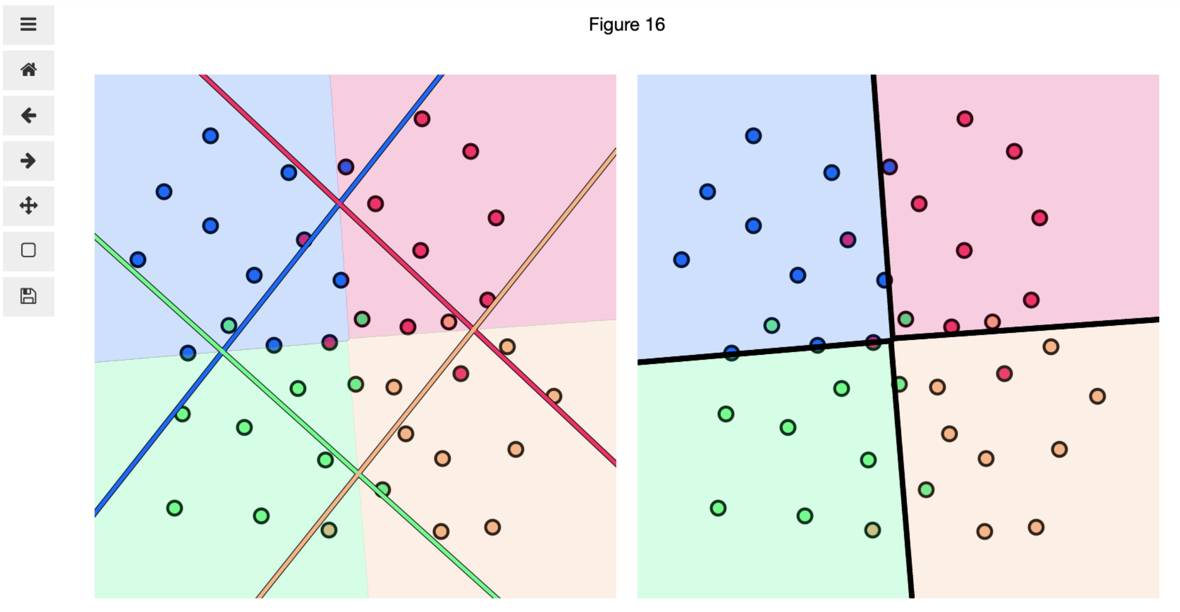
7.2

```
from mlrefined_libraries import calculus_library as calib
from mlrefined_libraries import math_optimization_library as optlib
from mlrefined_libraries import superlearn_library as superlearn
csvname = 'mlrefined_exercises/ed_2/mlrefined_datasets/superlearn_datasets/4class_data.csv'
data = np.loadtxt(csvname,delimiter = ',')
p = superlearn.ova_illustrator.Visualizer(data)
# p.show_dataset()
p.solve_2class_subproblems()
p.show_complete_coloring()
```

Figure 16



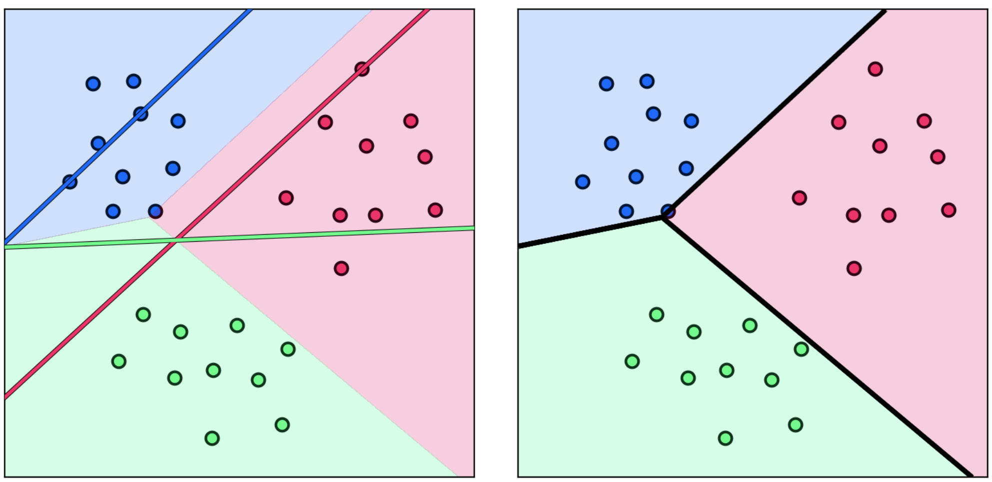
x=0.301 y=−0.003

7.3

```
def multiclass_perceptron(w):
    evals = model(x,w)
    a = np.max(evals,axis = 0)
    b = evals[y.astype(int).flatten(),np.arange(np.size(y))]
    cost = np.sum(a - b)
    cost = cost + (10**-3)*np.linalg.norm(w[1:,:],'fro')**2
    return cost/np.size(y)
```

```
data = np.loadtxt('mlrefined_exercises/ed_2/mlrefined_datasets/superlearn_datasets/3class_data.csv',delimiter = ',')
p = superlearn.multiclass_illustrator.Visualizer(data)
x = data[:-1,:]
y = data[-1:,:]
g = multiclass_perceptron
w = 0.1*np.random.randn(3,3); maxx = 1000; a = 10**(-1);
wh,ch = optimizers.gradient_descent(g,alpha_choice,max_its,w)
p.show_complete_coloring(wh, cost = multiclass_perceptron)
```

Figure 34



7.4

$$g(w_0^0, w^0, w_0^1, w^1) = \sum_{p=1}^{P} \max\left(w_0^0 + x_p^T w^0,\ w_0^1 + x_p^T w^1\right) - \underbrace{\left(w_0^{y_p} + x_p^T w^{y_p}\right)}_{=A}$$

$$= \sum_{p=1}^{P} \max\left(w_0^0 + x_p^T w^0 - A,\ w_0^1 + x_p^T w^1 - A\right)$$

$$= \sum_{p=1}^{P} \max\left(0,\ w_0^1 - w_0^0 + x_p^T (w^1 - w^0)\right) + \sum_{p=1}^{P} \max\left(0,\ w_0^0 - w_0^1 + x_p^T (w^0 - w^1)\right)$$

Let $y_p \in \{0, 1\}$ be $y_p \in \{-1, 1\}$.

$$\Rightarrow g(w_0^{-1}, w^{-1}, w_0^1, w^1) = \sum_{p=1}^{P} \max\left(0,\ w_0^1 - w_0^{-1} + x_p^T (w^{-1} - w^1)\right) +$$

$$\sum_{p=1}^{P} \max\left(0,\ w_0^{-1} - w_0^1 + x_p^T (w^{-1} - w^1)\right)$$

Thus, to simplify, we got:

$$g(w_0, \mathbf{w}) = \sum_{p=1}^{P} \max\left(0,\ -y_p\left(w_0 + \mathbf{x}_p^T \mathbf{w}\right)\right)$$

7.6

If $C=2$. if. $\sum_{c=1}^{C}\sum_{p\in\Omega_c}\log\left(1+\sum_{j=1}^{C}e^{(b_j-b_c)+x_p^T(w_j-w_c)}\right)$.

$\Rightarrow \sum_{p=\Omega_1}\log\left(1+e^{(b_j-b_c)+x_p^T(w_j-w_c)}\right)+\sum_{p=\Omega_2}$ same as left

$\Rightarrow \sum_{p=1}^{P}\log\left(1+e^{-y_p((b_2-b_1)+x_p^T(w_2-w_1))}\right)$

simply $\Rightarrow \sum_{p=1}^{P}\log\left(1+e^{-y_p(b+x_p^T w)}\right)$

7.8

Notice we always have that:

I. Addition of two (or more) convex functions is always convex.

II. Linear and affine functions are convex.

III. The max, exponential, and negative logarithm functions are all convex.

IV. Composition of two convex functions remains convex.

Each of the statements above can be verified easily using the following definition of convexity:

A function g is convex if and only if for all w1 and w2 in the domain of g and all

$\lambda \in [0,1]$, we have

$$g(\lambda w1 + (1 - \lambda) w2) \leq \lambda g(w1) + (1 - \lambda) g(w2)$$

Thus, the multi-class perceptron and softmax costs are convex.