Assignment 4

1 AdaBoost Example (25 points)

We fill in the table below. We get the final model $h(x) = 0.424h_1(x) + 0.401h_2(x) + 0.321h_3(x)$. The calculation are done in the notebook exercise 1.ipynb.

Data			$h_1(\vec{x}_i) = sign(x_1 - 0.25)$					$h_2(\vec{x}_i) = sign(-x_2 + 0.55)$					$h_3(\vec{x}_i) = sign(x_2 - 0.35)$					
x_1	x_2	y	$w_i^{(1)}$	$h_1(\vec{x}_i)$	$\varepsilon_i^{(1)}$	$\gamma_i^{(1)}$	$w_i^{(1)}\gamma_i^{(1)}$	$w_i^{(2)}$	$h_2(\vec{x}_i)$	$\varepsilon_i^{(2)}$	$\gamma_i^{(2)}$	$w_i^{(2)}\gamma_i^{(2)}$	$w_i^{(3)}$	$h_3(\vec{x}_i)$	$\varepsilon_i^{(3)}$	$\gamma_i^{(3)}$	$w_i^{(3)}\gamma_i^{(3)}$	$h(\vec{x}_i)$
0.1	0.15	-1	0.1	-1	0	0.655	0.065	0.071	1	0.071	1.493	0.107	0.115	-1	0	0.725	0.084	-1
0.15	0.25	-1	0.1	-1	0	0.655	0.065	0.071	1	0.071	1.493	0.107	0.115	-1	0	0.725	0.084	-1
0.3	0.7	-1	0.1	1	0.1	1.528	0.152	0.167	-1	0	0.670	0.112	0.121	1	0.121	1.378	0.166	1
0.6	0.6	-1	0.1	1	0.1	1.528	0.152	0.167	-1	0	0.670	0.112	0.121	1	0.121	1.378	0.166	1
0.9	0.3	-1	0.1	1	0.1	1.528	0.152	0.167	1	0.167	1.493	0.249	0.269	-1	0	0.725	0.195	1
0.3	0.1	1	0.1	1	0	0.655	0.065	0.071	1	0	0.670	0.048	0.052	-1	0.052	1.378	0.071	1
0.35	0.45	1	0.1	1	0	0.655	0.065	0.071	1	0	0.670	0.048	0.052	1	0	0.725	0.038	1
0.45	0.1	1	0.1	1	0	0.655	0.065	0.071	1	0	0.670	0.048	0.052	-1	0.052	1.378	0.071	1
0.45	0.5	1	0.1	1	0	0.655	0.065	0.071	1	0	0.670	0.048	0.052	1	0	0.725	0.038	1
0.6	0.4	1	0.1	1	0	0.655	0.065	0.071	1	0	0.670	0.048	0.052	1	0	0.725	0.038	1
			$\varepsilon_1 = 0.3$		$\alpha_1 = 0.424$		$\Gamma^{(1)} = 0.917$	$\varepsilon_2 = 0.310$		$\alpha_2 = 0.401$		$\Gamma^{(2)} = 0.925$	$\varepsilon_3 = 0.345$		$\alpha_3 = 0.321$		$\Gamma^{(3)} = 0.951$	

2 Transforming AdaBoost Weights (25 points)

We want to prove by induction that,

$$w_i^{(b+1)} = \frac{\exp(-y_i f_b(x_i))}{\sum_{j=1}^n \exp(-y_j f_b(x_i))}$$

We can actually prove the stronger statement, that,

$$w_i^{(b)} = \frac{\exp(-y_i f_{b-1}(x_i))}{\sum_{j=1}^n \exp(-y_j f_{b-1}(x_i))}$$

holds for all $b \in \{1, \cdots, B\}$, and not just for the weight updates.

Induction start: By definition $f_0 = 0$, which implies that,

$$\frac{\exp(-y_i f_0(x_i))}{\sum_{j=1}^n \exp(-y_j f_0(x_j))} = \frac{\exp(0)}{\sum_{j=1}^n \exp(0)} = 1/n$$

Which is equal to $w_i^{(1)}$.

Induction step: Assume that

$$w_i^{(b)} = \frac{\exp(-y_i f_{b-1}(x_i))}{\sum_{j=1}^n \exp(-y_j f_{b-1}(x_j))}$$

The updated weight is given by,

$$w_i^{(b+1)} = \frac{w_i^{(b)} \exp(-\alpha_b y_i h_b(x_i))}{\sum_{j=1}^n w_j^{(b)} \exp(-\alpha_b y_j h_b(x_j))}$$

Notice that,

$$\exp(-y_i f_{b-1}(x_i))\exp(-\alpha_b y_i h_b(x_i)) = \exp\left(-y_i\left(\alpha_b h_b(x_i) + \sum_{p\le b-1}\alpha_p h_p(x_i)\right)\right) = \exp\left(-y_i \sum_{p\le b}\alpha_p h_p(x_i)\right) = \exp(-y_i f_b(x_i)) \tag{1}$$

Let $c = \sum_{j=1}^n \exp(-y_j f_{b-1}(x_j))$. By the induction hypothesis, we can then write the updated weights as,

$$w_i^{(b+1)} = \frac{w_i^{(b)}\exp(-\alpha_b y_i h_b(x_i))}{\sum_{j=1}^n w_j^{(b)}\exp(-\alpha_b y_j h_b(x_j))} = \frac{\frac{\exp(-y_i f_{b-1}(x_i))}{c}\exp(-\alpha_b y_i h_b(x_i))}{\sum_{j=1}^n \frac{\exp(-y_j f_{b-1}(x_j))}{c}\exp(-\alpha_b y_j h_b(x_j))}$$

Now using (1), and noticing that the c's cancel, we have that,

$$w_i^{(b+1)} = \frac{\exp(-y_i f_b(x_i))}{\sum_{j=1}^n \exp(-y_j f_b(x_j))}$$

By the principle of simple induction, we have shown that the AdaBoost-weights can be written as,

$$w_i^{(b)} = \frac{\exp(-y_i f_{b-1}(x_i))}{\sum_{j=1}^n \exp(-y_j f_{b-1}(x_j))}$$

for all $b \in \{1, \cdots, B\}$.

Landcover Classification (35 points)

1.

Since the class label is only the landcover class corresponding to the central pixel, it is a reasonable baseline estimate to only consider the central pixel at the 6 bands and 12 timestamps, to predict the class of the central pixel. By only considering the central pixel, we drastically decrease the amount of features, and thereby speed up training time. We may however also lose some information by only considering the single pixel. If the central pixel looks similar to two classes, or perhaps is on the border between different landcover classes, the surrounding pixels could help to identify the central pixel. We therefore also lose some information by only considering the central pixel.

2.

As weak learners we consider binary decision trees. These are implemented in the `DecisionTreeClassifier` class from the `sklearn.tree` module, using the Gini index as the `criterion` parameter. We keep the rest of the parameter values default, but specify a `max_depth` of 1, 2 and 3, respectively, for use in our cross validation. For applying AdaBoost, we make use of the `AdaBoost` class from `sklearn.ensemble` with `algorithm` set to SAMME. For doing 2-fold cross-validation we make use of the `GridSearchCV` class from `sklearn.model_selection`, with `cv` set to 2, and the `param_grid` parameter set to the dictionary containing the decision trees, and the values 50, 100 and 200 for the number of boosted trees.

We obtain a training accuracy of 0.7742 and a test accuracy of 0.7194. Below we plot a confusion matrix for the test data. We normalize over the true labels, such that the rows sum to 1.

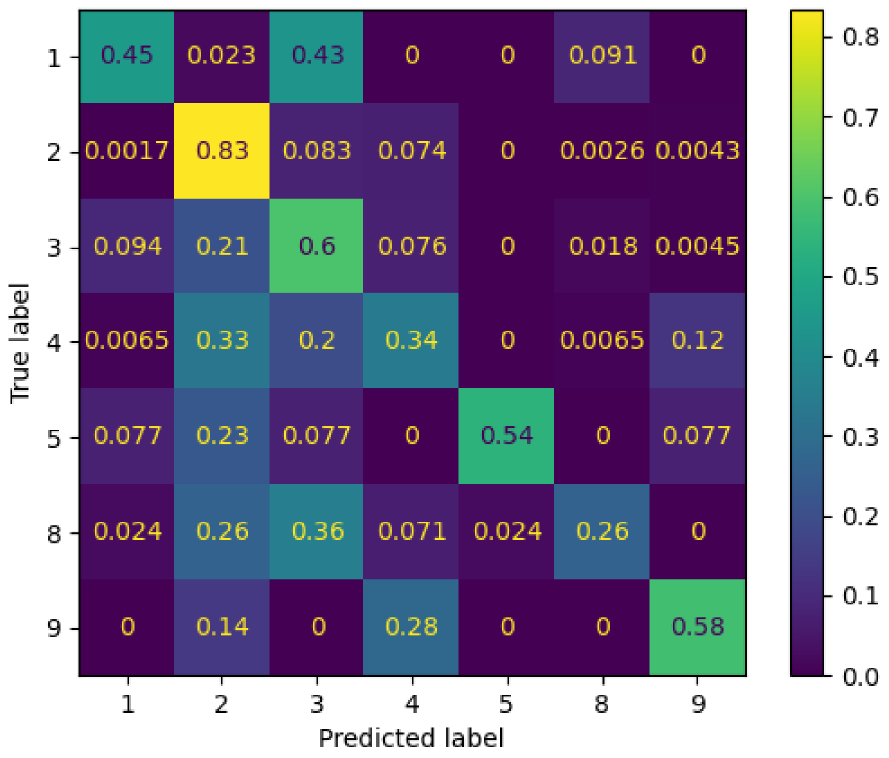

Figure 1: Confusion matrix for the test data

We notice that our model especially has a hard time predicting class 4 and 8 correctly. Interestingly, our model has a hard time distinguishing between class 1 and 3, when the true label is 1, but only predicts class 1 rarely when the true class is 3.

4 PAC-Bayes vs. Occam (15 points)

1.

Per assignment 2, we have that w.p. at least $1-\delta$ for all $h \in \mathcal{H}$.

$$\mathrm{kl}(\hat{L}(h,S)||L(h)) \leq \frac{\log \frac{1}{\pi(h)} + \log \frac{n+1}{\delta}}{n}$$

Let ρ be arbitrary. Per corollary 3.30,

$$\mathrm{kl}(\mathbb{E}_\rho[\hat{L}(h,S)]||\mathbb{E}_\rho[L(h)]) \leq \mathbb{E}_\rho[\mathrm{kl}(\hat{L}(h,S)||L(h))]$$

Combining the inequalities, we have that with probability at least $1-\delta$ for all $h \in \mathcal{H}$,

$$\mathrm{kl}(\mathbb{E}_\rho[\hat{L}(h,S)]||\mathbb{E}_\rho[L(h)]) \leq \mathbb{E}_\rho\left[\frac{\log \frac{1}{\pi(h)} + \log \frac{n+1}{\delta}}{n}\right] = \frac{\mathbb{E}_\rho\left[\log \frac{1}{\pi(h)}\right] + \log \frac{n+1}{\delta}}{n}$$

Since ρ is arbitrary, this holds for all ρ, which is what we wanted to show.

2.

By definition $\mathrm{KL}(\rho(h)||\pi(h)) = \mathbb{E}_\rho\left[\frac{\rho(h)}{\pi(h)}\right]$, but per p. 39 this can also be decomposed as,

$$\mathbb{E}_\rho\left[\frac{\rho(h)}{\pi(h)}\right] = \mathbb{E}_\rho\left[\log\frac{1}{\pi}\right] - H(\rho)$$

We then have that,

$$\frac{\mathrm{KL}(\rho||\pi) + \log\frac{n+1}{\delta}}{n} = \frac{\mathbb{E}_\rho\left[\log\frac{1}{\pi}\right] - H(\rho) + \log\frac{n+1}{\delta}}{n} \leq \frac{\mathbb{E}_\rho\left[\log\frac{1}{\pi}\right] + \log\frac{n+1}{\delta}}{n}$$

Where we have used the fact that the entropy of a discrete distribution is always non-negative. We have thus shown that the PAC-Bayes-kl ineaquality is always at least as tight as the Occam's razor bound derived in question 1.