Assignment 4

1 How to Split a Sample into Training and Test Sets (20 points)

Since $\hat{L}(\hat{h}_i^*, S_i^{\text{test}})$ is an unbiased estimate of $L(\hat{h}_i^*)$, per Hoeffdings inequality we have that for each i

$$\mathbb{P}\left(L(\hat{h}_i^*) \geq \hat{L}(\hat{h}_i^*, S_i^{\text{test}}) + \sqrt{\frac{\log m/\delta}{2n_i}}\right) \leq \frac{\delta}{m}$$

Applying a union bound we can now get that

$$\mathbb{P}\left(\exists i \in \{1, \cdots, m\} : L(\hat{h}_i^*) \geq \hat{L}(\hat{h}_i^*, S_i^{\text{test}}) + \sqrt{\frac{\log m/\delta}{2n_i}}\right) \leq \sum_{i=1}^{m} \mathbb{P}\left(L(\hat{h}_i^*) \geq \hat{L}(\hat{h}_i^*, S_i^{\text{test}}) + \sqrt{\frac{\log m/\delta}{2n_i}}\right) \leq \frac{\delta}{m} = \sum_{i=1}^{m} \frac{\delta}{m} = \delta$$

We also have that

$$\mathbb{P}\left(\exists i \in \{1, \cdots, m\} : L(\hat{h}_i^*) \geq \hat{L}(\hat{h}_i^*, S_i^{\text{test}}) + \sqrt{\frac{\log m/\delta}{2n_i}}\right) = 1 - \mathbb{P}\left(\forall i \in \{1, \cdots, m\} : L(\hat{h}_i^*) \leq \hat{L}(\hat{h}_i^*, S_i^{\text{test}}) + \sqrt{\frac{\log m/\delta}{2n_i}}\right)$$

Combining these expressions yields that,

$$\mathbb{P}\left(\forall i \in \{1, \cdots, m\} : L(\hat{h}_i^*) \leq \hat{L}(\hat{h}_i^*, S_i^{\text{test}}) + \sqrt{\frac{\log m/\delta}{2n_i}}\right) \geq 1 - \delta$$

Or in other words that simoultaneously for all $\hat{h}_i^*$,

$$L(\hat{h}_i^*) \leq \hat{L}(\hat{h}_i^*, S_i^{\text{test}}) + \sqrt{\frac{\log m/\delta}{2n_i}}$$

with probability at least $1 - \delta$.

Early stopping (30 points)

1.

(a)

We always have that $h_{t^*} = h_{100}$ independently of S_{val}. More formally $s \in S_{\text{val}}$ is exchangeable with another random variable from the distribution that generated S_{val} from the perspective of h_{t^*}. Therefore, in this case, the validation error is an unbiased estimator of the true error.

(b)

Even though the stopping time is predefined, h_{t^*} depends on S_{val}. That is, for $s \in S_{\text{val}}$ is *not* exchangeable with another random variable from the distribution that generated S_{val} from the perspective of h_{t^*}, as this might change t^*. Therefore, in this case, the validation error is a biased estimator of the true error.

(c)

With adaptive stopping neither the number of epochs nor t^* is independent of S_{val}. Per the same argument as in (b), the validation error is a biased estimator of the true error.

2. High probability bounds

We assume throughout that $|S_{\text{val}}| = n$. When we write it 'follows immediately from theorem 3.x', what we really mean is that because $\hat{L}(h_t, S_{\text{val}})$ is an unbiased estimator of $L(h_t)$ *for each individual hypothesis, when no selection is made*, we could essentially replicate theorem 3.1, 3.2, 3.3, replacing S with S_{val}.

Predefined stopping

Our hypothesis set is $\mathcal{H} = \{h_{100}\}$. It follows immediately from theorem 3.1 that with probability at least $1 - \delta$

$$L(h_{t^*}) = L(h_{100}) \leq \hat{L}(h_{100}, S_{\text{val}}) + \sqrt{\frac{\log \frac{1}{\delta}}{2n}}$$

Non-adaptive stopping

Our hypothesis set is $\mathcal{H} = \cup_{i=1}^{T} h_i$, with $|\mathcal{H}| = T$. It follows immediately from theorem 3.2 that with probability at least $1 - \delta$

$$L(h_{t^*}) \leq \hat{L}(h_{t^*}, S_{\text{val}}) + \sqrt{\frac{\log \frac{T}{\delta}}{2n}}$$

Adaptive stopping

Our hypothesis set is $\mathcal{H} = \cup_{i=1}^{\infty} h_i$. Set $\pi(h_i) = \frac{1}{i(i+1)}$. Clearly $\pi(h_i) \geq 0$ for all i, and π is independent from S and

$$\sum_{h \in \mathcal{H}} \pi(h_i) = \sum_{i=1}^{\infty} \pi(h_i) = 1$$

It follows immediately from theorem 3.3, that with probability at least $1 - \delta$

$$L(h_{t^*}) \leq \hat{L}(h_{t^*}, S_{\text{val}}) + \sqrt{\frac{\log \frac{1}{\pi(h_{t^*})\delta}}{2n}} = \hat{L}(h_{t^*}, S_{\text{val}}) + \sqrt{\frac{\log \frac{t^*(t^*+1)}{\delta}}{2n}}$$

3.

It does not make sense to continue training after the generalisation bound provided in exercise 2. exceeds one. We derive an order of magnitude $T_{\max}$ as a function of n, for which this holds. As we are dealing with an order of magnitude we disregard the validation error. We have that

$$\sqrt{\frac{\log \frac{T_{\max}(T_{\max}+1)}{\delta}}{2n}} \geq 1 \Leftrightarrow \sqrt{T_{\max}^2 + T_{\max}} \geq \sqrt{\delta} e^n$$

On the left hand side, for sufficiently large $T_{\max}$, $T_{\max}$ becomes negligable compared to $T_{\max}^2$. On the right hand side, as δ is fixed, the exponential function is dominating. We conclude that it does not make sense to continue training for $t > T_{\max}$ if

$$T_{\max} >> e^n$$

That is $T_{\max} = O(e^n)$

4.

Now setting $\pi(h_i) = 2^{-i}$, the generalisation bound for the adaptive stopping is given by

$$L(h_t) \leq \hat{L}(h_t, S_{\text{val}}) + \sqrt{\frac{\log \frac{1}{\pi(h_t)\delta}}{2n}} = \hat{L}(h_t, S_{\text{val}}) + \sqrt{\frac{\log \frac{2^t}{\delta}}{2n}}$$

With similar calculation to the ones in 3., we see that

$$\sqrt{\frac{\log \frac{2^{T_{\max}}}{\delta}}{2n}} \geq 1 \Leftrightarrow T_{\max} \geq \frac{\log(\delta) + 2n}{\log(2)}$$

As $\log(\delta)$ and $\log(2)$ are constant, we conclude that it does not make sense to continue training for $t > T_{\max}$ if

$$T_{\max} >> 2n$$

That is, the maximum number of epochs can only be twice as big as the number of samples, which is significantly less than in the previous question, where $T_{\max}$ scaled exponentially with the number of samples.

5.

We compare the adaptive procedure to the non-adaptive. We assume the procedures use the same initialisation. Assume that the adaptive procedure has reached $T_{\max}$ and returns the model corresponding to epoch $t^* \leq T_{\max}$. Let T denote the number of epochs in the non-adaptive approach, and let T^* be the model returned under the non-adaptive approach. Assume throughout that $\delta \leq \frac{1}{2}$ s.t. $\delta \leq \frac{1}{2} \leq \frac{T}{T+1}$. We also introduce the shorthand notation $\hat{L}(\cdot) = \hat{L}(\cdot, S_{\text{val}})$.

(a)

Assume that $T \leq T_{\max}$. Since the adaptive procedure returned t^* and not T^*, while the adaptive model was able to select T^*, we must have that

$$\hat{L}(h_{t^*}) + \sqrt{\frac{\log(t^*(t^*+1)/\delta)}{2n}} \leq \hat{L}(h_{T^*}) + \sqrt{\frac{\log(T^*(T^*+1)/\delta)}{2n}}$$

We have the trivial bound $T \leq T^*$, this implies

$$\hat{L}(h_{T^*}) + \sqrt{\frac{\log(T^*(T^*+1)/\delta)}{2n}} \leq \hat{L}(h_{T^*}) + \sqrt{\frac{\log(T(T+1)/\delta)}{2n}}$$

Since $\delta \leq \frac{T}{T+1}$, we also have that $\frac{T}{\delta} \geq T+1$, this now implies that

$$\hat{L}(h_{T^*}) + \sqrt{\frac{\log(T(T+1)/\delta)}{2n}} \leq \hat{L}(h_{T^*}) + \sqrt{\frac{\ln\left(\left(\frac{T}{\delta}\right)^2\right)}{2n}} = \hat{L}(h_{T^*}) + \sqrt{2}\sqrt{\frac{\log T/\delta}{2n}}$$

Further, as $\sqrt{2} \geq 1$, we have that

$$\hat{L}(h_{T^*}) + \sqrt{2}\sqrt{\frac{\log T/\delta}{2n}} \leq \sqrt{2}\left(\hat{L}(h_{T^*}) + \sqrt{\frac{\log T/\delta}{2n}}\right)$$

We recognize what is inside the parenthesis as the non-adaptive generalisation bound. Reading from start to finish, we have that

$$\hat{L}(h_{t^*}) + \sqrt{\frac{\log(t^*(t^*+1)/\delta)}{2n}} \leq \sqrt{2}\left(\hat{L}(h_{T^*}) + \sqrt{\frac{\log T/\delta}{2n}}\right)$$

That is, the adaptive bound is at most a multiplicative factor of $\sqrt{2}$ larger than the non-adaptive bound.

(b)

Now, assume that $T > T_{\max}$. If we once again use the fact that $T+1 \leq \frac{T}{\delta}$, we have for the non-adaptive bound that

$$\hat{L}(h_{T^*}) + \sqrt{\frac{\log T/\delta}{2n}} \geq \hat{L}(h_{T^*}) + \sqrt{\frac{\log T+1}{2n}}$$

As $T > T_{\max}$, we have per 3. that $T \geq e^n$, and in particular $T+1 \geq e^n$. Inserting this yields that

$$\hat{L}(h_{T^*}) + \sqrt{\frac{\log T+1}{2n}} \geq \hat{L}(h_{T^*}) + \sqrt{\frac{\log e^n}{2n}} = \hat{L}(h_{T^*}) + \sqrt{\frac{1}{2}} \geq \sqrt{\frac{1}{2}}$$

Reading from start to finish, we have obtained that

$$\hat{L}(h_{T^*}) + \sqrt{\frac{\log T/\delta}{2n}} \geq \sqrt{\frac{1}{2}}$$

That is, the non-adaptive bound is at least $\sqrt{\frac{1}{2}}$. Seeing as the true loss is bounded above by 1, we trivially have that the adaptive bound is 1 in this case, no matter what t^* is.

(c)

We have shown that when $\delta \leq \frac{1}{2}$, the adaptive bound is never much worse than the non-adaptive. Now let us consider when the adaptive bound is significantly smaller than the non-adaptive bound. Assume throughout $\delta \leq \frac{1}{2}$ (such that our previous work still holds) and $T < T_{\max}$.

Firtst, assume that $t^* = T^*$ (This is possible as we have assumed $T < T_{\max}$). Assume that $T >> T^* = t^*$. That is, we get good performance many epochs earlier than at the maximum number of epochs for the non-adaptive stopping. The validation loss terms in the bounds are identical, so we can just consider

$$\sqrt{\frac{\log(t^*(t^*+1)/\delta)}{2n}} \quad \text{v.s} \quad \sqrt{\frac{\log T/\delta}{2n}}$$

For sufficiently large T and sufficiently small t^*, we can have that the adaptive bound is much smaller than the adaptive bound.

Now consider the case where the validation loss is large for all $t \leq T$, but there is a substantial increase in performance for $T_{\max} \geq t > T$. Then $\hat{L}(h_{T^*}) >> \hat{L}(h_{t^*})$. Notice that the increase in cost for doing another epoch, is relatively small for when the number of epochs is already large due to the logarithm. Thereby the performance term in the generalisation bounds dominate in this case, and the adaptive bound can be significantly smaller than the non-adaptive. Conclusively, the adaptive bound can be significantly smaller than the non-adaptive, when T is chosen such that either too few or too many epochs are run.

4 Random Forests (50 points)

4.1 Analysing Satelite Data (35 points)

1.

We train a random forest with 10 trees and bootrstrap samples using the training data. At every node we test all features, using the Gini index as our impurity measure. For building the trees, we do not impose restrictions on the depth of the trees.

We first load in the data and split it into labels and features

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

S_train = np.genfromtxt("landsat_train.csv", delimiter=",")
S_val = np.genfromtxt("landsat_validation.csv",delimiter=",")
S_area = np.genfromtxt("landsat_area.csv", delimiter = ",")

S_train_labels = S_train[:,0] # Get labels of train set
S_val_labels = S_val[:,0] # Get labels of validation set
S_train_features = S_train[:,1:] # Get features of train set
S_val_features = S_val[:,1:] # Get features of validation set
```

We train the model using the `fit` method from the `RandomForestClassifier` class from `sklearn.ensemble`.

```
#Initialize random forest with 10 trees, no max depth, bootstrap,
# gini index, and consider all features at split
RF = sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion="gini", bootstrap=True,
 max_depth=None,max_features=None)
RF.fit(S_train_features, S_train_labels) #Fit the model to data
```

And finally we compute the validation accuracy using the `score` method

```
#Validation accuracy
RF.score(S_val_features, S_val_labels)
```

We get that the validation accuracy is 0.75335.

2.

We apply the model to all instances from `landsat_area.csv` using the `predict` method, and visulize the results with each label encoded as a color,

```
area_predict = RF.predict(S_area) #Predict on data
mat = np.reshape(area_predict,(3000,3000)) #Reshape results into 3000x3000 matrix
plt.imshow(mat) #plot matrix
plt.axis('off') #axis are nonsensical in this case
```

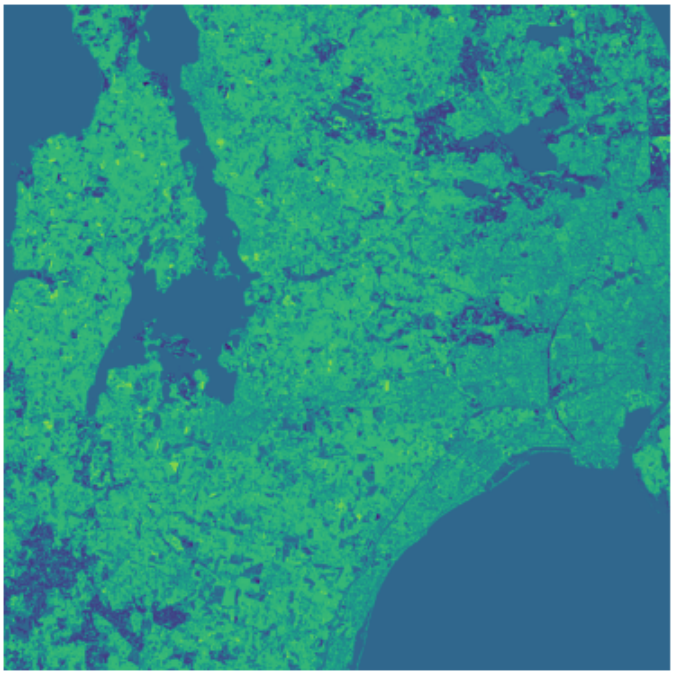

Figure 1: Area predicted by the model

The following is an image of the Roskilde Fjord and Køge Bugt area taken from Google Maps

Figure 2: Google Maps image of Roskilde Fjord and Køge Bugt

Which bears a striking resemblance to the plot of our predicted area.

3.

Assume that we are given n points and we built a full binary decision tree based on these points. We claim that the maximum depth of this tree is $n-1$. We proof this by (simple) induction after n.

Induction start: Assume that $n = 1$, which is the smallest interesting case. Seeing as there is only a single data point to grow the tree from, we only have a single node. Therefore the single leaf node and the root node are the same. That is, the depth is 0.

Induction step: Assume that for n data points the tree has maximum depth $n-1$. Now consider adding a datapoint, such that we have $n+1$ data points. Then, each node can split at most one more time. That is, the maximum depth of the new tree is n, which completes the induction step.

By the principle of simple induction the maximum depth of a full binary decision tree built on n data points is $n-1$.

4.2 Normalization (15 points)

Nearest neighbor classfication

Recall that we can rescale a component to 0 mean and unit variance, by subtracting the mean of the component and dividing by the standard deviation of the component. Consider observing

$$x_1 = (1000, 2) \quad \text{and} \quad x_2(2000, 1)$$

with labels $y_1 = 1$ and $y_2 = -1$. If we now observe a new data point $x = (1600, 1.8)$ and would like to predict its label, according to the nearest neighbor algorithm, we would calculate the distances

$$d(x, x_1) = \sqrt{600^2 + 0.2^2} \approx 600$$
$$d(x, x_2) = \sqrt{400^2 + 0.8^2} \approx 400$$

And we would assign the label $y = -1$ to x based on the nearest neighbor algorithm. Now consider scaling the components to unit variance and 0 mean, then

$$\tilde{x}_1 = (-0.707, 0.707) \quad \text{and} \quad \tilde{x}_2(0.707, -0.707)$$

And $\tilde{x} = (0.141, 0.424)$. Computing the distances we get that

$$d(\tilde{x}, \tilde{x}_1) = 0.8$$
$$d(\tilde{x}, \tilde{x}_2) = 1.6$$

And now the nearest neighbor algorithm would predict the label $y = 1$ for x. That is, the nearest neighbor classifier is affected by scaling the components to 0 mean and unit variance.

Random Forest Classification

Random forest classification is not affected by rescaling components to unit variance and 0 mean, because only one feature is considered at each node. More precisely, when a tree is built, each inner node is only associated with one component/feature d and a threshhold θ. The split of the node is then decided by how x_d compares to θ. Clearly scaling x_d to $\tilde{x}_d$ and thereby also θ to $\tilde{\theta}$, does not change whether $\tilde{x}_d$ is larger or smaller than $\tilde{\theta}$. Therefore classification trees are invariant under scaling to unit variance and 0 mean. As a random forest is made up of individual trees, also the random forest classifier will be invariant under scaling to unit variance and 0 mean.