CS 4033/5033: Machine Learning
Assignment 6, Fall 2017

You will perform feature selection on a dataset from the UCI (University of California at Irvine) Machine-Learning Repository. The particular dataset comes from the MAGIC (Major Atmospheric Gamma-Imaging Cherenkov) telescope and can be found here. The task is to distinguish signals produced by a gamma particle from those produced by a hadron.

Throughout this assignment, please let gamma (γ) be the positive class (label = 1) and hadron (H) be the negative class. If you all use the same convention, your assignments will be easier to grade.

There are 19 020 examples and 10 predictor variables, all real-valued. You can find descriptions of the 10 predictor variables, descriptions of the two labels (γ and H), and other metadata here.

All students must answer Question 4. If you're in CS 4033, pick one of the first 3 questions to answer. If you're in CS 5033, pick 2 of the first 3 questions to answer. You may do one of the additional questions for extra credit (note that this is a lot of extra credit!). However, if you're in CS 4033, you cannot do 2 questions for extra credit.

All feature-selection methods in this assignment require an underlying machine-learning model (*i.e.*, they are "wrapper methods," as opposed to "filter methods," which do not require a machine-learning model). Your underlying model should be a feed-forward neural network (the kind we discussed in class). You don't have to implement the neural net yourself – you can use any publicly available library. We suggest using `sklearn.neural_network.MLPClassifier` in Python. **However, the feature-selection must be your own.**

You can use the default hyperparameters for the neural net – this assignment is not about tuning hyperparameters. If you decide not to use the default hyperparameters (*e.g.*, if you change the learning rate, number of layers, layer sizes, etc.), please justify your decision. You will not lose points for choosing the "wrong" hyperparameters, as long as they don't completely break your learning. For example, if you choose a learning rate of 1000 (the default is 0.001) or layer size of 2 nodes (the default is 100), this will break your learning and you will lose points. But as

long as your hyperparameters are reasonable, feel free to use whichever values you want.

The loss function used throughout this assignment will be cross-entropy (Equation 1). N is the number of examples; y_i is the true label for the i^{th} example (1 if γ-particle, 0 if hadron); and f_i is the predicted probability that the i^{th} example is a γ-particle.

$$\epsilon = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log_2(f_i) + (1 - y_i) \log_2(1 - f_i)] \tag{1}$$

Before answering any of the questions, split your data into a training, validation, and testing set. You may assume that all examples are mutually independent, so you can do this randomly. Use 50% of the data for training, 30% for validation, and 20% for testing.

1. **(42.5 points) Sequential forward selection**

   The procedure for sequential forward selection (SFS) is described below.

   (a) Initialize $X_{selected}$ as an empty set.

   (b) Initialize $X_{remaining}$ as the set of all 10 features.

   (c) For each feature in $X_{remaining}$, train a neural net using only this feature. Find the cross-entropy (Equation 1) of each trained neural net **on the validation data** (but use only the training set to train the neural net). Find the neural net with the lowest cross-entropy ϵ_{min}. Find the feature x^* that gave you this lowest cross-entropy. Add x^* to $X_{selected}$ and remove it from $X_{remaining}$.

   (d) Now you have 9 features left in $X_{remaining}$. For each x_j in $X_{remaining}$, train a neural net using only $X_{selected} + \{x_j\}$. In other words, use x_j plus the features that have already been selected. Find the cross-entropy of each trained neural net on the validation data. Find the neural net with the lowest cross-entropy ϵ_{min}. Find the feature x^* that gave you this lowest cross-entropy. Add x^* to $X_{selected}$ and remove it from $X_{remaining}$.

   (e) Now you have 8 features left in $X_{remaining}$. For each x_j in $X_{remaining}$, train a neural net using only $X_{selected} + \{x_j\}$. Find the neural net with the lowest cross-entropy ϵ_{min} on the validation data. Find the feature x^* that

gave you this lowest cross-entropy. Add x^* to $X_{selected}$ and remove it from $X_{remaining}$.

(f) Keep going until ϵ_{min} no longer decreases by more than 1% from one step to the next. For example, if you reach the step with 6 features in $X_{selected}$ and find that you can no longer decrease ϵ_{min} by more than 1% by adding any of the 4 features in $X_{remaining}$, stop with the 6 features in $X_{selected}$. These are your **selected features**.

Once you've finished SFS and have your $X_{selected}$, report the cross-entropy of the corresponding neural net **on validation data only**. Also report the following:

- Cross-entropy on testing data
- Area under the ROC curve (AUC) on validation data. You may use `sklearn.metrics.roc_auc_score`, or any other publicly available method, to calculate this. You can also use the method in my answer key for Assignment 5.
- AUC on testing data

Report these 4 values ($\epsilon_{validation}$, $\epsilon_{testing}$, $\text{AUC}_{validation}$, and $\text{AUC}_{testing}$) in your write-up. Also, tell me which feature was selected at each step. For example, you could say "`fLength` was selected at step 1; `fWidth` was selected at step 2; `fM3Long` was selected at step 3; `fAlpha` was selected at step 4; then the algorithm stopped".

2. **(42.5 points) Sequential backward selection**

The procedure for sequential backward selection (SBS) is described below.

(a) Initialize $X_{selected}$ as the set of all 10 features.

(b) Train a neural net using all 10 features. Find the cross-entropy **on the validation data only**. Let this cross-entropy be ϵ_{min} (we need a starting value).

(c) Now you have 10 features left in $X_{selected}$. For each x_j in $X_{selected}$, train a neural net using all features in $X_{selected}$ except for x_j. In other words, train a neural net using the set $X_{selected} - \{x_j\}$. Find the cross-entropy (Equation 1) of each trained neural net **on the validation data** (but use

only the training set to train the neural net). Find the neural net with the lowest cross-entropy ϵ_{min}. Find the feature x^* that gave you this lowest cross-entropy. Remove x^* from $X_{selected}$.

(d) Now you have 9 features left in $X_{selected}$. For each x_j in $X_{selected}$, train a neural net using the features $X_{selected} - \{x_j\}$. Find the cross-entropy of each trained neural net on the validation data. Find the neural net with the lowest cross-entropy ϵ_{min}. Find the feature x^* that gave you this lowest cross-entropy. Remove x^* from $X_{selected}$.

(e) Keep going until ϵ_{min} no longer decreases by more than -1% from one step to the next (in other words, ϵ_{min} is allowed to **increase**, but by no more than 1%, from one step to the next). For example, if you reach the step with 3 features in $X_{selected}$ and find that you can no longer decrease ϵ_{min} by more than -1% by removing any of the 3 features, stop with the 3 features in $X_{selected}$. These are your **selected features**.

Once you've finished SBS and have your $X_{selected}$, report the following values for the corresponding neural net (the one trained on $X_{selected}$):

- Cross-entropy on validation data
- Cross-entropy on testing data
- AUC on validation data
- AUC on testing data

Also, tell me which feature was removed at each step. For example, you could say "`fAsym` was removed at step 1; `fAlpha` was removed at step 2; `fM3Trans` was removed at step 3; then the algorithm stopped".

3. **(42.5 points) Permutation selection**

The procedure for permutation selection is described below.

(a) Initialize $X_{permuted}$ as an empty set.

(b) Initialize $X_{remaining}$ as the set of all 10 features.

(c) Train a neural net using all 10 features. Find the cross-entropy **on the validation data only**. Let this cross-entropy be ϵ_{max} (we need a starting value).

(d) For each feature x_j in $X_{remaining}$, train a neural net with values of x_j randomly shuffled. (In other words, randomly permute the x_j-values of all 9510 training examples.) Find the cross-entropy (Equation 1) of each trained neural net **on the validation data** (but use only the training set to train the neural net). Find the neural net with the highest cross-entropy ϵ_{max}. Find the feature x^* that gave you this highest cross-entropy. Add x^* to $X_{permuted}$ and remove it from $X_{remaining}$.

(e) Now you have 9 features left in $X_{remaining}$. For each x_j in $X_{remaining}$, train a neural net with values of x_j randomly shuffled. Find the cross-entropy of each trained neural net on the validation data. Find the neural net with the highest cross-entropy ϵ_{max}. Find the feature x^* that gave you this highest cross-entropy. Add x^* to $X_{permuted}$ and remove it from $X_{remaining}$.

(f) Keep going until you have only one feature left in $X_{remaining}$.

At the beginning of each step, values should be shuffled only for the features in $X_{permuted}$. For the features in $X_{remaining}$, values should be in the correct order (even though you shuffled all of them at the last step). At each step you are training the neural net with 10 features. The only difference is which features are permuted.

Report the following for each step of the permutation algorithm:

- Which feature was added to $X_{permuted}$ (*i.e.*, which feature was permuted permanently)
- The corresponding ϵ_{max}

Also, report the ϵ obtained by training the neural with all 10 features non-permuted (this is the ϵ_{max} from step 3).

4. **(15 points) Conceptual questions**

(a) **(3 points)** Sequential forward selection (SFS) is usually faster than sequential backward selection (SBS). Why?

(b) **(3 points)** When the underlying ML algorithm is something simple (like linear regression), SFS is usually more stable than SBS. Why?

(c) **(3 points)** SFS keeps going until cross-entropy no longer decreases by $\geq$ 1%. Why not just say that cross-entropy has to keep decreasing, regardless

of the amount? (Hint: this usually leads to fewer features in $X_{selected}$. Why would you want a model with fewer features?)

(d) **(3 points)** SBS keeps going until cross-entropy no longer decreases by $\geq$ -1%. In other words, if cross-entropy **increases** by 0.9%, SBS keeps going. Why? (Hint: the answer is very similar to the above.)

(e) **(3 points)** The "permutation selection" algorithm given in Question 3 only ranks features, rather than explicitly selecting them. How could you tweak the algorithm to make it explicitly select a subset of features?