Lab Class ML:IV

Exercise 1 : Perceptron Learning

Solve the following problems on learning boolean functions with perceptrons. Use the values 0 for *false* and 1 for *true*, and the threshold function $\varphi(x) = \max(sign(x), 0)$.

(a) Design a single perceptron with two inputs x_A and x_B. This perceptron shall implement the boolean formula $A \land \neg B$ with a suitable function $y(x_A, x_B)$. Hint: to start, determine the training data and draw it, and a suitable decision boundary, in a coordinate system; then, determine a set of suitable weights $\mathbf{w} = (w_0, w_1, w_2)$.

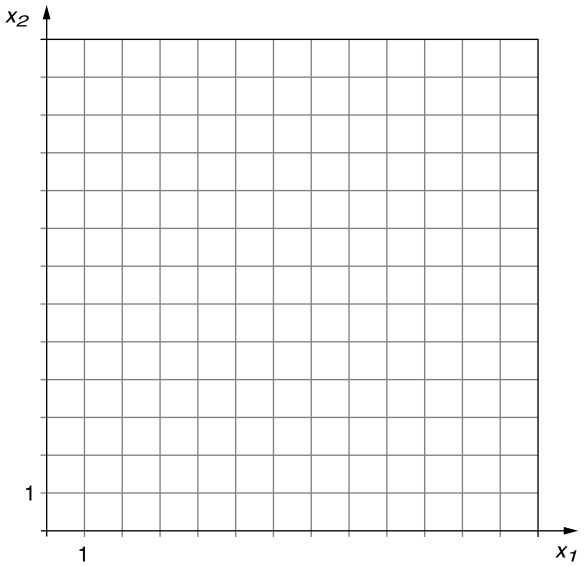
(b) Why can the boolean formula A *XOR* B not be learned by a single perceptron? Justify your answer with a drawing.

Exercise 2 : Perceptron Learning

Given is the following training data:

Number	1	2	3	4	5	6	7	8	9	10	11	12
x_1	1	7	7	2	3	2	5	10	11	12	10	9
x_2	3	4	6	6	2	1	7	3	5	8	5	8
$c(\mathbf{x})$	1	0	1	1	0	0	1	0	0	1	0	1

(a) Draw the data into a coordinate system like the one below.



(b) The classes can be separated with straight lines. Determine such a straight line and specify its linear equation in the following form: $w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2 = 0$

(c) Add the values for the left hand side of the equation to the following table. Do you spot a pattern?

Number	1	2	3	4	5	6	7	8	9	10	11	12
x_1	1	7	7	2	3	2	5	10	11	12	10	9
x_2	3	4	6	6	2	1	7	3	5	8	5	8
$c(\mathbf{x})$	1	0	1	1	0	0	1	0	0	1	0	1
$w_0 + w_1x_1 + w_2x_2$												

(d) Given a perceptron that is initialized with the weights $w_0 = 1, w_1 = -1, w_2 = 3$. Which straight line belongs to this perceptron and how is it used for classification?

(e) Assuming the same initial weights, which weight changes are made by the incremental perceptron training algorithm (PT) if at the learning rate $\eta = 1$, the whole dataset is presented in order three times? Compute the first three weight vectors by hand, and solve the remaining 33 iterations with a computer (you may write a short program for this, but note that a simple spreadsheet would also suffice). Use the following table as a guide:

Iteration	x_0	x_1	x_2	$c(\mathbf{x})$	w_0	w_1	w_2	$y(\mathbf{x})$	δ
1	1	1	3	1	1	-1	3	?	?
2	1	7	4	0	?	?	?	?	?
3	1	7	6	1	?	?	?	?	?
⋮									
13	1	1	3	1	?	?	?	?	?
⋮									
36	1	9	8	1	?	?	?	?	?

Exercise 3 : Decision Trees

Construct by hand a decision tree corresponding to each of the following Boolean functions. The examples $\mathbf{x} \in D$ have one attribute for each Boolean variable A, B, ... in the formula; the target concept $c(\mathbf{x})$ is the truth value of the formula itself. Assume the set D contains examples with all possible combinations of attribute values.

Hint: It may be helpful to write out the set D, i.e., the truth table for the variables and formula, first.

(a) $A \wedge \neg B$

(b) A *XOR* B

(c) $A \vee (B \wedge C)$

(d) $(A \wedge B) \vee (C \wedge D)$

(e) $(A \vee B) \wedge (C$ *XOR* $D)$

Exercise 4 : Decision Trees

Given the following training set with dogs data:

Color	Fur	Size	Class
brown	ragged	small	well-behaved
black	ragged	big	dangerous
black	smooth	big	dangerous
black	curly	small	well-behaved
white	curly	small	well-behaved
white	smooth	small	dangerous
red	ragged	big	well-behaved

(a) Use the ID3 algorithm to determine a decision tree, where the attributes are to be chosen with the maximum average information gain *iGain*:

$$\textit{iGain}(D, A) \equiv H(D) - \sum_{a \in A} \frac{|D_a|}{|D|} \cdot H(D_a) \quad \text{with} \quad H(D) = -p_{\oplus} \log_2(p_{\oplus}) - p_{\ominus} \log_2(p_{\ominus})$$

Thoroughly explain the steps of your calculations.

(b) Classify the new example (Color=black, Fur=ragged, Size=small) using your decision tree.

Exercise 5 : Overfitting

(a) What is overfitting?

(b) Overfitting can have a variety of reasons, the main reason being the inductive bias of the model. Explain the relationship between inductive bias and overfitting.

(c) Give at least three examples of how overfitting can be reduced and inductive bias can be controlled.

Exercise 6 : $\boxed{\text{P}}$ Multi-Class Web Page Classification with Neural Networks

In this exercise, we will revisit the web page function classification task from the previous exercise. Using the more complex multilayer perceptron model you have now gotten to know, we will try to distinguish all six classes.

(a) Download and read in the provided training dataset. As before, the package `pandas` will prove useful. The resulting dataframe has a column "`main-function`" that represents the class attribute $c(\mathbf{x})$, and several columns whose names are prefixed with "`feature_`" – these represent the attributes of the input vectors $\mathbf{x}$. Bring the data into the correct format for the unrestricted classification problem setting as described for the Multilayer Perceptron in the lecturenotes. Pay particular attention to the following points.

- The class attribute $c(\mathbf{x})$ is given in the provided dataset as one of the $k = 6$ possible values {discuss, entertain, form, inform, link, sell}. Turn these class values into vectors from the set $C = \{0, 1\}^k$ as required by the multilayer perceptron. To do so, decide on an ordering of the possible class values (for example, the one given two sentences back), and map the first value to the vector `[1,0,0,0,0,0]`, the second to `[0,1,0,0,0,0]`, and so on.
- The attributes of the input vectors $\mathbf{x}$ have very different scales. Use a feature scaling technique from the previous programming exercise to normalize them.
- Remember to ensure that each input vector $\mathbf{x}$ gets an extra attribute $x_0 = 1$.

(b) Implement the IGD_{MLP} algorithm from the lecture slides. You may use the numpy implementation discussed during lab class as a basis. Note the following points.

- The `predict()` method should return one of the 6 possible class values `['discuss', ..., 'sell']`. To do so, select the class value where the network output vector $\mathbf{y}(\mathbf{x})$ is maximal.
- In the `fit()` method, pay attention to the fact that each `cx` is now also a vector.

(c) Choose values for the hidden layer size l, the learning rate η, and number of iterations t_{max}, and train a multilayer perceptron model.

(d) Using the `predict()` method, generate a list of predictions for the examples in the training dataset (I.e., you should end up with a list or array of 2 057 elements, each of which is one of the six possible classes {discuss, entertain, form, inform, link, sell}). The column "`main-function`" from the training dataset contains the list of true classes for all 2 057 examples. Using both of the aforementioned lists, compute the training error of your classifier.

(e) Modify your implementation of the `fit()` method to use the batch gradient descent (BGD) algorithm, instead of incremental gradient descent. To do so, discard the inner loop over individual examples in the dataset; instead, use matrix operations to process the entire dataset as a single batch. Inspect any differences in training speed, and classification error.

Bonus: Challenge

(This exercise is not mandatory)

As in the previous exercise, optimize and tweak your implementation using everything you have learned. Once satisfied with the performance, classify the examples in the separate test dataset, and submit your classifications as a JSON file.

Hints:

- Evaluating your model only on the training set will not be enough to get a realistic estimate of your model's performance on unseen data. Consider splitting off a holdout set, or using cross-validation. The library functions `sklearn.model_selection.StratifiedShuffleSplit` or `sklearn.cross_validation.StratifiedKFold` may be helpful here.
- Likewise, the average misclassification rate alone may be insufficient to judge the quality of your model in the k-class setting. Consider using the library function `sklearn.metrics.classification_report` for further insights.
- If time permits, experiment with using a decision tree model instead.
- Results should be submitted as a `pandas.DataFrame` written to a JSON file. It should contain the columns "page-id" and the corresponding "prediction." To write the dataframe to file, use: `your_data_frame.to_json(filename, orient="records")`! The filename should be: "test_set_prediction_[group].json", where [group] is **your** group number, for example "L01" for Leipzig or "W6-13" for Weimar.