Lab Class ML:III

Exercise 1 : Properties of the Sigmoid Function

This exercise regards some mathematical properties of the sigmoid function σ, which make it very suitable for machine learning.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

(a) Show that $\sigma(-x) = 1 - \sigma(x)$.

(b) For the inverse of the sigmoid function, show that if $y = \sigma(x)$, then $x = \sigma^{-1}(y) = \ln(y/(1-y))$.

(c) Show that the derivative of the sigmoid function is $\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$.

Note: This property is especially useful if you have calculated a sigmoid output, and also need to calculate its gradient, for example in a neural network environment.

Exercise 2 : Pointwise Loss Functions

In the lecturenotes, the rightmost plot on slide ML:III-38 shows the pointwise squared loss for a linear regression model, for the case where $c(\mathbf{x}) = 1$, i.e. $\mathbf{x}$ is a positive example. Slide ML:III-43 shows the pointwise logistic loss for a logistic regression model, also for the positive case.

Your task is to investigate the corresponding case for negative examples, i.e. $c(x) = -1$ for linear regression, and $c(x) = 0$ for logistic regression.

(a) For the linear regression model, let $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. Write down the formula for $\ell_2(y(\mathbf{x}), -1)$, where ℓ_2 is the pointwise squared loss, and $\mathbf{x}$ is a negative example.

(b) Draw a plot for the squared loss for negative examples, with $\mathbf{w}^T\mathbf{x}$ on the x-axis, and the squared loss on the y-axis. Add the zero/one-loss to your plot, as well.

(c) For the logistic regression model, let $z(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ and $y(\mathbf{x}) = \sigma(z(\mathbf{x})) = \frac{1}{1+e^{-z(\mathbf{x})}}$. Write down the formula for $\ell_\sigma(z(\mathbf{x}), 0)$, where ℓ_σ is the logistic loss, and $\mathbf{x}$ is a negative example.

(d) Draw a plot for the logistic loss for negative examples, with $\mathbf{w}^T\mathbf{x}$ on the x-axis, and the logistic loss on the y-axis. Add the zero/one-loss to your plot, as well.

Exercise 3 : Overfitting

Let the following simple classification algorithm called "1NN" be defined for a binary classification setting where the feature space $X \subseteq \mathbf{R}^p$ is an inner product space, and $d : X \times X \to [0, \infty)$ is an arbitrary metric or distance function.

Hint: the Wikipedia article on metrics defines the properties of such a function.

Algorithm: 1NN-Learn.
Input: Dataset $D_{tr} = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\}$ and distance function d.

1NN-Learn(D_{tr}, d)
1. **Store:** save D_{tr} and d for future use, otherwise do nothing.

Algorithm: 1NN-Predict.
Input: Dataset $D_{tr} = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\}$ distance function d, and unknown point $\mathbf{x}^{(?)}$.
Output: Classification for $\mathbf{x}^{(?)}$
1NN-Predict$\left(D_{tr}, d, \mathbf{x}^{(?)}\right)$
1. **Find closest known point:** $(\mathbf{x}^*, c(\mathbf{x}^*)) := \mathsf{argmin}_{(\mathbf{x}, c(\mathbf{x})) \in D_{tr}} \quad d(x, x^{(?)})$
2. **Return** $c(\mathbf{x}^*)$

Suppose that our dataset D contains each feature vector $\mathbf{x}$ at most once. We divide D into equal-sized training and validation sets, i.e., we have $D = D_{tr} \cup D_{val}$ and $|D_{tr}| = |D_{val}|$.

(a) Without knowing any further specifics about the dataset, what do you know about the training error $Err_{tr}(y)$, when y is a 1NN-classifier as defined above?

(b) We train both a 1NN-classifier, and a logistic regression classifier in this setting, with the following results: For logistic regression, we get $Err_{tr}(y) = 0.2$ and $Err(y, D_{val}) = 0.3$. For the 1NN-classifier, we get an *average* training and val error of $\frac{Err_{tr}(y)+Err(y,D_{val})}{2} = 0.18$. Which of the two classification algorithms should we prefer based on these results, and why?

Exercise 4 : Regularization

Suppose we are estimating the regression coefficients in a linear regression model by minimizing the objective function $\mathcal{L}$.

$$\mathcal{L}(\mathbf{w}) = \mathsf{RSS}_{tr}(\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}$$

The term $\mathsf{RSS}_{tr}(\mathbf{w}) = \sum_{(x_i, y_i) \in D_{tr}} \left(y_i - \mathbf{w}^T \mathbf{x}_i\right)^2$ refers to the residual sum of squares computed on the set D_{tr} that is used for parameter estimation. Assume that we can also compute an RSS_{test} on a separate set D_{test} that we don't use during training.

When we vary the hyperparameter λ, starting from 0 and gradually increase it, what will happen to the following quantities? Explain your answers.

(a) The value of $\mathsf{RSS}_{tr}(\mathbf{w})$ will...
- ☐ remain constant.
- ☐ steadily increase.
- ☐ steadily decrease.
- ☐ increase initially, then eventually start decreasing in an inverted U shape.
- ☐ decrease initially, then eventually start increasing in a U shape.

(b) The value of $\mathsf{RSS}_{test}(\mathbf{w})$ will...
- ☐ remain constant.
- ☐ steadily increase.
- ☐ steadily decrease.
- ☐ increase initially, then eventually start decreasing in an inverted U shape.
- ☐ decrease initially, then eventually start increasing in a U shape.

Exercise 5 : [P] Implementation of Basic Learning Algorithms

This exercise is about implementing regression models. We will provide you with a data set of the data you collectively labeled in the first exercise sheet and a subset of 5 feature columns. This exercise will cover implementing and evaluating linear and logistic regression models.

For simplicity: We consider a binary classification and restrict the task to trying to predict if the main function of a webpage is to "sell" or not to "sell."

(a) Download and read in the data. (The package `pandas` may be useful here!) The data contains the target variable annotated during the first exercise sheet and the values of some of the feature functions implemented by you, namely: 'number_of_ads,' 'linkration,' 'numberOfListItems,' 'FractionInput,' 'input.'

(b) To evaluate the model performance, we need a measurement of how well the classification works on the given data: Implement a Python function to compute the misclassification error (compare ML:II-98). The signature should look as follows:

```
def misclassification_rate(truth, predictions):
    """Given two one-dimensional arrays `truth` and `predictions`
    of the same length, compute the misclassification rate.
    """
    # ... your code here ...
```

This function can then serve as an evaluation for the following implementation of the learning algorithm.

(c) The features you have implemented may have vastly different value scales. This can be hurtful to your learning algorithm, especially gradient methods. For details and a rationale on how that influences the training process, you can watch this video by Andrew Ng. Implement at least one of the feature scaling methods explained in the video.

(d) Linear Regression: Consider the linear hypothesis $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, denoting a function y that takes a vector $\mathbf{x}$ as an input and has parameter vector $\mathbf{w}$. Implement the Least Mean Squares (LMS) algorithm from the lecture to estimate the weights $\mathbf{w}$.

(e) Next consider the logistic regression hypothesis $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$ (ML:III). Implement the Batch Gradient Descent (BGD) algorithm to determine the parameter vector $\mathbf{w}$ by minimizing the logistic loss.

(f) Now we have implemented two hypotheses to fit the set of training data we have. To decide which one might work better in general, we have to estimate the performance on data that is not used to estimate the parameters. To find out, implement a k-fold cross-validation procedure.
Evaluate the average misclassification error for both models on a 5-fold cross-validation to decide which of the models is better suited to the data and the feature set at hand.

Hints: We will provide you with a notebook containing some code fragments and useful hints. The two packages to help you implement this are `pandas` and `numpy`.

Bonus: Challenge

(This exercise is not mandatory!)

We will also provide a separate test data set (without the correct predictions). After implementing the regression algorithms, the challenge is to train the best classifier you can, make predictions on the test set

and then submit the results to Moodle. We will evaluate the predictions and create a Leaderboard. It is important that the submission conforms to the specified format.

Format:

- Results should be submitted as a `pandas.DataFrame` written to a JSON file. It should contain the columns “page_id” and the corresponding “prediction.”
- The column “prediction” should contain either “mainly” or “not” as per your models prediction.
- The test data set is a table containing the “page_id” and features. These “page_ids” are used to match your prediction with the solution data.
- 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.

Hints: The quality of the model depends on a lot of hyperparameters and settings you can experiment with to get the best results, e.g. learning rate, feature scaling method, number of iterations, etc.