Machine Learning, Fall 2019: Project 2
out on 9/24/19 - due on 10/10/19 by noon on Blackboard

Anastasija Mensikova

Header: Resources used for the completion of the project:

- Python programming language
- Jupyter Notebook
- Perceptron description:
  https://medium.com/@thomascountz/19-line-line-by-line-python-perceptron-b6f113b161f3
- Lecture slides
- MatPlotLib for plots

1 Perceptron: Iris flower classification

This version of the iris data set contains 2 classes of 50 instances each, where each class refers to a type of iris plant. The task is to select if a given flower is Iris-setosa or Iris-virginica. This Perceptron is implemented using 10-fold-stratified-cross-validation.

The step-by-step execution of the algorithm can be seen in the Jupyter Notebook attached. However, in short, the main aspect of this algorithm involves calculating weights for each input vector, which is done by multiplying the input vector by the weights (that are initially non-zero, that is, 1.0), and including the learning rate and the error calculated by the prediction. After that the weights are passed through the prediction, which acts as the activation function, which essentially uses the threshold of 0.0 and checks if the activated results passes that threshold. It is, however, important to note that all of that is performed on 10-fold-stratified-cross-validated data, and the species names of Iris-setosa and Iris-virginica are changed to integer values of 0 and 1, respectively.

1. **(20 points)** What happens when the learning rate is 0.00005, 0.001, and 0.005?

   As it can be observed in my Jupyter Notebook, increasing the learning rate generally tends to increase the total accuracy of the classifier, as it can be seen below:

   - 0.00005 – accuracy=50%
   - 0.001 – accuracy=100%
   - 0.005 – accuracy=97%

   The purpose of a learning rate is to control how dramatically weights get updated with each epoch. This explains why accuracy remains very low and steady for a very low learning rate, and improves with a higher learning rate.

2. **(20 points)** Does the algorithm converge? Plot the classification accuracy for each learning rate from 1 to 20 training epochs.

   As it can be seen in the Figures 1 through 3 below, when the learning rate is 0.001 (Figure 2) and 0.005 (Figure 3) the algorithm does converge, and it doesn't when the learning rate is extremely low, 0.00005 (Figure 1). It, however, is important to note that the algorithm does seem to overfit with the highest learning rate of 0.005 after 3 epochs. The learning rate of 0.001 seems to be ideal for this specific case.

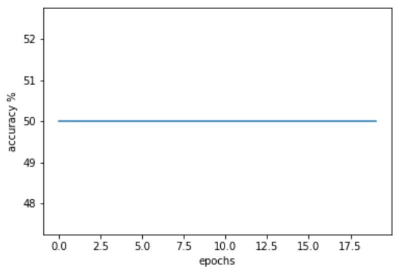

Figure 1: 0.00005

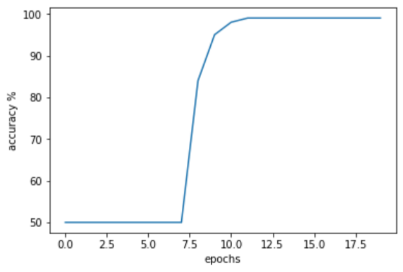

Figure 2: 0.001

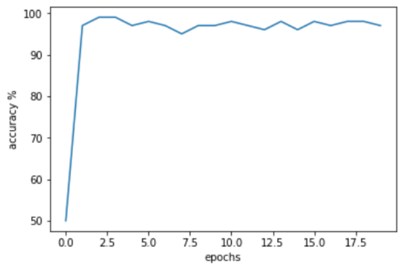

Figure 3: 0.005

However, early stopping could be implemented as 11 epochs seems to be enough to achieve almost perfect accuracy.

3. **(20 points)** Confidence

   The activation function, as it can be observed in more detail in the attached Jupyter Notebook, performs a multiplication of the given weights by the given sample in the training data. If the result of that passes the threshold then the sample is classified ”positively”, and ”negatively” otherwise. The confidence value is thereafter calculated as the countervalue of the error value. As training continues with more epochs the confidence value increases.

4. **(20 points)** Is this dataset linearly separable? Justify your answer with a scatterplot. Explain why and how you created this scatterplot.

The classification results, given their sepal length and sepal width, are plotted in the scatter plot below (Figure 4). From the first glance the dataset doesn't look linearly separable. However, considering some minor errors and the general good accuracy of this particular classification we can say that it is, in fact, linearly separable. More details can be seen in the Jupyter Notebook.

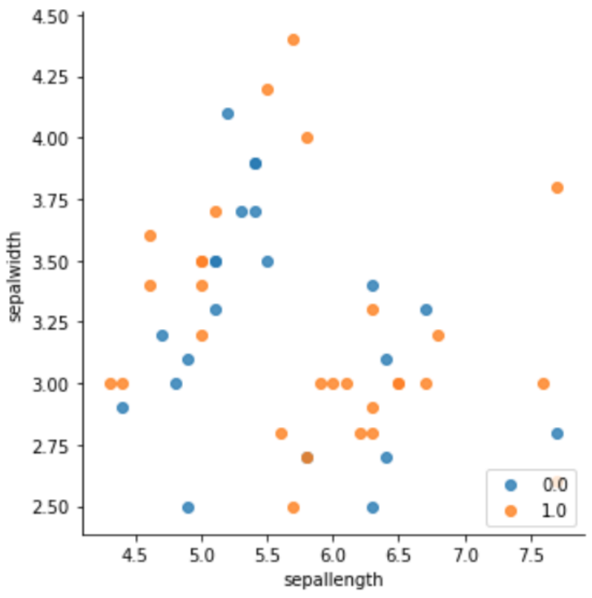

Figure 4: Prediction Scatter Plot

2 Parameter Estimation: MLE and MAP estimates

If X (e.g. packet arrival density) is Poisson distributed, then it has pmf

$$P(X|\lambda) = \frac{\lambda^X e^{-\lambda}}{X!}$$

1. **(10 points)** Show that $\hat{\lambda} = \frac{1}{n}\sum_i X_i$ is the maximum likelihood estimate of λ and that it is unbiased (that is, show that $\mathbb{E}[\hat{\lambda}] - \lambda = 0$).

$$\mathbf{E}_\lambda[\hat{\lambda}] = \mathbf{E}\left[\frac{1}{n}\sum_{i=0}^{n} X_i\right] = \frac{1}{n}\sum_{i=0}^{n}\mathbf{E}[X_i] = \frac{1}{n}\sum_{i=1}^{n}\lambda = \lambda.$$

   Each X_i from the Poisson distribution with λ has $\mathbf{E}[X_i] = \lambda$.

   From $\mathbf{E}_\lambda[\hat{\lambda}] - \lambda = 0$ and notes from the lecture slide it is clear that the MLE of λ is **unbiased**.

2. **(10 points)** Recall that the Gamma distribution has pdf:

$$p(\lambda|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}, \qquad \lambda > 0$$

   Assuming that λ is distributed according to $\Gamma(\lambda|\alpha,\beta)$, compute the posterior distribution over λ.

   $\mathrm{P}(\lambda \mid \mathcal{D}) \propto P(\mathcal{D} \mid \lambda) \cdot P(\lambda) = \left(\prod_{i=1}^{n} \frac{\lambda^{X_i} e^{-\lambda}}{X_i!}\right) \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}$

$\Gamma(\lambda; \sum_i X_i + \alpha, n + \beta)$ (?)

3. **(5 points)** Derive an analytic expression for the maximum a posteriori (MAP) estimate of λ under a $\Gamma(\alpha,\beta)$ prior.

mode$(\Gamma(\alpha,\beta)) = (\alpha-1)/\beta$

MAP = mode$(\Gamma(\sum_i X_i + \alpha, n+\beta))$ =

$$\frac{\sum_i X_i + \alpha - 1}{n+\beta}$$