Foundations of Data Analysis – WS20
Lab Assignment:
Supervised Learning

Due: 11:30 AM on the 12th of November
Contact: Alex (alex.markham@univie.ac.at)

0 Instructions and the data set

Please follow these submission instructions carefully, so that I can focus on grading and providing helpful feedback. Not following these instructions will result in a -5 point penalty, and I may ask you to resubmit it in the correct format. Upload your submission to Moodle as exactly *two* files, your python code and comments in the file `<last_name>.py` and your write-up containing the requested figures and question answers in `<last_name>.pdf`, replacing `<last_name>` with your last name(s). I suggest you use LaTeX for the pdf, but you can also hand-write yours answers and include them in the pdf as photos, or use any other markup language/text formatter you would like, as long as the submission is a pdf. Working in groups is allowed (and even encouraged), provided you do *all* of the following:

1. clearly include the names of the people you worked with (inside the write-up, not in the file name)
2. do all of the write-up yourself, in your own words
3. use the group for discussing *ideas* and not just sharing answers

For any questions, post in the dedicated lab discussion forum on Moodle (this is preferred, but you can also email me, Alex).

The data set `lab_iris_data.csv` on Moodle is a modified version of the classic Iris flower data set (make sure you use this version and not the version Moritz uploaded during the LDA tutorial). It has two classes and three features (whereas the original has an additional feature and an additional class).

Your code must be well-documented and readable; see the Style Guide for Python Code for the style conventions. The algorithm is not very complicated to implement and you should not need to use any sources other than the assignment sheet, your previous python tutorial, the lectures, and the documentation for numpy and matplotlib. However, if you insist on using other sources, you must thoroughly document and cite them—excessive use will be penalized, and undocumented/uncited use will be considered plagiarism and taken very seriously. If you're unsure, simply post your question on the Moodle forum or email me (Alex).

The *only* python packages you are allowed to use are `numpy` and `matplotlib` (plus you will need `from mpl_toolkits.mplot3d import Axes3D` to use matplotlib's 3-d scatterplot function).

1 Visualize the data (10%)

The first step is to load the data into Python. I suggest using the the numpy function `loadtxt()` along with the converters argument to change the class labels from strings to 0 and 1. Also make sure to give the argument `encoding='utf8'`, otherwise your converter function will need to additionally convert from bytes to strings.

Produce a 3-d scatterplot (see the documentation for help) of the data (each dimension corresponding to a feature), with the data points colored differently according to their class, and put this plot in your write-up.

2 Implement and train a regularized logistic regression model using stochastic gradient descent (30%)

Tasks

Recall that in logistic regression, our hypothesis is $h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = \frac{1}{1+\exp(-\mathbf{x}^{(i)}\boldsymbol{\theta})}$, where $\boldsymbol{\theta}$ is a column vector of parameters (this replaces $\underline{w}$ and b from the notes[1]), returns a probability of the sample $x^{(i)}$ being from a certain class. So, you are given a matrix $\mathbf{X} \in \mathbb{R}^{m,n}$ of samples (with m rows of samples and n columns of features) and a vector $\mathbf{y} \in \{0,1\}^m$ of labels to train the model. Then, using the trained model, you can classify a sample $\mathbf{x}^{(i)}$ in class 1 if $P(y = 1 \mid \mathbf{x}^{(i)}) = h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \frac{1}{2}$ and class 0 otherwise. We define the loss function as

$$\ell_{\text{l.r.}}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{m} \left[-y^{(i)} \log\left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right) - (1-y^{(i)}) \log\left(1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right)\right] + \lambda \sum_{j=1}^{n} |\theta_j|$$

and we can thus compute the (sub)gradient(s) (over the entire training set; but it could analogously be defined for any subset of samples) as follows

$$\frac{\partial}{\partial \boldsymbol{\theta}} \ell_{\text{l.r.}}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{m} \left[\left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\right) {\mathbf{x}^{(i)}}^{\top} \right] + \lambda \cdot \text{sign}(\boldsymbol{\theta}),$$

where $\text{sign}(\boldsymbol{\theta})$ returns a vector $\mathbf{s}$ such that $s_i = \begin{cases} 0 & \text{if } \theta_i = 0 \\ 1 & \text{if } \theta_i > 0 \\ -1 & \text{if } \theta_i < 0 \end{cases}$

[1] do not forget to add a column of 1s to $\mathbf{X}$ for the bias term in $\boldsymbol{\theta}$ so that the decision boundary is not constrained to pass through the origin

Recall the stochastic gradient descent algorithm:

Algorithm 1: Stochastic gradient descent

Data: Training samples $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{x}^{(m)} \end{bmatrix}$ and corresponding labels $\mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$

Parameters: T, number of iterations; $\eta_{\texttt{init}}$, initial learning rate; and k, batch size

initialize $\hat{\boldsymbol{\theta}} = \mathbf{0}$;
for t *in* $(1, ..., T)$ **do**
  $\eta \leftarrow \frac{\eta_{\texttt{init}}}{\sqrt{t}}$;
  randomly draw k samples from $(\mathbf{X}, \mathbf{y})$ to get $\mathbf{X}_k$ and $\mathbf{y}_k$;
  $\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} - \eta \cdot \frac{1}{k} \cdot \frac{\partial}{\partial \boldsymbol{\theta}} \ell(\hat{\boldsymbol{\theta}}, \mathbf{X}_k, \mathbf{y}_k)$;
end

Result: $\hat{\boldsymbol{\theta}}$ that minimizes ℓ

Implement stochastic gradient descent for regularized logistic regression. Make sure can you can easily adjust the regularization factor λ, the initial learning rate η, and the number of iterations T.

Train the classifier on the given data set, starting with values $\lambda = 1$, $T = 100$, $k = 20$, and $\eta_{\texttt{init}} = 0.1$. Answer the following questions in your write-up:

- do these parameter values result in a well-trained model?
- how do you know?
- experiment by trying different values and see if you find a set that performs better or faster.
- explicitly describe how varying each of the four parameters affects training and the final model.

Note: The Gradient Descent algorithm, loss functions, and gradients are usually with respect to the entire training set, but the (Batch) Stochastic Gradient Descent algorithm that you are implementing is a stochastic approximation of the Gradient Descent algorithm, so it approximates the true gradient by computing it and updating $\boldsymbol{\theta}$ after each (set of) training sample(s) rather than summing over all samples before updating $\boldsymbol{\theta}$.

3 Plot the separating hyperplane (30%)

Produce another 3-d scatterplot like in Task 1 above, but this time also include the separating hyperplane you found in Task 2, and include this in your write-up.

4 Further questions (30%)

Include your (thoughtful) answers to the following questions in your write-up, with 2–3 sentences to explain your reasoning for each answer:

- Given the above Note in Task 2 about non-stochastic Gradient Descent, why do the think the different variations of the algorithm exist, i.e. when is one more useful than the other?
- Under what conditions do they produce the same trained model?
- How (if at all) would you change the pseudocode in Algorithm 1 to use a different loss function?
- How does the regularized model we used here differ from the result if we had not used a regularization term?