Supervised Learning

0 Description and instructions

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 a *single* PDF comprising photos of your (legibly!) hand-written work or a document that has been typeset in TeX. This PDF should be named `<last_name>.pdf`, replacing `<last_name>` with your last name(s). 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

1 Stochastic Gradient Descent

You might find this section easier if you wait until gradient descent is covered in lecture (still well before the due date of this assignment). Nevertheless, all the necessary information is here and you should be able to complete these tasks before the lecture.

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 $\underset{\sim}{w}$ 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 $\mathbf{X} \subset \mathbb{R}^{m,n}$ of samples (with m rows of samples and n columns of features) and a vector $\mathbf{y} \subset \{0,1\}^m$ of labels to train the model. Then,

[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

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

$$\ell_{\text{l.r.}}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{m} \big[-y^{(i)} \log \big(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\big) - (1 - y^{(i)}) \log \big(1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\big)\big],$$

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

$$\frac{\partial}{\partial \boldsymbol{\theta}} \ell_{\text{l.r.}}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{m} \big(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\big) \mathbf{x}^{(i)\top}.$$

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 $\ell$
```

Tasks

For each of the following tasks, show your detailed calculations at every step. Simply providing the value without showing how it was calculated will not earn any credit. Round to at least three decimal places. You may use a scientific calculator, and you can represent the sigma notation summation as matrix multiplication to (slightly) condense your calculations, but you **may not** use more advanced calculators or features of e.g. Python or GNU Octave that automatically compute gradients or perform gradient descent.

1. Using an initial learning rate of $\eta_{\texttt{init}} = 0.1$, perform $T = 3$ iterations of stochastic gradient descent. Assume we have a large data set $\mathbf{X}$, from which we draw $k = 3$ samples:

$$\mathbf{x}^{(1)} = (1, 3); \quad y^{(1)} = 0$$
$$\mathbf{x}^{(2)} = (3, 1); \quad y^{(2)} = 1$$
$$\mathbf{x}^{(3)} = (1, 4); \quad y^{(3)} = 0$$

   or equivalently written using matrix notation:

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \mathbf{x}^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 3 & 1 \\ 1 & 4 \end{bmatrix} \text{ and the corresponding labels } \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Provide the learned values for $\hat{\boldsymbol{\theta}}$ and the final value for η. Note that you should use all of $\mathbf{X}_k$ for each iteration and that $\mathbf{X}_k \subset \mathbf{X}$ is the same for each iteration, even though in practice it would likely be a different subset each iteration, because it is randomly drawn from $\mathbf{X}$. (*tip*: make sure you understand the dimensions of $\boldsymbol{\theta}, \mathbf{X}, \mathbf{Y}, \mathbf{x}^{(i)}, y^{(i)}$, and how they are all related.)

2. What is the loss $\ell_{\text{l.r.}}$ of your learned $\hat{\boldsymbol{\theta}}$ (computed over the three samples) after each iteration?

3. Using the $\hat{\boldsymbol{\theta}}$ parameters learned in the first task, classify the following data point and state the probability that it belongs in that class (i.e., estimate $y^{(4)}$ and state the corresponding posterior):

$$x^{(4)} = (2, 2)$$