The objective of this lab is for you explore the behavior of gradient descent (GD) and stochastic gradient descent (SGD) for a multilayer perceptron (MLP) with a single hidden, sigmoidal layer (for nonlinear regression), and apply them to some datasets. The TA will first demonstrate the results of the algorithms on several datasets. Then, you will replicate those results and further explore the datasets with the algorithms. This lab is very similar to the previous one (GD/SGD for linear regression or classification). The only difference is that the model (an MLP) is nonlinear, and the gradient expressions are more complicated.

We provide you with the following:

- The script `lab08_linregr.m` sets up the problem (toy dataset) and plots several figures.
- The functions `mlpgd.m` and `mlpsgd.m` train an MLP (from D dimensions to D' dimensions) by gradient descent or by stochastic gradient descent, respectively. See also the functions `mlp.m` and `sigmoid.m`.

Important: when testing the algorithms, focus on 1D regression problems only, i.e., with inputs $\mathbf{x} \in \mathbb{R}$ and outputs $\mathbf{y} \in \mathbb{R}$, and use small datasets ($N = 10$ to 100 points), because training MLPs is slow.

I Datasets

Construct your own toy dataset as a noisy sample from a known function, e.g. $y_n = f(x_n) + \epsilon_n$ where $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ and $f(x) = ax + b$ or $f(x) = \sin x$.

II (Stochastic) gradient descent for nonlinear regression with MLPs

Review Consider nonlinear least-squares regression given a sample $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ with $\mathbf{x}_n \in \mathbb{R}^D$ and $\mathbf{y}_n \in \mathbb{R}^{D'}$:

$$E(\mathbf{W}, \mathbf{V}; \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^N) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n; \mathbf{W}, \mathbf{V})\|^2 = \frac{1}{2} \sum_{n=1}^N \sum_{i=1}^{D'} (y_{in} - f_i(\mathbf{x}_n; \mathbf{W}, \mathbf{V}))^2$$

where $\mathbf{f}$: $\mathbb{R}^D \to \mathbb{R}^{D'}$ is an MLP with one hidden layer having H units, where the hidden units are sigmoidal and the output units are linear:

$$f_i(\mathbf{x}) = \sum_{h=1}^H v_{ih} z_h + v_{i0}, \quad i = 1, \dots, D', \qquad z_h = \sigma\left(\sum_{d=1}^D w_{hd} x_d + w_{h0}\right), \qquad \sigma(t) = \frac{1}{1 + e^{-t}}.$$

The gradients (computed using the chain rule) of E wrt the weights are:

$$\frac{\partial E}{\partial v_{ih}} = \sum_{n=1}^N \big(-(y_{in} - f_i(\mathbf{x}_n))\, z_{hn} \big) \qquad \frac{\partial E}{\partial w_{hd}} = \sum_{n=1}^N \left(\left(\sum_{i=1}^{D'} -(y_{in} - f_i(\mathbf{x}_n))\, v_{ih} \right) z_{hn} \,(1 - z_{hn})\, x_{dn} \right).$$

Using these expressions we can implement gradient descent updates $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} + \boldsymbol{\Delta\Theta}$ with $\boldsymbol{\Delta\Theta} = -\eta \nabla E(\boldsymbol{\Theta})$, where $\boldsymbol{\Theta} = \{\mathbf{W}, \mathbf{V}\}$ are the weights of the MLP. Likewise, implementing stochastic gradient descent is done by summing only over a minibatch of points, instead of over all N points. GD/SGD proceed as in the previous lab on GD/SGD for linear regression, but now using the MLP gradient.

Exploration: toy problem Run each algorithm (GD and SGD) for, say, 100 iterations $\boldsymbol{\Theta}^{(0)}, \boldsymbol{\Theta}^{(1)}, \dots, \boldsymbol{\Theta}^{(100)}$ from an initial $\boldsymbol{\Theta} = \boldsymbol{\Theta}^{(0)}$ (equal to small random numbers, e.g. uniform in $[-0.01, 0.01]$). To visualize the results, we plot the following for each algorithm (GD and SGD):

- The dataset (y_n vs x_n) and the MLP function $f(x)$.
- The error $E(\boldsymbol{\Theta})$ over iterations, evaluated on the training set, and also on a validation set.

Note: unlike with linear regression, where the error decreases quickly in a few iterations, with an MLP you will need to run far more iterations (thousands to hundreds of thousands), even with a well-tuned η, to achieve convergence with GD.

Consider the following questions:

- Proceed as in the previous lab (GD/SGD for linear regression) in comparing GD with SGD, observing the effect on the error E of the learning rate η, minibatch size $|\mathcal{B}|$, etc.
- Plot the training error and the validation error over iterations.
  - The training error should decrease monotonically with GD if η is small enough, and will usually show flat, wide regions (where the error decreases very slowly), and steep, short regions (where the error decreases much more quickly). Why?
  - How about the validation error?
- Train MLPs with $H \in \{1, 2, 5, 10, 30, 50\}$ hidden units on the same dataset, and plot the resulting training and validation error. How do they look like?
- For an MLP with $H = 10$ hidden units, try weight decay, i.e., adding to the error function a term λw^2 for every weight w in the MLP (v_{ih} or w_{hd}), and hence adding $2\lambda w$ to the corresponding gradient. Try $\lambda \in \{0, 10^{-5}, 10^{-2}, 10^{0}\}$. How does the resulting MLP look like?
- Try different initial weights (randomly generated in $[-0.01, 0.01]$). Does GD converge to the same result every time? Try using as initial weights random values in $[-10, 10]$, what happens?

See the end of file `lab08_linregr.m` for suggestions of things to explore.