C477: Computational Optimisation
Coursework 1

Handed out: 09:00 26-10-2018
Due: 19:00 09-11-2018

Exercise 1. (Recognising Hand-written Digits in the MNIST Dataset).

Note 1: This exercise develops interesting optimisation algorithms. Therefore, in your submitted solution, please do not use any pre-packaged optimisation routines such as Matlab's `fminsearch` or the Optimization Toolbox.

Note 2: We are providing several Matlab files for your use. But there is no *requirement* to use our routines; please feel free to develop your own code. For this problem, please submit (i) a report answering the relevant mathematical questions and (ii) your modified `SolveMNIST_Gradient.m` file or (if you're not using our routines) sufficient material to reproduce your results.

Preliminaries: This exercise uses first-order methods to build a classifier recognising hand-written digits from images. Instead of using the digit image pixel values directly, file `mnist.mat` provides random Fourier features for the classification task.[1] These features allow us to learn a nonlinear decision function with a linear classifier. We formulate the problem as multiclass classification with n input features, $d = 10$ output labels (digits 0 – 9), and m training examples.

The training data in Matlab file `mnist.mat` contains input features $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ and output labels $\boldsymbol{y} \in \mathbb{Z}^m$ where $y_i \in \{0, 1, \ldots, 9\}$ $\forall\, i \in \{1, \ldots, m\}$. The optimisation problem is to learn feature weights $\boldsymbol{B} \in \mathbb{R}^{n \times d}$ for each digit $\{0, 1, \ldots, 9\}$ by minimising the negative log-likelihood over m training examples. We also include an ℓ_1

[1] `gridworld.wordpress.com/2015/07/17/random-features-for-large-scale-kernel-machines/`

regularisation term weighted by scalar $\lambda \in \mathbb{R}$ such that $\lambda > 0$:

$$\min_{\boldsymbol{\beta_1}, \ldots, \boldsymbol{\beta_{10}}} g(\boldsymbol{B}) + h(\boldsymbol{B}) = \min_{\boldsymbol{\beta_1}, \ldots, \boldsymbol{\beta_{10}}} \sum_{i=1}^{m} \left(\log \sum_{k=1}^{10} \exp\left(\boldsymbol{x_i}^\top \boldsymbol{\beta_k}\right) - \boldsymbol{x_i}^\top \boldsymbol{\beta_{y_i+1}} \right) + \lambda \sum_{k=1}^{10} \|\boldsymbol{\beta_k}\|_1,$$

where we define $\boldsymbol{\beta_k} \in \mathbb{R}^n \; \forall \, k \in \{1, \ldots, 10\}$ and $\boldsymbol{x_i} \in \mathbb{R}^n \; \forall \, i \in \{1, \ldots, m\}$ as vectors within $\boldsymbol{B}$ and $\boldsymbol{X}$, respectively:

$$\boldsymbol{B} = \begin{bmatrix} | & | & & | \\ \boldsymbol{\beta_1} & \boldsymbol{\beta_2} & \cdots & \boldsymbol{\beta_{10}} \\ | & | & & | \end{bmatrix}, \; \boldsymbol{X} = \begin{bmatrix} - & \boldsymbol{x_1}^\top & - \\ - & \boldsymbol{x_2}^\top & - \\ & \vdots & \\ - & \boldsymbol{x_m}^\top & - \end{bmatrix}.$$

We define $\boldsymbol{Y} \in \mathbb{R}^{m \times 10}$ as the *one-hot* encoding of the training labels: $Y_{i,\,k} = 1$ if example i has label $k - 1$ and $Y_{i,\,k} = 0$ otherwise. We state (without proof) that the gradient of $g(\boldsymbol{B})$, the negative log-likelihood, is:

$$\nabla_{\boldsymbol{B}} g(\boldsymbol{B}) = \boldsymbol{X}^\top \left(\boldsymbol{Z} \exp\left(\boldsymbol{X}\boldsymbol{B}\right) - \boldsymbol{Y} \right),$$

where $\exp(\cdot)$ is applied elementwise and $\boldsymbol{Z} \in \mathbb{R}^{m \times m}$ is the diagonal matrix with:

$$Z_{ii} = \frac{1}{\sum_{k=1}^{10} \exp\left(\boldsymbol{x_i}^\top \boldsymbol{\beta_k}\right)} \; \forall \, i \in \{1, \ldots, m\}.$$

Part 1. Prove that the optimisation problem is convex. Proceed by decomposition, i.e., prove that:

1. Log-Sum-Exp $\left(\log \sum_{k=1}^{10} \exp\left(B_{j\,k}\right) \right)$ is convex; **Marks=8**
2. Composition of a convex function with an affine mapping $\left(\log \sum_{k=1}^{10} \exp\left(\boldsymbol{x_i}^\top \boldsymbol{\beta_k}\right) \right)$ is convex; **Marks=3**
3. Affine functions $\left(-\boldsymbol{x_i}^\top \boldsymbol{\beta_{y_i+1}}\right)$ are convex; **Marks=3**
4. ℓ_1 Regularisation $\|\boldsymbol{\beta_k}\|_1$ is convex; **Marks=3**
5. The entire optimisation problem is convex. **Marks=3**

Part 2. The first-order methods covered in C477 assume the objective is continuously differentiable.[2] For our optimisation problem, $g(\boldsymbol{B}) \in \mathcal{C}^1$, but $h(\boldsymbol{B}) \notin \mathcal{C}^1$

[2]Extending first order methods to include special functions such as the 1-norm is not difficult; start by looking up the *proximal gradient* method.

1. Show that $h(\boldsymbol{B}) = \lambda \sum_{k=1}^{10} \|\boldsymbol{\beta_k}\|_1 \notin \mathcal{C}^1$

Marks=5

2. For the rest of the problem, we consider ℓ_2 regularisation so that the optimisation problem becomes:

$$\min_{\boldsymbol{\beta_1},\ldots,\boldsymbol{\beta_{10}}} f(\boldsymbol{B}) = \min_{\boldsymbol{\beta_1},\ldots,\boldsymbol{\beta_{10}}} \sum_{i=1}^{m} \left(\log \sum_{k=1}^{10} \exp\left(\boldsymbol{x_i}^\top \boldsymbol{\beta_k}\right) - \boldsymbol{x_i}^\top \boldsymbol{\beta_{y_i+1}} \right) + \lambda \sum_{k=1}^{10} \|\boldsymbol{\beta_k}\|_2^2,$$

Show that this new optimisation problem is convex and $f(\boldsymbol{B}) \in \mathcal{C}^1$.

Marks=10

3. Starting from `SolveMNIST_Gradient.m`, please run:

```
ReturnVal = SolveMNIST_Gradient(0.0001, 5000, 0.0001, 1);
```

`SolveMNIST_Gradient` will (soon) implement a gradient-based method. The algorithm runs with termination tolerance $\epsilon = 10^{-4}$, maximum 5000 iterations, constant step size $\alpha = 10^{-4}$, and regularisation parameter $\lambda = 1$. Lines 60, 65, & 70 of `SolveMNIST_Gradient` are wrong. Rather than checking tolerances:

$$\begin{aligned} \|\nabla f(\boldsymbol{B}^{(j)})\|_2 &< \epsilon, \\ \|\boldsymbol{B}^{(j+1)} - \boldsymbol{B}^{(j)}\|_2 &< \epsilon, \\ |f(\boldsymbol{B}^{(j+1)}) - f(\boldsymbol{B}^{(j)})| &< \epsilon, \end{aligned}$$

they are all set to 1. Please fix these lines so that `SolveMNIST_Gradient.m` will terminate when either one of the tolerances is satisfied or the number of iterations is reached. Note that we have not covered matrix norms in C477 and we are therefore happy for you to use the vector norm, i.e., to compute the norm of the vector of decision variables and the gradient provided in the Matlab files.

Marks=5

4. How do the tolerances in Part 2.3 correspond to the FONC, SONC, and SOSC? Show how each of Lines 60, 65, & 70 in `SolveMNIST_Gradient` now correspond to an optimality condition and state the relevant condition. **Marks=10**

5. Make gradient descent actually work! On line 43 of `SolveMNIST_Gradient.m`, implement a constant step-size strategy:

$$\boldsymbol{B}^{(j+1)} = \boldsymbol{B}^{(j)} - \alpha \nabla f(\boldsymbol{B}^{(j)})$$

Run the gradient descent algorithm. In your written report, provide a plot of the function value (`fcn_val_iter`) versus number of iterations.

Marks=10

6. What would be an exact line search strategy for this algorithm? Write out the line search optimisation problem. **Marks=10**

7. Implement a line search strategy such as Golden Section or a Secant-type method. Can you beat the constant step-size strategy? In your written report, please compare the constant step-size strategy to your new line search strategy. Provide a plot of the function value (`fcn_val_iter`) versus number of iterations for both the constant step size and line search strategy. **Marks=30**