INFR10086 Machine Learning (MLG) Semester 2, 2023/4

Coursework

Instructions

- Due date: 11 March, Monday, at 12pm
- The submission is through Gradescope `https://www.gradescope.com/courses/741901`.
- It's best to typeset your answers, but it is fine to submit hand-written answers.
- To get full marks for the last question, you need to paste the code and the plots into a PDF file, and submit that along with answers to other questions.

Questions

1. In this question, we are going to work out the convergence rate of gradient descent on a particular family of functions.

   (a) Show that if f is convex, then

   $$f(y) - f(x) - \nabla f(x)^\top (y - x) \leq (\nabla f(y) - \nabla f(x))^\top (y - x) \tag{1}$$

   for any x and y.

   [2 marks]

   Remind yourself what Cauchy–Schwarz is. Show that if f is convex, then

   $$f(y) - f(x) - \nabla f(x)^\top (y - x) \leq \|\nabla f(y) - \nabla f(x)\|_2 \|y - x\|_2 \tag{2}$$

   for any x and y.

   [2 marks]

   A function f is said to be L-smooth if

   $$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2 \tag{3}$$

   for all x and y. Using the above, show that if f is convex and L-smooth, then

   $$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + L\|x - y\|_2^2 \tag{4}$$

   for any x and y.

   [2 marks]

   We can conclude that if f is convex and L-smooth, you can always find a parabola that bounds the function from above.

(b) Let's consider the case where we do gradient descent on a convex and L-smooth function f. The gradient descent algorithm can be summarized as

$$x_t = x_{t-1} - \eta_t \nabla f(x_{t-1}) \tag{5}$$

for $t = 1, \dots, T$. We know that if f is convex and L-smooth, then

$$f(y) \le f(x) + \nabla f(x)^\top (y - x) + L\|x - y\|_2^2 \tag{6}$$

for any x and y. Plug in $x = x_{t-1}$ and $y = x_t$ to the above and show that

$$f(x_{t-1}) - f(x_t) \ge \eta_t(1 - L\eta_t)\|\nabla f(x_{t-1})\|_2^2. \tag{7}$$

[2 marks]

Consider $g(s) = s(1 - Ls)$. This is a concave[1] parabola. Show that g has a maximum of $\frac{1}{4L}$ when $s = \frac{1}{2L}$, and g is 0 when $s = \frac{1}{L}$.

[2 marks]

Using the above results, we can choose $\eta_t = \frac{1}{2L}$ and conclude

$$f(x_{t-1}) - f(x_t) \ge \frac{1}{4L}\|\nabla f(x_{t-1})\|_2^2 \ge 0. \tag{8}$$

In other words, if we know the function is convex and L-smooth and we do gradient descent with a step size of $\frac{1}{2L}$, we are guaranteed to decrease the objective after every gradient update.

(c) A function f is said to be μ-strongly convex if

$$f(y) \ge f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2}\|y - x\|_2^2 \tag{9}$$

for any x and y. Note the direction of the inequality. Compared to L-smooth functions that have parabolas bounded from above, strongly convex functions have parabolas bounded from below.

Show that, for a particular x, the quadratic function on the right hand side

$$g(z) = f(x) + \nabla f(x)^\top (z - x) + \frac{\mu}{2}\|z - x\|_2^2 \tag{10}$$

is minimized when $z = x - \frac{1}{\mu}\nabla f(x)$.

[2 marks]

Based on the result above, show that

$$f(y) \ge g(y) \ge \min_z g(z) = f(x) - \frac{1}{2\mu}\|\nabla f(x)\|_2^2, \tag{11}$$

for any x and y.

[2 marks]

[1] A function f is concave if $-f$ is convex.

Now choose $y = x^*$ where x^* is the minimizer of f, and conclude that

$$f(x) - f(x^*) \leq \frac{1}{2\mu}\|\nabla f(x)\|_2^2. \tag{12}$$

[2 marks]

In other words, if a function is strongly convex, we know how far the optimal solution is just by looking at the norm of the gradient.

(d) We know that if we perform gradient descent on a L-smooth function f then

$$f(x_t) \leq f(x_{t-1}) - \frac{1}{4L}\|\nabla f(x_{t-1})\|_2^2. \tag{13}$$

We can subtract $f(x^*)$ from both sides to get

$$f(x_t) - f(x^*) \leq f(x_{t-1}) - f(x^*) - \frac{1}{4L}\|\nabla f(x_{t-1})\|_2^2. \tag{14}$$

If we further assume that f is μ-strongly convex, show that

$$f(x_t) - f(x^*) \leq f(x_{t-1}) - f(x^*) + \frac{\mu}{2L}(f(x^*) - f(x_{t-1})) \tag{15}$$

$$= \left(1 - \frac{\mu}{2L}\right)(f(x_{t-1}) - f(x^*)). \tag{16}$$

[2 marks]

Apply this result repeatedly, and conclude that

$$f(x_t) - f(x^*) \leq \left(1 - \frac{\mu}{2L}\right)^t (f(x_0) - f(x^*)). \tag{17}$$

[2 marks]

In sum, this is the convergence rate if we run gradient descent with a constant step size of $\frac{1}{2L}$ on an L-smooth, μ-strongly convex function.
Is the convergence quadratic, linear, or sublinear?

[2 marks]

2. In this question, we are going to study the properties of hinge loss.

(a) Show that

$$\max(a + b, c + d) \leq \max(a, c) + \max(b, d) \tag{18}$$

for any a, b, c, and d.

[2 marks]

Use the above result and show that if f and g are both convex, then

$$h(x) = \max(f(x), g(x)) \tag{19}$$

is also convex.

[2 marks]

(b) In class, we have talked about the hinge loss for binary classification

$$\ell_{\text{hinge}}(w; x, y) = \max(0, 1 - yw^\top x) \tag{20}$$

where (x, y) is a sample and $y \in \{+1, -1\}$. Use the result above and show that the hinge loss is convex in w.

[2 marks]

(c) Recall that the zero-one loss is

$$\ell_{01}(w; x, y) = \mathbb{1}_{yw^\top x < 0} \tag{21}$$

where (x, y) is a sample and $y \in \{+1, -1\}$. Show that the hinge loss is always an upper bound of the zero-one loss. (Hint: Enumerate the possible values of y and the possible signs of $w^\top x$.)

[2 marks]

(d) Show that if f is convex and g is μ-strongly convex, then

$$h(x) = f(x) + \lambda g(x) \tag{22}$$

is $\lambda\mu$-strongly convex for $\lambda > 0$. (Hint: Use the definition of convex and strongly convex functions.)

[2 marks]

Show that $g(x) = \frac{1}{2}\|x\|_2^2$ is 1-strongly convex.

[2 marks]

If we optimize the loss function

$$\frac{1}{n}\sum_{i=1}^{n} \ell_{\text{hinge}}(w; x_i, y_i) + \frac{\lambda}{2}\|w\|_2^2 \tag{23}$$

on a data set $\{(x_1, y_1), \dots, (x_n, y_n)\}$, conclude that this loss function is λ-strongly convex.

[2 marks]

Compare the above objective to support vector machines, and convince yourself that support vector machines are optimizing the hinge loss.

3. MNIST is a data set consisting of hand-written digits. In this question, we are going to implement a linear classifier using only `numpy`. You are **not** allowed to use any packages other than `numpy`, `matplotlib`, and those in the Python standard library.

Download the tar ball from `https://homepages.inf.ed.ac.uk/htang2/mlg2023/mnist.tar.gz`. It includes the data set, and a file `mnist.py` to help you load the data set.

To get full marks for this question, you need to paste all your code and plots in a PDF and submit that with your answers to other questions.

(a) Use the snippet below to load the training set.

```python
import mnist
images = mnist.load_images('train-images-idx3-ubyte')
```

Since each image is 28×28, we can linearize every image to a 784-dimensional vector. Simply use `numpy.reshape` to reshape the 28×28 matrix to a 784-dimensional vector. Write a script to find the mean of the whole data set. Reshape the mean vector back to a 28×28 matrix and use `pyplot.imshow` to plot the mean "image". What does the mean image look like?

[4 marks]

(b) Remind yourself about the log loss

$$\ell(w; x, y) = -\log p(y|x) = -\log \frac{\exp(w_y^\top x)}{\sum_{y' \in \{0,\dots,9\}} \exp(w_{y'}^\top x)} \tag{24}$$

$$= -w_y^\top x + \log \sum_{y' \in \{0,\dots,9\}} \exp(w_{y'}^\top x) \tag{25}$$

for multiclass classification. Note that you will need a weight vector for each label. In other words, we need weight vectors $w_0, \dots, w_9$ for labels $0, \dots 9$. What is the gradient $\nabla_{w_{\tilde{y}}} \ell$?

[2 marks]

Mind the different y's. The symbol y is used for the ground truth label, the symbol y' is used in the sum, and the symbol $\tilde{y}$ is some label that we take gradient of.

(c) Use the snippet below to load the labels of the data set.

```
import mnist
labels = mnist.load_labels('train-labels-idx1-ubyte')
```

Implement stochastic gradient descent (SGD) to optimize the log loss above with a batch size of 1. Recall that an epoch is a pass over the data set. The training set have 60,000 samples, so one epoch should give you 60,000 gradient updates. Print the loss value on a sample before the gradient update. Run SGD for 10 epochs. Average the losses printed, and you should end up with 10 average loss values. It is a good practice to save a classifier after every epoch, in case you want to continue from where you left off.

Plot the losses where the x-axis is the number of epochs and the y-axis is the averaged loss values per epoch. Repeat this process for different step sizes and overlay the loss curves on top of each other.

What plot do you get after overlaying the loss curves? What is the best step size that leads to the lowest loss values?

[6 marks]

Write a script to load the two files `t10k-images-idx3-ubyte` and `t10k-labels-idx1-ubyte`. Implement

$$\hat{y} = \operatorname*{argmax}_{y \in \{0,\dots,9\}} w_y^\top x \tag{26}$$

for prediction, and measure the zero-one loss of the classifiers you have. What is the zero-one loss that you get for your classifiers?

[2 marks]

```python
import array
import sys
import numpy

def load_images(filename):
    f = open(filename, 'rb')

    sig = f.read(4)
    dim1 = int.from_bytes(f.read(4), byteorder='big', signed=False)
    dim2 = int.from_bytes(f.read(4), byteorder='big', signed=False)
    dim3 = int.from_bytes(f.read(4), byteorder='big', signed=False)

    data = numpy.array(array.array('B', f.read()))
    result = data.reshape(dim1, dim2, dim3)

    f.close()

    return result


def load_labels(filename):
    f = open(filename, 'rb')

    sig = f.read(4)
    dim1 = int.from_bytes(f.read(4), byteorder='big', signed=False)

    result = numpy.array(array.array('B', f.read()))

    f.close()

    return result
```

Listing 1: File `mnist.py`.