Machine Learning - Homework 3

Parita Pooj (psp2133)

October 12, 2016

Problem 1

(a) Let $L(\boldsymbol{w})$ be the objective function we are trying to minimize:

$$L(\boldsymbol{w}) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S}(\langle \boldsymbol{w}, \boldsymbol{x}\rangle - y)^2$$

$$L(\boldsymbol{w}) = \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S}(\langle \boldsymbol{w}, \boldsymbol{x}\rangle^2 - 2\langle \boldsymbol{w}, \boldsymbol{x}\rangle y + y^2)$$

Gradient:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \lambda\boldsymbol{w} + \frac{2}{|S|}\sum_{(\boldsymbol{x},y)\in S}(\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{w} - y\boldsymbol{x})$$

Hessian:

$$\frac{\partial^2 L}{\partial \boldsymbol{w}\partial \boldsymbol{w}^T} = \lambda I + \frac{2}{|S|}\sum_{(\boldsymbol{x},y)\in S}\boldsymbol{x}\boldsymbol{x}^T$$

where I is an identity matrix of size $d \times d$ From above we know that:
λI is positive semidefinite
$\boldsymbol{x}\boldsymbol{x}^T$ is positive semidefinite
Also, the sum of positive semidefinite matrices is also positive semidefinite.
From this, it implies that the hessian of the objective function is positive semidefinite and hence, convex.
Thus we can say that this is a convex optimization problem considering that it does not have any more constraints.

(b) With the gradient calculation as given above: Gradient:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \lambda\boldsymbol{w} + \frac{2}{|S|}\sum_{(\boldsymbol{x},y)\in S}(\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{w} - 2y\boldsymbol{x})$$

we can write the gradient descent algorithm for the given optimization problem as below:

Algorithm 1 Problem 1: Gradient Descent

procedure GRADIENT_DESCENT($\boldsymbol{x}, y, \boldsymbol{w}, \boldsymbol{\eta}, N, \lambda$)
  $t \leftarrow 1$
  $\boldsymbol{w}^{(t)} \leftarrow \mathbf{0}$
  for $t = 1, 2, 3, \ldots, N$ **do**
    $\bigtriangledown \leftarrow$ COMPUTE_GRADIENTS($\boldsymbol{x}, y, \boldsymbol{w}^{(t)}, \lambda$)
    $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta_t \bigtriangledown$; ▷ Update weight vector
    $t \leftarrow t + 1$
  return $\boldsymbol{w}$ ▷ The final weight vector for the classifier

procedure COMPUTE_GRADIENTS($\boldsymbol{x}, y, \boldsymbol{w}, \lambda$)
  $n \leftarrow |\mathrm{S}|$ ▷ Number of data points = cardinality of S
  $\bigtriangledown \leftarrow \lambda \boldsymbol{w} + \frac{2}{n} \sum_{i=1}^{n} \{\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle \boldsymbol{x}_i - y_i \boldsymbol{x}_i\}$ ▷ Gradient Computation
  return $\bigtriangledown$

(c) The constraint:
$w_i^2 \leq 1$ for all $i = 1, 2, 3, \ldots, d$.
can be written as:
$w_i^2 - 1 \leq 0$ for all $i = 1, 2, 3, \ldots, d$.
This function is a quadratic function which represents a convex set where each feature is less than or equal to one.
If we consider the second derivative for each i, we can see that we get a positive result which implies that the function is convex
If this is written as a vector as well for R^d, we get the hessian as $2I$ which we know is positive semidefinite.
Thus, the function is convex and hence, the optimization problem remains convex with the addition of this constraint.

(d) The constraint:
$w_{2i-1} = 1 - w_{2i}$ for all $i = 1, 2, 3, \ldots, d$.
can be written as:
$w_{2i-1} + w_{2i} - 1 \leq 0$ for all $i = 1, 2, 3, \ldots, d$.
$1 - w_{2i-1} - w_{2i} \leq 0$ for all $i = 1, 2, 3, \ldots, d$.
Both of the above equations represent the equation of a hyperplane in d-dimensional space. We know that a line, plane or a hyperplane is always convex.
Thus, both the above functions are convex and hence, the optimization problem is still convex.

(e) The constraint:
$w_i^2 = 1$ for all $i = 1, 2, 3, \ldots, d$.
can be written as:
$w_i^2 - 1 \leq 0$ for all $i = 1, 2, 3, \ldots, d$.
$1 - w_i^2 \leq 0$ for all $i = 1, 2, 3, \ldots, d$.
Here, the first equation is convex but the second isn't. This is because the Hessian for the first equation will be I, and the Hessian for the second equation will be -I.
This can also be looked at by considering that a set of points such that $w_i = \pm 1$ will not form a convex set.
Thus, one of the two additional constraints is convex and with this constraint, the optimization problem will not remain convex.

Problem 2

(a) Let $\boldsymbol{w}$ be $[\boldsymbol{\beta}, \beta_0]$ and $\boldsymbol{x}_i$ for all $i = 1, 2, \ldots, n$ transform to $\boldsymbol{x}_i = [\boldsymbol{x}_i, 1]$
The above is based on the lifting trick.
Thus, the optimization problem becomes:

$$\min_{\boldsymbol{w} \in \mathbb{R}^{d+1}} \frac{1}{n} \sum_{i=1}^{n} \{\ln(1 + \exp(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) - y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle\}$$

Gradient of the objective function can be written as:

$$\bigtriangledown = \frac{1}{n} \sum_{i=1}^{n} \frac{\exp(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)}{1 + \exp(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)} \boldsymbol{x}_i - y_i \boldsymbol{x}_i$$

$$\bigtriangledown = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{1 + \exp(-\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)} \boldsymbol{x}_i - y_i \boldsymbol{x}_i$$

Thus, for every iteration t of gradient descent, we update $\boldsymbol{w}$ as below:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta_t \bigtriangledown$$

Thus, we have:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta_t \frac{1}{n} \sum_{i=1}^{n} \frac{1}{1 + \exp(-\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)} \boldsymbol{x}_i - y_i \boldsymbol{x}_i$$

With this, we can write the pseudocode as below:

Algorithm 2 Problem 2: Gradient Descent

procedure GRADIENT_DESCENT($\boldsymbol{x}, y, \boldsymbol{w}, \boldsymbol{\eta}, N$)
  $t \leftarrow 1$
  $\boldsymbol{w}^{(t)} \leftarrow \mathbf{0}$
  for $t = 1, 2, 3, \ldots, N$ **do**
    $\lambda \leftarrow$ COMPUTE_GRADIENTS($\boldsymbol{x}, y, \boldsymbol{w}^{(t)}$)
    $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta_t \lambda$; ▷ Update weight vector
    $t \leftarrow t + 1$
  return $\boldsymbol{w}$ ▷ The final weight vector for the classifier

procedure COMPUTE_GRADIENTS($\boldsymbol{x}, y, \boldsymbol{w}$)
  $n \leftarrow$ *number of data points* ▷ The size of $\boldsymbol{x}$
  $\lambda \leftarrow \frac{1}{n} \sum_{i=1}^{n} \{ \frac{1}{1 + \text{EXP}(-\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)} \boldsymbol{x}_i - y_i \boldsymbol{x}_i \}$ ▷ Gradient Computation
  return λ

(b) Code for gradient descent algorithm implemented in file: *hw2_p2b.m*
The algorithm stops after ∼4658 iterations.
It stops with the final objective value of 0.650639998712717
The final weight vector is as below:

$$\beta_0 = -0.957915500965539$$

$$\boldsymbol{\beta} = \begin{bmatrix} -0.007426394047568 \\ 2.095272030923660 \\ -0.001124127969434 \end{bmatrix}$$

(c) Code for gradient descent algorithm implemented in file: *hw2_p2b.m*
The algorithm stops after ∼377 iterations.
It stops with the final objective value of 0.650639985720403
The final weight vector is as below:

$$\beta_0 = -0.955376722802542$$

$$\boldsymbol{\beta} = \begin{bmatrix} -0.153439800249147 \\ 2.099782135695885 \\ -0.027573523361225] \end{bmatrix}$$

By plotting the data, we observe that the features 1, 3 range from from 0.0 to 1.0 approximately, while the feature 2 ranges from 0.0 to 20.0 approximately.

For gradient descent, we can scale the data so that the gradients computed are normalized and hence, we move in the direction of gradient faster, without approaching a zig-zag path.
If we look at the gradient computation for our objective function, we see that it is a sum of scaled data points. Thus, if the data points are normalized, the gradient vector have normalized features and hence, will point towards descent faster.
This can be visually depicted by showing the objective function which initially mapped as an elipse, where we moved along a zig-zag path. With normalization, the ellipse becomes circular, with gradient vectors pointing towards the minimum while following an almost straight line path.

The A matrix used to scale data is given as below:

$$A = \begin{bmatrix} 0.050016536494057 & 0 & 0 \\ 0 & 1.000207478593686 & 0 \\ 0 & 0 & 0.050027566620478 \end{bmatrix}$$

(d) Code for gradient descent algorithm implemented in file: *hw2_p2d.m*
Part 1: Original Data
(1) Number of iterations: 512
(2) Final objective value: 0.655069623459742
(3) Final hold-out error rate: 38.292682926829272 %

Part 2: Transformed Data
(1) Number of iterations: 32
(2) Final objective value: 0.664759252137848
(3) Final hold-out error rate: 37.926829268292686 %

References

[1] `https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf`