Tutotial 5: Gradient Descent

In this tutorial, we will implement the soft margin SVM using different gradient descent methods. Our training data consists of n pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. Define a hyperplane by

$$\{x : f(x) = x^T\mathbf{w} + b = 0\}.$$

A classification rule induced by $f(x)$ is

$$G(x) = \text{sign}(x^T\mathbf{w} + b).$$

To recap, to estimate the $\mathbf{w}$, b of the soft margin SVM, we can minimize the cost:

$$f(\mathbf{w}, b) = \frac{1}{2}\sum_{j=1}^{d}(w^{(j)})^2 + C\sum_{i=1}^{n}\max\left\{0, 1 - y_i\left(\sum_{j=1}^{d} w^{(j)}x_i^{(j)} + b\right)\right\}. \tag{1}$$

Define $L(\mathbf{w}, b; x_i, y_i) = \max\{0, 1 - y_i(\sum_{j=1}^{d} w^{(j)}x_i^{(j)} + b)\}$. In order to minimize the cost function, we first obtain the gradient with respect to $w^{(j)}$, the jth item in the vector $\mathbf{w}$, and b as follows:

$$\begin{aligned}
\nabla_{w^{(j)}} f(\mathbf{w}, b) &= \frac{\partial f(\mathbf{w}, b)}{\partial w^{(j)}} = w_j + C\sum_{i=1}^{n}\frac{\partial L(\mathbf{w}, b; x_i, y_i)}{\partial w^{(j)}}, \\
\nabla_b f(\mathbf{w}, b) &= \frac{\partial f(\mathbf{w}, b)}{\partial b} = C\sum_{i=1}^{n}\frac{\partial L(\mathbf{w}, b; x_i, y_i)}{\partial b},
\end{aligned} \tag{2}$$

where

$$\begin{aligned}
\frac{\partial L(\mathbf{w}, b; x_i, y_i)}{\partial w^{(j)}} &= \begin{cases} 0 & \text{if } y_i(x_i^T\mathbf{w} + b) \geq 1 \\ -y_i x_i^{(j)} & \text{otherwise.} \end{cases} \\
\frac{\partial L(\mathbf{w}, b; x_i, y_i)}{\partial b} &= \begin{cases} 0 & \text{if } y_i(x_i^T\mathbf{w} + b) \geq 1 \\ -y_i & \text{otherwise.} \end{cases}
\end{aligned}$$

Now, we will implement and compare the following gradient descent techniques:

- **Batch gradient descent:** Iterate through the entire dataset and update the parameters as follows:

  $k = 0$
  while convergence criteria not reached **do**
    for $j = 1, \ldots, d$ **do**
      Update $w^{(j)} \leftarrow w^{(j)} - \eta\nabla_{w^{(j)}} f(\mathbf{w}, b)$
    end for
    Update $b \leftarrow b - \eta\nabla_b f(\mathbf{w}, b)$
    Update $k \leftarrow k + 1$
  end while

  where,
  n is the number of samples in the training data,
  d is the dimensions of $\mathbf{w}$,
  η is the learning rate of the gradient descent, and
  $\nabla_{w^{(j)}} f(\mathbf{w}, b)$ and $\nabla_b f(\mathbf{w}, b)$ are the values compuated from equation (2).
  The *convergence criteria* for the above algorithm is $\Delta_{\%cost} < \epsilon$, where

  $$\Delta_{\%cost} = \frac{|f_{k-1}(\mathbf{w}, b) - f_k(\mathbf{w}, b)| \times 100}{f_{k-1}(\mathbf{w}, b)}. \tag{3}$$

  Here,
  $f_k(\mathbf{w}, b)$ is the value of equation (1) at kth iteration,

$\Delta_{\%cost}$ is computed at the end of each iteration of the while loop.
Initialize $\mathbf{w} = 0$, $b = 0$ and compute $f_0(\mathbf{w}, b)$ with these values.
For this method, use $\eta = 0.0000003$, $\epsilon = 0.25$.

- **Stochastic gradient descent:** Go through the dataset and update the parameters, one training sample at a time, as follows:

  Randomly shuffle the training data
  $i = 1$, $k = 0$
  while convergence criteria not reached **do**
    for $j = 1, \ldots, d$ **do**
      Update $w^{(j)} \leftarrow w^{(j)} - \eta \nabla_{w^{(j)}} f_i(\mathbf{w}, b)$
    end for
    Update $b \leftarrow b - \eta \nabla_b f_i(\mathbf{w}, b)$
    Update $i \leftarrow (i \bmod n) + 1$
    Update $k \leftarrow k + 1$
  end while

  where,
  n is the number of samples in the training data,
  d is the dimension of $\mathbf{w}$,
  η is the learning rate and
  $\nabla_{w^{(j)}} f_i(\mathbf{w}, b)$ is defined for a single training sample as follows:

  $$\nabla_{w^{(j)}} f_i(\mathbf{w}, b) = \frac{\partial f_i(\mathbf{w}, b)}{\partial w^{(j)}} = w_j + C \frac{\partial L(\mathbf{w}, b; x_i, y_i)}{\partial w^{(j)}}$$

  $\nabla_b f_i(\mathbf{w}, b)$ is similar.

  The *convergence criteria* here is $\Delta_{cost}^{(k)} < \epsilon$, where

  $$\Delta_{cost}^{(k)} = 0.5 \Delta_{cost}^{(k-1)} + 0.5 \Delta_{\%cost},$$

  where,
  k is the iteration number, and
  $\Delta_{\%cost}$ is the same as above (from equation 3).
  Calcuate Δ_{cost}, $\Delta_{\%cost}$ at the end of each iteration of the while loop.
  Initialize $\Delta_{cost} = 0$, $\mathbf{w} = 0$, $b = 0$ and compute $f_0(\mathbf{w}, b)$ with these values.
  For this method, use $\eta = 0.0001, \epsilon = 0.001$.

- **Mini batch gradient descent:** Go through the dataset in batches of predetermined size and update the parameters, one training sample at a time, as follows:

  Randomly shuffle the training data
  $l = 1$, $k = 0$
  while convergence criteria not reached **do**
    for $j = 1, \ldots, d$ **do**
      Update $w^{(j)} \leftarrow w^{(j)} - \eta \nabla_{w^{(j)}} f_l(\mathbf{w}, b)$
    end for
    Update $b \leftarrow b - \eta \nabla_b f_l(\mathbf{w}, b)$
    Update $l \leftarrow (l + 1) \bmod ((n + batch\_size - 1)/batch\_size)$
    Update $k \leftarrow k + 1$
  end while

  where,
  n is the number of samples in the training data,
  d is the dimension of $\mathbf{w}$,
  η is the learning rate and

batch_size is the number of training samples considered in each batch, and $\nabla_{w^{(j)}} f_l(\mathbf{w}, b)$ is defined for a batch of training sample as follows:

$$\nabla_{w^{(j)}} f_l(\mathbf{w}, b) = \frac{\partial f_l(\mathbf{w}, b)}{\partial w^{(j)}} = w_j + C \sum_{i=l*batch\_size+1}^{\min\{n,(l+1)*batch\_size\}} \frac{\partial L(\mathbf{w}, b; x_i, y_i)}{\partial w^{(j)}}$$

The *convergence criteria* here is $\Delta_{cost}^{(k)} < \epsilon$, where

$$\Delta_{cost}^{(k)} = 0.5\Delta_{cost}^{(k-1)} + 0.5\Delta_{\%cost},$$

where,
k is the iteration number, and
$\Delta_{\%cost}$ is the same as above (equation 3).
Calcuate Δ_{cost}, $\Delta_{\%cost}$ at the end of each iteration of the while loop.
Initialize $\Delta_{cost} = 0$, $\mathbf{w} = 0$, $b = 0$ and compute $f_0(\mathbf{w}, b)$ with these values.
For this method, use $\eta = 0.00001, \epsilon = 0.01$, batch_size = 20.

When implementing the SVM algorithm for all the the above mentioned gradient descent techniques, use $C = 100$. For all other parameters, use the values specified in the description of the technique. We update w in iteration $k + 1$ using the values computed in iteration k.

The data set contains the following files:

1. `features.txt`: Each line contains features (comma-separated values) for a single datapoint. It have 6414 datapoints (rows) and 122 features (columns).

2. `target.txt`: Each line contains the response variable (y = -1 or 1) for the corresponding row in `features.txt`.