# CSCE 4013-002: Assignment 5

Due 11:59pm Sunday, December 1, 2019

## 1 Implementation of SVM via Batch Gradient Descent

You will implement the soft margin SVM using the batch gradient descent method as we learned in class. To recap, to estimate the $\mathbf{w}, b$ of the soft margin SVM, we can minimize the cost:

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

In order to minimize the function, we first obtain the gradient with respect to $w^{(j)}$, the $j$th item in the vector $\mathbf{w}$, as follows:

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

where:

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

Now, we will implement the batch gradient descent technique:

**Batch Gradient Descent**

---

**Algorithm 1:** 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 $w^{(j)} \leftarrow w^{(j)} - \eta\nabla_{w^{(j)}} f(\mathbf{w}, b)$;
    Update $b \leftarrow b - \eta\nabla_b f(\mathbf{w}, b)$;
    Update $k \leftarrow k + 1$;

---

where,
$n$ is the number of samples in the training dataset,
$d$ is the dimensions of $\mathbf{w}$,
$\eta$ is the learning rate of the gradient descent, and
$\nabla_{w^{(j)}} f(\mathbf{w}, b)$ is the value computed from computing Eq. (2) above and $\nabla_b f(\mathbf{w}, b)$ is the value computed from your answer in question (a) below.

The *convergence criteria* for the above algorithm is $\Delta_{\%cost} < \epsilon$, where

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

where,
$f_k(\mathbf{w}, b)$ is the value of Eq. (1) at $k$th iteration, and
$\Delta_{\%cost}$ is computed at the end of each iteration of the while loop.

Initialize $\mathbf{w} = \mathbf{0}, b = 0$ and compute $f_0(\mathbf{w}, b)$ with these values. Use $\eta = 0,0000003$, $\epsilon = 0.25$, and $C = 100$.

**Question (a)**

Notice that we have not given you the equation for $\nabla_b f(\mathbf{w}, b)$.
**Task:** What is $\nabla_b f(\mathbf{w}, b)$ used for the Batch Gradient Descent Algorithm?
*(Hint: It should be very similar to $\nabla_{w^{(j)}} f(\mathbf{w}, b)$.)*

**Question (b)**

**Task:** Implement the SVM algorithm for the batch gradient descent algorithm. **Note:** update $w$ in iteration $i + 1$ using the values computed in iteration $i$. Do not update using values computed in the current iteration!

Run your implementation on the following training dataset (we don't consider the testing dataset in this assignment):

1. `features.train.txt`: Each line contains features (comma-separated values) for a single datapoint (i.e., $\mathbf{x}_i$). It has 6000 datapoints (rows) and 122 features (columns).
2. `target.train.txt`: Each line contains the target variable (i.e., $y_i$) for the corresponding row in `features.train.txt`.

**Task:** Plot the value of the cost function $f_k(\mathbf{w}, b)$ vs. the number of iterations ($k$).

As a sanity check, Batch GD should converge in 10-300 iterations. If your implementation consistently takes longer though, you may have a bug.

**What to submit**

1. Equation for $\nabla_b f(\mathbf{w}, b)$.
2. Plot of $f_k(\mathbf{w}, b)$ vs. the number of iterations ($k$).
3. The source code.

# 2 Decision tree

In this problem, we want to construct a decision tree to find out if a person will enjoy beer.

**Definitions:** Let there be $k$ binary-valued attributes in the data. We pick an attribute that maximizes the gain at each node using Gini index (a close alternative to information gain):

$$G = I(D) - (I(D_L) + I(D_R)), \tag{4}$$

where $D$ is the given dataset, and $D_L$ and $D_R$ are the subsets on left and right hand-side branches after splitting. Ties may be broken arbitrarily. We define $I(D)$ as follows:

$$I(D) = |D| \times \sum_i p_i(1 - p_i) = |D| \times \left(1 - \sum_i p_i^2\right),$$

where $|D|$ is the number of items in $D$; $p_i$ is the probability distribution of the items in $D$, or in other words, $p_i$ is the fraction of items that take value $i \in \{+, -\}$. Put differently, $p_+$ is the fraction of positive items and $p_-$ is the fraction of negative items in $D$.

**Task:** Let $k = 3$. We have three binary attributes that we could use: "likes wine", "likes running" and "likes pizza". Suppose the following:

- There are 100 people in sample set, 60 of whom like beer and 40 who don't.
- Out of the 100 people, 50 like wine; out of those 50 people who like wine, 30 like beer.
- Out of the 100 people, 30 like running; out of those 30 people who like running, 20 like beer.
- Out of the 100 people, 80 like pizza; out of those 80 people who like pizza, 50 like beer.

What are the values of $G$ (defined in Eq. (4)) for wine, running and pizza attributes? Which attribute would you use to split the data at the root if you were to maximize the gain $G$ using the Gini index metric defined above?

## 2.1 What to submit

- Values of Gini index $G$ for wine, running and pizza attributes.
- The attribute you would use for splitting the data at the root.