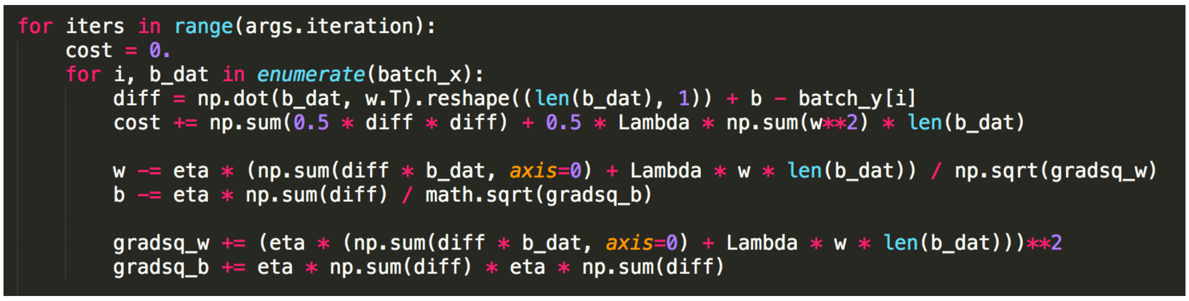
# Machine Learning 2016 report

陳翰浩 資工所碩一 R05922021
r05922021_HanHao

## 1. Linear Regression

$$C = \tfrac{1}{2}\sum_{x,y}(w^T x + b - y)^2 + \lambda \sum (w)^2$$

Define a basic linear regression cost function with regularization term and implement min-batch gradient decent with adagrad or momentum to accelerate training process.



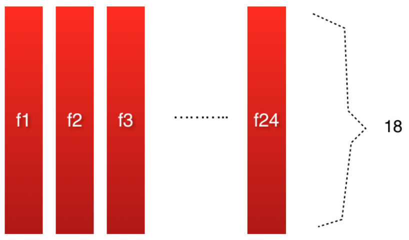
```python
for iters in range(args.iteration):
    cost = 0.
    for i, b_dat in enumerate(batch_x):
        diff = np.dot(b_dat, w.T).reshape((len(b_dat), 1)) + b - batch_y[i]
        cost += np.sum(0.5 * diff * diff) + 0.5 * Lambda * np.sum(w**2) * len(b_dat)

        w -= eta * (np.sum(diff * b_dat, axis=0) + Lambda * w * len(b_dat)) / np.sqrt(gradsq_w)
        b -= eta * np.sum(diff) / math.sqrt(gradsq_b)

        gradsq_w += (eta * (np.sum(diff * b_dat, axis=0) + Lambda * w * len(b_dat)))**2
        gradsq_b += eta * np.sum(diff) * eta * np.sum(diff)
```

⇧sample code

## 2. Training feature

For raw train.csv file, there are 18*24 features in each day. Since, the test.csv need us to use 9 hours feature to predict the 10th hour's PM2.5, the naive way to extract the training pair is concat first 9 hour features and use the 10th hour's PM2.5 to be its label. By using this feature extraction, we extract feature from train.csv continually. We will get many 162 dimensions feature with one label pairs (x, y)
**note that** I use 0. to replace non-numeric feature "NR" in train.csv and test.csv.

here I add element-wise multiple pair of  last hours' features (9th) , so the feature dimension will be 333.



## 3. Training tips

I have used following training tips in the implementation to promote the performance or accelerate the training speed.

- Adagrad

  By using adagrad we can use first derivative to estimate second derivative, dynamically tuning learning rate. Therefore, adagrad can accelerate reaching lower cost.

- momentum

Momentum will update the w by adding previous stage's gradient orientation. This can accelerate the training parse to reach the minimum cost. Moreover, momentum can breakthrough the saddle point sometime. (kaggle best use this method)

**update :**

$v = \lambda v' - gw$

$w += v$

here I use lambda = 0.8

- min-batch

  Instead of updating weight by each training pair, min-batch update the weight after seeing one batch size training pair. This can speed up the training greatly.
  here I use batch size = 100

---

## 4. Discussion on regularization and learning rate

I have ran different value of lambda for regularization experiment, and choose the lowest validation error's lambda to test different learning rate. (validation = 1/10 training data, iteration = 10000)

| lambda with leaning rate = 5e-08 | Ein | validation | Eout |
|---|---|---|---|
| 0 | 5.66895721 | 6.08750895 | 5.68389 |
| 0.01 | 5.66373723 | 5.87784274 | 5.65408 |
| 0.1 | 5.73604453 | 5.40544896 | 5.71525 |
| 1 | 5.76924327 | 5.9129623 | 5.81771 |
| 10 | 5.99296468 | 6.12578817 | 6.23277 |

Theoretically, larger lambda will be harder to gain lower Ein in training parse, and Eout result will be more close to Ein result. However, in this experiment we have contrary result. One possible explanation is that our model or data is not complex enough to fit the real model(model bias too big), so higher lambda value cause a constrain to fit the real distribution.

| learning rate with $\lambda$=0.1 | first iteration Ein | final Ein | first iteration Val | final Val | Eout |
|---|---|---|---|---|---|
| 5e-06 | 61373.7836206 | 1270.14720687 | 58492.8776499 | 1253.31481486 | 1318.01892 |
| 5e-07 | 26576.2102119 | 32.9930097 | 26669.4367021 | 29.19723164 | 32.32163 |
| 5e-08 | 24.53904351 | 5.7445634 | 24.25423968 | 5.2678001 | 5.69482 |
| 5e-09 | 14.57505361 | 5.74998903 | 13.61784249 | 5.59069474 | 5.80435 |
| 5e-10 | 16.76859625 | 6.35612318 | 15.23907208 | 6.33090286 | 6.61520 |

It is important to fine tune the initial learning rate (eta) to gain a better result. Too small eta will take much more iterations to reach lower cost, and too large eta can't even reach lower cost after many iterations.