Machine Learning Undergrad Course@SUTD
Assignment 2

Linear Regression, Ridge Regression

Due on Wednesday, September 23, 2015

1. upload all files for all tasks in one zip-(or .tar.gz)-file. for the images provide a description which image is for what, e.g. give the descriptions in one file which contains all other task results (equations, textual explanations) and descriptions for all images. best is one long pdf (or word) file with embedded images, equations, textual explanations.

2. function data_generator1: it should provide N pairs (x, y) of features x with labels y with N being a parameter of the data generation function according to the format defined below:

   (a) The feature is a 48-dimensional vector with shape $(48, 1)$, the label a real number.

   (b) return values of the function: the features in a numpy-matrix with shape $(48, N)$, the labels in a numpy matrix with shape $(N, 1)$, use *numpy.reshape* if necessary, check outputs of variable z with *print z.shape*

   ```
   def data_generator1(N):
   ... wild coding here ...
   return x,y
   ```

   (c) the feature is drawn from a 48-dimensional distribution with mean being the zero vector, and covariance matrix Σ being a diagonal matrix such that the diagonal entry for dimension d is $1.0/(d + 1.0)$, that means

$$\Sigma = \begin{pmatrix} \frac{1.0}{0+1.0} & 0 & 0 \dots 0 & 0 \\ 0 & \frac{1.0}{1+1.0} & 0 \dots 0 & 0 \\ 0 & 0 & \frac{1.0}{2+1.0} \dots 0 & 0 \\ \vdots & \vdots & \vdots \dots \vdots & \vdots \\ 0 & 0 & 0 \dots 0 & \frac{1.0}{48-1+1.0} \end{pmatrix}$$

   (d) check whether you can find a way to create this feature from 48 one-dimensional draws from a normal in one dimension

   (e) for a given feature x the label is given as $y = v^\top x + \epsilon$, where ϵ is a draw from a one-dimensional normal distribution with mean zero and standard deviation equal to 2, and v is a vector having only ones.

3. draw in a separate function a random vector v in 48 dimensions with the following contraints: for 12 of these dimensions the dimensions should have a random value in [0.6 1], for 36 of these dimensions the dimensions should have a random value in [0 0.2]

```
def gen_coefficients():

...

return v
```

4. function data_generator2: it should provide N pairs (x, y) of features x with labels y with N being a parameter of the data generation function, and it should the model coefficients v as input

   (a) The feature is a 48-dimensional vector with shape $(48, 1)$, the label a real number.

   (b) return values of the function: the features in a numpy-matrix with shape $(48, N)$, the labels in a numpy matrix with shape $(N, 1)$, the vector v in a numpy-matrix with shape $(48, 1)$

   ```
   def data_generator2(N,v):
   ...
   return x,y,v
   ```

   (c) the feature is drawn from a 48-dimensional distribution with mean being the zero vector, and covariance matrix Σ being a diagonal matrix such that the diagonal entry is 1.0

   (d) check whether you can find a way to create this feature from 48 one-dimensional draws from a normal in one dimension

   (e) for a given feature x the label is given as $y = v^\top x + \epsilon$, where ϵ is a draw from a one-dimensional normal distribution with mean zero and standard deviation equal to 2, and v is a vector generated with the constraints ([0.6 1] in 12 dims,[0 0.2] in 36 dims) above

5. Ridge regression

   (a) deliver the whole code for this task in one .py file. it should be executable and runnable.

   (b) program ridge regression. see the regression recap slides for the algorithm. note that ridge regression has a lambda parameter for the regularization strength

   ```
   def train_ridgeregression(x,y,lambda):
   ...
   return w
   ```

6. ridge regression on data_generator1

   (a) run ridge regression in the following way:

   (b) repeat ten times: $(i = 1, \dots, 10)$:

Algorithm 1 ...

```
1: generate 100 training samples and 1000 testing samples using function data_generator1
2: set set $\lambda = 1e - 30$
3: train on the training samples, compute the MSE on the testing samples. denote this MSE as $m_{1,i}$
4: keep the data samples
5: set $\lambda = 5$
6: train on the training samples, compute the MSE on the testing samples. denote this MSE as $m_{2,i}$
```

   (c) for both choices of lambda separately: compute the average of $m_{1,i}$ for these 10 runs, compute the average of $m_{2,i}$ for these 10 runs

   (d) print and compare these averaged mse values. For which of the two lambdas you get a better result?

   (e) now repeat above procedure ten times with 500 training samples and 1000 testing samples.

   (f) compare the difference of MSE between $\lambda = 5$ and $\lambda = 1e - 30$ when running it with 100 training samples and with 500 training samples.

   (g) do you have a guess how the strength of regularization, as expressed by the size of λ, should be chosen, as you have more and more training data available?

7. cross-validation versus holdout on data_generator2

   run ridge regression in the following way:

   (a) repeat ten times ($i = 1, \ldots, 10$):

Algorithm 2 Example of a Holdout protocol for ridge regression (for $\lambda = 1e-2$)

```
1: generate 500 samples by data_generator2
2: set $\lambda = 1e-2$
3: select randomly 400 samples for training, 100 for testing
4: train on the training samples, compute the MSE on the testing samples. Let this MSE for the i-th repetition
   ($i = 1, \ldots, 10$) be $\mathbf{m_{1,i}}$.
```

   (b) compute the average of the MSE scores $\mathbf{m_{1,i}}, i = 1 \ldots, 10$ over the ten repetitions and the variance of the MSE over these ten repetitions. We denote this average as A_1 and this variance as V_1

   (c) Do a ten-fold repetition of the following protocol involving 5-fold crossvalidation:

Algorithm 3 Example of a 5-fold cross-validation protocol for ridge regression (for $\lambda = 1e-2$)

```
1: generate 500 samples by data_generator2
2: set $\lambda = 1e-2$
3: generate a random split of the samples into 5 non-overlapping sets, each with 100 samples (check numpy.random
   for what you can use for that)
4: for Iteration $t = 1, \ldots, 5$ do
5:     test data set is the $t$-th set of above split
6:     training data set are all remaining sets of above split
7:     train on the training samples, compute the MSE on the testing samples. Let this MSE for $t$-th set part of
       crossvalidation be $e_t$
8: end for
9: compute the average MSE of the crossvalidation procedure from the $e_t$: $e = \frac{1}{5}\sum_{t=1}^{5} e_t$
```

   (d) We denote the average MSE of the crossvalidation procedure for the i-th repetition as $\mathbf{m_{2,i}}$

   (e) compute the average of the MSE scores $\mathbf{m_{2,i}}, i = 1 \ldots, 10$ over the ten repetitions and the variance of the MSE scores $m_{2,i}, i = 1 \ldots, 10$ over these ten repetitions. We denote this average as A_2 and this variance as V_2

   (f) print and compare the variance of the MSE for the averaged holdout V_1 versus the averaged crossvalidation V_2. For which protocol the variance is smaller? Which protocol gives a better estimate of the MSE?