Comp 6321 - Machine Learning - Assignment 2

Federico O'Reilly Regueiro

October 18th, 2016

Question 1:

1.a Partition the data into training / testing, 90 to 10 and perform and plot $L2$ regularization

The data was partitioned pseudo-randomly[1].

```
phi = load('hw2x.dat');
phi = [phi, ones(size(phi,1),1)];
y = load('hw2y.dat');

% Partition the data randomly.
idxs = randperm(size(phi, 1));
idx_train = idxs(1:89);
idx_test = idxs(90:99);

phi_train = phi(idx_train, :);
y_train = y(idx_train);
phi_test = phi(idx_test, :);
y_test = y(idx_test);
```

Next, a range of lambdas was chosen, going from 0 to almost 125000 in order to get a good sense of the trend of both the error and the coefficients. The former was plotted for a range of small values as well as along the whole set of lambdas for which the RMS error was calculated. Two plots were done in order to allow us to see the behaviour of the test and training errors for lower values of λ as well as the overall trend of the RMS the resulting plot can be seen in Figure 1.

```
lambdas = 0:0.1:50;
lambdas = lambdas .^ 3;
```

[1] I have included the permutation indexes yielded by matlab for the instance of the 90 / 10 partition from which the plots and values were drawn. Suffice it to uncomment two lines and comment-out two others in order to partition the data randomly, yet similar results (at different scales of λ) can be observed.

```
w = zeros(length(lambdas), size(phi,2));

for lambda = lambdas
  idx = lambda == lambdas;

  % Train the model
  w(idx, :) = pinv(phi_train' * phi_train ...
                  + (lambda * eye(size(phi_train, 2)))) ...
                  * (phi_train' * y_train);

  h_phi_train(:, idx) = phi_train * w(idx, :)';
  j_h_train(idx) = rms(h_phi_train(:, idx) - y_train);

  % Now compare to the test
  h_phi_test(:, idx) = phi_test * w(idx, :)';
  j_h_test(idx) = rms(h_phi_test(:, idx) - y_test);
end
```

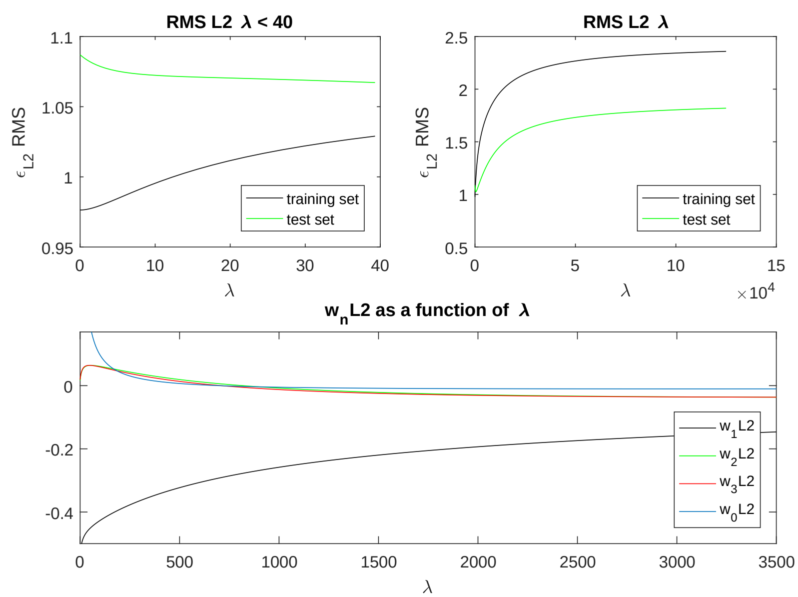

Figure 1: Plot of the RMS training and testing error as well as the coefficients over a wide range of λ.

On the top-left plot of figure 1 can observe that as expected, the test error goes down as λ grows. Howerver, as λ continues to increase, the test error closely follows the training error as they both grow since the restrictions on the coefficients make for a worse fit at a certain point.

1.b Use the `quadprog` function in Matlab for $L1$ regularization

In order to use `quadprog` for regularization, we must first find the Hessian matrix `H` as well as other parameters `f`, `A`, `b` in the specific format that Matlab requires.

We recall that for $L1$ regularization the expression we must minimize is:

$$\arg\min_{w} \frac{1}{2}(\boldsymbol{\Phi w} - \boldsymbol{y})^T(\boldsymbol{\Phi w} - \boldsymbol{y}) + \frac{\lambda}{2}\sum_{k=0}^{K-1}|w_k| \tag{1}$$

Which is equivalent to finding:

$$\begin{gathered}\arg\min_{w}(\boldsymbol{\Phi w} - \boldsymbol{y})^T(\boldsymbol{\Phi w} - \boldsymbol{y}) \\ \sum_{k=0}^{K-1}|w_k| \leq \eta\end{gathered} \tag{2}$$

And expands to:

$$\begin{gathered}\arg\min_{w}\boldsymbol{w}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi w} - 2y^T\boldsymbol{\Phi w} \\ \sum_{k=0}^{K-1}|w_k| \leq \eta\end{gathered} \tag{3}$$

And for which we can remove the constant term $\boldsymbol{y}^T\boldsymbol{y}$, yielding:

$$\begin{gathered}\arg\min_{w}\boldsymbol{w}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi w} - \boldsymbol{2y}^T\boldsymbol{\Phi w}^T \\ \sum_{k=0}^{K-1}|w_k| \leq \eta\end{gathered} \tag{4}$$

Matlab's `quadprog(H, f, A, b)` function, gives the optimal $\boldsymbol{x}$ corresponding to the expression $\arg\min_{x} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Hx} + \boldsymbol{f}^T\boldsymbol{x}$, subject to constraints $\boldsymbol{Ax} \leq \boldsymbol{b}$. We can thus take `H :=` $2\boldsymbol{\Phi}^T\boldsymbol{\Phi}$, then `f :=` $-2\boldsymbol{y}^T\boldsymbol{\Phi}$, `A :=`$\boldsymbol{P}$, where for a system with n variables, $\eta\boldsymbol{P}$ is the matrix with 2^n permutations of $[b_1, b_2, \ldots b_n], b \in \{-1, 1\}$ and lastly `b :=` $c\ \vec{\mathbf{1}}$, where $\vec{\mathbf{1}}$ is an all-one vector of length 2^n that places an upper bound, η, to the expression $\sum_{k=0}^{K-1}|w_k|$, such that $\sum_{k=0}^{K-1}|w_k| \leq \eta$. We note that η is roughly equivalent to $\frac{1}{\lambda}$ which we use so that we may compare the effects of $L1$ and $L2$ regularizations on a similar range of values.

Thus we end up with the following code:

```
etas = 1./lambdas;
for eta = etas
  idx = eta == etas;
  w_quad(idx,:) = quadprog(2*(phi_train'*phi_train), ...
                  -2*(phi_train'*y_train), ...
                     [ 1, 1, 1, 1;    1, 1,-1, 1; ...
                       1,-1, 1, 1;    1,-1,-1, 1; ...
```

```
                        -1, 1, 1, 1;   -1, 1,-1, 1; ...
                         1,-1, 1, 1;   -1,-1,-1, 1; ...
                         1, 1, 1, -1;   1, 1,-1, -1; ...
                         1,-1, 1, -1;   1,-1,-1, -1; ...
                        -1, 1, 1, -1;  -1, 1,-1, -1; ...
                         1,-1, 1, -1;  -1,-1,-1, -1], ...
                    eta * [1; 1; 1; 1; 1; 1; 1; 1]);

  h_phi_train_quad(:, idx) = phi_train * w_quad(idx, :)';
  j_h_train_quad(idx) = rms(h_phi_train_quad(:, idx) - y_train);

  % Now validate
  h_phi_test_quad(:, idx) = phi_test * w_quad(idx, :)';
  j_h_test_quad(idx) = rms(h_phi_test_quad(:, idx) - y_test);
end
```

1.c Plot $L1$ RMS and coefficients, w against λ and comment

Although it is not exactly equivalent, we shall simplify the comparison of $L1$ and $L2$ throughout this section by using $\lambda \approx \frac{1}{\eta}$ as it gives a clearer idea. In Figure 2, we can again notice how the test error slightly decreases for the lowest values of $\approx \lambda$ and then monotonically increases as the coefficients are all forced towards 0. We also note that $L1$ yields a lower minimum error (1.0098) than $L2$ regularization (1.0257).

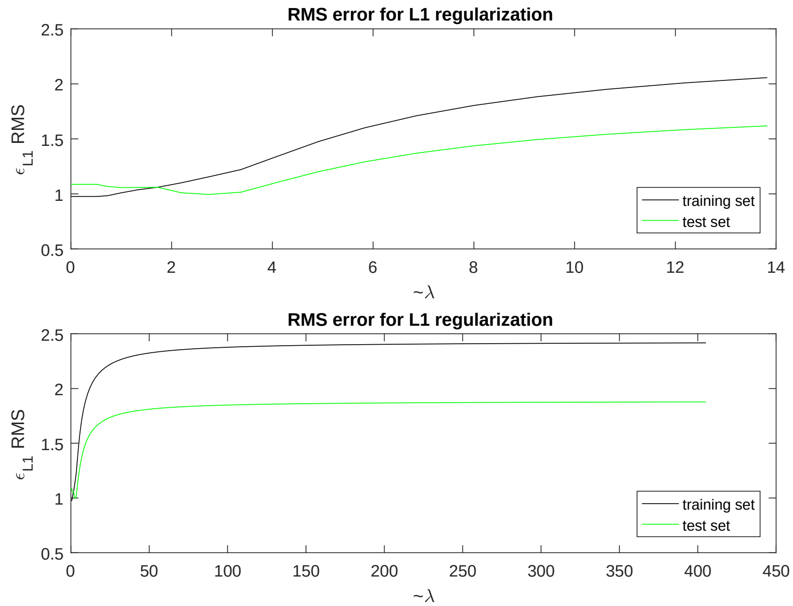

Figure 2: The RMS training and testing error for a wide range of $\frac{1}{\eta} \approx \lambda$.

By the same token, in figure 3 we can see how both w_2 and w_3 sharply decrease to 0 when $\approx \lambda = 2$ and the model relies solely on w_1[2] which then decreases gradually, as opposed to figure 1, where we can observe how all coefficients approach 0 at a similar rate during $L2$ regularization. Conversely, as is expected we can see that both errors and coefficients are equal between $L1$ and $L2$ regularization when $\lambda = 0$.

This particular data-set would lead to believe that the data was generated mainly by some function $f(w_1)+\epsilon$. This hypothesis is supported by the following output:

```
corr(y, phi(:,1:3))
ans =
   -0.848189   -0.017339   -0.024943
```

which reveals that the correlation between $\Phi_{:,1}$ and y is much larger than between other columns of Φ and y.

---

[2] w_4 is the bias term, so we can't really say the model relies on it a it is not an input.

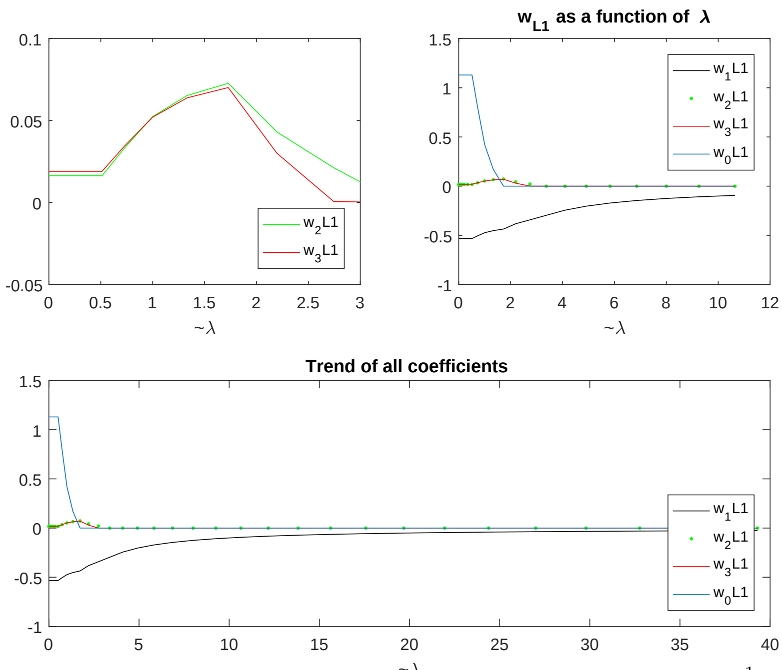

Figure 3: The RMS training and testing error for a wide range of $\frac{1}{\eta} \approx \lambda$.

Question 2: Dealing with missing data, fill in $x_{i,n}$ with class-conditional means?

First, we write $\mu_{c,i}$ to represent $E(x_i|y = c)$ and we assume independence between features. Then, since our classifier is Gaussian, we know that $P(x|y = 1)$ and $P(x|y = 0)$ are modelled as follows:

$$P(x|y = c) = \frac{1}{\sqrt{2\pi|\mathbf{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_c})^T \mathbf{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_c}), \quad c\in\{0,1\}} \tag{5}$$

We turn our attention to the numerator of the exponent, which can also be written in the following manner:

$$\sum_{i=1}^{n} [(x_i - \mu_{c,i} \sum_{j=1}^{n} (\Sigma^{-1}{}_{i,j}(x_j - \mu_{c,j})] \tag{6}$$

For the value of a given feature n, replaced by its class-conditional mean, $\mu_{c,n}$, this expression becomes 0; we further develop this by analyzing the log-odds:

$$log \frac{P(y = 1|x)}{P(y = 0|x)} = log \frac{P(y = 1)}{P(y = 0)} + log \frac{\frac{1}{\sqrt{2\pi|\mathbf{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_1})^T \mathbf{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_1})}}{\frac{1}{\sqrt{2\pi|\mathbf{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_0})^T \mathbf{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_0})}} \tag{7}$$

Since the matrix sigma is shared, we can write:

$$log\frac{P(y=1|x)}{P(y=0|x)} = log\frac{P(y=1)}{P(y=0)} + log\frac{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_1})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_1})}{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_0})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_0})} \tag{8}$$

Then we expand the exponent:

$$log\frac{P(y=1|x)}{P(y=0|x)} = log\frac{P(y=1)}{P(y=0)} + log\frac{-\frac{1}{2}\sum_{i=1}^{n}[(x_i-\mu_{c,i}\sum_{j=1}^{n}(\Sigma^{-1}{}_{i,j}(x_j-\mu_{c,j})])}{-\frac{1}{2}\sum_{i=1}^{n}[(x_i-\mu_{c,i}\sum_{j=1}^{n}(\Sigma^{-1}{}_{i,j}(x_j-\mu_{c,j})])} \tag{9}$$

Where we can clearly see that the contribution of x_n does not change the ratio of the log-odds given by all other features, since we have choosen $x_n = \mu_{c,n}$ for $c \in \{0, 1\}$ where $\mu_{c,n}$ is the class-conditional means for class c

Question 3: Naive Bayes assumption, suppose a feature gets repeated in the model