HW3 Solutions

Gabe Hope
February 2017

1  Problem 1

1.1  Part a (and b)

Our problem is to find the MAP estimate of $w$:

$$w_{MAP} = \max_w p(w|T)$$

Using Bayes’ rule and the fact that $p(T)$ does not depend on $w$ we see that:

$$w_{MAP} = \max_w \frac{p(T|w)p(w)}{p(T)} = \max_w p(T|w)p(w)$$

$$= \min_w -\log p(T|w) - \log p(w)$$

The negative log likelihood of the training set ($p(T|w)$) is the same as in standard linear regression:

$$-\log p(T|w) = \sum_{i=1}^{N} \left(\frac{y_i - f_w(x_i))^2}{2\sigma^2}\right) - \log \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)$$

The negative log prior is simply the negative log pdf of a multivariate Gaussian distribution with mean 0 and covariance $aI$:

$$-\log p(w) = \frac{1}{2}w^T(aI)^{-1}w - \log \left(\frac{1}{\sqrt{(2\pi)^d|\Sigma|}}\right)$$

Putting these two pieces together and dropping the terms that do not depend on $w$ we get:

$$w_{MAP} = \min_w \frac{1}{2a}w^Tw + \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f_w(x_i))^2$$

Note that above we made use of the fact that:

$$\frac{1}{aI} = \frac{1}{\frac{1}{a}I}$$
Finally multiplying through by the constant $\sigma^2$ we get:

$$w_{MAP} = \min_w \frac{\sigma^2}{2a} w^T w + \frac{1}{2} \sum_{i=1}^{N} (y_i - f_w(x_i))^2$$

This is equivalent to our desired form with $\lambda = \frac{\sigma^2}{a}$.

**2 Problem 2**

Using the fact that the $n$ random variables in our problem are independent, we can write the distribution over random vectors as a product of Bernoulli distributions:

$$p(x|u) = \prod_{i=1}^{n} u_i^{x_{ji}} (1 - u_i)^{(1-x_{ji})}$$

Since the likelihood over the training set is the product of the likelihoods of all the samples, we can formulate the maximum likelihood estimator for $u$ as:

$$u_{ML} = \max_u p(T|u) = \max_u \prod_{j=1}^{k} \prod_{i=1}^{n} u_i^{x_{ji}} (1 - u_i)^{(1-x_{ji})}$$

Where we are using $x_{ji}$ to refer to the $i^{th}$ entry of sample $j$. Taking the log of the likelihood we get:

$$u_{ML} = \max_u \sum_{j=1}^{k} \sum_{i=1}^{n} (x_{ji}) \log(u_i) + (1 - x_{ji}) \log(1 - u_i)$$

Taking the derivative with respect to $u_i$ and setting it equal to zero gives us the following as in our original derivation of the Bernoulli MLE:

$$\frac{\partial}{\partial u_i} = 0 \rightarrow \sum_{j=1}^{k} \left( \frac{x_{ji}}{u_i} + \frac{1 - x_{ji}}{1 - u_i} (-1) \right) = 0$$

Solving for $u_i$ we get:

$$u_{iML} = \frac{\sum_{j=1}^{k} x_{ji}}{k}$$

Note that this is equivalent to using the Bernoulli ML estimator for each dimension independently.
3 Question 3

3.1 Part a

For our naive Bayes model, we are interested in the maximum likelihood estimate of $u$ using our training data $T$:

$$u_{ML} = \max_u p(T|u)$$

$$= \max_u \log p(T|u)$$

We can break up our likelihood $p(T|u)$ using the assumption that the samples in our training set are independent:

$$u_{ML} = \max_u \sum_{j=1}^{k} \log p(x_j, y_j|u) = \max_u \sum_{j=1}^{k} (\log p(x_j|y_j, u) + \log p(y_j|u))$$

$$= \max_u \sum_{j=1}^{k} \log p(x_j|y_j, u)$$

Above we used the fact that $p(y_j|u) = p(y_j)$, noting that the class probabilities are not dependent on the parameter $u$.

We can now use our modelling assumptions that the pixel probabilities are independent conditioned on the class and each pixel is drawn from a Bernoulli distribution specific to the class and model. This means that we can further expand our likelihood as follows:

$$u_{ML} = \max_u \sum_{j=1}^{k} \sum_{i=1}^{n} \log p(x_{ji}|y_j, u)$$

$$= \max_u \sum_{j=1}^{k} \sum_{i=1}^{n} (x_{ji}) \log(u_{yi,i}) + (1 - x_{ji}) \log(1 - u_{yi,i})$$

Taking the derivative with respect to the parameter for a particular class $c$ and feature $i$ we get:

$$\frac{\partial}{\partial u_{ji}} = 0 \rightarrow \sum_{y_j=c} \left( \frac{x_{ji}}{u_{ci}} + \frac{1 - x_{ji}}{1 - u_{ci}} (-1) \right) = 0$$

Where our sum is over all samples in $T$ with class label $c$. As in the Bernoulli MLE example, it follows that the MLE for a parameter $u_{ci}$ is simply a ratio of the count of 1s at pixel $i$ from samples with class $c$ divided by the total number of samples with class $c$:

$$u_{ci} = \frac{\sum_{y_j=c} x_{ji}}{\sum_{y_j=c} 1}$$
3.2 Part b
Visualization of MLE:

![Images of digits](image1.png)

3.3 Part c
Fraction of digits correctly classified: **0.7956**
Confusion matrix:

```
  437  0  3  0  2  32  13  0  12  1
  0  471  2  3  0  13  4  0  7  0
  8  12 379  36  8  2  6 12  32  5
  1  10  4 415  6 23  5 13 10  13
  1  1  0  1 364  5 13  4  7  94
14  2  3  67 19 346  7  5 17  20
  7  9  8  0  6  33 414  0  3  0
  1 18  8  4 14  0  2 396  9  48
  4 16  3 50 12 18  1 4 347  35
  2  8  4  8 50  7  0  7  5 409
```

3.4 Code

```matlab
load('digits.mat')
```
% Setup our collections of data in cell arrays, so we can loop through them later
alltrain = {train0, train1, train2, train3, train4, train5, train6, train7, train8, train9};
alltest = {test0, test1, test2, test3, test4, test5, test6, test7, test8, test9};

figure
u = cell(10);

% For each class compute the MLE of the parameter u and display the result as a heatmap
for digit = 1:10
    T = alltrain(digit);
    u{digit} = sum(T, 1) / size(T, 1);
    subplot(2, 5, digit);
    imagesc(reshape(u{digit}, 28, 28)');
    axis off
end

% Classify our test data and build a confusion matrix
confusion = zeros(10,10);
for truelabel = 1:10
    T = alltest(truelabel);
    log_probs = zeros(10, 500);
    % Compute the log-likelihoods of the images given each class
    for model = 1:10
        log_probs(model, :) = log(u{model} + 1e-20) * T' + log(1 - u{model} + 1e-20) * (1-T)';
    end
    % Make predictions based on the most probable class
    [maxp, predictions] = max(log_probs, [], 1);
    counts = zeros(10, 500);
    for digit = 1:10
        counts(digit, :) = predictions == digit;
    end
    % Fill the appropriate row of the confusion matrix
    confusion(truelabel, :) = sum(counts, 2);
end
%Print the accuracy and confusion matrix
accuracy = sum(diag(confusion)) / sum(sum(confusion))
confusion